GTM-IP

Generic Timer Module

GTM-IP Specification

Revision: 3.1.5.1
(Released on 24.03.2016)

Robert Bosch GmbH
Automotive Electronics (AE)

Revision Number Notice

The specification revision number of this document consists of four decimal integers separated by a dot.

The first decimal integer represents the major release number, the second represents the minor release number and the third represents the delivery number of specification.

A GTM-IP release always refers to the first three decimal integer of the specification revision number and is extended by a design step identifier. This GTM-IP release number can be read out of register GTM_REV.

The fourth decimal integer of specification revision number is related to updated versions of the specification which are independent of a GTM-IP release.

During silicon validation and qualification process it may turn out that the current specification revision related to the silicon is incomplete, inconsistent or ambiguous. It may also be the case that the silicon behaviour diverges for a specific functional feature from specification but the behaviour of silicon is also acceptable for intended GTM applications. In these cases Bosch AE will update the specification and indicate this update by an increase of the fourth decimal integer of specification revision number.

An increased fourth decimal integer means that either the new specification is more precise or that a functional feature was limited or removed. It never means that there was added a new feature.

Table of Contents

1 Introduction

1.1 Overview

This document is the specification for the Generic Timer Module (GTM). It contains a module framework with sub-modules of different functionality. These sub-modules can be combined in a configurable manner to form a complex timer module that serves different application domains and different classes within one application domain. Because of this scalability and configurability the timer is called generic.

The scalability and configurability is reached with an architecture philosophy where dedicated hardware sub-modules are located around a central routing unit (called Advanced Routing Unit (ARU)). The ARU can connect the sub-modules in a flexible manner. The connectivity is software programmable and can be configured during runtime.

Nevertheless, the GTM-IP is designed to unload the CPU or a peripheral core from a high interrupt load. Most of the tasks inside the GTM-IP can run -once setup by an external CPU- independent and in parallel to the software. There may be special situations, where the CPU has to take action but the goal of the GTM design was to reduce these situations to a minimum.

The hardware sub-modules have dedicated functionality's, e.g. there are timer input modules where incoming signals can be captured and characterized together with a notion of time. By combination of several sub-modules through the ARU complex functions can be established. E.g. the signals characterized at an input module can be routed to a signal processing unit where an intermediate value about the incoming signal frequency can be calculated.

The modules that help to implement such complex functions are called *infrastructure components* further on. These components are present in all GTM variants. However, the number of these components may vary from device to device.

Other sub-modules have a more general architecture and can fulfill typical timer functions, e.g. there are PWM generation units. The third classes of sub-modules are those fulfilling a dedicated functionality for a certain application domain, e.g. the DPLL serves engine management applications. A fourth group of sub-modules is responsible for supporting the implementation of safety functions to fulfill a defined safety level. The module ICM is responsible for interrupt services and defines the fifth group.

Each GTM-IP is build up therefore with sub-modules coming from those four groups. The application class is defined by the amount of components of those sub-modules integrated into the implemented GTM-IP.

1.2 Document Structure

The structure of this document is motivated out of the aforementioned sub-module classes. Chapter 2 describes the dedicated GTM-IP implementation this specification is written for. It gives an overview about the implemented sub-modules.

The following chapters 3 up to 10 deals with the so called infrastructure components for routing, clock management and common time base functions. Chapters 11 to 14 describe the signal input and output modules while the following chapter 15 explains the signal processing and generation sub-module with 16 its memory configuration. The next section chapters 17 to 19 provids a detailed description of application specific modules like the MAP, DPLL and SPE. The last section chapters 21 to 22 provides to safety related modules like CMP and MON sub-modules. Chapter 20 describes a module that bundles several interrupts coming from the other sub-modules and connect them to the outside world.

1.2.1 Sub-module groups

Chapter	Sub-module	Group
3	Advanced Routing Unit (ARU)	Infrastructure components
4	Broadcast Module (BRC)	Infrastructure components
5	First In First Out Module (FIFO)	Infrastructure components
6	AEI-to-FIFO Data Interface (AFD)	Infrastructure components
7	FIFO-to-ARU Interface (F2A)	Infrastructure components
8	Clock Management Unit (CMU)	Infrastructure components
9	Cluster Configuration Module (CCM)	Infrastructure components
10	Time Base Unit (TBU)	Infrastructure components
11	Timer Input Module (TIM)	IO Modules
12	Timer Output Module (TOM)	IO Modules
13	ARU-connected Timer Output Module (ATOM)	IO Modules
14	Dead Time Module (DTM)	IO Modules
15	Multi Channel Sequencer (MCS)	Signal generation and processing
16	Memory Configuration (MCFG)	Memory for signal generation and processing
17	TIM0 Input Mapping Module (MAP)	Dedicated
18	Digital PLL (DPLL)	Dedicated
19	Sensor Pattern Evaluation Module (SPE)	BLDC support
20	Interrupt Concentrator Module (ICM)	Interrupt services
21	Output Compare Unit (CMP)	Safety features
22	Monitoring Unit (MON)	Safety features

2 GTM Architecture

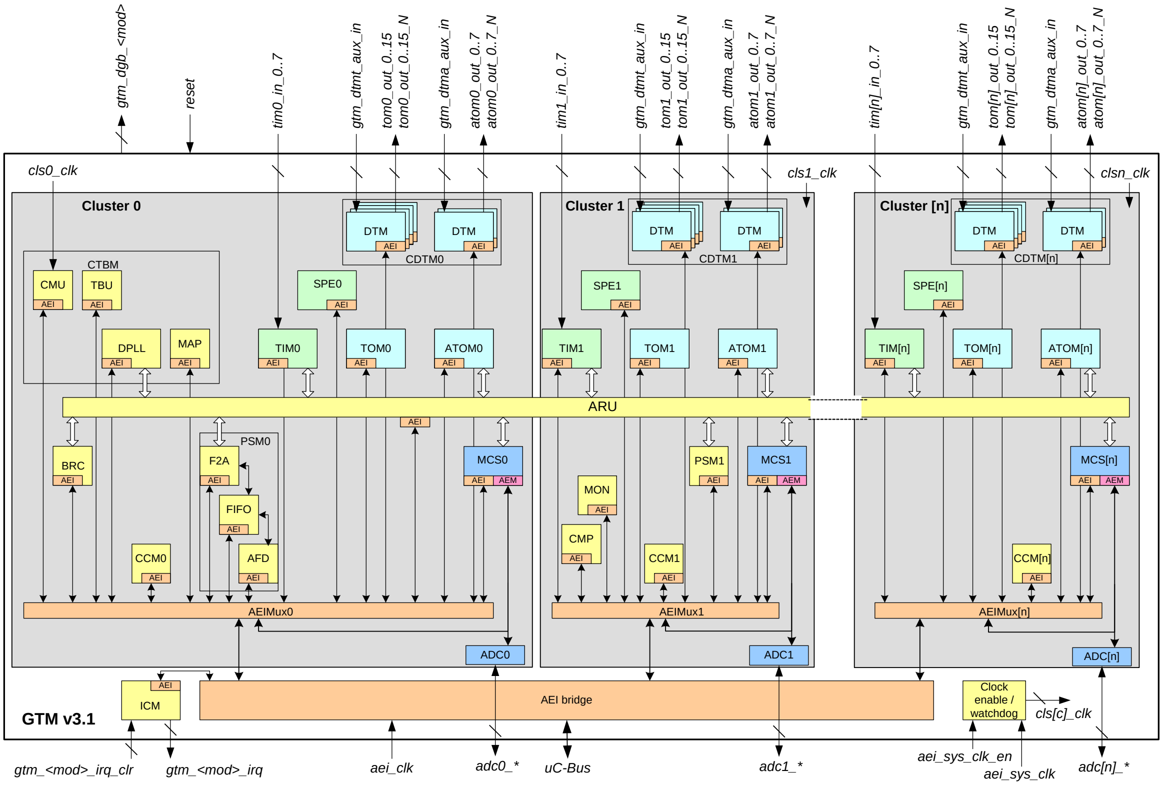
2.1 Overview

As already mentioned in Chapter 1 the GTM-IP forms a generic timer platform that serves different application domains and different classes within these application domains. Depending on these multiple requirements of application domains multiple device configurations with different number of sub-modules (i.e. ATOM, BRC, MCS, PSM, SPE, TIM, TOM, DTM) and different number of channel per sub-module (if applicable) are possible.
The device dependent configuration (i.e. the number of sub-modules) is listed in the device specific appendix B of this document [1].
The Parameter Storage Module (PSM) is only a virtual hierarchy and consists of the sub-modules F2A, FIFO and AFD.
The Cluster Dead Time Module (CDTM) is also a virtual hierarchy and consists of up to six DTM modules. It depends on the GTM device configuration which of the six DTM instances are available. Please refer to Appendix B for list of available DTM instances. In general, the first four DTM modules inside a CDTM[n] hierarchy are connected to the outputs of the TOM instance [n] of the cluster [n], the other two DTM instances are connected to the outputs of the ATOM instance [n] of this cluster [n].

The cluster view of a GTM-IP architecture is depicted in figure 2.1.1. This is a generic figure which shows an exemplarily GTM-IP device configuration.

The device dependent configuration (i.e. the count of sub-modules and channels per sub-module) is listed in the device specific appendix B of this document [1].

2.1.1 GTM Architecture Block Diagram



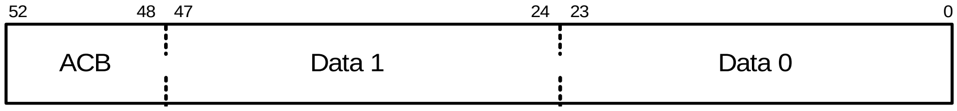
The GTM-IP is divided in multiple clusters 0...n. A certain amount of modules exist in each cluster. The operating frequency of a cluster can be configured to OFF, aei_sys_clk or aei_sys_clk/2. The clock enable generation can be implemented internal to the GTM_IP or external. In case of an external enable generation aei_sys_clk_en is used to generate the internal clocks. In addition an enable watchdog is implemented to monitor the correctness of the external applied enable signals aei_sys_clk_en.

The central component of the GTM-IP is the Advanced Routing Unit (ARU) where most of the sub-modules are located around and connected to. This ARU forms together with the Broadcast (BRC) and the Parameter Storage Module (PSM) the infrastructural part of the GTM. The ARU is able to route data from a connected source sub-module to a connected destination sub-module. The routing is done in a deterministic manner with a round-robin scheduling scheme of connected channels which receive data from ARU and with a worst case round-trip time.

The routed data word size of the ARU is 53 bit. The data word can logically be split into three parts. These parts are shown in figure 2.1.2. Bits 0 to 23 and bits 24 to 47 typically hold data for the operation registers of the GTM-IP. This can be, for example, the duty cycle and period duration of a measured PWM input signal or the output characteristic of an output PWM to be generated. Another possible content of Data0 and Data1 can be two 24 bit values of the GTM-IP time bases TBU_TS0, TBU_TS1 and TBU_TS2. Bits 48 to 52 can contain control bits to send control information from one sub-module to another. These ARU Control Bits (ACB) can have a different meaning for different sub-modules.

It is also possible to route data from a source to a destination and the destination can act later on as source for another destination. These routes through the GTM-IP are further on called *data streams*. For a detailed description of the ARU sub-module please refer to chapter 3.

2.1.2 ARU Data Word Description



The BRC is able to distribute data from one source module to more than one destination modules connected to the ARU. The PSM sub-module consists of three sub-units, the AEI-to-FIFO Data Interface (AFD), FIFO-to-ARU Interface (F2A) and the FIFO itself. The PSM can serve as a data storage for incoming data characteristics or as parameter storage for outgoing data. This data is stored in a RAM that is logically located inside the FIFO sub-unit, but physically the RAM is implemented and integrated by the silicon vendor with his RAM implementation technology. Therefore, the GTM-IP provides the interface to the RAM at its module boundary. The AFD sub-unit is the interface between the FIFO and the GTM SoC system bus interface AEI (please see section 2.2.1 for detailed discussion). The F2A sub-unit is the interface between the FIFO sub-unit and the ARU.

Signals are transferred into the GTM-IP at the Timer Input Modules (TIM). These modules are able to filter the input signals and annotate additional information. Each channel is for example able to measure pulse high or low times and the period of a PWM signal in parallel and route the values to ARU for further processing. The internal operation registers of the TIM sub-module are 24 bits wide.

The Clock Management Unit (CMU) serves up to 13 different clocks for the GTM and up to three external clock pins *GTM_ECLK0...2*. It acts as a clock divider for the system clock. The counters implemented inside other sub-modules are typically driven from this sub-module. Please note, that the CMU clocks are implemented as enable signals for the counters while the whole system runs with the GTM global clock *SYS_CLK*. This global clock typically corresponds to the micro controller bus clock the GTM-IP is connected to and should not exceed 100MHz because of the power dissipation of the used transistors where the GTM is implemented with.

The TBU provides up to three independent common time bases for the GTM-IP. In general, the number of time bases depends on the implemented device. If three time bases are implemented, two of these time bases can also be clocked with the digital PLL (DPLL) *sub_inc1c* and *sub_inc2c* outputs. The DPLL generates the higher frequent clock signals *sub_inc1*, *sub_inc2*, *sub_inc1c* and *sub_inc2c* on behalf of the frequencies of up to two input signals. These two input signals can be selected out of six incoming signals from the TIM0 sub-module. In this sub-module the incoming signals are filtered and transferred to the MAP sub-module where two of these six signals are selected for further processing inside the DPLL.

Signal outputs are generated with the Dead Time Module (DTM), Timer Output Modules (TOM) and the ARU-connected TOMs (ATOM). Each TOM channel is able to generate a PWM signal at its output. Because of the integrated shadow register even the generation of complex PWM outputs is possible with the TOM channels by serving the parameters with the CPU. It is possible to trigger TOM channels for a successor TOM sub-module through a trigger line between TOM(x)_CH(15) and TOM(x+1)_CH(0). But to avoid long trigger paths the GTM-IP integrator can configure after which TOM sub-module instance a register is placed into the trigger signal chain. Each register results in one SYS_CLK cycle delay of the trigger signal. Please refer to device specification of silicon vendor for unregistered trigger chain length.

In addition, each TOM sub-module can integrate functions to drive one BLDC engine. This BLDC support is established together with the TIM and Sensor Pattern Evaluation (SPE) sub-module.

The ATOMs offer the additional functionality to generate complex output signals without CPU interaction by serving these complex waveform characteristics by other sub-modules that are connected to the ARU like the PSM or Multi Channel Sequencer (MCS). While the internal operation and shadow registers of the TOM channels are 16 bit wide, the operation and shadow registers of the ATOM channels are 24 bit wide to have a higher resolution and to have the opportunity to compare against time base values coming from the TBU.

It is possible to trigger ATOM channels for a successor ATOM sub-module through a trigger line between ATOM(x)_CH(7) and ATOM(x+1)_CH(0). But to avoid long trigger paths the GTM-IP integrator can configure after which ATOM sub-module instance a register is placed into the trigger signal chain. Each register results in one SYS_CLK cycle delay of the trigger signal. Please refer to device specification of silicon vendor for unregistered trigger chain length.

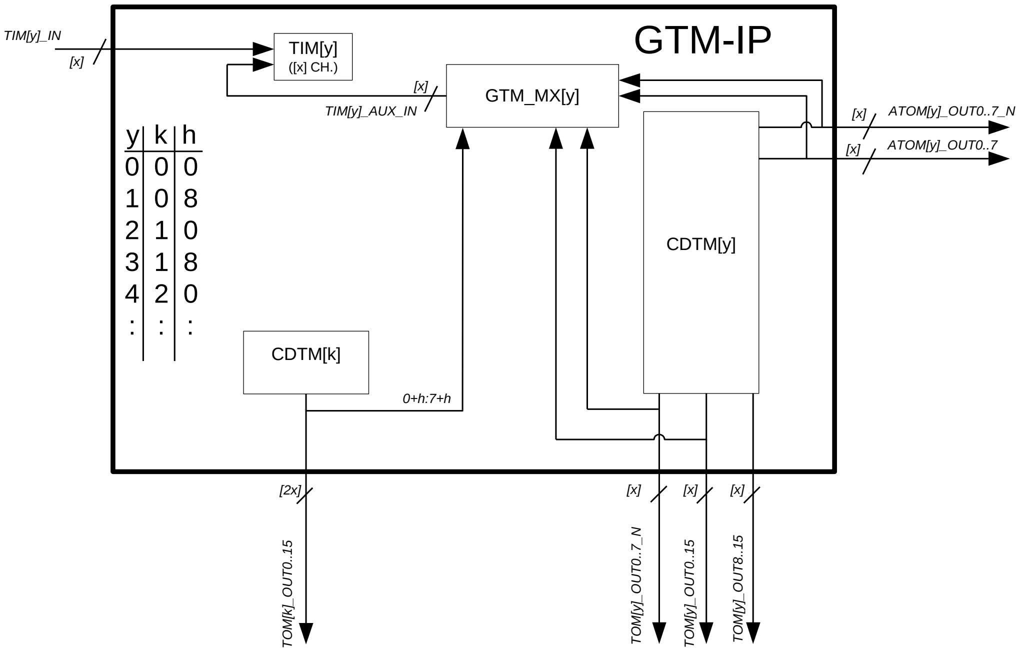
Together with the MCS the ATOM is able to generate an arbitrary predefined output sequence at the GTM-IP output pins. The output sequence is defined by instructions located in RAM connected to the MCS sub-module. The instructions define the points were an output signal should change or to react on other signal inputs. The output points can be one or two time stamps (or even angle stamp in case of an engine management system) provided by the TBU. Since the MCS is able to read data from the ARU it is also able to operate on incoming data routed from the TIM. Additionally, the MCS can process data that is located in its connected RAMs. The MCS RAM is located logically inside the MCS while the silicon vendor has to implement its own RAM technology there.

The two modules Compare Module (CMP) and Monitor Module (MON) implement safety related features. The CMP compares two output channels of the DTM and sends the result to the MON sub-module were the error is signaled to the CPU. The MON module is also able to monitor the ARU and CMU activities.

In the described implementation the sub-modules of the GTM-IP have a huge amount of different interrupt sources. These interrupt sources are grouped and concentrated by the Interrupt Concentrator Module (ICM) to form a much easier manageable bunch of interrupts that are visible outside of the GTM-IP.

On the GTM-IP top level there are some configurable signal connections from the signal output of the DTM modules to the input signals of the TIM modules.

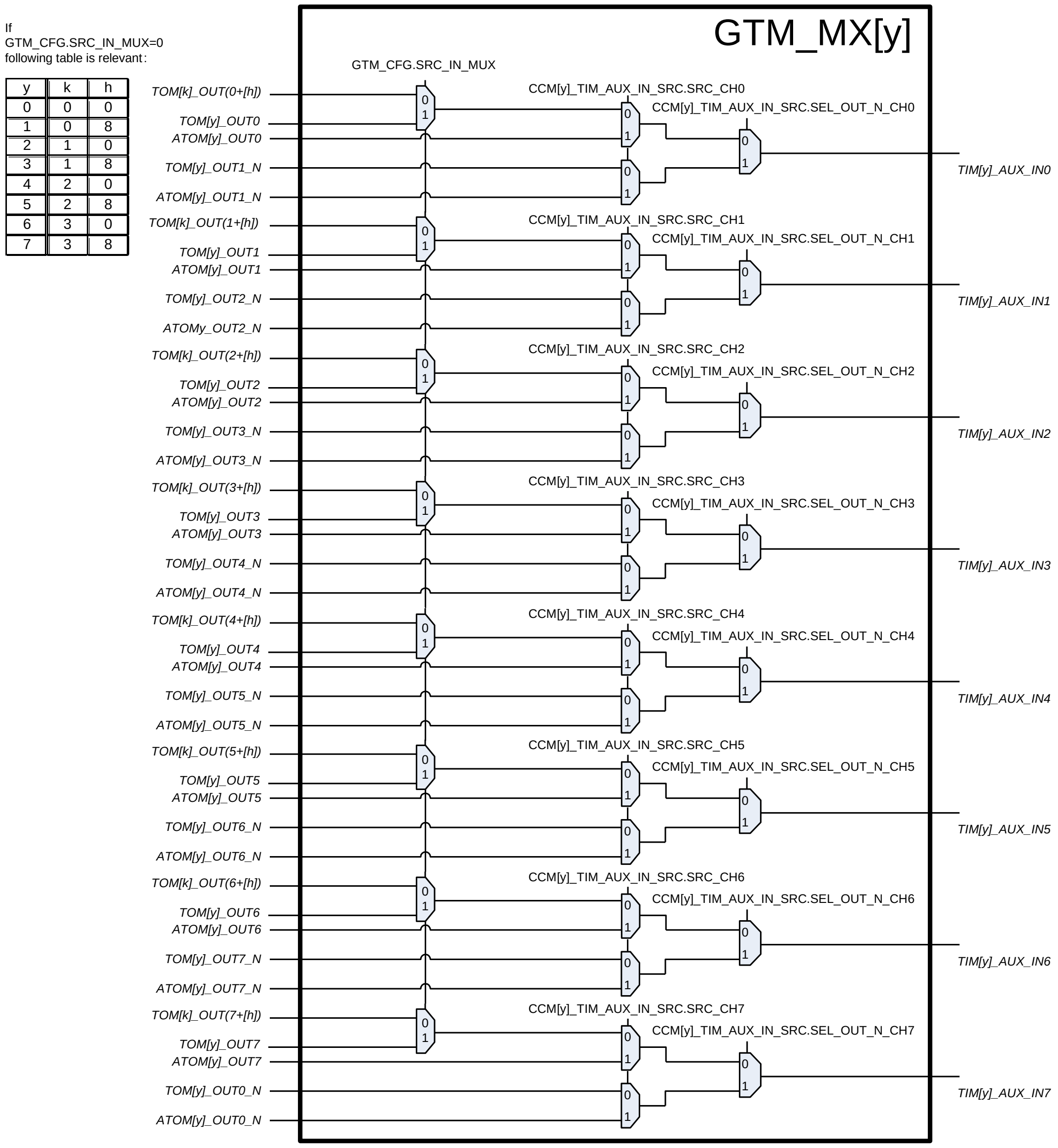
2.1.3 GTM-IP signal multiplex



y	k	h
0	0	0
1	0	8
2	1	0
3	1	8
4	2	0
:	:	:

The next diagram gives an overview of the connectivity for different configuration of GTM global bit **SRC_IN_MUX** of register **GTM_CFG** and the cluster configuration register **CCM[y]_TIM_AUX_IN_SRC**. The source selection is defined per channel with the bit **SRC_CH[x]** and **SEL_OUT_N_CH[x]** in the register **CCM[y]_TIM_AUX_IN_SRC**.

2.1.4 TIM auxiliary input multiplexing

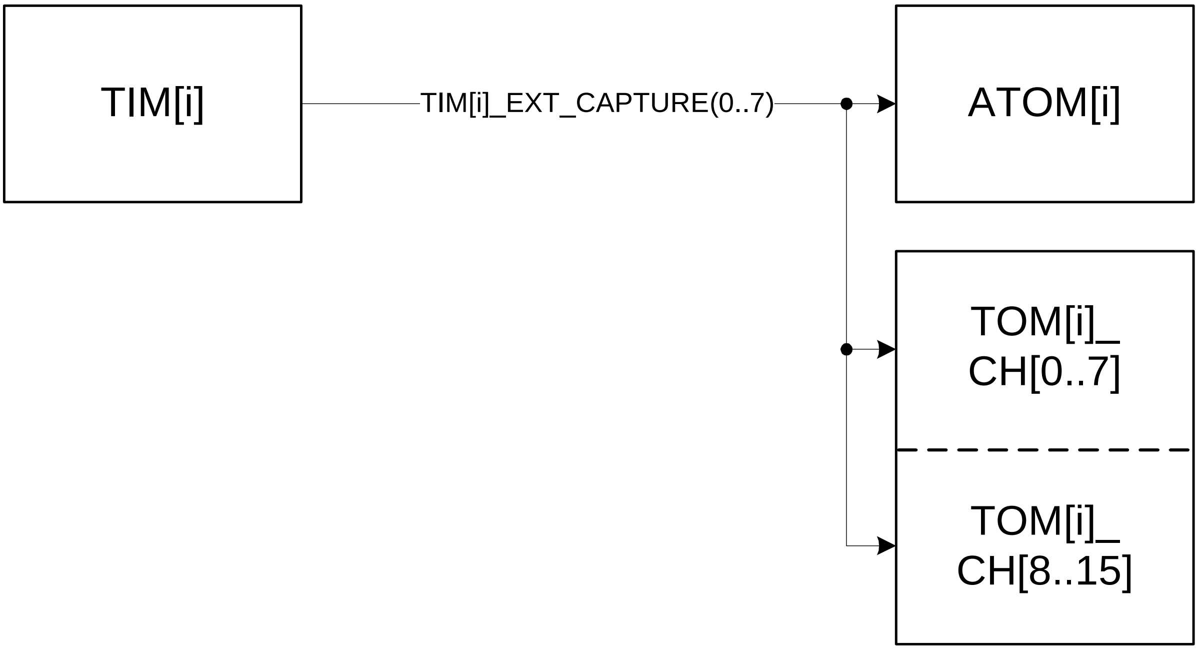


If
GTM_CFG.SRC_IN_MUX=0
following table is relevant:

y	k	h
0	0	0
1	0	8
2	1	0
3	1	8
4	2	0
5	2	8
6	3	0
7	3	8

The trigger out of TIM (i.e. the signals TIM[i]_EXT_CAPTURE(7:0) of each TIM instance i) are routed to ATOM instance [i] and TOM instance [i] with i=0...cITIM-1 (cITIM defines the number of available TIM instances, please refer to device specific Appendix B).
This TIM trigger can be used to trigger inside the ATOM or TOM instance either a channel or the global control register of AGC or TGC0/TGC1 unit.

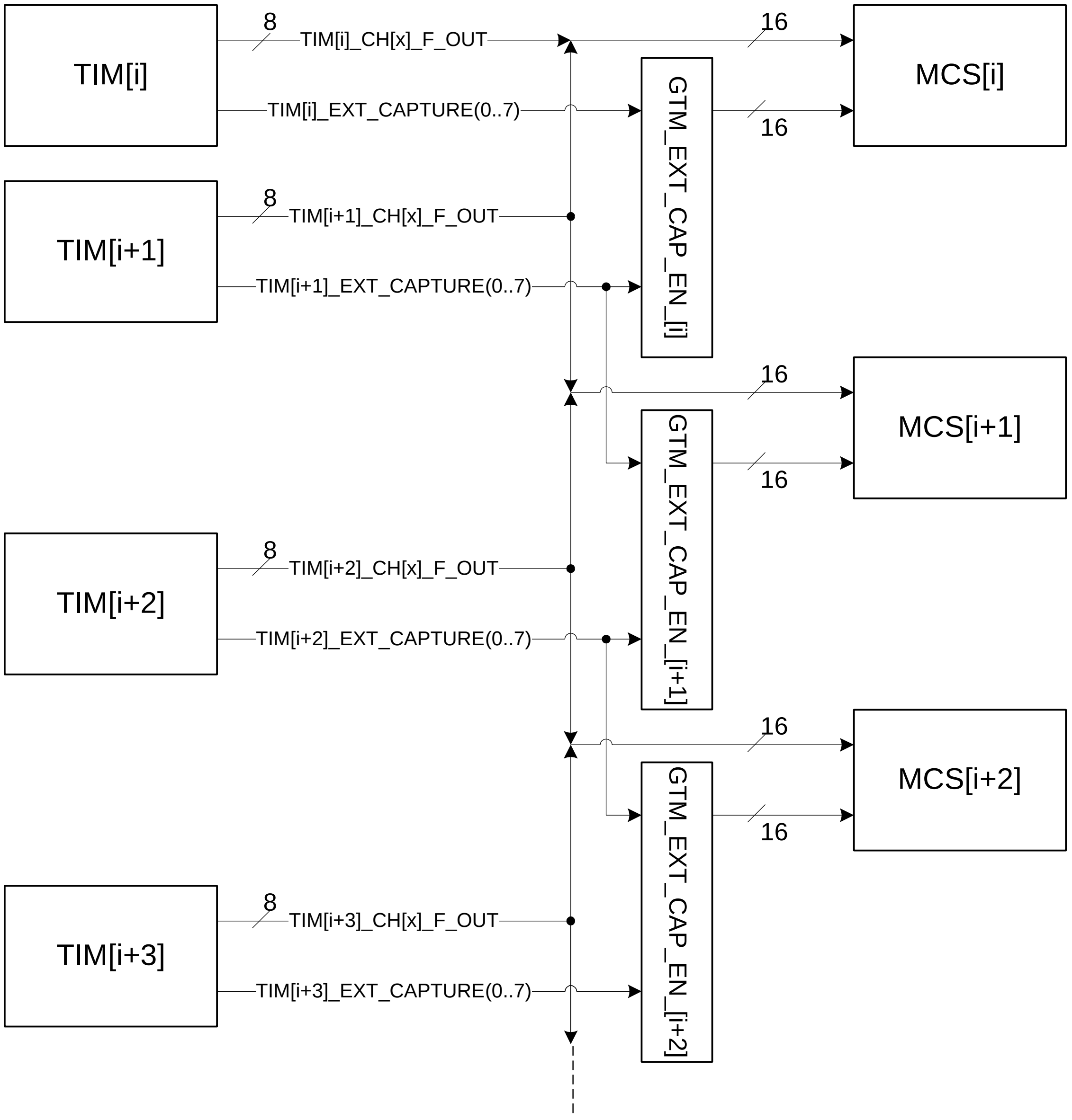
2.1.5 TIM external capture forwarding to TOM and ATOM



The trigger out of TIM (i.e. the signals TIM[i]_EXT_CAPTURE(7:0) of each TIM instance i) are additionally routed to the MCS instance [i]. This trigger forwarding can be enabled by register **CCM[i]_EXT_CAP_EN**.

2.1.6 TIM to MCS signal forwarding

x=0,..,cCTIM with cCTIM = number of available TIM channel of instance



2.2 GTM-IP Interfaces

In general the GTM-IP can be divided into four interface groups. Two interface groups represent the ports of the GTM-IP where incoming signals are assembled and outgoing signals are created. These interfaces are therefore connected to the GTM-IP input sub-module TIM and to the GTM-IP output sub-modules DTM.

Another interface is the bus interface where the GTM-IP can be connected to the SoC system bus. This generic bus interface is described in more detail in section 2.2.1. The last interface is the interrupt controller interface. The GTM-IP provides several interrupt lines coming from the various sub-modules. These interrupt lines are concentrated inside the ICM and have to be adapted to the dedicated micro controller environment where each interrupt handling can look different. The interrupt concept is described in more detail in section 2.5.

2.2.1 GTM-IP Generic Bus Interface (AEI)

The GTM-IP is equipped with a generic bus interface that can be widely adapted to different SoC bus systems. This generic bus interface is called AE-Interface (AEI). The adaptation of the AEI to SoC buses is typically done with a bridge module translating the AEI signals to the SoC bus signals of the silicon vendor. The AEI bus signals are depicted in the following table:

2.2.1.1 AEI bus signals

Signal name	I/O	Description	Bit width
AEI_SEL	I	GTM-IP select line	1
AEI_ADDR	I	GTM-IP address	32
AEI_PIPE	I	AEI Address phase signal	1
AEI_W1R0	I	Read/Write access	1
AEI_WDATA	I	Write data bus	32
AEI_RDATA	O	Read data bus	32
AEI_READY	O	Data ready signal	1
AEI_STATUS	O	AEI Access status	2

The AEI Status Signal may drive one of the following values:

2.2.1.2 AEI Status Signal

AEI_STATUS	Description
0b00	No Error
0b01	Illegal Byte Addressing
0b10	Illegal Address Access
0b11	Unsupported Address

The signal value 0b00 is returned if no error occurred during AEI access.
The signal value 0b01 is returned if the bus address is not an integer multiple of 4 (byte addressing).
The signal value 0b11 is returned if the address is not handled in the GTM.
The signal value 0b10 is returned

if the written register is a protected register (e.g. protected by bit RF_PROT) or
if the register is temporarily not writable because of sub-module internal state or the clock of the relevant cluster is disabled.

In case of an illegal write access signaled by status 0b10 the register will not be modified.
NOTE: Exception for register **CMU_CLK_CTRL**. In case of write access signalled by aei_status 0b10 the register will be modified each completely disabled bit.

Reading registers will never return status 0b10.

The detailed list of register addresses with return status 0b10 can be found in [1].

2.2.2 GTM-IP Multi-master and multitasking support

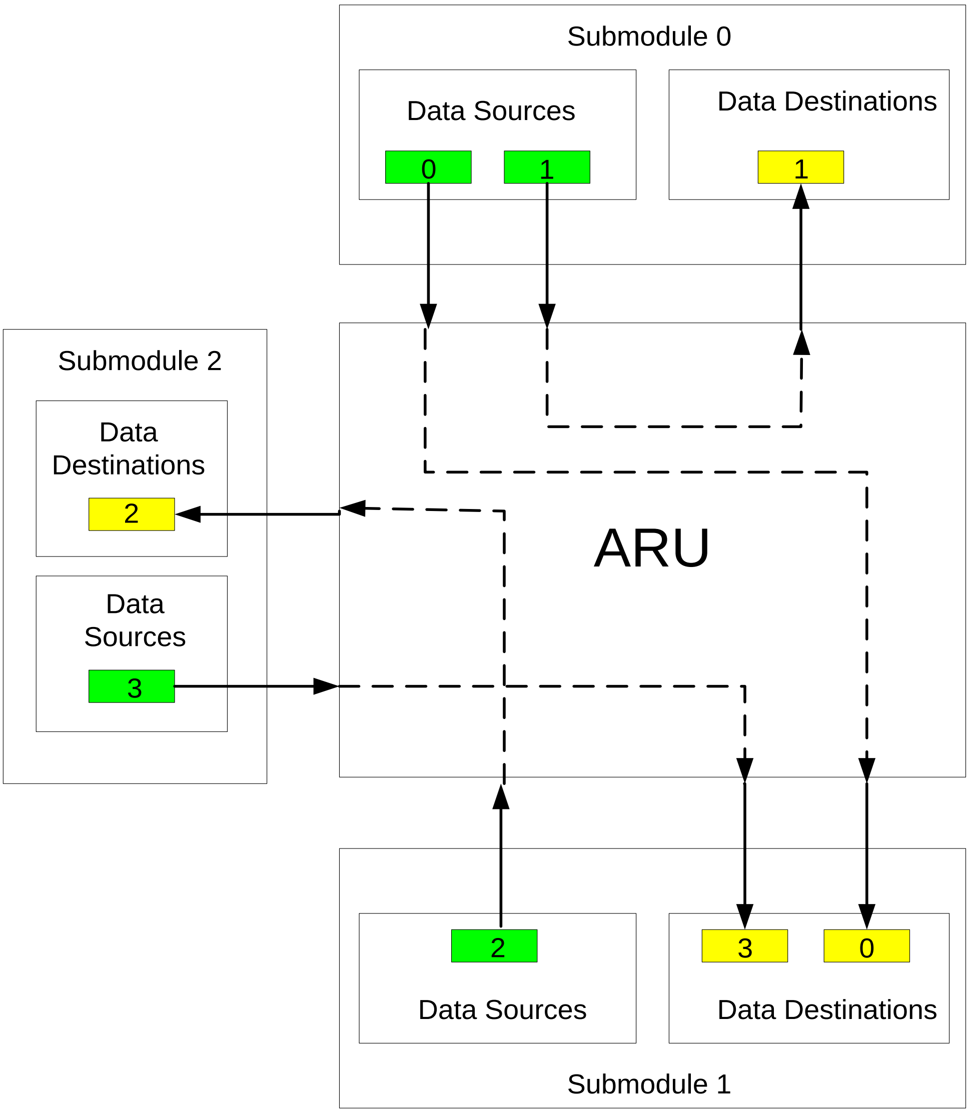
To support multi-master and multitask access to the registers of the GTM-IP a dedicated write-access scheme is used for critical control bits inside the IP that need such a mechanism. This can be for example a shared register where more than one channel can be controlled globally by one register write access. Such register bits are implemented inside the GTM-IP with a double bit mechanism, where the writing of 0b00 and 0b11 has no effect on the register bit and where 0b01 sets the bit and 0b10 resets the bit. If the CPU wants to read the status of the bit it always gets a 0b00 if the bit is reset and it gets an 0b11 if the bit is set.

2.3 ARU Routing Concept

One central concept of the GTM-IP is the routing mechanism of the ARU sub-module for data streams. Each data word transferred between the ARU and its connected sub-module is 53 bit wide. It is important to understand this concept in order to use the resources of the GTM-IP effectively. Each module that is connected to the ARU may provide an arbitrary number of ARU write channels and an arbitrary number of ARU read channels. In the following, the ARU write channels are named data sources and the ARU read channels are named data destinations.

The concept of the ARU intends to provide a flexible and resource efficient way for connecting any data source to an arbitrary data destination. In order to save resource costs, the ARU does not implement a switch matrix, but it implements a data router with serialized connectivity providing the same interconnection flexibility. Figure 2.3.1 shows the ARU data routing principle. Data sources are marked with a green rectangle and the data destinations are marked with yellow rectangles. The dashed lines in the ARU depict the configurable connections between data sources and data destinations. A connection between a data source and a data destination is also called a data stream.

2.3.1 Principle of data routing using ARU



The configuration of the data streams is realized according to the following manner: Each data source has its fixed and unique source address: The ARU read ID.
The fixed address of each data source is pointed out by the numbers in the green boxes of figure 2.3.1. The address definitions of all available data sources in the GTM-IP can be obtained from table 23.3. The connection from a specific data source to a specific data destination is defined by configuring the corresponding address of a data source (i.e. the ARU read ID) in the desired data destination. The configured address of each data destination is pointed out by the numbers in the yellow boxes of figure 2.3.1.

Normally, the destination is idle and waits for data from the source. If the source offers new data, the destination does a destructive read, processes the data and goes idle again. The same data is never read twice.

There is one sub-module for which this destructive read access does not hold. This is the BRC sub-module configured in Maximal Throughput Mode. For a detailed description of this module please refer to chapter 4.

The functionality of the ARU is as follows: The ARU sequentially polls the data destinations of the connected modules in a round-robin order. If a data destination requests new data from its configured data source and the data source has data available, the ARU delivers the data to the destination and it informs both, the data source and destination that the data is transferred. The data source marks the delivered ARU data as invalid which means that the destination consumed the data.

It should be noted that each data source should only be connected to a single data destination. This is because the destinations consume the data. If two destinations would reference the same source one destination would consume the data before the other destination could consume it. Since the data transfers are blocking, the second destination would block until it receives new data from the source. If a data source should be connected to more than one data destination the sub-module Broadcast (BRC) has to be used. On the other hand, the transfer from a data source to the ARU is also blocking, which means that the source channel can only provide new data to the ARU when an old data word is consumed by a destination. In order to speed up the process of data transfers, the ARU handles two different data destinations in parallel.

Following table gives an overview about the number of data destinations and data sources of each GTM-IP instance type.

2.3.2 ARU source and destination address count per instance

Sub-module	Number of data sources per instance	Number of data destinations per instance
ARU	1	0
DPLL	24	24
TIM	8	0
MCS	24	8
BRC	22	12
TOM	0	0
ATOM	8	8
DTM	0	0
PSM	8	8
ICM	0	0
CMP	0	0
MON	0	0
CCM	0	0

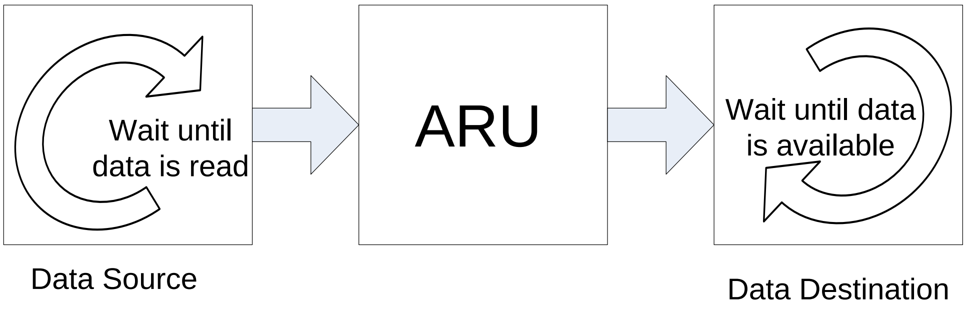
2.3.3 ARU Round Trip Time

The ARU uses a round-robin arbitration scheme with a fixed round trip time for all connected data destinations. This means that the time between two adjacent read requests resulting from a data destination channel always takes the round trip time, independently if the read request succeeds or fails.

2.3.4 ARU Blocking Mechanism

Another important concept of the ARU is its blocking mechanism that is implemented for transferring data from a data source to a data destination. This mechanism is used by ARU connected sub-modules to synchronize the sub-modules to the routed data streams. Figure 2.3.4.1 explains the blocking mechanism.

2.3.4.1 Graphical representation of ARU blocking mechanism



If a data destination requests data from a data source over the ARU but the data source does not have any data yet, it has to wait until the data source provides new data. In this case the sub-module that owns the data destination may perform other tasks. When a data source produces new data faster than a data destination can consume the data the source raises an error interrupt and signals that the data could not be delivered in time. The new data is marked as valid for further transfers and the old data is overwritten.

In any case the round trip time for the ARU has a fixed reset value for a specific device configuration. The end value of the roundtrip counter can be changed with a configuration register **ARU_CADDR_END** inside the ARU. For more details see the ARU specific chapter.
Please refer also to device specific Appendix B of this specification for detailed information [1].

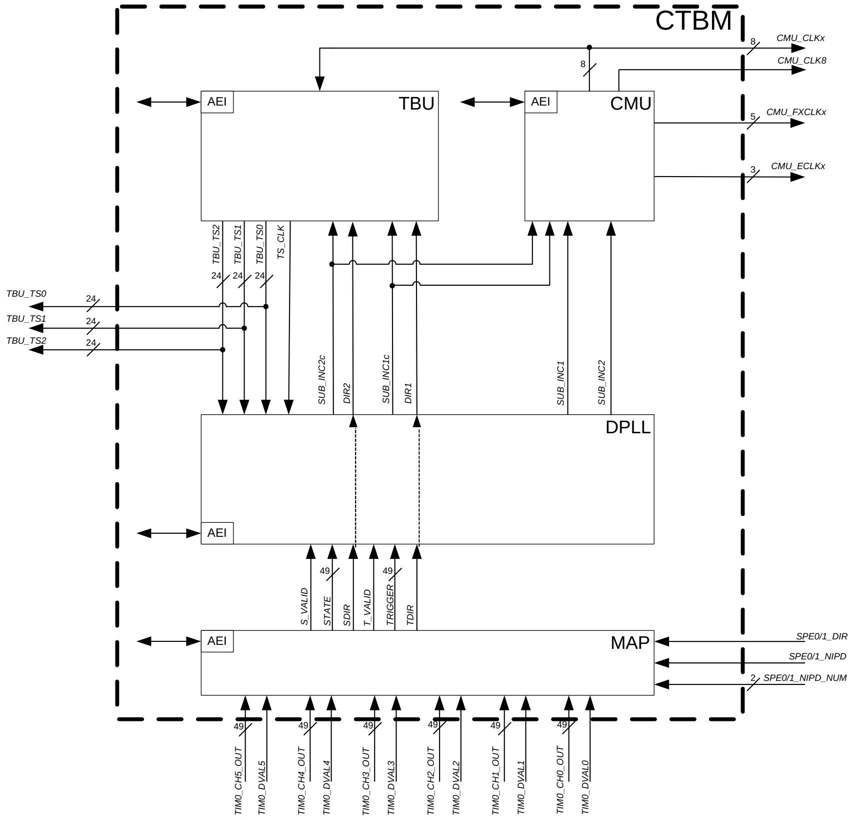
It is possible to reset the ARU roundtrip counter **ARU_CADDR** manually synchronous to CMU clock enable from configuration register inside CMU module. Please refer to CMU specific chapter for more details.

One exception is the BRC sub-module when configured in Maximal Throughput Mode. Please refer to chapter 4 for a detailed description.

2.4 GTM-IP Clock and Time Base Management (CTBM)

Inside the GTM-IP several sub-units are involved in the clock and time base management of the whole GTM. Figure 2.4.1 shows the connections and sub blocks involved in these tasks. The sub blocks involved are called Clock and Time Base Management (CTBM) modules further on.

2.4.1 GTM-IP Clock and time base management architecture



One important module of the CTBM is the Clock Management Unit (CMU) which generates up to 14 clocks for the sub-modules of the GTM and up to three GTM external clocks *CMU_ECLK[z]* (z: 0...2). For a detailed description of the CMU functionality and clocks please refer to Chapter 8.

The five (5) *CMU_FXCLK[y]* (y: 0...4) clocks are used by the TOM sub-module for PWM generation.
A maximum of nine (9) *CMU_CLK[x]* (x: 0...8) clocks are used by other sub-modules of the GTM for signal generation.

Inside the Time Base Unit (TBU) one of *CMU_CLK[x]* (x: 0...7) clocks is used per channel to generate a common time base for the GTM. Besides the *CMU_CLK[x]* signals, the TBU can use the compensated *SUB_INC[i]c* (i: 1,2) signals coming from the DPLL sub-module for time base generation. This time base then typically represents an angle clock for an engine management system. For the meaning of compensated (*SUB_INC[i]c*) and uncompensated (*SUB_INC[i]*) DPLL signals please refer to the DPLL chapter 18. The *SUB_INC[i]c* signals in combination with the two direction signal lines *DIR[i]* the TBU time base can be controlled to run forwards or backwards. The TBU functionality is described in Chapter 10.

The TBU sub-module generates the three time base signals *TBU_TS0*, *TBU_TS1* and *TBU_TS2* which are widely used inside the GTM as common time bases for signal characterization and generation.

Besides the time base 1 and 2 which may represent a relative angle clock for an engine management system it is helpful to have an absolute angle clock for CPU/MCS internal angle algorithm calculations. This absolute angle clock is represented by the TBU base 3. The TBU channel 0 up to 2 are widely used inside the GTM as common time (channel 0, 1 and/or 2) or angle (channel 1 and/or 2) bases for signal characterization and generation. The TBU channel 3 is only configurable and readable by MCS0 or CPU.

As stated before, the DPLL sub-module provides the four clock signals *SUB_INC[i]* and *SUB_INC[i]c* which can be seen as a clock multiplier generated out of the two input signal vectors *TRIGGER* and *STATE* coming from the MAP sub-module. For a detailed description of the DPLL functionality please refer to chapter 18.

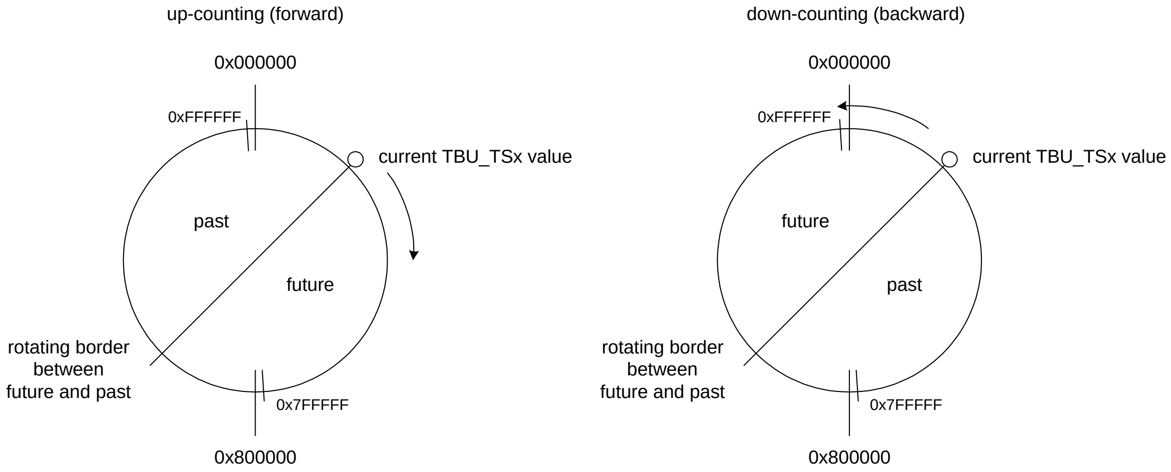
The MAP sub-module is used to select the *TRIGGER* and *STATE* signals for the DPLL out of six input signals coming from TIM0 sub-module. Besides this, the MAP sub-module is able to generate a *TDIR* (TRIGGER Direction) and *SDIR* (STATE Direction) signal for the DPLL and TBU coming from the SPE0 and SPE1 signal lines. The direction signals are generated out of a defined input pattern. For a detailed description of the MAP sub-module please refer to section 17.

2.4.2 Cyclic Event Compare

With the time base module (TBU) the GTM provides three counters, where the counter of TBU_CH0 represents a time and the counter TBU_CH1 and TBU_CH2 may represent a time (if clock source is CMU_CLK generated inside CMU) or an angle (if clock source is a DPLL sub_inc signal provided via CMU).

From application point of view it is necessary to divide the cyclic event counter representing time or angle into two parts, the past and the future.
The border of past/future is a moving border depending on current time or angle value. The cyclic event counting and the moving border of past/future is depicted in the figure below.

2.4.2.1 *Cyclic event counter represeting time or angle*



Inside different submodules of GTM a grater-equal compare (in case of up-counting) or a less-equal compare (in case of down-counting) against a TBU base value (representing time or angle) always means that it is checked if the reference value is in relation to the current TBU value in the future or in the past.

2.5 GTM-IP Interrupt Concept

The sub-modules of the GTM-IP can generate thousands of interrupts on behalf of internal events. This high amount of interrupts is combined inside the Interrupt Concentrator Module (ICM) into interrupt groups. In these interrupt groups the GTM-IP sub-module interrupt signals are bundled to a smaller set of interrupts. From these interrupt sets, a smaller amount of interrupt signals is created and signaled outside of the GTM-IP as a signal *GTM_<MOD>_IRQ,* where <MOD> identifies the name of the corresponding GTM-IP sub-module.

Moreover, each output signal *GTM_<MOD>_IRQ* has a corresponding input signal *GTM_<MOD>_IRQ_CLR* that can be used for clearing the interrupts. These input signals can be used by the surrounding micro controller system as:

- acknowledge signal from a DMA controller
- validation signal from ADC
- clear signal from a GTM-external interrupt controller to do an atomic clear while entering an ISR routine

The controlling of the individual interrupts is done inside the sub-modules. If a sub-module consists of several sub-module channels that are most likely to work independent from each other (like TIM, PSM, MCS, TOM, and ATOM), each sub-module channel has its own interrupt control and status register set, named as interrupt set in the following. Other sub-modules (SPE, ARU, DPLL, BRC, CMP and global GTM functionality) have a common interrupt set for the whole sub-module.

The interrupt set consists of four registers: The **IRQ_EN** register, the **IRQ_NOTIFY** register, the **IRQ_FORCINT** register, and the **IRQ_MODE** register. While the registers **IRQ_EN**, **IRQ_NOTIFY,** and **IRQ_FORCINT** signalize the status and allow controlling of each individual interrupt source within an interrupt set, the register **IRQ_MODE** configures the interrupt mode that is applied to all interrupts that belong to the same interrupt set.

In order to support a wide variety of micro controller architectures and interrupt systems with different interrupt signal output characteristics and internal interrupt handling the following four modes can be configured:

- Level mode
- Pulse mode
- Pulse-Notify mode
- Single-Pulse mode

These interrupt modes are described in more details in the following subsections.

The register **IRQ_EN** allows the enabling and disabling of an individual interrupt within an interrupt set. Independent of the configured mode, only enabled interrupts can signalize an interrupts on its signal *GTM_<MOD>_IRQ*.

The register **IRQ_NOTIFY** collects the occurrence of interrupt events. The behavior for setting a bit in this register depends on the configured mode and thus it is described later on in the mode descriptions.

Independent of the configured mode any write access with value '1' to a bit in the register **IRQ_NOTIFY** always clears the corresponding **IRQ_NOTIFY** bit.

Moreover, the enabling of a disabled interrupt source with a write access to the register **IRQ_EN** also clears the corresponding bit in the **IRQ_NOTIFY** register but only if the error interrupt source **EIRQ_EN** is disabled. However, if the enabling of a disabled interrupt is simultaneous to an incoming interrupt event, the interrupt event is dominant and the register **IRQ_NOTIFY** is not cleared.

Additionally, each write access to the register **IRQ_MODE**, clears all bits in the **IRQ_NOTIFY** register. It should be notified that the clearing of **IRQ_NOTIFY** is applied independently of the written data (e.g. no mode change).

Thus, a secure way for reconfiguring the interrupt mode of an interrupt set, is to disable all interrupts of the interrupt set with the register **IRQ_EN**, define the new interrupt mode by writing register **IRQ_MODE**, followed by enabling the desired interrupts with the register **IRQ_EN**.

Thus, a secure way for reconfiguring the interrupt mode of an error interrupt set, is to disable all error interrupts of the error interrupt set with the register **EIRQ_EN**, define the new interrupt mode by writing register **IRQ_MODE**, followed by enabling the desired error interrupts with the register **EIRQ_EN**.

The register **IRQ_FORCINT** is used by software for triggering individual interrupts with a write access with value '1'. Since a write access to **IRQ_FORCINT** only generates a single pulse, **IRQ_FORCINT** is not implemented as a true register and thus any read access to **IRQ_FORCINT** always results with a value of '0'.

The mechanism for triggering interrupts with **IRQ_FORCINT** is globally disabled after reset. It has to be explicitly enabled by clearing the bit **RF_PROT** in the register **GTM_CTRL** (see section 2.9.3)

For the modules AEI-bridge, BRC, FIFO, TIM, MCS, DPLL, SPE and CMP each interrupt may configured to raise instead of the normal interrupt an error interrupt if enabled by the corresponding error interrupt enable bit in register **EIRQ_EN**.
It is possible for one source to enable the normal interrupt and the error interrupt in parallel. Because both interrupt clear signals could reset the notify bit this is expected to cause problems in a system and therefore it is strongly recommended to not enable both interrupt types at the same point in time.

Similar to enabling an interrupt, the enabling of a disabled error interrupt source with a write access to the register **EIRQ_EN** also clears the corresponding bit in the **IRQ_NOTIFY** register only if the interrupt source **IRQ_EN** is disabled. However, if the enabling of a disabled error interrupt is simultaneous to an incoming interrupt event, the interrupt event is dominant and the register **IRQ_NOTIFY** is not cleared.

All enabled error interrupts are OR-combined inside the ICM and assigned to the dedicated GTM port *gtm_err_irq*. A corresponding input *gtm_err_irq_clr* allows the reset of this error interrupt from outside the GTM (hardware clear).

To be able to detect the module source of the error interrupt the ICM provides the register **ICM_IRQG_MEI**.
The error interrupt causing channel can be determined for the module FIFO by evaluating the ICM register **ICM_IRQG_CEI0**.

The error interrupt causing channel can be determined for the modules TIM by evaluating the ICM register **ICM_IRQG_CEI1…2**.

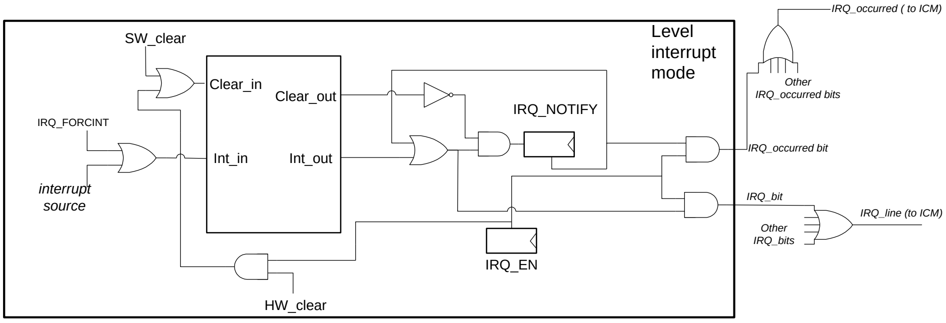
The error interrupt causing channel can be determined for the modules MCS with all possible channel by evaluating the ICM register **ICM_IRQG_MCS[i]_CEI**.
In case of usage only the first 8 channels of each MCS the error interrupt causing channel can be determined by evaluating the ICM register **ICM_IRQG_CEI3…4**.

2.5.1 Level interrupt mode

The default interrupt mode is the Level Interrupt Mode. In this mode each occurred interrupt event is collected in the register **IRQ_NOTIFY**, independent of the corresponding enable bit of register **IRQ_EN** and **EIRQ_EN**.

An interrupt event, which is defined as a pulse on the signal *Int_out* of figure 2.5.1.1, may be triggered by the interrupt source of the sub-module or by software performing a write access to the corresponding register **IRQ_FORCINT**, with a disabled bit **RF_PROT** in register **GTM_CTRL**.

2.5.1.1 Level interrupt mode scheme



A collected interrupt bit in register **IRQ_NOTIFY** may be cleared by a clear event, which is defined as a pulse on signal *Clear_out* of figure 2.5.1.1. A clear event can be performed by writing '1' to the corresponding bit in the register **IRQ_NOTIFY** leading to a pulse on signals *SW_clear*. A clear event may also result from an externally connected signal *GTM_<MOD>_IRQ_CLR*, which is routed to the signal *HW_clear* of figure 2.5.1.1. However, the hardware clear mechanism is only possible, if the corresponding interrupt is enabled by register **IRQ_EN**.

As table 2.5.1.2 shows, interrupt events are dominant in the case of a simultaneous interrupt event and clear event.

2.5.1.2 Priority of Interrupt Events and Clear Events

Int_in	Clear_in	Int_out	Clear_out
0	0	0	0
0	1	0	1
1	0	1	0
1	1	1	0

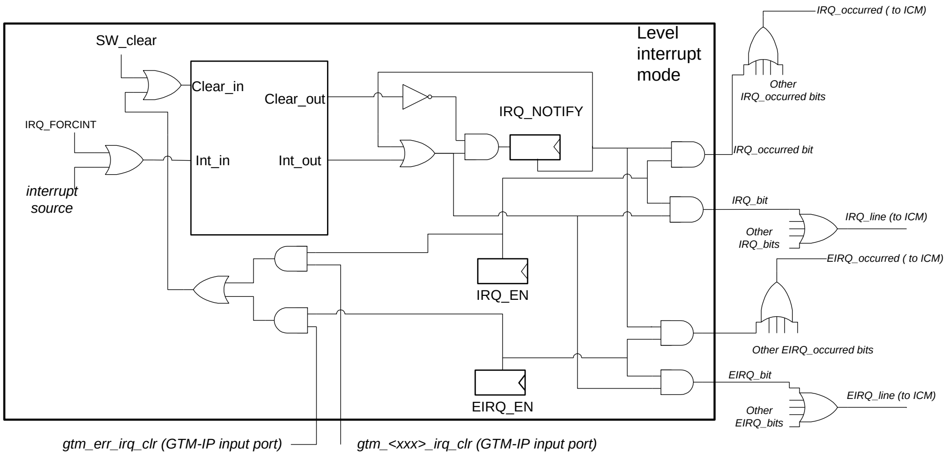
As shown in figure 2.5.1.1 an occurred interrupt event is signaled as a constant signal level with value 1 to the signal *IRQ_bit*, if the corresponding interrupt is enabled in register **IRQ_EN**.

With exception of the sub-modules ARU and DPLL, the signal *IRQ_bit* is OR-combined with the neighboring *IRQ_bit* signals of the same interrupt set and they are routed as a signal *IRQ_line* to the interrupt concentrator module (ICM). The interrupt signals *IRQ_bit* of the sub-modules DPLL and ARU are routed directly as a signal *IRQ_line* to the sub-module ICM. In some cases (sub-modules TOM and ATOM) the ICM may further OR-combine several *IRQ_line* signals to an outgoing interrupt signal *GTM_<MOD>_IRQ*. In the other cases the *IRQ_line* signals are directly connected to the outgoing signals *GTM_<MOD>_IRQ,* within the sub-module ICM.

The signal *IRQ_occurred* is connected in a similar way as the signal *IRQ_line*, however this signal is used for monitoring the interrupt state of the register **IRQ_NOTIFY** in the registers of the ICM.

The additional error interrupt enable mechanism for level interrupt is shown below.

2.5.1.3 *Level interrupt scheme for modules AEI-bridge, BRC, FIFO, TIM, MCS, DPLL, SPE, CMP*



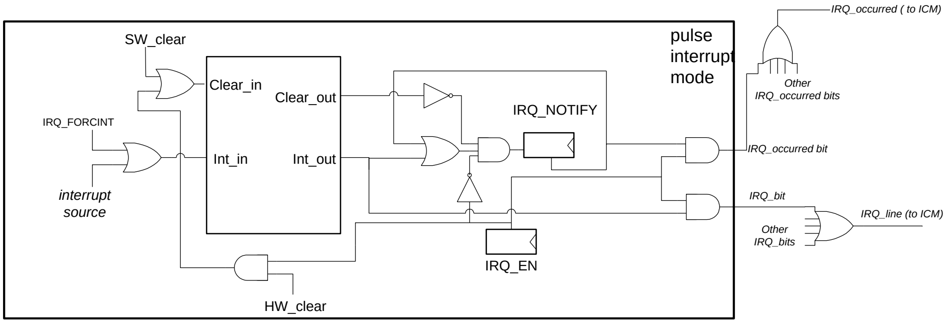
A collected interrupt bit in register **IRQ_NOTIFY** may be cleared by a clear event, which is defined as a pulse on signal *Clear_out* of figure 2.5.1.3. A clear event can be performed by writing '1' to the corresponding bit in the register **IRQ_NOTIFY** leading to a pulse on signals *SW_clear*. A clear event may also result from the externally connected signal *gtm_<MOD>_irq_clr* or *gtm_err_irq_clr*, which is routed as an HW_*clear* to *Clear_in* of figure 2.5.1.3. However, the hardware clear mechanism is only possible, if the corresponding interrupt or error interrupt is enabled by register **IRQ_EN** or **EIRQ_EN**.

As it can be seen from the figure 2.5.1.3 an occurred interrupt event is signaled as a constant signal level with value 1 to the signal *IRQ_bit*, if the corresponding interrupt is enabled in register **IRQ_EN**.

2.5.2 Pulse interrupt mode

The Pulse interrupt mode behavior can be observed from figure 2.5.2.1.

2.5.2.1 Pulse interrupt mode scheme



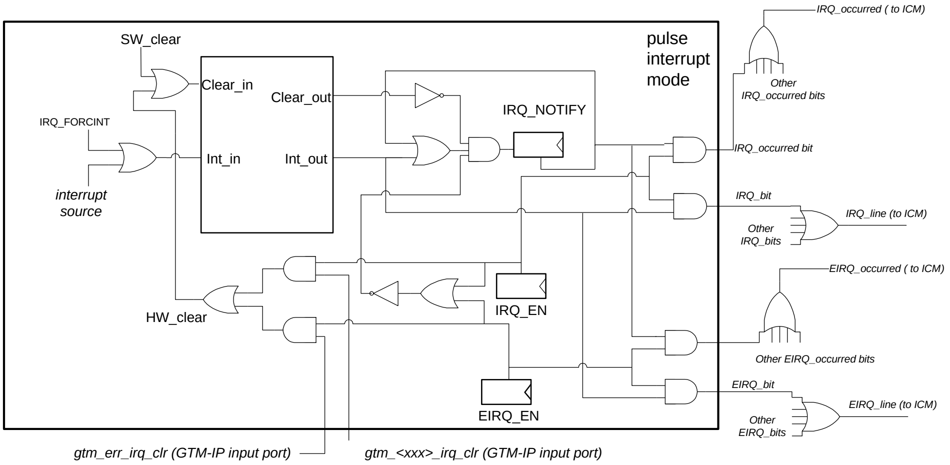
In Pulse Interrupt Mode each Interrupt Event will generate a pulse on the *IRQ_bit* signal if **IRQ_EN** is enabled.

As it can be seen from the figure, the interrupt bit in **IRQ_NOTIFY** register is always cleared if **IRQ_EN** is enabled.

However, if an interrupt is disabled in the register **IRQ_EN**, an occurred interrupt event is captured in the register **IRQ_NOTIFY**, in order to allow polling for disabled interrupts by software.

Disabled interrupts may be cleared by an interrupt clear event.
In Pulse interrupt mode, the signal IRQ_occurred is always 0.
The additional error interrupt enable mechanism for pulse interrupt is shown below.

2.5.2.2 Pulse interrupt scheme for modules AEI-bridge, BRC, FIFO, TIM, MCS, DPLL, SPE, CMP



In Pulse Interrupt Mode each Interrupt Event will generate a pulse on the *EIRQ_bit* signal if **EIRQ_EN** is enabled.

As it can be seen from the figure, the interrupt bit in **IRQ_NOTIFY** register is always cleared if **EIRQ_EN** or **IRQ_EN** are enabled.

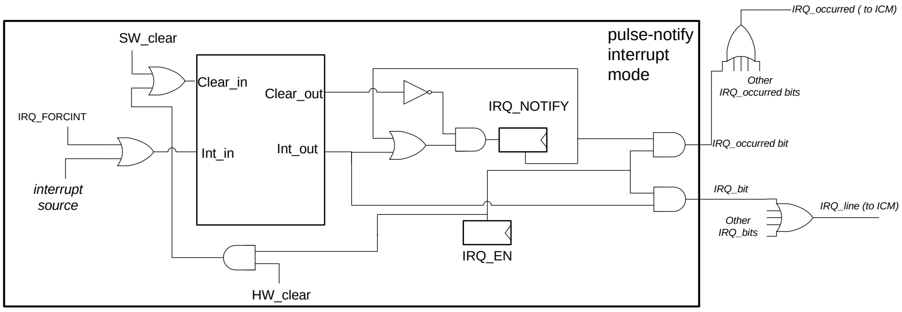
However, if an error interrupt is disabled in the register **EIRQ_EN**, an occurred error interrupt event is captured in the register **IRQ_NOTIFY**, in order to allow polling for disabled error interrupts by software.

Disabled error interrupts may be cleared by an error interrupt clear event.
In Pulse interrupt mode, the signal EIRQ_occurred is always 0.

2.5.3 Pulse-notify interrupt mode

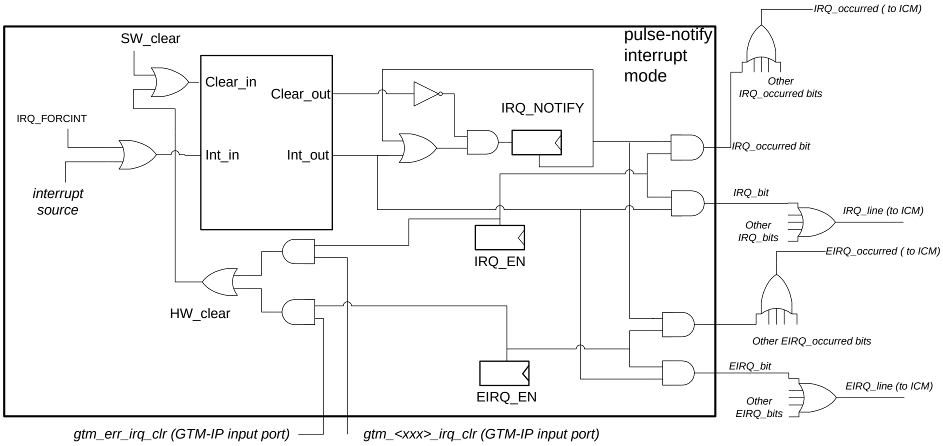
In Pulse-notify Interrupt mode, all interrupt events are captured in the register **IRQ_NOTIFY**. If an interrupt is enabled by the register **IRQ_EN**, each interrupt event will also generate a pulse on the *IRQ_bit* signal. The signal *IRQ_occurred* will be high if interrupt is enabled in register **IRQ_EN** and the corresponding bit of register **IRQ_NOTIFY** is set. The Pulse-notify interrupt mode is shown in figure 2.5.3.1.

2.5.3.1 Pulse-notify interrupt mode scheme



The additional error interrupt enable mechanism for pulse-notify interrupt is shown below

2.5.3.2 *Pulse-notify interrupt scheme for modules AEI-bridge, BRC, FIFO, TIM, MCS, DPLL, SPE, CMP*



In Pulse-notify Interrupt mode, all error interrupt events are captured in the register **IRQ_NOTIFY**. If an error interrupt is enabled by the register **EIRQ_EN**, each error interrupt event will also generate a pulse on the *EIRQ_bit* signal. The signal *EIRQ_occurred* will be high if error interrupt is enabled in register **EIRQ_EN** and the corresponding bit of register **IRQ_NOTIFY** is set. The Pulse-notify interrupt mode for error interrupts is shown in figure 2.5.3.2.

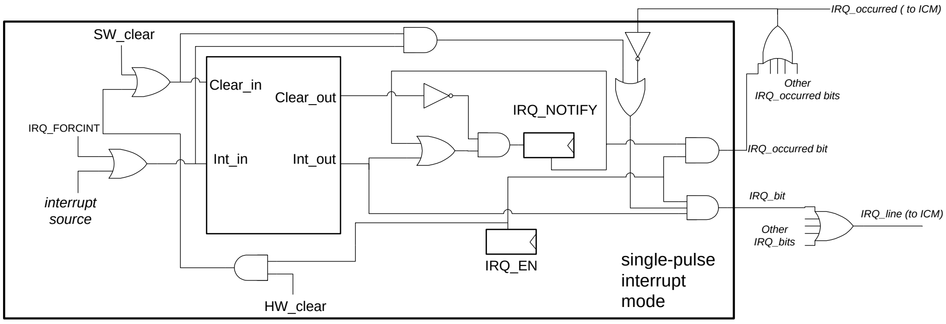
2.5.4 Single-pulse interrupt mode

In Single-pulse Interrupt Mode, an interrupt event is always captured in the register **IRQ_NOTIFY**, independent of the state of **IRQ_EN**. However, only the first interrupt event of an enabled interrupt within a common interrupt set is forwarded to signal *IRQ_line*. Additional interrupt events of the same interrupt set cannot generate pulses on the signal *IRQ_line*, until the corresponding bits in register **IRQ_NOTIFY** of enabled interrupts are cleared by a clear event. The *IRQ_occurred* signal line will be high, if the

IRQ_EN and the **IRQ_NOTIFY** register bits are set. The Single-pulse interrupt mode is shown in figure 2.5.4.1.

The only exceptions are the modules ARU and DPLL. In these modules the *IRQ_occurred* bit of each interrupt is directly connected (without OR-conjunction of neighboring *IRQ_occurred* bits) to the inverter for suppressing further interrupt pulses.

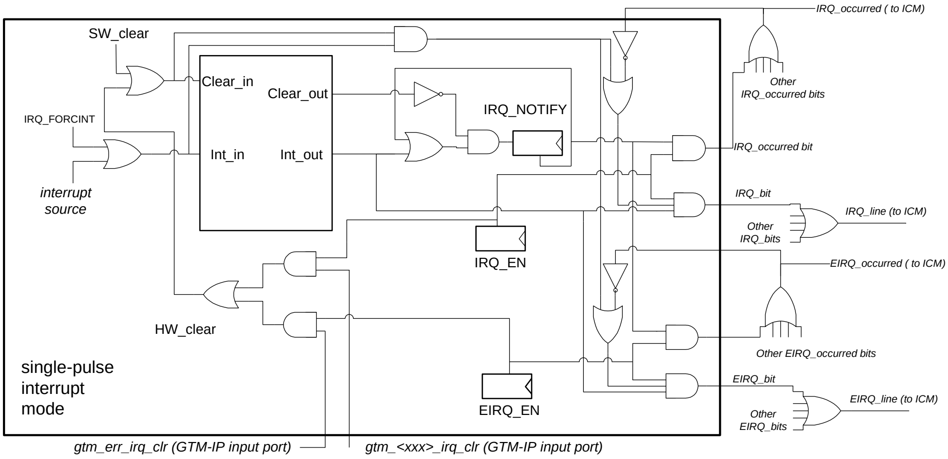
2.5.4.1 Single-pulse interrupt mode scheme



To avoid unexpected IRQ behavior in the single pulse mode, all desired interrupt sources should be enabled by a single write access to **IRQ_EN** and the notification bits should be cleared by a single write access to the register **IRQ_NOTIFY**.

The additional error interrupt enable mechanism for single-pulse interrupt is shown below

2.5.4.2 Single-pulse interrupt scheme for modules AEI-bridge, BRC, FIFO, TIM, MCS, DPLL, SPE, CMP



In Single-pulse Interrupt Mode, an error interrupt event is always captured in the register **IRQ_NOTIFY**, independent of the state of **EIRQ_EN**. However, only the first error interrupt event of an enabled error interrupt within a common error interrupt set is forwarded to signal *EIRQ_line*. Additional error interrupt events of the same error interrupt set cannot generate pulses on the signal *EIRQ_line*, until the corresponding bits in register **IRQ_NOTIFY** of enabled error interrupts are cleared by a clear event. The *EIRQ_occurred* signal line will be high, if the **EIRQ_EN** and the **IRQ_NOTIFY** register bits are set. The Single-pulse interrupt mode for error interrupts is shown in figure 2.5.4.2.

To avoid unexpected EIRQ behavior in the single pulse mode, all desired error interrupt sources should be enabled by a single write access to **EIRQ_EN** and the notification bits should be cleared by a single write access to the register **IRQ_NOTIFY**.

The only exceptions are the modules ARU and DPLL. In these modules the *EIRQ_occurred* bit of each error interrupt is directly connected (without OR-conjunction of neighboring *EIRQ_occurred* bits) to the inverter for suppressing further error interrupt pulses.

2.5.5 GTM-IP Interrupt concentration method

Because of the grouping of interrupts inside the ICM, it can be necessary for the software to access the ICM sub-module first to determine the interrupt set that is responsible for an interrupt. A second access to the responsible register **IRQ_NOTIFY** is then necessary to identify the interrupt source, serve it and to reset the interrupt flag in register **IRQ_NOTIFY** afterwards. The interrupt flags are never reset by an access to the ICM. For a detailed description of the ICM sub-module please refer to chapter 20.

2.6 GTM-IP Software Debugger Support

For software debugger support the GTM-IP comes with several features. E.g. status register bits must not be altered by a read access from a software debugger. To avoid this behavior to reset a status register bit by software, the CPU has to write a '1' explicitly to the register bit to reset its content.

The table 2.6.1 describes the behavior of some GTM-IP registers with special functionality on behalf of read accesses from the AEI bus interface.

2.6.1 Register behavior in case of Software Debugger accesses

Module	Register	Description

AFD	AFD[i]_CH[x]_BUFFACC	The FIFO read access pointers are not altered on behalf of a Debugger read access to this register.
TIM	TIM[i]_CH[x]_GPR0/1	The overflow bit is not altered in case of a Debugger read access to this registers.
ATOM	ATOM[i]_CH[x]_SR0/1	In SOMC mode a read access to this register by the Debugger does not release the channel for a new compare/match event.

Further on, some important states inside the GTM-IP sub-module have to be signaled to the outside world, when reached and should for example trigger the software debugger to stop program execution. For this internal state signaling please refer to the GTM-IP module integration guide.

The GTM provides an external signal *gtm_halt*, which disables clock signal *SYS_CLK* for debugging purposes. If *SYS_CLK* is disabled, a connected debugger can read any GTM related register and the GTM internal RAMs using AEI. Moreover, the debugger can also perform write accesses to the internal RAMs and to all GTM related registers in order to enable advanced debugging features (e.g. modifications of register contents in single step mode).

2.7 GTM-IP Programming conventions

To serve different application domains the GTM-IP is a highly configurable module with many configuration modes. In principle the sub-modules of the GTM-IP are intended to be configured at system startup to fulfill certain functionality for the application domain the micro controller runs in.

For example, a TIM input channel can be used to monitor an application specific external signal, and this signal has to be filtered. Therefore, the configuration of the TIM channel filter mode will be specific to the external signal characteristic. While it can be necessary to adapt the filter thresholds during runtime an adaptation of the filter mode during runtime is not reasonable. Thus, the change of the filter mode during runtime can lead to an unexpected behavior.

In general, the programmer has to be careful when reprogramming configuration registers of the GTM-IP sub-modules during runtime. It is recommended to disable the channels before reconfiguration takes place to avoid unexpected behavior of the GTM-IP.

2.8 GTM-IP TOP-Level Configuration Register Overview

2.8.1 GTM-IP TOP-Level Configuration Register Overview Table

Register name	Description	Details in Section
GTM_REV	GTM Version control register	2.9.1
GTM_RST	GTM Global reset register	2.9.2
GTM_CTRL	GTM Global control register	2.9.3
GTM_AEI_ADDR_XPT	GTM AEI Timeout exception address register	2.9.4
GTM_AEI_STA_XPT	GTM AEI Non zero status register	2.9.5
GTM_IRQ_NOTIFY	GTM Interrupt notification register	2.9.6
GTM_IRQ_EN	GTM Interrupt enable register	2.9.7
GTM_EIRQ_EN	GTM Error interrupt enable register	2.9.14
GTM_IRQ_FORCINT	GTM Software interrupt generation register	2.9.8
GTM_IRQ_MODE	GTM top level interrupts mode selection	2.9.9
GTM_BRIDGE_MODE	GTM AEI bridge mode register	2.9.10
GTM_BRIDGE_PTR1	GTM AEI bridge pointer 1 register	2.9.11
GTM_BRIDGE_PTR2	GTM AEI bridge pointer 2 register	2.9.12
GTM_MCS_AEM_DIS	GTM MCS master port disable register	2.9.13
GTM_CLS_CLK_CFG	GTM Cluster Clock Configuration	2.9.15
GTM_CFG	GTM Configuration register	2.9.16

2.9 GTM TOP-Level Configuration Registers Description

2.9.1 Register GTM_REV

Address Offset:	**see Appendix B**																**Initial Value:**								**0xXXXX_XXXX**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	DEV_CODE2				DEV_CODE1				DEV_CODE0				MAJOR				MINOR				NO				STEP							
Mode	R				R				R				R				R				R				R							
Initial Value	0xX				0xX				0xX				0xX				0xX				0xX				0xXX							

Bit 7:0 **STEP:** release step

GTM Release step

Bit 11:8 **NO:** delivery number
Define delivery number of GTM-IP specification.

Bit 15:12 **MINOR:** minor version number
Define minor version number of GTM-IP specification.

Bit 19:16 **MAJOR:** major version number
Define major version number of GTM-IP specification.

Bit 23:20 **DEV_CODE0:** Device encoding digit 0.
Device encoding digit 0.

Bit 27:24 **DEV_CODE1:** Device encoding digit 1.
Device encoding digit 1.

Bit 31:28 **DEV_CODE2:** Device encoding digit 2.
Device encoding digit 2.
Note: The numbers are encoded in BCD. Values "A" - "F" are characters.

Note: See device specific Appendix B [1] for reset value.

2.9.2 Register GTM_RST

Address Offset:	see Appendix B																Initial Value: 0x0000_0000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved				BRIDGE_MODE_	Reserved																										RST
Mode	R				RPw	R																										RAw
Initial Value	0x0000_0000				0	0x0000_0000																										0

Bit 0 **RST:** GTM-IP Reset.
0 = No reset action
1 = Initiate reset action for all sub-modules
Note: This bit is automatically cleared by hardware after it was written. Therefore, the register is always read as zero (0) by the software.

Note: This bit is write protected by bit RF_PROT of 2.9.3

Bit 26:1 **Reserved**
Note: Read as zero, should be written as zero.

Bit 27 **BRIDGE_MODE_WRDIS:** GTM_BRIDGE_MODE write disable.
0 = writing of GTM_BRIDGE_MODE register is enabled
1 = writing of GTM_BRIDGE_MODE register is disabled
Note: This bit is write protected by bit RF_PROT of 2.9.3

Bit 31:28 **Reserved**
Note: Read as zero, should be written as zero.

2.9.3 Register GTM_CTRL

Address Offset:	see Appendix B																Initial Value:					0x0000_0001										
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																AEIM_CLUSTER				Reserved			TO_VAL					Reserved		TO_MODE	RF_PROT
Mode	R																R				R			RW					R		RW	RW
Initial Value	0x000000																0000				000			00000					00		0	1

Bit 0 **RF_PROT:** RST and FORCINT protection.
0 = SW RST (global), SW interrupt FORCINT, and SW RAM reset functionality is enabled
1 = SW RST (global), SW interrupt FORCINT, and SW RAM reset functionality is disabled

Bit 1 **TO_MODE:** AEI timeout mode.
0 = Observe: If timeout_counter=0 the address and rw signal in addition with timeout flag will be stored to the **GTM_AEI_ADDR_XPT** register. Following timeout_counter=0 accesses will not overwrite the first entry in the aei_addr_timeout register. Clearing the timeout flag/aei_status error_code will reenable the storing of a next faulty access.
1 = Abort: In addition to observe mode the pending access will be aborted by signaling an illegal module access on aei_status and sending ready. In case of a read deliver as data 0 by serving of next AEI accesses.

Bit 3:2 **Reserved:** Read as zero, should be written as zero.
Note: Read as zero, should be written as zero.

Bit 8:4 **TO_VAL:** AEI timeout value.
Note: These bits define the number of cycles after which a timeout event occurs. When TO_VAL equals zero (0) the AEI timeout functionality is disabled.

Bit 11:9 **Reserved:** Read as zero, should be written as zero.
Note: Read as zero, should be written as zero.

Bit 15:12 **AEIM_CLUSTER:** AEIM cluster number

Note: These bits show the number of the AEI master port cluster which throws the interrupts *AEIM_USP_ADDR*, *AEIM_IM_ADDR* and *AEIM_USP_BE* depending on the AEI master port access status.

Note: If one of the corresponding irq notify bits (6:4) is set, this bit field will be frozen until the interrupt notify bits (6:4) are cleared.

Bit 31:16 **Reserved**

Note: Read as zero, should be written as zero.

2.9.4 Register GTM_AEI_ADDR_XPT

<table>
<tr><td><b>Address Offset:</b></td><td colspan="16"><b>see Appendix B</b></td><td colspan="8"><b>Initial Value:</b></td><td colspan="8"><b>0x0000_0000</b></td></tr>
<tr><td></td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>Bit</td><td colspan="11">Reserved</td><td>TO_W1R0</td><td colspan="20">TO_ADDR</td></tr>
<tr><td>Mode</td><td colspan="11">R</td><td>R</td><td colspan="20">R</td></tr>
<tr><td>Initial Value</td><td colspan="11">0x000</td><td>0</td><td colspan="20">0x00000</td></tr>
</table>

Bit 19:0 **TO_ADDR:** AEI timeout address.

Note: This bit field defines the AEI address for which the AEI timeout event occurred.

Bit 20 **TO_W1R0:** AEI timeout Read/Write flag.

Note: This bit defines the AEI Read/Write flag for which the AEI timeout event occurred.

Bit 31:21 **Reserved**

Note: Read as zero, should be written as zero.

2.9.5 Register GTM_AEI_STA_XPT

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved											W1R0	ADDR																			
Mode	R											R	R																			
Initial Value	0x000											0	0x00000																			

Bit 19:0 **ADDR:** AEI exception address.

Note: This bit field captures the address of the first AEI access resulting with a non-zero AEI status signal. The bit field can be cleared by clearing the interrupt flags **AEI_USP_ADDR**, **AEI_USP_BE**, and **AEI_IM_ADDR** in the register **GTM_IRQ_NOTIFY**.

Bit 20 **W1R0:** AEI exception Read/Write flag.

Note: This bit field captures the address of the first AEI access resulting with a non-zero AEI status signal. The bit field can be cleared by clearing the interrupt flags **AEI_USP_ADDR, AEI_USP_BE**, and **AEI_IM_ADDR** in the register **GTM_IRQ_NOTIFY**.

Bit 31:21 **Reserved**

Note: Read as zero, should be written as zero.

2.9.6 Register GTM_IRQ_NOTIFY

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	CLK_EN_EXP_STATE				CLK_EN_ERR_STATE				Reserved															CLK_PER_ERR	CLK_EN_ERR	AEIM_USP_BE	AEIM_IM_ADDR	AEIM_USP_ADDR	AEI_USP_BE	AEI_IM_ADDR	AEI_USP_ADDR	AEI_TO_XPT
Mode	R				R				R															RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw
Initial Value	0x00000				0x00000				0x0000000															0	0	0	0	0	0	0	0	0

Bit 0 **AEI_TO_XPT:** AEI timeout exception occurred.
0 = No interrupt occurred
1 = *AEI_TO_XPT* interrupt was raised by the AEI Timeout detection unit
Note: This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged.

Bit 1 **AEI_USP_ADDR:** AEI unsupported address interrupt.
0 = No interrupt occurred
1 = *AEI_USP_ADDR* interrupt was raised by the AEI interface
Note: This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged.

Bit 2 **AEI_IM_ADDR:** AEI illegal Module address interrupt.
0 = No interrupt occurred
1 = *AEI_IM_ADDR* interrupt was raised by the AEI interface
Note: This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged.

Bit 3 **AEI_USP_BE:** AEI unsupported byte enable interrupt.
0 = No interrupt occurred
1 = *AEI_USP_BE* interrupt was raised by the AEI interface
Note: This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged.

Bit 4 **AEIM_USP_ADDR:** AEI master port unsupported address interrupt.
0 = No interrupt occurred
1 = *AEIM_USP_ADDR* interrupt was raised by the AEI master port interface
Note: This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged.

Bit 5 **AEIM_IM_ADDR:** AEI master port illegal Module address interrupt.
0 = No interrupt occurred
1 = *AEIM_IM_ADDR* interrupt was raised by the AEI master port interface
Note: This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged.

Bit 6 **AEIM_USP_BE:** AEI master port unsupported byte enable interrupt.
0 = No interrupt occurred
1 = *AEIM_USP_BE* interrupt was raised by the AEI master port interface
Note: This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged.

Bit 7 **CLK_EN_ERR:** Clock enable error interrupt.
0 = No interrupt occurred
1 = *CLK_EN_ERR* interrupt was raised by clock enable watchdog

Note: This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged.

Note: Read as zero in case of INT_CLK_EN_GEN = 0b1.

Bit 8 **CLK_PER_ERR:** Clock period error interrupt.
0 = No interrupt occurred
1 = *CLK_PER_ERR* interrupt was raised by clock enable watchdog
Note: This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged.

Note: Read as zero in case of INT_CLK_EN_GEN = 0b1.

Bit 23:9 **Reserved**
Note: Read as zero, should be written as zero.

Bit 27:24 **CLK_EN_ERR_STATE:** Erroneous clock enable state.
This bit field defines the GTM external clk enable state at occurrence of the CLK_EN_ERR event.
Function only available with INT_CLK_EN_GEN = 0b0:
0b00-- = CLK_EN_ERR_STATE(0) enable state for internal clock aei_sys_clk
CLK_EN_ERR_STATE(1) enable state for internal clock aei_sys_clk / 2
0b1--- = reserved
0b-1-- = reserved

Note: Read as zero in case of INT_CLK_EN_GEN = 0b1.

Bit 31:28 **CLK_EN_EXP_STATE:** Expected clock enable state.
This bit field defines the GTM expected clk enable state at occurrence of the CLK_EN_ERR event.
Function only available with INT_CLK_EN_GEN = 0b0:
0b00-- = CLK_EN_EXP_STATE(0) enable state for internal clock aei_sys_clk
CLK_EN_EXP_STATE(1) enable state for internal clock aei_sys_clk / 2
0b1--- = reserved
0b-1-- = reserved

Note: Read as zero in case of INT_CLK_EN_GEN = 0b1.

2.9.7 Register GTM_IRQ_EN

<table>
<tr><td><b>Address Offset:</b></td><td colspan="16"><b>see Appendix B</b></td><td colspan="16"><b>Initial Value: 0x0000_0000</b></td></tr>
<tr><td></td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>Bit</td><td colspan="23">Reserved</td><td>CLK_PER_ERR_I</td><td>CLK_EN_ERR_IR</td><td>AEIM_USP_BE_IR</td><td>AEIM_IM_ADDR_I</td><td>AEIM_USP_ADDR</td><td>AEI_USP_BE_IRQ</td><td>AEI_IM_ADDR_IR</td><td>AEI_USP_ADDR_I</td><td>AEI_TO_XPT_IRQ</td></tr>
<tr><td>Mode</td><td colspan="23">R</td><td>RW</td><td>RW</td><td>RW</td><td>RW</td><td>RW</td><td>RW</td><td>RW</td><td>RW</td><td>RW</td></tr>
<tr><td>Initial Value</td><td colspan="23">0x0000000</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
</table>

Bit 0 **AEI_TO_XPT_IRQ_EN:** *AEI_TO_XPT_IRQ* interrupt enable.
0 = Disable interrupt, interrupt is not visible outside GTM-IP
1 = Enable interrupt, interrupt is visible outside GTM-IP

Bit 1 **AEI_USP_ADDR_IRQ_EN:** *AEI_USP_ADDR_IRQ* interrupt enable.
0 = Disable interrupt, interrupt is not visible outside GTM-IP
1 = Enable interrupt, interrupt is visible outside GTM-IP

Bit 2 **AEI_IM_ADDR_IRQ_EN:** *AEI_IM_ADDR_IRQ* interrupt enable.
0 = Disable interrupt, interrupt is not visible outside GTM-IP
1 = Enable interrupt, interrupt is visible outside GTM-IP

Bit 3 **AEI_USP_BE_IRQ_EN:** *AEI_USP_BE_IRQ* interrupt enable.
0 = Disable interrupt, interrupt is not visible outside GTM-IP
1 = Enable interrupt, interrupt is visible outside GTM-IP

Bit 4 **AEIM_USP_ADDR_IRQ_EN:** *AEI_MUSP_ADDR_IRQ* interrupt enable.
0 = Disable interrupt, interrupt is not visible outside GTM-IP
1 = Enable interrupt, interrupt is visible outside GTM-IP

Bit 5 **AEIM_IM_ADDR_IRQ_EN:** *AEIM_IM_ADDR_IRQ* interrupt enable.
0 = Disable interrupt, interrupt is not visible outside GTM-IP
1 = Enable interrupt, interrupt is visible outside GTM-IP

Bit 6 **AEIM_USP_BE_IRQ_EN:** *AEIM_USP_BE_IRQ* interrupt enable.
0 = Disable interrupt, interrupt is not visible outside GTM-IP
1 = Enable interrupt, interrupt is visible outside GTM-IP

Bit 7 **CLK_EN_ERR_IRQ_EN:** *CLK_EN_ERR_IRQ* interrupt enable.
0 = Disable interrupt, interrupt is not visible outside GTM-IP
1 = Enable interrupt, interrupt is visible outside GTM-IP

Note: Read as zero in case of INT_CLK_EN_GEN = 0b1.

Bit 8 **CLK_PER_ERR_IRQ_EN:** *CLK_PER_ERR_IRQ* interrupt enable.

0 = Disable interrupt, interrupt is not visible outside GTM-IP
1 = Enable interrupt, interrupt is visible outside GTM-IP

Note: Read as zero in case of INT_CLK_EN_GEN = 0b1.

Bit 31:9 **Reserved**
Note: Read as zero, should be written as zero.

2.9.8 Register GTM_IRQ_FORCINT

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_0000**								
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																							TRG_CLK_PER_E	TRG_CLK_EN_ER	TRG_AEIM_USP_	TRG_AEIM_IM_A	TRG_AEIM_USP_	TRG_AEI_USP_B	TRG_AEI_IM_ADD	TRG_AEI_USP_A	TRG_AEI_TO_XP
Mode	R																							RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw
Initial Value	0x0000 000																							0	0	0	0	0	0	0	0	0

Bit 0 **TRG_AEI_TO_XPT:** Trigger *AEI_TO_XPT_IRQ* interrupt by software.
0 = No interrupt triggering
1 = Assert *AEI_TO_XPT_IRQ* interrupt for one clock cycle
Note: This bit is cleared automatically after write.
Note: This bit is write protected by bit RF_PROT of 2.9.3

Bit 1 **TRG_AEI_USP_ADDR:** Trigger *AEI_USP_ADDR_IRQ* interrupt by software.
0 = No interrupt triggering
1 = Assert *AEI_USP_ADDR_IRQ* interrupt for one clock cycle
Note: This bit is cleared automatically after write.
Note: This bit is write protected by bit RF_PROT of 2.9.3

Bit 2 **TRG_AEI_IM_ADDR:** Trigger *AEI_IM_ADDR_IRQ* interrupt by software.
0 = No interrupt triggering
1 = Assert *AEI_IM_ADDR_IRQ* interrupt for one clock cycle
Note: This bit is cleared automatically after write.
Note: This bit is write protected by bit RF_PROT of 2.9.3

Bit 3 **TRG_AEI_USP_BE:** Trigger *AEI_USP_BE_IRQ* interrupt by software.
0 = No interrupt triggering
1 = Assert *AEI_USP_BE_IRQ* interrupt for one clock cycle
Note: This bit is cleared automatically after write.

Note: This bit is write protected by bit RF_PROT of 2.9.3

Bit 4 **TRG_AEIM_USP_ADDR:** Trigger *AEIM_USP_ADDR_IRQ* interrupt by software.
0 = No interrupt triggering
1 = Assert *AEIM_USP_ADDR_IRQ* interrupt for one clock cycle
Note: This bit is cleared automatically after write.
Note: This bit is write protected by bit RF_PROT of 2.9.3

Bit 5 **TRG_AEIM_IM_ADDR:** Trigger *AEIM_IM_ADDR_IRQ* interrupt by software.
0 = No interrupt triggering
1 = Assert *AEIM_IM_ADDR_IRQ* interrupt for one clock cycle
Note: This bit is cleared automatically after write.
Note: This bit is write protected by bit RF_PROT of 2.9.3

Bit 6 **TRG_AEIM_USP_BE:** Trigger *AEIM_USP_BE_IRQ* interrupt by software.
0 = No interrupt triggering
1 = Assert *AEIM_USP_BE_IRQ* interrupt for one clock cycle
Note: This bit is cleared automatically after write.
Note: This bit is write protected by bit RF_PROT of 2.9.3

Bit 7 **TRG_CLK_EN_ERR:** Trigger *CLK_EN_ERR_IRQ* interrupt by software.
0 = No interrupt triggering
1 = Assert *CLK_EN_ERR_IRQ* interrupt for one clock cycle
Note: This bit is cleared automatically after write.
Note: This bit is write protected by bit RF_PROT of 2.9.3

Note: Read as zero in case of INT_CLK_EN_GEN = 0b1.

Bit 8 **TRG_CLK_PER_ERR:** Trigger *CLK_PER_ERR_IRQ* interrupt by software.
0 = No interrupt triggering
1 = Assert *CLK_PER_ERR_IRQ* interrupt for one clock cycle
Note: This bit is cleared automatically after write.
Note: This bit is write protected by bit RF_PROT of 2.9.3

Note: Read as zero in case of INT_CLK_EN_GEN = 0b1.

Bit 31:9 **Reserved**
Note: Read as zero, should be written as zero.

2.9.9 Register GTM_IRQ_MODE

Address Offset:	see Appendix B																Initial Value:								0x0000_000X							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																														IRQ_MODE	
Mode	R																														RW	
Initial Value	0x0000 0000																														xx	

Bit 1:0 **IRQ_MODE:** Interrupt strategy mode selection for the AEI timeout and address monitoring interrupts.
0b00 = Level mode
0b01 = Pulse mode
0b10 = Pulse-Notify mode
0b11 = Single-Pulse mode
Note: The interrupt modes are described in section 2.5.
Note: This mode selection is only valid for the interrupts described in section 2.9.6.

Bit 31:2 **Reserved**
Note: Read as zero, should be written as zero.

2.9.10 Register GTM_BRIDGE_MODE

Address Offset:	see Appendix B																Initial Value:								0xXX00_X00X							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	BUFF_DPT								Reserved							BRG_RST	Reserved			SYNC_INPUT_RE	Reserved		BUFF_OVL	MODE_UP_PGR	Reserved					BYPASS_SYNC	MSK_WR_RSP	BRG_MODE
Mode	R								R							RAw	R			R	R		RCw	R	R					RW	RW	RW
Initial Value	0xXX								0x00							0	0x0			X	0x0		0	0	0x00					0	0	X

Bit 0 **BRG_MODE:** Defines the operation mode for the AEI bridge.
0 = AEI bridge operates in sync_bridge mode
1 = AEI bridge operates in async_bridge mode

Note: Reset value depends on the hardware configuration chosen by silicon vendor.

Bit 1 **MSK_WR_RSP:** Mask write response.

0 = Do not mask the write response. Depending on the selected address the latency for execution can vary due to GTM internal arbitration. After this time the status of the access will be signaled by the signal AEI_STATUS to the bus interface.

1 = Mask write response. The write buffer of the bridge is activated, the actual access will be stored to the write buffer and without latency on the bus interface the acceptance of the access is signaled. AEI_STATUS=0b00 will be signaled. In case of a full write buffer the actual access will be postponed until the next write buffer entry becomes free.

Note: The status of the executed write accesses can be observed by using the notify bits AEI_USP_ADDR ,AEI_IM_ADDR, AEI_USP_BE in the register GTM_IRQ_NOTIFY.

Note: With active write buffer MSK_WR_RSP=1, execution of actions can be delayed due to previous inserted write actions in the transaction buffer which wait to be serviced. This can lead to the fact that an access on the bus to a different peripheral than the GTM might be executed earlier in time than the write access buffered in the GTM. Applications must be setup up with this in mind otherwise unexpected operation can happen.

Bit 2 **BYPASS_SYNC:** Bypass synchronizer flipflops.

Function only available with BRG_MODE=1

0 = synchronizer flip-flops in use, latency increase due to synchronization (aei_clk -> aei_sys_clk and back aei_sys_clk -> aei_clk). This setting must be used if aei_clk and aei_sys_clk operate fully asynchronous by independent clock sources.

1 = synchronizer flip-flops are bypassed. No additional latency due to synchronization. This setting can be used if aei_clk and aei_sys_clk are generated by clock gating or clock division out of a common clock source. Clock edges on aei_clk and aei_sys_clk generated out of the same clock edge of the common clock source must have zero skew.

Bit 7:3 **Reserved**

Note: Read as zero, should be written as zero.

Bit 8 **MODE_UP_PGR:** Mode update in progress.

0 = No update in progress.

1 = Update in progress.

Bit 9 **BUFF_OVL:** Buffer overflow register.
0 = No buffer overflow occurred.
1 = Buffer overflow occurred.
Note: A buffer overflow can occur while multiple aborts are issued by the external bus or a pipelined instruction is started while FBC = 0 (see GTM_BRIDGE_PTR1 register).

Bit 11:10 **Reserved**
Note: Read as zero, should be written as zero.

Bit 12 **SYNC_INPUT_REG:** additional pipelined stage in synchronous bridge mode
0 = No additional pipelined stage implemented.
1 = additional pipelined stage implemented. All accesses in synchronous mode will be increased by one clock cycle.

Note: Reset value depends on the hardware configuration chosen by silicon vendor.

Bit 15:13 **Reserved**
Note: Read as zero, should be written as zero.
Bit 16 **BRG_RST:** Bridge software reset.
0 = No bridge reset request.
1 = Bridge reset request.
Note: This bit is cleared automatically after write.

Bit 23:17 **Reserved**
Note: Read as zero, should be written as zero.

Bit 31:24 **BUFF_DPT:** Buffer depth of AEI bridge.
Signals the buffer depth of the GTM AEI bridge implementation.
Note: Reset value depends on the hardware configuration chosen by silicon vendor.
Note: All writable bits are write protected by bit BRIDGE_MODE_WRDIS of 2.9.2

2.9.11 Register GTM_BRIDGE_PTR1

Address Offset:	**see Appendix B**			**Initial Value:**	**0x0XX0_0000**	
	31 30 29 28 27 26	25 24 23 22 21 20	19 18 17 16 15	14 13 12 11 10	9 8 7 6 5	4 3 2 1 0
Bit	RSP_TRAN_RDY	FBC	ABT_TRAN_PGR	TRAN_IN_PGR	FIRST_RSP_PTR	NEW_TRAN_PTR
Mode	R	R	R	R	R	R
Initial Value	0x00	0xXX	0x0	0x0	0x0	0x0

Bit 4:0 **NEW_TRAN_PTR:** New transaction pointer.
Signals the actual value of the new transaction pointer.

Bit 9:5 **FIRST_RSP_PTR:** First response pointer.
Signals the actual value of first response pointer.

Bit 14:10 **TRAN_IN_PGR:** Transaction in progress pointer (acquire)
Transaction in progress pointer.

Bit 19:15 **ABT_TRAN_PGR:** Aborted transaction in progress pointer.
Aborted transaction in progress pointer.

Bit 25:20 **FBC:** Free buffer count.
Number of free buffer entries.
Note: Initial value depends on the hardware configuration chosen by silicon vendor. (see BUFF_DPT in GTM_BRIDGE_MODE register).

Bit 31:26 **RSP_TRAN_RDY:** Response transactions ready.
Amount of ready response transactions.

Note: This register operates on the AEI_CLK domain.
Note: This register holds diagnosis information about the AEI bus bridge. Each access to the GTM_IP will update the defined pointer bit fields. Depending on the mode of GTM_MODE_BRIDGE (BRG_MODE, MSK_WR_RESP), the AEI protocol and operating frequency which is use, the 4 pointer bit fields will change at different clock cycles relative to the start of the transaction. This leads to the fact that reading the register can show values not equal to the defined Initial Value, even directly after a write to GTM_BRIDGE_MODE with BRG_RST=1 was done.

2.9.12 Register GTM_BRIDGE_PTR2

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_0000**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																											TRAN_IN_PGR2				
Mode	R																											R				
Initial Value	0x0000000																											0x0				

Bit 4:0 **TRAN_IN_PGR2:** Transaction in progress pointer (aquire2)
Transaction in progress pointer 2.

Bit 31:5 **Reserved**
Note: Read as zero, should be written as zero.

Note: This register operates on the GTM_CLK domain.

2.9.13 Register GTM_MCS_AEM_DIS

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_0000**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								DIS_CLS11		DIS_CLS10		DIS_CLS9		DIS_CLS8		DIS_CLS7		DIS_CLS6		DIS_CLS5		DIS_CLS4		DIS_CLS3		DIS_CLS2		DIS_CLS1		DIS_CLS0	
Mode	R								R		R		R		R		R		R		R		R		R		R		R		R	
Initial Value	0x00								00		00		00		00		00		00		00		00		00		00		00		00	

Bit 1:0 **DIS_CLS0:** Disable MCS AEIM access in cluster 0
Multicore encoding in use (**DIS_CLSx(1)** defines the state of the signal)
0b00 = State is 0; MCS AEM access in cluster x enabled (ignore write access)
0b01 = Change state to 0
0b10 = Change state to 1
0b11 = State is 1; MCS AEM access in cluster x disabled(ignore write access)

Note: Any read access to a **DIS_CLSx** bit field will always result in a value 00 or 11 indicating current state. A modification of the state is only performed with the values 01 and 10. Writing the values 00 and 11 is always ignored.

Bit 3:2 **DIS_CLS1:** Disable MCS AEIM access in cluster 1, see bit DIS_CLS0
Bit 5:4 **DIS_CLS2:** Disable MCS AEIM access in cluster 2, see bit DIS_CLS0
Bit 7:6 **DIS_CLS3:** Disable MCS AEIM access in cluster 3, see bit DIS_CLS0
Bit 9:8 **DIS_CLS4:** Disable MCS AEIM access in cluster 4, see bit DIS_CLS0
Bit 11:10 **DIS_CLS5:** Disable MCS AEIM access in cluster 5, see bit DIS_CLS0
Bit 13:12 **DIS_CLS6:** Disable MCS AEIM access in cluster 6, see bit DIS_CLS0
Bit 15:14 **DIS_CLS7:** Disable MCS AEIM access in cluster 7, see bit DIS_CLS0
Bit 17:16 **DIS_CLS8:** Disable MCS AEIM access in cluster 8, see bit DIS_CLS0
Bit 19:18 **DIS_CLS0:** Disable MCS AEIM access in cluster 9, see bit DIS_CLS0
Bit 21:20 **DIS_CLS10:** Disable MCS AEIM access in cluster 10, see bit DIS_CLS0
Bit 23:22 **DIS_CLS11:** Disable MCS AEIM access in cluster 11, see bit DIS_CLS0
Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

2.9.14 Register GTM_EIRQ_EN

Address Offset:	**see Appendix B**																**Initial Value:**							**0x0000_0000**								
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																							CLK_PER_ERR_E	CLK_EN_ERR_EI	AEIM_USP_BE_EI	AEIM_IM_ADDR_	AEIM_USP_ADDR	AEI_USP_BE_EIR	AEI_IM_ADDR_EI	AEI_USP_ADDR_	AEI_TO_XPT_EIR
Mode	R																							RW	RW	RW	RW	RW	RW	RW	RW	RW
Initial Value	0x0000 00																							1	1	0	0	0	0	0	0	0

Bit 0 **AEI_TO_XPT_EIRQ_EN:** *AEI_TO_XPT_EIRQ* error interrupt enable.
0 = Disable error interrupt, interrupt is not visible outside GTM-IP
1 = Enable error interrupt, interrupt is visible outside GTM-IP

Bit 1 **AEI_USP_ADDR_EIRQ_EN:** *AEI_USP_ADDR_EIRQ* error interrupt enable.
0 = Disable error interrupt, interrupt is not visible outside GTM-IP
1 = Enable error interrupt, interrupt is visible outside GTM-IP

Bit 2 **AEI_IM_ADDR_EIRQ_EN:** *AEI_IM_ADDR_EIRQ* error interrupt enable.

0 = Disable error interrupt, interrupt is not visible outside GTM-IP
1 = Enable error interrupt, interrupt is visible outside GTM-IP

Bit 3 **AEI_USP_BE_EIRQ_EN:** *AEI_USP_BE_EIRQ* error interrupt enable.

0 = Disable error interrupt, interrupt is not visible outside GTM-IP
1 = Enable error interrupt, interrupt is visible outside GTM-IP

Bit 4 **AEIM_USP_ADDR_EIRQ_EN:** *AEIM_USP_ADDR_EIRQ* error interrupt enable.
0 = Disable error interrupt, interrupt is not visible outside GTM-IP
1 = Enable error interrupt, interrupt is visible outside GTM-IP

Bit 5 **AEIM_IM_ADDR_EIRQ_EN:** *AEIM_IM_ADDR_EIRQ* error interrupt enable.
0 = Disable error interrupt, interrupt is not visible outside GTM-IP
1 = Enable error interrupt, interrupt is visible outside GTM-IP

Bit 6 **AEIM_USP_BE_EIRQ_EN:** *AEIM_USP_BE_EIRQ* error interrupt enable.
0 = Disable error interrupt, interrupt is not visible outside GTM-IP
1 = Enable error interrupt, interrupt is visible outside GTM-IP

Bit 7 **CLK_EN_ERR_EIRQ_EN:** *CLK_EN_ERR_EIRQ* interrupt enable.
0 = Disable interrupt, interrupt is not visible outside GTM-IP
1 = Enable interrupt, interrupt is visible outside GTM-IP

Note: Read as zero in case of INT_CLK_EN_GEN = 0b1.

Bit 8 **CLK_PER_ERR_EIRQ_EN:** *CLK_PER_ERR_EIRQ* interrupt enable.
0 = Disable interrupt, interrupt is not visible outside GTM-IP
1 = Enable interrupt, interrupt is visible outside GTM-IP

Note: Read as zero in case of INT_CLK_EN_GEN = 0b1.

Bit 31:9 **Reserved**
Note: Read as zero, should be written as zero.

2.9.15 Register GTM_CLS_CLK_CFG

Address Offset:	see Appendix B																Initial Value:								0x00XX_XXXX							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								CLS11_CLK_DIV		CLS10_CLK_DIV		CLS9_CLK_DIV		CLS8_CLK_DIV		CLS7_CLK_DIV		CLS6_CLK_DIV		CLS5_CLK_DIV		CLS4_CLK_DIV		CLS3_CLK_DIV		CLS2_CLK_DIV		CLS1_CLK_DIV		CLS0_CLK_DIV	
Mode	R								RPw		RPw		RPw		RPw		RPw		RPw		RPw		RPw		RPw		RPw		RPw		RPw	
Initial Value	0x00								0bxx		0bxx		0bxx		0bxx		0bxx		0bxx		0bxx		0bxx		0bxx		0bxx		0bxx		0bxx	

Bit 1:0 **CLS0_CLK_DIV:** Cluster 0 Clock Divider

0b00 : Cluster is disabled.
0b01 : Cluster is enabled without clock divider.
0b10 : Cluster is enabled with clock divider 2.
0b11 : Reserved

Note: These bits are only writable if bit field RF_PROT of register GTM_CTRL is cleared.

Note: The configuration value CLS0_CLK_DIV defines the primary input clock period for CMU.

Note: If CLS0_CLK_DIV is configured to a value 0b10 (i.e. clock divider 2), the maximum CMU clock frequency for all other cluster c=1..n is also limited to configured CMU clock frequency of cluster 0.

Bit 3:2 **CLS1_CLK_DIV:** Cluster 1 Clock Divider

0b00 : Cluster is disabled.
0b01 : Cluster is enabled without clock divider.
0b10 : Cluster is enabled with clock divider 2.
0b11 : Reserved

Note: These bits are only writable if bit field RF_PROT of register GTM_CTRL is cleared.

Bit 5:4 **CLS2_CLK_DIV:** Cluster 2 Clock Divider

0b00 : Cluster is disabled.
0b01 : Cluster is enabled without clock divider.
0b10 : Cluster is enabled with clock divider 2.
0b11 : Reserved

Note: These bits are only writable if bit field RF_PROT of register GTM_CTRL is cleared.

Bit 7:6 **CLS3_CLK_DIV:** Cluster 3 Clock Divider

0b00 : Cluster is disabled.
0b01 : Cluster is enabled without clock divider.
0b10 : Cluster is enabled with clock divider 2.
0b11 : Reserved

Note: These bits are only writable if bit field RF_PROT of register GTM_CTRL is cleared.

Bit 9:8 **CLS4_CLK_DIV:** Cluster 4 Clock Divider
0b00 : Cluster is disabled.
0b01 : Cluster is enabled without clock divider.
0b10 : Cluster is enabled with clock divider 2.
0b11 : Reserved

Note: These bits are only writable if bit field RF_PROT of register GTM_CTRL is cleared.

Bit 11:10 **CLS5_CLK_DIV:** Cluster 5 Clock Divider
0b00 : Cluster is disabled.
0b01 : Cluster is enabled without clock divider.
0b10 : Cluster is enabled with clock divider 2.
0b11 : Reserved

Note: These bits are only writable if bit field RF_PROT of register GTM_CTRL is cleared.

Bit 13:12 **CLS6_CLK_DIV:** Cluster 6 Clock Divider
0b00 : Cluster is disabled.
0b01 : Cluster is enabled without clock divider.
0b10 : Cluster is enabled with clock divider 2.
0b11 : Reserved

Note: These bits are only writable if bit field RF_PROT of register GTM_CTRL is cleared.

Bit 15:14 **CLS7_CLK_DIV:** Cluster 7 Clock Divider
0b00 : Cluster is disabled.
0b01 : Cluster is enabled without clock divider.
0b10 : Cluster is enabled with clock divider 2.
0b11 : Reserved

Note: These bits are only writable if bit field RF_PROT of register GTM_CTRL is cleared.

Bit 17:16 **CLS8_CLK_DIV:** Cluster 8 Clock Divider
0b00 : Cluster is disabled.
0b01 : Cluster is enabled without clock divider.

0b10 : Cluster is enabled with clock divider 2.
0b11 : Reserved

Note: These bits are only writable if bit field RF_PROT of register GTM_CTRL is cleared.

Bit 19:18 **CLS9_CLK_DIV:** Cluster 9 Clock Divider
0b00 : Cluster is disabled.
0b01 : Cluster is enabled without clock divider.
0b10 : Cluster is enabled with clock divider 2.
0b11 : Reserved

Note: These bits are only writable if bit field RF_PROT of register GTM_CTRL is cleared.

Bit 21:20 **CLS10_CLK_DIV:** Cluster 10 Clock Divider
0b00 : Cluster is disabled.
0b01 : Cluster is enabled without clock divider.
0b10 : Cluster is enabled with clock divider 2.
0b11 : Reserved

Note: These bits are only writable if bit field RF_PROT of register GTM_CTRL is cleared.

Bit 23:22 **CLS11_CLK_DIV:** Cluster 11 Clock Divider
0b00 : Cluster is disabled.
0b01 : Cluster is enabled without clock divider.
0b10 : Cluster is enabled with clock divider 2.
0b11 : Reserved

Note: These bits are only writable if bit field RF_PROT of register GTM_CTRL is cleared.

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

Note: The availability of configuration bits is indicated by value of bit CFG_CLOCK_RATE in register CCM[c]_HW_CFG.

If CFG_CLOCK_RATE=0, only the values 0b00 and 0b01 are valid for bit fields CLS[c]_CLK_DIV.

If CFG_CLOCK_RATE=1, only the values 0b00, 0b01 and 0b10 are valid for bit fields CLS[c]_CLK_DIV.

If CFG_CLOCK_RATE=1, limitation for some cluster are possible to disable high frequency clock usage. In this case only values 0b00 and 0b10 are valid for bit fields CLS[c]_CLK_DIV. Please refer also to device specific Appendix B of this specification for detailed information [1].

Note: Writing a value to a bit field CLS[c]_CLK_DIV that is not available in the device, an AEI status 0b10 is returned.

(with c = 0 to NCCM-1)

2.9.16 Register GTM_CFG

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																															SRC_IN_MUX
Mode	R																															RW
Initial Value	0x0000 0000																															0

Bit 0 **SRC_IN_MUX**: GTM_TIM[i]_AUX_IN input source selection
0 = use for TIM[i] output of TOM[k]
1 = use for TIM[i] output of TOM[i] (same cluster)
See figure 2.1.4 for details.

Bit 31:1 **Reserved**
Note: Read as zero, should be written as zero.

3 Advanced Routing Unit (ARU)

3.1 Overview

The Advanced Routing Unit (ARU) is a flexible infrastructure component for transferring 53 bit wide data (five control bits and two 24 bit values) between several sub-modules of the GTM core in a configurable manner.

Since the concept of the ARU has already been described in section 2.3, this section only describes additional ARU features that can be used by the software for configuring and debugging ARU related data streams.
Also the definition of 'streams' and 'channels' in the ARU context is done in section 2.3.

The principle of ARU data routing is desribed in chapter 2.3.
In the real GTM implementation the ARU serves in parallel per clock period two individual data destinations, one destination at port ARU-0 and at port ARU-1. Both ARU ports ARU-0 and ARU-1 are running by default in parallel but can be configured in dynamic routing mode (see below) to run in an individual mode.

As already defined in the chapter 2.3, the ARU read ID is the address of the data source that is configured in the data destination module.
These ARU read ID's are selected by the individual counter of ARU ports ARU-0 and ARU-1.
Via the the ARU ports ARU-0 and ARU-1 with each ARU read ID two independent GTM sub-modules are addressed and served.
The combination of ARU port (ARU-0 or ARU-1) and the ARU read ID addresses one ARU wdata source (i.e. the ARU write port of a GTM sub-module).

The assignment of ARU write ports of GTM sub-modules to the ARU ports ARU-0 and ARU-1 and the ARU read ID's is device depending and can be found in Appendix B [1].

3.2 Special Data Sources

Besides the addresses of the sub-module related data sources as described in Table 23.3, the ARU provides two special data sources that can be used for the configuration of data streams. These data sources are defined as follows:

Address 0x1FF: Data source that provides always a 53 bit data word with zeros. A read access to this memory location will never block a requesting data destination.

Address 0x1FE: Data source that never provides a data word. A read access to this memory location will always block a requesting data destination. This is the reset value of the read registers inside the data destinations.

Address 0x000: This address is reserved and can be used to bring data through the ARU registers **ARU_DATA_H** and **ARU_DATA_L** into the system by writing the write address 0x000 into the **ARU_ACCESS** register. This means that software test data can be brought into the GTM-IP by the CPU.

3.3 ARU Access via AEI

Besides the data transfer between the connected sub-modules, there are two possibilities to access ARU data via the AEI.

3.3.1 Default ARU Access

The default ARU access incorporates the registers **ARU_ACCESS**, which is used for initiation of a read or write request and the registers **ARU_DATA_H** and **ARU_DATA_L** that provide the ARU data word to be transferred.

The status of a read or write transfer can be determined by polling specific bits in register **ARU_ACCESS**. Furthermore the *acc_ack* bit in the interrupt notify register is set after the read or write access is performed to avoid data loss e.g. on access cancelation.

A pending read or write request may also be canceled by clearing the associated bit.
In the case of a read request, the AEI access behaves as a read request initiated by a data destination of a module. The read request is served by the ARU immediately when no other destination has a pending read request. This means, that an AEI read access does not take part in the scheduling of the destination channels and that the time between two consecutive read accesses is not limited by the round trip time.

On the other hand, the AEI access has the lowest priority behind the ARU scheduler that serves the destination channels. Thus, in worst case, the read request is served after one round trip of the ARU, when all destination channels would request data at the same point in time.

In the case of the write request, the ARU provides the write data at the address defined by the ADDR bit field inside the **ARU_ACCESS** register.

To avoid data loss, the reserved ARU address 0x0 has to be used to bring data into the system. Otherwise, in case the address specified inside the ADDR bit field is defined for another sub-module that acts as a source at the ARU data loss may occur and no deterministic behavior is guaranteed.

This is because the regular source sub-module is not aware that its address is used by the ARU itself to provide data to a destination.

It is guaranteed that the ARU write data is send to the destination in case of both modules want to provide data at the same time.

Configuring both read and write request bits results in a read request, if the write request bit inside the register isn't already set. The read request bit will be set but not the write request bit. The following table describes the important cases of the bit 12 (RREQ) and bit 13 (WREQ) of the **ARU_ACCESS** register:

*3.3.1.1 WREQ and RREQ in **ARU_ACCESS** register*

AEI write access : aei_wdata (13:12)	**actual value of ARU_ACCESS(13:12)**	**next value of ARU_ACCESS(13:12)**	**comment**
0 0	0 1	0 0	cancel read request
0 0	1 0	0 0	cancel write request
0 1	1 0	1 0	unchanged register
1 0	0 1	0 1	unchanged register
1 1	0 0	0 1	both read and write request results in a read request
1 1	1 0	1 0	as before but WREQ bit is already set -> unchanged register

3.3.2 Debug Access

The debug access mode enables to inspect routed data of configured data streams during runtime.
The ARU provides two independent debug channels, whereas each is configured by a dedicated ARU read address in register **ARU_DBG_ACCESS0** and **ARU_DBG_ACCESS1** respectively.

The registers **ARU_DBG_DATA0_H** and **ARU_DBG_DATA0_L** (**ARU_DBG_DATA1_H** and **ARU_DBG_DATA1_L**) provide read access to the latest data word that the corresponding data source sent through the ARU.

Any time when data is transferred through the ARU from a data source to the destination requesting the data the interrupt signal *ARU_NEW_DATA0*_IRQ (*ARU_NEW_DATA1*_IRQ) is raised.

For advanced debugging purposes, the interrupt signal can also be triggered by software using the register **ARU_IRQ_FORCINT**.

Please note, that the debug mechanism should not be used by the application, when a HW-Debugger is used to trace the ARU communication. In that case, the debug registers are used by the HW-Debugger to specify the ARU streams that should be traced.

3.4 ARU dynamic routing

A dynamic routing feature of the ARU is implemented and can be configured using the additional AEI registers **:**

ARU_CTRL,
ARU_[x]_DYN_CTRL,
ARU_[x]_DYN_RDADDR,
ARU_[x]_DYN_ROUTE_LOW,
ARU_[x]_DYN_ROUTE_HIGH,
ARU_[x]_DYN_ROUTE_SR_LOW and
ARU_[x]_DYN_ROUTE_SR_HIGH.

For further information see the register part of this chapter.

3.4.1 Dynamic routing - CPU controlled

The dynamic routing feature can be enabled separately for ARU-0 and ARU-1 by setting the corresponding bit fields of the register **ARU_CTRL**.

The enabling of the dynamic routing feature is synchronized to the normal routing scheme if ARU master ID-0 is addressed. The dynamic route will started with additional ARU master DYN_READ_ID0.

With the dynamic routing feature it is possible to insert additional ARU master ID's, DYN_READ_IDy (y:0-5), in a defined manner into the normal ARU routing scheme.

While inserting additional ARU master ID's the normal ARU routing scheme is paused. Therefore please consider that inserting additional ARU master ID's will lengthen the normal routing scheme.

It is possible to configure 6 additional ARU master ID's in the **ARU_[x]_DYN_ROUTE_LOW/_HIGH** registers for both ARU-0 and ARU-1.

In the bit field **DYN_CLK_WAIT** of **ARU_[x]_DYN_ROUTE_HIGH** register the number of clock cycles has to be configured, after which one of the additional ARU master ID's will be inserted.

After each configured number of clock cycles the defined ARU master ID's will be inserted cyclic one after each other in the following manner :

... -> DYN_READ_ID0 -> DYN_READ_ID1 -> DYN_READ_ID2 -> DYN_READ_ID3 -> DYN_READ_ID4 -> DYN_READ_ID5 -> DYN_READ_ID0 -> ...

In the shadow registers **ARU_[x]_DYN_ROUTE_SR_LOW/_HIGH** further 6 ARU master, DYN_READ_IDy (y:6-11), can be configured.

The bit DYN_UPDATE_EN in the **ARU_[x]_DYN_ROUTE_SR_HIGH** register controls whether the **ARU_[x]_DYN_ROUTE_LOW/_HIGH** registers are updated from its shadow registers except DYN_UPDATE_EN, it is not updated. The update is executed once after writing **ARU_[x]_DYN_ROUTE_SR_HIGH**. If update started DYN_UPDATE_EN is reset.

With the DYN_ROUTE_SWAP option in the **ARU_[x]_DYN_CTRL** register it is possible to swap the registers **ARU_[x]_DYN_ROUTE_LOW/HIGH** with its shadow registers **ARU_[x]_DYN_ROUTE_SR_LOW/HIGH**. The swapping is executed always after the 6 ARU master DYN_READ_ID's are inserted. So it is possible to insert a maximum of 12 ARU master DYN_READ_ID's cyclic after each configured number of clock cycles. If swap started DYN_UPDATE_EN is reset.

Setting the bit field **DYN_CLK_WAIT** of **ARU_[x]_DYN_ROUTE_HIGH** register to zero, only the defined ARU master DYN_READ_ID's will be executed. The normal ARU routing scheme is stopped.

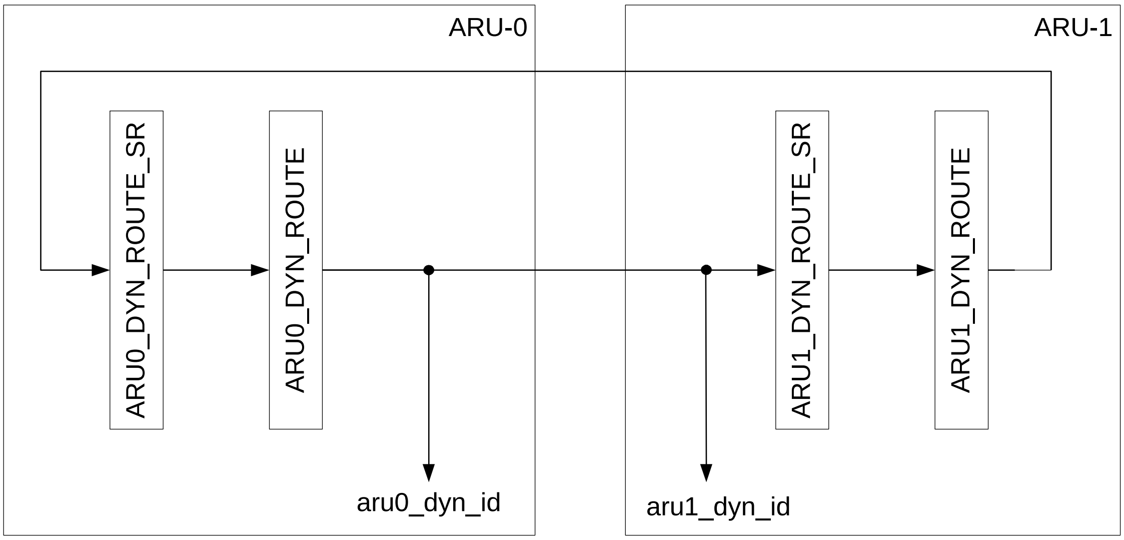
Setting the bit field **DYN_CLK_WAIT** of **ARU_[x]_DYN_ROUTE_HIGH** register to 15, only the normal ARU routing scheme is executed. Inserting of additional ID's is stopped.

To reset the ARU caddr counter and ARU dynamic route counter set bit ARU_ADDR_RSTGLB of **CMU_GLB_CTRL** following by a write access to register **CMU_CLK_EN**.

3.4.1.1 Dynamic routing ring mode

In dynamic routing ring mode it is possible to use all 24 DYN_READ_ID's from both ARU-0 and ARU-1 by setting bit field ARU_DYN_RING_MODE in **ARU_CTRL** register to 1. In this mode all 4 registers **ARU_[x]_DYN_ROUTE_LOW/_HIGH** and **ARU_[x]_DYN_ROUTE_SR_LOW/_HIGH** are connected as a ring, so all 24 DYN_READ_ID's can be used from both ARU's. The ring structure is shown in figure 3.4.1.1.1. The data register shift direction is shown by the arrows in the ring.

3.4.1.1.1 ARU dynamic routing - ring mode



Enabling the dynamic routing ring mode will automatically reset the caddr counter of both ARU-0 and ARU-1. This is necessary to synchronize both ARU's in this mode.

Enabling the dynamic routing ring mode will ignored DYN_ROUTE_SWAP and DYN_UPDATE_EN.
Note: DYN_ARU_UPDATE_EN should be disabled in dynamic routing ring mode.
It is possible to enable the dynamic routing ring mode for both ARU-0 and ARU-1 or only for one of the ARU's by setting the corresponding bit field ARU_0_DYN_EN/ARU_1_DYN_EN of the register **ARU_CTRL.**

Because of the fact that to each ARU port ARU-0 and ARU-1 with the same ARU read ID two diffrent GTM sub-modules are served it may make sense to enable ARU dynamic routing only for one port ARU-0 or ARU-1 if configured to ring-mode. The other port is then served in the default round robin manner.

In dynamic routing ring mode **ARU_[x]_DYN_ROUTE_LOW/_HIGH** and **ARU_[x]_DYN_ROUTE_SR_LOW/_HIGH** are not write-protected.
NOTE: Avoid modification of **ARU_[x]_DYN_ROUTE_LOW/_HIGH** and **ARU_[x]_DYN_ROUTE_SR_LOW/_HIGH** in active dynamic routing ring mode.

3.4.2 Dynamic routing - ARU controlled

Furthermore it is possible to reload the **ARU_[x]_DYN_ROUTE_SR_LOW/_HIGH** registers by ARU itself.
Therefore the ARU has its own master port which will be served in the normal ARU routing scheme. The ARU read address for this master port has to be configured in the register **ARU_[x]_DYN_RDADDR.**

This feature can be enabled by setting bit DYN_ARU_UPDATE_EN of **ARU_[x]_DYN_CTRL** register.
The following mapping of the ARU word to the **ARU_[x]_DYN_ROUTE_LOW/_HIGH** registers is implemented :

ARU_[x]_DYN_ROUTE_SR_LOW(23:0) = aru_data(23:0)
ARU_[x]_DYN_ROUTE_SR_HIGH(28:0) = aru_data(52:24)

The bit field aru_data(51:48) controls the configuration bits DYN_CLK_WAIT and the bit aru_data(52) controls the configuration bit DYN_UPDATE_EN. Both functions are described in the chapter above 3.4.1.

In opposite to the dynamic routing scheme controlled from CPU/AEI (only the 6 additional ARU master DYN_REA_ID's are inserted) two additional ID's are served. One is the ARU master ID itself for reloading and the other is the default ID-0. The ID-0 is only added to the inserted routing scheme if bit field **DYN_CLK_WAIT** of **ARU_[x]_DYN_ROUTE_HIGH** is set to zero (only the inserted routing scheme is executed). This ensures that a debug access can take place even if only the inserted routing scheme is executed.

The following dynamic routing scheme is executed for 15 > **DYN_CLK_WAIT** > 0 :

... -> ARU-master_ID -> DYN_READ_ID0 -> DYN_READ_ID1 -> DYN_READ_ID2 -> DYN_READ_ID3 -> DYN_READ_ID4 -> DYN_READ_ID5 -> ARU-master_ID -> ...

The following dynamic routing scheme is executed for **DYN_CLK_WAIT** = 0 :

... -> ARU-master_ID -> DYN_READ_ID0 -> DYN_READ_ID1 -> DYN_READ_ID2 -> DYN_READ_ID3 -> DYN_READ_ID4 -> DYN_READ_ID5 -> default_ID0 -> ARU-master_ID -> ...

With the possibility of reloading the dynamic routing scheme over ARU, a FIFO or MCS is able to deliver the dynamic routing scheme data.

3.5 ARU Interrupt Signals

3.5.1 ARU Interrupt Signals Table

Signal	**Description**
*ARU_NEW_DATA0*_IRQ	Indicates that data is transferred through the ARU using debug channel **ARU_DBG_ACCESS0**.
*ARU_NEW_DATA1*_IRQ	Indicates that data is transferred through the ARU using debug channel **ARU_DBG_ACCESS1**.
ARU_ACC_ACK_IRQ	ARU access acknowledge IRQ.

3.6 ARU Configuration Register Overview

3.6.1 ARU Configuration Register Overview Table

Register name	**Description**	**Details in Section**
ARU_ACCESS	ARU access register	3.7.1
ARU_DATA_H	ARU access register upper data word	3.7.2
ARU_DATA_L	ARU access register lower data word	3.7.3
ARU_DBG_ACCESS0	ARU debug access channel 0	3.7.4
ARU_DBG_DATA0_H	ARU debug access 0 transfer register upper data word	3.7.5
ARU_DBG_DATA0_L	ARU debug access 0 transfer register lower data word	3.7.6
ARU_DBG_ACCESS1	ARU debug access channel 0	3.7.7
ARU_DBG_DATA1_H	ARU debug access 1 transfer register upper data word	3.7.8
ARU_DBG_DATA1_L	ARU debug access 1 transfer register lower data word	3.7.9
ARU_IRQ_NOTIFY	ARU interrupt notification register	3.7.10
ARU_IRQ_EN	ARU interrupt enable register	3.7.11
ARU_IRQ_FORCINT	ARU force interrupt register	3.7.12
ARU_IRQ_MODE	ARU interrupt mode register	3.7.13
ARU_CADDR_END	ARU caddr counter end value	3.7.14
ARU_CADDR	ARU caddr counter value	3.7.15
ARU_CTRL	ARU enable dynamic routing	3.7.16
ARU_[z]_DYN_CTRL (z:0...1)	ARU z dynamic routing control register	3.7.17

ARU_[z]_DYN_RDADDR (z:0...1)	ARU z read ID for dynamic routing	3.7.18
ARU_[z]_DYN_ROUTE_LOW (z:0...1)	ARU z lower bits of DYN_ROUTE register	3.7.19
ARU_[z]_DYN_ROUTE_HIGH (z:0...1)	ARU z higher bits of DYN_ROUTE register	3.7.20
ARU_[z]_DYN_ROUTE_SR_LOW (z:0...1)	ARU z shadow register for ARU_[z]_DYN_ROUTE_LOW	3.7.21
ARU_[z]_DYN_ROUTE_SR_HIGH (z:0...1)	ARU z shadow register for ARU_[z]_DYN_ROUTE_HIGH	3.7.22

3.7 ARU Configuration Register Description

3.7.1 Register ARU_ACCESS

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_01FE**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																		WREQ	RREQ	Reserved			ADDR								
Mode	R																		RAw	RAw	R			RPw								
Initial Value	0x0000 0																		0	0	000			0x1FE								

Bit 8:0 **ADDR**: ARU address

Define the ARU address used for transferring data

Note: For an ARU write request, the preferred address 0x0 have to be used.

Note: A write request to the address 0x1FF (always full address) or 0x1FE (always empty address) are ignored and doesn't have any effect.

Note: ARU address bits ADDR are only writable if RREQ and WREQ bits are zero

Bit 11:9 **Reserved**

Note: Read as zero, should be written as zero

Bit 12 **RREQ**: Initiate read request

0 = No read request is pending
1 = Set read request to source channel addressed by ADDR
Note: This bit is cleared automatically after transaction. Moreover, it can be cleared by software to cancel a read request.

Note: RREQ bit is only writable if WREQ bit is zero, so to switch from RREQ to WREQ a cancel request has to be performed before.

Note: Configuring both RREQ and WREQ bits results in a read request, so RREQ bit will be set if the WREQ bit of the register isn't already set.

Note: The ARU read request on address ADDR is served immediately when no other destination has actually a read request when the RREQ bit is set by CPU. In a worst case scenario, the read request is served after one round trip of the ARU, but this is only the case when every destination channel issues a read request at consecutive points in time.

Bit 13 **WREQ**: Initiate write request
0 = No write request is pending
1 = Mark data in registers ARU_DATA_H and ARU_DATA_L as valid
Note: This bit is cleared automatically after transaction. Moreover, it can be cleared by software to cancel a write request.

Note: WREQ bit is only writable if RREQ bit is zero, so to switch from WREQ to RREQ a cancel request has to be performed before.

Note: Configuring both RREQ and WREQ bits results in a read request, so WREQ bit will not be set
Note: The data is provided at address ADDR. This address has to be programmed as the source address in the destination sub-module channel. In worst case, the data is provided after one full ARU round trip.

Bit 31:14 **Reserved**
Note: Read as zero, should be written as zero

Note: The register ARU_ACCESS can be used either for reading or for writing at the same point in time.

3.7.2 Register ARU_DATA_H

Address Offset:	see Appendix B																Initial Value:							0x0000_0000								
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved			DATA																												
Mode	R			RW																												
Initial Value	0x0			0x0000000																												

Bit 28:0 **DATA**: Upper ARU data word

Note: Transfer upper ARU data word addressed by ADDR. The data bits 24 to 52 of an ARU word are mapped to the data bits 0 to 28 of this register

Bit 31:29 **Reserved**

Note: Read as zero, should be written as zero

3.7.3 Register ARU_DATA_L

Address Offset:	see Appendix B																Initial Value:							0x0000_0000								
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved			DATA																												
Mode	R			RW																												
Initial Value	0x0			0x0000000																												

Bit 28:0 **DATA**: Lower ARU data word

Note: Transfer lower ARU data word addressed by ADDR. The data bits 0 to 23 of an ARU word are mapped to the data bits 0 to 23 of this register and the data bits 48 to 52 of an ARU word are mapped to the data bits 24 to 28 of this register when data is read by the CPU.

Note: For writing data into the ARU by the CPU the bits 24 to 28 are **not** transferred to bit 48 to 52 of the ARU word. Only bits 0 to 23 are written to bits 0 to 23 of the ARU word

Bit 31:29 **Reserved**
Note: Read as zero, should be written as zero

3.7.4 Register ARU_DBG_ACCESS0

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_01FE**								
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																							ADDR								
Mode	R																							RW								
Initial Value	0x000000																							0x1FE								

Bit 8:0 **ADDR**: ARU debugging address
Note: Define address of ARU debugging channel 0.

Bit 31:9 **Reserved**
Note: Read as zero, should be written as zero

3.7.5 Register ARU_DBG_DATA0_H

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_0000**								
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved			DATA																												
Mode	R			R																												
Initial Value	0x0			0x0000000																												

Bit 28:0 **DATA**: Upper debug data word
Note: Transfer upper ARU data word addressed by register **DBG_ACCESS0**. The data bits 24 to 52 of an ARU word are mapped to the data bits 0 to 28 of this register

Note: The interrupt *ARU_NEW_DATA0_IRQ* is raised if a new data word is available.

Bit 31:29 **Reserved**

Note: Read as zero, should be written as zero

3.7.6 Register ARU_DBG_DATA0_L

Address Offset:	see Appendix B																Initial Value: 0x0000_0000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved			DATA																												
Mode	R			R																												
Initial Value	0x0			0x0000 000																												

Bit 28:0 **DATA**: Lower debug data word

Note: Transfer lower ARU data word addressed by register **DBG_ACCESS0**. The data bits 0 to 23 of an ARU word are mapped to the data bits 0 to 23 of this register and the data bits 48 to 52 of an ARU word is mapped to the data bits 24 to 28 of this register.

Note: The interrupt *ARU_NEW_DATA0_IRQ* is raised if a new data word is available.

Bit 31:29 **Reserved**

Note: Read as zero, should be written as zero

3.7.7 Register ARU_DBG_ACCESS1

Address Offset:	see Appendix B																Initial Value:	0x0000_01FE														
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																							ADDR								
Mode	R																							RW								
Initial Value	0x0000 00																							0x1FE								

Bit 8:0 **ADDR**: ARU debugging address

Note: Define address of ARU debugging channel 1.

Bit 31:9 **Reserved**

Note: Read as zero, should be written as zero

3.7.8 Register ARU_DBG_DATA1_H

Address Offset:	see Appendix B																Initial Value:	0x0000_0000														
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved			DATA																												
Mode	R			R																												
Initial Value	0x0			0x0000 000																												

Bit 28:0 **DATA**: Upper debug data word

Note: Transfer upper ARU data word addressed by register **DBG_ACCESS1**. The data bits 24 to 52 of an ARU word are mapped to the data bits 0 to 28 of this register

Note: The interrupt *ARU_NEW_DATA1_IRQ* is raised if a new data word is available.

Bit 31:29 **Reserved**

Note: Read as zero, should be written as zero

3.7.9 Register ARU_DBG_DATA1_L

Address Offset:	see Appendix B																Initial Value: 0x0000_0000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved			DATA																												
Mode	R			R																												
Initial Value	0x0			0x0000000																												

Bit 28:0 **DATA**: Lower debug data word

Note: Transfer lower ARU data word addressed by register **DBG_ACCESS1**. The data bits 0 to 23 of an ARU word are mapped to the data bits 0 to 23 of this register and the data bits 48 to 52 of an ARU word is mapped to the data bits 24 to 28 of this register.

Note: The interrupt *ARU_NEW_DATA1_IRQ* is raised if a new data word is available.

Bit 31:29 **Reserved**

Note: Read as zero, should be written as zero

3.7.10 Register ARU_IRQ_NOTIFY

Address Offset:	see Appendix B																Initial Value: 0x0000_0000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																													ACC_ACK	NEW_DATA1	NEW_DATA0
Mode	R																													RCw	RCw	RCw
Initial Value	0x00000000																													0	0	0

Bit 0 **NEW_DATA0**: Data was transferred for addr **ARU_DBG_ACCESS0**

0 = No interrupt occurred

1 = *ARU_NEW_DATA0_IRQ* interrupt was raised by the ARU

Note: This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged.

Bit 1 **NEW_DATA1**: Data was transferred for addr **ARU_DBG_ACCESS1**
0 = No interrupt occurred
1 = *ARU_NEW_DATA1_IRQ* interrupt was raised by the ARU
Note: This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged.

Bit 2 **ACC_ACK**: AEI to ARU access finished, on read access data are valid
Note: This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged.

Bit 31:3 **Reserved**
Note: Read as zero, should be written as zero

3.7.11 Register ARU_IRQ_EN

Address Offset:	**see Appendix B**																**Initial Value: 0x0000_0000**																
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																													ACC_ACK_IRQ_E	NEW_DATA1_IRQ	NEW_DATA0_IRQ
Mode	R																													RW	RW	RW
Initial Value	0x0000 0000																													0	0	0

Bit 0 **NEW_DATA0_IRQ_EN**: *ARU_NEW_DATA0_IRQ* interrupt enable
0 = Disable interrupt, interrupt is not visible outside GTM-IP
1 = Enable interrupt, interrupt is visible outside GTM-IP

Bit 1 **NEW_DATA1_IRQ_EN**: *ARU_NEW_DATA1_IRQ* interrupt enable
0 = Disable interrupt, interrupt is not visible outside GTM-IP
1 = Enable interrupt, interrupt is visible outside GTM-IP

Bit 2 **ACC_ACK_IRQ_EN**: *ACC_ACK_IRQ* interrupt enable
0 = Disable interrupt, interrupt is not visible outside GTM-IP
1 = Enable interrupt, interrupt is visible outside GTM-IP

Bit 31:3 **Reserved**
Note: Read as zero, should be written as zero

3.7.12 Register ARU_IRQ_FORCINT

Address Offset:	see Appendix B																Initial Value:							0x0000_0000								
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																													TRG_ACC_ACK	TRG_NEW_DATA	TRG_NEW_DATA
Mode	R																													RAw	RAw	RAw
Initial Value	0x0000 0000																													0	0	0

Bit 0 **TRG_NEW_DATA0**: Trigger new data 0 interrupt
0 = corresponding bit in status register will not be forced
1 = Assert corresponding field in **ARU_IRQ_NOTIFY** register
Note: This bit is cleared automatically after write.
Note: This bit is write protected by bit RF_PROT of register GTM_CTRL

Bit 1 **TRG_NEW_DATA**1: Trigger new data 1 interrupt
0 = corresponding bit in status register will not be forced
1 = Assert corresponding field in **ARU_IRQ_NOTIFY** register
Note: This bit is cleared automatically after write.
Note: This bit is write protected by bit RF_PROT of register GTM_CTRL

Bit 2 **TRG_ACC_ACK**: Trigger ACC_ACK interrupt
0 = corresponding bit in status register will not be forced
1 = Assert corresponding field in **ARU_IRQ_NOTIFY** register
Note: This bit is cleared automatically after write.
Note: This bit is write protected by bit RF_PROT of register GTM_CTRL

Bit 31:3 **Reserved**
Note: Read as zero, should be written as zero

3.7.13 Register ARU_IRQ_MODE

Address Offset:	see Appendix B																Initial Value:								0x0000_000X							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																														IRQ_MODE	
Mode	R																														RW	
Initial Value	0x0000 0000																														XX	

Bit 1:0 **IRQ_MODE**: IRQ mode selection
0b00 = Level mode
0b01 = Pulse mode
0b10 = Pulse-Notify mode
0b11 = Single-Pulse mode
Note: The interrupt modes are described in section 2.5.

Bit 31:2 **Reserved**
Note: Read as zero, should be written as zero

3.7.14 Register ARU_CADDR_END

Address Offset:	see Appendix B																Initial Value:								0x0000_00XX							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																									CADDR_END						
Mode	R																									RW						
Initial Value	0x0000 _000																									0bXXX XXXX						

Bit 6:0 **CADDR_END** set end value of ARU caddr counter
Note: The ARU roundtrip counter aru_caddr runs from zero to caddr_end value.
Note: Shorten the ARU roundtrip cycle by setting a smaller number than the defined reset value will cause that not all ARU-connected modules will be served.

Note: Making the roundtrip cycle longer than the reset value would cause longer ARU roundtrip time and as a result some ARU-connected modules will not be served as fast as possible for this device.

Note: The reset value is device-specific. For more information please refer to Appendix B.

Note: This bit is write protected by bit RF_PROT of register GTM_CTRL

Bit 31:7 **Reserved**

Note: Read as zero, should be written as zero

3.7.15 Register ARU_CADDR

Address Offset:	**see Appendix B**																**Initial Value: 0x0000_0000**															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved									CADDR_1							Reserved									CADDR_0						
Mode	R									R							R									R						
Initial Value	0b0000_0000_0									0b000_0000							0b0000_0000_0									0b000_0000						

Bit 6:0 **CADDR_0** value of ARU-0 caddr counter

Bit 15:7 **Reserved**

Note: Read as zero, should be written as zero

Bit 22:16 **CADDR_1** value of ARU-1 caddr counter

Bit 31:23 **Reserved**

Note: Read as zero, should be written as zero

Note: The registers **CADDR_0** and **CADDR_1** start incrementing with each clock cycle just after reset. Due to this the initial reset value cannot be read back.

3.7.16 Register ARU_CTRL

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_0000**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																											ARU_DYN_RING_	ARU_1_DYN_EN		ARU_0_DYN_EN	
Mode	R																											RW	RW		RW	
Initial Value	0x0000 000																											0	00		00	

Bit 1:0 **ARU_0_DYN_EN** enable dynamic routing for ARU-0
Dynamic routing enable of ARU-0.
Write of following double bit values is possible:
0b00 = no change
0b01 = disable dynamic routing
0b10 = enable dynamic routing
0b11 = no change

Note: If dynamic routing is disabled, the normal ARU routing scheme for ARU-0 is executed.

Bit 3:2 **ARU_1_DYN_EN** enable dynamic routing for ARU-1
Dynamic routing enable of ARU-1
Write of following double bit values is possible:
0b00 = no change
0b01 = disable dynamic routing
0b10 = enable dynamic routing
0b11 = no change

Note: If dynamic routing is disabled, the normal ARU routing scheme for ARU-1 is executed.

Bit 4 **ARU_DYN_RING_MODE** enable dynamic routing ring mode
Dynamic routing ring mode for both ARU-0 and ARU-1
0 : different dynamic routing scheme for ARU-0 and ARU-1
1 : same dynamic routing scheme for ARU-0 and ARU-1 with 24 possible read-ID's (dynamic routing ring mode)

Bit 31:5 **Reserved**
Note: Read as zero, should be written as zero

3.7.17 Register ARU_[z]_DYN_CTRL (z:0...1)

Address Offset:	see Appendix B																Initial Value:							0x0000_0000								
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																														DYN_ROUTE_SW	DYN_ARU_UPDA
Mode	R																														RW	RW
Initial Value	0x0000 0000																														0	0

Bit 0 **DYN_ARU_UPDATE_EN** enable reload of DYN_ROUTE register from ARU itself

Enable reload of DYN_ROUTE register from ARU itself

Bit 1 **DYN_ROUTE_SWAP** enable swapping **DYN_ROUTE_SR** with **DYN_ROUTE** register
Enable swapping **DYN_ROUTE_SR** with **DYN_ROUTE** register

Bit 31:2 **Reserved**
Note: Read as zero, should be written as zero

3.7.18 Register ARU_[z]_DYN_RDADDR (z:0...1)

Address Offset:	see Appendix B																Initial Value:							0x0000_0000								
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																							DYN_ARU_RDADDR								
Mode	R																							RW								
Initial Value	0x0000 00																							0x000								

Bit 8:0 **DYN_ARU_RDADDR** ARU read address ID to reload the DYN_ROUTE register

ARU read address ID to reload the DYN_ROUTE register from ARU itself

Bit 31:9 **Reserved**
Note: Read as zero, should be written as zero

3.7.19 Register ARU_[z]_DYN_ROUTE_LOW (z:0...1)

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								DYN_READ_ID2								DYN_READ_ID1								DYN_READ_ID0							
Mode	R								RW								RW								RW							
Initial Value	0x00								0x00								0x00								0x00							

Bit 7:0 **DYN_READ_ID0** ARU read ID 0
ARU read ID 0 for dynamic routing

Bit 15:8 **DYN_READ_ID1** ARU read ID 2
ARU read ID 1 for dynamic routing

Bit 23:16 **DYN_READ_ID2** ARU read ID 2
ARU read ID 2 for dynamic routing

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero

3.7.20 Register ARU_[z]_DYN_ROUTE_HIGH (z:0...1)

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved				DYN_CLK_WAIT				DYN_READ_ID5								DYN_READ_ID4								DYN_READ_ID3							
Mode	R				RW				RW								RW								RW							
Initial Value	0x0				0x0				0x00								0x00								0x00							

Bit 7:0 **DYN_READ_ID3** ARU read ID 3
ARU read ID 3 for dynamic routing

Bit 15:8 **DYN_READ_ID4** ARU read ID 4
ARU read ID 4 for dynamic routing

Bit 23:16 **DYN_READ_ID5** ARU read ID 5
ARU read ID 5 for dynamic routing

Bit 27:24 **DYN_CLK_WAIT** number of clk cycles for dynamic routing
Defines the number of clk cycles between each dynamic routing ID

Bit 31:28 **Reserved**
Note: Read as zero, should be written as zero

3.7.21 Register ARU_[z]_DYN_ROUTE_SR_LOW (z:0...1)

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_0000**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								DYN_READ_ID8								DYN_READ_ID7								DYN_READ_ID6							
Mode	R								RW								RW								RW							
Initial Value	0x00								0x00								0x00								0x00							

Bit 7:0 **DYN_READ_ID6** ARU read ID 6
ARU read ID 6 for dynamic routing
NOTE : These bits are mapped to ARU data bits aru_data(7:0)

Bit 15:8 **DYN_READ_ID7** ARU read ID 7
ARU read ID 7 for dynamic routing
NOTE : These bits are mapped to ARU data bits aru_data(15:8)

Bit 23:16 **DYN_READ_ID8** ARU read ID 8
ARU read ID 8 for dynamic routing
NOTE : These bits are mapped to ARU data bits aru_data(23:16)

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero

NOTE : This is the shadow register for register **ARU_[x]_DYN_ROUTE_LOW**

3.7.22 Register ARU_[z]_DYN_ROUTE_SR_HIGH (z:0...1)

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved			DYN_UPDATE_EN	DYN_CLK_WAIT				DYN_READ_ID11								DYN_READ_ID10								DYN_READ_ID9							
Mode	R			RW	RW				RW								RW								RW							
Initial Value	0x0			0	0x0				0x00								0x00								0x00							

Bit 7:0 **DYN_READ_ID9** ARU read ID 9
ARU read ID 9 for dynamic routing
NOTE : These bits are mapped to ARU data bits aru_data(31:24)

Bit 15:8 **DYN_READ_ID10** ARU read ID 10
ARU read ID 10 for dynamic routing
NOTE : These bits are mapped to ARU data bits aru_data(39:32)

Bit 23:16 **DYN_READ_ID11** ARU read ID 11
ARU read ID 11 for dynamic routing
NOTE : These bits are mapped to ARU data bits aru_data(47:40)

Bit 27:24 **DYN_CLK_WAIT** number of clk cycles for dynamic routing
Defines the number of clk cycles between each dynamic routing ID

NOTE : These bits are mapped to ARU data bits aru_data(51:48)

Bit 28 **DYN_UPDATE_EN** update enable from shadow register
Enable update **ARU_[x]_DYN_ROUTE_LOW/_HIGH** registers from shadow registers **ARU_[x]_DYN_ROUTE_SR_LOW/_HIGH**.

NOTE : This bit are mapped to ARU data bits aru_data(52)

Bit 31:29 **Reserved**
Note: Read as zero, should be written as zero

NOTE : This is the shadow register for register **ARU_[x]_DYN_ROUTE_HIGH**

4 Broadcast Module (BRC)

4.1 Overview

Since each write address for the sub-module channels of the GTM-IP that are able to write to the ARU can only be read by a single module, it is impossible to provide a data stream to different modules in parallel (This statement holds not for sources, which do not invalidate their data after the data were read by any consumer, e.g. DPLL).

To overcome this issue for regular modules, the sub-module Broadcast (BRC) enables to duplicate data streams multiple times.

The BRC sub-module provides 12 input channels as well as 22 output channels.
In order to clone an incoming data stream, the corresponding input channel can be mapped to zero or more output channels.

When mapped to zero no channel is read.
To destroy an incoming data stream, the **EN_TRASHBIN** bit inside the **BRC_SRC_[x]_DEST** register has to be set.
The total number of output channels that are assigned to a single input channel is variable. However, the total number of assigned output channels must be less than or equal to 22.

4.2 BRC Configuration

As it is the case with all other sub-modules connected to the ARU, the input channels can read arbitrary ARU address locations and the output channels provide the broadcast data to fixed ARU write address locations.

The associated write addresses for the BRC sub-module are fixed and can be obtained from Chapter 23.

The read address for each input channel is defined by the corresponding register **BRC_SRC_[x]_ADDR** (x: 0..11).
The mapping of an input channel to several output channels is defined by setting the appropriate bits in the register **BRC_SRC_[x]_DEST** (x: 0..11). Each output channel is represented by a single bit in the register **BRC_SRC_[x]_DEST**. The address of the output channel is defined in Chapter 23.

If no output channel bit is set within a register **BRC_SRC_[x]_DEST**, no data is provided to the corresponding ARU write address location from the defined read input specified by **BRC_SRC_[x]_ADDR**. This means that the channel does not broadcast any data and is disabled (reset state).

Besides the possibility of mapping an input channel to several output channels, the bit **EN_TRASHBIN** of register **BRC_SRC_[x]_DEST** may be set, which results in dropping an incoming data stream. In this case the data of an input channel defined by **BRC_SRC_[x]_ADDR** is consumed by the BRC module and not routed to any succeeding sub-module. In consequence, the output channels defined in the register **BRC_SRC_[x]_DEST** are ignored. Therefore, the bits 0 to 21 are set to zero (0) when trash bin functionality is enabled.

In general, the BRC sub-module can work in two independent operation modes. In the first operation mode the data consistency is guaranteed since a BRC channel requests only new data from a source when all destination channels for the BRC have consumed the old data value. This mode is called *Data Consistency Mode* (DCM).

In a second operation mode the BRC channel always requests data from a source and distributes this data to the destination regardless whether all destinations have already consumed the old data. This mode is called *Maximum Throughput Mode* (MTM).

MTM ensures that always the newest available data is routed through the system, while it is not guaranteed data consistency since some of the destination channels can be provided with the old data while some other destination channels are provided with the new data. If this is the case, the Data Inconsistency Detected Interrupt *BRC_DID_IRQ[x]* is raised but the channel continues to work.

Furthermore in MTM mode it is guaranteed that it is not possible to read a data twice by a read channel. This is blocked.

The channel mode can be configured inside the **BRC_SRC_[x]_ADDR** register.
To avoid invalid configurations of the registers **BRC_SRC_[x]_DEST**, the BRC also implements a plausibility check for these configurations. If the software assigns an already used output channel to a second input channel, BRC performs an auto correction of the lastly configured register **BRC_SRC_[x]_DEST** and it triggers the interrupt *BRC_DEST_ERR*.

Consider the following example for clarification of the auto correction mechanism. Assume that the following configuration of the 22 lower significant bits for the registers **BRC_SRC_[x]_DEST**:

BRC_SRC_0_DEST:	00 0000 0000 1000 1000 0000 (binary)
BRC_SRC_1_DEST:	00 0000 0000 0100 0000 0100 (binary)
BRC_SRC_2_DEST:	00 0000 0000 0001 0100 0010 (binary)
BRC_SRC_3_DEST:	00 0000 0000 0010 0001 1001 (binary)

If the software overwrites the value for register **BRC_SRC_2_DEST** with

BRC_SRC_2_DEST:	00 0000 0000 1001 0010 0010 (binary)

(changed bits are underlined), then the BRC releases a *BRC_DEST_ERR* interrupt since bit 11 is already assigned in register **BRC_SRC_0_DEST**. The auto correction forces bit 11 to be cleared. The modifications of the bits 5 and 6 are accepted, since there is no violation with previous configurations. So the result of the write access mentioned above results in the following modified register configuration:

BRC_SRC_2_DEST: 00 0000 0000 0001 0010 0010 (binary)

For debug purposes, the interrupt *BRC_DEST_ERR* can also be released by writing to register **BRC_IRQ_FORCINT**. Nevertheless, the interrupt has to be enabled to be visible outside of the GTM-IP.

4.3 BRC Interrupt Signals

Signal	**Description**
BRC_DEST_ERR_IRQ	Indicating configuration errors for BRC module
BRC_DID_IRQ[x]	Data inconsistency occurred in MTM mode (x:0..11)

4.4 BRC Configuration Register Overview

4.4.1 BRC Configuration Register Overview Table

Register Name	**Description**	**Details in Section**
BRC_SRC_[z]_ADDR (z:0...11)	BRC read address for input channel z	4.5.1
BRC_SRC_[z]_DEST (z:0...11)	BRC destination channels for input channel z	4.5.2
BRC_IRQ_NOTIFY	BRC interrupt notification register	4.5.3
BRC_IRQ_EN	BRC interrupt enable register	4.5.4
BRC_EIRQ_EN	BRC error interrupt enable register	4.5.7
BRC_IRQ_FORCINT	BRC force interrupt register	4.5.5
BRC_RST	BRC software reset register	4.5.8
BRC_IRQ_MODE	BRC interrupt mode configuration register	4.5.6

4.5 BRC Configuration Register Description

4.5.1 Register BRC_SRC_[z]_ADDR (z:0...11)

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_01FE**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																			BRC_MODE	Reserved			ADDR								
Mode	R																			RW	R			RPw								
Initial Value	0x00000																			0	000			0x1FE								

Bit 8:0 **ADDR:** Source ARU address. Define an ARU read address used as data source for input channel z (z:0...11).
Note: this bit field is only writable if channel is disabled.

Bit 11:9 **Reserved:** Reserved
Note: Read as zero, should be written as zero

Bit 12 **BRC_MODE:** BRC Operation mode select.
0 = Consistency Mode (DCM) selected
1 = Maximum Throughput Mode (MTM) selected
Note: this bit field is only writable if channel is disabled.

Bit 31:13 **Reserved:** Reserved
Note: Read as zero, should be written as zero

4.5.2 Register BRC_SRC_[z]_DEST (z:0...11)

Address Offset:	see Appendix B																Initial Value: 0x0000_0000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved									EN_TRASHBIN	EN_DEST21	EN_DEST20	EN_DEST19	EN_DEST18	EN_DEST17	EN_DEST16	EN_DEST15	EN_DEST14	EN_DEST13	EN_DEST12	EN_DEST11	EN_DEST10	EN_DEST9	EN_DEST8	EN_DEST7	EN_DEST6	EN_DEST5	EN_DEST4	EN_DEST3	EN_DEST2	EN_DEST1	EN_DEST0
Mode	R									RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Initial Value	0x00									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit 0 **EN_DEST0:** Enable BRC destination address 0
0 = Destination address 0 not mapped to source **BRC_SRC_[z]_ADDR**
1 = Destination address 0 mapped to source **BRC_SRC_[z]_ADDR**

Note: The destination address 0 for BRC channel is defined in section 23.3

Bit 1 **EN_DEST1:** Enable BRC destination address 1, see bit 0.
Bit 2 **EN_DEST2:** Enable BRC destination address 2, see bit 0.
Bit 3 **EN_DEST3:** Enable BRC destination address 3, see bit 0.
Bit 4 **EN_DEST4:** Enable BRC destination address 4, see bit 0.
Bit 5 **EN_DEST5:** Enable BRC destination address 5, see bit 0.
Bit 6 **EN_DEST6:** Enable BRC destination address 6, see bit 0.
Bit 7 **EN_DEST7:** Enable BRC destination address 7, see bit 0.
Bit 8 **EN_DEST8:** Enable BRC destination address 8, see bit 0.
Bit 9 **EN_DEST9:** Enable BRC destination address 9, see bit 0.
Bit 10 **EN_DEST10:** Enable BRC destination address 10, see bit 0.
Bit 11 **EN_DEST11:** Enable BRC destination address 11, see bit 0.
Bit 12 **EN_DEST12:** Enable BRC destination address 12, see bit 0.
Bit 13 **EN_DEST13:** Enable BRC destination address 13, see bit 0.
Bit 14 **EN_DEST14:** Enable BRC destination address 14, see bit 0.
Bit 15 **EN_DEST15:** Enable BRC destination address 15, see bit 0.
Bit 16 **EN_DEST16:** Enable BRC destination address 16, see bit 0.
Bit 17 **EN_DEST17:** Enable BRC destination address 17, see bit 0.
Bit 18 **EN_DEST18:** Enable BRC destination address 18, see bit 0.
Bit 19 **EN_DEST19:** Enable BRC destination address 19, see bit 0.
Bit 20 **EN_DEST20:** Enable BRC destination address 20, see bit 0.
Bit 21 **EN_DEST21:** Enable BRC destination address 21, see bit 0.
Bit 22 **EN_TRASHBIN:** Control trash bin functionality.
0 = Trash bin functionality disabled
1 = Trash bin functionality enabled
Note: When bit **EN_TRASHBIN** is enabled bits 0 to 21 are ignored for this input channel. Therefore, the bits 0 to 21 are set to zero (0) when trash bin functionality is enabled.

Bit 31:23 **Reserved:** Reserved
Note: Read as zero, should be written as zero

Note: The bits 0 to 21 are cleared by auto correction mechanism if a destination channel is assigned to multiple source channels.

Note: When a BRC input channel is disabled (all **EN_DESTz** (z:0...21) bits are reset to zero) the internal states are reset to their reset value.

4.5.3 Register BRC_IRQ_NOTIFY

Address Offset:	**see Appendix B**																**Initial Value: 0x0000_0000**															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																			DID11	DID10	DID9	DID8	DID7	DID6	DID5	DID4	DID3	DID2	DID1	DID0	DEST_ERR
Mode	R																			RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw
Initial Value	0x0000 0000																			0	0	0	0	0	0	0	0	0	0	0	0	0

Bit 0 **DEST_ERR:** Configuration error interrupt for BRC sub-module
0 = No BRC configuration error occurred
1 = BRC configuration error occurred
Note: This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged.

Bit 1 **DID0:** Data inconsistency occurred in MTM mode.
Note: This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged.

Bit 2 **DID1:** Data inconsistency occurred in MTM mode.
Note: This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged.

Bit 3 **DID2:** Data inconsistency occurred in MTM mode.
Note: This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged.

Bit 4 **DID3:** Data inconsistency occurred in MTM mode.
Note: This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged.

Bit 5 **DID4:** Data inconsistency occurred in MTM mode.
Note: This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged.

Bit 6 **DID5:** Data inconsistency occurred in MTM mode.

Note: This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged.

Bit 7 **DID6:** Data inconsistency occurred in MTM mode.
Note: This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged.

Bit 8 **DID7:** Data inconsistency occurred in MTM mode.
Note: This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged.

Bit 9 **DID8:** Data inconsistency occurred in MTM mode.
Note: This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged.

Bit 10 **DID9:** Data inconsistency occurred in MTM mode.
Note: This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged.

Bit 11 **DID10:** Data inconsistency occurred in MTM mode.
Note: This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged.

Bit 12 **DID11:** Data inconsistency occurred in MTM mode.
Note: This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged.

Bit 31:13 **Reserved:** Reserved
Note: Read as zero, should be written as zero

4.5.4 Register BRC_IRQ_EN

Address Offset:	see Appendix B																Initial Value: 0x0000_0000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																			DID_IRQ_EN11	DID_IRQ_EN10	DID_IRQ_EN9	DID_IRQ_EN8	DID_IRQ_EN7	DID_IRQ_EN6	DID_IRQ_EN5	DID_IRQ_EN4	DID_IRQ_EN3	DID_IRQ_EN2	DID_IRQ_EN1	DID_IRQ_EN0	DEST_ERR_IRQ_
Mode	R																			RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Initial Value	0x0000 0																			0	0	0	0	0	0	0	0	0	0	0	0	0

Bit 0 **DEST_ERR_IRQ_EN:** *BRC_DEST_ERR_IRQ* interrupt enable
0 = Disable interrupt, interrupt is not visible outside GTM-IP
1 = Enable interrupt, interrupt is visible outside GTM-IP

Bit 1 **DID_IRQ_EN0:** Enable DID interrupt, see bit 0 for description.
Bit 2 **DID_IRQ_EN1:** Enable DID interrupt, see bit 0 for description.
Bit 3 **DID_IRQ_EN2:** Enable DID interrupt, see bit 0 for description.

Bit 4 **DID_IRQ_EN3:** Enable DID interrupt, see bit 0 for description.
Bit 5 **DID_IRQ_EN4:** Enable DID interrupt, see bit 0 for description.
Bit 6 **DID_IRQ_EN5:** Enable DID interrupt, see bit 0 for description.
Bit 7 **DID_IRQ_EN6:** Enable DID interrupt, see bit 0 for description.
Bit 8 **DID_IRQ_EN7:** Enable DID interrupt, see bit 0 for description.
Bit 9 **DID_IRQ_EN8:** Enable DID interrupt, see bit 0 for description.
Bit 10 **DID_IRQ_EN9:** Enable DID interrupt, see bit 0 for description.
Bit 11 **DID_IRQ_EN10:** Enable DID interrupt, see bit 0 for description.
Bit 12 **DID_IRQ_EN11:** Enable DID interrupt, see bit 0 for description.
Bit 31:13 **Reserved:** Reserved
Note: Read as zero, should be written as zero

4.5.5 Register BRC_IRQ_FORCINT

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_0000**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																			TRG_DID11	TRG_DID10	TRG_DID9	TRG_DID8	TRG_DID7	TRG_DID6	TRG_DID5	TRG_DID4	TRG_DID3	TRG_DID2	TRG_DID1	TRG_DID0	TRG_DEST_ERR
Mode	R																			RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw
Initial Value	0x0000 0000																			0	0	0	0	0	0	0	0	0	0	0	0	0

Bit 0 **TRG_DEST_ERR:** Trigger destination error interrupt.
0 = corresponding bit in status register will not be forced
1 = Assert corresponding field in **BRC_IRQ_NOTIFY** register

Note: This bit is cleared automatically after write.
Note: This bit is write protected by bit RF_PROT of register GTM_CTRL
Bit 1 **TRG_DID0:** Trigger DID interrupt, see bit 0 for description.
Bit 2 **TRG_DID1:** Trigger DID interrupt, see bit 0 for description.
Bit 3 **TRG_DID2:** Trigger DID interrupt, see bit 0 for description.
Bit 4 **TRG_DID3:** Trigger DID interrupt, see bit 0 for description.
Bit 5 **TRG_DID4:** Trigger DID interrupt, see bit 0 for description.
Bit 6 **TRG_DID5:** Trigger DID interrupt, see bit 0 for description.
Bit 7 **TRG_DID6:** Trigger DID interrupt, see bit 0 for description.
Bit 8 **TRG_DID7:** Trigger DID interrupt, see bit 0 for description.
Bit 9 **TRG_DID8:** Trigger DID interrupt, see bit 0 for description.
Bit 10 **TRG_DID9:** Trigger DID interrupt, see bit 0 for description.
Bit 11 **TRG_DID10:** Trigger DID interrupt, see bit 0 for description.
Bit 12 **TRG_DID11:** Trigger DID interrupt, see bit 0 for description.

Bit 31:13 **Reserved:** Reserved
Note: Read as zero, should be written as zero

4.5.6 Register BRC_IRQ_MODE

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_000X**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																														IRQ_MODE	
Mode	R																														RW	
Initial Value	0x0000 0000																														0bXX	

Bit 1:0 **IRQ_MODE**: IRQ mode selection
0b00 = Level mode
0b01 = Pulse mode
0b10 = Pulse-Notify mode
0b11 = Single-Pulse mode
Note: The interrupt modes are described in section 2.5.

Bit 31:2 **Reserved**
Note: Read as zero, should be written as zero

4.5.7 Register BRC_EIRQ_EN

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_0000**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																			DID_EIRQ_EN11	DID_EIRQ_EN10	DID_EIRQ_EN9	DID_EIRQ_EN8	DID_EIRQ_EN7	DID_EIRQ_EN6	DID_EIRQ_EN5	DID_EIRQ_EN4	DID_EIRQ_EN3	DID_EIRQ_EN2	DID_EIRQ_EN1	DID_EIRQ_EN0	DEST_ERR_EIRQ
Mode	R																			RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Initial Value	0x0000_0000																			0	0	0	0	0	0	0	0	0	0	0	0	0

Bit 0 **DEST_ERR_EIRQ_EN:** *BRC_DEST_ERR_EIRQ* error interrupt enable

0 = Disable error interrupt, error interrupt is not visible outside GTM-IP
1 = Enable error interrupt, error interrupt is visible outside GTM-IP

Bit 1 **DID_EIRQ_EN0:** Enable DID interrupt, see bit 0 for description.
0 = Disable error interrupt, error interrupt is not visible outside GTM-IP
1 = Enable error interrupt, error interrupt is visible outside GTM-IP

Bit 2 **DID_EIRQ_EN1:** Enable DID interrupt, see bit 0 for description.
Bit 3 **DID_EIRQ_EN2:** Enable DID interrupt, see bit 0 for description.
Bit 4 **DID_EIRQ_EN3:** Enable DID interrupt, see bit 0 for description.
Bit 5 **DID_EIRQ_EN4:** Enable DID interrupt, see bit 0 for description.
Bit 6 **DID_EIRQ_EN5:** Enable DID interrupt, see bit 0 for description.
Bit 7 **DID_EIRQ_EN6:** Enable DID interrupt, see bit 0 for description.
Bit 8 **DID_EIRQ_EN7:** Enable DID interrupt, see bit 0 for description.
Bit 9 **DID_EIRQ_EN8:** Enable DID interrupt, see bit 0 for description.
Bit 10 **DID_EIRQ_EN9:** Enable DID interrupt, see bit 0 for description.
Bit 11 **DID_EIRQ_EN10:** Enable DID interrupt, see bit 0 for description.
Bit 12 **DID_EIRQ_EN11:** Enable DID interrupt, see bit 0 for description.
Bit 31:13 **Reserved:** Reserved
Note: Read as zero, should be written as zero

4.5.8 Register BRC_RST

Address Offset:	see Appendix B																Initial Value: 0x0000_0000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																															RST
Mode	R																															RAw
Initial Value	0x0000 0000																															0

Bit 0 **RST:** Software reset
0 = No action
1 = Reset BRC
Note: This bit is cleared automatically after write by CPU. The channel registers are set to their reset values and channel operation is stopped immediately.

Bit 31:1 **Reserved:** Reserved
Note: Read as zero, should be written as zero

5 First In First Out Module (FIFO)

5.1 Overview

The FIFO unit is the storage part of the PSM sub-module. The F2A described in chapter 7 and the AFD described in chapter 6 implement the interface part of the FIFO sub-module to the ARU and the AEI bus. Each FIFO unit embeds eight logical FIFOs. These logical FIFOs are configurable in the following manner:

- FIFO size (defines start and end address)
- FIFO operation modes (normal mode or ring buffer operation mode)
- Fill level control / memory region read protection

Each logical FIFO represents a data stream between the sub-modules of the GTM and the microcontroller connected to AFD sub-module (see section 6). The FIFO RAM counts 1K words, where the word size is 29 bit. This gives the freedom to program or receive 24 bit of data together with the five control bits inside an ARU data word.

The FIFO unit provides three ports for accessing its content. One port is connected to the F2A interface, one port is connected to the AFD interface and one port has its own AEI bus interface.

The AFD interface has always the highest priority. Accesses to the FIFO from AFD interface and direct AEI interface in parallel - which means at the same time - is not possible, because both interfaces are driven from the same AEI bus interface of the GTM.

The priority between F2A and direct AEI interface can be defined by software. This can be done by using the register **FIFO[i]_CH[x]_CTRL** for all FIFO channels of the sub-module.

The FIFO is organized as a single RAM that is also accessible through the FIFO AEI interface connected to one of the FIFO ports. To provide the direct RAM access, the RAM is mapped into the address space of the microcontroller. The addresses for accessing the RAM via AEI can be found in [1].

After reset, the FIFO RAM isn't initialized by hardware.
The FIFO channels can be flushed individually. Each of the eight FIFO channels can be used either in normal FIFO operation mode or in ring buffer operation mode.

Beside the possibility of flushing each FIFO channel directly, a write access to FIFO[i]_CH[x]_END_ADDR or to FIFO[i]_CH[x]_START_ADDR will also flush the regarding channel which means that the read and write pointer and also the fill level of the regarding channel will be reset. In consequence of this existing data in the concerned FIFO channel are not longer valid- thereafter the channel is empty.

5.2 Operation Modes

5.2.1 FIFO Operation Mode

In normal FIFO operation mode the content of the FIFO is written and read in first-in first-out order, where the data is destroyed after it is delivered to the system bus or the F2A sub-module (see section 7).

The upper and lower watermark registers (registers **FIFO[i]_CH[x]_UPPER_WM** and **FIFO[i]_CH[x]_LOWER_WM**) are used for controlling the FIFO's fill level. If the fill level falls below the lower watermark or it exceeds the upper watermark, an interrupt signal is triggered by the FIFO sub-module if enabled inside the **FIFO[i]_IRQ_EN**.

The interrupt signals are sent to the Interrupt Concentrator Module (ICM) (see chapter 20). The ICM can also initiate specific DMA transfers.

5.2.2 Ring Buffer Operation Mode

The ring buffer mode can be used to provide a continuous data or configuration stream to the other GTM sub-modules without CPU interaction. In ring buffer mode the FIFO provides a continuous data stream to the F2A sub-module. The first word of the FIFO is delivered first and after the last word is provided by the FIFO to the ARU, the first word can be obtained again.

If in ring buffer mode the read pointer reaches the write pointer it will be set again to the configured start address. So the read pointer always rotates cyclic between the configured start address of the regarding FIFO channel (first written data) and the write pointer which points to the last written data of the channel.

It is possible to add data to the FIFO channel via the AEI to FIFO interface (AFD) using the register **AFD[i]_CH[x]_BUF_ACC** while running in ring buffer mode. The new written data will be added in the next ring buffer cycle. However, the register **AFD[i]_CH[x]_BUF_ACC** should not be read in ring buffer mode.

It is recommended to fill the FIFO channel first before enabling the data stream in the FIFO to ARU interface (F2A).

Modifications of the continuous data stream can be achieved by using direct memory access which is provided by the FIFO AEI interface.

5.2.3 DMA Hysteresis Mode

The DMA hysteresis mode can be enabled by setting bit DMA_HYSTERESIS=1 in the **FIFO[i]_CH[x]_IRQ_MODE** register.

In the DMA hysteresis mode the lower and upper watermark will be masked to generate the DMA request (=*fifo_irq*) in the following manner.

If a DMA is writing data to a FIFO (configured by setting bit DMA_HYST_DIR=1 in register **FIFO[i]_CH[x]_IRQ_MODE**), the DMA request will be generated by the lower watermark. The upper watermark does not generate a DMA request. The next DMA request will be generated by the next lower watermark until the upper watermark was reached.

If a DMA is reading data from a FIFO (configured by setting bit DMA_HYST_DIR=0 in register **FIFO[i]_CH[x]_IRQ_MODE**), the DMA request will be generated by the upper watermark. The lower watermark does not generate a DMA request. The next DMA request will be generated by the next upper watermark until the lower watermark was reached.

Note that the watermarks have to achieve the following condition depending on the irq mode.

- Level / Pulse / Pulse-Notify mode :
  upper watermark > lower watermark

- Single-Pulse mode :
  upper watermark > lower watermark + 1

5.3 FIFO Interrupt Signals

Signal	**Description**
FIFO[i]*_CH[x]_EMPTY*	Indicating empty FIFO x (x:0...7) was reached
FIFO[i]*_CH[x]_FULL*	Indicating full FIFO x (x:0...7) was reached
FIFO[i]*_CH[x]_LOWER_WM*	Indicating FIFO x (x:0...7) reached lower watermark.
FIFO[i]*_CH[x]_UPPER_WM*	Indicating FIFO x (x:0...7) reached upper watermark.

5.4 FIFO Configuration Register Overview

Register Name	Description	Details in Section
FIFO[i]_CH[z]_CTRL (z:0...7)	FIFOi channel z control register	5.5.1
FIFO[i]_CH[z]_END_ADDR (z:0...7)	FIFOi channel z end address register	5.5.2
FIFO[i]_CH[z]_START_ADDR (z:0...7)	FIFOi channel z start address register	5.5.3
FIFO[i]_CH[z]_UPPER_WM (z:0...7)	FIFOi channel z upper watermark register	5.5.4
FIFO[i]_CH[z]_LOWER_WM (z:0...7)	FIFOi channel z lower watermark register	5.5.5
FIFO[i]_CH[z]_STATUS (z:0...7)	FIFOi channel z status register	5.5.6
FIFO[i]_CH[z]_FILL_LEVEL (z:0...7)	FIFOi channel z fill level register	5.5.7
FIFO[i]_CH[z]_WR_PTR (z:0...7)	FIFOi channel z write pointer register	5.5.8
FIFO[i]_CH[z]_RD_PTR (z:0...7)	FIFOi channel z read pointer register	5.5.9
FIFO[i]_CH[z]_IRQ_NOTIFY (z:0...7)	FIFOi channel z interrupt notification register	5.5.10
FIFO[i]_CH[z]_IRQ_EN (z:0...7)	FIFOi channel z interrupt enable register	5.5.11
FIFO[i]_CH[z]_EIRQ_EN (z:0...7)	FIFOi channel z error interrupt enable register	5.5.14
FIFO[i]_CH[z]_IRQ_FORCINT (z:0...7)	FIFOi channel z force interrupt register	5.5.12
FIFO[i]_CH[z]_IRQ_MODE (z:0...7)	FIFOi channel z interrupt mode control register	5.5.13

5.5 FIFO Configuration Registers Description

5.5.1 Register FIFO[i]_CH[z]_CTRL (z:0...7)

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																												WULOCK	FLUSH	RAP	RBM
Mode	R																												RW	RAw	RW	RW
Initial Value	0x0000000																												0	0	0	0

Bit 0 **RBM:** Ring buffer mode enable
0 = Normal FIFO operation mode
1 = Ring buffer mode

Bit 1 **RAP:** RAM access priority
0 = FIFO ports have higher access priority than AEI-IF
1 = AEI-IF has higher access priority than FIFO ports
Note: RAP bit is only functional in register FIFO_0_CTRL. The priority is defined for all FIFO channels there

Bit 2 **FLUSH:** FIFO Flush control
0 = Normal operation
1 = Execute FIFO flush (bit is automatically cleared after flush).
Note: A FIFO Flush operation resets the **FIFO[i]_CH[z]_FILL_LEVEL**, **FIFO[i]_CH[z]_WR_PTR** and **FIFO[i]_CH[z]_RD_PTR** registers to their initial values.

Bit 3 **WULOCK:** RAM write unlock. Enable/disable direct RAM write access to the memory mapped FIFO region.
0 = Direct RAM write access disabled
1 = Direct RAM write access enabled
Note: Only the bit WULOCK of register FIFO[i]_CH0_CTRL enables/disables the direct RAM write access for all FIFO channel (whole FIFO RAM). The WULOCK bits of the other channels are writeable but have no effect.

Bit 31:4 **Reserved:** reserved
Note: read as zero, should be written as zero

5.5.2 Register FIFO[i]_CH[z]_END_ADDR (z:0…7)

Address Offset:	see Appendix B																Initial Value: 0x0000_0XXX															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																						ADDR									
Mode	R																						RW									
Initial Value	0x00000																						0xXXX									

Bit 9:0 **ADDR:** End address for FIFO channel z, (z:0...7)
Note: value for ADDR is calculated as ADDR = 128*(x+1)−1
Note: A write access will flush the regarding channel

Bit 31:10 **Reserved:** reserved
Note: read as zero, should be written as zero

5.5.3 Register FIFO[i]_CH[z]_START_ADDR (z:0...7)

Address Offset:	see Appendix B																Initial Value: 0x0000_0XXX															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																						ADDR									
Mode	R																						RW									
Initial Value	0x00000																						0xXXX									

Bit 9:0 **ADDR:** Start address for FIFO channel z, (z:0...7)
Note: Initial value for ADDR is calculated as ADDR = 128*z
Note: A write access will flush the regarding channel

Bit 31:10 **Reserved:** reserved
Note: read as zero, should be written as zero

5.5.4 Register FIFO[i]_CH[z]_UPPER_WM (z:0...7)

Address Offset:	see Appendix B																Initial Value:								0x0000_0060							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																						ADDR									
Mode	R																						RW									
Initial Value	0x000000																						0x60									

Bit 9:0 **ADDR:** Upper watermark address.

Note: **The upper watermark is configured as a relative fill level of the FIFO.** ADDR must be in range:

0 <= ADDR <= **FIFO[i]_CH[z]_END_ADDR – FIFO[i]_CH[z]_START_ADDR**.

Initial value for ADDR is defined as ADDR = 0x60.

Bit 31:10 **Reserved:** reserved

Note: read as zero, should be written as zero

5.5.5 Register FIFO[i]_CH[z]_LOWER_WM (z:0...7)

Address Offset:	see Appendix B																Initial Value:								0x0000_0020							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																						ADDR									
Mode	R																						RW									
Initial Value	0x000000																						0x20									

Bit 9:0 **ADDR:** Lower watermark address.

Note: **The lower watermark is configured as a relative fill level of the FIFO.** ADDR must be in range:

0 <= ADDR <= **FIFO[i]_CH[z]_END_ADDR – FIFO[i]_CH[z]_START_ADDR**.

Initial value for ADDR is defined as ADDR = 0x20.

Bit 31:10 **Reserved:** reserved
Note: read as zero, should be written as zero

5.5.6 Register FIFO[i]_CH[z]_STATUS (z:0…7)

Address Offset:	see Appendix B																Initial Value:								0x0000_0005							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																												UP_WM	LOW_WM	FULL	EMPTY
Mode	R																												R	R	R	R
Initial Value	0x0000000																												0	1	0	1

Bit 0 **EMPTY:** FIFO is empty.
0 = Fill level > 0
1 = Fill level = 0
Note: Bit only applicable in normal mode

Bit 1 **FULL:** FIFO is full.
0 = Fill level < **FIFO[i]_CH[z]_END_ADDR** – **FIFO[i]_CH[z]_START_ADDR** + 1
1 = Fill level = **FIFO[i]_CH[z]_END_ADDR** – **FIFO[i]_CH[z]_START_ADDR** + 1

Note: Bit only applicable in normal mode

Bit 2 **LOW_WM:** Lower watermark reached
0 = Fill level > lower watermark
1 = Fill level <= lower watermark
Note: Bit only applicable in normal mode

Bit 3 **UP_WM:** Upper watermark reached
0 = Fill level < upper watermark
1 = Fill level >= upper watermark
Note: Bit only applicable in normal mode

Bit 31:4 **Reserved:** reserved
Note: read as zero, should be written as zero

5.5.7 Register FIFO[i]_CH[z]_FILL_LEVEL (z:0…7)

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																					LEVEL										
Mode	R																					R										
Initial Value	0x00000																					0x000										

Bit 10:0 **LEVEL:** Fill level of the current FIFO

Note: LEVEL is in range:

0 ≤ LEVEL ≤ **FIFO[i]_CH[z]_END_ADDR** – **FIFO[i]_CH[z]_START_ADDR** + 1.

Register content is compared to the upper and lower watermark values for this channel to detect watermark over- and underflow.

Bit 31:11 **Reserved:** reserved

Note: read as zero, should be written as zero

5.5.8 Register FIFO[i]_CH[z]_WR_PTR (z:0...7)

Address Offset:	see Appendix B																Initial Value:								0x0000_0XXX							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																						ADDR									
Mode	R																						R									
Initial Value	0x00000																						0xXXX									

Bit 9:0 **ADDR:** Position of the write pointer

Note: ADDR must be in range 0 ≤ ADDR ≤ 1023. Initial value for ADDR is defined as ADDR = **FIFO[i]_CH[z]_START_ADDR**

Bit 31:10 **Reserved:** reserved

Note: read as zero, should be written as zero

5.5.9 Register FIFO[i]_CH[z]_RD_PTR (z:0...7)

Address Offset:	see Appendix B																Initial Value:								0x0000_0XXX							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																						ADDR									
Mode	R																						R									
Initial Value	00																						0xXXX									

Bit 9:0 **ADDR:** Position of the read pointer

Note: ADDR must be in range 0 ≤ ADDR ≤ 1023. Initial value for ADDR is defined as ADDR = **FIFO[i]_CH[z]_START_ADDR**

Bit 31:10 **Reserved:** reserved

Note: read as zero, should be written as zero

5.5.10 Register FIFO[i]_CH[z]_IRQ_NOTIFY (z:0...7)

Address Offset:	see Appendix B																Initial Value:								0x0000_0005							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																												FIFO_UWWM	FIFO_LWM	FIFO_FULL	FIFO_EMPTY
Mode	R																												RCw	RCw	RCw	RCw
Initial Value	0x0000 000																												0	1	0	1

Bit 0 **FIFO_EMPTY:** FIFO is empty

0 = No interrupt occurred.

1 = FIFO is empty interrupt occurred.

Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 1 **FIFO_FULL:** FIFO is full. See bit 0.

Bit 2 **FIFO_LWM:** FIFO Lower watermark was under-run. See bit 0.
Bit 3 **FIFO_UWM:** FIFO Upper watermark was overrun. See bit 0.
Bit 31:4 **Reserved:** reserved
Note: read as zero, should be written as zero

5.5.11 Register FIFO[i]_CH[z]_IRQ_EN (z:0...7)

Address Offset:	see Appendix B																Initial Value: 0x0000_0000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																												FIFO_UWM_IRQ_	FIFO_LWM_IRQ_	FIFO_FULL_IRQ_	FIFO_EMPTY_IRQ
Mode	R																												RW	RW	RW	RW
Initial Value	0x0000 000																												0	0	0	0

Bit 0 **FIFO_EMPTY_IRQ_EN:** interrupt enable
0 = Disable interrupt, interrupt is not visible outside GTM-IP.
1 = Enable interrupt, interrupt is visible outside GTM-IP.

Bit 1 **FIFO_FULL_IRQ_EN:** interrupt enable. See bit 0.
Bit 2 **FIFO_LWM_IRQ_EN:** interrupt enable. See bit 0.
Bit 3 **FIFO_UWM_IRQ_EN:** interrupt enable. See bit 0.
Bit 31:4 **Reserved:** reserved
Note: read as zero, should be written as zero

5.5.12 Register FIFO[i]_CH[z]_IRQ_FORCINT (z:0...7)

Address Offset:	see Appendix B																Initial Value: 0x0000_0000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																												TRG_FIFO_UWM	TRG_FIFO_LWM	TRG_FIFO_FULL	TRG_FIFO_EMPT
Mode	R																												RAw	RAw	RAw	RAw
Initial Value	0x0000000																												0	0	0	0

Bit 0 **TRG_FIFO_EMPTY:** Force interrupt of FIFO empty status.
0 = corresponding bit in status register will not be forced
1 = Assert corresponding field in **FIFO[i]_CH[z]_IRQ_NOTIFY** register

Note: This bit is cleared automatically after write.
Note: This bit is write protected by bit RF_PROT of register GTM_CTRL

Bit 1 **TRG_FIFO_FULL:** Force interrupt of FIFO full status. See bit 0.

Bit 2 **TRG_FIFO_LWM:** Force interrupt of lower watermark. See bit 0.

Bit 3 **TRG_FIFO_UWM:** Force interrupt of upper watermark. See bit 0.

Bit 31:4 **Reserved:** reserved
Note: read as zero, should be written as zero

5.5.13 Register FIFO[i]_CH[z]_IRQ_MODE (z:0…7)

Address Offset:	see Appendix B																Initial Value: 0x0000_000X															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																												DMA_HYST_DIR	DMA_HYSTERESI	IRQ_MODE	
Mode	R																												RW	RW	RW	
Initial Value	0x0000000																												0	0	XX	

Bit 1:0 **IRQ_MODE**: IRQ mode selection
0b00 = Level mode
0b01 = Pulse mode
0b10 = Pulse-Notify mode
0b11 = Single-Pulse mode

Note: The interrupt modes are described in section 2.5.

Bit 2 **DMA_HYSTERESIS**: Enable DMA hysteresis mode.
0 = Disable FIFO hysteresis for DMA access.
1 = Enable FIFO hysteresis for DMA access.

Bit 3 **DMA_HYST_DIR**: DMA direction in hysteresis mode
0 = DMA direction read in hysteresis mode.
1 = DMA direction write in hysteresis mode.

Note: In the case of DMA writing data to a FIFO the DMA requests must be generated by the lower watermark. If the DMA hysteresis is enabled, the FIFO does not generate a new DMA request until the upper watermark is reached.

Note: In the case of DMA reading data from FIFO the DMA requests must be generated by the upper watermark. If the DMA hysteresis is enabled, the FIFO does not generate a new DMA request until the lower watermark is reached.

Bit 31:4 **Reserved**
Note: Read as zero, should be written as zero

5.5.14 Register FIFO[i]_CH[z]_EIRQ_EN (z:0...7)

Address Offset:	see Appendix B																Initial Value:	0x0000_0000														
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																												FIFO_UWM_EIRQ	FIFO_LWM_EIRQ	FIFO_FULL_EIRQ	FIFO_EMPTY_EIR
Mode	R																												RW	RW	RW	RW
Initial Value	0x0000 000																												0	0	0	0

Bit 0 **FIFO_EMPTY_EIRQ_EN:** error interrupt enable
0 = Disable error interrupt, error interrupt is not visible outside GTM-IP.
1 = Enable error interrupt, error interrupt is visible outside GTM-IP.

Bit 1 **FIFO_FULL_EIRQ_EN:** interrupt enable. See bit 0.
Bit 2 **FIFO_LWM_EIRQ_EN:** interrupt enable. See bit 0.
Bit 3 **FIFO_UWM_EIRQ_EN:** interrupt enable. See bit 0.
Bit 31:4 **Reserved:** reserved
Note: read as zero, should be written as zero

6 AEI to FIFO Data Interface (AFD)

6.1 Overview

The AFD sub-module implements a data interface between the AEI bus and the FIFO sub-module, which consists of eight logical FIFO channels.

The AFD sub-module provides one buffer registers that are dedicated to the logical channels of the FIFO. Access to the corresponding FIFO channel is given by reading or writing this buffer registers **AFD[i]_CH[x]_BUF_ACC**.

An AEI write access to the buffer register where the corresponding FIFO channel is full will be ignored. The data will be lost.

An AEI read access to the buffer register where the corresponding FIFO channel is empty will be served with zero data.

6.2 AFD Register overview

Register Name	Description	Details in Section
AFD[i]_CH[z]_BUF_ACC (z:0..7)	AFD i FIFO z buffer access register	6.3.1

6.3 AFD Register description

6.3.1 Register AFD[i]_CH[z]_BUF_ACC (z:0...7)

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_0000**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved			DATA																												
Mode	R			RW																												
Initial Value	0x0			0x0000000																												

Bit 28:0 **DATA:** Read/write data from/to FIFO

Bit 31:29 **Reserved:** reserved

Note: Read as zero, should be written as zero

7 FIFO to ARU Unit (F2A)

7.1 Overview

The F2A is the interface between the ARU and the FIFO sub-module. Since the data width of the ARU (ARU word) is 53 bit (two 24 bit values and five control bits) and the data width of the FIFO is only 29 bit, the F2A has to distribute the data from and to the FIFO channels in a configurable manner.

The data transfer between FIFO and ARU is organized with eight different streams that are connected to the eight different channels of the corresponding FIFO module. A stream represents a data flow from/to ARU to/from the FIFO via the F2A.

The general definition of 'channels' and 'streams' in the ARU context is done in section 2.3.
Each FIFO channel can act as a write stream (data flow from FIFO to ARU) or as a read stream (data flow from ARU to FIFO).

Within these streams the F2A can transmit/receive the lower, the upper or both 24 bit values of the ARU together with the ARU control bits according to the configured transfer modes as described in section 7.2

Each stream can be enabled/disabled separately within the register **F2A[i]_ENABLE**. If a stream will be disabled, the stream data which are stored inside the F2A will be deleted. This is necessary to ensure, that no old data are transferred after enabling a stream.

7.2 Transfer modes

The F2A unit provides several transfer modes to map 29 bit data of the FIFO from/to 53 bit data of the ARU. E.g. it is configurable that the 24 bit FIFO data is written to the lower ARU data entry (means bits 0 to 23) or to the higher 24 bit ARU data entry (means bits 24 to 47). Bits 24 to 28 of the FIFO data entry (the five control bits) are written/read in both cases to/from bits 48 to 52 of the ARU entry.

When both values of the ARU have to be stored in the FIFO the values are stored behind each other inside the FIFO if the FIFO is not full.

If there is only space for one 24 bit data word plus the five control bits, the F2A transfers one part of the 53 bits first and then waits for transferring the second part before new data is requested from the ARU.

When two values from the FIFO have to be written to one ARU location the words have to be located behind each other inside the FIFO.

The transfer to ARU is only established when both parts could be read out of the FIFO otherwise if only one 29 bit word was provided by the FIFO the F2A waits until the second part is available before the data is made available at the ARU.

Figure 7.2.1 shows the data ordering of the FIFO when both ARU values must be transferred between ARU and FIFO.

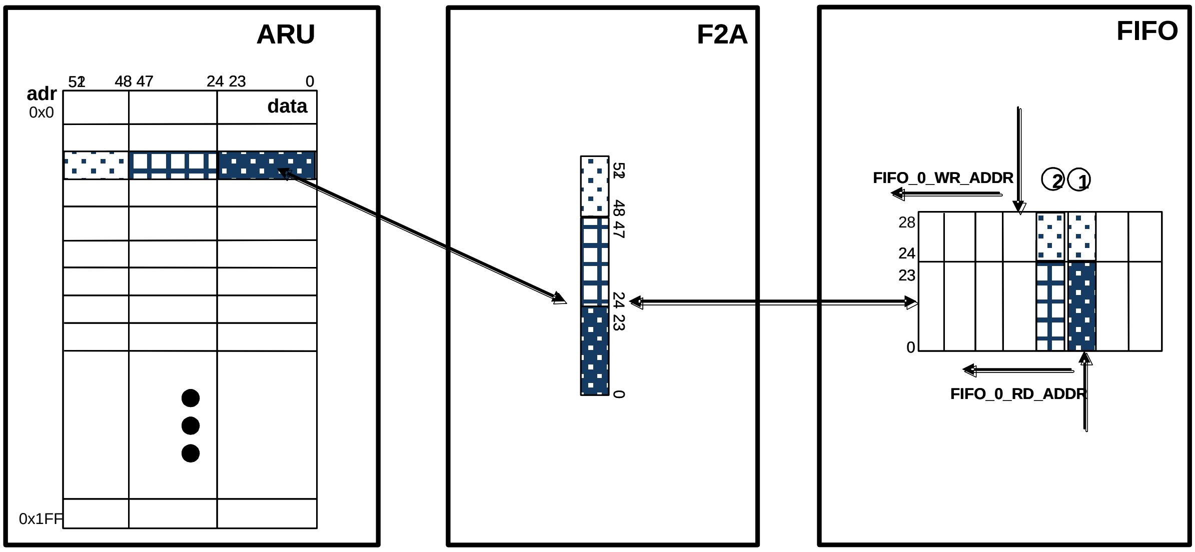
When reading from the ARU the F2A first writes the lower word to the FIFO.
In case of writing to the ARU the F2A reads the lower word first from the FIFO, thus the lower word must be written first to the FIFO through the AFD interface.

Please note, that the five control bits (bits 48 to 52 of the ARU data word) are duplicated as bit 24 to 28 of both FIFO words in case of reading from ARU.

In the case of writing to the ARU, bits 24 to 28 of the last written FIFO word (the higher ARU word) are copied to bits 48 to 52 of the corresponding ARU location.

The transfer modes can be configured with the **TMODE** bits of registers **F2A[i]_CH[x]_STR_CFG** (x: 0..7).

7.2.1 Data transfer of both ARU words between ARU and FIFO

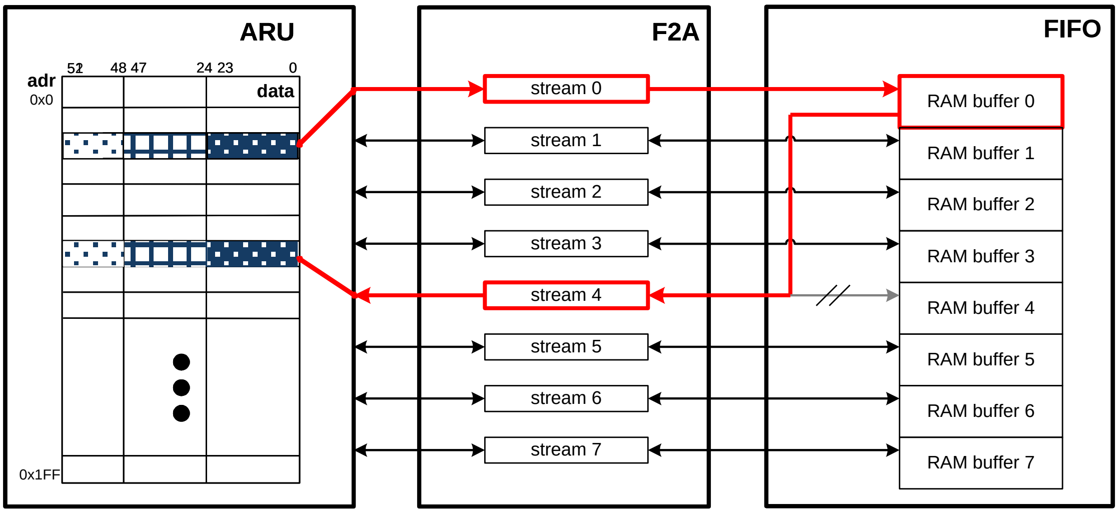


7.3 Internal buffer mode

It is possible to use a FIFO channel as a buffer which is accessed only internally from ARU side. To do this, a read and a write stream of the F2A to one FIFO channel are needed. Therefore it is possible to reconfigure the upper 4 F2A streams 4..7 to the lower 4 FIFO channels 0..3 in the following manner :

F2A stream 4 (+ F2A stream 0) -> FIFO channel 0
F2A stream 5 (+ F2A stream 1) -> FIFO channel 1
F2A stream 6 (+ F2A stream 2) -> FIFO channel 2
F2A stream 7 (+ F2A stream 3) -> FIFO channel 3

7.3.1 Reconfiguration of F2A stream 4 to FIFO channel 0



The configuration of each 4 upper F2A streams can be done separately for each stream in the configuration register **F2A[i]_CTRL**.

Note that the corresponding upper FIFO channel (4..7) cannot be used in this configuration.

7.4 F2A Configuration Register Overview

Register name	Description	Details in Section
F2A[i]_ENABLE	F2Ai stream activation register	7.5.1
F2A[i]_CH[z]_ARU_RD_FIFO (z:0…7)	F2Ai channel z read address register	7.5.2
F2A[i]_CH[z]_STR_CFG (z:0…7)	F2Ai stream z configuration register	7.5.3
F2A[i]_CTRL	F2Ai stream control register	7.5.4

7.5 F2A Configuration Register description

7.5.1 Register F2A[i]_ENABLE

Address Offset:	**see Appendix B**																**Initial Value: 0x0000_0000**															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																STR7_EN		STR6_EN		STR5_EN		STR4_EN		STR3_EN		STR2_EN		STR1_EN		STR0_EN	
Mode	R																RW		RW		RW		RW		RW		RW		RW		RW	
Initial Value	0x0000																00		00		00		00		00		00		00		00	

Bit 1:0 **STR0_EN**: Enable/disable stream 0
Write / Read:
0b00 = Don't care, bits 1:0 will not be changed / Stream disabled
0b01 =Stream 0 is disabled and internal states are reset / --
0b10 = Stream 0 is enabled / --
0b11 = Don't care, bits 1:0 will not be changed / Stream enabled

Note : stream data inside F2A will be deleted on stream disabling.

Bit 3:2 **STR1_EN**: Enable/disable stream 1
See bits 1:0

Bit 5:4 **STR2_EN**: Enable/disable stream 2
See bits 1:0

Bit 7:6 **STR3_EN**: Enable/disable stream 3
See bits 1:0

Bit 9:8 **STR4_EN**: Enable/disable stream 4
See bits 1:0

Bit 11:10 **STR5_EN**: Enable/disable stream 5
See bits 1:0

Bit 13:12 **STR6_EN**: Enable/disable stream 6
See bits 1:0

Bit 15:14 **STR7_EN**: Enable/disable stream 7
See bits 1:0

Bit 31:16 **Reserved**
Note: Read as zero, should be written as zero

7.5.2 Register F2A[i]_CH[z]_ARU_RD_FIFO (z: 0...7)

Address Offset:	see Appendix B																Initial Value:								0x0000_01FE							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																							ADDR								
Mode	R																							RPw								
Initial Value	0x0000 0																							0x1FE								

Bit 8:0 **ADDR**: ARU Read address
Note: this bit field is only writable if channel is disabled.

Bit 31:9 **Reserved**
Note: Read as zero, should be written as zero

7.5.3 Register F2A[i]_CH[z]_STR_CFG (z: 0...7)

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved													DIR	TMODE		Reserved															
Mode	R													RPw	RPw		R															
Initial Value	0x0000													0	00		0x0000															

Bit 15:0 **Reserved**
Note: Read as zero, should be written as zero

Bit 17:16 **TMODE**: Transfer mode for 53 bit ARU data from/to FIFO
0b00 = Transfer low word (ARU bits 23:0) from/to FIFO
0b01 = Transfer high word (ARU bits 47:24) from/to FIFO
0b10 = Transfer both words from/to FIFO
0b11 = Reserved

Bit 18 **DIR**: Data transfer direction
0 = Transport from ARU to FIFO

1 = Transport from FIFO to ARU

Bit 31:19 **Reserved**
Note: Read as zero, should be written as zero

Note: The write protected bits of register **F2A_STR_[x]_CFG** are only writable if the corresponding enable bit STRx_EN of register **F2A_ENABLE** is cleared.

7.5.4 Register F2A[i]_CTRL

Address Offset:	see Appendix B																Initial Value: 0x0000_0000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																								STR7_CONF		STR6_CONF		STR5_CONF		STR4_CONF	
Mode	R																								RPw		RPw		RPw		RPw	
Initial Value	0x0000 00																								00		00		00		00	

Bit 1:0 **STR4_CONF**: Reconfiguration of stream 4 to FIFO channel 0
Write / Read:
0b00 = Don't care, bits 1:0 will not be changed / Stream 4 is mapped to FIFO buffer 4
0b01 = Stream 4 is mapped to FIFO buffer 4 / --
0b10 = Stream 4 is mapped to FIFO buffer 0 / --
0b11 = Don't care, bits 1:0 will not be changed / Stream 4 is mapped to FIFO buffer 0

Note: The write protected bits of register **F2A[i]_CTRL** are only writable if the corresponding enable bit STR0_EN and STR4_EN of register **F2A_ENABLE** is cleared.

Bit 3:2 **STR5_CONF**: Reconfiguration of stream 5 to FIFO channel 1
Write / Read:
0b00 = Don't care, bits 1:0 will not be changed / Stream 5 is mapped to FIFO buffer 5
0b01 = Stream 5 is mapped to FIFO buffer 5 / --
0b10 = Stream 5 is mapped to FIFO buffer 1 / --
0b11 = Don't care, bits 1:0 will not be changed / Stream 5 is mapped to FIFO buffer 1

Note: The write protected bits of register **F2A[i]_CTRL** are only writable if the corresponding enable bit STR1_EN and STR5_EN of register **F2A_ENABLE** is cleared.

Bit 5:4 **STR6_CONF**: Reconfiguration of stream 6 to FIFO channel 2
Write / Read:
0b00 = Don't care, bits 1:0 will not be changed / Stream 6 is mapped to FIFO buffer 6
0b01 = Stream 6 is mapped to FIFO buffer 6 / --
0b10 = Stream 6 is mapped to FIFO buffer 2 / --
0b11 = Don't care, bits 1:0 will not be changed / Stream 6 is mapped to FIFO buffer 2

Note: The write protected bits of register **F2A[i]_CTRL** are only writable if the corresponding enable bit STR2_EN and STR6_EN of register **F2A_ENABLE** is cleared.

Bit 7:6 **STR7_CONF**: Reconfiguration of stream 7 to FIFO channel 3
Write / Read:
0b00 = Don't care, bits 1:0 will not be changed / Stream 7 is mapped to FIFO buffer 7
0b01 = Stream 7 is mapped to FIFO buffer 7 / --
0b10 = Stream 7 is mapped to FIFO buffer 3 / --
0b11 = Don't care, bits 1:0 will not be changed / Stream 7 is mapped to FIFO buffer 3

Note: The write protected bits of register **F2A[i]_CTRL** are only writable if the corresponding enable bit STR3_EN and STR7_EN of register **F2A_ENABLE** is cleared.

Bit 31:8 **Reserved**
Note: Read as zero, should be written as zero

8 Clock Management Unit (CMU)

8.1 Overview

The Clock Management Unit (CMU) is responsible for clock generation of the counters and of the GTM-IP. The CMU consists of three sub-units that generate different clock sources for the whole GTM-IP. The primary clock source for this sub-module is the cluster 0 clock signal *cls0_clk* which is defined by the value of bit field CLS0_CLK_DIV in register **GTM_CLS_CLK_CFG**. Figure 8.1.1 shows a block diagram of the CMU.

The Configurable Clock Generation (CFGU) sub-unit provides eight dedicated clock sources for the following GTM modules: TIM, ATOM, TBU, and MON. Each instance of such a module can choose an arbitrary clock source, in order to specify wide-ranging time bases.

The Fixed Clock Generation (FXU) sub-unit generates predefined non-configurable clocks *CMU_FXCLK[y]* (y: 0..4) for the TOM modules and the MON module. The *CMU_FXCLK[y]* signals are derived from the *CMU_GCLK_EN* signal generated by the Global Clock Divider. The dividing factors are defined as 2^0, 2^4, 2^8, 2^{12}, and 2^{16}.

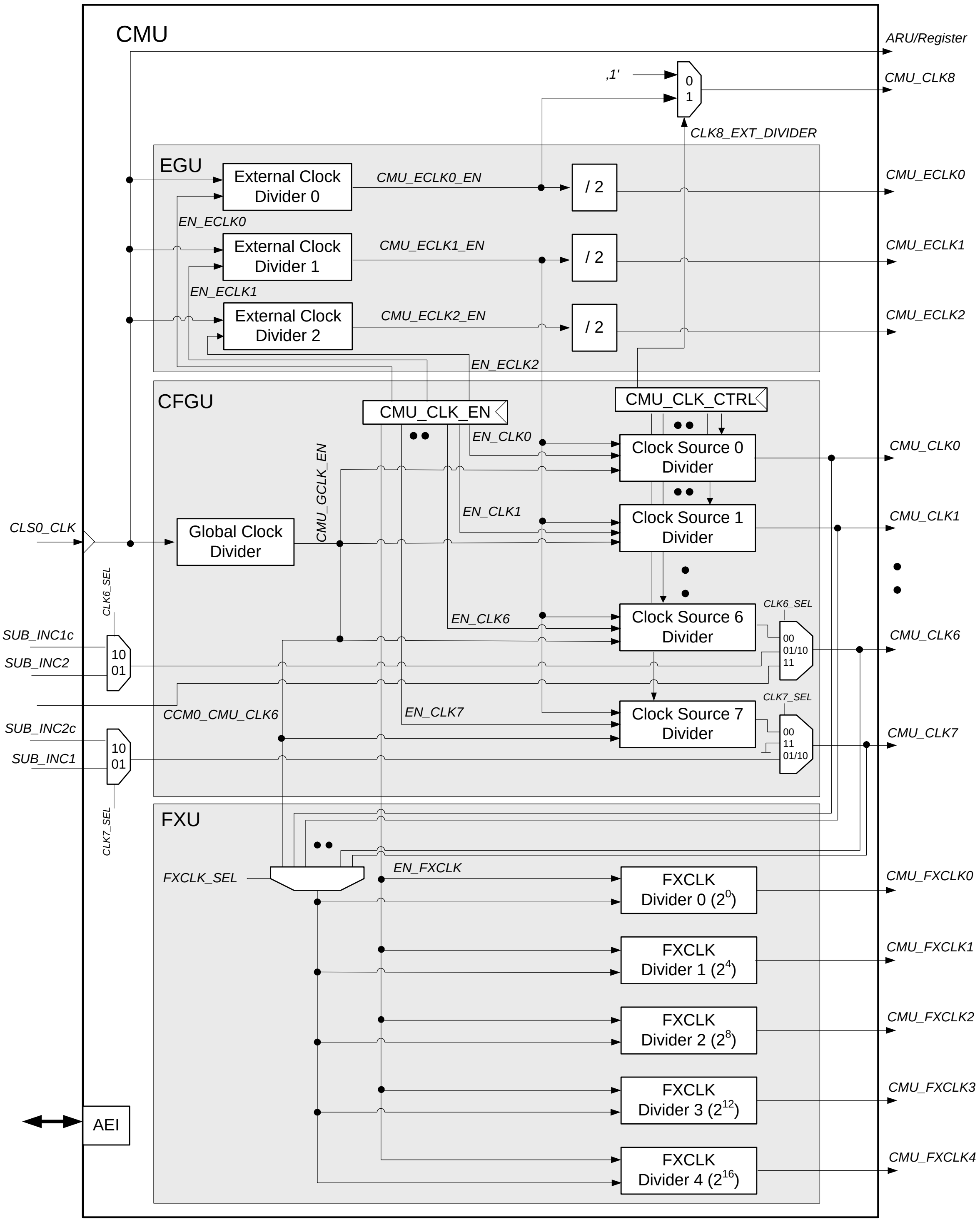
The External Clock Generation (EGU) sub-unit is able to generate up to three chip external clock signals visible at *CMU_ECLK[z]* (z: 0..2) with a duty cycle of about 50%.

The External Clock Generation (EGU) sub-unit is able to generate clock *CMU_CLK8* for module CCM to manage 2 clock domains.

The clock source signals *CMU_CLK[x]* (x: 0..7) and *CMU_FXCLK[y]* are implemented in form of enable signals for the corresponding registers, which means that the actual clock signal of all registers always use the *CLS0_CLK* signal.

The four configurable clock signals *CMU_CLK0*, *CMU_CLK1, CMU_CLK6* and *CMU_CLK7* are used for the TIM filter counters.

8.1.1 CMU Block Diagram



8.2 Global Clock Divider

The sub block Global Clock Divider can be used to divide the CMU primary source signal *CLS0_CLK* into a common subdivided clock signal.

The divided clock signal of the sub block Global Clock Divider is implemented as an enable signal that enables dedicated clocks from the *CLS0_CLK* signal to generate the user specified divided clock frequency.

The resulting fractional divider (*Z/N)* specified through equation:

$$T_{CMU\_GCLK\_EN}=(Z/N)*T_{CLS0\_CLK}$$

is implemented according the following algorithm

(*Z: CMU_GCLK_NUM(23:0) ; N: CMU_GCLK_DEN(23:0)* ; *Z,N >0*):
(1) Set remainder (*R*), operand1 (*OP1*) and operand2 (*OP2*) register during INIT-phase (with implicit conversion to signed):

R=Z, OP1=N, OP2=N-Z;

(2) After leaving INIT-phase (at least one *CMU_CLK[x]* has been enabled) the sign of remainder R for each *CLS0_CLK* cycle will be checked:
(3) If *R>0* keep updating remainder and keep *CMU_GCLK_EN='0'*:

R=R-OP1;

(4) If *R<0* update remainder and set *CMU_GCLK_EN='1'*:

R=R-OP2;

After at most (*Z/N*+1) subtractions (3) there will be a negative *R* and an active phase of the generated clock enable (for one cycle) will be triggered (4). The remainder *R* is a measure for the distance to a real *Z/N* clock and will be regarded for the next generated clock enable cycle phase. The new *R* value will be *R=R+(Z-N)*. In the worst case the remainder *R* will sum up to an additional cycle in the generated clock enable period after *Z*-cycles. In the other cases equally distributed additional cycles will be inserted for the generated clock enable. If *Z* is an integer multiple of *N* no additional cycles will be included for the generated clock enable at all.

Note that for a better resource sharing all arithmetic has been reduced to subtractions and the initialization of the remainder *R* uses the complement of (*Z-N*).

8.3 Configurable Clock Generation sub-unit (CFGU)

The CMU sub-unit CFGU provides up to eight configurable clock divider blocks that divide the common *CMU_GCLK_EN* signal into dedicated enable signals for the GTM-IP sub blocks.

The configuration of the eight different clock signals *CMU_CLK[x]* (x: 0…7) always depends on the configuration of the global clock enable signal *CMU_GCLK_EN*. Additionally, each clock source has its own configuration data, provided by the control register **CMU_CLK_[x]_CTRL** (x: 0…7).

According to the configuration of the Global Clock Divider, the configuration of the Clock Source x Divider is done by setting an appropriate value in the bit field **CLK_CNT[x]** of the register **CMU_CLK_[x]_CTRL**.

The frequency $f_x = 1/T_x$ of the corresponding clock enable signal *CMU_CLK[x]* can be determined by the unsigned representation of **CLK_CNT[x]** of the register **CMU_CLK_[x]_CTRL** in the following way:

$$T_{CMU\_CLK[x]} = (\mathbf{CLK\_CNT[x]}+1) * T_{CMU\_GCLK\_EN}$$

The corresponding wave form is shown in Figure 8.3.1

Each clock signal *CMU_CLK[x]* can be enabled individually by setting the appropriate bit field **EN_CLK[x]** in the register **CMU_CLK_EN**. Except for *CMU_CLK6* and *CMU_CLK7* individual enabling and disabling is active only if **CLK6_SEL** and **CLK7_SEL** is reset.

Alternatively, clock source six and seven (*CMU_CLK6* and *CMU_CLK7*) may provide the signal *SUB_INC1* and *SUB_INC2* coming from module DPLL as clock enable signal depending on the bit field **CLK6_SEL(1:0)** of the register **CMU_CLK_6_CTRL** and on the bit field **CLK7_SEL(1:0)** of the register **CMU_CLK_7_CTRL**.

CMU_CLK8 is switched by **CLK8_EXT_DIVIDER** of the register **CMU_CLK_CTRL** between *CLS0_CLK* and *CMU_ECLK0*.
To switch the clock reference CMU_GCLK_EN with CMU_ECLK1_EN an input selector are used in all Clock Source Divider. The CMU_ECLK1_EN source is enabled by setting the appropriate bit field **CMU[x]_EXT_DIVIDER** in the register **CMU_CLK_CTRL**.

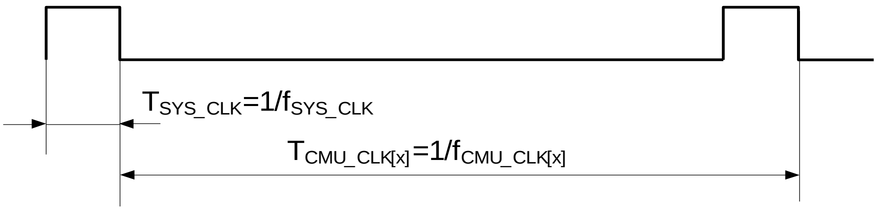
To avoid unexpected behavior of the hardware, the configuration of register **CMU_CLK_[x]_CTRL** and **CMU_CLK_CTRL** can only be changed, when the corresponding clock signal *CMU_CLK[x]* and *CMU_ECLK[1]* is disabled.

Further, any changes to the registers **CMU_GCLK_NUM** and **CMU_GCLK_DEN** can only be performed, when all clock enable signals *CMU_CLK[x]* and the **EN_FXCLK** bit inside the **CMU_CLK_EN** register are disabled.

The clock source signals *CMU_CLK[x]* (x: 0..7) and *CMU_FXCLK[y]* are implemented in form of enable signals for the corresponding registers, which means that the actual clock signal of all registers always use the *CLS0_CLK* signal.

The hardware guarantees that all clock signals *CMU_CLK[x]* (x: 0..7)*,* which were enabled simultaneous, are synchronized to each other. Simultaneous enabling does mean that the bits **EN_CLK[x]** in the register **CMU_CLK_EN** are set by the same write access.

8.3.1 Wave Form of Generated Clock Signal CMU_CLK[x]


T_{SYS_CLK}=1/f_{SYS_CLK}
T_{CMU_CLK[x]}=1/f_{CMU_CLK[x]}

8.4 Fixed Clock Generation (FXU)

The FXU sub-unit generates fixed clock enables out of the CMU_GCLK_EN or one of the eight *CMU_CLK[x]* enable signal depending on the **FXCLK_SEL** bit field of the **CMU_FXCLK_CTRL** register. These clock enables are used for the PWM generation inside the TOM modules.

All clock enables *CMU_FXCLK[y]* can be enabled or disabled simultaneous by setting the appropriate bit field **EN_FXCLK** in the register **CMU_CLK_EN**.

The dividing factors are defined as 2^0, 2^4, 2^8, 2^{12}, and 2^{16}. The signals *CMU_FXCLK[y]* are implemented in form of enable signals for the corresponding registers (see also Chapter 8.3.1)

8.5 External Generation Unit (EGU)

The EGU sub-unit generate up to three separate clock output signals *CMU_ECLK[z]* (z: 0..2).
Each of these clock signals is derived from the corresponding External Clock Divider z sub block, which generates a clock signal derived from the GTM-IP input clock *CLS0_CLK*.

In contrast to the signals *CMU_CLK[x]* and *CMU_FXCLK[y]*, which are treated as simple enable signals for the registers, the signals *CMU_ECLK[z]* have a duty cycle of about 50% that is used as a true clock signal for external peripheral components.

To manage a second global frequency CMU_GCLK_EN could be replaced by ECLK[1]_EN for CMU_CLK[x](x:0..7). Also the all-time enabled CMU_CLK8 could be replaced by ECLK[0]_EN.

Each of the external clocks divider are enabled and disabled by setting the appropriate bit field **EN_ECLK[z]** in the register **CMU_CLK_EN**.

The clock frequencies $f_{CMU\_ECLK[z]} = 1/T_{CMU\_ECLK[z]}$ of the external clocks are controlled with the registers **CMU_ECLK_[z]_NUM** and **CMU_ECLK_[z]_DEN** as follows:

$$T_{CMU\_ECLK[z]\_EN} = (\mathbf{ECLK[z]\_NUM/ECLK[z]\_DEN}) * T_{CLS0\_CLK}$$

and is implemented according the following algorithm
(*Z: CMU_ECLK_[z]_NUM(23:0) ; N: CMU_ECLK_[z]_DEN(23:0) ; Z,N >0 ; Z>=N ; CMU_ECLK[z]*='0'):

(1) Set remainder (*R*), operand1 (*OP1*) and operand2 (*OP2*) register during INIT-phase (with implicit conversion to signed):
R=Z, OP1=N, OP2=N-Z;

(2) After leaving INIT-phase (*CMU_ECLK[z]* has been enabled) the sign of remainder R for each *CLS0_CLK* cycle will be checked:

(3) If *R>0* keep updating remainder and keep *CMU_ECLK[z]*:
R=R-OP1;
(4) If *R<0* update remainder and toggle *CMU_ECLK[z]*:
R=R-OP2;
After at most (*Z/N*+1) subtractions (3) there will be a negative *R* and an active phase of the generated clock enable (for one cycle) will be triggered (4). The remainder *R* is a measure for the distance to a real *Z/N* clock and will be regarded for the next generated clock toggle phase. The new *R* value will be *R=R+(Z-N)*. In the worst case the remainder *R* will sum up to an additional cycle in the generated clock toggle period after *Z*-cycles. In the other cases equally distributed additional cycles will be inserted for the generated clock toggle. If *Z* is an integer multiple of *N* no additional cycles will be included for the generated clock toggle at all.
Note that for a better resource sharing all arithmetic has been reduced to subtractions and the initialization of the remainder *R* uses the complement of (*Z-N*).

The default value of the *CMU_ECLK[z]* output is low.

8.6 CMU Configuration Register Overview

8.6.1 CMU Configuration Register Overview Table

Register Name	**Description**	**Details in Section**
CMU_CLK_EN	CMU clock enable	8.7.1
CMU_GCLK_NUM	CMU global clock control numerator	8.7.2

CMU_GCLK_DEN	CMU global clock control denominator	8.7.3
CMU_CLK_[z]_CTRL (z:0...5)	CMU control for clock source z	8.7.4
CMU_CLK_6_CTRL	CMU control for clock source 6	8.7.5
CMU_CLK_7_CTRL	CMU control for clock source 7	8.7.6
CMU_ECLK_[z]_NUM (z:0...2)	CMU external clock z control numerator	8.7.7
CMU_ECLK_[z]_DEN (z:0...2)	CMU external clock z control denominator	8.7.8
CMU_FXCLK_CTRL	CMU control FXCLK sub-unit input clock	8.7.9
CMU_GLB_CTRL	CMU synchronizing ARU and clock source	8.7.10
CMU_CLK_CTRL	CMU control for clock source selection	8.7.11

8.7 CMU Configuration Register Description

8.7.1 Register CMU_CLK_EN

Address Offset:	see Appendix B																Initial Value: 0x0000_0000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								EN_FXCLK		EN_ECLK2		EN_ECLK1		EN_ECLK0		EN_CLK7		EN_CLK6		EN_CLK5		EN_CLK4		EN_CLK3		EN_CLK2		EN_CLK1		EN_CLK0	
Mode	R								RW		RW		RW		RW		RW		RW		RW		RW		RW		RW		RW		RW	
Initial Value	0x000								00		00		00		00		00		00		00		00		00		00		00		00	

Bit 1:0 **EN_CLK0:** Enable clock source 0
0b00 = clock source is disabled (ignore write access)
0b01 = disable clock signal and reset internal states
0b10 = enable clock signal
0b11 = clock signal enabled (ignore write access)

Note: Any read access to an **EN_CLK[z]**, **EN_ECLK[z]** or **EN_FXCLK** bit field will always result in a value 00 or 11 indicating current state.

A modification of the state is only performed with the values 01 and 10. Writing the values 0b00 and 0b11 is always ignored.

Note: Any disabling to **EN_CLK[x]** will be reset internal counters for configurable clocks.

Bit 3:2 **EN_CLK1:** Enable clock source 1, see bits 1:0
Bit 5:4 **EN_CLK2:** Enable clock source 2, see bits 1:0
Bit 7:6 **EN_CLK3:** Enable clock source 3, see bits 1:0
Bit 9:8 **EN_CLK4:** Enable clock source 4, see bits 1:0
Bit 11:10 **EN_CLK5:** Enable clock source 5, see bits 1:0
Bit 13:12 **EN_CLK6:** Enable clock source 6, see bits 1:0
Bit 15:14 **EN_CLK7:** Enable clock source 7, see bits 1:0
Bit 17:16 **EN_ECLK0:** Enable ECLK 0 generation sub-unit, see bits 1:0
Bit 19:18 **EN_ECLK1:** Enable ECLK 1 generation sub-unit, see bits 1:0
Bit 21:20 **EN_ECLK2:** Enable ECLK 2 generation sub-unit, see bits 1:0
Bit 23:22 **EN_FXCLK:** Enable all CMU_FXCLK, see bits 1:0

Note: An enable to **EN_FXCLK** from disable state will be reset internal fixed clock counters.

Bit 31:24 **Reserved:** Reserved bits

Note: Read as zero, should be written as zero

8.7.2 Register CMU_GCLK_NUM

Address Offset:	**see Appendix B**																**Initial Value: 0x0000_0001**															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								GCLK_NUM																							
Mode	R								RPw																							
Initial Value	0x00								0x000001																							

Bit 23:0 Numerator for global clock divider. Defines numerator of the fractional divider.

Note: Value can only be modified when all clock enables **EN_CLK[x]** and the **EN_FXCLK** are disabled.

Note: The CMU hardware alters the content of **CMU_GCLK_NUM** and **CMU_GCLK_DEN** automatically to 0x1, if **CMU_GCLK_NUM** is specified less than **CMU_GCLK_DEN** or one of the values is specified with a value zero. Thus, a secure way for altering the values is writing twice to the register **CMU_GCLK_NUM** followed by a single write to register **CMU_GCLK_DEN**.

Bit 31:24 Reserved
Note: Read as zero, should be written as zero

8.7.3 Register CMU_GCLK_DEN

Address Offset:	**see Appendix B**																**Initial Value: 0x0000_0001**															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								GCLK_DEN																							
Mode	R								RPw																							
Initial Value	0x00								0x000001																							

Bit 23:0 Denominator for global clock divider. Defines denominator of the fractional divider
Note: Value can only be modified when all clock enables **EN_CLK[x]** and the **EN_FXCLK** are disabled.
Note: The CMU hardware alters the content of **CMU_GCLK_NUM** and **CMU_GCLK_DEN** automatically to 0x1, if **CMU_GCLK_NUM** is specified less than **CMU_GCLK_DEN** or one of the values is specified with a value zero. Thus, a secure way for altering the values is writing twice to the register **CMU_GCLK_NUM** followed by a single write to register **CMU_GCLK_DEN**.

Bit 31:24 Reserved
Note: Read as zero, should be written as zero

8.7.4 Register CMU_CLK_[z]_CTRL (z:0...5)

<table>
<tr><th>Address Offset:</th><th colspan="16">see Appendix B</th><th colspan="16">Initial Value: 0x0000_0000</th></tr>
<tr><td></td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>Bit</td><td colspan="8">Reserved</td><td colspan="24">CLK_CNT</td></tr>
<tr><td>Mode</td><td colspan="8">R</td><td colspan="24">RPw</td></tr>
<tr><td>Initial Value</td><td colspan="8">0x00</td><td colspan="24">0x000000</td></tr>
</table>

Bit 23:0 **CLK_CNT:** Clock count. Defines count value for the clock divider.
Note: Value can only be modified when clock enable **EN_CLK[z]** (z:0...5) and **EN_ECLK1** are disabled.

Bit 31:24 **Reserved:** Reserved bits
Note: Read as zero, should be written as zero

8.7.5 Register CMU_CLK_6_CTRL

<table>
<tr><th>Address Offset:</th><th colspan="16">see Appendix B</th><th colspan="16">Initial Value: 0x0000_0000</th></tr>
<tr><td></td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>Bit</td><td colspan="6">Reserved</td><td colspan="2">CLK6_SEL</td><td colspan="24">CLK_CNT</td></tr>
<tr><td>Mode</td><td colspan="6">R</td><td colspan="2">RPw</td><td colspan="24">RPw</td></tr>
<tr><td>Initial Value</td><td colspan="6">0x00</td><td colspan="2">00</td><td colspan="24">0x000000</td></tr>
</table>

Bit 23:0 **CLK_CNT:** Clock count. Define count value for the clock divider of clock source *CMU_CLK6*.
Note: Value can only be modified when clock enable **EN_CLK6** and **EN_ECLK1** are disabled.

Bit 25:24 **CLK6_SEL:** Clock source selection for *CMU_CLK6*.
0b00 = use Clock Source 6 Divider
0b01 = use signal *SUB_INC2* of module DPLL
0b10 = use signal *SUB_INC1c* of module DPLL
0b11 = use signal *CCM0_CMU_CLK6* of sub-module CCM0

Note: Value can only be modified when clock enable **EN_CLK6** and **EN_ECLK1** are disabled.

Bit 31:26 **Reserved:** Reserved bits

Note: Read as zero, should be written as zero

8.7.6 Register CMU_CLK_7_CTRL

Address Offset:	**see Appendix B**																**Initial Value: 0x0000_0000**															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved						CLK7_SEL		CLK_CNT																							
Mode	R						RPw		RPw																							
Initial Value	0x00						0b000		0x000000																							

Bit 23:0 **CLK_CNT:** Clock count. Define count value for the clock divider of clock source *CMU_CLK7*.

Note: Value can only be modified when clock enable **EN_CLK7** and **EN_ECLK1** are disabled.

Bit 25:24 **CLK7_SEL:** Clock source selection for *CMU_CLK7*.

0b00 = use Clock Source 7 Divider

0b01 = use signal *SUB_INC1* of module DPLL

0b10 = use signal *SUB_INC2c* of module DPLL

0b11 = Reserved, no clock is selected

Note: Value can only be modified when clock enable **EN_CLK7** and **EN_ECLK1** are disabled.

Bit 31:26 **Reserved:** Reserved bits

Note: Read as zero, should be written as zero

8.7.7 Register CMU_ECLK_[z]_NUM (z:0...2)

Address Offset:	see Appendix B																Initial Value:								0x0000_0001							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								ECLK_NUM																							
Mode	R								RPw																							
Initial Value	0x00								0x000001																							

Bit 23:0 Numerator for external clock divider. Defines numerator of the fractional divider.
Note: Value can only be modified when clock enable **EN_ECLK[z]** is disabled.

Note: The CMU hardware alters the content of **CMU_ECLK_[z]_NUM** and **CMU_ECLK_[z]_DEN** automatically to 0x1, if **CMU_ECLK_[z]_NUM** is specified less than **CMU_ECLK_[z]_DEN** or one of the values is specified with a value zero. Thus, a secure way for altering the values is writing twice to the register **CMU_ECLK_[z]_NUM** followed by a single write to register **CMU_ECLK_[z]_DEN**.

Bit 31:24 Reserved
Note: Read as zero, should be written as zero

8.7.8 Register CMU_ECLK_[z]_DEN (z:0…2)

Address Offset:	see Appendix B																Initial Value:								0x0000_0001							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								ECLK_DEN																							
Mode	R								RPw																							
Initial Value	0x00								0x000001																							

Bit 23:0 Denominator for external clock divider. Defines denominator of the fractional divider
Note: Value can only be modified when clock enable **EN_ECLK[z]** is disabled.

Note: The CMU hardware alters the content of **CMU_ECLK_[z]_NUM** and **CMU_ECLK_[z]_DEN** automatically to 0x1, if **CMU_ECLK_[z]_NUM** is specified less than **CMU_ECLK_[z]_DEN** or one of the values is specified with a value zero. Thus, a secure way for altering the values is writing twice to the register **CMU_ECLK_[z]_NUM** followed by a single write to register **CMU_ECLK_[z]_DEN**.

Bit 31:24 Reserved
Note: Read as zero, should be written as zero

8.7.9 Register CMU_FXCLK_CTRL

Address Offset:	**see Appendix B**																**Initial Value: 0x0000_0000**															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																												FXCLK_SEL			
Mode	R																												RPw			
Initial Value	0x0000000																												0x0			

Bit 3:0 **FXCLK_SEL:** Input clock selection for *EN_FXCLK* line.
0b0000 = *CMU_GCLK_EN* selected.
0b0001 = *CMU_CLK0* selected.
0b0010 = *CMU_CLK1* selected.
0b0011 = *CMU_CLK2* selected.
0b0100 = *CMU_CLK3* selected.
0b0101 = *CMU_CLK4* selected.
0b0110 = *CMU_CLK5* selected.
0b0111 = *CMU_CLK6* selected.
0b1000 = *CMU_CLK7* selected.

Note: This value can only be written, when the CMU_FXCLK generation is disabled. See bits 23..22 in register **CMU_CLK_EN**.

Note: Other values for FXCLK_SEL are reserved and should not be used, but they behave like FXCLK_SEL = 0.

Bit 31:4 **Reserved:** Reserved bits
Note: Read as zero, should be written as zero

8.7.10 Register CMU_GLB_CTRL

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																															ARU_ADDR_RST
Mode	R																															RPw
Initial Value	0x0000 0000																															0

Bit 0 **ARU_ADDR_RSTGLB:** Reset ARU caddr counter and ARU dynamic route counter

Note: Writing value 1 to this bit field results in a request to reset the ARU caddr counter and ARU dynamic route counter. The next following write access to register **CMU_CLK_EN** applies the ARU caddr counter reset, ARU dynamic route counter reset and resets this bit.

This feature can be used to synchronize the ARU round trip time to the CMU clocks.

Note: This bit is write protected. Before writing to this bit set bit RF_PROT of register **GTM_CTRL** to 0.

Bit 31:1 **Reserved:** Reserved bits

Note: Read as zero, should be written as zero

8.7.11 Register CMU_CLK_CTRL

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																							CLK8_EXT_DIVID	CLK7_EXT_DIVID	CLK6_EXT_DIVID	CLK5_EXT_DIVID	CLK4_EXT_DIVID	CLK3_EXT_DIVID	CLK2_EXT_DIVID	CLK1_EXT_DIVID	CLK0_EXT_DIVID
Mode	R																							RPw	RPw	RPw	RPw	RPw	RPw	RPw	RPw	RPw
Initial Value	0x0000 00																							0	0	0	0	0	0	0	0	0

Bit 0 **CLK0_EXT_DIVIDER:** Clock source selection for **CMU_CLK_0_CTRL**.

0 = use Clock Source CMU_GCLK_EN
1 = use Clock Source CMU_ECLK1
Note: Value can only be modified when clock enable **EN_CLK0** and **EN_ECLK1** are disabled.

Bit 1 **CLK1_EXT_DIVIDER:** Clock source selection for **CMU_CLK_1_CTRL**.
See bit 0.
Note: Value can only be modified when clock enable **EN_CLK1** and **EN_ECLK1** are disabled.

Bit 2 **CLK2_EXT_DIVIDER:** Clock source selection for **CMU_CLK_2_CTRL**.
See bit 0.
Note: Value can only be modified when clock enable **EN_CLK2** and **EN_ECLK1** are disabled.

Bit 3 **CLK3_EXT_DIVIDER:** Clock source selection for **CMU_CLK_3_CTRL**.
See bit 0.
Note: Value can only be modified when clock enable **EN_CLK3** and **EN_ECLK1** are disabled.

Bit 4 **CLK4_EXT_DIVIDER:** Clock source selection for **CMU_CLK_4_CTRL**.
See bit 0.
Note: Value can only be modified when clock enable **EN_CLK4** and **EN_ECLK1** are disabled.

Bit 5 **CLK5_EXT_DIVIDER:** Clock source selection for **CMU_CLK_5_CTRL**.
See bit 0.
Note: Value can only be modified when clock enable **EN_CLK5** and **EN_ECLK1** are disabled.

Bit 6 **CLK6_EXT_DIVIDER:** Clock source selection for **CMU_CLK_6_CTRL**.
See bit 0.
Note: Value can only be modified when clock enable **EN_CLK6** and **EN_ECLK1** are disabled.

Bit 7 **CLK7_EXT_DIVIDER:** Clock source selection for **CMU_CLK_7_CTRL**.
See bit 0.
Note: Value can only be modified when clock enable **EN_CLK7** and **EN_ECLK1** are disabled.

Bit 8 **CLK8_EXT_DIVIDER:** Clock source selection for *CMU_CLK8.*
0 = use Clock Source CLS0_CLK
1 = use Clock Source CMU_ECLK0
Note: Value can only be modified when **EN_ECLK0** is disabled.

Bit 31:9 **Reserved:** Reserved bits
Note: Read as zero, should be written as zero

9 Cluster Configuration Module (CCM)

9.1 Overview

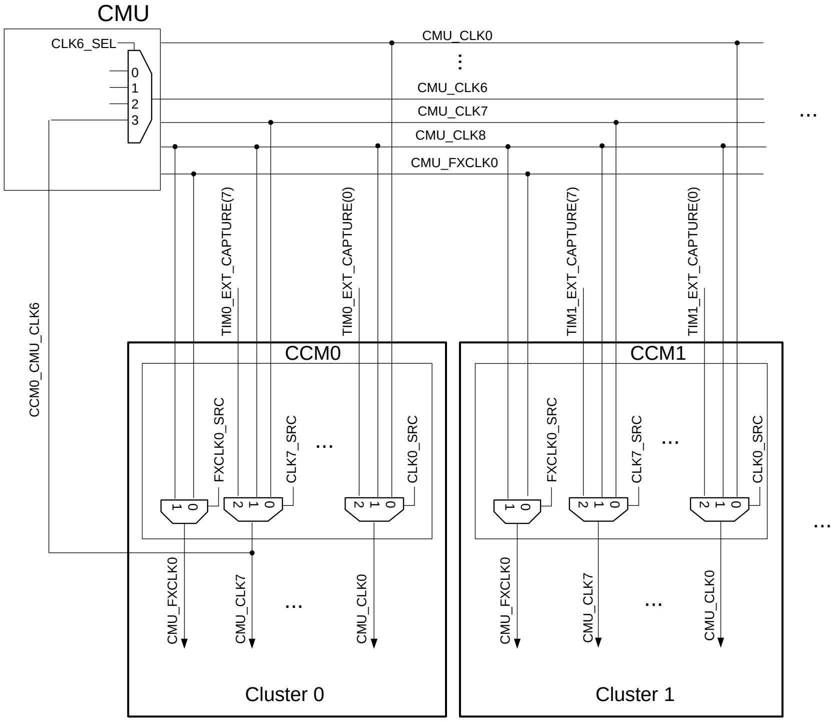
As already mentioned in chapter 2, each submodule of the GTM is aligned explicitly to a cluster. The Cluster Configuration Module (CCM) enables the configuration of several cluster specific options namely

- cluster's clock frequency,
- module clock gating,
- Status observation of the cluster's MCS bus master (AEM),
- address range protection, and
- global architecture configuration.

The register **CCM[i]_CFG** allows disabling the system clock signal for unused sub modules of the i-th cluster. The registers **CCM[i]_CMU_CLK_CFG** and **CCM[i]_CMU_FXCLK_CFG** allows the configuration of various cluster clock frequencies.

Figure 9.1.1 shows important details about the wiring of the cluster's local clock signals.

9.1.1 Cluster Clock Signal Wiring



The register **CCM[i]_AEIM_STA** captures the address and the reason of the first invalid AEIM bus master access of the cluster's MCS module.

The registers **CCM[i]_HW_CONF**, **CCM[i]_AUX_IN_SRC**, **CCM[i]_EXT_CAP_EN**, **CCM[i]_TOM_OUT**, and **CCM[i]_ATOM_OUT** are global status and configuration registers that are mirrored from the group of TOP-Level registers. The intention of these registers is to bring up cluster specific configuration registers into the address space of the bus master of the cluster's MCS module.

9.2 Address Range Protection

The CCM also provides up to NARP so called address range protectors (ARPs), where the number NARP depends on the actual device configuration (defined in Appendix B). An ARP can be used to define a configurable write protected address range in order to support enhanced safety features. The address width of and ARP is also device dependent and it is determined by the parameter AAW as defined in Appendix B.

The protected address range is mapped to the address range of the cluster's MCS RAM port.
Each ARP z (with z = 0 ... NARP-1) can be configured by the registers **CCM[i]_ARP[z]_CTRL** and **CCM[i]_ARP[z]_PROT**, where the register **CCM[i]_ARP[z]_CTRL** enables to configure the size and base address offset for an ARP and the register **CCM[i]_ARP[z]_PROT** configures, that an MCS channel x with a set bit field **WPROTx** cannot write to the corresponding z-th ARP. Whenever an MCS channel x is writing to an ARP that does not allow a write access from channel x by the configuration register **CCM[i]_ARP[z]_PROT**, the write access is discard. The bit field

WPROT_AEI of register **CCM[i]_ARP[z]_CTRL** allows to configure if a CPU write access (via AEI slave interface) to the z-th ARP is protected. If the CPU wants to write to the z-th ARP while **WPROT_AEI** is set, the write access will be discard and the AEI status signal will signalize an invalid module access.

Considering the size and base address of an ARP, it should be noted that the configuration possibilities are limited. Details about the configuration can be found in the register description of **MCS[i]_ARP[z]_CTRL**.

The bit field **DIS_PROT** of register **CCM[i]_ARP[z]_CTRL** changes the meaning of an ARP configuration in a way that it explicitly allows an MCS channel x with a set bit field **WPROTx** to write to the z-th ARP. Accordingly, if the bit **DIS_PROT** is set while the bit **WPROT_AEI** is also set in the register **CCM[i]_ARP[z]_CTRL**, the z-th ARP explicitly allows a write access from the CPU to the z-th ARP. A meaningful application of an ARP z with a set bit field **DIS_PROT** for an MCS channel x has another ARP with a surrounding wider address range that is defining a write protection for MCS channel x and some other MCS channels.

Since the address range of an ARP can surround another ARP it is possible to configure contradictory conditions for MCS channels or the CPU within the overlapping area (e.g. if ARP y surrounds ARP z and ARP y allows a write access for an MCS channel x but ARP z prohibits a write access for MCS channel x). In order to resolve this ambiguity, the following rule is defined: A write protection for a specific address c concerning MCS channel x (the CPU) is active, if and only if, address c is covered by at least one ARP with a cleared bit **DIS_PROT** and a set bit **WPROTx** (**WPROT_AEI**) and there exists no ARP covering address c with a set bit field **DIS_PROT** and a set bit field **WPROTx** (**WPROT_AEI**).

9.3 CCM Configuration Register Overview

9.3.1 CCM Configuration Register Overview Table

Register name	Description	Details in Section
CCM[i]_PROT	CCMi Protection Register	9.4.1
CCM[i]_CFG	CCMi Configuration Register	9.4.2
CCM[i]_CMU_CLK_CFG	CCMi CMU Clock Configuration Register	9.4.3
CCM[i]_CMU_FXCLK_CFG	CCMi CMU Fixed Clock Configuration Register	9.4.4
CCM[i]_AEIM_STA	CCMi MCS Bus Master Status Register	9.4.5

CCM[i]_ARP[z]_CTRL (z:0...NARP-1)	CCMi Address Range Protector z Control Register	9.4.6
CCM[i]_ARP[z]_PROT (z:0...NARP-1)	CCMi Address Range Protector z Protection Register	9.4.7
CCM[i]_HW_CONF	CCMi Hardware Configuration Register	9.4.8
CCM[i]_TIM_AUX_IN_SRC	CCMi TIM AUX input source Register.	9.4.9
CCM[i]_EXT_CAP_EN	CCMi External Capture Enable Register.	9.4.10
CCM[i]_TOM_OUT	CCMi TOM Output Register.	9.4.11
CCM[i]_ATOM_OUT	CCMi ATOM Output Register.	9.4.12

9.4 CCM Configuration Register description

9.4.1 Register CCM[i]_PROT

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_0001**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																															CLS_PROT
Mode																																RW
Initial Value	0x0000000																															1

Bit 0 **CLS_PROT**: Cluster Protection
0 = Write Protection of cluster configuration registers disabled.
1 = Write Protection of cluster configuration registers enabled.

Bit 31:1 **Reserved:** Read as zero, should be written as zero.

9.4.2 Register CCM[i]_CFG

Address Offset:	see Appendix B																Initial Value: 0x000X_00FF															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	TBU_DIR2	TBU_DIR1	Reserved												CLS_CLK_DIV		Reserved								EN_CMP_MON	EN_PSM	EN_BRC	EN_DPLL_MAP	EN_MCS	EN_ATOM_ADTM	EN_TOM_SPE_TD	EN_TIM
Mode	R	R													R										RPw	RPw	RPw	RPw	RPw	RPw	RPw	RPw
Initial Value	0b0	0b0	0x0000												0bxx		0x00								X	X	X	X	X	X	X	X

Bit 0 **EN_TIM:** Enable TIM

0 = Disable clock signal for sub module TIM.

1 = Enable clock signal for sub module TIM.

NOTE: This bit is only writable if bit field **CLS_PROT** of register **CCM[i]_PROT** is cleared.

Bit 1 **EN_TOM_SPE_TDTM:** Enable TOM, SPE and TDTM

0 = Disable clock signal for modules TOM, SPE, and its related DTM modules.

1 = Enable clock signal for modules TOM, SPE, and its related DTM modules.

NOTE: This bit is only writable if bit field **CLS_PROT** of register **CCM[i]_PROT** is cleared.

Bit 2 **EN_ATOM_ADTM:** Enable ATOM and ADTM

0 = Disable clock signal for modules ATOM and its related DTM modules.

1 = Enable clock signal for modules ATOM and its related DTM modules.

NOTE: This bit is only writable if bit field **CLS_PROT** of register **CCM[i]_PROT** is cleared.

Bit 3 **EN_MCS** Enable MCS

0 = Disable clock signal for module MCS.

1 = Enable clock signal for module MCS.

NOTE: This bit is only writable if bit field **CLS_PROT** of register **CCM[i]_PROT** is cleared.

Bit 4 **EN_DPLL_MAP:** Enable DPLL and MAP

0 = Disable clock signal for modules DPLL and MAP.

1 = Enable clock signal for modules DPLL and MAP.

NOTE: This bit is only writable if bit field **CLS_PROT** of register **CCM[i]_PROT** is cleared.

Bit 5 **EN_BRC:** Enable BRC
0 = Disable clock signal for module BRC.
1 = Enable clock signal for module BRC.
NOTE: This bit is only writable if bit field **CLS_PROT** of register **CCM[i]_PROT** is cleared.

Bit 6 **EN_PSM:** Enable PSM
0 = Disable clock signal for module PSM.
1 = Enable clock signal for module PSM.
NOTE: This bit is only writable if bit field **CLS_PROT** of register **CCM[i]_PROT** is cleared.

Bit 7 **EN_CMP_MON:** Enable CMP and MON
0 = Disable clock signal for modules CMP and MON.
1 = Enable clock signal for modules CMP and MON.
NOTE: This bit is only writable if bit field **CLS_PROT** of register **CCM[i]_PROT** is cleared.

Bit 15:8 **Reserved:** Read as zero, should be written as zero.
Bit 17:16 **CLS_CLK_DIV:** Cluster Clock Divider.
0b00 = Cluster is disabled
0b01 = Cluster is enabled without clock divider
0b10 = Cluster is enabled with clock divider 2
0b11 = Reserved

NOTE: The value of this bit field mirrors the bit field **CLS[i]_CLK_DIV** of register **GTM_CLS_CLK_CFG**, whereas i equals the cluster index.

Bit 29:18 **Reserved:** Read as zero, should be written as zero.
Bit 30 **TBU_DIR1:** DIR1 input signal of module TBU.
0b0 = indicating forward direction
0b1 = indicating backward direction

Bit 31 **TBU_DIR2:** DIR2 input signal of module TBU.
0b0 = indicating forward direction
0b1 = indicating backward direction

NOTE: The module specific clock enable registers (bit field **EN_***) are only implemented if the corresponding module is available in the i-th cluster.

9.4.3 Register CCM[i]_CMU_CLK_CFG

Address Offset:	see Appendix B								Initial Value:	0x0000_0000						
	31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16	15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
Bit	Reserved	CLK7_SRC	Reserved	CLK6_SRC	Reserved	CLK5_SRC	Reserved	CLK4_SRC	Reserved	CLK3_SRC	Reserved	CLK2_SRC	Reserved	CLK1_SRC	Reserved	CLK0_SRC
Mode	R	RPw	R	RPw	R	RPw	R	RPw	R	RPw	R	RPw	R	RPw	R	RPw
Initial Value	0b00	0b00	0b00	0b00	0b00	0b00	0b00	0b00	0b00	0b00	0b00	0b00	0b00	0b00	0b00	0b00

Bit 1:0 **CLK0_SRC:** Clock 0 source signal selector

0b00 = Use CMU_CLK0 signal of CMU as CMU_CLK0 signal within Cluster.

0b01 = Use CMU_CLK8 signal of CMU as CMU_CLK0 signal within Cluster.

0b10 = Use TIM[i]_EXT_CAPTURE(0) signal as CMU_CLK0 signal within cluster.

0b11 = Reserved

Bit 3:2 **Reserved:** Read as zero, should be written as zero.

Bit 5:4 **CLK1_SRC:** Clock 1 source signal selector

0b00 = Use CMU_CLK1 signal of CMU as CMU_CLK1 signal within Cluster.

0b01 = Use CMU_CLK8 signal of CMU as CMU_CLK1 signal within Cluster.

0b10 = Use TIM[i]_EXT_CAPTURE(1) signal as CMU_CLK1 signal within cluster.

0b11 = Reserved

Bit 7:6 **Reserved:** Read as zero, should be written as zero.

Bit 9:8 **CLK2_SRC:** Clock 2 source signal selector

0b00 = Use CMU2_CLK signal of CMU as CMU_CLK2 signal within Cluster.

0b01 = Use CMU_CLK8 signal of CMU as CMU_CLK2 signal within Cluster.

0b10 = Use TIM[i]_EXT_CAPTURE(2) signal as CMU_CLK2 signal within cluster.

0b11 = Reserved

Bit 11:10 **Reserved:** Read as zero, should be written as zero.

Bit 13:12 **CLK3_SRC:** Clock 3 source signal selector

0b00 = Use CMU_CLK3 signal of CMU as CMU_CLK3 signal within Cluster.
0b01 = Use CMU_CLK8 signal of CMU as CMU_CLK3 signal within Cluster.
0b10 = Use TIM[i]_EXT_CAPTURE(3) signal as CMU_CLK3 signal within cluster.
0b11 = Reserved

Bit 15:14 **Reserved:** Read as zero, should be written as zero.
Bit 17:16 **CLK4_SRC:** Clock 4 source signal selector
0b00 = Use CMU_CLK4 signal of CMU as CMU_CLK4 signal within Cluster.
0b01 = Use CMU_CLK8 signal of CMU as CMU_CLK4 signal within Cluster.
0b10 = Use TIM[i]_EXT_CAPTURE(4) signal as CMU_CLK4 signal within cluster.
0b11 = Reserved

Bit 19:18 **Reserved:** Read as zero, should be written as zero.
Bit 21:20 **CLK5_SRC:** Clock 5 source signal selector
0b00 = Use CMU_CLK5 signal of CMU as CMU_CLK5 signal within Cluster.
0b01 = Use CMU_CLK8 signal of CMU as CMU_CLK5 signal within Cluster.
0b10 = Use TIM[i]_EXT_CAPTURE(5) signal as CMU_CLK5 signal within cluster.
0b11 = Reserved

Bit 23:22 **Reserved:** Read as zero, should be written as zero.
Bit 25:24 **CLK6_SRC:** Clock 6 source signal selector
0b00 = Use CMU_CLK6 signal of CMU as CMU_CLK6 signal within Cluster.
0b01 = Use CMU_CLK8 signal of CMU as CMU_CLK6 signal within Cluster.
0b10 = Use TIM[i]_EXT_CAPTURE(6) signal as CMU_CLK6 signal within cluster.
0b11 = Reserved

Bit 27:26 **Reserved:** Read as zero, should be written as zero.
Bit 29:28 **CLK7_SRC:** Clock 7 source signal selector
0b00 = Use CMU_CLK7 signal of CMU as CMU_CLK7 signal within Cluster.

0b01 = Use CMU_CLK8 signal of CMU as CMU_CLK7 signal within Cluster.
0b10 = Use TIM[i]_EXT_CAPTURE(7) signal as CMU_CLK7 signal within cluster.
0b11 = Reserved

Bit 31:30 **Reserved:** Read as zero, should be written as zero.

NOTE: The bit fields of this register are only writable if bit field **CLS_PROT** of register **CCM[i]_PROT** is cleared.

9.4.4 Register CCM[i]_CMU_FXCLK_CFG

Address Offset:	see Appendix B																Initial Value: 0x0000_0000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																												FXCLK0_SRC			
Mode																													RPw			
Initial Value	0x0000000																												0x0			

Bit 3:0 **FXCLK0_SRC:** Fixed clock 0 source signal selector
0 = Use CMU_FXCLK0 signal of CMU as CMU_FXCLK0 signal within Cluster.
1 = Use CMU_CLK8 signal of CMU as CMU_FXCLK0 signal within Cluster.

NOTE: Bit field values that are not mentioned above are reserved.

NOTE: These bits are only writable if bit field **CLS_PROT** of register **CCM[i]_PROT** is cleared.

Bit 31:4 **Reserved:** Read as zero, should be written as zero.

9.4.5 Register CCM[i]_AEIM_STA

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved						AEIM_XPT_STA		Reserved								AEIM_XPT_ADDR															
Mode							RAw										RAw															
Initial Value	0x00						0b00		0x000								0x0000															

Bit 15:0 **AEIM_XPT_ADDR**: Exception Address.
Invalid bus master (AEIM) address of MCS module.

Bit 23:16 **Reserved:** Read as zero, should be written as zero.

Bit 25:24 **AEIM_XPT_STA:** AEIM Exception status.
0b00 = No invalid MCS bus master access occurred
0b01 = Invalid byte addressing of MCS bus master access.
0b10 = Illegal module access of MCS bus master access.
0b11 = Invalid MCS bus master access to an unsupported address.

Bit 31:26 **Reserved:** Read as zero, should be written as zero.

NOTE: Only the first invalid AEIM bus master access of the MCS is updating this register with the invalid AEIM address (bit field **AEIM_XPT_ADDR**) and the reason of the invalid access (bit field **AEIM_XPT_STA**). A write access to this register (independent of the written data), always resets the bit fields **AEIM_XPT_STA** and **AEIM_XPT_ADDR** and the next invalid AEIM access is captured by this register, again.

NOTE: If the i-th cluster does not provide an MCS module, this register is not available.

9.4.6 Register CCM[i]_ARP[z]_CTRL (z: 0..NARP-1)

Address Offset:	**see Appendix B**																**Initial Value: 0x0003_0000**															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	WPROT_AEI	Reserved						DIS_PROT	Reserved				SIZE				ADDR															
Mode	RPw	R						RPw	R				RPw				RPw															
Initial Value	0	0x00						0	0x0				0x3				0x0000															

Bit 15:0 **ADDR**: ARP base address.

Base address for address range protector z.

Note: Only the bits 5 to AAW-1 of this bit field are implemented as registers. The bits AAW to 15 are reserved bits and always read and written as zeros. The bits 0 to 4 are functionally used for the definition of an ARP but they are always read and written as zeros.

Note: The actual base address for a protected address range is only defined by the upper AAW-(**SIZE**+2) bits (bit position 2+**SIZE** to bit position AAW-1) of bit field **ADDR**. The lower **SIZE**+2 bits (bit 0 to **SIZE**+1) are ignored for the address calculation and assumed as zeros.

NOTE: This bit field is only writable if bit field **CLS_PROT** of register **CCM[i]_PROT** is cleared.

Bit 19:16 **SIZE:** Size of ARP

Size of memory range protector z.

Note: The actual size of a protected memory range is defined as $2^{\mathbf{SIZE}}$ address locations, whereas the bit field **SIZE** is interpreted as an unsigned integer number.

Note: The values 0, 1 and 2 are not supported for this bit field and cannot be configured. A write access with such a value to this register signalizes an invalid modue access at the AEI status signal and it sets the bit field **SIZE** to the value 3.

NOTE: This bit field is only writable if bit field **CLS_PROT** of register **CCM[i]_PROT** is cleared.

Bit 23:20 **Reserved:** Reserved

Bit 24 **DIS_PROT**: Disable ARP protection.

0 = Bit **WPROTx** (**WPROT_AEI**) defines write protection for selected address range.

1 = Bit **WPROTx** (**WPROT_AEI**) explicitly allows write access to selected address range.

NOTE: This bit field is only writable if bit field **CLS_PROT** of register **CCM[i]_PROT** is cleared.

Bit 30:25 **Reserved:** Reserved

Bit 31 **WPROT_AEI:** AEI slave write protection.
0 = Write protection to address range from AEI slave is disabled.
1 = Write protection to address range from AEI slave is enabled.

Note: The address range interval that is protected by this ARP can be calculated as [(**ADDR** AND NOT 4*($2^{\textbf{SIZE}}$-1)); (**ADDR** AND NOT 4*($2^{\textbf{SIZE}}$-1)) + 4*($2^{\textbf{SIZE}}$-1)] assuming a byte wise addressing, an unsigned integer representation for the bit fields **SIZE** and **ADDR**. NOT and AND are bitwise logical operators. The incrementation interval for neighboring memory location is always 4.

NOTE: This bit field is only writable if bit field **CLS_PROT** of register **CCM[i]_PROT** is cleared.

9.4.7 Register CCM[i]_ARP[z]_PROT (z:0...NARP-1)

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_0000**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																								WPROT7	WPROT6	WPROT5	WPROT4	WPROT3	WPROT2	WPROT1	WPROT0
Mode	R																								RPw	RPw	RPw	RPw	RPw	RPw	RPw	RPw
Initial Value	0x0000 00																								0	0	0	0	0	0	0	0

Bit 0 **WPROT0:** Write Protection MCS channel 0.
0 = Write protection to ARP's address range for MCS channel 0 is disabled.
1 = Write protection to ARP's address range for MCS channel 0 is enabled.

Bit 1 **WPROT1:** Write Protection MCS channel 1.
0 = Write protection to ARP's address range for MCS channel 1 is disabled.
1 = Write protection to ARP's address range for MCS channel 1 is enabled.

Bit 2 **WPROT2:** Write Protection MCS channel 2.
0 = Write protection to ARP's address range for MCS channel 2 is disabled.
1 = Write protection to ARP's address range for MCS channel 2 is enabled.

Bit 3 **WPROT3:** Write Protection MCS channel 3.
0 = Write protection to ARP's address range for MCS channel 3 is disabled.
1 = Write protection to ARP's address range for MCS channel 3 is enabled.

Bit 4 **WPROT4:** Write Protection MCS channel 4.
0 = Write protection to ARP's address range for MCS channel 4 is disabled.
1 = Write protection to ARP's address range for MCS channel 4 is enabled.

Bit 5 **WPROT5:** Write Protection MCS channel 5.
0 = Write protection to ARP's address range for MCS channel 5 is disabled.
1 = Write protection to ARP's address range for MCS channel 5 is enabled.

Bit 6 **WPROT6:** Write Protection MCS channel 6.
0 = Write protection to ARP's address range for MCS channel 6 is disabled.
1 = Write protection to ARP's address range for MCS channel 6 is enabled.

Bit 7 **WPROT7:** Write Protection MCS channel 7.
0 = Write protection to ARP's address range for MCS channel 7 is disabled.
1 = Write protection to ARP's address range for MCS channel 7 is enabled.

Bit 31:8 **Reserved:** Reserved
Note: Only the first T bits of this register (bit 0 to T-1) are functionally implemented. The other bits (bit T to 31) are reserved bits. Parameter T reflects the number of available MCS channels in the cluster's MCS module.

NOTE: This bit fields of this register are only writable if bit field **CLS_PROT** of register **CCM[i]_PROT** is cleared.

NOTE: The meaning of the bit fields **WPROTx** can be changed by the bit field **DIS_PROT** of register **CCM[i]_ARP[z]_CTRL**.

9.4.8 Register CCM[i]_HW_CONF

Address Offset:	see Appendix B																Initial Value: 0xXXXX_XXXX															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved		INT_CLK_EN_GE	TOM_TRIG_INTCHAIN					ATOM_TRIG_INTCHAIN				IRQ_MODE_SING	IRQ_MODE_PULS	IRQ_MODE_PULS	IRQ_MODE_LEVE	Reserved	ARU_CONNECT_	ERM	RAM_INIT_RST	TOM_TRIG_CHAIN			TOM_OUT_RST	ATOM_TRIG_CHAIN			ATOM_OUT_RST	CFG_CLOCK_RAT	SYNC_INPUT_RE	BRIDGE_MODE_R	GRSTEN
Mode	R		R	R					R				R	R	R	R	R	R	R	R	R			R	R			R	R	R	R	R
Initial Value	0b00		X	0xXX					0xX				X	X	X	X	0	X	X	X	0xX			X	0xX			X	X	X	X	X

Bit 0 **GRSTEN**: Global Reset Enable
0 = Global GTM reset register disabled
1 = Global GTM reset register enabled

Bit 1 **BRIDGE_MODE_RST**: Bridge mode after reset
0 = Bridge starts in synchronous mode after reset
1 = Bridge starts in asynchronous mode after reset

Bit 2 **SYNC_INPUT_REG:** additional pipelined stage in synchronous bridge mode
0 = No additional pipelined stage implemented.
1 = additional pipelined stage implemented. All accesses in synchronous mode will be increased by one clock cycle.
Note: this register is only relevant (if existing) for synchronous bridge mode

Bit 3 **CFG_CLOCK_RATE**: clocks per ARU transfer
0 = Each system clock an ARU transfer is scheduled
1 = Each second system clock an ARU transfer is scheduled. ARU transfer rate is half the system clock frequency.

Note: This value defines also the availability of configuration bits in register GTM_CLS_CLK_CFG.
if CFG_CLOCK_RATE=0, only the values 0b00 and 0b01 are valid for bit fields CLS[x]x_CLK_DIV.
if CFG_CLOCK_RATE=1, only the values 0b00, 0b01 and 0b10 are valid for bit fields CLS[x]x_CLK_DIV.

Bit 4 **ATOM_OUT_RST**: ATOM_OUT reset level
0 = ATOM_OUT reset level is '0'

1 = ATOM_OUT reset level is '1'

Note: This value represents the ATOM output level after reset. The inverse value of this bit is the reset value of bit SL in all ATOM channels.

Bit 7:5 **ATOM_TRIG_CHAIN**: ATOM trigger chain length without synchronization register

It defines after which ATOM instance count a synchronization register is introduced into trigger chain (after *ATOM_TRIG_<i>* output if instance i and *ATOM_TRIG_<i+1>* input of instance i+1).

Valid values are 1 to 7. 1 means that after each instance a synchronization register is placed.

Bit 8 **TOM_OUT_RST**: TOM_OUT reset level

0 = TOM_OUT reset level is '0'

1 = TOM_OUT reset level is '1'

Note: This value represents the TOM output level after reset. The inverse value of this bit is the reset value of bit SL in all TOM channels.

Bit 11:9 **TOM_TRIG_CHAIN**: TOM trigger chain length without synchronization register

It defines after which TOM instance count a synchronization register is introduced into trigger chain (after *TOM_TRIG_<i>* output if instance i and *TOM_TRIG_<i+1>* input of instance i+1).

Valid values are 1 to 7. 1 means that after each instance a synchronization register is placed.

Bit 12 **RAM_INIT_RST**: RAM initialization from reset

0 = RAM is not initialized after reset

1 = RAM is initialized after reset

Bit 13 **ERM**: enable RAM1 MSB for available MCS modules

0 = MSB of MCS RAM1 address not used

1 = MSB of MCS RAM1 address used

Note: The bit reflects the state of the configuration parameter ERM mentioned in the specification of MCFG.

Bit 14 **ARU_CONNECT_CONFIG**: Defines number of parallel ARU ports

0 = 2 ARU ports available (two independent counter)

1 = 1 ARU port available

Bit 15 **Reserved**

Note: Read as zero, should be written as zero.

Bit 16 **IRQ_MODE_LEVEL**

0 = level mode not available

1 = level mode available

Bit 17 **IRQ_MODE_PULSE**
0 = pulse mode not available
1 = pulse mode available

Bit 18 **IRQ_MODE_PULSE_NOTIFY**
0 = pulse notify mode not available
1 = pulse notify mode available

Bit 19 **IRQ_MODE_SINGLE_PULSE**
0 = single pulse mode not available
1 = single pulse mode available

Bit 23:20 **ATOM_TRIG_INTCHAIN**: ATOM internal trigger chain length without synchronization register
It defines after which ATOM channel count a synchronization register is introduced into trigger chain.
Valid values are 1 to 8. 4 means that in channel 4 of the atom instances a synchronization register is placed.

Bit 28:24 **TOM_TRIG_INTCHAIN**: TOM internal trigger chain length without synchronization register
It defines after which TOM channel count a synchronization register is introduced into trigger chain.
Valid values are 1 to 16. 8 means that in channel 8 of the tom instances a synchronization register is placed.

Bit 29 **INT_CLK_EN_GEN**: Internal clock enable generation
0 = GTM external clock enable signals in use
1 = GTM internal clock enable signals in use

Bit 31:30 **Reserved**
Note: Read as zero, should be written as zero.

Note: Reset value depends on the hardware configuration chosen by silicon vendor.

9.4.9 Register CCM[i]_TIM_AUX_IN_SRC

Address Offset:	see Appendix B																Initial Value:															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								SEL_OUT_N_CH7	SEL_OUT_N_CH6	SEL_OUT_N_CH5	SEL_OUT_N_CH4	SEL_OUT_N_CH3	SEL_OUT_N_CH2	SEL_OUT_N_CH1	SEL_OUT_N_CH0	Reserved								SRC_CH7	SRC_CH6	SRC_CH5	SRC_CH4	SRC_CH3	SRC_CH2	SRC_CH1	SRC_CH0
Mode	R								RW	RW	RW	RW	RW	RW	RW	RW	R								RW	RW	RW	RW	RW	RW	RW	RW
Initial Value	0x00								0	0	0	0	0	0	0	0	0x00								0	0	0	0	0	0	0	0

Bit 0 **SRC_CH0:** Defines AUX_IN source of TIM[i] channel 0
SEL_OUT_N_CH0 = 0 / SEL_OUT_N_CH0 = 1:
0 = CDTM[i].DTM0 Output DTM_OUT0 selected / CDTM[i].DTM0 Output DTM_OUT1_N selected
1 = CDTM[i].DTM4 Output DTM_OUT0 selected / CDTM[i].DTM4 Output DTM_OUT1_N selected

Bit 1 **SRC_CH1:** Defines AUX_IN source of TIM[i] channel 1
SEL_OUT_N_CH1 = 0 / SEL_OUT_N_CH1 = 1:
0 = CDTM[i].DTM0 Output DTM_OUT1 selected / CDTM[i].DTM0 Output DTM_OUT2_N selected
1 = CDTM[i].DTM4 Output DTM_OUT1 selected / CDTM[i].DTM4 Output DTM_OUT2_N selected

Bit 2 **SRC_CH2:** Defines AUX_IN source of TIM[i] channel 2
SEL_OUT_N_CH2 = 0 / SEL_OUT_N_CH2 = 1:
0 = CDTM[i].DTM0 Output DTM_OUT2 selected / CDTM[i].DTM0 Output DTM_OUT3_N selected
1 = CDTM[i].DTM4 Output DTM_OUT2 selected / CDTM[i].DTM4 Output DTM_OUT3_N selected

Bit 3 **SRC_CH3:** Defines AUX_IN source of TIM[i] channel 3
SEL_OUT_N_CH3 = 0 / SEL_OUT_N_CH3 = 1:
0 = CDTM[i].DTM0 Output DTM_OUT3 selected / CDTM[i].DTM1 Output DTM_OUT0_N selected
1 = CDTM[i].DTM4 Output DTM_OUT3 selected / CDTM[i].DTM5 Output DTM_OUT0_N selected

Bit 4 **SRC_CH4:** Defines AUX_IN source of TIM[i] channel 4
SEL_OUT_N_CH4 = 0 / SEL_OUT_N_CH4 = 1:
0 = CDTM[i].DTM1 Output DTM_OUT0 selected / CDTM[i].DTM1 Output DTM_OUT1_N selected
1 = CDTM[i].DTM5 Output DTM_OUT0 selected / CDTM[i].DTM5 Output DTM_OUT1_N selected

Bit 5 **SRC_CH5:** Defines AUX_IN source of TIM[i] channel 5
SEL_OUT_N_CH5 = 0 / SEL_OUT_N_CH5 = 1:
0 = CDTM[i].DTM1 Output DTM_OUT1 selected / CDTM[i].DTM1 Output DTM_OUT2_N selected
1 = CDTM[i].DTM5 Output DTM_OUT1 selected / CDTM[i].DTM5 Output DTM_OUT2_N selected

Bit 6 **SRC_CH6:** Defines AUX_IN source of TIM[i] channel 6
SEL_OUT_N_CH6 = 0 / SEL_OUT_N_CH6 = 1:
0 = CDTM[i].DTM1 Output DTM_OUT2 selected / CDTM[i].DTM1 Output DTM_OUT3_N selected
1 = CDTM[i].DTM5 Output DTM_OUT2 selected / CDTM[i].DTM5 Output DTM_OUT3_N selected

Bit 7 **SRC_CH7:** Defines AUX_IN source of TIM[i] channel 7
SEL_OUT_N_CH7 = 0 / SEL_OUT_N_CH7 = 1:
0 = CDTM[i].DTM1 Output DTM_OUT3 selected / CDTM[i].DTM0 Output DTM_OUT0_N selected
1 = CDTM[i].DTM5 Output DTM_OUT3 selected / CDTM[i].DTM4 Output DTM_OUT0_N selected

Bit 15:8 **Reserved**
Note: Read as zero, should be written as zero.

Bit 16 **SEL_OUT_N_CH0:** Use DTM_OUT or DTM_OUT_N signals as AUX_IN source of TIM[i] channel 0
0 = Use DTM_OUT signal as AUX_IN source of TIM[i]
1 = Use DTM_OUT_N signal as AUX_IN source of TIM[i]

Bit 17 **SEL_OUT_N_CH1:** Use DTM_OUT or DTM_OUT_N signals as AUX_IN source of TIM[i] channel 1

Bit 18 **SEL_OUT_N_CH2:** Use DTM_OUT or DTM_OUT_N signals as AUX_IN source of TIM[i] channel 2

Bit 19 **SEL_OUT_N_CH3:** Use DTM_OUT or DTM_OUT_N signals as AUX_IN source of TIM[i] channel 3

Bit 20 **SEL_OUT_N_CH4:** Use DTM_OUT or DTM_OUT_N signals as AUX_IN source of TIM[i] channel 4

Bit 21 **SEL_OUT_N_CH5:** Use DTM_OUT or DTM_OUT_N signals as AUX_IN source of TIM[i] channel 5

Bit 22 **SEL_OUT_N_CH6:** Use DTM_OUT or DTM_OUT_N signals as AUX_IN source of TIM[i] channel 6

Bit 23 **SEL_OUT_N_CH7:** Use DTM_OUT or DTM_OUT_N signals as AUX_IN source of TIM[i] channel 7

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

9.4.10 Register CCM[i]_EXT_CAP_EN

<table>
<tr><th><b>Address Offset:</b></th><th colspan="16"><b>see Appendix B</b></th><th colspan="16"><b>Initial Value:</b></th></tr>
<tr><td></td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>Bit</td><td colspan="16">Reserved</td><td colspan="8">TIM_IP1_EXT_CAP_EN</td><td colspan="8">TIM_I_EXT_CAP_EN</td></tr>
<tr><td>Mode</td><td colspan="16">R</td><td colspan="8">RW</td><td colspan="8">RW</td></tr>
<tr><td>Initial Value</td><td colspan="16">0x0000</td><td colspan="8">0x00</td><td colspan="8">0x00</td></tr>
</table>

Bit 7:0 **TIM_EXT_CAP_EN:** TIM[i]_EXT_CAPTURE signal forwarding enable
0 = disable forwarding of signal TIM[i]_EXT_CAPTURE to MCS[i]
1 = enable forwarding of signal TIM[i]_EXT_CAPTURE to MCS[i]

Note: The trigger event forwarding is possible from TIM[i] and TIM[i+1] to MCS[i].

Bit 15:8 **TIM_IP1_EXT_CAP_EN:** TIM[i+1]_EXT_CAPTURE signal forwarding enable
0 = disable forwarding of signal TIM[i+1]_EXT_CAPTURE to MCS[i]
1 = enable forwarding of signal TIM[i+1]_EXT_CAPTURE to MCS[i]

Bit 31:16 **Reserved**
Note: Read as zero, should be written as zero.

9.4.11 Register CCM[i]_TOM_OUT

<table>
<tr><th><b>Address Offset:</b></th><th colspan="16"><b>see Appendix B</b></th><th colspan="16"><b>Initial Value:</b></th></tr>
<tr><td></td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>Bit</td><td colspan="16">TOM_OUT_N</td><td colspan="16">TOM_OUT</td></tr>
<tr><td>Mode</td><td colspan="16">R</td><td colspan="16">R</td></tr>
<tr><td>Initial Value</td><td colspan="16">0xXXXX</td><td colspan="16">0xXXXX</td></tr>
</table>

Bit 15:0 **TOM_OUT:** Output level snapshot of TOM[i]_OUT all channels

actual level of primary output ports TOM[i]_OUT of channel 0 to 15 (after DTM)

Note: Reset value depends on the hardware configuration chosen by silicon vendor. See CCM[i]_HW_CONF for chosen value.

Bit 31:16 **TOM_OUT_N:** Output level snapshot of TOM[i]_OUT_N all channels
actual level of primary output ports TOM[i]_OUT_N of channel 0 to 15 (after DTM)

Note: Reset value depends on the hardware configuration chosen by silicon vendor. See CCM[i]_HW_CONF for chosen value.

9.4.12 Register CCM[i]_ATOM_OUT

Address Offset:	see Appendix B																Initial Value:															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	ATOM_IP1_OUT_N								ATOM_IP1_OUT								ATOM_I_OUT_N								ATOM_I_OUT							
Mode	R								R								R								R							
Initial Value	0xXX								0xXX								0xXX								0xXX							

Bit 7:0 **ATOM_I_OUT:** Output level snapshot of ATOM[i]_OUT all channels
actual level of primary output ports ATOM[i]_OUT of channel 0 to 7 (after DTM)

Note: Reset value depends on the hardware configuration chosen by silicon vendor. See CCM[i]_HW_CONF for chosen value.

Bit 15:8 **ATOM_I_OUT_N:** Output level snapshot of ATOM[i]_OUT_N all channels
actual level of primary output ports ATOM[i]_OUT_N of channel 0 to 7 (after DTM)

Note: Reset value depends on the hardware configuration chosen by silicon vendor. See CCM[i]_HW_CONF for chosen value.

Bit 23:16 **ATOM_IP1_OUT:** Output level snapshot of ATOM[i+1]_OUT all channels
actual level of primary output ports ATOM[i+1]_OUT of channel 0 to 7 (after DTM)

Note: Reset value depends on the hardware configuration chosen by silicon vendor. See CCM[i]_HW_CONF for chosen value.

Bit 31:24 **ATOM_IP1_OUT_N:** Output level snapshot of ATOM[i+1]_OUT_N all channels
actual level of primary output ports ATOM[i+1]_OUT_N of channel 0 to 7 (after DTM)

Note: Reset value depends on the hardware configuration chosen by silicon vendor. See CCM[i]_HW_CONF for chosen value.

10 Time Base Unit (TBU)

10.1 Overview

The Time Base Unit TBU provides common time bases for the GTM-IP. The TBU sub-module is organized in channels, where the number of channels is device dependent. There are up to four channels implemented inside the TBU.
The time base register **TBU_CH0_BASE** of TBU channel 0 is 27 bits wide and it is configurable whether the lower 24 bit or the upper 24 bit are provided to the GTM as signal *TBU_TS0*.
The two TBU channels 1 and 2 have a time base register **TBU_CH[y]_BASE** (y: 1, 2) of 24 bit length. The time base register value *TBU_TS[y]* is provided to subsequent sub-modules of the GTM.
The time base register of TBU channel 3 **TBU_CH3_BASE** is 24 bits wide. It is used as a modulo counter by **TBU_CH3_BASE_MARK** to get a relative angle clock to **TBU_CH[y]_BASE**. The absolute angle clock value for the current **TBU_CH3_BASE** is captured in **TBU_CH3_BASE_CAPTURE**.

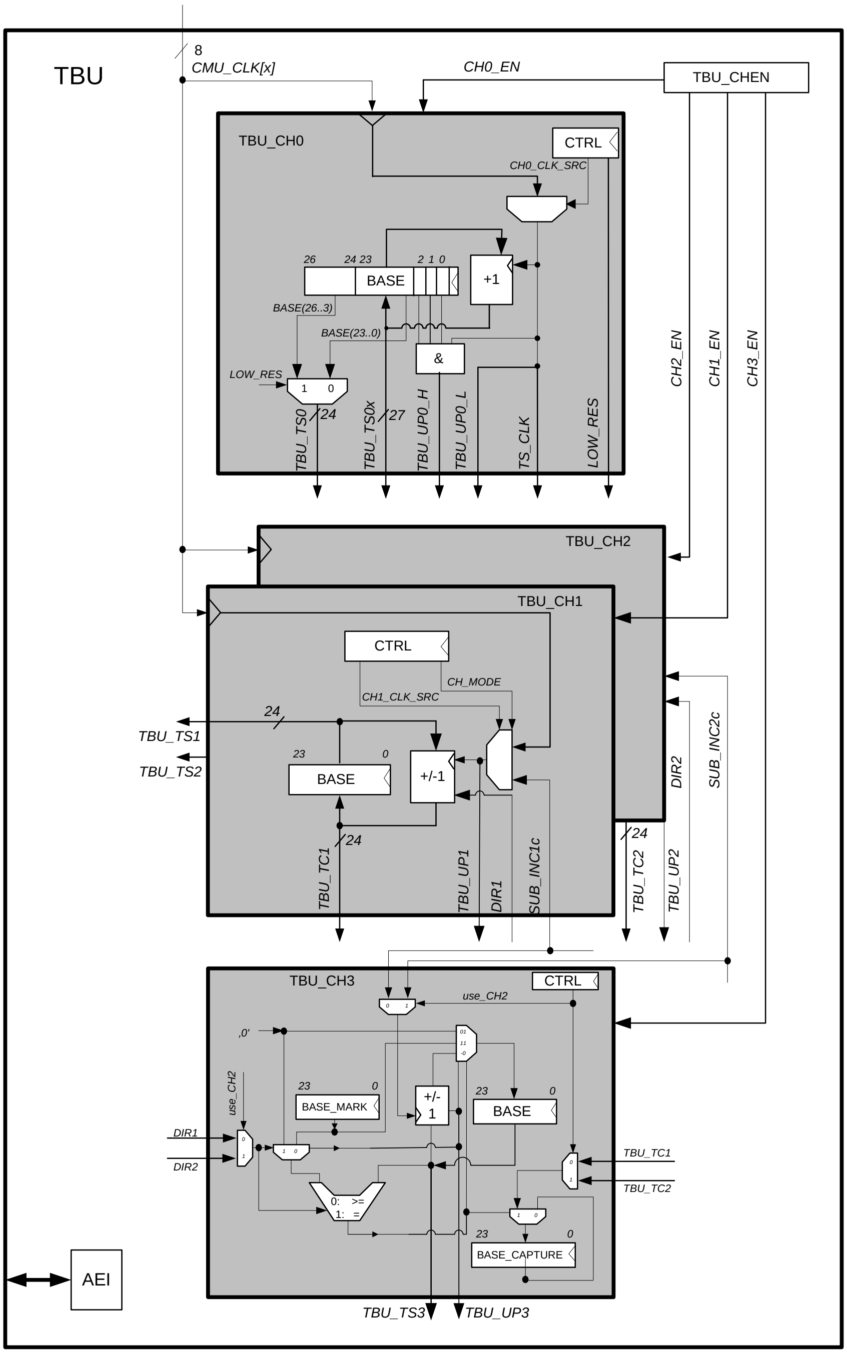
TBU_CH[y]_BASE = TBU_CH3_BASE_CAPTURE + ...
... + TBU_CH3_BASE - DIRy * TBU_CH3_BASE_MARK

DIRy : direction value for time base y (y:1..2)
0 up counter
1 down counter
Note: the right-hand sum is limited to 24 bit.

The *TBU_UP[y]* (y: 1..2) signals are set to high for a single SYS_CLK period, whenever the corresponding signal *TBU_TS[y]* (y: 1..2) is getting updated. The signal *TBU_UP0_L* is set to high for a single SYS_CLK period if the signal TBU_TS0 and TBU_TS0x is getting updated and *TBU_UP0_H* is set to high for a single SYS_CLK period, whenever the upper 24 bit of TBU_TS0 are updated.

The time base channels can run independently of each other and can be enabled and disabled synchronously by control bits in a global TBU channel enable register **TBU_CHEN**. Chapter 10.1.1 shows a block diagram of the Time Base Unit.

10.1.1 TBU Block Diagram



Dependent on the device a third TBU channel exists which offers the same functionality as the time base channel 1.

The configuration of the independent time base channels TBU_CH[z]_BASE is done via the AEI interface. The TBU channel 0 to 2 may select one of the eight *CMU_CLK[x]* (x: 0..7) signals coming from the CMU sub-module.

For TBU channels 1 and 2 an additional clock signal *SUB_INC[y]c* (y: 1, 2) coming from the DPLL can be selected as input clock for the TBU_CH[y]_BASE. This clock in combination with the *DIR[y]* signals determines the counter direction of the TBU_CH[y]_BASE.

The selected time stamp clock signal for the TBU_CH0 sub-unit is served via the *TS_CLK* signal line to the DPLL sub-module. The *TS_CLK* signal equals the signal *TBU_UP0*.

10.2 TBU Channels

The time base values are generated within the TBU time base channels in two independent and one dependent operation modes.

In all modes, the time base register **TBU_CH[z]_BASE** (z: 0..3) can be initialized with a start value just before enabling the corresponding TBU channel.

Moreover, the time base register **TBU_CH[z]_BASE** (z: 0..3) can always be read in order to determine the actual value of the counter.

10.2.1 Independent Modes

10.2.1.1 Free Running Counter Mode

TBU channel 0 provides a 27 bit counter in a free running counter mode. Dependent on the bit field **LOW_RES** of register **TBU_CH0_CTRL,** the lower 24 bits (bit 0 to 23) or the upper 24 bits (bits 3 to 26) are provided to the GTM sub-modules.

TBU channel 1 and 2 provides a 24 bit counter in a free running counter mode enabled by reset CH_MODE of register **TBU_CH[y]_CTRL** (y:1..2).

In TBU Free running counter mode, the time base register **TBU_CH[v]_BASE** (v:0..2) is updated on every specified incoming clock event by the selected signal *CMU_CLK[x]* (x: 0..8) (dependent on **TBU_CH[v]_CTRL** (v:0..2) register). In general the time base register **TBU_CH[v]_BASE** is incremented on every *CMU_CLK[x]* clock tick.

10.2.1.2 Forward/Backward Counter Mode

TBU channel 1 and 2 provides a 24 bit forward/backward counter enabled by set CH_MODE of register **TBU_CH[y]_CTRL** (y:1..2). In this mode the *DIR[y]* signal provided by the DPLL is taken into account.

The value of the time base register **TBU_CH[y]_BASE** is incremented in case when the *DIR[y]* signal equals ‘0’ and decremented in case when the *DIR[y]* signal is ‘1’.

10.2.2 Dependent Mode

10.2.2.1 Modulo Counter Mode

TBU channel 3 provides a 24 bit forward/backward modulo counter. The clock SUB_INC[y]c and counter direction DIR[y] provided by DPLL is selected by use_CH2 of register TBU_CH3_CTRL.

The modulo value is defined in TBU_CH3_BASE_MARK. In forward counter mode if TBU_CH3_BASE value is reaching TBU_CH3_BASE_MARK TBU_CH3_BASE is reset and TBU_TS[y] is captured in TBU_CH3_BASE_CAPTURE. In backward counter mode if TBU_CH3_BASE value is reaching '0' TBU_CH3_BASE is set to TBU_CH3_BASE_MARK and TBU_TS[y] is captured in TBU_CH3_BASE_CAPTURE.

10.3 TBU Configuration Register Overview

10.3.1 TBU Configuration Register Overview Table

Register Name	**Description**	**Details in Section**
TBU_CHEN	TBU global channel enable	10.4.1
TBU_CH0_CTRL	TBU channel 0 control	10.4.2
TBU_CH0_BASE	TBU channel 0 base	10.4.3
TBU_CH1_CTRL	TBU channel 1 control	10.4.4
TBU_CH1_BASE	TBU channel 1 base	10.4.6
TBU_CH2_CTRL	TBU channel 2 control	10.4.5
TBU_CH2_BASE	TBU channel 2 base	10.4.6
TBU_CH3_CTRL	TBU channel 3 control	10.4.7
TBU_CH3_BASE	TBU channel 3 base	10.4.8
TBU_CH3_BASE_MARK	TBU channel 3 modulo value	10.4.9
TBU_CH3_BASE_CAPTURE	TBU channel 3 base captured	10.4.10

Note: In a typical application the Time Base Unit (TBU) considers channels 0, 1 and 3 only. In this case register addresses 0x20...0x2C are reserved and shall be read as zero. Channel 2 can be additionally implemented on special high-end application requirements.

10.4 TBU Register description

10.4.1 Register TBU_CHEN

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																								ENDIS_CH3		ENDIS_CH2		ENDIS_CH1		ENDIS_CH0	
Mode	R																								RW		RW		RW		RW	
Initial Value	0x0000 00																								0b00		0b00		0b00		0b00	

Bit 1:0 **ENDIS_CH0:** TBU channel 0 enable/disable control.
Write / Read :
0b00 = don't care, bits 1:0 will not be changed / channel disabled
0b01 = channel disabled: is read as 00 (see below) / --
0b10 = channel enabled: is read as 11 (see below) / --
0b11 = don't care, bits 1:0 will not be changed / channel enabled

Bit 3:2 **ENDIS_CH1:** TBU channel 1 enable/disable control. See bits 1:0
Bit 5:4 **ENDIS_CH2:** TBU channel 2 enable/disable control. See bits 1:0
Bit 7:6 **ENDIS_CH3:** TBU channel 3 enable/disable control. See bits 1:0
Note: These bits are only applicable if channel is implemented for this device, otherwise read and write as zero

Bit 31:8 **Reserved:** Reserved
Note: Read as zero should be written as zero

10.4.2 Register TBU_CH0_CTRL

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																												CH_CLK_SRC			LOW_RES
Mode	R																												RPw			RPw
Initial Value	0x0000 000																												0b000			0

Bit 0 **LOW_RES:** TBU_CH0_BASE register resolution.

0 = TBU channel uses lower counter bits (bit 0 to 23)

1 = TBU channel uses upper counter bits (bit 3 to 26)

Note: The two resolutions for the TBU channel 0 can be used in the TIM channel 0 and the DPLL sub-modules.

Note: This value can only be modified if channel 0 is disabled.

Bit 3:1 **CH_CLK_SRC:** Clock source for channel x (x:0...2) time base counter

0b000 = *CMU_CLK0* selected

0b001 = *CMU_CLK1* selected

0b010 = *CMU_CLK2* selected

0b011 = *CMU_CLK3* selected

0b100 = *CMU_CLK4* selected

0b101 = *CMU_CLK5* selected

0b110 = *CMU_CLK6* selected

0b111 = *CMU_CLK7* selected

Note: This value can only be modified if channel 0 is disabled.

Bit 31:4 **Reserved:** Reserved

Note: Read as zero should be written as zero

10.4.3 Register TBU_CH0_BASE

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved					BASE																										
Mode	R					RPw																										
Initial Value	0x00					0x0000 00																										

Bit 26:0 **BASE:** Time base value for channel 0.

Note: The value of **BASE** can only be written if the TBU channel 0 is disabled

Note: If channel 0 is enabled, a read access to this register provides the current value of the underlying 27 bit counter.

Bit 31:27 **Reserved:** Reserved

Note: Read as zero should be written as zero

10.4.4 Register TBU_CH1_CTRL

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																												CH_CLK_SRC			CH_MODE
Mode	R																												RPw			RPw
Initial Value	0x0000 000																												0b000			0

Bit 0 **CH_MODE:** Channel mode

0 = Free running counter mode

1 = Forward/backward counter mode

Note: This value can only be modified if channel 1 is disabled. In Free running counter mode the CMU clock source specified by **CH_CLK_SRC** is used for the counter. In Forward/Backward counter mode the *SUB_INC1c* clock signal in combination with the *DIR1* input signal is used to determine the counter direction and clock frequency.

Bit 3:1 **CH_CLK_SRC:** Clock source for channel 1 time base counter
0b000 = *CMU_CLK0* selected
0b001 = *CMU_CLK1* selected
0b010 = *CMU_CLK2* selected
0b011 = *CMU_CLK3* selected
0b100 = *CMU_CLK4* selected
0b101 = *CMU_CLK5* selected
0b110 = *CMU_CLK6* selected
0b111 = *CMU_CLK7* selected

Note: This value can only be modified if channel 1 was disabled

Bit 31:4 **Reserved:** Reserved
Note: Read as zero should be written as zero

10.4.5 Register TBU_CH2_CTRL

Address Offset:	see Appendix B																Initial Value:	0x0000_0000														
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																												CH_CLK_SRC			CH_MODE
Mode	R																												RPw			RPw
Initial Value	0x0000 000																												0b000			0

Bit 0 **CH_MODE:** Channel mode
0 = Free running counter mode
1 = Forward/backward counter mode
Note: This value can only be modified if channel 2 is disabled. In Free running counter mode the CMU clock source specified by **CH_CLK_SRC** is used for the counter. In Forward/Backward counter mode the *SUB_INC2c* clock signal in combination with the *DIR2* input signal is used to determine the counter direction and clock frequency.

Bit 3:1 **CH_CLK_SRC:** Clock source for channel 2 time base counter
0b000 = *CMU_CLK0* selected
0b001 = *CMU_CLK1* selected
0b010 = *CMU_CLK2* selected
0b011 = *CMU_CLK3* selected

0b100 = *CMU_CLK4* selected
0b101 = *CMU_CLK5* selected
0b110 = *CMU_CLK6* selected
0b111 = *CMU_CLK7* selected

Note: This value can only be modified if channel 2 was disabled

Bit 31:4 **Reserved:** Reserved
Note: Read as zero should be written as zero

10.4.6 Register TBU_CH[y]_BASE (y:1,2)

Address Offset:	see Appendix B																Initial Value: 0x0000_0000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								BASE																							
Mode	R								RPw																							
Initial Value	0x00								0x000000																							

Bit 23:0 **BASE:** Time base value for channel y (y: 1, 2)
Note: The value of **BASE** can only be written if the corresponding TBU channel y is disabled
Note: If the corresponding channel y is enabled, a read access to this register provides the current value of the underlying counter.

Bit 31:24 **Reserved:** Reserved
Note: Read as zero should be written as zero

10.4.7 Register TBU_CH3_CTRL

Address Offset:	see Appendix B																Initial Value:								0x0000_0001							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																											USE_CH2	Reserved			CH_MODE
Mode	R																											RPw	R			R
Initial Value	0x0000 000																											0	0b000			1

Bit 0 **CH_MODE:** Channel mode
1 = Forward/backward counter mode

Bit 3:1 **Reserved:** Reserved
Note: Read as zero should be written as zero

Bit 4 **USE_CH2:** Channel selector for modulo counter
0 = TBU_CH1 values used. (*SUB_INC1c* for clock, *DIR1* for counter direction, *TBU_TC1* for capturing)
1 = TBU_CH2 values used. (*SUB_INC2c* for clock, *DIR2* for counter direction, *TBU_TC2* for capturing)

Note: This value can only be modified if channel 3 was disabled

Bit 31:5 **Reserved:** Reserved
Note: Read as zero should be written as zero

10.4.8 Register TBU_CH3_BASE

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								BASE																							
Mode	R								RPw																							
Initial Value	0x00								0x0000 00																							

Bit 23:0 **BASE:** Time base value for channel 3
Note: The value of **BASE** can only be written if the corresponding TBU channel 3 is disabled

Note: If the corresponding channel 3 is enabled, a read access to this register provides the current value of the underlying counter.

Bit 31:24 **Reserved:** Reserved
Note: Read as zero should be written as zero

10.4.9 Register TBU_CH3_BASE_MARK

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_0000**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								BASE_MARK																							
Mode	R								RPw																							
Initial Value	0x00								0x000000																							

Bit 23:0 **BASE_MARK:** Modulo value for channel 3
Note: The value of **BASE_MARK** can only be written if the corresponding TBU channel 3 is disabled

Bit 31:24 **Reserved:** Reserved
Note: Read as zero should be written as zero

10.4.10 Register TBU_CH3_BASE_CAPTURE

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_0000**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								BASE_CAPTURE																							
Mode	R								R																							
Initial Value	0x00								0x000000																							

Bit 23:0 **BASE_CAPTURE:** Captured value of time base channel 1 or 2

Note: When USE_CH2=0 TBU_TC1 is captured and if USE_CH2 is setting TBU_TC2 is captured.

Bit 31:24 **Reserved:** Reserved

Note: Read as zero should be written as zero

11 Timer Input Module (TIM)

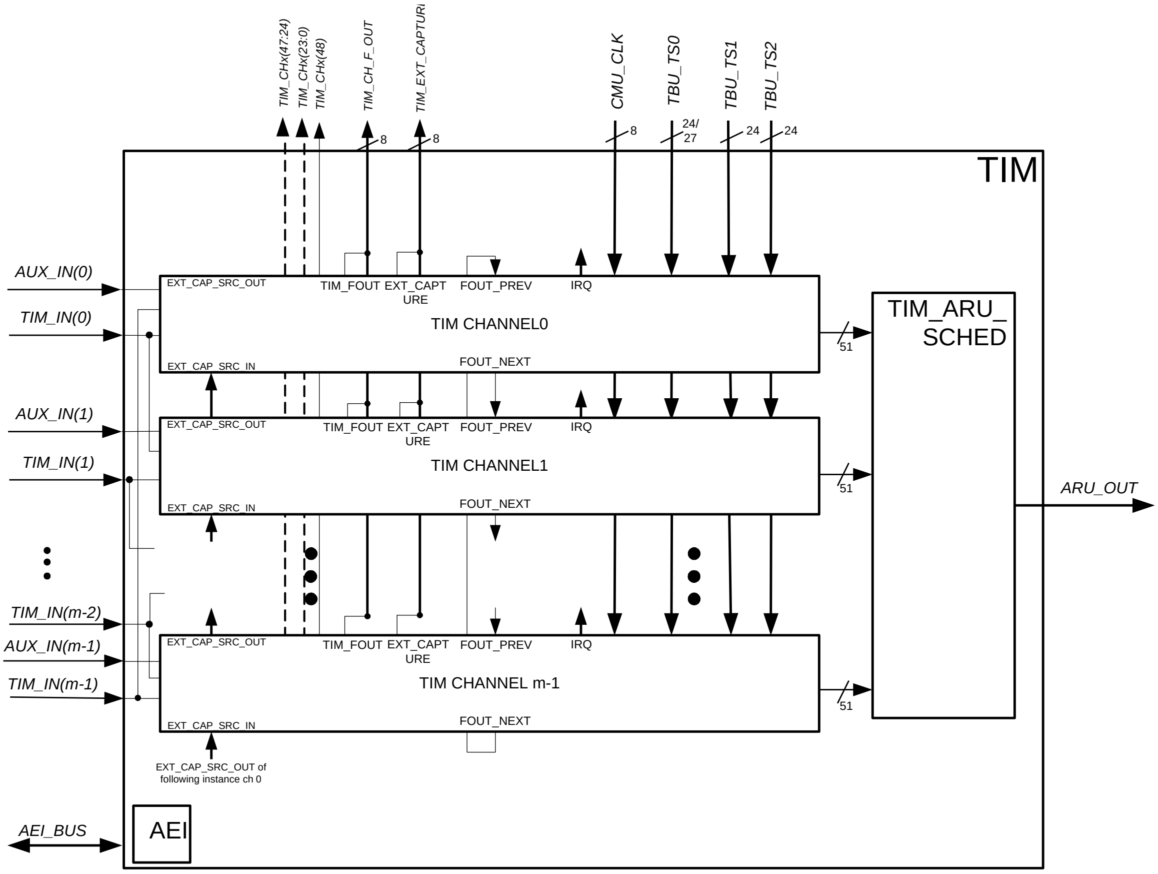
11.1 Overview

The Timer Input Module (TIM) is responsible for filtering and capturing input signals of the GTM. Several characteristics of the input signals can be measured inside the TIM channels. For advanced data processing the detected input characteristics of the TIM module can be routed through the ARU to subsequent processing units of the GTM.

Input characteristics mean either time stamp values of detected input rising or falling edges together with the new signal level or the number of edges received since channel enable together with the actual time stamp or PWM signal duration for a whole PWM period.

The architecture of TIM is shown in Figure 11.1.1.

11.1.1 TIM Block Diagram



The number of channels *m* inside a TIM sub-module depends on the device.

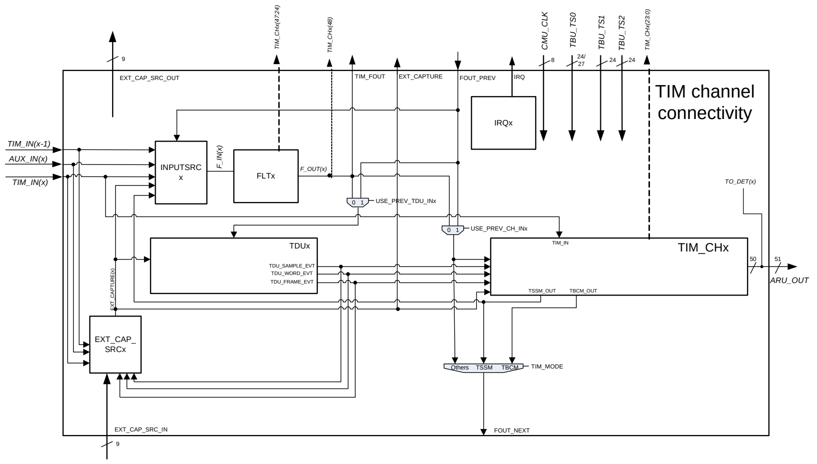
Each of the *m* dedicated input signals are filtered inside the FLTx sub-unit of the TIM Module. It should be noted that the incoming input signals are synchronized to the clock SYS_CLK, resulting in a delay of two SYS_CLK periods for the incoming signals.

The measurement values can be read by the CPU directly via the AEI-Bus or they can be routed through the ARU to other sub-modules of the GTM.

For the GTM-IP TIM0 sub-module only, the dashed signal outputs *TIM[i]_CH[x](23:0)*, *TIM[i]_CH[x](47:24)* and *TIM[i]_CH[x](48)* come from the TIM0 sub-module channels zero (0) to five (5) and are connected to MAP sub-module. There, they are used for further processing and for routing to the DPLL.

The two (three) time bases coming from the TBU are connected to the TIM channels to annotate time stamps to incoming signals. For TIM0 the extended 27 bit width time base TBU_TS0 is connected to the TIM channels, and the user has to select if the lower 24 bits (*TBU_TS0(23..0)*) or the higher 24 bits (*TBU_TS0(26..3)*) are stored inside the **GPR0** and **GPR1** registers.

11.1.2 TIM channel internal connectivity



Above figure gives an overview of the channel internal connectivity of the sub units. The sub units with the major functionality are listed next:

INPUT_SRCx: Select signal for processing by the Filter unit FLTx

FLTx: The filter unit provides different filter mechanisms described in more detail in Chapter 11.2.

TDUx: Timeout detection unit (no subsequent edge detected during a specified duration)

TIM_CHx: Measurement unit; different measurements strategies configurable on the filtered signal

IRQx: Local interrupt controller (enabling, status, ..)

EXT_CAP_SRCx: Selects a local signal ext_capture(x) which is needed by certain functions

Details are given in the next chapters.

Depending on the values of the configuration bit fields **USE_PREV_TDU_INx**, **USE_PREV_CH_INx** it is possible to operate on the signal of the local channel x or the previous channel x-1.

Depending on the value of the configuration bit field **TIM_MODEx** it is possible to provide different signals (via FOUT_NEXT) to the next channel.

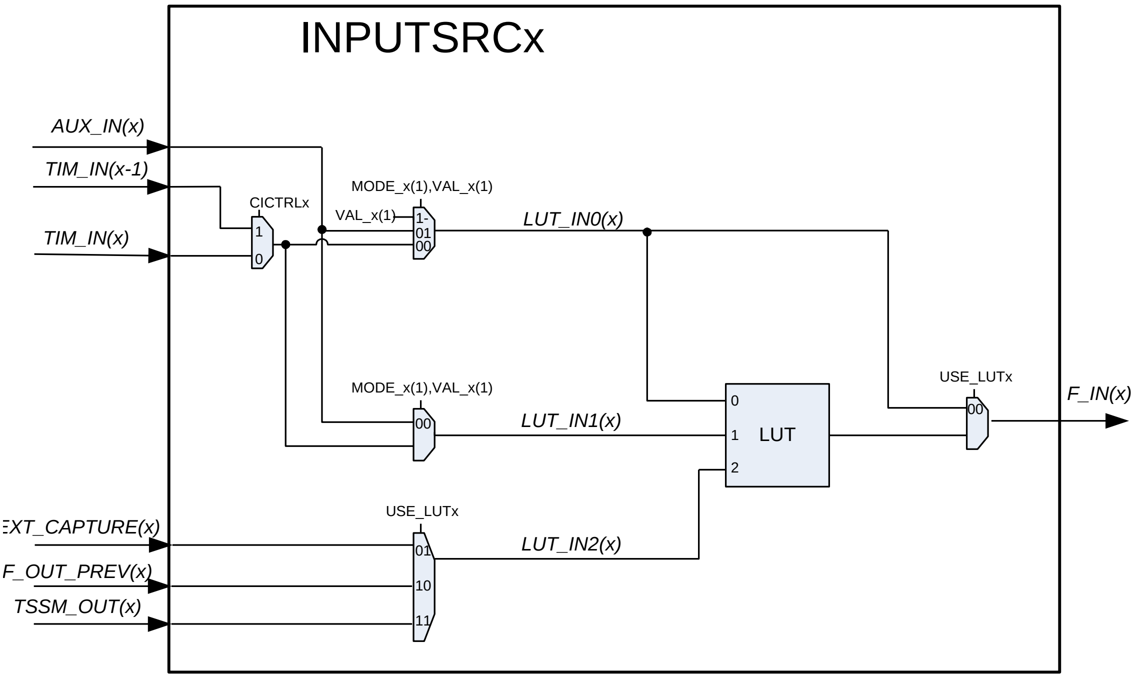
In TBCM mode each capture event selected by the sensitive edges (CNTS) will be forwarded with the value of ECNT[0] to the following channel (via FOUT_NEXT).

11.1.3 Input source selection INPUTSRCx

It can be configured which source shall be used for processing in the FLT,TDU,TIM_CH units. It can be selected by the bit fields **CICTRL** and **MODE_x**, **VAL_x** in the register **TIM[i]_IN_SRC** which source is in use.

Alternatively the signal **F_IN(x)** can be generated by a 8 bit lookup table, which allows to define any function of 3 input sources.

11.1.3.1 INPUTSRC Block Diagram



If **USE_LUT**=b00 is set the lookup table signal generation is bypassed and the signal selection is performed as follows:

In a certain **MODE_x**, **VAL_x** combination the input signal *F_IN(x)* can be driven by **VAL_x(1)** with 0 or 1 directly.

Due to the fact that all 8 channels are bundled in the register **TIM[i]_IN_SRC** a synchronous control of all 8 input channels is possible.

Two adjacent channels can be combined by setting the **CICTRL** bit field in the corresponding **TIM[i]_CH[x]_CTRL** register. This allows for a combination of complex measurements on one input signal with two TIM channels.

The additional signal **AUX_IN[x]** can be selected as an input signal. The source of this signal is defined in the chapter 2.1.4.

If **USE_LUT** !=0b00 is set, the lookup table signal generation with following inputs is in use. See Figure 11.1.3.1:

Input **LUT_INO(x)** selection:
TIM_IN(x) if **CICTRLx**=0 and **MODE_x**(1)=0 and **VAL_x**(1)=0
TIM_IN(x-1) if **CICTRLx**=1 and **MODE_x**(1)=0 and **VAL_x**(1)=0
AUX_IN(x) if **MODE_x**(1)=0 and **VAL_x**(1)=1
VAL_x(1) if **MODE_x**(1)=1

Input **LUT_IN1(x)** selection:
AUX_IN(x) if **MODE_x**(1)=0 and **VAL_x**(1)=0
TIM_IN(x) if **CICTRLx**=0
TIM_IN(x-1) if **CICTRLx**=1

Input **LUT_IN2(x)** selection:
EXT_CAPTURE(x) if **USE_LUT**=0b01
FOUT_PREV(x) if **USE_LUT**=0b10
TSSM_OUT(x) if **USE_LUT**=0b11

The lookup table is defined by the contents of the bit field **TO_CNT2x**. The lookup_table_index is defined by **LUT_IN2(x)** & **LUT_IN1(x)** & **LUT_IN0(x).** The signal **F_IN(x)** is generated by **TO_CNT2x**[lookup_table_index].

If **USE_LUT** !=0b00 is set, only limited functionality is available in the TDU. See bit field **Slicing** in the register TIM[i]_CH[x]_TDUV.

11.1.4 Input observation

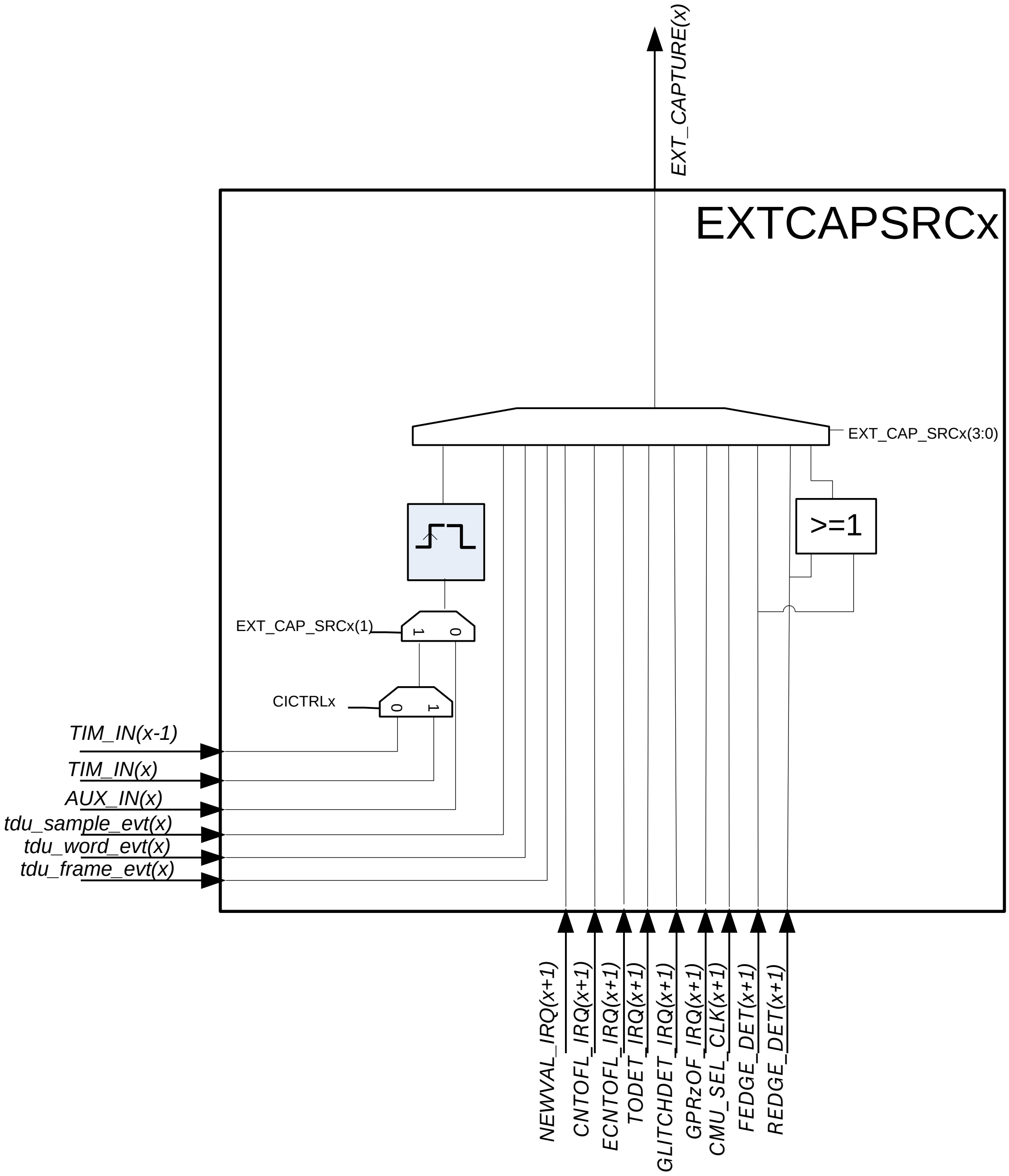
It is possible to observe for all channels of one instance by reading **TIM_INP_VAL** the actual signal values of the following processing stages:

- TIM_IN(7:0) signals after TIM input synchronization
- TIM F_IN(7:0) signals after TIM INPUTSRC selection (input to TIM_FLT)
- TIM F_OUT(7:0) signals after TIM filter functionality (output of TIM_FLT)

11.1.5 External capture source selection EXTCAPSRCx

Each channel can operate on an external capture signal **EXT_CAPTURE**. The source to use for this signal can be configured by the bit field **EXT_CAP_SRCx** in the register TIM[i]_CH[x]_ECTRL .

*11.1.5.1 **EXTCAPSRC Block Diagram***



The external capture functionality can be enabled for the TIM channel **x** with the bit **EXT_CAP_EN** in the register **TIM[i]_CH[x]_CTRL**, it will trigger on each rising edge. A pulse generation for each rising edge of the selected input signal **TIM_IN[x]** and **AUX_IN[x]** is applied.

The six TIM channel interrupt sources can be triggered by the operation in the certain TIM channel modes. Alternatively they can be issued by a soft trigger using the corresponding bits in the register **TIM[i]_CH[x+1]_FORCINT**.

11.2 TIM Filter Functionality (FLT)

11.2.1 Overview

The TIM sub-module provides a configurable filter mechanism for each input signal. These filter mechanism is provided inside the FLT sub-unit.

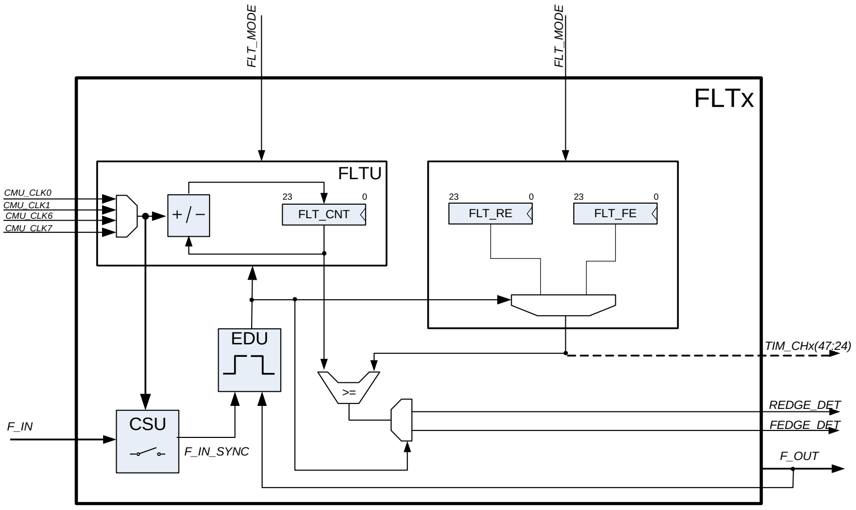
FLT architecture is shown in Figure 11.2.1.1.
The filter includes a clock synchronization unit (CSU), an edge detection unit (EDU), and a filter counter associated to the filter unit (FLTU).

The CSU is synchronizing the incoming signal *F_IN* to the selected filter clock frequency, which is controlled with the bit field **FLT_CNT_FRQ** of register **TIM[i]_CH[x]_CTRL**.

The synchronized input signal *F_IN_SYNC* is used for further processing within the filter.
It should be noted that glitches with a duration go less than the selected CMU clock period is lost.
The filter modes can be applied individually to the falling and rising edges of an input signal. The following filter modes are available:

- immediate edge propagation mode,
- individual de-glitch time mode (up/down counter), and
- individual de-glitch time mode (hold counter).
- individual de-glitch time mode (reset counter).

11.2.1.1 FLT Architecture



The filter parameters (deglitch and acceptance time) for the rising and falling edge can be configured inside the two filter parameter registers **FLT_RE** (rising edge) and **FLT_FE** (falling edge). The exact meaning of the parameter depends on the filter mode.

However the delay time T of both filter parameters **FLT_xE** can always be determined by:

T=(**FLT_xE**+1)*T_{FLT_CLK},

whereas T_{FLT_CLK} is the clock period of the selected CMU clock signal in bit field **FLT_CNT_FRQ** of register **TIM[i]_CH[x]_CTRL**.

When a glitch is detected on an input signal a status flag **GLITCHDET** is set inside the **TIM[i]_CH[x]_IRQ_NOTIFY** register.

Table 11.2.1.2 gives an overview about the meanings for the registers **FLT_RE** and **FLT_FE**. In the individual deglitch time modes, the actual filter threshold for a detected regular edge is provided on the *TIM[i]_CH[x](47:24)* output line. In the case of immediate edge propagation mode, a value of zero is provided on the *TIM[i]_CH[x](47:24)* output line.

The *TIM[i]_CH[x](47:24)* output line is used by the MAP sub-module for further processing (please see chapter 17).

11.2.1.2 Filter Parameter summary for the different Filter Modes

Filter mode	Meaning of FLT_RE	Meaning of FLT_FE
Immediate edge propagation	Acceptance time for rising edge	Acceptance time for falling edge

Individual de-glitch time (up/down counter)	De-glitch time for rising edge	De-glitch time for falling edge
Individual de-glitch time (hold counter)	De-glitch time for rising edge	De-glitch time for falling edge
Individual de-glitch time (reset counter)	De-glitch time for rising edge	De-glitch time for falling edge

A counter **FLT_CNT** is used to measure the glitch and acceptance times.
The frequency of the **FLT_CNT** counter is configurable in bit field **FLT_CNT_FRQ** of register **TIM[i]_CH[x]_CTRL**.
The counter **FLT_CNT** can either be clock with the *CMU_CLK0, CMU_CLK1, CMU_CLK6* or the *CMU_CLK7* signal. These signals are coming from the CMU sub-module.

The **FLT_CNT**, **FLT_FE** and **FLT_RE** registers are 24-bit width. For example, when the resolution of the *CMU_CLK0* signal is 50ns this allows maximal de-glitch and acceptance times of about 838ms for the filter.

11.2.2 TIM Filter Modes

11.2.2.1 Immediate Edge Propagation Mode

In immediate edge propagation mode after detection of an edge the new signal level on *F_IN_SYNC* is propagated to *F_OUT* with a delay of one T_{FLT_CLK} period and the new signal level remains unchanged until the configured acceptance time expires.

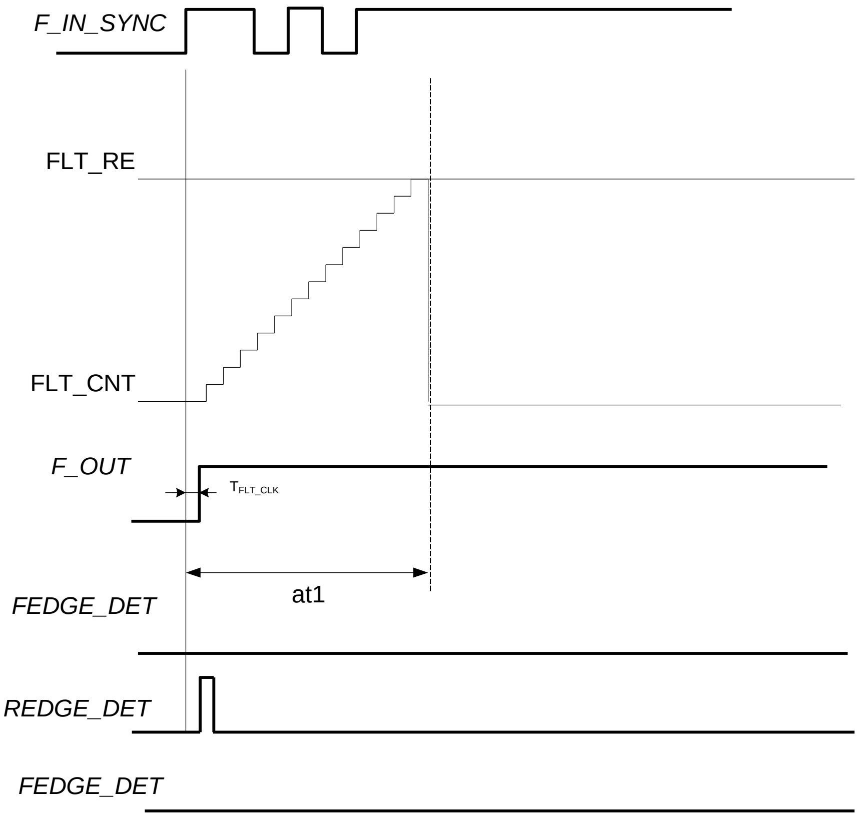
For each edge type the acceptance time can be specified separately in the **FLT_RE** and **FLT_FE** registers.
Each signal change on the input *F_IN_SYNC* during the duration of the acceptance time has no effect on the output signal level *F_OUT* of the filter but it sets the glitch **GLITCHDET** bit in the **TIM[i]_CH[x]_IRQ_NOTIFY** register.

After it expires an acceptance time the input signal *F_IN_SYNC* is observed and on signal level change the filter raises a new detected edge and the new signal level is propagated to *F_OUT*.

Independent of a signal level change the value of *F_OUT* is always set to *F_IN_SYNC,* when the acceptance time expires (see also Figure 11.2.2.1.2).

Figure 11.2.2.1.1 shows an example for the immediate edge propagation mode, in the case of rising edge detection. Both, the signal before filtering (*F_IN*) and after filtering (*F_OUT*) are shown. The acceptance time *at1* is specified in the register **FLT_RE**.

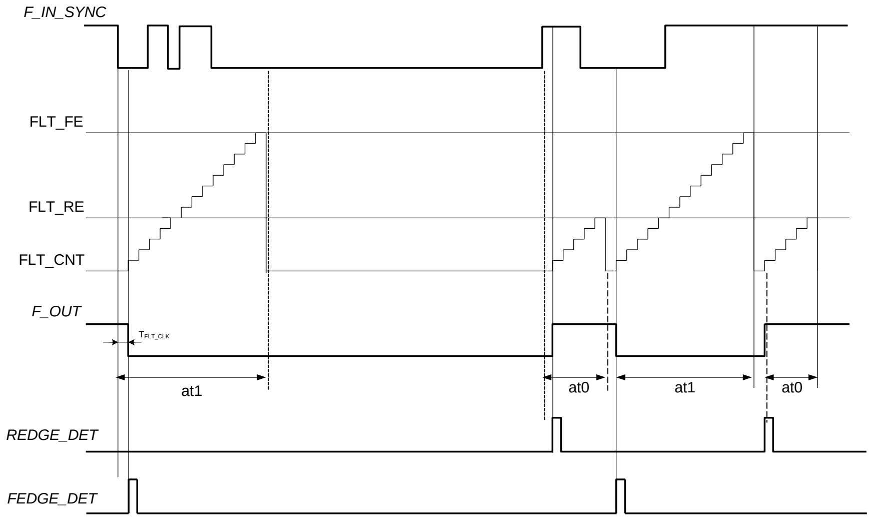
11.2.2.1.1 Immediate Edge Propagation Mode in the case of a rising edge



In immediate edge propagation mode the glitch measurement mechanism is not applied to the edge detection. Detected edges on *F_IN_SYNC* are transferred directly to *F_OUT*.

The counter **FLT_CNT** is incremented until acceptance time threshold is reached. Figure 11.2.2.1.2 shows a more complex example of the TIM filter, in which both, rising and falling edges are configured in immediate edge propagation mode.

11.2.2.1.2 Immediate Edge Propagation Mode in the case of a rising and falling edge



If the **FLT_CNT** has reached the acceptance time for a specific signal edge and the signal *F_IN_SYNC* has already changed to the opposite level of *F_OUT*, the opposite signal level is set to *F_OUT* and the acceptance time measurement is started immediately. Figure 11.2.2.1.2 shows this scenario at the detection of the first rising edge and the second falling edge.

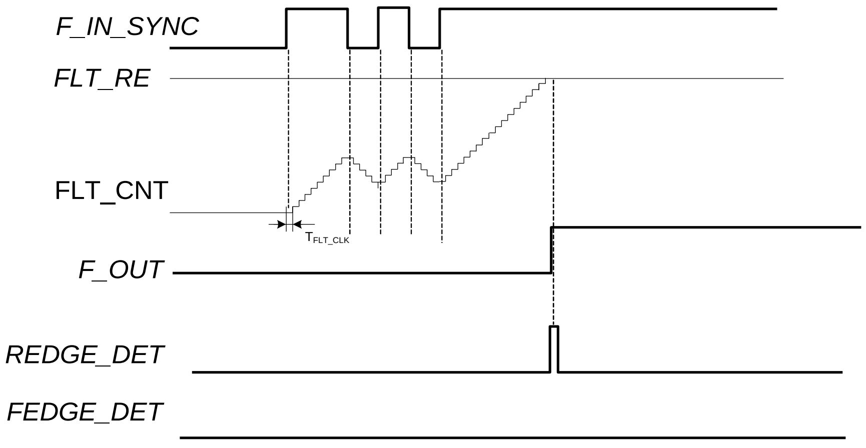
11.2.2.2 Individual De-glitch Time Mode (up/down counter)

In individual de-glitch time mode (up/down counter) each edge of an input signal can be filtered with an individual de-glitch threshold filter value mentioned in the registers **FLT_RE** and **FLT_FE,** respectively.

The filter counter register **FLT_CNT** is incremented when the signal level on *F_IN_SYNC* is unequal to the signal level on *F_OUT* and decremented if *F_IN_SYNC* equals *F_OUT*.

After **FLT_CNT** has reached a value of zero during decrementation the counter is stopped immediately.
If a glitch is detected a glitch detection bit **GLITCHDET** is set in the **TIM[i]_CH[x]_IRQ_NOTIFY** register.
The detected edge signal together with the new signal level is propagated to *F_OUT* after the individual de-glitch threshold is reached. Figure 11.2.2.2.1 shows the behavior of the filter in individual de-glitch time (up/down counter) mode in the case of the rising edge detection.

11.2.2.2.1 Individual De-glitch Time Mode (up/down counter) in the case of a rising edge



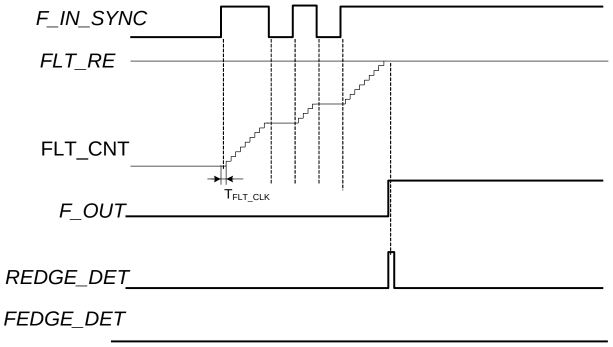
11.2.2.3 Individual De-glitch Time Mode (hold counter)

In individual de-glitch time mode (hold counter) each edge of an input signal can be filtered with an individual de-glitch threshold filter value mentioned in the registers **FLT_RE** and **FLT_FE,** respectively.

The filter counter register **FLT_CNT** is incremented when the signal level on *F_IN_SYNC* is unequal to the signal level on *F_OUT* and the counter value of **FLT_CNT** is hold if *F_IN* equals *F_OUT*.

If a glitch is detected the glitch detection bit **GLITCHDET** is set in the **TIM[i]_CH[x]_IRQ_NOTIFY** register.
The detected edge signal together with the new signal level is propagated to *F_OUT* after the individual de-glitch threshold is reached. Figure 11.2.2.3.1 shows the behavior of the filter in individual de-glitch time (hold counter) mode in the case of the rising edge detection.

11.2.2.3.1 Individual De-glitch Time Mode (hold counter) in the case of a rising edge



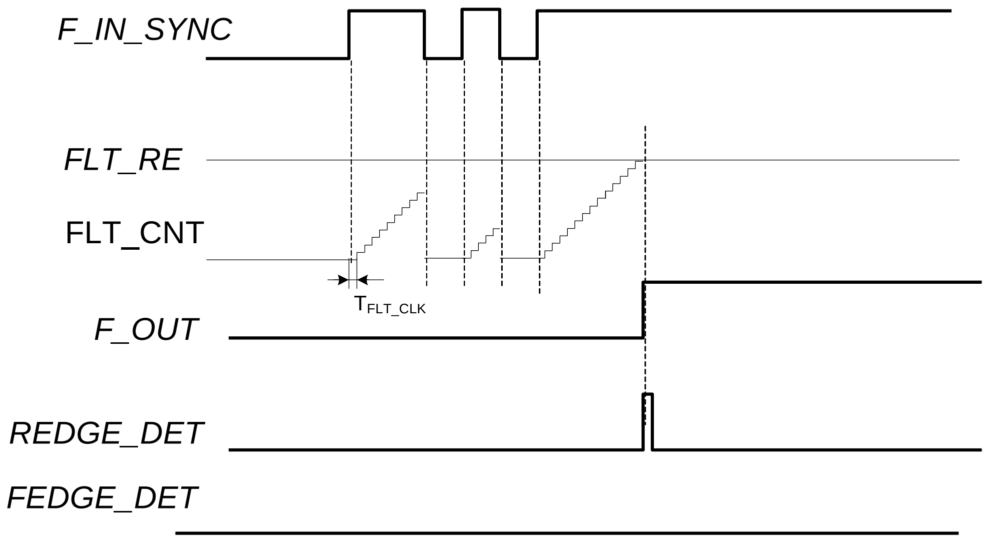
11.2.2.4 Individual De-glitch Time Mode (reset counter)

In individual de-glitch time mode (reset counter) each edge of an input signal can be filtered with an individual de-glitch threshold filter value mentioned in the registers **FLT_RE** and **FLT_FE,** respectively.

The filter counter register **FLT_CNT** is incremented when the signal level on *F_IN_SYNC* is unequal to the signal level on *F_OUT* and the counter value of **FLT_CNT** is reset to 0x000000 if *F_IN* equals *F_OUT*.

If a glitch is detected the glitch detection bit **GLITCHDET** is set in the **TIM[i]_CH[x]_IRQ_NOTIFY** register.
The detected edge signal together with the new signal level is propagated to *F_OUT* after the individual de-glitch threshold is reached. Figure 11.2.2.4.1 shows the behavior of the filter in individual de-glitch time (reset counter) mode in the case of the rising edge detection.

11.2.2.4.1 Individual De-glitch Time Mode (reset counter) in the case of a rising edge



11.2.2.5 Immediate Edge Propagation and Individual De-glitch Mode

As already mentioned, the four different filter modes can be applied individually to each edge of the measured signal.

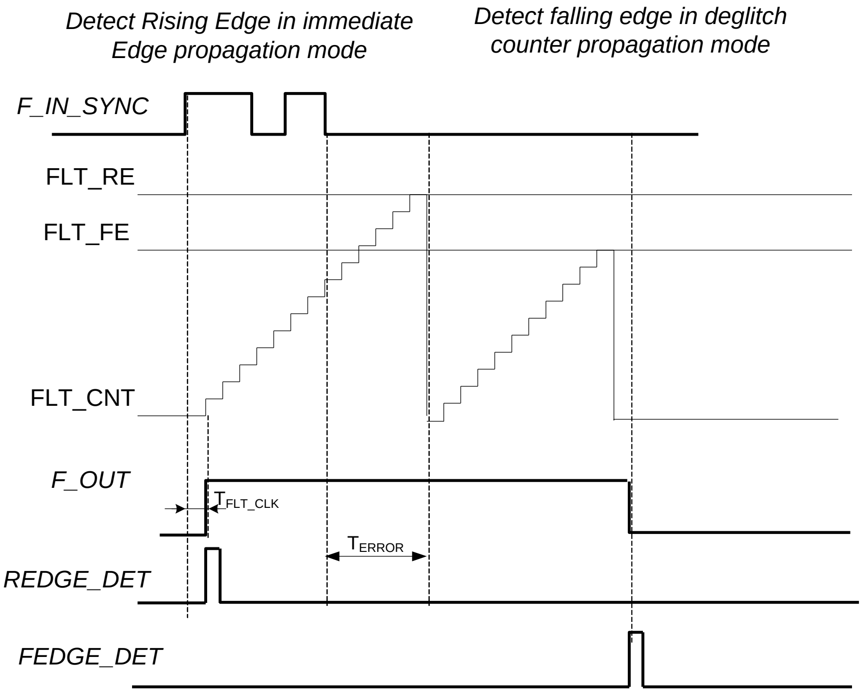
However, if one edge is configured with immediate edge propagation and the other edge with an individual deglitch mode (whether up/down counter, hold counter or reset counter) a special consideration has to be applied.

Assume that the rising edge is configured for immediate edge propagation and the falling edge with individual deglitch mode (up/down counter) as shown in Figure 11.2.2.5.1.

If the falling edge of the incoming signal already occurs during the measuring of the acceptance time of the rising edge, the measurement of the deglitch time on the falling edge is started delayed, but immediately after the acceptance time measurement phase of the rising edge has finished.

Consequently, the deglitch counter cannot measure the time T_{ERROR}, as shown in Figure 11.2.2.5.1.

11.2.2.5.1 Mixed mode measurement


Detect Rising Edge in immediate Edge propagation mode
Detect falling edge in deglitch counter propagation mode
F_IN_SYNC
FLT_RE
FLT_FE
FLT_CNT
F_OUT
TFLT_CLK
TERROR
REDGE_DET
FEDGE_DET

11.2.3 TIM Filter reconfiguration

If **FLT_EN**=1 a change of **FLT_RE** or **FLT_FE** will take place immediately.

If **FLT_EN**=1 a change of **FLT_MODE_RE** or **FLT_MODE_FE** will be used with the next occurring corresponding edge. If the mode is changed while the filter unit is processing a certain mode, it will end this edge filtering in the mode as started.

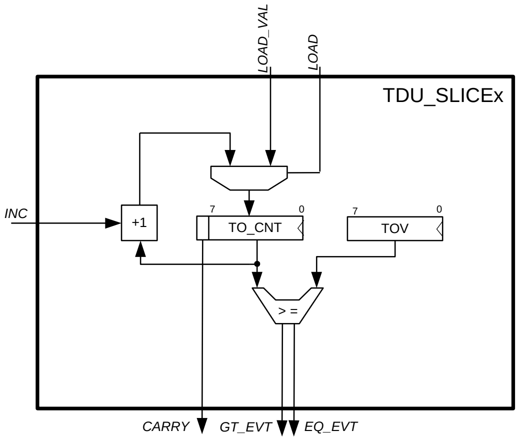
If **FLT_EN**=1 a change of **FLT_CTR_RE**, **FLT_CTR_FE, EFLT_CTR_RE** or **EFLT_CTR_FE** will take place immediately.

11.3 Timeout Detection Unit (TDU)

The Timeout Detection Unit (TDU) is responsible for timeout detection of the TIM input signals.
Each channel of the TIM sub-module has its own Timeout Detection Unit (TDU) where a timeout event can be set up on the filtered input signal of the corresponding channel.

In each timeout unit exist 3 8 bit counter/comparator slices. A counter/comparator slice is shown below. The counter TO_CNT will increment by signal *INC*. The counter can be loaded with the value *LOAD_VAL* if *LOAD* = 1. GT_EVT will be 1 if TO_CNT > TOV is fulfilled. EQ_EVT will be 1 if TO_CNT = TOV is fulfilled.

11.3.1 Counter/comparator slice



The counter/comparator slices can be cascaded depending on the application needs to operate as :

3x 8 bit counter
1x 16 bit counter and 1x 8 bit counter
1x 24 bit counter
2x 8 bit counter.

This allows the user to use the functions

timeout on input signals

local CMU clock prescaler 8 bit
trigger event generation 8 bit (external capture, todet_irq)
in parallel.
With usage of the 3x 8 bit counter it is possible to define different timeout values for the 2 signal levels.

Following table shows which functions can be used in parallel.

11.3.2 Used parallel functions

Counter type	**Timeout functionality**	**Generate local TIM CMU clk**	**Source for external capture to previous channel**	**Source for TODET_IRQ**
24 bit	24 bit	no	tdu_timeout_evt tdu_sample_evt	tdu_timeout_evt tdu_sample_evt
1 x 8 bit 1 x 16 bit	16 bit local clk tdu_sample_evt usable	yes	tdu_timeout_evt, tdu_frame_evt, tdu_sample_evt	tdu_timeout_evt, tdu_frame_evt, tdu_sample_evt
3x 8 bit	8 bit local clk tdu_sample_evt usable	yes	tdu_timeout_evt, tdu_sample_evt, tdu_word_evt, tdu_frame_evt	tdu_timeout_evt, tdu_sample_evt, tdu_word_evt, tdu_frame_evt
3x 8 bit	no	yes	tdu_timeout_evt, tdu_sample_evt, tdu_word_evt, tdu_frame_evt	tdu_timeout_evt, tdu_sample_evt, tdu_word_evt, tdu_frame_evt
2x 8 bit	no	no	tdu_timeout_evt, tdu_word_evt, tdu_frame_evt	tdu_timeout_evt, tdu_word_evt, tdu_frame_evt

Next table shows which of the available 8 bit resources are cascaded with a chosen **SLICING**.

11.3.3 Which of the available 8 bit resources are cascaded with a chosen SLICING

Counter type	**Counters count on**	**Counter resource generates**	**CLK selection**

24 bit	CNT on TCS	CNT= TO_CNT2 & TO_CNT1 & TO_CNT; TCMP = TOV2 & TOV1 & TOV; CNT >= TCMP generates tdu_sample_evt
		tdu_timeout_evt = tdu_sample_evt
		tdu_frame_evt = 0
		tdu_word_evt = 0
			TCS selected

3x 8 bit	TO_CNT2 on TCS TO_CNT on tdu_sample_evt TO_CNT1 on tdu_word_evt	TO_CNT2 >= TOV2 generates tdu_sample_evt TO_CNT >= TOV generates tdu_word_evt TO_CNT1 >= TOV1 generates tdu_frame_evt tdu_timeout_evt = tdu_word_evt	TO_CNT2: TCS selected TO_CNT: tdu_sample_evt selected with TCS_USE_SAMPLE_EVT=1 TO_CNT1: tdu_word_evt selected with TDU_SAME_CNT_CLK=0
3x 8 bit	TO_CNT2 on TCS TO_CNT on tdu_sample_evt TO_CNT1 on tdu_sample_evt	TO_CNT2 >= TOV2 generates tdu_sample_evt TO_CNT >= TOV generates tdu_word_evt TO_CNT1 >= TOV1 generates tdu_frame_evt tdu_timeout_evt = tdu_word_evt or tdu_frame_evt	TO_CNT2: TCS selected TO_CNT: tdu_sample_evt selected with TCS_USE_SAMPLE_EVT=1 TO_CNT1: tdu_sample_evt selected with TDU_SAME_CNT_CLK=1
3x 8 bit	TO_CNT2 on TCS TO_CNT on TCS TO_CNT1 on tdu_word_evt	TO_CNT2 >= TOV2 generates tdu_sample_evt TO_CNT >= TOV generates tdu_word_evt	TO_CNT2: TCS selected TO_CNT: TCS selected with TCS_USE_SAMPLE_EVT=0 TO_CNT1: tdu_word_evt selected with TDU_SAME_CNT_CLK=0

		TO_CNT1 >= TOV1 generates tdu_frame_evt tdu_timeout_evt = tdu_word_evt	
3x 8 bit	TO_CNT2 on TCS TO_CNT on TCS TO_CNT1 on TCS	TO_CNT2 >= TOV2 generates tdu_sample_evt TO_CNT >= TOV generates tdu_word_evt TO_CNT1 >= TOV1 generates tdu_frame_evt tdu_timeout_evt = tdu_word_evt or tdu_frame_evt	TO_CNT2: TCS selected TO_CNT: TCS selected with TCS_USE_SAMPLE_EVT=0 TO_CNT1: TCS selected with TDU_SAME_CNT_CLK=1

2x 8 bit	TO_CNT on TCS
	TO_CNT1 on tdu_word_evt
		TO_CNT >= TOV generates tdu_word_evt
		TO_CNT1 >= TOV1 generates tdu_frame_evt
		tdu_timeout_evt = tdu_word_evt
		tdu_sample_evt = 0
			TO_CNT: TCS selected
			TO_CNT1: tdu_word_evt selected with TDU_SAME_CNT_CLK=0

2x 8 bit	TO_CNT on TCS
	TO_CNT1 on TCS
		TO_CNT >= TOV generates tdu_word_evt
		TO_CNT1 >= TOV1 generates tdu_frame_evt
		tdu_timeout_evt = tdu_word_evt
		tdu_sample_evt = 0
			TO_CNT: TCS selected
			TO_CNT1: TCS selected with TDU_SAME_CNT_CLK=1

1 x 8 bit
1 x 16 bit
	TO_CNT2 on TCS
	CNT on TCS
		TO_CNT2 >= TOV2 generates tdu_sample_evt
		CNT = TO_CNT1 & TO_CNT; TCMP = TOV1 & TOV; CNT >= TCMP generates tdu_frame_evt
		tdu_timeout_evt = tdu_frame_evt
		tdu_word_evt = 0
			TO_CNT2: TCS selected
			CNT: TCS selected with TCS_USE_SAMPLE_EVT=0

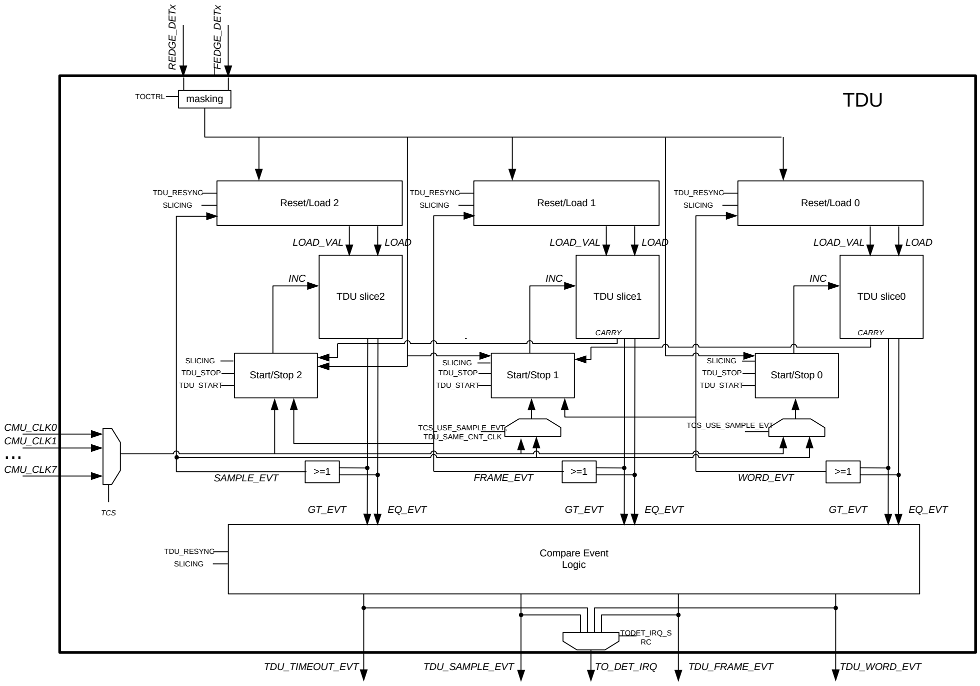
1 x 8 bit
1 x 16 bit
	TO_CNT2 on TCS CNT on tdu_sample_evt	TO_CNT2 >= TOV2 generates tdu_sample_evt
		CNT = TO_CNT1 & TO_CNT; TCMP = TOV1 & TOV; CNT >= TCMP generates tdu_frame_evt
		tdu_timeout_evt = tdu_frame_evt
		tdu_word_evt = 0
			TO_CNT2: TCS selected
			CNT: tdu_sample_evt selected with TCS_USE_SAMPLE_EVT=1

Based on a chosen counter configuration by **SLICING** it is possible to control the start behavior of the counters by **TDU_START** in multiple ways. In addition the stopping of the counters can be controlled by **TDU_STOP**. Depending on the application needs it can be decided how the individual counter slices can be reset/reloaded by the configuration field **TDU_RESYNC**.

Depending on the counter configuration, up to 4 internal compare events tdu_timeout_evt, tdu_sample_evt, tdu_word_evt, tdu_frame_evt out of the 3 comparator slices can be generated. It can be chosen by **TODET_IRQ_SRC** which shall be used as *TIM_TODETx_IRQ* signal which will be accessible by the **TODET** bit inside the **TIM[i]_CH[x]_IRQ_NOTIFY** register

The TDU architecture is shown in 11.3.4.

11.3.4 Architecture of the TDU Sub-unit



Each TDU_slice has its own start/stop control, based on the chosen configuration it will decide if the counter inside the TDU slice will increment on the resolution of the applied clock/event. The reset/load control decides based on the configuration settings and the compare result of the TDU slices those the counters **TO_CNT**,**TO_CNT1**,**TO_CNT2** have to be reloaded. Depending on the chosen counter/compare configuration the compare event logic will generate based on the compare results of the 3 TDU slices and the chosen resolution the events *tdu_sample_evt*, *tdu_word_evt*, *tdu_frame_evt*.

The primary resolution on which the TDU is working can be specified with the bit field **TCS** of the register **TIM[i]_CH[x]_TDUV**. The corresponding input signal *CMU_CLKx* will be used to clock the TDU. The individual timeout/counter values have to be specified in number of ticks of the selected input clock signal in the fields **TOV**, **TOV1**, **TOV2** of the timeout value register **TIM[i]_CH[x]_TDUV** of the TIM channel x.

In case of cascading the bit slices by usage of **SLICING** and **TCS_USE_SAMPLE_EVT** and **TDU_SAME_CLK** the resolution for counting can be switched to the events tdu_sample_evt or tdu_word_evt. More details see table above.

The counter compare units start operation on occurrence of the first "start event" configured by **TDU_START**. They continue their operation until the first "stop event" configured by **TDU_STOP** occurs.

In case of occurrence of a start event and a compare/count resolution event in the same clock cycle, the counters will increment or reload/reset based on **TDU_RESYNC** immediately. No *tdu_sample_evt*, *tdu_word_evt*, *tdu_frame_evt* will be generated.

In case of occurrence of a stop event the counters will not change their values. In case of occurrence of a stop event and a compare/count resolution event in the same clock cycle the corresponding events *tdu_sample_evt*, *tdu_word_evt*, *tdu_frame_evt* will be generated.

In case of occurrence of a start event and a stop event in the same clock cycle the counters will not change their values. No *tdu_sample_evt*, *tdu_word_evt*, *tdu_frame_evt* will be generated.

The function of the timeout unit (configured to **TDU_RESYNC**=0000,**TDU_START**=000) can be started or stopped inside the **TIM[i]_CH[x]_CTRL** register by setting/resetting the **TOCTRL** bit.

Timeout detection can be enabled to be sensitive to falling, rising or both edges of the input signal by writing the corresponding values to the bit field **TOCTRL**.

The TDU generates an interrupt signal *TIM_TODETx_IRQ* whenever a timeout is detected for an individual input signal, and the **TODET** bit is set inside the **TIM[i]_CH[x]_IRQ_NOTIFY** register.

In addition, when the ARU access is enabled with the **ARU_EN** bit inside the **TIM[i]_CH[x]_CTRL** register, the actual values stored inside the registers **TIM[i]_CH[x]_GPR0** and **TIM[i]_CH[x]_GPR1** are sent together with the last stored signal level to the ARU if a timeout event *TDU_TIMEOUT_EVT* occurs.

To signal that a timeout occurred, the ARU_OUT(50) bit (ACB(2)) is set. The bit ACB(0) will be updated with the timeout event to the signal level on which the timeout was detected. Timeout signaling with ACB(2) is only possible with **TODET_IRQ_SRC**= 0000.

Thus, a destination could determine if a timeout occurred at the TIM input by evaluating ACB bit 2.
Since the TIM channel still monitors its input pin although the timeout happened, a valid edge could occur at the input pin while the timeout information is still valid at the ARU. In that case, the new edge associated data is stored inside the registers **TIM[i]_CH[x]_GPR0** and **TIM[i]_CH[x]_GPR1**, the GPR overflow detected bit is set together in the ACB field (ACB(1)) with the timeout bit (ACB(2)) and the values are marked as valid to the ARU.

The ACB bit 2 is cleared, when a successful ARU write access by the TIM channel took place.
The ACB bit 1 is cleared, when a successful ARU write access by the TIM channel took place.
When a valid edge initiates an ARU write access which has not ended while a new timeout occurs the GPR overflow detected bit (ACB(1)) is set. The bit ACB(0) will be updated to the level on which the timeout occurred.

When a timeout occurred and initiates an ARU write access which has not ended while a new timeout occurs the GPR overflow detected bit (ACB(1)) is not set.

The following table clarifies the meaning of the ACB Bits for valid data provided by a TIM channel:

11.3.5 ACB Bits for valid data provided by a TIM channel

ACB4/3	ACB2	ACB1	ACB0	Description
dc	0	0	SL	Valid edge detected
dc	0	1	SL	Input edge overwritten by subsequent edge
dc	1	0	SL	Timeout detected without valid edge
dc	1	1	SL	Timeout detected with subsequent valid edge detected

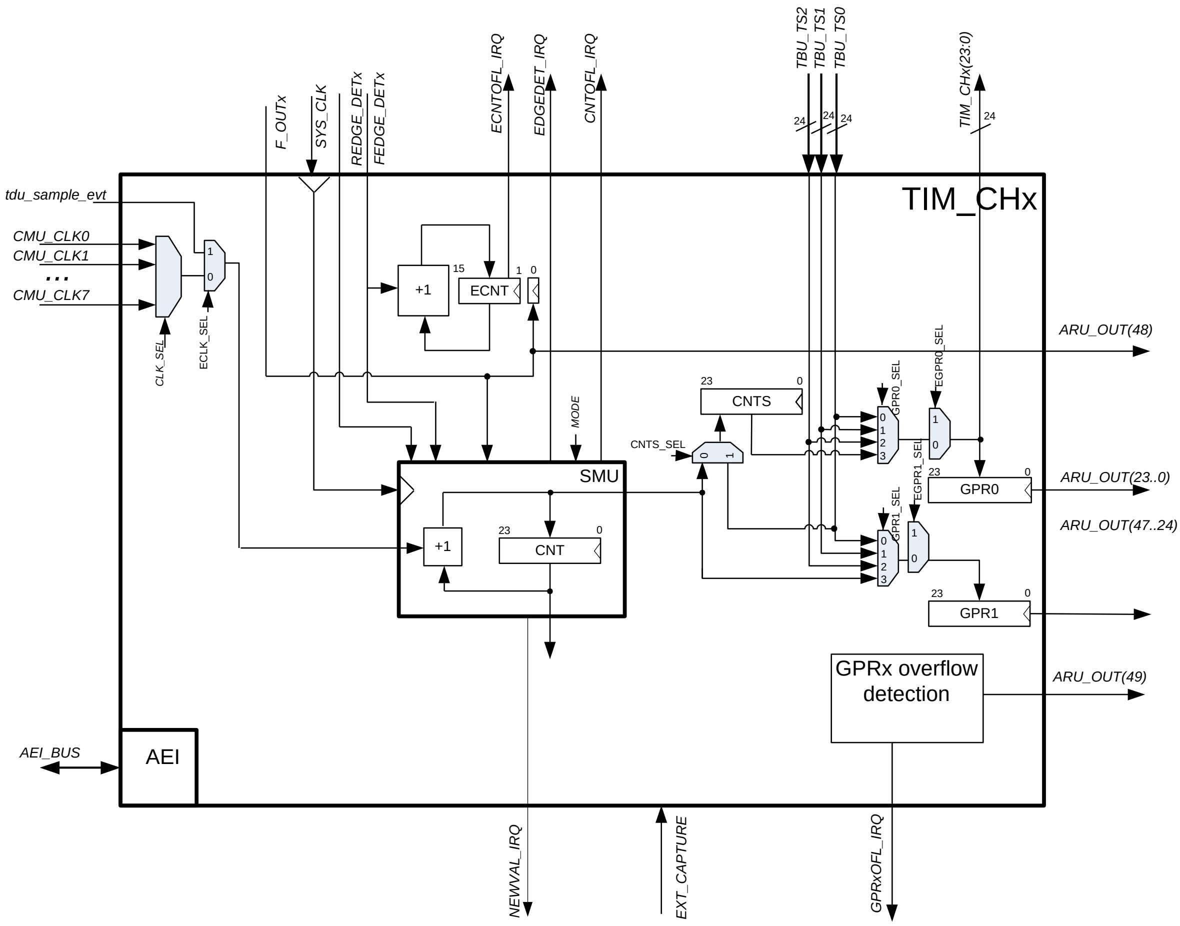
11.4 TIM Channel Architecture

11.4.1 Overview

Each TIM channel consist of an input edge counter **ECNT**, a Signal Measurement Unit (SMU) with a counter **CNT**, a counter shadow register **CNTS** for SMU counter and two general purpose registers **GPR0** and **GPR1** for value storage.

The value **TOV** of the timeout register **TIM[i]_CH[x]_TDU** is provided to TDU sub-unit of each individual channel for timeout measurement. The architecture of the TIM channel is depicted in Figure 11.4.1.1.

11.4.1.1 TIM Channel Architecture



Each TIM channel receives both input trigger signals *REDGE_DETx* and *FEDGE_DETx,* generated by the corresponding filter module in order to signalize a detected echo of the input signal *F_INx*. The signal *F_OUTx* shows the filtered signal of the channel's input signal *F_INx*.

The edge counter **ECNT** counts every incoming filtered edge (rising and falling). The counter value is uneven in case of detected rising, and even in case of detected falling edge. Thus, the input signal level is part of the counter and can be obtained by bit 0 of **ECNT**. (However, the actual counter implementation counts only falling edges on ECNT[n:1] bits. It generates **ECNT** by composing the ECNT[n:1] bits with F_OUTx as bit 0).

Thus, the whole ECNT counter value is always odd, when a positive edge was received and always even, when a negative edge was received.

The current **ECNT[7:0]** register content is made visible on the bits 31 down to 24 of the registers **GPR0**, **GPR1,** and **CNTS**. This allows the software to detect inconsistent read accesses to registers **GPR0**, **GPR1**, and **CNTS**. However, the update strategy of these registers depends on the selected TIM modes, and thus the consistency check has to be adapted carefully.

It can be chosen with the bit field **FR_ECNT_OFL** when an **ECNT** overflow is signaled on **ECNTOFL**. An ECNT overflow can be signaled on 8 bit or full range resolution.

While reading the register **TIM[i]_CH[x]_ECNT** the bit **ECNT[0]** shows the input signal value F_OUTx independent of the state (enabled / disabled) of the channel. If a channel gets disabled (OSM mode or resetting TIM_EN) the content of **TIM[i]_CH[x]_ECNT** will be frozen until a read of the register takes place. This read will reset the **ECNT** counter. Continuing reads will show the input signal value in bit **ECNT[0]** again.

When new data is written into **GPR0** and **GPR1** the **NEWVAL** bit is set in **TIM[i]_CH[x]_IRQ_NOTIFY** register and depending on corresponding enable bit value the *NEWVALx_IRQ* interrupt is raised.

Each TIM input channel has an ARU connection for providing data via the ARU to the other GTM sub-modules. The data provided to the ARU depends on the TIM channel mode and its corresponding adjustments (e.g. multiplexer configuration).

The bit **ARU_EN** of register **TIM[i]_CH[x]_CTRL** decides, whether the measurement results of registers **GPR0** and **GPR1** are consumed by another sub-module via ARU (**ARU_EN** = 1) or the CPU via AEI (**ARU_EN** = 0).

To guarantee a consistent delivery of data from the **GPR0** and **GPR1** registers to the ARU or the CPU each TIM channel has to ensure that the data is consumed before it is overwritten with new values.

If new data was produced by the TIM channel (bit **NEWVAL** is set inside **TIM[i]_CH[x]_IRQ_NOTIFY** register) while the old data is not consumed by the ARU (**ARU_EN** = 1) or CPU (**ARU_EN** = 0), the TIM channel sets the **GPROFL** bit inside the status register **TIM[i]_CH[x]_IRQ_NOTIFY** and it overwrites the data inside the registers **GPR0** and **GPR1**. In addition when **ARU_EN**=1 the bit ACB(1) is set to 1 to indicate the overflow in the ARU data.

If the CPU is selected as consumer for the registers **GPR0** and **GPR1** (**ARU_EN** = 0), the acknowledge for reading out data is performed by a read access to the register **GPR0**. Thus, register **GPR1** should be read always before **GPR0**.

If the ARU is selected as consumer for the registers **GPR0** and **GPR1** (**ARU_EN** = 1), the acknowledge for reading out data is performed by the ARU itself. However, the registers **GPR0** and **GPR1** could be read by CPU without giving an acknowledge.

11.4.2 TIM Channel Modes

The TIM provides seven different measurement modes that can be configured with the bit field **TIM_MODE** of register **TIM[i]_CH[x]_CTRL**. The measurement modes are described in the following subsections. Besides these different basic measurement modes, there exist distinct configuration bits in the register **TIM[i]_CH[x]_CTRL** for a more detailed controlling of each mode. The meanings of these bits are as follows:

- **DSL**: control the signal level for the measurement modes (e.g. if a measurement is started with rising edge or falling edge, or if high level pulses or low level pulses are measured.

- **EGPR0_SEL, GPR0_SEL** and **EGPR1_SEL, GPR1_SEL**: control the actual content of the registers **GPR0** and **GPR1** after a measurement has finished.

- **CNTS_SEL**: control the content of the registers **CNTS.** The actual time for updating the **CNTS** register is mode dependent.

- **OSM**: activate measurement in one-shot mode or continuous mode. In one-shot mode only one measurement cycle is performed and after that the channel is disabled.

- **NEWVAL**: The NEWVAL IRQ interrupt is triggered at the end of a measurement cycle, signaling that the registers GPR0 and GPR1 are updated.

- **ARU_EN**: enables sending of the registers **GPR0** and **GPR1** together with the actual signal level (in bit 48) and the overflow signal GPROFL (in bit 49), and the timeout status information (bit 50) to the ARU.

- **EXT_CAP_EN**: forces an update of the registers **GPR0** and **GPR1** and **CNTS** (TIM channel mode dependent) only on each rising edge of the EXT_CAPTURE signal and triggers a NEWVAL IRQ interrupt. If this mode is disabled the NEWVAL IRQ interrupt is triggered at the end of each measurement cycle.

For each channel the source of the EXT_CAPTURE signal can be configured with the bit fields **EXT_CAP_SRC** in the register **TIM[i]_CH[x]_ECTRL.**

11.4.2.1 TIM PWM Measurement Mode (TPWM)

In TIM PWM Measurement Mode the TIM channel measures duty cycle and period of an incoming PWM signal. The **DSL** bit defines the polarity of the PWM signal to be measured.

When measurement of pulse high time and period is requested (PWM with a high level duty cycle, **DSL**=1) and **IMM_START**=0, the channel starts measuring after the first rising edge is detected by the filter.

If **IMM_START**=1 the measurement starts immediately after activating the channel by **TIM_EN**=1.

Measurement is done with the **CNT** register counting with the configured clock coming from **CMU_CLKx** until a falling edge is detected.

Assume: **SWAP_CAPTURE**=0,**ECNT_RESET**=0
Then the counter value is stored inside the shadow register **CNTS** (if **CNTS_SEL** = 0) and the counter **CNT** counts continuously until the next rising edge is reached.

On this following rising edge the content of the **CNTS** register is transferred to **GPR0** and the content of **CNT** register is transferred to **GPR1**, assuming settings for the selectors **EGPR0_SEL**=0,**GPR0_SEL**=11 and **EGPR1_SEL**=0,**GPR1_SEL**=11. By this, **GPR0** contains the duty cycle length and **GPR1** contains the period. It should be noted, that the bits 1 to 7 of the **ECNT** may be used to check data consistency of the registers **GPR0** and **GPR1**.

In addition the **CNT** register is cleared **NEWVAL** status bit inside of **TIM[i]_CH[x]_IRQ_NOTIFY** status register and depending on corresponding interrupt enable condition *TIM_NEWVALx_IRQ* interrupt is raised.

The CNTS register update is not performed until the measurement is started. Afterwards each edge leaving the level defined by DSL is performing a CNTS register update.

If a PWM with a low level duty cycle should be measured (**DSL** = 0) and **IMM_START**=0, the channel waits for a falling edge until measurement is started. On this edge the low level duty cycle time is stored first in **CNTS** and then finally in **GPR0** and the period is stored in **GPR1**.

When a PWM period was successfully measured, the data in the registers **GPR0** and **GPR1** is marked as valid for reading by the ARU when the **ARU_EN** bit is set inside **TIM[i]_CH[x]_CTRL** register, the **NEWVAL** bit is set inside the **TIM[i]_CH[x]_IRQ_NOTIFY** register, and a new measurement is started.

If the preceding PWM values were not consumed by a reader attached to the ARU (**ARU_EN** bit enabled) or by the CPU the TIM channel set **GPROFL** status bit in **TIM[i]_CH[x]_IRQ_NOTIFY** and depending on corresponding interrupt enable bit value raises a *GPROFL_IRQ* and overwrites the old values in **GPR0** and **GPR1**. A new measurement is started afterwards.

If the register **CNT** produces an overflow during the measurement, the bit **CNTOFL** is set inside the register **TIM[i]_CH[x]_IRQ_NOTIFY** and interrupt *TIM_CNTOFL[x]_IRQ* is raised depending on corresponding interrupt enable condition.

If the register **ECNT** produces an overflow during the measurement, the bit **ECNTOFL** is set inside the register **TIM[i]_CH[x]_IRQ_NOTIFY** and interrupt *TIM_ECNTOFL[x]_IRQ* is raised depending on corresponding interrupt enable condition.

If **ECNT_RESET**=0 the counter **CNT** will be reset to 0 on active edge (defined by **DSL**) of the input signal. If **ECNT_RESET**=1 the counter **CNT** will be reset to 0 on each edge of the input signal.

Assume **EXT_CAP_EN**=0 and **SWAP_CAPTURE**=0:
On every input edge to the active level defined by **DSL** will capture the data selected by **EGPR1_SEL**, **GPR1_SEL** to the registers **GPR1.** Every edge to the inactive level will capture the data selected by **CNTS_SEL** to the registers **CNTS**.

Assume **EXT_CAP_EN**=0 and **SWAP_CAPTURE**=1:
On every input edge to the inactive level defined by **DSL** will capture the data selected by **EGPR1_SEL**, **GPR1_SEL** to the registers **GPR1.** Every edge to the active level will capture the data selected by **CNTS_SEL** to the registers **CNTS**.

11.4.2.1.1 External capture TIM PWM Measurement Mode (TPWM)

If external capture is enabled **EXT_CAP_EN**=1, the pwm measurement is done continuously. The actual measurement values are captured to GPRx if an external capture event occurs.

On every external capture event the data selected by **CNTS_SEL, EGPR0_SEL**, **GPR0_SEL** will be captured to the registers **CNTS, GPR0.**

If **SWAP_CAPTURE**=0 every external capture event will capture the data selected by **EGPR1_SEL**, **GPR1_SEL** to the registers **GPR1.** Every input edge to the level != **DSL** will capture the data selected by **CNTS_SEL** to the registers **CNTS**.

If **SWAP_CAPTURE**=1 every input edge to the inactive level != **DSL** will capture the data selected by **EGPR1_SEL**, **GPR1_SEL** to the registers **GPR1.**

Assume **SWAP_CAPTURE**=0:
Operation is done depending on CMU clock, **ISL, DSL** bit and the input signal value defined in next table (Assume CNTS_SEL= 0):

11.4.2.1.1.1 Operation depending on CMU clock, **ISL**, **DSL** and the input signal value (Assume **CNTS_SEL**= 0)

Input signal F_OUTx	selected CMU Clock	External capture	ISL	DSL	Action description
0	1	0	-	0	CNT++
1	1	0	-	0	no

rising edge	-	0	0	0	capture CNT value in CNTS
falling edge	-	0	0	0	CNT=0
rising edge	-	0	1	0	no
falling edge	-	0	1	0	capture CNT value in CNTS; CNT=0
1	1	0	-	1	CNT++
0	1	0	-	1	no
falling edge	-	0	0	1	capture CNT value in CNTS
rising edge	-	0	0	1	CNT=0
falling edge	-	0	1	1	no
rising edge	-	0	1	1	capture CNT value in CNTS; CNT=0
-	-	rising edge	-	-	do GPRx capture ; issue NEWVAL_IRQ
-	0	0	-	-	no

The CNTS register update is not performed until the measurement is started (first edge defined by DSL is detected). Afterwards the update of the CNTS register is defined by ISL,DSL combinations in the table above.

11.4.2.2 TIM Pulse Integration Mode (TPIM)

In TIM Pulse Integration Mode each TIM channel is able to measure a sum of pulse high or low times on an input signal, depending on the selected signal level bit **DSL** of register **TIM[i]_CH[x]_CTRL** register.

If **IMM_START**=0 the pulse integration measurement is started with occurrence of the first edge defined by DSL on the input signal. If **IMM_START**=1 the measurement starts immediately after activating the channel by **TIM_EN**=1.

The pulse times are measured by incrementing the TIM channel counter **CNT** until the counter is stopped with occurance of a input signal edge to the opposite signal level defined by **DSL**.

The counter **CNT** counts with the *CMU_CLKx* clock specified by the *CLK_SEL* bit field of the **TIM[i]_CH[x]_CTRL** register.

The **CNT** register is reset at the time the channel is activated (enabling via AEI write access) and it accumulates pulses while the channel is staying enabled.

Assume **EXT_CAP_EN**=0 and **SWAP_CAPTURE**=0:

After measurement is started, every falling(DSL=1) or rising(DSL=0) input edge will issue a *TIM_NEWVALx_IRQ* interrupt, and the registers **CNTS**, **GPR0** and **GPR1** are updated according to settings of its corresponding input multiplexers, using the bits **EGPR0_SEL**, **EGPR1_SEL**, **GPR0_SEL**, **GPR1_SEL** and **CNTS_SEL**. It should be noted, that the bits 1 to 7 of the **ECNT** may be used to check data consistency of the registers **GPR0** and **GPR1**.

Assume **EXT_CAP_EN**=0 and **SWAP_CAPTURE**=1:
After measurement is started, every falling(DSL=1) or rising(DSL=0) input edge will issue a *TIM_NEWVALx_IRQ* interrupt, and the registers **CNTS**, **GPR0** are updated according to settings of its corresponding input multiplexers, using the bits **EGPR0_SEL**, **GPR0_SEL** and **CNTS_SEL**.
Every input edge to the active level defined by **DSL** (rising DSL=1;falling DSL=0) will capture the data selected by **EGPR1_SEL**, **GPR1_SEL** to the registers **GPR1.**

When the **ARU_EN** bit is set inside the **TIM[i]_CH[x]_CTRL** register the measurement results of the registers **GPR0** and **GPR1** can be send to subsequent sub-modules attached to the ARU.

11.4.2.2.1 External capture TIM Pulse Integration Mode (TPIM)

If external capture is enabled **EXT_CAP_EN**=1, the pulse integration is done until next external capture event occurs.

On every external capture event the data selected by **CNTS_SEL, EGPR0_SEL, GPR0_SEL** will be captured to the registers **CNTS, GPR0.**

If **SWAP_CAPTURE**=0 every external capture event will capture the data selected by **EGPR1_SEL, GPR1_SEL** to the registers **GPR1.**

If **SWAP_CAPTURE**=1 every input edge to the inactive level != **DSL** will capture the data selected by **EGPR1_SEL**, **GPR1_SEL** to the registers **GPR1.**

Assume **SWAP_CAPTURE**=0; **IMM_START**=0:
Operation is done depending on CMU clock, **DSL** bit and the input signal value defined in next table (inc_cnt = false if TIM channel is enabled) :

11.4.2.2.1.1 Operation depending on CMU clock, **DSL** and the input signal value (inc_cnt = false if TIM channel is enabled)

Input signal F_OUTx	**selected CMU Clock**	**External capture**	**ISL**	**DSL**	**Action description**
falling edge	-	0	-	0	inc_cnt = true

rising edge	-	0	-	0	if inc_cnt == true then { do capture GPRx, CNTS ; issue NEWVAL_IRQ } inc_cnt = false
rising edge	-	0	-	1	inc_cnt = true
falling edge	-	0	-	1	if inc_cnt == true then { do capture GPRx, CNTS ; issue NEWVAL_IRQ } inc_cnt = false
-	1	0	-	-	if inc_cnt == true then CNT++;
-	-	rising edge	-	-	do capture GPRx, CNTS ; issue NEWVAL_IRQ; CNT=0
-	0	0	-	-	no

11.4.2.3 TIM Input Event Mode (TIEM)

In TIM Input Event Mode the TIM channel is able to count edges.
It is configurable if rising, falling or both edges should be counted. This can be done with the bit fields **DSL** and **ISL** in **TIM[i]_CH[x]_CTRL** register.

In addition, a *TIM[i]_NEWVAL[x]_IRQ* interrupt is raised when the configured edge was received and this interrupt was enabled.

The counter register **CNT** is used to count the number of edges, and the bit fields **EGPR0_SE**L, **EGPR1_SE**L,**GPR0_SEL**, **GPR1_SEL**, and **CNTS_SEL** can be used to configure the desired update values for the registers **GPR0**, **GPR1** and **CNTS**. These register are updated whenever the edge counter **CNT** is incremented due to the arrival of a desired edge.

If the preceding data was not consumed by a reader attached to the ARU or by the CPU the TIM channel sets **GPROFL** status bit and raises a *GPROFL[x]_IRQ* if it was enabled in **TIM[i]_CH[x]_IRQ_EN** register and overwrites the old values in **GPR0** and **GPR1** with the new ones.

If the register **CNT** produces an overflow during the measurement, the bit **CNTOFL** is set inside the register **TIM[i]_CH[x]_IRQ_NOTIFY** and interrupt *TIM_CNTOFL[x]_IRQ* is raised depending on corresponding interrupt enable condition.

If the register **ECNT** produces an overflow during the measurement, the bit **ECNTOFL** is set inside the register **TIM[i]_CH[x]_IRQ_NOTIFY** and interrupt *TIM_ECNTOFL[x]_IRQ* is raised depending on corresponding interrupt enable condition.

The TIM Input Event Mode does not depend on the bit field **CLK_SEL** of register **TIM[i]_CH[x]_CTRL**.

11.4.2.3.1 External capture TIM Input Event Mode (TIEM)

If external capture is enabled, capturing is done depending on the **DSL**, **ISL** bit and the input signal value defined in next table:

11.4.2.3.1.1 Capturing depended on the **DSL**, **ISL** and the input signal value, if external capture is enabled

Input signal F_OUTx	**External capture**	**ISL**	**DSL**	**Action description**
-	rising edge	1	-	do capture; issue NEWVAL_IRQ; CNT++
-	0	1	-	no
1	rising edge	0	1	do capture; issue NEWVAL_IRQ; CNT++
0	-	0	1	no
0	rising edge	0	0	do capture; issue NEWVAL_IRQ; CNT++
1	-	0	0	no

11.4.2.4 TIM Input Prescaler Mode (TIPM)

In the TIM Input Prescaler Mode the number of edges which should be detected before a *TIM[i]_NEWVAL[x]_IRQ* is raised is programmable. In this mode it must be specified in the **CNTS** register after how many edges the interrupt has to be raised.

A value of 0 in **CNTS** means that after one edge an interrupt is raised and a value of 1 means that after two edges an interrupt is raised, and so on.

The edges to be counted can be selected by the bit fields **DSL** and **ISL** of register **TIM[i]_CH[x]_CTRL**.
With each triggered interrupt, the registers **GPR0** and **GPR1** are updated according to bits **EGPR0_SE**L, **EGPR1_SE**L,**GPR0_SEL** and **GPR1_SEL**.

If the register **ECNT** produces an overflow during the measurement, the bit **ECNTOFL** is set inside the register **TIM[i]_CH[x]_IRQ_NOTIFY** and interrupt

TIM_ECNTOFL[x]_IRQ is raised depending on corresponding interrupt enable condition.

The TIM Input Prescaler Mode does not depend on the bit field **CLK_SEL** of register **TIM[i]_CH[x]_CTRL**.

11.4.2.4.1 External capture TIM Input Prescaler Mode (TIPM)

If external capture is enabled, the external capture events are counted instead of the input signal edges.
Operation is done depending on the external capture signal, **DSL**, **ISL** bit and the input signal value defined in next table:

11.4.2.4.1.1 Operation depending on the external capture signal, **DSL**, **ISL** and the input signal value

Input signal F_OUTx	**External capture**	**ISL**	**DSL**	**Action description**

-	rising edge	1	-	if CNT >= CNTS then
				do capture ;issue NEWVAL_IRQ; CNT=0
				else
				CNT++
				endif

-	0	1	-	no

1	rising edge	0	1	if CNT >= CNTS then
				do capture ;issue NEWVAL_IRQ; CNT=0
				else
				CNT++
				endif

0	-	0	1	no

0	rising edge	0	0	if CNT >= CNTS then
				do capture ;issue NEWVAL_IRQ; CNT=0
				else
				CNT++
				endif

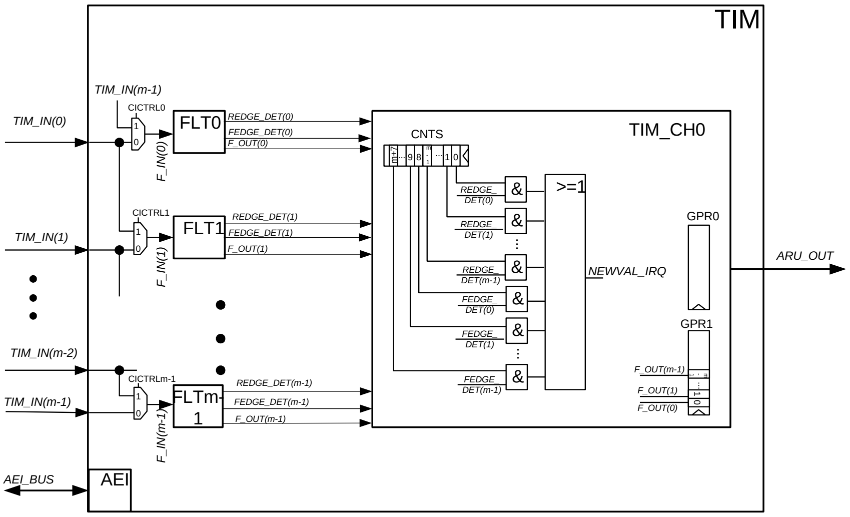
1	-	0	0	no

11.4.2.5 TIM Bit Compression Mode (TBCM)

The TIM Bit Compression Mode can be used to combine all filtered input signals of a TIM sub-module to a parallel *m* bit data word, which can be routed to the ARU, where *m* is the number of channels available in the TIM sub-module.

Figure 11.4.2.5.1 gives an overview of the TIM bit compression mode.

11.4.2.5.1 TIM Bit Compression Mode



The register **CNTS** of a channel is used to configure the event that releases the *NEWVAL_IRQ* and samples the input signals F_IN(0) to F_IN(*m*-1) in ascending order as a parallel data word in **GPR1**.

The bits 0 to *m-1* of the **CNTS** register are used to select the *REDGE_DET* signals of the TIM filters 0 to *m-1* as a sampling event, and the bits 8 to (7+*m*) are used to select the *FEDGE_DET* signals of the TIM filters 0 to *m-1*, respectively. If multiple events are selected, the events are OR-combined (see also Figure 11.4.2.5.1).

EGPR0_SEL,**GPR0_SEL** selects the timestamp value, which is routed through the ARU. **GPR1_SEL** is not applicable in TBCM mode.

If the bit **ARU_EN** of register **TIM[i]_CH[x]_CTRL** is set, the sampled data of register **GPR1** is routed together with a time stamp of register **GPR0** to the ARU, whenever the NEWVAL_IRQ is released.

In TIM Bit compression mode, the register **ECNT** increments with each *NEWVAL_IRQ*, which means that the value of ECNT may depend on all *m* input signals. Consequently, the LSB of **ECNT** does not reflect the actual level of the input signal TIM_IN(x).

If the register **ECNT** produces an overflow during the measurement, the bit **ECNTOFL** is set inside the register **TIM[i]_CH[x]_IRQ_NOTIFY** and interrupt *TIM_ECNTOFL[x]_IRQ* is raised depending on corresponding interrupt enable condition.

The TIM Bit Compression Mode does not depend on the bit field **CLK_SEL** of register **TIM[i]_CH[x]_CTRL**.

11.4.2.5.2 External capture Bit Compression Mode (TBCM)

If external capture is enabled, capturing is done depending on the **DSL**, **ISL** bit and the input signal value defined in next table:

11.4.2.5.2.1 Capturing depended on the **DSL**, **ISL** and the input signal value, if external capture is enabled

Input signal F_OUTx	**External capture**	**ISL**	**DSL**	**Action description**
-	rising edge	1	-	do capture ;issue NEWVAL_IRQ; CNT++
-	0	1	-	no
1	rising edge	0	1	do capture ;issue NEWVAL_IRQ; CNT++
0	-	0	1	no
0	rising edge	0	0	do capture ;issue NEWVAL_IRQ; CNT++
1	-	0	0	no

11.4.2.6 TIM Gated Periodic Sampling Mode (TGPS)

In the TIM Gated Periodic Sampling Mode the number of CMU clock cycles which should elapse before capturing and raising *TIM[i]_NEWVAL[x]_IRQ* is programmable. In this mode it must be specified in the **CNTS** register after how many CMU clock cycles the interrupt has to be raised.

A value of 0 in **TIM[i]_CH[x]_CNTS** means that after one **CLK_SEL** edge a trigger/interrupt is raised, and a value of 1 means that after two edges a trigger/interrupt is raised, and so on.

In the **TIM[i]_CH[x]_CNT** register the elapsed cycles were incremented and compared against **TIM[i]_CH[x]_CNTS**. If **TIM[i]_CH[x]_CNT** is greater or equal to

TIM[i]_CH[x]_CNTS a trigger will be raised. This allows by writing a value to **TIM[i]_CH[x]_CNTS** that the actual period time can be changed on the fly.

Operation is done depending on CMU clock, **DSL**, **ISL** bit and the input signal value defined in next table:

11.4.2.6.1 Operation depending on CMU clock, **DSL**, **ISL** and the input signal value

Input signal F_OUTx	selected CMU Clock	External capture	ISL	DSL	Action description

-	1	0	1	-	if CNT >= CNTS then
					do capture ;issue NEWVAL_IRQ; CNT=0
					else
					CNT++
					endif

0	0	0	0	1	no

1	1	0	0	1	if CNT >= CNTS then
					do capture ;issue NEWVAL_IRQ; CNT=0
					else
					CNT++
					endif

0	0	-	0	1	no

0	1	0	0	0	if CNT >= CNTS then
					do capture ;issue NEWVAL_IRQ; CNT=0
					else
					CNT++
					endif

1	0	0	0	0	no
-	0	0	-	-	no

In this mode the **TIM[i]_CH[x]_GPR1** operates as a shadow register for **TIM[i]_CH[x]_CNTS**. This would allow that the period for the next sampling period could be specified. The update of **TIM[i]_CH[x]_CNTS** will only take place once on a trigger if the **TIM[i]_CH[x]_GPR1** was written by the CPU. This means that the captured value from the previous trigger can be read by the CPU from **TIM[i]_CH[x]_GPR1** and afterwards the new sampling period for the next sampling period (the one after the actual sampling period) could be written.

With each triggered interrupt, the registers **GPR0** and **GPR1** are updated according to bits **GPR0_SEL**, **GPR1_SEL, EGPR0_SEL** and **EGPR1_SEL**.

When selecting **ECNT** as a source for the capture registers, GPRx will show the edge count and the input signal value at point of capture. Selecting **GPR0_SEL** = '11' and **EGPR0_SEL** = '0' for TIM channel 0 all 8 TIM input signals will be captured to **GPR0[7:0]**.

In the TGPS Mode the bit field **CLK_SEL** of register **TIM[i]_CH[x]_CTRL** will define the selected CMU clock which will be used.

The behavior of the **ECNT** counter is configurable by **ECNT_RESET**. If set to 1 on each interrupt (period expired) the **ECNT** will be reset. Otherwise it operates in wrap around mode.

If the register **ECNT** produces an overflow during the measurement, the bit **ECNTOFL** is set inside the register **TIM[i]_CH[x]_IRQ_NOTIFY** and interrupt *TIM_ECNTOFL[x]_IRQ* is raised depending on corresponding interrupt enable condition.

11.4.2.6.2 External capture TIM Gated Periodic Sampling Mode (TGPS)

If external capture is enabled, the external capture events will capture the GPRx, reset the counter **CNT** and issue a *NEWVAL_IRQ*.

Operation is done depending on the CMU clock, external capture signal, **DSL**, **ISL** bit and the input signal value defined in next table:

11.4.2.6.2.1 Operation depending on the CMU clock, external capture signal, **DSL**, **ISL** and the input signal value

Input signal F_OUTx	**selected CMU Clock**	**External capture**	**ISL**	**DSL**	**Action description**

-	1	0	1	-	if CNT >= CNTS then
					do capture; issue NEWVAL_IRQ; CNT=0
					else
					CNT++
					endif

0	0	0	0	1	no

1	1	0	0	1	if CNT >= CNTS then
					do capture; issue NEWVAL_IRQ; CNT=0
					else
					CNT++

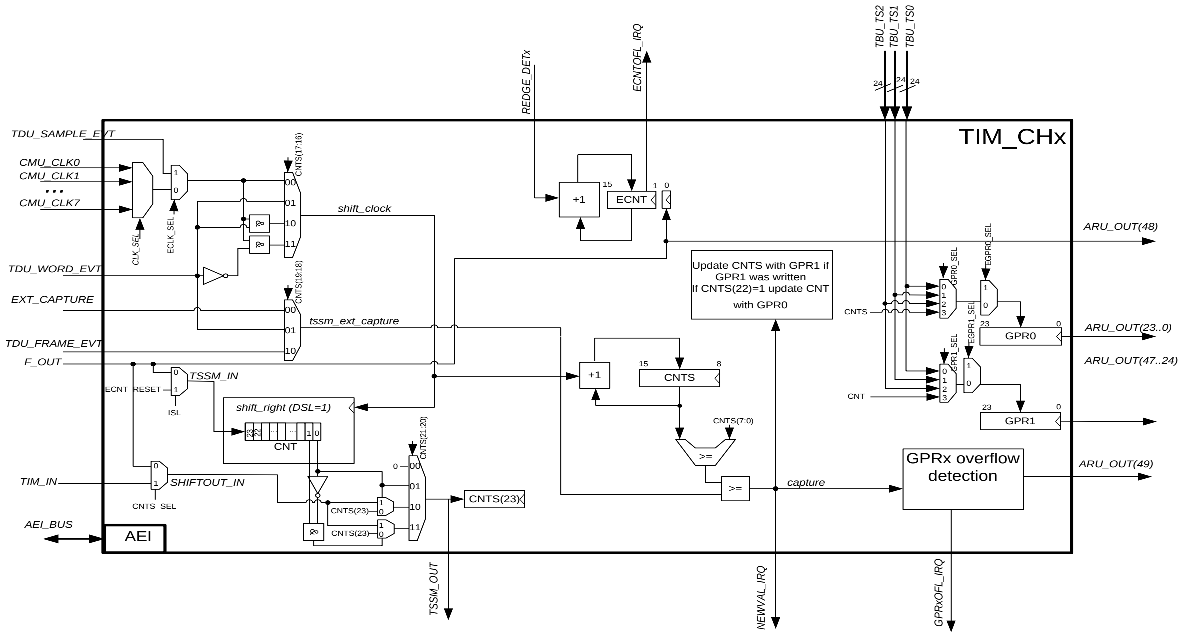
					endif
0	0	-	0	1	no

0	1	0	0	0	if CNT >= CNTS then
					do capture; issue NEWVAL_IRQ; CNT=0
					else
					CNT++
					endif

1	0	0	0	0	no
-	0	0	-	-	no
-	-	rising edge	-	-	do capture; issue NEWVAL_IRQ; CNT =0

11.4.2.7 TIM Serial Shift Mode (TSSM)

11.4.2.7.1 TIM Serial Shift Mode



In the TIM Serial Shift Mode on each shift clock event the actual value of the input signal **TSSM_INx** will be registered in dependence of **DSL** in the register **TIM[i]_CH[x]_CNT**.

If **ISL**=0 is set **FOUTx** will be used as shift in value **TSSM_INx**, with **ISL**=1 the bit field **ECNT_RESET** defines the value for **TSSM_INx**.

With **DSL**=0 **TSSM_INx** will be stored in **TIM[i]_CH[x]_CNT[0]** and **TIM[i]_CH[x]_CNT[22:0]** will be shifted left. With **DSL**=1 **TSSM_OUTx** will be stored in **TIM[i]_CH[x]_CNT[23]** and **TIM[i]_CH[x]_CNT[23:1]** will be shifted right.

Operation is done depending on the shift clock, external capture signal, **DSL**, **ISL** bit and the input signal value defined in next table:

11.4.2.7.2 Operation depending on the shift clock, external capture signal, **DSL**, **ISL** and the input signal value

Input signal TSSM_INx	**shift clock**	**tssm_ext_capture**	**ISL**	**DSL**	**Action description**
-	0	0	-	-	no

-	-	1	-	-	if EXT_CAP_EN=1 then
					see function table in next chapter
					else
					no
					endif

value	1	0	0	0	CNT[23:1]= CNT[22:0];
					CNT[0]= value
					if CNTS[15:8] >= CNTS[7:0] then
					do capture; issue NEWVAL_IRQ;
					CNTS[15:8]=0
					else
					CNTS[15:8]++
					endif

value	1	0	0	1	CNT[22:0]= CNT[23:1];
					CNT[23]= value
					if CNTS[15:8] >= CNTS[7:0] then
					do capture; issue NEWVAL_IRQ;
					CNTS[15:8]=0
					else
					CNTS[15:8]++
					endif

value	1	0	1	0	CNT[23:1]= CNT[22:0];
					CNT[0]= value
					if CNTS[15:8] >= CNTS[7:0] then
					do capture; issue NEWVAL_IRQ;
					CNTS[15:8]=0

					CNT[23:0]=ECNT_RESET
					else
					CNTS[15:8]++
					endif

value	1	0	1	1	CNT[22:0]= CNT[23:1];
					CNT[23]= value
					if CNTS[15:8] >= CNTS[7:0] then
					do capture; issue NEWVAL_IRQ;
					CNTS[15:8]=0
					CNT[23:0]=ECNT_RESET
					else
					CNTS[15:8]++
					endif

The register **TIM[i]_CH[x]_CNTS[7:0]** define the amount of bits which will be stored inside **TIM[i]_CH[x]_CNT.**
Each shift clock will increment the register **TIM[i]_CH[x]_CNTS[15:8].** If the condition **TIM[i]_CH[x]_CNTS[15:8] >= TIM[i]_CH[x]_CNTS[7:0]** is met a capture event is raised and *TIM[i]_NEWVAL[x]_IRQ* is asserted.

With each capture event the registers **GPR0** and **GPR1** are updated according to bits **GPR0_SEL**, **GPR1_SEL, EGPR0_SEL** and **EGPR1_SEL**.

If the bit field **ISL** is set to 1 the register bits **TIM[i]_CH[x]_CNT** are set to the value defined by **ECNT_RESET** in case of a capture event.

In this mode the **TIM[i]_CH[x]_GPR1** operates as a shadow register for **TIM[i]_CH[x]_CNTS**. This allows that the amount of bits to sample can be specified. The update of **TIM[i]_CH[x]_CNTS** will only take place once on a trigger if the **TIM[i]_CH[x]_GPR1** was written by the CPU. This means that the captured value from the previous trigger can be read by the CPU from **TIM[i]_CH[x]_GPR1** and afterwards the new amount of bits to sample for the next sampling period (the one after the actual sampling period) could be written.

The shift clock which will be in use is selectable by **TIM[i]_CH[x]_CNTS[17:16]**:
b00: source selection by **USE_TDU_CLK_SRC** is in use. It can be set to any CMU_CLK source or to the local TDU sample clock tdu_sample_evt.
b01: the tdu_word_evt signal will be used as shift clock source.
b10: the clk source selected by **USE_TDU_CLK_SRC** is used and gated with tdu_word_evt. If tdu_word_evt=0 then shift clock will be 0.
b11: the clk source selected by **USE_TDU_CLK_SRC** is used and gated with tdu_word_evt. If tdu_word_evt=1 then shift clock will be 0.

11.4.2.7.3 Signal Generation with TIM Serial Shift Mode

If **TIM[i]_CH[x]_CNTS[22]** is 1 the **TIM[i]_CH[x]_GPR0** operates as a shadow register for **TIM[i]_CH[x]_CNT**. This allows that the bits for shifting out can be specified. The update of **TIM[i]_CH[x]_CNT** will only take place once on a trigger if the **TIM[i]_CH[x]_GPR0** was written by the CPU. This means that the captured value from the previous trigger can be read by the CPU from **TIM[i]_CH[x]_GPR0** and afterwards the new bits to shift out could be written.

In addition the TIM Serial Shift Mode is able to generate a signal **TSSM_OUT** which can be used internally to the TIM channel.
On each system clock the value for **TSSM_OUT** is generated as defined next. The actual value can be read by the register bit **TIM[i]_CH[x]_CNTS[23]**.
Following functionality for **TSSM_OUTx** is selectable by **TIM[i]_CH[x]_CNTS[21:20]:**
b00: Constant output; **TSSM_OUTx** = 0.
b01: Shift output; If **DSL**=0 (shift left) then **TSSM_OUTx** = **TIM[i]_CH[x]_CNT[23]** else (shift right) **TSSM_OUTx** = **TIM[i]_CH[x]_CNT[0].**
b10: Latched output;If **DSL**=0 and **TIM[i]_CH[x]_CNT[23]**=1 then **TSSM_OUTx** = **SHIFTOUT_INx** elsif **DSL**=1 and **TIM[i]_CH[x]_CNT[0]**=1 then **TSSM_OUTx** = **SHIFTOUT_INx.**
b10: Registered output;If **DSL**=0 and **TIM[i]_CH[x]_CNT[23:22]**=b01 then **TSSM_OUTx** = **SHIFTOUT_INx** elsif **DSL**=1 and **TIM[i]_CH[x]_CNT[1:0]**=b10 then **TSSM_OUTx** = **SHIFTOUT_INx.**
In case of registered or latched output mode the signal **SHIFTOUT_INx** is selectable by **CNTS_SEL**.
If **CNTS_SEL**=0 is set **FOUTx** will be used for **SHIFTOUT_INx**, with **CNTS_SEL**=1 the signal **TIM_INx** is in us for **SHIFTOUT_INx**.

11.4.2.7.4 External capture TIM Serial Shift Mode (TSSM)

If external capture is enabled (**EXT_CAP_EN**=1), the external capture events will capture the GPRx, reset the counter **CNT** depending on **ISL** and issue a *NEWVAL_IRQ*. Functionality from previous table will be applied.

The source which will be used as external capture event for TSSM mode is selectable by **TIM[i]_CH[x]_CNTS[19:18]**:
b00: source selection by **EXT_CAP_SRC** is in use.
b01: tdu_word_evt signal will be used as source.
b10: tdu_frame_evt signal will be used as source.
b11: reserved

Operation is done depending on the shift clock, external capture signal, **DSL**, **ISL** bit and the input signal value defined in next table:

11.4.2.7.4.1 Operation depending on the shift clock, external capture signal, **DSL**, **ISL** and the input signal value

Input signal F_OUTx	**shift clock**	**tssm_ext_capture**	**ISL**	**DSL**	**Action description**
-	0	1	1	-	do capture; issue NEWVAL_IRQ; CNTS[15:8]=0 CNT[23:0]=ECNT_RESET
-	0	1	0	-	do capture; issue NEWVAL_IRQ; CNTS[15:8]=0
value	1	1	1	0	CNT[23:1]= CNT[22:0]; CNT[0]= value do capture; issue NEWVAL_IRQ; CNTS[15:8]=0 CNT[23:0]=ECNT_RESET
value	1	1	1	1	CNT[22:0]= CNT[23:1]; CNT[23]= value do capture; issue NEWVAL_IRQ; CNTS[15:8]=0 CNT[23:0]=ECNT_RESET
value	1	1	0	0	CNT[23:1]= CNT[22:0]; CNT[0]= value do capture; issue NEWVAL_IRQ; CNTS[15:8]=0
value	1	1	0	1	CNT[22:0]= CNT[23:1]; CNT[23]= value do capture; issue NEWVAL_IRQ; CNTS[15:8]=0

11.5 MAP Submodule Interface

The GTM-IP provides one dedicated TIM sub-module TIM0 where channels zero (0) to five (5) are connected to the MAP sub-module described in chapter 17. There, the TIM0 sub-module channels provide the input signal level together with the actual filter value and the annotated time stamp for the edge together in a 49 bit wide signal to the MAP sub-module. This 49 bit wide data signal is marked as valid with a separate valid signal tim0_map_dval[x] (x: 0..5).

11.5.1 Structure of map data

tim0_map_data[x](48)	signal level bit from tim0_ch[x]
tim0_map_data[x](47:24)	actual filter value **TIM0_CH[x]_FLT_RE/ TIM0_CH[x]_FLT_FE** if corresponding channel x bit field **FLT_MODE_RE/ FLT_MODE_FE** is 1 else 0 is assigned.
tim0_map_data[x](23:0)	time stamp value selected by **TBU0_SEL, GRP0_SEL, EGPR0_SEL, CNTS_SEL** of channel x if bit field **TIM_EN**= 1
tim0_map_dval[x]	mark *tim0_map_data[x]* valid for one clock cycle

Note: With **TIM_EN**=1 the MAP interface starts operation, it is not dependent on the setting of the bit fields **TIM_MODE, ISL, DSL**.

Note: While the MAP interface is in use the following guidelines have to be fulfilled, otherwise inconsistent filter values can be transferred.
Change **TIM0_CH[x]_FLT_RE** only between occurrence of rising and falling edge.
Change **TIM0_CH[x]_FLT_FE** only between occurrence of falling and rising edge.

11.6 TIM Interrupt Signals

11.6.1 TIM Interrupt Signals Table

Signal	**Description**
TIM[i]_NEWVAL[x]_IRQ	New measurement value detected by SMU of channel x (x: 0...m-1)
TIM[i]_ECNTOFL[x]_IRQ	ECNT counter overflow of channel x (x: 0...m-1)
TIM[i]_CNTOFL[x]_IRQ	SMU CNT counter overflow of channel x (x: 0...m-1)
TIM[i]_GPROFL[x]_IRQ	GPR0 and GPR1 data overflow, old data was not read out before new data has arrived at input pin of channel x (x: 0...m-1)
TIM[i]_TODET[x]_IRQ	Time out reached for input signal of channel x (x: 0...m-1)
TIM[i]_GLITCHDET[x]_IRQ	A glitch was detected by the TIM filter of channel x (x: 0...m-1)

11.7 TIM Configuration Register Overview

11.7.1 TIM Configuration Register Overview Table

Register Name	Description	Detail in Section
TIM[i]_CH[x]_CTRL	TIMi channel x control register	11.8.1
TIM0_CH[x]_CTRL	TIM 0 channel x control register	11.8.2
TIM[i]_CH[x]_ECTRL	TIMi channel x extended control register	11.8.19
TIM[i]_CH[x]_FLT_RE	TIMi channel x filter parameter 0 register	11.8.3
TIM[i]_CH[x]_FLT_FE	TIMi channel x filter parameter 1 register	11.8.4
TIM[i]_CH[x]_TDUV	TIMi channel x TDU control register	11.8.16
TIM[i]_CH[x]_TDUC	TIMi channel x TDU counter register	11.8.17
TIM[i]_CH[x]_GPR0	TIMi channel x general purpose 0 register	11.8.5
TIM[i]_CH[x]_GPR1	TIMi channel x general purpose 1 register	11.8.6
TIM[i]_CH[x]_CNT	TIMi channel x SMU counter register	11.8.7
TIM[i]_CH[x]_ECNT	TIMi channel x SMU edge counter register	11.8.18
TIM[i]_CH[x]_CNTS	TIMi channel x SMU shadow counter register	11.8.8
TIM[i]_CH[x]_IRQ_NOTIFY	TIMi channel x interrupt notification register	11.8.9
TIM[i]_CH[x]_IRQ_EN	TIMi channel x interrupt enable register	11.8.10
TIM[i]_CH[x]_EIRQ_EN	TIMi channel x error interrupt enable register	11.8.15
TIM[i]_CH[x]_IRQ_FORCINT	TIMi channel x force interrupt register	11.8.11
TIM[i]_CH[x]_IRQ_MODE	TIMi interrupt mode configuration register	11.8.12
TIM[i]_RST	TIMi global software reset register	11.8.11

TIM[i]_IN_SRC	TIMi AUX IN source selection register	11.8.14
TIM[i]_INP_VAL	TIMi input value observation register	11.8.20

11.8 TIM Configuration Registers Description

11.8.1 Register TIM[i]_CH[x]_CTRL

Address Offset:	**see Appendix B**																**Initial Value: 0x0000_0000**															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	TOCTRL		EGPR1_SEL	EGPR0_SEL	FR_ECNT_OFL	CLK_SEL			FLT_CTR_FE	FLT_MODE_FE	FLT_CTR_RE	FLT_MODE_RE	EXT_CAP_EN	FLT_CNT_FRQ		FLT_EN	ECNT_RESET	ISL	DSL	CNTS_SEL	GPR1_SEL		GPR0_SEL		Reserved	CICTRL	ARU_EN	OSM	TIM_MODE			TIM_EN
Mode	RW		RW	RW	RW	RW			RW	RW	RW	RW	RW	RW		RW	RW	RW	RW	RW	RW		RW		R	RW	RW	RW	RW			RW
Initial Value	00		0	0	0	000			0	0	0	0	0	00		0	0	0	0	0	00		00		0	0	0	0	000			0

Bit 0 **TIM_EN:** TIM channel x (x:0...7) enable
0 = Channel disabled
1 = Channel enabled
Note: Enabling of the channel resets the registers **ECNT, TIM[i]_CH[x]_CNT, TIM[i]_CH[x]_GPR0,** and **TIM[i]_CH[x]_GPR1** to their reset values.

Note: After finishing the action in one-shot mode the **TIM_EN** bit is cleared automatically. Otherwise, the bit must be cleared manually.

Bit 3:1 **TIM_MODE:** TIM channel x (x:0...7) mode
0b000 = PWM Measurement Mode (TPWM)
0b001 = Pulse Integration Mode (TPIM)
0b010 = Input Event Mode (TIEM)
0b011 = Input Prescaler Mode (TIPM)
0b100 = Bit Compression Mode (TBCM)
0b101 = Gated Periodic Sampling Mode (TGPS)
0b110 = Serial Shift Mode (TSSM)

Note: If an undefined value is written to the TIM_MODE register, the hardware switches automatically to TIM_MODE = 0b000 (TPWM mode).

Note: The TIM_MODE register should not be changed while the TIM channel is enabled.

Note: If the TIM channel is enabled and operating in TPWM or TPIM mode after the first valid edge defined by DSL has occurred, a reconfiguration of DSL,ISL,TIM_MODE will not change the channel behavior. Reading these bit fields after reconfiguration will show the newly configured settings but the initial channel behavior will not change. Only a disabling of the TIM channel by setting TIM_EN= 0 and reenabling with TIM_EN= 1 will change the channel operation mode.

Bit 4 **OSM:** One-shot mode
0 = Continuous operation mode
1 = One-shot mode
Note: After finishing the action in one-shot mode the **TIM_EN** bit is cleared automatically.

Bit 5 **ARU_EN:** GPR0 and GPR1 register values routed to ARU
0 = Registers content not routed
1 = Registers content routed

Bit 6 **CICTRL:** Channel Input Control.
0 = use signal TIM_IN(x) as input for channel x
1 = use signal TIM_IN(x-1) as input for channel x (or TIM_IN(m-1) if x is 0)

Bit 7 **Reserved:** Reserved
Note: Read as zero, should be written as zero

Bit 9:8 **GPR0_SEL:** Selection for GPR0 register
EGPR0_SEL =0 / EGPR0_SEL =1 :
0b00 = use TBU_TS0 as input / use ECNT as input
0b01 = use TBU_TS1 as input / use TIM_INP_VAL as input
0b10 = use TBU_TS2 as input / reserved
0b11 = use CNTS as input; if TGPS mode in channel 0 is selected use TIM Filter F_OUT as input / reserved

Note: If a reserved value is written to the EGPR0_SEL, GPR0_SEL bit fields , the hardware will use TBU_TS0 input.

Bit 11:10 **GPR1_SEL:** Selection for GPR1 register
EGPR1_SEL =0 / EGPR1_SEL =1 :
0b00 = use TBU_TS0 as input / use ECNT as input
0b01 = use TBU_TS1 as input / use TIM_INP_VAL as input
0b10 = use TBU_TS2 as input / reserved
0b11 = use CNT as input / reserved

Note: In TBCM mode: EGPR1_SEL=1, GPR1_SEL=01 selects TIM_INP_VAL as input, in all other cases TIM Filter F_OUT is used

Note: If a reserved value is written to the EGPR1_SEL, GPR1_SEL bit fields, the hardware will use TBU_TS0 input.

Bit 12 **CNTS_SEL:** Selection for CNTS register
0 = use CNT register as input
1 = use TBU_TS0 as input
Note: **CNTS_SEL** in TSSM mode selects the source signal for registered or latched shift out operation.
0 = use F_OUTx
1 = use TIM_INx

Bit 13 **DSL:** Signal level control
0 = Measurement starts with falling edge (low level measurement)
1 = Measurement starts with rising edge (high level measurement)

Note: In TIM_MODE=0b110 (TSSM) the bit field DSL defines the shift direction.
0 = Shift left
1 = Shift right

Bit 14 **ISL:** Ignore signal level
0 = use DSL bit for selecting active signal level (TIEM)
1 = ignore DSL and treat both edges as active edge (TIEM)
This bit is mode dependent and will have different meanings (see details in the TIM Channel mode description).

Bit 15 **ECNT_RESET:** Enables resetting of counter in certain modes
If TIM_MODE=0b101 (TGPS) / TIM_MODE=0b000 (TPWM)
0 = ECNT counter operating in wrap around mode / ECNT counter operating in wrap around mode, CNT is reset on active input edge defined by DSL
1 = ECNT counter is reset with periodic sampling / ECNT counter operating in wrap around mode, CNT is reset on active and inactive input edge
else
ECNT counter operating in wrap around mode;

Note: In TIM_MODE=0b110 (TSSM) the bit field ECNT_RESET defines the initial polarity for the shift register.

Bit 16 **FLT_EN:** Filter enable for channel x (x:0...7)
0 = Filter disabled and internal states are reset
1 = Filter enabled

Note: If the filter is disabled all filter related units (including CSU) are bypassed, which means that the signal *F_IN* is directly routed to signal *F_OUT*.

Bit 18:17 **FLT_CNT_FRQ:** Filter counter frequency select
0b00 = FLT_CNT counts with *CMU_CLK0*
0b01 = FLT_CNT counts with *CMU_CLK1*
0b10 = FLT_CNT counts with *CMU_CLK6*
0b11 = FLT_CNT counts with *CMU_CLK7*

Bit 19 **EXT_CAP_EN:** Enables external capture mode. The selected TIM mode is only sensitive to external capture pulses the input event changes are ignored.
0 = External capture disabled
1 = External capture enabled

Bit 20 **FLT_MODE_RE:** Filter mode for rising edge.
0 = Immediate edge propagation mode
1 = individual de-glitch mode

Bit 21 **FLT_CTR_RE:** Filter counter mode for rising edge.
If FLT_MODE_RE=1 / FLT_MODE_RE=0
EFLT_CTR_RE,FLT_CTR_RE:
0b00 = Up-Down Counter individual de-glitch mode / Immediate edge propagation mode
0b01 = Hold Counter individual de-glitch mode / Immediate edge propagation mode
0b10 = Reset Counter individual de-glitch mode / reserved
0b11 = reserved / reserved

Bit 22 **FLT_MODE_FE:** Filter mode for falling edge.
0 = Immediate edge propagation mode
1 = individual de-glitch mode

Bit 23 **FLT_CTR_FE:** Filter counter mode for falling edge.
If FLT_MODE_FE=1 / FLT_MODE_FE=0
EFLT_CTR_FE ,FLT_CTR_FE:
0b00 = Up-Down Counter individual de-glitch mode / Immediate edge propagation mode
0b01 = Hold Counter individual de-glitch mode / Immediate edge propagation mode
0b10 = Reset Counter individual de-glitch mode / reserved
0b11 = reserved / reserved

Bit 26:24 **CLK_SEL:** CMU clock source select for channel.
If ECLK_SEL =0 / ECLK_SEL =1:
0b000 = *CMU_CLK0* selected / tdu_sample_evt of TDU selected
0b001 = *CMU_CLK1* selected / reserved
0b010 = *CMU_CLK2* selected / reserved
0b011 = *CMU_CLK3* selected / reserved

0b100 = *CMU_CLK4* selected / reserved
0b101 = *CMU_CLK5* selected / reserved
0b110 = *CMU_CLK6* selected / reserved
0b111 = *CMU_CLK7* selected / reserved

Bit 27 **FR_ECNT_OFL:** Extended Edge counter overflow behavior
0 = Overflow will be signaled on ECNT bit width = 8
1 = Overflow will be signaled on EECNT bit width (full range)

Bit 28 **EGPR0_SEL:** Extension of GPR0_SEL bit field.
Details described in GPR0_SEL bit field.

Bit 29 **EGPR1_SEL:** Extension of GPR1_SEL bit field.
Details described in GPR1_SEL bit field.

Bit 31:30 **TOCTRL:** Timeout control
0b00 = Timeout feature disabled
0b11 = Timeout feature enabled for both edges
0b01 = Timeout feature enabled for rising edge only
0b10 = Timeout feature enabled for falling edge only
Note: It has to mention that writing of TOCTRL= 0 will every time stop the TDU, independent of the previous state of TOCTRL.

11.8.2 Register TIM[i]_CH[x]_CTRL (i:0)

Address Offset:	**see Appendix B**																**Initial Value:**				**0x0000_0000**											
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	TOCTRL		EGPR1_SEL	EGPR0_SEL	FR_ECNT_OFL	CLK_SEL			FLT_CTR_FE	FLT_MODE_FE	FLT_CTR_RE	FLT_MODE_RE	EXT_CAP_EN	FLT_CNT_FRQ		FLT_EN	ECNT_RESET	ISL	DSL	CNTS_SEL	GPR1_SEL		GPR0_SEL		TBU0_SEL	CICTRL	ARU_EN	OSM	TIM_MODE			TIM_EN
Mode	RW		RW	RW	RW	RW			RW	RW	RW	RW	RW	RW		RW	RW	RW	RW	RW	RW		RW		RW	RW	RW	RW	RW			RW
Initial Value	00		0	0	0	000			0	0	0	0	0	00		0	0	0	0	0	00		00		0	0	0	0	000			0

Bit 0 **TIM_EN:** TIM channel x (x:0...m-1) enable
0 = Channel disabled
1 = Channel enabled
Note: Enabling of the channel resets the registers **ECNT, TIM[i]_CH[x]_CNT, TIM[i]_CH[x]_GPR0,** and **TIM[i]_CH[x]_GPR1** to their reset values.

Note: After finishing the action in one-shot mode the **TIM_EN** bit is cleared automatically. Otherwise, the bit must be cleared manually.

Bit 3:1 **TIM_MODE:** TIM channel x (x:0...m-1) mode
0b000 = PWM Measurement Mode (TPWM)
0b001 = Pulse Integration Mode (TPIM)
0b010 = Input Event Mode (TIEM)
0b011 = Input Prescaler Mode (TIPM)
0b100 = Bit Compression Mode (TBCM)
0b101 = Gated Periodic Sampling Mode (TGPS)
0b110 = Serial Shift Mode (TSSM)

Note: If an undefined value is written to the TIM_MODE register, the hardware switches automatically to TIM_MODE = 0b000 (TPWM mode).

Note: The TIM_MODE register should not be changed while the TIM channel is enabled.

Note: If the TIM channel is enabled and operating in TPWM or TPIM mode after the first valid edge defined by DSL has occurred, a reconfiguration of DSL,ISL,TIM_MODE will not change the channel behavior. Reading these bit fields after reconfiguration will show the newly configured settings but the initial channel behavior will not change. Only a disabling of the TIM channel by setting TIM_EN= 0 and reenabling with TIM_EN= 1 will change the channel operation mode.

Bit 4 **OSM:** One-shot mode
0 = Continuous operation mode
1 = One-shot mode
Note: After finishing the action in one-shot mode the **TIM_EN** bit is cleared automatically.

Bit 5 **ARU_EN:** GPR0 and GPR1 register values routed to ARU
0 = Registers content not routed
1 = Registers content routed

Bit 6 **CICTRL:** Channel Input Control.
0 = use signal TIM_IN(x) as input for channel x
1 = use signal TIM_IN(x-1) as input for channel x (or TIM_IN(m-1) if x is 0)

Bit 7 **TBU0_SEL:** *TBU_TS0* bits input select for TIM0_CH[x]_GPRz (z: 0, 1)
0 = Use *TBU_TS0(23..0)* to store in **TIM0_CH[x]_GPR0/TIM0_CH[x]_GPR1**
1 = Use *TBU_TS0(26..3)* to store in **TIM0_CH[x]_GPR0/TIM0_CH[x]_GPR1**

Note: This bit is only applicable for TIM0

Bit 9:8 **GPR0_SEL:** Selection for GPR0 register
If EGPR0_SEL =0 / EGPR0_SEL =1 :

0b00 = use TBU_TS0 as input / use ECNT as input
0b01 = use TBU_TS1 as input / use TIM_INP_VAL as input
0b10 = use TBU_TS2 as input / reserved
0b11 = use CNTS as input; if TGPS mode in channel = 0 is selected use TIM Filter F_OUT as input / reserved

Note: If a reserved value is written to the EGPR0_SEL, GPR0_SEL bit fields , the hardware will use TBU_TS0 input.

Bit 11:10 **GPR1_SEL:** Selection for GPR1 register
If EGPR1_SEL =0 / EGPR1_SEL =1:
0b00 = use TBU_TS0 as input / use ECNT as input
0b01 = use TBU_TS1 as input / use TIM_INP_VAL as input
0b10 = use TBU_TS2 as input / reserved
0b11 = use CNT as input / reserved
Note: In TBCM mode: EGPR1_SEL=1, GPR1_SEL=01 selects TIM_INP_VAL as input, in all other cases TIM Filter F_OUT is used

Note: If a reserved value is written to the EGPR1_SEL, GPR1_SEL bit fields , the hardware will use TBU_TS0 input.

Bit 12 **CNTS_SEL:** Selection for CNTS register
0 = use CNT register as input
1 = use TBU_TS0 as input
Note: The functionality of the **CNTS_SEL** is disabled in the modes TIPM,TGPS and TBCM.

Note: **CNTS_SEL** in TSSM mode selects the source signal for registered or latched shift out operation.
0 = use F_OUTx
1 = use TIM_INx

Bit 13 **DSL:** Signal level control
0 = Measurement starts with falling edge (low level measurement)
1 = Measurement starts with rising edge (high level measurement)

Note: In TIM_MODE=0b110 (TSSM) the bit field DSL defines the shift direction.
0 = Shift left
1 = Shift right

Bit 14 **ISL:** Ignore signal level
0 = use DSL bit for selecting active signal level (TIEM)
1 = ignore DSL and treat both edges as active edge (TIEM)
This bit is mode dependent and will have different meanings (see details in the TIM Channel mode description).

Bit 15 **ECNT_RESET:** Enables resetting of counter in certain modes
If TIM_MODE=0b101 (TGPS) / TIM_MODE=0b000 (TPWM)
0 = ECNT counter operating in wrap around mode / ECNT counter operating in wrap around mode, CNT is reset on active input edge defined by DSL
1 = ECNT counter is reset with periodic sampling / ECNT counter operating in wrap around mode, CNT is reset on active and inactive input edge
else
ECNT counter operating in wrap around mode;

Note: In TIM_MODE=0b110 (TSSM) the bit field ECNT_RESET defines the initial polarity for the shift register.

Bit 16 **FLT_EN:** Filter enable for channel x (x:0...m-1)
0 = Filter disabled and internal states are reset
1 = Filter enabled
Note: If the filter is disabled all filter related units (including CSU) are bypassed, which means that the signal *F_IN* is directly routed to signal *F_OUT*.

Bit 18:17 **FLT_CNT_FRQ:** Filter counter frequency select
0b00 = FLT_CNT counts with *CMU_CLK0*
0b01 = FLT_CNT counts with *CMU_CLK1*
0b10 = FLT_CNT counts with *CMU_CLK6*
0b11 = FLT_CNT counts with *CMU_CLK7*

Bit 19 **EXT_CAP_EN:** Enables external capture mode. The selected TIM mode is only sensitive to external capture pulses the input event changes are ignored.
0 = External capture disabled
1 = External capture enabled

Bit 20 **FLT_MODE_RE:** Filter mode for rising edge.
0 = Immediate edge propagation mode
1 = individual de-glitch mode

Bit 21 **FLT_CTR_RE:** Filter counter mode for rising edge.
If FLT_MODE_RE=1 / FLT_MODE_RE=0
EFLT_CTR_RE ,FLT_CTR_RE:
0b00 = Up-Down Counter individual de-glitch mode / Immediate edge propagation mode
0b01 = Hold Counter individual de-glitch mode / Immediate edge propagation mode
0b10 = Reset Counter individual de-glitch mode / reserved
0b11 = reserved / reserved

Bit 22 **FLT_MODE_FE:** Filter mode for falling edge.

0 = Immediate edge propagation mode
1 = individual de-glitch mode

Bit 23 **FLT_CTR_FE:** Filter counter mode for falling edge.
If FLT_MODE_FE=1 / FLT_MODE_FE=0
EFLT_CTR_FE ,FLT_CTR_FE:
0b00 = Up-Down Counter individual de-glitch mode / Immediate edge propagation mode
0b01 = Hold Counter individual de-glitch mode / Immediate edge propagation mode
0b10 = Reset Counter individual de-glitch mode / reserved
0b11 = reserved / reserved

Bit 26:24 **CLK_SEL:** CMU clock source select for channel.
If ECLK_SEL =0 / ECLK_SEL =1:
0b000 = *CMU_CLK0* selected / tdu_sample_evt of TDU selected
0b001 = *CMU_CLK1* selected / reserved
0b010 = *CMU_CLK2* selected / reserved
0b011 = *CMU_CLK3* selected / reserved
0b100 = *CMU_CLK4* selected / reserved
0b101 = *CMU_CLK5* selected / reserved
0b110 = *CMU_CLK6* selected / reserved
0b111 = *CMU_CLK7* selected / reserved

Bit 27 **FR_ECNT_OFL:** Extended Edge counter overflow behavior
0 = Overflow will be signaled on ECNT bit width = 8
1 = Overflow will be signaled on EECNT bit width (full range)

Bit 28 **EGPR0_SEL:** Extension of GPR0_SEL bit field.
Details described in GPR0_SEL bit field.

Bit 29 **EGPR1_SEL:** Extension of GPR1_SEL bit field.
Details described in GPR1_SEL bit field.

Bit 31:30 **TOCTRL:** Timeout control
0b00 = Timeout feature disabled
0b01 = Timeout feature enabled for rising edge only
0b10 = Timeout feature enabled for falling edge only
0b11 = Timeout feature enabled for both edges

Note: It has to mention that writing of TOCTRL= 0 will every time stop the TDU, independent of the previous state of TOCTRL.

11.8.3 Register TIM[i]_CH[x]_FLT_RE

Address Offset:	see Appendix B																Initial Value: 0x0000_0000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								FLT_RE																							
Mode	R								RW																							
Initial Value	0x00								0x000000																							

Bit 23:0 **FLT_RE:** Filter parameter for rising edge.
Note: FLT_RE has different meanings in the various filter modes.
Immediate edge propagation mode = acceptance time for rising edge
Individual deglitch time mode = deglitch time for rising edge

Bit 31:24 **Reserved:** Reserved
Note: Read as zero, should be written as zero

11.8.4 Register TIM[i]_CH[x]_FLT_FE

Address Offset:	see Appendix B																Initial Value: 0x0000_0000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								FLT_FE																							
Mode	R								RW																							
Initial Value	0x00								0x000000																							

Bit 23:0 **FLT_FE:** Filter parameter for falling edge.
Note: FLT_FE has different meanings in the various filter modes.
Immediate edge propagation mode = acceptance time for falling edge
Individual deglitch time mode = deglitch time for falling edge

Bit 31:24 **Reserved:** Reserved
Note: Read as zero, should be written as zero

11.8.5 Register TIM[i]_CH[x]_GPR0

Address Offset:	**see Appendix B**																**Initial Value:**							**0x0X00_0000**								
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	ECNT								GPR0																							
Mode	R								R																							
Initial Value	0x00								0x0000 00																							

Bit 23:0 **GPR0:** Input signal characteristic parameter 0.

Note: The content of this register has different meaning for the TIM channels modes. The content directly depends on the bit fields **EGPR0_SEL**, **GPR0_SEL** of register **TIM[i]_CH[x]_CTRL**.

Bit 31:24 **ECNT:** Edge counter.

Note: The **ECNT** counts every incoming filtered edge (rising and falling). The counter value is uneven in case of detected rising, and even in case of detected falling edge. Thus, the input signal level is part of the counter and can be obtained by bit 0 of **ECNT**.

Note: The **ECNT** register is reset to its initial value when the channel is enabled. Please note, that bit 0 depends on the input level coming from the filter unit and defines the reset value immediately.

11.8.6 Register TIM[i]_CH[x]_GPR1

Address Offset:	**see Appendix B**																**Initial Value:**							**0x0X00_0000**								
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	ECNT								GPR1																							
Mode	R								RPw																							
Initial Value	0x00								0x0000 00																							

Bit 23:0 **GPR1:** Input signal characteristic parameter 1.

Note: The content of this register has different meaning for the TIM channels modes. The content directly depends on the bit fields **EGPR1_SEL**, **GPR1_SEL** of register **TIM[i]_CH[x]_CTRL**.

Note: In TBCM mode if EGPR1_SEL=1, GPR1_SEL=01 then TIM_INP_VAL is used as input in all other cases TIM Filter F_OUT is used as input and Bits GPR1(23:8) = 0

Note: The content of this register can only be written in TIM channel mode TGPS and TSSM.

Bit 31:24 **ECNT:** Edge counter.

Note: The **ECNT** counts every incoming filtered edge (rising and falling). The counter value is uneven in case of detected rising, and even in case of detected falling edge. Thus, the input signal level is part of the counter and can be obtained by bit 0 of **ECNT**.

Note: The **ECNT** register is reset to its initial value when the channel is enabled. Please note, that bit 0 depends on the input level coming from the filter unit and defines the reset value immediately.

11.8.7 Register TIM[i]_CH[x]_CNT

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								CNT																							
Mode	R								R																							
Initial Value	0x00								0x000000																							

Bit 23:0 **CNT:** Actual SMU counter value

Note: The meaning of this value depends on the configured mode:

TPWM = actual duration of PWM signal.

TPIM = actual duration of all pulses (sum of pulses).

TIEM = actual number of received edges.

TIPM = actual number of received edges.

TGPS = elapsed time for periodic sampling.

TSSM = shift data.

Bit 31:24 **Reserved:** Reserved
Note: Read as zero, should be written as zero

11.8.8 Register TIM[i]_CH[x]_CNTS

Address Offset:	**see Appendix B**																**Initial Value: 0x0X00_0000**															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	ECNT								CNTS																							
Mode	R								RPw																							
Initial Value	0b0000 000X								0x0000 00																							

Bit 23:0 **CNTS:** Counter shadow register.
Note: The content of this register has different meaning for the TIM channels modes. The content depends directly on the bit field **CNTS_SEL** of register **TIM[i]_CH[x]_CTRL**.

Note: The register **TIM[i]_CH[x]_CNTS** is only writable in TIPM,TBCM ,TGPS and TSSM mode.

Bit 31:24 **ECNT:** Edge counter.
Note: The **ECNT** counts every incoming filtered edge (rising and falling). The counter value is uneven in case of detected rising, and even in case of detected falling edge. Thus, the input signal level is part of the counter and can be obtained by bit 0 of **ECNT**.

Note: The **ECNT** register is reset to its initial value when the channel is enabled. Please note, that bit 0 depends on the input level coming from the filter unit and defines the reset value immediately.

11.8.9 Register TIM[i]_CH[x]_IRQ_NOTIFY

Address Offset:	see Appendix B																Initial Value: 0x0000_0000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																										GLITCHDET	TODET	GPROFL	CNTOFL	ECNTOFL	NEWVAL
Mode	R																										RCw	RCw	RCw	RCw	RCw	RCw
Initial Value	0x0000 000																										0	0	0	0	0	0

Bit 0 **NEWVAL:** New measurement value detected by in channel x (x:0...m-1)
0 = No event was occurred
1 = *NEWVAL* was occurred on the TIM channel
Note: This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged.

Bit 1 **ECNTOFL: ECNT** counter overflow of channel x, (x:0...m-1). See bit 0.

Bit 2 **CNTOFL:** SMU **CNT** counter overflow of channel x, (x:0...m-1). See bit 0.

Bit 3 **GPROFL: GPR0** and **GPR1** data overflow, old data not read out before new data has arrived at input pin, (x:0...m-1). See bit 0.

Bit 4 **TODET:** Timeout reached for input signal of channel x, (x:0...m-1). See bit 0.

Bit 5 **GLITCHDET:** Glitch detected on channel x, (x:0...m-1).
0 = no glitch detected for last edge
1 = glitch detected for last edge
Note: This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged.

Bit 31:6 **Reserved:** Reserved
Note: Read as zero, should be written as zero

11.8.10 Register TIM[i]_CH[x]_IRQ_EN

Address Offset:	see Appendix B																Initial Value:	0x0000_0000													
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																										GLITCHDET_IRQ_	TODET_IRQ_EN	GPROFL_IRQ_EN	CNTOFL_IRQ_EN	ECNTOFL_IRQ_E	NEWVAL_IRQ_EN
Mode	R																										RW	RW	RW	RW	RW	RW
Initial Value	0x0000 000																										0	0	0	0	0	0

Bit 0 **NEWVAL_IRQ_EN:** *TIM_NEWVALx_IRQ* interrupt enable
0 = Disable interrupt, interrupt is not visible outside GTM-IP
1 = Enable interrupt, interrupt is visible outside GTM-IP

Bit 1 **ECNTOFL_IRQ_EN:** *TIM_ECNTOFLx_IRQ* interrupt enable, see bit 0.
Bit 2 **CNTOFL_IRQ_EN:** *TIM_CNTOFLx_IRQ* interrupt enable, see bit 0.
Bit 3 **GPROFL_IRQ_EN:** *TIM_GPROFL_IRQ* interrupt enable, see bit 0.
Bit 4 **TODET_IRQ_EN:** *TIM_TODETx_IRQ* interrupt enable, see bit 0.
Bit 5 **GLITCHDET_IRQ_EN:** *TIM_GLITCHDETx_IRQ* interrupt enable, see bit 0.
Bit 31:6 **Reserved:** Reserved
Note: Read as zero, should be written as zero

11.8.11 Register TIM[i]_CH[x]_IRQ_FORCINT

Address Offset:	see Appendix B																Initial Value:	0x0000_0000													
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																										TRG_GLITCHDET	TRG_TODET	TRG_GPROFL	TRG_CNTOFL	TRG_ECNTOFL	TRG_NEWVAL
Mode	R																										RAw	RAw	RAw	RAw	RAw	RAw
Initial Value	0x0000 00																										0	0	0	0	0	0

Bit 0 **TRG_NEWVAL:** Trigger **NEWVAL** bit in **TIM_CHx_IRQ_NOTIFY** register by software
0 = No interrupt triggering
1 = Assert corresponding field in **TIM[i]_CH[x]_IRQ_NOTIFY** register

Note: This bit is cleared automatically after write.
Note: This bit is write protected by bit RF_PROT of register GTM_CTRL

Bit 1 **TRG_ECNTOFL:** Trigger **ECNTOFL** bit in **TIM_CHx_IRQ_NOTIFY** register by software, see bit 0.

Bit 2 **TRG_CNTOFL:** Trigger **CNTOFL** bit in **TIM_CHx_IRQ_NOTIFY** register by software, see bit 0.

Bit 3 **TRG_GPROFL:** Trigger **GPROFL** bit in **TIM_CH[x]_IRQ_NOTIFY** register by software, see bit 0.

Bit 4 **TRG_TODET:** Trigger **TODET** bit in **TIM_CHx_IRQ_NOTIFY** register by software, see bit 0.

Bit 5 **TRG_GLITCHDET:** Trigger **GLITCHDET** bit in **TIM_CHx_IRQ_NOTIFY** register by software, see bit 0.

Bit 31:6 **Reserved:** Reserved
Note: Read as zero, should be written as zero

11.8.12 Register TIM[i]_CH[x]_IRQ_MODE

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_000X**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																														IRQ_MODE	
Mode	R																														RW	
Initial Value	0x0000 0000																														XX	

Bit 1:0 **IRQ_MODE**: IRQ mode selection
0b00 = Level mode
0b01 = Pulse mode
0b10 = Pulse-Notify mode
0b11 = Single-Pulse mode
Note: The interrupt modes are described in section 2.5.

Bit 31:2 **Reserved:** Reserved
Note: Read as zero, should be written as zero

11.8.13 Register TIM[i]_RST

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_0000**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																								RST_CH7	RST_CH6	RST_CH5	RST_CH4	RST_CH3	RST_CH2	RST_CH1	RST_CH0
Mode	R																								RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw
Initial Value	0x0000 00																								0	0	0	0	0	0	0	0

Bit 0 **RST_CH0:** Software reset of channel 0

0 = No action

1 = Reset channel 0

Note: This bit is cleared automatically after write by CPU. The channel registers are set to their reset values and channel operation is stopped immediately.

Bit 1 **RST_CH1:** Software reset of channel 1, see bit 0.
Bit 2 **RST_CH2:** Software reset of channel 2, see bit 0.
Bit 3 **RST_CH3:** Software reset of channel 3, see bit 0.
Bit 4 **RST_CH4:** Software reset of channel 4, see bit 0.
Bit 5 **RST_CH5:** Software reset of channel 5, see bit 0.
Bit 6 **RST_CH6:** Software reset of channel 6, see bit 0.
Bit 7 **RST_CH7:** Software reset of channel 7, see bit 0.

Note: Please note, that the RST field width of this register depends on the number of implemented channels *m* within this sub-module. This register description represents a register layout for *m* = 8.

Bit 31:8 **Reserved:** Reserved

Note: Read as zero, should be written as zero

11.8.14 Register TIM[i]_IN_SRC

Address Offset:	**see Appendix B**								**Initial Value:**				**0x0000_0000**			
	31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16	15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
Bit	MODE_7	VAL_7	MODE_6	VAL_6	MODE_5	VAL_5	MODE_4	VAL_4	MODE_3	VAL_3	MODE_2	VAL_2	MODE_1	VAL_1	MODE_0	VAL_0
Mode	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Initial Value	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Bit 1:0 **VAL_0:** Value to be fed to Channel 0

Multi-core encoding in use (**VAL_x(1)** defines the state of the signal)
0b00 = State is 0 (ignore write access)
0b01 = Change state to 0
0b10 = Change state to 1
0b11 = State is 1 (ignore write access)
Function depends on the combination of **VAL_x(1)** and **MODE_x(1)** see **MODE_0** description.

Note: Any read access to a **VAL_x** bit field will always result in a value 00 or 11 indicating current state. A modification of the state is only performed with the values 01 and 10. Writing the values 00 and 11 is always ignored.

Bit 3:2 **MODE_0:** Input source to Channel 0

Multi-core encoding in use (**MODE_x(1)** defines the state of the signal)
0b00 = State is 0 (ignore write access)
0b01 = Change state to 0
0b10 = Change state to 1
0b11 = State is 1 (ignore write access)

Function table:

MODE_x(1)=0 , VAL_x(1)=0: The input signal defined by bit field **CICTRL** of the TIM channel is used as input source.

MODE_x(1)=0 , VAL_x(1)=1: The signal TIM_AUX_IN of the TIM channel is used as input source.

MODE_x(1)=1 : The state **VAL_x(1)** defines the input level for the TIM channel.

Note: Any read access to a **MODE_x** bit field will always result in a value 00 or 11 indicating current state. A modification of the state is only performed with the values 01 and 10. Writing the values 00 and 11 is always ignored.

Bit 5:4 **VAL_1:** Value to be fed to Channel 1, see bits 1:0.

Bit 7:6 **MODE_1:** Input source to Channel 1, see bits 3:2.

Bit 9:8 **VAL_2:** Value to be fed to Channel 2, see bits 1:0.

Bit 11:10 **MODE_2:** Input source to Channel 2, see bits 3:2.

Bit 13:12 **VAL_3:** Value to be fed to Channel 3, see bits 1:0.

Bit 15:14 **MODE_3:** Input source to Channel 3, see bits 3:2.

Bit 17:16 **VAL_4:** Value to be fed to Channel 4, see bits 1:0.

Bit 19:18 **MODE_4:** Input source to Channel 4, see bits 3:2.

Bit 21:20 **VAL_5:** Value to be fed to Channel 5, see bits 1:0.

Bit 23:22 **MODE_5:** Input source to Channel 5, see bits 3:2.

Bit 25:24 **VAL_6:** Value to be fed to Channel 6, see bits 1:0.

Bit 27:26 **MODE_6:** Input source to Channel 6, see bits 3:2.

Bit 29:28 **VAL_7:** Value to be fed to Channel 7, see bits 1:0.

Bit 31:30 **MODE_7:** Input source to Channel 7, see bits 3:2.

11.8.15 Register TIM[i]_CH[x]_EIRQ_EN

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_0000**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																										GLITCHDET_EIRQ	TODET_EIRQ_EN	GPROFL_EIRQ_E	CNTOFL_EIRQ_E	ECNTOFL_EIRQ_	NEWVAL_EIRQ_E
Mode	R																										RW	RW	RW	RW	RW	RW
Initial Value	0x0000 000																										0	0	0	0	0	0

Bit 0 **NEWVAL_EIRQ_EN:** *TIM_NEWVALx_EIRQ* error interrupt enable
0 = Disable error interrupt, error interrupt is not visible outside GTM-IP

1 = Enable error interrupt, error interrupt is visible outside GTM-IP

Bit 1 **ECNTOFL_EIRQ_EN:** *TIM_ECNTOFLx_IRQ* interrupt enable, see bit 0.
Bit 2 **CNTOFL_EIRQ_EN:** *TIM_CNTOFLx_IRQ* interrupt enable, see bit 0.
Bit 3 **GPROFL_EIRQ_EN:** *TIM_GPROFL_IRQ* interrupt enable, see bit 0.
Bit 4 **TODET_EIRQ_EN:** *TIM_TODETx_IRQ* interrupt enable, see bit 0.
Bit 5 **GLITCHDET_EIRQ_EN:** *TIM_GLITCHDETx_IRQ* interrupt enable, see bit 0.
Bit 31:6 **Reserved:** Reserved
Note: Read as zero, should be written as zero

11.8.16 Register TIM[i]_CH[x]_TDUV

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved	TCS			TDU_SAME_CNT_	TCS_USE_SAMPL	SLICING		TOV2								TOV1								TOV							
Mode	R	RW			RW	RW	RW		RW								RW								RW							
Initial Value	0	000			0	0	00		0x00								0x00								0x00							

Bit 7:0 **TOV:** Time out compare value slice0 for channel x (x:0...m-1).
Compare value for TO_CNT

Bit 15:8 **TOV1:** Time out compare value slice1 for channel x (x:0...m-1).
Compare value for TO_CNT1

Bit 23:16 **TOV2:** Time out compare value slice2 for channel x (x:0...m-1).
SLICING != 0b11: Compare value for TO_CNT2
SLICING = 0b11 : TOV2 operate as a shadow register for TO_CNT

Bit 25:24 **SLICING:** Cascading of counter slices
Slicing by use_lut
If USE_LUT=0b00 / USE_LUT !=0b00
SLICING:
0b00 = combine slice2,slice1,slice0 to 1x24 bit counter / reserved
0b01 = combine slice1,slice0 to 1x16bit counter use slice2 as 1x8 bit counter / combine slice1,slice0 to 1x16bit slice2 not usable
0b10 = use slice2,slice1,slice0 as 3x8 bit counter / use slice1,slice0 as 2x8 bit counter slice2 not usable
0b11 = use slice1,slice0 as 2x8 bit counter / use slice1,slice0 as 2x8 bit counter

Bit 26 **TCS_USE_SAMPLE_EVT:** Use tdu_sample_evt as Timeout Clock
0 = CMU_CLK selected by TCS is in use by TO_CNT,TO_CNT2
1 = CMU_CLK selected by TCS is in use by TO_CNT2;
tdu_sample_evt is in use by TO_CNT

Bit 27 **TDU_SAME_CNT_CLK:** Define clocking of TO_CNT, TO_CNT1.
0 = TO_CNT clock selected by (TCS,TCS_USE_SAMPLE_EVT);
TO_CNT1 clocked on tdu_word_event
1 = TO_CNT1 uses same clock as TO_CNT

Bit 30:28 **TCS:** Timeout Clock selection
0b000 = *CMU_CLK0* selected
0b001 = *CMU_CLK1* selected
0b010 = *CMU_CLK2* selected
0b011 = *CMU_CLK3* selected
0b100 = *CMU_CLK4* selected
0b101 = *CMU_CLK5* selected
0b110 = *CMU_CLK6* selected
0b111 = *CMU_CLK7* selected

Bit 31 **Reserved:** Reserved
Note: Read as zero, should be written as zero

11.8.17 Register TIM[i]_CH[x]_TDUC

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								TO_CNT2								TO_CNT1								TO_CNT							
Mode	R								RPw								RPw								RPw							
Initial Value	0x00								0x00								0x00								0x00							

Bit 7:0 **TO_CNT:** Current Timeout value slice0 for channel x (x:0...m-1).
SLICING != 0b11: counter will be reset to 0X0 on TDU_RESYNC condition
SLICING = 0b11 : counter will be loaded with TOV2 on TDU_RESYNC condition

Bit 15:8 **TO_CNT1:** Current Timeout value slice1 for channel x (x:0...m-1). Counter will be reset to 0X0 on TDU_RESYNC condition

Bit 23:16 **TO_CNT2:** Current Timeout value slice2 for channel x (x:0...m-1). Counter will be reset to 0X0 on TDU_RESYNC condition

Bit 31:24 **Reserved:** Reserved

Note: Read as zero, should be written as zero

Note: The register **TIM[i]_CH[x]_TDUC** is writable if Timeout unit is disabled (TOCTRL=0b00).

Note: If USE_LUT != 0b00 (input signal generation by lookup table) the bit field TO_CNT2 is writable at any time, TO_CNT,TO_CNT1 will not be changed.

11.8.18 Register TIM[i]_CH[x]_ECNT

Address Offset:	**see Appendix B**																**Initial Value: 0x0000_0000**															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																ECNT															
Mode	R																R															
Initial Value	0x0000																0x0000															

Bit 15:0 **ECNT:** Edge counter

Note: If TIM channel is disabled the content of ECNT gets frozen. A read will auto clear the bits [15:1]. Further read accesses to ECNT will show on Bit 0 the actual input signal value of the channel.

Bit 31:16 **Reserved:** Reserved

Note: Read as zero, should be written as zero

11.8.19 Register TIM[i]_CH[x]_ECTRL

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_0000**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	USE_PREV_CH_I	ECLK_SEL	IMM_START	SWAP_CAPTURE	Reserved		EFLT_CTR_FE	EFLT_CTR_RE	USE_LUT		Reserved		TDU_RESYNC				Reserved	TDU_STOP			Reserved	TDU_START			TODET_IRQ_SRC		USE_PREV_TDU_	Reserved	EXT_CAP_SRC			
Mode	RW	RW	RW	RW	R		RW	RW	RW		R		RW				R	RW			R	RW			RW		RW	R	RW			
Initial Value	0	0	0	0	00		0	0	0		000		0x0				0	000			0x0	000			00		0	0x0	0000			

Bit 3:0 **EXT_CAP_SRC:** Defines selected source for triggering the EXT_CAPTURE functionality.
0b0000 = NEW_VAL_IRQ of following channel selected
0b0001 = AUX_IN selected
0b0010 = CNTOFL_IRQ of following channel selected
0b0011 and CICTRL = 1 : use signal TIM_IN(x) as input for channel x
0b0011 and CICTRL = 0 : use signal TIM_IN(x-1) as input for channel x (or TIM_IN(m-1) if x is 0)
0b0100 = ECNTOFL_IRQ of following channel selected
0b0101 =TODET_IRQ of following channel selected
0b0110 = GLITCHDET_IRQ of following channel selected
0b0111 = GPROFL_IRQ of following channel selected
0b1000 = cmu_clk selected by CLK_SEL of following channel
0b1001 = REDGE_DET of following channel selected
0b1010 = FEDGE_DET of following channel selected
0b1011 = logical or of(FEDGE_DET, REDGE_DET*)* of following channel selected
0b1100 = tdu_sample_evt of local TDU selected
0b1101 = tdu_word_evt of local TDU selected
0b1110 = tdu_frame_evt of local TDU selected
0b1111 = reserved

Note : Undefined values will not be written and AEI_STATUS will signal 0b10

Bit 4 **Reserved:** Reserved
Note: Read as zero, should be written as zero

Bit 5 **USE_PREV_TDU_IN:** Select input data source for TDU.
0 = use input data of local filter for TDU
1 = use input data of previous channel (after filter unit) for TDU

Bit 7:6 **TODET_IRQ_SRC:** selection of source for TODET_IRQ
0b00 = use tdu_timeout_evt
0b01 = use tdu_word_evt
0b10 = use tdu_frame_evt
0b11 = use tdu_sample_evt

Note: With TODET_IRQ_SRC=0b00 the ACB bit 2 will be driven by signal tdu_timeout_evt, if TODET_IRQ_SRC!=0b00 ACB2 will be 0.

Bit 10:8 **TDU_START:** Defines condition which will start the TDU unit.

0b000 = start once immediate on tdu_start_000_event (see Note)

0b001 = start once with occurrence of first cmu_clk selected by CLK_SEL when measure unit is enabled by TIM_EN=1

0b010 = start once with occurrence of first active edge selected by TOCTRL; restart on tdu_frame_evt if TDU is stopped

0b011 = start once with occurrence of first active edge selected by TOCTRL

0b100 = start/restart with occurrence of external capture event (if TDU is stopped, restart again)

0b101 = start/restart with occurrence of first cmu_clk selected by CLK_SEL when measure unit is enabled by TIM_EN=1 (if TDU is stopped, restart again)

0b110 = start once with occurrence of external capture event ; restart on tdu_frame_evt if TDU is stopped

0b111 = start/restart with occurrence of first active edge selected by TOCTRL (if TDU is stopped, restart again)

Note: tdu_start_000_event is defined as: Each writing of TOCTRL != 0 (independent of current TOCTRL) while TDU_START=0b000 and TDU is stopped (initially or stopped by TDU_STOP event). This event will last 1 system clock cycle.

Note: In mode SLICING=0b11 every start/restart will load the TO_CNT with value TOV2.

Bit 11 **Reserved:** Reserved

Note: Read as zero, should be written as zero

Bit 14:12 **TDU_STOP:** Defines condition which will stop the TDU unit.

0b000 = immediate stop counting of TDU on tdu_toctrl_0_event (see Note)

0b001 = stop counting of TDU on tdu_word_evt or on tdu_toctrl_0_event (see Note)

0b010 = stop counting of TDU on tdu_frame_evt or on tdu_toctrl_0_event (see Note)

0b011 = stop counting of TDU on tdu_timeout_evt or on tdu_toctrl_0_event (see Note)

0b100 = stop counting of TDU on external capture event or on tdu_toctrl_0_event (see Note)

0b101 = if SLICING =0b10|0b11 then
stop counting of TO_CNT on tdu_word_evt or on tdu_toctrl_0_event (see Note);
stop counting of TO_CNT1 on tdu_frame_evt or on tdu_toctrl_0_event (see Note);
stop counting of TO_CNT2 on tdu_toctrl_0_event (see Note);
else reserved, no action performed

0b11- = reserved, no action performed

Note: tdu_toctrl_0_event is defined as: Each writing of TOCTRL = 0 (independent of current TOCTRL) while TDU is started. This event will last 1 system clock cycle.

Bit 15 **Reserved:** Reserved
Note: Read as zero, should be written as zero

Bit 19:16 **TDU_RESYNC:** Defines condition which will resynchronize the TDU unit.
Behavior with SLICING != 0b11:

0b0000 : reset counter TO_CNT2 on each active edge selected by TOCTRL or tdu_timeout_evt or on tdu_start_000_event (see Note);
reset counters TO_CNT,TO_CNT1 on tdu_timeout_evt or on tdu_start_000_event (see Note);
if SLICING=0b10 and TO_CTRL=0b-1 then reset TO_CNT on rising input edge;
if SLICING=0b10 and TO_CTRL=0b1- then reset TO_CNT1 on falling input edge;
if SLICING!=0b10 then reset counters TO_CNT,TO_CNT1 on each active edge selected by TOCTRL;

0b0--1 : if SLICING=0b10 and TO_CTRL=0bx1 then reset TO_CNT on rising input edge;
if SLICING=0b10 and TO_CTRL=0b1- then reset TO_CNT1 on falling input edge;
if SLICING!=0b10 then reset counters TO_CNT,TO_CNT1 on each active edge selected by TOCTRL;
if SLICING=0b00 then reset TO_CNT2 on each active edge selected by TOCTRL ;

0b0x1- : reset counters TO_CNT on tdu_word_evt;

0b01-- : reset counter TO_CNT1 on tdu_frame_evt;
if SLICING=0b01 then reset TO_CNT on tdu_frame_evt;

0b1000 : reset counters TO_CNT,TO_CNT1,TO_CNT2 on event selected by EXT_CAP_SRC;

0b1--- : if SLICING!=0b00 then reset counter TO_CNT2 on tdu_sample_evt;

0b1--1 : reset counter TO_CNT2 on each active edge selected by TOCTRL;
if SLICING=0b10 and TO_CTRL=0b-1 then reset TO_CNT on rising input edge;
if SLICING=0b10 and TO_CTRL=0b1- then reset TO_CNT1 on falling input edge;
if SLICING!=0b10 then reset counters TO_CNT,TO_CNT1 on each active edge selected by TOCTRL;

0b1-1- : reset counters TO_CNT on tdu_word_evt;

0b11-- : reset counter TO_CNT1 on tdu_frame_evt;
if SLICING=0b01 then reset TO_CNT on tdu_frame_evt;

Behavior with SLICING = 0b11:

0b0000 : load counter TO_CNT with TOV2 on each active edge selected by TOCTRL or tdu_timeout_evt or on tdu_start_000_event (see Note);
reset counter TO_CNT1 on each active edge selected by TOCTRL or tdu_timeout_evt or on tdu_start_000_event (see Note)

0b---1 : load counter TO_CNT with TOV2 on each active edge selected by TOCTRL;
reset counter TO_CNT1 on each active edge selected by TOCTRL

0b0-1- : load counter TO_CNT with TOV2 on tdu_word_evt

0b1-1- : reset counter TO_CNT on tdu_word_evt

0b-1-- : reset counter TO_CNT1 on tdu_frame_evt

0b1000 = load counter TO_CNT with TOV2; reset counter TO_CNT1 on event selected by EXT_CAP_SRC

Note: tdu_start_000_event is defined as: Each writing of TOCTRL != 0 (independent of current TOCTRL) while TDU_START=0b000 and TDU is stopped (initially or stopped by TDU_STOP event). This event will last 1 system clock cycle.

Bit 21:20 **Reserved:** Reserved
Note: Read as zero, should be written as zero

Bit 23:22 **USE_LUT:** generate Filter input by lookup table
0b00 = lookup table not in use, lut_in0(x) used as filter input
0b01 = use 3 bit lookup table with index = ext_capture(x) & lut_in1(x) & lut_in0(x) . Filter input is defined by TO_CNT2[index].
0b10 = use 3 bit lookup table with index = fout_prev(x) & lut_in1(x) & lut_in0(x) . Filter input is defined by TO_CNT2[index].
0b11 = use 3 bit lookup table with index = tssm_out(x) & lut_in1(x) & lut_in0(x) . Filter input is defined by TO_CNT2[index].

Bit 24 **EFLT_CTR_RE:** Extension of bit field FLT_CTR_RE.
Details described in FLT_CTR_RE bit field of register TIM[i]_CH[x]_CTRL.

Bit 25 **EFLT_CTR_FE:** Extension of bit field FLT_CTR_FE.
Details described in FLT_CTR_FE bit field of register TIM[i]_CH[x]_CTRL.

Bit 27:26 **Reserved:** Reserved
Note: Read as zero, should be written as zero

Bit 28 **SWAP_CAPTURE:** swap point of time of capturing CNTS and GPR1
0 = inactive edge will capture data in CNTS; NEWVAL_IRQ event will capture data in GPR1
1 = swap time of capture: inactive edge will capture data in GPR1; NEWVAL_IRQ event will capture data in CNTS

This bit is only applicable in TPWM and TPIM mode. Set to 0 in all other modes.

Bit 29 **IMM_START:** Start immediately the measurement
0 = start with first active edge the measurement
1 = start immediately after enable (TIM_EN=1) the measurement
This bit is only applicable in TPWM and TPIM mode. Set to 0 in all other modes.

Bit 30 **ECLK_SEL:** Extension of bit field CLK_SEL.
Details described in CLK_SEL bit field of register TIM[i]_CH[x]_CTRL.

Bit 31 **USE_PREV_CH_IN:** Select input data source for TIM channel.
0 = use input data of local filter unit for channel measurements
1 = use input data of previous channel (after filter unit) for channel measurements

11.8.20 Register TIM[i]_INP_VAL

Address Offset:	**see Appendix B**																**Initial Value: 0x0000_0000**															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								TIM_IN								F_IN								F_OUT							
Mode	R								R								R								R							
Initial Value	0x0000								0x0000								0x0000								0x0000							

Bit 7:0 **F_OUT:** signals after TIM FLT unit
Bit 15:8 **F_IN:** signals after INPSRC selection, before TIM FLT unit
Bit 23:16 **TIM_IN:** signals after TIM input signal synchronization
Bit 31:24 **Reserved:** Reserved
Note: Read as zero, should be written as zero

12 Timer Output Module (TOM)

12.1 Overview

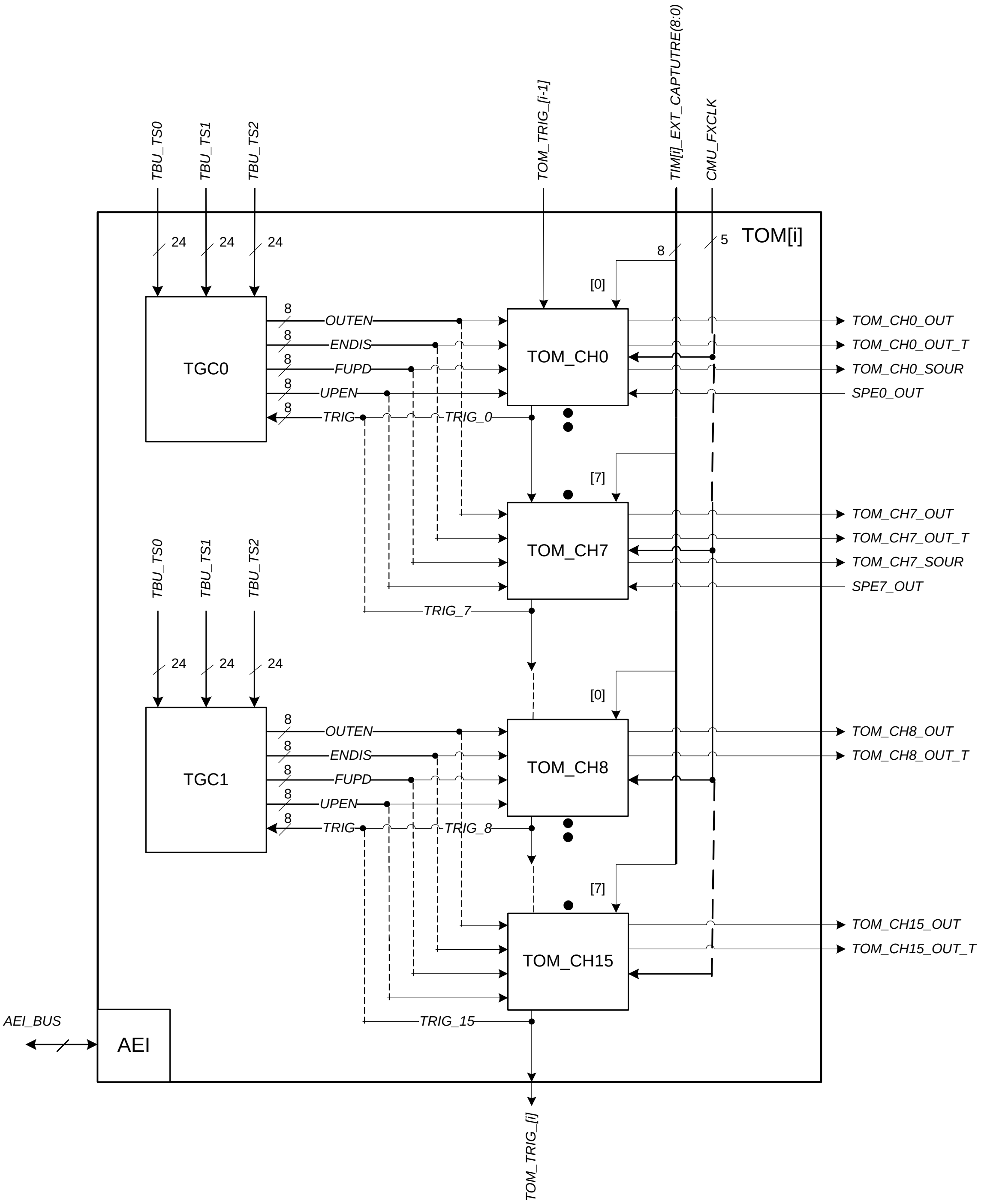
The Timer Output Module (TOM) offers up to 16 independent channels (index x) to generate simple PWM signals at each output pin *TOM[i]_CH[x]_OUT*.

Additionally, at TOM output *TOM[i]_CH15_OUT* a pulse count modulated signal can be generated.
The architecture of the TOM sub-module is depicted in figure 12.1.1.
Indices and their range as used inside this chapter are:
y=0,1
z=0..7

The following design variables are used inside this chapter. Please refer to device specific Appendix B for correct value.

cCTO : TOM channel count; number of channels per instance - 1.

12.1.1 TOM block diagram



The two sub-modules TGC0 and TGC1 are global channel control units that control the enabling/disabling of the channels and their outputs as well as the update of their period and duty cycle register.

The module TOM receives two (three) timestamp values *TBU_TS0*, *TBU_TS1* (and *TBU_TS2*) in order to realize synchronized output behavior on behalf of a common time base.

The 5 dedicated clock line inputs *CMU_FXCLK* are providing divided clocks that can be selected to clock the output pins.

The trigger signal *TOM_TRIG_[i-1]* of TOM instance i comes from the preceding instance i-1, the trigger *TOM_TRIG_[i]* is routed to succeeding instance i+1.
Note, TOM0 is connected to its own output *TOM_TRIG_0*, i.e. the last channel of TOM instance 0 can trigger the first channel of TOM instance 0 (this path is registered, which means delayed by one SYS_CLK period).

12.2 TOM Global Channel Control (TGC0, TGC1)

12.2.1 Overview

There exist two global channel control units (TGC0 and TGC1) to drive a number of individual TOM channels synchronously by external or internal events.

Each TGC[y] can drive up to eight TOM channels where TGC0 controls TOM channels 0 to 7 and TGC1 controls TOM channels 8 to 15.

The TOM sub-module supports four different kinds of signaling mechanisms:

- Global enable/disable mechanism for each TOM channel with control register **TOM[i]_TGC[y]_ENDIS_CTRL** and status register **TOM[i]_TGC[y]_ENDIS_STAT**

- Global output enable mechanism for each TOM channel with control register **TOM[i]_TGC[y]_OUTEN_CTRL** and status register **TOM[i]_TGC[y]_OUTEN_STAT**

- Global force update mechanism for each TOM channel with control register **TOM[i]_TGC[y]_FUPD_CTRL**
- Update enable of the register **CM0**, **CM1** and **CLK_SRC** for each TOM channel with the control bit field **UPEN_CTRL[z]** of **TOM[i]_TGC[y]_GLB_CTRL**

12.2.2 TGC Sub-unit

Each of the first three individual mechanisms (enable/disable of the channel, output enable and force update) can be driven by three different trigger sources.
The three trigger sources are :

- the host CPU (bit **HOST_TRIG** of register **TOM[i]_TGC[y]_GLB_CTRL**)
- the TBU time stamp (signal *TBU_TS0, TBU_TS1, TBU_TS2* if available)
- the internal trigger signal *TRIG* (bunch of trigger signals *TRIG_[x]*)
  which can be either the trigger *TRIG_CCU0* of channel x,
  the trigger of preceding channel x-1 (i.e. signal *TRIG_[x-1]*) or
  the external trigger *TIM_EXT_CAPTURE(t)* of assigned TIM channel t.

The first way is to trigger the control mechanism by a direct register write access via host CPU (bit **HOST_TRIG** of register **TOM[i]_TGC[y]_GLB_CTRL**).

The second way is provided by a compare match trigger on behalf of a specified time base coming from the module TBU (selected by bits **TBU_SEL**) and the time stamp compare value defined in the bit field **ACT_TB** of register **TOM[i]_TGC[y]_ACT_TB**. Note, a signed compare of **ACT_TB** and selected *TBU_TS*[*x*] with x=0,1,2 is performed.

The third possibility is the input *TRIG* (bunch of trigger signals *TRIG_[x]*) coming from the TOM channels 0 to 7 / 8 to 15.

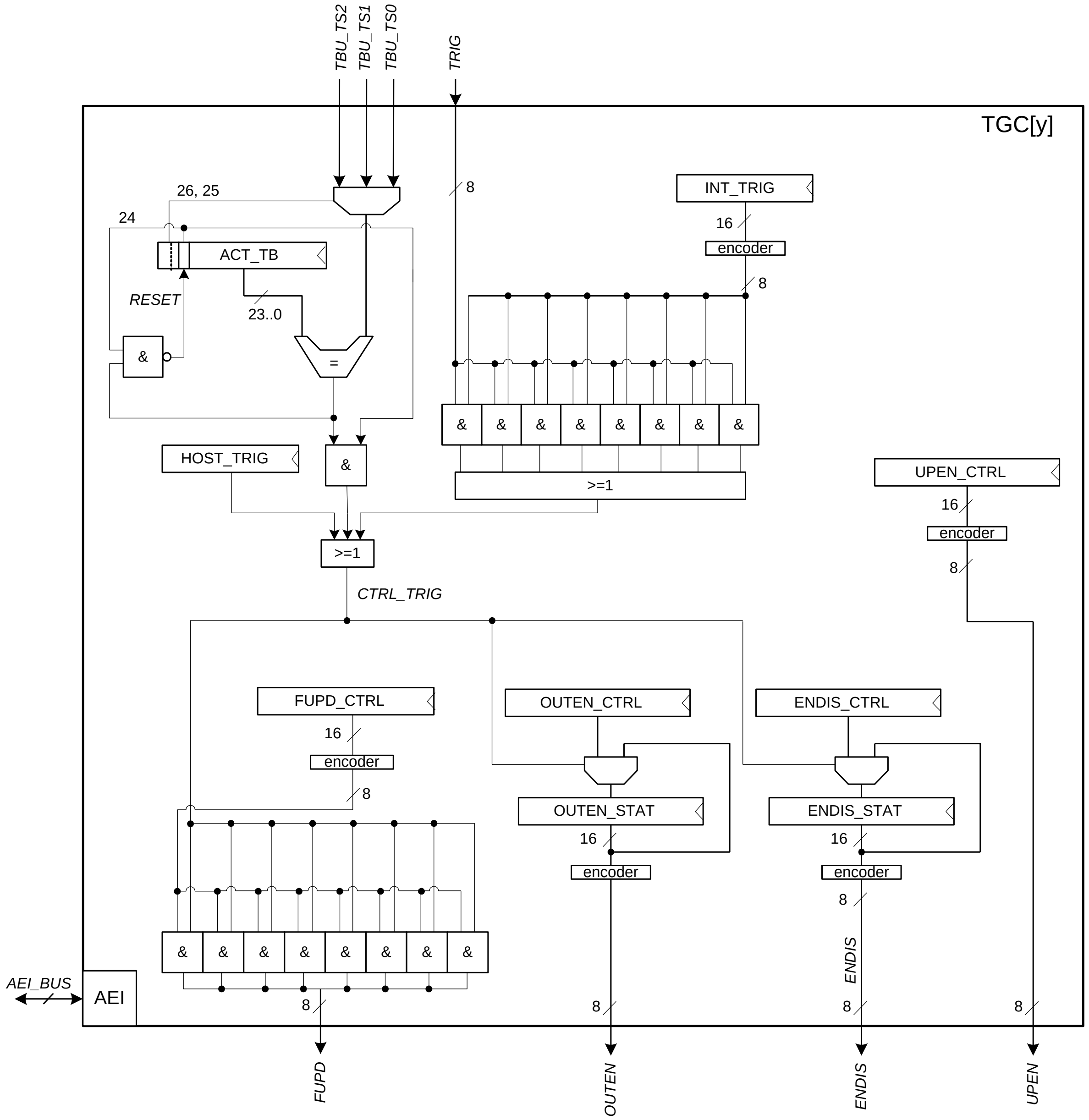
The corresponding trigger signal *TRIG_[x]* coming from channel [x] can be masked by the register **TOM[i]_TGC[y]_INT_TRIG**.

To enable or disable each individual TOM channel, the register **TOM[i]_TGC[y]_ENDIS_CTRL** and/or **TOM[i]_TGC[y]_ENDIS_STAT** have to be used.

The register **TOM[i]_TGC[y]_ENDIS_STAT** controls directly the signal *ENDIS*. A write access to this register is possible.

The register **TOM[i]_TGC[y]_ENDIS_CTRL** is a shadow register that overwrites the value of register **TOM[i]_TGC[y]_ENDIS_STAT** if one of the three trigger conditions matches.

12.2.2.1 TOM Global channel control mechanism



The output of the individual TOM channels can be controlled using the register **TOM[i]_TGC[y]_OUTEN_CTRL** and **TOM[i]_TGC[y]_OUTEN_STAT**.

The register **TOM[i]_TGC[y]_OUTEN_STAT** controls directly the signal *OUTEN*. A write access to this register is possible.

The register **TOM[i]_TGC[y]_OUTEN_CTRL** is a shadow register that overwrites the value of register **TOM[i]_TGC[y]_OUTEN_STAT** if one of the three trigger conditions matches.

If a TOM channel is disabled by the register **TOM[i]_TGC[y]_OUTEN_STAT**, the actual value of the channel output at *TOM_CH[x]_OUT* is defined by the signal level bit (**SL**) defined in the channel control register **TOM[i]_CH[x]_CTRL**.

If the output is enabled, the output at *TOM_CH[x]_OUT* depends on value of flip-flop **SOUR**.

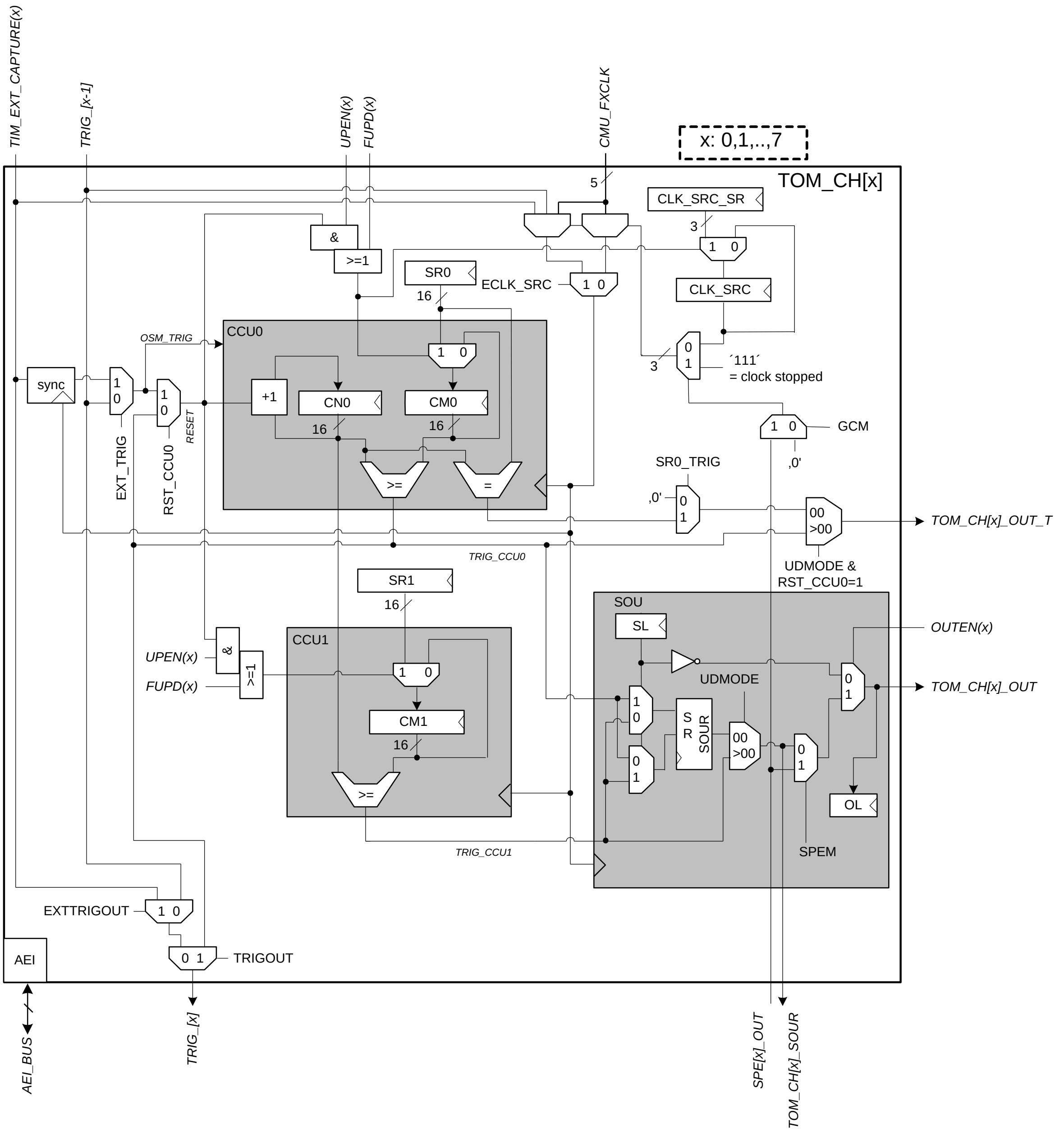
The register **TOM[i]_TGC[y]_FUPD_CTRL** defines which of the TOM channels receive a *FORCE_UPDATE* event if the trigger signal *CTRL_TRIG* is raised.
Note: The force update request is stored and executed synchronized to the selected CMU_CLK.

The register bits **UPEN_CTRL[z]** defines for which TOM channel the update of the working register **CM0**, **CM1** and **CLK_SRC** by the corresponding shadow register **SR0**, **SR1** and **CLK_SRC_SR** is enabled. If update is enabled, the register **CM0**, **CM1** and **CLK_SRC** will be updated on reset of counter register **CN0** (see figures 12.3.1 and 12.3.2). An exception is the configuration of SR0_TRIG=1 which enable the trigger generation defined by **SR0**. Then **CM0** is not updated with **SR0**.

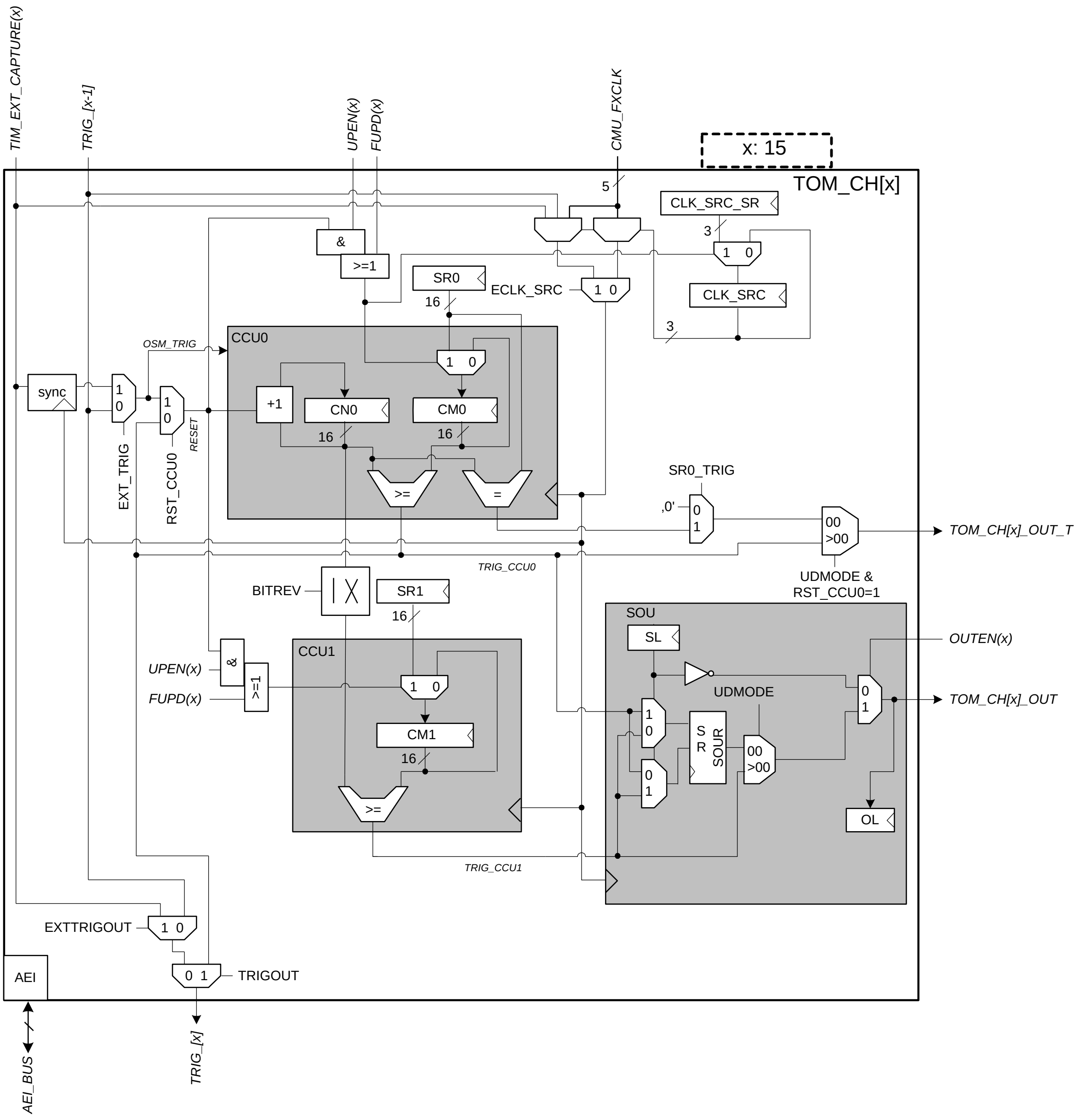
12.3 TOM Channel

Each individual TOM channel comprises a Counter Compare Unit 0 (CCU0) which contains the counter register **CN0** and the period register **CM0**, a Counter Compare Unit 1 (CCU1) which contains the duty cycle register **CM1** and the Signal Output Generation Unit (SOU) which contains the output register **SOUR**. The architecture is depicted in figure 12.3.1 for channels 0 to 7 and in 12.3.2 for channels 8 to 15.

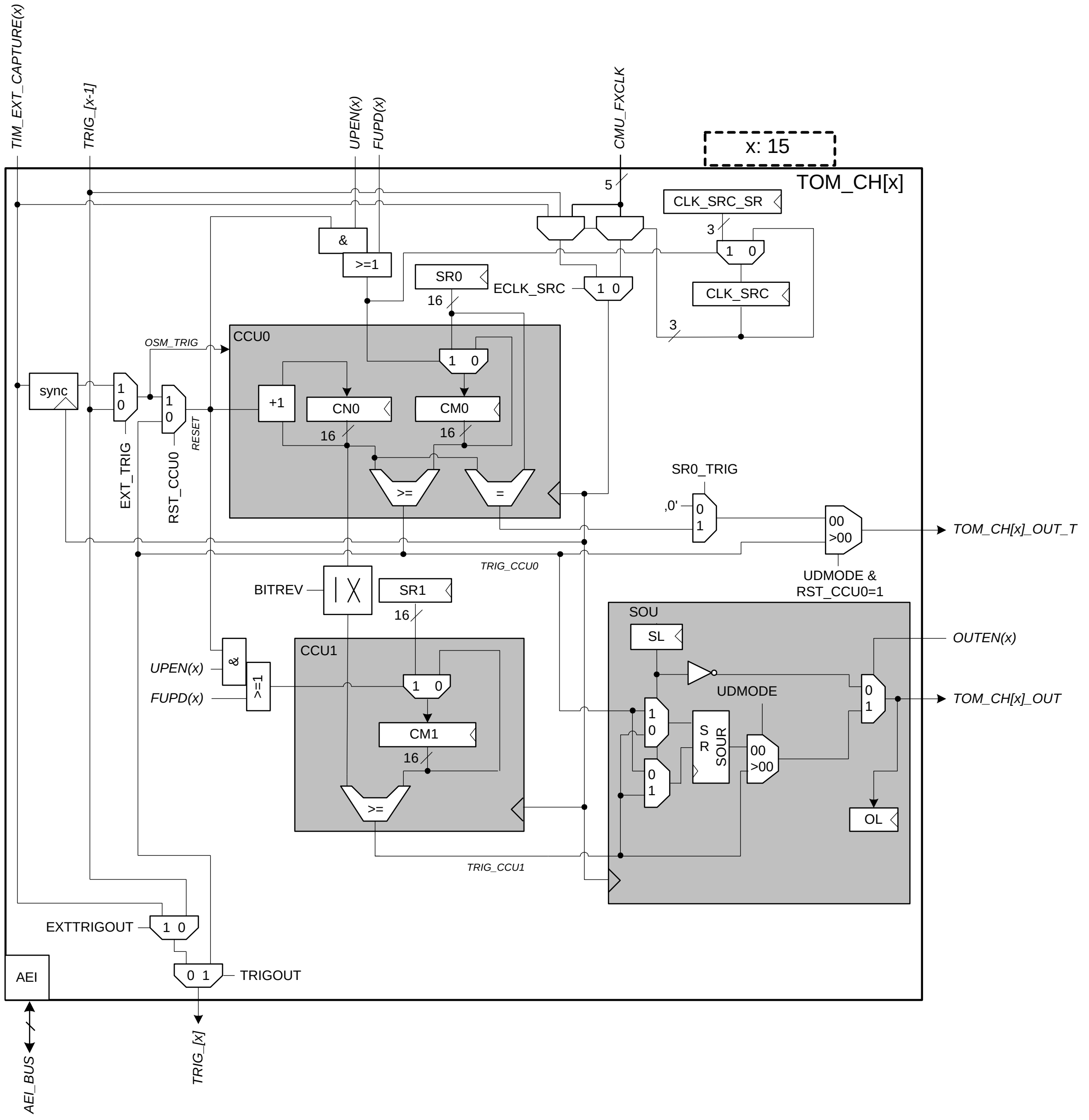
12.3.1 TOM Channel 0..7 architecture



12.3.2 TOM Channel 8..14 architecture



12.3.3 TOM Channel 15 architecture



The CCU0 contains a counter **CN0** which is clocked with one of the selected input frequencies (*CMU_FXCLK*) provided from outside of the sub-module.

Depending on configuration bits RST_CCU0 of register **TOM[i]_CH[x]_CTRL** the counter register **CN0** can be reset either when the counter value is equal to the compare value **CM0** (i.e. **CN0** counts only from 0 to **CM0**-1 and is then reset to 0) or when signaled by the TOM[i] trigger signal *TRIG_[x-1]* of the preceding channel [x-1] (which can also be the last channel of preceding instance TOM[i-1]) or the trigger signal *TIM_EXT_CAPTURE(x)* of the assigned TIM channel [x].

Note: As an exception, the input *TRIG_[0]* of instance TOM0 is triggered by its own last channel cCTO via signal *TRIG_[cCTO]*. Please refer to device specific Appendix B for value cCTO of TOM0.

When the counter register **CN0** is greater or equal than the value **CM0** (in fact **CM0**-1)**,** the sub-unit CCU0 triggers the SOU sub-unit and the succeeding TOM sub-module channel (signal *TRIG_CCU0*).

In the sub-unit CCU1 the counter register **CN0** is compared with the value of register **CM1**. If **CN0** is greater or equal than **CM1** the sub-unit CCU1 triggers the SOU sub-unit (signal *TRIG_CCU1*).

If counter register **CN0** of channel x is reset by its own CCU0 unit (i.e. the compare match of **CN0**>=**CM0**-1 configured by RST_CCU0=0), following statements are valid:
- **CN0** counts from 0 to **CM0**-1 and is then reset to 0.
- When **CN0** is reset from **CM0** to 0, an edge to SL is generated.
- When **CN0** is incrementing and reaches **CN0** > **CM1**, an edge to !SL is generated.
- if **CM0**=0 or **CM0**=1, the counter **CN0** is constant 0.
- if **CM1**=0, the output is !SL = 0% duty cycle
- if **CM1** >= **CM0** and **CM0**>1, the output is SL = 100% duty cycle

If the counter register **CN0** of channel x is reset by the trigger signal coming from another channel or the assigned TIM module (configured by RST_CCU0=1), following statements are valid:
- **CN0** counts from 0 to MAX-1 and is then reset to 0 by trigger signal
- **CM0** defines the edge to SL value, **CM1** defines the edge to !SL value.
- if **CM0**=**CM1**, the output switches to SL if **CN0**=**CM0**=**CM1** (**CM0** has higher priority)
- if **CM0**=0 and **CM1**=MAX, the output is SL = 100% duty cycle
- if **CM0** > MAX, the output is !SL = 0% duty cycle, independent of **CM1**.

The hardware ensures that for both 0% and 100% duty cycle no glitch occurs at the output of the TOM channel.
The SOU sub-unit is responsible for output signal generation. On a trigger *TRIG_CCU0* from sub-unit CCU0 or *TRIG_CCU1* from sub-unit CCU1 an SR flip-flop of sub-unit SOU is either set or reset. If it is set or reset depends on the configuration bit **SL** of the control register **TOM[i]_CH[x]_CTRL**. The initial signal output level for the channel is the reverse value of the bit **SL**.

Figure 12.3.5.1 clarifies the PWM output behavior with respect to the **SL** bit definition. The output level on the TOM channel output pin *TOM[i]_CH[x]_OUT* is captured in bit **OL** of register **TOM[i]_CH[x]_STAT**.

12.3.4 Duty cycle, Period and Clock Frequency Update Mechanisms

The two action register **CM0** and **CM1** can be reloaded with the content of the shadow register **SR0** and **SR1**. The register **CLK_SRC** that determines the clock frequency of the counter register **CN0** can be reloaded with its shadow register **CLK_SRC_SR** (bit field in register **TOM[i]_CH[x]_CTRL**)

The update of the register **CM0**, **CM1** and **CLK_SRC** with the content of its shadow register is done when the reset of the counter register **CN0** is requested (via signal *RESET*). This reset of **CN0** is done if the comparison of **CN0** greater or equal than **CM0** is true or when the reset is triggered by another TOM channel [x-1] via the signal *TRIG_[x-1]* or when signaled via the signal *TIM_EXT_CAPTURE(x)* of the assigned TIM channel [x].

With the update of the register **CLK_SRC** at the end of a period a new counter **CN0** clock frequency can easily be adjusted.

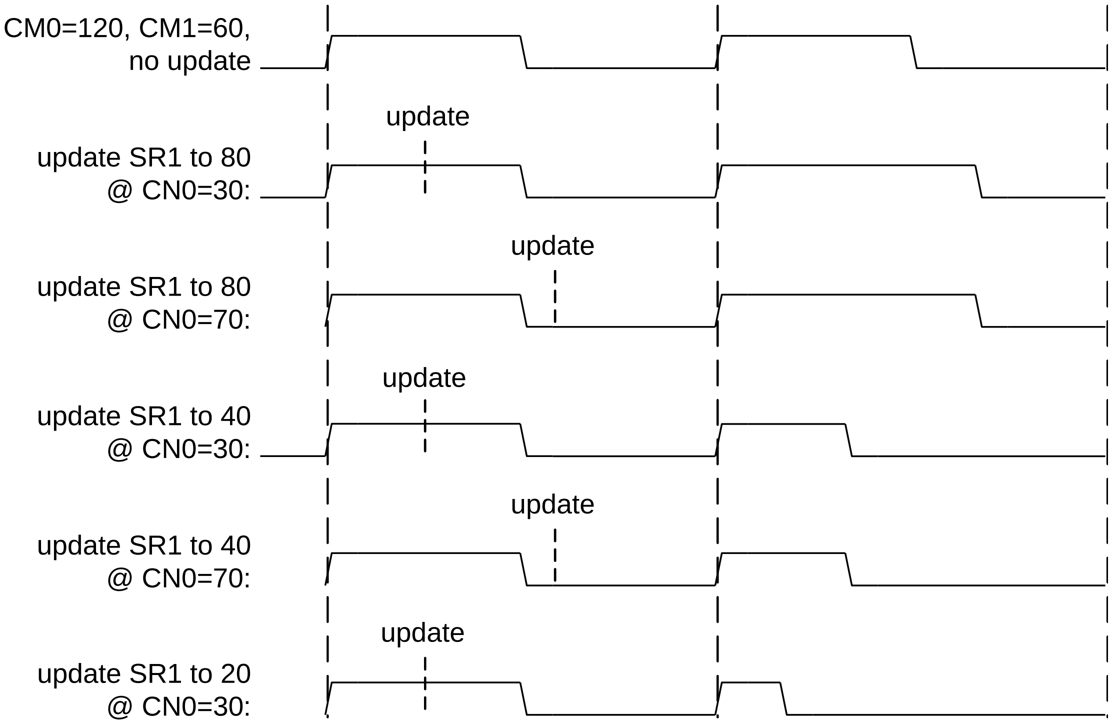
In case of RST_CCU0=1 and update enabled by **UPEN_CTRL[z]** the register **CM0**, **CM1** and **CLK_SRC** will be updated when **CN0** is reset.

An update of duty cycle, period and counter **CN0** clock frequency becoming effective synchronously with start of a new period can easily be reached by performing following steps:
1. disable the update of the action register with the content of the corresponding shadow register by setting the channel specific configuration bit **UPEN_CTRL[z]** of register **TOM[i]_TGC[y]_GLB_CTRL** to '0'.
2. write new desired values to **SR0**, **SR1** , **CLK_SRC_SR**
3. enable update of the action register by setting the channel specific configuration bit **UPEN_CTRL[z]** of register **TOM[i]_TGC[y]_GLB_CTRL** to '1'.

*12.3.4.1 **Synchronous Update Of Duty Cycle Only***

A synchronous update of only the duty cycle can be done by simply writing the desired new value to register **SR1** without preceding disable of the update mechanism (as described in the chapter above). The new duty cycle is then applied in the period following the period where the update of register **SR1** was done.

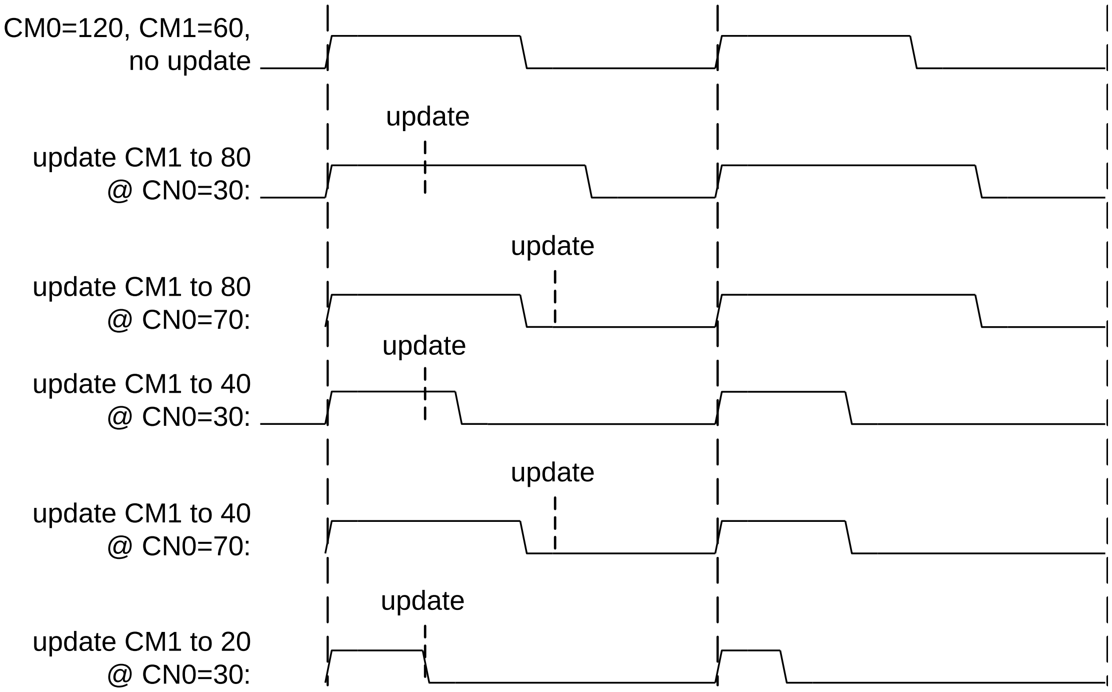
12.3.4.1.1 Synchronous update of duty cycle



12.3.4.2 Asynchronous Update Of Duty Cycle Only

If the update of the duty cycle should be performed independent of the start of a new period (asynchronous), the desired new value can be written directly to register **CM1**. In this case it is recommended to additionally either disable the synchronous update mechanism as a whole (i.e. clearing bits **UPEN_CTRL[z]** of corresponding channel [x] in register **TOM[i]_TGX[y]_GLB_CTRL**) or updating **SR1** with the same value as **CM1** before writing to **CM1**.
Depending on the point of time of the update of **CM1** in relation to the actual value of **CN0** and **CM1**, the new duty cycle is applied in the current period or the following period (see Figure 12.3.4.2.1). In any case the creation of glitches are avoided. The new duty cycle may jitter from update to update by a maximum of one period (given by **CM0**). However, the period remains unchanged.

12.3.4.2.1 Asynchronous update of duty cycle



12.3.5 Continuous Counting Up Mode

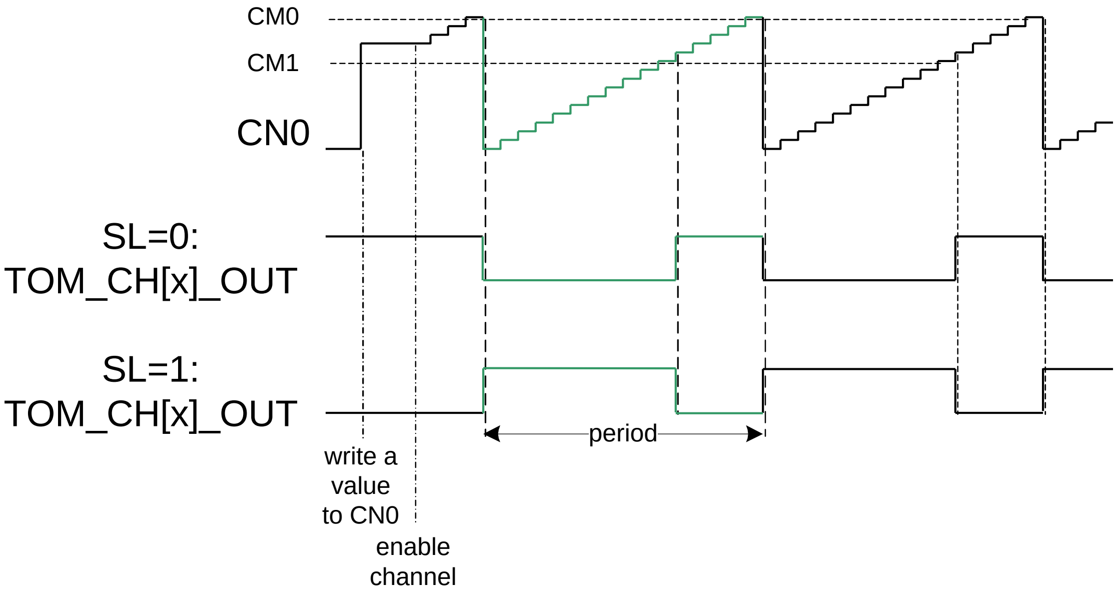
In continuous mode the TOM channel starts incrementing the counter register **CN0** once it is enabled by setting the corresponding bits in register **TOM[i]_TGC[y]_ENDIS_STAT** (refer to chapter 12.2.2 for details of enabling a TOM channel).

The signal level of the generated output signal can be configured with the configuration bit **SL** of the channel configuration register **TOM[i]_CH[x]_CTRL**.

If the counter **CN0** is reset from **CM0** back to zero, the first edge of a period is generated at *TOM[i]_CH[x]_OUT*.

The second edge of the period is generated if **CN0** has reached **CM1**.
Every time the counter **CN0** has reached the value of **CM0** it is reset back to zero and proceeds with incrementing.

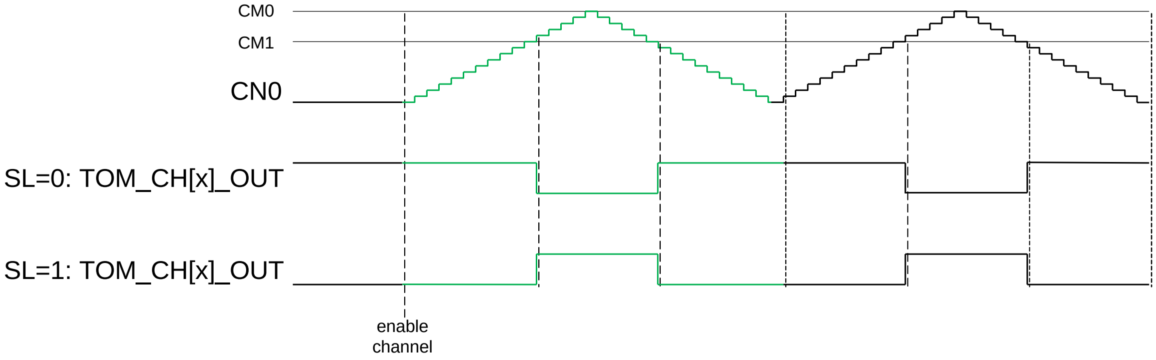
12.3.5.1 PWM Output with respect to configuration bit ***SL*** *in continuous mode*



12.3.6 Continuous Counting Up-Down Mode

In continuous mode, if **CN0** counts up and down (UDMODE != 0b00), depending on configuration bits RST_CCU0 of register **TOM[i]_CH[x]_CTRL** the counter register **CN0** changes direction either when the counter value is equal to the compare value **CM0,** has counted down to 0 or when triggered by the TOM[i] trigger signal *TRIG_[x-1]* of the preceding channel [x-1] (which can also be the last channel of preceding instance TOM[i-1]) or the trigger signal *TIM_EXT_CAPTURE(x)* of the assigned TIM channel [x].
In this case, if UPEN_CTRL[x]=1, also the working register **CM0**, **CM1** and **CLK_SRC** are updated depending on UDMODE.

12.3.6.1 PWM Output behavior with respect to the SL bit in the ATOM[i]_CH[x]_CTRL register if UDMODE != 0b00



The clock of the counter register **CN0** can be one of the CMU clocks CMU_CLKx. The clock for **CN0** is defined by CLK_SRC_SR value in register **TOM[i]_CH[x]_CTRL.** The duration of a period in multiples of selected **CN0** counter clock ticks is defined by the **CM0** configuration value (i.e. **CM0** defines half of period in up-down mode). **CM1** defines the duty cycle value in clock ticks of selected **CN0** counter clock (i.e. **CM0** defines half of duty cycle in up-down mode).

If counter register **CN0** of channel x is reset by its own CCU0 unit (i.e. the compare match of **CN0**>=**CM0** configured by RST_CCU0=0), following statements are valid:
- **CN0** counts continuously first up from 0 to **CM0**-1 and then down to 0.
- if **CN0** >= **CM1**, the output is set to SL
- if **CM1**=0, the output is SL (i.e. 100% duty cycle)
- if **CM1**>= **CM0**, the output is !SL (i.e. 0% duty cycle)
- On output *TOM[i]_CHx]_OUT* a PWM signal is generated. The period is defined by **CM0**, the duty cycle is defined by **CM1**.

This behavior is depicted in figure 12.3.6.1.

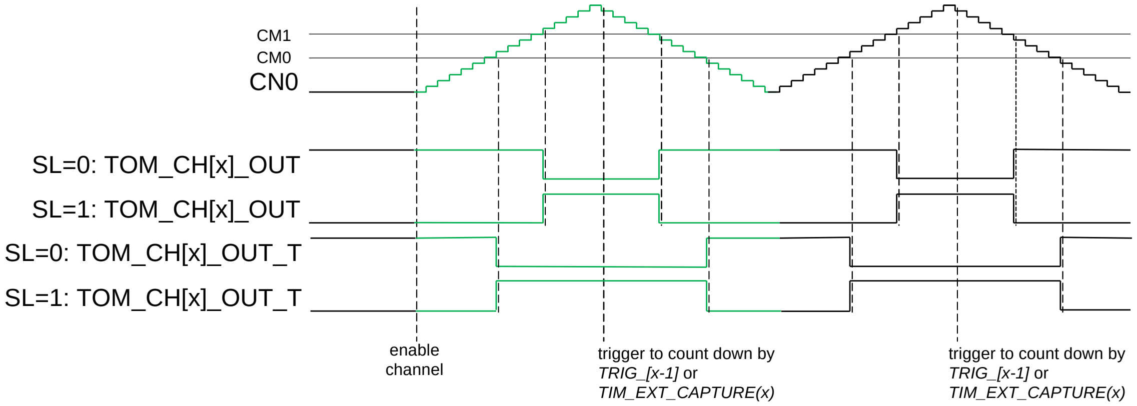
If the counter register **CN0** of channel x is reset by the trigger signal coming from another channel or the assigned TIM module (configured by RST_CCU0=1), following statements are valid:
- **CN0** counts continuously first up. On a trigger signal the counter switches to count down mode. If **CN0** has reached 0, it switches to count up mode.
- if **CN0** >= **CM1**, the output is set to SL
- if **CM1**=0, the output is SL (i.e. 100% duty cycle)
- if **CM1**>= **CM0**, the output is !SL (i.e. 0% duty cycle)
- On output *TOM[i]_CHx]_OUT* a PWM signal is generated. The period is defined by the CCU0 trigger of triggering channel, the duty cycle is defined by **CM1**.
- On output *TOM[i]_CHx]_OUT_T* a PWM signal is generated. The period is defined by the CCU0 trigger of triggering channel, the duty cycle is defined by **CM0**.

This behavior is depicted in figure 12.3.6.2.

Note that in case of up-down counter mode and RST_CCU0=1 it is recommended that
- the triggering channel and the triggered channel are both running in up-down mode
- the time between two trigger signals is equal to the time needed for CN0 of triggered channel to count back to 0 and again up to the same upper value.

The second recommendation can be reached by synchronizing the start of triggering channel and of triggered channel, i.e. let both channel start with a CN0 value 0.
Note that if there is a synchronization register in the trigger chain (indicated by value TOM_TRIG_CHAIN in register **CCM[i]_HW_CONF**), the additional delay of the trigger by one clock period has to be taken into account by starting at triggering channel with a CN0 vaue 1 (+1 compared to CN0 of triggered channel).

12.3.6.2 PWM Output behavior in case of RST_CCU0=1 and UDMODE != 0b00


CM1
CM0
CN0
SL=0: TOM_CH[x]_OUT
SL=1: TOM_CH[x]_OUT
SL=0: TOM_CH[x]_OUT_T
SL=1: TOM_CH[x]_OUT_T
enable channel
trigger to count down by *TRIG_[x-1]* or *TIM_EXT_CAPTURE(x)*
trigger to count down by *TRIG_[x-1]* or *TIM_EXT_CAPTURE(x)*

12.3.7 One-shot Counting Up Mode

In one-shot mode, the TOM channel generates one pulse with a signal level specified by the configuration bit **SL** in the channel [x] configuration register **TOM[i]_CH[x]_CTRL**.

First the channel has to be enabled by setting the corresponding **TOM[i]_TGC[y]_ENDIS_STAT** value and the one-shot mode has to be enabled by setting bit **OSM** in register **TOM[i]_CH[x]_CTRL**.

In one-shot mode the counter **CN0** will not be incremented once the channel is enabled.
A write access to the register **CN0** triggers the start of pulse generation (i.e. the increment of the counter register **CN0**).

If SPE mode of TOM[i] channel 2 is enabled (set bit **SPEM** of register **TOM[i]_CH2_CTRL**), also the trigger signal *SPE[i]_NIPD* can trigger the reset of register **CN0** to zero and a start of the pulse generation.

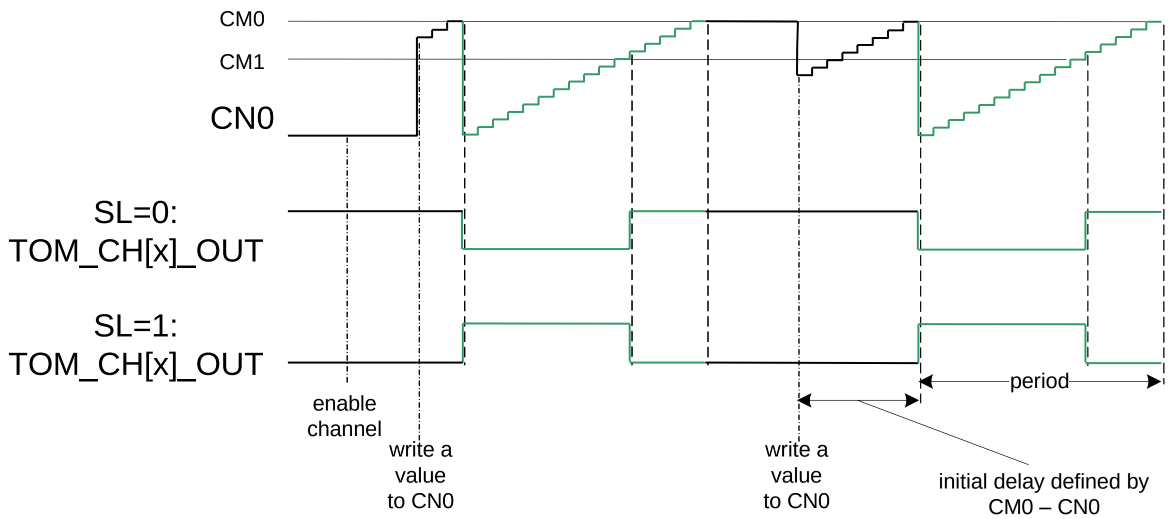
The new value of **CN0** determines the start delay of the first edge. The delay time of the first edge is given by (**CM0-CN0**) multiplied with period defined by current value of **CLK_SRC**.

If the counter **CN0** is reset from **CM0** back to zero, the first edge at *TOM[i]_CH[x]_OUT* is generated.
To avoid an update of **CMx** register with content of **SRx** register at this point in time, the automatic update should be disabled by setting UPEN_CTRL[x] = 00 (in register **TOM[i]_TGC[y]_GLB_CTRL**)

The second edge is generated if **CN0** is greater or equal than **CM1** (i.e. **CN0** was incremented until it has reached **CM1** or **CN0** is greater than **CM1** after an update of **CM1**).

If the counter **CN0** has reached the value of **CM0** a second time, the counter stops.

12.3.7.1 PWM Output with respect to configuration bit SL in one-shot mode: trigger by writing to CN0



Further output of single periods can be started by a write access to register CN0.
If CN0 is already incrementing (i.e. started by writing to CN0 a value CN0start < CM0), the affect of a second write access to CN0 depends on the phase of CN0:
phase 1: update of CN0 before CN0 reaches first time CM0
phase 2: update of CN0 after CN0 has reached first time CM0 but is less than CM1
phase 3: update of CN0 after CN0 has reached first time CM0 and CN0 is greater than or equal CM1

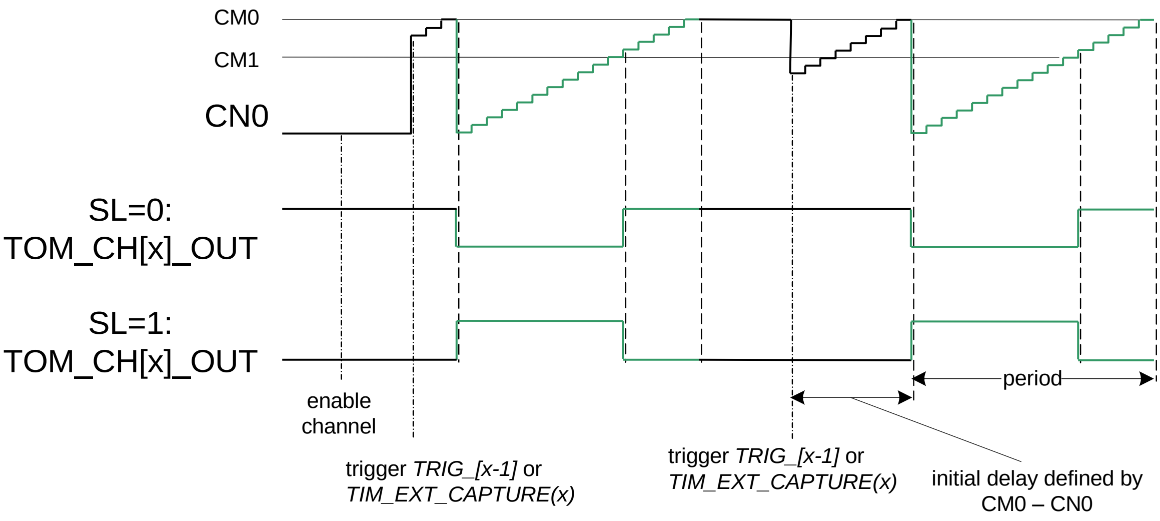
In phase 1: writing to counter CN0 a value CN0new < CM0 leads to a shift of first edge (generated if CN0 reaches CM0 first time) by the time CM0-CN0new.

In phase 2: writing to incrementing counter CN0 a value CN0new < CM1 while CN0old is below CM1 leads to a lengthening of the pulse. The counter CN0 stops if it reaches CM0.

In phase 3: Writing to incrementing counter CN0 a value CN0new while CN0old is already greater than or equal CM1 leads to an immediate restart of a single pulse generation inclusive the initial delay defined by CM0 - CN0new.

If a channel is configured to one-shot mode and configuration bit OSM_TRIG is set to 1, the trigger signal *OSM_TRIG* (i.e. *TRIG_[x-1]* or *TIM_EXT_CAPTURE(x)*) triggers start of one pulse generation.

12.3.7.2 PWM Output with respect to configuration bit SL in one-shot mode: trigger by TRIG_[x-1] or TIM_EXT_CAPTURE(x)


CM0
CM1
CN0
SL=0:
TOM_CH[x]_OUT
SL=1:
TOM_CH[x]_OUT
enable channel
trigger TRIG_[x-1] or TIM_EXT_CAPTURE(x)
trigger TRIG_[x-1] or TIM_EXT_CAPTURE(x)
period
initial delay defined by CM0 – CN0

12.3.8 One-shot Counting Up-Down Mode

The TOM channel can operate in one-shot counting up-down mode when the bit OSM = 1 and the UDMODE != 0b00. One-shot mode means that a single pulse with the pulse level defined in bit SL is generated on the output line.

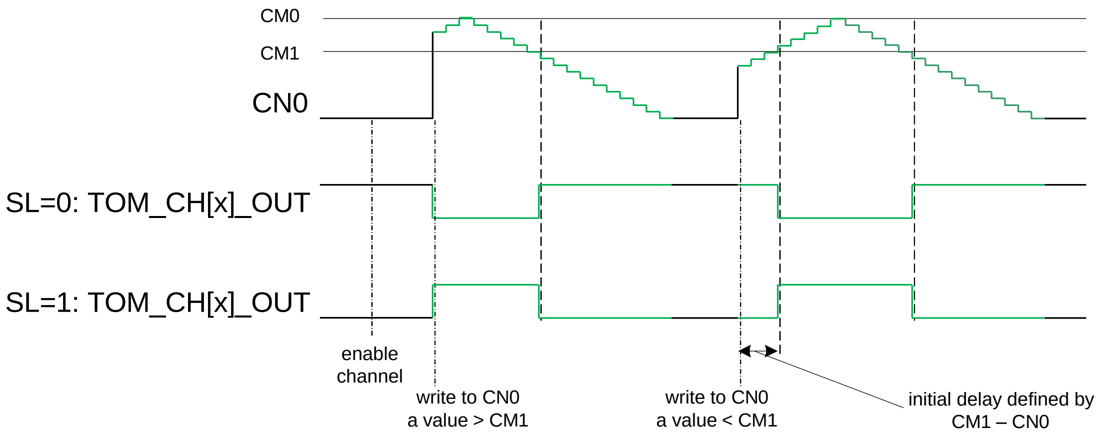
First the channel has to be enabled by setting the corresponding **ENDIS_STAT** value. In One-shot mode the counter **CN0** will not be incremented once the channel is enabled.
A write access to the register **CN0** triggers the start of pulse generation (i.e. the increment of the counter register **CN0**).

To avoid an update of **CMx** register with content of **SRx** register at this point in time, the automatic update should be disabled by setting UPEN_CTRL[x] = 0b00 (in register **TOM[i]_CH[x]_CTRL**)

If the counter **CN0** is greater or equal than **CM1**, the output *TOM[i]_CH[x]_OUT* is set to SL value.
If the counter **CN0** is less than **CM1**, the output *TOM[i]_CH[x]_OUT* is set to !SL value.
If the counter **CN0** has reached the value 0 (by counting down), it stops.
The new value of **CN0** determines the start delay of the first edge. The delay time of the first edge is given by (**CM1-CN0**) multiplied with period defined by current value of **CLK_SRC**.

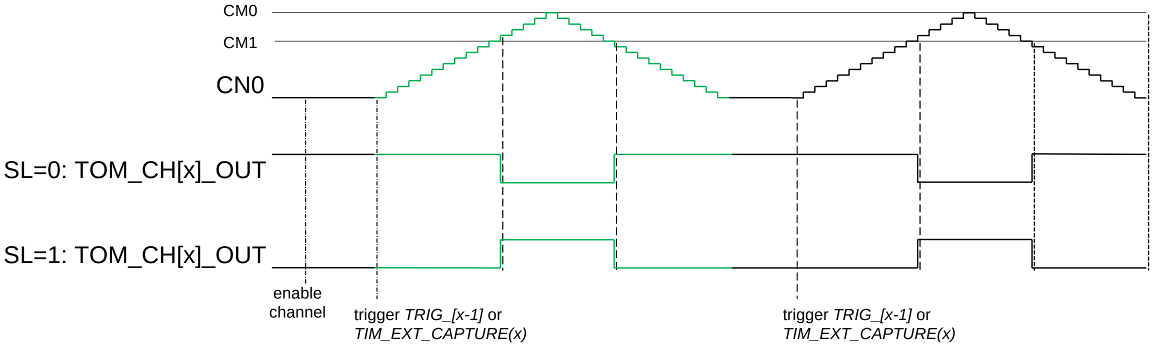
Figure 12.3.8.1 depicts the pulse generation in one-shot mode.

12.3.8.1 PWM Output with respect to configuration bit SL in one-shot counting up-down mode and UDMODE != 0b00: trigger by writing to CN0


CM0
CM1
CN0
SL=0: TOM_CH[x]_OUT
SL=1: TOM_CH[x]_OUT
enable channel
write to CN0 a value > CM1
write to CN0 a value < CM1
initial delay defined by CM1 – CN0

Further output of single pulses can be started by writing to register **CN0**.
If a channel is configured to one-shot counting up-down mode and configuration bit OSM_TRIG is set to 1, the trigger signal *OSM_TRIG* (i.e. *TRIG_[x-1]* or *TIM_EXT_CAPTURE(x)*) triggers start of one pulse generation.

12.3.8.2 PWM Output with respect to configuration bit SL in one-shot counting up-down mode and UDMODE != 0b00: trigger by TRIG_[x-1] or TIM_EXT_CAPTURE(x)


CM0
CM1
CN0
SL=0: TOM_CH[x]_OUT
SL=1: TOM_CH[x]_OUT
enable channel
trigger *TRIG_[x-1]* or *TIM_EXT_CAPTURE(x)*
trigger *TRIG_[x-1]* or *TIM_EXT_CAPTURE(x)*

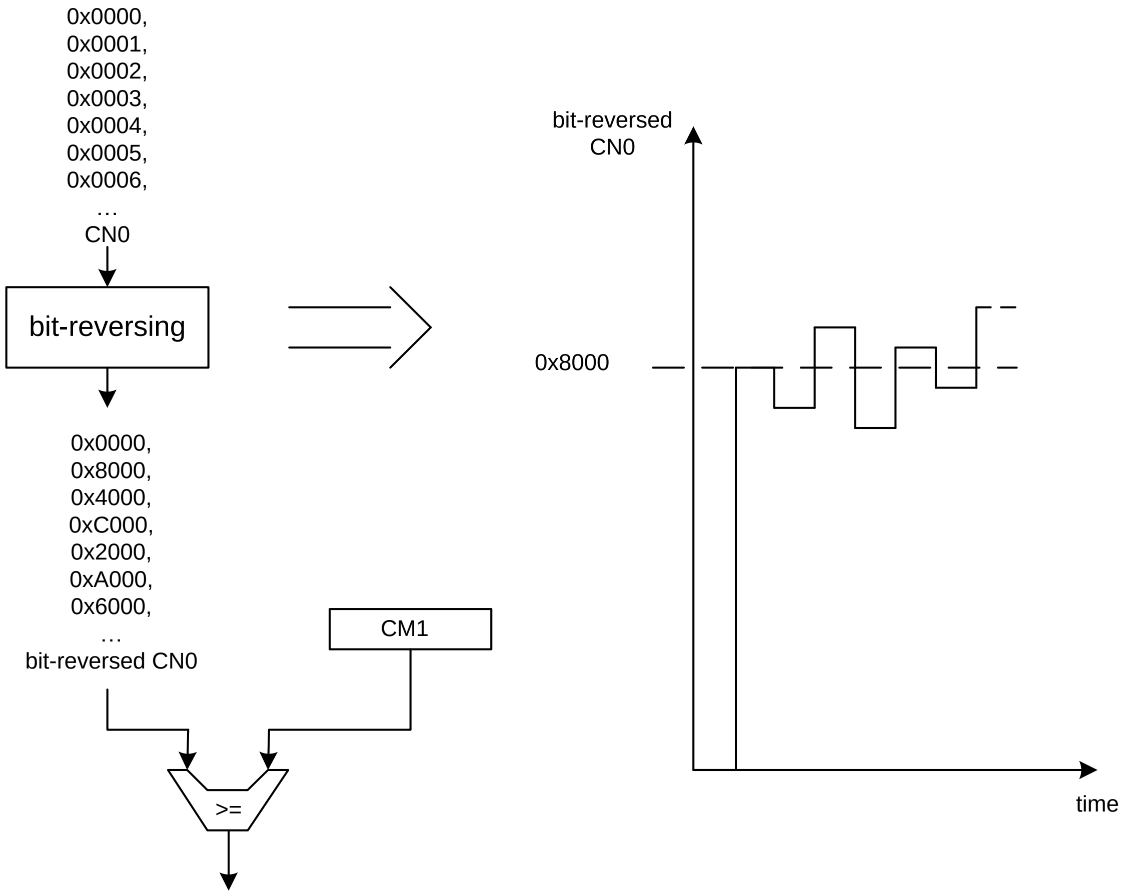
12.3.9 Pulse Count Modulation Mode

At the output *TOM[i]_CH15_OUT* a pulse count modulated signal can be generated instead of the simple PWM output signal.

Figure 12.3.3 outlines the circuit for Pulse Count Modulation.
The PCM mode is enabled by setting bit **BITREV** to 1.
With the configuration bit **BITREV**=1 a bit-reversing of the counter output **CN0** is configured. In this case the bits LSB and MSB are swapped, the bits LSB+1 and MSB-1 are swapped, the bits LSB+2 and MSB-2 are swapped and so on.

The effect of bit-reversing of the **CN0** register value is shown in the following figure 12.3.9.1.

*12.3.9.1 Bit reversing of counter **CN0** output*



In the PCM mode the counter register **CN0** is incremented by every clock tick depending on configured CMU clock (*CMU_FXCLK*).

The output of counter register **CN0** is first bit-reversed and then compared with the configured register value **CM1**.

If the bit-reversed value of register **CN0** is greater than **CM1**, the SR flip-flop of sub-module SOU is set (depending on configuration register **SL**) otherwise the SR flip-flop is reset. This generates at the output *TOM[i]_CH15_OUT* a pulse count modulated signal.

In PCM mode the **CM0** register - in which the period is defined - normally has to be set to its maximum value 0xFFFF.

To reduce time period of updating duty cycle value in **CM1** register, it is additionally possible to setup period value in **CM0** register to smaller values than maximum value as described before.
Possible values for **CM0** register are each even numbered values to the power of 2 e.g. 0x8000, 0x4000, 0x2000 ... .
In this case the duty cycle has to be configured in the following manner.

Depending on how much the period in CM0 register is decreased - means shifted right starting from 0x10000 - the duty cycle in CM1 register has to be shifted left (= rotated: shift MSB back into LSB) with same value, e.g. :

period CM0 = 0x0100 -> shifted 8 bits right from 0x10000
--> so duty cycle has to be shifted left 8 bit :
e.g. 50% duty cycle = 0x00080 -> shift 8 bits left -> CM1 = 0x8000

More examples :

period CM0	->	duty cycle	->	shift	->	CM1
0xFFFF	->	0x8000	->	no shift	->	0x8000
0x8000	->	0x4000	->	shift 1 bit left	->	0x8000
0x4000	->	0x1000	->	shift 2 bits left	->	0x4000
0x2000	->	0x0FFF	->	shift 3 bits left	->	0x7FF8
0x1000	->	0x0333	->	shift 4 bits left	->	0x3330
0x0800	->	0x0055	->	shift 5 bits left	->	0x0AA0
...						
0x0020	->	0x0008	->	shift 19 bits left	->	0x4000
0x0010	->	0x0005	->	shift 20 bits left	->	0x5000
...						

Note: In this mode the interrupt CCU1TC (see register **TOM[i]_CH[x]_IRQ_NOTIFY**) is set every time if bit reverse value of **CN0** is greater or equal than **CM1** which may be multiple times during one period. Therefore, from application point of view it is not useful to enable this interrupt.

12.3.10 Trigger Generation

For applications with constant PWM period defined by CM0, it is not necessary to update regularly the **CM0** register with **SR0** register. For these applications the **SR0** register can be used to define an additional output signal and interrupt trigger event.
If bit SR0_TRIG in register **TOM[i]_CH[x]_CTRL** is set, the register **SR0** is no longer used as a shadow register for register **CM0**. Instead, **SR0** is compared against **CN0** and if both are equal, a pulse of signal level '1' is generated at the output *TOM[i]_CH[x]_OUT_T*.
The bit SR0_TRIG should only be set if bit RST_CCU0 of this channel is 0.

If bit SR0_TRIG is set the interrupt notify flag CCU1TC is no longer set on a compare match of **CM1** and **CN0**. Instead, the CCU1TC interrupt notify flag is set in case of a compare equal match of **SR0** and **CN0**.

With configuration bit TRIG_PULSE one can select if the output *TOM[i]_CH[x]_OUT_T* is high as long as CN0=SR0 (TRIG_PULSE=0) or if there will be only one pulse of length one SYS_CLK period when CN0 becomes SR0 (TRIG_PULSE=1).

The TOM output signal routing to DTM or GTM-IP top level is described in chapter 14.7

12.4 TOM BLDC Support

The TOM sub-module offers in combination with the SPE sub-module a BLDC support. To drive a BLDC engine TOM channels 0 to 7 can be used.

The BLDC support can be configured by setting the **SPEM** bit inside the **TOM[i]_CH[z]_CTRL** register. When this bit is set the TOM channel output is controlled through the SPE_OUT(z) signal coming from the SPE sub-module (see figure 12.3.1). Please refer to chapter 19 for a detailed description of the SPE sub-module.

The TOM[i]_CH2,6,7,8 or 9 can be used together with the SPE module to trigger a delayed update of the **SPE_OUT_CTRL** register (i.e. commutation delay) after new input pattern detected by SPE (signaled by *SPE[i]_NIPD*). This feature is configured on TOM[i]_CH2,6,7,8 or 9 by setting SPE_TRIG=1 and OSM=1. With this configuration the TOM channel i generates one single PWM pulse on trigger by signal *SPE_NIPD*.
For details please refer to chapter of SPE sub-module description.

12.5 TOM Gated Counter Mode

Each TOM - SPE module combination provides also the feature of a gated counter mode. This is reached by using the *FSOI* input of a TIM module to gate the clock of a CCU0 sub-module.
To configure this mode, register of module SPE should be set as following:

- the SPE should be enabled (bit **SPE_EN** = 1),
- all three TIM inputs should be disabled (**SIE0** = **SIE1** = **SIE2** = 0),
- **SPE[i]_OUT_CTRL** should be set to 00005555h (set **SPE_OUT()** to '0') ,
- mode FSOM should be enabled (**FSOM**=1),
- set in bit field **FSOL** bit c if channel c of module TOM is chosen for gated counter mode

Additionally in module TOM

- the SPE mode should be disabled (**SPEM**=0) and
- the gated counter mode should be enabled (**GCM**=1)

As a result of this configuration, the counter **CN0** in sub-module CCU0 of TOM channel c counts as long as input *FSOI* is '0'.

12.6 TOM Interrupt signals

Signal	Description
TOM_CCU0TCx_IRQ	CCU0 Trigger condition interrupt for channel x
TOM_CCU1TCx_IRQ	CCU1 Trigger condition interrupt for channel x

12.7 TOM Configuration Register Overview

Register name	Description	Details in Section
TOM[i]_TGC[y]_GLB_CTRL (y:0...1)	TOMi TGC y global control register	12.8.1
TOM[i]_TGC[y]_ENDIS_CTRL (y:0...1)	TOMi TGC y enable/disable control register	12.8.2
TOM[i]_TGC[y]_ENDIS_STAT (y:0...1)	TOMi TGC y enable/disable status register	12.8.3
TOM[i]_TGC[y]_ACT_TB (y:0...1)	TOMi TGC y action time base register	12.8.4
TOM[i]_TGC[y]_OUTEN_CTRL (y:0...1)	TOMi TGC y output enable control register	12.8.5
TOM[i]_TGC[y]_OUTEN_STAT (y:0...1)	TOMi TGC y output enable status register	12.8.6
TOM[i]_TGC[y]_FUPD_CTRL (y:0...1)	TOMi TGC y force update control register	12.8.7
TOM[i]_TGC[y]_INT_TRIG (y:0...1)	TOMi TGC y internal trigger control register	12.8.8
TOM[i]_CH[x]_CTRL	TOMi channel x control register	12.8.9
TOM[i]_CH[x]_CN0	TOMi channel x CCU0 counter register	12.8.10
TOM[i]_CH[x]_CM0	TOMi channel x CCU0 compare register	12.8.11
TOM[i]_CH[x]_SR0	TOMi channel x CCU0 compare shadow register	12.8.12
TOM[i]_CH[x]_CM1	TOMi channel x CCU1 compare register	12.8.13
TOM[i]_CH[x]_SR1	TOMi channel x CCU1 compare shadow register	12.8.14
TOM[i]_CH[x]_STAT	TOMi channel x status register	12.8.15
TOM[i]_CH[x]_IRQ_NOTIFY	TOMi channel x interrupt notification register	12.8.16
TOM[i]_CH[x]_IRQ_EN	TOMi channel x interrupt enable register	12.8.17
TOM[i]_CH[x]_IRQ_FORCINT	TOMi channel x force interrupt register	12.8.18
TOM[i]_CH[x]_IRQ_MODE	TOMi channel x interrupt mode register	12.8.19

12.8 TOM Configuration Register Description

12.8.1 Register TOM[i]_TGC[y]_GLB_CTRL (y:0...1)

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	UPEN_CTRL7		UPEN_CTRL6		UPEN_CTRL5		UPEN_CTRL4		UPEN_CTRL3		UPEN_CTRL2		UPEN_CTRL1		UPEN_CTRL0		RST_CH7	RST_CH6	RST_CH5	RST_CH4	RST_CH3	RST_CH2	RST_CH1	RST_CH0	Reserved							HOST_TRIG
Mode	RW		RW		RW		RW		RW		RW		RW		RW		RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw	R							RAw
Initial Value	0b00		0b00		0b00		0b00		0b00		0b00		0b00		0b00		0	0	0	0	0	0	0	0	0x00							0

Bit 0 **HOST_TRIG** : trigger request signal (see TGC0, TGC1) to update the register ENDIS_STAT and OUTEN_STAT
0 = no trigger request
1 = set trigger request
Note: this flag is reset automatically after triggering the update

Bit 7:1 **Reserved**
Note: Read as zero, should be written as zero

Bit 8 **RST_CH0** : Software reset of channel 0
0 = No action
1 = Reset channel
Note: This bit is cleared automatically after write by CPU. The channel register are set to their reset values and channel operation is stopped immediately. The SR flip-flop SOUR is set to '1'.

Bit 9 **RST_CH1** : Software reset of channel 1
See bit 8

Bit 10 **RST_CH2** : Software reset of channel 2
See bit 8

Bit 11 **RST_CH3** : Software reset of channel 3
See bit 8

Bit 12 **RST_CH4** : Software reset of channel 4
See bit 8

Bit 13 **RST_CH5** : Software reset of channel 5
See bit 8

Bit 14 **RST_CH6** : Software reset of channel 6
See bit 8

Bit 15 **RST_CH7** : Software reset of channel 7
See bit 8

Bit 17:16 **UPEN_CTRL0**: TOM channel 0 enable update of register CM0, CM1 and CLK_SRC from SR0, SR1 and CLK_SRC_SR.
Write / Read :
0b00 = don't care, bits 1:0 will not be changed / update disabled
0b01 = disable update / --
0b10 = enable update / --
0b11 = don't care, bits 1:0 will not be changed / update enabled

Bit 19:18 **UPEN_CTRL1**: TOM channel 1 enable update of register CM0, CM1 and CLK_SRC
See bits 17:16

Bit 21:20 **UPEN_CTRL2**: TOM channel 2 enable update of register CM0, CM1 and CLK_SRC
See bits 17:16

Bit 23:22 **UPEN_CTRL3**: TOM channel 3 enable update of register CM0, CM1 and CLK_SRC
See bits 17:16

Bit 25:24 **UPEN_CTRL4**: TOM channel 4 enable update of register CM0, CM1 and CLK_SRC
See bits 17:16

Bit 27:26 **UPEN_CTRL5**: TOM channel 5 enable update of register CM0, CM1 and CLK_SRC
See bits 17:16

Bit 29:28 **UPEN_CTRL6**: TOM channel 6 enable update of register CM0, CM1 and CLK_SRC
See bits 17:16

Bit 31:30 **UPEN_CTRL7**: TOM channel 7 enable update of register CM0, CM1 and CLK_SRC
See bits 17:16

12.8.2 Register TOM[i]_TGC[y]_ENDIS_CTRL (y:0...1)

Address Offset:	**see Appendix B**																**Initial Value:**				**0x0000_0000**											
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																ENDIS_CTRL7		ENDIS_CTRL6		ENDIS_CTRL5		ENDIS_CTRL4		ENDIS_CTRL3		ENDIS_CTRL2		ENDIS_CTRL1		ENDIS_CTRL0	
Mode	R																RW		RW		RW		RW		RW		RW		RW		RW	
Initial Value	0x0000																0b00		0b00		0b00		0b00		0b00		0b00		0b00		0b00	

Bit 1:0 **ENDIS_CTRL0**: TOM channel 0 enable/disable update value.
If FREEZE=0:
If a TOM channel is disabled, the counter CN0 is stopped and the output register of SOU unit is set to the inverse value of control bit SL. On an enable event, the counter CN0 starts counting from its current value.
If FREEZE=1:
If a TOM channel is disabled, the counter CN0 is stopped. On an enable event, the counter CN0 starts counting from its current value.

Write of following double bit values is possible:
0b00 = don't care, bits 1:0 of register ENDIS_STAT will not be changed on an update trigger
0b01 = disable channel on an update trigger
0b10 = enable channel on an update trigger
0b11 = don't change bits 1:0 of this register

Note: if the output is disabled (OUTEN[0]=0), the TOM channel 0 output *TOM[i]_CH0_OUT* is the inverted value of bit SL.

Bit 3:2 **ENDIS_CTRL1**: TOM channel 1 enable/disable update value.
See bits 1:0

Bit 5:4 **ENDIS_CTRL2**: TOM channel 2 enable/disable update value.
See bits 1:0

Bit 7:6 **ENDIS_CTRL3**: TOM channel 3 enable/disable update value.
See bits 1:0

Bit 9:8 **ENDIS_CTRL4**: TOM channel 4 enable/disable update value.
See bits 1:0

Bit 11:10 **ENDIS_CTRL5**: TOM channel 5 enable/disable update value.
See bits 1:0

Bit 13:12 **ENDIS_CTRL6**: TOM channel 6 enable/disable update value.
See bits 1:0

Bit 15:14 **ENDIS_CTRL7**: TOM channel 7 enable/disable update value.
See bits 1:0

Bit 31:16 **Reserved**
Note: Read as zero, should be written as zero

12.8.3 Register TOM[i]_TGC[y]_ENDIS_STAT (y:0...1)

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																ENDIS_STAT7		ENDIS_STAT6		ENDIS_STAT5		ENDIS_STAT4		ENDIS_STAT3		ENDIS_STAT2		ENDIS_STAT1		ENDIS_STAT0	
Mode	R																RW		RW		RW		RW		RW		RW		RW		RW	
Initial Value	0x0000																0b00		0b00		0b00		0b00		0b00		0b00		0b00		0b00	

Bit 1:0 **ENDIS_STAT0**: TOM channel 0 enable/disable
If FREEZE=0:
If a TOM channel is disabled, the counter CN0 is stopped and the output register of SOU unit is set to the inverse value of control bit SL. On an enable event, the counter CN0 starts counting from its current value.
If FREEZE=1:
If a TOM channel is disabled, the counter CN0 is stopped. On an enable event, the counter CN0 starts counting from its current value.

Write / Read :
0b00 = don't care, bits 1:0 will not be changed / channel disabled
0b01 = disable channel / --
0b10 = enable channel / --
0b11 = don't care, bits 1:0 will not be changed / channel enabled

Bit 3:2 **ENDIS_STAT1**: TOM channel 1 enable/disable
See bits 1:0

Bit 5:4 **ENDIS_STAT2**: TOM channel 2 enable/disable
See bits 1:0

Bit 7:6 **ENDIS_STAT3**: TOM channel 3 enable/disable
See bits 1:0

Bit 9:8 **ENDIS_STAT4**: TOM channel 4 enable/disable
See bits 1:0

Bit 11:10 **ENDIS_STAT5**: TOM channel 5 enable/disable
See bits 1:0

Bit 13:12 **ENDIS_STAT6**: TOM channel 6 enable/disable

See bits 1:0

Bit 15:14 **ENDIS_STAT7**: TOM channel 7 enable/disable

See bits 1:0

Bit 31:16 **Reserved**

Note: Read as zero, should be written as zero

12.8.4 Register TOM[i]_TGC[y]_ACT_TB (y:0…1)

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_0000**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved					TBU_SEL		TB_TRIG	ACT_TB																							
Mode	R					RW		RAw	RW																							
Initial Value	0b00000					0b00		0	0x00_0000																							

Bit 23:0 **ACT_TB**: specifies the signed compare value with selected signal *TBU_TS*[x], x=0..2. If selected *TBU_TS*[x] value is in the interval [**ACT_TB**-007FFFFFh,**ACT_TB**] the event is in the past and the trigger is generated immediately. Otherwise the event is in the future and the trigger is generated if selected *TBU_TS*[x] is equal to **ACT_TB**.

Bit 24 **TB_TRIG**: Set trigger request

0 = no trigger request

1 = set trigger request

Note: This flag is reset automatically if the selected time base unit (*TBU_TS0* or *TBU_TS1* or *TBU_TS2* if present) has reached the value ACT_TB and the update of the register were triggered.

Bit 26:25 **TBU_SEL**: Selection of time base used for comparison

0b00 = *TBU_TS0* selected

0b01 = *TBU_TS1* selected

0b10 = *TBU_TS2* selected

0b11 = same as 0b00

Note: The bit combination 0b10 is only applicable if the TBU of the device contains three time base channels. Otherwise, this bit combination is also reserved. Please refer to GTM Architecture block diagram on page 3 to determine the number of channels for TBU of this device.

Bit 31:27 **Reserved**

Note: Read as zero, should be written as zero

12.8.5 Register TOM[i]_TGC[y]_OUTEN_CTRL (y:0…1)

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_0000**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																OUTEN_CTRL7		OUTEN_CTRL6		OUTEN_CTRL5		OUTEN_CTRL4		OUTEN_CTRL3		OUTEN_CTRL2		OUTEN_CTRL1		OUTEN_CTRL0	
Mode	R																RW		RW		RW		RW		RW		RW		RW		RW	
Initial Value	0x0000																0b00		0b00		0b00		0b00		0b00		0b00		0b00		0b00	

Bit 1:0 **OUTEN_CTRL0**: Output TOM[i]_CH0_OUT enable/disable update value
Write of following double bit values is possible:
0b00 = don't care, bits 1:0 of register OUTEN_STAT will not be changed on an update trigger
0b01 = disable channel output on an update trigger
0b10 = enable channel output on an update trigger
0b11 = don't change bits 1:0 of this register

Note: if the channel is disabled (ENDIS[0]=0) or the output is disabled (OUTEN[0]=0), the TOM channel 0 output *TOM[i]_CH0_OUT* is the inverted value of bit SL.

Bit 3:2 **OUTEN_CTRL1**: Output TOM[i]_CH1_OUT enable/disable update value
See bits 1:0

Bit 5:4 **OUTEN_CTRL2**: Output TOM[i]_CH2_OUT enable/disable update value
See bits 1:0

Bit 7:6 **OUTEN_CTRL3**: Output TOM[i]_CH3_OUT enable/disable update value
See bits 1:0

Bit 9:8 **OUTEN_CTRL4**: Output TOM[i]_CH4_OUT enable/disable update value
See bits 1:0

Bit 11:10 **OUTEN_CTRL5**: Output TOM[i]_CH5_OUT enable/disable update value
See bits 1:0

Bit 13:12 **OUTEN_CTRL6**: Output TOM[i]_CH6_OUT enable/disable update value
See bits 1:0

Bit 15:14 **OUTEN_CTRL7**: Output TOM[i]_CH7_OUT enable/disable update value
See bits 1:0

Bit 31:16 **Reserved**
Note: Read as zero, should be written as zero

12.8.6 Register TOM[i]_TGC[y]_OUTEN_STAT (y:0…1)

Address Offset:	see Appendix B																Initial Value: 0x0000_0000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																OUTEN_STAT7		OUTEN_STAT6		OUTEN_STAT5		OUTEN_STAT4		OUTEN_STAT3		OUTEN_STAT2		OUTEN_STAT1		OUTEN_STAT0	
Mode	R																RW		RW		RW		RW		RW		RW		RW		RW	
Initial Value	0x0000																0b00		0b00		0b00		0b00		0b00		0b00		0b00		0b00	

Bit 1:0 **OUTEN_STAT0**: Control/status of output TOM[i]_CH0_OUT
Write / Read :
0b00 = don't care, bits 1:0 will not be changed / output disabled
0b01 = disable output / --
0b10 = enable output / --
0b11 = don't care, bits 1:0 will not be changed / output enabled

Bit 3:2 **OUTEN_STAT1**: Control/status of output TOM[i]_CH1_OUT
See bits 1:0

Bit 5:4 **OUTEN_STAT2**: Control/status of output TOM[i]_CH2_OUT
See bits 1:0

Bit 7:6 **OUTEN_STAT3**: Control/status of output TOM[i]_CH3_OUT
See bits 1:0

Bit 9:8 **OUTEN_STAT4**: Control/status of output TOM[i]_CH4_OUT
See bits 1:0

Bit 11:10 **OUTEN_STAT5**: Control/status of output TOM[i]_CH5_OUT
See bits 1:0

Bit 13:12 **OUTEN_STAT6**: Control/status of output TOM[i]_CH6_OUT
See bits 1:0

Bit 15:14 **OUTEN_STAT7**: Control/status of output TOM[i]_CH7_OUT
See bits 1:0

Bit 31:16 **Reserved**
Note: Read as zero, should be written as zero

12.8.7 Register TOM[i]_TGC[y]_FUPD_CTRL (y:0…1)

Address Offset:	**see Appendix B**								**Initial Value:**				**0x0000_0000**			
	31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16	15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
Bit	RSTCN0_CH7	RSTCN0_CH6	RSTCN0_CH5	RSTCN0_CH4	RSTCN0_CH3	RSTCN0_CH2	RSTCN0_CH1	RSTCN0_CH0	FUPD_CTRL7	FUPD_CTRL6	FUPD_CTRL5	FUPD_CTRL4	FUPD_CTRL3	FUPD_CTRL2	FUPD_CTRL1	FUPD_CTRL0
Mode	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Initial Value	0b00	0b00	0b00	0b00	0b00	0b00	0b00	0b00	0b00	0b00	0b00	0b00	0b00	0b00	0b00	0b00

Bit 1:0 **FUPD_CTRL0**: Force update of TOM channel 0 operation register
If enabled, force update of register CM0, CM1 and CLK_SRC triggered by HOST_TRIG, ACT_TB compare match or internal trigger.
Write / Read :
0b00 = don't care, bits 1:0 will not be changed / force update disabled
0b01 = disable force update / --
0b10 = enable force update / --
0b11 = don't care, bits 1:0 will not be changed / force update enabled
Note: The force update request is stored and executed synchronized to the selected FXCLK.

Bit 3:2 **FUPD_CTRL1**: Force update of TOM channel 1 operation register
See bits 1:0

Bit 5:4 **FUPD_CTRL2**: Force update of TOM channel 2 operation register
See bits 1:0

Bit 7:6 **FUPD_CTRL3**: Force update of TOM channel 3 operation register
See bits 1:0

Bit 9:8 **FUPD_CTRL4**: Force update of TOM channel 4 operation register
See bits 1:0

Bit 11:10 **FUPD_CTRL5**: Force update of TOM channel 5 operation register
See bits 1:0

Bit 13:12 **FUPD_CTRL6**: Force update of TOM channel 6 operation register
See bits 1:0

Bit 15:14 **FUPD_CTRL7**: Force update of TOM channel 7 operation register
See bits 1:0

Bit 17:16 **RSTCN0_CH0**: Reset CN0 of channel 0 on force update event
If enabled, reset CN0 triggered by HOST_TRIG, ACT_TB compare match or internal trigger.
Write / Read :
0b00 = don't care, bits 1:0 will not be changed / CN0 is not reset on forced update
0b01 = do not reset CN0 on forced update / --
0b10 = reset CN0 on forced update / --

0b11 = don't care, bits 1:0 will not be changed / CN0 is reset on forced update

Bit 19:18 **RSTCN0_CH1**: Reset CN0 of channel 1 on force update event
See bits 17:16

Bit 21:20 **RSTCN0_CH2**: Reset CN0 of channel 2 on force update event
See bits 17:16

Bit 23:22 **RSTCN0_CH3**: Reset CN0 of channel 3 on force update event
See bits 17:16

Bit 25:24 **RSTCN0_CH4**: Reset CN0 of channel 4 on force update event
See bits 17:16

Bit 27:26 **RSTCN0_CH5**: Reset CN0 of channel 5 on force update event
See bits 17:16

Bit 29:28 **RSTCN0_CH6**: Reset CN0 of channel 6 on force update event
See bits 17:16

Bit 31:30 **RSTCN0_CH7**: Reset CN0 of channel 7 on force update event
See bits 17:16

12.8.8 Register TOM[i]_TGC[y]_INT_TRIG (y:0...1)

Address Offset:	see Appendix B																Initial Value:									0x0000_0000						
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																INT_TRIG7		INT_TRIG6		INT_TRIG5		INT_TRIG4		INT_TRIG3		INT_TRIG2		INT_TRIG1		INT_TRIG0	
Mode	R																RW		RW		RW		RW		RW		RW		RW		RW	
Initial Value	0x0000																0b00		0b00		0b00		0b00		0b00		0b00		0b00		0b00	

Bit 1:0 **INT_TRIG0**: Select input signal *TRIG_0* as a trigger source
Write / Read :
0b00 = don't care, bits 1:0 will not be changed / internal trigger from channel 0 (*TRIG_0*) not used
0b01 = do not use internal trigger from channel 0 (*TRIG_0*) / --
0b10 = use internal trigger from channel 0 (*TRIG_0*) / --
0b11 = don't care, bits 1:0 will not be changed / internal trigger from channel 0 (*TRIG_0*) used

Bit 3:2 **INT_TRIG1**: Select input signal *TRIG_1* as a trigger source
See bits 1:0

Bit 5:4 **INT_TRIG2**: Select input signal *TRIG_2* as a trigger source

See bits 1:0

Bit 7:6 **INT_TRIG3**: Select input signal *TRIG_3* as a trigger source
See bits 1:0

Bit 9:8 **INT_TRIG4**: Select input signal *TRIG_4* as a trigger source
See bits 1:0

Bit 11:10 **INT_TRIG5**: Select input signal *TRIG_5* as a trigger source
See bits 1:0

Bit 13:12 **INT_TRIG6**: Select input signal *TRIG_6* as a trigger source
See bits 1:0

Bit 15:14 **INT_TRIG7**: Select input signal *TRIG_7* as a trigger source
See bits 1:0

Bit 31:16 **Reserved**
Note: Read as zero, should be written as zero

12.8.9 Register TOM[i]_CH[x]_CTRL

Address Offset:	**see Appendix B**																**Initial Value:**					**0x0000_0X00**											
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	FREEZE	Reserved	GCM	SPEM	BITREV	OSM	SPE_TRIG	TRIGOUT	EXTTRIGOUT	EXT_TRIG	OSM_TRIG	RST_CCU0	UDMODE		TRIG_PULSE	Reserved	ECLK_SRC	CLK_SRC_SR			SL	Reserved			SR0_TRIG	Reserved						
Mode	RW	R	RW	RW	RW	RW	RW	RW	RW	RW	RPw	RPw	RW		RW	R	RW	RW			RW	R			RPw	R						
Initial Value	0	0	0	0	0	0	0	0	0	0	0	0	0b00		0	0	0	0b000			X	0b000			0	0x00						

Bit 6:0 **Reserved**
Note: Read as zero, should be written as zero

Bit 7 **SR0_TRIG**: **SR0** is used to generate a trigger on output *TOM[i]_CH[x]_OUT_T* if equal to **CN0**.
0 = SR0 is used as a shadow register for register CM0.
1 = SR0 is not used as a shadow register for register CM0. SR0 is compared with CN0 and if both are equal, a trigger pulse is generated at output *TOM[i]_CH[x]_OUT_T* .
Note: This bit should only be set if RST_CCU0 of this channel is 0.

Bit 10:8 **Reserved**
Note: Read as zero, should be written as zero

Bit 11 **SL**: Signal level for duty cycle
0 = Low signal level
1 = High signal level

If the output is disabled, the output TOM_OUT[x] is set to inverse value of SL.

Note: Reset value depends on the hardware configuration chosen by silicon vendor.

Bit 14:12 **CLK_SRC_SR**: Clock source select for channel

The register CLK_SRC is updated with the value of CLK_SRC_SR together with the update of register CM0 and CM1.

The input of the FX clock divider depends on the value of FXCLK_SEL (see CMU).

if ECLK_SRC=0 / ECLK_SRC=1:

0b000 = *CMU_FXCLK*(0) selected / *CMU_FXCLK*(0) selected
0b001 = *CMU_FXCLK*(1) selected / *CMU_FXCLK*(1) selected
0b010 = *CMU_FXCLK*(2) selected / *CMU_FXCLK*(2) selected
0b011 = *CMU_FXCLK*(3) selected / *CMU_FXCLK*(3) selected
0b100 = *CMU_FXCLK*(4) selected / *CMU_FXCLK*(4) selected
0b101 = clock of channel stopped / *TRIG[x-1]* selected
0b110 = clock of channel stopped / *TIM_EXT_CAPTURE[x]* selected
0b111 = clock of channel stopped / Reserved

Note: This register is a shadow register for the register CLK_SRC. Thus, if the CMU_CLK source for PWM generation should be changed during operation, the old CMU_CLK has to operate until the update of the ATOM channels internal CLK_SRC register by the CLK_SRC_SR content is done either by an end of a period or a forced update.

Note: if clock of channel is stopped (i.e. ECLK_SRC=0 and CLK_SRC=0b101/0b110/0b111) , the channel can only be restarted by resetting CLK_SRC_SR to a value of 0b000 to 0b100 and forcing an update via the force update mechanism.

Bit 15 **ECLK_SRC:** Extend CLK_SRC

0: CLK_SRC_SR set 1 selected (see bit CLK_SRC_SR)
1: CLK_SRC_SR set 2 selected (see bit CLK_SRC_SR)

Bit 16 **Reserved**

Note: Read as zero, should be written as zero

Bit 17 **TRIG_PULSE**: Trigger output pulse length of one SYS_CLK period

0 = output on TOM[i]_OUT[x]_T is 1 as long as CN0=SR0 (if SR=_TRIG=1)

1 = output on TOM[i]_OUT[x]_T is 1 for only one SYS_CLK period if CN0=SR0 (if SR=_TRIG=1)

Bit 19:18 **UDMODE**: up-down counter mode

0b00 = up-down counter mode disabled: CN0 counts always up

0b01 = up-down counter mode enabled: CN0 counts up and down, CM0, CM1 are updated if CN0 reaches 0 (i.e. changes from down to up)

0b10 = up-down counter mode enabled: CN0 counts up and down, CM0, CM1 are updated if CN0 reaches CM0 (i.e. changes from up to down)

0b11 = up-down counter mode enabled: CN0 counts up and down, CM0, CM1 are updated if CN0 reaches 0 or CM0 (i.e. changes direction)

Bit 20 **RST_CCU0**: Reset source of CCU0

0 = Reset counter register CN0 to 0 on matching comparison CM0

1 = Reset counter register CN0 to 0 on trigger *TRIG_[x-1]* or TIM_*EXT_CAPTURE(x)*.

Note: On TOM channel 2 SPEM=1 has special meaning.

If SPEM = 1, the signal *SPE_NIPD* triggers the reset of CN0 independent of RST_CN0.

Note: This bit should only be set if bit OSM=0 (i.e. in continuous mode)

Bit 21 **OSM_TRIG**: enable trigger of one-shot pulse by trigger signal *OSM_TRIG*

0 = signal OSM_TRIG cannot trigger start of single pulse generation

1 = signal OSM_TRIG can trigger start of single pulse generation (only if bit OSM = 1)

Note: This bit should only be set if bit OSM=1 and bit RST_CCU0=0.

Bit 22 **EXT_TRIG**: select *TIM_EXT_CAPTURE(x)* as trigger signal

0 = signal *TIM_[x-1]* is selected as trigger to reset CN0 or to start single pulse generation.

1 = signal *TIM_EXT_CAPTURE(x)* is selected

Bit 23 EXTTRIGOUT: select *TIM_EXT_CAPTURE(x)* as potential output signal *TRIG_[x]*

0 = signal *TRIG_[x-1]* is selected as output on *TRIG_[x]* (if TRIGOUT=1)

1 = signal *TIM_EXT_CAPTURE(x)* is selected as output on *TRIG_[x]* (if TRIGOUT=1)

Bit 24 **TRIGOUT**: Trigger output selection (output signal *TRIG_[x]*) of module TOM_CH[x]

0 = *TRIG_[x]* is *TRIG_[x-1]* or TIM_*EXT_CAPTURE(x)*.

1 = *TRIG_[x]* is *TRIG_CCU0*

Bit 25 **SPE_TRIG:** SPE trigger to reset CN0

For TOM channel 2,6 and 7 this bit defines in combination with bit SPEM the source of output pin *TOM[i]_CH[x]_OUT* and if CN0 can be reset by TOM input signal *SPE[i]_NIPD*.

If SPEM=0 / SPEM=1 :

0 = *TOM[i]_CH[x]_OUT* defined by TOM[i] channel x SOUR register, CN0 reset is defined by configuration of bit RST_CCU0 / *TOM[i]_CH[x]_OUT* is defined by *SPE[i]_OUT[x]*, CN0 is reset by signal *SPE[i]_NIPD*

1 = *TOM[i]_CH[x]_OUT* defined by TOM[i] channel x SOUR register, CN0 is reset by signal *SPE[i]_NIPD* / *TOM[i]_CH[x]_OUT* is defined by *SPE[i]_OUT[x]*, CN0 reset is defined by configuration of bit RST_CCU0

Note: For TOM channel 8 and 9 this bit defines only if CN0 reset is defined by input signal *SPE[i]_NIPD* or by configuration of RST_CCU0. The output *TOM[i]_CH[x]_OUT* is not affected.
The configuration bit SPEM is not available for these channels and thus assumed to be 0.

Note: If a configuration of SPEM | SPE_TRIG = 0 | 1 or 1 | 0 is chosen (i.e. CN0 is reset by signal *SPE[i]_NIPD*), the one-shot mode in corresponding TOM channel should also be enabled by setting bit OSM=1 to generate one PWM pulse in case of trigger *SPE[i]_NIPD*.
Note: In SPE module one of the trigger signals *TOM[i]_CH2_TRIG_CCU1*, *TOM[i]_CH6_TRIG_CCU1*, *TOM[i]_CH7_TRIG_CCU1*, *TOM[i]_CH8_TRIG_CCU1*, or *TOM[i]_CH9_TRIG_CCU1* can be used to trigger the update of register SPE[i]_OUT_CTRL.

Bit 26 **OSM**: One-shot mode. In this mode the counter CN0 counts for only one period. The length of period is defined by CM0. A write access to the register CN0 triggers the start of counting.
0 = One-shot mode disabled
1 = One-shot mode enabled

Bit 27 **BITREV**: Bit-reversing of output of counter register **CN0**.
Note: This bit enables the PCM mode of channel 15.

Bit 28 **SPEM**: SPE output mode enable for channel.
0 = SPE output mode disabled: *TOM[i]_CH[x]_OUT* defined by TOM[i] channel x SOUR register
1 = SPE output mode enabled: *TOM[i]_CH[x]_OUT* is defined by *SPE[i]_OUT[x]*

Note: The SPE output mode is only implemented for TOM instances connected to a SPE module and only for TOM channels 0 to 7.

Note: For TOM channel 2,6 and 7 this bit defines in combination with bit SPE_TRIG the source of output pin *TOM[i]_CH[x]_OUT* and if CN0 can be reset by TOM input signal *SPE[i]_NIPD*.

Bit 29 **GCM**: Gated Counter Mode enable
0 = Gated Counter mode disabled
1 = Gated Counter mode enabled
Note: The Gated Counter mode is only available for TOM instances connected to a SPE module and only for channels 0 to 7.

Bit 30 **Reserved**

Note: Read as zero, should be written as zero

Bit 31 **FREEZE**

0 = a channel disable/enable may change internal register and output register

1 = a channel enable/disable does not change an internal or output register but stops counter **CN0**

12.8.10 Register TOM[i]_CH[x]_CN0

Address Offset:	see Appendix B																Initial Value: 0x0000_0000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																CN0															
Mode	R																RW															
Initial Value	0x0000																0x0000															

Bit 15:0 **CN0**: TOM CCU0 counter register

This counter is stopped if the TOM channel is disabled and not reset on an enable event of TOM channel.

Bit 31:16 **Reserved**

Note: Read as zero, should be written as zero

12.8.11 Register TOM[i]_CH[x]_CM0

Address Offset:	see Appendix B																Initial Value: 0x0000_0000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																CM0															
Mode	R																RW															
Initial Value	0x0000																0x0000															

Bit 15:0 **CM0**: TOM CCU0 compare register
Setting CM0 < CM1 configures a duty cycle of 100%.

Bit 31:16 **Reserved**
Note: Read as zero, should be written as zero

12.8.12 Register TOM[i]_CH[x]_SR0

<table>
<tr><td><b>Address Offset:</b></td><td colspan="16"><b>see Appendix B</b></td><td colspan="16"><b>Initial Value: 0x0000_0000</b></td></tr>
<tr><td></td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>Bit</td><td colspan="16">Reserved</td><td colspan="16">SR0</td></tr>
<tr><td>Mode</td><td colspan="16">R</td><td colspan="16">RW</td></tr>
<tr><td>Initial Value</td><td colspan="16">0x0000</td><td colspan="16">0x0000</td></tr>
</table>

Bit 15:0 **SR0**: TOM channel x shadow register SR0 for update of compare register CM0

Bit 31:16 **Reserved**
Note: Read as zero, should be written as zero

12.8.13 Register TOM[i]_CH[x]_CM1

<table>
<tr><td><b>Address Offset:</b></td><td colspan="16"><b>see Appendix B</b></td><td colspan="16"><b>Initial Value: 0x0000_0000</b></td></tr>
<tr><td></td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>Bit</td><td colspan="16">Reserved</td><td colspan="16">CM1</td></tr>
<tr><td>Mode</td><td colspan="16">R</td><td colspan="16">RW</td></tr>
<tr><td>Initial Value</td><td colspan="16">0x0000</td><td colspan="16">0x0000</td></tr>
</table>

Bit 15:0 **CM1**: TOM CCU1 compare register
Setting CM1 = 0 configures a duty cycle of 0% independent of the configured value of CM0.

Bit 31:16 **Reserved**
Note: Read as zero, should be written as zero

12.8.14 Register TOM[i]_CH[x]_SR1

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_0000**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																SR1															
Mode	R																RW															
Initial Value	0x0000																0x0000															

Bit 15:0 **SR1**: TOM channel x shadow register SR1 for update of compare register CM1

Bit 31:16 **Reserved**
Note: Read as zero, should be written as zero

12.8.15 Register TOM[i]_CH[x]_STAT

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_000x**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																															OL
Mode	R																															R
Initial Value	0x0000 000																															x

Bit 0 **OL**: Output level of output *TOM_OUT(x)*
Note: Reset value is the inverted value of SL bit which depends on the hardware configuration chosen by silicon vendor.

Bit 31:1 **Reserved**

Note: Read as zero, should be written as zero

12.8.16 Register TOM[i]_CH[x]_IRQ_NOTIFY

Address Offset:	**see Appendix B**																**Initial Value:**					**0x0000_0000**										
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																														CCU1TC	CCU0TC
Mode	R																														RCw	RCw
Initial Value	0x0000000																														0	0

Bit 0 **CCU0TC**: CCU0 Trigger condition interrupt for channel x
0 = No interrupt occurred
1 = The condition CN0 >= CM0 was detected.
The notification of the interrupt is only triggered one time after reaching the condition CN0 >= CM0. To enable re-trigger of the notification first the condition CN0 < CM1 has to be reached.

Bit 1 **CCU1TC**: CCU1 Trigger condition interrupt for channel x
0 = No interrupt occurred
If SR0_TRIG=0 / SR0_TRIG=1 :
1 = the condition CN0 >= CM1 was detected / the condition SR0=CN0 was detected
Note: The notification of the interrupt is only triggered one time after reaching the condition CN0 >= CM1. To enable re-trigger of the notification first the condition CN0 < CM1 has to be reached.

Bit 31:2 **Reserved**
Note: Read as zero, should be written as zero

12.8.17 Register TOM[i]_CH[x]_IRQ_EN

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																														CCU1TC_IRQ_EN	CCU0TC_IRQ_EN
Mode	R																														RW	RW
Initial Value	0x0000 000																														0	0

Bit 0 **CCU0TC_IRQ_EN**: *TOM_CCU0TC_IRQ* interrupt enable
0 = Disable interrupt, interrupt is not visible outside GTM-IP
1 = Enable interrupt, interrupt is visible outside GTM-IP

Bit 1 **CCU1TC_IRQ_EN**: *TOM_CCU1TC_IRQ* interrupt enable
See bit 0

Bit 31:2 **Reserved**
Note: Read as zero, should be written as zero

12.8.18 Register TOM[i]_CH[x]_IRQ_FORCINT

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																														TRG_CCU1TC0	TRG_CCU0TC0
Mode	R																														RAw	RAw
Initial Value	0x0000 000																														0	0

Bit 0 **TRG_CCU0TC0**: Trigger *TOM_CCU0TC0_IRQ* interrupt by software
0 = No interrupt triggering
1 = Assert *CCU0TC0_IRQ* interrupt for one clock cycle
Note: This bit is cleared automatically after write.
Note: This bit is write protected by bit RF_PROT of register GTM_CTRL

Bit 1 **TRG_CCU1TC0**: Trigger *TOM_CCU1TC0_IRQ* interrupt by software
0 = No interrupt triggering

1 = Assert *CCU1TC0_IRQ* interrupt for one clock cycle
Note: This bit is cleared automatically after write.
Note: This bit is write protected by bit RF_PROT of register GTM_CTRL

Bit 31:2 **Reserved**
Note: Read as zero, should be written as zero

12.8.19 Register TOM[i]_CH[x]_IRQ_MODE

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_000X**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																														IRQ_MODE	
Mode	R																														RW	
Initial Value	0x0000 0000																														XX	

Bit 1:0 **IRQ_MODE**: IRQ mode selection
0b00 = Level mode
0b01 = Pulse mode
0b10 = Pulse-Notify mode
0b11 = Single-Pulse mode
Note: The interrupt modes are described in section 2.5.

Bit 31:2 **Reserved**
Note: Read as zero, should be written as zero

13 ARU-connected Timer Output Module (ATOM)

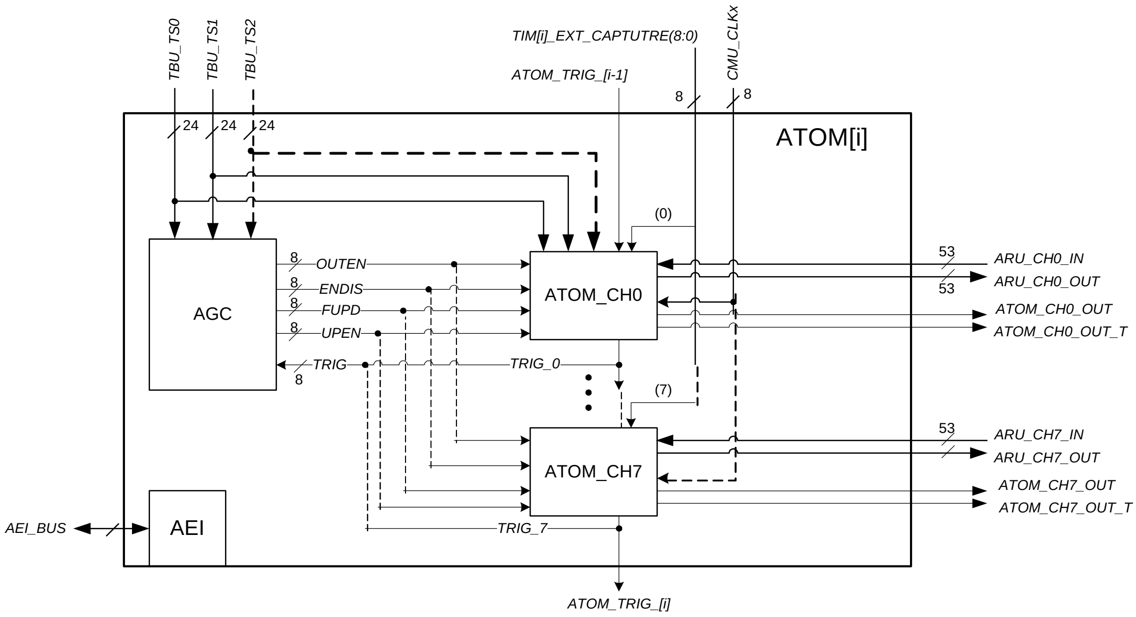
13.1 Overview

The ARU-connected Timer Output Module (ATOM) is able to generate complex output signals without CPU interaction due to its connectivity to the ARU. Typically, output signal characteristics are provided over the ARU connection through sub-modules connected to ARU like e.g. the MCS, DPLL or PSM. Each ATOM sub-module contains eight output channels which can operate independently from each other in several configurable operation modes. A block diagram of the ATOM sub-module is depicted in figure 13.1.1.

The following design variables are used inside this chapter. Please refer to device specific Appendix B for correct value.

cCATO : ATOM channel count; number of channels per instance - 1

13.1.1 ATOM block diagram



The architecture of the ATOM sub-module is similar to the TOM sub-module, but there are some differences. First, the ATOM integrates only eight output channels. Hence, there exists one ATOM Global Control sub-unit (AGC) for the ATOM channels. The ATOM is connected to the ARU and can set up individual read requests from the ARU and write requests to the ARU. Furthermore, the ATOM channels are able to generate signals on behalf of time stamps and the ATOM channels are able to generate a serial output signal on behalf of an internal shift register.

Each ATOM channel provides five modes of operation:

- ATOM Signal Output Mode Immediate (SOMI)
- ATOM Signal Output Mode Compare (SOMC)
- ATOM Signal Output Mode PWM (SOMP)
- ATOM Signal Output Mode Serial (SOMS)
- ATOM Signal Output Mode Buffered Compare (SOMB)

These modes are described in more detail in section 13.3.
The ATOM channels' operation registers (e.g. counter, compare registers) are 24 bit wide. Moreover, the input clocks for the ATOM channels come from the configurable *CMU_CLKx* signals of the CMU sub-module. This gives the freedom to select a programmable input clock for the ATOM channel counters. The ATOM channel is able to generate a serial bit stream, which is shifted out at the *ATOM[i]_CH[x]_OUT* output. When configured in this serial shift mode (SOMS) the selected CMU clock defines the shift frequency.

Each ATOM channel provides a so called *operation* and *shadow* register set. With this architecture it is possible to work with the operation register set, while the shadow register set can be reloaded with new parameters over CPU and/or ARU.

When update via ARU is selected, it is possible to configure for ATOM SOMP mode if both shadow registers are updated via ARU or only one of the shadow registers is updated.

On the other hand, the shadow registers can be used to provide data to the ARU when one or both of the compare units inside an ATOM channel match. This feature is only applicable in SOMC mode.

In TOM channels it is possible to reload the content of the operation registers with the content of the corresponding shadow registers and change the clock input signal for the counter register simultaneously. This simultaneous change of the input clock frequency together with reloading the operation registers is also implemented in the ATOM channels.

In addition to the feature that the CPU can select another *CMU_CLKx* during operation (i.e. updating the shadow register bit field CLK_SRC_SR of the **ATOM[i]_CH[x]_CTRL** register), the selection can also be changed via the ARU. Then, for the clock source update, the ACBI register bits of the **ATOM[i]_CH[x]_STAT** register are used as a shadow register for the new clock source.

In general, the behavior of the compare units CCU0 and CCU1 and the output signal behavior is controlled with the ACB bit field inside the **ATOM[i]_CH[x]_CTRL** register when the ARU connection is disabled and the behavior is controlled via ARU through the ACBI bit field of the **ATOM[i]_CH[x]_STAT** register, when the ARU is enabled.

Since the ATOM is connected to the ARU, the shadow registers of an ATOM channel can be reloaded via the ARU connection or via CPU over its AEI interface. When loaded via the ARU interface, the shadow registers act as a buffer between the ARU and the channel operation registers. Thus, a new parameter set for a PWM can be reloaded via ARU into the shadow registers, while the operation registers work on the actual parameter set.

The trigger signal *ATOM_TRIG_[i-1]* of ATOM instance i comes from the preceding instance i-1, the trigger *ATOM_TRIG_[i]* is routed to succeeding instance i+1.
Note, ATOM0 is connected to its own output ATOM_TRIG_0, i.e. the last channel of ATOM instance 0 can trigger the first channel of ATOM instance 0 (this path is registered, which means delayed by one SYS_CLK period).

13.1.2 ATOM Global Control (AGC)

Synchronous start, stop and update of work register of up to 8 channels is possible with the AGC sub-unit. This sub-unit has the same functionality as the TGC sub-unit of the TOM sub-module.

13.1.2.1 Overview

There exists one global channel control unit (AGC) to drive a number of individual ATOM channels synchronously by external or internal events.

An AGC can drive up to eight ATOM channels.
The ATOM sub-module supports four different kinds of signaling mechanisms:

- Global enable/disable mechanism for each ATOM channel with control register **ATOM[i]_AGC_ENDIS_CTRL** and status register **ATOM[i]_AGC_ENDIS_STAT**

- Global output enable mechanism for each ATOM channel with control register **ATOM[i]_AGC_OUTEN_CTRL** and status register **ATOM[i]_AGC_OUTEN_STAT**

- Global force update mechanism for each ATOM channel with control register **ATOM[i]_AGC_FUPD_CTRL**
- Update enable of the register **CM0**, **CM1** and **CLK_SRC** for each ATOM channel with the control bit field **UPEN_CTRL[z] of ATOM[i]_AGC_GLB_CTRL**

13.1.2.2 AGC Sub-unit

Each of the first three individual mechanisms (enable/disable of the channel, output enable and force update) can be driven by three different trigger sources.
The three trigger sources are :

- the host CPU (bit **HOST_TRIG** of register **ATOM[i]_AGC_GLB_CTRL**)
- the TBU time stamp (signal *TBU_TS0..2* if available)
- the internal trigger signal *TRIG* (bunch of trigger signals *TRIG_[x]*)
  which can be either the trigger *TRIG_CCU0* of channel x,
  the trigger of preceding channel x-1 (i.e. signal *TRIG_[x-1]*) or
  the external trigger *TIM_EXT_CAPTURE(x)* of assigned TIM channel x.

The first way is to trigger the control mechanism by a direct register write access via host CPU (bit **HOST_TRIG** of register **AOM[i]_AGC_GLB_CTRL**).

The second way is provided by a compare match trigger on behalf of a specified time base coming from the module TBU (selected by bits **TBU_SEL**) and the time stamp compare value defined in the bit field **ACT_TB** of register **ATOM[i]_AGC_ACT_TB**.
Note, a cyclic event compare of **ACT_TB** and selected *TBU_TS*[*x*] is performed.

The third possibility is the input *TRIG* (bunch of trigger signals *TRIG_[x]*) coming from the ATOM channels 0 to 7.

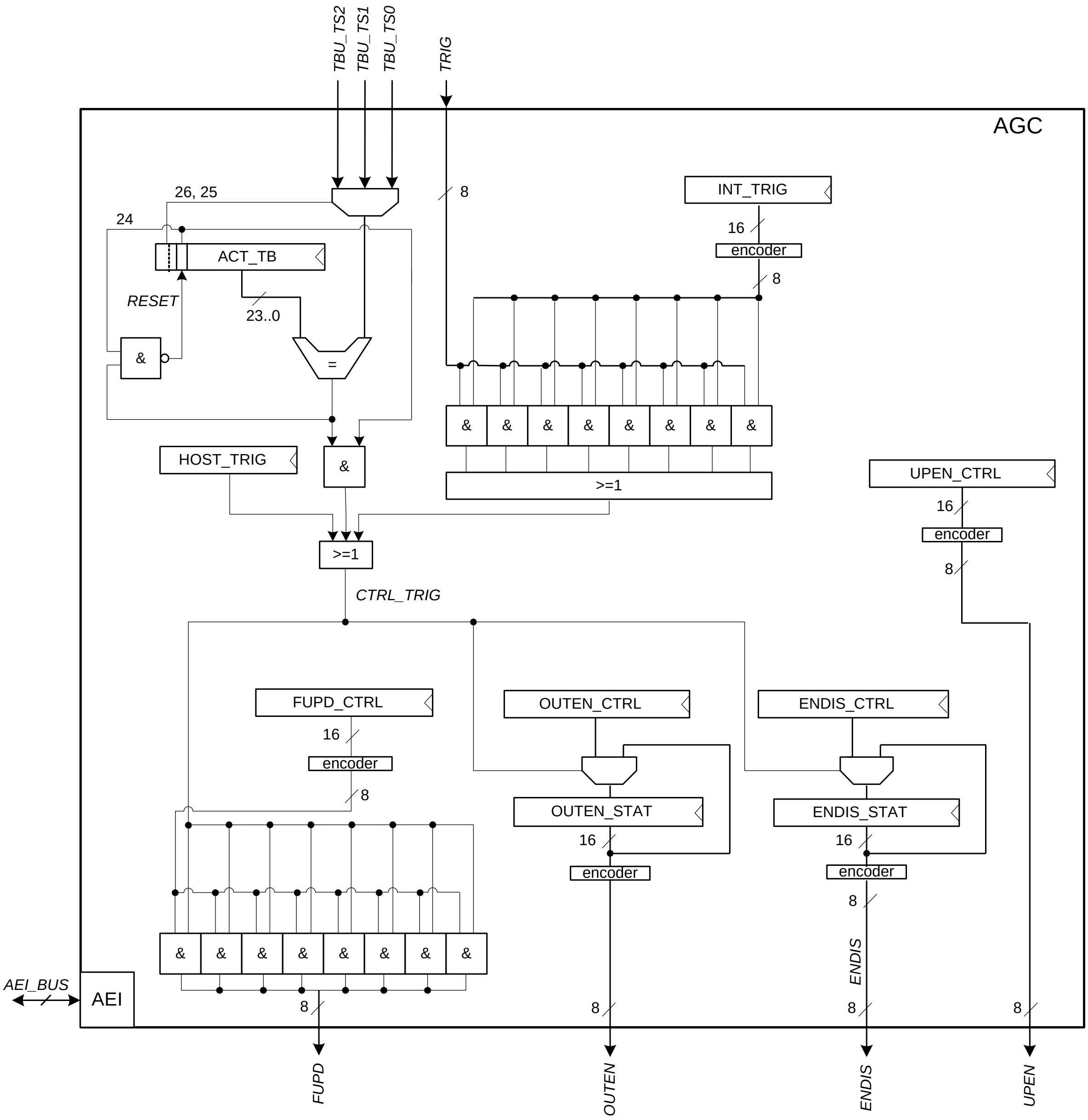
The corresponding trigger signal *TRIG_[x]* coming from channel [x] can be masked by the register **ATOM[i]_AGC_INT_TRIG**.

To enable or disable each individual ATOM channel, the registers **ATOM[i]_AGC_ENDIS_CTRL** and/or **ATOM[i]_AGC_ENDIS_STAT** have to be used.

The register **ATOM[i]_AGC_ENDIS_STAT** controls directly the signal *ENDIS*. A write access to this register is possible.

The register **ATOM[i]_AGC_ENDIS_CTRL** is a shadow register that overwrites the value of register **AOM[i]_AGC_ENDIS_STAT** if one of the three trigger conditions matches.

13.1.2.2.1 ATOM Global channel control mechanism



The output of the individual ATOM channels can be controlled using the register **ATOM[i]_AGC_OUTEN_CTRL** and **ATOM[i]_AGC_OUTEN_STAT**.

The register **ATOM[i]_AGC_OUTEN_STAT** controls directly the signal *OUTEN*. A write access to this register is possible.

The register **ATOM[i]_AGC_OUTEN_CTRL** is a shadow register that overwrites the value of register **ATOM[i]_AGC_OUTEN_STAT** if one of the three trigger conditions matches.

If an ATOM channel is disabled by the register **ATOM[i]_AGC_OUTEN_STAT**, the actual value of the channel output at *ATOM_CH[x]_OUT* is defined by the signal level bit (**SL**) defined in the channel control register **ATOM[i]_CH[x]_CTRL**.

If the output is enabled, the output at *ATOM_CH[x]_OUT* depends on value of Flip-flop **SOUR**.

The register **ATOM[i]_AGC_FUPD_CTRL** defines which of the ATOM channels receive a *FORCE_UPDATE* event if the trigger signal *CTRL_TRIG* is raised.
Note: In SOMP mode the force update request is stored and executed synchronized to the selected CMU_CLK. In all other modes the force update request is executed immediately.

The register bits **UPEN_CTRL[x]** defines for which ATOM channel the update of the working register **CM0**, **CM1** and **CLK_SRC** by the corresponding shadow register **SR0**, **SR1** and **CLK_SRC_SR** is enabled. If update is enabled, the register **CM0**, **CM1** and **CLK_SRC** will be updated on reset of counter register **CN0** (see figure 13.2.1).

13.1.3 ATOM Channel Mode Overview

Each ATOM channel offers the following different operation modes:
In ATOM Signal Output Mode Immediate (SOMI), the ATOM channels generate an output signal immediately after receiving an ARU word according to the two signal level output bits of the ARU word received through the ACBI bit field. Due to the fact, that the ARU destination channels are served in a round robin order, the output signal can jitter in this mode with a jitter of the ARU round trip time.

In ATOM Signal Output Mode Compare (SOMC), the ATOM channel generates an output signal on behalf of time stamps that are located in the ATOM operation registers. These time stamps are compared with the time stamps, the TBU generates. The ATOM is able to receive new time stamps either by CPU or via the ARU. The new time stamps are directly loaded into the channels operation register. The shadow registers are used as capture registers for two time base values, when a compare match of the channels operation registers occurs.

In ATOM Signal Output Mode PWM (SOMP), the ATOM channel is able to generate simple and complex PWM output signals like the TOM sub-module by comparing its operation registers with a sub-module internal counter. In difference to the TOM, the ATOM shadow registers can be reloaded by the CPU and by the ARU in the background, while the channel operates on the operation registers.

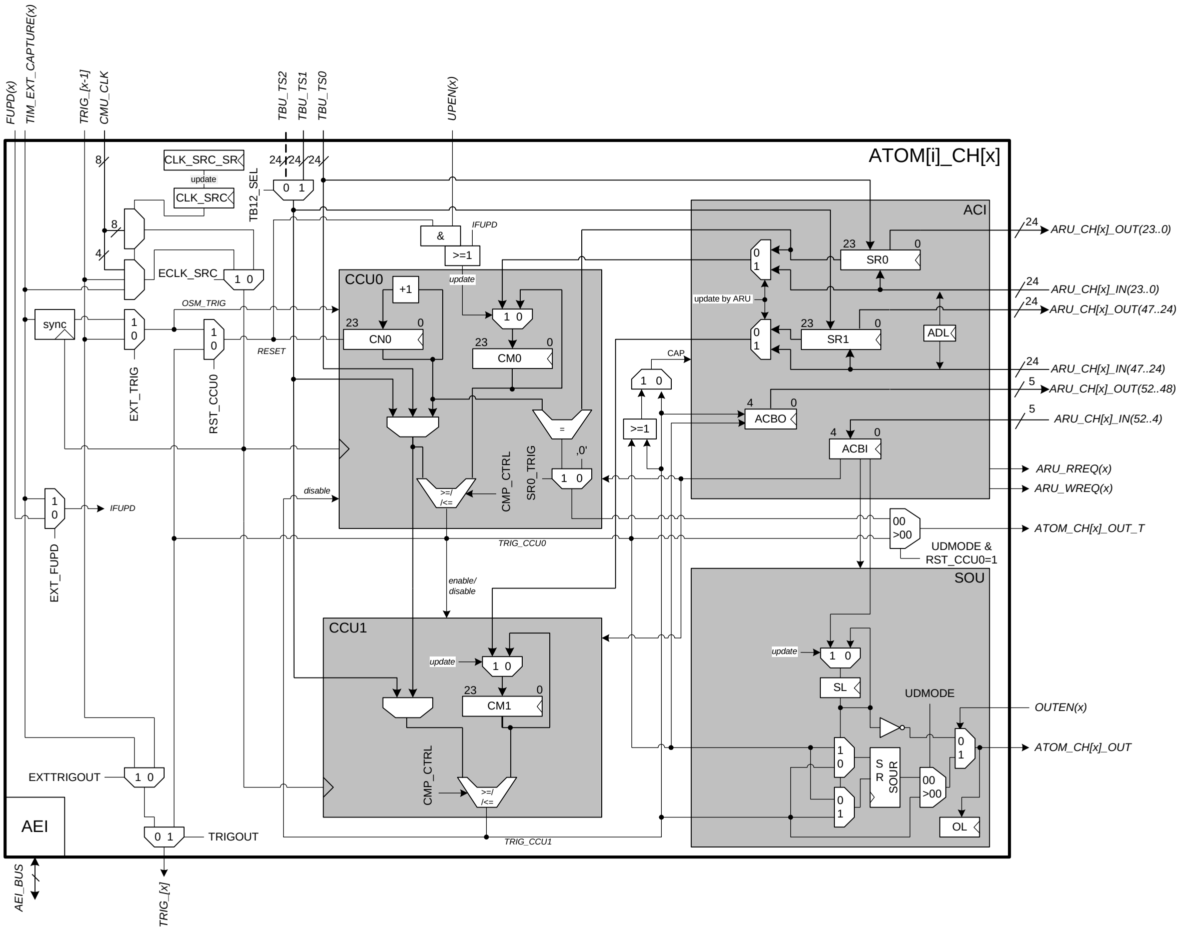
In ATOM Signal Output Mode Serial (SOMS), the ATOM channel generates a serial output bit stream on behalf of a shift register. The number of bits shifted and the shift direction is configurable. The shift frequency is determined by one of the *CMU_CLKx* clock signals. Please refer to section 13.3.4 for further details.

In ATOM Signal Output Buffered Compare (SOMB), the ATOM channel generates an output signal on behalf of time stamps that located in the ATOM operation registers. These time stamps are compared with the time stamps, the TBU generates. The ATOM is able to receive new compare values either by CPU or via the ARU. The new compare values received via ARU are stored first in the shadow register and only if previous compare match is occurred, the operation register are updated with the content of the shadow register.

13.2 ATOM Channel Architecture

Each ATOM channel is able to generate output signals according to five operation modes. The architecture of the ATOM channels is similar to the architecture of the TOM channels. The general architecture of an ATOM channel is depicted in figure 13.2.1.

13.2.1 ATOM channel architecture



In all ATOM channels the operation registers **CN0**, **CM0** and **CM1** and the shadow registers **SR0** and **SR1** are the 24 bit width. The comparators inside CCU0 and CCU1 provide a selectable signed greater-equal or less-equal comparison to compare against the GTM time bases *TBU_TS0*, *TBU_TS1* and, if available, *TBU_TS2* . Please refer to TBU chapter 10 for further details. The CCU0 and CCU1 units have different tasks for the different ATOM channel modes.

The cyclic event compare is used to detect time base overflows and to guarantee, that a compare match event can be set up for the future even when the time base will first overflow and then reach the compare value. Please note, that for a correct behavior of this cyclic event compare, the new compare value must not be specified larger/smaller than half of the range of the total time base value (0x7FFFFF).

In SOMC/SOMB mode, the two compare units CCUx can be used in combination to each other. When used in combination, the trigger lines *TRIG_CCU0* and *TRIG_CCU1* can be used to enable/disable the other compare unit on a match event. Please refer to section 13.3.2 and 13.3.5 for further details.

The Signal Output Unit (SOU) generates the output signal for each ATOM channel. This output signal level depends on the ATOM channel mode and on the **SL** bit of the **ATOM[i]_CH[x]_CTRL** register in combination with the two control bits. These two control bits **ACB(1)** and **ACB(0)** can either be received via CPU in the ACB register field of the **ATOM[i]_CH[x]_CTRL** register or via ARU in the ACBI bit field of the **ATOM[i]_CH[x]_STAT** register.

The **SL** bit in the **ATOM[i]_CH[x]_CTRL** register defines in all modes the operational behavior of the ATOM channel.

When the channel and its output are disabled, the output signal level of the channel is the inverse of the **SL** bit.

In SOMI, SOMC and SOMB mode the output signal level depends on the **SL**, **ACB0** and **ACB1** bits. In SOMP mode the output signal level depends on the two trigger signals *TRIG_CCU0* and *TRIG_CCU1* since theses two triggers define the PWM timing characteristics and the **SL** bit defines the level of the duty cycle. In SOMS mode the output signal level is defined by the bit pattern that has to be shifted out by the ATOM channel. The bit pattern is located inside the **CM1** register.

The ARU Communication Interface (ACI) sub-unit is responsible for requesting data routed through ARU to the ATOM channel in SOMI, SOMP, SOMB and SOMS modes, and additionally for providing data to the ARU in SOMC mode.

In SOMC mode the ACI shadow registers have a different behavior and are used as output buffer registers for data send to ARU.

13.2.2 ARU Communication Interface

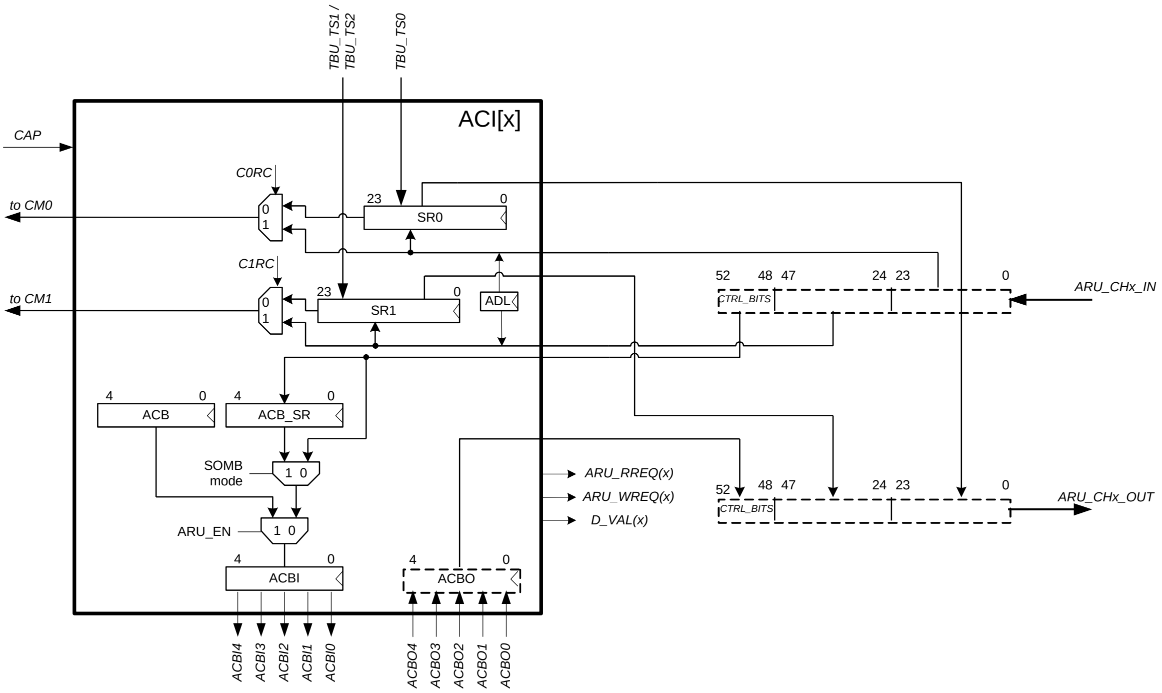
The ATOM channels have an ARU Communication Interface (ACI) sub-unit. This sub-unit is responsible for data exchange from and to the ARU. This is done with the two implemented registers **SR0**, **SR1**, and the **ACBI** and **ACBO** bit fields that are part of the **ATOM[i]_CH[x]_STAT** register. The ACI architecture is shown in figure 13.2.2.1.

If the **ARU_EN** bit is set inside the **ATOM[i]_CH[x]_CTRL** register, the ATOM channel is enabled by setting the enable bits inside the **ATOM[i]_AGC_ENDIS_STAT** register and the CPU hasn't written data not equal to zero into the **CM0**, **CM1, SR0, SR1** register, the ATOM channel will first request data from the ARU before the signal generation starts in SOMP, SOMS, SOMC and SOMB mode.

Note: if in SOMP mode there is data inside the **CM0** or **SR0** register not equal to 0 the channel counter **CN0** will start counting immediately, regardless whether the channel has received ARU data yet.

Note: if in SOMS mode there is data inside the **CM0** or **SR0** register not equal to 0 the channel will start shifting immediately, regardless whether the channel has received ARU data yet.

13.2.2.1 ACI architecture overview



Incoming ARU data (53 bit width signal *ARU_CHx_IN*) is split into three parts by the ACI and communicated to the ATOM channel registers. In SOMI, SOMP, SOMS and SOMB modes incoming ARU data *ARU_CHx_IN* is split in a way that the lower 24 bits of the ARU data (23 downto 0) are stored in the **SR0** register, the upper bits (47 downto 24) are stored in the **SR1** register. The bits 52 downto 48 (*CTRL_BITS*) are stored in SOMI, SOMP and SOMS mode in the **ACBI** bit field of the register **ATOM[i]_CH[x]_STAT**, in SOMB mode in the internal ACB_SR register.

The ATOM channel has to ensure, that in a case when the channel operation registers **CM0** and **CM1** are updated with the **SR0** and **SR1** register content and an ARU transfer to these shadow registers happens in parallel that either the old data in both shadow registers is transferred into the operation registers or both new values from the ARU are transferred.

In SOMC mode incoming ARU data *ARU_CHx_IN* is written directly to the ATOM channel operation register in the way that the lower 24 bits (23 down to 0) are written to **CM0**, and the bits 47 down to 24 are written to register **CM1**. The bits 52 down to 48 are stored in the **ACBI** bit field of the **ATOM[i]_CH[x]_STAT** register and control the behavior of the compare units and the output signal of the ATOM channel.

In SOMC mode the **SR0** and **SR1** registers serve as capture registers for the time stamps coming from TBU whenever a compare match event is signaled by the CCU0 and/or CCU1 sub-units via the *CAP* signal line. These two time stamps are then provided together with actual ATOM channel status information located in the **ACBO** bit field to the ARU at the dedicated ARU write address of the ATOM channel when the ARU is enabled.

The encoding of the ARU control bits in the different ATOM operation modes is described in more detail in the following chapters.

13.3 ATOM Channel Modes

As described above, each ATOM channel can operate independently from each other in one of five dedicated output modes:

- ATOM Signal Output Mode Immediate (SOMI)
- ATOM Signal Output Mode Compare (SOMC)
- ATOM Signal Output Mode PWM (SOMP)
- ATOM Signal Output Mode Serial (SOMS)
- ATOM Signal Output Mode Buffered Compare (SOMB)

The Signal Output Mode PWM (SOMP) is principally the same like the output mode for the TOM sub-module. In addition, it is possible to reload the shadow registers via the ARU without the need of a CPU interaction. The other modes provide additional functionality for signal output control. All operation modes are described in more detail in the following sections.

Note that in any output mode, if a channel is enabled, one-shot mode is disabled (**OSM**=0; only used in modes SOMP and SOMS) and **CM0** >= **CN0**, the counter **CN0** is incrementing until it reaches **CM0**.

To avoid unintended counting of **CN0** after enabling a channel, it is recommended to reset a channel (or at least **CN0** and **CM0**) before any change on the mode bits **MODE**, **ARU_EN** and **OSM**.

13.3.1 ATOM Signal Output Mode Immediate (SOMI)

In ATOM Signal Output Mode Immediate (SOMI), the ATOM channel generates output signals on the *ATOM[i]_CH[x]_OUT* output port immediate after update of the bit **ACBI(0)** of register **ATOM[i]_CH[x]_STAT** or **ACB(0)** bit of register **ATOM[i]_CH[x]_CTRL**.

If ARU access is enabled by setting bit ARU_EN in register **ATOM[i]_CH[x]_CTRL**, the update of the output *ATOM[i]_CH[x]_OUT* depends on the bit ACBI(0) of register **ATOM[i]_CH[x]_STAT** received at the ACI sub-unit and the bit **SL** bit of register **ATOM[i]_CH[x]_CTRL**. The remaining 48 ARU bits (47 downto 0) have no meaning in this mode.

If ARU access is disabled, the update of the output *ATOM[i]_CH[x]_OUT* depends on the bit **ACB(0)** and the bit **SL** of register **ATOM[i]_CH[x]_CTRL**.

The initial ATOM channel port pin *ATOM[i]_CH[x]_OUT* signal level has to be specified by the **SL** bit field of the **ATOM[i]_CH[x]_CTRL** register when **OUTEN_CTRL** register bit field **OUTEN_CTRLx** is disabled (see section 13.6.5) for details.

In SOMI mode the output behavior depends on the **SL** bit of register **ATOM[i]_CH[x]_CTRL** and the bit **ACBI(0)** of the **ATOM[i]_CH[x]_STAT** register or the bit **ACB0** of register **ATOM[i]_CH[x]_CTRL**:

13.3.1.1 Output behavior in SOMI mode

SL	**ACBI(0)/ ACB(0)**	**Output behavior**
0	0	Set output to inverse of SL (1)
0	1	Set output to SL (0)
1	0	Set output to inverse of SL (0)
1	1	Set output to SL (1)

The signal level bit **ACBI(0)** is transferred to the SOU sub-unit of the ATOM and made visible at the output port according to the table above immediately after the data was received by the ACI. This can introduce a jitter on the output signal since the ARU channels are served in a time multiplexed fashion.

13.3.1.2 Register ATOM[i]_CH[x]_CTRL in SOMI mode

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_0x00**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	FREEZE	Not used	Not used	Reserved	Not used	Not used	Not used	Not used	Not used			Not used	Not used		Not used	Not used	Not used	Not used			SL	Not used	Not used	Not used				ACB(0)	ARU_EN	Not used	MODE	
Mode	RW	RW	RW	R	RW	RW	R	RW	RW			RW	RW		RW	RW	RW	RW			RW	RW	RW	RW				RW	RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	0b000			0	0b00		0	0	0	0b000			x	0	0	0b0000				0	0	0	0b00	

Bit 1:0 **MODE**: ATOM channel mode select.
0b00 = ATOM Signal Output Mode Immediate (SOMI)

Bit 2 **Not used**
Note: Not used in this mode.

Bit 3 **ARU_EN**: ARU Input stream enable
0 = ARU Input stream disabled
1 = ARU Input stream enabled

Bit 4 **ACB(0)**: ACB bit 0
0 = Set output to inverse of SL bit
1 = Set output to SL bit

Bit 8:5 **Not used**
Note: Not used in this mode.

Bit 9 **Not used**
Note: Not used in this mode.

Bit 10 **Not used**
Note: Not used in this mode.

Bit 11 **SL**: Initial signal level after channel is enabled
0 = Low signal level
1 = High signal level
Note: Reset value depends on the hardware configuration chosen by silicon vendor.
Note: If the output is disabled, the output ATOM_OUT[x] is set to inverse value of SL.
If FREEZE=0, following note is valid:
Note: If the channel is disabled, the output register of SOU unit is set to inverse value of SL.
If FREEZE=1, following note is valid:
Note: If the channel is disabled, the output register of SOU unit is not changed and output ATOM_OUT[x] is not changed.

Bit 14:12 **Not used**
Note: Not used in this mode.

Bit 15	**Not used** Note: Not used in this mode.
Bit 16	**Not used** Note: Not used in this mode.
Bit 17	**Not used** Note: Not used in this mode.
Bit 19:18	**Not used** Note: Not used in this mode.
Bit 20	**Not used** Note: Not used in this mode.
Bit 23:21	**Not used** Note: Not used in this mode.
Bit 24	**Not used** Note: Not used in this mode.
Bit 25	**Not used** Note: Not used in this mode.
Bit 26	**Not used** Note: Not used in this mode.
Bit 27	**Not used** Note: Not used in this mode.
Bit 28	**Reserved** Note: Read as zero, should be written as zero.
Bit 29	**Not used** : not used in this mode Note: Not used in this mode.
Bit 30	**Not used** Note: Not used in this mode.
Bit 31	**FREEZE** 0 = a channel disable/enable may change internal register and output register 1 = a channel enable/disable does not change an internal or output register but stops counter CN0 (in SOMP mode), comparison (in SOMC/SOMB mode) and shifting (in SOMS mode)

13.3.2 ATOM Signal Output Mode Compare (SOMC)

13.3.2.1 Overview

In ATOM Signal Output Mode Compare (SOMC) the output action is performed in dependence of the comparison between input values located in **CM0** and/or **CM1** registers and the two (three) time base values *TBU_TS0* or *TBU_TS1* (or *TBU_TS2*) provided by the TBU. For a description of the time base generation please refer to the TBU specification in chapter 10. It is configurable, which of the two (three) time bases is to be compared with one or both values in **CM0** and **CM1**.

The behavior of the two compare units CCU0 and CCU1 is controlled either with the bits 4 downto 2 of **ACB** bit field inside the **ATOM[i]_CH[x]_CTRL** register, when the ARU connection is disabled or with the ACBI bit field of the **ATOM[i]_CH[x]_STAT** register, when the ARU is enabled. In that case the **ACB** bit field is updated via the ARU control bits 52 downto 48.

The CCUx trigger signals *TRIG_CCU0* and *TRIG_CCU1* always create edges, dependent on the predefined signal level in **SL** bit in combination with two control bits that can be specified by either ARU or CPU within the aforementioned **ATOM[i]_CH[x]_CTRL** or **ATOM[i]_CH[x]_STAT** registers.

In SOMC mode the channel is always disabled after the specified compare match event occurred. The shadow registers are used to store two time stamp values at the match time. The channel compare can be re-enabled by first reading the shadow registers, either by CPU or ARU and by providing new data for CMx registers through CPU or ARU. For a detailed description please refer to the sections 13.3.2.2 and 13.3.2.3.

If three time bases exist for the GTM-IP there must be a preselection between *TBU_TS1* and *TBU_TS2* for the ATOM channel. This can be done with **TB12_SEL** bit in the **ATOM[i]_CH[x]_CTRL** register.

The comparison in CCU0/1 with time base TBU_TS1 or TBU_TS2 can be done on a greater-equal or less-equal compare according to the **CMP_CTRL** bit. This control bit has no effect to a compare unit CCU0 or CCU1 that compares against TBU_TS0. In this case always a greater-equal compare is done. The bit **CMP_CTRL** is part of the **ATOM[i]_CH[x]_CTRL** register.

When configured in SOMC mode, the channel port pin has to be initialized to an initial signal level. This initial level after enabling the ATOM channel is determined by the **SL** bit in the **ATOM[i]_CH[x]_CTRL** register. If the output is disabled, the signal level is set to the inverse level of the **SL** bit.

If the channel is disabled, the register SOUR is set to the **SL** bit in the **ATOM[i]_CH[x]_CTRL** register.
On a compare match event the shadow register **SR0** and **SR1** are used to capture the TBU time stamp values. **SR0** always holds *TBU_TS0* and **SR1** either holds *TBU_TS1* or *TBU_TS2* dependent on the **TB12_SEL** bit in the **ATOM[i]_CH[x]_CTRL** register.

Please note, that when the channel is disabled and the compare registers are written, the compare registers CMx are loaded with the written value and the channel starts with the comparison on behalf of this values, when the channel is enabled.

13.3.2.2 SOMC Mode under CPU control

As already mentioned above the ATOM channel can be controlled either by CPU or by ARU. When the channel should be controlled by CPU, the **ARU_EN** bit inside the **ATOM[i]_CH[x]_CTRL** register has to be reset.

The output of the ATOM channel is set on a compare match event depending on the **ACB10** bit field in combination with the **SL** bit both located in the **ATOM[i]_CH[x]_CTRL** register. The output behavior according to the **ACB10** bit field in the control register is shown in the following table:

13.3.2.2.1 Output behavior according to the **ACB10** bit field in the control register

SL	**ACB10(5)**	**ACB10(4)**	**Output behavior**
0	0	0	No signal level change at output (exception in table 13.3.2.2.4 mode ACB42=001)
0	0	1	Set output signal level to 1
0	1	0	Set output signal level to 0
0	1	1	Toggle output signal level (exception in table 13.3.2.2.4 mode ACB42=001)
1	0	0	No signal level change at output (exception in table 13.3.2.2.4 mode ACB42=001)
1	0	1	Set output signal level to 0
1	1	0	Set output signal level to 1
1	1	1	Toggle output signal level (exception in table 13.3.2.2.4 mode ACB42=001)

The capture/compare strategy of the two CCUx units can be controlled with the **ACB42** bit field inside the **ATOM[i]_CH[x]_CTRL** register. The meaning of these bits is shown in the following table:

13.3.2.2.2 Capture/compare strategy of the two CCUx units controlled by **ACB42** bit field

ACB42(8)	**ACB42(7)**	**ACB42(6)**	**CCUx control**
0	0	0	Serve First: Compare in CCU0 using *TBU_TS0* and in parallel in CCU1 using *TBU_TS1* or *TBU_TS2*. Disable other CCUx on compare match. Output signal level on the compare match of the matching CCUx unit is defined by combination of SL, ACB10(5) and ACB10(4). Details see table 13.3.2.2.4

0	0	1	Serve First: Compare in CCU0 using *TBU_TS0* and in parallel in CCU1 using *TBU_TS1* or *TBU_TS2*. Disable other CCUx on compare match. Output signal level on the compare match of the matching CCUx unit is defined by combination of SL, ACB10(5) and ACB10(4). Details see table 13.3.2.2.4
0	1	0	Compare in CCU0 only, use time base *TBU_TS0*. Output signal level is defined by combination of SL, ACB10(5) and ACB10(4) bits.
0	1	1	Compare in CCU1 only, use time base *TBU_TS1* or *TBU_TS2*. Output signal level is defined by combination of SL, ACB10(5) and ACB10(4) bits.
1	0	0	Serve Last: Compare in CCU0 and then in CCU1 using *TBU_TS0*. Output signal level when CCU0 matches is defined by combination of SL, ACB10(5) and ACB10(4). On the CCU1 match the output level is toggled.
1	0	1	Serve Last: Compare in CCU0 and then in CCU1 using *TBU_TS1* or *TBU_TS2*. Output signal level when CCU0 matches is defined by combination of SL, ACB10(5) and ACB10(4). On the CCU1 match the output level is toggled.
1	1	0	Serve Last: Compare in CCU0 using *TBU_TS0* and then in CCU1 using *TBU_TS1* or *TBU_TS2*. Output signal level when CCU1 matches is defined by combination of SL, ACB10(5) and ACB10(4).
1	1	1	Cancels pending comparison independent on ARU_EN.

The behavior of the **ACBI/ACB42** bit combinations 0b000 and 0b001 is described in more detail in tables 13.3.2.2.3 and 13.3.2.2.4.

13.3.2.2.3 ATOM CCUx Serve first definition ACB42 = 0b000

ACB4	ACB3	ACB2		ACB1	ACB0		SL		CCU0 match	CCU1 match		Pin level new
0	0	0		0	0		0		0	1		hold
									1	0		hold
									1	1		hold
0	0	0		0	1		0		0	1		1
									1	0		1
									1	1		1
0	0	0		1	0		0		0	1		0
									1	0		0
									1	1		0
0	0	0		1	1		0		0	1		toggle
									1	0		toggle
									1	1		toggle
0	0	0		0	0		1		0	1		hold
									1	0		hold
									1	1		hold
0	0	0		0	1		1		0	1		0
									1	0		0
									1	1		0
0	0	0		1	0		1		0	1		1
									1	0		1
									1	1		1
0	0	0		1	1		1		0	1		toggle
									1	0		toggle
									1	1		toggle

13.3.2.2.4 ATOM CCUx Serve first definition ACB42 = 0b001

ACB4	ACB3	ACB2	ACB1	ACB0	SL	CCU0 match	CCU1 match	Pin level new
0	0	1	0	0	0	0	1	hold
						1	0	toggle
						1	1	hold
0	0	1	0	1	0	0	1	0
						1	0	1
						1	1	0
0	0	1	1	0	0	0	1	1
						1	0	0
						1	1	1
0	0	1	1	1	0	0	1	toggle
						1	0	hold
						1	1	toggle
0	0	1	0	0	1	0	1	hold
						1	0	toggle
						1	1	hold
0	0	1	0	1	1	0	1	1
						1	0	0
						1	1	1
0	0	1	1	0	1	0	1	0
						1	0	1
						1	1	0
0	0	1	1	1	1	0	1	toggle
						1	0	hold
						1	1	toggle

If the ATOM channel is enabled, the **CM0** and/or **CM1** registers and the **ACB42** bit field of the **ATOM[i]_CH[x]_CTRL** register can be updated by the CPU as long as the first match event occurs in case of a 'serve last' compare strategy or as long as the overall match event in case of the other compare strategies.

After a compare match event that causes an update of the shadow registers **SR0/SR1** and before reading the **SR0** and/or **SR1** register via ARU, the update of the registers **CM0** and/or **CM1** is possible but has no effect.

To set up a new compare action, first the **SR0** and/or **SR1** register containing captured values have to be read and then new compare values have to be written into the register **CM0** and/or **CM1**.

Which **CMx** register has to be updated depends on the compare strategy defined in the **ACB42** bit field of the channel control register. Since the channel immediately starts with the comparison after the **CMx** register was/were written, the compare strategy has to be updated before the **CMx** registers are written.

For the 'serve last' compare strategies, if the register **CM0** and **CM1** are updated, it can happen that one or both compare values are already located in the past. In any way the ATOM channel will first wait until both compare values are written before it starts the time base comparisons to avoid a deadlock.

The CPU can check at any time if at least one of the ATOM channels' capture compare register contains valid data and waits for a compare event to happen. This is signaled by the **DV** bit inside the **ATOM[i]_CH[x]_STAT** register.
Note, for 'serve last' compare strategies, if **DV** bit is currently not set, writing to **CM0** or **CM1** sets immediately the **DV** bit although the compare is only started if both values are written.

An exception for update of register **CM0/CM1** exists in SOMC mode and CCUx control mode 'serve last'. If in this mode the CCU0 compare match event occurred, the update of register **CM0/CM1** via CPU is blocked until the CCU1 compare match event.

In the 'serve last' mode (ACB42= 0b100 or ACB42=0b101) it is possible to generate very small spikes on the output pin by loading **CM0** and **CM1** with two time stamp values for *TBU_TS0*, *TBU_TS1* or *TBU_TS2* close together. The output pin will then be set or reset dependent on the **SL** bit and the specified **ACB10(5)** and **ACB10(4)** bits in the **ACB10** bit field of the **ATOM[i]_CH[x]_CTRL** register on the first match event and the output will toggle on the second compare event in the CCU1 compare unit.

It is important to note, that the bigger (smaller) time stamp has to be loaded into the **CM1** register, since the CCU0 will enable the CCU1 once it has reached its comparison time stamp. The order of the comparison time stamps depends on the defined greater-equal or less-equal comparison of the CCUx units.

In addition to storing the captured time stamps in the shadow registers, the ATOM channel provides the result of the compare match event in the **ACBO(4)** and **ACBO(3)** bits of the **ATOM[i]_CH[x]_STAT** register. The meaning of the bits is shown in the following table:

13.3.2.2.5 Compare match event **ACBO(4)** and **ACBO(3)** bits of **ATOM[i]_CH[x]_STAT**

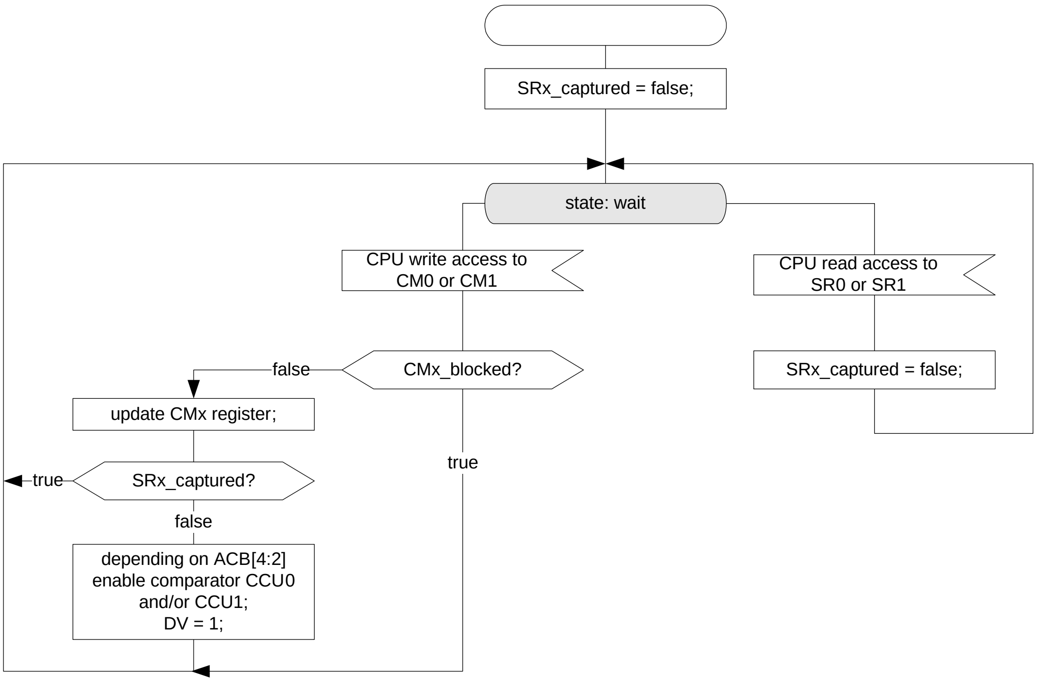
ACBO(4)	**ACBO(3)**	**Indication**
0	1	CCU0 compare match occurred
1	0	CCU1 compare match occurred

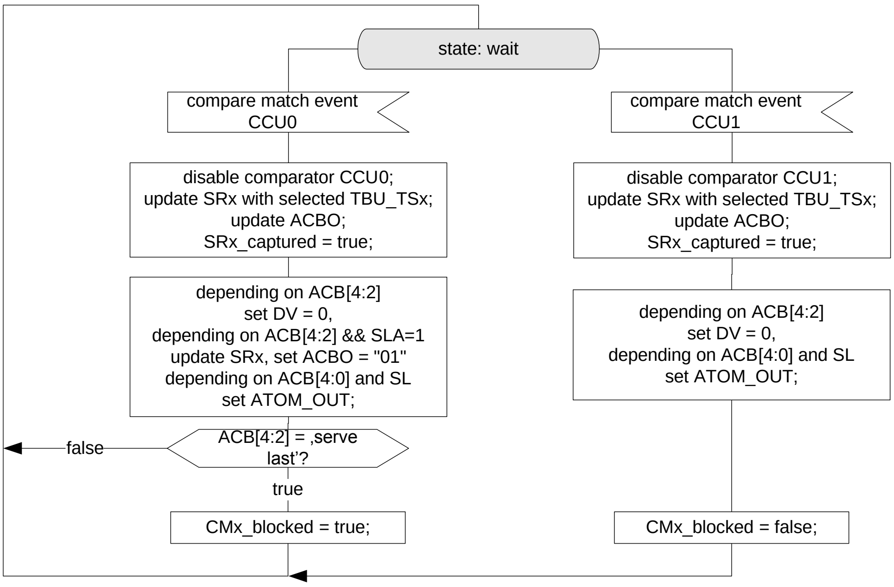
Please note, that in case of the 'serve last' compare strategy, when the bit **SLA** in the **ATOM[i]_CH[x]_CTRL** register is not set, the **ACBO(4)** bit is always set and the **ACBO(3)** bit is always reset after the compare match event occurred.

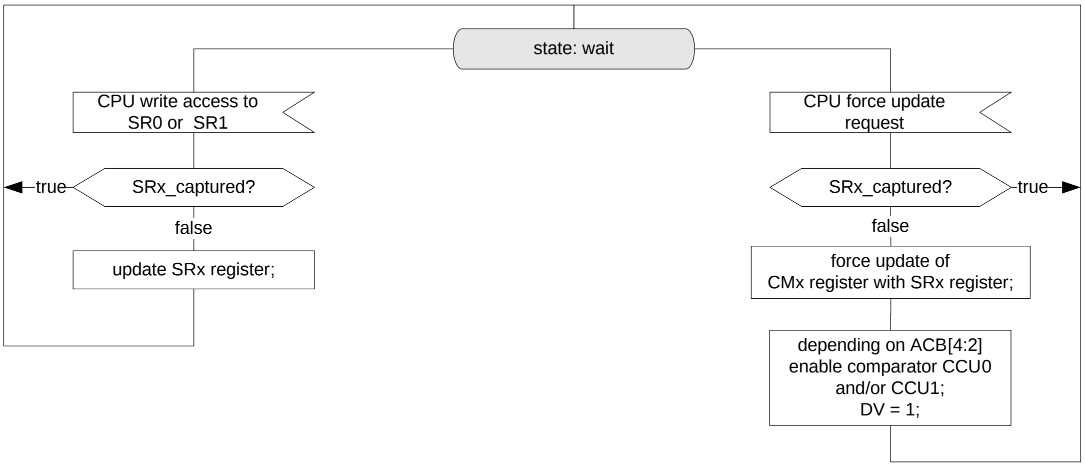
The **ACBO** bit field is reset, when the **DV** bit is set.
Depending on the capture compare unit where the time base matched the interrupt *CCU0TCx_IRQ* or *CCU1TCx_IRQ* is raised.
Note that in case of 'serve first' compare strategy, if both events CCU0 and CCU1 occur at the same point in time, both interrupts will be raised.

The behavior of an ATOM channel in SOMC mode under CPU control is depicted in figure 13.3.2.2.6.

13.3.2.2.6 SOMC state diagram for channel under CPU control


SRx_captured = false;
state: wait
CPU write access to CM0 or CM1
CPU read access to SR0 or SR1
CMx_blocked?
false
true
SRx_captured = false;
update CMx register;
SRx_captured?
true
false
depending on ACB[4:2] enable comparator CCU0 and/or CCU1; DV = 1;


state: wait
compare match event CCU0
compare match event CCU1
disable comparator CCU0; update SRx with selected TBU_TSx; update ACBO; SRx_captured = true;
disable comparator CCU1; update SRx with selected TBU_TSx; update ACBO; SRx_captured = true;
depending on ACB[4:2] set DV = 0, depending on ACB[4:2] && SLA=1 update SRx, set ACBO = "01" depending on ACB[4:0] and SL set ATOM_OUT;
depending on ACB[4:2] set DV = 0, depending on ACB[4:0] and SL set ATOM_OUT;
ACB[4:2] = ‚serve last'?
false
true
CMx_blocked = true;
CMx_blocked = false;


state: wait
CPU write access to SR0 or SR1
CPU force update request
SRx_captured?
true
false
update SRx register;
SRx_captured?
true
false
force update of CMx register with SRx register;
depending on ACB[4:2] enable comparator CCU0 and/or CCU1; DV = 1;

13.3.2.3 SOMC Mode under ARU control

When the channel should be controlled by ARU, the **ARU_EN** bit inside the **ATOM[i]_CH[x]_CTRL** register has to be set.

In case, the ATOM channel is under ARU control the content for the compare registers **CM0** and **CM1** as well as the update of the compare strategy can be loaded via the 53 bit ARU word.

The ARU word 23 to 0 is loaded into the **CM0** register while the ARU word 47 to 24 is loaded into the **CM1** register. The five ARU control bits 52 to 48 are loaded into the **ACBI** bit field of the **ATOM[i]_CH[x]_STAT** register and control the channel compare strategy as well as the output behavior in case of compare match events.

For the five ARU control bits 52 to 48 the bits 49 and 48 are loaded into the **ACBI** bits 1 and 0. The output behavior also depends on the setting of the **SL** bit inside of the **ATOM[i]_CH[x]_CTRL** register and is shown in the following table:

13.3.2.3.1 Output behavior depends on **SL** bit inside of the **ATOM[i]_CH[x]_CTRL** and **ACBI** bits 1 and 0

SL	**ACBI(1)**	**ACBI(0)**	**Output behavior**
0	0	0	No signal level change at output (exception in tables 13.3.2.2.3 and 13.3.2.2.4 mode ACB42=001)
0	0	1	Set output signal level to 1
0	1	0	Set output signal level to 0
0	1	1	Toggle output signal level (exception in table 13.3.2.2.3 and 13.3.2.2.4 mode ACB42=001)
1	0	0	No signal level change at output (exception in table's 13.3.2.2.3 and 13.3.2.2.4 mode ACB42=001)
1	0	1	Set output signal level to 0
1	1	0	Set output signal level to 1
1	1	1	Toggle output signal level (exception in table 13.3.2.2.3 and 13.3.2.2.4 mode ACB42=001)

For the five ARU control bits 52 to 48 the bits 52 to 50 are loaded into the **ACBI** bits 4 to 2. With these three bits the capture/compare units CCUx can be controlled as shown in the following table:

13.3.2.3.2 Capture/compare units CCUx controlled by **ACBI** bits 4 to 2

ACBI(4)	ACBI(3)	ACBI(2)	CCUx control
0	0	0	Serve First: Compare in CCU0 using *TBU_TS0* and in parallel in CCU1 using *TBU_TS1* or *TBU_TS2*. Disable other CCUx on compare match. Output signal level on the compare match of the matching CCUx unit is defined by combination of SL, ACBI(1) and ACBI(0). Details see table 13.3.2.2.4
0	0	1	Serve First: Compare in CCU0 using *TBU_TS0* and in parallel in CCU1 using *TBU_TS1* or *TBU_TS2*. Disable other CCUx on compare match. Output signal level on the compare match of the matching CCUx unit is defined by combination of SL, ACBI(1) and ACBI(0). Details see table 13.3.2.2.3
0	1	0	Compare in CCU0 only, use time base *TBU_TS0*. Output signal level is defined by combination of SL, ACBI(1) and ACBI(0) bits.
0	1	1	Compare in CCU1 only, use time base *TBU_TS1* or *TBU_TS2*. Output signal level is defined by combination of SL, ACBI(1) and ACBI(0) bits.
1	0	0	Serve Last: Compare in CCU0 and then in CCU1 using *TBU_TS0*. Output signal level when CCU0 matches is defined by combination of SL, ACBI(1) and ACBI(0). On the CCU1 match the output level is toggled.
1	0	1	Serve Last: Compare in CCU0 and then in CCU1 using *TBU_TS1* or *TBU_TS2*. Output signal level when CCU0 matches is defined by combination of SL, ACBI(1) and ACBI(0). On the CCU1 match the output level is toggled.
1	1	0	Serve Last: Compare in CCU0 using *TBU_TS0* and then in CCU1 using *TBU_TS1* or *TBU_TS2*. Output signal level when CCU1 matches is defined by combination of SL, ACBI(1) and ACBI(0).
1	1	1	Change ARU read address to ATOM_RDADDR1 DV flag is not set. Neither ACBI(1) nor ACBI(0) is evaluated.

It is important to note that the bit combination 0b111 for the **ACBI(4)**, **ACBI(3)** and **ACBI(2)** bits forces the channel to request new compare values from another destination read address defined in the **ATOM_RDADDR1** bit field of the **ATOM[i]_CH[x]_RDADDR** register. After data was successfully received and the compare event occurred the ATOM channel switches back to **ATOM_RDADDR0** to receive the next data from there.

After the specified compare match event, the captured time stamps are stored in **SR0** and **SR1** and the compare result is stored in the **ACBO** bit field of the **ATOM[i]_CH[x]_STAT** register. The meaning of the **ACBO(4)** and **ACBO(3)** bits of the **ATOM[i]_CH[x]_STAT** is shown in the following table:

13.3.2.3.3 Compare match event **ACBO(4)** and **ACBO(3)** bits of **ATOM[i]_CH[x]_STAT**

ACBO(4)	**ACBO(3)**	**Return value to ARU**
0	1	CCU0 compare match occurred
1	0	CCU1 compare match occurred

Please note, that in case of the 'serve last' compare strategy, when the bit **SLA** in the **ATOM[i]_CH[x]_CTRL** register is not set, the **ACBO(4)** bit is always set and the **ACBO(3)** bit is always reset after the compare match event occurred.

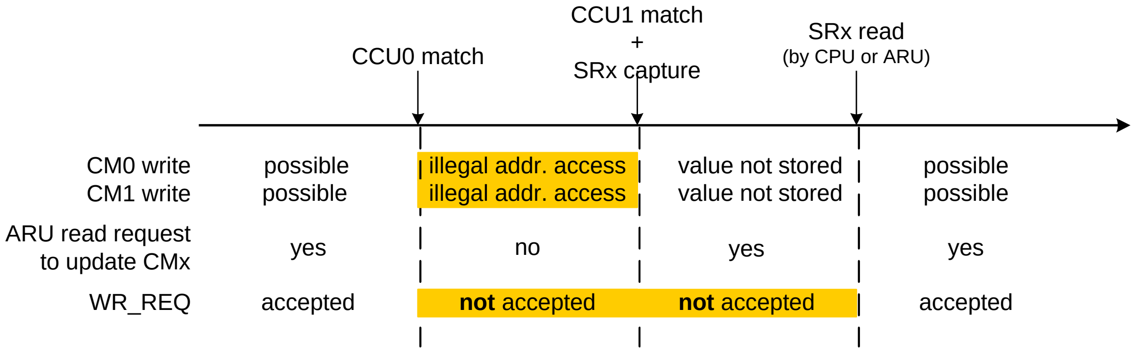
The **ACBO** bit field is reset, when the **DV** bit is set.
Depending on the capture compare unit where the time base matched the interrupt *CCU0TCx_IRQ* or *CCU1TCx_IRQ* is raised.

When CCU0 and CCU1 is used for comparison it is possible to generate very small spikes on the output pin by loading **CM0** and **CM1** with two time stamp values for *TBU_TS0*, *TBU_TS1* or *TBU_TS2* close together. The output pin will then be set or reset dependent on the **SL** bit and the specified **ACBI(0)** and **ACBI(1)** bits in the **ACBI** bit field of the **ATOM[i]_CH[x]_STAT** register on the first match event and the output will toggle on the second match event.

It is important to note, that the bigger (smaller) time stamp has to be loaded into the **CM1** register, since the CCU0 will enable the CCU1 once it has reached its comparison time stamp. The order of the comparison time stamps depends on the defined greater-equal or less-equal comparison of the CCUx units.

For compare strategy 'serve last' the CCU0 and CCU1 compare match may occur sequentially. During different phases of compare match the CPU access rights to register **CM0** and **CM1** as well as to **WR_REQ** bit is different.
For the case of bit **ABM**=0 and **EUPM**=0 (register **ATOM[i]_CH[x]_CTRL**) these access rights by CPU to register **CM0** and **CM1** and the **WR_REQ** are depicted in the following figure.

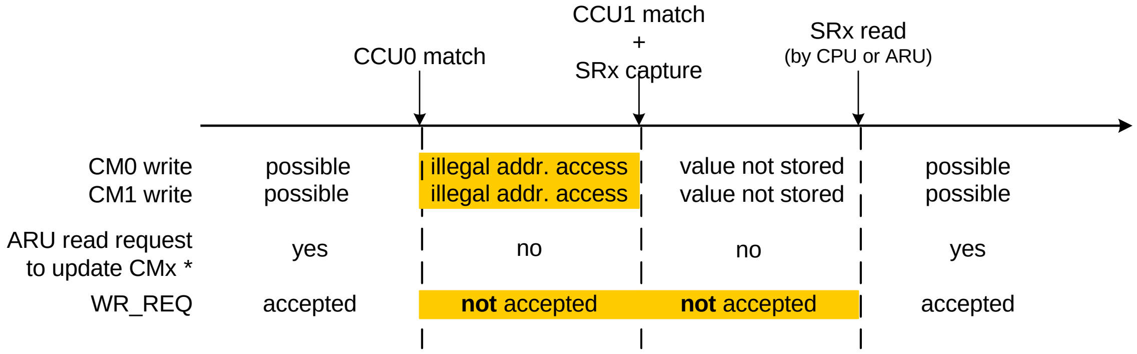
13.3.2.3.4 CPU access rights in case of compare strategy 'serve last', ABM=0 and EUPM=0



CM0 write	possible	illegal addr. access	value not stored	possible
CM1 write	possible	illegal addr. access	value not stored	possible
ARU read request to update CMx	yes	no	yes	yes
WR_REQ	accepted	**not accepted**	**not accepted**	accepted

For the case of bit **ABM**=1 and **EUPM**=0 (register **ATOM[i]_CH[x]_CTRL**) these access rights by CPU to register **CM0** and **CM1** and the **WR_REQ** are depicted in the following figure.

13.3.2.3.5 CPU access rights in case of compare strategy 'serve last', ABM=1 and EUPM=0

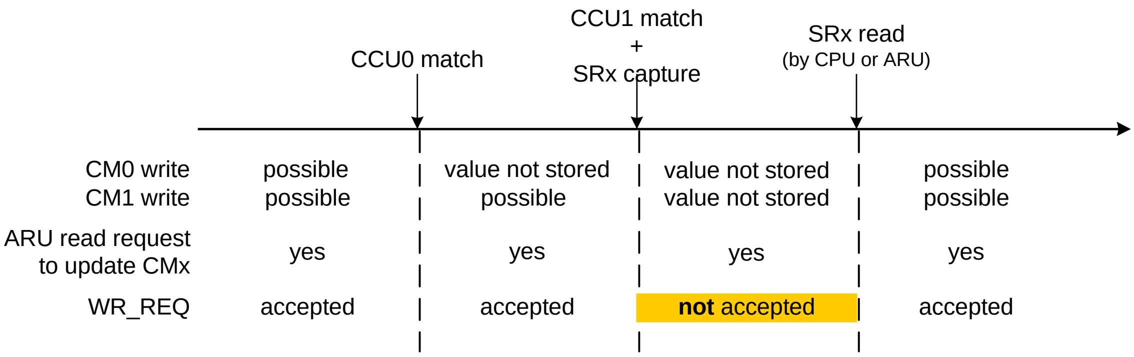


CM0 write	possible	illegal addr. access	value not stored	possible
CM1 write	possible	illegal addr. access	value not stored	possible
ARU read request to update CMx *	yes	no	no	yes
WR_REQ	accepted	**not accepted**	**not accepted**	accepted

* ARU read request until valid data is stored in CM0/CM1
(i.e. indicated by DV bit in register ATOM[i]_CH[x]_STAT set)

For the case of bit **EUPM**=1 (register **ATOM[i]_CH[x]_CTRL**), after CCU0 compare match (and before CCU1 compare match) an update of **CM1** as well as a late update via **WR_REQ** is possible. The value is used for compare. After CCU0 compare match an update of **CM0** is not possible, means the value is not stored.
The ARU read request is not paused between the compare matches.
This behavior is depicted in the following figures.

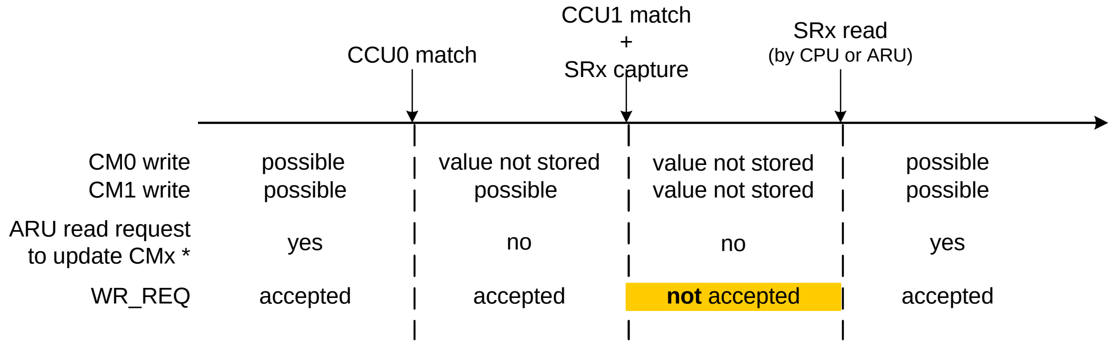
13.3.2.3.6 CPU access rights in case of compare strategy 'serve last', ABM=0 and EUPM=1



CM0 write CM1 write	possible possible	value not stored possible	value not stored value not stored	possible possible
ARU read request to update CMx	yes	yes	yes	yes
WR_REQ	accepted	accepted	**not accepted**	accepted

The behavior in case of EUPM=1 and ABM=1 is depicted in the following figure:

13.3.2.3.7 CPU access rights in case of compare strategy 'serve last', ABM=1 and EUPM=1



CM0 write CM1 write	possible possible	value not stored possible	value not stored value not stored	possible possible
ARU read request to update CMx *	yes	no	no	yes
WR_REQ	accepted	accepted	**not accepted**	accepted

* ARU read request until valid data is stored in CM0/CM1
(i.e. indicated by DV bit in register ATOM[i]_CH[x]_STAT set)

In case of EUPM=1 a write access to CM0 or CM1 never causes an AEI write status 0b10.

13.3.2.3.8 ARU Non-Blocking mode

When the compare registers are updated via ARU the update behavior of the channel is configurable with the ABM bit inside the **ATOM[i]_CH[x]_CTRL** register. When the **ABM** bit is reset, the ATOM channel is in ARU non-blocking mode.

In the ARU non-blocking mode, data received via ARU is continuously transferred to the registers **CM0** and **CM1** and the bit field **ACBI** of register **ATOM[i]_CH[x]_STAT** as long as no specified compare match event occurs.

After a compare match event that causes an update of the shadow register **SR0/SR1** and before reading the **SR0/SR1** register via CPU or ARU, the update of the registers **CM0/ CM1** via CPU or ARU is possible but the data is not accepted to be valid (no **DV** bit is set i register **ATOM[i]_CH[x]_CTRL**).

To set up a new compare action, first the **SR0/SR1** register containing captured values have to be read and then new compare values have to be written into the register **CM0/CM1**. This can be done either by ARU or by CPU.

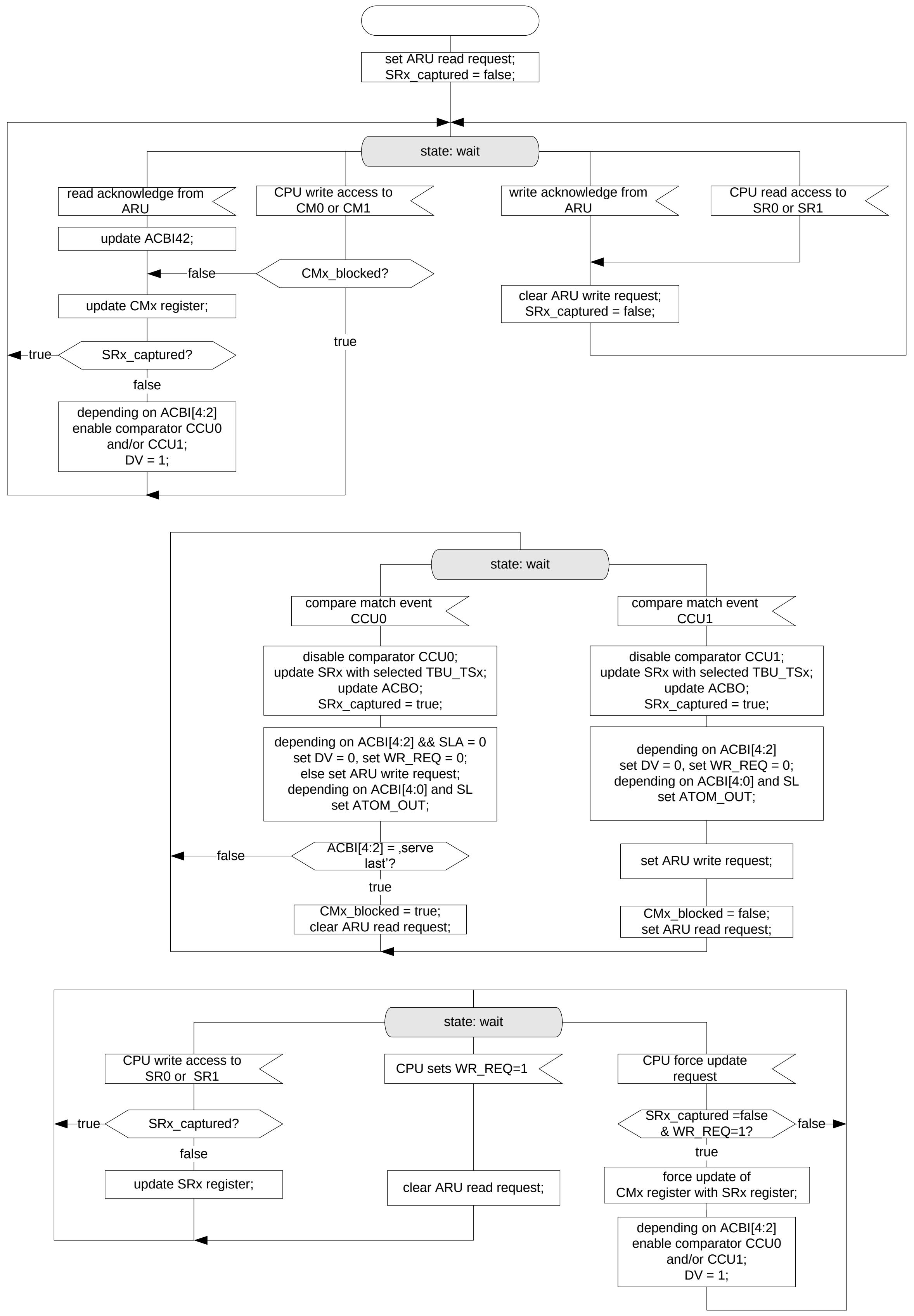
When the CPU does the register accesses, only one of the shadow registers has to be read. Dependent on the compare strategy, the CPU has to write one or both of the compare registers.

An exception for update of register **CM0/CM1** exists in SOMC mode and CCUx control mode 'serve last' if EUPM=0. If in this mode the CCU0 compare match event occurred, the update of register **CM0/CM1** via CPU is not possible until the CCU1 compare match event occurs.
Note that a write access to either **CM0** or **CM1** in this case leads to a write status 0b10.

The CPU can check at any time if the ATOM channel has received valid data from the ARU and waits for a compare event to happen. This is signaled by the **DV** bit inside the **ATOM[i]_CH[x]_STAT** register.

The behavior of an ATOM channel in SOMC mode, when ARU is enabled and ARU blocking mode is disabled is shown in figure 13.3.2.3.8.1.

13.3.2.3.8.1 SOMC State diagram for SOMC mode, ARU enabled, ABM disabled


set ARU read request;
SRx_captured = false;
state: wait
read acknowledge from ARU
CPU write access to CM0 or CM1
write acknowledge from ARU
CPU read access to SR0 or SR1
update ACBI42;
false
CMx_blocked?
update CMx register;
true
SRx_captured?
true
false
depending on ACBI[4:2] enable comparator CCU0 and/or CCU1; DV = 1;
clear ARU write request; SRx_captured = false;
state: wait
compare match event CCU0
compare match event CCU1
disable comparator CCU0; update SRx with selected TBU_TSx; update ACBO; SRx_captured = true;
disable comparator CCU1; update SRx with selected TBU_TSx; update ACBO; SRx_captured = true;
depending on ACBI[4:2] && SLA = 0 set DV = 0, set WR_REQ = 0; else set ARU write request; depending on ACBI[4:0] and SL set ATOM_OUT;
depending on ACBI[4:2] set DV = 0, set WR_REQ = 0; depending on ACBI[4:0] and SL set ATOM_OUT;
false
ACBI[4:2] = ‚serve last'?
true
set ARU write request;
CMx_blocked = true; clear ARU read request;
CMx_blocked = false; set ARU read request;
state: wait
CPU write access to SR0 or SR1
CPU sets WR_REQ=1
CPU force update request
true
SRx_captured?
SRx_captured =false & WR_REQ=1?
false
false
true
update SRx register;
clear ARU read request;
force update of CMx register with SRx register;
depending on ACBI[4:2] enable comparator CCU0 and/or CCU1; DV = 1;

13.3.2.3.9 ARU Blocking mode

When the compare registers are updated by ARU, the ATOM channel can be configured to receive ARU data in a blocking manner. This can be configured by setting the **ABM** bit in the **ATOM[i]_CH[x]_CTRL** register.

If the **ABM** and **ARU_EN** bits are set, depending on compare strategy, **CM0** and/or **CM1** can be updated via ARU with new compare values. If the compare registers **CM0** and/or **CM1** are accepting these new data to be valid (indicated by bit **DV** in register **ATOM[i]_CH[x]_STAT**), the ATOM channel stops requesting new data via ARU and waits for the compare match event to happen.

When the specified compare match event happens, the shadow registers **SR0** and **SR1** are updated together with the **ACBO** bits in the **ATOM[i]_CH[x]_STAT** register. The data in the shadow registers is marked as valid for the ARU and the **DV** bit or register **ATOM[i]_CH[x]_CTRL** is reset.

If the register **SR0** and **SR1** holding the captured TBU time stamp values are read by either the ARU or the CPU, the next write access to or update of the register **CM0** or **CM1** via ARU or the CPU enables the new compare match check again.

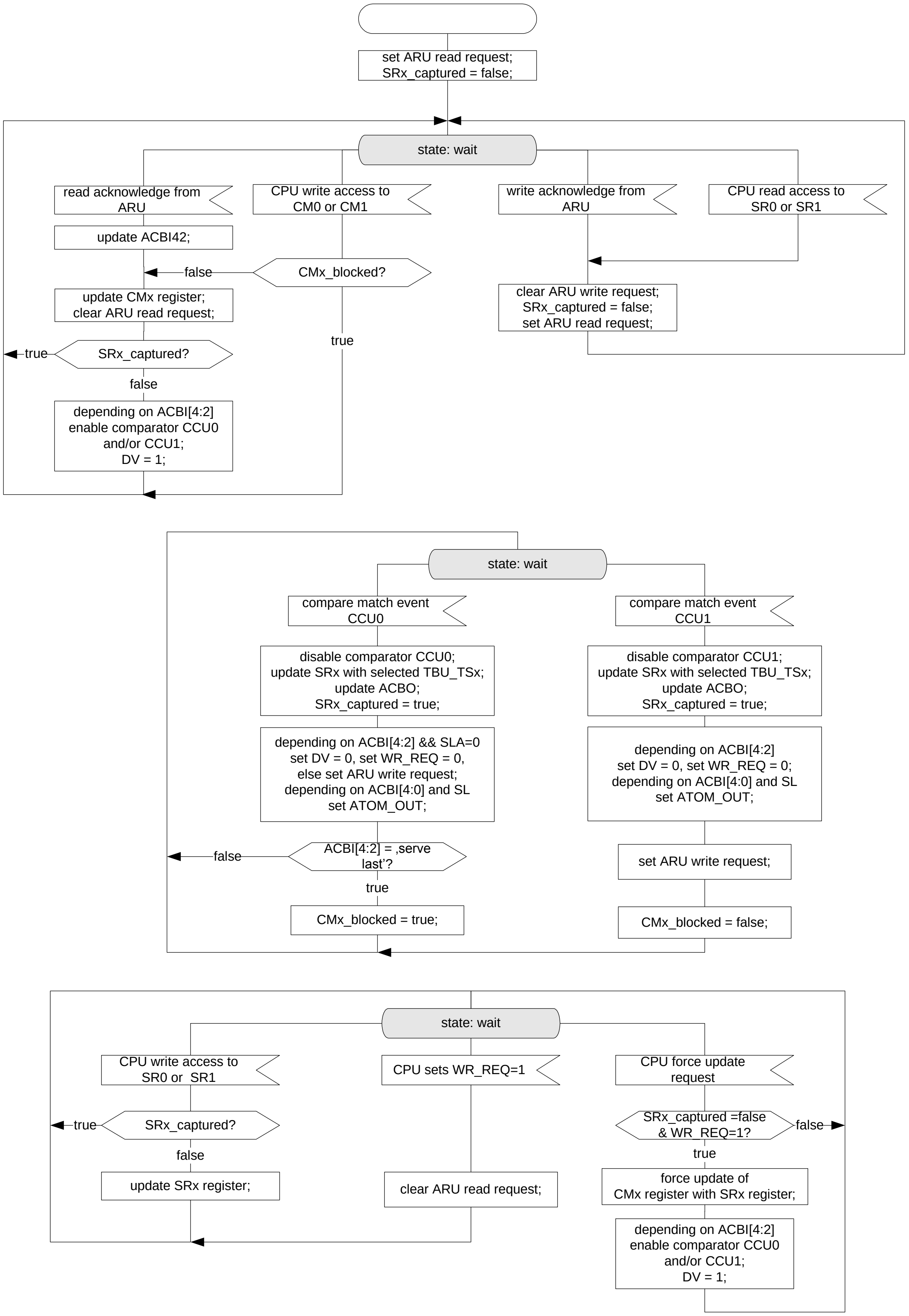
At least one of the registers **SR0** or **SR1** has to be read either via ARU or by CPU, before new data is requested via ARU.

Note that in case of ABM=1 the application has to handle the situation that the ATOM does not request update of new data for CM0/CM1 until the captured values are read. E.g. if an MCS task starts to write via ARU new data (with AWR(I) command) after capture of data in SR0/SR1, the task sticks in the command until captured data is read by another taks via ARU or via the CPU interface.

The CPU can check at any time if the ATOM channel has received valid data from the ARU and waits for a compare event to happen. This is signaled by a set **DV** bit inside the **ATOM[i]_CH[x]_STAT** register.

The behavior of an ATOM channel in SOMC mode, when ARU is enabled and ARU blocking mode is enabled is shown in figure 13.3.2.3.9.1.

13.3.2.3.9.1 SOMC State diagram for SOMC mode, ARU enabled and ABM enabled


set ARU read request;
SRx_captured = false;
state: wait
read acknowledge from ARU
CPU write access to CM0 or CM1
write acknowledge from ARU
CPU read access to SR0 or SR1
update ACBI42;
false
CMx_blocked?
update CMx register;
clear ARU read request;
clear ARU write request;
SRx_captured = false;
set ARU read request;
true
true
SRx_captured?
false
depending on ACBI[4:2] enable comparator CCU0 and/or CCU1; DV = 1;
state: wait
compare match event CCU0
compare match event CCU1
disable comparator CCU0;
update SRx with selected TBU_TSx;
update ACBO;
SRx_captured = true;
disable comparator CCU1;
update SRx with selected TBU_TSx;
update ACBO;
SRx_captured = true;
depending on ACBI[4:2] && SLA=0
set DV = 0, set WR_REQ = 0,
else set ARU write request;
depending on ACBI[4:0] and SL
set ATOM_OUT;
depending on ACBI[4:2]
set DV = 0, set WR_REQ = 0;
depending on ACBI[4:0] and SL
set ATOM_OUT;
false
ACBI[4:2] = ‚serve last’?
set ARU write request;
true
CMx_blocked = true;
CMx_blocked = false;
state: wait
CPU write access to SR0 or SR1
CPU sets WR_REQ=1
CPU force update request
true
SRx_captured?
SRx_captured =false & WR_REQ=1?
false
false
true
update SRx register;
clear ARU read request;
force update of CMx register with SRx register;
depending on ACBI[4:2] enable comparator CCU0 and/or CCU1; DV = 1;

13.3.2.3.10 ATOM SOMC Late update mechanism

Although, the ATOM channel may be controlled by data received via the ARU, the CPU is able to request at any time a late update of the compare register. This can be initiated by setting the **WR_REQ** bit inside the **ATOM[i]_CH[x]_CTRL** register. By doing this, the ATOM will request no further data from ARU (if ARU access was enabled). The channel will in any case continue to compare against the values stored inside the compare registers (if bit **DV** was set). The CPU can now update the new compare values until the compare event happens by writing to the shadow registers, and force the ATOM channel to update the compare registers by writing to the force update register bits in the **AGC** register.

If the **WR_REQ** bit is set and a compare match event happens, any further access to the shadow registers **SR0**, **SR1** is blocked and the force update of this channel is blocked. In addition, the **WRF** bit is set in the **ATOM[i]_CH[x]_STAT** register. Thus, the CPU can determine that the late update failed by reading the **WRF** bit.

In case of bit **EUPM**=0 (register **ATOM[i]_CH[x]_CTRL**) the following statements are true:
If a compare match event already happened, the **WR_REQ** bit could not be set until the channel is unlocked for a new compare match event by reading the shadow registers. In addition, the **WRF** bit is set if the CPU tries to write the **WR_REQ** bit in that case.

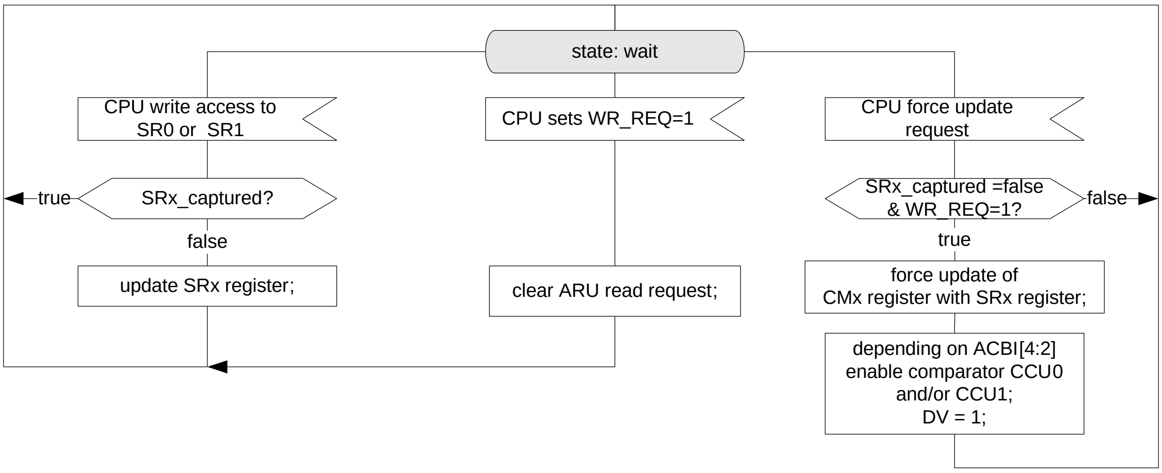
In case of bit **EUPM**=1 (register **ATOM[i]_CH[x]_CTRL**) the following statements are true:
If in case of 'serve last' strategy a CCU1 or in any other compare strategy a CCU0 or CCU1 compare match event already happened, the **WR_REQ** bit could not be set until the channel is unlocked for a new compare match event by reading the shadow registers. In addition, the **WRF** bit is set if the CPU tries to write the **WR_REQ** bit in that case.

In general, for a late update the following has to be taken into account:
If between a correct **WR_REQ** bit set, a correct shadow register write, and before the force update is requested by the AGC a match event occurs on the old compare values, the **WRF** bit will be set. The force update will be blocked.

The **WRF** bit will be set in any case if the CPU tries to write to a blocked shadow register.
The **WR_REQ** bit and the **DV** bit will be reset on a compare match event.
After a capture event for register **SR0** and/or **SR1** the force update mechanism will be blocked until a read access to the register **SR0** or **SR1** by either the ARU or the CPU happens.
Writing to **SR0** or **SR1** after compare match causes an AEI write status 0b10.

The ATOM SOMC late update mechanism from CPU is shown in figure 13.3.2.3.10.1.

13.3.2.3.10.1 SOMC State diagram for late update requests by CPU


state: wait
CPU write access to SR0 or SR1
CPU sets WR_REQ=1
CPU force update request
true
SRx_captured?
false
SRx_captured =false & WR_REQ=1?
false
true
update SRx register;
clear ARU read request;
force update of CMx register with SRx register;
depending on ACBI[4:2] enable comparator CCU0 and/or CCU1; DV = 1;

13.3.2.3.11 Register ATOM[i]_CH[x]_CTRL in SOMC mode

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_0x00**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	FREEZE	Not used	Not used	Reserved	ABM	Not used	SLA	TRIGOUT	EXTTRIGOUT	Not used		Not used	Not used		Not used	WR_REQ	Not used	Not used			SL	EUPM	CMP_CTRL	ACB42			ACB10		ARU_EN	TB12_SEL	MODE	
Mode	RW	RW	RW	R	RW	RW	RW	RW	RW	R		RW	RW		RW	RW	RW	RW			RW	RW	RW	RW			RW		RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	0	0b00		0	0b00		0	0	0	0b000			x	0	0	0b000			0b00		0	0	0b00	

Bit 1:0 **MODE**: ATOM channel mode select.
0b01 = ATOM Signal Output Mode Compare (SOMC)

Bit 2 **TB12_SEL**: Select time base value *TBU_TS1* or *TBU_TS2*.
0 = *TBU_TS1* selected for comparison
1 = *TBU_TS2* selected for comparison
Note: This bit is only applicable if three time bases are present in the GTM-IP. Otherwise, this bit is reserved.

Bit 3 **ARU_EN**: ARU Input stream enable.
0 = ARU Input stream disabled
1 = ARU Input stream enabled

Bit 5:4 **ACB10**: Signal level control bits.
0b00 = No signal level change at output (exception in tables 13.3.2.2.3 and 13.3.2.2.4 mode ACB42=001).

0b01 = Set output signal level to 1 when SL bit = 0 else output signal level to 0.

0b10 = Set output signal level to 0 when SL bit = 0 else output signal level to 1.

0b11 = Toggle output signal level (exception in tables 13.3.2.2.3 and 13.3.2.2.4 mode ACB42=001).

Note: These bits are only applicable if ARU_EN = '0'.

Bit 8:6 **ACB42**: ATOM control bits ACB(4), ACB(3), ACB(2)

0b000 = Compare in CCU0 and CCU1 in parallel, disable the CCUx on a compare match on either of compare units. Use *TBU_TS0* in CCU0 and *TBU_TS1* or *TBU_TS2* in CCU1.

0b001 = Compare in CCU0 and CCU1 in parallel, disable the CCUx on a compare match on either compare units. Use *TBU_TS0* in CCU0 and *TBU_TS1* or *TBU_TS2* in CCU1.

0b010 = Compare in CCU0 only against *TBU_TS0*.

0b011 = Compare in CCU1 only against *TBU_TS1* or *TBU_TS2*.

0b100 = Compare first in CCU0 and then in CCU1. Use *TBU_TS0*.

0b101 = Compare first in CCU0 and then in CCU1. Use *TBU_TS1* or *TBU_TS2*.

0b110 = Compare first in CCU0 and then in CCU1. Use *TBU_TS0* in CCU0 and *TBU_TS1* or *TBU_TS2* in CCU1.

0b111 = Cancel pending compare events.

Note: These bits are defining the compare strategy only if ARU_EN = 0.

Note: Independent of ARU_EN, a writing of 0b111 cancels any pending CCU0 or CCU1 compare.

Bit 9 **CMP_CTRL**: CCUx compare strategy select.

0 = Greater-equal compare against TBU time base values (TBU_TS1/2 >= CM0/1)

1 = Less-equal compare against TBU time base values (TBU_TS1/2 <= CM0/1)

Note: The compare unit CCU0 or CCU1 that compares against TBU_TS0 (depending on CCUx control mode defined by ACBI(4:2) or ACB42) always performs a greater-equal comparison, independent on CMP_CTRL bit.

Bit 10 **EUPM:** Extended Update Mode

0 = No extended update of CM0 and CM1 via CPU or ARU

1 = Extended update mode in case of compare strategy 'serve last': update of CM1 after CCU0 compare match possible via ARU or CPU.

Note: If EUPM=1 a write access to CM0 or CM1 never causes an AEI write status 0b10.

Note: this bit is only applicable in SOMC and SOMB mode.

Bit 11 **SL**: Initial signal level after channel enable.

0 = Low signal level
1 = High signal level
Note: Reset value depends on the hardware configuration chosen by silicon vendor.
Note: If the output is disabled, the output ATOM_OUT[x] is set to inverse value of SL.
If FREEZE=0, following note is valid:
Note: If the channel is disabled, the output register of SOU unit is set to value of SL.
If FREEZE=1, following note is valid:
Note: If the channel is disabled, the output register of SOU unit is not changed and output ATOM_OUT[x] is not changed.

Bit 14:12 **Not used**
Note: Not used in this mode.
Bit 15 **Not used**
Note: Not used in this mode.
Bit 16 **WR_REQ**: CPU write request bit
0 = No late update requested by CPU
1 = Late update requested by CPU
Note: The CPU can disable subsequent ARU read requests by the channel and can update the shadow registers with new compare values, while the compare units operate on old compare values received by former ARU accesses, if occurred.
Note: On a compare match event, the WR_REQ bit will be reset by hardware.
Note: At the point of the force update only the shadow registers SR0 and SR1 are transferred into the **CM0**, **CM1** registers. The output action is still defined by the ACBI bit field described by the ARU together with the old compare values for **CM0**/**CM1**.

Bit 17 **Not used**
Note: Not used in this mode.
Bit 19:18 **Not used**
Note: Not used in this mode.
Bit 20 **Not used** : not used in this mode
Note: Not used in this mode.
Bit 22:21 **Not used**
Note: Not used in this mode.
Bit 23 **EXTTRIGOUT**: select *TIM_EXT_CAPTURE(x)* as potential output signal *TRIG_[x]*
0 = signal *TRIG_[x-1]* is selected as output on *TRIG_[x]* (if TRIGOUT=1)
1 = signal *TIM_EXT_CAPTURE(x)* is selected as output on *TRIG_[x]* (if TRIGOUT=1)

Bit 24 **TRIGOUT**: Trigger output selection (output signal *TRIG_CHx*) of module ATOM_CHx.

0 = *TRIG_[x]* is *TRIG_[x-1]* or *TIM_EXT_CAPTURE(x)*
1 = *TRIG_[x]* is *TRIG_CCU0*

Bit 25 **SLA**: 'serve last' ARU communication strategy.
0 = Capture SRx time stamps after CCU0 match event not provided to ARU
1 = Capture SRx time stamps after CCU0 match event provided to ARU

Note: Please note, that setting of this bit has only effect, when ACBI(4:2) is configured for 'serve last' compare strategy ("100", "101", or "110").

Note: When this bit is not set, the captured time stamps in the shadow registers SRx are only provided after the CCU1 match occurred. The ACBO(4:3) bits always return "10" in that case.

Note: By setting this bit, the ATOM channel also provides the captured time stamps after the CCU0 match event to the ARU. The ACBO(4:3) bits are set to “01” in that case. After the CCU1 match event, the time stamps are captured again in the SRx registers and provided to the ARU. The ACBO(4:3) bits are set to “10”. When the data in the shadow registers after the CCU0 match was not consumed by an ARU destination and the CCU1 match occurs, the data in the shadow registers is overwritten by the new captured time stamps. The ATOM channel does not request new data from the ARU when the CCU0 match values are read from an ARU destination.

Bit 26 **Not used** : not used in this mode
Note: Not used in this mode.

Bit 27 **ABM:** ARU blocking mode
0 = ARU blocking mode disabled: ATOM reads continuously from ARU and updates CM0,CM1 and ACB bits independent of pending compare match event.
1 = ARU blocking mode enabled: after update of CM0, CM1 and ACB bit via ARU, no new data is read via ARU until compare match event occurred and SR0 and/or SR1 are read.

Bit 28 **Reserved**
Note: Read as zero, should be written as zero.

Bit 29 **Not used** : not used in this mode
Note: Not used in this mode.

Bit 30 **Not used**
Note: Not used in this mode.

Bit 31 **FREEZE**
0 = a channel disable/enable may change internal register and output register

1 = a channel enable/disable does not change an internal or output register but stops counter CN0 (in SOMP mode), comparison (in SOMC/SOMB mode) and shifting (in SOMS mode)

13.3.3 ATOM Signal Output Mode PWM (SOMP)

In ATOM Signal Output Mode PWM (SOMP) the ATOM sub-module channel is able to generate complex PWM signals with different duty cycles and periods. Duty cycles and periods can be changed synchronously and asynchronously. Synchronous change of the duty cycle and/or period means that the duty cycle or period duration changes after the end of the preceding period. An asynchronous change of period and/or duty cycle means that the duration changes during the actual running PWM period.

The signal level of the pulse generated inside the period can be configured inside the channel control register (**SL** bit of **ATOM[i]_CH[x]_CTRL** register). The initial signal output level for the channel is the inverse pulse level defined by the **SL** bit. Figure 13.3.3.1.1 depicts this behavior.

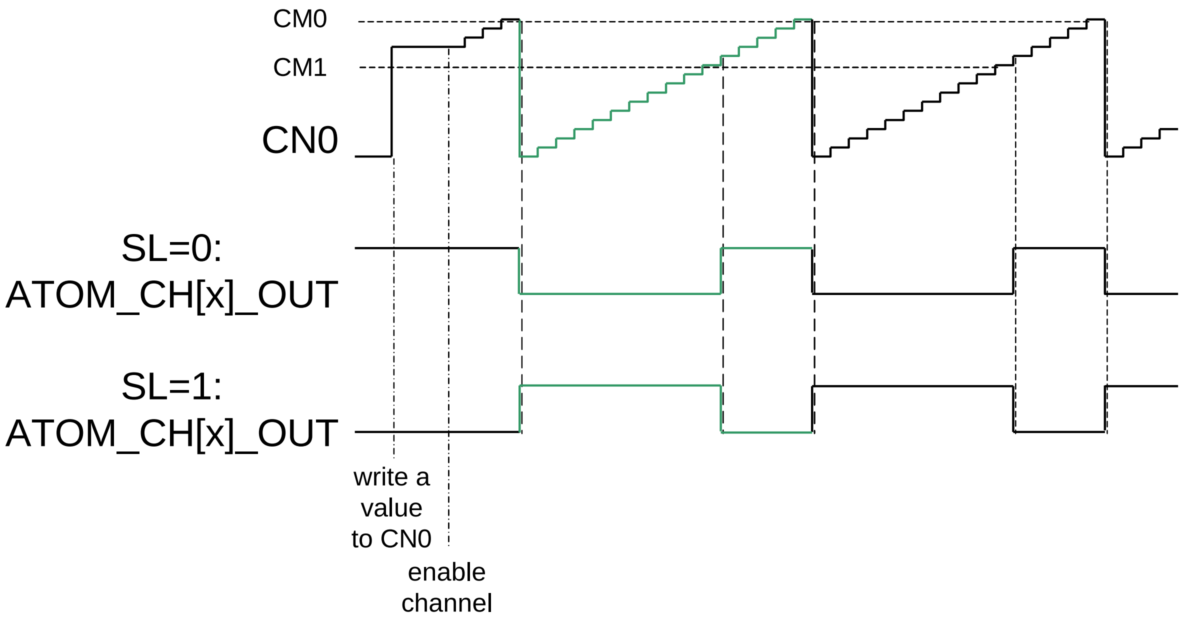
The counter **CN0** of each channel can run in two different modes depending on configuration of **UDMODE** in register **ATOM[i]_CH[x]_CTRL.**
By default the counter counts only up until it reaches **CM0** and is then reset to 0.
In the up down counter mode **CN0** switches between counting up and counting down.

13.3.3.1 Continuous Counting Up Mode

In SOMP mode with **UDMODE**=0b00 (i.e. **CN0** counts only up), depending on configuration bits RST_CCU0 of register **ATOM[i]_CH[x]_CTRL** the counter register **CN0** can be reset either when the counter value is equal to the compare value **CM0** (i.e. **CN0** counts only from 0 to **CM0**-1 and is then reset to 0) or when signaled by the ATOM[i] trigger signal *TRIG_[x-1]* of the preceding channel [x-1] (which can also be the last channel of preceding instance TOM[i-1]) or the trigger signal *TIM_EXT_CAPTURE(x)* of the assigned TIM channel [x].
In this case, if **UPEN_CTRL[x]**=1, also the working register **CM0**, **CM1** and **CLK_SRC** are updated.

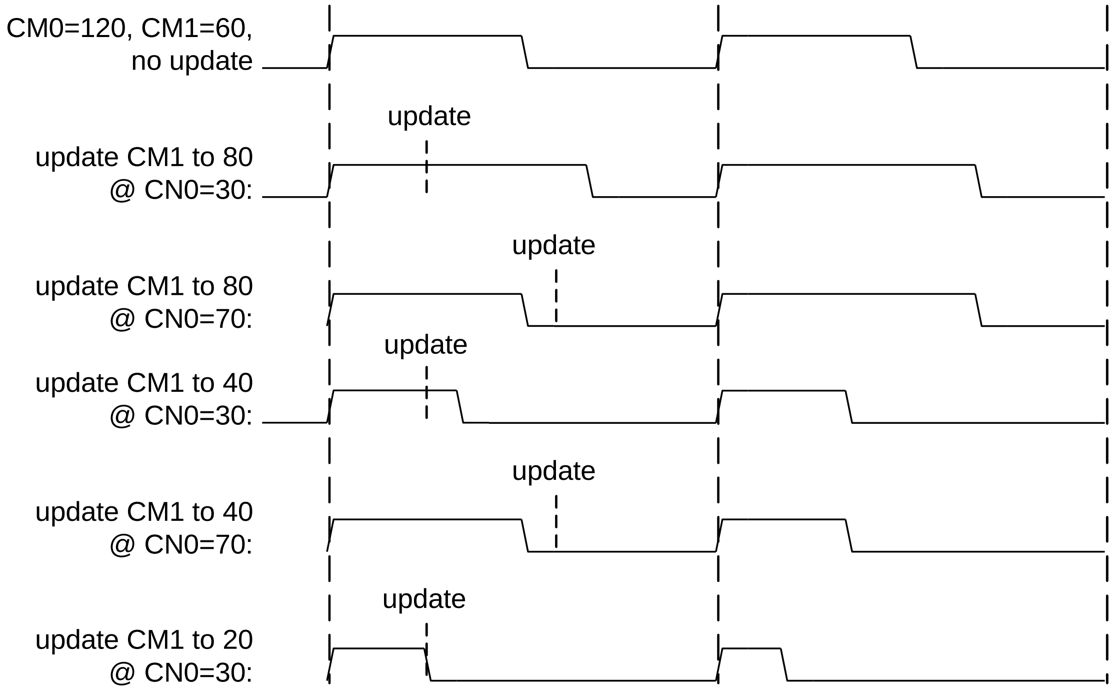
Note: As an exception, the input *TRIG_[0]* of instance ATOM0 is triggered by its own last channel cCATO via signal *TRIG_[cCATO]*. Please refer to device specific Appendix B for value cCATO of ATOM0.

13.3.3.1.1 PWM Output behavior with respect to the SL bit in the ATOM[i]_CH[x]_CTRL register if UDMODE = 0b00



On an asynchronous update, it is guaranteed, that no spike occurs at the output port of the channel due to a too late update of the operation registers. The behavior of the output signal due to the different possibilities of an asynchronous update during a PWM period is shown in figure 13.3.3.1.2.

13.3.3.1.2 PWM Output behavior in case of an asynchronous update of the duty cycle


CM0=120, CM1=60, no update
update
update CM1 to 80 @ CN0=30:
update
update CM1 to 80 @ CN0=70:
update
update CM1 to 40 @ CN0=30:
update
update CM1 to 40 @ CN0=70:
update
update CM1 to 20 @ CN0=30:

The duration of the pulse high or low time and period is measured with the counter in sub-unit CCU0. The trigger of the counter is one of the eight CMU clock signals configurable in the channel control register **ATOM[i]_CH[x]_CTRL**. The register **CM0** holds the duration of the period and the register **CM1** holds the duration of the duty cycle in clock ticks of the selected CMU clock.

If counter register **CN0** of channel x is reset by its own CCU0 unit (i.e. the compare match of **CN0**>=**CM0**-1 configured by **RST_CCU0**=0), following statements are valid:
- **CN0** counts from 0 to **CM0**-1 and is then reset to 0
- When **CN0** is reset from **CM0** to 0, an edge to SL is generated.
- When **CN0** is incrementing and reches **CN0** > **CM1**, an edge to !SL is generated.
- if **CM0**=0 or **CM0**=1, the counter **CN0** is constant 0.
- if **CM1**=0, the output is !**SL** = 0% duty cycle
- if **CM1** >= **CM0** and **CM0**>1, the output is **SL** = 100% duty cycle

If the counter register **CN0** of channel x is reset by the trigger signal coming from another channel or the assigned TIM module (configured by **RST_CCU0**=1), following statements are valid:
- **CN0** counts from 0 to MAX-1 and is then reset to 0 by trigger signal
- **CM0** defines the edge to **SL** value, **CM1** defines the edge to !**SL** value.
- if **CM0**=**CM1**, the output switches to **SL** if **CN0**=**CM0**=**CM1** (**CM0** has higher priority)
- if **CM0**=0 and **CM1**=MAX, the output is **SL** = 100% duty cycle
- if **CM0** > MAX, the output is !**SL** = 0% duty cycle, independent of **CM1**.

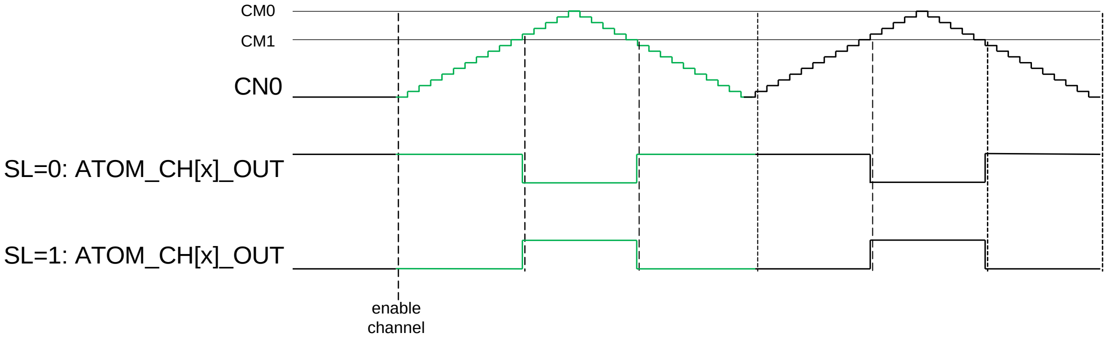
In case the counter value **CN0** reaches the compare value in register **CM0** (in fact **CM0**-1) or the channel receives an external update trigger via the *FUPD(x)* signal, a synchronous update is performed. A synchronous update means that the registers **CM0** and **CM1** are updated with the content of the shadow registers **SR0** and **SR1** and the **CLK_SRC** register is updated with the value of the **CLK_SRC_SR** register.

The clock source for the counter can be changed synchronously at the end of a period. If ARU access is disabled, this is done by using the bit field **CLK_SRC_SR** of register **ATOM[i]_CH[x]_CTRL** as shadow registers for the next CMU clock source.

13.3.3.2 Continuous Counting Up-Down Mode

In SOMP mode, if **CN0** counts up and down (**UDMODE** != 0b00), depending on configuration bit **RST_CCU0** of register **ATOM[i]_CH[x]_CTRL** the counter register **CN0** changes the direction either when the counter value is equal to the compare value **CM0** (in fact **CM0**-1), has counted down to 0 or when triggered by the ATOM[i] trigger signal *TRIG_[x-1]* of the preceding channel [x-1] (which can also be the last channel of preceding instance ATOM[i-1]) or the trigger signal *TIM_EXT_CAPTURE(x)* of the assigned TIM channel [x].
In this case, if **UPEN_CTRL[x]**=1, also the working register **CM0**, **CM1** and **CLK_SRC** are updated depending on **UDMODE**.

13.3.3.2.1 PWM Output behavior with respect to the SL bit in the ATOM[i]_CH[x]_CTRL register if UDMODE != 0b00



The clock of the counter register **CN0** can be one of the CMU clocks *CMU_CLK[x]*.
If **ARU_EN**=0, the clock for **CN0** is defined by **CLK_SRC_SR** value in register **ATOM[i]_CH[x]_CTRL.**
If **ARU_EN**=1, the clock for **CN0** is defined by **CLK_SRC** value received via ARU.
The duration of a period in multiples of selected **CN0** counter clock ticks is defined by the **CM0** configuration value (i.e. **CM0** defines half of period in up-down mode).
CM1 defines the duty cycle value in clock ticks of selected **CN0** counter clock (i.e. **CM0** defines half of duty cycle in up-down mode).

If counter register **CN0** of channel x is reset by its own CCU0 unit (i.e. the compare match of **CN0**>=**CM0**-1 configured by **RST_CCU0**=0), following statements are valid:
- **CN0** counts continuously first up from 0 to **CM0**-1 and then down to 0.
- if **CN0** >= **CM1**, the output is set to **SL**
- if **CM1**=0, the output is **SL** (i.e. 100% duty cycle)
- if **CM1**>= **CM0**, the output is !**SL** (i.e. 0% duty cycle)
- On output *ATOM[i]_CHx]_OUT* a PWM signal is generated. The period is defined by **CM0**, the duty cycle is defined by **CM1**.

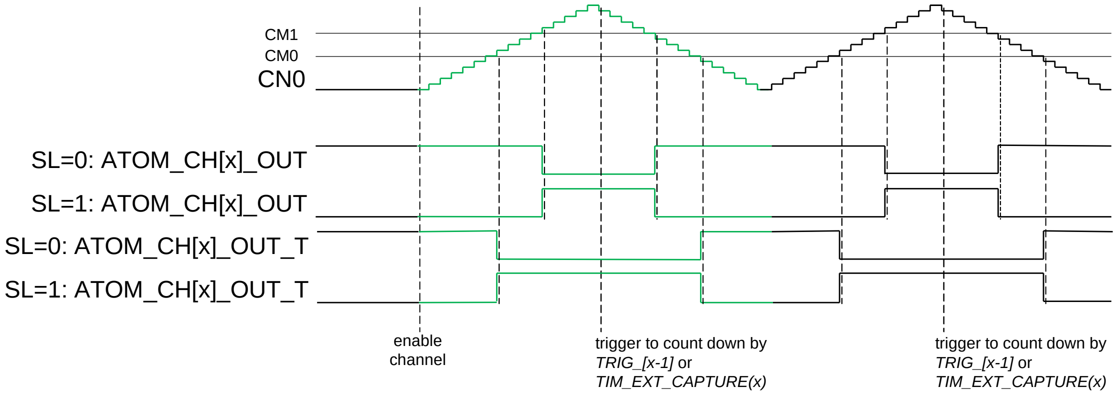
This behavior is depicted in figure 13.3.3.2.1.

If the counter register **CN0** of channel x is reset by the trigger signal coming from another channel or the assigned TIM module (configured by **RST_CCU0**=1), following statements are valid:
- **CN0** counts continuously first up. On a trigger signal the counter switches to count down mode. If **CN0** has reached 0, it counts up again.
- if **CN0** >= **CM1**, the output is set to **SL**
- if **CM1**=0, the output is **SL** (i.e. 100% duty cycle)
- if **CM1**>= **CM0**, the output is !**SL** (i.e. 0% duty cycle)
- On output *ATOM[i]_CHx]_OUT* a PWM signal is generated. The period is defined by the CCU0 trigger of triggering channel, the duty cycle is defined by **CM1**.
- On output *ATOM[i]_CHx]_OUT_T* a PWM signal is generated. The period is defined by the CCU0 trigger of triggering channel, the duty cycle is defined by **CM0**.

This behavior is depicted in figure 13.3.3.2.2.

Note that in case of up-down counter mode and RST_CCU0=1 it is recommended that
- the triggering channel and the triggered channel are both running in up-down mode and that
- the time between two trigger signals is equal to the time needed for CN0 of triggered channel to count back to 0 and again up to the same upper value.
The second recommendation can be reached by synchronizing the start of triggering channel and of triggered channel, i.e. let both channel start with a CN0 value 0.
Note that if there is a synchronization register in the trigger chain (indicated by value ATOM_TRIG_CHAIN in register **CCM[i]_HW_CONF**), the additional delay of the trigger by one clock period has to be taken into account by starting at triggering channel with a CN0 vaue 1 (+1 compared to CN0 of triggered channel).

13.3.3.2.2 PWM Output behavior in case of RST_CCU0=1


CM1
CM0
CN0
SL=0: ATOM_CH[x]_OUT
SL=1: ATOM_CH[x]_OUT
SL=0: ATOM_CH[x]_OUT_T
SL=1: ATOM_CH[x]_OUT_T
enable channel
trigger to count down by *TRIG_[x-1]* or *TIM_EXT_CAPTURE(x)*
trigger to count down by *TRIG_[x-1]* or *TIM_EXT_CAPTURE(x)*

13.3.3.3 ARU controlled update

If ARU access is enabled, the bits **ACBI(4)**, **ACBI(3)** and **ACBI(2)** received via ARU and stored in register **ATOM_[i]_CH[x]_STAT** are used as shadow register for the update of the CMU clock source register **CLK_SRC**.

For the synchronous update mechanism the generation of a complex PWM output waveform is possible without CPU interaction by reloading the shadow registers **SR0**, **SR1** and the **ACBI** bit field over the ACI sub-unit from the ARU, while the ATOM channel operates on the **CM0** and **CM1** registers.

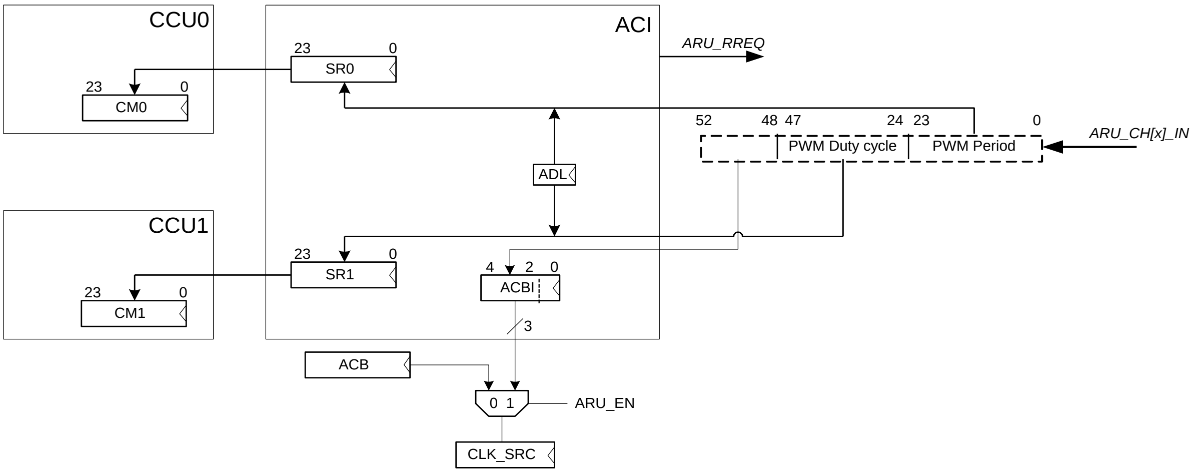
This internal update mechanism is established, when the old PWM period ends. The shadow registers are loaded into the operation registers, the counter register is reset, the new clock source according to the **CLK_SRC_SR** and **ACBI(4)**, **ACBI(3)** and **ACBI(2)** bits is selected and the new PWM generation starts.

In parallel, the ATOM channel issues a read request to the ARU to reload the shadow registers with new values while the ATOM channel operates on the operation registers. To guarantee the reloading, the PWM period must not be smaller than the worst case ARU round trip time and source for the PWM characteristic must provide the new data within this time. Otherwise, the old PWM values are used from the shadow registers.

When updated over the ARU the user has to ensure that the new period duration is located in the lower (bits 23 to 0) and the duty cycle duration is located in the upper (bits 47 to 24) ARU data word and the new clock source is specified in the ARU control bits 52 to 50.

This pipelined data stream character is shown in figure 13.3.3.3.1.

13.3.3.3.1 ARU Data input stream pipeline structure for SOMP mode



When an ARU transfer is in progress which means the *ARU_RREQ* is served by the ARU, the ACI locks the update mechanism of **CM0**, **CM1** and **CLK_SRC** until the read request has finished. The CCU0 and CCU1 operate on the old values when the update mechanism is locked.

13.3.3.4 CPU controlled update

The shadow registers **SR0** and **SR1** can also be updated over the AEI bus interface. In this case, **ARU_EN** hat to be set to 0.
When updated via the AEI bus the **CM0** and **CM1** update mechanism has to be locked via the **AGC_GLB_CTRL** register with the *UPENx* signal in the AGC sub-unit. To select the new clock source in this case, the CPU has to write to the CLK_SRC_SR bit field of the **ATOM[i]_CH[x]_CTRL** register.

For an asynchronous update of the duty cycle and/or period the new values must be written directly into the compare registers **CM0** and/or **CM1** while the counter **CN0** continues counting. This update can be done only via the AEI bus interface immediately by the CPU or by the *FUPD(x)* trigger signal triggered from the AGC global trigger logic. Values received through the ARU interface are never loaded asynchronously into the operation registers **CM0** and **CM1**. Therefore, the ATOM channel can generate a PWM signal on the output port pin *ATOM[i]_CH[x]_OUT* on behalf of the content of the **CM0** and **CM1** registers, while it receives new PWM values via the ARU interface ACI in its shadow registers.

On a compare match of **CN0** and **CM0** or **CM1** the output signal level of *ATOM[i]_CH[x]_OUT* is toggled according to the signal level output bit **SL** in the **ATOM[i]_CH[x]_CTRL** register.

Thus, the duty cycle output level can be changed during runtime by writing the new duty cycle level into the **SL** bit of the channel configuration register. The new signal level becomes active for the next trigger *CCU_TRIGx* (since bit **SL** is written).

Since the *ATOM[i]_CH[x]_OUT* signal level is defined as the reverse duty cycle output level when the ATOM channel is enabled, a PWM period can be shifted earlier by writing an initial offset value to **CN0** register. By doing this, the ATOM channel first counts until **CN0** reaches **CM0** and then it toggles the output signal at *ATOM[i]_CH[x]_OUT*.

13.3.3.5 One-shot Counting Up Mode

The ATOM channel can operate in One-shot mode when the **OSM** bit is set in the channel control register. One-shot mode means that a single pulse with the pulse level defined in bit **SL** is generated on the output line.

First the channel has to be enabled by setting the corresponding **ENDIS_STAT** value. In one-shot mode the counter **CN0** will not be incremented once the channel is enabled.
A write access to the register **CN0** triggers the start of pulse generation (i.e. the increment of the counter register **CN0**).

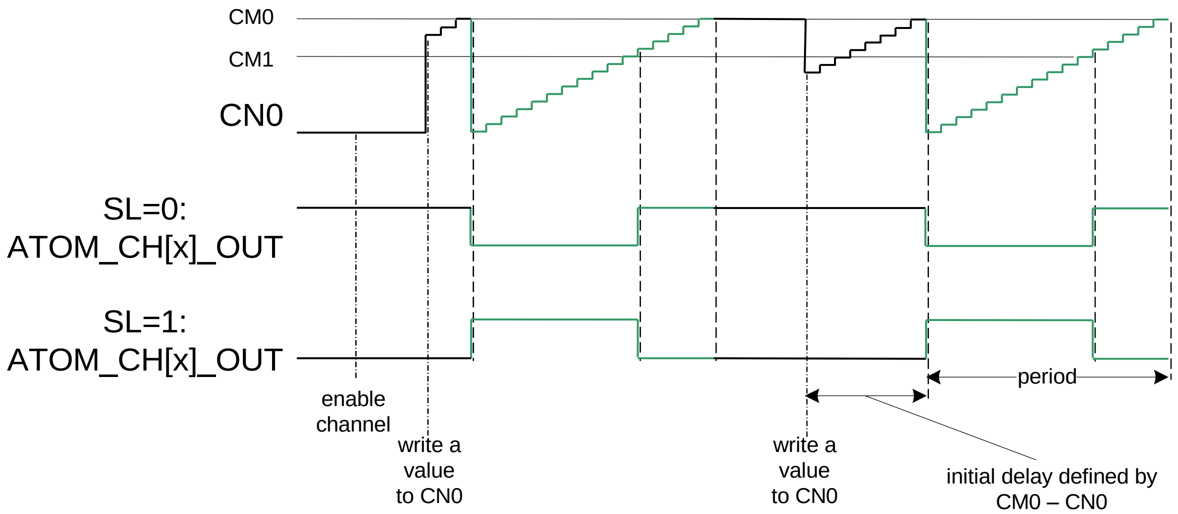
If the counter **CN0** is reset from **CM0**-1 back to zero, the first edge at *ATOM[i]_CH[x]_OUT* is generated.
To avoid an update of **CMx** register with content of **SRx** register at this point in time, the automatic update should be disabled by setting **UPEN_CTRL[x]** = 0b00 (in register **ATOM[i]_CH[x]_CTRL**)

The second edge is generated if **CN0** is greater or equal than **CM1** (i.e. **CN0** was incremented until it has reached **CM1** or **CN0** is greater than **CM1** after an update of **CM1**).

If the counter **CN0** has reached the value of **CM0**-1 a second time, the counter stops. The new value of **CN0** determines the start delay of the first edge. The delay time of the first edge is given by (**CM0-CN0**) multiplied with period defined by current value of **CLK_SRC**.

Figure 13.3.3.5.1 depicts the pulse generation in SOMP one-shot mode.

13.3.3.5.1 PWM Output with respect to configuration bit SL in One-shot counting up mode: trigger by writing to CN0



Further output of single pulses can be started by a write access to register **CN0**.
If **CN0** is already incrementing (i.e. started by writing to **CN0** a value CN0start < **CM0**), the effect of a second write access to **CN0** depends on the phase of CN0:
phase 1: update of **CN0** before **CN0** reaches first time **CM0** (in fact **CM0**-1)
phase 2: update of **CN0** after **CN0** has reached first time **CM0** (in fact **CM0**-1) but is less than **CM1**
phase 3: update of **CN0** after **CN0** has reached first time **CM0** (in fact **CM0**-1) and **CN0** is greater than or equal **CM1**

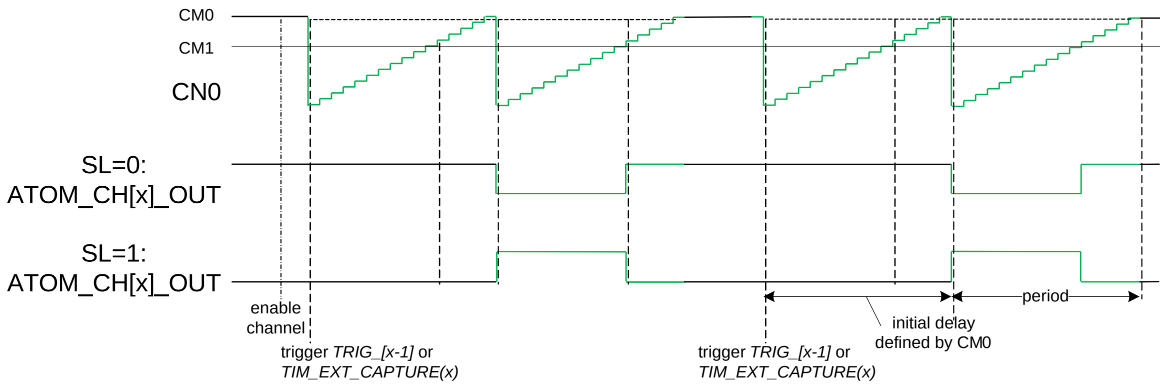
In phase 1: writing to counter **CN0** a value CN0new < **CM0** leads to a shift of first edge (generated if **CN0** is reset first time from **CM0**-1 back to 0) by the time **CM0**-CN0new.

In phase 2: writing to incrementing counter **CN0** a value CN0new < **CM1** while CN0old is below **CM1** leads to a lengthening of the pulse. The counter **CN0** stops if it reaches **CM0**.

In phase 3: Writing to incrementing counter **CN0** a value CN0new while CN0old is already greater than or equal **CM1** leads to an immediate restart of a single pulse generation inclusive the initial delay defined by **CM0** - CN0new.

If a channel is configured to one-shot mode and configuration bit **OSM_TRIG** is set to 1, the trigger signal *OSM_TRIG* (i.e. *TRIG_[x-1]* or *TIM_EXT_CAPTURE(x)*) triggers start of one pulse generation.

13.3.3.5.2 PWM Output with respect to configuration bit SL in one-shot mode: trigger by *TRIG_[x-1]* or TIM_*EXT_CAPTURE(x)*



13.3.3.6 One-shot Counting Up-Down Mode

The ATOM channel can operate in one-shot counting up-down mode when the bit **OSM** = 1 and the **UDMODE** != 0b00. One-shot mode means that a single pulse with the pulse level defined in bit **SL** is generated on the output line.

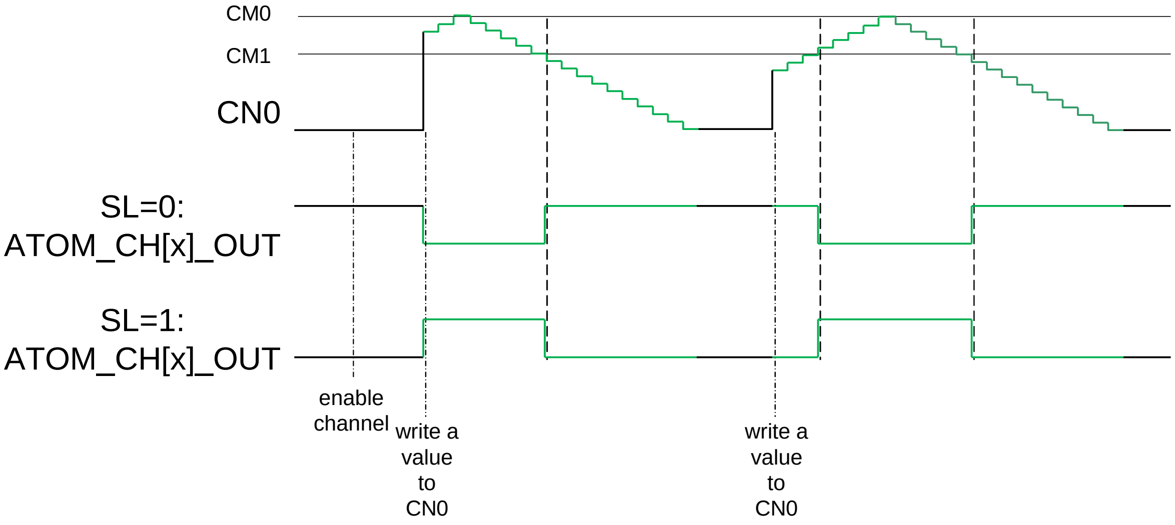
First the channel has to be enabled by setting the corresponding **ENDIS_STAT** value. In one-shot mode the counter **CN0** will not be incremented once the channel is enabled.
A write access to the register **CN0** triggers the start of pulse generation (i.e. the increment of the counter register **CN0**).

To avoid an update of **CMx** register with content of **SRx** register at this point in time, the automatic update should be disabled by writing **UPEN_CTRL[x]** = 0b01 (see register **ATOM[i]_AGC_GLB_CTRL**).

If the counter **CN0** is greater or equal than **CM1**, the output *ATOM[i]_CH[x]_OUT* is set to **SL** value.
If the counter **CN0** is less than **CM1**, the output *ATOM[i]_CH[x]_OUT* is set to **!SL** value.
If the counter **CN0** has reached the value 0 (by counting down), it stops.
The new value of **CN0** determines the start delay of the first edge. The delay time of the first edge is given by (**CM1-CN0**) multiplied with period defined by current value of **CLK_SRC**.

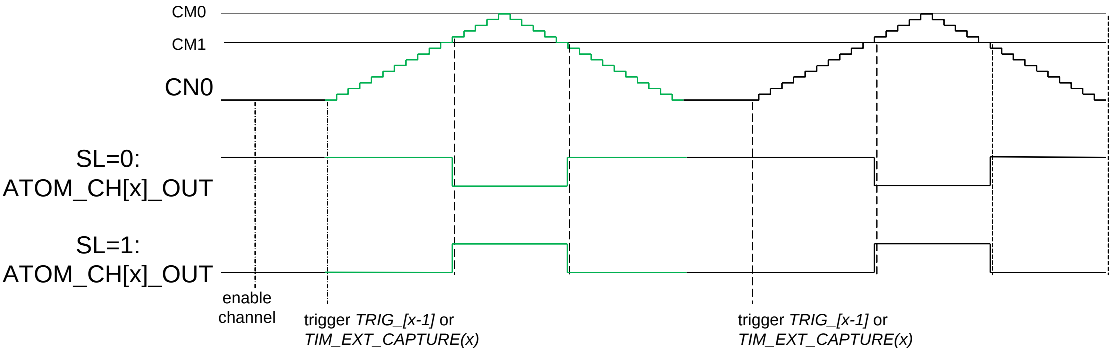
Figure 13.3.3.6.1 depicts the pulse generation in SOMP one-shot mode.

13.3.3.6.1 PWM Output with respect to configuration bit SL in one-shot counting up-down mode: trigger by writing to CN0



Further output of single pulses can be started by writing to register **CN0**.
If a channel is configured to one-shot counting up-down mode and configuration bit **OSM_TRIG** is set to 1, the trigger signal *OSM_TRIG* (i.e. *TRIG_[x-1]* or *TIM_EXT_CAPTURE(x)*) triggers start of one pulse generation.

13.3.3.6.2 PWM Output with respect to configuration bit SL in one-shot counting up-down mode: trigger by *TRIG_[x-1]* or TIM_*EXT_CAPTURE(x)*



13.3.3.7 Pulse Count Modulation Mode

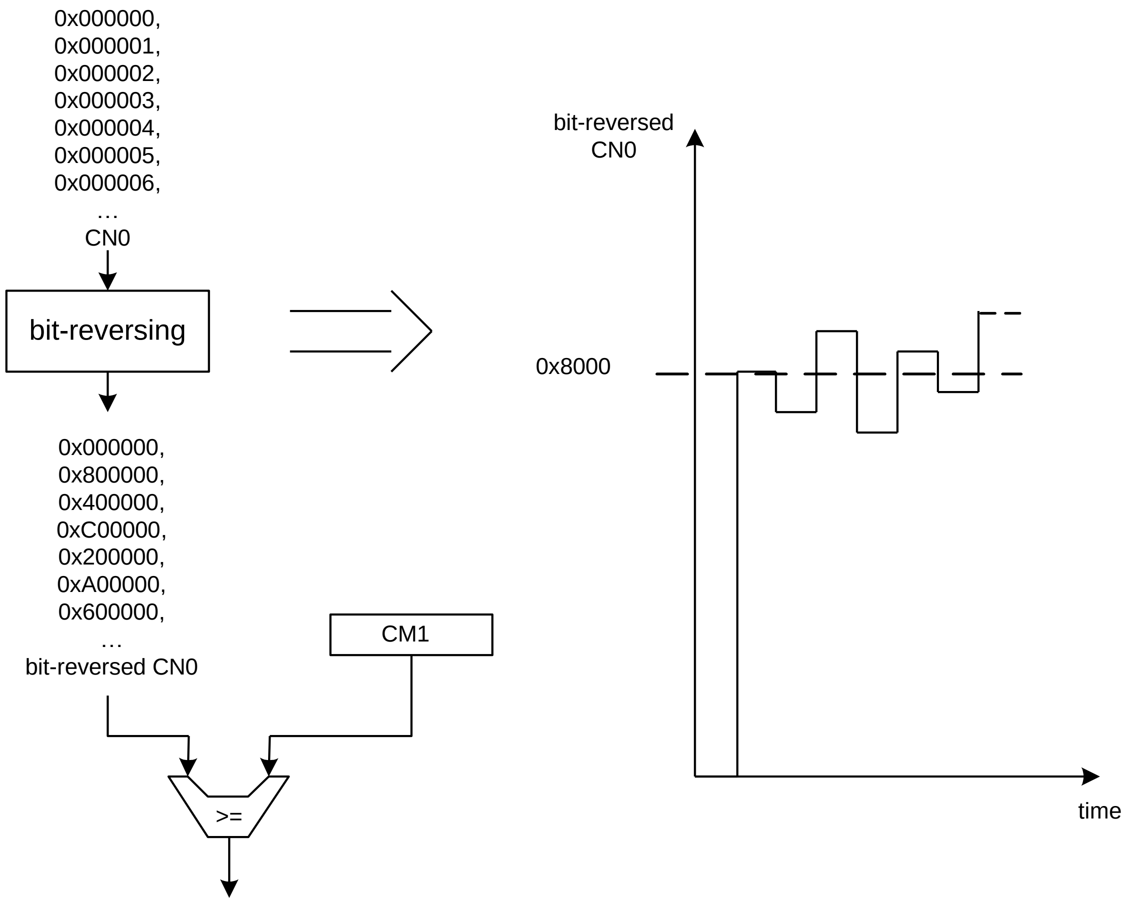
At the output *ATOM[i]_CH[x]_OUT* a pulse count modulated signal can be generated instead of the simple PWM output signal in SOMP mode.

The PCM mode is enabled by setting bit **BITREV** to 1 (bit 6 in **ATOM[i]_CH[x]_CTRL** register).
Please note that it is device specific, in which channel the PCM mode is available. Please refer to device specific Appendix B for this information.

With the configuration bit **BITREV**=1 a bit-reversing of the counter output **CN0** is configured. In this case the bits LSB and MSB are swapped, the bits LSB+1 and MSB-1 are swapped, the bits LSB+2 and MSB-2 are swapped and so on.

The effect of bit-reversing of the **CN0** register value is shown in the following figure 13.3.3.7.1.

13.3.3.7.1 Bit reversing of counter **CN0** output



In the PCM mode the counter register **CN0** is incremented by every clock tick depending on configured CMU clock (*CMU_CLK*).

The output of counter register **CN0** is first bit-reversed and then compared with the configured register value **CM1**.

If the bit-reversed value of register **CN0** is greater or equal than **CM1**, the SR-FlipFlop of sub-module SOU is set (i.e. set to inverse value of **SL**) otherwise the SR-FlipFlop is reset (i.e. to the value of **SL**). This generates at the output *ATOM[i]_CH[x]_OUT* a pulse count modulated signal.

In PCM mode the **CM0** register - in which the period is defined - normally has to be set to its maximum value 0xFFFFFF.

To reduce time period of updating duty cycle value in **CM1** register, it is additionally possible to setup period value in **CM0** register to smaller values than maximum value as described before.
Possible values for **CM0** register are each even numbered values to the power of 2 e.g. 0x800000, 0x400000, 0x200000 ... .
In this case the duty cycle has to be configured in the following manner.

Depending on how much the period in **CM0** register is decreased - means shifted right starting from 0x1000000 - the duty cycle in **CM1** register has to be shifted left (= rotated: shift MSB back into LSB) with same value, e.g. :

period **CM0** = 0x001000 -> shifted 8 bits right from 0x1000000
--> so duty cycle has to be shifted left 8 bit :
e.g. 50% duty cycle = 0x0008000 -> shift 8 bits left -> **CM1** = 0x800000

More examples :

period CM0	->	duty cycle	->	shift	->	CM1
0xFFFFFF	->	0x800000	->	no shift	->	0x800000
0x800000	->	0x400000	->	shift 1 bit left	->	0x800000
0x400000	->	0x100000	->	shift 2 bits left	->	0x400000
0x200000	->	0x0FFFFF	->	shift 3 bits left	->	0x7FFFF8
0x100000	->	0x033333	->	shift 4 bits left	->	0x333330
0x080000	->	0x005555	->	shift 5 bits left	->	0x0AAAA0
...						
0x000020	->	0x000008	->	shift 19 bits left	->	0x400000
0x000010	->	0x000005	->	shift 20 bits left	->	0x500000
...						

Note: In this mode the interrupt CCU1TC (see register **ATOM[i]_CH[x]_IRQ_NOTIFY**) is set every time if bit reverse value of **CN0** is greater or equal than **CM1** which may be multiple times during one period. Therefore, from application point of view it is not useful to enable this interrupt.

13.3.3.8 Trigger generation

For applications with constant PWM period defined by **CM0**, it is not necessary to update regularly the **CM0** register with **SR0** register. For these applications the **SR0** register can be used to define an additional output signal and interrupt trigger.
If bit **SR0_TRIG** in register **ATOM[i]_CH[x]_CTRL** is set, the register **SR0** is no longer used as a shadow register for register **CM0**. Instead, **SR0** is compared against **CN0** and if both are equal, a pulse of signal level 1 is generated at the output *ATOM[i]_CH[x]_OUT_T*.
The bit **SR0_TRIG** should only be set if bit RST_CCU0 of this channel is 0.
Note: If **ARU_EN**=1 and both **SR0** and **SR1** are updated via ARU, the new **SR0** value is used immediately after update. Update of **SR0** via ARU can be suppressed by ADL configuration in register **ATOM[i]_CH[x]_CTRL.**

If bit **SR0_TRIG** is set the interrupt notify flag **CCU1TC** is no longer set on a compare match of **CM1** and **CN0**. Instead, the **CCU1TC** interrupt notify flag is set in case of a compare equal match of **SR0** and **CN0**.

With configuration bit **TRIG_PULSE** one can select if the output *ATOM[i]_CH[x]_OUT_T* is high as long as **CN0**=**SR0** (**TRIG_PULSE**=0) or if there will be only one pulse of length one *SYS_CLK* period when **CN0** becomes **SR0** (**TRIG_PULSE**=1).

The ATOM output signal routing to DTM or GTM-IP top level is described in chapter 14.7

13.3.3.8.1 Register ATOM[i]_CH[x]_CTRL in SOMP mode

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_0x00**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	FREEZE	Not used	EXT_FUPD	Reserved	Not used	OSM	Not used	TRIGOUT	EXTTRIGOUT	EXT_TRIG	OSM_TRIG	RST_CCU0	UDMODE		TRIG_PULSE	Not used	ECLK_SRC	CLK_SRC_SR			SL	Not used	Not used	Not used	SR0_TRIG	BITREV	ADL		ARU_EN	Not used	MODE	
Mode	RW	RW	RW	R	RW	RW	RW	RW	RW	RW	RW	RW	RW		RW	RW	RW	RW			RW	RW	RW	RW	RW	RW	RW		RW	RW	RW	
Initial Value	0	0	0	0b00	0	0	0	0	0	0	0	0	0b00		0	0	0	0b000			x	0	0	0	0	0	0b00		0	0	0b00	

Bit 1:0 **MODE**: ATOM channel mode select.
0b10 = ATOM Signal Output Mode PWM (SOMP)

Bit 2 **Not used**
Note: Not used in this mode.

Bit 3 **ARU_EN**: ARU Input stream enable
0 = ARU Input stream disabled
1 = ARU Input stream enabled

Bit 5:4 **ADL**: ARU data select for SOMP.
0b00 = Load both ARU words into shadow registers
0b01 = Load ARU low word (Bits 23..0) into shadow register SR0
0b10 = Load ARU high word (Bits 47..24) into shadow register SR1
0b11 = Reserved
Note: This bit field is only relevant in SOMP mode to select the ARU data source.

Bit 6 **BITREV**: Bit-reversing of output of counter register **CN0**. This bit enables the PCM mode
Note : It is device specific, in which channel the PCM mode is available. Please refer to device specific Appendix B for this information.

Bit 7 **SR0_TRIG**: **SR0** is used to generate a trigger on output *ATOM[i]_CH[x]_OUT_T* if equal to **CN0**.
0 = SR0 is used as a shadow register for register CM0.
1 = SR0 is not used as a shadow register for register CM0. SR0 is compared with CN0 and if both are equal, a trigger pulse is generated at output *ATOM[i]_CH[x]_OUT_T* .
Note: This bit is only relevant in SOMP mode.
Note: This bit should only be set if RST_CCU0 of this channel is 0.

Bit 8 **Not used**
Note: Not used in this mode.

Bit 9 **Not used**
Note: Not used in this mode.

Bit 10 **Not used**

Note: Not used in this mode.

Bit 11 **SL**: Signal level for pulse of PWM.

0 = Low signal level

1 = High signal level

Note: Reset value depends on the hardware configuration chosen by silicon vendor.

Note: If the output is disabled, the output ATOM_OUT[x] is set to inverse value of SL.

If FREEZE=0, the following note is valid

Note: If the channel is disabled, the output register of SOU unit is set to inverse value of SL.

If FREEZE=1, the following note is valid

Note: If the channel is disabled, the output register of SOU unit is not changed and output ATOM_OUT[x] is not changed.

Bit 14:12 **CLK_SRC_SR**: Shadow register for CMU clock source register CLK_SRC

If ECLK_SRC=0 / ECLK_SRC=1:

0b000 = *CMU_CLK0* selected / *CMU_CLK0* selected

0b001 = *CMU_CLK1* selected / *CMU_CLK1* selected

0b010 = *CMU_CLK2* selected / *CMU_CLK2* selected

0b011 = *CMU_CLK3* selected / Reserved

0b100 = *CMU_CLK4* selected / clock stopped

0b101 = *CMU_CLK5* selected / *TRIG[x-1]* selected

0b110 = *CMU_CLK6* selected / *TIM_EXT_CAPTURE[x]* selected

0b111 = *CMU_CLK7* selected / *CMU_CLK7* selected

Note: This register is a shadow register for the register CLK_SRC. Thus, if the CMU_CLK source for PWM generation should be changed during operation, the old CMU_CLK has to operate until the update of the ATOM channels internal CLK_SRC register by the CLK_SRC_SR content is done either by an end of a period or a FORCE_UPDATE.

Note: After (channel) reset the selected CLK_SRC value is the SYS_CLK (input of Global Clock Divider). To use in SOMP mode one of the CMU_CLKx, it is recommended to perform a forced update of CLK_SRC with the value of CLK_SRC_SR value before/with enabling the channel.

Note: In case of ECLK_SRC=1 and CLK_SRC_SR = 0b011/0b100/0b101/0b110 a force update leads to an immediate update of CM0, CM1 and CLK_SRC.

Bit 15 **ECLK_SRC:** Extend CLK_SRC

0 = CLK_SRC_SR set 1 selected

1 = CLK_SRC_SR set 2 selected

See bit CLK_SRC_SR description for details.

Bit 16 **Not used**

Note: Not used in this mode.

Bit 17 **TRIG_PULSE**: Trigger output pulse length of one SYS_CLK period

0 = output on TOM[i]_OUT[x]_T is 1 as long as CN0=SR0 (if SR=_TRIG=1)
1 = output on TOM[i]_OUT[x]_T is 1 for only one SYS_CLK period if CN0=SR0 (if SR=_TRIG=1)

Bit 19:18 **UDMODE**: up/down counter mode
0b00 = up/down counter mode disabled: CN0 counts always up
0b01 = up/down counter mode enabled: CN0 counts up and down, CM0,CM1 are updated if CN0 reaches 0 (i.e. changes from down to up)
0b10 = up/down counter mode enabled: CN0 counts up and down, CM0,CM1 are updated if CN0 reaches CM0 (i.e. changes from up to down)
0b11 = up/down counter mode enabled: CN0 counts up and down, CM0,CM1 are updated if CN0 reaches 0 or CM0 (i.e. changes direction)

Note: This mode is only applicable in SOMP mode.

Bit 20 **RST_CCU0**: Reset source of CCU0
0 = Reset counter register CN0 to 0 on matching comparison with CM0
1 = Reset counter register CN0 to 0 on trigger *TRIG_[x-1]* or *TIM_EXT_CAPTURE(x)* .
Note: If RST_CCU0=1 and UPEN_CTRLx=1 are set, *TRIG_[x-1]* or TIM_*EXT_CAPTURE(x)* triggers also the update of work register (CM0, CM1 and CLK_SRC).

Bit 21 **OSM_TRIG**: enable trigger of one-shot pulse by trigger signal *OSM_TRIG*
0 = signal OSM_TRIG cannot trigger start of single pulse generation
1 = signal OSM_TRIG can trigger start of single pulse generation (if bit OSM = 1)

Note: This bit should only be set if bit OSM=1 and bit RST_CCU0=0.

Bit 22 **EXT_TRIG**: select *TIM_EXT_CAPTURE(x)* as trigger signal
0 = signal *TIM_[x-1]* is selected as trigger to reset CN0 or to start single pulse generation.
1 = signal *TIM_EXT_CAPTURE(x)* is selected

Bit 23 **EXTTRIGOUT**: select *TIM_EXT_CAPTURE(x)* as potential output signal *TRIG_[x]*
0 = signal *TRIG_[x-1]* is selected as output on *TRIG_[x]* (if TRIGOUT=1)
1 = signal *TIM_EXT_CAPTURE(x)* is selected as output on *TRIG_[x]* (if TRIGOUT=1)

Bit 24 **TRIGOUT**: Trigger output selection (output signal *TRIG_CHx*) of module ATOM_CHx.
0 = *TRIG_[x[* is *TRIG_[x-1*] or *TIM_EXT_CAPTURE(x)*.

1 = *TRIG_[x]* is *TRIG_CCU0*

Bit 25 **Not used**
Note: Not used in this mode.

Bit 26 **OSM**: One-shot mode
0 = Continuous PWM generation after channel enable
1 = A single pulse is generated

Bit 27 **Not used**
Note: Not used in this mode.

Bit 28 **Reserved**
Note: Read as zero, should be written as zero.

Bit 29 **EXT_FUPD:** external forced update
0 = use FUPD(x) signal from AGC to force update
1 = use TIM_EXT_CAPTURE signal to force update

Note: This bit is only applicable in SOMP and SOMS mode.

Bit 30 **Not used**
Note: Not used in this mode.

Bit 31 **FREEZE**
0 = a channel disable/enable may change internal register and output register
1 = a channel enable/disable does not change an internal or output register but stops counter CN0 (in SOMP mode), comparison (in SOMC/SOMB mode) and shifting (in SOMS mode)

Note: if channel is disabled and ouptut is enabled, in SOMP mode with UDMODE!=0b00 the output is dependng directly on SL bit, independent on FREEZE mode.

13.3.4 ATOM Signal Output Mode Serial (SOMS)

In ATOM Signal Output Mode Serial (SOMS) the ATOM channel acts as a serial output shift register where the content of the **CM1** register in the CCU1 unit is shifted out whenever the unit is triggered by the selected *CMU_CLK* input clock signal. The shift direction is configurable with the **ACB(0)** bit inside the **ATOM[i]_CH[x]_CTRL** register when ARU is disabled and the **ACBI(0)** bit inside the **ATOM[i]_CH[x]_STAT** register when ARU is enabled.

The data inside the **CM1** register has to be aligned according to the selected shift direction in the **ACB(0)**/**ACBI(0)** bit. This means that when a right shift is selected, that the data word has to be aligned to bit 0 of the **CM1** register and when a left shift is selected, that the data has to be aligned to bit 23 of the **CM1** register.

13.3.4.1 SOMS Mode output generation

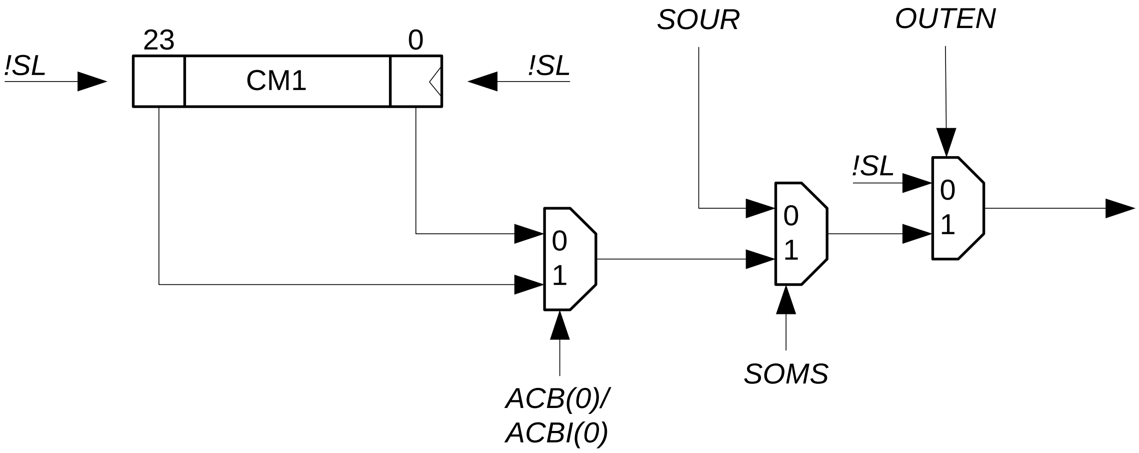
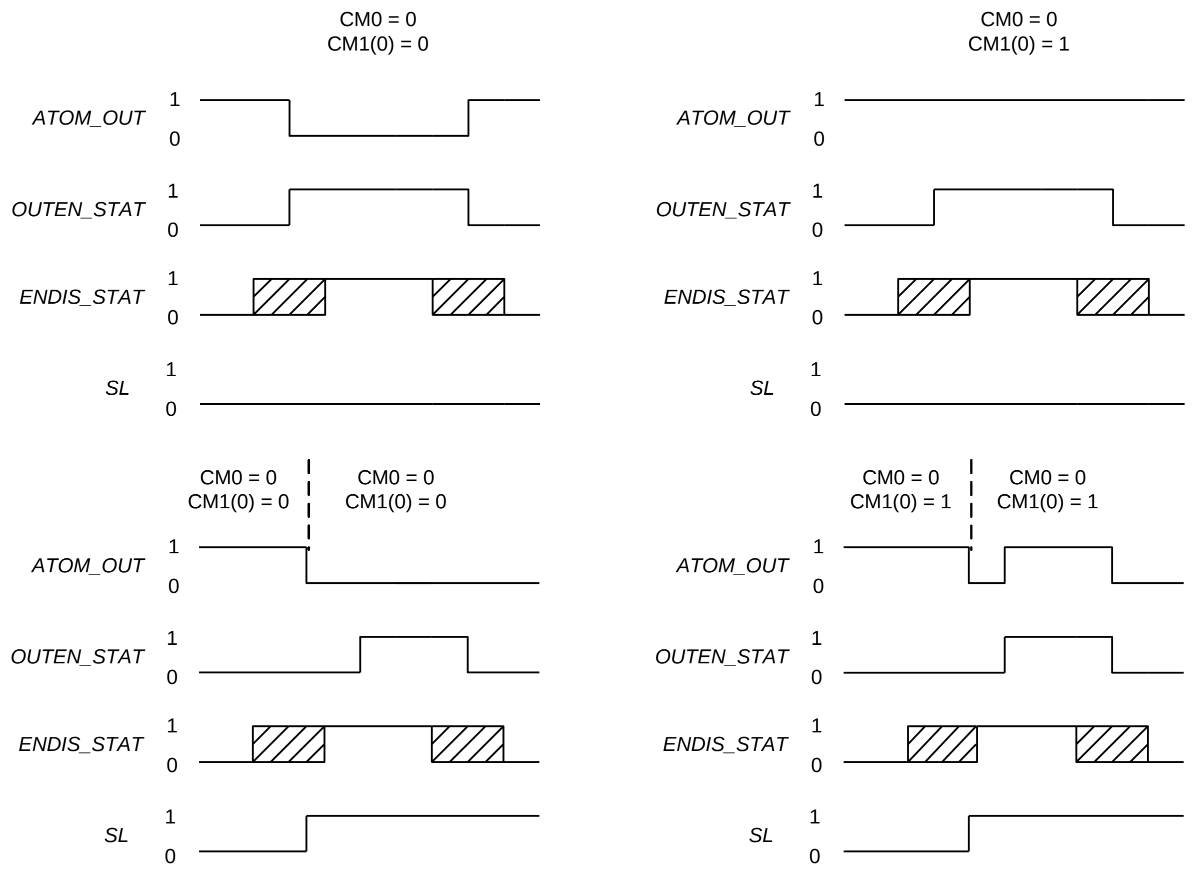


Figure 13.3.4.1 shows the output generation in case of SOMS mode is selected. In SOMS mode CCU0 runs in counter/compare mode and counts the number of bits shifted out so far. The total number of bits that should be shifted is defined as **CM0**. The total number of bits that are visible at *ATOM_OUT* is **CM0**+1.

When the output is disabled the *ATOM_OUT* is set to the inverse **SL** bit definition. When the content of the **CM1** register is shifted out, the inverse signal level is shifted into the **CM1** register.

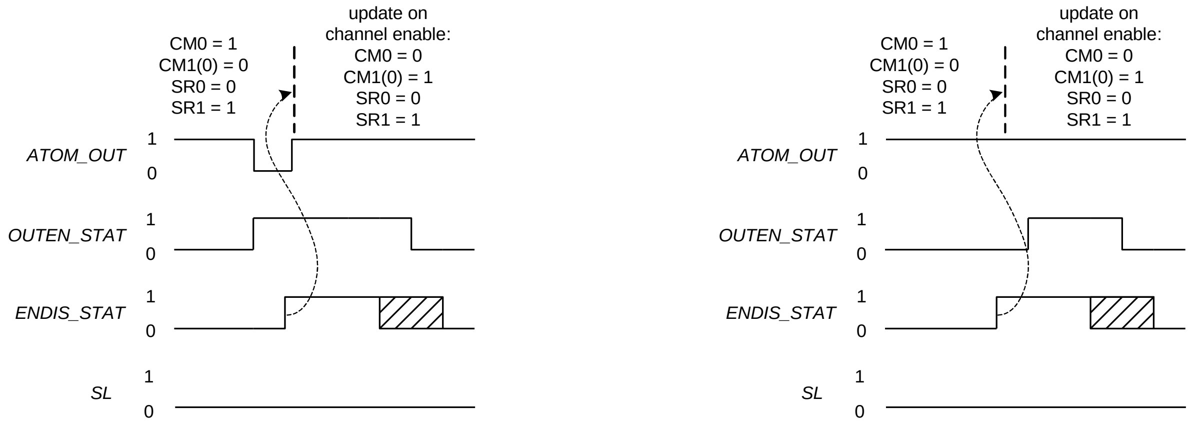
When the output is enabled while **UPEN_CTRL[x]** is disabled, the ATOM_OUT signal level is defined by **CM1** bit 0 or 23, dependent on the shift direction defined by **ACB(0)** or **ACBI(0)** register setting. Figure 13.3.4.2 should clarify the ATOM channel startup behavior in this case for right shift. For left shift the **CM1** bit 0 in 13.3.4.2 has to be replaced by **CM1** bit 23.

13.3.4.2 SOMS Output signal level at startup, UPEN_CTRL[x] disabled



If **UPEN_CTRL[x]** is set and the channel is enabled, the output level is defined by bit 0 or 23 of **CM1** register dependent on the shift direction. Figure 13.3.4.3 shows the output behavior in that case.

13.3.4.3 SOMS Output signal level at startup, UPEN_CTRL[x] enabled



When the serial data to be shifted is provided via ARU the number of bits that should be shifted has to be defined in the lower 24 bits of the ARU word (23 to 0) and the data that is to be shifted has to be defined in the ARU bits 47 to 24 aligned according to the shift direction. This shift direction has to be defined in the ARU word bit 48 (**ACB0** bit).

If bit **UPEN_CTRL[x]** of a channel x is set, after update of **CM0/CM1** register with the content of the **SR0/SR1** register, a new ARU read request is set up.

If bit **UPEN_CTRL[x]** of a channel x is not set, no (further) ARU read request is set up (because the **SR0/SR1** register are never used for update) and the ATOM may stop shifting after **CN0** has reached **CM0**. Note, that in this case also no automatic restart of shifting is possible.

If a channel is enabled with the settings SOMS mode and **ARU_EN** = 1, the first received values from ARU are stored in register **SR0** and **SR1.** If **CN0** and **CM0** are 0 (i.e. **CN0** is not counting) and the update of channel x is enabled (**UPEN_CTRL[x]**=1), an immediate update of the register **CM0** and **CM1** is also done. This update of **CM0** and **CM1** triggers the start of shifting.

It is recommended to configure the ATOM channel in One-shot mode when the **ARU_EN** bit is not set, since the ATOM channel would reload new values from the shadow registers when **CN0** reaches **CM0**.

13.3.4.4 SOMS mode with ARU_EN = 1 and OSM = 0, UPEN_CTRL[x] = 1:

In case of bit **ARU_EN** is set and bit **OSM** is not set, the channel is running in the SOMS continuous mode. Then, if the content of the **CM0** register equals the counter **CN0**, the **CM0** and **CM1** registers are reloaded with the **SR0** and **SR1** content and new values are requested from the ARU. If the update of the shadow registers does not happen before **CN0** reaches **CM0** the old values of **SR0** and **SR1** are used to reload the operation registers.

In contrast to controlling the channel via AEI, the shift direction defined by ARU word bit 48 has only effect after the update of **CMx** operation registers from the **SRx** registers.

13.3.4.5 SOMS mode with ARU_EN = 1 and OSM = 1, UPEN_CTRL[x] = 1:

In case of bit **ARU_EN** is set and bit **OSM** is set, the channel is running in the SOMS one-shot mode. Then, if the content of the **CM0** register equals the counter **CN0** and if new values are available in **SR0** and **SR1** (bit DV set), the **CM0** and **CM1** registers are reloaded with the **SR0** and **SR1** content and new values are requested from the ARU. If no new values are available in **SR0** and **SR1**, the register **CM0** and **CM1** will not be updated, the counter **CN0** stops and the ATOM channel continues to request new data from ARU. A later reception of new ARU data in **SR0** and **SR1** will immediately force the update of the register **CM0** and **CM1** and restart the counter **CN0**.

13.3.4.6 SOMS mode with ARU_EN = 0 and OSM = 0, UPEN_CTRL[x] = 1:

In case of bit **ARU_EN** is not set and bit **OSM** is not set, the ATOM channel updates its CM0/CM1 register with the content of the SR0/SR1 register and restarts shifting immediately. The first bit of new CM1 register value will be applied at the output without any gap to the last bit of the previous CM1 register value.

13.3.4.7 SOMS mode with ARU_EN = 0 and OSM = 1, UPEN_CTRL[x] = 1:

In case of bit **ARU_EN** is not set and bit **OSM** is set, the ATOM channel stops shifting when **CN0** reaches **CM0** and no update of **CM0** and **CM1** is performed.

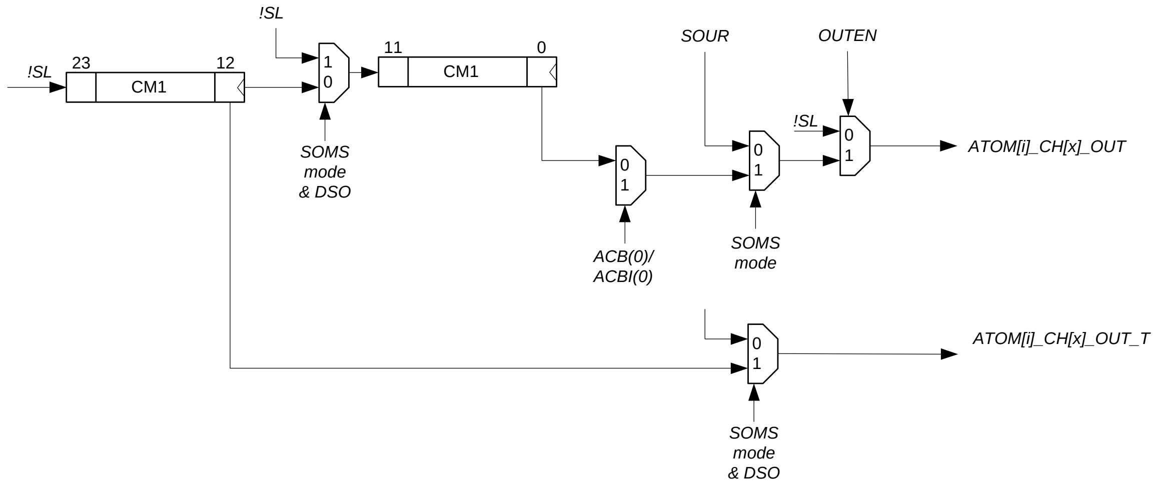
Then, the shifting of the channel can be restarted again by writing a zero to the **CN0** register again. Please note, that the **CN0** register should be written with a zero since the **CN0** register counts the number of bits shifted out by the ATOM channel.
The writing of a zero to **CN0** causes also an immediate update of **CM0**/**CM1** register with the content of **SR0/SR1** register.

13.3.4.8 SOMS mode with double output

If in SOMS mode additionally the mode bit DSO is set (in register **ATOM[i]_CH[x]_CTRL**) two 12 bit data streams can be shifted out on the outputs *ATOM_OUT* and *ATOM_OUT_T* in parallel.
This is reached by splitting the register **CM1** into two parts. The lower 12 bits are used as a shift register of output *ATOM_OUT* (i.e. bit 0 is assigned to output *ATOM_OUT*), the upper 12 bits are used as a shift register of output *ATOM_OUT_T* (i.e. bit 12 is assigned to output *ATOM_OUT_T*).
On bit 23 and 11 of register **CM1** the value !**SL** is shifted in.
Note: In this mode only shift right is possible. Bit **ACB0** is ignored.

This behavior is depicted in the following figure:

13.3.4.8.1 Double Output Shift Mode



13.3.4.9 Interrupts in SOMS mode

In ATOM Signal Output Mode Serial only the interrupt CCU0TC (**ATOM[i]_CH[x]_IRQ_NOTIFY**) in case of **CN0** >= **CM0** is generated. The interrupt **CCU1TC** has no meaning and is not generated.

13.3.4.9.1 Register ATOM[i]_CH[x]_CTRL in SOMS mode

Address Offset:	**see Appendix B**																**Initial Value:**					**0x0000_0x00**										
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	FREEZE	Not used	EXT_FUPD	Reserved	Not used	OSM	Not used	Not used	Not used			Not used	Not used		Not used	Not used	ECLK_SRC	CLK_SRC_SR			SL	Not used	Not used	Not used	DSO	Not used		ACB0	ARU_EN	Not used	MODE	
Mode	RW	RW	RW	R	RW	RW	RW	RW	R			RW	RW		RW	RW	RW	RW			RW	RW	RW	RW	RW	RW		RW	RW	RW	RW	
Initial Value	0	0	0	0b00	0	0	0	0	0b00			0	0b00		0	0	0	0b000			x	0	0	0	0	0b00		0	0	0	0b00	

Bit 1:0 **MODE**: ATOM channel mode select.
0b11: ATOM Signal Output Mode Serial (SOMS)

Bit 2 **Not used**
Note: Not used in this mode.

Bit 3 **ARU_EN** : ARU Input stream enable
0 = ARU Input stream disabled
1 = ARU Input stream enabled

Bit 4 **ACB0**: Shift direction for **CM1** register
0 = Right shift of data is started from bit 0 of CM1
1 = Left shift of data is started from bit 23 of CM1
Note: the data that has to be shifted out has to be aligned inside the CM1 register according to the defined shift direction.
Note: this bit is only applicable if ARU_EN = 0.
Note: if the direction (ACB0) is changed the output ATOM_OUT[x] switches immediately to the other 'first' bit of CM1 (bit 0 if ACB0 = 0, bit 23 if ACB0 = 1).

Bit 6:5 **Not used**
Note: Not used in this mode.

Bit 7 **DSO**: Double Shift Output
0 = CM1 is used as a 24 bit shift register
1 = CM1 is split into two 12 bit shift register
Note: if DSO=1, only shift right is possible

Bit 8 **Not used**
Note: Not used in this mode.

Bit 9 **Not used**
Note: Not used in this mode.

Bit 10 **Not used**
Note: Not used in this mode.

Bit 11 **SL**: Defines signal level when channel and output is disabling
0 = High signal level
1 = Low signal level
Note: Reset value depends on the hardware configuration chosen by silicon vendor.
Note: If the output is disabled, the output ATOM_OUT[x] is set to inverse value of SL.
Note: If the output is enabled, the output ATOM_OUT[x] is set to bit 0 or 23 of CM1 register.
Note: The inverse value of SL is shifted into the CM1 register.
Note: An enable or disable of the channel x has no effect on ATOM_OUT[x].

0 = Low signal level
1 = High signal level
Note: Reset value depends on the hardware configuration chosen by silicon vendor.
Note: If the output is enabled, the output ATOM_OUT[x] is set to bit 0 or 23 of CM1 register.
Note: The inverse value of SL is shifted into the CM1 register.
Note: If the output is disabled, the output ATOM_OUT[x] is set to inverse value of SL.
If FREEZE=0, the following note is valid

Note: If the channel is disabled, the output register of SOU unit is set to inverse value of SL.

If FREEZE=1, the following note is valid

Note: If the channel is disabled, the output register of SOU unit is not changed and output ATOM_OUT[x] is not changed.

Bit 14:12 **CLK_SRC**: Shift frequency select for channel

If ECLK_SRC=0 / ECLK_SRC=1:

0b000 = *CMU_CLK0* selected / *CMU_CLK0* selected
0b001 = *CMU_CLK1* selected / *CMU_CLK1* selected
0b010 = *CMU_CLK2* selected / *CMU_CLK2* selected
0b011 = *CMU_CLK3* selected / Reserved
0b100 = *CMU_CLK4* selected / clock stopped
0b101 = *CMU_CLK5* selected / *TRIG[x-1]* selected
0b110 = *CMU_CLK6* selected / *TIM_EXT_CAPTURE[x]* selected
0b111 = *CMU_CLK7* selected / *CMU_CLK7* selected

Note: This register is a shadow register for the register CLK_SRC. Thus, if the CMU_CLK source for PWM generation should be changed during operation, the old CMU_CLK has to operate until the update of the ATOM channels internal CLK_SRC register by the CLK_SRC_SR content is done either by an end of a period or a FORCE_UPDATE.

Note: After (channel) reset the selected CLK_SRC value is the SYS_CLK (input of Global Clock Divider). To use in SOMP mode one of the CMU_CLKx, it is recommended to perform a forced update of CLK_SRC with the value of CLK_SRC_SR value before/with enabling the channel.

Bit 15 **ECLK_SRC:** Extend CLK_SRC

0 = CLK_SRC_SR set 1 selected
1 = CLK_SRC_SR set 2 selected

See bit CLK_SRC_SR description for details.

Bit 16 **Not used**

Note: Not used in this mode.

Bit 17 **Not used**

Note: Not used in this mode.

Bit 19:18 **Not used**

Note: Not used in this mode.

Bit 20 **Not used**

Note: Not used in this mode.

Bit 23:21 **Not used**

Note: Not used in this mode.

Bit 24 **Not used**

Note: Not used in this mode.

Bit 25 **Not used**

Note: Not used in this mode.

Bit 26 **OSM:** One-shot mode
0 = Continuous shifting is enabled
1 = Channel stops, after number of bits defined in CM0 is shifted out

Bit 27 **Not used**
Note: Not used in this mode.

Bit 28 **Reserved**
Note: Read as zero, should be written as zero.

Bit 29 **EXT_FUPD:** external forced update
0 = use FUPD(x) signal from AGC to force update
1 = use TIM_EXT_CAPTURE signal to force update

Note: This bit is only applicable in SOMP and SOMS mode.

Bit 30 **Not used**
Note: Not used in this mode.

Bit 31 **FREEZE**
0 = a channel disable/enable may change internal register and output register
1 = a channel enable/disable does not change an internal or output register but stops counter CN0 (in SOMP mode), comparison (in SOMC/SOMB mode) and shifting (in SOMS mode)

13.3.5 ATOM Signal Output Mode Buffered Compare(SOMB)

13.3.5.1 Overview

In ATOM Signal Output Mode buffered Compare (SOMB) the output action is performed according to the comparison result of the input values located in **CM0** and/or **CM1** registers and the two (three) time base values *TBU_TS0* or *TBU_TS1* (or *TBU_TS2*) provided by the TBU. For a description of the time base generation please refer to the TBU specification in chapter 10. It is configurable, which of the two (three) time bases is to be compared with one or both values in **CM0** and **CM1**.

The compare strategy of the two compare units CCU0 and CCU1 is controlled by the value of bit field ACBI of register **ATOM[i]_CH[x]_STAT**. This bit field is only readable by CPU. If ARU is disabled, the bit field ACBI can only be updated with the value of bit field **ACB** of register **ATOM[i]_CH[x]_CTRL**. If ARU is enabled, the **ACBI** bit field can be updated with the value of shadow register **ACB_SR** which contains a value received via ARU or the value of bit field **ACB** of register **ATOM[i]_CH[x]_CTRL**.

The table below lists all valid control configurations for bit field **ACBI** of register **ATOM[i]_CH[x]_STAT**.

13.3.5.1.1 ATOM SOMB compare strategies

ACBI(4)	ACBI(3)	ACBI(2)	CCUx control
0	0	0	Reserved. Has no effect.
0	0	1	Reserved. Has no effect.
0	1	0	Compare in CCU0 only, use time base *TBU_TS0*. Output signal level is defined by combination of SL, ACB10/ACBI(1..0) bits.
0	1	1	Compare in CCU1 only, use time base *TBU_TS1* or *TBU_TS2*. Output signal level is defined by combination of SL, ACBI[1:0] bits.
1	0	0	Serve Last: Compare in CCU0 and then in CCU1 using *TBU_TS0*. Output signal level when CCU0 matches is defined by combination of SL, ACBI[1:0] . On the CCU1 match the output level is toggled.
1	0	1	Serve Last: Compare in CCU0 and then in CCU1 using *TBU_TS1* or *TBU_TS2*. Output signal level when CCU0 matches is defined by combination of SL, ACBI[1:0] . On the CCU1 match the output level is toggled.
1	1	0	Serve Last: Compare in CCU0 using *TBU_TS0* and then in CCU1 using *TBU_TS1* or *TBU_TS2*. Output signal level when CCU1 matches is defined by combination of SL, ACBI[1:0]
1	1	1	Cancels pending comparison independent on ARU_EN

The CCUx trigger signals *TRIG_CCU0* and *TRIG_CCU1* creates edges depending on the combination of the predefined signal level in **SL** bit and the two control bits **ACBI[1:0]**.

In SOMB mode, if ARU access is enabled, the new compare values received via ARU are always stored in the shadow register **SR0** and **SR1** and the **ACB** bits are stores in an internal register **ACB_SR**.

If the scheduled compare matches in CCU0 and/or CCU1 are occurred and the **SRx** register contain new valid values, the register **CM0** and **CM1** are updated automatically with the content of the corresponding **SRx** register, the **ACBI** bit field is updated with the content of internal **ACB_SR** register and the DV bit of register **ATOM[i]_CH[x]_STAT** is set. If the **SRx** register and the **CMx** register contain no valid value, the compare units are waiting in an idle state.

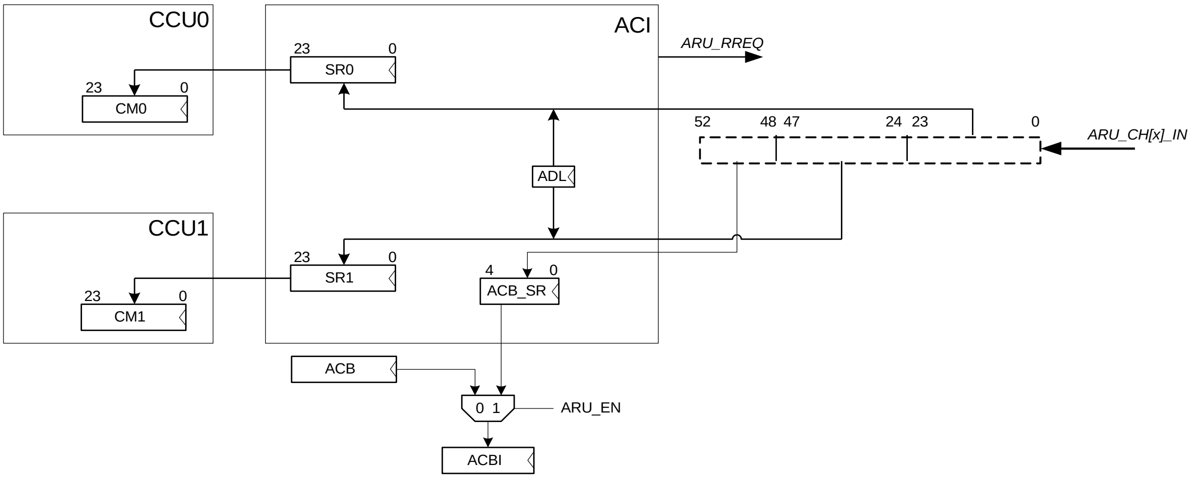
On a compare match of one of the compare units CCUx units the output ATOM_OUT is set according to combination of **ACBI** bit 1 down to 0 (in register **ATOM[i]_CH[x]_STAT**) and the **SL** bit of register **ATOM[i]_CH[x]_CTRL**.

13.3.5.1.2 ATOM SOMB output control by ACBI[1:0] and SL

SL	ACBI(1)	ACBI(0)	Output Behavior
0	0	0	No signal level change at output.
0	0	1	Set output signal level to 1.
0	1	0	Set output signal level to 0.
0	1	1	Toggle output signal level.
1	0	0	No signal level change at output.
1	0	1	Set output signal level to 0.
1	1	0	Set output signal level to 1.
1	1	1	Toggle output signal level.

In opposite to SOMC mode no time stamp value of TBU is captured in **SRx** register.

13.3.5.1.3 **ARU interface behavior in SOMB mode**



The flag **DV** of register **ATOM[i]_CH[x]_STAT** indicates that at least one of the **CMx** register contains valid data and a compare event may be pending (if channel is enabled).
The **DV** flag is reset if none of the **CMx** register contains valid data.

13.3.5.2 SOMB under CPU control

If bit **ARU_EN** of register **ATOM[i]_CH[x]_CTRL** is not set, the ATOM channel can only be controlled via CPU.
Writing to one of the **CMx** register sets automatically the **DV** bit to validate the new compare value. A comparison depending on value **ACBI** of register **ATOM[i]_CH[x]_STAT** is started immediately.
Because only the **ACB** bit of register **ATOM[i]_CH[x]_CTRL** can be written and this bit field serves as a shadow register for the work register **ACBI** (bit field of register **ATOM[i]_CH[x]_STAT)**, it is recommended to first update the **ACB** bit field before updating **CMx/SRx** register.

The compare strategy is controlled by the value stored in bit field **ACBI** of register **ATOM[i]_CH[x]_STAT**. If ARU is disabled, this bit field can only be updated with the value of bit field **ACB** of register **ATOM[i]_CH[x]_CTRL**.

The update of bit field **ACBI** can be triggered by a forced update or the normal update mechanism controlled by bit **UPEN_CTRL[x]** in register **ATOM[i]_AGC_GLB_CTRL**.

Writing to one of the **SRx** register and triggering a forced update, updates the **CMx** register with the value of **SRx** register and the **ACBI** bit field with the content of **ACB** bit field of register **ATOM[i]_CH[x]_CTRL**. A new comparison is started.

Writing to one of the **SRx** register while update of **CMx** register is disabled (**UPEN_CTRL[x]**= 0 in **ATOM[i]_AGC_GLB_CTRL**) and enabling update afterwards, triggers the update of **CMx** register and the **ACBI** bit field and starts comparison if previous comparison is finished (**DV** bit was reset).

If ARU access is disabled (**ARU_EN**=0), a force update updates the **CMx** register with the content of **SRx** register and the **ACBI** bit field with the content of **ACB** bit field of register **ATOM[i]_CH[x]_CTRL.**

13.3.5.3 SOMB under ARU control

If both compare units CCU0/CCU1 are finished with previous job (depending on compare strategy) and the **SRx** register contain no new value, they are waiting until new data was received via ARU and stored in **SRx** register. Then, an immediately update takes place.

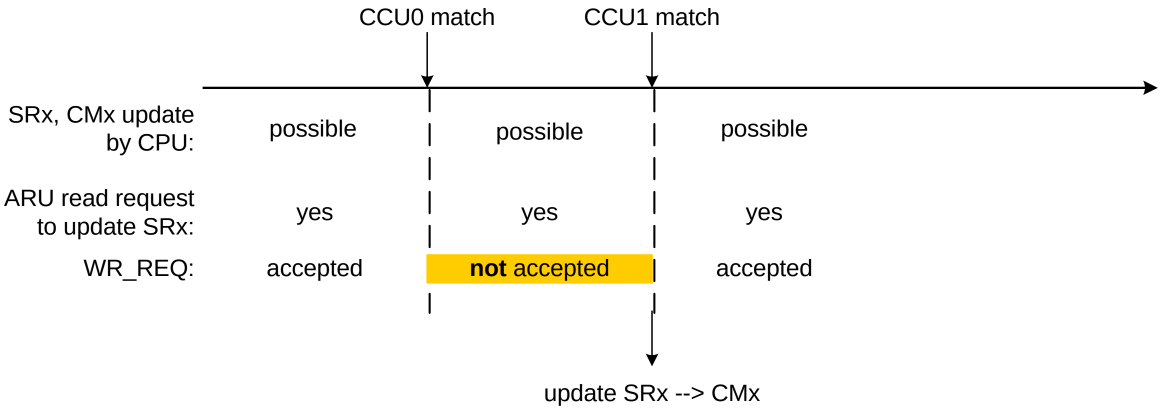
If both compare units are finished with previous job (depending on compare strategy) and there are new data available in **SRx** register, the update the **CMx** register with the value of the **SRx** register and the **ACBI** bit field with the value of internal **ACB_SR** register takes place and a new compare job is started immediately.

After an update of the **CMx** register, a new ARU read request is set.
New compare values received via the ARU are stored in shadow register **SRx**. The **ACB** bits received via ARU are stored in the internal register **ACB_SR**.

If ARU access is enabled (**ARU_EN**=1), a force update updates the **CMx** register with the content of **SRx** register and the **ACBI** bit field with the content of internal **ACB_SR** register.

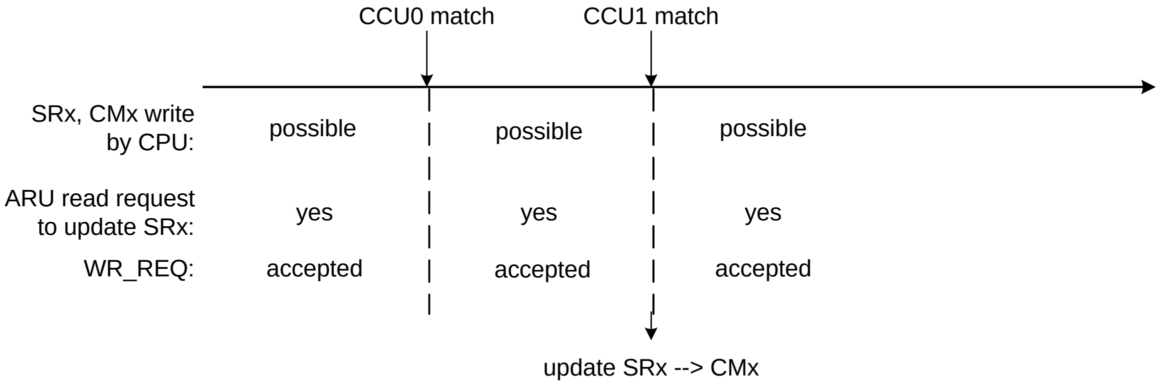
For compare strategy 'serve last' the CCU0 and CCU1 compare match may occur sequentially. During different phases of compare match the CPU access rights to register **CM0** and **CM1** as well as to **WR_REQ** bit is different. These access rights by CPU to register **CM0** and **CM1** and the **WR_REQ** are depicted in the following figure for the case of **EUPM**=0 (register **ATOM[i]_CH[x]_CTRL**)

13.3.5.3.1 CPU access rights in case of compare strategy 'serve last' and EUPM=0



The access rights by CPU to register **CM0** and **CM1** and the **WR_REQ** are depicted in the following figure for the case of **EUPM**=1 (register **ATOM[i]_CH[x]_CTRL**)

13.3.5.3.2 CPU access rights in case of compare strategy 'serve last' in case of EUPM=1



13.3.5.3.3 ARU Non-blocking mode

If bit **ABM** in register **ATOM[i]_CH[x]_CTRL** is not set, the ARU blocking mode is disabled. In this case the ATOM channel is continuously reading via ARU and storing new values in the **SRx** register and the ACB shadow register **ACB_SR**.

If **ARU_EN** is not set, the bit **ABM** has no meaning.

13.3.5.3.4 ARU Blocking mode

If bit **ABM** in register **ATOM[i]_CH[x]_CTRL** is set, the ARU blocking mode is enabled. In this case the ATOM channel stops requesting new **SRx** values via ARU after reception of a new **SRx** value and restarts requesting a new value via ARU after compare match on both compare units (depending on compare strategy) followed by the immediate update of the **CMx** register with content of **SRx** register and an update of **ACBI** with the content of **ACB_SR**.

If **ARU_EN** is not set, the bit **ABM** has no meaning.

13.3.5.3.5 Late Update by CPU

Although, the ATOM channel may be controlled by data received via the ARU, the CPU is able to request at any time a late update of the compare register. This can be initiated by setting the **WR_REQ** bit inside the **ATOM[i]_CH[x]_CTRL** register.

If none of the two compare match event happened, the ATOM channel accepts the setting of **WR_REQ** bit. In this case, the ATOM will request no further data from ARU (if ARU access was enabled) and will disable the update of **CMx** register with the content of **SRx** register on a compare match event.

If at least one of the requested compare match events happened (depending on strategy) the **WR_REQ** bit is not set and the **WRF** flag in register **ATOM[i]_CH[x]_STAT** is set to indicate that the late update was not successful.

The channel will in any case continue to compare against the values stored inside the compare registers (if bit **DV** was set). The CPU can now update the compare values by writing to the shadow registers and force the ATOM channel to update the compare registers by writing to the force update register bits in the AGC register.

With a force update the **WR_REQ** bit is reset automatically and the ARU read request is set up again (if ARU access was enabled).

13.3.5.3.6 Register ATOM[i]_CH[x]_CTRL in SOMB mode

Address Offset:	**see Appendix B**																**Initial Value:**					**0x0000_0x00**										
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	FREEZE	SOMB	Not used	Reserved	ABM	Not used	Not used	TRIGOUT	EXTTRIGOUT	Not used		Not used	Not used		Not used	WR_REQ	Not used	Not used			SL	EUPM	CMP_CTRL	ACB[4:2]			ACB[1:0]		ARU_EN	TB12_SEL	MODE	
Mode	RW	RW	RW	R	RW	RW	RW	RW	RW	R		RW	RW		RW	RW	RW	RW			RW	RW	RW	RW			RW		RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	0	0b00		0	0b00		0	0	0	0b000			x	0	0	0b000			0b00		0	0	0b00	

Bit 1:0 **MODE**: ATOM channel mode select.
Not used in ATOM SOMB mode.

Bit 2 **TB12_SEL**: Select time base value *TBU_TS1* or *TBU_TS2*.
0 = *TBU_TS1* selected for comparison
1 = *TBU_TS2* selected for comparison
Note: This bit is only applicable if three time bases are present in the GTM-IP. Otherwise, this bit is reserved.

Bit 3 **ARU_EN**: ARU Input stream enable.
0 = ARU Input stream disabled
1 = ARU Input stream enabled

Bit 5:4 **ACB[1:0]**: Signal level control bits.
For details see 13.3.5.1.2
0b00 = No signal level change at output.
0b01 = Set output signal level to 1 when SL bit = 0 else output signal level to 0.
0b10 = Set output signal level to 0 when SL bit = 0 else output signal level to 1.
0b11 = Toggle output signal level.
Note: These bits are only applicable if ARU_EN = 0.

Bit 8:6 **ACB[4:2]**: ATOM SOMB compare strategy
For details see 13.3.5.1.1
0b000 = Reserved. Has no effect.
0b001 = Reserved. Has no effect.
0b010 = Compare in CCU0 only against *TBU_TS0*.
0b011 = Compare in CCU1 only against *TBU_TS1* or *TBU_TS2*.
0b100 = Compare first in CCU0 and then in CCU1. Use *TBU_TS0*.

0b101 = Compare first in CCU0 and then in CCU1. Use *TBU_TS1* or *TBU_TS2*.
0b110 = Compare first in CCU0 and then in CCU1. Use *TBU_TS0* in CCU0 and *TBU_TS1* or *TBU_TS2* in CCU1.
0b111 = Cancel pending comparisons independent on ARU_EN.
Note: These bits are only applicable if ARU_EN = 0.

Bit 9 **CMP_CTRL**: CCUx compare strategy select.
0 = Greater-equal compare against TBU time base values (TBU_TS1/2 >= CM0/1)
1 = Less-equal compare against TBU time base values (TBU_TS1/2 <= CM0/1)

Note: The compare unit CCU0 or CCU1 that compares against TBU_TS0 (depending on CCUx control mode defined by ACB_CM(4:2)) always performs a greater-equal comparison, independent on CMP_CTRL bit.

Bit 10 **EUPM:** Extended update mode
0 = No extended update of CM0 and CM1 via CPU or ARU
1 = Extended update mode in case of compare strategy 'serve last': update of CM1 after CCU0 compare match possible via ARU or CPU.
Note: If EUPM=1 a write access to CM0 or CM1 never causes an AEI write status 0b10.
Note: this bit is only applicable in SOMC and SOMB mode.

Bit 11 **SL**: Initial signal level after channel enable.
0 = Low signal level
1 = High signal level
Note: Reset value depends on the hardware configuration chosen by silicon vendor.
Note: If the output is disabled, the output ATOM_OUT[x] is set to inverse value of SL.
If FREEZE=0, the following notes is valid
Note: If the channel is disabled, the output register of SOU unit is set to value of SL.
If FREEZE=1, the following note is valid
Note: If the channel is disabled, the output register of SOU unit is not changed and output ATOM_OUT[x] is not changed.

Bit 14:12 **Not used**
Note: Not used in this mode.

Bit 15 **Not used**
Note: Not used in this mode.

Bit 16 **WR_REQ**: CPU Write request bit for late compare register update.
0 = No late update requested by CPU

1 = Late update requested by CPU

Note: The CPU can disable subsequent ARU read requests by the channel and can update the shadow registers with new compare values, while the compare units operate on old compare values received by former ARU accesses, if occurred.

Note: On a compare match event, the WR_REQ bit will be reset by hardware.

Note: At the point of the force update only the shadow registers SR0 and SR1 are transferred into the CM0, CM1 registers. The output action is still defined by the ACBI bit field described by the ARU together with the old compare values for CM0/CM1.

Bit 17 **Not used**

Note: Not used in this mode.

Bit 19:18 **Not used**

Note: Not used in this mode.

Bit 20 **Not used** : not used in this mode

Note: Not used in this mode.

Bit 22:21 **Not used** : not used in this mode

Note: Not used in this mode.

Bit 23 **EXTTRIGOUT**: select *TIM_EXT_CAPTURE(x)* as potential output signal *TRIG_[x]*

0 = signal *TRIG_[x-1]* is selected as output on *TRIG_[x]* (if TRIGOUT=1)

1 = signal *TIM_EXT_CAPTURE(x)* is selected as output on *TRIG_[x]* (if TRIGOUT=1)

Bit 24 **TRIGOUT**: Trigger output selection (output signal *TRIG_CHx*) of module ATOM_CHx.

0 = *TRIG_[x]* is *TRIG_[x-1]* or TIM_*EXT_CAPTURE(x)*.

1 = *TRIG_[x]* is *TRIG_CCU0*

Bit 25 **Not used** : not used in this mode

Note: Not used in this mode.

Bit 26 **Not used** : not used in this mode

Note: Not used in this mode.

Bit 27 **ABM:** ARU blocking mode

0 = ARU blocking mode disabled: ATOM reads continuously from ARU and updates SR0, SR1 and ACB bits independent of pending compare match event.

1 = ARU blocking mode enabled: after update of SR0, SR1 and ACB bits via ARU, no new data is read via ARU until compare match event occurred.

Bit 28 **Reserved**

Note: Read as zero, should be written as zero.

Bit 29 **Not used** : not used in this mode

Note: Not used in this mode.

Bit 30 **SOMB**: SOMB mode

0 = ATOM channel mode defined by bit filed **MODE**
1 = ATOM SOMB mode enabled

Bit 31 **FREEZE**
0 = a channel disable/enable may change internal register and output register
1 = a channel enable/disable does not change an internal or output register but stops counter CN0 (in SOMP mode), comparison (in SOMC/SOMB mode) and shifting (in SOMS mode)

13.4 ATOM Interrupt Signals

Signal	**Description**
CCU0TCx_IRQ	CCU0 Trigger condition interrupt for channel x
CCU1TCx_IRQ	CCU1 Trigger condition interrupt for channel x

13.5 ATOM Register Overview

Register name	**Description**	**Details in Section**
ATOM[i]_AGC_GLB_CTRL	ATOMi AGC global control register	13.6.1
ATOM[i]_AGC_ENDIS_CTRL	ATOMi AGC enable/disable control register	13.6.2
ATOM[i]_AGC_ENDIS_STAT	ATOMi AGC enable/disable status register	13.6.3
ATOM[i]_AGC_ACT_TB	ATOMi AGC action time base register	13.6.4
ATOM[i]_AGC_OUTEN_CTRL	ATOMi AGC output enable control register	13.6.5
ATOM[i]_AGC_OUTEN_STAT	ATOMi AGC output enable status register	13.6.6
ATOM[i]_AGC_FUPD_CTRL	ATOMi AGC force update control register	13.6.7
ATOM[i]_AGC_INT_TRIG	ATOMi AGC internal trigger control register	13.6.8
ATOM[i]_CH[x]_CTRL	ATOMi channel x control register	13.6.9
ATOM[i]_CH[x]_STAT	ATOMi channel x status register	13.6.10
ATOM[i]_CH[x]_RDADDR	ATOMi channel x ARU read address register	13.6.11

ATOM[i]_CH[x]_CN0	ATOMi channel x CCU0 counter register	13.6.12
ATOM[i]_CH[x]_CM0	ATOMi channel x CCU0 compare register	13.6.13
ATOM[i]_CH[x]_SR0	ATOMi channel x CCU0 compare shadow register	13.6.14
ATOM[i]_CH[x]_CM1	ATOMi channel x CCU1 compare register	13.6.15
ATOM[i]_CH[x]_SR1	ATOMi channel x CCU1 compare shadow register	13.6.16
ATOM[i]_CH[x]_IRQ_NOTIFY	ATOMi channel x interrupt notification register	13.6.17
ATOM[i]_CH[x]_IRQ_EN	ATOMi channel x interrupt enable register	13.6.18
ATOM[i]_CH[x]_IRQ_FORCINT	ATOMi channel x software interrupt generation	13.6.19
ATOM[i]_CH[x]_IRQ_MODE	ATOMi channel x interrupt mode configuration register	13.6.20

13.6 ATOM Register Description

13.6.1 Register ATOM[i]_AGC_GLB_CTRL

Address Offset:	see Appendix B								Initial Value:						0x0000_0000		
	31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16	15	14	13	12	11	10	9	8	7 6 5 4 3 2 1	0
Bit	UPEN_CTRL7	UPEN_CTRL6	UPEN_CTRL5	UPEN_CTRL4	UPEN_CTRL3	UPEN_CTRL2	UPEN_CTRL1	UPEN_CTRL0	RST_CH7	RST_CH6	RST_CH5	RST_CH4	RST_CH3	RST_CH2	RST_CH1	RST_CH0	Reserved	HOST_TRIG
Mode	RW	RW	RW	RW	RW	RW	RW	RW	RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw	R	RAw
Initial Value	0b00	0b00	0b00	0b00	0b00	0b00	0b00	0b00	0	0	0	0	0	0	0	0	0x00	0

Bit 0 **HOST_TRIG** : trigger request signal (see AGC) to update the register ENDIS_STAT and OUTEN_STAT
0 = no trigger request
1 = set trigger request
Note: this flag is reset automatically after triggering the update

Bit 7:1 **Reserved**
Note: Read as zero, should be written as zero

Bit 8 **RST_CH0** : Software reset of channel 0
0 = No action
1 = Reset channel
Note: This bit is cleared automatically after write by CPU. The channel registers are set to their reset values and channel operation is stopped immediately. The output register of SOU unit is reset to inverse reset value of SL bit.

Bit 9 **RST_CH1** : Software reset of channel 1
See bit 8

Bit 10 **RST_CH2** : Software reset of channel 2
See bit 8

Bit 11 **RST_CH3** : Software reset of channel 3
See bit 8

Bit 12 **RST_CH4** : Software reset of channel 4
See bit 8

Bit 13 **RST_CH5** : Software reset of channel 5
See bit 8

Bit 14 **RST_CH6** : Software reset of channel 6
See bit 8

Bit 15 **RST_CH7** : Software reset of channel 7
See bit 8

Bit 17:16 **UPEN_CTRL0**: ATOM channel 0 enable update of register CM0, CM1 and CLK_SRC from SR0, SR1 and CLK_SRC_SR.
Write / Read :
0b00 = don't care, bits 1:0 will not be change / update disabled
0b01 = disable update / --
0b10 = enable update / --
0b11 = don't care, bits 1:0 will not be changed / update enabled

Bit 19:18 **UPEN_CTRL1**: ATOM channel 1 enable update of register CM0, CM1 and CLK_SRC
See bits 17:16

Bit 21:20 **UPEN_CTRL2**: ATOM channel 2 enable update of register CM0, CM1 and CLK_SRC
See bits 17:16

Bit 23:22 **UPEN_CTRL3**: ATOM channel 3 enable update of register CM0, CM1 and CLK_SRC
See bits 17:16

Bit 25:24 **UPEN_CTRL4**: ATOM channel 4 enable update of register CM0, CM1 and CLK_SRC
See bits 17:16

Bit 27:26 **UPEN_CTRL5**: ATOM channel 5 enable update of register CM0, CM1 and CLK_SRC

See bits 17:16

Bit 29:28 **UPEN_CTRL6**: ATOM channel 6 enable update of register CM0, CM1 and CLK_SRC
See bits 17:16

Bit 31:30 **UPEN_CTRL7**: ATOM channel 7 enable update of register CM0, CM1 and CLK_SRC
See bits 17:16

13.6.2 Register ATOM[i]_AGC_ENDIS_CTRL

Address Offset:	**see Appendix B**																**Initial Value: 0x0000_0000**															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																ENDIS_CTRL7		ENDIS_CTRL6		ENDIS_CTRL5		ENDIS_CTRL4		ENDIS_CTRL3		ENDIS_CTRL2		ENDIS_CTRL1		ENDIS_CTRL0	
Mode	R																RW		RW		RW		RW		RW		RW		RW		RW	
Initial Value	0x0000																0b00		0b00		0b00		0b00		0b00		0b00		0b00		0b00	

Bit 1:0 **ENDIS_CTRL0**: ATOM channel 0 enable/disable update value.

If FREEZE=0:

If an ATOM channel is disabled, the counter **CN0** is stopped and the output register of SOU unit is set to the inverse value of control bit SL. On an enable event, the counter **CN0** starts counting from its current value.

If FREEZE=1:

If an ATOM channel is disabled, the counter **CN0** is stopped (SOMP, SOMS mode) and each comparison is stopped (SOMC, SOMB mode). On an enable event, the counter **CN0** starts counting from its current value or a comparison is restarted.

Write of following double bit values is possible:

0b00 = don't care, bits 1:0 of register ENDIS_STAT will not be changed on an update trigger

0b01 = disable channel on an update trigger

0b10 = enable channel on an update trigger

0b11 = don't change bits 1:0 of this register

Note: if the output is disabled (OUTEN[0]=0), the ATOM channel 0 output ATOM_OUT[0] is the inverted value of bit SL.

Bit 3:2 **ENDIS_CTRL1**: ATOM channel 1 enable/disable update value.
See bits 1:0

Bit 5:4 **ENDIS_CTRL2**: ATOM channel 2 enable/disable update value.
See bits 1:0

Bit 7:6 **ENDIS_CTRL3**: ATOM channel 3 enable/disable update value.
See bits 1:0

Bit 9:8 **ENDIS_CTRL4**: ATOM channel 4 enable/disable update value.
See bits 1:0

Bit 11:10 **ENDIS_CTRL5**: ATOM channel 5 enable/disable update value.
See bits 1:0

Bit 13:12 **ENDIS_CTRL6**: ATOM channel 6 enable/disable update value.
See bits 1:0

Bit 15:14 **ENDIS_CTRL7**: ATOM channel 7 enable/disable update value.
See bits 1:0

Bit 31:16 **Reserved**
Note: Read as zero, should be written as zero

13.6.3 Register ATOM[i]_AGC_ENDIS_STAT

Address Offset:	**see Appendix B**																**Initial Value: 0x0000_0000**															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																ENDIS_STAT7		ENDIS_STAT6		ENDIS_STAT5		ENDIS_STAT4		ENDIS_STAT3		ENDIS_STAT2		ENDIS_STAT1		ENDIS_STAT0	
Mode	R																RW		RW		RW		RW		RW		RW		RW		RW	
Initial Value	0x0000																0b00		0b00		0b00		0b00		0b00		0b00		0b00		0b00	

Bit 1:0 **ENDIS_STAT0**: ATOM channel 0 enable/disable
If FREEZE=0:
If an ATOM channel is disabled, the counter CN0 is stopped and the output register of SOU unit is set to the inverse value of control bit SL. On an enable event, the counter CN0 starts counting from its current value.
If FREEZE=1:
If an ATOM channel is disabled, the counter CN0 is stopped (SOMP, SOMS mode) and each comparison is stopped (SOMC, SOMB mode). On an enable event, the counter CN0 starts counting from its current value or a comparison is restarted.

Write of following double bit values is possible:

0b00 = don't care, bits 1:0 of register ENDIS_STAT will not be changed on an update trigger
0b01 = disable channel on an update trigger
0b10 = enable channel on an update trigger
0b11 = don't change bits 1:0 of this register

Bit 3:2 **ENDIS_STAT1**: ATOM channel 1 enable/disable
See bits 1:0

Bit 5:4 **ENDIS_STAT2**: ATOM channel 2 enable/disable
See bits 1:0

Bit 7:6 **ENDIS_STAT3**: ATOM channel 3 enable/disable
See bits 1:0

Bit 9:8 **ENDIS_STAT4**: ATOM channel 4 enable/disable
See bits 1:0

Bit 11:10 **ENDIS_STAT5**: ATOM channel 5 enable/disable
See bits 1:0

Bit 13:12 **ENDIS_STAT6**: ATOM channel 6 enable/disable
See bits 1:0

Bit 15:14 **ENDIS_STAT7**: ATOM channel 7 enable/disable
See bits 1:0

Bit 31:16 **Reserved**
Note: Read as zero, should be written as zero

13.6.4 Register ATOM[i]_AGC_ACT_TB

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_0000**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved					TBU_SEL		TB_TRIG	ACT_TB																							
Mode	R					RW		RAw	RW																							
Initial Value	0b0000					0b00		0	0x00_0000																							

Bit 23:0 **ACT_TB**: specifies the signed compare value with selected signal TB*U_TS*[x], x=0..2. If selected *TBU_TS*[x] value is in the interval [**ACT_TB**-007FFFFFh,**ACT_TB**] the event is in the past and the trigger is generated immediately. Otherwise the event is in the future and the trigger is generated if selected *TBU_TS*[x] is equal to **ACT_TB**.

Bit 24 **TB_TRIG**: Set trigger request
0 = no trigger request

1 = set trigger request
Note: This flag is reset automatically if the selected time base unit (*TBU_TS0* or *TBU_TS1* or *TBU_TS2* if present) has reached the value **ACT_TB** and the update of the register were triggered.

Bit 26:25 **TBU_SEL**: Selection of time base used for comparison
0b00 = *TBU_TS0* selected
0b01 = *TBU_TS1* selected
0b10 = *TBU_TS2* selected
0b11 = same as 0b00
Note: The bit combination 0b10 is only applicable if the TBU of the device contains three time base channels. Otherwise, this bit combination is also reserved. Please refer to GTM Architecture block diagram on page 3 to determine the number of channels for TBU of this device.

Bit 31:27 **Reserved**
Note: Read as zero, should be written as zero

13.6.5 Register ATOM[i]_AGC_OUTEN_CTRL

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																OUTEN_CTRL7		OUTEN_CTRL6		OUTEN_CTRL5		OUTEN_CTRL4		OUTEN_CTRL3		OUTEN_CTRL2		OUTEN_CTRL1		OUTEN_CTRL0	
Mode	R																RW		RW		RW		RW		RW		RW		RW		RW	
Initial Value	0x0000																0b00		0b00		0b00		0b00		0b00		0b00		0b00		0b00	

Bit 1:0 **OUTEN_CTRL0**: Output ATOM_OUT(0) enable/disable update value
Write of following double bit values is possible:
0b00 = don't care, bits 1:0 of register OUTEN_STAT will not be changed on an update trigger
0b01 = disable channel output on an update trigger
0b10 = enable channel output on an update trigger
0b11 = don't change bits 1:0 of this register

Note: if the channel is disabled (ENDIS[0]=0) or the output is disabled (OUTEN[0]=0), the TOM channel 0 output ATOM_OUT[0] is the inverted value of bit SL.

Bit 3:2 **OUTEN_CTRL1**: Output ATOM_OUT(1) enable/disable update value
See bits 1:0

Bit 5:4 **OUTEN_CTRL2**: Output ATOM_OUT(2) enable/disable update value
See bits 1:0

Bit 7:6 **OUTEN_CTRL3**: Output ATOM_OUT(3) enable/disable update value
See bits 1:0

Bit 9:8 **OUTEN_CTRL4**: Output ATOM_OUT(4) enable/disable update value
See bits 1:0

Bit 11:10 **OUTEN_CTRL5**: Output ATOM_OUT(5) enable/disable update value
See bits 1:0

Bit 13:12 **OUTEN_CTRL6**: Output ATOM_OUT(6) enable/disable update value
See bits 1:0

Bit 15:14 **OUTEN_CTRL7**: Output ATOM_OUT(7) enable/disable update value
See bits 1:0

Bit 31:16 **Reserved**
Note: Read as zero, should be written as zero

13.6.6 Register ATOM[i]_AGC_OUTEN_STAT

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																OUTEN_STAT7		OUTEN_STAT6		OUTEN_STAT5		OUTEN_STAT4		OUTEN_STAT3		OUTEN_STAT2		OUTEN_STAT1		OUTEN_STAT0	
Mode	R																RW		RW		RW		RW		RW		RW		RW		RW	
Initial Value	0x0000																0b00		0b00		0b00		0b00		0b00		0b00		0b00		0b00	

Bit 1:0 **OUTEN_STAT0**: Control/status of output ATOM_OUT(0)
Write / Read :
0b00 = don't care, bits 1:0 will not be changed / output disabled
0b01 = disable output / --
0b10 = enable output / --
0b11 = don't care, bits 1:0 will not be changed / output enabled

Bit 3:2 **OUTEN_STAT1**: Control/status of output ATOM_OUT(1)
See bits 1:0

Bit 5:4 **OUTEN_STAT2**: Control/status of output ATOM_OUT(2)
See bits 1:0

Bit 7:6 **OUTEN_STAT3**: Control/status of output ATOM_OUT(3)

See bits 1:0

Bit 9:8 **OUTEN_STAT4**: Control/status of output ATOM_OUT(4)
See bits 1:0

Bit 11:10 **OUTEN_STAT5**: Control/status of output ATOM_OUT(5)
See bits 1:0

Bit 13:12 **OUTEN_STAT6**: Control/status of output ATOM_OUT(6)
See bits 1:0

Bit 15:14 **OUTEN_STAT7**: Control/status of output ATOM_OUT(7)
See bits 1:0

Bit 31:16 **Reserved**
Note: Read as zero, should be written as zero

13.6.7 Register ATOM[i]_AGC_FUPD_CTRL

Address Offset:	**see Appendix B**																**Initial Value:**				**0x0000_0000**											
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	RSTCN0_CH7		RSTCN0_CH6		RSTCN0_CH5		RSTCN0_CH4		RSTCN0_CH3		RSTCN0_CH2		RSTCN0_CH1		RSTCN0_CH0		FUPD_CTRL7		FUPD_CTRL6		FUPD_CTRL5		FUPD_CTRL4		FUPD_CTRL3		FUPD_CTRL2		FUPD_CTRL1		FUPD_CTRL0	
Mode	RW		RW		RW		RW		RW		RW		RW		RW		RW		RW		RW		RW		RW		RW		RW		RW	
Initial Value	0b00		0b00		0b00		0b00		0b00		0b00		0b00		0b00		0b00		0b00		0b00		0b00		0b00		0b00		0b00		0b00	

Bit 1:0 **FUPD_CTRL0**: Force update of ATOM channel 0 operation registers
If enabled, force update of register CM0, CM1 and CLK_SRC triggered by HOST_TRIG, ACT_TB compare match or internal trigger.
Write / Read :
0b00 = don't care, bits 1:0 will not be changed / force update disabled
0b01 = disable force update / --
0b10 = enable force update / --
0b11 = don't care, bits 1:0 will not be changed / force update enabled
Note: In SOMP mode the force update request is stored and executed synchronized to the selected CMU_CLK. In all other modes the force update request is executed immediately.
Note: In SOMP mode, in case of ECLK_SRC=1 and CLK_SRC_SR = 0b011/0b100/0b101/0b110 a force update leads to an immediate update of CM0, CM1 and CLK_SRC.

Bit 3:2 **FUPD_CTRL1**: Force update of ATOM channel 1 operation registers
See bits 1:0

Bit 5:4 **FUPD_CTRL2**: Force update of ATOM channel 2 operation registers

See bits 1:0

Bit 7:6 **FUPD_CTRL3**: Force update of ATOM channel 3 operation registers
See bits 1:0

Bit 9:8 **FUPD_CTRL4**: Force update of ATOM channel 4 operation registers
See bits 1:0

Bit 11:10 **FUPD_CTRL5**: Force update of ATOM channel 5 operation registers
See bits 1:0

Bit 13:12 **FUPD_CTRL6**: Force update of ATOM channel 6 operation registers
See bits 1:0

Bit 15:14 **FUPD_CTRL7**: Force update of ATOM channel 7 operation registers
See bits 1:0

Bit 17:16 **RSTCN0_CH0**: Reset CN0 of channel 0 on force update event
If enabled, reset CN0 triggered by HOST_TRIG, ACT_TB compare match or internal trigger.
Write / Read :
0b00 = don't care, bits 1:0 will not be changed / CN0 is not reset on forced update
0b01 = do not reset CN0 on forced update / --
0b10 = reset CN0 on forced update / --
0b11 = don't care, bits 1:0 will not be changed / CN0 is reset on forced update

Bit 19:18 **RSTCN0_CH1**: Reset CN0 of channel 1 on force update event
See bits 17:16

Bit 21:20 **RSTCN0_CH2**: Reset CN0 of channel 2 on force update event
See bits 17:16

Bit 23:22 **RSTCN0_CH3**: Reset CN0 of channel 3 on force update event
See bits 17:16

Bit 25:24 **RSTCN0_CH4**: Reset CN0 of channel 4 on force update event
See bits 17:16

Bit 27:26 **RSTCN0_CH5**: Reset CN0 of channel 5 on force update event
See bits 17:16

Bit 29:28 **RSTCN0_CH6**: Reset CN0 of channel 6 on force update event
See bits 17:16

Bit 31:30 **RSTCN0_CH7**: Reset CN0 of channel 7 on force update event
See bits 17:16

13.6.8 Register ATOM[i]_AGC_INT_TRIG

Address Offset:	**see Appendix B**																**Initial Value: 0x0000_0000**															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																INT_TRIG7		INT_TRIG6		INT_TRIG5		INT_TRIG4		INT_TRIG3		INT_TRIG2		INT_TRIG1		INT_TRIG0	
Mode	R																RW		RW		RW		RW		RW		RW		RW		RW	
Initial Value	0x0000																0b00		0b00		0b00		0b00		0b00		0b00		0b00		0b00	

Bit 1:0 **INT_TRIG0**: Select input signal *TRIG_0* as a trigger source

Write / Read :

0b00 = don't care, bits 1:0 will not be changed / internal trigger from channel 0 (TRIG_0) not used

0b01 = do not use internal trigger from channel 0 (TRIG_0) / --

0b10 = use internal trigger from channel 0 (TRIG_0) / --

0b11 = don't care, bits 1:0 will not be changed / internal trigger from channel 0 (TRIG_0) used

Bit 3:2 **INT_TRIG1**: Select input signal *TRIG_1* as a trigger source

See bits 1:0

Bit 5:4 **INT_TRIG2**: Select input signal *TRIG_2* as a trigger source

See bits 1:0

Bit 7:6 **INT_TRIG3**: Select input signal *TRIG_3* as a trigger source

See bits 1:0

Bit 9:8 **INT_TRIG4**: Select input signal *TRIG_4* as a trigger source

See bits 1:0

Bit 11:10 **INT_TRIG5**: Select input signal *TRIG_5* as a trigger source

See bits 1:0

Bit 13:12 **INT_TRIG6**: Select input signal *TRIG_6* as a trigger source

See bits 1:0

Bit 15:14 **INT_TRIG7**: Select input signal *TRIG_7* as a trigger source

See bits 1:0

Bit 31:16 **Reserved**

Note: Read as zero, should be written as zero

13.6.9 Register ATOM[i]_CH[x]_CTRL

Address Offset:	see Appendix B																Initial Value:					0x0000_0X00										
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	FREEZE	SOMB	EXT_FUPD	Reserved	ABM	OSM	SLA	TRIGOUT	EXTTRIGOUT	EXT_TRIG	OSM_TRIG	RST_CCU0	UDMODE		TRIG_PULSE	WR_REQ	ECLK_SRC	CLK_SRC_SR			SL	EUPM	CMP_CTRL	ACB					ARU_EN	TB12_SEL	MODE	
Mode	RW	RW	RW	R	RW	RW	RW	RW	RW	RW	RPw	RW	RW		RW	RW	RW	RW			RW	RW	RW	RW					RW	RW	RW	
Initial Value	0	0	0	0	0	0	0	0	0	0	0	0	0b00		0	0	0	0b000			x	0	0	0b00000					0	0	0b00	

Bit 1:0 **MODE**: ATOM channel mode select.
0b00 = ATOM Signal Output Mode Immediate (SOMI)
0b01 = ATOM Signal Output Mode Compare (SOMC)
0b10 = ATOM Signal Output Mode PWM (SOMP)
0b11 = ATOM Signal Output Mode Serial (SOMS)

Bit 2 **TB12_SEL**: Select time base value *TBU_TS1* or *TBU_TS2*.
0 = *TBU_TS1* selected for comparison
1 = *TBU_TS2* selected for comparison
Note: this bit is only applicable in SOMC mode.

Bit 3 **ARU_EN**: ARU Input stream enable.
0 = ARU Input stream disabled
1 = ARU Input stream enabled

Bit 8:4 **ACB**: ATOM Mode control bits.
Note: These bits have different meaning in the different ATOM channel modes. Please refer to the mode description sections.
SOMI : 13.3.1 and 13.3.1.2 for register description
SOMC : 13.3.2 and 13.3.2.3.11 for register description
SOMP : 13.3.3 and 13.3.3.8.1 for register description
SOMS : 13.3.4 and 13.3.4.9.1 for register description
SOMB : 13.3.5 and 13.3.5.3.6 for register description

Bit 9 **CMP_CTRL**: CCUx compare strategy select.
0 = Greater-equal compare against TBU time base values (TBU_TSx >= CMx)
1 = Less-equal compare against TBU time base values (TBU_TSx <= CMx)
Note: this bit is only applicable in SOMC mode.

Bit 10 **EUPM:** Extended update mode
0 = No extended update of CM0 and CM1 via CPU or ARU

1 = Extended update mode in case of compare strategy 'serve last': update of CM1 after CCU0 compare match possible via ARU or CPU.

Note: If EUPM=1 a write access to CM0 or CM1 never causes an AEI write status 0b10.

Note: this bit is only applicable in SOMC and SOMB mode.

Bit 11 **SL**: Initial signal level.

0 = Low signal level

1 = High signal level

Note: Reset value depends on the hardware configuration chosen by silicon vendor.

Note: If the output is disabled, the output ATOM_OUT[x] is set to inverse SL independent of the ATOM channel mode.

If FREEZE=0, following notes are valid:

Note: In SOMP, SOMI, SOMS mode, if the channel is disabled, the internal register SOUR inside ATOM sub unit SOU is set to inverse value of SL. By enabling the channel the register SOUR is not changed. Thus, if the output is enabled afterwards, the output ATOM_OUT[x] is the inverse value of SL.

Note: In SOMC mode, if the channel is disabled, the internal register SOUR inside ATOM sub unit SOU is set to value of SL. By enabling the channel the register SOUR is not changed. Thus, if the output is enabled and the channel is disabled, the output ATOM_OUT[x] is the value of SL.

If FREEZE=1, the following notes are valid:

Note: If the channel is disabled, the output register of SOU unit is not changed and output ATOM_OUT[x] is not changed.

Bit 14:12 **CLK_SRC/CLK_SRC_SR**: actual CMU clock source (SOMS)/ shadow register for CMU clock source (SOMP).

If ECLK_SRC=0 / ECLK_SRC=1:

0b000 = *CMU_CLK0* selected / *CMU_CLK0* selected

0b001 = *CMU_CLK1* selected / *CMU_CLK1* selected

0b010 = *CMU_CLK2* selected / *CMU_CLK2* selected

0b011 = *CMU_CLK3* selected / Reserved

0b100 = *CMU_CLK4* selected / clock stopped

0b101 = *CMU_CLK5* selected / *TRIG[x-1]* selected

0b110 = *CMU_CLK6* selected / *TIM_EXT_CAPTURE[x]* selected

0b111 = *CMU_CLK7* selected / *CMU_CLK7* selected

Note: This register is a shadow register for the register CLK_SRC. Thus, if the CMU_CLK source for PWM generation should be changed during operation, the old CMU_CLK has to operate until the update of the ATOM channels internal CLK_SRC register by the CLK_SRC_SR content is done either by an end of a period or a forced update.

Note: After (channel) reset the selected CLK_SRC value is the SYS_CLK (input of Global Clock Divider). To use in SOMP mode one of the

CMU_CLKx, it is recommended to perform a forced update of CLK_SRC with the value of CLK_SRC_SR value before/with enabling the channel.

Note: In case of ECLK_SRC=1 and CLK_SRC_SR = 0b011/0b100/0b101/0b110 a force update leads to an immediate update of CM0, CM1 and CLK_SRC.

Bit 15 **ECLK_SRC:** Extend CLK_SRC

0 = CLK_SRC_SR set 1 selected

1 = CLK_SRC_SR set 2 selected

See bit CLK_SRC_SR description for details.

Note: This bit is only applicable in SOMP and SOMS mode.

Bit 16 **WR_REQ**: CPU Write request bit for late compare register update.

0 = No late update requested by CPU

1 = Late update requested by CPU

Note: The CPU can disable subsequent ARU read requests by the channel and can update the shadow registers with new compare values, while the compare units operate on old compare values received by former ARU accesses, if occurred.

Note: On a compare match event, the WR_REQ bit will be reset by hardware.

Note: At the point of the force update only the shadow registers SR0 and SR1 are transferred into the CM0, CM1 registers. The output action is still defined by the ACBI bit field described by the ARU together with the old compare values for CM0/CM1.

Note: This bit is only applicable in SOMC and SOMB mode.

Bit 17 **TRIG_PULSE**: Trigger output pulse length of one SYS_CLK period

0 = output on TOM[i]_OUT[x]_T is '1' as long as CN0=SR0 (if SR=_TRIG=1)

1 = output on TOM[i]_OUT[x]_T is '1' for only one SYS_CLK period if CN0=SR0 (if SR=_TRIG=1)

Bit 19:18 **UDMODE**: up/down counter mode

0b00 = up/down counter mode disabled: CN0 counts always up

0b01 = up/down counter mode enabled: CN0 counts up and down, CM0,CM1 are updated if CN0 reaches 0 (i.e. changes from down to up)

0b10 = up/down counter mode enabled: CN0 counts up and down, CM0,CM1 are updated if CN0 reaches CM0 (i.e. changes from up to down)

0b11 = up/down counter mode enabled: CN0 counts up and down, CM0,CM1 are updated if CN0 reaches 0 or CM0 (i.e. changes direction)

Note: This mode is only applicable in SOMP mode.

Bit 20 **RST_CCU0**: Reset source of CCU0

0 = Reset counter register CN0 to 0 on matching comparison with CM0

1 = Reset counter register CN0 to 0 on trigger *TRIG_[x-1]* or TIM_*EXT_CAPTURE(x)*.

Note: If RST_CCU0=1 and UPEN_CTRLx=1 are set, *TRIG_[x-1]* or TIM_*EXT_CAPTURE(x)* triggers also the update of work register (CM0, CM1 and CLK_SRC).

Note: this bit is only applicable in SOMP mode.

Note: This bit should only be set if bit OSM=0 (i.e. in continuous mode)

Bit 21 **OSM_TRIG**: enable trigger of one-shot pulse by trigger signal *OSM_TRIG*

0 = signal OSM_TRIG cannot trigger start of single pulse generation

1 = signal OSM_TRIG can trigger start of single pulse generation (only if bit OSM = 1)

Note: This bit should only be set if bit OSM=1 and bit RST_CCU0=0.

Bit 22 **EXT_TRIG**: select *TIM_EXT_CAPTURE(x)* as trigger signal

0 = signal *TIM_[x-1]* is selected as trigger to reset CN0 or to start single pulse generation.

1 = signal *TIM_EXT_CAPTURE(x)* is selected

Bit 23 **EXTTRIGOUT**: select *TIM_EXT_CAPTURE(x)* as potential output signal *TRIG_[x]*

0 = signal *TRIG_[x-1]* is selected as output on *TRIG_[x]* (if TRIGOUT=1)

1 = signal *TIM_EXT_CAPTURE(x)* is selected as output on *TRIG_[x]* (if TRIGOUT=1)

Bit 24 **TRIGOUT**: Trigger output selection (output signal *TRIG_CHx*) of module ATOM_CHx.

0 = *TRIG_[x[* is *TRIG_[x-1]* or TIM_*EXT_CAPTURE(x)*.

1 = *TRIG_[x]* is *TRIG_CCU0*

Bit 25 **SLA**: 'serve last' ARU communication strategy

0 = Capture SRx time stamps after CCU0 match event not provided to ARU

1 = Capture SRx time stamps after CCU0 match event provided to ARU

Note: This bit is only applicable in SOMC mode.

Note: Please note, that setting of this bit has only effect, when ACBI(4:2) is configured for 'serve last' compare strategy (0b100, 0b101, or 0b110).

Note: When this bit is not set, the captured time stamps in the shadow registers SRx are only provided after the CCU1 match occurred. The ACBO(4:3) bits always return 0b10 in that case.

Note: By setting this bit, the ATOM channel also provides the captured time stamps after the CCU0 match event to the ARU. The

ACBO(4:3) bits are set to 0b01 in that case. After the CCU1 match event, the time stamps are captured again in the SRx registers and provided to the ARU. The ACBO(4:3) bits are set to 0b10. When the data in the shadow registers after the CCU0 match was not consumed by an ARU destination and the CCU1 match occurs, the data in the shadow registers is overwritten by the new captured time stamps. The ATOM channel does not request new data from the ARU when the CCU0 match values are read from an ARU destination.

Bit 26 **OSM:** One-shot mode
0 = Continuous PWM generation after channel enable
1 = A single pulse is generated
Note: this bit is only applicable in SOMP and SOMS modes.

Bit 27 **ABM:** ARU blocking mode
0 = ARU blocking mode disabled: ATOM reads continuously from ARU and updates CM0,CM1 and ACB bits in case of SOMC mode or SR0,SR1 and ACB bits in case of SOMB mode independent of pending compare match event.
1 = ARU blocking mode enabled: after update of CM0, CM1 and ACB bit in case of SOMC mode or SR0, SR1 and ACB bits in case of SOMB mode via ARU, no new data is read via ARU until compare match event occurred and in case of SOMC mode SR0 and/or SR1 are read.
Note: This bit is only applicable in SOMC and SOMB mode.

Bit 28 **Reserved**
Note: Read as zero, should be written as zero.

Bit 29 **EXT_FUPD:** external forced update
0 = use FUPD(x) signal from AGC to force update
1 = use TIM_EXT_CAPTURE signal to force update

Note: This bit is only applicable in SOMP and SOMS mode.

Bit 30 **SOMB**: SOMB mode
0 = ATOM channel mode defined by bit field MODE
1 = ATOM SOMB mode enabled

Bit 31 **FREEZE**
0 = a channel disable/enable may change internal register and output register
1 = a channel enable/disable does not change an internal or output register but stops counter CN0 (in SOMP mode), comparison (in SOMC/SOMB mode) and shifting (in SOMS mode)

Note: if channel is disabled and ouptut is enabled, in SOMP mode with UDMODE!=0b00 the output is dependng directly on SL bit, independent on FREEZE mode.

13.6.10 Register ATOM[i]_CH[x]_STAT

Address Offset:	see Appendix B																Initial Value: 0x0000_000X															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved			ACBO					DR	WRF	DV	ACBI					Reserved															OL
Mode	R			R					R	RCw	R	R					R															R
Initial Value	0b000			0b00000					0	0	0	0b00000					0x0000															X

Bit 0 **OL**: Actual output signal level of *ATOM_CHx_OUT*.
0 = Actual output signal level is low
1 = Actual output signal level is high
Note: Reset value is the inverted value of bit SL which depends on the hardware configuration chosen by silicon vendor.

Bit 15:1 **Reserved**
Note: Read as zero, should be written as zero.

Bit 20:16 **ACBI**: ATOM Mode control bits.
Note: For ATOM SOMI, SOMC, SOMP and SOMS mode this register serves as a mirror for the five ARU control bits received through the ARU interface. The bits are valid, when the DV bit is set.

Note: For SOMB mode this bit field serves as the work register of the compare strategy. It can be updated with the value of bit field ACB of register ATOM[i]_CH[x]_CTRL or the value of internal shadow register ACB_SR.

Bit 21 **DV**: Valid ARU Data stored in compare registers.
0 = No valid data stored in register CM0 and/or CM1, no comparison is activated.
1 = Valid data stored in CM0 and/or CM1, comparison activated.
Note: This bit is only applicable in SOMC and SOMB mode. The CPU can determine the status of the ARU transfers with this bit. After the compare event occurred, the bit is reset by hardware.

Bit 22 **WRF**: Write request of CPU failed for late update.
0 = Late update was successful, CCUx units wait for comparison.
1 = Late update failed.

The bit WRF can be reset by writing a 1 to it.
Note: This bit is only applicable in SOMC and SOMB mode.

Bit 23 **DR:** ARU data rejected flag
0 = received ARU data stored
1 = received ARU data rejected
Note: The flag is cleared if valid data is received and stored via ARU.

Bit 28:24 **ACBO**: ATOM Internal status bits.
ACBO[3] = 1: CCU0 Compare match occurred
ACBO[4] = 1: CCU1 Compare match occurred
Note: These bits are only set in SOMC mode.
Note: ACBO is reset to 0b00000 on an update of register CM0 or CM1 (via ARU or CPU)

Note: In SOMC mode these bits are sent as ARU control bits 52 .. 48.

Bit 31:29 **Reserved**
Note: Read as zero, should be written as zero.

13.6.11 Register ATOM[i]_CH[x]_RDADDR

Address Offset:	see Appendix B																Initial Value: 0x01FE_01FE															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved							RDADDR1									Reserved							RDADDR0								
Mode	R							RPw									R							RPw								
Initial Value	0x00							0x1FE									0x00							0x1FE								

Bit 8:0 **RDADDR0**: ARU Read address 0.
Note: This read address is used by the ATOM channel to receive data from ARU immediately after the channel and ARU access is enabled (see ATOM[i]_CH[x]_CTRL register for details).

Note: this bit field is only writable if channel is disabled.

Bit 15:9 **Reserved**
Note: Read as zero, should be written as zero.

Bit 24:16 **RDADDR1**: ARU Read address 1.
Note: The ATOM channel switches to this read address, when requested in the ARU control bits 52 to 48 with the pattern "111--". The channel

switches back to the RDADDR0 after one ARU data package was received on RDADDR1 and the compare match event is occurred.

Note: This read address is only applicable in SOMC mode.

Note: this bit field is only writable if channel is disabled.

Bit 31:25 **Reserved**

Note: Read as zero, should be written as zero.

13.6.12 Register ATOM[i]_CH[x]_CN0

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_0000**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								CN0																							
Mode	R								RW																							
Initial Value	0x00								0x000000																							

Bit 23:0 **CN0**: ATOM CCU0 counter register.

Bit 31:24 **Reserved**

Note: Read as zero, should be written as zero.

13.6.13 Register ATOM[i]_CH[x]_CM0

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_0000**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								CM0																							
Mode	R								RW																							
Initial Value	0x00								0x000000																							

Bit 23:0 **CM0**: ATOM CCU0 compare register.

Bit 31:24 **Reserved**

Note: Read as zero, should be written as zero.

Note: This register is write protected in SOMC mode and returns AEI_STATUS=0b10 on write access, when in 'serve last' compare strategy the first match of CCU0 occurred.

13.6.14 Register ATOM[i]_CH[x]_SR0

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								SR0																							
Mode	R								RW																							
Initial Value	0x00								0x000000																							

Bit 23:0 **SR0**: ATOM channel x shadow register SR0.

Note: The SR0 register is used as shadow register for CM0 in SOMP and SOMS modes and is used as capture register for time base TBU_TS0 in SOMC mode.

Bit 31:24 **Reserved**

Note: Read as zero, should be written as zero.

13.6.15 Register ATOM[i]_CH[x]_CM1

Address Offset:	**see Appendix B**																**Initial Value: 0x0000_0000**															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								CM1																							
Mode	R								RPw																							
Initial Value	0x00								0x000000																							

Bit 23:0 **CM1**: ATOM CCU1 compare register.

Bit 31:24 **Reserved**

Note: Read as zero, should be written as zero.

Note: This register is write protected in SOMC mode and returns AEI_STATUS=0b10 on write access, when in 'serve last' compare strategy the first match of CCU0 occurred.

13.6.16 Register ATOM[i]_CH[x]_SR1

Address Offset:	**see Appendix B**																**Initial Value: 0x0000_0000**															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								SR1																							
Mode	R								RW																							
Initial Value	0x00								0x000000																							

Bit 23:0 **SR1**: ATOM channel x shadow register SR0.

Note: The SR1 register is used as shadow register for CM1 in SOMP and SOMS modes and is used as capture register for time base TBU_TS1 or TBU_TS2 (when selected in ATOM[i]_CH[x]_CTRL register) in SOMC mode.

Bit 31:24 **Reserved**

Note: Read as zero, should be written as zero.

13.6.17 Register ATOM[i]_CH[x]_IRQ_NOTIFY

<table>
<tr><td><b>Address Offset:</b></td><td colspan="16"><b>see Appendix B</b></td><td colspan="16"><b>Initial Value: 0x0000_0000</b></td></tr>
<tr><td></td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>Bit</td><td colspan="30">Reserved</td><td>CCU1TC</td><td>CCU0TC</td></tr>
<tr><td>Mode</td><td colspan="30">R</td><td>RCw</td><td>RCw</td></tr>
<tr><td>Initial Value</td><td colspan="30">0x0000000</td><td>0</td><td>0</td></tr>
</table>

Bit 0 **CCU0TC**: CCU0 Trigger condition interrupt for channel x.
0 = No interrupt occurred.
1 = CCU0 Trigger condition interrupt was raised by ATOM channel x.
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 1 **CCU1TC**: CCU1 Trigger condition interrupt for channel x.
See bit 0.
Note: If bit SR0_TRIG is set to 1 (only valid in SOMP mode), this interrupt notify flag is set in case of **SR0** is equal to **CN0** and not set in case of **CM1** >=/<= **CN0**.

Bit 31:2 **Reserved**
Note: Read as zero, should be written as zero

13.6.18 Register ATOM[i]_CH[x]_IRQ_EN

<table>
<tr><td><b>Address Offset:</b></td><td colspan="16"><b>see Appendix B</b></td><td colspan="16"><b>Initial Value: 0x0000_0000</b></td></tr>
<tr><td></td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>Bit</td><td colspan="30">Reserved</td><td>CCU1TC_IRQ_EN</td><td>CCU0TC_IRQ_EN</td></tr>
<tr><td>Mode</td><td colspan="30">R</td><td>RW</td><td>RW</td></tr>
<tr><td>Initial Value</td><td colspan="30">0x0000000</td><td>0</td><td>0</td></tr>
</table>

Bit 0 **CCU0TC_IRQ_EN**: *ATOM_CCU0TC_IRQ* interrupt enable.

0 = Disable interrupt, interrupt is not visible outside GTM-IP.
1 = Enable interrupt, interrupt is visible outside GTM-IP.

Bit 1 **CCU1TC_IRQ_EN**: *ATOM_CCU1TC_IRQ* interrupt enable.
See bit 0.

Bit 31:2 **Reserved**
Note: Read as zero, should be written as zero

13.6.19 Register ATOM[i]_CH[x]_IRQ_FORCINT

Address Offset:	**see Appendix B**																**Initial Value: 0x0000_0000**															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																														TRG_CCU1TC	TRG_CCU0TC
Mode	R																														RAw	RAw
Initial Value	0x0000 000																														0	0

Bit 0 **TRG_CCU0TC**: Trigger *ATOM_CCU0TC_IRQ* interrupt by software.
0 = No interrupt triggering.
1 = Assert *CCU0TC_IRQ* interrupt for one clock cycle.
Note: This bit is cleared automatically after write.
Note: This bit is write protected by bit RF_PROT of register GTM_CTRL

Bit 1 **TRG_CCU1TC**: Trigger *ATOM_CCU1TC_IRQ* interrupt by software
0 = No interrupt triggering.
1 = Assert *CCU1TC_IRQ* interrupt for one clock cycle.
Note: This bit is cleared automatically after write.
Note: This bit is write protected by bit RF_PROT of register GTM_CTRL

Bit 31:2 **Reserved**
Note: Read as zero, should be written as zero

13.6.20 Register ATOM[i]_CH[x]_IRQ_MODE

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_000X**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																														IRQ_MODE	
Mode	R																														RW	
Initial Value	0x0000 0000																														0bXX	

Bit 1:0 **IRQ_MODE**: IRQ mode selection
0b00 = Level mode
0b01 = Pulse mode
0b10 = Pulse-Notify mode
0b11 = Single-Pulse mode
Note: The interrupt modes are described in section 2.5.

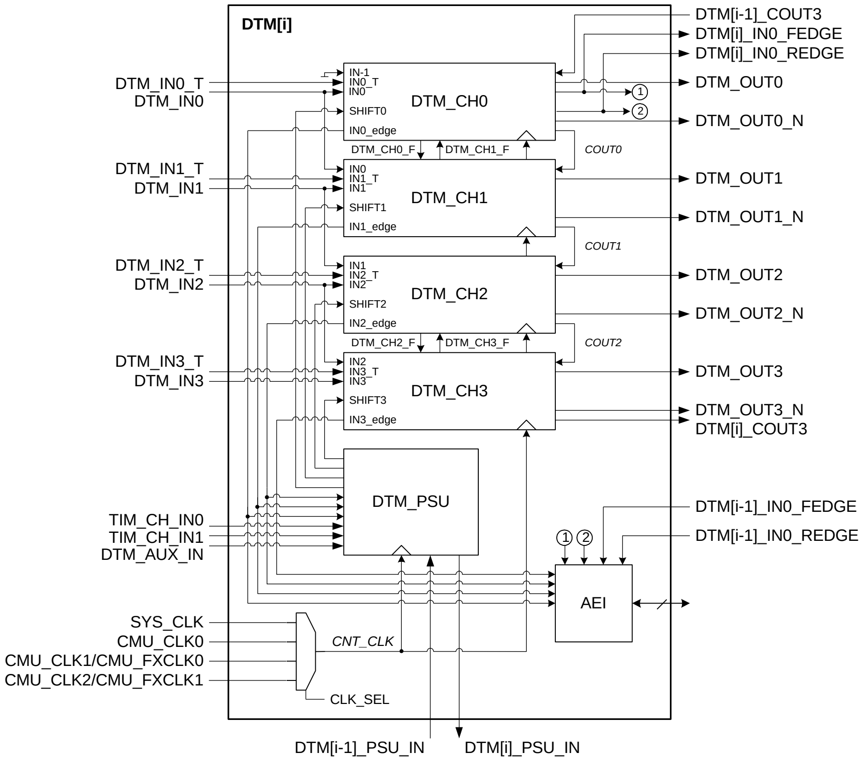
Bit 31:2 **Reserved**
Note: Read as zero, should be written as zero

14 Dead Time Module (DTM)

14.1 Overview

The following figure gives an overview of the structure of the Dead Time Module (DTM).

14.1.1 DTM overview



The main function of the DTM is to derive for each input *DTM_IN0* to *DTM_IN3* the individual inverse signal (*DTM[i]_OUT[x]_N*) and to apply an edge specific delay between the edge of the original signal and the edge of the derived inverted signal (i.e., the dead time). This function is mainly used for controlling of half bridges.

A second function provided by DTM is to set the outputs of one channel to the value of the preceding channel if requested by a trigger on input *TIM_CH_IN0*, *TIM_CH_IN1* or *DTM_AUX_IN*. This feature allows a phase shift on one PWM signal to the phase of the preceding PWM signal up to the next edge on this channel.

The third function provided by DTM is to (N)AND/(N)OR/X(N)OR combine the input *DTM_IN[x]* signal of one DTM channel with the signal on input *TIM_CH_IN0, TIM_CH_IN1* or *DTM_AUX_IN* (selected inside DTM_PSU and assigned to one of the

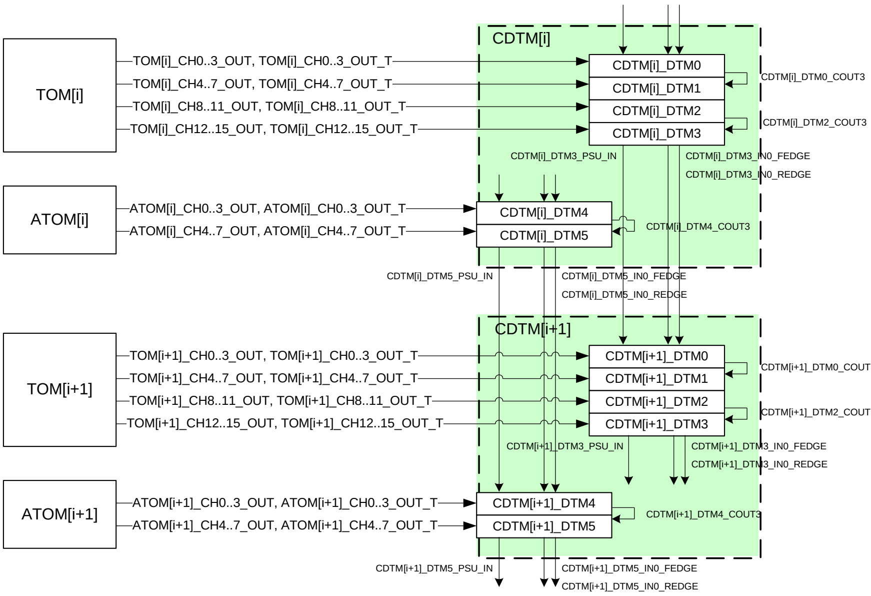
signals *SHIFT[x]*) or with the combinational output (signal *COUT[x]*) of preceding channel.
As a result *COUT2* may be the combined signal of *DTM_IN0* and *TIM_CH_IN0, TIM_CH_IN1* or *DTM_AUX_IN* and the signal *DTM_IN1*. For *COUT3* this chain can be combined again with signal *DTM_IN3*.

The outputs of each channel may be swapped individually to provide the function of combining signals on each output of a channel.

In general, the DTM instances are placed behind the TOM and the ATOM instances, i.e., the outputs *TOM_OUT[x] and TOM_OUT[x]_T* or *ATOM_OUT[x] and ATOM_OUT[x]_T* are each routed to the DTM instance inputs *DTM_IN[y]* and *DTM_IN[y]_T*.
Four DTM instances behind a TOM instance i and two DTM instances behind an ATOM instance i are grouped together in a Cluster DTM hierarchy called CDTM[i].
The connections between DTM and the modules TOM and ATOM are depicted in 14.1.2.

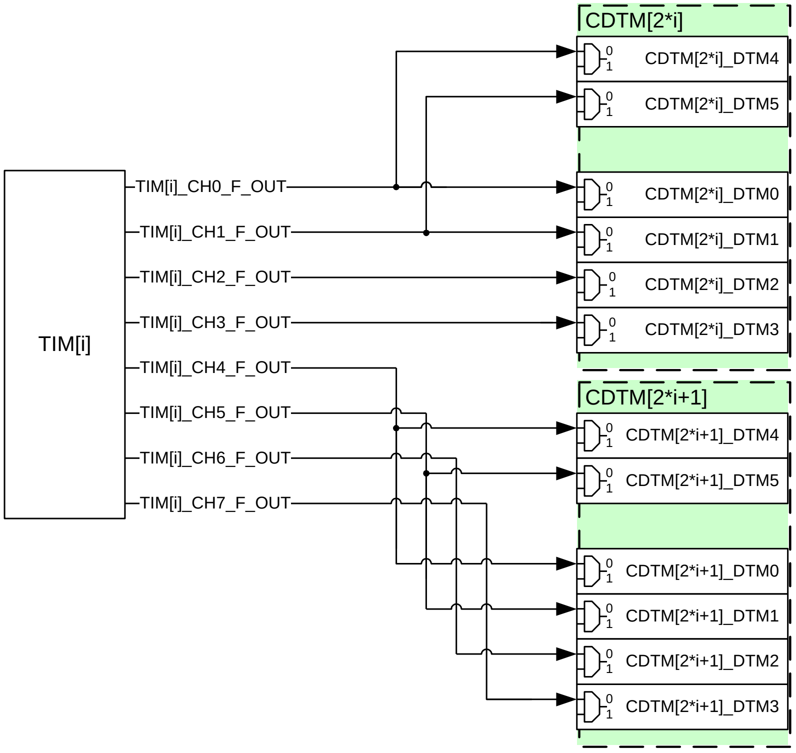
Note, depending on device configuration, not every DTM instance is available. E.g. a device may only have one DTM connected to the first four channels of ATOM. In this case, the other four channels (4 to 7) are connected directly to GTM outputs.
For detailed information, which DTM instance is available, refer to corresponding device specific Appendix B of this specification [1].

14.1.2 Connections of TOM and ATOM to DTM inputs DTM_IN[y]/DTM_IN[y]_T

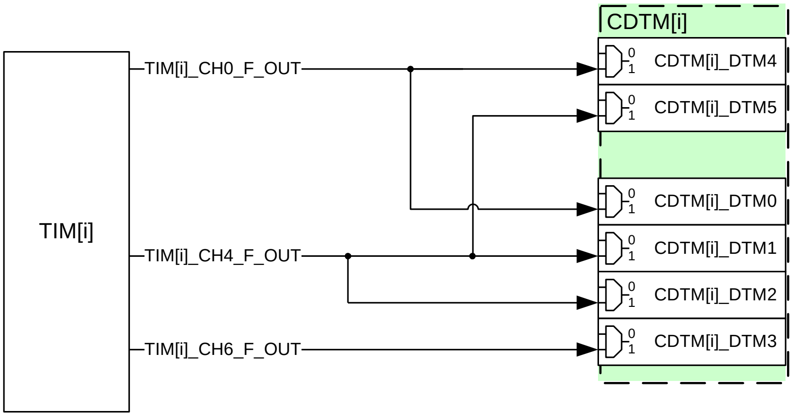


Additionally, the DTM instances have inputs *TIM_CH_IN0/TIM_CH_IN1* which are driven by TIM output signals *TIM[i]_CH[x]_F_OUT*.
There are two configurations of TIM to DTM connections possible depending on the DTM channel specific configuration bit TIM_SEL.
In case of TIM_SEL=0 the connected TIM input may not be of the same cluster as the DTM.
In case of TIM_SEL=1 the TIM input is of the same cluster as the DTM.

14.1.3 Connections of TIM to DTM inputs TIM_CH_IN0/TIM_CH_IN1 for TIM_SEL=0



14.1.4 Connections of TIM to DTM inputs TIM_CH_IN0/TIM_CH_IN1 for TIM_SEL=1

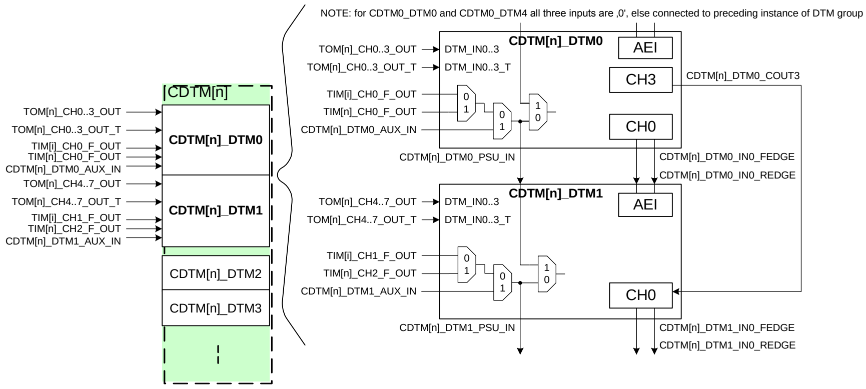


There are also connections between DTM instances of same CDTM instance.
For each pair of DTM instance 2i and 2i+1 the combinatorial output *COUT(3)* of DTM[2i] channel 3 is connected to DTM[2i+1] channel 0 *COUT(0-1)*.
With this a combinatorial chain over two neighbored DTM instances can be configured.
If one of the two neighbored DTM instances is not available (i.e. it is an empty instance) the inputs used for connections between two neighbored DTM instances are left open.

Note: For channel x=0 of DTM instance 2i input signals *COUT[x-1]* is unused and **I1SEL_[x]** is defined as 0.
An additional link between DTM instances behind an ATOM is a forwarding of *DTM[i]_PSU_IN* signal to next available instance of DTM behind an ATOM (e.g. *DTM[i+1]_PSU_IN*).
The same link is available between all available DTM behind a TOM.
Note, for unavailable DTM[i] instances (i.e. the instance DTM[i] is called empty) the signal *DTM[i-1]_PSU_IN* is passed through empty instance DTM[i] to *DTM[i+1]_PSU_IN*, *DTM[i-1]_IN0_FEDGE* and *DTM[i-1]_IN0_REDGE* are passed through DTM[i] to *DTM[i+1]_IN0_FEDGE* and *DTM[i+1]_IN0_REDGE*.

Further connections between neighbored DTM instances are depicted in the following figure:

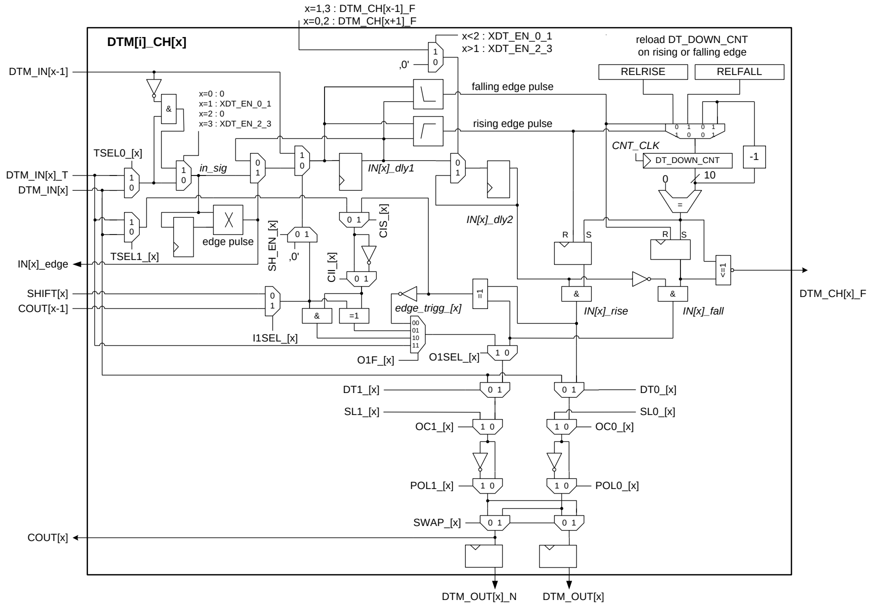
14.1.5 Connections between DTM instances



14.2 DTM Channel

The following figure depicts the functions of a DTM channel.

14.2.1 DTM channel overview



The main feature of each channel is to derive the inverse signal out of the input signal *DTM_IN[x]*, apply an edge dependent delay on the two resulting signal paths and provide these signals at the outputs *DTM[i]_OUT[x]* and *DTM[i]_OUT[x]_N*.

There are two possibilities to apply dead time on GTM output signals.
One is to use one DTM channel per TOM/ATOM channel and generate inside the DTM the second inverse signal. This is called the standard dead time generation.
The second way is to generate two signals out of two TOM/ATOM channel and to apply inside the DTM only the dead time by using two cross linked DTM channel. This is called the cross dead time generation.

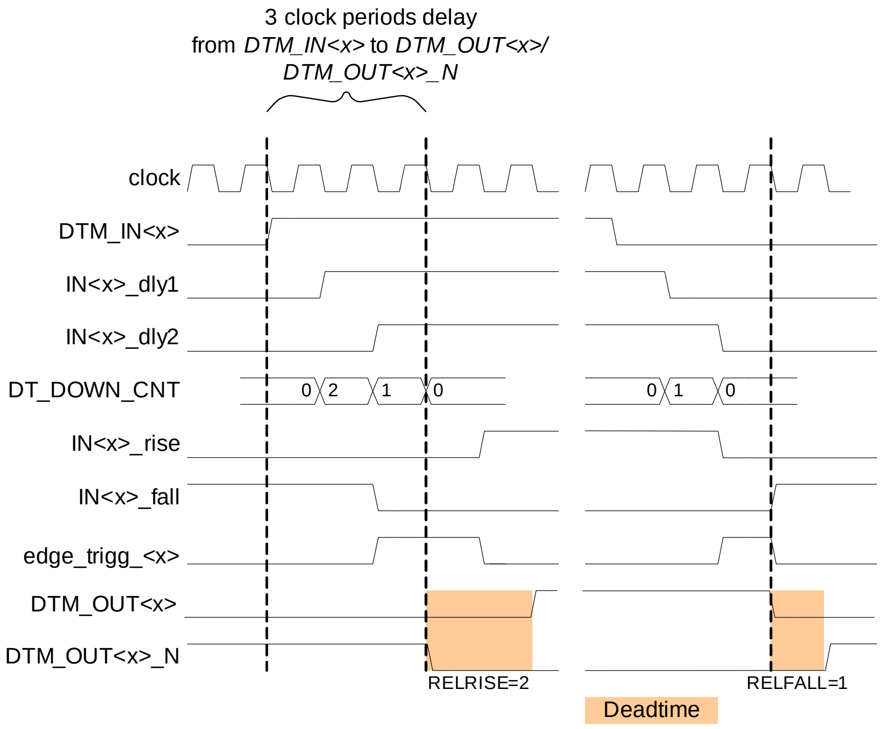
14.2.2 Standard dead time generation

The dead time can be configured for each edge individually. The bit field **RELRISE** in register **DTM[i]_CH[x]_DTV** contains the reload value for the counter and defines the delay for rising edges in multiples of selected clock ticks.
The bit field **RELFALL** in register **DTM[i]_CH[x]_DTV** contains the reload value for the counter and defines the delay for falling edges in multiples of selected clock ticks.

The counter is reloaded with the value of **RELRISE** on a rising edge and reloaded with the value of **RELFALL** on a falling edge on input *DTM_IN[x]* (or *DTM_IN[x-1]* in case of shift enable **SH_EN_[x]**).
On a reload of the counter the flip-flop following the counter output comparator is reset and stays reset until the counter has reached 0.
After reload, the counter **DT_DOWN_CNT** counts down until it reaches 0 and stops at 0.

The signal flow for function of standard dead time signal generation is depicted in following figure

14.2.2.1 Wave signals for function of dead time generation



Note: The delay from the input signal *DTM_IN[x]* to the output signals *DTM[i]_OUT[x]* and *DTM[i]_OUT[x]_N* is three system clock periods by disabled feed through (see **DT0/1_[x]** in **DTM[i]_CH_CTRL2**).

Note: The delay from the input signal *DTM_IN[x]* to the output signals *DTM[i]_OUT[x]* and *DTM[i]_OUT[x]_N* is one system clock periods by enabled feed through (see **DT0/1_[x]** in **DTM[i]_CH_CTRL2**).

Note: The delay from the input signal *DTM_IN[x]_T* to the output signals *DTM[i]_OUT[x]* and *DTM[i]_OUT[x]_N* is three system clock periods in case of disabled feed through (see **O1F_[x]** and **O1SEL_[x]** in **DTM[i]_CH_CTRL2**).

Note: The delay from the input signal *DTM_IN[x]_T* to the output signals *DTM[i]_OUT[x]* and *DTM[i]_OUT[x]_N* is one system clock periods in case of enabled feed through (see **O1F_[x]** and **O1SEL_[x]** in **DTM[i]_CH_CTRL2**).

Note: The reset level of the output signals *DTM[i]_OUT[x]* connected from ATOM module depends on the hardware configuration value atom_out_reset_level_c chosen by silicon vendor.

Note: The reset level of the output signals *DTM[i]_OUT[x]_N* connected from ATOM module is defined by the inverse hardware configuration value atom_out_reset_level_c chosen by silicon vendor.

Note: The reset level of the output signals *DTM[i]_OUT[x]* connected from TOM module is defined by the hardware configuration value tom_out_reset_level_c chosen by silicon vendor.

Note: The reset level of the output signals *DTM[i]_OUT[x]_N* connected from TOM module is defined by the inverse hardware configuration value tom_out_reset_level_c chosen by silicon vendor.

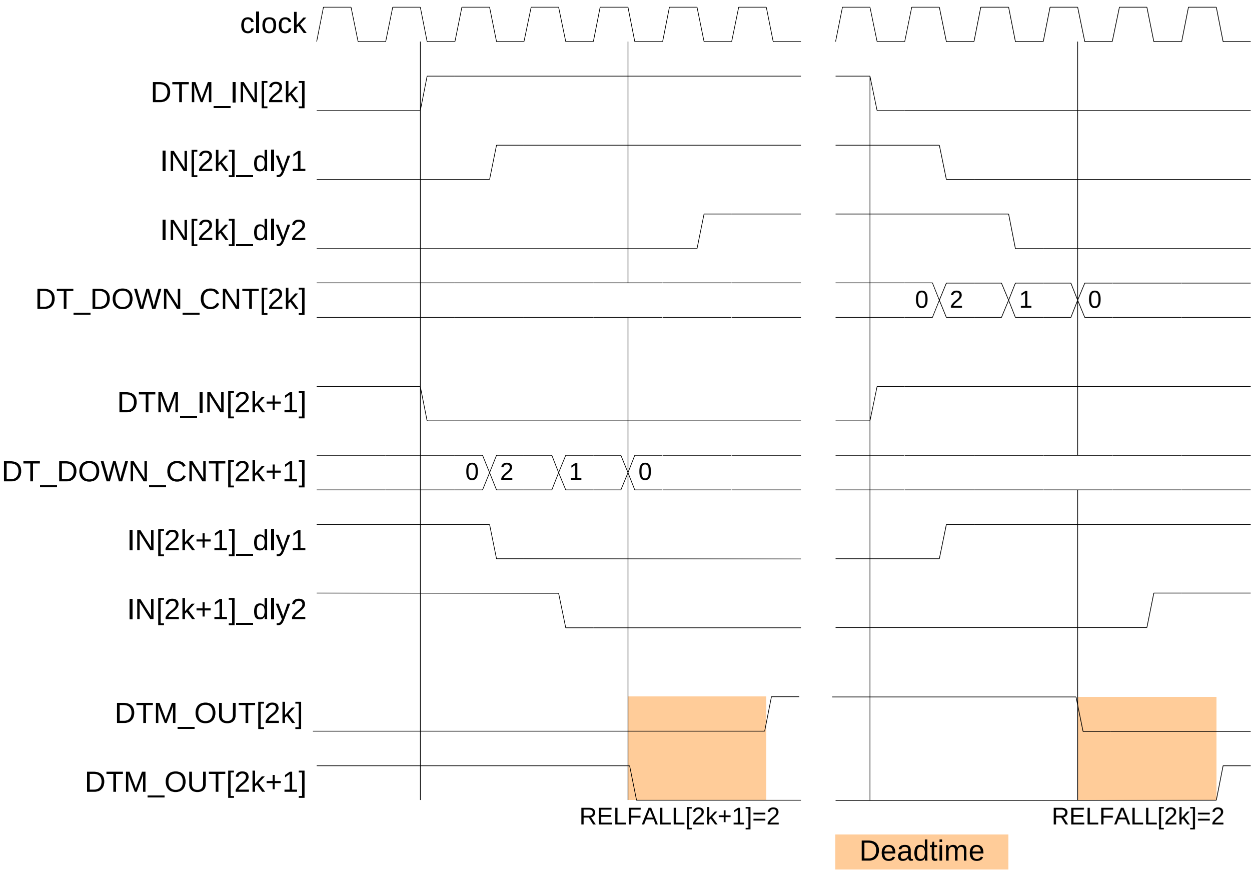
14.2.3 Cross channel dead time

A second way to apply a dead time value on two output signals is the cross channel dead time.
In opposite to the dead time described in chapter 14.2.2 the cross channel dead time mode does not generate out of one signal the corresponding inverse signal but tries to apply the dead time on the input signals of two neighbored DTM channel.
To do this, two neighbored DTM input signals (on DTM channel (2k) and (2k+1) for k=0,1) are cross linked together in the way that a falling edge on one channel leads to a hold phase of current signal value on the cross linked channel.

This behavior is reached by the following:
A falling edge on e.g. channel (2k) reloads the **DT_DOWN_CNT** with the value of **RELFALL**. While this counter is counting down, the output signal of the cross linked channel (2k+1) keeps its value. If the counter **DT_DOWN_CNT** has reached 0 again, the channel (2k+1) output is released and can follow the value on its input.
The timing of the cross channel dead time is depicted in figure 14.2.3.1.

The following figure 14.2.3.1 shows the behavior in case of input edges at *DTM_IN[2k]* and *DTM_IN[2k+1]* occur at the same point in time.
Then the falling edge is forwarded immediately (with only two clock cycles delay) and the rising edge is delayed additionally by the number of clock ticks specified by the **RELFALL** parameter of the cross linked channel.

14.2.3.1 Cross channel dead time timing diagram

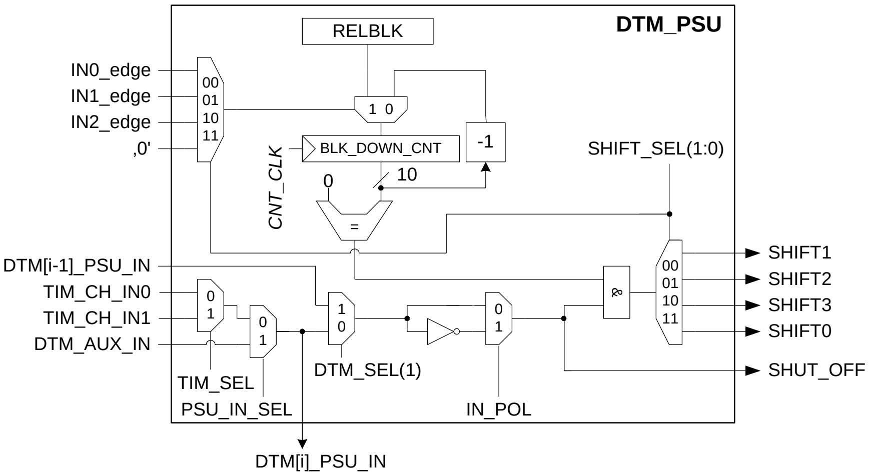


In case of high level (i.e. 1) at the DTM inputs *DTM_IN[2k]* and *DTM_IN[2k+1]* at the same point in time, the channel of (2k) has higher priority than the corresponding channel (2k+1). This means that in this case the input *DTM_IN[2k+1]* is forced immediately at channel input to low level (i.e. 0).
As a result the DTM output of channel *DTM_OUT[2k+1]* can never be high if the cross linked channel *DTM_OUT[2k]* is high.

14.3 Phase Shift Control Unit

The phase shift unit (DTM_PSU) is depicted in the following figure. It supports the second major function of the DTM module to allow phase shifting of PWM signal on one of the channels.

14.3.1 Phase Shift Unit overview

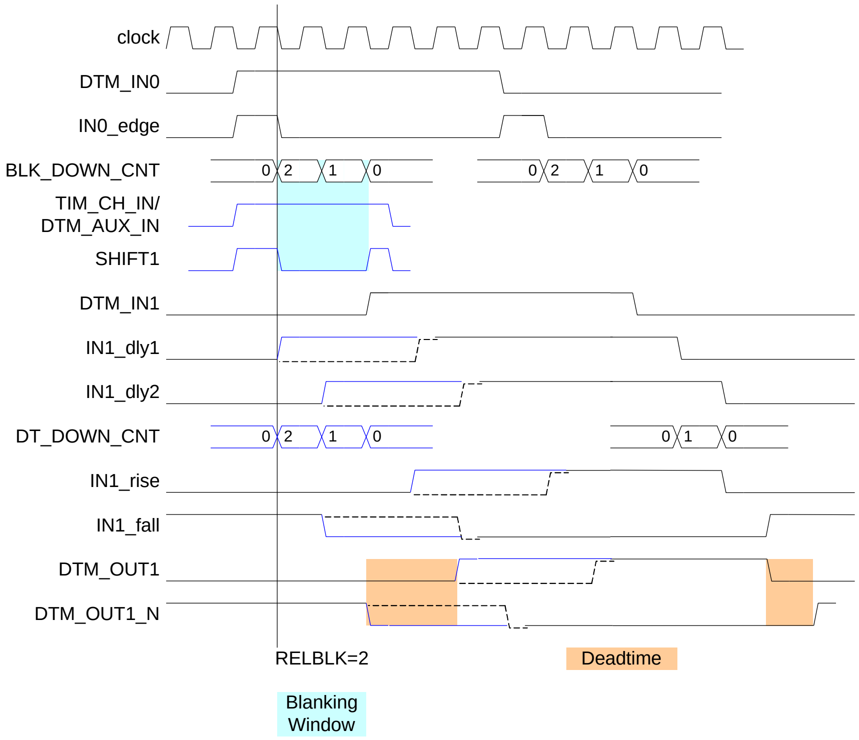


This sub-module provides an additional counter **BLK_DOWN_CNT** and reload register **RELBLK** (bit field of register **DTM[i]_PS_CTRL**). The counter is reloaded on an edge detected on one of the selected signals *IN0_edge* to *IN2_edge* (selected by bit field **SHIFT_SEL** in register **DTM[i]_PS_CTRL**). Then, the counter counts down until it reaches 0. While the counter is counting down, it blocks the trigger (i.e. the selected one of the signals *SHIFT[x]*) of one of the channels by one of the input signals *TIM_CH_IN0, TIM_CH_IN1* or *DTM_AUX_IN*.

If the counter **BLK_DOWN_CNT** is not counting, a pulse on the input *TIM_CH_IN0, TIM_CH_IN1* or *DTM_AUX_IN* is forwarded to one of the selected DTM_PSU outputs *SHIFT[x]*. This signal triggers in the selected channel (if **SH_EN_x**=1) the update of the first flip-flop on channel x (i.e. representing *IN[x]_DLY*) to the input value *DTM_IN[x-1]* of the preceding channel. If this update leads to an edge, the succeeding part of DTM channel derives the inverse signal and applies the corresponding dead time (i.e. the edge delay) to the output signals of the channel.

Note: For channel x=0 input signals *DTM_IN[x-1]* is unused and **SH_EN_[x]** is defined as 0.
The following figure shows an example of phase shifting on channel 1.

14.3.2 Example wave of phase shift on channel 1



14.4 Multiple output signal combination

Each channel provides additionally the possibility to combine the channel inputs *DTM_IN[x]* and *SHIFT[x]* or *COUT[x-1]* (selected by **I1SEL_[x]**) by an AND or an XOR gate (selected by **O1F_[x]**).

It is recommended to use the combination of signals only if bit field **RELBLK** of register **DTM[i]_PS_CTRL** is 0. Otherwise, the signal *TIM_CH_IN0, TIM_CH_IN1 or DTM_AUX_IN* may be disturbed by the blanking window counter.

Together with the inverter inside sub-module DTM_PSU (selected by **IN_POL**), the inverter on each output of a channel (selected by **POL0_[x]/POL1_[x]**) and the possibility to change polarity of *DTM_IN[x]* inside connected TOM/ATOM channel, a (N)AND, (N)OR or X(N)OR combination of the signals is possible.

14.4.1 Combination of input signal TIM_CH_IN/AUX_IN with TOM/ATOM signal

If the input selection **I1SEL_[x]** of a channel x is set to 0, the output selection **O1SEL_[x]** is set to 1 and **SWAP_[x]** is set to 0, depending on **PSU_IN_SEL** either *TIM_CH_IN0, TIM_CH_IN1* or *DTM_AUX_IN* can be combined with signal *DTM_IN[x]*.

The function of combination on DTM output *DTM[i]_OUT[x]_N* (and also *COUT[x]*) is defined by **O1F_[x]** in the following way:

14.4.1.1 Function of combination on DTM channel x=0 output DTM[i]_OUT[x]_N (and also COUT[x])

	O1F_x	**POL1_x**	**IN_POL**	**(A)TOM output inverted**
XOR	01	0	0	no
AND	10	0	0	no
XNOR	01	1	0	no
NAND	10	1	0	no
XNOR	01	1	1	yes
OR	10	1	1	yes
XOR	01	0	1	yes
NOR	10	0	1	yes

Note: The inversion of the (A)TOM output can be reached by switching the **SL** bit (for TOM and ATOM SOMP/SOMC mode).

14.4.2 Combination of multiple TOM/ATOM output signals

If the input selection **I1SEL_[x]** of a channel x (with x=1..3) is set to 1, the output selection **O1SEL_[x]** is set to 1 and **SWAP_[x]** is set to 0, the output of the preceding DTM channel *COUT[x-1]* can be combined with signal *DTM_IN[x]*.
The function of combination on DTM output *DTM[i]_OUT[x]_N* (and also *COUT[x]*) is defined by **O1F_[x]** in the following way:

14.4.2.1 Function of combination on DTM on channel x=1..3 output DTM[i]_OUT[x]_N (and also COUT[x])

	O1F_x	**POL1_x**	**POL1_x-1**	**(A)TOM output inverted**
XOR	01	0	0	no
AND	10	0	0	no
XNOR	01	1	0	no
NAND	10	1	0	no
XNOR	01	1	1	yes

OR	10	1	1	yes
XOR	01	0	1	yes
NOR	10	0	1	yes

By setting **I1SEL_[x]** to 1 on all four channel, a combination of all four signals *DTM_IN0* to *DTM_IN3* can be achieved (combinatorial chain).

To allow also combination of signals generated for output *DTM[i]_OUT[x]*, the outputs 0 and 1 can be swapped by setting bit **SWAP_ [x]** for channel x.

14.4.3 Pulse generation on edge

Another feature of the DTM is to generate on the second output *DTM[i]_OUT[x]_N* a pulse on every edge of corresponding input signal *DTM[i]_IN[x]*.
This can be reached by configuring **O1SEL_[x]** to 1, i.e. selecting signal *edge_trigg_[x]* as the output signal (**O1F_[x]** has to be 0b00). The signal *edge_trigg_[x]* is depicted in figure 14.2.2.1

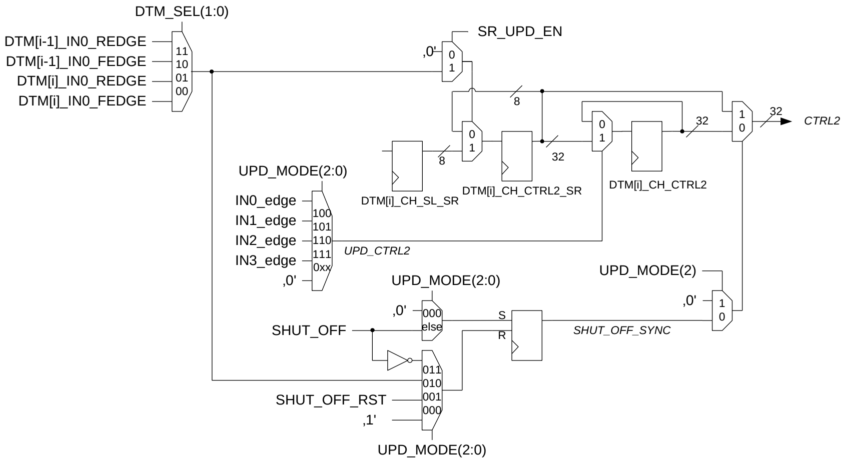
The pulse length can be adjusted individually for each edge type by the configuration value **REL_RISE** and **REL_FALL** of register **DTM[i]_CH[x]_DV**.
The parameter **REL_RISE** defines the pulse length in case of a rising edge on input *DTM[i]_IN[x]*, the parameter **REL_FALL** define the pulse length in case of a falling edge on input *DTM[i]_IN[x]*.

The generated edge signal *edge_trigg_[x]* can be combined with the output signal of the preceding DTM channel x-1 at channel input *COUT[x-1]* (see figure 14.2.1).
With the configuration of **CIS[x]**=1 and **I1SEL_[x]**=1, **CII[x]**=0 and **POL1_[x]**=1, the signal *edge_trigg_[x]* of channel x is ORed with the inverse signal at channel input *COUT[x-1]*. The signal at *COUT[x-1]* can be inverted by changing **POL1_[x-1]** of channel x-1.
As a result of this configuration one can generate at each edge on DTM input *DTM_IN[x]* a pulse signal and OR-combine these generated pulse signals with the generated signal of preceding DTM channel. If the combinatorial chain is configured over all four DTM channel the final signal is available at last DTM output *DTM_OUT3_N*.

14.5 Synchronous update of channel control register 2

It is possible to use the shadow register **DTM[i]_CH_CTRL2_SR** and a selected edge of one of the channel 0 to 3 to update the work register **DTM[i]_CH_CTRL2**.
The update mechanism and its configuration are depicted in the following figure.

14.5.1 Synchronous update mechanism of register DTM[i]_CH_CTRL2



If enabled by the bit field **UPD_MODE** of register **DTM[i]_CTRL** (i.e. **UPD_MODE**=1xx), the register **DTM[i]_CH_CTRL2_SR** serves as a shadow register of register **DTM[i]_CH_CTRL2**. The update is then triggered by an edge on one of the selected inputs *DTM_IN0* to *DTM_IN3*.

The synchronous update allows the user to change output polarity, the selection of constant signal level, the constant signal level itself and the switch to/from feed through path on all four channels in parallel synchronized to one of the input edges on *DTM_IN0* to *DTM_IN3*.

14.6 DTM output shut off

A fast shut off for the eight outputs of DTM instance i can be triggered by one of the two assigned inputs TIM[n]_CH_IN or DTM[i]_AUX_IN or the two inputs TIM[m]_CH_IN or DTM[i-1]_AUX_IN of the previous DTM instance i-1. The selection of the trigger signal source is done by the bits **PSU_IN_SEL** and **DTM_SEL(1)** (see figure 14.3.1). The selected trigger signal is named *SHUT_OFF*.
Enabling of the shut off feature is done by setting **UPD_MODE(2:0)** to one of the values 0b001, 0b010 or 0b011.

The shut off behavior of the DTM outputs is defined by the value of register **DTM[i]_CH_CTRL2_SR**.
If the shut off feature is enabled by **UPD_MODE**, as long as the signal *SHUT_OFF_SYNC* is 0, the register **DTM[i]_CH_CTRL2** defines the output signal behavior.
If the signal *SHUT_OFF_SYNC* is 1, the register **DTM[i]_CH_CTRL2_SR** defines the output signal behavior.

The signal *SHUT_OFF_SYNC* is set to 1 if signal *SHUT_OFF* switches to 1. The reset depends on value of **UPD_MODE(2:0)**.

There are three different ways to reset the signal *SHUT_OFF_SYNC* to 0:
- the CPU writes a 1 to bit **SHUT_OFF_RST** of register **DTM_CH_CTRL1**
- synchronous to an edge on DTM channel 0 input of this DTM instance i or on an edge on DTM channel 0 input of preceding DTM instance i-1.
- asynchronous if signal *SHUT_OFF* switches back to 0

Additionally, setting **UPD_MODE(2:0)** to a value 0b000 or 0b1xx resets also the signal *SHUT_OFF_SYNC*.
Figure 14.5.1 depicts the shut off feature and the different shut off release possibilities.
Note: The reset of *SHUT_OFF_SYNC* has lower priority than the set of this signal.

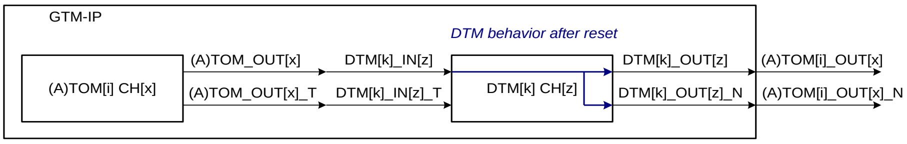
A second shadow register **DTM[i]_CH_SR** exist for the eight **SL** bits (SLx_y_SR) of the shadow register **DTM[i]_CH_CTRL2_SR**.
If enabled by configuration bit **SR_UPD_EN** of register **DTM[i]_CTRL**, the update of **SL** bits of register **DTM[i]_CH_CTRL2_SR** can be triggered by one of the signals selected by bit field **DTM_SEL** of register **DTM[i]_CTRL**. This trigger signal is either the rising or the falling edge detected on *DTM[i]_IN0* of instance i or the rising or the falling edge on *DTM[i-1]_IN0* of preceding instance i-1.

As depicted in figure 14.1.3, 14.1.4 and 14.3.1 the DTM input signal *TIM_CH_IN0, TIM_CH_IN1 or DTM_AUX_IN* can be forwarded to the succeeding instance. Thus, it can be used to trigger shut off in two consecutive DTM instances.

14.7 DTM connections on GTM-IP top level

The DTM, if present, is placed behind the outputs of a TOM or ATOM. The outputs of the DTM are routed directly to the top level ports of GTM-IP.
If there is a DTM placed behind a TOM or ATOM depends on the GTM-IP device configuration.
In case of a DTM behind a TOM or ATOM, the outputs (A)TOM_OUT and (A)TOM_OUT_T are connected to DTM inputs DTM_IN and DTM_IN_T. The outputs of the DTM are routed directly to the top level of GTM-IP.
The behavior of DTM after reset is shown in figure 14.7.1.

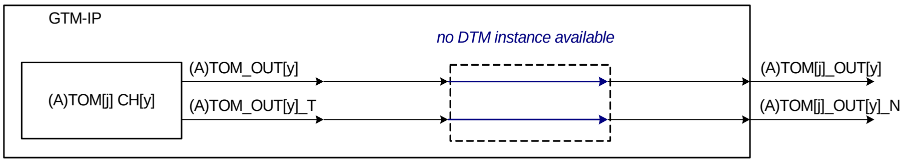
14.7.1 DTM behavior after reset



To route the signal *DTM[k]_IN[z]_T* to the DTM output DTM_[k]_OUT[z]_N, the following DTM channel configuration has to be chosen:
O1F_[x] = 0b11, **O1SEL_[x]**=1 and **DT1_[x]**=1.

The signals names and the signal routing in the case of no DTM instance is placed behind a TOM or ATOM is shown in figure 14.7.2.

14.7.2 (A)TOM output signal routing in case of no DTM instance available



14.8 Configuration Register Overview

Register name	Description	Details in Section
CDTM[i]_DTM[j]_CTRL (j:0...5)	CDTMi DTMj global configuration and control register	14.9.1
CDTM[i]_DTM[j]_CH_CTRL1 (j:0...5)	CDTMi DTMj channel control register 1	14.9.2
CDTM[i]_DTM[j]_CH_CTRL2 (j:0...5)	CDTMi DTMj channel control register 2	14.9.3
CDTM[i]_DTM[j]_CH_CTRL2_SR (j:0...5)	CDTMi DTMj channel control register 2 shadow	14.9.4
CDTM[i]_DTM[j]_CH_CTRL3 (j:0...5)	CDTMi DTMj channel control register 3	14.9.5
CDTM[i]_DTM[j]_PS_CTRL (j:0...5)	CDTMi DTMj phase shift unit configuration and control register	14.9.6
CDTM[i]_DTM[j]_CH[z]_DTV (j=0...5, z:0...3)	CDTMi DTMj dead time reload values	14.9.7
CDTM[i]_DTM[j]_CH_SR(j:0...5)	CDTM DTMj channel shadow register	14.9.8

14.9 Configuration Register Description

14.9.1 Register CDTM[i]_DTM[j]_CTRL (j:0…5)

Address Offset:	see Appendix B																Initial Value:							0x0000_0000								
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved															SHUT_OFF_RST	Reserved							SR_UPD_EN	Reserved	UPD_MODE			DTM_SEL		CLK_SEL	
Mode	R															RAw	R							RW	R	RW			RW		RW	
Initial Value	0x0000															0	000000 0							0	0	000			00		00	

Bit 1:0 **CLK_SEL**: clock source select
0b00 = *SYS_CLK* selected
0b01 = *CMU_CLK0* selected
0b10 = *CMU_CLK1* selected (if DTM is connected to an ATOM)/ *CMU_FXCLK0* selected (if DTM is connected to TOM)
0b11 = *CMU_CLK2* selected (if DTM is connected to an ATOM)/ *CMU_FXCLK1* selected (if DTM is connected to TOM)

Bit 3:2 **DTM_SEL**: select DTM update and SHUT_OFF reset signal
0b00 = select falling edge on DTM[i] channel 0 input
0b01 = select rising edge on DTM[i] channel 0 input
0b10 = select falling edge on DTM[i-1] channel 0 input
0b11 = select rising edge on DTM[i-1] channel 0 input
0b0- = shut off by signal *TIM_CH_IN0*, *TIM_CH_IN1* or *DTM_AUX_IN*
0b1- = shut off by signal *DTM[i-1]_PSU_IN*

Bit 6:4 **UPD_MODE**: update mode
0b000 = asynchronous update - **DTM[i]_CH_CTRL2_SR** not used for update of **DTM[i]_CH_CTRL2**
0b001 = shut off release by writing 1 to bit SHUT_OFF_RST of register **DTM[i]_CTRL**
0b010 = shut off release by an edge on DTM[i]_IN0 or DTM[i-1]_IN0 (defined by bit field DTM_SEL of register **DTM[i]_CTRL**)
0b011 = shut off release by shut off signal *SHUT_OFF* (defined by bits PSU_IN_SEL and IN_POL of register **DTM[i]_PS_CTRL** and DTM_SEL(2) of register **DTM[i]_CTRL**)
0b100 = Signal *IN0_edge* used to trigger update of **DTM[i]_CH_CTRL2** with content of **DTM[i]_CH_CTRL2_SR**
0b101 = Signal *IN1_edge* used to trigger update of **DTM[i]_CH_CTRL2** with content of **DTM[i]_CH_CTRL2_SR**
0b110 = Signal *IN2_edge* used to trigger update of **DTM[i]_CH_CTRL2** with content of **DTM[i]_CH_CTRL2_SR**

0b111= Signal *IN3_edge* used to trigger update of **DTM[i]_CH_CTRL2** with content of **DTM[i]_CH_CTRL2_SR**

Note: If an *INx_edge* is not implemented the value is unused. A write with this unused value returns 0b10 on status.

Bit 7 **Reserved**
Note: Read as zero, should be written as zero

Bit 8 **SR_UPD_EN**: shadow register update enable
0 = no update of SLx_y_SR register bits in register **DTM[i]_CH_CTRL2_SR**
1 = update of SLx_y_SR register bits in register **DTM[i]_CH_CTRL2_SR** on trigger

Bit 15:9 **Reserved**
Note: Read as zero, should be written as zero

Bit 16 **SHUT_OFF_RST**: shut off reset
Writing a 1 releases shut off (resets signal *SHUT_OFF_SYNC* if selected by UPD_MODE(2:0)=0b001)

Bit 31:17 **Reserved**
Note: Read as zero, should be written as zero

14.9.2 Register CDTM[i]_DTM[j]_CH_CTRL1 (j:0...5)

<table>
<tr><td><b>Address Offset:</b></td><td colspan="16"><b>see Appendix B</b></td><td colspan="16"><b>Initial Value: 0x0000_0000</b></td></tr>
<tr><td></td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>Bit</td><td colspan="2">Reserved</td><td colspan="2">O1F_3</td><td>SWAP_3</td><td>SH_EN_3</td><td>I1SEL_3</td><td>O1SEL_3</td><td>Reserved</td><td>XDT_EN_2_3</td><td colspan="2">O1F_2</td><td>SWAP_2</td><td>SH_EN_2</td><td>I1SEL_2</td><td>O1SEL_2</td><td colspan="2">Reserved</td><td colspan="2">O1F_1</td><td>SWAP_1</td><td>SH_EN_1</td><td>I1SEL_1</td><td>O1SEL_1</td><td>Reserved</td><td>XDT_EN_0_1</td><td colspan="2">O1F_0</td><td>SWAP_0</td><td>Reserved</td><td>I1SEL_0</td><td>O1SEL_0</td></tr>
<tr><td>Mode</td><td colspan="2">R</td><td colspan="2">RW</td><td>RW</td><td>RW</td><td>RW</td><td>RW</td><td>R</td><td>RW</td><td colspan="2">RW</td><td>RW</td><td>RW</td><td>RW</td><td>RW</td><td colspan="2">R</td><td colspan="2">RW</td><td>RW</td><td>RW</td><td>RW</td><td>RW</td><td>R</td><td>RW</td><td colspan="2">RW</td><td>RW</td><td>R</td><td>RW</td><td>RW</td></tr>
<tr><td>Initial Value</td><td colspan="2">00</td><td colspan="2">00</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td colspan="2">00</td><td>0</td><td>0</td><td>0</td><td>0</td><td colspan="2">00</td><td colspan="2">00</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td colspan="2">00</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
</table>

Bit 0 **O1SEL_0**: output 1 select channel 0
0 = inverse dead time signal selected
1 = special function on output 1 selected (defined by **O1F_0**)

Bit 1 **I1SEL_0**: input 1 select channel 0
0 = signal *SHIFT0* selected
1 = signal *COUT3* from DTM[i-1] selected
Note: If i is even I1SEL_0 is not implemented. Then the bit is read as zero and should be written as zero.

Bit 2 **Reserved**
Note: Read as zero, should be written as zero

Bit 3 **SWAP_0:** swap outputs DTM[i]_CH[0]_OUT0 and DTM[i]_CH[0]_OUT1 (before final output register)
0 = outputs not swapped
1 = swap outputs *DTM[i]_OUT0* and *DTM[i]_OUT0_N*

Bit 5:4 **O1F_0**: output 1 function channel 0
0b00 = Signal edge_trigg is selected
0b01 = XOR of *DTM[i]_IN0* and signal *SHIFT0*
0b10 = AND of *DTM[i]_IN0* and signal *SHIFT0*
0b11 = *DTM[i]_IN0_T* selected

Bit 6 **XDT_EN_0_1**: cross dead time enable on channel 0 and 1
0 = cross dead time disabled on channel 0 and 1
1 = cross dead time enabled on channel 0 and 1
Note: When a '1' is written to bit XDT_EN_0_1, the internal register *IN[x]_dly1*, *IN[x]_dly2* and *DT_DOWN_CNT* is reset to '0' (x:0,1)

Note: TSEL0_[x] and SH_EN_1 must be '0' for using cross dead time to avoid wrong input signals. (x:0,1)

Bit 7 **Reserved**
Note: Read as zero, should be written as zero

Bit 8 **O1SEL_1**: output 1 select channel 1
0 = inverse dead time signal selected
1 = special function on output 1 selected (defined by **O1F_1**)

Bit 9 **I1SEL_1**: input 1 select channel 1
0 = signal *SHIFT1* selected
1 = signal *OUT1* selected

Bit 10 **SH_EN_1**: shift enable channel 1
0 = *DTM[i]_IN0* is not used; no input signal shift
1 = signal selected by **I1SEL_1** triggers update of *DTM[i]_IN1* with input of *DTM[i]_IN0* -> input signal shift

Bit 11 **SWAP_1:** swap outputs DTM[i]_CH[1]_OUT0 and DTM[i]_CH[1]_OUT1 (before final output register)
0 = outputs not swapped
1 = swap outputs *DTM[i]_OUT1* and *DTM[i]_OUT1_N*

Bit 13:12 **O1F_1**: output 1 function channel 1
0b00 = Signal edge_trigg is selected
0b01 = XOR of *DTM[i]_IN1* and signal *SHIFT1/OUT0*
0b10 = AND of *DTM[i]_IN1* and signal *SHIFT1/OUT0*
0b11 = *DTM[i]_IN1_T* selected

Bit 15:14 **Reserved**
Note: Read as zero, should be written as zero

Bit 16 **O1SEL_2**: output 1 select channel 2

	0 = inverse dead time signal selected 1 = special function on output 1 selected (defined by **O1F_2**)
Bit 17	**I1SEL_2**: input 1 select channel 2 0 = signal *SHIFT1* selected 1 = signal *OUT1* selected
Bit 18	**SH_EN_2**: shift enable channel 2 0 = *DTM[i]_IN1* is not used; no input signal shift 1 = signal selected by **I1SEL_2** triggers update of *DTM[i]_IN2* with input of *DTM[i]_IN1* -> input signal shift
Bit 19	**SWAP_2:** swap outputs DTM[i]_CH[2]_OUT0 and DTM[i]_CH[2]_OUT1 (before final output register) 0 = outputs not swapped 1 = swap outputs *DTM[i]_OUT2* and *DTM[i]_OUT2_N*
Bit 21:20	**O1F_2**: output 1 function channel 2 0b00 = Signal edge_trigg is selected 0b01 = XOR of *DTM[i]_IN2* and signal *SHIFT2/OUT1* 0b10 = AND of *DTM[i]_IN2* and signal *SHIFT2/OUT1* 0b11 = *DTM[i]_IN2_T* selected
Bit 22	**XDT_EN_2_3**: cross dead time enable on channel 0 and 1 0 = cross dead time disabled on channel 2 and 3 1 = cross dead time enabled on channel 2 and 3 Note: When a '1' is written to bit XDT_EN_2_3, the internal register *IN[x]_dly1*, *IN[x]_dly2* and *DT_DOWN_CNT* is reset to '0' (x:2,3) Note: TSEL0_[x] and SH_EN_[x] must be '0' for using cross dead time to avoid wrong input signals. (x:2,3)
Bit 23	**Reserved** Note: Read as zero, should be written as zero
Bit 24	**O1SEL_3**: output 1 select channel 3 0 = inverse dead time signal selected 1 = special function on output 1 selected (defined by **O1F_3**)
Bit 25	**I1SEL_3**: input 1 select channel 3 0 = signal *SHIFT2* selected 1 = signal *OUT2* selected
Bit 26	**SH_EN_3**: shift enable channel 3 0 = *DTM[i]_IN2* is not used; no input signal shift 1 = signal selected by **I1SEL_3** triggers update of *DTM[i]_IN3* with input of *DTM[i]_IN2* -> input signal shift
Bit 27	**SWAP_3:** swap outputs DTM[i]_CH[3]_OUT0 and DTM[i]_CH[3]_OUT1 (before final output register) 0 = outputs not swapped 1 = swap outputs *DTM[i]_OUT3* and *DTM[i]_OUT3_N*

Bit 29:28 **O1F_3**: output 1 function channel 3
0b00 = Signal edge_trigg is selected
0b01 = XOR of *DTM[i]_IN3* and signal *SHIFT3/OUT2*
0b10 = AND of *DTM[i]_IN3* and signal *SHIFT3/OUT2*
0b11 = *DTM[i]_IN3_T* selected

Bit 31:30 **Reserved**
Note: Read as zero, should be written as zero

14.9.3 Register CDTM[i]_DTM[j]_CH_CTRL2 (j:0...5)

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_0000**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	DT1_3	SL1_3	OC1_3	POL1_3	DT0_3	SL0_3	OC0_3	POL0_3	DT1_2	SL1_2	OC1_2	POL1_2	DT0_2	SL0_2	OC0_2	POL0_2	DT1_1	SL1_1	OC1_1	POL1_1	DT0_1	SL0_1	OC0_1	POL0_1	DT1_0	SL1_0	OC1_0	POL1_0	DT0_0	SL0_0	OC0_0	POL0_0
Mode	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Initial Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit 0 **POL0_0**: polarity on output 0 channel 0
0 = Output signal not inverted
1 = Output signal inverted

Bit 1 **OC0_0**: output 0 control channel 0
0 = Functional output
1 = Constant output defined by **SL0_0**

Bit 2 **SL0_0**: signal level on output 0 channel 0
0 = Signal Level is 0 on output if **OC0_0**=1
1 = Signal Level is 1 on output if **OC0_0**=1

Bit 3 **DT0_0**: dead time path enable on output 0 channel 0
0 = feed through from *DTM_IN0* to *DTM[i]_OUT0* enabled
1 = dead time path enabled

Bit 4 **POL1_0**: polarity on output 1 channel 0
0 = Output signal not inverted
1 = Output signal inverted

Bit 5 **OC1_0**: output 1 control channel 0
0 = Functional output
1 = Constant output defined by **SL1_0**

Bit 6 **SL1_0**: signal level on output 1 channel 0
0 = Signal Level is 0 on output if **OC1_0**=1
1 = Signal Level is 1 on output if **OC1_0**=1

Bit 7 **DT1_0**: dead time path enable on output 1 channel 0
0 = feed through from *DTM_IN0* to *DTM[i]_OUT0_N* enabled
1 = dead time path enabled

Bit 8 **POL0_1**: polarity on output 0 channel 1
0 = Output signal not inverted
1 = Output signal inverted

Bit 9 **OC0_1**: output 0 control channel 1
0 = Functional output
1 = Constant output defined by **SL0_1**

Bit 10 **SL0_1**: signal level on output 0 channel 1
0 = Signal Level is 0 on output if **OC0_1**=1
1 = Signal Level is 1 on output if **OC0_1**=1

Bit 11 **DT0_1**: dead time path enable on output 0 channel 1
0 = feed through from *DTM_IN1* to *DTM[i]_OUT1* enabled
1 = dead time path enabled

Bit 12 **POL1_1**: polarity on output 1 channel 1
0 = Output signal not inverted
1 = Output signal inverted

Bit 13 **OC1_1**: output 1 control channel 1
0 = Functional output
1 = Constant output defined by **SL1_1**

Bit 14 **SL1_1**: signal level on output 1 channel 1
0 = Signal Level is 0 on output if **OC1_1**=1
1 = Signal Level is 1 on output if **OC1_1**=1

Bit 15 **DT1_1**: dead time path enable on output 1 channel 1
0 = feed through from *DTM_IN1* to *DTM[i]_OUT1_N* enabled
1 = dead time path enabled

Bit 16 **POL0_2**: polarity on output 0 channel 2
0 = Output signal not inverted
1 = Output signal inverted

Bit 17 **OC0_2**: output 0 control channel 2
0 = Functional output
1 = Constant output defined by **SL0_2**

Bit 18 **SL0_2**: signal level on output 0 channel 2
0 = Signal Level is 0 on output if **OC0_2**=1
1 = Signal Level is 1 on output if **OC0_2**=1

Bit 19 **DT0_2**: dead time path enable on output 0 channel 2
0 = feed through from *DTM_IN2* to *DTM[i]_OUT2* enabled
1 = dead time path enabled

Bit 20 **POL1_2**: polarity on output 1 channel 2
0 = Output signal not inverted
1 = Output signal inverted

Bit 21 **OC1_2**: output 1 control channel 2
0 = Functional output
1 = Constant output defined by **SL1_2**

Bit 22 **SL1_2**: signal level on output 1 channel 2
0 = Signal Level is 0 on output if **OC1_2**=1

	1 = Signal Level is 1 on output if **OC1_2**=1
Bit 23	**DT1_2**: dead time path enable on output 1 channel 2 0 = feed through from *DTM_IN2* to *DTM[i]_OUT2_N* enabled 1 = dead time path enabled
Bit 24	**POL0_3**: polarity on output 0 channel 3 0 = Output signal not inverted 1 = Output signal inverted
Bit 25	**OC0_3**: output 0 control channel 3 0 = Functional output 1 = Constant output defined by **SL0_3**
Bit 26	**SL0_3**: signal level on output 0 channel 3 0 = Signal Level is 0 on output if **OC0_3**=1 1 = Signal Level is 1 on output if **OC0_3**=1
Bit 27	**DT0_3**: dead time path enable on output 0 channel 3 0 = feed through from *DTM_IN3* to *DTM[i]_OUT3* enabled 1 = dead time path enabled
Bit 28	**POL1_3**: polarity on output 1 channel 3 0 = Output signal not inverted 1 = Output signal inverted
Bit 29	**OC1_3**: output 1 control channel 3 0 = Functional output 1 = Constant output defined by **SL1_3**
Bit 30	**SL1_3**: signal level on output 1 channel 3 0 = Signal Level is 0 on output if **OC1_3**=1 1 = Signal Level is 1 on output if **OC1_3**=1
Bit 31	**DT1_3**: dead time path enable on output 1 channel 3 0 = feed through from *DTM_IN3* to *DTM[i]_OUT3_N* enabled 1 = dead time path enabled

14.9.4 Register CDTM[i]_DTM[j]_CH_CTRL2_SR (j:0...5)

Address Offset:	see Appendix B																Initial Value: 0x0000_0000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	DT1_3_SR	SL1_3_SR	OC1_3_SR	POL1_3_SR	DT0_3_SR	SL0_3_SR	OC0_3_SR	POL0_3_SR	DT1_2_SR	SL1_2_SR	OC1_2_SR	POL1_2_SR	DT0_2_SR	SL0_2_SR	OC0_2_SR	POL0_2_SR	DT1_1_SR	SL1_1_SR	OC1_1_SR	POL1_1_SR	DT0_1_SR	SL0_1_SR	OC0_1_SR	POL0_1_SR	DT1_0_SR	SL1_0_SR	OC1_0_SR	POL1_0_SR	DT0_0_SR	SL0_0_SR	OC0_0_SR	POL0_0_SR
Mode	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Initial Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit 0 **POL0_0_SR**: polarity on output 0 channel 0 shadow register

0 = Output signal not inverted
1 = Output signal inverted

Bit 1 **OC0_0_SR**: output 0 control channel 0 shadow register
0 = Functional output
1 = Constant output defined by **SL0_0**

Bit 2 **SL0_0_SR**: signal level on output 0 channel 0 shadow register
0 = Signal Level is 0 on output if **OC0_0**=1
1 = Signal Level is 1 on output if **OC0_0**=1

Bit 3 **DT0_0_SR**: dead time path enable on output 0 channel 0 shadow register
0 = feed through from *DTM_IN0* to *DTM[i]_OUT0* enabled
1 = dead time path enabled

Bit 4 **POL1_0_SR**: polarity on output 1 channel 0 shadow register
0 = Output signal not inverted
1 = Output signal inverted

Bit 5 **OC1_0_SR**: output 1 control channel 0 shadow register
0 = Functional output
1 = Constant output defined by **SL1_0**

Bit 6 **SL1_0_SR**: signal level on output 1 channel 0 shadow register
0 = Signal Level is 0 on output if **OC1_0**=1
1 = Signal Level is 1 on output if **OC1_0**=1

Bit 7 **DT1_0_SR**: dead time path enable on output 1 channel 0 shadow register
0 = feed through from *DTM_IN0* to *DTM[i]_OUT0_N* enabled
1 = dead time path enabled

Bit 8 **POL0_1_SR**: polarity on output 0 channel 1 shadow register
0 = Output signal not inverted
1 = Output signal inverted

Bit 9 **OC0_1_SR**: output 0 control channel 1 shadow register
0 = Functional output
1 = Constant output defined by **SL0_1**

Bit 10 **SL0_1_SR**: signal level on output 0 channel 1 shadow register
0 = Signal Level is 0 on output if **OC0_1**=1
1 = Signal Level is 1 on output if **OC0_1**=1

Bit 11 **DT0_1_SR**: dead time path enable on output 0 channel 1 shadow register
0 = feed through from *DTM_IN1* to *DTM[i]_OUT1* enabled
1 = dead time path enabled

Bit 12 **POL1_1_SR**: polarity on output 1 channel 1 shadow register
0 = Output signal not inverted
1 = Output signal inverted

Bit 13 **OC1_1_SR**: output 1 control channel 1 shadow register
0 = Functional output
1 = Constant output defined by **SL1_1**

Bit 14 **SL1_1_SR**: signal level on output 1 channel 1 shadow register
0 = Signal Level is 0 on output if **OC1_1**=1
1 = Signal Level is 1 on output if **OC1_1**=1

Bit 15 **DT1_1_SR**: dead time path enable on output 1 channel 1 shadow register
0 = feed through from *DTM_IN1* to *DTM[i]_OUT1_N*
1 = dead time path enabled

Bit 16 **POL0_2_SR**: polarity on output 0 channel 2 shadow register
0 = Output signal not inverted
1 = Output signal inverted

Bit 17 **OC0_2_SR**: output 0 control channel 2 shadow register
0 = Functional output
1 = Constant output defined by **SL0_2**

Bit 18 **SL0_2_SR**: signal level on output 0 channel 2 shadow register
0 = Signal Level is 0 on output if **OC0_2**=1
1 = Signal Level is 1 on output if **OC0_2**=1

Bit 19 **DT0_2_SR**: dead time path enable on output 0 channel 2 shadow register
0 = feed through from *DTM_IN2* to *DTM[i]_OUT2*
1 = dead time path enabled

Bit 20 **POL1_2_SR**: polarity on output 1 channel 2 shadow register
0 = Output signal not inverted
1 = Output signal inverted

Bit 21 **OC1_2_SR**: output 1 control channel 2 shadow register
0 = Functional output
1 = Constant output defined by **SL1_2**

Bit 22 **SL1_2_SR**: signal level on output 1 channel 2 shadow register
0 = Signal Level is 0 on output if **OC1_2**=1
1 = Signal Level is 1 on output if **OC1_2**=1

Bit 23 **DT1_2_SR**: dead time path enable on output 1 channel 2 shadow register
0 = feed through from *DTM_IN2* to *DTM[i]_OUT2_N*
1 = dead time path enabled

Bit 24 **POL0_3_SR**: polarity on output 0 channel 3 shadow register
0 = Output signal not inverted
1 = Output signal inverted

Bit 25 **OC0_3_SR**: output 0 control channel 3 shadow register
0 = Functional output
1 = Constant output defined by **SL0_3**

Bit 26 **SL0_3_SR**: signal level on output 0 channel 3 shadow register
0 = Signal Level is 0 on output if **OC0_3**=1
1 = Signal Level is 1 on output if **OC0_3**=1

Bit 27 **DT0_3_SR**: dead time path enable on output 0 channel 3 shadow register

0 = feed through from *DTM_IN3* to *DTM[i]_OUT3*
1 = dead time path enabled

Bit 28 **POL1_3_SR**: polarity on output 1 channel 3 shadow register
0 = Output signal not inverted
1 = Output signal inverted

Bit 29 **OC1_3_SR**: output 1 control channel 3 shadow register
0 = Functional output
1 = Constant output defined by **SL1_3**

Bit 30 **SL1_3_SR**: signal level on output 1 channel 3 shadow register
0 = Signal Level is 0 on output if **OC1_3**=1
1 = Signal Level is 1 on output if **OC1_3**=1

Bit 31 **DT1_3_SR**: dead time path enable on output 1 channel 3 shadow register

0 = feed through from *DTM_IN3* to *DTM[i]_OUT3_N*
1 = dead time path enabled

14.9.5 Register CDTM[i]_DTM[j]_CH_CTRL3 (j:0...5)

Address Offset:	**see Appendix B**																**Initial Value: 0x0000_0000**															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved				TSEL1_3	TSEL0_3	CIS3	CII3	Reserved				TSEL1_2	TSEL0_2	CIS2	CII2	Reserved				TSEL1_1	TSEL0_1	CIS1	CII1	Reserved				TSEL1_0	TSEL0_0	CIS0	CII0
Mode	R				RW	RW	RW	RW	R				RW	RW	RW	RW	R				RW	RW	RW	RW	R				RW	RW	RW	RW
Initial Value	0x0				0	0	0	0	0x0				0	0	0	0	0x0				0	0	0	0	0x0				0	0	0	0

Bit 0 **CII0**: combinational input invert channel 0
0 = do not invert input
1 = invert input

Bit 1 **CIS0**: combinational input select channel 0
0 = select input *DTM[i]_IN0*
1 = select internal signal *edge_trigg_0*

Bit 2 **TSEL0_0**: input selection for dead time / edge trigger generation
0 = use *DTM[i]_IN0* as input for dead time / edge trigger generation
1 = use *DTM[i]_IN0_T* as input for dead time / edge trigger generation

Bit 3 **TSEL1_0**: input selection combinational logic path
0 = use *DTM[i]_IN0* as input for combinational logic path
1 = use *DTM[i]_IN0_T* as input for combinational logic path

Bit 7:4 **Reserved**
Note: Read as zero, should be written as zero

Bit 8 **CII1**: combinational input invert channel 1
0 = do not invert input
1 = invert input

Bit 9 **CIS1**: combinational input select channel 1
0 = select input *DTM[i]_IN1*
1 = select internal signal *edge_trigg_1*

Bit 10 **TSEL0_1**: input selection for dead time / edge trigger generation
0 = use *DTM[i]_IN1* as input for dead time / edge trigger generation
1 = use *DTM[i]_IN1_T* as input for dead time / edge trigger generation

Bit 11 **TSEL1_1**: input selection combinational logic path

0 = use *DTM[i]_IN1* as input for combinational logic path
1 = use *DTM[i]_IN1_T* as input for combinational logic path

Bit 15:12 **Reserved**
Note: Read as zero, should be written as zero

Bit 16 **CII2**: combinational input invert channel 2
0 = do not invert input
1 = invert input

Bit 17 **CIS2**: combinational input select channel 2
0 = select input *DTM[i]_IN2*
1 = select internal signal *edge_trigg_2*

Bit 18 **TSEL0_2**: input selection for dead time / edge trigger generation
0 = use *DTM[i]_IN2* as input for dead time / edge trigger generation
1 = use *DTM[i]_IN2_T* as input for dead time / edge trigger generation

Bit 19 **TSEL1_2**: input selection combinational logic path
0 = use *DTM[i]_IN2* as input for combinational logic path
1 = use *DTM[i]_IN2_T* as input for combinational logic path

Bit 23:20 **Reserved**
Note: Read as zero, should be written as zero

Bit 24 **CII3**: combinational input invert channel 3
0 = do not invert input
1 = invert input

Bit 25 **CIS3**: combinational input select channel 3
0 = select input *DTM[i]_IN3*
1 = select internal signal *edge_trigg_3*

Bit 26 **TSEL0_3**: input selection for dead time / edge trigger generation
0 = use *DTM[i]_IN3* as input for dead time / edge trigger generation
1 = use *DTM[i]_IN3_T* as input for dead time / edge trigger generation

Bit 27 **TSEL1_3**: input selection combinational logic path
0 = use *DTM[i]_IN3* as input for combinational logic path
1 = use *DTM[i]_IN3_T* as input for combinational logic path

Bit 31:28 **Reserved**
Note: Read as zero, should be written as zero

14.9.6 Register CDTM[i]_DTM[j]_PS_CTRL (j:0...5)

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_0000**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved										SHIFT_SEL		Reserved	TIM_SEL	IN_POL	PSU_IN_SEL	Reserved						RELBLK									
Mode	R										RW		R	RW	RW	RW	R						RW									
Initial Value	0x000										0b00		0	0	0	0	0x00						0x000									

Bit 9:0 **RELBLK:** reload value blanking window
Note: a value of 0x000 resets counter BLK_DOWN_CNT

Bit 15:10 **Reserved**
Note: Read as zero, should be written as zero

Bit 16 **PSU_IN_SEL**: PSU input select
0 = *TIM_CH_IN0* or *TIM_CH_IN1* selected
1 = *DTM_AUX_IN* selected

Bit 17 **IN_POL**: input polarity
0 = input signal is not inverted
1 = input signal is inverted

Bit 18 **TIM_SEL:** TIM input select
0 = select TIM_IN0
1 = select TIM_IN1

Bit 19 **Reserved**
Note: Read as zero, should be written as zero

Bit 21:20 **SHIFT_SEL**: shift select
0b00 = DTM channel 1 is connected via signal *SHIFT1* with *TIM_CH_IN0, TIM_CH_IN1 or DTM_AUX_IN*
0b01 = DTM channel 2 is connected via signal *SHIFT2* with *TIM_CH_IN0, TIM_CH_IN1 or DTM_AUX_IN*
0b10 = DTM channel 3 is connected via signal *SHIFT3* with *TIM_CH_IN0, TIM_CH_IN1 or DTM_AUX_IN*
0b11 = DTM channel 0 is connected via signal *SHIFT0* with *TIM_CH_IN0, TIM_CH_IN1 or DTM_AUX_IN*

Note: If a channel is not implemented the value is unused. A write with this unused value returns "10" on status.

Bit 31:22 **Reserved**
Note: Read as zero, should be written as zero

14.9.7 Register CDTM[i]_DTM[j]_CH[z]_DTV (j:0...5, z:0...3)

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved						RELFALL										Reserved						RELRISE									
Mode	R						RW										R						RW									
Initial Value	0x00						0x000										0x00						0x000									

Bit 9:0 **RELRISE**: reload value for rising edge dead time

Bit 15:10 **Reserved**
Note: Read as zero, should be written as zero

Bit 25:16 **RELFALL**: reload value for falling edge dead time

Bit 31:26 **Reserved**
Note: Read as zero, should be written as zero

14.9.8 Register CDTM[i]_DTM[j]_CH_SR (j:0...5)

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																								SL1_3_SR_SR	SL0_3_SR_SR	SL1_2_SR_SR	SL0_2_SR_SR	SL1_1_SR_SR	SL0_1_SR_SR	SL1_0_SR_SR	SL0_0_SR_SR
Mode	R																								RW	RW	RW	RW	RW	RW	RW	RW
Initial Value	0x0000 00																								0	0	0	0	0	0	0	0

Bit 0 **SL0_0_SR_SR**: shadow register for bit SL0_0_SR of register DTM[i]_CH_CTRL2_SR

Bit 1 **SL1_0_SR_SR**: shadow register for bit SL1_0_SR of register DTM[i]_CH_CTRL2_SR

Bit 2 **SL0_1_SR_SR**: shadow register for bit SL0_1_SR of register DTM[i]_CH_CTRL2_SR

Bit 3 **SL1_1_SR_SR**: shadow register for bit SL1_1_SR of register DTM[i]_CH_CTRL2_SR

Bit 4 **SL0_2_SR_SR**: shadow register for bit SL0_2_SR of register DTM[i]_CH_CTRL2_SR

Bit 5 **SL1_2_SR_SR**: shadow register for bit SL1_2_SR of register DTM[i]_CH_CTRL2_SR

Bit 6 **SL0_3_SR_SR**: shadow register for bit SL0_3_SR of register DTM[i]_CH_CTRL2_SR

Bit 7 **SL1_3_SR_SR**: shadow register for bit SL1_3_SR of register DTM[i]_CH_CTRL2_SR

Bit 31:8 **Reserved**
Note: Read as zero, should be written as zero

15 Multi Channel Sequencer (MCS)

15.1 Overview

The Multi Channel Sequencer (MCS) sub module is a generic data processing module that is connected to the ARU. One of its major applications is to calculate complex output sequences that may depend on the time base values of the TBU and are processed in combination with the ATOM sub module. Other applications can use the MCS sub module to perform extended data processing of input data resulting from the TIM sub module. Moreover, some applications may process data provided by the CPU within the MCS sub module, and the calculated results are sent to the outputs using the ATOM sub modules.

Table 15.1.1 summarizes all available generic design parameters of the MCS hardware structure.

15.1.1 Generic Design Parameters

Design Parameter	Description
W	Word width of the data path
T	Number of available MCS channels
RDW	RAM data width of connected RAM
RAW	RAM address width used by the MCS for addressing memory
USR	Use second RAM port (0 - one RAM port available, 1 - two RAM ports available)
BAW	Bus Master Address Width
BDW	Bus Master Data Width
URIP	Use RAM input pipeline registers (0 - no register, 1 - use register)
UROP	Use RAM output pipeline registers (0 - no register , 1 - use register)

UDP	Use Decoder Pipeline register (0 - no register , 1 - use register)
UAP	Use ALU Pipeline register (0 - no register , 1 - use register)
NPS	Total number of pipeline stages (with NPS = 3 + URIP + UROP + UDP + UAP)

All MCS instances in the GTM use the values T=8, W=24, RDW=32, RAW=12, USR=1, BAW = 14, BDW = 32, URIP = 1, UROP = 1, UDP = 1, UAP = 1, and NPS = 7.

15.2 Architecture

15.2.1 MCS Architecture

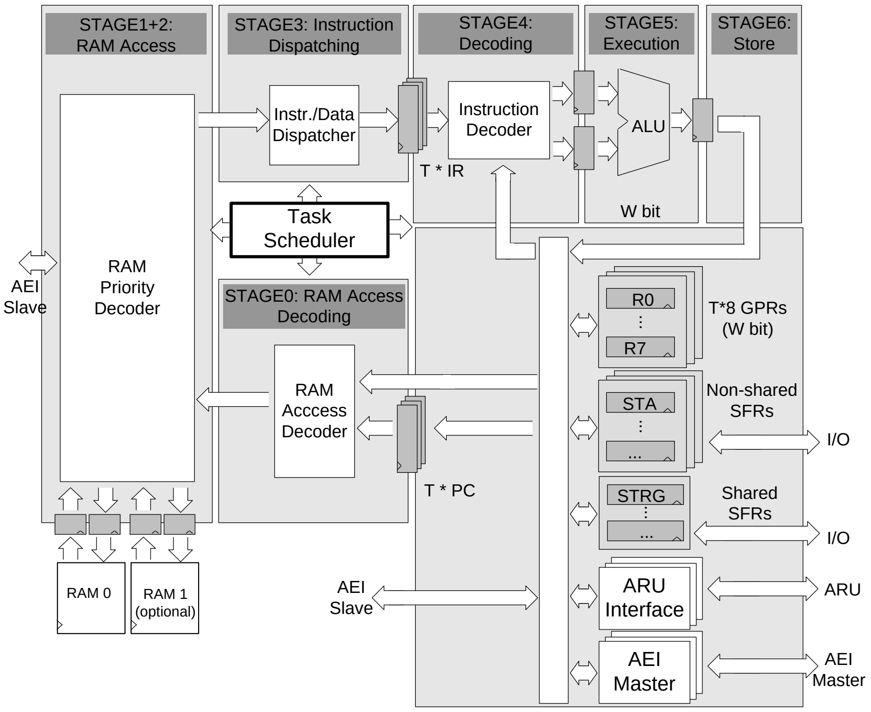


Figure 15.2.1 gives an overview of the MCS architecture assuming that all pipeline registers are implemented.

The data path of the MCS is shared by T so called MCS-channels, whereas each MCS-channel executes a dedicated micro-program that is stored inside the RAM connected to the MCS module.

The connected RAM may contain arbitrary sized code and data sections that are accessible by all MCS-channels and an externally connected master (e.g. a CPU) via AEI Slave interface. More details about the RAM can be found in section 15.4.

An MCS-channel can also be considered as an individual task of a processor that is scheduled to the commonly used data path at a specific point in time. The execution of the different MCS-channels on the different pipeline stages is controlled by a central hardware related task scheduler, which enables immediate task switches in parallel to the program execution. Details about the task scheduler and the available scheduling algorithms can be found in section 15.3.

Typically, if data has to be exchanged between different MCS-channels and/or the CPU, the connected RAM, which is accessible by all MCS-channels and the CPU, can be used.

Besides the commonly used data path, each MCS channel has

- a set of eight General Purpose Registers (GPRs), each W bit wide,
- a set of non-shared Special Function Registers (SFRs) that are only accessible within a dedicated MCS channels,
- a set of shared SFRs that are accessible by all MCS channels,
- a channel specific instruction register (IR),
- a channel specific program counter register (PC),
- a dedicated ARU interface for communication with other ARU connected modules,
- and an AEI Bus Master Interface for controlling other GTM sub modules.

Generally, the GPRs of an MCS channel x are only accessible by its corresponding MCS channel x. However, the MCS provides a configuration that allows an MCS channel x to access the GPRs of its successor MCS channel x+1. This feature can be used to enlarge the number of registers for a specific MCS channel x and/or to exchange data between neighboring channels.

For safety reasons, the register **MCS[i]_REG_PROT** can be used to define write protections for the neighboring registers of the individual MCS channels.

In order to enable synchronization between different MCS channels and/or the CPU, the MCS provides a common 24 bit wide trigger register that can be accessed as a shared SFR by all MCS channels located in the same module. Writing to **STRG** sets bits and writing to **CTRG** clears bits in the common trigger register. To enable triggering of MCS-channels by CPU, the CPU can set bits in the common trigger register by writing to **MCS[i]_STRG** and clear bits by writing to **MCS[i]_CTRG**.

Considering the architecture in the figure above and assuming that all available pipeline stages are implemented (the generic parameters URIP, UROP, UDP, and UAP are set to 1), the main actions of the different pipeline stages are as follows:

- Pipeline stage 0 performs a setup of address, input data, and control signals for the next RAM access of a specific MCS-channel.
- The actual RAM access of a specific MCS-channel is executed in pipeline stage 1 and 2, assuming an external connection of a synchronous RAM with a latency of one clock cycle.
- Pipeline stage 3 performs pre-decoding and dispatching of instructions and data resulting from the RAM.
- In pipeline stage 4 the instructions are decoded and data from the registers are loaded.
- After that, in pipeline stage 5 the instruction is executed meaning that arithmetic operations are applied.
- Finally, in pipeline stage 6 the calculated results are stored in the registers.

If any of the pipeline registers is not implemented, the adjacent pipeline stages are merged and thus processed within the same clock cycle.

The RAM priority decoder arbitrates RAM accesses that are requested by the CPU via AEI and by the active MCS-channel. If both, CPU and an MCS-channel request a memory access to the same memory module the MCS-channel is prioritized.

Since the internal registers of the MCS can be updated by different sources (MCS write access by various instructions, CPU write access via AEI slave, MCS write access by neighboring channel) a write conflict occurs if more than one source wants to write to the same register. In this case the result of the register is unpredictable. However, the software should setup its application in a way that such conflicts do not occur.

One exception is the common trigger register, which may be written by multiple sources (different MCS channels and CPU) in order to enable triggering of different MCS channels. Typically, the software should setup its application in a manner that different sources should not write the same bits in the trigger register.

15.3 Scheduling

The MCS provides a hardware related task scheduler, which globally controls the execution of the tasks in the different pipeline stages. The task scheduler implements four different scheduling modes, that can be selected by the **SCD_MODE** bit field in the **MCS[i]_CTRL_STAT** register. Depending on the selected scheduling mode, the task scheduler is selecting a dedicated MCS channel that will be executed in pipeline stage 0 in the next clock cycle. Additionally, MCS channels that are already present in the pipeline are shifted to its successor pipeline stage, with each clock cycle. This means, that the execution time of an MCS-channel in a specific pipeline stage is always one clock cycle.

The MCS task scheduler may also schedule an empty cycle to pipeline stage 0, in order to grant a time slice to the CPU for accessing the connected RAM.

It should be noted, if the task scheduler assigns an MCS channel to pipeline stage 0, but this channel does not access the RAM, the CPU can access the corresponding RAM, even if the scheduler did not reserve an empty clock cycle.

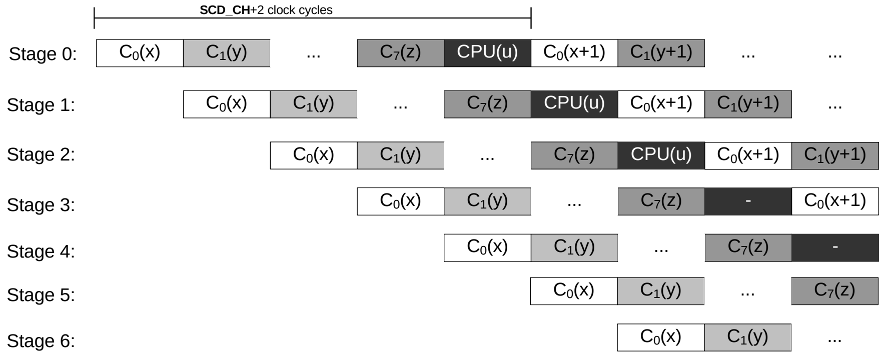
In the following, the available scheduling modes are described.

15.3.1 Round Robin Scheduling

The Round Robin Scheduling Mode implements the simplest scheduling algorithm. This algorithm schedules a predefined set of MCS channels in the range [0; **SCD_CH**] in ascending order. After the last channel **SCD_CH** has been assigned to the pipeline, an empty cycle is scheduled in order to enable RAM access for the CPU. The parameter **SCD_CH** can be controlled by the register **MCS[i]_CTRL_STAT**. If the value of **SCD_CH** is greater than T-1, the scheduler assumes a value of T-1 for bit field **SCD_CH**.

Figure 15.3.1.1 shows a timing example of the Round Robin Scheduling with T=8 MCS-channels (marked as C_0 to C_7) that are scheduled together with a CPU access to a pipeline with and NPS=7 stages. It is assumed that bit field **SCD_CH** is set to 7.

15.3.1.1 Timing of Round Robin Scheduling


SCD_CH+2 clock cycles
Stage 0: C0(x) C1(y) ... C7(z) CPU(u) C0(x+1) C1(y+1) ... ...
Stage 1: C0(x) C1(y) ... C7(z) CPU(u) C0(x+1) C1(y+1) ...
Stage 2: C0(x) C1(y) ... C7(z) CPU(u) C0(x+1) C1(y+1)
Stage 3: C0(x) C1(y) ... C7(z) - C0(x+1)
Stage 4: C0(x) C1(y) ... C7(z) -
Stage 5: C0(x) C1(y) ... C7(z)
Stage 6: C0(x) C1(y) ...

The identifier $C_i(x)$ denotes that MCS-channel i is currently executing the instruction or data located in the memory at position x in the corresponding pipeline stage. The figure shows, which MCS-channel is activated in specific pipeline stage at a specific point in time.

Moreover, the figure shows that the Round Robin scheduling is always repeated after **SCD_CH**+2 clock cycles, which means that the time duration of an instruction cycle is

SCD_CH+2 clock cycles. However, if the value **SCD_CH** + 2 is less than NPS, the duration of an instruction cycle is limited by the depth of the pipeline to NPS clock cycles. Thus the effective execution time of a single cycle instruction is always MIN(**SCD_CH**+2, NPS) clock cycles, ignoring the latency of the pipeline.

If NPS is greater than T+1, NPS-T-1 additional empty cycles are inserted at the end of a round trip cycle. In this case the round trip time for the scheduler is determined by NPS, and thus the time duration for an instruction cycle is always NPS clock cycles.

The Round Robin scheduling algorithm has the characteristic that it fairly distributes all time slices to all MCS-channels and the CPU. This means, that the program execution time of a specific task is independent from the activity of any neighboring task or the CPU RAM access, and thus a correct estimation of the actual program execution time is very easy. However, the round-robin scheduling may waste clock cycles by scheduling MCS-channels that are not ready to execute an instruction (e.g. MCS-channel is disabled by CPU). The following scheduling modes overcome this issue.

15.3.2 Accelerated Scheduling

In order to improve the computational performance, the accelerated scheduling mode provides two key features. Firstly, the scheduler only selects MCS-channels that are not suspended and thus can actually execute an instruction. Secondly, the scheduler applies instruction prefetching to minimize empty cycles in the pipeline. An MCS-channel is entering suspended state due to one of the reasons:

- An MCS-channel is executing a read or write request to an ARU connected sub module (instruction ARD, AWR, ARDI, AWRI, NARD, NARDI).
- An MCS-channel is executing a read or write request at its bus master interface (instruction BRD, BWR, BRDI, BWRI).
- An MCS-channel waits on a register match event (e.g. instruction WURM), in order to wait on a desired register value (e.g. trigger event from another MCS channel).
- An MCS-channel is disabled.

In the case of instruction prefetching, the scheduler will assign an MCS-channel C_p to pipeline stage 0, which is already present in another pipeline stage. This means, that the execution of the last instruction of C_p located in the memory MEM(PC/4) is not yet finished completely, whereas PC is the current value of the program counter of MCS-channel C_p. Thus, the newly scheduled MCS-channel C_p will prefetch a successor instruction MEM(PC/4+PFO) under the assumption that there will be no branch and no memory access in the program between the instructions MEM(PC/4) and MEM(PC/4+PFO). The prefetch offset value PFO is determined by counting the number of already scheduled MCS channels Cp in the pipeline. However, if the assumption fails, the pipeline will be flushed by replacing all MCS-channel C_p of the pipeline with an empty cycle, as soon as the instruction decoder detects a branch or a memory access. All other MCS channels unequal to MCS channel C_p within are not affected by the flushing action. The flushing action is always synchronized to the last pipeline stage NPS-1.

Besides the flushing conditions mentioned above, there exist also other conditions that cause a flush of the pipeline for a specific MCS-channel. In the following all possible flushing events are summarized:

- An MCS-channel is enabled.
- An MCS-channel is entering a suspended state.
- An MCS-channel is taking a conditional or unconditional branch (instruction JMP, JBS, JBC, CALL, RET, JMPI, JBSI, JBCI, CALLI).
- An MCS-channel accessing memory for data transfer (instruction MRD, MWR, MRDI, MWRI, MRDIO, MWRIO, MWRL, MWRIL, PUSH, POP).
- An MCS-channel is executing a read or write request at its bus master interface (instruction BRD, BWR, BRDI, BWRI).
- An MCS-channel is modifying the trigger register (write access to **CTRG** or **STRG**) and the same channel is reading back this register (read access to **CTRG** or **STRG**) while the delay between both accesses is less than UAP+UDP+1 clock cycles.

In general, each MCS-channel can accept instruction prefetching. However, there are some cases in which an upcoming flushing of the pipeline can be easily detected by the MCS hardware due to evaluation of internal states. Therefore, it is defined that an MCS-channel accepts instruction prefetching only under the following conditions:

- An MCS-channel is currently not in the second cycle of a two-cycle control flow instruction (instruction CALL, RET).
- An MCS-channel is currently not in the second cycle of a three-cycle memory access instruction (instruction MWRL, MWRIL).

The accelerated scheduling mode guarantees, that the time duration of an instruction cycle varies between 1 and T+1 cycles. Hence, a single cycle instructions has an effective execution time between 1 to T+1 clock cycles, depending on the number of suspended MCS-channels and the actual instruction sequence. The worst case execution time occurs if all channels are active and the CPU also accesses the RAM. The best case occurs e.g. if only one MCS-channel is enabled and the executed program sequence has only linear code without branches and memory access.

The algorithm of the accelerated scheduling mode first, evaluates the state of all available MCS-channels as well as a CPU request to the RAMs and then it decides if a specific MCS-channel or an empty cycle is assigned to pipeline stage 0 in the next clock cycle. It should be noted that the accelerated scheduling mode treats RAM access requests from the CPU in a similar manner as MCS-channels, which means that empty cycles for RAM requests are only inserted into the pipeline if there is an active RAM request from the CPU or no other task can be scheduled.

In order to fairly trade all available MCS-channels as well as CPU RAM requests and to guarantee a worst case execution time of T+1 clock cycles, an additional task prioritization scheme is applied used that dynamically prioritizes all MCS-channels and a CPU memory access depending on the history of the scheduler's decisions. The algorithm of the accelerated scheduler mode is executed every clock cycle and it works in the following manner:

- Try to find an MCS-channel C_r with highest priority that is not suspended and not already scheduled to the pipeline stages 0 to NPS-2. If C_r is found assign C_r to pipeline stage 0 and finish scheduling for current clock cycle.
- Otherwise, try to find an MCS-channel C_p with highest priority that is not suspended and accepts instruction prefetching. If C_p is found assign C_p to pipeline stage 0 and finish scheduling for current clock cycle.
- Otherwise, try to find an MCS-channel C_s with highest priority that is suspended and accepts instruction prefetching. If C_s is found assign C_s to pipeline stage 0 and finish scheduling for current clock cycle.
- Otherwise, assign an empty cycle to pipeline stage 0 and finish scheduling for current clock cycle.

The underlying task prioritization scheme tracks the history of the scheduled MCS-channels in a list consisting of T+1 items. The list is initialized with all MCS-channels followed by a reserved time slot for the CPU RAM access. The position of an MCS-channel within this list implicitly defines the priority, while the back of this list holds the MCS-channel with highest priority. Whenever the scheduling algorithm described above has found an MCS-channel C_r or C_p to be scheduled in the next clock cycle, it removes this item from the list and put it to the front of the list. In order to fairly prioritize all MCS-channels, the algorithm also removes the item at the back of the list to the second position in the list, after the inserted scheduled front item. Since the list always contains all possible MCS-channels and with each clock cycles each nonscheduled item is moved at least one position towards the end of list, it is obvious that each MCS-channel will have the highest priority not later than T+1 clock cycles.

Figure 15.3.2.1 shows a timing example of the accelerated scheduling with NPS=7 pipeline stages.

15.3.2.1 Timing of Accelerated Scheduling

Cycle:	0	1	2	3	4	5	6	7	8	9	10
Stage 0:	$C_0(x)$	$C_1(y)$	$C_0(x+1)$	$C_1(y+1)$	$C_0(x+2)$	$C_0(x+3)$	$C_0(x+4)$	$C_0(x+5)$	$C_0(x+6)$	$C_0(x+4)$	$C_0(x+5)$
Stage 1:		$C_0(x)$	$C_1(y)$	$C_0(x+1)$	$C_1(y+1)$	$C_0(x+2)$	$C_0(x+3)$	$C_0(x+4)$	$C_0(x+5)$	-	$C_0(x+4)$
Stage 2:			$C_0(x)$	$C_1(y)$	$C_0(x+1)$	$C_1(y+1)$	$C_0(x+2)$	$C_0(x+3)$	$C_0(x+4)$	-	-
Stage 3:				$C_0(x)$	$C_1(y)$	$C_0(x+1)$	$C_1(y+1)$	$C_0(x+2)$	$C_0(x+3)$	-	-
Stage 4:					$C_0(x)$	$C_1(y)$	$C_0(x+1)$	$C_1(y+1)$	$C_0(x+2)$	-	-
Stage 5:						$C_0(x)$	$C_1(y)$	$C_0(x+1)$	-	-	-
Stage 6:							$C_0(x)$	$C_1(y)$	$C_0(x+1)$	-	-

15.3.2.2 MCS Code example for Accelerated Scheduling

MCS-Channel 0

Memory Location	Instruction
x+0	ADDL R0, 7
x+1	JBC STA, Z, x+4
x+2	MOVL R2, 5

MCS-Channel 1

Memory Location	Instruction
y+0	WURM STRG, R1, 0xFFFF
y+1	ADDL R2, 5
y+2	ADD R3, R2

The example assumes that initially MCS-channels 0 and 1 are enabled and the program for each MCS-channel is located in the RAM as shown in Figure 15.3.2.2. Since both channels are ready to run, the scheduler fairly selects the channels in an alternating order, as it can be obtained in stage 0 at the clock cycles before cycle 5. Since MCS-channel 1 is entering suspended state (to wait on a trigger bit) at cycle 7 in stage 6 with the instruction of memory location y, the scheduler will only select MCS-channel 0 in the following by applying instruction prefetching. Moreover, entering the suspended state in channel 1 also flushes the remaining channels 1 out of the pipeline. But it should be noted, the scheduler applies instruction prefetching during the whole sequence, due to the fact that the number of enabled channels is always less than the available number of pipeline stages NPS.

The actual state of pipeline in cycle 9 and 10 depends on conditional branch instruction of memory location x + 3 (cycle 8 stage 6). If the branch is not taken, the linear code execution of MCS-channel 0 is continued. However, if the branch to memory location x+4 is taken, as shown in the Figure, the scheduler will fetch the instruction $C_0(x+4)$ in cycle 9 at stage 0 and flush the stages 1 to NPS-1. Note, the flushing of the pipeline only concerns the prefetched instructions of the MCS-channel that is currently executed in the last stage. If pipeline stage 1 of cycle 9 would belong to another channel than 0, only the stages greater than 2 would have been flushed.

15.3.3 Single Prioritization Scheduling

The Single Prioritization Scheduling mode is an extended variant of the Accelerated Scheduling mode, which additionally applies a task prioritization of a single MCS-channel. In this mode, the bit field **SCD_CH** of register **MCS[i]_CTRL_STAT** is used to identify a dedicated MCS-channel that is always preferred during scheduling. This means, that the scheduler will assign preferred MCS-channel **SCD_CH** to pipeline stage 0, as long as this channel is not suspended. If the preferred MCS-channel is entering its suspended state, the scheduling algorithm switches to the accelerated scheduling as previously described in section 15.3.2. Whenever the MCS-channel **SCD_CH** is resuming from its suspended state, the scheduler switches back and assign the channel **SCD_CH** to pipeline stage 0 until the next suspension event occurs. If the bit field **SCD_CH** contains the value T or higher, the task scheduler will always prioritize CPU access to the RAM. This means, whenever the task scheduler detects that the CPU wants to access an MCS-RAM, the scheduler will assign an empty cycle into pipeline stage 0. If the CPU does not access the RAM any more, it switches back to the accelerated mode, as described previously in section 15.3.2.

In consequence, the Single Prioritization Scheduling mode cannot guarantee a maximum time duration of an instruction cycle for the overall execution of all MCS-channels, since it strongly depends on the activity of the prioritized MCS-channel **SCD_CH**. However, the Single Prioritization Scheduling mode provides the fastest possible execution for MCS-channel **SCD_CH**. Moreover, during the time spawn, in which the prioritized MCS-channel **SCD_CH** is suspended, this mode guarantees a duration of 1 to T+1 clock cycles of an instruction cycle for all non-prioritized channels.

15.3.4 Multiple Prioritization Scheduling

The Multiple Prioritization Scheduling mode is an extended variant of the Accelerated Scheduling mode, which additionally applies a task prioritization for multiple MCS-channels. In this mode, the bit field **SCD_CH** of register **MCS[i]_CTRL_STAT** is used to identify a set of dedicated MCS-channels, which are always preferred during scheduling. The identifiers of the prioritized MCS-channels are in the range [0; **SCD_CH**] and the non-prioritized channels are in the range [**SCD_CH**+1; T-1]. The individual priority for the set of prioritized MCS-channels is applied in descending order, which means that MCS-channel 0 has the highest priority, followed MCS-channel 1, which has the second highest priority, and so on. The non-prioritized MCS-channels do not have any priority. A value of T-1 or higher for the bit field **SCD_CH** means that all T MCS-channels are prioritized MCS-channels.

With each clock cycle, the Multiple Prioritization Scheduling mode will assign the non-suspended MCS-channel with the highest priority from the set of prioritized MCS-channels to pipeline stage 0, as long as there are non-suspended prioritized MCS-channels available. If all prioritized MCS-channels are suspended, the scheduling algorithm switches to the accelerated scheduling as previously described in section 15.3.2 and it schedules the non-prioritized channels. Whenever a prioritized MCS-channel is resuming from its suspended state, the scheduler switches back and applies the described prioritization scheme until the next suspension event of occurs.

In consequence, the Multiple Prioritization Scheduling mode cannot guarantee a maximum time duration of an instruction cycle for the overall execution of all MCS-channels, since it strongly depends on the activity of the prioritized MCS-channels. However, the Multiple Prioritization Scheduling mode provides the fastest possible execution for prioritized MCS-channels. Moreover, during the time spawn, in which all prioritized MCS-channels are suspended, this mode guarantees a duration of 1 to T+1 clock cycles of an instruction cycle for all non-prioritized channels.

15.4 Memory Organization

The MCS module supports a memory layout of up to $2^{RAW+USR}$ memory locations each RDW bit wide leading to a maximum byte wise address range from 0 to $2^{RAW+USR+2}$-1.

If two RAM ports are used (USR = 1) the entire address space of the MCS is divided into two seamless memory pages.

Further, if the GTM provides a memory configuration sub module (MCFG), memory page 0 begins from (byte wise) address 0 and ranges to address MP0-4 and memory page 1 ranges from MP0 to MP1-4, while MP0 and MP1 are configuration parameters provided by MCFG. If USR is 1 but there is no MCFG module available, the actual parameters MP0 and MP1 can be found in Appendix B. The base address for accessing the memory via AEI can also be found in Appendix B.

The RAM priority decoder of the MCS will always handle a RAM access from an MCS channel with a higher priority compared to a RAM access from AEI.

However, if a set of active MCS channels are only accessing one common RAM port, the MCS will grant any AEI accesses to the other RAM port in parallel to the related RAM accesses of the running MCS channels, which means that AEI may get the full bandwidth to a dedicated RAM.

Basically, the actual access time to the RAMs via AEI depends on the actual scheduling mode and the activity of tasks. In the modes Round Robin Scheduling and Accelerated Scheduling the scheduler guarantees a maximum write access time of T + 4 clock cycles and a maximum read access time of T + 6 clock cycles. In the scheduling modes Single Prioritization Scheduling and Multiple Prioritization Scheduling, the scheduler cannot guarantee a maximum access time for AEI RAM access.

Depending on the silicon vendor configuration, the connected RAM pages are initialized with zeros in the case of an MCS module reset.

If an ECC Error occurs while an MCS-channel reads data from a memory module, the corresponding MCS-channel is disabled and the ERR bit in register STA is raised.

If the GTM sub module CCM provides several so called address range protectors (ARPs), some code and data sections of the MCS RAM can be write protected. If an MCS channels x writes to such a protected memory region, the MCS channel x is halted, the **ERR** bit in register **STA** is set and the bit field **ERR_SRC_ID** of register **MCS[i]_CTRL_STAT** is updated.

15.5 AEI Bus Master Interface

The MCS module provides an AEI bus master interface, which enables to communicate with externally connected modules. The data width of this interface is BDW bit and the address width is BAW bit leading to a maximum byte wise address ranging from 0 to 2^{BAW+2}-1. The bus master interface is shared among all available MCS channels meaning that each MCS channel may initiate a read or write access on the bus but only one channel can be served at a specific point in time.

However, the AEI bus master interface guarantees, that a bus access is always completed within two instruction cycles and bus access of different MCS channels do not modify the latency of each other. The only exceptions are bus accesses to RAM modules (e.g. accessing memory location in a DPLL RAM or FIFO RAM). AEI bus master accesses to RAM modules cannot be completed within a single clock cycle and thus additional wait cycles have to be inserted into the bus protocol leading to the fact that the MCS channel that is accessing the RAM is entering a suspended state. Moreover, if an MCS channel is accessing a RAM module, the latency of a bus access in another MCS channel can also be modified even if the neighboring channel is accessing only a configuration register.

The AEI bus master interface of an MCS module is connected to AEI slave interface GTM-IP in order to control the sub modules of the GTM within MCS. However, it is not possible to access the entire GTM by a single MCS module. The n-th MCS instance can only access the GTM sub modules that are located within the n-th cluster of the GTM. Details about the available clusters of the GTM can be found in section 2.1.

Additionally, the address map for accessing GTM with the AEI bus master interface of an MCS differs from the address map for an externally connected CPU that is using the GTM's AEI slave interface. The address map for accessing GTM with the AEI bus master interface of an MCS can be found in Appendix B.

Since the sub modules of the GTM can be accessed by the CPU and the AEI bus master of an MCS, the GTM-IP provides an additional arbitration scheme to manage parallel accesses from both master interfaces. If CPU and an MCS want to access a GTM sub module of the same cluster, the arbiter will grant the access to the MCS. However, if the CPU and all the MCS instances want to access GTM sub modules of different clusters, the accesses can be executed in parallel.

The AEI bus master interface can be controlled by the MCS instructions BRD, BRDI, BWR, and BWRI. These instructions are described in section 15.7.

15.6 ADC Interface

The clusters of the GTM can provide a dedicated interface for the connection of up to 32 external Analog-Digital-Converter (ADC) channels, which can be mapped arbitrarily to physical instances of single- or multi-channel ADCs. See Appendix B for details about the availability of ADC interfaces.

An ADC Interface is directly mapped into the address map of the AEI bus master interface of the current cluster's MCS meaning that the available AEI bus master instructions (BRD and BRDI as described in section 15.7) are used to control the connected ADCs.

Since the control of the connected ADCs is silicon vendor specific, the GTM specification does not provide a complete specification for controlling connected ADCs. However, to ensure software compatibility at least for the basic features of an ADC, the functionality described in the following are common to all silicon vendors.

15.6.1 Basic ADC Functions

The address map of the AEI bus master interface reserves two unique address items for each ADC channel. The address items can be referred by the labels **ADC_CH[y]_DATA** and **ADC_CH[y]_STA** for the channel y in the range from 0 to 31. The actual address for these labels can be found in Appendix B.

The MCS can read from address **ADC_CH[y]_DATA** in order to get the conversion result of the ADC that is connected to ADC channel y. The conversion result is represented as a signed 24 bit value and it is stored in the register A (A $\in$ GREG) as referred by the corresponding MCS instruction BRD or BRDI. Additionally, each read access to **ADC_CH[y]_DATA** triggers the ADC that is connected to channel y. Any read access to **ADC_CH[y]_DATA** also provides 8 status bits that are stored in register **MHB**. The bit **MHB**[7] has always the mnemonic **ADC_ACK** and the bit **MHB**[6] has always the mnemonic **ADC_NEW_DATA**. If bit **ADC_ACK** is set the result of the data conversion (register A) and the corresponding status bits (bits **MHB**[6:0]) are validated. If **ADC_NEW_DATA** is set the current conversion result is new and has never been read by a previous bus read access. The meaning of the bits **MHB**[5:0] are vendor specific. Otherwise, if **ADC_ACK** is cleared the read data is invalid and the **MHB**[4:0] indicate the channel identifier (with **MHB**[4:0] ≠ y) that is currently processed by the ADC. A write access to **ADC_CH[y]_DATA** has no functionality and is always ignored.

The MCS can read from address **ADC_CH[y]_STA** to get additional 31-bit wide vendor specific status information of ADC channel y. The lower 24 bits of the status information is stored in register A (A ∊ GREG) as referred by the corresponding MCS instruction BRD or BRDI. The upper 7 bit of the status information is stored in register **MHB**[6:0]. The bit **MHB**[7] has always the mnemonic **ADC_ACK**. If bit **ADC_ACK** is set the result of status information in register A and bits **MHB**[6:0] are validated. Otherwise, if **ADC_ACK** is cleared the status information is invalidated. A write access to **ADC_CH[y]_STA** has no functionality and is always ignored.

Any read or write access to a register **ADC_CH[y]_STA** or **ADC_CH[y]_DATA** updates the AEI status signal that is evaluated in the sub module CCM. The following status information is defined for the AEI status values:

00 : no error occurred
01 : optional information register not implemented (only register **ADC_CH[y]_STA**)
10 : illegal ADC access (e.g. ADC not enabled)
11 : unsupported address (ADC channel y not available)

Note: If the received status AEI is unequal to "00" **ADC_NEW_DATA** is always set and **ADC_ACK** is always cleared.

15.7 Instruction Set

This section describes the entire instruction set of the MCS sub module. First, a brief overview over all available instructions is given and a detailed description of each instruction can be found in sections 15.7.6 and the following sections.

In general, each instruction is RDW bit wide but the duration of each instruction varies between several instruction cycles. As already described in section 15.3, the number of required clock cycles for an instruction cycle can be fixed or variable, depending on the selected scheduling mode. In the case of the Round Robin Scheduling, the duration is fixed with T+1 clock cycles, in the case of the Accelerated Scheduling the duration is variable in the range between 1 and T+1 clock cycles, and in all other Scheduling modes the duration is also variable and may even be more than T+1 clock cycles, depending on the application.

Before the available instructions are described, some commonly used terms, abbreviations and expressions are introduced:

OREG: The operation register set OREG = {R0, R1, ..., R7} ∪ { STA, ACB, CTRG, STRG, TBU_TS0, TBU_TS1, TBU_TS2, MHB} include all MCS accessible internal channel specific GPRs {R0, R1, ..., R7} and the sub set {STA, ACB, CTRG, STRG, TBU_TS0, TBU_TS1, TDU_TS2, MHB} of SFRs.

XOREG: The extended operation register set XOREG = OREG $\cup$ {RS0, RS1, ..., RS7} $\cup$ {GMI0, GMI1, DSTA, DSTAX} extends the operation registers set OREG by the GPRs of the succeeding MCS channel {RS0, RS1, ..., RS7} and the SFRs {GMI0, GMI1, DSTA, DSTAX}.

WXREG: The extended wait instruction operation register set WXREG = OREG $\cup$ {GMI0, GMI1, DSTA, DSTAX} extends the operation registers set OREG by the SFRs {GMI0, GMI1, DSTA, DSTAX}.

AREG: The ARU register set AREG = {R0, R1, R2, ..., R7, ZERO} includes the all registers that can be written by incoming ARU transfers (ARD, ARDI, NARD, and NARDI instructions). These registers include all eight general purpose registers. The dummy register ZERO may be used to discard an incoming 24 bit ARU word.

GREG: The general purpose register set GREG = {R0, R1, R2, ..., R7} includes the all channel specific GPRs without GPRs of neighboring channels.

BAREG: The base address register set BAREG = OREG $\cup$ {RS0, RS1, ..., RS7} extends the register set OREG by the GPRs of neighboring channels.

Note: If the extended operation register set XOREG is disabled (bit **EN_XOREG** of register **MCS[i]_CTRL_STAT** is cleared) the sets **XOREG**, **WXREG**, and **BAREG** only contains the operation register set OREG.

Note: In the following, the register sets **OREG**, **XOREG**, **GREG, WXREG**, **BAREG** and **AREG** are referred by the instructions. Typically, an operation announces W data bits. Whenever, a register of a register set implements less than W bits, it is assumed that these register bits only define the LSBs of an operation. The missing MSBs are always read and written as zeros.

WLIT: The set WLIT = {0,1, ..., 2^{W}-1} is a W bit wide literal value used for encoding immediate operands.
ALIT: The set ALIT = {0,1, ..., $2^{RAW+USR}$-1} is a RAW + USR bit wide literal value used for encoding memory addresses.

AOLIT: The set AOLIT = {-$2^{RAW+USR-1}$, ..., -1,0,1, ..., $2^{RAW+USR-1}$-1} is a RAW + USR bit wide literal value used for encoding relative memory address offsets.

ARDLIT: The set ARDLIT = {0,1, ..., 2^{9}-1} is a 9 bit literal used for ARU read addresses.
AWRLIT: The set AWRLIT = {0,1, ..., 23} is used as ARU write indexes, selecting one of the 24 ARU write address.

BALIT: The set BALIT = {0,1, ..., 2^{BAW}-1} is a BAW bit wide literal used for encoding bus master addresses.
SFTLIT: The set SFTLIT = {0,1, ..., W} is used as literal value for shift instructions.

BWSLIT: The set BWSLIT = {1, ..., W} is used as literal value for multiplication instructions.
BITLIT: The set BITLIT = {0,1, ..., 15} is a 4 bit literal used for bit indexing.
XBITLIT: The set XBITLIT = {0,1, ..., W-1} is a literal used for bit indexing of register bits.
MSKLIT: The set MSKLIT = {0,1, ..., 2^{15}-1} is a 16 bit literal used for bit-masking.
BIT SELECTION: The expression VAR[i] represents the i-th bit of a variable VAR.
BIT RANGE SELECTION: The expression VAR[m:n] represents the bit slice of variable VAR that is ranging from bit n to bit m.

MEMORY ADDRESSING: The expression MEM(X) represents the RDW bit wide value at location x (x $\in$ ALIT) of the memory. The expression MEM(x)[m:n] represents the bit slice ranging from bit n to m of the RDW bit wide word at memory location x.

ARU ADDRESSING: In the case of ARU reading, the expression ARU(x) represents the 2*W+5 bit wide ARU word of ARU channel at read address x (x $\in$ ARDLIT). In the case of ARU writing, the expression ARU(x) represents a 2*W+5 bit wide ARU word that is written to an ARU channel indexed by the index x (x $\in$ AWRLIT). The index x selects a single ARU write channel from the pool of the MCS sub module's allocated ARU write channels. An MCS sub module has 24 dedicated ARU write channels, indexed by values 0 to 23. The expression ARU(x)[m:n] represents the bit slice ranging from bit n to m of the 2*W+5 bit wide ARU word.

BUS MASTER ADDRESSING: In the case of reading/writing from the bus master interface, the expression BUS(x) represents the BDW bit wide data word that is read/written at address x (x $\in$ BALIT). The expression BUS(x)[m:n] represents the bit slice ranging from bit n to m of the BDW bit wide data word at the bus.

Table 15.7.1 summarize the entire instruction set of the MCS and Table 15.7.4 shows the encoding of the individual instructions.

15.7.1 Instruction Set Summary (part 1)

Class	Mnemonic	Operation	instruction cycles	Synopsis
Data transfer	MOVL A, C	A ← C	1	Move Literal, A in OREG, C in WLIT
	MOV A, B	A ← B	1	Move, A in XOREG, B in XOREG

	MRD A, C	A ← MEM(C)[W-1:0];
		MHB ← MEM(C)[RDW-1:W]
			2[1)]	Memory Read, A in OREG, C in ALIT

	MWR A, C	MEM(C)[W-1:0] ← A; MEM(C)[RDW-1:W] ← MHB	2[1)]	Memory Write, A in OREG, C in ALIT

	MRDI A, B [, C]	A ← MEM(B[RAW+USR+1:2]+C)[W-1:0];
		MHB ← MEM(B[RAW+USR+1:2]+C)[RDW-1:W]
			2[1)]	Memory Read Indirect, A in OREG, B in OREG, C in AOLIT (default C=0)

	MWRI A, B [, C]	MEM(B[RAW+USR+1:2]+C)[W-1:0] ← A; MEM(B[RAW+USR+1:2]+C)[RDW-1:W] ← MHB	2[1)]	Memory Write Indirect, A in OREG, B in OREG, C in AOLIT (default C=0)

	MRDIO A, B	A ← MEM(B[RAW+USR+1:2] + R5[RAW+USR+1:2])[W-1:0];
		MHB ← MEM(B[RAW+USR+1:2] + R5[RAW+USR+1:2])[RDW-1:W]
			2[1)]	Memory Read Indirect with Offset, A in XOREG, B in BAREG

	MWRIO A, B	MEM(B[RAW+USR+1:2] + R5[RAW+USR+1:2])[W-1:0] ← A; MEM(B[RAW+USR+1:2] + R5[RAW+USR+1:2])[RDW-1:W] ← MHB	2[1)]	Memory Write Indirect with Offset, A in XOREG, B in BAREG

	POP A	A ← MEM(R7[RAW+USR+1:2]]);
		MHB ← MEM(R7[RAW+USR+1:2])[RDW-1:W]
		R7 ← R7 - 4
			2[1)]	Pop from stack, A in OREG

	PUSH A	R7 ← R7 + 4 MEM(R7[RAW+USR+1:2])[W-1:0] ← A MEM(R7[RAW+USR+1:2])[RDW-1:W] ← MHB	2[1)]	Push to stack, A in OREG
	MWRL A, C	MEM(C)[W-1:0] ← A	3[2)]	Memory Write Literal, A in OREG, C in ALIT
	MWRIL A, B	MEM(B[RAW+USR+1:2])[W-1:0] ← A	3[2)]	Memory Write Indirect Literal, A in OREG, B in OREG

ARU Transfer	ARD A, B,C	A ← ARU(C)[W-1:0]
		B ← ARU(C)[2*W-1:W]
		ACB ← ARU(C)[5+2*W:2*W]
			>= 1	Blocking ARU Read, A in AREG, B in AREG, C in ARDLIT

	AWR A, B, C	ARU(C)[W-1:0] ← A ARU(C)[2*W:W] ← B ARU(C)[5+2*W:2*W] ← ACB	>= 1	Blocking ARU Write, A in OREG, B in OREG, C in AWRLIT

	ARDI A, B	A ← ARU(R6[8:0])[W-1:0]
		B ← ARU(R6[8:0])[2*W-1:W]
		ACB ← ARU(R6[8:0])[5+2*W:2*W]
			>= 1	Blocking ARU Read Indirect, A in AREG, B in AREG

	AWRI A, B	ARU(R6[4:0])[W-1:0] ← A ARU(R6[4:0])[2*W-1:W] ← B ARU(R6[4:0])[5+2*W:2*W] ← ACB	>= 1	Blocking ARU Write Indirect, A in OREG, B in OREG

	NARD A, B, C	A ← ARU(C[8:0])[W-1:0]
		B ← ARU(C[8:0])[2*W-1:W]
		ACB ← ARU(C[8:0])[5+2*W:2*W]
			>= 1[7)]	Non-Blocking ARU Read, A in AREG, B in AREG

	NARDI A, B	A ← ARU(R6[8:0])[W-1:0]
		B ← ARU(R6[8:0])[2*W-1:W]
		ACB ← ARU(R6[8:0])[5+2*W:2*W]
			>= 1[7)]	Non-Blocking ARU Read Indirect, A in AREG, B in AREG

Bus Master	BRD A, C	A ← BUS(C)[W-1:0]
		MHB ← BUS(C)[BDW-1:W]
			>= 1[8)]	Bus Master Read, A in GREG, C in BALIT

	BWR A, C	BUS(C)[W-1:0] ← A BUS(C)[BDW-1:W] ← MHB	>= 1[8)]	Bus Master Write, A in GREG, B C in BALIT

	BRDI A, B	A ← BUS(B[BAW+1:2])[W-1:0]
		MHB ← BUS(B[BAW+1:2])[BDW-1:W]
			>= 1[8)]	Bus Master Read Indirect, A in GREG, B in GREG

	BWRI A, B	BUS(B[BAW+1:2])[W-1:0] ← A BUS(B[BAW+1:2])[BDW-1:W] ← MHB	>= 1[8)]	Bus Master Write Indirect, A in GREG, B in GREG

15.7.2 Instruction Set Summary (part 2)

Class	Mnemonic	Operation	instruction cycles	Synopsis
Arith. / Logic	ADDL A, C	A ← A + C	1	Add Literal, A in OREG, C in WLIT
	ADD A, B	A ← A + B	1	Add, A in XOREG, B in XOREG
	ADDC A, B	A ← A + B + CY	1	Add with carry, A in XOREG, B in XOREG
	SUBL A, C	A ← A - C	1	Subtract Literal, A in OREG, C in WLIT
	SUB A, B	A ← A - B	1	Subtract, A in XOREG, B in XOREG
	SUBC A, B	A ← A - B - CY	1	Subtract with carry, A in XOREG, B in XOREG
	NEG A, B	A ← – B	1	Negate, A in XOREG, B in XOREG
	ANDL A, C	A ← A AND C	1	AND Litral, A in OREG, C in WLIT
	AND A, B	A ← A AND B	1	AND, A in XOREG, B in XOREG
	ORL A, C	A ← A OR C	1	OR Literal, A in OREG, C in WLIT
	OR A, B	A ← A OR B	1	OR, A in XOREG, B in XOREG
	XORL A, C	A ← A XOR C	1	XOR Literal, A in OREG, C in WLIT
	XOR A, B	A ← A XOR B	1	XOR, A in XOREG, B in XOREG
	SETB A, B	A[B[4:0]] ← 1	1	Set Bit, A in XOREG, B in XOREG
	CLRB A, B	A[B[4:0]] ← 0	1	Clear Bit, A in XOREG, B in XOREG
	XCHB A, B	A[B[4:0]] ↔ CY	1	Exchange Bit with CY, A in XOREG, B in XOREG
	SHR A, C	A ← A >> C	1	Shift Right, A in XOREG, C in SFTLIT
	SHL A, C	A ← A << C	1	Shift Left, A in XOREG, C in SFTLIT
	ASRU A, B	A ← A >> B	1	Shift Right, A in XOREG, B in XOREG
	ASRS A, B	A ← A >> B	1	Shift Right, A in XOREG, B in XOREG
	ASL A, B	A ← A << B	1	Shift Left, A in XOREG, B in XOREG
	MULU A, B[, C]	[[R4,] A] ← A[(C-1):0] * B[(C-1):0]	1	Multiply Unsigned, A in XOREG, B in XOREG, C in BWSLIT (default C=W)
	MULS A, B[, C]	[[R4,] A] ← A[(C-1):0] * B[(C-1):0]	1	Multiply Signed, A in XOREG, B in XOREG , C in BWSLIT (default C=W)
	DIVU A, B[, C]	R4 ? A[(C-1):0] - B[(C-1):0] * ⌊A[(C-1):0] / B[(C-1):0]⌋; A ? ⌊A[(C-1):0] / B[(C-1):0]⌋	C[9]	Divide Unsigned, A in XOREG, B in XOREG, C in BWSLIT (default C=W)
	DIVS A, B[, C]	R4 ? A[(C-1):0] - B[(C-1):0] * ⌊A[(C-1):0] / B[(C-1):0]⌋; A ? ⌊A[(C-1):0] / B[(C-1):0]⌋	C+4[10]	Divide Signed, A in XOREG, B in XOREG, C in BWSLIT (default C=W)
	MINU A, B	A ← MIN(A, B)	1	Minimum Unsigned, A in XOREG, B in XOREG
	MINS A, B	A ← MIN(A, B)	1	Minimum Signed, A in XOREG, B in XOREG
	MAXU A, B	A ← MAX(A, B)	1	Maximum Unsigned, A in XOREG, B in XOREG
	MAXS A, B	A ← MAX(A, B)	1	Maximum Signed, A in XOREG, B in XOREG
Test	ATUL A, C	A < C ⇔ CY is set A = C ⇔ Z is set	1	Arithmetic Test Unsigned Literal, A in OREG, C in WLIT
	ATU A, B	A < B ⇔ CY is set A = C ⇔ Z is set	1	Arithmetic Test Unsigned, A in XOREG, B in XOREG
	ATSL A, C	A < C ⇔ CY is set A = C ⇔ Z is set	1	Arithmetic Test Signed Literal, A in OREG, C in WLIT
	ATS A, B	A < B ⇔ CY is set A = C ⇔ Z is set	1	Arithmetic Test Signed, A in XOREG, B in XOREG
	BTL A, C	A AND C	1	Bit Test Literal, A in OREG, C in WLIT
	BT A, B	A AND B	1	Bit Test, A in XOREG, B in XOREG

15.7.3 Instruction Set Summary (part 3)

Class	Mnemonic	Operation	instruction cycles	Synopsis
Control Flow	JMP C	PC ← C << 2	1[3]	Unconditional Jump, C in ALIT
	JBS A, B, C	PC ← C << 2 if A[B] is set	1[4]	Jump if Bit Set, A in OREG, B in BITLIT, C in ALIT
	JBC A, B, C	PC ← C << 2 if A[B] is clear	1[4]	Jump if Bit Cleared, A in OREG, B in BITLIT, C in ALIT

	CALL C	R7 ← R7 + 4
		MEM(R7[RAW+USR+1:2])[RAW+USR+1:0] ← PC+ 4
		PC ← C << 2
			2[5]	Call Subroutine, C in ALIT

	RET	PC ← MEM(R7[RAW+USR+1:2])[RAW+USR+1:0]
		R7 ← R7 - 4
			2[5]	Return from Subroutine

	JMPI	PC ← R6[RAW+USR+1:2] << 2	1[3]	Unconditional Jump Indirect
	JBSI A, B	PC ← R6[RAW+USR+1:2] << 2 if A[B] is set	1[4]	Jump if Bit Set Indirect, A in XOREG, B in XBITLIT
	JBCI A, B	PC ← R6[RAW+USR+1:2] << 2 if A[B] is clear	1[4]	Jump if Bit Clear Indirect, A in OREG, B in XBITLIT

	CALLI	R7 ← R7 + 4
		MEM(R7[RAW+USR+1:2])[RAW+USR+1:0] ← PC+ 4
		PC ← R6[RAW+USR+1:2] << 2
			2[5]	Call Subroutine Indirect

Others	WURM A, B, C	wait until A = (B AND ((0xFF << 16) + C))	>= 1[6]	Wait Until Register Match, A in OREG, B in OREG, C in MSKLIT
	WURMX A, B	wait until A = (B AND R6)	>= 1[6]	Wait Until Register Match, A in OREG, B in WXREG
	WURCX A, B	wait until A ? (B AND R6)	>= 1[6]	Wait Until Register Change, A in OREG, B in WXREG
	WUCE A, B	wait until cyclic event comparison matches	>= 1[6]	Wait Until Cyclic Event, A in OREG, B in OREG
	NOP		1	No Operation

Footnotes:

1) Not faster than 1+NPS clock cycles due to pipeline flushing.
2) Not faster than 1+2*NPS clock cycles due to pipeline flushing.
3) Not faster than NPS clock cycles due to pipeline flushing.
4) If the jump is executed, it is not faster than NPS clock cycles due to pipeline flushing.
5) Not faster than 2*NPS clock cycles due to pipeline flushing.
6) If the MCS is configured in Single Prioritization or Multiple Prioritization Scheduling Mode
   the worst case latency for reactivating a prioritized MCS-channel is 2+NPS clock cycles.
7) Always faster than one ARU round trip cycle.
8) Suspends current MCS-channel if addressed slave inserts at least one wait cycle otherwise 1 instruction cycle
9) Not faster than C+NPS-1 clock cycles due to pipeline flushing.
10) Not faster than C+3+NPS clock cycles due to pipeline flushing.

15.7.4 Instruction Codes (part 1)

Menemonic	Instruction Code
MOVL	0001aaaacccccccccccccccccccccccc
MOV	1010aaaabbbb0000-a-b------------
MRD	1010aaaa----0001-ccccccccccccc--
MWR	1010aaaa----0010-ccccccccccccc--
MRDI	1010aaaabbbb0011-ccccccccccccc--
MWRI	1010aaaabbbb0100-ccccccccccccc--
POP	1010aaaa----0101----------------
PUSH	1010aaaa----0110----------------
MWRL	1010aaaa----0111-ccccccccccccc--
MWRIL	1010aaaabbbb1000----------------
BRD	1010-aaa----1001cccccccccccccc--
BWR	1010-aaa----1010cccccccccccccc--
BRDI	1010-aaa-bbb1011----------------
BWRI	1010-aaa-bbb1100----------------
MRDIO	1010aaaabbbb1101-a-b------------
MWRIO	1010aaaabbbb1110-a-b------------
XCHB	1010aaaabbbb1111-a-b------------
ARD	1011aaaabbbb0000-------ccccccccc
AWR	1011aaaabbbb0001-----------ccccc
NARD	1011aaaabbbb0010-------ccccccccc
NARDI	1011aaaabbbb0011----------------
ARDI	1011aaaabbbb0100----------------
AWRI	1011aaaabbbb0101----------------
SETB	1011aaaabbbb0110-a-b------------
CLRB	1011aaaabbbb0111-a-b------------
ADDL	0010aaaacccccccccccccccccccccccc
ADD	1100aaaabbbb0000-a-b------------
SUBL	0011aaaacccccccccccccccccccccccc
SUB	1100aaaabbbb0001-a-b------------
NEG	1100aaaabbbb0010-a-b------------
ANDL	0100aaaacccccccccccccccccccccccc
AND	1100aaaabbbb0011-a-b------------
ORL	0101aaaacccccccccccccccccccccccc
OR	1100aaaabbbb0100-a-b------------
XORL	0110aaaacccccccccccccccccccccccc
XOR	1100aaaabbbb0101-a-b------------
SHR	1100aaaa----0110-a---------ccccc
SHL	1100aaaa----0111-a---------ccccc

15.7.5 Instruction Codes (part 2)

Menemonic	Instruction Code
MULU	1100aaaabbbb1000-a-b-------ccccc
MULS	1100aaaabbbb1001-a-b-------ccccc
DIVU	1100aaaabbbb1010-a-b-------ccccc
DIVS	1100aaaabbbb1011-a-b-------ccccc
MINU	1100aaaabbbb1100-a-b------------
MINS	1100aaaabbbb1101-a-b------------
MAXU	1100aaaabbbb1110-a-b------------
MAXS	1100aaaabbbb1111-a-b------------
ASL	1101aaaabbbb0011-a-b------------
ASRU	1101aaaabbbb0100-a-b------------
ASRS	1101aaaabbbb0101-a-b------------
ADDC	1101aaaabbbb0110-a-b------------
SUBC	1101aaaabbbb0111-a-b------------
ATUL	0111aaaacccccccccccccccccccccccc
ATU	1101aaaabbbb0000-a-b------------
ATSL	1000aaaacccccccccccccccccccccccc
ATS	1101aaaabbbb0001-a-b------------
BTL	1001aaaacccccccccccccccccccccccc
BT	1101aaaabbbb0010-a-b------------
JMP	1110--------0000-ccccccccccccc--
JBS	1110aaaabbbb0001-ccccccccccccc--
JBC	1110aaaabbbb0010-ccccccccccccc--
CALL	1110--------0011-ccccccccccccc--
RET	1110--------0100----------------
JMPI	1110--------0101----------------
JBSI	1110aaaabbbb0110-a-b------------
JBCI	1110aaaabbbb0111-a-b------------
CALLI	1110--------1000----------------
WURM	1111aaaabbbb0000cccccccccccccccc
WURMX	1111aaaabbbb0001---b------------
WURCX	1111aaaabbbb0010---b------------
WUCE	1111aaaabbbb0011----------------
NOP	0000----------------------------

The individual instructions are decoded by evaluating the bits '0' and '1' at its expected positions, as mentioned in the table above. If the instruction decoder detects an invalid combination of these bits, the corresponding MCS-channel is disabled and the ERR bit in the register STA is set. Bit positions marked as '-' are not relevant for the instruction. The bit position 'a', 'b', and 'c' are reserved for binary encoding of the instruction arguments A, B, and C.

Moreover, each instruction can set the **ERR** bit of register **STA** and stop the program execution, if a register write protection of an associated MCS channel is activated by the register **MCS[i]_REG_PROT**. This behavior is not explicitely mentioned in the instruction descriptions below. If an error occurs due to a write access to a protected register it is ensured that the protected register is not overwritten. However, it is not ensured that other operations (e.g. updating of the PC) of the bad instruction are exeuted.

15.7.6 MOVL Instruction

Syntax: MOVL A, C
Operation: A ← C
Status: Z
Duration: 1 instruction cycle
Description: Transfer literal value C (C ∊ WLIT) to register A (A ∊ OREG).
The zero bit Z of status register STA is set, if the transferred value is zero, otherwise the zero bit is cleared.

The program counter PC is incremented by the value 4.

15.7.7 MOV Instruction

Syntax: MOV A, B
Operation: A ← B
Status: Z
Duration: 1 instruction cycle

Description: Transfer register B (B ∊ XOREG) to register A (A ∊ XOREG).
The zero bit Z of status register STA is set, if the transferred value is zero, otherwise the zero bit is cleared.

The program counter PC is incremented by the value 4.

15.7.8 MRD Instruction

Syntax: MRD A, C
Operation: A ← MEM(C)[W-1:0];

MHB ← MEM(C)[RDW-1:W]

Status: Z
Duration: 2 instruction cycles but not faster than 1+NPS clock cycles due to pipeline flushing.
Description: Transfer the lower W bits of memory content at location C (C ∊ ALIT) to register A (A ∊ OREG).
The upper RDW-W bits of the memory content at location C are transferred to the MHB register.
The zero bit Z of status register STA is set, if the lower W bits of the transferred value are zero, otherwise the zero bit is cleared.

If the MHB register is selected as destination register A (A ∊ OREG), the bits 0 to RDW-W-1 of the referred memory location are transferred to MHB.

The program counter PC is incremented by the value 4.

15.7.9 MWR Instruction

Syntax: MWR A, C
Operation: MEM(C)[W-1:0] ← A;

MEM(C)[RDW-1:W] ← MHB
Status: -
Duration: 2 instruction cycles but not faster than 1+NPS clock cycles due to pipeline flushing.
Description: Transfer W bit value of register A (A ∊ OREG) together with the MHB register to the memory at location C (C ∊ ALIT).

The W bit value of register A is stored in the LSBs (bit 0 to W-1) of the memory location.
The MHB register is stored in bits W to RDW-W-1 of the referred memory location.
The program counter PC is incremented by the value 4.

15.7.10 MWRL Instruction

Syntax: MWRL A, C
Operation: MEM(C)[W-1:0] ← A
Status: -
Duration: 3 instruction cycles but not faster than 1+2*NPS clock cycles due to pipeline flushing.
Description: Transfer W bit value of register A (A ∊ OREG) to memory at location C (C ∊ ALIT).
The W bit value of register A is stored in the LSBs (bit 0 to W-1) of the memory location and the bits W to RDW-W are left unchanged.

The program counter PC is incremented by the value 4.
It should be noted that this operation is not an atomic instruction.

15.7.11 MRDI Instruction

Syntax: MRDI A, B [, C]
Operation: A ← MEM(B[RAW+USR+1:2] + C)[W-1:0]

MHB ← MEM(B[RAW+USR+1:2] + C)[RDW-1:W]

Status: Z

Duration: 2 instruction cycles but not faster than 1+NPS clock cycles due to pipeline flushing.

Description: Transfer the bits 0 to W-1 of a memory location to register A (A ∈ OREG) using indirect addressing.

The upper RDW-W bits of this memory location are transferred to MHB register.

The memory location where to read from depends on register B (B ∈ OREG) and literal C (C ∈ AOLIT) and it is defined as B[RAW+USR+1:2] + C.

If the optional operand C is not available in the assembler syntax, the MCS assembler generates code with a default value of 0 for operand C.

The zero bit Z of status register STA is set, if the transferred bits 0 to W-1 are zero, otherwise the zero bit is cleared.

If the MHB register is selected as destination register A (A ∈ OREG), the bits 0 to RDW-W-1 of the referred memory location are transferred to MHB.

The program counter PC is incremented by the value 4.

15.7.12 MRDIO Instruction

Syntax: MRDIO A, B

Operation: A ← MEM(B[RAW+USR+1:2] + R5[RAW+USR+1:2])[W-1:0]

MHB ← MEM(B[RAW+USR+1:2] + R5[RAW+USR+1:2])[RDW-1:W]

Status: Z

Duration: 2 instruction cycles but not faster than 1+NPS clock cycles due to pipeline flushing.

Description: Transfer the bits 0 to W-1 of a memory location to register A (A ∈ XOREG) using indirect addressing with offset calculation.

The upper RDW-W bits of this memory location are transferred to MHB register.

The memory location where to read from depends on register B (B ∈ BAREG) and register R5 and it is defined as B[RAW+USR+1:2] + R5[RAW+USR+1:2].

The zero bit Z of status register STA is set, if the transferred bits 0 to W-1 are zero, otherwise the zero bit is cleared.

If the MHB register is selected as destination register A, the bits 0 to RDW-W-1 of the referred memory location are transferred to MHB.

The program counter PC is incremented by the value 4.

15.7.13 MWRI Instruction

Syntax: MWRI A, B [, C]
Operation: MEM(B[RAW+USR+1:2] + C)[W-1:0] ← A;
MEM(B[RAW+USR+1:2] + C)[RDW-1:W] ← MHB
Status: -
Duration: 2 instruction cycles but not faster than 1+NPS clock cycles due to pipeline flushing.

Description: Transfer value of register A (A ∊ OREG) to the LSBs 0 to W-1 of a memory location using indirect addressing.

The MHB register is moved to the bits W to RDW-1 at the same memory location.
If the optional operand C is not available in the assembler syntax, the MCS assembler generates code with a default value of 0 for operand C.

The memory location where to write to depends on register B (B ∊ OREG) and literal C (C ∊ AOLIT) and it is defined as B[RAW+USR+1:2] + C.

The program counter PC is incremented by the value 4.

15.7.14 MWRIO Instruction

Syntax: MWRIO A, B
Operation: MEM(B[RAW+USR+1:2] + R5[RAW+USR+1:2])[W-1:0] ← A;
MEM(B[RAW+USR+1:2] + R5[RAW+USR+1:2])[RDW-1:W] ← MHB
Status: -
Duration: 2 instruction cycles but not faster than 1+NPS clock cycles due to pipeline flushing.

Description: Transfer value of register A (A ∊ XOREG) to the LSBs 0 to W-1 of a memory location using indirect addressing with offset calculation.

The MHB register is moved to the bits W to RDW-1 at the same memory location.
The memory location where to write to depends on register B (B ∊ BAREG) and register R5 and it is defined as B[RAW+USR+1:2] + R5[RAW+USR+1:2].

The program counter PC is incremented by the value 4.

15.7.15 MWRIL Instruction

Syntax: MWRIL A, B
Operation: MEM(B[RAW+USR+1:0])[W-1:0] ← A;
Status: -
Duration: 3 instruction cycles but not faster than 1+2*NPS clock cycles due to pipeline flushing.

Description: Transfer W bit value of A (A ∈ OREG) to memory using indirect addressing.
The memory location where to write to is defined by the bits 2 to RAW+1 of register B (B ∈ OREG).
The W bit value is stored in the LSBs (bit 0 to W-1) of the memory location and the bits W to RDW-1 are left unchanged.

The program counter PC is incremented by the value 4.
It should be noted that this operation is not an atomic instruction.

15.7.16 POP Instruction

Syntax: POP A
Operation: A ← MEM(R7[RAW+USR+1:2])[W-1:0];
MHB ← MEM(R7[RAW+USR+1:2])[RDW-1:W];
R7 ← R7 - 4;
SP_CNT ← SP_CNT - 1

Status: Z, EN
Duration: 2 instruction cycles but not faster than 1+NPS clock cycles due to pipeline flushing.

Description: Transfer the LSBs (bit 0 to W-1) from the top of stack to register A (A ∈ OREG), followed by decrementing the stack pointer register R7 with the value 4.

The upper bits W to RDW-1 from the top of the stack are transferred to register MHB.
If the MHB register is selected as destination register A (A ∈ OREG), the bits 0 to RDW-W-1 from the top of the stack are transferred to MHB.

The memory location for the top of the stack is identified by the bits 2 to RAW+1 of the stack pointer register R7.

The zero bit Z of status register STA is set, if the lower W bit of the transferred value is zero, otherwise the zero bit is cleared.

The program counter PC is incremented by the value 4.
The **SP_CNT** bit field inside the **MCS[i]_CH[x]_CTRL** register is decremented.
If an underflow on the **SP_CNT** bit field occurs, the *STK_ERR[i]_IRQ* is raised.
If an underflow on the **SP_CNT** bit field occurs and the bit **HLT_SP_OFL** of register **MCS[i]_CTRL** is set, the current MCS-channel is disabled by clearing the **EN** bit of **STA**.

15.7.17 PUSH Instruction

Syntax: PUSH A
Operation: R7 ← R7 + 4;
MEM(R7[RAW+USR+1:2])[W-1:0] ← A;
MEM(R7[RAW+USR+1:2])[RDW-1:W] ← MHB
SP_CNT ← SP_CNT + 1;
Status: EN
Duration: 2 instruction cycles but not faster than 1+NPS clock cycles due to pipeline flushing.

Description: Increment the stack pointer register R7 with the value 4, followed by transferring a W bit value of operand A (A ∈ OREG) together with a MHB register to the new top of the stack. The W bit value of A is stored in the bits 0 to W-1 of the memory location.

The content of the MHB register is stored in the bit W to RDW-1 of the memory location. The memory location for the top of the stack is referred by the bits 2 to RAW+1 of the stack pointer register.
The program counter PC is incremented by the value 4.
The **SP_CNT** bit field inside the **MCS[i]_CH[x]_CTRL** register is incremented.
If an overflow on the **SP_CNT** bit field occurs, the *STK_ERR[i]_IRQ* is raised.
If an overflow on the **SP_CNT** bit field occurs and the bit **HLT_SP_OFL** of register **MCS[i]_CTRL** is set, the current MCS-channel is disabled by clearing the **EN** bit of **STA**.

If an overflow on the **SP_CNT** bit field occurs and the bit **HLT_SP_OFL** of register **MCS[i]_CTRL** is set, the memory write operation for the A and MHB is discarded.

15.7.18 ARD Instruction

Syntax: ARD A, B, C
Operation: A ← ARU(C)[W-1:0];

B ← ARU(C)[2*W-1:W];
ACB ← ARU(C)[4+2*W:2*W]

Status: CAT, SAT
Duration: Suspends current MCS-channel until ARU transfer finished.
Description: Perform a blocking read access to the ARU and transfer both W bit values received at the ARU port to the registers A and B (A ∈ AREG, B ∈ AREG), whereas A holds the lower W bit ARU word and B holds the upper W bit ARU word.

If A and B refer to the same register, only the upper W bit ARU word is stored and the lower W bit ARU word is discarded.

If any transferred W bit value from the ARU should not be stored in a register, the dummy register ZERO ∈ AREG can be selected in A or B to discard the corresponding ARU data. The binary encoding of the address for the dummy register ZERO can be chosen by an arbitrary value within the range 8 to 15.

The received ARU control bits are stored in the register ACB.
The literal C (C ∈ ARDLIT) define the ARU address where to read from.
At the beginning of the instruction execution the CAT bit in register STA is always cleared.
After the execution of the instruction the SAT flag of the register STA is updated in order to show if the transfer was successful (SAT = 1) or if the transfer failed (SAT = 0) due to a cancellation by the CPU.

The program counter PC is incremented by the value 4.

15.7.19 ARDI Instruction

Syntax: ARDI A, B
Operation: A ← ARU(R6[8:0])[W-1:0];
B ← ARU(R6[8:0])[2*W-1:W];
ACB ← ARU(R6[8:0])[4+2*W:2*W]

Status: CAT, SAT
Duration: Suspends current MCS-channel until ARU transfer finished.
Description: Perform a blocking read access to the ARU and transfer both W bit values received at the ARU port to the registers A and B (A ∈ AREG, B ∈ AREG), whereas A holds the lower W bit ARU word and B holds the upper W bit ARU word.

If A and B refer to the same register, only the upper W bit ARU word is stored and the lower W bit ARU word is discarded.

If any transferred W bit value from the ARU should not be stored in a register, the dummy register ZERO ∈ AREG can be selected in A or B to discard the corresponding

ARU data. The binary encoding of the address for the dummy register ZERO can be chosen by an arbitrary value within the range 8 to 15.

The received ARU control bits are stored in the register ACB.
The read address is obtained from the bits 0 to 8 of the channels register R6.
At the beginning of the instruction execution the CAT bit in register STA is always cleared.
After the execution of the instruction the SAT flag of the register STA is updated in order to show if the transfer was successful (SAT = 1) or if the transfer failed (SAT = 0) due to a cancellation by the CPU.

The program counter PC is incremented by the value 4.

15.7.20 AWR Instruction

Syntax: AWR A, B, C
Operation: ARU(C)[W-1:0] ← A;
ARU(C)[2*W-1:W] ← B;
ARU(C)[4+2*W:2*W] ← ACB;
Status: CAT, SAT
Duration: Suspends current MCS-channel until ARU transfer finished.
Description: Perform a blocking write access to the ARU and transfer two W bit values to the ARU port using the registers A and B (A ∈ OREG, B ∈ OREG), whereas A holds the lower W bit ARU word and B holds the upper W bit ARU word.

The ARU control bits are taken from the register ACB.
The literal C (C ∈ AWRLIT) defines an index into the pool of ARU write addresses that are used for writing data. This index is mapped to an ARU write address as shown in column "MCS write index" of table "ARU Write Addresses" in Appendix B.

Each MCS sub module has a pool of several write addresses that can be shared between all MCS-channels arbitrarily.

At the beginning of the instruction execution the CAT bit of the register STA is always cleared.
After the execution of the instruction the SAT flag of the register STA is updated in order to show if the transfer was successful (SAT = 1) or if the transfer failed (SAT = 0) due to a cancellation by the CPU.

The program counter PC is incremented by the value 4.

15.7.21 AWRI Instruction

Syntax: AWRI A, B
Operation: ARU(R6[4:0])[W-1:0] ← A;
ARU(R6[4:0])[2*W-1:W] ← B;
ARU(R6[4:0])[4+2*W:2*W] ← ACB;

Status: CAT, SAT
Duration: Suspends current MCS-channel until ARU transfer finished.
Description: Perform a blocking write access to the ARU and transfer two W bit values to the ARU port using the registers A and B (A ∊ OREG, B ∊ OREG), whereas A holds the lower W bit ARU word and B holds the upper W bit ARU word.

The ARU control bits are taken from the register ACB.
The bits 0 to 4 of the register R6 define an index into the pool of ARU write addresses that are used for writing data. This index is mapped to an ARU write address as shown in column "MCS write index" of table "ARU Write Addresses" in Appendix B.

Each MCS sub module has a pool of several write addresses that can be shared between all MCS-channels arbitrarily.

At the beginning of the instruction execution the CAT bit of the register STA is always cleared.
After the execution of the instruction the SAT flag of the register STA is updated in order to show if the transfer was successful (SAT = 1) or if the transfer failed (SAT = 0) due to a cancellation by the CPU.

The program counter PC is incremented by the value 4.

15.7.22 NARD Instruction

Syntax: NARD A, B, C
Operation: A ← ARU(C)[W-1:0];
B ← ARU(C)[2*W:W];
ACB ← ARU(C)[4+2*W:2*W]
Status: SAT
Duration: Suspends current MCS-channel until the ARU is selecting the MCS-channel.
Description: Perform a non-blocking read access to the ARU trying to transfer both W bit values received at the ARU port to the registers A and B (A ∊ AREG, B ∊ AREG), whereas A holds the lower W bit ARU word, B holds the upper W bit ARU word, and the ACB register holds the received ARU control bits. The literal C (C ∊ ARDLIT) define the ARU address where to read from.

Non-blocking ARU read acces means that the instruction is suspending the MCS channel until the ARU scheduler is selecting the requesting MCS channel. If the transfer finished successfully, the bit SAT of the register STA is set and the transferred values are stored in the registers A, B, and ACB. If the transfer failed due to missing data at the requested source, the bit SAT of the register STA is cleared and registers A, B, and ACB are not changed.

If A and B refer to the same register, only the upper W bit ARU word is stored and the lower W bit ARU word is discard.

If any transferred W bit value from the ARU should not stored in a register, the dummy register ZERO ∈ AREG can be selected in A or B to discard the corresponding ARU data. The binary encoding of the address for the dummy register ZERO can be chosen by an arbitrary value within the range 8 to 15.

The program counter PC is incremented by the value 4.

15.7.23 NARDI Instruction

Syntax: NARDI A, B
Operation: A ← ARU(R6[8:0])[W-1:0];
B ← ARU(R6[8:0])[2*W-1:W];
ACB ← ARU(R6[8:0])[4+2*W:2*W]

Status: SAT
Duration: Suspends current MCS-channel until the ARU is selecting the MCS-channel.
Description: Perform a non-blocking read access to the ARU trying to transfer both W bit values received at the ARU port to the registers A and B (A ∈ AREG, B ∈ AREG), whereas A holds the lower W bit ARU word, B holds the upper W bit ARU word, and the ACB register holds the received ARU control bits. The read address is obtained from the bits 0 to 8 of the channels register R6.

Non-blocking ARU read acces means that the instruction is suspending the MCS channel until the ARU scheduler is selecting the requesting MCS channel. If the transfer finished successfully, the bit SAT of the register STA is set and the transferred values are stored in the registers A, B, and ACB. If the transfer failed due to missing data at the requested source, the bit SAT of the register STA is cleared and registers A, B, and ACB are not changed.

If A and B refer to the same register, only the upper W bit ARU word is stored and the lower 24 bit ARU word is discard.

If any transferred W bit value from the ARU should not stored in a register, the dummy register ZERO ∈ AREG can be selected in A or B to discard the corresponding ARU data. The binary encoding of the address for the dummy register ZERO can be chosen by an arbitrary value within the range 8 to 15.

The program counter PC is incremented by the value 4.

15.7.24 BRD Instruction

Syntax: BRD A, C
Operation: A ← BUS(C)[W-1:0];
MHB ← BUS(C)[BDW-1:W]
Status: -
Duration: Suspends current MCS-channel if addressed slave inserts at least one wait cycle (e.g. accessing a RAM module) otherwise 1 instruction cycle.

Description: Initiate a read access at the bus master interface using the address C (C ∊ BALIT) and transfer the lower W bits of the received data to register A (A ∊ GREG).

The upper BDW-W bits of the received data are transferred to the MHB register.
If the delay between the a BRD instruction and its successor instruction is one or two system clock cycles (e.g. in accelerated scheduling mode) and the successor instruction is reading data resulting from the BRD instruction, a data hazard in the pipeline occurs resulting in a pipeline flush. This means, if very fast program execution is required (e.g. only one task is activated in accelerated scheduling mode) a program sequence like BRD R1, 0x0288; ADD R3, R1; (9 clock cycles) can be accelerated by reformulating the sequence as BRD R1, 0x0288; NOP; NOP; ADD R3, R1; (4 clock cycles).

Since the MHB register is always transferred via AEI bus master it also figures out another data dependency, which can cause a data hazard resulting in a pipline flush. Therefore a sequence like BRD R1, 0x0288; BWR R3, 0x304 (9 clock cycles) could also be optimized by the sequence BRD R1, 0x0288; NOP; NOP; BWR R3, 0x304(4 clock cycles).

The program counter PC is incremented by the value 4.

15.7.25 BWR Instruction

Syntax: BWR A, C
Operation: BUS(C)[W-1:0] ← A;
BUS(C)[BDW-1:W] ← MHB
Status: -
Duration: Suspends current MCS-channel if addressed slave inserts at least one wait cycle (e.g. accessing a RAM module) otherwise 1 instruction cycle.

Description: Initiate a write access at the bus master interface using the address C (C ∊ BALIT) and transfer the content of register A (A ∊ GREG) to the bits 0 to W-1 of the bus.

The content of the MHB register is transferred to the bits W to BDW-W-1 of the bus. The program counter PC is incremented by the value 4.

15.7.26 BRDI Instruction

Syntax: BRDI A, B
Operation: A ← BUS(B[BAW+1:2])[W-1:0];
MHB ← BUS(B[BAW+1:2])[BDW-1:W]
Status: -
Duration: Suspends current MCS-channel if addressed slave inserts at least one wait cycle (e.g. accessing a RAM module) otherwise 1 instruction cycle.

Description: Initiate a read access at the bus master interface using indirect addressing and transfer the lower W bits of the received data to register A (A ∈ GREG).

The upper BDW-W bits of the received data are transferred to the MHB register.
The address for the transfer is identified by the bits 2 to BAW+1 of register B (B ∈ GREG).
If the delay between the a BRDI instruction and its successor instruction is one or two system clock cycles (e.g. in accelerated scheduling mode) and the successor instruction is reading data resulting from the BRDI instruction, a data hazard in the pipeline occurs resulting in a pipeline flush. This means, if very fast program execution is required (e.g. only one task is activated in accelerated scheduling mode) a program sequence like BRDI R1, R6; ADD R3, R1; (9 clock cycles) can be accelerated by reformulating the sequence as BRDI R1, R6; NOP; NOP; ADD R3, R1; (4 clock cycles).

Since the MHB register is always transferred via AEI bus master it also figures out another data dependency, which can cause a data hazard resulting in a pipline flush. Therefore a sequence like BRDI R1, R2; BWRI R3, R4 (9 clock cycles) could also be optimized by the sequence BRDI R1, R2; NOP; NOP; BWRI R3, R4 (4 clock cycles).

The program counter PC is incremented by the value 4.

15.7.27 BWRI Instruction

Syntax: BWRI A, B
Operation: BUS(B[BAW+1:2])[W-1:0] ← A;
BUS(B[BAW+1:2])[BDW-1:W] ← MHB
Status: -
Duration: Suspends current MCS-channel if addressed slave inserts at least one wait cycle (e.g. accessing a RAM module) otherwise 1 instruction cycle.

Description: Initiate a write access at the bus master interface using the indirect addressing and transfer the content of register A (A ∈ GREG) to the bits 0 to W-1 of the bus.

The content of the MHB register is transferred to the bits W to BDW-W-1 of the bus.
The address for the transfer is identified by the bits 2 to BAW+1 of register B (B ∈ GREG).
The program counter PC is incremented by the value 4.

15.7.28 ADDL Instruction

Syntax: ADDL A, C
Operation: A ← A + C
Status: Z, CY, N, V
Duration: 1 instruction cycle
Description: Perform addition operation of a register A (A ∈ OREG) with a W bit literal value C (C ∈ WLIT) and store the result in register A.

The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the zero bit is cleared.

The carry bit CY of status register STA is set, if an unsigned overflow/underflow occurred during addition, otherwise the bit is cleared. An unsigned overflow has occurred when the result of the operation cannot be represented in the interval [0; 2^{W}-1], assuming that both operands A and C are unsigned values within the interval [0; 2^{W}-1].

The overflow bit V of status register STA is set, if a signed overflow/underflow occurred during addition, otherwise the bit is cleared. A signed overflow/underflow has occurred when the result of the operation cannot be represented in the interval [-2^{W-1}; 2^{W-1}-1], assuming that both operands A and C are signed values within the interval [-2^{W-1}; 2^{W-1}-1].

The negative bit N of status register STA equals the MSB of the operation result, in order to determine if a calculated signed result is negative (N=1) or positive (N=0), assuming that no overflow/underflow occurred.

The program counter PC is incremented by the value 4.

15.7.29 ADD Instruction

Syntax: ADD A, B
Operation: A ← A + B

Status: Z, CY, N, V
Duration: 1 instruction cycle
Description: Perform addition operation of a register A (A ∈ XOREG) with an operand B (B ∈ XOREG). The result is stored in the register A.

The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the zero bit is cleared.

The carry bit CY of status register STA is set, if an unsigned overflow occurred during addition, otherwise the bit is cleared. An unsigned overflow has occurred when the result of the operation cannot be represented in the interval [0; 2^{W}-1], assuming that both operands A and B are unsigned values within the interval [0; 2^{W}-1].

The overflow bit V of status register STA is set, if a signed overflow/underflow occurred during addition, otherwise the bit is cleared. A signed overflow/underflow has occurred when the result of the operation cannot be represented in the interval [-2^{W-1}; 2^{W-1}-1], assuming that both operands A and B are signed values within the interval [-2^{W-1}; 2^{W-1}-1].

The negative bit N of status register STA equals the MSB of the operation result, in order to determine if a calculated signed result is negative (N=1) or positive (N=0), assuming that no overflow/underflow occurred.

The program counter PC is incremented by the value 4.

15.7.30 ADDC Instruction

Syntax: ADDC A, B
Operation: A ← A + B + CY
Status: Z, CY, N, V
Duration: 1 instruction cycle
Description: Perform addition operation of a register A (A ∈ XOREG) with an operand B (B ∈ XOREG) and the carry flag CY. The result is stored in the register A.

The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the zero bit is cleared.

The carry bit CY of status register STA is set, if an unsigned overflow occurred during addition, otherwise the bit is cleared. An unsigned overflow has occurred when the result of the operation cannot be represented in the interval [0; 2^{W}-1], assuming that both operands A and B are unsigned values within the interval [0; 2^{W}-1].

The overflow bit V of status register STA is set, if a signed overflow/underflow occurred during addition, otherwise the bit is cleared. A signed overflow/underflow has occurred when the result of the operation cannot be represented in the interval [-2^{W-1}; 2^{W-1}-1], assuming that both operands A and B are signed values within the interval [-2^{W-1}; 2^{W-1}-1].

The negative bit N of status register STA equals the MSB of the operation result, in order to determine if a calculated signed result is negative (N=1) or positive (N=0), assuming that no overflow/underflow occurred.

The program counter PC is incremented by the value 4.

15.7.31 SUBL Instruction

Syntax: SUBL A, C
Operation: A ← A - C
Status: Z, CY, N, V
Duration: 1 instruction cycle
Description: Perform subtraction operation of a register A (A ∈ OREG) with a W bit literal value C (C ∈ WLIT). The result is stored in register A.

The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the zero bit is cleared.

The carry bit CY of status register STA is set, if an unsigned underflow occurred during subtraction, otherwise the bit is cleared. An unsigned underflow has occurred when the result of the operation cannot be represented in the interval [0; 2^W-1], assuming that both operands A and C are unsigned values within the interval [0; 2^W-1].

The overflow bit V of status register STA is set, if a signed overflow/underflow occurred during subtraction, otherwise the bit is cleared. A signed overflow/underflow has occurred when the result of the operation cannot be represented in the interval [-2^{W-1}; 2^{W-1}-1], assuming that both operands A and C are signed values within the interval [-2^{W-1}; 2^{W-1}-1].

The negative bit N of status register STA equals the MSB of the operation result, in order to determine if a calculated signed result is negative (N=1) or positive (N=0), assuming that no overflow/underflow occurred.

The program counter PC is incremented by the value 4.

15.7.32 SUB Instruction

Syntax: SUB A, B
Operation: A ← A - B
Status: Z, CY, N, V

Duration: 1 instruction cycle

Description: Perform subtraction operation of a register A (A ∊ XOREG) with an operand B (B ∊ XOREG). The result is stored in register A.

The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the zero bit is cleared.

The carry bit CY of status register STA is set, if an unsigned underflow occurred during subtraction, otherwise the bit is cleared. An unsigned underflow has occurred when the result of the operation cannot be represented in the interval [0; 2^W-1], assuming that both operands A and B are unsigned values within the interval [0; 2^W-1].

The overflow bit V of status register STA is set, if a signed overflow/underflow occurred during subtraction, otherwise the bit is cleared. A signed overflow/underflow has occurred when the result of the operation cannot be represented in the interval [-2^{W-1}; 2^{W-1}-1], assuming that both operands A and B are signed values within the interval [-2^{W-1}; 2^{W-1}-1].

The negative bit N of status register STA equals the MSB of the operation result, in order to determine if a calculated signed result is negative (N=1) or positive (N=0), assuming that no overflow/underflow occurred.

The program counter PC is incremented by the value 4.

15.7.33 SUBC Instruction

Syntax: SUBC A, B

Operation: A ← A - B - CY

Status: Z, CY, N, V

Duration: 1 instruction cycle

Description: Perform subtraction operation of a register A (A ∊ XOREG) with an operand B (B ∊ XOREG) and the carry flag CY. The result is stored in register A.

The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the zero bit is cleared.

The carry bit CY of status register STA is set, if an unsigned underflow occurred during subtraction, otherwise the bit is cleared. An unsigned underflow has occurred when the result of the operation cannot be represented in the interval [0; 2^W-1], assuming that both operands A and B are unsigned values within the interval [0; 2^W-1].

The overflow bit V of status register STA is set, if a signed overflow/underflow occurred during subtraction, otherwise the bit is cleared. A signed overflow/underflow has occurred when the result of the operation cannot be represented in the interval [-2^{W-1}; 2^{W-1}-1], assuming that both operands A and B are signed values within the interval [-2^{W-1}; 2^{W-1}-1].

The negative bit N of status register STA equals the MSB of the operation result, in order to determine if a calculated signed result is negative (N=1) or positive (N=0), assuming that no overflow/underflow occurred.

The program counter PC is incremented by the value 4.

15.7.34 NEG Instruction

Syntax: NEG A, B
Operation: A ← - B
Status: Z, N, V
Duration: 1 instruction cycle
Description: Perform negation operation (2's Complement) with an operand B (B ∈ XOREG) and store the result in a register A (A ∈ XOREG).

The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the zero bit is cleared.

The overflow bit V of status register STA is set, if a signed overflow/underflow occurred during subtraction, otherwise the bit is cleared. A signed overflow/underflow has occurred when the result of the operation cannot be represented in the interval [-2^{W-1}; 2^{W-1}-1], assuming that both operands A and B are signed values within the interval [-2^{W-1}; 2^{W-1}-1].

The negative bit N of status register STA equals the MSB of the operation result, in order to determine if a calculated signed result is negative (N=1) or positive (N=0), assuming that no overflow/underflow occurred.

The program counter PC is incremented by the value 4.

15.7.35 ANDL Instruction

Syntax: ANDL A, C
Operation: A ← A AND C
Status: Z
Duration: 1 instruction cycle
Description: Perform bitwise AND conjunction of a register A (A ∈ OREG) with a W bit literal value C (C ∈ WLIT) and store the result in register A.

The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the zero bit is cleared.

The program counter PC is incremented by the value 4.

15.7.36 AND Instruction

Syntax: AND A, B
Operation: A ← A AND B
Status: Z
Duration: 1 instruction cycle
Description: Perform bitwise AND conjunction of a register A (A ∈ XOREG) with an operand B (B ∈ XOREG) and store the result in register A.

The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the zero bit is cleared.

The program counter PC is incremented by the value 4.

15.7.37 ORL Instruction

Syntax: ORL A, C
Operation: A ← A OR C
Status: Z
Duration: 1 instruction cycle
Description: Perform bitwise OR conjunction of a register A (A ∈ OREG) with a W bit literal value C (C ∈ WLIT) and store the result in register A.

The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the zero bit is cleared.

The program counter PC is incremented by the value 4.

15.7.38 OR Instruction

Syntax: OR A, B
Operation: A ← A OR B
Status: Z
Duration: 1 instruction cycle
Description: Perform bitwise OR conjunction of a register A (A ∈ XOREG) with an operand B (B ∈ XOREG) and store the result in register A.

The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the zero bit is cleared.

The program counter PC is incremented by the value 4.

15.7.39 XORL Instruction

Syntax: XORL A, C
Operation: A ← A XOR C
Status: Z
Duration: 1 instruction cycle
Description: Perform bitwise XOR conjunction of a register A (A ∊ OREG) with a W bit literal value C (C ∊ WLIT) and store the result in register A.

The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the zero bit is cleared.

The program counter PC is incremented by the value 4.

15.7.40 XOR Instruction

Syntax: XOR A, B
Operation: A ← A XOR B
Status: Z
Duration: 1 instruction cycle
Description: Perform bitwise XOR conjunction of a register A (A ∊ XOREG) with an operand B (B ∊ XOREG) and store the result in register A.

The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the zero bit is cleared.

The program counter PC is incremented by the value 4.

15.7.41 SHR Instruction

Syntax: SHR A, C
Operation: A ← A >> C
Status: Z, CY
Duration: 1 instruction cycle
Description: Perform right shift operation C (C ∊ SFTLIT) times of register A (A ∊ XOREG). The MSBs that are shifted into A are cleared.

The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the zero bit is cleared.

The carry bit CY of status register STA is updated to the last LSB that is shifted out of the register. If the shift value C is 0 the carry bit CY is cleared.

The program counter PC is incremented by the value 4.

15.7.42 SHL Instruction

Syntax: SHL A, C
Operation: A ← A << C
Status: Z, CY
Duration: 1 instruction cycle
Description: Perform left shift operation C (C ∊ SFTLIT) times of register A (A ∊ XOREG). The LSBs that are shifted into A are cleared.

The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the zero bit is cleared.

The carry bit CY of status register STA is updated to the previous MSB that is shifted out of the register. If the register A contains less than W bits or if C is 0 the carry bit CY is always cleared.

The program counter PC is incremented by the value 4.

15.7.43 ASRU Instruction

Syntax: ASRU A, B
Operation: A ← A >> B
Status: Z
Duration: 1 instruction cycle
Description: Perform arithmetic unsigned right shift operation, which means that the unsigned operand of register A (A ∊ XOREG) is right shifted B times (B ∊ XOREG). Operand B is also an unsigned type. The MSBs that are shifted into A are cleared.

The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the zero bit is cleared.

The program counter PC is incremented by the value 4.

15.7.44 ASRS Instruction

Syntax: ASRS A, B
Operation: A ← A >> B
Status: Z
Duration: 1 instruction cycle
Description: Perform arithmetic signed right shift operation, which means that the signed operand of register A (A ∊ XOREG) is right shifted B times (B ∊ XOREG). Operand B is an unsigned type. The operation also performs a sign extension, which means that value of the MSBs that are shifted into A are determined by the MSB of the original operand A.

The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the zero bit is cleared.

The program counter PC is incremented by the value 4.

15.7.45 ASL Instruction

Syntax: ASL A, B
Operation: A ← A << B
Status: Z, CY, V
Duration: 1 instruction cycle
Description: Perform arithmetic left shift operation for signed and unsigned numbers, which means that the operand of register A (A ∊ XOREG) is left shifted B times (B ∊ XOREG). Operand B is always an unsigned type.

The carry bit CY of status register STA is set, if an unsigned overflow occurred during shifting, otherwise the bit is cleared. An unsigned overflow has occurred if the calculated result $A*2^B$ cannot be represented in the interval $[0; 2^W-1]$, assuming that both operands A and B are unsigned values within the interval $[0; 2^W-1]$.

The overflow bit V of status register STA is set, if a signed overflow/underflow occurred during shifting, otherwise the bit is cleared. A signed overflow/underflow has occurred when the calculated result $A*2^B$ cannot be represented in the interval $[-2^{W-1}; 2^{W-1}-1]$, assuming that signed operand A is within the interval $[-2^{W-1}; 2^{W-1}-1]$ and the unsigned operand B is within the interval $[0; 2^{W-1}-1]$.

The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the zero bit is cleared.

The program counter PC is incremented by the value 4.

15.7.46 MULU Instruction

Syntax: MULU A, B[, C]
Operation: [[R4,] A] ← A[(C-1):0] * B[(C-1):0]
Status: Z
Duration: 1 instruction cycle
Description: Perform an unsigned multiplication operation of an operand A (A ∈ XOREG) with an operand B (B ∈ XOREG). The multiplication is only performed with the bits 0 to C-1 (C ∈ BWSLIT) of both operands A and B and the bits C to W-1 are ignored. If C is less than or equal to W/2, the product of the multiplication is stored in register A and register R4 is left unchanged. If C is greater than W/2, the bits 0 to W-1 are stored in A and the bits W to 2*C-1 are stored in R4. The results stored in the registers are always zero extended to W bits.

If the optional operand C is not specified in the assembler code, the MCS assembler generates code with a default value of W for operand C.

The zero bit Z of status register STA is set, if the calculated product is zero, otherwise the zero bit is cleared.

If the delay between the a MULU instruction and its successor instruction is one system clock cycles (e.g. in accelerated scheduling mode) and the successor instruction is either a WURM, WURMX, WURCX, WUCE, BRDI, BWR, or BWRI instruction that is accessing the multiplication result as argument a data hazard in the pipeline occurs resulting in a pipeline flush.

This means, if very fast program execution is required (e.g. only one task is activated in accelerated scheduling mode) a program sequence like MULU R1, R2; BWRI R1, R3; (9 clock cycles) can be accelerated by reformulating the sequence as MULU R1, R2; NOP; BWRI R1, R3; (3 clock cycles).

The program counter PC is incremented by the value 4.

15.7.47 MULS Instruction

Syntax: MULS A, B[, C]
Operation: [[R4,] A] ← A[(C-1):0] * B[(C-1):0]
Status: Z, N
Duration: 1 instruction cycle
Description: Perform a signed multiplication operation of an operand A (A ∈ XOREG) with an operand B (B ∈ XOREG). The multiplication is only performed with the bits 0 to C-1 (C ∈ BWSLIT) of both operands A and B, in which bit C-1 is used as sign bit (-2^(C-1)) and the bits C to W-1 are ignored. If C is less than or equal to W/2, the product of the multiplication is stored in register A and register R4 is left unchanged. If C is

greater than W/2, the bits 0 to W-1 are stored in A and the bits W to 2*C-1 are stored in R4. The results stored in the registers are always sign extended to W bits.

If the optional operand C is not specified in the assembler code, the MCS assembler generates code with a default value of W for operand C.

The zero bit Z of status register STA is set, if the calculated product is zero, otherwise the zero bit is cleared.

The negative bit N of status register STA equals the MSB of the operation result, in order to determine if a calculated signed result is negative (N=1) or positive (N=0).

If the delay between the a MULS instruction and its successor instruction is one system clock cycles (e.g. in accelerated scheduling mode) and the successor instruction is either a WURM, WURMX, WURCX, WUCE, BRDI, BWR, or BWRI instruction that is accessing the multiplication result as argument a data hazard in the pipeline occurs resulting in a pipeline flush.

This means, if very fast program execution is required (e.g. only one task is activated in accelerated scheduling mode) a program sequence like MULS R1, R2; BWRI R1, R3; (9 clock cycles) can be accelerated by reformulating the sequence as MULS R1, R2; NOP; BWRI R1, R3; (3 clock cycles).

The program counter PC is incremented by the value 4.

15.7.48 DIVU Instruction

Syntax: DIVU A, B[, C]
Operation: R4 $\leftarrow$ A[(C-1):0] - B[(C-1):0] * $\lfloor$A[(C-1):0] / B[(C-1):0]$\rfloor$;
A $\leftarrow$ $\lfloor$A[(C-1):0] / B[(C-1):0]$\rfloor$
Status: CY, Z, ERR
Duration: C instruction cycles but not faster than C+NPS-1 clock cycles due to pipeline flushing.

Description: Perform an unsigned division operation of operand A (A $\in$ XOREG \ {R4, B}) divided by operand B (B $\in$ XOREG \ {R4, A}). The division is only performed with the bits 0 to C-1 (C $\in$ BWSLIT) of the operands and the remaining bits C to W-1 are ignored. This means that the dynamic range of A and B is defined in the interval [0; 2^{C}-1]. The integral part of the quotient is stored in the register A and the remainder of the division is stored in register R4. The resulting quotient A and remainder R4 are always zero extended to W bits.

If the optional operand C is not specified in the assembler code, the MCS assembler generates code with a default value of W for operand C.

If the bits 0 to C-1 of operand B are zero, the MCS channel is disabled and the ERR bit in the status register STA is set.

The zero bit Z of status register STA is set, if the calculated quotient in A is zero, otherwise the zero bit is cleared.

The carry bit CY of status register STA is set, if the calculated remainder in R4 is not zero, otherwise the zero bit is cleared.

The program counter PC is incremented by the value 4.

15.7.49 DIVS Instruction

Syntax: DIVS A, B[, C]
Operation: R4 $\leftarrow$ A[(C-1):0] - B[(C-1):0] * $\lfloor$A[(C-1):0] / B[(C-1):0]$\rfloor$;
A $\leftarrow$ $\lfloor$A[(C-1):0] / B[(C-1):0]$\rfloor$

Status: CY, Z, N, V, ERR
Duration: C + 4 instruction cycles but not faster than C+3+NPS clock cycles due to pipeline flushing.
Description: Perform a signed division operation of operand A (A $\in$ XOREG \ {R4, B}) divided by operand B (B $\in$ XOREG \ {R4, A}). The division is only performed with the bits 0 to C-1 (C $\in$ BWSLIT) of both operands, in which bit C-1 is used as sign bit (-2^{C-1}) and the bits C to W-1 are ignored. This means that the dynamic range of A[(C-1):0] and B[(C-1):0] is defined in the interval [-2^{C-1}; $2^{C-1}-1$]. The integral part of the quotient is stored in the register A and the remainder of the division is stored in register R4. The resulting quotient A and remainder R4 are always sign extended to W bits. The integral part of the quotient is always truncated towards 0. The sign of the remainder is always the same sign as the dividend A[(C-1):0]. The absolute value of the remained is always less than the divisor B[(C-1):0].

If the optional operand C is not specified in the assembler code, the MCS assembler generates code with a default value of W for operand C.

If the bits 0 to C-1 of operand B are zero, the MCS channel is disabled and the ERR bit in the status register STA is set.

The zero bit Z of status register STA is set, if the calculated quotient in A is zero, otherwise the zero bit is cleared.

The carry bit CY of status register STA is set, if the calculated remainder in R4 is not zero, otherwise the zero bit is cleared.

The overflow bit V of status register STA is set, if the calculated quotient in A cannot be represented in the interval [-2^{W-1}; $2^{W-1}-1$], otherwise the overflow bit is cleared.

The negative bit N of status register STA equals the MSB of the quotient, in order to determine if a calculated signed result is negative (N=1) or positive (N=0).

The program counter PC is incremented by the value 4.

15.7.50 MINU Instruction

Syntax: MINU A, B
Operation: A ← MIN(A, B)
Status: Z
Duration: 1 instruction cycle
Description: Determine the minimum of an unsigned operand A (A ∊ XOREG) and an unsigned operand B (B ∊ XOREG). If A is less than or equal to B, A is left unchanged. Otherwise, if A is greater than B, the operand B is moved to A.

The zero bit Z of status register STA is set, if the calculated result of A is zero, otherwise the zero bit is cleared.

15.7.51 MINS Instruction

Syntax: MINS A, B
Operation: A ← MIN(A, B)
Status: Z
Duration: 1 instruction cycle
Description: Determine the minimum of a signed operand A (A ∊ XOREG) and a signed operand B (B ∊ XOREG). If A is less than or equal to B, A is left unchanged. Otherwise, if A is greater than B, the operand B is moved to A.

The zero bit Z of status register STA is set, if the calculated result of A is zero, otherwise the zero bit is cleared.

15.7.52 MAXU Instruction

Syntax: MAXU A, B
Operation: A ← MAX(A, B)
Status: Z
Duration: 1 instruction cycle
Description: Determine the maximum of an unsigned operand A (A ∊ XOREG) and an unsigned operand B (B ∊ XOREG). If A is greater than or equal to B, A is left unchanged. Otherwise, if A is less than B, the operand B is moved to A.

The zero bit Z of status register STA is set, if the calculated result of A is zero, otherwise the zero bit is cleared.

15.7.53 MAXS Instruction

Syntax: MAXS A, B
Operation: A ← MAX(A, B)
Status: Z
Duration: 1 instruction cycle
Description: Determine the maximum of a signed operand A (A ∈ XOREG) and a signed operand B (B ∈ XOREG). If A is greater than or equal to B, A is left unchanged. Otherwise, if A is less than B, the operand B is moved to A.

The zero bit Z of status register STA is set, if the calculated result of A is zero, otherwise the zero bit is cleared.

15.7.54 ATUL Instruction

Syntax: ATUL A, C
Operation: A - C
Status: Z, CY
Duration: 1 instruction cycle
Description: Arithmetic test with an unsigned operand A (A ∈ OREG) and an unsigned W bit literal value C (C ∈ WLIT).

The carry bit CY of status register STA is set if unsigned operand A is less than unsigned literal C.
Otherwise, the carry bit CY of status register STA is cleared if unsigned operand A is greater than or equal to unsigned literal C.

The zero bit Z of status register STA is set, if A equals to C.
Otherwise, the zero bit Z of status register STA is cleared, if A is unequal to C.
The program counter PC is incremented by the value 4.

15.7.55 ATU Instruction

Syntax: ATU A, B
Operation: A - B
Status: Z, CY
Duration: 1 instruction cycle

Description: Arithmetic Test with an unsigned operand A (A ∈ XOREG) and an unsigned operand B (B ∈ XOREG).
The carry bit CY of status register STA is set if unsigned operand A is less than unsigned operand B.
Otherwise, the carry bit CY of status register STA is cleared if unsigned operand A is greater than or equal to unsigned operand B.

The zero bit Z of status register STA is set, if A equals to B.
Otherwise, the zero bit Z of status register STA is cleared, if A is unequal to B.
The program counter PC is incremented by the value 4.

15.7.56 ATSL Instruction

Syntax: ATSL A, C
Operation: A - C
Status: Z, CY
Duration: 1 instruction cycle
Description: Arithmetic Test with a signed operand A (A ∈ OREG) and a signed W bit literal value C (C ∈ WLIT).
The carry bit CY of status register STA is set if signed operand A is less than signed literal C.
Otherwise, the carry bit CY of status register STA is cleared if signed operand A is greater than or equal to signed literal C.

The zero bit Z of status register STA is set, if A equals to C.
Otherwise, the zero bit Z of status register STA is cleared, if A is unequal to C.
The program counter PC is incremented by the value 4.

15.7.57 ATS Instruction

Syntax: ATS A, B
Operation: A - B
Status: Z, CY
Duration: 1 instruction cycle
Description: Arithmetic Test with a signed operand A (A ∈ XOREG) and a signed operand B (B ∈ XOREG).
The carry bit CY of status register STA is set if signed operand A is less than signed operand B.
Otherwise, the carry bit CY of status register STA is cleared if signed operand A is greater than or equal to signed operand B.

The zero bit Z of status register STA is set, if A equals to B.
Otherwise, the zero bit Z of status register STA is cleared, if A is unequal to B.

The program counter PC is incremented by the value 4.

15.7.58 BTL Instruction

Syntax: BTL A, C
Operation: A AND C
Status: Z
Duration: 1 instruction cycle
Description: Bit test of an operand A (A ∊ OREG) with a W bit literal bit mask C (C ∊ WLIT).
The bit test is performed by applying a bitwise logical AND operation with operand A and the bit mask C without storing the result.

The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the zero bit is cleared.

The program counter PC is incremented by the value 4.

15.7.59 BT Instruction

Syntax: BT A, B
Operation: A AND B
Status: Z
Duration: 1 instruction cycle
Description: Bit test of an operand A (A ∊ XOREG) with an operand B (B ∊ XOREG), whereas usually one of the operands is a register holding a bit mask.

The bit test is performed by applying a bitwise logical AND operation with register A and register B without storing the result.

The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the zero bit is cleared.

The program counter PC is incremented by the value 4.

15.7.60 SETB Instruction

Syntax: SETB A, B
Operation: A[B[4:0]] ← 1
Status: Z
Duration: 1 instruction cycle

Description: Set the B[4:0]-th bit (B ∊ XOREG) of an operand A (A ∊ XOREG) to true. Only the bits 0 to 4 of operand B are used as bit index of operand A and the other bits of B are ignored. If the value B[4:0] is greater than or equal to W, the operation of SETB does not modify operand A but the status flag Z is updated.

The zero bit Z of status register STA is set if the modified value of A is zero, otherwise the bit Z is cleared. The Z bit is set e.g. if the B[4:0]-th bit of A is not writable, its value is zero and all other bits of A are cleared.

The program counter PC is incremented by the value 4.

15.7.61 CLRB Instruction

Syntax: CLRB A, B
Operation: A[B[4:0]] ← 0
Status: Z
Duration: 1 instruction cycle
Description: Clear the B[4:0]-th bit (B ∊ XOREG) of an operand A (A ∊ XOREG). Only the bits 0 to 4 of operand B are used as bit index of operand A and the other bits of B are ignored. If the value B[4:0] is greater than or equal to W, the operation of CLRB does not modify operand A but the status flag Z is updated.

The zero bit Z of status register STA is set if the modified value of A is zero, otherwise the zero bit is cleared.

The program counter PC is incremented by the value 4.

15.7.62 XCHB Instruction

Syntax: XCHB A, B
Operation: A[B[4:0]] ↔ CY
Status: Z, CY
Duration: 1 instruction cycle
Description: Exchange the B[4:0]-th bit (B ∊ XOREG) of an operand A (A ∊ XOREG) with the CY bit in the status register. Only the bits 0 to 4 of operand B are used as bit index of operand A and the other bits of B are ignored. If the value B[4:0] is greater than or equal to W, the operation of XCHB does not modify operand A but the status flag Z is updated and the bit CY is cleared.

The zero bit Z of status register STA is set if the modified value of A is zero, otherwise the zero bit is cleared.

The program counter PC is incremented by the value 4.

15.7.63 JMP Instruction

Syntax: JMP C
Operation: PC ← C << 2
Status: -
Duration: 1 instruction cycle but not faster than NPS clock cycles due to pipeline flushing.

Description: Execute unconditional jump to the memory location C (C ∈ ALIT).

The program counter PC is loaded with literal C.

15.7.64 JBS Instruction

Syntax: JBS A, B, C
Operation: PC ← C << 2 if A[B] is set
PC ← PC + 4 if A[B] is clear

Status: -
Duration: 1 instruction cycle but if the jump is executed, it is not faster than NPS clock cycles due to pipeline flushing.

Description: Execute conditional jump to the memory location C (C ∈ ALIT).

The program counter PC is loaded with literal C if the bit at position B (B ∈ BITLIT) of operand A (A ∈ OREG) is set.

Otherwise, if the bit is cleared the program counter PC is incremented by the value 4.

15.7.65 JBC Instruction

Syntax: JBC A, B, C
Operation: PC ← C << 2 if A[B] is clear
PC ← PC + 4 if A[B] is set

Status: -
Duration: 1 instruction cycle but if the jump is executed it is not faster than NPS clock cycles due to pipeline flushing.

Description: Execute conditional jump to the memory location C (C ∊ ALIT).

The program counter PC is loaded with literal C if the bit at position B (B ∊ BITLIT) of operand A (A ∊ OREG) is cleared.

Otherwise, if the bit is set the program counter PC is incremented by the value 4.

15.7.66 CALL Instruction

Syntax: CALL C
Operation: R7 ← R7 + 4;
MEM(R7[RAW+USR+1:2])[RAW+USR+1:0] ← PC + 4;
PC ← C << 2;

SP_CNT ← SP_CNT + 1
Status: EN
Duration: 2 instruction cycles but not faster than 2*NPS clock cycles due to pipeline flushing.

Description: Call subprogram at memory location C (C ∊ ALIT).
The stack pointer register R7 is incremented by the value 4.
The memory location for the top of the stack is identified by the bits 2 to RAW+1 of the stack pointer register.

After the stack pointer is incremented, the incremented value of the PC is transferred to the top of the stack.

The program counter PC is loaded with literal C.
The **SP_CNT** bit field inside the **MCS[i]_CH[x]_CTRL** register is incremented.
If an overflow on the **SP_CNT** bit field occurs, the *STK_ERR[i]_IRQ* is raised.
If an overflow on the **SP_CNT** bit field occurs and the bit **HLT_SP_OFL** of register **MCS[i]_CTRL_STAT** is set, the channel current MCS-channel is disabled by clearing the **EN** bit of **STA**.

If an overflow on the **SP_CNT** bit field occurs and the bit **HLT_SP_OFL** of register **MCS[i]_CTRL_STAT** is set, the memory write operation of the incremented PC is discarding.

15.7.67 RET Instruction

Syntax: RET
Operation: PC ← MEM(R7[RAW+USR+1:2])[RAW+USR+1:2] << 2;

R7 ← R7 - 4;
SP_CNT ← SP_CNT - 1

Status: EN
Duration: 2 instruction cycles but not faster than 2*NPS clock cycles due to pipeline flushing.

Description: Return from subprogram.
The program counter PC is loaded with current value on the top of the stack.
Finally, the stack pointer register R7 is decremented by the value 4.
The memory location for the top of the stack is identified by the bits 2 to RAW+1 of the stack pointer register.

The **SP_CNT** bit field inside the **MCS[i]_CH[x]_CTRL** register is decremented.
If an underflow on the **SP_CNT** bit field occurs, the *STK_ERR[i]_IRQ* is raised.
If an underflow on the **SP_CNT** bit field occurs and the bit **HLT_SP_OFL** of register **MCS[i]_CTRL** is set, the channel current MCS-channel is disabled by clearing the **EN** bit of **STA**.

15.7.68 JMPI Instruction

Syntax: JMPI
Operation: PC ← R6[RAW+USR+1:2] << 2
Status: -
Duration: 1 instruction cycle but not faster than NPS clock cycles due to pipeline flushing.

Description: Execute indirect unconditional jump to the memory location provided in register R6. The destination address is only defined by the bits 2 to RAW+USR+1 of R6 and the other bits are ignored.

The program counter PC is loaded with (R6[RAW+USR+1:2] << 2).

15.7.69 JBSI Instruction

Syntax: JBSI A, B
Operation: PC ← R6[RAW+USR+1:2] << 2 if A[B] is set
PC ← PC + 4 if A[B] is clear

Status: -
Duration: 1 instruction cycle but if the jump is executed it is not faster than NPS clock cycles due to pipeline flushing.

Description: Execute indirect conditional jump to the memory location provided in register R6. The destination address is only defined by the bits 2 to RAW+USR+1 of R6 and the other bits are ignored.

The program counter PC is loaded with (R6[RAW+USR+1:2] << 2) only if the bit at position B (B ∊ XBITLIT) of operand A (A ∊ XOREG) is set.

Otherwise, if the bit is cleared the program counter PC is incremented by the value 4.

15.7.70 JBCI Instruction

Syntax: JBCI A, B
Operation: PC ← R6[RAW+USR+1:2] << 2 if A[B] is clear
PC ← PC + 4 if A[B] is set

Status: -
Duration: 1 instruction cycle but if the jump is executed it is not faster than NPS clock cycles due to pipeline flushing.

Description: Execute indirect conditional jump to the memory location provided in register R6. The destination address is only defined by the bits 2 to RAW+USR+1 of R6 and the other bits are ignored.

The program counter PC is loaded with (R6[RAW+USR+1:2] << 2) only if the bit at position B (B ∊ XBITLIT) of operand A (A ∊ XOREG) is cleared.

Otherwise, if the bit is set the program counter PC is incremented by the value 4.

15.7.71 CALLI Instruction

Syntax: CALLI
Operation: R7 ← R7 + 4;
MEM(R7[RAW+USR+1:2])[RAW+USR+1:0] ← PC + 4;
PC ← R6[RAW+USR+1:2] << 2;

SP_CNT ← SP_CNT + 1
Status: EN
Duration: 2 instruction cycles but not faster than 2*NPS clock cycles due to pipeline flushing.

Description: Call subprogram indirectly, where the register R6 is identifying the target memory location. The destination address is only defined by the bits 2 to RAW+USR+1 of R6 and the other bits are ignored.

The stack pointer register R7 is incremented by the value 4.
The memory location for the top of the stack is identified by the bits 2 to RAW+1 of the stack pointer register.

After the stack pointer is incremented, the incremented value of the PC is transferred to the top of the stack.

The program counter PC is loaded with (R6[RAW+USR+1:2] << 2),
The **SP_CNT** bit field inside the **MCS[i]_CH[x]_CTRL** register is incremented.
If an overflow on the **SP_CNT** bit field occurs, the *STK_ERR[i]_IRQ* is raised.
If an overflow on the **SP_CNT** bit field occurs and the bit **HLT_SP_OFL** of register **MCS[i]_CTRL_STAT** is set, the channel current MCS-channel is disabled by clearing the **EN** bit of **STA**.

If an overflow on the **SP_CNT** bit field occurs and the bit **HLT_SP_OFL** of register **MCS[i]_CTRL_STAT** is set, the memory write operation of the incremented PC is discarding.

15.7.72 WURM Instruction

Syntax: WURM A, B, C
Operation: Wait until register match.
Status: CWT
Duration: Suspends current MCS-channel. If the MCS is configured in Single Prioritization or Multiple Prioritization Scheduling Mode, the worst case latency for reactivating a prioritized MCS-channel is 2+NPS clock cycles. This is the delay between the match event of the corresponding WURM instruction and the beginning of the following MCS instruction.

Description: Suspend current MCS-channel until the following register match condition occurs:

A = (B AND MASK),

whereas A $\in$ OREG, B $\in$ OREG, AND is a bitwise AND operation with bitmask MASK. The bits 16 to 23 of MASK are set to true and the bits 0 to 15 are copied from the instructions literal C $\in$ MSKLIT. If the match condition evaluates to true, the suspended MCS channel is resumed and the program counter PC is incremented by the value 4 meaning that the MCS channel continues its program. However, if the match condition is true at the beginning of the instruction execution, the instruction does not suspend the channel and the program counter PC is incremented by the value 4.

At the beginning of the instruction execution the CWT bit in the register STA is always cleared. After the execution of the instruction the CWT bit is updated in order to show if the instruction finished successfully (CWT = 0) or it was canceled by the CPU (CWT = 1).

This instruction can be used to wait for one or more trigger events generated by other MCS-channels or the CPU. In this case register B is the trigger register **STRG**, A is a general purpose register holding the bits with the trigger condition to wait for and C is the bitmask that enables trigger bits of interest. The trigger bits can be set by other MCS channels with a write access (e.g. using a MOVL instruction) to the **STRG** register or the CPU with a write access to the **MCS[i]_STRG** register. The trigger bits are not cleared automatically by hardware after resuming an MCS-channel, but they have to be cleared explicitly with a write access to the register **CTRG** by the MCS-channel or with a write access to the register **MCS[i]_CTRG** by the CPU. Please note that more than one channel can wait for the same trigger bit to continue.

The instruction can also be used to wait on a specific time/angle event provided by the TBU. In this case register B is the interesting TBU register TBU_TS0 or TBU_TS1, register A is a general purpose register holding the value to wait for and bitmask C should be set to 0xFFFF.

15.7.73 WURMX Instruction

Syntax: WURMX A, B
Operation: Wait until extended register match.
Status: CWT
Duration: Suspends current MCS-channel. If the MCS is configured in Single Prioritization or Multiple Prioritization Scheduling Mode, the worst case latency for reactivating a prioritized MCS-channel is 2+NPS clock cycles. This is the delay between the match event of the corresponding WURMX instruction and the beginning of the following MCS instruction.

Description: Suspend current MCS-channel until the following register match condition occurs:

A = B AND R6,

whereas A $\in$ OREG, B $\in$ WXREG, and AND is a bitwise AND operation. If the match condition evaluates to true, the suspended MCS channel is resumed and the program counter PC is incremented by the value 4 meaning that the MCS channel continues its program. However, if the match condition is true at the beginning of the instruction execution, the instruction does not suspend the channel and the program counter PC is incremented by the value 4.

At the beginning of the instruction execution the CWT bit in the register STA is always cleared. After the execution of the instruction the CWT bit is updated in order to show if the instruction finished successfully (CWT = 0) or it was canceled by the CPU (CWT = 1).

15.7.74 WURCX Instruction

Syntax: WURCX A, B
Operation: Wait until extended register change.
Status: CWT
Duration: Suspends current MCS-channel. If the MCS is configured in Single Prioritization or Multiple Prioritization Scheduling Mode, the worst case latency for reactivating a prioritized MCS-channel is 2+NPS clock cycles. This is the delay between the match event of the corresponding WURCX instruction and the beginning of the following MCS instruction.

Description: Suspend current MCS-channel until the following register change condition occurs:

$A \neq B$ AND R6,

whereas $A \in$ OREG, $B \in$ WXREG, and AND is a bitwise AND operation. If the change condition evaluates to true, the suspended MCS channel is resumed and the program counter PC is incremented by the value 4 meaning that the MCS channel continues its program. However, if the change condition is true at the beginning of the instruction execution, the instruction does not suspend the channel and the program counter PC is incremented by the value 4.

At the beginning of the instruction execution the CWT bit in the register STA is always cleared. After the execution of the instruction the CWT bit is updated in order to show if the instruction finished successfully (CWT = 0) or it was canceled by the CPU (CWT = 1).

The WURCX instruction can be used for observation of volatile registers (e.g. register DSTAX) in order to react on status signal changes.

15.7.75 WUCE Instruction

Syntax: WUCE A, B
Operation: Wait until cyclic event.
Status: CWT
Duration: Suspends current MCS-channel. If the MCS is configured in Single Prioritization or Multiple Prioritization Scheduling Mode, the worst case latency for

reactivating a prioritized MCS-channel is 2+NPS clock cycles. This is the delay between the match event of the corresponding WUCE instruction and the beginning of the following MCS instruction.

Description: Suspend current MCS-channel until a cyclic event compare matches. The meaning of a cyclic event is described in section 2.4.2. The WUCE instruction can be used to synchronize an MCS program to a cyclic event generated by a TBU channel. If the event is in the future, the MCS channel suspends until the event occurs. If the event is in the past, the WUCE instruction is finished immediatly.

The cyclic event compare is used to detect time base overflows and to guarantee, that a compare match event can be set up for the future even when the time base will first overflow and then reach the compare value. Please note, that for a correct behavior of this cyclic event compare, the compare value must not be specified larger/smaller than half of the range of the total time base value (0x7FFFFF).

The actual implementation of the WUCE implementation simply performs the subtraction B - A with each clock cycle and it suspends the MCS channel as long as bit W-1 of the subtraction result is set. If the subtration result is cleared the MCS channel is resumed immediatly.

In order to setup a WUCE instruction correctly, the counting direction of the TBU channel has to be considered. If the TBU channel is counting forward (incrementing), the operand A (A ∈ OREG) must refer the compare value and operand B (B ∈ OREG) must refer the desired TBU counter register (e.g. TBU_TS0). On the other hand, if the TBU channel is counting backward (decrementing), the operand A refers the desired TBU counter register and operand B referes the compare value.

If the comparison condition evaluates to true, the suspended MCS channel is resumed and the program counter PC is incremented by the value 4 meaning that the MCS channel continues its program. However, if the condition is true at the beginning of the instruction execution, the instruction does not suspend the channel and the program counter PC is incremented by the value 4.

At the beginning of the instruction execution the CWT bit in the register STA is always cleared. After the execution of the instruction the CWT bit is updated in order to show if the instruction finished successfully (CWT = 0) or it was canceled by the CPU (CWT = 1).

15.7.76 NOP Instruction

Syntax: NOP
Operation: -
Status: -
Duration: 1 instruction cycle
Description: No operation is performed.

The program counter PC is incremented by the value 4.

15.8 MCS Internal Register Overview

15.8.1 MCS Internal Register Overview

Register Name	Description	Details in Section
R[y] (y: 0..7)	General Purpose Register y	15.9.1
RS[y] (y: 0..7)	Mirror of succeeding channels register R[y]	15.9.2
STA	Status Register	15.9.3
ACB	ARU Control Bit Register	15.9.4
CTRG	Clear Trigger Bits Register	15.9.5
STRG	Set Trigger Bits Register	15.9.6
TBU_TS0	TBU Timestamp TS0 Register	15.9.7
TBU_TS1	TBU Timestamp TS1 Register	15.9.8
TBU_TS2	TBU Timestamp TS2 Register	15.9.9
MHB	Memory High Byte Register	15.9.10
GMI0	GTM Module Interrupt 0 Register	15.9.11
GMI1	GTM Module Interrupt 1 Register	15.9.12
DSTA	DPLL Status Register	15.9.13
DSTAX	DPLL Extended Status Register	15.9.14

15.9 MCS Internal Register Description

This section describes the MCS internal registers that can be directly addressed with the MCS instruction set. Many of the registers can also be addressed by the CPU but with another Register Label (for details see section 15.11). Some of the internal registers are also shared between neighboring MCS channels.

15.9.1 Register R[y] (y:0...7)

Address Offset:	**0x0+y**																**Initial Value:**								**0x000000**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	n/a								DATA																							
Mode									RW																							
Initial Value									0x0000_00																							

Bit 23:0 **DATA**: data field of general purpose register.
Bit 31:24 n/a

Note: Register **R4** is also used as destination register of upper multiplication result from instructions MULU and MULS.

Note: Register **R5** is also used as offset register for the instructions MRDIO and MWRIO.

Note: Register **R6** is also used as a mask register for the instruction WURMX and WURCX.

Note: Register **R6** is also used as address destination register for the instructions JMPI, JBSI, JBCI, and CALLI.

Note: Register **R6** used also as index/address register for indirect ARU addressing instructions.

Note: Register **R7** is also used as stack pointer register, if stack operations are used in the MCS micro program.

15.9.2 Register RS[y] (y:0...7)

Address Offset:	0x10+y																Initial Value: 0x000000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	n/a								DATA																							
Mode									RW																							
Initial Value									0x0000_00																							

Bit 23:0 **DATA**: data field of general purpose register.
Bit 31:24 n/a

Note: The register **RS[y]** (with y = 0 ... 7) mirrors the internal general purpose register **R[y]** of the succeeding MCS channel. The successor of MCS channel T-1 is MCS channel 0.

Note: The registers **RS[y]** can only be accessed if bit **EN_XOREG** of register **MCS[i]_CTRL_STAT** is set.

15.9.3 Register STA

Address Offset:	0x8																Initial Value: 0x000000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	n/a								Reserved					SP_CNT			Reserved					SAT	CWT	CAT	N	V	Z	CY	MCA	ERR	IRQ	EN
Mode									R					R			R					R	R	R	R	R	R	R	RAw	RW	RW	RW
Initial Value									0x00					000			0x00					0	0	0	0	0	0	0	0	0	0	0

Bit 0 **EN**: Enable current MCS-channel.
0 = Disable current MCS-channel.
1 = Enable current MCS-channel.

Bit 1 **IRQ**: Trigger IRQ.
0 = No triggered IRQ signal.
1 = Trigger IRQ signal.

Note: An MCS-channel triggers an IRQ by writing value 1 to bit IRQ. Writing a value 0 to this bit does not cancel the IRQ, and thus has no effect.

Note: This bit mirrors bit 0 of the register **MCS[i]_CH[x]_IRQ_NOTIFY**.

Note: The IRQ bit can only be cleared by CPU, by writing a 1 to the corresponding **MCS[i]_CH[x]_IRQ_NOTIFY** register (see section 15.11.6).

Note: An MCS-channel can read the IRQ bit in order to determine the current state of the IRQ handling. The MCS-channel reads a value 1 if an IRQ was released but not cleared by CPU. If an MCS-channel reads a value 0 no IRQ was released or it has been cleared by CPU.

Note: If NPS > 5 and an MCS program triggers the IRQ (e.g. by MOVL STA, 0x2) the actual interrupt event is delayed by NPS-5 clock cycles, which means that an immediate read of the interrupt notify flag (e.g. by MOV R2, STA) may signalize the state of the IRQ bit before the trigger.

Bit 2 **ERR**: Set Error Signal.

0 = No Error occurred.

1 = Error occurred.

Note: The **ERR** bit of an MCS-channel reflects an Error status that may be caused by one of the following conditions:

- MCS-channel sets the **ERR** bit by software (e.g. with instruction ORL STA, 0x4)
- ECC RAM Error occurred while accessing the connected RAM (also disables the MCS-channel by clearing bit **EN** and the first error occurred updates bit field **ERR_SRC_ID** of register **MCS[i]_CTRL_STAT**)
- Decoding an instruction with an invalid opcode (also disables the MCS-channel by clearing bit **EN** and the first error occurred updates bit field **ERR_SRC_ID** of register **MCS[i]_CTRL_STAT**)
- A memory address range overflow occurred (also disables the MCS-channel by clearing bit **EN** and the first error occurred updates bit field **ERR_SRC_ID** of register **MCS[i]_CTRL_STAT**)
- Division by zero resulting from a **DIVU** or **DIVS** instruction (also disables the MCS-channel by clearing bit **EN** and the first error occurred updates bit field **ERR_SRC_ID** of register **MCS[i]_CTRL_STAT**).
- MCS channel wants to write to a GPR that is write protected by register **MCS[i]_REG_PROT** (also disables the MCS-channel by clearing bit **EN** and the first error occurred updates bit field **ERR_SRC_ID** of register **MCS[i]_CTRL_STAT**).
- MCS channel wants to write to a protected memory range defined by an address range protector (ARP) of the sub module CCM (also disables the MCS-channel by clearing bit **EN** and the first error

occurred updates bit field **ERR_SRC_ID** of register **MCS[i]_CTRL_STAT**).

- MCS channel performs an invalid AEI bus master access while the bit field **HLT_AEIM_ERR** of register **MCS[i]_CTRL_STAT** is set (also disables the MCS-channel by clearing bit **EN** and the first error occurred updates bit field **ERR_SRC_ID** of register **MCS[i]_CTRL_STAT**)

Note: If the ERR bit is set due to a memory address range overflow any read or write access to the RAM is blocked.

Note: If the GTM includes a MON sub module, the ERR signal is always captured by this module.

Note: An MCS-channel can set the error bit by writing value 1 to bit ERR. Writing a value 0 to this bit does not cancel the error signal, and thus has no effect. In Addition, writing a value 1 to ERR always triggers the ERR interrupt, independently from the current state of the error signal.

Note: The ERR bit can only be cleared by CPU, by writing a 1 to the **MCS[i]_ERR** register (see section 15.11.18).

Note: An MCS-channel can read the ERR bit in order to determine the current state of the error signal. The MCS-channel reads a value 1 if an ERR occurred previously, but not cleared by CPU. If an MCS-channel reads a value 0 no error was set or it has been cleared by CPU.

Bit 3 **MCA**: MON Activity signaling for MCS channel.

0 = No activity signaled to sub module MON.

1 = Activity singled to sub module MON.

Note: When this bit is set the corresponding channel in the MON sub module register **MON_ACTIVITY** is set (see 22.8.2. This bit is automatically cleared after writing it by the MCS channel program.

Bit 4 **CY**: Carry bit.

The carry bit is updated by several arithmetic and logic instructions. In arithmetic operations, the carry bit indicates an unsigned under/overflow.

Bit 5 **Z**: Zero bit.

The zero bit is updated by several arithmetic, logic and data transfer instructions to indicate a result of zero.

Bit 6 **V**: Overflow bit.

The overflow bit is updated by arithmetic instructions in order to indicate a signed under/overflow.

Bit 7 **N**: Negative bit.

The negative bit is updated by arithmetic instructions in order to indicate a negative result.

Bit 8 **CAT**: Cancel ARU transfer bit.

0 = No cancellation request for ARU transfer.

1 = CPU requests cancellation for ARU transfer.

Note: This bit is always cleared at the beginning of the execution of a blocking ARU instruction.

Bit 9 **CWT**: Cancel WURM instruction bit.

0 = Last WURM instruction was not canceled

1 = CPU canceled last WURM instruction of channel.

Note: This bit is updated after each WURM instruction and it should be evaluated immediately after the WURM instruction. Otherwise, the CPU could set the bit leading to a bad status information in the MCS program.

Bit 10 **SAT**: Successful ARU transfer bit.

0 = ARU data transfer failed.

1 = ARU data transfer finished successfully.

Note: This bit is always updated after the execution of ARU instructions in order to show if the ARU data transfer was successful or not. In the case of non-blocking ARU instructions (NARD, NARDI) a cleared SAT flag signalizes that the data source has no data available and in the case of blocking ARU instructions (ARD, ARDI, AWR, AWRI) a cleared SAT flag signalizes that the data transfer was canceled by the CPU.

Bit 15:11 **Reserved:** Read as zero, should be written as zero.

Bit 18:16 **SP_CNT**: Stack pointer counter value.

Note: Actual stack depth of channel. The bit field is incremented on behalf of a CALL or PUSH instruction and decremented on behalf of a RET or POP instruction. The MCS channel STK_ERR_IRQ is raised, when an overflow or underflow is detected on this bit field.

Bit 23:19 **Reserved:** Read as zero, should be written as zero.

Bit 31:24 n/a

Note: Writing to bits of the register STA with instructions that do implicitly a read-modify-write operation (e.g. "ANDL STA 0xFFFFFE" or "OR STA R0") is dangerous, since writing back the possibly modified content of the read access (which reflects status information) may cause undesirable results. A secure way for writing to bits of the register STA is to use instructions that do not read the content of STA (e.g. "MOVL STA 0x0, MOV STA R1, CLRB STA R0, or SETB STA R1").

15.9.4 Register ACB

Address Offset:	**0x9**																**Initial Value:**							**0x000000**								
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	n/a								Reserved																			ACB4	ACB3	ACB2	ACB1	ACB0
Mode									R																			RW	RW	RW	RW	RW
Initial Value									0x0000																			0	0	0	0	0

Bit 0 **ACB0**: ARU Control bit 0.
Note: This bit is updated by each ARU read access and its value is sent to ARU by each ARU write access on bit 48 of the ARU word.

Bit 1 **ACB1**: ARU Control bit 1.
Note: This bit is updated by each ARU read access and its value is sent to ARU by each ARU write access on bit 49 of the ARU word.

Bit 2 **ACB2**: ARU Control bit 2.
Note: This bit is updated by each ARU read access and its value is sent to ARU by each ARU write access on bit 50 of the ARU word.

Bit 3 **ACB3**: ARU Control bit 3.
Note: This bit is updated by each ARU read access and its value is sent to ARU by each ARU write access on bit 51 of the ARU word.

Bit 4 **ACB4**: ARU Control bit 4.
Note: This bit is updated by each ARU read access and its value is sent to ARU by each ARU write access on bit 52 of the ARU word.

Bit 23:5 **Reserved:** Read as zero, should be written as zero.
Bit 31:24 n/a

15.9.5 Register CTRG

Address Offset:	**0xA**																**Initial Value: 0x000000**															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	n/a								TRG23	TRG22	TRG21	TRG20	TRG19	TRG18	TRG17	TRG16	TRG15	TRG14	TRG13	TRG12	TRG11	TRG10	TRG9	TRG8	TRG7	TRG6	TRG5	TRG4	TRG3	TRG2	TRG1	TRG0
Mode									RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Initial Value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit 0 **TRG0:** trigger bit 0.
READ access:
state of current trigger bit **TRG0** if **EN_TIM_FOUT** = 0
state of input signal **TIM[i]_CH0_F_OUT** if **EN_TIM_FOUT** = 1
WRITE access:
0 = do nothing
1 = clear trigger bit **TRG0**

Bit 1 **TRG1:** trigger bit 1.
READ access:
state of current trigger bit **TRG1** if **EN_TIM_FOUT** = 0
state of input signal **TIM[i]_CH1_F_OUT** if **EN_TIM_FOUT** = 1
WRITE access:
0 = do nothing
1 = clear trigger bit **TRG1**

Bit 2 **TRG2:** trigger bit 2.
READ access:
state of current trigger bit **TRG2** if **EN_TIM_FOUT** = 0
state of input signal **TIM[i]_CH2_F_OUT** if **EN_TIM_FOUT** = 1
WRITE access:
0 = do nothing
1 = clear trigger bit **TRG2**

Bit 3 **TRG3:** trigger bit 3.
READ access:
state of current trigger bit **TRG3** if **EN_TIM_FOUT** = 0
state of input signal **TIM[i]_CH3_F_OUT** if **EN_TIM_FOUT** = 1
WRITE access:
0 = do nothing
1 = clear trigger bit **TRG3**

Bit 4 **TRG4:** trigger bit 4.
READ access:

state of current trigger bit **TRG4** if **EN_TIM_FOUT** = 0
state of input signal **TIM[i]_CH4_F_OUT** if **EN_TIM_FOUT** = 1
WRITE access:
0 = do nothing
1 = clear trigger bit **TRG4**

Bit 5 **TRG5:** trigger bit 5.
READ access:
state of current trigger bit **TRG5** if **EN_TIM_FOUT** = 0
state of input signal **TIM[i]_CH5_F_OUT** if **EN_TIM_FOUT** = 1
WRITE access:
0 = do nothing
1 = clear trigger bit **TRG5**

Bit 6 **TRG6:** trigger bit 6.
READ access:
state of current trigger bit **TRG6** if **EN_TIM_FOUT** = 0
state of input signal **TIM[i]_CH6_F_OUT** if **EN_TIM_FOUT** = 1
WRITE access:
0 = do nothing
1 = clear trigger bit **TRG6**

Bit 7 **TRG7:** trigger bit 7.
READ access:
state of current trigger bit **TRG7** if **EN_TIM_FOUT** = 0
state of input signal **TIM[i]_CH7_F_OUT** if **EN_TIM_FOUT** = 1
WRITE access:
0 = do nothing
1 = clear trigger bit **TRG7**

Bit 8 **TRG8:** trigger bit 8.
READ access:
state of current trigger bit **TRG8** if **EN_TIM_FOUT** = 0
state of input signal **TIM[i+1]_CH0_F_OUT** if **EN_TIM_FOUT** = 1
WRITE access:
0 = do nothing
1 = clear trigger bit **TRG8**

Bit 9 **TRG9:** trigger bit 9.
READ access:
state of current trigger bit **TRG9** if **EN_TIM_FOUT** = 0
state of input signal **TIM[i+1]_CH1_F_OUT** if **EN_TIM_FOUT** = 1
WRITE access:
0 = do nothing
1 = clear trigger bit **TRG9**

Bit 10 **TRG10:** trigger bit 10.

READ access:
state of current trigger bit **TRG10** if **EN_TIM_FOUT** = 0
state of input signal **TIM[i+1]_CH2_F_OUT** if **EN_TIM_FOUT** = 1
WRITE access:
0 = do nothing
1 = clear trigger bit **TRG10**

Bit 11 **TRG11:** trigger bit 11.
READ access:
state of current trigger bit **TRG11** if **EN_TIM_FOUT** = 0
state of input signal **TIM[i+1]_CH3_F_OUT** if **EN_TIM_FOUT** = 1
WRITE access:
0 = do nothing
1 = clear trigger bit **TRG11**

Bit 12 **TRG12:** trigger bit 12.
READ access:
state of current trigger bit **TRG12** if **EN_TIM_FOUT** = 0
state of input signal **TIM[i+1]_CH4_F_OUT** if **EN_TIM_FOUT** = 1
WRITE access:
0 = do nothing
1 = clear trigger bit **TRG12**

Bit 13 **TRG13:** trigger bit 13.
READ access:
state of current trigger bit **TRG13** if **EN_TIM_FOUT** = 0
state of input signal **TIM[i+1]_CH5_F_OUT** if **EN_TIM_FOUT** = 1
WRITE access:
0 = do nothing
1 = clear trigger bit **TRG13**

Bit 14 **TRG14:** trigger bit 14.
READ access:
state of current trigger bit **TRG14** if **EN_TIM_FOUT** = 0
state of input signal **TIM[i+1]_CH6_F_OUT** if **EN_TIM_FOUT** = 1
WRITE access:
0 = do nothing
1 = clear trigger bit **TRG14**

Bit 15 **TRG15:** trigger bit 15.
READ access:
state of current trigger bit **TRG15** if **EN_TIM_FOUT** = 0
state of input signal **TIM[i+1]_CH7_F_OUT** if **EN_TIM_FOUT** = 1
WRITE access:
0 = do nothing
1 = clear trigger bit **TRG15**

Bit 16	**TRG16:** trigger bit 16. 0 = READ: trigger bit is cleared / WRITE : do nothing 1 = READ: trigger bit is set / WRITE : clear trigger bit
Bit 17	**TRG17:** trigger bit 17. 0 = READ: trigger bit is cleared / WRITE : do nothing 1 = READ: trigger bit is set / WRITE : clear trigger bit
Bit 18	**TRG18:** trigger bit 18. 0 = READ: trigger bit is cleared / WRITE : do nothing 1 = READ: trigger bit is set / WRITE : clear trigger bit
Bit 19	**TRG19:** trigger bit 19. 0 = READ: trigger bit is cleared / WRITE : do nothing 1 = READ: trigger bit is set / WRITE : clear trigger bit
Bit 20	**TRG20:** trigger bit 20. 0 = READ: trigger bit is cleared / WRITE : do nothing 1 = READ: trigger bit is set / WRITE : clear trigger bit
Bit 21	**TRG21:** trigger bit 21. 0 = READ: trigger bit is cleared / WRITE : do nothing 1 = READ: trigger bit is set / WRITE : clear trigger bit
Bit 22	**TRG22:** trigger bit 22. 0 = READ: trigger bit is cleared / WRITE : do nothing 1 = READ: trigger bit is set / WRITE : clear trigger bit
Bit 23	**TRG23:** trigger bit 23. 0 = READ: trigger bit is cleared / WRITE : do nothing 1 = READ: trigger bit is set / WRITE : clear trigger bit

Note: The trigger bits **TRGx** are accessible by all MCS channels as well as the CPU. Setting a trigger bit can be performed with the **STRG** register, in the case of an MCS-channel or the **MCS[i]_STRG** register in the case of the CPU. Clearing a trigger bit can be performed with the **CTRG** register, in the case of an MCS-channel or the **MCS[i]_CTRG** register in the case of the CPU. Trigger bits can be used for signalizing specific events to MCS-channels or the CPU. An MCS-channel suspended with a WURM instruction can be resumed by setting the appropriate trigger bit.

Note: Besides setting the trigger bits with register **STRG/MCS[i]_STRG**, the k-th trigger bit **TRGk** (with k < 16) can also be set by the external capture event that is enabled by the k-th bit of register **CCM[i]_EXT_CAP_EN**. If bit k bit is disabled, the k-th trigger bit **TRGk** can only be set by MCS or CPU.

Bit 31:24 n/a

Note: The result of a read access to this register differs in dependency of the bit field **EN_TIM_FOUT** of register **MCS[i]_CTRL_STAT**.

15.9.6 Register STRG

Address Offset:	0xB																Initial Value:								0x000000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	n/a								TRG23	TRG22	TRG21	TRG20	TRG19	TRG18	TRG17	TRG16	TRG15	TRG14	TRG13	TRG12	TRG11	TRG10	TRG9	TRG8	TRG7	TRG6	TRG5	TRG4	TRG3	TRG2	TRG1	TRG0
Mode									RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Initial Value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit 0 **TRG0:** trigger bit 0.
0 = READ: trigger bit is cleared / WRITE : do nothing
1 = READ: trigger bit is set / WRITE : set trigger bit

Bit 1 **TRG1:** trigger bit 1.
0 = READ: trigger bit is cleared / WRITE : do nothing
1 = READ: trigger bit is set / WRITE : set trigger bit

Bit 2 **TRG2:** trigger bit 2.
0 = READ: trigger bit is cleared / WRITE : do nothing
1 = READ: trigger bit is set / WRITE : set trigger bit

Bit 3 **TRG3:** trigger bit 3.
0 = READ: trigger bit is cleared / WRITE : do nothing
1 = READ: trigger bit is set / WRITE : set trigger bit

Bit 4 **TRG4:** trigger bit 4.
0 = READ: trigger bit is cleared / WRITE : do nothing
1 = READ: trigger bit is set / WRITE : set trigger bit

Bit 5 **TRG5:** trigger bit 5.
0 = READ: trigger bit is cleared / WRITE : do nothing
1 = READ: trigger bit is set / WRITE : set trigger bit

Bit 6 **TRG6:** trigger bit 6.
0 = READ: trigger bit is cleared / WRITE : do nothing
1 = READ: trigger bit is set / WRITE : set trigger bit

Bit 7 **TRG7:** trigger bit 7.
0 = READ: trigger bit is cleared / WRITE : do nothing
1 = READ: trigger bit is set / WRITE : set trigger bit

Bit 8 **TRG8:** trigger bit 8.
0 = READ: trigger bit is cleared / WRITE : do nothing
1 = READ: trigger bit is set / WRITE : set trigger bit

Bit 9 **TRG9:** trigger bit 9.
0 = READ: trigger bit is cleared / WRITE : do nothing
1 = READ: trigger bit is set / WRITE : set trigger bit

Bit 10 **TRG10:** trigger bit 10.
0 = READ: trigger bit is cleared / WRITE : do nothing

1 = READ: trigger bit is set / WRITE : set trigger bit

Bit 11 **TRG11:** trigger bit 11.
0 = READ: trigger bit is cleared / WRITE : do nothing
1 = READ: trigger bit is set / WRITE : set trigger bit

Bit 12 **TRG12:** trigger bit 12.
0 = READ: trigger bit is cleared / WRITE : do nothing
1 = READ: trigger bit is set / WRITE : set trigger bit

Bit 13 **TRG13:** trigger bit 13.
0 = READ: trigger bit is cleared / WRITE : do nothing
1 = READ: trigger bit is set / WRITE : set trigger bit

Bit 14 **TRG14:** trigger bit 14.
0 = READ: trigger bit is cleared / WRITE : do nothing
1 = READ: trigger bit is set / WRITE : set trigger bit

Bit 15 **TRG15:** trigger bit 15.
0 = READ: trigger bit is cleared / WRITE : do nothing
1 = READ: trigger bit is set / WRITE : set trigger bit

Bit 16 **TRG16:** trigger bit 16.
0 = READ: trigger bit is cleared / WRITE : do nothing
1 = READ: trigger bit is set / WRITE : set trigger bit

Bit 17 **TRG17:** trigger bit 17.
0 = READ: trigger bit is cleared / WRITE : do nothing
1 = READ: trigger bit is set / WRITE : set trigger bit

Bit 18 **TRG18:** trigger bit 18.
0 = READ: trigger bit is cleared / WRITE : do nothing
1 = READ: trigger bit is set / WRITE : set trigger bit

Bit 19 **TRG19:** trigger bit 19.
0 = READ: trigger bit is cleared / WRITE : do nothing
1 = READ: trigger bit is set / WRITE : set trigger bit

Bit 20 **TRG20:** trigger bit 20.
0 = READ: trigger bit is cleared / WRITE : do nothing
1 = READ: trigger bit is set / WRITE : set trigger bit

Bit 21 **TRG21:** trigger bit 21.
0 = READ: trigger bit is cleared / WRITE : do nothing
1 = READ: trigger bit is set / WRITE : set trigger bit

Bit 22 **TRG22:** trigger bit 22.
0 = READ: trigger bit is cleared / WRITE : do nothing
1 = READ: trigger bit is set / WRITE : set trigger bit

Bit 23 **TRG23:** trigger bit 23.
0 = READ: trigger bit is cleared / WRITE : do nothing
1 = READ: trigger bit is set / WRITE : set trigger bit

Note: The trigger bits **TRGx** are accessible by all MCS channels as well as the CPU. Setting a trigger bit can be performed with the **STRG** register, in the case of an MCS-channel or the **MCS[i]_STRG** register in the case of the CPU. Clearing a trigger bit can be performed with the **CTRG** register, in the case of an MCS-channel or the **MCS[i]_CTRG** register in the case of the CPU. Trigger bits can be used for signalizing specific events to MCS-channels or the

CPU. An MCS-channel suspended with a WURM instruction can be resumed by setting the appropriate trigger bit.

Note: Besides setting the trigger bits with register **STRG/MCS[i]_STRG**, the k-th trigger bit **TRGk** (with k < 16) can also be set by the external capture event that is enabled by the k-th bit of register **CCM[i]_EXT_CAP_EN**. If bit k bit is disabled, the k-th trigger bit **TRGk** can only be set by MCS or CPU.

Bit 31:24 n/a

15.9.7 Register TBU_TS0

Address Offset:	0xC																Initial Value:								0x000000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	n/a								TS																							
Mode									R																							
Initial Value									0x000000																							

Bit 23:0 **TS**: Current TBU time stamp 0.
Bit 31:24 n/a

15.9.8 Register TBU_TS1

Address Offset:	0xD																Initial Value:								0x000000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	n/a								TS																							
Mode									R																							
Initial Value									0x000000																							

Bit 23:0 **TS**: Current TBU time stamp 1.

Bit 31:24 n/a

15.9.9 Register TBU_TS2

Address Offset:	0xE																Initial Value:								0x000000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	n/a								TS																							
Mode									R																							
Initial Value									0x000000																							

Bit 23:0 **TS**: Current TBU time stamp 2.
Bit 31:24 n/a

15.9.10 Register MHB

Address Offset:	0xF																Initial Value:								0x000000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	n/a								Reserved																DATA							
Mode									R																RW							
Initial Value									0x0000																0x00							

Bit 7:0 **DATA**: High Byte of a memory transfer.
Bit 23:8 **Reserved:** Read as zero, should be written as zero.
Bit 31:24 n/a

15.9.11 Register GMI0

Address Offset:	**0x18**																**Initial Value:**								**0x000000**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	n/a								ATOM_CH7_IRQ	ATOM_CH6_IRQ	ATOM_CH5_IRQ	ATOM_CH4_IRQ	ATOM_CH3_IRQ	ATOM_CH2_IRQ	ATOM_CH1_IRQ	ATOM_CH0_IRQ	TOM_CH14_IRQ	TOM_CH12_IRQ	TOM_CH10_IRQ	TOM_CH8_IRQ	TOM_CH6_IRQ	TOM_CH4_IRQ	TOM_CH2_IRQ	TOM_CH0_IRQ	TIM_CH7_IRQ	TIM_CH6_IRQ	TIM_CH5_IRQ	TIM_CH4_IRQ	TIM_CH3_IRQ	TIM_CH2_IRQ	TIM_CH1_IRQ	TIM_CH0_IRQ
Mode									RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw
Initial Value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit 0 **TIM_CH0_IRQ**: TIM[i]_CH0 IRQ.
READ access:
IRQ signal *TIM[i]_CH0_IRQ*.
WRITE access:
0 = do nothing
1 = issue hw_clear on the connected IRQs.

Bit 1 **TIM_CH_IRQ**: TIM[i]_CH1 IRQ.
READ access:
IRQ signal *TIM[i]_CH1_IRQ*.
WRITE access:
0 = do nothing
1 = issue hw_clear on the connected IRQs.

Bit 2 **TIM_CH2_IRQ**: TIM[i]_CH2 IRQ.
READ access:
IRQ signal *TIM[i]_CH2_IRQ*.
WRITE access:
0 = do nothing
1 = issue hw_clear on the connected IRQs.

Bit 3 **TIM_CH3_IRQ**: TIM[i]_CH3 IRQ.
READ access:
IRQ signal *TIM[i]_CH3_IRQ*.
WRITE access:
0 = do nothing
1 = issue hw_clear on the connected IRQs.

Bit 4 **TIM_CH4_IRQ**: TIM[i]_CH4 IRQ.
READ access:

IRQ signal *TIM[i]_CH4_IRQ*.
WRITE access:
0 = do nothing
1 = issue hw_clear on the connected IRQs.

Bit 5 **TIM_CH5_IRQ**: TIM[i]_CH5 IRQ.
READ access:
IRQ signal *TIM[i]_CH5_IRQ*.
WRITE access:
0 = do nothing
1 = issue hw_clear on the connected IRQs.

Bit 6 **TIM_CH6_IRQ**: TIM[i]_CH6 IRQ.
READ access:
IRQ signal *TIM[i]_CH6_IRQ*.
WRITE access:
0 = do nothing
1 = issue hw_clear on the connected IRQs.

Bit 7 **TIM_CH7_IRQ**: TIM[i]_CH7 IRQ.
READ access:
IRQ signal *TIM[i]_CH7_IRQ*.
WRITE access:
0 = do nothing
1 = issue hw_clear on the connected IRQs.

Bit 8 **TOM_CH0_IRQ**: TOM[i]_CH0 or TOM[i]_CH1 IRQ.
READ access:
Logical OR conjunction of IRQ signals *TOM[i]_CH0_IRQ* and *TOM[i]_CH1_IRQ*.
WRITE access:
0 = do nothing
1 = issue hw_clear on the connected IRQs.

Bit 9 **TOM_CH2_IRQ**: TOM[i]_CH2 or TOM[i]_CH3 IRQ.
READ access:
Logical OR conjunction of IRQ signals *TOM[i]_CH2_IRQ* and *TOM[i]_CH3_IRQ*.
WRITE access:
0 = do nothing
1 = issue hw_clear on the connected IRQs.

Bit 10 **TOM_CH4_IRQ**: TOM[i]_CH4 or TOM[i]_CH5 IRQ.
READ access:
Logical OR conjunction of IRQ signals *TOM[i]_CH4_IRQ* and *TOM[i]_CH5_IRQ*.
WRITE access:
0 = do nothing
1 = issue hw_clear on the connected IRQs.

Bit 11 **TOM_CH6_IRQ**: TOM[i]_CH6 or TOM[i]_CH7 IRQ.
READ access:
Logical OR conjunction of IRQ signals *TOM[i]_CH6_IRQ* and *TOM[i]_CH7_IRQ*.
WRITE access:
0 = do nothing
1 = issue hw_clear on the connected IRQs.

Bit 12 **TOM_CH8_IRQ**: TOM[i]_CH8 or TOM[i]_CH9 IRQ.
READ access:
Logical OR conjunction of IRQ signals *TOM[i]_CH8_IRQ* and *TOM[i]_CH9_IRQ*.
WRITE access:
0 = do nothing
1 = issue hw_clear on the connected IRQs.

Bit 13 **TOM_CH10_IRQ**: TOM[i]_CH10 or TOM[i]_CH11 IRQ.
READ access:
Logical OR conjunction of IRQ signals *TOM[i]_CH10_IRQ* and *TOM[i]_CH11_IRQ*.
WRITE access:
0 = do nothing
1 = issue hw_clear on the connected IRQs.

Bit 14 **TOM_CH12_IRQ**: TOM[i]_CH12 or TOM[i]_CH13 IRQ.
READ access:
Logical OR conjunction of IRQ signals *TOM[i]_CH12_IRQ* and *TOM[i]_CH13_IRQ*.
WRITE access:
0 = do nothing
1 = issue hw_clear on the connected IRQs.

Bit 15 **TOM_CH14_IRQ**: TOM[i]_CH14 or TOM[i]_CH15 IRQ.

READ access:
Logical OR conjunction of IRQ signals *TOM[i]_CH14_IRQ* and *TOM[i]_CH15_IRQ*.
WRITE access:
0 = do nothing
1 = issue hw_clear on the connected IRQs.

Bit 16 **ATOM_CH0_IRQ**: ATOM[i]_CH0 IRQ.
READ access:
IRQ signal *ATOM[i]_CH0_IRQ*.
WRITE access:
0 = do nothing
1 = issue hw_clear on the connected IRQs.

Bit 17 **ATOM_CH1_IRQ**: ATOM[i]_CH1 IRQ.
READ access:
IRQ signal *ATOM[i]_CH1_IRQ*.
WRITE access:
0 = do nothing
1 = issue hw_clear on the connected IRQs.

Bit 18 **ATOM_CH2_IRQ**: ATOM[i]_CH2 IRQ.
READ access:
IRQ signal *ATOM[i]_CH2_IRQ*.
WRITE access:
0 = do nothing
1 = issue hw_clear on the connected IRQs.

Bit 19 **ATOM_CH3_IRQ**: ATOM[i]_CH3 IRQ.
READ access:
IRQ signal *ATOM[i]_CH3_IRQ*.
WRITE access:
0 = do nothing
1 = issue hw_clear on the connected IRQs.

Bit 20 **ATOM_CH4_IRQ**: ATOM[i]_CH4 IRQ.
READ access:
IRQ signal *ATOM[i]_CH4_IRQ*.
WRITE access:
0 = do nothing
1 = issue hw_clear on the connected IRQs.

Bit 21 **ATOM_CH5_IRQ**: ATOM[i]_CH5 IRQ.
READ access:
IRQ signal *ATOM[i]_CH5_IRQ*.
WRITE access:
0 = do nothing
1 = issue hw_clear on the connected IRQs.

Bit 22 **ATOM_CH6_IRQ**: ATOM[i]_CH6 IRQ.
READ access:
IRQ signal *ATOM[i]_CH6_IRQ*.
WRITE access:
0 = do nothing
1 = issue hw_clear on the connected IRQs.

Bit 23 **ATOM_CH7_IRQ**: ATOM[i]_CH7 IRQ.
READ access:
IRQ signal *ATOM[i]_CH7_IRQ*.
WRITE access:
0 = do nothing
1 = issue hw_clear on the connected IRQs.

Bit 31:24 n/a

15.9.12 Register GMI1

Address Offset:	**0x19**																**Initial Value:**								**0x000000**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	n/a								MCS0_CH7_IRQ	MCS0_CH6_IRQ	MCS0_CH5_IRQ	MCS0_CH4_IRQ	MCS0_CH3_IRQ	MCS0_CH2_IRQ	MCS0_CH1_IRQ	MCS0_CH0_IRQ	TTA_IP1_CH7_IR	TTA_IP1_CH6_IR	TTA_IP1_CH5_IR	TTA_IP1_CH4_IR	TTA_IP1_CH3_IR	TTA_IP1_CH2_IR	TTA_IP1_CH1_IR	TTA_IP1_CH0_IR	MCS_IP1_CH7_IR	MCS_IP1_CH6_IR	MCS_IP1_CH5_IR	MCS_IP1_CH4_IR	MCS_IP1_CH3_IR	MCS_IP1_CH2_IR	MCS_IP1_CH1_IR	MCS_IP1_CH0_IR
Mode									RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw
Initial Value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit 0 **MCS_IP1_CH0_IRQ**: MCS[i+1]_CH0 IRQ.
READ access:
IRQ signal *MCS[i+1]_CH0_IRQ*.
WRITE access:

0 = do nothing
1 = issue hw_clear on the connected IRQs.

Bit 1 **MCS_IP1_CH1_IRQ**: MCS[i+1]_CH1 IRQ.
READ access:
IRQ signal *MCS[i+1]_CH1_IRQ*.
WRITE access:
0 = do nothing
1 = issue hw_clear on the connected IRQs.

Bit 2 **MCS_IP1_CH2_IRQ**: MCS[i+1]_CH2 IRQ.
READ access:
IRQ signal *MCS[i+1]_CH2_IRQ*.
WRITE access:
0 = do nothing
1 = issue hw_clear on the connected IRQs.

Bit 3 **MCS_IP1_CH3_IRQ**: MCS[i+1]_CH3 IRQ.
READ access:
IRQ signal *MCS[i+1]_CH3_IRQ*.
WRITE access:
0 = do nothing
1 = issue hw_clear on the connected IRQs.

Bit 4 **MCS_IP1_CH4_IRQ**: MCS[i+1]_CH4 IRQ.
READ access:
IRQ signal *MCS[i+1]_CH4_IRQ*.
WRITE access:
0 = do nothing
1 = issue hw_clear on the connected IRQs.

Bit 5 **MCS_IP1_CH5_IRQ**: MCS[i+1]_CH5 IRQ.
READ access:
IRQ signal *MCS[i+1]_CH5_IRQ*.
WRITE access:
0 = do nothing
1 = issue hw_clear on the connected IRQs.

Bit 6 **MCS_IP1_CH6_IRQ**: MCS[i+1]_CH6 IRQ.
READ access:
IRQ signal *MCS[i+1]_CH6_IRQ*.

WRITE access:
0 = do nothing
1 = issue hw_clear on the connected IRQs.

Bit 7 **MCS_IP1_CH7_IRQ**: MCS[i+1]_CH7 IRQ.
READ access:
IRQ signal *MCS[i+1]_CH7_IRQ*.
WRITE access:
0 = do nothing
1 = issue hw_clear on the connected IRQs.

Bit 8 **TTA_IP1_CH0_IRQ**: Neighboring TIM, TOM, ATOM IRQs.
READ access:
Logical OR conjunction of IRQ signals *TIM[i+1]_CH0_IRQ*, *TOM[i+1]_CH0_IRQ*, *TOM[i+1]_CH1_IRQ*, and *ATOM[i+1]_CH0_IRQ*.
WRITE access:
0 = do nothing
1 = issue hw_clear on the connected IRQs.

Bit 9 **TTA_IP1_CH1_IRQ**: Neighboring TIM, TOM, ATOM IRQs.
READ access:
Logical OR conjunction of IRQ signals *TIM[i+1]_CH1_IRQ*, *TOM[i+1]_CH2_IRQ*, *TOM[i+1]_CH3_IRQ*, and *ATOM[i+1]_CH1_IRQ*.
WRITE access:
0 = do nothing
1 = issue hw_clear on the connected IRQs.

Bit 10 **TTA_IP1_CH2_IRQ**: Neighboring TIM, TOM, ATOM IRQs.
READ access:
Logical OR conjunction of IRQ signals *TIM[i+1]_CH2_IRQ*, *TOM[i+1]_CH4_IRQ*, *TOM[i+1]_CH5_IRQ*, and *ATOM[i+1]_CH2_IRQ*.
WRITE access:
0 = do nothing
1 = issue hw_clear on the connected IRQs.

Bit 11 **TTA_IP1_CH3_IRQ**: Neighboring TIM, TOM, ATOM IRQs.
READ access:

Logical OR conjunction of IRQ signals *TIM[i+1]_CH3_IRQ*, *TOM[i+1]_CH6_IRQ*, *TOM[i+1]_CH7_IRQ*, and *ATOM[i+1]_CH3_IRQ*.
WRITE access:
0 = do nothing
1 = issue hw_clear on the connected IRQs.

Bit 12 **TTA_IP1_CH4_IRQ**: Neighboring TIM, TOM, ATOM IRQs.
READ access:
Logical OR conjunction of IRQ signals *TIM[i+1]_CH4_IRQ*, *TOM[i+1]_CH8_IRQ*, *TOM[i+1]_CH9_IRQ*, and *ATOM[i+1]_CH4_IRQ*.
WRITE access:
0 = do nothing
1 = issue hw_clear on the connected IRQs.

Bit 13 **TTA_IP1_CH5_IRQ**: Neighboring TIM, TOM, ATOM IRQs.
READ access:
Logical OR conjunction of IRQ signals *TIM[i+1]_CH5_IRQ*, *TOM[i+1]_CH10_IRQ*, *TOM[i+1]_CH10_IRQ*, and *ATOM[i+1]_CH5_IRQ*.
WRITE access:
0 = do nothing
1 = issue hw_clear on the connected IRQs.

Bit 14 **TTA_IP1_CH6_IRQ**: Neighboring TIM, TOM, ATOM IRQs.
READ access:
Logical OR conjunction of IRQ signals *TIM[i+1]_CH6_IRQ*, *TOM[i+1]_CH12_IRQ*, *TOM[i+1]_CH13_IRQ*, and *ATOM[i+1]_CH6_IRQ*.
WRITE access:
0 = do nothing
1 = issue hw_clear on the connected IRQs.

Bit 15 **TTA_IP1_CH7_IRQ**: Neighboring TIM, TOM, ATOM IRQs.
READ access:
Logical OR conjunction of IRQ signals *TIM[i+1]_CH7_IRQ*, *TOM[i+1]_CH14_IRQ*, *TOM[i+1]_CH15_IRQ*, and *ATOM[i+1]_CH7_IRQ*.
WRITE access:
0 = do nothing
1 = issue hw_clear on the connected IRQs.

Bit 16 **MCS0_CH0_IRQ**: MCS0_CH0 IRQs.
READ access:
IRQ signal *MCS0_CH0_IRQ*.
WRITE access:
0 = do nothing
1 = issue hw_clear on the connected IRQs.

Bit 17 **MCS0_CH1_IRQ**: MCS0_CH1 IRQs.
READ access:
IRQ signal *MCS0_CH1_IRQ*.
WRITE access:
0 = do nothing
1 = issue hw_clear on the connected IRQs.

Bit 18 **MCS0_CH2_IRQ**: MCS0_CH2 IRQs.
READ access:
IRQ signal *MCS0_CH2_IRQ*.
WRITE access:
0 = do nothing
1 = issue hw_clear on the connected IRQs.

Bit 19 **MCS0_CH3_IRQ**: MCS0_CH3 IRQs.
READ access:
IRQ signal *MCS0_CH3_IRQ*.
WRITE access:
0 = do nothing
1 = issue hw_clear on the connected IRQs.

Bit 20 **MCS0_CH4_IRQ**: MCS0_CH4 IRQs.
READ access:
IRQ signal *MCS0_CH4_IRQ*.
WRITE access:
0 = do nothing
1 = issue hw_clear on the connected IRQs.

Bit 21 **MCS0_CH5_IRQ**: MCS0_CH5 IRQs.
READ access:
IRQ signal *MCS0_CH5_IRQ*.
WRITE access:
0 = do nothing
1 = issue hw_clear on the connected IRQs.

Bit 22 **MCS0_CH6_IRQ**: MCS0_CH6 IRQs.
READ access:
IRQ signal *MCS0_CH6_IRQ*.
WRITE access:
0 = do nothing
1 = issue hw_clear on the connected IRQs.

Bit 23 **MCS0_CH7_IRQ**: MCS0_CH7 IRQs.
READ access:
IRQ signal *MCS0_CH7_IRQ*.
WRITE access:
0 = do nothing
1 = issue hw_clear on the connected IRQs.

Bit 31:24 n/a

15.9.13 Register DSTA

Address Offset:	0x1A															Initial Value:								0x000000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	n/a								Reserved			CDTI	SISI	SASI	TISI	TASI	STA_S								STA_T							
Mode									R			RCw	RCw	RCw	RCw	RCw	R								R							
Initial Value									0x0			0	0	0	0	0	0								0							

Bit 7:0 **STA_T**: Status Trigger FSM.

Actual status of DPLL Trigger FSM. The description of the FSM states can be found in section 18.12.103.

Bit 15:8 **STA_S**: Status State FSM.
Actual status of DPLL State FSM. The description of the FSM states can be found in section 18.12.103.

Bit 16 **TASI**: Trigger Active Slope Interrupt.

READ access:

IRQ signal TASI of DPLL.
WRITE access:
0 = do nothing
1 = issue hw_clear on the connected IRQs.

Bit 17 **TISI**: Trigger Inactive Slope Interrupt.

READ access:
IRQ signal TISI of DPLL.
WRITE access:
0 = do nothing
1 = issue hw_clear on the connected IRQs.

Bit 18 **SASI**: State Active Slope Interrupt.

READ access:
IRQ signal SASI of DPLL.
WRITE access:
0 = do nothing
1 = issue hw_clear on the connected IRQs.

Bit 19 **SISI**: State Inactive Slope Interrupt.

READ access:
IRQ signal SISI of DPLL.
WRITE access:
0 = do nothing
1 = issue hw_clear on the connected IRQs.

Bit 20 **CDTI**: Calculation of trigger duration interrupt.

READ access:
IRQ signal CDTI of DPLL.
WRITE access:
0 = do nothing
1 = issue hw_clear on the connected IRQs.

Bit 23:21 **Reserved:** Read as zero, should be written as zero.
Bit 31:24 n/a

Note: This register is only implemented in MCS instance 0. In other MCS instances, a read access always returns 0 and a write access is always ignored.

15.9.14 Register DSTAX

Address Offset:	0x1B																Initial Value: 0x000000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	n/a								Reserved			CDTI	SISI	SASI	TISI	TASI	Reserved												INC_CNT2_FLAG	INC_CNT1_FLAG	STA_FLAG_S	STA_FLAG_T
Mode									R			RCw	RCw	RCw	RCw	RCw	R												RCw	RCw	RCw	RCw
Initial Value									0x0			0	0	0	0	0	0x0												0	0	0	0

Bit 0 **STA_FLAG_T**: DPLL status trigger flag.

DPLL status trigger flag as described in bit field definition **STA_FLAG_T** of register **DPLL_STA_FLAG** (section 18.12.109).

Bit 1 **STA_FLAG_S**: DPLL status state flag.
DPLL status state flag as described in bit field definition **STA_FLAG_S** of register **DPLL_STA_FLAG** (section 18.12.109).

Bit 2 **INC_CNT1_FLAG:** DPLL INC_CNT1 Flag.

DPLL status state flag as described in bit field definition **INC_CNT1_FLAG** of register **DPLL_STA_FLAG** (section 18.12.109).

Bit 3 **INC_CNT2_FLAG:** DPLL INC_CNT2 Flag.
DPLL status state flag as described in bit field definition **INC_CNT2_FLAG** of register **DPLL_STA_FLAG** (section 18.12.109).

Bit 15:4 **Reserved:** Read as zero, should be written as zero.
Bit 16 **TASI**: Trigger Active Slope Interrupt.

READ access:
IRQ signal TASI of DPLL.
WRITE access:
0 = do nothing

1 = issue hw_clear on the connected IRQs.

Bit 17 **TISI**: Trigger Inactive Slope Interrupt.

READ access:
IRQ signal TISI of DPLL.
WRITE access:
0 = do nothing
1 = issue hw_clear on the connected IRQs.

Bit 18 **SASI**: State Active Slope Interrupt.

READ access:
IRQ signal SASI of DPLL.
WRITE access:
0 = do nothing
1 = issue hw_clear on the connected IRQs.

Bit 19 **SISI**: State Inactive Slope Interrupt.

READ access:
IRQ signal SISI of DPLL.
WRITE access:
0 = do nothing
1 = issue hw_clear on the connected IRQs.

Bit 20 **CDTI**: Calculation of trigger duration interrupt.

READ access:
IRQ signal CDTI of DPLL.
WRITE access:
0 = do nothing
1 = issue hw_clear on the connected IRQs.

Bit 23:21 **Reserved:** Read as zero, should be written as zero.
Bit 31:24 n/a

Note: This register is only implemented in MCS instance 0. In other MCS instances, a read access always returns 0 and a write access is always ignored.

15.10MCS Configuration Register Overview

The MCS Configuration registers of the MCS module are accessible by the AEI bus interface. Some of these registers simply mirror MCS Internal registers to the AEI. Details can be found in the table below and in the individual register descriptions.

15.10.1 MCS Configuration Register Overview

Register Name	Description	Details in Section
MCS[i]_CH[x]_CTRL	MCSi channel x control register	15.11.1
MCS[i]_CH[x]_ACB	MCSi channel x ARU control Bit register	15.11.4
MCS[i]_CH[x]_MHB	MCSi channel x memory high byte register	15.11.5
MCS[i]_CH[x]_PC	MCSi channel x program counter register	15.11.2
MCS[i]_CH[x]_R[y] (y:0..7)	MCSi channel x general purpose register y	15.11.3
MCS[i]_CH[x]_IRQ_NOTIFY	MCSi channel x interrupt notification register	15.11.6
MCS[i]_CH[x]_IRQ_EN	MCSi channel x interrupt enable register	15.11.7
MCS[i]_CH[x]_IRQ_FORCINT	MCSi channel x force interrupt register	15.11.8
MCS[i]_CH[x]_IRQ_MODE	MCSi channel x IRQ mode configuration register	15.11.9
MCS[i]_CH[x]_EIRQ_EN	MCSi channel x error interrupt enable register	15.11.10
MCS[i]_CTRL_STAT	MCSi control and status register	15.11.11
MCS[i]_REG_PROT	MCSi write protection register	15.11.12
MCS[i]_CTRG	MCSi clear trigger control register	15.11.13
MCS[i]_STRG	MCSi set trigger control register	15.11.14
MCS[i]_RESET	MCSi reset register	15.11.15
MCS[i]_ERR	MCSi error register	15.11.18
MCS[i]_CAT	MCSi cancel ARU transfer instruction	15.11.16
MCS[i]_CWT	MCSi cancel WURM instruction	15.11.17

15.11MCS Configuration Register Description

15.11.1 Register MCS[i]_CH[x]_CTRL

Address Offset:	see Appendix B																Initial Value:								0x00000000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved													SP_CNT			Reserved					SAT	CWT	CAT	N	V	Z	CY	Reserved	ERR	IRQ	EN
Mode	R													R			R					R	R	R	R	R	R	R	R	R	R	RPw
Initial Value	0x0000													000			0x00					0	0	0	0	0	0	0	0	0	0	0

Bit 0 **EN**: Enable MCS-channel
0 = Disable current MCS-channel.
1 = Enable current MCS-channel.
Note: Enabling or disabling of an MCS-channel is synchronized to the ending of an instruction and thus it may take several clock cycles, e.g. active memory transfers or pending WURM transfers have to be finished before disabling the MCS-channel. The internal state of a channel can be obtained by reading the bit EN.

Note: To disable an MCS channel reliably the EN bit should be cleared followed by setting the CAT and CWT bit in order to cancel any pending WURM or ARU instructions.

Note: The EN bit is write protected during RAM reset phase.

Bit 1 **IRQ**: Interrupt state.
0 = No interrupt pending in MCS-channel x.
1 = Interrupt is pending in MCS-channel x.
Note: This bit is read only and it mirrors the internal IRQ state.

Bit 2 **ERR**: Error state.
0 = No error signal pending in MCS-channel x.
1 = Error signal is pending in MCS-channel x.
Note: This bit is read only and it mirrors the internal error state.

Bit 3 **Reserved:** Read as zero, should be written as zero.

Bit 4 **CY**: Carry bit state.
Note: This bit is read only and it mirrors the internal carry flag CY.

Bit 5 **Z**: Zero bit state.
Note: This bit is read only and it mirrors the internal zero flag Z.

Bit 6 **V**: Overflow bit state.
Note: This bit is read only and it mirrors the internal carry flag V.

Bit 7 **N**: Negative bit state.

Note: This bit is read only and it mirrors the internal zero flag N.

Bit 8 **CAT:** Cancel ARU transfer state.

Note: This bit is read only and it mirrors the internal cancel ARU transfer status flag CAT.

Bit 9 **CWT**: Cancel WURM instruction state.

Note: This bit is read only and it mirrors the internal cancel WURM instruction status flag CWT.

Bit 10 **SAT**: Successful ARU transfer bit.

0 = ARU data transfer failed.

1 = ARU data transfer finished successfully.

Note: This bit is read only and it mirrors the internal state of the ARU transfer status flag SAT.

Bit 15:11 **Reserved:** Read as zero, should be written as zero.

Bit 18:16 **SP_CNT**: Stack pointer counter value.

Actual stack depth of channel. The bit field is incremented on behalf of a CALL or PUSH instruction and decremented on behalf of a RET or POP instruction. The MCS channel STK_ERR_IRQ is raised, when an overflow or underflow is detected on this bit field.

Bit 31:19 **Reserved:** Read as zero, should be written as zero.

15.11.2 Register MCS[i]_CH[x]_PC

Address Offset:	**see Appendix B**																**Initial Value:**								**0x00000000 + 4*x**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																PC															
Mode	R																RPw															
Initial Value	0x0000																0x0000 +4*x															

Bit 15:0 **PC**: Current Program Counter.

Note: The program counter is only writable if the corresponding MCS-channel is disabled. The bits 0 and 1 are always written as zeros.

Note: The actual width of the program counter depends on the MCS configuration. The actual width is RAW+USR+2 bits meaning that only the bits 0 to RAW+USR+1 are available and the other bits (RAW+USR+2 to 31) are reserved.

Bit 31:16 **Reserved:** Read as zero, should be written as zero.

15.11.3 Register MCS[i]_CH[x]_R[y] (y:0...7)

Address Offset:	see Appendix B																Initial Value:								0x00000000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								DATA																							
Mode	R								RW																							
Initial Value	0x00								0x000000																							

Bit 23:0 **DATA**: Data of general purpose register R[y].
Note: This register is the same as described in 15.9.1.
Note: For the register **MCS[i]_CH[x]_R6 15.9.1** an additional write protection during an active ARDI or NARDI instruction is applied.

Bit 31:24 **Reserved:** Read as zero, should be written as zero.

15.11.4 Register MCS[i]_CH[x]_ACB

Address Offset:	see Appendix B																Initial Value:								0x00000000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																											ACB4	ACB3	ACB2	ACB1	ACB0
Mode	R																											R	R	R	R	R
Initial Value	0x0000000																											0	0	0	0	0

Bit 0 **ACB0**: ARU Control bit 0.
Note: This bit is read only and it mirrors the internal state.

Bit 1 **ACB1**: ARU Control bit 1.
Note: This bit is read only and it mirrors the internal state.

Bit 2 **ACB2**: ARU Control bit 2.
Note: This bit is read only and it mirrors the internal state.

Bit 3 **ACB3**: ARU Control bit 3.

Note: This bit is read only and it mirrors the internal state.

Bit 4 **ACB4**: ARU Control bit 4.

Note: This bit is read only and it mirrors the internal state.

Bit 31:5 **Reserved:** Read as zero, should be written as zero.

15.11.5 Register MCS[i]_CH[x]_MHB

Address Offset:	see Appendix B																Initial Value:								0x00000000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																								DATA							
Mode	R																								R							
Initial Value	0x000000																								0x00							

Bit 7:0 **DATA**: Data of memory high bit register MHB.

Bit 31:8 **Reserved:** Read as zero, should be written as zero.

15.11.6 Register MCS[i]_CH[x]_IRQ_NOTIFY

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																													ERR_IRQ	STK_ERR_IRQ	MCS_IRQ
Mode	R																													RCw	RCw	RCw
Initial Value	0x00000000																													0	0	0

Bit 0 **MCS_IRQ:** Interrupt request by MCS-channel x.

0 = No IRQ released

1 = IRQ released by MCS-channel

Note: This bit will be cleared on a CPU write access with a value '1'. A read access leaves the bit unchanged.

Note: By writing a '1' to this register, the IRQ flag in the MCS channel status register **STA** is cleared.

Bit 1 **STK_ERR_IRQ:** Stack counter overflow/underflow of channel x.
0 = No IRQ released
1 = A stack counter overflow or underflow occurred
Note: This bit will be cleared on a CPU write access with a value '1'. A read access leaves the bit unchanged.

Bit 2 **ERR_IRQ:** MCS channel x ERR interrupt.
0 = No IRQ released
1 = MCS-channel ERR IRQ released.
Note: If the ERR bit of register **STA** is triggered the ERR_IRQ will also be set.
Note: This bit will be cleared on a CPU write access with a value '1'. A read access leaves the bit unchanged.

Bit 31:3 **Reserved:** Read as zero, should be written as zero.

15.11.7 Register MCS[i]_CH[x]_IRQ_EN

Address Offset:	**see Appendix B**																**Initial Value:**								**0x00000000**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																													ERR_IRQ_EN	STK_ERR_IRQ_E	MCS_IRQ_EN
Mode	R																													RW	RW	RW
Initial Value	0x0000 0000																													0	0	0

Bit 0 **MCS_IRQ_EN:** MCS channel x MCS_IRQ interrupt enable
0 = Disable interrupt
1 = Enable interrupt

Bit 1 **STK_ERR_IRQ_EN:** MCS channel x STK_ERR_IRQ interrupt enable
0 = Disable interrupt
1 = Enable interrupt

Bit 2 **ERR_IRQ_EN:** MCS channel x ERR_IRQ interrupt enable
0 = Disable interrupt
1 = Enable interrupt

Bit 31:3 **Reserved:** Read as zero, should be written as zero.

15.11.8 Register MCS[i]_CH[x]_IRQ_FORCINT

Address Offset:	see Appendix B																Initial Value: 0x00000000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																													TRG_ERR_IRQ	TRG_STK_ERR_I	TRG_MCS_IRQ
Mode	R																													RAw	RAw	RAw
Initial Value	0x0000 0000																													0	0	0

Bit 0 **TRG_MCS_IRQ:** Trigger **IRQ** bit in **MCS_CH_[x]_IRQ_NOTIFY** register by software
0 = No interrupt triggering
1 = Assert corresponding field in **MCS[i]_CH[x]_IRQ_NOTIFY** register
Note: This bit is cleared automatically after write.
Note: This bit is write protected by bit **RF_PROT** of register **GTM_CTRL**

Bit 1 **TRG_STK_ERR_IRQ:** Trigger **IRQ** bit in **MCS_CH_[x]_IRQ_NOTIFY** register by software
0 = No interrupt triggering
1 = Assert corresponding field in **MCS[i]_CH[x]_IRQ_NOTIFY** register
Note: This bit is cleared automatically after write.
Note: This bit is write protected by bit **RF_PROT** of register **GTM_CTRL**

Bit 2 **TRG_ERR_IRQ:** Trigger **IRQ** bit in **MCS_CH_[x]_IRQ_NOTIFY** register by software
0 = No interrupt triggering
1 = Assert corresponding field in **MCS[i]_CH[x]_IRQ_NOTIFY** register
Note: This bit is cleared automatically after write.
Note: This bit is write protected by bit **RF_PROT** of register **GTM_CTRL**

Bit 31:3 **Reserved:** Read as zero, should be written as zero.

15.11.9 Register MCS[i]_CH[x]_IRQ_MODE

Address Offset:	see Appendix B																Initial Value:								0x0000_000X							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																														IRQ_MODE	
Mode	R																														RW	
Initial Value	0x0000 0000																														XX	

Bit 1:0 **IRQ_MODE**: IRQ mode selection
0b00 = Level mode
0b01 = Pulse mode
0b10 = Pulse-Notify mode
0b11 = Single-Pulse mode
Note: The interrupt modes are described in section 2.5.

Bit 31:2 **Reserved**
Note: Read as zero, should be written as zero

15.11.10 Register MCS[i]_CH[x]_EIRQ_EN

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																													ERR_EIRQ_EN	STK_ERR_EIRQ_	MCS_EIRQ_EN0
Mode	R																													RW	RW	RW
Initial Value	0x0000 0000																													0	0	0

Bit 0 **MCS_EIRQ_EN:** MCS channel x MCS_EIRQ error interrupt enable
0 = Disable error interrupt
1 = Enable error interrupt

Bit 1 **STK_ERR_EIRQ_EN:** MCS channel x STK_ERR_IRQ error interrupt enable
0 = Disable error interrupt
1 = Enable error interrupt

Bit 2 **ERR_EIRQ_EN:** MCS channel x ERR_EIRQ error interrupt enable

0 = Disable error interrupt
1 = Enable error interrupt

Bit 31:3 **Reserved:** Read as zero, should be written as zero.

15.11.11 Register MCS[i]_CTRL_STAT

Address Offset:	**see Appendix B**																**Initial Value: 0x000X_0000**															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved					HLT_AEIM_ERR	EN_XOREG	EN_TIM_FOUT	Reserved	ERR_SRC_ID			Reserved		HLT_SP_OFL	RAM_RST	Reserved				SCD_CH				Reserved						SCD_MODE	
Mode	R					RW	RW	RW	R	R			R		RW	RPw	R				RW				R						RW	
Initial Value	0x00					0	0	0	0	000			00		0	X	0x0				0x0				0x00						00	

Bit 1:0 **SCD_MODE:** Select MCS scheduling mode
0b00 = Accelerated Scheduling.
0b01 = Round Robin Scheduling.
0b10 = Single Priority Scheduling.
0b11 = Multiple Priority Scheduling.

Bit 7:2 **Reserved:** Read as zero, should be written as zero.

Bit 11:8 **SCD_CH:** Channel selection for scheduling algorithm.
MCS-channel identifier used by several scheduling modes.
Note: The actual width of the bit field **SCD_CH** is calculated as ⌈log2(T+1)⌉.
Unused MSBs are reserved and read as zero.

Bit 15:12 **Reserved:** Read as zero, should be written as zero.

Bit 16 **RAM_RST:** RAM reset bit.
0 = READ: no RAM reset is active / WRITE : do nothing.
1 = READ: MCS currently resets RAM content / WRITE : trigger RAM reset.

Note: The RAM reset initializes the memory content with zeros. RAM access and enabling of MCS channels is disabled during active RAM reset.

Note: This bit is only writable if the bit **RF_PROT** in register **GTM_CTRL** is cleared and all MCS-channels are disabled.

Note: The actual reset value of this bit depends on the silicon vendor configuration. The reset value is 1, if the RAM reset is performed together with the sub module reset, otherwise the reset value is 0. If the reset value is 1, the reset value is changed to 0 by hardware, when the RAM reset finished.

Bit 17 **HLT_SP_OFL:** Halt on stack pointer overflow.
0 = No halt on MCS-channel stack pointer counter over/underflow.
1 = MCS-channel is disabled if a stack pointer counter over/underflow occurs.

Bit 19:18 **Reserved:** Read as zero, should be written as zero.
Bit 22:20 **ERR_SRC_ID:** Error source identifier.
0b000 = No HW generated Error occurred.
0b001 = Detected ECC error.
0b010 = Detected memory overflow.
0b011 = Detected invalid opcode.
0b100 = Divide by zero.
0b101 = Invalid register write access to GPR from write protected channel.
0b110 = Invalid memory write access to protected memory region.
0b111 = Invalid AEI bus master access.

Note: This register is updated once, if an error was detected by the MCS. The register is set to its initial value 000 after each write access to an existing ERR bit in the register **MCS[i]_ERR**. If multiple errors occur, **ERR_SRC_ID** is holding the first type of error which has occurred.

Bit 23 **Reserved:** Read as zero, should be written as zero.
Bit 24 **EN_TIM_FOUT:** Enable routing of *TIM[i]_CH[x]_F_OUT* signal.
0 = Read access to register **CTRG/MCS[i]_CTRG** provides state of the internal trigger registers.
1 = Read access to register **CTRG/MCS[i]_CTRG** provides state of the external signal *TIM[i]_CH[x]_F_OUT*.

Bit 25 **EN_XOREG:** Enable extended register set.
0 = Extended operation register sets XOREG, BAREG, and WXREG are disabled.
1 = Extended operation register sets XOREG, BAREG, and WXREG are enabled.

NOTE: If the extended operation register sets are disabled, the MCS instructions can only use the subset OREG of the register set as arguments in the instructions. In this case the upper address bits in the instructions are always read as zeros, which leads to unexpected results of the MCS program if arguments A or B refer a register that is not part or OREG.

Bit 26 **HLT_AEIM_ERR:** Halt on AEI bus master error.
0 = Ignore invalid AEI bus master access.
1 = Halt MCS-channel on invalid AEI bus master access.

NOTE: If the register **HLT_AEIM_ERR** is set and an MCS channel x is executing an invalid bus master access, the MCS channel x is halted, the **ERR** bit of its register **STA** is set and the bit field **ERR_SRC_ID** of this register is updated.

NOTE: If the bus master is accessing a slave that does not insert wait cycles (e.g. register access) it takes two additional clock cycles until the MCS channel is halted. Within that time spawn the MCS channel can continue with its program execution, depending on the selected scheduling mode.

NOTE: The registers **AEIM_XPT_STA** and **AEIM_XPT_ADDR** of the GTM sub module CCM are always updated on the first invalid AEI bus master access, independently of the state of **HLT_AEIM_ERR**.

Bit 31:27 **Reserved:** Read as zero, should be written as zero.

15.11.12 Register MCS[i]_REG_PROT

Address Offset:	see Appendix B																Initial Value:								0x00000000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																WPROT7		WPROT6		WPROT5		WPROT4		WPROT3		WPROT2		WPROT1		WPROT0	
Mode	R																RPw		RPw		RPw		RPw		RPw		RPw		RPw		RPw	
Initial Value	0x0000																0		0		0		0		0		0		0		0	

Bit 1:0 **WPROT0**: Register Write Protection of MCS-channel 0.
0b00 = no register write protection activated
0b01 = predecessor MCS channel cannot write to its RS[y] registers
0b10 = current MCS channel cannot write to its R[y] registers
0b11 = reserved

Bit 3:2 **WPROT1**: Register Write Protection of MCS-channel 1.

0b00 = no register write protection activated
0b01 = predecessor MCS channel cannot write to its RS[y] registers
0b10 = current MCS channel cannot write to its R[y] registers
0b11 = reserved

Bit 5:4 **WPROT2**: Register Write Protection of MCS-channel 2.
0b00 = no register write protection activated
0b01 = predecessor MCS channel cannot write to its RS[y] registers
0b10 = current MCS channel cannot write to its R[y] registers
0b11 = reserved

Bit 7:6 **WPROT3**: Register Write Protection of MCS-channel 3.
0b00 = no register write protection activated
0b01 = predecessor MCS channel cannot write to its RS[y] registers
0b10 = current MCS channel cannot write to its R[y] registers
0b11 = reserved

Bit 9:8 **WPROT4**: Register Write Protection of MCS-channel 4.
0b00 = no register write protection activated
0b01 = predecessor MCS channel cannot write to its RS[y] registers
0b10 = current MCS channel cannot write to its R[y] registers
0b11 = reserved

Bit 11:10 **WPROT5**: Register Write Protection of MCS-channel 5.
0b00 = no register write protection activated
0b01 = predecessor MCS channel cannot write to its RS[y] registers
0b10 = current MCS channel cannot write to its R[y] registers
0b11 = reserved

Bit 13:12 **WPROT6**: Register Write Protection of MCS-channel 6.
0b00 = no register write protection activated
0b01 = predecessor MCS channel cannot write to its RS[y] registers
0b10 = current MCS channel cannot write to its R[y] registers
0b11 = reserved

Bit 15:14 **WPROT7**: Register Write Protection of MCS-channel 7.
0b00 = no register write protection activated
0b01 = predecessor MCS channel cannot write to its RS[y] registers
0b10 = current MCS channel cannot write to its R[y] registers
0b11 = reserved

Bit 31:16 **Reserved:** Read as zero, should be written as zero.
Note: Only the first T bit fields of this register (bit 0 to 2*T-1) are functionally implemented. The other bits (bit 2*T to 31) are reserved bit fields.

Note: The predecessor channel of MCS channel 0 is MCS channel T-1.

Note: If an MCS channel x is writing to a general purpose register that is write protected by register **MCS[i]_REG_PROT** the **ERR** bit of the register **STA** is set, the MCS channel x is halted and the **ERR_SRC_ID** bit field of register **MCS[i]_CTRL_STAT** is updated.

Note: This register is only writable if the bit **RF_PROT** in register **GTM_CTRL** is cleared.

15.11.13 Register MCS[i]_CTRG

Address Offset:	**see Appendix B**																**Initial Value:**	**0x0000_0000**														
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								TRG23	TRG22	TRG21	TRG20	TRG19	TRG18	TRG17	TRG16	TRG15	TRG14	TRG13	TRG12	TRG11	TRG10	TRG9	TRG8	TRG7	TRG6	TRG5	TRG4	TRG3	TRG2	TRG1	TRG0
Mode	R								RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Initial Value	0x0000								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit 0 **TRG0:** trigger bit 0.
READ access:
state of current trigger bit **TRG0** if **EN_TIM_FOUT** = 0
state of input signal **TIM[i]_CH0_F_OUT** if **EN_TIM_FOUT** = 1
WRITE access:
0 = do nothing
1 = clear trigger bit **TRG0**

Bit 1 **TRG1:** trigger bit 1.
READ access:
state of current trigger bit **TRG1** if **EN_TIM_FOUT** = 0
state of input signal **TIM[i]_CH1_F_OUT** if **EN_TIM_FOUT** = 1
WRITE access:
0 = do nothing
1 = clear trigger bit **TRG1**

Bit 2 **TRG2:** trigger bit 2.
READ access:
state of current trigger bit **TRG2** if **EN_TIM_FOUT** = 0
state of input signal **TIM[i]_CH2_F_OUT** if **EN_TIM_FOUT** = 1
WRITE access:
0 = do nothing

1 = clear trigger bit **TRG2**

Bit 3 **TRG3:** trigger bit 3.
READ access:
state of current trigger bit **TRG3** if **EN_TIM_FOUT** = 0
state of input signal **TIM[i]_CH3_F_OUT** if **EN_TIM_FOUT** = 1
WRITE access:
0 = do nothing
1 = clear trigger bit **TRG3**

Bit 4 **TRG4:** trigger bit 4.
READ access:
state of current trigger bit **TRG4** if **EN_TIM_FOUT** = 0
state of input signal **TIM[i]_CH4_F_OUT** if **EN_TIM_FOUT** = 1
WRITE access:
0 = do nothing
1 = clear trigger bit **TRG4**

Bit 5 **TRG5:** trigger bit 5.
READ access:
state of current trigger bit **TRG5** if **EN_TIM_FOUT** = 0
state of input signal **TIM[i]_CH5_F_OUT** if **EN_TIM_FOUT** = 1
WRITE access:
0 = do nothing
1 = clear trigger bit **TRG5**

Bit 6 **TRG6:** trigger bit 6.
READ access:
state of current trigger bit **TRG6** if **EN_TIM_FOUT** = 0
state of input signal **TIM[i]_CH6_F_OUT** if **EN_TIM_FOUT** = 1
WRITE access:
0 = do nothing
1 = clear trigger bit **TRG6**

Bit 7 **TRG7:** trigger bit 7.
READ access:
state of current trigger bit **TRG7** if **EN_TIM_FOUT** = 0
state of input signal **TIM[i]_CH7_F_OUT** if **EN_TIM_FOUT** = 1
WRITE access:
0 = do nothing
1 = clear trigger bit **TRG7**

Bit 8 **TRG8:** trigger bit 8.
READ access:
state of current trigger bit **TRG8** if **EN_TIM_FOUT** = 0
state of input signal **TIM[i+1]_CH0_F_OUT** if **EN_TIM_FOUT** = 1
WRITE access:

0 = do nothing
1 = clear trigger bit **TRG8**

Bit 9 **TRG9:** trigger bit 9.
READ access:
state of current trigger bit **TRG9** if **EN_TIM_FOUT** = 0
state of input signal **TIM[i+1]_CH1_F_OUT** if **EN_TIM_FOUT** = 1
WRITE access:
0 = do nothing
1 = clear trigger bit **TRG9**

Bit 10 **TRG10:** trigger bit 10.
READ access:
state of current trigger bit **TRG10** if **EN_TIM_FOUT** = 0
state of input signal **TIM[i+1]_CH2_F_OUT** if **EN_TIM_FOUT** = 1
WRITE access:
0 = do nothing
1 = clear trigger bit **TRG10**

Bit 11 **TRG11:** trigger bit 11.
READ access:
state of current trigger bit **TRG11** if **EN_TIM_FOUT** = 0
state of input signal **TIM[i+1]_CH3_F_OUT** if **EN_TIM_FOUT** = 1
WRITE access:
0 = do nothing
1 = clear trigger bit **TRG11**

Bit 12 **TRG12:** trigger bit 12.
READ access:
state of current trigger bit **TRG12** if **EN_TIM_FOUT** = 0
state of input signal **TIM[i+1]_CH4_F_OUT** if **EN_TIM_FOUT** = 1
WRITE access:
0 = do nothing
1 = clear trigger bit **TRG12**

Bit 13 **TRG13:** trigger bit 13.
READ access:
state of current trigger bit **TRG13** if **EN_TIM_FOUT** = 0
state of input signal **TIM[i+1]_CH5_F_OUT** if **EN_TIM_FOUT** = 1
WRITE access:
0 = do nothing
1 = clear trigger bit **TRG13**

Bit 14 **TRG14:** trigger bit 14.
READ access:
state of current trigger bit **TRG14** if **EN_TIM_FOUT** = 0
state of input signal **TIM[i+1]_CH6_F_OUT** if **EN_TIM_FOUT** = 1

WRITE access:
0 = do nothing
1 = clear trigger bit **TRG14**

Bit 15 **TRG15:** trigger bit 15.
READ access:
state of current trigger bit **TRG15** if **EN_TIM_FOUT** = 0
state of input signal **TIM[i+1]_CH7_F_OUT** if **EN_TIM_FOUT** = 1
WRITE access:
0 = do nothing
1 = clear trigger bit **TRG15**

Bit 16 **TRG16:** trigger bit 16.
0 = READ: trigger bit is cleared / WRITE : do nothing
1 = READ: trigger bit is set / WRITE : clear trigger bit

Bit 17 **TRG17:** trigger bit 17.
0 = READ: trigger bit is cleared / WRITE : do nothing
1 = READ: trigger bit is set / WRITE : clear trigger bit

Bit 18 **TRG18:** trigger bit 18.
0 = READ: trigger bit is cleared / WRITE : do nothing
1 = READ: trigger bit is set / WRITE : clear trigger bit

Bit 19 **TRG19:** trigger bit 19.
0 = READ: trigger bit is cleared / WRITE : do nothing
1 = READ: trigger bit is set / WRITE : clear trigger bit

Bit 20 **TRG20:** trigger bit 20.
0 = READ: trigger bit is cleared / WRITE : do nothing
1 = READ: trigger bit is set / WRITE : clear trigger bit

Bit 21 **TRG21:** trigger bit 21.
0 = READ: trigger bit is cleared / WRITE : do nothing
1 = READ: trigger bit is set / WRITE : clear trigger bit

Bit 22 **TRG22:** trigger bit 22.
0 = READ: trigger bit is cleared / WRITE : do nothing
1 = READ: trigger bit is set / WRITE : clear trigger bit

Bit 23 **TRG23:** trigger bit 23.
0 = READ: trigger bit is cleared / WRITE : do nothing
1 = READ: trigger bit is set / WRITE : clear trigger bit

Note: The trigger bits **TRGx** are accessible by all MCS channels as well as the CPU. Setting a trigger bit can be performed with the **STRG** register, in the case of an MCS-channel or the **MCS[i]_STRG** register in the case of the CPU. Clearing a trigger bit can be performed with the **CTRG** register, in the case of an MCS-channel or the **MCS[i]_CTRG** register in the case of the CPU. Trigger bits can be used for signalizing specific events to MCS-channels or the CPU. An MCS-channel suspended with a WURM instruction can be resumed by setting the appropriate trigger bit.

Note: Besides setting the trigger bits with register **STRG/MCS[i]_STRG**, the k-th trigger bit **TRGk** (with k < 16) can also be set by the external capture event that is enabled by the k-th bit of register **CCM[i]_EXT_CAP_EN**. If bit k bit is disabled, the k-th trigger bit **TRGk** can only be set by MCS or CPU.

Note: In the scheduling modes Accelerated Scheduling and Round Robin Scheduling, a write access to **MCS[i]_CTRG** may take up to T + 1 clock cycles, since the write access is scheduled to the next CPU time slot determined by the MCS scheduler. In the modes Single Prioritization Scheduling and Multiple Prioritization Scheduling, no upper limit access time for a write access to **MCS[i]_CTRG** can be guaranteed. The High Prioritized tasks have to be disabled in order to guarantee fast write access to **MCS[i]_CTRG.**

Bit 31:24 **Reserved:** Read as zero, should be written as zero.
Note: The result of a read access to this register differs in dependency of the bit field **EN_TIM_FOUT** of register **MCS[i]_CTRL_STAT**.

15.11.14 Register MCS[i]_STRG

Address Offset:	**see Appendix B**																**Initial Value: 0x0000_0000**															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								TRG23	TRG22	TRG21	TRG20	TRG19	TRG18	TRG17	TRG16	TRG15	TRG14	TRG13	TRG12	TRG11	TRG10	TRG9	TRG8	TRG7	TRG6	TRG5	TRG4	TRG3	TRG2	TRG1	TRG0
Mode	R								RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Initial Value	0x0000								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit 0 **TRG0:** trigger bit 0.
0 = READ: trigger bit is cleared / WRITE : do nothing
1 = READ: trigger bit is set / WRITE : set trigger bit

Bit 1 **TRG1:** trigger bit 1.
0 = READ: trigger bit is cleared / WRITE : do nothing
1 = READ: trigger bit is set / WRITE : set trigger bit

Bit 2 **TRG2:** trigger bit 2.
0 = READ: trigger bit is cleared / WRITE : do nothing
1 = READ: trigger bit is set / WRITE : set trigger bit

Bit 3 **TRG3:** trigger bit 3.
0 = READ: trigger bit is cleared / WRITE : do nothing

1 = READ: trigger bit is set / WRITE : set trigger bit

Bit 4 **TRG4:** trigger bit 4.
0 = READ: trigger bit is cleared / WRITE : do nothing
1 = READ: trigger bit is set / WRITE : set trigger bit

Bit 5 **TRG5:** trigger bit 5.
0 = READ: trigger bit is cleared / WRITE : do nothing
1 = READ: trigger bit is set / WRITE : set trigger bit

Bit 6 **TRG6:** trigger bit 6.
0 = READ: trigger bit is cleared / WRITE : do nothing
1 = READ: trigger bit is set / WRITE : set trigger bit

Bit 7 **TRG7:** trigger bit 7.
0 = READ: trigger bit is cleared / WRITE : do nothing
1 = READ: trigger bit is set / WRITE : set trigger bit

Bit 8 **TRG8:** trigger bit 8.
0 = READ: trigger bit is cleared / WRITE : do nothing
1 = READ: trigger bit is set / WRITE : set trigger bit

Bit 9 **TRG9:** trigger bit 9.
0 = READ: trigger bit is cleared / WRITE : do nothing
1 = READ: trigger bit is set / WRITE : set trigger bit

Bit 10 **TRG10:** trigger bit 10.
0 = READ: trigger bit is cleared / WRITE : do nothing
1 = READ: trigger bit is set / WRITE : set trigger bit

Bit 11 **TRG11:** trigger bit 11.
0 = READ: trigger bit is cleared / WRITE : do nothing
1 = READ: trigger bit is set / WRITE : set trigger bit

Bit 12 **TRG12:** trigger bit 12.
0 = READ: trigger bit is cleared / WRITE : do nothing
1 = READ: trigger bit is set / WRITE : set trigger bit

Bit 13 **TRG13:** trigger bit 13.
0 = READ: trigger bit is cleared / WRITE : do nothing
1 = READ: trigger bit is set / WRITE : set trigger bit

Bit 14 **TRG14:** trigger bit 14.
0 = READ: trigger bit is cleared / WRITE : do nothing
1 = READ: trigger bit is set / WRITE : set trigger bit

Bit 15 **TRG15:** trigger bit 15.
0 = READ: trigger bit is cleared / WRITE : do nothing
1 = READ: trigger bit is set / WRITE : set trigger bit

Bit 16 **TRG16:** trigger bit 16.
0 = READ: trigger bit is cleared / WRITE : do nothing
1 = READ: trigger bit is set / WRITE : set trigger bit

Bit 17 **TRG17:** trigger bit 17.
0 = READ: trigger bit is cleared / WRITE : do nothing
1 = READ: trigger bit is set / WRITE : set trigger bit

Bit 18 **TRG18:** trigger bit 18.
0 = READ: trigger bit is cleared / WRITE : do nothing
1 = READ: trigger bit is set / WRITE : set trigger bit

Bit 19 **TRG19:** trigger bit 19.

0 = READ: trigger bit is cleared / WRITE : do nothing
1 = READ: trigger bit is set / WRITE : set trigger bit

Bit 20 **TRG20:** trigger bit 20.
0 = READ: trigger bit is cleared / WRITE : do nothing
1 = READ: trigger bit is set / WRITE : set trigger bit

Bit 21 **TRG21:** trigger bit 21.
0 = READ: trigger bit is cleared / WRITE : do nothing
1 = READ: trigger bit is set / WRITE : set trigger bit

Bit 22 **TRG22:** trigger bit 22.
0 = READ: trigger bit is cleared / WRITE : do nothing
1 = READ: trigger bit is set / WRITE : set trigger bit

Bit 23 **TRG23:** trigger bit 23.
0 = READ: trigger bit is cleared / WRITE : do nothing
1 = READ: trigger bit is set / WRITE : set trigger bit

Note: The trigger bits **TRGx** are accessible by all MCS channels as well as the CPU. Setting a trigger bit can be performed with the **STRG** register, in the case of an MCS-channel or the **MCS[i]_STRG** register in the case of the CPU. Clearing a trigger bit can be performed with the **CTRG** register, in the case of an MCS-channel or the **MCS[i]_CTRG** register in the case of the CPU. Trigger bits can be used for signalizing specific events to MCS-channels or the CPU. An MCS-channel suspended with a WURM instruction can be resumed by setting the appropriate trigger bit.

Note: Besides setting the trigger bits with register **STRG/MCS[i]_STRG**, the k-th trigger bit **TRGk** (with k < 16) can also be set by the external capture event that is enabled by the k-th bit of register **CCM[i]_EXT_CAP_EN**. If bit k bit is disabled, the k-th trigger bit **TRGk** can only be set by MCS or CPU.

Note: In the scheduling modes Accelerated Scheduling and Round Robin Scheduling, a write access to **MCS[i]_STRG** may take up to T + 1 clock cycles, since the write access is scheduled to the next CPU time slot determined by the MCS scheduler. In the modes Single Prioritization Scheduling and Multiple Prioritization Scheduling, no upper limit access time for a write access to **MCS[i]_STRG** can be guaranteed. The High Prioritized tasks have to be disabled in order to guarantee fast write access to **MCS[i]_STRG.**

Bit 31:24 **Reserved:** Read as zero, should be written as zero.

15.11.15 Register MCS[i]_RESET

Address Offset:	**see Appendix B**																**Initial Value:**								**0x00000000**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																								RST7	RST6	RST5	RST4	RST3	RST2	RST1	RST0
Mode	R																								RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw
Initial Value	0x0000 00																								0	0	0	0	0	0	0	0

Bit 0 **RST0:** Software reset of channel 0
0 = No action
1 = Reset channel

Bit 1 **RST1:** Software reset of channel 1
0 = No action
1 = Reset channel

Bit 2 **RST2:** Software reset of channel 2
0 = No action
1 = Reset channel

Bit 3 **RST3:** Software reset of channel 3
0 = No action
1 = Reset channel

Bit 4 **RST4:** Software reset of channel 4
0 = No action
1 = Reset channel

Bit 5 **RST5:** Software reset of channel 5
0 = No action
1 = Reset channel

Bit 6 **RST6:** Software reset of channel 6
0 = No action
1 = Reset channel

Bit 7 **RST7:** Software reset of channel 7
0 = No action
1 = Reset channel

Note: The **RSTx** (x = 0 ...T-1) bits is cleared automatically after write access of CPU. All channel related registers of channel x are set to their reset values and channel operation is stopped immediately.

Note: Channel related registers of channel x are all registers **MCS[i]_CH[x]_***, all MCS internal registers accessible by the corresponding channel, with exception of the common trigger register (accessed by **MCS[i]_CTRG/MCS[i]_STRG**).

Bit 31:8 **Reserved:** Read as zero, should be written as zero.

Note: Only the first T bits of this register (bit 0 to T-1) are functionally implemented. The other bits (bit T to 31) are reserved bits.

15.11.16 Register MCS[i]_CAT

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																								CAT7	CAT6	CAT5	CAT4	CAT3	CAT2	CAT1	CAT0
Mode	R																								RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw
Initial Value	0x0000 00																								0	0	0	0	0	0	0	0

Bit 0 **CAT0:** Cancel ARU transfer for channel 0.
0 = Do nothing.
1 = Cancel any pending blocking ARU read or write transfer.

Bit 1 **CAT1:** Cancel ARU transfer for channel 1.
0 = Do nothing.
1 = Cancel any pending blocking ARU read or write transfer.

Bit 2 **CAT2:** Cancel ARU transfer for channel 2.
0 = Do nothing.
1 = Cancel any pending blocking ARU read or write transfer.

Bit 3 **CAT3:** Cancel ARU transfer for channel 3.
0 = Do nothing.
1 = Cancel any pending blocking ARU read or write transfer.

Bit 4 **CAT4:** Cancel ARU transfer for channel 4.
0 = Do nothing.
1 = Cancel any pending blocking ARU read or write transfer.

Bit 5 **CAT5:** Cancel ARU transfer for channel 5.
0 = Do nothing.
1 = Cancel any pending blocking ARU read or write transfer.

Bit 6 **CAT6:** Cancel ARU transfer for channel 6.
0 = Do nothing.
1 = Cancel any pending blocking ARU read or write transfer.

Bit 7 **CAT7:** Cancel ARU transfer for channel 7.
0 = Do nothing.
1 = Cancel any pending blocking ARU read or write transfer.

Note: The CATx (x = 0 ...T-1) bit inside the **STA** register of the corresponding MCS-channel is set and any pending blocking ARU read or write request is canceled. The MCS-channel resumes with the instruction after the blocking ARU transfer instruction.

Note: The CATx (x = 0 ...T-1) bit is cleared by the corresponding MCS channel, when the channel is entering a blocking ARU read or write instruction.

Bit 31:8 **Reserved:** Read as zero, should be written as zero.
Note: Only the first T bits of this register (bit 0 to T-1) are functionally implemented. The other bits (bit T to 31) are reserved bits.

15.11.17 Register MCS[i]_CWT

Address Offset:	**see Appendix B**																**Initial Value: 0x0000_0000**															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																								CWT7	CWT6	CWT5	CWT4	CWT3	CWT2	CWT1	CWT0
Mode	R																								RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw
Initial Value	0x000000																								0	0	0	0	0	0	0	0

Bit 0 **CWT0:** Cancel WURM instruction for channel 0.
0 = Do nothing.
1 = Cancel any pending WURM instruction.

Bit 1 **CWT1:** Cancel WURM instruction for channel 1.
0 = Do nothing.
1 = Cancel any pending WURM instruction.

Bit 2 **CWT2:** Cancel WURM instruction for channel 2.
0 = Do nothing.
1 = Cancel any pending WURM instruction.

Bit 3 **CWT3:** Cancel WURM instruction for channel 3.
0 = Do nothing.
1 = Cancel any pending WURM instruction.

Bit 4 **CWT4:** Cancel WURM instruction for channel 4.
0 = Do nothing.
1 = Cancel any pending WURM instruction.

Bit 5 **CWT5:** Cancel WURM instruction for channel 5.
0 = Do nothing.
1 = Cancel any pending WURM instruction.

Bit 6 **CWT6:** Cancel WURM instruction for channel 6.

0 = Do nothing.
1 = Cancel any pending WURM instruction.

Bit 7 **CWT7:** Cancel WURM instruction for channel 7.
0 = Do nothing.
1 = Cancel any pending WURM instruction.

Note: The CWTx (x = 0 ...T-1) bit inside the **STA** register of the corresponding MCS-channel is set and any pending WURM instruction is canceled. The MCS-channel resumes with the instruction after the WURM instruction.

Note: The CWTx (x = 0 ...T-1) bit is cleared by the corresponding MCS channel, when the channel reaches a WURM instruction.

Bit 31:8 **Reserved:** Read as zero, should be written as zero.

Note: Only the first T bits of this register (bit 0 to T-1) are functionally implemented. The other bits (bit T to 31) are reserved bits.

15.11.18 Register MCS[i]_ERR

Address Offset:	**see Appendix B**																**Initial Value:**								**0x00000000**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																								ERR7	ERR6	ERR5	ERR4	ERR3	ERR2	ERR1	ERR0
Mode	R																								RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw
Initial Value	0x000000																								0	0	0	0	0	0	0	0

Bit 0 **ERR0**: Error State of MCS-channel 0.
0 = No error signal.
1 = Error signal is pending.

Bit 1 **ERR1**: Error State of MCS-channel 1.
0 = No error signal.
1 = Error signal is pending.

Bit 2 **ERR2**: Error State of MCS-channel 2.
0 = No error signal.
1 = Error signal is pending.

Bit 3 **ERR3**: Error State of MCS-channel 3 .
0 = No error signal.
1 = Error signal is pending.

Bit 4 **ERR4**: Error State of MCS-channel 4.

0 = No error signal.
1 = Error signal is pending.

Bit 5 **ERR5**: Error State of MCS-channel 5.
0 = No error signal.
1 = Error signal is pending.

Bit 6 **ERR6**: Error State of MCS-channel 6.
0 = No error signal.
1 = Error signal is pending.

Bit 7 **ERR7**: Error State of MCS-channel 7.
0 = No error signal.
1 = Error signal is pending.

Note: The CPU can read the **ERRx** (x = 0...T-1) bits in order to determine the current error state of the corresponding MCS-channel x.

Note: The error state is also evaluated by the sub module MON, if this module is available.

Note: Writing a value 1 to this bit resets the corresponding error state and resets the channel internal ERR bit in the **STA** and channel **CTRL** registers. Moreover, each write access to this bit also sets the **ERR_SRC_ID** bit field of register **MCS[i]_CTRL_STAT** to its reset value.

Bit 31:8 **Reserved:** Reserved
Note: Only the first T bits of this register (bit 0 to T-1) are functionally implemented. The other bits (bit T to 31) are reserved bits.

15.11.19 Memory MCS[i]_MEM

Address Offset:	**see Appendix B**																**Initial Value:**								**0x00000000**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	DATA																															
Mode	RW																															
Initial Value	0x0000_0000																															

Bit 31:0 **DATA:** MCS memory location.

16 Memory Configuration (MCFG)

16.1 Overview

The Memory Configuration submodule (MCFG) is an infrastructure module that organizes physical memory blocks and maps them to the RAM ports 0 and 1 of available Multi Channel Sequencer (MCS) modules.

The following parameters are design variables for the MCFG hardware structure that can vary in its range for different devices:

MAW - Memory address width of a large physical memory block.
ERM - Enable RAM1 MSB (0 - RAM1 MSB disabled,
1 - RAM1 MSB enabled)

The actual values for these parameters can be obtained from the device specific Appendix.
It should be noted that the actual value of the parameter ERM can be obtained by the bit **ERM** of the register **CCM[i]_HW_CONF**.

Depending on the value of parameter ERM, the MCFG module assumes externally connected physical RAM modules with different sizes. If ERM = 0, MCFG assumes that each MCS instance provides a large physical memory block with 2^{MAW} memory locations each 32 bit wide which leads to a RAM module with 2^{MAW+2} (byte wise) memory addresses. Further each MCS instance provides a small physical memory block with 2^{MAW-1} memory locations each 32 bit wide leading to a RAM module with 2^{MAW+1} (byte wise) memory addresses. If ERM = 1, MCFG assumes that each MCS instance provides two large physical memory block each with 2^{MAW} memory locations each 32-bit leading to a RAM module with 2^{MAW+2} (byte wise) memory addresses.

In order to support different memory sizes for different MCS instances, the MCFG module provides three layout configurations for reorganization of memory pages mapped to the RAM ports of neighboring MCS modules. Figure 16.1.1 shows all layout configurations for the case that ERM = 0 and Figure 16.1.3 shows the layout configurations for the case that ERM = 1. Each box in these pictures represents a physical memory block.

The layout configuration DEFAULT is always assigning a memory block of size 2^{MAW}x32 bits to MCS RAM port 0. Depending on ERM, RAM port 1 of each MCS is assigned to a memory block of size 2^{MAW-1}x32 bits (ERM = 0) or a memory block of size 2^{MAW}x32 bits (ERM = 1).

The layout configuration SWAP is swapping the memory block assigned to RAM port 1 of the current MCS instance with the memory block assigned to RAM port 0 of the successive MCS instance. If ERM = 0, this means that the memory of the current MCS instance is increased by 2^{MAW-1}x32 bits but the memory of the successor is decreased by 2^{MAW-1}x32 bits compared to the DEFAULT configuration. If ERM = 1, the SWAP configuration has no effect on the memory sizes of the individual MCS instances.

The layout configuration BORROW is borrowing the memory block assigned to RAM port 0 of the successive MCS instance for the current instance. This means, the memory of the current MCS module is increased by 2^{MAW}x32 bits but the memory of the successor is decreased by 2^{MAW}x32 bits compared to the DEFAULT configuration.

Considering the order the mentioned MCS modules, it should be noted that the successor of the last MCS instance is the first MCS instance MCS0.

The actual sizes of the memory pages mapped to the MCS RAM ports 0 and 1 depends on the layout configuration for of current instance MCS[i] and the layout configuration of the preceding memory instance MCS[i-1]. The sizes of these memory pages can be obtained by the layout parameters MP0 and MP1, as described in the specification of the MCS.

Table 16.1.2 and Figure 16.1.4 summarize the layout parameters MP0 and MP1 of MCS instance MCS[i] for the case that ERM = 0 and ERM = 1. Note that the predecessor of instance MCS0 is last available MCS instance.

The addressing of memory port 0 ranges from 0 to MP0-4 and the addressing of memory page 1 ranges from MP0 to MP1-4.

This document assumes that the GTM implementation embeds 8 MCS instances. However, the actual number of implemented MCS instances can be obtained from [1].

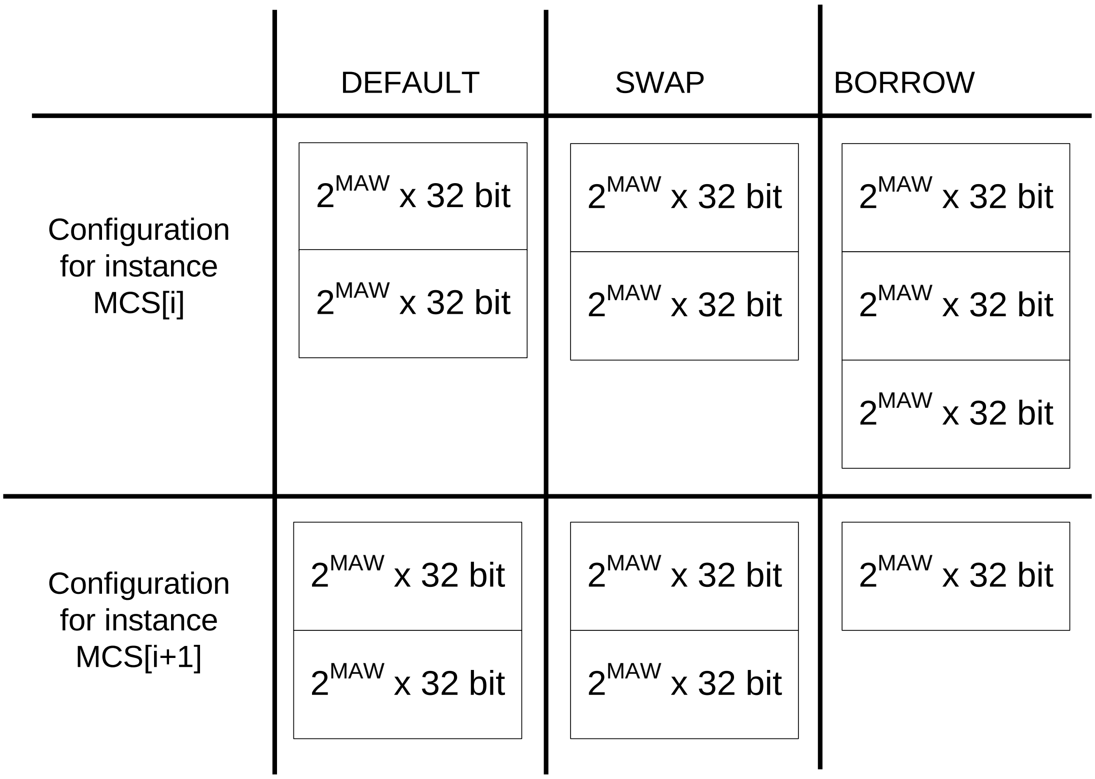
16.1.1 Memory Layout Configurations (ERM = 0)

	DEFAULT	SWAP	BORROW
Configuration for instance MCS[i]	2^{MAW} x 32 bit 2^{MAW-1} x 32 bit	2^{MAW} x 32 bit 2^{MAW} x 32 bit	2^{MAW} x 32 bit 2^{MAW} x 32 bit 2^{MAW-1} x 32 bit
Configuration for instance MCS[i+1]	2^{MAW} x 32 bit 2^{MAW-1} x 32 bit	2^{MAW-1} x 32 bit 2^{MAW-1} x 32 bit	2^{MAW-1} x 32 bit

16.1.2 Memory Layout Parameters (ERM = 0)

			Memory Layout Option of preceding MCS instance MCS[i-1]		
			DEFAULT	**SWAP**	**BORROW**
Memory Layout Option of current MCS instance MCS[i]	**DEFAULT**	MP0	2^{MAW+2}	2^{MAW+1}	0
		MP1	$2^{MAW+2}+2^{MAW+1}$	2^{MAW+2}	2^{MAW+1}
	SWAP	MP0	2^{MAW+2}	2^{MAW+1}	0
		MP1	2^{MAW+3}	$2^{MAW+2}+2^{MAW+1}$	2^{MAW+2}
	BORROW	MP0	2^{MAW+2}	2^{MAW+1}	0
		MP1	$2^{MAW+3}+2^{MAW+1}$	2^{MAW+3}	$2^{MAW+2}+2^{MAW+1}$

16.1.3 Memory Layout Configurations (ERM = 1)



	DEFAULT	SWAP	BORROW
Configuration for instance MCS[i]	2^{MAW} x 32 bit 2^{MAW} x 32 bit	2^{MAW} x 32 bit 2^{MAW} x 32 bit	2^{MAW} x 32 bit 2^{MAW} x 32 bit 2^{MAW} x 32 bit
Configuration for instance MCS[i+1]	2^{MAW} x 32 bit 2^{MAW} x 32 bit	2^{MAW} x 32 bit 2^{MAW} x 32 bit	2^{MAW} x 32 bit

16.1.4 Memory Layout Parameters (ERM = 1)

Memory Layout Option of current MCS instance MCS[i]			Memory Layout Option of preceding MCS instance MCS[i-1]		
			DEFAULT	SWAP	BORROW
	DEFAULT	MP0	2^{MAW+2}	2^{MAW+2}	0
		MP1	2^{MAW+3}	2^{MAW+3}	2^{MAW+2}
	SWAP	MP0	2^{MAW+2}	2^{MAW+2}	0
		MP1	2^{MAW+3}	2^{MAW+3}	2^{MAW+2}
	BORROW	MP0	2^{MAW+2}	2^{MAW+2}	0
		MP1	$2^{MAW+2}+2^{MAW+3}$	$2^{MAW+2}+2^{MAW+3}$	2^{MAW+3}

16.2 MCFG Configuration Registers Overview

16.2.1 MCFG Configuration Registers Overview Table

Register Name	Description	Details in Section
MCFG_CTRL	MCFG Memory layout configuration.	16.3.1

16.3 MCFG Configuration Registers

16.3.1 Register MCFG_CTRL

Address Offset:	see Appendix B																Initial Value: 0x00000000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved												MEM9		MEM8		MEM7		MEM6		MEM5		MEM4		MEM3		MEM2		MEM1		MEM0	
Mode	R												RPw		RPw		RPw		RPw		RPw		RPw		RPw		RPw		RPw		RPw	
Initial Value	0x0000 00												00		00		00		00		00		00		00		00		00		00	

Bit 1:0 **MEM0**: Configure Memory pages for MCS-instance MCS0.
0b00 = DEFAULT configuration
0b01 = SWAP configuration
0b10 = BORROW configuration
0b11 = Reserved

Bit 3:2 **MEM1**: Configure Memory pages for MCS-instance MCS1.
0b00 = DEFAULT configuration
0b01 = SWAP configuration
0b10 = BORROW configuration
0b11 = Reserved

Bit 5:4 **MEM2**: Configure Memory pages for MCS-instance MCS2.
0b00 = DEFAULT configuration
0b01 = SWAP configuration
0b10 = BORROW configuration
0b11 = Reserved

Bit 7:6 **MEM3**: Configure Memory pages for MCS-instance MCS3.
0b00 = DEFAULT configuration
0b01 = SWAP configuration
0b10 = BORROW configuration
0b11 = Reserved

Bit 9:8 **MEM4**: Configure Memory pages for MCS-instance MCS4.
0b00 = DEFAULT configuration
0b01 = SWAP configuration
0b10 = BORROW configuration
0b11 = Reserved

Bit 11:10 **MEM5**: Configure Memory pages for MCS-instance MCS 5.
0b00 = DEFAULT configuration
0b01 = SWAP configuration
0b10 = BORROW configuration
0b11 = Reserved

Bit 13:12 **MEM6**: Configure Memory pages for MCS-instance MCS6.
0b00 = DEFAULT configuration
0b01 = SWAP configuration
0b10 = BORROW configuration

	0b11 = Reserved
Bit 15:14	**MEM7**: Configure Memory pages for MCS-instance MCS7. 0b00 = DEFAULT configuration 0b01 = SWAP configuration 0b10 = BORROW configuration 0b11 = Reserved
Bit 17:16	**MEM8**: Configure Memory pages for MCS-instance MCS8. 0b00 = DEFAULT configuration 0b01 = SWAP configuration 0b10 = BORROW configuration 0b11 = Reserved
Bit 19:18	**MEM9**: Configure Memory pages for MCS-instance MCS9. 0b00 = DEFAULT configuration 0b01 = SWAP configuration 0b10 = BORROW configuration 0b11 = Reserved
Bit 31:20	**Reserved:** Read as zero, should be written as zero.

NOTE: It should be noted that the actual GTM-IP implementation may embed less MCS instances than mentioned in this register (see [1]). In this case this register only implements the register bits for available MCS instances.

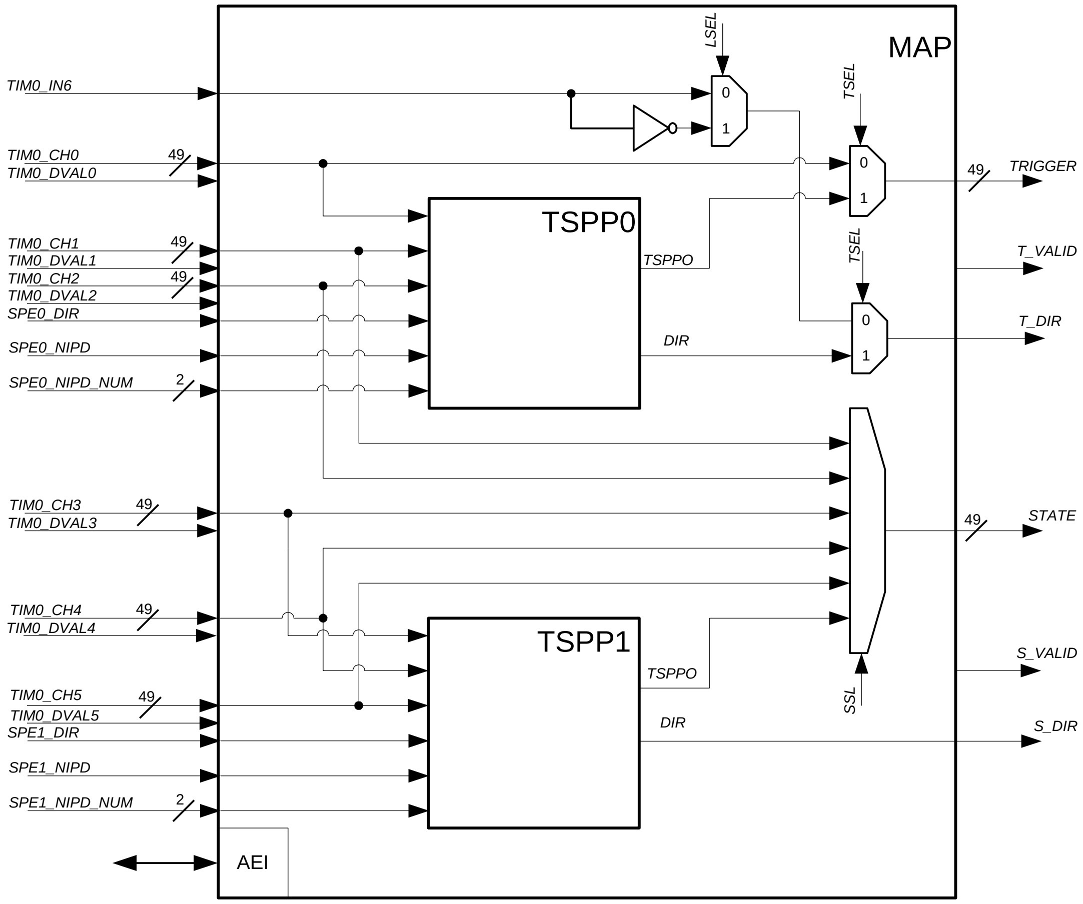
NOTE: This register is only writable if the bit **RF_PROT** in register **GTM_CTRL** is cleared.

17 TIM0 Input Mapping Module (MAP)

17.1 Overview

The MAP submodule generates the two input signals *TRIGGER* and *STATE* for the submodule DPLL by evaluating the output signals of the channel 0 up to channel 5 of submodule TIM0. By using the TIM as input submodule, the filtering of the input signals can be done inside the TIM channels themselves. The MAP submodule architecture is depicted in figure 17.1.1.

17.1.1 MAP Submodule architecture



Generally, the MAP submodule can route the channel signals coming from TIM0 in three ways. First, it is possible to route the whole 49 bits of data coming from channel 0 of module TIM0 (TIM0_CH0) to the *TRIGGER* signal which is then provided to the DPLL together with the *T_VALID* signal.

Second, the MAP module can route one of the five signals coming from the module TIM0 (i.e. the signals coming from channel 1 up to channel 5) to the output signal *STATE* which is then provided to the module DPLL together with the *S_VALID* signal.

Third, the *TRIGGER*, *T_VALID*, *STATE* and *S_VALID* signals can be generated out of the TIM Signal Preprocessing (TSPP) subunits. This is done in combination with the Sensor Pattern Evaluation (SPE) submodule described in chapter 19.

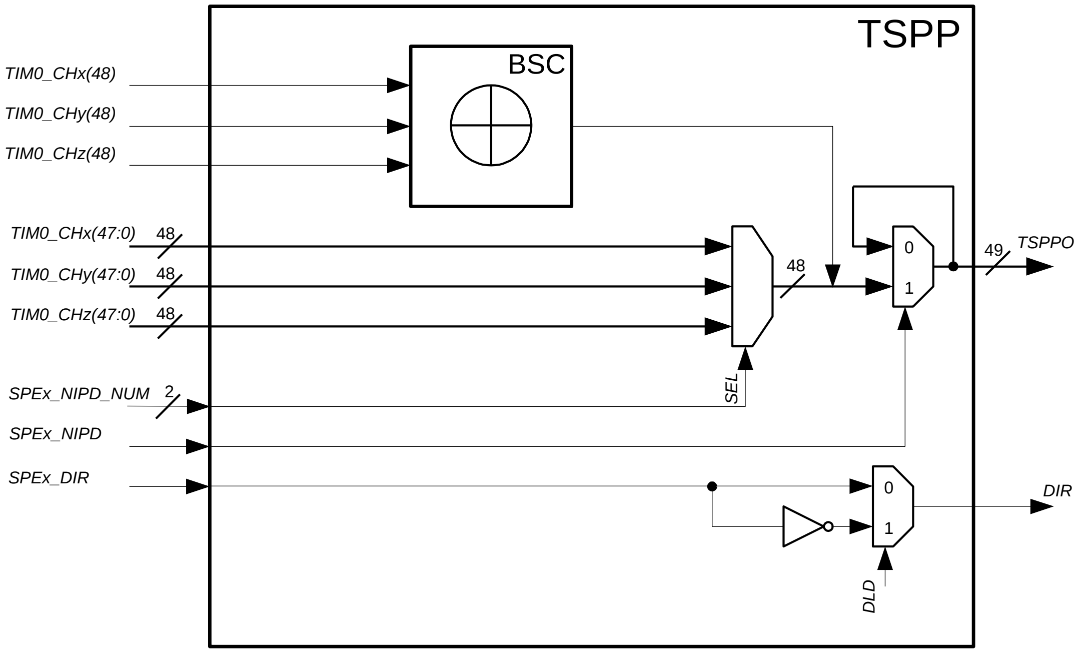
There, the signal *TRIGGER* is generated in subunit TSPP0 out of the TIM0 signals coming from channel 0 up to 2.
The signal *STATE* is generated in subunit TSPP1 out of the TIM signals coming from channel 3 up to channel 5.
This is only be done, when the TSSPx subunits are enabled and when the *SPEx_NIPD* signal is raised by the SPE submodule. The *SPEx_NIPD_NUM* signal encodes, which of the 3 *TIMx_CHy* input signals has been changed. The *SPEx_DIR* signal is routed through the TSPPx subunit and implements the *T_DIR* or *S_DIR* signal.

A third method to provide a direction signal to DPLL is to use TIM0 channel 6 input (*TIM0_IN6*) and to route it instead of the *DIR* signal coming from TSSOP0 to the MAP output *T_DIR* (set TSEL=0)

17.2 TIM Signal Preprocessing (TSPP)

The TSPP combines the three 49 bit input streams coming from the TIM0 submodule and generates one combined 49 bit output stream *TSPPO*. The input stream combination is done in the unit Bit Stream Combination (BSC). The architecture of the TSPP is shown in figure 17.2.1.

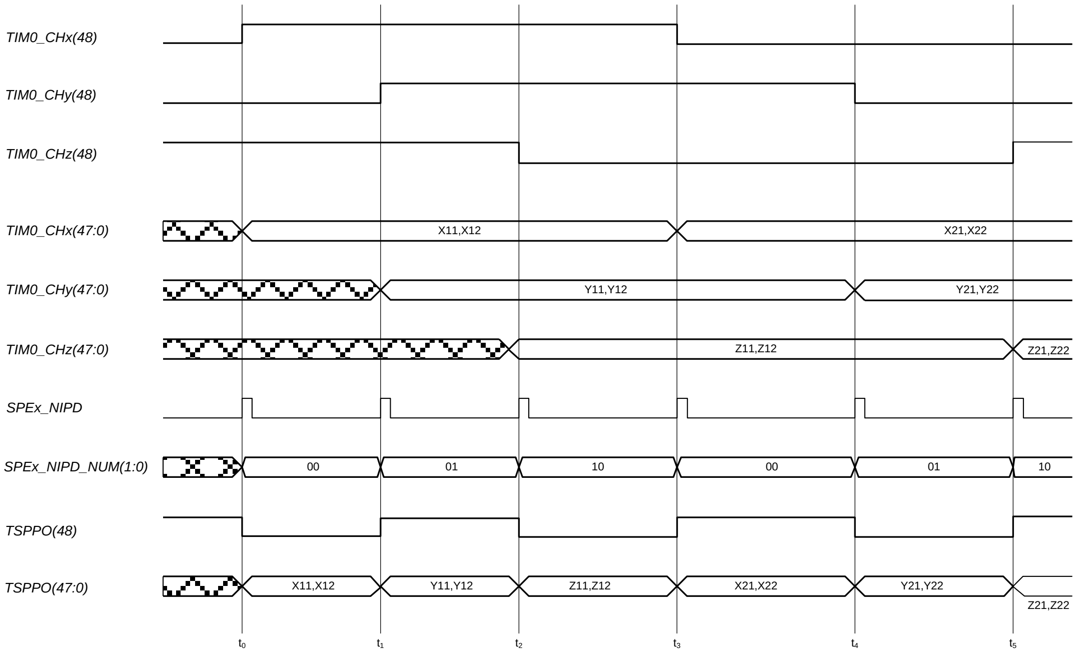
17.2.1 TIM Signal Preprocessing (TSPP) subunit architecture



17.2.2 Bit Stream Combination

The BSC subunit is used to xor-combine the three most significant bits *TIM0_CHx(48)*, *TIM0_CHy(48)* and *TIM0_CHz(48)* of the TIM0 inputs. The xor-combined signal is merged with the remaining 48 bits of one of the three input signals *TIM0_CHx(47...0)*, *TIM0_CHy(47...0)* or *TIM0_CHz(47...0)* the *TSPPO* signal. The selection is done with the *SPEx_NIPD_NUM* input signal coming from the SPE submodule. The action, when the 49 bits are transferred to the TSPPO and the T_VALID or S_VALID signal is raised is determined by the SPEx_NIPD signal coming from the SPE submodule. The *TSPPO* output signal generation is shown in the example in Figure 17.3.

17.2.2.1 TSPP Signal generation for signal TSPPO



The *SPEx_NIPD_NUM* input signal determines, which data is routed to the *TSPPO* signal. At the first edge of *TIM0_CHx(48)* the new data X11 and X12 are routed to *TSPPO(47:0)*. The values X11 and X12 are the two 24 bit values coming from the TIM input channel TIM0_CHx. The next edge is at time t_1 on signal *TIM0_CHy(48)*. Therefore, at time t_1 the *TSPPO(48)* signal level changes and the *TSPPO(47:0)* is set to Y11 and Y12 and so forth.

17.3 MAP Register overview

Register name	Description	Details in Section
MAP_CTRL	MAP Control register	17.4.1

17.4 MAP Register description

17.4.1 Register MAP_CTRL

Address Offset:	**see Appendix B**																**Initial Value:**							**0x0000_0000**								
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved	TSPP1_I2V	TSPP1_I1V	TSPP1_I0V	Reserved		TSPP1_DLD	TSPP1_EN	Reserved	TSPP0_I2V	TSPP0_I1V	TSPP0_I0V	Reserved		TSPP0_DLD	TSPP0_EN	Reserved											LSEL	SSL			TSEL
Mode	R	RW	RW	RW	R		RW	RW	R	RW	RW	RW	R		RW	RW	R											RW	RW			RW
Initial Value	0	0	0	0	0		0	0	0	0	0	0	00		0	0	0											0	000			0

Bit 0 **TSEL**: *TRIGGER* signal output select.
0 = *TIM0_CH0* selected as *TRIGGER* output signal.
TIM0_IN6 (TIM0 channel 6 input) is used as direction signal T_DIR.
1 = *TSPP0_TSPPO* selected as *TRIGGER* output signal.

Bit 3:1 **SSL**: *STATE* signal output select.
0b000: *TIM0_CH1* selected as *STATE* output signal.
0b001: *TIM0_CH2* selected as *STATE* output signal.
0b010: *TIM0_CH3* selected as *STATE* output signal.
0b011: *TIM0_CH4* selected as *STATE* output signal.
0b100: *TIM0_CH5* selected as *STATE* output signal.
0b101: *TSPP1_TSPPO* selected as *STATE* output signal.
0b110: same as '000'
0b111: same as '000'

Bit 4 **LSEL**: TIM0_IN6 input level selection
0 = *TIM0_IN6* input level '0' encodes TRIGGER in forward direction.
1 = *TIM0_IN6* input level '1' encodes TRIGGER in forward direction.

Bit 15:5 **Reserved**
Note: Read as zero, should be written as zero.

Bit 16 **TSPP0_EN**: Enable of TSPP0 subunit.
0 = TSPP0 disabled.
1 = TSPP0 enabled.

Bit 17 **TSPP0_DLD**: DIR level definition bit.
0 = *SPEx_DIR* signal is routed through as is.
1 = *SPEx_DIR* signal is inverted.

Bit 19:18 **Reserved**
Note: Read as zero, should be written as zero.

Bit 20 **TSPP0_I0V**: Disable of TSPP0 *TIM0_CHx(48)* input line.
0 = Input line enabled.
1 = Input line disabled; input for TSPP0 is set to zero (0).

Bit 21 **TSPP0_I1V**: Disable of TSPP0 *TIM0_CHy(48)* input line.
0 = Input line enabled.

1 = Input line disabled; input for TSPP0 is set to zero (0).

Bit 22 **TSPP0_I2V**: Disable of TSPP0 *TIM0_CHz(48)* input line.
0 = Input line enabled.
1 = Input line disabled; input for TSPP0 is set to zero (0).

Bit 23 **Reserved**
Note: Read as zero, should be written as zero.

Bit 24 **TSPP1_EN**: Enable of TSPP1 subunit.
0 = TSPP1 disabled.
1 = TSPP1 enabled.

Bit 25 **TSPP1_DLD**: DIR level definition bit.
0 = *SPEx_DIR* signal is routed through as is.
1 = *SPEx_DIR* signal is inverted.

Bit 27:26 **Reserved**
Note: Read as zero, should be written as zero.

Bit 28 **TSPP1_I0V**: Disable of TSPP1 *TIM0_CHx(48)* input line.
0 = Input line enabled.
1 = Input line disabled; input for TSPP1 is set to zero (0).

Bit 29 **TSPP1_I1V**: Disable of TSPP1 *TIM0_CHy(48)* input line.
0 = Input line enabled.
1 = Input line disabled; input for TSPP1 is set to zero (0).

Bit 30 **TSPP1_I2V**: Disable of TSPP1 *TIM0_CHz(48)* input line.
0 = Input line enabled.
1 = Input line disabled; input for TSPP1 is set to zero (0).

Bit 31 **Reserved**
Note: Read as zero, should be written as zero.

18 Digital PLL Module (DPLL)

18.1 Overview

The digital PLL (DPLL) sub-module is used for frequency multiplication. The purpose of this module is to get a higher precision of position or value information also in the case of applications with rapidly changed input frequencies. There are two input signals *TRIGGER* and *STATE* for which periodic events are processed. The time period between two active events is called an increment. Each increment is divided into a given number of sub increments by pulses called SUB_INC. The resolution of the generated pulses is restricted by the period of the CMU_CLK0 clock or the TS_CLK respectively (see description of the modules TBU, CMU). The input signals *TRIGGER* and *STATE* can have the meaning of position information of linear or angle motions, mass flow values, temperature, pressure or level of liquids.
By means of the DPLL the load of the CPU can be reduced essentially by relieving it from repeated or periodic standard tasks.

The DPLL has to perform the following tasks:

- prediction of the duration of the current increment in chapter 18.6
- generation of SUB_INC1,2 pulses for up to 2 position counters in normal or emergency mode (see chapter 18.8.3)
- synchronization of the actual position (under CPU control, see chapter 18.8.6.2)
- possibility of seamless switch to emergency mode and back under CPU control, see configuration register DPLL_CTRL_0 at chapter 18.12
- prediction of position and time related events in chapter 18.7

18.2 Requirements and demarcation

The two input signals *TRIGGER* and *STATE* can be sensor signals from the same device or from two independent devices. When they come from the same device the *TRIGGER* input is typically a more frequent signal and *STATE* is a less frequent signal. In such a case the *STATE* signal can support an emergency mode, when no *TRIGGER* signal is available. There are also applications supported when *STATE* and *TRIGGER* are independent signals from different devices. Both input signals are combined with a validation signal *T_ VALID* or *S_ VALID* respectively, which shows the appearance of new data and must result in a data fetch and a start of the correspondent state machine to perform the calculations (see explanation below).

When *STATE* is a redundant signal of the same device only the *TRIGGER* input is used to generate the SUB_INC1 pulses in normal mode. There is a configuration possible, called emergency mode, for which the SUB_INC1 pulses are generated using the *STATE* input signal.
The decision to switch in the emergency mode and back is made outside the DPLL. The CPU must switch the configuration bit RMO (reference mode) in the DPLL_CTRL_0 register (see chapter 18.12). Because a switch in emergency mode can appear suddenly, the information of the last increment duration of the *STATE* input up to FULL_SCALE should be stored always as a precaution.

The filtering as well as the combination or choice of the input signals is made in the TIM sub-module (see chapter 10) by use of a configurable filter algorithm for each slope and signal as well as in the MAP module (see chapter 16) the right *TRIGGER* or *STATE* signal is selected by a multiplexer or in the SPE module (see chapter 18) different signals are combined to a *TRIGGER* or *STATE* signal by using an antivalence operation.

The filter delay value of the signal is transmitted from the TIM module in the *FT* part of the corresponding signal, because the delay conditions of the signals can change during application.

The filter delays depend also on the filter algorithms used. Only the effective filter delay can be considered in the DPLL.

In order to provide the timing conditions to the DPLL the input trigger signals should have a time stamp (and optional in addition a filter value and a signal level value, as stated above) with an appropriate resolution. The resolution of the time stamps can be either the same resolution as the input time base TBU_TS0 (see chapter 18.4.1) or 8 times higher, selected by configuration bits in the DPLL_CTRL_1 register (see chapter 18.12.2). The time base TBU_TS0 is used to predict events in the future, called actions.

At the SUB_INCx outputs a predefined number of pulses between each active slope of the *TRIGGER/STATE* signal is generated, when the correspondent pulse generator is enabled by the enable bits SGEx=1 in the DPLL_CTRL_1 register (see chapter 18.12.2).

Dependent on configuration different strategies can be used to correct a wrong pulse number.

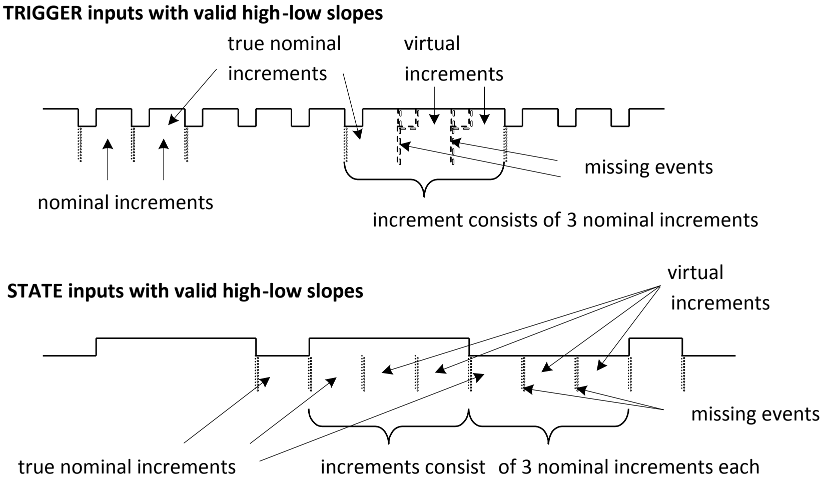
The FULL_SCALE range is divided into a fix number of nominal increments. Nominal increments do have the same size. The number of nominal increments in HALF_SCALE is specified in the DPLL_CTRL_0 register (see chapter 18.12).
For synchronization purposes some *TRIGGER/STATE* input signals can be suppressed in dependency on the current position. Therefore an increment as duration between two active input events can be either a nominal increment or it can consist of more than one nominal increment.

While a true nominal increment starts with an active event a virtual increment (of always nominal size) is an increment which starts with a missing event. Each increment which represents a gap (e.g. for synchronization purposes) consists of exactly one true nominal increment and at least one virtual increment, each of them having the same nominal duration (see figure below).

18.3 Input signal courses

Typical input signal courses are shown in the figure below.

18.3.1 Trigger and State Input Signal


TRIGGER inputs with valid high-low slopes
true nominal increments
virtual increments
missing events
nominal increments
increment consists of 3 nominal increments
STATE inputs with valid high-low slopes
virtual increments
missing events
true nominal increments
increments consist of 3 nominal increments each

18.4 Block and interface description

The block description of the DPLL is shown in the following figure.

18.4.1 DPLL Block Diagram

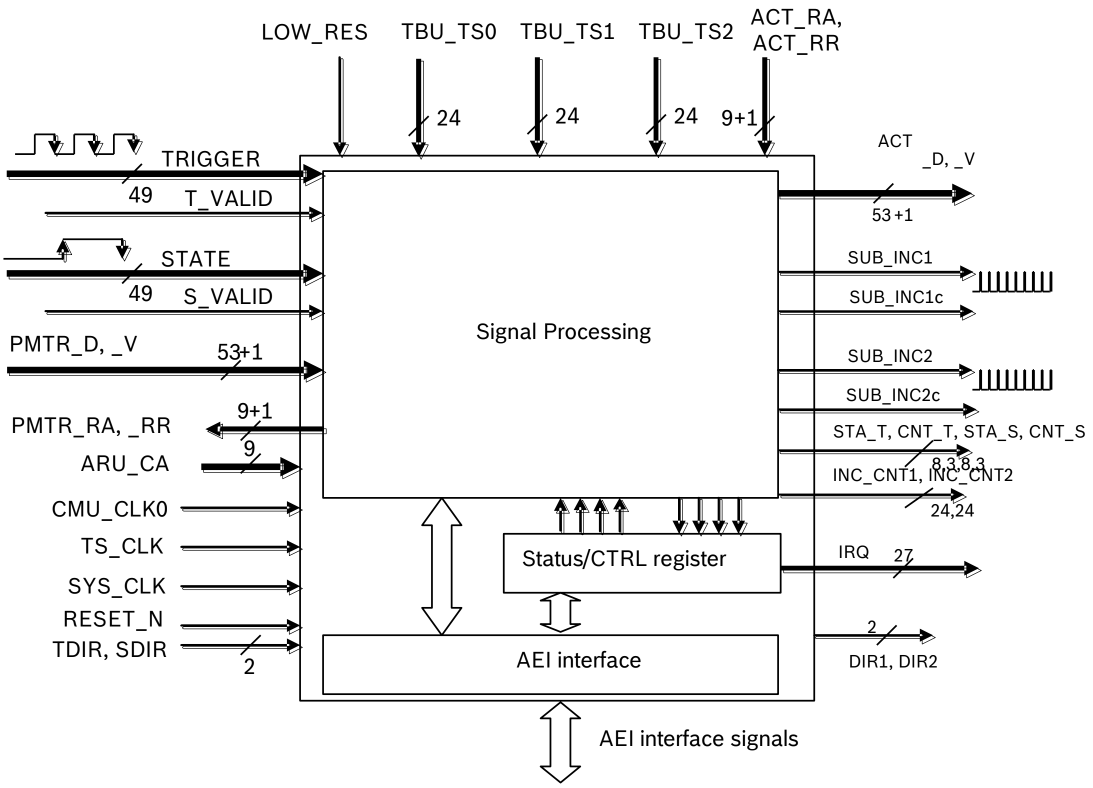


Table 18.4.2 summarizes the interface signals of the DPLL shown by the block diagram above.

18.4.2 Interface description of DPLL

Name	Width	I/O	Description	Comment
TRIGGER	49	I	Normal Signal for triggering DPLL by positions/values Bit(48)= TRIGGER_S Bits(47:24)= TRIGGER_FT Bits(23:0)= TRIGGER_TS	One bit signal value (SV), 24 bits filter delay value info and 24 bits time stamp, filtered in different modes.
T_VALID	1	I	The values of *TRIGGER* are valid	Announces the arrival of a new *TRIGGER* value
STATE	49	I	Assistance signal for synchronization STATE(48)= STATE_S STATE(47:24)= STATE_FT STATE(23:0)= STATE_TS	Replacement of signal *TRIGGER* for emergency situations, or signal from an independent device; bits like above, corresponding

S_VALID	1	I	The values of STATE are valid	Announces the arrival of a new STATE value
PMTR_D	53	I	Position minus time request data, delivered by ARU on request for up to 24 requests PMTR_RR; SV_i=PMTR_D(52:48): ACB bits, directly written to the correspondent DPLL_ACB_j registers PSA[i]=PMTR_D(47:24): position value for action DLA[i]=PMTR_D(23:0) time delay value for action	Data values for calculation of actual Actions; the values are requested by AENi=1[1] and CAIP=0[2] ; a served request is shown by PMTR_V which signals that valid PMTR data arrived and they are written immediately after that to the corresponding RAM regions and registers; The DLA[i] values must have the same resolution as the TBU_TS0 input.
PMTR_V	1	I	signals a valid PMTR_D value, that means data is delivered on request	when valid: PMTR_D overwrites data in the PSA[i] and DLA[i] registers, also when the corresponding ACT_N[i][3] bit =1;
ARU_CA	9	I	Channel address; for valid PMTR addresses: demand data by setting PMTR_RR=1 when enabled by AENi=1[1] and CAIP=0[2] ;	counter value of ARU selects PMTR_RA and PMTR_RR when a valid address
PMTR_RA	9	O	read address of PMTR access	reflects ID_PMTR_i according to the selected channel address
PMTR_RR	1	O	read request of PMTR access; suppressed for CAIPi=1 (see DPLL_STATUS register)	reflects the value of the corresponding AENi[1] bit while the correspondent bit CAIPi=0[2]
ACT_D	53	O	Output of a time stamp, a position and a control signal for a calculated action; SV_i=ACT_D(52:48) : ACB bits, directly written from the correspondent PMTR_D signals; ACT_D(47:24) is the calculated position	Future time stamp with the resolution as TBU_TS0 input, additional position information and additional control bits;

			value PSAC[i] for the action in relation to TBU_TS1 or 2 [6)] and ACT_D(23:0) is the time stamp value TSAC[i] for the action in relation to TBU_TS0 [6)]	
ACT_V	1	O	ACT_D value is available and valid; blocking read access	for a valid action address: ACT_V reflects the shadow value of ACT_N[i][3)] (ACT_N[i] is 1 when new PMTR values are available and the shadow register is updated, when a calculation of the actual PMTR values was done); reset after reading of the ACT_D values
ACT_RA	9	I	ACTION read address;	address bits for selection of all 24 action channels
ACT_RR	1	I	read request of selected action	the action data is demanded from another module
IRQ	27	O	Interrupt request output	Interrupts of DPLL;
SUB_INC1	1	O	Pulse output for *TRIGGER* input filter	sub-position increment provided continuously
SUB_INC2	1	O	Pulse output for *STATE* input filter (when *TRIGGER* and *STATE* are used for 2 independent devices)	sub-position increment provided continuously
SUB_INC1c	1	O	Pulse output for time base unit 1 in compensation mode (can stop in automatic end mode)	sub-position increment related to *TRIGGER* input
SUB_INC2c	1	O	Pulse output for time base unit 2 in compensation mode (can stop in automatic end mode)	sub-position increment related to *STATE* input (when *TRIGGER* and *STATE* are used for 2 independent devices)
TS_CLK	1	I	Time stamp clock	used for generation of the time stamps; this clock is used to generate the SUB_INC1,2 pulses
CMU_CLK0	1	I	CMU clock 0	used for rapid pulse correction of SUB_INC1,2

SYS_CLK	1	I	System clock	High frequency clock
RESET_N	1	I	Asynchronous reset signal	Low active; After Reset he DPLL is available only after performing the RAM reset procedures by the DPLL hardware.

LOW_RES	1	I	low resolution of TBU_TS0 selected; shows which of the 27 bits of TBU_TS0 are connected to the DPLL	LOW_RES=0:
				TBU_TS0(DPLL)= lower 24 Bits of TBU_TS0(TBU);
				LOW_RES=1:
				TBU_TS0(DPLL)= higher 24 Bits of TBU_TS0(TBU);
				In the case LOW_RES=1 the TS0_HRT and/or TS0_HRS bits can be set [5]

TBU_TS0	24	I	Actual time stamp from TBU; is needed to decide, if a calculated action is already in the past	24 bit time input, with a resolution of the time stamp clock
TBU_TS1	24	I	Actual position/value stamp 1; for calculation of position stamps (*TRIGGER*/*STATE*)	24 bit pos./val. input, with a resolution of the SUB_INC1 pulses
TBU_TS2	24	I	Actual position/value stamp 2; to be implemented for an additional independent position	ditto for SUB_INC2 for calculation of position stamps (*STATE*) for SMC[5]=RMO[4]=1
TDIR	1	I	Direction of *TRIGGER* input values (TDIR=0 does mean a forward direction and TDIR=1 a backward direction)	direction information from multiple sensors valid only for SMC[5]=1 or IDDS[5]=1
SDIR	1	I	Direction of *STATE* input values (SDIR=0 does mean a forward direction and SDIR=1 a backward direction)	direction information from multiple sensors valid only for SMC[5]=1
DIR1	1	O	Direction information of SUB_INC1 (count forwards for DIR1=0 and backwards for DIR1=1)	count direction of TBU_CH1_BASE; DIR1 changes always after the evaluation of the corresponding valid *TRIGGER* slope and after incrementing/decrementing of the address pointer

DIR2	1	O	Direction information of SUB_INC2 (count forwards for DIR2=0 and backwards for DIR2=1)	count direction of TBU_CH2_BASE; DIR2 changes always after the evaluation of the corresponding valid *STATE* slope and after incrementing/decrementing of the address pointer
STA_T	8	O	Status of state machine TRIGGER	Output to MCS0. Signals accessible via uC interface as well (DPLL_STA)
CNT_T	3	O	Count TRIGGER	Output to MCS0. This reflects the count of active *TRIGGER* slopes (mod8). Signals accessible via uC interface as well (DPLL_STA)
STA_S	8	O	Status of state machine STATE	Output to MCS0. Signals accessible via uC interface as well (DPLL_STA)
CNT_S	3	O	Count STATE	Output to MCS0. This reflects the count of active *STATE* slopes (mod8). Signals accessible via uC interface as well (DPLL_STA)
INC_CNT1	24	O	Increment counter of pulse generator 1 (automatic end mode)	Output to MCS0. Signals accessible via uC interface as well (DPLL_INC_CNT1)
INC_CNT2	24	O	Increment counter of pulse generator 2 (automatic end mode)	Output to MCS0. Signals accessible via uC interface as well (DPLL_INC_CNT2)

For references above the following hints are used:

[1] see DPLL_CTRL_x register, x=2,3,4; see 18.12.3,18.12.4,18.12.5

[2] see DPLL_STATUS register; see 18.12.30

[3] see DPLL_ACT_STA register; see 18.12.7

[4] see DPLL_CTRL_0 register; see 18.12.1

[5] see DPLL_CTRL_1 register; see 18.12.2

[6] see DPLL input signal description; see 18.1

18.5 DPLL Architecture

18.5.1 Purpose of the module

The DPLL generates a predefined number of incremental signal pulses within the period between two events of an input *TRIGGER* or *STATE* signal, when the corresponding pulse generator is enabled. The resolution of the pulses is restricted by the frequency of the time stamp clock (TS_CLK). Changes in the period length of the predicted time period of the current increment will result in a change of the pulse frequency in order to get the same number of pulses. This adoption can be performed by DPLL hardware, software or with support of DPLL hardware in different modes.

The basic part of a DPLL is to make a prediction of the current period between two *TRIGGER* and/or *STATE* signal edges. Disturbances and systematic failures must be considered as well as changes of increment duration caused by acceleration and deceleration of the supervised process. Therefore, a good estimation is to be done using some measuring values from the past. When the process to be predicted takes a steady and differentiable course not only the current increment but also some more increments for the future can be predicted. In utilization of such calculations actions for the future can be predicted.

18.5.2 Explanation of the prediction methodology

As already shown in chapter 18.1 the DPLL has to perform different tasks. The basic function for all these tasks is the prediction of the current increment which is based on a relation between increments in the past. Because the relation between two succeeding intervals at a fixed position remains also valid in the case of acceleration or deceleration the prediction of the duration of the current time interval is done by a similarity transformation. Having a good estimation of the current time interval, all the other tasks can be done easily by calculations explained in chapter 18.6.

18.5.3 Clock topology

All registers are read using the system clock *SYS_CLK*. The SUB_INC1,2 pulses generated have in the normal case the highest frequency not higher then *CMU_CLK0* or the half of *TS_CLK* respectively. For individual pulses the frequency can be doubled. All operations can be performed using the system clock.

18.5.4 Clock generation

The clock is generated outside the DPLL.

18.5.5 Typical frequencies

For the system clock a reasonable clock frequency should be applied to give the DPLL module sufficient computational power to calculate all needed values (prediction of next increment, actions) in time. The typical system clock frequency is in the range from 40 MHz up to 150 MHz.

18.5.6 Time stamps and systematic corrections

The time stamps for the input signals *TRIGGER* and *STATE* have 24 bits each. These bits represent the value of the 24 bit free running counter running with a clock frequency selected by the configuration of the TBU. Using a typical frequency of 20 MHz the time stamp represents a relative value of time with a resolution of 50 ns.

The input signals have to be filtered. The filter is not part of the DPLL. The time stamps can have a delay caused by the filter algorithm used. There are delayed and undelayed filter algorithms available and the delay value can depend on a time or a position value.

Systematic deviations of *TRIGGER* inputs can be corrected by a profile, which also considers systematic missing *TRIGGER*s. The increments containing missing *TRIGGER*S are divided into the corresponding number of nominal increments whereas the duration of a nominal increment is the greatest divider of all increments duration.
For each increment this number of enclosed nominal increments is stored in a profile as NT value for *TRIGGER*. When the increment is a nominal increment the NT value is 1.
For the *TRIGGER* input the value NT is stored in the ADT_T field in RAM region 2c.

In the case of **AMT**[4)] = 1 the **ADT_T[i]** values in the RAM region 2c must also contain the adapting information for the *TRIGGER* signal, which considers for each increment a systematic physical deviation **PD** from the perfect nominal increment value with a resolution according to the chosen value of MLT+1, which describes the number of SUB_INC1 pulses for a nominal increment.

The value **PD** for the *TRIGGER* describes the amount of missing or surplus pulses with a sint13 value, to be added to MLT+1 directly. The correction value is in this way also applicable in the case of missing *TRIGGER* inputs for the synchronization gaps. In this

case the amount of provided SUB_INC1 pulses for a nominal increment (MLT+1) + PD is multiplied by NT.
The NT value of the current increment is stored in the variable SYN_T (see **NUTC** register in chapter 18.12.14).

In the case of **RMO**4) = 1 for **SMC**5)=0 (emergency mode) the time stamp of *STATE* is used to generate the output signal SUB_INC1.

More inaccuracy should be accepted in emergency mode because usually there are only fewer events available for FULL_SCALE according to the value SNU4).

For the *STATE* signal the systematic deviations of the increments can be corrected in the same way as for *TRIGGER* by profile and adaptation information as described below.

Systematic deviations of *STATE* inputs can be corrected by a profile, which also considers systematic missing *STATE* events. The increments containing missing *STATEs* are divided into the corresponding number of nominal increments whereas the duration of a nominal increment is the greatest divider of all increments duration.
For each increment this number of enclosed nominal increments is stored in a profile as NS value for *STATE* . When the increment is a nominal increment the NS value is 1.
For the *STATE* input the value NS is stored in the ADT_S field in RAM region 1c3.

In the case of **AMS**4) = 1 the **ADT_S[i]** values in the RAM region 1c3 must contain the adapting information for the STATE signal, which considers for each increment a systematic physical deviation **PD_S** from the perfect nominal increment value with a resolution according to the chosen value of MLS1, which describes the number of SUB_INC1 pulses for a nominal increment (see below).

The number of pulses SUB_INC1 for a nominal *STATE* increment in emergency mode (for SMC=0) is given by the value of MLS1= (MLT + 1)* (TNU + 1) /(SNU + 1) in order to get the same number of pulses in FULL_SCALE for normal and emergency mode. This value has to be configured by the CPU.

The value **PD_S** for the *STATE* describes the amount of missing or surplus pulses with a sint16 value, to be added to MLS1 directly. The correction value is in this way also applicable in the case of missing *STATE* inputs for the synchronization gaps. In this case the amount of provided SUB_INC1 pulses for a nominal increment MLS1 + PD_S is multiplied by NS.
The current NS value is stored in the variable SYN_S (see **NUSC** register in chapter 18.12.15).

For references above the following hints are used:

[1] see DPLL_CTRL_x register, x=2,3,4; see 18.12.3,18.12.4,18.12.5
[2] see DPLL_STATUS register; see 18.12.30
[3] see DPLL_ACT_STA register; see 18.12.7
[4] see DPLL_CTRL_0 register; see 18.12.1
[5] see DPLL_CTRL_1 register; see 18.12.2
[6] see DPLL input signal description; see 18.1

18.5.7 DPLL Architecture overview

As shown in 18.4.1 the DPLL can process different input signals. The signal *TRIGGER* is the normal input signal which gives the detailed information of the supervised process. It can be for instance the information of water or other liquid level representing the volume of the liquid, where each millimeter increasing results in a *TRIGGER* signal generation. In order to get a predefined filling level, without overflow also the inertia of the system must be taken into account. Hence, some delay for closing the inlet valve and also the remaining water amount in the pipe must be considered in order to start the closing action earlier as the filling level will be reached.

A second input signal *STATE* sends an additional (redundant) information for instance at some centimeters and because of intervals with different distances it gives also information about the system state with the direction of the water flow (in or out), while the *TRIGGER* signal must not contain information concerning the flow direction. In some applications the inactive slope of *TRIGGER* can be utilized to transmit direction information. In the case of faults in the *TRIGGER* signal the *STATE* signal is to be processed in order to reach the desired value nevertheless, maybe with some loss of accuracy.

The measuring scale can have some systematic failures, because not all millimeter or centimeter distances measured mean the same value. This could be due to changes in the thickness of the measuring cylinder or the inaccurate position of the marks. These systematic failures are well known by the system and for improvement of the prediction the signals *ADT_T* and *ADT_S* for the correction of the systematic failures of *TRIGGER* and *STATE* respectively are stored in the internal RAM.

The input signals *TRIGGER* and *STATE* are represented as a time stamp signal each, which is stored in the 24 bit TS-part of the corresponding signal.

Information concerning the delay of this signal by filtering of disturbances is stored in the 24 bit FT-part of the signal.

In order to establish the relation of time stamps to the actual time the TBU_TS0[6] value is also provided showing the actual time value used for prediction of actions in the future.

After reaching the desired water level the water is filled in a bottle by draining. After that the water filling is repeated. The water level at draining is observed by the same sensor signals (the same number of *TRIGGER* pulses), but the duration of the draining could be different from the filling time. Both times together form the FULL_SCALE region, while one of them is a HALF_SCALE region, which can differ in time but not in the number of pulses, especially for *TRIGGER*.

For synchronization purposes some *TRIGGER* marks can be omitted in order to set the system to a proper synchronization value (maybe before the upper filling value is reached).

In emergency situations, when the *TRIGGER* signals are missed the *STATE* signal is used instead of.

The PMTR_i [6] signals announce the request for a position minus time calculation for up to 24 events.
All 24 events can be activated using the 24 AENi[1] (action enable) bits. Each of these enable bits are asked by the routing engine for a read access. The corresponding read request is generated by the AENi bit while CAIPx is zero. CAIP1 and CAIP2 are two bits of the DPLL_STATUS register for 12 actions each with the meaning “calculation of actions in progress”, controlled by the state machine (see 18.2) for scheduling the operations.

When such a request is serviced by the ARU (in the case CAIPx=0) the values for position and time are written in the corresponding RAM 1a region (0x0200... 0x025C for the position value and 0x0260... 0x02BC for the delay value), the control bits for the corresponding action are set accordingly. When a new PMTR value arrives, an old value is overwritten without notice and the shadow bit of ACT_N[i] is cleared while the ACT_N[i] (new action) bit in the DPLL_ACT_STA register is set. The ACT_N[i] is cleared, when the currently calculated action value is in the past. Overwriting of old information is possible without data inconsistency because the read request to ARU is suppressed during action calculations by the CAIP1,2 bits. In this way always the last possible PMTR value is used consistently.

For references above the following hints are used:
[1] see DPLL_CTRL_x register, x=2,3,4; see 18.12.3,18.12.4,18.12.5
[2] see DPLL_STATUS register; see 18.12.30
[3] see DPLL_ACT_STA register; see 18.12.7
[4] see DPLL_CTRL_0 register; see 18.12.1
[5] see DPLL_CTRL_1 register; see 18.12.2

[6] see DPLL input signal description; see 18.1

18.5.8 DPLL Architecture description

The DPLL block diagram 18.4.1 will now be explained in detail in combination with some example configurations of the control registers. There are different configuration bits available which can adopt the DPLL to the use case (see chapter 18.12).
Let for example in HALF_SCALE the *TRIGGER* number TNU[4] be 0x3B (which is for TNU+1 = 60 decimal that does mean 120 events in FULL_SCALE) and the number of SUB_INC1 pulses between two *TRIGGER*s MLT[4] be 0x257 (this means 600 pulses per *TRIGGER* event). Than the FULL_SCALE region can be divided into 72000 parts each of them associated with its own SUB_INC1 pulse. For a run through FULL_SCALE all 72000 pulses should appear but maybe with a different pulse frequency between two *TRIGGER* events. For this example after each 600 pulses at the *SUB_INC1* output the next *TRIGGER* event is to be expected with the corresponding new time stamp.

Missing SUB_INC1 pulses due to acceleration have to be taken into account within the next increment. Not one pulse has to be missed or added because of calculation inaccuracy in average for a sufficient number of FULL_SCALE periods. This means that not one pulse is sent in addition and all missing pulses are to be caught up on afterwards.

For the systematic arrangement of *TRIGGER* inputs **the profile** (as already mentioned in chapter 18.5.6 is stored in the RAM region 2c (see chapter 18.14.3). In this field the relative position of gaps can be stored in the NT value and also physical deviations in the PD value.

For the consideration of systematic missing *TRIGGER*s the actual NT value of the profile is stored in the SYN_T bits of the NUTC register (see chapter 18.12.14).

In normal mode the physical deviation values PD in the ADT_T field could be used to balance the local systematic inaccuracy of the *TRIGGER* signal. The value of PD (see chapter 18.14.3) is the pulse difference in the corresponding nominal increment and does mean the number of sub pulses to be added to the nominal number of pulses. PD is a signed integer value using 13 bits: up to +/-4096 pulses can be added for each increment.

The NT value of the profile ADT_T has the value 1, when a nominal increment is assumed. An integer number greater than 1 shows the number of nominal increments to be considered for a gap. For the actual increment after synchronization the corresponding NT value is stored in SYN_T of the NUTC register.

Using the *STATE* input there are similar configuration bits available (see chapter 18.12)
.
Let for example in HALF_SCALE the *STATE* number SNU[4] be 0xB (which is for SNU+1 =12 decimal and while SYSF[4] =0 that does mean 24 events in FULL_SCALE). In order to get the same number of SUB_INC1 pulses for FULL_SCALE as above for *TRIGGER*s the value (MLT+1)=600 is divided by 2*(SNU+1)=24 and multiplied with 2*(TNU+1)=120. The result 3000 must be stored in MLS1 by the CPU (see chapter 18.12.75).

For the systematic arrangement of *STATE* inputs **the profile** (as already mentioned in chapter 18.5.6 is stored in the RAM region 1c3 (see chapter 18.12.90). In this field the relative position of gaps can be stored in the NS value and also physical deviations in the PD_S value.

For the consideration of systematic missing *TRIGGER*s the actual NS value of the profile is stored in the SYN_S bits of the NUSC register (see chapter 18.12.15).

In emergency mode the physical deviation values PD_S in the ADT_S field could be used to balance the local systematic inaccuracy of the *STATE* signal. The value of PD_S (see chapter 18.12.90) is the pulse difference in the corresponding increment and does mean the number of sub pulses to be added to the nominal number of pulses per increment. PD_S is a signed integer value using 16 bits: up to +/-32768 pulses can be added for each increment.

In emergency mode the physical deviation values PD_S in the ADT_S field could be used to balance the local systematic inaccuracy of the *STATE* signal. The value of PD_S (see chapter 18.12.90) is the pulse difference in the corresponding nominal increment and does mean the number of sub pulses to be added to the nominal number of pulses. PD_S is a signed integer value using 16 bits: up to +/-32768 pulses can be added for each increment.

The NS value of the profile ADT_S has the value 1, when a nominal increment is assumed. An integer number greater than 1 shows the number of nominal increments to be considered for a gap. For the actual increment after synchronization the corresponding NS value is stored in SYN_S of the NUSC register.

For references above the following hints are used:

[1] see DPLL_CTRL_x register, x=2,3,4; see 18.12.3,18.12.4,18.12.5

[2] see DPLL_STATUS register; see 18.12.30

[3] see DPLL_ACT_STA register; see 18.12.7

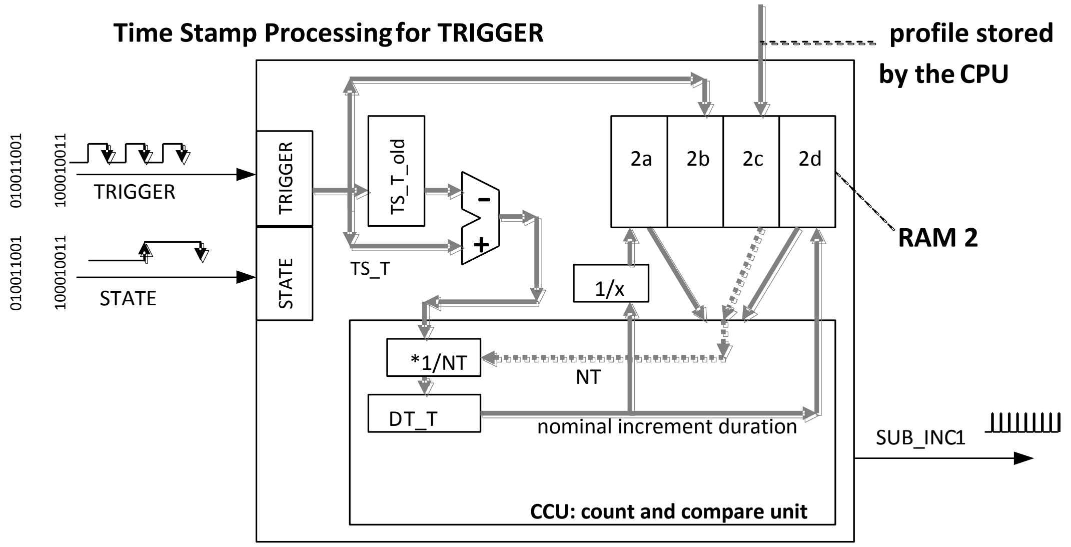
[4] see DPLL_CTRL_0 register; see 18.12.1

[5] see DPLL_CTRL_1 register; see 18.12.2

[6] see DPLL input signal description; see 18.1

18.5.9 Block diagrams of time stamp processing.

18.5.9.1 Time Stamp Processing Trigger

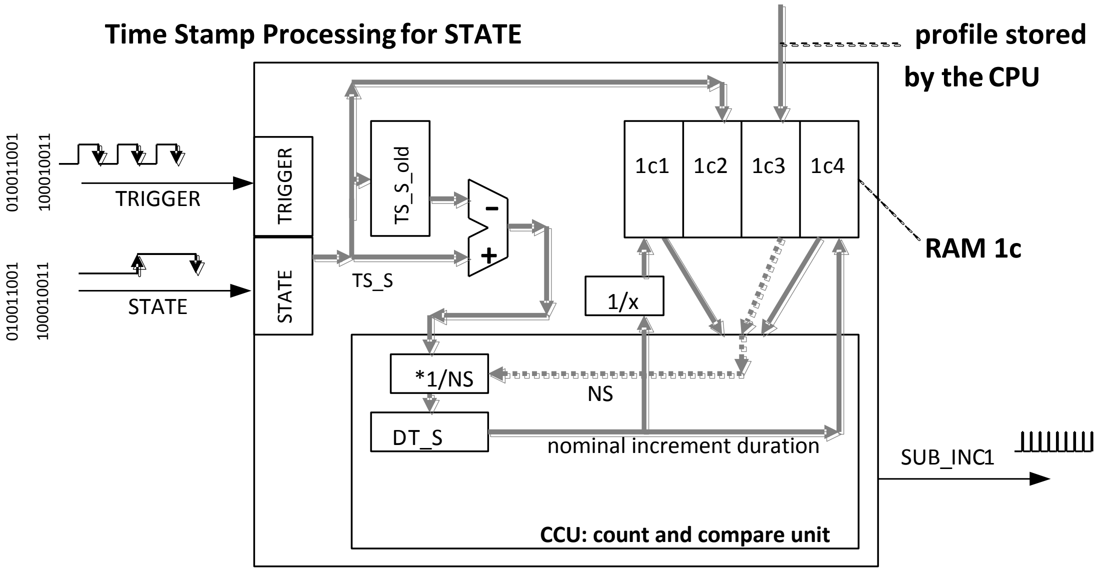


As shown in the block diagram above the time stamp difference of two succeeding input events is calculated. For the prediction of the current increment duration such values from the past are used. For this purpose the measured and calculated values of the last FULL_SCALE period are stored in the RAM. For the *TRIGGER* input there are 4 different RAM parts in the RAM region 2:

2a	stores the reciprocals of each nominal increment duration RDT_T
2b	stores the time stamps of each active input event TSF_T
2c	is used for the profile ADT_T and
2d	for the nominal increment duration DT_T.

Because the prediction is based on the relations of increments in the past this relation can be calculated easily by the multiplication of increment duration values with the reciprocal value of another increment. In order not to be forced to distinguish between gaps and "normal" increments duration also for gaps only the nominal duration and the correspondent reciprocal values are stored in the RAM field. This is possible by consideration of the NT value in the profile: the measured increment duration is divided by NT.

18.5.9.2 Time Stamp Processing State



For the *STATE* input there are also 4 different RAM parts in the RAM region 1c:

1c1	stores the reciprocals of each nominal increment duration RDT_S
1c2	stores the time stamps of each active input event TSF_S
1c3	is used for the profile ADT_S and
1c4	for the nominal increment duration DT_S.

The calculations are performed similar as for the *TRIGGER* input. The NS value in the profile shows the appearance of a gap.

18.5.10 Register and RAM address overview

The address map of the DPLL is divided into register and memory regions as defined in Table 18.5.10.1. The addresses from 0x0000 to 0x00FC are reserved for registers, from 0x0100 to 0x01FC is reserved for action registers to serve the ARU at immediately read request.
The RAM is divided into 3 independent accessible parts 1a, 1b+c and 2.
The part 1a from 0x0200 to 0x037C is used for PMTR values got from ARU and intermediate calculation values; there is no write access from the CPU possible, while the DPLL is enabled.
The RAM 1b part from 0x0400 to 0x05FC is reserved for RAM variables and the RAM part 1c from 0x0600 to 0x09FC is used for the *STATE* signal values.
The RAM region 2 from 0x4000 to 0x7FFC is reserved for the *TRIGGER* signal values. RAM region 1a has a size of 288 bytes, Ram 1b+c uses 1,125 Kbytes while RAM region 2 is configurable from 1,5 to 12 Kbytes, depending on the number of *TRIGGER* events in FULL_SCALE. The AOSV_2 register is used to determine the beginning of each part.

The table in 18.5.10.1 gives the DPLL Address map overview
Registers are used to control the DPLL and to show its status. Also parameters are stored in registers when useful. The table below shows the addresses for status and

control registers as well as values stored in additional registers. The register meaning explained in the register overview (chapter18.11) while the bit positions of the status and control registers are described in detail in chapter 18.12.

Time stamps for TRIGGER and *STATE* can have either the same resolution as the TBU_TS0 input or 8 times higher. This is configured in the DPLL_CTRL_1 register (see chapter 18.12.2). While the TBU_TS0 is used for action predictions the higher resolution of *TRIGGER* and *STATE* inputs can be used for a more accurate pulse generation.

The time stamp fields of *TRIGGER* and *STATE* are stored in the corresponding RAM regions in such a way, that for a gap also entries for the virtual increments are provided. This is due to the necessity to calculate time differences between a given number of (real and virtual) input events independent of a gap. Therefore the gap is extended in the RAM fields 2b and 1c2.
For all other RAM regions in RAM 2 and RAM 1c the gap is considered as one increment.

For the access to the RAM fields there must be address pointers. When the device starts all address pointers have a zero value and the first measured and calculated values are stored in the beginning of the corresponding RAM field.
Because the position of the device is usually unknown at the beginning no profile information can be used. The profile regions must have their own address pointers each which are set by the CPU as soon as the position is known. By setting the appropriate value to the address pointer APT_2C of the *TRIGGER* profile or APS_1C3 of the *STATE* profile respectively the synchronization bits in the DPLL_STATUS register SYT or SYS are set respectively. In the following the gap information can be used.

Because the time stamp fields are extended at the gaps there must be additional address pointers for these regions: APT_2B for *TRIGGER* time stamps and APS_1C2 for *STATE* time stamps. These address pointers must be incremented by NT or NS respectively when a gap appears.

18.5.10.1 Register and RAM address map

Addr. range Start	Addr. range End	Value number	Byte #	Content	Indication	Region	RAM size
0x0000	0x0FC	64	256	Register	used/reserved	0	no RAM

0x100	0x1FC	64	192	ACTION registers	direct read from ARU	0	no RAM
0x0200	0x03FC	128	384	PMTR values RAM 1a	CPU R/Pw access, when DPLL disabled; ARU has highest priority	1a with own ports	RAM part 1a: 384 bytes
0x0400	0x05FC	128	384	Variables RAM 1b	R and monitored W access by the CPU	1b	RAM part 1b+c: 1,125 Kbytes
0x0600	0x09FC	256	768	*STATE* data	R and monitored W access by the CPU	**1c**	
0x0600	0x06FC	64	192	RDT_S[i]	*STATE* reciprocal values	1c1	
0x0700	0x07FC	64	192	TSF_S[i]	*STATE* TS values	1c2	
0x0800	0x08FC	64	192	ADT_S[i]	adapted values of *STATE*	1c3	
0x0900	0x09FC	64	192	DT_S[i]	nom. STATE inc.	1c4	
0x4000	0x47FC ... 0x7FFC	512 ... 4096	1536 ... 12288	*TRIGGER* data	R and monitored W access of CPU	**2**	RAM part 2: 1,5... 12 Kbytes
0x4000	0x41FC...4FFC	128... 1024	384...3072	RDT_T[i]	*TRIGGER* reciprocal values	2a	
0x4200...5000	0x43FC...5FFC	128... 1024	384...3072	TSF_T[i]	*TRIGGER* TS values	2b	

0x4400...6000	0x45FC...6FFC	128...1024	384...3072	ADT_T[i]	adapted values of *TRIGGER*	2c	
0x4600...7000	0x47FC...7FFC	128...1024	384...3072	DT_T[i]	nom. *TRIGGER* increments	2d	

18.5.10.2 RAM Region 1

RAM region 1 has a size of 1,5 Kbytes and is used to store variables and parameters as well as the measured and calculated values for increments of *STATE*. The RAM 1 region is divided into two independent accessible RAM parts (a and b+c) with own ports. The address information is shown in the table above and the detailed description is performed in the following chapters. The RAM 1a is used to store the PMTR values got from ARU and in addition some intermediate calculation results of actions. RAM region 1b is used for variables needed for the prediction of increments, while RAM 1c is used to store time stamps, profile and duration of all the *STATE* inputs of the last FULL_SCALE region. All variables and values of RAM 1b+c part use a data width of up to 24 bits.

The RAM is to be initialized by the DPLL after HW-reset. All RAM cells must have a zero value after performing the initialize procedure. This is performed when setting The Init_RAM bit in the DPLL_RAM_INI register. The DPLL is only available after finishing this procedure. The initialization progress is shown in the status bits of the same register.

- **RAM Region 1a:** used for storage of PMTR values got from ARU; read and write access by the CPU is only possible, when the DPLL is disabled. The CPU Address range: 0x0200 – 0x03FC

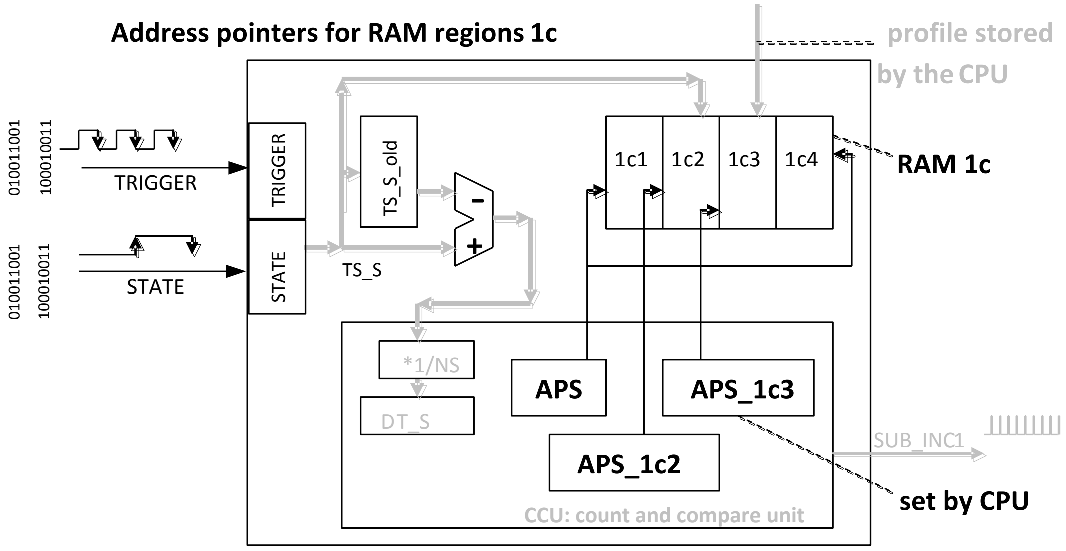
- **RAM Region 1b:** usable for intermediate calculations and auxiliary values, data width of 3 bytes used for 24 bit values; A write access to this region results in an interrupt to the CPU, when enabled. Address range: 0x0400 – 0x05FC

- **RAM Region 1c**: Values of all *STATE* increments in FULL_SCALE, data width of 3 bytes used for 24 bit values; A write access to this region results in an interrupt to the CPU, when enabled. Address range: 0x0600 – 0x09FC

In RAM region 1c there is a difference in the amount of data. While for the RAM regions 1c1, 1c3 and 1c4 there are **2*(SNU+1-SYN_NS)** entries for SYSF=0 or **2*(SNU+1) - SYN_NS** entries for SYSF=1, for the RAM region 1c2 there are **2*(SNU+1)** entries (see DPLL_CTRL_0 and _1 registers). For the latter also the virtual events are considered, that means the gap is divided into equidistant parts each having the same position share as increments without a gap. For that reason the CPU must extend the stored TSF_S[i] values in the RAM region 1c2 before the APS_1C3 is written. The write access to APS_1C3 sets the SYS bit in the DPLL_Status register in order to show the end of the synchronization process. Only when the SYS bit is set the PMTR values can consider more than the last increment duration for the action prediction by setting NUSE to a corresponding value.

Note: RAM regions 1b and 1c have a common port.

18.5.10.2.1 Address Pointer for RAM 1c



The address pointers for RAM region 1c are shown in the diagram above. While the address pointer APS points to the RAM regions 1c1 and 1c4, the address pointer APS_1C2 points to the time stamp field in the region 1c2. This is necessary, because in the time stamp field the gaps are extended to the number of nominal increments (see explanation above and also to the synchronization procedure explained in chapter 18.8.6.2).

The address pointer APS_1C3 is set by the CPU when the position is known and therefore the relation to the other address pointers is calculated. This setting of this profile address pointer synchronizes the RAM fields to one another. The synchronization is shown in the DPLL_STATUS register (see chapter 18.12.30) by the SYS bit.

18.5.10.3 RAM Region 2

The RAM region 2 has a configurable size of 1,5 to 12 Kbytes and is used to store measured and calculated values for increments of *TRIGGER*. The address information is explained in chapter 18.11 while the meaning is explained in this chapter.

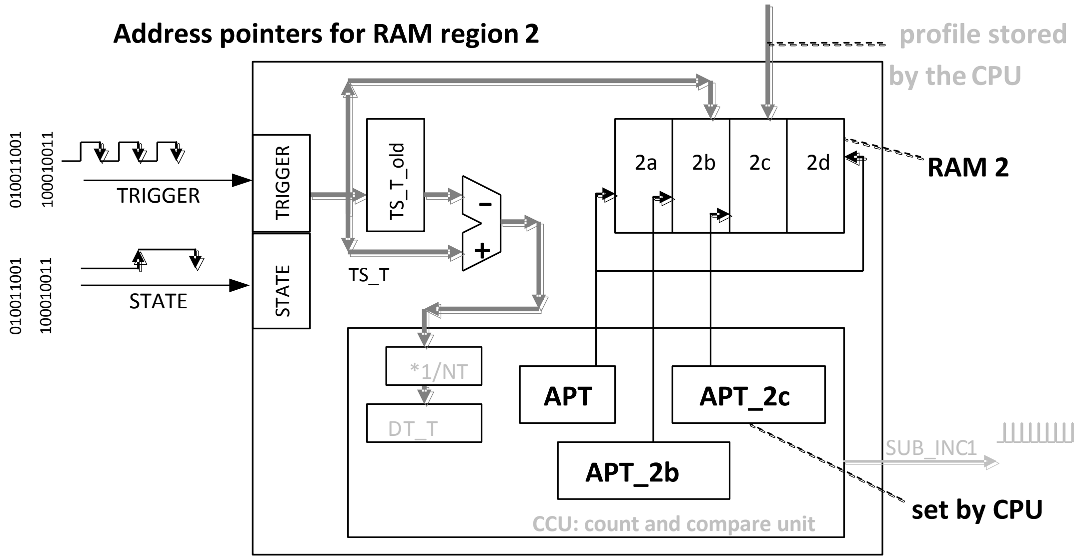
Because of up to 512 *TRIGGER* events in HALF_SCALE the fields 2a, b c and d must have up to 1024 storage places each. For 3 Bytes word size this does mean up to 12 k Byte of RAM region 2.

In order to save RAM size for configurations with less *TRIGGER* events the RAM is configurable by the offset switch Register OSW (0x001C) and the address offset value register of RAM region 2 AOSV_2 (0x0020). The RAM is to be initialized by the DPLL after HW-reset. All RAM cells must have a zero value after performing the initialize procedure. The DPLL is only available after finishing this procedure.

In RAM region 2 there is a difference in the amount of data. While for the RAM regions 2a, 2c and 2d there are **2*(TNU+1-SYN_NT)** entries, for the RAM region 2b there are **2*(TNU+1)** entries (see DPLL_CTRL_0 and _1 registers). For the latter also the virtual events are considered, that means the gap is divided into equidistant parts each having the same position share as increments without a gap. For that reason the CPU must extend the stored TSF_T[i] values in the RAM region 2b before the APT_2C is written.

The write access to APT_2C sets the SYT bit in the DPLL_Status register in order to show the end of the synchronization process. Only when the SYT bit is set the PMTR values can consider more than the last increment duration for the action prediction by setting NUTE to a value greater than one.

18.5.10.3.1 Address Pointer for RAM 2



The address pointers for RAM region 2 are shown in the diagram above. While the address pointer APT points to the RAM regions 2a and 2d, the address pointer APT_2B points to the time stamp field in the region 2b. This is necessary, because in the time stamp field the gaps are extended to the number of nominal increments (see explanation above and also to the synchronization procedure explained in chapter 18.8.6.2).
The address pointer APT_2C is set by the CPU when the position is known and therefore the relation to the other address pointers is calculated. This setting of this profile address pointer synchronizes the RAM fields to one another. The synchronization is shown in the DPLL_STATUS register (see chapter 18.12.30) by the SYT bit.

18.5.11 Software reset and DPLL deactivation

The DPLL module allows different options of deactivation and/or reset.
To stop the operation of the DPLL module it is possible to deactivate the DPLL by setting of DPLL_CTRL_1.DEN = 0 . This stops the calculations for the generation of the sub increments and the actions. Some control register areas are only configurable in this mode, some but not all register signals are set into an initial state. The RAM memory is not affected by DPLL deactivation at all. The behavior of the DPLL output signals and registers when deactivated is described in this document.

The deeper option to reconfigure the DPLL is the use of the software reset. When the DPLL module is deactivated setting DPLL_CTRL_1.SWR = 1 performs a reset of all DPLL registers and state controllers. The RAM memory is not affected by the software reset at all. After the software reset the DPLL module remains in deactivated state and the control registers must be configured again before operation (activation by DEN = 1) can start again.

The RAM modules can be reset (written to all zero) by activation of the memory init control bit (INIT_RAM) of the register DPLL_RAM_INI.
If the RAM initialization is automatically done after power on reset or not depends on the GTM implementation.

A special case is the configuration of the control bit DPLL_CTRL_11.STATE_EXT. If this bit shall be modified during operation a software reset of the DPLL module is strongly recommended. A RAM initialisation should also be considered depending on the given application case.

18.6 Prediction of the current increment duration

18.6.1 The use of increments in the past

Past values to be considered for the prediction of TRIGGER
In order to take into account values of increments for *TRIGGER*s in the past, the NUTE value is configured to determine the number of past values. In addition the VTN has a value according to the number of virtual increments in the NUTE region. Because gaps come in to the NUTE region or leave it the VTN value must be updated by the CPU until NUTE is set to HALF_SCALE or FULL_SCALE. For the RAM regions 2a and 2d the value NUTE-VTN is to be considered while for the RAM region 2b only the NUTE value is to be considered. This is due to the fact that the time stamp entries in a gap are extended to the number of nominal increments, but duration entries not.

Past values to be considered for the prediction of STATE
In order to take into account values of increments for *STATE* in the past, the NUSE value is configured to determine the number of past values. In addition the VSN has a value according to the number of virtual increments in the NUSE region. Because gaps come in to the NUSE region or leave it the VSN value must be updated by the CPU until NUSE is set to HALF_SCALE or FULL_SCALE. For the RAM regions 1c1 and 1c4 in the past the value NUSE-VSN is to be considered while for the RAM region 1c2 only the NUSE value is to be considered. This is due to the fact that time stamp entries in a gap are extended to the number of nominal increments, but duration entries not.

18.6.2 Increment prediction in Normal Mode and for first PMSM forwards

For the prediction of increments and actions in normal mode the values are calculated as described in the following equations.

Please note, that the ascending order of calculation must be hold in order not to lose results still needed. It is important for *TRIGGER* values to calculate and store in the RAM region 2 all values according to equations up to DPLL-14 before DPLL-1a4...7, DPLL-1b1 and DPLL-1c1, while the last one overwrites DT_T[i] when NUTE (see chapter 18.12.14) is set to the FULL_SCALE range. Because the old value of DT_T[i] is also needed for equation DPLL-10 and DPLL-11 this value is stored temporarily at DT_T_ACT as shown by equation DPLL-1a or DPLL-1b respectively until all prediction calculations are done and after that equation DPLL-1a4...7, DPLL-1b1 and DPLL-1b1 updates DT_T[i]: update DT_T[i] after calculations of equation DPLL-14. For p=APT calculates in normal mode.

When using filter information of TRIGGER_FT, selected by IDT=1, it must be distinguished by IFP, if this filter information is time or position related.
In order to make possible to perform the automatic resolution corrections of equations DPLL-1a1a the filter unit in TIM module must operate using the time stamp clock.

18.6.2.1 Equations DPLL-1a to calculate TRIGGER time stamps

For calculation of time stamps use the filter delay information and an additional TRIGGER input delay value stored in register TIDEL (initial zero)

TS_T_1= TRIGGER_TS - TIDEL (DPLL-1a0)
TS_T= TS_T_1 - FTV_Tx (for IDT=1 and IFP=0) (DPLL-1a1)
with
FTV_Tx = FTV_T/8 (for LOW_RES = 1 and TS0_HRT = 0) (DPLL-1a1a)
FTV_Tx = FTV_T (for LOW_RES = 0 or TS0_HRT = 1) (DPLL-1a1b)
and
TS_T= TS_T_1 - FTV_T *(CDT_TX/NMB_T)_old[10] for (IDT=1 and IFP=1) (DPLL-1a2)

this can be also calculated using the value of ADD_IN_CALN:

TS_T= TS_T_1 - FTV_T *(1/ADD_IN_CALN_old[10]) for (IDT=1 and IFP=1) (DPLL-1a3)

[10] Consider values, calculated for the last increment; position related filter values are only considered up to at least 1 ms time between two *TRIGGER* events. The reciprocal value is stored using a 32 bit fractional part, while only the 24 lower bits are used - for explanation see note [4] at DPLL_CTRL_0 register. The value of 1/ADD_IN_CALN_old or (CDT_TX/NMB_T)_old is set to 0xFFFFFF in the case of an overflow.

NOTE: CDT_TX is the predicted duration of the last *TRIGGER* increment and NMB_T the calculated number of SUB_INC1 events in the last increment, because the new calculations are done by equations DPLL-5 and DPLL-21 for the current increment after that. Therefore in equation DPLL-1a3 the value ADD_IN of the last increment is used (see equation DPLL-25). SYN_T_OLD is the number of TRIGGER events including missing TRIGGERs as specified in the NUTC register for the last increment, with the initial value of 1.

For storage of time stamps in the RAM see also equations DPLL-1a4 ff. after calculation of actions, chapter 18.7.5.1.

18.6.2.2 Equation DPLL-1b to calculate DT_T_ACT (nominal value)

DT_T_ACT= (TS_T - TS_T_OLD)/SYN_T_OLD (DPLL-1b)

For the case SYT=0 (still no synchronization to the profile) the values SYN_T and SYN_T_OLD are still assumed as having the value 1.

Correct the current increment duration value got by equation DPLL-1b in the case of physical deviations (ADT=1) by
DT_T_ACT = DT_T_ACT * (1 - PDC_T + PDC_T^2 -PDC_T^3)
(DPLL-1b1)
with the relative correction value for the last increment (for SMC=0)
PDC_T = PD_OLD / (MLT+1) (DPLL-1b2)
for SMC=1 use
PDC_T = PD_OLD / (MLS1) (DPLL-1b3)
Note: The term (1 - PDC_T + PDC_T^2 -PDC_T^3) is representing the third order Taylor series of the term 1/(1+PDC_T), which is chosen to reduce the additional time delay due to the more complex computation.

18.6.2.3 Equation DPLL-1c to calculate RDT_T_ACT (nominal value)

RDT_T_ACT = 1 / DT_T_ACT (DPLL-1c)

18.6.2.4 Equation DPLL-2a1 to calculate QDT_T_ACT

Relation of the recent last two increment values for APT=p in forward direction (DIR1=0)
QDT_T_ACT = DT_T_ACT * RDT_T[p-1] (DPLL-2a1)
QDT_T_ACT as well as QDT_T[i] have a 24 bit value using a 6 bit integer part and an 18 bit fractional part.

Note: QDT_T_ACT uses a 6 bit integer part and an 18 bit fractional part.

18.6.2.5 Equation DPLL-3 to calculate the error of last prediction

When q = NUTE-VTN considers for the error calculation only the last valid prediction values for DIR1=0:
Calculate the error of the last prediction when using only RDT_T_FS1, DT_T[p-q], DT_T[p-q-1] and DT_T[p-1] for the prediction of DT_T[p]:

EDT_T = DT_T_ACT - (DT_T[p-1] * QDT_T[p-q]) (DPLL-3)
with
QDT_T[p-q] = DT_T[p-q] * RDT_T[p-q-1] for FST=0 (DPLL-2b1)

QDT_T[p-q] = DT_T[p-q] * RDT_T_FS1 for FST=1 (DPLL-2b2)
and FST has the meaning: NUTE=FULL_SCALE (see NUTC register)
while
QDT_T_ACT as well as QDT_T[i] have a 24 bit value using a 6 bit integer part and an 18 bit fractional part.

Note: QDT_T[p-q] uses a 6 bit integer part and an 18 bit fractional part.

18.6.2.6 Equation DPLL-4 to calculate the weighted average error

for SYT=1 calculate:

MEDT_T := (EDT_T + MEDT_T) / 2 (DPLL-4)

18.6.2.7 Equations DPLL-5 to calculate the current increment value

nominal increment value (for ADT=0):
CDT_TX _nom = (DT_T_ACT + MEDT_T) * QDT_T[p-q+1] (DPLL-5a1)
nominal increment value (for ADT=1):
CDT_TX _nom_corr = CDT_TX _nom * (1 + CDC_T)
(DPLL-5a2)
with for SMC=0
CDC_T = PD / (MLT+1) (DPLL-5a3)
or for SMC=1 use
CDC_T = PD / (MLS1) (DPLL-5a4)

and with (for q>1):
QDT_T[p-q+1] = DT_T[p-q+1] * RDT_T[p-q] (DPLL-2c)
and for q=1 use equation DPLL-2a1.

while
QDT_T_ACT as well as QDT_T[i] have a 24 bit value using a 6 bit integer part and an 18 bit fractional part.

The CDT_TX_nom value is limited by the relation
CTN_MIN < CDT_TX_nom < CTN_MAX (DPLL-5c)
When the calculated value exceeds one of the limits, it is replaced by the corresponding limit value.

The expected duration to the next *TRIGGER* event is
(for ADT=0):

CDT_TX = CDT_TX _nom * SYN_T
(for ADT=1):
CDT_TX = CDT_TX _nom_corr * SYN_T (DPLL-5b)

Note: QDT_T[p-q+1] uses a 6 bit integer part and an 18 bit fractional part.
Note: In the case of an overflow in equations DPLL-5a or b set the value to 0xFFFFFF and the corresponding CTO bit in the DPLL_STATUS register. In the case of negative values set CDT_TX to 0x0 without any effect to the CTO bit.

18.6.3 Increment prediction in Emergency Mode and for second PMSM forwards

Please note, that the ascending order of calculations for *STATE* and storage of the values in the RAM region 1c must be hold in order not to lose results still needed. The same considerations as done for DT_T_ACT are valid for DT_S_ACT (equation DPLL-6a4...7, DPLL-6b1 and DPLL-6b1): update TD_S[i] only after calculations of equation DPLL-14.

When using filter information of STATE_FT, selected by IDS=1, it must be distinguished by IFP, if this filter information is time or position related.
In order to make possible to perform the automatic resolution corrections of equations DPLL-6a1a the filter unit in TIM must operate using the time stamp clock.

18.6.3.1 Equations DPLL-6a to calculate STATE time stamps

For calculation of time stamps use the filter delay information, the additional STATE input delay value stored in the register SIDEL (initial zero) and use p=APS while DIR2=0:

TS_S_1 = STATE_TS - SIDEL (DPLL-6a0)
TS_S = TS_S_1 - FTV_Sx (for IDS=1 and IFP=0) (DPLL-6a1)
with
FTV_Sx = FTV_S / 8 (for LOW_RES = 1 and TS0_HRS = 0) (DPLL-6a1a)
FTV_Sx = FTV_S (for LOW_RES = 0 or TS0_HRS = 1) (DPLL-6a1b)
and
TS_S = TS_S_1 - FTV_S * (CDT_SX / NMB_S)_old[10) (for IDS=1 and IFP=1) (DPLL-6a2)

this can be also calculated using the value of ADD_IN_CALE:
TS_S = TS_S_1 - FTV_S * (1 / ADD_IN_CALE)_old[10) (for IDS=1 and IFP=1) (DPLL-6a3)
with
see also equations DPLL-6a4 ff. at chapter 18.6.2 for *TRIGGER*.

[10] Consider values, calculated for the last increment; position related filter values are only considered up to at least 1 ms time between two *STATE* events. The reciprocal value is stored using a 32 bit fractional part, while only the 24 lower bits are used - for explanation see note [4] at DPLL_CTRL_0 register. The value of 1/ADD_IN_CALE_old or (CDT_SX/NMB_S)_old is set to 0xFFFFFF in the case of an overflow.

Note: CDT_SX is the predicted duration of the last *STATE* increment and NMB_S the calculated number of SUB_INC1 events in the last increment, because the new calculations are done by equations DPLL-10 and DPLL-22 respectively for the current increment after that. Therefore in equation DPLL-6a3 the value ADD_IN of the last increment is used (see equation DPLL-26). SYN_S_OLD is the number of increments including missing *STATE*s as specified in the **NUSC** register for the last increment with the initial value of 1. The update to the RAM region 1c4 is done after all related calculations (see equation DPLL-6b1 for this reason).

18.6.3.2 Equation DPLL-6b to calculate DT_S_ACT (nominal value)

DT_S_ACT= (TS_S - TS_S_OLD) / SYN_S_OLD (DPLL-6b)

For the case SYS=0 (still no synchronization to the profile) the values SYN_S and SYN_S_OLD are still assumed as having the value 1.

Correct the current increment duration value got by equation DPLL-6b in the case of physical deviations (ADS=1) by

DT_S_ACT = DT_S_ACT * (1 - PDC_S + PDC_S^2 - PDC_S^3)
(DPLL-6b1)

with the relative correction value for the last increment (for SMC=0)

PDC_S = PD_S_OLD/(MLS1) (DPLL-6b2)

for SMC=1 use

PDC_S = PD_S_OLD/(MLS2) (DPLL-6b3)

Note: The term (1 - PDC_S + PDC_S^2 -PDC_S^3) is representing the third order Taylor series of the term 1/(1+PDC_S), which is chosen to reduce the additional time delay due to the more complex computation.

18.6.3.3 Equation DPLL-6c to calculate RDT_S_ACT (nominal value)

RDT_S_ACT = 1 / DT_S_ACT (DPLL-6c)

18.6.3.4 Equation DPLL-7a1 to calculate QDT_S_ACT

for APS=p in forward direction (DIR2=0)

QDT_S_ACT = DT_S_ACT * RDT_S[p-1] (DPLL-7a1)

QDT_S_ACT as well as QDT_S[i] have a 24 bit value using a 6 bit integer part and an 18 bit fractional part.

Note: QDT_S_ACT uses a 6 bit integer part and an 18 bit fractional part.

18.6.3.5 Equation DPLL-8 to calculate the error of last prediction

with q= NUSE-VSN when using QDT_S[p-q] and DT_S[p-1] for the prediction of DT_S[p]

EDT_S = DT_S_ACT - (DT_S[p-1] * QDT_S[p-q]) (DPLL-8)
and with
QDT_S[p-q] = DT_S[p-q] * RDT_S[p-q-1] for FSS=0 (DPLL-7b1)
QDT_S[p-q] = DT_S[p-q] * RDT_S_FS1   for FSS=1 (DPLL-7b2)
and FSS has the meaning: NUSE=FULL_SCALE (see NUSC register)

QDT_S_ACT as well as QDT_S[i] have a 24 bit value using a 6 bit integer part and an 18 bit fractional part.

Note: QDT_S[p-q] uses a 6 bit integer part and an 18 bit fractional part.

18.6.3.6 Equation DPLL-9 to calculate the weighted average error

for SYS=1 calculate:

MEDT_S := (EDT_S + MEDT_S) / 2 (DPLL-9)

18.6.3.7 Equations DPLL-10 to calculate the current increment (nominal value)

nominal increment value (for ADS=0):
CDT_SX _nom = (DT_S_ACT + MEDT_S) * QDT_S[p-q+1] (DPLL-10a1)
or nominal increment value (for ADS=1):
CDT_SX _nom_corr = CDT_SX _nom * (1+CDC_S) (DPLL-10a2)
with for SMC=0
CDC_S = PD / (MLS1) (DPLL-10a3)
for SMC=1 use
CDC_S = PD / (MLS2) (DPLL-10a4)

and with
QDT_S[p-q+1] = DT_S[p-q+1] * RDT_S[p-q] (for q>1) (DPLL-7c)
and see equation DPLL-7a1 for q=1
while
QDT_S_ACT as well as QDT_S[i] have a 24 bit value using a 6 bit integer part and an 18 bit fractional part.

The CDT_SX_nom value is limited by the relation
CSN_MIN < CDT_SX_nom < CSN_MAX (DPLL-10c)
When the calculated value exceeds one of the limits, it is replaced by the corresponding limit value.

The expected duration to the next *STATE* event is
(for ADT=0):
CDT_SX = CDT_SX _nom * SYN_T
(for ADT=1):
CDT_SX = CDT_SX _nom_corr * SYN_S (DPLL-10b)

Note: QDT_S[p-q+1] uses a 6 bit integer part and an 18 bit fractional part.
Note: In the case of an overflow in equations DPLL-10a or b set the value to 0xFFFFFF and the corresponding CSO bit in the DPLL_STATUS register. In the case of negative values set CDT_SX to 0x0 without any effect to the CSO bit.
All 5 steps above (DPLL-6 to DPLL-10) are only needed in emergency mode. For the normal mode the calculations of equations DPLL-6 and DPLL-7 are done solely in order to get the values needed for a sudden switch to emergency mode.

18.6.4 Increment prediction in Normal Mode and for first PMSM backwards

18.6.4.1 Equations DPLL-2a2 to calculate QDT_T_ACT backwards

QDT_T_ACT = DT_T_ACT * RDT_T[p+1] (DPLL-2a2)

QDT_T_ACT as well as QDT_T[i] have a 24 bit value using a 6 bit integer part and an 18 bit fractional part.

18.6.4.2 Equation DPLL-3a to calculate of the error of last prediction

When q = NUTE-VTN and DIR1=1 using only QT_T[p+q] and DT_T[p+1] for the prediction of DT_T[p]

EDT_T = DT_T_ACT - (DT_T[p+1] * QDT_T[p+q] (DPLL-3a)
with
QDT_T[p+q] = DT_T[p+q] * RDT_T[p+q+1] for FST=0 (DPLL-2b3)
QDT_T[p+q] = DT_T[p+q] * RDT_T_FS1 for FST=1 (DPLL-2b4)
and FST has the meaning: NUTE=FULL_SCALE (see NUTC register)

QDT_T_ACT as well as QDT_T[i] have a 24 bit value using a 6 bit integer part and an 18 bit fractional part.

18.6.4.3 Equation DPLL-4 to calculate the weighted average error

For SYT=1 calculate:
MEDT_T := (EDT_T + MEDT_T) / 2 (DPLL-4)

18.6.4.4 Equation DPLL-5 to calculate the current increment value

nominal increment value (for ADT=0):
CDT_TX _nom = (DT_T_ACT + MEDT_T) * QDT_T[p+q-1] (DPLL-5a5)
nominal increment value (for ADT=1):
CDT_TX _nom_corr = CDT_TX _nom * (1 + CDC_T)
(DPLL-5a6)
with for SMC=0
CDC_T = PD / (MLT + 1) (DPLL-5a3)
for SMC=1 use
CDC_T = PD / (MLS1) (DPLL-5a4)

and with
QDT_T[p+q-1] = DT_T[p+q-1] * RDT_T[p+q] (for q>1) (DPLL-2c1)

for q=1 use equation DPLL-2a1.

while
QDT_T_ACT as well as QDT_T[i] have a 24 bit value using a 6 bit integer part and an 18 bit fractional part.

The CDT_TX_nom value is limited by the relation
CTN_MIN < CDT_TX_nom < CTN_MAX (DPLL-5c)
When the calculated value exceeds one of the limits, it is replaced by the corresponding limit value.

and the expected duration to the next *TRIGGER* event
(for ADT=0):
CDT_TX = CDT_TX _nom * SYN_T
(for ADT=1):
CDT_TX = CDT_TX _nom_corr * SYN_T (DPLL-5b)

Note: In the case of an overflow in equations DPLL-5a1 or b set the value to 0xFFFFFF and the corresponding CTO bit in the DPLL_STATUS register. In the case of negative values set CDT_TX(_nom) to 0x0.

18.6.5 Increment prediction in Emergency Mode and for second PMSM backwards

18.6.5.1 Equation DPLL-7a2 to calculate QDT_S_ACT backwards

QDT_S_ACT = DT_S_ACT * RDT_S[p+1] (DPLL-7a2)

QDT_S_ACT as well as QDT_S[i] have a 24 bit value using a 6 bit integer part and an 18 bit fractional part.

18.6.5.2 Equation DPLL-8a to calculate the error of the last prediction

While q= NUSE-VSN, use only QDT_S[p+q] and DT_S[p+1] for the prediction of DT_S[p]
EDT_S = DT_S_ACT - (DT_S[p+1] * QDT_S[p+q]) (DPLL-8a)
with
QDT_S[p-q] = DT_S[p+q] * RDT_S[p+q+1] for FSS=0 (DPLL-7b3)
QDT_S[p-q] = DT_T[p+q] * RDT_S_FS1 for FSS=1 (DPLL-7b4)
and FSS has the meaning: NUSE=FULL_SCALE (see NUSC register)

QDT_S_ACT as well as QDT_S[i] have a 24 bit value using a 6 bit integer part and an 18 bit fractional part.

18.6.5.3 Equation DPLL-9 to calculate the weighted average error

For SYS=1 calculate:
MEDT_S := (EDT_S + MEDT_S) / 2 (DPLL-9)

18.6.5.4 Equations DPLL-10 to calculate the current increment value

nominal increment value (for ADS=0):
CDT_SX _nom = (DT_S_ACT + MEDT_S) * QDT_S[p+q-1] (DPLL-10a5)
or nominal increment value (for ADS=1):
CDT_SX _nom_corr = CDT_SX _nom * (1+CDC_S) (DPLL-10a6)
with for SMC=0
CDC_S = PD / (MLS1) (DPLL-10a3)
for SMC=1 use
CDC_S = PD / (MLS2) (DPLL-10a4)

and with
QDT_S[p+q-1] = DT_S[p+q-1] * RDT_S[p+q] (for q>1) (DPLL-7c1)
for q=1 use equation DPLL-7a.

while
QDT_S_ACT as well as QDT_S[i] have a 24 bit value using a 6 bit integer part and an 18 bit fractional part.

The CDT_SX_nom value is limited by the relation
CSN_MIN < CDT_SX_nom < CSN_MAX (DPLL-10c)
When the calculated value exceeds one of the limits, it is replaced by the corresponding limit value.

and calculate the expected duration to the next *STATE* event
(for ADT=0):
CDT_SX = CDT_SX _nom * SYN_T
(for ADT=1):
CDT_SX = CDT_SX _nom_corr * SYN_S (DPLL-10b)

Note: In the case of an overflow in equations DPLL-10a1 or b set the value to 0xFFFFFF and the corresponding CSO bit in the DPLL_STATUS register. In the case of negative values set CDT_SX(_nom) to 0x0.
All 5 steps above (DPLL-6 to DPLL-10) are only needed in emergency mode. For the normal mode the calculations of equations DPLL-6 and DPLL-7 are done solely in order to get the values needed for a sudden switch to emergency mode.

18.7 Calculations for actions

As already shown for the calculation of the current interval by equations DPLL-1 to DPLL-10 for the prediction of actions a similar calculation is to be done, as shown by the equations DPLL-11. to DPLL-14. The calculation of actions is also needed when the DPLL is used for synchronous motor control applications (SMC=1, see DPLL_CTRL_1 register). For action prediction purposes the measured time periods of the past (one FULL_SCALE back, when the corresponding NUTE or NUSE values are set properly by the CPU) are used. The calculation can be explained by the following assumptions, which are considerably simple:

Take the corresponding increments for prediction in the past and put the sum of it in relation to the increment (DT_T[k], DT_S[k], with k >= 0, which is represented by the time stamp difference) which is exactly one FULL_SCALE period in the past (DPLL-11 or DPLL-13 respectively). Make a prediction for the coming sum of increments using the current measured increment (DT_T_ACT or DT_S_ACT respectively, that means DPLL-1 or DPLL-6 respectively) and add a weighted average error (DPLL-3 and DPLL-4 or DPLL-8 and DPLL-9 respectively, calculated for one increment prediction) before multiplication with the relation of equation DPLL-11 or DPLL-13 respectively in order to get the result as described by equations DPLL-12 or DPLL-14 respectively.

In order to avoid division operations instead of the increment (DT_T[k], DT_S[k], with k > 0) in the past its reciprocal value (RDT_T[k], RDT_S[k], with k > 0) is used, which is stored also in RAM. For the calculation of actions perform always a new refined calculation as long as the resulting time stamp is not in the past. In the other case the TSAC/PSAC values (time/position stamp of action calculated) is set to the time/position stamp of the last input event (TRIGGER/STATE), the ACT_N[i] bit in the DPLL_ACT_STA register is reset, while the corresponding ACT_N[i] bit in the DPLL_ACT_STA_shadow register is set. Each new PMTR_i value will set this ACT_N[i] bit again and reset the correspondent shadow bit until a new calculation is performed.

Please make sure that the prediction parameters are chosen such that under all conditions (acceleration/deceleration) the values of PDT_T, PDT_S and DTA respectively do not exceed the value 0xFFFFFF. This requirement can limit the predicted position range in the case of very low speed.

Action updates at highest speed

Up to 32 action values can be calculated. For the shortest increment duration (23,4 µs) not all of them can be updated with each active input event. Please notice the following conditions and parameters for an estimation of possible results.

All time estimation values are given for a system clock frequency of 100 MHz and the assumption, that the calculation of the DPLL is not impeded by a remote read or write access to the DPLL RAMs. Each RAM access is to be considered by an additional delay of about 40 ns ($t_{remote\_RAM\_acces}$, to be precised later). When using a different system clock frequency the calculation duration is extended accordingly.

1.) Typical time needed for basic operations (RAM update, pointer calculation and SUB_INC generation for normal, emergency mode or one PMSM:

t_{basic_0} = 9,9 µs.

2.) Typical time needed basic operations (RAM update, pointer calculation and SUB_INC generation for two PMSMs:

t_{basic_1} = 11,0 µs.

3.) Typical time needed to calculate one action

t_{action_i} = 3,7 µs.

Please notice that the above mentioned values are observed worst case values, when the two state machines of TRIGGER and STATE are both in operation.

These values allow the calculation of at least 3 action values for each input event for all specified increments duration. The complete time needed for the basic operation, n action calculations and k remote RAM access operations can be calculated as follows

$t_{complete} = t_{basic\_0/1} + n{*}t_{action\_i} + k{*}t_{remote\_RAM\_access}$.

Typical applications

Normal and emergency mode

For a typical application with the shortest increment duration of 100 µs in normal or emergency mode the calculation of up to 24 action values can be performed for each active input event.

One PMSM

For one PMSM and a typical shortest increment time of 39 µs there is the calculation of up to 7 action values possible for each input event.

Two PMSMs with restricted action calculations

When only one PMSM uses the action calculation service and the shortest increment duration is 39 µs, there can up to 7 actions served for each active input event.

Two PMSMs with unrestricted action calculations

When 2 PMSMs are used and both use the action calculation service at a minimal increment duration of 39 µs there are up to 7 action calculations possible for each of the two engines - that means up to 14 action calculations per increment in average.

18.7.1 Action calculations for TRIGGER forwards

valid for RMO=0 or for SMC=1 with

p=APT_2B, t=APT, m=NA[i] (part w), mb=NA[i](part b)/1024, NUTE-VTN=q, NUTE=n

Note: All 5 steps in equations DPLL-11 to DPLL-12 are only calculated in normal mode.

18.7.1.1 Equation DPLL-11a1 to calculate the time prediction for an action

For DIR1=0 and q>m calculate:

PDT_T[i] = (TSF_T[p+m-n] - TSF_T[p-n] +
mb* DT_Tx[t-q+1]) * RDT_T[t-q] (DPLL-11a1)
with
DT_Tx[t-q+1] = DT_T[t-q+1] for TS0_HRT=0 (DPLL-11b2)
or
DT_Tx[t-q+1] = DT_T[t-q+1]/8 for TS0_HRT=1 (DPLL-11b3)
and while the multiplication with mb does mean the fractional part of NA[i].

For SMC=0 and RMO=0 calculate for DIR1=0 all 32 actions in forward direction, if requested; in the case SMC=1 calculate up to 16 actions 0 to 15 in dependence of the *TRIGGER* input.

18.7.1.2 Equation DPLL-11a2 to calculate the time prediction for an action

For SYT=1, NUTE= 2*(TNU+1), q>m and DIR1=0 equation DPLL-11a2 is equal to

PDT_T[i] = (TSF_T[p+m] - TSF_T[p]+ mb* DT_Tx[t-q+1]) * RDT_T[t] (DPLL-11a2)
with

DT_Tx[t-q+1] = DT_T[t-q+1] for TS0_HRT=0 (DPLL-11b2)
or
DT_Tx[t-q+1] = DT_T[t-q+1]/8 for TS0_HRT=1 (DPLL-11b3)

18.7.1.3 Equation DPLL-11b to calculate the time prediction for an action

for DIR1=0, NUTE-VTN=q, q (< or =) m, n>1 and t=APT:

PDT_T[i] = (m+mb)*DT_Tx[t-q+1] * RDT_T[t-q] (DPLL-11b) with

DT_Tx[t-q+1] = DT_T[t-q+1] for TS0_HRT=0 (DPLL-11b2)
or
DT_Tx[t-q+1] = DT_T[t-q+1]/8 for TS0_HRT=1 (DPLL-11b3)

Note: Make the calculations above before updating the TSF_T[i] values according to equations DPLL-1c3 ff.

18.7.1.4 Equation DPLL-11c to calculate the time prediction for an action

for n=1 (this is always valid for SYT=0)

PDT_T[i] = (m+mb)* DT_T_ax* RDT_T[t-1] (DPLL-11c)
with

DT_T_ax = DT_T_ACT for TS0_HRT=0 (DPLL-1a4a)
or
DT_T_ax = DT_T_ACT/8 for TS0_HRT=1 (DPLL-1a4b)

Note: For the relevant last increment add the fractional part of DT_T_ACT as described in NA[i].

18.7.1.5 Equation DPLL-12 to calculate the duration value until action

DTA[i] = (DT_T_ACT + MEDT_T)* PDT_T[i] (DPLL-12)

18.7.2 Action calculations for TRIGGER backwards

valid for RMO=0 or for SMC=1 with
p=APT_2B, t=APT, m=NA[i] (part w), mb=NA[i](part b)/1024, q= NUTE-VTN and n=NUTE

For SMC=0 and RMO=0 calculate for DIR1=1 all 32 actions in backward direction for special purposes; in the case SMC=1 calculate up to 16 actions 0 to 15 in dependence of the *TRIGGER* input.

Note: All 5 steps in equations DPLL-11 to DPLL-12 are only calculated in normal mode or when SMC=1.

18.7.2.1 Equation DPLL-11a3 to calculate the time prediction for an action

For DIR1=1 and q>m calculate:
PDT_T[i] = (TSF_T[p-m+n] - TSF_T[p+n] +
mb* DT_Tx[t+q-1]) * RDT_T[t+q] (DPLL-11a3)
with
DT_Tx[t+q-1] = DT_T[t+q-1] for TS0_HRT=0 (DPLL-11b4)
or
DT_Tx[t+q-1] = DT_T[t+q-1]/8 for TS0_HRT=1 (DPLL-11b5)

18.7.2.2 Equation DPLL-11a4 to calculate the time prediction for an action

For SYT=1 and NUTE = 2*(TNU+1), q>m, VTN=2*SYN_NT and hence **NUTE-VTN = 2*(TNU+1-SYN_NT)** for DIR1=1 this is equal to
PDT_T[i] = (TSF_T[p-m] - TSF_T[p]+ mb* DT_Tx[t+q-1]) * RDT_T[t] (DPLL-11a4)
with
DT_Tx[t+q-1] = DT_T[t+q-1] for TS0_HRT=0 (DPLL-11b4)
or
DT_Tx[t+q-1] = DT_T[t+q-1]/8 for TS0_HRT=1 (DPLL-11b5)

Note: Make the calculations above before updating the TSF_T[i] values according to equations DPLL-1c3 ff.

18.7.2.3 Equation DPLL-11b1 to calculate the time prediction for an action

For NUTE-VTN =q, q (< or =) m the following equation is valid for n>1 and t=APT:

PDT_T[i] = (m+mb)*DT_Tx[t+q-1] * RDT_T[t+q] (DPLL-11b1)
with

DT_Tx[t+q-1] = DT_T[t+q-1] for TS0_HRT=0 (DPLL-11b4)
or
DT_Tx[t+q-1] = DT_T[t+q-1]/8 for TS0_HRT=1 (DPLL-11b5)

18.7.2.4 Equation DPLL-11c1 to calculate the time prediction for an action

for n=1 (this is always valid for SYT=0)

PDT_T[i] = (m+mb)* DT_T_ax * RDT_T[t+1] (DPLL-11c1)
with

DT_T_ax = DT_T_ACT for TS0_HRT=0 (DPLL-1a4a)
or
DT_T_ax = DT_T_ACT/8 for TS0_HRT=1 (DPLL-1a4b)

Note: For the relevant last increment add the fractional part of DT_T_ACT as described in NA[i].

18.7.2.5 Equation DPLL-12 to calculate the duration value for an action

DTA[i] = (DT_T_ACT + MEDT_T)* PDT_T[i] (DPLL-12)

Use the results of equations DPLL-1a, b, DPLL-3 and DPLL-4 for the above calculation

18.7.3 Action calculations for STATE forwards

valid for RMO=1 with
p=APS_1C2, t=APS, m=NA[i](part w) mb=NA[i](part b)/1024, NUSE-VSN = q and NUSE=n>m

For SMC=0 and RMO=1 calculate for DIR2=0 all 32 actions in forward direction, if requested; in the case SMC=1 and RMO=1 calculate up to 16 actions 16 to 31 in dependence of the *STATE* input.

Note: All 5 steps of equations DPLL-13 to DPLL-14 are only calculated in emergency mode or for SMC=1 in combination with RMO=1.

18.7.3.1 Equation DPLL-13a1 to calculate the time prediction for an action

For DIR2=0 and q>m calculate:
PDT_S[i] = (TSF_S[p+m-n] - TSF_S[p-n] +
mb* DT_Sx[t-q+1] * RDT_S[t-q] (DPLL-13a1)
with

DT_Sx[t-q+1]= DT_S[t-q+1] for TS0_HRS=0 (DPLL-13b2)
or
DT_Sx[t-q+1] = DT_S[t-q+1]/8 for TS0_HRS=1 (DPLL-13b3)

18.7.3.2 Equation DPLL-13a2 to calculate the time prediction for an action

For **SYS=1 and NUSE=2*(SNU+1)**, q>m, SYSF=0, VSN=2*SYN_NS and hence NUSE-VSN = 2*(SNU+1-SYN_NS) equation DPLL-13a1 is equal to

PDT_S[i] = (TSF_S[p+m] - TSF_S[p]+ mb*DT_Sx[t-q+1]) * RDT_S[t] (DPLL-13a2)
with

DT_Sx[t-q+1] = DT_S[t-q+1] for TS0_HRS=0 (DPLL-13b2)
or
DT_Sx[t-q+1] = DT_S[t-q+1]/8 for TS0_HRS=1 (DPLL-13b3)

18.7.3.3 Equation DPLL-13b to calculate the time prediction for an action

For NUSE -VTN=q, q (< or =) m and n>1:

PDT_S[i] = (m+mb)*DT_Sx[t-q+1] * RDT_S[t-q] (DPLL-13b)
with

DT_Sx[t-q+1] = DT_S[t-q+1] for TS0_HRS=0 (DPLL-13b2)
or
DT_Sx[t-q+1] = DT_S[t-q+1]/8 for TS0_HRS=1 (DPLL-13b3)

18.7.3.4 Equation DPLL-13c to calculate the time prediction for an action

for n=1

PDT_S[i] = (m+mb)* DT_S_ax * RDT_S[t-1] (DPLL-13c)
with

DT_S_ax = DT_S_ACT for TS0_HRS=0 (DPLL-6a4a)
or
DT_S_ax = DT_S_ACT/8 for TS0_HRS=1 (DPLL-6a4b)

18.7.3.5 Equation DPLL-14 to calculate the duration value for an action

DTA[i] = (DT_S_ACT + MEDT_S)* PDT_S[i] (DPLL-14)

Use the results of DPLL-7, DPLL-8 and DPLL-9 for the above calculation

18.7.4 Action calculations for STATE backwards

valid for RMO=1 with
p=APS_1C2, t=APS, m=NA[i](part w) mb=NA[i](part b)/1024, NUSE-VSN = q and NUSE=n

For SMC=0 and RMO=1 calculate for DIR1=1 all 32 actions in backwards mode for special purposes; in the case SMC=1 and RMO=1 calculate up to 16 actions 16 to 31 in dependence of the *STATE* input.

Note: All 5 steps of equations DPLL-13 to DPLL-14 are only calculated in emergency mode or for SMC=1 in combination with RMO=1.

18.7.4.1 Equation DPLL-13a3 to calculate the time prediction for an action

For (DIR2= 1 (SMC=1) or DIR1=1 (SMC=0)) and q>m calculate

PDT_S[i] = (TSF_S[p-m+n] - TSF_S[p+n] +
mb* DT_Sx[t+q-1]) * RDT_S[t+q] (DPLL-13a3)
with

DT_Sx[t+q-1]= DT_S[t+q-1] for TS0_HRS=0 (DPLL-13b4)
or
DT_Sx[t+q-1] = DT_S[t+q-1]/8 for TS0_HRS=1 (DPLL-13b5)

18.7.4.2 Equation DPLL-13a4 to calculate the time prediction for an action

For **SYS=1, NUSE=2*(SNU+1)**, q>m, SYSF=0, VSN=2*SYN_NS and hence NUSE-VSN = 2*(SNU+1-SYN_NS) equation DPLL-13a3 is equal to

PDT_S[i] = (TSF_S[p-m] - TSF_S[p]+ mb* DT_Sx[t+q-1]) * RDT_S[t] (DPLL-13a4)
with

DT_Sx[t+q-1]= DT_S[t+q-1] for TS0_HRS=0 (DPLL-13b4)
or
DT_Sx[t+q-1] = DT_S[t+q-1]/8 for TS0_HRS=1 (DPLL-13b5)

18.7.4.3 Equation DPLL-13b1 to calculate the time prediction for an action

For NUSE-VSN =q, q (< or =) m, NUSE=n and n>1:

PDT_S[i] = m*DT_Sx[t+q-1] * RDT_S[t+q] (DPLL-13b1)
with

DT_Sx[t+q-1] = DT_S[t+q-1] for TS0_HRS=0 (DPLL-13b4)
or
DT_Sx[t+q-1] = DT_S[t+q-1]/8 for TS0_HRS=1 (DPLL-13b5)

18.7.4.4 Equation DPLL-13c1 to calculate the time prediction for an action

for n=1

PDT_S[i] = (m+mb)* DT_S_ax * RDT_S[t+1] (DPLL-13c1)
with

DT_S_ax = DT_S_ACT for TS0_HRS=0 (DPLL-6a4a)
or
DT_S_ax = DT_S_ACT/8 for TS0_HRS=1 (DPLL-6a4b)

18.7.4.5 Equation DPLL-14 to calculate the duration value until action

DTA[i] = (DT_S_ACT + MEDT_S)* PDT_S[i] (DPLL-14)

Use the results of DPLL-7, DPLL-8 and DPLL-9 for the above calculation

18.7.5 Update of RAM in Normal and Emergency Mode

After considering the calculations for up to all 24 actions according to equations (DPLL-11, DPLL-12), only when going back to state 1 or 21 (because of a new TRIGGER or STATE event, that means when no further PMTR values are to be considered) set time stamp values and duration of increments in the RAM.

18.7.5.1 Equation DPLL-1a4 to update the time stamp values for TRIGGER

TSF_T[s]=TS_Tx (DPLL-1a4)
using the following equations for the determination of TS_Tx

For TS0_HRT=0:
TS_Tx=TS_T (DPLL-1a4w)
DT_T_ax = DT_T_ACT (DPLL-1a4a)

For TS0_HRT=1:
TS_Tx(20:0)=TS_T/8 (DPLL-1a4x)
TS_Tx(23:21)=TBU_TS0_T(23:21) (DPLL-1a4y)
        for TBU_TS0_T(20:0) > or = TS_Tx(20:0)
TS_Tx(23:21)=TBU_TS0_T(23:21) -1 (DPLL-1a4z)
        for TBU_TS0_T(20:0) < TS_Tx(20:0)
DT_T_ax = DT_T_ACT/8 (DPLL-1a4b)
Note: the combination of values LOW_RES=0 and TS0_HRT=1 is not possible.

Store the time stamp values in the time stamp field according to the address pointer APT_2B=s, but make this update only after the calculation of actions 18.7 because the old TSF_T[i] values are still needed for these calculations. Please note that the address pointer after a gap is still incremented by SYN_T_OLD in that case (see state machine step 1 in chapter 18.8.6).

18.7.5.2 Equation DPLL-1a5-7 to extend the time stamp values for TRIGGER in forward direction

when SYT=1 and SYN_T_OLD=r>1 and DIR1=0

TSF_T[s-1] = TSF_T[s] -DT_T_ax (DPLL-1a5)
TSF_T[s-2] = TSF_T[s-1] -DT_T_ax (DPLL-1a6)
until
TSF_T[s- r+1] = TSF_T[s- r+2] -DT_T_ax (DPLL-1a7)

after the incrementation of the pointer APT_2B by SYN_T_OLD

18.7.5.3 Equations DPLL-1a5-7 for backward direction

when SYT=1 and SYN_T_OLD=r>1 and DIR1=1

TSF_T[s+1] = TSF_T[s] -DT_T_ax (DPLL-1a5)
TSF_T[s+2] = TSF_T[s+1] -DT_T_ax (DPLL-1a6)
until
TSF_T[s+r-1] = TSF_T[s+r-2] -DT_T_ax (DPLL-1a7)

after the decrementation of the pointer APT_2B by SYN_T_OLD

18.7.5.4 Equations DPLL-1b1 and DPLL-1c1 to update the RAM after calculation

DT_T[p] = DT_T_ACT (DPLL-1b1)
save old reciprocal value from RAM before overwriting:
RDT_T_FS1= RDT_T[p] (DPLL-1c1)
after that store new value in RAM
RDT_T[p]= RDT_T_ACT (DPLL-1c2)

Store increment duration and reciprocal value in RAM region 2 in normal mode after calculation of actions only when a new active *TRIGGER* slope is detected and in emergency mode directly after calculation of DT_T_ACT or RDT_T_ACT respectively.

18.7.5.5 Equation DPLL-6a4 to update the time stamp values for STATE

TSF_S[s]=TS_Sx (DPLL-6a4)
using the following equations for the determination of TS_Sx

For TS0_HRS=0:
TS_Sx=TS_S (DPLL-6a4)
DT_S_ax = DT_S_ACT (DPLL-6a4a)

For TS0_HRS=1:
TS_Sx(20:0)=TS_S/8 (DPLL-6a4x)
TS_Sx(23:21)=TBU_TS0_S(23:21) (DPLL-6a4y)
for TBU_TS0_S(20:0) > or = TS_Sx(20:0)
TS_Sx(23:21)=TBU_TS0_S(23:21) -1 (DPLL-6a4z)
for TBU_TS0_S(20:0) < TS_Sx(20:0)
DT_S_ax = DT_S_ACT/8 (DPLL-6a4b)
Note: the combination of values LOW_RES=0 and TS0_HRS=1 is not possible.

Store the time stamp value in the time stamp field according to the address pointer APS_1C2=s, but make this update only after the calculation of actions (equations DPLL-13a2, 18.7.3.2 or DPLL-13a4 18.7.4.2, if applicable) because the old TSF_S[i] values are still needed for these calculations. Please note, that the address pointer after a gap is still incremented by SYN_S_OLD in that case (see state machine step 21 in chapter 18.8.6).

18.7.5.6 Equations DPLL-6a5-7 to extend the time stamp values for STATE

When SYS=1 and SYN_S_OLD=r>1 and DIR2=0 or DIR1=0 respectively calculate

TSF_S[s-1] = TSF_S[s] - DT_S_ax (DPLL-6a5)
TSF_S[s-2] = TSF_S[s-1] - DT_S_ax (DPLL-6a6)
until
TSF_S[s- r+1] = TSF_S[s- r+2] - DT_S_ax (DPLL-6a7)

after incrementation of the pointer APS_2b by SYN_S_OLD

18.7.5.7 Equations DPLL-6a5-7 for backward direction

When SYS=1 and SYN_S_OLD=r>1 and DIR2=1 or DIR1=1 respectively calculate

TSF_S[s+1] = TSF_S[s] - DT_S_ax (DPLL-6a5)
TSF_S[s+2] = TSF_S[s+1] - DT_S_ax (DPLL-6a6)
until
TSF_S[s+r-1] = TSF_S[s+r-2] - DT_S_ax (DPLL-6a7)

after the incrementation of the pointer APS_1C2 by SYN_S_OLD

18.7.5.8 *Equations DPLL-6b1 and DPLL-6c2 to update the RAM after calculation*

DT_S[p] = DT_S_ACT (DPLL-6b1)
save old reciprocal value from RAM before overwriting:
RDT_S_FS1= RDT_S[p] (DPLL-6c1)
after that store new value in RAM
RDT_S[p] = RDT_S_ACT (DPLL-6c2)

when a new active *STATE* slope is detected in emergency mode or in normal mode (SMC=RMO=0) directly after calculation of the values above.

Store increment duration and reciprocal value in RAM region 1c in emergency mode after calculation of actions only when a new active *STATE* slope is detected and in normal mode directly after calculation of DT_S_ACT or RDT_S_ACT respectively.

18.7.6 Time and position stamps for actions in Normal Mode

18.7.6.1 *Equation DPLL-15 to calculate the action time stamp*

TSAC[i]= DTA[i] - DLA[i] + TS_Tx (for DTA[i] > DLA[i] and
DTA[i] - DLA[i] < 0x800000) (DPLL-15a)
TSAC[i]= TS_Tx (for DTA[i] < DLA[i]) (DPLL-15b)
TSAC[i]= 0x7FFFFF + TS_Tx (for DTA[i] > DLA[i] and
DTA[i] - DLA[i] > 0x7FFFFF) (DPLL-15c)
Note: For TS_Tx see equations (DPLL-1a4 and following), chapter 18.7.5.1.

The calculation is done after the calculation of the current expected duration value according to equation DPLL-12 at chapter 18.7.2.5. The time stamp of the action can be calculated as shown above in equation DPLL-15 using the delay value of the action and the current time stamp.

18.7.6.2 *Equations DPLL-17 to calculate the position stamp forwards*

for **DIR1=0** and TS0_HRT=0:
PSAC[i] = PSA[i] - (DLA[i]*RCDT_TX_NOM)*(MLT+1) (DPLL-17)

with
RCDT_TX_NOM= (1/CDT_TX_NOM) * SYN_T (DPLL-17a)
and
RCDT_TX= 1/CDT_TX (DPLL-17b)

for **DIR1=0** and TS0_HRT=1:
PSAC[i] = PSA[i] - (8*DLA[i]*RCDT_TX_NOM)*(MLT+1) (DPLL-17d)

with
RCDT_TX_NOM= (1/CDT_TX_NOM) * SYN_T (DPLL-17a)
and
RCDT_TX= 1/CDT_TX (DPLL-17b)

replace (MLT+1) in equations (DPLL-17) and (DPLL-17d) by MLS1 for SMC=1

use the calculated value of (DPLL-17b) also for the generation of SUB_INCi and serve the action by transmission of TSAC[i] and PSAC[i] to ACT_D_i.
The action is to be updated for each new *TRIGGER* event until the calculated time stamp is in the past.
In this case the values of TSAC[i] and PSAC[i] depend on the DPLL_CTRL_11.ACBU signal.
When DPLL_CTRL_11.ACBU = '0': Use the time stamp of the last input event instead of the calculated value and the calculated position stamp of the actual increment as target position value.

When DPLL_CTRL_11.ACBU = '1':
For ACB_[z][1]= '1': is used as input signal to control if "action in past" shall be checked based on position information. If the position has reached "past" use the calculated position stamp of the actual increment as target position value.
For ACB_[z][1]= '0': In this case the PSAC[i] is used as calculated by the DPLL.

For ACB_[z][0]= '1': is used as input signal to control if "action in past" shall be checked based on time information. If the time has reached "past" use the time stamp of the last input event instead of the calculated TSAC[i] value.
For ACB_[z][0]= '0': In this case the TSAC[i] is used as calculated by the DPLL.

Set the corresponding shadow bit in the DPLL_ACT_STA register. Because of the blocking read operation the ACT_D values can be read only once.

18.7.6.3 Equations DPLL-17 to calculate the position stamp backwards

For **DIR1=1** and TS0_HRT=0:
PSAC[i] = PSA[i] + (DLA[i]*RCDT_TX_NOM)*(MLT+1) (DPLL-17c)

with
RCDT_TX_NOM= (1/CDT_TX_NOM) * SYN_T (DPLL-17a)
and
RCDT_TX= 1/CDT_TX (DPLL-17b)

For **DIR1=1** and TS0_HRT=1:
PSAC[i] = PSA[i] + (8*DLA[i]*RCDT_TX_NOM)*(MLT+1) (DPLL-17e)

with
RCDT_TX_NOM= (1/CDT_TX_NOM) * SYN_T (DPLL-17a)
and
RCDT_TX= 1/CDT_TX (DPLL-17b)

replace (MLT+1) in equations (DPLL-17c) and (DPLL-17e) by MLS1 for SMC=1

use the calculated value of (DPLL-17b) also for the generation of SUB_INCi and serve the action by transmission of TSAC[i] and PSAC[i] to ACT_D_i.
The action is to be updated for each new *TRIGGER* event until the calculated time stamp is in the past.
In this case the values of TSAC[i] and PSAC[i] depend on the DPLL_CTRL_11.ACBU signal.
When DPLL_CTRL_11.ACBU = '0': Use the time stamp of the last input event instead of the calculated value and the calculated position stamp of the actual increment as target position value.

When DPLL_CTRL_11.ACBU = '1':
For ACB_[z][1]= '1': is used as input signal to control if "action in past" shall be checked based on position information. If the position has reached "past" use the calculated position stamp of the actual increment as target position value.
For ACB_[z][1]= '0': In this case the PSAC[i] is used as calculated by the DPLL.

For ACB_[z][0]= '1': is used as input signal to control if "action in past" shall be checked based on time information. If the time has reached "past" use the time stamp of the last input event instead of the calculated TSAC[i] value.

For ACB_[z][0]= '0': In this case the TSAC[i] is used as calculated by the DPLL.

Set the corresponding shadow bit in the DPLL_ACT_STA register. Because of the blocking read operation the ACT_D values can be read only once.

18.7.7 The use of the RAM

The RAM is used to store the data of the last FULL_SCALE period. The use of single port RAMs is recommended. The data width of the RAM is usual 3 bytes, but could be extended to 4 bytes in future applications. There are 3 different RAMs, each with separate access ports. The RAM 1a is used to store the position minus time requests, got from the ARU. No CPU access is possible to this RAM during operation (when the DPLL is enabled).

Ram 1b is used for configuration parameters and variables needed for calculations. Within RAM 1c the values of the *STATE* events are stored. RAM 1b and RAM 1c do have a common access port and are also marked as RAM 1bc in order to clarify this fact.

RAM 2 is used for values of the *TRIGGER* events.

Because of the access of the DPLL internal state machine at the one side and the CPU at the other side the access priority has to be controlled for both RAMs 1bc and 2. The access priority is defined as stated below. The CPU access procedure via AE-interface goes in a wait state (waiting for data valid) while it needs a colliding RAM access during serving a corresponding state machine RAM access. In order not to provoke unexpected behavior of the algorithms the writing of the CPU to the RAM regions 1b, 1c or 2 will be monitored and results in interrupt requests when enabled.

CPU access is specified at follows:

1. CPU has highest priority for a single read/write access. The DPLL algorithm is stalled during external bus RAM accesses.
2. After serving the CPU access to the RAM the DPLL gets the highest RAM access priority for 8 clock cycles. Afterwards continue with 1.

The RAM address space has to be implemented in the address space of the CPU.

18.7.8 Time and position stamps for actions in Emergency Mode

18.7.8.1 Equation DPLL-18 to calculate the action time stamp

TSAC[i]= DTA[i] - DLA[i] + TS_Sx (for DTA[i] > DLA[i] and
DTA[i] - DLA[i] < 0x800000) (DPLL-18a)
TSAC[i]= TS_Sx (for DTA[i] < DLA[i]) (DPLL-18b)
TSAC[i]= 0x7FFFFF + TS_Sx (for DTA[i] > DLA[i] and
DTA[i] - DLA[i] > 0x7FFFFF) (DPLL-18c)

Note: For TS_Sx see equations (DPLL-6a4 and following), chapter 18.7.5.5.

The calculation is done after the calculation of the current expected duration value according to equation DPLL-14 at chapter 18.7.3.5. The time stamp of the action can be calculated as shown in equation DPLL-18 using the delay value of the action and the current time stamp.

18.7.8.2 Equations DPLL-20 to calculate the position stamp forwards

for **DIR2=0** or DIR1=0 respectively and TS0_HRS=0:
PSAC[i] = PSA[i] - (DLA[i]*RCDT_SX_NOM)*MLS1 (DPLL-20)

with
RCDT_SX_NOM= (1/CDT_SX_NOM) * SYN_S (DPLL-20a)
and
RCDT_SX= 1/CDT_SX (DPLL-20b)

for **DIR2=0** or DIR1=0 respectively and TS0_HRS=1:
PSAC[i] = PSA[i] - (8*DLA[i]*RCDT_SX_NOM)*MLS1 (DPLL-20d)

with
RCDT_SX_NOM= (1/CDT_SX_NOM) * SYN_S (DPLL-20a)
and
RCDT_SX= 1/CDT_SX (DPLL-20b)

replace MLS1 in equations (DPLL-20) and (DPLL-20d) by MLS2 for (SMC=1 and RMO=1)

use the calculated value of (DPLL-17b) also for the generation of SUB_INCi and serve the action by transmission of TSAC[i] and PSAC[i] to ACT_D_i.
The action is to be updated for each new *STATE* event until the calculated time stamp is in the past.
In this case the values of TSAC[i] and PSAC[i] depend on the DPLL_CTRL_11.ACBU signal.

When DPLL_CTRL_11.ACBU = '0': Use the time stamp of the last input event instead of the calculated value and the calculated position stamp of the actual increment as target position value.

When DPLL_CTRL_11.ACBU = '1':

For ACB_[z][1]= '1': is used as input signal to control if "action in past" shall be checked based on position information. If the position has reached "past" use the calculated position stamp of the actual increment as target position value.

For ACB_[z][1]= '0': In this case the PSAC[i] is used as calculated by the DPLL.

For ACB_[z][0]= '1': is used as input signal to control if "action in past" shall be checked based on time information. If the time has reached "past" use the time stamp of the last input event instead of the calculated TSAC[i] value.

For ACB_[z][0]= '0': In this case the TSAC[i] is used as calculated by the DPLL.

Set the corresponding shadow bit in the DPLL_ACT_STA register. Because of the blocking read operation the ACT_D values can be read only once.

18.7.8.3 Equations DPLL-20 to calculate the position stamp backwards

For **DIR2=1** or DIR1=1 respectively and TS0_HRS=0:
PSAC[i] = PSA[i] + (DLA[i]*RCDT_SX_NOM)*MLS1 (DPLL-20c)
with
RCDT_SX_NOM= (1/CDT_SX_NOM) * SYN_S (DPLL-20a)
and
RCDT_SX= 1/CDT_SX (DPLL-20b)

For **DIR2=1** or DIR1=1 respectively and TS0_HRS=1:
PSAC[i] = PSA[i] + (8*DLA[i]*RCDT_SX_NOM)*MLS1 (DPLL-20e)
with
RCDT_SX_NOM= (1/CDT_SX_NOM) * SYN_S (DPLL-20a)
and
RCDT_SX= 1/CDT_SX (DPLL-20b)

replace MLS1 in equations (DPLL-20c) and (DPLL-20e) by MLS2 for (SMC=1 and RMO=1)

use the calculated value of (DPLL-20b) also for the generation of SUB_INCi and serve the action by transmission of TSAC[i] and PSAC[i] to ACT_D.

The action is to be updated for each new *STATE* event until the event is in the past. In this case use the time stamp of the last input event instead of the calculated value and the calculated position stamp of the actual increment as target position value. Set the corresponding shadow bit in the DPLL_ACT_STA register. Because of the blocking read operation the ACT_D values can be read only once.

use the calculated value of (DPLL-17b) also for the generation of SUB_INCi and serve the action by transmission of TSAC[i] and PSAC[i] to ACT_D_i.
The action is to be updated for each new *STATE* event until the calculated time stamp is in the past.
In this case the values of TSAC[i] and PSAC[i] depend on the DPLL_CTRL_11.ACBU signal.
When DPLL_CTRL_11.ACBU = '0': Use the time stamp of the last input event instead of the calculated value and the calculated position stamp of the actual increment as target position value.

When DPLL_CTRL_11.ACBU = '1':
For ACB_[z][1]= '1': is used as input signal to control if "action in past" shall be checked based on position information. If the position has reached "past" use the calculated position stamp of the actual increment as target position value.
For ACB_[z][1]= '0': In this case the PSAC[i] is used as calculated by the DPLL.

For ACB_[z][0]= '1': is used as input signal to control if "action in past" shall be checked based on time information. If the time has reached "past" use the time stamp of the last input event instead of the calculated TSAC[i] value.
For ACB_[z][0]= '0': In this case the TSAC[i] is used as calculated by the DPLL.

Set the corresponding shadow bit in the DPLL_ACT_STA register. Because of the blocking read operation the ACT_D values can be read only once.

18.8 Signal processing

18.8.1 Time stamp processing

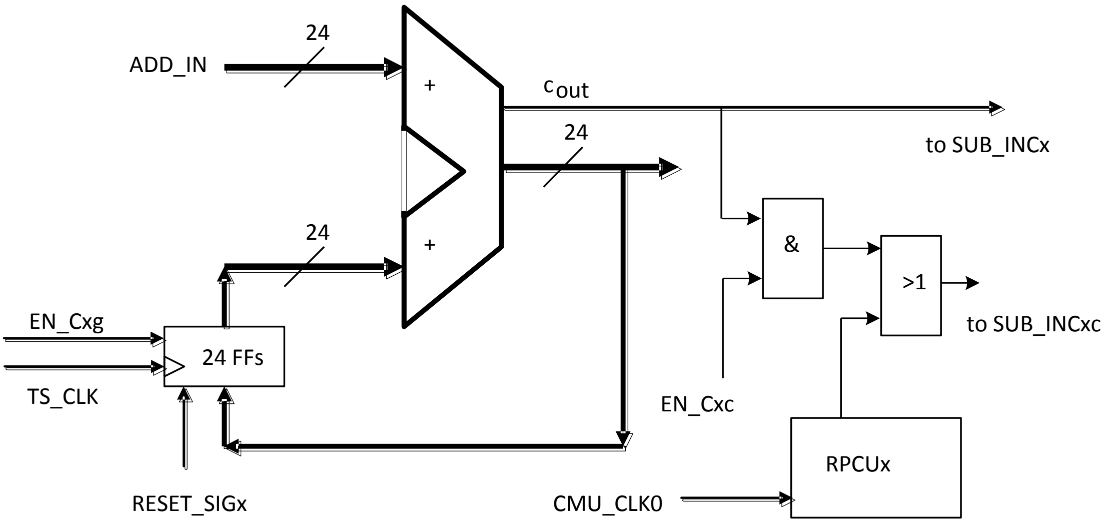
Signal processing does mean the computation of the time stamps in order to calculate at which time the outputs have to appear. For such purposes the time stamp values have to be stored in the RAM and by calculating the difference between old and new values the duration of the last time interval is determined simply. This difference should be also stored in the RAM in order to see the changes between the intervals by changing the conditions and the speed of the observed process.

18.8.2 Count and compare unit

The count and compare unit processes all input signals taking into account the configuration values. It uses a state machine and provides the output signals as described above.

18.8.3 Sub pulse generation for SMC=0

18.8.3.1 *Adder for generation of SUB_INCx by the carry c_{out}.*



Note: The *SUB_INC* generation by the circuit above has the advantage, that the resolution for higher speed values is better as for a simple down counter.
After RESET and after EN_Cxg=0 the flip-flops (FFs) should have a zero value. EN_Cxg has to be zero until reliable ADD_IN values are available and the pulse generation starts. This is controlled by the configuration bits SGE1,2 in the DPLL_CONTROL_1 register. The calculated values for the increment prediction using equations DPLL-2c 18.6.2.7, DPLL-2c1 18.6.4.4, DPLL-7c 18.6.3.7 or DPLL-7c1 18.6.4.4 respectively are valid only when at least NUTE>1 *TRIGGER* values or at least NUSE>1 *STATE* values are available. For NUTE =1 or NUSE=1 respectively the equations DPLL-25 18.8.3.4 and DPLL-26 18.8.3.6 use the actual increment value subtracted by the weighted average error.

The generation of *SUB_INC1* pulses depends on the configuration of the DPLL.
In automatic end mode the counter **INC_CNT1** resets the enable signal EN_C1 when the number of pulses desired is reached. In this case only the uncompensated output *SUB_INC1* remains active in order to provide pulses for the input filter unit. In the case of acceleration missing pulses can be determined at the next *TRIGGER/STATE* event in normal/emergency mode easily. For the correction strategy COA = 0 those missing

pulses are sent out **with CMU_CLK0** frequency as soon they are determined. During this time period the EN_Cxg remains cleared. After calculation or providing of a new ADD_IN value the FFs are enabled by EN_Cxg. In this way no pulse is lost. The new pulses are sent out afterwards, when **INC_CNT1** is set to the desired value, maybe by adding MLT+1 or MLS1 respectively for the new *TRIGGER/STATE* event.

Because the used DIV procedure of the algorithms results only in integer values, a systematic failure could appear. The pulse generation at *SUB_INC1* will stop in automatic end mode when the **INC_CNT1** register reaches zero or all remaining pulses at a new increment will be considered in the next calculation. In this way the loss of pulses can be avoided.
When a new *TRIGGER/STATE* appears the value of SYN_T*(MLT+1) or SYN_S*MLS1 respectively is added to **INC_CNT1**, **when SGE1=1.** Therefore for FULL_SCALE 2*(TNU+1)*(MLT+1) pulses *SUB_INC1* generated, when **INC_CNT1** reaches the zero value. The generation of *SUB_INC1* pulses has to be done as fast as possible. The calculations for the ADD_IN value must be done first. Therefore all values needed for calculation are to be fetched in a forecast.

18.8.3.2 Equation DPLL-21 to calculate the number of pulses to be sent in normal mode using the automatic end mode condition

For RMO=0, SMC=0 and DMO=0

NMB_T = ((MLT+1) + PD_store) *SYN_T + MP + MPVAL1 (DPLL-21)
with

PD_store = ADT_T[12:0] , prefetched during last increment
SYN_T = ADT_T[18:16], prefetched during last increment
MPVAL1 = pulse correction value for PCM1_SHADOW_TRIGGER=1

while the value for PD_store is zero for AMT=0
and
the value of MP is zero for COA=0

In order to get a higher resolution for higher speed a generator for the sub-pulses is chosen using an adder. All missing pulses MP are considered using equation DPLL-21 and are determined by counting the number of pulses of the last increment. The value SYN_T is stored from the last increment using NT of the ADT_T[i] value at RAM region 2c.

18.8.3.3 Equations DPLL-22-24 to calculate the number of pulses to be sent in emergency mode using the automatic end mode condition for SMC=0

For RMO=1, SMC=0 and DMO=0;
the value for PD_S_store is zero for AMS=0

NMB_S = (MLS1 + PD_S_store) *SYN_S+ MP (DPLL-22)
with
MLS1= (MLT+1) * (TNU+1) / (SNU+1) (DPLL-23)
PD_S_store = ADT_S[15:0], prefetched during last increment
SYN_S = ADT_S[21:16], prefetched during last increment
MPVAL1 = pulse correction value for PCM1_SHADOW_STATE=1

while the value for PD_S_store is zero for AMS=0
and
the value of MP is zero for COA=0

Please note, that these calculations above in equations DPLL-21 and DPLL-22 are only valid for an automatic end mode (DMO = 0).
For calculation of the number of generated pulses a value of 0.5 is added as shown in equations DPLL-25 or DPLL-26 respectively in order to compensate rounding down errors at the succeeding arithmetic operations. Because in automatic end mode the number of pulses is limited by **INC_CNT1** it is guaranteed, that not more pulses as needed are generated and in the same way missing pulses are caught up for the next increment.

18.8.3.4 Equation DPLL-25 to calculate ADD_IN in normal mode for SMC=0

In normal mode (for RMO=0) calculate
in the case LOW_RES=TS0_HRT

ADD_IN_CALN= (NMB_T+0.5) *RCDT_TX (DPLL-25)
with
RCDT_TX is the 2^{32} time value of the quotient in equation DPLL-17b
18.7.6.3.

In normal mode (for RMO=0) calculate
in the case LOW_RES=1 and TS0_HRT=0

ADD_IN_CALN= (NMB_T+0.5) *(RCDT_TX /8) (DPLL-25a)
with
RCDT_TX is the 2^{32} time value of the quotient in equation DPLL-17b

18.7.6.3.

For RMO=0 and SMC=0:

ADD_IN_CAL1 = ADD_IN_CALN (DPLL-25b)

LOW_RES=0 and TS0_HRT=1 is not possible. For such a configuration the RCT bit in the DPLL_STATUS register is set together with the ERR bit.

In the automatic end mode (DMO=0) missing pulses should be sent to the input RPCUx (rapid pulse catch up on) in 18.8.3.1, to be caught up on with CMU_CLK0 (for COA=0). When normal and rapid pulses are generated simultaneously, the SUB_INCx frequency is doubled at this moment in order to count two pulses at the TBU_CHx_BASE register. In order to make the frequency doubling possible, the CMU_CLK0 should be having a frequency which does not exceed half the frequency of TS_CLK. In addition the ADD_IN value should never exceed the value 0x800000. This limitation is only necessary for DMO=0 and COA=0 (see DPLL_CTRL_1 register).

For the normal mode replace ADD_IN of the ADDER (see Figure 18.8.3.1) by ADD_IN_CAL1 (when calculated, DLM=0) or ADD_IN_LD1 (when provided by the CPU, DLM=1).
The sub-pulse generation in this case is done by the following calculations using a 24 bit adder with a carry out c_{out} and the following inputs:

- ADD_IN
- the second input is the output of the adder, stored one time stamp clock before

In order not to complicate the calculation procedure use a Multiplier with a sufficient bit width at the output and use the corresponding shifted output bits.

18.8.3.5 Enabling of the compensated output for pulses

The c_{out} of the adder influences directly the *SUB_INC1* output of the DPLL (see Figure 18.8.3.1). The compensated output SUB_INCxc is in automatic end mode only enabled by EN_Cxc when **INC_CNTx** >0.

18.8.3.6 Equation DPLL-26 to calculate ADD_IN in emergency mode for SMC=0

In emergency mode (RMO=1) calculate

in the case LOW_RES=TS0_HRS

ADD_IN_CALE= (NMB_S+0.5)* RCDT_SX (DPLL-26)
while
RCDT_SX is the 2^{32} time value of the quotient in equation DPLL-20b 18.7.8.2.

In emergency mode (RMO=1) calculate
in the case LOW_RES=1 and TS0_HRS=0

ADD_IN_CALE= (NMB_S+0.5)* RCDT_SX /8 (DPLL-26a)
while
RCDT_SX is the 2^{32} time value of the quotient in equation DPLL-20b 18.7.8.2.

For RMO=1 and SMC=0:

ADD_IN_CAL1 = ADD_IN_CALE (DPLL-26b)

LOW_RES=0 and TS0_HRS=1 is not possible. For such a configuration the RCS bit in the DPLL_STATUS register is set together with the ERR bit.

In the automatic end mode (DMO=0) missing pulses should be sent to the input RPCUx (rapid pulse catch up on) in 18.8.3.1, to be caught up on with CMU_CLK0 (for COA=0). When normal and rapid pulses are generated simultaneously, the SUB_INCx frequency is doubled at this moment in order to count two pulses at the TBU_CHx_BASE register. In order to make the frequency doubling possible, the CMU_CLK0 should be having a frequency which does not exceed half the frequency of the system clock. In addition the ADD_IN value should never exceed the value 0x800000 when the TS_CLK frequency exceeds half the frequency of the system clock. This limitation is only necessary for DMO=0 and COA=0 (see DPLL_CTRL_1 register).
For the emergency mode replace ADD_IN of the ADDER (see Figure 18.8.3.1) by ADD_IN_CAL1 (when calculated, DLM=0) or ADD_IN_LD1 (when provided by the CPU, DLM=1).
The sub-pulse generation in this case is done by the following calculations using a 24 bit adder with a carry out c_{out} and the following inputs:

- ADD_IN
- the second input is the output of the adder, stored one time stamp clock before.

In order not to complicate the calculation procedure use a Multiplier with a sufficient bit width at the output and use the corresponding shifted output bits.

18.8.4 Sub pulse generation for SMC=1

18.8.4.1 Necessity of two pulse generators

The Adder of picture 18.8.3.1 must be implemented twice in the case of SMC=1: one for SUB_INC1 controlled by the *TRIGGER* input and (while RMO=1) one for SUB_INC2, controlled by the *STATE* input. In the case described in the chapter above for SMC=0 only one Adder is used to generate SUB_INC1 controlled by the *TRIGGER* in normal mode or by *STATE* in emergency mode.

18.8.4.2 Equation DPLL-27 to calculate the number of pulses to be sent for the first device using the automatic end mode condition

For SMC=1 and DMO=0

NMB_T = (MLS1 + PD_store) *SYN_T + MP + MPVAL1 (DPLL-27)

with
PD_store = ADT_T[12:0] , prefetched during last increment
SYN_T = ADT_T[18:16], prefetched during last increment
MPVAL1 = pulse correction value for PCM1_SHADOW_TRIGGER=1

while the value for PD_store is zero for AMT=0
and
for COA=0 use zero instead of the value of MP

18.8.4.3 Equation DPLL-28 to calculate the number of pulses to be sent for the second device using the automatic end mode condition

for RMO=1, SMC=1 and DMO=0

NMB_S = (MLS2 + PD_S_store) *SYN_S+ MP + MPVAL2 (DPLL-28)

with
PD_S_store = ADT_S[15:0], prefetched during last increment
SYN_S = ADT_S[21:16], prefetched during last increment
MPVAL2 = pulse correction value for PCM2_SHADOW_STATE=1

while the value for PD_S_store is zero for AMS=0
and
for COA=0 use zero instead of the value of MP

Please note, that these calculations above in equations DPLL-27 and DPLL-28 are only valid for an automatic end mode (DMO = 0). In addition the number of generated pulses is added by 0.5 as shown in equations DPLL-30 or DPLL-31 respectively in order to compensate rounding down errors at the succeeding division operation. Because in automatic end mode the number of pulses is limited by **INC_CNTx** it is guaranteed, that not more pulses as needed are generated and in the same way missing pulses are made up for the next increment.

18.8.4.4 Equation DPLL-30 to calculate ADD_IN for the first device for SMC=1

The sub-pulse generation in this case is done by the following calculations using a 24 bit adder with a carry out c_{out} and the following inputs:

- ADD_IN
- the second input is the (delayed) output of the adder, stored with each time stamp clock.

Replace ADD_IN by ADD_IN_CAL1 (when calculated, DLM1=0) or ADD_IN_LD1 (when provided by the CPU, DLM1=1) respectively while:

For SMC=1 and LOW_RES=TS0_HRT

ADD_IN_CAL1= (NMB_T+0.5) * RCDT_TX (DPLL-30)

When RCDT_TX is the 2^{32} time value of the quotient in equation DPLL-17b 18.7.6.3.

For SMC=1,LOW_RES= 1 and TS0_HRT=0

ADD_IN_CAL1= (NMB_T+0.5) * (RCDT_TX /8) (DPLL-30a)

When RCDT_TX is the 2^{32} time value of the quotient in equation DPLL-17b 18.7.6.3.

In order not to complicate the calculation procedure use a Multiplier with a sufficient bit width at the output and use the corresponding shifted output bits.

ADD_IN_CAL1 is a 24 bit integer value. The CDT_TX is the expected duration of current *TRIGGER* increment.
The c_{out} of the adder influences directly the *SUB_INC1* output of the DPLL (see 18.8.3.1). The SUB_INC1 output is in automatic end mode only enabled by EN_C1 when **INC_CNT1** >0.

18.8.4.5 Equation DPLL-31 to calculate ADD_IN for the second device for SMC=1

Replace ADD_IN by ADD_IN_CAL2 (when calculated, DLM2=0) or ADD_IN_LD2 (when provided by the CPU, DLM2=1) respectively while:

for SMC=1, RMO=1 and LOW_RES=TS0_HRS:
ADD_IN_CAL2= (NMB_S+0.5)* RCDT_SX (DPLL-31)

When RCDT_SX is the 2^{32} time value of the quotient in equation DPLL-20b 18.7.8.2.

for SMC=1, RMO=1, LOW_RES=1 and TS0_HRS=0:
ADD_IN_CAL2= (NMB_S+0.5)* (RCDT_SX /8) (DPLL-31a)

When RCDT_SX is the 2^{32} time value of the quotient in equation DPLL-20b 18.7.8.2.

In order not to complicate the calculation procedure use a Multiplier with a sufficient bit width at the output and use the corresponding shifted output bits.

The c_{out} of the adder2 influences directly the *SUB_INC2* output of the DPLL (see chapter 18.8.3.1).
The SUB_INC2 output is in automatic end mode only enabled by EN_C2 when **INC_CNT2** >0.
Note:
Please note, that after RESET and after EN_Cxc=0 (after stopping in automatic end mode) the flip-flops (FFs) have a zero value and also EN_Cxg has to be zero until reliable ADD_IN values are available and the pulse generation starts. The calculated values for the increment prediction using equations DPLL-2c 18.6.2.7, DPLL-2c1 18.6.4.4, DPLL-7c 18.6.3.7 or DPLL-7c1 18.6.4.4 respectively are valid only when NUTE>1 or NUSE>1 respectively. For NUTE=1 or NUSE=1 respectively the equations DPLL-30 (see chapter 18.8.4.4) and DPLL-31 (see chapter 18.8.4.5) use the actual increment value subtracted by the weighted average error.

The generation of *SUB_INCx* pulses depends on the configuration of the DPLL.

In automatic end mode the counter **INC_CNTx** resets the enable signal EN_Cxcu when the number of pulses desired is reached. In this case only the uncompensated outputs SUB_INCx remain active in order to provide pulses for the input filter units. A new *TRIGGER* or *STATE* input respectively can reset the FFs and also ADD_IN, especially when EN_Cxc was zero before. In the case of acceleration missing pulses can be determined at the next *TRIGGER/STATE* event easily. For the correction strategy COA = 0 those missing pulses are sent out with CMU_CLK0 frequency as soon they are determined. After that the pulse counter **INC_CNTx** should be always zero and the new pulses are sent out afterwards, when **INC_CNTx** is set to the desired value by adding MLS1 or MLS2 for the new *TRIGGER* or *STATE* event respectively.

Because the used DIV procedure of the algorithms results only in integer values, a systematic failure could appear. The pulse generation will stop when the **INC_CNTx** register reaches zero or all remaining pulses at a new increment will be considered in the next calculation. In this way the loss of pulses can be avoided.

When a new *TRIGGER* appears the value of SYN_T*MLS1 is added to **INC_CNT1**. Therefore for FULL_SCALE 2*(TNU+1)*MLS1 pulses *SUB_INC1* generated, when **INC_CNT1** reaches the zero value. The generation of SUB_INC1 pulses has to be done as fast as possible.

When a new *STATE* appears the value of SYN_S*MLS2 is added to **INC_CNT2**. Therefore for FULL_SCALE 2*(SNU+1)*MLS2 pulses *SUB_INC2* generated, when **INC_CNT2** reaches the zero value. The generation of SUB_INC2 pulses has to be done as fast as possible.

18.8.5 Calculation of the Accurate Position Values

All appearing *TRIGGER* and *STATE* signals do have a time stamp and a position stamp assigned after the input filter procedure. For the calculation of the exact time stamp the filter values are considered in the calculations of equations DPLL-1a 18.6.2.1 or DPLL-6a 18.6.3.1 respectively. A corresponding calculation is to be performed for the calculation of position values.
The PSTC and PSSC values can be corrected by the CPU, when needed.

After reset, while FTD=0 and no active TRIGGER slope is detected:
PSTC = 0 (DPLL-32a)

Calculate the new Position value for each active TRIGGER event:
PSTC= PSTC_old + NMB_T_TAR_OLD (DPLL-32b)
when FTD=1 and SGE1=1

with
PSTC_old is the last PSTC value and
NMB_T_old is the number of pulses which are calculated
and provided for sending out in the last increment.

After reset, while FSD=0 and no active STATE slope is detected:
PSSC= 0 (DPLL-33a)

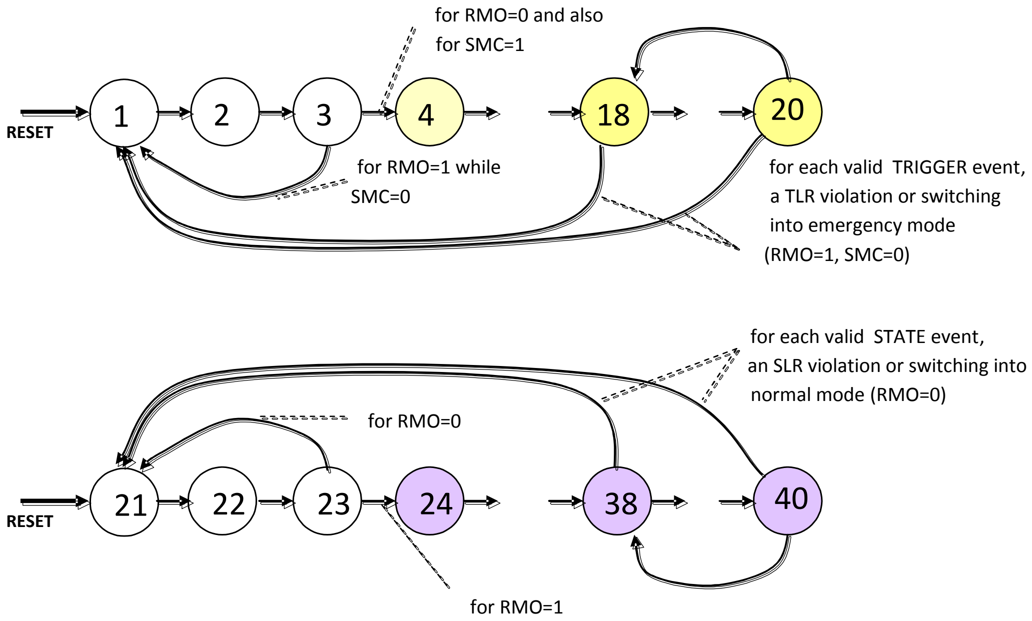
Calculate the new Position value for each STATE event:
PSSC= PSSC_old + NMB_S_TAR_OLD (DPLL-32b)
when FSD=1 and SGE1=1 (SMC=0) or SGE2=1 (SMC=1)
respectively with
PSSC_old is the last PSSC value and
NMB_S_old is the number of pulses which are calculated
and provided for sending out in the last increment.

18.8.6 Scheduling of the Calculation

After enabling the DPLL with each active *TRIGGER* or *STATE* event respectively a cycle of operations is performed to calculate all the results shown in detail in the table below 18.2. A state machine controls this procedure and consists of two parts, the first is triggered by an active slope of the signal *TRIGGER*, begins at step 1 and ends at step 20 (in normal mode and for SMC=1). The second state machine is controlled by an active slope of the signal *STATE*, begins at step 21 and ends at step 40 (in emergency mode and also for SMC=RMO=1). Depending on the mode used all 20 steps are executed or already after 2 steps the jump into the initial state is performed, as shown in the state machine descriptions below. For each new extended cycle (without this jump) all prediction values for actions in the case SMC=0 are calculated once more (with maybe improved accuracy because of better parameters) and all pending decisions are made using these new values when transmitted to the decision device.

In 18.8.6.7.1 the steps of the state machine are described. Please note, that the elaboration of the steps depends on the configuration bits described in the comments. The steps 4 to 17 are only calculated in normal mode (in the state machine explanation below marked yellow in 18.2), but steps 24 to 37 are only calculated in emergency mode (in the state machine explanation below marked cyan in 18.2) when SMC=0.

18.8.6.1 State machine partitioning for normal and emergency mode.



18.8.6.2 *Synchronization description*

TRIGGER:

The APT (address pointer for duration and reciprocal duration values of *TRIGGER* increments) is initially set to zero and incremented with each active *TRIGGER* event. Therefore data are stored in the RAM beginning from the first available value. The actual duration of the last increment is stored at DT_T_ACT. For the prediction of the next increment it is assumed, that the same value is valid as long as NUTE is one.

A missing *TRIGGER* is assumed, when at least after TOV* DT_T_ACT no active *TRIGGER* event appears.

The data of equations DPLL-1b1 and DPLL-1c2 18.7.5.4 are written in the corresponding RAM regions and APT is incremented accordingly up to 2*TNU-2*SYN_NT+1.

The APT_2B (address pointer for the time stamp field of *TRIGGER*) is initially set to zero and incremented with each active *TRIGGER* event. When no gap is detected because of the incomplete synchronization process at the beginning, for all *TRIGGER* events the time stamp values are written in the RAM up to 2*(TNU+1) entries, although only 2*(TNU+1-SYN_NT) events in FULL_SCALE appear. When the current position is detected, the synchronization procedure can be performed as described below:

Before the CPU sets the APT_2C address pointer in order to synchronize to the profile, it writes the corresponding increment value for the necessary extension of the RAM region 2b value APT_2B_EXT into the register APT_2B_sync and sets the status bit APT_2C_status. This value can be e.g. 2*SYN_NT, when all gaps in FULL_SCALE

already passed the input data stream of *TRIGGER*, or less then this value, when up to now e.g. only a single gap is to be considered in the data stream stored already in the RAM region 2b. The number of virtual increments to be considered depends on the number of inputs already got. After writing APT_2C by the CPU, with the next *TRIGGER* event the APT_2B address pointer is incremented (as usual) and then the additional offset value APT_2B_EXT is added to it once (while APT_2B_STATUS=1 and for forward direction). For that reason the APT_2B_STATUS bit is reset after it. The old APT_2B value before adding the offset is stored in the APT_2B_OLD register as information for the CPU where to start the extension procedure. In the following the CPU fills in the time stamp field around the APT_2B_OLD position taking into account the corresponding number of virtual entries stored in the APT_2B_EXT value and the corresponding NT values in the profile. The extension procedure ends when all gaps considered in the APT_2B_EXT value are treated once. In the consequence all storage locations of RAM region 2b up to now do have the corresponding entries. Future gaps are treated by the DPLL.
For a backward direction the APT_2C_ext value is subtracted accordingly.

When the CPU writes the APT_2C address pointer the SYT bit is set simultaneously. For SYT=1 in normal mode (SMC=0) the LOCK1 bit is set with the system clock, when the right number of increments between two synchronization gaps is detected by the DPLL. An unexpected missing *TRIGGER* or an additional *TRIGGER* between two synchronization gaps does reset the LOCK1 bit in normal mode. In that case the CPU must correct the SUB_INC pulse number and maybe correct the APT_2C pointer. For this purpose the LL1I interrupt can be used.

When SYT is set the calculations of equations DPLL-1 to DPLL-5 are performed accordingly and the values are stored in (and distributed to) the right RAM positions.
This includes the multiple time stamp storage by the DPLL for a gap according to equations DPLL-1a5 to 7 forwards 18.7.5.2 or backwards 18.7.5.3. The APT_2B pointer is for that reason incremented or decremented before this operation considering the virtual increments in addition.
Please note, that for the APT and APT_2C pointers the gap is considered as a single increment.

STATE:
The APS (address pointer for duration and reciprocal duration values of *STATE*) is initially set to zero and incremented with each active *STATE* event. Therefore data are stored in the RAM field beginning at the first location. The actual duration of the last increment is stored at DT_S_ACT. For the prediction of the next increment it is assumed, that the same value is valid as long as NUSE is one.

A missing *STATE* is assumed, when at least after SOV* DT_S_ACT no active *STATE* event appears.

The data of equations DPLL-6b1 and DPLL-6c2 18.7.5.8 is written in the corresponding RAM regions and APS is incremented accordingly up to 2*SNU-2*SYN_NS+1 (for SYSF=0).

The APS_1C2 (address pointer for the time stamp field of *STATE*) is initially set to zero and incremented with each active *STATE* event. When no gap is detected because of the incomplete synchronization process at the beginning, for all *STATE* events the time stamp values are written in the RAM up to 2*(SNU+1) entries, although (e.g. for SYSF=0) only 2*(SNU+1-SYN_NS) events in FULL_SCALE appear. When the current position is detected, the synchronization procedure can be performed as described below:

Before the CPU sets the APS_1C3 address pointer in order to synchronize to the profile, it writes the corresponding increment value APS_1C2_EXT for the necessary extension of the RAM region 1c2 into the register DPLL_APS_SYNC and sets the APS_1C2_STATUS bit there. This value can be e.g. 2*SYN_NS (for SYSF=0) or SYN_NS (for SYSF=1), when all gaps in FULL_SCALE already passed the input data stream of *STATE*. Also less then this value can be considered, when up to now only a single gap is to be considered in the data stream stored already in the RAM region 1c2. The number of increments to be considered depends on the number of inputs already got. After writing APS_1C3 by the CPU, with the next active *STATE* slope the APS_1C2 address pointer is incremented (as usual) and then the additional offset value APS_1C2_EXT is added to it once (while APS_1C2_STATUS=1 and forward direction). For that reason the APS_1C2_STATUS bit is reset after it. The old APS_1C2 value is stored in the APS_1C2_OLD register as information for the CPU where to start the extension procedure. In the following the CPU extends the time stamp field beginning from the APS_1C2_OLD position taking into account the corresponding number of virtual entries according to the APS_1C2_EXT value and also the correspondent NS values in the profile. The extension procedure ends when all gaps considered in the APS_1C2_EXT value are treated once. In the consequence all storage locations of RAM region 1c2 up to now do have the corresponding entries. Future gaps are treated by the DPLL.

For a backward direction the APS_1C2_EXT value is subtracted accordingly.

When the CPU writes the APS_1C3 address pointer the SYS bit is set simultaneously. For SYS=1 in emergency mode (SMC=0 and DMO=1) the LOCK1 bit is set with the system clock, when the right number of increments between two synchronization gaps is detected by the DPLL. An unexpected missing *STATE* or an additional *STATE* between two synchronization gaps does reset the LOCK1 bit in emergency mode. In that case the CPU must correct the SUB_INC1 pulse number and maybe correct the APS_1C3 pointer. For this purpose the LL1I interrupt can be used.

When SYS is set the calculations of equations DPLL-5 to DPLL-10 are performed accordingly and the values are stored in (and distributed to) the right RAM positions.

This includes the multiple time stamp storage by the DPLL for a gap according to equations DPLL-6a5 to 7 forwards 18.7.5.6 or backwards 18.7.5.7. The APS_1C2 pointer is for that reason incremented or decremented before this operation considering the virtual increments in addition.

Please note, that for the APS and APS_2c pointers the gap is considered as a single increment.

SMC=1:
For SMC=1 it is assumed, that the starting position is known by measuring the characteristic of the device. In this way the APT and APT_2C as well the APS and APS_1C3 values are set properly, maybe with an unknown repetition rate. When no gap is to be considered for *TRIGGER* or *STATE* signals the APT_2B and APS_1C2 address pointers are set equal to APT or APS respectively. It is assumed, that all missing *TRIGGER*s and missing *STATE*s can be also considered from the beginning, when a valid profile with the corresponding adapted values is written in the RAM regions 1c3 and 2c respectively. In that case the TSF_T[i] and TSF_S[i] must be extended by the DPLL according to the profile. . Thus the SYT and SYS bits could be set from the beginning and the LOCK1 and LOCK2 bits are set after recognition of the corresponding gaps accordingly. When no gap exists (SYN_NT=0 or SYN_NS=0), the LOCK bits are set immediately. The CPU can correct the APT_2C and APS_1C3 pointer according to the recognized repetition rate later once more without the loss of Lock1,2.

18.8.6.3 Operation for direction change in normal and emergency mode (SMC=0)

When for SMC=0 in normal mode a backwards condition is detected for the TRIGGER input signal (e.g. when THMI is not violated), the LOCK1 bit in the DPLL_STATUS register is reset, the NUTE value in NUTC register is set to 1 (the same for NUSE in NUSC). The address pointers APT_2C as described below (and after that decremented for each following active slope of *TRIGGER* as long as the DIR1 bit shows the backward direction).
Please notice, that in the case of the change of the direction the ITN and ISN bit in the DPLL_STATUS register are reset.

For this transition to the backward direction no change of address pointer APT and APT_2B is necessary.
profile update for TRIGGER when changing direction
The profile address pointer APT_2C is changed step by step in order to update the profile information in SYN_T, SYN_T old and PD_store:
- decrement APT_2C, load SYN_T
- decrement APT_2C, load SYN_T
- decrement APT_2C, load SYN_T, PD_store, update SYN_T_OLD
- decrement APT_2C, **make calculations**, load SYN_T and PD_store, update SYN_T_OLD and PD_store_old and wait for a new *TRIGGER* event.

Note: The update of SYN_T_OLD and the loading of PD_store can be performed in all steps above. The value of APT_2B needs not to be corrected. For a direction change from backwards to forwards make the same corrections by incrementing APT_2C.

Make calculations does mean: the operation of the state machine starts with the calculations of NMB_T and INC_CNT1 using the actual APT_2C address pointer value, see 18.2.

The TBU_TB1 value is to be corrected by the number of pulses sent out in the wrong direction mode during the last and current increment. This correction is done by sending out SUB_INC1 pulses for decrementing TBU_TB1 (while DIR1=1).

Save inc_cnt1 value at direction change to inc_cnt1_save.
Calculate the new inc_cnt1 value as follows:
1. Stop sending pulses and save inc_cnt1 at the moment of direction change as inc_cnt1_save .
2. Set inc_cnt1 to the target value of the last increment

nmb_t_tar_old

3. Add the target number of trigger which were calculated for the current increment when this value was already added to inc_cnt1 before the direction change is detected

+ nmb_t_tar

4. Subtract the value of still not sent pulses (remaining value at inc_cnt1_save)

- inc_cnt1_save

5. Calculate the new target pulses to be sent considering the new values of SYN_T and PD_store and add them:

+ nmb_t_tar_new

This does mean the following equation:
inc_cnt1 = nmb_t_tar_old + nmb_t_tar
- inc_cnt1_save + nmb_t_tar_new
All pulses summarized at inc_cnt1 are sent out by the maximum possible frequency, because no speed information is available for the first increment after changing the direction. Please notice that no pulse correction using PCM1 of DPLL_CTRL1 is possible during direction change.
When PSTC was incremented/decremented at the active slope and after that the direction change was detected at the same input event, correct **PSTC** once by

- nmb_t_tar_old when changed to backwards
+ nmb_t_tar_old when changed to forwards

in order to compensate the former operation. When the direction information is known before an intended change of PSTC, do not change them.
Store the new calculated value **nmb_t_tar_new** at **nmb_t_tar** for the correct calculation of PSTC at the next input event.

consequences for STATE
With the next active *STATE* event the direction information is already given. The profile pointer APS_1C3 is to be corrected by a two times decrement in order to point to the profile of the next following increment. In the following it is decremented with each *STATE* event while DIR1=1. The SYN_S and PD_S_store values must be updated accordingly, including SYN_S_OLD and PD_S_store_old.

Because the right direction is already known when an input event appears, make the following corrections:
- decrement APS_1C3, load SYN_S and PD_S_store, update SYN_S_OLD and PD_S_store_old
- decrement APS_1C3, **make calculations**, load SYN_S and PD_S_store, update SYN_S_OLD and PD_S_store_old and wait for a new *STATE* event.

Note: The update of SYN_S_OLD and the loading of PD_S_store can be performed in all steps above. The value of APS_1C2 needs not to be corrected.

When a new *STATE* event occurs, all address pointers are decremented accordingly as long as DIR1=1.

In **emergency mode** the pulses are corrected as follows:

Save inc_cnt1 value at direction change to inc_cnt1_save.

Calculate the new inc_cnt1 value as follows:

1. Stop sending pulses and save inc_cnt1 at the moment of direction change as inc_cnt1_save.
2. Set inc_cnt1 to the target value of the current increment

nmb_s_tar

Please notice, that in difference to the normal mode, nmb_s_tar is to be used instead of nmb_s_tar_old, because direction information in emergency mode is only given from the TRIGGER input and occurs of a STATE event independently.

That means: The calculations at the last STATE event were done for the correct former direction. In addition still no pulse calculations are performed for the current increment, because the direction change is known at the moment of the recent STATE event. Later direction changes are considered at the next STATE event.

3. Do not add the calculated number of state pulses because no new STATE event occurred.
4. Subtract the value of still not sent target pulses (remaining value at inc_cnt1_save)

- inc_cnt1_save

5. Add the new calculated target pulses for the current increment

+ nmb_s_tar_new

when for the calculation all new conditions of PD_S_store and SYN_S are considered.

inc_cnt1 = nmb_s_tar_old - inc_cnt1_save + nmb_s_tar_new

All pulses summarized at inc_cnt1 are sent out by the maximum possible frequency, because no speed information is available for the first increment after changing the direction. Please notice that no pulse correction using PCM1 of DPLL_CTRL1 is possible during direction change.

Do not change PSSC and suppress incrementing/decrementing of PSSC at the event directly following to the direction change information.

Store the new calculated value **nmb_s_tar_new** at **nmb_s_tar** for the correct calculation of PSTC at the next input event.

repeated change to forward direction for TRIGGER

The DIR1 bit remains set as long as the THMI value remains none violated for the following *TRIGGER* events and is reset when for an inactive TRIGGER slope the THMI is violated.

Resetting the DIR1 to 0 results (after repeated reset of LOCK1, ITN, ISN) the opposite correction of the profile address pointer considered.

This does mean two increment operations of the address pointer APS_1C3 including the update of SYN_S and PD_S_store with the automatic update of SYN_S_OLD and PD_S_store_old for STATE and four increment operations of the address pointer APT_2C including the update of SYN_T and PD_store with the automatic update of SYN_T_OLD and PD_store_old for TRIGGER.

The correction of TBU_CH1 is done by sending out the correction pulses with the highest possible frequency at SUB_INC1 while DIR1=0. The number of pulses is calculated as shown above.

consequences for STATE
see corrections above. After that the address pointers are incremented again with each following active *STATE* event as long as DIR1=0.

18.8.6.4 Operation for direction change for TRIGGER (SMC=1)

When for SMC=1 a backwards condition is detected for the *TRIGGER* input signal (TDIR=1, resulting in DIR1=1), the LOCK1 bit in the DPLL_STATUS register is reset, the NUTE value in NUTC register is set to 1. The address pointers APT and APT_2C as well as APT_2B are decremented for each active slope of *TRIGGER* as long as the DIR1 bit shows the backward direction.
Please notice, that in the case of the change of the direction the ITN bit in the DPLL_STATUS register is reset.

profile update for TRIGGER
Make the same update steps for the profile address pointer as shown in chapter 18.8.6.3: Decrement APT_2C for 2 times with the update of the SYN_T and PD_store values at each step with an automatic update of SYN_T_OLD and PD_store_old:
- decrement APT_2C, load SYN_T, PD_store, update SYN_T_OLD
- decrement APT_2C, **make calculations**, load SYN_T and PD_store, update SYN_T_OLD and PD_store_old and wait for a new *TRIGGER* event.

In the normal case no correction of wrong pulses sent is necessary, because the direction change is detected by the pattern immediately.
Nevertheless a correction is necessary as shown below. In the other case: see treatment of pulses TBU_CH1_BASE in normal mode at chapter 18.8.6.3.

Save inc_cntx value at direction change to inc_cnt1_save.
Calculate the new inc_cnt1 value as follows:
1. Clear inc_cnt1.
2. Set inc_cnt1 to the target value of the last increment
nmb_t_tar
Please notice, that in difference to the normal mode, nmb_t_tar is to be used instead of nmb_t_tar_old, because the direction information is known before the calculation takes place.
3. Do not add the calculated number of trigger pulses because it is not calculated yet before the direction change information is known.
4. Subtract the value of still not sent pulses (remaining value at inc_cnt1_save)

- inc_cnt1_save

5. Add the new calculated target pulses for the current increment

+ nmb_t_tar_new

when for the calculation all new conditions of PD_S_store and SYN_S are considered.

inc_cnt1 = nmb_t_tar_old - inc_cnt1_save + nmb_t_tar_new

All pulses summarized at inc_cnt1 are sent out by the maximum possible frequency, because no speed information is available for the first increment after changing the direction. Please notice that no pulse correction using PCM1 of DPLL_CTRL1 is possible during direction change.

Suppress changing of PSTC for the TRIGGER event when a direction change is detected.

Store the new calculated value **nmb_t_tar_new** at **nmb_t_tar** for the correct calculation of PSTC at the next input event.

repeated change to forward direction for TRIGGER

The DIR1 bit remains set as long as the TDIR bit is set for the following *TRIGGER* events and is reset when for an active *TRIGGER* slope the TDIR is zero.

Resetting the DIR1 to 0 results (after repeated reset of LOCK1 and ITN) the opposite correction of the address pointer use.

This does mean two increment operations of the address pointer including the update of SYN_T and PD_store.

A complex correction of TBU_CH1_BASE and INC_CNT1 is in the normal case not necessary, when all increments are equal (SYN_NT=0) and no adapt information is used. In this case only the MLS1 value is added to INC_CNT1 in order to back count the value for the last increment. In the other case: see treatment of pulses TBU_CH1_BASE and ICN_CNT1 in normal mode at chapter 18.8.6.3.

18.8.6.5 Operation for direction change for STATE (SMC=1)

When for SMC=1 a backwards condition is detected for the *STATE* input signal (SDIR=1, resulting in DIR2=1), the LOCK2 bit in the DPLL_STATUS register is reset, the NUSE value in NUSC register is set to 1 and the address pointers APS and APS_1C3_f and APS_1C2 are decremented for each active slope of *STATE* as long as the DIR2 bit shows the backward direction.

Please notice, that in the case of the change of the direction the ISN bit in the DPLL_STATUS register is reset.

For this transition to the backward direction no change of address pointer APS and APS_1C2 is necessary.

profile update for STATE

Make the same update steps for the profile address pointer as shown in chapter 18.8.6.3: Decrement APS_1C3 for 2 times with the update of the SYN_S, SYN_S_OLD, PD_S_store and PD_S_store_old values at each step:

- decrement APT_1c3, load SYN_S, PD_S_store, update SYN_S_OLD
- decrement APT_1c3, **make calculations**, load SYN_S and PD_S_store, update SYN_S_OLD and PD_S_store_old and wait for a new *STATE* event.

A complex correction of TBU_CH2_BASE and INC_CNT2 is in the normal case not necessary, when all increments are equal (SYN_NS=0) and no adapt information is used. In this case only the MLS2 value is added to INC_CNT2 in order to back count the value for the last increment. In the other case: see treatment of pulses TBU_CH1_BASE and ICN_CNT1 in normal mode at chapter 18.8.6.3.

For the second PMSM the pulses are corrected as follows:
Save inc_cnt2 value at direction change to inc_cnt2_save.
Calculate the new inc_cnt2 value as follows:
1. Clear inc_cnt2.
2. Set inc_cnt2 to the target value of the last increment
nmb_s_tar
Please notice, that in difference to the normal mode, nmb_s_tar is to be used instead of nmb_s_tar_old, because no new calculation is performed so far.
3. Do not add the calculated number of state pulses because it is not calculated yet before the direction change information is known.
4. Subtract the value of still not sent pulses (remaining value at inc_cnt2_save)
- inc_cnt2_save
5. Add the new calculated target pulses for the current increment
+ nmb_s_tar_new
when for the calculation all new conditions of PD_S_store and SYN_S are considered.
inc_cnt2 = nmb_s_tar_old - inc_cnt2_save + nmb_s_tar_new
All pulses summarized at inc_cnt2 are sent out by the maximum possible frequency, because no speed information is available for the first increment after changing the direction. Please notice that no pulse correction using PCM2 of DPLL_CTRL1 is possible during direction change.
Do not change PSSC for a STATE event when a direction change is detected.
Store the new calculated value **nmb_s_tar_new** at **nmb_s_tar** for the correct calculation of PSTC at the next input event.

repeated change to forward direction for STATE
The DIR2 bit remains set as long as the SDIR bit is set for the following *STATE* events and is reset when for an active *STATE* slope SDIR is zero.

Resetting the DIR2 to 0 results (after repeated reset of LOCK2 and FSD) in the opposite correction of the address pointer use.

After a last decrementing of all address pointers the APS_1C3 is incremented 2 times with a repeated update of SYN_S, SYN_S_OLD and PD_S_store after each increment.

18.8.6.6 DPLL reaction in the case of non plausible input signals

When the DPLL is synchronized concerning the *TRIGGER* signal by setting the FTD, SYT and LOCK1 bits in the DPLL_STATUS register, the number of active *TRIGGER* events between the gaps is to be checked continuously.

When additional events appear while a gap is expected, the LOCK1 bit is reset and the ITN bit in the DPLL_STATUS register is set.

When an unexpected gap appears (missing *TRIGGER*S), the NUTE value in the NUTC register is set to 1, the LOCK1 bit is reset and the ITN bit in the DPLL_STATUS register is set. The address pointers are incremented with the next active *TRIGGER* slope accordingly.

The TOR Bit in the DPLL_STATUS register is set, when the time to the next active *TRIGGER* slope exceeds the value of the last nominal *TRIGGER* duration multiplied with the value of the TLR register (see chapter 18.12.72). In this case also the TORI interrupt is generated, when enabled.

When in the following the direction DIR1 changes as described in the chapters above the ITN bit in the DPLL_STATUS register is reset, the use of the address pointers APT_2C is switched and the pulse correction takes place as described above.

In all other cases the CPU can interact to leave the instable state. This can be done by setting the APT_2C address pointer which results in a reset of the ITN bit. In the following NUTE can also be set to higher values.

When the DPLL is synchronized concerning the *STATE* signal by setting the FSD, SYS and LOCK1 (for SMC=0) or LOCK2 (for SMC=1) bits in the DPLL_STATUS register, the number of active *STATE* events between the gaps is to be checked continuously.

When additional events appear while a gap is expected or while an unexpected missing *STATE* event appears, the LOCK1,2 bit is reset and the ISN bit in the DPLL_STATUS register is set.

When an unexpected gap appears for RMO=SMC=1 (missing *STATE*s for synchronous motor control), the NUSE value in the NUSC register is set to 1, the LOCK2 bit is reset and the ISN bit in the DPLL_STATUS register is set. The address pointers are incremented with the next active *STATE* slope accordingly.

When the *STATE* locking range SLR is violated[7)], the state machine 2 will remain in state 21 and the address pointer APS, APS_1C2 and APS_1C3 will remain unchanged until the CPU sets the APS_1C3 accordingly. In this case also the NUSE value in the NUSC register is set to 1. The DPLL stops the generation of the SUB_INC1,2 pulses respectively and will perform no other actions - remaining in step21 of the second state machine (see 18.2).

[7] The SOR Bit in the DPLL_STATUS register is set, when the time to the next active *STATE* slope exceeds the value of the last nominal *STATE* duration multiplied with the value of the SLR register (see chapter 18.12.73).
In this case also the SORI interrupt is generated, when enabled.

When in the following the direction DIR2 changes as described in the chapters above the ISN bit in the DPLL_STATUS register is reset, the use of the address pointers APS_1C3 is switched and the pulse correction takes place as described above. In all other cases the CPU must interact to leave the instable state. This can be done by setting the APS_1C3 address pointers which results in a reset of the ISN bit. In the following NUSE can also be set to higher values.

18.8.6.7 State description of the State Machine.

18.8.6.7.1 State description of the State Machine Table

Step	Description	Comments

always for DEN=1	**for each inactive TRIGGER slope with TEN=1:**
	check, if the last active TRIGGER slope was passing the PVT check; only in this case perform the following tasks:
	calculate the time stamp difference ΔT to the last active event, store this value at THVAL;
	when THMI >0 is violated (ΔT < THMI):
	generate TINI interrupt,
	set DIR1=0 (forwards)
	set BWD1=0 (see DPLL_STATUS register)
	else (only for THMI >0):
	set DIR1= 1 (backwards);
	set BWD1=1 (see DPLL_STATUS register)
	after changing the direction correct the pulses WP sent with wrong direction information and send the pulses for the actual increment in addition with highest possible frequency: WP=NMB_T-DPLL_INC_CNT1;
		for SMC=0;
		set DIR1 always after inc./ decr. the address pointers APT, APT_x;
		go to step 1;
		stop output of SUB_INC1 and correct pulses after changing DIR1 after incr./ decr. of APS_x
		set DIR2 always after incr./decr. the address pointers APS, APS_x;
		go to step 1

	correct INC_CNT1 by addition of 2*WP before sending the correction pulses; generate the TISI interrupt; check THMA, when THMA is violated, generate the TAXI interrupt; go to step 1 **for each inactive *STATE* slope with SEN=1:** set DIR2=DIR1	
always for DEN=1 and (TEN=1 or SEN=1, respectively)	set DIR1=BWD1=TDIR , set DIR2=BWD2=SDIR; for each change of TDIR go to step 1 after performing the following calculations: correct INC_CNT1 correct the pulses (WP, see above) sent with wrong direction information and send the pulses for the actual increment in addition with highest possible frequency. For each change of SDIR go to step 21 after performing the following calculations: update of SYN_S, PD_S_store according to chapter 18.8.6.3 correct INC_CNT1,2 correct the pulses sent with wrong direction information and send the pulses for the actual increment in addition with highest possible frequency.	**for SMC=1**; set the direction bits always after incr./decr. the corresponding address pointers;
1	When DEN = 0 or TEN=0: stay in step 1 until DEN=1, TEN=1 and at least one active *TRIGGER* has been detected (FTD=1); the following steps are performed always (not necessarily in step 1, but also in steps 18 to 20 (when waiting for new PMTR values to be calculated): compare TRIGGER_S with TSL (active slope); **When no active TRIGGER slope appears** and when TS_T_CHECK time is reached:	Depending on TSL, TEN, DEN step one is leaving with the next *TRIGGER* input; **Note:** Step 1 is also left in emergency mode when an active *TRIGGER* event appears in order to make a switch back to normal mode possible; _old - values are values valid at the last but one active *TRIGGER* event ;

	• send missing *TRIGGER* INT, also when a gap is expected according to the profile; set MT=1 (missing *TRIGGER* bit) in the DPLL_STATUS register; do not leave the active step, until a valid active *TRIGGER* appears. **When an active TRIGGER slope appears check PVT** **- when the PVT value is violated:** generate the PWI interrupt, ignore the *TRIGGER* input and wait for the next active TRIGGER slope (ignore each inactive slope); do not store any value **- When the PVT value is fulfilled:** store the actual position stamp at PSTM (value at the TRIGGER event) update the RAM region 2 by equation DPLL-1a-c (see chapter 18.7.5) **store the actual INC_CNT1 value at MP1 as missing pulses** *(instead of calculation in step 5)* store all relevant configuration bits **X** of the DPLL_CTRL(0,1) Registers in **shadow** registers and consider them for all corresponding calculations of steps 2 to 20 accordingly; the relevant bits are explained in the registers itself generate the TASI interrupt; for FTD=0: • set PSTC=PSTM • set FTD (first *TRIGGER* detected) • do not change PSTC,APT, APT_2B • for (RMO=0 or SMC=1) **and SGE1=1**: increment INC_CNT1 by (MLT+1)*) +MPVAL1***) • send SUB_INC1 pulses with highest possible frequency **when SGE1=1 and DPLL_CTRL_11.SIP1 = 0.**	for the whole table: use always MLS1 instead of (MLT+1) for the case SMC=1 ; **dir_crement** does mean: increment for DIR1=0 decrement for DIR1=1 *)replace (MLT+1) by MLS1 for SMC=1 **) NMB_T_TAR is the target value of NMB_T of the last increment (see step 5 ff.) ***) add MPVAL1 once to INC_CNT1, that means only when PCM1=1 ****) **SGE1_delay** is the value of SGE1 delayed by one active TRIGGER event *****) PD_store = 0 for AMT=0 (see DPLL_CTRL_0 register)

	for SYT=0 and FTD =1: • **dir_crement** APT and APT_2B by one; • **dir_crement** for **SGE1_delay**[****)]**=1:** PSTC by NMB_T_TAR[**)] • for (RMO=0 or SMC=1) **and SGE1=1**: increment INC_CNT1 by (MLT+1)[*)] +MPVAL1[***)] for SYT=1 : • **dir_crement** APT, APT_2C**,** **dir_crement** APT_2B by SYN_T_OLD • **dir_crement** for **SGE1_delay**[****)]**=1** PSTC by NMB_T_TAR[**)] • for (RMO=0 or SMC=1) **and SGE1=1**: increment INC_CNT1 by SYN_T*((MLT+1)[*)]+ PD_store)[*****)] + MPVAL1[***)] PD_store is 0 for AMT=0 within the DPLL_STATUS register: • set LOCK1 bit accordingly;	
2	calculate TS_T according to equations DPLL-1a; calculate DT_T_ACT = TS_T - TS_T_OLD calculate RDT_T_ACT calculate QDT_TX according to equation DPLL-2	
3	send CDTI interrupt when NTI_CNT is zero or decrement NTI_CNT when not zero; calculate EDT_T and MEDT_T according to equations DPLL-3 and DPLL-4 for (RMO=1 and SMC=0): update SYN_T, PD_store and go back to step 1	Note: There are different behaviors of RM and HW-IP: For the HW-IP the values of SYN_T and PD_store are not updated until a new active TRIGGER slope occurs.

4	calculate CDT_TX according to equation DPLL-5a and b;	for RMO=0 or SMC=1;

5	calculate missing pulses:
	MP1 = INC_CNT1(at the moment of an active TRIGGER slope)
	calculate target pulses:
	NMB_T_TAR = ((MLT+1)$^{*)}$ *+ PD_store) *SYN_T + MPVAL1
	(instead of PD_store use zero in the case AMT=0)
		for RMO=0 or SMC=1;
		$^{*)}$replace (MLT+1) by MLS1 for SMC=1;
		add MPVAL1 only for PCM=1 and reset PCM1 after that;

6	sent MP with highest possible frequency and set NMB_T = NMB_T_TAR	for RMO=0 or SMC=1, DMO=0 and COA=0
7	calculate the number of pulses to be sent NMB_T = NMB_T_TAR + MP (see equations DPLL-21 or DPLL-27 respectively)	for RMO=0 or SMC=1, DMO=0 and COA=1
8	NMB_T = SYN_T*CNT_NUM_1	for RMO=0 or SMC=1, DMO=1
9	update SYN_T and PD_store;	Note: There are different behaviors of RM and HW-IP: For the HW-IP the values of SYN_T and PD_store are not updated until a new active TRIGGER slope occurs.

10	calculate ADD_IN_CAL1 according to equation DPLL-25 and DPLL-25b or DPLL-31 and store this value in RAM
	use ADD_IN_CAL1 as ADD_IN value for the case DLM=0
	use ADD_IN_LD1 as ADD_IN for the case DLM=1, but do this update immediately (without waiting for this step 10);
	for DMO=DLM=0 and EN_C1u=0:
	reset the flip-flops in the SUB_INC1 generator;
	start sending SUB_INC1;
		for RMO=0 or SMC=1
		for DLM=0
		for DLM=1

11	calculate TS_T_CHECK = TS_T + DT_T_ACT *(TOV) ;	for RMO=0 or SMC=1;

12	automatic setting of actions masking bits in the DPLL_STATUS register:
	for SMC=0: set CAIP1=CAIP2=1
	for SMC=1: set only CAIP1=1
		steps 12 to 16 are not valid for the combination: (SMC=0 and RMO=1)

13	for all correspondent actions with ACT_N[i]=1 calculate: NA[i] = (PSA[i] - PSTC)/(MLT+1)$^{*)}$for forward direction with w= integer part and b = remainder of the division (fractional part); for backward direction use NA[i] = (PSTC - PSA[i])/(MLT+1)$^{*)}$ and consider in both cases the time base overflow in order to get a positive difference	actions 0...11 for SMC=1 actions 0...23 for SMC=0 depending on ACT_N[i] in **DPLL_ACT_STA** register; replace MLT+1 by MLS1 for SMC=1
14	calculate PDT_T[i] and DTA[i] for up to 24 action values according to equations DPLL-11 and DPLL-12;	actions 0...11 for SMC=1 actions 0...23 for SMC=0
15	calculate TSAC[i] according to equation DPLL-15 and PSAC[i] according to equation DPLL-17	actions 0...11 for SMC=1 actions 0...23 for SMC=0

16	automatic resetting of actions masking bits in the DPLL_STATUS register:
	for SMC=0: set CAIP1=CAIP2=0
	for SMC=1: set only CAIP1=0;
	set the corresponding ACT_N[i] bits in the DPLL_ACT_STA register
		Set ACT_N[i] for all enabled actions concerned:
		0...11 for SMC=1
		0...23 for SMC=0

17	check the relation of the last increment to its predecessor according to the profile and taking into account TOV: set the ITN status bit and reset the corresponding LOCK bit, when not plausible; go to step 18, when no active *TRIGGER* appears **for all following steps 18 to 20:** **go immediately back to step 1, when an active TRIGGER event occurs, interrupt all calculations there and reset all CAIP in that case;** **when going back to step 1:** store TS_T in RAM 2b according to APT_2B; update RAM 2a and RAM 2d	for all conditions

18	wait for a new PMTR value; set the corresponding CAIPx values and go to step 19 in that case	go immediately to step 1 and update the RAM according to step 17 when an active *TRIGGER* event occurs
19	make the requested action calculation according to new PMTR values	go immediately to step 1 and update the RAM according to step 17 when an active *TRIGGER* event occurs
20	reset CAIPx and go back to step 18	go immediately to step 1 and update the RAM according to step 17 when an active *TRIGGER* event occurs
21	When DEN = 0 or SEN=0: make sure that the first active slope of STATE is detected; stay in step 1 until DEN=1, SEN=1 and at least one active *STATE* has been detected (FSD=1); the following steps are performed always (not necessarily in step 21, but also in steps 38 to 40 (when waiting for new PMTR values to be calculated): compare STATE_S with SSL (active slope); for each inactive slope: generate a SISI interrupt; • send missing *STATE* INT when TS_S_CHECK time is reached and set MS=1 (missing *STATE* bits) in that case; do not leave step 21 while no active *STATE* appears. **When an active *STATE* slope appears:** store the actual position stamp at PSSM (value at the STATE event) update RAM by equation DPLL-6a-c (see chapter 18.7.5); **store the actual INC_CNT1/2 at MP1/MP2 respectively as missing pulses** *(instead of calculations in step 25)*	Depending on SSL, SEN, DEN step 21 is leaving with the next *STATE* input; for the steps 22-37: for SMC=1 replace: MLS1 by MLS2, LOCK1 by LOCK2; SUB_INC1 by SUB_INC2; CNT_NUM_1 by CNT_NUM_2; MPVAL1 by MPVAL2; EN_C1u by EN_C2u; **dir_crement** does mean: increment for DIR2=0 decrement for DIR2=1 or DIR1 respectively *) target number of pulses of the last increment (see step 25 ff.) **) add MPVAL1 or MPVAL2 only once, that means as long as PCM1 or PCM2 is set respectively ***) **SGE1_delay** is the value of SGE1 delayed by one active STATE event

	store all relevant configuration bits **X** of the DPLL_CTRL(0,1) Registers in **shadow** registers and consider them for all corresponding calculations of steps 22 to 37 accordingly; the relevant bits are explained in the registers itself for FSD=0: • set PSSC=PSSM • set FSD (first *STATE* detected) • do not increment PSSC • for (RMO=1 and SMC=0) **and SGE1=1**: increment INC_CNT1 by MLS1+MPVAL1**) • for (RMO=1 and SMC=1) **and SGE2=1**: increment INC_CNT2 by MLS2+MPVAL2**) for SYS=0, FSD =1: • **dir_crement** PSSC by NMB_S_TAR*) for (SMC=0 **and SGE1_delay** ***) **=1**) or (SMC=1 and **SGE2_delay** ****) **=1**) • increment INC_CNT1 by MLS1+MPVAL1**) (for SMC=0, **SGE1=1** and RMO=1); • increment INC_CNT2 by MLS2+MPVAL2**) (for SMC=1, **SGE2=1** and RMO=1); • **dir_crement** APS and APS_1C2 for SYS=1 : • **dir_crement** APS and APS_1C3 • **dir_crement** APS_1C2 by SYN_S_OLD • for RMO=1 and SMC=0: for **SGE1_delay** ***) **=1** **dir_crement** PSSC by NMB_S_TAR*) ; for **SGE1=1** increment	****) **SGE2_delay** is the value of SGE2 delayed by one active STATE event *****) PD_S_store = 0 for AMS=0 (see DPLL_CTRL_0 register)

	INC_CNT1 by SYN_S*(MLS1 + PD_S_store) + MPVAL1**) • for RMO=1 and SMC=1: for **SGE2_delay****)=1** **dir_crement** PSSC by NMB_S_TAR*) ; for **SGE2=1** increment INC_CNT2 by SYN_S*(MLS2 + PD_S_store)*****) +MPVAL2**) • within the DPLL_STATUS register: set LOCK1 or 2 bit accordingly;	
22	calculate TS_S according to equations DPLL-6a; calculate DT_S_ACT = TS_S - TS_S_OLD calculate RDT_S_ACT calculate QDT_SX	

23	send CDSI interrupt;
	calculate EDT_S and MEDT_S according to equations DPLL-8 and DPLL-9
	for RMO=0:
	go back to step 21 for RMO=0 and update SYN_S and PD_S_store using the current ADT_S[i] values in that case;
		Note: There are different behaviors of RM and HW-IP: For the HW-IP the values of SYN_S and PD_S_store are not updated until a new active STATE slope occurs.

24	calculate CDT_SX according to equation DPLL-10a and b;	only for RMO=1

25	calculate missing pulses
	- for TBU_CH1:
	MP1 = INC_CNT1(active STATE slope)
	- for TBU_CH2:
	MP2 = INC_CNT2(active STATE slope)
	calculate target number of pulses:
	NMB_S_TAR = (MLS1 + PD_S_store)*SYN_S + PD_S_store +MPVAL1 (for SMC=0)
	NMB_S_TAR = MLS2* (SYN_S + PD_S_store) + MPVAL2 (for SMC=1)
	(instead of PD_S_store use zero in the case AMS=0)
		only for RMO=1
		for SMC=0
		instead of MPVAL1 use zero for PCM1=0
		for SMC=1
		instead of MPVAL2 use zero for PCM2=0;
		add MPVAL1/2 once to INC_CNT1/2 and reset PCM1/2 after that

26	sent MPx with highest possible frequency and set	only for RMO=1, DMO=0 and COA=0

	NMB_S = NMB_S_TAR	
27	calculate number of pulses to be sent according to DPLL-22 or NMB_S = NMB_S_TAR + MPx	only for RMO=1, DMO=0 and COA=1

28	NMB_S = SYN_S*CNT_NUM_1 (SMC=0)
	NMB_S = SYN_S*CNT_NUM_2 (SMC=1)
		only for RMO=1,
		DMO=1

29	update SYN_S and PD_S_store;	Note: There are different behaviors of RM and HW-IP: For the HW-IP the values of SYN_S and PD_S_store are not updated until a new active STATE slope occurs.

30	calculate ADD_IN_CAL2 according to equation DPLL-26 and DPLL-26b or DPLL-31 respectively and store this value in RAM
	use ADD_IN_CAL2 as ADD_IN value for the case DLM=0
	use ADD_IN_LD2 as ADD_IN for the case DLM=1, but do this update immediately (without waiting for this step 30);
	for RMO=1, DMO=DLM=0 and EN_C1u=0 (EN_C1u=0):
	reset the flip-flops in the SUB_INC1 or SUB_INC2 generator respectively;
	start sending SUB_INC1 / SUB_INC2;
		only for RMO=1
		
		for DLM=0
		
		for DLM=1

31	calculate
	TS_S_CHECK = TS_S +
	DT_S _ACT * (SOV);
		only for RMO=1;

32	automatic setting of actions masking bits in the DPLL_STATUS register: CAIP1 and CAIP2 for SMC=0 only CAIP2 for SMC=1	for RMO=1

33	for all actions with ACT_N[i]=0 calculate:
	NA[i] = (PSA[i] - PSSC)/MLS1
	for forward direction with
	w = integer part and
	b = remainder of the division (fractional part)
	for backward direction use
	NA[i] = (PSSC - PSA[i])/(MLS1)
		for SMC=0: 24 actions,
		for SMC=1: 12 actions;
		depending on ACT_N[i] in **DPLL_ACT_STA** register

	and consider in both cases the time base overflow in order to get a positive difference use MLS2 as divider in the case of SMC=1	

34	calculate PDT_S[i] and DTA[i] for up to 24 action values according to equations DPLL-13 and DPLL-14;	only for RMO=1;
		for SMC=0 actions 0...23
		for SMC=1 actions 12...23

35	calculate TSAC[i] according to equation DPLL-18 and PSAC[i] according to equation DPLL-20	for the relevant actions (see above) and RMO=1
36	automatic reset of the actions masking bit CAIP in the DPLL_STATUS register: CAIP1=CAIP2=0 for SMC=0 and only CAIP2=0 for SMC=1 set the corresponding ACT_N[i] bits in the DPLL_ACT_STA register	for the relevant actions (see above) and RMO=1 Set ACT_N[i] and reset ACT_WRi for all enabled actions
37	check the duration of the last increment to its predecessor according to the profile and taking into account SOV: set the ISN status bit and reset the corresponding LOCK bit, when not plausible; go to step 38, when no active *STATE* appears **for all following steps 38 to 40:** **go immediately back to step 21, when an active *STATE* event occurs, interrupt all calculations there and reset all CAIPx in that case;** **when going back to step 21:** store TS_S in RAM 1c2 according to APS_1C2; update RAM 1c1 and RAM 1c4	for all conditions
38	wait for a new PMTR value; set the corresponding CAIPx values and go to step 39 in that case	go immediately to step 21 and update the RAM according to step 37 when an active *STATE* event occurs
39	make the requested action calculation according to new PMTR values	go immediately to step 21 and update the RAM according to step 37 when

		an active *STATE* event occurs
40	reset CAIP and go back to step 38	go immediately to step 21 and update the RAM according to step 37 when an active *STATE* event occurs

18.9 DPLL Interrupt signals

The DPLL provides 27 interrupt lines. These interrupts are shown below.

18.9.1 DPLL Interrupt signals

Signal	Description
DPLL_DCGI_IRQ	Direction change
DPLL_SORI_IRQ	*STATE* is out of range
DPLL_TORI_IRQ	*TRIGGER* is out of range
DPLL_CDSI_IRQ	*STATE* duration calculated for last increment
DPLL_CDTI_IRQ	*TRIGGER* duration calculated for last increment
DPLL_TE4_IRQ	*TRIGGER* event interrupt 4 request[3)]
DPLL_TE3_IRQ	*TRIGGER* event interrupt 3 request[3)]
DPLL_TE2_IRQ	*TRIGGER* event interrupt 2 request[3)]
DPLL_TE1_IRQ	*TRIGGER* event interrupt 1 request[3)]
DPLL_TE0_IRQ	*TRIGGER* event interrupt 0 request[3)]
DPLL_LL2_IRQ	Loss of lock interrupt for SUB_INC2 request
DPLL_GL2_IRQ	Get of lock interrupt for SUB_INC2 request
DPLL_E_IRQ	Error interrupt request
DPLL_LL1_IRQ	Loss of lock interrupt for SUB_INC1 request
DPLL_GL1_IRQ	Get of lock interrupt for SUB_INC1 request
DPLL_W1_IRQ	Write access to RAM region 1b or 1c interrupt request
DPLL_W2_IRQ	Write access to RAM region 2 interrupt request
DPLL_PW_IRQ	Plausibility window violation interrupt of *TRIGGER* request
DPLL_TAS_IRQ	*TRIGGER* active slope while NTI_CNT is zero interrupt request
DPLL_SAS_IRQ	*STATE* active slope interrupt request
DPLL_MT_IRQ	Missing *TRIGGER* interrupt request
DPLL_MS_IRQ	Missing *STATE* interrupt request

DPLL_TIS_IRQ	*TRIGGER* inactive slope interrupt request
DPLL_SIS_IRQ	*STATE* inactive slope interrupt request
DPLL_TAX_IRQ	*TRIGGER* maximum hold time violation interrupt request
DPLL_TIN_IRQ	*TRIGGER* minimum hold time violation interrupt request
DPLL_PE_IRQ	DPLL enable interrupt request
DPLL_PD_IRQ	DPLL disable interrupt request

Note: TEi_IRQ depends on the TINT value in ADT_T[i][1] and is only active when SYT[2] =1.

[1] see RAM region 2 explanations ; see 18.14
[2] see DPLL STATUS register; see 18.12.30
[3] see TINT value in the corresponding ADT_T[i] section of RAM region 2; see 18.14.3

18.10 MCS to DPLL interface

A reduced AEI interface is implemented in the DPLL, which can only be accessed by the MCS Bus Master interface in the same cluster. The purpose of this interface is to enable a faster interchange of data between the MCS and the DPLL, while enabling a certain control over the DPLL internal state machine.

18.10.1 Architecture and organization

The implemented interface has an address width of 4 bits, while the size of the data interface is 24 bits.
The following table shows the implemented AEI addresses from the MCS side. Label RD refers to the label used for the address when reading from the MCS or writing from the DPLL, whereas Label WR refers to the label used for the address when writing from the MCS or reading from the DPLL.

18.10.2 General functionality

In order to have a better understanding of the implications when this interface is used, the following working concepts are informally defined here. They refer to the STATE engine operation when DPLL_CTRL_11.STATE_EXT is set.

Update of ram: Operation which stores TSF_S, DT_S_ACTUAL and RDT_S_ACTUAL back to the RAM and reads the profile.

Calculation of sub-increments: Calculation of DT_S_ACTUAL, RDT_S_ACTUAL, NMB_S and ADD_IN.

Change of direction: Update of profile and its increment or decrement (only in the STATE processor).

Calculation of actions (PMT): Where the calculation of PSAC and TSAC is performed (only in state processor).

If **DPLL_CTRL_11.STATE_EXT is not set**, the DPLL will ignore the data written to this interface from the MCS. The DPLL will not update the interface either and a read done to this interface from the MCS can obtain out-of-date information.

If **DPLL_CTRL_11.STATE_EXT is set**, some modifications are done to the way that the DPLL module works when using the STATE engine.
Up to 128 STATE events can be handled.
RAM1c is not used anymore. Instead, the data needed to perform each of the already described operations is fetched from registers in this interface. The data that would have to be written back to RAM1c is also written to this interface.
In each of the procedures described above, the DPLL will enter in one or more stalled states, in which it will wait for one or more words to be written to MCS2DPLL_DEB15 (STATUS_INFO)

18.10.2.1 Correspondence between STA_S values and their unlocking keywords

Operation	**STA_S value**	**First Keyword**	**Second Keyword**
Update of ram	0b00001_001	0xE	0x1
Calculation of sub-increments	0b00010_000	0xD	0x2
Calculation of actions	0b01110_000	0xC	0x3
Change of direction	0b00000_100	0xB	0x4
Change of direction	0b00000_110	0xB	0x4

The stalled STATE in the DPLL is freed by writing the upper keywords to MCS2DPLL_DEB15. Please note the requirements on DPLL level (described in 18.15) regarding the data on the interface that has to be written by the MCS program. If this data is incorrectly delivered or the STATE state machine is unlocked before delivery, the proper signal processing of the DPLL cannot be assured.

For the particular case of an update of ram after a virtual increment, the data field TSF_S is not calculated completely by the DPLL STATE processing unit. Instead, the values needed in order to fill this data field are provided (18.7.5.6 and 18.7.5.7)

18.10.3 MCS to DPLL Register overview

18.10.3.1 MCS to DPLL Register overview

Address offset (see AppendixB)	Common Label	Label RD	Label WR	Details in section
0x0	MCS2DPLL_DEB0	DT_S_P	DT_S_P1	18.15.1
0x4	MCS2DPLL_DEB1	not used	RDT_S_P1	18.15.2
0x8	MCS2DPLL_DEB2	TS_SX	RDT_S_PQ1	18.15.3
0xC	MCS2DPLL_DEB3	DT_SX	DT_S_PQ	18.15.4
0x10	MCS2DPLL_DEB4	SYN_S_OLD	RDT_S_PQ	18.15.5
0x14	MCS2DPLL_DEB5	M_DW	DT_S_PQ1	18.15.6
0x18	MCS2DPLL_DEB6	not used	ADT_S_P	18.15.7
0x1C	MCS2DPLL_DEB7	RDT_S_P_RD	S_P_RD	18.15.8
0x20	MCS2DPLL_DEB8	not used	TSF_S_P	18.15.9
0x24	MCS2DPLL_DEB9	not used	TSF_S_P_MQ	18.15.10
0x28	MCS2DPLL_DEB10	not used	TSF_S_P_PM_MQ	18.15.11
0x2C	MCS2DPLL_DEB11	not used	TSF_S_P_PM	18.15.12
0x30	MCS2DPLL_DEB12	not used	ADT_S_P1	18.15.13
0x34	MCS2DPLL_DEB13	not used	not used	18.15.14
0x38	MCS2DPLL_DEB14	not used	not used	18.15.15
0x3C	MCS2DPLL_DEB15	not used	STATUS_INFO	18.15.16

18.11 DPLL Register Memory overview

The available registers and the size of the RAM area 2 depends on the chosen device. Please refer to Appendix B.

18.11.1 Available DPLL register overview

Register name	Description	Details in Section
DPLL_CTRL_0	DPLL Control Register 0	18.12.1

DPLL_CTRL_1	DPLL Control Register 1	18.12.2
DPLL_CTRL_2	DPLL Control Register 2 (actions 0-7 enable)	18.12.3
DPLL_CTRL_3	DPLL Control Register 3 (actions 8-15 enable)	18.12.4
DPLL_CTRL_4	DPLL Control Register 4 (actions 16-23 enable)	18.12.5
DPLL_CTRL_5	DPLL Control Register 5 (actions 24-31 enable) [2)]	18.12.6
DPLL_ACT_STA	DPLL ACTION Status Register with connected shadow register	18.12.7
DPLL_OSW	DPLL Offset and switch old/new address register	18.12.8
DPLL_AOSV_2	DPLL Address offset register for APT in RAM region 2	18.12.9
DPLL_APT	DPLL Actual RAM pointer to RAM regions 2a, b and d	18.12.10
DPLL_APS	DPLL Actual RAM pointer to regions 1c1, 1c2 and 1c4	18.12.11
DPLL_APT_2C	DPLL Actual RAM pointer to RAM region 2c	18.12.12
DPLL_APS_1C3	DPLL Actual RAM pointer to RAM region 1c3	18.12.13
DPLL_NUTC	DPLL Number of recent TRIGGER events used for calculations (mod 2*(TNU +1-SYN_NT))	18.12.14
DPLL_NUSC	DPLL Number of recent STATE events used for calculations (e.g. mod 2*(SNU +1-SYN_NS) for SYSF=0)	18.12.15
DPLL_NTI_CNT	DPLL Number of active TRIGGER events to interrupt	18.12.16
DPLL_IRQ_NOTIFY	DPLL Interrupt notification register	18.12.17
DPLL_IRQ_EN	DPLL Interrupt enable register	18.12.18
DPLL_IRQ_FORCINT	DPLL Interrupt force register	18.12.19

DPLL_IRQ_MODE	DPLL Interrupt mode register	18.12.20
DPLL_EIRQ_EN	DPLL Error interrupt enable register	18.12.21
DPLL_INC_CNT1	DPLL Counter for pulses for TBU_CH1_BASE to be sent in automatic end mode	18.12.22
DPLL_INC_CNT2	DPLL Counter for pulses for TBU_CH2_BASE to be sent in automatic end mode when SMC=RMO=1	18.12.23
DPLL_APT_SYNC	DPLL old RAM pointer and offset value for TRIGGER	18.12.24
DPLL_APS_SYNC	DPLL old RAM pointer and offset value for STATE	18.12.25
DPLL_TBU_TS0_T	DPLL TBU_CH0_BASE value at last TRIGGER event	18.12.26
DPLL_TBU_TS0_S	DPLL TBU_CH0_BASE value at last STATE event	18.12.27
DPLL_ADD_IN_LD1	DPLL direct load input value for SUB_INC1	18.12.28
DPLL_ADD_IN_LD2	DPLL direct load input value for SUB_INC2	18.12.29
DPLL_STATUS	DPLL Status Register	18.12.30
DPLL_ID_PMTR_[z] (z:0...31)	DPLL 9 bit ID information for input signals PMT z [3)]	18.12.31
DPLL_CTRL_0_SHADOW_TRIGGER	DPLL shadow register of DPLL_CTRL_0	18.12.32
DPLL_CTRL_0_SHADOW_STATE	DPLL shadow register of DPLL_CTRL_0	18.12.33
DPLL_CTRL_1_SHADOW_TRIGGER	DPLL shadow register of DPLL_CTRL_1	18.12.34
DPLL_CTRL_1_SHADOW_STATE	DPLL shadow register of DPLL_CTRL_1	18.12.35
DPLL_RAM_INI	DPLL initialization control and status for RAMs	18.12.36

[2)] **Note:** This register is not available for all devices. Please refer to appendix B.

[3)] **Note:** The registers DPLL_ID_PMTR 24-31 are not available for all devices. Please refer to appendix B.

18.11.2 RAM Region 1a map description

Memory name	Description	Details in section
PSA[i] (i:0...NOAC-1)	Position/Value request for action i	18.13.1
DLA[i] (i:0...NOAC-1)	Time to react before PSAi	18.13.2
NA[i] (i:0...NOAC-1)	Number of TRIGGER/STATE increments to ACTION i	18.13.3
DTA[i] (i:0...NOAC-1)	Calculated relative time to ACTION i	18.13.4

[1)]**Note:** The values PSA24-31, DLA24-31, NA24-31 and DTA24-31 in RAM 1a are not available for all devices. Please refer to appendix B.

18.11.3 RAM Region 1b map description

Memory name	Description	Details in section
TS_T	Actual signal TRIGGER time stamp register TRIGGER_TS	18.12.37 and 18.12.38
TS_T_OLD	Previous signal TRIGGER time stamp register TRIGGER_TS_OLD	18.12.37 and 18.12.38
FTV_T	Actual signal TRIGGER filter value	18.12.39
TS_S	Actual signal STATE time stamp register STATE_TS	18.12.40 and 18.12.41
TS_S_OLD	Previous signal STATE time stamp register STATE_TS_OLD	18.12.40 and 18.12.41
FTV_S	Actual signal STATE filter value	18.12.42
THMI	TRIGGER hold time min. value	18.12.43
THMA	TRIGGER hold time max. value	18.12.44
THVAL	measured last pulse time from active to inactive TRIGGER slope	18.12.45
TOV	Time out value of TRIGGER, according to the last nominal increment for a missing TRIGGER	18.12.46

TOV_S	Time out value of STATE, according to the last nominal increment for a missing STATE	18.12.47
ADD_IN_CAL1	calculated ADD_IN value for SUB_INC1 generation	18.12.48
ADD_IN_CAL2	calculated ADD_IN value for SUB_INC2 generation	18.12.49
MPVAL1	missing pulses to be added/subtracted directly to SUB_INC1 and INC_CNT1 once	18.12.50
MPVAL2	missing pulses to be added/subtracted directly to SUB_INC2 and INC_CNT2 once	18.12.51
NMB_T_TAR	target number of TRIGGER pulses	18.12.52
NMB_T_TAR_OLD	target number of TRIGGER pulses	18.12.53
NMB_S_TAR	target number of STATE pulses	18.12.54
NMB_S_TAR_OLD	target number of STATE pulses	18.12.55
RCDT_TX	reciprocal value of expected increment duration (T)	18.12.56
RCDT_SX	reciprocal value of expected increment duration (S)	18.12.57
RCDT_TX_NOM	reciprocal value of the expected nominal increment duration (T)	18.12.58
RCDT_SX_NOM	reciprocal value of the expected nominal increment duration (S)	18.12.59
RDT_T_ACT	actual reciprocal value of TRIGGER	18.12.60
RDT_S_ACT	actual reciprocal value of STATE	18.12.61
DT_T_ACT	Duration of last TRIGGER increment	18.12.62
DT_S_ACT	Duration of last STATE increment	18.12.63
EDT_T	Absolute error of prediction for last TRIGGER increment	18.12.64
MEDT_T	Average absolute error of prediction up to the last TRIGGER increment	18.12.65
EDT_S	absolute error of prediction for last STATE increment	18.12.66

MEDT_S	Average absolute error of prediction up to the last STATE increment	18.12.67
CDT_TX	Expected duration of current TRIGGER increment	18.12.68
CDT_SX	Expected duration of current STATE increment	18.12.69
CDT_TX_NOM	Expected nominal duration of current TRIGGER increment (without consideration of missing events)	18.12.70
CDT_SX_NOM	Expected nominal duration of current STATE increment (without consideration of missing events)	18.12.71
TLR	TRIGGER locking range value; the TOR bit in the DPLL_STATUS register is set when violated	18.12.72
SLR	STATE locking range value; the SOR bit is set when violated	18.12.73
PDT_[i] (i:0...NOAC-1)	predicted time to ACTION i	18.12.74
MLS1	Calculated number of sub-pulses between two STATE events (to be set by CPU)	18.12.75
MLS2	Calculated number of sub-pulses between two STATE events (to be set by CPU) for the use when SMC=RMO=1	18.12.76
CNT_NUM_1	number of sub-pulses of SUB_INC1 in continuous mode, updated by the host only	18.12.77
CNT_NUM_2	number of sub-pulses of SUB_INC2 in continuous mode, updated by the host only	18.12.78
PVT	Plausibility value of next active TRIGGER slope	18.12.79
PSTC	Accurate calculated position stamp of last TRIGGER input;	18.12.80
PSSC	Accurate calculated position stamp of last STATE input;	18.12.81
PSTM	Measured position stamp at last active TRIGGER input	18.12.82 and 18.12.83
PSTM_OLD	Measured position stamp at last but one active TRIGGER input	18.12.82 and 18.12.83

PSSM	Measured position stamp at last active STATE input	18.12.84 and 18.12.85
PSSM_OLD	Measured position stamp at last but one active STATE input	18.12.84 and 18.12.85
NMB_T	Number of pulses of current increment in normal mode for SUB_INC1 (see equation DPLL-21 or for SMC=1 equation DPLL-27 respectively)	18.12.86
NMB_S	Number of pulses of current increment in emergency mod for SUB_INC1 (see equation DPLL-22) or in the case SMC=1 for SUB_INC2 (see equation DPLL-28)	18.12.87

[2)] **Note:** The values PDT_24 to PDT_31 in RAM1b are not available for all devices. Please refer to appendix B.

18.11.4 RAM Region 1c map description

Memory name	Description	Details in section
RDT_S[i] (i:0...63)	Part of RAM1c1. Reciprocal value of the corresponding successive increment i, for each true nominal increment.	18.12.88
TSF_S[i] (i:0...63)	Part of RAM1c2. Time stamp field for state events, for each true nominal increment plus each virtual increment.	18.12.89
ADT_S[i] (i:0...63)	Part of RAM1c3. Adapt values for the current STATE increment, for each true nominal increment.	18.12.90
DT_S[i] (i:0...63)	Part of RAM1c4. Uncorrected last increment value of STATE for full scale, for each true nominal increment.	18.12.91

18.11.5 Register Region EXT description

Register name	Description	Details in

DPLL_TSAC[z] (z:0...NOAC-1)	DPLL calculated action time stamps for action z	18.12.92
DPLL_PSAC[z] (z:0...NOAC-1)	DPLL calculated action position stamps for action z	18.12.93
DPLL_ACB_[z] (z:0...(NOAC/4)-1)	DPLL control bits for actions ((4*z)...(4*z)+3)	18.12.94
DPLL_CTRL_11	DPLL control register	18.12.95
DPLL_THVAL2	DPLL immediate THVAL value	18.12.96
DPLL_TIDEL	DPLL additional TRIGGER input delay	18.12.97
DPLL_SIDEL	DPLL additional STATE input delay	18.12.98
DPLL_CTN_MIN	CDT_T_NOM minimum value	18.12.99
DPLL_CTN_MAX	CDT_T_NOM maximum value	18.12.100
DPLL_CSN_MIN	CDT_S_NOM minimum value	18.12.101
DPLL_CSN_MAX	CDT_S_NOM maximum value	18.12.102
DPLL_STA	DPLL state machine status information	18.12.103
DPLL_INCF1_OFFSET	DPLL ADD_IN_ADDER1 offset for fast pulse generation	18.12.104
DPLL_INCF2_OFFSET	DPLL ADD_IN_ADDER2 offset for fast pulse generation	18.12.105
DPLL_DT_T_START	DPLL first value of DPLL_DT_T_ACT for the first increment after setting SIP1 from 0 to 1.	18.12.106
DPLL_DT_S_START	DPLL first value of DPLL_DT_S_ACT for the first increment after setting SIP2 from 0 to 1.	18.12.107
DPLL_STA_MASK	DPLL trigger masks for signals DPLL_STA_T and DPLL_STA_S	18.12.108
DPLL_STA_FLAG	DPLL STA_T/S and INC_CNT1/2 flags	18.12.109
DPLL_INC_CNT1_MASK	DPLL INC_CNT1 trigger mask	18.12.110
DPLL_INC_CNT2_MASK	DPLL INC_CNT2 trigger mask	18.12.111

DPLL_NUSC_EXT1	Extension register number 1 for DPLL_NUSC [4)]	18.12.112
DPLL_NUSC_EXT2	Extension register number 2 for DPLL_NUSC [4)]	18.12.113
DPLL_APS_EXT	Extension register for DPLL_APS [4)]	18.12.114
DPLL_APS_1C3_EXT	Extension register for DPLL_APS_1C3 [4)]	18.12.115
DPLL_APS_SYNC_EXT	Extension register for DPLL_APS_SYNC [4)]	18.12.116
DPLL_CTRL_EXT	Extension register for DPLL_CTRL [4)]	18.12.117

[4)] **Note:** These registers will return AEI_STATUS = b#10 if DPLL_CTRL_11.STATE_EXT is not set.

18.11.6 RAM Region 2 map description

Memory name	Description	Details in section
RDT_T[i] (i:0...AOSV_2B/4-1)	Region 2a. Reciprocal value of the corresponding successive increment i, for each true nominal increment.	18.14.1
TSF_T[i] (i:0...AOSV_2B/4-1)	Region 2b. Time Stamp Field for TRIGGER event i, for each true nominal increment plus each virtual increment.	18.14.2
ADT_T[i] (i:0...AOSV_2B/4-1)	Region 2c. Adapt values for the current TRIGGER increment i, for each true nominal increment.	18.14.3
DT_T[i] (i:0...AOSV_2B/4-1)	Region 2d. Uncorrected last increment value of TRIGGER i, for each true nominal increment.	18.14.4

Note: For each of the regions, the maximum number of entries is restricted to a value corresponding to the OSS value in the DPLL_OSW register.

The description of registers is beginning at the register DPLL_CTRL_0. The description of RAM regions is beginning at RAM 1a (see below):

Bits 31 down to 24 in each RAM region are not implemented and therefore always read as zero (reserved). Other bits which are declared as reserved are not protected against writing. Reserved address regions are not protected against writing.

The description of the memory region RAM 1a begins with memory element PSA[i]:
The RAM region 1a is writeable only for DEN=0 (see DPLL_CTRL_1 register).

The description of memory region RAM1b begins with memory element TS_T.
The description of memory region RAM1c begins with memory element RDT_S.
The description of register region EXT begins with the register DPLL_TSAC[z]:
This is an extension of the normal register region above in order to allow up to 32 action calculations and later specification modifications.

The description of the memory region RAM 2 begins with memory element RDT_T.

18.12 DPLL Register and Memory description

18.12.1 Register DPLL_CTRL_0

Address Offset:	**see Appendix B**																**Initial Value: 0x003B_BA57**															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	RMO	TEN	SEN	IDT	IDS	AMT	AMS	TNU									SNU					IFP	MLT									
Mode	RW	RW	RW	RW	RW	RW	RW	RPw									RPw					RW	RW									
Initial Value	0	0	0	0	0	0	0	0x03B									0x17					0	0x257									

Control Register 0

Bit 9:0 **MLT $^{1)}$: multiplier for TRIGGER;** MLT+1 is number of *SUB_INC1* pulses between two *TRIGGER* events in normal mode (1...1024);

Note: For emergency mode the number of *SUB_INC1* pulses between two *STATE* events is calculated by the CPU using the formula MLS1=(MLT+1)* (TNU+1) / (SNU+1) in order to get the same number of *SUB_INC1* pulses for FULL_SCALE. This value is stored in RAM at 0x05C0. Change of MLT by the CPU must result in the corresponding change of MLS1 by the CPU for SMC=0.

Note: The number of MLT events is the binary value plus 1. The value MLT+1 is replaced by MLS1 in the case of SMC=1 (see DPLL_CTRL_1 register) for all relevant calculations.

Bit 10 **IFP [1),2),4)]: Input filter position;** value contains position or time related information.

0 = TRIGGER_FT and STATE_FT mean time related values, that means the number of time stamp clocks

1 = TRIGGER_FT and STATE_FT mean position related values, that means the number of SUB_INC1 (or SUB_INC2 in the case SMC=1) pulses respectively

Bit 15:11 **SNU[3)]: STATE number;** SNU+1 is number of nominal *STATE* events in HALF_SCALE (1...32).

Note: The number of nominal *STATE* events is the decimal value plus 1. This value can only be written when (RMO=0 and SMC=0) or DEN=0. To make sure that this signal is not changed during a mode change RMO=0 means that the status of RMO=0 must be given before and during writing to the register.

Set SSL=00 before changing this value and set RMO=1 only after FULL_SCALE with SSL>0.

Note: This register can only be written when DPLL_CTRL_11.STATE_EXT is not set. If DPLL_CTRL_11.STATE_EXT is set, the signal cannot be written, the read value is zero.

Bit 24:16 **TNU[3)]: TRIGGER number;** TNU+1 is number of nominal *TRIGGER* events in HALF_SCALE (1...512).

Note: The number of nominal *TRIGGER* events is the decimal value plus 1.This value can only be written when (RMO=1 and SMC=0) or DEN=0.

To make sure that this signal is not changed during a mode change RMO=0 means that the status of RMO=0 must be given before and during writing to the register.

Set TSL=00 before changing this value and set RMO=0 only after FULL_SCALE with TSL>0.

Bit 25 **AMS [2)]: Adapt mode STATE;** Use of adaptation information of *STATE*.

0 = No adaptation information is used for *STATE*

1 = Immediate adapting mode; the values for physical deviation PD_S of ADT_S[i] are considered to calculate SUB_INC1 pulses in emergency mode (SMC=0) or SUB_INC2 pulses for SMC=1

Bit 26 **AMT [1)]: Adapt mode TRIGGER;** Use of adaptation information of *TRIGGER*.

0 = No adaptation information for *TRIGGER* is used

1 = Immediate adapting mode; the values for physical deviation PD of ADT_T[i] are considered to calculate the SUB_INC1 pulses in normal mode and for SMC=1

Bit 27 **IDS [2]: Input delay STATE;** Use of input delay information transmitted in FT part of the *STATE* signal.
0 = Delay information is not used
1 = Up to 24 bits of the FT part contain the delay value of the input signal, concerning the corresponding edge

Bit 28 **IDT [1]: Input delay TRIGGER;** use of input delay information transmitted in FT part of the *TRIGGER* signal.
0 = Delay information is not used
1 = Up to 24 bits of the FT part contain the delay value of the input signal, concerning the corresponding edge

Bit 29 **SEN: *STATE* enable.**
0 = *STATE* signal is not enabled (no signal considered)
1 = *STATE* signal is enabled

Bit 30 **TEN: *TRIGGER* enable.**
0 = *TRIGGER* signal is not enabled (no signal considered)
1 = *TRIGGER* signal is enabled

Bit 31 **RMO [1],[2]: Reference mode;** selection of the relevant the input signal for generation of SUB_INC1.
0 = Normal mode; the signal *TRIGGER* is used to generate the *SUB_INC1* signals
1 = Emergency mode for SMC=0; signal *STATE* is used to generate the *SUB_INC1* signals ; Double synchronous mode for SMC=1: signal *TRIGGER* is used to generate the *SUB_INC1* signals and *STATE* is used to generate the *SUB_INC2* signals

Note: for SMC=0: *TRIGGER* and *STATE* are prepared to calculate SUB_INC1. The RMO bit gives a decision only, which of them is used. For changing from normal mode to emergency mode at the following TRIGGER slope (according to the RMO value in the shadow register)[1] the PSSC value is calculated by PSSC = PSSM + correction value (forward direction) or PSSC = PSSM - correction value (backward direction) with the correction value = inc_cnt1 - nmb_t. For changing from emergency mode to normal mode at the following STATE slope (according to the RMO value in the shadow register)[2] the PSTC value is calculated by PSTC = PSTM + correction value (forward direction) or PSTC = PSTM - correction value (backward direction) with the correction value = inc_cnt1 - nmb_s. In case no further TRIGGER or STATE events the CPU has to perform the above corrections.

[1] stored in an independent shadow register for an active *TRIGGER* event and for DEN = 1.

[2] stored in an independent shadow register for an active *STATE* event and for DEN = 1.
[3] the time between two active *STATE* or *TRIGGER* events must be always greater then 23,4 µs; in addition the TS_CLK and the resolution must be chosen such that for each nominal increment the time stamps at the beginning and the end of the increment differ at least in the value of 257
[4] for IFP=1 the time between two active *TRIGGER* or *STATE* events must be always greater then 2,34 ms and the value x of MLT, MLS1 or MLS2 must be chosen such that the number of time stamp pulses between two SUB_INC events must be less than 65536. This is fulfilled when x is greater than 256.

18.12.2 Register DPLL_CTRL_1

Address Offset:	see Appendix B																Initial Value: 0xB000_0000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	TSL		SSL		SMC	TS0_HRT	TS0_HRS	SYSF	SWR	LCD	SYN_NT						SYN_NS					PCM2	DLM2	SGE2	PCM1	DLM1	SGE1	PIT	COA	IDDS	DEN	DMO
Mode	RPw		RPw		RPw	RPw	RPw	RPw	RAw	RPw	RPw						RPw					RW	RW	RW	RW	RW	RW	RW	RW	RPw	RW	RW
Initial Value	0x2		0x3		0	0	0	0	0x0	0x0	0x00						0x00					0	0	0	0	0	0	0	0	0	0	0

Control Register 1

Bit 0 **DMO [1], [2]: DPLL mode select.**

0 = **Automatic end mode**; if the number of pulses for an increment is reached, no further pulse is generated until the next active *TRIGGER*/*STATE* is received; in the case of getting a new active *TRIGGER*/*STATE* before the defined number of pulses is reached, the pulse frequency is changed according to the conditions described below **(COA)**.

1 = **Continuous mode**; in this mode a difference between the predefined number of pulses and the actual number of generated pulses can influence the pulse frequency by writing a corresponding pulse number into CNT_NUM_1 or CNT_NUM_2 respectively in RAM region 1b.

Bit 1 **DEN: DPLL enable.**

0 = The DPLL is not enabled; Disabling the DPLL will result in a reset state of the DPLL_STATUS register which remains in this state until DEN=1. No DPLL related interrupt will be generated in that case.

1 = The DPLL is enabled.

Note: The bits 31 down to 0 of the DPLL_STATUS register are cleared, when the DPLL is disabled. Some bits of the control registers can be set only when DEN=0. The protected bits in the DPLL_CTRL_1 register cannot be written when simultaneously DEN is set to 1.

Bit 2 **IDDS: Input direction detection strategy in the case of SMC=0**

0 = The input direction is detected comparing the THMI value with the duration between active and inactive slope of TRIGGER.

1 = The input direction is detected using TDIR input signal also in the case SMC=0.

Note: This bit can only be written when the DPLL is disabled and be fixed to zero, when not needed for an implementation. Independent of the value of IDDS is the direction information for TRIGGER in the case SMC=0 always considered at the moment when the inactive slope appears.

Bit 3 **COA 1), 2): Correction strategy in automatic end mode** (DMO=0).

0 = The pulse frequency of the CMU_CLK0 will be used to make up for missing pulses from last increment; the output of the calculated new pulses will start after resetting the FFs in the pulse generation unit. The frequency of CMU_CLK0 should not exceed half the frequency of the system clock (see 18.8.3.1)

1 = missing pulses of the last increment are distributed evenly to the next increment, calculations are done when the next active input event appears. The number of missing sub-pulses will be determined by the pulse counter difference between the last two active *TRIGGER/STATE* events respectively; the FFs in the pulse generation unit are not reset before sending new pulses.

Note: For SMC=RMO=1: **COA** is used for SUB_INC1 **and** SUB_INC2

Bit 4 **PIT 1): Plausibility** value PVT to next active TRIGGER is **time related**

0 = the plausibility value is position related (PVT contains the number of SUB_INC1 pulses)

1 = the plausibility value is time related (the PVT value is to be multiplied with the duration of the last increment DT_T_ACT and divided by 1024)

Bit 5 **SGE1 1), 2): SUB_INC1 generator enable.**

0 = The SUB_INC1 generator is not enabled

1 = The SUB_INC1 generator is enabled

Bit 6 **DLM1 1),2): Direct Load Mode** for SUB_INC1 generation

0 = the DPLL uses the calculated ADD_IN_CAL value for the SUB_INC1 generation

1 = the ADD_IN_LD value is used for the SUB_INC1 generation and is provided by the CPU; the value remains valid until the CPU writes a new one; the calculated ADD_IN values are stored as ADD_IN_CAL in the RAM at different locations for normal and emergency mode

Bit 7 **PCM1**[1),2),3)]**: Pulse Correction Mode** for SUB_INC1 generation.

0 = the DPLL does not use the correction value stored in MPVAL1

1 = the DPLL uses the correction value stored in MPVAL1 in normal and emergency mode

Bit 8 **SGE2**[2)]**: SUB_INC2 generator enable**.

0 = The SUB_INC2 generator is not enabled.

1 = The SUB_INC2 generator is enabled

Bit 9 **DLM2**[2)]**: Direct Load Mode** for SUB_INC2 generation

0 = the DPLL uses the calculated ADD_IN_CAL value for the SUB_INC2 generation.

1 = the ADD_IN_LD value is used for the SUB_INC2 generation and is provided by the CPU; the value remains valid until the CPU writes a new one; the calculated ADD_IN values are stored as ADD_IN_CAL in the RAM at different locations for normal and emergency mode

Bit 10 **PCM2**[2),3)]**: Pulse Correction Mode** for SUB_INC2 generation.

0 = the DPLL does not use the correction value stored in MPVAL2.

1 = the DPLL uses the correction value stored in MPVAL2

Bit 15:11 **SYN_NS: Synchronization number of *STATE*;** summarized number of virtual increments in HALF_SCALE

sum of all systematic missing *STATE* events in HALF_SCALE (for SYSF=0) or FULL SCALE (for SYSF=1) ; the SYN_NS missing *STATES* can be divided up to an arbitrary number of blocks. The pattern of events and missing events in FULL_SCALE is shown in RAM region 1c3 as value NS in addition to the adapted values. The number of stored increments in FULL_SCALE must be equal to 2*(SNU+1-SYN_NS) for SYSF=0 or 2*(SNU+1)-SYN_NS for SYSF=1 . This pattern is written by the CPU beginning from a fixed reference point (maybe beginning of the FULL_SCALE region). The relation to the actual increment is established by setting of the profile RAM pointer APS_1C3 in an appropriate relation to the RAM pointer APS of the actual increment by the CPU.

Note: This value can only be written when (RMO=0 and SMC=0) or DEN=0. Set SSL=00 before changing this value and set RMO=1 only after FULL_SCALE with SSL>0. To make shure that this signal is not changed during a mode change SMC=0 means that the status of SMC=0 must be given before and during writing to the register.

Note: This register can only be written when DPLL_CTRL_11.STATE_EXT is not set. If DPLL_CTRL_11.STATE_EXT is set, the signal cannot be written, the read value is zero.

Bit 21:16 **SYN_NT: Synchronization number of *TRIGGER*;** summarized number of virtual increments in HALF_SCALE

sum of all systematic missing *TRIGGER* events in HALF_SCALE; the SYN_NT missing *TRIGGER* can be divided up to an arbitrary number of blocks. The pattern of events and missing events in FULL_SCALE is shown in RAM region 2c as value NT in addition to the adapted values. The number of stored increments in FULL_SCALE must be equal to 2*(TNU-SYN_NT). This pattern is written by the CPU beginning from a fixed reference point (maybe beginning of the FULL_SCALE region). The relation to the actual increment is established by setting of the profile RAM pointer APT_2C in an appropriate relation to the RAM pointer APT of the actual increment by the CPU.

Note: This value can only be written when (RMO=1 and SMC=0) or DEN=0. Set TSL=00 before changing this value and set RMO=0 only after FULL_SCALE with TSL>0. To make shure that this signal is not changed during a mode change SMC=0 means that the status of SMC=0 must be given before and during writing to the register.

Bit 22 **LCD: Locking condition definition**

0 = locking condition definition is one times missing TRIGGERs as expected by the profile in HALF_SCALE (one gap).

1 = locking condition definition is n-1 times missing TRIGGERs as expected by the profile in HALF_SCALE (one additional tooth)

Note: This bit can only be written when the DPLL is disabled and be fixed to zero, when not needed for an implementation.

Bit 23 **SWR: Software reset**

resets all register and internal states of the DPLL

0 = no software reset enabled

1 = software reset enabled

Note: Setting the SWR bit results only in a software reset when the DPLL is not enabled (DEN=0).

Bit 24 **SYSF: SYN_NS for FULL_SCALE**

the value SYN_NS does mean the sum of all systematic missing *STATE* events in HALF_SCALE (for SYSF=0) or FULL SCALE (for SYSF=1).

0 = the SYN_NS value is valid for HALF_SCALE
1 = the SYN_NS value is valid for FULL_SCALE

Note: This value can only be written when (RMO=0 and SMC=0) or DEN=0. Set SSL=00 before changing this value and set RMO=1 only after FULL_SCALE with SSL>0. To make shure that this signal is not changed during a mode change SMC=0 means that the status of SMC=0 must be given before and during writing to the register.

Bit 25 **TS0_HRS: Time stamp high resolution STATE**

0 = the resolution of the used DPLL input TBU_TS0 bits is equal to the *STATE* input time stamp resolution
1 = the *STATE* input time stamps have a 8 times higher resolution as the TBU_TS0 DPLL input

Note: This bit can only be written when the DPLL is disabled.

Bit 26 **TS0_HRT: Time stamp high resolution TRIGGER**

0 = the resolution of the used DPLL input TBU_TS0 bits is equal to the *TRIGGER* input time stamp resolution
1 = the *TRIGGER* input time stamps have a 8 times higher resolution as the TBU_TS0 input

Note: This bit can only be written when the DPLL is disabled.

Bit 27 **SMC: Synchronous Motor Control**

0 = *TRIGGER* and *STATE* inputs are used for a control different to SMC.
1 = the *TRIGGER* input reflects a combined sensor signal for SMC and in the case of RMO=1 also *STATE* reflects a different combined sensor signal

Note: This bit can only be written when the DPLL is disabled.

Bit 29:28 **SSL: STATE slope select;** Definition of active slope for signal *STATE* each active slope is an event defined by SNU. Set by DEN=0 only.

If DPLL_STATUS.FSD = '1'
"slope sensitive after detection of first *STATE* input signal"

0b00 =	No slope of *STATE* will be used; this value makes only sense in normal mode
0b01 =	Low high slope will be used as active slope, only inputs with a signal value of "1" will be considered
0b10 =	High low slope will be used as active slope, only inputs with a signal value of "0" will be considered
0b11 =	Both slopes will be used as active slopes

If DPLL_STATUS.FSD = '0'
"level sensitive for first *STATE* input signal edge"

0b00 =	No input signal of *STATE* will be used; this value makes only sense in normal mode
0b01 =	"high" input signal level will be used as active slope, only inputs with a signal value of "1" will be considered
0b10 =	"low" input signal level will be used as active slope, only inputs with a signal value of "0" will be considered
0b11 =	Both input signal levels will be used as active slopes

Note: This value can only be written when (RMO=0 and SMC=0) or DEN=0. To make shure that this signal is not changed during a mode change SMC=0 means that the status of SMC=0 must be given before and during writing to the register.

Bit 31:30 **TSL: TRIGGER slope select;** Definition of active slope for signal *TRIGGER* each active slope is an event defined by TNU. Set by DEN=0 only.

If DPLL_STATUS.FTD = '1'
"slope sensitive after detection of first *TRIGGER* input signal"

0b00 =	No slope of *TRIGGER* will be used; this value makes only sense in emergency mode
0b01 =	Low high slope will be used as active slope, only inputs with a signal value of "1" will be considered
0b10 =	High low slope will be used as active slope, only inputs with a signal value of "0" will be considered
0b11 =	Both slopes will be used as active slopes

If DPLL_STATUS.FTD = '0'
"level sensitive for first *TRIGGER* input signal edge"

0b00 =	No input signal of *TRIGGER* will be used; this value makes only sense in normal mode
0b01 =	"high" input signal level will be used as active slope, only inputs with a signal value of "1" will be considered
0b10 =	"low" input signal level will be used as active slope, only inputs with a signal value of "0" will be considered
0b11 =	Both input signal levels will be used as active slopes

Note: This value can only be written when (RMO=1 and SMC=0) or DEN=0. To make shure that this signal is not changed during a mode change SMC=0 means that the status of SMC=0 must be given before and during writing to the register.

[1] stored in an independent shadow register for an active *TRIGGER* event and for DEN = 1.

[2] stored in an independent shadow register for an active *STATE* event and for DEN = 1.

[3] Bit is cleared, when transmitted to shadow register

18.12.3 Register DPLL_CTRL_2

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								WAD7	WAD6	WAD5	WAD4	WAD3	WAD2	WAD1	WAD0	AEN7	AEN6	AEN5	AEN4	AEN3	AEN2	AEN1	AEN0	Reserved							
Mode	R								RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw	RPw	RPw	RPw	RPw	RPw	RPw	RPw	RPw	R							
Initial Value	0x00								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0x00							

Action Enable Register

Bit 7:0 **Reserved**
Note: Read as zero, should be written as zero.

Bit 8 **AEN0 [1]:** ACTION_0 enable.
0 = the corresponding action is not enabled
1 = the corresponding action is enabled

Bit 9 **AEN1 [1]:** ACTION_1 enable.
see bit 8

Bit 10 **AEN2 [1]:** ACTION_2 enable.
see bit 8

Bit 11 **AEN3 [1]:** ACTION_3 enable.
see bit 8

Bit 12 **AEN4 [1]:** ACTION_4 enable.
see bit 8

Bit 13 **AEN5 [1]:** ACTION_5 enable.

	see bit 8
Bit 14	**AEN6 [1]:** ACTION_6 enable. see bit 8
Bit 15	**AEN7 [1]:** ACTION_7 enable. see bit 8
Bit 16	**WAD0:** Write control bit of Action_0. 0 = the corresponding AENi bit is not writeable 1 = the corresponding AENi bit is writeable
Bit 17	**WAD1:** Write control bit of Action_1. see bit 16
Bit 18	**WAD2:** Write control bit of Action_2. see bit 16
Bit 19	**WAD3:** Write control bit of Action_3. see bit 16
Bit 20	**WAD4:** Write control bit of Action_4. see bit 16
Bit 21	**WAD5:** Write control bit of Action_5. see bit 16
Bit 22	**WAD6:** Write control bit of Action_6. see bit 16
Bit 23	**WAD7:** Write control bit of Action_7. see bit 16
Bit 31:24	**Reserved** Note: Read as zero, should be written as zero.

[1] **Note**: This bit can be written only if the correspondent WADi Bit is set in the same access. It can be set for debug purposes by CPU also, when DPLL is disabled. The enable bit becomes active only when the DPLL is in operation (DEN=1).

Note: For WADi =1 only the corresponding AENi bits are writable. The AENi bits remain unchanged when the corresponding WADi=0.

18.12.4 Register DPLL_CTRL_3

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								WAD15	WAD14	WAD13	WAD12	WAD11	WAD10	WAD9	WAD8	AEN15	AEN14	AEN13	AEN12	AEN11	AEN10	AEN9	AEN8	Reserved							
Mode	R								RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw	RPw	RPw	RPw	RPw	RPw	RPw	RPw	RPw	R							
Initial Value	0x00								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0x00							

Action Enable Register

Bit 7:0 **Reserved**
Note: Read as zero, should be written as zero.

Bit 8 **AEN8 [1]:** ACTION_8 enable.
0 = the corresponding action is not enabled
1 = the corresponding action is enabled

Bit 9 **AEN9 [1]:** ACTION_9 enable
see bit 8

Bit 10 **AEN10 [1]:** ACTION_10enable.
see bit 8

Bit 11 **AEN11 [1]:** ACTION_11 enable.
see bit 8

Bit 12 **AEN12 [1]:** ACTION_12 enable.
see bit 8

Bit 13 **AEN13 [1]:** ACTION_13 enable.
see bit 8

Bit 14 **AEN14 [1]:** ACTION_14 enable.
see bit 8

Bit 15 **AEN15 [1]:** ACTION_15 enable.
see bit 8

Bit 16 **WAD8:** Write control bit of Action_8.
0 = the corresponding AENi bit is not writeable
1 = the corresponding AENi bit is writeable

Bit 17 **WAD9:** Write control bit of Action_9.
see bit 16

Bit 18 **WAD10:** Write control bit of Action_10.
see bit 16

Bit 19 **WAD11:** Write control bit of Action_11.
see bit 16

Bit 20 **WAD12:** Write control bit of Action_12.
see bit 16

Bit 21 **WAD13:** Write control bit of Action_13.
see bit 16

Bit 22 **WAD14:** Write control bit of Action_14.

see bit 16

Bit 23 **WAD15:** Write control bit of Action_15.
see bit 16

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

[1] **Note**: This bit can be written only if the correspondent WADi Bit is set in the same access. It can be set for debug purposes by CPU also, when DPLL is disabled. The enable bit becomes active only when the DPLL is in operation (DEN=1).

Note: For WADi =1 only the corresponding AENi bits are writable. The AENi bits remain unchanged when the corresponding WADi=0.

18.12.5 Register DPLL_CTRL_4

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								WAD23	WAD22	WAD21	WAD20	WAD19	WAD18	WAD17	WAD16	AEN23	AEN22	AEN21	AEN20	AEN19	AEN18	AEN17	AEN16	Reserved							
Mode	R								RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw	RPw	RPw	RPw	RPw	RPw	RPw	RPw	RPw	R							
Initial Value	0x00								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0x00							

Action Enable Register

Bit 7:0 **Reserved**
Note: Read as zero, should be written as zero.

Bit 8 **AEN16 [1]:** ACTION_16 enable.
0 = the corresponding action is not enabled
1 = the corresponding action is enabled

Bit 9 **AEN17 [1]:** ACTION_17 enable
see bit 8

Bit 10 **AEN18 [1]:** ACTION_18 enable.
see bit 8

Bit 11 **AEN19 [1]:** ACTION_19 enable.
see bit 8

Bit 12 **AEN20 [1]:** ACTION_20 enable.
see bit 8

Bit 13 **AEN21 [1]:** ACTION_21 enable.
see bit 8

Bit 14 **AEN22 [1]:** ACTION_22 enable.
see bit 8

Bit 15 **AEN23** [1]**:** ACTION_23 enable.
see bit 8

Bit 16 **WAD16:** Write control bit of Action_16.
0 = the corresponding AENi bit is not writeable
1 = the corresponding AENi bit is writeable

Bit 17 **WAD17:** Write control bit of Action_17.
see bit 16

Bit 18 **WAD18:** Write control bit of Action_18.
see bit 16

Bit 19 **WAD19:** Write control bit of Action_19.
see bit 16

Bit 20 **WAD20:** Write control bit of Action_20.
see bit 16

Bit 21 **WAD21:** Write control bit of Action_21.
see bit 16

Bit 22 **WAD22:** Write control bit of Action_22.
see bit 16

Bit 23 **WAD23:** Write control bit of Action_23.
see bit 16

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

[1] **Note**: This bit can be written only if the correspondent WADi Bit is set in the same access. It can be set for debug purposes by CPU also, when DPLL is disabled. The enable bit becomes active only when the DPLL is in operation (DEN=1).

Note: For WADi =1 only the corresponding AENi bits are writable. The AENi bits remain unchanged when the corresponding WADi=0.

18.12.6 Register DPLL_CTRL_5

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_0000**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								WAD31	WAD30	WAD29	WAD28	WAD27	WAD26	WAD25	WAD24	AEN31	AEN30	AEN29	AEN28	AEN27	AEN26	AEN25	AEN24	Reserved							
Mode	R								RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw	RPw	RPw	RPw	RPw	RPw	RPw	RPw	RPw	R							
Initial Value	0x00								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0x00							

Action Enable Register

Bit 7:0 **Reserved**

	Note: Read as zero, should be written as zero.
Bit 8	**AEN24 1):** ACTION_24 enable. 0 = the corresponding action is not enabled 1 = the corresponding action is enabled
Bit 9	**AEN25 1):** ACTION_25 enable see bit 8
Bit 10	**AEN26 1):** ACTION_26 enable. see bit 8
Bit 11	**AEN27 1):** ACTION_27 enable. see bit 8
Bit 12	**AEN28 1):** ACTION_28 enable. see bit 8
Bit 13	**AEN29 1):** ACTION_29 enable. see bit 8
Bit 14	**AEN30 1):** ACTION_30 enable. see bit 8
Bit 15	**AEN31 1):** ACTION_31 enable. see bit 8
Bit 16	**WAD24:** Write control bit of Action_24. 0 = the corresponding AENi bit is not writeable 1 = the corresponding AENi bit is writeable
Bit 17	**WAD25:** Write control bit of Action_25. see bit 16
Bit 18	**WAD26:** Write control bit of Action_26. see bit 16
Bit 19	**WAD27:** Write control bit of Action_27. see bit 16
Bit 20	**WAD28:** Write control bit of Action_28. see bit 16
Bit 21	**WAD29:** Write control bit of Action_29. see bit 16
Bit 22	**WAD30:** Write control bit of Action_30. see bit 16
Bit 23	**WAD31:** Write control bit of Action_31. see bit 16
Bit 31:24	**Reserved** Note: Read as zero, should be written as zero.

1) **Note**: This bit can be written only if the correspondent WADi Bit is set in the same access. It can be set for debug purposes by CPU also, when DPLL is disabled. The enable bit becomes active only when the DPLL is in operation (DEN=1).

Note: For WADi =1 only the corresponding AENi bits are writable. The AENi bits remain unchanged when the corresponding WADi=0.

18.12.7 Register DPLL_ACT_STA

Address Offset:	see Appendix B																Initial Value: 0x0000_0000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	ACT_N																															
Mode	RPw																															
Initial Value	0x000000																															

Action Status Register including Shadow Register

Bit 31:0 **ACT_N[i],(i=0...NOAC-1):** New output data values concerning to action i provided

0 = no new output data available after a recent PMT request or actual event value is in the past or invalid.

1 = either new PMTR data received or calculation is repeated to be more precise by taking into account new *TRIGGER* or *STATE* values

Note: ACT_N[i] is .

- set (for AENi=1 and a new valid PMTR), that means when new action data are to be calculated for the correspondent action. After each calculation of the new actions values the ACT_N[i] bit updates the corresponding bit in the connected shadow register. The status of the ACT_N[i] bits in the shadow register is reflected by the corresponding DPLL output signal ACT_V (valid bit).
- reset together with the corresponding shadow register bit for AENi=0;
- reset without the corresponding shadow register bit when the calculated event is in the past (the shadow register bit is set, when it was not set before in that case)
- the corresponding shadow register bit is reset, when new PMTR data are written or when the provided action data are read (blocking read)
- writeable for debugging purposes together with the corresponding shadow register when DEN=0

Note: These bits can only be written for test purposes when the DPLL is disabled.

18.12.8 Register DPLL_OSW

Address Offset:	**see Appendix B**																**Initial Value:**									**0x0000_0200**						
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																						OSS		Reserved						SWON_T	SWON_S
Mode	R																						RPw		R						R	R
Initial Value	0x0000 00																						10		0x00						0	0

Offset and Switch old/new Address Register

Bit 0 **SWON_S: Switch of new STATE;** Switch bit for LSB address of *STATE*.

This bit is changed for each write access to TS_S/TS_S_OLD. Using this unchanged address bit SWON_S for any access to TS_S results always in an access to TS_S_OLD. For writing to this address the former old (TS_S_OLD_old) value is overwritten by the new one while the SWON_S bit changes. Thus the former new one is now the old one and the next access is after changing SWON_S directed to this place. Therefore write to TS_S first and after that immediately to FTV_S and PSSM, always before a new TS_S value is to be written.

Note: After writing TS_S, FTV_S and PSSM in this order the address pointer AP with LSB(AP)=SWON_S shows for the corresponding address to TS_S_OLD, FTV_S and PSSM while LSB(AP)=/SWON_S results in an access to TS_S, FTV_S_old and PSSM_OLD respectively. The value can be read only. This bit is reset when disabling the DPLL (DEN=0).

Bit 1 **SWON_T: Switch of new TRIGGER;** Switch bit for LSB address of *TRIGGER*.

This bit is changed for each write access to TS_T/TS_T_OLD. Using this unchanged address bit SWON_T for any access to TS_T results always in an access to TS_T_OLD. For writing to this address the former old (TS_T_OLD_old) value is overwritten by the new one while the SWON_T bit changes. Thus the former new one is now the old one and the next access is after changing SWON_T directed to this place. Therefore write to TS_T first and after that immediately to FTV_T and PSTM, always before a new TS_T value is to be written.

Note: After writing TS_T, FTV_T and PSTM in this order the address pointer AP with LSB(AP)=SWON_T shows for the corresponding address to TS_T_OLD, FTV_T and PSTM while LSB(AP)=/SWON_T results in an access to TS_T, FTV_T_old and PSTM_OLD respectively. The value can be read only. This bit is reset when disabling the DPLL (DEN=0).

Bit 7:2 **Reserved**

Note: Read as zero, should be written as zero.

Bit 9:8 **OSS: Offset size** of RAM region 2

0x0 = Offset size 128 of RAM region 2.

0x1 = Offset size 256 of RAM region 2.

0x2 = Offset size 512 of RAM region 2.

0x3 = Offset size 1024 of RAM region 2.

Note: At least 128 and at most 1024 values can be stored in each of the RAM 2 regions a to d accordingly. The value can be set only for DEN=0. The change of the OSS value results in an automatic change of the offset values in the DPLL_AOSV_2 register

Note: This value can only be written when the DPLL is disabled.

Bit 31:10 **Reserved**

Note: Read as zero, should be written as zero.

18.12.9 Register DPLL_AOSV_2

Address Offset:	**see Appendix B**																**Initial Value:**								**0x1810_0800**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	AOSV_2D								AOSV_2C								AOSV_2B								AOSV_2A							
Mode	R								R								R								R							
Initial Value	0x18								0x10								0x08								0x00							

Address Offset Register of RAM 2 Regions

Bit 7:0 **AOSV_2A: Address offset value** of the RAM **2A** region.

The value in this field is to be multiplied by 256 (shift left 8 Bits) and added with the start address of the RAM in order to get the start address

of RAM region 2a. When the APT value is added to this start address, the current RAM cell RDT_Tx is addressed.
value is set automatically when OSS in the PPLL_OSW register ist set:

OSS=0x0: AOSV_2A= 0x00
OSS=0x1: AOSV_2A= 0x00
OSS=0x2: AOSV_2A= 0x00
OSS=0x3: AOSV_2A= 0x00

Bit 15:8 **AOSV_2B: Address offset value** of the RAM **2B** region.
The value in this field is to be multiplied by 256 (shift left 8 Bits) and added with the start address of the RAM in order to get the start address of RAM region 2b. When the APT value is added to this start address, the current RAM cell TSF_Tx is addressed.
value is set automatically when OSS in the PPLL_OSW register ist set:

OSS=0x0: AOSV_2B= 0x02
OSS=0x1: AOSV_2B= 0x04
OSS=0x2: AOSV_2B= 0x08
OSS=0x3: AOSV_2B= 0x10

Bit 23:16 **AOSV_2C: Address offset valu**e of the RAM **2C** region.
The value in this field is to be multiplied by 256 (shift left 8 Bits) and added with the start address of the RAM in order to get the start address of RAM region 2c. When the APT value is added to this start address, the current RAM cell ADT_Tx is addressed.
value is set automatically when OSS in the PPLL_OSW register ist set:

OSS=0x0: AOSV_2C= 0x04
OSS=0x1: AOSV_2C= 0x08
OSS=0x2: AOSV_2C= 0x10
OSS=0x3: AOSV_2C= 0x20

Bit 31:24 **AOSV_2D: Address offset value** of the RAM **2D** region.
The value in this field is to be multiplied by 256 (shift left 8 Bits) and added with the start address of the RAM in order to get the start address of RAM region 2d. When the APT value is added to this start address, the current RAM cell DT_Tx is addressed.
value is set automatically when OSS in the PPLL_OSW register ist set:

OSS=0x0: AOSV_2D= 0x06
OSS=0x1: AOSV_2D= 0x0C
OSS=0x2: AOSV_2D= 0x18
OSS=0x3: AOSV_2D= 0x30

Note: The offset values are needed to support a scalable RAM size of region 2 from 1,5 Kbytes to 12 Kbytes. The values above must be in correlation with the offset size defined in the OSW register. All offset values are set automatically in accordance to the OSS value in the DPLL_OSW register. This value can be set only for DEN=0.

18.12.10 Register DPLL_APT

Address Offset:	see Appendix B																Initial Value:							0x0000_0000								
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								APT_2B										WAPT_2B	Reserved	APT										WAPT	Reserved
Mode	R								RPw										RAw	R	RPw										RAw	R
Initial Value	0x00								0x000										0	0	0x000										0	0

Actual RAM Pointer Address for TRIGGER

Bit 0 **Reserved**

Note: Read as zero, should be written as zero.

Bit 1 **WAPT:** Write bit for address pointer APT, read as zero.

0 = the APT is not writeable

1 = the APT is writeable

Bit 11:2 **APT: Address pointer TRIGGER;** Actual RAM pointer address value offset for DT_T[i] and RDT_T[i] in FULL_SCALE for 2*(TNU+1-SYN_NT) TRIGGER events.

this pointer is used for the RAM region 2 subsections 2a and 2d. The pointer APT is incremented for each active *TRIGGER* event (simultaneously with APT_2B, APT_2C) for DIR1=0. For DIR1=1 the APT is decremented.

The APT offset value is added in the above shown bit position with the subsection address offset of the corresponding RAM region

Note: The APT pointer value is directed to the RAM position, in which the data values are to be written, which corresponds to the last increment. The APT value is not to be changed, when the direction (shown by DIR1) changes, because it points always to a storage place after the considered increment. Changing of DIR1 takes place always after an active *TRIGGER* event and the resulting increment/decrement.

Note: This value can only be written when the WAPT bit is set.

Bit 12 **Reserved**

Note: Read as zero, should be written as zero.

Bit 13 **WAPT_2B:** Write bit for address pointer APT_2B, read as zero.

0 = the APT_2B is not writeable

1 = the APT_2B is writeable

Bit 23:14 **APT_2B: Address pointer TRIGGER for RAM region 2b;** Actual RAM pointer address value for TSF_T[i]

Actual RAM pointer address of *TRIGGER* events in FULL_SCALE for 2*(TNU+1) *TRIGGER* periods; this pointer is used for the RAM region 2b. The RAM pointer is initially set to zero.

For SYT=1: The pointer APT_2B is incremented by SYN_T_OLD for each active *TRIGGER* event (simultaneously with APT and APT_2C) for DIR1=0 when an active *TRIGGER* input appears. For DIR1=1 (backwards) the APT is decremented by SYN_T_OLD.

For SYT=0: APT_2B is incremented or decremented by 1.

In addition when the APT_2C value is written by the CPU - in order to synchronize the DPLL- with the next active *TRIGGER* event the APT_2B_EXT value is added/subtracted (while APT_2B_STATUS is one; see DPLL_APT_SYNC register at chapter 18.12.24).

Note: This value can only be written when the WAPT_2B bit is set.

Bit 31:24 **Reserved**

Note: Read as zero, should be written as zero.

18.12.11 Register DPLL_APS

Address Offset:	see Appendix B																Initial Value: 0x0000_0000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved												APS_1C2						WAPS_1C2	Reserved					APS						WAPS	Reserved
Mode	R												RPw						RAw	R					RPw						RAw	R
Initial Value	0x000												0x00						0	0					0x00						0	0

Actual RAM Pointer Address for STATE

Bit 0 **Reserved**

Note: Read as zero, should be written as zero.

Bit 1 **WAPS:** Write bit for address pointer APS, read as zero.

0 = the APS is not writeable

1 = the APS is writeable

Bit 7:2 **APS: Address pointer STATE;** Actual RAM pointer address value for DT_S[i] and RDT_S[i]

Actual RAM pointer and synchronization position/value of *STATE* events in FULL_SCALE for up to 64 *STATE* events but limited to 2*(SNU+1-SYN_NS) in normal and emergency mode for SYSF=0 or to 2*(SNU+1)-SYN_NS for SYSF=1 respectively; this pointer is used for the RAM region 1c1 and 1c4.

APS is incremented (decremented) by one for each active *STATE* event and DIR2=0 DIR2=1). The APS offset value is added in the above shown bit position with the subsection offset of the RAM region.

Note: The APS pointer value is directed to the RAM position, in which the data values are to be written, which correspond to the last increment. The APS value is not to be changed, when the direction (shown by DIR2) changes, because it points always to a storage place after the considered increment. Changing of DIR2 takes place always after an active *STATE* event and the resulting increment/decrement.

Note: This value can only be written when the WAPS bit is set.

Bit 12:8 **Reserved**

Note: Read as zero, should be written as zero.

Bit 13 **WAPS_1C2:** Write bit for address pointer APS_1C2, read as zero.

0 = the APS_1C2 is not writeable

1 = the APS_1C2 is writeable

Bit 19:14 **APS_1C2: Address pointer STATE for RAM region 1c2;** Actual RAM pointer address value for TSF_S[i].

Initial value: zero (0x00). Actual RAM pointer and synchronization position/value of *STATE* events in FULL_SCALE for up to 64 *STATE* events but limited to 2*(SNU+1) in normal and emergency mode; this pointer is used for the RAM region 1c2.

For SYS=1: APS_1C2 is incremented (decremented) by SYN_S_OLD for each active *STATE* event and DIR2=0 (DIR2=1).

For SYS=0: APT_1c2 is incremented or decremented by 1 respectively.

The APS_1C2 offset value is added in the above shown bit position with the subsection offset of the RAM region.

In addition when the APS_1C3 value is written by the CPU - in order to synchronize the DPLL- with the next active *STATE* event the APS_1C2_EXT value is added/subtracted (while APS_1C2_STATUS is one; see DPLL_APT_SYNC register at chapter 18.12.25).

Note: This value can only be written when the WAPS_1C2 bit is set

Bit 31:20 **Reserved**

Note: Read as zero, should be written as zero.

Note: This register is only used when DPLL_CTRL_11.STATE_EXT is not set. If DPLL_CTRL_11.STATE_EXT is set any read/write access to this register will return AEI_STATUS = 0b10.

18.12.12 Register DPLL_APT_2C

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																				APT_2C										Reserved	
Mode	R																				RW										R	
Initial Value	0x00000																				0x000										00	

Actual RAM Pointer Address for Region 2c

Bit 1:0 **Reserved**

Note: Read as zero, should be written as zero.

Bit 11:2 **APT_2C: Address pointer TRIGGER for RAM region 2c;** Actual RAM pointer address value for ADT_T[i].

Actual RAM pointer address value of *TRIGGER* adapt events in FULL_SCALE for 2*(TNU+1-SYN_NT) *TRIGGER* periods depending on the size of the used RAM 2; this pointer is used for the RAM region 2 for the subsection 2c only. The RAM pointer is initially set to zero. The APT_2C value is set by the CPU when the synchronization condition was detected. Within the RAM region 2c initially the conditions for synchronization gaps and adapted values are stored by the CPU.

Bit 31:12 **Reserved**

Note: Read as zero, should be written as zero.

Note: The APT_2C pointer values are directed to the RAM position of the profile element in RAM region 2c, which correspond to the current increment. For DIR1=0 (DIR1=1) the pointers APT_2C_x are incremented (decremented) by one simultaneously with APT. For SMC=0 the change of DIR1 takes place always after an active *TRIGGER* event (by evaluation of the inactive slope) and the resulting increment/decrement. In the case SMC=1 the direction change is known before the input event is processed.

The correction of the APT_2C pointer differs: for SMC=0 correct 4 times and for SMC=1 correct only 2 times.

The APT_2C_x offset value is added in the above shown bit position with the subsection address offset of the corresponding RAM region.

18.12.13 Register DPLL_APS_1C3

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																								APS_1C3						Reserved	
Mode	R																								RW						R	
Initial Value	0x000000																								0x00						00	

Actual RAM Pointer Address for RAM region 1c3

Bit 1:0 **Reserved**

Note: Read as zero, should be written as zero.

Bit 7:2 **APS_1C3: Address pointer STATE for RAM region 1c3;** Actual RAM pointer address value for ADT_S[i]

Initial value: zero (0x00). Actual RAM pointer and synchronization position/value of *STATE* events in FULL_SCALE for up to 64 *STATE* events but limited to 2*(SNU+1-SYN_NS) in normal and emergency mode for SYSF=0 or to 2*(SNU+1)-SYN_NS for SYSF=1 respectively; this pointer is used for the RAM region 1c3. The RAM pointer is set by the CPU accordingly, when the synchronization condition was detected.

Bit 31:8 **Reserved**

Note: Read as zero, should be written as zero.

Note: The APS_1C3 pointer value is directed to the RAM position of the profile element in RAM region 1c2, which corresponds to the current increment. When changing the direction DIR1 or DIR2 respectively, this is always known before an active *STATE* event is processed. This is because of the pattern recognition in SPE (for PMSM) or because of the direction change recognition by TRIGGER. This direction change results in an automatic increment (forwards) or decrement (backwards) when the input event occurs in addition with a 2 times correction.

The APS_1C3_x offset value is added in the above shown bit position with the subsection address offset of the corresponding RAM region.

Note: This register is only used when DPLL_CTRL_11.STATE_EXT is not set. If DPLL_CTRL_11.STATE_EXT is set any read/write access to this register will return AEI_STATUS = 0b10.

18.12.14 Register DPLL_NUTC

Address Offset:	see Appendix B																Initial Value:								0x0001_2001							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	WVTN	WSYN	WNUT	Reserved				VTN						SYN_T_OLD			SYN_T			Reserved		FST	NUTE									
Mode	RAw	RAw	RAw	R				RPw						RPw			RPw			R		RPw	RPw									
Initial Value	0	0	0	0x0				0x00						001			001			00		0	0x001									

Number of Recent TRIGGER Events used for Calculations

Bit 9:0 **NUTE:** Number of recent *TRIGGER* events used for SUB_INC1 and action calculations modulo 2*(TNU$_{max}$+1).

NUTE: number of last nominal increments to be considered for the calculations.

No gap is considered in that case for this value, but in the VTN value (see below):

This value is set by the CPU, but reset automatically to “1” by a change of direction or loss of LOCK. Each other value can be set by the CPU, maybe Full_SCALE, HALF_SCALE or parts of them. For FULL_SCALE set NUTE= 2*(TNU +1) and for HALF_SCALE NUTE= TNU +1. The relation values QDT_Tx are calculated using NUTE values in the past with its maximum value of 2*(TNU +1). The value zero (in combination with the value FST=1) does mean 2^{11} values in the past.

Note: To prevent that inconsistencies between internal pointer in which NUTE is used and the case decision of different prediction method's for prediction of the next event and PMT(position minus time) occur, the NUTE value is stored internally at that point of time when the internal pointers are calculated for the next event cycle

0 = the NUTE value is less then FULL_SCALE

1 = the NUTE value is equal to FULL_SCALE

This value is set by the CPU, but reset automatically to “0” by a change of direction or loss of LOCK.

Note: This value can only be written when the WNUT bit is set.

Bit 10 **FST: FULL_SCALE of TRIGGER;** this value is to be set, when NUTE is set to FULL_SCALE

Note: This value can only be written when the WNUT bit is set.

Bit 12:11 **Reserved**

Note: Read as zero, should be written as zero.

Bit 15:13 **SYN_T:** number of real and virtual events to be considered for the current increment.

This value reflects the NT value of the last valid increment, stored in ADT_T[i]; to be updated after all calculations in step 17 of Table 18.8.6.7.1.

Note: This value can only be written when the WSYN bit in this register is set.

Bit 18:16 **SYN_T_OLD:** number of real and virtual events to be considered for the last increment.

This value reflects the NT value of the last but one valid increment, stored in ADT_T[i]; is updated automatically when writing SYN_T.

Note: This value is updated by the SYN_T value when the WSYN bit in this register is set.

Bit 24:19 **VTN: Virtual TRIGGER number;** number of virtual increments in the current NUTE region

This value reflects the number of virtual increments in the current NUTE region; for NUTE=1 this value is zero, when the CPU sets NUTE to a value > 1 , it must also set VTN to the correspondent value; for NUTE is set to FULL_SCALE including NUTE=zero (2^{11} modulo 2^{11}) the VTN is to be set to 2* SYN_NT.

The VTN value is subtracted from the NUTE value in order to get the corresponding APT value for the past; the VTN value is not used for the APT_2B pointer.

VTN is to be updated by the CPU when a new gap is to be considered for NUTE or a gap is leaving the NUTE region; for this purpose the TINT values in the profile can be used to generate an interrupt for the CPU at the corresponding positions; no further update of VTN is necessary when NUTE is set to FULL_SCALE

Note: This value can only be written when the WVTN bit is set.

Bit 28:25 **Reserved**

Note: Read as zero, should be written as zero.

Bit 29 **WNUT:** write control bit for NUTE and FST; read as zero.

0 = the NUTE value is not writeable

1 = the NUTE value is writeable

Bit 30 **WSYN:** write control bit for SYN_T and SYN_T_OLD; read as zero.

0 = the SYN_T value is not writeable

1 = the SYN_T value is writeable

Bit 31 **WVTN:** write control bit for VTN; read as zero.

0 = the VTN value is not writeable

1 = the VTN value is writeable

Note: DPLL Number of recent TRIGGER events used for calculations (mod 2*(TNU +1-SYN_NT)).

18.12.15 Register DPLL_NUSC

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_2081**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	WVSN	WSYN	WNUS	Reserved				VSN						SYN_S_OLD						SYN_S						FSS	NUSE					
Mode	RAw	RAw	RAw	R				RPw						RPw						RPw						RPw	RPw					
Initial Value	0	0	0	0x0				0x00						0x01						0x01						0	0x01					

Number of Recent STATE Events used for Calculations

Bit 5:0 **NUSE:** Number of recent *STATE* events used for SUB_INCx calculations modulo 2*(SNU$_{max}$+1).

No gap is considered in that case for this value, but in the VSN value (see below):

This register is set by the CPU but reset automatically to “1” by a change of direction or loss of LOCK. Each other value can be set by the CPU, maybe Full_SCALE, HALF_SCALE or parts of them. The relation values QDT_Sx are calculated using NUSE values in the past with its maximum value of 2*SNU+1.

Note: To prevent that inconsistencies between internal pointer in which NUSE is used and the case decision of different prediction method's for prediction of the next event and PMT(position minus time) occur, the NUSE value is stored internally at that point of time when the internal pointers are calculated for the next event cycle.

Note: This value can only be written when the WNUS bit is set.

Bit 6 **FSS: FULL_SCALE of STATE;** this value is to be set, when NUSE is set to FULL_SCALE

0 = the NUSE value is less then FULL_SCALE

1 = the NUSE value is equal to FULL_SCALE

This value is set by the CPU, but reset automatically to “0” by a change of direction or loss of LOCK.

Note: This value can only be written when the WNUS bit is set.

Bit 12:7 **SYN_S:** number of real and virtual events to be considered for the current increment.

This value reflects the NS value of the last valid increment, stored in ADT_S[i]; to be updated after all calculations in step 37 of Table 18.8.6.7.1.

Note: This value can only be written when the WSYN bit in this register is set.

Bit 18:13 **SYN_S_OLD:** number of real and virtual events to be considered for the last increment.

This value reflects the NS value of the last but one valid increment, stored in ADT_S[i]; is updated automatically when writing SYN_S

Note: This value is updated by the SYN_S value when the WSYN bit in this register is set.

Bit 24:19 **VSN: virtual STATE number;** number of virtual state increments in the current NUSE region.

This value reflects the number of virtual increments in the current NUSE region; for NUSE=1 this value is zero, when the CPU sets NUSE to a value > 1 or zero(2^7 modulo 2^7) , it must also set VSN to the correspondent value;

the VSN value is subtracted from the NUSE value in order to get the corresponding APS value for the past; the VSN value is not used for the APS_1C2 pointer.

VSN is to be updated by the CPU when a new gap is to be considered for NUSE or a gap is leaving the NUSE region; for this purpose the SASI interrupt can be used; no further update of VSN is necessary when NUSE is set to FULL_SCALE

Note: This value can only be written when the WVSN bit is set.

Bit 28:25 **Reserved**

Note: Read as zero, should be written as zero.

Bit 29 **WNUS:** write control bit for NUSE; read as zero**.**

0 = the NUSE value is not writeable

1 = the NUSE value is writeable

Bit 30 **WSYN:** write control bit for SYN_S and SYN_S_OLD; read as zero.

0 = the SYN_S value is not writeable

1 = the SYN_S value is writeable

Bit 31 **WVSN:** write control bit for VSN; read as zero.

0 = the VSN value is not writeable

1 = the VSN value is writeable

Note: This register is only used when DPLL_CTRL_11.STATE_EXT is not set. If DPLL_CTRL_11.STATE_EXT is set any read/write access to this register will return AEI_STATUS = 0b10.

18.12.16 Register DPLL_NTI_CNT

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																						NTI_CNT									
Mode	R																						RW									
Initial Value	0x0000 00																						0x000									

Number of Active TRIGGER Events to Interrupt

Bit 9:0 **NTI_CNT: Number of TRIGGERs to interrupt;** Number of active TRIGGER events to the next DPLL_CDTI interrupt.
This value shows the remaining *TRIGGER* events until an active TRIGGER slope results in a DPLL_CDTI interrupt;
the value is to be count down for each active *TRIGGER* event.

Bit 31:10 **Reserved**
Note: Read as zero, should be written as zero.

18.12.17 Register DPLL_IRQ_NOTIFY

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved				DCGI	SORI	TORI	CDSI	CDTI	TE4I	TE3I	TE2I	TE1I	TE0I	LL2I	GL2I	EI	LL1I	GL1I	W1I	W2I	PWI	TASI	SASI	MTI	MSI	TISI	SISI	TAXI	TINI	PEI	PDI
Mode	R				RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw
Initial Value	0x0				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Interrupt Register

Bit 0 **PDI:** DPLL disable interrupt; announces the switch off of the DEN bit.
0 = The DPLL disable interrupt is not requested
1 = The DPLL disable interrupt is requested

Note: This event is combined with the PEI interrupt to the common PDI + PEI interrupt line number 1.

Bit 1 **PEI:** DPLL enable interrupt; announces the switch on of the DEN bit.
0 = The DPLL enable interrupt is not requested
1 = The DPLL enable interrupt is requested

Note: This event is combined with the PDI interrupt to the common PDI + PEI interrupt line number 1.

Bit 2 **TINI:** *TRIGGER* minimum hold time violation interrupt (dt <= THMI > 0).
0 = No violation of minimum hold time of *TRIGGER* is detected
1 = A violation of minimum hold time of *TRIGGER* is detected

Bit 3 **TAXI:** *TRIGGER* maximum hold time violation interrupt (dt > THMA > 0).
0 = No violation of maximum hold time of *TRIGGER* is detected
1 = A violation of maximum hold time of *TRIGGER* is detected

Bit 4 **SISI:** *STATE* inactive slope interrupt.
0 = No inactive slope of *STATE* is detected
1 = An inactive slope of *STATE* is detected

Bit 5 **TISI:** *TRIGGER* inactive slope interrupt.
0 = No inactive slope of *TRIGGER* is detected
1 = An inactive slope of *TRIGGER* is detected

Note: The TISI bit is only set for an inactive slope when the preceding active slope was accepted. In the case of suppression of the last active slope by the plausibility check the next inactive slope is to be ignored. No set of TISI is performed in this case.

Bit 6 **MSI:** Missing *STATE* interrupt.
0 = The missing *STATE* interrupt is not requested
1 = The missing *STATE* interrupt is requested

Bit 7 **MTI:** Missing *TRIGGER* interrupt.
0 = The missing *TRIGGER* interrupt is not requested
1 = The missing *TRIGGER* interrupt is requested

Bit 8 **SASI:** *STATE* active slope interrupt.
0 = No active slope of *STATE* is detected
1 = An active slope of *STATE* is detected

Bit 9 **TASI:** *TRIGGER* active slope interrupt.

0 = No active slope of *TRIGGER* is detected
1 = An active slope of *TRIGGER* is detected

Bit 10 **PWI:** Plausibility window (PVT) violation interrupt of *TRIGGER*.
0 = The plausibility window is not violated
1 = The plausibility window is violated

Bit 11 **W2I:** RAM write access to RAM region 2 interrupt.
0 = The RAM write access interrupt is not requested
1 = The RAM write access interrupt is requested

Bit 12 **W1I:** Write access to RAM region 1b or 1c interrupt.
0 = The RAM write access interrupt is not requested
1 = The RAM write access interrupt is requested

Bit 13 **GL1I:** Get of lock interrupt, for SUB_INC1.
0 = The lock getting interrupt is not requested
1 = The lock getting interrupt is requested

Bit 14 **LL1I:** Loss of lock interrupt for SUB_INC1.
0 = The lock loss interrupt is not requested
1 = The lock loss interrupt is requested

Bit 15 **EI:** Error interrupt (see status register bit 31).
0 = The error interrupt is not requested
1 = The error interrupt is requested

Bit 16 **GL2I:** Get of lock interrupt, for SUB_INC2.
0 = The lock getting interrupt is not requested
1 = The lock getting interrupt is requested

Bit 17 **LL2I:** Loss of lock interrupt for SUB_INC2.
0 = The lock loss interrupt is not requested
1 = The lock loss interrupt is requested

Bit 18 **TE0I:** TRIGGER event interrupt 0.
0 = No Interrupt on *TRIGGER* event 0 requested
1 = Interrupt on *TRIGGER* event 0 requested

Bit 19 **TE1I:** TRIGGER event interrupt 1.
0 = No Interrupt on *TRIGGER* event 1 requested
1 = Interrupt on *TRIGGER* event 1 requested

Bit 20 **TE2I:** TRIGGER event interrupt 2.
0 = No Interrupt on *TRIGGER* event 2 requested
1 = Interrupt on *TRIGGER* event 2 requested

Bit 21 **TE3I:** TRIGGER event interrupt 3.
0 = No Interrupt on *TRIGGER* event 3 requested
1 = Interrupt on *TRIGGER* event 3 requested

Bit 22 **TE4I:** TRIGGER event interrupt 4.
0 = No Interrupt on *TRIGGER* event 4 requested
1 = Interrupt on *TRIGGER* event 4 requested

Bit 23 **CDTI:** Calculation of TRIGGER duration done, only while NTI_CNT is zero.
0 = No Interrupt on calculated *TRIGGER* duration requested or NTI_CNT is not zero
1 = Interrupt on calculated *TRIGGER* duration requested while NTI_CNT is zero

Bit 24 **CDSI:** Calculation of STATE duration done
0 = No Interrupt on calculated *STATE* duration requested
1 = Interrupt on calculated *STATE* duration requested

Bit 25 **TORI:** TRIGGER out of range interrupt
0 = *TRIGGER* is not out of range
1 = *TRIGGER* is out of range, the TOR bit in the DPLL_STATUS register is set to 1

Bit 26 **SORI:** STATE out of range
0 = *STATE* is not out of range
1 = *STATE* is out of range, the SOR bit in the DPLL_STATUS register is set to 1

0 = No direction change of *TRIGGER* is detected
1 = Direction change of *TRIGGER* is detected

Note: The interrupt occurs at line number 0.
Bit 27 **DCGI:** Direction change interrupt
Bit 31:28 **Reserved**
Note: Read as zero, should be written as zero.

Note: All bits in the DPLL_IRQ_NOTIFY register are set permanently until writing a one bit value is performed to the corresponding bit.

18.12.18 Register DPLL_IRQ_EN

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved				DCGI_IRQ_EN	SORI_IRQ_EN	TORI_IRQ_EN	CDSI_IRQ_EN	CDTI_IRQ_EN	TE4I_IRQ_EN	TE3I_IRQ_EN	TE2I_IRQ_EN	TE1I_IRQ_EN	TE0I_IRQ_EN	LL2I_IRQ_EN	GL2I_IRQ_EN	EI_IRQ_EN	LL1I_IRQ_EN	GL1I_IRQ_EN	W1I_IRQ_EN	W2I_IRQ_EN	PWI_IRQ_EN	TASI_IRQ_EN	SASI_IRQ_EN	MTI_IRQ_EN	MSI_IRQ_EN	TISI_IRQ_EN	SISI_IRQ_EN	TAXI_IRQ_EN	TINI_IRQ_EN	PEI_IRQ_EN	PDI_IRQ_EN
Mode	R				RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Initial Value	0x0				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Interrupt Enable Register

Bit 0 **PDI_IRQ_EN:** DPLL disable interrupt enable, when switch off of the DEN bit.
0 = The DPLL disable interrupt is not enabled
1 = The DPLL disable interrupt is enabled

Bit 1 **PEI_IRQ_EN:** DPLL enable interrupt enable, when switch on of the DEN bit.
0 = The DPLL enable interrupt is not enabled
1 = The DPLL enable interrupt is enabled

Bit 2 **TINI_IRQ_EN:** *TRIGGER* minimum hold time violation interrupt enable bit.
0 = Minimum hold time violation of *TRIGGER* interrupt is not enabled
1 = The minimum hold time violation of *TRIGGER* interrupt is enabled

Bit 3 **TAXI_IRQ_EN:** *TRIGGER* maximum hold time violation interrupt enable bit.
0 = Maximum hold time violation of *TRIGGER* interrupt is not enabled
1 = The maximum hold time violation of *TRIGGER* interrupt is enabled

Bit 4 **SISI_IRQ_EN:** *STATE* inactive slope interrupt enable bit.
0 = The interrupt at the inactive slope of *STATE* is not enabled
1 = The interrupt at the inactive slope of *STATE* is enabled

Bit 5 **TISI_IRQ_EN:** *TRIGGER* inactive slope interrupt enable bit.
0 = The interrupt at the inactive slope of *TRIGGER* is not enabled
1 = The interrupt at the inactive slope of *TRIGGER* is enabled

Bit 6 **MSI_IRQ_EN:** Missing *STATE* interrupt enable.

0 = The missing *STATE* interrupt is not enabled
1 = The missing *STATE* interrupt is enabled

Bit 7 **MTI_IRQ_EN:** Missing *TRIGGER* interrupt enable.
0 = The missing *TRIGGER* interrupt is not enabled
1 = The missing *TRIGGER* interrupt is enabled

Bit 8 **SASI_IRQ_EN:** *STATE* active slope interrupt enable.
0 = The active slope *STATE* interrupt is not enabled.
1 = The active slope *STATE* interrupt is enabled

Bit 9 **TASI_IRQ_EN:** *TRIGGER* active slope interrupt enable.
0 = The active slope *TRIGGER* interrupt is not enabled
1 = The active slope *TRIGGER* interrupt is enabled

Bit 10 **PWI_IRQ_EN:** Plausibility window (PVT) violation interrupt of *TRIGGER* enable.
0 = The plausibility violation interrupt is not enabled
1 = The plausibility violation interrupt is enabled

Bit 11 **W2I_IRQ_EN:** RAM write access to RAM region 2 interrupt enable.
0 = The RAM write access interrupt is not enabled
1 = The RAM write access interrupt is enabled

Bit 12 **W1I_IRQ_EN:** Write access to RAM region 1b or 1c interrupt.
0 = The RAM write access interrupt is not enabled
1 = The RAM write access interrupt is enabled.

Bit 13 **GL1I_IRQ_EN:** Get of lock interrupt enable, when lock arises.
0 = The lock getting interrupt is not enabled
1 = The lock getting interrupt is enabled

Bit 14 **LL1I_IRQ_EN:** Loss of lock interrupt enable.
0 = The lock loss interrupt is not enabled
1 = The lock loss interrupt is enabled

Bit 15 **EI_IRQ_EN:** Error interrupt enable (see status register).
0 = The error interrupt is not enabled
1 = The error interrupt is enabled

Bit 16 **GL2I_IRQ_EN:** Get of lock interrupt enable for SUB_INC2.
0 = The lock getting interrupt is not requested
1 = The lock getting interrupt is requested

Bit 17 **LL2I_IRQ_EN:** Loss of lock interrupt enable for SUB_INC2.
0 = The lock loss interrupt is not requested
1 = The lock loss interrupt is requested

Bit 18 **TE0I_IRQ_EN:** TRIGGER event interrupt 0 enable.
0 = No Interrupt on *TRIGGER* event 0 enabled
1 = Interrupt on *TRIGGER* event 0 enabled

Bit 19 **TE1I_IRQ_EN:** TRIGGER event interrupt 1 enable.
0 = No Interrupt on *TRIGGER* event 1 enabled
1 = Interrupt on *TRIGGER* event 1 enabled

Bit 20 **TE2I_IRQ_EN:** TRIGGER event interrupt 2 enable.
0 = No Interrupt on *TRIGGER* event 2 enabled
1 = Interrupt on *TRIGGER* event 2 enabled

Bit 21 **TE3I_IRQ_EN:** TRIGGER event interrupt 3 enable.
0 = No Interrupt on *TRIGGER* event 3 enabled
1 = Interrupt on *TRIGGER* event 3 enabled

Bit 22 **TE4I_IRQ_EN:** TRIGGER event interrupt 4 enable.
0 = No Interrupt on *TRIGGER* event 4 enabled
1 = Interrupt on *TRIGGER* event 4 enabled

Bit 23 **CDTI_IRQ_EN:** Enable interrupt when calculation of TRIGGER duration done
0 = No Interrupt on calculated *TRIGGER* duration enabled
1 = Interrupt on calculated *TRIGGER* duration enabled

Bit 24 **CDSI_IRQ_EN:** Enable interrupt when calculation of TRIGGER duration done
0 = No Interrupt on calculated *STATE* duration enabled
1 = Interrupt on calculated *STATE* duration enabled

Bit 25 **TORI_IRQ_EN:** TRIGGER out of range interrupt
0 = No Interrupt when *TRIGGER* is out of range enabled
1 = Interrupt when *TRIGGER* is out of range enabled

Bit 26 **SORI_IRQ_EN:** STATE out of range
0 = No Interrupt when *STATE* is out of range enabled
1 = Interrupt when *STATE* is out of range enabled

Bit 27 **DCGI_IRQ_EN:** Direction change interrupt
0 = No Interrupt when a direction change of *TRIGGER* is detected
1 = Interrupt when a direction change of *TRIGGER* is detected

Bit 31:28 **Reserved**
Note: Read as zero, should be written as zero.

18.12.19 Register DPLL_IRQ_FORCINT

Address Offset:	see Appendix B																Initial Value: 0x0000_0000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved				TRG_DCGI	TRG_SORI	TRG_TORI	TRG_CDSI	TRG_CDTI	TRG_TE4I	TRG_TE3I	TRG_TE2I	TRG_TE1I	TRG_TE0I	TRG_LL2I	TRG_GL2I	TRG_EI	TRG_LL1I	TRG_GL1I	TRG_W1I	TRG_W2I	TRG_PWI	TRG_TASI	TRG_SASI	TRG_MTI	TRG_MSI	TRG_TISI	TRG_SISI	TRG_TAXI	TRG_TINI	TRG_PEI	TRG_PDI
Mode	R				RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw
Initial Value	0x0				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Force Interrupt Register

Bit 0 **TRG_PDI:** Force Interrupt PDI
0 = the corresponding interrupt is not forced
1 = the corresponding interrupt is forced for one clock

Note: This bit is cleared automatically after write.
Note: This bit is write protected by bit RF_PROT of register GTM_CTRL

Bit 1 **TRG_PEI:** Force Interrupt PEI
see bit 0

Bit 2 **TRG_TINI:** Force Interrupt TINI
see bit 0

Bit 3 **TRG_TAXI:** Force Interrupt TAXI
see bit 0

Bit 4 **TRG_SISI:** Force Interrupt SISI
see bit 0

Bit 5 **TRG_TISI:** Force Interrupt TISI
see bit 0

Bit 6 **TRG_MSI:** Force Interrupt MSI
see bit 0

Bit 7 **TRG_MTI:** Force Interrupt MTI
see bit 0

Bit 8 **TRG_SASI:** Force Interrupt SASI
see bit 0

Bit 9 **TRG_TASI:** Force Interrupt TASI
see bit 0

Bit 10 **TRG_PWI:** Force Interrupt PWI
see bit 0

Bit 11 **TRG_W2I:** Force Interrupt W2IF
see bit 0

Bit 12 **TRG_W1I:** Force Interrupt W1I
see bit 0

Bit 13 **TRG_GL1I:** Force Interrupt GL1I

see bit 0

Bit 14 **TRG_LL1I:** Force Interrupt LL1I
see bit 0

Bit 15 **TRG_EI:** Force Interrupt EI
see bit 0

Bit 16 **TRG_GL2I:** Force Interrupt GL2I
see bit 0

Bit 17 **TRG_LL2I:** Force Interrupt LL2I
see bit 0

Bit 18 **TRG_TE0I:** Force Interrupt TE0I
see bit 0

Bit 19 **TRG_TE1I:** Force Interrupt TE1I
see bit 0

Bit 20 **TRG_TE2I:** Force Interrupt TE2I
see bit 0

Bit 21 **TRG_TE3I:** Force Interrupt TE3I
see bit 0

Bit 22 **TRG_TE4I:** Force Interrupt TE4I
see bit 0

Bit 23 **TRG_CDTI:** Force Interrupt CDTI
see bit 0

Bit 24 **TRG_CDSI:** Force Interrupt CDSI
see bit 0

Bit 25 **TRG_TORI:** Force Interrupt TORI
see bit 0

Bit 26 **TRG_SORI:** Force Interrupt SORI
see bit 0

Bit 27 **TRG_DCGI:** Force interrupt DCGI
see bit 0

Bit 31:28 **Reserved**
Note: Read as zero, should be written as zero.

18.12.20 Register DPLL_IRQ_MODE

Address Offset:	see Appendix B																Initial Value:								0x0000_000X							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																														IRQ_MODE	
Mode	R																														RW	
Initial Value	0x0000 0000																														XX	

Interrupt Request Mode

Bit 1:0 **IRQ_MODE**: IRQ mode selection
0b00 = Level mode
0b01 = Pulse mode
0b10 = Pulse-Notify mode
0b11 = Single-Pulse mode
Note: The interrupt modes are described in chapter 2.5.

Bit 31:2 **Reserved**
Note: Read as zero, should be written as zero

18.12.21 Register DPLL_EIRQ_EN

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved				DCGI_EIRQ_EN	SORI_EIRQ_EN	TORI_EIRQ_EN	CDSI_EIRQ_EN	CDTI_EIRQ_EN	TE4I_EIRQ_EN	TE3I_EIRQ_EN	TE2I_EIRQ_EN	TE1I_EIRQ_EN	TE0I_EIRQ_EN	LL2I_EIRQ_EN	GL2I_EIRQ_EN	EI_EIRQ_EN	LL1I_EIRQ_EN	GL1I_EIRQ_EN	W1I_EIRQ_EN	W2I_EIRQ_EN	PWI_EIRQ_EN	TASI_EIRQ_EN	SASI_EIRQ_EN	MTI_EIRQ_EN	MSI_EIRQ_EN	TISI_EIRQ_EN	SISI_EIRQ_EN	TAXI_EIRQ_EN	TINI_EIRQ_EN	PEI_EIRQ_EN	PDI_EIRQ_EN
Mode	R				RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Initial Value	0x0				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Error Interrupt Enable Register

Bit 0 **PDI_EIRQ_EN:** DPLL disable interrupt enable, when switch off of the DEN bit.
0 = The DPLL disable interrupt is not enabled
1 = The DPLL disable interrupt is enabled

Bit 1 **PEI_EIRQ_EN:** DPLL enable interrupt enable, when switch on of the DEN bit.
0 = The DPLL enable interrupt is not enabled
1 = The DPLL enable interrupt is enabled

Bit 2 **TINI_EIRQ_EN:** *TRIGGER* minimum hold time violation interrupt enable bit.
0 = Minimum hold time violation of *TRIGGER* interrupt is not enabled
1 = The minimum hold time violation of *TRIGGER* interrupt is enabled

Bit 3 **TAXI_EIRQ_EN:** *TRIGGER* maximum hold time violation interrupt enable bit.
0 = Maximum hold time violation of *TRIGGER* interrupt is not enabled
1 = The maximum hold time violation of *TRIGGER* interrupt is enabled

Bit 4 **SISI_EIRQ_EN:** *STATE* inactive slope interrupt enable bit.
0 = The interrupt at the inactive slope of *STATE* is not enabled
1 = The interrupt at the inactive slope of *STATE* is enabled

Bit 5 **TISI_EIRQ_EN:** *TRIGGER* inactive slope interrupt enable bit.
0 = The interrupt at the inactive slope of *TRIGGER* is not enabled
1 = The interrupt at the inactive slope of *TRIGGER* is enabled

Bit 6 **MSI_EIRQ_EN:** Missing *STATE* interrupt enable.
0 = The missing *STATE* interrupt is not enabled
1 = The missing *STATE* interrupt is enabled

Bit 7 **MTI_EIRQ_EN:** Missing *TRIGGER* interrupt enable.
0 = The missing *TRIGGER* interrupt is not enabled
1 = The missing *TRIGGER* interrupt is enabled

Bit 8 **SASI_EIRQ_EN:** *STATE* active slope interrupt enable.
0 = The active slope *STATE* interrupt is not enabled.
1 = The active slope *STATE* interrupt is enabled

Bit 9 **TASI_EIRQ_EN:** *TRIGGER* active slope interrupt enable.
0 = The active slope *TRIGGER* interrupt is not enabled
1 = The active slope *TRIGGER* interrupt is enabled

Bit 10 **PWI_EIRQ_EN:** Plausibility window (PVT) violation interrupt of *TRIGGER* enable.
0 = The plausibility violation interrupt is not enabled
1 = The plausibility violation interrupt is enabled

Bit 11 **W2I_EIRQ_EN:** RAM write access to RAM region 2 interrupt enable.
0 = The RAM write access interrupt is not enabled
1 = The RAM write access interrupt is enabled

Bit 12 **W1I_EIRQ_EN:** Write access to RAM region 1b or 1c interrupt.
0 = The RAM write access interrupt is not enabled
1 = The RAM write access interrupt is enabled.

Bit 13 **GL1I_EIRQ_EN:** Get of lock interrupt enable, when lock arises.
0 = The lock getting interrupt is not enabled
1 = The lock getting interrupt is enabled

Bit 14 **LL1I_EIRQ_EN:** Loss of lock interrupt enable.
0 = The lock loss interrupt is not enabled
1 = The lock loss interrupt is enabled

Bit 15 **EI_EIRQ_EN:** Error interrupt enable (see status register).
0 = The error interrupt is not enabled
1 = The error interrupt is enabled

Bit 16 **GL2I_EIRQ_EN:** Get of lock interrupt enable for SUB_INC2.
0 = The lock getting interrupt is not requested
1 = The lock getting interrupt is requested

Bit 17 **LL2I_EIRQ_EN:** Loss of lock interrupt enable for SUB_INC2.
0 = The lock loss interrupt is not requested
1 = The lock loss interrupt is requested

Bit 18 **TE0I_EIRQ_EN:** TRIGGER event interrupt 0 enable.
0 = No Interrupt on *TRIGGER* event 0 enabled
1 = Interrupt on *TRIGGER* event 0 enabled

Bit 19 **TE1I_EIRQ_EN:** TRIGGER event interrupt 1 enable.
0 = No Interrupt on *TRIGGER* event 1 enabled
1 = Interrupt on *TRIGGER* event 1 enabled

Bit 20 **TE2I_EIRQ_EN:** TRIGGER event interrupt 2 enable.
0 = No Interrupt on *TRIGGER* event 2 enabled
1 = Interrupt on *TRIGGER* event 2 enabled

Bit 21 **TE3I_EIRQ_EN:** TRIGGER event interrupt 3 enable.
0 = No Interrupt on *TRIGGER* event 3 enabled
1 = Interrupt on *TRIGGER* event 3 enabled

Bit 22 **TE4I_EIRQ_EN:** TRIGGER event interrupt 4 enable.
0 = No Interrupt on *TRIGGER* event 4 enabled

1 = Interrupt on *TRIGGER* event 4 enabled

Bit 23 **CDTI_EIRQ_EN:** Enable interrupt when calculation of TRIGGER duration done
0 = No Interrupt on calculated *TRIGGER* duration enabled
1 = Interrupt on calculated *TRIGGER* duration enabled

Bit 24 **CDSI_EIRQ_EN:** Enable interrupt when calculation of TRIGGER duration done
0 = No Interrupt on calculated *STATE* duration enabled
1 = Interrupt on calculated *STATE* duration enabled

Bit 25 **TORI_EIRQ_EN:** TRIGGER out of range interrupt
0 = No Interrupt when *TRIGGER* is out of range enabled
1 = Interrupt when *TRIGGER* is out of range enabled

Bit 26 **SORI_EIRQ_EN:** STATE out of range
0 = No Interrupt when *STATE* is out of range enabled
1 = Interrupt when *STATE* is out of range enabled

Bit 27 **DCGI_EIRQ_EN:** Direction change interrupt
0 = No Interrupt when a direction change of *TRIGGER* is detected
1 = Interrupt when a direction change of *TRIGGER* is detected

Bit 31:28 **Reserved**
Note: Read as zero, should be written as zero.

18.12.22 Register DPLL_INC_CNT1

Address Offset:	**see Appendix B**																**Initial Value: 0x0000_0000**															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								INC_CNT1																							
Mode	R								RPw																							
Initial Value	0x00								0x000000																							

Counter Value of Sent SUB_INC1 Pulses

Bit 23:0 **INC_CNT1:** Actual number of pulses to be still sent out at the current increment until the next active input signal in automatic end mode;

Automatic addition of the number of demanded pulses MLT/MLS1 when getting an active *TRIGGER*/*STATE* input in normal or emergency mode respectively **when SGE1=1**; writeable only for test purposes when DEN=0

In the case of a change of the direction the wrong number of pulses is corrected twice:

Add the difference between NMB_T and INC_CNT1 twice to INC_CNT1 before sending out the correction pulses.

Note: This value can only be written when the DPLL is disabled.

Bit 31:24 **Reserved**

Note: Read as zero, should be written as zero.

18.12.23 Register DPLL_INC_CNT2

Address Offset:	see Appendix B																Initial Value:							0x0000_0000								
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								INC_CNT2																							
Mode	R								RPw																							
Initial Value	0x00								0x000000																							

Counter Value of sent SUB_INC2 values (for SMC=1 and RMO=1)

Bit 23:0 **INC_CNT2:** Actual number of pulses to be still sent out at the current increment until the next active input signal in automatic end mode;

Automatic addition of the number of demanded pulses MLS2 when getting an active *TRIGGER/STATE* input in normal or emergency mode respectively **when SGE2=1**;

writeable only for test purposes when DEN=0

In the case of a change of the direction the wrong number of pulses is corrected twice:

Add the difference between NMB_S and INC_CNT2 twice to INC_CNT2 before sending out the correction pulses.

Note: This value can only be written when the DPLL is disabled.

Bit 31:24 **Reserved**

Note: Read as zero, should be written as zero.

18.12.24 Register DPLL_APT_SYNC

<table>
<tr><th>Address Offset:</th><th colspan="16">see Appendix B</th><th colspan="16">Initial Value: 0x0000_0000</th></tr>
<tr><td></td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>Bit</td><td colspan="8">Reserved</td><td colspan="10">APT_2B_OLD</td><td colspan="7">Reserved</td><td>APT_2B_STATUS</td><td colspan="6">APT_2B_EXT</td></tr>
<tr><td>Mode</td><td colspan="8">R</td><td colspan="10">RW</td><td colspan="7">R</td><td>RW</td><td colspan="6">RW</td></tr>
<tr><td>Initial Value</td><td colspan="8">0x00</td><td colspan="10">0x000</td><td colspan="7">0</td><td>0</td><td colspan="6">0x000</td></tr>
</table>

TRIGGER Time Stamp Field Offset at Synchronization Time

Bit 5:0 **APT_2B_EXT: Address pointer 2b extension;** this offset value determines, by which value the APT_2B is changed at the synchronization time; set by CPU before the synchronization is performed.

This offset value is the number of virtual increments to be inserted in the TSF for an imminent intended synchronization; the CPU sets its value dependent on the gaps until the synchronization time taking into account the considered NUTE value to be set and including the next future increment (when SYN_T_OLD is still 1). When the synchronization takes place, this value is to be added to the APT_2B address pointer (for forward direction, DIR1=0) and the APT_2B_STATUS bit is cleared after it. For backward direction subtract APT_2B_EXT accordingly. This correction is done after updating the RAM TSF with the last TS_T value.

Note: When the synchronization is intended and the NUTE value is to be set to FULL_SCALE after it, the APT_2B_EXT value must be set to 2*SYN_NT in order to be able to fill all gaps in the extended TSF_T with the corresponding values by the CPU.

When still not all values for FULL_SCALE are available, the APT_2B_EXT value considers only a share according to the corresponding NUTE value to be set after the synchronization.

Bit 6 **APT_2B_STATUS: Address pointer 2b status;** set by CPU before the synchronization is performed. The value is cleared when the APT_2B_OLD value is written.
0 = APT_2B_EXT is not to be considered.
1 = APT_2B_EXT has to be considered for time stamp field extension.

Bit 13:7 **Reserved**
Note: Read as zero, should be written as zero.

Bit 23:14 **APT_2B_OLD: Address pointer TRIGGER for RAM region 2b at synchronization time;** this value is set by the current APT_2B value when the synchronization takes place for the first active TRIGGER event after writing APT_2C but before adding the offset value APT_2B_EXT (that means: when APT_2B_STATUS=1).

Address pointer APT_2B value at the moment of synchronization, before the offset value is added, that means the pointer with this value points to the last value before the additional inserted gap

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

18.12.25 Register DPLL_APS_SYNC

Address Offset:	**see Appendix B**																**Initial Value:**									**0x0000_0000**						
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved												APS_1C2_OLD						Reserved							APS_1C2_STATU	APS_1C2_EXT					
Mode	R												RW						R							RW	RW					
Initial Value	0x000												0x00						0							0	0x00					

STATE Time Stamp Field Offset at Synchronization Time

Bit 5:0 **APS_1C2_EXT: Address pointer 1c2 extension;** this offset value determines, by which value the APS_1C2 is changed at the synchronization time; set by CPU before the synchronization is performed.

This offset value is the number of virtual increments to be inserted in the TSF for an imminent intended synchronization; the CPU sets its value dependent on the gaps until the synchronization time taking into account the considered NUSE value to be set and including the next future increment (when SYN_S_OLD is still 1). When the synchronization takes place, this value is to be added to the APS_1C2 address pointer (for forward direction, DIR2=0) and the APT_1c2_status bit is cleared after it. For backward direction subtract APS_1C2_EXT accordingly.

Note: When the synchronization is intended and the NUSE value is to be set to FULL_SCALE after it, the APS_1C2_EXT value must be set to SYN_NS (for SYSF=1) or 2*SYN_NS (for SYSF=0) in order to be able to fill all gaps in the extended TSF_S with the corresponding values by the CPU.

When still not all values for FULL_SCALE are available, the APS_1C2_EXT value considers only a share according to the NUSE value to be set after the synchronization.

Bit 6 **APS_1C2_STATUS: Address pointer 1c2 status;** set by CPU before the synchronization is performed. The value is cleared automatically when the APS_1C2_OLD value is written.
0 = APS_1C2_EXT is not to be considered.
1 = APS_1C2_EXT has to be considered for time stamp field extension.

Bit 13:7 **Reserved**
Note: Read as zero, should be written as zero.

Bit 19:14 **APS_1C2_OLD: Address pointer STATE for RAM region 1c2 at synchronization time;** this value is set by the current APS_1C2 value when the synchronization takes place for the first active STATE event after writing APS_1C3 but before adding the offset value APS_1C2_EXT (that means: when APS_1C2_STATUS=1).
Address pointer APS_1C2 value at the moment of synchronization, before the offset value is added, that means the pointer with this value points to the last value before the additional inserted gap

Bit 31:20 **Reserved**
Note: Read as zero, should be written as zero.

Note: This register is only used when DPLL_CTRL_11.STATE_EXT is not set. If DPLL_CTRL_11.STATE_EXT is set any read/write access to this register will return AEI_STATUS = 0b10.

18.12.26 Register DPLL_TBU_TS0_T

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_0000**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								TBU_TS0_T																							
Mode	R								RPw																							
Initial Value	0x00								0x0000 00																							

Time Stamp Value for the last active TRIGGER

Bit 23:0 **TBU_TS0_T:** value of TBU_TS0 at the last TRIGGER event; for each T_valid the value of TBU_TS0 is stored in this register;

the register is writeable only for test purposes when DEN=0.

Note: This value can only be written when the DPLL is disabled.

Bit 31:24 **Reserved**

Note: Read as zero, should be written as zero.

18.12.27 Register DPLL_TBU_TS0_S

Address Offset:	see Appendix B																Initial Value: 0x0000_0000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								TBU_TS0_S																							
Mode	R								RPw																							
Initial Value	0x00								0x0000 00																							

Time Stamp Value for the last active STATE

Bit 23:0 **TBU_TS0_S:** value of TBU_TS0 at the last STATE event;

for each S_VALID the value of TBU_TS0 is stored in this register; the register is writeable only for test purposes when DEN=0.

Note: This value can only be written when the DPLL is disabled.

Bit 31:24 **Reserved**

Note: Read as zero, should be written as zero.

18.12.28 Register DPLL_ADD_IN_LD1

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_0000**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								ADD_IN_LD1																							
Mode	R								RW																							
Initial Value	0x00								0x000000																							

ADD_IN Value in Direct Load Mode for TRIGGER

Bit 23:0 **ADD_IN_LD_1:** Input value for SUB_INC1 generation, given by CPU. This value can be used in normal und emergency mode (SMC=0) as well as for SMC=1.

For DLM1 = 1:

Note: The value is loaded by the CPU but used by the DPLL only for DLM1=1 (see DPLL_CTRL_1 register). When switching DLM1 to 1, the value in the register is used for the SUB_INC1 generation beginning from the next active *TRIGGER* or *STATE* event respectively independently if new values are written by the CPU or not.

Note: When a new value is written the output frequency changes according to the given value beginning immediately from the moment of writing. Do not wait for performing step 10 in the state machine for ADD_IN calculations.

Note: If the ADD_IN_LD1 value is zero all pulses are sent with the highest possible frequency.

For DLM1 = 0:

Note: The value loaded by the CPU is stored directly in the internal add_in register which is used to control the sub increment pulse generator directly (see DPLL_CTRL_1 register, DLM1 = 0).

Note: When a new ADD_IN_LD1 value is written the output frequency is immediately changed from the moment of writing. The ADD_IN values calculated internally of the DPLL are written to the internal ADD_IN register as well. In the moment when the internal calculation of the ADD_IN values is writing the results into the internal ADD_IN register of the pulse generator the internally calculated ADD_IN values does always have higher priority compared to the values written via the ADD_IN_LD1 register.

Note: If the ADD_IN_LD1 value is zero all pulses are sent with the highest possible frequency.

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

18.12.29 Register DPLL_ADD_IN_LD2

Address Offset:	**see Appendix B**																**Initial Value: 0x0000_0000**															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								ADD_IN_LD2																							
Mode	R								RW																							
Initial Value	0x00								0x0000 00																							

ADD_IN Value in Direct Load Mode for STATE

Bit 23:0 **ADD_IN_LD_2:** Input value for SUB_INC2 generation, given by CPU. This value can be used for SMC=1 while RMO=1.

For DLM2 = 1:

Note: The value is loaded by the CPU but used by the DPLL only for DLM2=1 (see DPLL_CTRL_1 register). When switching DLM2 to 1, the value in the register is used for the SUB_INC2 generation beginning from the next *STATE* event respectively independently if new values are written by the CPU or not.

Note: When a new value is written the output frequency changes according to the given value beginning immediately from the moment of writing. Do not wait for performing step 30 in the state machine for ADD_IN calculations.

Note: If the ADD_IN_LD2 value is zero all pulses are sent with the highest possible frequency.

For DLM2 = 0:

Note: The value loaded by the CPU is stored directly in the internal add_in register which is used to control the sub increment pulse generator directly (see DPLL_CTRL_1 register, DLM2 = 0).

Note: When a new ADD_IN_LD2 value is written the output frequency is immediately changed from the moment of writing. The ADD_IN values calculated internally of the DPLL are written to the internal ADD_IN register as well. In the moment when the internal calculation of the ADD_IN values is writing the results into the internal ADD_IN register of the pulse generator the internally calculated ADD_IN values does always have higher priority compared to the values written via the ADD_IN_LD2 register.

Note: If the ADD_IN_LD2 value is zero all pulses are sent with the highest possible frequency.

Bit 31:24 **Reserved**

Note: Read as zero, should be written as zero.

18.12.30 Register DPLL_STATUS

Address Offset:	**see Appendix B**																**Initial Value: 0x0000_0000**															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	ERR	LOCK1	FTD	FSD	SYT	SYS	LOCK2	Reserved	BWD1	BWD2	ITN	ISN	CAIP1	CAIP2	CSVT	CSVS	LOW_RES	Reserved		RAM2_ERR	MT	TOR	MS	SOR	PSE	RCT	RCS	CRO	CTO	Reserved	CSO	FPCE
Mode	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R		RCw	RCw	RCw	RCw	RCw	R	R	R	RCw	RCw	R	RCw	RCw
Initial Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0x00		0	0	0	0	0	0	0	0	0	0	0	0	0

Status Register

Bit 0 **FPCE: Fast pulse correction error**

0 = No error at fast pulse correction detected

1 = negative value of MPVAL1/2 used for fast pulse correction mode

Bit 1 **CSO: Calculated STATE duration overflow;** Bit is set when equations DPLL-10a or DPLL-10b lead to an overflow

0 = No overflow at equation DPLL-10a or b

1 = overflow at equation DPLL-10a or b

Bit 2 **Reserved**

Note: Read as zero, should be written as zero.

Bit 3 **CTO: Calculated TRIGGER duration overflow;** Bit is set when equations DPLL-5a or DPLL-5b lead to an overflow

0 = No overflow at equation DPLL-5a or b

1 = overflow at equation DPLL-5a or b

Note: When one of the above bits is set the corresponding register contains the maximum value 0xFFFFFF.

Bit 4 **CRO: Calculated Reciprocal value overflow;** Bit is set when the calculation of RDT_T_ACT or RDT_S_ACT leads to an overflow

0 = No overflow at any reciprocal calculation
1 = overflow for at least one reciprocal calculation

Note: An overflow in calculation of reciprocal values can occur, when the condition of Note [3] to the DPLL_CTRL_0 register is violated (see chapter 18.12.1). Such an overflow can occur according to the calculations in equations (DPLL-1c) or (DPLL-6c).

The overflow is detected when after the calculation and shifting left 32 bits at least one of the bits 31 to 24 is not zero. In that case the corresponding register is set to 0xFFFFFF.

Bit 5 **RCS: Resolution conflict STATE**.
0 = No resolution conflict detected
1 = the TS0_HRS value is set to 1 while LOW_RES=0

Bit 6 **RCT: Resolution conflict TRIGGER**.
0 = No resolution conflict detected
1 = the TS0_HRT value is set to 1 while LOW_RES=0

Bit 7 **PSE: Prediction space configuration error**
0 = No prediction space error detected
1 = Configured offset value of RAM2 is too small in order to store all TNU+1 values twice in FULL_SCALE

Bit 8 **SOR[7]: *STATE* out of range**
0 = all *STATE* signal events appear within SLR interval or a direction change was detected
1 = at least one *STATE* signal event is out of SLR;

Bit 9 **MS: Missing *STATE*** detected according to SOV.
0 = No missing *STATE* detected or a new active *STATE* slope occurred
1 = At least one missing *STATE* detected after the last active slope

Bit 10 **TOR[8]: *TRIGGER* out of range**
0 = all *TRIGGER* signal events appear within TLR interval or a direction change was detected
1 = at least one *TRIGGER* signal event is out of TLR;

Bit 11 **MT: Missing *TRIGGER*** detected according to TOV
0 = No missing *TRIGGER* detected or a new active *TRIGGER* slope occurred
1 = At least one missing *TRIGGER* detected after the last active slope

Bit 12 **RAM2_ERR**: DPLL internal access to not configured RAM2 memory space
0 = No access to not configured RAM2 memory space

1 = access to not configured RAM2 memory space

Bit 14:13 **Reserved**

Note: Read as zero, should be written as zero.

Bit 15 **LOW_RES: low resolution** of TBU_TS0 is used for DPLL input; this value reflects the input signal LOW_RES

0 = the lower 24 Bits of TBU_TS0 are used as input for the DPLL

1 = the higher 24 Bits of TBU_TS0 are used as input for the DPLL

Bit 16 **CSVS: Current signal value STATE**

0 = the last STATE_S value was 0

1 = the last STATE_S value was 1

Bit 17 **CSVT: Current signal value TRIGGER**

0 = the last TRIGGER_S value was 0

1 = the last TRIGGER_S value was 1

Bit 18 **CAIP2: Calculation of upper half actions in progress**

0 = currently no action calculation, new data requests possible

1 = action calculation in progress, no new data requests possible

Bit 19 **CAIP1: Calculation of lower half actions in progress**

0 = currently no action calculation, new data requests possible

1 = action calculation in progress, no new data requests possible

Bit 20 **ISN: Increment** number of **STATE** is **not** plausible; Bit is set when the number of STATES is different to profile

0 = the number of *STATE* events between synchronization gaps is plausible, a direction change is detected or the APS_1C3 pointer is written

1 = after setting LOCK1 in emergency mode (SMC=0 and RMO=1) or LOCK2 for SMC=RMO=1 missing or additional *STATE* signals detected; bit is cleared when a direction change is detected or the APS_1C3 is written

Bit 21 **ITN: Increment** number of **TRIGGER** is **not** plausible; Bit is set when the number of TRIGGERS is different to profile

0 = the number of *TRIGGER* events between synchronization gaps is plausible, a direction change is detected or the address pointer APT_2C is written

1 = after setting LOCK1 in normal mode (for SMC=0 or SMC=1) or in emergency mode (only for SMC=0) for missing or additional *TRIGGER* signals detected; bit is cleared when a direction change is detected or the APT_2C is written

Bit 22 **BWD2: Backwards drive** of SUB_INC**2**

0 = forward direction

1 = backward direction

Bit 23 **BWD1: Backwards drive** of SUB_INC**1**

Note: see bit 22

Bit 24 **Reserved**
Note: Read as zero, should be written as zero.

Bit 25 **LOCK2:** DPLL **Lock status** concerning SUB_INC**2**
0 = The DPLL is not locked concerning *STATE* for SMC=1
1 = The DPLL is locked concerning *STATE* for SMC=1
Note: Locking of SUB_INC2 appears
for RMO=SMC=1: Bit is set, when SYS is set and the number of events between two missing *STATE*s is as expected by the SYN_S values.

Note: LOCK2 is set
for SMC=RMO=1:
for an active STATE event when SYS is set and SYN_NS=0
or when SYS is set and the profile stored in the ADT_Si field matches once between two gaps.

LOCK2 is reset: for SMC=RMO=1
- when a missing STATE event occurs while SYN_S=1. This does mean an unexpected missing STATE.
- when the corresponding input signal STATE is out of locking range SLR

Bit 26 **SYS: Synchronizatio**n condition of ***STATE*** fixed.
This bit is set when the CPU writes to the APS_1C3 address pointer.

Bit 27 **SYT: Synchronizatio**n condition of ***TRIGGER*** fixed.
This bit is set when the CPU writes to the APT_2C address pointer.

Bit 28 **FSD: First *STATE* detected**.
0 = Still no active *STATE* event was detected after enabling DPLL
1 = At least one active *STATE* event was detected after enabling DPLL

Note: No change of FSD for switching from normal to emergency mode or vice versa.

Bit 29 **FTD: First *TRIGGER* detected**.
0 = No active *TRIGGER* event was detected after enabling DPLL
1 = At least one active *TRIGGER* event was detected after enabling DPLL

Note: No change of FTD for switching from normal to emergency mode or vice versa.

Bit 30 **LOCK1:** DPLL **Lock status** concerning SUB_INC**1**

0 = The DPLL is not locked for *TRIGGER*
(while SMC=RMO=0 or SMC=1)
or for *STATE* (while SMC=0 and RMO=1)
1 = The DPLL is locked for *TRIGGER*
(while SMC=RMO=0 or SMC=1)
or for *STATE* (while SMC=0 and RMO=1)

Note: LOCK1 is set :

- in normal mode (for RMO=SMC=0, LCD=0): Bit is set for an active TRIGGER event when SYT is set and the number of events between two gaps is as expected by the profile (NT values in the ADT_T[i] field) or when SYN_NT=0 and SYT=1.
- in normal mode (for RMO=SMC=0, LCD=1): Bit is set for an active TRIGGER event when SYT is set and the number of events between two increments without missing TRIGGER (no gap) is as expected by the profile (NT values in the ADT_T[i] field).
- in emergency mode (for RMO=1 and SMC=0): Bit is set for an active STATE event, when SYS is set and the received events are in correspondence to the profile (NS values in the ADT_S[i] field) for at least two expected missing STATE events or when SYN_NS=0.
- for SMC=1: Bit is set for an active TRIGGER even when SYT is set and SYN_NT=0 or when SYT is set and the profile stored in the ADT_T[i] field matches once between two gaps.

LOCK1 is reset
for RMO=SMC=0:

- when a corresponding missing TRIGGER event occurs while SYN_T=1. This does mean an unexpected missing TRIGGER.
- when the corresponding input signal TRIGGER is out of locking range TLR,
- when a corresponding direction change is detected

for RMO=1 and SMC=0:

- when a corresponding missing STATE event occurs while SYN_S=1. This does mean an unexpected missing STATE.
- when the corresponding input signal STATE is out of locking range TLR

for SMC=1:

- when a corresponding missing TRIGGER event occurs while SYN_T=1. This does mean an unexpected missing TRIGGER.
- when the corresponding input signal TRIGGER is out of locking range TLR,
- when a corresponding direction change is detected

Bit 31 **ERR: Error** during configuration or operation resulting in unexpected values.
0 = when all bits in position 8 to 0 and 10 and 12 are zero
1 = when at least one bit in position 8 to 0 or 10 or 12 is one

[7] The SOR bit is set, when the time to the next active *STATE* slope exceeds the value of the last nominal *STATE* duration multiplied with the value of the SLR register (see chapter 18.12.73) and is reset, when at the current or last active input event a direction change was detected. The SYS bit is not influenced by setting the SOR bit.

[8] The TOR bit is set, when the time to the next active *TRIGGER* slope exceeds the value of the last nominal *TRIGGER* duration multiplied with the value of the TLR register (see chapter 18.12.72) and is reset, when at the current or last active input event a direction change was detected. The SYT bit is not influenced by setting the TOR bit

The DPLL_STATUS register is reset, when the DPLL is disabled (switching DEN from 1 to 0).

18.12.31 Register DPLL_ID_PMTR_[z] (z:0...NOAC-1)

Address Offset:	**see Appendix B**																**Initial Value:**							**0x0000_01FE**								
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																							ID_PMTR_X								
Mode	R																							RPw								
Initial Value	0x000000																							0x1FE								

ID Information for Input Signal PMTR[z] (Position minus Time Request), z=0...NOAC-1[1)]

Bit 8:0 **ID_PMTR_X:** ID information to the input signal PMTR[i] from the ARU.
Note: This value can only be written when the action [i] is disabled by the correspondent bit AENi=0 of the registers DPLL_CTRL_2, ...5 respectively
or when the DPLL is disabled (DEN=0).

Bit 31:9 **Reserved**
Note: Read as zero, should be written as zero.

18.12.32 Register DPLL_CTRL_0_SHADOW_TRIGGER

Address Offset:	see Appendix B																Initial Value:								0x0000_0257							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	RMO	Reserved		IDT	Reserved	AMT	Reserved															IFP	MLT									
Mode	R	R		R	R	R	R															R	R									
Initial Value	0	00		0	0	0	0x0000															0	0x257									

Shadow Register of DPLL_CTRL_0 controlled by an active TRIGGER Slope

Bit 9:0 **MLT [1]: multiplier for TRIGGER;** MLT+1 is number of *SUB_INC1* pulses between two *TRIGGER* events in normal mode (1...1024);

Bit 10 **IFP [1]: Input filter position;** value contains position or time related information.

Bit 25:11 **reserved**
Note: Read as zero, should be written as zero.

Bit 26 **AMT [1]: Adapt mode TRIGGER;** Use of adaptation information of *TRIGGER*.

Bit 27 **reserved**
Note: Read as zero, should be written as zero.

Bit 28 **IDT [1]: Input delay TRIGGER;** use of input delay information transmitted in FT part of the *TRIGGER* signal.

Bit 30:29 **reserved**.
Note: Read as zero, should be written as zero.

Bit 31 **RMO [1]: Reference mode;** selection of the relevant the input signal for generation of SUB_INC1.

Note: Only the values characterized by [1] are stored for an active TRIGGER slope. All other values remain 0. When DEN=0 the relevant bit values of the original register DPLL_CTRL_0 are transferred without any input event at the next system clock. This results in the above reset value.

18.12.33 Register DPLL_CTRL_0_SHADOW_STATE

Address Offset:	see Appendix B	Initial Value:	0x0000_0000

	31	30 29 28	27	26	25	24 23 22 21 20 19 18 17 16 15 14 13 12 11	10	9 8 7 6 5 4 3 2 1 0
Bit	RMO	Reserved	IDS	Reserved	AMS	Reserved	IFP	Reserved
Mode	R	R	R	R	R	R	R	R
Initial Value	0	000	0	0	0	0x0000	0	0x000

Shadow Register of DPLL_CTRL_0 controlled by an active STATE Slope

Bit 9:0 **reserved**
Note: Read as zero, should be written as zero.

Bit 10 **IFP [2]: Input filter position;** value contains position or time related information.

Bit 24:11 **reserved**
Note: Read as zero, should be written as zero.

Bit 25 **AMS [2]: Adapt mode STATE;** Use of adaptation information of *STATE*.

Bit 26 **reserved**
Note: Read as zero, should be written as zero.

Bit 27 **IDS [2]: Input delay STATE;** Use of input delay information transmitted in FT part of the *STATE* signal.

Bit 30:28 **reserved**
Note: Read as zero, should be written as zero.

Bit 31 **RMO [2]: Reference mode;** selection of the relevant the input signal for generation of SUB_INC1.

Note: Only the values characterized by [2] are stored for an active STATE slope. All other values remain 0. When DEN=0 the relevant bit values of the original register DPLL_CTRL_0 are transferred without any input event at the next system clock.

18.12.34 Register DPLL_CTRL_1_SHADOW_TRIGGER

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																								PCM1	DLM1	SGE1	PIT	COA	Reserved		DMO
Mode	R																								R	R	R	R	R	R		R
Initial Value	0x00_0000																								0	0	0	0	0	00		0

Shadow Register of DPLL_CTRL_1 controlled by an active TRIGGER Slope

Bit 0 **DMO [1]: DPLL mode select**.

Bit 2:1 **reserved**

Note: Read as zero, should be written as zero.

Bit 3 **COA [1]: Correction strategy in automatic end mode** (DMO=0).

Bit 4 **PIT [1]: Plausibility** value PVT to next active TRIGGER is **time related**

Bit 5 **SGE1 [1]: SUB_INC1 generator enable**.

Bit 6 **DLM1 [1]: Direct Load Mode** for SUB_INC1 generation

Bit 7 **PCM1 [1]: Pulse Correction Mode** for SUB_INC1 generation.

Bit 31:8 **reserved**

Note: Read as zero, should be written as zero.

Note: Only the values characterized by [1] are stored for an active TRIGGER slope. All other values remain 0. When DEN=0 the relevant bit values of the original register DPLL_CTRL_1 are transferred without any input event at the next system clock.

18.12.35 Register DPLL_CTRL_1_SHADOW_STATE

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																					PCM2	DLM2	SGE2	PCM1	DLM1	SGE1	Reserved	COA	Reserved		DMO
Mode	R																					R	R	R	R	R	R	R	R	R		R
Initial Value	0x00000																					0	0	0	0	0	0	0	0	00		0

DPLL Shadow Register of DPLL_CTRL_1 controlled by an active STATE Slope

Bit 0 **DMO [2]: DPLL mode select**.

Bit 2:1 **reserved**
Note: Read as zero, should be written as zero.

Bit 3 **COA [2]: Correction strategy in automatic end mode** (DMO=0).

Bit 4 **reserved**
Note: Read as zero, should be written as zero.

Bit 5 **SGE1 [2]: SUB_INC1 generator enable.**

Bit 6 **DLM1[2]: Direct Load Mode** for SUB_INC1 generation

Bit 7 **PCM1 [2]: Pulse Correction Mode** for SUB_INC1 generation.

Bit 8 **SGE2 [2]: SUB_INC2 generator enable.**

Bit 9 **DLM2 [2]: Direct Load Mode** for SUB_INC2 generation

Bit 10 **PCM2 [2]: Pulse Correction Mode** for SUB_INC2 generation.

Bit 31:11 **reserved**
Note: Read as zero, should be written as zero.

Note: Only the values characterized by [2] are stored for an active STATE slope. All other values remain 0. When DEN=0 the relevant bit values of the original register DPLL_CTRL_1 are transferred without any input event at the next system clock.

18.12.36 Register DPLL_RAM_INI

Address Offset:	see Appendix B																Initial Value: 0x0000_0000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																											INIT_RAM	Reserved	INIT_2	INIT_1BC	INIT_1A
Mode	R																											RAw	R	R	R	R
Initial Value	0x0000_000																											0	0	0	0	0

Register to control the RAM Initialization

Bit 0 **INIT_1A:** RAM region 1a initialization in progress
0 = No initialization of considered RAM region in progress
1 = Initialization of considered RAM region in progress

Bit 1 **INIT_1BC:** RAM region 1b and 1c initialization in progress
see bit 0

Bit 2 **INIT_2:** RAM region 2 initialization in progress
see bit 0

Bit 3 **Reserved**
Note: Read as zero, should be written as zero.

Bit 4 **INIT_RAM:** RAM regions 1a, 1b and 2 are to be initialized.
0 = Do not start initialization of all RAM regions

1 = Start initialization of all RAM regions

Note: Setting the INIT_RAM bit results only in a RAM reset when the DPLL is not enabled (DEN=0).

Note: Depending on the vendor configuration the connected RAM regions are initialized to zero in the case of a module HW reset or for setting the RST bit in the GTM_RST register.

Note: In the case of no RAM initialization it must be ensured that all relevant parameters are configured correctly. Otherwise there is no guarantee to get a predictable behavior.

Bit 31:5 **Reserved**
Note: Read as zero, should be written as zero.

18.12.37 Memory DPLL_TS_T

Address Offset:	see Appendix B																Initial Value: 0x0000_0000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								TRIGGER_TS																							
Mode	R								RW																							
Initial Value	0x00								0x0000 00																							

Actual TRIGGER Time Stamp Value

Bit 23:0 **TRIGGER_TS:** Time stamp value of the last active *TRIGGER* input. measured TRIGGER time stamp
Note: The LSB address is determined using the SWON_T value in the OSW register (see chapter 18.12.8).

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

18.12.38 Memory DPLL_TS_T_OLD

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								TRIGGER_TS_OLD																							
Mode	R								RW																							
Initial Value	0x00								0x0000																							

Previous TRIGGER Time Stamp Value

Bit 23:0 **TRIGGER_TS_OLD:** Time stamp value of the last but one active TR*IGGER* input.
previous measured TRIGGER time stamp
Note: The LSB address is determined using the SWON_T value in the OSW register (see chapter 18.12.8).

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

18.12.39 Memory DPLL_FTV_T

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								TRIGGER_FT																							
Mode	R								RW																							
Initial Value	0x00								0x0000																							

Actual TRIGGER Filter value

Bit 23:0 **TRIGGER_FT:** Filter value of the last active *TRIGGER* input.
transmitted filter value
Note: The LSB address is determined using the SWON_T value in the OSW register (see chapter 18.12.8).

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

18.12.40 Memory DPLL_TS_S

<table>
<tr><th>Address Offset:</th><th colspan="16">see Appendix B</th><th colspan="16">Initial Value: 0x0000_0000</th></tr>
<tr><td></td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>Bit</td><td colspan="8">Reserved</td><td colspan="24">STATE_TS</td></tr>
<tr><td>Mode</td><td colspan="8">R</td><td colspan="24">RW</td></tr>
<tr><td>Initial Value</td><td colspan="8">0x00</td><td colspan="24">0x000000</td></tr>
</table>

Actual STATE Time Stamp Register

Bit 23:0 **STATE_TS:** Time stamp value of the last active *STATE* input.
Note: The LSB address is determined using the SWON_S value in the OSW register (see chapter 18.12.8).

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

18.12.41 Memory DPLL_TS_S_OLD

<table>
<tr><th>Address Offset:</th><th colspan="16">see Appendix B</th><th colspan="16">Initial Value: 0x0000_0000</th></tr>
<tr><td></td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>Bit</td><td colspan="8">Reserved</td><td colspan="24">STATE_TS_OLD</td></tr>
<tr><td>Mode</td><td colspan="8">R</td><td colspan="24">RW</td></tr>
<tr><td>Initial Value</td><td colspan="8">0x00</td><td colspan="24">0x000000</td></tr>
</table>

Previous STATE Time Stamp Register

Bit 23:0 **STATE_TS_OLD:** Time stamp value of the last active *STATE* input.

Note: The LSB address is determined using the SWON_S value in the OSW register (see chapter 18.12.8).

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

18.12.42 Memory DPLL_FTV_S

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_0000**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								STATE_FT																							
Mode	R								RW																							
Initial Value	0x00								0x0000																							

Actual STATE Filter Value

Bit 23:0 **STATE_FT:** Filter value of the last active *STATE* input.
transmitted filter value
Note: The LSB address is determined using the SWON_S value in the OSW register (see chapter 18.12.8).

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

18.12.43 Memory DPLL_THMI

Address Offset:	**see Appendix B**		**Initial Value: 0x0000_0000**
	31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Bit	Reserved	NOT_USED	THMI
Mode	R	RW	RW
Initial Value	0x00	0x00	0x0000

TRIGGER Hold Time Min. Value

Bit 15:0 **THMI:** minimal time between active and inactive *TRIGGER* slope (uint16); the time value corresponds to the time stamp clock counts: this does mean the clock selected for the TBU_CH0_BASE (see TBU_CH0_CTRL register)
set min. value; generate the TINI interrupt in the case of a violation for THMI>0.

Note: Typical retention time values after an active slope can be e.g. between 45 µs (forwards) and 90 µs (backwards). When THMI is zero, consider always a THMI violation (forwards).

Bit 23:16 **Not used**
Note: must be written to zero.

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

18.12.44 Memory DPLL_THMA

Address Offset:	**see Appendix B**		**Initial Value: 0x0000_0000**
	31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Bit	Reserved	NOT_USED	THMA
Mode	R	RW	RW
Initial Value	0x00	0x00	0x0000

TRIGGER Hold Time Max. Value

Bit 15:0 **THMA:** maximal time between active and inactive *TRIGGER* slope (uint16); the time value corresponds to the time stamp clock counts: this does mean the clock selected for the TBU_CH0_BASE (see TBU_CH0_CTRL register)

max. value to be set; generate the TAX interrupt in the case of a violation for THMA>0.

Bit 23:16 **Not used**

Note: must be written to zero.

Bit 31:24 **Reserved**

Note: Read as zero, should be written as zero.

18.12.45 Memory DPLL_THVAL

Address Offset:	see Appendix B																Initial Value: 0x0000_0000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								THVAL																							
Mode	R								RW																							
Initial Value	0x00								0x000000																							

Measured TRIGGER Hold Time Value

Bit 23:0 **THVAL:** measured time from the last active slope to the next inactive *TRIGGER* slope in time stamp clock counts: this does mean the clock selected for the TBU_CH0_BASE (uint16);

The measured value considers all input slope filter delays. From the received input the corresponding filter delays are subtracted before the time stamp difference of active and inactive slope is calculated.

Bit 31:24 **Reserved**

Note: Read as zero, should be written as zero.

Note: In the case of LOW_RES=1 and TBU_HRT=0 the difference between the time stamps of active and inactive slope is multiplied by 8. The register contains this value.

18.12.46 Memory DPLL_TOV

<table>
<tr><td><b>Address Offset:</b></td><td colspan="16"><b>see Appendix B</b></td><td colspan="16"><b>Initial Value: 0x0000_0000</b></td></tr>
<tr><td></td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>Bit</td><td colspan="8">Reserved</td><td colspan="8">NOT_USED</td><td colspan="6">TOV_DW</td><td colspan="10">TOV_DB</td></tr>
<tr><td>Mode</td><td colspan="8">R</td><td colspan="8">RW</td><td colspan="6">RW</td><td colspan="10">RW</td></tr>
<tr><td>Initial Value</td><td colspan="8">0x00</td><td colspan="8">0x00</td><td colspan="6">0x00</td><td colspan="10">0x000</td></tr>
</table>

Time Out Value of Active TRIGGER Slope (for missing TRIGGER generation)

Bit 9:0 **DB:** Decision value (fractional part) for missing TRIGGER interrupt.

Bit 15:10 **DW:** Decision value (integer part) for missing TRIGGER interrupt.

TOV(15:0) is to be multiplied with the duration of the last increment and divided by 1024 in order to get the time-out time value for a missing TRIGGER event

Note: For the case of LOW_RES=1 (see DPLL_STATUS register) consider for the calculation of the time out value the following cases:

LOW_RES=1 and DPLL_CTRL_1/TS0_HRT=1:
multiply the TBU_TS0 value by 8

LOW_RES=1 and DPLL_CTRL_1/TS0_HRT=0:
multiply the TBU_TS0 value by 8
multiply the estimated time point value (using TS_T, dt_t_ACT and TOV) by 8

LOW_RES=0 and DPLL_CTRL_1/TS0_HRT=0:
use TBU_TS0 and the estimated time point value unchanged.

Bit 23:16 **Not used**
Note: must be written to zero.

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

18.12.47 Memory DPLL_TOV_S

Address Offset:	see Appendix B																Initial Value:						0x0000_0000									
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								NOT_USED								DW						DB									
Mode	R								RW								RW						RW									
Initial Value	0x00								0x00								0x00						0x000									

Time Out Value of active STATE Slope (for missing STATE generation)

Bit 9:0 **DB:** Decision value (fractional part) for missing STATE interrupt.

Bit 15:10 **DW:** Decision value (integer part) for missing STATE interrupt.

TOV_S (15:0) is to be multiplied with the duration of the last increment and divided by 1024 in order to get the time-out time value for a missing STATE event.

Note: For the case of LOW_RES=1 (see DPLL_STATUS register) consider for the calculation of the time out value the following cases:

LOW_RES=1 and DPLL_CTRL_1/TS0_HRS=1:
multiply the TBU_TS0 value by 8

LOW_RES=1 and DPLL_CTRL_1/TS0_HRS=0:
multiply the TBU_TS0 value by 8
multiply the estimated time point value (using TS_T, dt_s_ACT and SOV) by 8

LOW_RES=0 and DPLL_CTRL_1/TS0_HRS=0:
use TBU_TS0 and the estimated time point value unchanged.

Bit 23:16 **Not used**
Note: must be written to zero.

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

18.12.48 Memory DPLL_ADD_IN_CAL1

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								ADD_IN_CAL1																							
Mode	R								RW																							
Initial Value	0x00								0x0000 00																							

Calculated ADD_IN Value for SUB_INC1 Generation

Bit 23:0 **ADD_IN_CAL_1:** Calculated input value for SUB_INC1 generation, calculated by the DPLL.
calculated value
The update of the ADD_IN value by the new calculated value ADD_IN_CAL1 is suppressed for one increment when an unexpected missing TRIGGER (SMC=1 or RMO=0) or an unexpected STATE (RMO=1 and SMC=0) is detected.

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

18.12.49 Memory DPLL_ADD_IN_CAL2

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								ADD_IN_CAL2																							
Mode	R								RW																							
Initial Value	0x00								0x0000 00																							

Calculated ADD_IN Value for SUB_INC2 Generation

Bit 23:0 **ADD_IN_CAL_2:** Input value for SUB_INC2 generation, calculated by the DPLL for SMC=RMO=1.
calculated value

The update of the ADD_IN value by the calculated value ADD_IN_CAL2 is suppressed for one increment when an unexpected missing STATE (RMO=SMC=1) is detected.

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

18.12.50 Memory DPLL_MPVAL1

Address Offset:	**see Appendix B**			**Initial Value:**	**0x0000_0000**
	31 … 24	23 … 16	15 … 0		
Bit	Reserved	SIX1	MPVAL1		
Mode	R	RW	RW		
Initial Value	0x00	0x00	0x0000		

Missing Pulses to be Added or Subtracted Directly

Bit 15:0 **MPVAL1:** missing pulses for direct correction of SUB_INC1 pulses by the CPU (sint16);
used only for RMO=0 or SMC=1 for the case PCM1=1.
Add MPVAL1 once to INC_CNT1 and reset PCM1 after applying once

Bit 23:16 **SIX1:** sign extension for MPVAL1
0x00 = MPVAL1 is a positive number
0xFF = MPVAL1 is a negative number
Note: All bits must be written to either all zeros or all ones.

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

Note: Do not provide negative values which exceed the amount of NT*(MLT+1) or MLS1 respectively; when considered negative PD values the sum of both (MPVAL1 + NT*PD) should not exceed the amount of NT*(MLT+1) or MLS1 respectively.

18.12.51 Memory DPLL_MPVAL2

Address Offset:	see Appendix B																Initial Value: 0x0000_0000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								SIX2								MPVAL2															
Mode	R								RW								RW															
Initial Value	0x00								0x00								0x0000															

Missing Pulses to be Added or Subtracted Directly

Bit 15:0 **MPVAL2:** missing pulses for direct correction of SUB_INC2 pulses by the CPU (sint16);
used only for SMC=RMO=1 for the case PCM2=1. Add MPVAL2 once to INC_CNT2 and reset PCM2 after applying once

Note: Do not provide negative values which exceed the amount of MLS2; when considered negative PD_S values the sum of both (MPVAL2 + NS *PD_S) should not exceed the amount of MLS2.

Bit 23:16 **SIX2:** sign extension for MPVAL2
0x00 = MPVAL2 is a positive number
0xFF = MPVAL2 is a negative number
Note: All bits must be written to either all zeros or all ones.

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

18.12.52 Memory DPLL_NMB_T_TAR

<table>
<tr><td><b>Address Offset:</b></td><td colspan="16"><b>see Appendix B</b></td><td colspan="16"><b>Initial Value: 0x0000_0000</b></td></tr>
<tr><td></td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>Bit</td><td colspan="8">Reserved</td><td colspan="8">NOT_USED</td><td colspan="16">NMB_T_TAR</td></tr>
<tr><td>Mode</td><td colspan="8">R</td><td colspan="8">RW</td><td colspan="16">RW</td></tr>
<tr><td>Initial Value</td><td colspan="8">0x00</td><td colspan="8">0x00</td><td colspan="16">0x0000</td></tr>
</table>

Target Number of Pulses to be sent in Normal Mode

Bit 15:0 **NMB_T_TAR: Target Number of pulses for TRIGGER;** Calculated number of pulses in normal mode for the current *TRIGGER* increment without missing pulses.

calculated target pulse number

Note: The LSB address is determined using the SWON_T value in the OSW register (see chapter 18.12.8).

Bit 23:16 **Not used**

Note: must be written to zero.

Bit 31:24 **Reserved**

Note: Read as zero, should be written as zero.

18.12.53 Memory DPLL_NMB_T_TAR_OLD

<table>
<tr><td><b>Address Offset:</b></td><td colspan="16"><b>see Appendix B</b></td><td colspan="16"><b>Initial Value: 0x0000_0000</b></td></tr>
<tr><td></td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>Bit</td><td colspan="8">Reserved</td><td colspan="8">NOT_USED</td><td colspan="16">NMB_T_TAR_OLD</td></tr>
<tr><td>Mode</td><td colspan="8">R</td><td colspan="8">RW</td><td colspan="16">RW</td></tr>
<tr><td>Initial Value</td><td colspan="8">0x00</td><td colspan="8">0x00</td><td colspan="16">0x0000</td></tr>
</table>

Last but one Target Number of Pulses to be sent in Normal Mode

Bit 15:0 **NMB_T_TAR_OLD: Target Number of pulses for TRIGGER;** Calculated number of pulses in normal mode for the current *TRIGGER* increment without missing pulses.

calculated target pulse number

Note: The LSB address is determined using the SWON_T value in the OSW register (see chapter 18.12.8).

Bit 23:16 **Not used**
Note: must be written to zero.

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

18.12.54 Memory DPLL_NMB_S_TAR

Address Offset:	**see Appendix B**																**Initial Value: 0x0000_0000**															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								NOT_USED				NMB_S_TAR																			
Mode	R								RW				RW																			
Initial Value	0x00								0x0				0x00000																			

Target Number of Pulses to be sent in Emergency Mode

Bit 19:0 **NMB_S_TAR: Target Number of pulses for STATE;** Calculated number of pulses in emergency mode for the current *STATE* increment without missing pulses.
calculated target pulse number

Note: The LSB address is determined using the SWON_S value in the OSW register (see chapter 18.12.8).

Bit 23:20 **Not used**
Note: must be written to zero.

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

18.12.55 Memory DPLL_NMB_S_TAR_OLD

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								NOT_USED				NMB_S_TAR_OLD																			
Mode	R								RW				RW																			
Initial Value	0x00								0x0				0x0000 0																			

Last but one Target Number of Pulses to be sent in Emergency Mode

Bit 19:0 **NMB_S_TAR_OLD: Target Number of pulses for STATE;** Calculated number of pulses in emergency mode for the current *STATE* increment without missing pulses.

calculated target pulse number

Note: The LSB address is determined using the SWON_S value in the OSW register (see chapter 18.12.8).

Bit 23:20 **Not used**

Note: must be written to zero.

Bit 31:24 **Reserved**

Note: Read as zero, should be written as zero.

18.12.56 Memory DPLL_RCDT_TX

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								RCDT_TX																							
Mode	R								RW																							
Initial Value	0x00								0x0000 00																							

Reciprocal Value of the Expected Increment Duration of TRIGGER

Bit 23:0 **RCDT_TX:** Reciprocal value of expected increment duration $*2^{32}$ while only the lower 24 bits are used.

calculated value; when an overflow occurs in calculation the value is set to 0xFFFFFF.

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

18.12.57 Memory DPLL_RCDT_SX

Address Offset:	see Appendix B																Initial Value: 0x0000_0000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								RCDT_SX																							
Mode	R								RW																							
Initial Value	0x00								0x000000																							

Reciprocal Value of the Expected Increment Duration of STATE

Bit 23:0 **RCDT_SX:** Reciprocal value of expected increment duration $*2^{32}$ while only the lower 24 bits are used.
calculated value; when an overflow occurs in calculation the value is set to 0xFFFFFF.

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

18.12.58 Memory DPLL_RCDT_TX_NOM

Address Offset:	see Appendix B																Initial Value: 0x0000_0000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								RCDT_TX_NOM																							
Mode	R								RW																							
Initial Value	0x00								0x000000																							

Reciprocal Value of the Expected Nominal Increment Duration of TRIGGER

Bit 23:0 **RCDT_TX_NOM:** Reciprocal value of nominal increment duration $*2^{32}$ while only the lower 24 bits are used.
calculated value; when an overflow occurs in calculation the value is set to 0xFFFFFF.

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

18.12.59 Memory DPLL_RCDT_SX_NOM

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_0000**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								RCDT_SX_NOM																							
Mode	R								RW																							
Initial Value	0x00								0x000000																							

Reciprocal Value of the Expected Nominal Increment Duration of STATE

Bit 23:0 **RCDT_SX_NOM:** Reciprocal value of nominal increment duration $*2^{32}$ while only the lower 24 bits are used.
calculated value; when an overflow occurs in calculation the value is set to 0xFFFFFF.

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

Note: RCDT_TX_NOM and RCDT_SX_NOM are calculated by the values RCDT_TX and RCDT_SX to be multiplied with SYN_T or SYN_S respectively.

18.12.60 Memory DPLL_RDT_T_ACT

Address Offset:	see Appendix B	Initial Value:	0x0000_0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								RDT_T_ACT																							
Mode	R								RW																							
Initial Value	0x00								0x0000 00																							

Reciprocal Value of the Last Increment of TRIGGER

Bit 23:0 **RDT_T_ACT:** Reciprocal value of last *TRIGGER* increment *2^{32}, only the lower 24 bits are used; the LSB is rounded up when the next truncated bit is 1.

calculated value; when an overflow occurs in calculation the value is set to 0xFFFFFF and the CRO bit in the DPLL_STATUS register is set (see chapter 18.12.30).

Bit 31:24 **Reserved**

Note: Read as zero, should be written as zero.

18.12.61 Memory DPLL_RDT_S_ACT

Address Offset:	see Appendix B	Initial Value:	0x0000_0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								RDT_S_ACT																							
Mode	R								RW																							
Initial Value	0x00								0x0000 00																							

Reciprocal Value of the Last Increment of STATE

Bit 23:0 **RDT_S_ACT:** Reciprocal value of last *STATE* increment *2^{32}, only the lower 24 bits are used; the LSB is rounded up when the next truncated bit is 1.

calculated value; when an overflow occurs in calculation the value is set to 0xFFFFFF and the CRO bit in the DPLL_STATUS register is set (see chapter 18.12.30).

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

18.12.62 Memory DPLL_DT_T_ACT

Address Offset:	**see Appendix B**																**Initial Value: 0x0000_0000**															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								DT_T_ACT																							
Mode	R								RW																							
Initial Value	0x00								0x000000																							

Duration of the Last TRIGGER Increment

Bit 23:0 **DT_T_ACT:** Calculated duration of the last *TRIGGER* increment.
calculated duration of the last increment;
Value will be written into the corresponding RAM field, when all calculations for the considered increment are done and APT is valid.

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

18.12.63 Memory DPLL_DT_S_ACT

Address Offset:	see Appendix B																Initial Value: 0x0000_0000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								DT_S_ACT																							
Mode	R								RW																							
Initial Value	0x00								0x000000																							

Duration of the Last STATE Increment

Bit 23:0 **DT_S_ACT:** Calculated duration of the last *STATE* increment.
Calculated increment duration
Value will be written into the corresponding RAM field, when all calculations for the considered increment are done and APS is valid.

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

18.12.64 Memory DPLL_EDT_T

Address Offset:	see Appendix B																Initial Value: 0x0000_0000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								EDT_T																							
Mode	R								RW																							
Initial Value	0x00								0x000000																							

Difference of Prediction to Actual Value of the Last TRIGGER Increment

Bit 23:0 **EDT_T:** *Signed* difference between actual value and a simple prediction of the last *TRIGGER* increment: **sint24**
calculated error value

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

18.12.65 Memory DPLL_MEDT_T

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								MEDT_T																							
Mode	R								RW																							
Initial Value	0x00								0x0000 00																							

Weighted Difference of Prediction Errors of TRIGGER

Bit 23:0 **MEDT_T:** *Signed* middle weighted difference between actual value and prediction of the last *TRIGGER* increments: **sint24**; only calculated for SYT=1

calculated medium error value, see chapter 18.6.2.6

The value is calculated only after synchronization (SYT=1) and the update is suppressed for one increment when an unexpected missing TRIGGER is detected.

Bit 31:24 **Reserved**

Note: Read as zero, should be written as zero.

18.12.66 Memory DPLL_EDT_S

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								EDT_S																							
Mode	R								RW																							
Initial Value	0x00								0x0000 00																							

Difference of Prediction to Actual Value of the Last STATE Increment

Bit 23:0 **EDT_S:** *Signed* difference between actual value and prediction of the last *STATE* increment: **sint24**
calculated error value, see chapter 18.6.3.5

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

18.12.67 Memory DPLL_MEDT_S

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_0000**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								MEDT_S																							
Mode	R								RW																							
Initial Value	0x00								0x0000 00																							

Weighted Difference of Prediction Errors of STATE

Bit 23:0 **MEDT_S:** *Signed* middle weighted difference between actual value and prediction of the last *STATE* increments: **sint24**; only calculated for SYS=1
calculated medium error value, see chapter 18.6.3.6
The value is calculated only after synchronization (SYS=1) and the update is suppressed for one increment when an unexpected missing STATE is detected.

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

18.12.68 Memory DPLL_CDT_TX

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								CDT_TX																							
Mode	R								RW																							
Initial Value	0x00								0x000000																							

Prediction of the Actual TRIGGER Increment Duration

Bit 23:0 **CDT_TX:** Calculated duration of the current *TRIGGER* increment.
calculated value

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

18.12.69 Memory DPLL_CDT_SX

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								CDT_SX																							
Mode	R								RW																							
Initial Value	0x00								0x000000																							

Prediction of the Actual STATE Increment Duration

Bit 23:0 **CDT_SX:** Calculated duration of the current *STATE* increment.
calculated value

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

18.12.70 Memory DPLL_CDT_TX_NOM

Address Offset:	see Appendix B																Initial Value: 0x0000_0000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								CDT_TX_NOM																							
Mode	R								RW																							
Initial Value	0x00								0x000000																							

Prediction of the Nominal TRIGGER Increment Duration

Bit 23:0 **CDT_TX_NOM:** Calculated duration of the current nominal *TRIGGER* event.
calculated value

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

18.12.71 Memory DPLL_CDT_SX_NOM

Address Offset:	see Appendix B																Initial Value: 0x0000_0000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								CDT_SX_NOM																							
Mode	R								RW																							
Initial Value	0x00								0x000000																							

Prediction of the Nominal STATE Increment Duration

Bit 23:0 **CDT_SX_NOM:** Calculated duration of the current nominal *STATE* event.
calculated value

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

18.12.72 Memory DPLL_TLR

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								NOT_USED																TLR							
Mode	R								RW																RW							
Initial Value	0x00								0x0000																0x00							

TRIGGER Locking Range

Bit 7:0 **TLR:** Value is to be multiplied with the last nominal TRIGGER duration in order to get the range for the next TRIGGER event without setting TOR in the DPLL_STATUS register
multiply value with the last nominal increment duration and check violation; when TLR = 0 don't perform the check

Bit 23:8 **Not used**
Note: must be written to zero.

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

18.12.73 Memory DPLL_SLR

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								NOT_USED																SLR							
Mode	R								RW																RW							
Initial Value	0x00								0x0000																0x00							

STATE Locking Range

Bit 7:0 **SLR:** Value is to be multiplied with the last nominal STATE duration in order to get the range for the next STATE event without setting SOR in the DPLL_STATUS register
multiply value with the last nominal increment duration and check violation; when SLR = 0 don't perform the check

Bit 23:8 **Not used**
Note: must be written to zero.

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

18.12.74 Memory DPLL_PDT_[z] (z:0...NOAC-1)

Address Offset:	see Appendix B																Initial Value: 0x0000_0000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								DW										DB													
Mode	R								RW										RW													
Initial Value	0x00								0x000										0x0000													

Projected Increment Sum Relations for Action [z]

Bit 13:0 **DB:** Fractional part of relation between *TRIGGER* and *STATE* increments.

Bit 23:14 **DW:** Integer part of relation between *TRIGGER* and *STATE* increments. Definition of relation values between *TRIGGER* and *STATE* increments PDT[i] according to Equations DPLL-11 or DPLL-13 (z:0...NOAC-1)[1)]

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

[1)] **Note:** The PDT[z] values for actions i=24...31 are not available for all devices. Please refer to appendix B.

18.12.75 Memory DPLL_MLS1

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								NOT_USED						MLS1																	
Mode	R								RW						RW																	
Initial Value	0x00								0x00						0x0000_0																	

Calculated Number of Sub-Pulses between two nominal STATE Events for SMC = 0

Bit 17:0 **MLS1:** number of pulses between two *STATE* events (to be set and updated by the CPU).

For SMC=0 the value of MLS1 is calculated once by the CPU for fixed values in the DPLL_CTRL_0 register by the formula MLS1 = ((MLT+1)*(TNU+1)/(SNU+1)) and set accordingly

FOR SMC=1 the value of MLS1 represents the number of pulses between two nominal *TRIGGER* events (to be set and updated by the CPU)

Bit 23:18 **Not used**

Note: must be written to zero.

Bit 31:24 **Reserved**

Note: Read as zero, should be written as zero.

18.12.76 Memory DPLL_MLS2

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								NOT_USED						MLS2																	
Mode	R								RW						RW																	
Initial Value	0x00								0x00						0x0000_0																	

Calculated Number of Sub-Pulses between two nominal STATE Events for SMC=1 and RMO=1

Bit 17:0 **MLS2:** number of pulses between two *STATE* events (to be set and updated by the CPU).
Using adapt information and the missing *STATE* event information SYN_S, this value can be corrected for each increment automatically.

Bit 23:18 **Not used**
Note: must be written to zero.

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

18.12.77 Memory DPLL_CNT_NUM_1

Address Offset:	**see Appendix B**																**Initial Value: 0x0000_0000**															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								CNT_NUM_1																							
Mode	R								RW																							
Initial Value	0x00								0x0000 00																							

Bit 23:0 **CNT_NUM_1: Counter for number of SUB_INC1 pulses;** Number of pulses in continuous mode for a nominal increment in normal and emergency mode for SUB_INC1, given and updated by CPU only.
count value for continuous mode

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

18.12.78 Memory DPLL_CNT_NUM_2

Address Offset:	see Appendix B	Initial Value: 0x0000_0000
	31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Bit	Reserved	CNT_NUM_2
Mode	R	RW
Initial Value	0x00	0x0000 00

Bit 23:0 **CNT_NUM_2: Counter for number of SUB_INC2 pulses;** Number of pulses in continuous mode for a nominal increment in normal and emergency mode for SUB_INC2, given and updated by CPU only.
count value for continuous mode

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

18.12.79 Memory DPLL_PVT

Address Offset:	see Appendix B	Initial Value: 0x0000_0000
	31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Bit	Reserved	PVT
Mode	R	RW
Initial Value	0x00	0x0000 00

Plausibility Value of Next TRIGGER Slope

Bit 23:0 **PVT:** Plausibility value of next active *TRIGGER* slope.
The meaning of the value depends on the value of the PIT value in the DPLL_CTRL_1 register.
For PIT=0: the number of SUB_INC1 pulses to be waited for until a next active *TRIGGER* event is accepted.
For PIT=1: PVT is to be multiplied with the **last nominal increment time DT_T_ACT** and divided by 1024 and reduced to a 24 bit value in order to get the time to be waited for until the next active *TRIGGER* event is accepted. The wait time must be exceeded for an active slope.

Note: When an active *TRIGGER* slope is detected while the wait condition is not fulfilled the interrupt PWI is generated. Please note, that the SGE1 must be set, when PIT=0 in order to provide the necessary SUB_INC1 pulses for checking. After an unexpected missing TRIGGER the plausibility check is suppressed for the following increment. In case of direction change the PVT value is automatically set to zero in order to deactivate the check.

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

18.12.80 Memory DPLL_PSTC

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_0000**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								PSTC																							
Mode	R								RW																							
Initial Value	0x00								0x0000 00																							

Actual Calculated Position Stamp of TRIGGER

Bit 23:0 **PSTC:** calculated position stamp of last *TRIGGER* input;
value is set by the DPLL and can be updated by the CPU when filter values are to be considered for the exact position
(see **DPLL_STATUS** and **DPLL_CTRL** registers for explanation of the status and control bits used). For each active slope of *TRIGGER* in normal mode

when FTD=0: PSTC is set from actual position value, for the first active *TRIGGER* event (no filter delay considered) the CPU must update the value once, taking into account the filter value

when FTD=1: PSTC is incremented at each *TRIGGER* event by
SMC=0: ((MLT+1) +PD) * SYN_T; while PD=0 for AMT=0
SMC=1: (MLS1 + PD)*(SYN_T); while PD=0 for AMT=0

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

18.12.81 Memory DPLL_PSSC

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								PSSC																							
Mode	R								RW																							
Initial Value	0x00								0x000000																							

Actual Calculated Position Stamp of STATE

Bit 23:0 **PSSC:** calculated position stamp for the last *STATE* input;
first value is set by the DPLL and can be updated by the CPU when the filter delay is to be considered. For each active slope of *STATE* in emergency mode

when FSD=0: PSSC is set from actual position value(no filter delay considered), the CPU must update the value once, taking into account the filter value

when FSD=1: at each active slope of *STATE* (PD_S_store=0 for AMS=0):
SMC=0: add (MLS1 + PD_S_store)*(SYN_S);
SMC=1: add (MLS2 + PD_S_store)*(SYN_S);

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

18.12.82 Memory DPLL_PSTM

Address Offset:	see Appendix B																Initial Value: 0x0000_0000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								PSTM																							
Mode	R								RW																							
Initial Value	0x00								0x000000																							

Measured Position Stamp at Last TRIGGER Input

Bit 23:0 **PSTM: Position stamp of TRIGGER, measured;** Measured position stamp of last active *TRIGGER* input.

Store the value TBU_TS1 when an active TRIGGER event occurs. The value of PSTM is invalid for (RMO=1 and SMC=0).

Bit 31:24 **Reserved**

Note: Read as zero, should be written as zero.

Note: The LSB address is determined using the SWON_T value in the OSW register (see chapter 18.12.8).

18.12.83 Memory DPLL_PSTM_OLD

Address Offset:	see Appendix B																Initial Value: 0x0000_0000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								PSTM_OLD																							
Mode	R								RW																							
Initial Value	0x00								0x000000																							

Measured Position Stamp at Last but one TRIGGER Input

Bit 23:0 **PSTM_OLD: Last but one position stamp of TRIGGER, measured;** Measured position stamp of last but one active *TRIGGER* input.

last PSTM value: see explanation of PSTM

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

Note: The LSB address is determined using the SWON_T value in the OSW register (see chapter 18.12.8).

18.12.84 Memory DPLL_PSSM

Address Offset:	**see Appendix B**																**Initial Value: 0x0000_0000**															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								PSSM																							
Mode	R								RW																							
Initial Value	0x00								0x0000 00																							

Measured Position Stamp at Last STATE Input

Bit 23:0 **PSSM: Position stamp of STATE, measured;** Measured position stamp of last active *STATE* input.
Store the value TBU_TS1 or TBU_TS2 respectively at the moment when an active STATE event occurs. The value of PSSM is invalid for (RMO=0 and SMC=0).

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

Note: The LSB address is determined using the SWON_S value in the OSW register (see chapter 18.12.8).

18.12.85 Memory DPLL_PSSM_OLD

Address Offset:	see Appendix B																Initial Value: 0x0000_0000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								PSSM_OLD																							
Mode	R								RW																							
Initial Value	0x00								0x000000																							

Measured Position Stamp at Last but one STATE Input

Bit 23:0 **PSSM_OLD: Last but one position stamp of STATE, measured;** Measured position stamp of last but one active *STATE* input.
last PSSM value: see explanation of PSSM

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

Note: The LSB address is determined using the SWON_S value in the OSW register (see chapter 18.12.8).

18.12.86 Memory DPLL_NMB_T

Address Offset:	see Appendix B																Initial Value: 0x0000_0000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								NOT_USED								NMB_T															
Mode	R								RW								RW															
Initial Value	0x00								0x00								0x0000															

Number of Pulses to be sent in Normal Mode

Bit 15:0 **NMB_T: Number of pulses for TRIGGER;** Calculated number of pulses in normal mode for the current *TRIGGER* increment.
calculated pulse number

Bit 23:16 **Not used**
Note: must be written to zero.

Bit 31:24 **Reserved**

Note: Read as zero, should be written as zero.

18.12.87 Memory DPLL_NMB_S

Address Offset:	see Appendix B																Initial Value: 0x0000_0000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								NOT_USED				NMB_S																			
Mode	R								RW				RW																			
Initial Value	0x00								0x0				0x00000																			

Number of Pulses to be sent in Emergency Mode

Bit 19:0 **NMB_S: Number of pulses for STATE;** Calculated number of pulses in emergency mode for the current *STATE* increment.
calculated pulse number

Bit 23:20 **Not used**
Note: must be written to zero.

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

18.12.88 Memory DPLL_RDT_S[i]

Address Offset:	see Appendix B																Initial Value: 0x0000_0000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								RDT_S																							
Mode	R								RW																							
Initial Value	0x00								0x000000																							

Reciprocal Values of the Nominal STATE Increment Duration in FULL_SCALE

Bit 23:0 **RDT_S: Reciprocal difference time of STATE;** nominal reciprocal value of the number of time stamp clocks measured in the corresponding increment *2^{32} while only the lower 24 bits are used; no gap considered. The LSB is rounded up when the next truncated bit is 1.

Note: There are 2*(SNU+1-SYN_NS) entries for SYSF=0 or 2*(SNU+1)-SYN_NS entries for SYSF=1 respectively.

Bit 31:24 **Reserved**

Note: Read as zero, should be written as zero.

Note: If DPLL_CTRL_11.STATE_EXT is set, this memory range is not used by the DPLL, but emulated outside the DPLL. The DPLL will access the MCS to DPLL interface 18.15 and will expect data to be correctly stored there. This means in fact, that the handling of the RDT_S values has to be done outside, in the MCS integrated in the same cluster.

18.12.89 Memory DPLL_TSF_S[i]

Address Offset:	**see Appendix B**																**Initial Value: 0x0000_0000**															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								TSF_S																							
Mode	R								RW																							
Initial Value	0x00								0x0000 00																							

Time Stamp Values of the Nominal STATE Events in FULL_SCALE

Bit 23:0 **TSF_S: Time stamp field of STATE;** Time stamp value of each active *STATE* event.

Note: There are 2* (SNU+1) entries.

Bit 31:24 **Reserved**

Note: Read as zero, should be written as zero.

Note: If DPLL_CTRL_11.STATE_EXT is set, this memory range is not used by the DPLL, but emulated outside the DPLL. The DPLL will access the MCS to DPLL interface 18.15 and will expect data to be correctly stored there. This means in fact, that the handling of the TSF_S values has to be done outside, in the MCS integrated in the same cluster.

18.12.90 Memory DPLL_ADT_S[i]

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_0000**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								NOT_USED		NS						PD_S															
Mode	R								RW		RW						RW															
Initial Value	0x00								0x0		0x00																					

Adapt and Profile Values of the STATE Increments in FULL_SCALE

Bit 15:0 **PD_S: Physical deviation of STATE;** Adapt values for each *nominal STATE* increment in FULL_SCALE (sint16);

This value represents the number of pulses to be added to the correspondent nominal increment. The absolute value of a negative PD_S must not exceed MLS1 or MLS2 respectively. The PD value does mean the number of SUB_INC1 pulses per nominal tooth to be added to NS*((MLS1/2+1) + PD_S);

Bit 21:16 **NS: Number of STATEs;** number of nominal *STATE* parts in the corresponding increment.

Note: There are 2*(SNU+1-SYN_NS) entries for SYSF=0 or 2*(SNU+1)-SYN_NS entries for SYSF=1 respectively.

Bit 23:22 **Not used**

Note: must be written to zero.

Bit 31:24 **Reserved**

Note: Read as zero, should be written as zero.

Note: If DPLL_CTRL_11.STATE_EXT is set, this memory range is not used by the DPLL, but emulated outside the DPLL. The DPLL will access the MCS to DPLL interface 18.15 and will expect data to be correctly stored there. This means in fact, that the handling of the ADT_S values has to be done outside, in the MCS integrated in the same cluster.

18.12.91 Memory DPLL_DT_S[i]

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								DT_S																							
Mode	R								RW																							
Initial Value	0x00								0x000000																							

Nominal STATE Increment Duration in FULL_SCALE

Bit 23:0 **DT_S: Difference time of STATE;** nominal increment duration values for each *STATE* increment in FULL_SCALE (considering no gap).

Note: There are 2*(SNU+1-SYN_NS) entries for SYSF=0 or 2*(SNU+1)-SYN_NS entries for SYSF=1 respectively.

Bit 31:24 **Reserved**

Note: Read as zero, should be written as zero.

Note: If DPLL_CTRL_11.STATE_EXT is set, this memory range is not used by the DPLL, but emulated outside the DPLL. The DPLL will access the MCS to DPLL interface 18.15 and will expect data to be correctly stored there. This means in fact, that the handling of the DT_S values has to be done outside, in the MCS integrated in the same cluster.

18.12.92 Register DPLL_TSAC[z] z:(0...NOAC-1)

Address Offset:	see Appendix B																Initial Value:								0x007F_FFFF							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								TSAC																							
Mode	R								RPw																							
Initial Value	0x00								0x7FFFFF																							

Calculated Time Value to start Action z

Bit 23:0 **TSAC** calculated time stamp for ACTION z (z = 0...NOAC-1)[1)]

Note: This value can only be written when the DPLL is disabled.

Bit 31:24 **Reserved**

Note: Read as zero, should be written as zero.

18.12.93 Register DPLL_PSAC[z] z:(0…NOAC-1)

Address Offset:	see Appendix B																Initial Value:								0x007F_FFFF							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								PSAC																							
Mode	R								RPw																							
Initial Value	0x00								0x7FFFFF																							

Calculated Position Value to start Action z

Bit 23:0 **PSAC:** Calculated position value for the start of ACTION z in normal or emergency mode according to equations DPLL-17 or DPLL-20 respectively (z = 0…NOAC-1).

Note: This value can only be written when the DPLL is disabled.

Bit 31:24 **Reserved**

Note: Read as zero, should be written as zero.

18.12.94 Register DPLL_ACB_[z]

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved			ACB_3					Reserved			ACB_2					Reserved			ACB_1					Reserved			ACB_0				
Mode	R			RPw					R			RPw					R			RPw					R			RPw				
Initial Value	0x0			00000					0			00000					0			00000					0			00000				

Control Bits for up to 32 Actions

Bit 4:0 **ACB_0:** Action Control Bits of ACTION_i, reflects ACT_D[i](52:48), i=4*z

Bit 7:5 **Reserved**

Note: Read as zero, should be written as zero.

Bit 12:8 **ACB_1:** Action Control Bits of ACTION_(i + 1) , reflects ACT_D[i+1](52:48), i=4*z

Note:

When DPLL_CTRL_11.ACBU = '0': ACB_1[4:0] are taken as received by ARU interface and are transmitted unchanged as result of action (PMT) calculation.

When DPLL_CTRL_11.ACBU = '1': ACB_1[4:2] are taken as received by ARU interface and are transmitted unchanged as result of action (PMT) calculation.

ACB_1[1]= '1' is used as input signal to control if "action in past" shall be checked based on position information. ACB_1[1] is written to '1' if action channel has reached "action in past" condition after action has been calculated, written to '0' if action has not reached "past" so far.

ACB_1[0] is used as input signal to control if "action in past" shall be checked based on time information. ACB_1[0] is written to '1' if action channel has reached "action in past" condition after action has been calculated, written to '0' if action has not reached "past" so far.

Note: This value can only be written via AEI-interface when the DPLL is disabled.

Bit 15:13 **Reserved**

Note: Read as zero, should be written as zero.

Bit 20:16 **ACB_2:** Action Control Bits of ACTION_(i + 2), reflects ACT_D[i+2](52:48), i=4*z

Note:

When DPLL_CTRL_11.ACBU = '0': ACB_2[4:0] are taken as received by ARU interface and are transmitted unchanged as result of action (PMT) calculation.

When DPLL_CTRL_11.ACBU = '1': ACB_2[4:2] are taken as received by ARU interface and are transmitted unchanged as result of action (PMT) calculation.

ACB_2[1]= '1' is used as input signal to control if "action in past" shall be checked based on position information. ACB_2[1] is written to '1' if action channel has reached "action in past" condition after action has been calculated, written to '0' if action has not reached "past" so far.

ACB_2[0] is used as input signal to control if "action in past" shall be checked based on time information. ACB_2[0] is written to '1' if action channel has reached "action in past" condition after action has been calculated, written to '0' if action has not reached "past" so far.

Note: This value can only be written via AEI-interface when the DPLL is disabled.

Bit 23:21 **Reserved**
Note: Read as zero, should be written as zero.

Bit 28:24 **ACB_3:** Action Control Bits of ACTION_(i + 3), reflects ACT_D[i+3](52:48), i=4*z

Note:

When DPLL_CTRL_11.ACBU = '0': ACB_3[4:0] are taken as received by ARU interface and are transmitted unchanged as result of action (PMT) calculation.

When DPLL_CTRL_11.ACBU = '1': ACB_3[4:2] are taken as received by ARU interface and are transmitted unchanged as result of action (PMT) calculation.
ACB_3[1]= '1' is used as input signal to control if "action in past" shall be checked based on position information. ACB_3[1] is written to '1' if action channel has reached "action in past" condition after action has been calculated, written to '0' if action has not reached "past" so far.
ACB_3[0] is used as input signal to control if "action in past" shall be checked based on time information. ACB_3[0] is written to '1' if action channel has reached "action in past" condition after action has been calculated, written to '0' if action has not reached "past" so far.

Note: This value can only be written via AEI-interface when the DPLL is disabled.

Bit 31:29 **Reserved**
Note: Read as zero, should be written as zero.

Note:

When DPLL_CTRL_11.ACBU = '0': ACB_0[4:0] are taken as received by ARU interface and are transmitted unchanged as result of action (PMT) calculation.

When DPLL_CTRL_11.ACBU = '1': ACB_0[4:2] are taken as received by ARU interface and are transmitted unchanged as result of action (PMT) calculation.
ACB_0[1]= '1' is used as input signal to control if "action in past" shall be checked based on position information. ACB_0[1] is written to '1' if action channel has reached "action in past" condition after action has been calculated, written to '0' if action has not reached "past" so far.
ACB_0[0] is used as input signal to control if "action in past" shall be checked based on time information. ACB_0[0] is written to '1' if action channel has reached "action in past" condition after action has been calculated, written to '0' if action has not reached "past" so far.

Note: This value can only be written via AEI-interface when the DPLL is disabled.

18.12.95 Register DPLL_CTRL_11

Address Offset:	see Appendix B															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Bit	WACBU	WSTATE_EXT	WPCMF2_INCCNT	WINCF2	WFSYL2	WPCMF2	WERZ2	WSIP2	WADS	WADT	WPCMF1_INCCNT	WINCF1	WFSYL1	WPCMF1	WERZ1	WSIP1
Mode	RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw
Initial Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Initial Value:	0x0000_0000															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	ACBU	STATE_EXT	PCMF2_INCCNT_	INCF2	FSYL2	PCMF2	ERZ2	SIP2	ADS	ADT	PCMF1_INCCNT_	INCF1	FSYL1	PCMF1	ERZ1	SIP1
Mode	RPw	RPw	RPw	RPw	RAw	RPw	RPw	RPw	RPw	RPw	RPw	RPw	RAw	RPw	RPw	RPw
Initial Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Control Register 11

Bit 0 **SIP1: simplified increment prediction** in normal mode and for the first engine in the case SMC=1.

0 = **Increment prediction calculation**; the current increment duration CDT_TX is calculated using the relation between increment duration in the past like explained by the corresponding equations.

1 = **Increment prediction continuation**; in this mode for the increment prediction value calculation of CDT_TX the value of QDT_T is replaced by 1 for all calculations when **NUTE-VTN=1**; in the other case the value of SIP1 is ignored and the calculation is performed like for SIP1=0.

For the first increment after setting SIP1 from 0 to 1 the value of DT_T_ACT is replaced by the value of the DT_T_START register. This results in a CDT_TX value which is equal to DT_T_START. Please notice that this DT_T_Start value must be always > 256.

Note: The value of SIP1 influences only the increment prediction CDT_TX and when NUTE-VTN=1. The calculation of QDT_T itself is not influenced by the SIP1 bit. The value of SIP1 can be only written when WSIP1=1.

When SIP1= 1 is set the first pulses of the subincrement genrator are not generated with highest frequency for the first increment (DPLL_STATUS.ftd = 0, DPLL_CTRL_1.SGE1 = 1).

Note: When SIP1= 1 is set the first pulses of the subincrement generator are not generated with highest frequency for the first increment (DPLL_STATUS.ftd = 0, DPLL_CTRL_1.SGE1 = 1).

Bit 1 **ERZ1: Error is assumed as zero** in normal mode and for the first engine for SMC=1.

0 = The MEDT_T value is considered as provided in the corresponding equations.

1 = Instead of using MEDT_T the value "0" is used in the corresponding equations.

Note: The calculation of EDT_T and MEDT_T is performed independently from the ERZ1 value in all modes without any influence to the MEDT_T value itself. The ERZ1 value influences the use of MEDT_T in normal mode and for SMC=1. The value of ERZ1 can be only written when WERZ1=1.

Bit 2 **PCMF1: Pulse correction mode fast** for INC_CNT1

0 = No fast update of pulses, provided by MPVAL1.

1 = When PCM1 is set while PCMF1=1, the pulses provided by MPVAL1 are sent using the rapid pulse generator RPCUx without waiting for a new input event.

Note: The fast pulse generation is performed immediately within the current increment.

MPVAL1 must be positive integers for the fast pulse correction mode - in the case of negative values the correction is suppressed and the FPCE (fast pulse correction error) bit in the DPLL_STATUS register is set, causing the EI (error interrupt) when enabled.

The setting of PCMF1 prevents the transfer of control bits PCM1 to the corresponding shadow registers with an active input event and prevents therefore the distribution of the MPVAL1 values over the current or next increment. The MPVAL1 pulses are sent with the fast clock CMU_CLK0 by the rapid pulse generator RPCUx (see chapter 18.8.3.6 of specification v3.0) triggered in the state 6/26 or 18/38 of the state machines (see chapters 18.8.6.1 and 18.8.6.7 of specification v2.1.0) respectively. The INC_CNT1 is incremented by MPVAL1 respectively.

When taken the MPVAL1 value to RPCUx and INC_CNT1 the PCM1 bit is reset immediately and after that also the PCMF1 bit. The value of PCMF1 can be only written when WPCMF1=1.

Be careful when using the fast pulse correction during a direction change. Because of sending the correction pulses before, during or after the direction change recognition the result is typically unpredictable. No automatic correction of the fast correction pulses is provided. The necessary corrections must be performed on responsibility of the user.

Bit 3 **FSYL1: Force Synchronization Loss of LOCK1.**

0 = no force of synchronization loss.

1 = Reset LOCK1, and reset SYT (in normal mode and for SMC=1) or reset SYS (in emergency mode)

Note: The synchronization loss resets SYT/SYS and prevents the use of profiles respectively. The above described effect for FSYL1=1 is only active when WFSYL1=1 simultaneously.

Bit 4 **INCF1: INC_CNT1 fast correction**

0 = the calculation of a new INC_CNT1 is performed after an active slope was detected and the plausibility check was performed.

1 = the calculation of a new INC_CNT1 is prepared before an active slope is detected; the plausibility check is supported by an additional HW checker in order to get the decision earlier and after this decision the pulse generator for SUB_INC1 starts immediately sending out pulses

The calculation of ADD_IN for the SUB_INC generation is performed without adding the 0.5 value to NMB_T/S in equations DPLL_25 ff.

The Signal RESET_SIGx of the pulse generator (see chapter 18.8.3.6) is activated for each new active input slope in order to reset the register values.

Note: The INCF1 value can be only written when WINCF1=1.

The INCF1 bit should only be written when DPLL_CTRL_1.DEN = '0' (DPLL disabled) to prevent generation of wrong number of sub increments.

Bit 5 **PCMF1_INCCNT_B: no increment of INC_CNT1 when PCMF1 active** (automatic end mode).

0 = Add MPVAL1 value is as well to the INC_CNT1 when fast pulse correction is done by PCM1 or PCMF1.

1 = Do not add MPVAL1 value to the INC_CNT1 register when fast pulse correction is done by PCM1 or PCMF1. This means that just fast pulses are done by decrementing current content of INC_CNT1 register as long as INC_CNT1 is not zero (automatic end mode). The number of pulses (MPVAL1) shall be sufficiently smaller than INC_CNT1 when MPVAL1 is written.

Note: The PCMF1_INCCNT_B value can be only written when WPCMF1_INCCNT_B =1.

Bit 6 **ADT: correction of DT_T_ACTUAL, CDT_TX_nom_corr by PD_T**

0 = No correction of DT_T_ACTUAL, CDT_TX_nom_corr by physical deviation (PD_T) defined in profile of TRIGGER processing unit.

1 = Correction of DT_T_ACTUAL, CDT_TX_nom_corr by physical deviation (PD_T) defined in profile of TRIGGER processing unit.

Bit 7 **ADS: correction of DT_S_ACTUAL, CDT_SX_nom_corr by PD_S**

0 = No correction of DT_S_ACTUAL, CDT_SX_nom_corr by physical deviation (PD_S) defined in profile of STATE processing unit.

1 = Correction of DT_S_ACTUAL, CDT_SX_nom_corr by physical deviation (PD_S) defined in profile of STATE processing unit.

Bit 8 **SIP2: simplified increment prediction** in emergency mode and for the second engine in the case RMO=1.

0 = **Increment prediction calculation**; the current increment duration CDT_SX is calculated using the relation between increments duration in the past like explained by the corresponding equations.

1 = **Increment prediction continuation**; in this mode for the increment prediction value calculation CDT_SX the value of QDT_S is replaced by 1 for all calculations when **NUSE-VSN=1**; in the other case the value of SIP2 is ignored and the calculation is performed like for SIP2=0.

For the first increment after setting SIP2 from 0 to 1 the value of DT_S_ACT is replaced by the value of the DT_S_START register. This results in a CDT_SX value which is equal to DT_S_START. Please notice that this DT_S_START value must be always > 256.

Note: The value of SIP2 influences only the increment prediction and error accumulation when NUSE-VSN=1. The calculation of QDT_S itself is not influenced by the SIP2 bit. The value of SIP2 can be only written when WSIP2=1.

Bit 9 **ERZ2: Error is assumed as zero** in emergency mode and for the second engine for SMC=1.

0 = The MEDT_S value is considered as provided in the corresponding equations.

1 = Instead of using MEDT_S the value "0" is used in the corresponding equations.

Note: The calculation of EDT_S and MEDT_S is performed independently from the ERZ2 value in all modes without any influence to the MEDT_S value itself. The ERZ2 value influences the use of MEDT_S in emergency mode and for SMC=1 with RMO=1. The value of ERZ2 can be only written when WERZ2=1.

Bit 10 **PCMF2: Pulse correction mode fast for INC_CNT2**

0 = No fast update of pulses, provided by MPVAL2.

1 = When PCM2 is set while PCMF2=1, the pulses provided by MPVAL2 are sent using the rapid pulse generator RPCUx without waiting for a new input event.

Note: The fast pulse generation is performed immediately within the current increment.

MPVAL2 must be positive integers for the fast pulse correction mode - in the case of negative values the correction is suppressed and the FPCE (fast pulse correction error) bit in the DPLL_STATUS register is set, causing the EI (error interrupt) when enabled.

The setting of PCMF2 prevents the transfer of control bits PCM2 to the corresponding shadow registers with an active input event and prevents therefore the distribution of the MPVAL1 values over the current or next increment. The MPVAL2 pulses are sent with the fast clock CMU_CLK0 by the rapid pulse generator RPCUx (see chapter 18.8.3.6 of specification v3.0) triggered in the state 6/26 or 18/38 of the state machines (see chapters 18.8.6.1 and 18.8.6.7 of specification v2.1.0) respectively. The INC_CNT2 is incremented by MPVAL2 respectively.

When taken the MPVAL2 value to RPCUx and INC_CNT2 the PCM2 bit is reset immediately and after that also the PCMF2 bit.

The value of PCMF2 can be only written when WPCMF2=1.

Be careful when using the fast pulse correction during a direction change. Because of sending the correction pulses before, during or after the direction change recognition the result is typically unpredictable. No automatic correction of the fast correction pulses is provided. The necessary corrections must be performed on responsibility of the user.

Bit 11 **FSYL2: Force Synchronization Loss of LOCK2**.

0 = no force of synchronization loss.

1 = Reset LOCK2, and reset SYS (in emergency mode and for SMC=1).

Note: The synchronization loss resets SYS and prevents the use of profiles respectively. The above described effect for FSYL2=1 is only active when WFSYL2=1 simultaneously

Bit 12 **INCF: INC_CNT2 fast**

0 = the calculation of a new INC_CNT2 is performed after an active slope was detected and the plausibility check was performed.

1 = the calculation of a new INC_CNT2 is prepared before an active slope is detected; the plausibility check is supported by an additional HW checker in order to get the decision earlier and after this decision the pulse generator for SUB_INC2 starts immediately sending out

pulses. The calculation of ADD_IN for the SUB_INC generation is performed without adding the 0.5 value to NMB_S in equations DPLL_25 ff. The Signal RESET_SIGx of the pulse generator (see chapter 18.8.3.6) is activated for each new active input slope in order to reset the register values.

Note: The INCF2 value can be only written when WINCF2=1.

Bit 13 **PCMF2_INCCNT_B: no increment of INC_CNT2 when PCMF2 active** (automatic end mode).

0 = Add MPVAL2 value is as well added to the INC_CNT2 when fast pulse correction is done by PCM2 or PCMF2.

1 = Do not add MPVAL2 value to the INC_CNT2 register when fast pulse correction is done by PCM2 or PCMF2. This means that just fast pulses are done by decrementing current content of INC_CNT2 register as long as INC_CNT2 is not zero (automatic end mode). The number of pulses (MPVAL2) shall be sufficiently smaller than INC_CNT2 when MPVAL2 is written.

Note: The PCMF2_INCCNT_B value can be only written when WPCMF2_INCCNT_B=1.

Bit 14 **STATE_EXT:** Use of STATE engine extension

0 = STATE extension is not considered.

1 = STATE extension is enabled up to 128 STATE events

Note: The STATE_EXT value can be only written when WSTATE_EXT= 1 and the DPLL is disabled. See 18.10 for a further explanation. If this bit shall be modified during operation a software reset of the DPLL module is strongly recommended. A RAM initialisation should also be considered depending on the given application case.

Bit 15 **ACBU: ACB use;** the ACB values of PMTR are used to decide if an action is in the past

0 = ACB values of PMTR are not considered in DPLL; the decision if an action is in the past is made considering the calculated time value.

1 = ACB values of PMTR are considered in DPLL as follows:

if ACB[1] = 1, consider if the calculated position value of the corresponding action is in the past.

if ACB[0] = 1, consider if the calculated time value of the corresponding action is in the past.

ACB[1] and ACB[0] can be set also simultaneously to 1

Note: Return ACB values together with actions as zero, when the actions are in future;

set ACB[1]=1, when calculated position value is in the past and the ACB[1] of PMTR was 1.
Set ACB[0]=1, when calculated time value is in the past and the ACB[0] of PMTR was 1.
The value of ACBU can be only written when WACBU=1.

Bit 16 **WSIP1: write enable for simplified increment prediction 1**.
0 = Writing to SIP1 is not enabled.
1 = Writing to SIP1 is enabled.
Note: Enable writing.

Bit 17 **WERZ1: write enable for error zero 1.**
0 = Writing to ERZ1 is not enabled.
1 = Writing to ERZ1 is enabled.

Note: Enable writing.

Bit 18 **WPCMF1: write enable for pulse correction mode fast 1**
0 = Writing to PCM1 is not enabled.
1 = Writing to PCM1 is enabled.
Note: Enable writing.

Bit 19 **WFSYL1: write enable for Force Synchronization Loss 1**.
0 = Writing to FSYL1 is not enabled.
1 = Writing to FSYL1 is enabled.
Note: Enable writing

Bit 20 **WINCF: write enable for INC_CNT1 fast**
0 = Writing to INCF1 is not enabled.
1 = Writing to INCF1 is enabled.
Note: Enable writing.

Bit 21 **WPCMF1_INCCNT_B: write enable of PCMF1_INCCNT_B**
0 = Writing to PCMF1_INCCNT_B is not enabled.
1 = Writing to PCMF1_INCCNT_B is enabled.

Bit 22 **WADT: write enable of ADT**
0 = Writing to ADT is not enabled.
1 = Writing to ADT is enabled.

Bit 23 **WADS: write enable of ADS**
0 = Writing to ADS is not enabled.
1 = Writing to ADS is enabled.

Bit 24 **WSIP2: write enable for simplified increment prediction 2**.
0 = Writing to SIP2 is not enabled.
1 = Writing to SIP2 is enabled.
Note: Enable writing.

Bit 25 **WERZ2: write enable for error zero 2.**
0 = Writing to ERZ2 is not enabled
1 = Writing to ERZ2 is enabled.
Note: Enable writing.

Bit 26 **WPCMF2: write enable for pulse correction mode fast 2**
0 = Writing to PCMF2 is not enabled
1 = Writing to PCMF2 is enabled
Note: Enable writing.

Bit 27 **WFSYL2: write enable for Force Synchronization Loss 2**.
0 = Writing to FSYL2 is not enabled.
1 = Writing to FSYL2 is enabled

Note: Enable writing

Bit 28 **WINCF: write enable for INC_CNT2 fast**
0 = Writing to INCF2 is not enabled
1 = Writing to INCF2 is enabled.

Note: Enable writing.

Bit 29 **WPCMF2_INCCNT_B: write enable of PCMF2_INCCNT_B**
0 = Writing to PCMF2_INCCNT_B is not enabled.
1 = Writing to PCMF2_INCCNT_B is enabled.

Bit 30 **WSTATE_EXT: write enable of STATE_EXT**
0 = Writing to STATE_EXT is not enabled.
1 = Writing to STATE_EXT is enabled.

Bit 31 **WACBU: write enable for ACB use;** the ACB values of PMTR are used to decide if an action is in the past
0 = Writing to ACBU is not enabled.
1 = Writing to ACBU is enabled.
Note: Enable writing.

18.12.96 Register DPLL_THVAL2

Address Offset:	see Appendix B																Initial Value: 0x0000_0000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								THVAL																							
Mode	R								R																							
Initial Value	0x00								0																							

Bit 23:0 **THVAL:** measured last pulse time from active to inactive slope of TRIGGER after correction of input slope filter delays

Note: This value is available immediately after the inactive slope of TRIGGER. The measured value considers all input slope filter delays. From the received input the corresponding filter delays are subtracted before the time stamp difference of active and inactive slope is calculated.

Bit 31:24 **Reserved**

Note: Read as zero, should be written as zero.

18.12.97 Register DPLL_TIDEL

Address Offset:	see Appendix B																Initial Value: 0x0000_0000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								TIDEL																							
Mode	R								RW																							
Initial Value	0x00								0x00_0000																							

Bit 23:0 **TIDEL:** TRIGGER input delay

Transmit this value with each active TRIGGER slope into a shadow register. Subtract this shadow register value from each TRIGGER time stamp (active and inactive slope). This feature is always active and cannot be disabled by a control bit.

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

18.12.98 Register DPLL_SIDEL

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_0000**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								SIDEL																							
Mode	R								RW																							
Initial Value	0x00								0x00_0000																							

Bit 23:0 **SIDEL:** STATE input delay
Transmit this value with each active STATE slope into a shadow register. Subtract this shadow register value from each STATE time stamp (active and inactive slope). This feature is always active and cannot be disabled by a control bit.

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

18.12.99 Register DPLL_CTN_MIN

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_0000**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								CTN_MIN																							
Mode	R								RW																							
Initial Value	0x00								0																							

Bit 23:0 **CTN_MIN:** CDT_T_NOM min value
Use this register value as CDT_TX_NOM value when the calculated value for the nominal increment prediction of TRIGGER is less than the register value

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

18.12.100 Register DPLL_CTN_MAX

Address Offset:	**see Appendix B**																**Initial Value:**								**0x00FF_FFFF**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								CTN_MAX																							
Mode	R								RW																							
Initial Value	0x00								0xFF_FFFF																							

Bit 23:0 **CTN_MAX:** CDT_T_NOM max value
Use this register value as CDT_TX_NOM value when the calculated value for the nominal increment prediction of TRIGGER is greater than the register value

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

18.12.101 Register DPLL_CSN_MIN

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								CSN_MIN																							
Mode	R								RW																							
Initial Value	0x00								0																							

Bit 23:0 **CSN_MIN:** CDT_S_NOM min value

Use this register value as CDT_SX_NOM value when the calculated value for the nominal increment prediction of STATE is less than the register value

Bit 31:24 **Reserved**

Note: Read as zero, should be written as zero.

18.12.102 Register DPLL_CSN_MAX

Address Offset:	see Appendix B																Initial Value:								0x00FF_FFFF							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								CSN_MAX																							
Mode	R								RW																							
Initial Value	0x00								0xFF_FFFF																							

Bit 23:0 **CTN_MAX:** CDT_S_NOM max value
Use this register value as CDT_SX_NOM value when the calculated value for the nominal increment prediction of STATE is greater than the register value

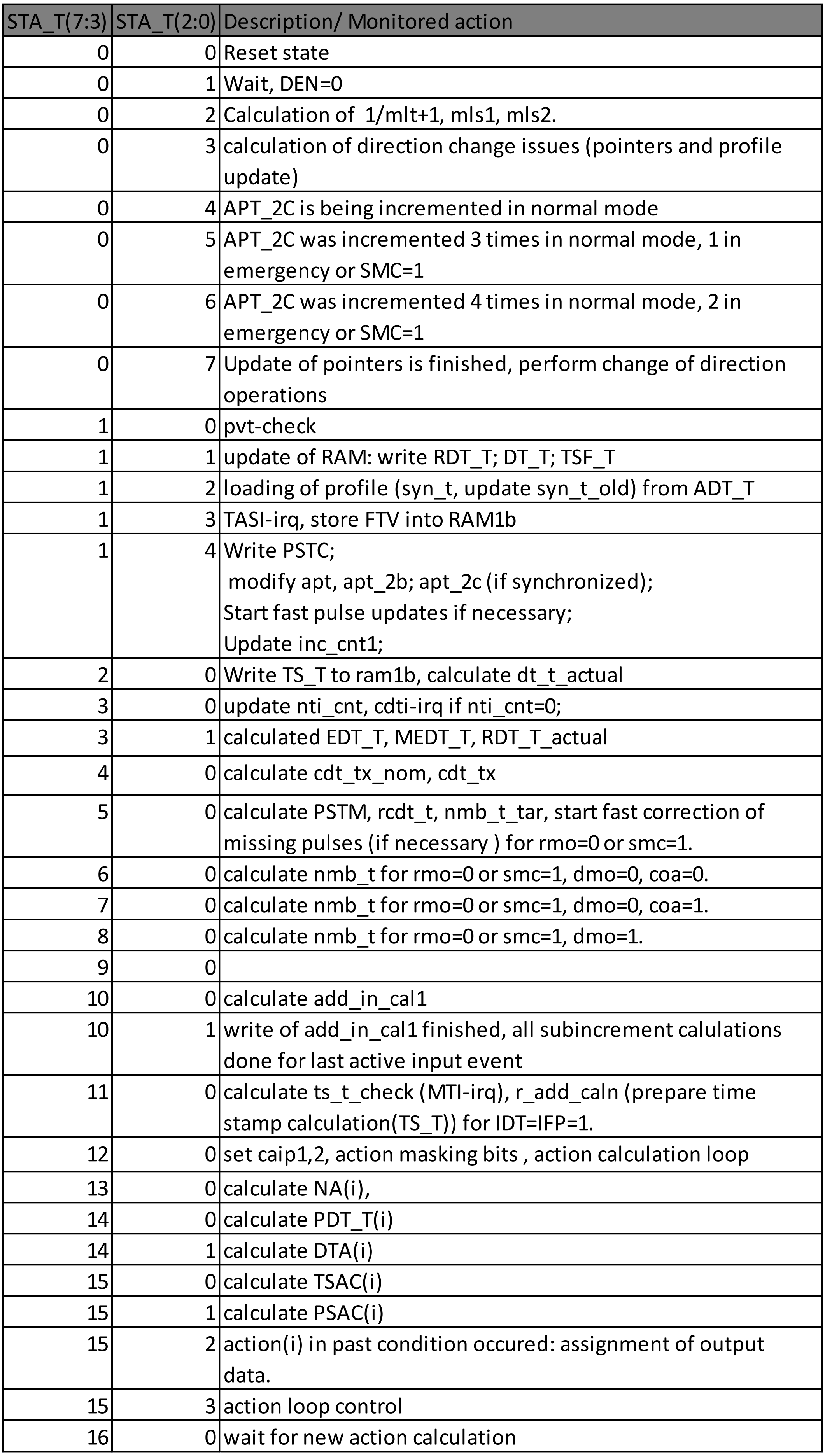
Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

18.12.103 Register DPLL_STA

Address Offset:	see Appendix B																Initial Value:				0x0000_0000											
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								CNT_S			Reserved	STA_S								CNT_T			Reserved	STA_T							
Mode	R								R			R	R								R			R	R							
Initial Value	0x00								0x0			0	0x000								0x0			0	0x000							

Status of the state machine states

Bit 7:0 **STA_T: Status of TRIGGER state machine;** state binary coded
This bit field reflects the status of the TRIGGER state machine
Note: The decimal step number 1 to 20 of the state machine is binary coded from 0x01 to 0x14 respectively using the upper 5 bits (7:3). The lower 3 bit (2:0) show substates of the corresponding state machine. When the DPLL is disabled this field is 0x000.



STA_T(7:3)	STA_T(2:0)	Description/ Monitored action
0	0	Reset state
0	1	Wait, DEN=0
0	2	Calculation of 1/mlt+1, mls1, mls2.
0	3	calculation of direction change issues (pointers and profile update)
0	4	APT_2C is being incremented in normal mode
0	5	APT_2C was incremented 3 times in normal mode, 1 in emergency or SMC=1
0	6	APT_2C was incremented 4 times in normal mode, 2 in emergency or SMC=1
0	7	Update of pointers is finished, perform change of direction operations
1	0	pvt-check
1	1	update of RAM: write RDT_T; DT_T; TSF_T
1	2	loading of profile (syn_t, update syn_t_old) from ADT_T
1	3	TASI-irq, store FTV into RAM1b

1	4	Write PSTC;
		modify apt, apt_2b; apt_2c (if synchronized);
		Start fast pulse updates if necessary;
		Update inc_cnt1;

2	0	Write TS_T to ram1b, calculate dt_t_actual
3	0	update nti_cnt, cdti-irq if nti_cnt=0;
3	1	calculated EDT_T, MEDT_T, RDT_T_actual
4	0	calculate cdt_tx_nom, cdt_tx
5	0	calculate PSTM, rcdt_t, nmb_t_tar, start fast correction of missing pulses (if necessary) for rmo=0 or smc=1.
6	0	calculate nmb_t for rmo=0 or smc=1, dmo=0, coa=0.
7	0	calculate nmb_t for rmo=0 or smc=1, dmo=0, coa=1.
8	0	calculate nmb_t for rmo=0 or smc=1, dmo=1.
9	0	
10	0	calculate add_in_cal1
10	1	write of add_in_cal1 finished, all subincrement calulations done for last active input event
11	0	calculate ts_t_check (MTI-irq), r_add_caln (prepare time stamp calculation(TS_T)) for IDT=IFP=1.
12	0	set caip1,2, action masking bits , action calculation loop
13	0	calculate NA(i),
14	0	calculate PDT_T(i)
14	1	calculate DTA(i)
15	0	calculate TSAC(i)
15	1	calculate PSAC(i)
15	2	action(i) in past condition occured: assignment of output data.
15	3	action loop control
16	0	wait for new action calculation

Bit 8 **Reserved**

Note: Read as zero, should be written as zero.

Bit 11:9 **CNT_T: Count TRIGGER;** this reflects the count of active *TRIGGER* slopes (mod8).

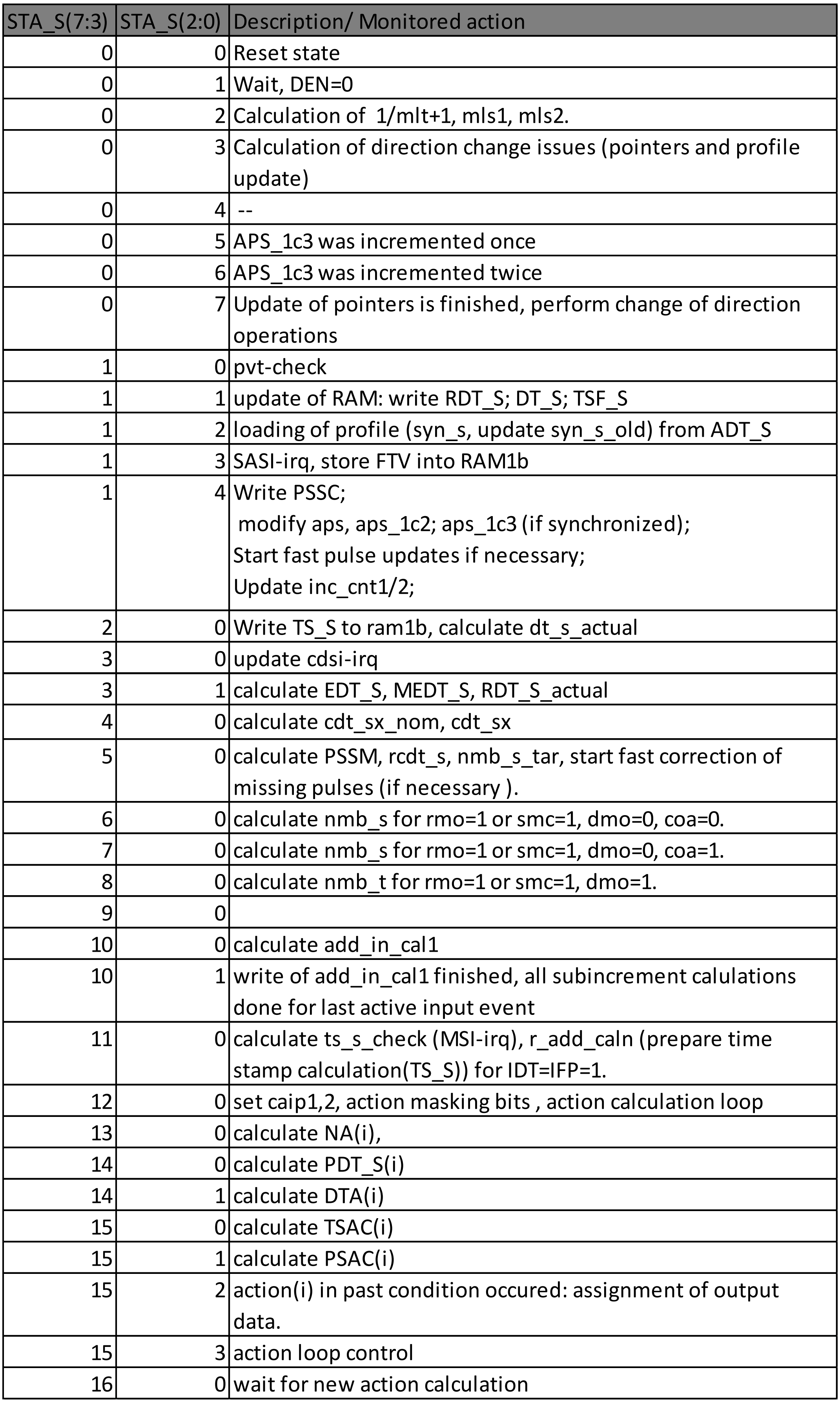
This value shows the number of active TRIGGER slopes (mod8)

Note: This value allows distinguishing if the above state machine status is consistent to other status values read before or after it.

Bit 19:12 **STA_S: Status of STATE state machine;** state binary coded

This bit field reflects the status of the STATE state machine

Note: The decimal step number 21 to 40 of the state machine is binary coded from 0x01 to 0x14 respectively using the upper 5 bits (19:15) after subtraction of 20 to the decimal value. The lower 3 bits (14:12) show substates of the corresponding state machine. When the DPLL is disabled this field is 0x000.



STA_S(7:3)	STA_S(2:0)	Description/ Monitored action
0	0	Reset state
0	1	Wait, DEN=0
0	2	Calculation of 1/mlt+1, mls1, mls2.
0	3	Calculation of direction change issues (pointers and profile update)
0	4	--
0	5	APS_1c3 was incremented once
0	6	APS_1c3 was incremented twice
0	7	Update of pointers is finished, perform change of direction operations
1	0	pvt-check
1	1	update of RAM: write RDT_S; DT_S; TSF_S
1	2	loading of profile (syn_s, update syn_s_old) from ADT_S
1	3	SASI-irq, store FTV into RAM1b

1	4	Write PSSC;
		modify aps, aps_1c2; aps_1c3 (if synchronized);
		Start fast pulse updates if necessary;
		Update inc_cnt1/2;

2	0	Write TS_S to ram1b, calculate dt_s_actual
3	0	update cdsi-irq
3	1	calculate EDT_S, MEDT_S, RDT_S_actual
4	0	calculate cdt_sx_nom, cdt_sx
5	0	calculate PSSM, rcdt_s, nmb_s_tar, start fast correction of missing pulses (if necessary).
6	0	calculate nmb_s for rmo=1 or smc=1, dmo=0, coa=0.
7	0	calculate nmb_s for rmo=1 or smc=1, dmo=0, coa=1.
8	0	calculate nmb_t for rmo=1 or smc=1, dmo=1.
9	0	
10	0	calculate add_in_cal1
10	1	write of add_in_cal1 finished, all subincrement calulations done for last active input event
11	0	calculate ts_s_check (MSI-irq), r_add_caln (prepare time stamp calculation(TS_S)) for IDT=IFP=1.
12	0	set caip1,2, action masking bits , action calculation loop
13	0	calculate NA(i),
14	0	calculate PDT_S(i)
14	1	calculate DTA(i)
15	0	calculate TSAC(i)
15	1	calculate PSAC(i)
15	2	action(i) in past condition occured: assignment of output data.
15	3	action loop control
16	0	wait for new action calculation

Bit 20 **Reserved**

Note: Read as zero, should be written as zero.

Bit 23:21 **CNT_S: Count STATE;** this reflects the count of active *STATE* slopes (mod8).

This value shows the number of active STATE slopes (mod8)

Note: This value allows distinguishing if the above state machine status is consistent to other status values read before or after it.

Bit 31:24 **Reserved**

Note: Read as zero, should be written as zero.

18.12.104 Register DPLL_INCF1_OFFSET

Address Offset:	**see Appendix B**	**Initial Value:**	**0x0000_0000**
	31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	
Bit	Reserved	DPLL_INCF1_OFFSET	
Mode	R	RW	
Initial Value	0x00	0x00_0000	

Bit 23:0 **DPLL_INCF1_OFFSET:** start value of the ADD_IN_ADDER1

In the case of set DPLL_CTRL_11-INCF1 the ADD_IN_ADDER1 starts always after an active new input event (TRIGGER in normal mode or STATE in emergency mode respectively) with this offset value. In the case of choosing DPLL_INCF1_OFFSET= 0xFFFFFF the generation of the first SUB_INC1 pulse is performed with the next TS_CLK. In the case of DPLL_INCF1_OFFSET= 0x000000 the first pulse is delayed by a full SUB_INC1 period and in the case of DPLL_INCF1_OFFSET= 0x7FFFFF the first pulse is delayed by a half SUB_INC1 period. Any other value is possible.

Bit 31:24 **Reserved**

Note: Read as zero, should be written as zero.

18.12.105 Register DPLL_INCF2_OFFSET

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								DPLL_INCF2_OFFSET																							
Mode	R								RW																							
Initial Value	0x00								0x00_0000																							

Bit 23:0 **DPLL_INCF2_OFFSET:** start value of the ADD_IN_ADDER2

In the case of set DPLL_CTRL_11-INCF2 the ADD_IN_ADDER2 starts always after an active new input event (STATE) with this offset value. In the case of choosing DPLL_INCF2_OFFSET= 0xFFFFFF the generation of the first SUB_INC2 pulse is performed with the next TS_CLK. In the case of DPLL_INCF2_OFFSET= 0x000000 the first pulse is delayed by a full SUB_INC2 period and in the case of DPLL_INCF2_OFFSET= 0x7FFFFF the first pulse is delayed by a half SUB_INC2 period. Any other value is possible.

Bit 31:24 **Reserved**

Note: Read as zero, should be written as zero.

18.12.106 Register DPLL_DT_T_START

Address Offset:	see Appendix B																Initial Value:								0x0000_0101							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								DPLL_DT_T_START																							
Mode	R								RW																							
Initial Value	0x00								0x00_0101																							

Bit 23:0 **DPLL_DT_T_START:** start value of DPLL_DT_T_ACT for the first increment after SIP1 is set to 1

For the first increment after setting SIP1 from 0 to 1 the value of DPLL_DT_T_START is taken instead of the calculated DPLL_DT_T_ACT for the current increment duration. This value should be always > 256 in order to avoid an overflow during the calculation of DPLL_RDT_T_ACT.

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

18.12.107 Register DPLL_DT_S_START

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_0101**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								DPLL_DT_S_START																							
Mode	R								RW																							
Initial Value	0x00								0x00_0101																							

Bit 23:0 **DPLL_DT_S_START:** start value of DPLL_DT_S_ACT for the first increment after SIP2 is set to 1
For the first increment after setting SIP2 from 0 to 1 the value of DPLL_DT_S_START is taken instead of the calculated DPLL_DT_S_ACT for the current increment duration. This value should be always > 256 in order to avoid an overflow during the calculation of DPLL_RDT_S_ACT.

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

18.12.108 Register DPLL_STA_MASK

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																STA_NOTIFY_S								STA_NOTIFY_T							
Mode	R																RW								RW							
Initial Value	0x0000																0x00								0x00							

Bit 7:0 **STA_NOTIFY_T:** notify value for STA_T of register DPLL_STA.

The STA_NOTIFY_T is representing a trigger mask of DPLL_STA.STA_T. When DPLL_STA.STA_T reaches the value of STA_NOTIFY_T the flag DPLL_STA_FLAG.STA_FLAG_T is set to '1' when DPLL_STA.STA_T is leaving the state STA_NOTIFY_T.

The signal is visible to MCS0 sub module as part of the special function register.

Bit 15:8 **STA_NOTIFY_S:** notify value for STA_S of register DPLL_STA.

The STA_NOTIFY_S is representing a trigger mask of DPLL_STA.STA_S. When DPLL_STA.STA_S reaches the value of STA_NOTIFY_S the flag DPLL_STA_FLAG.STA_FLAG_S is set to '1' when DPLL_STA.STA_S is leaving the state STA_NOTIFY_S.

Bit 31:16 **Reserved**

Note: Read as zero, should be written as zero.

18.12.109 Register DPLL_STA_FLAG

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																					INC_CNT2_FLAG	INC_CNT1_FLAG	STA_FLAG_S	Reserved							STA_FLAG_T
Mode	R																					RCw	RCw	RCw	R							RCw
Initial Value	0x000000																					0	0	0	0x00							0

Bit 0 **STA_FLAG_T:** Flag according to DPLL_MASK.STA_NOTIFY_T

The STA_FLAG_T is set to '1' indicating that the signal DPLL_STA.STA_T has left the state defined by the trigger mask of DPLL_STA_MASK.STA_NOTIFY_T.

The Flag is reset when this bit of the register is written to '1'.

The signal is visible to MCS0 sub module as part of the special function register.

Bit 7:1 **Reserved**

Note: Read as zero, should be written as zero.

Bit 8 **STA_FLAG_S:** Flag according to DPLL_STA_MASK.STA_NOTIFY_S

The STA_FLAG_S is set to '1' indicating that the signal DPLL_STA.STA_S has left the state defined by the trigger mask of DPLL_STA_MASK.STA_NOTIFY_S.

The Flag is reset when this bit of the register is written to '1'.

The signal is visible to MCS0 sub module as part of the special function register.

Bit 9 **INC_CNT1_FLAG:** Flag according to DPLL_INC_CNT1_MASK.INC_CNT1_NOTIFY

The INC_CNT1_FLAG is set to '1' indicating that the signal DPLL_INC:CNT1.INC_CNT1 has left the state defined by the trigger mask of DPLL_INC_CNT1_MASK.INC_CNT1_NOTIFY.

The Flag is reset when this bit of the register is written to '1'.

The signal is visible to MCS0 sub module as part of the special function register.

Bit 10 **INC_CNT2_FLAG:** Flag according to DPLL_INC_CNT2_MASK.INC_CNT2_NOTIFY

The INC_CNT2_FLAG is set to '1' indicating that the signal DPLL_INC_CNT2.INC_CNT2 has left the state defined by the trigger mask of DPLL_INC_CNT2_MASK.INC_CNT2_NOTIFY.

The Flag is reset when this bit of the register is written to '1'.

The signal is visible to MCS0 sub module as part of the special function register.

Bit 31:11 **Reserved**
Note: Read as zero, should be written as zero.

18.12.110 Register DPLL_INC_CNT1_MASK

Address Offset:	**see Appendix B**																**Initial Value: 0x0000_0000**															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								INC_CNT1_NOTIFY																							
Mode	R								RW																							
Initial Value	0x00								0x000000																							

Bit 23:0 **INC_CNT1_NOTIFY:** notify value for INC_CNT1 of register DPLL_INC_CNT1.

The INC_CNT1_NOTIFY is representing a trigger mask of DPLL_INC_CNT1.INC_CNT1. When DPLL_INC_CNT1.INC_CNT1 reaches the value of INC_CNT1_NOTIFY the flag DPLL_STA_FLAG.INC_CNT1_FLAG is set to '1' when DPLL_INC_CNT1.INC_CNT1 is leaving the state INC_CNT1_NOTIFY.

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

18.12.111 Register DPLL_INC_CNT2_MASK

Address Offset:	see Appendix B																Initial Value:								0x00000000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								INC_CNT2_NOTIFY																							
Mode	R								RW																							
Initial Value									0x000000																							

Bit 23:0 **INC_CNT2_NOTIFY:** notify value for INC_CNT2 of register DPLL_INC_CNT2.

The INC_CNT2_NOTIFY is representing a trigger mask of DPLL_INC_CNT2.INC_CNT2. When DPLL_INC_CNT2.INC_CNT2 reaches the value of INC_CNT2_NOTIFY the flag DPLL_STA_FLAG.INC_CNT2_FLAG is set to '1' when DPLL_INC_CNT2.INC_CNT2 is leaving the state INC_CNT2_NOTIFY.

Bit 31:24 **Reserved**

Note: Read as zero, should be written as zero.

18.12.112 Register DPLL_NUSC_EXT1

Address Offset:	see Appendix B																Initial Value:								0x0001_0001							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved	WSYN	Reserved							SYN_S_OLD							Reserved									SYN_S						
Mode	R	RAw	R							RPw							R									RPw						
Initial Value	0x00	0	0x00							0x01							0x00									0x01						

Number of Recent STATE Events used for Calculations

Bit 6:0 **SYN_S:** number of real and virtual events to be considered for the current increment.

This value reflects the NS value of the last valid increment, stored in ADT_S[i]; to be updated after all calculations in step 37 of Table 18.8.6.7.1.

Note: This value can only be written when the WSYN bit in this register is set.

Bit 15:7 **Reserved**

Note: Read as zero, should be written as zero.

Bit 22:16 **SYN_S_OLD:** number of real and virtual events to be considered for the last increment.

This value reflects the NS value of the last but one valid increment, stored in ADT_S[i]; is updated automatically when writing SYN_S.

Note: This value is updated by the SYN_S value when the WSYN bit in this register is set.

Bit 29:23 **Reserved**

Note: Read as zero, should be written as zero.

Bit 30 **WSYN:** write control bit for SYN_S and SYN_S_OLD; read as zero.

0 = the SYN_S value is not writeable

1 = the SYN_S value is writeable

Bit 31 **Reserved**

Note: Read as zero, should be written as zero.

Note: This register is only used when DPLL_CTRL_11.STATE_EXT is set. If DPLL_CTRL_11.STATE_EXT is not set any read/write access to this register will return AEI_STATUS = 0b10.

18.12.113 Register DPLL_NUSC_EXT2

Address Offset:	see Appendix B																Initial Value:									0x0000_0001						
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	WVSN	Reserved	WNUS	Reserved						VSN							FSS	Reserved								NUSE						
Mode	RAw	R	RAw	R						RPw							RPw	R								RPw						
Initial Value	0	0x00	0	0x0						0x00							0	0x00								0x01						

Number of Recent STATE Events used for Calculations

Bit 6:0 **NUSE:** Number of recent *STATE* events used for SUB_INCx calculations modulo 2*(SNU$_{max}$+1).

No gap is considered in that case for this value, but in the VSN value (see below): This register is set by the CPU but reset automatically to “1” by a change of direction or loss of LOCK. Each other value can be set by the CPU, maybe Full_SCALE, HALF_SCALE or parts of them. The relation values QDT_Sx are calculated using NUSE values in the past with its maximum value of 2*SNU+1.

Note: This value can only be written when the WNUS bit is set.

Bit 14:7 **Reserved**

Note: Read as zero, should be written as zero.

Bit 15 **FSS: FULL_SCALE of STATE;** this value is to be set, when NUSE is set to FULL_SCALE

0 = the NUSE value is less then FULL_SCALE

1 = the NUSE value is equal to FULL_SCALE

This value is set by the CPU, but reset automatically to “0” by a change of direction or loss of LOCK.

Note: This value can only be written when the WNUS bit is set.

Bit 22:16 **VSN: virtual STATE number;** number of virtual state increments in the current NUSE region.

This value reflects the number of virtual increments in the current NUSE region; for NUSE=1 this value is zero, when the CPU sets NUSE to a value > 1 or zero(2^7 modulo 2^7) , it must also set VSN to the correspondent value;

the VSN value is subtracted from the NUSE value in order to get the corresponding APS value for the past; the VSN value is not used for the APS_1C2 pointer.

VSN is to be updated by the CPU when a new gap is to be considered for NUSE or a gap is leaving the NUSE region; for this purpose the SASI interrupt can be used; no further update of VSN is necessary when NUSE is set to FULL_SCALE

Note: This value can only be written when the WVSN bit is set.

Bit 28:23 **Reserved**
Note: Read as zero, should be written as zero.

Bit 29 **WNUS:** write control bit for NUSE; read as zero.
0 = the NUSE value is not writeable
1 = the NUSE value is writeable

Bit 30 **Reserved**
Note: Read as zero, should be written as zero.

Bit 31 **WVSN:** write control bit for VSN; read as zero.
0 = the VSN value is not writeable
1 = the VSN value is writeable

Note: This register is only used when DPLL_CTRL_11.STATE_EXT is set. If DPLL_CTRL_11.STATE_EXT is not set any read/write access to this register will return AEI_STATUS = 0b10.

18.12.114 Register DPLL_APS_EXT

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved											APS_1C2							WAPS_1C2	Reserved				APS							WAPS	Reserved
Mode	R											RPw							RAw	R				RPw							RAw	R
Initial Value	0x000											0x00							0	0				0x00							0	0

Actual RAM Pointer Address for STATE

Bit 0 **Reserved**
Note: Read as zero, should be written as zero.

Bit 1 **WAPS:** Write bit for address pointer APS, read as zero.
0 = the APS is not writeable
1 = the APS is writeable

Bit 8:2 **APS: Address pointer STATE;** Actual RAM pointer address value for DT_S[i] and RDT_S[i]

Actual RAM pointer and synchronization position/value of *STATE* events in FULL_SCALE for up to 128 *STATE* events but limited to 2*(SNU+1-SYN_NS) in normal and emergency mode for SYSF=0 or to 2*(SNU+1)-SYN_NS for SYSF=1 respectively; See 18.10.

APS is incremented (decremented) by one for each active *STATE* event and DIR2=0 DIR2=1). The APS offset value is added in the above shown bit position with the subsection offset of the RAM region.

Note: The APS pointer value is directed to the RAM position, in which the data values are to be written, which correspond to the last increment. The APS value is not to be changed, when the direction (shown by DIR2) changes, because it points always to a storage place after the considered increment. Changing of DIR2 takes place always after an active *STATE* event and the resulting increment/decrement.

Note: This value can only be written when the WAPS bit is set.

Bit 12:9 **Reserved**

Note: Read as zero, should be written as zero.

Bit 13 **WAPS_1C2:** Write bit for address pointer APS_1C2, read as zero.

0 = the APS_1C2 is not writeable

1 = the APS_1C2 is writeable

Bit 20:14 **APS_1C2: Address pointer STATE for RAM region 1c2;** Actual RAM pointer address value for TSF_S[i].

Initial value: zero (0x00). Actual RAM pointer and synchronization position/value of *STATE* events in FULL_SCALE for up to 128 *STATE* events but limited to 2*(SNU+1) in normal and emergency mode; this pointer is used for the RAM region 1c2.

For SYS=1: APS_1C2 is incremented (decremented) by SYN_S_OLD for each active *STATE* event and DIR2=0 (DIR2=1).

For SYS=0: APT_1c2 is incremented or decremented by 1 respectively.

The APS_1C2 offset value is added in the above shown bit position with the subsection offset of the RAM region.

In addition when the APS_1C3 value is written by the CPU - in order to synchronize the DPLL- with the next active *STATE* event the APS_1C2_EXT value is added/subtracted (while APS_1C2_STATUS is one; see DPLL_APT_SYNC register at chapter 18.12.25).

Note: This value can only be written when the WAPS_1C2 bit is set

Bit 31:21 **Reserved**

Note: Read as zero, should be written as zero.

Note: This register is only used when DPLL_CTRL_11.STATE_EXT is set. If DPLL_CTRL_11.STATE_EXT is not set any read/write access to this register will return AEI_STATUS = 0b10.

18.12.115 Register DPLL_APS_1C3_EXT

Address Offset:	see Appendix B																Initial Value:	0x0000_0000														
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																							APS_1C3							Reserved	
Mode	R																							RW							R	
Initial Value	0x000000																							0x00							00	

Bit 1:0 **Reserved**

Note: Read as zero, should be written as zero.

Bit 8:2 **APS_1C3: Address pointer STATE for RAM region 1c3;** Actual RAM pointer address value for ADT_S[i]

Initial value: zero (0x00). Actual RAM pointer and synchronization position/value of *STATE* events in FULL_SCALE for up to 128 *STATE* events but limited to 2*(SNU+1-SYN_NS) in normal and emergency mode for SYSF=0 or to 2*(SNU+1)-SYN_NS for SYSF=1 respectively; this pointer is used for the RAM region 1c3. See 18.10.

The RAM pointer is set by the CPU accordingly, when the synchronization condition was detected.

Bit 31:9 **Reserved**

Note: Read as zero, should be written as zero.

Note: The APS_1C3 pointer value is directed to the RAM position of the profile element in RAM region 1c2, which corresponds to the current increment. When changing the direction DIR1 or DIR2 respectively, this is always known before an active *STATE* event is processed. This is because of the pattern recognition in SPE (for PMSM) or because of the direction change recognition by TRIGGER. This direction change results in an automatic increment (forwards) or decrement (backwards) when the input event occurs in addition with a 2 times correction.

The APS_1C3_x offset value is added in the above shown bit position with the subsection address offset of the corresponding RAM region.

Note: This register is only used when DPLL_CTRL_11.STATE_EXT is set. If DPLL_CTRL_11.STATE_EXT is not set any read/write access to this register will return AEI_STATUS = 0b10.

18.12.116 Register DPLL_APS_SYNC_EXT

Address Offset:	see Appendix B																Initial Value: 0x0000_0000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved									APS_1C2_OLD							APS_1C2_STATU	Reserved								APS_1C2_EXT						
Mode	R									RW							RW	R								RW						
Initial Value	0x000									0x00							0	0								0x00						

STATE Time Stamp Field Offset at Synchronization Time

Bit 6:0 **APS_1C2_EXT: Address pointer 1c2 extension;** this offset value determines, by which value the APS_1C2 is changed at the synchronization time; set by CPU before the synchronization is performed.

This offset value is the number of virtual increments to be inserted in the TSF for an imminent intended synchronization; the CPU sets its value dependent on the gaps until the synchronization time taking into account the considered NUSE value to be set and including the next future increment (when SYN_S_OLD is still 1). When the synchronization takes place, this value is to be added to the APS_1C2 address pointer (for forward direction, DIR2=0) and the APT_1c2_status bit is cleared after it. For backward direction subtract APS_1C2_EXT accordingly.

Note: When the synchronization is intended and the NUSE value is to be set to FULL_SCALE after it, the APS_1C2_EXT value must be set to SYN_NS (for SYSF=1) or 2*SYN_NS (for SYSF=0) in order to be able to fill all gaps in the extended TSF_S with the corresponding values by the CPU.

When still not all values for FULL_SCALE are available, the APS_1C2_EXT value considers only a share according to the NUSE value to be set after the synchronization.

Bit 14:7 **Reserved**

Note: Read as zero, should be written as zero.

Bit 15 **APS_1C2_STATUS: Address pointer 1c2 status;** set by CPU before the synchronization is performed. The value is cleared automatically when the APS_1C2_OLD value is written.
0 = APS_1C2_EXT is not to be considered.
1 = APS_1C2_EXT has to be considered for time stamp field extension.

Bit 22:16 **APS_1C2_OLD: Address pointer STATE for RAM region 1c2 at synchronization time;** this value is set by the current APS_1C2 value when the synchronization takes place for the first active STATE event after writing APS_1C3 but before adding the offset value APS_1C2_EXT (that means: when APS_1C2_STATUS=1).
Address pointer APS_1C2 value at the moment of synchronization, before the offset value is added, that means the pointer with this value points to the last value before the additional inserted gap

Bit 31:23 **Reserved**
Note: Read as zero, should be written as zero.

Note: This register is only used when DPLL_CTRL_11.STATE_EXT is set. If DPLL_CTRL_11.STATE_EXT is not set any read/write access to this register will return AEI_STATUS = 0b10.

18.12.117 Register DPLL_CTRL_EXT

<table>
<tr><td><b>Address Offset:</b></td><td colspan="16"><b>see Appendix B</b></td><td colspan="16"><b>Initial Value: 0x0000_0017</b></td></tr>
<tr><td></td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>Bit</td><td colspan="10">Reserved</td><td colspan="6">SYN_NS</td><td colspan="10">Reserved</td><td colspan="6">SNU</td></tr>
<tr><td>Mode</td><td colspan="10">R</td><td colspan="6">RPw</td><td colspan="10">R</td><td colspan="6">RPw</td></tr>
<tr><td>Initial Value</td><td colspan="10">0x000</td><td colspan="6">0x00</td><td colspan="10">0</td><td colspan="6">0x17</td></tr>
</table>

STATE Time Stamp Field Offset at Synchronization Time

Bit 5:0 **SNU[3]: STATE number;** SNU+1 is number of nominal *STATE* events in HALF_SCALE (1...32).
Note: This bit can only be written when the DPLL is disabled.

Note: The number of nominal *STATE* events is the decimal value plus 1. This value can only be written when (RMO=0 and SMC=0) or DEN=0. Set SSL=00 before changing this value and set RMO=1 only after FULL_SCALE with SSL>0.

Bit 15:6 **Reserved**

Note: Read as zero, should be written as zero.

Bit 21:16 **SYN_NS: Synchronization number of *STATE*;** summarized number of virtual increments in HALF_SCALE

sum of all systematic missing *STATE* events in HALF_SCALE (for SYSF=0) or FULL SCALE (for SYSF=1) ; the SYN_NS missing *STATES* can be divided up to an arbitrary number of blocks. The pattern of events and missing events in FULL_SCALE is shown in RAM region 1c3 as value NS in addition to the adapted values. The number of stored increments in FULL_SCALE must be equal to 2*(SNU+1-SYN_NS) for SYSF=0 or 2*(SNU+1)-SYN_NS for SYSF=1 . This pattern is written by the CPU beginning from a fixed reference point (maybe beginning of the FULL_SCALE region). The relation to the actual increment is established by setting of the profile RAM pointer APS_1C3 in an appropriate relation to the RAM pointer APS of the actual increment by the CPU.

Note: This value can only be written when the DPLL is disabled.

Note: This value can only be written when (RMO=0 and SMC=0) or DEN=0. Set SSL=00 before changing this value and set RMO=1 only after FULL_SCALE with SSL>0.

Bit 31:22 **Reserved**

Note: Read as zero, should be written as zero.

Note: This register is only used when DPLL_CTRL_11.STATE_EXT is set. If DPLL_CTRL_11.STATE_EXT is not set any read/write access to this register will return AEI_STATUS = 0b10.

18.12.118 Memory DPLL_RR1A

Address Offset:	see Appendix B																Initial Value: 0x0000_0000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	DATA																															
Mode	RW																															
Initial Value	0x0000_0000																															

Bit 31:0 **DATA**

18.12.119 Memory DPLL_RR2

Address Offset:	see Appendix B																Initial Value: 0x0000_0000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit									DATA																							
Mode									RW																							
Initial Value									0x0000_0000																							

Bit 23:0 **DATA**

18.13 DPLL RAM Region 1a value description

18.13.1 Memory DPLL_PSA[i]

Address Offset:	see Appendix B																Initial Value: 0x0000_0000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								PSA																							
Mode	R								RPw																							
Initial Value	0x00								0x000000																							

Position Request for Action i, (RAM1a, i=0...NOAC-1)[1)]

Bit 23:0 **PSA** Position information of a desired action (i=0...NOAC-1)[1)].
Note: This value can only be written when the DPLL is disabled.

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

[1)] **Note:** The PSA values for actions 24...31 are not available for all devices. Please refer to appendix B.

18.13.2 Memory DPLL_DLA[i]

Address Offset:	see Appendix B																Initial Value: 0x0000_0000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								DLA																							
Mode	R								RPw																							
Initial Value	0x00								0x000000																							

Time to React for Action i, (RAM1a, i=0...NOAC-1)[1)]

Bit 23:0 **DLA** Time to react before the corresponding position value of a desired action is reached (x=0...NOAC-1)[1)]. In the case of LOW_RES=1 (see chapter 18.4.2) this delay value must be also given as low resolution value.
Note: This value can only be written when the DPLL is disabled.

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

[1] **Note:** The DLA values for actions 24...31 are only available for all devices. Please refer to appendix B.

18.13.3 Memory DPLL_NA[i]

Address Offset:	**see Appendix B**																**Initial Value: 0x0000_0000**															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								NOT_USED				DW										DB									
Mode	R								RW				RPw										RPw									
Initial Value	0x00								0x0				0x000										0x000									

Calculated Relative Time to Action i, (RAM1a, i=0...NOAC-1)[1]

Bit 9:0 **DB:** number of events to Action_i (fractional part, i=0...NOAC-1)[1].
Note: This value can only be written when the DPLL is disabled.

Bit 19:10 **DW:** number of events to Action_i (integer part, i=0...NOAC-1)[1].
Note: Use the maximum value for NA_DW=0x3FF in the case of a calculated value which exceeds the represent able value.

Note: This value can only be written when the DPLL is disabled.

Bit 23:20 **Not used**
Note: must be written to zero.

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

[1] **Note:** The NA values for actions 24...31 are not available for devices. Please refer to appendix B.

18.13.4 Memory DPLL_DTA[i]

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								DTA																							
Mode	R								RPw																							
Initial Value	0x00								0x000000																							

Calculated Relative Time to Action i, (RAM1a, i=0...NOAC-1)[1)]

Bit 23:0 **DTA:** calculated relative time to ACTION_i (i=0...NOAC-1)[1)]

Note: This value can only be written when the DPLL is disabled. The DTA value is a positive integer value. When calculations using equations DPLL-12 or DPLL-14 result in a negative value, it is replaced by zero.

Bit 31:24 **Reserved**

Note: Read as zero, should be written as zero.

[1)] **Note:** The DTA values for actions 24...31 are not available for all devices. Please refer to appendix B.

18.14 DPLL RAM Region 2 value description

Note: Bits 31 to 24 of RAM region 2 are not implemented and therefore always read as zero (reserved). Other bits which are declared as reserved are not protected against writing. Unused address regions are not protected against writing when implemented.

18.14.1 Memory DPLL_RDT_T[i]

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								RDT_T																							
Mode	R								RW																							
Initial Value	0x00								0x0000 00																							

Reciprocal Values of the Nominal TRIGGER Increments Duration in FULL_SCALE

Bit 23:0 **RDT_T: Reciprocal difference time of TRIGGER;** 2* (TNU+1-SYN_NT) stored values nominal reciprocal value of the number of time stamp clocks measured in the corresponding increment (which is divided by the number of nominal increments); multiplied by $*2^{32}$ while only the lower 24 bits are used; the LSB is rounded up, when the next truncated bit is 1.

Note: There are 2* (TNU+1- SYN_NT) entries. The maximum number of entries is restricted to a value corresponding to the OSS value in the DPLL_OSW register.

Bit 31:24 **Reserved**

Note: Read as zero, should be written as zero.

Note: The starting index for Memory DPLL_RDT_T[i] in RAM2 is defined by the parameter AOSV_2A in DPLL_AOSV2 Register

18.14.2 Memory DPLL_TSF_T[i]

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								TSF_T																							
Mode	R								RW																							
Initial Value	0x00								0x0000 00																							

Time Stamp Values of the Nominal TRIGGER Increments in FULL_SCALE

Bit 23:0 **TSF_T:** Time stamp field of active TRIGGER slopes

Note: There are 2* (TNU+1) entries. The maximum number of entries is restricted to a value corresponding to the OSS value in the DPLL_OSW register.

Bit 31:24 **Reserved**
Note: Read as zero, should be written as zero.

Note: The starting index for Memory DPLL_TSF_T[i] in RAM2 is defined by the parameter AOSV_2C in DPLL_AOSV2 Register

18.14.3 Memory DPLL_ADT_T[i]

Address Offset:	**see Appendix B**																**Initial Value:**	**0x0000_0000**														
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								NOT_USED					NT			TINT			PD												
Mode	R								RW					RW			RW			R												
Initial Value	0x00								0x00					000			000															

Adapt and Profile Values of the TRIGGER Increments in FULL_SCALE

Bit 12:0 **PD: Physical deviation;** Adapt values for each nominal *TRIGGER* increment in FULL_SCALE (sint13);

The PD value does mean the number of SUB_INC1 pulses to be added to NT*((MLT+1) + PD);
the absolute value of a negative PD must not exceed (MLT+1) or MLS1 respectively;
systematic missing *TRIGGER* events must be considered for the value of PD;

Bit 15:13 **TINT:** *TRIGGER* Interrupt information;
depending on the value up to 7 different interrupts can be generated. In the current version the 5 interrupts TE0_IRQ ... TE4_IRQ are supported by TINT=”001”, “010”, "011”, “100”, “101” respectively. For the values "000", "110" and "111" no interrupt is generated and no other reaction is performed.
The corresponding interrupt is activated, when the TINT value is read by the DPLL together with the other values (PD, NT) according to the profile.

Bit 18:16 **NT: Number of TRIGGERs;** number of nominal *TRIGGER* parts in the corresponding increment.

Note: There are 2* (TNU+1- SYN_NT) entries. The maximum number of entries is restricted to a value corresponding to the OSS value in the DPLL_OSW register.

Bit 23:19 **Not used**

Note: must be written to zero.

Bit 31:24 **Reserved**

Note: Read as zero, should be written as zero.

Note: The starting index for Memory DPLL_ADT_T[i] in RAM2 is defined by the parameter AOSV_2C in DPLL_AOSV2 Register

18.14.4 Memory DPLL_DT_T[i]

Address Offset:	**see Appendix B**																**Initial Value: 0x0000_0000**															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								DT_T																							
Mode	R								RW																							
Initial Value	0x00								0x000000																							

Nominal TRIGGER Increments Duration in FULL_SCALE

Bit 23:0 **DT_T: Difference time of TRIGGER;** increment duration values for each *TRIGGER* increment in FULL_SCALE divided by the number of nominal increments (nominal value).

Note: There are 2* (TNU+1- SYN_NT) entries. The maximum number of entries is restricted to a value corresponding to the OSS value in the DPLL_OSW register.

Bit 31:24 **Reserved**

Note: Read as zero, should be written as zero.

Note: The starting index for Memory DPLL_DT_T[i] in RAM2 is defined by the parameter AOSV_2D in DPLL_AOSV2 Register

18.15 MCS to DPLL Register description

As already mentioned in section 18.10, the following registers of the MCS2DPLL interface are exclusively accessible by the MCS instance of cluster 0 and cannot be accessed directly via CPU.

18.15.1 Register MCS2DPLL_DEB0

Address Offset:	**see Appendix B**																**Initial Value:**															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	n/a								DATA																							
Mode									RW																							
Initial Value									0x000000																							

Bit 23:0 DATA: Data exchange buffer 0. Actual content depends on whether it is a Read or Write operation from MCS.

READ access from MCS:

Duration of the last increment DT_S_ACT (18.7.5). This value is updated by the DPLL during the update of ram (STA_S = 0b0000_1001) and is ready to be read when STA_S is modified to 0b0000_1010.

WRITE access from MCS:

The DPLL expects DT_S[p-1] (18.6.3.5) or DT_S[p+1](18.6.5.2) during the increment prediction (STA_S = 0b0001_0000).

Bit 31:24 n/a

Note: The data write from MCS should be performed between the two writes to MCS2DPLL_DEB15 (MCS2DPLL_STATUS_INFO)

18.15.2 Register MCS2DPLL_DEB1

Address Offset:	see Appendix B																Initial Value:															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	n/a								DATA																							
Mode									RW																							
Initial Value									0x000000																							

Bit 23:0 DATA: Data exchange buffer 1.

READ access from MCS:

Reads as 0.

WRITE access from MCS:

The DPLL expects RDT_S[p-1] (18.6.3.4) or RDT_S[p+1](18.6.5.1) during the increment prediction (STA_S = 0b0001_0000) and RDT_S[t-1] (18.7.3) or RDT_S[t+1](18.7.4) during the action calculation (STA_S = 0b0111_0000)

Bit 31:24 n/a

Note: In both cases, the data write from MCS should be performed between the two writes to MCS2DPLL_DEB15 (MCS2DPLL_STATUS_INFO).

18.15.3 Register MCS2DPLL_DEB2

Address Offset:	see Appendix B																Initial Value:															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	n/a								DATA																							
Mode									RW																							
Initial Value									0x000000																							

Bit 23:0 DATA: Data exchange buffer 2. Actual content depends on whether it is a Read or Write operation from MCS.

READ access from MCS:

TS_Sx (18.7.5.5). Use to compute/update the time stamp values for STATE during the update of ram (STA_S = 0b0000_1001)

WRITE access from MCS:

The DPLL expects RDT_S[p-q-1] (18.6.3.5) or RDT_S[p+q+1](18.6.5.2) during the increment prediction (STA_S = 0b0001_0000).

Bit 31:24 n/a

Note: The data read from MCS should be performed between the two writes to MCS2DPLL_DEB15 (MCS2DPLL_STATUS_INFO).

Note: The data write from MCS should be performed between the two writes to MCS2DPLL_DEB15 (MCS2DPLL_STATUS_INFO)

18.15.4 Register MCS2DPLL_DEB3

Address Offset:	see Appendix B																Initial Value:															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	n/a								DATA																							
Mode									RW																							
Initial Value									0x000000																							

Bit 23:0 DATA: Data exchange buffer 3. Actual content depends on whether it is a Read or Write operation from MCS.

READ access from MCS:

DT_Sx (18.7.5.5). Use to compute/update the time stamp values for STATE during the update of ram (STA_S = 0b0000_1001)

WRITE access from MCS:

The DPLL expects DT_S[p-q] (18.6.3.5) or DT_S[p+q](18.6.5.2) during the increment prediction (STA_S = 0b0001_0000).

Bit 31:24 n/a

Note: The data read from MCS should be performed between the two writes to MCS2DPLL_DEB15 (MCS2DPLL_STATUS_INFO).

18.15.5 Register MCS2DPLL_DEB4

Address Offset:	see Appendix B																Initial Value:															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	n/a								DATA																							
Mode									RW																							
Initial Value									0x000000																							

Bit 23:0 DATA: Data exchange buffer 4. Actual content depends on whether it is a Read or Write operation from MCS.

READ access from MCS:

SYN_S_OLD (18.7.5.5). Use to compute/update the time stamp values for STATE during the update of ram (STA_S = 0b0000_1001)

WRITE access from MCS:

The DPLL expects RDT_S[p-q] (18.6.3.7) or RDT_S[p+q](18.6.5.4) during the increment prediction (STA_S = 0b0001_0000) and RDT_S[t-q] (18.7.3) or RDT_S[t+q](18.7.4) during the action calculation (STA_S = 0b0111_0000)

Bit 31:24 n/a

Note: The data read from MCS should be performed between the two writes to MCS2DPLL_DEB15 (MCS2DPLL_STATUS_INFO).

Note: The data write from MCS should be performed between the two writes to MCS2DPLL_DEB15 (MCS2DPLL_STATUS_INFO)

18.15.6 Register MCS2DPLL_DEB5

Address Offset:	see Appendix B																Initial Value:															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	n/a								DATA																							
Mode									RW																							
Initial Value									0x0000 00																							

Bit 23:0 DATA: Data exchange buffer 5. Actual content depends on whether it is a Read or Write operation from MCS.

READ access from MCS:

M_DW (m in 18.7.3 and 18.7.4). Use to provide the proper Time Stamp Field value during the action calculation (STA_S = 0b0111_0000)

WRITE access from MCS:

The DPLL expects DT_S[p-q+1] (18.6.3.7) or DT_S[p+q-1](18.6.5.4) during the increment prediction (STA_S = 0b0001_0000) and DT_S[t-q+1] (18.7.3) or DT_S[t+q-1](18.7.4) during the action calculation (STA_S = 0b0111_0000)

Bit 31:24 n/a

Note: The data write from MCS should be performed between the two writes to MCS2DPLL_DEB15 (MCS2DPLL_STATUS_INFO).

18.15.7 Register MCS2DPLL_DEB6

Address Offset:	see Appendix B																Initial Value:															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	n/a								DATA																							
Mode									RW																							
Initial Value									0x0000 00																							

Bit 23:0 DATA: Data exchange buffer 6.
READ access from MCS:
Reads as 0.

WRITE access from MCS:
The DPLL expects ADT_S[APS_1C2] during the update of RAM (STA_S = 0b0000_1001) and change of direction (STA_S = 0b0000_0100 and STA_S = 0b0000_0110)

Bit 31:24 n/a

Note: In both cases, the current ADT_S[APS_1C2] value should be stored in the register before unlocking the state machine the second time, i.e.: between the two writes to MCS2DPLL_DEB15 (MCS2DPLL_STATUS_INFO).

Note: The format of ADT_S should match the defined in 18.12.90

18.15.8 Register MCS2DPLL_DEB7

Address Offset:	**see Appendix B**																**Initial Value:**															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	n/a								DATA																							
Mode									RW																							
Initial Value									0x000000																							

Bit 23:0 DATA: Data exchange buffer 7. Actual content depends on whether it is a Read or Write operation from MCS.
READ access from MCS:
Duration of the reciprocal of the last increment RDT_S_ACT (18.7.5). This value is written by the DPLL during the update of ram (STA_S = 0b0000_1001) and is ready to be read when STA_S is modified to 0b0000_1010.

WRITE access from MCS:
The DPLL expects the reciprocal of the last increment RDT_S[APS] (18.7.5) before it is overwritten with RDT_S_ACT during the update of ram (STA_S = 0b0000_1001). For the action calculation, this value is needed as well as RDT_S[t] in 18.7.3 and 18.7.4.

Bit 31:24 n/a

Note: During an update of ram, perform the write before unlocking the state machine (STA_S = 0b0000_1001), i.e.: after the first write to MCS2DPLL_DEB15 (MCS2DPLL_STATUS_INFO) but before the second write. During the action calculation, the data write from MCS should be performed between the two writes to MCS2DPLL_DEB15 (MCS2DPLL_STATUS_INFO).

18.15.9 Register MCS2DPLL_DEB8

Address Offset:	see Appendix B																Initial Value:															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	n/a								DATA																							
Mode									RW																							
Initial Value									0x0000 00																							

Bit 23:0 DATA: Data exchange buffer 8.

READ access from MCS:

Reads as 0.

WRITE access from MCS:

The DPLL expects TSF_S[p] (18.7.3 and 18.7.4) during the action calculation (STA_S = 0b0111_0000)

Bit 31:24 n/a

Note: The data write from MCS should be performed between the two writes to MCS2DPLL_DEB15 (MCS2DPLL_STATUS_INFO)

18.15.10 Register MCS2DPLL_DEB9

Address Offset:	see Appendix B																Initial Value:															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	n/a								DATA																							
Mode									RW																							
Initial Value									0x0000 00																							

Bit 23:0 DATA: Data exchange buffer 9.
READ access from MCS:
Reads as 0.

WRITE access from MCS:
The DPLL expects TSF_S[p-n] (18.7.3) or TSF_S[p+n](18.7.4) during the action calculation (STA_S = 0b0111_0000)

Bit 31:24 n/a

Note: The data write from MCS should be performed between the two writes to MCS2DPLL_DEB15 (MCS2DPLL_STATUS_INFO)

18.15.11 Register MCS2DPLL_DEB10

Address Offset:	see Appendix B																Initial Value:															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	n/a								DATA																							
Mode									RW																							
Initial Value									0x0000 00																							

Bit 23:0 DATA: Data exchange buffer 10.
READ access from MCS:
Reads as 0.

WRITE access from MCS:

The DPLL expects TSF_S[p+m-n] (18.7.3) or TSF_S[p-m+n](18.7.4) during the action calculation (STA_S = 0b0111_0000)

Bit 31:24 n/a

Note: The data write from MCS should be performed between the two writes to MCS2DPLL_DEB15 (MCS2DPLL_STATUS_INFO)

18.15.12 Register MCS2DPLL_DEB11

Address Offset:	**see Appendix B**																**Initial Value:**															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	n/a								DATA																							
Mode									RW																							
Initial Value									0x000000																							

Bit 23:0 DATA: Data exchange buffer 11.

READ access from MCS:

Reads as 0.

WRITE access from MCS:

The DPLL expects TSF_S[p+m] (18.7.3) or TSF_S[p-m](18.7.4) during the action calculation (STA_S = 0b0111_0000)

Bit 31:24 n/a

Note: The data write from MCS should be performed between the two writes to MCS2DPLL_DEB15 (MCS2DPLL_STATUS_INFO)

18.15.13 Register MCS2DPLL_DEB12

Address Offset:	**see Appendix B**																**Initial Value:**															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	n/a								DATA																							
Mode									RW																							
Initial Value									0x0000 00																							

Bit 23:0 DATA: Data exchange buffer 12.

READ access from MCS:

Reads as 0.

WRITE access from MCS:

The DPLL expects the future adapt information ADT_S[APS_1C2+1] (when in forwards) or ADT_S[APS_1C2-1] (when in backwards) (18.6.3.5) during the update of RAM (STA_S = 0b0000_1001) and change of direction (STA_S = 0b0000_0100 and STA_S = 0b0000_0110)

Bit 31:24 n/a

Note: In both cases, the current ADT_S[APS_1C2+1] or ADT_S[APS_1C2-1] value should be stored in the register before unlocking the state machine the second time, i.e.: between the two writes to MCS2DPLL_DEB15 (MCS2DPLL_STATUS_INFO).

Note: The format of ADT_S should match the defined in 18.12.90

18.15.14 Register MCS2DPLL_DEB13

Address Offset:	see Appendix B																Initial Value:															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	n/a								DATA																							
Mode									R																							
Initial Value									0x000000																							

Bit 23:0 DATA: Data exchange buffer 13.
READ access from MCS:
Reads as 0.

WRITE access from MCS:
Ignored during DPLL processing.

Bit 31:24 n/a

18.15.15 Register MCS2DPLL_DEB14

Address Offset:	see Appendix B																Initial Value:															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	n/a								DATA																							
Mode									R																							
Initial Value									0x000000																							

Bit 23:0 DATA: Data exchange buffer 14.
READ access from MCS:
Reads as 0.

WRITE access from MCS:
Ignored during DPLL processing.

Bit 31:24 n/a

18.15.16 Register MCS2DPLL_DEB15

Address Offset:	**see Appendix B**																**Initial Value:**															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	n/a								DATA																							
Mode									RW																							
Initial Value									0x0000 00																							

Bit 23:0 DATA: Data exchange buffer 15.

READ access from MCS:

Reads as 0.

WRITE access from MCS:

Unlocks the DPLL STATE state machine. See 18.10.2.

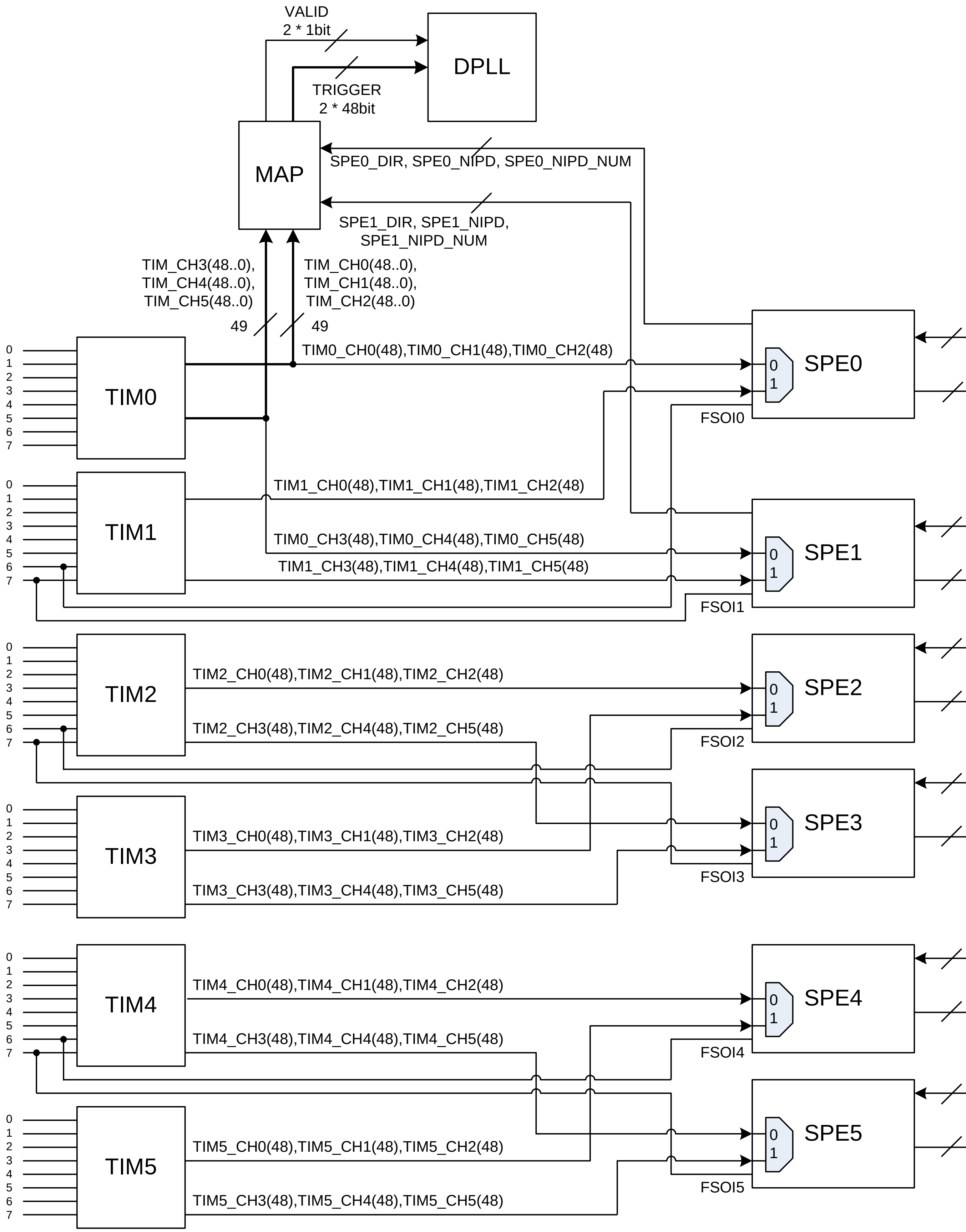
Bit 31:24 n/a

19 Sensor Pattern Evaluation (SPE)

19.1 Overview

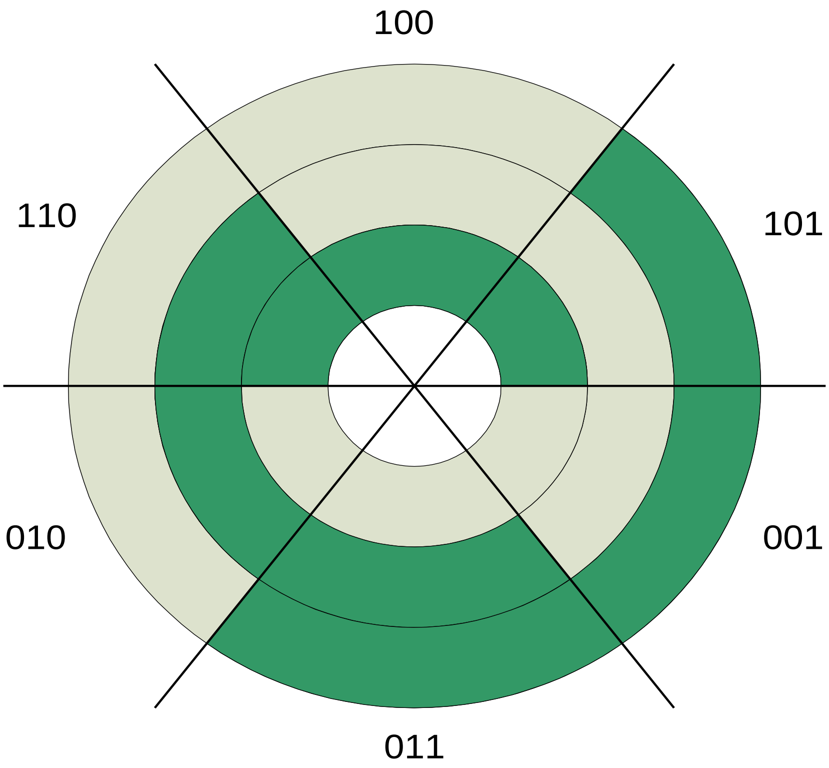
The Sensor Pattern Evaluation (SPE) submodule can be used to evaluate three hall sensor inputs and together with the TOM module to support the drive of BLDC engines. Thus, the input signals are filtered already in the connected TIM channels. In addition, the SPE submodule can be used as an input stage to the MAP submodule if the DPLL should be used to calculate the rotation speed of one or two electric engine(s). The integration of the SPE submodule into the overall GTM-IP architecture concept is shown in figure 19.1.1.

19.1.1 SPE Submodule integration concept into GTM-IP


VALID
2 * 1bit
DPLL
TRIGGER
2 * 48bit
MAP
SPE0_DIR, SPE0_NIPD, SPE0_NIPD_NUM
SPE1_DIR, SPE1_NIPD,
SPE1_NIPD_NUM
TIM_CH3(48..0),
TIM_CH4(48..0),
TIM_CH5(48..0)
TIM_CH0(48..0),
TIM_CH1(48..0),
TIM_CH2(48..0)
49
49
TIM0_CH0(48),TIM0_CH1(48),TIM0_CH2(48)
TIM0
SPE0
FSOI0
TIM1_CH0(48),TIM1_CH1(48),TIM1_CH2(48)
TIM1
TIM0_CH3(48),TIM0_CH4(48),TIM0_CH5(48)
TIM1_CH3(48),TIM1_CH4(48),TIM1_CH5(48)
SPE1
FSOI1
TIM2
TIM2_CH0(48),TIM2_CH1(48),TIM2_CH2(48)
TIM2_CH3(48),TIM2_CH4(48),TIM2_CH5(48)
SPE2
FSOI2
TIM3
TIM3_CH0(48),TIM3_CH1(48),TIM3_CH2(48)
TIM3_CH3(48),TIM3_CH4(48),TIM3_CH5(48)
SPE3
FSOI3
TIM4
TIM4_CH0(48),TIM4_CH1(48),TIM4_CH2(48)
TIM4_CH3(48),TIM4_CH4(48),TIM4_CH5(48)
SPE4
FSOI4
TIM5
TIM5_CH0(48),TIM5_CH1(48),TIM5_CH2(48)
TIM5_CH3(48),TIM5_CH4(48),TIM5_CH5(48)
SPE5
FSOI5
0
1
2
3
4
5
6
7

The SPE submodule can determine a rotation direction out of the combined *TIM[i]_CHx(48)*, *TIM[i]_CHy(48)* and *TIM[i]_CHz(48)* signals. On this input signals a pattern match algorithm is applied to generate the *SPEx_DIR* signal on behalf of the temporal relation between these input patterns. A possible sample pattern of the three input signals is shown in figure 19.1.2. In general, the input pattern is programmable within the SPE submodule.

19.1.2 SPE Sample input pattern for *TIM[i]_CH[x,y,z](48)*



In figure 19.1.2 the input signals define the pattern from the input sensors which have a 50% high and 50% low phase. The pattern according to figure 19.1.2 is as follows:

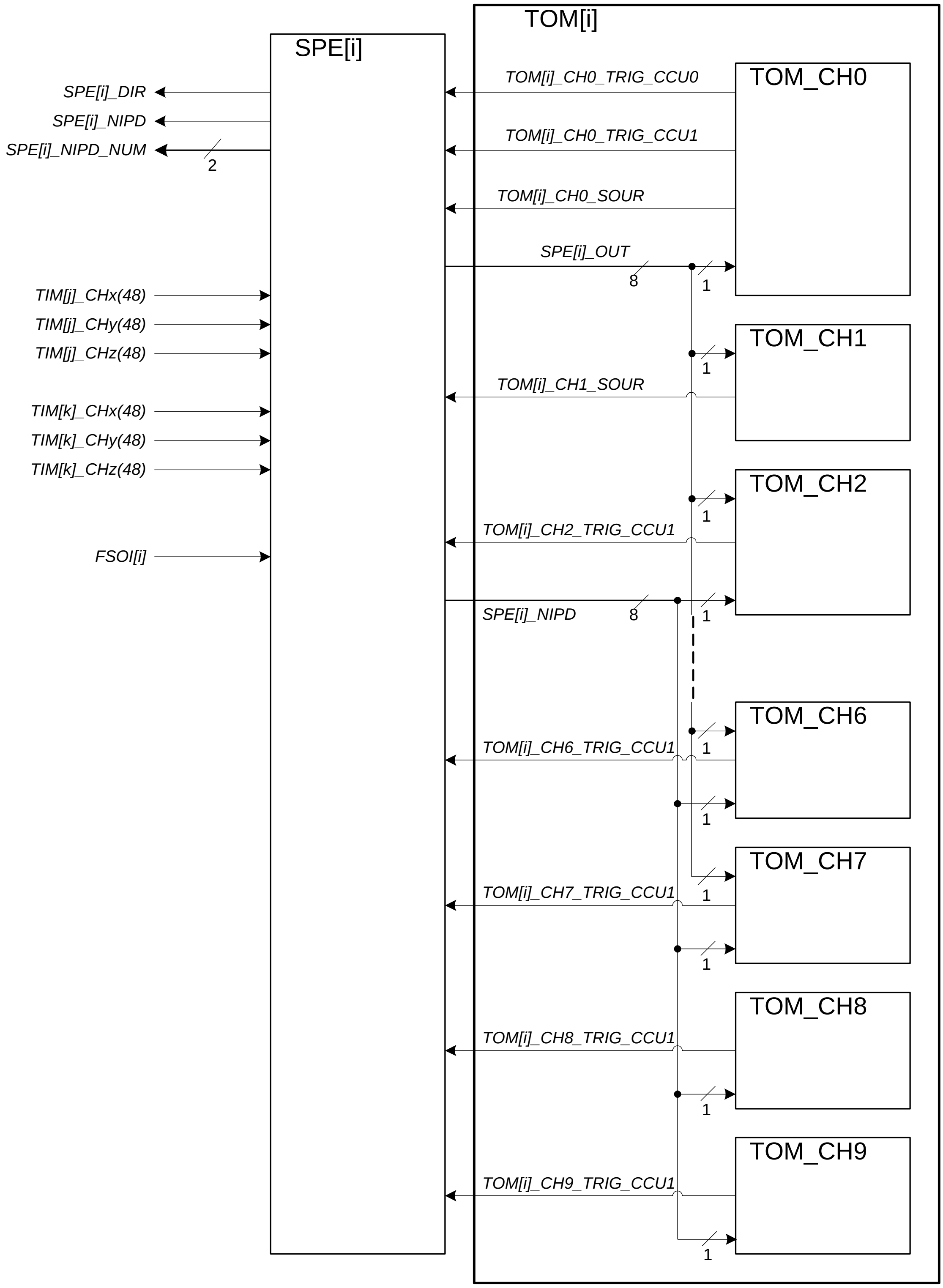
100 – 110 – 010 – 011 – 001 – 101 – 100
where the first bit (smallest circle) represents *TIM[i]_CH[x](48)*, the second bit represents *TIM[i]_CH[y](48)*, and the third bit (greatest circle) represents *TIM[i]_CH[z](48)*.

Note that the SPE module expects that with every new pattern only one of the three input signals changes its value.

19.2 SPE Submodule description

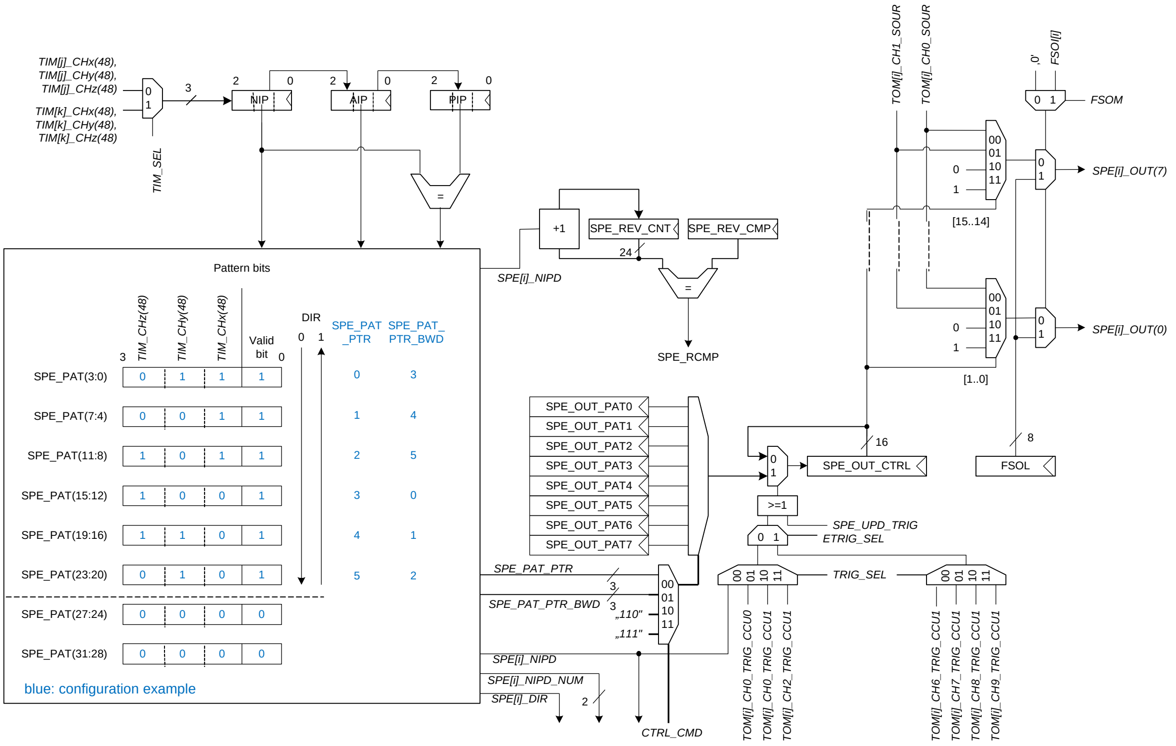
The SPE submodule can handle sensor pattern inputs. Every time if one of the input signals *TIM[i]_CH[x](48)*,*TIM[i]_CH[y](48) or TIM[i]_CH[z](48)* changes its value, a sample of all three input signals is made. Derived from the sample of the three inputs the encoded rotation direction and the validity of the input pattern sequence is determined and signaled. When a valid input pattern is detected, the SPE submodule can control the outputs of a dedicated connected TOM submodule. This connection is shown in figure 19.2.1.

19.2.1 SPE to TOM Connections



The *TOM[i]_CH0_TRIG_CCU[x]* and *TOM[i]_CH[x]_SOUR* signal lines are used to evaluate the current state of the TOM outputs, whereas the *SPE[i]_OUT* output vector is used to control the TOM output depending on the new input pattern. The *SPE[i]_OUT* output vector is defined inside the SPE submodule in a pattern definition table **SPE[i]_OUT_PAT[x]**. The internal SPE submodule architecture is shown in figure 19.2.2.

19.2.2 SPE Submodule architecture



	TIM_CHz(48)	TIM_CHy(48)	TIM_CHx(48)	Valid bit	SPE_PAT_PTR	SPE_PAT_PTR_BWD
SPE_PAT(3:0)	0	1	1	1	0	3
SPE_PAT(7:4)	0	0	1	1	1	4
SPE_PAT(11:8)	1	0	1	1	2	5
SPE_PAT(15:12)	1	0	0	1	3	0
SPE_PAT(19:16)	1	1	0	1	4	1
SPE_PAT(23:20)	0	1	0	1	5	2
SPE_PAT(27:24)	0	0	0	0		
SPE_PAT(31:28)	0	0	0	0		

blue: configuration example

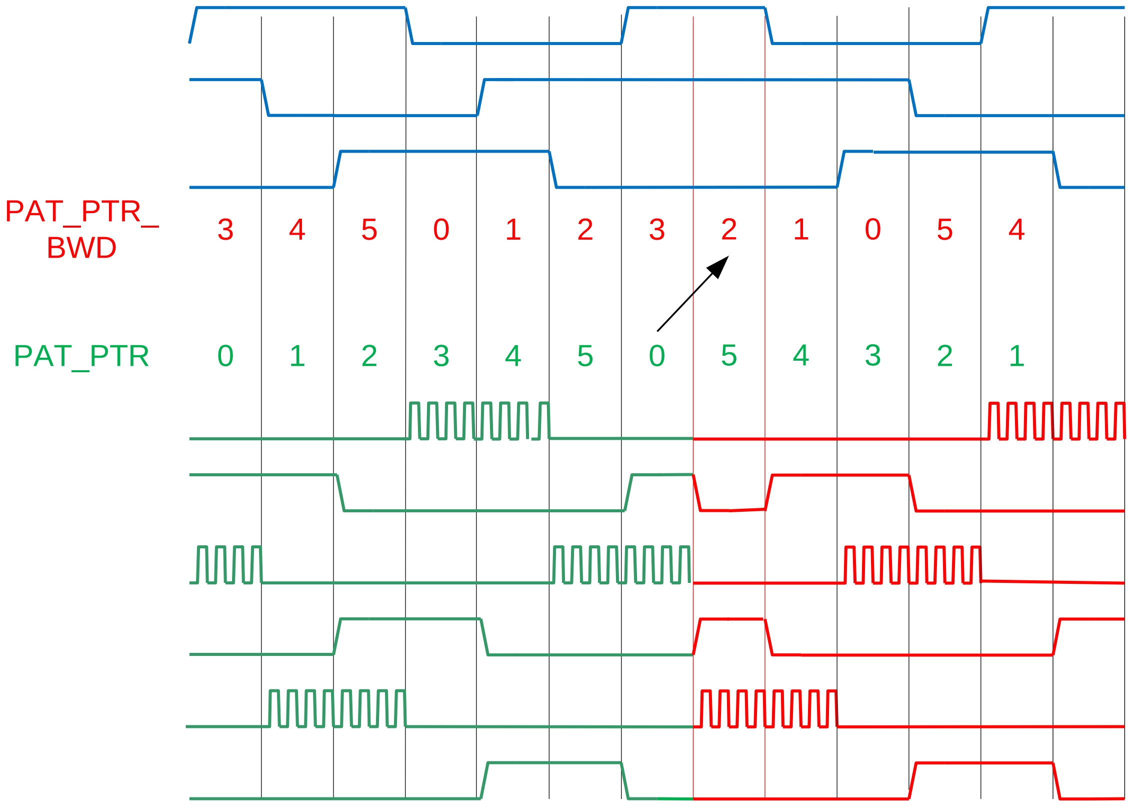
The **SPE[i]_PAT** register holds the valid input pattern for the three input patterns *TIM[i]_CH[x](48)*, *TIM[i]_CH[y](48)* and *TIM[i]_CH[z](48)*. The input pattern is programmable. The valid bit shows if the programmed pattern is a valid one. Figure 19.2.2 shows the programming of the **SPE[i]_PAT** register for the input pattern defined in figure 19.1.2.

The rotation direction is determined by the order of the valid input pattern. This rotation direction defines if the *SPE_PAT_PTR* is incremented (DIR = 0) or decremented (DIR = 1). Whenever a valid input pattern is detected, the *NIPD* signal is raised, the *SPE_PAT_PTR* is incremented/decremented and a new output control signal *SPE[i]_OUT(x)* is send to the corresponding TOM submodule.

To command directly the forward or backward rotation the SPE provides with **SPE[i]_APT_PTR** and **SPE[i]_PAT_PTR_BWD** two pointers to array **SPE[i]_OUT_PAT[z].** Both can point to different values of **SPE[i]_OUT_PAT[z]** at the same point in time. **SPE[i]_APT_PTR** is intended to point to the pattern for forward commanding and **SPE[i]_PAT_PTR_BWD** is intended to point to the pattern for backward commanding.
On startup both pointers have to be configured to an initial value that corresponds to different direction depending start pattern of **SPE[i]_OUT_PAT[z]**.

With each valid new input pattern indicated by *SPE_NIPD* both pointers will be incremented or decremented according to the detected direction.
Switching from command forward to command backward can then be done by changing the selected pointer to **SPE[i]_OUT_PAT[z]** array, i.e. changing **SPE_CTRL_CMD** in register **SPE[i]_CMD** from selecting **SPE[i]_PAT_PTR** to selecting **SPE[i]_PAT_PTR_BWD** or vice versa.
The intended behavior is depicted in the following figure:

19.2.3 SPE forward - backward commanding



With command **SPE_CTRL_CMD** = 0b10 or 0b11 a dedicated configurable output pattern configured to the pattern **SPE_OUT_PAT6** or **SPE_OUT_PAT7** can be commanded to the outputs.
An example is the introduction of a SW dead time if switching from pointer **SPE[i]_PAT_PTR** to **SPE[i]_PAT_PTR_BWD** or vice versa. E.g. if in **SPE[i]_OUT_PAT6** the value 0b10 for each output (i.e. set *SPE_OUT(n)* to 0) is programmed, this can be used as an intermediate step to introduce this 'all off' when switching between **SPE[i]_PAT_PTR** to **SPE[i]_PAT_PTR_BWD** or vice versa.

Selectable by **TRIG_SEL** and **ETRIG_SEL** the CCU1 trigger of either the TOM channel 2,6,7,8 and 9 can be used together with the SPE module to trigger a delayed update of the **SPE_OUT_CTRL** register after new input pattern detected by SPE (signaled by *SPE[i]_NIPD*).

To do this, the TOM channel z=2,6,7,8 or 9 has to be configured to work in one-shot mode (set bit **OSM** in register **TOM[i]_CH[z]_CTRL**). The SPE trigger of this channel has to be enabled, too (set description of bit **SPEM** and bit **SPE_TRIG** in register **TOM[i]_CH[z]_CTRL**). The SPE module has to be configured to update **SPE_OUT_CTRL** on *TOM[i]_CH[z]_TRIG_CCU1* (set in **SPE[i]_CTRL_STAT** bits **TRIG_SEL** to 0b11). Then, on new input detected by SPE, the signal *SPE[i]_NIPD* triggers the start of the TOM channel z to generates one PWM period by resetting **CN0** to 0. On second PWM edge triggered by CCU1 of TOM channel z, the signal *TOM[i]_CH[z]_TRIG_CCU1* triggers the update of **SPE_OUT_CTRL**.

The update of **SPE[i]_OUT_CTRL** with the content of one of the **SPE_OUT_PAT[z]** register can be triggered at any time by writing a 1 to bit **SPE_UPD_TRIG** in register **SPE[i]_CMD**.

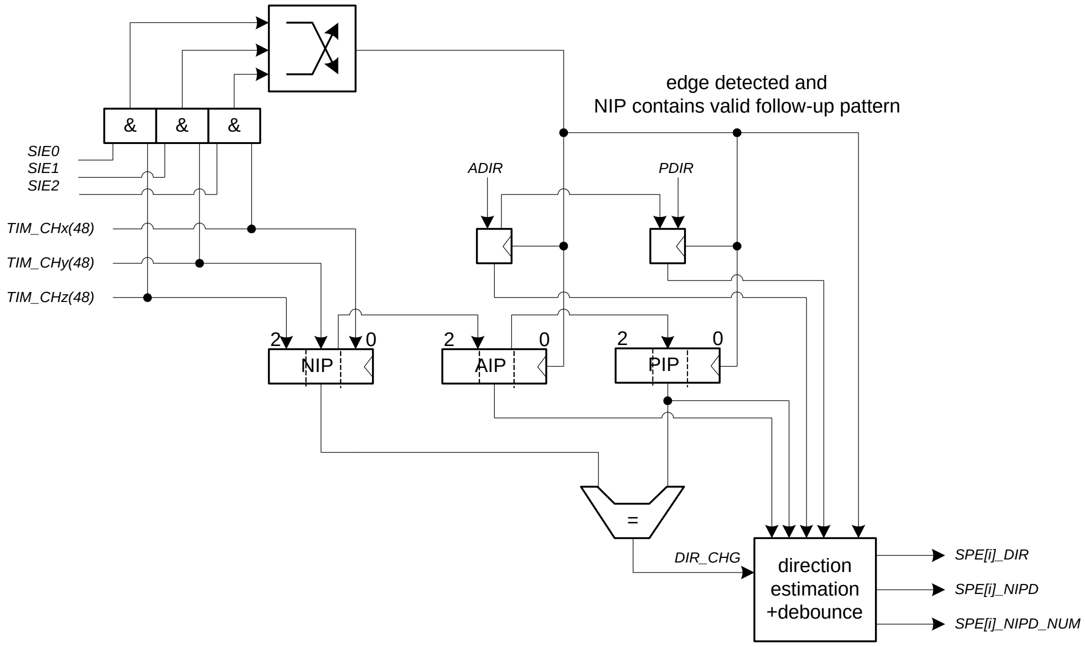
The regular trigger for update of **SPE[i]_OUT_CTRL** (commutation trigger) is selected by **TRIG_SEL** and **ETRIG_SEL**.

According to figure 19.2.2, the two input patterns 0b000 and 0b111 are not allowed combinations and will end in a *SPE[i]_PERR* interrupt. These two patterns can be used to determine a sensor input error. A *SPE[i]_PERR* interrupt will also be raised, if the input patterns occur in a wrong order, e.g. if the pattern 0b010 does not follow the pattern 0b110 or 0b011.

The register **SPE[i]_IN_PAT** bit field inside the **SPE[i]_CTRL_STAT** register is implemented, where the input pattern history is stored by the SPE submodule. The CPU can determine a broken sensor when the SPE[i]_PERR interrupt occurs by analyzing the bit pattern readable via bit field **NIP** inside the **SPE[i]_CTRL_STAT** register. The input pattern in the **SPE[i]_CTRL_STAT** register is updated whenever a valid edge is detected on one of the input lines *TIM[i]_CH[x](48)*, *TIM[i]_CH[y](48)* or *TIM[i]_CH[z](48)*. The pattern bit fields are then shifted. The input pattern history generation inside the **SPE[i]_CTRL_STAT** register is shown in figure 19.2.4.

Additionally to the sensor pattern evaluation the SPE module also provides the feature of fast shutoff for all TOM channels controlled by the SPE module. The feature is enabled by setting bit **FSOM** in register **SPE[i]_CTRL_STAT**. The fast shutoff level itself is defined in the bit field **FSOL** of register **SPE[i]_CTRL_STAT**. The TIM input used to trigger the fast shutoff is either TIM channel 6 or TIM channel 7 depending on the TIM instance connected to the SPE module. For details of connections please refer to figure 19.1.1.

19.2.4 SPE[i]_IN_PAT register representation



The CPU can disable one of the three input signals, e.g. when a broken input sensor was detected, by disabling the input with the three input enable bits **SIE** inside the **SPE[i]_CTRL_STAT** register.

Whenever at least one of the input signal *TIM[i]_CH[x](48)*, *TIM[i]_CH[y](48)* or *TIM[i]_CH[z](48)* changes the SPE submodule stores the new bit pattern in an internal register NIP (New Input Pattern). If the current input pattern in **NIP** is the same as in the Previous Input Pattern (**PIP**) the direction of the engine changed, the *SPE[i]_DCHG* interrupt is raised, the direction change is stored internally and the pattern in the **PIP** bit field is filled with the **AIP** bit field and the **AIP** bit field is filled with the **NIP** bit field. The **SPE[i]_DIR** bit inside the **SPE[i]_CTRL_STAT** register is toggled and the *SPE[i]_DIR* signal is changed.

If the SPE encounters that with the next input pattern detected new input pattern **NIP** the direction change again, the input signal is categorized as bouncing and the bouncing input signal interrupt *SPE[i]_BIS* is raised.

Immediately after update of register **NIP**, when the new detected input pattern doesn't match the **PIP** pattern (i.e. no direction change was detected), the SPE shifts the value of register **AIP** to register **PIP** and the value of register **NIP** to register **AIP**. The *SPE[i]_NIPD* interrupt is raised.

The number of the channel that has been changed and thus leads to the new input pattern is encoded in the signal *SPE[i]_NIPD_NUM*.

If a sensor error was detected, the CPU has to define upon the pattern in the **SPE[i]_CTRL_STAT** register, which input line comes from the broken sensor. The faulty signal line has to be masked by the CPU and the SPE submodule determines the rotation direction on behalf of the two remaining *TIM[i]_CH[x]* input lines.

The pattern history can be determined by the CPU by reading the two bit fields AIP and PIP of the **SPE[i]_CTRL_STAT** register. The **AIP** register field holds the actual detected input pattern at *TIM[i]_CH[x](48)*, *TIM[i]_CH[y](48)* and *TIM[i]_CH[z](48)* and the PIP holds the previous detected pattern.

After reset the register **NIP**, **AIP** and **PIP** as well as the register **SPE[i]_PAT_PTR** and **SPE[i]_OUT_CTRL** will not contain valid startup values which would allow correct behavior after enabling SPE and detecting the first input patterns.
Thus, it is necessary to initialize these register to correct values.
To do this, before enabling the SPE, the bit field **NIP** of register **SPE[i]_CTRL_STAT** can be read and depending on this value the initialization values for the register **AIP**, **PIP**, **SPT_PAT_PTR** and **SPE[i]_OUT_CTRL** can be determined.

19.2.5 SPE Revolution detection

The SPE submodule is able to detect and count the number of valid input patterns detected at the specified input ports. This is done with a 24 bit revolution counter **SPE_REV_CNT**. The counter is incremented by a value of one (1) when a new valid input pattern indicating forward direction is detected. The counter is decremented by a value of one (1) when a new valid input pattern indicating backward direction is detected.

In addition there exists a 24 bit **SPE_REV_CMP** register. The user can initialize this register with a compare value, where an interrupt *SPE[i]_RCMP* is raised, when the revolution counter equals the compare value either in forward or backward direction.

Both register may be written by software at any time.

19.3 SPE Interrupt signals

Signal	Description
SPE[i]_NIPD	SPE New valid input pattern detected.
SPE[i]_DCHG	SPE Rotation direction change detected on behalf of input pattern.
SPE[i]_PERR	SPE Invalid input pattern detected.
SPE[i]_BIS	SPE Bouncing input signal detected at input.
SPE[i]_RCMP	SPE Revolution counter compare value reached.

19.4 SPE Register overview

Register name	Description	Details in Section
SPE[i]_CTRL_STAT	SPEi Control status register	19.5.1
SPE[i]_PAT	SPEi Input pattern definition register.	19.5.2
SPE[i]_OUT_PAT[z] (z:0…7)	SPEi Output definition register.	19.5.3
SPE[i]_OUT_CTRL	SPEi output control register	19.5.4
SPE[i]_REV_CNT	SPEi input revolution counter	19.5.5
SPE[i]_REV_CMP	SPEi Revolution counter compare value	19.5.6
SPE[i]_IRQ_NOTIFY	SPEi Interrupt notification register.	19.5.7
SPE[i]_IRQ_EN	SPEi Interrupt enable register.	19.5.8
SPE[i]_EIRQ_EN	SPEi Error interrupt enable register.	19.5.11
SPE[i]_IRQ_FORCINT	SPEi Interrupt generation by software.	19.5.9
SPE[i]_IRQ_MODE	SPEi Interrupt mode configuration register	19.5.10
SPE[i]_CTRL_STAT2	SPEi Control status register 2	19.5.12
SPE[i]_CMD	SPEi Command register	19.5.13

19.5 SPE Register description

19.5.1 Register SPE[i]_CTRL_STAT

Address Offset:	see Appendix B	Initial Value:	0x0000_0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	FSOL								ETRIG_SEL	NIP			PDIR	PIP			ADIR	AIP			Reserved	SPE_PAT_PTR			FSOM	TIM_SEL	TRIG_SEL		SIE2	SIE1	SIE0	EN
Mode	RW								RW	R			RW	RW			RW	RW			R	RW			RW	RW	RW		RW	RW	RW	RW
Initial Value	0x00								0	0b000			0	0b000			0	0b000			0	0b000			0	0	0b00		0	0	0	0

Bit 0 **SPE_EN:** SPE Submodule enable.
0 = SPE disabled.

1 = SPE enabled.

Bit 1 **SIE0:** SPE Input enable for TIM_CHx(48).
0 = SPE Input is disabled.
1 = SPE Input is enabled.
Note: When the input is disabled, a 0 signal is sampled for this input. However, the bit field NIP of this register shows the true value of the input signal.

Bit 2 **SIE1:** SPE Input enable for TIM_CHy(48).
See bit 1.

Bit 3 **SIE2:** SPE Input enable for TIM_CHz(48).
See bit 1.

Bit 5:4 **TRIG_SEL:** Select trigger input signal.
ETRIG_SEL = 0 / ETRIG_SEL = 1
0b00 = *SPE[i]_NIPD* selected / *TOM_CH6_TRIG_CCU1* selected
0b01 = *TOM_CH0_TRIG_CCU0* selected / *TOM_CH7_TRIG_CCU1* selected
0b10 = *TOM_CH0_TRIG_CCU1* selected / *TOM_CH8_TRIG_CCU1* selected
0b11 = *TOM_CH2_TRIG_CCU1* selected / *TOM_CH9_TRIG_CCU1* selected
Note: In case of ETRIG_SEL=1, according to selected *TOM_CH[x]_TRIG_CCU1* signal the configuration bits SPE_TRIG and OSM of register TOM_CH[x]_CTRL have to be set in same TOM channel x to enable the trigger signal generation in one-shot mode.

Bit 6 **TIM_SEL:** select TIM input signal
The GTM supports up to 6 SPE modules.
SPE0:
0 = TIM0_CH0..2
1 = TIM1_CH0..2
SPE1:
0 = TIM0_CH3..5
1 = TIM1_CH3..5
SPE2:
0 = TIM2_CH0..2
1 = TIM3_CH0..2
SPE3:
0 = TIM2_CH3..5
1 = TIM3_CH3..5
SPE4:
0 = TIM4_CH0..2
1 = TIM5_CH0..2
SPE5:
0 = TIM4_CH3..5
1 = TIM5_CH3..5

Bit 7 **FSOM:** Fast Shutoff Mode
0 = Fast Shutoff mode disabled
1 = Fast Shutoff mode enabled

Bit 10:8 **SPE_PAT_PTR:** Pattern selector for TOM output signals.
Actual index into the SPE[i]_OUT_PAT[x] register table.
Each register SPE[i]_OUT_PAT[x] is fixed assigned to one bit field IPx_PAT of register SPE[i]_PAT. Thus, the pointer SPE[i]_PAT_PTR represents an index to the selected SPE[i]_OUT_PAT[x] register as well as the actual detected input pattern IPx_PAT.
0b000 = SPE[i]_OUT_PAT0 selected

Bit 11 **Reserved:**
Note: Read as zero, should be written as zero

Bit 14:12 **AIP:** Actual input pattern that was detected by a regular input pattern change.

Bit 15 **ADIR:** Actual rotation direction.
0 = Rotation direction is 0 according to SPE[i]_PAT register.
1 = Rotation direction is 1 according to SPE[i]_PAT register.

Bit 18:16 **PIP:** Previous input pattern that was detected by a regular input pattern change.

Bit 19 **PDIR:** Previous rotation direction.
0 = Rotation direction is 0 according to SPE[i]_PAT register.
1 = Rotation direction is 1 according to SPE[i]_PAT register.

Bit 22:20 **NIP:** New input pattern that was detected.
Note: This bit field mirrors the new input pattern. SPE internal functionality is triggered on each change of this bit field.

Bit 23 **ETRIG_SEL: extended trigger selection**
ETRIG_SEL = 0 / ETRIG_SEL = 1 :
0b00 = *SPE[i]_NIPD* selected / *TOM_CH6_TRIG_CCU1* selected
0b01 = *TOM_CH0_TRIG_CCU0* selected / *TOM_CH7_TRIG_CCU1* selected
0b10 = *TOM_CH0_TRIG_CCU1* selected / *TOM_CH8_TRIG_CCU1* selected
0b11 = *TOM_CH2_TRIG_CCU1* selected / *TOM_CH9_TRIG_CCU1* selected

Bit 31:24 **FSOL:** Fast Shutoff Level for TOM[i] channel 0 to 7

19.5.2 Register SPE[i]_PAT

Address Offset:	see Appendix B								Initial Value: 0x0000_0000							
	31 30 29	28	27 26 25	24	23 22 21	20	19 18 17	16	15 14 13	12	11 10 9	8	7 6 5	4	3 2 1	0
Bit	IP7_PAT	IP7_VAL	IP6_PAT	IP6_VAL	IP5_PAT	IP5_VAL	IP4_PAT	IP4_VAL	IP3_PAT	IP3_VAL	IP2_PAT	IP2_VAL	IP1_PAT	IP1_VAL	IP0_PAT	IP0_VAL
Mode	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Initial Value	0b000	0	0b000	0	0b000	0	0b000	0	0b000	0	0b000	0	0b000	0	0b000	0

Bit 0 **IP0_VAL:** Input pattern 0 is a valid pattern.
0 = Pattern invalid.
1 = Pattern valid.

Bit 3:1 **IP0_PAT:** Input pattern 0.
Bit field defines the first input pattern of the SPE input signals.
Bit 1 defines the TIM[i]_CHx(48) input signal.
Bit 2 defines the TIM[i]_CHy(48) input signal.
Bit 3 defines the TIM[i]_CHz(48) input signal.

Bit 4 **IP1_VAL:** Input pattern 1 is a valid pattern.
See bit 0.

Bit 7:5 **IP1_PAT:** Input pattern 1.
See bits 3:1.

Bit 8 **IP2_VAL:** Input pattern 2 is a valid pattern.
See bit 0.

Bit 11:9 **IP2_PAT:** Input pattern 2.
See bits 3:1.

Bit 12 **IP3_VAL:** Input pattern 3 is a valid pattern.
See bit 0.

Bit 15:13 **IP3_PAT:** Input pattern 3.
See bits 3:1.

Bit 16 **IP4_VAL:** Input pattern 4 is a valid pattern
See bit 0.

Bit 19:17 **IP4_PAT:** Input pattern 4.
See bits 3:1.

Bit 20 **IP5_VAL:** Input pattern 5 is a valid pattern
See bit 0.

Bit 23:21 **IP5_PAT:** Input pattern 5.
See bits 3:1.

Bit 24 **IP6_VAL:** Input pattern 6 is a valid pattern
See bit 0.

Bit 27:25 **IP6_PAT:** Input pattern 6.
See bits 3:1.

Bit 28 **IP7_VAL:** Input pattern 7 is a valid pattern

See bit 0.

Bit 31:29 **IP7_PAT:** Input pattern 7.
See bits 3:1.

Note: Only the first block of valid input patterns defines the commutation. All input pattern following the first marked invalid input pattern are ignored.

19.5.3 Register SPE[i]_OUT_PAT[z] (z:0...7)

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_0000**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																SPE_OUT_PAT															
Mode	R																RW															
Initial Value	0x0000																0x0000															

Bit 15:0 **SPE_OUT_PAT:** SPE output control value for TOM_CH0 to TOM_CH7
SPE_OUT_PAT[n+1:n] defines output select signal of TOM[i]_CH[n]
0b00 = set *SPE_OUT(n)* to *TOM_CH0_SOUR*
0b01 = set *SPE_OUT(n)* to *TOM_CH1_SOUR*
0b10 = set *SPE_OUT(n)* to *0*
0b11 = set *SPE_OUT(n)* to *1*
with n:0...7

Bit 31:16 **Reserved:**
Note: Read as zero, should be written as zero

Note: Register SPE_OUT_PAT[x] defines the output selection for TOM[i]_CH0 to TOM[i]_CH7 depending on actual input pattern IP[z]_PAT with z:0...7.

19.5.4 Register SPE[i]_OUT_CTRL

Address Offset:	see Appendix B																Initial Value: 0x0000_0000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																SPE_OUT_CTRL															
Mode	R																RW															
Initial Value	0x0000																0x0000															

Bit 15:0 **SPE_OUT_CTRL:** SPE output control value for TOM_CH0 to TOM_CH7
SPE_OUT_CTRL[n+1:n] defines output select signal of TOM_CHn
0b00 = set *SPE_OUT(n)* to *TOM_CH0_SOUR*
0b01 = set *SPE_OUT(n)* to *TOM_CH1_SOUR*
0b10 = set *SPE_OUT(n)* to *0*
0b11 = set *SPE_OUT(n)* to *1*
with n:0...7

Note: Current output control selection for SPE[i]_OUT(0..7).

Bit 31:16 **Reserved:**
Note: Read as zero, should be written as zero

19.5.5 Register SPE[i]_REV_CNT

Address Offset:	see Appendix B																Initial Value: 0x0000_0000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								REV_CNT																							
Mode	R								RW																							
Initial Value	0x0000								0x000000																							

Bit 23:0 **REV_CNT:** Input signal revolution counter
The counter is running if SPE module is enabled (bit SPE_EN).
REV_CNT is incrementing if SPE_PAT_PTR is incrementing
REV_CNT is decrementing if SPE_PAT_PTR is decrementing

Bit 31:24 **Reserved:**
Note: Read as zero, should be written as zero

19.5.6 Register SPE[i]_REV_CMP

Address Offset:	see Appendix B																Initial Value: 0x0000_0000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								REV_CMP																							
Mode	R								RW																							
Initial Value	0x0000								0x0000 00																							

Bit 23:0 **REV_CMP:** Input signal revolution counter compare value
The interrupt *SPE[i]_RCMP* is raised when the SPE[i]_REV_CNT value equals the SPE[i]_REV_CMP register. It should be noted that *SPE[i]_RCMP* is only raised if an incrementation or decrementation of SPE[i]_REV_CNT is applied, due to an input signal change. Any update of SPE[i]_REV_CNT or SPE[i]_REV_CMP via AEI does not raise a *SPE[i]_RCMP* interrupt.

Bit 31:24 **Reserved:**
Note: Read as zero, should be written as zero

19.5.7 Register SPE[i]_IRQ_NOTIFY

Address Offset:	see Appendix B																Initial Value:							0x0000_0000								
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																											SPE_RCMP	SPE_BIS	SPE_PERR	SPE_DCHG	SPE_NIPD
Mode	R																											RCw	RCw	RCw	RCw	RCw
Initial Value	0x0000 000																											0	0	0	0	0

Bit 0 **SPE_NIPD:** New input pattern interrupt occurred.
0 = No interrupt occurred.
1 = New input pattern detected interrupt occurred.
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 1 **SPE_DCHG:** SPE_DIR bit changed on behalf of new input pattern.
See bit 0.

Bit 2 **SPE_PERR:** Wrong or invalid pattern detected at input.
See bit 0.

Bit 3 **SPE_BIS:** Bouncing input signal detected.
See bit 0.

Bit 4 **SPE_RCMP:** SPE revolution counter match event.
See bit 0.

Bit 31:5 **Reserved:**
Note: Read as zero, should be written as zero

19.5.8 Register SPE[i]_IRQ_EN

Address Offset:	see Appendix B																Initial Value:							0x0000_0000								
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																											SPE_RCMP_IRQ_	SPE_BIS_IRQ_EN	SPE_PERR_IRQ_	SPE_DCHG_IRQ_	SPE_NIPD_IRQ_E
Mode	R																											RW	RW	RW	RW	RW
Initial Value	0x0000 000																											0	0	0	0	0

Bit 0 **SPE_NIPD_IRQ_EN:** *SPE_NIPD_IRQ* interrupt enable.

0 = Disable interrupt, interrupt is not visible outside GTM-IP.
1 = Enable interrupt, interrupt is visible outside GTM-IP.

Bit 1 **SPE_DCHG_IRQ_EN:** *SPE_DCHG_IRQ* interrupt enable.
See bit 0.

Bit 2 **SPE_PERR_IRQ_EN:** *SPE_PERR_IRQ* interrupt enable.
See bit 0.

Bit 3 **SPE_BIS_IRQ_EN:** *SPE_BIS_IRQ* interrupt enable.
See bit 0.

Bit 4 **SPE_RCMP_IRQ_EN:** *SPE_RCMP_IRQ* interrupt enable.
See bit 0.

Bit 31:5 **Reserved:**
Note: Read as zero, should be written as zero

19.5.9 Register SPE[i]_IRQ_FORCINT

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																											TRG_SPE_RCMP	TRG_SPE_BIS	TRG_SPE_PERR	TRG_SPE_DCHG	TRG_SPE_NIPD
Mode	R																											RAw	RAw	RAw	RAw	RAw
Initial Value	0x0000000																											0	0	0	0	0

Bit 0 **TRG_SPE_NIPD:** Force interrupt of *SPE_NIPD*.
0 = Corresponding bit in status register will not be forced.
1 = Assert corresponding field in **SPE_IRQ_NOTIFY** register.

Note: This bit is cleared automatically after interrupt is released
Note: This bit is write protected by bit RF_PROT of register GTM_CTRL

Bit 1 **TRG_SPE_DCHG:** Force interrupt of *SPE_DCHG*.
See bit 0.

Bit 2 **TRG_SPE_PERR:** Force interrupt of *SPE_PERR*.
See bit 0.

Bit 3 **TRG_SPE_BIS:** Force interrupt of *SPE_BIS*.
See bit 0.

Bit 4 **TRG_SPE_RCMP:** Force interrupt of *SPE_RCMP*.
See bit 0.

Bit 31:5 **Reserved:**
Note: Read as zero, should be written as zero

19.5.10 Register SPE[i]_IRQ_MODE

Address Offset:	see Appendix B																Initial Value: 0x0000_000X															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																														IRQ_MODE	
Mode	R																														RW	
Initial Value	0x0000 0000																														XX	

Bit 1:0 **IRQ_MODE**: IRQ mode selection
0b00 = Level mode
0b01 = Pulse mode
0b10 = Pulse-Notify mode
0b11 = Single-Pulse mode
Note: The interrupt modes are described in section 2.5.

Bit 31:2 **Reserved**
Note: Read as zero, should be written as zero

19.5.11 Register SPE[i]_EIRQ_EN

Address Offset:	see Appendix B																Initial Value: 0x0000_0000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																											SPE_RCMP_EIRQ	SPE_BIS_EIRQ_E	SPE_PERR_EIRQ	SPE_DCHG_EIRQ	SPE_NIPD_EIRQ_
Mode	R																											RW	RW	RW	RW	RW
Initial Value	0x0000 000																											0	0	0	0	0

Bit 0 **SPE_NIPD_EIRQ_EN:** *SPE_NIPD_EIRQ* interrupt enable.
0 = Disable error interrupt, error interrupt is not visible outside GTM-IP.
1 = Enable error interrupt, error interrupt is visible outside GTM-IP.

Bit 1 **SPE_DCHG_EIRQ_EN:** *SPE_DCHG_EIRQ* error interrupt enable.
See bit 0.

Bit 2 **SPE_PERR_EIRQ_EN:** *SPE_PERR_EIRQ* error interrupt enable.
See bit 0.

Bit 3 **SPE_BIS_EIRQ_EN:** *SPE_BIS_EIRQ* error interrupt enable.
See bit 0.

Bit 4 **SPE_RCMP_EIRQ_EN:** *SPE_RCMP_EIRQ* error interrupt enable.
See bit 0.

Bit 31:5 **Reserved:**
Note: Read as zero, should be written as zero

19.5.12 Register SPE[i]_CTRL_STAT2

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_0000**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																					SPE_PAT_PTR_BWD			Reserved							
Mode	R																					RW			R							
Initial Value	0x0000 0																					0b000			0x00							

Bit 7:0 **Reserved:**
Note: Read as zero, should be written as zero

Bit 10:8 **SPE_PAT_PTR_BWD:** Pattern selector for TOM output signals in case of SPE_CTRL_CMD = 0b01 (e.g. backward direction).

Index into the SPE[i]_OUT_PAT[z] register table in case of SPE_CTRL_CMD = 0b01 which may be used for backward direction.

Each register SPE[i]_OUT_PAT[x] is fixed assigned to one bit field IPx_PAT of register SPE[i]_PAT. Thus, the pointer SPE[i]_PAT_PTR_BWD represents an index to the selected SPE[i]_OUT_PAT[x] register as well as the actual detected input pattern IPx_PAT.

The index pointer SPE_PAT_PTR_BWD is used if SPE_CTRL_CMD = 0b01.

The index pointer SPE_PAT_PTR is used if SPE_CTRL_CMD = 0b00 (by default).

0b000 = SPE[i]_OUT_PAT0 selected

Bit 31:11 **Reserved:**
Note: Read as zero, should be written as zero

19.5.13 Register SPE[i]_CMD

Address Offset:	see Appendix B																Initial Value: 0x0000_0000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved															SPE_UPD_TRIG	Reserved														SPE_CTRL_CMD	
Mode	R															RW	R														RW	
Initial Value	0x0000															0	0x0000														0b00	

Bit 1:0 **SPE_CTRL_CMD:** SPE control command
0b00 = use SPE_PAT_PTR as an index pointer to select SPE[i]_OUT_PAT[z]
0b01 = use SPE_PAT_PTR_BWD as an index pointer to select SPE[i]_OUT_PAT[z]
0b10 = select SPE[i]_OUT_PAT6
0b11 = select SPE[i]_OUT_PAT7

Note: on switch between 0b00 and 0b01 the direction flag will be set according to used pointer.

Bit 15:2 **Reserved:**
Note: Read as zero, should be written as zero

Bit 16 **SPE_UPD_TRIG:** SPE updater trigger
1 = trigger update of SPE_OUT_CTRL with register selected by CTR_CMD multiplexer.
Note: This bit is automatically reset to 0.

Bit 31:17 **Reserved:**
Note: Read as zero, should be written as zero

20 Interrupt Concentrator Module (ICM)

20.1 Overview

The Interrupt Concentrator Module (ICM) is used to bundle the GTM-IP interrupt lines of the individual sub-modules in a reasonable manner into interrupt groups. By this bundling a smaller amount of interrupt lines is visible at the outside of the GTM-IP.

The individual interrupts of the GTM-IP sub-modules and channels have to be enabled or disabled inside the sub-modules and channels.

The feed through architecture of bundled interrupt lines is used for the sub-modules AEI, ARU, BRC, CMP, SPE, PSM, TIM, DPLL, TOM, ATOM and MCS.

To determine the detailed interrupt source the microcontroller has to read the sub-module/channel interrupt notification register **NOTIFY** and serve the channel individual interrupt.

Please note, that the interrupts are only visible inside the ICM and in consequence outside of the GTM-IP, when the interrupt is enabled inside the sub-modules themselves.

20.2 Bundling

The GTM-IP sub-module individual interrupt sources are connected to the ICM. There, the individual interrupt lines are either feed through and signaled to the outside world or bundled a second time into groups and are then signaled to the outside world.
The ICM interrupt bundling is described in the following sections.

20.2.1 GTM Infrastructure Interrupt Bundling

The first interrupt group contains interrupts of the infrastructure and safety components of the GTM. This interrupt group includes therefore interrupt lines coming from the AEI, ARU, BRC, PSM, SPE and CMP sub-modules. In this interrupt group each individual channel of the sub-modules has its own interrupt line to the outside world.

Thus, the active interrupt line can be used by the CPU to determine the GTM-IP sub-module channel that raised the interrupt. The interrupts are also represented in the **ICM_IRQG_0** register. This register is typically not read by the CPU, but it is readable.

In addition the interrupt line status for 8 channels of each FIFO are shown in the **ICM_IRQG_PSM_0_CI** register. Typically, the interrupt source is determined by the corresponding interrupt line and the **ICM_IRQG_PSM_0_CI** register are typically not read out by the CPU, but they are readable.

In addition the interrupt line status for each SPE are shown in the **ICM_IRQG_SPE_CI** register. Typically, the interrupt source is determined by the corresponding interrupt line and the **ICM_IRQG_SPE_CI** register are typically not read out by the CPU, but they are readable.

20.2.2 DPLL Interrupt Bundling

The DPLL Interrupt group handles the interrupts coming from the DPLL sub-module of the GTM-IP. Each of the individual DPLL interrupt lines has its own dedicated interrupt line to the outside world. The interrupts are additionally identified in the **ICM_IRQG_1** interrupt group register. This register is typically not read out by the CPU, but it is readable.

20.2.3 TIM Interrupt Bundling

Inside this group sub-modules which handle GTM-IP input signals are treated. This is the case for the TIM[i] sub-modules. Each TIM sub-module channel is able to generate six (6) individual interrupts if enabled inside the TIM channel. This six interrupts are bundled into one interrupt per TIM channel connected to the ICM.

The ICM does no further bundling. Thus, for the GTM-IP 32 interrupt lines *TIM[i]_IRQ*[y] are provided for the external microcontroller. The channel responsible for the interrupt can be determined by the raised interrupt line.

In addition, the **ICM_IRQG_2** and **ICM_IRQG_3** registers are mirrors for the TIM sub-module channel interrupts and typically not read out by the CPU, but it is readable.

20.2.4 MCS Interrupt Bundling

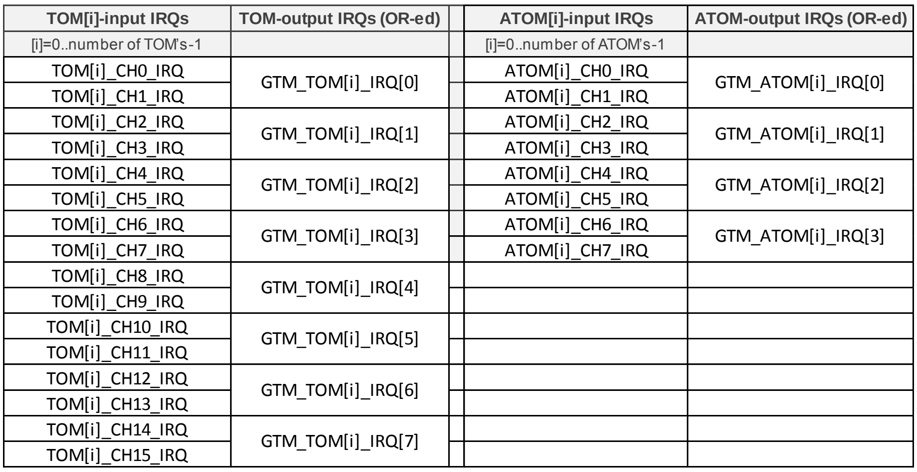
For complex signal output generation, the MCS sub-modules are used inside the GTM-IP. Each of these MCS sub-modules could have 8 channels with one interrupt line. This interrupt line is connected to the ICM sub-module and is feed through directly to the outside world.

In addition the interrupt line status for the first 8 channels of each MCS is shown in the **ICM_IRQG_4** and **ICM_IRQG_5** register. The interrupt line status for all used channels of each MCS are shown in the **ICM_IRQG_MCS[i]_CI** register. Typically, the interrupt source is determined by the corresponding interrupt line and the **ICM_IRQ4(/_5)** and **ICM_IRQG_MCS[i]_CI** register are typically not read out by the CPU, but they are readable.

20.2.5 TOM and ATOM Interrupt Bundling

For the TOM and ATOM sub-modules, the interrupts are bundled within the ICM sub-module a second time to reduce external interrupt lines. The interrupts are ORed in a manner that one GTM-IP external interrupt line represents two adjacent TOM or ATOM channel interrupts. For TOM[i] and ATOM[i] the bundling is shown in table 20.2.5.1.

20.2.5.1 TOM and ATOM interrupt bundling within ICM



TOM[i]-input IRQs	TOM-output IRQs (OR-ed)	ATOM[i]-input IRQs	ATOM-output IRQs (OR-ed)
[i]=0..number of TOM's-1		[i]=0..number of ATOM's-1	
TOM[i]_CH0_IRQ	GTM_TOM[i]_IRQ[0]	ATOM[i]_CH0_IRQ	GTM_ATOM[i]_IRQ[0]
TOM[i]_CH1_IRQ		ATOM[i]_CH1_IRQ	
TOM[i]_CH2_IRQ	GTM_TOM[i]_IRQ[1]	ATOM[i]_CH2_IRQ	GTM_ATOM[i]_IRQ[1]
TOM[i]_CH3_IRQ		ATOM[i]_CH3_IRQ	
TOM[i]_CH4_IRQ	GTM_TOM[i]_IRQ[2]	ATOM[i]_CH4_IRQ	GTM_ATOM[i]_IRQ[2]
TOM[i]_CH5_IRQ		ATOM[i]_CH5_IRQ	
TOM[i]_CH6_IRQ	GTM_TOM[i]_IRQ[3]	ATOM[i]_CH6_IRQ	GTM_ATOM[i]_IRQ[3]
TOM[i]_CH7_IRQ		ATOM[i]_CH7_IRQ	
TOM[i]_CH8_IRQ	GTM_TOM[i]_IRQ[4]		
TOM[i]_CH9_IRQ			
TOM[i]_CH10_IRQ	GTM_TOM[i]_IRQ[5]		
TOM[i]_CH11_IRQ			
TOM[i]_CH12_IRQ	GTM_TOM[i]_IRQ[6]		
TOM[i]_CH13_IRQ			
TOM[i]_CH14_IRQ	GTM_TOM[i]_IRQ[7]		
TOM[i]_CH15_IRQ			

The interrupts coming from the TOM[i] sub-modules are registered in the **ICM_IRQG_6 / ICM_IRQG_7 / ICM_IRQG_8** register. Always two TOM's are bundled in one ICM register, TOM0 and TOM1 are bundled in **ICM_IRQG_6**. To identify the TOM sub-module channel where the interrupt occurred, the CPU has to read out the **ICM_IRQG_6(/_7/_8)** register first before it goes to the TOM sub-module channel itself.

The **ICM_IRQG_6(/_7/_8)** register bits are cleared automatically, when their corresponding interrupt in the sub-module channels is cleared.

The interrupts coming from the ATOM[i] sub-modules are registered in the **ICM_IRQG_9** / **ICM_IRQG_10 /ICM_IRQG_11** register. Always four ATOM's are bundled in one ICM register. ATOM0, ATOM1, ATOM2 and ATOM3 are bundled in **ICM_IRQG_9.** To identify the ATOM sub-module channel where the interrupt occurred, the CPU has to read out the **ICM_IRQG_9(/_10/_11)** register first before it goes to the ATOM sub-module channel itself.

The **ICM_IRQG_9(/_10/_11)** register bits are cleared automatically, when their corresponding interrupt in the sub-module channels is cleared.

In addition the interrupt line status of two 16 channels TOM are shown in each **ICM_IRQG_TOM_[k]_CI (k:0..2)** register, TOM0 and TOM1 are bundled in **ICM_IRQG_TOM_0_CI**. Typically, the interrupt source is determined by the corresponding interrupt line and the **ICM_IRQG_TOM_[k]_CI** register are typically not read out by the CPU, but they are readable.

In addition the interrupt line status of four 8 channels ATOM are shown in each **ICM_IRQG_ATOM_[k]_CI (k:0..2)** register, ATOM0, ATOM1, ATOM2 and ATOM3 are bundled in **ICM_IRQG_ATOM_0_CI**. Typically, the interrupt source is determined by the corresponding interrupt line and the **ICM_IRQG_ATOM_[k]_CI** register are typically not read out by the CPU, but they are readable.

20.2.6 Module Error Interrupt Bundling

The Module Error Interrupt group handles the error interrupts coming from the BRC, FIFO, TIM, MCS, SPE, CMP, DPLL sub-module of the GTM-IP. The Module Error interrupts are additionally identified in the **ICM_IRQG_MEI** error interrupt group register. This register is typically not read out by the CPU, but it is readable.

In addition the error interrupt line status for each SPE are shown in the **ICM_IRQG_SPE_CEI** register. Typically, the error interrupt source is determined by the corresponding interrupt line and the **ICM_IRQG_SPE_CEI** register are typically not read out by the CPU, but they are readable.

20.2.7 FIFO Channel Error Interrupt Bundling

The FIFO Channel Error Interrupt group handles the error interrupts coming from the FIFO channel of the GTM-IP. The FIFO Channel Error interrupts are additionally identified in the **ICM_IRQG_CEI0** error interrupt group register. This register is typically not read out by the CPU, but it is readable.

The **ICM_IRQG_CEI0** register bits are cleared automatically, when their corresponding error interrupt in the sub-module channel is cleared.

In addition the error interrupt line status for 8 channels of each FIFO are shown in the **ICM_IRQG_PSM_0_CEI** register. Typically, the error interrupt source is determined by the corresponding interrupt line and the **ICM_IRQG_PSM_0_CEI** register are typically not read out by the CPU, but they are readable.

20.2.8 TIM Channel Error Interrupt Bundling

The TIM Channel Error Interrupt group handles the error interrupts coming from the TIM channel of the GTM-IP. The TIM Channel Error interrupts are additionally identified for the sub-modules TIM0, TIM1, TIM2 and TIM3 in the **ICM_IRQG_CEI1** error interrupt group register and for the sub-modules TIM4, TIM5 and TIM6 in the **ICM_IRQG_CEI2** error interrupt group register. These register are typically not read out by the CPU, but they are readable.

The **ICM_IRQG_CEI1** and **ICM_IRQG_CEI2** register bits are cleared automatically, when their corresponding error interrupt in the sub-module channel is cleared.

20.2.9 MCS Channel Error Interrupt Bundling

The MCS Channel Error Interrupt group handles the error interrupts coming from the MCS channel of the GTM-IP. All used 8 MCS Channel Error interrupts are additionally identified for each sub-modules MCS[i] in the **ICM_IRQG_MCS[i]_CEI** error interrupt group register. The first 8 MCS Channel Error interrupts are additionally identified for the sub-modules MCS0, MCS1, MCS2 and MCS3 in the **ICM_IRQG_CEI3** error interrupt group register and for the sub-modules MCS4, MCS5, MCS6 and MCS7 in the **ICM_IRQG_CEI4** error interrupt group register. These register are typically not read out by the CPU, but they are readable.

The **ICM_IRQG_MCS[i]_CEI**, **ICM_IRQG_CEI3** and **ICM_IRQG_CEI4** register bits are cleared automatically, when their corresponding error interrupt in the sub-module channel is cleared.

20.2.10 Error Interrupt Cluster Bundling

The Error Interrupt lines of up to 4 clusters are bundled in each **ICM_IRQG_CLS_[i]_MEI**.
Actually each cluster collects one EIRQ of one TIM, MCS, SPE and FIFO.
These register are typically not read out by the CPU, but they are readable.

20.3ICM Interrupt Signals

Signal	Description
GTM_AEI_IRQ	AEI Shared interrupt
GTM_ARU_IRQ[2:0]	[0]: ARU_NEW_DATA0 Interrupt [1]: ARU_NEW_DATA1 Interrupt [2]: ARU_ACC_ACK Interrupt

GTM_BRC_IRQ	BRC Shared interrupt
GTM_CMP_IRQ	CMP Shared interrupt
GTM_SPE[i]_IRQ	SPE Shared interrupt (i: 0..number of SPE's-1)
GTM_PSM[i]_IRQ[x]	PSM Shared interrupts (x: 0…7) (i: 0..number of PSM's-1)
GTM_DPLL_IRQ[0]	*DPLL_DCGI:* DPLL direction change interrupt
GTM_DPLL_IRQ[1]	*DPLL_EDI;* DPLL enable or disable interrupt
GTM_DPLL_IRQ[2]	*DPLL_TINI:* DPLL *TRIG*. min. hold time (THMI) viol. detected
GTM_DPLL_IRQ[3]	*DPLL_TAXI:* DPLL *TRIG*. max. hold time (THMA) viol. detected
GTM_DPLL_IRQ[4]	*DPLL_SISI:* DPLL *STATE* inactive slope detected
GTM_DPLL_IRQ[5]	*DPLL_TISI:* DPLL *TRIGGER* inactive slope detected
GTM_DPLL_IRQ[6]	*DPLL_MSI:* DPLL Missing *STATE* interrupt
GTM_DPLL_IRQ[7]	*DPLL_MTI:* DPLL Missing *TRIGGER* interrupt
GTM_DPLL_IRQ[8]	*DPLL_SASI:* DPLL *STATE* active slope detected
GTM_DPLL_IRQ[9]	*DPLL_TASI:* DPLL *TRIG*. active slope det. while NTI_CNT is 0
GTM_DPLL_IRQ[10]	*DPLL_PWI:* DPLL Plausibility window (PVT) viol. int. of *TRIG*.
GTM_DPLL_IRQ[11]	*DPLL_W2I:* DPLL Write access to RAM region 2 interrupt
GTM_DPLL_IRQ[12]	*DPLL_W1I:* DPLL Write access to RAM region 1b or 1c int.
GTM_DPLL_IRQ[13]	*DPLL_GL1I:* DPLL Get of lock interrupt for *SUB_INC1*
GTM_DPLL_IRQ[14]	*DPLL_LL1I:* DPLL Lost of lock interrupt for *SUB_INC1*
GTM_DPLL_IRQ[15]	*DPLL_EI:* DPLL Error interrupt
GTM_DPLL_IRQ[16]	*DPLL_GL2I:* DPLL Get of lock interrupt for *SUB_INC2*
GTM_DPLL_IRQ[17]	*DPLL_LL2I:* DPLL Lost of lock interrupt for *SUB_INC2*
GTM_DPLL_IRQ[18]	*DPLL_TE0I:* DPLL *TRIGGER* event interrupt 0
GTM_DPLL_IRQ[19]	*DPLL_TE1I:* DPLL *TRIGGER* event interrupt 1
GTM_DPLL_IRQ[20]	*DPLL_TE2I:* DPLL *TRIGGER* event interrupt 2
GTM_DPLL_IRQ[21]	*DPLL_TE3I:* DPLL *TRIGGER* event interrupt 3
GTM_DPLL_IRQ[22]	*DPLL_TE4I;* DPLL *TRIGGER* event interrupt 4
GTM_DPLL_IRQ[23]	*DPLL_CDTI;* DPLL calculated duration interrupt for *TRIGGER*
GTM_DPLL_IRQ[24]	*DPLL_CDSI;* DPLL calculated duration interrupt for *STATE*
GTM_DPLL_IRQ[25]	*DPLL_TORI; TRIGGER* out of range interrupt
GTM_DPLL_IRQ[26]	*DPLL_SORI; STATE* out of range interrupt
GTM_TIM[i]_IRQ[x]	TIM Shared interrupts (i: 0..number of TIM's-1) (x: 0..7)
GTM_MCS[i]_IRQ[x]	MCS Interrupt for channel x (x: 0…8) (i: 0..number of MCS's-1)
GTM_TOM[i]_IRQ[x]	TOM Shared interrupts for x:0..7 = {ch0||ch1,…,ch14||ch15}

	(i: 0..number of TOM's-1)
GTM_ATOM[i]_IRQ[x]	ATOM Shared interrupts for x:0..3 = {ch0||ch1,...,ch6||ch7} (i: 0..number of ATOM's-1)
GTM_ERR_IRQ	GTM Error Interrupt

20.4ICM Configuration Register Overview

20.4.1 ICM Configuration Register Overview Table

Register Name	**Description**	**Details in Section**
ICM_IRQG_0	ICM Interrupt group register covering infrastructural and safety components (ARU, BRC, AEI, PSM0, PSM1, MAP, CMP,SPE)	20.5.1
ICM_IRQG_1	ICM Interrupt group register covering DPLL	20.5.2
ICM_IRQG_2	ICM Interrupt group register covering TIM0, TIM1, TIM2, TIM3	20.5.3
ICM_IRQG_3	ICM Interrupt group register covering TIM4, TIM5, TIM6, TIM7	20.5.4
ICM_IRQG_4	ICM Interrupt group register covering MCS0 to MCS3 sub-modules	20.5.5
ICM_IRQG_5	ICM Interrupt group register covering MCS4 to MCS6 sub-modules	20.5.6
ICM_IRQG_6	ICM Interrupt group register covering GTM-IP output sub-modules TOM0 to TOM1	20.5.7
ICM_IRQG_7	ICM Interrupt group register covering GTM-IP output sub-modules TOM2 to TOM3	20.5.8
ICM_IRQG_8	ICM Interrupt group register covering GTM-IP output sub-modules TOM4 to TOM5	20.5.9
ICM_IRQG_9	ICM Interrupt group register covering GTM-IP output sub-modules ATOM0, ATOM1, ATOM2 and ATOM3	20.5.10
ICM_IRQG_10	ICM Interrupt group register covering GTM-IP output sub-modules ATOM4 to ATOM7	20.5.11

ICM_IRQG_11	ICM Interrupt group register covering GTM-IP output sub-modules ATOM8 to ATOM11	20.5.12
ICM_IRQG_MEI	ICM Interrupt group register for module error interrupt information	20.5.13
ICM_IRQG_CEI0	ICM Interrupt group register 0 for channel error interrupt information	20.5.14
ICM_IRQG_CEI1	ICM Interrupt group register 1 for channel error interrupt information	20.5.15
ICM_IRQG_CEI2	ICM Interrupt group register 2 for channel error interrupt information	20.5.16
ICM_IRQG_CEI3	ICM Interrupt group register 3 for channel error interrupt information	20.5.17
ICM_IRQG_CEI4	ICM Interrupt group register 4 for channel error interrupt information	20.5.18
ICM_IRQG_MCS[i]_CI	ICM Interrupt group MCS i for Channel Interrupt information	20.5.19
ICM_IRQG_MCS[i]_CEI	ICM Interrupt group MCS i for Channel Error Interrupt information	20.5.20
ICM_IRQG_SPE_CI	ICM Interrupt group SPE for module Interrupt information	20.5.21
ICM_IRQG_SPE_CEI	ICM Interrupt group SPE for module Error Interrupt information	20.5.22
ICM_IRQG_PSM_0_CI	ICM Interrupt group PSM 0 for Channel Interrupt information of FIFO0, FIFO1, FIFO2	20.5.23
ICM_IRQG_PSM_0_CEI	ICM Interrupt group PSM 0 for Channel Error Interrupt information of FIFO0, FIFO1, FIFO2	20.5.24
ICM_IRQG_TOM_[k]_CI (k:0…2)	ICM Interrupt group TOM k for Channel Interrupt information of TOMm (m=2*k+(0..1))	20.5.25
ICM_IRQG_ATOM_[k]_CI (k:0…2)	ICM Interrupt group ATOM k for Channel Interrupt information of ATOMm (m=4*k+(0..3))	20.5.26
ICM_IRQG_CLS_[k]_MEI (k:0…2)	ICM Interrupt group for module Error Interrupt information for each TIMm, MCSm, SPEm, FIFOm (m=4*k+(0..3))	20.5.27

20.5ICM Configuration Register Description

20.5.1 Register ICM_IRQG_0

Address Offset:	see Appendix B																Initial Value: 0x0000_0000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	PSM1_CH7_IRQ	PSM1_CH6_IRQ	PSM1_CH5_IRQ	PSM1_CH4_IRQ	PSM1_CH3_IRQ	PSM1_CH2_IRQ	PSM1_CH1_IRQ	PSM1_CH0_IRQ	PSM0_CH7_IRQ	PSM0_CH6_IRQ	PSM0_CH5_IRQ	PSM0_CH4_IRQ	PSM0_CH3_IRQ	PSM0_CH2_IRQ	PSM0_CH1_IRQ	PSM0_CH0_IRQ	Reserved				SPE5_IRQ	SPE4_IRQ	SPE3_IRQ	SPE2_IRQ	SPE1_IRQ	SPE0_IRQ	CMP_IRQ	AEI_IRQ	BRC_IRQ	ARU_ACC_ACK_I	ARU_NEW_DATA	ARU_NEW_DATA
Mode	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R				R	R	R	R	R	R	R	R	R	R	R	R
Initial Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0x00				0	0	0	0	0	0	0	0	0	0	0	0

Bit 0 **ARU_NEW_DATA0_IRQ:** *ARU_NEW_DATA0* interrupt

0 = no interrupt occurred

1 = interrupt was raised by the corresponding sub-module

Note: This bit is only set, when the interrupt is enabled in the interrupt enable register of the corresponding sub-module.

Bit 1 **ARU_NEW_DATA1_IRQ:** *ARU_NEW_DATA1* interrupt. See bit 0.

Bit 2 **ARU_ACC_ACK_IRQ:** *ARU_ACC_ACK* interrupt. See bit 0.

Bit 3 **BRC_DID_IRQ:** BRC shared sub-module interrupt. See bit 0.

Bit 4 **AEI_IRQ:** *AEI_IRQ* interrupt. See bit 0.

Note: Set this bit represents an OR function of the interrupt sources *AEI_TO_XPT*, *AEI_USP_ADDR*, *AEI_IM_ADDR*, *AEI_USP_BE*, *AEIM_USP_ADDR*, *AEIM_IM_ADDR,* *AEIM_USP_BE*, *CLK_EN_ERR* or *CLK_PER_ERR*.

Bit 5 **CMP_IRQ:** CMP shared sub-module interrupt. See bit 0.

Bit 6 **SPE0_IRQ:** SPE0 shared sub-module interrupt. See bit 0.

Note: Set this bit represents an OR function of the five interrupt sources *SPE_NIPD*, *SPE_DCHG*, *SPE_PERR*, *SPE_BIS* or *SPE_RCMP* of SPE instance 0.

Bit 7 **SPE1_IRQ:** SPE1 shared sub-module interrupt. See bit 0 and bit 6.

Bit 8 **SPE2_IRQ:** SPE2 shared sub-module interrupt. See bit 0 and bit 6.

Bit 9 **SPE3_IRQ:** SPE3 shared sub-module interrupt. See bit 0 and bit 6.

Bit 10 **SPE4_IRQ:** SPE4 shared sub-module interrupt. See bit 0 and bit 6.

Bit 11 **SPE5_IRQ:** SPE5 shared sub-module interrupt. See bit 0 and bit 6.

Bit 15:12 **Reserved**

Note: Read as zero, should be written as zero

Bit 16 **PSM0_CH0_IRQ:** PSM0 shared sub-module channel 0 interrupt. See bit 0

Note: Set this bit represents an OR function of the four interrupt sources *FIFO_EMPTY*, *FIFO_FULL*, *FIFO_LOWER_WM* or *FIFO_UPPER_WM* of FIFO instance 0 channel 0.

Bit 17 **PSM0_CH1_IRQ:** PSM0 shared sub-module channel 1 interrupt. See bit 0 and 16.

Bit 18 **PSM0_CH2_IRQ:** PSM0 shared sub-module channel 2 interrupt. See bit 0 and 16.

Bit 19 **PSM0_CH3_IRQ:** PSM0 shared sub-module channel 3 interrupt. See bit 0 and 16.

Bit 20 **PSM0_CH4_IRQ:** PSM0 shared sub-module channel 4 interrupt. See bit 0 and 16.

Bit 21 **PSM0_CH5_IRQ:** PSM0 shared sub-module channel 5 interrupt. See bit 0 and 16.

Bit 22 **PSM0_CH6_IRQ:** PSM0 shared sub-module channel 6 interrupt. See bit 0 and 16.

Bit 23 **PSM0_CH7_IRQ:** PSM0 shared sub-module channel 7 interrupt. See bit 0 and 16.

Bit 24 **PSM1_CH0_IRQ:** PSM1 shared sub-module channel 0 interrupt. See bit 0 and 16.

Bit 25 **PSM1_CH1_IRQ:** PSM1 shared sub-module channel 1 interrupt. See bit 0 and 16.

Bit 26 **PSM1_CH2_IRQ:** PSM1 shared sub-module channel 2 interrupt. See bit 0 and 16.

Bit 27 **PSM1_CH3_IRQ:** PSM1 shared sub-module channel 3 interrupt. See bit 0 and 16.

Bit 28 **PSM1_CH4_IRQ:** PSM1 shared sub-module channel 4 interrupt. See bit 0 and 16.

Bit 29 **PSM1_CH5_IRQ:** PSM1 shared sub-module channel 5 interrupt. See bit 0 and 16.

Bit 30 **PSM1_CH6_IRQ:** PSM1 shared sub-module channel 6 interrupt. See bit 0 and 16.

Bit 31 **PSM1_CH7_IRQ:** PSM1 shared sub-module channel 7 interrupt. See bit 0 and 16.

20.5.2 Register ICM_IRQG_1

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_0000**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved					DPLL_SORI_IRQ	DPLL_TORI_IRQ	DPLL_CDSI_IRQ	DPLL_CDTI_IRQ	DPLL_TE4I_IRQ	DPLL_TE3I_IRQ	DPLL_TE2I_IRQ	DPLL_TE1I_IRQ	DPLL_TE0I_IRQ	DPLL_LL2I_IRQ	DPLL_GL2I_IRQ	DPLL_EI_IRQ	DPLL_LL1I_IRQ	DPLL_GL1I_IRQ	DPLL_W1I_IRQ	DPLL_W2I_IRQ	DPLL_PWI_IRQ	DPLL_TASI_IRQ	DPLL_SASI_IRQ	DPLL_MTI_IRQ	DPLL_MSI_IRQ	DPLL_TISI_IRQ	DPLL_SISI_IRQ	DPLL_TAXI_IRQ	DPLL_TINI_IRQ	DPLL_EDI_IRQ	DPLL_DCGI_IRQ
Mode	R					R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Initial Value	0x00					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit 0 **DPLL_DCGI_IRQ:** *TRIGGER* direction change detected.
0 = no interrupt occurred
1 = interrupt was raised by the corresponding sub-module
Note: This bit is only set, when the interrupt is enabled in the interrupt enable register of the corresponding sub-module.

Bit 1 **DPLL_EDI_IRQ:** DPLL enable/disable interrupt. See bit 0.
Note: Set this bit represents an OR function of the two interrupt sources *DPLL_PDI* or *DPLL_PEI* .

Bit 2 **DPLL_TINI_IRQ:** *TRIGGER* minimum hold time (THMI) violation detected interrupt. See bit 0.

Bit 3 **DPLL_TAXI_IRQ:** *TRIGGER* maximum hold time (THMA) violation detected interrupt. See bit 0.

Bit 4 **DPLL_SISI_IRQ:** *STATE* inactive slope detected interrupt. See bit 0.

Bit 5 **DPLL_TISI_IRQ:** *TRIGGER* inactive slope detected interrupt. See bit 0.

Bit 6 **DPLL_MSI_IRQ:** Missing *STATE* interrupt. See bit 0.

Bit 7 **DPLL_MTI_IRQ:** Missing *TRIGGER* interrupt. See bit 0.

Bit 8 **DPLL_SASI_IRQ:** *STATE* active slope detected. See bit 0.

Bit 9 **DPLL_TASI_IRQ:** *TRIGGER* active slope detected while NTI_CNT is zero. See bit 0.

Bit 10 **DPLL_PWI_IRQ:** Plausibility window (PVT) violation interrupt of *TRIGGER*. See bit 0.

Bit 11 **DPLL_W2I_IRQ:** Write access to RAM region 2 interrupt. See bit 0.

Bit 12 **DPLL_W1I_IRQ:** Write access to RAM region 1b or 1c interrupt. See bit 0.

Bit 13 **DPLL_GL1I_IRQ:** Get of lock interrupt for *SUB_INC1*. See bit 0.

Bit 14 **DPLL_LL1I_IRQ:** Lost of lock interrupt for *SUB_INC1*. See bit 0.

Bit 15 **DPLL_EI_IRQ:** Error interrupt. See bit 0.

Bit 16 **DPLL_GL2I_IRQ:** Get of lock interrupt for *SUB_INC2*. See bit 0.

Bit 17 **DPLL_LL2I_IRQ:** Lost of lock interrupt for *SUB_INC2*. See bit 0.

Bit 18 **DPLL_TE0I_IRQ:** *TRIGGER* event interrupt 0. See bit 0.

Bit 19 **DPLL_TE1I_IRQ:** *TRIGGER* event interrupt 1. See bit 0.

Bit 20 **DPLL_TE2I_IRQ:** *TRIGGER* event interrupt 2. See bit 0.

Bit 21 **DPLL_TE3I_IRQ:** *TRIGGER* event interrupt 3. See bit 0.

Bit 22 **DPLL_TE4I_IRQ:** *TRIGGER* event interrupt 4. See bit 0.

Bit 23 **DPLL_CDTI_IRQ:** DPLL calculated duration interrupt for trigger. See bit 0.

Bit 24 **DPLL_CDSI_IRQ:** DPLL calculated duration interrupt for state. See bit 0.

Bit 25 **DPLL_TORI_IRQ:** DPLL calculated duration interrupt for state. See bit 0.

Bit 26 **DPLL_SORI_IRQ:** DPLL calculated duration interrupt for state. See bit 0.

Bit 31:27 **Reserved:** Reserved
Note: Read as zero, should be written as zero

20.5.3 Register ICM_IRQG_2

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_0000**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	TIM3_CH7_IRQ	TIM3_CH6_IRQ	TIM3_CH5_IRQ	TIM3_CH4_IRQ	TIM3_CH3_IRQ	TIM3_CH2_IRQ	TIM3_CH1_IRQ	TIM3_CH0_IRQ	TIM2_CH7_IRQ	TIM2_CH6_IRQ	TIM2_CH5_IRQ	TIM2_CH4_IRQ	TIM2_CH3_IRQ	TIM2_CH2_IRQ	TIM2_CH1_IRQ	TIM2_CH0_IRQ	TIM1_CH7_IRQ	TIM1_CH6_IRQ	TIM1_CH5_IRQ	TIM1_CH4_IRQ	TIM1_CH3_IRQ	TIM1_CH2_IRQ	TIM1_CH1_IRQ	TIM1_CH0_IRQ	TIM0_CH7_IRQ	TIM0_CH6_IRQ	TIM0_CH5_IRQ	TIM0_CH4_IRQ	TIM0_CH3_IRQ	TIM0_CH2_IRQ	TIM0_CH1_IRQ	TIM0_CH0_IRQ
Mode	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Initial Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit 0 **TIM0_CH0_IRQ:** TIM0 shared interrupt channel 0.
0 = no interrupt occurred
1 = interrupt was raised by the corresponding sub-module
Note: This bit is only set, when the interrupt is enabled in the interrupt enable register of the corresponding sub-module.

Note: Set this bit represents an OR function of the six interrupt sources *NEWVALx_IRQ*, *ECNTOFLx_IRQ*, *CNTOFLx_IRQ*, *GPRXOFLx_IRQ*, GLITCHDETx_IRQ or *TODETx_IRQ* of TIM instance 0 channel x.

Bit 1 **TIM0_CH1_IRQ:** TIM0 shared interrupt channel 1. See bit 0.

Bit 2 **TIM0_CH2_IRQ:** TIM0 shared interrupt channel 2 . See bit 0.

Bit 3 **TIM0_CH3_IRQ:** TIM0 shared interrupt channel 3. See bit 0.

Bit 4 **TIM0_CH4_IRQ:** TIM0 shared interrupt channel 4. See bit 0.

Bit 5 **TIM0_CH5_IRQ:** TIM0 shared interrupt channel 5. See bit 0.

Bit 6 **TIM0_CH6_IRQ:** TIM0 shared interrupt channel 6. See bit 0.

Bit 7 **TIM0_CH7_IRQ:** TIM0 shared interrupt channel 7. See bit 0.

Bit 8 **TIM1_CH0_IRQ:** TIM1 shared interrupt channel 0. See bit 0.

Bit 9 **TIM1_CH1_IRQ:** TIM1 shared interrupt channel 1. See bit 0.
Bit 10 **TIM1_CH2_IRQ:** TIM1 shared interrupt channel 2. See bit 0.
Bit 11 **TIM1_CH3_IRQ:** TIM1 shared interrupt channel 3. See bit 0.
Bit 12 **TIM1_CH4_IRQ:** TIM1 shared interrupt channel 4. See bit 0.
Bit 13 **TIM1_CH5_IRQ:** TIM1 shared interrupt channel 5. See bit 0.
Bit 14 **TIM1_CH6_IRQ:** TIM1 shared interrupt channel 6. See bit 0.
Bit 15 **TIM1_CH7_IRQ:** TIM1 shared interrupt channel 7. See bit 0.
Bit 16 **TIM2_CH0_IRQ:** TIM2 shared interrupt channel 0. See bit 0.
Bit 17 **TIM2_CH1_IRQ:** TIM2 shared interrupt channel 1. See bit 0.
Bit 18 **TIM2_CH2_IRQ:** TIM2 shared interrupt channel 2. See bit 0.
Bit 19 **TIM2_CH3_IRQ:** TIM2 shared interrupt channel 3. See bit 0.
Bit 20 **TIM2_CH4_IRQ:** TIM2 shared interrupt channel 4. See bit 0.
Bit 21 **TIM2_CH5_IRQ:** TIM2 shared interrupt channel 5. See bit 0.
Bit 22 **TIM2_CH6_IRQ:** TIM2 shared interrupt channel 6. See bit 0.
Bit 23 **TIM2_CH7_IRQ:** TIM2 shared interrupt channel 7. See bit 0.
Bit 24 **TIM3_CH0_IRQ:** TIM3 shared interrupt channel 0. See bit 0.
Bit 25 **TIM3_CH1_IRQ:** TIM3 shared interrupt channel 1. See bit 0.
Bit 26 **TIM3_CH2_IRQ:** TIM3 shared interrupt channel 2. See bit 0.
Bit 27 **TIM3_CH3_IRQ:** TIM3 shared interrupt channel 3. See bit 0.
Bit 28 **TIM3_CH4_IRQ:** TIM3 shared interrupt channel 4. See bit 0.
Bit 29 **TIM3_CH5_IRQ:** TIM3 shared interrupt channel 5. See bit 0.
Bit 30 **TIM3_CH6_IRQ:** TIM3 shared interrupt channel 6. See bit 0.
Bit 31 **TIM3_CH7_IRQ:** TIM3 shared interrupt channel 7. See bit 0.

20.5.4 Register ICM_IRQG_3

Address Offset:	**see Appendix B**																**Initial Value:**						**0x0000_0000**									
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	TIM7_CH7_IRQ	TIM7_CH6_IRQ	TIM7_CH5_IRQ	TIM7_CH4_IRQ	TIM7_CH3_IRQ	TIM7_CH2_IRQ	TIM7_CH1_IRQ	TIM7_CH0_IRQ	TIM6_CH7_IRQ	TIM6_CH6_IRQ	TIM6_CH5_IRQ	TIM6_CH4_IRQ	TIM6_CH3_IRQ	TIM6_CH2_IRQ	TIM6_CH1_IRQ	TIM6_CH0_IRQ	TIM5_CH7_IRQ	TIM5_CH6_IRQ	TIM5_CH5_IRQ	TIM5_CH4_IRQ	TIM5_CH3_IRQ	TIM5_CH2_IRQ	TIM5_CH1_IRQ	TIM5_CH0_IRQ	TIM4_CH7_IRQ	TIM4_CH6_IRQ	TIM4_CH5_IRQ	TIM4_CH4_IRQ	TIM4_CH3_IRQ	TIM4_CH2_IRQ	TIM4_CH1_IRQ	TIM4_CH0_IRQ
Mode	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Initial Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit 0 **TIM4_CH0_IRQ:** TIM4 shared interrupt channel 0.
0 = no interrupt occurred
1 = interrupt was raised by the corresponding sub-module
Note: This bit is only set, when the interrupt is enabled in the interrupt enable register of the corresponding sub-module.

Note: Set this bit represents an OR function of the six interrupt sources *NEWVALx_IRQ*, *ECNTOFLx_IRQ*, *CNTOFLx_IRQ*, *GPRXOFLx_IRQ*, GLITCHDETx_IRQ or *TODETx_IRQ* of TIM instance 4 channel x.

Bit 1 **TIM4_CH1_IRQ:** TIM4 shared interrupt channel 1. See bit 0.
Bit 2 **TIM4_CH2_IRQ:** TIM4 shared interrupt channel 2 . See bit 0.
Bit 3 **TIM4_CH3_IRQ:** TIM4 shared interrupt channel 3. See bit 0.
Bit 4 **TIM4_CH4_IRQ:** TIM4 shared interrupt channel 4. See bit 0.
Bit 5 **TIM4_CH5_IRQ:** TIM4 shared interrupt channel 5. See bit 0.
Bit 6 **TIM4_CH6_IRQ:** TIM4 shared interrupt channel 6. See bit 0.
Bit 7 **TIM4_CH7_IRQ:** TIM4 shared interrupt channel 7. See bit 0.
Bit 8 **TIM5_CH0_IRQ:** TIM5 shared interrupt channel 0. See bit 0.
Bit 9 **TIM5_CH1_IRQ:** TIM5 shared interrupt channel 1. See bit 0.
Bit 10 **TIM5_CH2_IRQ:** TIM5 shared interrupt channel 2. See bit 0.
Bit 11 **TIM5_CH3_IRQ:** TIM5 shared interrupt channel 3. See bit 0.
Bit 12 **TIM5_CH4_IRQ:** TIM5 shared interrupt channel 4. See bit 0.
Bit 13 **TIM5_CH5_IRQ:** TIM5 shared interrupt channel 5. See bit 0.
Bit 14 **TIM5_CH6_IRQ:** TIM5 shared interrupt channel 6. See bit 0.
Bit 15 **TIM5_CH7_IRQ:** TIM5 shared interrupt channel 7. See bit 0.
Bit 16 **TIM6_CH0_IRQ:** TIM6 shared interrupt channel 0. See bit 0.
Bit 17 **TIM6_CH1_IRQ:** TIM6 shared interrupt channel 1. See bit 0.
Bit 18 **TIM6_CH2_IRQ:** TIM6 shared interrupt channel 2. See bit 0.
Bit 19 **TIM6_CH3_IRQ:** TIM6 shared interrupt channel 3. See bit 0.
Bit 20 **TIM6_CH4_IRQ:** TIM6 shared interrupt channel 4. See bit 0.
Bit 21 **TIM6_CH5_IRQ:** TIM6 shared interrupt channel 5. See bit 0.
Bit 22 **TIM6_CH6_IRQ:** TIM6 shared interrupt channel 6. See bit 0.
Bit 23 **TIM6_CH7_IRQ:** TIM6 shared interrupt channel 7. See bit 0.
Bit 24 **TIM7_CH0_IRQ:** TIM7 shared interrupt channel 0. See bit 0.
Bit 25 **TIM7_CH1_IRQ:** TIM7 shared interrupt channel 1. See bit 0.
Bit 26 **TIM7_CH2_IRQ:** TIM7 shared interrupt channel 2. See bit 0.
Bit 27 **TIM7_CH3_IRQ:** TIM7 shared interrupt channel 3. See bit 0.
Bit 28 **TIM7_CH4_IRQ:** TIM7 shared interrupt channel 4. See bit 0.
Bit 29 **TIM7_CH5_IRQ:** TIM7 shared interrupt channel 5. See bit 0.
Bit 30 **TIM7_CH6_IRQ:** TIM7 shared interrupt channel 6. See bit 0.
Bit 31 **TIM7_CH7_IRQ:** TIM7 shared interrupt channel 7. See bit 0.

20.5.5 Register ICM_IRQG_4

Address Offset:	**see Appendix B**																**Initial Value: 0x0000_0000**															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	MCS3_CH7_IRQ	MCS3_CH6_IRQ	MCS3_CH5_IRQ	MCS3_CH4_IRQ	MCS3_CH3_IRQ	MCS3_CH2_IRQ	MCS3_CH1_IRQ	MCS3_CH0_IRQ	MCS2_CH7_IRQ	MCS2_CH6_IRQ	MCS2_CH5_IRQ	MCS2_CH4_IRQ	MCS2_CH3_IRQ	MCS2_CH2_IRQ	MCS2_CH1_IRQ	MCS2_CH0_IRQ	MCS1_CH7_IRQ	MCS1_CH6_IRQ	MCS1_CH5_IRQ	MCS1_CH4_IRQ	MCS1_CH3_IRQ	MCS1_CH2_IRQ	MCS1_CH1_IRQ	MCS1_CH0_IRQ	MCS0_CH7_IRQ	MCS0_CH6_IRQ	MCS0_CH5_IRQ	MCS0_CH4_IRQ	MCS0_CH3_IRQ	MCS0_CH2_IRQ	MCS0_CH1_IRQ	MCS0_CH0_IRQ
Mode	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Initial Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit 0 **MCS0_CH0_IRQ:** MCS0 channel 0 interrupt
0 = no interrupt occurred
1 = interrupt was raised by the corresponding sub-module
Note: This bit is only set, when the interrupt is enabled in the interrupt enable register of the corresponding sub-module.

Note: Set this bit represents an OR function of the three interrupt sources *MCS_IRQ*, *STK_ERR_IRQ* or *ERR_IRQ* of MCS instance 0 channel 0.

Bit 1 **MCS0_CH1_IRQ:** MCS0 channel 1 interrupt. See bit 0.
Bit 2 **MCS0_CH2_IRQ:** MCS0 channel 2 interrupt. See bit 0.
Bit 3 **MCS0_CH3_IRQ:** MCS0 channel 3 interrupt. See bit 0.
Bit 4 **MCS0_CH4_IRQ:** MCS0 channel 4 interrupt. See bit 0.
Bit 5 **MCS0_CH5_IRQ:** MCS0 channel 5 interrupt. See bit 0.
Bit 6 **MCS0_CH6_IRQ:** MCS0 channel 6 interrupt. See bit 0.
Bit 7 **MCS0_CH7_IRQ:** MCS0 channel 7 interrupt. See bit 0.
Bit 8 **MCS1_CH0_IRQ:** MCS1 channel 0 interrupt. See bit 0.
Bit 9 **MCS1_CH1_IRQ:** MCS1 channel 1 interrupt. See bit 0.
Bit 10 **MCS1_CH2_IRQ:** MCS1 channel 2 interrupt. See bit 0.
Bit 11 **MCS1_CH3_IRQ:** MCS1 channel 3 interrupt. See bit 0.
Bit 12 **MCS1_CH4_IRQ:** MCS1 channel 4 interrupt. See bit 0.
Bit 13 **MCS1_CH5_IRQ:** MCS1 channel 5 interrupt. See bit 0.
Bit 14 **MCS1_CH6_IRQ:** MCS1 channel 6 interrupt. See bit 0.
Bit 15 **MCS1_CH7_IRQ:** MCS1 channel 7 interrupt. See bit 0.
Bit 16 **MCS2_CH0_IRQ:** MCS2 channel 0 interrupt. See bit 0.
Bit 17 **MCS2_CH1_IRQ:** MCS2 channel 1 interrupt. See bit 0.
Bit 18 **MCS2_CH2_IRQ:** MCS2 channel 2 interrupt. See bit 0.
Bit 19 **MCS2_CH3_IRQ:** MCS2 channel 3 interrupt. See bit 0.
Bit 20 **MCS2_CH4_IRQ:** MCS2 channel 4 interrupt. See bit 0.
Bit 21 **MCS2_CH5_IRQ:** MCS2 channel 5 interrupt. See bit 0.
Bit 22 **MCS2_CH6_IRQ:** MCS2 channel 6 interrupt. See bit 0.
Bit 23 **MCS2_CH7_IRQ:** MCS2 channel 7 interrupt. See bit 0.
Bit 24 **MCS3_CH0_IRQ:** MCS3 channel 0 interrupt. See bit 0.

Bit 25 **MCS3_CH1_IRQ:** MCS3 channel 1 interrupt. See bit 0.
Bit 26 **MCS3_CH2_IRQ:** MCS3 channel 2 interrupt. See bit 0.
Bit 27 **MCS3_CH3_IRQ:** MCS3 channel 3 interrupt. See bit 0.
Bit 28 **MCS3_CH4_IRQ:** MCS3 channel 4 interrupt. See bit 0.
Bit 29 **MCS3_CH5_IRQ:** MCS3 channel 5 interrupt. See bit 0.
Bit 30 **MCS3_CH6_IRQ:** MCS3 channel 6 interrupt. See bit 0.
Bit 31 **MCS3_CH7_IRQ:** MCS3 channel 7 interrupt. See bit 0.

20.5.6 Register ICM_IRQG_5

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_0000**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	MCS7_CH7_IRQ	MCS7_CH6_IRQ	MCS7_CH5_IRQ	MCS7_CH4_IRQ	MCS7_CH3_IRQ	MCS7_CH2_IRQ	MCS7_CH1_IRQ	MCS7_CH0_IRQ	MCS6_CH7_IRQ	MCS6_CH6_IRQ	MCS6_CH5_IRQ	MCS6_CH4_IRQ	MCS6_CH3_IRQ	MCS6_CH2_IRQ	MCS6_CH1_IRQ	MCS6_CH0_IRQ	MCS5_CH7_IRQ	MCS5_CH6_IRQ	MCS5_CH5_IRQ	MCS5_CH4_IRQ	MCS5_CH3_IRQ	MCS5_CH2_IRQ	MCS5_CH1_IRQ	MCS5_CH0_IRQ	MCS4_CH7_IRQ	MCS4_CH6_IRQ	MCS4_CH5_IRQ	MCS4_CH4_IRQ	MCS4_CH3_IRQ	MCS4_CH2_IRQ	MCS4_CH1_IRQ	MCS4_CH0_IRQ
Mode	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Initial Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit 0 **MCS4_CH0_IRQ:** MCS4 channel 0 interrupt
0 = no interrupt occurred
1 = interrupt was raised by the corresponding sub-module
Note: This bit is only set, when the interrupt is enabled in the interrupt enable register of the corresponding sub-module.

Note: Set this bit represents an OR function of the three interrupt sources *MCS_IRQ*, *STK_ERR_IRQ* or *ERR_IRQ* of MCS instance 4 channel 0.

Bit 1 **MCS4_CH1_IRQ:** MCS4 channel 1 interrupt. See bit 0.
Bit 2 **MCS4_CH2_IRQ:** MCS4 channel 2 interrupt. See bit 0.
Bit 3 **MCS4_CH3_IRQ:** MCS4 channel 3 interrupt. See bit 0.
Bit 4 **MCS4_CH4_IRQ:** MCS4 channel 4 interrupt. See bit 0.
Bit 5 **MCS4_CH5_IRQ:** MCS4 channel 5 interrupt. See bit 0.
Bit 6 **MCS4_CH6_IRQ:** MCS4 channel 6 interrupt. See bit 0.
Bit 7 **MCS4_CH7_IRQ:** MCS4 channel 7 interrupt. See bit 0.
Bit 8 **MCS5_CH0_IRQ:** MCS5 channel 0 interrupt. See bit 0.
Bit 9 **MCS5_CH1_IRQ:** MCS5 channel 1 interrupt. See bit 0.
Bit 10 **MCS5_CH2_IRQ:** MCS5 channel 2 interrupt. See bit 0.
Bit 11 **MCS5_CH3_IRQ:** MCS5 channel 3 interrupt. See bit 0.
Bit 12 **MCS5_CH4_IRQ:** MCS5 channel 4 interrupt. See bit 0.
Bit 13 **MCS5_CH5_IRQ:** MCS5 channel 5 interrupt. See bit 0.

Bit 14 **MCS5_CH6_IRQ:** MCS5 channel 6 interrupt. See bit 0.
Bit 15 **MCS5_CH7_IRQ:** MCS5 channel 7 interrupt. See bit 0.
Bit 16 **MCS6_CH0_IRQ:** MCS1 channel 0 interrupt. See bit 0.
Bit 17 **MCS6_CH1_IRQ:** MCS6 channel 1 interrupt. See bit 0.
Bit 18 **MCS6_CH2_IRQ:** MCS6 channel 2 interrupt. See bit 0.
Bit 19 **MCS6_CH3_IRQ:** MCS6 channel 3 interrupt. See bit 0.
Bit 20 **MCS6_CH4_IRQ:** MCS6 channel 4 interrupt. See bit 0.
Bit 21 **MCS6_CH5_IRQ:** MCS6 channel 5 interrupt. See bit 0.
Bit 22 **MCS6_CH6_IRQ:** MCS6 channel 6 interrupt. See bit 0.
Bit 23 **MCS6_CH7_IRQ:** MCS6 channel 7 interrupt. See bit 0.
Bit 24 **MCS7_CH0_IRQ:** MCS7 channel 0 interrupt. See bit 0.
Bit 25 **MCS7_CH1_IRQ:** MCS7 channel 1 interrupt. See bit 0.
Bit 26 **MCS7_CH2_IRQ:** MCS7 channel 2 interrupt. See bit 0.
Bit 27 **MCS7_CH3_IRQ:** MCS7 channel 3 interrupt. See bit 0.
Bit 28 **MCS7_CH4_IRQ:** MCS7 channel 4 interrupt. See bit 0.
Bit 29 **MCS7_CH5_IRQ:** MCS7 channel 5 interrupt. See bit 0.
Bit 30 **MCS7_CH6_IRQ:** MCS7 channel 6 interrupt. See bit 0.
Bit 31 **MCS7_CH7_IRQ:** MCS7 channel 7 interrupt. See bit 0.

20.5.7 Register ICM_IRQG_6

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_0000**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	TOM1_CH15_IRQ	TOM1_CH14_IRQ	TOM1_CH13_IRQ	TOM1_CH12_IRQ	TOM1_CH11_IRQ	TOM1_CH10_IRQ	TOM1_CH9_IRQ	TOM1_CH8_IRQ	TOM1_CH7_IRQ	TOM1_CH6_IRQ	TOM1_CH5_IRQ	TOM1_CH4_IRQ	TOM1_CH3_IRQ	TOM1_CH2_IRQ	TOM1_CH1_IRQ	TOM1_CH0_IRQ	TOM0_CH15_IRQ	TOM0_CH14_IRQ	TOM0_CH13_IRQ	TOM0_CH12_IRQ	TOM0_CH11_IRQ	TOM0_CH10_IRQ	TOM0_CH9_IRQ	TOM0_CH8_IRQ	TOM0_CH7_IRQ	TOM0_CH6_IRQ	TOM0_CH5_IRQ	TOM0_CH4_IRQ	TOM0_CH3_IRQ	TOM0_CH2_IRQ	TOM0_CH1_IRQ	TOM0_CH0_IRQ
Mode	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Initial Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit 0 **TOM0_CH0_IRQ:** TOM0 channel 0 shared interrupt
0 = no interrupt occurred
1 = interrupt was raised by the corresponding sub-module
Note: This bit is only set, when the interrupt is enabled in the interrupt enable register of the corresponding sub-module.

Note: Set this bit represents an OR function of the two interrupt sources *TOM_CCU0TCx_IRQ* or *TOM_CCU1TCx_IRQ* of TOM instance 0 channel x.

Bit 1 **TOM0_CH1_IRQ:** TOM0 channel 1 shared interrupt. See bit 0.
Bit 2 **TOM0_CH2_IRQ:** TOM0 channel 2 shared interrupt. See bit 0.

Bit 3 **TOM0_CH3_IRQ:** TOM0 channel 3 shared interrupt. See bit 0.
Bit 4 **TOM0_CH4_IRQ:** TOM0 channel 4 shared interrupt. See bit 0.
Bit 5 **TOM0_CH5_IRQ:** TOM0 channel 5 shared interrupt. See bit 0.
Bit 6 **TOM0_CH6_IRQ:** TOM0 channel 6 shared interrupt. See bit 0.
Bit 7 **TOM0_CH7_IRQ:** TOM0 channel 7 shared interrupt. See bit 0.
Bit 8 **TOM0_CH8_IRQ:** TOM0 channel 8 shared interrupt. See bit 0.
Bit 9 **TOM0_CH9_IRQ:** TOM0 channel 9 shared interrupt. See bit 0.
Bit 10 **TOM0_CH10_IRQ:** TOM0 channel 10 shared interrupt. See bit 0.
Bit 11 **TOM0_CH11_IRQ:** TOM0 channel 11 shared interrupt. See bit 0.
Bit 12 **TOM0_CH12_IRQ:** TOM0 channel 12 shared interrupt. See bit 0.
Bit 13 **TOM0_CH13_IRQ:** TOM0 channel 13 shared interrupt. See bit 0.
Bit 14 **TOM0_CH14_IRQ:** TOM0 channel 14 shared interrupt. See bit 0.
Bit 15 **TOM0_CH15_IRQ:** TOM0 channel 15 shared interrupt. See bit 0.
Bit 16 **TOM1_CH0_IRQ:** TOM1 channel 0 shared interrupt. See bit 0.
Bit 17 **TOM1_CH1_IRQ:** TOM1 channel 1 shared interrupt. See bit 0.
Bit 18 **TOM1_CH2_IRQ:** TOM1 channel 2 shared interrupt. See bit 0.
Bit 19 **TOM1_CH3_IRQ:** TOM1 channel 3 shared interrupt. See bit 0.
Bit 20 **TOM1_CH4_IRQ:** TOM1 channel 4 shared interrupt. See bit 0.
Bit 21 **TOM1_CH5_IRQ:** TOM1 channel 5 shared interrupt. See bit 0.
Bit 22 **TOM1_CH6_IRQ:** TOM1 channel 6 shared interrupt. See bit 0.
Bit 23 **TOM1_CH7_IRQ:** TOM1 channel 7 shared interrupt. See bit 0.
Bit 24 **TOM1_CH8_IRQ:** TOM1 channel 8 shared interrupt. See bit 0.
Bit 25 **TOM1_CH9_IRQ:** TOM1 channel 9 shared interrupt. See bit 0.
Bit 26 **TOM1_CH10_IRQ:** TOM1 channel 10 shared interrupt. See bit 0.
Bit 27 **TOM1_CH11_IRQ:** TOM1 channel 11 shared interrupt. See bit 0.
Bit 28 **TOM1_CH12_IRQ:** TOM1 channel 12 shared interrupt. See bit 0.
Bit 29 **TOM1_CH13_IRQ:** TOM1 channel 13 shared interrupt. See bit 0.
Bit 30 **TOM1_CH14_IRQ:** TOM1 channel 14 shared interrupt. See bit 0.
Bit 31 **TOM1_CH15_IRQ:** TOM1 channel 15 shared interrupt. See bit 0.

20.5.8 Register ICM_IRQG_7

Address Offset:	**see Appendix B**																**Initial Value:**	**0x0000_0000**														
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	TOM3_CH15_IRQ	TOM3_CH14_IRQ	TOM3_CH13_IRQ	TOM3_CH12_IRQ	TOM3_CH11_IRQ	TOM3_CH10_IRQ	TOM3_CH9_IRQ	TOM3_CH8_IRQ	TOM3_CH7_IRQ	TOM3_CH6_IRQ	TOM3_CH5_IRQ	TOM3_CH4_IRQ	TOM3_CH3_IRQ	TOM3_CH2_IRQ	TOM3_CH1_IRQ	TOM3_CH0_IRQ	TOM2_CH15_IRQ	TOM2_CH14_IRQ	TOM2_CH13_IRQ	TOM2_CH12_IRQ	TOM2_CH11_IRQ	TOM2_CH10_IRQ	TOM2_CH9_IRQ	TOM2_CH8_IRQ	TOM2_CH7_IRQ	TOM2_CH6_IRQ	TOM2_CH5_IRQ	TOM2_CH4_IRQ	TOM2_CH3_IRQ	TOM2_CH2_IRQ	TOM2_CH1_IRQ	TOM2_CH0_IRQ
Mode	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Initial Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit 0 **TOM2_CH0_IRQ:** TOM2 channel 0 shared interrupt

0 = no interrupt occurred
1 = interrupt was raised by the corresponding sub-module

Note: This bit is only set, when the interrupt is enabled in the interrupt enable register of the corresponding sub-module.

Note: Set this bit represents an OR function of the two interrupt sources *TOM_CCU0TCx_IRQ* or *TOM_CCU1TCx_IRQ* of TOM instance 2 channel x.

Bit 1 **TOM2_CH1_IRQ:** TOM2 channel 1 shared interrupt. See bit 0.

Bit 2 **TOM2_CH2_IRQ:** TOM2 channel 2 shared interrupt. See bit 0.

Bit 3 **TOM2_CH3_IRQ:** TOM2 channel 3 shared interrupt. See bit 0.

Bit 4 **TOM2_CH4_IRQ:** TOM2 channel 4 shared interrupt. See bit 0.

Bit 5 **TOM2_CH5_IRQ:** TOM2 channel 5 shared interrupt. See bit 0.

Bit 6 **TOM2_CH6_IRQ:** TOM2 channel 6 shared interrupt. See bit 0.

Bit 7 **TOM2_CH7_IRQ:** TOM2 channel 7 shared interrupt. See bit 0.

Bit 8 **TOM2_CH8_IRQ:** TOM2 channel 8 shared interrupt. See bit 0.

Bit 9 **TOM2_CH9_IRQ:** TOM2 channel 9 shared interrupt. See bit 0.

Bit 10 **TOM2_CH10_IRQ:** TOM2 channel 10 shared interrupt. See bit 0.

Bit 11 **TOM2_CH11_IRQ:** TOM2 channel 11 shared interrupt. See bit 0.

Bit 12 **TOM2_CH12_IRQ:** TOM2 channel 12 shared interrupt. See bit 0.

Bit 13 **TOM2_CH13_IRQ:** TOM2 channel 13 shared interrupt. See bit 0.

Bit 14 **TOM2_CH14_IRQ:** TOM2 channel 14 shared interrupt. See bit 0.

Bit 15 **TOM2_CH15_IRQ:** TOM2 channel 15 shared interrupt. See bit 0.

Bit 16 **TOM3_CH0_IRQ:** TOM3 channel 0 shared interrupt. See bit 0.

Bit 17 **TOM3_CH1_IRQ:** TOM3 channel 1 shared interrupt. See bit 0.

Bit 18 **TOM3_CH2_IRQ:** TOM3 channel 2 shared interrupt. See bit 0.

Bit 19 **TOM3_CH3_IRQ:** TOM3 channel 3 shared interrupt. See bit 0.

Bit 20 **TOM3_CH4_IRQ:** TOM3 channel 4 shared interrupt. See bit 0.

Bit 21 **TOM3_CH5_IRQ:** TOM3 channel 5 shared interrupt. See bit 0.

Bit 22 **TOM3_CH6_IRQ:** TOM3 channel 6 shared interrupt. See bit 0.

Bit 23 **TOM3_CH7_IRQ:** TOM3 channel 7 shared interrupt. See bit 0.

Bit 24 **TOM3_CH8_IRQ:** TOM3 channel 8 shared interrupt. See bit 0.

Bit 25 **TOM3_CH9_IRQ:** TOM3 channel 9 shared interrupt. See bit 0.

Bit 26 **TOM3_CH10_IRQ:** TOM3 channel 10 shared interrupt. See bit 0.

Bit 27 **TOM3_CH11_IRQ:** TOM3 channel 11 shared interrupt. See bit 0.

Bit 28 **TOM3_CH12_IRQ:** TOM3 channel 12 shared interrupt. See bit 0.

Bit 29 **TOM3_CH13_IRQ:** TOM3 channel 13 shared interrupt. See bit 0.

Bit 30 **TOM3_CH14_IRQ:** TOM3 channel 14 shared interrupt. See bit 0.

Bit 31 **TOM3_CH15_IRQ:** TOM3 channel 15 shared interrupt. See bit 0.

20.5.9 Register ICM_IRQG_8

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	TOM5_CH15_IRQ	TOM5_CH14_IRQ	TOM5_CH13_IRQ	TOM5_CH12_IRQ	TOM5_CH11_IRQ	TOM5_CH10_IRQ	TOM5_CH9_IRQ	TOM5_CH8_IRQ	TOM5_CH7_IRQ	TOM5_CH6_IRQ	TOM5_CH5_IRQ	TOM5_CH4_IRQ	TOM5_CH3_IRQ	TOM5_CH2_IRQ	TOM5_CH1_IRQ	TOM5_CH0_IRQ	TOM4_CH15_IRQ	TOM4_CH14_IRQ	TOM4_CH13_IRQ	TOM4_CH12_IRQ	TOM4_CH11_IRQ	TOM4_CH10_IRQ	TOM4_CH9_IRQ	TOM4_CH8_IRQ	TOM4_CH7_IRQ	TOM4_CH6_IRQ	TOM4_CH5_IRQ	TOM4_CH4_IRQ	TOM4_CH3_IRQ	TOM4_CH2_IRQ	TOM4_CH1_IRQ	TOM4_CH0_IRQ
Mode	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Initial Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit 0 **TOM4_CH0_IRQ:** TOM4 channel 0 shared interrupt
0 = no interrupt occurred
1 = interrupt was raised by the corresponding sub-module
Note: This bit is only set, when the interrupt is enabled in the interrupt enable register of the corresponding sub-module.

Note: Set this bit represents an OR function of the two interrupt sources *TOM_CCU0TCx_IRQ* or *TOM_CCU1TCx_IRQ* of TOM instance 4 channel x.

Bit 1 **TOM4_CH1_IRQ:** TOM4 channel 1 shared interrupt. See bit 0.
Bit 2 **TOM4_CH2_IRQ:** TOM4 channel 2 shared interrupt. See bit 0.
Bit 3 **TOM4_CH3_IRQ:** TOM4 channel 3 shared interrupt. See bit 0.
Bit 4 **TOM4_CH4_IRQ:** TOM4 channel 4 shared interrupt. See bit 0.
Bit 5 **TOM4_CH5_IRQ:** TOM4 channel 5 shared interrupt. See bit 0.
Bit 6 **TOM4_CH6_IRQ:** TOM4 channel 6 shared interrupt. See bit 0.
Bit 7 **TOM4_CH7_IRQ:** TOM4 channel 7 shared interrupt. See bit 0.
Bit 8 **TOM4_CH8_IRQ:** TOM4 channel 8 shared interrupt. See bit 0.
Bit 9 **TOM4_CH9_IRQ:** TOM4 channel 9 shared interrupt. See bit 0.
Bit 10 **TOM4_CH10_IRQ:** TOM4 channel 10 shared interrupt. See bit 0.
Bit 11 **TOM4_CH11_IRQ:** TOM4 channel 11 shared interrupt. See bit 0.
Bit 12 **TOM4_CH12_IRQ:** TOM4 channel 12 shared interrupt. See bit 0.
Bit 13 **TOM4_CH13_IRQ:** TOM4 channel 13 shared interrupt. See bit 0.
Bit 14 **TOM4_CH14_IRQ:** TOM4 channel 14 shared interrupt. See bit 0.
Bit 15 **TOM4_CH15_IRQ:** TOM4 channel 15 shared interrupt. See bit 0.
Bit 16 **TOM5_CH0_IRQ:** TOM5 channel 0 shared interrupt. See bit 0.
Bit 17 **TOM5_CH1_IRQ:** TOM5 channel 1 shared interrupt. See bit 0.
Bit 18 **TOM5_CH2_IRQ:** TOM5 channel 2 shared interrupt. See bit 0.
Bit 19 **TOM5_CH3_IRQ:** TOM5 channel 3 shared interrupt. See bit 0.
Bit 20 **TOM5_CH4_IRQ:** TOM5 channel 4 shared interrupt. See bit 0.
Bit 21 **TOM5_CH5_IRQ:** TOM5 channel 5 shared interrupt. See bit 0.
Bit 22 **TOM5_CH6_IRQ:** TOM5 channel 6 shared interrupt. See bit 0.
Bit 23 **TOM5_CH7_IRQ:** TOM5 channel 7 shared interrupt. See bit 0.
Bit 24 **TOM5_CH8_IRQ:** TOM5 channel 8 shared interrupt. See bit 0.

Bit 25 **TOM5_CH9_IRQ:** TOM5 channel 9 shared interrupt. See bit 0.
Bit 26 **TOM5_CH10_IRQ:** TOM5 channel 10 shared interrupt. See bit 0.
Bit 27 **TOM5_CH11_IRQ:** TOM5 channel 11 shared interrupt. See bit 0.
Bit 28 **TOM5_CH12_IRQ:** TOM5 channel 12 shared interrupt. See bit 0.
Bit 29 **TOM5_CH13_IRQ:** TOM5 channel 13 shared interrupt. See bit 0.
Bit 30 **TOM5_CH14_IRQ:** TOM5 channel 14 shared interrupt. See bit 0.
Bit 31 **TOM5_CH15_IRQ:** TOM5 channel 15 shared interrupt. See bit 0.

20.5.10 Register ICM_IRQG_9

Address Offset:	**see Appendix B**																**Initial Value:**					**0x0000_0000**										
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	ATOM3_CH7_IRQ	ATOM3_CH6_IRQ	ATOM3_CH5_IRQ	ATOM3_CH4_IRQ	ATOM3_CH3_IRQ	ATOM3_CH2_IRQ	ATOM3_CH1_IRQ	ATOM3_CH0_IRQ	ATOM2_CH7_IRQ	ATOM2_CH6_IRQ	ATOM2_CH5_IRQ	ATOM2_CH4_IRQ	ATOM2_CH3_IRQ	ATOM2_CH2_IRQ	ATOM2_CH1_IRQ	ATOM2_CH0_IRQ	ATOM1_CH7_IRQ	ATOM1_CH6_IRQ	ATOM1_CH5_IRQ	ATOM1_CH4_IRQ	ATOM1_CH3_IRQ	ATOM1_CH2_IRQ	ATOM1_CH1_IRQ	ATOM1_CH0_IRQ	ATOM0_CH7_IRQ	ATOM0_CH6_IRQ	ATOM0_CH5_IRQ	ATOM0_CH4_IRQ	ATOM0_CH3_IRQ	ATOM0_CH2_IRQ	ATOM0_CH1_IRQ	ATOM0_CH0_IRQ
Mode	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Initial Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit 0 **ATOM0_CH0_IRQ:** ATOM0 channel 0 shared interrupt
0 = no interrupt occurred
1 = interrupt was raised by the corresponding sub-module
Note: This bit is only set, when the interrupt is enabled in the interrupt enable register of the corresponding sub-module.

Note: Set this bit represents an OR function of the two interrupt sources *CCU0TCx_IRQ* or *CCU1TCx_IRQ* of ATOM instance 0 channel x.

Bit 1 **ATOM0_CH1_IRQ:** ATOM0 channel 1 shared interrupt. See bit 0.
Bit 2 **ATOM0_CH2_IRQ:** ATOM0 channel 2 shared interrupt. See bit 0.
Bit 3 **ATOM0_CH3_IRQ:** ATOM0 channel 3 shared interrupt. See bit 0.
Bit 4 **ATOM0_CH4_IRQ:** ATOM0 channel 4 shared interrupt. See bit 0.
Bit 5 **ATOM0_CH5_IRQ:** ATOM0 channel 5 shared interrupt. See bit 0.
Bit 6 **ATOM0_CH6_IRQ:** ATOM0 channel 6 shared interrupt. See bit 0.
Bit 7 **ATOM0_CH7_IRQ:** ATOM0 channel 7 shared interrupt. See bit 0.
Bit 8 **ATOM1_CH0_IRQ:** ATOM1 channel 0 shared interrupt. See bit 0.
Bit 9 **ATOM1_CH1_IRQ:** ATOM1 channel 1 shared interrupt. See bit 0.
Bit 10 **ATOM1_CH2_IRQ:** ATOM1 channel 2 shared interrupt. See bit 0.
Bit 11 **ATOM1_CH3_IRQ:** ATOM1 channel 3 shared interrupt. See bit 0.
Bit 12 **ATOM1_CH4_IRQ:** ATOM1 channel 4 shared interrupt. See bit 0.
Bit 13 **ATOM1_CH5_IRQ:** ATOM1 channel 5 shared interrupt. See bit 0.
Bit 14 **ATOM1_CH6_IRQ:** ATOM1 channel 6 shared interrupt. See bit 0.

Bit 15 **ATOM1_CH7_IRQ:** ATOM1 channel 7 shared interrupt. See bit 0.
Bit 16 **ATOM2_CH0_IRQ:** ATOM2 channel 0 shared interrupt. See bit 0.
Bit 17 **ATOM2_CH1_IRQ:** ATOM2 channel 1 shared interrupt. See bit 0.
Bit 18 **ATOM2_CH2_IRQ:** ATOM2 channel 2 shared interrupt. See bit 0.
Bit 19 **ATOM2_CH3_IRQ:** ATOM2 channel 3 shared interrupt. See bit 0.
Bit 20 **ATOM2_CH4_IRQ:** ATOM2 channel 4 shared interrupt. See bit 0.
Bit 21 **ATOM2_CH5_IRQ:** ATOM2 channel 5 shared interrupt. See bit 0.
Bit 22 **ATOM2_CH6_IRQ:** ATOM2 channel 6 shared interrupt. See bit 0.
Bit 23 **ATOM2_CH7_IRQ:** ATOM2 channel 7 shared interrupt. See bit 0.
Bit 24 **ATOM3_CH0_IRQ:** ATOM3 channel 0 shared interrupt. See bit 0.
Bit 25 **ATOM3_CH1_IRQ:** ATOM3 channel 1 shared interrupt. See bit 0.
Bit 26 **ATOM3_CH2_IRQ:** ATOM3 channel 2 shared interrupt. See bit 0.
Bit 27 **ATOM3_CH3_IRQ:** ATOM3 channel 3 shared interrupt. See bit 0.
Bit 28 **ATOM3_CH4_IRQ:** ATOM3 channel 4 shared interrupt. See bit 0.
Bit 29 **ATOM3_CH5_IRQ:** ATOM3 channel 5 shared interrupt. See bit 0.
Bit 30 **ATOM3_CH6_IRQ:** ATOM3 channel 6 shared interrupt. See bit 0.
Bit 31 **ATOM3_CH7_IRQ:** ATOM3 channel 7 shared interrupt. See bit 0.

20.5.11 Register ICM_IRQG_10

Address Offset:	**see Appendix B**																**Initial Value: 0x0000_0000**															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	ATOM7_CH7_IRQ	ATOM7_CH6_IRQ	ATOM7_CH5_IRQ	ATOM7_CH4_IRQ	ATOM7_CH3_IRQ	ATOM7_CH2_IRQ	ATOM7_CH1_IRQ	ATOM7_CH0_IRQ	ATOM6_CH7_IRQ	ATOM6_CH6_IRQ	ATOM6_CH5_IRQ	ATOM6_CH4_IRQ	ATOM6_CH3_IRQ	ATOM6_CH2_IRQ	ATOM6_CH1_IRQ	ATOM6_CH0_IRQ	ATOM5_CH7_IRQ	ATOM5_CH6_IRQ	ATOM5_CH5_IRQ	ATOM5_CH4_IRQ	ATOM5_CH3_IRQ	ATOM5_CH2_IRQ	ATOM5_CH1_IRQ	ATOM5_CH0_IRQ	ATOM4_CH7_IRQ	ATOM4_CH6_IRQ	ATOM4_CH5_IRQ	ATOM4_CH4_IRQ	ATOM4_CH3_IRQ	ATOM4_CH2_IRQ	ATOM4_CH1_IRQ	ATOM4_CH0_IRQ
Mode	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Initial Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit 0 **ATOM4_CH0_IRQ:** ATOM4 channel 0 shared interrupt
0 = no interrupt occurred
1 = interrupt was raised by the corresponding sub-module
Note: This bit is only set, when the interrupt is enabled in the interrupt enable register of the corresponding sub-module.

Note: Set this bit represents an OR function of the two interrupt sources *CCU0TCx_IRQ* or *CCU1TCx_IRQ* of ATOM instance 4 channel x.

Bit 1 **ATOM4_CH1_IRQ:** ATOM4 channel 1 shared interrupt. See bit 0.
Bit 2 **ATOM4_CH2_IRQ:** ATOM4 channel 2 shared interrupt. See bit 0.
Bit 3 **ATOM4_CH3_IRQ:** ATOM4 channel 3 shared interrupt. See bit 0.
Bit 4 **ATOM4_CH4_IRQ:** ATOM4 channel 4 shared interrupt. See bit 0.

Bit 5 **ATOM4_CH5_IRQ:** ATOM4 channel 5 shared interrupt. See bit 0.
Bit 6 **ATOM4_CH6_IRQ:** ATOM4 channel 6 shared interrupt. See bit 0.
Bit 7 **ATOM4_CH7_IRQ:** ATOM4 channel 7 shared interrupt. See bit 0.
Bit 8 **ATOM5_CH0_IRQ:** ATOM5 channel 0 shared interrupt. See bit 0.
Bit 9 **ATOM5_CH1_IRQ:** ATOM5 channel 1 shared interrupt. See bit 0.
Bit 10 **ATOM5_CH2_IRQ:** ATOM5 channel 2 shared interrupt. See bit 0.
Bit 11 **ATOM5_CH3_IRQ:** ATOM5 channel 3 shared interrupt. See bit 0.
Bit 12 **ATOM5_CH4_IRQ:** ATOM5 channel 4 shared interrupt. See bit 0.
Bit 13 **ATOM5_CH5_IRQ:** ATOM5 channel 5 shared interrupt. See bit 0.
Bit 14 **ATOM5_CH6_IRQ:** ATOM5 channel 6 shared interrupt. See bit 0.
Bit 15 **ATOM5_CH7_IRQ:** ATOM5 channel 7 shared interrupt. See bit 0.
Bit 16 **ATOM6_CH0_IRQ:** ATOM6 channel 0 shared interrupt. See bit 0.
Bit 17 **ATOM6_CH1_IRQ:** ATOM6 channel 1 shared interrupt. See bit 0.
Bit 18 **ATOM6_CH2_IRQ:** ATOM6 channel 2 shared interrupt. See bit 0.
Bit 19 **ATOM6_CH3_IRQ:** ATOM6 channel 3 shared interrupt. See bit 0.
Bit 20 **ATOM6_CH4_IRQ:** ATOM6 channel 4 shared interrupt. See bit 0.
Bit 21 **ATOM6_CH5_IRQ:** ATOM6 channel 5 shared interrupt. See bit 0.
Bit 22 **ATOM6_CH6_IRQ:** ATOM6 channel 6 shared interrupt. See bit 0.
Bit 23 **ATOM6_CH7_IRQ:** ATOM6 channel 7 shared interrupt. See bit 0.
Bit 24 **ATOM7_CH0_IRQ:** ATOM7 channel 0 shared interrupt. See bit 0.
Bit 25 **ATOM7_CH1_IRQ:** ATOM7 channel 1 shared interrupt. See bit 0.
Bit 26 **ATOM7_CH2_IRQ:** ATOM7 channel 2 shared interrupt. See bit 0.
Bit 27 **ATOM7_CH3_IRQ:** ATOM7 channel 3 shared interrupt. See bit 0.
Bit 28 **ATOM7_CH4_IRQ:** ATOM7 channel 4 shared interrupt. See bit 0.
Bit 29 **ATOM7_CH5_IRQ:** ATOM7 channel 5 shared interrupt. See bit 0.
Bit 30 **ATOM7_CH6_IRQ:** ATOM7 channel 6 shared interrupt. See bit 0.
Bit 31 **ATOM7_CH7_IRQ:** ATOM7 channel 7 shared interrupt. See bit 0.

20.5.12 Register ICM_IRQG_11

Address Offset:	**see Appendix B**																**Initial Value: 0x0000_0000**															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	ATOM11_CH7_IR	ATOM11_CH6_IR	ATOM11_CH5_IR	ATOM11_CH4_IR	ATOM11_CH3_IR	ATOM11_CH2_IR	ATOM11_CH1_IR	ATOM11_CH0_IR	ATOM10_CH7_IR	ATOM10_CH6_IR	ATOM10_CH5_IR	ATOM10_CH4_IR	ATOM10_CH3_IR	ATOM10_CH2_IR	ATOM10_CH1_IR	ATOM10_CH0_IR	ATOM9_CH7_IRQ	ATOM9_CH6_IRQ	ATOM9_CH5_IRQ	ATOM9_CH4_IRQ	ATOM9_CH3_IRQ	ATOM9_CH2_IRQ	ATOM9_CH1_IRQ	ATOM9_CH0_IRQ	ATOM8_CH7_IRQ	ATOM8_CH6_IRQ	ATOM8_CH5_IRQ	ATOM8_CH4_IRQ	ATOM8_CH3_IRQ	ATOM8_CH2_IRQ	ATOM8_CH1_IRQ	ATOM8_CH0_IRQ
Mode	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Initial Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit 0 **ATOM8_CH0_IRQ:** ATOM8 channel 0 shared interrupt
0 = no interrupt occurred
1 = interrupt was raised by the corresponding sub-module

Note: This bit is only set, when the interrupt is enabled in the interrupt enable register of the corresponding sub-module.

Note: Set this bit represents an OR function of the two interrupt sources *CCU0TCx_IRQ* or *CCU1TCx_IRQ* of ATOM instance 8 channel x.

Bit 1	**ATOM8_CH1_IRQ:** ATOM8 channel 1 shared interrupt. See bit 0.
Bit 2	**ATOM8_CH2_IRQ:** ATOM8 channel 2 shared interrupt. See bit 0.
Bit 3	**ATOM8_CH3_IRQ:** ATOM8 channel 3 shared interrupt. See bit 0.
Bit 4	**ATOM8_CH4_IRQ:** ATOM8 channel 4 shared interrupt. See bit 0.
Bit 5	**ATOM8_CH5_IRQ:** ATOM8 channel 5 shared interrupt. See bit 0.
Bit 6	**ATOM8_CH6_IRQ:** ATOM8 channel 6 shared interrupt. See bit 0.
Bit 7	**ATOM8_CH7_IRQ:** ATOM8 channel 7 shared interrupt. See bit 0.
Bit 8	**ATOM9_CH0_IRQ:** ATOM9 channel 0 shared interrupt. See bit 0.
Bit 9	**ATOM9_CH1_IRQ:** ATOM9 channel 1 shared interrupt. See bit 0.
Bit 10	**ATOM9_CH2_IRQ:** ATOM9 channel 2 shared interrupt. See bit 0.
Bit 11	**ATOM9_CH3_IRQ:** ATOM9 channel 3 shared interrupt. See bit 0.
Bit 12	**ATOM9_CH4_IRQ:** ATOM9 channel 4 shared interrupt. See bit 0.
Bit 13	**ATOM9_CH5_IRQ:** ATOM9 channel 5 shared interrupt. See bit 0.
Bit 14	**ATOM9_CH6_IRQ:** ATOM9 channel 6 shared interrupt. See bit 0.
Bit 15	**ATOM9_CH7_IRQ:** ATOM9 channel 7 shared interrupt. See bit 0.
Bit 16	**ATOM10_CH0_IRQ:** ATOM10 channel 0 shared interrupt. See bit 0.
Bit 17	**ATOM10_CH1_IRQ:** ATOM10 channel 1 shared interrupt. See bit 0.
Bit 18	**ATOM10_CH2_IRQ:** ATOM10 channel 2 shared interrupt. See bit 0.
Bit 19	**ATOM10_CH3_IRQ:** ATOM10 channel 3 shared interrupt. See bit 0.
Bit 20	**ATOM10_CH4_IRQ:** ATOM10 channel 4 shared interrupt. See bit 0.
Bit 21	**ATOM10_CH5_IRQ:** ATOM10 channel 5 shared interrupt. See bit 0.
Bit 22	**ATOM10_CH6_IRQ:** ATOM10 channel 6 shared interrupt. See bit 0.
Bit 23	**ATOM10_CH7_IRQ:** ATOM10 channel 7 shared interrupt. See bit 0.
Bit 24	**ATOM11_CH0_IRQ:** ATOM11 channel 0 shared interrupt. See bit 0.
Bit 25	**ATOM11_CH1_IRQ:** ATOM11 channel 1 shared interrupt. See bit 0.
Bit 26	**ATOM11_CH2_IRQ:** ATOM11 channel 2 shared interrupt. See bit 0.
Bit 27	**ATOM11_CH3_IRQ:** ATOM11 channel 3 shared interrupt. See bit 0.
Bit 28	**ATOM11_CH4_IRQ:** ATOM11 channel 4 shared interrupt. See bit 0.
Bit 29	**ATOM11_CH5_IRQ:** ATOM11 channel 5 shared interrupt. See bit 0.
Bit 30	**ATOM11_CH6_IRQ:** ATOM11 channel 6 shared interrupt. See bit 0.
Bit 31	**ATOM11_CH7_IRQ:** ATOM11 channel 7 shared interrupt. See bit 0.

20.5.13 Register ICM_IRQG_MEI

Address Offset:	**see Appendix B**																**Initial Value: 0x0000_0000**															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved						DPLL_EIRQ	CMP_EIRQ	SPE3_EIRQ	SPE2_EIRQ	SPE1_EIRQ	SPE0_EIRQ	MCS7_EIRQ	MCS6_EIRQ	MCS5_EIRQ	MCS4_EIRQ	MCS3_EIRQ	MCS2_EIRQ	MCS1_EIRQ	MCS0_EIRQ	TIM7_EIRQ	TIM6_EIRQ	TIM5_EIRQ	TIM4_EIRQ	TIM3_EIRQ	TIM2_EIRQ	TIM1_EIRQ	TIM0_EIRQ	FIFO1_EIRQ	FIFO0_EIRQ	BRC_EIRQ	GTM_EIRQ
Mode	R						R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Initial Value	0x00						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit 0 **AEI_EIRQ:** AEI Error interrupt request
0 = no interrupt occurred
1 = interrupt was raised by the corresponding sub-module
Note: This bit is only set, when the error interrupt is enabled in the error interrupt enable register of the corresponding sub-module.

Note: Set this bit represents an OR function of the interrupt sources *AEI_TO_XPT_EIRQ*, *AEI_USP_ADDR_EIRQ*, *AEI_IM_ADDR_EIRQ*, *AEI_USP_BE_EIRQ*, *AEIM_USP_ADDR_EIRQ*, *AEIM_IM_ADDR_EIRQ*, *AEIM_USP_BE_EIRQ, CLK_EN_ERR* or *CLK_PER_ERR*.

Bit 1 **BRC_EIRQ:** BRC error interrupt. See bit 0.
Bit 2 **FIFO0_EIRQ:** FIFO0 error interrupt. See bit 0.
Bit 3 **FIFO1_EIRQ:** FIFO1 error interrupt. See bit 0.
Bit 4 **TIM0_EIRQ:** TIM0 error interrupt. See bit 0.
Bit 5 **TIM1_EIRQ:** TIM1 error interrupt. See bit 0.
Bit 6 **TIM2_EIRQ:** TIM2 error interrupt. See bit 0.
Bit 7 **TIM3_EIRQ:** TIM3 error interrupt. See bit 0.
Bit 8 **TIM4_EIRQ:** TIM4 error interrupt. See bit 0.
Bit 9 **TIM5_EIRQ:** TIM5 error interrupt. See bit 0.
Bit 10 **TIM6_EIRQ:** TIM6 error interrupt. See bit 0.
Bit 11 **TIM7_EIRQ:** TIM7 error interrupt. See bit 0.
Bit 12 **MCS0_EIRQ:** MCS0 error interrupt. See bit 0.
Bit 13 **MCS1_EIRQ:** MCS1 error interrupt. See bit 0.
Bit 14 **MCS2_EIRQ:** MCS2 error interrupt. See bit 0.
Bit 15 **MCS3_EIRQ:** MCS3 error interrupt. See bit 0.
Bit 16 **MCS4_EIRQ:** MCS4 error interrupt. See bit 0.
Bit 17 **MCS5_EIRQ:** MCS5 error interrupt. See bit 0.
Bit 18 **MCS6_EIRQ:** MCS6 error interrupt. See bit 0.
Bit 19 **MCS7_EIRQ:** MCS7 error interrupt. See bit 0.
Bit 20 **SPE0_EIRQ:** SPE0 error interrupt. See bit 0.
Bit 21 **SPE1_EIRQ:** SPE1 error interrupt. See bit 0.
Bit 22 **SPE2_EIRQ:** SPE2 error interrupt. See bit 0.

Bit 23 **SPE3_EIRQ:** SPE3 error interrupt. See bit 0.

Bit 24 **CMP_EIRQ:** CMP error interrupt. See bit 0.

Bit 25 **DPLL_EIRQ:** DPLL error interrupt. See bit 0.

Bit 31:26 **Reserved:** Read as zero, should be written as zero.
Note: Read as zero, should be written as zero

20.5.14 Register ICM_IRQG_CEI0

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								FIFO2_CH7_EIRQ	FIFO2_CH6_EIRQ	FIFO2_CH5_EIRQ	FIFO2_CH4_EIRQ	FIFO2_CH3_EIRQ	FIFO2_CH2_EIRQ	FIFO2_CH1_EIRQ	FIFO2_CH0_EIRQ	FIFO1_CH7_EIRQ	FIFO1_CH6_EIRQ	FIFO1_CH5_EIRQ	FIFO1_CH4_EIRQ	FIFO1_CH3_EIRQ	FIFO1_CH2_EIRQ	FIFO1_CH1_EIRQ	FIFO1_CH0_EIRQ	FIFO0_CH7_EIRQ	FIFO0_CH6_EIRQ	FIFO0_CH5_EIRQ	FIFO0_CH4_EIRQ	FIFO0_CH3_EIRQ	FIFO0_CH2_EIRQ	FIFO0_CH1_EIRQ	FIFO0_CH0_EIRQ
Mode	R								R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Initial Value	0x00								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit 0 **FIFO0_CH0_EIRQ:** FIFO0 channel 0 error interrupt
0 = no interrupt occurred
1 = error interrupt was raised by the corresponding sub-module
Note: This bit is only set, when the error interrupt is enabled in the error interrupt enable register of the corresponding sub-module.

Bit 1 **FIFO0_CH1_EIRQ:** FIFO0 channel 1 error interrupt. See bit 0.

Bit 2 **FIFO0_CH2_EIRQ:** FIFO0 channel 2 error interrupt. See bit 0.

Bit 3 **FIFO0_CH3_EIRQ:** FIFO0 channel 3 error interrupt. See bit 0.

Bit 4 **FIFO0_CH4_EIRQ:** FIFO0 channel 4 error interrupt. See bit 0.

Bit 5 **FIFO0_CH5_EIRQ:** FIFO0 channel 5 error interrupt. See bit 0.

Bit 6 **FIFO0_CH6_EIRQ:** FIFO0 channel 6 error interrupt. See bit 0.

Bit 7 **FIFO0_CH7_EIRQ:** FIFO0 channel 7 error interrupt. See bit 0.

Bit 8 **FIFO1_CH0_EIRQ:** FIFO1 channel 0 error interrupt. See bit 0.

Bit 9 **FIFO1_CH1_EIRQ:** FIFO1 channel 1 error interrupt. See bit 0.

Bit 10 **FIFO1_CH2_EIRQ:** FIFO1 channel 2 error interrupt. See bit 0.

Bit 11 **FIFO1_CH3_EIRQ:** FIFO1 channel 3 error interrupt. See bit 0.

Bit 12 **FIFO1_CH4_EIRQ:** FIFO1 channel 4 error interrupt. See bit 0.

Bit 13 **FIFO1_CH5_EIRQ:** FIFO1 channel 5 error interrupt. See bit 0.

Bit 14 **FIFO1_CH6_EIRQ:** FIFO1 channel 6 error interrupt. See bit 0.

Bit 15 **FIFO1_CH7_EIRQ:** FIFO1 channel 7 error interrupt. See bit 0.

Bit 16 **FIFO2_CH0_EIRQ:** FIFO2 channel 0 error interrupt. See bit 0.

Bit 17 **FIFO2_CH1_EIRQ:** FIFO2 channel 1 error interrupt. See bit 0.

Bit 18 **FIFO2_CH2_EIRQ:** FIFO2 channel 2 error interrupt. See bit 0.

Bit 19 **FIFO2_CH3_EIRQ:** FIFO2 channel 3 error interrupt. See bit 0.
Bit 20 **FIFO2_CH4_EIRQ:** FIFO2 channel 4 error interrupt. See bit 0.
Bit 21 **FIFO2_CH5_EIRQ:** FIFO2 channel 5 error interrupt. See bit 0.
Bit 22 **FIFO2_CH6_EIRQ:** FIFO2 channel 6 error interrupt. See bit 0.
Bit 23 **FIFO2_CH7_EIRQ:** FIFO2 channel 7 error interrupt. See bit 0.
Bit 31:24 **Reserved:** Read as zero, should be written as zero.
Note: Read as zero, should be written as zero

20.5.15 Register ICM_IRQG_CEI1

Address Offset:	see Appendix B																Initial Value: 0x0000_0000															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	TIM3_CH7_EIRQ	TIM3_CH6_EIRQ	TIM3_CH5_EIRQ	TIM3_CH4_EIRQ	TIM3_CH3_EIRQ	TIM3_CH2_EIRQ	TIM3_CH1_EIRQ	TIM3_CH0_EIRQ	TIM2_CH7_EIRQ	TIM2_CH6_EIRQ	TIM2_CH5_EIRQ	TIM2_CH4_EIRQ	TIM2_CH3_EIRQ	TIM2_CH2_EIRQ	TIM2_CH1_EIRQ	TIM2_CH0_EIRQ	TIM1_CH7_EIRQ	TIM1_CH6_EIRQ	TIM1_CH5_EIRQ	TIM1_CH4_EIRQ	TIM1_CH3_EIRQ	TIM1_CH2_EIRQ	TIM1_CH1_EIRQ	TIM1_CH0_EIRQ	TIM0_CH7_EIRQ	TIM0_CH6_EIRQ	TIM0_CH5_EIRQ	TIM0_CH4_EIRQ	TIM0_CH3_EIRQ	TIM0_CH2_EIRQ	TIM0_CH1_EIRQ	TIM0_CH0_EIRQ
Mode	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Initial Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit 0 **TIM0_CH0_EIRQ:** TIM0 channel 0 error interrupt
0 = no error interrupt occurred
1 = error interrupt was raised by the corresponding sub-module
Note: This bit is only set, when the error interrupt is enabled in the error interrupt enable register of the corresponding sub-module.

Bit 1 **TIM0_CH1_EIRQ:** TIM0 channel 1 error interrupt. See bit 0.
Bit 2 **TIM0_CH2_EIRQ:** TIM0 channel 2 error interrupt. See bit 0.
Bit 3 **TIM0_CH3_EIRQ:** TIM0 channel 3 error interrupt. See bit 0.
Bit 4 **TIM0_CH4_EIRQ:** TIM0 channel 4 error interrupt. See bit 0.
Bit 5 **TIM0_CH5_EIRQ:** TIM0 channel 5 error interrupt. See bit 0.
Bit 6 **TIM0_CH6_EIRQ:** TIM0 channel 6 error interrupt. See bit 0.
Bit 7 **TIM0_CH7_EIRQ:** TIM0 channel 7 error interrupt. See bit 0.
Bit 8 **TIM1_CH0_EIRQ:** TIM1 channel 0 error interrupt. See bit 0.
Bit 9 **TIM1_CH1_EIRQ:** TIM1 channel 1 error interrupt. See bit 0.
Bit 10 **TIM1_CH2_EIRQ:** TIM1 channel 2 error interrupt. See bit 0.
Bit 11 **TIM1_CH3_EIRQ:** TIM1 channel 3 error interrupt. See bit 0.
Bit 12 **TIM1_CH4_EIRQ:** TIM1 channel 4 error interrupt. See bit 0.
Bit 13 **TIM1_CH5_EIRQ:** TIM1 channel 5 error interrupt. See bit 0.
Bit 14 **TIM1_CH6_EIRQ:** TIM1 channel 6 error interrupt. See bit 0.
Bit 15 **TIM1_CH7_EIRQ:** TIM1 channel 7 error interrupt. See bit 0.
Bit 16 **TIM2_CH0_EIRQ:** TIM2 channel 0 error interrupt. See bit 0.

Bit 17 **TIM2_CH1_EIRQ:** TIM2 channel 1 error interrupt. See bit 0.
Bit 18 **TIM2_CH2_EIRQ:** TIM2 channel 2 error interrupt. See bit 0.
Bit 19 **TIM2_CH3_EIRQ:** TIM2 channel 3 error interrupt. See bit 0.
Bit 20 **TIM2_CH4_EIRQ:** TIM2 channel 4 error interrupt. See bit 0.
Bit 21 **TIM2_CH5_EIRQ:** TIM2 channel 5 error interrupt. See bit 0.
Bit 22 **TIM2_CH6_EIRQ:** TIM2 channel 6 error interrupt. See bit 0.
Bit 23 **TIM2_CH7_EIRQ:** TIM2 channel 7 error interrupt. See bit 0.
Bit 24 **TIM3_CH0_EIRQ:** TIM3 channel 0 error interrupt. See bit 0.
Bit 25 **TIM3_CH1_EIRQ:** TIM3 channel 1 error interrupt. See bit 0.
Bit 26 **TIM3_CH2_EIRQ:** TIM3 channel 2 error interrupt. See bit 0.
Bit 27 **TIM3_CH3_EIRQ:** TIM3 channel 3 error interrupt. See bit 0.
Bit 28 **TIM3_CH4_EIRQ:** TIM3 channel 4 error interrupt. See bit 0.
Bit 29 **TIM3_CH5_EIRQ:** TIM3 channel 5 error interrupt. See bit 0.
Bit 30 **TIM3_CH6_EIRQ:** TIM3 channel 6 error interrupt. See bit 0.
Bit 31 **TIM3_CH7_EIRQ:** TIM3 channel 7 error interrupt. See bit 0.

20.5.16 Register ICM_IRQG_CEI2

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_0000**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	TIM7_CH7_EIRQ	TIM7_CH6_EIRQ	TIM7_CH5_EIRQ	TIM7_CH4_EIRQ	TIM7_CH3_EIRQ	TIM7_CH2_EIRQ	TIM7_CH1_EIRQ	TIM7_CH0_EIRQ	TIM6_CH7_EIRQ	TIM6_CH6_EIRQ	TIM6_CH5_EIRQ	TIM6_CH4_EIRQ	TIM6_CH3_EIRQ	TIM6_CH2_EIRQ	TIM6_CH1_EIRQ	TIM6_CH0_EIRQ	TIM5_CH7_EIRQ	TIM5_CH6_EIRQ	TIM5_CH5_EIRQ	TIM5_CH4_EIRQ	TIM5_CH3_EIRQ	TIM5_CH2_EIRQ	TIM5_CH1_EIRQ	TIM5_CH0_EIRQ	TIM4_CH7_EIRQ	TIM4_CH6_EIRQ	TIM4_CH5_EIRQ	TIM4_CH4_EIRQ	TIM4_CH3_EIRQ	TIM4_CH2_EIRQ	TIM4_CH1_EIRQ	TIM4_CH0_EIRQ
Mode	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Initial Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit 0 **TIM4_CH0_EIRQ:** TIM4 channel 0 error interrupt
0 = no error interrupt occurred
1 = error interrupt was raised by the corresponding sub-module
Note: This bit is only set, when the error interrupt is enabled in the error interrupt enable register of the corresponding sub-module.

Bit 1 **TIM4_CH1_EIRQ:** TIM4 channel 1 error interrupt. See bit 0.
Bit 2 **TIM4_CH2_EIRQ:** TIM4 channel 2 error interrupt. See bit 0.
Bit 3 **TIM4_CH3_EIRQ:** TIM4 channel 3 error interrupt. See bit 0.
Bit 4 **TIM4_CH4_EIRQ:** TIM4 channel 4 error interrupt. See bit 0.
Bit 5 **TIM4_CH5_EIRQ:** TIM4 channel 5 error interrupt. See bit 0.
Bit 6 **TIM4_CH6_EIRQ:** TIM4 channel 6 error interrupt. See bit 0.
Bit 7 **TIM4_CH7_EIRQ:** TIM4 channel 7 error interrupt. See bit 0.
Bit 8 **TIM5_CH0_EIRQ:** TIM5 channel 0 error interrupt. See bit 0.
Bit 9 **TIM5_CH1_EIRQ:** TIM5 channel 1 error interrupt. See bit 0.

Bit 10 **TIM5_CH2_EIRQ:** TIM5 channel 2 error interrupt. See bit 0.
Bit 11 **TIM5_CH3_EIRQ:** TIM5 channel 3 error interrupt. See bit 0.
Bit 12 **TIM5_CH4_EIRQ:** TIM5 channel 4 error interrupt. See bit 0.
Bit 13 **TIM5_CH5_EIRQ:** TIM5 channel 5 error interrupt. See bit 0.
Bit 14 **TIM5_CH6_EIRQ:** TIM5 channel 6 error interrupt. See bit 0.
Bit 15 **TIM5_CH7_EIRQ:** TIM5 channel 7 error interrupt. See bit 0.
Bit 16 **TIM6_CH0_EIRQ:** TIM6 channel 0 error interrupt. See bit 0.
Bit 17 **TIM6_CH1_EIRQ:** TIM6 channel 1 error interrupt. See bit 0.
Bit 18 **TIM6_CH2_EIRQ:** TIM6 channel 2 error interrupt. See bit 0.
Bit 19 **TIM6_CH3_EIRQ:** TIM6 channel 3 error interrupt. See bit 0.
Bit 20 **TIM6_CH4_EIRQ:** TIM6 channel 4 error interrupt. See bit 0.
Bit 21 **TIM6_CH5_EIRQ:** TIM6 channel 5 error interrupt. See bit 0.
Bit 22 **TIM6_CH6_EIRQ:** TIM6 channel 6 error interrupt. See bit 0.
Bit 23 **TIM6_CH7_EIRQ:** TIM6 channel 7 error interrupt. See bit 0.
Bit 24 **TIM7_CH0_EIRQ:** TIM7 channel 0 error interrupt. See bit 0.
Bit 25 **TIM7_CH1_EIRQ:** TIM7 channel 1 error interrupt. See bit 0.
Bit 26 **TIM7_CH2_EIRQ:** TIM7 channel 2 error interrupt. See bit 0.
Bit 27 **TIM7_CH3_EIRQ:** TIM7 channel 3 error interrupt. See bit 0.
Bit 28 **TIM7_CH4_EIRQ:** TIM7 channel 4 error interrupt. See bit 0.
Bit 29 **TIM7_CH5_EIRQ:** TIM7 channel 5 error interrupt. See bit 0.
Bit 30 **TIM7_CH6_EIRQ:** TIM7 channel 6 error interrupt. See bit 0.
Bit 31 **TIM7_CH7_EIRQ:** TIM7 channel 7 error interrupt. See bit 0.

20.5.17 Register ICM_IRQG_CEI3

Address Offset:	**see Appendix B**																**Initial Value: 0x0000_0000**															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	MCS3_CH7_EIRQ	MCS3_CH6_EIRQ	MCS3_CH5_EIRQ	MCS3_CH4_EIRQ	MCS3_CH3_EIRQ	MCS3_CH2_EIRQ	MCS3_CH1_EIRQ	MCS3_CH0_EIRQ	MCS2_CH7_EIRQ	MCS2_CH6_EIRQ	MCS2_CH5_EIRQ	MCS2_CH4_EIRQ	MCS2_CH3_EIRQ	MCS2_CH2_EIRQ	MCS2_CH1_EIRQ	MCS2_CH0_EIRQ	MCS1_CH7_EIRQ	MCS1_CH6_EIRQ	MCS1_CH5_EIRQ	MCS1_CH4_EIRQ	MCS1_CH3_EIRQ	MCS1_CH2_EIRQ	MCS1_CH1_EIRQ	MCS1_CH0_EIRQ	MCS0_CH7_EIRQ	MCS0_CH6_EIRQ	MCS0_CH5_EIRQ	MCS0_CH4_EIRQ	MCS0_CH3_EIRQ	MCS0_CH2_EIRQ	MCS0_CH1_EIRQ	MCS0_CH0_EIRQ
Mode	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Initial Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit 0 **MCS0_CH0_EIRQ:** MCS0 channel 0 error interrupt
0 = no error interrupt occurred
1 = error interrupt was raised by the corresponding sub-module
Note: This bit is only set, when the error interrupt is enabled in the error interrupt enable register of the corresponding sub-module.

Bit 1 **MCS0_CH1_EIRQ:** MCS0 channel 1 error interrupt. See bit 0.
Bit 2 **MCS0_CH2_EIRQ:** MCS0 channel 2 error interrupt. See bit 0.

Bit 3 **MCS0_CH3_EIRQ:** MCS0 channel 3 error interrupt. See bit 0.
Bit 4 **MCS0_CH4_EIRQ:** MCS0 channel 4 error interrupt. See bit 0.
Bit 5 **MCS0_CH5_EIRQ:** MCS0 channel 5 error interrupt. See bit 0.
Bit 6 **MCS0_CH6_EIRQ:** MCS0 channel 6 error interrupt. See bit 0.
Bit 7 **MCS0_CH7_EIRQ:** MCS0 channel 7 error interrupt. See bit 0.
Bit 8 **MCS1_CH0_EIRQ:** MCS1 channel 0 error interrupt. See bit 0.
Bit 9 **MCS1_CH1_EIRQ:** MCS1 channel 1 error interrupt. See bit 0.
Bit 10 **MCS1_CH2_EIRQ:** MCS1 channel 2 error interrupt. See bit 0.
Bit 11 **MCS1_CH3_EIRQ:** MCS1 channel 3 error interrupt. See bit 0.
Bit 12 **MCS1_CH4_EIRQ:** MCS1 channel 4 error interrupt. See bit 0.
Bit 13 **MCS1_CH5_EIRQ:** MCS1 channel 5 error interrupt. See bit 0.
Bit 14 **MCS1_CH6_EIRQ:** MCS1 channel 6 error interrupt. See bit 0.
Bit 15 **MCS1_CH7_EIRQ:** MCS1 channel 7 error interrupt. See bit 0.
Bit 16 **MCS2_CH0_EIRQ:** MCS2 channel 0 error interrupt. See bit 0.
Bit 17 **MCS2_CH1_EIRQ:** MCS2 channel 1 error interrupt. See bit 0.
Bit 18 **MCS2_CH2_EIRQ:** MCS2 channel 2 error interrupt. See bit 0.
Bit 19 **MCS2_CH3_EIRQ:** MCS2 channel 3 error interrupt. See bit 0.
Bit 20 **MCS2_CH4_EIRQ:** MCS2 channel 4 error interrupt. See bit 0.
Bit 21 **MCS2_CH5_EIRQ:** MCS2 channel 5 error interrupt. See bit 0.
Bit 22 **MCS2_CH6_EIRQ:** MCS2 channel 6 error interrupt. See bit 0.
Bit 23 **MCS2_CH7_EIRQ:** MCS2 channel 7 error interrupt. See bit 0.
Bit 24 **MCS3_CH0_EIRQ:** MCS3 channel 0 error interrupt. See bit 0.
Bit 25 **MCS3_CH1_EIRQ:** MCS3 channel 1 error interrupt. See bit 0.
Bit 26 **MCS3_CH2_EIRQ:** MCS3 channel 2 error interrupt. See bit 0.
Bit 27 **MCS3_CH3_EIRQ:** MCS3 channel 3 error interrupt. See bit 0.
Bit 28 **MCS3_CH4_EIRQ:** MCS3 channel 4 error interrupt. See bit 0.
Bit 29 **MCS3_CH5_EIRQ:** MCS3 channel 5 error interrupt. See bit 0.
Bit 30 **MCS3_CH6_EIRQ:** MCS3 channel 6 error interrupt. See bit 0.
Bit 31 **MCS3_CH7_EIRQ:** MCS3 channel 7 error interrupt. See bit 0.

20.5.18 Register ICM_IRQG_CEI4

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_0000**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	MCS7_CH7_EIRQ	MCS7_CH6_EIRQ	MCS7_CH5_EIRQ	MCS7_CH4_EIRQ	MCS7_CH3_EIRQ	MCS7_CH2_EIRQ	MCS7_CH1_EIRQ	MCS7_CH0_EIRQ	MCS6_CH7_EIRQ	MCS6_CH6_EIRQ	MCS6_CH5_EIRQ	MCS6_CH4_EIRQ	MCS6_CH3_EIRQ	MCS6_CH2_EIRQ	MCS6_CH1_EIRQ	MCS6_CH0_EIRQ	MCS5_CH7_EIRQ	MCS5_CH6_EIRQ	MCS5_CH5_EIRQ	MCS5_CH4_EIRQ	MCS5_CH3_EIRQ	MCS5_CH2_EIRQ	MCS5_CH1_EIRQ	MCS5_CH0_EIRQ	MCS4_CH7_EIRQ	MCS4_CH6_EIRQ	MCS4_CH5_EIRQ	MCS4_CH4_EIRQ	MCS4_CH3_EIRQ	MCS4_CH2_EIRQ	MCS4_CH1_EIRQ	MCS4_CH0_EIRQ
Mode	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Initial Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit 0 **MCS4_CH0_EIRQ:** MCS4 channel 0 error interrupt

0 = no error interrupt occurred
1 = error interrupt was raised by the corresponding sub-module
Note: This bit is only set, when the error interrupt is enabled in the error interrupt enable register of the corresponding sub-module.

Bit 1 **MCS4_CH1_EIRQ:** MCS4 channel 1 error interrupt. See bit 0.
Bit 2 **MCS4_CH2_EIRQ:** MCS4 channel 2 error interrupt. See bit 0.
Bit 3 **MCS4_CH3_EIRQ:** MCS4 channel 3 error interrupt. See bit 0.
Bit 4 **MCS4_CH4_EIRQ:** MCS4 channel 4 error interrupt. See bit 0.
Bit 5 **MCS4_CH5_EIRQ:** MCS4 channel 5 error interrupt. See bit 0.
Bit 6 **MCS4_CH6_EIRQ:** MCS4 channel 6 error interrupt. See bit 0.
Bit 7 **MCS4_CH7_EIRQ:** MCS4 channel 7 error interrupt. See bit 0.
Bit 8 **MCS5_CH0_EIRQ:** MCS5 channel 0 error interrupt. See bit 0.
Bit 9 **MCS5_CH1_EIRQ:** MCS5 channel 1 error interrupt. See bit 0.
Bit 10 **MCS5_CH2_EIRQ:** MCS5 channel 2 error interrupt. See bit 0.
Bit 11 **MCS5_CH3_EIRQ:** MCS5 channel 3 error interrupt. See bit 0.
Bit 12 **MCS5_CH4_EIRQ:** MCS5 channel 4 error interrupt. See bit 0.
Bit 13 **MCS5_CH5_EIRQ:** MCS5 channel 5 error interrupt. See bit 0.
Bit 14 **MCS5_CH6_EIRQ:** MCS5 channel 6 error interrupt. See bit 0.
Bit 15 **MCS5_CH7_EIRQ:** MCS5 channel 7 error interrupt. See bit 0.
Bit 16 **MCS6_CH0_EIRQ:** MCS6 channel 0 error interrupt. See bit 0.
Bit 17 **MCS6_CH1_EIRQ:** MCS6 channel 1 error interrupt. See bit 0.
Bit 18 **MCS6_CH2_EIRQ:** MCS6 channel 2 error interrupt. See bit 0.
Bit 19 **MCS6_CH3_EIRQ:** MCS6 channel 3 error interrupt. See bit 0.
Bit 20 **MCS6_CH4_EIRQ:** MCS6 channel 4 error interrupt. See bit 0.
Bit 21 **MCS6_CH5_EIRQ:** MCS6 channel 5 error interrupt. See bit 0.
Bit 22 **MCS6_CH6_EIRQ:** MCS6 channel 6 error interrupt. See bit 0.
Bit 23 **MCS6_CH7_EIRQ:** MCS6 channel 7 error interrupt. See bit 0.
Bit 24 **MCS7_CH0_EIRQ:** MCS7 channel 0 error interrupt. See bit 0.
Bit 25 **MCS7_CH1_EIRQ:** MCS7 channel 1 error interrupt. See bit 0.
Bit 26 **MCS7_CH2_EIRQ:** MCS7 channel 2 error interrupt. See bit 0.
Bit 27 **MCS7_CH3_EIRQ:** MCS7 channel 3 error interrupt. See bit 0.
Bit 28 **MCS7_CH4_EIRQ:** MCS7 channel 4 error interrupt. See bit 0.
Bit 29 **MCS7_CH5_EIRQ:** MCS7 channel 5 error interrupt. See bit 0.
Bit 30 **MCS7_CH6_EIRQ:** MCS7 channel 6 error interrupt. See bit 0.
Bit 31 **MCS7_CH7_EIRQ:** MCS7 channel 7 error interrupt. See bit 0.

20.5.19 Register ICM_IRQG_MCS[i]_CI

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																								MCS_CH7_IRQ	MCS_CH6_IRQ	MCS_CH5_IRQ	MCS_CH4_IRQ	MCS_CH3_IRQ	MCS_CH2_IRQ	MCS_CH1_IRQ	MCS_CH0_IRQ
Mode	R																								R	R	R	R	R	R	R	R
Initial Value	0x000000																								0	0	0	0	0	0	0	0

Bit 0 **MCS_CH0_IRQ:** MCS channel 0 interrupt.
0 = no interrupt occurred
1 = interrupt was raised by the corresponding sub-module
Note: This bit is only set, when the interrupt is enabled in the interrupt enable register of the corresponding sub-module.

Bit 1 **MCS_CH1_IRQ:** MCS channel 1 interrupt. See bit 0.
Bit 2 **MCS_CH2_IRQ:** MCS channel 2 interrupt. See bit 0.
Bit 3 **MCS_CH3_IRQ:** MCS channel 3 interrupt. See bit 0.
Bit 4 **MCS_CH4_IRQ:** MCS channel 4 interrupt. See bit 0.
Bit 5 **MCS_CH5_IRQ:** MCS channel 5 interrupt. See bit 0.
Bit 6 **MCS_CH6_IRQ:** MCS channel 6 interrupt. See bit 0.
Bit 7 **MCS_CH7_IRQ:** MCS channel 7 interrupt. See bit 0.
Bit 31:8 **Reserved:** Read as zero, should be written as zero.
Note: Read as zero, should be written as zero

20.5.20 Register ICM_IRQG_MCS[i]_CEI

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																								MCS_CH7_EIRQ	MCS_CH6_EIRQ	MCS_CH5_EIRQ	MCS_CH4_EIRQ	MCS_CH3_EIRQ	MCS_CH2_EIRQ	MCS_CH1_EIRQ	MCS_CH0_EIRQ
Mode	R																								R	R	R	R	R	R	R	R
Initial Value	0x000000																								0	0	0	0	0	0	0	0

Bit 0 **MCS_CH0_EIRQ:** MCS channel 0 error interrupt.

0 = no interrupt occurred
1 = interrupt was raised by the corresponding sub-module
Note: This bit is only set, when the interrupt is enabled in the interrupt enable register of the corresponding sub-module.

Bit 1 **MCS_CH1_EIRQ:** MCS channel 1 error interrupt. See bit 0.
Bit 2 **MCS_CH2_EIRQ:** MCS channel 2 error interrupt. See bit 0.
Bit 3 **MCS_CH3_EIRQ:** MCS channel 3 error interrupt. See bit 0.
Bit 4 **MCS_CH4_EIRQ:** MCS channel 4 error interrupt. See bit 0.
Bit 5 **MCS_CH5_EIRQ:** MCS channel 5 error interrupt. See bit 0.
Bit 6 **MCS_CH6_EIRQ:** MCS channel 6 error interrupt. See bit 0.
Bit 7 **MCS_CH7_EIRQ:** MCS channel 7 error interrupt. See bit 0.
Bit 31:8 **Reserved:** Read as zero, should be written as zero.
Note: Read as zero, should be written as zero

20.5.21 Register ICM_IRQG_SPE_CI

Address Offset:	**see Appendix B**																**Initial Value: 0x0000_0000**															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																										SPE5_IRQ	SPE4_IRQ	SPE3_IRQ	SPE2_IRQ	SPE1_IRQ	SPE0_IRQ
Mode	R																										R	R	R	R	R	R
Initial Value	0x000000																										0	0	0	0	0	0

Bit 0 **SPE0_IRQ:** SPE channel 0 interrupt.
0 = no interrupt occurred
1 = interrupt was raised by the corresponding sub-module
Note: This bit is only set, when the interrupt is enabled in the interrupt enable register of the corresponding sub-module.

Bit 1 **SPE1_IRQ:** SPE channel 1 interrupt. See bit 0.
Bit 2 **SPE2_IRQ:** SPE channel 2 interrupt. See bit 0.
Bit 3 **SPE3_IRQ:** SPE channel 3 interrupt. See bit 0.
Bit 4 **SPE4_IRQ:** SPE channel 4 interrupt. See bit 0.
Bit 5 **SPE5_IRQ:** SPE channel 5 interrupt. See bit 0.
Bit 31:6 **Reserved:** Read as zero, should be written as zero.
Note: Read as zero, should be written as zero

20.5.22 Register ICM_IRQG_SPE_CEI

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																										SPE5_EIRQ	SPE4_EIRQ	SPE3_EIRQ	SPE2_EIRQ	SPE1_EIRQ	SPE0_EIRQ
Mode	R																										R	R	R	R	R	R
Initial Value	0x0000 00																										0	0	0	0	0	0

Bit 0 **SPE0_EIRQ:** SPE channel 0 error interrupt.
0 = no interrupt occurred
1 = interrupt was raised by the corresponding sub-module
Note: This bit is only set, when the interrupt is enabled in the interrupt enable register of the corresponding sub-module.

Bit 1 **SPE1_EIRQ:** SPE channel 1 error interrupt. See bit 0.
Bit 2 **SPE2_EIRQ:** SPE channel 2 error interrupt. See bit 0.
Bit 3 **SPE3_EIRQ:** SPE channel 3 error interrupt. See bit 0.
Bit 4 **SPE4_EIRQ:** SPE channel 4 error interrupt. See bit 0.
Bit 5 **SPE5_EIRQ:** SPE channel 5 error interrupt. See bit 0.
Bit 31:6 **Reserved:** Read as zero, should be written as zero.
Note: Read as zero, should be written as zero

20.5.23 Register ICM_IRQG_PSM_0_CI

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								PSM_M2_CH7_IR	PSM_M2_CH6_IR	PSM_M2_CH5_IR	PSM_M2_CH4_IR	PSM_M2_CH3_IR	PSM_M2_CH2_IR	PSM_M2_CH1_IR	PSM_M2_CH0_IR	PSM_M1_CH7_IR	PSM_M1_CH6_IR	PSM_M1_CH5_IR	PSM_M1_CH4_IR	PSM_M1_CH3_IR	PSM_M1_CH2_IR	PSM_M1_CH1_IR	PSM_M1_CH0_IR	PSM_M0_CH7_IR	PSM_M0_CH6_IR	PSM_M0_CH5_IR	PSM_M0_CH4_IR	PSM_M0_CH3_IR	PSM_M0_CH2_IR	PSM_M0_CH1_IR	PSM_M0_CH0_IR
Mode	R								R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Initial Value	0x0000								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit 0 **PSM_M0_CH0_IRQ:** PSMm channel 0 shared interrupt (m=4*0+0).

0 = no interrupt occurred
1 = interrupt was raised by the corresponding sub-module
Note: This bit is only set, when the interrupt is enabled in the interrupt enable register of the corresponding sub-module.

Note: Set this bit represents an OR function of the four interrupt sources *FIFO_EMPTY*, *FIFO_FULL*, *FIFO_LOWER_WM* or *FIFO_UPPER_WM* of FIFO instance 0 channel 0.

Bit 1 **PSM_M0_CH1_IRQ:** PSMm channel 1 shared interrupt (m=4*0+0). See bit 0.

Bit 2 **PSM_M0_CH2_IRQ:** PSMm channel 2 shared interrupt (m=4*0+0). See bit 0.

Bit 3 **PSM_M0_CH3_IRQ:** PSMm channel 3 shared interrupt (m=4*0+0). See bit 0.

Bit 4 **PSM_M0_CH4_IRQ:** PSMm channel 4 shared interrupt (m=4*0+0). See bit 0.

Bit 5 **PSM_M0_CH5_IRQ:** PSMm channel 5 shared interrupt (m=4*0+0). See bit 0.

Bit 6 **PSM_M0_CH6_IRQ:** PSMm channel 6 shared interrupt (m=4*0+0). See bit 0.

Bit 7 **PSM_M0_CH7_IRQ:** PSMm channel 7 shared interrupt (m=4*0+0). See bit 0.

Bit 8 **PSM_M1_CH0_IRQ:** PSMm channel 0 shared interrupt (m=4*0+1). See bit 0.

Bit 9 **PSM_M1_CH1_IRQ:** PSMm channel 1 shared interrupt (m=4*0+1). See bit 0.

Bit 10 **PSM_M1_CH2_IRQ:** PSMm channel 2 shared interrupt (m=4*0+1). See bit 0.

Bit 11 **PSM_M1_CH3_IRQ:** PSMm channel 3 shared interrupt (m=4*0+1). See bit 0.

Bit 12 **PSM_M1_CH4_IRQ:** PSMm channel 4 shared interrupt (m=4*0+1). See bit 0.

Bit 13 **PSM_M1_CH5_IRQ:** PSMm channel 5 shared interrupt (m=4*0+1). See bit 0.

Bit 14 **PSM_M1_CH6_IRQ:** PSMm channel 6 shared interrupt (m=4*0+1). See bit 0.

Bit 15 **PSM_M1_CH7_IRQ:** PSMm channel 7 shared interrupt (m=4*0+1). See bit 0.

Bit 16 **PSM_M2_CH0_IRQ:** PSMm channel 0 shared interrupt (m=4*0+2). See bit 0.

Bit 17 **PSM_M2_CH1_IRQ:** PSMm channel 1 shared interrupt (m=4*0+2). See bit 0.

Bit 18 **PSM_M2_CH2_IRQ:** PSMm channel 2 shared interrupt (m=4*0+2). See bit 0.

Bit 19 **PSM_M2_CH3_IRQ:** PSMm channel 3 shared interrupt (m=4*0+2). See bit 0.

Bit 20 **PSM_M2_CH4_IRQ:** PSMm channel 4 shared interrupt (m=4*0+2). See bit 0.

Bit 21 **PSM_M2_CH5_IRQ:** PSMm channel 5 shared interrupt (m=4*0+2). See bit 0.

Bit 22 **PSM_M2_CH6_IRQ:** PSMm channel 6 shared interrupt (m=4*0+2). See bit 0.

Bit 23 **PSM_M2_CH7_IRQ:** PSMm channel 7 shared interrupt (m=4*0+2). See bit 0.

Bit 31:24 **Reserved:** Read as zero, should be written as zero.
Note: Read as zero, should be written as zero

20.5.24 Register ICM_IRQG_PSM_0_CEI

Address Offset:	**see Appendix B**																**Initial Value: 0x0000_0000**															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								PSM_M2_CH7_EI	PSM_M2_CH6_EI	PSM_M2_CH5_EI	PSM_M2_CH4_EI	PSM_M2_CH3_EI	PSM_M2_CH2_EI	PSM_M2_CH1_EI	PSM_M2_CH0_EI	PSM_M1_CH7_EI	PSM_M1_CH6_EI	PSM_M1_CH5_EI	PSM_M1_CH4_EI	PSM_M1_CH3_EI	PSM_M1_CH2_EI	PSM_M1_CH1_EI	PSM_M1_CH0_EI	PSM_M0_CH7_EI	PSM_M0_CH6_EI	PSM_M0_CH5_EI	PSM_M0_CH4_EI	PSM_M0_CH3_EI	PSM_M0_CH2_EI	PSM_M0_CH1_EI	PSM_M0_CH0_EI
Mode	R								R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Initial Value	0x0000								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit 0 **PSM_M0_CH0_EIRQ:** PSMm channel 0 error interrupt (m=4*0+0).
0 = no interrupt occurred
1 = error interrupt was raised by the corresponding sub-module
Note: This bit is only set, when the error interrupt is enabled in the error interrupt enable register of the corresponding sub-module.

Bit 1 **PSM_M0_CH1_EIRQ:** PSMm channel 1 error interrupt (m=4*0+0). See bit 0.

Bit 2 **PSM_M0_CH2_EIRQ:** PSMm channel 2 error interrupt (m=4*0+0). See bit 0.

Bit 3 **PSM_M0_CH3_EIRQ:** PSMm channel 3 error interrupt (m=4*0+0). See bit 0.

Bit 4 **PSM_M0_CH4_EIRQ:** PSMm channel 4 error interrupt (m=4*0+0). See bit 0.

Bit 5 **PSM_M0_CH5_EIRQ:** PSMm channel 5 error interrupt (m=4*0+0). See bit 0.

Bit 6 **PSM_M0_CH6_EIRQ:** PSMm channel 6 error interrupt (m=4*0+0). See bit 0.

Bit 7 **PSM_M0_CH7_EIRQ:** PSMm channel 7 error interrupt (m=4*0+0). See bit 0.

Bit 8 **PSM_M1_CH0_EIRQ:** PSMm channel 0 error interrupt (m=4*0+1). See bit 0.

Bit 9 **PSM_M1_CH1_EIRQ:** PSMm channel 1 error interrupt (m=4*0+1). See bit 0.

Bit 10 **PSM_M1_CH2_EIRQ:** PSMm channel 2 error interrupt (m=4*0+1). See bit 0.

Bit 11 **PSM_M1_CH3_EIRQ:** PSMm channel 3 error interrupt (m=4*0+1). See bit 0.

Bit 12 **PSM_M1_CH4_EIRQ:** PSMm channel 4 error interrupt (m=4*0+1). See bit 0.

Bit 13 **PSM_M1_CH5_EIRQ:** PSMm channel 5 error interrupt (m=4*0+1). See bit 0.

Bit 14 **PSM_M1_CH6_EIRQ:** PSMm channel 6 error interrupt (m=4*0+1). See bit 0.

Bit 15 **PSM_M1_CH7_EIRQ:** PSMm channel 7 error interrupt (m=4*0+1). See bit 0.

Bit 16 **PSM_M2_CH0_EIRQ:** PSMm channel 0 error interrupt (m=4*0+2). See bit 0.

Bit 17 **PSM_M2_CH1_EIRQ:** PSMm channel 1 error interrupt (m=4*0+2). See bit 0.

Bit 18 **PSM_M2_CH2_EIRQ:** PSMm channel 2 error interrupt (m=4*0+2). See bit 0.

Bit 19 **PSM_M2_CH3_EIRQ:** PSMm channel 3 error interrupt (m=4*0+2). See bit 0.

Bit 20 **PSM_M2_CH4_EIRQ:** PSMm channel 4 error interrupt (m=4*0+2). See bit 0.

Bit 21 **PSM_M2_CH5_EIRQ:** PSMm channel 5 error interrupt (m=4*0+2). See bit 0.

Bit 22 **PSM_M2_CH6_EIRQ:** PSMm channel 6 error interrupt (m=4*0+2). See bit 0.

Bit 23 **PSM_M2_CH7_EIRQ:** PSMm channel 7 error interrupt (m=4*0+2). See bit 0.

Bit 31:24 **Reserved:** Read as zero, should be written as zero.
Note: Read as zero, should be written as zero

20.5.25 Register ICM_IRQG_TOM_[k]_CI (k:0..2)

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_0000**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	TOM_M1_CH15_I	TOM_M1_CH14_I	TOM_M1_CH13_I	TOM_M1_CH12_I	TOM_M1_CH11_I	TOM_M1_CH10_I	TOM_M1_CH9_IR	TOM_M1_CH8_IR	TOM_M1_CH7_IR	TOM_M1_CH6_IR	TOM_M1_CH5_IR	TOM_M1_CH4_IR	TOM_M1_CH3_IR	TOM_M1_CH2_IR	TOM_M1_CH1_IR	TOM_M1_CH0_IR	TOM_M0_CH15_I	TOM_M0_CH14_I	TOM_M0_CH13_I	TOM_M0_CH12_I	TOM_M0_CH11_I	TOM_M0_CH10_I	TOM_M0_CH9_IR	TOM_M0_CH8_IR	TOM_M0_CH7_IR	TOM_M0_CH6_IR	TOM_M0_CH5_IR	TOM_M0_CH4_IR	TOM_M0_CH3_IR	TOM_M0_CH2_IR	TOM_M0_CH1_IR	TOM_M0_CH0_IR
Mode	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Initial Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit 0 **TOM_M0_CH0_IRQ:** TOMm channel 0 interrupt (m=2*k+0).
0 = no interrupt occurred
1 = interrupt was raised by the corresponding sub-module
Note: This bit is only set, when the interrupt is enabled in the interrupt enable register of the corresponding sub-module.

Note: Set this bit represents an OR function of the two interrupt sources *TOM_CCU0TCx_IRQ* or *TOM_CCU1TCx_IRQ* of TOM instance 0 channel x.

Bit 1 **TOM_M0_CH1_IRQ:** TOMm channel 1 interrupt (m=2*k+0). See bit 0.
Bit 2 **TOM_M0_CH2_IRQ:** TOMm channel 2 interrupt (m=2*k+0). See bit 0.
Bit 3 **TOM_M0_CH3_IRQ:** TOMm channel 3 interrupt (m=2*k+0). See bit 0.
Bit 4 **TOM_M0_CH4_IRQ:** TOMm channel 4 interrupt (m=2*k+0). See bit 0.
Bit 5 **TOM_M0_CH5_IRQ:** TOMm channel 5 interrupt (m=2*k+0). See bit 0.
Bit 6 **TOM_M0_CH6_IRQ:** TOMm channel 6 interrupt (m=2*k+0). See bit 0.
Bit 7 **TOM_M0_CH7_IRQ:** TOMm channel 7 interrupt (m=2*k+0). See bit 0.
Bit 8 **TOM_M0_CH8_IRQ:** TOMm channel 8 interrupt (m=2*k+0). See bit 0.
Bit 9 **TOM_M0_CH9_IRQ:** TOMm channel 9 interrupt (m=2*k+0). See bit 0.
Bit 10 **TOM_M0_CH10_IRQ:** TOMm channel 10 interrupt (m=2*k+0). See bit 0.
Bit 11 **TOM_M0_CH11_IRQ:** TOMm channel 11 interrupt (m=2*k+0). See bit 0.
Bit 12 **TOM_M0_CH12_IRQ:** TOMm channel 12 interrupt (m=2*k+0). See bit 0.
Bit 13 **TOM_M0_CH13_IRQ:** TOMm channel 13 interrupt (m=2*k+0). See bit 0.
Bit 14 **TOM_M0_CH14_IRQ:** TOMm channel 14 interrupt (m=2*k+0). See bit 0.
Bit 15 **TOM_M0_CH15_IRQ:** TOMm channel 15 interrupt (m=2*k+0). See bit 0.
Bit 16 **TOM_M1_CH0_IRQ:** TOMm channel 0 interrupt (m=2*k+1). See bit 0.
Bit 17 **TOM_M1_CH1_IRQ:** TOMm channel 1 interrupt (m=2*k+1). See bit 0.
Bit 18 **TOM_M1_CH2_IRQ:** TOMm channel 2 interrupt (m=2*k+1). See bit 0.
Bit 19 **TOM_M1_CH3_IRQ:** TOMm channel 3 interrupt (m=2*k+1). See bit 0.
Bit 20 **TOM_M1_CH4_IRQ:** TOMm channel 4 interrupt (m=2*k+1). See bit 0.
Bit 21 **TOM_M1_CH5_IRQ:** TOMm channel 5 interrupt (m=2*k+1). See bit 0.
Bit 22 **TOM_M1_CH6_IRQ:** TOMm channel 6 interrupt (m=2*k+1). See bit 0.
Bit 23 **TOM_M1_CH7_IRQ:** TOMm channel 7 interrupt (m=2*k+1). See bit 0.
Bit 24 **TOM_M1_CH8_IRQ:** TOMm channel 8 interrupt (m=2*k+1). See bit 0.

Bit 25 **TOM_M1_CH9_IRQ:** TOMm channel 9 interrupt (m=2*k+1). See bit 0.
Bit 26 **TOM_M1_CH10_IRQ:** TOMm channel 10 interrupt (m=2*k+1). See bit 0.
Bit 27 **TOM_M1_CH11_IRQ:** TOMm channel 11 interrupt (m=2*k+1). See bit 0.
Bit 28 **TOM_M1_CH12_IRQ:** TOMm channel 12 interrupt (m=2*k+1). See bit 0.
Bit 29 **TOM_M1_CH13_IRQ:** TOMm channel 13 interrupt (m=2*k+1). See bit 0.
Bit 30 **TOM_M1_CH14_IRQ:** TOMm channel 14 interrupt (m=2*k+1). See bit 0.
Bit 31 **TOM_M1_CH15_IRQ:** TOMm channel 15 interrupt (m=2*k+1). See bit 0.

20.5.26 Register ICM_IRQG_ATOM_[k]_CI (k:0..2)

Address Offset:	**see Appendix B**																**Initial Value:**					**0x0000_0000**										
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	ATOM_M3_CH7_I	ATOM_M3_CH6_I	ATOM_M3_CH5_I	ATOM_M3_CH4_I	ATOM_M3_CH3_I	ATOM_M3_CH2_I	ATOM_M3_CH1_I	ATOM_M3_CH0_I	ATOM_M2_CH7_I	ATOM_M2_CH6_I	ATOM_M2_CH5_I	ATOM_M2_CH4_I	ATOM_M2_CH3_I	ATOM_M2_CH2_I	ATOM_M2_CH1_I	ATOM_M2_CH0_I	ATOM_M1_CH7_I	ATOM_M1_CH6_I	ATOM_M1_CH5_I	ATOM_M1_CH4_I	ATOM_M1_CH3_I	ATOM_M1_CH2_I	ATOM_M1_CH1_I	ATOM_M1_CH0_I	ATOM_M0_CH7_I	ATOM_M0_CH6_I	ATOM_M0_CH5_I	ATOM_M0_CH4_I	ATOM_M0_CH3_I	ATOM_M0_CH2_I	ATOM_M0_CH1_I	ATOM_M0_CH0_I
Mode	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Initial Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit 0 **ATOM_M0_CH0_IRQ:** ATOMm channel 0 interrupt (m=4*k+0).
0 = no interrupt occurred
1 = interrupt was raised by the corresponding sub-module
Note: This bit is only set, when the interrupt is enabled in the interrupt enable register of the corresponding sub-module.

Note: Set this bit represents an OR function of the two interrupt sources *CCU0TCx_IRQ* or *CCU1TCx_IRQ* of ATOM instance 0 channel x.

Bit 1 **ATOM_M0_CH1_IRQ:** ATOMm channel 1 interrupt (m=4*k+0). See bit 0.
Bit 2 **ATOM_M0_CH2_IRQ:** ATOMm channel 2 interrupt (m=4*k+0). See bit 0.
Bit 3 **ATOM_M0_CH3_IRQ:** ATOMm channel 3 interrupt (m=4*k+0). See bit 0.
Bit 4 **ATOM_M0_CH4_IRQ:** ATOMm channel 4 interrupt (m=4*k+0). See bit 0.
Bit 5 **ATOM_M0_CH5_IRQ:** ATOMm channel 5 interrupt (m=4*k+0). See bit 0.
Bit 6 **ATOM_M0_CH6_IRQ:** ATOMm channel 6 interrupt (m=4*k+0). See bit 0.
Bit 7 **ATOM_M0_CH7_IRQ:** ATOMm channel 7 interrupt (m=4*k+0). See bit 0.

Bit 8 **ATOM_M1_CH0_IRQ:** ATOMm channel 0 interrupt (m=4*k+1). See bit 0.

Bit 9 **ATOM_M1_CH1_IRQ:** ATOMm channel 1 interrupt (m=4*k+1). See bit 0.

Bit 10 **ATOM_M1_CH2_IRQ:** ATOMm channel 2 interrupt (m=4*k+1). See bit 0.

Bit 11 **ATOM_M1_CH3_IRQ:** ATOMm channel 3 interrupt (m=4*k+1). See bit 0.

Bit 12 **ATOM_M1_CH4_IRQ:** ATOMm channel 4 interrupt (m=4*k+1). See bit 0.

Bit 13 **ATOM_M1_CH5_IRQ:** ATOMm channel 5 interrupt (m=4*k+1). See bit 0.

Bit 14 **ATOM_M1_CH6_IRQ:** ATOMm channel 6 interrupt (m=4*k+1). See bit 0.

Bit 15 **ATOM_M1_CH7_IRQ:** ATOMm channel 7 interrupt (m=4*k+1). See bit 0.

Bit 16 **ATOM_M2_CH0_IRQ:** ATOMm channel 0 interrupt (m=4*k+2). See bit 0.

Bit 17 **ATOM_M2_CH1_IRQ:** ATOMm channel 1 interrupt (m=4*k+2). See bit 0.

Bit 18 **ATOM_M2_CH2_IRQ:** ATOMm channel 2 interrupt (m=4*k+2). See bit 0.

Bit 19 **ATOM_M2_CH3_IRQ:** ATOMm channel 3 interrupt (m=4*k+2). See bit 0.

Bit 20 **ATOM_M2_CH4_IRQ:** ATOMm channel 4 interrupt (m=4*k+2). See bit 0.

Bit 21 **ATOM_M2_CH5_IRQ:** ATOMm channel 5 interrupt (m=4*k+2). See bit 0.

Bit 22 **ATOM_M2_CH6_IRQ:** ATOMm channel 6 interrupt (m=4*k+2). See bit 0.

Bit 23 **ATOM_M2_CH7_IRQ:** ATOMm channel 7 interrupt (m=4*k+2). See bit 0.

Bit 24 **ATOM_M3_CH0_IRQ:** ATOMm channel 0 interrupt (m=4*k+3). See bit 0.

Bit 25 **ATOM_M3_CH1_IRQ:** ATOMm channel 1 interrupt (m=4*k+3). See bit 0.

Bit 26 **ATOM_M3_CH2_IRQ:** ATOMm channel 2 interrupt (m=4*k+3). See bit 0.

Bit 27 **ATOM_M3_CH3_IRQ:** ATOMm channel 3 interrupt (m=4*k+3). See bit 0.

Bit 28 **ATOM_M3_CH4_IRQ:** ATOMm channel 4 interrupt (m=4*k+3). See bit 0.

Bit 29 **ATOM_M3_CH5_IRQ:** ATOMm channel 5 interrupt (m=4*k+3). See bit 0.

Bit 30 **ATOM_M3_CH6_IRQ:** ATOMm channel 6 interrupt (m=4*k+3). See bit 0.

Bit 31 **ATOM_M3_CH7_IRQ:** ATOMm channel 7 interrupt (m=4*k+3).See bit 0.

20.5.27 Register ICM_IRQG_CLS_[k]_MEI (k:0..2)

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved				FIFO_M3_EIRQ	SPE_M3_EIRQ	MCS_M3_EIRQ	TIM_M3_EIRQ	Reserved				FIFO_M2_EIRQ	SPE_M2_EIRQ	MCS_M2_EIRQ	TIM_M2_EIRQ	Reserved				FIFO_M1_EIRQ	SPE_M1_EIRQ	MCS_M1_EIRQ	TIM_M1_EIRQ	Reserved				FIFO_M0_EIRQ	SPE_M0_EIRQ	MCS_M0_EIRQ	TIM_M0_EIRQ
Mode	R				R	R	R	R	R				R	R	R	R	R				R	R	R	R	R				R	R	R	R
Initial Value	0x0				0	0	0	0	0x0				0	0	0	0	0x0				0	0	0	0	0x0				0	0	0	0

Bit 0 **TIM_M0_EIRQ:** error interrupt TIMm_EIRQ (m=4*k+0).
0 = no interrupt occurred
1 = error interrupt was raised by the corresponding sub-module
Note: This bit is only set, when the error interrupt is enabled in the error interrupt enable register of the corresponding sub-module.

Bit 1 **MCS_M0_EIRQ:** error interrupt MCSm_EIRQ (m=4*k+0). See bit 0.
Bit 2 **SPE_M0_EIRQ:** error interrupt SPEm_EIRQ (m=4*k+0). See bit 0.
Bit 3 **FIFO_M0_EIRQ:** error interrupt FIFOm_EIRQ (m=4*k+0). See bit 0.
Bit 7:4 **Reserved:** Read as zero, should be written as zero.
Note: Read as zero, should be written as zero
Bit 8 **TIM_M1_EIRQ:** error interrupt TIMm_EIRQ (m=4*k+1). See bit 0.
Bit 9 **MCS_M1_EIRQ:** error interrupt MCSm_EIRQ (m=4*k+1). See bit 0.
Bit 10 **SPE_M1_EIRQ:** error interrupt SPEm_EIRQ (m=4*k+1). See bit 0.
Bit 11 **FIFO_M1_EIRQ:** error interrupt FIFOm_EIRQ (m=4*k+1). See bit 0.
Bit 15:12 **Reserved:** Read as zero, should be written as zero.
Note: Read as zero, should be written as zero
Bit 16 **TIM_M2_EIRQ:** error interrupt TIMm_EIRQ (m=4*k+2). See bit 0.
Bit 17 **MCS_M2_EIRQ:** error interrupt MCSm_EIRQ (m=4*k+2). See bit 0.
Bit 18 **SPE_M2_EIRQ:** error interrupt SPEm_EIRQ (m=4*k+2). See bit 0.
Bit 19 **FIFO_M2_EIRQ:** error interrupt FIFOm_EIRQ (m=4*k+2). See bit 0.
Bit 23:20 **Reserved:** Read as zero, should be written as zero.
Note: Read as zero, should be written as zero
Bit 24 **TIM_M3_EIRQ:** error interrupt TIMm_EIRQ (m=4*k+3). See bit 0.
Bit 25 **MCS_M3_EIRQ:** error interrupt MCSm_EIRQ (m=4*k+3). See bit 0.
Bit 26 **SPE_M3_EIRQ:** error interrupt SPEm_EIRQ (m=4*k+3). See bit 0.
Bit 27 **FIFO_M3_EIRQ:** error interrupt FIFOm_EIRQ (m=4*k+2). See bit 0.
Bit 31:28 **Reserved:** Read as zero, should be written as zero.
Note: Read as zero, should be written as zero

21 Output Compare Unit (CMP)

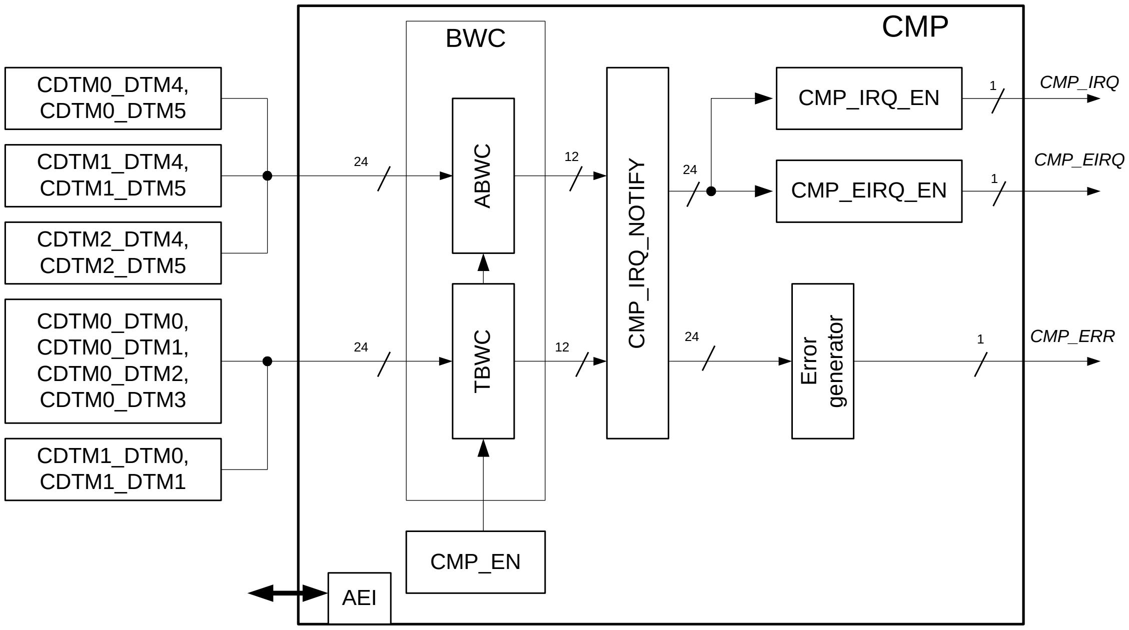
21.1 Overview

The Output Compare Unit (CMP) is designed for the use in safety relevant applications. The main idea is to have the possibility to duplicate outputs in order to be compared in this unit. Because of the simple EXOR function used it is necessary to ensure the total cycle accurate output behavior of the output modules to be compared. This is given when two neighbored DTM channel (CDTM[n]_DTM[2*i] and CDTM[n]_DTM[2*i+1]) generate identical signals with phase shift zero at their outputs. This can be reached if they start their output generation at the same time. This start of synchronization is possible by means of the trigger mechanisms *TRIG_x* provided by the TOM or ATOM as shown in the TOM chapter 12 or ATOM chapter 13. It is not necessary to compare each output channel with each other.

The CMP enables the comparison of 2x24 channels of the CDTM0, CDTM1 and CDTM2 and is restricted to neighbored channels. The first 24 CMP channels are the first 24 DTM channels placed behind TOM0 and TOM1 and the second 24 CMP channels are the first 24 DTM channels placed behind the ATOM0, ATOM1 and ATOM2.

Note : When the channels were generated with a higher frequency than the frequency of cluster 1 it is not certain to catch the interrupt in the notify register. Avoid a comparison if freqency is unequal cluster 1 AND (cluster 0 OR cluster 2)

21.1.1 Architecture of the Compare Unit



21.2 Bitwise Compare Unit (BWC)

21.2.1 ABWC compare unit (1)

ABWC Comparator	Comparator Input 1	Comparator Input 2
ABWC0	CDTM0_DTM4_CH0_OUT	CDTM0_DTM4_CH1_OUT
ABWC1	CDTM0_DTM4_CH2_OUT	CDTM0_DTM4_CH3_OUT
ABWC2	CDTM0_DTM5_CH0_OUT	CDTM0_DTM5_CH1_OUT
ABWC3	CDTM0_DTM5_CH2_OUT	CDTM0_DTM5_CH3_OUT
ABWC4	CDTM1_DTM4_CH0_OUT	CDTM1_DTM4_CH1_OUT
ABWC5	CDTM1_DTM4_CH2_OUT	CDTM1_DTM4_CH3_OUT
ABWC6	CDTM1_DTM5_CH0_OUT	CDTM1_DTM5_CH1_OUT
ABWC7	CDTM1_DTM5_CH2_OUT	CDTM1_DTM5_CH3_OUT
ABWC8	CDTM2_DTM4_CH0_OUT	CDTM2_DTM4_CH1_OUT
ABWC9	CDTM2_DTM4_CH2_OUT	CDTM2_DTM4_CH3_OUT
ABWC10	CDTM2_DTM5_CH0_OUT	CDTM2_DTM5_CH1_OUT
ABWC11	CDTM2_DTM5_CH2_OUT	CDTM2_DTM5_CH3_OUT

The Bitwise Compare Unit TBWC compares in pairs the combinations shown in following table

21.2.2 TBWC compare unit

TBWC Comparator	Comparator Input 1	Comparator Input 2
TBWC0	CDTM0_DTM0_CH0_OUT	CDTM0_DTM0_CH1_OUT
TBWC1	CDTM0_DTM0_CH2_OUT	CDTM0_DTM0_CH3_OUT
TBWC2	CDTM0_DTM1_CH0_OUT	CDTM0_DTM1_CH1_OUT
TBWC3	CDTM0_DTM1_CH2_OUT	CDTM0_DTM1_CH3_OUT
TBWC4	CDTM0_DTM2_CH0_OUT	CDTM0_DTM2_CH1_OUT
TBWC5	CDTM0_DTM2_CH2_OUT	CDTM0_DTM2_CH3_OUT
TBWC6	CDTM0_DTM3_CH0_OUT	CDTM0_DTM3_CH1_OUT
TBWC7	CDTM0_DTM3_CH2_OUT	CDTM0_DTM3_CH3_OUT
TBWC8	CDTM1_DTM0_CH0_OUT	CDTM1_DTM0_CH1_OUT
TBWC9	CDTM1_DTM0_CH2_OUT	CDTM1_DTM0_CH3_OUT
TBWC10	CDTM1_DTM1_CH0_OUT	CDTM1_DTM1_CH1_OUT
TBWC11	CDTM1_DTM1_CH2_OUT	CDTM1_DTM1_CH3_OUT

21.3 Configuration of the Compare Unit

Because of the restrictions described in the section above the Compare Unit consists of 24 antivalence (EXOR) elements, a select register **CMP_EN** which selects the corresponding comparisons and a status register **CMP_IRQ_NOTIFY** which shows and stores each mismatching result, when selected.

For each with **CMP_IRQ_EN** enabled mismatching error an interrupt signal on *CMP_IRQ* is generated.
For each with **CMP_EIRQ_EN** enabled mismatching error an interrupt signal on *CMP_EIRQ* is generated.

21.4 Error Generator

The error generator generates an error signal to be transmitted directly to the MON unit and independently from the *CMP_IRQ* and *CMP_EIRQ*. The error is set when in the **CMP_IRQ_NOTIFY** register at least one bit is set. The CMP_IRQ_NOTIFY bits are not mask able for this purpose.

Additionally *CMP_ERR* is a primary output port for interrupt actions by CPU itself.

21.5 CMP Interrupt Signal

21.5.1 CMP Interrupt Signal table

Signal	Description
CMP_EIRQ	Mismatching interrupt of outputs to be compared, when enabled
CMP_IRQ	Mismatching interrupt of outputs to be compared, when enabled

The CMP sub-module has two interrupt signals, one normal interrupt and one error interrupt. The source of both interrupt can be determined by reading the **CMP_IRQ_NOTIFY** register under consideration of **CMP_IRQ_EN** register and **CMP_EIRQ_EN** register. Each source can be forced separately for debug purposes using the interrupt force **CMP_IRQ_FORCINT** register. **CMP_IRQ_MODE** configures interrupt output characteristic. All interrupt modes are described in detail in section 2.5.

21.6 CMP Configuration Register Overview

21.6.1 CMP Configuration Register Overview Table

Register Name	Description	Details in Section
CMP_EN	CMP comparator enable register	21.7.1
CMP_IRQ_NOTIFY	CMP event notification register	21.7.2
CMP_IRQ_EN	CMP interrupt enable register	21.7.3
CMP_IRQ_FORCINT	CMP interrupt force register	21.7.4
CMP_IRQ_MODE	CMP interrupt mode configuration register	21.7.5
CMP_EIRQ_EN	CMP error interrupt enable register	21.7.6

21.7 CMP Configuration Register Description

21.7.1 Register CMP_EN

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_0000**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								TBWC11_EN	TBWC10_EN	TBWC9_EN	TBWC8_EN	TBWC7_EN	TBWC6_EN	TBWC5_EN	TBWC4_EN	TBWC3_EN	TBWC2_EN	TBWC1_EN	TBWC0_EN	ABWC11_EN	ABWC10_EN	ABWC9_EN	ABWC8_EN	ABWC7_EN	ABWC6_EN	ABWC5_EN	ABWC4_EN	ABWC3_EN	ABWC2_EN	ABWC1_EN	ABWC0_EN
Mode	R								RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Initial Value	0x00								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit 0 **ABWC0_EN:** Enable comparator 0 in ABWC (see chapter 21.2)
0 = ABWC Comparator 0 is disabled
1 = ABWC Comparator 0 is enabled

Bit 1 **ABWC1_EN:** Enable comparator 1 in ABWC (see chapter 21.2)

Bit 2 **ABWC2_EN:** Enable comparator 2 in ABWC (see chapter 21.2)

Bit 3 **ABWC3_EN:** Enable comparator 3 in ABWC (see chapter 21.2)

Bit 4 **ABWC4_EN:** Enable comparator 4 in ABWC (see chapter 21.2)

Bit 5 **ABWC5_EN:** Enable comparator 5 in ABWC (see chapter 21.2)

Bit 6 **ABWC6_EN:** Enable comparator 6 in ABWC (see chapter 21.2)

Bit 7 **ABWC7_EN:** Enable comparator 7 in ABWC (see chapter 21.2)

Bit 8 **ABWC8_EN:** Enable comparator 8 in ABWC (see chapter 21.2)

Bit 9 **ABWC9_EN:** Enable comparator 9 in ABWC (see chapter 21.2)

Bit 10 **ABWC10_EN:** Enable comparator 10 in ABWC (see chapter 21.2)

Bit 11 **ABWC11_EN:** Enable comparator 11 in ABWC (see chapter 21.2)

Bit 12 **TBWC0_EN:** Enable comparator 0 in TBWC (see chapter 21.2)
0 = TBWC comparator 0 is disabled
1 = TBWC comparator 0 is enabled

Bit 13 **TBWC1_EN:** Enable comparator 1 in TBWC (see chapter 21.2)

Bit 14 **TBWC2_EN:** Enable comparator 2 in TBWC (see chapter 21.2)

Bit 15 **TBWC3_EN:** Enable comparator 3 in TBWC (see chapter 21.2)

Bit 16 **TBWC4_EN:** Enable comparator 4 in TBWC (see chapter 21.2)

Bit 17 **TBWC5_EN:** Enable comparator 5 in TBWC (see chapter 21.2)

Bit 18 **TBWC6_EN:** Enable comparator 6 in TBWC (see chapter 21.2)

Bit 19 **TBWC7_EN:** Enable comparator 7 in TBWC (see chapter 21.2)

Bit 20 **TBWC8_EN:** Enable comparator 8 in TBWC (see chapter 21.2)

Bit 21 **TBWC9_EN:** Enable comparator 9 in TBWC (see chapter 21.2)

Bit 22 **TBWC10_EN:** Enable comparator 10 in TBWC (see chapter 21.2)

Bit 23 **TBWC11_EN:** Enable comparator 11 in TBWC (see chapter 21.2)

Bit 31:24 **Reserved:** Reserved
Note: Read as zero, should be written as zero

21.7.2 Register CMP_IRQ_NOTIFY

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_0000**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								TBWC11	TBWC10	TBWC9	TBWC8	TBWC7	TBWC6	TBWC5	TBWC4	TBWC3	TBWC2	TBWC1	TBWC0	ABWC11	ABWC10	ABWC9	ABWC8	ABWC7	ABWC6	ABWC5	ABWC4	ABWC3	ABWC2	ABWC1	ABWC0
Mode	R								RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw
Initial Value	0x00								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit 0 **ABWC0:** error indication for ABWC0
0 = no error recognized on DTMA sub-modules bits 0 and 1 (see chapter 21.2)
1 = an error was recognized on corresponding DTMA sub-modules bits

Note: This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged.

Bit 1 **ABWC1:** error indication for ABWC1. See bit 0.
Bit 2 **ABWC2:** error indication for ABWC2. See bit 0.
Bit 3 **ABWC3:** error indication for ABWC3. See bit 0.
Bit 4 **ABWC4:** error indication for ABWC4. See bit 0.
Bit 5 **ABWC5:** error indication for ABWC5. See bit 0.
Bit 6 **ABWC6:** error indication for ABWC6. See bit 0.
Bit 7 **ABWC7:** error indication for ABWC7. See bit 0.
Bit 8 **ABWC8:** error indication for ABWC8. See bit 0.
Bit 9 **ABWC9:** error indication for ABWC9. See bit 0.
Bit 10 **ABWC10:** error indication for ABWC10. See bit 0.
Bit 11 **ABWC11:** error indication for ABWC11. See bit 0.
Bit 12 **TBWC0:** TOM sub-modules outputs bitwise comparator 0 error indication
0 = no error recognized on TOM sub-modules bits 0 and 1 (see chapter 21.2)
1 = an error was recognized on corresponding TOM sub-modules bits

0 = no error recognized on DTMT sub-modules bits 0 and 1 (see chapter 21.2)
1 = an error was recognized on corresponding DTMT sub-modules bits

Note: This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged.

Bit 13 **TBWC1:** TOM sub-modules outputs bitwise comparator 1 error indication. See bit 12.

Bit 14 **TBWC2:** TOM sub-modules outputs bitwise comparator 2 error indication. See bit 12.

Bit 15 **TBWC3:** TOM sub-modules outputs bitwise comparator 3 error indication. See bit 12.

Bit 16 **TBWC4:** TOM sub-modules outputs bitwise comparator 4 error indication. See bit 12.

Bit 17 **TBWC5:** TOM sub-modules outputs bitwise comparator 5 error indication. See bit 12.

Bit 18 **TBWC6:** TOM sub-modules outputs bitwise comparator 6 error indication. See bit 12.

Bit 19 **TBWC7:** TOM sub-modules outputs bitwise comparator 7 error indication. See bit 12.

Bit 20 **TBWC8:** TOM sub-modules outputs bitwise comparator 8 error indication. See bit 12.

Bit 21 **TBWC9:** TOM sub-modules outputs bitwise comparator 9 error indication. See bit 12.

Bit 22 **TBWC10:** TOM sub-modules outputs bitwise comparator 10 error indication. See bit 12.

Bit 23 **TBWC11:** TOM sub-modules outputs bitwise comparator 11 error indication. See bit 12.

Bit 31:24 **Reserved:** reserved

Note: Read as zero, should be written as zero

21.7.3 Register CMP_IRQ_EN

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_0000**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								TBWC11_EN_IRQ	TBWC10_EN_IRQ	TBWC9_EN_IRQ	TBWC8_EN_IRQ	TBWC7_EN_IRQ	TBWC6_EN_IRQ	TBWC5_EN_IRQ	TBWC4_EN_IRQ	TBWC3_EN_IRQ	TBWC2_EN_IRQ	TBWC1_EN_IRQ	TBWC0_EN_IRQ	ABWC11_EN_IRQ	ABWC10_EN_IRQ	ABWC9_EN_IRQ	ABWC8_EN_IRQ	ABWC7_EN_IRQ	ABWC6_EN_IRQ	ABWC5_EN_IRQ	ABWC4_EN_IRQ	ABWC3_EN_IRQ	ABWC2_EN_IRQ	ABWC1_EN_IRQ	ABWC0_EN_IRQ
Mode	R								RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Initial Value	0x00								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit 0 **ABWC0_EN_IRQ:** enable ABWC0 interrupt source for *CMP_IRQ* line

0 = interrupt source ABWC0 is disabled

1 = interrupt source ABWC0 is enabled

Bit 1 **ABWC1_EN_IRQ:** enable ABWC1 interrupt source for *CMP_IRQ* line. See bit 0.

Bit 2 **ABWC2_EN_IRQ:** enable ABWC2 interrupt source for *CMP_IRQ* line. See bit 0.

Bit 3 **ABWC3_EN_IRQ:** enable ABWC3 interrupt source for *CMP_IRQ* line. See bit 0.

Bit 4 **ABWC4_EN_IRQ:** enable ABWC4 interrupt source for *CMP_IRQ* line. See bit 0.

Bit 5 **ABWC5_EN_IRQ:** enable ABWC5 interrupt source for *CMP_IRQ* line. See bit 0.

Bit 6 **ABWC6_EN_IRQ:** enable ABWC6 interrupt source for *CMP_IRQ* line. See bit 0.

Bit 7 **ABWC7_EN_IRQ:** enable ABWC7 interrupt source for *CMP_IRQ* line. See bit 0.

Bit 8 **ABWC8_EN_IRQ:** enable ABWC8 interrupt source for *CMP_IRQ* line. See bit 0.

Bit 9 **ABWC9_EN_IRQ:** enable ABWC9 interrupt source for *CMP_IRQ* line. See bit 0.

Bit 10 **ABWC10_EN_IRQ:** enable ABWC10 interrupt source for *CMP_IRQ* line. See bit 0.

Bit 11 **ABWC11_EN_IRQ:** enable ABWC11 interrupt source for *CMP_IRQ* line. See bit 0.

Bit 12 **TBWC0_EN_IRQ:** enable TBWC0 interrupt source for *CMP_IRQ* line
0 = interrupt source TBWC0 is disabled
1 = interrupt source TBWC0 is enabled

Bit 13 **TBWC1_EN_IRQ:** enable TBWC1 interrupt source for *CMP_IRQ* line. See bit 12.

Bit 14 **TBWC2_EN_IRQ:** enable TBWC2 interrupt source for *CMP_IRQ* line. See bit 12.

Bit 15 **TBWC3_EN_IRQ:** enable TBWC3 interrupt source for *CMP_IRQ* line. See bit 12.

Bit 16 **TBWC4_EN_IRQ:** enable TBWC4 interrupt source for *CMP_IRQ* line. See bit 12.

Bit 17 **TBWC5_EN_IRQ:** enable TBWC5 interrupt source for *CMP_IRQ* line. See bit 12.

Bit 18 **TBWC6_EN_IRQ:** enable TBWC6 interrupt source for *CMP_IRQ* line. See bit 12.

Bit 19 **TBWC7_EN_IRQ:** enable TBWC7 interrupt source for *CMP_IRQ* line. See bit 12.

Bit 20 **TBWC8_EN_IRQ:** enable TBWC8 interrupt source for *CMP_IRQ* line. See bit 12.

Bit 21 **TBWC9_EN_IRQ:** enable TBWC9 interrupt source for *CMP_IRQ* line. See bit 12.

Bit 22 **TBWC10_EN_IRQ:** enable TBWC10 interrupt source for *CMP_IRQ* line. See bit 12.

Bit 23 **TBWC11_EN_IRQ:** enable TBWC11 interrupt source for *CMP_IRQ* line. See bit 12.

Bit 31:24 **Reserved:** reserved

Note: Read as zero, should be written as zero

21.7.4 Register CMP_IRQ_FORCINT

Address Offset:	**see Appendix B**																**Initial Value: 0x0000_0000**															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								TRG_TBWC11	TRG_TBWC10	TRG_TBWC9	TRG_TBWC8	TRG_TBWC7	TRG_TBWC6	TRG_TBWC5	TRG_TBWC4	TRG_TBWC3	TRG_TBWC2	TRG_TBWC1	TRG_TBWC0	TRG_ABWC11	TRG_ABWC10	TRG_ABWC9	TRG_ABWC8	TRG_ABWC7	TRG_ABWC6	TRG_ABWC5	TRG_ABWC4	TRG_ABWC3	TRG_ABWC2	TRG_ABWC1	TRG_ABWC0
Mode	R								RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw
Initial Value	0x00								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit 0 **TRG_ABWC0:** Trigger **ABWC0** bit in **CMP_IRQ_NOTIFY** register by software
0 = No event triggering
1 = Assert corresponding field in **CMP_IRQ_NOTIFY** register
Note: This bit is cleared automatically after write.
Note: This bit is write protected by bit RF_PROT of register GTM_CTRL

Bit 1 **TRG_ABWC1:** Trigger **ABWC1** bit in **CMP_IRQ_NOTIFY** register by software. See bit 0.

Bit 2 **TRG_ABWC2:** Trigger **ABWC2** bit in **CMP_IRQ_NOTIFY** register by software. See bit 0.

Bit 3 **TRG_ABWC3:** Trigger **ABWC3** bit in **CMP_IRQ_NOTIFY** register by software. See bit 0.

Bit 4 **TRG_ABWC4:** Trigger **ABWC4** bit in **CMP_IRQ_NOTIFY** register by software. See bit 0.

Bit 5 **TRG_ABWC5:** Trigger **ABWC5** bit in **CMP_IRQ_NOTIFY** register by software. See bit 0.

Bit 6 **TRG_ABWC6:** Trigger **ABWC6** bit in **CMP_IRQ_NOTIFY** register by software. See bit 0.

Bit 7 **TRG_ABWC7:** Trigger **ABWC7** bit in **CMP_IRQ_NOTIFY** register by software. See bit 0.

Bit 8 **TRG_ABWC8:** Trigger **ABWC8** bit in **CMP_IRQ_NOTIFY** register by software. See bit 0.

Bit 9 **TRG_ABWC9:** Trigger **ABWC9** bit in **CMP_IRQ_NOTIFY** register by software. See bit 0.

Bit 10 **TRG_ABWC10:** Trigger **ABWC10** bit in **CMP_IRQ_NOTIFY** register by software. See bit 0.

Bit 11 **TRG_ABWC11:** Trigger **ABWC11** bit in **CMP_IRQ_NOTIFY** register by software. See bit 0.

Bit 12 **TRG_TBWC0:** Trigger **TBWC0** bit in **CMP_IRQ_NOTIFY** register by software
0 = No event triggering
1 = Assert corresponding field in **CMP_IRQ_NOTIFY** register
Note: This bit is cleared automatically after write.

Bit 13 **TRG_TBWC1:** Trigger **TBWC1** bit in **CMP_IRQ_NOTIFY** register by software. See bit 12.

Bit 14 **TRG_TBWC2:** Trigger **TBWC2** bit in **CMP_IRQ_NOTIFY** register by software. See bit 12.

Bit 15 **TRG_TBWC3:** Trigger **TBWC3** bit in **CMP_IRQ_NOTIFY** register by software. See bit 12.

Bit 16 **TRG_TBWC4:** Trigger **TBWC4** bit in **CMP_IRQ_NOTIFY** register by software. See bit 12.

Bit 17 **TRG_TBWC5:** Trigger **TBWC5** bit in **CMP_IRQ_NOTIFY** register by software. See bit 12.

Bit 18 **TRG_TBWC6:** Trigger **TBWC6** bit in **CMP_IRQ_NOTIFY** register by software. See bit 12.

Bit 19 **TRG_TBWC7:** Trigger **TBWC7** bit in **CMP_IRQ_NOTIFY** register by software. See bit 12.

Bit 20 **TRG_TBWC8:** Trigger **TBWC8** bit in **CMP_IRQ_NOTIFY** register by software. See bit 12.

Bit 21 **TRG_TBWC9:** Trigger **TBWC9** bit in **CMP_IRQ_NOTIFY** register by software. See bit 12.

Bit 22 **TRG_TBWC10:** Trigger **TBWC10** bit in **CMP_IRQ_NOTIFY** register by software. See bit 12.

Bit 23 **TRG_TBWC11:** Trigger **TBWC11** bit in **CMP_IRQ_NOTIFY** register by software. See bit 12.

Bit 31:24 **Reserved:** reserved
Note: Read as zero, should be written as zero

21.7.5 Register CMP_IRQ_MODE

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_000X**							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																														IRQ_MODE	
Mode	R																														RW	
Initial Value	0x0000 0000																														XX	

Bit 1:0 **IRQ_MODE**: IRQ mode selection

0b00 = Level mode
0b01 = Pulse mode
0b10 = Pulse-Notify mode
0b11 = Single-Pulse mode
Note: The interrupt modes are described in section 2.5.

Bit 31:2 **Reserved:** reserved
Note: Read as zero, should be written as zero

21.7.6 Register CMP_EIRQ_EN

Address Offset:	**see Appendix B**																**Initial Value:**	**0x0000_0000**														
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								TBWC11_EN_EIR	TBWC10_EN_EIR	TBWC9_EN_EIRQ	TBWC8_EN_EIRQ	TBWC7_EN_EIRQ	TBWC6_EN_EIRQ	TBWC5_EN_EIRQ	TBWC4_EN_EIRQ	TBWC3_EN_EIRQ	TBWC2_EN_EIRQ	TBWC1_EN_EIRQ	TBWC0_EN_EIRQ	ABWC11_EN_EIR	ABWC10_EN_EIR	ABWC9_EN_EIRQ	ABWC8_EN_EIRQ	ABWC7_EN_EIRQ	ABWC6_EN_EIRQ	ABWC5_EN_EIRQ	ABWC4_EN_EIRQ	ABWC3_EN_EIRQ	ABWC2_EN_EIRQ	ABWC1_EN_EIRQ	ABWC0_EN_EIRQ
Mode	R								RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Initial Value	0x00								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit 0 **ABWC0_EN_EIRQ:** enable ABWC0 interrupt source for *CMP_EIRQ* line
0 = interrupt source ABWC0 is disabled
1 = interrupt source ABWC0 is enabled

Bit 1 **ABWC1_EN_EIRQ:** enable ABWC1 interrupt source for *CMP_EIRQ* line. See bit 0.

Bit 2 **ABWC2_EN_EIRQ:** enable ABWC2 interrupt source for *CMP_EIRQ* line. See bit 0.

Bit 3 **ABWC3_EN_EIRQ:** enable ABWC3 interrupt source for *CMP_EIRQ* line. See bit 0.

Bit 4 **ABWC4_EN_EIRQ:** enable ABWC4 interrupt source for *CMP_EIRQ* line. See bit 0.

Bit 5 **ABWC5_EN_EIRQ:** enable ABWC5 interrupt source for *CMP_EIRQ* line. See bit 0.

Bit 6 **ABWC6_EN_EIRQ:** enable ABWC6 interrupt source for *CMP_EIRQ* line. See bit 0.

Bit 7 **ABWC7_EN_EIRQ:** enable ABWC7 interrupt source for *CMP_EIRQ* line. See bit 0.

Bit 8 **ABWC8_EN_EIRQ:** enable ABWC8 interrupt source for *CMP_EIRQ* line. See bit 0.

Bit 9 **ABWC9_EN_EIRQ:** enable ABWC9 interrupt source for *CMP_EIRQ* line. See bit 0.

Bit 10 **ABWC10_EN_EIRQ:** enable ABWC10 interrupt source for *CMP_EIRQ* line. See bit 0.

Bit 11 **ABWC11_EN_EIRQ:** enable ABWC11 interrupt source for *CMP_EIRQ* line. See bit 0.

Bit 12 **TBWC0_EN_EIRQ:** enable TBWC0 interrupt source for *CMP_EIRQ* line
0 = interrupt source TBWC0 is disabled
1 = interrupt source TBWC0 is enabled

Bit 13 **TBWC1_EN_EIRQ:** enable TBWC1 interrupt source for *CMP_EIRQ* line. See bit 12.

Bit 14 **TBWC2_EN_EIRQ:** enable TBWC2 interrupt source for *CMP_EIRQ* line. See bit 12.

Bit 15 **TBWC3_EN_EIRQ:** enable TBWC3 interrupt source for *CMP_EIRQ* line. See bit 12.

Bit 16 **TBWC4_EN_EIRQ:** enable TBWC4 interrupt source for *CMP_EIRQ* line. See bit 12.

Bit 17 **TBWC5_EN_EIRQ:** enable TBWC5 interrupt source for *CMP_EIRQ* line. See bit 12.

Bit 18 **TBWC6_EN_EIRQ:** enable TBWC6 interrupt source for *CMP_EIRQ* line. See bit 12.

Bit 19 **TBWC7_EN_EIRQ:** enable TBWC7 interrupt source for *CMP_EIRQ* line. See bit 12.

Bit 20 **TBWC8_EN_EIRQ:** enable TBWC8 interrupt source for *CMP_EIRQ* line. See bit 12.

Bit 21 **TBWC9_EN_EIRQ:** enable TBWC9 interrupt source for *CMP_EIRQ* line. See bit 12.

Bit 22 **TBWC10_EN_EIRQ:** enable TBWC10 interrupt source for *CMP_EIRQ* line. See bit 12.

Bit 23 **TBWC11_EN_EIRQ:** enable TBWC11 interrupt source for *CMP_EIRQ* line. See bit 12.

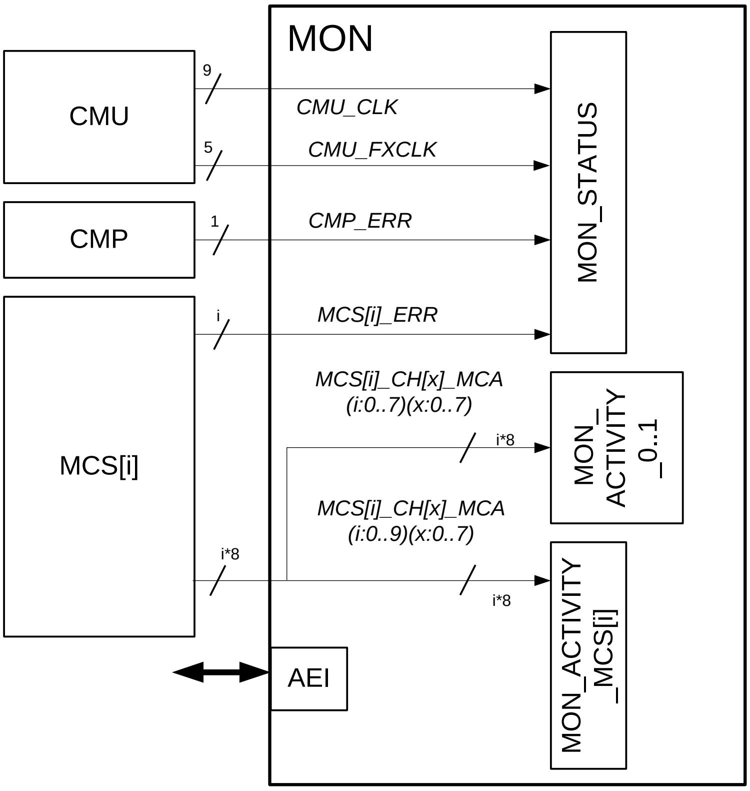
Bit 31:24 **Reserved:** reserved
Note: Read as zero, should be written as zero

22 Monitor Unit (MON)

22.1 Overview

The Monitor Unit (MON) is designed for the use in safety relevant applications. The main idea is to have a possibility to supervise common used circuitry and resources. In this way the activity of the clocks is supervised. In addition the characteristics of output signals can be checked in a MCS channel by a re-read-in via TIM and routing to the MCS. When the comparison fails an error signal is generated in MCS and sent to the monitor unit. One error signal per MCS summarizes the errors of all channels. By generating of an activity signal per channel for each such performed comparison, the activity of TIM, ARU and the used clocks is checked implicitly.
In addition the ARU cycle time could be also compared in a MCS channel to given values.

22.1.1 MON Block Diagram



22.1.2 Realization without Activity Checker of the clock signals

An activity checker of the clock signals used is not needed because these signals are only enables to be used in combination with the system clock. Therefore the clock enables are to be checked to have a high value.

22.2 Clock Monitoring

The monitor unit has a connection to each of the 9 clocks *CMU_CLK[x]* (x=0..8), provided by the CMU. Some of these clocks can be used for special tasks (see chapter 8).

In addition the 5 clock inputs of the TOMs *CMU_FXCLK[y]* (y=0..4) are also connected to the MON unit.
The supervising of the clocks is done by scanning for activity of each clock.
A high value is defined as the state to be monitored.
When a high value of the clock enable is detected, the corresponding bit in the status register **MON_STATUS** is set.

The status register bits are reset by writing a one.
When the register is polled by the CPU and the time between two read accesses is higher than the period of the slowest clock, all bits of the corresponding clocks must have been set.

When polling in shorter time distances, not for all clocks an activity can be shown, although they are still working.

Because of the realization without a select register for the clock signals only the bits of the status register are to be considered for which the clock signal is enabled in the CMU.

22.3 CMP error Monitoring

The signal CMP_ERR is to be received directly from module CMP and is set if an error occurred.

22.4 Checking the Characteristics of Signals by MCS

By use of the MCS some given properties of signals can be checked. Such signals can be generated output signals of TOM or ATOM channels including DTM function, which are reread in into a TIM and the time stamp information is routed via ARU to the MCS module.

The corresponding MCS signal performs the check according to given properties. In this way signal high or low time as well as signal periods can be checked, also taking into account tolerances. When the check fails a MCS internal error signal is generated and ORed with the error signals of the other channels of the MCS module to a summarized error signal *MCS[i]_ERR*.

For each MCS a summarized error signal is transmitted to MON and monitored in the MON_STATUS register.
In order to check the execution of the comparison for each MCS channel an activity signal is generated. In the MCS[i]_CH[x]_MCA (i=0..9) (x=0..7) vector 8 bits for each MCS[i](i=0..9) instance are combined. The activity signals are stored in the **MON_ACTIVITY_MCS[i]** register. In addition the first 8 bits of MCS0..3 are stored in **MON_ACTIVITY_0** and the first 8 bits of MCS4..7 are stored in **MON_ACTIVITY_1**. The bits are set by a one signal and reset by writing a one to it (preferably after polling the status of the register).

Because the activity signal shows the execution of a comparison, the involved units for providing the signals and execution of comparison (like TIM, ARU and MCS itself) are checked implicitly to work accordingly. Also the involved clocks and time bases are checked in this way.

22.5 Checking ARU Cycle Time

The cycle time of the ARU can be checked, when this is essential for safety purposes. This check can be performed by an MCS channel. It should be noted that the MCS program for measuring the ARU round trip time must add a tolerance value.

The resulting error is reported to the MON unit using the summarized error signal *MCS[i]_ERR* for each MCS module in addition to an interrupt, generated in MCS. The same signals and status bits are used as in the case of checking the signal characteristics.

The corresponding MCS is programmed to get a fixed data value at address 0x1FF. The data value is always zero and is not blocked. When getting the access the time stamp value TBU_TS0 is stored in a register. The next time getting the access the new TBU_TS0 value is stored and the difference between both values is compared with a given value. When the comparison fails, an error flag is set in the MCS internal status register, an interrupt is generated and the error signal *MCS[i]_ERR* is provided.

When the check is performed, an activity signal MCS[i]_CH[x]_MCA (i=0..9)(x=0..7) is provided for each channel x for each MCS[i](i=0..9) instance together with a summarized interrupt MCS[i]_ERR for each MCS.

The activity signal sets a bit in the MON_ACTIVITY register.
The bits in the MON_ACTIVITY registers are reset by writing a one.
When the check fails, an interrupt is generated and the error signal MCS[i]_ERR is provided for the MON unit.
Figure 22.1.1 shows the block diagram of the Monitor Unit.

22.6 MON Interrupt Signals

The MON submodule has no interrupt signals.

22.7 MON Register Overview

Register Name	Description	Details in Section
MON_STATUS	MON status register	22.8.1
MON_ACTIVITY_0	MON activity register 0	22.8.2
MON_ACTIVITY_1	MON activity register 1	22.8.3
MON_ACTIVITY_MCS[z] (z:0...9)	MON activity register for MCS z	22.8.4

22.8 MON Configuration Register Description

22.8.1 Register MON_STATUS

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved		MCS9_ERR	MCS8_ERR	MCS7_ERR	MCS6_ERR	MCS5_ERR	MCS4_ERR	MCS3_ERR	MCS2_ERR	MCS1_ERR	MCS0_ERR	Reserved			CMP_ERR	Reserved	ACT_CMU8	Reserved	ACT_CMUFX4	ACT_CMUFX3	ACT_CMUFX2	ACT_CMUFX1	ACT_CMUFX0	ACT_CMU7	ACT_CMU6	ACT_CMU5	ACT_CMU4	ACT_CMU3	ACT_CMU2	ACT_CMU1	ACT_CMU0
Mode	R		R	R	R	R	R	R	R	R	R	R	R			R	R	RCw	R	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw
Initial Value	0x00		0	0	0	0	0	0	0	0	0	0	0x00			0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit 0 **ACT_CMU0**: CMU_CLK0 activity
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 1 **ACT_CMU1**: CMU_CLK1 activity
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 2 **ACT_CMU2**: CMU_CLK2 activity
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 3 **ACT_CMU3**: CMU_CLK3 activity
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 4 **ACT_CMU4**: CMU_CLK4 activity
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 5 **ACT_CMU5**: CMU_CLK5 activity
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 6 **ACT_CMU6**: CMU_CLK6 activity
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 7 **ACT_CMU7**: CMU_CLK7 activity
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 8 **ACT_CMUFX0**: CMU_CLKFX0 activity
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 9 **ACT_CMUFX1**: CMU_CLKFX1 activity
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 10 **ACT_CMUFX2**: CMU_CLKFX2 activity
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 11 **ACT_CMUFX3**: CMU_CLKFX3 activity
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 12 **ACT_CMUFX4**: CMU_CLKFX4 activity
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.
Note: Bits 0 to 12 are set, when a rising edge is detected at the considered clock

Bit 13 **Reserved:** Reserved bits
Note: Read as zero should be written as zero

Bit 14 **ACT_CMU8**: CMU_CLK8 activity
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.
Note: Bit is set, when a rising edge is detected at the considered clock

Bit 15 **Reserved:** Reserved bits
Note: Read as zero should be written as zero

Bit 16 **CMP_ERR:** Error detected at CMP

Note: This bit will be readable only.

Bit 19:17 **Reserved:** Reserved bits
Note: Read as zero should be written as zero

Bit 20 **MCS0_ERR:** Error detected at MCS0
Note: This bit will be readable only.

Bit 21 **MCS1_ERR:** Error detected at MCS1
Note: This bit will be readable only.

Bit 22 **MCS2_ERR:** Error detected at MCS2
Note: This bit will be readable only.

Bit 23 **MCS3_ERR:** Error detected at MCS3
Note: This bit will be readable only.

Bit 24 **MCS4_ERR:** Error detected at MCS4

Note: This bit will be readable only.

Bit 25 **MCS5_ERR:** Error detected at MCS5
Note: This bit will be readable only.

Bit 26 **MCS6_ERR:** Error detected at MCS6
Note: This bit will be readable only.

Bit 27 **MCS7_ERR:** Error detected at MCS7
Note: This bit will be readable only.

Bit 28 **MCS8_ERR:** Error detected at MCS8
Note: This bit will be readable only.

Bit 29 **MCS9_ERR:** Error detected at MCS9
Note: This bit will be readable only.

Bit 31:30 **Reserved:** Reserved bits
Note: Read as zero should be written as zero

Note: Bits16 and 20 to 29 are set, when the corresponding unit reports an error

Note: The MCS can be programmed to generate an error, when the comparison of signal values (duty time, cycle time) fails or also when the cycle time of the ARU (checking of the TBU_TS0 between two periodic accesses) is out of the expected range.

22.8.2 Register MON_ACTIVITY_0

Address Offset:	**see Appendix B**																**Initial Value:** **0x0000_0000**															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	MCA_3_7	MCA_3_6	MCA_3_5	MCA_3_4	MCA_3_3	MCA_3_2	MCA_3_1	MCA_3_0	MCA_2_7	MCA_2_6	MCA_2_5	MCA_2_4	MCA_2_3	MCA_2_2	MCA_2_1	MCA_2_0	MCA_1_7	MCA_1_6	MCA_1_5	MCA_1_4	MCA_1_3	MCA_1_2	MCA_1_1	MCA_1_0	MCA_0_7	MCA_0_6	MCA_0_5	MCA_0_4	MCA_0_3	MCA_0_2	MCA_0_1	MCA_0_0
Mode	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw
Initial Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit 0 **MCA_0_0**: activity of check performed in module MCS 0 at channel 0
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 1 **MCA_0_1**: activity of check performed in module MCS 0 at channel 1
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 2 **MCA_0_2**: activity of check performed in module MCS 0 at channel 2
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 3 **MCA_0_3**: activity of check performed in module MCS 0 at channel 3
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 4 **MCA_0_4**: activity of check performed in module MCS 0 at channel 4
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 5 **MCA_0_5**: activity of check performed in module MCS 0 at channel 5
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 6 **MCA_0_6**: activity of check performed in module MCS 0 at channel 6
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 7 **MCA_0_7**: activity of check performed in module MCS 0 at channel 7
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 8 **MCA_1_0**: activity of check performed in module MCS 1 at channel 0
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 9 **MCA_1_1**: activity of check performed in module MCS 1 at channel 1
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 10 **MCA_1_2**: activity of check performed in module MCS 1 at channel 2
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 11 **MCA_1_3**: activity of check performed in module MCS 1 at channel 3

Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 12 **MCA_1_4**: activity of check performed in module MCS 1 at channel 4
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 13 **MCA_1_5**: activity of check performed in module MCS 1 at channel 5
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 14 **MCA_1_6**: activity of check performed in module MCS 1 at channel 6
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 15 **MCA_1_7**: activity of check performed in module MCS 1 at channel 7
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 16 **MCA_2_0**: activity of check performed in module MCS 2 at channel 0
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 17 **MCA_2_1**: activity of check performed in module MCS 2 at channel 1
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 18 **MCA_2_2**: activity of check performed in module MCS 2 at channel 2
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 19 **MCA_2_3**: activity of check performed in module MCS 2 at channel 3
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 20 **MCA_2_4**: activity of check performed in module MCS 2 at channel 4
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 21 **MCA_2_5**: activity of check performed in module MCS 2 at channel 5
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 22 **MCA_2_6**: activity of check performed in module MCS 2 at channel 6
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 23 **MCA_2_7**: activity of check performed in module MCS 2 at channel 7
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 24 **MCA_3_0**: activity of check performed in module MCS 3 at channel 0
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 25 **MCA_3_1**: activity of check performed in module MCS 3 at channel 1
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 26 **MCA_3_2**: activity of check performed in module MCS 3 at channel 2
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 27 **MCA_3_3**: activity of check performed in module MCS 3 at channel 3

Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 28 **MCA_3_4**: activity of check performed in module MCS 3 at channel 4
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 29 **MCA_3_5**: activity of check performed in module MCS 3 at channel 5
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 30 **MCA_3_6**: activity of check performed in module MCS 3 at channel 6
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 31 **MCA_3_7**: activity of check performed in module MCS 3 at channel 7
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Note: When not all MCS modules are implemented or the channels are not used for check purposes with supervising, the corresponding activity bits remain zero.

22.8.3 Register MON_ACTIVITY_1

Address Offset:	**see Appendix B**																**Initial Value: 0x0000_0000**															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	MCA_7_7	MCA_7_6	MCA_7_5	MCA_7_4	MCA_7_3	MCA_7_2	MCA_7_1	MCA_7_0	MCA_6_7	MCA_6_6	MCA_6_5	MCA_6_4	MCA_6_3	MCA_6_2	MCA_6_1	MCA_6_0	MCA_5_7	MCA_5_6	MCA_5_5	MCA_5_4	MCA_5_3	MCA_5_2	MCA_5_1	MCA_5_0	MCA_4_7	MCA_4_6	MCA_4_5	MCA_4_4	MCA_4_3	MCA_4_2	MCA_4_1	MCA_4_0
Mode	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw
Initial Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit 0 **MCA_4_0**: activity of check performed in module MCS 4 at channel 0
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 1 **MCA_4_1**: activity of check performed in module MCS 4 at channel 1
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 2 **MCA_4_2**: activity of check performed in module MCS 4 at channel 2
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 3 **MCA_4_3**: activity of check performed in module MCS 4 at channel 3
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 4 **MCA_4_4**: activity of check performed in module MCS 4 at channel 4

Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 5 **MCA_4_5**: activity of check performed in module MCS 4 at channel 5
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 6 **MCA_4_6**: activity of check performed in module MCS 4 at channel 6
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 7 **MCA_4_7**: activity of check performed in module MCS 4 at channel 7
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 8 **MCA_5_0**: activity of check performed in module MCS 5 at channel 0
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 9 **MCA_5_1**: activity of check performed in module MCS 5 at channel 1
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 10 **MCA_5_2**: activity of check performed in module MCS 5 at channel 2
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 11 **MCA_5_3**: activity of check performed in module MCS 5 at channel 3
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 12 **MCA_5_4**: activity of check performed in module MCS 5 at channel 4
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 13 **MCA_5_5**: activity of check performed in module MCS 5 at channel 5
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 14 **MCA_5_6**: activity of check performed in module MCS 5 at channel 6
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 15 **MCA_5_7**: activity of check performed in module MCS 5 at channel 7
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 16 **MCA_6_0**: activity of check performed in module MCS 6 at channel 0
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 17 **MCA_6_1**: activity of check performed in module MCS 6 at channel 1
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 18 **MCA_6_2**: activity of check performed in module MCS 6 at channel 2
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 19 **MCA_6_3**: activity of check performed in module MCS 6 at channel 3
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 20 **MCA_6_4**: activity of check performed in module MCS 6 at channel 4

Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 21 **MCA_6_5**: activity of check performed in module MCS 6 at channel 5
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 22 **MCA_6_6**: activity of check performed in module MCS 6 at channel 6
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 23 **MCA_6_7**: activity of check performed in module MCS 6 at channel 7
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 24 **MCA_7_0**: activity of check performed in module MCS 7 at channel 0
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 25 **MCA_7_1**: activity of check performed in module MCS 7 at channel 1
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 26 **MCA_7_2**: activity of check performed in module MCS 7 at channel 2
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 27 **MCA_7_3**: activity of check performed in module MCS 7 at channel 3
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 28 **MCA_7_4**: activity of check performed in module MCS 7 at channel 4
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 29 **MCA_7_5**: activity of check performed in module MCS 7 at channel 5
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 30 **MCA_7_6**: activity of check performed in module MCS 7 at channel 6
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 31 **MCA_7_7**: activity of check performed in module MCS 7 at channel 7
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Note: When not all MCS modules are implemented or the channels are not used for check purposes with supervising, the corresponding activity bits remain zero.

22.8.4 Register MON_ACTIVITY_MCS[z] (z:0...9)

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved																								MCA_7	MCA_6	MCA_5	MCA_4	MCA_3	MCA_2	MCA_1	MCA_0
Mode	R																								RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw
Initial Value	0x0000 00																								0	0	0	0	0	0	0	0

Bit 0 **MCA_0**: activity of check performed in module MCS[i] at channel 0
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 1 **MCA_1**: activity of check performed in module MCS[i] at channel 1
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 2 **MCA_2**: activity of check performed in module MCS[i] at channel 2
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 3 **MCA_3**: activity of check performed in module MCS[i] at channel 3
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 4 **MCA_4**: activity of check performed in module MCS[i] at channel 4
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 5 **MCA_5**: activity of check performed in module MCS[i] at channel 5
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 6 **MCA_6**: activity of check performed in module MCS[i] at channel 6
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Bit 7 **MCA_7**: activity of check performed in module MCS[i] at channel 7
Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.
Note: Unused MCA bits are reserved

Bit 31:8 **Reserved:** Reserved bits
Note: Read as zero should be written as zero

23 Appendix A

23.1 Register Bit Attributes

Mode	Description
R	Read access
W	Write access
Cr	Clear on read access
Sr	Set on read access
Cw	Clear by write 1 (clears only those bits with value 1)
Sw	Set by write 1 (sets only those bits with value 1)
Aw	Auto clear after write (e.g. trigger something)
Pw	Protected write (separate write enable bit, e.g. init)

Below the bit name in a register table, the attributes "Access Mode" and "Reset Value" of each bit are described with the syntax above.

Note: When using Cw or Sw for a bit field e.g. representing a number, a clear / set has to be applied to all bits of the data field, to avoid construction of unintended values different to 0b00..00 and 0b11..11.

23.2 Register Reset Value

Reset Value	Description
0	logic value is 0 after reset
1	logic value is 1 after reset

23.3 ARU Write Address Overview

The ARU write address map is specified in Appendix B [1].

23.4 GTM Configuration Register Address Map

The addresses of the implemented sub-modules are specified in Appendix B [1].
The start and end address of the configured rams are specified in Appendix B [1].
The full address map of all implemented registers, the start and end addresses of configured rams are recorded in Appendix B [1].

23.5 GTM Application Constraints

The constraints put on applications by GTM implementation are specified in Appendix B [1].

23.6 GTM Internal functional dependencies

23.6.1 GTM Internal functional dependencies (part 1)

Signal paths between GTM-IP modules

from: \ to:	map	mcs	mon	psm	spe	tbu	tim	tom
aru								
aru		53 bit data		53 bit data				
atom							ATOM[i]_OUT	
brc								
ccm							CCM[i]_ CMU_CLK[x]	CCM[i]_ CMU_FXCLK[x]
cmp			CMP_ERR					
cmu			CMU_CLK[x], CMU_FXCLK[x]			CMU_CLK[x]	CMU_CLK[x]	CMU_FXCLK[x]
dpll						SUB_INC1c, SUB_INC2c,..		
dtm								
icm								
map	X							
mcs		X	MCS[i]_ERR, MCS[i]_CH[x]_MCA					
mon			X					
psm				X				
spe	SPE[i]_OUT, SPE[i]_NIPD, ...				X			SPE[i]_OUT, SPE[i]_NIPD, ...
tbu		TBU_TS[x]				X	TBU_TS[x]	TBU_TS[x]
tim	TIM0_CH0(48:0)	TIM[i]_EXT_ CAPTURE			TIM[i]_CH[x](48)		X	TIM[i]_EXT_ CAPTURE
tom					TOM[i]_CH0_TRIG_CCU0, TOM[i]_CH0_TRIG_CCU1, TOM[i]_CH[x]_SOUR, ...		TOM[i]_OUT	X

23.6.2 GTM Internal functional dependencies (part 2)

Signal paths between GTM-IP modules

from: \ to:	arch	aru	atom	brc	ccm	cmp	cmu	dpll	dtm	icm
arch										IRQ + EIRQ signals
aru			53 bit data	53 bit data				53 bit data		IRQ signals
atom		53 bit data							ATOM[i]_OUT	IRQ signals
brc		53 bit data								IRQ signals
ccm			CCM[i]_ CMU_CLK[x]				CCM[i]_ CMU_CLK6	CCM[i]_ CMU_CLK[x]	CCM[i]_ CMU_CLK[x]/FX CLK[x]	IRQ signals
cmp										IRQ + EIRQ signals
cmu			CMU_CLK[x]		CMU_CLK[x]			CMU_CLK0		
dpll		53 bit data					SUB_INC1, SUB_INC2...			IRQ + EIRQ signals
dtm						DTM[i]_OUT				
icm										
map								TRIGGER, STATE T_DIR, S_DIR, ...		
mcs		53 bit data								IRQ + EIRQ signals
mon										
psm		53 bit data								IRQ + EIRQ signals
spe										IRQ + EIRQ signals
tbu			TBU_TS[x]					TBU_TS[x], LOW_RES, TS_CLK		
tim		53 bit data	TIM[i]_EXT_ CAPTURE						TIM[i]_CH[x] _F_OUT	IRQ + EIRQ signals
tom									TOM[i]_OUT	IRQ signals

23.7 Compatibility Notes

23.7.1 DPLL

The following features of DPLL have changed since GTM v3.1.0 release:
(1) In case of TORI/SORI the DPLL internal pointer handling is continued and the inc_cnt is not frozen.
(2) The acceptance of input signals by configuration of DPLL_CTRL_1.TSL/SSL is extended such that after enabling the by setting DPLL by DPLL_CTRL_1.DEN=1 while DPLL_STATUS.FTD/FSD still '0' the first input signal is treated as level sensitive. In this case the signal level only is used to decide if an input signal is treated as active input signal or not.

23.7.2 MCS

Since GTM v2.x following feaures of MCS have changed:
(1) MCS[i]_CTRL was replaced by MCS[i]_CTRL_STAT: MCS[i]_CTRL is obsolete. MCS[i]_CTRL and MCS[i]_CTRL_STAT have different addresses.

(2) MCS[i]_RST was replaced by MCS[i]_RESET: MCS[i]_RST is obsolete.
MCS[i]_RST and MCS[i]_RESET have different addresses.

(3) the new register MCS[i]_CAT contains bits of obsolete register MCS[i]_CTRL

(4) the new register MCS[i]_CWT contains bits of obsolete register MCS[i]_CTRL

(5) the instruction execution time in accelerated mode compared to a GTM v1.x may be faster due to instruction pre-fetching.

Since GTM v3.1.x following features of MCS have changed:
(6) the instruction execution time in accelerated mode compared to a GTM v2.x and v1.x may be different due to increased pipeline depth.

(7) On mode switch to SCD_MODE = 0b01 - Round Robin Scheduling - the value of SCD_CH has to be set to appropriate number of tasks. The reset value 0b0000 for SCD_CH means that only task 0 is scheduled.

24 Revision History

24.1 Revision History Table

Issue	Date	Remark
3.1.1.0	24.10.2014	GTM v3.1.1.0 changes to v3.1.0.0 at all: changed multi bit definitions (00 01 100 011 ...) to (0b00 0b01 0b100 0b011 ...) 1st word in column Description of register overview tables is module name GTM: 2.1 Information about appendix B Information about virtual hierarchy PSM Information about virtual hierarchy CDTM Information about new cluster structure 2.1.3 changed figure GTM-IP signal multiplex 2.1.4 changed figure TIM auxiliary input multiplexing 2.1.6 changed figure TIM to MCS signal forwarding 2.3.1 table Overview about the number of channels replaced by table ARU source and destination address count per instance 2.9.2 new bits BRIDGE_MODE_WREN 2.9.5 new bits CLK_EN_ERR, CLK_PER_ERR, CLK_EN_ERR_STATE and CLK_EN_EXP_STATE 2.9.6 new bits CLK_EN_ERR_IRQ_EN and CLK_PER_ERR_IRQ_EN 2.9.7 new bits TRG_CLK_EN_ERR and TRG_CLK_PER_ERR 2.8 new register GTM_CFG 2.9.9 detailed information: MSK_WR_RSP new bit BYPASS_SYNC 2.9.13 new bits CLK_EN_ERR_EIRQ_EN and CLK_PER_ERR_EIRQ_EN 2.9.14 changed definition of SRC_CH0...7 new bits SEL_OUT_N_CH0...7 2.9.15 new bit INT_CLK_EN_GEN 2.9.19 new register GTM_CFG

		ARU: 3.5 ARU prefix added to access acknowledge IRQ bit 3.6 changed index x to z 3.7.17ff changed index x to z BRC: 4.4 changed index x to z 4.5.1ff changed index x to z FIFO: 5.4 changed index x to z 5.5.1ff changed index x to z AFD: 6.2 changed index x to z 6.3.1 changed index x to z F2A: 7.1 additional stream information about enable and disable 7.4 changed index x to z 7.5.1 additional stream information 7.5.2ff changed index x to z 7.5.4 additional stream information CMU: 8.1.1 updated CMU block diagram 8.3 removed sources for CMU_CLK6 and CMU_CLK7 8.8.4ff additional hint that EN_ECLK1 must be disabled, too changed index x to z 8.8.5 removed bit for source selector of CMU_CLK6 CCM: 9.1 information about cluster clock configuration about CCM[i]_(CMU_CLK_CFG, CMU_FXCLK_CFG) 9.1.1 information about mirroring of new registers CCM[i]_(HW_CONF, AUX_IN_SRC, EXT_CAP_EN, TOM_OUT, ATOM_OUT) 9.3 GTM global register added

		9.4.2 bits CLK1_SRC, CLK6_SRC, FXCLK0_SRC, TBU_CH2_SRC removed, bit CLS_CLK_DIV added 9.4.3 bit TBU_TS2 replaced by CLK0_SRC, CLK1_SRC added 9.4.4 new register 9.4.8-12 new registers CCM[i]_(HW_CONF, TIM_AUX_IN_SRC, EXT_CAP_EN, TOM_OUT, ATOM_OUT) TBU: 10.1 new channel for relative angle 10.1.1 updated TBU block diagram 10.4.4ff separated description for CH1 and CH2_CTRL registers 10.4.5 removed bit TS2_SRC TIM: 11.1.1 changed figure **TIM Block Diagram** 11.1.2 new sub chapter internal connectivity contains information from 11.1.1 partly 11.1.3 changed figure **INPUTSRC Block Diagram** added additional functionality 11.4.1.1 changed figure **TIM Channel Architecture** 11.4.2.7 inserted figure **TIM Serial Shift Mode** introduced configurable init value 11.4.2.7.1 new sub chapter **Signal Generation with TIM Serial Shift Mode** 11.4.2.7.2 extended external capture functionality 11.8.1ff detailed definition: GPR0_SEL, GPR1_SEL, CNTS_SEL, DSL, ECNT_RESET, FLT_CTR_RE, FLT_CTR_FE and CLK_SEL 11.8.7 reworded for mode TSSM in CNT 11.8.16 detailed definition: SLICING 11.8.17 additional note 11.8.19 detailed definition: behavior TDU_RESYNC new bit USE_LUT TOM: 12.2.2 naming of SOMP mode removed 12.7 reworded column "Description" 12.8.1ff detailed definition: double bits 12.8.1 reworded bit description UPEN_CTRL 12.8.3 reworded bit description ENDIS_STAT 12.8.6ff detailed information about write/read function

		12.8.7 reworded bit description FUPD_CTRL0 and RSTCN0_CH0 12.8.7 reworded bit description INT_TRIG0 12.8.9 reworded bit description CLK_SRC_SR, SPE_TRIG and SPEM new bit ECLK_SRC 12.8.16 reworded bit description CCU1TC ATOM: 13.1 reworded update via ARU in SOMP mode 13.3 exception of bit reverse mode removed 13.3.1.1 reworded bit description SL, additional "Not used" bits 13.3.2.2 reworded exception sentence 13.3.2.3 reworded exception sentence 13.3.2.4 reworded bit description SL, bit ESLS renamed to bit EUPD, additional "Not used" bits 13.3.3.9 reworded bit description SL and CLK_SRC_SR, new bits ECLK_SRC and EXT_FUPD 13.3.4.10 reworded bit description SL and CLK_SRC_SR, new bits ECLK_SRC and EXT_FUPD 13.3.5.4 reworded bit description SL, bit ESLS renamed to bit EUPD, additional "Not used" bits 13.5 reworded column "Description" 13.6.1 reworded bit description UPEN_CTRL0 13.6.6 reworded bit description OUTEN_STAT0 13.6.7 reworded bit description FUPD_CTRL0 and RSTCN0_CH0 13.6.8 reworded bit description INT_TRIG0 13.6.9 reworded bit description SL and CLK_SRC/CLK_SRC_SR new bits ECLK_SRC and EXT_FUPD, bit ESLS renamed to bit EUPD DTM: 14.1ff changed figure Overview renaming OUT to COUT, TIM_CH_IN1 added to description, cluster DTM (CDTM) introduced, detailed specification of connections between TIM, TOM, ATOM and DTM 14.3.1 additional TIM input port 14.4 additional TIM input port

		14.6 additional TIM input port 14.8ff cluster DTM prefix (CDTM) added to register name 14.9.6 new bit TIM_SEL MCS: 15.1 updated parameter list 15.4 clarified memory description 15.7 updated register set XOREG 15.7.1 2-parts-table **Instruction Set Summary** divided into 3 parts 15.7.2 table **Instruction Code** divided into 2 parts 15.7.15ff clarified description of ARU instruction. 15.7.45ff updated description of new instructions DIVU, DIVS, and MODU 17.7.73 renamed instruction WURL to WURLE 17.7.74 new instructions WUBS 15.8 new registers DSTAX 15.9.3 additional information in note 15.9.14 new register DSTAX 15.11.12 corrected note of register MCS[i]_REG_PROT MCFG: 16.3.1 new bits MEM7 to MEM9 DPLL: 18.5.6,8 modified details to PD/PD_S 18.6.2.2 some corrections for equation DPLL-1b in DPLL-1b1, DPLL-1b2 and DPLL-1b3 18.6.2.7 additional formula DPLL-5a2 to DPLL-5a3 and DPLL-5c 18.6.3.2 some corrections for equation DPLL-6b in DPLL-6b1, DPLL-6b2 and DPLL-6b3 18.7.4 some changes index p to t 18.8.3 modified equation DPLL-21 and DPLL-22 18.8.4 modified equation DPLL-27 and DPLL-28 18.8.6.7 removed TOR=0 condition for SYT=1 in step 1 corrected usage of PD_store in step 5 removed SOR=0 condition for SYS=0,FSD=1 and SYS=1 in step 21 corrected usage of PD_store in step 25 18.10 new interface chapter 18.11 corrected notes 2 and 3 for register table modified description of address offset for RAM 1a data fields

		corrected notes 1 and 2 for RAM1 table corrected register names and new registers for region EXT removed registers DPLL_(CTN_MIN, CTN_MAX, CSN_MIN, CSN_MAX) corrected notes 3 and 4 for region EXT table additional information for other devices 18.12.1 additional note for SNU improved note for RMO 18.12.2 additional note for SYN_NS 18.12.6 removed note about device 4 18.12.7 detailed information about new PMTR 18.12.11 additional note 18.12.13 additional note 18.12.15 additional note for NUSE, FSS, SYN_S, SYN_S_OLD and VSN 18.12.25 additional note for APS_1C2_EXT, APS_1C2_STATUS and APS_1C2_OLD 18.12.31 removed note for special devices 18.12.50ff additional information about negative PD and PD_S 18.12.74 removed information about device 4 18.12.80ff corrected description about FTD/FSD=1 18.12.88-91 new register note 18.12.90 additional details about PD_S 18.12.92-94 removed information about device 4 18.12.94 additional notes for ACB_x 18.12.95 new bits STATE_EXT and WSTATE_EXT 18.12.110-115 new registers DPLL_(INC_CNT1_MASK, INC_CNT2_MASK, NUSC_EXT1, NUSC_EXT2, APS_SYNC_EXT, CTRL_EXT) 18.13.2ff additional information for other devices 18.14.3 modified definition of PD parameter 18.15 new register chapter SPE: 19.1.1 changed figure SPE Submodule integration concept... 19.2.1 changed figure SPE to TOM Connections 19.2.2 changed figure SPE Submodule architecture 19.5.1 reworded bit description TRIG_SEL and ETRIG_SEL ICM:

		20.2.1 additional details about new SPE and PSM CI registers 20.2.4 MCS channels changed from 32 to 9 20.2.5 additional details about new TOM and ATOM CI registers 20.2.6 additional details about new SPE CEI register 20.2.7 additional details about new PSM CEI register 20.2.9 MCS channels changed from 32 to 9 changed last MCS instance to 7 20.2.10 new register ICM_IRQG_CLS_[i]_MEI 20.4 changed table at all 20.5.1 additional notes to AEI_IRQ, SPE0_IRQ and PSM0_CH0_IRQ removed description and note to PSM1_CH0_IRQ new bits SPE4_IRQ and SPE5_IRQ 20.5.2 additional note to DPLL_EDI_IRQ 20.5.3 additional note to TIM0_CH0_IRQ 20.5.4 additional note to TIM4_CH0_IRQ new bits TIM7_CHx_IRQ 20.5.5 additional note to MCS0_CH0_IRQ 20.5.6 additional note to MCS4_CH0_IRQ new bits MCS7_CHx_IRQ 20.5.7-9 additional notes to TOM(0, 2, 4)_CH0_IRQ 20.5.10-12 additional notes to ATOM(0, 4, 8)_CH0_IRQ 20.5.13 additional note to GTM_EIRQ new bits TIM7_EIRQ and MCS7_EIRQ 20.5.14 new bits FIFO2_CH0_EIRQ to FIFO2_CH7_EIRQ 20.5.16 new bits TIM7_CH0_EIRQ to TIM7_CH7_EIRQ 20.5.18 new bits MCS7_CH0_EIRQ to MCS7_CH7_EIRQ 20.5.19ff changed range ICM_IRQG_MCS0_CI to ICM_IRQG_MCS6_CI to ICM_IRQG_MCS[i]_CI changed range ICM_IRQG_MCS0_CEI to ICM_IRQG_MCS6_CEI to ICM_IRQG_MCS[i]_CEI new registers ICM_IRQG_SPE_CI, ICM_IRQG_SPE_CEI, ICM_IRQG_PSM_0_CI, ICM_IRQG_PSM_0_CEI, ICM_IRQG_TOM_[k]_CI,

		ICM_IRQG_ATOM_[k]_CI and ICM_IRQG_CLS_[k]_MEI CMP: 21.1 changed instance naming to new cluster structure 21.1.1 changed figure **Architecture of the Compare Unit** 21.2 changed bitwise compare table included TBWC unit table 21.4 additional information about new primary port MON: 22.1.1 changed figure **MON Block Diagram** 22.2 changed CMU clocks to 9 22.4ff changed number of maximum MCS instances to 10 changed number of maximum MCS channels to 9 22.8.1 new bits ACT_CMU8 and MCS7_ERR to MCS9_ERR 22.8.3 new bits MCA_7_0 to MCA_7_7 22.8.4 changed index i to z removed bits MCA_9 to MCA_31 APP: 23.6 removed wrong figure number changed figure **GTM internal functional dependencies**
3.1.2.0	17.12.2014	GTM v3.1.2.0 changes to v3.1.1.0 at all: Each Table and figure do have their own headlines. at all Register Overview tables: make clear to XML output GTM: 2.4.2 Information about cyclic event compare 2.9.15 new note for bit CFG_CLOCK_RATE 2.9.18 new note for bit TIM_EXT_CAP_EN 2.9.19 new note ARU:

		3.1 introduction of keyword ARU read ID 3.4.1.1 description of dynamic routing ring mode improved CMU: 8.1.1 updated CMU block diagram 8.8.11 included bit level definition CCM: 9.4.2 bits TBU_DIR1 and TBU_DIR2 added 9.4.3 only read mode for reserved bits decimal bit level description changed to binary TIM: 11.1.5 changed figure **EXTCAPSRC Block Diagram** 11.4.2.2 detailed information about pulse times and DSL 11.4.2.2.1 ??? ATOM: 13.1.2.2 changed "signed" with "cyclic event" in a note 13.2.1 changed "signed" with "cyclic event" twice 13.3.2.3.9 reworded capture event for SR0/1 13.6.9 reworded bit description EUPM DTM: 14.1.4 detailed information to neighbored DTM instances 14.2.1 reduced to one channel overview figure 14.6 changed reset priority of SHUT_OFF_SYNC from high to low 14.9.2 new bit I1SEL_0 MCS: 15.7 updated register set XOREG by register DSTAX added register set WXREG and BAREG 15.7.2 new note about ERR bit 15.7.9 changed XOREG to BAREG for register B 15.7.11 changed XOREG to BAREG for register B 15.7.43ff updated description of instructions DIVU and DIVS removed instruction MODU (modulo result is now part of DIVU and DIVS)

		17.7.70ff changed XOREG to OREG for register A changed XOREG to WXREG for register B 17.7.72 replaced WURLE by WUCE instruction removed instruction WUBS (can be realized by WURMX) 15.9.3 removed MODU for bit ERR 15.9.13 added note for MCS instance 0 15.11.11 adapted for bit EN_XOREG DPLL: 18.6.2.7 additional formula DPLL-5c 18.6.3.7 additional formula DPLL-10c 18.6.4.4 additional formula DPLL-5c 18.6.5.4 additional formula DPLL-10c 18.8.6.7 removed TOR=0 condition for SYT=1 in step 1 18.11.1 changed table structure as usual as other register overview 18.11.2ff spliited ram region 1 table into 4 tables 18.11.6 changed table structure as usual as other register overview 18.12.1 additional information for SNU and TNU 18.12.2 additional information for SYN_NS, SYN_NT, SYSF, SSL and TSL 18.12.31 NOAC is defined in appendix B 18.12.99-102 re-added registers DPLL_(CTN_MIN, CTN_MAX, CSN_MIN, CSN_MAX) 18.12.109 additional information to bit STA_FLAG_s, INC_CNT1_FLAG and INC_CNT2_FLAG about visible to MCS0 18.12.110ff removed information about visible to MCS0 18.14.1ff additional information about starting index ICM: 20.5.19ff removed bit 8 for MCS channel 8 MON: 22.8.4 removed bits MCA_8 to MCA_9
3.1.3.0	13.05.2015	GTM v3.1.3.0 changes to v3.1.2.0 at all: Each Table and figure do have their own headlines.

		at all Register Overview tables: make clear to XML output GTM: 2.1 Changed title of figure 2.1.1 2.2.1.2 changed description for returned 0b10 2.9.5 new register GTM_AEI_STA_XPT 2.9.20 changed initial values to 0bx additional hints for usage CFG_CLOCK_RATE FIFO: 5.2.2 precising add data behaviour CMU: 8.3 removed alternatively CLK6 sentence corrected signal name to CMU_ECLK1_EN corrected description for ECLK usage. CCM: 9.4.2 changed initial value for CLS_CLK_DIV to 0bxx 9.4.5 changed bit vector to 15:0 9.4.6 changed initial value from SIZE to 0b0011 TBU: 10.1 precising description of the channel usage corrected TBU_CH[y]_BASE equation 10.1.1 updated figure, corrected usage TBU_TS1+2 and TBU_TC1+2 10.4.7 changed TBU_TSx to TBU_TCx 10.4.10 changed TBU_TSx to TBU_TCx TIM: 11.1.3 corrected register name to TIM[i]_IN_SRC 11.4.2.2 corrected DSL usage TOM: 12.3.3 precising description about CN0 counter 12.3.6.1 additional not about up-down counter mode ATOM: 13.3.2.3.5 precising description about usage EUPM=1 13.3.2.3.8 precising description about ARU Non-Blocking mode 13.3.2.3.9 precising description about ARU Blocking mode

		13.3.2.4 precising description about EUPM and ABM in SOMC mode 13.3.3.1ff additional not about up-down counter mode 13.3.3.5.2 precising description to avoid an update 13.3.3.9 additional description for CLK_SRC_SR and FREEZE in SOMP mode 13.3.4.10 additional description for CLK_SRC_SR in SOMS mode 13.3.5.2 corrected register name ATOM[i]_CH[x]_CTRL for ACB bits 13.3.5.4 additional description for ACB[4:2], EUPM and ABM in SOMB mode 13.6.7 additional hints for FUPD_CTRL0 13.6.9 precising description for EUPM, CLK_SRC_SR, ABM and FREEZE DTM: 14.1.2ff corrected signal input names for TIM_CH_IN 14.1.4 additional note 14.2.4ff corrected index for channel, DTM_IN and DTM_OUT 14.3.1 corrected note 14.6 corrected description of SHUT_OFF 14.9.2 additional notes to XDT_EN bits MCS: 15.2 changed bit width of common triger register to 24 15.6 precising ADC interface 15.7.21ff precising instruction BRD, BWR, BRDI, BWRI, SHR, SHL, MULU, MULS, DIVU, DIVS and WUCE 15.11.11 additional not for HLT_AEIM_ERR 15.11.15 corrected note DPLL: 18.6.2.2 additional note 18.6.2.7 corrected equations 5a2,5b 18.6.3.2 additional note 18.6.3.7 corrected equations 10a2,10b 18.6.4.3 corrected equations 5a6,5b 18.6.5.4 corrected equations 10a6,10b 18.7.6.2ff corrected equation 17a additional description about ACBU bit of CTRL11 register 18.7.8.2ff corrected equation 20a

		additional description about ACBU bit of CTRL11 register 18.10.2.1 corrected table 18.12.14 additional note for NUTE 18.12.15 additional note for NUSE 18.12.79 precising note 18.12.95 additional bits ADT,ADS, WADT and WADS 18.12.114 corrected STATE events to 128 18.15.7 corrected STA_S values 18.15.13 corrected STA_S values SPE: 19.5.13 additional note for bits SPE_CTRL_CMD ICM: 20.4 removed ICM_IRQG_8 20.5.1 two additional bit for AEI_IRQ 20.5.13 two additional bit for AEI_EIRQ APP-A: 23.7.2 new compatability notes for MCS
3.1.4.0	12.06.2015	GTM v3.1.4.0 changes to v3.1.3.0 ARU: 3.4.1.1.1 precising usage of parameter in ring mode DTM: 14.1.4 updated figure; no TIMi_CH2_F_OUT connection DPLL: 18.7.8.2 corrected indices for ACB 18.7.8.3 corrected indices for ACB
3.1.5.0	06.07.2015	GTM v3.1.5.0 changes to v3.1.4.0 GTM: 2.9.20 additional note for cmu frquency limitation CMU:

		8.1ff corrected primary clock source from sys_clk to cls0_clk 8.6 corrected equation $T_{CMU\_ECLK[z]}$ to $T_{CMU\_ECLK[z]\_EN}$ corrected primary clock source from sys_clk to cls0_clk 8.8.11 corrected primary clock source from sys_clk to cls0_clk CCM: 9.4.12 precised meaning of index i MCS: 15.7.2.1 additional information about write access to protected register 15.7.15ff limitation of duration 15.7.19ff corrected : duration and description DPLL: 18.8.6.7 corrected TISI interrupt additional SIP1 information for FTD=0 in step 1 18.12.95 additional SIP=1 information
3.1.5.1	24.03.2016	GTM v3.1.5.1 changes to v3.1.5.0 changed description: GTM_EXT_CAP_EN_[i] to CCM[i]_EXT_CAP_EN GTM_HW_CONF to CCM[i]_HW_CONF GTM: 2.2.1.2 additional note for behaviour of register CMU_CLK_CTRL 2.4.1 information about absolute angle clock precising 2.8.1 register decription TIM_AUX_IN_SRC, HW_CONF, TOM_OUT, ATOM_OUT and EXT_CAP_EN moved to chapter CCM 2.9.10 additional note for write protection ARU: 3.7.15 additional note for behaviour of register ARU_CADDR

		FIFO: 5.2.2 information about continuous data stream precised CMU: 8.7.4 description for CLK_CNT corrected CCM: 9.4.2 register name corrected 9.4.8ff register decription TIM_AUX_IN_SRC, HW_CONF, TOM_OUT, ATOM_OUT and EXT_CAP_EN moved from chapter 2 TBU: 10.1.1 block diagramm corrected TOM: 12.8.9 description of bit 20 RST_CCU0 corrected DTM: 14.2.2 statement about standard dead time removed MCS: 15.2.1 MCS architecture figure desciption precised 15.4 maximum write access time corrected 15.7.24,26 additional information about MHB register 15.7.75 additional information about subtraction result 15.11.19 MCS[i]_MEM description included DPLL: 18.7.1-4 Note about "All 5 steps.." moved from the end to the beginning of the subchapter 18.8.3 description of SUB_INCx generation moved from end to the beginning of the subchapter 18.12.3-6 Note about protection moved to the end of the bit field description 18.12.11,13,15,25,88-91,94,112-117 Note about usage moved to the end of the bit field description 18.12.14 additional note usage og recent TRIGGER events 18.12.30 description about pointer behaviour removed for SOR and TOR 18.12.94 NOAC limitation removed

		18.12.95 description for INCF1 precised 18.12118ff DPLL MEM description included 18.13.3 Action description corrected 18.15 MS2DPLL interface precised CMP: 20.1 additional note for combination frequency and comparison App: 23.7.1 additional hint 2 23.7.2 additional hint 7

25 Conventions

The following conventions are used within this document.

ARIAL BOLD CAPITALS	Names of register and register bits
Arial italic	Names of signals
`Courier`	Extracts of files

26 References

This document refers to the following documents.

Ref	Authors(s), Title, Revision
1	AE/PJ-SCI, GTM-IP Specification Appendix B, v3.1.5.1

27 Disclaimer

LEGAL NOTICE

The content of this document is subject to continuous developments and improvements. All particulars and its use contained in this document are given by BOSCH in good faith.

NO WARRANTIES: TO THE MAXIMUM EXTENT PERMITTED BY LAW, NEITHER THE INTELLECTUAL PROPERTY OWNERS, COPYRIGHT HOLDERS AND CONTRIBUTORS, NOR ANY PERSON, EITHER EXPRESSLY OR IMPLICITLY, WARRANTS ANY ASPECT OF THIS SPECIFICATION, SOFTWARE RELATED THERETO, CODE AND/OR PROGRAM RELATED THERETO, INCLUDING ANY OUTPUT OR RESULTS OF THIS SPECIFICATION, SOFTWARE RELATED THERETO, CODE AND/OR PROGRAM RELATED THERETO UNLESS AGREED TO IN WRITING. THIS SPECIFICATION, SOFTWARE RELATED THERETO, CODE AND/OR PROGRAM RELATED THERETO IS BEING PROVIDED "AS IS", WITHOUT ANY WARRANTY OF ANY TYPE OR NATURE, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, AND ANY WARRANTY THAT THIS SPECIFICATION, SOFTWARE RELATED THERETO, CODE AND/OR PROGRAM RELATED THERETO IS FREE FROM DEFECTS.
ASSUMPTION OF RISK: THE RISK OF ANY AND ALL LOSS, DAMAGE, OR UNSATISFACTORY PERFORMANCE OF THIS SPECIFICATION (RESPECTIVELY THE PRODUCTS MAKING USE OF IT IN PART OR AS A WHOLE), SOFTWARE RELATED THERETO, CODE AND/OR PROGRAM RELATED THERETO RESTS WITH YOU AS THE USER. TO THE MAXIMUM EXTENT PERMITTED BY LAW, NEITHER THE INTELLECTUAL PROPERTY OWNERS, COPYRIGHT HOLDERS AND CONTRIBUTORS, NOR ANY PERSON EITHER EXPRESSLY OR IMPLICITLY, MAKES ANY REPRESENTATION OR WARRANTY REGARDING THE APPROPRIATENESS OF THE USE, OUTPUT, OR RESULTS OF THE USE OF THIS SPECIFICATION, SOFTWARE RELATED THERETO, CODE AND/OR PROGRAM RELATED THERETO IN TERMS OF ITS CORRECTNESS, ACCURACY, RELIABILITY, BEING CURRENT OR OTHERWISE. NOR DO THEY HAVE ANY OBLIGATION TO CORRECT ERRORS, MAKE CHANGES, SUPPORT THIS SPECIFICATION, SOFTWARE RELATED THERETO, CODE AND/OR PROGRAM RELATED THERETO, DISTRIBUTE UPDATES, OR PROVIDE NOTIFICATION OF ANY ERROR OR DEFECT, KNOWN OR UNKNOWN. IF YOU RELY UPON THIS SPECIFICATION, SOFTWARE RELATED THERETO, CODE AND/OR PROGRAM RELATED THERETO, YOU DO SO AT YOUR OWN RISK, AND YOU ASSUME THE RESPONSIBILITY FOR THE RESULTS. SHOULD THIS SPECIFICATION, SOFTWARE RELATED THERETO, CODE AND/OR PROGRAM RELATED THERETO PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL LOSSES, INCLUDING, BUT NOT LIMITED TO, ANY NECESSARY SERVICING, REPAIR OR

CORRECTION OF ANY PROPERTY INVOLVED TO THE MAXIMUM EXTEND PERMITTED BY LAW.

DISCLAIMER: IN NO EVENT, UNLESS REQUIRED BY LAW OR AGREED TO IN WRITING, SHALL THE INTELLECTUAL PROPERTY OWNERS, COPYRIGHT HOLDERS OR ANY PERSON BE LIABLE FOR ANY LOSS, EXPENSE OR DAMAGE, OF ANY TYPE OR NATURE ARISING OUT OF THE USE OF, OR INABILITY TO USE THIS SPECIFICATION, SOFTWARE RELATED THERETO, CODE AND/OR PROGRAM RELATED THERETO, INCLUDING, BUT NOT LIMITED TO, CLAIMS, SUITS OR CAUSES OF ACTION INVOLVING ALLEGED INFRINGEMENT OF COPYRIGHTS, PATENTS, TRADEMARKS, TRADE SECRETS, OR UNFAIR COMPETITION.

INDEMNIFICATION: TO THE MAXIMUM EXTEND PERMITTED BY LAW YOU AGREE TO INDEMNIFY AND HOLD HARMLESS THE INTELLECTUAL PROPERTY OWNERS, COPYRIGHT HOLDERS AND CONTRIBUTORS, AND EMPLOYEES, AND ANY PERSON FROM AND AGAINST ALL CLAIMS, LIABILITIES, LOSSES, CAUSES OF ACTION, DAMAGES, JUDGMENTS, AND EXPENSES, INCLUDING THE REASONABLE COST OF ATTORNEYS' FEES AND COURT COSTS, FOR INJURIES OR DAMAGES TO THE PERSON OR PROPERTY OF THIRD PARTIES, INCLUDING, WITHOUT LIMITATIONS, CONSEQUENTIAL, DIRECT AND INDIRECT DAMAGES AND ANY ECONOMIC LOSSES, THAT ARISE OUT OF OR IN CONNECTION WITH YOUR USE, MODIFICATION, OR DISTRIBUTION OF THIS SPECIFICATION, SOFTWARE RELATED THERETO, CODE AND/OR PROGRAM RELATED THERETO, ITS OUTPUT, OR ANY ACCOMPANYING DOCUMENTATION.

GOVERNING LAW: THE RELATIONSHIP BETWEEN YOU AND ROBERT BOSCH GMBH SHALL BE GOVERNED SOLELY BY THE LAWS OF THE FEDERAL REPUBLIC OF GERMANY. THE STIPULATIONS OF INTERNATIONAL CONVENTIONS REGARDING THE INTERNATIONAL SALE OF GOODS SHALL NOT BE APPLICABLE. THE EXCLUSIVE LEGAL VENUE SHALL BE DUESSELDORF, GERMANY.

MANDATORY LAW SHALL BE UNAFFECTED BY THE FOREGOING PARAGRAPHS.

INTELLECTUAL PROPERTY OWNERS/COPYRIGHT OWNERS/CONTRIBUTORS: ROBERT BOSCH GMBH, ROBERT BOSCH PLATZ 1, 70839 GERLINGEN, GERMANY AND ITS LICENSORS.