BOSCH
Invented for life

GTM-IP

Generic Timer Module

GTM-IP Specification

Revision: 1.5.5.1

(Released on 20.06.2013)

Robert Bosch GmbH
Automotive Electronics (AE)

Copyright Notice and Proprietary Information

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.

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

References

This document refers to the following documents.

Ref	Authors(s), Title, Revision		
1	AE/EIN2,	GTM-IP Specification Appendix B,	1.5.5.1

Terms and Abbreviations

This document uses the following terms and abbreviations.

Term	**Meaning**
GTM	Generic Timer Module
IRC	Interrupt Controller
DPLL	Digital Phase Locked Loop
FULL_SCALE	Range in which all positions/values depend on the information of TRIGGER and STATE signals
HALF_SCALE	Range in which all positions/values depend on the information of TRIGGER signal only; two consecutive HALF_SCALE periods form a FULL_SCALE period
TS	Time stamp representation
PS	Position (or value) stamp representation; common description
[i]	Numbering of Instances of a module (e.g. ATOM[i] references to instance i of module ATOM)
[x]	Numbering of Channels of a module (e.g. ATOM[i] references to channel x of instance i of module ATOM)
div	Calculates the quotient of integer division
mod	Calculates the remainder of integer division

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 submodules of different functionality. These submodules 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 submodules are located around a central routing unit (called Advanced Routing Unit (ARU)). The ARU can connect the submodules 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 submodules have dedicated functionalities, 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 submodules 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 *infrastructural components* further on. These components are present in all GTM variants. However, the number of these components may vary from device to device.

Other submodules have a more general architecture and can fulfil typical timer functions, e.g. there are PWM generation units. The third class of submodules are those fulfilling a dedicated functionality for a certain application domain, e.g. the DPLL serves engine management applications. A fourth group of submodules is responsible for supporting the implementation of safety functions to fulfil 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 submodules coming from those four groups. The application class is defined by the amount of components of those submodules integrated into the implemented GTM-IP.

1.2 Document Structure

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

The following chapters 3 up to 9 deal with the so called infrastructural components for routing, clock management and common time base functions. Chapters 10 to 12 describe the signal input and output modules while the following chapter 13 explains the signal processing and generation submodule. Chapter 14 outlines a memory configuration module for the described signal processing and generation submodule. The next sections provide a detailed description of application specific and safety related modules like the MAP, DPLL, SPE, CMP and MON submodules. Chapter 18 describes a module that bundles several interrupts coming from the other submodules and connect them to the outside world.

These submodule groups are shown in the following table:

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

2 GTM Architecture

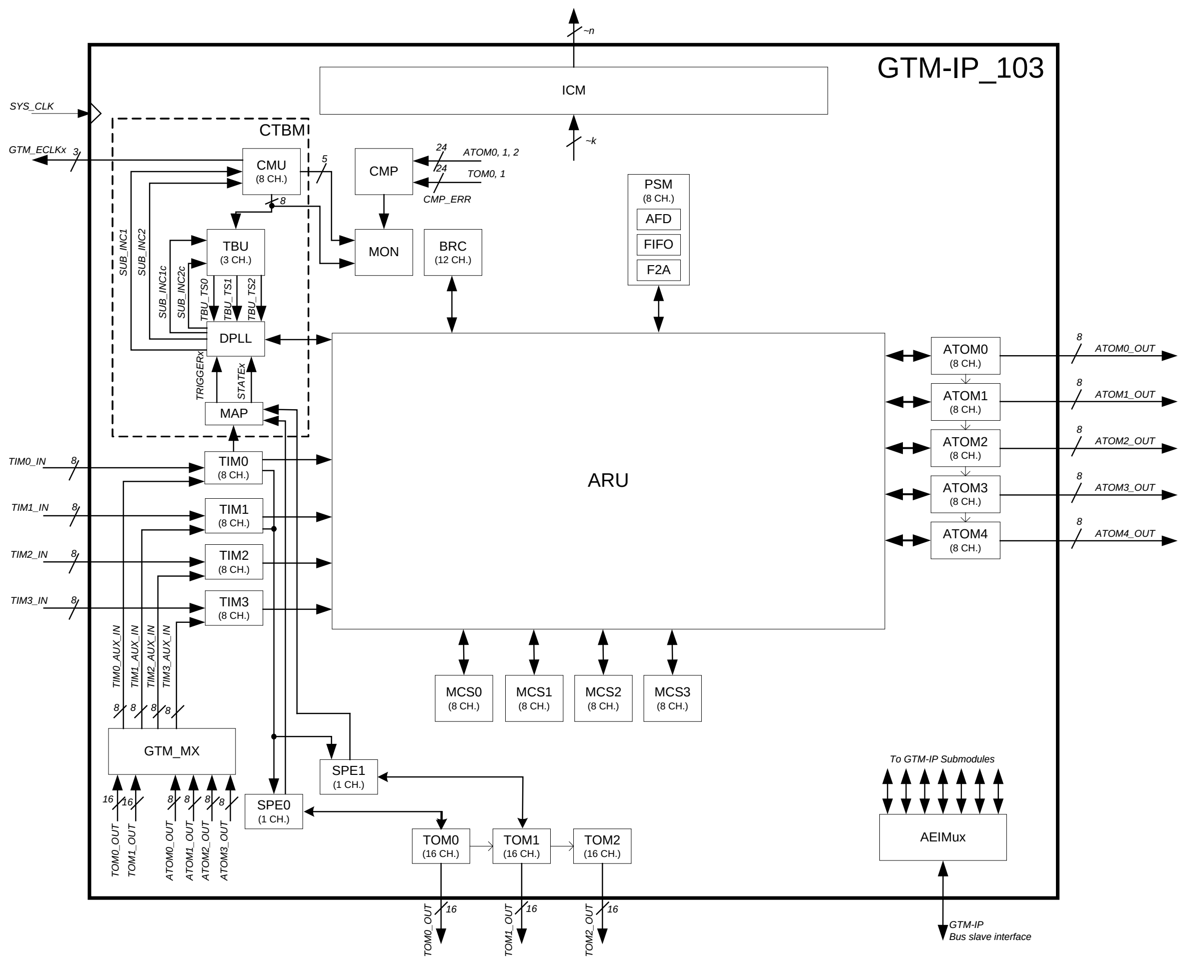
2.1 Overview

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 count of submodules (i.e. ATOM, BRC, MCS, PSM, SPE, TIM, TOM) and different count of channel per submodule (if applicable) are possible. In this section as an example of possible device configurations the GTM-IP_103 realization is outlined. The architecture of the GTM-IP_103 is depicted in figure 2.1.1. Please note, that the size of the submodules in the figure does not reflect the die size of the modules in the final RTL implementation.
The device dependent configuration (i.e the count of submodules) is listed in the device specific appendix B of this document [1].

In this section as an example of possible device configurations the GTM-IP_103 realization is outlined. The architecture of the GTM-IP_103 is depicted in figure 2.1.1.

Please note, that the size of the submodules in the figure does not reflect the die size of the modules in the final RTL implementation.
The device dependent configuration (i.e. the count of submodules and channels per submodule) is listed in the device specific appendix B of this document [1].

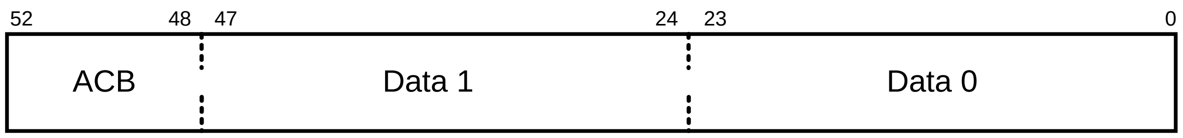
2.1.1 GTM Architecture Block Diagram



The central component of the GTM-IP is the Advanced Routing Unit (ARU) where most of the submodules 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 submodule to a connected destination submodule. 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 submodule to another. These ARU Control Bits (ACB) can have a different meaning for different submodules.
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 submodule 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 submodule consists of three subunits, 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 subunit, 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 subunit 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 subunit is the interface between the FIFO subunit 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 submodule 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 submodules are typically driven from this submodule. 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 microcontroller 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 submodule. In this submodule the incoming signals are filtered and transferred to the MAP submodule where two of these six signals are selected for further processing inside the DPLL.

Signal outputs are generated with the 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 submodule 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 submodule 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 submodule can integrate functions to drive one BLDC engine. This BLDC support is established together with the TIM and Sensor Pattern Evaluation (SPE) submodule.

The ATOMs offer the additional functionality to generate complex output signals without CPU interaction by serving these complex waveform characteristics by other submodules 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 submodule 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 submodule 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.

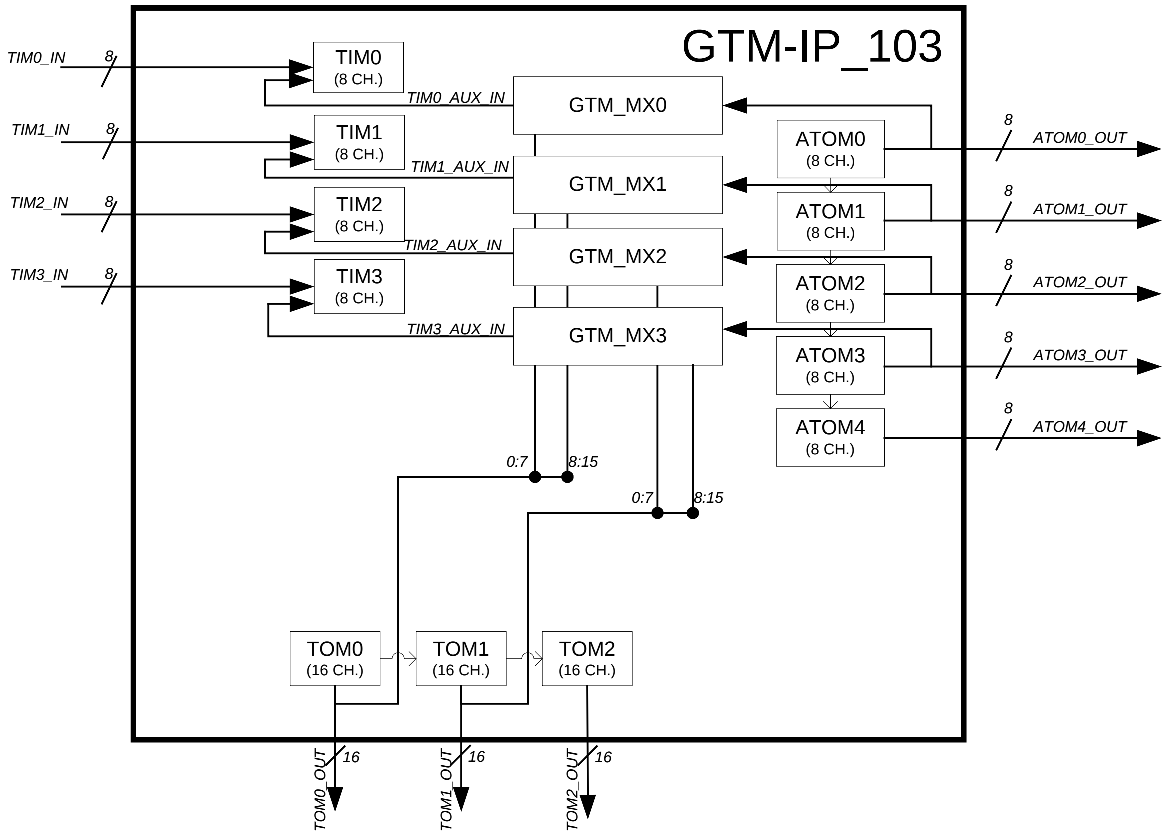
Together with the MCS the ATOM is able to generate an arbitrary predefined output sequence at the GTM-IP output pins. The output sequence is defined by instructions located in RAM connected to the MCS submodule. The instructions define the points were an output signal should change or to react on other 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 an ATOM or TOM and sends the result to the MON submodule were the error is signalled to the CPU. The MON module is also able to monitor the ARU and CMU activities.

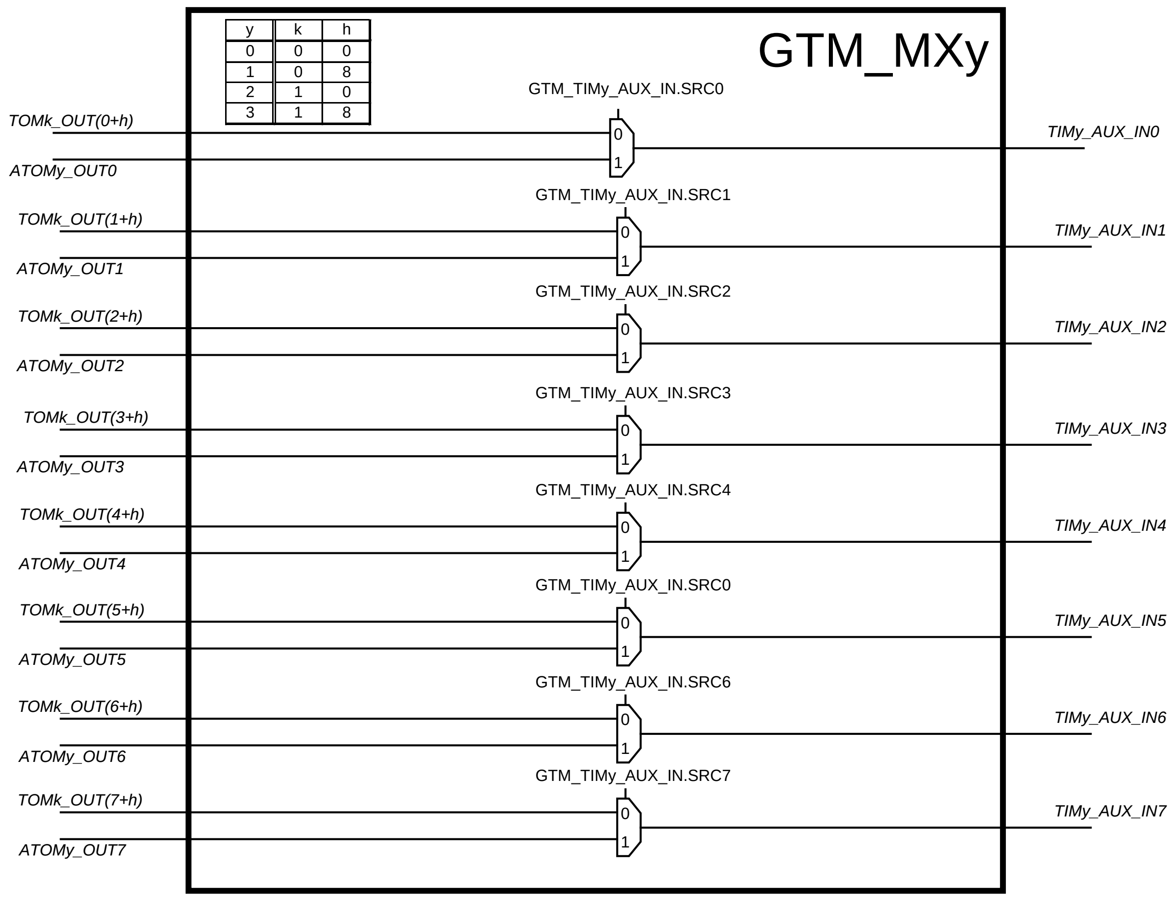
In the described implementation the submodules of the GTM-IP have a huge amount of different interrupt sources. These interrupt sources are grouped and concentrated by the Interrupt Concentrator Module (ICM) to form a much easier manageable bunch of interrupts that are visible outside of the GTM-IP.

On the GTM-IP toplevel there are some configurable signal connections from the signal output of the TOM, ATOM modules to the input signals of the TIM modules.

2.1.3 GTM-IP signal multiplex



The next diagram gives an overview of the connectivity. The source selection is defined with the bit **SRXx** in the register **GTM_TIM[y]_AUX_IN_SRC** .



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 submodule TIM and to the GTM-IP output submodules TOM and ATOM.

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 submodules. These interrupt lines are concentrated inside the ICM and have to be adapted to the dedicated microcontroller 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:

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:

AEI_STATUS	Description
00	No Error
01	Illegal Byte Addressing
10	Illegal Address Access
11	Unsupported Address

The signal value "00" is driven if no error occurred during AEI access.
The signal value "01" is driven if the bus address is not an integer multiple of 4 (byte addressing).
The signal value "11" is driven if the address is not handled in the GTM.
The signal value "10" is driven if an illegal write access to one of the following register is performed:
a) register is protected (e.g. protected by bit RF_PROT).
In case of an illegal write access signaled by status "10" the register will not be modified.

Reading registers will never return status "10".

Write access to following addresses returns status "10" under special conditions:

ARU_IRQ_FORCINT
ATOM[i]_CH[x]_CM0
ATOM[i]_CH[x]_CM1
ATOM[i]_CH[x]_SR0
ATOM[i]_CH[x]_SR1
ATOM[i]_CH[x]_RDADDR
ATOM[i]_CH[x]_IRQ_FORCINT
BRC_IRQ_FORCINT
BRC_SRC_[x]_ADDR (x=0..11)
CMP_IRQ_FORCINT

CMU_GCLK_NUM
CMU_GCLK_DEN
CMU_CLK[x]_CTRL (x=0..7)
CMU_ECLK_NUM
CMU_ECLK_DEN
CMU_FXCLK_CTRL
DPLL_ACT_STA
DPLL_OSW
DPLL_IRQ_FORCINT
DPLL_ID_PMTR_[x] (x=0..23*)
DPLL_INC_CNT1
DPLL_INC_CNT2
DPLL_TSAC[x] (x=0..23*)
DPLL_PSAC[x] (x=0..23*)
DPLL_ACB_[x] (x=0..5)**
F2A[i]_CH[x]_ARU_RD_FIFO
F2A[i]_CH[x]_STR_CFG
FIFO[i]_CH[x]_IRQ_FORCINT
GTM_IRQ_FORCINT
GTM_RST
SPE[i]_IRQ_FORCINT
TIM[i]_CH[x]_CNTS
TIM[i]_CH[x]_GPR1
TIM[i]_CH[x]_IRQ_FORCINT
TOM[i]_CH[x]_IRQ_FORCINT
MCS[i]_CH[x]_CTRL
MCS[i]_CH[x]_PC
MCS[i]_CH[x]_IRQ_FORCINT
MCS[i]_CTRL
TBU_CH[x]_BASE
TBU_CH[x]_CTRL
*** 31 for device 4**
**** 7 for device 4**

MCS RAM during initialization
DPLL RAM during initialization

2.2.2 GTM-IP Multi-master and multi-tasking support

To support multi-master and multi-task 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

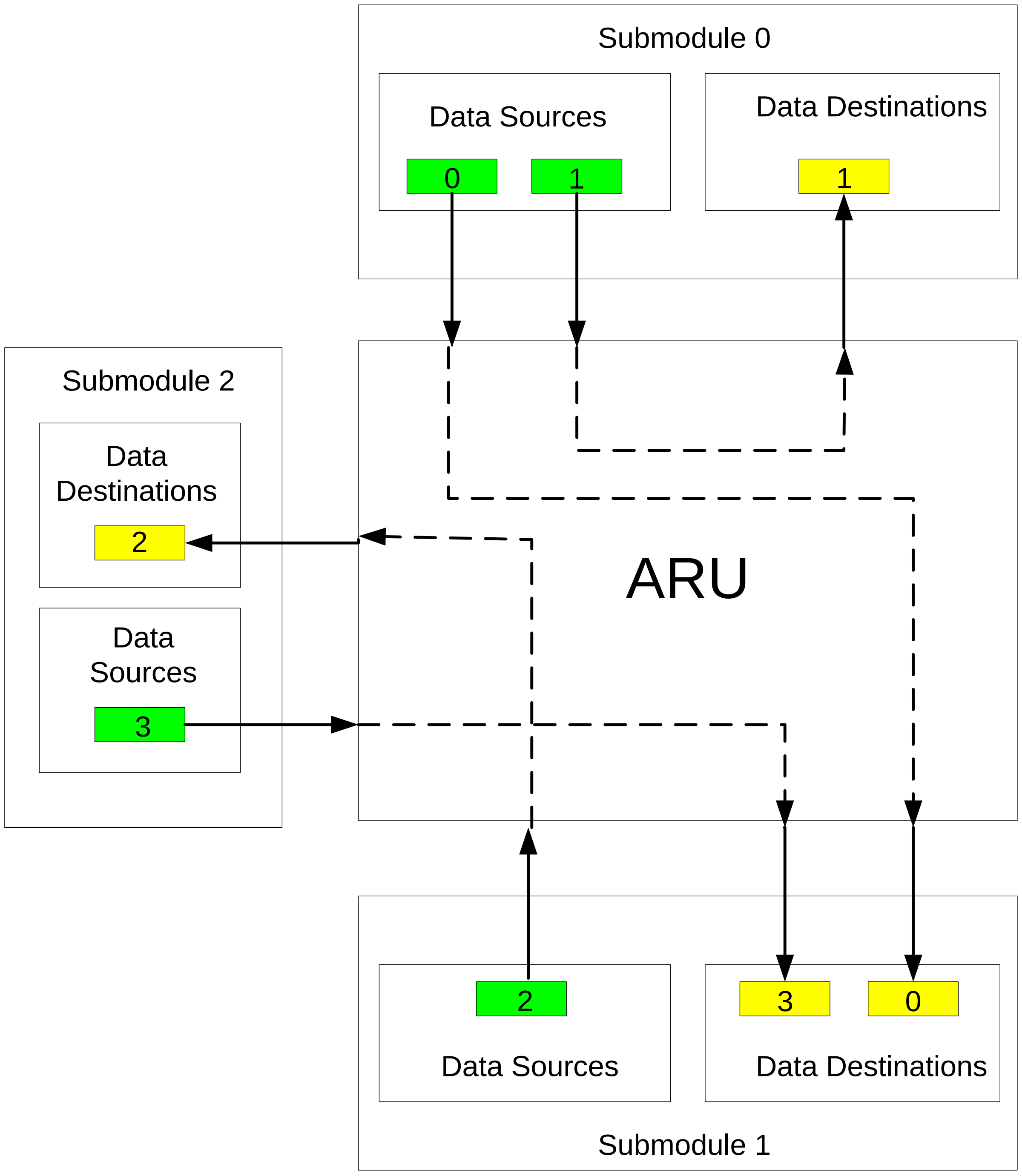
'00' and '11' has no effect on the register bit and where '01' sets the bit and '10' resets the bit. If the CPU wants to read the status of the bit it always gets a '00' if the bit is reset and it gets a '11' if the bit is set.

2.3 ARU Routing Concept

One central concept of the GTM-IP is the routing mechanism of the ARU submodule for data streams. Each data word transferred between the ARU and its connected submodule 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 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 21.3. The connection from a specific data source to a specific data destination is defined by configuring the corresponding address of a data source 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 submodule for which this destructive read access does not hold. This is the BRC submodule 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 submodule 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 channels for the GTM-IP_103 variant used as a reference within this chapter.

Submodule	**Number of data sources**	**Number of data destinations**
ARU	1	0
DPLL	24	24
TIM 0-3	32	0
MCS 0-3	96	32
BRC	22	12
TOM	0	0
ATOM 0-4	40	40
PSM 0	8	8
ICM	0	0
CMP	0	0
MON	0	0
Total	223	116

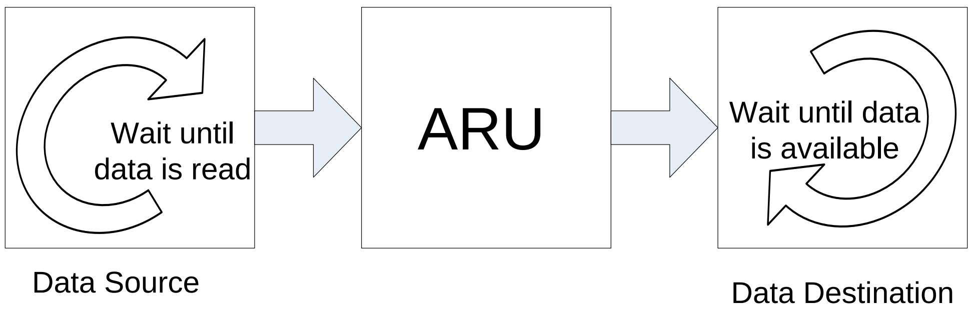
2.3.2 ARU Round Trip Time

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.

2.3.3 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 submodules to synchronize the submodules to the routed data streams. Figure 2.3.3.1 explains the blocking mechanism.

2.3.3.1 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 submodule 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.

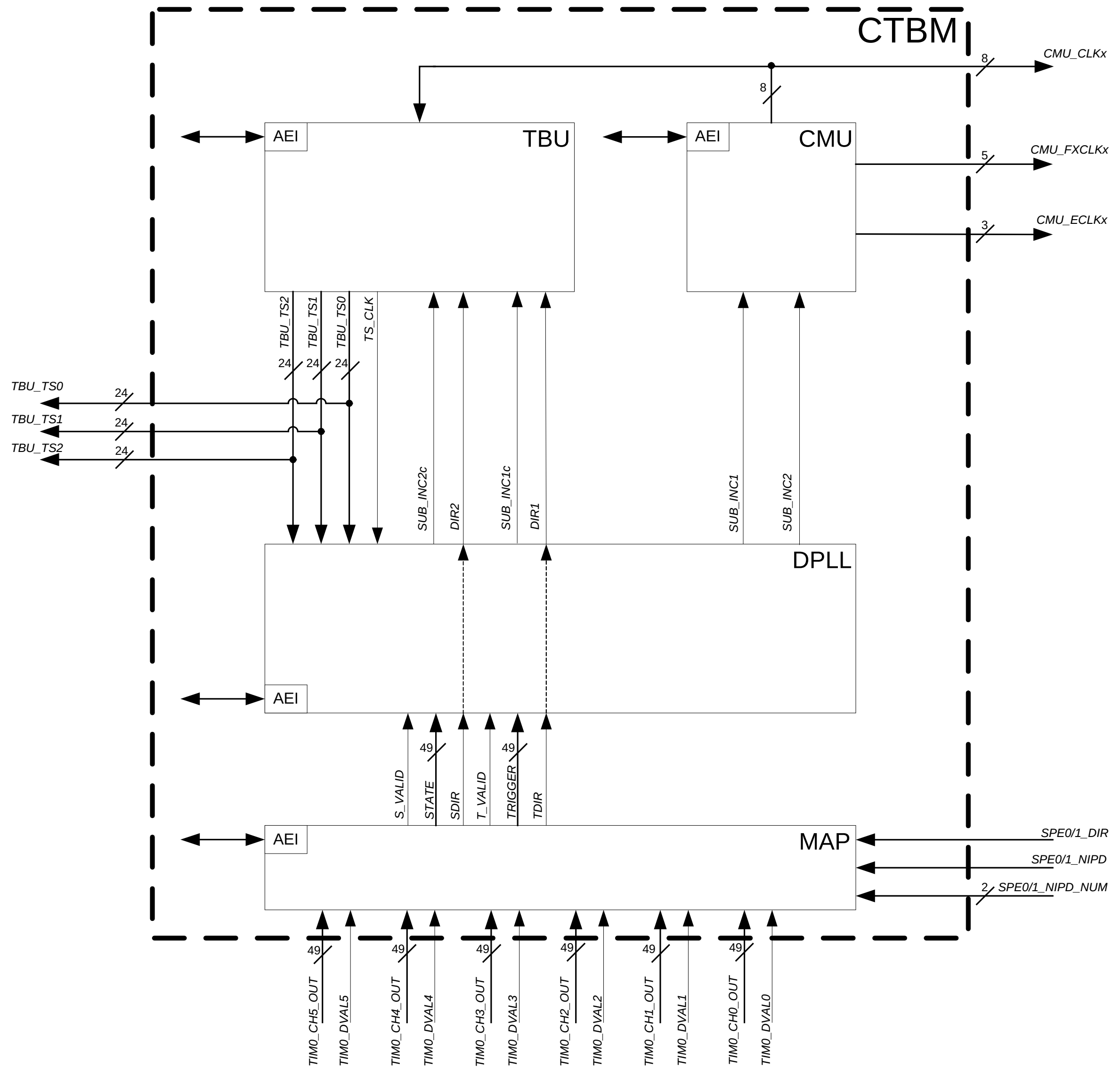
In any case the round trip time for the ARU is always fixed for a specific device configuration.
Please refer to device specific Appendix B of this specification for detailed information [1].

One exception is the BRC submodule 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 subunits 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 13 clocks for the submodules 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 submodule for PWM generation. A maximum of eight (8) *CMU_CLK[x]* (x: 0..7) clocks are used by other submodules of the GTM for signal generation.

Inside the Time Base Unit (TBU) one of these maximum eight 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 submodule 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 16. 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 9.

In this device the TBU submodule 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.

As stated before, the DPLL submodule 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 submodule. For a detailed description of the DPLL functionality please refer to chapter 16.

The MAP submodule is used to select the *TRIGGER* and *STATE* signals for the DPLL out of six input signals coming from TIM0 submodule. Besides this, the MAP submodule 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 submodule please refer to section 15.

2.5 GTM-IP Interrupt Concept

The submodules 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 this interrupt groups the GTM-IP submodule interrupt signals are bundled to a smaller set of interrupts. Out of these interrupt sets a smaller amount of interrupt signals is created and signalled outside of the GTM-IP as a signal *GTM_<MOD>_IRQ,* whereas <MOD> identifies the name of the corresponding GTM-IP submodule.

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

- acknowledge signal from a DMA controller
- validation signal from ADC

- clear signal from an GTM-external interrupt controller to do an atomic clear while entering an ISR routine

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

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 microcontroller 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 behaviour 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 sources 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'.

It should be noted, that 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**.
Note, it is possible for one source to enable the normal interrupt and the error interrupt in parallel. Because of 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).

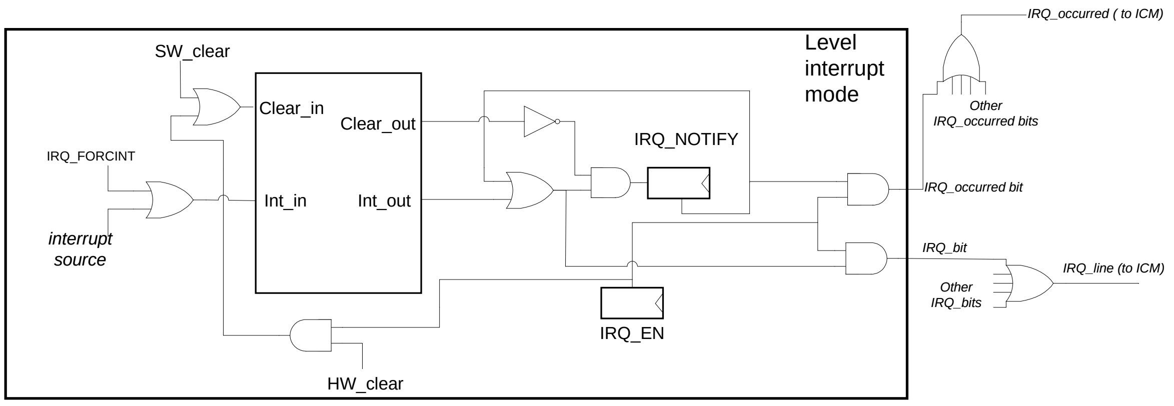
To be able to detect the module source of the error interrupt the ICM provides the register **ICM_IRQG_MEI**.
The error interrupt causing channel can be determined for the modules FIFO, TIM and MCS by evaluating the ICM register **ICM_IRQG_CEI0** to **ICM_IRQG_CEI4**.

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 submodule 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 with a write access with value '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 the 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

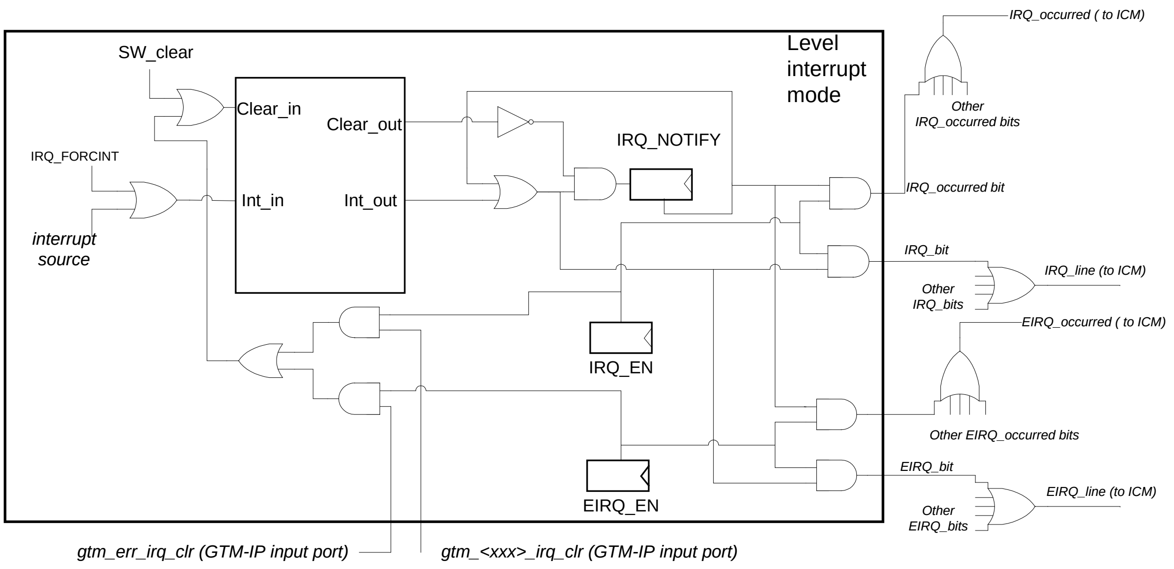
As it can be seen from the figure 2.5.1.1 an occurred interrupt event is signalled as a constant signal level with value 1 to the signal *IRQ_bit*, if the corresponding interrupt is enabled in register **IRQ_EN**.

With exception of the submodules ARU and DPLL, the signal *IRQ_bit* is OR-combined with the neighbouring *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 submodules DPLL and ARU are routed directly as a signal *IRQ_line* to the submodule ICM. In some cases (submodules 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 submodule 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



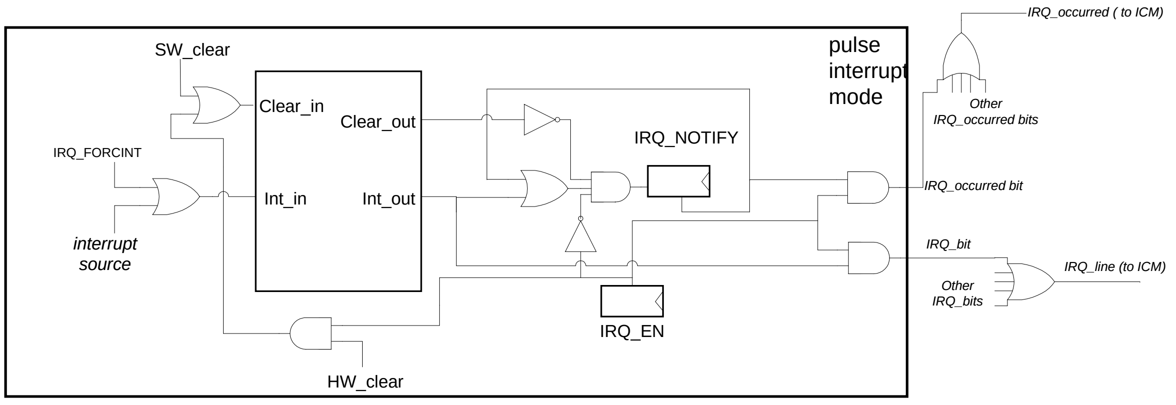
A collected interrupt bit in register **IRQ_NOTIFY** may be cleared by a clear event, which is defined as a pulse on signal *Clear_out* of figure 2.5.1.3. A clear event can be performed with a write access with value '1' to the corresponding bit in the register **IRQ_NOTIFY** leading to a pulse on signals *SW_clear*. A clear event may also result from externally connected signal *gtm_<MOD>_irq_clr* or *gtm_err_irq_clr*, which is routed as a *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 signalled 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 behaviour can be observed from figure 2.5.2.1.

2.5.2.1 Pulse interrupt mode scheme



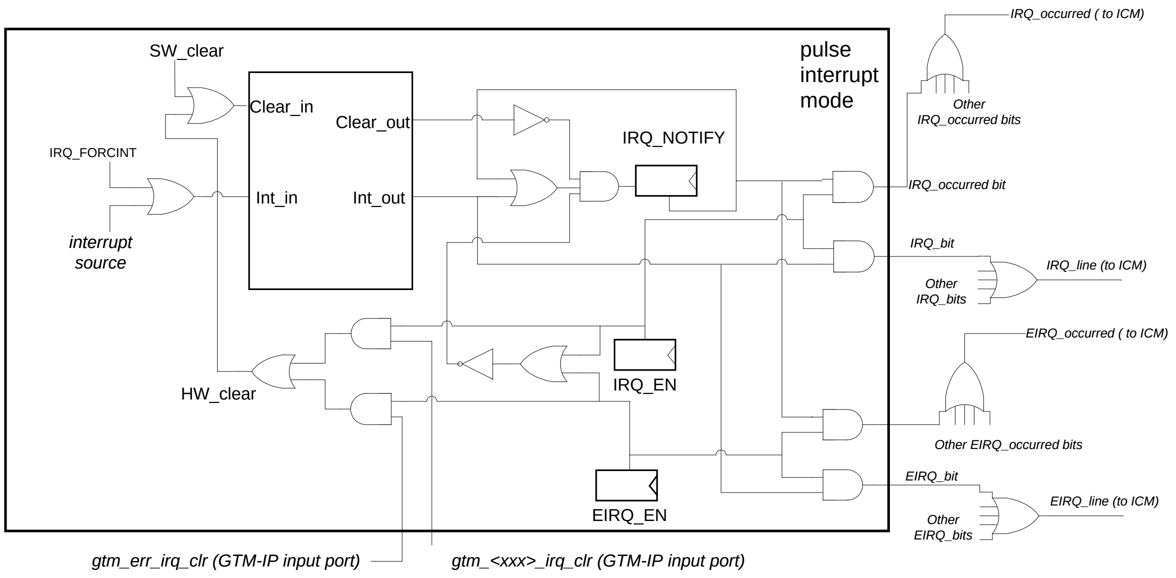
In Pulse Interrupt Mode each Interrupt Event will generate a pulse on the *IRQ_bit* signal if **IRQ_EN** is enabled.

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.

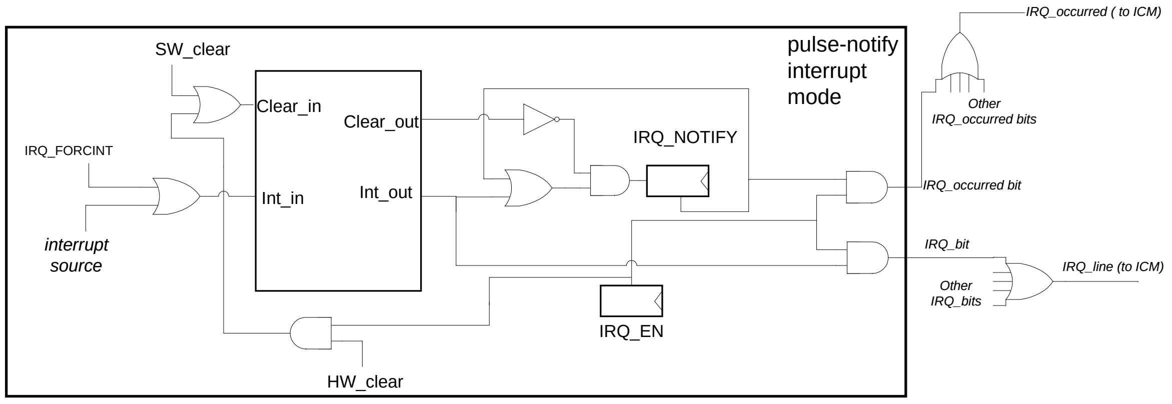
However, if an error interrupt is disabled in the register **EIRQ_EN**, an occurred error interrupt event is captured in the register **IRQ_NOTIFY**, in order to allow polling for disabled error interrupts by software.

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

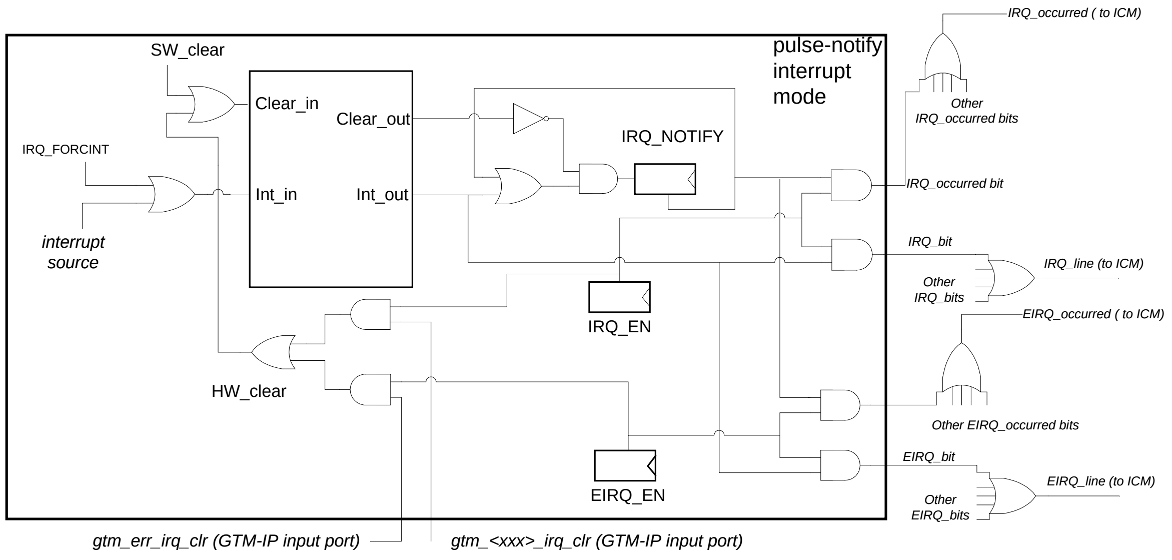
In Pulse-notify Interrupt mode, all interrupt events are captured in the register **IRQ_NOTIFY**. If an interrupt is enabled by the register **IRQ_EN**, each interrupt event will also generate a pulse on the *IRQ_bit* signal. The signal *IRQ_occurred* will be high if interrupt is enabled in register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



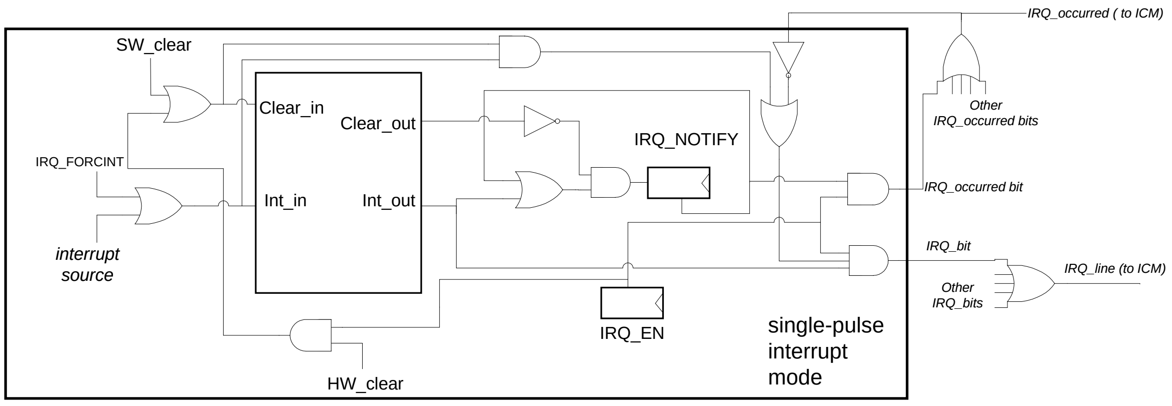
In Pulse-notify Interrupt mode, all error interrupt events are captured in the register **IRQ_NOTIFY**. If an error interrupt is enabled by the register **EIRQ_EN**, each error interrupt event will also generate a pulse on the *EIRQ_bit* signal. The signal *EIRQ_occurred* will be high if error interrupt 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 neighbouring *IRQ_occurred* bits) to the inverter for suppressing further interrupt pulses.

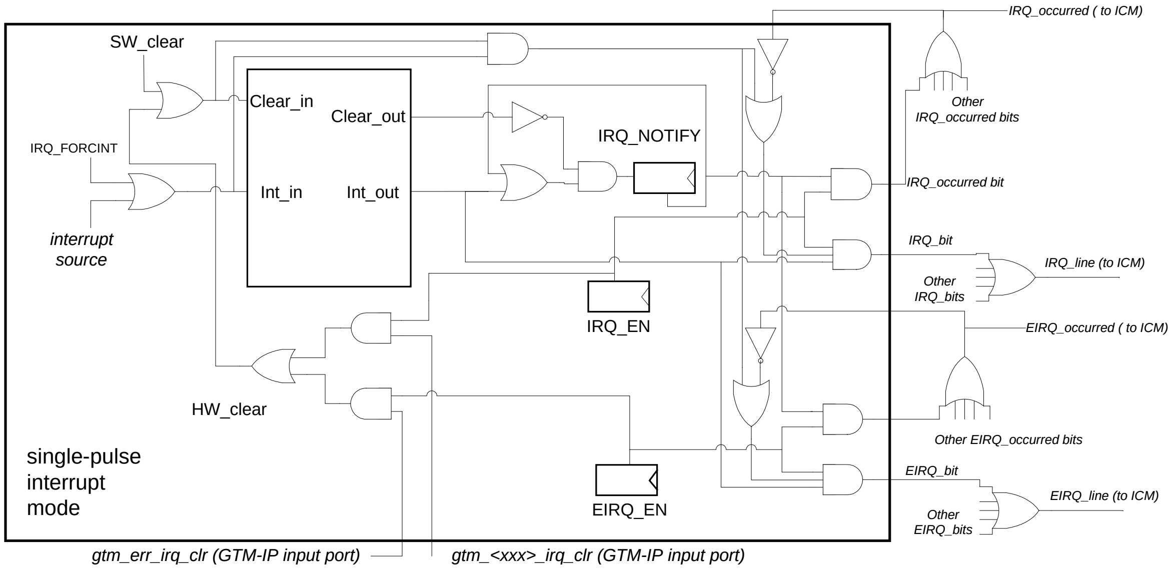
2.5.4.1 Single-pulse interrupt mode scheme



To avoid unexpected IRQ behaviour 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 behaviour 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 neighbouring *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 submodule 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 submodule please refer to chapter 18.

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 behaviour 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 behaviour of some GTM-IP registers with special functionality on behalf of read accesses from the AEI bus interface.

2.6.1 Register behaviour 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 submodule have to be signalled to the outside world, when reached and should for example trigger the software debugger to stop program execution. For this internal state signalling 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 submodules of the GTM-IP are intended to be configured at system start-up to fulfil certain functionality for the application domain the microcontroller 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 behaviour.

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

2.8 GTM-IP TOP-Level Configuration Registers Overview

GTM-IP TOP-level contains following configuration registers:

Register name	**Description**	**Details in Section**
GTM_REV	GTM-IP Version control register	2.9.1
GTM_RST	GTM-IP Global reset register	2.9.2
GTM_CTRL	GTM-IP Global control register	2.9.3
GTM_AEI_ADDR_XPT	GTM-IP AEI Timeout exception address register	2.9.4
GTM_IRQ_NOTIFY	GTM-IP Interrupt notification register	2.9.5
GTM_IRQ_EN	GTM-IP Interrupt enable register	2.9.6
GTM_EIRQ_EN	GTM-IP Error interrupt enable register	2.9.12
GTM_IRQ_FORCINT	GTM-IP Software interrupt generation register	2.9.7
GTM_IRQ_MODE	GTM-IP top level interrupts mode selection. Please note that this mode selection is only valid for the three interrupts described in section 2.9.5	2.9.8
GTM_TIM[i]_AUX_IN_SRC (i= 0.. n)	GTM-IP TIM[i] module AUX_IN source selection register	2.9.13

2.9 GTM-IP 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																															RST
Mode	R																															RAw
Initial Value	0x0000 00																															0

Bit 0 **RST:** GTM-IP Reset.
0 = No reset action
1 = Initiate reset action for all submodules
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 31:1 **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																							TO_VAL					Reserved		TO_MODE	RF_PROT
Mode	R																							RW					R		RW	RW
Initial Value	0x0000 00																							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 signalling 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 31:9	**Reserved** Note: Read as zero, should be written as zero.

2.9.4 Register GTM_AEI_ADDR_XPT

Address Offset:	**see Appendix B**	**Initial Value:**	**0x0000_0000**
	31 … 21	20	19 … 0
Bit	Reserved	TO_W1R0	TO_ADDR
Mode	R	R	R
Initial Value	0x000	0	0x00000

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_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																												AEI_USP_BE	AEI_IM_ADDR	AEI_USP_ADDR	AEI_TO_XPT
Mode	R																												RCw	RCw	RCw	RCw
Initial Value	0x0000000																												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 31:4 **Reserved**
Note: Read as zero, should be written as zero.

2.9.6 Register GTM_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																												AEI_USP_BE_IRQ_EN	AEI_IM_ADDR_IRQ_EN	AEI_USP_ADDR_IRQ_EN	AEI_TO_XPT_IRQ_EN
Mode	R																												RW	RW	RW	RW
Initial Value	0x0000000																												0	0	0	0

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

2.9.7 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_AEI_USP_BE	TRG_AEI_IM_ADDR	TRG_AEI_USP_ADDR	TRG_AEI_TO_XPT
Mode	R																												RAw	RAw	RAw	RAw
Initial Value	0x0000000																												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 31:4	**Reserved** Note: Read as zero, should be written as zero.

2.9.8 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.
00 = Level mode
01 = Pulse mode
10 = Pulse-Notify mode
11 = 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.

2.9.9 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_REG	Reserved		BUFF_OVL	MODE_UP_PGR	Reserved						MSK_WR_RSP	BRG_MODE
Mode	R								R							RAw	R			R	R		RCw	R	R						RW	RW
Initial Value	0xXX								0x00							0	0x0			X	0x0		0	0	0x00						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

	1 = Mask write response
Bit 7:2	**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.

2.9.10 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.

2.9.11 Register GTM_BRIDGE_PTR2

Address Offset:	**see Appendix B**	**Initial Value:** **0x0000_0000**
	31 … 5	4 … 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.

2.9.12 Register GTM_EIRQ_EN

Address Offset:	**see Appendix B**	**Initial Value:** **0x0000_0000**			
	31 … 4	3	2	1	0
Bit	Reserved	AEI_USP_BE_EIRQ_EN	AEI_IM_ADDR_EIRQ_EN	AEI_USP_ADDR_EIRQ_EN	AEI_TO_XPT_EIRQ_EN
Mode	R	RW	RW	RW	RW
Initial Value	0x0000000	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 31:4 **Reserved**
Note: Read as zero, should be written as zero.

2.9.13 Register GTM_TIM[i]_AUX_IN_SRC (i= 0.. n)

Address Offset:	**see Appendix B**																								**Initial Value:**					**0x0000_0000**		
	31	30	29	28	27	26	25	24	23	22	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_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
Initial Value	0x0000 00																								0	0	0	0	0	0	0	0

Bit 0 **SRC_CH0:** Defines AUX_IN source of TIM[i] channel 0 x=0
0 = TOM Output selected TOM[a] channel [b] with a= (i* 8 +x) div 16; b=(i* 8 +x) mod 16;
1 = ATOM Output selected ATOM[i] channel[x]

Bit 1 **SRC_CH1:** Defines AUX_IN source of TIM[i] channel 1 x=1, see bit 0

Bit 2 **SRC_CH2:** Defines AUX_IN source of TIM[i] channel 2 x=2, see bit 0

Bit 3 **SRC_CH3:** Defines AUX_IN source of TIM[i] channel 3 x=3, see bit 0

Bit 4 **SRC_CH4:** Defines AUX_IN source of TIM[i] channel 4 x=4, see bit 0

Bit 5 **SRC_CH5:** Defines AUX_IN source of TIM[i] channel 5 x=5, see bit 0

Bit 6 **SRC_CH6:** Defines AUX_IN source of TIM[i] channel 6 x=6, see bit 0

Bit 7 **SRC_CH7:** Defines AUX_IN source of TIM[i] channel 7 x=7, see bit 0

Bit 31:8	**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 submodules 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.

3.2 Special Data Sources

Besides the addresses of the submodule related data sources as described in Table 21.3, 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 submodules, 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 cancelled 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 submodule that acts as a source at the ARU data loss may occur and no deterministic behaviour is guaranteed.

This is because the regular source submodule 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:

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 Interrupt Signals

The following table describes ARU interrupt signals:

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**.

ACC_ACK_IRQ	ARU access acknowledge IRQ.

3.5 ARU Configuration Registers Overview

The following table shows a conclusion of configuration registers address offsets and initial values.

Register name	**Description**	**Details in Section**
ARU_ACCESS	ARU access register	3.6.1
ARU_DATA_H	ARU access register upper data word	3.6.2
ARU_DATA_L	ARU access register lower data word	3.6.3
ARU_DBG_ACCESS0	Debug access channel 0	3.6.4
ARU_DBG_DATA0_H	Debug access 0 transfer register upper data word	3.6.5
ARU_DBG_DATA0_L	Debug access 0 transfer register lower data word	3.6.6
ARU_DBG_ACCESS1	Debug access channel 0	3.6.7
ARU_DBG_DATA1_H	Debug access 1 transfer register upper data word	3.6.8
ARU_DBG_DATA1_L	Debug access 1 transfer register lower data word	3.6.9
ARU_IRQ_NOTIFY	ARU Interrupt notification register	3.6.10
ARU_IRQ_EN	ARU Interrupt enable register	3.6.11
ARU_IRQ_FORCINT	Register for forcing the *ARU_NEW_DATA*_IRQ interrupt	3.6.12
ARU_IRQ_MODE	IRQ mode configuration register	3.6.13

3.6 ARU Configuration Registers Description

3.6.1 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 are 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 are 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 submodule 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.

3.6.2 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	0000			0x0000 000																												

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

3.6.3 Register ARU_DATA_L

Address Offset:	**see Appendix B**	**Initial Value:**	**0x0000_0000**
	31 30 29	28 … 0	
Bit	Reserved	DATA	
Mode	R	RW	
Initial Value	0x00	0x0000 000	

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

3.6.4 Register ARU_DBG_ACCESS0

Address Offset:	**see Appendix B**	**Initial Value:**	**0x0000_01FE**
	31 … 9	8 … 0	
Bit	Reserved	ADDR	
Mode	R	RW	
Initial Value	0x0000 0	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

3.6.5 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			0x0000 000																												

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

3.6.6 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

3.6.7 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 0																							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

3.6.8 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			0x0000000																												

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

3.6.9 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

3.6.10 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	0x0000 0000																													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

3.6.11 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_EN	NEW_DATA1_IRQ_EN	NEW_DATA0_IRQ_EN
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

3.6.12 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_DATA1	TRG_NEW_DATA0
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

3.6.13 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 00 = Level mode 01 = Pulse mode 10 = Pulse-Notify mode 11 = 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 Broadcast Module (BRC)

4.1 Overview

Since each write address for the submodule 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 submodule Broadcast (BRC) enables to duplicate data streams multiple times.

The BRC submodule 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 submodules 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 submodule are fixed and can be obtained from Chapter 21.
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 21.

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 submodule. 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 submodule 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

Interrupt signals are defined in following table:

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 Registers Overview

Following table shows a conclusion of configuration registers address offsets and initial values.

Register Name	**Description**	**Details in Section**
BRC_SRC_[x]_ADDR	Read address for input channel x (x:0..11)	4.5.1
BRC_SRC_[x]_DEST	Destination channels for input channel x (x: 0..11)	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	Register for forcing the *BRC_DEST_ERR* interrupt	4.5.5
BRC_RST	Software reset	4.5.8
BRC_IRQ_MODE	IRQ mode configuration register	4.5.6

4.5 BRC Configuration Registers Description

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

Address Offset:	**see Appendix B**			**Initial Value:**	**0x0000_01FE**
	31 … 13	12	11 10 9	8 … 0	
Bit	Reserved	BRC_MODE	Reserved	ADDR	
Mode	R	RW	R	RPw	
Initial Value	0x0000 0	0	000	0x1FE	

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

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

4.5.2 Register BRC_SRC_[x]_DEST (x: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_[x]_ADDR** 1 = Destination address 0 mapped to source **BRC_SRC_[x]_ADDR** Note: The destination address 0 for BRC channel is defined in section 21.3
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.

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_DESTx** (x: 0...21) bits are reset to zero) the internal states are reset to their reset value.

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

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 submodule
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_EN
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_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 interrup, see bit 0 for descriptiont.
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 interrup, see bit 0 for descriptiont.
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 interrup, see bit 0 for descriptiont.
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																														XX	

Bit 1:0 **IRQ_MODE**: IRQ mode selection
00 = Level mode
01 = Pulse mode
10 = Pulse-Notify mode
11 = 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_EN
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_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 interrup, see bit 0 for descriptiont.
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 FIFO submodule. The F2A described in chapter 7 and the AFD described in chapter 6 implement the interface part of the FIFO submodule 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 submodules of the GTM and the microcontroller connected to AFD submodule (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 submodule.

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 whether 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 Normal 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 submodule (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 declines the lower watermark or it exceeds the upper watermark, an interrupt signal is triggered by the FIFO submodule if enabled inside the **FIFO[i]_IRQ_EN**.

The interrupt signals are sending to the Interrupt Concentrator Module (ICM) (see chapter 18). The ICM can also initiate specific DMA transfers.

5.2.2 Ring Buffer Operation Mode

The ring buffer mode is a powerful tool to provide a continuous data or configuration stream to the other GTM submodules without CPU interaction. In ring buffer mode the FIFO provides a continuous data stream to the F2A submodule. 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 via the AEI to FIFO interface (AFD) to the FIFO channel while running in ring buffer mode. The new written data will be added in the next ring buffer cycle.

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

There could be the requirement that the user must be able to change some data inside the continuous data stream to the GTM submodules or system bus. This is possible through direct memory access provided by the FIFO AEI interface.

5.3 FIFO Interrupt Signals

Interrupt signals are defined in following table:

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 Registers Overview

The following table shows a conclusion of configuration registers address offsets and initial values:

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

FIFO[i]_CH[x]_IRQ_MODE (x:0…7)	FIFO x IRQ mode control register	5.5.13

5.5 FIFO Configuration Registers Description

5.5.1 Register FIFO[i]_CH[x]_CTRL (x: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	0x0000 000																												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[x]_FILL_LEVEL**, **FIFO[i]_CH[x]_WR_PTR** and **FIFO[i]_CH[x]_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[x]_END_ADDR (x:0...7)

Address Offset:	**see Appendix B**	**Initial Value:**	**0x0000_0XXX**
	31 … 10		9 … 0
Bit	Reserved		ADDR
Mode	R		RW
Initial Value	0x0000 0		0xXXX

Bit 9:0	**ADDR:** End address for FIFO channel x, (x: 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[x]_START_ADDR (x:0...7)

Address Offset:	**see Appendix B**	**Initial Value:**	**0x0000_0XXX**
	31 … 10		9 … 0
Bit	Reserved		ADDR
Mode	R		RW
Initial Value	0x0000 0		0xXXX

Bit 9:0	**ADDR:** Start address for FIFO channel x, (x: 0...7)

Note: Initial value for ADDR is calculated as ADDR = 128*x
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[x]_UPPER_WM (x: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	0x0000 00		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[x]_END_ADDR** – **FIFO[i]_CH[x]_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[x]_LOWER_WM (x: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	0x0000 00																						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[x]_END_ADDR** – **FIFO[i]_CH[x]_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[x]_STATUS (x: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	0x0000 000																												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[x]_END_ADDR** – **FIFO[i]_CH[x]_START_ADDR** + 1

1 = Fill level = **FIFO[i]_CH[x]_END_ADDR** – **FIFO[i]_CH[x]_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[x]_FILL_LEVEL (x:0...7)

Address Offset:	**see Appendix B**	**Initial Value:**	**0x0000_0000**
	31 … 11		10 … 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[x]_END_ADDR** – **FIFO[i]_CH[x]_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[x]_WR_PTR (x: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	0x0000 0																						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[x]_START_ADDR**

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

5.5.9 Register FIFO[i]_CH[x]_RD_PTR (x: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[x]_START_ADDR**

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

5.5.10 Register FIFO[i]_CH[x]_IRQ_NOTIFY (x: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_UWM	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 over-run. See bit 0.
Bit 31:4	**Reserved:** reserved Note: read as zero, should be written as zero

5.5.11 Register FIFO[i]_CH[x]_IRQ_EN (x: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_EN	FIFO_LWM_IRQ_EN	FIFO_FULL_IRQ_EN	FIFO_EMPTY_IRQ_EN
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[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_FIFO_UWM	TRG_FIFO_LWM	TRG_FIFO_FULL	TRG_FIFO_EMPTY
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[i]_IRQ_NOTIFY** register

Note: This bit is cleared automatically after write.
Note: This bit is cleared automatically after write.
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[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																												DMA_HYST_DIR	DMA_HYSTERESIS	IRQ_MODE	
Mode	R																												RW	RW	RW	
Initial Value	0x0000000																												0	0	XX	

Bit 1:0 **IRQ_MODE**: IRQ mode selection
00 = Level mode
01 = Pulse mode
10 = Pulse-Notify mode
11 = 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[x]_EIRQ_EN (x: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_EN	FIFO_LWM_EIRQ_EN	FIFO_FULL_EIRQ_EN	FIFO_EMPTY_EIRQ_EN
Mode	R																												RW	RW	RW	RW
Initial Value	0x0000000																												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 submodule implements a data interface between the AEI bus and the FIFO submodule, which consists of eight logical FIFO channels.

The AFD submodule 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

Following table shows a conclusion of configuration registers address offsets and initial values.

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

6.3 AFD Register description

6.3.1 Register AFD[i]_CH[x]_BUF_ACC (x: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 submodule. 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

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 than 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.

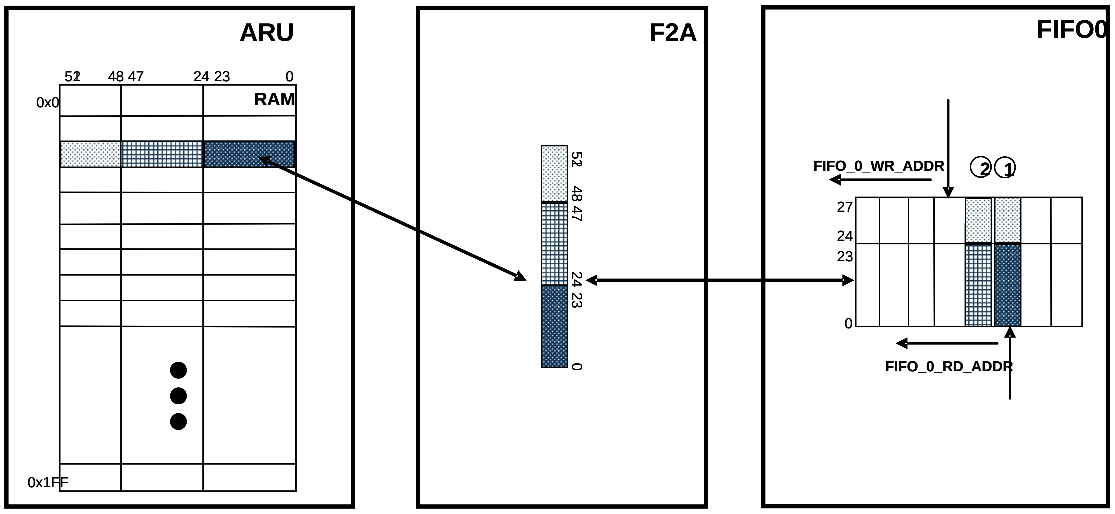
When reading from the ARU the F2A first writes the lower word to the FIFO.
In case of writing to the ARU the F2A reads the lower word first from the FIFO, thus the lower word must be written first to the FIFO through the AFD interface.

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 F2A Configuration Registers Overview

The following table shows a conclusion of configuration registers address offsets and initial values.

Register name	Description	Details in Section

F2A[i]_ENABLE	F2A stream activation register	7.4.1
F2A[i]_CH[x]_ARU_RD_FIFO	F2A read channel address register (x: 0..7)	7.4.2
F2A[i]_CH[x]_STR_CFG	F2A stream x configuration register (x: 0..7)	7.4.3

7.4 F2A Configuration Registers description

7.4.1 Register F2A[i]_ENABLE

Address Offset:	**see Appendix B**	**Initial Value:**	**0x0000_0000**						
	31 … 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 of following double bit values is possible : 00 = Don't care, bits 1:0 will not be changed 01 =Stream 0 is disabled and internal states are reset 10 = Stream 0 is enabled 11 = Don't care, bits 1:0 will not be changed Read of following double values means : 00 = Stream disabled 11 = Stream enabled
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

7.4.2 Register F2A[i]_CH[x]_ARU_RD_FIFO (x: 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	0x00000																							0x1FE								

Bit 8:0	**ADDR**: ARU Read address Note: this bit field is only writeable if channel is disabled.
Bit 31:9	**Reserved** Note: Read as zero, should be written as zero

7.4.3 Register F2A[i]_CH[x]_STR_CFG (x: 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 00 = Transfer low word (ARU bits 23:0) from/to FIFO 01 = Transfer high word (ARU bits 47:24) from/to FIFO 10 = Transfer both words from/to FIFO 11 = 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.

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 subunits that generate different clock sources for the whole GTM-IP. Figure 8.1.1 shows a block diagram of the CMU.

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

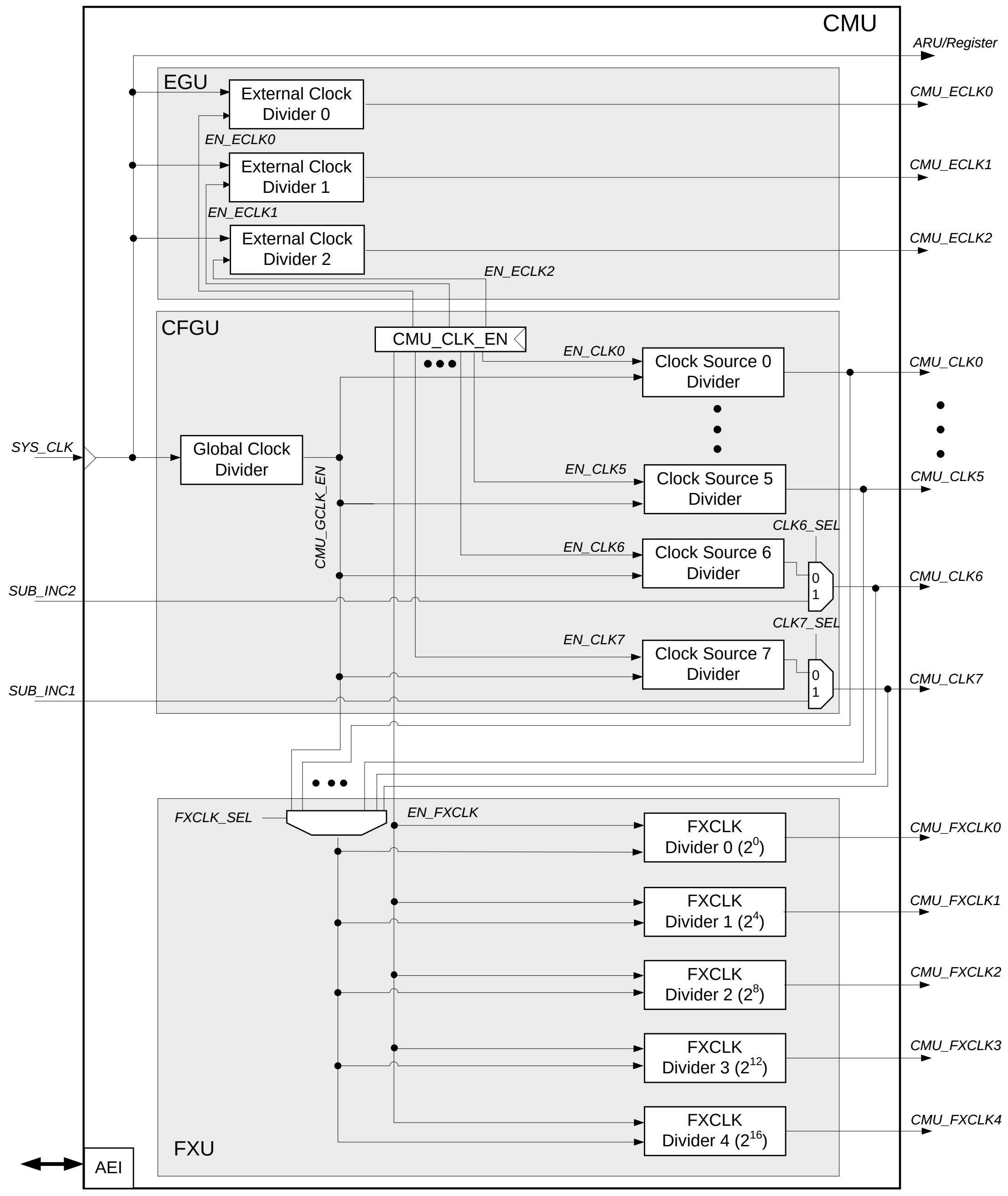
The Fixed Clock Generation (FXU) subunit generates predefined non-configurable clocks *CMU_FXCLK[y]* (y: 0..4) for the TOM submodules and the MON submodule. The *CMU_FXCLK[y]* signals are derived from the *CMU_GCLK_EN* signal generated by the Global Clock Divider. The dividing factors are defined as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

The External Clock Generation (EGU) subunit 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 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 *SYS_CLK* signal.

The four configurable clock signals *CMU_CLK0*, *CMU_CLK1, CMU_CLK6* and *CMU_CLK7* are connected to 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 GTM-IP global input clock signal *SYS_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 *SYS_CLK* signal to generate the user specified divided clock frequency.

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

$T_{CMU\_GCLK\_EN}=(Z/N)*T_{SYS\_CLK}$
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 *SYS_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 Subunit (CFGU)

The CMU subunit 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 8.4

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 unset.

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

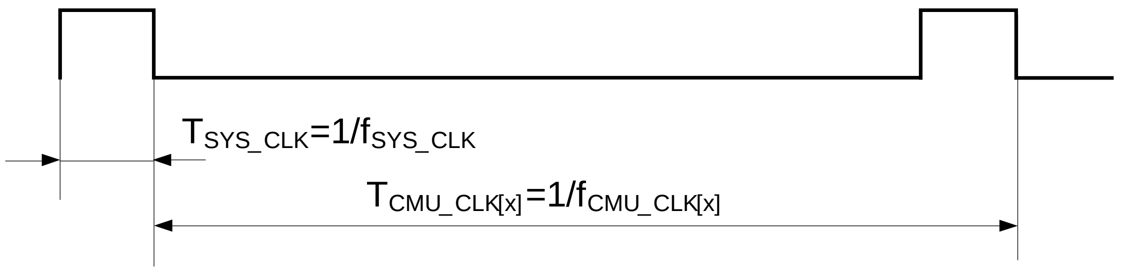
To avoid unexpected behaviour of the hardware, the configuration of a register **CMU_CLK_[x]_CTRL** can only be changed, when the corresponding clock signal *CMU_CLK[x]* is disabled.

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 *SYS_CLK* signal.

The hardware guarantees that all clock signals *CMU_CLK[x],* 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.

8.4 Wave Form of Generated Clock Signal CMU_CLK[x]


$T_{SYS\_CLK}=1/f_{SYS\_CLK}$
$T_{CMU\_CLK[x]}=1/f_{CMU\_CLK[x]}$

8.5 Fixed Clock Generation (FXU)

The FXU subunit 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 submodules.

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 8.4)

8.6 External Generation Unit (EGU)

The EGU subunit 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 *SYS_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.

Each of the external clocks 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:

$$T_{CMU\_ECLK[z]}=2*(\mathbf{ECLK[z]\_NUM/ECLK[z]\_DEN})*T_{SYS\_CLK}$$

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 *SYS_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.

8.7 CMU Configuration Registers Overview

Following configuration registers are considered in CMU submodule:

Register Name	Description	Details in Section
CMU_CLK_EN	Clock enable	8.8.1
CMU_GCLK_NUM	Global clock control numerator	8.8.2
CMU_GCLK_DEN	Global clock control denominator	8.8.3
CMU_CLK_0_CTRL	Control for clock source 0	8.8.4
CMU_CLK_1_CTRL	Control for clock source 1	8.8.4
CMU_CLK_2_CTRL	Control for clock source 2	8.8.4
CMU_CLK_3_CTRL	Control for clock source 3	8.8.4
CMU_CLK_4_CTRL	Control for clock source 4	8.8.4
CMU_CLK_5_CTRL	Control for clock source 5	8.8.4
CMU_CLK_6_CTRL	Control for clock source 6	8.8.5
CMU_CLK_7_CTRL	Control for clock source 7	8.8.6
CMU_ECLK_0_NUM	External clock 0 control numerator	8.8.7
CMU_ECLK_0_DEN	External clock 0 control denominator	8.8.8
CMU_ECLK_1_NUM	External clock 1 control numerator	8.8.7
CMU_ECLK_1_DEN	External clock 1 control denominator	8.8.8
CMU_ECLK_2_NUM	External clock 2 control numerator	8.8.7
CMU_ECLK_2_DEN	External clock 2 control denominator	8.8.8
CMU_FXCLK_CTRL	Control FXCLK subunit input clock	8.8.9

8.8 CMU Configuration Register Description

8.8.1 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
00 = clock source is disabled (ignore write access)
01 = disable clock signal and reset internal states
10 = enable clock signal
11 = clock signal enabled (ignore write access)

Note: Any read access to an **EN_CLK[x]**, **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 00 and 11 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 subunit, see bits 1:0
Bit 19:18 **EN_ECLK1:** Enable ECLK 1 generation subunit, see bits 1:0
Bit 21:20 **EN_ECLK2:** Enable ECLK 2 generation subunit, 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

8.8.2 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								0x0000 1																							

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

8.8.3 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								0x0000 01																							

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

8.8.4 Register CMU_CLK_[x]_CTRL (x: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								CLK_CNT																							
Mode	R								RPw																							
Initial Value	0x00								0x000000																							

Bit 23:0 **CLK_CNT:** Clock count. Defines count value for the clock divider of clock source CMU_CLK[x] (x: 0...5).
Note: Value can only be modified when clock enable **EN_CLK[x]** (x:0...5) is disabled.

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

8.8.5 Register CMU_CLK_6_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							CLK6_SEL	CLK_CNT																							
Mode	R							RPw	RPw																							
Initial Value	0x00							0	0x0000 00																							

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** is disabled

Bit 24 **CLK6_SEL:** Clock source selection for *CMU_CLK6*.
0 = use Clock Source 6 Divider
1 = use signal *SUB_INC2* of submodule DPLL
Note: Value can only be modified when clock enable **EN_CLK6** is disabled.

Bit 31:25 **Reserved:** Reserved bits
Note: Read as zero, should be written as zero

8.8.6 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							0	0x0000 00																							

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** is disabled

Bit 24 **CLK7_SEL:** Clock source selection for *CMU_CLK7*.

0 = use Clock Source 7 Divider
1 = use signal *SUB_INC1* of submodule DPLL
Note: Value can only be modified when clock enable **EN_CLK7** is disabled.

Bit 31:25	**Reserved:** Reserved bits Note: Read as zero, should be written as zero

8.8.7 Register CMU_ECLK_[z]_NUM (z:0...2)

Address Offset:	**see Appendix B**	**Initial Value:** 0x0000_0001
	31 ... 24	23 ... 0
Bit	Reserved	ECLK_NUM
Mode	R	RPw
Initial Value	0x00	0x0000 1

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

8.8.8 Register CMU_ECLK_[z]_DEN (z:0...2)

Address Offset:	**see Appendix B**	**Initial Value:**	**0x0000_0001**
	31 … 24	23 … 0	
Bit	Reserved	ECLK_DEN	
Mode	R	RPw	
Initial Value	0x00	0x0000 01	

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

8.8.9 Register CMU_FXCLK_CTRL

Address Offset:	**see Appendix B**	**Initial Value:**	**0x0000_0000**
	31 … 4	3 … 0	
Bit	Reserved	FXCLK_SEL	
Mode	R	RPw	
Initial Value	0x0000 000	0x0	

Bit 3:0 **FXCLK_SEL:** Input clock selection for *EN_FXCLK* line.
0000 = *CMU_GCLK_EN* selected.
0001 = *CMU_CLK0* selected.
0010 = *CMU_CLK1* selected.

	0011 = *CMU_CLK2* selected. 0100 = *CMU_CLK3* selected. 0101 = *CMU_CLK4* selected. 0110 = *CMU_CLK5* selected. 0111 = *CMU_CLK6* selected. 1000 = *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

9 Time Base Unit (TBU)

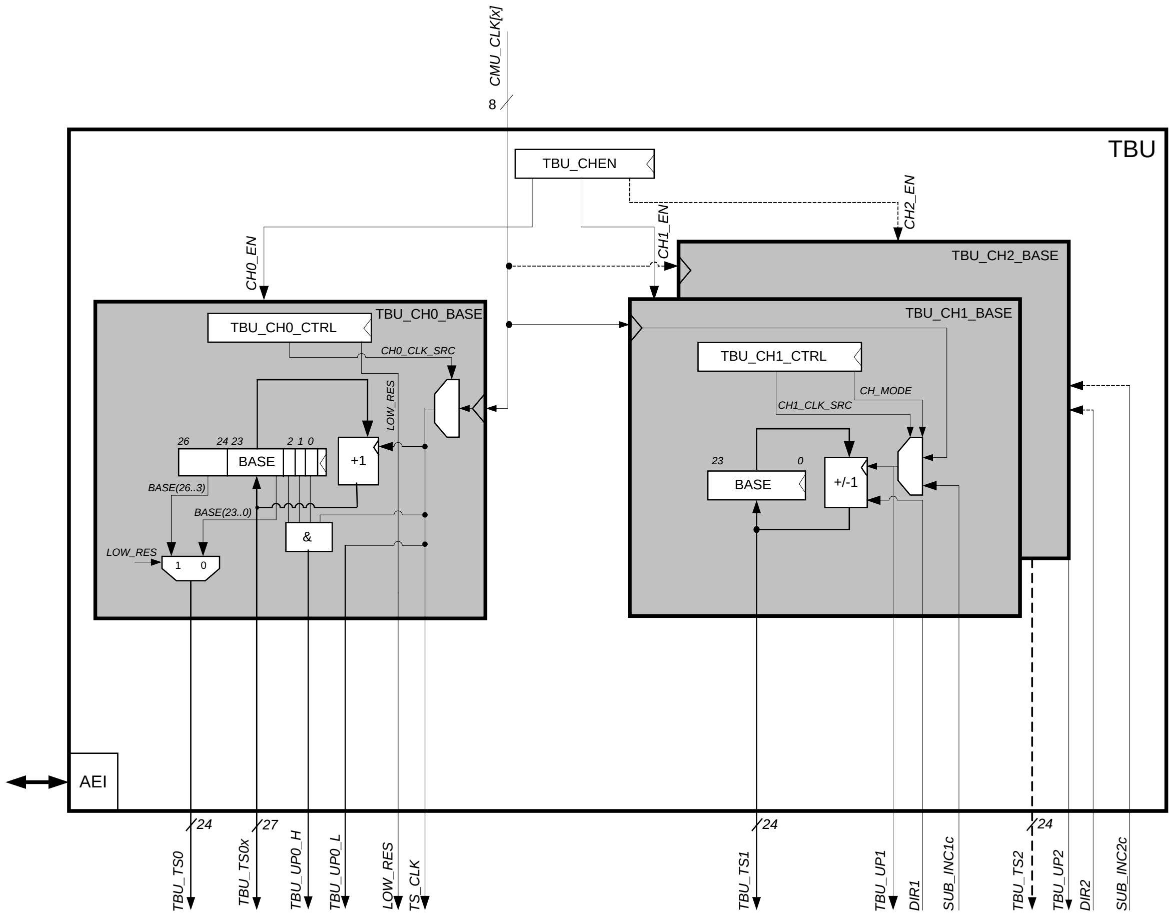
9.1 Overview

The Time Base Unit TBU provides common time bases for the GTM-IP. The TBU submodule is organized in channels, where the number of channels is device dependent. There are at most three channels implemented inside the TBU. The TBU channel 0 time base register **TBU_CH0_BASE** is 27 bits 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]* are provided to subsequent submodules of the GTM.

The *TBU_UP[z]* (z: 1..2) signals are set to high for a single SYS_CLK period, whenever the corresponding signal *TBU_TS[z]* (z: 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 if 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 9.1.1 shows a block diagram of the Time Base Unit.

9.1.1 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_BASE_[z] is done via the AEI interface. Each TBU channel may select one of the eight *CMU_CLK[x]* (x: 0..7) signals coming from the CMU submodule.

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_BASE_[y]. This clock in combination with the *DIR[y]* signals determines the counter direction of the TBU_BASE_[y].

The selected time stamp clock signal for the TBU_BASE_0 subunit is served via the *TS_CLK* signal line to the DPLL submodule. The *TS_CLK* signal equals the signal *TBU_UP0*.

9.2 TBU Time Base Channels

The time base values are generated within the TBU time base channels in two independent operation modes.

9.2.1 TBU Channel Modes

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 submodules.

TBU channel 1 and channel 2 can run in two modes; the free running counter mode and forward/backward counter mode, where the time base can run backwards dependent on the *DIR[y]* input signal values.

In both modes, the time base register **TBU_CH[z]_BASE** can be initialized with a start value just before enabling the corresponding TBU channel.

Moreover, the time base register **TBU_CH[z]_BASE** can always be read in order to determine the actual value of the counter.

9.2.1.1 Free Running Counter Mode

In TBU Free running counter mode, the time base register **TBU_CH[y]_BASE** is updated on every specified incoming clock event by the selected signal *CMU_CLK[x]* (dependent on **TBU_CH[z]_CTRL** register). In general the time base register **TBU_CH[y]_BASE** is incremented on every *CMU_CLK[x]* clock tick.

9.2.1.2 Forward/Backward Counter Mode

As mentioned above TBU channels 1 and 2 can also be configured to run in Forward/Backward Counter Mode. 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’.

9.3 TBU Configuration Registers Overview

Following table shows a conclusion of configuration registers address offsets and initial values.

Register Name	Description	Details in Section

TBU_CHEN	TBU global channel enable	9.4.1
TBU_CH0_CTRL	TBU channel 0 control	9.4.2
TBU_CH0_BASE	TBU channel 0 base	9.4.3
TBU_CH1_CTRL	TBU channel 1 control	9.4.4
TBU_CH1_BASE	TBU channel 1 base	9.4.5
TBU_CH2_CTRL	TBU channel 2 control	9.4.4
TBU_CH2_BASE	TBU channel 2 base	9.4.5

Note: In a typical application the Time Base Unit (TBU) considers channels 0 and 1 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.

9.4 TBU Registers description

9.4.1 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_CH2		ENDIS_CH1		ENDIS_CH0	
Mode	R																										RW		RW		RW	
Initial Value	0x0000 00																										00		00		00	

Bit 1:0 **ENDIS_CH0:** TBU channel 0 enable/disable control.
Write of following double bit values is possible:
00 = don't care, bits 1:0 will not be changed
01 = channel disabled: is read as 00 (see below)
10 = channel enabled: is read as 11 (see below)
11 = don't care, bits 1:0 will not be changed
Note: Read of following double values means:
00 = channel disabled

	11 = 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 Note: These bits are only applicable if channel is implemented for this device, otherwise read and write as zero
Bit 31:6	**Reserved:** Reserved Note: Read as zero should be written as zero

9.4.2 Register TBU_CH0_CTRL

Address Offset:	**see Appendix B**			**Initial Value:**	**0x0000_0000**
	31 … 4	3	2	1	0
Bit	Reserved	CH_CLK_SRC			LOW_RES
Mode	R	RPw			RPw
Initial Value	0x0000 000	000			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 submodules. 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 000 = *CMU_CLK0* selected 001 = *CMU_CLK1* selected 010 = *CMU_CLK2* selected 011 = *CMU_CLK3* selected 100 = *CMU_CLK4* selected 101 = *CMU_CLK5* selected 110 = *CMU_CLK6* selected 111 = *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

9.4.3 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					0x000000																										

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

9.4.4 Register TBU_CH[y]_CTRL (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																												CH_CLK_SRC			CH_MODE
Mode	R																												RPw			RPw
Initial Value	0x0000000																												000			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 y (y:1,2) 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_INC[y]c* clock signal in combination with the *DIR[y]* input signal is used to determine the counter direction and clock frequency.

Bit 3:1	**CH_CLK_SRC:** Clock source for channel x (x: 1...2) time base counter 000 = *CMU_CLK0* selected 001 = *CMU_CLK1* selected 010 = *CMU_CLK2* selected 011 = *CMU_CLK3* selected 100 = *CMU_CLK4* selected 101 = *CMU_CLK5* selected 110 = *CMU_CLK6* selected 111 = *CMU_CLK7* selected Note: This value can only be modified if channel y was disabled
Bit 31:4	**Reserved:** Reserved Note: Read as zero should be written as zero

9.4.5 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 Timer Input Module (TIM)

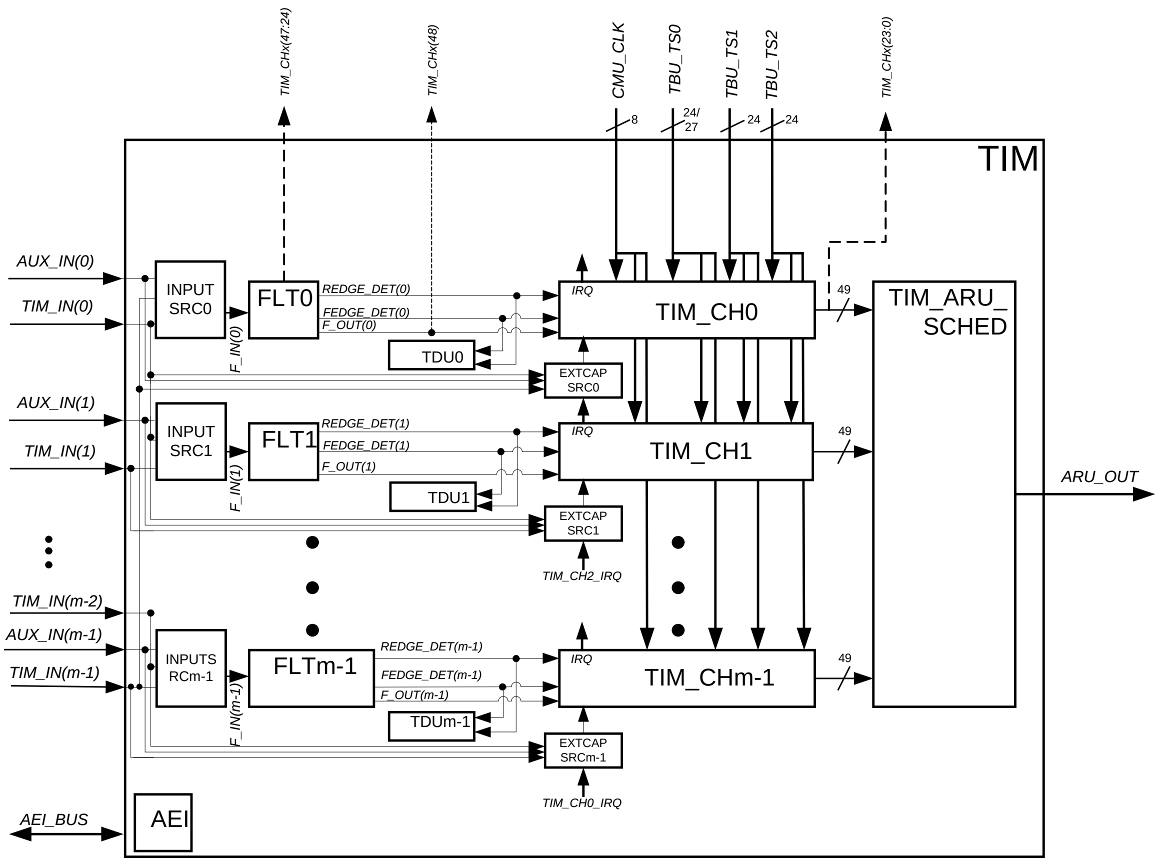
10.1 Overview

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 durations for a whole PWM period.

The architecture of TIM is shown in Figure 10.1.1.

10.1.1 TIM Block Diagram



The number of channels *m* inside a TIM submodule depends on the device.
Each of the *m* dedicated input signals are filtered inside the FLTx subunit 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 submodule TIM provides different filter mechanisms described in more detail in Chapter 10.2. After filtering, the signal is routed to the corresponding TIM channel.

The measurement values can be read by the CPU directly via the AEI-Bus or they can be routed through the ARU to other submodules of the GTM.

For timeout detection of an incoming signal (no subsequent edge detected during a specified duration) each individual channel has a Timeout Detection Unit (TDU).

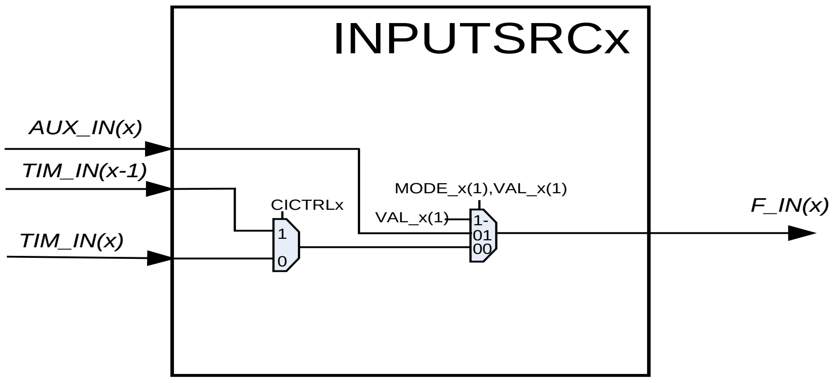
For the GTM-IP TIM0 submodule only, the dashed signal outputs *TIM[i]_CH[x](23:0)*, *TIM[i]_CH[x](47:24)* und *TIM[i]_CH[x](48)* come from the TIM0 submodule channels zero (0) to five (5) and are connected to MAP submodule. There, they are used for further processing and for routing to the DPLL.

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.

10.1.2 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]_CH[x]_IN_SRC** which source is in use.

10.1.2.1 INPUTSRC Block Diagram



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]_CH[x]_IN_SRC** a synchronous control of all 8 input channels is possible.

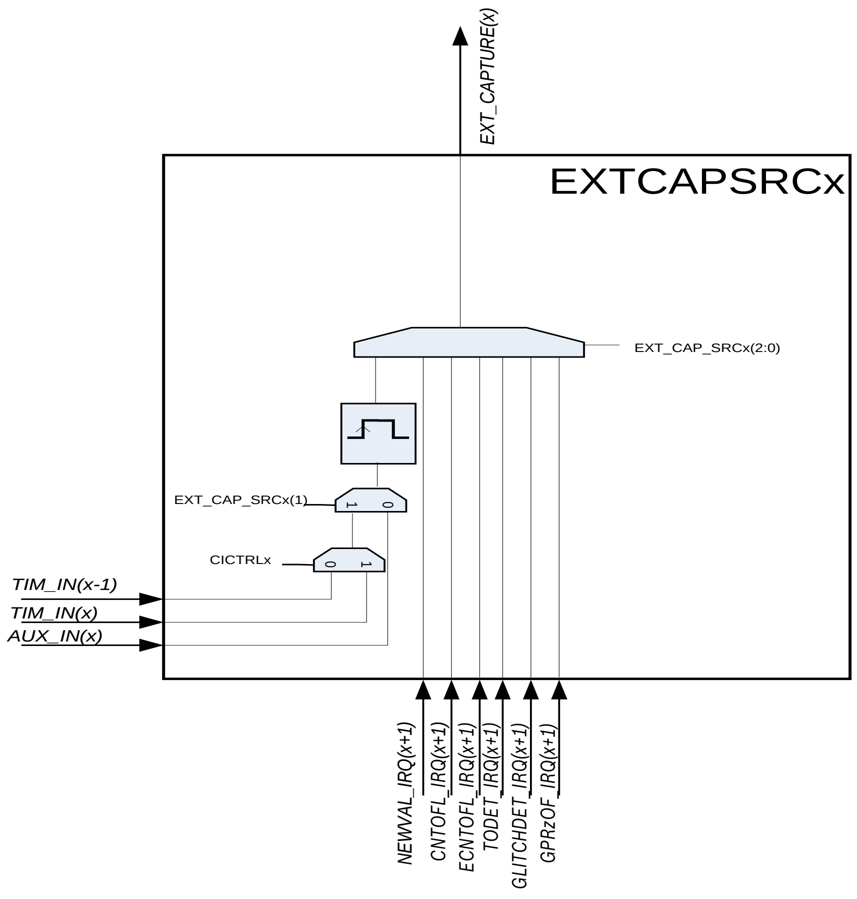
Two adjacent channels can be combined by setting the **CICTRL** bit field in the corresponding **TIM[i]_CH[x]_CTRL** register. This allows for a combination of complex measurements on one input signal with two TIM channels.

The additional input signal **AUX_IN[x]** can be selected as an input signal. The source of this signal is defined in the chapter 2.1.3.

10.1.3 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 .

10.1.3.1 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**.

10.2 TIM Filter Functionality (FLT)

10.2.1 Overview

The TIM submodule provides a configurable filter mechanism for each input signal. These filter mechanism is provided inside the FLT subunit.

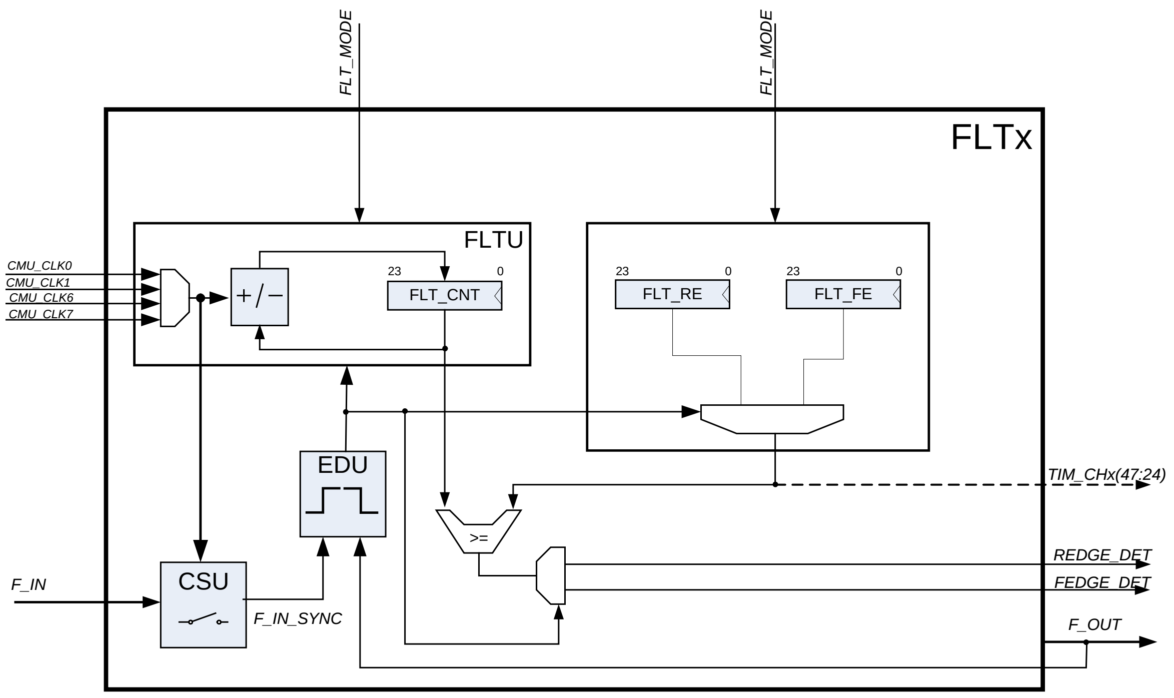
FLT architecture is shown in Figure 10.2.1.1.
The filter includes a clock synchronisation unit (CSU), an edge detection unit (EDU), and a filter counter associated to the filter unit (FLTU).

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 less than the selected CMU clock period are 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).

10.2.1.1 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 10.2.1.2 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 submodule for further processing (please see ej cr vgt"15).

10.2.1.2 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

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 submodule.

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.

10.2.2 TIM Filter Modes

10.2.2.1 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.

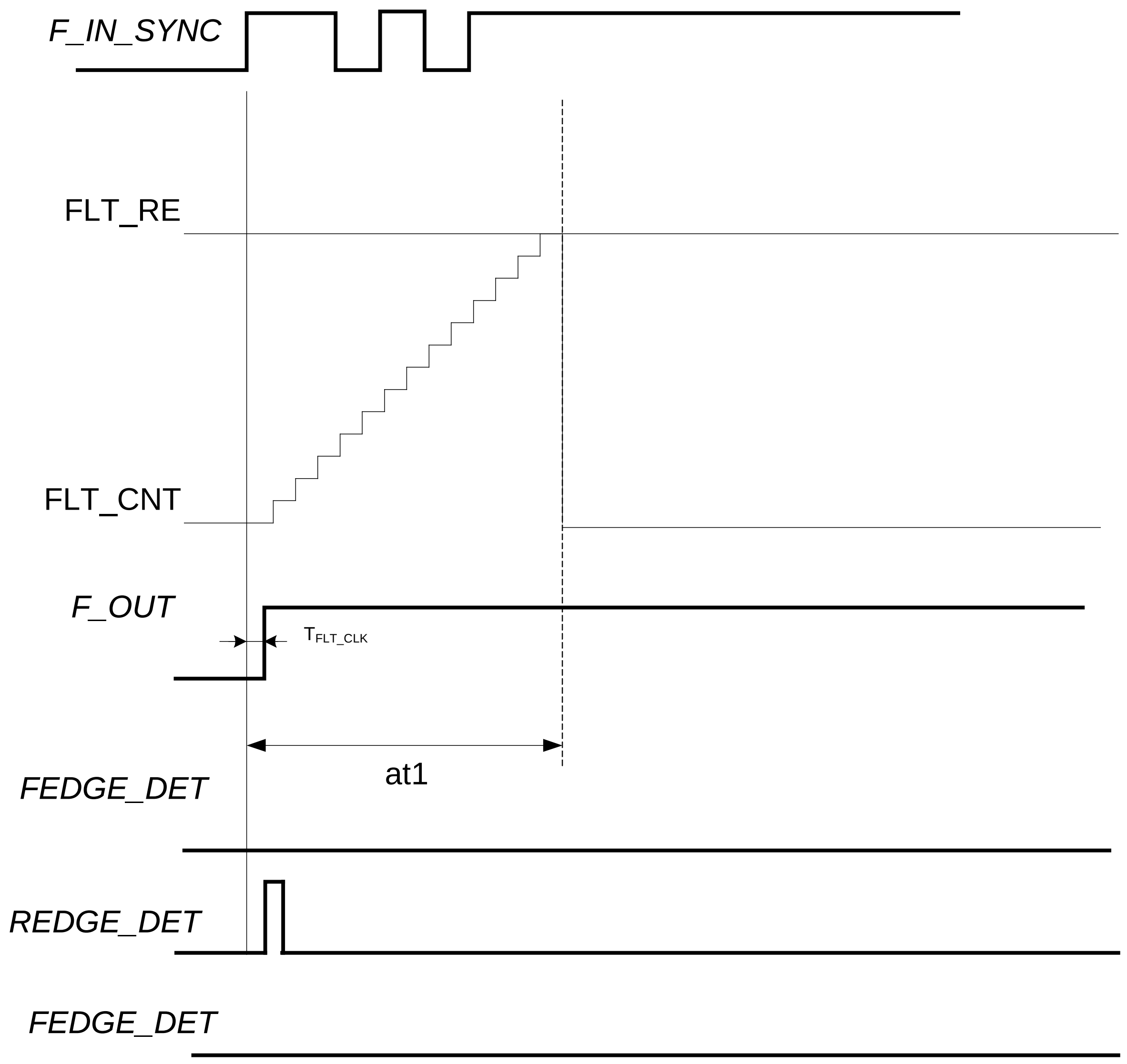
For each edge type the acceptance time can be specified separately in the **FLT_RE** and **FLT_FE** registers.
Each signal change on the input *F_IN_SYNC* during the duration of the acceptance time has no effect on the output signal level *F_OUT* of the filter but it sets the glitch **GLITCHDET** bit 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 10.2.2.1.2).

Figure 10.2.2.1.1 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**.

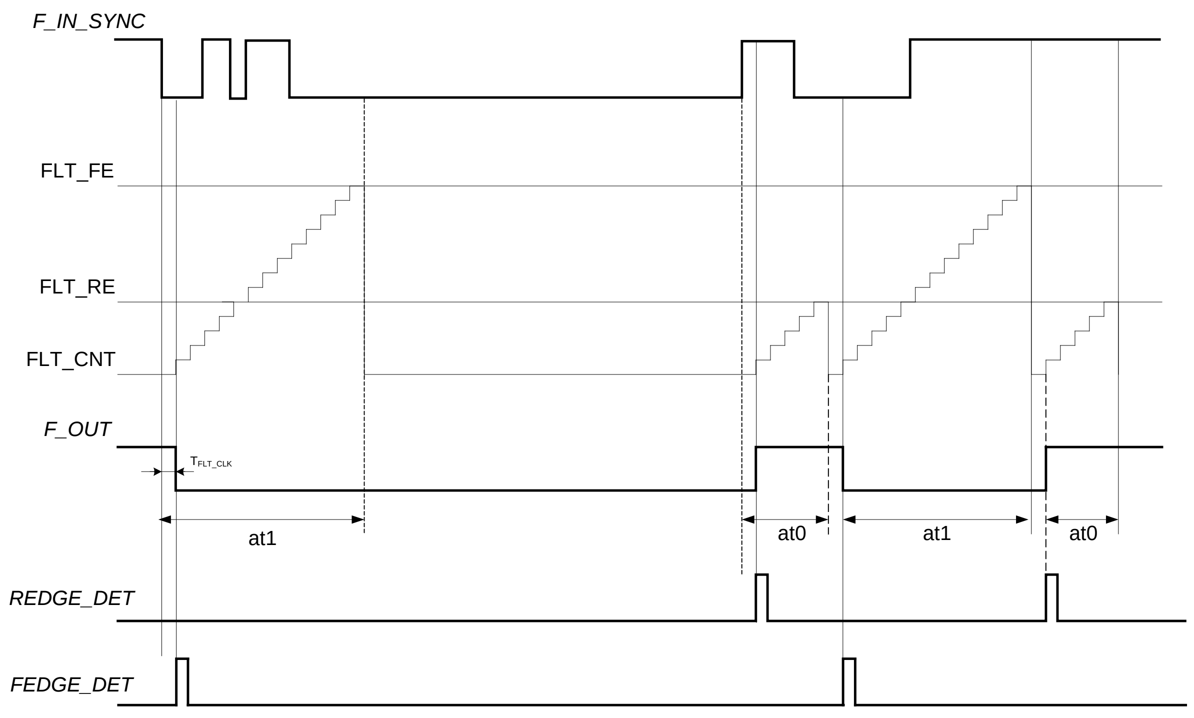
10.2.2.1.1 Immediate Edge Propagation Mode in the case of a rising edge



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 10.2.2.1.2 shows a more complex example of the TIM filter, in which both, rising and falling edges are configured in immediate edge propagation mode.

10.2.2.1.2 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 10.2.2.1.2 shows this scenario at the detection of the first rising edge and the second falling edge.

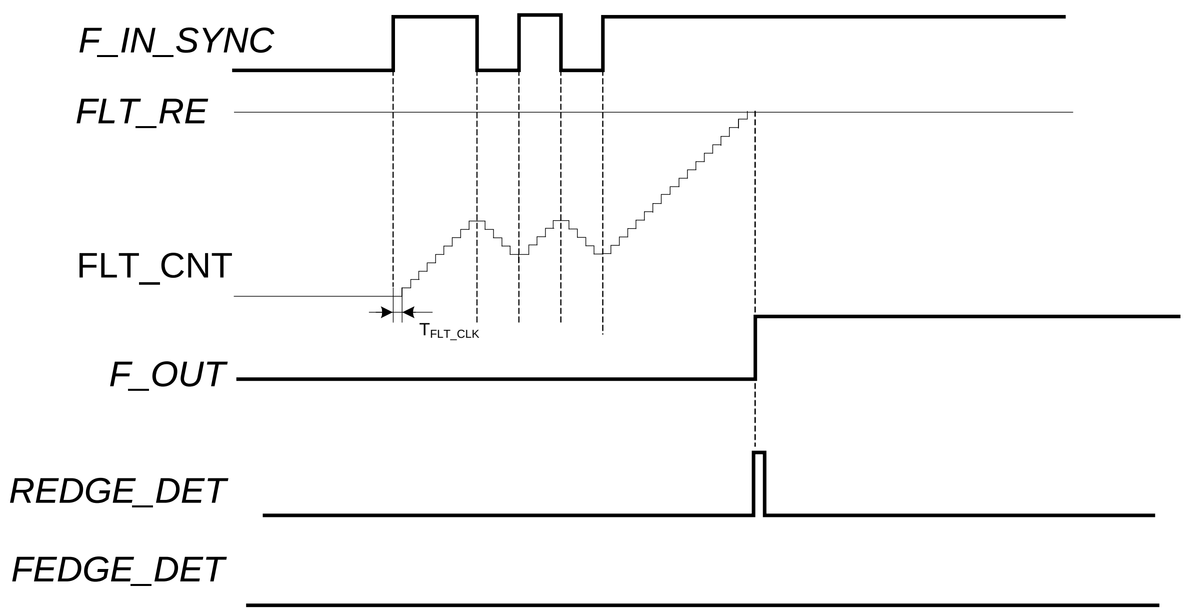
10.2.2.2 Individual De-Glitch Time Mode (up/down counter)

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 10.2.2.2.1 shows the behaviour of the filter in individual de-glitch time (up/down counter) mode in the case of the rising edge detection.

10.2.2.2.1 Individual De-Glitch Time Mode (up/down counter) in the case of a rising edge



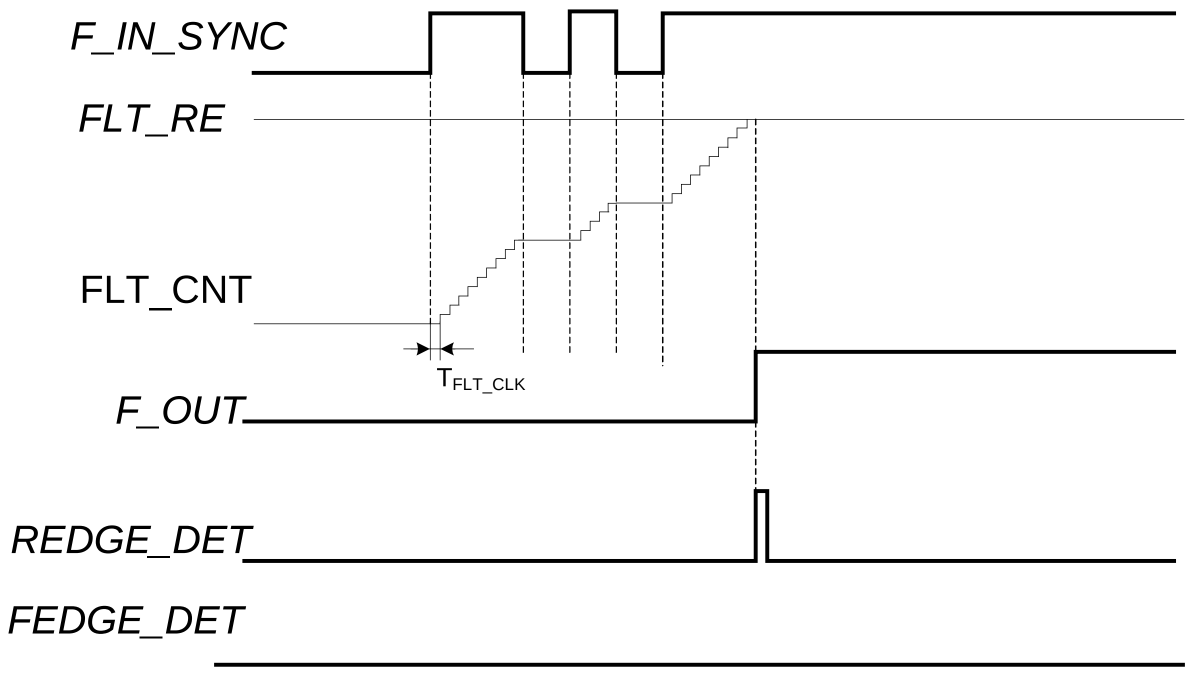
10.2.2.3 Individual De-Glitch Time Mode (hold counter)

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 10.2.2.3.1 shows the behaviour of the filter in individual de-glitch time (hold counter) mode in the case of the rising edge detection.

10.2.2.3.1 Individual De-Glitch Time Mode (hold counter) in the case of a rising edge



10.2.2.4 Immediate Edge Propagation and Individual De-Glitch Mode

As already mentioned, the three different filter modes can be applied individually to each edge of the measured signal.

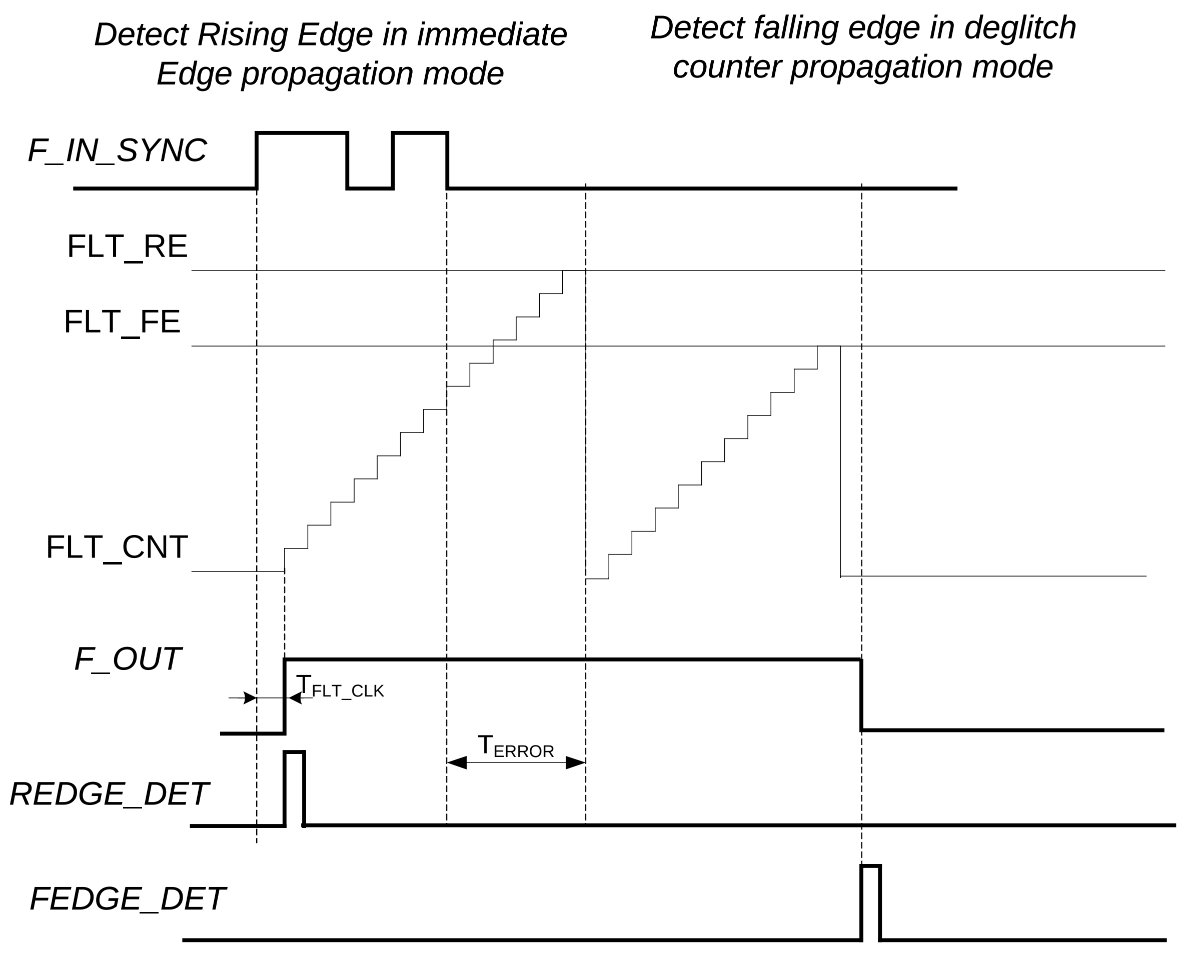
However, if one edge is configured with immediate edge propagation and the other edge with an individual deglitch mode (whether up/down counter or hold counter) a special consideration has to be applied.

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 10.2.2.4.1.

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 can not measure the time T_{ERROR}, as shown in Figure 10.2.2.4.1.

10.2.2.4.1 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
T_FLT_CLK
T_ERROR
REDGE_DET
FEDGE_DET

10.2.3 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 occuring 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** or **FLT_CTR_FE** will take place immediately.

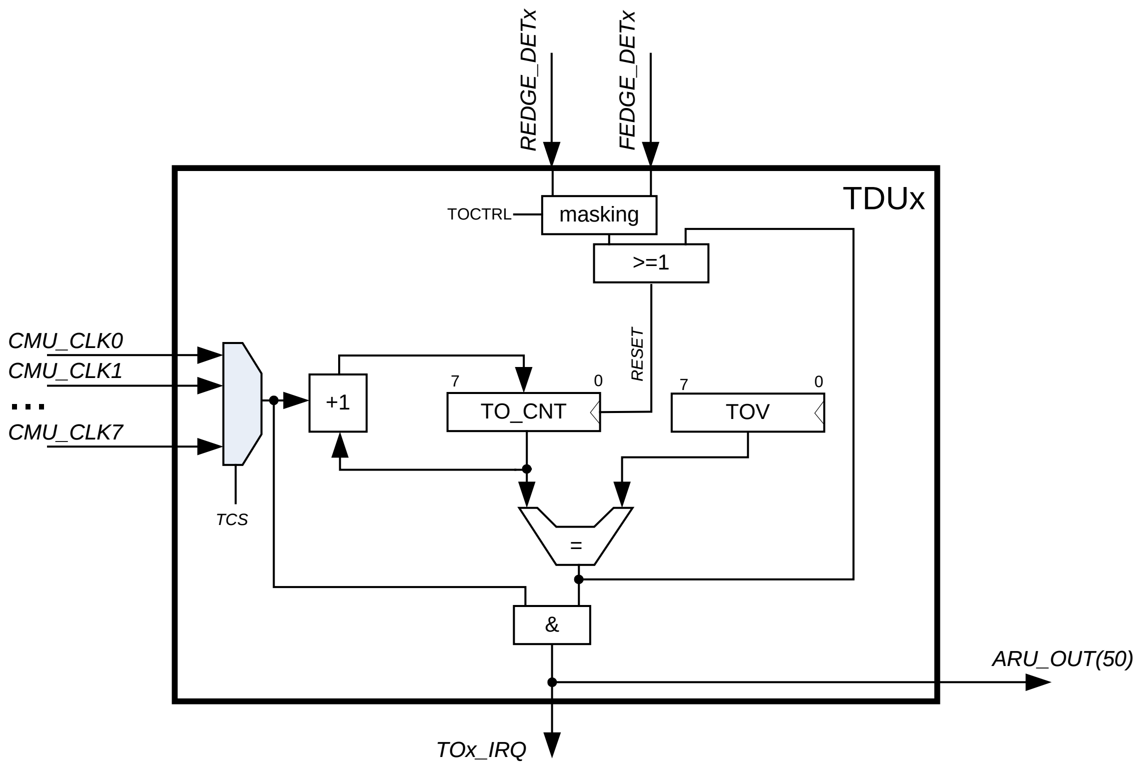
10.3 Timeout Detection Unit (TDU)

The Timeout Detection Unit (TDU) is responsible for timeout detection of the TIM input signals.

Each channel of the TIM submodule has its own Timeout Detection Unit (TDU) where a timeout event can be set up on the filtered input signal of the corresponding channel.

The TDU architecture is shown in 10.3.1.

10.3.1 Architecture of the TDU Subunit



It is possible to detect timeouts with the resolution of the specified *CMU_CLKx* input signal selected with the bit field **TCS** of the register **TIM[i]_CH[x]_TDUV**. The individual timeout values have to be specified in number of ticks of the selected input clock signal and have to be specified in the field **TOV** of timeout value register **TIM[i]_CH[x]_TDUV** of the TIM channel x.

The exact time out value T_{TDU} can be calculated with:

T_{TDU}=(**TOV**+1)*T_{CMU_CLKx},

whereas T_{CMU_CLKx} is the clock period of the selected CMU clock signal.

Timeout detection can be enabled or disabled individually 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 counter **TO_CNT** is reset by each detected valid input edge coming either from the filtered input signal or when the timeout value **TOV** is reached by the counter **TO_CNT**.

After such a reset or by enabling the channel inside the **TIM[i]_CH[x]_CTRL** register the counter **TO_CNT** starts counting again from value 0 with the specified clock input signal.

Otherwise, timeout measurements starts immediately after the **TOCTRL** bit inside the **TIM[i]_CH[x]_CTRL** register is written (enabled).

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 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.

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 valide 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 occured.

When a timeout occured 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:

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

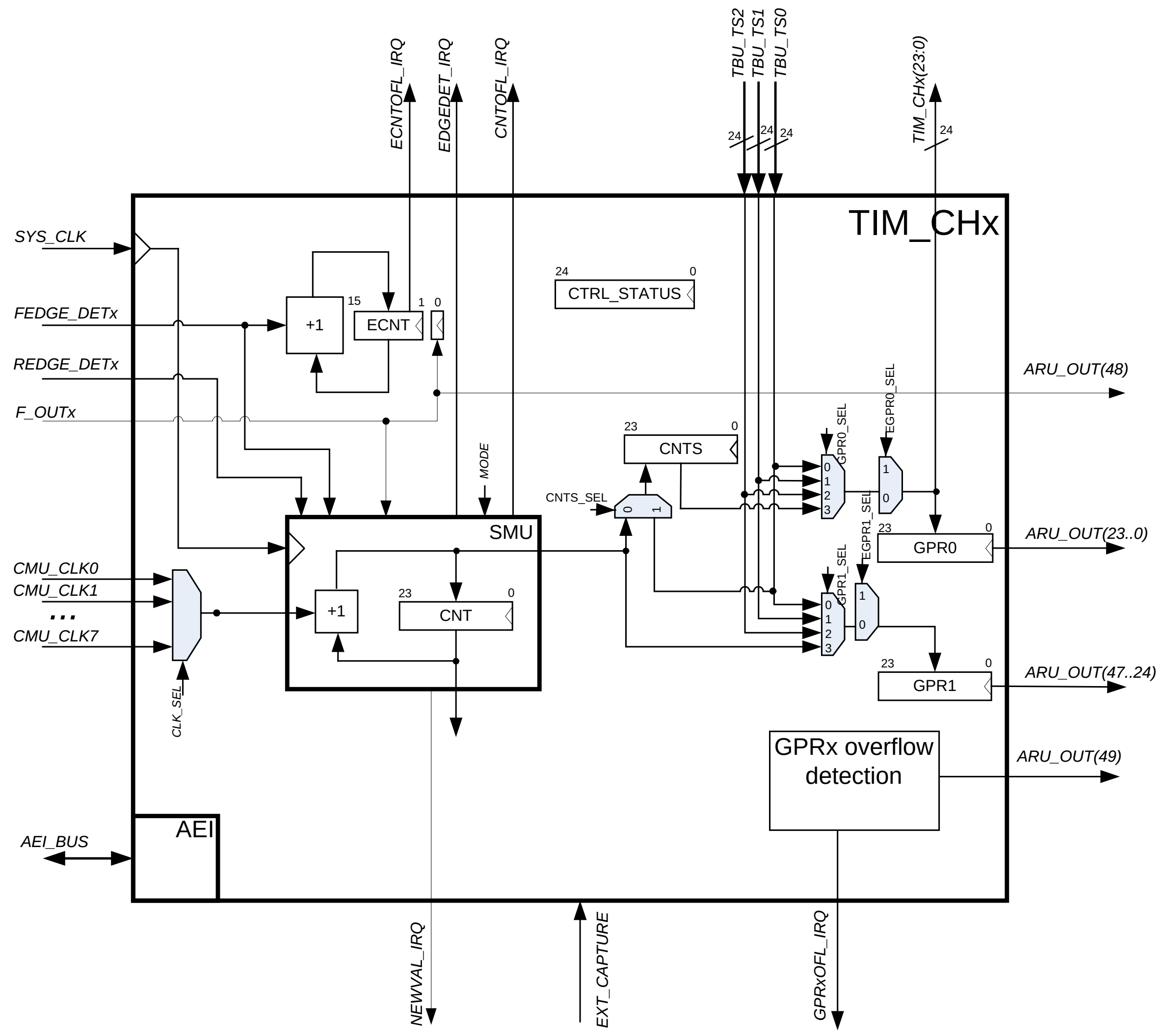
10.4 TIM Channel Architecture

10.4.1 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 subunit of each individual channel for timeout measurement. The architecture of the TIM channel is depicted in Figure 10.4.1.1.

10.4.1.1 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 submodules. 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 submodule 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.

10.4.2 TIM Channel Modes

The TIM provides six 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, signalling 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 dependant) 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.**

10.4.2.1 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), the channel starts measuring after the first rising edge is detected by the filter.

Measurement is done with the **CNT** register counting with the configured clock coming from **CMU_CLKx** until a falling edge is detected.

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 **GPR0_SEL**=11 and **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 (first edge defined by DSL is detected). Afterwards each edge leaving the level defined by DSL is perfoming a CNTS register update.

If a PWM with a low level duty cycle should be measured (**DSL** = 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.

10.4.2.1.1 External capture TIM PWM Measurement Mode (TPWM)

If external capture is enabled, the pwm measurement is done continuously. The actual measurement values are captured to GPRx if an external capture events occurs.

Operation is done depending on cmu clock, **ISL, DSL** bit and the input signal value defined in next table:

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.

10.4.2.2 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.

The pulse times are measured by incrementing the TIM channel counter **CNT** whenever the pulse has the specified signal level **DSL**. The counter is stopped whenever the input signal has the opposite signal level.

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.

Whenever the counter is stopped, the registers **CNTS**, **GPR0** and **GPR1** are updated according to settings of its corresponding input multiplexers, using the bits **EGPR0_SE**L, **EGPR1_SE**L, **GPR0_SE**L, **GPR1_SE**L, 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**.

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 submodules attached to the ARU.

10.4.2.2.1 External capture TIM Pulse Integration Mode (TPIM)

If external capture is enabled, the pulse integration is done until next external capture event occurs.
Operation is done depending on cmu clock, **DSL** bit and the input signal value defined in next table:

Input signal F_OUTx	**selected CMU Clock**	**External capture**	**ISL**	**DSL**	**Action description**
0	1	0	-	0	CNT++
1	1	0	-	0	no
1	1	0	-	1	CNT++
0	1	0	-	1	no
-	-	rising edge	-	-	do capture ; issue NEWVAL_IRQ; CNT=0
-	0	0	-	-	no

10.4.2.3 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**.

10.4.2.3.1 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:

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

10.4.2.4 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**.

10.4.2.4.1 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:

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

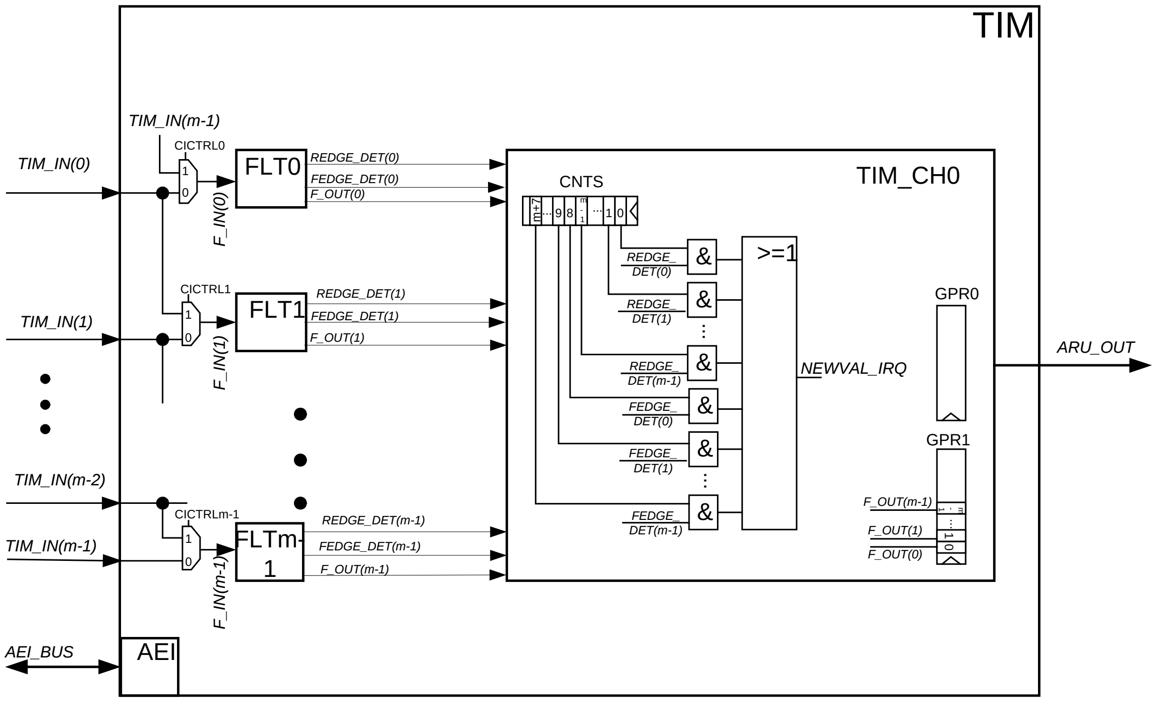
1	-	0	0	no

10.4.2.5 TIM Bit Compression Mode (TBCM)

The TIM Bit Compression Mode can be used to combine all filtered input signals of a TIM submodule 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 submodule.

Since this mode uses all available input signals with its input filters, it is only available within TIM channel 0 of each TIM submodule. Figure 10.4.2.5.1 gives an overview of the TIM bit compression mode.

10.4.2.5.1 TIM Bit Compression Mode



The register **CNTS** of TIM channel 0 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 10.4.2.5.1).

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]_CH0_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_IN0.

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**.

10.4.2.5.2 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:

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

10.4.2.6 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:

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 behaviour 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.

10.4.2.6.1 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:

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

10.5 MAP Submodule Interface

The GTM-IP provides one dedicated TIM submodule TIM0 where channels zero (0) to five (5) are connected to the MAP submodule f guetkdgf "kp"Ej cr vgt"15. There, the TIM0 submodule 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 submodule. This 49 bit wide data signal is marked as valid with a separate valid signal tim0_map_dval[x] (x: 0..5).

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 occurance of rising and falling edge.
Change **TIM0_CH[x]_FLT_FE** only between occurance of falling and rising edge.

10.6 TIM Interrupt Signals

TIM provides 6 interrupt lines per channel. These interrupts are shown below:

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_IRQ	A glitch was detected by the TIM filter of channel x (x: 0...m-1)

10.7 TIM Configuration Registers Overview

TIM contains following configuration registers:

Register Name	**Description**	**Detail in Section**
TIM[i]_CH[x]_CTRL	channel x (x:0...m-1) control	10.8.1 10.8.2
TIM[i]_CH[x]_ECTRL	channel x (x:0...m-1) extended control	10.8.20

TIM[i]_CH[x]_FLT_RE	channel x (x:0...m-1) filter parameter 0	10.8.3
TIM[i]_CH[x]_FLT_FE	channel x (x:0...m-1) filter parameter 1	10.8.4
TIM[i]_CH[x]_TDUV	channel x (x:0...m-1) TDU control.	10.8.17
TIM[i]_CH[x]_TDUC	channel x (x:0...m-1) TDU counter.	10.8.18
TIM[i]_CH[x]_GPR0	channel x (x:0...m-1) general purpose 0	10.8.6
TIM[i]_CH[x]_GPR1	channel x (x:0...m-1) general purpose 1	10.8.7
TIM[i]_CH[x]_CNT	channel x (x:0...m-1) SMU counter	10.8.8
TIM[i]_CH[x]_ECNT	channel x (x:0...m-1) SMU edge counter	10.8.19
TIM[i]_CH[x]_CNTS	channel x (x:0...m-1) SMU shadow counter	10.8.9
TIM[i]_CH[x]_IRQ_NOTIFY	channel x (x:0...m-1) interrupt notification	10.8.10
TIM[i]_CH[x]_IRQ_EN	channel x (x:0...m-1) interrupt enable	10.8.11
TIM[i]_CH[x]_EIRQ_EN	channel x (x:0...m-1) error interrupt enable	10.8.16
TIM[i]_CH[x]_IRQ_FORCINT	channel x (x:0...m-1) software interrupt force	10.8.12
TIM[i]_CH[x]_IRQ_MODE	IRQ mode configuration register (x=0...m-1)	10.8.13
TIM[i]_RST	TIM global software reset	10.8.14
TIM[i]_IN_SRC	TIM AUX IN source selection	10.8.15

10.8 TIM Configuration Registers Description

10.8.1 Register TIM[i]_CH[x]_CTRL (x:0...m-1) (i=1...n)

Address Offset:	**see Appendix B**																**Initial Value:** **0x0000_0000**															
	31	30	29	28	27	26	25	24	23	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
000 = PWM Measurement Mode (TPWM)
001 = Pulse Integration Mode (TPIM)
010 = Input Event Mode (TIEM)
011 = Input Prescaler Mode (TIPM)
100 = Bit Compression Mode (TBCM)
101 = Gated Periodic Sampling Mode (TGPS)

Note: The Bit Compression Mode is only available in TIM channel 0. If this mode is selected in any other channels, the TIM_MODE = 000 (TPWM mode) is used instead. However, the register TIM_MODE is read with value 100.

Note: If an undefined value is written to the TIM_MODE register, the hardware switches automatically to TIM_MODE = 000 (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 occured, a reconfiguration of DSL,ISL,TIM_MODE will not change the channel behaviour. Reading these bit fields after reconfiguration will show the newly configured settings but the inital channel

behaviour 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 If EGPR0_SEL =0: 00 = use TBU_TS0 as input 01 = use TBU_TS1 as input 10 = use TBU_TS2 as input 11 = use CNTS as input;if TGPS mode in channel = 0 is selected use TIM Filter F_OUT as input If EGPR0_SEL =1: 00 = use ECNT as input 01 = reserved 10 = reserved 11 = 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: 00 = use TBU_TS0 as input 01 = use TBU_TS1 as input 10 = use TBU_TS2 as input 11 = use CNT as input If EGPR1_SEL =1: 00 = use ECNT as input 01 = reserved 10 = reserved 11 = reserved

Note: In TBCM mode EGPR1_SEL, GPR1_SEL are ignored TIM Filter F_OUT is used as input

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 and TBCM.
Bit 13	**DSL:** Signal level control 0 = Measurement starts with falling edge (low level measurement) 1 = Measurement starts with rising edge (high level measurement)
Bit 14	**ISL:** Ignore signal level 0 = use DSL bit for selecting active signal level 1 = ignore DSL and treat both edges as active edge **Note:** This bit is only applicable in Input Event mode (TIEM)
Bit 15	**ECNT_RESET:** Enables resetting the ECNT counter in periodic sampling mode 0 = ECNT counter operating in wrap around mode 1 = ECNT counter is reset with periodic sampling
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 00 = FLT_CNT counts with *CMU_CLK0* 01 = FLT_CNT counts with *CMU_CLK1* 10 = FLT_CNT counts with *CMU_CLK6* 11 = 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. 0 = Up/Down Counter 1 = Hold Counter

Note: This bit is only applicable in Individual Deglitch Time Mode

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.
0 = Up/Down Counter
1 = Hold Counter
Note: This bit is only applicable in Individual Deglitch Time Mode

Bit 26:24 **CLK_SEL:** CMU clock source select for channel.
000 = *CMU_CLK0* selected
001 = *CMU_CLK1* selected
010 = *CMU_CLK2* selected
011 = *CMU_CLK3* selected
100 = *CMU_CLK4* selected
101 = *CMU_CLK5* selected
110 = *CMU_CLK6* selected
111 = *CMU_CLK7* selected

Bit 27 **FR_ECNT_OFL:** Extended Edge counter overflow behaviour
0 = Overflow will be signalled on ECNT bit width = 8
1 = Overflow will be signalled 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
00 = Timeout feature disabled
11 = Timeout feature enabled for both edges
01 = Timeout feature enabled for rising edge only
10 = Timeout feature enabled for falling edge only

10.8.2 Register TIM0_CH[x]_CTRL (x:0...m-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	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
000 = PWM Measurement Mode (TPWM)
001 = Pulse Integration Mode (TPIM)
010 = Input Event Mode (TIEM)
011 = Input Prescaler Mode (TIPM)
100 = Bit Compression Mode (TBCM)
101 = Gated Periodic Sampling Mode (TGPS)

Note: The Bit Compression Mode is only available in TIM channel 0. If this mode is selected in any other channels, the TIM_MODE = 000 (TPWM mode) is used instead. However, the register TIM_MODE is read with value 100.

Note: If an undefined value is written to the TIM_MODE register, the hardware switches automatically to TIM_MODE = 000 (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 occured, a reconfiguration of DSL,ISL,TIM_MODE will not change the channel behaviour. Reading these bit fields after reconfiguration will show the newly configured settings but the inital channel

behaviour will not change. Only a disabling of the TIM channel by setting TIM_EN= 0 will stop the channel operation.

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:
  00 = use TBU_TS0 as input
  01 = use TBU_TS1 as input
  10 = use TBU_TS2 as input
  11 = use CNTS as input;if TGPS mode in channel = 0 is selected use TIM Filter F_OUT as input
If EGPR0_SEL =1:
  00 = use ECNT as input
  01 = reserved
  10 = reserved
  11 = 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:
  00 = use TBU_TS0 as input
  01 = use TBU_TS1 as input
  10 = use TBU_TS2 as input
  11 = use CNT as input
If EGPR1_SEL =1:
  00 = use ECNT as input

	01 = reserved 10 = reserved 11 = reserved **Note**: In TBCM mode EGPR1_SEL, GPR1_SEL are ignored TIM Filter F_OUT is used as input **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 and TBCM.
Bit 13	**DSL:** Signal level control 0 = Measurement starts with falling edge (low level measurement) 1 = Measurement starts with rising edge (high level measurement)
Bit 14	**ISL:** Ignore signal level 0 = use DSL bit for selecting active signal level 1 = ignore DSL and treat both edges as active edge Note: This bit is only applicable in Input Event mode (TIEM and TIPM)
Bit 15	**ECNT_RESET:** Enables resetting the ECNT counter in periodic sampling mode 0 = ECNT counter operating in wrap around mode 1 = ECNT counter is reset with periodic sampling
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 00 = FLT_CNT counts with *CMU_CLK0* 01 = FLT_CNT counts with *CMU_CLK1* 10 = FLT_CNT counts with *CMU_CLK6* 11 = 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. 0 = Up/Down Counter 1 = Hold Counter **Note**: This bit is only applicable in Individual Deglitch Time Mode
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. 0 = Up/Down Counter 1 = Hold Counter **Note**: This bit is only applicable in Individual Deglitch Time Mode
Bit 26:24	**CLK_SEL:** CMU clock source select for channel. 000 = *CMU_CLK0* selected 001 = *CMU_CLK1* selected 010 = *CMU_CLK2* selected 011 = *CMU_CLK3* selected 100 = *CMU_CLK4* selected 101 = *CMU_CLK5* selected 110 = *CMU_CLK6* selected 111 = *CMU_CLK7* selected
Bit 27	**FR_ECNT_OFL:** Extended Edge counter overflow behaviour 0 = Overflow will be signalled on ECNT bit width = 8 1 = Overflow will be signalled 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 00 = Timeout feature disabled 01 = Timeout feature enabled for rising edge only 10 = Timeout feature enabled for falling edge only 11 = Timeout feature enabled for both edges

10.8.3 Register TIM[i]_CH[x]_FLT_RE (x:0...m-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								FLT_RE																							
Mode	R								RW																							
Initial Value	0x00								0x000000																							

Bit 23:0 **FLT_RE:** Filter parameter for rising edge.
Note: This register 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

10.8.4 Register TIM[i]_CH[x]_FLT_FE (x:0...m-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								FLT_FE																							
Mode	R								RW																							
Initial Value	0x00								0x000000																							

Bit 23:0 **FLT_FE:** Filter parameter for falling edge.
Note: This register 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

10.8.5 Register TIM[i]_CH[x]_TDU (x:0...m-1)

Functionality is replaced by register TIM[i]_CH[x]_TDUV, TIM[i]_CH[x]_TDUC and bit field TOCTRL in Register TIM[i]_CH[x]_CTRL

10.8.6 Register TIM[i]_CH[x]_GPR0 (x:0...m-1)

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	0x000000

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.

10.8.7 Register TIM[i]_CH[x]_GPR1 (x:0...m-1)

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 EGPR1_SEL, GPR1_SEL are ignored; TIM Filter F_OUT is used as input, Bits GPR1(23:8) = 0

Note: The content of this register can only be written in TIM channel mode TGPS.

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.

10.8.8 Register TIM[i]_CH[x]_CNT (x:0...m-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																							
Mode	R								R																							
Initial Value	0x00								0x0000 00																							

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.

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

10.8.9 Register TIM[i]_CH[x]_CNTS (x:0...m-1)

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	0x00								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 and TGPS 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.

10.8.10 Register TIM[i]_CH[x]_IRQ_NOTIFY (x:0...m-1)

Address Offset:	**see Appendix B**	**Initial Value:**	**0x0000_0000**				
	31 … 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

10.8.11 Register TIM[i]_CH[x]_IRQ_EN (x:0...m-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																										GLITCHDET_IRQ_EN	TODET_IRQ_EN	GPROFL_IRQ_EN	CNTOFL_IRQ_EN	ECNTOFL_IRQ_EN	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

10.8.12 Register TIM[i]_CH[x]_IRQ_FORCINT (x:0...m-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																										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

10.8.13 Register TIM[i]_CH[x]_IRQ_MODE (x:0...m-1)

Address Offset:	**see Appendix B**	**Initial Value:**	**0x0000_000X**
	31 … 2		1 0
Bit	Reserved		IRQ_MODE
Mode	R		RW
Initial Value	0x0000 0000		XX

Bit 1:0	**IRQ_MODE**: IRQ mode selection 00 = Level mode 01 = Pulse mode 10 = Pulse-Notify mode 11 = 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

10.8.14 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 submodule. This register description represents a register layout for *m* = 8.

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

10.8.15 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
multicore encoding in use (**VAL_x(1)** defines the state of the signal)
00 = State is 0 (ignore write access)
01 = Change state to 0
10 = Change state to 1
11 = 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
multicore encoding in use (**MODE_x(1)** defines the state of the signal)
00 = State is 0 (ignore write access)
01 = Change state to 0
10 = Change state to 1
11 = 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.

10.8.16 Register TIM[i]_CH[x]_EIRQ_EN (x:0...m-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																										GLITCHDET_EIRQ_EN	TODET_EIRQ_EN	GPROFL_EIRQ_EN	CNTOFL_EIRQ_EN	ECNTOFL_EIRQ_EN	NEWVAL_EIRQ_EN
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

10.8.17 Register TIM[i]_CH[x]_TDUV (x:0...m-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	TCS			Reserved																				TOV							
Mode	R	RW																							RW							
Initial Value	0	000			0x0000																				0x00							

Bit 7:0	**TOV:** Time out duration for channel x (x:0...m-1).
Bit 27:8	**Reserved:** Reserved **Note:** Read as zero, should be written as zero
Bit 30:28	**TCS:** Timeout Clock selection 000 = *CMU_CLK0* selected 001 = *CMU_CLK1* selected 010 = *CMU_CLK2* selected 011 = *CMU_CLK3* selected 100 = *CMU_CLK4* selected 101 = *CMU_CLK5* selected 110 = *CMU_CLK6* selected 111 = *CMU_CLK7* selected
Bit 31	**Reserved:** Reserved **Note:** Read as zero, should be written as zero

10.8.18 Register TIM[i]_CH[x]_TDUC (x:0...m-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																								TO_CNT							
Mode	R																								R							
Initial Value	0x000000																								0x00							

Bit 7:0 **TO_CNT:** Current Timeout value for channel x (x:0...m-1).

Bit 31:8 **Reserved:** Reserved

Note: Read as zero, should be written as zero

10.8.19 Register TIM[i]_CH[x]_ECNT (x:0...m-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																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

10.8.20 Register TIM[i]_CH[x]_ECTRL (x:0...m-1)

Address Offset:	**see Appendix B**	**Initial Value:**	**0x0000_0000**
	31 … 3		2 … 0
Bit	Reserved		EXT_CAP_SRC
Mode	R		RW
Initial Value	0x0000 0000		000

Bit 2:0 **EXT_CAP_SRC:** Defines selected source for triggering the EXT_CAPTURE functionality.
000 = NEW_VAL_IRQ of following channel selected
001 = AUX_IN selected
010 = CNTOFL_IRQ of following channel selected
011 and CICTRL = 1 : use signal TIM_IN(x) as input for channel x
011 and CICTRL = 0 : use signal TIM_IN(x-1) as input for channel x (or TIM_IN(m-1) if x is 0)
100 = ECNTOFL_IRQ of following channel selected
101 =TODET_IRQ of following channel selected
110 = GLITCHDET_IRQ of following channel selected
111 = GPROFL_IRQof following channel selected

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

11 Timer Output Module (TOM)

11.1 Overview

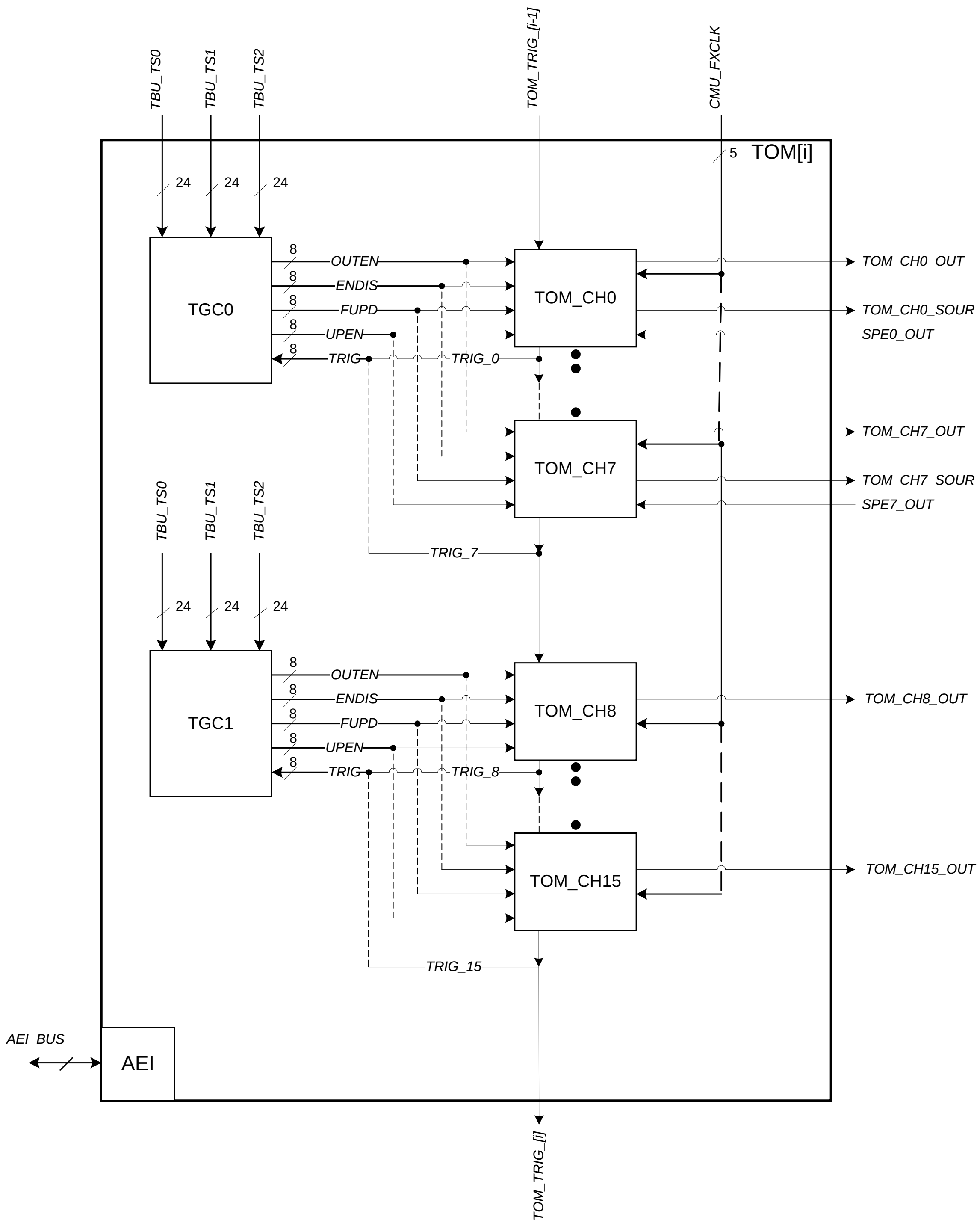
The Timer Output Module (TOM) offers up to 16 independent channels (index x) to generate simple PWM signals at each output pin *TOM[i]_CH[x]_OUT*.

Additionally, at TOM output *TOM[i]_CH15_OUT* a pulse count modulated signal can be generated.
The architecture of the TOM submodule is depicted in figure 11.1.1.
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 Appenedix B for correct value.

cCTO : TOM channel count; number of channels per instance - 1.

11.1.1 TOM Block diagram



The two submodules 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 behaviour 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.

11.2 TOM Global Channel Control (TGC0, TGC1)

11.2.1 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 submodule supports four different kinds of signalling 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**

11.2.2 TGC Subunit

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]*)
  Note: The trigger signal is only active for one configured CMU clock period.

As a consequence, if the configured CMU clock period of the channel generating the trigger *TRIG_[X]* is smaller than the CMU clock period of triggered channel, the triggered channel may not recognize the trigger signal *TRIG_[X]* .
Thus, it is recommended to configure all channels connected via the trigger*TRIG_[x]* to the same CMU clock period.

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.

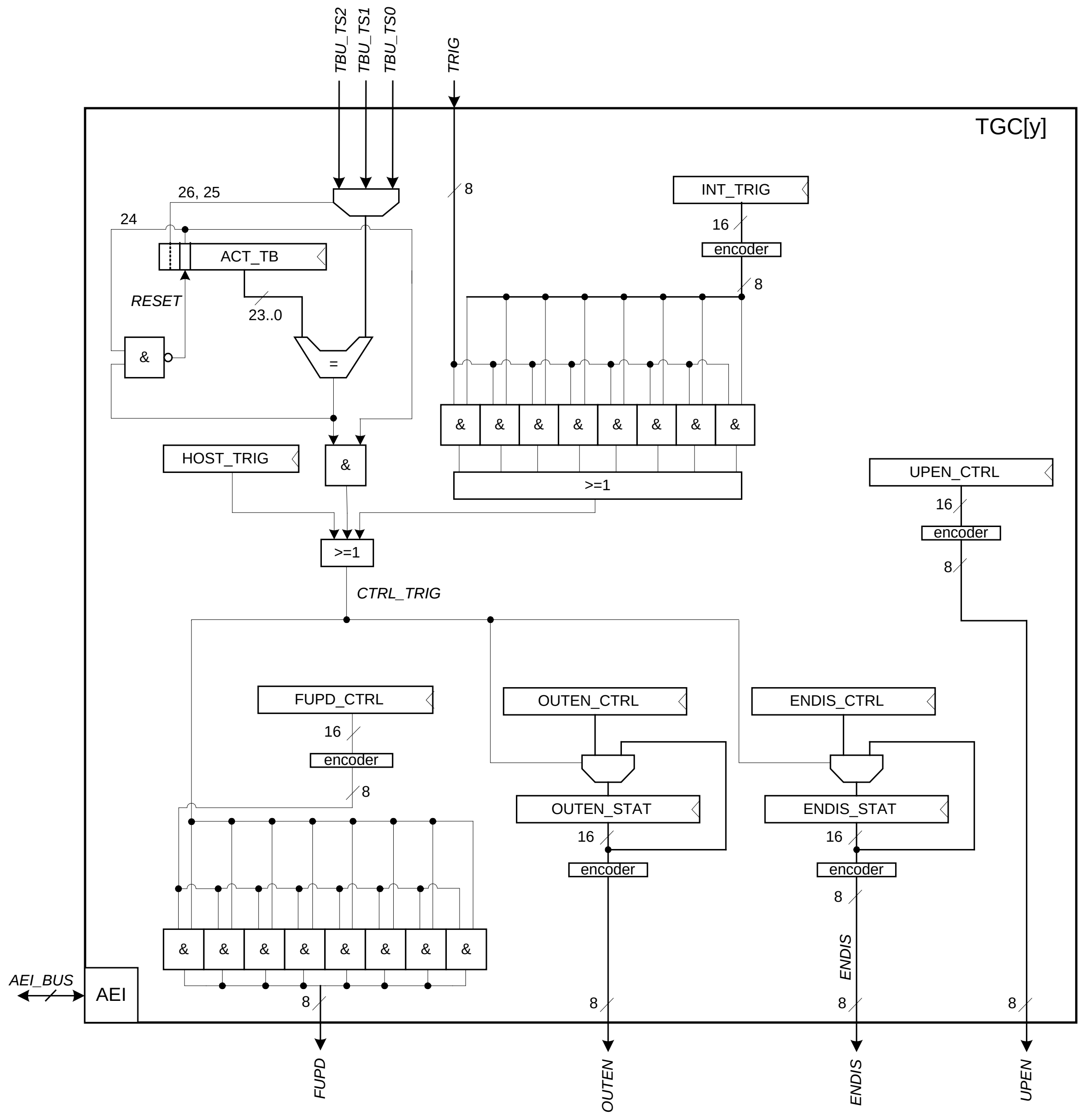
The corresponding trigger signal *TRIG_[x]* coming from channel [x] can be masked by the register **TOM[i]_TGC[y]_INT_TRIG**.

To enable or disable each individual TOM channel, the registers **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.

11.2.2.1 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 FlipFlop **SOUR**.

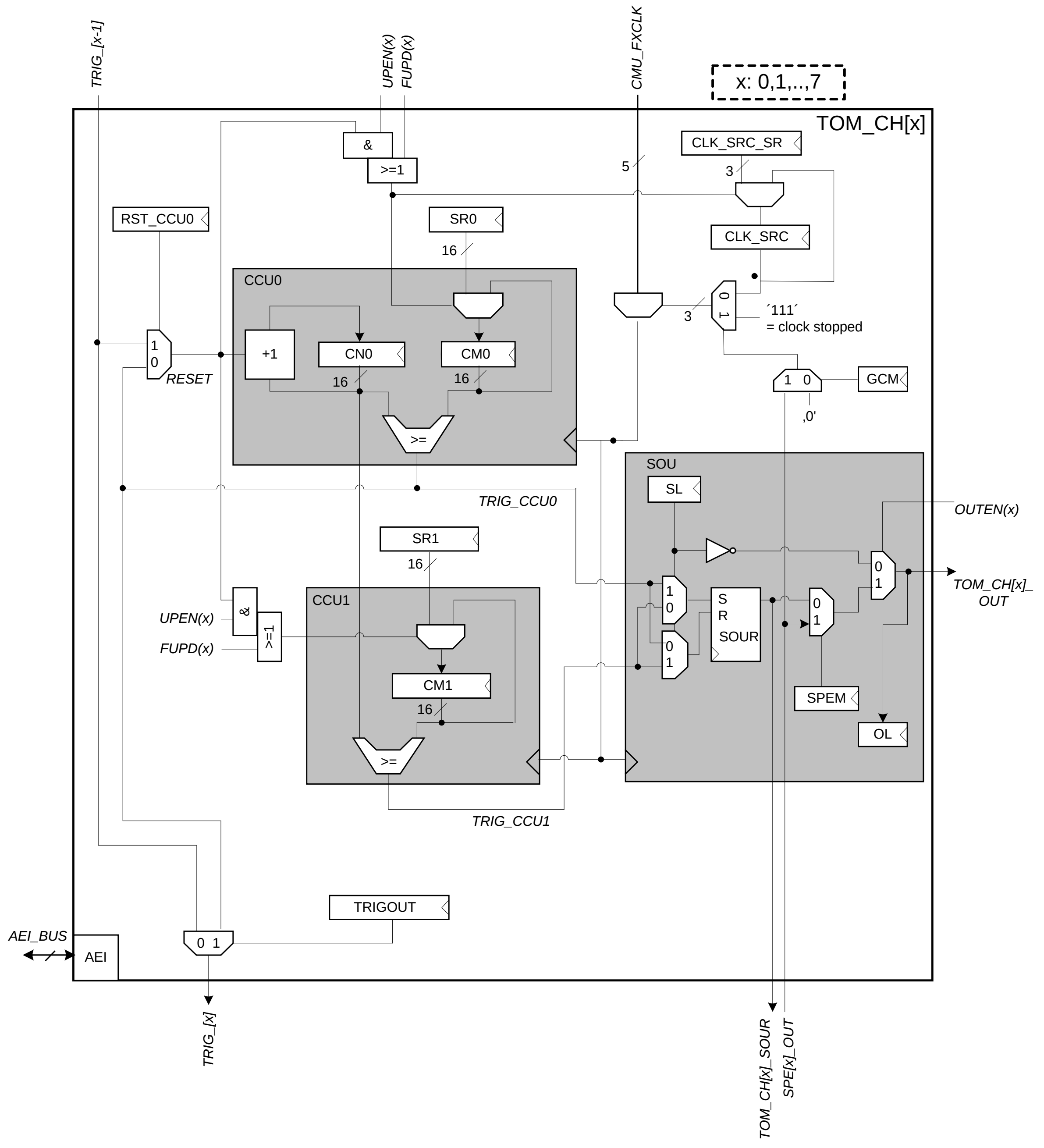
The register **TOM[i]_TGC[y]_FUPD_CTRL** defines which of the TOM channels receive a *FORCE_UPDATE* event if the trigger signal *CTRL_TRIG* is raised.

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 11.3.1 and 11.3.2).

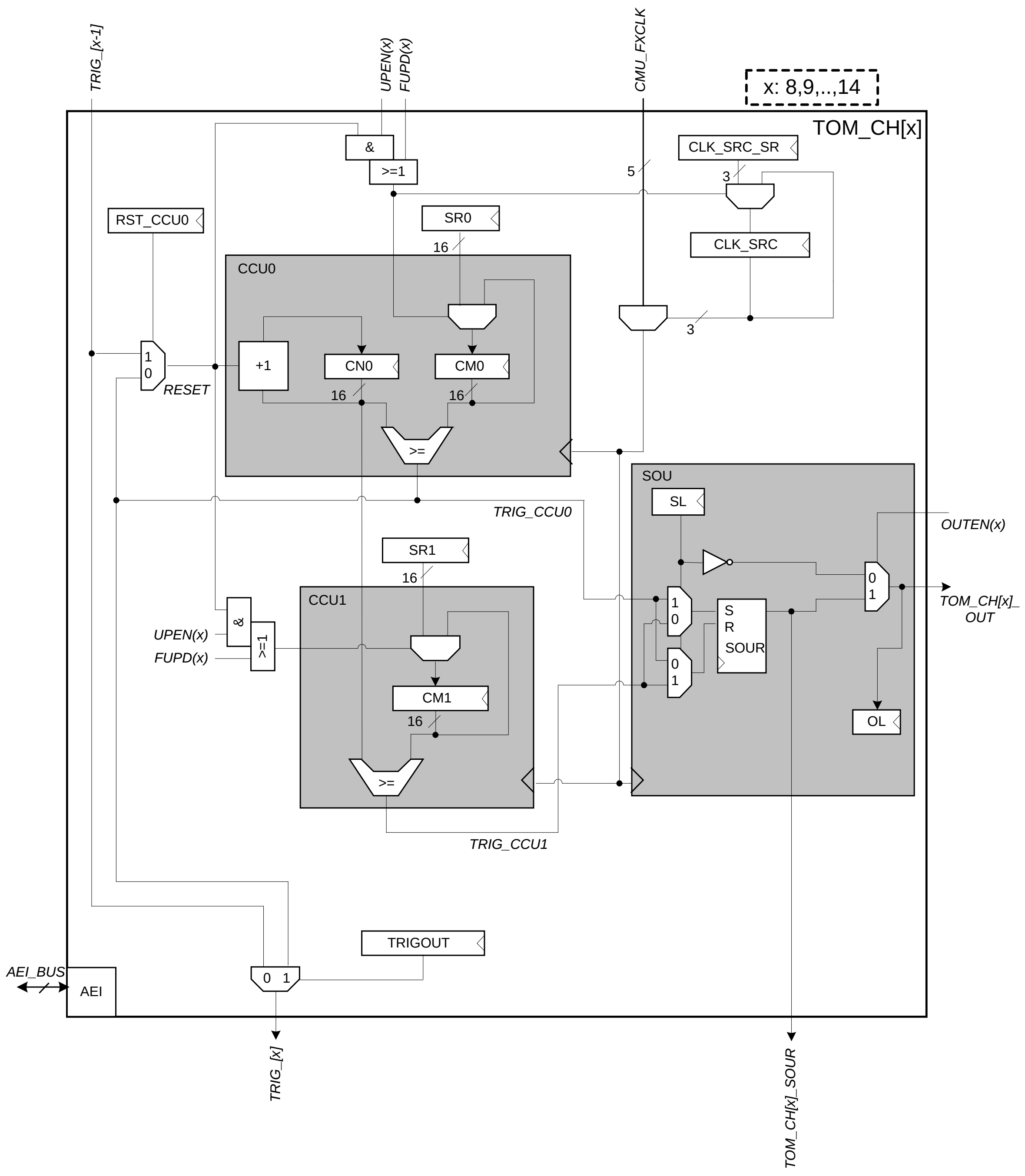
11.3 TOM Channel (TOM_CH[x])

Each individual TOM channel comprises a Counter Compare Unit 0 (CCU0), a Counter Compare Unit 1 (CCU1) and the Signal Output Generation Unit (SOU). The architecture is depicted in figure 11.3.1 for channels 0 to 7 and in 11.3.2 for channels 8 to 15.

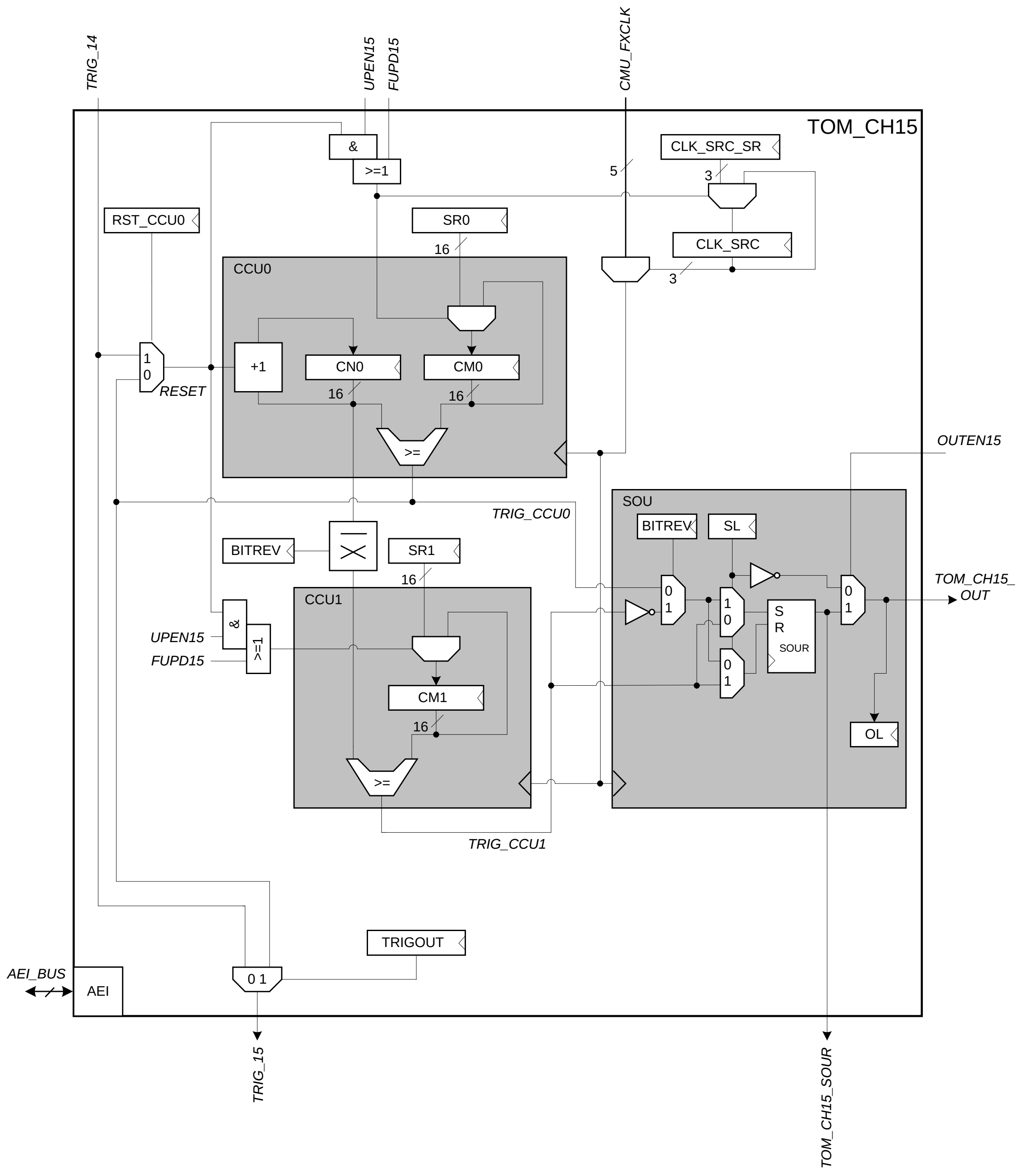
11.3.1 TOM Channel 0..7 architecture



11.3.2 TOM Channel 8..14 architecture



11.3.3 TOM Channel 15 architecture for PCM generation



The CCU0 contains a counter **CN0** which is clocked with one of the selected input frequencies (*CMU_FXCLK*) provided from outside of the submodule.

Depending on configuration bits RST_CCU0 of register **TOM[i]_CH[x]_CTRL** the counter can be reset either when the counter value is equal to the compare value **CM0** or when signalled 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]).

When the counter register **CN0** is greater or equal than the register **CM0,** the subunit CCU0 triggers the SOU subunit and the succeeding TOM submodule channel (signal *TRIG_CCU0*).

In the subunit CCU1 the counter register **CN0** is compared with the value of register **CM1**. If **CN0** is greater or equal than **CM1** the subunit CCU1 triggers the SOU subunit (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** configured by RST_CCU0=0), following statements are valid:
- the configuration of **CM1**=0 represents 0% duty cycle at the output,
- the configuration of **CM1** >= **CM0** represents 100% duty cycle
- if both registers are configured to 0 (**CM0**=**CM1**=0), the output is 0% duty cycle
- if **CM0**=0, 0% duty cylce is generated independent of **CM1**.

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:
- **CM0** defines the edge to SL value, **CM1** defines the edge to !SL value.
- if **CM0**=**CM1**, the output is 100% SL (**CM0** has higher priority)
- if **CM0**=0, the output stays at its last value (**CN0** stops counting)

The hardware ensures that for both 0% and 100% duty cycle no glitch occurs at the output of the TOM channel.
The SOU subunit is responsible for output signal generation. On a trigger *TRIG_CCU0* from subunit CCU0 or *TRIG_CCU1* from subunit CCU1 a SR-FlipFlop of subunit 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 11.3.5.1 clarifies the PWM output behaviour 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**.

11.3.4 Duty cycle, period and selected counter clock frequency update mechanisms

The two action registers **CM0** and **CM1** can be reloaded with the content of the shadow registers **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]*.
For TOM0 channel 0 *TRIG_[-1]* is '0' hard coded.

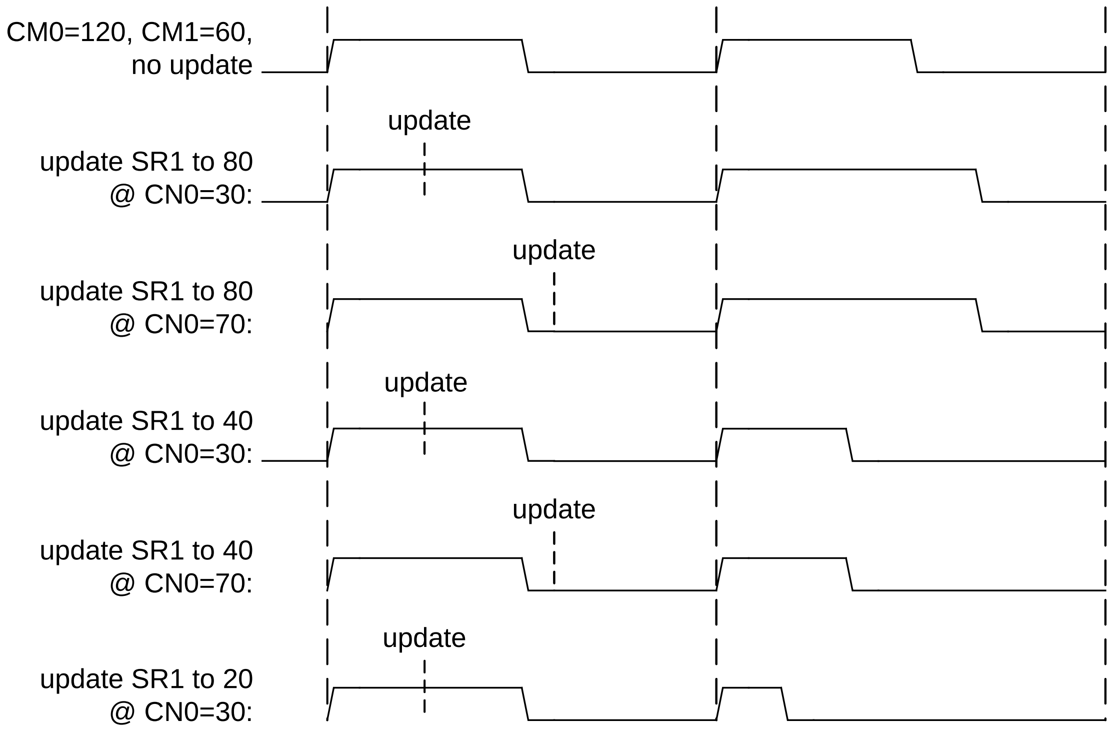
With the update of the register **CLK_SRC** at the end of a period a new counter **CN0** clock frequency can easily be adjusted.

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'.

*11.3.4.1 **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.

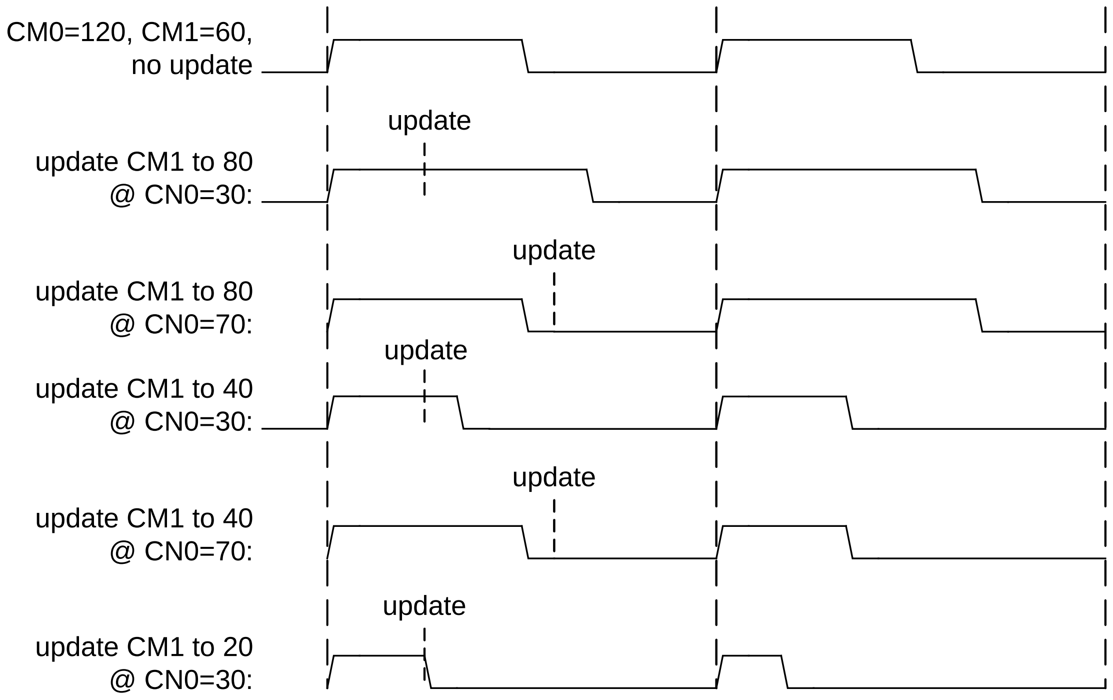
11.3.4.1.1 synchronous update of duty cycle



11.3.4.2 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 11.3.4.2.1). 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.

11.3.4.2.1 asynchronous update of duty cycle



11.3.5 TOM continuous mode

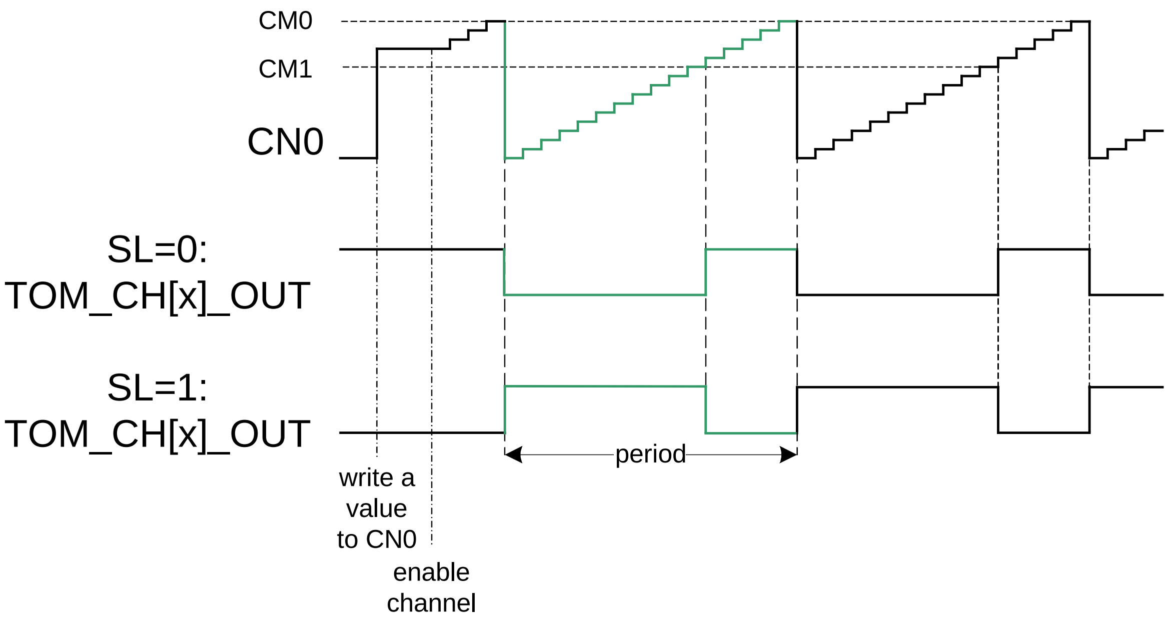
In continuous mode the TOM channel starts incrementing the counter register **CN0** once it is enabled by setting the corresponding bits in register **TOM[i]_TGC[y]_ENDIS_STAT** (refer to chapter 11.2.2 for details of enabling a TOM channel).

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.

*11.3.5.1 PWM Output with respect to configuration bit **SL** in continuous mode*



11.3.6 TOM One shot 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**.

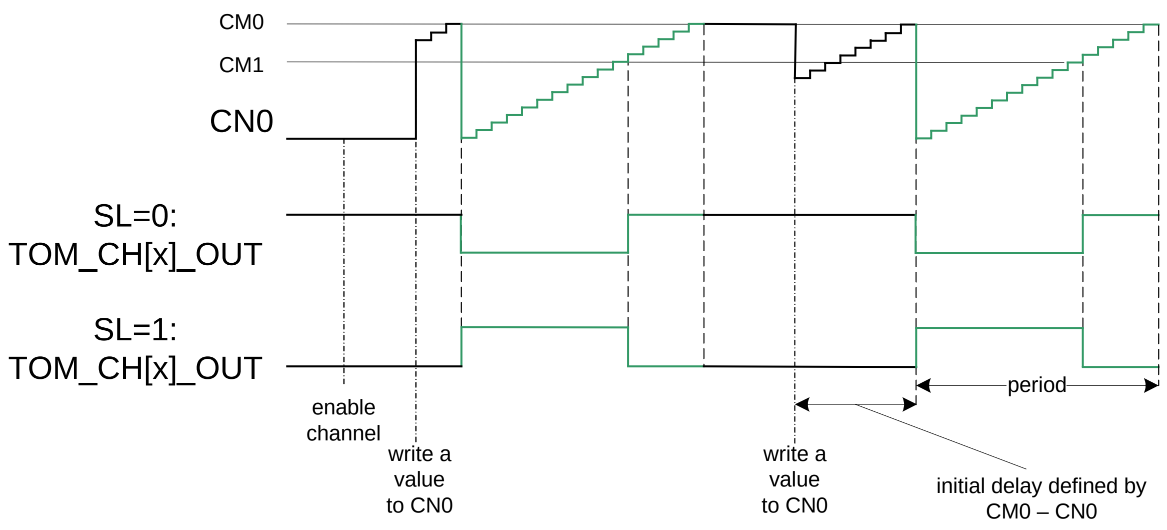
If the counter **CN0** is reset from **CM0** back to zero, the first edge at *TOM[i]_CH[x]_OUT* is generated.

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]_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** a second time, the counter stops.

11.3.6.1 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.

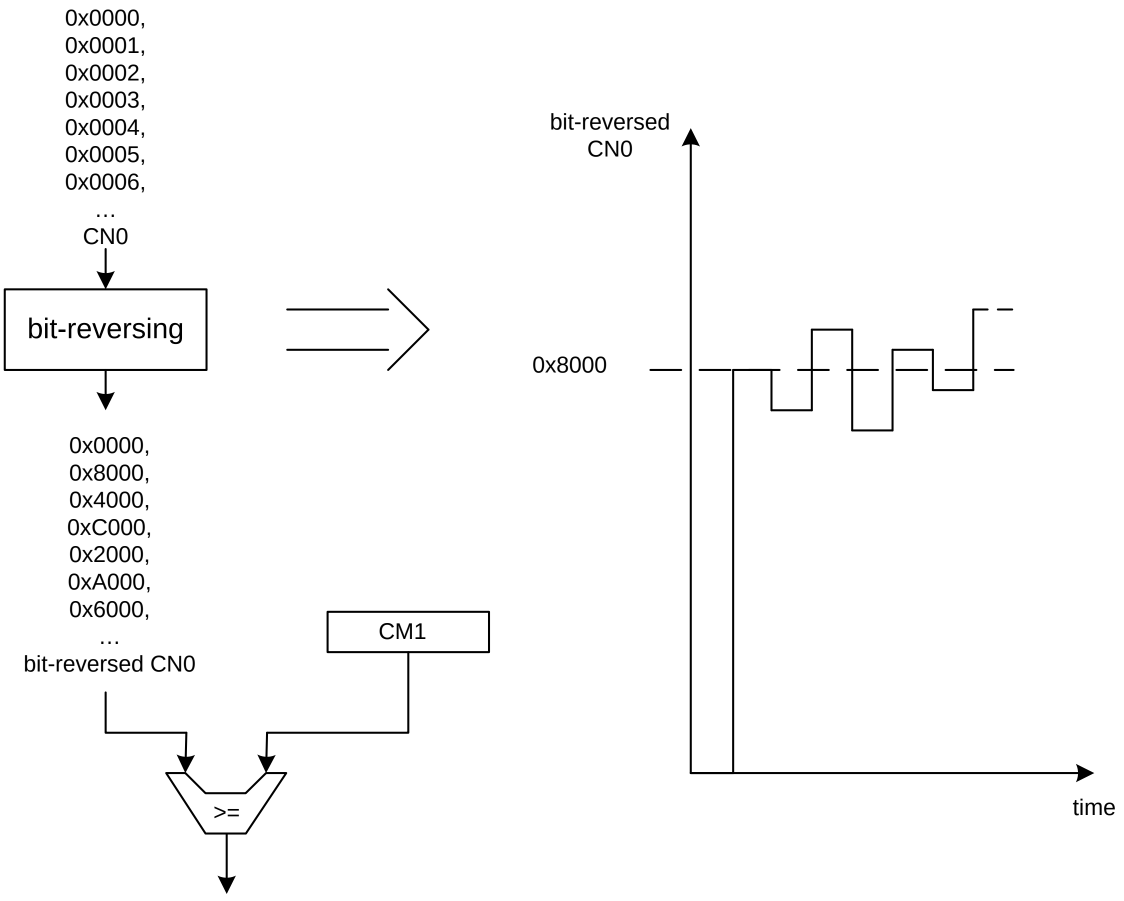
11.3.7 Pulse count modulation

At the output *TOM[i]_CH15_OUT* a pulse count modulated signal can be generated instead of the simple PWM output signal.

Figure 11.3.3 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 11.3.7.1.

*11.3.7.1 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_FXCLK*).

The output of counter register **CN0** is first bit-reversed and than compared with the configured register value **CM1**.

If the bit-reversed value of register **CN0** is greater than **CM1**, the SR-FlipFlop of submodule SOU is set (depending on configuration register **SL**) otherwise the SR-FlipFlop 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.

Note: In this mode the interrupt CCU1TC (see register **ATOM[i]_CH[x]_IRQ_NOTIFY**) is set every thime if bitreverse value of **CN0** is greater or equalt than **CM1** which may be multiple times during one period. Therefore, from application point of view it is not usefull to enable this interrupt.

11.4 TOM BLDC Support

The TOM submodule offers in combination with the SPE submodule 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 submodule (see figure 11.3.1). Please refer to chapter 17 for a detailed description of the SPE submodule.

The TOM[i]_CH2 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 (signalled by *SPE[i]_NIPD*). This feature is configured on TOM[i]_CH2 by setting SPEM=1 and OSM=1. For details please refer to chapter of SPE submodule description.

11.5 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 submodule.
To configure this mode, registers 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 submodule CCU0 of TOM channel c counts as long as input *FSOI* is '0'.

11.6 TOM Interrupt signals

The following table describes 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

11.7 TOM Configuration Register overview

The following table gives an overview of the TOM configuration register.

Register name	**Description**	**Details in Section**
TOM[i]_TGC0_GLB_CTRL	TGC0 global control reg	11.8.1
TOM[i]_TGC0_ENDIS_CTRL	TGC0 enable/disable control reg	11.8.2
TOM[i]_TGC0_ENDIS_STAT	TGC0 enable/disable status reg	11.8.3
TOM[i]_TGC0_ACT_TB	TGC0 action time base register	11.8.4
TOM[i]_TGC0_OUTEN_CTRL	TGC0 output enable control reg	11.8.5
TOM[i]_TGC0_OUTEN_STAT	TGC0 output enable status reg	11.8.6
TOM[i]_TGC0_FUPD_CTRL	TGC0 force update control reg	11.8.7
TOM[i]_TGC0_INT_TRIG	TGC0 internal trigger control	11.8.8
TOM[i]_TGC1_GLB_CTRL	TGC1 global control register	11.8.9
TOM[i]_TGC1_ENDIS_CTRL	TGC1 enable/disable control reg	11.8.10
TOM[i]_TGC1_ENDIS_STAT	TGC1 enable/disable status reg	11.8.11
TOM[i]_TGC1_ACT_TB	TGC1 action time base register	11.8.12
TOM[i]_TGC1_OUTEN_CTRL	TGC1 output enable control reg	11.8.13
TOM[i]_TGC1_OUTEN_STAT	TGC1 output enable status reg	11.8.14
TOM[i]_TGC1_FUPD_CTRL	TGC1 force update control reg	11.8.15
TOM[i]_TGC1_INT_TRIG	TGC1 internal trigger control	11.8.16
TOM[i]_CH[x]_CTRL	TOM Channel x control register (x=0…14)	11.8.17

TOM[i]_CH15_CTRL	TOM Channel 15 control reg	11.8.18
TOM[i]_CH[x]_CN0	TOM Channel x CCU0 counter register (x=0…15)	11.8.19
TOM[i]_CH[x]_CM0	TOM Channel x CCU0 compare register (x=0…15)	11.8.20
TOM[i]_CH[x]_SR0	TOM Channel x CCU0 compare shadow register (x=0…15)	11.8.21
TOM[i]_CH[x]_CM1	TOM Channel x CCU1 compare register (x=0…15)	11.8.22
TOM[i]_CH[x]_SR1	TOM Channel x CCU1 compare shadow register (x=0…15)	11.8.23
TOM[i]_CH[x]_STAT	TOM channel status (x=0…15)	11.8.24
TOM[i]_CH[x]_IRQ_NOTIFY	TOM channel x interrupt notification register (x=0…15)	11.8.25
TOM[i]_CH[x]_IRQ_EN	TOM channel x interrupt enable register (x=0…15)	11.8.26
TOM[i]_CH[x]_IRQ_FORCINT	TOM channel x software interrupt generation (x=0…15)	11.8.27
TOM[i]_CH[x]_IRQ_MODE	IRQ mode configuration register (x=0…15)	11.8.28

11.8 TOM Configuration Registers Description

11.8.1 Register TOM[i]_TGC0_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		Aw	Aw	Aw	Aw	Aw	Aw	Aw	Aw	R							Aw
Initial Value	00		00		00		00		00		00		00		00		0	0	0	0	0	0	0	0	0							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 registers are set to their reset values and channel operation is stopped immediately. The S-R FlipFlop **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 of following double bit values is possible: 00 = don’t care, bits 1:0 will not be changed 01 = update disabled: is read as 00 (see below) 10 = update enabled: is read as 11 (see below) 11 = don’t care, bits 1:0 will not be changed Read of following double values means: 00 = channel disabled 11 = channel 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

11.8.2 Register TOM[i]_TGC0_ENDIS_CTRL

Address Offset:	**see Appendix B**	**Initial Value:** **0x0000_0000**							
	31 … 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	00	00	00	00	00	00	00	00

Bit 1:0 **ENDIS_CTRL0**: TOM channel 0 enable/disable update value.
If a TOM channel is disabled, the counter **CN0** is stopped and the FlipFlop **SOUR** is set to the inverse value of control bit SL. On an enable event, the counter **CN0** starts counting from its current value.
Write of following double bit values is possible:
00 = don't care, bits 1:0 of register ENDIS_STAT will not be changed on an update trigger
01 = disable channel on an update trigger
10 = enable channel on an update trigger
11 = 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_OUT[0] 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

11.8.3 Register TOM[i]_TGC0_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																00		00		00		00		00		00		00		00	

Bit 1:0 **ENDIS_STAT0**: TOM channel 0 enable/disable

If a TOM channel is disabled, the counter **CN0** is stopped and the FlipFlop **SOUR** is set to the inverse value of control bit SL. On an enable event, the counter **CN0** starts counting from its current value.

Write of following double bit values is possible:

00 = don't care, bits 1:0 will not be changed

01 = channel disabled: is read as 00 (see below)

10 = channel enabled: is read as 11 (see below)

11 = don't care, bits 1:0 will not be changed

Read of following double values means:

00 = channel disable

11 = channel enable

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

11.8.4 Register TOM[i]_TGC0_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	00000					00		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 00 = *TBU_TS0* selected 01 = *TBU_TS1* selected

10 = *TBU_TS2* selected
11 = same as 00
Note: The bit combination “10” 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

11.8.5 Register TOM[i]_TGC0_OUTEN_CTRL

Address Offset:	**see Appendix B**	**Initial Value:**	**0x0000_0000**						
	31–16	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	00	00	00	00	00	00	00	00

Bit 1:0 **OUTEN_CTRL0**: Output TOM_OUT(0) enable/disable update value
Write of following double bit values is possible:
00 = don’t care, bits 1:0 of register OUTEN_STAT will not be changed on an update trigger
01 = disable channel output on an update trigger
10 = enable channel output on an update trigger
11 = 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_OUT[0] is the inverted value of bit SL.

Bit 3:2 **OUTEN_CTRL1**: Output TOM_OUT(1)enable/disable update value
See bits 1:0
Bit 5:4 **OUTEN_CTRL2**: Output TOM_OUT(2) enable/disable update value
See bits 1:0
Bit 7:6 **OUTEN_CTRL3**: Output TOM_OUT(3) enable/disable update value
See bits 1:0
Bit 9:8 **OUTEN_CTRL4**: Output TOM_OUT(4) enable/disable update value

	See bits 1:0
Bit 11:10	**OUTEN_CTRL5**: Output TOM_OUT(5) enable/disable update value See bits 1:0
Bit 13:12	**OUTEN_CTRL6**: Output TOM_OUT(6) enable/disable update value See bits 1:0
Bit 15:14	**OUTEN_CTRL7**: Output TOM_OUT(7) enable/disable update value See bits 1:0
Bit 31:16	**Reserved** Note: Read as zero, should be written as zero

11.8.6 Register TOM[i]_TGC0_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																00		00		00		00		00		00		00		00	

Bit 1:0	**OUTEN_STAT0**: Control/status of output TOM_OUT(0) Write of following double bit values is possible: 00 = don’t care, bits 1:0 will not be changed 01 = channel disabled: is read as 00 (see below) 10 = channel enabled: is read as 11 (see below) 11 = don’t care, bits 1:0 will not be changed Read of following double values means: 00 = channel disable 11 = channel enable
Bit 3:2	**OUTEN_STAT1**: Control/status of output TOM_OUT(1) See bits 1:0
Bit 5:4	**OUTEN_STAT2**: Control/status of output TOM_OUT(2) See bits 1:0
Bit 7:6	**OUTEN_STAT3**: Control/status of output TOM_OUT(3) See bits 1:0
Bit 9:8	**OUTEN_STAT4**: Control/status of output TOM_OUT(4) See bits 1:0
Bit 11:10	**OUTEN_STAT5**: Control/status of output TOM_OUT(5) See bits 1:0

Bit 13:12	**OUTEN_STAT6**: Control/status of output TOM_OUT(6) See bits 1:0
Bit 15:14	**OUTEN_STAT7**: Control/status of output TOM_OUT(7) See bits 1:0
Bit 31:16	**Reserved** Note: Read as zero, should be written as zero

11.8.7 Register TOM[i]_TGC0_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	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Bit 1:0	**FUPD_CTRL0**: Force update of TOM channel 0 operation registers Write of following double bit values is possible: 00 = don't care, bits 1:0 will not be changed 01 = force update disabled: is read as 00 (see below) 10 = force update enabled: is read as 11 (see below) 11 = don't care, bits 1:0 will not be changed Read of following double values means: 00 = force update disabled 11 = force channel enabled
Bit 3:2	**FUPD_CTRL1**: Force update of TOM channel 1 operation registers See bits 1:0
Bit 5:4	**FUPD_CTRL2**: Force update of TOM channel 2 operation registers See bits 1:0
Bit 7:6	**FUPD_CTRL3**: Force update of TOM channel 3 operation registers See bits 1:0
Bit 9:8	**FUPD_CTRL4**: Force update of TOM channel 4 operation registers See bits 1:0
Bit 11:10	**FUPD_CTRL5**: Force update of TOM channel 5 operation registers See bits 1:0
Bit 13:12	**FUPD_CTRL6**: Force update of TOM channel 6 operation registers See bits 1:0
Bit 15:14	**FUPD_CTRL7**: Force update of TOM channel 7 operation registers

	See bits 1:0
Bit 17:16	**RSTCN0_CH0**: Reset CN0 of channel 0 on force update event Write of following double bit values is possible: 00 = don't care, bits 1:0 will not be changed 01 = CN0 is not reset on forced update: is read as 00 (see below) 10 = CN0 is reset on forced update: is read as 11 (see below) 11 = don't care, bits 1:0 will not be changed Read of following double values means: 00 = CN0 is not reset on forced update 11 = 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

11.8.8 Register TOM[i]_TGC0_INT_TRIG

Address Offset:	**see Appendix B**	**Initial Value:**	**0x0000_0000**						
	31–16	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	00	00	00	00	00	00	00	00

Bit 1:0	**INT_TRIG0**: Select input signal *TRIG_0* as a trigger source Write of following double bit values is possible: 00 = don't care, bits 1:0 will not be changed

	01 = internal trigger from channel 0 (*TRIG_0*) not used: is read as 00 (see below) 10 = internal trigger from channel 0 (*TRIG_0*) used: is read as 11 (see below) 11 = don't care, bits 1:0 will not be changed Read of following double values means: 00 = internal trigger from channel 0 (*TRIG_0*) not used 11 = 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

11.8.9 Register TOM[i]_TGC1_GLB_CTRL

Controls channel 8 to 15
For description see 11.8.1

11.8.10 Register TOM[i]_TGC1_ENDIS_CTRL

Controls channel 8 to 15
For description see 11.8.2

11.8.11 Register TOM[i]_TGC1_ENDIS_STAT

Controls channel 8 to 15
For description see 11.8.3

11.8.12 Register TOM[i]_TGC1_ACT_TB

Controls channel 8 to 15
For description see 11.8.4

11.8.13 Register TOM[i]_TGC1_OUTEN_CTRL

Controls channel 8 to 15
For description see 11.8.5

11.8.14 Register TOM[i]_TGC1_OUTEN_STAT

Controls channel 8 to 15
For description see 11.8.6

11.8.15 Register TOM[i]_TGC1_FUPD_CTRL

Controls channel 8 to 15
For description see 11.8.7

11.8.16 Register TOM[i]_TGC1_INT_TRIG

Controls channel 8 to 15
For description see 11.8.8

11.8.17 Register TOM[i]_CH[x]_CTRL (x:0...14)

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	Reserved		GCM	SPEM	Reserved	OSM	Reserved	TRIGOUT	Reserved			RST_CCU0	Reserved					CLK_SRC_SR			SL	Reserved										
Mode	R		RW	RW	R	RW	R	RW	R			RW	R					RW			RW	R										
Initial Value	0x0		0	0	0	0	0	0	000			0	00000					000			X	0x000										

Bit 10:0 **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).
000 = *CMU_FXCLK*(0) selected: clock selected by FXCLK_SEL
001 = *CMU_FXCLK*(1) selected: clock selected by FXCLK_SEL/ 2^4
010 = *CMU_FXCLK*(2) selected: clock selected by FXCLK_SEL/ 2^8
011 = *CMU_FXCLK*(3) selected: clock selected by FXCLK_SEL/ 2^12
100 = *CMU_FXCLK*(4) selected: clock selected by FXCLK_SEL/ 2^16
101 = no *CMU_FXCLK* selected, clock of channel stopped
110 = no *CMU_FXCLK* selected, clock of channel stopped
111 = no *CMU_FXCLK* selected, clock of channel stopped

Note: if clock of channel is stopped (i.e. CLK_SRC = 101/110/111) , the channel can only be restarted by resetting CLK_SRC_SR to a value of 000 to 100 and forcing an update via the force update mechanism.

Bit 19:15 **Reserved**
Note: Read as zero, should be written as zero

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]*

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.

Bit 23:21	**Reserved** Note: Read as zero, should be written as zero
Bit 24	**TRIGOUT**: Trigger output selection (output signal *TRIG_[x]*) of module TOM_CH[x] 0 = *TRIG_[x]* is *TRIG_[x-1]* 1 = *TRIG_[x]* is *TRIG_CCU0*
Bit 25	**Reserved** Note: Read as zero, should be written as zero
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	**Reserved**
Bit 28	**SPEM**: SPE mode enable for channel. 0 = SPE mode disabled 1 = SPE mode enabled Note:The SPE mode is only implemented for TOM instances connected to a SPE module and only for channels 0 to 7. Note: On TOM channel 2 SPEM=1 has special meaning. If SPEM = 1, the signal *SPE_NIPD* triggers the reset of CN0. If SPEM = 1 and OSM=1,the signal *SPE_NIPD* triggers the start of single pulse generation
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 31:30	**Reserved** Note: Read as zero, should be written as zero

11.8.18 Register TOM[i]_CH15_CTRL

Address Offset:	**see Appendix B**	**Initial Value:** 0x0000_0X00

	31:28	27	26	25	24	23:21	20	19:15	14:12	11	10:0
Bit	Reserved	BITREV	OSM	Reserved	TRIGOUT	Reserved	RST_CCU0	Reserved	CLK_SRC_SR	SL	Reserved
Mode	R	RW	RW	R	RW	R	RW	R	RW	RW	R
Initial Value	0x0	0	0	0	0	000	0	00000	000	X	0x000

Bit 10:0 **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 ATOM_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).
000 = *CMU_FXCLK*(0) selected: clock selected by FXCLK_SEL
001 = *CMU_FXCLK*(1) selected: clock selected by FXCLK_SEL/ 2^4
010 = *CMU_FXCLK*(2) selected: clock selected by FXCLK_SEL/ 2^8
011 = *CMU_FXCLK*(3) selected: clock selected by FXCLK_SEL/ 2^12
100 = *CMU_FXCLK*(4) selected: clock selected by FXCLK_SEL/ 2^16
101 = no *CMU_FXCLK* selected, clock of channel stopped
110 = no *CMU_FXCLK* selected, clock of channel stopped
111 = no *CMU_FXCLK* selected, clock of channel stopped

Note: if clock of channel is stopped (i.e. CLK_SRC = 101/110/111) , the channel can only be restarted by resetting CLK_SRC_SR to a value of 000 to 100 and forcing an update via the force update mechanism.

Bit 19:15 **Reserved**
Note: Read as zero, should be written as zero

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_14*

Bit 23:21	**Reserved** Note: Read as zero, should be written as zero
Bit 24	**TRIGOUT**: Trigger output selection (output signal *TRIG_15*) of module TOM_CH15 0 = *TRIG_15* is *TRIG_14* 1 = *TRIG_15* is *TRIG_CCU0*
Bit 25	**Reserved** Note: Read as zero, should be written as zero
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**. This bit enables the PCM mode of channel 15
Bit 31:28	**Reserved** Note: Read as zero, should be written as zero

11.8.19 Register TOM[i]_CH[x]_CN0 (x:0...15)

Address Offset:	**see Appendix B**																**Initial Value:**						**0x0000_0000**									
	31	30	29	28	27	26	25	24	23	22	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

11.8.20 Register TOM[i]_CH[x]_CM0 (x:0...15)

Address Offset:	**see Appendix B**	**Initial Value:** 0x0000_0000
	31 30 29 28 27 26 25 24 23 22 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

11.8.21 Register TOM[i]_CH[x]_SR0 (x:0...15)

Address Offset:	**see Appendix B**	**Initial Value:** 0x0000_0000
	31 30 29 28 27 26 25 24 23 22 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	0x0000	0x0000

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

11.8.22 Register TOM[i]_CH[x]_CM1 (x:0...15)

Address Offset:	**see Appendix B**	**Initial Value: 0x0000_0000**
	31 30 29 28 27 26 25 24 23 22 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	RW
Initial Value	0x0000	0x0000

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

11.8.23 Register TOM[i]_CH[x]_SR1 (x:0...15)

Address Offset:	**see Appendix B**	**Initial Value: 0x0000_0000**
	31 30 29 28 27 26 25 24 23 22 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

11.8.24 Register TOM[i]_CH[x]_STAT (x:0...15)

Address Offset:	**see Appendix B**	**Initial Value:**	**0x0000_000x**
	31 … 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

11.8.25 Register TOM[i]_CH[x]_IRQ_NOTIFY (x:0...15)

Address Offset:	**see Appendix B**	**Initial Value:**	**0x0000_0000**	
	31 … 2		1	0
Bit	Reserved		CCU1TC	CCU0TC
Mode	R		RCw	RCw
Initial Value	0x0000 000		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 re-trigger the notification first the condition CN0 < CM0 has to be occurred.

Bit 1 **CCU1TC**: CCU1 Trigger condition interrupt for channel x
0 = No interrupt occurred
1 = The condition CN0 >= CM1 was detected.

The notification of the interrupt is only triggered one time after reaching the condition CN0 >= CM1. To re-trigger the notification first the condition CN0 < CM1 has to be occurred.

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

11.8.26 Register TOM[i]_CH[x]_IRQ_EN (x:0...15)

Address Offset:	**see Appendix B**	**Initial Value:**	**0x0000_0000**	
	31 … 2		1	0
Bit	Reserved		CCU1TC_IRQ_EN	CCU0TC_IRQ_EN
Mode	R		RW	RW
Initial Value	0x0000000		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

11.8.27 Register TOM[i]_CH[x]_IRQ_FORCINT (x:0...15)

Address Offset:	**see Appendix B**	**Initial Value:**	**0x0000_0000**	
	31 … 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

11.8.28 Register TOM[i]_CH[x]_IRQ_MODE (x:0...15)

Address Offset:	**see Appendix B**	**Initial Value:**	**0x0000_000X**
	31 … 2		1 0
Bit	Reserved		IRQ_MODE
Mode	R		RW
Initial Value	0x0000 0000		XX

Bit 1:0 **IRQ_MODE**: IRQ mode selection
00 = Level mode

	01 = Pulse mode
	10 = Pulse-Notify mode
	11 = 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

12 ARU-connected Timer Output Module (ATOM)

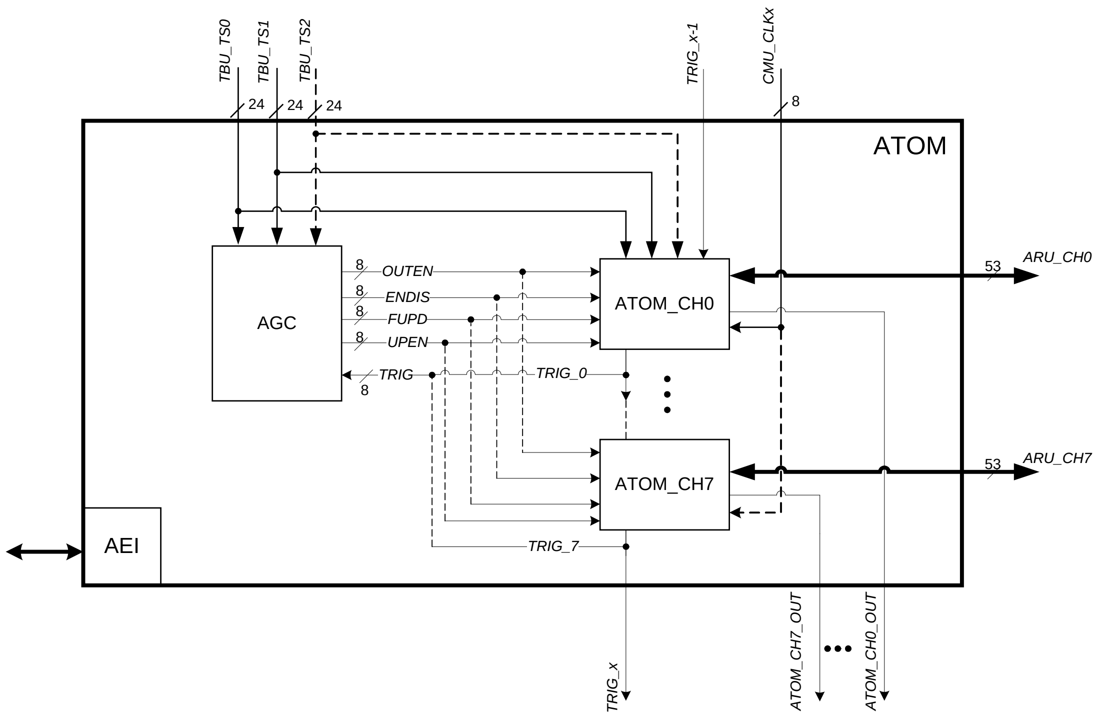
12.1 Overview

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 submodules connected to ARU like e.g. the MCS, DPLL or PSM. Each ATOM submodule contains eight output channels which can operate independently from each other in several configurable operation modes. A block diagram of the ATOM submodule is depicted in figure 12.1.1.

The following design variables are used inside this chapter. Please refer to device specific Appenedix B for correct value.

cCATO : ATOM channel count; number of channels per instance - 1

12.1.1 ATOM Block diagram



The architecture of the ATOM submodule is similar to the TOM submodule, but there are some differences. First, the ATOM integrates only eight output channels. Hence, there exists one ATOM Global Control subunit (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 four 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)

These modes are described in more detail in section 12.3.
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 submodule. 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 if both shadow registers are updated via ARU or only one of the shadow registers is updated for SOMP mode.

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 behaviour of the compare units CCU0 and CCU1 and the output signal behaviour is controlled with the ACB bit field inside the **ATOM[i]_CH[x]_CTRL** register when the ARU connection is disabled and the behaviour 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.

12.1.2 ATOM Global control (AGC)

Synchronous start and stop of more then one output channel is possible with the AGC subunit. This subunit has the same functionality as the TGC subunit of the TOM submodule. For a description of the AGC subunit functionality, please refer therefore to chapter 11.2.

12.1.3 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.

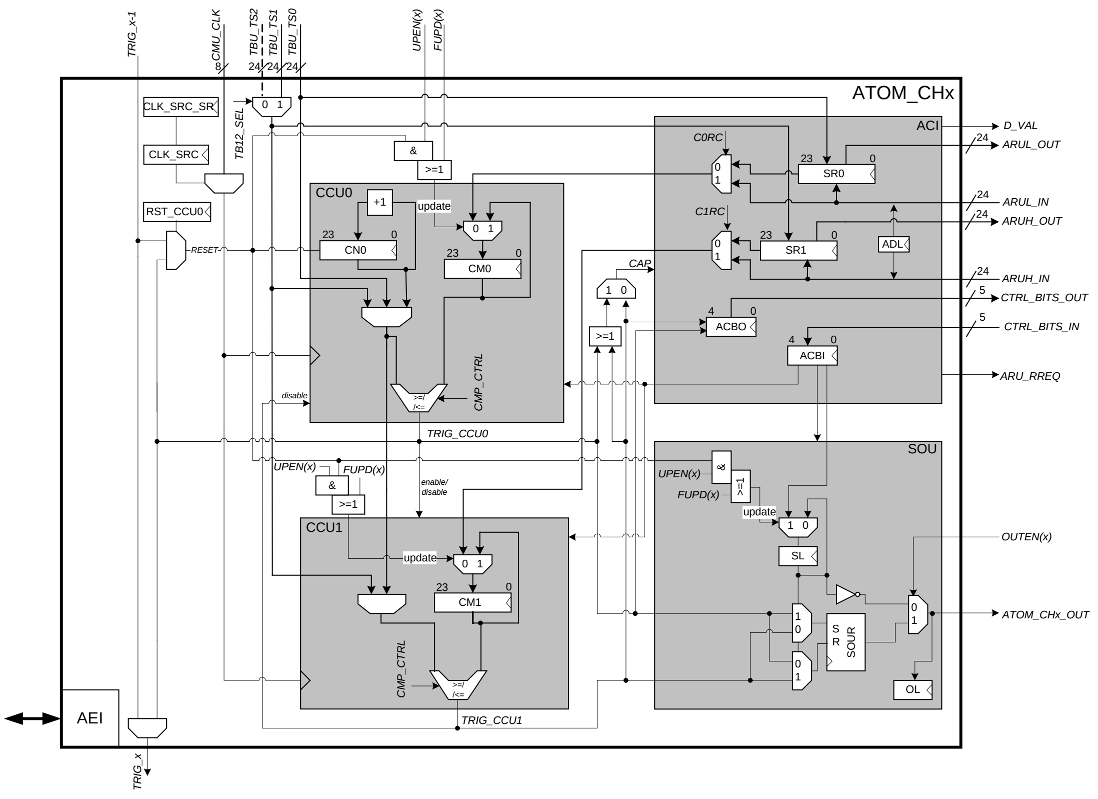
In ATOM Signal Output Mode PWM (SOMP), the ATOM channel is able to generate simple and complex PWM output signals like the TOM submodule by comparing its operation registers with a submodule internal counter. In difference to the TOM, the ATOM shadow registers can be reloaded by the CPU *and* by the 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 12.3.4 for further details.

12.2 ATOM Channel architecture

Each ATOM channel is able to generate output signals according to four 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 12.2.1.

12.2.1 ATOM Channel architecture



For all ATOM channels the bit width is 24 of the operation register **CN0**, **CM0** and **CM1** and the shadow register **SR0** and **SR1**. 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* and *TBU_TS1*. If there is a third time base *TBU_TS2* implemented inside the GTM, this time base can also be selected inside the ATOM channel with the TB12_SEL bit inside the **ATOM[i]_CH[x]_CTRL** register for comparison. Please refer to TBU chapter 9 for further details. For an overview of the implemented TBU submodule version please refer to chapter 2.1.1. The CCU0 and CCU1 units have different tasks for the different ATOM channel modes.

The signed 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 behaviour of this signed 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 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 12.3.2 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 behaviour of the ATOM channel.

When the channel and its output is disabled, the output signal level of the channel is the inverse of the SL bit.

In SOMI and SOMC 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) subunit is responsible for requesting data routed through ARU to the ATOM channel in SOMI, SOMP and SOMS modes, and additionally for providing data to the ARU in SOMC mode.

In SOMC mode the ACI shadow registers have a different behaviour and are used as output buffer registers for data send to ARU.

12.2.2 ARU Communication Interface

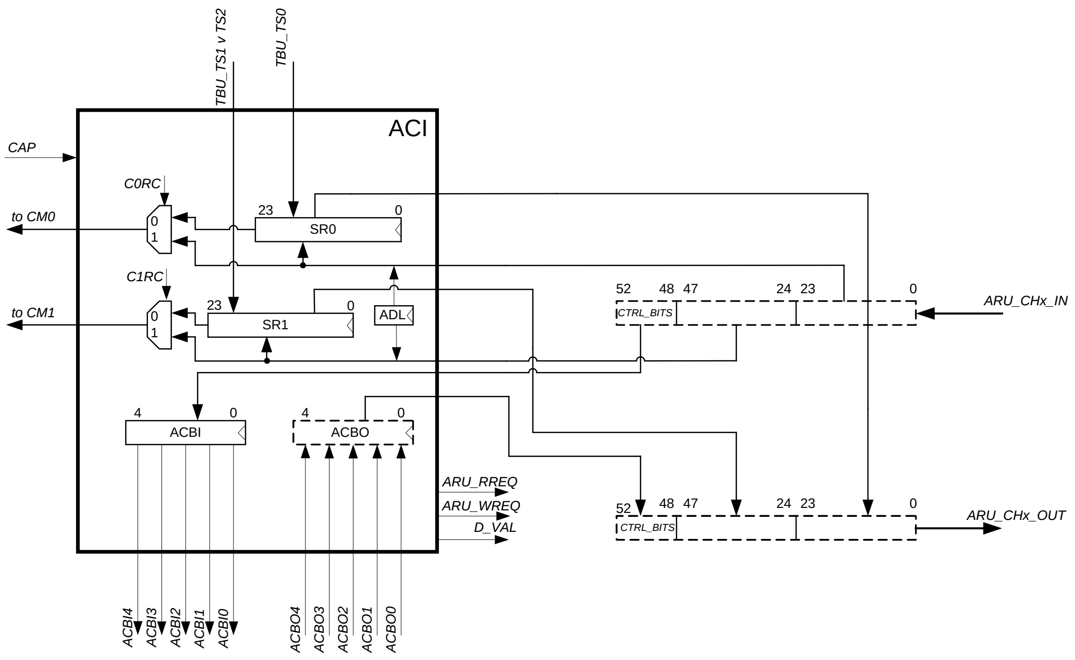
The ATOM channels have an ARU Communication Interface (ACI) subunit. This subunit is responsible for data exchange from and to the ARU. This is done with the two implemented registers **SR0**, **SR1**, and the ACBI and ACBO bit fields that are part of the **ATOM[i]_CH[x]_STAT** register. The ACI architecture is shown in figure 12.2.2.1.

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 and SOMC 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.

12.2.2.1 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 and SOMS 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 and the bits 52 downto 48 (*CTRL_BITS*) are stored in the **ACBI** bit field of the register **ATOM[i]_CH[x]_STAT**.

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 behaviour 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 signalled by the CCU0 and/or CCU1 subunits 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.

12.3 ATOM Channel modes

As described above, each ATOM channel can operate independently from each other in one of four 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)

The Signal Output Mode PWM (SOMP) is principally the same like the output mode for the TOM submodule except the bit reverse mode which is not included in the ATOM. In addition, it is possible to reload the shadow registers over the ARU without the need of a CPU interaction. The three 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.

12.3.1 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 subunit 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 11.8.5) for details.

In SOMI mode the output behaviour 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**:

SL	**ACBI(0)/ ACB(0)**	**Output behaviour**
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 subunit 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.

12.3.1.1 Register ATOM[i]_CH[x]_CTRL in SOMI mode (x: 0...7)

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	Reserved					Not used	Reserved	Not used	Not used			Not used	Reserved			Not used	Reserved	Not used			SL	Reserved		Not used				ACB(0)	ARU_EN	Not used	MODE	
Mode	R					RW	R	RW	RW			RW	R			RW	R	RW			RW	R		RW				RW	RW	RW	RW	
Initial Value	0					0	0	0	0			0	0			0	0	0			x	0		0				0	0	0	0	

Bit 1:0	**MODE**: ATOM channel mode select. 00: ATOM Signal Output Mode Immediate (SOMI)
Bit 2	**Not used**: Not used in this mode 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**: Not used in this mode Note: Not used in this mode.
Bit 10:9	**Reserved** Note: Read as zero, should be written as zero.
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: After reset and if channel is disabled, the register **SOUR** is set to the inverse reset value of bit SL (i.e. '1'). If the channel is disabled or the output is disabled, the output ATOM_OUT[x] is set to inverse value of SL.
Bit 14:12	**Not used**: Not used in this mode Note: Not used this mode.
Bit 15	**Reserved** Note: Read as zero, should be written as zero.
Bit 16	**Not used**: Not used in this mode Note: Not used this mode.
Bit 19:17	**Reserved** Note: Read as zero, should be written as zero.
Bit 20	**Not used**: Not used in this mode Note: Not used this mode.

Bit 23:21	**Not used**: Not used in this mode Note: Not used this mode.
Bit 24	**Not used**: Not used in this mode Note: Not used this mode.
Bit 25	**Reserved** Note: Read as zero, should be written as zero.
Bit 26	**Not used**: Not used in this mode Note: Not used this mode.
Bit 31:27	**Reserved** Note: Read as zero, should be written as zero.

12.3.2 ATOM Signal Output Mode Compare (SOMC)

12.3.2.1 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 9. It is configurable, which of the two (three) time bases is to be compared with one or both values in **CM0** and **CM1**.

The behaviour 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 can be enabled again 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 12.3.2.2 and 12.3.2.3.

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.

12.3.2.2 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 behaviour according to the ACB10 bit field in the control register is shown in the following table:

SL	**ACB10(5)**	**ACB10(4)**	**Output behaviour**
0	0	0	No signal level change at output (exception in table 12.3.2.2.2 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 12.3.2.2.2 mode ACB42=001)
1	0	0	No signal level change at output (exception in table 12.3.2.2.2 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 12.3.2.2.2 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:

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 12.3.2.2.2
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 12.3.2.2.2
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	Not used when ARU disabled.

The behaviour of the ACBI/ACB42 bit combinations '000' and '001' is described in more detail in tables 12.3.2.2.1 and 12.3.2.2.2.

12.3.2.2.1 ATOM CCUx Serve first definition ACB42 = '000'

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

12.3.2.2.2 ATOM CCUx Serve first definition ACB42 = '001'

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 signalled 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.

In SOMC mode and CCUx control mode 'serve last' exist an exception for update of register **CM0**/**CM1**. 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="100" or ACB42="101") 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:

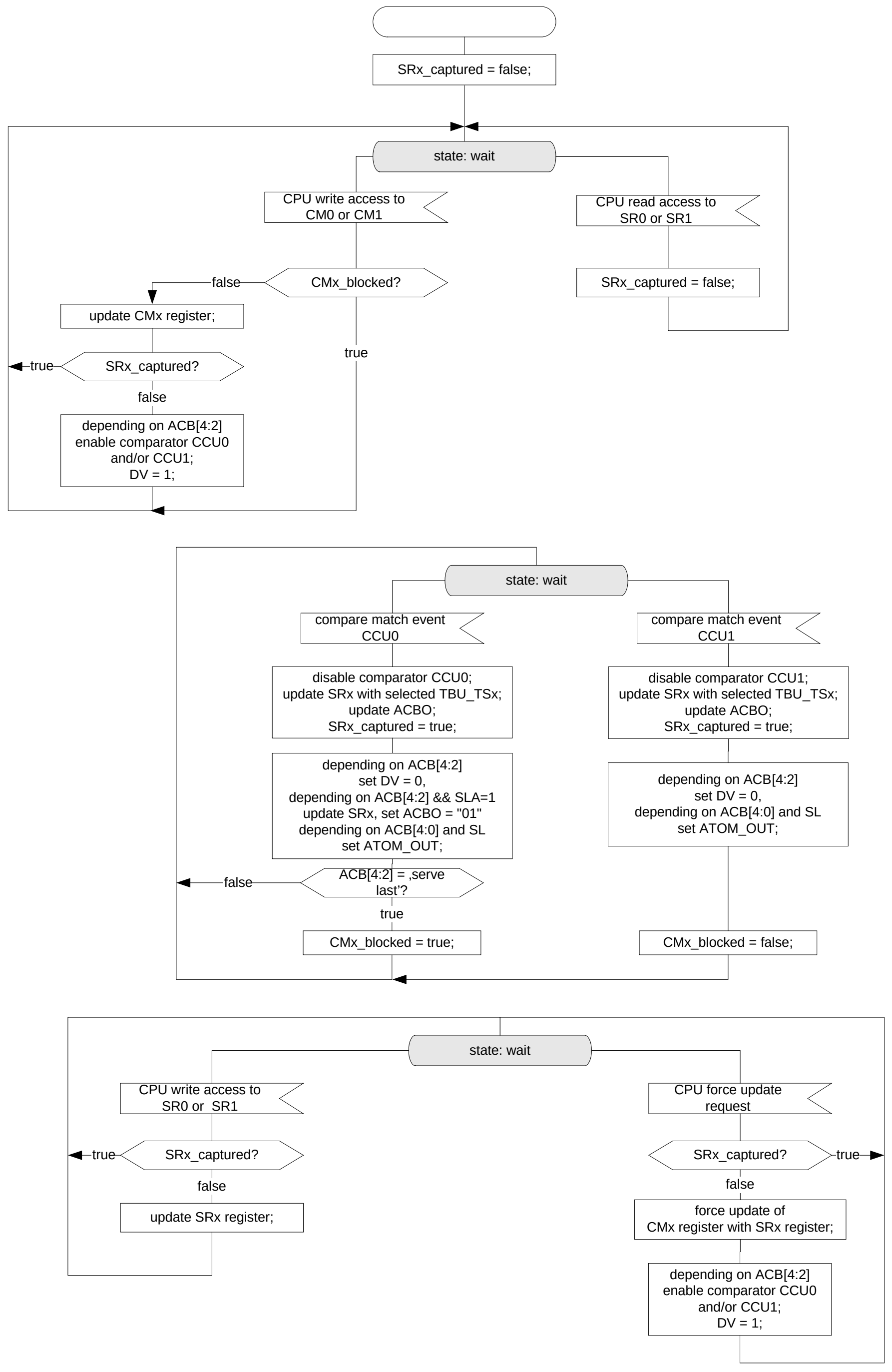
ACBO(4)	**ACBO(3)**	**Indication**
0	1	CCU0 compare match occurred
1	0	CCU1 compare match occurred

Please note, that in case of the 'serve last' compare strategy, when the SLA-bit 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.

The behaviour of an ATOM channel in SOMC mode under CPU control is visualized in figure 12.3.2.2.3.

12.3.2.2.3 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
SRx_captured?
true
false
update SRx register;
force update of CMx register with SRx register;
depending on ACB[4:2] enable comparator CCU0 and/or CCU1; DV = 1;

12.3.2.3 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 behaviour 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 behaviour 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:

SL	**ACBI(1)**	**ACBI(0)**	**Output behaviour**
0	0	0	No signal level change at output (exception in tables 12.3.2.2.1 and 12.3.2.2.2 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 12.3.2.2.1 and 12.3.2.2.2 mode ACB42=001)
1	0	0	No signal level change at output (exception in table's 12.3.2.2.1 and 12.3.2.2.2 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 12.3.2.2.1 and 12.3.2.2.2 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:

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 12.3.2.2.2
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 12.3.2.2.1
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 "111" 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:

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 SLA-bit 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.

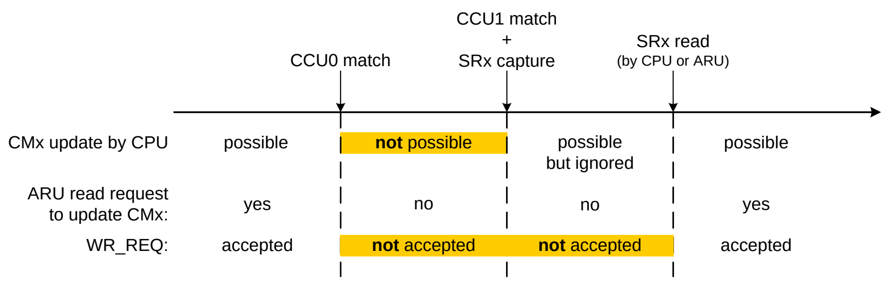
Depending on the capture compare unit where the time base matched the interrupt *CCU0TCx_IRQ* or *CCU1TCx_IRQ* is raised.

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 sequently. 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.

12.3.2.3.1 CPU access rights in case of compare strategy 'serve last'



12.3.2.3.2 ARU Non-Blocking mode

When the compare registers are updated via ARU the update behaviour 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 this 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 has no effect.

To set up a new compare action, first the **SR0**/**SR1** registers 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.

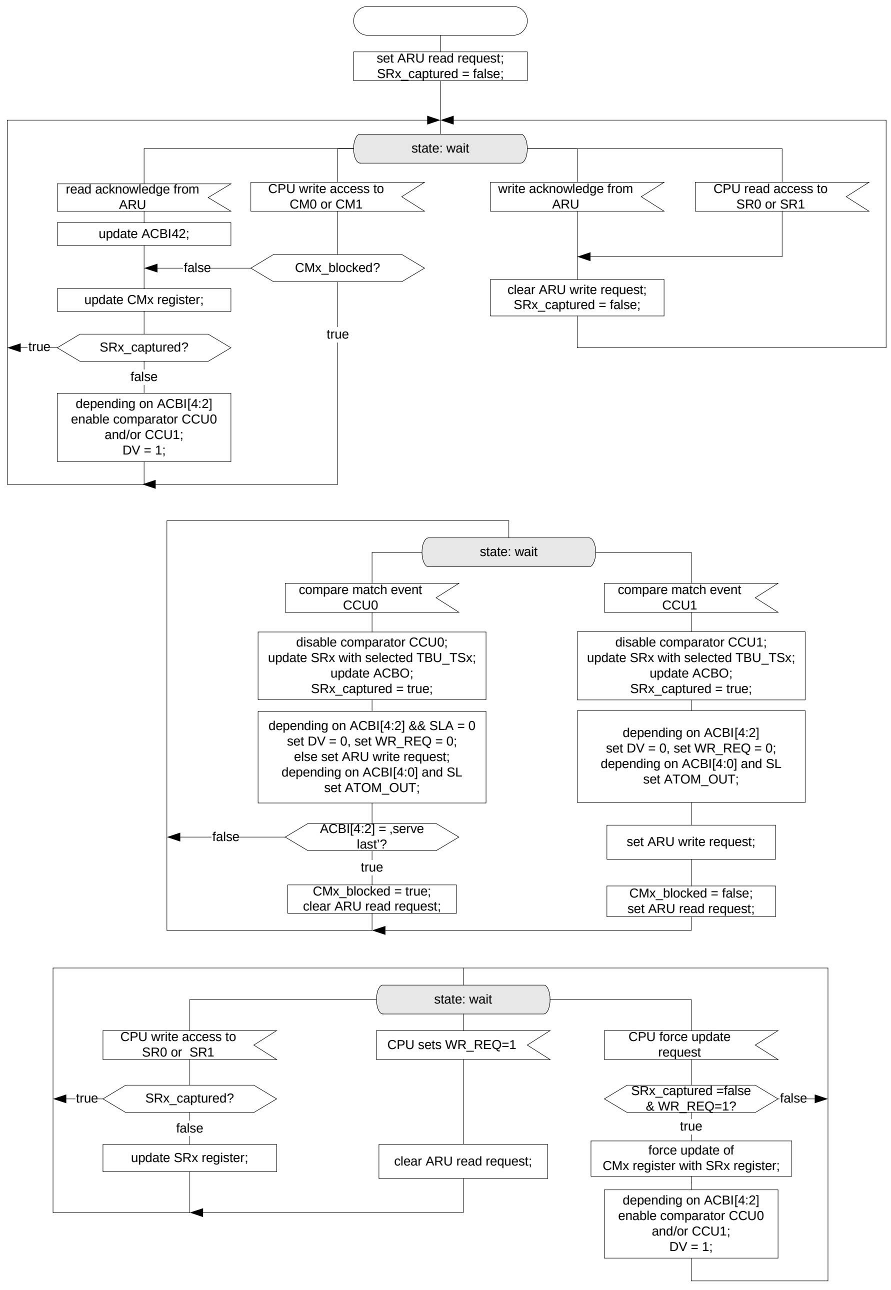
When the CPU does the register accesses, only one of the shadow registers has to be read. Dependent on the compare strategy, the CPU has to write one or both of the compare registers.

In SOMC mode and CCUx control mode 'serve last' exist an exception for update of register **CM0**/**CM1**. If in this mode the CCU0 compare match event occurred, the update of register **CM0**/**CM1** via CPU or ARU is blocked until the CCU1 compare match event occurs.

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 signalled by the DV bit inside the **ATOM[i]_CH[x]_STAT** register.

The behaviour of an ATOM channel in SOMC mode, when ARU is enabled and ARU blocking mode is disabled is shown in figure 12.3.2.3.2.1.

12.3.2.3.2.1 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?
false
true
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?
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;

12.3.2.3.3 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, the (one) two compare values for **CM0** and/or **CM1** can be provided by ARU or CPU. If the compare registers **CM0** and/or **CM1** are/is updated, the ATOM channel waits for the compare match event to happen. No further data is requested from the ARU.

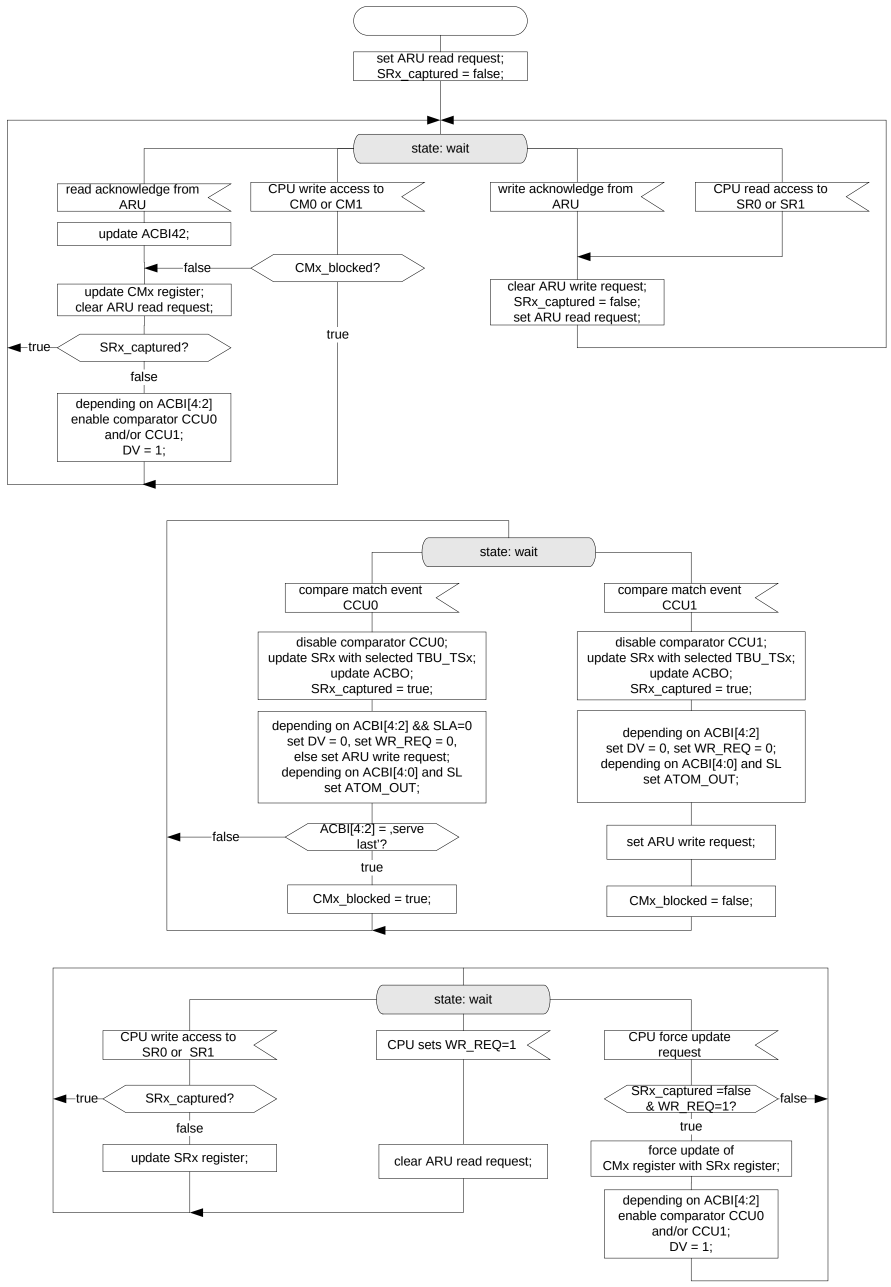
When the specified compare match event happens, the shadow registers **SR0** and **SR1** are updated together with the ACBO bits in the **ATOM[i]_CH[x]_STAT** register. The data in the shadow registers is marked as valid for the ARU and the DV bit is reset inside the **ATOM[i]_CH[x]_CTRL** register.

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, before new data is requested from ARU.
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 signalled by a set DV bit inside the **ATOM[i]_CH[x]_STAT** register.

The behaviour of an ATOM channel in SOMC mode, when ARU is enabled and ARU blocking mode is enabled is shown in figure 12.3.2.3.3.1.

12.3.2.3.3.1 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;
true
SRx_captured?
false
depending on ACBI[4:2] enable comparator CCU0 and/or CCU1; DV = 1;
true
clear ARU write request;
SRx_captured = false;
set ARU read request;
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;
CMx_blocked = false;
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;

12.3.2.3.4 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.

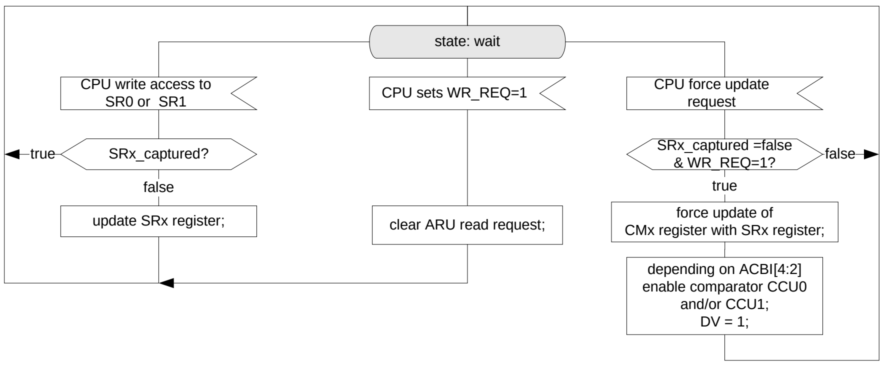
If a compare match event already happened, the WR_REQ bit could not be set until the channel is unlocked for a new compare match event by reading the shadow registers. In addition, the WRF bit is set if the CPU tries to write the WR_REQ bit in that case.

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 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.
A blocked force update mechanism will be enabled again after a read access to the register **SR0** or **SR1** by either the ARU or the CPU.

The ATOM SOMC late update mechanism from CPU is shown in figure 12.3.2.3.4.1.

12.3.2.3.4.1 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?
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;

12.3.2.4 Register ATOM[i]_CH[x]_CTRL in SOMC mode (x: 0...7)

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	Reserved				ABM	Not used	SLA	TRIGOUT	Not used			Not used	Reserved			WR_REQ	Reserved	Not used			SL	Reserved	CMP_CTRL	ACB42			ACB10		ARU_EN	TB12_SEL	MODE	
Mode	R				RW	RW	RW	RW	RW			RW	R			RW	R	RW			RW	R	RW	RW			RW		RW	RW	RW	
Initial Value	0				0	0	0	0	0			0	0			0	0	0			x	0	0	0			00		0	0	00	

Bit 1:0	**MODE**: ATOM channel mode select. 01: 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. 00: No signal level change at output (exception in tables 12.3.2.2.1 and 12.3.2.2.2 mode ACB42=001). 01: Set output signal level to 1 when SL bit = 0 else output signal level to 0. 10: Set output signal level to 0 when SL bit = 0 else output signal level to 1. 11: Toggle output signal level (exception in tables 12.3.2.2.1 and 12.3.2.2.2 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) 000: 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. 001: 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. 010: Compare in CCU0 only against *TBU_TS0*. 011: Compare in CCU1 only against *TBU_TS1* or *TBU_TS2*. 100: Compare first in CCU0 and then in CCU1. Use *TBU_TS0*. 101: Compare first in CCU0 and then in CCU1. Use *TBU_TS1* or *TBU_TS2*.

110: Compare first in CCU0 and then in CCU1. Use *TBU_TS0* in CCU0 and *TBU_TS1* or *TBU_TS2* in CCU1.
111: Reserved.
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 ACBI(4:2) or ACB42) always performs a greater/equal comparison, independent on CMP_CTRL bit.

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

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.
Note: If the channel and output are disabled, in MODE=01 (SOMC mode) the output register of SOU unit is set to value of SL. If the output is enabled afterwards, the output ATOM_OUT[x] is equal to the value of SL.

Bit 14:12 **Not used**: Not used in this mode
Note: Not used in this mode.

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

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 19:17 **Reserved**
Note: Read as zero, should be written as zero.

Bit 20 **Not used** : not used in this mode
Note: Not used in this mode.

Bit 23:21 **Not used**: Not used in this mode
Note: Not used this mode.

Bit 24 **TRIGOUT**: Trigger output selection (output signal *TRIG_CHx*) of module ATOM_CHx.
0 = *TRIG_[x]* is *TRIG_[x-1]*
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 independent of pending compare match event
1 = ARU blocking mode enabled: after updating CM0,CM1 via ARU, no new data is read from ARU until compare match event occurred and SR0 and/or SR1 are read.

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

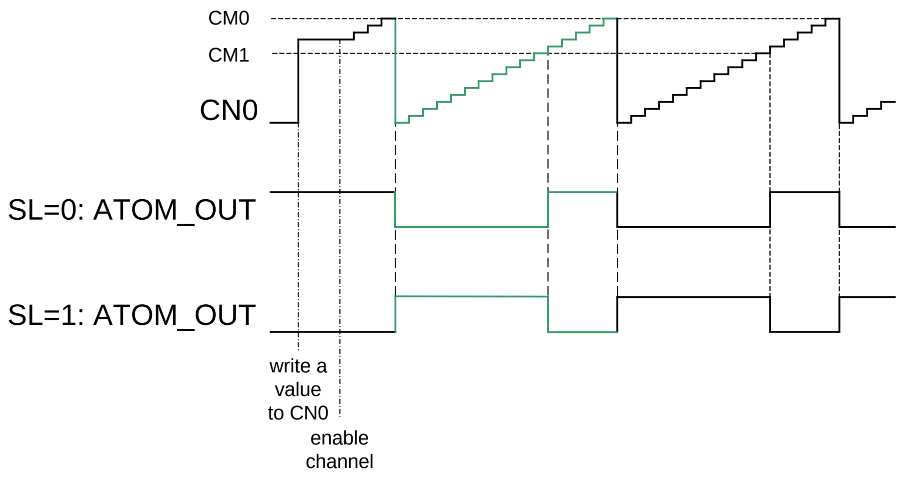
12.3.3 ATOM Signal Output Mode PWM (SOMP)

In ATOM Signal Output Mode PWM (SOMP) the ATOM submodule 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 reverse pulse level defined by the SL bit. Figure 12.3.3.1 clarifies this behaviour.

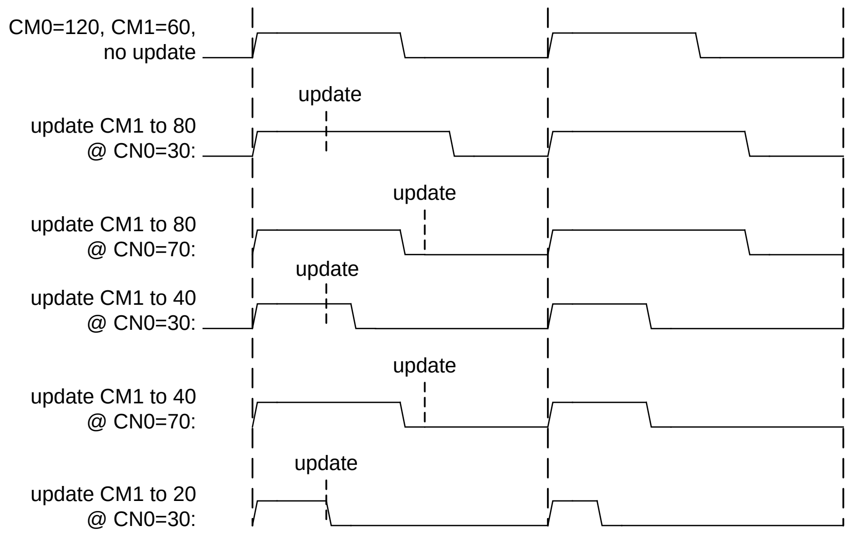
In SOMP mode, depending on configuration bits RST_CCU0 of register **ATOM[i]_CH[x]_CTRL** the counter register **CN0** can be reset either when the counter value is equal to the compare value **CM0** or when signaled by the ATOM[i] trigger signal *TRIG_[x-1]* of the preceding channel.
In this case, if UPEN_CTRL[x]=1, also the working register CM0, CM1 and CLK_SRC are updated.

12.3.3.1 PWM Output behaviour with respect to the SL bit in the ATOM[i]_CH[x]_CTRL register



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 behaviour of the output signal due to the different possibilities of an asynchronous update during a PWM period is shown in figure 12.3.3.2.

12.3.3.2 PWM Output behaviour in case of an asynchronous update of the duty cycle



The duration of the pulse high or low time and period is measured with the counter in subunit 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** configured by RST_CCU0=0), following statements are valid:
- the configuration of **CM1**=0 represents 0% duty cycle at the output,
- the configuration of **CM1** >= **CM0** represents 100% duty cycle
- if both registers are configured to 0 (**CM0**=**CM1**=0), the output is 0% duty cycle
- if **CM0**=0, 0% duty cylce is generated independent of **CM1**.

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:
- **CM0** defines the edge to SL value, **CM1** defines the edge to !SL value.
- if **CM0**=**CM1**, the output is 100% SL (**CM0** has higher priority)
- if **CM0**=0, the output stays at its last value (**CN0** stops counting)

In case the counter value **CN0** reaches the compare value in register **CM0** 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.

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 subunit from the ARU, while the ATOM channel operates on the **CM0** and **CM1** registers.

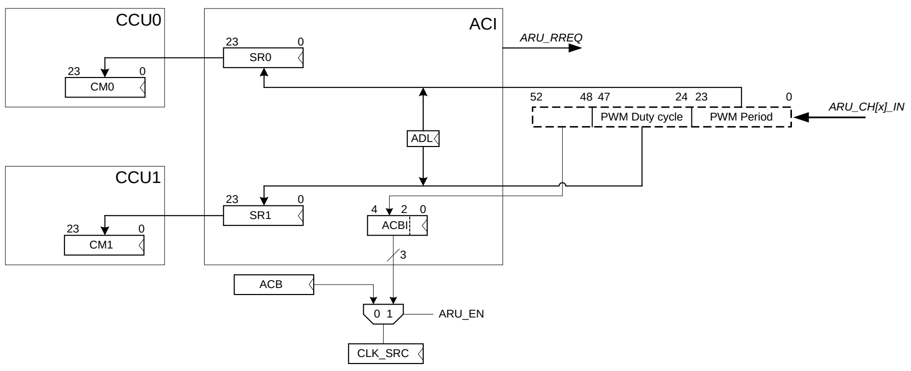
This internal update mechanism is established, when the old PWM period ends. The shadow registers are loaded into the operation registers, the counter register is reset, the new clock source according to the CLK_SRC_SR or ACBI(4), ACBI(3) and ACBI(2) bits is selected and the new PWM generation starts.

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 12.3.3.3.

12.3.3.3 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.

The shadow registers **SR0** and **SR1** can also be updated over the AEI bus interface. 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 subunit. 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*.

12.3.3.4 SOMP One-shot 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** back to zero, the first edge at *ATOM[i]_CH[x]_OUT* is generated.

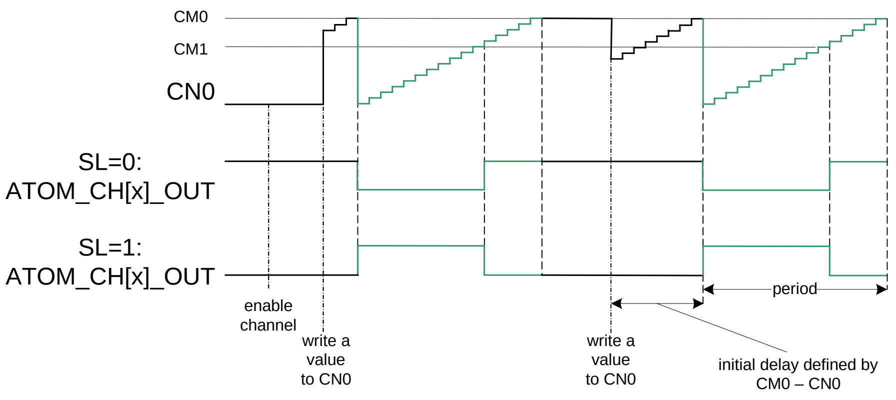
To avoid an update of **CMx** register with content of **SRx** register at this point in time, the automatic update should be disabled by setting UPEN_CTRL[x] = 00 (in register **ATOM[i]_CH[x]_CTRL**)

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** 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 12.3.3.4.1 clarifies the pulse generation in SOMP One-shot mode.

12.3.3.4.1 PWM Output with respect to configuration bit SL in One-shot mode: trigger by wrting 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 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.

12.3.3.5 Register ATOM[i]_CH[x]_CTRL in SOMP mode (x: 0...7)

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	Reserved				Not used	OSM	Reserved	TRIGOUT	Not used			RST_CCU0	Reserved			Not used	Reserved	CLK_SRC_SR			SL	Reserved		Not used			ADL		ARU_EN	Not used	MODE	
Mode	R				RW	RW	R	RW	RW			RW	R			RW	R	RW			RW	R		RW			RW		RW	RW	RW	
Initial Value	0				0	0	0	0	0			0	0			0	0	0			x	0					0		0	0	0	

Bit 1:0 **MODE**: ATOM channel mode select.
10: 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.
00: Load both ARU words into shadow registers
01: Load ARU low word (Bits 23..0) into shadow register SR0
10: Load ARU high word (Bits 47..24) into shadow register SR1
11: Reserved
Note: This bit field is only used in SOMP mode to select the ARU data source.

Bit 8:6 **Not used**
Note: Not used in this mode.

Bit 10:9 **Reserved**
Note: Read as zero, should be written as zero.

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.
If the output is disabled, the output ATOM_OUT[x] is set to inverse value of SL.

Bit 14:12	**CLK_SRC_SR**: Shadow register for CMU clock source register CLK_SRC 000: *CMU_CLK0* selected 001: *CMU_CLK1* selected 010: *CMU_CLK2* selected 011: *CMU_CLK3* selected 100: *CMU_CLK4* selected 101: *CMU_CLK5* selected 110: *CMU_CLK6* selected 111: *CMU_CLK7* selected Note: This register is a shadow register for the *CMU_CLKx* select. 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 one of the CMU_CLKx, it is required to perform a forced update of CLK_SRC with the value of CLK_SRC_SR value before/with enabling the channel. Note: These bits are only applicable if ARU_EN = '0'.
Bit 15	**Reserved** Note: Read as zero, should be written as zero.
Bit 16	**Not used** Note: Not used in this mode.
Bit 19:17	**Reserved** Note: Read as zero, should be written as zero.
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]* Note: If RST_CCU0=1 and UPEN_CTRLx=1 are set, *TRIG_[x-1]* triggers also the update of work register (**CM0**, **CM1** and **CLK_SRC**).
Bit 23:21	**Not used**: Not used in this mode Note: Not used this mode.
Bit 24	**TRIGOUT**: Trigger output selection (output signal *TRIG_CHx*) of module ATOM_CHx. 0 = *TRIG_[x[* is *TRIG_[x-1*] 1 = *TRIG_[x]* is *TRIG_CCU0*
Bit 25	**Reserved** Note: Read as zero, should be written as zero.
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 31:28	**Reserved**
	Note: Read as zero, should be written as zero.

12.3.4 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.

12.3.4.1 SOMS Mode output generation

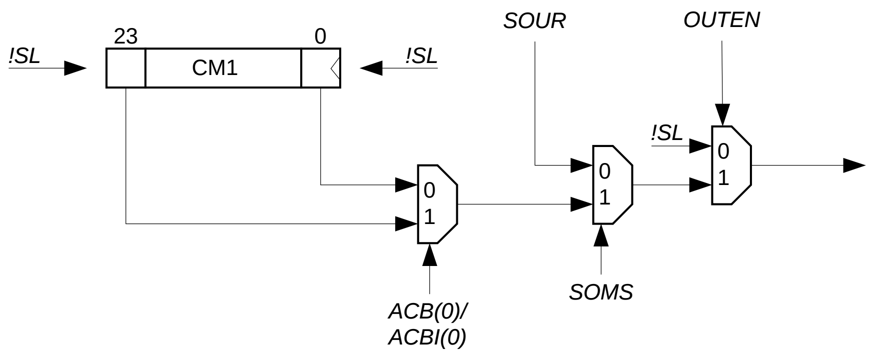
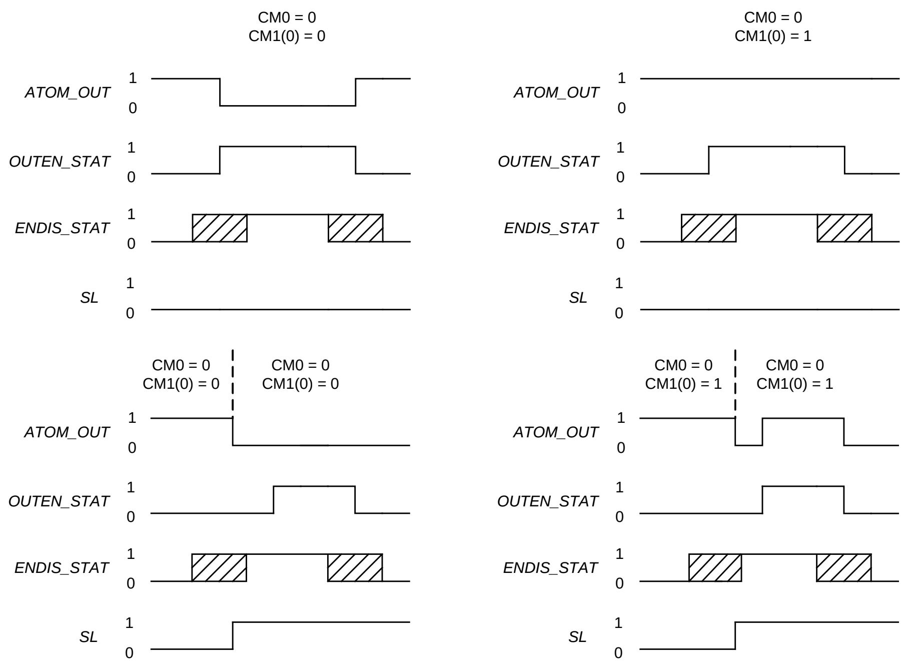


Figure 12.3.4.1 shows the output generation in case of SOMS mode is selected.
In SOMS mode CCU0 runs in counter/compare mode and counts the number of bits shifted out so far. The total number of bits that should be shifted is defined as **CM0**. The total number of bits that are visible at ATOM_OUT is **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.

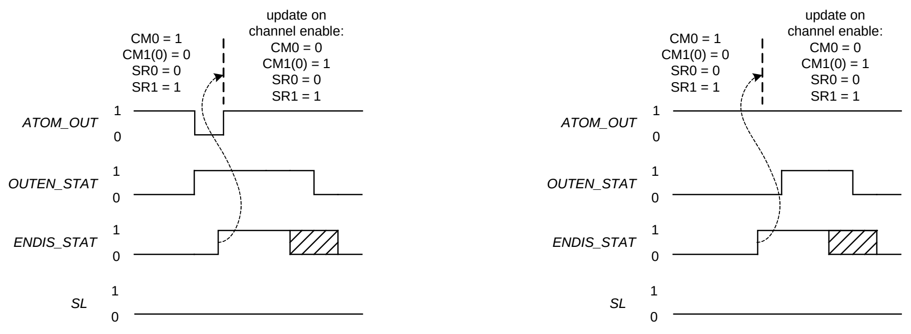
When the output is enabled while **UPEN_CTRL[x]** is disabled, the ATOM_OUT signal level is defined by **CM1** bit 0 or 23, dependent on the shift direction defined by ACB(0) or ACBI(0) register setting. Figure 12.3.4.2 should clarify the ATOM channel start-up behaviour in this case for right shift. For left shift the **CM1** bit 0 in 12.3.4.2 has to be replaced by **CM1** bit 23.

12.3.4.2 SOMS Output signal level at start-up, 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 12.3.4.3 shows the output behaviour in that case.

12.3.4.3 SOMS Output signal level at start-up, 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 (SL0 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**.

12.3.4.4 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.

12.3.4.5 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**.

12.3.4.6 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.

12.3.4.7 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 (0) 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.

12.3.4.8 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.

12.3.4.9 Register ATOM[i]_CH[x]_CTRL in SOMS mode (x: 0...7)

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	Reserved				Not used	OSM	Reserved	Not used	Not used			Not used	Reserved			Not used	Reserved	CLK_SRC_SR			SL	Reserved		Not used				ACB0	ARU_EN	Not used	MODE	
Mode	R				RW	RW	R	RW	R			RW	R			RW	R	RW			RW	R		RW				RW	RW	RW	RW	
Initial Value	0				0	0	0	0	0			0	0			0	0	0			x	0		0				0	0	0	0	

Bit 1:0	**MODE**: ATOM channel mode select. 11: ATOM Signal Output Mode Serial (SOMS)
Bit 2	**Not used**: Not used in this mode 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 8:5	**Not used**: Not used in this mode Note: Not used in this mode.
Bit 10:9	**Reserved**: Read as zero, should be written as zero Note: Read as zero, should be written as zero.
Bit 11	**SL**: Defines signal level when channel and output is disable 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].

Bit 14:12 **CLK_SRC**: Shift frequency select for channel
000: *CMU_CLK0* selected
001: *CMU_CLK1* selected
010: *CMU_CLK2* selected
011: *CMU_CLK3* selected
100: *CMU_CLK4* selected
101: *CMU_CLK5* selected
110: *CMU_CLK6* selected
111: *CMU_CLK7* selected
Note: This register is a shadow register for the *CMU_CLKx* select. Thus, if the channel should operate on another CMU_CLK then *CMU_CKL0* at the beginning, the different CMU_CLK has to be specified inside this register and the CMU_CLK has to be configured with a FORCE_UPDATE in that case before the channel operation would start.

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

Bit 16 **Not used**: Not used in this mode
Note: Not used in this mode.

Bit 19:17 **Reserved**: Read as zero, should be written as zero
Note: Read as zero, should be written as zero.

Bit 20 **Not used**: Not used in this mode
Note: Not used in this mode.

Bit 23:21 **Not used**: Not used in this mode
Note: Not used in this mode.

Bit 24 **Not used**: Not used in this mode
Note: Not used in this mode.

Bit 25 **Reserved**: Read as zero, should be written as zero
Note: Read as zero, should be written as zero.

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**: Not used in this mode
Note: Not used in this mode.

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

12.4 ATOM Interrupt signals

The following table describes ATOM interrupt signals:

Signal	**Description**
CCU0TCx_IRQ	CCU0 Trigger condition interrupt for channel x
CCU1TCx_IRQ	CCU1 Trigger condition interrupt for channel x

12.5 ATOM Register overview

The following table shows a conclusion of ATOM register address offset and initial values.

Register name	**Description**	**Details in Section**
ATOM[i]_AGC_GLB_CTRL	AGC Global control register	12.6.1
ATOM[i]_AGC_ENDIS_CTRL	AGC0 Enable/disable control register	12.6.2
ATOM[i]_AGC_ENDIS_STAT	AGC Enable/disable status register (represents status of ATOM channels)	12.6.3
ATOM[i]_AGC_ACT_TB	AGC Action time base register	12.6.4
ATOM[i]_AGC_OUTEN_CTRL	AGC Output enable control register	12.6.5
ATOM[i]_AGC_OUTEN_STAT	AGC Output enable status register	12.6.6
ATOM[i]_AGC_FUPD_CTRL	AGC Force update control register	12.6.7
ATOM[i]_AGC_INT_TRIG	AGC Internal trigger control register	12.6.8
ATOM[i]_CH[x]_CTRL	ATOM Channel x control register (x=0…7)	12.6.9
ATOM[i]_CH[x]_STAT	ATOM Channel x status register (x=0…7)	12.6.10
ATOM[i]_CH[x]_RDADDR	ATOM Channel x ARU read address register (x=0...7)	12.6.11
ATOM[i]_CH[x]_CN0	ATOM Channel x CCU0 counter register (x=0…7)	12.6.12
ATOM[i]_CH[x]_CM0	ATOM Channel x CCU0 compare register (x=0…7)	12.6.13
ATOM[i]_CH[x]_SR0	ATOM Channel x CCU0 compare shadow register (x=0…7)	12.6.14
ATOM[i]_CH[x]_CM1	ATOM Channel x CCU1 compare register (x=0…7)	12.6.15
ATOM[i]_CH[x]_SR1	ATOM Channel x CCU1 compare shadow register (x=0…7)	12.6.16
ATOM[i]_CH[x]_IRQ_NOTIFY	ATOM channel x interrupt notification register (x=0…7)	12.6.17
ATOM[i]_CH[x]_IRQ_EN	ATOM channel x interrupt enable register (x=0…7)	12.6.18
ATOM[i]_CH[x]_IRQ_FORCINT	ATOM channel x software interrupt generation (x=0…7)	12.6.19

ATOM[i]_CH[x]_IRQ_MODE	IRQ mode configuration register (x=0...7)	12.6.20

12.6 ATOM Register description

12.6.1 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		Aw	Aw	Aw	Aw	Aw	Aw	Aw	Aw	R							Aw
Initial Value	00		00		00		00		00		00		00		00		0	0	0	0	0	0	0	0	0							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 S-R Flip-Flop **SOUR** is reset 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**: ATOM channel 0 enable update of register CM0, CM1 and CLK_SRC from SR0, SR1 and CLK_SRC_SR. Write of following double bit values is possible: 00 = don't care, bits 1:0 will not be change 01 = update disabled: is read as 00 (see below) 10 = update enabled: is read as 11 (see below) 11 = don't care, bits 1:0 will not be changed Read of following double values means: 00 = channel disabled 11 = channel 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

12.6.2 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																00		00		00		00		00		00		00		00	

Bit 1:0	**ENDIS_CTRL0**: ATOM channel 0 enable/disable update value. If an ATOM channel is disabled, the counter **CN0** is stopped and the Flip-Flop **SOUR** is set to the inverse value of control bit SL. On an enable event, the counter **CN0** starts counting from its current value. Write of following double bit values is possible: 00 = don't care, bits 1:0 of register ENDIS_STAT will not be changed on an update trigger 01 = disable channel on an update trigger 10 = enable channel on an update trigger 11 = 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 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.
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

12.6.3 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			
Mode	R																RW		RW		RW		RW		RW		RW		RW			
Initial Value	0x0000																00		00		00		00		00		00		00			

If an ATOM channel is disabled, the counter **CN0** is stopped and the Flip-Flop **SOUR** is set to the inverse value of control bit SL. On an enable event, the counter **CN0** starts counting from its current value.
Write of following double bit values is possible:
00 = don’t care, bits 1:0 will not be changed
01 = channel disabled: is read as 00 (see below)
10 = channel enabled: is read as 11 (see below)
11 = don’t care, bits 1:0 will not be changed
Read of following double values means:
00 = channel disable
11 = channel enable

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

12.6.4 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	00000					00		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
00 = *TBU_TS0* selected
01 = *TBU_TS1* selected
10 = *TBU_TS2* selected
11 = same as 00
Note: The bit combination "10" 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

12.6.5 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	00	00	00	00	00	00	00	00

Bit 1:0	**OUTEN_CTRL0**: Output ATOM_OUT(0) enable/disable update value Write of following double bit values is possible: 00 = don't care, bits 1:0 of register OUTEN_STAT will not be changed on an update trigger 01 = disable channel output on an update trigger 10 = enable channel output on an update trigger 11 = 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

12.6.6 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																00		00		00		00		00		00		00		00	

Bit 1:0	**OUTEN_STAT0**: Control/status of output ATOM_OUT(0) Write of following double bit values is possible: 00 = don’t care, bits 1:0 will not be changed 01 = channel disabled: is read as 00 (see below) 10 = channel enabled: is read as 11 (see below) 11 = don’t care, bits 1:0 will not be changed Read of following double values means: 00 = channel disable 11 = channel enable
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

12.6.7 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	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Bit 1:0 **FUPD_CTRL0**: Force update of ATOM channel 0 operation registers
Write of following double bit values is possible:
00 = don’t care, bits 1:0 will not be changed
01 = force update disabled: is read as 00 (see below)
10 = force update enabled: is read as 11 (see below)
11 = don’t care, bits 1:0 will not be changed
Read of following double values means:
00 = force update disabled
11 = force channel enabled

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
Write of following double bit values is possible:
00 = don’t care, bits 1:0 will not be changed
01 = CN0 is not reset on forced update: is read as 00 (see below)
10 = CN0 is reset on forced update: is read as 11 (see below)
11 = don’t care, bits 1:0 will not be changed
Read of following double values means:
00 = CN0 is not reset on forced update
11 = 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

12.6.8 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																00		00		00		00		00		00		00		00	

Bit 1:0 **INT_TRIG0**: Select input signal *TRIG_0* as a trigger source
Write of following double bit values is possible:
00 = don't care, bits 1:0 will not be changed
01 = internal trigger from channel 0 (TRIG_0) not used: is read as 00 (see below)
10 = internal trigger from channel 0 (TRIG_0) used: is read as 11 (see below)
11 = don't care, bits 1:0 will not be changed
Read of following double values means:
00 = internal trigger from channel 0 (TRIG_0) not used
11 = 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

12.6.9 Register ATOM[i]_CH[x]_CTRL (x: 0…7)

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	Reserved				ABM	OSM	SLA	TRIGOUT	Reserved			RST_CCU0	Reserved			WR_REQ	Reserved	CLK_SRC_SR			SL	Reserved	CMP_CTRL	ACB					ARU_EN	TB12_SEL	MODE	
Mode	R				RW	RW	RW	RW	R			RW	R			RW	R	RW			RW	R	RW	RW					RW	RW	RW	
Initial Value	0x0				0	0	0	0	0			0	0			0	0	0			x	0	0	0					0	0	00	

Bit 1:0	**MODE**: ATOM channel mode select. 00: ATOM Signal Output Mode Immediate (SOMI) 01: ATOM Signal Output Mode Compare (SOMC) 10: ATOM Signal Output Mode PWM (SOMP) 11: 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.
	Note: These bits are only applicable when ARU_EN = '0'.
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	**Reserved**: Read as zero, should be written as zero.
	Note: Read as zero, should be written as zero.
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.
	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.
	Note: In SOMS mode, this bit is only applicable when the channel and its output are disabled.
Bit 14:12	**CLK_SRC/CLK_SRC_SR**: actual CMU clock source (SOMS)/ shadow register for CMU clock source (SOMP).
	000: *CMU_CLK0* selected
	001: *CMU_CLK1* selected
	010: *CMU_CLK2* selected
	011: *CMU_CLK3* selected
	100: *CMU_CLK4* selected
	101: *CMU_CLK5* selected
	110: *CMU_CLK6* selected
	111: *CMU_CLK7* selected
	Note: This register is a shadow register for the *CMU_CLKx* select. 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 required to perform a forced update of CLK_SRC with the value of CLK_SRC_SR value before/with enabling the channel.

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

Bit 16 **WR_REQ**: CPU Write request bit for late compare register update.
Note: This bit is only applicable in SOMC mode.

Bit 19:17 **Reserved**: Read as zero, should be written as zero.
Note: Read as zero, should be written as zero.

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]*
Note: If RST_CCU0=1 and UPEN_CTRLx=1 are set, *TRIG_[x-1]* triggers also the update of work register (**CM0**, **CM1** and **CLK_SRC**).
Note: this bit is only applicable in SOMP mode.

Bit 23:21 **Reserved** : Read as zero, should be written as zero
Note: Read as zero, should be written as zero.

Bit 24 **TRIGOUT**: Trigger output selection (output signal *TRIG_CHx*) of module ATOM_CHx.
0 = *TRIG_[x[* is *TRIG_[x-1]*
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 (“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	**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 independent of pending compare match event 1 = ARU blocking mode enabled: after updating CM0,CM1 via ARU, no new data is read from ARU until compare match event is occurred. Note: this bit is only applicable in SOMC mode.
Bit 31:28	**Reserved** : Read as zero, should be written as zero Note: Read as zero, should be written as zero.

12.6.10 Register ATOM[i]_CH[x]_STAT (x: 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			ACBO					Reserved	WRF	DV	ACBI					Reserved															OL
Mode	R			R					R	RCw	R	R					R															R
Initial Value	0			0					0	0	0	0x00					0															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** : Read as zero, should be written as zero Note: Read as zero, should be written as zero.

Bit 20:16	**ACBI**: ATOM Mode control bits received through ARU. Note: 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.
Bit 21	**DV**: Valid ARU Data stored in compare registers. 0 = No valid data was received by ARU 1 = Valid data received by ARU and stored in CM0 and/or CM1 Note: This bit is only applicable in SOMC 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 mode.
Bit 23	**Reserved**: Read as zero, should be written as zero. Note: Read as zero, should be written as zero.
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** : Read as zero, should be written as zero Note: Read as zero, should be written as zero.

12.6.11 Register ATOM[i]_CH[x]_RDADDR (x: 0...7)

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 writeable if channel is disabled.
Bit 15:9	**Reserved**: Read as zero, should be written as zero. 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. Note: This read address is only applicable in SOMC mode. Note: this bit field is only writeable if channel is disabled.
Bit 31:25	**Reserved**: Read as zero, should be written as zero. Note: Read as zero, should be written as zero.

12.6.12 Register ATOM[i]_CH[x]_CN0 (x: 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								CN0																							
Mode	R								RW																							
Initial Value	0x00								0x000000																							

Bit 23:0 **CN0**: ATOM CCU0 counter register.
Bit 31:24 **Reserved**: Read as zero, should be written as zero.
Note: Read as zero, should be written as zero.

12.6.13 Register ATOM[i]_CH[x]_CM0 (x: 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								CM0																							
Mode	R								RW																							
Initial Value	0x00								0x000000																							

Bit 23:0 **CM0**: ATOM CCU0 compare register.
Bit 31:24 **Reserved**: Read as zero, should be written as zero.
Note: Read as zero, should be written as zero.
Note: This register is write protected in SOMC mode and returns AEI_STATUS='10' on write access, when in serve last compare strategy the first match of CCU0 occurred.

12.6.14 Register ATOM[i]_CH[x]_SR0 (x: 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								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**: Read as zero, should be written as zero.
Note: Read as zero, should be written as zero.

12.6.15 Register ATOM[i]_CH[x]_CM1 (x: 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								CM1																							
Mode	R								RW																							
Initial Value	0x00								0x000000																							

Bit 23:0 **CM1**: ATOM CCU1 compare register.
Bit 31:24 **Reserved**: Read as zero, should be written as zero.
Note: Read as zero, should be written as zero.
Note: This register is write protected in SOMC mode and returns AEI_STATUS='10' on write access, when in serve last compare strategy the first match of CCU0 occurred.

12.6.16 Register ATOM[i]_CH[x]_SR1 (x: 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								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**: Read as zero, should be written as zero.
Note: Read as zero, should be written as zero.

12.6.17 Register ATOM[i]_CH[x]_IRQ_NOTIFY (x: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																														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 = 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.
Bit 31:2	**Reserved**: Read as zero, should be written as zero. Note: Read as zero, should be written as zero

12.6.18 Register ATOM[i]_CH[x]_IRQ_EN (x:0...7)

Address Offset:	**see Appendix B**	**Initial Value:**	**0x0000_0000**
	31 … 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**: *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**: Read as zero, should be written as zero. Note: Read as zero, should be written as zero

12.6.19 Register ATOM[i]_CH[x]_IRQ_FORCINT (x:0...7)

Address Offset:	**see Appendix B**	**Initial Value:**	**0x0000_0000**
	31 … 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**: Read as zero, should be written as zero. Note: Read as zero, should be written as zero

12.6.20 Register ATOM[i]_CH[x]_IRQ_MODE (x:0...7)

Address Offset:	**see Appendix B**	**Initial Value:**	**0x0000_000X**
	31 … 2		1 … 0
Bit	Reserved		IRQ_MODE
Mode	R		RW
Initial Value	0x0000 0000		XX

Bit 1:0	**IRQ_MODE**: IRQ mode selection 00 = Level mode 01 = Pulse mode 10 = Pulse-Notify mode 11 = 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 Multi Channel Sequencer (MCS)

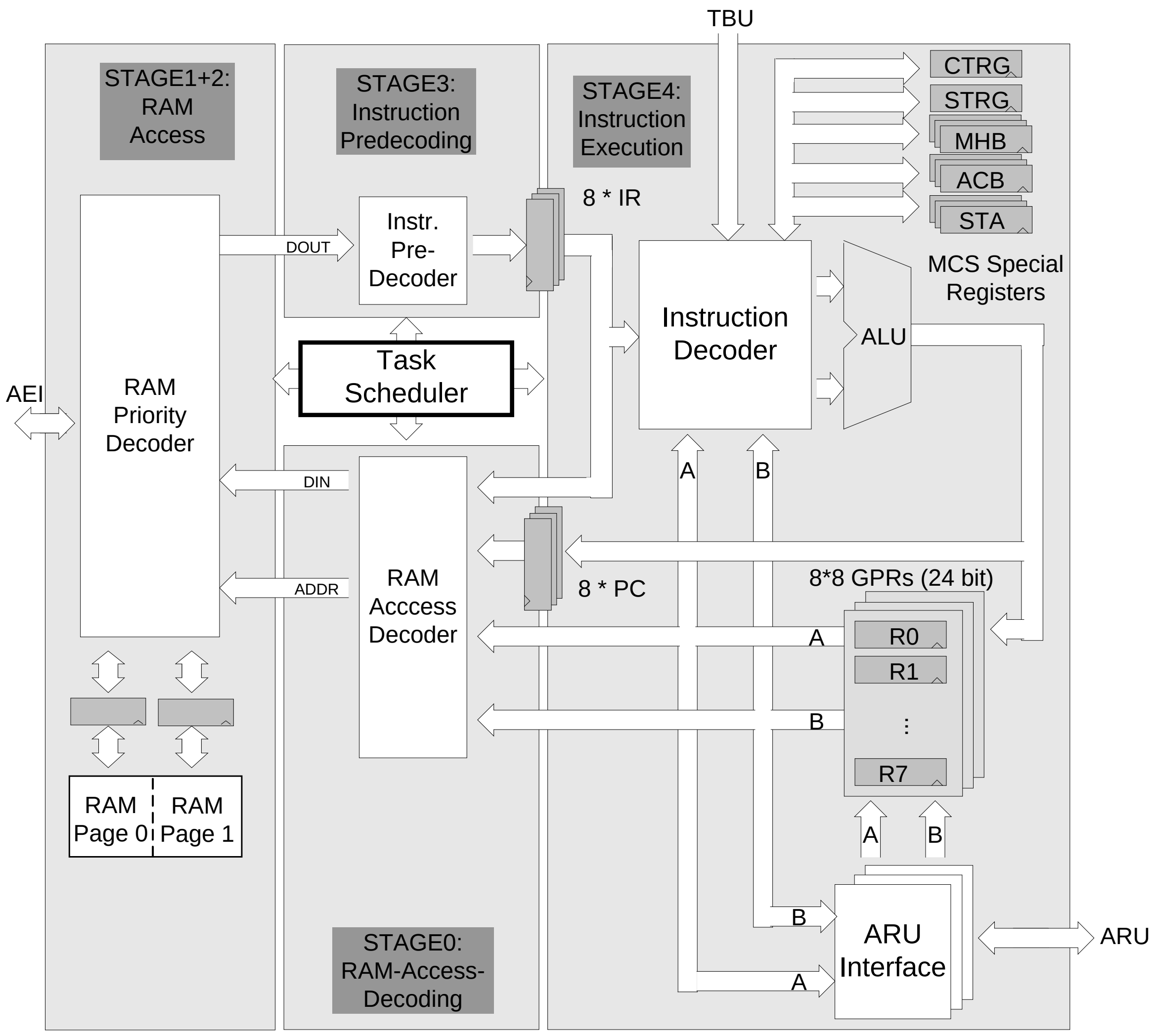
13.1 Overview

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.

13.1.1 Architecture

Figure 13.1.1.1 gives an overview of the MCS architecture.

13.1.1.1 MCS architecture



The MCS sub module mainly embeds a single data path with five pipeline stages, consisting of a simple 24-bit Arithmetic Logic Unit (ALU), several decoders, and a connection to two RAM pages located outside of the MCS sub module.

The data path of the MCS is shared by eight so called MCS-channels, whereas each MCS-channel executes a dedicated micro-program that is stored inside the RAM pages connected to the MCS sub module. The execution of the different MCS-channels is controlled by a central task scheduler.

Both RAM pages may contain arbitrary sized code and data sections that are accessible by all MCS-channels and the CPU via AEI. The addresses for accessing the RAM pages via AEI can be found in [1].

The MCS sub module supports a memory layout of up to 2^{12} memory locations each 32 bit wide leading to a maximum byte wise address range from 0 to 2^{14}-1.

This address space of the MCS is divided into two seamless memory pages.
Memory page 0 begins from address 0 and ranges to address MP0-4 and memory page 1 ranges from MP0 to MP1-4.

The parameters MP0 and MP1 are defined externally by the memory configuration sub module MCFG of section 14.

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.

An MCS-channel can also be considered as an individual task of a processor that is scheduled at a specific point in time.

A more detailed description of the scheduling can be found in section 13.1.2.
Moreover, each MCS-channel has a dedicated ARU interface for communication with other ARU connected modules, an Instruction Register (IR), a Program Counter Register (PC), a Status Register (STA), an ARU Control Bit Register (ACB), a Memory High Byte Register (MHB) and a Register Bank with eight 24 bit general purpose registers (R0, R1, ...R7).

All the registers, mentioned above, are only visible within its dedicated MCS-channel and thus the MCS-channels cannot exchange data using registers.

An exception is the common 16 bit wide trigger register that can be accessed by all MCS channels in order to trigger other MCS-channels located in the same sub module. STRG is used to set bits and CTRG is used to clear bits in the trigger register.

To enable triggering of MCS-channels by CPU, it can set bits in the common trigger register by writing to **MCS[i]_STRG** and clear bits by writing to **MCS[i]_CTRG**.

Whenever data has to be exchanged between different MCS-channels, the connected RAM pages, which are accessible by all MCS-channels, must be used.

However, since the data registers are writable by the Host CPU, an MCS channel may also exchange data with the CPU using its data registers.

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.

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.

Pipeline stage 3 performs pre-decoding of instruction and data resulting from the RAM.
Finally, in pipeline stage 4 the current instruction is executed.
Since the internal registers of the MCS can be updated by different sources (MCS write access, AEI write access, ARU read access) 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.

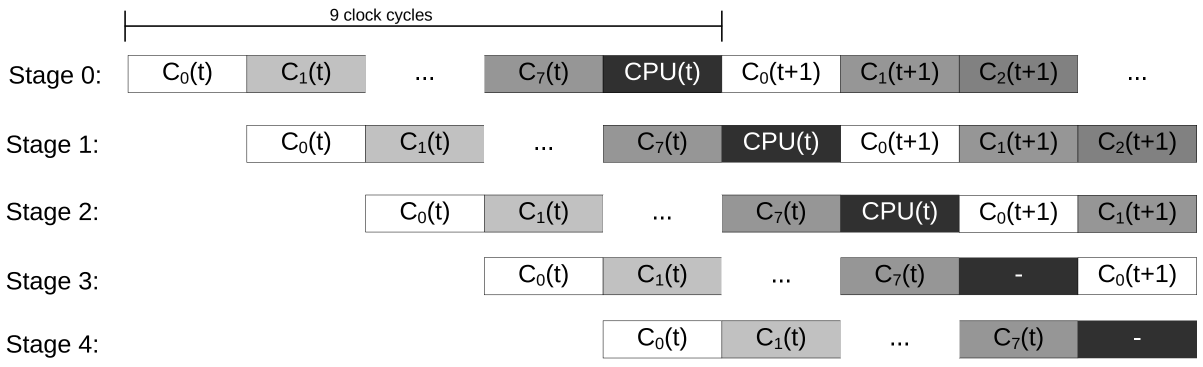
One exception is the common trigger register which may be written by multiple sources (different MCS channels and AEI) in order to enable fast triggering of different MCS channels. Typically, the software should setup its application in a manner that different sources should not write the same bits in the trigger register.

13.1.2 Scheduling

The MCS sub module provides two different scheduling schemes: *round-robin schedule* and *accelerated schedule*.
The scheduling scheme can be selected by the **SCHED** bit field in the global **MCS[i]_CTRL** register.
The round-robin order scheduling assigns all MCS-channels an equal amount of time slices.
In addition, the scheduler also assigns one time slice to the CPU, in order to guarantee at least one memory access by the CPU within each round-trip cycle.

Figure 13.1.2.1 shows the round-robin scheduling with 8 MCS-channels (C_0 to C_7) that are scheduled together with a single CPU access.

13.1.2.1 Scheduling



The figure also shows which MCS-channel is activated in specific pipeline stage at a specific point in time.

The execution time of an MCS-channel in a specific pipeline stage is always one clock cycle.
The index t marks all instruction parts of the corresponding MCS-channels belonging to the same round-trip cycle.

Consequently, a single cycle instruction of an MCS-channel requires an effective execution time of 9 clock cycles, ignoring the five clock cycles of pipeline latency.

To improve memory bandwidth between CPU and MCS memory, the time slices of any suspended MCS-channel is also granted to the CPU.

An MCS-channel can be suspended due to the following reasons:

- An MCS-channel is executing a blocking read or write request to an ARU connected sub module (instruction ARD, AWR, ARDI, AWRI).

- An MCS-channel waits on a register match event (instruction WURM), in order to wait on a desired register value (e.g. trigger event from another MCS channel).

- An MCS-channel is disabled.

The round-robin scheduling leads to a deterministic round trip time for the whole sub module, however it may waste clock cycles by scheduling MCS-channels that are not able to run at a specific point in time assuming that there is no high CPU bandwidth required.

The second scheduling mode, the *accelerated scheduling mode*, improves the computational performance of the MCS by skipping suspended MCS-channels (and channels that are currently in stage 0, 1, 2, or 3) during scheduling.

Whenever the scheduler detects an MCS-channel that is suspended or not in the last pipeline stage, it is selecting another non-suspended MCS-channel.

Considering the timing of Figure 13.1.2.1 in conjunction with the accelerated scheduling scheme, a single cycle instruction of an MCS-channel requires an effective execution time between 5 and 9 clock cycles, depending on the number of suspended MCS-channels.

In summary, the *round-robin scheduling* mode grants time slices of suspended MCS-channels to the CPU and the *accelerated scheduling* mode grants time slices of suspended MCS-channels to non-suspended MCS-channels.

13.2 Instruction Set

This section describes the entire instruction set of the MCS sub module.
After the introduction of the different instruction formats in section 13.2.1, the individual instructions are described in sections 13.2.2.1 to 13.2.7.

In general, each instruction is 32 bit wide but the duration of each instruction varies between several *instruction cycles*. An instruction cycle is defined as the time in system clock cycles that rest between two consecutive instructions of a channel.

As already described in section 13.1.2, the number of required clock cycles for a single instruction cycle can vary in the range of 5 to 9 clock cycles, depending on the number suspended MCS-channels, when the *accelerated scheduling* scheme is selected inside the **MCS[i]_CTRL** register. In the *round robin scheduling* scheme, each single cycle instruction takes exactly 9 clock cycles.

Before the instruction formats and the actual instructions are described, some commonly used terms, abbreviations and expressions are introduced:

OREG: The operation register set OREG = {R0, R1, R2, ..., R7, STA, ACB, CTRG, STRG, TBU_TS0, TBU_TS1, TBU_TS2, MHB} includes all MCS accessible internal registers, as well as the global time bases TBU_TS0, TBU_TS1, and TBU_TS2 that are provided by the sub module TBU.

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.

LIT4: The set LIT4 = {0,1, ..., 15} is an unsigned 4 bit literal.
LIT16: The set LIT16 = {0,1, ..., 2^{16}-1} is an unsigned 16 bit literal.
LIT24: The set LIT24 = {0,1, ..., 2^{24}-1} is an unsigned 24 bit literal.
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 ADRESSING: The expression MEM(X) represents the 32 bit value at address X of the memory.
Address X ranges between 0 and 2^{14}-4, whereas X must be an integral multiple of 4.
The expression MEM(X)[m:n] represents the bit slice ranging from bit n to m of the 32 bit word at memory location X.

ARU ADRESSING: In the case of ARU reading , the expression ARU(X) represents the 53 bit ARU word of ARU channel at address X.

The read address X ranges between 0 and 2^9-1.
In the case of ARU writing, the expression ARU(X) represents a 53 bit ARU word that is written to an ARU channel indexed by the index X.

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 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 53 bit ARU word.

13.2.1 Instruction Format

The first instruction format, called literal instruction format, embeds a primary 4 bit opcode OPC0, a 24 bit literal value C ∈ LIT24, and a 4 bit value A, which may be an element of set OREG or AREG, depending on the actual instruction.

Figure 13.2.1.1 shows the bit alignment of the literal instruction format.

13.2.1.1 Literal Instruction format

31 28	27 24	23 0
OPC0 (4bit)	A (4bit)	C (24bit)

The literal instruction format is primarily used for instructions that are accessing a 24 bit literal and a single 24 bit register as operands.

In the following subsections the binary codes of a 24 bit literal instruction is defined as "xxxxaaaacccccccccccccccccccccccc", whereas the digits 'x' encode the field OPC0, the digits 'a' encode the operand field A, and the digits 'c' encode the 24 bit literal field C.

If an instruction ignores several bits of field, the bits are defined as '-' in its code.
The second instruction format, called double operand instruction format, embeds a 4 bit primary opcode OPC0, a 4 bit secondary opcode OPC1, a 16 bit literal C ∈ LIT16 and two 4 bit values A and B, which may be an element of set OREG, AREG, or LIT4 depending on the actual instruction.

Figure 13.2.1.2 shows the bit alignment of the double operand instruction format.

13.2.1.2 Double Operand Instruction format

31 28	27 24	23 20	19 16	15 0
OPC0 (4bit)	A (4bit)	B (4bit)	OPC1 (4bit)	C (16bit)

The double operand instruction format is primarily used for instructions that are accessing two operands stored in the 24 bit registers.

In the following subsections the binary codes of a 16 bit literal instruction is defined as "xxxxaaaabbbbyyyycccccccccccccccc", whereas the digits 'x' encode the bit field OPC0, 'y' the digits of field OPC2, the digits 'a' encode the operand field A, the digits 'b' the operand field B, and the digits 'c' encode the 16 bit literal field C.

If an instruction ignores several bits of field, the bits are defined as '-' in its code.
If the instruction decoder detects an invalid combination of the fields opcode OPC0 and OPC1, the corresponding MCS-channel is disabled and the ERR bit in the register STA is set.

13.2.2 Data Transfer Instructions

13.2.2.1 MOVL Instruction

Syntax: MOVL A, C
Operation: A $\leftarrow$ C
Status: Z
Duration: 1 instruction cycle
Code: 0001aaaacccccccccccccccccccccccc
Description: Transfer literal value C (C $\in$ LIT24) to register A (A $\in$ 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.

13.2.2.2 MOV Instruction

Syntax: MOV A, B
Operation: A $\leftarrow$ B
Status: Z
Duration: 1 instruction cycle
Code: 1010aaaabbbb0000----------------
Description: Transfer value B (B $\in$ OREG) to register A (A $\in$ 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.

13.2.2.3 MRD Instruction

Syntax: MRD A, C
Operation: A ← MEM(C[13:0])[23:0];

MHB ← MEM(C[13:0])[31:24]

Status: Z
Duration: 2 instruction cycles
Code: 1010aaaa----0001--cccccccccccccc
Description: Transfer the lower 24 bits of memory content at address C (C ∈ LIT16) to register A (A ∈ OREG).
The upper 8 bits of the memory content at address C are transferred to MHB register.
The zero bit Z of status register STA is set, if the lower 24 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 7 of the referred memory location are transferred to MHB.

The program counter PC is incremented by the value 4.

13.2.2.4 MWR Instruction

Syntax: MWR A, C
Operation: MEM(C[13:0])[23:0] ← A;

MEM(C[13:0])[31:24] ← MHB
Status: -
Duration: 2 instruction cycles
Code: 1010aaaa----0010--cccccccccccccc
Description: Transfer 24 bit value of register A (A ∈ OREG) together with the MHB register to the memory location at address C (C ∈ LIT16).

The 24 bit value of register A is stored in the lower significant bits (bit 0 to 23) of the memory location.
The MHB register is stored in bits 24 to 31 of the referred memory location.
The program counter PC is incremented by the value 4.

13.2.2.5 MWR24 Instruction

Syntax: MWR24 A, C
Operation: MEM(C[13:0])[23:0] ← A
Status: -
Duration: 3 instruction cycles
Code: 1010aaaa----0111--cccccccccccccc
Description: Transfer 24 bit value of register A (A ∈ OREG) to memory at address C (C ∈ LIT16).
The 24 bit value of register A is stored in the lower significant bits (bit 0 to 23) of the memory location and the bits 24 to 31 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.

13.2.2.6 MRDI Instruction

Syntax: MRDI A, B
Operation: A ← MEM(B[13:0])[23:0]

MHB ← MEM(B[13:0])[31:24]
Status: Z
Duration: 2 instruction cycles
Code: 1010aaaabbbb0011----------------
Description: Transfer the lower 24 bits of a memory location to register A (A ∈ OREG) using indirect addressing.

The upper 8 bits of the memory location are transferred to MHB register.

The memory location where to read from is defined by the bits 0 to 13 of register B (B ∈ OREG).
The 24 bit value is received from the lower significant bits (bit 0 to 23) of the memory location.
The zero bit Z of status register STA is set, if the lower 24 bit of the transferred value is zero, otherwise the zero bit is cleared.

If the MHB register is selected as destination register A (A ∈ OREG), the bits 0 to 7 of the referred memory location are transferred to MHB.

The program counter PC is incremented by the value 4.

13.2.2.7 MWRI Instruction

Syntax: MWRI A, B
Operation: MEM(B[13:0])[23:0] ← A;

MEM(B[13:0])[31:24] $\leftarrow$ MHB
Status: -
Duration: 2 instruction cycles
Code: 1010aaaabbbb0100----------------
Description: Transfer 24 bit value of register A (A $\in$ OREG) to the 24 least significant bits of to the memory using indirect addressing.

The memory location where to write to is defined by the bits 0 to 13 of register B (B $\in$ OREG).
The 24 bit value is stored in the lower significant bits (bit 0 to 23) of the memory location and the MHB register is stored in bits 24 to 31.

The program counter PC is incremented by the value 4.

13.2.2.8 MWRI24 Instruction

Syntax: MWRI24 A, B
Operation: MEM(B[13:0])[23:0] $\leftarrow$ A;
Status: -
Duration: 3 instruction cycles
Code: 1010aaaabbbb1000----------------
Description: Transfer 24 bit value of A (A $\in$ OREG) to memory using indirect addressing.
The memory location where to write to is defined by the bits 0 to 15 of register B (B $\in$ OREG).
The 24 bit value is stored in the lower significant bits (bit 0 to 23) of the memory location and the bits 24 to 31 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.

13.2.2.9 POP Instruction

Syntax: POP A
Operation: A $\leftarrow$ MEM(R7[13:0])[23:0];
R7 $\leftarrow$ R7 - 4;
SP_CNT $\leftarrow$ SP_CNT - 1
MHB $\leftarrow$ MEM(R7[13:0])[31:24];
Status: Z, EN
Duration: 2 instruction cycles
Code: 1010aaaa----0101----------------

Description: Transfer the lower significant bits (bit 0 to 23) from the top of stack to register A (A $\in$ OREG), followed by decrementing the stack pointer register R7 with the value 4.

The upper 8 bits (bit 24 to 31) from the top of the stack are transferred to register MHB.
If the MHB register is selected as destination register A (A $\in$ OREG), the bits 0 to 7 from the top of the stack are transferred to MHB.

The memory location for the top of the stack is identified by the bits 0 to 13 of the stack pointer register R7.

The zero bit Z of status register STA is set, if the lower 24 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**.

13.2.2.10 PUSH Instruction

Syntax: PUSH A
Operation: R7 $\leftarrow$ R7 + 4;
SP_CNT $\leftarrow$ SP_CNT + 1;
MEM(R7[13:0])[23:0] $\leftarrow$ A;
MEM(R7[13:0])[31:24] $\leftarrow$ MHB
Status: EN
Duration: 2 instruction cycles
Code: 1010aaaa----0110----------------
Description: Increment the stack pointer register R7 with the value 4, followed by transferring a 24 bit value of operand A (A $\in$ OREG) together with a MHB register to the new top of the stack. The 24 bit value of A is stored in the bits 0 to 23 of the memory location.

The content of the MHB register is stored in the bit 24 to 31 of the memory location.
The memory location for the top of the stack is addressed by the bits 0 to 13 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 discard.

13.2.3 ARU Instructions

13.2.3.1 ARD Instruction

Syntax: ARD A, B, C
Operation: A ← ARU(C[8:0])[23:0];
B ← ARU(C[8:0])[47:24];
ACB[4:0] ← ARU(C[8:0])[52:48]
Status: CAT
Duration: suspends current MCS-channel
Code: 1011aaaabbbb0000-------ccccccccc
Description: Perform a blocking read access to the ARU and transfer both 24 bit values received at the ARU port to the registers A and B (A ∈ AREG, B ∈ AREG), whereas A holds the lower 24 bit ARU word and B holds the upper 24 bit ARU word.

If A and B refer to the same register, only the upper 24 bit ARU word is stored and the lower 24 bit ARU word is discard.

If any transferred 24 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. Actually, all address values of A and B that exceed the range 0 to 7 discard the corresponding ARU data.

The received ARU control bits are stored in the register ACB.
The lower significant bits of the literal C (C ∈ LIT16) define the ARU address where to read from.
At the beginning of the instruction execution the CAT bit in the register STA is always cleared. After the execution of the instruction the CAT bit is updated in order to show if the instruction finished successfully (CAT = 0) or it was cancelled by the CPU (CAT = 1).

The program counter PC is incremented by the value 4.

13.2.3.2 ARDI Instruction

Syntax: ARDI A, B
Operation: A ← ARU(R6[8:0])[23:0];
B ← ARU(R6[8:0])[47:24];
ACB[4:0] ← ARU(R6[8:0])[52:48]
Status: CAT
Duration: suspends current MCS-channel
Code: 1011aaaabbbb0100----------------
Description: Perform a blocking read access to the ARU and transfer both 24 bit values received at the ARU port to the registers A and B (A ∊ AREG, B ∊ AREG), whereas A holds the lower 24 bit ARU word and B holds the upper 24 bit ARU word.

If A and B refer to the same register, only the upper 24 bit ARU word is stored and the lower 24 bit ARU word is discard.

If any transferred 24 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. Actually, all address values of A and B that exceed the range 0 to 7 discard the corresponding ARU data.

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 the register STA is always cleared. After the execution of the instruction the CAT bit is updated in order to show if the instruction finished successfully (CAT = 0) or it was cancelled by the CPU (CAT = 1).

The program counter PC is incremented by the value 4.

13.2.3.3 AWR Instruction

Syntax: AWR A, B, C
Operation: ARU(C[4:0])[23:0] ← A;
ARU(C[4:0])[47:24] ← B;
ARU(C[4:0])[52:48] ← ACB[4:0];
Status: CAT
Duration: suspends current MCS-channel
Code: 1011aaaabbbb0001-----------ccccc
Description: Perform a blocking write access to the ARU and transfer two 24 bit values to the ARU port using the registers A and B (A ∊ OREG, B ∊ OREG), whereas A holds the lower 24 bit ARU word and B holds the upper 24 bit ARU word.

The ARU control bits to be sent are taken from the register ACB.

The lower significant bits (bit 0 to bit 4) of the literal C (C ∈ LIT16) define an index into the pool of ARU write address that is used for writing data.

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 in the register STA is always cleared. After the execution of the instruction the CAT bit is updated in order to show if the instruction finished successfully (CAT = 0) or it was cancelled by the CPU (CAT = 1).

The program counter PC is incremented by the value 4.

13.2.3.4 AWRI Instruction

Syntax: AWRI A, B
Operation: ARU(R6[4:0])[23:0] ← A;
ARU(R6[4:0])[47:24] ← B;
ARU(R6[4:0])[52:48] ← ACB[4:0];

Status: CAT
Duration: suspends current MCS-channel
Code: 1011aaaabbbb0101----------------
Description: Perform a blocking write access to the ARU and transfer two 24 bit values to the ARU port using the registers A and B (A ∈ OREG, B ∈ OREG), whereas A holds the lower 24 bit ARU word and B holds the upper 24 bit ARU word.

The ARU control bits to be sent are taken from the register ACB.
The bits 0 to 4 of the register R6 define an index into the pool of ARU write address that is used for writing data.

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 in the register STA is always cleared. After the execution of the instruction the CAT bit is updated in order to show if the instruction finished successfully (CAT = 0) or it was cancelled by the CPU (CAT = 1).

The program counter PC is incremented by the value 4.

13.2.3.5 NARD Instruction

Syntax: NARD A, B, C
Operation: A ← ARU(C[8:0])[23:0];
B ← ARU(C[8:0])[47:24];
ACB[4:0] ← ARU(C[8:0])[52:48]
Status: SAT
Duration: suspends current MCS-channel for a maximum of one ARU round trip cycle
Code: 1011aaaabbbb0010-------ccccccccc
Description: Perform a non-blocking read access to the ARU trying to transfer both 24 bit values received at the ARU port to the registers A and B (A ∈ AREG, B ∈ AREG), whereas A holds the lower 24 bit ARU word, B holds the upper 24 bit ARU word, and the ACB register holds the received ARU control bits.

The lower significant bits of the literal C (C ∈ LIT16) define the ARU address where to read from.
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 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 24 bit ARU word is stored and the lower 24 bit ARU word is discard.

If any transferred 24 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. Actually, all address values of A and B that exceed the range 0 to 7 discard the corresponding ARU data.

The program counter PC is incremented by the value 4.

13.2.3.6 NARDI Instruction

Syntax: NARDI A, B
Operation: A ← ARU(R6[8:0])[23:0];
B ← ARU(R6[8:0])[47:24];
ACB[4:0] ← ARU(R6[8:0])[52:48]
Status: SAT
Duration: suspends current MCS-channel for a maximum of one ARU round trip cycle

Code: 1011aaaabbbb0011----------------
Description: Perform a non-blocking read access to the ARU trying to transfer both 24 bit values received at the ARU port to the registers A and B (A ∈ AREG, B ∈

AREG), whereas A holds the lower 24 bit ARU word, B holds the upper 24 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.
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 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 24 bit ARU word is stored and the lower 24 bit ARU word is discard.

If any transferred 24 bit value from the ARU should not stored in a register, the dummy register ZERO $\in$ AREG can be selected in A or B to discard the corresponding ARU data. Actually, all address values of A and B that exceed the range 0 to 7 discard the corresponding ARU data.

The program counter PC is incremented by the value 4.

13.2.4 Arithmetic Logic Instructions

13.2.4.1 ADDL Instruction

Syntax: ADDL A, C
Operation: A $\leftarrow$ A + C
Status: Z, CY, N, V
Duration: 1 instruction cycle
Code: 0010aaaacccccccccccccccccccccccc
Description: Perform addition operation of a register A (A $\in$ OREG) with a 24 bit literal value C (C $\in$ LIT24) 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^{24}-1], assuming that both operands A and C are unsigned values within the interval [0; 2^{24}-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^{23}; 2^{23}-1], assuming that both operands A and C are signed values within the interval [-2^{23}; 2^{23}-1].

The negative bit N of status register STA equals the most significant bit 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.

13.2.4.2 ADD Instruction

Syntax: ADD A, B
Operation: A $\leftarrow$ A + B
Status: Z, CY, N, V
Duration: 1 instruction cycle
Code: 1100aaaabbbb0000----------------
Description: Perform addition operation of a register A (A $\in$ OREG) with an operand B (B $\in$ OREG) 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 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^{24}-1], assuming that both operands A and B are unsigned values within the interval [0; 2^{24}-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^{23}; 2^{23}-1], assuming that both operands A and B are signed values within the interval [-2^{23}; 2^{23}-1].

The negative bit N of status register STA equals the most significant bit 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.

13.2.4.3 SUBL Instruction

Syntax: SUBL A, C
Operation: A $\leftarrow$ A - C

Status: Z, CY, N, V
Duration: 1 instruction cycle
Code: 0011aaaacccccccccccccccccccccccc
Description: Perform subtraction operation of a register A (A ∈ OREG) with a 24 bit literal value C (C ∈ LIT24) 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 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^{24}-1], assuming that both operands A and C are unsigned values within the interval [0; 2^{24}-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^{23}; 2^{23}-1], assuming that both operands A and C are signed values within the interval [-2^{23}; 2^{23}-1].

The negative bit N of status register STA equals the most significant bit 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.

13.2.4.4 SUB Instruction

Syntax: SUB A, B
Operation: A $\leftarrow$ A - B
Status: Z, CY, N, V
Duration: 1 instruction cycle
Code: 1100aaaabbbb0001----------------
Description: Perform subtraction operation of a register A (A ∈ OREG) with an operand B (B ∈ OREG) 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 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^{24}-1], assuming that both operands A and B are unsigned values within the interval [0; 2^{24}-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^{23}; 2^{23}-1], assuming that both operands A and B are signed values within the interval [-2^{23}; 2^{23}-1].

The negative bit N of status register STA equals the most significant bit 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.

13.2.4.5 NEG Instruction

Syntax: NEG A, B
Operation: A $\leftarrow$ - B
Status: Z, N, V
Duration: 1 instruction cycle
Code: 1100aaaabbbb0010----------------
Description: Perform negation operation (2's Complement) with an operand B (B $\in$ OREG) and store the result in a register A (A $\in$ OREG).

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^{23}; 2^{23}-1], assuming that both operands A and B are signed values within the interval [-2^{23}; 2^{23}-1].

The negative bit N of status register STA equals the most significant bit 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.

13.2.4.6 ANDL Instruction

Syntax: ANDL A, C
Operation: A $\leftarrow$ A AND C
Status: Z
Duration: 1 instruction cycle

Code: 0100aaaacccccccccccccccccccccccc
Description: Perform bitwise AND conjunction of a register A (A ∈ OREG) with a 24 bit literal value C (C ∈ LIT24) 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.

13.2.4.7 AND Instruction

Syntax: AND A, B
Operation: A ← A AND B
Status: Z
Duration: 1 instruction cycle
Code: 1100aaaabbbb0011----------------
Description: Perform bitwise AND conjunction of a register A (A ∈ OREG) with an operand B (B ∈ OREG) 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.

13.2.4.8 ORL Instruction

Syntax: ORL A, C
Operation: A ← A OR C
Status: Z
Duration: 1 instruction cycle
Code: 0101aaaacccccccccccccccccccccccc
Description: Perform bitwise OR conjunction of a register A (A ∈ OREG) with a 24 bit literal value C (C ∈ LIT24) 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.

13.2.4.9 OR Instruction

Syntax: OR A, B
Operation: A ← A OR B
Status: Z
Duration: 1 instruction cycle
Code: 1100aaaabbbb0100----------------
Description: Perform bitwise OR conjunction of a register A (A ∊ OREG) with an operand B (B ∊ OREG) 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.

13.2.4.10 XORL Instruction

Syntax: XORL A, C
Operation: A ← A XOR C
Status: Z
Duration: 1 instruction cycle
Code: 0110aaaacccccccccccccccccccccccc
Description: Perform bitwise XOR conjunction of a register A (A ∊ OREG) with a 24 bit literal value C (C ∊ LIT24) 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.

13.2.4.11 XOR Instruction

Syntax: XOR A, B
Operation: A ← A XOR B
Status: Z
Duration: 1 instruction cycle
Code: 1100aaaabbbb0101----------------
Description: Perform bitwise XOR conjunction of a register A (A ∊ OREG) with an operand B (B ∊ OREG) 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.

13.2.4.12 SHR Instruction

Syntax: SHR A, C
Operation: A ← A >> C
Status: Z, CY
Duration: 1 instruction cycle
Code: 1100aaaa----0110-----------ccccc
Description: Perform right shift operation C (C ∊ LIT16) times of register A (A ∊ OREG), whereas C must be in the range [0; 24]. The most significant bits 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.
The program counter PC is incremented by the value 4.

13.2.4.13 SHL Instruction

Syntax: SHL A, C
Operation: A ← A << C
Status: Z, CY
Duration: 1 instruction cycle
Code: 1100aaaa----0111-----------ccccc
Description: Perform left shift operation C (C ∊ LIT16) times of register A (A ∊ OREG), whereas C must be in the range [0; 24]. The lower significant bits 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 MSB that is shifted out of the register.
The program counter PC is incremented by the value 4.

13.2.5 Test Instructions

13.2.5.1 ATUL Instruction

Syntax: ATUL A, C
Operation: A - C
Status: Z, CY
Duration: 1 instruction cycle
Code: 0111aaaacccccccccccccccccccccccc
Description: Arithmetic Test with an unsigned operand A (A ∈ OREG) and an unsigned 24 bit literal value C (C ∈ LIT24).

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.

13.2.5.2 ATU Instruction

Syntax: ATU A, B
Operation: A - B
Status: Z, CY
Duration: 1 instruction cycle
Code: 1101aaaabbbb0000----------------
Description: Arithmetic Test with an unsigned operand A (A ∈ OREG) and an unsigned operand B (B ∈ OREG).
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.

13.2.5.3 ATSL Instruction

Syntax: ATSL A, C
Operation: A - C
Status: Z, CY
Duration: 1 instruction cycle

Code: 1000aaaacccccccccccccccccccccccc
Description: Arithmetic Test with a signed operand A (A ∊ OREG) and a signed 24 bit literal value C (C ∊ LIT24).

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.

13.2.5.4 ATS Instruction

Syntax: ATS A, B
Operation: A - B
Status: Z, CY
Duration: 1 instruction cycle
Code: 1101aaaabbbb0001----------------
Description: Arithmetic Test with a signed operand A (A ∊ OREG) and a signed operand B (B ∊ OREG).
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.

13.2.5.5 BTL Instruction

Syntax: BTL A, C
Operation: A AND C
Status: Z
Duration: 1 instruction cycle
Code: 1001aaaacccccccccccccccccccccccc
Description: Bit test of an operand A (A ∊ OREG) with a 24 bit literal bit mask C (C ∊ LIT24).
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.

13.2.5.6 BT Instruction

Syntax: BT A, B
Operation: A AND B
Status: Z
Duration: 1 instruction cycle
Code: 1101aaaabbbb0010----------------
Description: Bit test of an operand A (A ∈ OREG) with an operand B (B ∈ OREG), 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.

13.2.6 Control Flow Instructions

13.2.6.1 JMP Instruction

Syntax: JMP C
Operation: PC $\leftarrow$ C[13:0]
Status: -
Duration: 1 instruction cycle
Code: 1110--------0000--cccccccccccccc
Description: Execute unconditional jump to the memory location C (C ∈ LIT16).

The program counter PC is loaded with literal C.

13.2.6.2 JBS Instruction

Syntax: JBS A, B, C

Operation: PC ← C[13:0] if A[B] is set
Status: -
Duration: 1 instruction cycle
Code: 1110aaaabbbb0001--cccccccccccccc
Description: Execute conditional jump to the memory location C (C ∈ LIT16).

The program counter PC is loaded with literal C, if the bit at position B (B ∈ LIT4) of operand A (A ∈ OREG) is set.

Otherwise, if the bit is cleared, the program counter PC is incremented by the value 4.

13.2.6.3 JBC Instruction

Syntax: JBC A, B, C
Operation: PC ← C[13:0] if A[B] is cleared
Status: -
Duration: 1 instruction cycle
Code: 1110aaaabbbb0010--cccccccccccccc
Description: Execute conditional jump to the memory location C (C ∈ LIT16).

The program counter PC is loaded with literal C, if the bit at position B (B ∈ LIT4) of operand A (A ∈ OREG) is cleared.

Otherwise, if the bit is set, the program counter PC is incremented by the value 4.

13.2.6.4 CALL Instruction

Syntax: CALL C
Operation: R7 ← R7 + 4;
    MEM(R7[15:0])[15:0] ← PC + 4;
    PC ← C[13:0];
    SP_CNT ← SP_CNT + 1

Status: EN
Duration: 2 instruction cycles
Code: 1110--------0011--cccccccccccccc
Description: Call subprogram at memory location C (C ∈ LIT16).
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 0 to 15 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** 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** is set, the memory write operation of the incremented PC is discarding.

13.2.6.5 RET Instruction

Syntax: RET
Operation: PC ← MEM(R7[15:0])[15:0];
    R7 ← R7 - 4;
    SP_CNT ← SP_CNT - 1
Status: EN
Duration: 2 instruction cycles
Code: 1110--------0100----------------
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 0 to 15 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**.

13.2.7 Other Instructions

13.2.7.1 WURM Instruction

Syntax: WURM A B C
Operation: Wait until register match
Status: CWT

Duration: suspends current MCS-channel
Code: 1111aaaabbbb0000cccccccccccccccc
Description: Suspend current MCS-channel until the following register match condition occurs:

A = (B AND MASK)

whereas A ∈ OREG, B ∈ 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 ∈ LIT16.

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.

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, TBU_TS1, or TBU_TS2, register A is a general purpose register holding the value to wait for and bitmask C should be set to 0xFFFF.

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 cancelled by the CPU (CWT = 1).

If the CWT bit is set simultaneously with the occurrence of the register match condition, the register match condition has the higher priority resulting in a cleared CWT bit.

The program counter PC is incremented by the value 4, when the trigger bit is set and the channel continues its operation.

13.2.7.2 NOP Instruction

Syntax: NOP
Operation: -
Status: -
Duration: 1 instruction cycle
Code: 0000----------------------------
Description: No operation is performed.
The program counter PC is incremented by the value 4.

13.3 MCS Internal Registers

This section describes MCS internal registers which are partly only accessible by the corresponding MCS-channel itself and partly global to all channels. Please see Table 13.3.1 for clarification.

These registers can be directly accessed with the entire MCS instruction set, e.g. using the ORL instruction to set a specific bit.

13.3.1 MCS Internal Registers Overview

The table describes the MCS internal registers. Only parts of this register set can be accessed by the CPU:

Register Name	**Description**	**Details in Section**
R[x]	General purpose register x (x: 0..7), MCS Task internal register.	13.3.2
STA	Status register MCS Task internal register.	13.3.3
ACB	ARU Control Bit register MCS Task internal register.	13.3.4
CTRG	Clear Trigger Bits register MCS Global register from task view.	13.3.5
STRG	Set Trigger Bits register MCS Global register from task view.	13.3.6
TBU_TS0	TBU Timestamp TS0 register MCS Global register from task view.	13.3.7
TBU_TS1	TBU Timestamp TS1 register MCS Global register from task view.	13.3.8
TBU_TS2	TBU Timestamp TS2 register MCS Global register from task view.	13.3.9
MHB	Memory High Byte register	13.3.10

	MCS Task internal register.	

13.3.2 General purpose register R[x] (x:0...7)

Address Offset:	**0x0 + x**																**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 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.

13.3.3 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									0x0000					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: 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: The IRQ bit can only be cleared by CPU, by writing a 1 to the corresponding **MCS[i]_CH[x]_NOTIFY** register (see section 13.4.7).

Bit 2 **ERR**: Set Error Signal.
0: No Error occurred.
1: Error occurred.
Note: The ERR signal of an MCS-channel reflects Error status that may be caused by one of the following conditions:

- MCS-channel program sets the ERR signal by writing to this bit
- ECC RAM Error occurred while accessing the connected RAM pages (also disables MCS-channel by clearing bit EN)
- Decoding an instruction with an invalid opcode (also disables MCS-channel by clearing bit EN)

Note: If the GTM includes a MON sub module, the ERR signal is always captured by this module.
Note: An MCS-channel releases an error signal 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.

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.

Note: The ERR bit can only be cleared by CPU, by writing a 1 to the **MCS[i]_ERR** register (see section 13.4.16).

Bit 3 **MCA**: MON Activity signalling for MCS channel.
0: No activity signalled to sub module MON.
1: Activity signalled to sub module MON.

Note: When this bit is set the corresponding channel in the MON sub module register **MON_ACTIVITY** is set (see 20.8.2. 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: Last ARU transfer was not cancelled.
1: CPU cancelled last ARU transfer.
Note: This bit is updated after each ARU transfer and it should be evaluated immediately after the ARU instruction. Otherwise, the CPU could set the bit leading to a bad status information in the MCS program.

Bit 9 **CWT**: Cancel WURM instruction bit.
0: Last WURM instruction was not cancelled.
1: CPU cancelled 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: Non-blocking ARU transfer failed due to missing data.
1: Non-blocking ARU transfer finished successfully.

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 23:19 **Reserved:** Read as zero, should be written as zero.

Bit 31:24 n/a

Note: If both, the CPU and the MCS-channel are writing to the same register at the clock cycle, the value of the CPU is written to the register and the value of the MCS-channel is discarding.

Note: Writing to bits of the register STA with instructions that do implicitely 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 undesireable 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 or MOV STA, R1").

13.3.4 Register ACB

Address Offset:	**0x9**		**Initial Value:**	**0x01FE00**				
	31 30 29 28 27 26 25 24	23 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

13.3.5 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								Reserved								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
Initial Value									0x000000								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 : clear trigger bit

Bit 1 **TRG1:** trigger bit 1.
0 : READ: trigger bit is cleared / WRITE : do nothing
1 : READ: trigger bit is set / WRITE : clear trigger bit

Bit 2 **TRG2:** trigger bit 2.
0 : READ: trigger bit is cleared / WRITE : do nothing
1 : READ: trigger bit is set / WRITE : clear trigger bit

Bit 3 **TRG3:** trigger bit 3.
0 : READ: trigger bit is cleared / WRITE : do nothing
1 : READ: trigger bit is set / WRITE : clear trigger bit

Bit 4 **TRG4:** trigger bit 4.
0 : READ: trigger bit is cleared / WRITE : do nothing
1 : READ: trigger bit is set / WRITE : clear trigger bit

Bit 5 **TRG5:** trigger bit 5.
0 : READ: trigger bit is cleared / WRITE : do nothing
1 : READ: trigger bit is set / WRITE : clear trigger bit

Bit 6 **TRG6:** trigger bit 6.
0 : READ: trigger bit is cleared / WRITE : do nothing
1 : READ: trigger bit is set / WRITE : clear trigger bit

Bit 7 **TRG7:** trigger bit 7.
0 : READ: trigger bit is cleared / WRITE : do nothing
1 : READ: trigger bit is set / WRITE : clear trigger bit

Bit 8 **TRG8:** trigger bit 8.
0 : READ: trigger bit is cleared / WRITE : do nothing
1 : READ: trigger bit is set / WRITE : clear trigger bit

Bit 9 **TRG9:** trigger bit 9.

	0 : READ: trigger bit is cleared / WRITE : do nothing
	1 : READ: trigger bit is set / WRITE : clear trigger bit
Bit 10	**TRG10:** trigger bit 10.
	0 : READ: trigger bit is cleared / WRITE : do nothing
	1 : READ: trigger bit is set / WRITE : clear trigger bit
Bit 11	**TRG11:** trigger bit 11.
	0 : READ: trigger bit is cleared / WRITE : do nothing
	1 : READ: trigger bit is set / WRITE : clear trigger bit
Bit 12	**TRG12:** trigger bit 12.
	0 : READ: trigger bit is cleared / WRITE : do nothing
	1 : READ: trigger bit is set / WRITE : clear trigger bit
Bit 13	**TRG13:** trigger bit 13.
	0 : READ: trigger bit is cleared / WRITE : do nothing
	1 : READ: trigger bit is set / WRITE : clear trigger bit
Bit 14	**TRG14:** trigger bit 14.
	0 : READ: trigger bit is cleared / WRITE : do nothing
	1 : READ: trigger bit is set / WRITE : clear trigger bit
Bit 15	**TRG15:** trigger bit 15.
	0 : READ: trigger bit is cleared / WRITE : do nothing
	1 : READ: trigger bit is set / WRITE : clear trigger bit
	Note: The trigger bits TRGx (x = 0..15) 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.
Bit 23:16	**Reserved:** Read as zero, should be written as zero.
Bit 31:24	n/a

13.3.6 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								Reserved								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
Initial Value									0x000000								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 **Note**: The trigger bits TRGx (x = 0..15) 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.
Bit 23:16	**Reserved:** Read as zero, should be written as zero.
Bit 31:24	n/a

13.3.7 Register TBU_TS0

Address Offset:	0xC																Initial Value:	0xXXXXXX														
	31	30	29	28	27	26	25	24	23	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									0xXXXXXX																							

Bit 23:0	**TS**: Current TBU time stamp 0.
Bit 31:24	n/a

Note: Any write access to a time base register discards the written data. A write access to a time base register may be used to destroy an unused 24-bit data word of an ARU read transfer.

13.3.8 Register TBU_TS1

Address Offset:	0xD																Initial Value:								0xXXXXXX							
	31	30	29	28	27	26	25	24	23	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									0xXXXXXX																							

Bit 23:0 **TS**: Current TBU time stamp 1.
Bit 31:24 n/a

Note: Any write access to a time base register discards the written data. A write access to a time base register may be used to destroy an unused 24-bit data word of an ARU read transfer.

13.3.9 Register TBU_TS2

Address Offset:	0xE																Initial Value:								0xXXXXXX							
	31	30	29	28	27	26	25	24	23	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									0xXXXXXX																							

Bit 23:0 **TS**: Current TBU time stamp 2.
Bit 31:24 n/a

Note: Any write access to a time base register discards the written data. A write access to a time base register may be used to destroy an unused 24-bit data word of an ARU read transfer.

13.3.10 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									0x000000																0x0000							

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

13.4 MCS Configuration Registers

This section describes the configuration registers of the MCS sub module.
These registers can only be accessed by the CPU using AEI, but not within the MCS-channel using MCS instructions.

13.4.1 MCS Configuration Registers Overview

The table describes the MCS registers that are visible and accessible by the CPU:

Register Name	**Description**	**Details in Section**
MCS[i]_CH[x]_CTRL	MCS Channel control register (x: 0..7)	13.4.2
MCS[i]_CH[x]_ACB	MCS Channel ACB register (x: 0..7)	13.4.6
MCS[i]_CH[x]_PC	MCS Channel Program counter register (x: 0..7)	13.4.3
MCS[i]_CH[x]_R[y]	MCS Channel GPRx registers (x: 0..7; y: 0..7)	13.4.4
MCS[i]_CH[x]_IRQ_NOTIFY	MCS Channel x interrupt notification	13.4.7

	register (x: 0..7)	
MCS[i]_CH[x]_IRQ_EN	MCS Channel x interrupt enable register (x: 0..7)	13.4.8
MCS[i]_CH[x]_IRQ_FORCINT	MCS Channel x software interrupt generation register (x: 0..7)	13.4.9
MCS[i]_CH[x]_IRQ_MODE	IRQ mode configuration register (x=0...7)	13.4.10
MCS[i]_CH[x]_EIRQ_EN	MCS Channel x error interrupt enable register (x: 0..7)	13.4.11
MCS[i]_CTRL	MCS Control register	13.4.12
MCS[i]_CTRG	MCS Clear trigger control register	13.4.13
MCS[i]_STRG	MCS Set trigger control register	13.4.14
MCS[i]_RST	MCS Channel reset register	13.4.15
MCS[i]_ERR	MCS Error register	13.4.16

13.4.2 Register MCS[i]_CH[x]_CTRL (x: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	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: In order 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: Non-blocking ARU transfer failed due to missing data. 1: Non-blocking ARU transfer finished successfully.
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.

13.4.3 Register MCS[i]_CH[x]_PC (x:0...7)

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	0x000																		0x000000+4*x													

Bit 13:0 **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.

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

13.4.4 Register MCS[i]_CH[x]_R[y] (x:0...7, y:0...5, 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 MCS general purpose register R[y].
Bit 31:24 **Reserved:** Read as zero, should be written as zero.
Note: This register is the same as described in 13.3.2.

13.4.5 Register MCS[i]_CH[x]_R6

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								RPw																							
Initial Value	0x00								0x000000																							

Bit 23:0	**DATA**: Data of MCS general purpose register R[y].
Bit 31:24	**Reserved:** Read as zero, should be written as zero.
	Note: This register is write protected during an active ARDI or NARDI instruction.

13.4.6 Register MCS[i]_CH[x]_ACB (x: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																											ACB4	ACB3	ACB2	ACB1	ACB0
Mode	R																											R	R	R	R	R
Initial Value	0x0000																											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.

13.4.7 Register MCS[i]_CH[x]_IRQ_NOTIFY (x: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																													MEM_ERR_IRQ	STK_ERR_IRQ	MCS_IRQ
Mode	R																													RCw	RCw	RCw
Initial Value	0x0000 0000																													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 **MEM_ERR_IRQ:** Memory access out of range in channel x.
0 = No IRQ released
1 = MCS-channel request a memory location out of range.
Note: This bit will be cleared on a CPU write access with a value '1'. A read access leaves the bit unchanged.
Note: In the case of a memory overflow, any read or write access to the RAM is always blocked. The read data is unpredictable.

Note: It should be noted that in the case of a memory overflow, the MCS channel will continue. Thus it is recommended that the CPU immediately stops the MCS channel.

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

13.4.8 Register MCS[i]_CH[x]_IRQ_EN (x: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																													MEM_ERR_IRQ_EN	STK_ERR_IRQ_EN	MCS_IRQ_EN
Mode	R																													RW	RW	RW
Initial Value	0x0000000																													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 **MEM_ERR_IRQ_EN:** MCS channel x MEM_ERR_IRQ interrupt enable
0 = Disable interrupt
1 = Enable interrupt

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

13.4.9 Register MCS[i]_CH[x]_IRQ_FORCINT (x: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																													TRG_MEM_ERR_IRQ	TRG_STK_ERR_IRQ	TRG_MCS_IRQ
Mode	R																													RAw	RAw	RAw
Initial Value	0x000000																													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_MEM_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.

13.4.10 Register MCS[i]_CH[x]_IRQ_MODE (x:0...7)

Address Offset:	**see Appendix B**	**Initial Value:**	**0x0000_000X**
	31 … 2		1 0
Bit	Reserved		IRQ_MODE
Mode	R		RW
Initial Value	0x0000 0000		XX

Bit 1:0	**IRQ_MODE**: IRQ mode selection 00 = Level mode 01 = Pulse mode 10 = Pulse-Notify mode 11 = 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

13.4.11 Register MCS[i]_CH[x]_EIRQ_EN (x: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																													MEM_ERR_EIRQ_EN	STK_ERR_EIRQ_EN	MCS_EIRQ_EN
Mode	R																													RW	RW	RW
Initial Value	0x0000000																													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	**MEM_ERR_EIRQ_EN:** MCS channel x MEM_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.

13.4.12 Register MCS[i]_CTRL

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															RAM_RST	Reserved														HLT_SP_OFL	SCHED
Mode	R															RPw	R														RW	RW
Initial Value	0x0000															X	0x0000														0	0

Bit 0	**SCHED:** MCS submodule scheduling scheme 0 = Accelerated scheduling scheme. 1 = Round-Robin scheduling scheme.

Bit 1 **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 15:2 **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 values 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 31:17 **Reserved:** Read as zero, should be written as zero.

13.4.13 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																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
Initial Value	0x000000																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 : clear trigger bit

Bit 1 **TRG1:** trigger bit 1.
0 : READ: trigger bit is cleared / WRITE : do nothing

	1 : READ: trigger bit is set / WRITE : clear trigger bit
Bit 2	**TRG2:** trigger bit 2. 0 : READ: trigger bit is cleared / WRITE : do nothing 1 : READ: trigger bit is set / WRITE : clear trigger bit
Bit 3	**TRG3:** trigger bit 3. 0 : READ: trigger bit is cleared / WRITE : do nothing 1 : READ: trigger bit is set / WRITE : clear trigger bit
Bit 4	**TRG4:** trigger bit 4. 0 : READ: trigger bit is cleared / WRITE : do nothing 1 : READ: trigger bit is set / WRITE : clear trigger bit
Bit 5	**TRG5:** trigger bit 5. 0 : READ: trigger bit is cleared / WRITE : do nothing 1 : READ: trigger bit is set / WRITE : clear trigger bit
Bit 6	**TRG6:** trigger bit 6. 0 : READ: trigger bit is cleared / WRITE : do nothing 1 : READ: trigger bit is set / WRITE : clear trigger bit
Bit 7	**TRG7:** trigger bit 7. 0 : READ: trigger bit is cleared / WRITE : do nothing 1 : READ: trigger bit is set / WRITE : clear trigger bit
Bit 8	**TRG8:** trigger bit 8. 0 : READ: trigger bit is cleared / WRITE : do nothing 1 : READ: trigger bit is set / WRITE : clear trigger bit
Bit 9	**TRG9:** trigger bit 9. 0 : READ: trigger bit is cleared / WRITE : do nothing 1 : READ: trigger bit is set / WRITE : clear trigger bit
Bit 10	**TRG10:** trigger bit 10. 0 : READ: trigger bit is cleared / WRITE : do nothing 1 : READ: trigger bit is set / WRITE : clear trigger bit
Bit 11	**TRG11:** trigger bit 11. 0 : READ: trigger bit is cleared / WRITE : do nothing 1 : READ: trigger bit is set / WRITE : clear trigger bit
Bit 12	**TRG12:** trigger bit 12. 0 : READ: trigger bit is cleared / WRITE : do nothing 1 : READ: trigger bit is set / WRITE : clear trigger bit
Bit 13	**TRG13:** trigger bit 13. 0 : READ: trigger bit is cleared / WRITE : do nothing 1 : READ: trigger bit is set / WRITE : clear trigger bit
Bit 14	**TRG14:** trigger bit 14. 0 : READ: trigger bit is cleared / WRITE : do nothing 1 : READ: trigger bit is set / WRITE : clear trigger bit
Bit 15	**TRG15:** trigger bit 15. 0 : READ: trigger bit is cleared / WRITE : do nothing 1 : READ: trigger bit is set / WRITE : clear trigger bit **Note**: The trigger bits TRGx (x = 0..15) 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: A write access to **MCS[i]_CTRG** may take up to 13 clock cycles, since the write access is scheduled to the next CPU time slot determined by the MCS scheduler.

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

13.4.14 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																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
Initial Value	0x0000 00																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 **Note**: The trigger bits TRGx (x = 0..15) 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: A write access to **MCS[i]_STRG** may take up to 13 clock cycles, since the write access is scheduled to the next CPU time slot determined by the MCS scheduler.

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

13.4.15 Register MCS[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								CWT7	CWT6	CWT5	CWT4	CWT3	CWT2	CWT1	CWT0	CAT7	CAT6	CAT5	CAT4	CAT3	CAT2	CAT1	CAT0	RST7	RST6	RST5	RST4	RST3	RST2	RST1	RST0
Mode	R								RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RAw	RAw	RAw	RAw	RAw	RAw	RAw	RAw
Initial Value	0x0000 00								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 **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 ...7) bits is cleared automatically after write access of CPU. All channel related registers are set to their reset values and channel operation is stopped immediately. The reset action **RSTx** has a higher priority than setting the bits **CWTx** or **CATx** (x = 0 ...7).

Note: Channel related registers 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 **CTRG/STRG**).

Bit 8	**CAT0:** Cancel ARU transfer for channel 0. 0 = Do nothing. 1 = Cancel any pending ARU read or write transfer.
Bit 9	**CAT1:** Cancel ARU transfer for channel 1. 0 = Do nothing. 1 = Cancel any pending ARU read or write transfer.
Bit 10	**CAT2:** Cancel ARU transfer for channel 2. 0 = Do nothing. 1 = Cancel any pending ARU read or write transfer.
Bit 11	**CAT3:** Cancel ARU transfer for channel 3. 0 = Do nothing. 1 = Cancel any pending ARU read or write transfer.
Bit 12	**CAT4:** Cancel ARU transfer for channel 4. 0 = Do nothing. 1 = Cancel any pending ARU read or write transfer.
Bit 13	**CAT5:** Cancel ARU transfer for channel 5. 0 = Do nothing. 1 = Cancel any pending ARU read or write transfer.
Bit 14	**CAT6:** Cancel ARU transfer for channel 6. 0 = Do nothing. 1 = Cancel any pending ARU read or write transfer.
Bit 15	**CAT7:** Cancel ARU transfer for channel 7. 0 = Do nothing. 1 = Cancel any pending ARU read or write transfer. **Note**: The CATx (x = 0 ...7) bit inside the **STA** register of the corresponding MCS-channel is set and any pending ARU read or write request is cancelled. The MCS-channel resumes with the instruction after the ARU transfer instruction. **Note**: The CATx (x = 0 ...7) bit is cleared by the corresponding MCS channel, when the channel reaches an ARU read or write instruction.
Bit 16	**CWT0:** Cancel WURM instruction for channel 0. 0 = Do nothing. 1 = Cancel any pending WURM instruction.
Bit 17	**CWT1:** Cancel WURM instruction for channel 1. 0 = Do nothing. 1 = Cancel any pending WURM instruction.
Bit 18	**CWT2:** Cancel WURM instruction for channel 2. 0 = Do nothing. 1 = Cancel any pending WURM instruction.
Bit 19	**CWT3:** Cancel WURM instruction for channel 3. 0 = Do nothing.

	1 = Cancel any pending WURM instruction.
Bit 20	**CWT4:** Cancel WURM instruction for channel 4. 0 = Do nothing. 1 = Cancel any pending WURM instruction.
Bit 21	**CWT5:** Cancel WURM instruction for channel 5. 0 = Do nothing. 1 = Cancel any pending WURM instruction.
Bit 22	**CWT6:** Cancel WURM instruction for channel 6. 0 = Do nothing. 1 = Cancel any pending WURM instruction.
Bit 23	**CWT7:** Cancel WURM instruction for channel 7. 0 = Do nothing. 1 = Cancel any pending WURM instruction. **Note**: The CWTx (x = 0 ...7) bit inside the **STA** register of the corresponding MCS-channel is set and any pending WURM instruction is cancelled. The MCS-channel resumes with the instruction after the WURM instruction. **Note**: The CWTx (x = 0 ...7) bit is cleared by the corresponding MCS channel, when the channel reaches a WURM instruction.
Bit 31:24	**Reserved:** Read as zero, should be written as zero.

13.4.16 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	0x0000																								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-channel3 . 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...7) bits in order to determine the current error state of the corresponding MCS-channel x. 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.
Bit 31:8	**Reserved:** Reserved **Note**: Read as zero, should be written as zero

14 Memory Configuration (MCFG)

14.1 Overview

The Memory Configuration submodule (MCFG) is an infrastructure module that organizes physical memory blocks and maps them to the instances of Multi Channel Sequencer (MCS) submodules.

The default configuration maps a memory of size 1K*32 bit = 4KB to MCS memory page 0 and a memory of size 0.5K*32 bit = 2 KB to MCS memory page 1.

In order to support different memory sizes for different MCS instances, the MCFG module provides two additional layout configurations for reorganization of memory pages between neighbouring MCS modules. Figure 14.1.1 shows all layout configurations.

The layout configuration SWAP is swapping the 2KB memory page of the current MCS instance with the 4KB memory page of the successive MCS instance. Thus, the memory of the current MCS module is increased by 2KB but the memory of the successor is decreased by 2KB.

The layout configuration BORROW is borrowing the 4KB memory page of the successive MCS instance for the current instance. Thus, the memory of the current MCS module is increased by 4KB but the memory of the successor is decreased by 4KB.

It should be noted that the successor of the last MCS instance is the first MCS instance MCS0.
The actual size of the memory pages for an MCS instance depends on the layout configuration for the current instance MCS[i] and the layout configuration of the preceding memory instance MCS[i-1].

Table 14.1.2 summarizes the layout parameters MP0 and MP1 for MCS instance MCS[i].
The addressing of memory page 0 ranges from 0 to MP0-4 and the addressing of memory page 1 ranges from MP0 to MP1-4.

Besides these software related Layout configurations, the MCFG submodule has an additional input port *MCS_RAM1_EN_ADDR_MSB* which is routed to the top level of the GTM-IP.

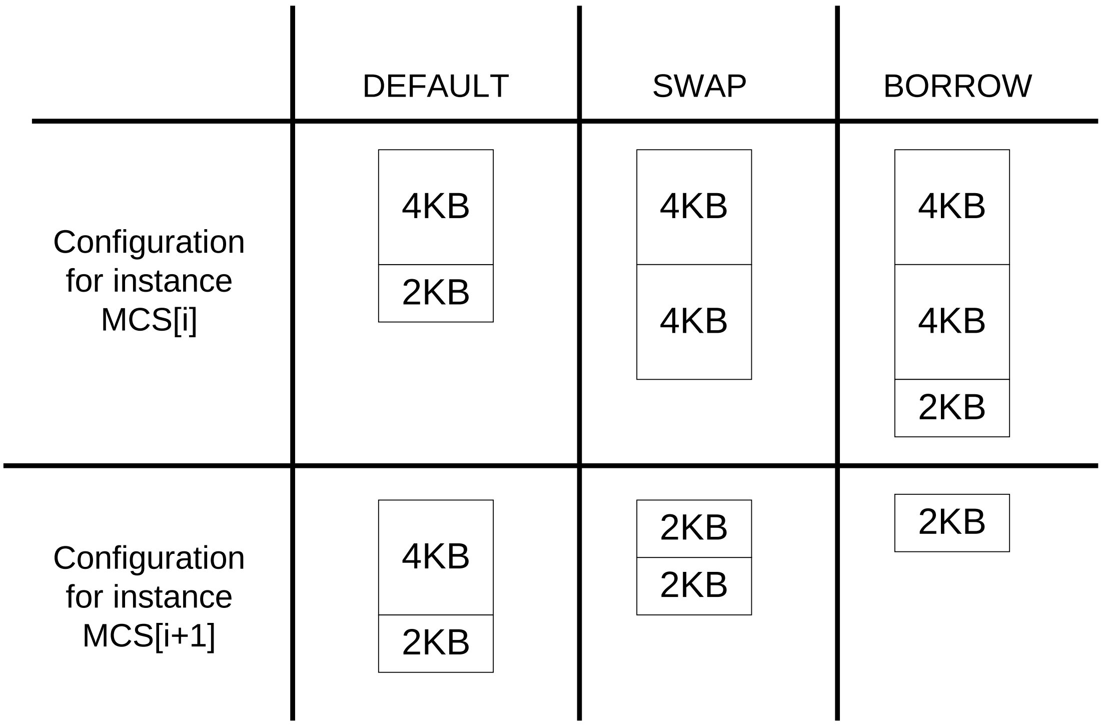
The above mentioned behaviour of the MCFG submodule is applied if this port is connected to a constant logic level of zero (0).

If a constant logic level of one (1) is applied to this port, the MCFG submodule assumes that a memory of size 1K*32 bit = 4KB is also mapped to MCS memory page 1.

In this case the memory layout configurations of the MCFG submodule change as shown in Figure 14.1.3 and the memory layout parameters are modified according to Figure 14.1.4.

This document assumes that the GTM implementation embeds 7 MCS instances. However, the actual number of implemented MCS instances can be obtained from [1].

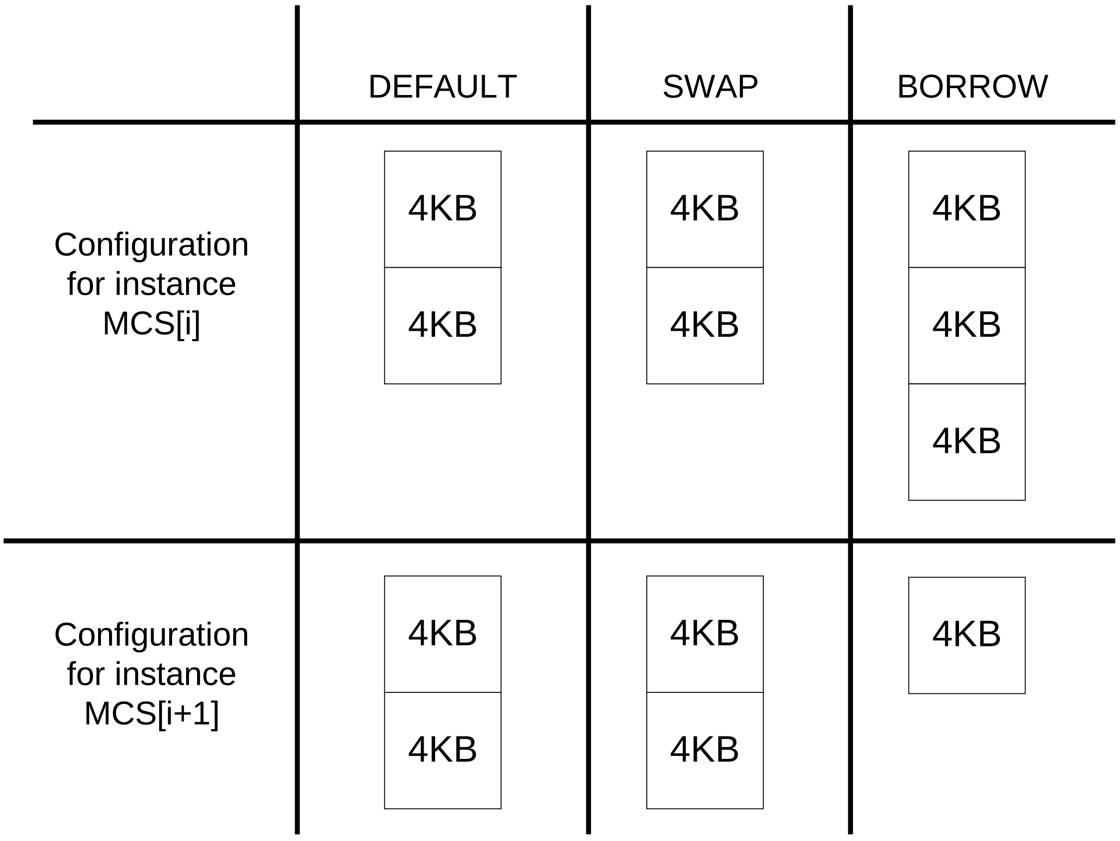
14.1.1 Memory Layout Configurations (MCS_RAM1_EN_ADDR_MSB = 0)


DEFAULT
SWAP
BORROW
Configuration for instance MCS[i]
4KB
2KB
4KB
4KB
4KB
4KB
2KB
Configuration for instance MCS[i+1]
4KB
2KB
2KB
2KB
2KB

14.1.2 Memory Layout Parameters (MCS_RAM1_EN_ADDR_MSB = 0)

			Memory Layout Configuration of preceding MCS instance MCS[i-1]		
			DEFAULT	**SWAP**	**BORROW**
Memory Layout Configuration of current MCS instance MCS[i]	**DEFAULT**	MP0	0x1000	0x800	0x0
		MP1	0x1800	0x1000	0x800
	SWAP	MP0	0x1000	0x800	0x0
		MP1	0x2000	0x1800	0x1000
	BORROW	MP0	0x1000	0x800	0x0
		MP1	0x2800	0x2000	0x1800

14.1.3 Memory Layout Configurations (MCS_RAM1_EN_ADDR_MSB = 1)


DEFAULT
SWAP
BORROW
Configuration for instance MCS[i]
4KB
4KB
4KB
4KB
4KB
4KB
4KB
Configuration for instance MCS[i+1]
4KB
4KB
4KB
4KB
4KB

14.1.4 Memory Layout Parameters (MCS_RAM1_EN_ADDR_MSB = 1)

			Memory Layout Configuration of preceding MCS instance MCS[i-1]		
			DEFAULT	SWAP	BORROW
Memory Layout Configuration of current MCS instance MCS[i]	DEFAULT	MP0	0x1000	0x1000	0x0
		MP1	0x2000	0x2000	0x1000
	SWAP	MP0	0x1000	0x1000	0x0
		MP1	0x2000	0x2000	0x1000
	BORROW	MP0	0x1000	0x1000	0x0
		MP1	0x3000	0x3000	0x2000

14.2 MCFG Configuration Registers

This section describes the configuration registers of the MCFG submodule.

Register Name	Description	Details in Section
MCFG_CTRL	Memory layout configuration.	14.3

14.3 Register MCFG_CTRL

Address Offset:	see Appendix B	Initial Value: 0x00000000						
	31 … 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
Bit	Reserved	MEM6	MEM5	MEM4	MEM3	MEM2	MEM1	MEM0
Mode	R	RW	RW	RW	RW	RW	RW	RW
Initial Value	0x0000 00	00	00	00	00	00	00	00

Bit 1:0 **MEM0**: Configure Memory pages for MCS-instance MCS0.
00: DEFAULT configuration
01: SWAP configuration
10: BORROW configuration

	11: DEFAULT configuration
Bit 3:2	**MEM1**: Configure Memory pages for MCS-instance MCS1.
	00: DEFAULT configuration
	01: SWAP configuration
	10: BORROW configuration
	11: DEFAULT configuration
Bit 5:4	**MEM2**: Configure Memory pages for MCS-instance MCS2.
	00: DEFAULT configuration
	01: SWAP configuration
	10: BORROW configuration
	11: DEFAULT configuration
Bit 7:6	**MEM3**: Configure Memory pages for MCS-instance MCS3.
	00: DEFAULT configuration
	01: SWAP configuration
	10: BORROW configuration
	11: DEFAULT configuration
Bit 9:8	**MEM4**: Configure Memory pages for MCS-instance MCS4.
	00: DEFAULT configuration
	01: SWAP configuration
	10: BORROW configuration
	11: DEFAULT configuration
Bit 11:10	**MEM5**: Configure Memory pages for MCS-instance MCS 5.
	00: DEFAULT configuration
	01: SWAP configuration
	10: BORROW configuration
	11: DEFAULT configuration
Bit 13:12	**MEM6**: Configure Memory pages for MCS-instance MCS6.
	00: DEFAULT configuration
	01: SWAP configuration
	10: BORROW configuration
	11: DEFAULT configuration
Bit 31:14	**Reserved:** Read as zero, should be written as zero.

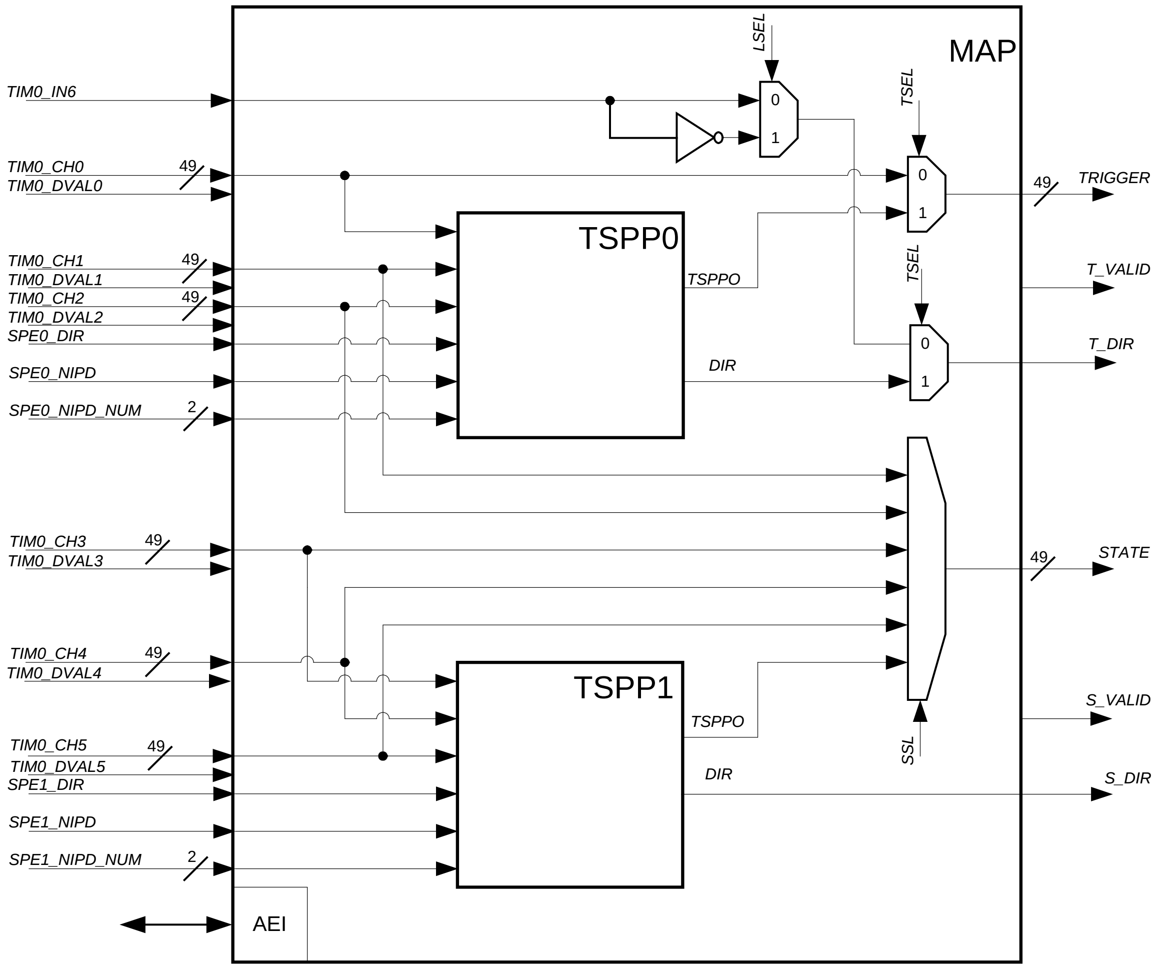
NOTE: It should be noted that the actual GTM-IP implementation may embed less than 7 MCS instances (see [1]). In this case this register only implements the register bits for available MCS instances.

15 TIM0 Input Mapping Module (MAP)

15.1 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 15.1.1.

15.1.1 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 17.

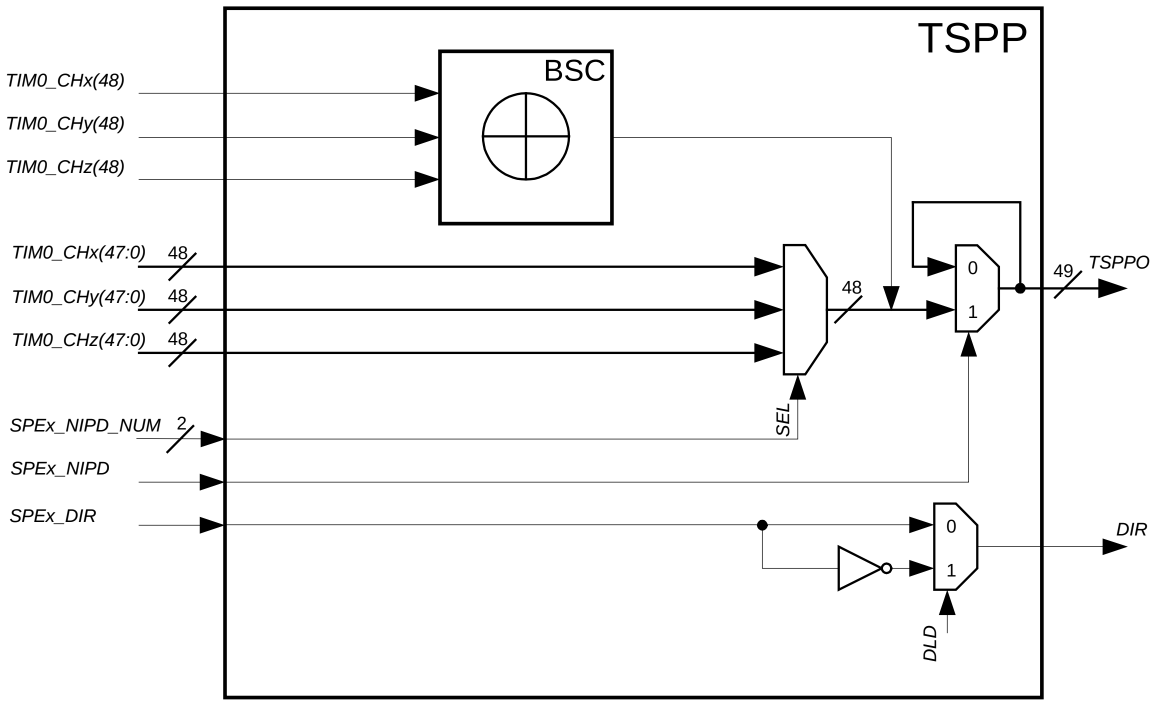
There, the signal *TRIGGER* is generated in subunit TSPP0 out of the TIM0 signals coming from channel 0 up to 2.
The signal *STATE* is generated in subunit TSPP1 out of the TIM signals coming from channel 3 up to channel 5.
This is only be done, when the TSSPx subunits are enabled and when the *SPEx_NIPD* 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)

15.2 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 15.2.1.

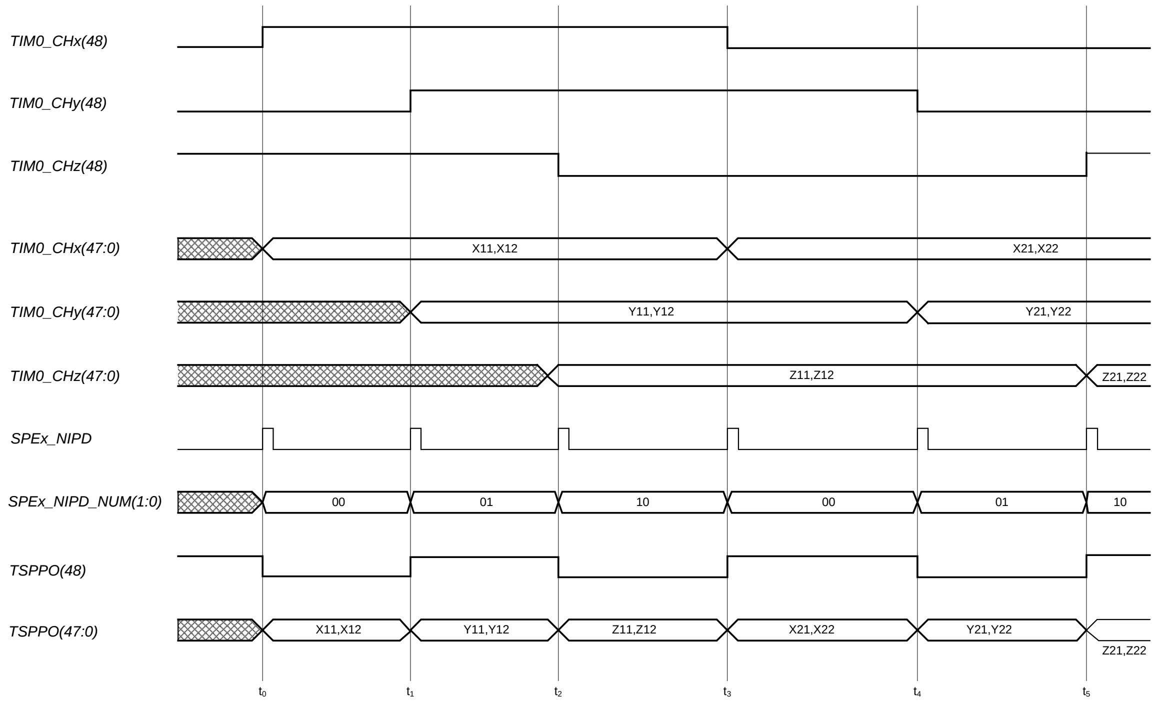
15.2.1 TIM Signal Preprocessing (TSPP) subunit architecture



15.2.2 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.

15.2.2.1 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.

15.3 MAP Register overview

The following table gives an overview about the MAP registers

Register name	**Description**	**Details in Section**
MAP_CTRL	MAP Control register	15.4.1

15.4 MAP Register description

15.4.1 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. 000: *TIM0_CH1* selected as *STATE* output signal. 001: *TIM0_CH2* selected as *STATE* output signal. 010: *TIM0_CH3* selected as *STATE* output signal. 011: *TIM0_CH4* selected as *STATE* output signal. 100: *TIM0_CH5* selected as *STATE* output signal. 101: *TSPP1_TSPPO* selected as *STATE* output signal. 110: same as '000' 111: 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.

16 Digital PLL Module (DPLL)

16.1 Overview

The digital PLL (DPLL) submodule 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 valid 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 16.6
- generation of SUB_INC1,2 pulses for up to 2 position counters in normal or emergency mode (see chapter 16.8.3)
- synchronization of the actual position (under CPU control, see chapter 16.8.6.2)
- possibility of seamless switch to emergency mode and back under CPU control, see configuration register DPLL_CTRL_0 at chapter 16.11
- prediction of position and time related events in chapter 16.7

16.2 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 16.11). Because a switch in emergency mode can appear suddenly, the information of the last increment durations 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 submodule (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 16.4.1) or 8 times higher, selected by configuration bits in the DPLL_CTRL_1 register (see chapter 16.11.2). 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 16.11.2).

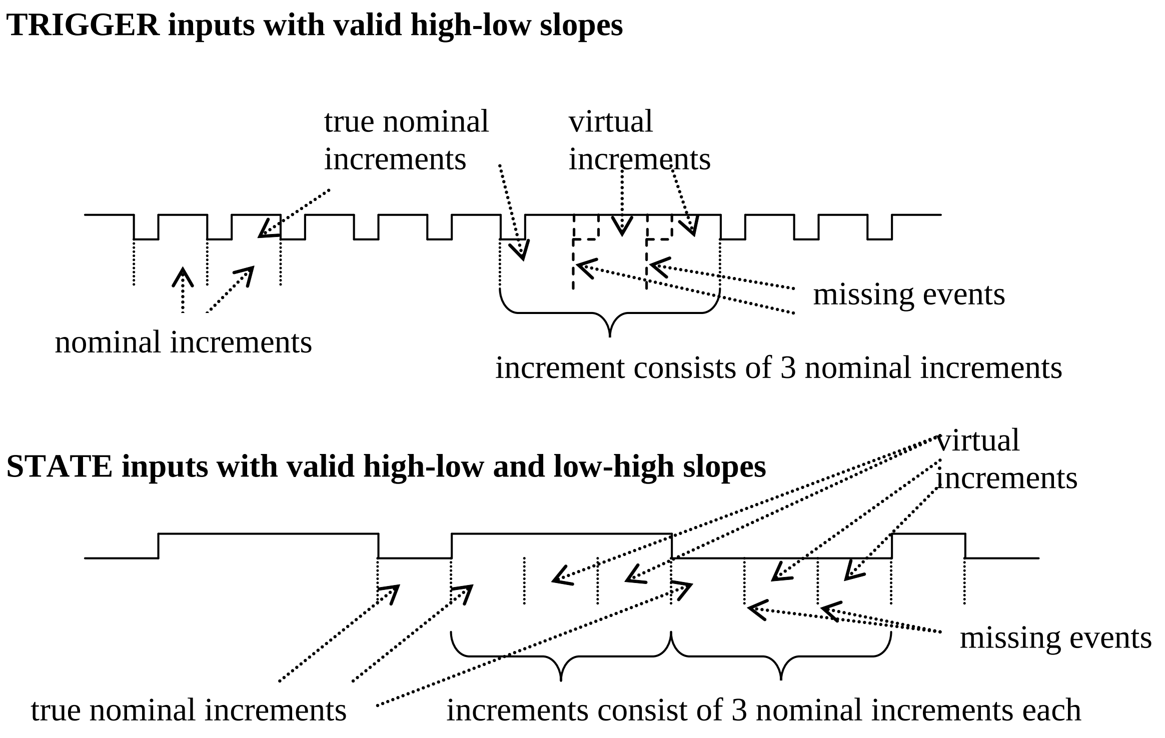
Dependent on configuration different strategies can be used to correct a wrong pulse number.

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 16.11).
For synchronization purposes some *TRIGGER/STATE* input signals can be suppressed in dependency on the current position. Therefore an increment as duration between two valid input events can be either a nominal increment or it can consist of more then one nominal increment.

While a true nominal increment starts with a valid 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).

16.3 Input signal courses

Typical input signal courses are shown in the figure below.


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 and low-high slopes
virtual increments
missing events
true nominal increments
increments consist of 3 nominal increments each

16.4 Block and interface description

The block description of the DPLL is shown in the following figure.

16.4.1 DPLL Block diagram

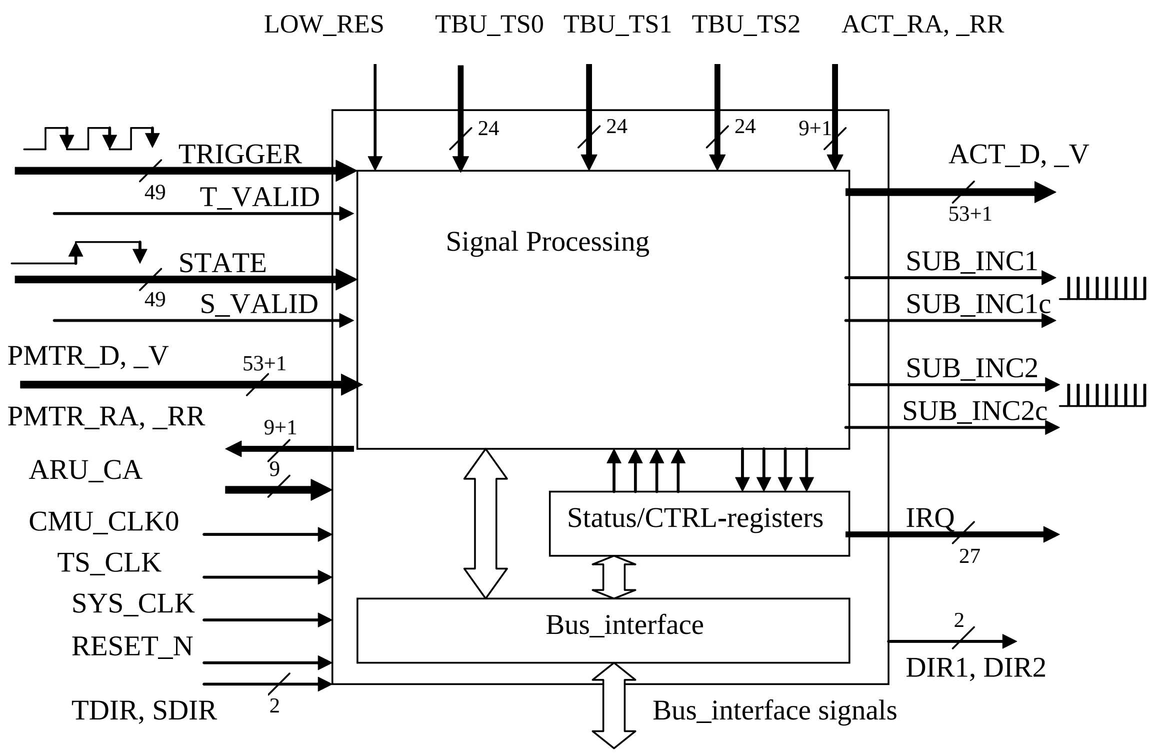


Table 16.4.2 summarizes the interface signals of the DPLL shown by the block diagram above.

16.4.2 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 synchronisation 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 value PSAC[i] for the action in relation to TBU_TS1 or 2 $^{6)}$ and ACT_D(23:0) is the time stamp value	Future time stamp with the resolution as TBU_TS0 input, additional position information and additional control bits;

			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 an other 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 witch 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

For references above the following hints are used:
1) see DPLL_CTRL_x register, x=2,3,4; see 16.11.3,16.11.4,16.11.5
2) see DPLL_STATUS register; see 16.11.30
3) see DPLL_ACT_STA register; see 16.11.7
4) see DPLL_CTRL_0 register; see 16.11.1
5) see DPLL_CTRL_1 register; see 16.11.2
6) see DPLL input signal description; see 16.1

16.5 DPLL Architecture

16.5.1 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 durations 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 utilisation of such calculations actions for the future can be predicted.

16.5.2 Explanation of the prediction methodology

As already shown in chapter 16.1 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 16.6.

16.5.3 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.

16.5.4 Clock generation

The clock is generated outside the DPLL.

16.5.5 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.

16.5.6 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 *TRIGGERS* are divided into the corresponding number of nominal increments whereas the duration of a nominal increment is the greatest divider of all increment durations.

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 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) is multiplied by NT first before the PD value is added.
The NT value of the current increment is stored in the variable SYN_T (see **NUTC** register in chapter 16.11.14).

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 increment durations.
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 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 is multiplied by NS first before the PD_S value is added.
The current NS value is stored in the variable SYN_S (see **NUSC** register in chapter 16.11.15).

For references above the following hints are used:
1) see DPLL_CTRL_x register, x=2,3,4; see 16.11.3,16.11.4,16.11.5
2) see DPLL_STATUS register; see 16.11.30
3) see DPLL_ACT_STA register; see 16.11.7
4) see DPLL_CTRL_0 register; see 16.11.1
5) see DPLL_CTRL_1 register; see 16.11.2
6) see DPLL input signal description; see 16.1

16.5.7 DPLL Architecture overview

As shown in 16.4.1 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 millimetre 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 centimetres 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 a 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 millimetre or centimetre 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 synchronisation purposes some *TRIGGER* marks can be omitted in order to set the system to a proper synchronisation 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 16.2) 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 16.11.3,16.11.4,16.11.5
2) see DPLL_STATUS register; see 16.11.30
3) see DPLL_ACT_STA register; see 16.11.7
4) see DPLL_CTRL_0 register; see 16.11.1
5) see DPLL_CTRL_1 register; see 16.11.2
6) see DPLL input signal description; see 16.1

16.5.8 DPLL Architecture description

The DPLL block diagram 16.4.1 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 16.11).
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 16.5.6 is stored in the RAM region 2c (see chapter 16.12.3). 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 16.11.14).

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 16.12.3) 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. 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 16.11) .
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 16.11.79).

For the systematic arrangement of *STATE* inputs **the profile** (as already mentioned in chapter 16.5.6 is stored in the RAM region 1c3 (see chapter 16.11.94). 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 16.11.15).

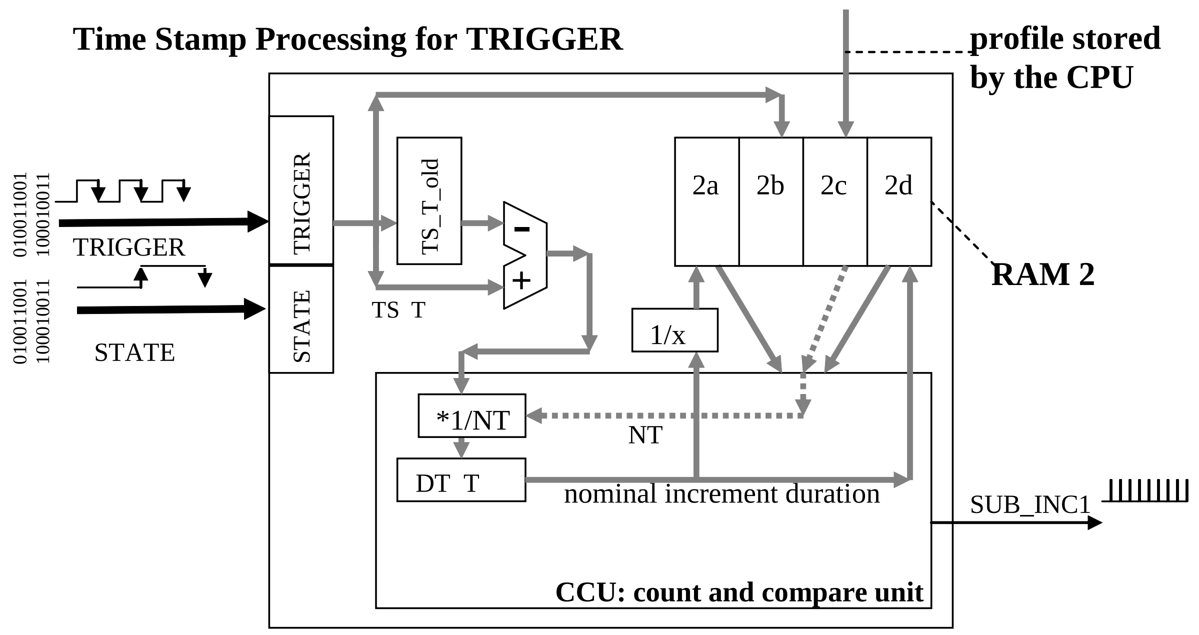
In emergency mode the physical deviation values PD_S in the ADT_S field could be used to balance the local systematic inaccuracy of the *STATE* signal. The value of PD_S (see chapter 16.11.94) is the pulse difference in the corresponding increment and does mean the number of sub pulses to be added to the nominal number of pulses. 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 16.11.3,16.11.4,16.11.5
[2] see DPLL_STATUS register; see 16.11.30
[3] see DPLL_ACT_STA register; see 16.11.7
[4] see DPLL_CTRL_0 register; see 16.11.1
[5] see DPLL_CTRL_1 register; see 16.11.2
[6] see DPLL input signal description; see 16.1

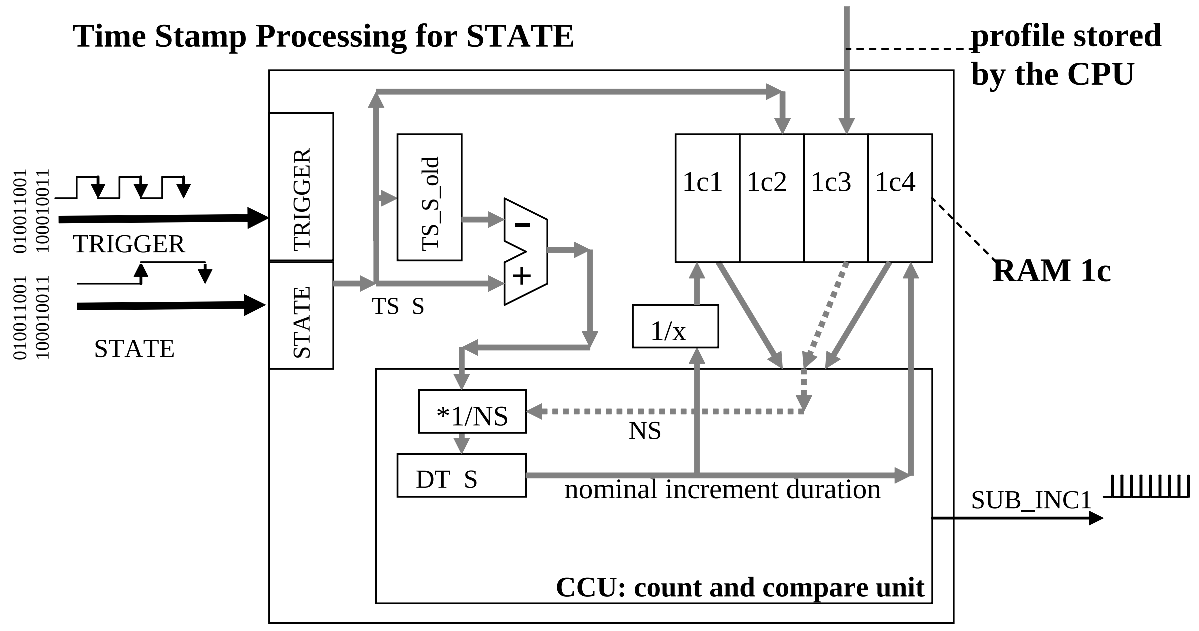
16.5.9 Block diagrams of time stamp processing.


Time Stamp Processing for TRIGGER
profile stored by the CPU
010011001 100010011
TRIGGER
010011001 100010011
STATE
TRIGGER
STATE
TS_T_old
TS T
-
+
2a
2b
2c
2d
RAM 2
1/x
*1/NT
NT
DT T
nominal increment duration
SUB_INC1
CCU: count and compare unit

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 valid input event TSF_T
2c	is used for the profile ADT_T and
2d	for the nominal increment durations 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 an other increment. In order not to be forced to distinguish between gaps and "normal" increment durations 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.



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 valid input event TSF_S
1c3	is used for the profile ADT_S and
1c4	for the nominal increment durations DT_S.

The calculations are performed similar as for the *TRIGGER* input. The NS value in the profile shows the appearance of a gap.

16.5.10 Register and RAM address overview

The address map of the DPLL is divided into register and memory regions as defined in Table 16.5.10.1. 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 16.5.10.1 gives the DPLL Address map overview

16.5.10.1 Register and RAM address map

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 (chapter16.10) while the bit positions of the status and control registers are described in detail in chapter 16.11.

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 16.11.2). 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.

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]	adapt 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]	adapt values of *TRIGGER*	2c	
0x4600…7000	0x47FC…7FFC	128…1024	384…3072	DT_T[i]	nom. *TRIGGER* increments	2d	

16.5.10.2 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 durations for 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

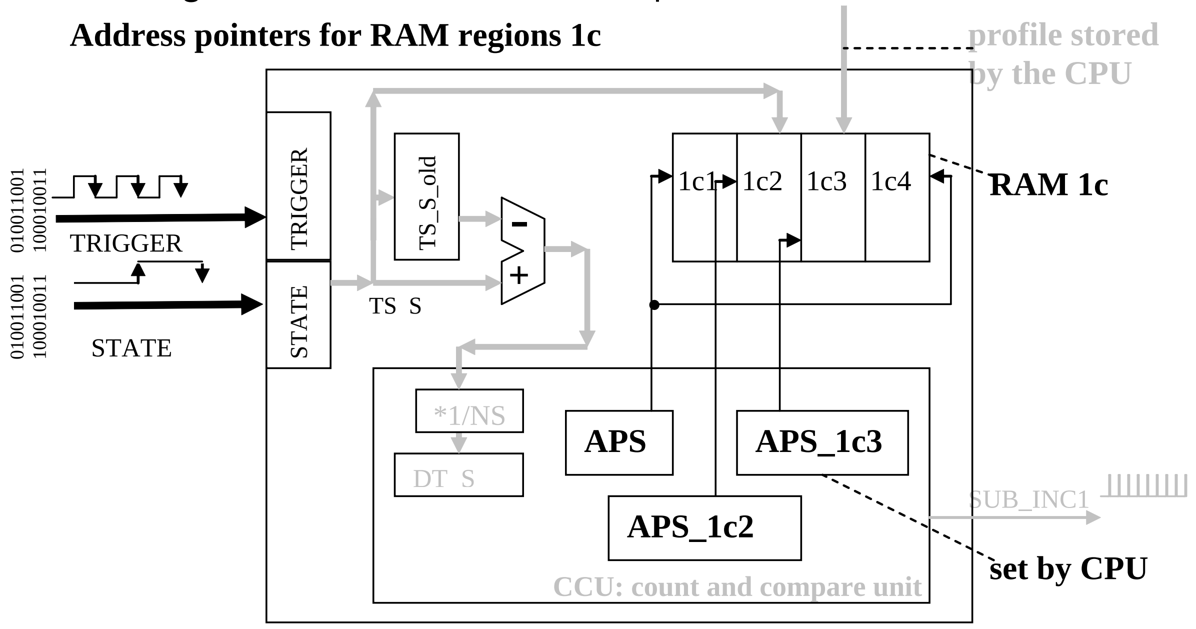
- **RAM Region 1c**: Values of all *STATE* increments in FULL_SCALE, data width of 3 bytes used for 24 bit values;

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 then 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.



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 16.8.6.2).
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 16.11.30) by the SYS bit.

16.5.10.3 RAM Region 2

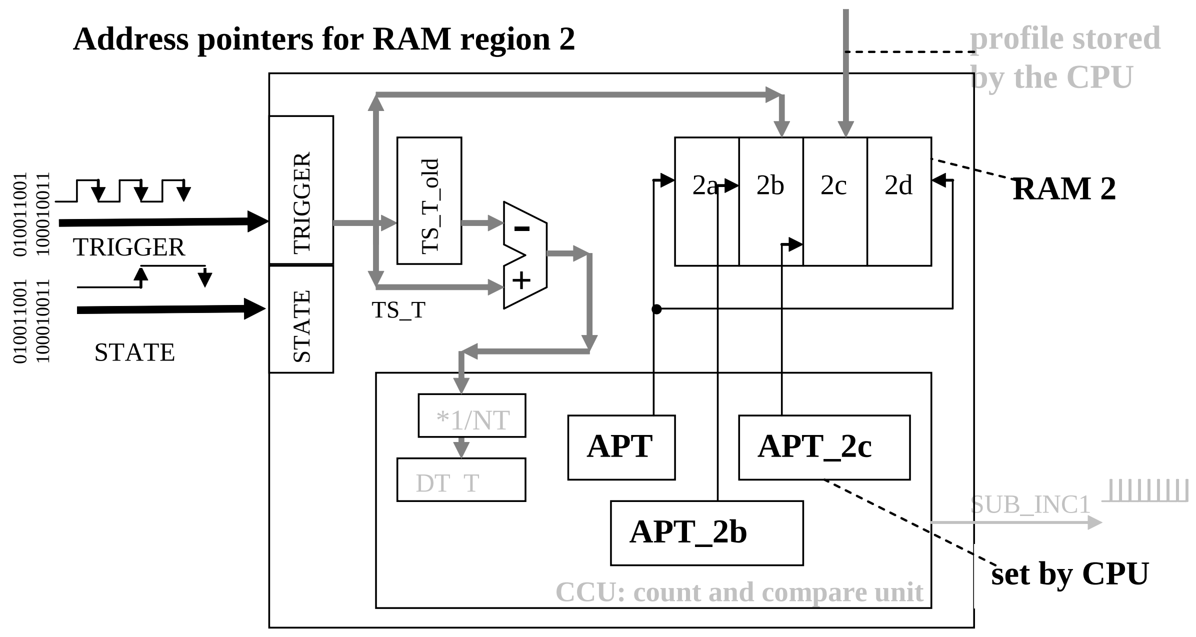
The RAM region 2 has a configurable size of 1,5 to 12 Kbytes and is used to store measured and calculated values for increments of *TRIGGER*. The address information is explained in chapter 16.10 while the meaning is explained in this chapter.

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 then the last increment duration for the action prediction by setting NUTE to a value greater then one.



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 16.8.6.2).

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 16.11.30) by the SYT bit.

16.6 Prediction of the current increment duration

16.6.1 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.

16.6.2 Increment prediction in Normal Mode forwards (DIR1=0)

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 16.11.14) 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.

16.6.2.1 *Equations DPLL-1a to calculate TRIGGER time stamps*

For calculation of time stamps use the filter delay information

TS_T= TRIGGER_TS (unchanged for IDT=0) (DPLL-1a0)
TS_T= TS_T - 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 - 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 - 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 16.7.5.1.

16.6.2.2 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.

16.6.2.3 Equation DPLL-1c to calculate RDT_T_ACT (nominal value)

RDT_T_ACT= 1/ DT_T_ACT (DPLL-1c)

16.6.2.4 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 a 18 bit fractional part.

Note: QDT_T_ACT uses a 6 bit integer part and a 18 bit fractional part.

16.6.2.5 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] 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 a 18 bit fractional part.

Note: QDT_T[p-q] uses a 6 bit integer part and a 18 bit fractional part.

16.6.2.6 Equation DPLL-4 to calculate the weighted average error

for SYT=1 calculate:

MEDT_T := (EDT_T + MEDT_T)/2 (DPLL-4)

16.6.2.7 Equations DPLL-5 to calculate the current increment value

nominal increment value:
CDT_TX _nom = (DT_T_ACT + MEDT_T)* QDT_T[p-q+1] (DPLL-5a)
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 a 18 bit fractional part.

and the expected duration to the next *TRIGGER* event
CDT_TX = CDT_TX _nom * SYN_T (DPLL-5b)

Note: QDT_T[p-q+1] uses a 6 bit integer part and a 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.

16.6.3 Increment prediction in Emergency Mode forwards (DIR2=0)

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.

16.6.3.1 Equations DPLL-6a to calculate STATE time stamps

For calculation of time stamps use the filter delay information and use p=APS while DIR2=0:

TS_S= STATE_TS (for IDS=0, received unchanged value) (DPLL-6a0)
TS_S= TS_S - 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 - 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 - 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 16.6.2 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-6d -after DPLL-14- for this reason).

16.6.3.2 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.

16.6.3.3 Equation DPLL-6c to calculate RDT_S_ACT (nominal value)

RDT_S_ACT = 1/ DT_S_ACT (DPLL-6c)

16.6.3.4 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 a 18 bit fractional part.

Note: QDT_S_ACT uses a 6 bit integer part and a 18 bit fractional part.

16.6.3.5 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 a 18 bit fractional part.

Note: QDT_S[p-q] uses a 6 bit integer part and a 18 bit fractional part.

16.6.3.6 Equation DPLL-9 to calculate the weighted average error

for SYS=1 calculate:

MEDT_S := (EDT_S + MEDT_S)/2 (DPLL-9)

16.6.3.7 Equations DPLL-10 to calculate the current increment (nominal value)

CDT_SX _nom= (DT_S_ACT + MEDT_S)* QDT_S[p-q+1] (DPLL-10a)
with
QDT_S[p-q+1] = DT_S[p-q+1] * RDT_S[p-q] (for q>1) (DPLL-7c)
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 a 18 bit fractional part.

and the expected duration to the next *STATE* event

CDT_SX = CDT_SX _nom * SYN_S (DPLL-10b)

Note: QDT_S[p-q+1] uses a 6 bit integer part and a 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.

16.6.4 Increment prediction in Normal Mode backwards (DIR1=1)

16.6.4.1 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 a 18 bit fractional part.

16.6.4.2 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 a 18 bit fractional part.

16.6.4.3 Equation DPLL-4 to calculate the weighted average error

For SYT=1 calculate:
MEDT_T := (EDT_T + MEDT_T)/2 (DPLL-4)

16.6.4.4 Equation DPLL-5 to calculate the current increment value

CDT_TX _nom = (DT_T_ACT + MEDT_T)* QDT_T[p+q-1] (DPLL-5a1)
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 a 18 bit fractional part.

and the expected duration to the next *TRIGGER* event
CDT_TX = CDT_TX _nom * 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.

16.6.5 Increment prediction in Emergency Mode backwards (DIR2=1)

16.6.5.1 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 a 18 bit fractional part.

16.6.5.2 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 a 18 bit fractional part.

16.6.5.3 Equation DPLL-9 to calculate the weighted average error

For SYS=1 calculate:
MEDT_S := (EDT_S + MEDT_S)/2 (DPLL-9)

16.6.5.4 Equations DPLL-10 to calculate the current increment value

CDT_SX _nom= (DT_S_ACT + MEDT_S)* QDT_S[p+q-1] (DPLL-10a1)
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 a 18 bit fractional part.

and calculate the expected duration to the next *STATE* event
CDT_SX = CDT_SX _nom * 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.

16.7 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.

Action updates at highest speed

Up to 32 action values can be calculated. For the shortest increment durations (23,4 µs) not all of them can be updated with each valid 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 increment durations. 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 valid 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 an the shortest increment duration is 39 µs, there can up to 7 actions served for each valid 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.

16.7.1 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

16.7.1.1 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.

16.7.1.2 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)

16.7.1.3 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.

16.7.1.4 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].

16.7.1.5 Equation DPLL-12 to calculate the duration value until action

DTA[i] = (DT_T_ACT + MEDT_T)* PDT_T[i] (DPLL-12)

Note: All 5 steps in equations DPLL-11 to DPLL-12 are only calculated in normal mode.

16.7.2 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.

16.7.2.1 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)

16.7.2.2 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.

16.7.2.3 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)

16.7.2.4 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].

16.7.2.5 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
Note: All 5 steps in equations DPLL-11 to DPLL-12 are only calculated in normal mode or when SMC=1.

16.7.3 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.

16.7.3.1 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)

16.7.3.2 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)

16.7.3.3 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)

16.7.3.4 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)

16.7.3.5 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
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.

16.7.4 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.

16.7.4.1 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[p+q-1]) * RDT_S[t+q] (DPLL-13a3)
with

DT_Sx[p+q-1]= DT_S[p+q-1] for TS0_HRS=0 (DPLL-13b4)
or
DT_Sx[p+q-1] = DT_S[p+q-1]/8 for TS0_HRS=1 (DPLL-13b5)

16.7.4.2 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[p+q-1]) * RDT_S[t] (DPLL-13a4)
with

DT_Sx[p+q-1]= DT_S[p+q-1] for TS0_HRS=0 (DPLL-13b4)
or
DT_Sx[p+q-1] = DT_S[p+q-1]/8 for TS0_HRS=1 (DPLL-13b5)

16.7.4.3 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[p+q-1] = DT_S[t+q-1] for TS0_HRS=0 (DPLL-13b4)
or
DT_Sx[p+q-1] = DT_S[t+q-1]/8 for TS0_HRS=1 (DPLL-13b5)

16.7.4.4 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)

16.7.4.5 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
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.

16.7.5 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 durations of increments in the RAM.

16.7.5.1 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 16.7 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 16.8.6).

16.7.5.2 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

16.7.5.3 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

16.7.5.4 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 valid *TRIGGER* slope is detected and in emergency mode directly after calculation of DT_T_ACT or RDT_T_ACT respectively.

16.7.5.5 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, 16.7.3.2 or DPLL-13a4 16.7.4.2, 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 16.8.6).

16.7.5.6 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

16.7.5.7 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

16.7.5.8 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 valid *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 valid *STATE* slope is detected and in normal mode directly after calculation of DT_S_ACT or RDT_S_ACT respectively.

16.7.6 Time and position stamps for actions in Normal Mode

16.7.6.1 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 16.7.5.1.

The calculation is done after the calculation of the current expected duration value according to equation DPLL-12 at chapter 16.7.2.5. 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.

16.7.6.2 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= RCDT_TX * 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= RCDT_TX * 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 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.

16.7.6.3 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= RCDT_TX * 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= RCDT_TX * 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 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.

16.7.7 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 behaviour 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.

16.7.8 Time and position stamps for actions in Emergency Mode

16.7.8.1 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 16.7.5.5.

The calculation is done after the calculation of the current expected duration value according to equation DPLL-14 at chapter 16.7.3.5. 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.

16.7.8.2 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= RCDT_SX * 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= RCDT_SX * 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-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.

16.7.8.3 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= RCDT_SX * 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= RCDT_SX * 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.

16.8 Signal processing

16.8.1 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.

16.8.2 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.

16.8.3 Sub pulse generation for SMC=0

16.8.3.1 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)*SYN_T + MP + PD_store + 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.

16.8.3.2 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 *SYN_S+ MP + PD_S_store (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.

16.8.3.3 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 16.7.6.3.

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 16.7.6.3.

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 16.8.3.6, 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 16.8.3.6) 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.

16.8.3.4 Enabling of the compensated output for pulses

The c_{out} of the adder influences directly the *SUB_INC1* output of the DPLL (see Figure 16.8.3.6). The compensated output SUB_INCxc is in automatic end mode only enabled by EN_Cxc when **INC_CNTx** >0.

16.8.3.5 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 16.7.8.2.

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 16.7.8.2.

For RMO=1 and SMC=0:

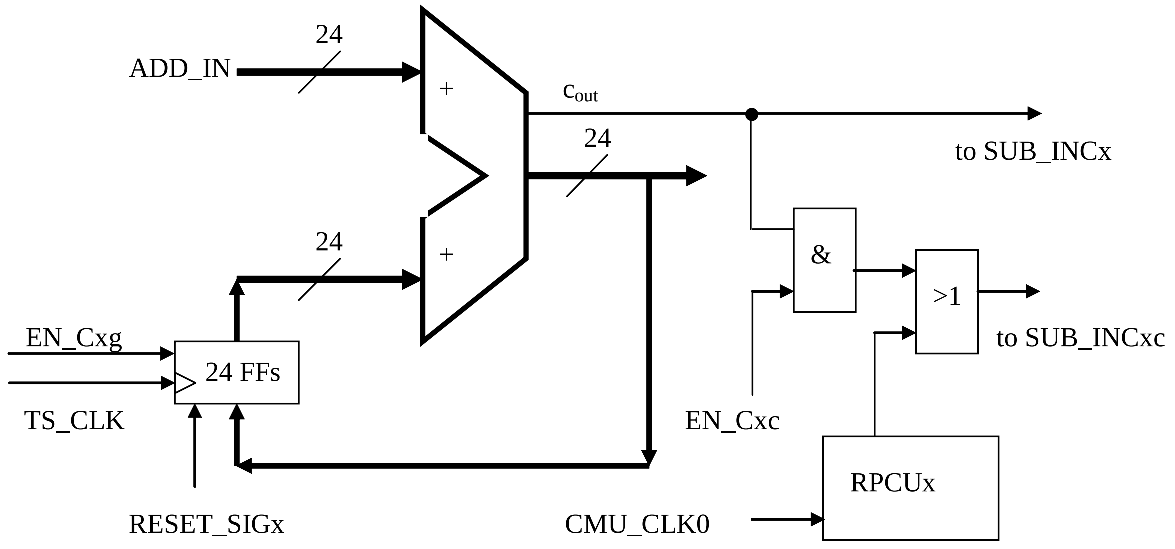
$$ADD\_IN\_CAL1 = ADD\_IN\_CALE \qquad \text{(DPLL-26b)}$$

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 16.8.3.6, 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 16.8.3.6) 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.

16.8.3.6 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 16.6.2.7, DPLL-2c1 16.6.4.4, DPLL-7c 16.6.3.7 or DPLL-7c1 16.6.4.4 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 16.8.3.3 and DPLL-26 16.8.3.5 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 lost 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.

16.8.4 Sub pulse generation for SMC=1

16.8.4.1 Necessity of two pulse generators

The Adder of picture 16.8.3.6 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.

16.8.4.2 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*SYN_T + MP + PD_store + 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

16.8.4.3 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 *SYN_S+ MP + PD_S_store + 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.

16.8.4.4 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 16.7.6.3.

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 16.7.6.3.

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 16.8.3.6). The SUB_INC1 output is in automatic end mode only enabled by EN_C1 when **INC_CNT1** >0.

16.8.4.5 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 16.7.8.2.

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 16.7.8.2.

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 16.8.3.6).
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 16.6.2.7, DPLL-2c1 16.6.4.4, DPLL-7c 16.6.3.7 or DPLL-7c1 16.6.4.4 respectively are valid only when NUTE>1 or NUSE>1 respectively. For NUTE=1 or NUSE=1 respectively the equations DPLL-30 (see chapter 16.8.4.4) and DPLL-31 (see chapter 16.8.4.5) 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 lost 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.

16.8.5 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 16.6.2.1 or DPLL-6a 16.6.3.1 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 valid 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)

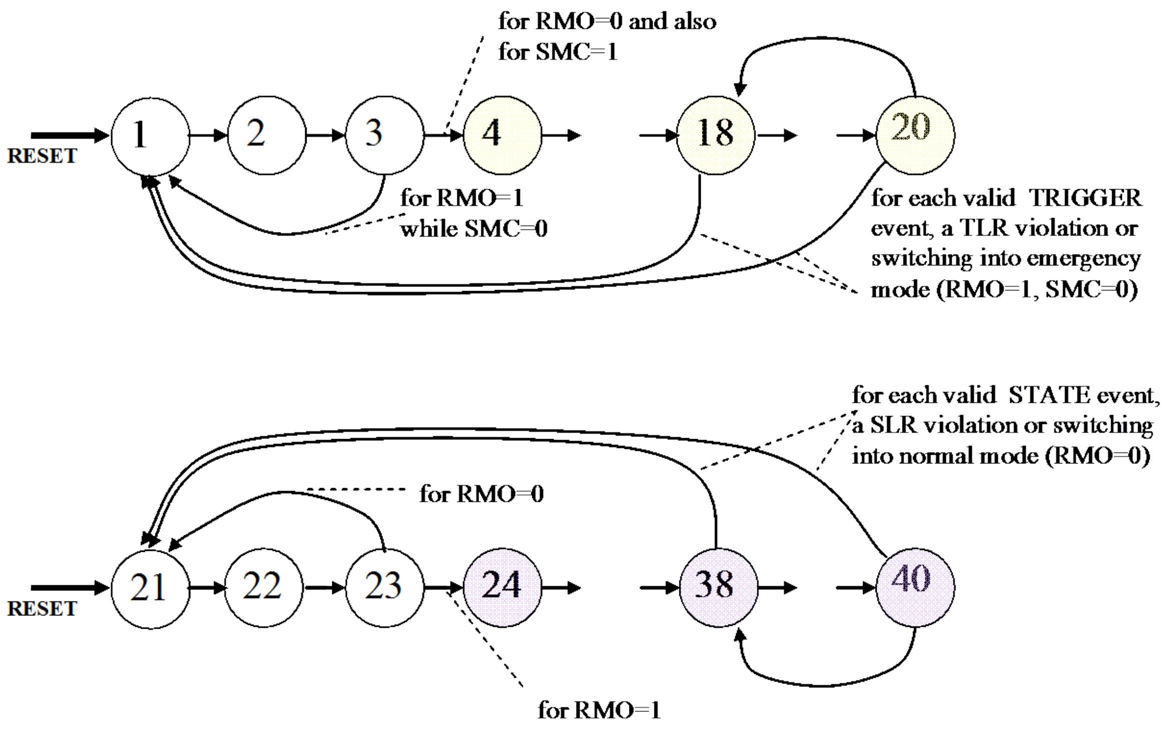
Calculate the new Position value for each STATE event:
PSSC= PSSC_old + NMB_S_TAR_OLD (DPLL-32b)
when FSD=1 and SGE1=1 (SMC=0) or SGE2=1 (SMC=1)
respectively with
PSSC_old is the last PSSC value and
NMB_S_old is the number of pulses which are calculated
and provided for sending out in the last increment.

16.8.6 Scheduling of the Calculation

After enabling the DPLL with each valid TRIGGER or *STATE* event respectively a cycle of operations is performed to calculate all the results shown in detail in the table below 16.2. A state machine controls this procedure and consists of two parts, the first is triggered by a valid 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 a valid 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 16.8.6.7 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 16.2), but steps 24 to 37 are only calculated in emergency mode (in the state machine explanation below marked cyan in 16.2) when SMC=0.

16.8.6.1 State machine partitioning for normal and emergency mode.


RESET
1
2
3
4
18
20
for RMO=0 and also for SMC=1
for RMO=1 while SMC=0
for each valid TRIGGER event, a TLR violation or switching into emergency mode (RMO=1, SMC=0)
RESET
21
22
23
24
38
40
for RMO=0
for each valid STATE event, a SLR violation or switching into normal mode (RMO=0)
for RMO=1

16.8.6.2 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 valid *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 valid *TRIGGER* event appears.
The data of equations DPLL-1b1 and DPLL-1c2 16.7.5.4 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 valid *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 16.7.5.2 or backwards 16.7.5.3. 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 valid *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 TOV_S* DT_S_ACT no valid *STATE* event appears.

The data of equations DPLL-6b1 and DPLL-6c2 16.7.5.8 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 valid *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 valid *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 16.7.5.6 or backwards 16.7.5.7. 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 adapt 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.

16.8.6.3 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 valid 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 16.2.

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 valid *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 independent of a STATE event.
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 invalid 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 valid *STATE* event as long as DIR1=0.

16.8.6.4 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 valid 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 16.8.6.3: 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 16.8.6.3.

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 a valid *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 16.8.6.3.

16.8.6.5 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 valid 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 16.8.6.3: 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 16.8.6.3.

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 a valid *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.

16.8.6.6 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 valid *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 valid *TRIGGER* slope accordingly.

When the *TRIGGER* locking range TLR is violated[8) ,
the state machine 1 will remain in state 1 and the address pointer APT, APT_2b and APT_2c will remain unchanged until the CPU sets the APT_2c accordingly. In this case also the NUTE value in the NUTC register is set to 1. The DPLL stops the generation of the SUB_INC1 pulses and will perform no other actions - remaining in step1 of the first state machine (see 16.2).
[8) 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 16.11.76). 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 valid *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 valid *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 16.2).
[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 16.11.77).
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.

16.8.6.7 State description of the State Machine.

Step	Description	Comments

always for DEN=1	**for each inactive TRIGGER slope:**
	generate the TISI interrupt; calculate the time stamp difference
	event, store this value at THVAL;
	when THMI >0 is violated (ΔT
	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;
	correct INC_CNT1 by addition of 2*WP before sending the correction pulses;
	check THMA, when THMA is violated, generate the TAXI interrupt; go to step 1
	for each inactive *STATE* slope:
	set DIR2=DIR1
		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

always for DEN=1	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 16.8.6.3 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	**for SMC=1**; set the direction bits always after incr./decr. the corresponding address pointers;

	frequency.	
1	When DEN = 0 or TEN=0: stay in step 1 until DEN=1, TEN=1 and at least one valid *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 (valid slope); **When no valid TRIGGER appears** and when TS_T_CHECK time is reached: • 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 *TRIGGER* appears. **When a valid TRIGGER appears check PVT** **- when the PVT value is violated:** generate the PWI interrupt, ignore the *TRIGGER* input and wait for the next valid TRIGGER slope (ignore each invalid 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 16.7.5) **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	Depending on TSL, TEN, DEN the leaving of step one is done with the next *TRIGGER* input; **Note:** Step 1 is also left in emergency mode when a valid *TRIGGER* event appears in order to make a switch back to normal mode possible; _old - values are values valid at the last but one valid *TRIGGER* event ; 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 valid TRIGGER event *****) PD_store = 0 for AMT=0 (see

	• 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** for SYT=0 and FTD =1: • **dir_crement** APT and APT_2b by one; • **dir_crement** for **SGE1_delay**$^{****)}$**=1 and TOR=0** PSTC by NMB_T_TAR$^{**)}$ • for (RMO=0 or SMC=1) **and SGE1=1, TOR=0**: increment INC_CNT1 by (MLT+1)$^{*)}$ +MPVAL1$^{***)}$ for SYT=1 and TOR=0 : • **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;	DPLL_CTRL_0 register)
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 behaviours of RM and HW-IP: For the HW-IP the values of SYN_T and PD_store are not updated until a new valid 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 a valid TRIGGER slope)
	
	calculate target pulses:
	NMB_T_TAR = (MLT+1)$^{*)}$*SYN_T + PD_store + 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 behaviours of RM and HW-IP: For the HW-IP the values of SYN_T and PD_store are not updated until a new valid 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
		for RMO=0 or SMC=1
		
		for DLM=0
		
		for DLM=1

	step 10);
	for DMO=DLM=0 and EN_C1u=0:
	reset the FlipFlops in the SUB_INC1 generator;
	start sending SUB_INC1;	

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 valid *TRIGGER* appears **for all following steps 18 to 20:**	for all conditions

	go immediately back to step 1, when a valid 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	
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 a valid *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 a valid *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 a valid *TRIGGER* event occurs
21	When DEN = 0 or SEN=0: stay in step 1 until DEN=1, SEN=1 and at least one valid *STATE* has been detected (FSD=1); the following steps are performed alsways (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 (valid slope); for each invalid 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 valid *STATE* appears. **When a valid *STATE* appears:** store the actual position stamp at PSSM (value at the STATE event) update RAM by equation DPLL-6a-c	Depending on SSL, SEN, DEN the leaving of step 21 one is done 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.)

	(see chapter 16.7.5); **store the actual INC_CNT1/2 at MP1/MP2 respectively as missing pulses** *(instead of calculations in step 25)* 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 and SOR=0: • **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_1c3 for SYS=1 and SOR=0: • **dir_crement** APS and APS_1c3 • **dir_crement** APS_1c2 by SYN_S_old	**) 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 valid STATE event ****) **SGE2_delay** is the value of SGE2 delayed by one valid STATE event *****) PD_S_store = 0 for AMS=0 (see DPLL_CTRL_0 register)

	• for RMO=1 and SMC=0: for **SGE1_delay**$^{***)}$**=1** **dir_crement** PSSC by NMB_S_TAR$^{*)}$; for **SGE1=1** increment 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 behaviours of RM and HW-IP: For the HW-IP the values of SYN_S and PD_S_store are not updated until a new valid 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(valid STATE slope)
	- for TBU_CH2:
	MP2 = INC_CNT2(valid STATE slope)
	calculate target number of pulses:
	NMB_S_TAR = MLS1*SYN_S + PD_S_store +MPVAL1 (for SMC=0)
	NMB_S_TAR = MLS2*SYN_S + PD_S_store + MPVAL2 (for SMC=1)
		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;

	(instead of PD_S_store use zero in the case AMS=0)	add MPVAL1/2 once to INC_CNT1/2 and reset PCM1/2 after that
26	sent MPx with highest possible frequency and set NMB_S = NMB_S_TAR	only for RMO=1, DMO=0 and COA=0
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 behaviours of RM and HW-IP: For the HW-IP the values of SYN_S and PD_S_store are not updated until a new valid 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 FlipFlops 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 * (TOV_S);
		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
		for SMC=0: 24 actions,
		for SMC=1: 12 actions;
		depending on ACT_N[i] in **DPLL_ACT_STA** register

	b = remainder of the division (fractional part) for backward direction use NA[i] = (PSSC - PSA[i])/(MLS1) 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 TOV_S: set the ISN status bit and reset the corresponding LOCK bit, when not plausible; go to step 38, when no valid *STATE* appears **for all following steps 38 to 40:** **go immediately back to step 21, when a valid *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

		a valid *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 a valid *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 a valid *STATE* event occurs

16.9 DPLL Interrupt signals

The DPLL provides 27 interrupt lines. These interrupts are shown below.

16.9.1 DPLL Interrupt signals

interrupt signals of table

Signal	Description
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	Lost 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	Lost 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 16.12
2) see DPLL STATUS register; see 16.11.30
3) see TINT value in the corresponding ADT_T[i] section of RAM region 2; see 16.12.3

16.10DPLL Register overview

DPLL register overview
The registers available and the size of RAM region 2 depends on the device chosen. There are 4 devices considered: device 1…4.

Details in Section	Name	Description	Init value
16.11.1	DPLL_CTRL_0	Control Register 0	0x003B_BA57
16.11.2	DPLL_CTRL_1	Control Register 1	0xB000_0000
16.11.3	DPLL_CTRL_2	Control Register 2 (actions 0-7 enable)	0x0000_0000
16.11.4	DPLL_CTRL_3	Control Register 3 (actions 8-15 enable)	0x0000_0000
16.11.5	DPLL_CTRL_4	Control Register 4 (actions 16-23 enable)	0x0000_0000
16.11.6	DPLL_CTRL_5[2)]	Control Register 5 (actions 24-31 enable)	0x0000_0000

16.11.7	DPLL_ACT_STA	ACTION Status Register with connected shadow register	0x0000_0000
16.11.8	DPLL_OSW	Offset and switch old/new address register	0x0000_0200
16.11.9	DPLL_AOSV_2	Address offset register for APT in RAM region 2	0x1810_0800
16.11.10	DPLL_APT	Actual RAM pointer to RAM regions 2a, b and d	0x0000_0000
16.11.11	DPLL_APS	Actual RAM pointer to regions 1c1, 1c2 and 1c4	0x0000_0000
16.11.12	DPLL_APT_2C	Actual RAM pointer to RAM region 2c	0x0000_0000
16.11.13	DPLL_APS_1C3	Actual RAM pointer to RAM region 1c3	0x0000_0000
16.11.14	DPLL_NUTC	Number of recent *TRIGGER* events used for calculations (mod 2*(TNU +1-SYN_NT))	0x0001_2001
16.11.15	DPLL_NUSC	Number of recent *STATE* events used for calculations (e.g. mod 2*(SNU +1-SYN_NS) for SYSF=0)	0x0000_2081
16.11.16	DPLL_NTI_CNT	Number of active *TRIGGER* events to interrupt	0x0000_0000
16.11.17	DPLL_IRQ_NOTIFY	Interrupt notification register	0x0000_0000
16.11.18	DPLL_IRQ_EN	Interrupt enable register	0x0000_0000
16.11.19	DPLL_IRQ_FORCINT	Interrupt force register	0x0000_0000
16.11.20	DPLL_IRQ_MODE	Interrupt mode register	0x0000_0000

16.11.21	DPLL_EIRQ_EN	Error interrupt enable register	0x0000_0000
16.11.22	INC_CNT1	Counter for pulses for TBU_CH1_BASE to be sent in automatic end mode	0x0000_0000
16.11.23	INC_CNT2	Counter for pulses for TBU_CH2_BASE to be sent in automatic end mode when SMC=RMO=1	0x0000_0000
16.11.24	DPLL_APT_SYNC	old RAM pointer and offset value for *TRIGGER*	0x0000_0000
16.11.25	DPLL_APS_SYNC	old RAM pointer and offset value for *STATE*	0x0000_0000
16.11.26	TBU_TS0_T	TBU_CH0_BASE value at last *TRIGGER* event	0x0000_0000
16.11.27	TBU_TS0_S	TBU_CH0_BASE value at last *STATE* event	0x0000_0000
16.11.28	ADD_IN_LD1	direct load input value for SUB_INC1	0x0000_0000
16.11.29	ADD_IN_LD2	direct load input value for SUB_INC2	0x0000_0000
16.11.30	DPLL_STATUS	Status Register	0x0000_0000
16.11.31	DPLL_ID_PMTR_0-31$^{3)}$	9 bit ID information for input signals PMT_0-31 (8:0)	0x0000_01FE
16.11.32	DPLL_CTRL_0_SHADOW_TRIGGER	shadow register of DPLL_CTRL_0	0x0000_0257
16.11.33	DPLL_CTRL_0_SHADOW_STATE	shadow register of DPLL_CTRL_0	0x0000_0000
16.11.34	DPLL_CTRL_1_SHADOW_TRIGGER	shadow register of DPLL_CTRL_1	0x0000_0000
16.11.35	DPLL_CTRL_1_SHADOW_STATE	shadow register	0x0000_0000

		of DPLL_CTRL_1	
16.11.36	DPLL_RAM_INI	initialization control and status for RAMs	0x0000_0000

[2]**Note:** This register is only available for device 4.
[3]**Note:** The registers DPLL_ID_PMTR 24-31 are only available for device 4.
RAM Region 1 map description.

Name	Description	Address offset in relation to DPLL start address
	RAM Region 1a 384 bytes for 128 words of 24 Bits; this RAM is only accessible for DEN=0	0x0200- 0x03FC
PSA0 - PSA31[1)]	ACTION_i Position/Value action request Register (i = 0...31)3)	0x0200 - 0x027C (device 4) 0x0200 - 0x025C (device 1-3) See 16.11.37
DLA0 - DLA31[1)]	ACTION_i time to react before PSAi (i = 0...31)	0x0280 - 0x02FC (device 4) 0x0260 - 0x02BC (device 1-3) See 16.11.38
NA0 - NA31[1)]	# of *TRIGGER/STATE* increments to ACTION_i (i = 0...31)	0x0300 - 0x037C (device 4) 0x02C0 - 0x031C (device 1-3) See 16.11.39
DTA0 - DTA31[1)]	calculated relative time to ACTION_i (i = 0...31)	0x0380 - 0x03FC (device 4) 0x0320 - 0x037C (device 1-3) See 16.11.40
	RAM Region 1b	0x0400- 0x05FC
	Note: the following registers for variables are located in RAM Region 1b, read access by AEI via bus interface possible, writing results in an interrupt; data width of 3 Bytes used for 24 bit	288 bytes for 96 words of 24 Bits

	values	
	***TRIGGER* signal information stored**	
TS_T	Actual signal *TRIGGER* time stamp register TRIGGER_TS	0x0400/ 0x0404 See 16.11.41 and 16.11.42
TS_T_OLD	Previous signal *TRIGGER* time stamp register TRIGGER_TS_old	0x0404/ 0x0400 See 16.11.41 and 16.11.42
	Note: the switch of the LSB address bits is performed using the SWON register at 0x0020	0x0400... 0x0404 See 16.11.41 and 16.11.42
FTV_T	Actual signal *TRIGGER* filter value	0x0408 See 16.11.43
	do not use	0x040C
	***STATE* signal information stored**	
TS_S	Actual signal *STATE* time stamp register STATE_TS	0x0410/ 0x0414 See 16.11.44 and 16.11.45
TS_S_OLD	Previous signal *STATE* time stamp register STATE_TS_old	0x0414/ 0x0410 See 16.11.44 and 16.11.45
	Note: The switch of the LSB address bits is performed using the SWON register at 0x0020.	0x0410... 0x0414 See 16.11.44 and 16.11.45
FTV_S	Actual signal *STATE* filter value	0x0418 See 16.11.46
	do not use	0x041C
THMI	*TRIGGER* hold time min value	0x0420 See 16.11.47
THMA	*TRIGGER* hold time max value	0x0424 See 16.11.48
THVAL	measured last pulse time from valid to invalid *TRIGGER* slope	0x0428 See 16.11.49
	do not use	0x042C
TOV	Time out value of *TRIGGER*, according to the last nominal increment for a missing TRIGGER	0x0430 See 16.11.50
TOV_S	Time out value of *STATE*, according to the last nominal increment for a missing STATE	0x0434 See 16.11.51

ADD_IN_CAL1	calculated ADD_IN value for SUB_INC1 generation	0x0438 See 16.11.52
ADD_IN_CAL2	calculated ADD_IN value for SUB_INC2 generation	0x043C See 16.11.53
MPVAL1	missing pulses to be added/subtracted directly to SUB_INC1 and INC_CNT1 once	0x0440 See 16.11.54
MPVAL2	missing pulses to be added/subtracted directly to SUB_INC2 and INC_CNT2 once	0x0444 See 16.11.55
NMB_T_TAR	target number of TRIGGER pulses	0x0448 See 16.11.56
NMB_T_TAR_OLD	target number of TRIGGER pulses	0x044C See 16.11.57
NMB_S_TAR	target number of STATE pulses	0x0450 See 16.11.58
NMB_S_TAR_OLD	target number of STATE pulses	0x0454 See 16.11.59
	do not use	0x0458... 0x045C
RCDT_TX	reciprocal value of expected increment duration (T)	0x0460 See 16.11.60
RCDT_SX	reciprocal value of expected increment duration (S)	0x0464 See 16.11.61
RCDT_TX_NOM	reciprocal value of the expected nominal increment duration (T)	0x0468 See 16.11.62
RCDT_SX_NOM	reciprocal value of the expected nominal increment duration (S)	0x046C See 16.11.63
RDT_T_ACT	actual reciprocal value of *TRIGGER*	0x0470 See 16.11.64
RDT_S_ACT	actual reciprocal value of *STATE*	0x0474 See 16.11.65
DT_T_ACT	Duration of last *TRIGGER* increment	0x0478 See 16.11.66
DT_S_ACT	Duration of last *STATE* increment	0x047C See 16.11.67
	Calculated immediate values (eq. DPLL-1 to DPLL-10)	
EDT_T	Absolute error of prediction for last *TRIGGER* increment	0x0480 See 16.11.68
MEDT_T	Average absolute error of prediction up to the last *TRIGGER* increment	0x0484 See 16.11.69
EDT_S	absolute error of prediction for last *STATE* increment	0x0488 See 16.11.70
MEDT_S	Average absolute error of prediction up to the last *STATE* increment	0x048C See 16.11.71

CDT_TX	Expected duration of current *TRIGGER* increment	0x0490 See 16.11.72
CDT_SX	Expected duration of current *STATE* increment	0x0494 See 16.11.73
CDT_TX_NOM	Expected nominal duration of current *TRIGGER* increment (without consideration of missing events)	0x0498 See 16.11.74
CDT_SX_NOM	Expected nominal duration of current *STATE* increment (without consideration of missing events)	0x049C See 16.11.75
TLR	*TRIGGER* locking range value; the TOR bit in the DPLL_STATUS register is set when violated	0x04A0 See 16.11.76
SLR	*STATE* locking range value; the SOR bit is set when viollated	0x04A4 See 16.11.77
	Relations of the sum of prediction increments to the reference increment in the past (see equations DPLL-11 or DPLL-13 for calculation)	
PDT_[i]$^{2)}$	predicted time to ACTION_[i]$^{2)}$	0x0500... 0x057C See 16.11.78
	do not use	0x0580 - 0x05BC
MLS1	Calculated number of sub-pulses between two *STATE* events (to be set by CPU)	0x05C0 See 16.11.79
MLS2	Calculated number of sub-pulses between two *STATE* events (to be set by CPU) for the use when SMC=RMO=1	0x05C4 See 16.11.80
CNT_NUM_1	number of sub-pulses of SUB_INC1 in continuous mode, updated by the host only	0x05C8 See 16.11.81
CNT_NUM_2	number of sub-pulses of SUB_INC2 in continuous mode, updated by the host only	0x05CC See 16.11.82
PVT	Plausibility value of next active *TRIGGER* slope	0x05D0 See 16.11.83
	do not use	0x05D4... 0x05DC
PSTC	Accurate calculated position stamp of last *TRIGGER* input;	0x05E0 See 16.11.84
PSSC	Accurate calculated position stamp of last *STATE* input;	0x05E4 See 16.11.85
PSTM	Measured position stamp at last valid*TRIGGER* input	0x05E8/

		0x05EC See 16.11.86 and 16.11.87
PSTM_OLD	Measured position stamp at last but one valid *TRIGGER* input	0x05EC/ 0x05E8 See 16.11.86 and 16.11.87
PSSM	Measured position stamp at last valid *STATE* input	0x05F0/ 0x05F4 See 16.11.88 and 16.11.89
PSSM_OLD	Measured position stamp at last but one valid *STATE* input	0x05F4/ 0x05F0 See 16.11.88 and 16.11.89
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)	0x05F8 See 16.11.90
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)	0x05FC See 16.11.91
	RAM Region 1c	0x0600 – 0x09FC
	Note: the following registers for the signal *STATE* are located in RAM Region 1c, read access by AEI via bus interface possible, writing results in an interrupt; data width of 3 Bytes used for 24 bit values	0,75 Kbytes for 256 words of 24 Bits
1c1	**Reciprocal values of the corresponding successive increments RDT_S[i] (see equations DPLL-6a,b); the values are calculated using the recent NUSE increments (see NUSC register at address 0x0038)**	
RDT_S0	RDT_S0	0x0600 See 16.11.92
RDT_S1	RDT_S1	0x0604 See 16.11.92

...	up to **2*(SNU+1)-SYN_NS valid entries**	...
RDT_S63	RDT_S63	0x06FC See 16.11.92
1c2	**Time stamp field for *STATE* events**	
TSF_S0	TSF_S0	0x0700 See 16.11.93
TSF_S1	TSF_S1	0x0704 See 16.11.93
...	up to **2*(SNU+1) valid entries**	...
TSF_S63	TSF_S63	0x07FC See 16.11.93
1c3	**Adapt values for the current *STATE* increment; time stamp values bits 24-27 in addition to the corresponding 24 bit value of TSF_Sx above, stored in bits 23:20 of ADT_S[i];**	
ADT_S0	ADT_S0	0x0800 See 16.11.94
ADT_S1	ADT_S1	0x0808 See 16.11.94
...	up to **2*(SNU+1)-SYN_NS valid entries**	...
ADT_S63	ADT_S63	0x08FC See 16.11.94
1c4	**Uncorrected last increment value of *STATE* (DT_S) for FULL_SCALE; measuring data of increments without corrections used for the CPU to generate ADT_S values**	
DT_S0	DT_S0	0x0900 See 16.11.95
DT_S1	DT_S1	0x0904 See 16.11.95
...	up to **2*(SNU+1)-SYN_NS valid entries**	...
DT_S63	DT_S63	0x09FC See 16.11.95

[1]**Note:** The values PSA24-31, DLA24-31, NA24-31 and DTA24-31 in RAM 1a are only available for device 4.
[2]**Note:** The values PDT_24 to PDT_31 in RAM1b are only available for device 4.
Register Region EXT map description

Address offset in relation to DPLL start address	Name	Description	Init value
0x0E00 - 0x0F1C	**Register Region EXT**	**extension of register region in order to allow up to 32 action calculations (device 4)**	
0x0E00 - 0x0E7C See 16.11.96	DPLL_TSAC0-TSAC31[3)]	calculated action time stamps for actions 0...31	0x007F_FFFF
0x0E80 - 0x0EFC See 16.11.97	DPLL_PSAC0-PSAC31[3)]	calculated action position stamps for actions 0...31	0x007F_FFFF
0x0F00 See 16.11.98	DPLL_ACB_0	control bits for actions 0...3	0x0000_0000
0x0F04 See 16.11.98	DPLL_ACB_1	control bits for actions 4...7	0x0000_0000
0x0F08 See 16.11.98	DPLL_ACB_2	control bits for actions 8...11	0x0000_0000
0x0F0C See 16.11.98	DPLL_ACB_3	control bits for actions 12...15	0x0000_0000
0x0F10 See 16.11.98	DPLL_ACB_4	control bits for actions 16...19	0x0000_0000
0x0F14 See 16.11.98	DPLL_ACB_5	control bits for actions 20...23	0x0000_0000
0x0F18 See 16.11.98	DPLL_ACB_6[3)]	control bits for actions 24...27	0x0000_0000
0x0F1C See 16.11.98	DPLL_ACB_7[3)]	control bits for actions 28...31	0x0000_0000

[3)]**Note:** The registers DPLL_TSAC24-31, DPLL_PSAC24-31 and DPLL_ACB_6-7 are only available for device 4.

RAM Region 2 map description.

Name	Description	Address offset in relation to DPLL start address
	RAM Region 2	0x4000 – 0x4800… 0x7FFC depending on device chosen See 16.12
	NOTE: the following registers for the signal *TRIGGER* are located in RAM region 2, read access by AEI via bus interface possible, write access results in an interrupt, when enabled; data width of 3 bytes used for 24 bit values The RAM field part contains up to 1024 values and all are configurable for 128, 256, 512 or 1024 values in order to select the RAM size needed. The maximum available RAM size depends on the device as shown below: device 1: 512 words á 24 bits device 2: 1024 words á 24 bits device 3: 2048 words á 24 bits device 4: 4096 words á 24 bits	size from 1,5 Kbytes to 12 Kbytes configurable for word sizes of 24 Bits
Region 2a	**Reciprocal values of the corresponding successive increments RDT_T[i] (see equations DPLL-2a,b);** **address offsets are given by the AOSV_2a**	AOSV_2a values are addresses after shift left by 8
RDT_T0 See 16.12.1	RDT_T0	AOSV_2a
RDT_T1 See 16.12.1	RDT_T1	+4
…	… **2*(TNU+1-SYN_NT) valid entries**	…
RDT_T1023 See 16.12.1	RDT_T1023	+4092
Region 2b	**Time stamp field for all *TRIGGER***	

	events in FULL_SCALE; 24 bit time stamp values	
TSF_T0 See 16.12.2	TSF_T0	AOSV_2b
TSF_T1 See 16.12.2	TSF_T1	+4
…	**… 2*(TNU+1) valid entries**	…
TSF_T1023 See 16.12.2	TSF_T1023	+4092
Region 2c	**ADT_T values to correct the measured *TRIGGER* signal values; time stamp values bits 24-27 in addition to the 24 bit value of TSF_Tx, stored in the bits 23:20 of ADT_T[i]**	
ADT_T0 See 16.12.3	ADT_T0	AOSV_2c
ADT_T1 See 16.12.3	ADT_T1	+4
…	**… 2*(TNU+1-SYN_NT) valid entries**	…
ADT_T1023 See 16.12.3	ADT_T1023	+4092
Region 2d	**Uncorrected last increment values of *TRIGGER* (DT_T); measuring raw data of increments**	
DT_T0 See 16.12.4	DT_T0	AOSV_2d
DT_T1 See 16.12.4	DT_T1	+4
…	**… 2*(TNU+1-SYN_NT) valid entries**	…
DT_T1023 See 16.12.4	DT_T1023	+4092

16.11 DPLL Register and Memory description

Description of Registers beginning from Register DPLL_CTRL_0.
Description of RAM Regions beginning from 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.

Description of memory region RAM 1a beginning from Memory PSA[i]:
The RAM Region 1a is writeable only for DEN=0 (see DPLL_CTRL_1 register).

Description of memory region RAM1b beginning from Memory TS_T.
Description of memory region RAM1c beginning from Memory RDT_S.
Description of register region EXT beginning from Register DPLL_TSAC[i]:
This is an extension of the normal register region above in order to allow up to 32 action calculations (device 4).

Description of memory region RAM 2 beginning from Memory RDT_T.

16.11.1 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);
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 binary value plus 1. This value can only be written when the DPLL is disabled.

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 binary value plus 1. This value can only be written when the DPLL is disabled.

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 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 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 the case of no further TRIGGER or STATE events appearing the CPU has to perform the corrections above accordingly.

[1)] stored in an independent shadow register for a valid *TRIGGER* event and for DEN = 1.

[2)] stored in an independent shadow register for a valid *STATE* event and for DEN = 1.

[3)] the time between two valid *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 valid *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 then 65536. This is fulfilled when x is greater then 256.

16.11.2 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 a increment is reached, no further pulse is generated until the next valid *TRIGGER*/*STATE* is received; in the case of getting a new valid *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 can not 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 valid and invalid 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 invalid 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 16.8.3.6).

1 = missing pulses of the last increment are distributed evenly to the next increment, calculations are done when the next valid input event appears. The number of missing sub-pulses will be determined by the pulse counter difference between the last two valid *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 valid 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 adapt 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.

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 adapt 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 the DPLL is disabled.

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

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 the DPLL is disabled.

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.

00: No slope of *STATE* will be used; this value makes only sense in normal mode
01: Low high slope will be used as active slope, only inputs with a signal value of “1” will be considered
10: High low slope will be used as active slope, only inputs with a signal value of “0” will be considered
11: Both slopes will be used as active slopes

Note: This bit can only be written when the DPLL is disabled.

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.

00: No slope of *TRIGGER* will be used; this value makes only sense in emergency mode

01: Low high slope will be used as active slope, only inputs with a signal value of “1” will be considered

10: High low slope will be used as active slope, only inputs with a signal value of “0” will be considered

11: Both slopes will be used as active slopes

Note: This bit can only be written when the DPLL is disabled.

[1] stored in an independent shadow register for a valid *TRIGGER* event and for DEN = 1.

[2] stored in an independent shadow register for a valid *STATE* event and for DEN = 1.

[3] Bit is cleared, when transmitted to shadow register

16.11.3 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

[1] **Note**: This bit can only be written when the correspondent WADi Bit is set. 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).

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 **Note:** For writing WADi =1 only the corresponding the AENx bits are written. The AENi bits remain unchanged when the corresponding WADi=0.
Bit 31:24	**Reserved** Note: Read as zero, should be written as zero.

16.11.4 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

[1] **Note**: This bit can only be written when the correspondent WADi Bit is set. 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).

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

Note: For writing WADx =1 only the corresponding the AENx bits are written. The AENx bits remain unchanged when the corresponding WADx=0.

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

16.11.5 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

[1] **Note**: This bit can only be written when the correspondent WADi Bit is set. 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).

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

Note: For writing WADx =1 only the corresponding the AENx bits are written. The AENx bits remain unchanged when the corresponding WADx=0.

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

16.11.6 Register DPLL_CTRL_5 [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								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

[1] **Note**: This bit can only be written when the correspondent WADi Bit is set. 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).

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

Note: For writing WADx =1 only the corresponding the AENx bits are written. The AENx bits remain unchanged when the corresponding WADx=0.

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

[2]**Note:** This register is only available for device 4.

16.11.7 Register DPLL_ACT_STA

Address Offset:	**see Appendix B**	**Initial Value:**	**0x0000_0000**
	31 … 16	15 … 0	
Bit	ACT_N		
Mode	RPw		
Initial Value	0x0000 00		

Action Status Register including Shadow Register

Bit 31:0 **ACT_N[i],(i=0...31):** 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 = new PMTR data received or calculation is to be precised 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: for devices 1-3: only the bits 0 to 23 are implemented.

Note: These bits can only be written for test purposes when the DPLL is disabled.

16.11.8 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	0x000000																						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 512of 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.

16.11.9 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.

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.

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.

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.

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.

16.11.10 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 fro 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 valid *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 a valid *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 valid *TRIGGER* event (simultaneously with APT and APT_2c) for DIR1=0 when a valid *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 valid *TRIGGER* event the APT_2b_ext value is added/subtracted (while APT_2b_status is one; see DPLL_APT_SYNC register at chapter 16.11.24).

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.

16.11.11 Register DPLL_APS

Address Offset:	see Appendix B						Initial Value: 0x0000_0000
	31 … 20	19 … 14	13	12 … 8	7 … 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 valid *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 a valid *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 valid *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 valid *STATE* event the APS_1c2_ext value is added/subtracted (while APS_1c2_status is one; see DPLL_APT_SYNC register at chapter 16.11.25).

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.

16.11.12 Register DPLL_APT_2C

Address Offset:	**see Appendix B**	**Initial Value:** **0x0000_0000**	
	31 … 12	11 … 2	1 0
Bit	Reserved	APT_2C	Reserved
Mode	R	RW	R
Initial Value	0x0000 0	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 adapt 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 a valid *TRIGGER* event (by evaluation of the invalid 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.

16.11.13 Register DPLL_APS_1C3

Address Offset:	**see Appendix B**	**Initial Value:**	**0x0000_0000**
	31 … 8	7 … 2	1 … 0
Bit	Reserved	APS_1C3	Reserved
Mode	R	RW	R
Initial Value	0x0000 00	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 a valid *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.

16.11.14 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 register is set by the CPU, but reset automatically to “1” by a change of direction or lost 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: 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

0= the NUTE value is less then FULL_SCALE

1= the NUTE value is equal 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 16.8.6.7.

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

16.11.15 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 lost 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 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

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 16.8.6.7.

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

16.11.16 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 valid *TRIGGER* event.

Bit 31:10 **Reserved**
Note: Read as zero, should be written as zero.

16.11.17 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
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

Bit 27 **DCGI:** Direction change interrupt
0 = No direction change of *TRIGGER* is detected
1 = Direction change of *TRIGGER* is detected

Note: The interrupt occurs at line number 0.

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.

16.11.18 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:** 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:** 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:** 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.

16.11.19 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.

16.11.20 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

00 = Level mode

01 = Pulse mode

10 = Pulse-Notify mode

11 = 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

16.11.21 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:** 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:** 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:** 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.

16.11.22 Register DPLL_INC_CNT1

Address Offset:	**see Appendix B**	**Initial Value:**	**0x0000_0000**
	31 … 24	23 … 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 valid input signal in automatic end mode;

Automatic addition of the number of demanded pulses MLT/MLS1 when getting a valid *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 are 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.

16.11.23 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	0x0000 00

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 valid input signal in automatic end mode;

Automatic addition of the number of demanded pulses MLS2 when getting a valid *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 are 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.

16.11.24 Register DPLL_APT_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	APT_2B_OLD	Reserved	APT_2B_STATUS	APT_2B_EXT
Mode	R	RW	R	RW	RW
Initial Value	0x00	0x000	0	0	0x000

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 valid 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.

16.11.25 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_STATUS	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 valid 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.

16.11.26 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 valid 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.

16.11.27 Register DPLL_TBU_TS0_S

Address Offset:	**see Appendix B**	**Initial Value:** **0x0000_0000**
	31 … 24	23 … 0
Bit	Reserved	TBU_TS0_S
Mode	R	RPw
Initial Value	0x00	0x000000

Time Stamp Value for the last valid 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.

16.11.28 Register DPLL_ADD_IN_LD1

Address Offset:	**see Appendix B**	**Initial Value:** **0x0000_0000**
	31 … 24	23 … 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.

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 valid *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.

Bit 31:24 **Reserved**

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

16.11.29 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								0x000000																							

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.

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 valid *STATE* event 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.

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

16.11.30 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	Reserved
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	R
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

Status Register

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

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_actual or RDT_S_actual 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 16.11.1). 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_HRT value is set to 1 while LOW_RES=0

Bit 6 **RCT: Resolution conflict TRIGGER**.

0 =	No resolution conflict detected
1 =	the TS0_HRS 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; address pointers APS, APS_1c2 and APS_1c3 are frozen and the generation of pulses SUB_INC1,2 respectively is stopped

Bit 9 **MS: Missing *STATE*** detected according to TOV_S.

0 =	No missing *STATE* detected or a new valid *STATE* slope occurred
1 =	At least one missing *STATE* detected after the last valid 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; address pointers APT, APT_2b and APT_2c are frozen and the generation of pulses SUB_INC1 is stopped

Bit 11 **MT: Missing *TRIGGER*** detected according to TOV

0 =	No missing *TRIGGER* detected or a new valid *TRIGGER* slope occurred
1 =	At least one missing *TRIGGER* detected after the last valid 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 actions** 12 to 23 **in progress** (**2**nd part) 0 = currently no action calculation, new data requests possible 1 = action calculation in progress, no new data requests possible
Bit 19	**CAIP1: Calculation of actions** 0 to 11 **in progress** (**1**st part) 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 a valid 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 valid *STATE* event was detected after enabling DPLL 1 = At least one valid *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 valid *TRIGGER* event was detected after enabling DPLL 1 = At least one valid *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 a valid 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 a valid 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 a valid 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 a valid 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 16.11.77) and is reset, when at the current or last valid 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 16.11.76) and is reset, when at the current or last valid 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).

16.11.31 Register DPLL_ID_PMTR_[x]

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	0x0000 00																							0x01FE								

ID Information for Input Signal PMTR_x (Position minus Time Request), x=0...31[1)]

Bit 8:0	**ID_PMTR_x:** ID information to the input signal PMTR_x from the ARU. **Note:** This value can only be written when the DPLL is disabled.
Bit 31:9	**Reserved** Note: Read as zero, should be written as zero.

[1)] **Note:** The registers DPLL_ID_PMTR_24-31 are only available for device 4.

16.11.32 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 a valid 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 a valid 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.

16.11.33 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 a valid 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 a valid 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.

16.11.34 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 a valid 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 valid 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 a valid 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.

16.11.35 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	SYN_NS																					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 a valid 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	**SYN_NS: Synchronization number of *STATE*;** summarized number of virtual increments in HALF_SCALE

Note: Only the values characterized by [2)] are stored for a valid 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.

16.11.36 Register DPLL_RAM_INI

Address Offset:	**see Appendix B**	**Initial Value:**	**0x0000_0000**				
	31 … 5			4	3	2	1	0
Bit	Reserved			INIT_RAM	Reserved	INIT_2	INIT_1B	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_1b:** RAM region 1b 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 behaviour.

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

16.11.37 Memory DPLL_PSA[i]

Address Offset:	**see Appendix B**	**Initial Value:**	**0x0000_0000**
	31 … 24	23 … 0	
Bit	Reserved	PSA	
Mode	R	RPw	
Initial Value	0x00	0x000000	

Position Request for Action i, (RAM1a, i=0...31)$^{1)}$
Bit 23:0 **PSA** Position information of a desired action (i=0...31)$^{1)}$.
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 only available for device 4.

16.11.38 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...31)[1)]

Bit 23:0	**DLA** Time to react before the corresponding position value of a desired action is reached (x=0...31)[1)]. In the case of LOW_RES=1 (see chapter 16.4.2) 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 device 4.

16.11.39 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	Reserved	DW	DB
Mode	R	RPw	RPw	RPw
Initial Value	0x00	0x0	0x000	0x000

Calculated Number of TRIGGER/STATE Increments to Action i, (RAM1a, i=0...31)[1)]

Bit 9:0	**DB:** number of events to Action_i (fractional part, i=0...31)[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...31)[1)]. **Note:** Use the maximum value for 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 **Reserved**

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 only available for device 4.

16.11.40 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...31)[1)]

Bit 23:0 **DTA:** calculated relative time to ACTION_i (i=0...31)[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 only available for device 4.

16.11.41 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 valid *TRIGGER* input.
measured TRIGGER time stamp
Note: The LSB address is determined using the SWON_T value in the OSW register (see chapter 16.11.8).

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

16.11.42 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_OL D																							
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 valid*TRIGGER* input.
previous measured TRIGGER time stamp
Note: The LSB address is determined using the SWON_T value in the OSW register (see chapter 16.11.8).

Bit 31:24 **Reserved**

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

16.11.43 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 valid *TRIGGER* input. transmitted filter value
Note: The LSB address is determined using the SWON_T value in the OSW register (see chapter 16.11.8).

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

16.11.44 Memory DPLL_TS_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_TS																							
Mode	R								RW																							
Initial Value	0x00								0x0000 00																							

Actual STATE Time Stamp Register

Bit 23:0 **STATE_TS:** Time stamp value of the last valid *STATE* input.
Note: The LSB address is determined using the SWON_S value in the OSW register (see chapter 16.11.8).

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

16.11.45 Memory DPLL_TS_S_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								STATE_TS_OLD																							
Mode	R								RW																							
Initial Value	0x00								0x000000																							

Previous STATE Time Stamp Register

Bit 23:0 **STATE_TS_old:** Time stamp value of the last valid *STATE* input.
Note: The LSB address is determined using the SWON_S value in the OSW register (see chapter 16.11.8).

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

16.11.46 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 valid *STATE* input.
transmitted filter value
Note: The LSB address is determined using the SWON register (see chapter 16.11.8).

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

16.11.47 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								Reserved								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 a valid 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 **Reserved**
Note: must be written to zero.

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

16.11.48 Memory DPLL_THMA

Address Offset:	**see Appendix B**		**Initial Value:** **0x0000_0000**
	31 … 24	23 … 16	15 … 0
Bit	Reserved	Reserved	THMA
Mode	R	RW	RW
Initial Value	0x00	0x00	0x0000

TRIGGER Hold Time Max Value

Bit 15:0 **THMA:** maximal time beween 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 **Reserved**
Note: must be written to zero.

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

16.11.49 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								Reserved								THVAL															
Mode	R								RW								RW															
Initial Value	0x00								0x00								0x0000															

Measured TRIGGER Hold Time Value

Bit 15:0	**THVAL:** measured time from the last valid slope to the next inactive *TRIGGER* slope in time stamp clock counts: this does mean the clock selected for the TBU_CH0_BASE (uint16); measured value
Bit 23:16	**Reserved** Note: must be written to zero.
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 valid and invalid slope is multiplied by 8. The register contains this value.

16.11.50 Memory DPLL_TOV

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	Reserved								Reserved								DW						DB									
Mode	R								RW								RW						RW									
Initial Value	0x00								0x00														0x000									

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

16.11.51 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	Reserved	DW	DB
Mode	R	RW	RW	RW
Initial Value	0x00	0x00	0x00	0x000

Time Out Value of Actvie 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 timeout 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 TOV_S) 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 **Reserved**
Note: must be written to zero.

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

16.11.52 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								0x000000																							

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.

16.11.53 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.

16.11.54 Memory DPLL_MPVAL1

Address Offset:	see Appendix B																Initial Value:								0x0000_0000							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	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 should not exceed the amount of NT*(MLT+1) or MLS1 respectively.

16.11.55 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 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.

16.11.56 Memory DPLL_NMB_T_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								Reserved								NMB_T_TAR															
Mode	R								RW								RW															
Initial Value	0x00								0x00								0x0000															

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_S value in the OSW register (see chapter 16.11.8).

Bit 23:16 **Reserved**
Note: must be written to zero.

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

16.11.57 Memory DPLL_NMB_T_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								Reserved								NMB_T_TAR_OLD															
Mode	R								RW								RW															
Initial Value	0x00								0x00								0x0000															

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_S value in the OSW register (see chapter 16.11.8).

Bit 23:16 **Reserved**
Note: must be written to zero.

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

16.11.58 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								Reserved				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 16.11.8).

Bit 23:20 **Reserved**
Note: must be written to zero.

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

16.11.59 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								Reserved				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 16.11.8).

Bit 23:20 **Reserved**
Note: Read as zero, should be written as zero.

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

16.11.60 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								0x000000																							

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.

16.11.61 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.

16.11.62 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.

16.11.63 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.

16.11.64 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								0x000000																							

Reciprocal Value of the Last Increment of TRIGGER

Bit 23:0 **RDT_T_actual:** 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 16.11.30).

Bit 31:24 **Reserved**

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

16.11.65 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	0x000000

Reciprocal Value of the Last Increment of STATE

Bit 23:0 **RDT_S_actual:** 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 16.11.30).

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

16.11.66 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_actual:** 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.

16.11.67 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_actual:** 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.

16.11.68 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								0x0000 00																							

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.

16.11.69 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 16.6.2.6
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.

16.11.70 Memory DPLL_EDT_S

Address Offset:	**see Appendix B**	**Initial Value:**	**0x0000_0000**
	31–24	23–0	
Bit	Reserved	EDT_S	
Mode	R	RW	
Initial Value	0x00	0x000000	

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 16.6.3.5

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

16.11.71 Memory DPLL_MEDT_S

Address Offset:	**see Appendix B**	**Initial Value:**	**0x0000_0000**
	31–24	23–0	
Bit	Reserved	MEDT_S	
Mode	R	RW	
Initial Value	0x00	0x000000	

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 16.6.3.6
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.

16.11.72 Memory DPLL_CDT_TX

Address Offset:	**see Appendix B**	**Initial Value:**	**0x0000_0000**
	31 … 24	23 … 0	
Bit	Reserved	CDT_TX	
Mode	R	RW	
Initial Value	0x00	0x0000 00	

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.

16.11.73 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	0x0000 00

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.

16.11.74 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	0x0000 00

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.

16.11.75 Memory DPLL_CDT_SX_NOM

Address Offset:	**see Appendix B**	**Initial Value:**	**0x0000_0000**

	31–24	23–0
Bit	Reserved	CDT_SX_NOM
Mode	R	RW
Initial Value	0x00	0x0000 00

Prediction of the Nominal STATE Increment Duration

Bit 23:0 **CDT_SX_nom:** Calculated duration of the current t nominal *STATE* event.
calculated value

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

16.11.76 Memory DPLL_TLR

Address Offset:	**see Appendix B**	**Initial Value:**	**0x00000000**

	31–24	23–8	7–0
Bit	Reserved	Reserved	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 **Reserved**
Note: must be written to zero.

Bit 31:24 **Reserved**

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

16.11.77 Memory DPLL_SLR

Address Offset:	see Appendix B	Initial Value: 0x00000000	
	31 … 24	23 … 8	7 … 0
Bit	Reserved	Reserved	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 **Reserved**
Note: must be written to zero.

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

16.11.78 Memory DPLL_PDT_[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								DW										DB													
Mode	R								RW										RW													
Initial Value	0x00								0x000										0x0000													

Projected Increment Sum Relations for Action [i]

Bit 13:0 **DB:** Fractional part of relation between *TRIGGER* or *STATE* increments.

Bit 23:14 **DW:** Integer part of relation between *TRIGGER* or *STATE* increments.
Definition of relation values between *TRIGGER* or *STATE* increments PDT_[i] according to Equations DPLL-11 or DPLL-13 (i = 0...31)$^{1)}$

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

$^{1)}$ **Note:** The PDT_i values for actions 24...31 are only available for device 4.

16.11.79 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								Reserved						MLS1																	
Mode	R								RW						RW																	
Initial Value	0x00								0x00						0x0000_0																	

Calculated Number of Sub-Pulses between two 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 *TRIGGER* events (to be set and updated by the CPU)

Bit 23:18 **Reserved**
Note: must be written to zero.

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

16.11.80 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								Reserved						MLS2																	
Mode	R								RW						RW																	
Initial Value	0x00								0x00						0x0000_0																	

Calculated Number of Sub-Pulses between two 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 **Reserved**
Note: must be written to zero.

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

16.11.81 Memory DPLL_CNT_NUM_1

Address Offset:	see Appendix B	Initial Value: 0x0000_0000
	31 … 24	23 … 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.

16.11.82 Memory DPLL_CNT_NUM_2

Address Offset:	see Appendix B	Initial Value: 0x0000_0000
	31 … 24	23 … 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.

16.11.83 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								0x000000																							

Plausibility Value of Next TRIGGER Slope

Bit 23:0 **PVT:** Plausibility value of next valid *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 valid *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 valid *TRIGGER* event is accepted. The wait time must be exceeded for a valid slope.

Note: When a valid *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.

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

16.11.84 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								0x000000																							

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 valid slope of *TRIGGER* in normal mode
when FTD=0: PSTC is set from actual position value, for the first valid *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)*(SYN_T) +PD; while PD=0 for AMT=0
SMC=1: (MLS1)*(SYN_T) +PD; while PD=0 for AMT=0

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

16.11.85 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 valid 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 valid slope of *STATE* (PD_S_store=0 for AMS=0):
SMC=0: add MLS1*(SYN_S) + PD_S_store;
SMC=1: add MLS2*(SYN_S) + PD_S_store;

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

16.11.86 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								0x0000 00																							

Measured Position Stamp at Last TRIGGER Input

Bit 23:0 **PSTM: Position stamp of TRIGGER, measured;** Measured position stamp of last valid *TRIGGER* input.
Store the value TBU_TS1 when a valid 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 16.11.8).

16.11.87 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 valid *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 16.11.8).

16.11.88 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								0x000000																							

Measured Position Stamp at Last STATE Input

Bit 23:0 **PSSM: Position stamp of STATE, measured;** Measured position stamp of last valid *STATE* input.
Store the value TBU_TS1 or TBU_TS2 respectively at the moment when a valid 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 16.11.8).

16.11.89 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								0x0000 00																							

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 valid *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 16.11.8).

16.11.90 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	Reserved	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 **Reserved**
Note: must be written to zero.

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

16.11.91 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	Reserved	NMB_S
Mode	R	RW	RW
Initial Value	0x00	0x0	0x0000 0

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 **Reserved**
Note: must be written to zero.

Bit 31:24 **Reserved**

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

16.11.92 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 Durations in FULL_SCALE

Bit 23:0 **RDT_S: Reciprocal difference time of TRIGGER;** 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.

16.11.93 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								0x000000																							

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 valid *STATE* event. **Note:** There are 2* (SNU+1) entries.
Bit 31:24	**Reserved** Note: Read as zero, should be written as zero.

16.11.94 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	Reserved	NS	PD_S
Mode	R	RW	RW	RW
Initial Value	0x00	0x0	000	0x0000

Adapt and Profile Values of the STATE Increments in FULL_SCALE

Bit 15:0	**PD_S: Physical deviation of STATE;** Adapt values for each *STATE* increment in FULL_SCALE (sint16); This value represents the number of pulses to be added to the correspondent increment. The absolute value of a negative PD_S must not exceed MLS1 or MLS2 respectively.
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	**Reserved** Note: must be written to zero.
Bit 31:24	**Reserved** Note: Read as zero, should be written as zero.

16.11.95 Memory DPLL_DT_S[i]

Address Offset:	**see Appendix B**	**Initial Value:**	**0x0000_0000**
	31 ... 24	23 ... 0	
Bit	Reserved	DT_S	
Mode	R	RW	
Initial Value	0x00	0x000000	

Nominal STATE Increment Durations 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.

16.11.96 Register DPLL_TSAC[i]

Address Offset:	**see Appendix B**	**Initial Value:**	**0x007F_FFFF**
	31 ... 24	23 ... 0	
Bit	Reserved	TSAC	
Mode	R	RPw	
Initial Value	0x00	0x7FFFFF	

Calculated Time Value to start Action i

Bit 23:0 **TSAC** calculated time stamp for ACTION_i (i = 0...31)[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 DPLL_TSAC24...31 are only available for device 4.

16.11.97 Register DPLL_PSAC[i]

Address Offset:	**see Appendix B**	**Initial Value:** **0x007F_FFFF**
	31 … 24	23 … 0
Bit	Reserved	PSAC
Mode	R	RPw
Initial Value	0x00	0x7FFFFF

Calculated Position Value to start Action i

Bit 23:0	**PSAC:** Calculated position value for the start of ACTION_i in normal or emergency mode according to equations DPLL-17 or DPLL-20 respectively (i = 0...31)[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 DPLL_PSAC24...31 are only available for device 4.

16.11.98 Register DPLL_ACB_[j] (j=0..7) [1]

Address Offset:	**see Appendix B**						**Initial Value:** **0x0000_0000**
	31 … 29	28 … 24	23 … 21	20 … 16	15 … 13	12 … 8	7 … 5	4 … 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*j **Note:** This value can only be written when the DPLL is disabled.
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*j **Note:** This value can only be written 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*j **Note:** This value can only be written 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*j **Note:** This value can only be written when the DPLL is disabled.
Bit 31:29	**Reserved** Note: Read as zero, should be written as zero.

[1)] **Note:** The DPLL_ACB_6 and DPLL_ACB_7 register are only available for device 4.

16.12DPLL 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.

16.12.1 Memory DPLL_RDT_T[i]

Address Offset:	**see Appendix B**	**Initial Value:** 0x0000_0000
	31 … 24	23 … 0
Bit	Reserved	RDT_T
Mode	R	RW
Initial Value	0x00	0x0000 00

Reciprocal Values of the Nominal TRIGGER Increment Durations 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.

16.12.2 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	0x000000

Time Stamp Values of the Nominal TRIGGER Increments in FULL_SCALE

Bit 23:0 **TSF_T:** Time stamp field of valid 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.

16.12.3 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	Reserved	NT	TINT	PD
Mode	R	RW	RW	RW	RW
Initial Value	0x00	0x00	000	000	0x0000

Adapt and Profile Values of the TRIGGER Increments in FULL_SCALE

Bit 12:0 **PD: Physical deviation;** Adapt values for each *TRIGGER* increment in FULL_SCALE (sint13);
the PD value does mean the number of SUB_INC1 pulses to be added to NT*(MLT+1);
the absolute value of a negative PD must not exceed NT*(MLT+1) or MLS1 respectively;
systematic missing *TRIGGER* events must not 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 **Reserved**
Note: must be written to zero.

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

16.12.4 Memory DPLL_DT_T[i]

Address Offset:	**see Appendix B**	**Initial Value:**	**0x0000_0000**
	31 … 24	23 … 0	
Bit	Reserved	DT_T	
Mode	R	RW	
Initial Value	0x00	0x0000 00	

Nominal TRIGGER Increment Durations in FULL_SCALE

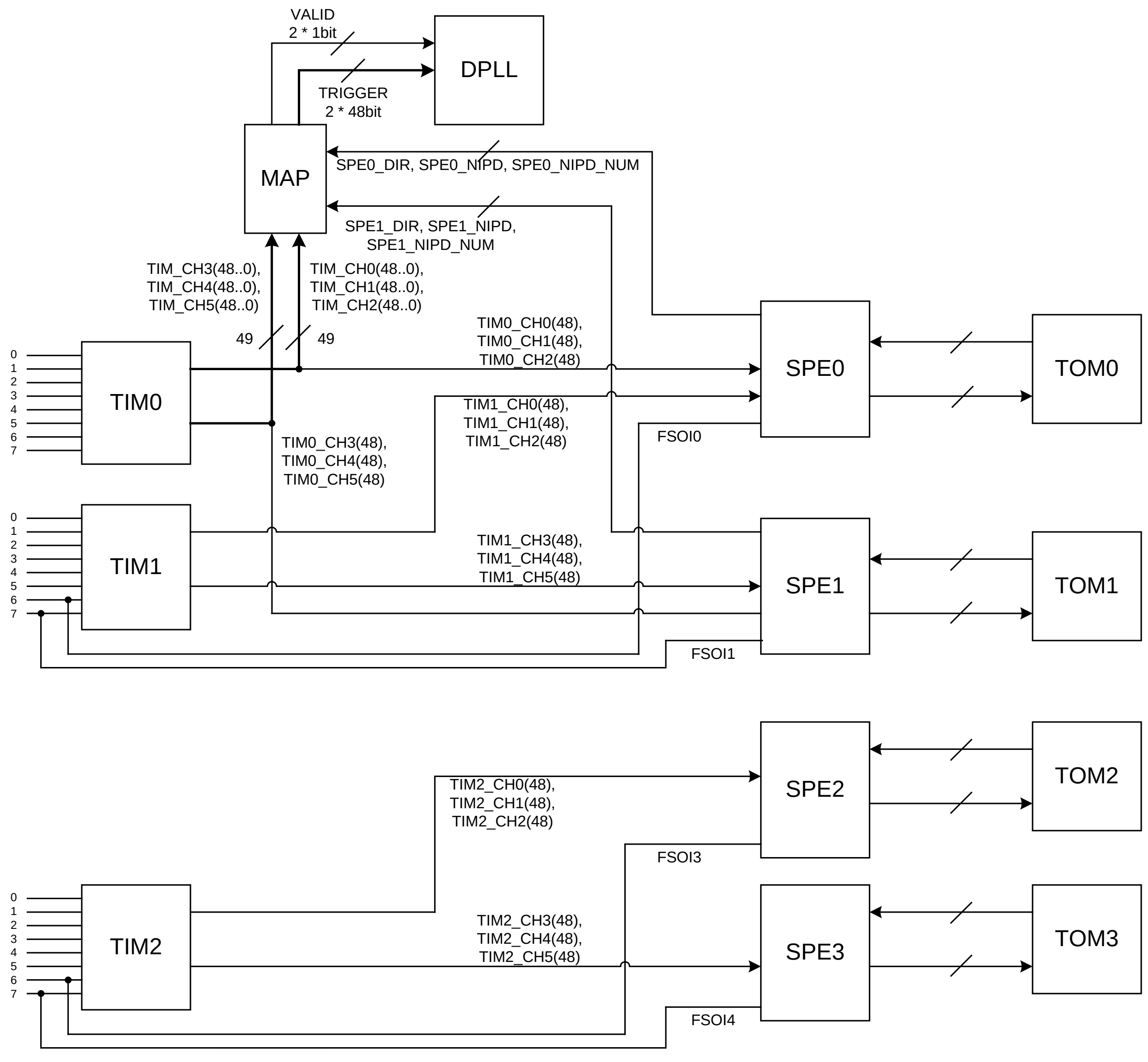
Bit 23:0	**DT_T: Difference time of TRIGGER;** increment duration values for each *TRIGGER* increment in FULL_SCALE divided by the number of nominal increments (nominal value). **Note:** There are 2* (TNU+1- SYN_NT) entries. The maximum number of entries is restricted to a value corresponding to the OSS value in the DPLL_OSW register.
Bit 31:24	**Reserved** Note: Read as zero, should be written as zero.

17 Sensor Pattern Evaluation (SPE)

17.1 Overview

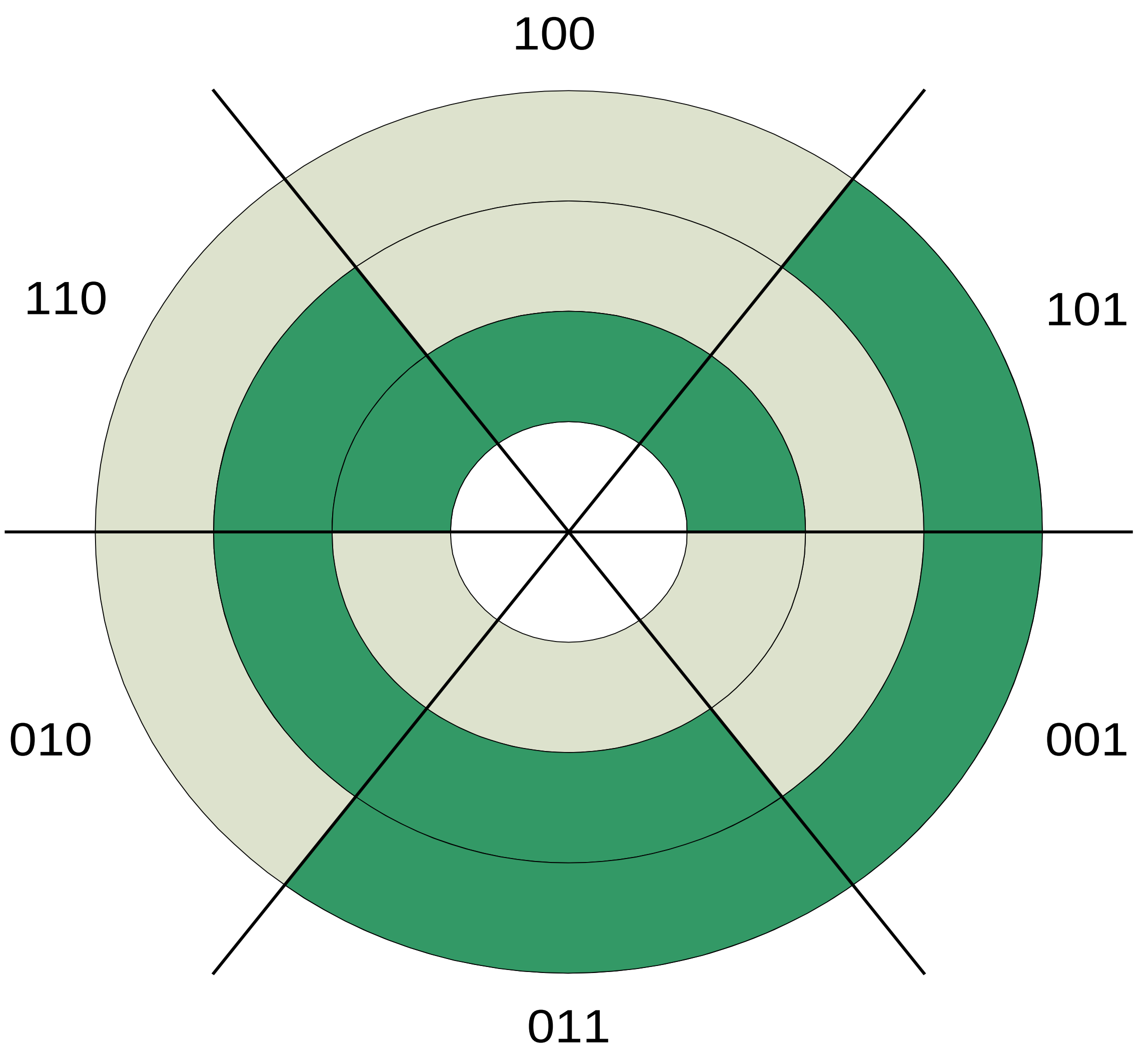
The Sensor Pattern Evaluation (SPE) submodule can be used to evaluate three hall sensor inputs and together with the TOM module to support the drive of BLDC engines. Thus, the input signals are filtered already in the connected TIM channels. In addition, the SPE submodule can be used as an input stage 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 17.1.1.

17.1.1 SPE Submodule integration concept into GTM-IP


VALID
2 * 1bit
TRIGGER
2 * 48bit
DPLL
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)
TIM1_CH0(48), TIM1_CH1(48), TIM1_CH2(48)
TIM0_CH3(48), TIM0_CH4(48), TIM0_CH5(48)
TIM1_CH3(48), TIM1_CH4(48), TIM1_CH5(48)
TIM0
TIM1
TIM2
SPE0
SPE1
SPE2
SPE3
TOM0
TOM1
TOM2
TOM3
FSOI0
FSOI1
FSOI3
FSOI4
TIM2_CH0(48), TIM2_CH1(48), TIM2_CH2(48)
TIM2_CH3(48), TIM2_CH4(48), TIM2_CH5(48)
0 1 2 3 4 5 6 7

As mentioned above, 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 matching 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 17.1.2. In general, the input pattern is programmable within the SPE submodule.

17.1.2 SPE Sample input pattern for *TIM[i]_CH[x,y,z](48)*



In figure 17.1.2 the input signals define the pattern from the input sensors which have a 50% high and 50% low phase. The pattern according to figure 17.1.2 is as follows:

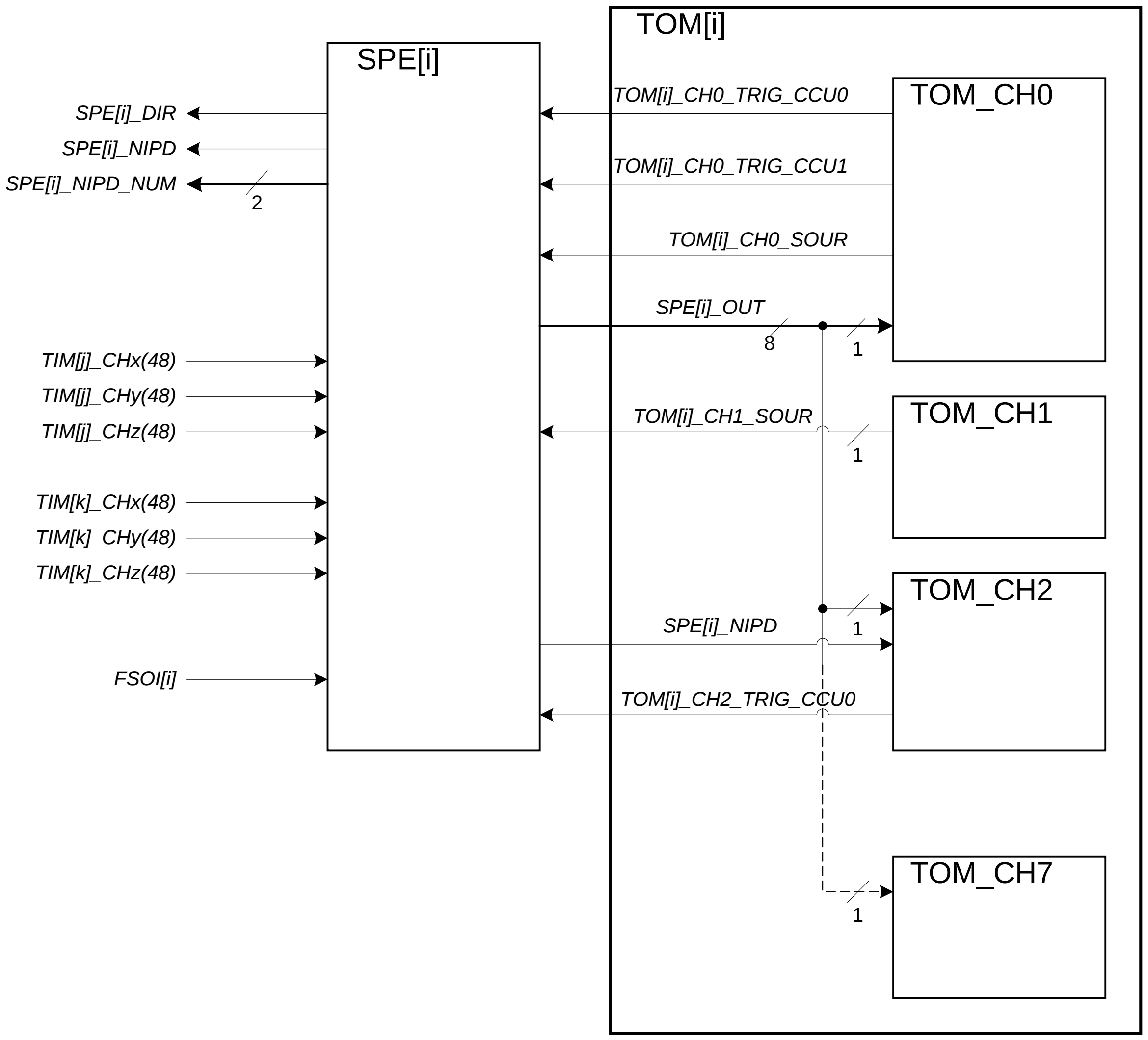
100 – 110 – 010 – 011 – 001 – 101 – 100
where the first bit (smallest circle) represents *TIM[i]_CH[x](48)*, the second bit represents *TIM[i]_CH[y](48)*, and the third bit (greatest circle) represents *TIM[i]_CH[z](48)*.

Note that the SPE module expects that with every new pattern only one of the three input signals changes its value.

17.2 SPE Submodule description

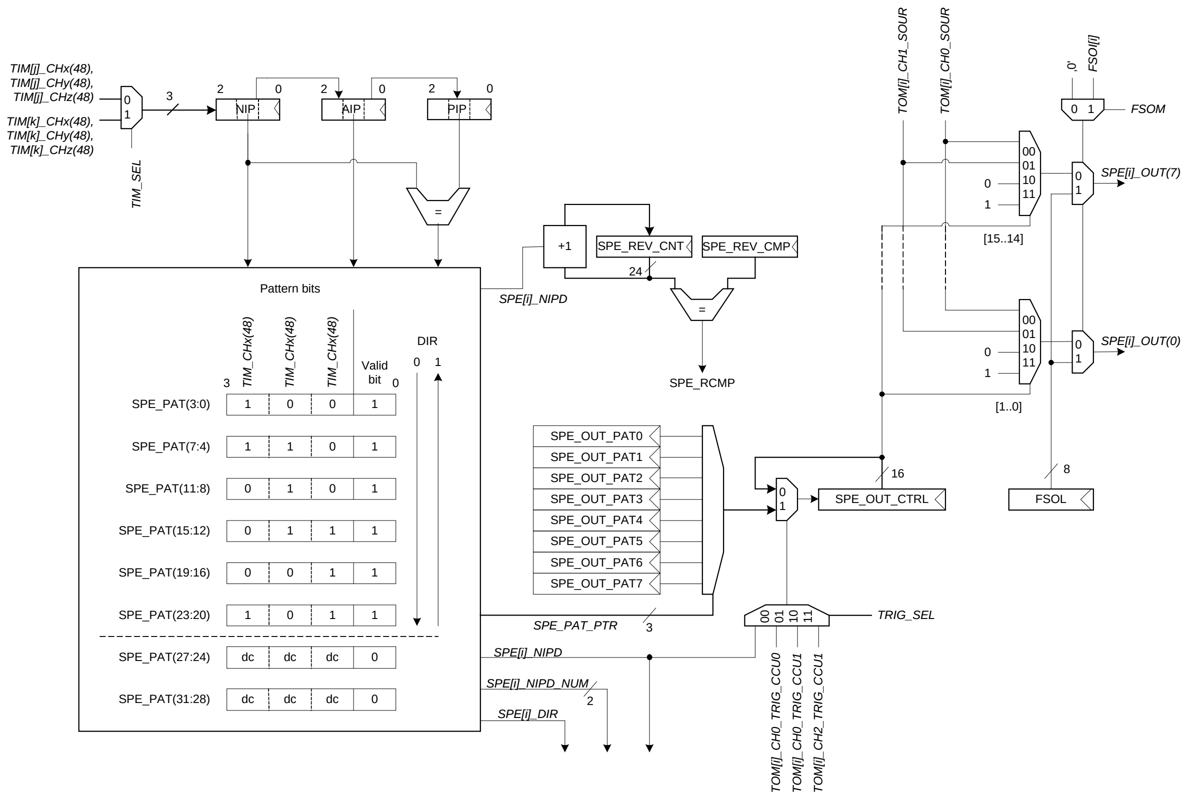
The SPE submodule can handle sensor pattern inputs. Every time if one of the input signals *TIM[i]_CH[x](48),TIM[i]_CH[y](48) or TIM[i]_CH[z](48)* changes its value, a sample of all three input signals is made. Derived from the sample of the three inputs the encoded rotation direction and the validity of the input pattern sequence can be detected and signalled. 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 17.2.1.

17.2.1 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 17.2.2.

17.2.2 SPE Submodule architecture



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 17.2.2 shows the programming of the **SPE[i]_PAT** register for the input pattern defined in figure 17.1.2.

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.

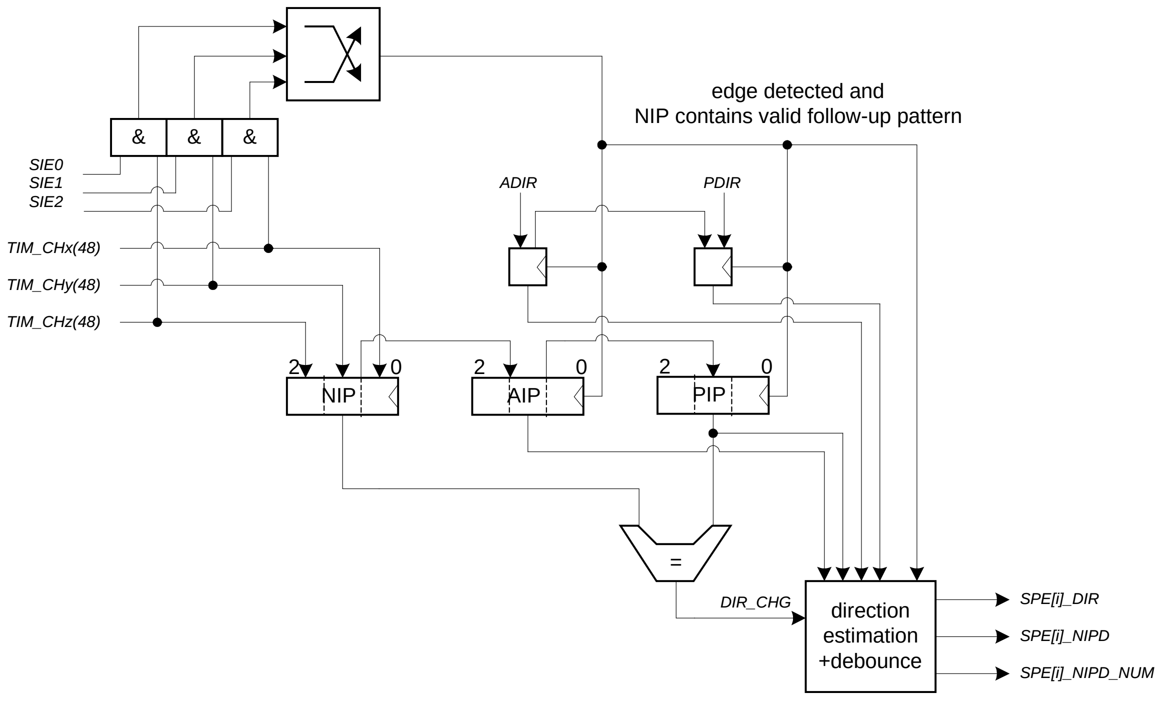
The TOM[i]_CH2 with i=0..3 can be used together with the SPE module to trigger a delayed update of the **SPE_OUT_CTRL** register after new input pattern detected by SPE (signalled by *SPE[i]_NIPD*).
To do this, the TOM[i]_CH2 has to be configured to work in one-shot mode (set bit **OSM** in register **TOM[i]_CH2_CTRL**). The SPE mode of this channel has to be enabled, too (set bit **SPEM** in register **TOM[i]_CH2_CTRL**). The SPE module has to be configured to update **SPE_OUT_CTRL** on *TOM[i]_CH2_TRIG_CCU1* (set in **SPE[i]_CTRL_STAT** bits **TRIG_SEL** to '11'). Then, on new input detected by SPE, the signal *SPE[i]_NIPD* triggers the start of the TOM channel 2 to generates one PWM period by resetting **CN0** to 0. On second PWM edge triggered by CCU1 of TOM channel 2, the signal *TOM[i]_CH2_TRIG_CCU1* triggers the update of **SPE_OUT_CTRL**.

According to figure 17.2.2, the two input patterns “000” and “111” 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 "010" does not follow the pattern "110" or "011".

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 analysing the bit pattern 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 17.2.3.

Additionally to the sensor pattern evaluation the SPE module also provides the feature of fast shut-off 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 shut-off level itself is defined in the bit field FSOL of register **SPE[i]_CTRL_STAT**. The TIM input used to trigger the fast shut-off 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 17.1.1.

17.2.3 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 behaviour 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.

17.2.4 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 24bit 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.

17.3 SPE Interrupt signals

The following table describes 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.

17.4 SPE Register overview

The following table shows an overview about the SPE register set.

Register name	**Description**	**Details in Section**
SPE[i]_CTRL_STAT	SPE Control status register	17.5.1
SPE[i]_PAT	SPE Input pattern definition register.	17.5.2
SPE[i]_OUT_PAT[x]	SPE Output definition registers. (x: 0...7)	17.5.3
SPE[i]_OUT_CTRL	SPE output control register	17.5.4
SPE[i]_REV_CNT	SPE input revolution counter	17.5.5
SPE[i]_REV_CMP	SPE Revolution counter compare value	17.5.6
SPE[i]_IRQ_NOTIFY	SPE Interrupt notification register.	17.5.7
SPE[i]_IRQ_EN	SPE Interrupt enable register.	17.5.8

SPE[i]_EIRQ_EN	SPE Error interrupt enable register.	17.5.11
SPE[i]_IRQ_FORCINT	SPE Interrupt generation by software.	17.5.9
SPE[i]_IRQ_MODE	IRQ mode configuration register	17.5.10

17.5 SPE Register description

17.5.1 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								Reserved	NIP			PDIR	PIP			ADIR	AIP			Reserved	SPE_PAT_PTR			FSOM	TIM_SEL	TRIG_SEL		SIE2	SIE1	SIE0	EN
Mode	RW								R	R			RW	RW			RW	RW			R	RW			RW	RW	RW		RW	RW	RW	RW
Initial Value	0x00								00000	000			0	000			0	000			0	000			0	0	0		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. 00 = *SPE[i]_NIPD* selected. 01 = *TOM_CH0_TRIG_CCU0* selected. 10 = *TOM_CH0_TRIG_CCU1* selected. 11 = *TOM_CH2_TRIG_CCU1* selected.

Bit 6	**TIM_SEL:** select TIM input signal SPE0: 0 = TIM0_CH0..2 1 = TIM1_CH0..2 SPE1: 0 = TIM0_CH3..5 1 = TIM1_CH3..5 SPE2: 0 = TIM2_CH0..2 1 = unused SPE3: 0 = TIM2_CH3..5 1 = unused
Bit 7	**FSOM:** Fast Shut-Off Mode 0 = Fast Shut-Off mode disabled 1 = Fast Shut-Off 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. 000: **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	**Reserved:** Note: Read as zero, should be written as zero

Bit 31:24	**FSOL:** Fast Shut-Off Level for TOM[i] channel 0 to 7

17.5.2 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	000			0	000			0	000			0	000			0	000			0	000			0	000			0	000			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 commutator. All input pattern following the first marked invalid input pattern are ignored.

17.5.3 Register SPE[i]_OUT_PAT[x] (x: 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]
00 = set *SPE_OUT(n)* to *TOM_CH0_SOUR*
01 = set *SPE_OUT(n)* to *TOM_CH1_SOUR*
10 = set *SPE_OUT(n)* to *'0'*
11 = 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[x]_PAT with x:0..7.

17.5.4 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 00 = set *SPE_OUT(n)* to *TOM_CH0_SOUR* 01 = set *SPE_OUT(n)* to *TOM_CH1_SOUR* 10 = set *SPE_OUT(n)* to *'0'* 11 = 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

17.5.5 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

17.5.6 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 decremention of **SPE[i]_REV_CNT** is applied, due to a input signal change. Any update of **SPE[i]_REV_CNT** or **SPE[i]_REV_CMP** via AEI does not raise an *SPE[i]_RCMP* interrupt.

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

17.5.7 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

17.5.8 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_EN	SPE_BIS_IRQ_EN	SPE_PERR_IRQ_EN	SPE_DCHG_IRQ_EN	SPE_NIPD_IRQ_EN
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

17.5.9 Register SPE[i]_IRQ_FORCINT

Address Offset:	**see Appendix B**	**Initial Value:**	**0x0000_0000**				
	31 … 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

17.5.10 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 00 = Level mode 01 = Pulse mode 10 = Pulse-Notify mode 11 = 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

17.5.11 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_EN	SPE_BIS_EIRQ_EN	SPE_PERR_EIRQ_EN	SPE_DCHG_EIRQ_EN	SPE_NIPD_EIRQ_EN
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

18 Interrupt Concentrator Module (ICM)

18.1 Overview

The Interrupt Concentrator Module (ICM) is used to bundle the GTM-IP interrupt lines of the individual submodules 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 submodules and channels have to be enabled or disabled inside the submodules and channels.

The feed through architecture of bundled interrupt lines is used for the submodules AEI, ARU, BRC, CMP, SPE, PSM, TIM, DPLL, TOM, ATOM and MCS.

To determine the detailed interrupt source the microcontroller has to read the submodule/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 submodules themselves.

18.2 Bundling

The GTM-IP submodule individual interrupt sources are connected to the ICM. There, the individual interrupt lines are either feed through and signalled to the outside world or bundled a second time into groups and are then signalled to the outside world.
The ICM interrupt bundling is described in the following sections.

18.2.1 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 submodules. In this interrupt group each individual channel of the submodules 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 submodule 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.

18.2.2 DPLL Interrupt Bundling

The DPLL Interrupt group handles the interrupts coming from the DPLL submodule 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.

18.2.3 TIM Interrupt Bundling

Inside this group submodules which handle GTM-IP input signals are treated. This is the case for the TIM[i] submodules. Each TIM submodule 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 submodule channel interrupts and typically not read out by the CPU, but it is readable.

18.2.4 MCS Interrupt Bundling

For complex signal output generation, the MCS submodules are used inside the GTM-IP. Each of these MCS submodules has eight channels with one interrupt line. This interrupt line is connected to the ICM submodule and is feed through directly to the outside world.

In addition the interrupt line status are shown in the **ICM_IRQG_4** and **ICM_IRQG_5** register. Typically, the interrupt source is determined by the corresponding interrupt line and the **ICM_IRQ4(/_5)** register is typically not read out by the CPU, but it is readable.

18.2.5 TOM and ATOM Interrupt Bundling

For the TOM and ATOM submodules, the interrupts are bundled within the ICM submodule a second time to reduce external interrupt lines. The interrupts are OR-ed 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 chapter 18.2.5.1.

*18.2.5.1 **TOM and ATOM interrupt bundling within ICM***

TOM[i]-input IRQs [i]=0..number of TOM's-1	TOM-output IRQs (OR-ed)		ATOM[i]-input IRQs [i]=0..number of ATOM's-1	ATOM-output IRQs (OR-ed)
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] submodules 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 submodule 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 submodule channel itself.

The **ICM_IRQG_6(/_7/_8)** register bits are cleared automatically, when their corresponding interrupt in the submodule channels is cleared.

The interrupts coming from the ATOM[i] submodules 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 submodule 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 submodule channel itself.

The interrupts coming from the ATOM[i] submodules are registered in the **ICM_IRQG_9** register. ATOM0,ATOM1 and ATOM2 are bundled in **ICM_IRQG_9.** To identify the ATOM submodule channel where the interrupt occurred, the CPU has to read out the **ICM_IRQG_9** register first before it goes to the ATOM submodule channel itself.

The **ICM_IRQG_9(/_10/_11)** register bits are cleared automatically, when their corresponding interrupt in the submodule channels is cleared.

18.2.6 Module Error Interrupt Bundling

The Module Error Interrupt group handles the error interrupts coming from the BRC, FIFO, TIM, MCS, SPE, CMP, DPLL submodule 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.

The **ICM_IRQG_MEI** register bits are cleared automatically, when their corresponding error interrupt in the submodule is cleared.

18.2.7 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 submodule channel is cleared.

18.2.8 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 submodules TIM0, TIM1, TIM2 and TIM3 in the **ICM_IRQG_CEI1** error interrupt group register and for the submodules 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 submodule channel is cleared.

18.2.9 MCS Channel Error Interrupt Bundling

The MCS Channel Error Interrupt group handles the error interrupts coming from the MCS channel of the GTM-IP. The MCS Channel Error interrupts are additionally identified for the submodules MCS0, MCS1, MCS2 and MCS3 in the **ICM_IRQG_CEI3** error interrupt group register and for the submodules MCS4, MCS5 and MCS6 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_CEI3** and **ICM_IRQG_CEI4** register bits are cleared automatically, when their corresponding error interrupt in the submodule channel is cleared.

18.3 ICM Interrupt Signals

Following table shows the GTM-IP interrupt lines that are visible at the outside of the IP.

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…7) (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

18.4 ICM Configuration Registers Overview

ICM contains following configuration registers:

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)	18.5.1
ICM_IRQG_1	ICM Interrupt group register covering DPLL	18.5.2
ICM_IRQG_2	ICM Interrupt group register covering TIM0, TIM1, TIM2, TIM3	18.5.3
ICM_IRQG_3	ICM Interrupt group register covering TIM4, TIM5, TIM6, TIM7	18.5.4
ICM_IRQG_4	ICM Interrupt group register covering MCS0 to MCS3 submodules	18.5.5
ICM_IRQG_5	ICM Interrupt group register covering MCS4 to MCS6 submodules	18.5.6
ICM_IRQG_6	ICM Interrupt group register covering GTM-IP output submodules TOM0 to TOM1	18.5.7
ICM_IRQG_7	ICM Interrupt group register covering GTM-IP output submodules TOM2 to TOM3	18.5.8
ICM_IRQG_8	ICM Interrupt group register covering GTM-IP output submodules TOM4 to TOM5	18.5.9
ICM_IRQG_9	ICM Interrupt group register covering GTM-IP output submodules ATOM0, ATOM1, ATOM2 and ATOM3	18.5.10
ICM_IRQG_10	ICM Interrupt group register covering GTM-IP output submodules ATOM4 to ATOM7	18.5.11
ICM_IRQG_11	ICM Interrupt group register covering GTM-IP output submodules ATOM8 to ATOM11	18.5.12
ICM_IRQG_MEI	ICM Interrupt group register for module error interrupt information	18.5.13
ICM_IRQG_CEI0	ICM Interrupt group register 0 for channel error interrupt information	18.5.14
ICM_IRQG_CEI1	ICM Interrupt group register 1 for channel error interrupt information	18.5.15
ICM_IRQG_CEI2	ICM Interrupt group register 2 for channel error interrupt information	18.5.16
ICM_IRQG_CEI3	ICM Interrupt group register 3 for channel error interrupt information	18.5.17
ICM_IRQG_CEI4	ICM Interrupt group register 4 for channel error interrupt information	18.5.18

18.5 ICM Configuration Registers Description

18.5.1 Register ICM_IRQG_0 (GTM Infrastructure Interrupt Group)

Address Offset:	**see Appendix B**																**Initial Value:**						**0x0000_0000**									
	31	30	29	28	27	26	25	24	23	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						SPE3_IRQ	SPE2_IRQ	SPE1_IRQ	SPE0_IRQ	CMP_IRQ	AEI_IRQ	BRC_IRQ	ARU_ACC_ACK_IRQ	ARU_NEW_DATA1_IRQ	ARU_NEW_DATA0_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
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

Bit 0	**ARU_NEW_DATA0_IRQ:** *ARU_NEW_DATA0* interrupt 0 = no interrupt occurred 1 = interrupt was raised by the corresponding submodule **Note**: This bit is only set, when the interrupt is enabled in the interrupt enable register of the corresponding submodule.
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_IRQ:** BRC shared submodule interrupt. See bit 0.
Bit 4	**AEI_IRQ:** *AEI_IRQ* interrupt. See bit 0.
Bit 5	**CMP_IRQ:** CMP shared submodule interrupt. See bit 0.
Bit 6	**SPE0_IRQ:** SPE0 shared submodule interrupt. See bit 0.
Bit 7	**SPE1_IRQ:** SPE1 shared submodule interrupt. See bit 0.
Bit 8	**SPE2_IRQ:** SPE2 shared submodule interrupt. See bit 0.
Bit 9	**SPE3_IRQ:** SPE3 shared submodule interrupt. See bit 0.
Bit 15:10	**Reserved** **Note:** Read as zero, should be written as zero
Bit 16	**PSM0_CH0_IRQ:** PSM0 shared submodule channel 0 interrupt 0 = no interrupt occurred 1 = interrupt was raised by the corresponding submodule **Note**: This bit is only set, when the interrupt is enabled in the interrupt enable register of the corresponding submodule. **Note:** When set this bit represents one of the four interrupt sources *FIFO_[x]_EMPTY*, *FIFO_[x]_FULL*, *FIFO_[x]_LOWER_WM* or *FIFO_[x]_UPPER_WM*

Bit 17	**PSM0_CH1_IRQ:** PSM0 shared submodule channel 1 interrupt. See bit 16.
Bit 18	**PSM0_CH2_IRQ:** PSM0 shared submodule channel 2 interrupt. See bit 16.
Bit 19	**PSM0_CH3_IRQ:** PSM0 shared submodule channel 3 interrupt. See bit 16.
Bit 20	**PSM0_CH4_IRQ:** PSM0 shared submodule channel 4 interrupt. See bit 16.
Bit 21	**PSM0_CH5_IRQ:** PSM0 shared submodule channel 5 interrupt. See bit 16.
Bit 22	**PSM0_CH6_IRQ:** PSM0 shared submodule channel 6 interrupt. See bit 16.
Bit 23	**PSM0_CH7_IRQ:** PSM0 shared submodule channel 7 interrupt. See bit 16.
Bit 24	**PSM1_CH0_IRQ:** PSM1 shared submodule channel 0 interrupt 0 = no interrupt occurred 1 = interrupt was raised by the corresponding submodule **Note**: This bit is only set, when the interrupt is enabled in the interrupt enable register of the corresponding submodule. **Note:** When set this bit represents one of the four interrupt sources *FIFO_[x]_EMPTY*, *FIFO_[x]_FULL*, *FIFO_[x]_LOWER_WM* or *FIFO_[x]_UPPER_WM*
Bit 25	**PSM1_CH1_IRQ:** PSM1 shared submodule channel 1 interrupt. See bit 16.
Bit 26	**PSM1_CH2_IRQ:** PSM1 shared submodule channel 2 interrupt. See bit 16.
Bit 27	**PSM1_CH3_IRQ:** PSM1 shared submodule channel 3 interrupt. See bit 16.
Bit 28	**PSM1_CH4_IRQ:** PSM1 shared submodule channel 4 interrupt. See bit 16.
Bit 29	**PSM1_CH5_IRQ:** PSM1 shared submodule channel 5 interrupt. See bit 16.
Bit 30	**PSM1_CH6_IRQ:** PSM1 shared submodule channel 6 interrupt. See bit 16.
Bit 31	**PSM1_CH7_IRQ:** PSM1 shared submodule channel 7 interrupt. See bit 16.

18.5.2 Register ICM_IRQG_1 (DPLL Interrupt Group)

Address Offset:	**see Appendix B**															**Initial Value:**				**0x0000_0000**												
	31	30	29	28	27	26	25	24	23	22	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 submodule **Note**: This bit is only set, when the interrupt is enabled in the interrupt enable register of the corresponding submodule.
Bit 1	**DPLL_EDI_IRQ:** DPLL enable/disable interrupt. See bit 0.
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

18.5.3 Register ICM_IRQG_2 (TIM Interrupt Group 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
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
Mode	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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit	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
Initial Value	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 submodule **Note**: This bit is only set, when the interrupt is enabled in the interrupt enable register of the corresponding submodule. **Note:** When set this bit represents one of the six interrupt sources *NEWVALx_IRQ*, *ECNTOFLx_IRQ*, *CNTOFLx_IRQ*, *GPRXOFLx_IRQ*, GLITCHDETx_IRQ or *TOx_IRQ*.
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.

18.5.4 Register ICM_IRQG_3 (TIM Interrupt Group 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								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
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 **TIM4_CH0_IRQ:** TIM4 shared interrupt channel 0.
0 = no interrupt occurred
1 = interrupt was raised by the corresponding submodule
Note: This bit is only set, when the interrupt is enabled in the interrupt enable register of the corresponding submodule.

Note: When set this bit represents one of the six interrupt sources *NEWVALx_IRQ*, *ECNTOFLx_IRQ*, *CNTOFLx_IRQ*, *GPRXOFLx_IRQ*, GLITCHDETx_IRQ or *TOx_IRQ*.

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

18.5.5 Register ICM_IRQG_4 (MCS Interrupt Group 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	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 submodule
Note: This bit is only set, when the interrupt is enabled in the interrupt enable register of the corresponding submodule.

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.

18.5.6 Register ICM_IRQG_5 (MCS Interrupt Group 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								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
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 **MCS4_CH0_IRQ:** MCS4 channel 0 interrupt
0 = no interrupt occurred
1 = interrupt was raised by the corresponding submodule
Note: This bit is only set, when the interrupt is enabled in the interrupt enable register of the corresponding submodule.

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

18.5.7 Register ICM_IRQG_6 (TOM Interrupt Group 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	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 submodule **Note:** This bit is only set, when the interrupt is enabled in the interrupt enable register of the corresponding submodule.
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.

18.5.8 Register ICM_IRQG_7 (TOM Interrupt Group 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	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 submodule
Note: This bit is only set, when the interrupt is enabled in the interrupt enable register of the corresponding submodule.

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.

18.5.9 Register ICM_IRQG_8 (TOM Interrupt Group 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	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 submodule
Note: This bit is only set, when the interrupt is enabled in the interrupt enable register of the corresponding submodule.

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.

18.5.10 Register ICM_IRQG_9 (ATOM Interrupt Group 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	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 submodule

Note: This bit is only set, when the interrupt is enabled in the interrupt enable register of the corresponding submodule.

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.

18.5.11 Register ICM_IRQG_10 (ATOM Interrupt Group 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	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 submodule **Note**: This bit is only set, when the interrupt is enabled in the interrupt enable register of the corresponding submodule.
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.

18.5.12 Register ICM_IRQG_11 (ATOM Interrupt Group 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	ATOM11_CH7_IRQ	ATOM11_CH6_IRQ	ATOM11_CH5_IRQ	ATOM11_CH4_IRQ	ATOM11_CH3_IRQ	ATOM11_CH2_IRQ	ATOM11_CH1_IRQ	ATOM11_CH0_IRQ	ATOM10_CH7_IRQ	ATOM10_CH6_IRQ	ATOM10_CH5_IRQ	ATOM10_CH4_IRQ	ATOM10_CH3_IRQ	ATOM10_CH2_IRQ	ATOM10_CH1_IRQ	ATOM10_CH0_IRQ	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 submodule
Note: This bit is only set, when the interrupt is enabled in the interrupt enable register of the corresponding submodule.

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.

18.5.13 Register ICM_IRQG_MEI (Module Error Interrupt)

Address Offset:	**see Appendix B**																**Initial Value:**								**0x0000_0000**							
	31	30	29	28	27	26	25	24	23	22	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	Reserved	MCS6_EIRQ	MCS5_EIRQ	MCS4_EIRQ	MCS3_EIRQ	MCS2_EIRQ	MCS1_EIRQ	MCS0_EIRQ	Reserved	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	**GTM_EIRQ:** GTM Error interrupt request 0 = no interrupt occurred 1 = interrupt was raised by the corresponding submodule **Note**: This bit is only set, when the error interrupt is enabled in the error interrupt enable register of the corresponding submodule.
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	**Reserved:** Read as zero, should be written as zero. **Note:** Read as zero, should be written as zero
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	**Reserved:** Read as zero, should be written as zero. **Note:** Read as zero, should be written as zero
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

18.5.14 Register ICM_IRQG_CEI0 (Channel Error Interrupt 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	Reserved																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
Initial Value	0x0000																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 submodule **Note**: This bit is only set, when the error interrupt is enabled in the error interrupt enable register of the corresponding submodule.
Bit 1	**FIFO0_CH1_EIRQ:** FIFO0 channel 1 shared interrupt. See bit 0.
Bit 2	**FIFO0_CH2_EIRQ:** FIFO0 channel 2 shared interrupt. See bit 0.
Bit 3	**FIFO0_CH3_EIRQ:** FIFO0 channel 3 shared interrupt. See bit 0.
Bit 4	**FIFO0_CH4_EIRQ:** FIFO0 channel 4 shared interrupt. See bit 0.
Bit 5	**FIFO0_CH5_EIRQ:** FIFO0 channel 5 shared interrupt. See bit 0.
Bit 6	**FIFO0_CH6_EIRQ:** FIFO0 channel 6 shared interrupt. See bit 0.
Bit 7	**FIFO0_CH7_EIRQ:** FIFO0 channel 7 shared interrupt. See bit 0.

Bit 8	**FIFO1_CH0_EIRQ:** FIFO1 channel 0 shared interrupt. See bit 0.
Bit 9	**FIFO1_CH1_EIRQ:** FIFO1 channel 1 shared interrupt. See bit 0.
Bit 10	**FIFO1_CH2_EIRQ:** FIFO1 channel 2 shared interrupt. See bit 0.
Bit 11	**FIFO1_CH3_EIRQ:** FIFO1 channel 3 shared interrupt. See bit 0.
Bit 12	**FIFO1_CH4_EIRQ:** FIFO1 channel 4 shared interrupt. See bit 0.
Bit 13	**FIFO1_CH5_EIRQ:** FIFO1 channel 5 shared interrupt. See bit 0.
Bit 14	**FIFO1_CH6_EIRQ:** FIFO1 channel 6 shared interrupt. See bit 0.
Bit 15	**FIFO1_CH7_EIRQ:** FIFO1 channel 7 shared interrupt. See bit 0.
Bit 31:16	**Reserved:** Read as zero, should be written as zero. **Note:** Read as zero, should be written as zero

18.5.15 Register ICM_IRQG_CEI1 (Channel Error Interrupt 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	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 submodule **Note:** This bit is only set, when the error interrupt is enabled in the error interrupt enable register of the corresponding submodule.
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.

18.5.16 Register ICM_IRQG_CEI2 (Channel Error Interrupt 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								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
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 **TIM4_CH0_EIRQ:** TIM4 channel 0 error interrupt
0 = no error interrupt occurred
1 = error interrupt was raised by the corresponding submodule
Note: This bit is only set, when the error interrupt is enabled in the error interrupt enable register of the corresponding submodule.

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

18.5.17 Register ICM_IRQG_CEI3 (Channel Error Interrupt 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	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 submodule **Note**: This bit is only set, when the error interrupt is enabled in the error interrupt enable register of the corresponding submodule.
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.

18.5.18 Register ICM_IRQG_CEI4 (Channel Error Interrupt 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								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
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	**MCS4_CH0_EIRQ:** MCS4 channel 0 error interrupt 0 = no error interrupt occurred 1 = error interrupt was raised by the corresponding submodule

	Note: This bit is only set, when the error interrupt is enabled in the error interrupt enable register of the corresponding submodule.
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 31:24	**Reserved:** Read as zero, should be written as zero. **Note:** Read as zero, should be written as zero

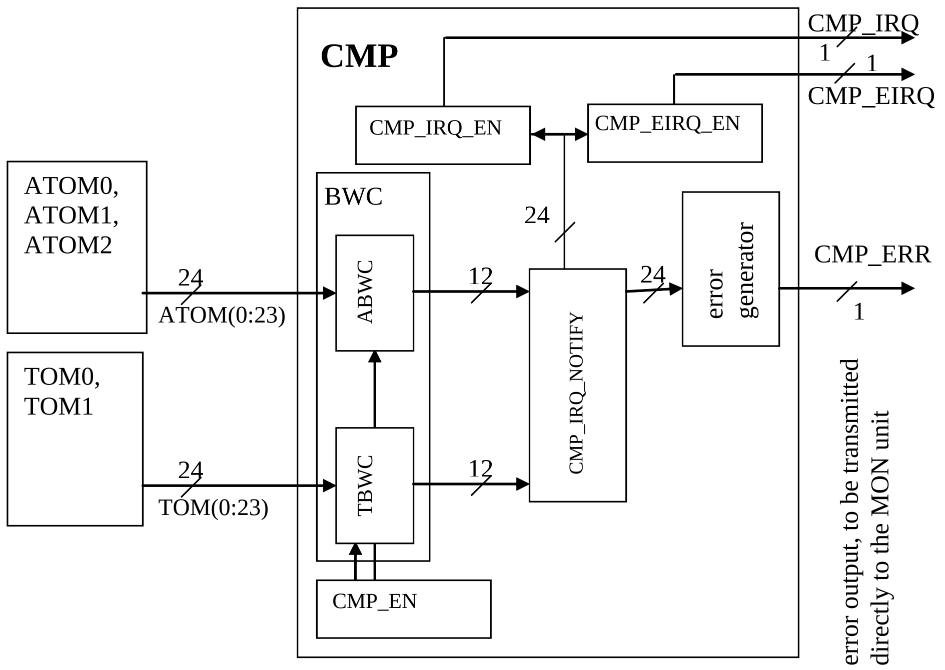
19 Output Compare Unit (CMP)

19.1 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 behaviour of the output modules to be compared. This is given when two ATOM units produce output signals at the same time stamp or when two TOMs have the same configuration and start their output generation at the same time. This is possible by means of the trigger mechanisms *TRIG_x* provided by the TOMs as shown in the chapter 11.1.1 (TOM Block Diagram). It is not necessary to compare each output channel with each other.

The CMP enables the comparison of 2x24 channels of the TOM and ATOM units respectively and is restricted to neighbour channels. Thus, channel 0 is compared with channel 1, channel 2 with 3 and so on until the comparison of channel 22 with channel 23.

19.1.1 Architecture of the Compare Unit



19.2 Bitwise Compare Unit (BWC)

The Bitwise Compare Unit compares in pairs the combinations shown in following table

TBWC/ABWC Comparator Number	**Compare TOM/ATOM Bit Number one**	**Compare TOM/ATOM Bit Number Two**	**Output Number**
0	0	1	0
1	2	3	1
2	4	5	2
3	6	7	3
4	8	9	4
5	10	11	5
6	12	13	6
7	14	15	7
8	16	17	8
9	18	19	9
10	20	21	10
11	22	23	11

19.3 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.

19.4 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.

The *CMP_ERR* output reflects its status in the status register of the Monitor Unit, which is to be polled by the CPU.

19.5 CMP Interrupt Signal

The CMP submodule 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.

Signal	**Description**
CMP_EIRQ	Mismatching interrupt of outputs to be compared, when enabled
CMP_IRQ	Mismatching interrupt of outputs to be compared, when enabled

19.6 CMP Configuration Registers Overview

CMP contains following configuration registers:

Register Name	Description	Details in Section
CMP_EN	Comparator enable register	19.7.1
CMP_IRQ_NOTIFY	Event notification register	19.7.2
CMP_IRQ_EN	Interrupt enable register	19.7.3
CMP_IRQ_FORCINT	Interrupt force register	19.7.4
CMP_IRQ_MODE	IRQ mode configuration register	19.7.5
CMP_EIRQ_EN	Error interrupt enable register	19.7.6

19.7 CMP Configuration Registers Description

19.7.1 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									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 19.2) 0 = ABWC Comparator 0 is disabled 1 = ABWC Comparator 0 is enabled
Bit 1	**ABWC1_EN:** Enable comparator 1 in ABWC (see chapter 19.2) see bit 0
Bit 2	**ABWC2_EN:** Enable comparator 2 in ABWC (see chapter 19.2) see bit 0
Bit 3	**ABWC3_EN:** Enable comparator 3 in ABWC (see chapter 19.2) see bit 0
Bit 4	**ABWC4_EN:** Enable comparator 4 in ABWC (see chapter 19.2)

	see bit 0
Bit 5	**ABWC5_EN:** Enable comparator 5 in ABWC (see chapter 19.2) see bit 0
Bit 6	**ABWC6_EN:** Enable comparator 6 in ABWC (see chapter 19.2) see bit 0
Bit 7	**ABWC7_EN:** Enable comparator 7 in ABWC (see chapter 19.2) see bit 0
Bit 8	**ABWC8_EN:** Enable comparator 8 in ABWC (see chapter 19.2) see bit 0
Bit 9	**ABWC9_EN:** Enable comparator 9 in ABWC (see chapter 19.2) see bit 0
Bit 10	**ABWC10_EN:** Enable comparator 10 in ABWC (see chapter 19.2) see bit 0
Bit 11	**ABWC11_EN:** Enable comparator 11 in ABWC (see chapter 19.2) see bit 0
Bit 12	**TBWC0_EN:** Enable comparator 0 in TBWC (see chapter 19.2) 0 = TBWC comparator 0 is disabled 1 = TBWC comparator 0 is enabled
Bit 13	**TBWC1_EN:** Enable comparator 1 in TBWC (see chapter 19.2) see bit 12
Bit 14	**TBWC2_EN:** Enable comparator 2 in TBWC (see chapter 19.2) see bit 12
Bit 15	**TBWC3_EN:** Enable comparator 3 in TBWC (see chapter 19.2) see bit 12
Bit 16	**TBWC4_EN:** Enable comparator 4 in TBWC (see chapter 19.2) see bit 12
Bit 17	**TBWC5_EN:** Enable comparator 5 in TBWC (see chapter 19.2) see bit 12
Bit 18	**TBWC6_EN:** Enable comparator 6 in TBWC (see chapter 19.2) see bit 12
Bit 19	**TBWC7_EN:** Enable comparator 7 in TBWC (see chapter 19.2) see bit 12
Bit 20	**TBWC8_EN:** Enable comparator 8 in TBWC (see chapter 19.2) see bit 12
Bit 21	**TBWC9_EN:** Enable comparator 9 in TBWC (see chapter 19.2) see bit 12
Bit 22	**TBWC10_EN:** Enable comparator 10 in TBWC (see chapter 19.2) see bit 12
Bit 23	**TBWC11_EN:** Enable comparator 11 in TBWC (see chapter 19.2) see bit 12
Bit 31:24	**Reserved:** Reserved **Note**: Read as zero, should be written as zero

19.7.2 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 ATOM sub modules bits 0 and 1 (see chapter 19.2) 1 = an error was recognized on corresponding ATOM 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. see bit 0
Bit 2	**ABWC2:** error indication for ABWC2. See bit 0. see bit 0
Bit 3	**ABWC3:** error indication for ABWC3. See bit 0. see bit 0
Bit 4	**ABWC4:** error indication for ABWC4. See bit 0. see bit 0
Bit 5	**ABWC5:** error indication for ABWC5. See bit 0. see bit 0
Bit 6	**ABWC6:** error indication for ABWC6. See bit 0. see bit 0
Bit 7	**ABWC7:** error indication for ABWC7. See bit 0. see bit 0
Bit 8	**ABWC8:** error indication for ABWC8. See bit 0. see bit 0
Bit 9	**ABWC9:** error indication for ABWC9. See bit 0. see bit 0
Bit 10	**ABWC10:** error indication for ABWC10. See bit 0. see bit 0
Bit 11	**ABWC11:** error indication for ABWC11. See bit 0. 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 19.2) 1 = an error was recognized on corresponding TOM 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. see bit 12
Bit 14	**TBWC2:** TOM sub modules outputs bitwise comparator 2 error indication. See bit 12. see bit 12
Bit 15	**TBWC3:** TOM sub modules outputs bitwise comparator 3 error indication. See bit 12. see bit 12
Bit 16	**TBWC4:** TOM sub modules outputs bitwise comparator 4 error indication. See bit 12. see bit 12
Bit 17	**TBWC5:** TOM sub modules outputs bitwise comparator 5 error indication. See bit 12. see bit 12
Bit 18	**TBWC6:** TOM sub modules outputs bitwise comparator 6 error indication. See bit 12. see bit 12
Bit 19	**TBWC7:** TOM sub modules outputs bitwise comparator 7 error indication. See bit 12. see bit 12
Bit 20	**TBWC8:** TOM sub modules outputs bitwise comparator 8 error indication. See bit 12. see bit 12
Bit 21	**TBWC9:** TOM sub modules outputs bitwise comparator 9 error indication. See bit 12. see bit 12
Bit 22	**TBWC10:** TOM sub modules outputs bitwise comparator 10 error indication. See bit 12. see bit 12
Bit 23	**TBWC11:** TOM sub modules outputs bitwise comparator 11 error indication. See bit 12. see bit 12
Bit 31:24	**Reserved:** reserved **Note**: Read as zero, should be written as zero

19.7.3 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									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. see bit 0
Bit 2	**ABWC2_EN_IRQ:** enable ABWC2 interrupt source for *CMP_IRQ* line. See bit 0. see bit 0
Bit 3	**ABWC3_EN_IRQ:** enable ABWC3 interrupt source for *CMP_IRQ* line. See bit 0. see bit 0
Bit 4	**ABWC4_EN_IRQ:** enable ABWC4 interrupt source for *CMP_IRQ* line. See bit 0. see bit 0
Bit 5	**ABWC5_EN_IRQ:** enable ABWC5 interrupt source for *CMP_IRQ* line. See bit 0. see bit 0
Bit 6	**ABWC6_EN_IRQ:** enable ABWC6 interrupt source for *CMP_IRQ* line. See bit 0. see bit 0
Bit 7	**ABWC7_EN_IRQ:** enable ABWC7 interrupt source for *CMP_IRQ* line. See bit 0. see bit 0
Bit 8	**ABWC8_EN_IRQ:** enable ABWC8 interrupt source for *CMP_IRQ* line. See bit 0. see bit 0
Bit 9	**ABWC9_EN_IRQ:** enable ABWC9 interrupt source for *CMP_IRQ* line. See bit 0. see bit 0
Bit 10	**ABWC10_EN_IRQ:** enable ABWC10 interrupt source for *CMP_IRQ* line. See bit 0. see bit 0
Bit 11	**ABWC11_EN_IRQ:** enable ABWC11 interrupt source for *CMP_IRQ* line. See bit 0.

	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. see bit 12
Bit 14	**TBWC2_EN_IRQ:** enable TBWC2 interrupt source for *CMP_IRQ* line. See bit 12. see bit 12
Bit 15	**TBWC3_EN_IRQ:** enable TBWC3 interrupt source for *CMP_IRQ* line. See bit 12. see bit 12
Bit 16	**TBWC4_EN_IRQ:** enable TBWC4 interrupt source for *CMP_IRQ* line. See bit 12. see bit 12
Bit 17	**TBWC5_EN_IRQ:** enable TBWC5 interrupt source for *CMP_IRQ* line. See bit 12. see bit 12
Bit 18	**TBWC6_EN_IRQ:** enable TBWC6 interrupt source for *CMP_IRQ* line. See bit 12. see bit 12
Bit 19	**TBWC7_EN_IRQ:** enable TBWC7 interrupt source for *CMP_IRQ* line. See bit 12. see bit 12
Bit 20	**TBWC8_EN_IRQ:** enable TBWC8 interrupt source for *CMP_IRQ* line. See bit 12. see bit 12
Bit 21	**TBWC9_EN_IRQ:** enable TBWC9 interrupt source for *CMP_IRQ* line. See bit 12. see bit 12
Bit 22	**TBWC10_EN_IRQ:** enable TBWC10 interrupt source for *CMP_IRQ* line. See bit 12. see bit 12
Bit 23	**TBWC11_EN_IRQ:** enable TBWC11 interrupt source for *CMP_IRQ* line. See bit 12. see bit 12
Bit 31:24	**Reserved:** reserved **Note**: Read as zero, should be written as zero

19.7.4 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. see bit 0
Bit 2	**TRG_ABWC2:** Trigger **ABWC2** bit in **CMP_IRQ_NOTIFY** register by software. See bit 0. see bit 0
Bit 3	**TRG_ABWC3:** Trigger **ABWC3** bit in **CMP_IRQ_NOTIFY** register by software. See bit 0. see bit 0
Bit 4	**TRG_ABWC4:** Trigger **ABWC4** bit in **CMP_IRQ_NOTIFY** register by software. See bit 0. see bit 0
Bit 5	**TRG_ABWC5:** Trigger **ABWC5** bit in **CMP_IRQ_NOTIFY** register by software. See bit 0. see bit 0
Bit 6	**TRG_ABWC6:** Trigger **ABWC6** bit in **CMP_IRQ_NOTIFY** register by software. See bit 0. see bit 0
Bit 7	**TRG_ABWC7:** Trigger **ABWC7** bit in **CMP_IRQ_NOTIFY** register by software. See bit 0. see bit 0
Bit 8	**TRG_ABWC8:** Trigger **ABWC8** bit in **CMP_IRQ_NOTIFY** register by software. See bit 0. see bit 0
Bit 9	**TRG_ABWC9:** Trigger **ABWC9** bit in **CMP_IRQ_NOTIFY** register by software. See bit 0. see bit 0
Bit 10	**TRG_ABWC10:** Trigger **ABWC10** bit in **CMP_IRQ_NOTIFY** register by software. See bit 0.

	see bit 0
Bit 11	**TRG_ABWC11:** Trigger **ABWC11** bit in **CMP_IRQ_NOTIFY** register by software. See bit 0. 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. see bit 12
Bit 14	**TRG_TBWC2:** Trigger **TBWC2** bit in **CMP_IRQ_NOTIFY** register by software. See bit 12. see bit 12
Bit 15	**TRG_TBWC3:** Trigger **TBWC3** bit in **CMP_IRQ_NOTIFY** register by software. See bit 12. see bit 12
Bit 16	**TRG_TBWC4:** Trigger **TBWC4** bit in **CMP_IRQ_NOTIFY** register by software. See bit 12. see bit 12
Bit 17	**TRG_TBWC5:** Trigger **TBWC5** bit in **CMP_IRQ_NOTIFY** register by software. See bit 12. see bit 12
Bit 18	**TRG_TBWC6:** Trigger **TBWC6** bit in **CMP_IRQ_NOTIFY** register by software. See bit 12. see bit 12
Bit 19	**TRG_TBWC7:** Trigger **TBWC7** bit in **CMP_IRQ_NOTIFY** register by software. See bit 12. see bit 12
Bit 20	**TRG_TBWC8:** Trigger **TBWC8** bit in **CMP_IRQ_NOTIFY** register by software. See bit 12. see bit 12
Bit 21	**TRG_TBWC9:** Trigger **TBWC9** bit in **CMP_IRQ_NOTIFY** register by software. See bit 12. see bit 12
Bit 22	**TRG_TBWC10:** Trigger **TBWC10** bit in **CMP_IRQ_NOTIFY** register by software. See bit 12. see bit 12
Bit 23	**TRG_TBWC11:** Trigger **TBWC11** bit in **CMP_IRQ_NOTIFY** register by software. See bit 12. see bit 12
Bit 31:24	**Reserved:** reserved **Note:** Read as zero, should be written as zero

19.7.5 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 00 = Level mode 01 = Pulse mode 10 = Pulse-Notify mode 11 = 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

19.7.6 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_EIRQ	TBWC10_EN_EIRQ	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_EIRQ	ABWC10_EN_EIRQ	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									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.

	see bit 0
Bit 2	**ABWC2_EN_EIRQ:** enable ABWC2 interrupt source for *CMP_EIRQ* line. See bit 0. see bit 0
Bit 3	**ABWC3_EN_EIRQ:** enable ABWC3 interrupt source for *CMP_EIRQ* line. See bit 0. see bit 0
Bit 4	**ABWC4_EN_EIRQ:** enable ABWC4 interrupt source for *CMP_EIRQ* line. See bit 0. see bit 0
Bit 5	**ABWC5_EN_EIRQ:** enable ABWC5 interrupt source for *CMP_EIRQ* line. See bit 0. see bit 0
Bit 6	**ABWC6_EN_EIRQ:** enable ABWC6 interrupt source for *CMP_EIRQ* line. See bit 0. see bit 0
Bit 7	**ABWC7_EN_EIRQ:** enable ABWC7 interrupt source for *CMP_EIRQ* line. See bit 0. see bit 0
Bit 8	**ABWC8_EN_EIRQ:** enable ABWC8 interrupt source for *CMP_EIRQ* line. See bit 0. see bit 0
Bit 9	**ABWC9_EN_EIRQ:** enable ABWC9 interrupt source for *CMP_EIRQ* line. See bit 0. see bit 0
Bit 10	**ABWC10_EN_EIRQ:** enable ABWC10 interrupt source for *CMP_EIRQ* line. See bit 0. see bit 0
Bit 11	**ABWC11_EN_EIRQ:** enable ABWC11 interrupt source for *CMP_EIRQ* line. See bit 0. 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. see bit 12
Bit 14	**TBWC2_EN_EIRQ:** enable TBWC2 interrupt source for *CMP_EIRQ* line. See bit 12. see bit 12
Bit 15	**TBWC3_EN_EIRQ:** enable TBWC3 interrupt source for *CMP_EIRQ* line. See bit 12. see bit 12
Bit 16	**TBWC4_EN_EIRQ:** enable TBWC4 interrupt source for *CMP_EIRQ* line. See bit 12. see bit 12

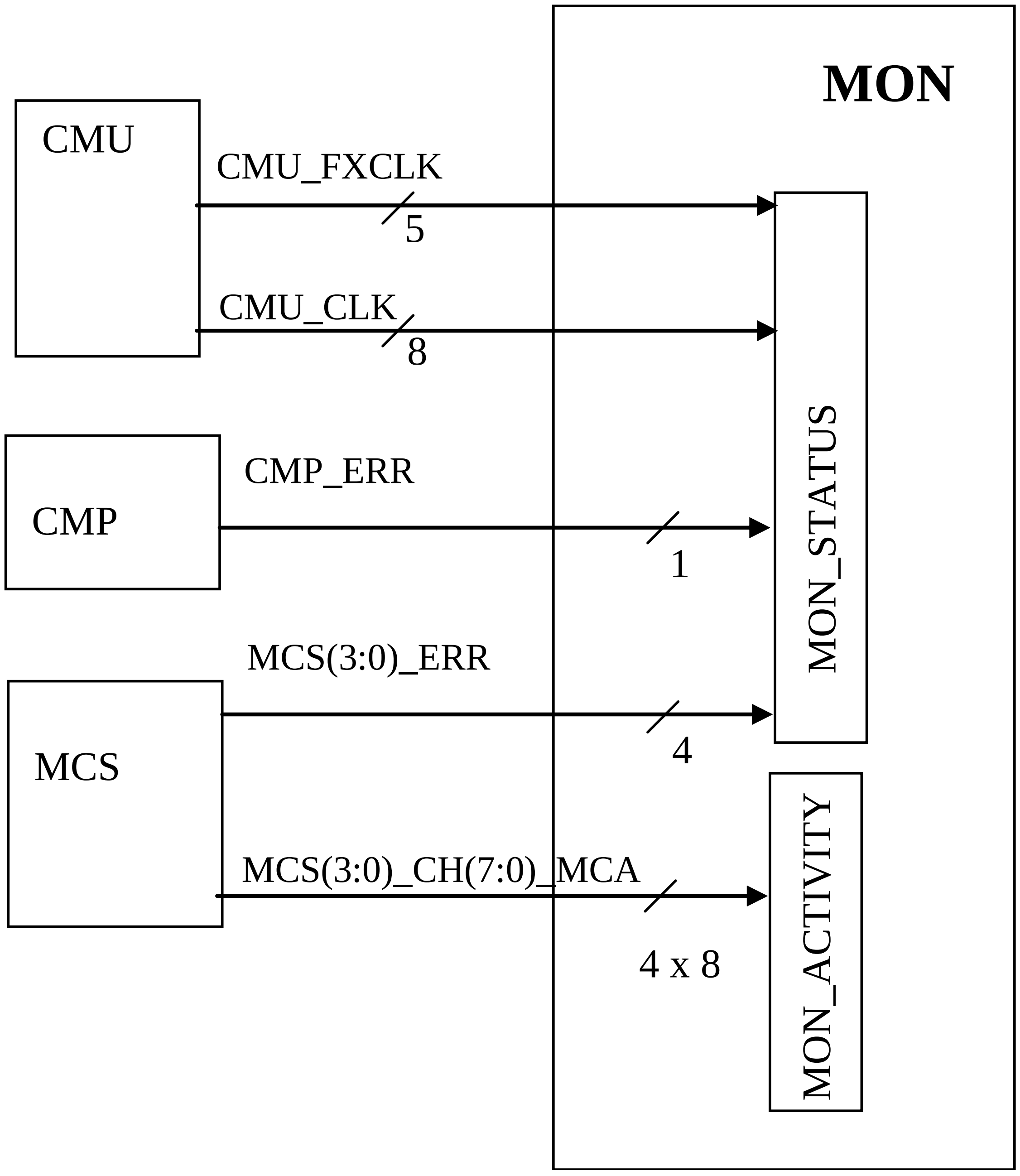
Bit 17	**TBWC5_EN_EIRQ:** enable TBWC5 interrupt source for *CMP_EIRQ* line. See bit 12. see bit 12
Bit 18	**TBWC6_EN_EIRQ:** enable TBWC6 interrupt source for *CMP_EIRQ* line. See bit 12. see bit 12
Bit 19	**TBWC7_EN_EIRQ:** enable TBWC7 interrupt source for *CMP_EIRQ* line. See bit 12. see bit 12
Bit 20	**TBWC8_EN_EIRQ:** enable TBWC8 interrupt source for *CMP_EIRQ* line. See bit 12. see bit 12
Bit 21	**TBWC9_EN_EIRQ:** enable TBWC9 interrupt source for *CMP_EIRQ* line. See bit 12. see bit 12
Bit 22	**TBWC10_EN_EIRQ:** enable TBWC10 interrupt source for *CMP_EIRQ* line. See bit 12. see bit 12
Bit 23	**TBWC11_EN_EIRQ:** enable TBWC11 interrupt source for *CMP_EIRQ* line. See bit 12. see bit 12
Bit 31:24	**Reserved:** reserved **Note**: Read as zero, should be written as zero

20 Monitor Unit (MON)

20.1 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.

20.1.1 MON Block Diagram



20.1.2 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.

20.2 Clock Monitoring

The monitor unit has a connection to each of the 8 clocks *CMU_CLK[x]* (x=0..7), 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.

20.3 CMP error Monitoring

The signal CMP_ERR is to be received directly from module CMP and is set if an error occurred.

20.4 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, which are re-read 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 an summarized error signal *MCS[z]_ERR* (z=0..3).

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 MCA_i_x (x=0..7) vector always for each MCS 8 bits for

each channels are combined. The activity signals are stored in the **MON_ACTIVITY_0** register for MCS0 to MCS3 and in the **MON_ACTIVITY_1** register for MCS4 to MCS6. 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.

20.5 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[z]_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[z]_ERR* is provided.

When the check is performed, an activity signal MCA_x (x=0..31) is provided for each channel x for each MCS[i](i=0..6) 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[z]_ERR is provided for the MON unit.
Figure 20.1.1 shows the block diagram of the Monitor Unit.

20.6 MON Interrupt Signals

The MON submodule has no interrupt signals.

20.7 MON Registers Overview

Following configuration registers are considered in MON sub module

Register Name	Description	Details in Section
MON_STATUS	Monitor Status register	20.8.1
MON_ACTIVITY_0	Monitor activity register 0	20.8.2
MON_ACTIVITY_1	Monitor activity register 1	20.8.3

20.8 MON Configuration Registers Description

20.8.1 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					MCS6_ERR	MCS5_ERR	MCS4_ERR	MCS3_ERR	MCS2_ERR	MCS1_ERR	MCS0_ERR	Reserved			CMP_ERR	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			RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw	RCw
Initial Value	0x00					0	0	0	0	0	0	0	0x00			0	0x0			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 high low slope is detected at the considered clock
Bit 15:13	**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 31:27	**Reserved:** Reserved bits

Note: Read as zero should be written as zero

Note: Bits16 and 20 to 26 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.

20.8.2 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.

20.8.3 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	Reserved								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	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	**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.

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.

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

21 Appendix A

21.1 Register Bit Attributes

Below the bit name in a register table, the attributes "Access Mode" and "Reset Value" of each bit are described with the following syntax:

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)

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 "00..00" and "11..11".

21.2 Register Reset Value

Reset Value	Description
0	logic value is 0 after reset
1	logic value is 1 after reset

21.3 ARU Write Address Overview

The ARU write address map is specified in Appendix B [1].

21.4 GTM Configuration Registers Address Map

The addresses of the implemented submodules are specified in Appendix B [1].
The start and end addresses 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].

21.5 GTM Application Contraints

The constraints put on applications by GTM implementation are specified in Appendix B [1].

21.6 GTM Internal functional dependencies

The Figure 1.6.1 shows the functional dependencies of GTM.

Signal paths between GTM-IP modules

from: \ to:	aru	atom	brc	cmp	cmu	dpll	icm	map	mcs	mon	psm	spe	tbu	tim	tom
aru		53 bit data	53 bit data			53 bit data	IRQ signals		53 bit data		53 bit data				
atom	53 bit data						IRQ signals							ATOM[i]_OUT	
brc	53 bit data						IRQ signals								
cmp							IRQ signals			CMP_ERR					
cmu		CMU_CLK[x]				CMU_CLK0							CMU_CLK[x]	CMU_CLK[x]	CMU_FXCLK[x]
dpll	53 bit data				SUB_INC1, SUB_INC2, ..		IRQ signals						SUB_INC1c, SUB_INC2x, ..		
icm															
map						TRIGGER, STATE, T_DIR, S_DIR, ..									
mcs	53 bit data						IRQ signals			MCS[i]_ERR, MCS[i]_CH[x]_MCA					
mon															
psm	53 bit data						IRQ signals								
spe							IRQ signals	SPE[i]_OUT, SPE[i]_NIPD, ..							SPE[i]_OUT, SPE[i]_NIPD, ..
tbu		TBU_TS[x]				TBU_TS[x], LOW_RES, TS_CLK			TBU_TS[x]					TBU_TS[x]	TBU_TS[x]
tim	53 bit data						IRQ signals	TIM0_CH0(48:0)				TIM[i]_CH[x](48)			
tom							IRQ signals					TOM[i]_CH[x]_SOUR, TOM[i]_CH0_TRIG_CCU0, TOM[i]_CH0_TRIG_CCU1, ...		TOM[i]_OUT	

22 Revision History

Issue	Date	Remark
0.6.1	17.10.2008	Initial version
0.6.2	08.12.2008	Specification after first review
0.6.3	11.08.2009	Specification after second review
0.6.4	06.11.2009	Specification after third review
0.6.5	17.12.2009	Specification after fourth review
0.6.6	03.03.2010	Specification completed (BLDC feature added)
1.0	23.03.2010	Specification completed (text & picture tuning)
1.0.1	25.03.2010	Specification with new interrupt concepts, MCS-RAM organisation and feedback from GTM-RM evaluation
1.0.2	26.03.2010	Specification base lined
1.0.3	30.03.2010	Text & picture tuning done, last check before v1.1
1.1	30.03.2010	Specification closed
1.1.1	27.05.2010	1. update for RM 1.1.2
1.1.1.2	07.06.2010	2. update for RM 1.1.2
1.1.1.3	14.06.2010	3. update for RM 1.1.2 (DPLL chapter 16 again)
1.1.1.4	16.06.2010	4. update for RM 1.1.2 (Minor: TOM,SPE; much Minor: ATOM,DPLL)
1.1.2	18.06.2010	RM 1.1.2 (Minor: DPLL, TIM; much Minor: ATOM, MCS
1.1.3	29.06.2010	RM 1.1.3 (Minor: CMP, ATOM, TIM, TBU, ARCH; much Minor: DPLL, ICM, MON) MCS: 13.2.2: Extended Memory Transfer instructions from 24 to 32 bit transfers 13.2.3: Modified ARU instructions 13.2.4: Removed rotate instructions 13.2.5: Modified comparison instructions in order to support singed arithmetic 13.2.7: Replaced and enhanced instruction WTRG and WTC by instruction WURM
1.1.4	12.07.2010	RM 1.1.4 No Spec update
1.1.5	28.07.2010	RM 1.1.5 BRC : renaming CTRL -> ADDR DPLL: 16.5.10ff: RAM1a range, 16.6.2.4ff: QDT_T resolution, 16.6.3.4ff: QDT_S resolution,

		16.6.4.4: QDT_T resolution, 16.6.5.1ff: QDT_S resolution, 16.7.6.2: blocking read of actions, 16.7.7.3: TBU_TS0_HRS, 16.8.6.2: setting LOCK bit when no gap, 16.8.6.7: steps 5 and 25 (missing pulses), 16.10: RAM 1a size, 16.11.6: PSE,LOCK,ERR Status bits, 16.12: RAM 1a protection, 16.13.18ff RCDT_Tx resolution AEI : 2.2.1: added registers which will issue an illegal module access on a protected write 2.8.5-2.8.7 added irq notify bit to signal an unsupported byte address access AFD: 6.1: write/read on full/empty fifo channel clarified TIM: 10.3.1;10.4.1 minor clarifications TOM: 11.8.4: TBU_SEL="10" only valid if TBU_CH2 exist ATOM: 12.3.2: TB12_SEL only valid if TBU_CH2 exist 12.3.4: SOMS mode: behaviour for ARU_EN=0 and OSM=1 (one shot mode) specified MCS: 13.1: architecture change 13.1.1: architecture figure change 13.2.4.5: no carry function 13.4.2: write protected register bit
1.1.6	04.08.2010	RM 1.1.6 Intro : 1.2: table of submodule Arch : 2.7 + programming conventions TOM : indices correction; clarify channel architecture figures MCS : 13.2ff: new abbreviation for commands (POP,

		ARD, AWR, SHR, SHL, CALL, RET); modification on register STA, MCS[i]_CH[x]_CTRL, MCS[i]_CH[x]_IRQ_NOTIFY, MCS[i]_CTRL DPLL : 16.11.21, 16.15.3: modification on register DPLL_ID_PMTR_x, ADT_T[i]
1.2	30.09.2010	RM 1.2 ARCH : 2.2.1 Updated list of registers that drive value “illegal module access” 2.5 Fixed bugs in IRQ Specification. BRC : 4.2 Clarified register bit settings in case of Trashbin functionality. 4.5.2 Clarified register bit settings in case of Trashbin functionality. FIFO: 5.5.12,13 Fixed bad naming of register labels. TIM : 10.4.2.5 Clarified modified behaviour of ECNT in mode TBCM. 10.8.1,2 Clarified behaviour if bad TIM modes are selected. TOM : 11.3.6 Clarified one shot mode in case of writing to CN0 while CN0 is counting. ATOM : 12.2.2 Clarified ATOM channel startup behaviour when ARU_EN is set and channel is enabled. 12.3.1.1, 12.3.2.3, 12.3.3.5, 12.3.4.1 Not used bits can be written in any mode, but should be written as zero when not used 12.3.4 Clarified interrupt behaviour in SOMS mode: in SOMS mode only CCU1TC IRQ possible 12.6.2 Corrected some bit manipulation properties for bits OSM, TRIGOUT, CLK_SRC/CLK_SRC_SR MCS : 13.1.1ff Fixed declaration of maximum address range and clarified priorities of register accesses. MCFG :

		14.1.1ff Documented additional memory layout parameters for the case that two large RAMs are connected to MCS. DPLL : 16.4.2 LOW_RES options 16.7.1 VTN 16.8.3ff NMB_T/S: MPVAL1/2 added 16.8.5ff PSTC/PSSC = 0 for FTD/FSD=0 16.8.6.7ff NMB_T/S calculation and use 16.11.1 Note 3): restriction for resolution choices 16.11.2 SWR and DEN 16.11.6 CRO, PSE, TOR, SOR, LOCK1/2 16.11.8/9 OSW and AOSV_2 16.11.28/29 old ADD_IN value used 16.13 no write protection of available RAM bits 16.13.9 THVAL for LOW_RES and !TSO_HRT 16.13.16ff all reciprocal values: SHL 32 16.14ff resolution of reciprocal values MON : 20.1.1ff Activity checker removed
1.2.1	21.10.2010	RM 1.2.1 ARCH : 2.2.1 Updated list of registers that drive value “illegal module access” 2.5.4 Changed IRQ behaviour in single pulse mode on simultaneous clear and interrupt event. 2.9.2 Protected write information ARU : 3.6.1 Clarified usage of ARU_ACCESS register. BRC : 4.5.1 ARU read address only writeable if channel disabled F2A : 7.4.2 ARU read address only writeable if channel disabled CMU : 8.3, 8.5, 8.6 : Removed confusing reset description. ATOM : 12.3 Clarified CN0 behaviour at channel enable 12.3.2ff SOMC mode description restructured 12.3.4ff SOMS mode description restructured 12.6.4 ARU read address only writeable if

		channel disabled MCS : 13.2.1 Specified behaviour in the case of bad opcode decoding. 13.2.3 Added two non blocking ARU read instructions (NARD/NARDI) 13.2.4 Clarified behaviour on flags CY and V in arithmetic instructions. 13.4.5 Added write protection to register MCS[i]_CH[x]_R6. DPLL : 16.6.2.1 higher reciprocal resolution TRIGGER 16.6.3.1 higher reciprocal resolution STATE 16.8.3.6 QDT is zero at the beginning 16.8.6.7 Store INC_CNTx when a valid event occurs and use it as MPx in step 5 or 25 16.11.1 condition for pulse number per increment 16.11.6 refined definitions of CRO, RCS, RCT, MT, MS, FTD, FSD and LOCKx 16.11.21 DPLL_ID_PMTR_x is now RPW 16.12.3 Bits 24:31 are read only 16.13.7ff Bits 24:31 are read only 16.13.12ff suppress update for unexpected missing TRIGGER or STATE respectively 16.13.16ff calculation of reciprocal values from predicted durations do not influence RCO 16.13.32ff Bits 24:31 are read only 16.13.43ff minor changes in formulations 16.13.47ff Bits 24:31 are read only 16.14.3 Bits 24:31 are read only 16.15.3 Bits 24:31 are read only
1.2.2	02.11.2010	RM 1.2.2 ARCH : 2.2.1 Updated list of registers that drive value “illegal module access” TBU : 9.4.3ff Address offset corrected ATOM : 12.3.2ff SOMC mode description restructured DPLL : 16.11.17 refined definitions of TINI and TAXI
1.3	15.12.2010	RM 1.3 ARCH : 2.2.1 Updated list of registers that may drive an aei_status signal with value “10” 2.5.4 Added note in single pulse mode

		2.6ff Extended description of software debugger support (e.g. added lists of registers with special behaviour in the case of debugger access). 2.9.1 Revision ID updated 2.9.3 Reset force protection includes SW RAM reset 2.9.9-11 Moved register description of AEI_bridge from module integration guide to this document. Added control and status information for diagnosis of the bridge status. ARU : 3.3.2 Clarified usage of debug mechanism. 3.6.1 Extended functionality of ARU_ACCESS register. BRC : 4.5.1 Note added TBU : 9.4.2 Second note added TOM : 11.2.2 Note to trigger signal added 11.8.2 Meaning of bits in ENDIS_CTRL changed 11.8.5 Meaning of bits in OUTEN_CTRL changed ATOM : 12.2.2 Clarified ATOM channel start-up behaviour in case of ARU_EN=1. 12.3.2.1 Added note that less/equal compare only applies for comparisons against TBU_TS1/2. 12.3.2.2.1 Added ACBI=00111 behaviour in table. 12.3.2.3.3 Clarified WR_REQ and WRF bit behaviour for ATOM SOMC late update mechanism. 12.3.2.4 Fixed ACB10 bit description. 12.3.3.5 Clarified CLK_SRC_SR register description. 12.3.4 Global ATOM SOMS mode behaviour update. 12.3.4.6 Update SL and CLK_SRC_SR register description. 12.6.2 Update SL and CLK_SRC_SR register description. MCS : 13.4.11 Added feature to reset connected RAMs by software. MAP :

		15.1.1 Fixed minor specification issues. 15.4.1 Completed SSL bit field description. DPLL : 16.5.10.2 Influence of SYSF to entry number 16.6.2.5 New Calculation for FULL_SCALE 16.6.2.7 New Meaning of CTO(N) 16.6.3.5 New Calculation for FULL_SCALE 16.6.3.7 New Meaning of CSO(N) 16.6.4.2 New Calculation for FULL_SCALE 16.6.5.2 New Calculation for FULL_SCALE 16.7ff Handling of actions in the past, considering of VTN, VSN and SYSF for calculations, use of RDT_T_FS1 and RDT_S_FS1 16.8.6.7 Use of NMB_T_TAR and NMB_S_TAR 16.10 New registers for RAM initialization, NMB_T_TAR and NMB_S_TAR 16.11.ff Notes to specify write abilities added. Influence of SYSF, new FST and FSS bits in NUTC/NUSC registers, new DCGI bit in DPLL_IRQ_NOTIFY register; new interrupt line connections; DPLL_RAM_INI register 16.12ff Notes to specify write abilities added. 16.13.4 NMB_T_TAR explained 16.13.8 NMB_S_TAR explained 16.13.16 Sign extension 16.13.17 Sign extension 16.14.1ff SYSF influence explained ICM : 18.5.2 Changed DPLL Interrupt ordering.
1.4	29.03.2011	RM 1.4 ARCH : 2.1.2 Restricted TOM/ATOM trigger lines to a certain length 2.3.1 Changed number of ARU data destinations 2.3.2 Introduced formula to determine the device dependent ARU round trip time 2.6.1 Specified register behaviour for GTM_halt more precisely 2.9.1 Revision ID updated 2.9.7 Note about protection included ARU : 3.6.1 Changed register bit field mode

		3.6.11	Register bit 0 name corrected
		3.6.12	Note about protection included
		BRC :	
		1.	Reset value corrected
		1.	Note about protection included
		FIFO :	
		5.1	Fixed specification error, FIFO RAM not initialized with zeros after reset
		5.5.12	Note cleared automatically included
		TBU :	
		9.1	Specified behaviour of TBU_UPx signals more precisely
		TIM :	
		10.ff	Clarified indices descriptions of registers and bit fields
		TOM :	
		11.2.2	Added note for signed compare of ACT_TB
		11.8.4	Changed naming and description of bit field ACT_TB
		11.8.27	Note about write protection included
		ATOM :	
		12.3.2.3.3	Specified behaviour of WR_REQ bit more precisely
		12.3.4.1ff	Specified ATOM SOMS mode more precisely
		12.6.12	Note about write protection included
		MCS :	
		13.1.1	Added hint for MCS RAM addresses and changed MCS block diagram
		13.3.3	Note cleared by CPU included
		13.4.6	Reset value corrected
		13.4.9	Note protection included
		13.4.11	Reset value corrected
			Note reset values dependencies included
		14.4.14	Note cleared by MCS channel included
		DPLL :	
		16.5.10.1	typo corrected: APS pointer
		16.7	number of actions calculated per

		increment in worst case and typical 16.7.1ff calculation of fractional part changed; decision of calculation case is dependent on the relation between q and m 16.7.7ff chapter ordering swapped with 16.7.8 16.8.4.2ff more detailed description 16.8.6.3ff direction change description changed 16.8.6.7 changed schedule of RAM update, PSTC and PSSC first value, SGE effect 16.10 introduced addresses for shadow registers, changed RAM addresses for calculated target pulse numbers 16.11.1 description of change between normal and emergency mode 16.11.2 write protection explained, more detailed description of SYN_NT/SYN_NS 16.11.12 explained differences for direction change in normal mode (SMC=0) and for SMC=1 16.11.13 explained direction change process 16.11.19 Note protection and cleared automatically included 16.11.24 more detailed explanation of address pointer value extension 16.11.25 more detailed explanation of address pointer value extension 16.11.33 New Register 16.11.34 New Register 16.11.35 New Register 16.11.36 New Register 16.12.2 explanation DLA value for low resolution 16.13.4 Removed NMB_T_TAR register 16.13.8 Removed NMB_S_TAR register 16.13.10 explanation TOV value for low resolution 16.13.11 explanation TOV_S value for low resolution 16.13.16 New RAM storage address 16.13.17 New RAM storage address 16.13.18 New RAM storage address 16.13.19 New RAM storage address 16.13.45 mode limited 16.13.46 mode limited 16.15.1ff start address of RAM2 changed SPE : 1. Note about protection included ICM :

		18.5.1ff Register bit mode changed to R CMP : 18.5.1ff Register bit mode changed to R 19.7.4 Note about protection included Appendix : 21.3 Changed reference to address map
1.4.1	31.05.2011	RM 1.4.1 ARCH : 2.1 Description of GTM device 103 declared to be one possible example of device configurations 2.1.2 GTM_IP_103 replaced by GTM_IP 2.2.1 updated list of register which return AEI status of illegal write access 2.3.3 (wrong) ARU round trip time calculation removed 2.9.1 GTM_REV reset value now specified in Appendix B ATOM : 12.6.6 Note about protection included 12.6.8 Note about protection included CMU: 8.1.1 Added clock source MUX to FXU. 8.8.ff Added clock selection bit to register CMU_CLK_5_CTRL. MCS : 13.1.1 Updated figure in Architecture section. 13.3.3 RAM ECC Error disables MCS channel and raises ERR bit. 13.3.5ff Bit9 naming corrected 13.4.12ff Bit9 naming corrected MCFG : 14.1 Reformulated MCFG Spec to be more generic and device independent. 14.2.1 Reformulated MCFG_CTRL register to be more generic and device independent. DPLL : 16.5.10.2 explanation of RAM init procedure 16.6.2 explanation of the use of filter values

		16.7.1ff precise case conditions 16.7.5.6ff precise case conditions 16.7.8.2ff precise case conditions 16.8.5 correct PSTC/PSSC calculation 16.11.2 precise SMC bit meaning 16.11.23 correct copy/paste error for INC_CNT2 ICM : 18.2.3ff number of submodules (TIM, MCS, TOM and ATOM) described generic with “[i]” 18.3 Interrupt signals for submodules TIM, MCS, TOM and ATOM described generic with “[i]” 18.4 Add missing ICM registers for device 4 18.5.1ff Add missing ICM registers for device 4 App A : 21.2ff ARU write address overview moved to Appendix B (separate document file)
1.4.2	29.07.2011	RM 1.4.2 ARCH : 2.2.1 updated list of register which return AEI status of illegal write access 2.3.1ff Table of ARU sources corrected CMU: 8.5 Added clock selection bit to description. 8.8.4ff Added clock selection bit to register CMU_CLK_5_CTRL. TIM: 10.7 Corrected link to sub section in overview table. TOM: 11.8.7 Description of bit field FUPD_CTRL ATOM : 12.3.2.2.1 Condition of bit SLA added 12.3.2.3 Condition of bit SLA added 12.3.2.4 Description of new mode bit SLA 12.3.3.5 Bit 16 is not used instead of reserved 12.3.4.9 Bit 16 is not used instead of reserved 12.6.2 Description of new mode bit SLA added MCS :

		13.1.1 Added ECC Error handling to specification. 13.3ff Clarified some minor parts of the description. 13.3.3 Clarified the behaviour of the ERR bit in STA register. DPLL : 16.6.2.7 Changed CTO/CTON behaviour in case of negative CDT_TX(_nom) values 16.6.3.7 Changed CSO/CSON behaviour in case of negative CDT_SX(_nom) values 16.6.4.4 Specified setting of CDT_TX(_nom) in case of negative values 16.6.5.4 Specified setting of CDT_SX(_nom) in case of negative values 16.7.6.1 Changed calculation of TSAC when action in far future (normal mode) 16.7.8.1 Changed calculation of TSAC when action in far future (emergency mode) 16.8.3.6 Removed deletion of FF's in case of new trigger/state event 16.8.6.3ff Specify that nmb_t/s_tar_new values replaces nmb_t/s_tar values in case of direction change 16.10 reconfiguration of some table elements for better reading 16.11.6 Introduced RAM2_ERR status flag, active when access to not configured memory space happened ICM : 18.5.1 Register description for max. device configuration 18.5.4 Register description for max. device configuration MON : 20.4 MON_ACTIVITY_1 register introduced 20.7 MON_ACTIVITY_1 register introduced 20.8.3 MON_ACTIVITY_1 register introduced
1.4.3	15.08.2011	RM 1.4.3
1.4.4	23.09.2011	RM 1.4.4 ARCH : 2.1.2 trigger chain length description modified 2.3.1 Explain ARU round trip time more precise

		2.5 ff Typos 2.6.1 Typo 2.9.9 Register structure updated; introduced Bit field SYNC_INPUT_REG TBU: 9.1 Removed TBU_TS0 indexing (x) TIM: 10.1 Removed specific channel number of eight Removed TBU_TS0 indexing (z) 10.2.2 Corrected sentence 10.2.2.2.1 Corrected naming of filter mode 10.2.2.3.1 Corrected naming of filter mode 10.3.1 Corrected timeout calculation formula 10.4.2.5 Removed specific channel number of eight 10.6 Removed specific channel number of eight 10.7 Removed specific channel number of eight 10.8.1-13 Removed specific channel number of eight 10.8.14 Inserted note on register bit width dependency ATOM : 12.3.1 SL bit comment added 12.3.2.3 in tables link to chapter corrected 12.3.2.4 bit ACB10 description 12.6.12 bit 1 TRG CCU1TC description DPLL : 16.5.3 declaration of limits for SUB_INC1,2 clocks 16.8.3 after eq. 16.25b, 16.26b: explanation of SUB_INC1,2 frequency doubling possibilities 16.8.6.3 explanation of “make calculation” steps 3, after 5 : nmb_t replaced by nmb_t_tar in emergency step 2: explanation added at the end of chapter: some corrections 16.8.6.4 some corrections like 16.8.3.8 at end 16.10 explanation of TBU_TS_x and INC_CNT1,2 RAM Region 2 end

		depends on device chosen 16.11.1 Bits 24:11 explained more detailed 16.11.3 Bits 3,4: explained more detailed, correction 16.11.24,25 heading correction, Bits 5:0 explanation ICM : 18.3 in table GTM_DPLL_xxx description corrected 18.5.1 bit 24 and 25 description corrected 18.5.12 description corrected MON : 20.8.1 extended MON_STATUS register for Device 4 20.8.3 MON_ACTIVITY_1 register at address 0x08
1.5.0	02.12.2011	RM 1.5.0 ARCH : 2.1.3 new functionality GTM-IP signal multiplexing added 2.2.1 CMU_CLK_CTRL, CMU_FXCLK_CTRL, TIM[i]_CH[x]_GPR1 included DPLL_PSA, DPLL_DLA, DPLL_NA, DPLL_DTA removed 2.5 ff new error interrupt 2.8 new address map due to new register 2.9.1 bit field YEAR replaced by STEP 2.9.9 NOTE text improved 2.9.12 new error interrupt 2.9.13 new register GTM_TIM[i]_AUX_IN_SRC BRC: 4.4 new error interrupt 4.5.7 new error interrupt FIFO: 5.4 new error interrupt 5.5.14 new error interrupt CMU: 8.5 selection of clocks for fxclk 8.7 new CMU_FXCLK_CTRL 8.8.1 clarifying disable/enable functionality 8.8.5 no exception for CMU_CLK5_CTRL

		8.8.9 selection of clocks for fxclk TIM: 10.1.1 Block diagram updated due to new functionality INPSRC, EXTCAPSRC 10.1.2 new functionality INPSRC selection added 10.1.3 new functionality EXTCAPSRC selection added 10.3 TDU Block diagram and description updated due to new functionality of edge selection 10.4.1 TIM channel block diagram and description updated due to new functionality (ECNT increased to16 bit, ECNT can be captured to GPRx_REG, External_capture) 10.4.2ff New functionality External_capture, EGPRx_SEL added to existing TIM modes 10.4.2.6 New TIM mode TGPS added 10.7 Configuration registers updated due to new added functions 10.8.1-2 New bitfields added to support new functionality 10.8.5 Register removed , rearranged to TDUV, TDUC register 10.8.15-20 Register description for new functions added TOM: 11.4 TOM_CH2 special mode hint added 11.8.17 SL bit: NOTE added CLK_SRC_SR bits: description improved RST_CCU0 bit: note added SPEM bit: note added 11.8.18 SL bit: NOTE added CLK_SRC_SR bits: description improved 11.8.24 OL bit: note added ATOM : 12.3.1.1 SL bit: NOTE added 12.3.2.2.1ff table design corrected 12.3.2.4 SL bit: NOTE added TRIGOUT bit: added

		12.3.3.5 SL bit: NOTE added 12.3.4.9 SL bit: NOTE added 12.6.2 SL bit: NOTE added CLK_SRC_SR bits: NOTE added 12.6.3 SL bit: NOTE added ACBO bits: NOTE added MCS : 13.1.1 Clarified usage of write accesses to MCS registers in the case of multiple source. Usage of accessing common trigger register is clarified. 13.4.11 Added Error interrupt register MCS[i]_CH[x]_EIRQ_EN 13.4.13-14 Added Note for writing common trigger bits via AEI. MAP : 15.1.1 figure updated, description of new DIR selection added 15.4.1 TSEL bit: TIM0_IN6 feature added LSEL bit: new configuration bit DPLL : 16.6.2ff TRIGGER time stamp resolution for filter values considered 16.6.3ff STATE time stamp resolution for filter values considered 16.7ff up to 32 actions served for device 4 16.10 additional registers and address map changed 16.11.2 some control bits added 16.11.6 control register for 8 additional actions 16.11.7 8 control bits for 8 additional actions 16.11.20-21 additional EIRQ enable register 16.11.30-35 status register with changed address and omitting CTON/CSON up to 32 ID_PMTR register changed SHADOW register bits 16.12.1-4 extension to serve 32 actions 16.13.38-39 centralization for TRIGGER and STATE calculation 16.15ff additional register address range for

		changed and additional registers to serve up to 32 actions SPE : 17.2.2 figure updated 17.2.4 new chapter for SPE revolution detection 17.3 new interrupt SPE_RCMP 17.4 new register added 17.5.5-6 new register 17.5.7 additional IRQ 17.5.8 additional IRQ 17.5.9 additional IRQ 17.5.11 new register ICM : 18.3 new GTM Error Interrupt 18.4 new BRC, FIFO[i]_CH[x], TIM[i]_CH[x], MCS[i]_CH[x], SPE[i], CMP, DPLL, GTM Error Interrupt register 18.5.13-18 description of new Error Interrupt Register CMP : 19.5 new error interrupt 19.6 new error interrupt 19.7.6 description of new Error Interrupt Register MON : 20.5 Note concerning ARU roundtrip time measurement by MCS added APP-A : 1. 21.3 include remark for Appendix B
1.5.1	15.12.2011	RM 1.5.1 TIM: 10.1.3 EXTCAP_SRC: enhanced functionality description 10.4.1 new functionality: TIM input signal value available via ECNT[0] if channel is disabled 10.4.2ff EXT_CAP_EN: enhanced functionality description Added to each TIM channel mode detailed description of external capture

			behaviour TGPS mode: enhanced functionality description new functionality: in TGPS mode GPR1 operates as a shadow register for CNTS
		10.8.7	Register description updated due to added TGPS functionality
1.5.2	27.01.2012	RM 1.5.2	
		ARCH:	
		2.2.1	List of register and address ranges which return AEI status "10" corrected.
		ARU:	
		3.3.1	Note for bit 12 (RREQ) and bit 13 (WREQ) of register ARU_ACCESS added.
		TIM:	
		10.8.19	Note added to ECNT register (behaviour of ECNT if channel gets disabled)
		MCS:	
		14ff	layout option changed to layout configuration
		DPLL:	
		16.4ff	clear index 9
		16.4.2	no hex for width
		16.5.5	IP like formulation of clock frequency requirements
		16.5.6	Remove undeclared abbreviation "CDG"
		16.6.2	Reduction of CTON, CTO function to just CTO flag
		16.6.3	Reduction of CSON, CSO function to just CSO flag
		16.11.30	Clarification of SYS/SYT flags, because not influenced for TOR/SOR states.
		ICM:	
		18.5.14ff	correct naming IRQ to EIRQ
		CMP:	
		19.1.1	include CMP_EIRQ
		19.3	include CMP_EIRQ_EN
		19.5	include CMP_EIRQ

1.5.3	14.03.2012	RM 1.5.3 ARCH: 2.5 more precisely explanation of clearing notify register 2.5.2.2 more precisely explanation of clearing notify register TOM: 11.2.1 bit field name CLK_SRC_STAT replaced by CLK_SRC 11.8.1 bit field name CLK_SRC_STAT replaced by CLK_SRC ATOM: 12.6.1 bit field name CLK_SRC_STAT replaced by CLK_SRC MCS: 13.2.2.10 reference to ACB replaced by reference to MHB DPLL: 16.13.7 more precisely explanation of THMI 16.13.8 more precisely explanation of THMA 16.13.9 more precisely explanation of THVAL SPE: 17.5.6 note related to SPE[i]_RCMP interrupt added
1.5.4	06.07.2012	RM: 1.5.4 ARCH: 2.1 channel and configuration advice GTM: 2.1.3 new GTM_MX block in figure; update figure of GTM_MXy 2.3.3 round trip time advice 2.5 correct register name from (..CEI1 to ..CEI5) to (..CEI0 to ..CEI4) 2.5.4 exception advice FIFO: 5.1 Explanation of flushing the FIFO after write access to START_ADDR or

		END_ADDR register 5.2.2 Further explanation of ring buffer mode 5.5.2ff Note added for further explanation 5.5.4ff Note added for further explanation CMU: 8.3 more precisely explanation of using clocks TIM: 10.1.2 more precisely explanation of F_in(x) 10.4.2.5.1 removed wrong explanation that channels 1 to m-1 have to be disabled 10.4.2.6 corrected TGPS mode explanation (counting is done via selected CMU clock). TOM: 11.1 new indices definition 11.3.6 additional one shot mode handling hint ATOM: 12.3.3.4 additional one shot mode handling hint 12.6.1 UPEN_CTRL0 Note deleted DPLL: 16.10.5.1ff address relation and names in EXT register region and RAM2 precised 16.6.2ff signal names changed 16.7 operation times precised for new version 16.8.3.4 description of sub_inc generation conditions 16.11.2 relation of CMU_CLK0 to system clock 16.11.3-6,8 activity of the enable bits only for DEN=1, reset of SWON_x for DEN=0 SPE: 17.5.4 additional note to SPE_OUT_CTRL ICM: 18.2.5ff more precisely explanation of interrupt bundling
1.5.5	20.12.2012	GTM: 2.5 More detailed description of **IRQ_NOTIFY** clearing.

		FIFO: 5.5.3 end replaced by start 5.5.4 upper replaced by lower 5.5.6 EMPTY and FULL description corrected TIM: 10.2.3 more detailed information about filter reconfiguration 10.5 more precising of MAP interface 10.8.1ff more detailed information about reconfiguration and GPRx_SEL 10.8.6ff additional information about GPRx 10.8.20 additional information about input selection for channels TOM: 11.1 Including design variables 11.3ff behaviour on reset of CN0 by CCU0 compare 11.3.6.1ff figure and description of new one shot trigger feature added ATOM: 12.1 Including design variables 12.1.2ff remove channel mode overview description 12.3.1.1 bit 23:21 description adapted 12.3.2.3.1 additional information about compare strategy; figure and description above added 12.3.3ff description of 0% and 100% duty cycle precised 12.3.3.4.1 precising PWM output trigger 12.3.3.5 more detailed information about CLK_SRC_SR and RST_CCU0 in SOMP mode 12.3.4.9 bit 23:21 description adapted MCS: 13.4.1 Added missing register MCS[i]_CH[x]_EIRQ_EN 13.4.7 Added missing NOTE for MEM_ERR_IRQ bit. MCFG: 14.1ff Rewritten specification of MCFG module.

		DPLL: 16ff all equation numbering transformed from 16.xxx to DPLL-xxx 16.10 register addresses listed in Appendix B document No detailed PDTx addressing
1.5.5.0	31.01.2013	Header: Revision number with 4 integer parts New part “Revision Number Notice”
1.5.5.1	20.06.2013	GTM MDG1 Intro: 1.1 Additional information about interrupt services GTM: 2.2.1 Additional information about status ‘10’ 2.4.1 precising the number of generated clocks 2.5.1ff corrected signal name as same in figure 2.9.1 Additional information about bits and values 2.9.3ff Additional information about reserved bits 2.9.8ff modified reset value ARU: 3.6.13 modified reset value BRC: 4.5.3-5,7 precising information about DID 4.5.6 modified reset value FIFO: 5.5.1 precising information about WULOCK 5.5.13 modified reset value AFD: 6.1 replaced AFD[i]_CH[x]_BUFF_ACC by AFD[i]_CH[x]_BUF_ACC CMU: 8.3,6 precising the number of generated clocks 8.8.1 removed Note: Any disabling to **EN_ECLK[z]** will be reset internal

		counters for external clocks TIM: 10.1.1 replaced GPRz by GPR0 and GPR1 10.1.2 state bit VAL_x(1) and MODE_x(1) introduced in diagram and textual description 10.1.3.1 precising enable external capture functionality 10.3.1 replaced GPRz by GPR0 and GPR1; Extended description of ACB(2:0) bit behaviour in case of a timeout event. 10.4.1 replaced GPRz by GPR0 and GPR1 10.4.2 replaced GPRzOFL by GPROFL 10.4.2.1 Extended description of CNTS register update 10.4.2.3 replaced GPRzOFL by GPROFL 10.6 replaced GPRzOF by GPROFL; replaced GPRz by GPR0 and GPR1 10.8.2 replaced GPRz by GPR0 and GPR1 10.8.10-12 replaced GPRzOFL by GPROFL 10.8.13 modified reset value 10.8.15 Description of VAL_0 and MODE_0 enhanced 10.8.16 replaced GPRzOFL by GPROFL 10.8.20 bit width of EXT_CAP_SRC corrected; new values added TOM: 11.3.6.1 precising of one shot mode handling 11.3.7.1 precising of pulse count modulation mode 11.8.17 modified reset value 11.8.18 modified reset value 11.8.28 modified reset value ATOM: 12.3.1.1 modified reset value; modified mode bit field 23:21 12.3.2.2.2 precising decription of serve last compare strategy 12.3.2.4 modified reset value; modified mode bit field 23:21 12.3.3.4.1 precising of one shot mode handling 12.3.3.5 modified reset value; modified mode bit field 23:21 12.3.4.9 modified reset value;

		modified mode bit field 23:21 12.5 replaced links to section 11 for AGC register by links to section 12 12.6.2-8 replaced linked register by detailed register description 12.6.9 additional note for RST_CCU0 12.6.20 modified reset value MCS: 13.3.3 precising usage of instructions in conjunction with STA 13.4.10 modified reset value DPLL: 16.8.6.7 replaced invalid by inactive 16.11 precising section headline and content 16.11ff replaced lower case letters in bit fields by upper case letters; added missing prefix "DPLL_" to all register or memory labels 16.11.1 precising bit fields 15:11 and 24:16 16.11.7 replaced bit field 23:0 by 31:0; additional note for limitations 16.11.14 precising NUTE definition; precising WNUT definition 16.11.17,18,21 replaced lost by loss 16.11.20 modified reset value 16.11.30 precising SOR, MS, TOR, LOCK2, LOCK1 definition 16.11.37ff replaced section 16.12ff by 16.11.37-40 16.11.41ff replaced section 16.13ff by 16.11.41-91 16.11.50 precising TOV definition 16.11.51 precising TOV_S definition 16.11.54 precising MPVAL1 definition 16.11.55 precising MPVAL2 definition 16.11.76 precising TLR definition 16.11.77 precising SLR definition 16.11.92ff replaced section 16.14ff by 16.11.92-95 16.11.94 precising PD_S definition 16.11.96ff replaced section 16-15ff by 16.11.96-98 16.11.98 replaced ACBj_ by ACB_ 16.12ff added missing prefix "DPLL_" to all register or memory labels 16.12.1 precising RDT_T definition 16.12.2 precising TSF_T definition 16.12.3 precising PD, NT definition

		16.12.4 precising DT_T definition SPE: 17.5.10 modified reset value ICM: 18.5.2 added missing suffix ”I” to all DPLL interrupt signals; replaced GLI by GL1I and LLI by LL1I CMP: 19.7.5 modified reset value MON: 20.4 replaced MCS[z]_CH[x]_MCA by MCA_i_x 20.8.1 replaced MCS[x] by MCSx 20.8.2,3 replaced MCS[z]_CH[x]_MCA by MCA_i_x