Table 5 **Sub-module groups** (cont'd)

Chapter	Sub-module	Group
Section 28.20	Digital PLL (DPLL)	Dedicated
Section 28.21	Sensor Pattern Evaluation Module (SPE)	BLDC support
Section 28.22	Interrupt Concentrator Module (ICM)	Interrupt services
Section 28.23	Output Compare Unit (CMP)	Safety features (not part of the Infineon safety manual)
Section 28.24	Monitoring Unit (MON)	Safety features (not part of the Infineon safety manual)

28.4 GTM Architecture

28.4.1 Overview

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

The cluster view of a GTM_IP architecture is depicted in **Figure 6**. This is a generic figure which shows a possible GTM-IP device configuration.

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

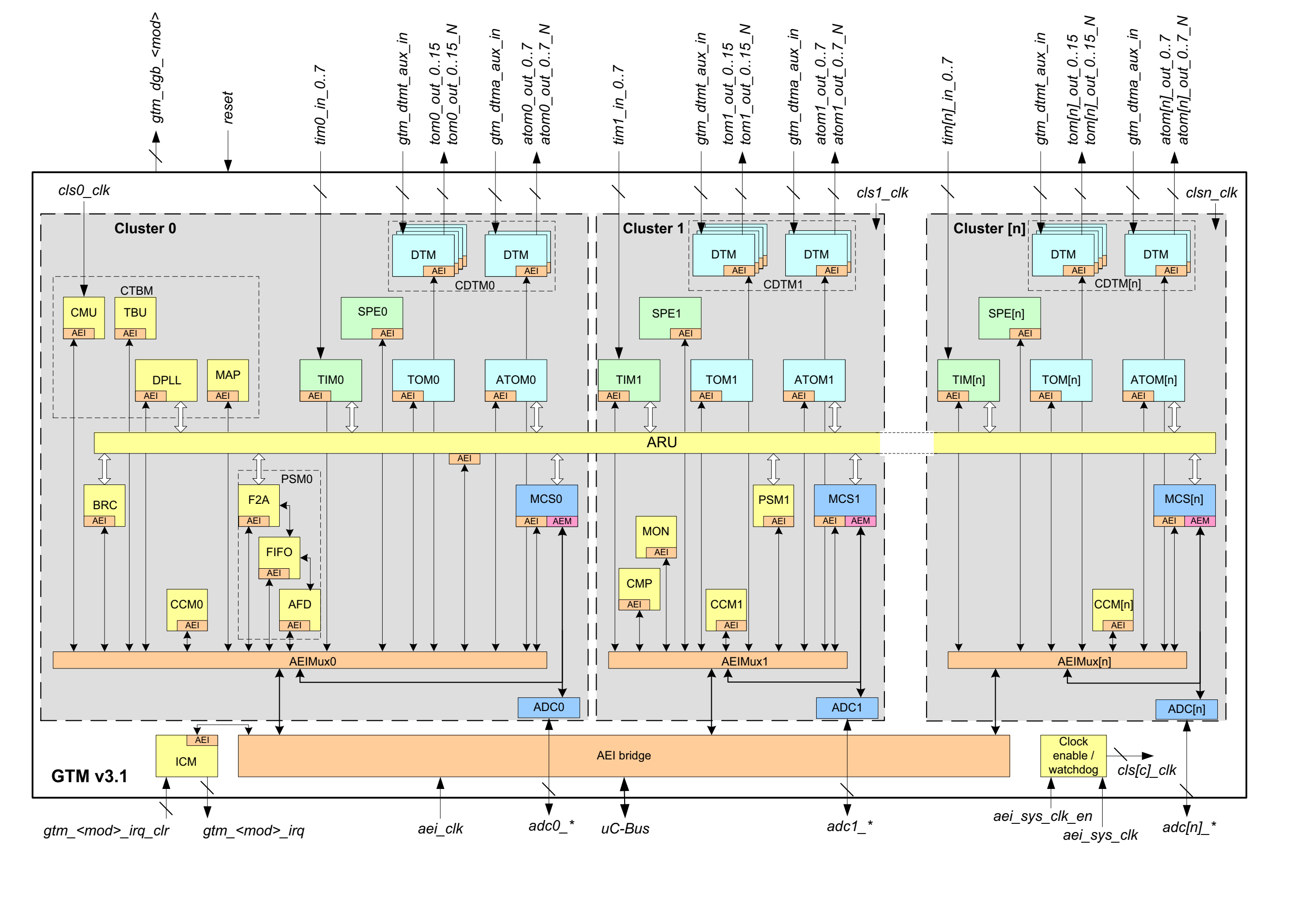


Figure 6 GTM Architecture Block Diagram

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

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

The routed data word size of the ARU is 53 bit. The data word can logically be split into three parts. These parts are shown in **Figure 7**. Bits 0 to 23 and bits 24 to 47 typically hold data for the operation registers of the GTM. 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 time bases TBU_TS0, TBU_TS1 and TBU_TS2. Bits 48 to 52 can contain control bits to send control information from one sub-module to another. These ARU Control Bits (ACB) can have a different meaning for different sub-modules.It is also possible to route data from a source to a destination and the destination can act later on as source for another destination. These routes through the GTM are further on called *data streams*. For a detailed description of the ARU sub-module please refer to the ARU chapter.

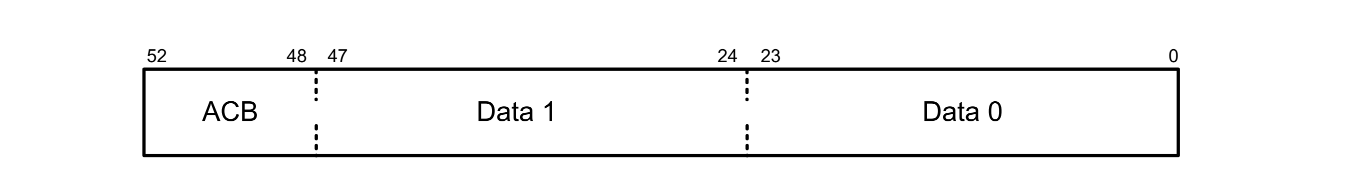


Figure 7 ARU Data Word Description

The BRC is able to distribute data from one source module to more than one destination modules connected to the ARU. The PSM sub-module consists of three sub-units, the AEI-to-FIFO Data Interface (AFD), FIFO-to-ARU Interface (F2A) and the FIFO itself. The PSM can serve as a data storage for incoming data characteristics or as parameter storage for outgoing data. This data is stored in a RAM that is logically located inside the FIFO sub-unit, but physically the RAM is implemented and integrated by the silicon vendor with his RAM implementation technology. Therefore, the GTM provides the interface to the RAM at its module boundary. The AFD sub-unit is the interface between the FIFO and the GTM SoC system bus interface AEI (see **Section 28.4.2.1** for detailed discussion). The F2A sub-unit is the interface between the FIFO sub-unit and the ARU.

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

The Clock Management Unit (CMU) serves up to 13 different clocks for the GTM and up to three external clock pins *GTM_ECLK0…2*. It acts as a clock divider for the system clock. The counters implemented inside other sub-modules are typically driven from this sub-module. Please note, that the CMU clocks are implemented as enable signals for the counters while the whole system runs with the GTM global clock *SYS_CLK*.[1] This global clock typically corresponds to the micro controller bus clock the GTM-IP is connected to and should not exceed 100MHz because of the power dissipation of the used transistors where the GTM is implemented with.

The TBU provides up to three independent common time bases for the GTM. 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_inc1*c and *sub_inc2c* outputs. The DPLL generates the higher frequent clock signals *sub_inc1*, *sub_inc2*, *sub_inc1c* and *sub_inc2c* on behalf of the frequencies of up to two input signals. These two input signals can be selected out of six incoming signals from the TIM0 sub-module. In this sub-module the incoming signals are filtered and transferred to the MAP sub-module where two of these six signals are selected for further processing inside the DPLL.

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

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

The ATOMs offer the additional functionality to generate complex output signals without CPU interaction by serving these complex waveform characteristics by other sub-modules that are connected to the ARU like the PSM or Multi Channel Sequencer (MCS). While the internal operation and shadow registers of the TOM channels

[1] SYS_CLK = f_{GTM} clock provided by the GTM wrapper. f_{GTM} max value is $2xf_{SPB}$ = 200MHz

are 16 bit wide, the operation and shadow registers of the ATOM channels are 24 bit wide to have a higher resolution and to have the opportunity to compare against time base values coming from the TBU.

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

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

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

In the described implementation the sub-modules of the GTM 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.

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

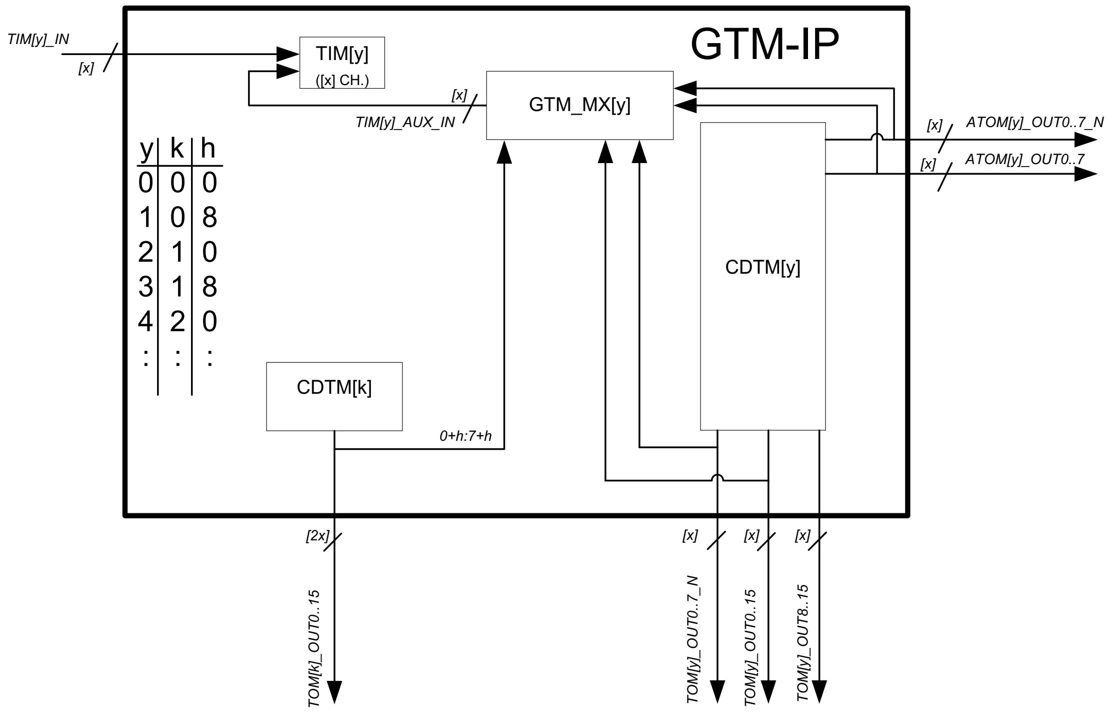
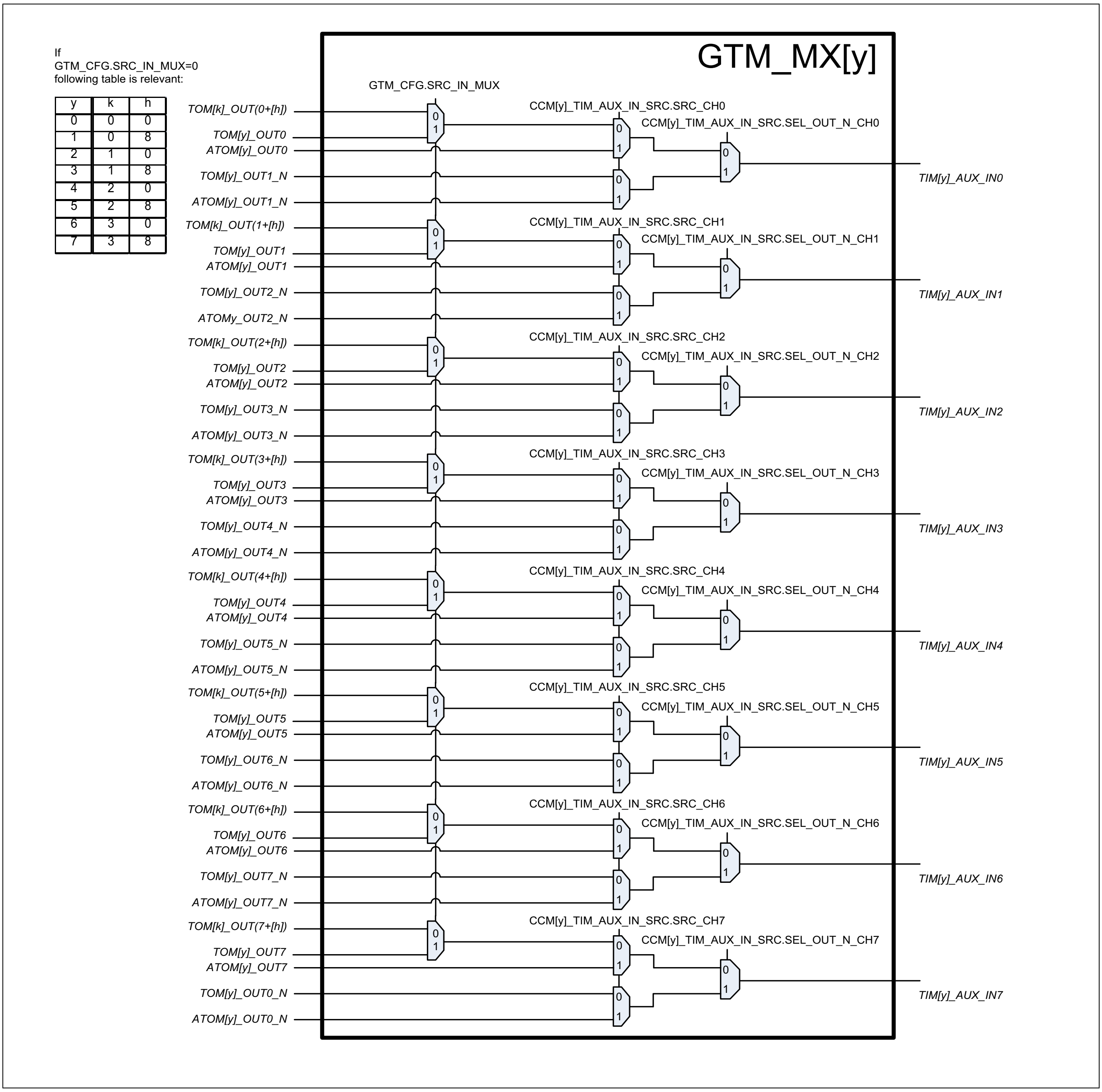


Figure 8 GTM signal multiplex

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


If
GTM_CFG.SRC_IN_MUX=0
following table is relevant:

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

Figure 9 TIM auxiliary input multiplexing

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

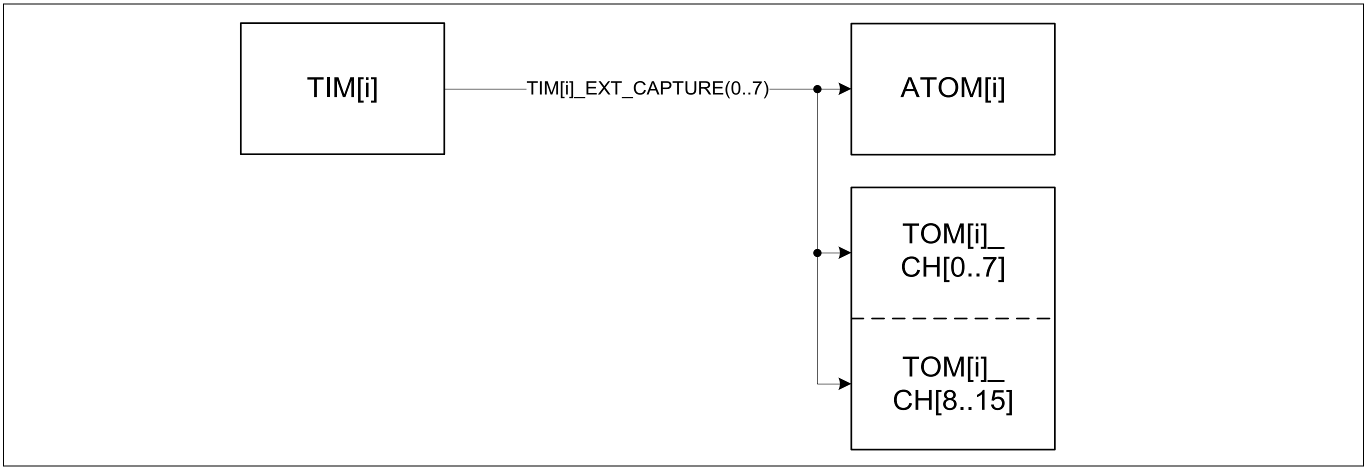


Figure 10 TIM external capture forwarding to TOM and ATOM

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

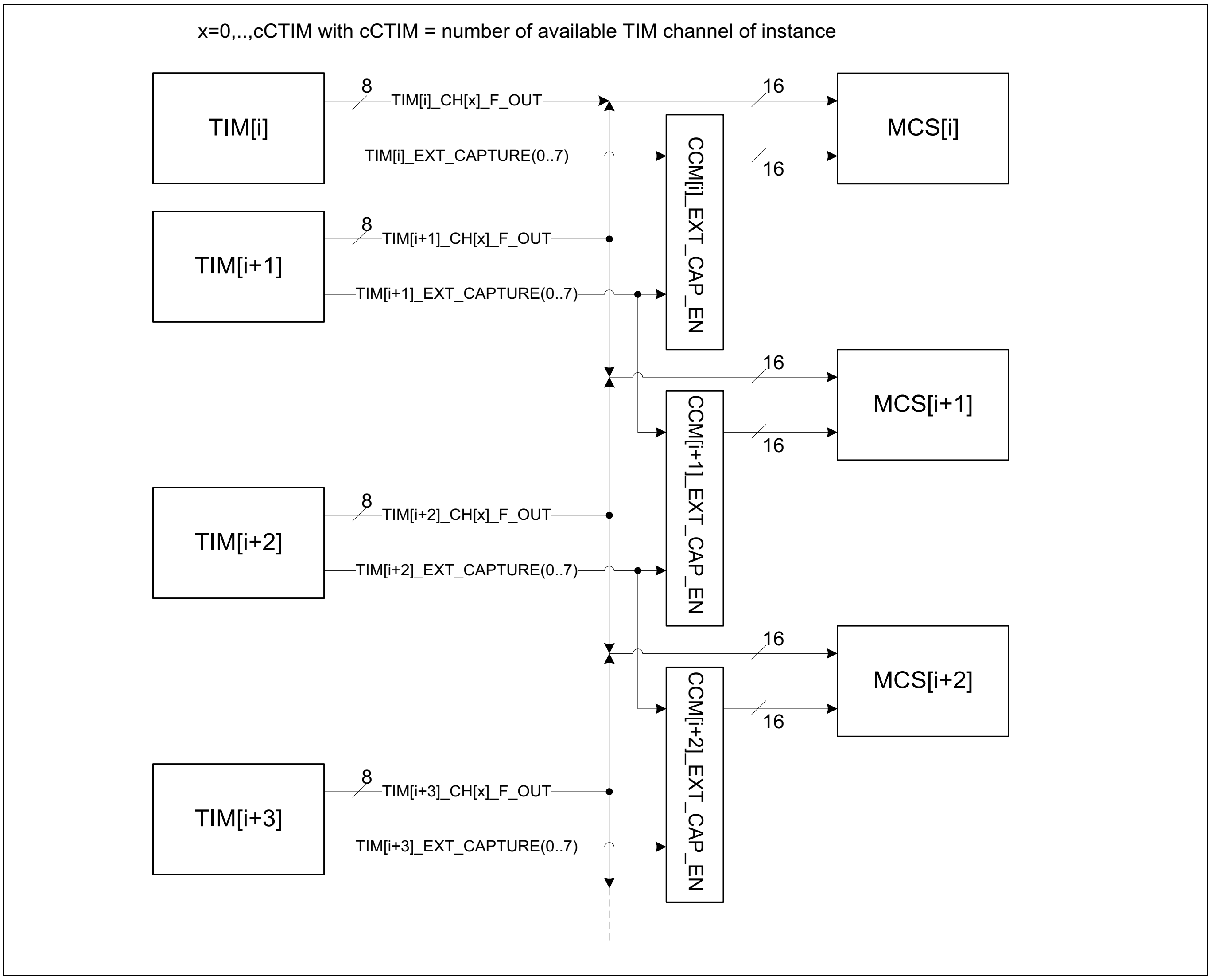


Figure 11 TIM to MCS signal forwarding

28.4.2 GTM Interfaces

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

Another interface is the bus interface where the GTM can be connected to the SoC system bus. This generic bus interface is described in more detail in **Section 28.4.2.1**. The last interface is the interrupt controller interface. The GTM provides several interrupt lines coming from the various sub-modules. These interrupt lines are concentrated inside the ICM and have to be adapted to the dedicated micro controller environment where each interrupt handling can look different. The interrupt concept is described in more detail in **Section 28.4.5**.

28.4.2.1 GTM Generic Bus Interface (AEI)

The GTM 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:

Table 6 AEI bus signals

Signal name	I/O	Description	Bit width
AEI_SEL	I	GTM select line	1
AEI_ADDR	I	GTM 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:

Table 7 AEI Status Signal

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

The signal value 0b00 is returned if no error occurred during AEI access.

The signal value 0b01 is returned if the bus address is not an integer multiple of 4 (byte addressing).

The signal value 0b11 is returned if the address is not handled in the GTM.

The signal value 0b10 is returned if the written register is a protected register (e.g. protected by bit RF_PROT) or if the register is temporarily not writable because of sub-module internal state or the clock of the relevant cluster is disabled.

In case of an illegal write access signaled by status 0b10 the register will not be modified.

Reading registers will never return status 0b10.

Note: *Exception for register CMU_CLK_CTRL. In case of write access signalled by aei_status 0b10 the register will be modified each completely disabled bit.*

The detailed list of register addresses with return status 0b10 can be found in the appendix.

28.4.2.2 GTM Multi-master and multitasking support

To support multi-master and multi-task access to the registers of the GTM 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 with a double bit mechanism, where the writing of 0b00 and 0b11 has no effect on the register bit and where 0b01 sets the bit and 0b10 resets the bit. If the CPU wants to read the status of the bit it always gets a 0b00 if the bit is reset and it gets an 0b11 if the bit is set.

28.4.3 ARU Routing Concept

One central concept of the GTM is the routing mechanism of the ARU sub-module for data streams. Each data word transferred between the ARU and its connected sub-module is 53 bit wide. It is important to understand this concept in order to use the resources of the GTM 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 12** 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.

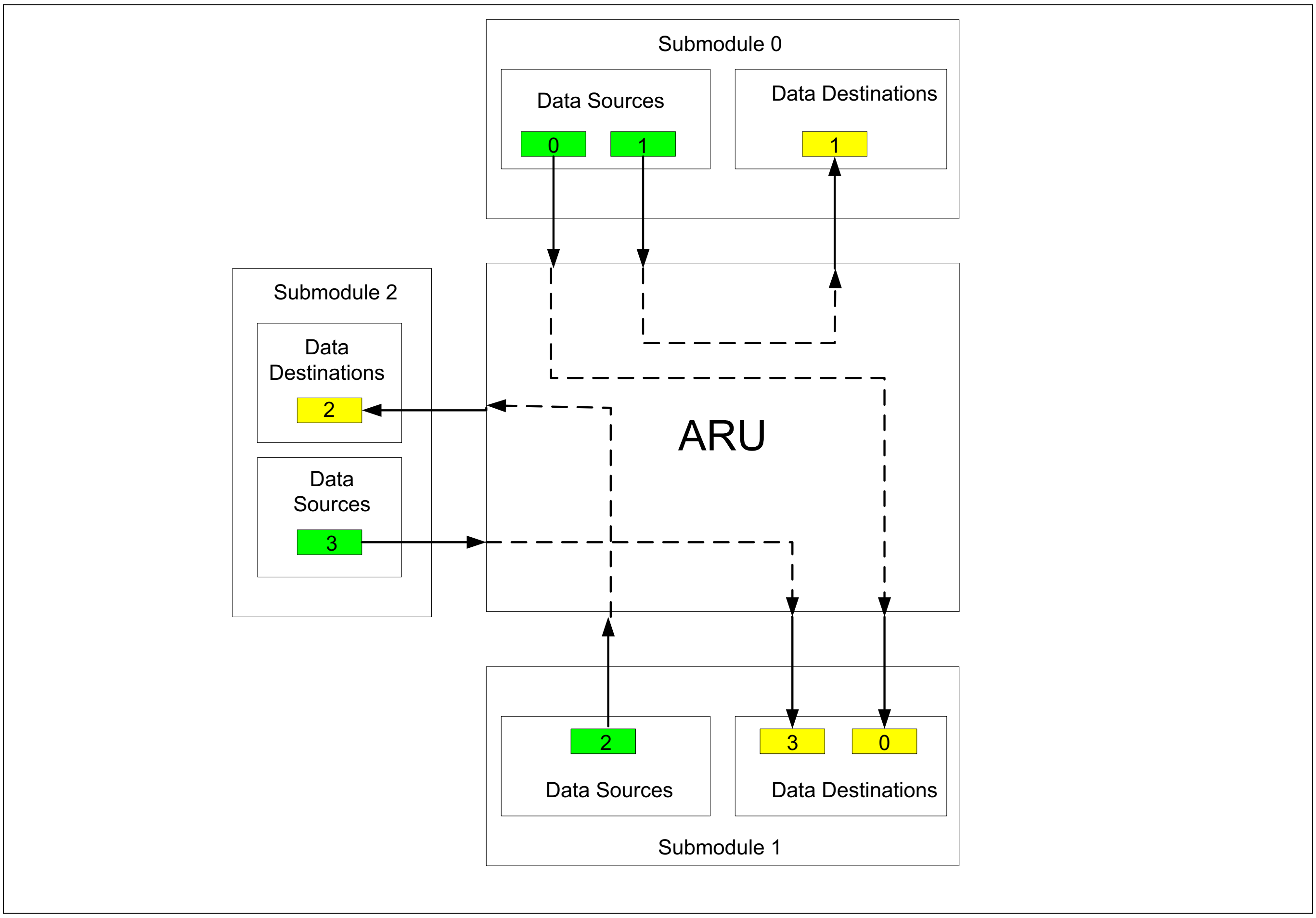


Figure 12 Principle of data routing using ARU

The configuration of the data streams is realized according to the following manner: Each data source has its fixed and unique source address: The ARU read ID.The fixed address of each data source is pointed out by the numbers in the green boxes of **Figure 12**. The address definitions of all available data sources in the GTM can be obtained from the product specific appendix. The connection from a specific data source to a specific data destination is defined by configuring the corresponding address of a data source (i.e. the ARU read ID) in the desired data destination. The configured address of each data destination is pointed out by the numbers in the yellow boxes of **Figure 12**.

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

There is one sub-module for which this destructive read access does not hold. This is the BRC sub-module configured in Maximum Throughput Mode (MTM). For a detailed description of this module please refer to chapter “Broadcast Module (BRC)”.

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

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

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

Table 8 ARU source and destination address count per instance

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

28.4.3.1 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.

28.4.3.2 ARU Blocking Mechanism

Another important concept of the ARU is its blocking mechanism that is implemented for transferring data from a data source to a data destination. This mechanism is used by ARU connected sub-modules to synchronize the sub-modules to the routed data streams. Figure 13 explains the blocking mechanism.

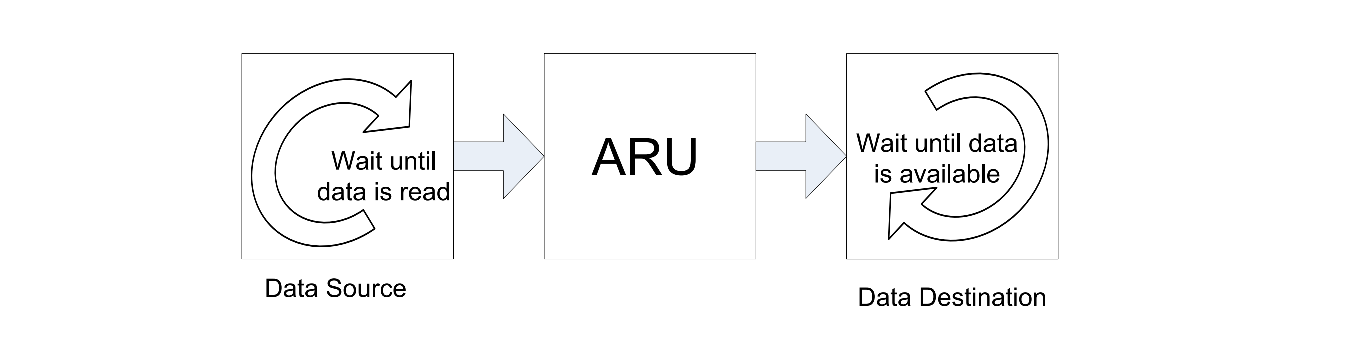


Figure 13 Graphical representation of ARU blocking mechanism

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

In any case the round trip time for the ARU has a fixed reset value for a specific device configuration. The end value of the round-trip counter can be changed with a configuration register **ARU_CADDR_END** inside the ARU. For more details see the ARU specific chapter.

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

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

28.4.4 GTM Clock and Time Base Management (CTBM)

Inside the GTM several sub-units are involved in the clock and time base management of the whole GTM. **Section 28.4.4.1** 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.

28.4.4.1 GTM Clock and time base management architecture

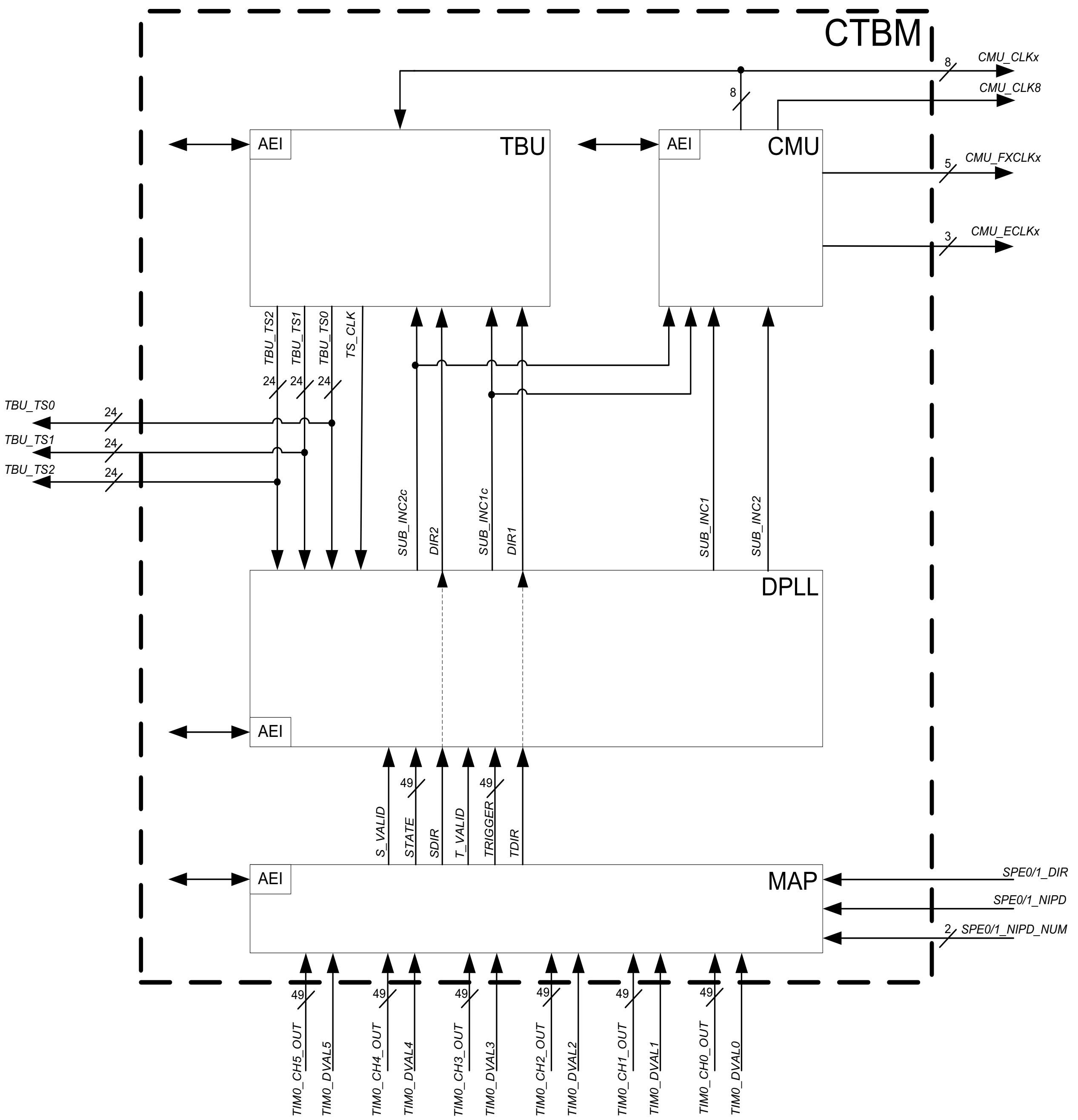


Figure 14 CTBM architecture

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

The five (5) *CMU_FXCLK[y]* (y: 0…4) clocks are used by the TOM sub-module for PWM generation.

A maximum of nine (9) *CMU_CLK[x]* (x: 0…8) clocks are used by other sub-modules of the GTM for signal generation.

Inside the Time Base Unit (TBU) one of *CMU_CLK[x]* (x: 0…7) clocks is used per channel to generate a common time base for the GTM. Besides the *CMU_CLK[x]* signals, the TBU can use the compensated *SUB_INC[i]c* (i: 1,2) signals coming from the DPLL sub-module for time base generation. This time base then typically represents an angle clock for an engine management system. For the meaning of compensated (*SUB_INC[i]c*) and uncompensated (*SUB_INC[i]*) DPLL signals please refer to the DPLL chapter. 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 “Time Base Unit (TBU)”.

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

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

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

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

28.4.4.2 Cyclic Event Compare

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

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

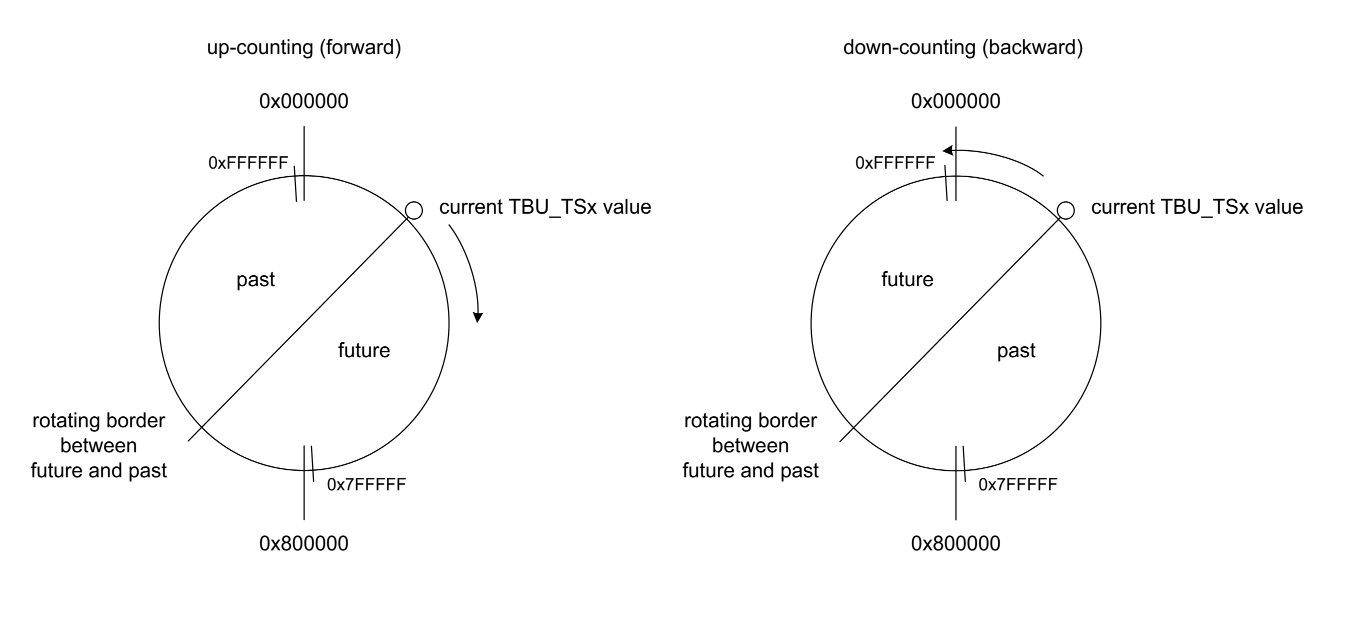


Figure 15 Cyclic event counter representing time or angle

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

28.4.5 GTM Interrupt Concept

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

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

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

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

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

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

- Level mode,

- Pulse mode,
- Pulse-Notify mode,
- Single-Pulse mode.

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

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

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

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

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

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

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

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

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

The mechanism for triggering interrupts with **IRQ_FORCINT** is globally disabled after reset. It has to be explicitly enabled by clearing the bit **RF_PROT** in the register **GTM_CTRL** (see **Chapter 28.4.9.3**)

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

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

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

To be able to detect the module source of the error interrupt the ICM provides the register **ICM_IRQG_MEI**.

The error interrupt causing channel can be determined for the module FIFO by evaluating the ICM register **ICM_IRQG_CEI0**.

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

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

28.4.5.1 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 16**, may be triggered by the interrupt source of the sub-module or by software performing a write access to the corresponding register **IRQ_FORCINT**, with a disabled bit **RF_PROT** in register **GTM_CTRL**.

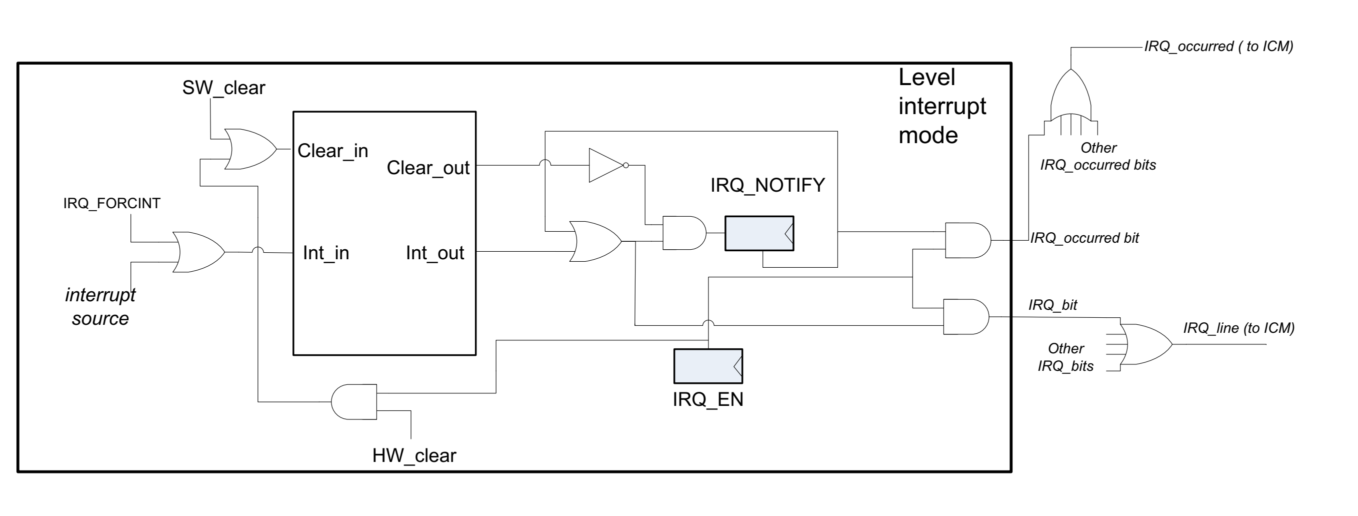


Figure 16 Level interrupt mode scheme

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 16**. A clear event can be performed by writing '1' to the corresponding bit in the register **IRQ_NOTIFY** leading to a pulse on signals *SW_clear*. A clear event may also result from an externally connected signal *GTM_<MOD>_IRQ_CLR*, which is routed to the signal *HW_clear* of **Figure 16**. However, the hardware clear mechanism is only possible, if the corresponding interrupt is enabled by register **IRQ_EN**.

As **Table 9** shows, interrupt events are dominant in the case of a simultaneous interrupt event and clear event.

Table 9 Priority of Interrupt Events and Clear Events

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

As shown in **Figure 16** an occurred interrupt event is signaled as a constant signal level with value 1 to the signal *IRQ_bit*, if the corresponding interrupt is enabled in register **IRQ_EN**.

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

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

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

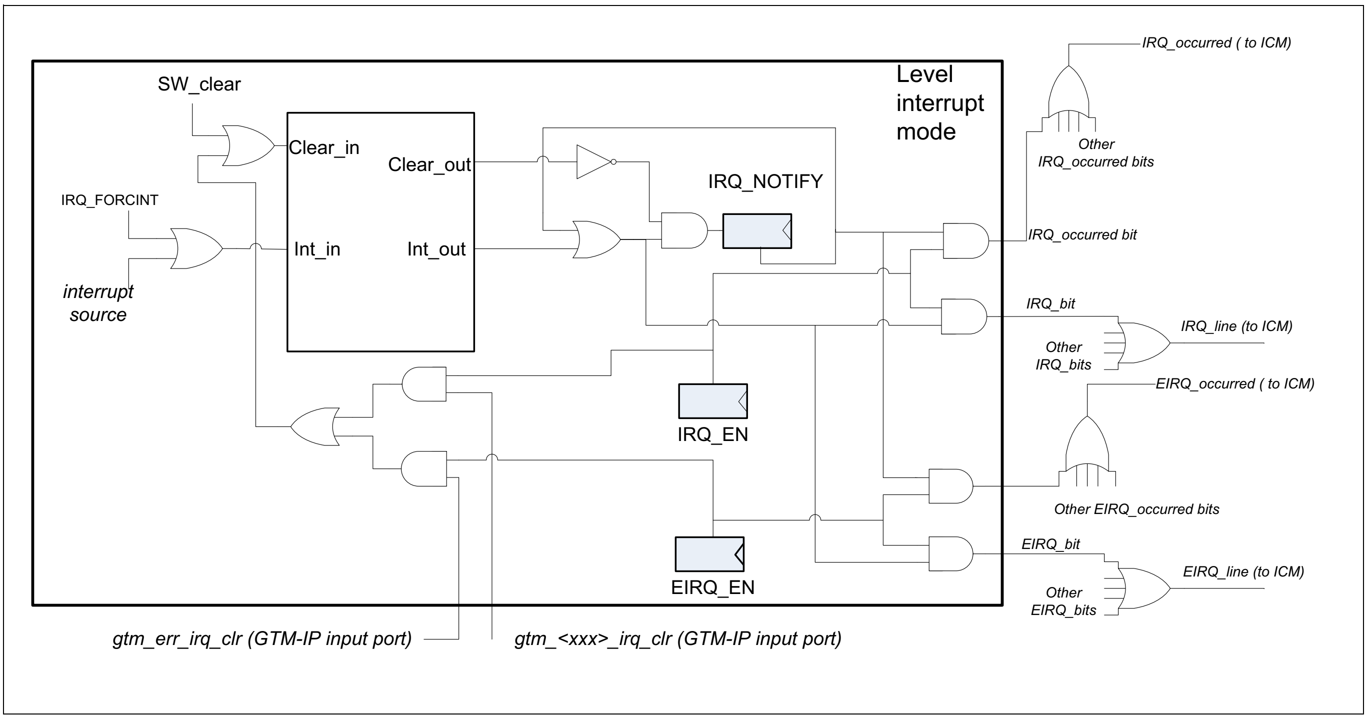


Figure 17 Level interrupt scheme for modules AEI-bridge, BRC, FIFO, TIM, MCS, DPLL, SPE, CMP

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 17**. A clear event can be performed by writing '1' to the corresponding bit in the register **IRQ_NOTIFY** leading to a pulse on signals *SW_clear*. A clear event may also result from the externally connected signal *gtm_<MOD>_irq_clr* or *gtm_err_irq_clr*, which is routed as an *HW_clear* to *Clear_in* of **Figure 17**. 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 17** an occurred interrupt event is signaled as a constant signal level with value 1 to the signal *IRQ_bit*, if the corresponding interrupt is enabled in register **IRQ_EN**.

28.4.5.2 Pulse interrupt mode

The Pulse interrupt mode behavior can be observed from **Figure 18**.

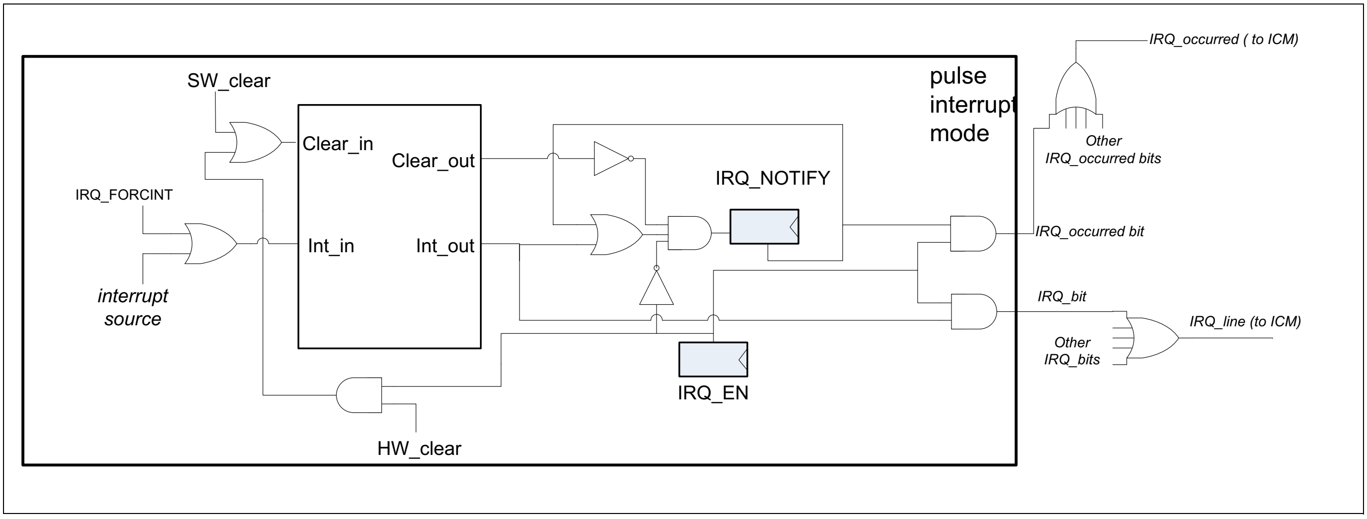


Figure 18 Pulse interrupt mode scheme

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.

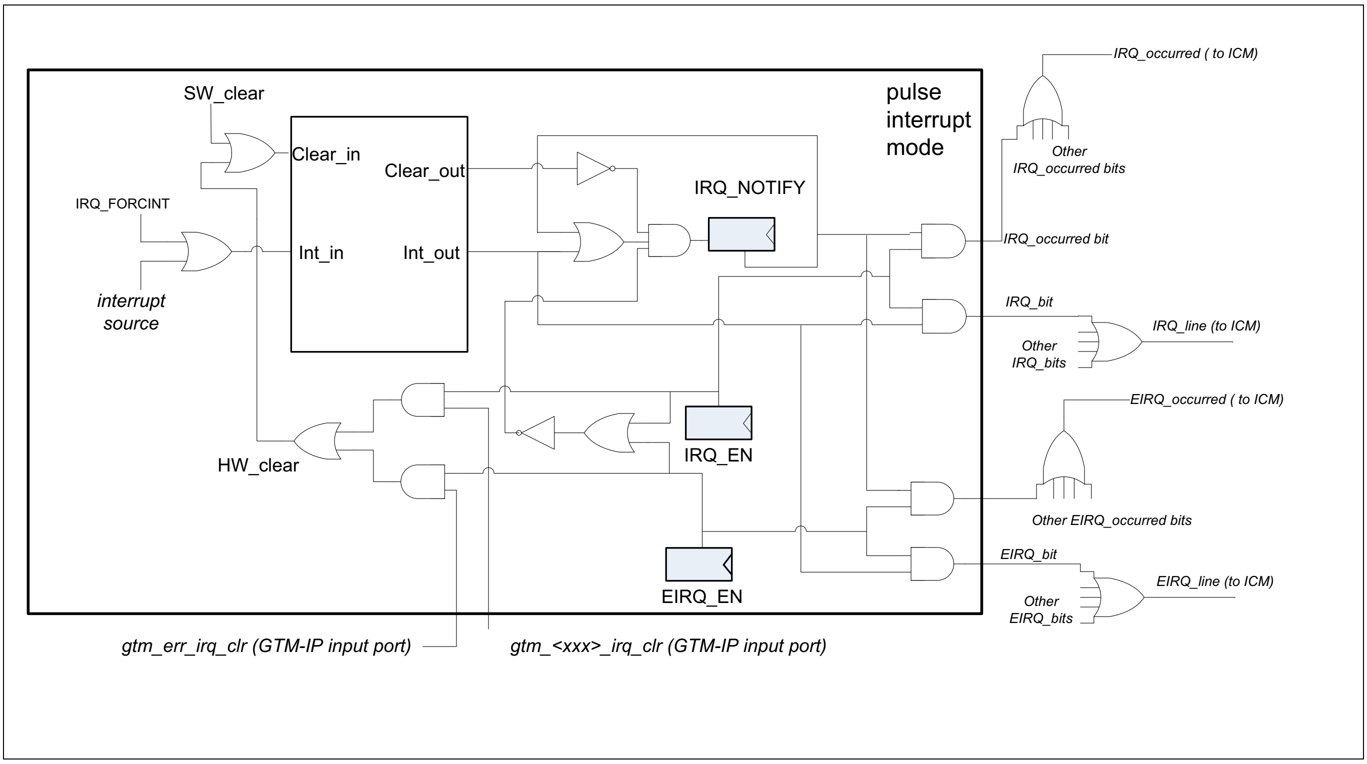


Figure 19 Pulse interrupt scheme for modules AEI-bridge, BRC, FIFO, TIM, MCS, DPLL, SPE, CMP

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.

28.4.5.3 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 20**.

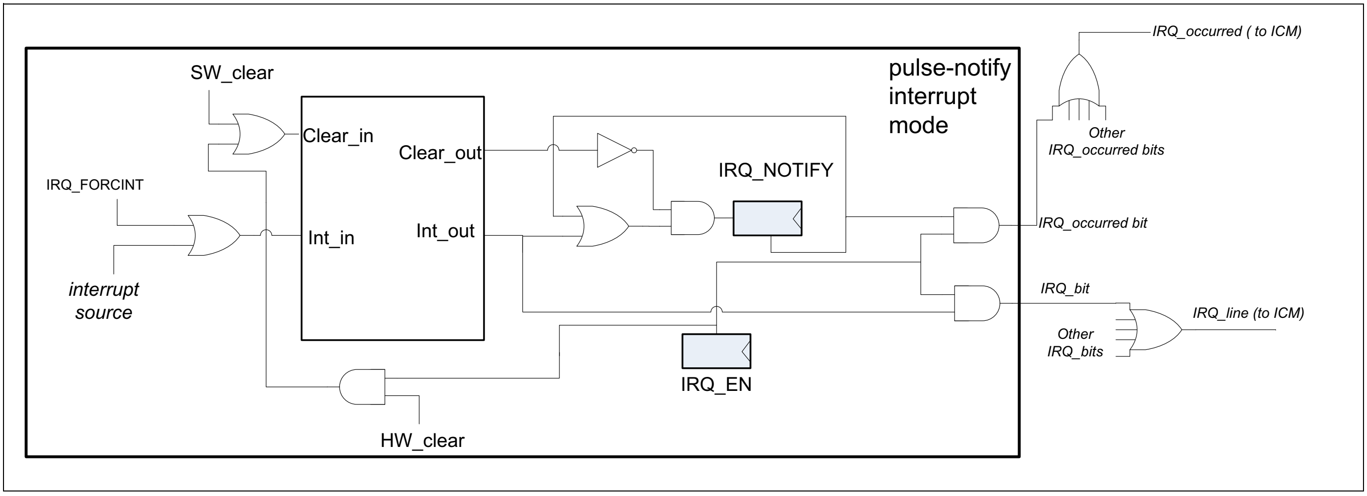


Figure 20 Pulse-notify interrupt mode scheme

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

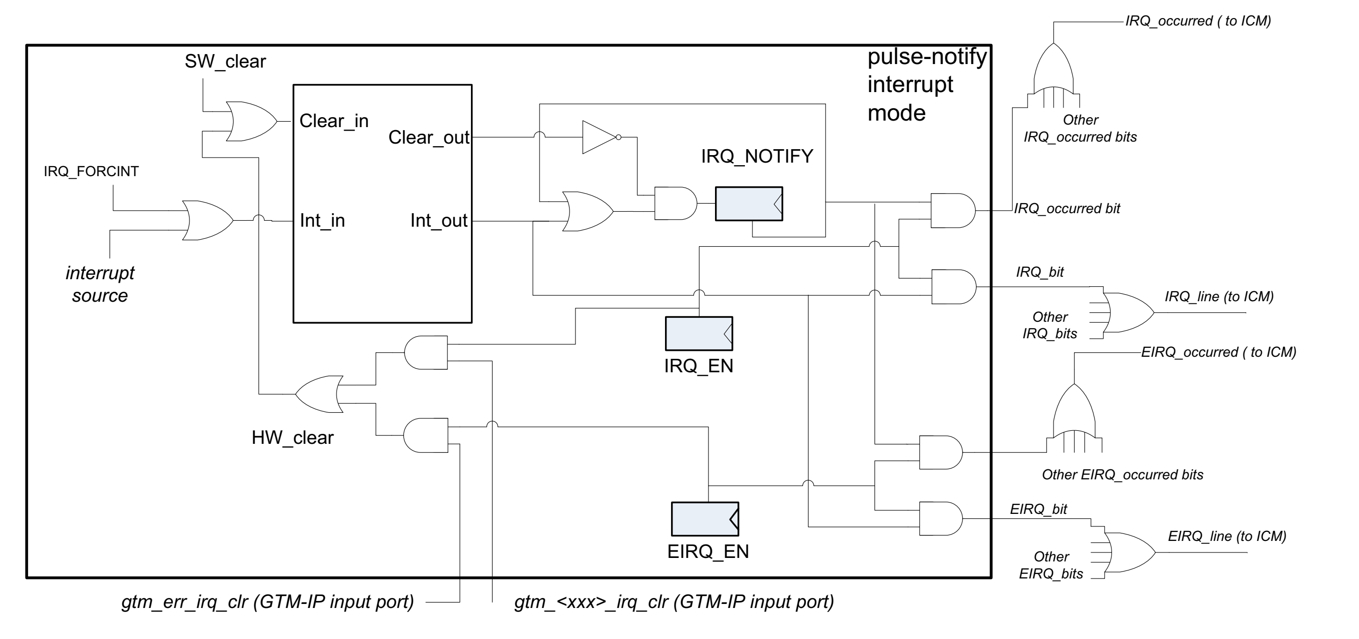


Figure 21 Pulse-notify interrupt scheme for modules AEI-bridge, BRC, FIFO, TIM, MCS, DPLL, SPE, CMP

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 21**.

28.4.5.4 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 22**.

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

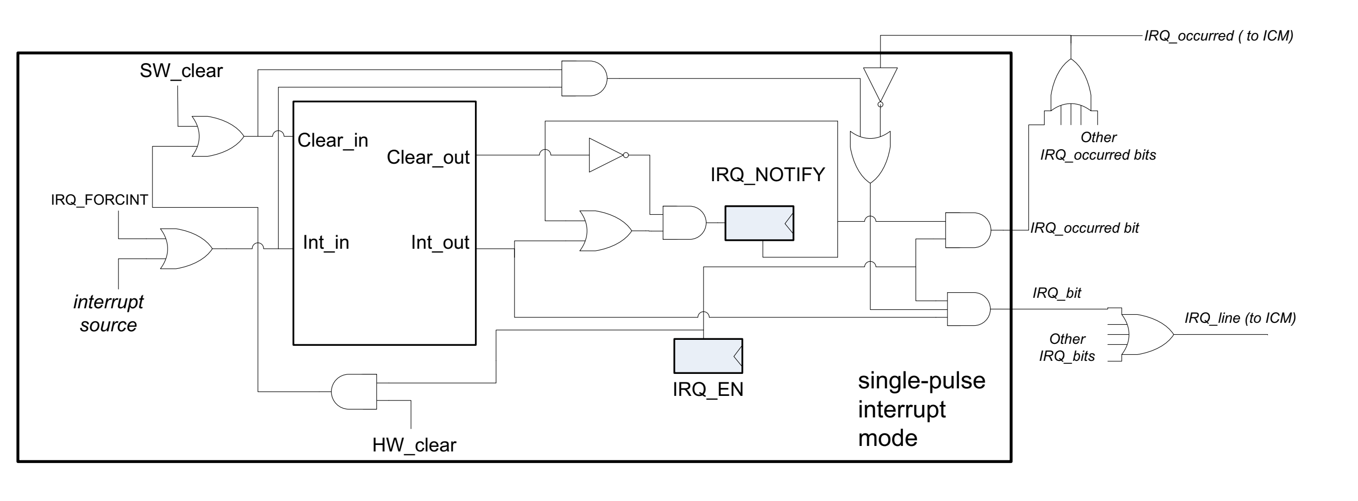


Figure 22 Single-pulse interrupt mode scheme

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

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

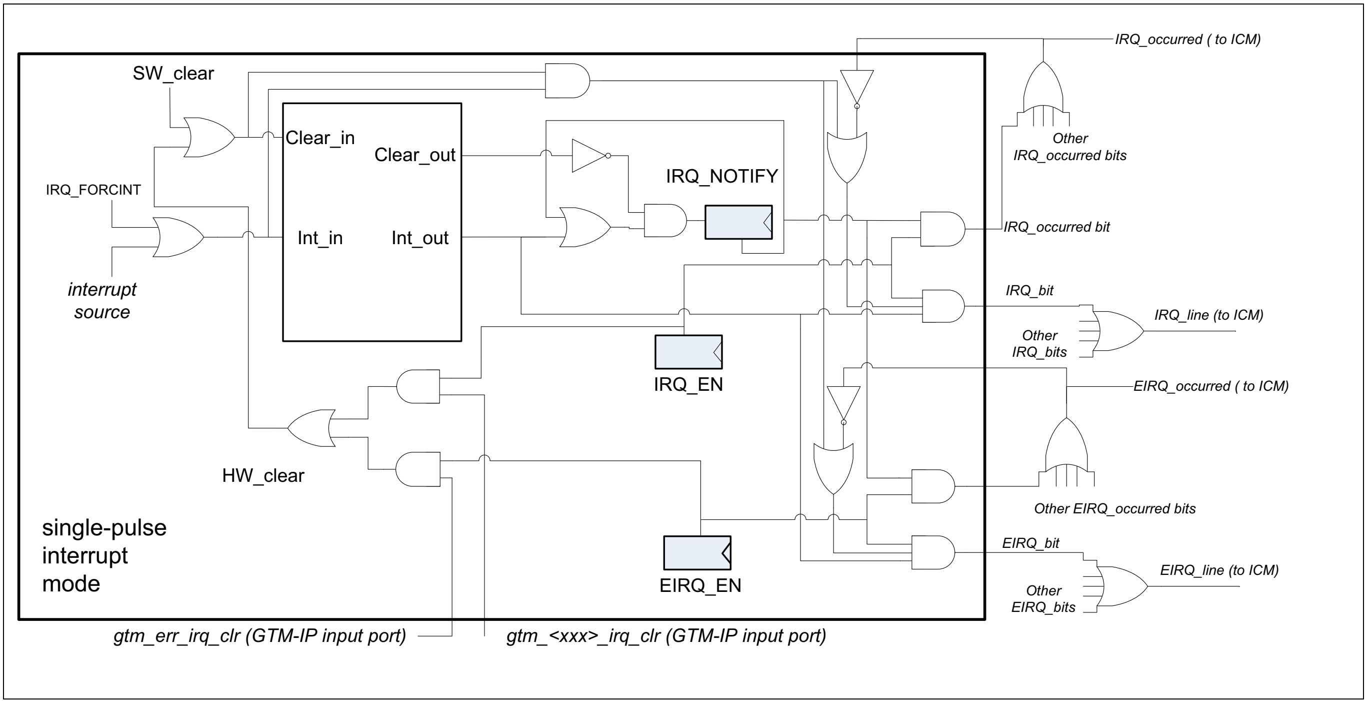


Figure 23 Single-pulse interrupt scheme for modules AEI-bridge, BRC, FIFO, TIM, MCS, DPLL, SPE, CMP

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 23**.

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

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

28.4.5.5 GTM 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 sub-module please refer to chapter “Interrupt Concentrator Module (ICM)”.

28.4.6 GTM Software Debugger Support

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

Table 10 describes the behavior of some GTM registers with special functionality on behalf of read accesses from the AEI bus interface.

Table 10 Register behavior in case of Software Debugger accesses

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

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

28.4.7 GTM Programming conventions

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

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

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

28.4.8 GTM TOP-Level Configuration Register Overview

Table 11 GTM TOP-Level Configuration Register Overview

Register name	Description	see Page
GTM_REV	GTM Version control register	30
GTM_RST	GTM Global reset register	31
GTM_CTRL	GTM Global control register	31
GTM_AEI_ADDR_XPT	GTM AEI Timeout exception address register	32
GTM_AEI_STA_XPT	GTM AEI Non zero status register	33
GTM_IRQ_NOTIFY	GTM Interrupt notification register	34
GTM_IRQ_EN	GTM Interrupt enable register	36
GTM_EIRQ_EN	GTM Error interrupt enable register	44

Table 11 **GTM TOP-Level Configuration Register Overview** (cont'd)

Register name	Description	see Page
GTM_IRQ_FORCINT	GTM Software interrupt generation register	37
GTM_IRQ_MODE	GTM top level interrupts mode selection	39
GTM_BRIDGE_MODE	GTM AEI bridge mode register	39
GTM_BRIDGE_PTR1	GTM AEI bridge pointer 1 register	41
GTM_BRIDGE_PTR2	GTM AEI bridge pointer 2 register	42
GTM_MCS_AEM_DIS	GTM MCS master port disable register	43
GTM_CLS_CLK_CFG	GTM Cluster Clock Configuration	45
GTM_CFG	GTM Configuration register	46

28.4.9 GTM TOP-Level Configuration Registers Description

28.4.9.1 Register GTM_REV

Please keep in mind, that the actual Revision number is the reset value. This reset value is dependent on the delivery done by Bosch AE for the actual device. In case of Infineon's decision to change via metal fix for a different version, the reset value contains the initial version.

GTM Version Control Register

Note: The numbers are encoded in BCD. Values "A" - "F" are characters.

REV

GTM Version Control Register **(000000_H)** **Application Reset Value: $3153\ 15B6_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DEV_CODE2				**DEV_CODE1**				**DEV_CODE0**				**MAJOR**			
r				r				r				r			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MINOR				**NO**				**STEP**							
r				r				r							

Field	Bits	Type	Description
STEP	7:0	r	**Release step** GTM Release step
NO	11:8	r	**Delivery number** Define delivery number of GTM specification.
MINOR	15:12	r	**Minor version number** Define minor version number of GTM specification.
MAJOR	19:16	r	**Major version number** Define major version number of GTM specification.
DEV_CODE0	23:20	r	**Device encoding digit 0** Device encoding digit 0.
DEV_CODE1	27:24	r	**Device encoding digit 1** Device encoding digit 1.
DEV_CODE2	31:28	r	**Device encoding digit 2** Device encoding digit 2.

28.4.9.2 Register GTM_RST

GTM Global Reset Register

RST
GTM Global Reset Register **(000004$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0				**BRIDGE_MODE_WRDIS**	**0**										
r				rw	r										

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															**0**
r															r

Field	Bits	Type	Description
BRIDGE_MODE_WRDIS	27	rw	**GTM_BRIDGE_MODE write disable** This bit is write protected by bit RF_PROT 0_B Writing of GTM_BRIDGE_MODE register is enabled 1_B Writing of GTM_BRIDGE_MODE register is disabled
0	0, 26:1, 31:28	r	**Reserved** Read as zero, shall be written as zero.

28.4.9.3 Register GTM_CTRL

GTM Global Control Register

CTRL
GTM Global Control Register **(000008$_H$)** **Application Reset Value: 0000 0001$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AEIM_CLUSTER				**0**			**TO_VAL**					**0**		**TO_MODE**	**RF_PROT**
r				r			rw					r		rw	rw

Field	Bits	Type	Description
RF_PROT	0	rw	**RST and FORCINT protection** 0_B SW RST (global), SW interrupt FORCINT, and SW RAM reset functionality is enabled 1_B SW RST (global), SW interrupt FORCINT, and SW RAM reset functionality is disabled
TO_MODE	1	rw	**AEI timeout mode** 0_B 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_B Abort: In addition to observe mode, the pending access will be aborted by signaling an illegal module access on aei_status and sending ready. In case of a read, deliver as data 0 by serving of next AEI accesses.
TO_VAL	8:4	rw	**AEI timeout value** 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.
AEIM_CLUSTER	15:12	r	**AEIM cluster number** These bits show the number of the AEI master port cluster which throws the interrupts *AEIM_USP_ADDR, AEIM_IM_ADDR* and *AEIM_USP_BE* depending on the AEI master port access status.Note: If one of the corresponding irq notify bits (6:4) is set, this bit field will be frozen until the interrupt notify bits (6:4) are cleared.
0	3:2, 11:9, 31:16	r	**Reserved** Read as zero, shall be written as zero.

28.4.9.4 Register GTM_AEI_ADDR_XPT

GTM AEI Timeout Exception Address Register

AEI_ADDR_XPT

GTM AEI Timeout Exception Address Register (00000C$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0											**TO_W1R0**	**TO_ADDR**			
r											r	r			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TO_ADDR															
r															

Field	Bits	Type	Description
TO_ADDR	19:0	r	**AEI timeout address** This bit field defines the AEI address for which the AEI timeout event occurred.
TO_W1R0	20	r	**AEI timeout Read/Write flag** This bit defines the AEI Read/Write flag for which the AEI timeout event occurred.
0	31:21	r	**Reserved** Read as zero, shall be written as zero.

28.4.9.5 Register GTM_AEI_STA_XPT

GTM AEI Non Zero Status Register

AEI_STA_XPT
GTM AEI Non Zero Status Register **(00002C$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0											**W1R0**	**ADDR**			
r											r	r			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR															
r															

Field	Bits	Type	Description
ADDR	19:0	r	**AEI exception address** This bit field captures the address of the first AEI access resulting with a non-zero AEI status signal. The bit field can be cleared by clearing the interrupt flags AEI_USP_ADDR,AEI_USP_BE, and AEI_IM_ADDR in the register GTM_IRQ_NOTIFY.
W1R0	20	r	**AEI exception Read/Write flag** This bit defines the AEI Read/Write flag for which the AEI non-zero event occurred. This bit field captures the address of the first AEI access resulting with a non-zero AEI status signal. The bit field can be cleared by clearing the interrupt flags AEI_USP_ADDR, AEI_USP_BE, and AEI_IM_ADDR in the register GTM_IRQ_NOTIFY.
0	31:21	r	**Reserved** Read as zero, shall be written as zero.

28.4.9.6 Register GTM_IRQ_NOTIFY

GTM Interrupt Notification Register

IRQ_NOTIFY

GTM Interrupt Notification Register **(000010$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0		**CLK_EN_EXP_STATE1**	**CLK_EN_EXP_STATE0**	**0**		**CLK_EN_ERR_STATE1**	**CLK_EN_ERR_STATE0**	**0**							
r		r	r	r		r	r	r							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0							**CLK_PER_ERR**	**CLK_EN_ERR**	**AEIM_USP_BE**	**AEIM_IM_ADDR**	**AEIM_USP_ADDR**	**AEI_USP_BE**	**AEI_IM_ADDR**	**AEI_USP_ADDR**	**AEI_TO_XPT**
r							rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
AEI_TO_XPT	0	rw	**AEI timeout exception occurred** This bit will be cleared on a CPU write access of value '1'. (As the bit is rw, otherwise no clear.) A read access leaves the bit unchanged. 0_B No interrupt occurred 1_B AEI_TO_XPT interrupt was raised by the AEI Timeout detection unit
AEI_USP_ADDR	1	rw	**AEI unsupported address interrupt** This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. 0_B No interrupt occurred 1_B AEI_USP_ADDR interrupt was raised by the AEI interface
AEI_IM_ADDR	2	rw	**AEI illegal Module address interrupt** This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. 0_B No interrupt occurred 1_B AEI_IM_ADDR interrupt was raised by the AEI interface
AEI_USP_BE	3	rw	**AEI unsupported byte enable interrupt** This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. 0_B No interrupt occurred 1_B AEI_USP_BE interrupt was raised by the AEI interface
AEIM_USP_ADDR	4	rw	**AEI master port unsupported address interrupt** This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. 0_B No interrupt occurred 1_B AEIM_USP_ADDR interrupt was raised by the AEI master port interface

Field	Bits	Type	Description
AEIM_IM_ADDR	5	rw	**AEI master port illegal Module address interrupt** This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. 0_B No interrupt occurred 1_B AEIM_IM_ADDR interrupt was raised by the AEI master port interface
AEIM_USP_BE	6	rw	**AEI master port unsupported byte enable interrupt** This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. 0_B No interrupt occurred 1_B AEIM_USP_BE interrupt was raised by the AEI master port interface
CLK_EN_ERR	7	rw	**Clock enable error interrupt** This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. Read as zero in case of INT_CLK_EN_GEN = 0b1. 0_B No interrupt occurred 1_B CLK_EN_ERR interrupt was raised by clock enable watchdog
CLK_PER_ERR	8	rw	**Clock period error interrupt** This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. Read as zero in case of INT_CLK_EN_GEN = 0b1. 0_B No interrupt occurred 1_B CLK_PER_ERR interrupt was raised by clock enable watchdog
CLK_EN_ERR_STATE0	24	r	**Erroneous clock enable state** This bit field defines the GTM external clk enable state for internal clock aei_sys_clkat occurrence of the CLK_EN_ERR event. **Function only available with INT_CLK_EN_GEN = 0b0:** Read as zero in case of INT_CLK_EN_GEN = 0b1. 0_B Internal clock aei_sys_clk disabled 1_B Internal clock aei_sys_clk enabled
CLK_EN_ERR_STATE1	25	r	**Erroneous clock enable state** This bit field defines the GTM external clk enable state for internal clock aei_sys_clk / 2 at occurrence of the CLK_EN_ERR event. **Function only available with INT_CLK_EN_GEN = 0b0:** Read as zero in case of INT_CLK_EN_GEN = 0b1. 0_B Internal clock aei_sys_clk / 2 disabled 1_B Internal clock aei_sys_clk / 2 enabled
CLK_EN_EXP_STATE0	28	r	**Expected clock enable state** This bit field defines the GTM expected clk enable state for internal clock aei_sys_clk at occurrence of the CLK_EN_ERR event. **Function only available with INT_CLK_EN_GEN = 0b0:** Read as zero in case of INT_CLK_EN_GEN = 0b1. 0_B Internal clock aei_sys_clk disabled 1_B Internal clock aei_sys_clk enabled

Field	Bits	Type	Description
CLK_EN_EXP_STATE1	29	r	**Expected clock enable state** This bit field defines the GTM expected clk enable state for internal clock aei_sys_clk / 2 at occurrence of the CLK_EN_ERR event. **Function only available with INT_CLK_EN_GEN = 0b0:** Read as zero in case of INT_CLK_EN_GEN = 0b1. 0_B Internal clock aei_sys_clk / 20 disabled 1_B Internal clock aei_sys_clk / 20 enabled
0	23:9, 27:26, 31:30	r	**Reserved** Read as zero, shall be written as zero.

28.4.9.7 Register GTM_IRQ_EN

GTM Interrupt Enable Register

IRQ_EN

GTM Interrupt Enable Register **(000014_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0							**CLK_PER_ERR_IRQ_EN**	**CLK_EN_ERR_IRQ_EN**	**AEIM_USP_BE_IRQ_EN**	**AEIM_IM_ADDR_IRQ_EN**	**AEIM_USP_ADDR_IRQ_E**	**AEI_USP_BE_IRQ_EN**	**AEI_IM_ADDR_IRQ_EN**	**AEI_USP_ADDR_IRQ_EN**	**AEI_TO_XPT_IRQ_EN**
r							rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
AEI_TO_XPT_IRQ_EN	0	rw	**AEI_TO_XPT_IRQ interrupt enable** 0_B Disable interrupt, interrupt is not visible outside GTM 1_B Enable interrupt, interrupt is visible outside GTM
AEI_USP_ADDR_IRQ_EN	1	rw	**AEI_USP_ADDR_IRQ interrupt enable** 0_B Disable interrupt, interrupt is not visible outside GTM 1_B Enable interrupt, interrupt is visible outside GTM
AEI_IM_ADDR_IRQ_EN	2	rw	**AEI_IM_ADDR_IRQ interrupt enable** 0_B Disable interrupt, interrupt is not visible outside GTM 1_B Enable interrupt, interrupt is visible outside GTM
AEI_USP_BE_IRQ_EN	3	rw	**AEI_USP_BE_IRQ interrupt enable** 0_B Disable interrupt, interrupt is not visible outside GTM 1_B Enable interrupt, interrupt is visible outside GTM
AEIM_USP_ADDR_IRQ_EN	4	rw	**AEI_MUSP_ADDR_IRQ interrupt enable** 0_B Disable interrupt, interrupt is not visible outside GTM 1_B Enable interrupt, interrupt is visible outside GTM

Field	Bits	Type	Description
AEIM_IM_ADDR_IRQ_EN	5	rw	**AEIM_IM_ADDR_IRQ interrupt enable** 0_B Disable interrupt, interrupt is not visible outside GTM 1_B Enable interrupt, interrupt is visible outside GTM
AEIM_USP_BE_IRQ_EN	6	rw	**AEIM_USP_BE_IRQ interrupt enable** 0_B Disable interrupt, interrupt is not visible outside GTM 1_B Enable interrupt, interrupt is visible outside GTM
CLK_EN_ERR_IRQ_EN	7	rw	**CLK_EN_ERR_IRQ interrupt enable** Read as zero in case of INT_CLK_EN_GEN = 0b1. 0_B Disable interrupt, interrupt is not visible outside GTM 1_B Enable interrupt, interrupt is visible outside GTM
CLK_PER_ERR_IRQ_EN	8	rw	**CLK_PER_ERR_IRQ interrupt enable** Read as zero in case of INT_CLK_EN_GEN = 0b1. 0_B Disable interrupt, interrupt is not visible outside GTM 1_B Enable interrupt, interrupt is visible outside GTM
0	31:9	r	**Reserved** Read as zero, shall be written as zero.

28.4.9.8 Register GTM_IRQ_FORCINT

GTM Software Interrupt Generation Register

IRQ_FORCINT

GTM Software Interrupt Generation Register (000018$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0							**TRG_CLK_PER_ERR**	**TRG_CLK_EN_ERR**	**TRG_AEIM_USP_BE**	**TRG_AEIM_IM_ADDR**	**TRG_AEIM_USP_ADDR**	**TRG_AEI_USP_BE**	**TRG_AEI_IM_ADDR**	**TRG_AEI_USP_ADDR**	**TRG_AEI_TO_XPT**
r							rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
TRG_AEI_TO_XPT	0	rw	**Trigger AEI_TO_XPT_IRQ interrupt by software** This bit is cleared automatically after write. This bit is write protected by bit RF_PROT of **CTRL** 0_B No interrupt triggering 1_B Assert AEI_TO_XPT_IRQ interrupt for one clock cycle
TRG_AEI_USP_ADDR	1	rw	**Trigger AEI_USP_ADDR_IRQ interrupt by software** This bit is cleared automatically after write. This bit is write protected by bit RF_PROT of **CTRL** 0_B No interrupt triggering 1_B Assert AEI_USP_ADDR_IRQ interrupt for one clock cycle

Field	Bits	Type	Description
TRG_AEI_IM_ADDR	2	rw	**Trigger AEI_IM_ADDR_IRQ interrupt by software** This bit is cleared automatically after write. This bit is write protected by bit RF_PROT of **CTRL** 0_B No interrupt triggering 1_B Assert AEI_IM_ADDR_IRQ interrupt for one clock cycle
TRG_AEI_USP_BE	3	rw	**Trigger AEI_USP_BE_IRQ interrupt by software** This bit is cleared automatically after write. This bit is write protected by bit RF_PROT of **CTRL** 0_B No interrupt triggering 1_B Assert AEI_USP_BE_IRQ interrupt for one clock cycle
TRG_AEIM_USP_ADDR	4	rw	**Trigger AEIM_USP_ADDR_IRQ interrupt by software** This bit is cleared automatically after write. This bit is write protected by bit RF_PROT of **CTRL** 0_B No interrupt triggering 1_B Assert AEIM_USP_ADDR_IRQ interrupt for one clock cycle
TRG_AEIM_IM_ADDR	5	rw	**Trigger AEIM_IM_ADDR_IRQ interrupt by software** This bit is cleared automatically after write. This bit is write protected by bit RF_PROT of **CTRL** 0_B No interrupt triggering 1_B Assert AEIM_IM_ADDR_IRQ interrupt for one clock cycle
TRG_AEIM_USP_BE	6	rw	**Trigger AEIM_USP_BE_IRQ interrupt by software** This bit is cleared automatically after write. This bit is write protected by bit RF_PROT of **CTRL** 0_B No interrupt triggering 1_B Assert AEIM_USP_BE_IRQ interrupt for one clock cycle
TRG_CLK_EN_ERR	7	rw	**Trigger CLK_EN_ERR_IRQ interrupt by software** This bit is cleared automatically after write. This bit is write protected by bit RF_PROT of **CTRL** Read as zero in case of INT_CLK_EN_GEN = 0b1. 0_B No interrupt triggering 1_B Assert CLK_EN_ERR_IRQ interrupt for one clock cycle
TRG_CLK_PER_ERR	8	rw	**Trigger CLK_PER_ERR_IRQ interrupt by software** This bit is cleared automatically after write. This bit is write protected by bit RF_PROT of **CTRL** Read as zero in case of INT_CLK_EN_GEN = 0b1. 0_B No interrupt triggering 1_B Assert CLK_PER_ERR_IRQ interrupt for one clock cycle
0	31:9	r	**Reserved** Read as zero, shall be written as zero.

28.4.9.9 Register GTM_IRQ_MODE

GTM Top Level Interrupts Mode Selection Register

IRQ_MODE
GTM Top Level Interrupts Mode Selection Register(00001C$_H$) **Application Reset Value: 0000 0000$_H$**

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="16">0</td></tr>
<tr><td colspan="16">r</td></tr>
</table>

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="14">0</td><td colspan="2">IRQ_MODE</td></tr>
<tr><td colspan="14">r</td><td colspan="2">rw</td></tr>
</table>

Field	Bits	Type	Description
IRQ_MODE	1:0	rw	**Interrupt strategy mode selection for the AEI timeout and address monitoring interrupts** The interrupt modes are described in **Section 28.4.5**.Note: This mode selection is only valid for the six interrupts described in section **Register GTM_IRQ_NOTIFY** 00$_B$ Level mode 01$_B$ Pulse mode 10$_B$ Pulse-Notify mode 11$_B$ Single-Pulse mode
0	31:2	r	**Reserved** Read as zero, shall be written as zero.

28.4.9.10 Register GTM_BRIDGE_MODE

GTM AEI Bridge Mode Register

Note: *All writable bits are write protected by bit BRIDGE_MODE_WRDIS*

BRIDGE_MODE
GTM AEI Bridge Mode Register **(000030$_H$)** **Application Reset Value: 0200 1001$_H$**

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="8">BUFF_DPT</td><td colspan="7">0</td><td>BRG_RST</td></tr>
<tr><td colspan="8">r</td><td colspan="7">r</td><td>rw</td></tr>
</table>

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="3">0</td><td>SYNC_INPUT_REG</td><td colspan="2">0</td><td>BUFF_OVL</td><td>MODE_UP_PGR</td><td colspan="5">0</td><td>BYPASS_SYNC</td><td>MSK_WR_RSP</td><td>BRG_MODE</td></tr>
<tr><td colspan="3">r</td><td>r</td><td colspan="2">r</td><td>r</td><td>r</td><td colspan="5">r</td><td>rw</td><td>rw</td><td>rw</td></tr>
</table>

Field	Bits	Type	Description
BRG_MODE	0	rw	**Defines the operation mode for the AEI bridge** Reset value depends on the hardware configuration chosen by silicon vendor. BRG_MODE shall not be written with 0. 0_B AEI bridge operates in sync_bridge mode 1_B AEI bridge operates in async_bridge mode
MSK_WR_RSP	1	rw	**Mask write response** With active write buffer MSK_WR_RSP=1, execution of actions can be delayed due to previous inserted write actions in the transaction buffer which wait to be serviced. This can lead to the fact that an access on the bus to a different peripheral than the GTM might be executed earlier in time than the write access buffered in the GTM. Applications must be setup up with this in mind otherwise unexpected operation can happen. 0_B Do not mask the write response. Depending on the selected address the latency for execution can vary due to GTM internal arbitration. After this time the status of the access will be signaled by the signal AEI_STATUS to the bus interface. 1_B Mask write response. The write buffer of the bridge is activated, the actual access will be stored to the write buffer, and without latency on the bus interface; the acceptance of the access is signaled. AEI_STATUS=0b00 will be signaled. In case of a full write buffer, the actual access will be postponed until the next write buffer entry becomes free. Note: The status of the executed write accesses can be observed by using the notify bits AEI_USP_ADDR ,AEI_IM_ADDR, AEI_USP_BE in the register GTM_IRQ_NOTIFY.
BYPASS_SYNC	2	rw	**Bypass synchronizer flipflops** Function only available with BRG_MODE=1 0_B Synchronizer flip-flops in use, latency increase due to synchronization (aei_clk -> aei_sys_clk and back aei_sys_clk -> aei_clk). This setting must be used if aei_clk and aei_sys_clk operate fully asynchronous by independent clock sources. 1_B Synchronizer flip-flops are bypassed. No additional latency due to synchronization. This setting can be used if aei_clk and aei_sys_clk are generated by clock gating or clock division out of a common clock source. Clock edges on aei_clk and aei_sys_clk generated out of the same clock edge of the common clock source must have zero skew.
MODE_UP_PGR	8	r	**Mode update in progress** 0_B No update in progress 1_B Update in progress
BUFF_OVL	9	r	**Buffer overflow register** 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). 0_B No buffer overflow occurred 1_B Buffer overflow occurred

Field	Bits	Type	Description
SYNC_INPUT_REG	12	r	**Additional pipelined stage in synchronous bridge mode** Reset value depends on the hardware configuration chosen by silicon vendor. 0_B No additional pipelined stage implemented 1_B Additional pipelined stage implemented. All accesses in synchronous mode will be increased by one clock cycle.
BRG_RST	16	rw	**Bridge software reset** This bit is cleared automatically after write. 0_B No bridge reset request 1_B Bridge reset request
BUFF_DPT	31:24	r	**Buffer depth of AEI bridge** Signals the buffer depth of the GTM AEI bridge implementation. Reset value depends on the hardware configuration chosen by silicon vendor.
0	7:3, 11:10, 15:13, 23:17	r	**Reserved** Read as zero, shall be written as zero.

28.4.9.11 Register GTM_BRIDGE_PTR1

GTM AEI Bridge Pointer 1 Register

Note: This register operates on the AEI_CLK domain.

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

BRIDGE_PTR1

GTM AEI Bridge Pointer 1 Register **(000034_H)** **Application Reset Value: $0020\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSP_TRAN_RDY						FBC						ABT_TRAN_PGR			
r						r						r			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABT_TRAN_PGR	TRAN_IN_PGR					FIRST_RSP_PTR					NEW_TRAN_PTR				
r	r					r					r				

Field	Bits	Type	Description
NEW_TRAN_PTR	4:0	r	**New transaction pointer** Signals the actual value of the new transaction pointer.

Field	Bits	Type	Description
FIRST_RSP_PTR	9:5	r	**First response pointer** Signals the actual value of first response pointer.
TRAN_IN_PGR	14:10	r	**Transaction in progress pointer (acquire)** Transaction in progress pointer.
ABT_TRAN_PGR	19:15	r	**Aborted transaction in progress pointer** Aborted transaction in progress pointer.
FBC	25:20	r	**Free buffer count** Number of free buffer entries. Initial value depends on the hardware configuration chosen by silicon vendor. (see BUFF_DPT in GTM_BRIDGE_MODE register).
RSP_TRAN_RDY	31:26	r	**Response transactions ready.** Amount of ready response transactions.

28.4.9.12 Register GTM_BRIDGE_PTR2

GTM AEI Bridge Pointer 2 Register

Note: This register operates on the GTM_CLK domain.

BRIDGE_PTR2

GTM AEI Bridge Pointer 2 Register **(000038_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0											**TRAN_IN_PGR2**				
r											r				

Field	Bits	Type	Description
TRAN_IN_PGR2	4:0	r	**Transaction in progress pointer (aquire2)** Transaction in progress pointer 2.
0	31:5	r	**Reserved** Read as zero, shall be written as zero.

28.4.9.13 Register GTM_MCS_AEM_DIS

GTM MCS Master Port Disable Register

MCS_AEM_DIS

GTM MCS Master Port Disable Register **($00003C_H$)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								**DIS_CLS11**		**DIS_CLS10**		**DIS_CLS9**		**DIS_CLS8**	
r								rw		rw		rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIS_CLS7		**DIS_CLS6**		**DIS_CLS5**		**DIS_CLS4**		**DIS_CLS3**		**DIS_CLS2**		**DIS_CLS1**		**DIS_CLS0**	
rw		rw		rw		rw		rw		rw		rw		rw	

Field	Bits	Type	Description
DIS_CLS0	1:0	rw	**Disable MCS AEIM access in cluster 0** Multicore encoding in use (**DIS_CLSx(1)** defines the state of the signal) Any read access to a **DIS_CLSx** bit field will always result in a value 00 or 11 indicating current state. A modification of the state is only performed with the values 01 and 10. Writing the values 00 and 11 is always ignored. 00_B State is 0; MCS AEM access in cluster x enabled (ignore write access) 01_B Change state to 0 10_B Change state to 1 11_B State is 1; MCS AEM access in cluster x disabled (ignore write access)
DIS_CLS1	3:2	rw	**Disable MCS AEIM access in cluster 1, see bit DIS_CLS0**
DIS_CLS2	5:4	rw	**Disable MCS AEIM access in cluster 2, see bit DIS_CLS0**
DIS_CLS3	7:6	rw	**Disable MCS AEIM access in cluster 3, see bit DIS_CLS0**
DIS_CLS4	9:8	rw	**Disable MCS AEIM access in cluster 4, see bit DIS_CLS0**
DIS_CLS5	11:10	rw	**Disable MCS AEIM access in cluster 5, see bit DIS_CLS0**
DIS_CLS6	13:12	rw	**Disable MCS AEIM access in cluster 6, see bit DIS_CLS0**
DIS_CLS7	15:14	rw	**Disable MCS AEIM access in cluster 7, see bit DIS_CLS0**
DIS_CLS8	17:16	rw	**Disable MCS AEIM access in cluster 8, see bit DIS_CLS0**
DIS_CLS9	19:18	rw	**Disable MCS AEIM access in cluster 9, see bit DIS_CLS0**
DIS_CLS10	21:20	rw	**Disable MCS AEIM access in cluster 10, see bit DIS_CLS0**
DIS_CLS11	23:22	rw	**Disable MCS AEIM access in cluster 11, see bit DIS_CLS0**
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

28.4.9.14 Register GTM_EIRQ_EN

GTM Error Interrupt Enable Register

EIRQ_EN

GTM Error Interrupt Enable Register **(000020$_H$)** **Application Reset Value: 0000 0180$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0							**CLK_PER_ERR_EIRQ_EN**	**CLK_EN_ERR_EIRQ_EN**	**AEIM_USP_BE_EIRQ_EN**	**AEIM_IM_ADDR_EIRQ_E**	**AEIM_USP_ADDR_EIRQ_E**	**AEI_USP_BE_EIRQ_EN**	**AEI_IM_ADDR_EIRQ_E**	**AEI_USP_ADDR_EIRQ_E**	**AEI_TO_XPT_EIRQ_EN**
r							rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
AEI_TO_XPT_EIRQ_EN	0	rw	**AEI_TO_XPT_EIRQ error interrupt enable** 0$_B$ Disable error interrupt, interrupt is not visible outside GTM 1$_B$ Enable error interrupt, interrupt is visible outside GTM
AEI_USP_ADDR_EIRQ_EN	1	rw	**AEI_USP_ADDR_EIRQ error interrupt enable** 0$_B$ Disable error interrupt, interrupt is not visible outside GTM 1$_B$ Enable error interrupt, interrupt is visible outside GTM
AEI_IM_ADDR_EIRQ_EN	2	rw	**AEI_IM_ADDR_EIRQ error interrupt enable** 0$_B$ Disable error interrupt, interrupt is not visible outside GTM 1$_B$ Enable error interrupt, interrupt is visible outside GTM
AEI_USP_BE_EIRQ_EN	3	rw	**AEI_USP_BE_EIRQ error interrupt enable** 0$_B$ Disable error interrupt, interrupt is not visible outside GTM 1$_B$ Enable error interrupt, interrupt is visible outside GTM
AEIM_USP_ADDR_EIRQ_EN	4	rw	**AEIM_USP_ADDR_EIRQ error interrupt enable** 0$_B$ Disable error interrupt, interrupt is not visible outside GTM 1$_B$ Enable error interrupt, interrupt is visible outside GTM
AEIM_IM_ADDR_EIRQ_EN	5	rw	**AEIM_IM_ADDR_EIRQ error interrupt enable** 0$_B$ Disable error interrupt, interrupt is not visible outside GTM 1$_B$ Enable error interrupt, interrupt is visible outside GTM
AEIM_USP_BE_EIRQ_EN	6	rw	**AEIM_USP_BE_EIRQ error interrupt enable** 0$_B$ Disable error interrupt, interrupt is not visible outside GTM 1$_B$ Enable error interrupt, interrupt is visible outside GTM
CLK_EN_ERR_EIRQ_EN	7	rw	**CLK_EN_ERR_EIRQ interrupt enable** Read as zero in case of INT_CLK_EN_GEN = 0b1. 0$_B$ Disable error interrupt, interrupt is not visible outside GTM 1$_B$ Enable error interrupt, interrupt is visible outside GTM

Field	Bits	Type	Description
CLK_PER_ERR_EIRQ_EN	8	rw	**CLK_PER_ERR_EIRQ interrupt enable** Read as zero in case of INT_CLK_EN_GEN = 0b1. Read as zero, shall be written as zero. 0_B Disable error interrupt, interrupt is not visible outside GTM 1_B Enable error interrupt, interrupt is visible outside GTM
0	31:9	r	**Reserved** Read as zero, shall be written as zero.

28.4.9.15 Register GTM_CLS_CLK_CFG

GTM Cluster Clock Configuration

Note: For clusters greater than 4 (only MAX 100 MHz capable), the allowed setting for the CLS_CLK_DIV are 00_B and 10_B (clock divider 2). For clusters < 5, 200 MHz is available. In case a device has a single 100 MHz cluster, the ARU will run with 100 MHz.

Note: Writing a value to a bit field CLS[c]_CLK_DIV that is not available in the device, an AEI status 10_B is returned.

Note: The availability of configuration bits is indicated by value of bit CFG_CLOCK_RATE in register CCM[c]_HW_CFG. If CFG_CLOCK_RATE=0, only the values 00_B and 01_B are valid for bit fields CLS[c]_CLK_DIV.

CLS_CLK_CFG
GTM Cluster Clock Configuration (0000B0$_H$) Application Reset Value: 00AA AAAA$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								**CLS11_CLK_DIV**		**CLS10_CLK_DIV**		**CLS9_CLK_DIV**		**CLS8_CLK_DIV**	
r								rw		rw		rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLS7_CLK_DIV		**CLS6_CLK_DIV**		**CLS5_CLK_DIV**		**CLS4_CLK_DIV**		**CLS3_CLK_DIV**		**CLS2_CLK_DIV**		**CLS1_CLK_DIV**		**CLS0_CLK_DIV**	
rw		rw		rw		rw		rw		rw		rw		rw	

Field	Bits	Type	Description
CLSc_CLK_DIV (c=0-11)	2*c+1:2*c	rw	**Cluster c Clock Divider** This bit is only writable if bit field **RF_PROT** of register **GTM_CTRL** is cleared. 00_B Cluster c is disabled 01_B Cluster c is enabled without clock divider 10_B Cluster c is enabled with clock divider 11_B Reserved, do not use.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

28.4.9.16 Register GTM_CFG

GTM Configuration Register

CFG

GTM Configuration Register **(000028$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															**SRC_IN_MUX**
r															rw

Field	Bits	Type	Description
SRC_IN_MUX	0	rw	**GTM_TIM[i]_AUX_IN input source selection** See **Figure 9** for details. 0_B Use for TIM[i] output of TOM[n] 1_B Use for TIM[i] output of TOM[i] (same cluster)
0	31:1	r	**Reserved** Read as zero, shall be written as zero.