XC2000 Derivatives

16/32-Bit Single-Chip Microcontroller with 32-Bit Performance

Volume 2 (of 2): Peripheral Units

Microcontrollers

Never stop thinking

Edition 2007-06

Published by
Infineon Technologies AG
81726 Munich, Germany

Legal Disclaimer

The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics. With respect to any examples or hints given herein, any typical values stated herein and/or any information regarding the application of the device, Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind, including without limitation, warranties of non-infringement of intellectual property rights of any third party.

Information

For further information on technology, delivery terms and conditions and prices, please contact the nearest Infineon Technologies Office (**www.infineon.com**).

Warnings

Due to technical requirements, components may contain dangerous substances. For information on the types in question, please contact the nearest Infineon Technologies Office.

Infineon Technologies components may be used in life-support devices or systems only with the express written approval of Infineon Technologies, if a failure of such components can reasonably be expected to cause the failure of that life-support device or system or to affect the safety or effectiveness of that device or system. Life support devices or systems are intended to be implanted in the human body or to support and/or maintain and sustain and/or protect human life. If they fail, it is reasonable to assume that the health of the user or other persons may be endangered.

XC2000 Derivatives

16/32-Bit Single-Chip Microcontroller with 32-Bit Performance

Volume 2 (of 2): Peripheral Units

Microcontrollers

Never stop thinking

Preliminary

XC2xxx
Revision History: V1.0, 2007-06

Previous Version(s):
V0.1, 2007-03, Draft Version

Page	Subjects (major changes since last revision)

We Listen to Your Comments

Any information within this document that you feel is wrong, unclear or missing at all? Your feedback will help us to continuously improve the quality of this document. Please send your proposal (including a reference to this document) to:

mcdocu.comments@infineon.com

Summary Of Chapters

This User's Manual consists of two Volumes, "System Units" and "Peripheral Units". For a quick overview this table of chapters summarizes both volumes, so you immediately can find the reference to the desired section in the corresponding document ([1] or [2]).

Table Of Contents

This User's Manual consists of two Volumes, "System Units" and "Peripheral Units". For your convenience this table of contents (and also the keyword and register index) lists both volumes, so you can immediately find the reference to the desired section in the corresponding document ([1] or [2]).

14 The General Purpose Timer Units

The General Purpose Timer Unit blocks GPT1 and GPT2 have very flexible multifunctional timer structures which may be used for timing, event counting, pulse width measurement, pulse generation, frequency multiplication, and other purposes. They incorporate five 16-bit timers that are grouped into the two timer blocks GPT1 and GPT2. Each timer in each block may operate independently in a number of different modes such as gated timer or counter mode, or may be concatenated with another timer of the same block. Each block has alternate input/output functions and specific interrupts associated with it.

Block GPT1 contains three timers/counters: The core timer T3 and the two auxiliary timers T2 and T4. The maximum resolution is $f_{GPT}/4$. The auxiliary timers of GPT1 may optionally be configured as reload or capture registers for the core timer. These registers are listed in **Section 14.1.6**.

- $f_{GPT}/4$ maximum resolution
- 3 independent timers/counters
- Timers/counters can be concatenated
- 4 operating modes:
  - Timer Mode
  - Gated Timer Mode
  - Counter Mode
  - Incremental Interface Mode
- Reload and Capture functionality
- Separate interrupt lines

Block GPT2 contains two timers/counters: The core timer T6 and the auxiliary timer T5. The maximum resolution is $f_{GPT}/2$. An additional Capture/Reload register (CAPREL) supports capture and reload operation with extended functionality. These registers are listed in **Section 14.2.7**. The core timer T6 may be concatenated with timers of the CAPCOM units (T0, T1, T7, and T8).

The following list summarizes the features which are supported:

- $f_{GPT}/2$ maximum resolution
- 2 independent timers/counters
- Timers/counters can be concatenated
- 3 operating modes:
  - Timer Mode
  - Gated Timer Mode
  - Counter Mode
- Extended capture/reload functions via 16-bit capture/reload register CAPREL
- Separate interrupt lines

14.1 Timer Block GPT1

From a programmer's point of view, the GPT1 block is composed of a set of SFRs as summarized below. Those portions of port and direction registers which are used for alternate functions by the GPT1 block are shaded.

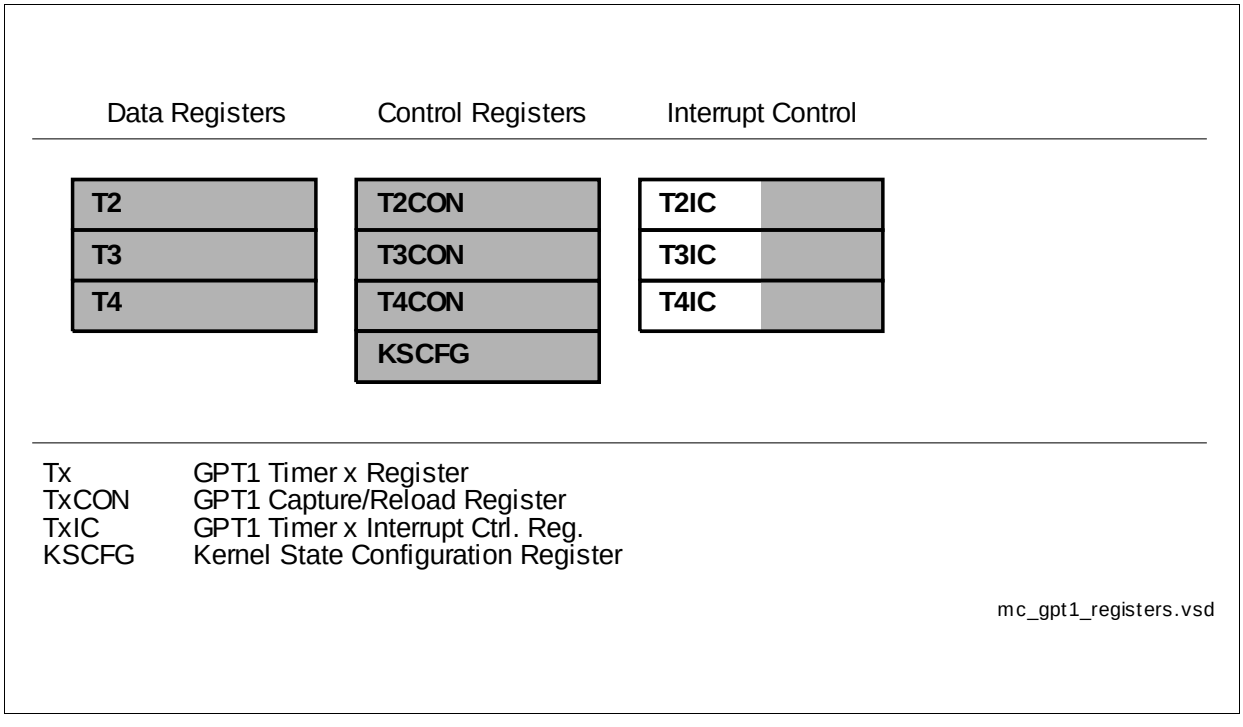


Figure 14-1 SFRs Associated with Timer Block GPT1

All three timers of block GPT1 (T2, T3, T4) can run in one of 4 basic modes: Timer Mode, Gated Timer Mode, Counter Mode, or Incremental Interface Mode. All timers can count up or down. Each timer of GPT1 is controlled by a separate control register TxCON.

Each timer has an input pin TxIN (alternate pin function) associated with it, which serves as the gate control in gated timer mode, or as the count input in counter mode. The count direction (up/down) may be programmed via software or may be dynamically altered by a signal at the External Up/Down control input TxEUD (alternate pin function). An overflow/underflow of core timer T3 is indicated by the Output Toggle Latch T3OTL, whose state may be output on the associated pin T3OUT (alternate pin function). The auxiliary timers T2 and T4 may additionally be concatenated with the core timer T3 (through T3OTL) or may be used as capture or reload registers for the core timer T3.

The current contents of each timer can be read or modified by the CPU by accessing the corresponding timer count registers T2, T3, or T4, located in the non-bitaddressable SFR space (see **Section 14.1.6**). When any of the timer registers is written to by the CPU in the state immediately preceding a timer increment, decrement, reload, or capture operation, the CPU write operation has priority in order to guarantee correct results.

The interrupts of GPT1 are controlled through the Interrupt Control Registers TxIC. These registers are not part of the GPT1 block. The input and output lines of GPT1 are connected to pins of ports P3 and P5. The control registers for the port functions are located in the respective port modules.

*Note: The timing requirements for external input signals can be found in **Section 14.1.5**, **Section 14.3** summarizes the module interface signals, including pins.*

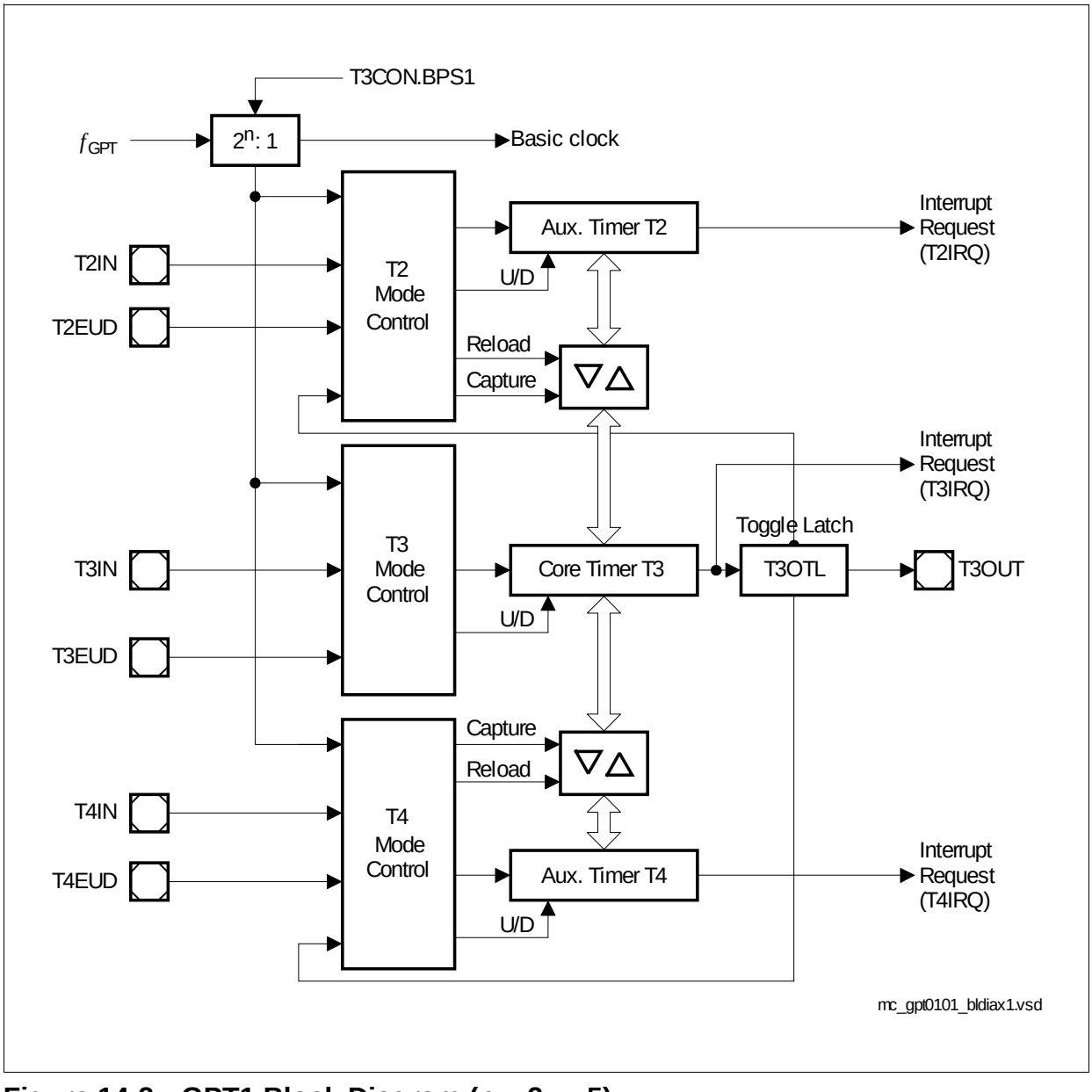


Figure 14-2 GPT1 Block Diagram (n = 2 … 5)

14.1.1 GPT1 Core Timer T3 Control

The current contents of the core timer T3 are reflected by its count register T3. This register can also be written to by the CPU, for example, to set the initial start value.

The core timer T3 is configured and controlled via its bitaddressable control register T3CON.

GPT12E_T3CON
Timer 3 Control Register **SFR ($FF42_H/A1_H$)** **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T3 R DIR	**T3 CH DIR**	**T3 ED GE**	**BPS1**		**T3 OTL**	**T3 OE**	**T3 UDE**	**T3 UD**	**T3R**	**T3M**			**T3I**		
rh	rwh	rwh	rw		rwh	rw	rw	rw	rw	rw			rw		

Field	Bits	Type	Description
T3RDIR	15	rh	**Timer T3 Rotation Direction Flag** 0 Timer T3 counts up 1 Timer T3 counts down
T3CHDIR	14	rwh	**Timer T3 Count Direction Change Flag** This bit is set each time the count direction of timer T3 changes. T3CHDIR must be cleared by SW. 0 No change of count direction was detected 1 A change of count direction was detected
T3EDGE	13	rwh	**Timer T3 Edge Detection Flag** The bit is set each time a count edge is detected. T3EDGE must be cleared by SW. 0 No count edge was detected 1 A count edge was detected
BPS1	[12:11]	rw	**GPT1 Block Prescaler Control** Selects the basic clock for block GPT1 (see also **Section 14.1.5**) 00 $f_{GPT}/8$ 01 $f_{GPT}/4$ 10 $f_{GPT}/32$ 11 $f_{GPT}/16$

Field	Bits	Type	Description
T3OTL	10	rwh	**Timer T3 Overflow Toggle Latch** Toggles on each overflow/underflow of T3. Can be set or reset by software (see separate description)
T3OE	9	rw	**Overflow/Underflow Output Enable** 0 Alternate Output Function Disabled 1 State of T3 toggle latch is output on pin T3OUT
T3UDE	8	rw	**Timer T3 External Up/Down Enable**[1)] 0 Input T3EUD is disconnected 1 Direction influenced by input T3EUD
T3UD	7	rw	**Timer T3 Up/Down Control**[1)] 0 Timer T3 counts up 1 Timer T3 counts down
T3R	6	rw	**Timer T3 Run Bit** 0 Timer T3 stops 1 Timer T3 runs
T3M	[5:3]	rw	**Timer T3 Mode Control** (Basic Operating Mode) 000 Timer Mode 001 Counter Mode 010 Gated Timer Mode with gate active low 011 Gated Timer Mode with gate active high 100 Reserved. Do not use this combination. 101 Reserved. Do not use this combination. 110 Incremental Interface Mode (Rotation Detection Mode) 111 Incremental Interface Mode (Edge Detection Mode)
T3I	[2:0]	rw	**Timer T3 Input Parameter Selection** Depends on the operating mode, see respective sections for encoding: **Table 14-7** for Timer Mode and Gated Timer Mode **Table 14-2** for Counter Mode **Table 14-3** for Incremental Interface Mode

1) See **Table 14-1** for encoding of bits T3UD and T3UDE.

Timer T3 Run Control

The core timer T3 can be started or stopped by software through bit T3R (Timer T3 Run Bit). This bit is relevant in all operating modes of T3. Setting bit T3R will start the timer, clearing bit T3R stops the timer.

In gated timer mode, the timer will only run if T3R = 1 and the gate is active (high or low, as programmed).

Note: When bit T2RC or T4RC in timer control register T2CON or T4CON is set, bit T3R will also control (start and stop) the auxiliary timer(s) T2 and/or T4.

Count Direction Control

The count direction of the GPT1 timers (core timer and auxiliary timers) can be controlled either by software or by the external input pin TxEUD (Timer Tx External Up/Down Control Input). These options are selected by bits TxUD and TxUDE in the respective control register TxCON. When the up/down control is provided by software (bit TxUDE = 0), the count direction can be altered by setting or clearing bit TxUD. When bit TxUDE = 1, pin TxEUD is selected to be the controlling source of the count direction. However, bit TxUD can still be used to reverse the actual count direction, as shown in **Table 14-1**. The count direction can be changed regardless of whether or not the timer is running.

Note: When pin TxEUD is used as external count direction control input, it must be configured as input (its corresponding direction control bit must be cleared).

Table 14-1 GPT1 Timer Count Direction Control

Pin TxEUD	Bit TxUDE	Bit TxUD	Count Direction	Bit TxRDIR
X	0	0	Count Up	0
X	0	1	Count Down	1
0	1	0	Count Up	0
1	1	0	Count Down	1
0	1	1	Count Down	1
1	1	1	Count Up	0

Timer 3 Output Toggle Latch

The overflow/underflow signal of timer T3 is connected to a block named ‘Toggle Latch’, shown in the timer mode diagrams. **Figure 14-3** illustrates the details of this block. An overflow or underflow of T3 will clock two latches: The first latch represents bit T3OTL in control register T3CON. The second latch is an internal latch toggled by T3OTL’s output. Both latch outputs are connected to the input control blocks of the auxiliary timers T2 and T4. The output level of the shadow latch will match the output level of T3OTL, but is delayed by one clock cycle. When the T3OTL value changes, this will result in a temporarily different output level from T3OTL and the shadow latch, which can trigger the selected count event in T2 and/or T4.

When software writes to T3OTL, both latches are set or cleared simultaneously. In this case, both signals to the auxiliary timers carry the same level and no edge will be detected. Bit T3OE (overflow/underflow output enable) in register T3CON enables the state of T3OTL to be monitored via an external pin T3OUT. When T3OTL is linked to an external port pin (must be configured as output), T3OUT can be used to control external HW. If T3OE = 1, pin T3OUT outputs the state of T3OTL. If T3OE = 0, pin T3OUT outputs a high level (as long as the T3OUT alternate function is selected for the port pin).

The trigger signals can serve as an input for the counter function or as a trigger source for the reload function of the auxiliary timers T2 and T4.

As can be seen from **Figure 14-3**, when latch T3OTL is modified by software to determine the state of the output line, also the internal shadow latch is set or cleared accordingly. Therefore, no trigger condition is detected by T2/T4 in this case.

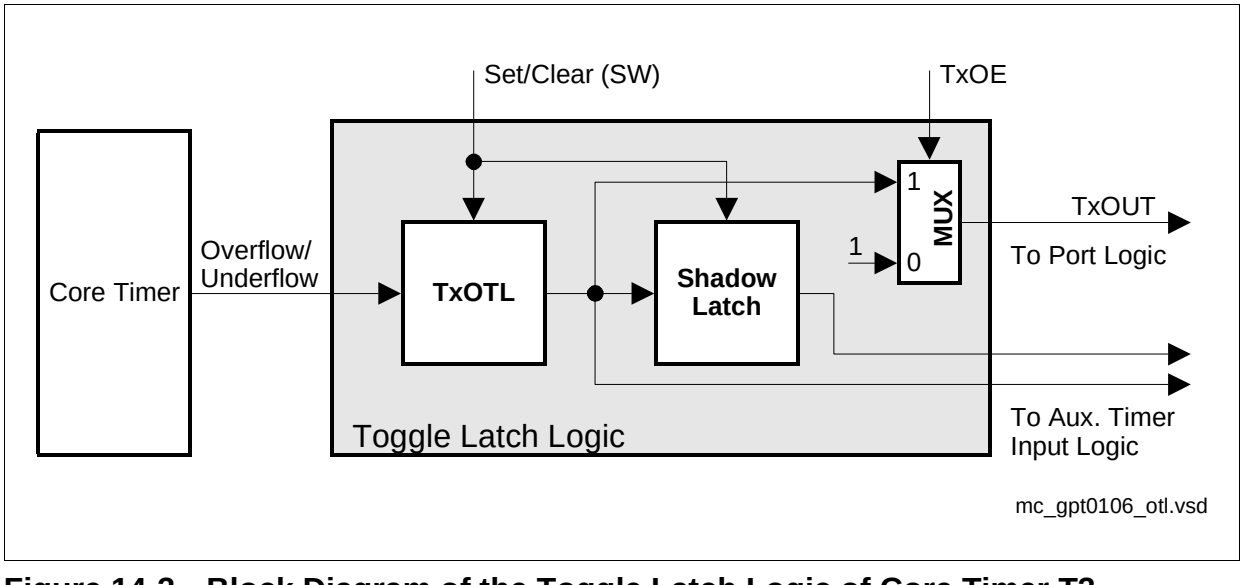


Figure 14-3 Block Diagram of the Toggle Latch Logic of Core Timer T3

14.1.2 GPT1 Core Timer T3 Operating Modes

Timer 3 in Timer Mode

Timer mode for the core timer T3 is selected by setting bitfield T3M in register T3CON to 000_B. In timer mode, T3 is clocked with the module's input clock f_{GPT} divided by two programmable prescalers controlled by bitfields BPS1 and T3I in register T3CON. Please see **Section 14.1.5** for details on the input clock options.

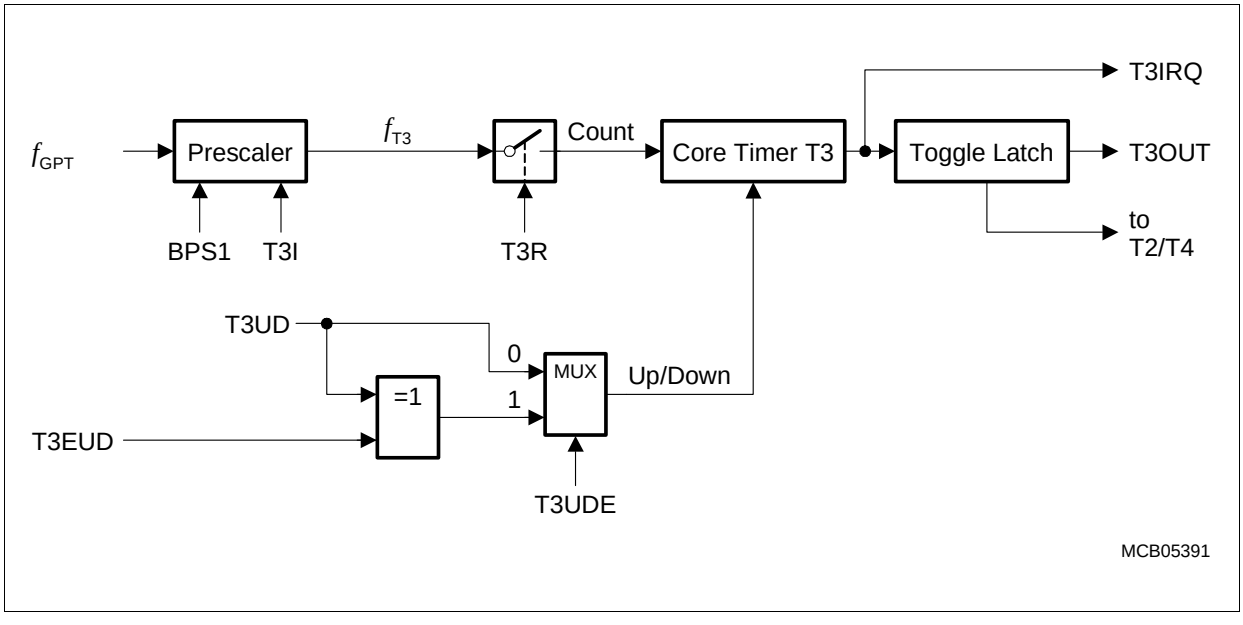


Figure 14-4 Block Diagram of Core Timer T3 in Timer Mode

Gated Timer Mode

Gated timer mode for the core timer T3 is selected by setting bitfield T3M in register T3CON to 010_B or 011_B. Bit T3M.0 (T3CON.3) selects the active level of the gate input. The same options for the input frequency are available in gated timer mode as in timer mode (see **Section 14.1.5**). However, the input clock to the timer in this mode is gated by the external input pin T3IN (Timer T3 External Input).
To enable this operation, the associated pin T3IN must be configured as input, that is, the corresponding direction control bit must contain 0.

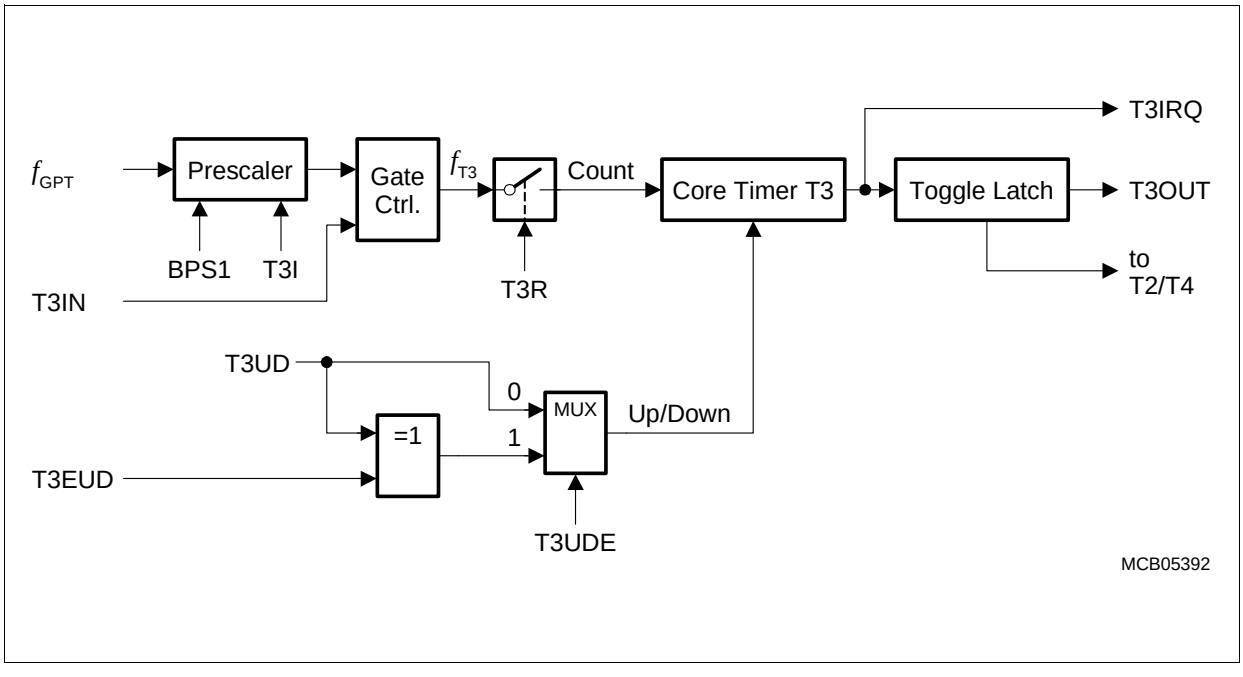


Figure 14-5 Block Diagram of Core Timer T3 in Gated Timer Mode

If T3M = 010_B, the timer is enabled when T3IN shows a low level. A high level at this line stops the timer. If T3M = 011_B, line T3IN must have a high level in order to enable the timer. Additionally, the timer can be turned on or off by software using bit T3R. The timer will only run if T3R is 1 and the gate is active. It will stop if either T3R is 0 or the gate is inactive.

Note: A transition of the gate signal at pin T3IN does not cause an interrupt request.

Counter Mode

Counter Mode for the core timer T3 is selected by setting bitfield T3M in register T3CON to 001_B. In counter mode, timer T3 is clocked by a transition at the external input pin T3IN. The event causing an increment or decrement of the timer can be a positive, a negative, or both a positive and a negative transition at this line. Bitfield T3I in control register T3CON selects the triggering transition (see **Table 14-2**).

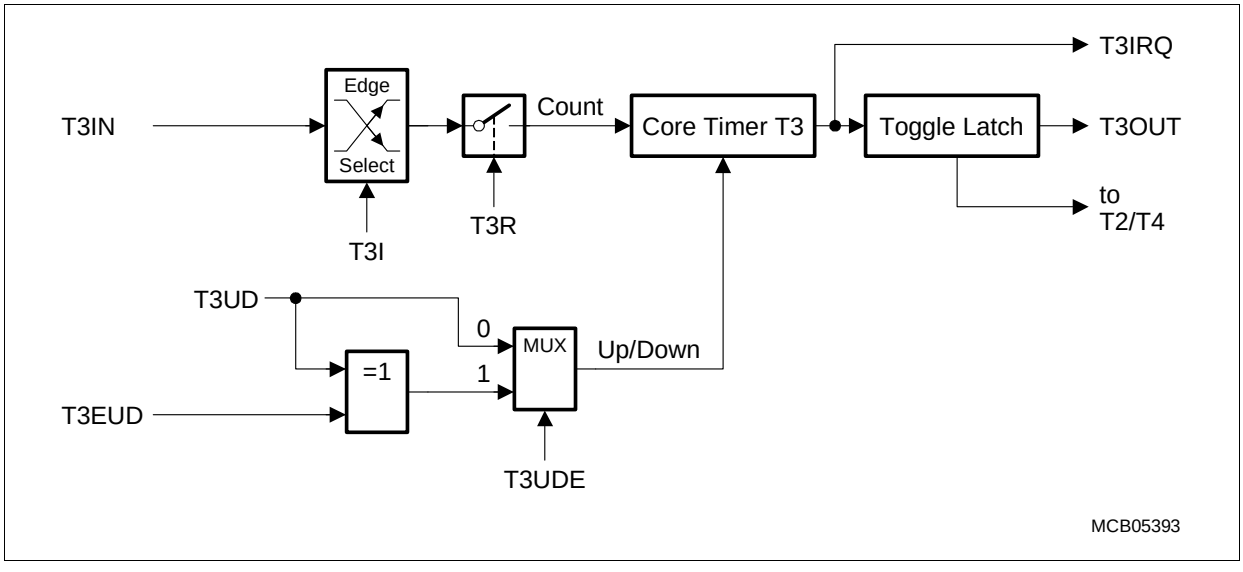


Figure 14-6 Block Diagram of Core Timer T3 in Counter Mode

Table 14-2 GPT1 Core Timer T3 (Counter Mode) Input Edge Selection

T3I	Triggering Edge for Counter Increment/Decrement
0 0 0	None. Counter T3 is disabled
0 0 1	Positive transition (rising edge) on T3IN
0 1 0	Negative transition (falling edge) on T3IN
0 1 1	Any transition (rising or falling edge) on T3IN
1 X X	Reserved. Do not use this combination

For counter mode operation, pin T3IN must be configured as input (the respective direction control bit DPx.y must be 0). The maximum input frequency allowed in counter mode depends on the selected prescaler value. To ensure that a transition of the count input signal applied to T3IN is recognized correctly, its level must be held high or low for a minimum number of module clock cycles before it changes. This information can be found in **Section 14.1.5**.

Incremental Interface Mode

Incremental interface mode for the core timer T3 is selected by setting bitfield T3M in register T3CON to 110_B or 111_B. In incremental interface mode, the two inputs associated with core timer T3 (T3IN, T3EUD) are used to interface to an incremental encoder. T3 is clocked by each transition on one or both of the external input pins to provide 2-fold or 4-fold resolution of the encoder input.

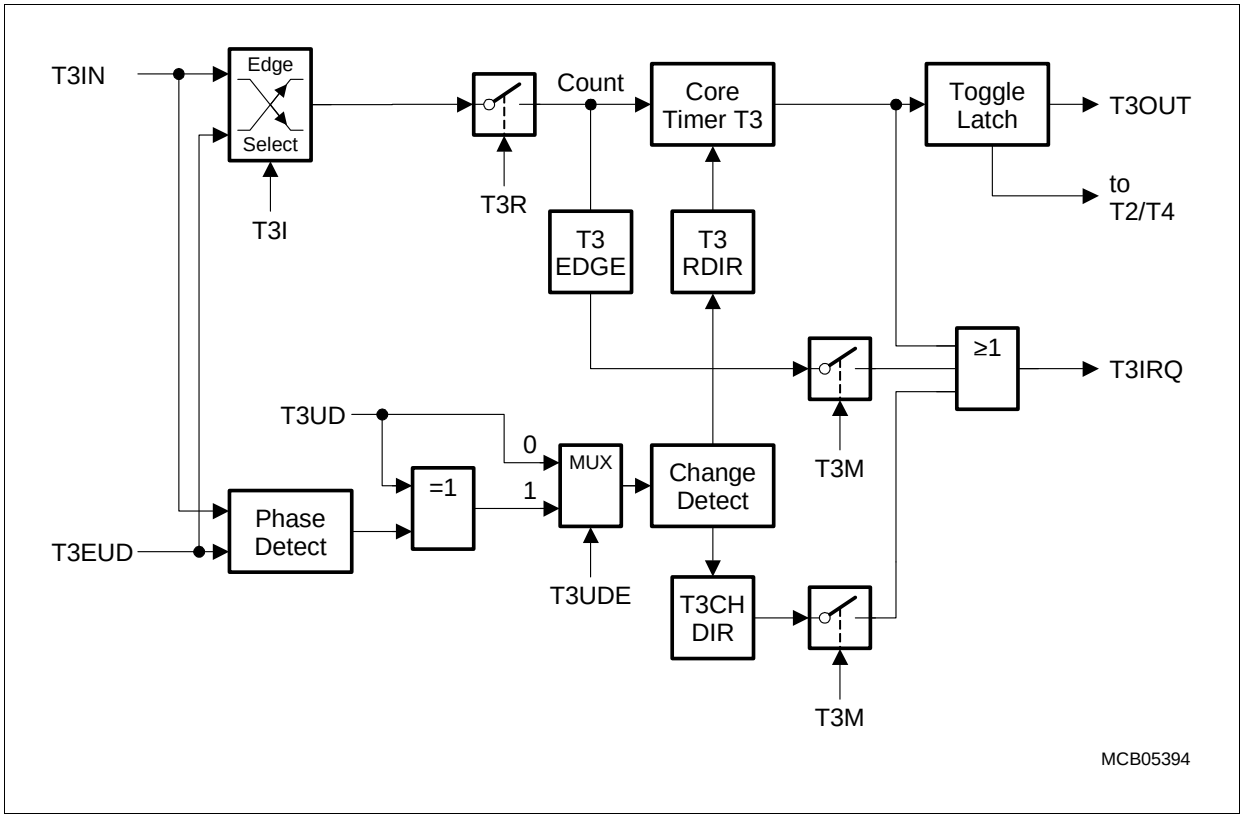


Figure 14-7 Block Diagram of Core Timer T3 in Incremental Interface Mode

Bitfield T3I in control register T3CON selects the triggering transitions (see **Table 14-3**). The sequence of the transitions of the two input signals is evaluated and generates count pulses as well as the direction signal. So T3 is modified automatically according to the speed and the direction of the incremental encoder and, therefore, its contents always represent the encoder's current position.

The interrupt request (T3IRQ) generation mode can be selected: In Rotation Detection Mode (T3M = 110_B), an interrupt request is generated each time the count direction of T3 changes. In Edge Detection Mode (T3M = 111_B), an interrupt request is generated each time a count edge for T3 is detected. Count direction, changes in the count direction, and count requests are monitored by status bits T3RDIR, T3CHDIR, and T3EDGE in register T3CON.

Table 14-3 Core Timer T3 (Incremental Interface Mode) Input Edge Selection

T3I	Triggering Edge for Counter Increment/Decrement
0 0 0	None. Counter T3 stops.
0 0 1	Any transition (rising or falling edge) on T3IN.
0 1 0	Any transition (rising or falling edge) on T3EUD.
0 1 1	Any transition (rising or falling edge) on any T3 input (T3IN or T3EUD).
1 X X	Reserved. Do not use this combination.

The incremental encoder can be connected directly to the XC2000 without external interface logic. In a standard system, however, comparators will be employed to convert the encoder's differential outputs (such as A, $\overline{A}$) to digital signals (such as A). This greatly increases noise immunity.

Note: The third encoder output T0, which indicates the mechanical zero position, may be connected to an external interrupt input and trigger a reset of timer T3 (for example via PEC transfer from ZEROS).

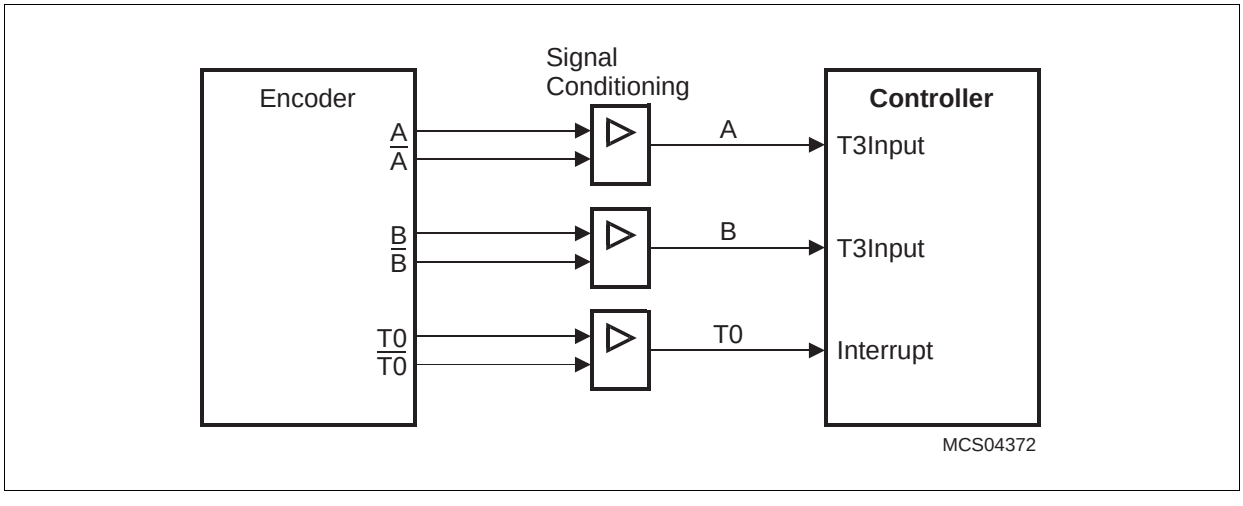


Figure 14-8 Connection of the Encoder to the XC2000

For incremental interface operation, the following conditions must be met:

- Bitfield T3M must be 110_B or 111_B.
- Both pins T3IN and T3EUD must be configured as input, i.e. the respective direction control bits must be 0.
- Bit T3UDE must be 1 to enable automatic external direction control.

The maximum count frequency allowed in incremental interface mode depends on the selected prescaler value. To ensure that a transition of any input signal is recognized correctly, its level must be held high or low for a minimum number of module clock cycles before it changes. This information can be found in **Section 14.1.5**.

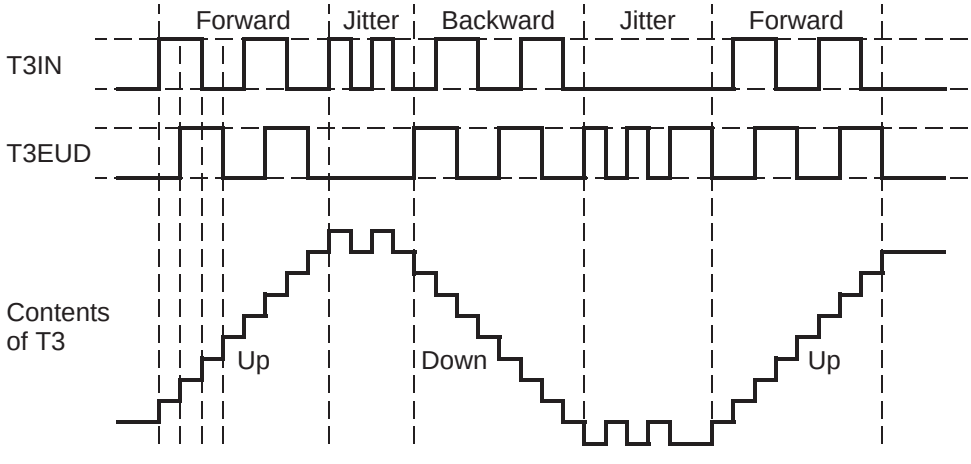
As in incremental interface mode two input signals with a 90° phase shift are evaluated, their maximum input frequency can be half the maximum count frequency.

In incremental interface mode, the count direction is automatically derived from the sequence in which the input signals change, which corresponds to the rotation direction of the connected sensor. **Table 14-4** summarizes the possible combinations.

Table 14-4 GPT1 Core Timer T3 (Incremental Interface Mode) Count Direction

Level on Respective other Input	T3IN Input		T3EUD Input	
	Rising ↑	**Falling ↓**	**Rising ↑**	**Falling ↓**
High	Down	Up	Up	Down
Low	Up	Down	Down	Up

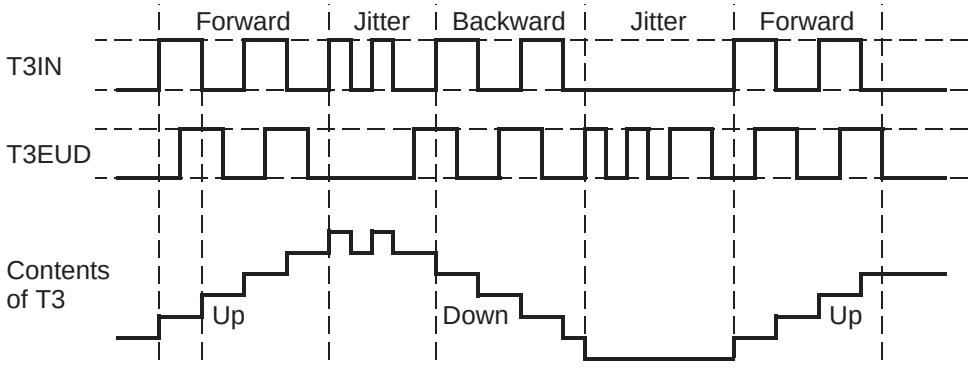
Figure 14-9 and **Figure 14-10** give examples of T3's operation, visualizing count signal generation and direction control. They also show how input jitter is compensated, which might occur if the sensor rests near to one of its switching points.



Note: This example shows the timer behaviour assuming that T3 counts upon any transition on input, i.e. T3I = '011 $_B$'.

MCT04373

Figure 14-9 Evaluation of Incremental Encoder Signals, 2 Count Inputs



Note: This example shows the timer behaviour assuming that T3 counts upon any transition on input T3IN, i.e. T3I = '001 $_B$'.

MCT04374

Figure 14-10 Evaluation of Incremental Encoder Signals, 1 Count Input

*Note: Timer T3 operating in incremental interface mode automatically provides information on the sensor's current position. Dynamic information (speed, acceleration, deceleration) may be obtained by measuring the incoming signal periods. This is facilitated by an additional special capture mode for timer T5 (see **Section 14.2.5**).*

14.1.3 GPT1 Auxiliary Timers T2/T4 Control

Auxiliary timers T2 and T4 have exactly the same functionality. They can be configured for timer mode, gated timer mode, counter mode, or incremental interface mode with the same options for the timer frequencies and the count signal as the core timer T3. In addition to these 4 counting modes, the auxiliary timers can be concatenated with the core timer, or they may be used as reload or capture registers in conjunction with the core timer. The start/stop function of the auxiliary timers can be remotely controlled by the T3 run control bit. Several timers may thus be controlled synchronously.

The current contents of an auxiliary timer are reflected by its count register T2 or T4, respectively. These registers can also be written to by the CPU, for example, to set the initial start value.

The individual configurations for timers T2 and T4 are determined by their bitaddressable control registers T2CON and T4CON, which are organized identically. Note that functions which are present in all 3 timers of block GPT1 are controlled in the same bit positions and in the same manner in each of the specific control registers.

Note: The auxiliary timers have no output toggle latch and no alternate output function.

GPT12E_T2CON
Timer 2 Control Register **SFR ($FF40_H/A0_H$)** **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T2 R DIR	**T2 CH DIR**	**T2 ED GE**	**T2 IR DIS**	**-**	**-**	**T2 RC**	**T2 UDE**	**T2 UD**	**T2R**	**T2M**			**T2I**		
rh	rwh	rwh	rw	-	-	rw	rw	rw	rw	rw			rw		

Field	Bits	Type	Description

GPT12E_T4CON
Timer 4 Control Register **SFR ($FF44_H/A2_H$)** **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T4 R DIR	**T4 CH DIR**	**T4 ED GE**	**T4 IR DIS**	**-**	**-**	**T4 RC**	**T4 UDE**	**T4 UD**	**T4R**	**T4M**			**T4I**		
rh	rwh	rwh	rw	-	-	rw	rw	rw	rw	rw			rw		

Field	Bits	Type	Description
TxRDIR	15	rh	**Timer Tx Rotation Direction** 0 Timer x counts up 1 Timer x counts down
TxCHDIR	14	rwh	**Timer Tx Count Direction Change** This bit is set each time the count direction of timer Tx changes. TxCHDIR must be cleared by SW. 0 No change in count direction was detected 1 A change in count direction was detected
TxEDGE	13	rwh	**Timer Tx Edge Detection** The bit is set each time a count edge is detected. TxEDGE must be cleared by SW. 0 No count edge was detected 1 A count edge was detected
TxIRDIS	12	rw	**Timer Tx Interrupt Request Disable** 0 Interrupt generation for TxCHDIR and TxEDGE interrupts in Incremental Interface Mode is enabled 1 Interrupt generation for TxCHDIR and TxEDGE interrupts in Incremental Interface Mode is disabled
TxRC	9	rw	**Timer Tx Remote Control** 0 Timer Tx is controlled by its own run bit TxR 1 Timer Tx is controlled by the run bit T3R of core timer 3, not by bit TxR
TxUDE	8	rw	**Timer Tx External Up/Down Enable**[1)] 0 Input TxEUD is disconnected 1 Direction influenced by input TxEUD
TxUD	7	rw	**Timer Tx Up/Down Control**[1)] 0 Timer Tx counts up 1 Timer Tx counts down
TxR	6	rw	**Timer Tx Run Bit** 0 Timer Tx stops 1 Timer Tx runs *Note: This bit only controls timer Tx if bit TxRC = 0.*

Field	Bits	Type	Description
TxM	[5:3]	rw	**Timer Tx Mode Control** (Basic Operating Mode) 000 Timer Mode 001 Counter Mode 010 Gated Timer Mode with gate active low 011 Gated Timer Mode with gate active high 100 Reload Mode 101 Capture Mode 110 Incremental Interface Mode (Rotation Detect.) 111 Incremental Interface Mode (Edge Detection)
TxI	[2:0]	rw	**Timer Tx Input Parameter Selection** Depends on the operating mode, see respective sections for encoding: **Table 14-7** for Timer Mode and Gated Timer Mode **Table 14-2** for Counter Mode **Table 14-3** for Incremental Interface Mode

1) See **Table 14-1** for encoding of bits TxUD and TxUDE.

Timer T2/T4 Run Control

Each of the auxiliary timers T2 and T4 can be started or stopped by software in two different ways:

- Through the associated timer run bit (T2R or T4R). In this case it is required that the respective control bit TxRC = 0.
- Through the core timer's run bit (T3R). In this case the respective remote control bit must be set (TxRC = 1).

The selected run bit is relevant in all operating modes of T2/T4. Setting the bit will start the timer, clearing the bit stops the timer.

In gated timer mode, the timer will only run if the selected run bit is set and the gate is active (high or low, as programmed).

Note: If remote control is selected T3R will start/stop timer T3 and the selected auxiliary timer(s) synchronously.

Count Direction Control

The count direction of the GPT1 timers (core timer and auxiliary timers) is controlled in the same way, either by software or by the external input pin TxEUD. Please refer to the description in **Table 14-1**.

Note: When pin TxEUD is used as external count direction control input, it must be configured as input (its corresponding direction control bit must be cleared).

14.1.4 GPT1 Auxiliary Timers T2/T4 Operating Modes

The operation of the auxiliary timers in the basic operating modes is almost identical with the core timer's operation, with very few exceptions. Additionally, some combined operating modes can be selected.

Timers T2 and T4 in Timer Mode

Timer mode for an auxiliary timer Tx is selected by setting its bitfield TxM in register TxCON to 000_B.

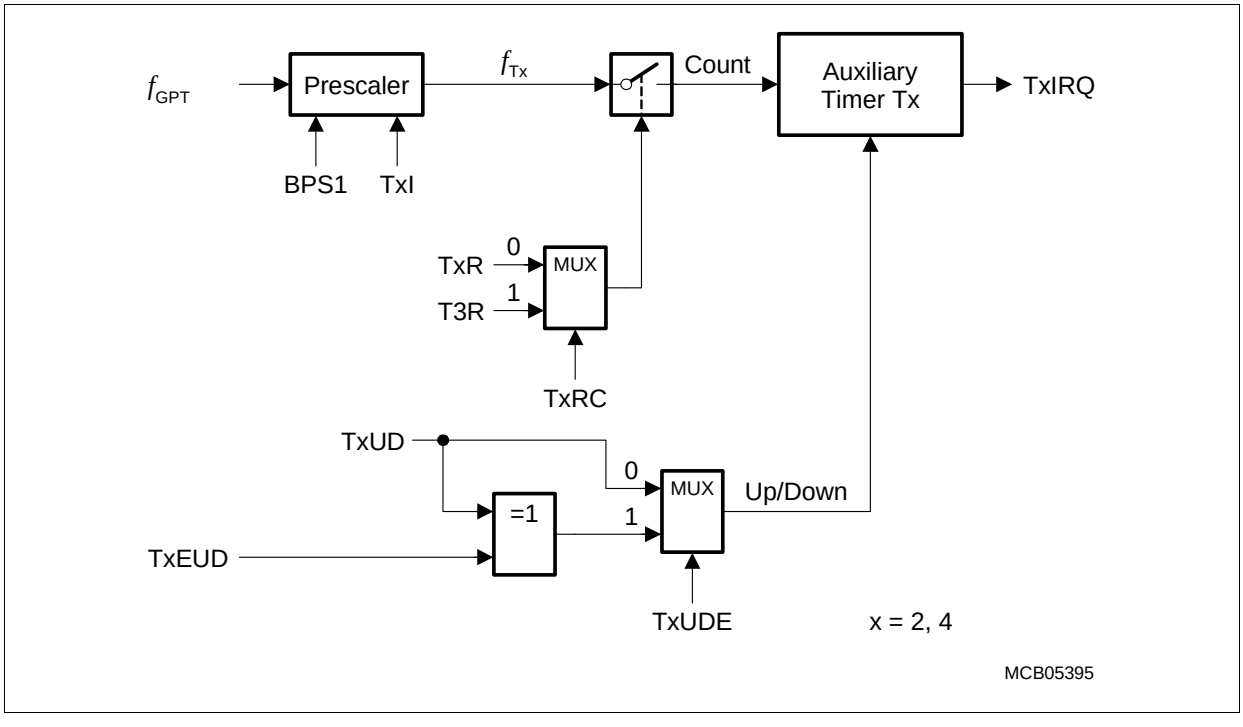


Figure 14-11 Block Diagram of an Auxiliary Timer in Timer Mode

Timers T2 and T4 in Gated Timer Mode

Gated timer mode for an auxiliary timer Tx is selected by setting bitfield TxM in register TxCON to 010_B or 011_B. Bit TxM.0 (TxCON.3) selects the active level of the gate input.

Note: A transition of the gate signal at line TxIN does not cause an interrupt request.

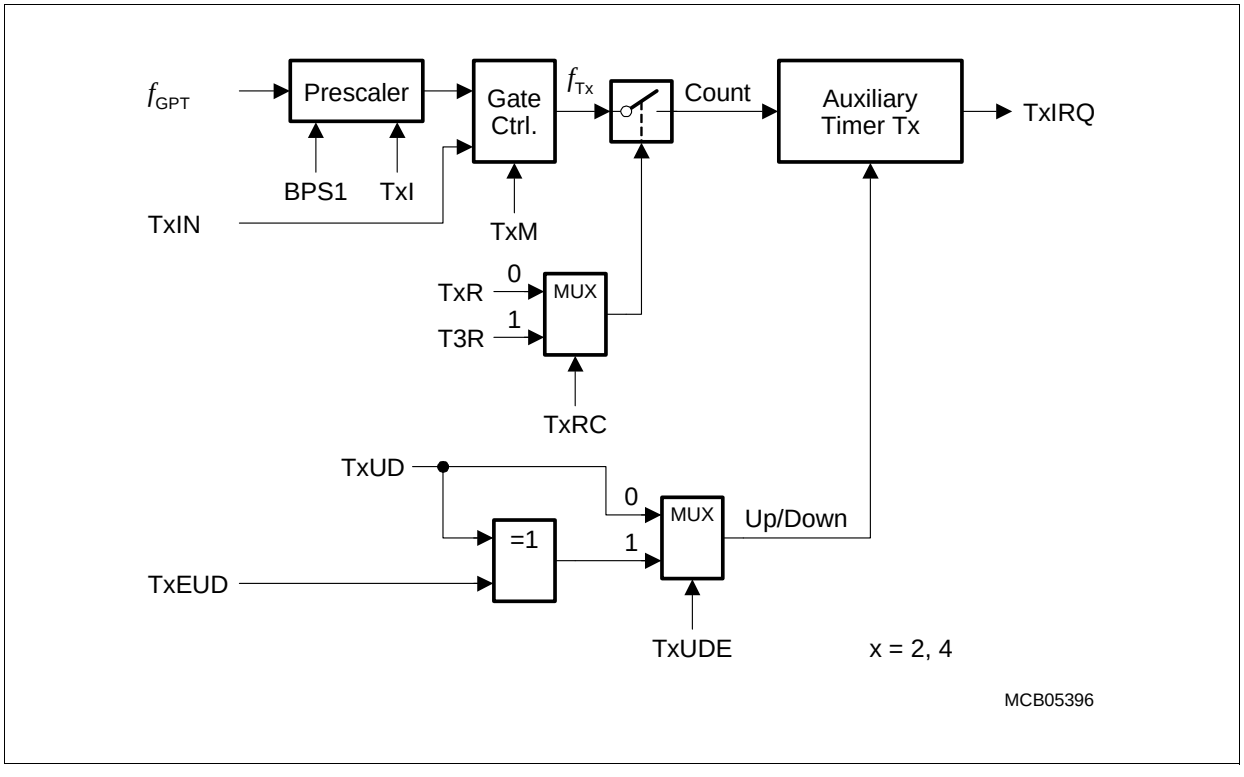


Figure 14-12 Block Diagram of an Auxiliary Timer in Gated Timer Mode

Note: There is no output toggle latch for T2 and T4.
Start/stop of an auxiliary timer can be controlled locally or remotely.

Timers T2 and T4 in Counter Mode

Counter mode for an auxiliary timer Tx is selected by setting bitfield TxM in register TxCON to 001_B. In counter mode, an auxiliary timer can be clocked either by a transition at its external input line TxIN, or by a transition of timer T3's toggle latch T3OTL. The event causing an increment or decrement of a timer can be a positive, a negative, or both a positive and a negative transition at either the respective input pin or at the toggle latch. Bitfield TxI in control register TxCON selects the triggering transition (see **Table 14-5**).

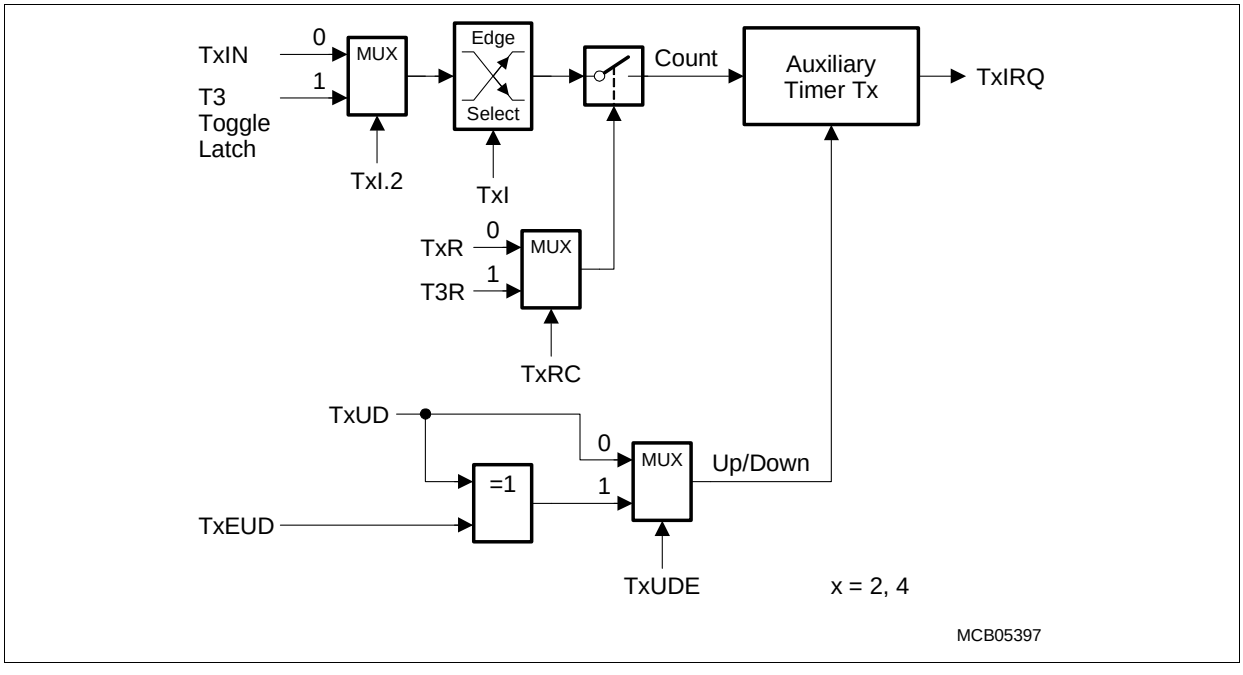


Figure 14-13 Block Diagram of an Auxiliary Timer in Counter Mode

Table 14-5 GPT1 Auxiliary Timer (Counter Mode) Input Edge Selection

T2I/T4I	Triggering Edge for Counter Increment/Decrement
X 0 0	None. Counter Tx is disabled
0 0 1	Positive transition (rising edge) on TxIN
0 1 0	Negative transition (falling edge) on TxIN
0 1 1	Any transition (rising or falling edge) on TxIN
1 0 1	Positive transition (rising edge) of T3 toggle latch T3OTL
1 1 0	Negative transition (falling edge) of T3 toggle latch T3OTL
1 1 1	Any transition (rising or falling edge) of T3 toggle latch T3OTL

Note: Only state transitions of T3OTL which are caused by the overflows/underflows of T3 will trigger the counter function of T2/T4. Modifications of T3OTL via software will NOT trigger the counter function of T2/T4.

For counter operation, pin TxIN must be configured as input (the respective direction control bit DPx.y must be 0). The maximum input frequency allowed in counter mode depends on the selected prescaler value. To ensure that a transition of the count input signal applied to TxIN is recognized correctly, its level must be held high or low for a minimum number of module clock cycles before it changes. This information can be found in **Section 14.1.5**.

Timers T2 and T4 in Incremental Interface Mode

Incremental interface mode for an auxiliary timer Tx is selected by setting bitfield TxM in the respective register TxCON to 110_B or 111_B. In incremental interface mode, the two inputs associated with an auxiliary timer Tx (TxIN, TxEUD) are used to interface to an incremental encoder. Tx is clocked by each transition on one or both of the external input pins to provide 2-fold or 4-fold resolution of the encoder input.

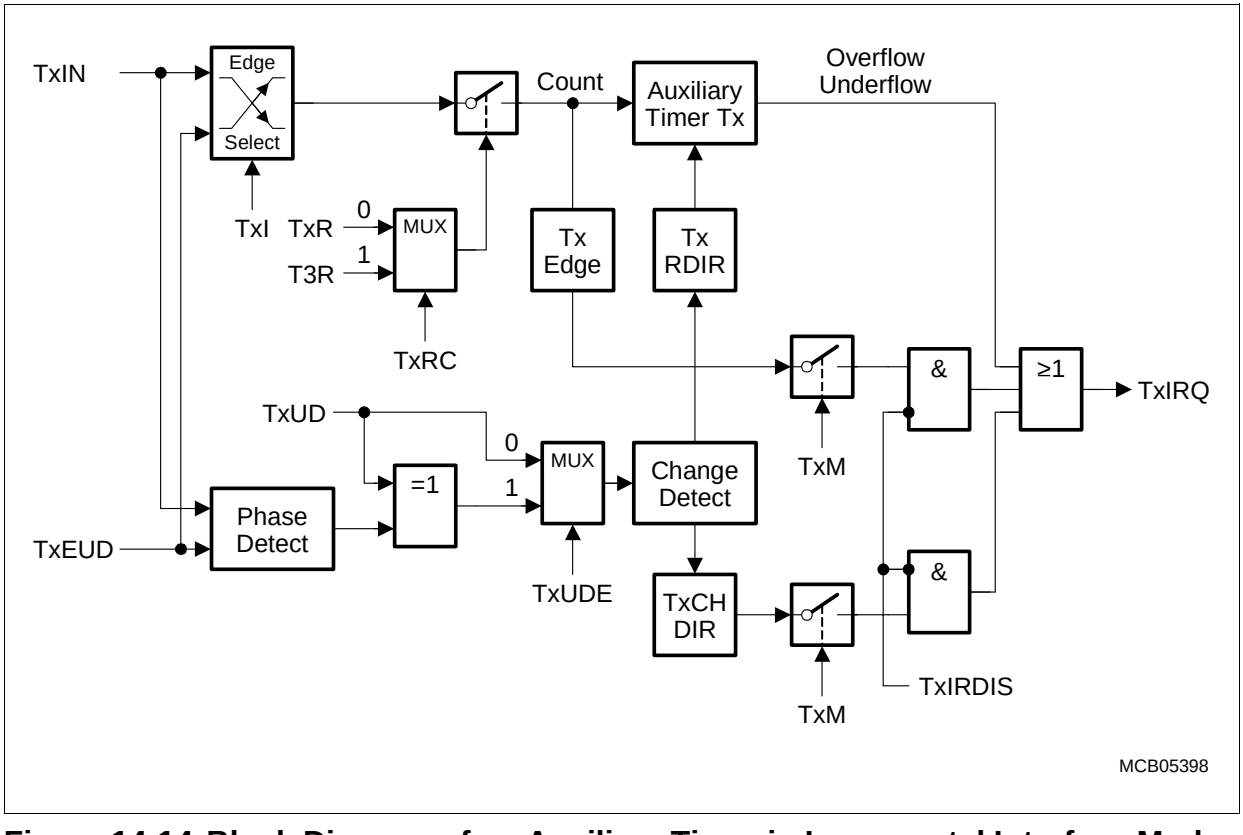


Figure 14-14 Block Diagram of an Auxiliary Timer in Incremental Interface Mode

The operation of the auxiliary timers T2 and T4 in incremental interface mode and the interrupt generation are the same as described for the core timer T3. The descriptions, figures and tables apply accordingly.

Timer Concatenation

Using the toggle bit T3OTL as a clock source for an auxiliary timer in counter mode concatenates the core timer T3 with the respective auxiliary timer. This concatenation forms either a 32-bit or a 33-bit timer/counter, depending on which transition of T3OTL is selected to clock the auxiliary timer.

- **32-bit Timer/Counter:** If both a positive and a negative transition of T3OTL are used to clock the auxiliary timer, this timer is clocked on every overflow/underflow of the core timer T3. Thus, the two timers form a 32-bit timer.
- **33-bit Timer/Counter:** If either a positive or a negative transition of T3OTL is selected to clock the auxiliary timer, this timer is clocked on every second overflow/underflow of the core timer T3. This configuration forms a 33-bit timer (16-bit core timer + T3OTL + 16-bit auxiliary timer).
  As long as bit T3OTL is not modified by software, it represents the state of the internal toggle latch, and can be regarded as part of the 33-bit timer.

The count directions of the two concatenated timers are not required to be the same. This offers a wide variety of different configurations.

T3, which represents the low-order part of the concatenated timer, can operate in timer mode, gated timer mode or counter mode in this case.

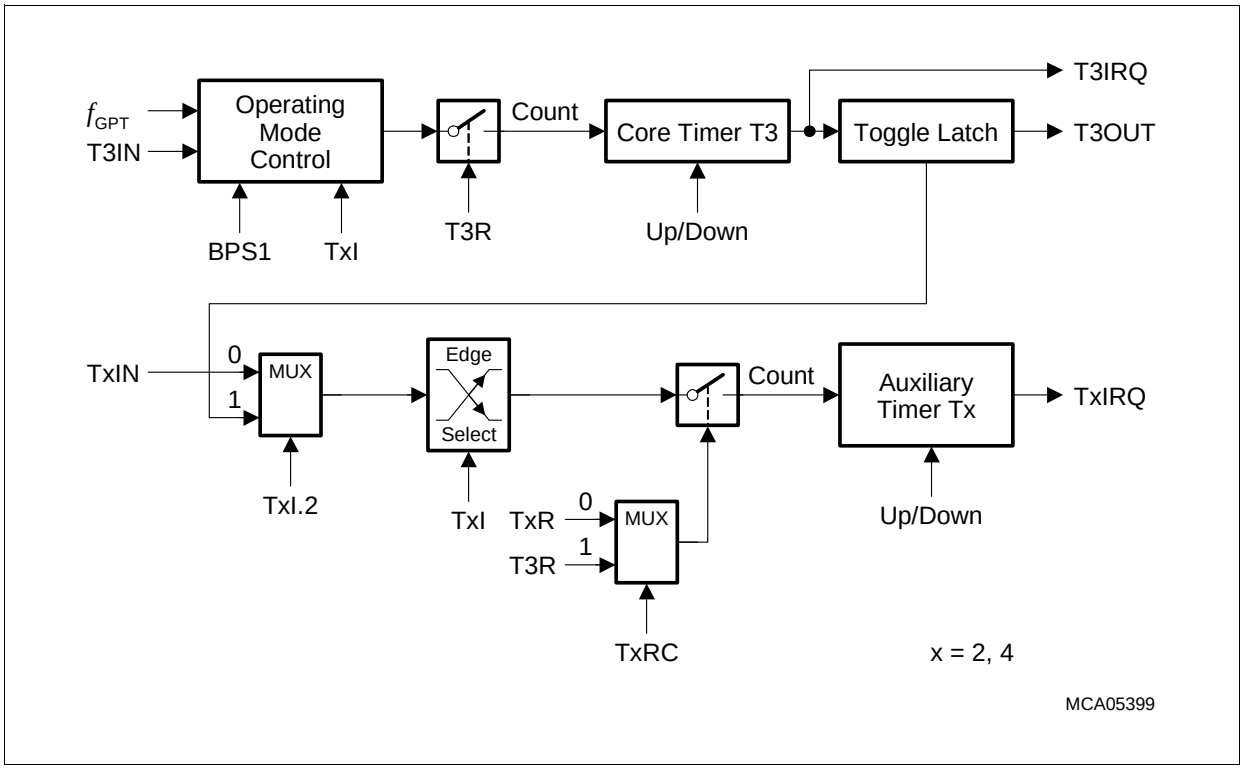


Figure 14-15 Concatenation of Core Timer T3 and an Auxiliary Timer

Auxiliary Timer in Reload Mode

Reload Mode for an auxiliary timer Tx is selected by setting bitfield TxM in the respective register TxCON to 100_B. In reload mode, the core timer T3 is reloaded with the contents of an auxiliary timer register, triggered by one of two different signals. The trigger signal is selected the same way as the clock source for counter mode (see **Table 14-5**), i.e. a transition of the auxiliary timer's input TxIN or the toggle latch T3OTL may trigger the reload.

Note: When programmed for reload mode, the respective auxiliary timer (T2 or T4) stops independently of its run flag T2R or T4R.
The timer input pin TxIN must be configured as input if it shall trigger a reload operation.

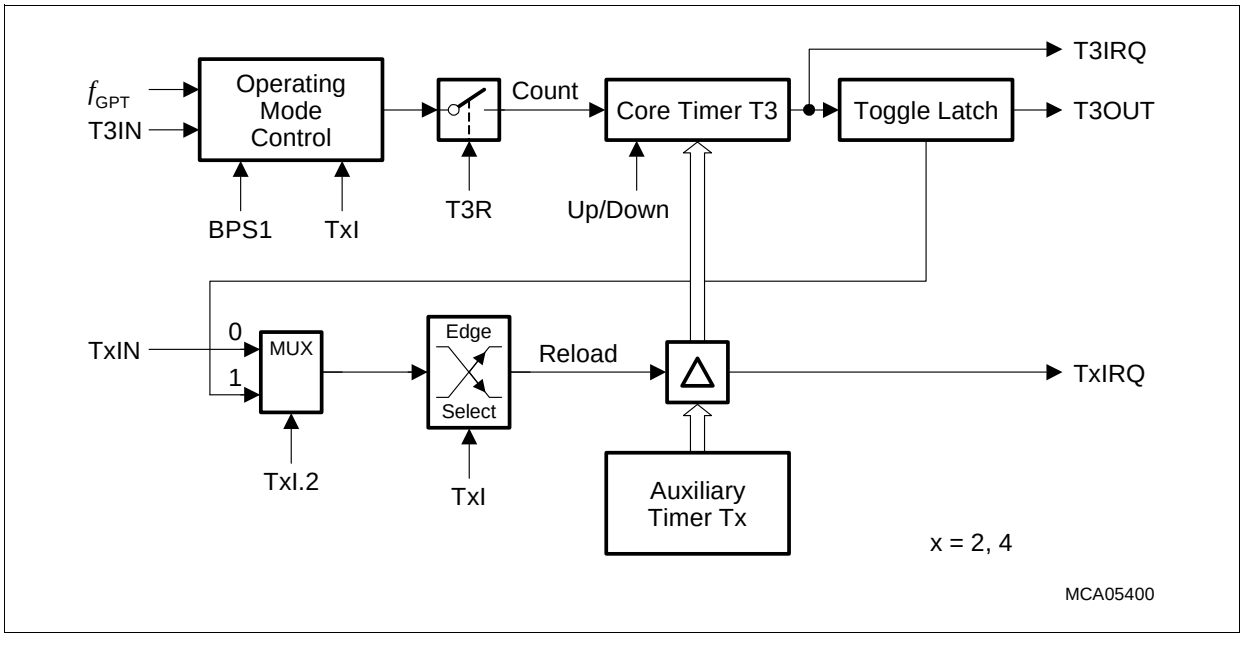


Figure 14-16 GPT1 Auxiliary Timer in Reload Mode

Upon a trigger signal, T3 is loaded with the contents of the respective timer register (T2 or T4) and the respective interrupt request flag (T2IR or T4IR) is set.

Note: When a T3OTL transition is selected for the trigger signal, the interrupt request flag T3IR will also be set upon a trigger, indicating T3's overflow or underflow. Modifications of T3OTL via software will NOT trigger the counter function of T2/T4.

To ensure that a transition of the reload input signal applied to TxIN is recognized correctly, its level must be held high or low for a minimum number of module clock cycles, detailed in **Section 14.1.5**.

The reload mode triggered by the T3 toggle latch can be used in a number of different configurations. The following functions can be performed, depending on the selected active transition:

- If both a positive and a negative transition of T3OTL are selected to trigger a reload, the core timer will be reloaded with the contents of the auxiliary timer each time it overflows or underflows. This is the standard reload mode (reload on overflow/underflow).
- If either a positive or a negative transition of T3OTL is selected to trigger a reload, the core timer will be reloaded with the contents of the auxiliary timer on every second overflow or underflow.
- Using this “single-transition” mode for both auxiliary timers allows to perform very flexible Pulse Width Modulation (PWM). One of the auxiliary timers is programmed to reload the core timer on a positive transition of T3OTL, the other is programmed for a reload on a negative transition of T3OTL. With this combination the core timer is alternately reloaded from the two auxiliary timers.

Figure 14-17 shows an example for the generation of a PWM signal using the “single-transition” reload mechanism. T2 defines the high time of the PWM signal (reloaded on positive transitions) and T4 defines the low time of the PWM signal (reloaded on negative transitions). The PWM signal can be output on pin T3OUT if T3OE = 1. With this method, the high and low time of the PWM signal can be varied in a wide range.

Note: The output toggle latch T3OTL is accessible via software and may be changed, if required, to modify the PWM signal.
However, this will NOT trigger the reloading of T3.

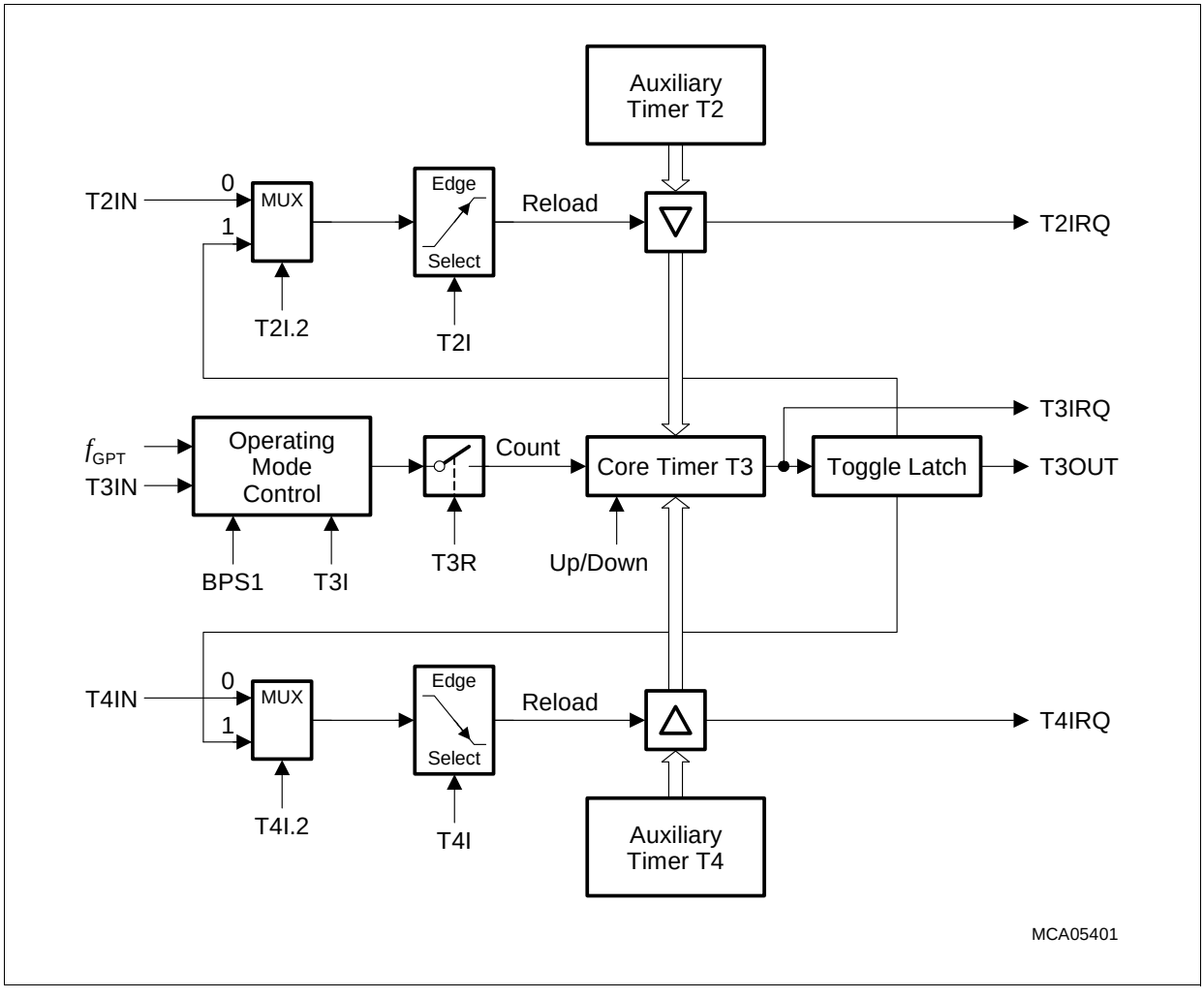


Figure 14-17 GPT1 Timer Reload Configuration for PWM Generation

Note: Although possible, selecting the same reload trigger event for both auxiliary timers should be avoided. In such a case, both reload registers would try to load the core timer at the same time. If this combination is selected, T2 is disregarded and the contents of T4 is reloaded.

Auxiliary Timer in Capture Mode

Capture mode for an auxiliary timer Tx is selected by setting bitfield TxM in the respective register TxCON to 101_B. In capture mode, the contents of the core timer T3 are latched into an auxiliary timer register in response to a signal transition at the respective auxiliary timer's external input pin TxIN. The capture trigger signal can be a positive, a negative, or both a positive and a negative transition.

The two least significant bits of bitfield TxI select the active transition (see **Table 14-5**). Bit 2 of TxI is irrelevant for capture mode and must be cleared (TxI.2 = 0).

Note: When programmed for capture mode, the respective auxiliary timer (T2 or T4) stops independently of its run flag T2R or T4R.

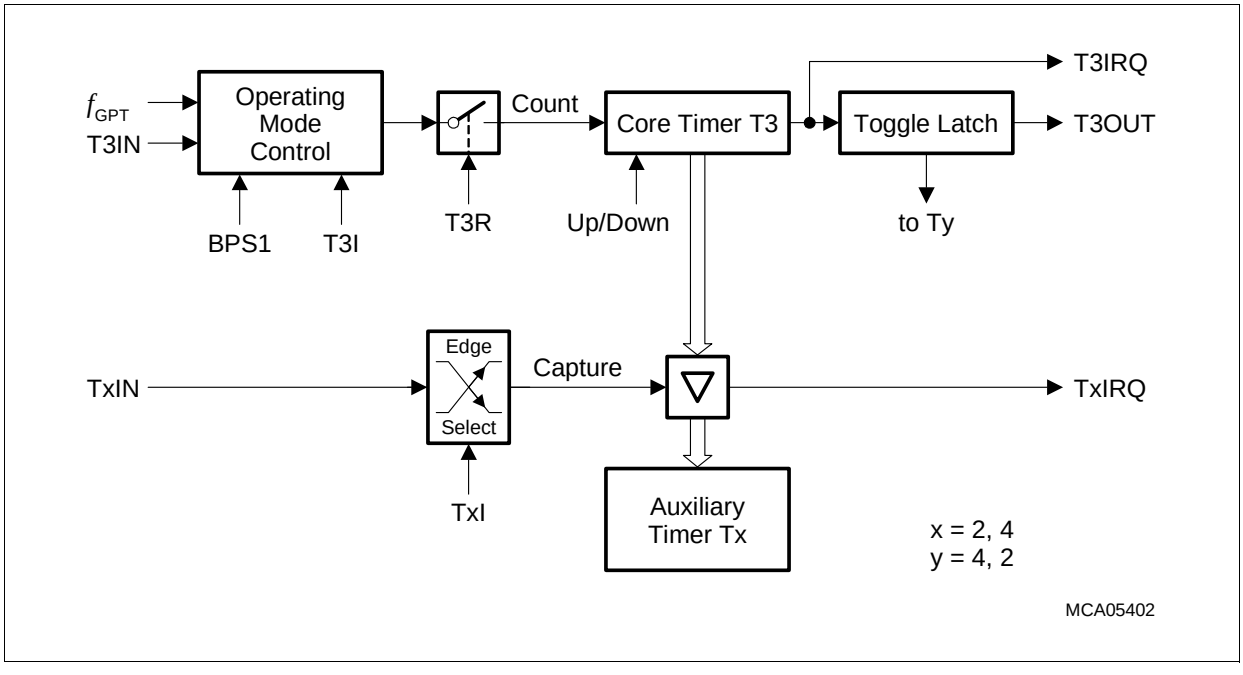


Figure 14-18 GPT1 Auxiliary Timer in Capture Mode

Upon a trigger (selected transition) at the corresponding input pin TxIN the contents of the core timer are loaded into the auxiliary timer register and the associated interrupt request flag TxIR will be set.

For capture mode operation, the respective timer input pin TxIN must be configured as input. To ensure that a transition of the capture input signal applied to TxIN is recognized correctly, its level must be held high or low for a minimum number of module clock cycles, detailed in **Section 14.1.5**.

14.1.5 GPT1 Clock Signal Control

All actions within the timer block GPT1 are triggered by transitions of its basic clock. This basic clock is derived from the system clock by a basic block prescaler, controlled by bitfield BPS1 in register T3CON (see **Figure 14-2**). The count clock can be generated in two different ways:

- **Internal count clock**, derived from GPT1's basic clock via a programmable prescaler, is used for (gated) timer mode.
- **External count clock**, derived from the timer's input pin(s), is used for counter mode.

For both ways, the basic clock determines the maximum count frequency and the timer's resolution:

Table 14-6 Basic Clock Selection for Block GPT1

Block Prescaler[1)]	BPS1 = 01_B	BPS1 = 00_B[2)]	BPS1 = 11_B	BPS1 = 10_B
Prescaling Factor for GPT1: F(BPS1)	F(BPS1) = 4	F(BPS1) = 8	F(BPS1) = 16	F(BPS1) = 32
Maximum External Count Frequency	$f_{GPT}/8$	$f_{GPT}/16$	$f_{GPT}/32$	$f_{GPT}/64$
Input Signal Stable Time	$4 \times t_{GPT}$	$8 \times t_{GPT}$	$16 \times t_{GPT}$	$32 \times t_{GPT}$

1) Please note the non-linear encoding of bitfield BPS1.

2) Default after reset.

Internal Count Clock Generation

In timer mode and gated timer mode, the count clock for each GPT1 timer is derived from the GPT1 basic clock by a programmable prescaler, controlled by bitfield TxI in the respective timer's control register TxCON.

The count frequency f_{Tx} for a timer Tx and its resolution r_{Tx} are scaled linearly with lower clock frequencies, as can be seen from the following formula:

$$f_{Tx} = \frac{f_{GPT}}{F(BPS1) \times 2^{<TxI>}} \qquad r_{Tx}[\mu s] = \frac{F(BPS1) \times 2^{<TxI>}}{f_{GPT}[MHz]} \qquad [14.1]$$

The effective count frequency depends on the common module clock prescaler factor F(BPS1) as well as on the individual input prescaler factor $2^{<TxI>}$. **Table 14-7** summarizes the resulting overall divider factors for a GPT1 timer that result from these cascaded prescalers.

Table 14-8 lists a timer's parameters (such as count frequency, resolution, and period) resulting from the selected overall prescaler factor and the applied system frequency. Note that some numbers may be rounded.

Table 14-7 GPT1 Overall Prescaler Factors for Internal Count Clock

Individual Prescaler for Tx	Common Prescaler for Module Clock[1)]			
	BPS1 = 01_B	BPS1 = 00_B	BPS1 = 11_B	BPS1 = 10_B
TxI = 000_B	4	8	16	32
TxI = 001_B	8	16	32	64
TxI = 010_B	16	32	64	128
TxI = 011_B	32	64	128	256
TxI = 100_B	64	128	256	512
TxI = 101_B	128	256	512	1024
TxI = 110_B	256	512	1024	2048
TxI = 111_B	512	1024	2048	4096

1) Please note the non-linear encoding of bitfield BPS1.

Table 14-8 GPT1 Timer Parameters

System Clock = 10 MHz			Overall Divider Factor	System Clock = 40 MHz		
Frequency	Resolution	Period		Frequency	Resolution	Period
2.5 MHz	400 ns	26.21 ms	4	10.0 MHz	100 ns	6.55 ms
1.25 MHz	800 ns	52.43 ms	8	5.0 MHz	200 ns	13.11 ms
625.0 kHz	1.6 µs	104.9 ms	16	2.5 MHz	400 ns	26.21 ms
312.5 kHz	3.2 µs	209.7 ms	32	1.25 MHz	800 ns	52.43 ms
156.25 kHz	6.4 µs	419.4 ms	64	625.0 kHz	1.6 µs	104.9 ms
78.125 kHz	12.8 µs	838.9 ms	128	312.5 kHz	3.2 µs	209.7 ms
39.06 kHz	25.6 µs	1.678 s	256	156.25 kHz	6.4 µs	419.4 ms
19.53 kHz	51.2 µs	3.355 s	512	78.125 kHz	12.8 µs	838.9 ms
9.77 kHz	102.4 µs	6.711 s	1024	39.06 kHz	25.6 µs	1.678 s
4.88 kHz	204.8 µs	13.42 s	2048	19.53 kHz	51.2 µs	3.355 s
2.44 kHz	409.6 µs	26.84 s	4096	9.77 kHz	102.4 µs	6.711 s

External Count Clock Input

The external input signals of the GPT1 block are sampled with the GPT1 basic clock (see **Figure 14-2**). To ensure that a signal is recognized correctly, its current level (high or low) must be held active for at least one complete sampling period, before changing. A signal transition is recognized if two subsequent samples of the input signal represent different levels. Therefore, a minimum of two basic clock periods are required for the sampling of an external input signal. Thus, the maximum frequency of an input signal must not be higher than half the basic clock.

Table 14-9 summarizes the resulting requirements for external GPT1 input signals.

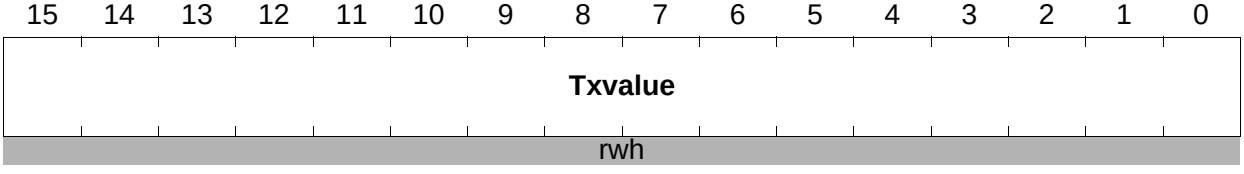
Table 14-9 GPT1 External Input Signal Limits

System Clock = 10 MHz		Input Frequ. Factor	GPT1 Divider BPS1	Input Phase Duration	System Clock = 40 MHz	
Max. Input Frequency	Min. Level Hold Time				Max. Input Frequency	Min. Level Hold Time
1.25 MHz	400 ns	$f_{GPT}/8$	01_B	$4 \times t_{GPT}$	5.0 MHz	100 ns
625.0 kHz	800 ns	$f_{GPT}/16$	00_B	$8 \times t_{GPT}$	2.5 MHz	200 ns
312.5 kHz	1.6 µs	$f_{GPT}/32$	11_B	$16 \times t_{GPT}$	1.25 MHz	400 ns
156.25 kHz	3.2 µs	$f_{GPT}/64$	10_B	$32 \times t_{GPT}$	625.0 kHz	800 ns

These limitations are valid for all external input signals to GPT1, including the external count signals in counter mode and incremental interface mode, the gate input signals in gated timer mode, and the external direction signals.

14.1.6 GPT1 Timer Registers

GPT12E_T2
Timer 2 Count Register **SFR ($FE40_H/20_H$)** **Reset Value: 0000_H**
GPT12E_T3
Timer 3 Count Register **SFR ($FE42_H/21_H$)** **Reset Value: 0000_H**
GPT12E_T4
Timer 4 Count Register **SFR ($FE44_H/22_H$)** **Reset Value: 0000_H**



Field	Bits	Type	Description
Txvalue	[15:0]	rwh	**Current timer value**

14.1.7 Interrupt Control for GPT1 Timers

When a timer overflows from $FFFF_H$ to 0000_H (when counting up), or when it underflows from 0000_H to $FFFF_H$ (when counting down), its interrupt request flag (T2IR, T3IR or T4IR) in register TxIC will be set. This will cause an interrupt to the respective timer interrupt vector (T2INT, T3INT or T4INT) or trigger a PEC service, if the respective interrupt enable bit (T2IE, T3IE or T4IE in register TxIC) is set. There is an interrupt control register for each of the three timers.

GPT12E_T2IC
Timer 2 Intr. Ctrl. Reg. **SFR ($FF60_H$/$B0_H$)** **Reset Value: - - 00_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	GPX	T2IR	T2IE	ILVL				GLVL	
-	-	-	-	-	-	-	rw	rwh	rw	rw				rw	

GPT12E_T3IC
Timer 3 Intr. Ctrl. Reg. **SFR ($FF62_H$/$B1_H$)** **Reset Value: - - 00_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	GPX	T3IR	T3IE	ILVL				GLVL	
-	-	-	-	-	-	-	rw	rwh	rw	rw				rw	

GPT12E_T4IC
Timer 4 Intr. Ctrl. Reg. **SFR ($FF64_H$/$B2_H$)** **Reset Value: - - 00_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	GPX	T4IR	T4IE	ILVL				GLVL	
-	-	-	-	-	-	-	rw	rwh	rw	rw				rw	

Note: Please refer to the general Interrupt Control Register description for an explanation of the control fields.

14.2 Timer Block GPT2

From a programmer's point of view, the GPT2 block is represented by a set of SFRs as summarized below. Those portions of port and direction registers which are used for alternate functions by the GPT2 block are shaded.

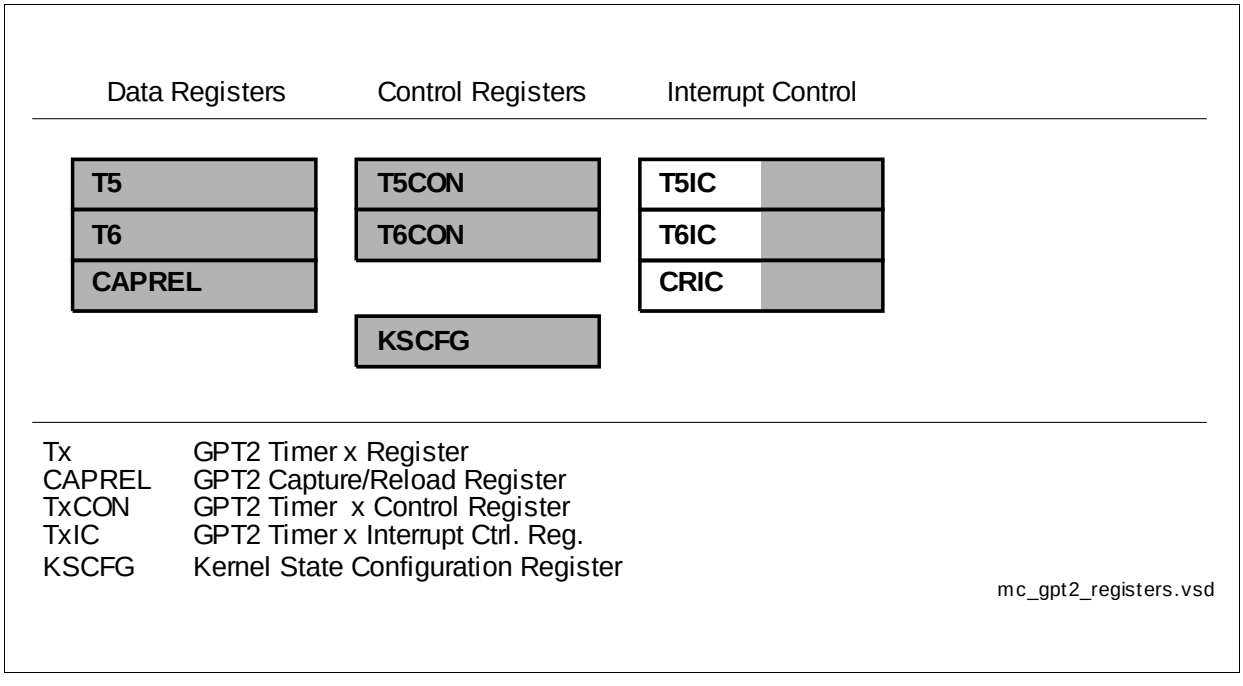


Figure 14-19 SFRs Associated with Timer Block GPT2

Both timers of block GPT2 (T5, T6) can run in one of 3 basic modes: Timer Mode, Gated Timer Mode, or Counter Mode. All timers can count up or down. Each timer of GPT2 is controlled by a separate control register TxCON.

Each timer has an input pin TxIN (alternate pin function) associated with it, which serves as the gate control in gated timer mode, or as the count input in counter mode. The count direction (up/down) may be programmed via software or may be dynamically altered by a signal at the External Up/Down control input TxEUD (alternate pin function). An overflow/underflow of core timer T6 is indicated by the Output Toggle Latch T6OTL, whose state may be output on the associated pin T6OUT (alternate pin function). The auxiliary timer T5 may additionally be concatenated with core timer T6 (through T6OTL).

The Capture/Reload register CAPREL can be used to capture the contents of timer T5, or to reload timer T6. A special mode facilitates the use of register CAPREL for both functions at the same time. This mode allows frequency multiplication. The capture function is triggered by the input pin CAPIN, or by GPT1 timer's T3 input lines T3IN and T3EUD. The reload function is triggered by an overflow or underflow of timer T6. Overflows/underflows of timer T6 may also clock the timers of the CAPCOM units.

The current contents of each timer can be read or modified by the CPU by accessing the corresponding timer count registers T5 or T6, located in the non-bitaddressable SFR

space (see **Section 14.2.7**). When any of the timer registers is written to by the CPU in the state immediately preceding a timer increment, decrement, reload, or capture operation, the CPU write operation has priority in order to guarantee correct results.

The interrupts of GPT2 are controlled through the Interrupt Control Registers TxIC. These registers are not part of the GPT2 block. The input and output lines of GPT2 are connected to pins of Ports P3 and P5. The control registers for the port functions are located in the respective port modules.

*Note: The timing requirements for external input signals can be found in **Section 14.2.6**, **Section 14.3** summarizes the module interface signals, including pins.*

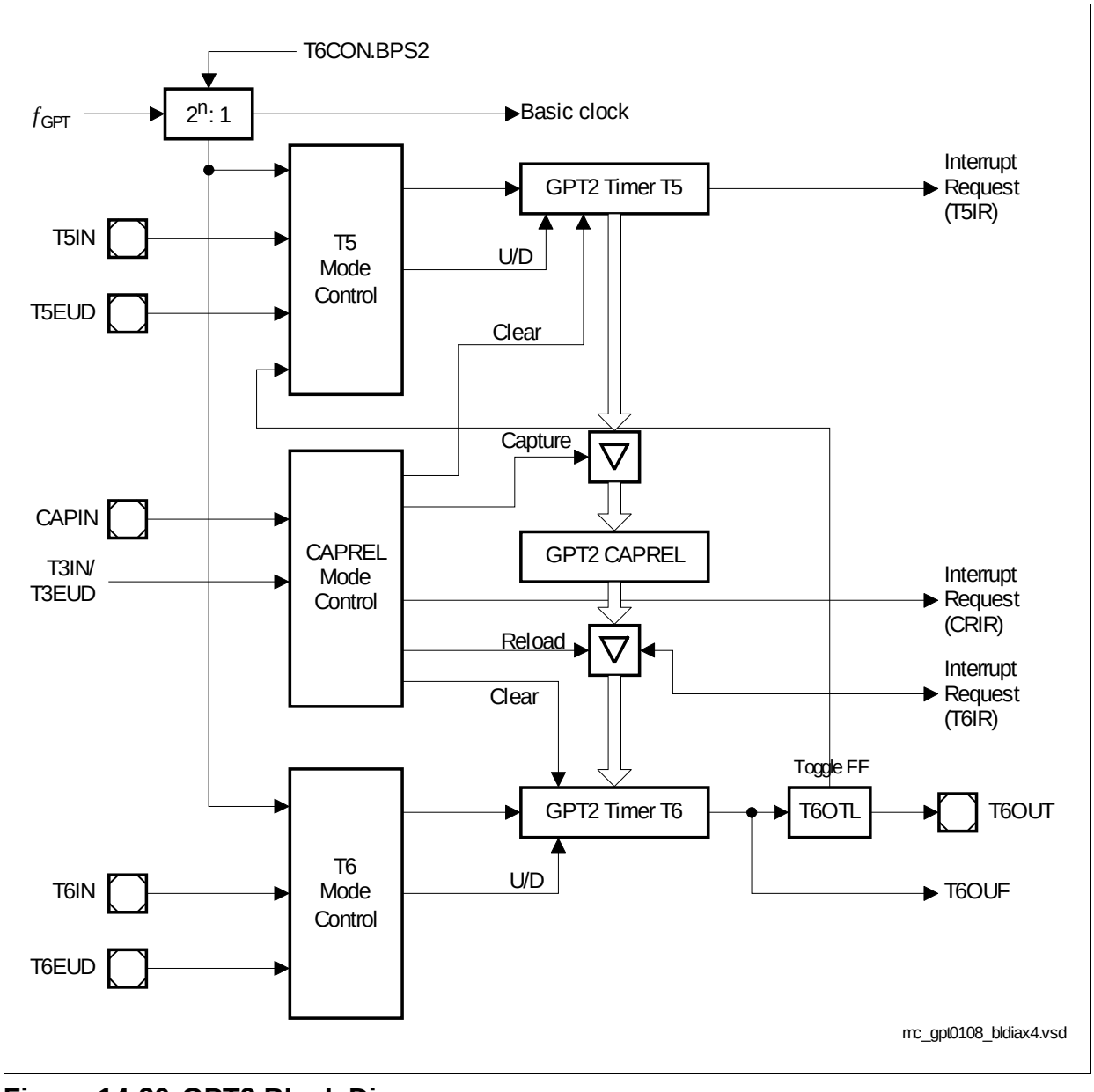


Figure 14-20 GPT2 Block Diagram

14.2.1 GPT2 Core Timer T6 Control

The current contents of the core timer T6 are reflected by its count register T6. This register can also be written to by the CPU, for example, to set the initial start value.

The core timer T6 is configured and controlled via its bitaddressable control register T6CON.

GPT12E_T6CON
Timer 6 Control Register **SFR ($FF48_H/A4_H$)** **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T6 SR	**T6 CLR**	**-**	**BPS2**		**T6 OTL**	**T6 OE**	**T6 UDE**	**T6 UD**	**T6R**	**T6M**			**T6I**		
rw	rw	-	rw		rwh	rw	rw	rw	rw	rw			rw		

Field	Bits	Type	Description
T6SR	15	rw	**Timer 6 Reload Mode Enable** 0 Reload from register CAPREL Disabled 1 Reload from register CAPREL Enabled
T6CLR	14	rw	**Timer T6 Clear Enable Bit** 0 Timer T6 is not cleared on a capture event 1 Timer T6 is cleared on a capture event
BPS2	[12:11]	rw	**GPT2 Block Prescaler Control** Selects the basic clock for block GPT2 (see also **Section 14.2.6**) 00 $f_{GPT}/4$ 01 $f_{GPT}/2$ 10 $f_{GPT}/16$ 11 $f_{GPT}/8$
T6OTL	10	rwh	**Timer T6 Overflow Toggle Latch** Toggles on each overflow/underflow of T6. Can be set or reset by software (see separate description)
T6OE	9	rw	**Overflow/Underflow Output Enable** 0 Alternate Output Function Disabled 1 State of T6 toggle latch is output on pin T6OUT
T6UDE	8	rw	**Timer T6 External Up/Down Enable[1)]** 0 Input T6EUD is disconnected 1 Direction influenced by input T6EUD

Field	Bits	Type	Description
T6UD	7	rw	**Timer T6 Up/Down Control**[1)] 0 Timer T6 counts up 1 Timer T6 counts down
T6R	6	rw	**Timer T6 Run Bit** 0 Timer T6 stops 1 Timer T6 runs
T6M	[5:3]	rw	**Timer T6 Mode Control** (Basic Operating Mode) 000 Timer Mode 001 Counter Mode 010 Gated Timer Mode with gate active low 011 Gated Timer Mode with gate active high 100 Reserved. Do not use this combination. 101 Reserved. Do not use this combination. 110 Reserved. Do not use this combination. 111 Reserved. Do not use this combination.
T6I	[2:0]	rw	**Timer T6 Input Parameter Selection** Depends on the operating mode, see respective sections for encoding: **Table 14-15** for Timer Mode and Gated Timer Mode **Table 14-11** for Counter Mode

1) See **Table 14-10** for encoding of bits T6UD and T6UDE.

Timer T6 Run Control

The core timer T6 can be started or stopped by software through bit T6R (timer T6 run bit). This bit is relevant in all operating modes of T6. Setting bit T6R will start the timer, clearing bit T6R stops the timer.

In gated timer mode, the timer will only run if T6R = 1 and the gate is active (high or low, as programmed).

Note: When bit T5RC in timer control register T5CON is set, bit T6R will also control (start and stop) the Auxiliary Timer T5.

Count Direction Control

The count direction of the GPT2 timers (core timer and auxiliary timer) can be controlled either by software or by the external input pin TxEUD (Timer Tx External Up/Down Control Input). These options are selected by bits TxUD and TxUDE in the respective control register TxCON. When the up/down control is provided by software (bit TxUDE = 0), the count direction can be altered by setting or clearing bit TxUD. When bit TxUDE = 1, pin TxEUD is selected to be the controlling source of the count direction. However, bit TxUD can still be used to reverse the actual count direction, as shown in **Table 14-10**. The count direction can be changed regardless of whether or not the timer is running.

Table 14-10 GPT2 Timer Count Direction Control

Pin TxEUD	Bit TxUDE	Bit TxUD	Count Direction
X	0	0	Count Up
X	0	1	Count Down
0	1	0	Count Up
1	1	0	Count Down
0	1	1	Count Down
1	1	1	Count Up

Timer 6 Output Toggle Latch

The overflow/underflow signal of timer T6 is connected to a block named ‘Toggle Latch’, shown in the timer mode diagrams. **Figure 14-21** illustrates the details of this block. An overflow or underflow of T6 will clock two latches: The first latch represents bit T6OTL in control register T6CON. The second latch is an internal latch toggled by T6OTL’s output. Both latch outputs are connected to the input control block of the auxiliary timer T5. The output level of the shadow latch will match the output level of T6OTL, but is delayed by one clock cycle. When the T6OTL value changes, this will result in a temporarily different output level from T6OTL and the shadow latch, which can trigger the selected count event in T5.

When software writes to T6OTL, both latches are set or cleared simultaneously. In this case, both signals to the auxiliary timers carry the same level and no edge will be detected. Bit T6OE (overflow/underflow output enable) in register T6CON enables the state of T6OTL to be monitored via an external pin T6OUT. When T6OTL is linked to an external port pin (must be configured as output), T6OUT can be used to control external HW. If T6OE = 1, pin T6OUT outputs the state of T6OTL. If T6OE = 0, pin T6OUT outputs a high level (while it selects the timer output signal).

As can be seen from **Figure 14-21**, when latch T6OTL is modified by software to determine the state of the output line, also the internal shadow latch is set or cleared accordingly. Therefore, no trigger condition is detected by T5 in this case.

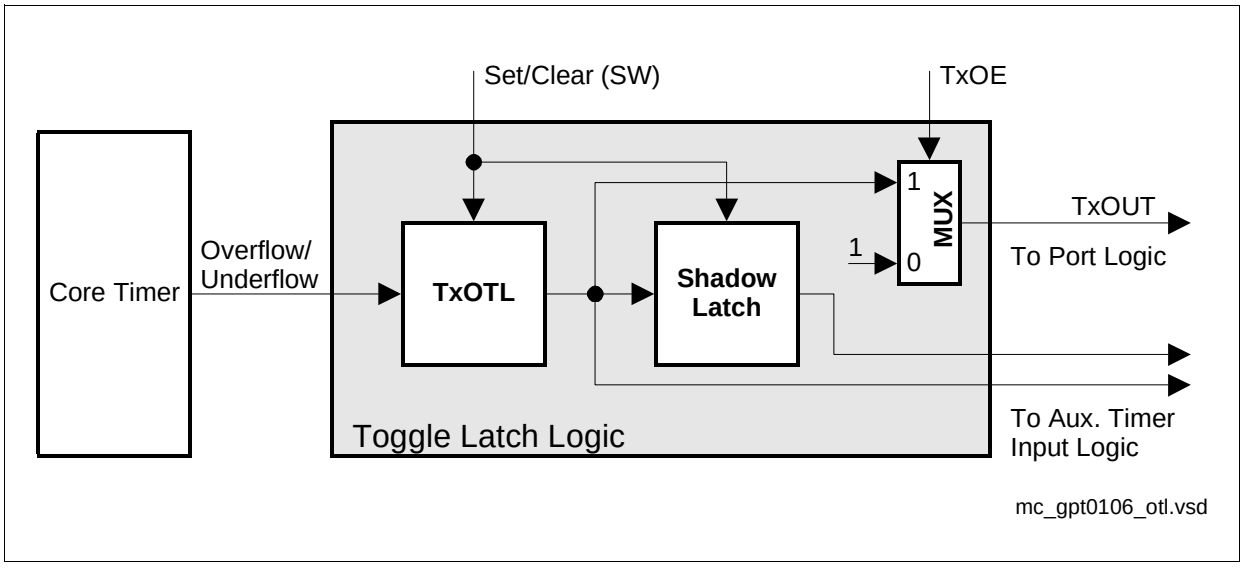


Figure 14-21 Block Diagram of the Toggle Latch Logic of Core Timer T6

Note: T6 is also used to clock the timers in the CAPCOM units. For this purpose, there is a direct internal connection between the T6 overflow/underflow line and the CAPCOM timers (signal T6OUF).

14.2.2 GPT2 Core Timer T6 Operating Modes

Timer 6 in Timer Mode

Timer mode for the core timer T6 is selected by setting bitfield T6M in register T6CON to 000_B. In this mode, T6 is clocked with the module's input clock f_{GPT} divided by two programmable prescalers controlled by bitfields BPS2 and T6I in register T6CON. Please see **Section 14.2.6** for details on the input clock options.

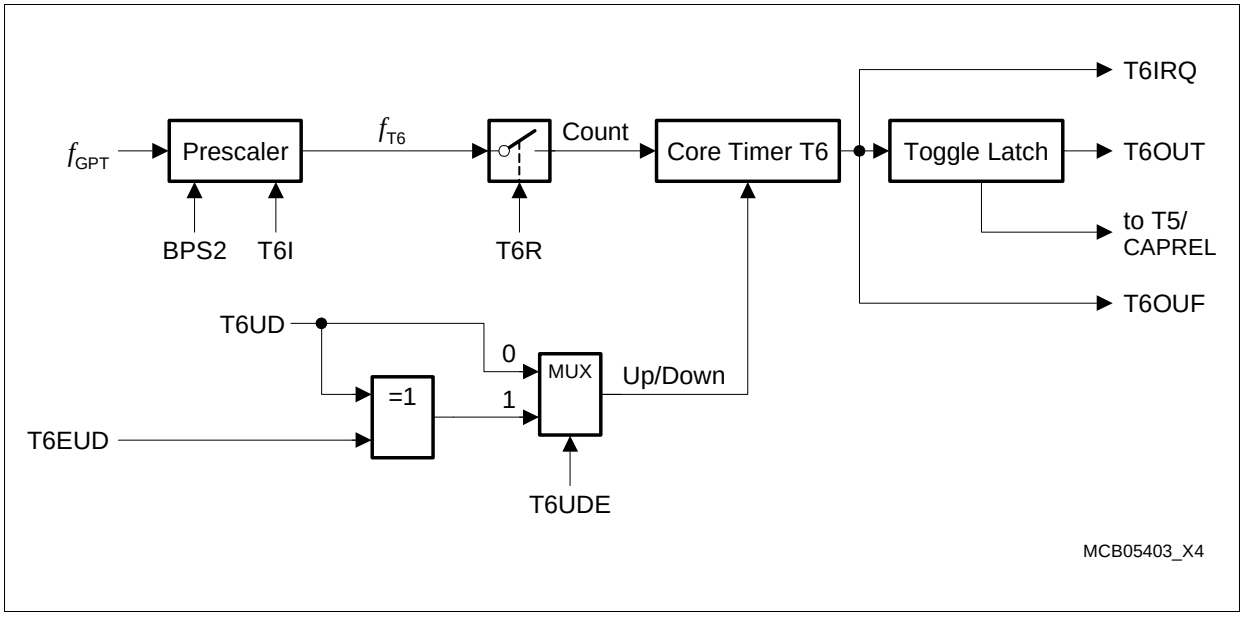


Figure 14-22 Block Diagram of Core Timer T6 in Timer Mode

Gated Timer Mode

Gated timer mode for the core timer T6 is selected by setting bitfield T6M in register T6CON to 010_B or 011_B. Bit T6M.0 (T6CON.3) selects the active level of the gate input. The same options for the input frequency are available in gated timer mode as in timer mode (see **Section 14.2.6**). However, the input clock to the timer in this mode is gated by the external input pin T6IN (Timer T6 External Input).
To enable this operation, the associated pin T6IN must be configured as input (the corresponding direction control bit must contain 0).

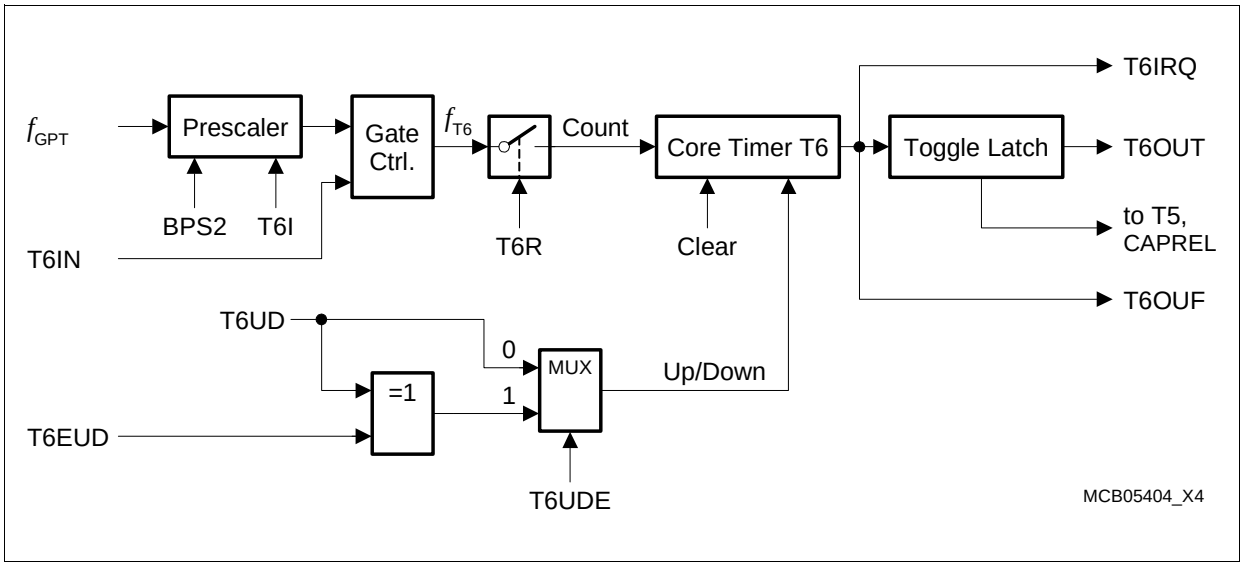


Figure 14-23 Block Diagram of Core Timer T6 in Gated Timer Mode

If T6M = 010_B, the timer is enabled when T6IN shows a low level. A high level at this line stops the timer. If T6M = 011_B, line T6IN must have a high level in order to enable the timer. Additionally, the timer can be turned on or off by software using bit T6R. The timer will only run if T6R is 1 and the gate is active. It will stop if either T6R is 0 or the gate is inactive.

Note: A transition of the gate signal at pin T6IN does not cause an interrupt request.

Counter Mode

Counter mode for the core timer T6 is selected by setting bitfield T6M in register T6CON to 001_B. In counter mode, timer T6 is clocked by a transition at the external input pin T6IN. The event causing an increment or decrement of the timer can be a positive, a negative, or both a positive and a negative transition at this line. Bitfield T6I in control register T6CON selects the triggering transition (see **Table 14-11**).

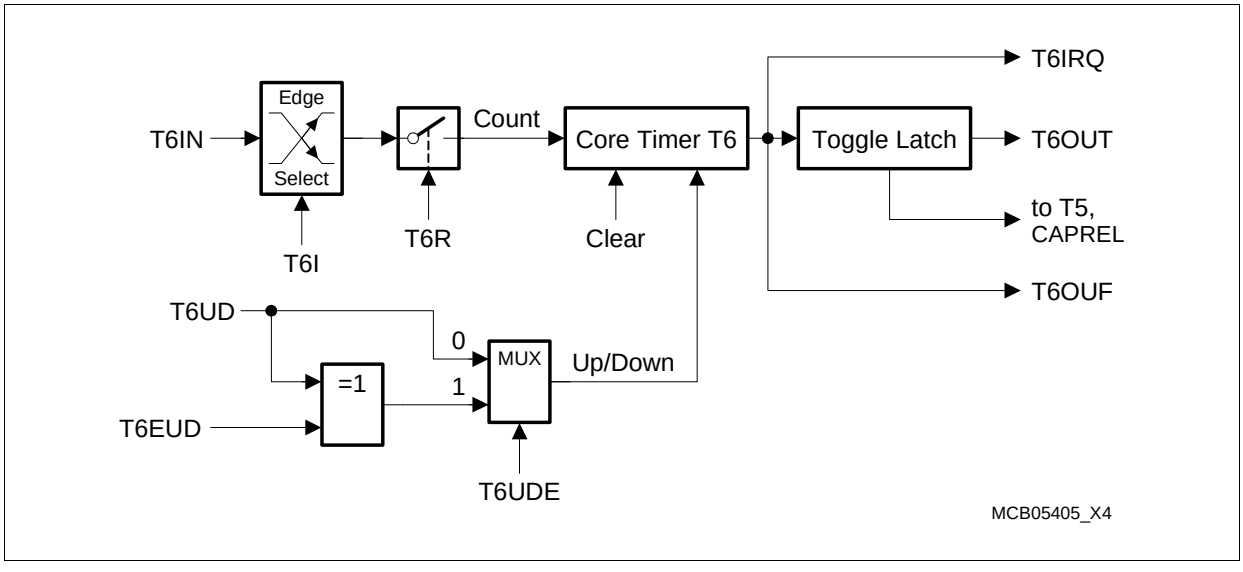


Figure 14-24 Block Diagram of Core Timer T6 in Counter Mode

Table 14-11 GPT2 Core Timer T6 (Counter Mode) Input Edge Selection

T6I	Triggering Edge for Counter Increment/Decrement
0 0 0	None. Counter T6 is disabled
0 0 1	Positive transition (rising edge) on T6IN
0 1 0	Negative transition (falling edge) on T6IN
0 1 1	Any transition (rising or falling edge) on T6IN
1 X X	Reserved. Do not use this combination

For counter mode operation, pin T6IN must be configured as input (the respective direction control bit DPx.y must be 0). The maximum input frequency allowed in counter mode depends on the selected prescaler value. To ensure that a transition of the count input signal applied to T6IN is recognized correctly, its level must be held high or low for a minimum number of module clock cycles before it changes. This information can be found in **Section 14.2.6**.

14.2.3 GPT2 Auxiliary Timer T5 Control

Auxiliary timer T5 can be configured for timer mode, gated timer mode, or counter mode with the same options for the timer frequencies and the count signal as the core timer T6. In addition to these 3 counting modes, the auxiliary timer can be concatenated with the core timer. The contents of T5 may be captured to register CAPREL upon an external or an internal trigger. The start/stop function of the auxiliary timers can be remotely controlled by the T6 run control bit. Several timers may thus be controlled synchronously.

The current contents of the auxiliary timer are reflected by its count register T5. This register can also be written to by the CPU, for example, to set the initial start value.

The individual configurations for timer T5 are determined by its bitaddressable control register T5CON. Some bits in this register also control the function of the CAPREL register. Note that functions which are present in all timers of block GPT2 are controlled in the same bit positions and in the same manner in each of the specific control registers.

Note: The auxiliary timer has no output toggle latch and no alternate output function.

GPT12E_T5CON
Timer 5 Control Register **SFR ($FF46_H$/$A3_H$)** **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T5SC	**T5CLR**	**CI**		**T5CC**	**CT3**	**T5RC**	**T5UDE**	**T5UD**	**T5R**	**T5M**			**T5I**		
rw	rw	rw		rw	rw	rw	rw	rw	rw	rw			rw		

Field	Bits	Type	Description
T5SC	15	rw	**Timer 5 Capture Mode Enable** 0 Capture into register CAPREL Disabled 1 Capture into register CAPREL Enabled
T5CLR	14	rw	**Timer T5 Clear Enable Bit** 0 Timer T5 is not cleared on a capture event 1 Timer T5 is cleared on a capture event

Field	Bits	Type	Description
CI	[13:12]	rw	**Register CAPREL Capture Trigger Selection** (depending on bit CT3) 00 Capture disabled 01 Positive transition (rising edge) on CAPIN or any transition on T3IN 10 Negative transition (falling edge) on CAPIN or any transition on T3EUD 11 Any transition (rising or falling edge) on CAPIN or any transition on T3IN or T3EUD
T5CC	11	rw	**Timer T5 Capture Correction** 0 T5 is just captured without any correction 1 T5 is decremented by 1 before being captured
CT3	10	rw	**Timer T3 Capture Trigger Enable** 0 Capture trigger from input line CAPIN 1 Capture trigger from T3 input lines T3IN and/or T3EUD
T5RC	9	rw	**Timer T5 Remote Control** 0 Timer T5 is controlled by its own run bit T5R 1 Timer T5 is controlled by the run bit T6R of core timer 6, not by bit T5R
T5UDE	8	rw	**Timer T5 External Up/Down Enable**[1)] 0 Input T5EUD is disconnected 1 Direction influenced by input T5EUD
T5UD	7	rw	**Timer T5 Up/Down Control**[1)] 0 Timer T5 counts up 1 Timer T5 counts down
T5R	6	rw	**Timer T5 Run Bit** 0 Timer T5 stops 1 Timer T5 runs *Note: This bit only controls timer T5 if bit T5RC = 0.*
T5M	[5:3]	rw	**Timer T5 Mode Control** (Basic Operating Mode) 000 Timer Mode 001 Counter Mode 010 Gated Timer Mode with gate active low 011 Gated Timer Mode with gate active high 1XX Reserved. Do not use this combination

Field	Bits	Type	Description
T5I	[2:0]	rw	**Timer T5 Input Parameter Selection** Depends on the operating mode, see respective sections for encoding: **Table 14-15** for Timer Mode and Gated Timer Mode **Table 14-11** for Counter Mode

1) See **Table 14-10** for encoding of bits T5UD and T5UDE.

Timer T5 Run Control

The auxiliary timer T5 can be started or stopped by software in two different ways:

- Through the associated timer run bit (T5R). In this case it is required that the respective control bit T5RC = 0.
- Through the core timer's run bit (T6R). In this case the respective remote control bit must be set (T5RC = 1).

The selected run bit is relevant in all operating modes of T5. Setting the bit will start the timer, clearing the bit stops the timer.

In gated timer mode, the timer will only run if the selected run bit is set and the gate is active (high or low, as programmed).

Note: If remote control is selected T6R will start/stop timer T6 and the auxiliary timer T5 synchronously.

14.2.4 GPT2 Auxiliary Timer T5 Operating Modes

The operation of the auxiliary timer in the basic operating modes is almost identical with the core timer's operation, with very few exceptions. Additionally, some combined operating modes can be selected.

Timer T5 in Timer Mode

Timer Mode for the auxiliary timer T5 is selected by setting its bitfield T5M in register T5CON to 000_B.

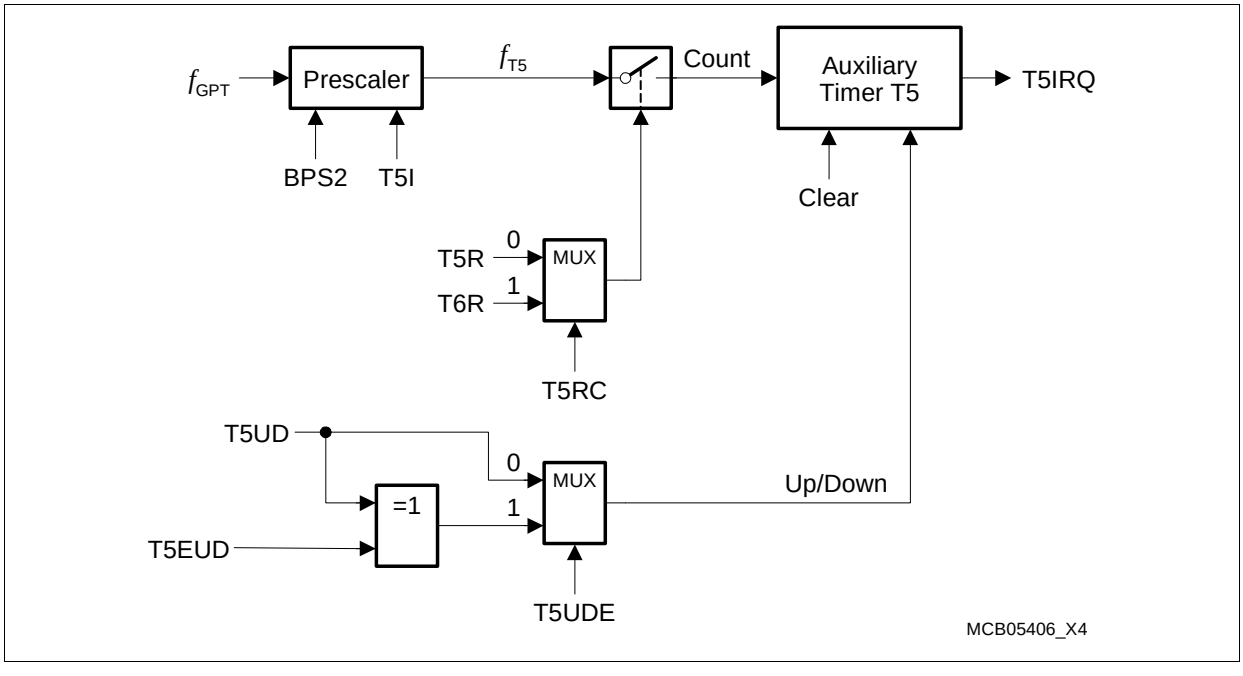


Figure 14-25 Block Diagram of Auxiliary Timer T5 in Timer Mode

Timer T5 in Gated Timer Mode

Gated timer mode for the auxiliary timer T5 is selected by setting bitfield T5M in register T5CON to 010_B or 011_B. Bit T5M.0 (T5CON.3) selects the active level of the gate input.

Note: A transition of the gate signal at line T5IN does not cause an interrupt request.

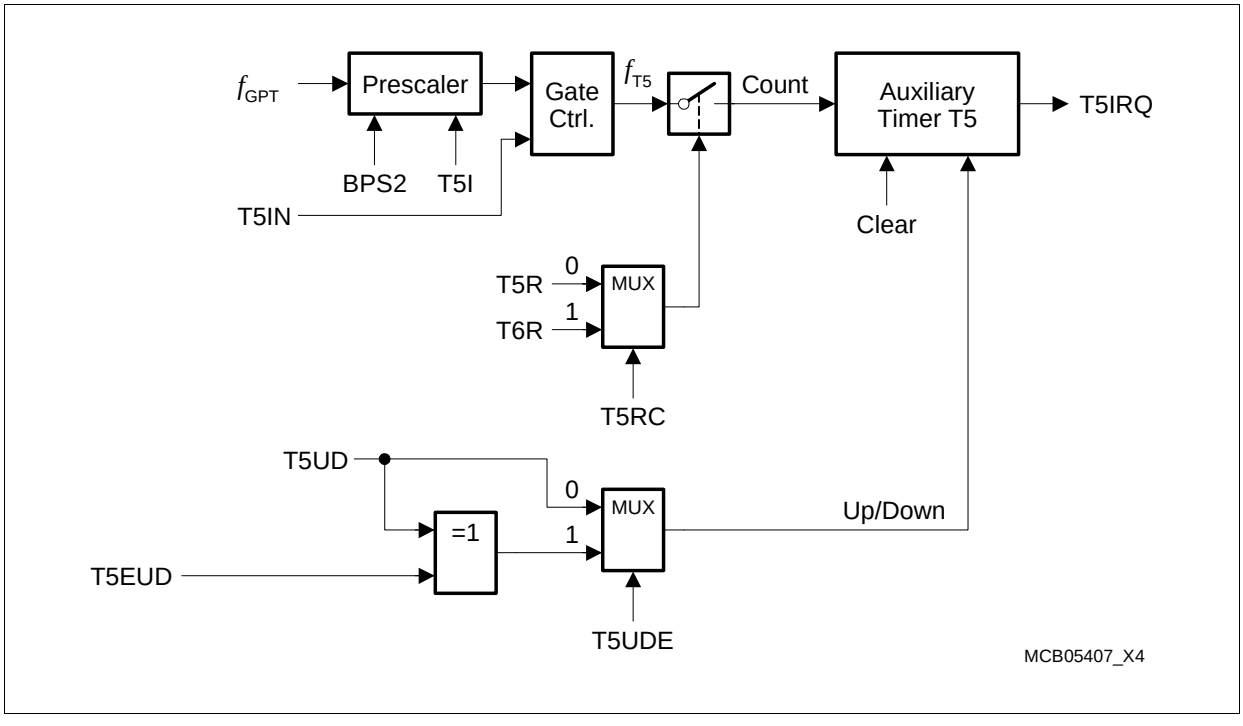


Figure 14-26 Block Diagram of Auxiliary Timer T5 in Gated Timer Mode

Note: There is no output toggle latch for T5.
Start/stop of the auxiliary timer can be controlled locally or remotely.

Timer T5 in Counter Mode

Counter mode for auxiliary timer T5 is selected by setting bitfield T5M in register T5CON to 001_B. In counter mode, the auxiliary timer can be clocked either by a transition at its external input line T5IN, or by a transition of timer T6's toggle latch T6OTL. The event causing an increment or decrement of a timer can be a positive, a negative, or both a positive and a negative transition at either the respective input pin or at the toggle latch. Bitfield T5I in control register T5CON selects the triggering transition (see **Table 14-12**).

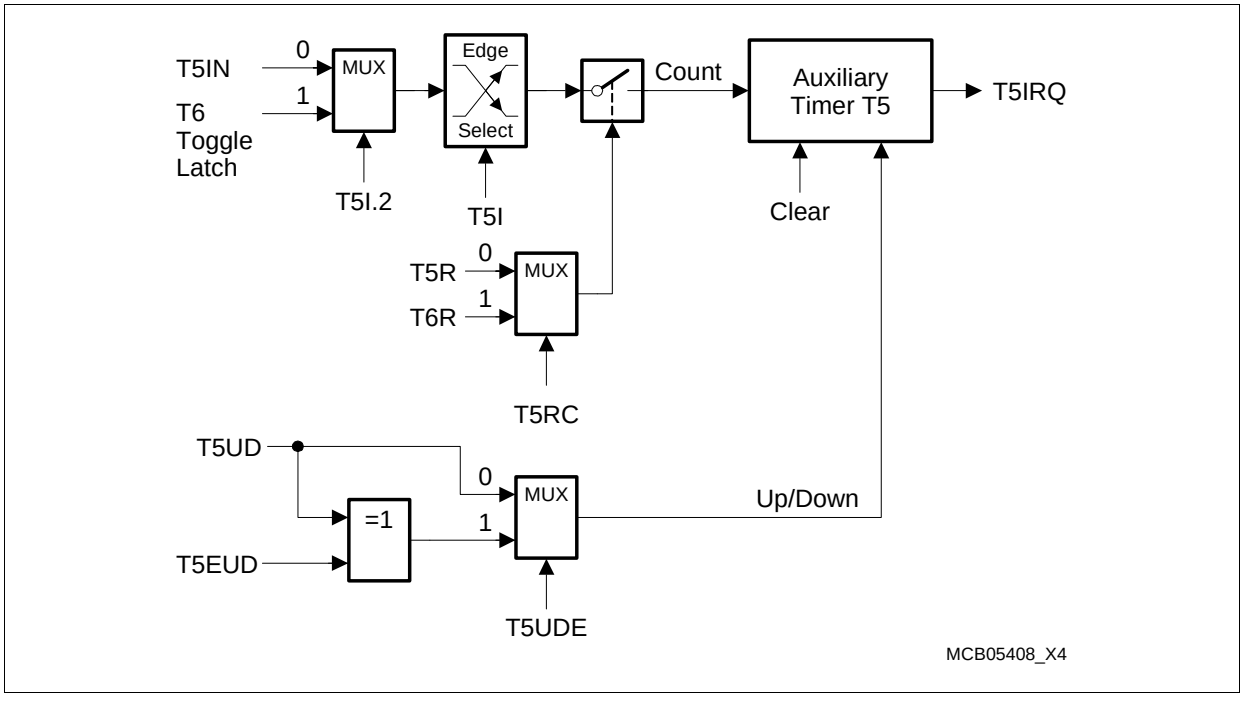


Figure 14-27 Block Diagram of Auxiliary Timer T5 in Counter Mode

Table 14-12 GPT2 Auxiliary Timer (Counter Mode) Input Edge Selection

T5I	Triggering Edge for Counter Increment/Decrement
X 0 0	None. Counter T5 is disabled
0 0 1	Positive transition (rising edge) on T5IN
0 1 0	Negative transition (falling edge) on T5IN
0 1 1	Any transition (rising or falling edge) on T5IN
1 0 1	Positive transition (rising edge) of T6 toggle latch T6OTL
1 1 0	Negative transition (falling edge) of T6 toggle latch T6OTL
1 1 1	Any transition (rising or falling edge) of T6 toggle latch T6OTL

Note: Only state transitions of T6OTL which are caused by the overflows/underflows of T6 will trigger the counter function of T5. Modifications of T6OTL via software will NOT trigger the counter function of T5.

For counter operation, pin T5IN must be configured as input (the respective direction control bit DPx.y must be 0). The maximum input frequency allowed in counter mode depends on the selected prescaler value. To ensure that a transition of the count input signal applied to T5IN is recognized correctly, its level must be held high or low for a minimum number of module clock cycles before it changes. This information can be found in **Section 14.2.6**.

Timer Concatenation

Using the toggle bit T6OTL as a clock source for the auxiliary timer in counter mode concatenates the core timer T6 with the auxiliary timer T5. This concatenation forms either a 32-bit or a 33-bit timer/counter, depending on which transition of T6OTL is selected to clock the auxiliary timer.

- **32-bit Timer/Counter:** If both a positive and a negative transition of T6OTL are used to clock the auxiliary timer, this timer is clocked on every overflow/underflow of the core timer T6. Thus, the two timers form a 32-bit timer.
- **33-bit Timer/Counter:** If either a positive or a negative transition of T6OTL is selected to clock the auxiliary timer, this timer is clocked on every second overflow/underflow of the core timer T6. This configuration forms a 33-bit timer (16-bit core timer + T6OTL + 16-bit auxiliary timer).
  As long as bit T6OTL is not modified by software, it represents the state of the internal toggle latch, and can be regarded as part of the 33-bit timer.

The count directions of the two concatenated timers are not required to be the same. This offers a wide variety of different configurations.

T6, which represents the low-order part of the concatenated timer, can operate in timer mode, gated timer mode or counter mode in this case.

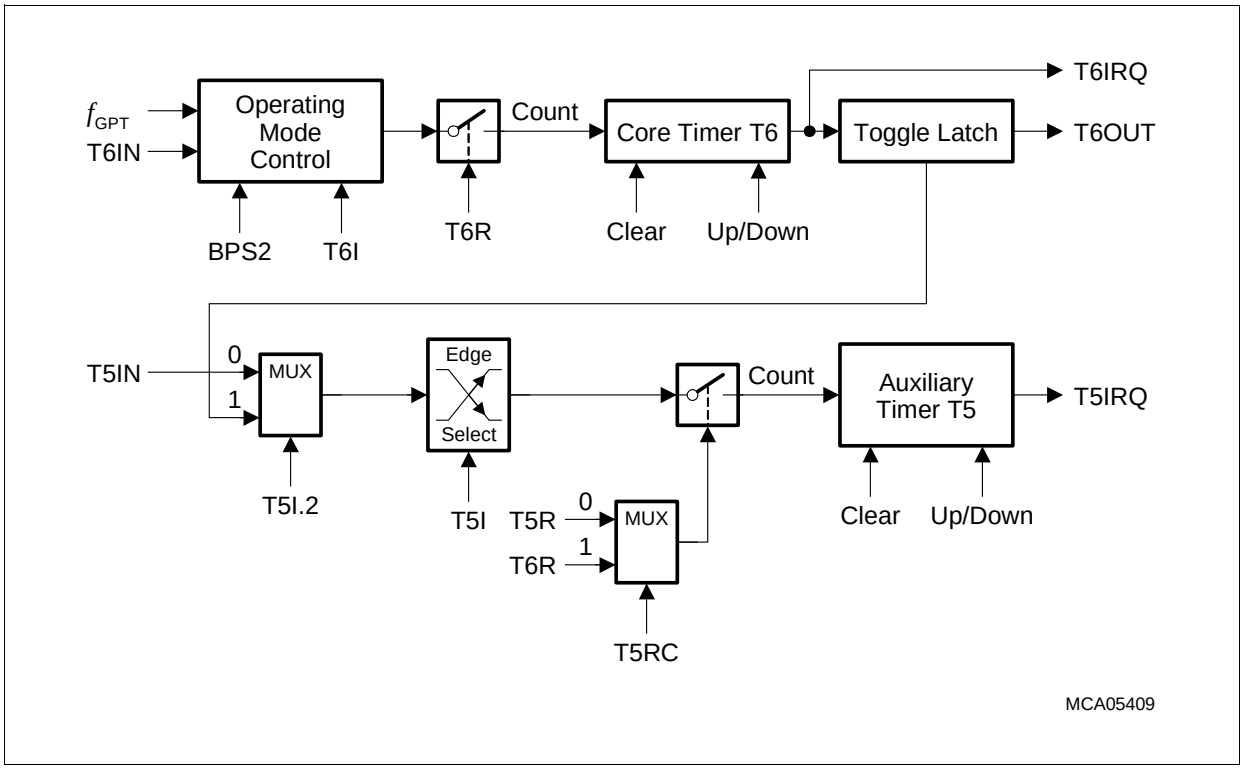


Figure 14-28 Concatenation of Core Timer T6 and Auxiliary Timer T5

14.2.5 GPT2 Register CAPREL Operating Modes

The Capture/Reload register CAPREL can be used to capture the contents of timer T5, or to reload timer T6. A special mode facilitates the use of register CAPREL for both functions at the same time. This mode allows frequency multiplication. The capture function is triggered by the input pin CAPIN, or by GPT1 timer's T3 input lines T3IN and T3EUD. The reload function is triggered by an overflow or underflow of timer T6.

In addition to the capture function, the capture trigger signal can also be used to clear the contents of timers T5 and T6 individually.

The functions of register CAPREL are controlled via several bit(field)s in the timer control registers T5CON and T6CON.

GPT2 Capture/Reload Register CAPREL in Capture Mode

Capture mode for register CAPREL is selected by setting bit T5SC in control register T5CON (set bitfield CI in register T5CON to a non-zero value to select a trigger signal). In capture mode, the contents of the auxiliary timer T5 are latched into register CAPREL in response to a signal transition at the selected external input pin(s). Bit CT3 selects the external input line CAPIN or the input lines T3IN and/or T3EUD of GPT1 timer T3 as the source for a capture trigger. Either a positive, a negative, or both a positive and a negative transition at line CAPIN can be selected to trigger the capture function, or transitions on input T3IN or input T3EUD or both inputs, T3IN and T3EUD. The active edge is controlled by bitfield CI in register T5CON. **Table 14-13** summarizes these options.

Table 14-13 CAPREL Register Input Edge Selection

CT3	CI	Triggering Signal/Edge for Capture Mode
X	0 0	None. Capture Mode is disabled.
0	0 1	Positive transition (rising edge) on CAPIN.
0	1 0	Negative transition (falling edge) on CAPIN.
0	1 1	Any transition (rising or falling edge) on CAPIN.
1	0 1	Any transition (rising or falling edge) on T3IN.
1	1 0	Any transition (rising or falling edge) on T3EUD.
1	1 1	Any transition (rising or falling edge) on T3IN or T3EUD.

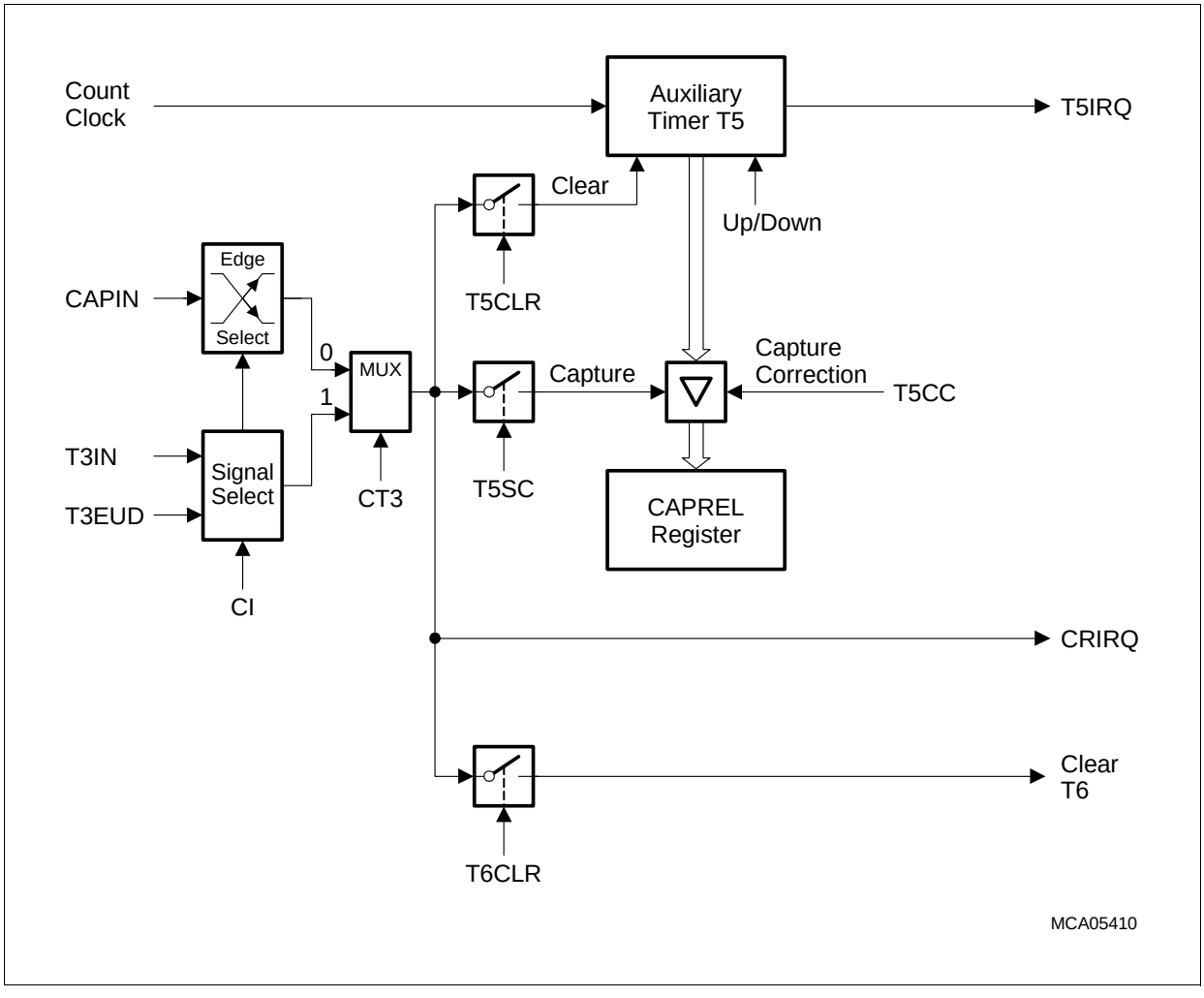


Figure 14-29 GPT2 Register CAPREL in Capture Mode

When a selected trigger is detected, the contents of the auxiliary timer T5 are latched into register CAPREL and the interrupt request line CRIRQ is activated. The same event can optionally clear timer T5 and/or timer T6. This option is enabled by bit T5CLR in register T5CON and bit T6CLR in register T6CON, respectively. If TxCLR = 0 the contents of timer Tx is not affected by a capture. If TxCLR = 1 timer Tx is cleared after the current timer T5 value has been latched into register CAPREL.

Note: Bit T5SC only controls whether or not a capture is performed. If T5SC is cleared the external input pin(s) can still be used to clear timer T5 and/or T6, or as external interrupt input(s). This interrupt is controlled by the CAPREL interrupt control register CRIC.

When capture triggers T3IN or T3EUD are enabled (CT3 = 1), register CAPREL captures the contents of T5 upon transitions of the selected input(s). These values can be used to measure T3's input signals. This is useful, for example, when T3 operates in

incremental interface mode, in order to derive dynamic information (speed, acceleration) from the input signals.

For capture mode operation, the selected pins CAPIN, T3IN, or T3EUD must be configured as input. To ensure that a transition of a trigger input signal applied to one of these inputs is recognized correctly, its level must be held high or low for a minimum number of module clock cycles, detailed in **Section 14.2.6**.

GPT2 Capture/Reload Register CAPREL in Reload Mode

Reload mode for register CAPREL is selected by setting bit T6SR in control register T6CON. In reload mode, the core timer T6 is reloaded with the contents of register CAPREL, triggered by an overflow or underflow of T6. This will not activate the interrupt request line CRIRQ associated with the CAPREL register. However, interrupt request line T6IRQ will be activated, indicating the overflow/underflow of T6.

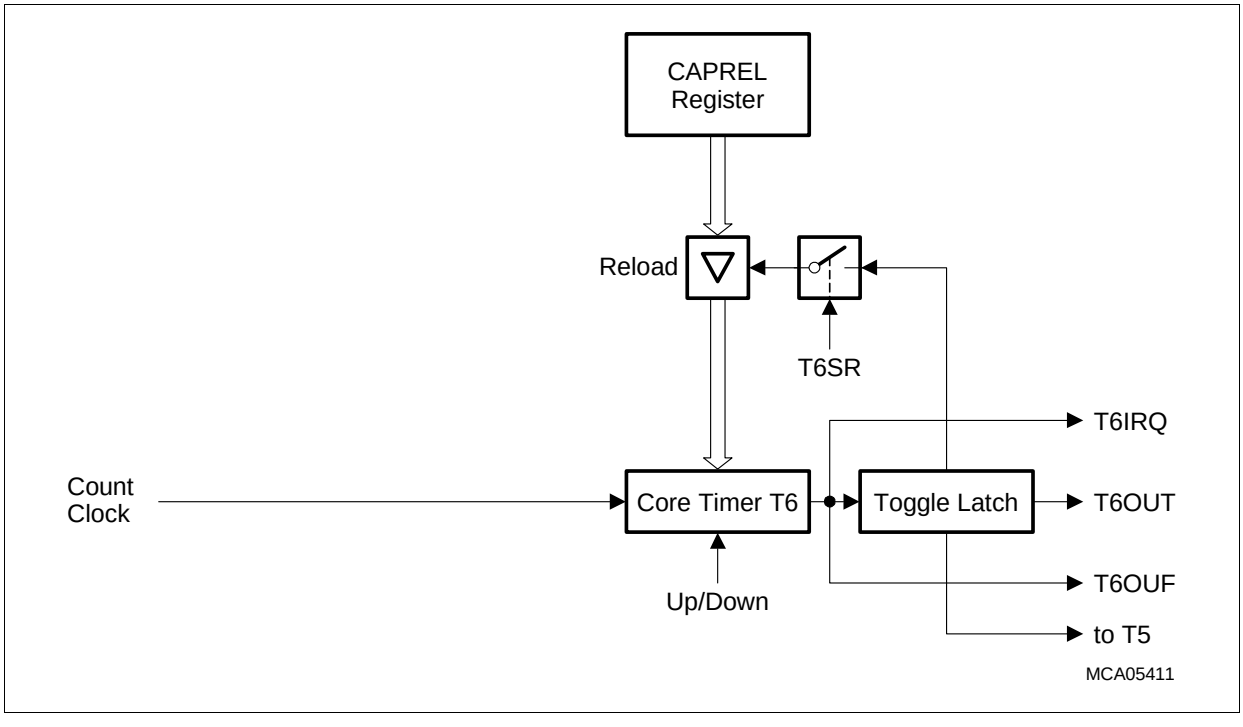


Figure 14-30 GPT2 Register CAPREL in Reload Mode

GPT2 Capture/Reload Register CAPREL in Capture-And-Reload Mode

Since the reload function and the capture function of register CAPREL can be enabled individually by bits T5SC and T6SR, the two functions can be enabled simultaneously by setting both bits. This feature can be used to generate an output frequency that is a multiple of the input frequency.

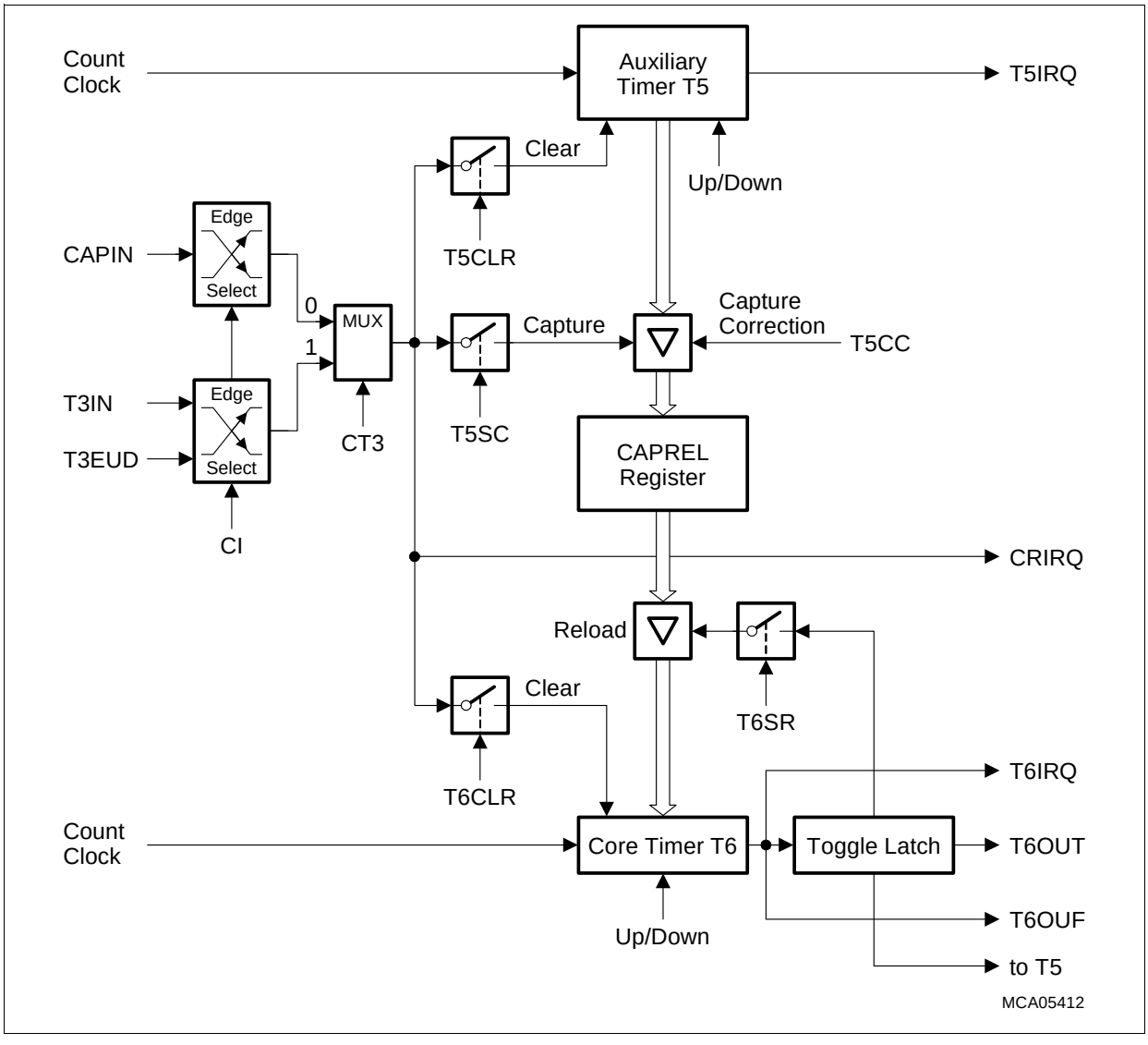


Figure 14-31 GPT2 Register CAPREL in Capture-And-Reload Mode

This combined mode can be used to detect consecutive external events which may occur aperiodically, but where a finer resolution, that means, more ‘ticks’ within the time between two external events is required.

For this purpose, the time between the external events is measured using timer T5 and the CAPREL register. Timer T5 runs in timer mode counting up with a frequency of e.g. $f_{GPT}/32$. The external events are applied to pin CAPIN. When an external event occurs,

the contents of timer T5 are latched into register CAPREL and timer T5 is cleared (T5CLR = 1). Thus, register CAPREL always contains the correct time between two events, measured in timer T5 increments. Timer T6, which runs in timer mode counting down with a frequency of e.g. $f_{GPT}/4$, uses the value in register CAPREL to perform a reload on underflow. This means, the value in register CAPREL represents the time between two underflows of timer T6, now measured in timer T6 increments. Since (in this example) timer T6 runs 8 times faster than timer T5, it will underflow 8 times within the time between two external events. Thus, the underflow signal of timer T6 generates 8 'ticks'. Upon each underflow, the interrupt request line T6IRQ will be activated and bit T6OTL will be toggled. The state of T6OTL may be output on pin T6OUT. This signal has 8 times more transitions than the signal which is applied to pin CAPIN.

Note: The underflow signal of Timer T6 can furthermore be used to clock one or more of the timers of the CAPCOM units, which gives the user the possibility to set compare events based on a finer resolution than that of the external events. This connection is accomplished via signal T6OUF.

Capture Correction

A certain deviation of the output frequency is generated by the fact that timer T5 will count actual time units (e.g. T5 running at 1 MHz will count up to the value $64_H/100_D$ for a 10 kHz input signal), while T6OTL will only toggle upon an underflow of T6 (i.e. the transition from 0000_H to $FFFF_H$). In the above mentioned example, T6 would count down from 64_H, so the underflow would occur after 101 timing ticks of T6. The actual output frequency then is 79.2 kHz, instead of the expected 80 kHz.

This deviation can be compensated for by activating the Capture Correction (T5CC = 1). If capture correction is active, the contents of T5 are decremented by 1 before being captured. The described deviation is eliminated (in the example, T5 would count up to the value $64_H/100_D$, but the CAPREL register will capture the decremented value $63_H/99_D$, T6 would count exactly 100 ticks, and the output frequency is 80 kHz).

14.2.6 GPT2 Clock Signal Control

All actions within the timer block GPT2 are triggered by transitions of its basic clock. This basic clock is derived from the system clock by a basic block prescaler, controlled by bitfield BPS2 in register T6CON (see **Figure 14-20**). The count clock can be generated in two different ways:

- **Internal count clock**, derived from GPT2's basic clock via a programmable prescaler, is used for (gated) timer mode.
- **External count clock**, derived from the timer's input pin(s), is used for counter mode.

For both ways, the basic clock determines the maximum count frequency and the timer's resolution:

Table 14-14 Basic Clock Selection for Block GPT2

Block Prescaler[1)]	BPS2 = 01_B	BPS2 = 00_B[2)]	BPS2 = 11_B	BPS2 = 10_B
Prescaling Factor for GPT2: F(BPS2)	F(BPS2) = 2	F(BPS2) = 4	F(BPS2) = 8	F(BPS2) = 16
Maximum External Count Frequency	$f_{GPT}/4$	$f_{GPT}/8$	$f_{GPT}/16$	$f_{GPT}/32$
Input Signal Stable Time	$2 \times t_{GPT}$	$4 \times t_{GPT}$	$8 \times t_{GPT}$	$16 \times t_{GPT}$

1) Please note the non-linear encoding of bitfield BPS2.

2) Default after reset.

Internal Count Clock Generation

In timer mode and gated timer mode, the count clock for each GPT2 timer is derived from the GPT2 basic clock by a programmable prescaler, controlled by bitfield TxI in the respective timer's control register TxCON.

The count frequency f_{Tx} for a timer Tx and its resolution r_{Tx} are scaled linearly with lower clock frequencies, as can be seen from the following formula:

$$f_{Tx} = \frac{f_{GPT}}{F(BPS2) \times 2^{<TxI>}} \qquad r_{Tx}[\mu s] = \frac{F(BPS2) \times 2^{<TxI>}}{f_{GPT}[MHz]} \qquad [14.2]$$

The effective count frequency depends on the common module clock prescaler factor F(BPS2) as well as on the individual input prescaler factor $2^{<TxI>}$. **Table 14-15** summarizes the resulting overall divider factors for a GPT2 timer that result from these cascaded prescalers.

Table 14-15 GPT2 Overall Prescaler Factors for Internal Count Clock

Individual Prescaler for Tx	Common Prescaler for Module Clock[1)]			
	$BPS2 = 01_B$	$BPS2 = 00_B$	$BPS2 = 11_B$	$BPS2 = 10_B$
$TxI = 000_B$	2	4	8	16
$TxI = 001_B$	4	8	16	32
$TxI = 010_B$	8	16	32	64
$TxI = 011_B$	16	32	64	128
$TxI = 100_B$	32	64	128	256
$TxI = 101_B$	64	128	256	512
$TxI = 110_B$	128	256	512	1024
$TxI = 111_B$	256	512	1024	2048

1) Please note the non-linear encoding of bitfield BPS2.

Table 14-16 lists a timer's parameters (such as count frequency, resolution, and period) resulting from the selected overall prescaler factor and the applied system frequency. Note that some numbers may be rounded.

Table 14-16 GPT2 Timer Parameters

System Clock = 10 MHz			Overall Divider Factor	System Clock = 40 MHz		
Frequency	Resolution	Period		Frequency	Resolution	Period
5.0 MHz	200 ns	13.11 ms	2	20.0 MHz	50 ns	3.28 ms
2.5 MHz	400 ns	26.21 ms	4	10.0 MHz	100 ns	6.55 ms
1.25 MHz	800 ns	52.43 ms	8	5.0 MHz	200 ns	13.11 ms
625.0 kHz	1.6 µs	104.9 ms	16	2.5 MHz	400 ns	26.21 ms
312.5 kHz	3.2 µs	209.7 ms	32	1.25 MHz	800 ns	52.43 ms
156.25 kHz	6.4 µs	419.4 ms	64	625.0 kHz	1.6 µs	104.9 ms
78.125 kHz	12.8 µs	838.9 ms	128	312.5 kHz	3.2 µs	209.7 ms
39.06 kHz	25.6 µs	1.678 s	256	156.25 kHz	6.4 µs	419.4 ms
19.53 kHz	51.2 µs	3.355 s	512	78.125 kHz	12.8 µs	838.9 ms
9.77 kHz	102.4 µs	6.711 s	1024	39.06 kHz	25.6 µs	1.678 s
4.88 kHz	204.8 µs	13.42 s	2048	19.53 kHz	51.2 µs	3.355 s

External Count Clock Input

The external input signals of the GPT2 block are sampled with the GPT2 basic clock (see **Figure 14-20**). To ensure that a signal is recognized correctly, its current level (high or low) must be held active for at least one complete sampling period, before changing. A signal transition is recognized if two subsequent samples of the input signal represent different levels. Therefore, a minimum of two basic clock periods are required for the sampling of an external input signal. Thus, the maximum frequency of an input signal must not be higher than half the basic clock.

Table 14-17 summarizes the resulting requirements for external GPT2 input signals.

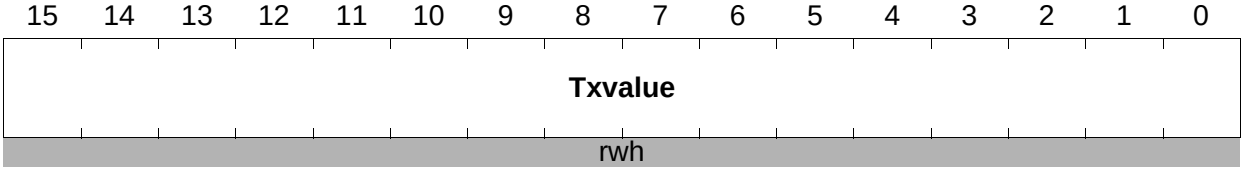
Table 14-17 GPT2 External Input Signal Limits

System Clock = 10 MHz		Input Frequ. Factor	GPT2 Divider BPS1	Input Phase Duration	System Clock = 40 MHz	
Max. Input Frequency	Min. Level Hold Time				Max. Input Frequency	Min. Level Hold Time
2.5 MHz	200 ns	$f_{GPT}/4$	01_B	$2 \times t_{GPT}$	10.0 MHz	50 ns
1.25 MHz	400 ns	$f_{GPT}/8$	00_B	$4 \times t_{GPT}$	5.0 MHz	100 ns
625.0 kHz	800 ns	$f_{GPT}/16$	11_B	$8 \times t_{GPT}$	2.5 MHz	200 ns
312.5 kHz	1.6 µs	$f_{GPT}/32$	10_B	$16 \times t_{GPT}$	1.25 MHz	400 ns

These limitations are valid for all external input signals to GPT2, including the external count signals in counter mode and the gate input signals in gated timer mode.

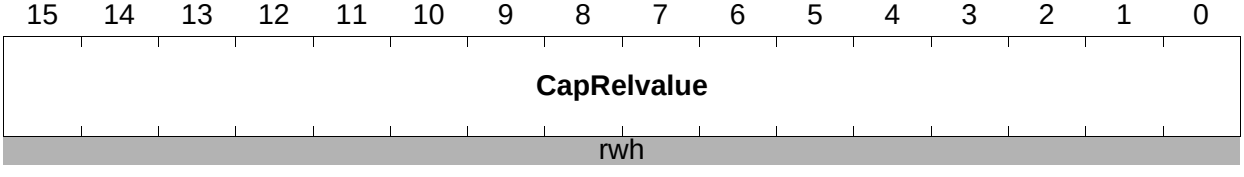
14.2.7 GPT2 Timer Registers

GPT12E_T5
Timer 5 Count Register **SFR ($FE46_H/23_H$)** **Reset Value: 0000_H**
GPT12E_T6
Timer 6 Count Register **SFR ($FE48_H/24_H$)** **Reset Value: 0000_H**



Field	Bits	Type	Description
Txvalue	[15:0]	rwh	**Current timer value**

GPT12E_CAPREL
Capture/Reload Register **SFR ($FE4A_H/25_H$)** **Reset Value: 0000_H**



Field	Bits	Type	Description
CapRel value	[15:0]	rwh	**Current reload value or Captured value**

14.2.8 Interrupt Control for GPT2 Timers and CAPREL

When a timer overflows from $FFFF_H$ to 0000_H (when counting up), or when it underflows from 0000_H to $FFFF_H$ (when counting down), its interrupt request flag (T5IR or T6IR) in register TxIC will be set. Whenever a transition according to the selection in bit field CI is detected at pin CAPIN, interrupt request flag CRIR in register CRIC is set. Setting any request flag will cause an interrupt to the respective timer or CAPREL interrupt vector (T5INT, T6INT or CRINT) or trigger a PEC service, if the respective interrupt enable bit (T5IE or T6IE in register TxIC, CRIE in register CRIC) is set. There is an interrupt control register for each of the two timers and for the CAPREL register.

GPT12E_T5IC
Timer 5 Intr. Ctrl. Reg. **SFR ($FF66_H/B3_H$)** **Reset Value: - - 00_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	GPX	T5IR	T5IE	ILVL				GLVL	
-	-	-	-	-	-	-	rw	rwh	rw	rw				rw	

GPT12E_T6IC
Timer 6 Intr. Ctrl. Reg. **SFR ($FF68_H/B4_H$)** **Reset Value: - - 00_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	GPX	T6IR	T6IE	ILVL				GLVL	
-	-	-	-	-	-	-	rw	rwh	rw	rw				rw	

GPT12E_CRIC
CAPREL Intr. Ctrl. Reg. **SFR ($FF6A_H/B5_H$)** **Reset Value: - - 00_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	GPX	CRIR	CRIE	ILVL				GLVL	
-	-	-	-	-	-	-	rw	rwh	rw	rw				rw	

Note: Please refer to the general Interrupt Control Register description for an explanation of the control fields.

14.2.9 KSCCFG Register

GPT12E_KSCCFG
Kernel State Configuration Register

SFR(FE1C$_H$) **Reset Value: 0000$_H$**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BP COM	**0**	**COMCFG**		**BP SUM**	**0**	**SUMCFG**		**BP NOM**	**0**	**NOMCFG**		**0**		**BP MOD EN**	**MOD EN**
w	r	rw		w	r	rw		w	r	rw		r		w	rw

Field	Bits	Type	Description
MODEN	0	rw	**Module Enable** This bit enables the module kernel clock and the module functionality. 0_B The module is switched off. It does not react on mode control actions and the module clock is switched off immediately (without stop condition). The module does not react on read accesses and ignores write accesses. 1_B The module is switched on. *Note: This bit is reset by an application reset.*
BPMODE	1	w	**Bit Protection for MODEN** This bit enables the write access to the bit MODEN. It always reads 0. It is only active during the write access cycle. 0_B MODEN is not changed. 1_B MODEN is updated with the written value. *Note: This bit is reset by an application reset.*
NOMCFG	[5:4]	rw	**Normal Operation Mode Configuration** This bit field defines the kernel mode applied in normal operation mode. $0X_B$ The module is switched on. $1X_B$ The module is switched off. This field is taken into account for CR = 00 or 11. *Note: This bit is reset by an application reset.*

Field	Bits	Type	Description
BPNOM	7	w	**Bit Protection for NOMCFG** This bit enables the write access to the bit field NOMCFG. It always reads 0. It is only active during the write access cycle. 0_B NOMCFG is not changed. 1_B NOMCFG is updated with the written value. *Note: This bit is reset by an application reset.*
SUMCFG	[9:8]	rw	**Suspend Mode Configuration** This bit field defines the kernel mode applied in suspend mode. $0X_B$ The module is switched on. $1X_B$ The module is switched off. This field is taken into account for CR = 01. *Note: This bit is reset by a debug reset.*
BPSUM	11	w	**Bit Protection for SUMCFG** This bit enables the write access to the bit field SUMCFG. It always reads 0. It is only active during the write access cycle. 0_B SUMCFG is not changed. 1_B SUMCFG is updated with the written value. *Note: This bit is reset by a debug reset.*
COMCFG	[13:12]	rw	**Clock Off Mode Configuration** This bit field defines the kernel mode applied in clock off mode. $0X_B$ The module is switched on. $1X_B$ The module is switched off. This field is taken into account for CR = 10. *Note: This bit is reset by an application reset.*
BPCOM	15	w	**Bit Protection for COMCFG** This bit enables the write access to the bit field COMCFG. It always reads 0. It is only active during the write access cycle. 0_B COMCFG is not changed. 1_B COMCFG is updated with the written value. *Note: This bit is reset by an application reset.*
0	[3:2], 6, 10, 14	r	**Reserved;** returns 0 if read; should be written with 0;

14.3 Interfaces of the GPT Module

Besides the described intra-module connections, the timer unit blocks GPT1 and GPT2 are connected to their environment in two basic ways (see **Figure 14-32**):

- **Internal connections** interface the timers with on-chip resources such as clock generation unit, interrupt controller, or other timers.
  The GPT module is clocked with the XC2000 system clock, so $f_{GPT} = f_{SYS}$.
- **External connections** interface the timers with external resources via port pins.

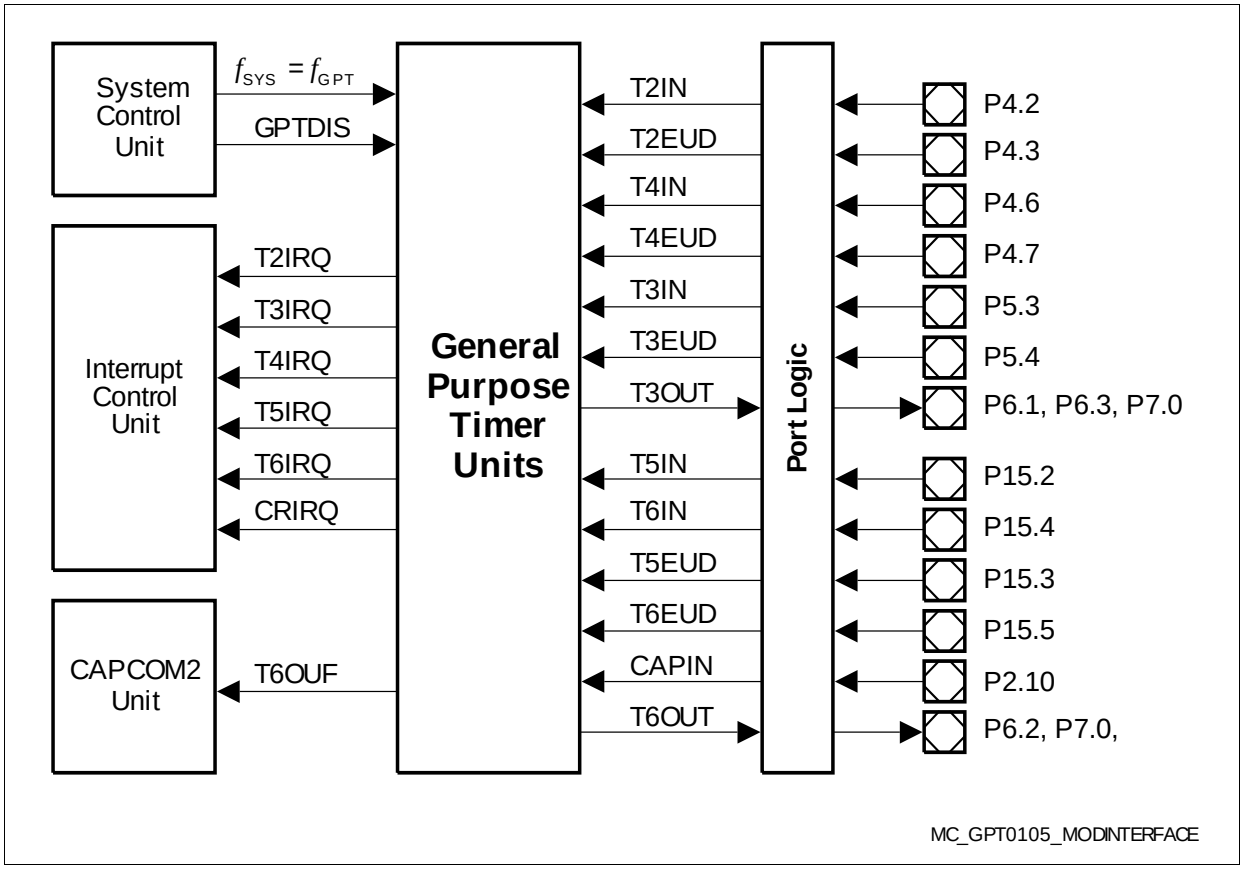


Figure 14-32 GPT Module Interfaces

Port pins to be used for timer input signals must be switched to input (bitfield PC in the respective port control register must be $0xxx_B$).

Port pins to be used for timer output signals must be switched to output and the alternate timer output signal must be selected (bitfield PC in the respective port control register must be $1xxx_B$).

*Note: For a description of the port control registers, please refer to **Chapter 7**.*

Interrupt nodes to be used for timer interrupt requests must be enabled and programmed to a specific interrupt level.

15 Real Time Clock

The Real Time Clock (RTC) module of the XC2000 basically consists of a chain of prescalers and timers. Its count clock is derived from the auxiliary oscillator or from the prescaled main oscillator. The RTC serves various purposes:

- 48-bit timer for long term measurements
- System clock to determine the current time and date
  (the RTC's structure supports the direct representation of time and date)
- Cyclic time based interrupt (can be generated by any timer of the chain)

A number of programming options as well as interrupt request signals adjust the operation of the RTC to the application's requirements. The RTC can continue its operation while the XC2000 is in certain power-saving modes, such that real time date and time information is provided.

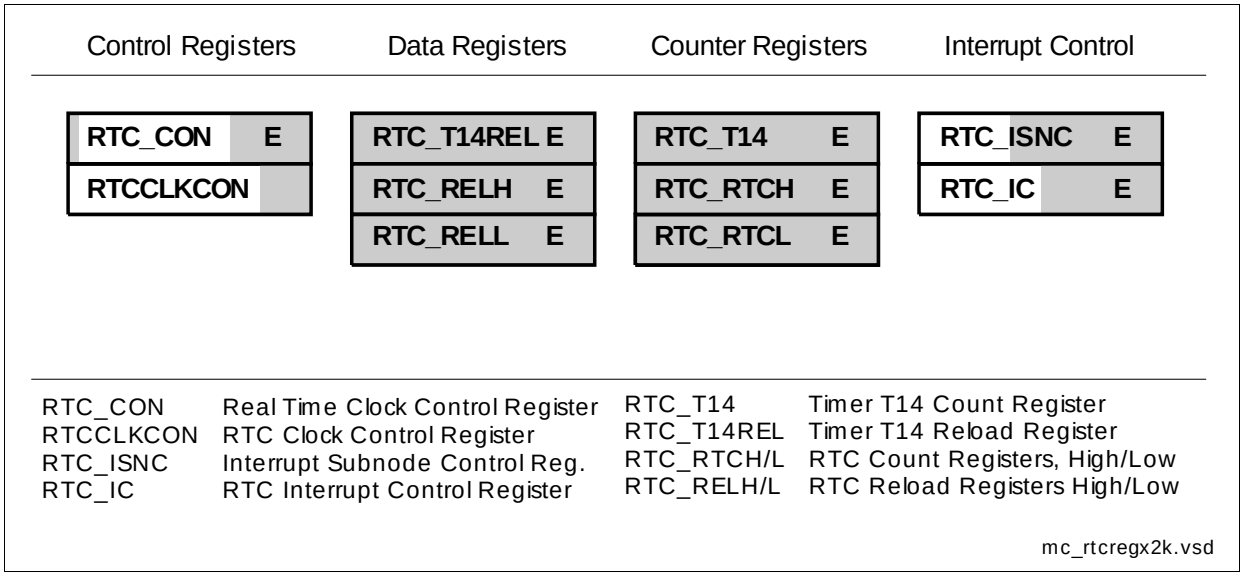


Figure 15-1 SFRs Associated with the RTC Module

The RTC module consists of a chain of 3 divider blocks:

- a selectable 8:1 divider (on - off)
- the reloadable 16-bit timer T14
- the 32-bit RTC timer block (accessible via RTC_RTCH and RTC_RTCL), made of:
  – the reloadable 10-bit timer CNT0
  – the reloadable 6-bit timer CNT1
  – the reloadable 6-bit timer CNT2
  – the reloadable 10-bit timer CNT3

All timers count upwards. Each of the five timers can generate an interrupt request. All requests are combined to a common node request.

Note: The RTC registers are not affected by a functional reset in order to maintain the correct system time even when intermediate resets are executed.

15.1 Defining the RTC Time Base

The timer chain of the RTC is clocked with the count clock signal f_{RTC} which is derived from internal sources (oscillators or PLL) or external sources (pins). The currently active clock source is selected by bitfield RTCCLKSEL in register RTCCLKCON. Optionally prescaled by a factor of 32 and/or 8, this is the basic RTC clock. Depending on the operating mode, timer T14 may provide the count increments used by the application and thus determine the input frequency of the RTC timer, that is, the RTC time base (see also **Table 15-2**).

The RTC is also supplied with the system clock f_{SYS} of the XC2000. This clock signal is used to control the RTC's logic blocks and its bus interface. To synchronize properly to the count clock, the system clock must run at least four times faster than the count clock, this means $f_{SYS} \geq 4 \times f_{CNT}$.

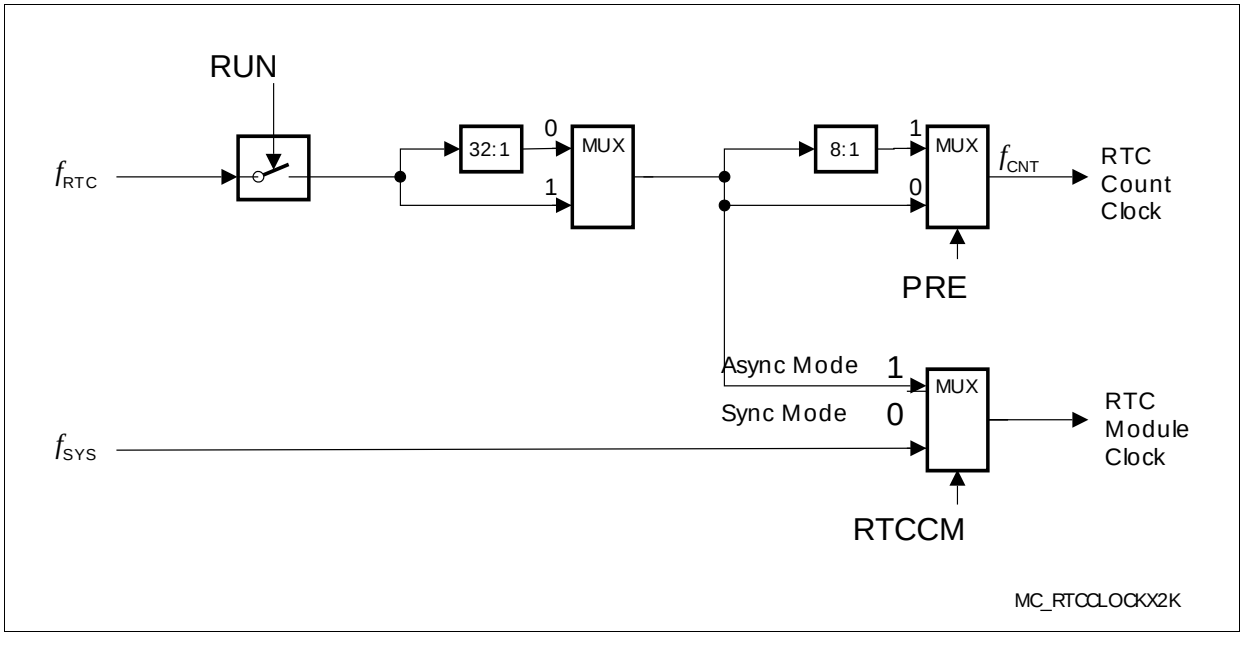


Figure 15-2 RTC Clock Supply Block Diagram

For an example, **Table 15-1** lists the interrupt period range and the T14 reload values (for a time base of 1 s and 1 ms):

Table 15-1 RTC Time Base Examples

Oscillator Frequency	T14 Intr. Period		Reload Value A		Reload Value B	
	Min.	Max.	T14REL	Base	T14REL	Base
32.768 kHz	30.52 µs	16.0 s	8000_H/$F000_H$	1.000 s	$FFDF_H$/ $FFFC_H$	1.007 ms/ 0.977 ms

Note: Select one value from the reload value pairs, depending if the 8:1 prescaler is disabled/enabled.

Asynchronous Operation

When the system clock frequency becomes lower than $4 \times f_{CNT}$ proper synchronization is not possible and count events may be missed. This can be the case, when the XC2000 reduces the system frequency to save power consumption.

In these cases the RTC can be switched to Asynchronous Mode (by clearing bit RTCCM in register RTCCLKCON). In this mode the count registers are directly controlled by the count clock independent of the system clock (hence the name). Asynchronous operation ensures correct time-keeping even during power-save modes.

However, as no synchronization between the count registers and the bus interface can be maintained in asynchronous mode, the RTC registers cannot be written. Read accesses may interfere with count events and, therefore, must be verified (e.g. by reading the same value with three consecutive read accesses).

Note: The access restrictions in asynchronous mode are only meaningful if the system clock is not switched off, of course.

Switching Clocking Modes

The clocking mode of the RTC (synchronous or asynchronous) is selected via bit RTCCM in register RTCCLKCON. After reset, the RTC operates in Synchronous Mode (RTCCM = 1) with the 8:1 prescaler enabled.

The selected clocking mode also affects the access to RTC registers. Bit ACCPOS in register RTC_CON indicates if full register access is possible (ACCPOS = 1, default after reset) or not (ACCPOS = 0). This also indicates the current clocking mode.

Attention: Software should poll bit ACCPOS to determine the proper transition to the intended clocking mode.

After switching to Asynchronous Mode (RTCCM = 0), bit ACCPOS = 0 indicates proper operation in Asynchronous Mode. In this case the system clock can be stopped or reduced.

After switching to Synchronous Mode, (RTCCM = 1), bit ACCPOS = 1 indicates proper operation in Synchronous Mode. In this case the RTC registers can again be accessed properly (read and write).

Note: The RTC might lose a counting event (edge of f_{CNT}) when switching from synchronous mode to asynchronous mode while the 8:1 prescaler is disabled. For these applications it is, therefore, recommended to set up the RTC with the 8:1 prescaler enabled.

Increased RTC Accuracy through Software Correction

The accuracy of the XC2000's RTC is determined by the oscillator frequency and by the respective prescaling factor (excluding or including T14 and the 8:1 prescaler). The accuracy limit generated by the prescaler is due to the quantization of a binary counter (where the average is zero), while the accuracy limit generated by the oscillator frequency is due to the difference between the ideal and real frequencies (and therefore accumulates over time). This effect is predictable and can be compensated. The total accuracy of the RTC can be further increased via software for specific applications that demand a high time accuracy.

The key to the improved accuracy is knowledge of the exact oscillator frequency. The relation of this frequency to the expected ideal frequency is a measure of the RTC's deviation. The number of cycles, N, after which this deviation causes an error of ±1 cycle can be easily computed. So, the only action is to correct the count by ±1 after each series of N cycles. The correction may be made cyclically, for instance, within an interrupt service routine, or by evaluating a formula when the RTC registers are read (for this the respective "last" RTC value must be available somewhere).

Note: For the majority of applications, however, the standard accuracy provided by the RTC's structure will be more than sufficient.

Adjusting the current RTC value would require reading and then writing the complete 48-bit value. This can only be accomplished by three successive accesses each. To avoid the hassle of reading/writing multi-word values, the RTC incorporates a correction option to simply add or suppress one count pulse.

This is done by setting bit T14INC or T14DEC, respectively, in register RTC_CON. This will add an extra count pulse (T14INC) upon the next count event, or suppress the next count event (T14DEC). The respective bit remains set until its associated action has been performed and is automatically cleared by hardware after this event.

Note: Setting both bits, T14INC and T14DEC, at the same time will have no effect on the count values.

15.2 RTC Run Control

If the RTC shall operate bit RUN in register RTC_CON must be set (default after reset). Bit RUN can be cleared, for example, to exclude certain operation phases from time keeping. The RTC can be completely disabled by clearing the corresponding bit MODEN in register RTC_KSCCFG.

Note: A valid count clock is required for proper RTC operation, of course.

The RTC is reset by a power reset, a functional reset does not affect the RTC registers and its operation (RTC_IC will be reset, however). The initialization software must ensure the proper RTC operating mode.

The RTC control register RTC_CON selects the basic operation of the RTC module.

RTC_CON
Control Register **ESFR ($F110_H/88_H$)** **Reset Value: 8003_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACC POS	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**-**	**REF CLK**	**T14 INC**	**T14 DEC**	**PRE**	**RUN**
rh	-	-	-	-	-	-	-	-	-	-	rw	rwh	rwh	rw	rw

Field	Bits	Type	Description
ACCPOS	15	rh	**RTC Register Access Possible** 0 No write access is possible, only asynchronous reads 1 Registers can be read and written
REFCLK	4	rw	**RTC Input Source Prescaler (32:1) Disable** 0 Input Prescaler enabled 1 Input Prescaler disabled
T14INC	3	rwh	**Increment Timer T14 Value** Setting this bit to 1 adds one count pulse upon the next count event, thus incrementing T14. This bit is cleared by hardware after incrementation.
T14DEC	2	rwh	**Decrement Timer T14 Value** Setting this bit to 1 suppresses the next count event, thus decrementing T14. This bit is cleared by hardware after decrementation.
PRE	1	rw	**RTC Input Source Prescaler (8:1) Enable** 0 Prescaler disabled 1 Prescaler enabled

Field	Bits	Type	Description
RUN	0	rw	**RTC Run Bit** 0 RTC stopped 1 RTC runs

15.3 RTC Operating Modes

The RTC can be configured for several operating modes according to the purpose it is meant to serve. These operating modes are configured by selecting appropriate reload values and interrupt signals.

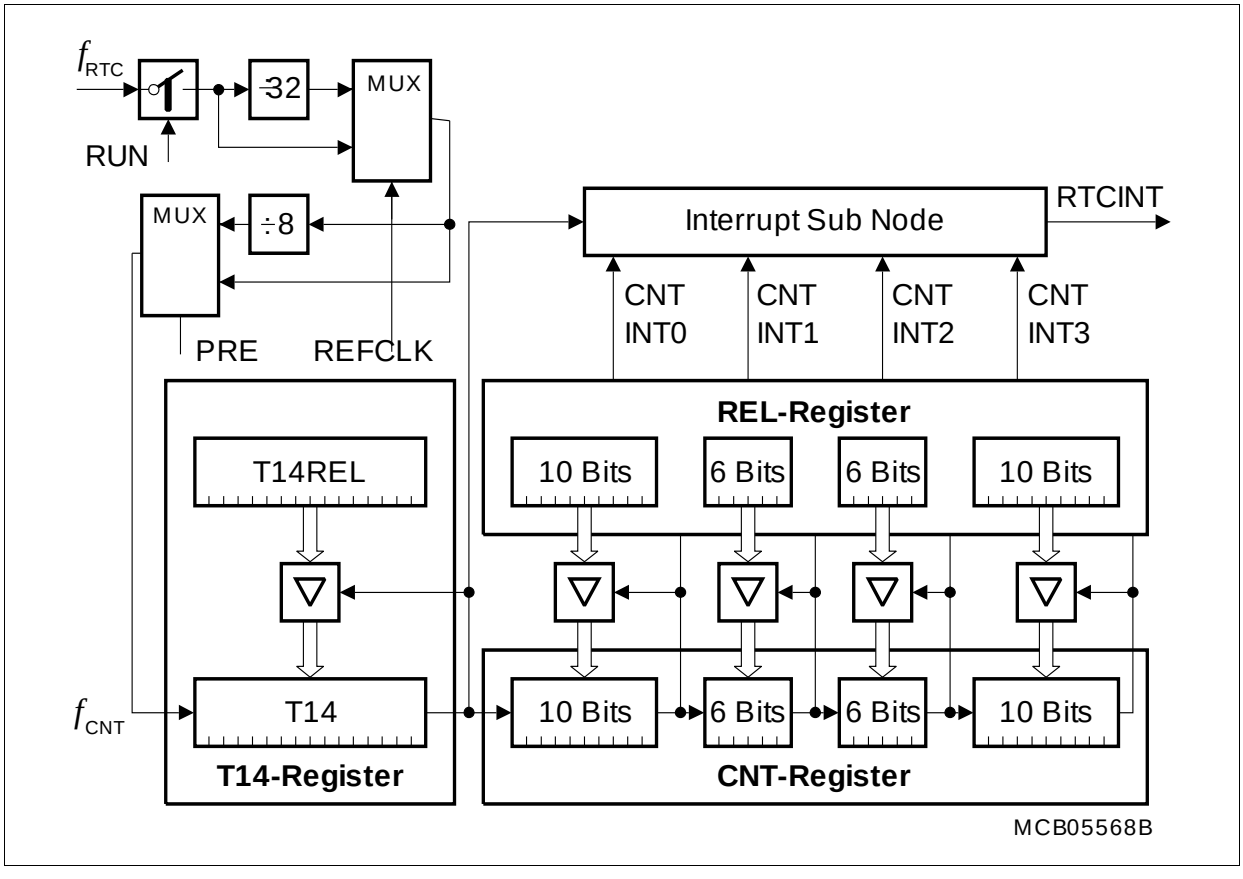


Figure 15-3 RTC Block Diagram

RTC Register Access

The actual value of the RTC is indicated by the three registers T14, RTCL, and RTCH. As these registers are concatenated to build the RTC counter chain, internal overflows occur while the RTC is running. When reading or writing the RTC value, such internal overflows must be taken into account to avoid reading/writing corrupted values.

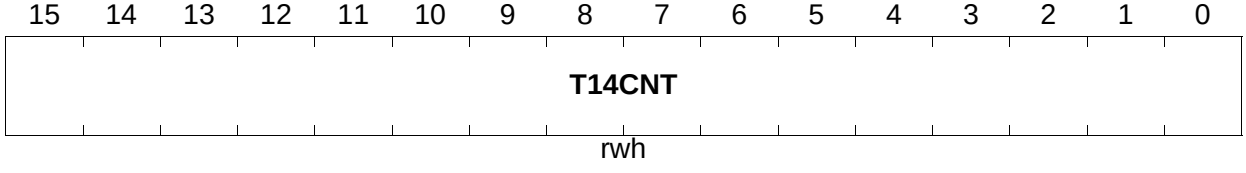
Care must be taken, when reading the timer(s), as this requires up to three read accesses to the different registers with an inherent time delay between the accesses. An overflow from T14 to RTCL and/or from RTCL to RTCH might occur between the accesses, which needs to be taken into account appropriately.

For example, reading/writing 0000_H from/to RTCH and then accessing RTCL could produce a corrupted value as RTCL may overflow before it can be accessed. In this case, RTCH would be 0001_H. The same precautions must be taken for T14 and T14REL.

Timer T14 and its reload register are accessed via dedicated locations. The four RTC counters CNT3 … CNT0 are accessed via the two 16-bit RTC timer registers, RTCH and RTCL. The associated four reload values REL3 … REL0 are accessed via the two 16-bit RTC reload registers, RELH and RELL.

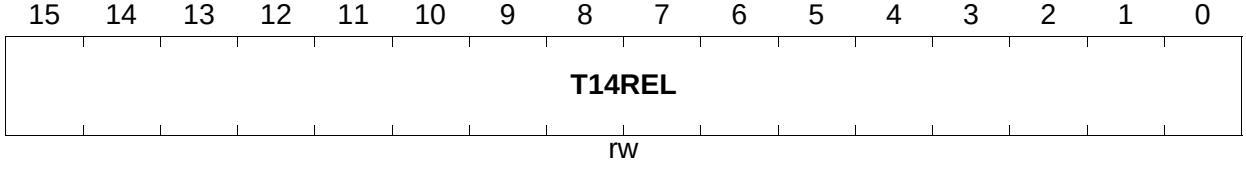
Prescaler T14 and T14 Reload Registers

RTC_T14
T14 Count Register **ESFR($F0D2_H$/69_H)** **Reset Value: 0000_H**



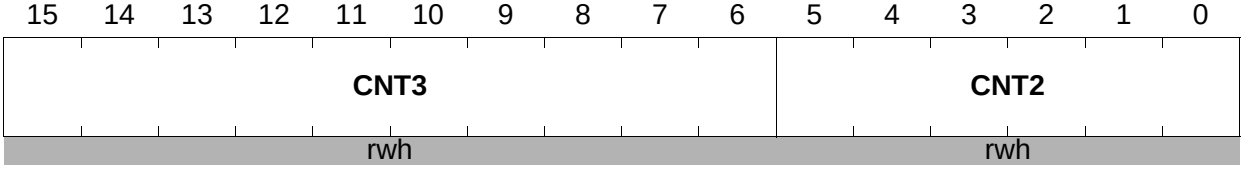
Field	Bits	Typ	Description
T14CNT	[15:0]	rwh	**T14 counter**

RTC_T14REL
T14 Reload Register **ESFR($F0D0_H$/68_H)** **Reset Value: 0000_H**



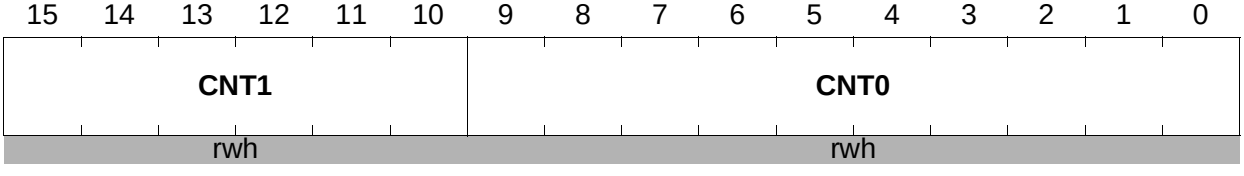
Field	Bits	Typ	Description
T14REL	[15:0]	rw	**T14 reload value**

RTC_RTCH
RTC Timer High Register **ESFR ($F0D6_H$/$6B_H$)** **Reset Value: 0000_H**



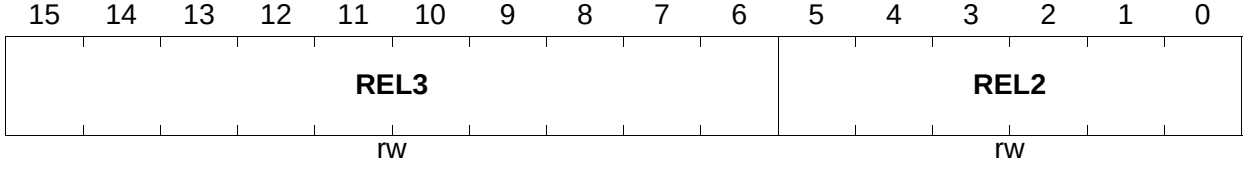
Field	Bits	Type	Description
CNTx (x = 3 … 2)	[15:6], [5:0]	rwh	**RTC Timer Count Section CNTx** An overflow of bitfield CNT2 triggers a count pulse to count section CNT3 followed by a reload of CNT2 from bitfield REL2. In addition, an interrupt request is triggered.

RTC_RTCL
RTC Timer Low Register **ESFR ($F0D4_H$/$6A_H$)** **Reset Value: 0000_H**



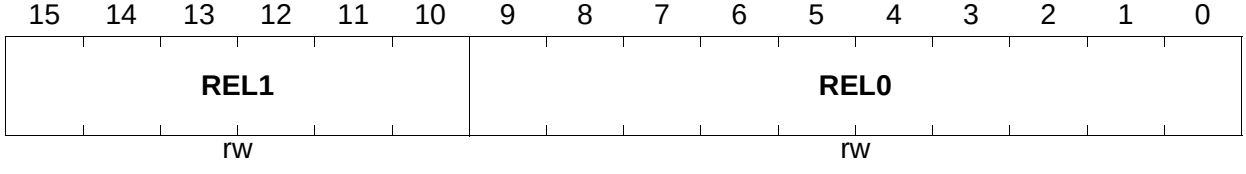
Field	Bits	Type	Description
CNTx (x = 1 … 0)	[15:10], [9:0]	rwh	**RTC Timer Count Section CNTx** An overflow of bitfield CNTx triggers a count pulse to the next count section CNTx+1 followed by a reload of CNTx from bitfield RELx. In addition, an interrupt request is triggered.

RTC_RELH
RTC Reload High Register **ESFR ($F0CE_H/67_H$)** **Reset Value: 0000_H**



Field	Bits	Type	Description
RELx (x = 3 … 2)	[15:6], [5:0]	rw	**RTC Reload Value RELx** This bitfield is copied to bitfield CNTx upon an overflow of count section CNTx.

RTC_RELL
RTC Reload Low Register **ESFR ($F0CC_H/66_H$)** **Reset Value: 0000_H**



Field	Bits	Type	Description
RELx (x = 1 … 0)	[15:10], [9:0]	rw	**RTC Reload Value RELx** This bitfield is copied to bitfield CNTx upon an overflow of count section CNTx.

Note: The registers of the RTC receive their reset values only upon a power reset.

15.4 48-bit Timer Operation

The concatenation of timers T14 and COUNT0 … COUNT3 can be regarded as a 48-bit timer which is clocked with the RTC input frequency, optionally divided by the prescaler. The reload registers T14REL, RELL, and RELH must be cleared to produce a true binary 48-bit timer. However, any other reload value may be used. Reload values other than zero must be used carefully, due to the individual sections of the RTC timer with their own individual overflows and reload values.

The maximum usable timespan is 2^{48} ($\approx 10^{14}$) T14 input clocks (assuming no prescaler), which would equal more than 200 years at an oscillator frequency of 32 kHz.

15.5 System Clock Operation

A real time system clock can be maintained that keeps on running also during power saving modes (optionally) and indicates the current time and date. This is possible because the RTC module is not affected by a functional reset. After a power reset, however, the RTC is reset and cleared.

The resolution for this clock information is determined by the input clock of timer T14. By selecting appropriate reload values each cascaded timer can represent directly a part of the current time and/or date. Due to its width, T14 can adjust the RTC to the intended range of operation (time or date). The maximum usable timespan is achieved when T14REL is loaded with 0000_H and so T14 divides by 2^{16}.

System Clock Example

The RTC count clock is f_{OSCa} (8:1 prescaler off). By selecting appropriate reload values the RTC timers directly indicate the current time (see **Figure 15-4** and **Table 15-2**).

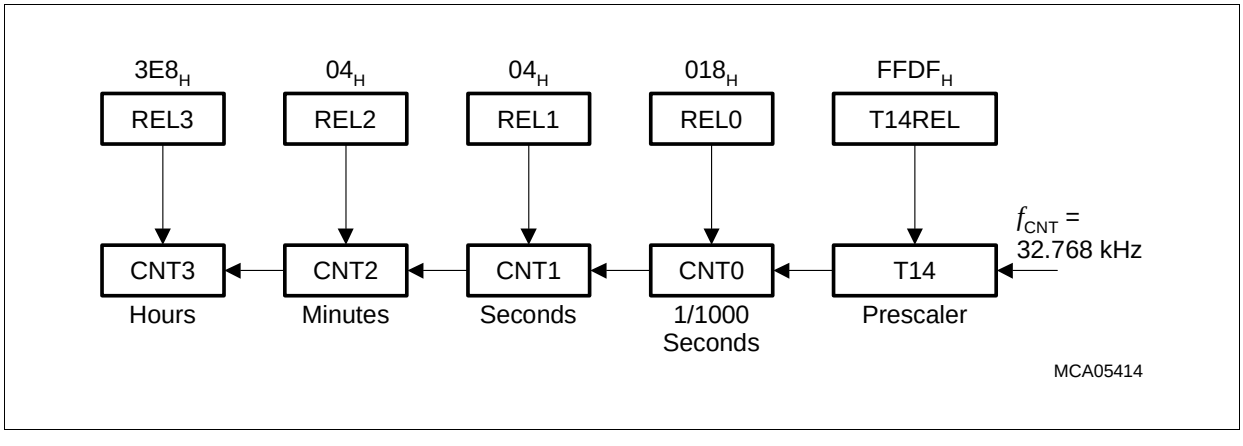


Figure 15-4 RTC Configuration Example

Note: This setup can generate an interrupt request every millisecond, every second, every minute, every hour, or every day.

Each timer in the chain divides the clock by ($2^{<timer\_width>}$ - <reload_value>) : 1, as the timers count up. **Table 15-2** shows the reload values which must be chosen for a specific scenario (i.e. operating mode of the RTC).

Table 15-2 Reload Value Scenarios

		REL3	REL2	REL1	REL0	T14REL
Time of Day (Figure 15-4)	**Formula**	2^{10} - 24	2^{6} - 60	2^{6} - 60	2^{10} - 1000	2^{16} - 33
	Rel. Value	$3E8_H$	04_H	04_H	018_H	$FFDF_H$
	Function	h	m	s	1/1000 s	Prescaler
	Intr. Period	day	hour	minute	second	millisec.[1]
Day of the Week	**Formula**	2^{10} - 7	2^{6} - 24	2^{6} - 60	2^{10} - 60	2^{16} - 32768
	Rel. Value	$3F9_H$	28_H	04_H	$3C4_H$	8000_H
	Function	day	h	m	s	Prescaler
	Intr. Period	week	day	hour	minute	second

1) T14 errors in the first example (ms) can be compensated either by choosing an adapted value for REL0, or by using software correction.

15.6 Cyclic Interrupt Generation

The RTC module can generate an interrupt request whenever one of the timers overflows and is reloaded. This interrupt request may be used, for example, to provide a system time tick independent of the CPU frequency without loading the general purpose timers, or to wake-up regularly from sleep mode. The interrupt cycle time can be adjusted by choosing appropriate reload values and by enabling the appropriate interrupt request.

In this mode, the other operating modes can be combined. For example, a reload value of T14REL = $F9C0_H$ (2^{16} - 1600) generates a T14 interrupt request every 50 ms to wake-up the system regularly. Still the subsequent timers can be configured to represent the time or build a binary counter, however with a different time base.

15.7 RTC Interrupt Generation

The overflow signals of each timer of the RTC timer chain can generate an interrupt request. The RTC's interrupt subnode control register ISNC combines these requests to activate the common RTC interrupt request line RTC_IRQ.

Each timer overflow sets its associated request flag in register ISNC. Individual enable bits for each request flag determine whether this request also activates the common interrupt line. The enabled requests are ORed together on this line (see **Figure 15-5**).

The interrupt handler can determine the source of an interrupt request via the specific request flags and must clear them after appropriate processing (not cleared by hardware). The common node request bit is automatically cleared when the interrupt handler is vectored to.

Note: If only one source is enabled, no additional software check is required, of course. Both the individual request and the common interrupt node must be enabled.

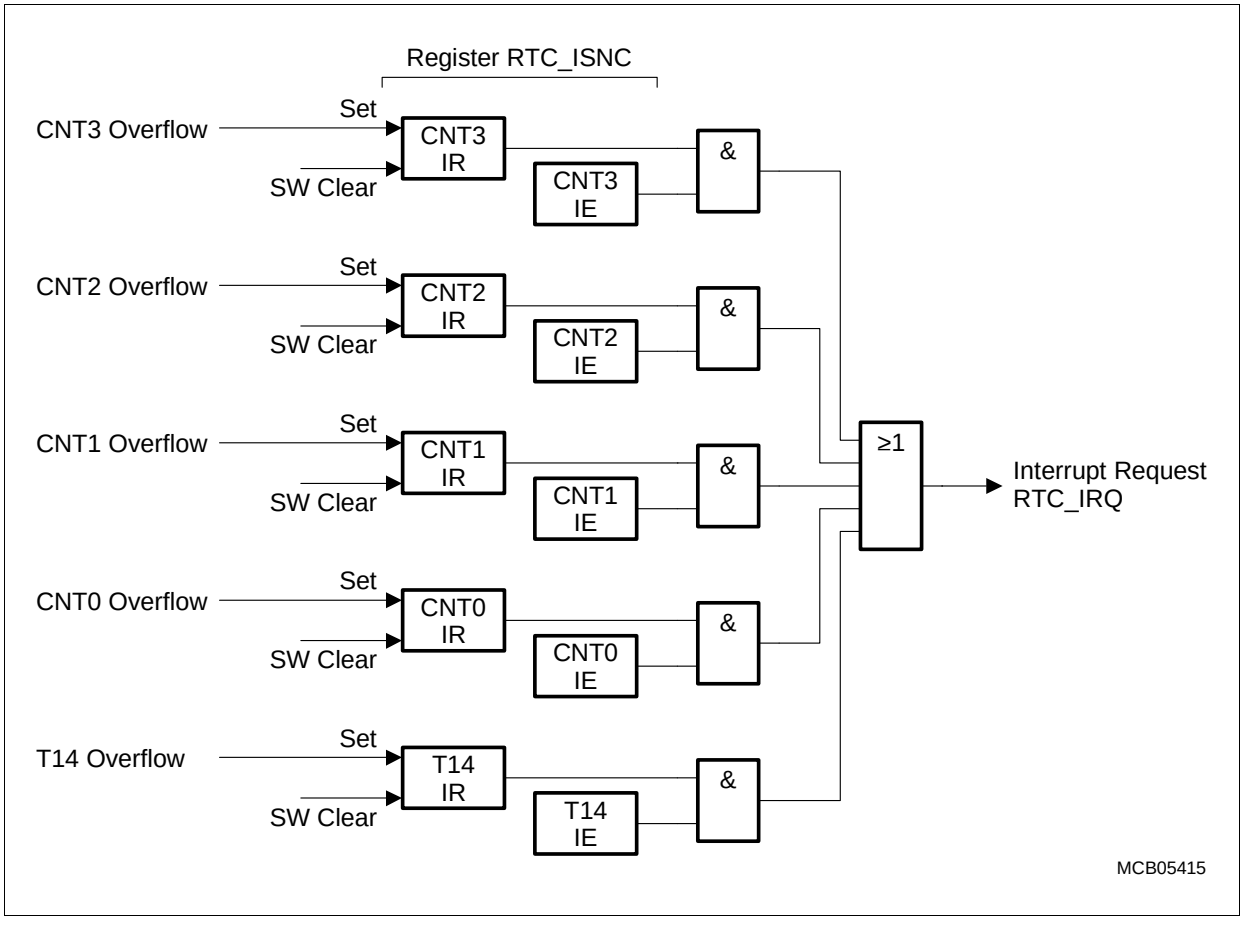


Figure 15-5 Interrupt Block Diagram

RTC_ISNC
Interrupt Subnode Ctrl. Reg. **ESFR ($F10C_H$/86_H)** **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	CNT 3IR	CNT 3IE	CNT 2IR	CNT 2IE	CNT 1IR	CNT 1IE	CNT 0IR	CNT 0IE	T14 IR	T14 IE
-	-	-	-	-	-	rwh	rw	rwh	rw	rwh	rw	rwh	rw	rwh	rw

Field	Bits	Type	Description
CNTxIR (x = 3 … 0)	9, 7, 5, 3	rwh	**Section CNTx Interrupt Request Flag** 0 No request pending 1 This source has raised an interrupt request
CNTxIE (x = 3 … 0)	8, 6, 4, 2	rw	**Section CNTx Interrupt Enable Control Bit** 0 Interrupt request is disabled 1 Interrupt request is enabled
T14IR	1	rwh	**T14 Overflow Interrupt Request Flag** 0 No request pending 1 This source has raised an interrupt request
T14IE	0	rw	**T14 Overflow Interrupt Enable Control Bit** 0 Interrupt request is disabled 1 Interrupt request is enabled

Note: The interrupt request flags in register ISNC must be cleared by software. They are not cleared automatically when the service routine is entered.

RTC_IC
RTC Interrupt Ctrl. Reg. **ESFR ($F19C_H$/CE_H)** **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	GPX	RTC IR	RTC IE	ILVL				GLVL	
-	-	-	-	-	-	-	rw	rwh	rw	rw				rw	

Note: Please refer to the general Interrupt Control Register description for an explanation of the control fields.
Register RTC_IC is not part of the RTC module and is reset with any system reset.

15.8 KSCCFG Register

RTC_KSCCFG
Kernel State Configuration Register

ESFR(F010$_H$) **Reset Value: 0001$_H$**

15	14	13 12	11	10	9 8	7	6	5 4	3 2	1	0
BP COM	**0**	**COMCFG**	**BP SUM**	**0**	**SUMCFG**	**BP NOM**	**0**	**NOMCFG**	**0**	**BP MOD EN**	**MOD EN**
w	r	rw	w	r	rw	w	r	rw	r	w	rw

Field	Bits	Type	Description
MODEN	0	rw	**Module Enable** This bit enables the module kernel clock and the module functionality. 0_B The module is switched off. It does not react on mode control actions and the module clock is switched off immediately (without stop condition). The module does not react on read accesses and ignores write accesses. 1_B The module is switched on. *Note: This bit is reset by an application reset.*
BPMODE	1	w	**Bit Protection for MODEN** This bit enables the write access to the bit MODEN. It always reads 0. It is only active during the write access cycle. 0_B MODEN is not changed. 1_B MODEN is updated with the written value. *Note: This bit is reset by an application reset.*
NOMCFG	[5:4]	rw	**Normal Operation Mode Configuration** This bit field defines the kernel mode applied in normal operation mode. $0X_B$ The module is switched on. $1X_B$ The module is switched off. This field is taken into account for CR = 00 or 11. *Note: This bit is reset by an application reset.*

Field	Bits	Type	Description
BPNOM	7	w	**Bit Protection for NOMCFG** This bit enables the write access to the bit field NOMCFG. It always reads 0. It is only active during the write access cycle. 0_B NOMCFG is not changed. 1_B NOMCFG is updated with the written value. *Note: This bit is reset by an application reset.*
SUMCFG	[9:8]	rw	**Suspend Mode Configuration** This bit field defines the kernel mode applied in suspend mode. $0X_B$ The module is switched on. $1X_B$ The module is switched off. This field is taken into account for CR = 01. *Note: This bit is reset by a debug reset.*
BPSUM	11	w	**Bit Protection for SUMCFG** This bit enables the write access to the bit field SUMCFG. It always reads 0. It is only active during the write access cycle. 0_B SUMCFG is not changed. 1_B SUMCFG is updated with the written value. *Note: This bit is reset by a debug reset.*
COMCFG	[13:12]	rw	**Clock Off Mode Configuration** This bit field defines the kernel mode applied in clock off mode. $0X_B$ The module is switched on. $1X_B$ The module is switched off. This field is taken into account for CR = 10. *Note: This bit is reset by an application reset.*
BPCOM	15	w	**Bit Protection for COMCFG** This bit enables the write access to the bit field COMCFG. It always reads 0. It is only active during the write access cycle. 0_B COMCFG is not changed. 1_B COMCFG is updated with the written value. *Note: This bit is reset by an application reset.*
0	[3:2], 6, 10, 14	r	**Reserved;** returns 0 if read; should be written with 0;

16 Analog to Digital Converter

The **A**nalog to **D**igital **C**onverter module (ADC) of the XC2000 allows the conversion of analog input values into discrete digital values based on the successive approximation method. With this method, the conversion result is elaborated bit by bit, starting with the most significant bit. As a consequence, an analog to digital conversion requires a certain number of clock cycles.

This chapter is structured as follows:

- Introduction (see **Section 16.1**)
- Operating the ADC (see **Section 16.2**)
- Module implementation in XC2000 (see **Section 16.3**)

16.1 Introduction

This section gives an overview about the feature set of the ADC module and introduces the general structure. It describes the:

- ADC block diagram with two kernels (see **Section 16.1.1**)
- Feature set description (see **Section 16.1.2**)
- Abbreviations (see **Section 16.1.3**)
- Kernel overview (see **Section 16.1.4**)
- Conversion request handling (see **Section 16.1.5**)
- Conversion result handling (see **Section 16.1.6**)
- Interrupt structure (see **Section 16.1.7**)
- Electrical models (see **Section 16.1.8**)
- Transfer characteristics and error definitions (see **Section 16.1.9**)

16.1.1 ADC Block Diagram

The ADC module contains 2 independent kernels (ADC0, ADC1) that can operate autonomously or can be synchronized to each other. An ADC kernel is a unit used to convert an analog input signal into a digital value and provides means for triggering conversions, data handling and storage.
With this structure, parallel conversion of up to two analog input channels is supported.

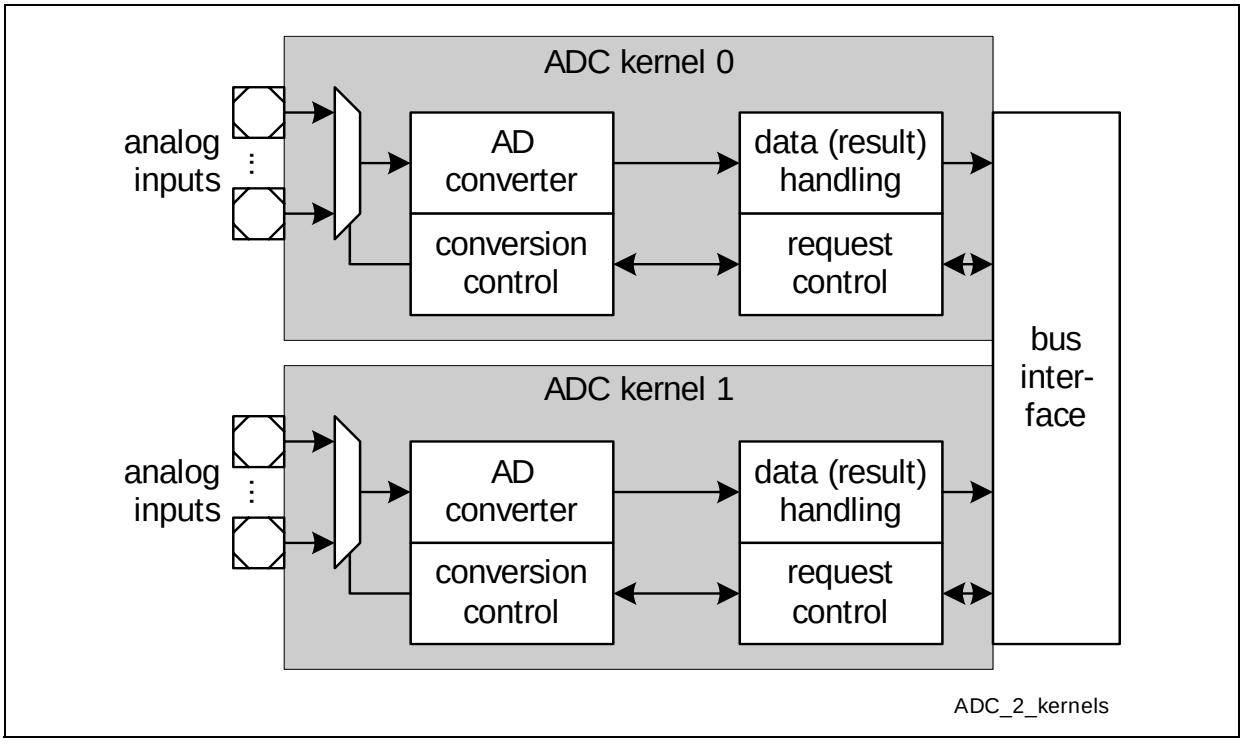


Figure 16-1 ADC Module Block Diagram

16.1.2 Feature Set

Features of each ADC kernel:

- Analog supply voltage range from 3.3 V (minimum) to 5 V (nominal) for V_{DDPA}
- Input voltage range from 0 V to analog supply voltage V_{DDPA}
- Input multiplexer for a maximum of 16 possible analog input channels
- Performance for 10-bit resolution (@f_{ADCI} = 16 MHz):
  - conversion time less than 1.5 µs, ±2 LSB_{10} TUE[1] @ supply voltage V_{DDPA} = 5 V
  - conversion time about 1.5 µs, **tbd** LSB_{10} TUE @ supply voltage V_{DDPA} = 3.3 V
- One standard reference input (VAREF) and one alternative reference input (CH0) available
- 3 conversion request sources for external or timer-driven events, auto-scan, programmable sequences, SW-driven conversions, etc.
- Synchronization of the ADC kernels for concurrent conversion starts and parallel sampling and measuring of analog input signals, e.g. for phase current measurements in AC drives
- Control capability for an external analog multiplexer, respecting the additional set up time
- Adjustable sampling times to accommodate output impedance of different analog signal sources (sensors, etc.)
- Possibility to cancel running conversions on demand with automatic restart
- Flexible interrupt generation (possibility of PEC support)
- Limit checking to reduce interrupt load (e.g. for temperature measurements or overload detection, only values exceeding a programmable level lead to an interrupt)
- Programmable data reduction filter, e.g. for digital anti-aliasing filtering, by adding a programmable number of conversion results
- Independent result registers (8 independent registers)
- Support of conversion result FIFO mechanism to allow a longer interrupt latency
- Support of suspend and power saving modes
- Individually programmable reference selection for each channel, e.g. to allow measurements of 3.3 V and 5 V signals in the full measurement range with the same ADC kernel

[1] This value reflects the ADC module capability in an adapted electrical environment, e.g. characterized by “clean” routing of analog and digital signals and separation of analog and digital PCB areas, low noise on analog power supply (< 30 mV), low switching activity of digital pins near to the ADC, etc.

16.1.3 Abbreviations

The following acronyms and terms are used in the ADC chapter:

Table 16-1 Abbreviations in ADC chapter

Abbreviation	Meaning
ADC	analog to digital converter
DNL	differential non-linearity error
FIFO	first-in-first-out data buffer mechanism
INL	integral non-linearity error
LSB_n	finest granularity of the analog value in digital format, represented by one least significant bit of the conversion result with n bits resolution (measurement range divided in 2^n equally distributed steps)
PEC	peripheral event controller (DMA-like mechanism of 16 bit controller)
SCU	system control unit of the device
TUE	total unadjusted error

16.1.4 ADC Kernel Overview

Each ADC kernel comprises:

- An **analog to digital converter** with a maximum of 16 analog inputs (CH0 - CH15). This block selects an input signal and translates the analog voltage into a digital value.
  Not all analog input channels are necessarily available in all packages, please refer to the implementation description in **Section 16.3**.
- A **conversion control** unit defining the conversion parameters like the length of the sample phase, the resolution and the reference for each conversion. The length of the sample phase and the resolution depend on the type of sensor (or other analog sources) connected to the ADC. These values are similar for several channels and, therefore, are grouped together to form the so-called input classes. Each channel can be individually assigned to an input class to define these parameters.
  The conversion control also handles the start conditions for the conversions, such as the immediate start (cancel-inject-repeat), overwrite of former results (wait-for-read), or synchronization of the ADC kernels (parallel conversions).
  Additionally, an external analog multiplexer can be controlled by the output signals EMUX[2:0] of each ADC kernel.
- A **request control** unit defining which analog input channel has to be converted next. It contains 3 request sources that can trigger conversions depending on different events, such as edges of PWM or timer signals or events at port pins. Each request source can trigger either 1, up to 4, or up to 16 conversions in a sequence.
- A **result handling** unit providing 8 result registers for the conversion results. The conversion result of each analog input channel can be directed to one of the result registers to be stored there. The result handling block also supports data reduction (e.g. for digital anti-aliasing filtering) by automatically adding up to 4 conversion results before informing the CPU that new data is available.
  Additionally, the results registers can be concatenated to FIFO structures to provide storage capability for more than one conversion result without overwriting previous data. This feature also helps to handle CPU latency effects.
- An **interrupt generation** unit issuing interrupt requests to the CPU depending on ADC events. The interrupt generation in the ADC kernels support different mechanisms, e.g. some interrupts can be coupled to a value range of the conversion result (limit checking), some interrupts can be used to transport conversion data to locations in memory for further treatment, and other interrupts are generated after a complete sequence of conversions.

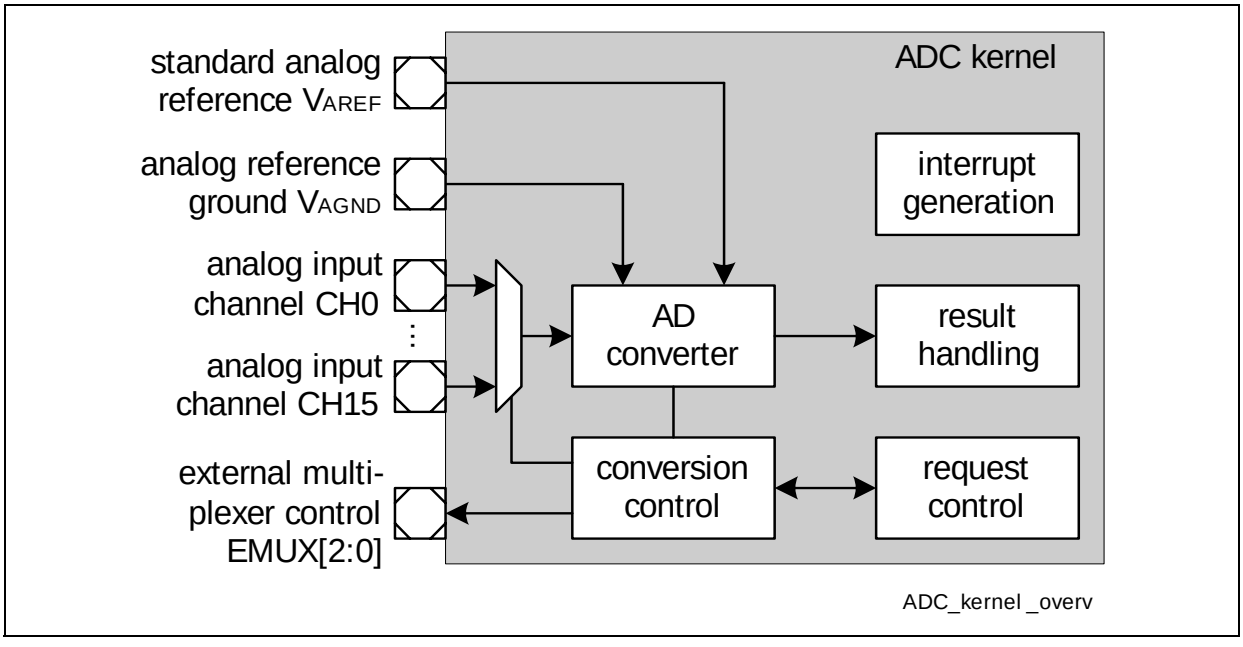


Figure 16-2 ADC Kernel Block Diagram

The number N_{CONV10} of f_{ADCI} clock cycles needed for a 10-bit conversion is given by the following equation:

N_{CONV10} = 17 (minimum sample time + conversion time) + additional sample time

An additional sample time can be programmed to adapt to application requirements (the minimum sample time of 2 clock cycles of f_{ADCA} part can be extended by a programmable value).

16.1.5 Conversion Request Unit

The conversion request unit of each ADC kernel autonomously handles the generation of conversion requests.
It contains three independent request sources that are connected to several modules to trigger the start of a conversion. A request source defines the analog input channel to be converted if a defined event occurs. For example, a trigger pulse from a timer unit generating a PWM signal can start the conversion of a single input channel or a programmed sequence of input channels.
Depending on the application, the request sources can be triggered by different events, either issued by other modules or under SW control. As a consequence, there can be two or more conversion requests pending at the same time. To allow the user to adapt the request source mechanism to the application needs, the trigger capability, the channel number(s) to be converted, and the priority can be individually programmed for each request source.
An arbiter block regularly scans the request sources for pending conversion requests and acts upon the conversion request with the highest priority. This conversion request is forwarded to the converter to start the conversion of the requested channel.

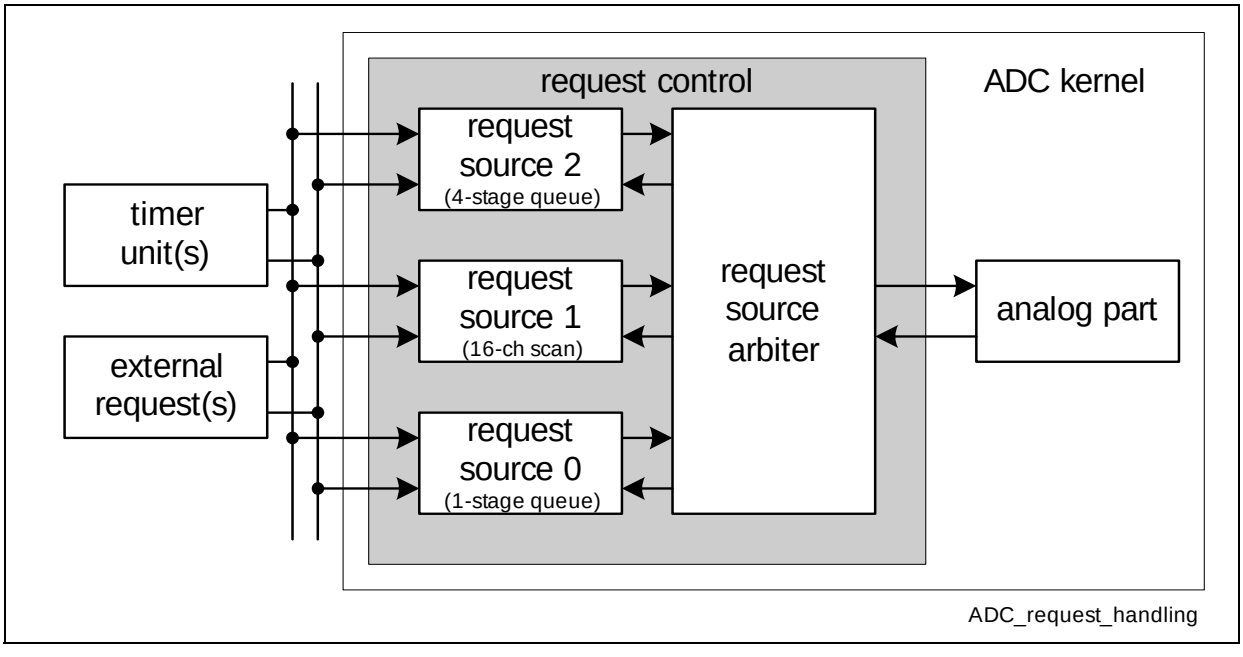


Figure 16-3 Conversion Request Unit

The functional characteristics of the request sources are adapted to the most common application requirements. In all request sources, a continuous operation or a single-shot operation can be selected. For continuous operation, the programmed sequence of conversions requests are continuously issued (once started), whereas in single-shot mode, each sequence of conversion requests has to be explicitly started. The trigger for a conversion request or a sequence can be handled under SW control or can be

synchronized to ADC-external events, such as timer signals or port pins. For each request source, the user can select an input signal (from 4 possible signals REQTRx[D:A]) as trigger input REQTRx and an input signal (from 4 possible signals REQGTx[D:A]) as gating input REQGTx.

- **Request source 0** (1-stage sequential source) can issue a conversion request for a single input channel. The channel number can directly be programmed.
  This mechanism could be used for SW-controlled conversion requests or HW-triggered conversions of a single input channel. If programmed with a high priority, it can interrupt the sequences of the other request sources to inject a single conversion.
- **Request source 1** (16-channel scan source) can issue conversion requests for a sequence of up to 16 input channels. It can be programmed which channel takes part in this sequence. The sequence always starts with the highest enabled channel number and continues towards lower channel numbers (order defined by the channel number, each channel can be converted only once per sequence).
  This mechanism could be used to scan input channels permanently or on a regular time base. For example, if programmed with a low priority, some input channels can be scanned in a background task to update information that is not time-critical.
- **Request source 2** (4-stage sequential source) can issue a conversion request for a sequence of up to 4 input channels. The channel numbers can be freely programmed, especially multiple conversions of the same channel within the sequence are supported.
  This mechanism could be used to support application-specific conversion sequences that can not be covered by the scanning mechanism of request source 1. Especially for timing-critical sequences containing multiple conversions of the same channel, request source 2 should be used. For example, if programmed with a medium priority, some input channels can be converted when a specified event occurs (e.g. synchronized to a PWM) while the scan of other input channels of the background task (handled by request source 1) is interrupted.

16.1.6 Conversion Result Unit

The conversion result unit comprises:

- A set of **8 result registers** for storing the conversion results. A pointer mechanism for each analog input channel distributes the conversion results to the result registers. Especially for auto-scan applications, this feature simplifies PEC use (only one PEC channel needed to transfer a complete auto-scan sequence into the device memory).
- The result registers provide **valid flags** to indicate if new data has been stored since it has been read out (new data indication).
- A **result FIFO mechanism** for conversion results handling with a “relaxed” CPU timing. Result registers not directly used as target for a conversion result can be concatenated to form a result FIFO. This structure allows to store a sequence of conversion results before the CPU has to interact.
- A **digital anti-aliasing or data reduction filter**, accumulating a programmable number of conversion results before generating a result event interrupt.
  This feature can be used to avoid CPU intervention on each conversion result if a certain number of conversion results are added before further treatment, especially for fast conversions sequences and averaging of results.
- A **wait-for-read mechanism** can be enabled independently for each result register to delay conversions targeting a result register that has not yet been read out.
- A **flexible interrupt generation** based on result register events. A result register event occurs if a new valid data word becomes available in a result register and can be read out. Especially when using data reduction or digital anti-aliasing filtering, the result register event indicates that the final result is available.
- Result register read view compatible to the ADC result register on XC16x devices available at one address and a new read view for data reduction capability at another address.
- **Debugger support** for ADC result registers supporting read out of ADC conversion results without changing the result status (new data indication).

16.1.7 Interrupt Structure

Each ADC kernel provides 4 independent service request output signals (SR[3:0]) used for interrupt handling (SRx signals connected to interrupt control registers). The interrupt generation inside the ADC kernel is based on three different types of events:

- **Channel events:**
  A channel event is detected if a conversion is finished and the conversion result is within a programmable value range.
  This type of event can be used to check if analog input values are inside or out of a nominal operating range, especially to reduce CPU load for background tasks. This allows the user to interrupt the CPU only if the specified conversion result range is met (or not met) instead of comparing each result by SW.
- **Result events:**
  A result event is detected if a new result is available in a result register and can be read out, e.g. to store the data in memory for further treatment by SW.
  This type of event can be used to trigger a read action by the CPU (or PEC). Especially when using data reduction or digital anti-aliasing filtering, not all finished conversion leads to a new result. Furthermore, when using a result FIFO, a result event decouples the CPU (PEC) read out from the channel events and tolerates a higher interrupt latency. The result register structure allows to use a single PEC channel for a complete auto-scan sequence by triggering the read out by a result event (if the conversion results of all channels taking part in the auto-scan sequence target the same result register, e.g. with FIFO mechanism or with a wait-for-read condition to avoid data loss).
- **Request source events:**
  A request source event is detected if a scan source has completely finished the requested conversion sequence. For a sequential source, the user can define where inside a conversion sequence a request source event is generated.
  This type of event can be used to inform the CPU that a conversion sequence has reached a defined state and SW can start the treatment of the related results in a block.

Each ADC event is indicated by a dedicated flag that can be cleared by SW. An interrupt can be generated (if enabled) for each event, independently from the status of the corresponding event indication flag. This structure ensures efficient PEC handling of ADC events (the ADC event can generate an interrupt without the need to clear the indication flag). A node pointer mechanism allows the user to group interrupts events by selecting which service request output signals SRx becomes activated by which event. Each ADC event can be individually directed to one of the service request output signals to adapt easily to application needs.

Note: A conversion can lead to three interrupts, one of each type. In this case, the ADC module first triggers the request source event interrupt, then the channel event interrupt, followed by the result event interrupt (all within a few f_{ADC} clock cycles).

16.1.8 Electrical Models

Each conversion of an analog input voltage into a digital value consists of two consecutive phases. During the sample phase, the input voltage is sampled and prepared for the following conversion phase. A simplified model for the sample phase describes the input signal path, whereas a second simplified model for the conversion phase is related to the reference voltage handling.

16.1.8.1 Input Signal Path

The ADC kernel in the XC2000 is based on one switched capacitor field for measurement with a total capacity represented by C_{AIN} and a small static capacitor at each input pin. During the sample phase, the capacitor field C_{AIN} is connected to one of the analog input CHx via an input multiplexer. The multiplexer is modeled by ideal switches and series resistors R_{AIN}. Only the switch to the selected analog input is closed during the sample phase. During the conversion phase or while no conversion is running (ADC is idle), all switches are open. The voltage at the analog input channel CHx is represented by V_{AINx}.

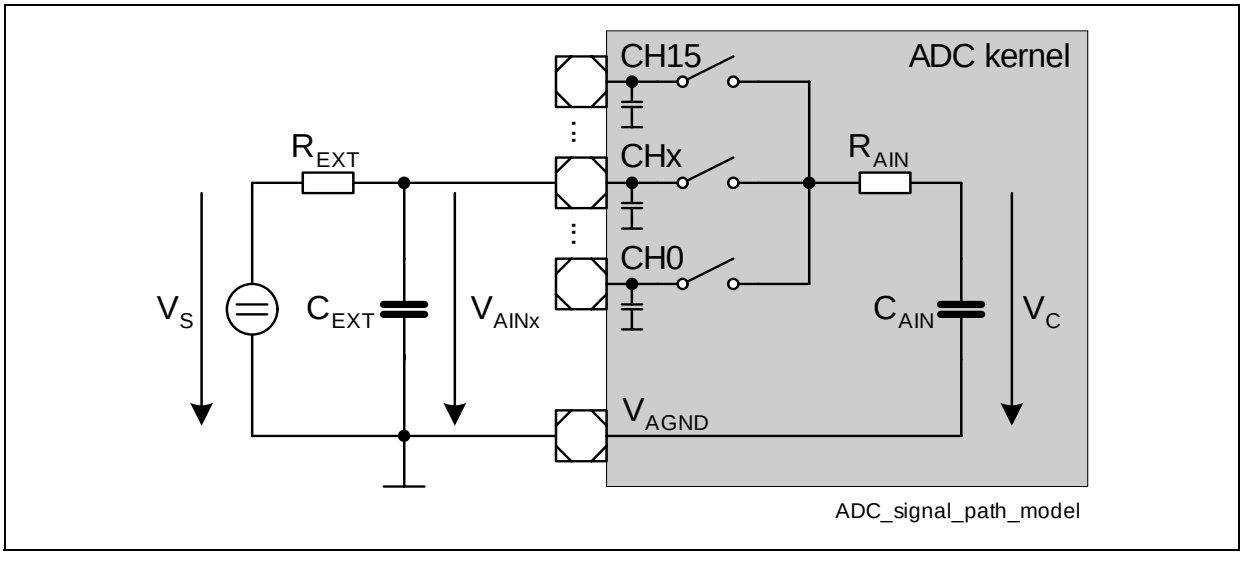


Figure 16-4 Signal Path Model

A simplified model for the analog input signal path is given in **Figure 16-4**. An analog voltage source (value V_S) with an internal impedance of R_{EXT} delivers the analog input that should be converted.

During the sample phase the corresponding switch is closed and the capacitor field C_{AIN} is charged. Due to the low-pass behavior of the resulting RC combination, the voltage V_C to be actually converted does not immediately follow V_S. The value R_{EXT} of the analog voltage source and the desired precision of the conversion strongly define the required length of the sample phase.

To reduce the influence of RR_{EXT} and to filter input noise, it is recommended to introduce

a fast external blocking capacitor C_{EXT} at the analog input pin of the ADC. Like this, mainly C_{EXT} delivers the charge during the sample phase. This structure allows a significantly shorter sample phase than without a blocking capacitor, because the low-pass time constant defining the sample time is mainly given by the values of R_{AIN} and C_{AIN}.
Additionally, the capacitor C_{AIN} is automatically precharged to a voltage of approximately the half of the standard reference voltage V_{AREF} to minimize the average difference between V_{AINx} and V_C at the beginning of a sample phase. Due to varying parameters and parasitic effects, the precharge voltage of C_{AIN} is typically smaller than V_{AREF} / 2.
On the other hand, the charge redistribution between C_{EXT} and C_{AIN} leads to a voltage change of V_{AINx} during the sample phase. In order to keep this voltage change lower than 1 LSB_n, it is recommended to use an external blocking capacitor C_{EXT} in the range of at least $2^n \times C_{AIN}$.
The resulting low-pass filter of R_{EXT} and C_{EXT} should be dimensioned in a way to allow V_{AINx} to follow V_S between two sample phases of the same analog input channel.
Please note that, especially at high temperatures, the analog input structure of an ADC can lead to a leakage current and introduces an error due to a voltage drop over R_{EXT}. The ADC input leakage current increases if the input voltage level is close to the analog supply ground V_{SS} or to the analog power supply V_{DDPA}. It is recommended to use an operating range for the input voltage between approximately 3% and 97% of V_{DDPA} to reduce input leakage values.
Furthermore, the leakage is influenced by an overload condition at adjacent analog inputs. During an overload condition, an input voltage exceeding the supply range is applied at an input and the built-in protection circuit limits the resulting input voltage. This leads to an overload current through the protection circuit that is translated (by a coupling factor) into an additional leakage at adjacent inputs.

16.1.8.2 Reference Path

During the conversion phase, parts of the capacitor field C_{AIN} are switched to a reference input or to VAGND. The ADC kernel supports two possible reference inputs, VAREF as standard reference and CH0 as alternative reference. The reference selection between both possibilities is handled individually for each analog input channel. For example, this structure allows conversions of 5 V and 3.3 V based analog input signals with the same ADC kernel.
A high accuracy of the conversion results requires a stable and noise-free reference voltage and analog supply voltages during the conversion phase. Instable voltages or noise on the supply or reference inputs lead to a reduced conversion accuracy. Please note that noise can also be introduced into the ADC module by other modules, e.g. by switching of neighboring pins. It is strongly recommended to carefully decouple analog from digital signal domains.
Due to the switching of parts of C_{AIN}, the ADC requires a dynamic current at the selected reference input. Thus, the impedance R_{AREF} of the reference voltage source V_R has to

be low enough to supply the reference current during the conversion phase. An external blocking capacitor C_{AREF} should be used to supply the peak currents and to minimize the current delivered by the reference source.
Due to the charge redistribution between C_{AREF} and parts of C_{AIN}, the voltage V_{AREF} decreases during the conversion phase. In order to limit the error introduced by this effect to 1/2 LSB_n, the external blocking capacitor C_{AREF} for the reference input should be at least $2^n \times C_{AIN}$.
The reference current I_{AREF} introduces a voltage drop at R_{AREF} that should not be neglected for the calculation of the overall accuracy. The average reference current during a conversion depends on the reference voltage level and the time t_{CONV} between two conversion starts.
$I_{AREF} = C_{AIN} \times V_{AREF} / t_{CONV}$

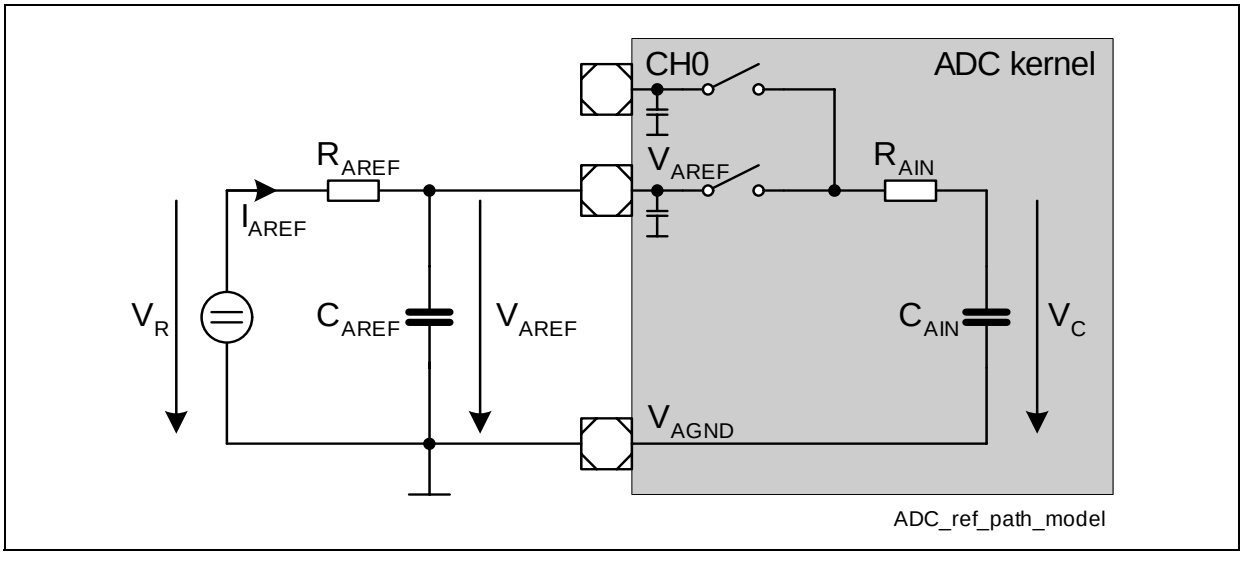


Figure 16-5 Reference Path Model

16.1.9 Transfer Characteristics and Error Definitions

The ideal transfer characteristic of the ADC translates a continuous analog input voltage into a discrete digital value out of a result range of 2^n steps for n bit resolution over a measurement range between 0 and a reference voltage. Each digital value in the available result range (from 0 to 2^n-1) represents an input voltage range that is defined by the reference voltage divided by 2^n. This range (called quantization step) represents the smallest granularity (called LSB_n) that can be handled by the ADC. Due to the discrete character of the digital result, each ADC conversion result has a system-inherent quantization uncertainty of ± 0.5 LSB_n. According to the ideal transfer characteristics, the first digital transition (between the digital values 0 and 1) takes place when the analog input reaches 0.5 LSB_n.
An analog input voltage above the reference voltage leads to a saturation of the digital result at 2^n-1.

Deviations of the conversion result from the ideal transfer characteristics can appear:

- An **offset error** is the deviation from the ideal transfer characteristics for an input voltage close to 0. It describes the difference between 0.5 LSB_n and the input voltage where the first digital transition (between the values of 0 and 1) occurs.
- A **gain error** is the deviation from the ideal transfer characteristics for an input voltage close to the reference voltage. It describes the difference between the reference voltage and the input voltage where the last digital transition (between the values of 2^n-2 and 2^n-1) occurs.
- A **differential non-linearity error** (DNL) describes the variations in the analog input voltage between two adjacent digital conversion results, over the full measurement range. If each step between the digital conversion results x and x+1 is exactly 1 LSB_n, the DNL value is zero. If the DNL value is lower than 1 LSB_n, the possibility of missing codes is excluded. A missing code occurs if not all values of the possible conversion result range can be reached.
- An **integral non-linearity error** (INL) describes the maximum difference between the transfer characteristics between the first and the last point of the measurement range and the real transfer characteristics (without quantization uncertainty, offset and gain errors).
- The **total unadjusted error** (TUE) describes the maximum deviation between a real conversion result and the ideal transfer characteristics over a given measurement range. Since some of these errors noted above can compensate each other, the TUE value generally is much less than the sum of the individual errors.
  The TUE also covers production process variations and internal noise effects (if switching noise is generated by the system, this generally leads to an increased TUE value).

16.2 Operating the ADC

This section describes the kernel functions and how to operate the kernel. It provides the functional description and the associated register descriptions.

- Register overview (see **Section 16.2.1**)

General module, kernel and arbiter operation:

- Enabling the ADC module for configuration of the behavior for the different device operating modes (see mode control description in **Section 16.2.2**).
- Enabling the converter for operation or selecting the desired power saving mode (see **Section 16.2.3**)
- Selecting the appropriate frequency for the converter and for the request source arbiter (see **Section 16.2.4**).
- General ADC registers (see **Section 16.2.5**)
- Configuring the request source arbiter (see **Section 16.2.6**)
- Arbiter registers (see **Section 16.2.7**)

Request source operation:

- Scan request source handling (see **Section 16.2.8**)
- Scan request source registers (see **Section 16.2.9**)
- Sequential request source handling (see **Section 16.2.10**)
- Sequential request source registers (see **Section 16.2.11**)

Channel and result register operation:

- Configuring the channel-related functions (see **Section 16.2.12**)
- Channel-related registers (see **Section 16.2.13**)
- Conversion result handling (see **Section 16.2.14**)
- Conversion request handling (see **Section 16.2.15**)

Additional features:

- External multiplexer control (see **Section 16.2.16**)
- Synchronization for parallel conversions (see **Section 16.2.17**)
- Additional feature registers (see **Section 16.2.18**)

16.2.1 Register Overview

Table 16-2 shows all registers required for programming the ADC module. It summarizes the ADC kernel registers and defines their relative addresses and the reset values. The relative addresses have to be added to the base addresses for the ADC kernels (see **Section 16.3.1**) to obtain the absolute address for each register.

The prefix "**ADCx**_" has to be added to the register names in this table for each ADC kernel to distinguish registers of the different kernels. In this naming convention, x indicates the kernel number.

All ADC registers (including KSCFG.NOMCFG and KSCFG.COMCFG) are reset by a class 3 reset, whereas bit field KSCFG.SUMCFG is reset by a class 1 reset.

Note: Register bits marked "w" always deliver 0 when read.

Table 16-2 ADC Module Register Summary

Short Name	Description	Rel. Addr. [1]	See Page
General Registers			
PISEL	Port Input Select Register	04_H	**Page 16-31**
KSCFG[2]	Kernel State Configuration Register	$0C_H$	**Page 16-24**
GLOBCTR	Global Control Register	10_H	**Page 16-26**
GLOBSTR	Global Status Register	12_H	**Page 16-28**
Arbiter Registers			
ASENR	Arbitration Slot Enable Register	18_H	**Page 16-37**
RSPR0	Request Source Priority Register 0	14_H	**Page 16-38**
Channel-Related Registers			
CHCTR0-15	Channel Control Register 0-15	20_H-$3E_H$	**Page 16-68**
INPCR0	Input Class Register 0	$C0_H$	**Page 16-70**

Table 16-2 ADC Module Register Summary (cont'd)

Short Name	Description	Rel. Addr. [1]	See Page
INPCR1	Input Class Register 1	$C2_H$	Page 16-70
LCBR0	Limit Checking Boundary Register 0	84_H	Page 16-71
LCBR1	Limit Checking Boundary Register 1	86_H	Page 16-71
LCBR2	Limit Checking Boundary Register 2	88_H	Page 16-71
LCBR3	Limit Checking Boundary Register 3	$8A_H$	Page 16-71
CHINFR	Channel Event Indication Flag Register	90_H	Page 16-72
CHINCR	Channel Event Indication Clear Register	92_H	Page 16-73
CHINPR0	Channel Interrupt Node Pointer Register 0	98_H	Page 16-74
CHINPR4	Channel Interrupt Node Pointer Register 4	$9A_H$	Page 16-74
CHINPR8	Channel Interrupt Node Pointer Register 8	$9C_H$	Page 16-75
CHINPR12	Channel Interrupt Node Pointer Register 12	$9E_H$	Page 16-76
ALR0	Alias Register 0	$1C_H$	Page 16-77
Result Registers			
RESR0-7	Result Register 0-7, normal view	40_H-$4E_H$	Page 16-86
RESRA0-7	Result Register 0-7, view A	50_H-$5E_H$	Page 16-87
RESRV0-7	Result Register 0-7, view V	60_H-$6E_H$	Page 16-86
RESRAV0-7	Result Register 0-7, view AV	70_H-$7E_H$	Page 16-87

Table 16-2 ADC Module Register Summary (cont'd)

Short Name	Description	Rel. Addr. [1]	See Page
VFR	Valid Flag Register	80_H	**Page 16-89**
RSSR	Result Status Shadow Register	82_H	**Page 16-88**
RCR0-7	Result Control Register 0-7	$B0_H$-BE_H	**Page 16-90**
EVINFR	Event Indication Flag Register	$A0_H$	**Page 16-92**
EVINCR	Event Indication Clear Register	$A2_H$	**Page 16-93**
EVINPR0	Event Interrupt Node Pointer Register 0	$A8_H$	**Page 16-94**
EVINPR8	Event Interrupt Node Pointer Register 8	AC_H	**Page 16-94**
EVINPR12	Event Interrupt Node Pointer Reg. 12	AE_H	**Page 16-95**
Request Source 0 Registers			
QMR0	Queue 0 Mode Register	$E0_H$	**Page 16-52**
QSR0	Queue 0 Status Register	$E2_H$	**Page 16-55**
Q0R0	Queue 0 Register 0	$E4_H$	**Page 16-57**
QBUR0	Queue 0 Backup Register	$E6_H$ shared	**Page 16-59**
QINR0	Queue 0 Input Register		**Page 16-61**
Request Source 1 Registers			
CRCR1	Conversion Request 1 Control Register	$E8_H$	**Page 16-43**
CRPR1	Conversion Request 1 Pending Register	EA_H	**Page 16-44**

Table 16-2 ADC Module Register Summary (cont'd)

Short Name	Description	Rel. Addr. [1]	See Page
CRMR1	Conversion Request 1 Mode Register	EC_H	**Page 16-45**
Request Source 2 Registers			
QMR2	Queue 2 Mode Register	$F0_H$	**Page 16-52**
QSR2	Queue 2 Status Register	$F2_H$	**Page 16-55**
Q0R2	Queue 2 Register 0	$F4_H$	**Page 16-57**
QBUR2	Queue 2 Backup Register	$F6_H$ shared	**Page 16-59**
QINR2	Queue 2 Input Register		**Page 16-61**
Additional Function Register			
EMCTR	External Multiplexer Control Register	$D0_H$	**Page 16-102**
EMENR	External Multiplexer Enable Register	$D6_H$	**Page 16-103**
SYNCTR	Synchronization Control Register	$1A_H$	**Page 16-104**

1) Short 8-bit addresses are not available for kernel registers of this module.

2) Register KSCFG is available only in the address range of ADC0, named ADC0_KSCFG.

Note: The addresses 00_H, 02_H, 06_H, 08_H, 16_H, $C4_H$, $C6_H$, $8C_H$, $8E_H$, $A4_H$, $A6_H$, and AA_H are reserved for future use.

16.2.2 Mode Control

The mode control concept for system control tasks, such as power saving, or suspend request for debugging, allows to program the module behavior under different device operating conditions. The behavior of the ADC kernels can be programmed for each of the device operating modes, that are requested by the global state control part of the SCU. It is advantageous that the ADC kernels of an ADC module show an identical behavior regarding the device operating modes (e.g. to avoid that a non-suspended kernel waits for a suspended kernel to start a synchronized conversion). Therefore, the ADC module has a common associated register **ADC0_KSCFG** defining the behavior of all kernels of the module in the following device operating modes:

- **Normal operation:**
  This operating mode is the default operating mode when neither a suspend request nor a clock-off request are pending. The module clock is not switched off and the ADC registers can be read or written. The kernel behavior is defined by KSCFG.NOMCFG.
- **Suspend mode:**
  This operating mode is requested when a suspend request (issued by a debugger) is pending in the device. The module clock is not switched off and the ADC registers can be read or written. The kernel behavior is defined by KSCFG.SUMCFG.
- **Clock-off mode:**
  This operating mode is requested for power saving purposes. The module clock is switched off automatically when all kernels of the ADC module reached their specified state in a stop mode. In this case, ADC registers can not be accessed. The kernel behavior is defined by KSCFG.COMCFG.

For the ADC module, the following internal actions can be influenced by mode control:

- A current conversion of an analog value:
  If the request control unit has found a pending conversion request, the conversion can be started. This start has to be enabled by the mode control. If the current kernel mode allows the conversion start (run modes 0 and 1), it will be executed. If the kernel mode does not allow a start (stop modes 0 and 1), the conversion is not started. The start request is not cancelled, but frozen. A “frozen” conversion is started as programmed if the kernel mode is changed to a run mode again.
- An arbiter round:
  The start of a new arbiter round has to be enabled by the kernel modes. In stop mode 1, a new arbiter round will not start.

The behavior of the ADC kernels can be programmed for each of the device operating modes (normal operation, suspend mode, clock-off mode). Therefore, the ADC kernels support four kernel modes, as shown in **Table 16-3**.

Table 16-3 ADC Kernel Behavior

Kernel Mode	Kernel Behavior	Code
run mode 0	kernel operation as specified, no impact on data transfer (same behavior for run mode 0 and run mode 1)	00_B
run mode 1		01_B
stop mode 0	A currently running AD conversion is completely finished and the result is treated. Pending conversion request to start a new conversion are not taken into account (but not deleted). They start conversions after entering a run mode as programmed. The arbiter continues as programmed.	10_B
stop mode 1	Like stop mode 0, but the arbiter is stopped after it has finished its arbitration round.	11_B

Generally, bit field KSCFG.NOMCFG should be configured for run mode 0 as default setting for standard operation. If the ADC kernels should not react to a suspend request (and to continue operation as in normal mode), bit field KSCFG.SUMCFG has to be configured with the same value as KSCFG.NOMCFG. If the ADC kernels should show a different behavior and stop operation when a specific stop condition is reached, the code for stop mode 0 or stop mode 1 has to be written to KSCFG.SUMCFG.
A similar mechanism applies for the clock-off mode with the possibility to program the desired behavior by bit field KSCFG.COMCFG.

Note: The stop mode selection strongly depends on the application needs and it is very unlikely that different stop modes are required in parallel in the same application. As a result, only one stop mode type (either 0 or 1) should be used in the bit fields in register KSCFG. Do not mix stop mode 0 and stop mode 1 and avoid transitions from stop mode 0 to stop mode 1 (or vice versa) for the ADC module.

If the module clock is disabled by KSCFG.MODEN = 0 or in clock-off mode when the stop condition is reached (in stop mode 0 or 1), the module can not be accessed by read or write operations (except register KSCFG that can always be accessed). As a consequence, it can not be configured.
Please note that bit KSCFG.MODEN should only be set by SW while all configuration fields are configured for run mode 0.

16.2.3 Module Activation and Power Saving Modes

The converter of the ADC supports specific power down modes allowing an automatic reduction of the power consumption between two conversions. The following modes are determined by bit field **GLOBSTR**.ANON:

- ANON = 00_B: **Converter switched off** (default after reset)
  The complete converter is switched off and held in its reset state, conversions are not possible. To start a conversion, ANON has to be programmed to the desired mode. A maximum wake-up time of about 10 µs has to be respected before starting a conversion. Furthermore, digital logic blocks are set to their initial state.
- ANON = 01_B: **Slow stand-by mode**
  The converter enters a power reduction mode after each conversion. It switches automatically to normal operation if a conversion is requested. A maximum wake-up time of about 10 µs has to be added to the sample time. This leads to the lowest power consumption for the ADC supply with wake-up capability.
- ANON = 10_B: **Fast stand-by mode**
  The converter enters a power reduction mode with less reduction than in slow stand-by mode after each conversion. It switches automatically to normal operation if a conversion is requested. A maximum wake-up time of about 3 µs has to be added to the sample time. This leads to a reduced power consumption for the ADC supply compared to normal operation.
- ANON = 11_B: **Normal operation**
  Conversions are always possible with the desired sample time. The converter stays active permanently.

16.2.4 Clocking Scheme

The different parts of an ADC kernel are driven by clock signals that are based on the clock f_{ADC} of the bus that is used to access the ADC module.

- The analog clock f_{ADCI} is used as internal clock for the converter and defines the conversion length and the sample time. It can be adjusted by programming bit field **GLOBCTR**.DIVA.
- The digital clock f_{ADCD} is used for the arbiter and defines the duration of an arbiter round. It can be adjusted by programming bit field **GLOBCTR**.DIVD.
- All other digital structures (such as interrupts, etc.) are directly driven by the module clock f_{ADC}.

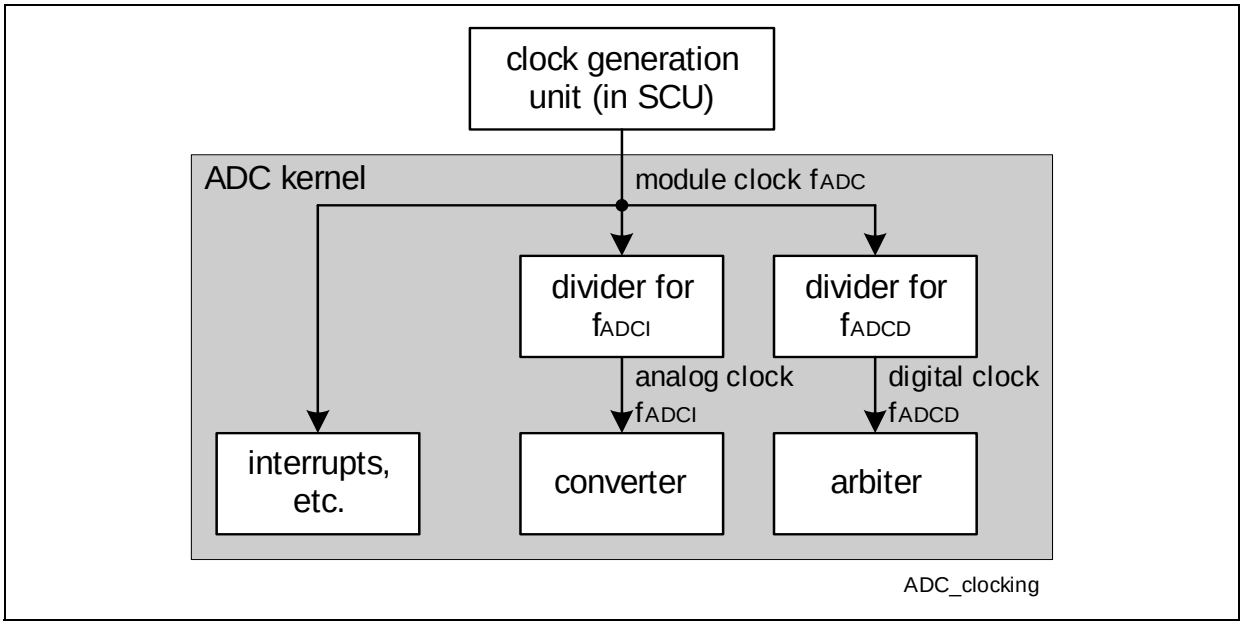


Figure 16-6 Clocking Scheme

Note: If the clock generation for the converter of the ADC falls below a minimum value or is stopped during a running conversion, the conversion result can be corrupted. For correct ADC results, the frequency of f_{ADCI} must not exceed the range indicated in the electrical characteristics chapter.

16.2.5 General ADC Registers

16.2.5.1 Kernel State Configuration Register

The kernel state configuration register KSCFG allows the selection of the desired kernel modes for the different device operating modes.
Bit fields KSCFG.NOMCFG and KSCFG.COMCFG are reset by a class 3 reset. Bit field KSCFG.SUMCFG is reset by a class 1 reset.
This register is a common register for all ADC kernels and can be accessed in the address range of ADC0.

*Note: The coding of the bit fields NOMCFG, SUMCFG and COMCFG is described in **Table 16-3**.*

ADC0_KSCFG
Kernel State Configuration Register

XSFR($0C_H$) **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BP COM	**0**	**COMCFG**		**BP SUM**	**0**	**SUMCFG**		**BP NOM**	**0**	**NOMCFG**		**0**		**BP MOD EN**	**MOD EN**
w	r	rw		w	r	rw		w	r	rw		r		w	rw

Field	Bits	Type	Description
MODEN	0	rw	**Module Enable** This bit enables the module kernel clock and the module functionality. 0_B The module is switched off immediately (without respecting a stop condition). It does not react on mode control actions and the module clock is switched off. The module does not react on read accesses and ignores write accesses (except to KSCFG). 1_B The module is switched on and can operate. After writing 1 to MODEN, it is recommended to read register KSCFG to avoid pipeline effects in the control block before accessing other ADC registers.

Field	Bits	Type	Description
BPMODEN	1	w	**Bit Protection for MODEN** This bit enables the write access to the bit MODEN. It always reads 0. 0_B MODEN is not changed. 1_B MODEN is updated with the written value.
NOMCFG	[5:4]	rw	**Normal Operation Mode Configuration** This bit field defines the kernel mode applied in normal operation mode. 00_B Run mode 0 is selected. 01_B Run mode 1 is selected. 10_B Stop mode 0 is selected. 11_B Stop mode 1 is selected.
BPNOM	7	w	**Bit Protection for NOMCFG** This bit enables the write access to the bit field NOMCFG. It always reads 0. 0_B NOMCFG is not changed. 1_B NOMCFG is updated with the written value.
SUMCFG	[9:8]	rw	**Suspend Mode Configuration** This bit field defines the kernel mode applied in suspend mode. Coding like NOMCFG.
BPSUM	11	w	**Bit Protection for SUMCFG** This bit enables the write access to the bit field SUMCFG. It always reads 0. 0_B SUMCFG is not changed. 1_B SUMCFG is updated with the written value.
COMCFG	[13:12]	rw	**Clock Off Mode Configuration** This bit field defines the kernel mode applied in clock-off mode. Coding like NOMCFG.
BPCOM	15	w	**Bit Protection for COMCFG** This bit enables the write access to the bit field COMCFG. It always reads 0. 0_B COMCFG is not changed. 1_B COMCFG is updated with the written value.
0	[3:2], 6, 10, 14	r	**Reserved** returns 0 if read; should be written with 0;

Note: The bit protection bits BPxxx allow partly modification of the configuration bits with a single write operation (without the need of a read-modify-write mechanism handled by the CPU).

16.2.5.2 Global Control Register

The global control register contains bits to control the timing of the arbiter and the general enable function for the converter.

GLOBCTR
Global Control Register **XSFR(10_H)** **Reset Value: $00FF_H$**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARBM	0			ARBRND		ANON		DIVD		DIVA					
rw	rw			rw		rw		rw		rw					

Field	Bits	Type	Description
DIVA	[5:0]	rw	**Divider Factor for Analog Internal Clock** This bit field defines the number of f_{ADC} clock cycles to generate the f_{ADCI} clock for the converter (used as internal base for the conversions and the sample time calculation). 00_H $f_{ADCI} = f_{ADC}$ 01_H $f_{ADCI} = f_{ADC} / 2$ 02_H $f_{ADCI} = f_{ADC} / 3$ ... $3F_H$ $f_{ADCI} = f_{ADC} / 64$
DIVD	[7:6]	rw	**Divider Factor for Digital Arbiter Clock** This bit field defines the number of f_{ADC} clock cycles within each arbitration slot (each arbitration slots last one periods of f_{ADCD}). It is recommended to use the default setting 00_B to obtain the minimum arbiter reaction time. 00_B $f_{ADCD} = f_{ADC}$ 01_B $f_{ADCD} = f_{ADC} / 2$ 10_B $f_{ADCD} = f_{ADC} / 3$ 11_B $f_{ADCD} = f_{ADC} / 4$
ANON	[9:8]	rw	**Analog Part Switched On** This bit field defines the setting of bit field **GLOBSTR**.ANON (bit description see there) if this kernel is the synchronization master or without synchronization feature. For a synchronization slave, this bit field is not taken into account.

Field	Bits	Type	Description
ARBRND	[11:10]	rw	**Arbitration Round Length** This bit field defines the number of arbitration slots per arbitration round (arbitration round length = t_{ARB}). 00_B An arbitration round contains 4 arbitration slots (t_{ARB} = 4 / f_{ADCD}). 01_B An arbitration round contains 8 arbitration slots (t_{ARB} = 8 / f_{ADCD}). 10_B An arbitration round contains 16 arbitration slots (t_{ARB} = 16 / f_{ADCD}). 11_B An arbitration round contains 20 arbitration slots (t_{ARB} = 20 / f_{ADCD}).
0	[14:12]	rw	**Reserved for Future Use** This bit field is reserved for future use and has to be written with 000_B.
ARBM	15	rw	**Arbitration Mode** This bit field defines whether the arbiter runs permanently or only while at least one conversion request is pending. 0_B The arbiter runs permanently. This setting has to be chosen in a synchronization slave (see **Section 16.2.17**). 1_B The arbiter only runs if at least one conversion request of an enabled request source is pending. This setting leads to a reproducible latency from an incoming request to the conversion start if the converter is idle. Synchronized conversions are not supported.

16.2.5.3 Global Status Register

The status control register contains bits indicating the current status of a conversion.

GLOBSTR
Global Status Register **XSFR(12_H)** **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		**CSRC**			**SYN RUN**	**ANON**		**CHNR**					**0**	**SAM PLE**	**BU SY**
r		rh			rh	rh		rh					r	rh	rh

Field	Bits	Type	Description
BUSY	0	rh	**Analog Part Busy** This bit indicates that a conversion is currently running. 0_B The converter is idle. 1_B A conversion is currently running.
SAMPLE	1	rh	**Sample Phase** This bit indicates that an analog input signal is currently sampled. 0_B The converter is not in the sampling phase. 1_B The converter is in the sampling phase.
CHNR	[7:3]	rh	**Channel Number** This bit field indicates which analog input channel is currently converted. This information is updated when a new conversion is started.

Field	Bits	Type	Description
ANON	[9:8]	rh	**Analog Part Switched On** This bit field defines the operation mode of the converter. It monitors either bit field GLOBCTR.ANON of the same ADC kernel (in master mode or without synchronization feature) or bit field GLOBCTR.ANON of the ADC kernel selected as synchronization master for this kernel (in slave mode). This ensures that all kernels of a synchronization group can be controlled with a single write operation to bit field GLOBCTR of the synchronization master. 00_B The converter is switched off and conversions are not possible. The arbiter finishes its current arbitration round (if running) and then remains in the idle state. 01_B The converter of the ADC module is switched on and conversions are possible. The automatic power-down capability of the converter is enabled, leading to the lowest power consumption of the ADC supply with wake-up capability, but a longer wake-up time before each conversion. 10_B The converter of the ADC module is switched on and conversions are possible. The automatic power-down capability of the converter is enabled, leading to a reduced power consumption of the ADC supply, but a shorter wake-up time before each conversion. 11_B The converter of the ADC module is switched on and conversions are possible. The automatic power-down capability of the converter is disabled leading to the nominal power consumption of the ADC supply.

Field	Bits	Type	Description
SYNRUN	10	rh	**Synchronous Conversion Running** This bit indicates that a synchronized (= parallel) conversion is currently running. 0_B There is no synchronized conversion running (either there is no conversion currently running or a synchronized conversion has not been requested). A running conversion can be cancelled and repeated in case of a new incoming conversion request with higher priority. 1_B A synchronized conversion is running. This conversion can not be cancelled while running. Higher priority requests can trigger conversions only after the end of a synchronized conversion.
CSRC	[13:11]	rh	**Currently Converted Request Source** This bit field indicates the arbitration slot number of the current conversion (if BUSY = 1, a conversion is still running) or of the last conversion (if BUSY = 0, no conversion is running). This bit field is updated with each conversion start. 000_B The channel requested by the request source of arbitration slot 0 is (has been) converted. 001_B The channel requested by the request source of arbitration slot 1 is (has been) converted. 010_B The channel requested by the request source of arbitration slot 2 is (has been) converted. other combinations are reserved
0	2, [15:14]	r	**Reserved** returns 0 if read; should be written with 0;

16.2.5.4 Input Select Registers

Register PISEL contains bit fields selecting the input signal for the trigger and gating inputs of the request sources. The connections depending on the device implementation, please refer to the implementation chapter for details.

Note: For signals connected to inputs with a synchronization stage, a synchronization delay of two f_{ADC} clock cycles has to be taken into account.

PISEL
Port Input Select Register **XSFR(04_H)** **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0			**0**	**REQTR2**		**REQGT2**		**REQTR1**		**REQGT1**		**REQTR0**		**REQGT0**	
r			rw	rw		rw		rw		rw		rw		rw	

Field	Bits	Type	Description
REQGT0	[1:0]	rw	**Input Select for REQGT0 of Source 0** This bit field defines the input signal used for the module input REQGT0 of the request source 0. 00_B The input signal REQGT0A is selected. 01_B The input signal REQGT0B is selected. 10_B The input signal REQGT0C is selected. 11_B The input signal REQGT0D is selected.
REQTR0	[3:2]	rw	**Input Select for REQTR0 of Source 0** This bit field defines the input signal used for the module input REQTR0 of the request source 0. 00_B The input signal REQTR0A is selected. 01_B The input signal REQTR0B is selected. 10_B The input signal REQTR0C is selected. 11_B The input signal REQTR0D is selected.
REQGT1	[5:4]	rw	**Input Select for REQGT1 of Source 1** This bit field defines the input signal used for the module input REQGT1 of the request source 1. 00_B The input signal REQGT1A is selected. 01_B The input signal REQGT1B is selected. 10_B The input signal REQGT1C is selected. 11_B The input signal REQGT1D is selected.

Field	Bits	Type	Description
REQTR1	[7:6]	rw	**Input Select for REQTR1 of Source 1** This bit field defines the input signal used for the module input REQTR1 of the request source 1. 00_B The input signal REQTR1A is selected. 01_B The input signal REQTR1B is selected. 10_B The input signal REQTR1C is selected. 11_B The input signal REQTR1D is selected.
REQGT2	[9:8]	rw	**Input Select for REQGT2 of Source 2** This bit field defines the input signal used for the module input REQGT2 of the request source 2. 00_B The input signal REQGT2A is selected. 01_B The input signal REQGT2B is selected. 10_B The input signal REQGT2C is selected. 11_B The input signal REQGT2D is selected.
REQTR2	[11:10]	rw	**Input Select for REQTR2 of Source 2** This bit field defines the input signal used for the module input REQTR2 of the request source 2. 00_B The input signal REQTR2A is selected. 01_B The input signal REQTR2B is selected. 10_B The input signal REQTR2C is selected. 11_B The input signal REQTR2D is selected.
0	12	rw	**Reserved for Future Extension** returns 0 if read; should be written with 0;
0	[15:13]	r	**Reserved** returns 0 if read; should be written with 0;

16.2.6 Request Source Arbiter

The request source arbiter evaluates which analog input channel has to be converted. Therefore, it regularly polls the request sources one after the other for pending conversion requests. The polling sequence is based on time slots with programmable length, called arbitration slots. If a request source is disabled or if no request source is available for an arbitration slot, the slot is considered as being empty and has no influence on the evaluation of the arbitration winner. After reset, all request sources are disabled and have to be enabled by bits in register **ASENR** to take part in the arbitration process.

The number of arbitration slots forming an arbitration round can be programmed to obtain a similar arbiter timing for different devices, even if the number of available request sources differs from one device to another. At the end of each arbitration round, the arbiter has determined the request source with the highest priority and a pending conversion request. This arbitration result is stored as arbitration winner for further actions. If a conversion is started in an arbitration round, this arbitration round does not deliver an arbitration winner.

In the XC2000, the following request sources are available:

- Request source 0 in arbitration slot 0: **1-stage sequential source**
  This request source can issue a conversion request for a single input channel.
- Request source 1 in arbitration slot 1: **16-channel scan source**
  This request source can issue a conversion request sequence of up to 16 input channels in a defined order.
- Request source 2 in arbitration slot 2: **4-stage sequential source**
  This request source can issue a conversion request sequence of up to 4 input channels in a freely programmable order.
- Last arbitration slot of the arbitration round: **Synchronization source**
  In this slot, the arbiter checks for a synchronized request from another ADC kernel and does not evaluate any internal request source. A request for a synchronized conversion is always handled with the highest priority in a synchronization slave kernel (pending requests from other sources are not considered).

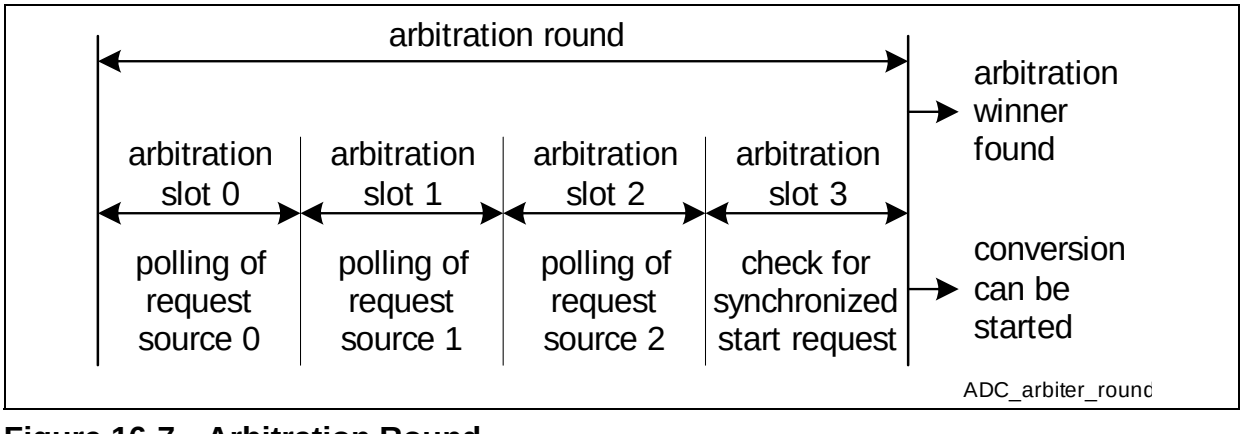


Figure 16-7 Arbitration Round

The period t_{ARB} of an arbitration round is given by:
$t_{ARB} = N \times (GLOBCTR.DIVD + 1) / f_{ADC}$
with N being 4, 8, 16, or 20 as defined by **GLOBCTR**.ARBRND

The period of the arbitration round introduces a timing granularity to detect an incoming conversion request signal and the earliest point to start the related conversion. This granularity can introduce a jitter of maximum one arbitration round. The jitter can be reduced by minimizing the period of an arbitration round (numbers of arbitration slots and their length).

To achieve a reproducible reaction time (constant delay without jitter) between the trigger event of a conversion request (e.g. by a timer unit or due to an external event) and the start of the related conversion, mainly the following two options exist. For both options, the converter has to be idle and other conversion requests must not be pending for at least one arbiter round before the trigger event occurs:

- If bit **GLOBCTR**.ARBM = 0, the **arbiter runs permanently**. In this mode, synchronized conversions of more than one ADC kernel are possible.
  The trigger for the conversion triggers has to be generated synchronously to the arbiter timing. Incoming triggers should have exactly n-times the granularity of the arbiter (n = 1, 2, 3, ...). In order to allow some flexibility, the duration of an arbitration slot can be programmed in cycles of f_{ADC}.
- If bit **GLOBCTR**.ARBM = 1, the **arbiter stops after an arbitration round** when no conversion request have been found pending any more. The arbiter is started again if at least one enabled request source indicates a pending conversion request. The trigger of a conversion request does need not to be synchronous to the arbiter timing. In this mode, parallel conversions are not possible for synchronization slave kernels.

16.2.6.1 Request Source Priority

Each request source has an individually programmable priority to be able to adapt to different applications (see register **RSPR0**). The priorities define the order the request sources are handled by the arbiter if two or more request sources indicate pending conversion requests at the same time.
Starting with request source 0, the arbiter checks if an enabled request source has a pending request for a conversion. The arbitration winner is the request source with a pending conversion request and the highest priority that has been found first in an arbitration round.

16.2.6.2 Conversion Start Modes

To start the requested conversion of the arbitration winner, the following aspects are automatically taken into consideration by the arbiter:

- If the converter is currently idle (no conversion running), the conversion of the arbitration winner is started immediately.

- If a conversion is currently running, the arbitration winner is compared to the priority of the currently running conversion. In the case that the current conversion has the same or a higher priority, it is completed. Then, the conversion of the arbitration winner is started.
- If a conversion is currently running, the arbitration winner is compared to the priority of the currently running conversion. In the case that the current conversion has the lower priority and the arbiter winner has been programmed for **wait-for-start mode**, the currently running conversion is completed. Then, the conversion of the arbitration winner is started.
  This mode can be used if the timing requirement for the higher priority conversions allow a jitter (between t3 and t4 in **Figure 16-8**) in the range of a running conversion.
- If a conversion is currently running, the arbitration winner is compared to the priority of the currently running conversion. In the case that the current conversion has the lower priority and the arbiter winner has been programmed for **cancel-inject-repeat mode**, the current conversion is aborted immediately if a new request with a higher priority has been found, unless both requests target the same result register with wait-for-read active (see **Section 16.2.14.2**). The conversion of the arbitration winner is started after the abort action. The aborted conversion request is restored in the request source that has requested the aborted conversion. As a consequence, it takes part again in the next arbitration round.
  Please note that the abort mechanism can take between 1 and 3 f_{ADCI} cycles, depending on the state of the current conversion.
  This mode can be used if higher priority conversions only tolerate a small jitter (between t8 and t9 in **Figure 16-8**).

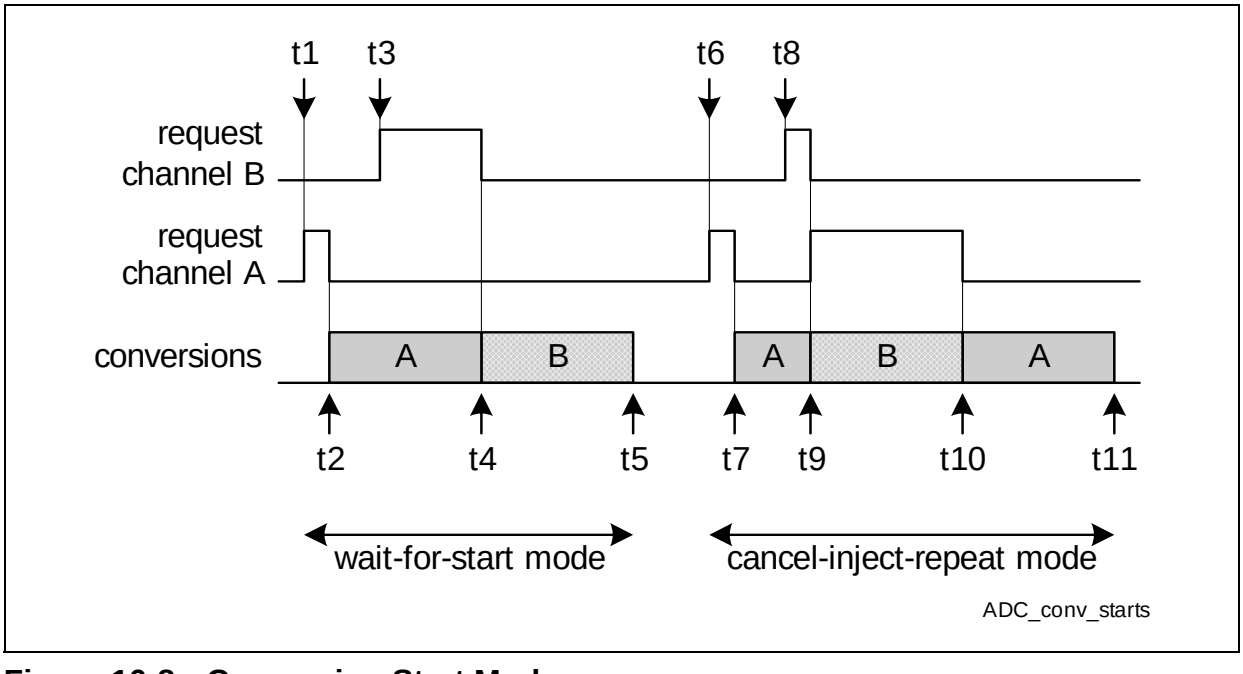


Figure 16-8 Conversion Start Modes

The conversion start mode can be individually programmed for each request source by bits in register **RSPR0** and is applied to all channels requested by the source. **Figure 16-8** shows the influence of both conversion start modes on the conversion sequence if two request sources generate conversion requests. In this example, channel A is issued by a request source with a lower priority than the request source requesting the conversion of channel B.

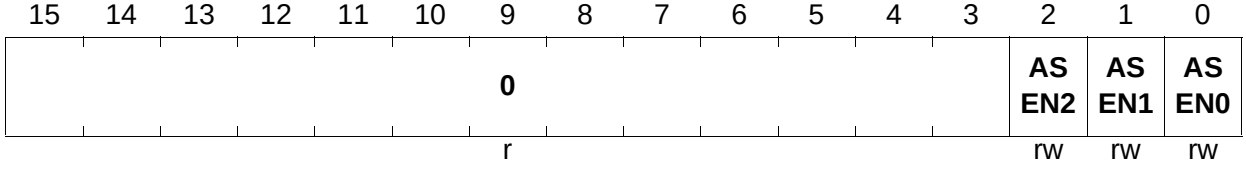
- t1: The trigger event for channel A occurs and a conversion request is activated.
- t2: At the end of the arbitration round, channel A is determined as arbitration winner, the conversion of channel A is started. With the start of the conversion, the related conversion request is cleared.
- t3: The trigger event for channel B occurs and a conversion request is activated. In wait-for-read mode, the currently running conversion of channel A is finished normally.
- t4: After the conversion of channel A is finished, the conversion of channel B is started. With the start of the conversion, the related conversion request is cleared.
- t5: The conversion of channel B is finished.
- t6: The trigger event for channel A occurs and a conversion request is activated.
- t7: At the end of the arbitration round, channel A is determined as arbitration winner, the conversion of channel A is started. With the start of the conversion, the related conversion request is cleared.
- t8: The trigger event for channel B occurs and a conversion request is activated.
- t9: At the end of the arbitration round, channel B is determined as arbitration winner. In cancel-inject-repeat mode, the currently running conversion of channel A is aborted and the conversion of channel B is started. With the abort of the conversion, the related conversion request is set again. With the start of the conversion, the related conversion request is cleared.
- t10: The conversion of channel B is finished. In the meantime, the pending request for channel A has been identified as arbitration winner and the conversion of channel A is started. With the start of the conversion, the related conversion request is cleared.
- t11: The conversion of channel A is finished.

16.2.7 Arbiter Registers

16.2.7.1 Arbitration Slot Enable Register

The arbitration slot enable register contains bits to enable/disable the conversion request treatment in the arbitration slots.

ASENR
Arbitration Slot Enable Register **XSFR(18_H)** **Reset Value: 0000_H**



Field	Bits	Type	Description
ASENx (x = 0-2)	x	rw	**Arbitration Slot x Enable** Each bit enables an arbitration slot of the arbiter round. ASEN0 enables the arbitration slot 0, ASEN1 the slot 1, etc. If an arbitration slot is disabled, the arbitration slot is considered as being empty. A pending conversion request of a request source connected to this slot is not taken into account for arbitration. The request source bits are not modified by write actions to ASENR. 0_B The corresponding arbitration slot is disabled. Conversions are not requested, even for the request source(s) with request bit(s) pending. To support ARBM = 1, the request source in arbitration slot x must not request any conversion while ASENx = 0. Conversion requests of the related request source should be cleared by SW before disabling a previously enabled arbitration slot. 1_B The corresponding arbitration slot is enabled. Conversions are requested for the request source(s) with pending request bit(s).
0	[15:3]	r	**Reserved** returns 0 if read; should be written with 0;

16.2.7.2 Request Source Priority Register

The request source priority register contains bits to define the request source priority and the conversion start mode. The priority and conversion start mode settings of an enabled request source must not be changed by SW. If a request source is disabled, the setting can be changed by SW if a currently running conversion requested by this source is finished.

RSPR0
Request Source Priority Register 0

XSFR(14_H) **Reset Value: 0000_H**

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

Field	Bits	Type	Description
PRIO0, PRIO1, PRIO2	[1:0], [5:4], [9:8]	rw	**Priority of Request Source x** This bit field defines the priority of the conversion request source x, located in arbitration slot x. 00_B Lowest priority is selected. ... 11_B Highest priority is selected.
CSM0, CSM1, CSM2	3, 7, 11	rw	**Conversion Start Mode of Request Source x** This bit defines the conversion start mode of the conversion request source x, located in arbitration slot x. 0_B Wait-for-start mode is selected. 1_B Cancel-inject-repeat mode is selected.
0	2, 6, 10, [15:12]	r	**Reserved** returns 0 if read; should be written with 0;

16.2.8 Scan Request Source Handling

A scan request source can issue conversion requests for a sequence of up to 16 input channels. It can be programmed individually for each input channel if it takes part in the scan sequence. The scan sequence always starts with the highest enabled channel number and continues towards lower channel numbers (order defined by the channel number, each channel can be converted only once per sequence).

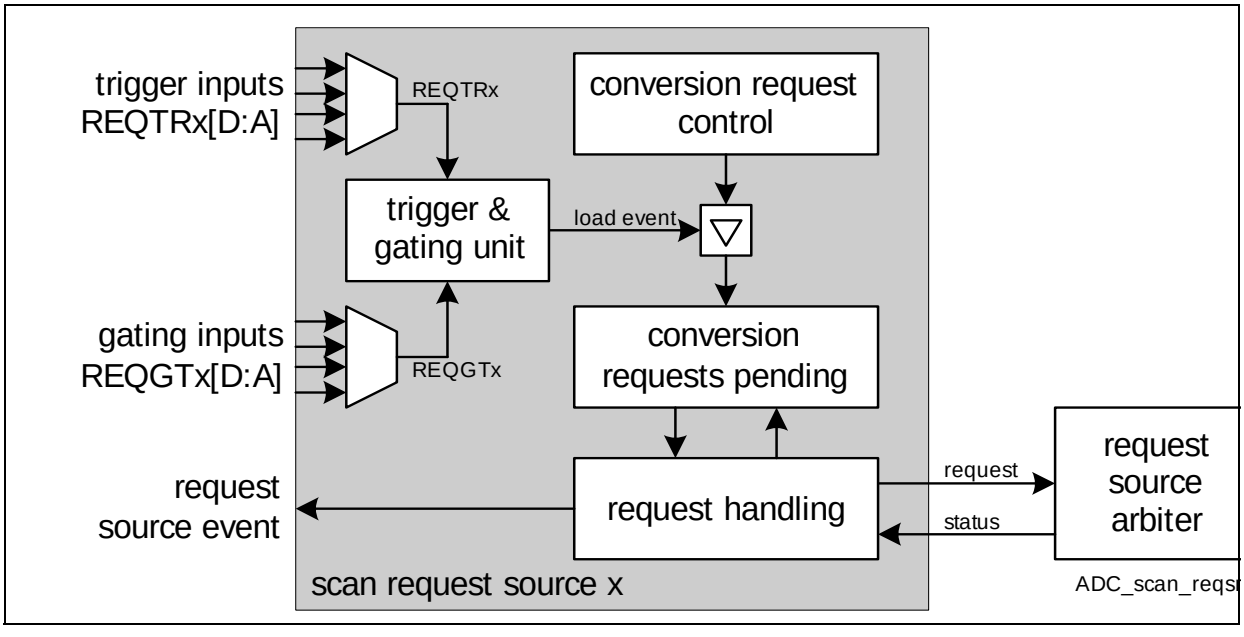


Figure 16-9 Scan Request Source

16.2.8.1 Overview

A scan request source performs the:

- Conversion request control:
  The conversion request control defines if an analog input channel takes part in the scan sequence (see bits in register **CRCR1**). The programmed register value is kept unchanged by an ongoing scan sequence.
- Conversion request pending:
  The pending conversion requests indicate if an input channel has to be converted in an ongoing scan sequence (see bits in register **CRPR1**). A conversion request can only be issued to the request source arbiter if at least one pending bit is set. With each conversion start that has been triggered by the scan request source, the corresponding pending bit is automatically cleared. The scan sequence is considered finished and a request source event is generated if the last conversion triggered by the scan source is finished and all pending bits have been cleared.
- Request handling:
  The request handling blocks interfaces with the request source arbiter. It requests conversion due to pending bits in the scan sequence and handles the conversion

status information. If a conversion triggered by the scan request source is aborted due to a conversion request from another request source with a higher priority, the corresponding pending bit is automatically set. This mechanism ensures that an aborted conversion takes part in the next arbitration round and does not get lost. The control of the scan sequence is done based on bits in register **CRMR1**.

- Trigger and gating signal handling:
  The trigger and gating unit interfaces with signals and modules outside the ADC module that can request conversions. For example, a timer unit can issue a request signal to synchronize conversions to PWM events. A load event starts a scan sequence by modifying the request pending bits according to the request control bits.

16.2.8.2 Scan Sequence Operation

To **operate a scan request source**, the following aspects should be taken into account:

- The bits in register CRCRx have to be programmed to define the channels participating in the scan sequence.
- If a trigger or gating function by external signals is desired, the gating and trigger inputs have to be defined by bit fields REQTRx and REQGTx in register **PISEL** (the value of x defines the number of the arbitration slot where the scan source is connected).
- The gating mechanism has to be defined by CRMRx.ENGT.
- The corresponding arbitration slot has to be enabled to accept conversion requests from the scan source (see register **ASENR**).
- The load event has to be defined by bits in CRMRx to start a scan sequence.
- If a load event occurs while CRMRx.LDM = 0, the content of CRCRx is copied to CRPRx (overwrite). This setting allows starting a new scan sequence and to “forget” remaining pending bits if a load event occurs while a scan sequence is running.
- If a load event occurs while CRMRx.LDM = 1, the content of CRCRx is bit-wisely logical OR-combined to CRPRx (no overwrite). This setting allows starting a new scan sequence without “forgetting” remaining pending bits if a load event occurs while a scan sequence is running.

To **start a scan sequence**, the following mechanisms are supported to generate a load event:

- An external trigger signal can be selected to start a scan sequence controlled by HW by an external module or signal, e.g. a timer unit or an input pin. The trigger feature is enabled by CRMRx.ENTR = 1. The load event is generated if a rising edge is detected at the selected trigger input.
- A load event is generated under SW control by writing CRMRx.LDEV = 1. This mechanism starts a scan sequence without modifying the bits in register CRCRx. A data write action to CRCRx does not lead to a load event (first prepare the channel control, then start the sequence).

- If SW writes data to register CRPRx, the written data is stored in register CRCRx and a load event is generated automatically. This mechanism starts a scan sequence with the channels defined by the written data (the sequence is defined and started with a single data write action, e.g. under PEC control).
- A load event is generated each time a scan sequence has finished and the request source event occurs if bit CRMRx.SCAN = 1. This setting leads to a permanent repetition of the scan sequence.

To **stop or abort an ongoing scan sequence**, the following mechanisms are supported:

- An external gating signal can be selected to stop and to continue a scan sequence at any point in time controlled by an external module or signal, e.g. a timer unit or an input pin. The gating feature can be enabled and the polarity of the gating signal REQGTx can be selected by CRMRx.ENGT. The gating mechanism does not modify the contents of the conversion pending bits, but only prevents the request handling block from issuing conversion requests to the arbiter.
- The arbiter can be disabled by SW for this arbiter slot by clearing the corresponding bit **ASENR**.ASENx. This mechanism does not modify the contents of the conversion pending bits, but only prevents the arbiter from accepting requests from the request handling block.
- The pending request bits can be cleared by writing bit CRMRx.CLRPND = 1. It is recommended to stop the scan sequence before clearing the pending bits.

16.2.8.3 Request Source Event and Interrupt

A request source event of a scan source occurs if the last conversion of a scan sequence is finished (all pending bits = 0). A request source event interrupt can be generated based on a request source event according to the structure shown in **Figure 16-10**. If a request source event is detected, it sets the corresponding indication flag in register **EVINFR**. These flags can also be set by writing a 1 to the corresponding bit position, whereas writing 0 has no effect. The indication flags can be cleared by SW by writing a 1 to the corresponding bit position in register **EVINCR**.

The service request output line SRx that is selected by the request source event interrupt node pointer bit fields in register **EVINPR0** becomes activated each time the related request source event is detected or the related bit position in register **EVINFR** is written with a 1.

The request source events and the result events share the same registers. The request source event is located at the bit position in register **EVINFR**:

- Event 1: Request source event of scan source in arbitration slot 1.

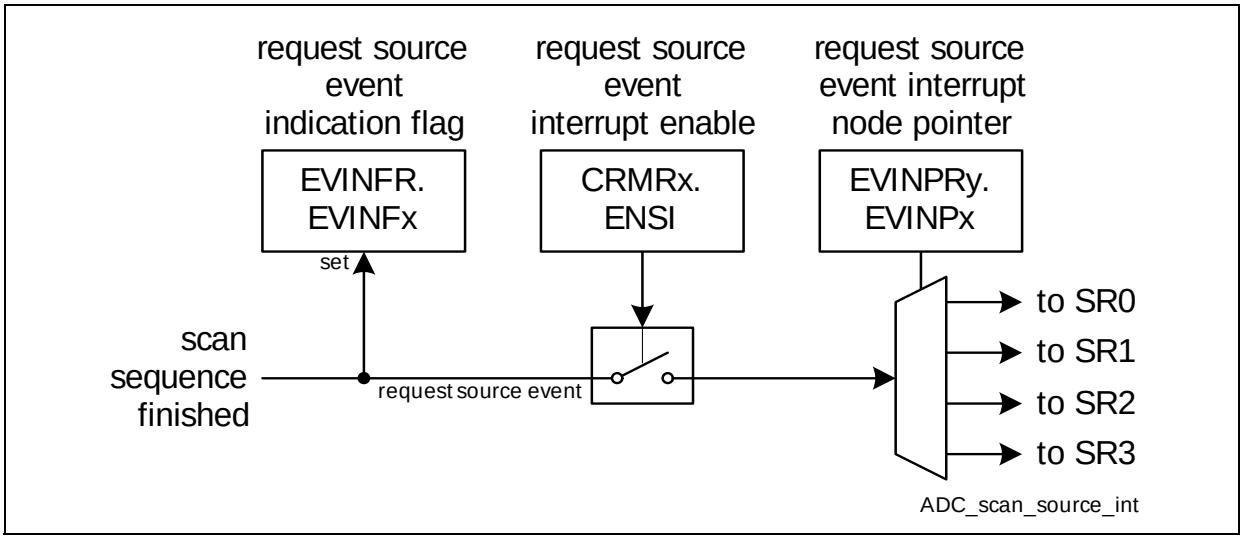


Figure 16-10 Interrupt Generation of a Scan Request Source

16.2.9 Scan Request Source Registers

16.2.9.1 Conversion Request Control Register

These registers contain the control and status bits of the scan request source(s). The index 1 describes the number of the arbitration slot where the request source is taking part in the arbitration.

The conversion request control register contains the bits that are copied to the pending register when the load event occurs. This register can be accessed at two different addresses. One address for read and write access is given for CRCRx (leading to the attribute “rw”). The second address only used for write actions is given for CRPRx (leading to the additional attribute “h”).

CRCR1
Conversion Request 1 Control Register

XSFR($E8_H$) **Reset Value: 0000_H**

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

Field	Bits	Type	Description
CHx (x = 0-15)	x	rwh	**Channel Bit x** Each bit corresponds to one analog input channel, the channel number CHx is defined by the bit position x in this register. 0_B The analog channel CHx will not be requested for conversion by this scan request source. 1_B The analog channel CHx will be requested for conversion by this scan request source.

16.2.9.2 Conversion Request Pending Register

The conversion request pending register contains the bits that are requesting a conversion of the corresponding analog channel.
A write operation to CRPRx leads to a data write to the bits in CRCRx with an automatic load event generation (leading to the attribute “w”). A read operation to CRPRx delivers the pending bits (leading to the attribute “rh”).

CRPR1
Conversion Request 1 Pending Register

XSFR(EA_H) **Reset Value: 0000_H**

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

Field	Bits	Type	Description
CHPx (x = 0-15)	x	rwh	**Channel Pending Bit x** Write view: A write to this address targets the bits in register CRCR1. Read view: Each bit corresponds to one analog channel, the channel number CHx is defined by the bit position in the register. 0_B The analog channel CHx is not requested for conversion by this request source. 1_B The analog channel CHx is requested for conversion by this request source.

16.2.9.3 Conversion Request Mode Register

The conversion request mode register contains bits to configure the desired operating mode of the scan request source.

CRMR1
Conversion Request 1 Mode Register

XSFR(EC_H) **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0						LDEV	CLRPND	REQGT	0	LDM	SCAN	ENSI	ENTR	ENGT	
r						w	w	rh	r	rw	rw	rw	rw	rw	

Field	Bits	Type	Description
ENGT	[1:0]	rw	**Enable Gate** This bit field enables the gating functionality for the request source. 00_B The request source does not issue conversion requests. 01_B The request source issues conversion requests if at least one pending bit is set. 10_B The request source issues conversion requests if at least one pending bit is set and the selected gating signal REQGTx = 1. 11_B The request source issues conversion requests if at least one pending bit is set and the selected gating signal REQGTx = 0.
ENTR	2	rw	**Enable External Trigger** This bit enables the external trigger possibility. If enabled, the load event takes place if a rising edge is detected at the selected trigger input signal REQTR. 0_B The external trigger is disabled. 1_B The external trigger is enabled.
ENSI	3	rw	**Enable Source Interrupt** This bit enables the request source interrupt generation if a request source event occurs (last pending conversion is finished). 0_B The request source interrupt is disabled. 1_B The request source interrupt is enabled.

Field	Bits	Type	Description
SCAN	4	rw	**Autoscan Enable** This bit enables a permanent scan functionality. If enabled, the load event is automatically generated if a request source event occurs. 0_B The permanent scan functionality is disabled. 1_B The permanent scan functionality is enabled.
LDM	5	rw	**Load Event Mode** This bit defines the transfer mechanism triggered by the load event. 0_B With the load event, the value of register CRCRx is copied to the pending register CRPRx (overwrite). 1_B With the load event, the value of register CRCRx is bit-wisely logical OR combined to the pending register CRPRx.
REQGT	7	rh	**Request Gate Level** This bit monitors the level at the REQGT input. 0_B The level is 0. 1_B The level is 1.
CLRPND	8	w	**Clear Pending Bits** 0_B No action. 1_B The bits in register CRPRx are cleared.
LDEV	9	w	**Generate Load Event** 0_B No action. 1_B A load event is generated.
0	6, [15:10]	r	**Reserved** returns 0 if read; should be written with 0;

16.2.10 Sequential Request Source Handling

Sequential request sources have been introduced to allow short conversion sequences with freely programmable channel numbers (contrary to a scan request source with a fixed conversion order for the enabled channels). Two versions of the sequential sources are available in each ADC kernel:

- Request source in arbitration slot 2:
  This request source can handle a sequence of up to 4 input channels (4-stage queue for 4 entries). This mechanism could be used to support application-specific conversion sequences, especially for timing-critical sequences containing multiple conversions of the same channel.
- Request source in arbitration slot 0:
  This request source can handle a single input channel (1-stage queue for 1 entry). This mechanism could be used for SW-controlled conversion requests or HW-triggered conversions of a single input channel (to “inject” a single conversion into a running sequence).

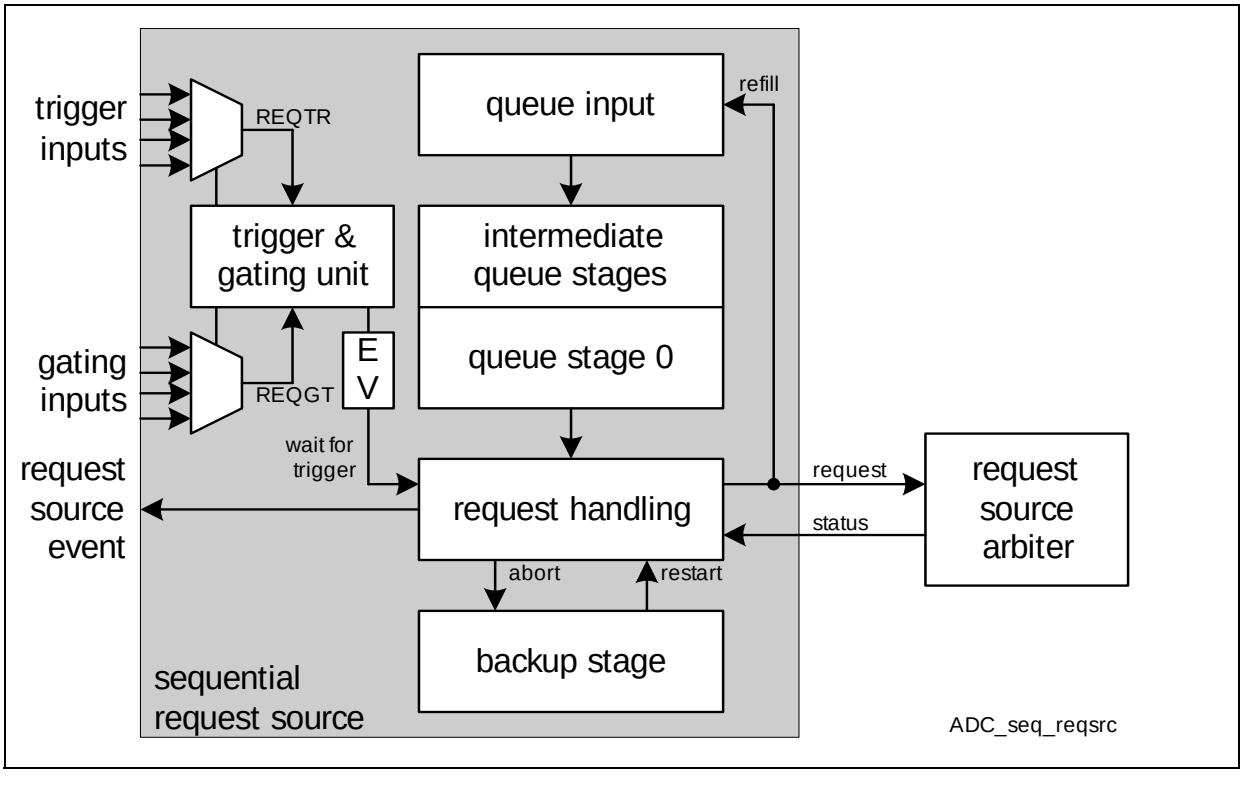


Figure 16-11 Sequential Request Source

The internal structure and the handling of the sequential sources is similar for both versions. The programmed sequence is stored in a queue buffer (based on a FIFO mechanism) with at least one queue stage (stage 0) and a backup stage for aborted conversions. The only difference between both versions is given by the number of intermediate queue stages for storing the sequence. The request source in arbitration

slot 0 does not provide intermediate queue stages (1-stage queue with only queue stage 0), whereas the one in arbitration slot 2 provides 3 intermediate queue stages in additional to queue stage 0 (leading to a 4-stage queue).

16.2.10.1 Overview

A sequential request source performs the:

- Queue input:
  The queue input represents the programming interface where the sequence is defined (see **QINR0**, **QINR2**). It does not provide any buffer capability, but handles the filling of the queue buffer (queue stage 0 plus optional intermediate queue stages) by writing data to it. The contents of the queue stages can not be directly modified by program, except by the command for flushing the complete queue.
  The queue input also handles the refill mechanism, an automatic re-insertion of a started conversion from queue stage 0 (including the control parameters) as new queue input. This feature allows a single setup (by SW) of a conversion sequence and multiple repetitions of the same sequence without the need to re-program it each time. A conversion sequence is repeated automatically if all queue entries of the sequence are setup for refill mode.
- Queue stage 0:
  The contents of this queue stage defines which channel will be requested next for a conversion (see **Q0R0**, **Q0R2**). It also defines if the request should be triggered by an external event or if the requested conversion should follow the previous one as soon as possible. It also enables the request source interrupt generation after the conversion.
  The contents of this queue stage is cleared when the requested conversion is started and the next queue entry can be handled (if available).
- Queue backup stage:
  The queue backup stage is used to store the request control parameters when a conversion requested by this request source is aborted. A validation bit indicates that the aborted conversion has to be requested next (before the current contents of queue stage 0) to maintain the original sequence (see **QBUR0**, **QBUR2**).
- Request handling:
  The request handling block interfaces with the request source arbiter. It requests a conversion due to a valid information in queue stage 0 and handles the conversion status information. The control of the queue sequence is done based on bits in registers **QMR0** (for the request source in arbitration slot 0) and **QMR2** (for the arbitration slot 2).
- Trigger and gating signal handling:
  The trigger and gating unit interfaces with signals and modules outside the ADC module that can request conversions. For example, a timer unit can issue a request signal to synchronize conversions to PWM events. A trigger event can start a conversion request for the entry in queue stage 0 (see **QMR0**, **QMR2**). An event flag

QSRx.EV indicates that a trigger event has been detected (rising edge of selected trigger input signal REQTRx if enabled by QMRx.ENTR or write action with QMRx.TREV = 1). This bit is cleared with each conversion start requested by this source or by writing bits CEV = 1, FLUSH = 1, or CLRV = 1.

16.2.10.2 Sequential Source Operation

To **operate a sequential request source**, the following aspects should be taken into account:

- The sequence has to be initialized by writing to the queue input **QINR0** (for arbitration slot 0) or **QINR2** (for arbitration slot 2) when using the refill mechanism. Each write access corresponds to one conversion request.
  The desired sequence should be completely initialized before enabling the request source, because with enabled refill feature, write accesses by SW to QINRx are not allowed.
- If a trigger or gating function by external signals is desired, the gating and trigger inputs have to be defined by bit fields REQTRx and REQGTx in register **PISEL** (the value of x defines the number of the arbitration slot where the request source is connected).
- The gating mechanism has to be defined by QMRx.ENGT.
- If an external trigger mechanism is desired, it has to be enabled by QMRx.ENTR = 1.
- The corresponding arbitration slot has to be enabled to accept conversion requests from the sequential source (see register **ASENR**).

To **start a sequence** of a sequential request source, the following mechanisms are supported:

- An external trigger signal can be selected to start a scan sequence controlled by HW by an external module or signal, e.g. a timer unit or an input pin. The trigger feature is enabled by QMRx.ENTR = 1. The trigger event is generated if a rising edge is detected at the selected trigger input.
- A trigger event is generated under SW control by writing QMRx.TREV = 1. This mechanism starts a request if queue stage 0 contains valid data (or the queue backup stage respectively).
- A write operation to a queue input leads to a (new) valid queue entry. If the queue is empty (no valid entry), the written data arrives in queue stage 0 and starts a conversion request (if enabled by QMRx.ENGT and without waiting for an external trigger). If the refill mechanism is used, the queue inputs must not be written while the queue is running. Write operations to a completely filled queue are ignored.

To **stop or abort an ongoing sequence** of a sequential request source, the following mechanisms are supported:

- An external gating signal can be selected to stop and to continue a sequence at any point in time controlled by an external module or signal, e.g. a timer unit or an input pin. The gating feature can be enabled and the polarity of the gating signal can be

selected by QMRx.ENGT. The gating mechanism does not modify the queue entries, but only prevents the request handling block from issuing conversion requests to the arbiter.
- The arbiter can be disabled by SW for this arbiter slot by clearing the corresponding bit **ASENR**.ASENx. This mechanism does not modify the queue entries, but only prevents the arbiter from accepting requests from the request handling block.
- The next pending queue entry is cleared by writing bit QMRx.CLRV = 1. It is recommended to stop the sequence before clearing a queue entry (ENGT = 00_B). If the queue backup stage contains a valid entry, this one is cleared, otherwise a valid entry in queue register 0 is cleared.
- All queue entries are cleared by writing bit QMRx.FLUSH = 1. It is recommended to stop the sequence before clearing queue entries.

16.2.10.3 Request Source Event and Interrupt

A request source event occurs when a conversion that has been requested by this source is completely finished. The interrupt enable bits are located in the queue 0 register (if this has not been a repeated start after an abort) or in the queue backup register (if this has been a repeated start after an abort).
A request source event interrupt can be generated based on a request source event according to the structure shown in **Figure 16-12**. If a request source event is detected, it sets the corresponding indication flag in register **EVINFR**. These flags can also be set by writing a 1 to the corresponding bit position, whereas writing 0 has no effect. The indication flags can be cleared by SW by writing a 1 to the corresponding bit position in register **EVINCR**.

The service request output line SRx that is selected by the request source event interrupt node pointer bit fields in register **EVINPR0** becomes activated each time the related request source event is detected or the related bit position in register **EVINFR** is written with a 1. The request source events and the result events share the same registers. The request source event is located at the bit position in register **EVINFR**:

- Event 0: Request source event of sequential source in arbitration slot 0.
- Event 2: Request source event of sequential source in arbitration slot 2.

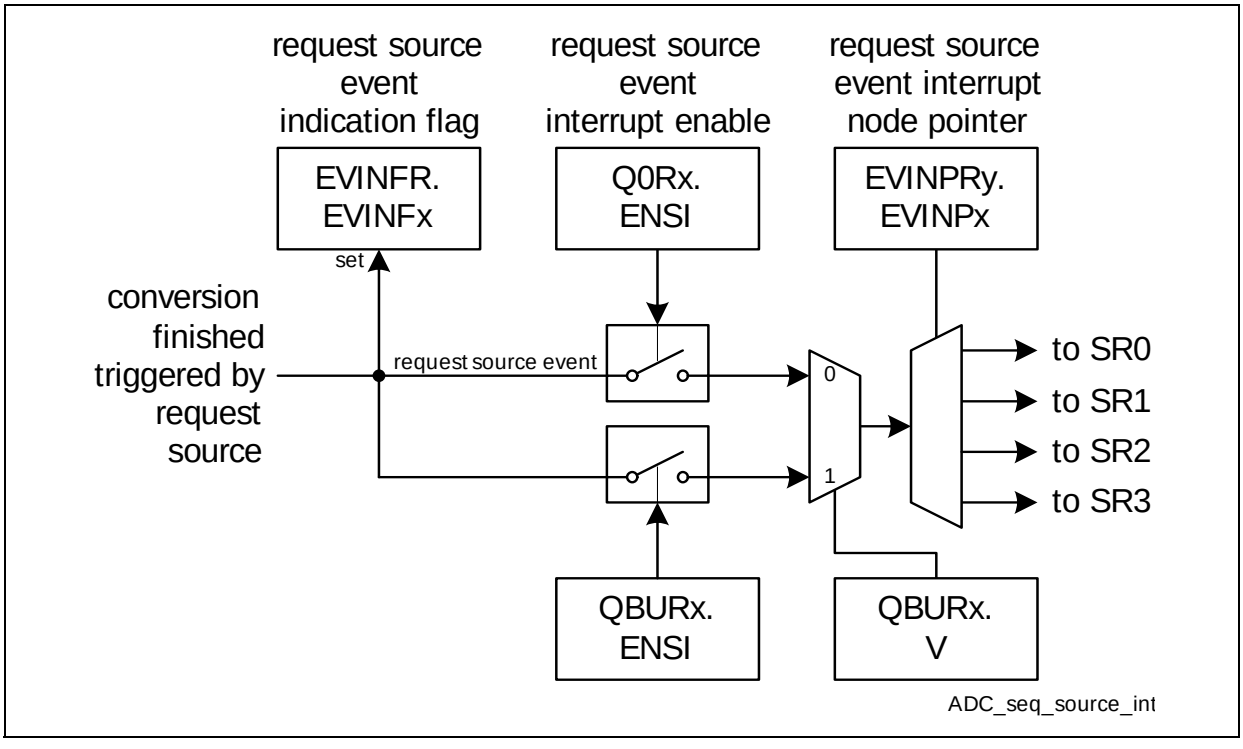


Figure 16-12 Interrupt Generation of a Sequential Request Source

16.2.11 Sequential Source Registers

16.2.11.1 Queue Mode Register

These registers contain the control and status bits of a sequential source.
The index 0/2 describes the number of the arbitration slot where the request source is taking part in the arbitration.

The queue mode register contains bits used to set the request source in the desired mode.

*Note: Before SW modifies the queue content by QMR.CLRV or QMR.FLUSH, all HW actions related to this queue have to be finished. Therefore, the arbitration slot has to be disabled and SW has to wait for at least two arbitration rounds (to be sure that this request source can no longer be an arbitration winner). Then, it has to check **GLOBSTR**.CRSC and **GLOBSTR**.BUSY to be sure that a conversion triggered by this request source is no longer running. Then SW can read QBURx and Q0Rx and can start modification of the queue content.*

QMR0
Queue 0 Mode Register **XSFR($E0_H$)** **Reset Value: 0000_H**
QMR2
Queue 2 Mode Register **XSFR($F0_H$)** **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				CEV	FLUSH	TREV	CLRV	0					ENTR	ENGT	
r				w	w	w	w	r					rw	rw	

Field	Bits	Type	Description
ENGT	[1:0]	rw	**Enable Gate** This bit field enables the gating functionality for the request source. 00_B The request source does not issue conversion requests. 01_B The request source issues conversion requests if a valid conversion request is pending in the queue 0 register or in the backup register. 10_B The request source issues conversion requests if a valid conversion request is pending in the queue 0 register or in the backup register and the selected gating signal REQGTx = 1. 11_B The request source issues conversion requests if a valid conversion request is pending in the queue 0 register or in the backup register and the selected gating signal REQGTx = 0.
ENTR	2	rw	**Enable External Trigger** This bit enables the external trigger possibility. 0_B The external trigger is disabled and the trigger event is not generated. 1_B The external trigger is enabled and a trigger event is generated if a rising edge is detected at the selected trigger input signal for REQTRx.
CLRV	8	w	**Clear V Bit** 0_B No action. 1_B The next pending valid queue entry in the sequence and the event flag EV are cleared. If there is a valid entry in the queue backup register (QBUR.V = 1), this entry is cleared, otherwise the entry in queue register 0 is cleared.

Field	Bits	Type	Description
TREV	9	w	**Trigger Event** 0_B No action. 1_B A trigger event is generated by SW. If the a valid entry in the request source waits for a trigger event, a conversion request is started.
FLUSH	10	w	**Flush Queue** 0_B No action. 1_B All entries in the queue (including the backup stage) and the event flag EV are cleared. The queue contains no more valid entry.
CEV	11	w	**Clear Event Flag** 0_B No action. 1_B Bit EV is cleared.
0	[7:3], [15:8]	r	**Reserved** returns 0 if read; should be written with 0;

16.2.11.2 Queue Status Register

The queue status register contains bits indicating the status of the sequential source. The filling level and the empty information refer to the queue intermediate stages (if available) and to the queue register 0. An aborted conversion stored in the backup stage is not indicated by these bits (therefore, see QBURx.V).

QSR0
Queue 0 Status Register **XSFR($E2_H$)** **Reset Value: 0020_H**
QSR2
Queue 2 Status Register **XSFR($F2_H$)** **Reset Value: 0020_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0							EV	REQ GT	0	EMP TY	0			FILL	
r							rh	rh	r	rh	r			rh	

Field	Bits	Type	Description
FILL	[1:0]	rh	**Filling Level[1)]** This bit field indicates how many queue entries are valid in the sequential source. It is incremented each time a new entry is written to QINRx or by an enabled refill mechanism. It is decremented each time a requested conversion has been started. A new entry is ignored if the filling level has reached its maximum value. 00_B EMPTY = 1:There is no valid entry in the queue. EMPTY = 0:There is 1 valid entries in the queue. 01_B There are 2 valid entries in the queue. 10_B There are 3 valid entries in the queue. 11_B There are 4 valid entries in the queue.
EMPTY	5	rh	**Queue Empty** This bit indicates if the sequential source contains valid entries. 0_B There are FILL+1 valid entries in the queue. 1_B There are no valid entries (queue is empty).

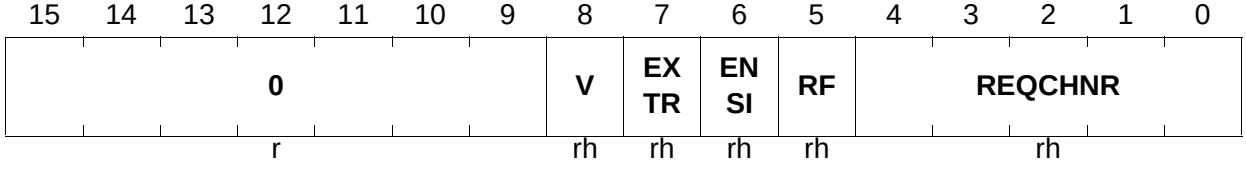
Field	Bits	Type	Description
REQGT	7	rh	**Request Gate Level** This bit monitors the level at the selected REQGT input. 0_B The level is 0. 1_B The level is 1.
EV	8	rh	**Event Detected** This bit indicates that an event has been detected while at least one valid entry has been in the queue (queue register 0 or backup stage). Once set, this bit is cleared automatically when the requested conversion is started. 0_B A trigger event has not been detected. 1_B A trigger event has been detected.
0	[4:2], 6, [15:9]	r	**Reserved** returns 0 if read; should be written with 0;

1) This bit field is always 00_B for the 1-stage queue in arbitration slot 0.

16.2.11.3 Queue 0 Register

The queue registers 0 monitor the status of the pending request (queue stage 0).

Q0R0
Queue 0 Register 0 **XSFR($E4_H$)** **Reset Value: 0000_H**
Q0R2
Queue 2 Register 0 **XSFR($F4_H$)** **Reset Value: 0000_H**



Field	Bits	Type	Description
REQCHNR	[4:0]	rh	**Request Channel Number** This bit field indicates the requested channel number.
RF	5	rh	**Refill** This bit indicates if the pending request is discarded after the conversion start or if it is automatically refilled into the queue input of the request queue. 0_B The request is discarded after the conversion start. 1_B The request is refilled into the queue after the conversion start.
ENSI	6	rh	**Enable Source Interrupt** This bit indicates if a request source event interrupt is generated when the conversion is finished. 0_B The request source event interrupt generation is disabled. 1_B The request source event interrupt generation is enabled.
EXTR	7	rh	**External Trigger** This bit indicates if a valid queue entry immediately leads to a conversion request or if the request handler waits for a trigger event. 0_B The request handler does not wait for a trigger event. 1_B The request handler waits for a trigger event.

Field	Bits	Type	Description
V	8	rh	**Request Channel Number Valid** This bit indicates if the queue register 0 contains a valid queue entry. 0_B The queue entry is not valid and does not lead to a conversion request. 1_B The queue entry is valid and leads to a conversion request.
0	[15:9]	r	**Reserved** returns 0 if read; should be written with 0;

16.2.11.4 Queue Backup Register

The queue backup registers monitor the status of an aborted sequential request.

The registers QBURx and QINRx share the same register address. A read operation at this register address will deliver the “rh” bits of register QBURx. A write operation to this address will target the “w” bits in register QINRx.

QBUR0
Queue 0 Backup Register **XSFR($E6_H$)** **Reset Value: 0000_H**
QBUR2
Queue 2 Backup Register **XSFR($F6_H$)** **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0							V	EXTR	ENSI	RF	REQCHNR				
r							rh	rh	rh	rh	rh				

Field	Bits	Type	Description
REQCHNR	[4:0]	rh	**Request Channel Number** This bit field contains the channel number of an aborted conversion that has been requested by this request source.
RF	5	rh	**Refill** This bit contains the refill bit of an aborted conversion that has been requested by this request source.
ENSI	6	rh	**Enable Source Interrupt** This bit contains the request source event interrupt enable bit of an aborted conversion that has been requested by this request source.
EXTR	7	rh	**External Trigger** This bit contains the external trigger bit of an aborted conversion that has been requested by this request source.

Field	Bits	Type	Description
V	8	rh	**Request Channel Number Valid** This bit indicates if the entry in the queue backup register is valid (REQCHNR, RF, TR and ENSI are valid). Bit V is set if a running conversion that has been requested by this request source is aborted. It is cleared when the repeated conversion is started. 0_B The backup register does not contain a valid entry. 1_B The backup register contains a valid entry. It will be requested before a valid entry in queue register 0 will be requested.
0	[15:9]	r	**Reserved** returns 0 if read; should be written with 0;

16.2.11.5 Queue Input Register

The queue input register is the entry point for conversion requests of a sequential request source.

The registers QBURx and QINRx share the same register address. A read operation at this register address will deliver the “rh” bits of register QBURx. A write operation to this address will target the “w” bits in register QINRx.

QINR0
Queue Input Register 0 **XSFR($E6_H$)** **Reset Value: 0000_H**
QINR2
Queue Input Register 2 **XSFR($F6_H$)** **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								EXTR	ENSI	RF	REQCHNR				
r								w	w	w	w				

Field	Bits	Type	Description
REQCHNR	[4:0]	w	**Request Channel Number** This bit field defines the requested channel number.
RF	5	w	**Refill** This bit defines the refill functionality for this queue entry. 0_B The content of this queue entry is not entered again in QINRx when the related conversion is started. 1_B The content of this queue entry is automatically entered again in QINRx when the related conversion is started.
ENSI	6	w	**Enable Source Interrupt** This bit defines the source interrupt functionality. 0_B A request source event interrupt is not generated if the related conversion is finished. 1_B A request source event interrupt is generated if the related conversion is finished.

Field	Bits	Type	Description
EXTR	7	w	**External Trigger** This bit defines the external trigger functionality. 0_B A valid queue entry immediately leads to a conversion request. 1_B A valid queue entry waits for a trigger event to occur before issuing a conversion request.
0	[15:8]	r	**Reserved** returns 0 if read; should be written with 0;

16.2.12 Channel-Related Functions

The channel control unit defines the conversion settings, that can be programmed individually for each analog input channel. Therefore, a channel control register **CHCTRx (x = 0 - 15)** is associated to each analog input channel CHx. After the arbiter has determined the channel to be converted, the defined settings are applied to the AD converter, comprising information about:

- **Conversion parameters:**
  Bit field ICLSEL defines which input class is taken into account for the conversion (see **Section 16.2.12.1**).
- **Reference selection:**
  Bit field REFSEL defines which reference input is used for the conversion (see **Section 16.2.12.2**)
- **Channel event handling:**
  Bit fields LCC, BNDASEL, and BNDBSEL define which boundaries are used for limit checking (see **Section 16.2.12.4**) and which channel event leads to a channel event interrupt (see **Section 16.2.12.5**).
- **Synchronous conversion request:**
  Bit SYNC defines if the channel triggers a synchronized conversion (see **Section 16.2.17**).

In addition to the general channel control, the ADC kernel supports a mechanism (named alias feature, see **Section 16.2.12.3**) to redirect a conversion request to another channel number.

16.2.12.1 Input Classes

An input class defines the length of the sample phase and the resolution of the conversion. In most applications, the characteristics of the input circuitries (RC input low-pass filter and impedance of the signal source) are quite similar for several analog input signals, leading to similar timings for the sample phase of these channels. As a consequence, input channels with similar parameters can be grouped together to form an input class.
All channels with the same ICLSEL setting belong to the same input class and have the same sample phase length and resolution. In the XC2000, 2 input classes are supported. Registers **INPCRx (x = 0 - 1)** can be programmed to adjust the sample time and the resolution to the application requirements independently for each input class.
The default setting of these registers lead to the minimum sample phase length of 2 f_{ADCI} cycles and conversions with 10 bits resolution. If this default setting fits to the application requirements, bit fields **CHCTRx (x = 0 - 15)**.ICLSEL and registers **INPCRx (x = 0 - 1)** need not to be changed.

16.2.12.2 Reference Selection

The conversion result of the ADC is always referring to a reference voltage. The maximum digital result value (full scale) is obtained if the analog input voltage equals the reference voltage. In order to support more than one measurement range with full scale digital representation, the user can select between the standard reference input VAREF and an alternative reference input at the analog input channel CH0 for each ADC kernel. The reference selection can be individually programmed for each input channel.
This feature can be used to connect 5 V based sensors and 3.3 V based sensors to the same ADC kernel. In this case, one set of input channels refers to the standard reference input, whereas the other one refers to the voltage level at input CH0.
Please note that the smallest granularity 1 LSB_n for n bit resolution refers to the selected reference voltage. The granularity becomes very small if a low reference voltage is applied, and as a consequence, the resulting TUE increases due to noise effects. Therefore it is recommended to avoid small reference voltages.

16.2.12.3 Alias Feature

The ADC kernel provides an alias feature, allowing a re-direction of conversion requests for channels CH0 or CH1 to other channel numbers. This feature can be used to measure the same input channel and to store the conversion results in two different result registers.

- The same signal can be measured twice without the need to read out the conversion result to avoid data loss. This allows triggering both conversions quickly one after the other and being independent from CPU interrupt latency.
- The sensor signal is connected to only one input channel (instead of two analog inputs). This saves input pins in low-cost applications and only the leakage of one input has to be considered in the error calculation.
- Even if the analog input CH0 is used as alternative reference (see **Figure 16-13**), the internal trigger and data handling features for channel CH0 can be used.
- The channel settings for both conversions can be different (boundary values, interrupts, etc.).
- If a sequential conversion request source has been set up, a conversion request for channels CH0 or CH1 can be easily directed to other input channels without flushing the queue.

In typical low-cost AC-drive applications, only one common current sensor is used to determine the phase currents. Depending on the applied PWM pattern, the measured value has different meanings and the sample points have to be precisely located in the PWM period. **Figure 16-13** shows an example where the sensor signal is connected to one input channel (CHx) but two conversions are triggered for two different channels (CHx and CH0). With the alias feature, a conversion request for CH0 leads to a conversion of the analog input CHx instead of CH0, but taking into account the settings for CH0. Although the same analog input (CHx) has been measured, the conversion

results can be stored and read out from the result registers RESRx (conversion triggered for CHx) and RESRy (conversion triggered for CH0). Additionally, different interrupts or limit boundaries can be selected, enabled or disabled.

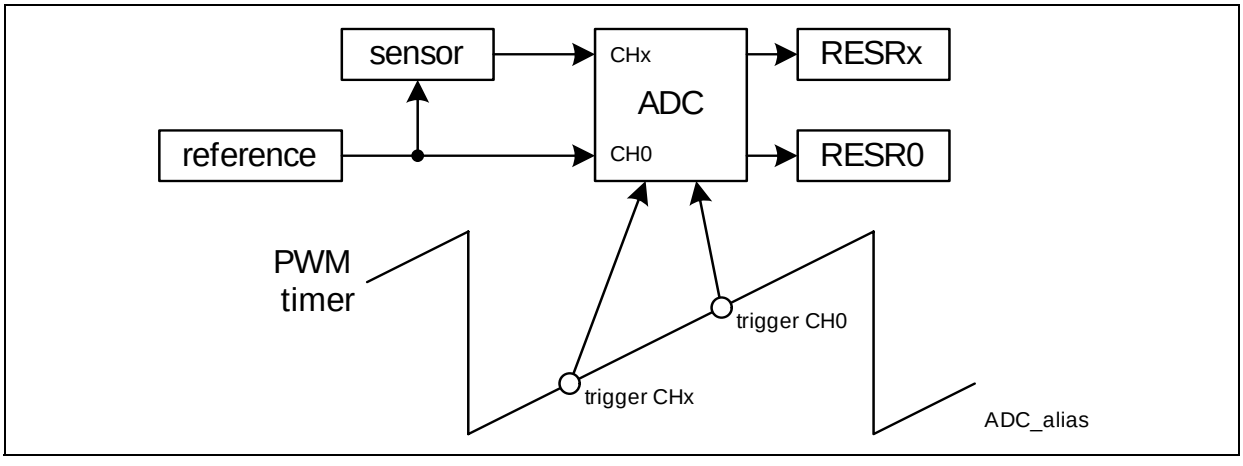


Figure 16-13 Alias Feature

16.2.12.4 Limit Checking

The limit checking mechanism automatically compares each conversion result to two boundary values (boundaries A and B). For each channel, the user can select these boundaries from a set of 4 programmable values (**LCBR0** to **LCBR3**).
With this structure, the conversion result range is split into three areas:

- Area I: The conversion result is below or equal to both boundaries.
- Area II: The conversion result is above one boundary and below or equal to the other boundary.
- Area III: The conversion result is above both boundaries.

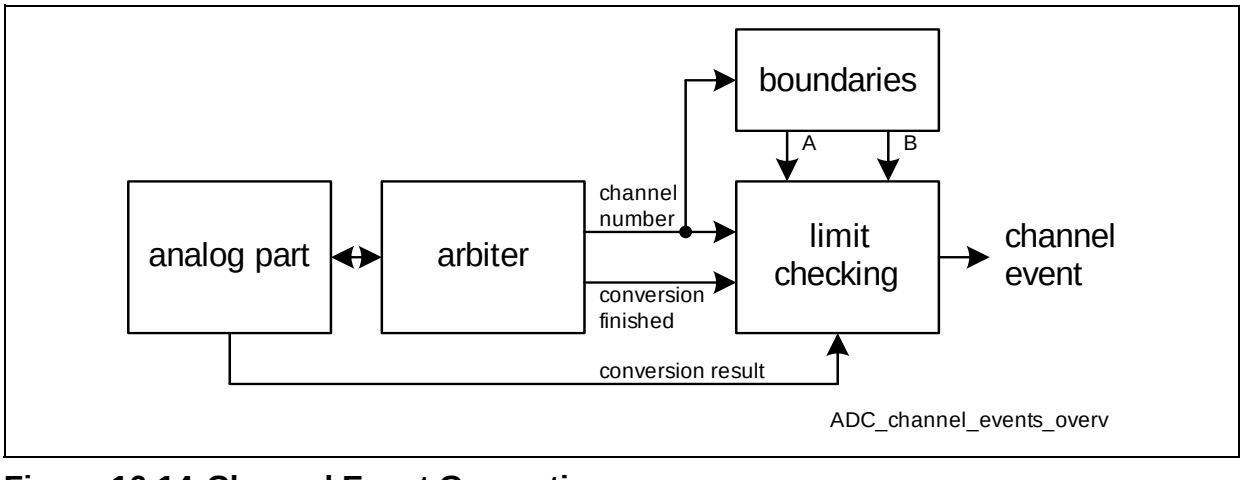


Figure 16-14 Channel Event Generation

Bit field LCC in the channel control register defines the condition to generate a channel event, leading to a channel event interrupt:

- LCC = 000_B: No trigger, the channel event generation is disabled.
- LCC = 001_B: A channel event is generated if the conversion result is not in area I.
- LCC = 010_B: A channel event is generated if the conversion result is not in area II.
- LCC = 011_B: A channel event is generated if the conversion result is not in area III.
- LCC = 100_B: A channel event is always generated (regardless of the boundaries).
- LCC = 101_B: A channel event is generated if the conversion result is in area I.
- LCC = 110_B: A channel event is generated if the conversion result is in area II.
- LCC = 111_B: A channel event is generated if the conversion result is in area III.

Figure 16-15 shows an example for limit checking where channel events are generated only if the conversion results are not in the normal operating range defined by area II (LCC = 010_B).
Typical applications for limit checking are temperature monitoring or overcurrent sensing. As long as the measured temperature value is below a boundary value, the CPU does not need to be informed. In this case, a channel event should be generated only if the conversion result is in area III (LCC = 111_B) to indicate an over-temperature condition. If the conversion of the analog temperature input signal is part of an auto-scan sequence autonomously triggered on a regular time base, the CPU load for the temperature monitoring is zero until the over-temperature condition is detected.
In the case of an over-current protection, the channel event can be used to disable PWM generation to reduce the current (in the XC2000, an interrupt output line of the ADC module is connected to a corresponding input of the CCU6x units to allow fast reactions without CPU intervention).

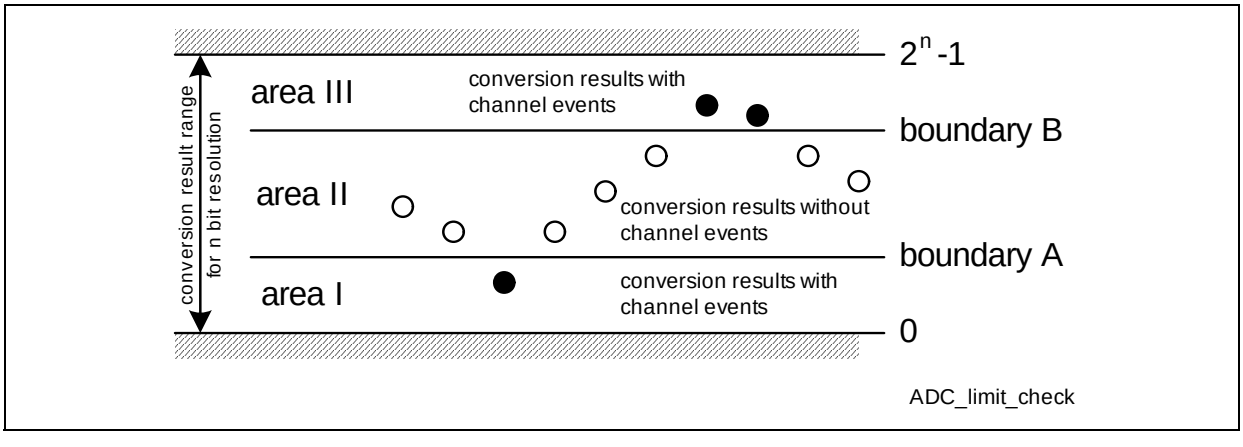


Figure 16-15 Limit Checking

Note: It is also possible to select the same boundary register for boundaries A and B. In this case, the conversion result range is split into two ranges (area II is empty).

16.2.12.5 Channel Event Interrupts

A channel event interrupt can be generated based on a channel event according to the structure shown in **Figure 16-16**. If a channel event is detected, it sets the corresponding indication flag in register **CHINFR**. These flags can also be set by writing a 1 to the corresponding bit position, whereas writing 0 has no effect. The indication flags can be cleared by SW by writing a 1 to the corresponding bit position in register **CHINCR**.

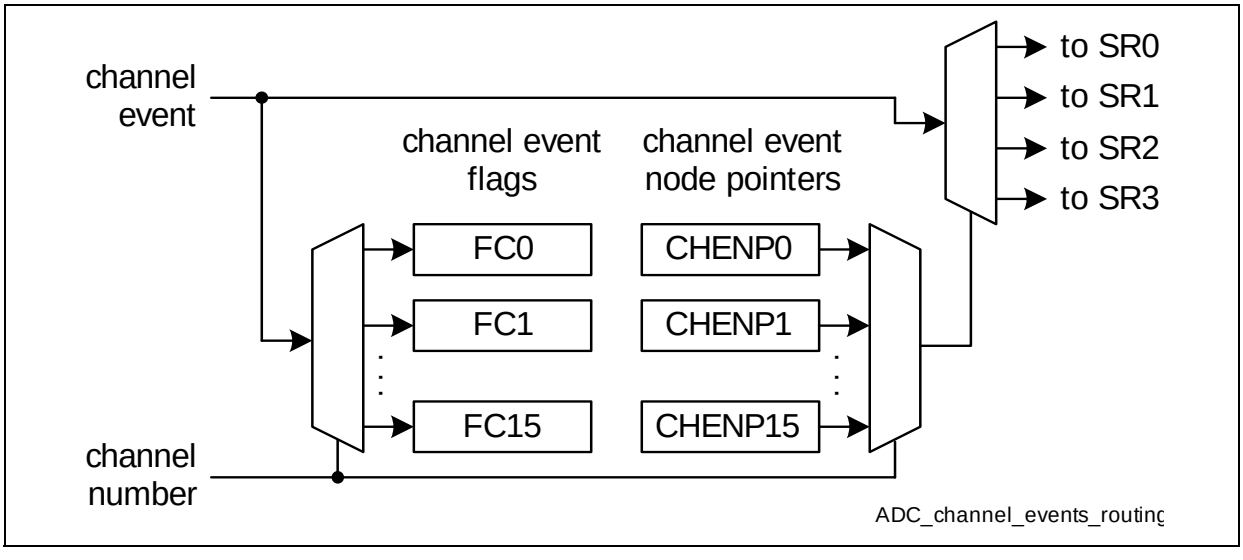


Figure 16-16 Channel Event Interrupt Generation

The service request output line SRx that is selected by the channel node pointer bit fields in registers **CHINPR0**, **CHINPR4**, **CHINPR8**, or **CHINPR12** is activated each time the related channel event is detected or the related bit position in register **CHINFR** is written with a 1.

16.2.13 Channel-Related Registers

16.2.13.1 Channel Control Registers

The channel control registers contain bits to select the targeted result register, to control the limit check mechanism and to select an input class.
The channel control register 0 defines the settings for the input channel CH0, etc.

CHCTRx (x = 0 - 15)
Channel x Control Register **XSFR(20_H + x * 2)** **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	RESR SEL			ICL SEL		REF SEL		SYN C	LCC			BNDB SEL		BNDA SEL	
r	rw			rw		rw		rw	rw			rw		rw	

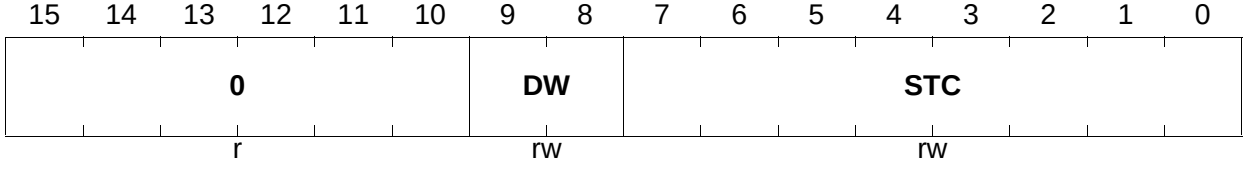
Field	Bits	Type	Description
BNDASEL	[1:0]	rw	**Boundary A Selection** This bit field defines which boundary will be taken as boundary A for the limit checking. 00_B The value given by register LCBR0 is selected. 01_B The value given by register LCBR1 is selected. 10_B The value given by register LCBR2 is selected. 11_B The value given by register LCBR3 is selected.
BNDBSEL	[3:2]	rw	**Boundary B Selection** This bit field defines which boundary will be taken as boundary B for the limit checking. 00_B The value given by register LCBR0 is selected. 01_B The value given by register LCBR1 is selected. 10_B The value given by register LCBR2 is selected. 11_B The value given by register LCBR3 is selected.
LCC	[6:4]	rw	**Limit Check Control** This bit field defines the behavior of the limit checking mechanism. Please refer to the coding in **Section 16.2.12.4** on **Page 16-65**.

Field	Bits	Type	Description
SYNC	7	rw	**Synchronization Request** This bit defines if a conversion request for this channel leads to a synchronized (parallel) conversion with other ADC kernels. This bit is only taken into account if the ADC kernel is a potential conversion master (**SYNCTR**.STSEL = 00), otherwise it is considered to be 0. 0_B This channel does not request a synchronized conversion. 1_B This channel requests a synchronized conversion if the ADC kernel is a potential synchronization master.
REFSEL	[9:8]	rw	**Reference Input Selection** This bit field defines the reference source for this channel. 00_B The standard reference input VAREF is selected. 01_B The alternative reference input CH0 is selected. 10_B reserved, do not use 11_B reserved, do not use
ICLSEL	[11,10]	rw	**Input Class Selection** These bits are used to select the input class. 00_B The input class 0 is selected. 01_B The input class 1 is selected. 10_B reserved, do not use 11_B reserved, do not use
RESRSEL	[14:12]	rw	**Result Register Selection** This bit field defines which result register will be the target of the conversion result of this channel. 000_B The result register 0 is selected. 001_B The result register 1 is selected. ... 111_B The result register 7 is selected.
0	15	r	**Reserved** returns 0 if read; should be written with 0;

16.2.13.2 Input Class Registers

The input class registers contain bits to control the sample time and the resolution for each input class.
The input class register 0 defines the settings for the input class 0, etc.

INPCRx (x = 0 - 1)
Input Class Register x **XSFR($C0_H$ + x * 2)** **Reset Value: 0000_H**



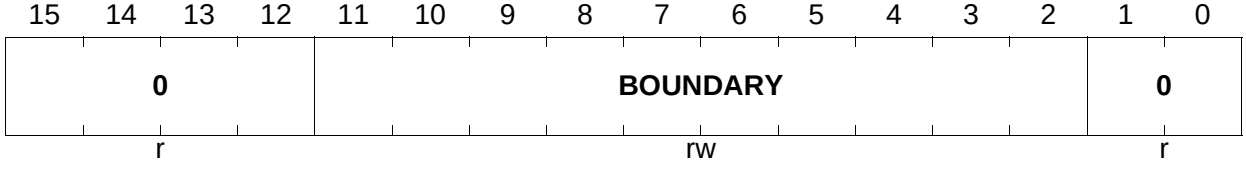
Field	Bits	Type	Description
STC	[7:0]	rw	**Sample Time Control** This bit field defines the additional length of the sample phase, given in analog clock cycles f_{ADCI}. A minimum sample phase of 2 analog clock cycles is extended by the programmed value. sample phase length = (2 + STC) / f_{ADCI}
DW	[9:8]	rw	**Data Width** This bit field defines how many bits are converted for the result[1]. The MSBs of conversion results with different DW settings are left aligned in the result bit fields. Bit positions that are not converted are 0. 00_B The result is 10 bits wide. 01_B reserved, do not use 10_B The result is 8 bits wide. 11_B reserved, do not use
0	[15:10]	r	**Reserved** returns 0 if read; should be written with 0;

1) The setting 00_B is default. In case a 10-bit AD converter is used, the combinations 01_B and 11_B are ignored by the converter and treated like 00_B.

16.2.13.3 Limit Check Boundary Registers

The bit fields in these registers define compare value (boundary) for the limit checking unit. The reset values of the boundaries are defined as 10%, 90%, 33% and 66% of the complete result range (in 12 bit representation).

LCBR0
Limit Check Boundary Register 0 XSFR(84_H) Reset Value: 0198_H
LCBR1
Limit Check Boundary Register 1 XSFR(86_H) Reset Value: $0E64_H$
LCBR2
Limit Check Boundary Register 2 XSFR(88_H) Reset Value: 0554_H
LCBR3
Limit Check Boundary Register 3 XSFR($8A_H$) Reset Value: $0AA8_H$



Field	Bits	Type	Description
BOUNDARY	[11:2]	rw	**Boundary for Limit Checking** This bit field contains the value for the limit checking unit that is compared to the actual conversion result. The result of the limit check is used for the generation of the channel event, see **Section 16.2.12.4**.
0	[1:0], [15:12]	r	**Reserved** returns 0 if read; should be written with 0;

16.2.13.4 Channel Event Indication Flag Register

The channel event indication flag register CHINFR monitors the detected channel events.

CHINFR
Channel Event Indication Flag RegisterXSFR(90_H) **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHINF15	**CHINF14**	**CHINF13**	**CHINF12**	**CHINF11**	**CHINF10**	**CHINF9**	**CHINF8**	**CHINF7**	**CHINF6**	**CHINF5**	**CHINF4**	**CHINF3**	**CHINF2**	**CHINF1**	**CHINF0**
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
CHINFx (x = 0 - 15)	x	rwh	**Channel x Event Indication Flag** Flag CHINFx indicates that a channel event for channel x has been detected. Writing a 0 has no effect, whereas writing a 1 sets the written bit position and generates the corresponding interrupt request. 0_B A channel x event has not yet been detected. 1_B A channel x event has been detected.

16.2.13.5 Clear Channel Event Indication Register

Writing a 1 to a bit position in the channel indication clear register CHINCR clears the corresponding channel event indication flag CHINFx in register **CHINFR**. If a channel event is detected when the corresponding bit position is written with a 1, flag CHINFx is cleared.

CHINCR
Channel Event Indication Clear Register

XSFR(92_H) **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHINC15	**CHINC14**	**CHINC13**	**CHINC12**	**CHINC11**	**CHINC10**	**CHINC9**	**CHINC8**	**CHINC7**	**CHINC6**	**CHINC5**	**CHINC4**	**CHINC3**	**CHINC2**	**CHINC1**	**CHINC0**
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
CHINCx (x = 0 - 15)	x	w	**Clear Channel Indication Flag** 0 No action. 1 Flag CHINFR.x is cleared.

16.2.13.6 Channel Interrupt Node Pointer Registers

The bit fields in these registers define the service request output ADCx_SR[3:0] that is used to signal a channel event interrupt.

CHINPR0
Channel Interrupt Node Pointer Register 0

XSFR(98_H) **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		**CHINP3**		**0**		**CHINP2**		**0**		**CHINP1**		**0**		**CHINP0**	
r		rw		r		rw		r		rw		r		rw	

Field	Bits	Type	Description
CHINP0, CHINP1, CHINP2, CHINP3	[1:0], [5:4], [9:8], [13:12]	rw	**Interrupt Node Pointer for Channel x** This bit field selects which service request output indicates a channel event interrupt of channel x. 00_B Service request output SR0 is selected. 01_B Service request output SR1 is selected. 10_B Service request output SR2 is selected. 11_B Service request output SR3 is selected.
0	[3:2], [7:6], [11:10], [15:14]	r	**Reserved** returns 0 if read; should be written with 0;

CHINPR4
Channel Interrupt Node Pointer Register 4

XSFR($9A_H$) **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		**CHINP7**		**0**		**CHINP6**		**0**		**CHINP5**		**0**		**CHINP4**	
r		rw		r		rw		r		rw		r		rw	

Field	Bits	Type	Description
CHINP4, CHINP5, CHINP6, CHINP7	[1:0], [5:4], [9:8], [13:12]	rw	**Interrupt Node Pointer for Channel x** This bit field selects which service request output indicates a channel event interrupt of channel x. 00_B Service request output SR0 is selected. 01_B Service request output SR1 is selected. 10_B Service request output SR2 is selected. 11_B Service request output SR3 is selected.
0	[3:2], [7:6], [11:10], [15:14]	r	**Reserved** returns 0 if read; should be written with 0;

CHINPR8
Channel Interrupt Node Pointer Register 8

XSFR($9C_H$) **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		**CHINP11**		**0**		**CHINP10**		**0**		**CHINP9**		**0**		**CHINP8**	
r		rw		r		rw		r		rw		r		rw	

Field	Bits	Type	Description
CHINP8, CHINP9, CHINP10, CHINP11	[1:0], [5:4], [9:8], [13:12]	rw	**Interrupt Node Pointer for Channel x** This bit field selects which service request output indicates a channel event interrupt of channel x. 00_B Service request output SR0 is selected. 01_B Service request output SR1 is selected. 10_B Service request output SR2 is selected. 11_B Service request output SR3 is selected.
0	[3:2], [7:6], [11:10], [15:14]	r	**Reserved** returns 0 if read; should be written with 0;

CHINPR12
Channel Interrupt Node Pointer Register 12

XSFR($9E_H$) **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		**CHINP15**		**0**		**CHINP14**		**0**		**CHINP13**		**0**		**CHINP12**	
r		rw		r		rw		r		rw		r		rw	

Field	Bits	Type	Description
CHINP12, CHINP13, CHINP14, CHINP15	[1:0], [5:4], [9:8], [13:12]	rw	**Interrupt Node Pointer for Channel x** This bit field selects which service request output indicates a channel event interrupt of channel x. 00_B Service request output SR0 is selected. 01_B Service request output SR1 is selected. 10_B Service request output SR2 is selected. 11_B Service request output SR3 is selected.
0	[3:2], [7:6], [11:10], [15:14]	r	**Reserved** returns 0 if read; should be written with 0;

16.2.13.7 Alias Register

The alias register contains bits to change a requested channel number from CH0 and CH1 to another channel number, see also **Section 16.2.12.3**. The programmed alias channel number is replacing the internally requested number for analog input multiplexer (of the converter). The internally requested channel number is taken into account for all other internal actions and the synchronization request.

ALR0
Alias Register 0 **XSFR($1C_H$)** **Reset Value: 0100_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0			**0**	**ALIAS1**				**0**			**0**	**ALIAS0**			
r			rw	rw				r			rw	rw			

Field	Bits	Type	Description
ALIAS0	[3:0]	rw	**Alias Value for CH0 Conversion Requests** The channel indicated in this bit field is converted instead of channel CH0. The conversion is done with the settings defined for channel CH0.
ALIAS1	[11:8]	rw	**Alias Value for CH1 Conversion Requests** The channel indicated in this bit field is converted instead of channel CH1. The conversion is done with the settings defined for channel CH1.
0	4, 12	rw	**Reserved for Future Use** Bit is reserved for future use and has to be written with 0_B.
0	[7:5], [15:13]	r	**Reserved** returns 0 if read; should be written with 0;

16.2.14 Conversion Result Handling

The result generation part handles the:

- Storage of the conversion results (see **Section 16.2.14.1**)
- Wait-for-read mode (see **Section 16.2.14.2**)
- Result event interrupts (see **Section 16.2.14.3**)
- Result FIFO buffer (see **Section 16.2.14.4**)
- Data reduction or anti-aliasing filtering (see **Section 16.2.14.5**)

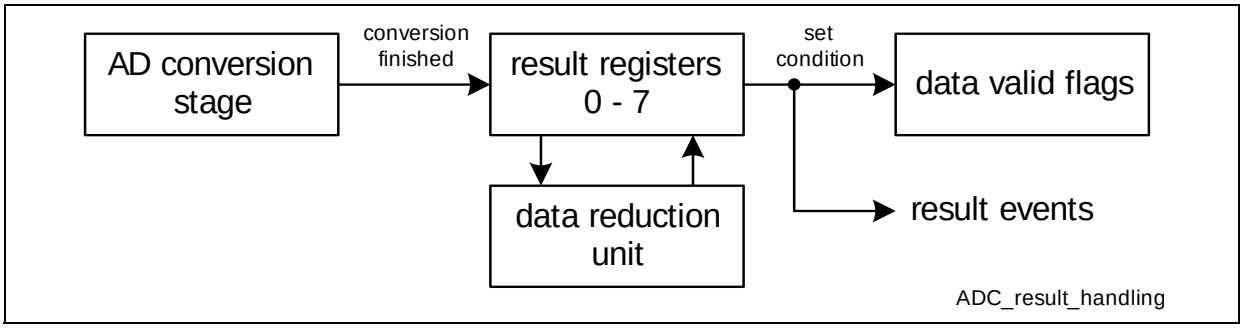


Figure 16-17 Conversion Result Handling

16.2.14.1 Storage of Conversion Results

For each analog input channel, the associated channel control register **CHCTRx (x = 0 - 15)** contains a pointer bit field (RESRSEL) defining the result register to store the conversion result of this channel. This structure allows the user to direct conversion results of different channels to one or more result registers. Depending on the application needs (data reduction, auto-scan, alias feature, result FIFO, etc.), the user can distribute the conversion results to minimize CPU load or to be more tolerant against interrupt latency.
An individual data valid flag **VFR**.VFx for each result register indicates that “new” valid data has been stored in the corresponding result register and can be read out.
Due to different result handling mechanisms, the conversion result can be represented in different ways:

- **Data reduction filter disabled:**
  The conversion result is maximum 10 bits wide with the MSB of the conversion result being always at bit position 11 and the remaining LSBs filled with 0.
  The data valid flag is set and a result event occurs each time a new conversion result is stored in the result register.
  It is possible to share a result register among several analog input channels.
- **Data reduction filter enabled:**
  The conversion result is maximum 10 bits wide with the MSB of the conversion result being always at bit position 11 and the remaining LSBs filled with 0. The additional bits [13:12] show the MSBs of the data accumulation.
  The data valid flag is set and a result event occurs each time a data reduction

sequence is finished and the final result is available in the result register.
The channel number is not included in the result register read view.

In order to support a wait-for-read and FIFO buffer features, the valid flag has to be cleared automatically when SW does a read access or the result is transferred into another FIFO element (if result FIFO buffering is enabled).
This behavior is contradictory to debugging requirements. For debugging, it has to be possible to introduce read or write commands into the normal program flow, e.g. to monitor conversion results. If a debugger reads out a result register, it would change the status of the conversion result from valid = “new” (not yet read out) to “old” (already read out). This would have an undesired impact on the application.
Therefore, the read views with “V” deliver the same value as the read views without “V”, but without clearing the valid bit. As a result, a debugger using read views with “V” can monitor the conversion results without influencing their status for the application.

The application requirements for results with enabled or disabled data reduction filter being different and debugger accesses can occur, four different scenarios with different result register read views are supported. The four read views refer to the same result register contents, but show a different behavior according to the address that has been read:

- Standard read view **RESRx (x = 0-7)**:
  The data reduction filter has to be disabled, the channel number is included to identify which channel has been converted, and a read action clears the corresponding valid bit. This representation is compatible to the ADC result register in XC16x devices.
- Read view **RESRAx (x = 0-7)**:
  The data reduction filter can be enabled, the channel number is not included, and a read action clears the corresponding valid bit.
- Read view **RESRVx (x = 0-7)** for debugger:
  The data reduction filter has to be disabled, the channel number is included, but a read action does not clear the corresponding valid bit.
- Read view **RESRAVx (x = 0-7)** for debugger:
  The data reduction filter can be enabled, the channel number is not included, but a read action does not clear the corresponding valid bit.

16.2.14.2 Wait-for-Read Mode

The wait-for-read mode is a feature of a result register allowing the CPU (or PEC) to treat each conversion result independently without the risk of data loss. Data loss could occur if the CPU does not read a conversion result from a result register before a new result overwrites the previous one.
Especially for auto-scan conversion sequences (or other sequences with “relaxed” timing requirements), the wait-for-read offers the possibility to request a conversion sequence according to an event (HW or SW), but to start a new conversion according to the CPU capability to read the formerly converted result.

If wait-for-read mode is enabled for a result register by setting bit WFR in register **RCRx (x = 0 - 7)**, a request source does not generate a conversion request while the targeted result register contains valid data (indicated by the valid flag VFx = 1) or if a currently running conversion targets the same result register.
A new conversion request is generated only after the targeted result register has been read out.
If two request sources target the same result register with wait-for-read selected, a lower priority request started before the higher priority source has requested its conversion can not be interrupted by the higher priority request. If a higher priority request targets a different result register, the lower priority conversion can be cancelled and repeated afterwards.

16.2.14.3 Result Event Interrupts

A result event interrupt can be generated based on a result event according to the structure shown in **Figure 16-18**. If a result event is detected, it sets the corresponding indication flag in register **EVINFR**. These flags can also be set by writing a 1 to the corresponding bit position, whereas writing 0 has no effect. The indication flags can be cleared by SW by writing a 1 to the corresponding bit position in register **EVINCR**.

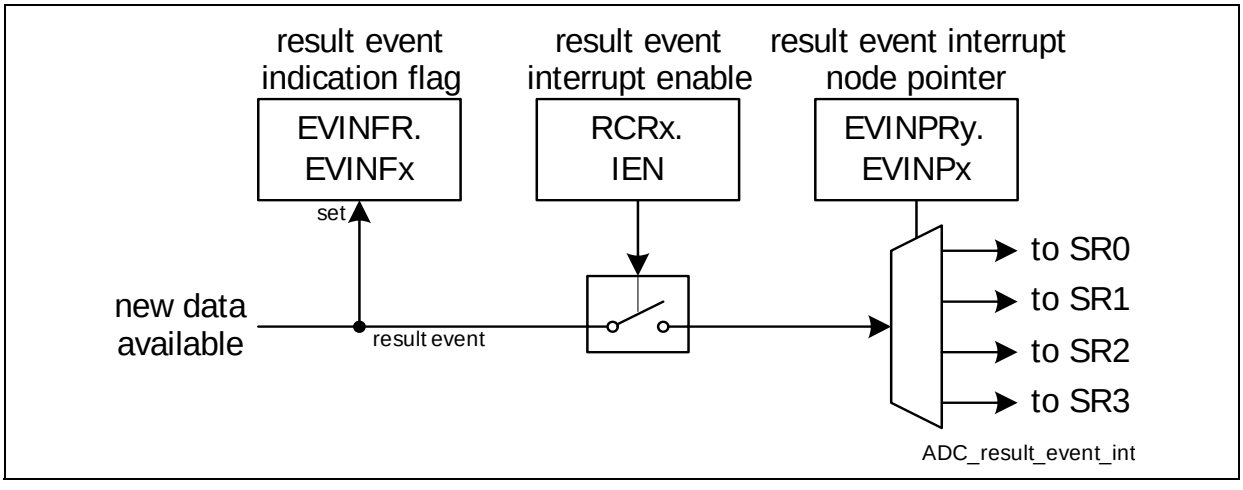


Figure 16-18 Result Event Interrupt Generation

The service request output line SRx that is selected by the result event interrupt node pointer bit fields in registers **EVINPR8** or **EVINPR12** issues an interrupt each time the related result event is detected or the related bit position in register **EVINFR** is written with a 1.

The result events and the request source events share the same registers. The result events are located at the following bit positions in register **EVINFR**:

- Event 8: Result event of result register 0.
- Event 9: Result event of result register 1.
- ...
- Event 15: Result event of result register 7.

16.2.14.4 Result FIFO Buffer

If a result register is not used as direct target for a conversion result, it can be concatenated with other result registers of the same ADC kernel to form a result FIFO buffer (first-in-first-out buffer mechanism). This allows to store measurement results and to read them out later with a “relaxed” CPU access timing. It is possible to set up more than one FIFO buffer structure with the available result registers.
A FIFO structure can be built by at least two “neighbor” result registers with the indices x and z = x+1, where result register z represents the input and result register x represents the output of the FIFO buffer. The conversion result has to be delivered by the converter stage to the FIFO input, whereas the buffered data has to be read out from the FIFO output.
The FIFO buffer function can be enabled by setting bit FEN in registers **RCRx (x = 0 - 7)**.

In the example shown in **Figure 16-19**, the result registers have been configured to form two FIFO buffers with two buffer stages (result registers 0/1 and 6/7, respectively), one FIFO buffer with three buffer stages (result registers 2/3/4), whereas result register 5 is used as “normal” result register without additional FIFO buffer functionality.

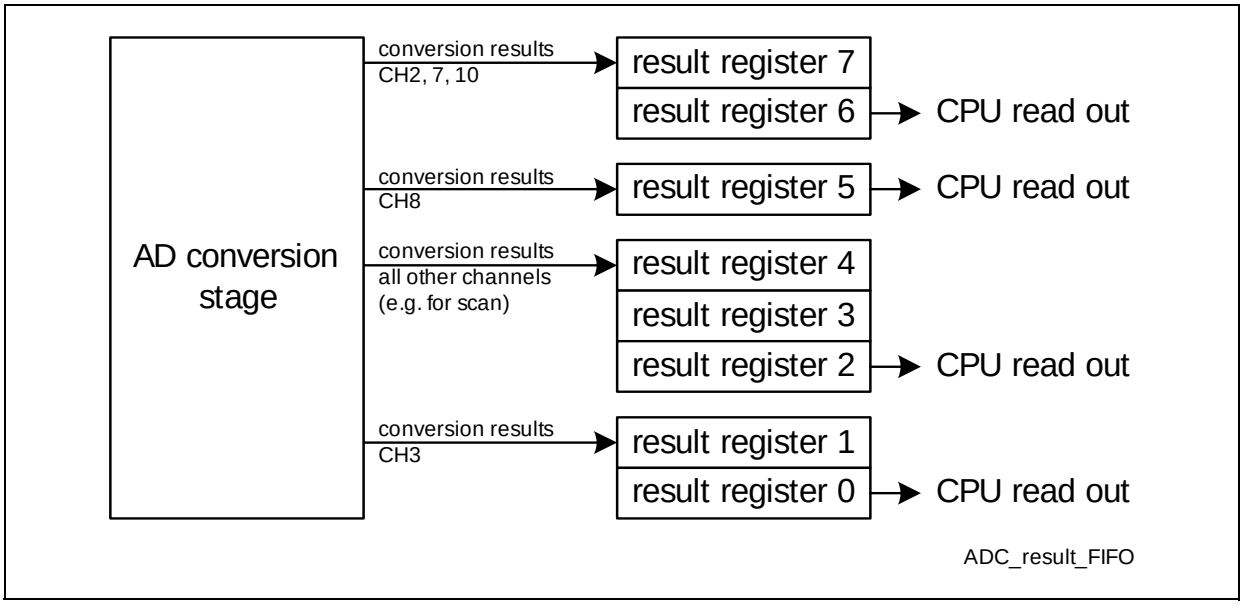


Figure 16-19 Result FIFO Buffers

If more than two result neighbor registers are concatenated to a FIFO buffer (from result register z to result register x, with z > x), the one with the highest index (z) is always the input and the one with the lowest index (x) is always the output. All intermediate result registers y (x < y < z) are used as intermediate FIFO stages without data input or data output functionality.

Result register features for each FIFO buffer:

- Result register z (**FIFO buffer input**):
  This result register can be enabled for data reduction. The wait-for-read mode is

supported to avoid data loss if the FIFO is full. Result event interrupt generation is not supported. Must not be read at a read view modifying the valid bit.

- Result register y (**intermediate buffer stage**):
  This/these result register(s) must not be enabled neither for wait-for-read mode, nor for data reduction. Result event interrupt generation is not supported. Must not be read at a read view modifying the valid bit, nor be the target of a conversion result.
- Result register x (**FIFO buffer output**):
  This result register can be enabled for result event interrupt generation to inform the CPU that new data can be read out from this register location. Data reduction and wait-for-read are not supported and have to be disabled. Must not be the target of a conversion result.
  If enabled, a result interrupt is generated for each data word in the FIFO.

16.2.14.5 Data Reduction Filter

The data reduction filter can be used as digital filter for anti-aliasing or decimation purposes. It can accumulate a maximum of 4 conversion results to generate a final result.

Each result register can be individually enabled for data reduction. The feature is controlled by bit field DRCTR in registers **RCRx (x = 0 - 7)**. The actual status is given by bit field DRC (data reduction counter) in the same register.
Conversion delivering results to other result registers do not influence the data reduction filter of result register x. As a consequence, other channels can be converted between two conversions targeting result register x.

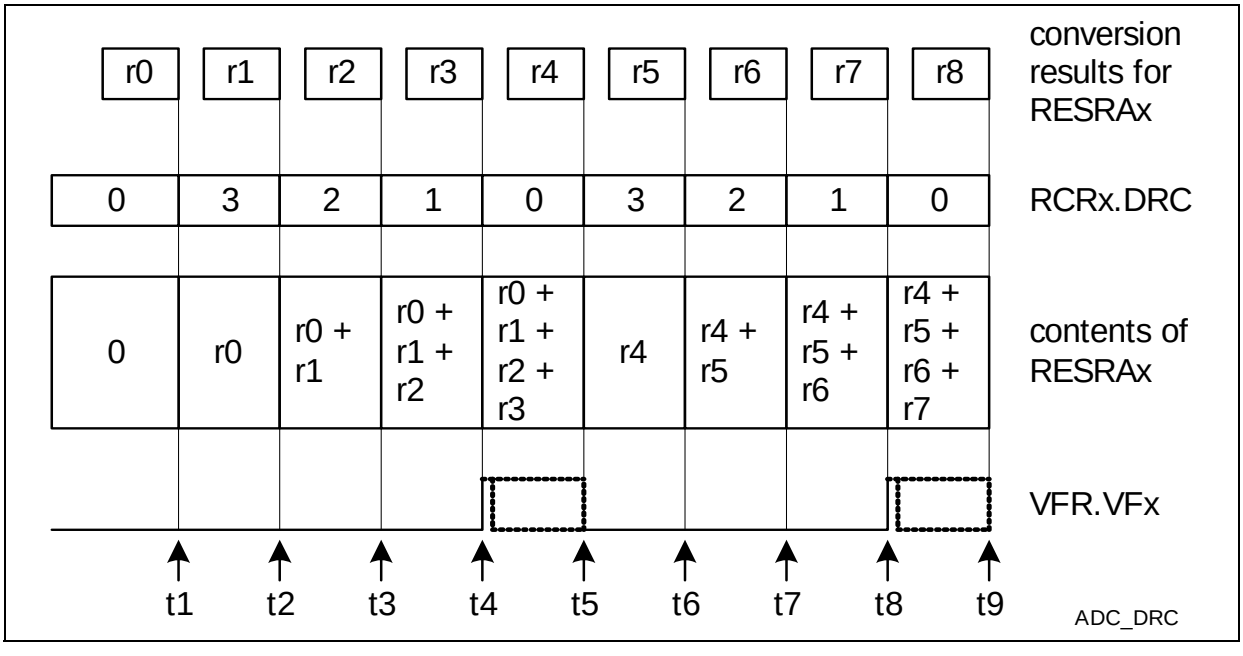


Figure 16-20 Data Reduction Filter

In the example given in **Figure 16-20**, a data reduction sequence of 4 accumulated conversion results is shown. The data reduction is based on three rules:

- Each time bit field DRC is 0 and a conversion targeting result register x is completed (t1, t5, t9), the contents of bit field RCRx.DRCTR is loaded into bit field DRC and the conversion result is stored in result register x.
- Each time bit field DRC is not 0 and a conversion targeting result register x is completed (t2, t3, t4 for the first final result and t6, t7, t8 for the next one), bit field DRC is decremented by 1 and the conversion result is added to the value already stored in result register x.
- Each time bit field DRC is 0 after decrementing or after loading it with RCRx.DRCTR = 0 (t4 for the first final result and t8 for the next one), the valid bit for the result register x becomes set and a result register event occurs.

The final result of a data reduction sequence has to be read out from result register x before the next data reduction sequence starts (interval between t4 and t5, or t8 and t9 respectively). With the read out of the final result from this register, the valid flag is automatically cleared.
If this interval is too short, it is recommended to associate a second result register z to result register x by enabling the result FIFO mechanism for result register x, see **Figure 16-21** (z = x + 1). In this case, result register x is loaded with the final result elaborated by result register z when a data reduction sequence is finished. The final result has to be read out from result register x before the next data reduction sequence is finished (interval between t4 and t8).

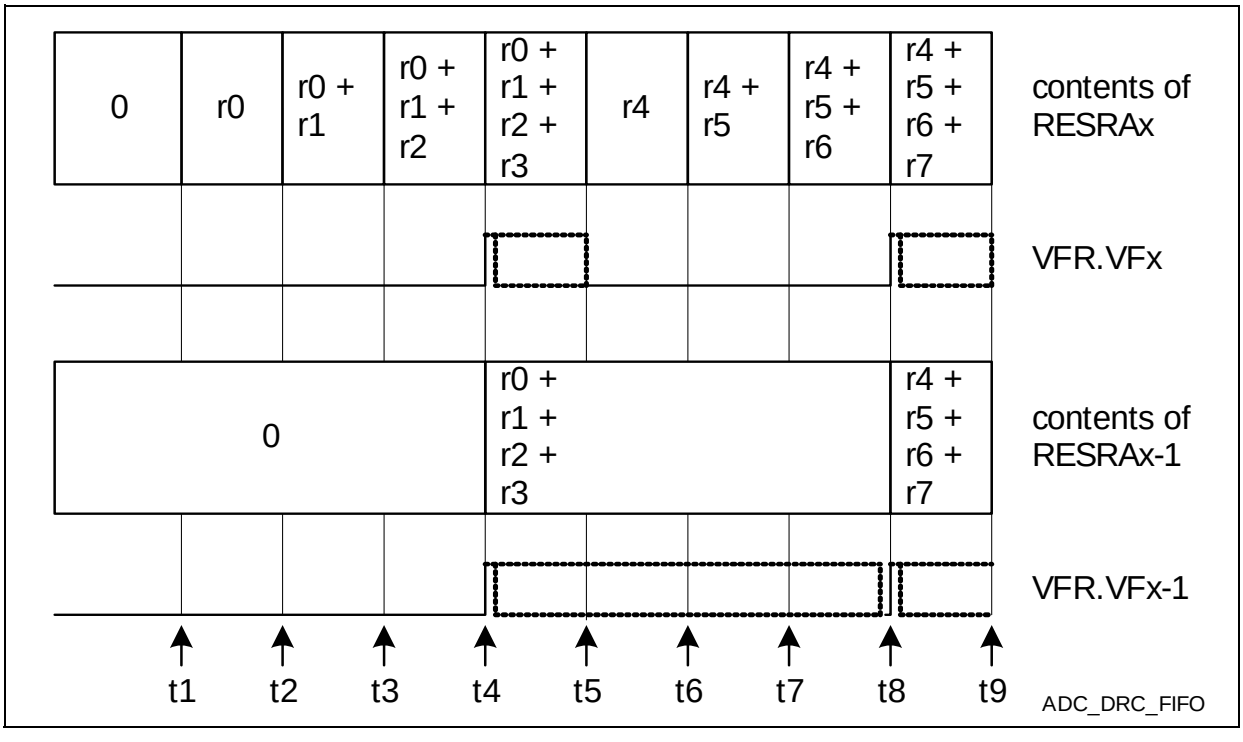


Figure 16-21 Data Reduction Filter with Result FIFO

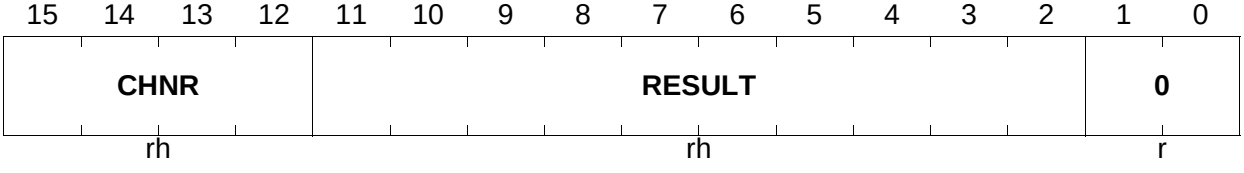
16.2.15 Conversion Result-Related Registers

16.2.15.1 Standard Views RESRx and RESRVx

These result registers deliver the conversion result and the related channel number.

The corresponding valid flag is cleared when register RESRx is read, whereas it is left unchanged when reading RESRVx.

RESRx (x = 0-7)
Result Register x **XSFR(40_H + 2 * x)** **Reset Value: 0000_H**
RESRVx (x = 0-7)
Result Register x, View V **XSFR(60_H + 2 * x)** **Reset Value: 0000_H**



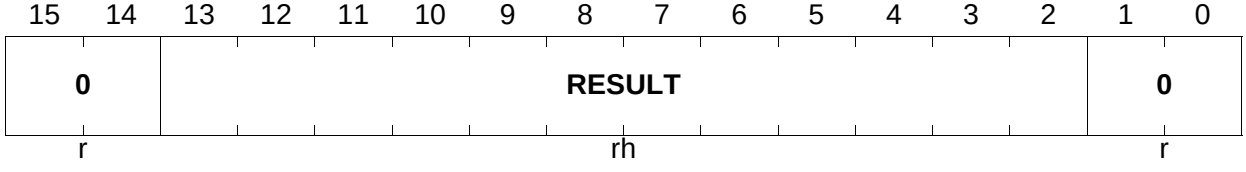
Field	Bits	Type	Description
RESULT	[11:2]	rh	**Conversion Result** This bit field contains the conversion result.
CHNR	[15:12]	rh	**Channel Number** This bit field contains the channel number of the latest register update. In case that the external multiplexer control is enabled, bits CHNR[3:1] are replaced by the multiplexer setting EMUX[2:0].
0	[1:0]	r	**Reserved** returns 0 if read; should be written with 0;

16.2.15.2 Data Reduction Read Views RESRAx and RESRAVx

These result registers deliver the accumulated conversion result and no channel number.

The corresponding valid flag is cleared when register RESRAx is read, whereas it is left unchanged when reading RESRAVx.

RESRAx (x = 0-7)
Result Register x, View A **XSFR(50_H + 2 * x)** **Reset Value: 0000_H**
RESRAVx (x = 0-7)
Result Register x, View AV **XSFR(70_H + 2 * x)** **Reset Value: 0000_H**

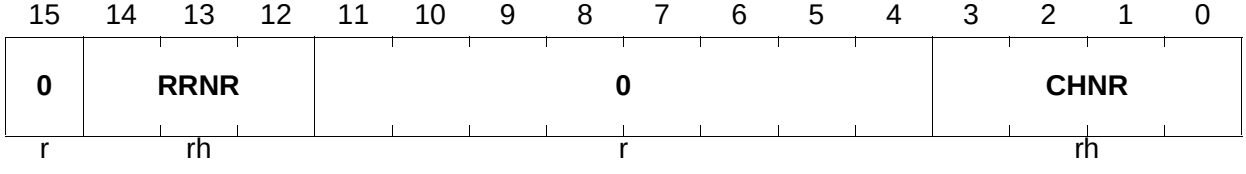


Field	Bits	Type	Description
RESULT	[13:0]	rh	**Conversion Result** This bit field contains the result of the data reduction filter.
0	[1:0], [15:14]	rh	**Reserved** returns 0 if read; should be written with 0;

16.2.15.3 Result Status Shadow Register

The result status shadow register contains the status information related to the latest result register (view without extension “V”) that has been read out. The update of the bit fields is done when a result register is read out.

RSSR
Result Status Shadow Register **XSFR(82_H)** **Reset Value: 0000_H**

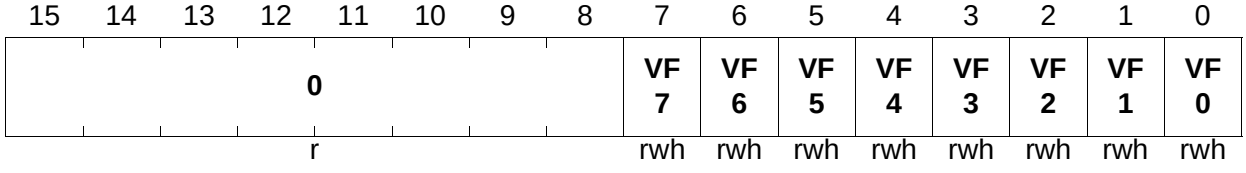


Field	Bits	Type	Description
CHNR	[3:0]	rh	**Channel Number** This bit field indicates the channel number related to the latest result that has been read out.
RRNR	[14:12]	rh	**Result Register Number** This bit field indicates to which result register the information stored in CHNR belongs.
0	[11:4], 15	r	**Reserved** returns 0 if read; should be written with 0;

16.2.15.4 Valid Flag Register

The valid flag register contains the flags indicating that the corresponding result register contents are valid (valid = “new” = not read out).

VFR
Valid Flag Register **XSFR(80_H)** **Reset Value: 0000_H**



Field	Bits	Type	Description
VFx (x = 0 - 7)	x	rwh	**Valid Flag for Result Register x** This bit indicates that the contents of the result register x is valid. Writing a 0 has no effect, whereas writing a 1 clears the written bit position. If a hardware event triggers the setting of a bit VFx and SW writes a 1 to the same bit position, the bit VFx is cleared (software overrules hardware). 0_B The result register x does not contain valid data. Either this register has been read out or no data has been moved to it. 1_B The result register x contains valid data that has not yet been read out.
0	[15:8]	r	**Reserved** returns 0 if read; should be written with 0;

16.2.15.5 Result Control Registers

The result control registers contain bits to control the behavior of the result registers and to monitor their status.

RCRx (x = 0 - 7)
Result Control Register x **XSFR($B0_H$ + x * 2)** **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0			VF	0		DRC		0	WFR	FEN	IEN	0		DRCTR	
r			rh	r		rh		r	rw	rw	rw	r		rw	

Field	Bits	Type	Description
DRCTR	[1:0]	rw	**Data Reduction Control** This bit field defines how many conversion results are accumulated for data reduction (see **Section 16.2.14.5**). It defines the reload value for bit field DRC. 00_B The data reduction filter is disabled. The reload value for DRC is 0, so no accumulation is done. 01_B The data reduction filter is enabled. The reload value for DRC is 1, so the accumulation is done over 2 conversions. 10_B The data reduction filter is enabled. The reload value for DRC is 2, so the accumulation is done over 3 conversions. 11_B The data reduction filter is enabled. The reload value for DRC is 3, so the accumulation is done over 4 conversions.
IEN	4	rw	**Interrupt Enable** This bit enables the result event interrupt if a result event is detected for result register x. 0_B The result event interrupt is disabled. 1_B The result event interrupt is enabled.
FEN	5	rw	**FIFO Enable** This bit enables the FIFO functionality for result register x, see **Section 16.2.14.4**. 0_B The FIFO functionality is disabled. 1_B The FIFO functionality is enabled.

Field	Bits	Type	Description
WFR	6	rw	**Wait-for-Read Mode** This bit enables the wait-for-read mode for result register x. 0_B The wait-for-read mode is disabled. 1_B The wait-for-read mode is enabled.
DRC	[9:8]	rh	**Data Reduction Counter** This bit field indicates how many conversion results have still to be accumulated to generate the final result for data reduction. The valid flag is automatically set and a result event is generated when this bit field becomes 0 (by decrementing or by reload). 00_B The final result is available in the result register. 01_B 1 more conversion result has to be added to obtain the final result in the result register. 10_B 2 more conversion results have to be added to obtain the final result in the result register. 11_B 3 more conversion results have to be added to obtain the final result in the result register.
VF	12	rh	**Valid Flag** This flag indicates that the contents of the result register x is valid. It is another view of the corresponding bit in register VFR. 0_B The result register x does not contain valid data. 1_B The result register x contains valid data.
0	[3:2], 7, [11:10], [15:13]	r	**Reserved** returns 0 if read; should be written with 0;

16.2.15.6 Event Indication Flag Register

The event indication flag register EVINFR monitors the detected request source events (flags EVINF0 - EVINF2) and result events (flags EVINF8 - EVINF15).

EVINFR
Event Indication Flag Register **XSFR($A0_H$)** **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EVINF15	**EVINF14**	**EVINF13**	**EVINF12**	**EVINF11**	**EVINF10**	**EVINF9**	**EVINF8**	**0**					**EVINF2**	**EVINF1**	**EVINF0**
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	r					rwh	rwh	rwh

Field	Bits	Type	Description
EVINFx (x = 0 - 2)	x	rwh	**Event Indication Flag for Request Source x** Flag EVINFx indicates that a request source event of request source x has been detected. Writing a 0 has no effect, whereas writing a 1 sets the written bit position and generates the corresponding interrupt request. 0_B An event of request source x has not yet been detected. 1_B An event of request source x has been detected.
EVINFx (x = 8 - 15)	x	rwh	**Event Indication Flag for Result Register x - 8** Flag EVINFx indicates that a result event of result register x-8 has been detected. Writing a 0 has no effect, whereas writing a 1 sets the written bit position and generates the corresponding interrupt request. 0_B An event of result register x-8 has not yet been detected. 1_B An event of result register x-8 has been detected.
0	[7:3]	r	**Reserved** returns 0 if read; should be written with 0;

16.2.15.7 Clear Event Indication Register

Writing a 1 to a bit position in the event indication clear register EVINCR clears the corresponding event indication flag EVINFx in register **EVINFR**. If a request source or result event is detected when the corresponding bit position is written with a 1, flag EVINFx is cleared.

EVINCR
Event Indication Clear Register **XSFR($A2_H$)** **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EVINC15	**EVINC14**	**EVINC13**	**EVINC12**	**EVINC11**	**EVINC10**	**EVINC9**	**EVINC8**	**0**					**EVINC2**	**EVINC1**	**EVINC0**
w	w	w	w	w	w	w	w	r					w	w	w

Field	Bits	Type	Description
EVINCx (x = 0 - 2)	x	w	**Clear Event Indication Flag for Request Source x** 0 No action. 1 Bit EVINFR.x is cleared.
EVINCx (x = 8 - 15)	x	w	**Clear Event Indication Flag for Result Reg. x-8** 0 No action. 1 Bit EVINFR.x is cleared.
0	[7:3]	r	**Reserved** returns 0 if read; should be written with 0;

16.2.15.8 Event Interrupt Node Pointer Registers

The bit fields in these registers define the service request output SR[3:0] is used to signal a request source or result event interrupt.

EVINPR0
Event Interrupt Node Pointer Register 0

XSFR($A8_H$) **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0						EVINP2		0		EVINP1		0		EVINP0	
r						rw		r		rw		r		rw	

Field	Bits	Type	Description
EVINP0, EVINP1, EVINP2	[1:0], [5:4], [9:8]	rw	**Interrupt Node Pointer for Request Source x** This bit field selects which service request output indicates an event interrupt of request source x. 00_B Service request output SR0 is selected. 01_B Service request output SR1 is selected. 10_B Service request output SR2 is selected. 11_B Service request output SR3 is selected.
0	[3:2], [7:6], [15:10]	r	**Reserved** returns 0 if read; should be written with 0;

EVINPR8
Event Interrupt Node Pointer Register 8

XSFR(AC_H) **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		EVINP11		0		EVINP10		0		EVINP9		0		EVINP8	
r		rw		r		rw		r		rw		r		rw	

Field	Bits	Type	Description
EVINP8, EVINP9, EVINP10, EVINP11	[1:0], [5:4], [9:8], [13:12]	rw	**Interrupt Node Pointer for Result Event x-8** This bit field selects which service request output indicates an event interrupt of result register x-8. 00_B Service request output SR0 is selected. 01_B Service request output SR1 is selected. 10_B Service request output SR2 is selected. 11_B Service request output SR3 is selected.
0	[3:2], [7:6], [11:10], [15:14]	r	**Reserved** returns 0 if read; should be written with 0;

EVINPR12
Event Interrupt Node Pointer Register 12

XSFR(AE_H) **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		**EVINP15**		**0**		**EVINP14**		**0**		**EVINP13**		**0**		**EVINP12**	
r		rw		r		rw		r		rw		r		rw	

Field	Bits	Type	Description
EVINP12, EVINP13, EVINP14, EVINP15	[1:0], [5:4], [9:8], [13:12]	rw	**Interrupt Node Pointer for Result Event x-8** This bit field selects which service request output indicates an event interrupt of result register x-8. 00_B Service request output SR0 is selected. 01_B Service request output SR1 is selected. 10_B Service request output SR2 is selected. 11_B Service request output SR3 is selected.
0	[3:2], [7:6], [11:10], [15:14]	r	**Reserved** returns 0 if read; should be written with 0;

16.2.16 External Multiplexer Control

If an application requires more analog inputs channels than available on the XC2000, the ADC kernel supports an extension of analog channels by adding an external analog multiplexer. Three output signals EMUX[2:0] are delivered by each ADC kernel to control the settings of an external analog multiplexer. They can be used to extend the number of analog input channels by adding an external 1-out-of-8 multiplexer.
The external multiplexer control behavior is defined by the bits in register **EMCTR**.
The current setting of EMUX[2:0] is given by bit field LASTEMUX. If another extended input channel should be converted, bit field SETEMUX has to be programmed to the desired value. This value is automatically applied with the start of the next conversion of the related analog ADC input channel.

In the example shown in **Figure 16-22** and in the description below, the analog input CH7 has been extended, leading to additional analog inputs named CH70 to CH77. In general, the external multiplexer support can be enabled for each of the input channels CH0 to CH7.

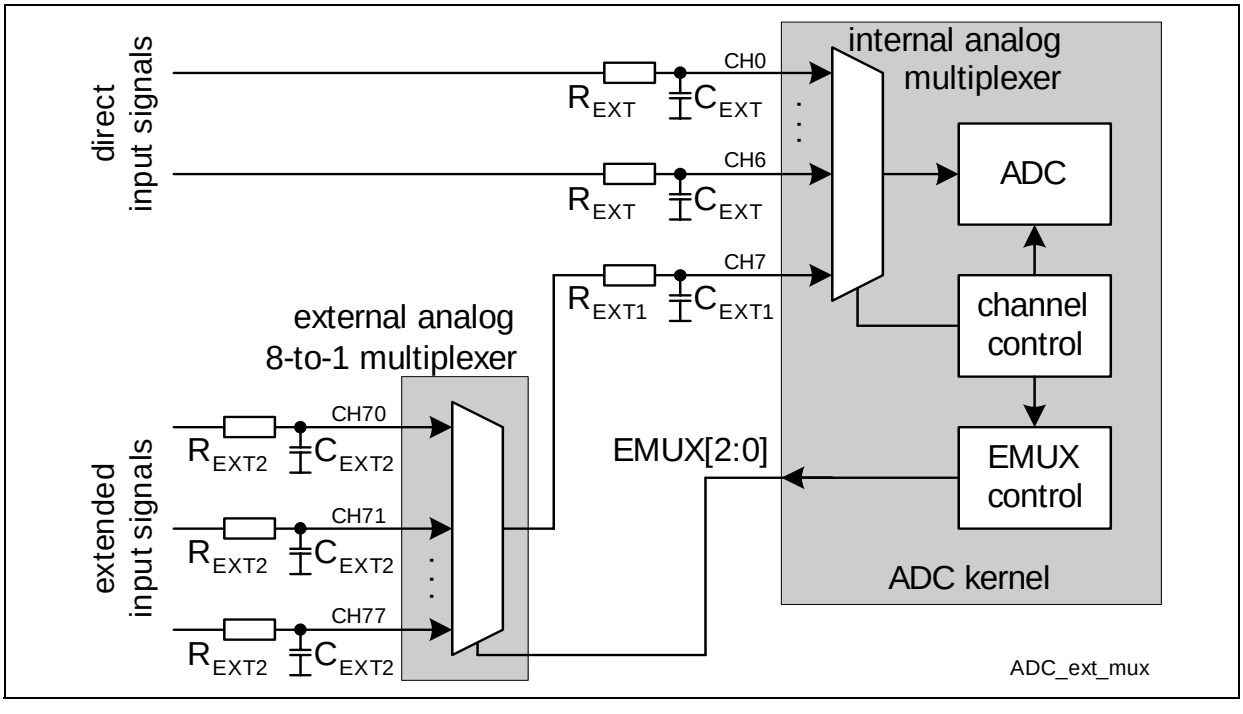


Figure 16-22 External Analog Multiplexer

If the external multiplexer is located far from the ADC analog input, it is recommended to introduce an RC filter R_{EXT1}-C_{EXT1} directly at the analog input CH7 of the ADC. If needed for signal filtering, local RC filters R_{EXT2}-C_{EXT2} can be optionally added at the inputs of the external analog multiplexer.
If the external multiplexer is located close to the analog ADC input, the components R_{EXT1} and C_{EXT1} are not necessarily needed. In this case it is strongly recommended to

introduce RC filters (R_{EXT2}, C_{EXT2}) at the multiplexer inputs.
Please note that each RC filter limits the bandwidth of the analog input signal.

The RC filters used with an external multiplexer may lead to another impedance “seen” by the ADC analog input CH7 than for the other (direct) analog inputs. The adaptation of the sample phase length can be done by using a different input class with a different value for the sample phase extension. This value can be adapted to execute conversions with an EMUX[2:0] setting that has changed a sufficiently long time before the conversion of CH7 starts. “A sufficiently long time before” signifies that signal transitions at the analog ADC input due to changing multiplexer setting are finished and the input signal is stable enough.
After changing the EMUX[2:0] setting of the external multiplexer, an additional settling time has to elapse before the switched analog signal is stable and can be measured. To compensate for this settling time, an alternative sample phase length (instead of the one given by the input class) is automatically applied for the first conversion of CH7 after EMUX[2:0] has changed. The alternative sample phase length can be programmed by bit field **EMCTR**.EMSAMPLE. If the first conversion of CH7 after the EMUX[2:0] setting has changed is aborted due to a higher priority request, the repeated conversion of CH7 also uses the value of EMSAMPLE. The settling time is considered to be finished after the complete conversion of CH7.

The automatic control of an external analog multiplexer related to conversions of an ADC input channel can be enabled by bits EMSELx in register **EMENR**. If bit EMSELx = 0, channel CHx does not influence any EMUX setting and the selected input class is always taken into account.
If bit EMSELx = 1, the automatic multiplexer control is enabled for channel CHx. With each conversion start of CHx, the value of **EMCTR**.SETEMUX is copied to **EMCTR**.LASTEMUX that is output at EMUX[2:0]. In addition, the control logic checks if the value of LASTEMUX has changed to adapt the sample phase length accordingly.

16.2.17 Synchronized Conversions for Parallel Sampling

The independent ADC kernels implemented in the XC2000 can be synchronized for simultaneous measurements of analog input channels. While no parallel conversion is requested, the kernels can work independently.
The synchronization mechanism for parallel conversions ensures that the sample phases of the related channel(s) start simultaneously. Different values for the resolution and the sample phase length of each kernel for a parallel conversion are supported.
A parallel conversion can be requested individually for each input channel (also several channels can be enabled for parallel conversions). In the example shown in the figure below, input channels CH3 of the ADC kernels ADC0 and ADC1 are converted synchronously, whereas other input channels do not lead to parallel conversions.

This leads to the following structure:

- A **synchronization master** ADC kernel can request a conversion of an analog channel. If this channel is selected for a synchronized conversion, it is also requested in the connected slave ADC kernel(s).
- A **synchronization slave** ADC kernel reacts to incoming synchronized conversion requests from its master. While no incoming master requests are active, the slave kernel can convert its own requests.
- All ADC kernels in an ADC module being similar, each kernel can be set up to be a synchronization master or a synchronization slave (depending on the application needs, such as trigger capability of request sources).
- A synchronization master can synchronize several slave kernels, whereas a slave kernel can only be synchronized to one master kernel.

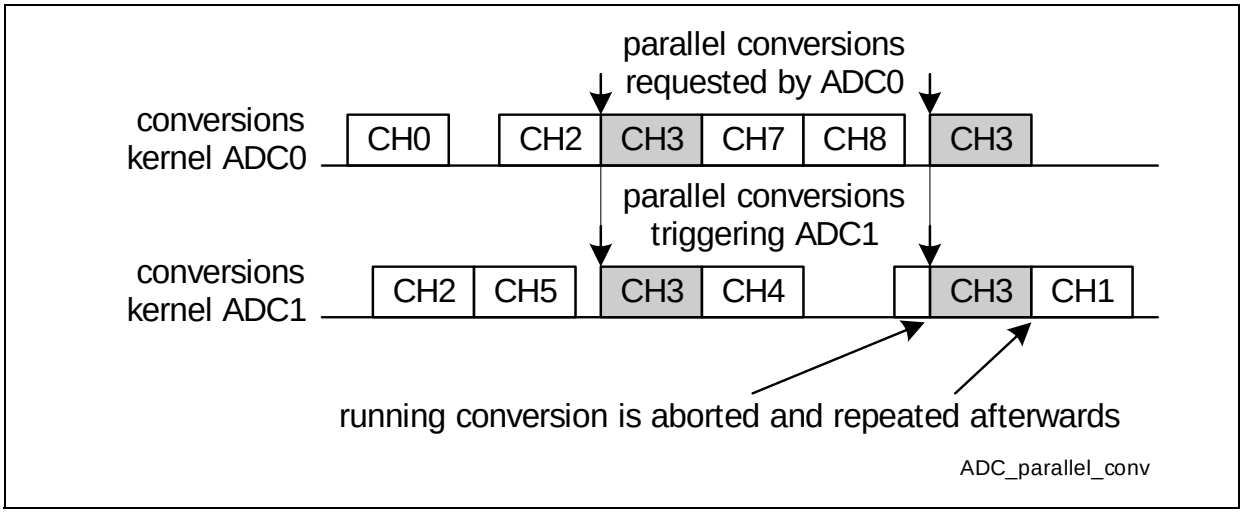


Figure 16-23 Parallel Conversions

The term “conversion group” has been introduced to define the kernel behavior allowing parallel sampling:

- Kernels in the same conversion group can execute parallel conversions.
- A conversion group contains at least 1 ADC kernel and can contain a maximum of all ADC kernels of the ADC module.
- Each conversion group contains exactly one synchronization master kernel that issues a parallel conversion request and defines the internal frequencies f_{ADCI} and f_{ADCD} and the channel number for a parallel conversion of the conversion group.
- All other kernels in a conversion group are synchronization slaves and have to be programmed with the same values of **GLOBCTR**.DIVA, DIVD and ARBRND as the synchronization master.
- If there is no need for parallel conversions, each kernel can be considered to form an own conversion group with only an ADC kernel as synchronization master, but without any synchronization slave.
- The channel number and the synchronization request are issued by the synchronization master to the kernels in the same synchronization group if a conversion is requested with **CHCTRx (x = 0 - 15)**.SYNC = 1 in the synchronization master kernel. Synchronization slaves can not issue synchronization requests.
- Once started, a parallel conversion can not be aborted.
- A parallel conversion request is always handled with highest priority and cancel-inject-repeat mode in a synchronization slave (see **Section 16.2.6.2**).
- Bit **GLOBCTR**.ARBM has to be 0 for synchronization slaves.
- The wait-for-read mode is supported for the master kernel, whereas the setting is ignored in the slave kernels (previous results may be overwritten).

The synchronization request issuing mechanism of the master to the slave kernels is based on bit field **GLOBSTR**.ANON. The information given by **GLOBCTR**.ANON is distributed by the synchronization master to all kernels in the conversion group (the bit fields **SYNCTR**.STSEL of all kernels must be programmed in a way that all kernels refer to the same information). In addition to the ANON information, the master delivers the requested channel number to the slave (not explicitly shown in **Figure 16-24**).
The start of the converters of all kernels of a conversion group is based on signals indicating when a kernel is ready and can start the sample phase of a parallel conversion. Bit **SYNCTR**.EVALR1 defines if a kernel has to wait for the other kernel(s) (to allow parallel conversions) or can start without waiting (no parallel conversions possible). To support parallel conversions, all ready signals of the kernels of a conversion group have to be considered.

The alias feature is independent of synchronized conversions. All kernels of a conversion group request the same channel number (defined by the master), but can convert analog signals from different inputs. The requested channel number can be redirected by its alias setting. E.g., if the channel number requested in a conversion group is channel CH0, but for a kernel, an alternative reference is connected to this input,

the actually converted analog input can be changed to any value. This can be done by programming bit field ALIAS0 accordingly.

Note: A parallel conversion in a slave ADC should not target a result register that is already used for data reduction of other channels.

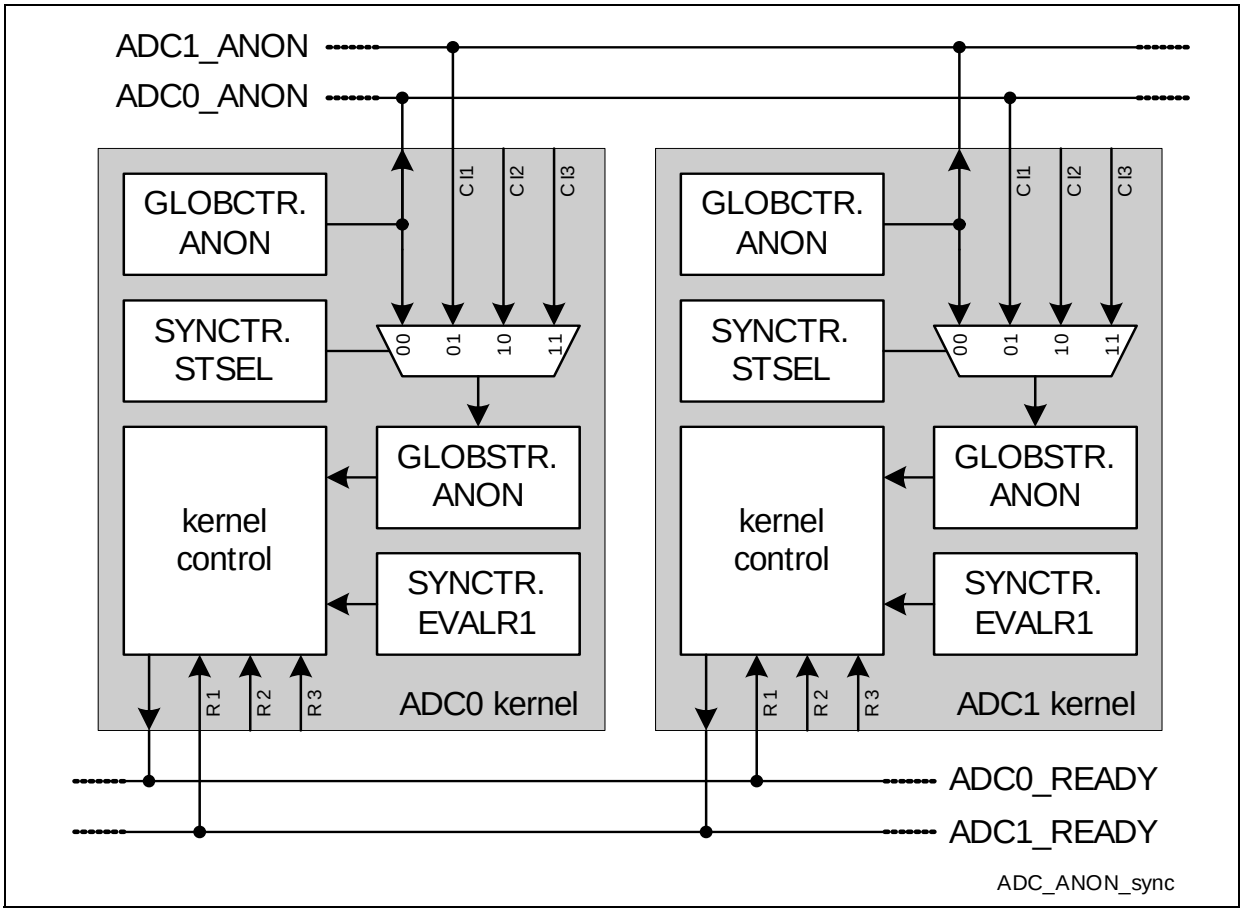


Figure 16-24 Synchronization via ANON and Ready Signals

Table 16-4 SYNCTR Setting for Kernel Synchronization

Operating Mode	SYNCTR.EVALR1	SYNCTR.STSEL
ADC0 Kernel (values to be programmed to ADC0_SYNCTR)		
no sync	0_B	00_B
master of ADC1	1_B	00_B
slave of ADC1	1_B	01_B
ADC1 Kernel (values to be programmed to ADC1_SYNCTR)		
no sync	0_B	00_B

Table 16-4 SYNCTR Setting for Kernel Synchronization (cont'd)

Operating Mode	SYNCTR.EVALR1	SYNCTR.STSEL
master of ADC0	1_B	00_B
slave of ADC0	1_B	01_B

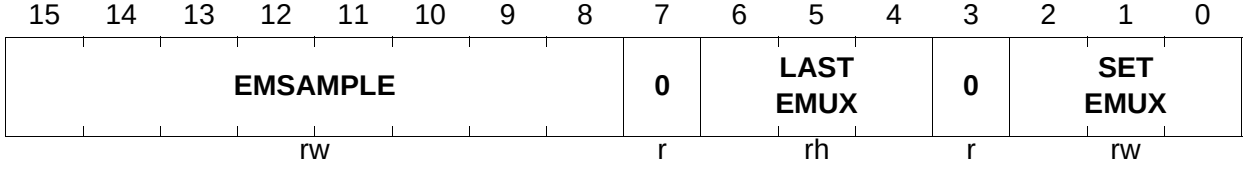
16.2.18 Additional Feature Registers

16.2.18.1 External Multiplexer Control Register

The external multiplexer control register defines the settings of an external analog multiplexer.

EMCTR
External Multiplexer Control Register

XSFR($D0_H$) **Reset Value: 0000_H**



Field	Bits	Type	Description
SETEMUX	[2:0]	rw	**New Setting for External Multiplexer** This bit field defines the EMUX[2:0] setting for the external multiplexer that will be selected for the next conversion of the enabled channel.
LASTEMUX	[6:4]	rh	**Last Setting of External Multiplexer** This bit field indicates the current EMUX[2:0] setting of the external multiplexer. This value has been selected for (at least) the last conversion of the enabled channel.
EMSAMPLE	[15:8]	rw	**External Multiplexer Sampling Time** This bit field defines the alternative sample phase length in the case the external multiplexer setting has changed with the start of a conversion with enabled external multiplexer (the value given by the selected input class is not taken into account). A minimum sample phase of 2 analog clock cycles is extended by the programmed value. sample phase length = (2 + EMSAMPLE) / f_{ADCI}
0	3, 7	r	**Reserved** returns 0 if read; should be written with 0;

16.2.18.2 External Multiplexer Enable Register

The external multiplexer enable register defines if a channel is enabled to control the settings of an external analog multiplexer. It is not allowed to enable the external multiplexer control for more than one analog input channel.

EMENR
External Multiplexer Enable Register

XSFR($D6_H$) **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**EMSEL7**	**0**	**EMSEL6**	**0**	**EMSEL5**	**0**	**EMSEL4**	**0**	**EMSEL3**	**0**	**EMSEL2**	**0**	**EMSEL1**	**0**	**EMSEL0**
r	rw	r	rw	r	rw	r	rw	r	rw	r	rw	r	rw	r	rw

Field	Bits	Type	Description
EMSELx (x = 0 - 7)	2*x	rw	**External Multiplexer Selection for Channel CHx** This bit defines if an external multiplexer will be controlled by channel CHx. 0_B The external multiplexer control is disabled. 1_B The external multiplexer control is enabled for CHx.
0	1, 3, 5, 7, 9, 11, 13, 15	r	**Reserved** returns 0 if read; should be written with 0;

16.2.18.3 Synchronization Control Register

The synchronization control register contains bits controlling the synchronization between several kernels for parallel conversions. The programming of register SYNCTR in the kernels of a conversion group has to be done while the bit fields **GLOBSTR**.ANON = 00_B in all ADC kernels of the conversion group. Bit field ANON of the synchronization master can be set to 11_B afterwards. It is recommended to avoid power saving modes (ANON = 01_B or 10_B) for parallel conversions.
The bits EVALRx are only taken into account if a synchronized, parallel conversion is requested by a master. This ensures that the conversions of the ADC kernels of the same synchronization group are started at the same time for parallel sampling (although a kernel might be idle, the master and all its connected slaves have to wait for all of them being ready).

SYNCTR
Synchronization Control Register XSFR($1A_H$) **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0									0		EVALR1	0		STSEL	
r									rw		rw	r		rw	

Field	Bits	Type	Description
STSEL	[1:0]	rw	**Start Selection** This bit field controls the synchronization mechanism of the ADC kernel. 00_B The kernel is a synchronization master. The kernel's own bit field GLOBCTR.ANON is taken into account. 01_B The kernel is a synchronization slave. The control information at input CI1 is taken into account instead. 10_B reserved, do not use (kernel is switched off) 11_B reserved, do not use (kernel is switched off)
EVALR1	4	rw	**Evaluate Ready Input R1** This bit defines if a kernel is considered to be part of a synchronization group. Parallel conversions can only be started if the synchronization master and all slaves of the conversion group indicate that they are ready to start a parallel conversion. 0_B The ready input R1 is not considered for the start of a parallel conversion of this conversion group. 1_B The ready input R1 is considered for the start of a parallel conversion of this conversion group.
0	[6:5]	r	**Reserved for Future Use** returns 0 if read; must be written with 0;
0	[3:2], [15:7]	r	**Reserved** returns 0 if read; should be written with 0;

16.3 Implementation

This section describes the implementation of the ADC kernels in the XC2000 device.

- Address map (see **Section 16.3.1**)
- Interrupt control registers (see **Section 16.3.2**)
- Analog connections of ADC0 (see **Section 16.3.3.1**)
- Analog connections of ADC1 (see **Section 16.3.3.2**)
- Digital connections of ADC0 (see **Section 16.3.4.1**)
- Digital connections of ADC1 (see **Section 16.3.4.2**)

16.3.1 Address Map

The ADC kernels ADC0 and ADC1 are available at the following base addresses.
The exact register address is given by the relative address of the register (given in **Table 16-2**) plus the kernel base address (given in **Table 16-5**) of the module.

Table 16-5 Registers Address Space

Module	Base Address	End Address	Note
ADC0	$E000_H$	$E0FF_H$	
ADC1	$E100_H$	$E1FF_H$	

16.3.2 Interrupt Control Registers

The interrupt control registers are located in the SFR area. They are described in the general interrupt chapter.

Table 16-6 ADC Interrupt Control Registers

Short Name	Description
ADC_0IC	Interrupt Control Register for SR0 of ADC0
ADC_1IC	Interrupt Control Register for SR1 of ADC0
ADC_2IC	Interrupt Control Register for SR2 of ADC0
ADC_3IC	Interrupt Control Register for SR3 of ADC0
ADC_4IC	Interrupt Control Register for SR0 of ADC1
ADC_5IC	Interrupt Control Register for SR1 of ADC1
ADC_6IC	Interrupt Control Register for SR2 of ADC1
ADC_7IC	Interrupt Control Register for SR3 of ADC1

16.3.3 Analog Connections

The input channels of both ADC kernels are distributed as follows:

- 16 channels of ADC0 connected to P5
- 8 channels of ADC1 connected to P15
- 4 channels of ADC1 connected to P5 and overlaid to ADC0 inputs

Each ADC kernel has its own reference input lines VAGND and VAREFx, with x = 0, 1. Depending on the package, these lines can be available as independent pins for high pin count packages or can be combined for low pin count packages.
The respective voltage supply lines of both converters are connected together.

16.3.3.1 Analog Connections of ADC0

The table below lists the analog connections of ADC0.

Table 16-7 ADC0 Analog Connections in XC2000

Signal	from/to Module	I/O to ADC0	Can be used to/as
Power supply and standard reference			
VDDPA	see pinning chapter	I	analog power supply
VSS		I	analog power supply
VAREF(0)		I	positive analog reference
VAGND		I	negative analog reference
Analog input channels			
CH0	P5.0	I	analog input channel 0
CH1	P5.1	I	analog input channel 1
CH2	P5.2	I	analog input channel 2
CH3	P5.3	I	analog input channel 3
CH4	P5.4	I	analog input channel 4
CH5	P5.5	I	analog input channel 5
CH6	P5.6	I	analog input channel 6
CH7	P5.7	I	analog input channel 7
CH8	P5.8	I	analog input channel 8 overlaid with ADC1 channel 8
CH9	P5.9	I	analog input channel 9 overlaid with ADC1 channel 9

Table 16-7 ADC0 Analog Connections in XC2000 (cont'd)

Signal	from/to Module	I/O to ADC0	Can be used to/as
CH10	P5.10	I	analog input channel 10 overlaid with ADC1 channel 10
CH11	P5.11	I	analog input channel 11 overlaid with ADC1 channel 11
CH12	P5.12	I	analog input channel 12
CH13	P5.13	I	analog input channel 13
CH14	P5.14	I	analog input channel 14
CH15	P5.15	I	analog input channel 15

16.3.3.2 Analog Connections of ADC1

The table below lists the analog connections of ADC1.

Table 16-8 ADC1 Analog Connections in XC2000

Signal	from/to Module	I/O to ADC1 analog	Can be used to/as
Power supply and standard reference			
VDDPA	see pinning chapter	I	analog power supply
VSS		I	
VAREF(1)		I	positive analog reference
VAGND		I	negative analog reference
Analog input channels			
CH0	P15.0	I	analog input channel 0
CH1	P15.1	I	analog input channel 1
CH2	P15.2	I	analog input channel 2
CH3	P15.3	I	analog input channel 3
CH4	P15.4	I	analog input channel 4
CH5	P15.5	I	analog input channel 5
CH6	P15.6	I	analog input channel 6
CH7	P15.7	I	analog input channel 7

Table 16-8 ADC1 Analog Connections in XC2000 (cont'd)

Signal	from/to Module	I/O to ADC1 analog	Can be used to/as
CH8	P5.8	I	analog input channel 8 overlaid with ADC0 channel 8
CH9	P5.9	I	analog input channel 9 overlaid with ADC0 channel 9
CH10	P5.10	I	analog input channel 10 overlaid with ADC0 channel 10
CH11	P5.11	I	analog input channel 11 overlaid with ADC0 channel 11
CH12..15	n.c.	I	not available, do not request

16.3.4 Digital Connections

The following table shows the digital connections of the ADC kernels with other modules or pins in the XC2000 device.

The following sections refer to the inter-module connections, whereas the connections of the service request outputs SR[3:0] of each kernel to the interrupt control registers is given in **Section 16.3.2**.

Note: The functional inputs of the ADC that are marked I(s) are additionally synchronized to f_{SYS} before they can affect the module internal logic. The resulting delay of $2/f_{SYS}$ and an uncertainty of $1/f_{SYS}$ have to be taken into account for precise timing calculation. An edge of an input signal can only be correctly detected if the high phase and the low phase of the input signal are both longer than $1/f_{SYS}$.
The functional inputs of the ADC that are marked I are already considered as synchronous to f_{SYS}.

16.3.4.1 Digital Connections of ADC0

The table below lists the digital connections of ADC0.

Table 16-9 ADC0 Digital Connections in XC2000

Signal	from/to Module	I/O to ADC0	Can be used to/as
External multiplexer control			
EMUX[0]	P6.0	O	control of external analog multiplexer(s)
EMUX[1]	P6.1	O	control of external analog multiplexer(s)
EMUX[2]	P6.2	O	control of external analog multiplexer(s)
Request source 0			
REQGT0A	ERU_GOUT0	I	source gate input A
REQGT0B	ERU_GOUT1	I	source gate input B
REQGT0C	P6.0	I (s)	source gate input C
REQGT0D	0	I (s)	source gate input D
REQTR0A	ERU_TOUT1	I	source trigger input A
REQTR0B	CCU60_T13_PM	I	source trigger input B
REQTR0C	P6.1	I (s)	source trigger input C
REQTR0D	P6.3	I (s)	source trigger input D
Request source 1			
REQGT1A	ERU_GOUT0	I	source gate input A

Table 16-9 ADC0 Digital Connections in XC2000 (cont'd)

Signal	from/to Module	I/O to ADC0	Can be used to/as
REQGT1B	ERU_GOUT1	I	source gate input B
REQGT1C	P6.0	I (s)	source gate input C
REQGT1D	0	I (s)	source gate input D
REQTR1A	ERU_TOUT1	I	source trigger input A
REQTR1B	CCU60_T13_PM	I	source trigger input B
REQTR1C	P6.1	I (s)	source trigger input C
REQTR1D	P6.3	I (s)	source trigger input D
Request source 2			
REQGT2A	ERU_GOUT0	I	source gate input A
REQGT2B	ERU_GOUT1	I	source gate input B
REQGT2C	P6.0	I (s)	source gate input C
REQGT2D	0	I (s)	source gate input D
REQTR2A	ERU_TOUT1	I	source trigger input A
REQTR2B	CCU62_T13_PM	I	source trigger input B
REQTR2C	P6.1	I (s)	source trigger input C
REQTR2D	P6.3	I (s)	source trigger input D

16.3.4.2 Digital Connections of ADC1

The table below lists the digital connections of ADC1.

Table 16-10 ADC1 Digital Connections in XC2000

Signal	from/to Module	I/O to ADC1	Can be used to/as
External multiplexer control			
EMUX[0]	P7.2	O	control of external analog multiplexer(s)
EMUX[1]	P7.3	O	control of external analog multiplexer(s)
EMUX[2]	P7.4	O	control of external analog multiplexer(s)
Request source 0			
REQGT0A	ERU_GOUT0	I	source gate input A
REQGT0B	ERU_GOUT1	I	source gate input B
REQGT0C	P6.0	I (s)	source gate input C
REQGT0D	0	I (s)	source gate input D
REQTR0A	ERU_TOUT1	I	source trigger input A
REQTR0B	CCU61_T13_PM	I	source trigger input B
REQTR0C	P6.1	I (s)	source trigger input C
REQTR0D	P6.3	I (s)	source trigger input D
Request source 1			
REQGT1A	ERU_GOUT0	I	source gate input A
REQGT1B	ERU_GOUT1	I	source gate input B
REQGT1C	P6.0	I (s)	source gate input C
REQGT1D	0	I (s)	source gate input D
REQTR1A	ERU_TOUT1	I	source trigger input A
REQTR1B	CCU61_T13_PM	I	source trigger input B
REQTR1C	P6.1	I (s)	source trigger input C
REQTR1D	P6.3	I (s)	source trigger input D
Request source 2			
REQGT2A	ERU_GOUT0	I	source gate input A
REQGT2B	ERU_GOUT1	I	source gate input B
REQGT2C	P6.0	I (s)	source gate input C
REQGT2D	0	I (s)	source gate input D

Table 16-10 ADC1 Digital Connections in XC2000 (cont'd)

Signal	from/to Module	I/O to ADC1	Can be used to/as
REQTR2A	ERU_TOUT1	I	source trigger input A
REQTR2B	CCU63_T13_PM	I	source trigger input B
REQTR2C	P6.1	I (s)	source trigger input C
REQTR2D	P6.3	I (s)	source trigger input D

17 Capture/Compare Unit 2

The XC2000 provides a Capture/Compare (CAPCOM2) unit which provides 16 capture/compare channels, which interact with 2 timers. A CAPCOM2 channel can **capture** the contents of a timer on specific internal or external events, or it can **compare** a timer's contents with given values, and modify output signals in case of a match.

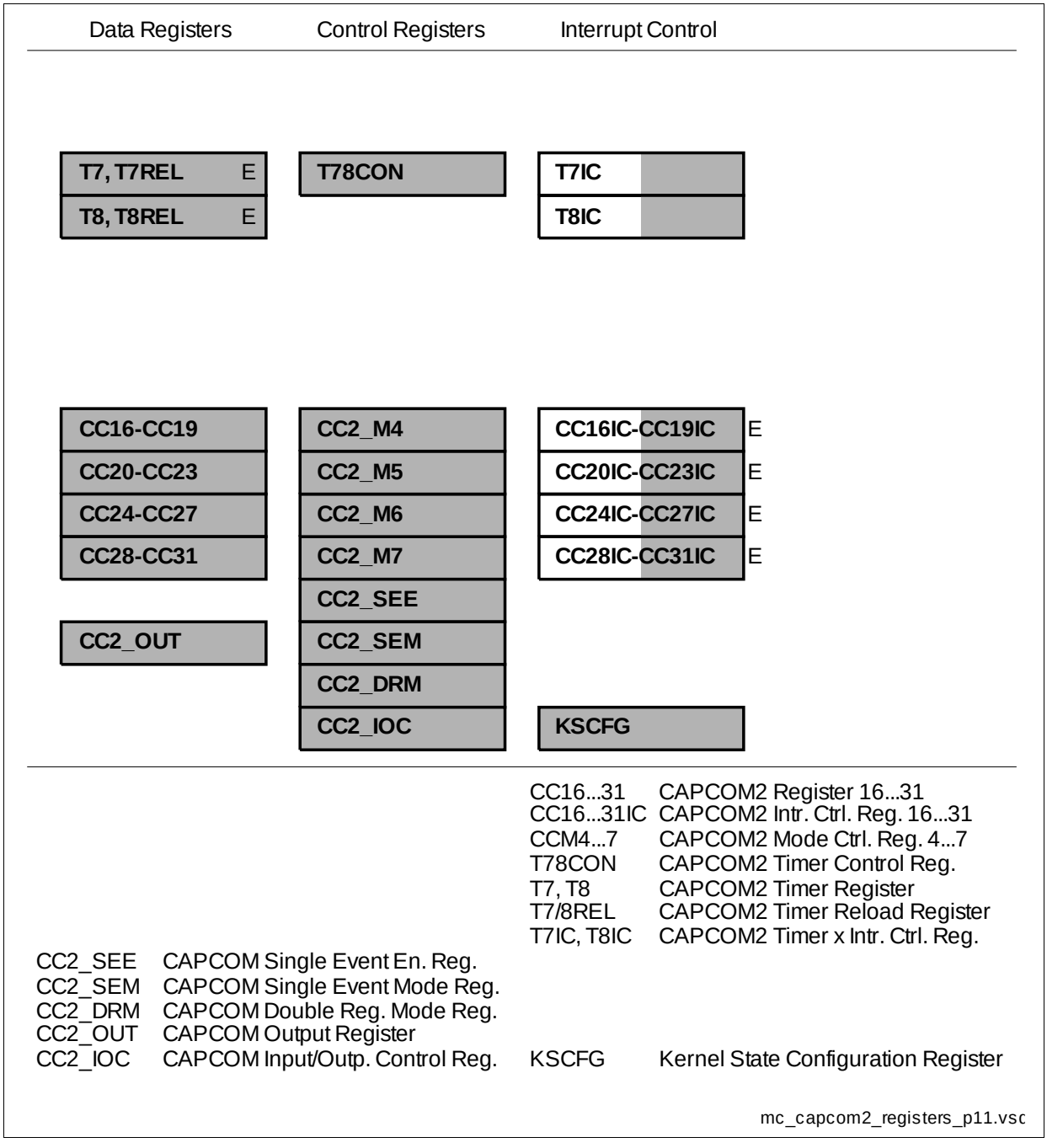


Figure 17-1 SFRs Associated with the CAPCOM Units

With this mechanism, the CAPCOM2 unit supports generation and control of timing sequences on up to 16 channels with a minimum of software intervention.

From the programmer's point of view, the term 'CAPCOM2 unit' refers to a set of registers which are associated with this peripheral, including the port pins which may be used for alternate input/output functions, and their direction control bits (see also **Figure 17-1**).

A CAPCOM2 unit is typically used to handle high speed IO tasks such as pulse and waveform generation, pulse width modulation, or recording of the time when a specific event occurs. It also supports the implementation of up to 16 software-controlled interrupt events.

The CAPCOM2 Unit consists of two 16-bit timers (T7, T8), each with its own reload register (TxREL), and a bank of sixteen dual-purpose 16-bit capture/compare registers (CCy).

The input clock for the CAPCOM2 timers is programmable to several prescaled values of the module input clock (f_{CC}), or it can be derived from the overflow/underflow of timer T6. T7 may also operate in counter mode (from an external input), clocked by external events.

Each capture/compare register may be programmed individually for capture or compare operation, and each register may be allocated to either of the two timers. Each capture/compare register has one signal associated with it, which serves as an input signal for the capture operation or as an output signal for the compare operation.

The capture operation causes the current timer contents to be latched into the respective capture/compare register, triggered by an event (transition) on the associated input signal. This event also activates the associated interrupt request line.

The compare operation may cause an output signal transition on the associated output signal, when the allocated timer increments to the value stored in a capture/compare register. The compare match event also activates the associated interrupt request line. In Double-register compare mode a pair of registers controls one common output signal.

The compare output signals are available via a dedicated output register, and may also control the output latches of the connected port pins. The output path can be selected.

For the switching of the output signals two timing schemes (see **Section 17.8**) can be selected:

In **Staggered Mode**[1)] the output signals are switched consecutively in 8 steps, which distributes the switching steps over a certain time. In staggered mode, the maximum resolution is 8 t_{CC}.

In **Non-Staggered Mode** the output signals are switched immediately at the same time. In non-staggered mode, the maximum resolution is 1 t_{CC}.

1) Staggered mode is compatible with the CAPCOM units of previous 16-bit controllers.

Figure 17-2 shows the basic structure of a CAPCOM unit.

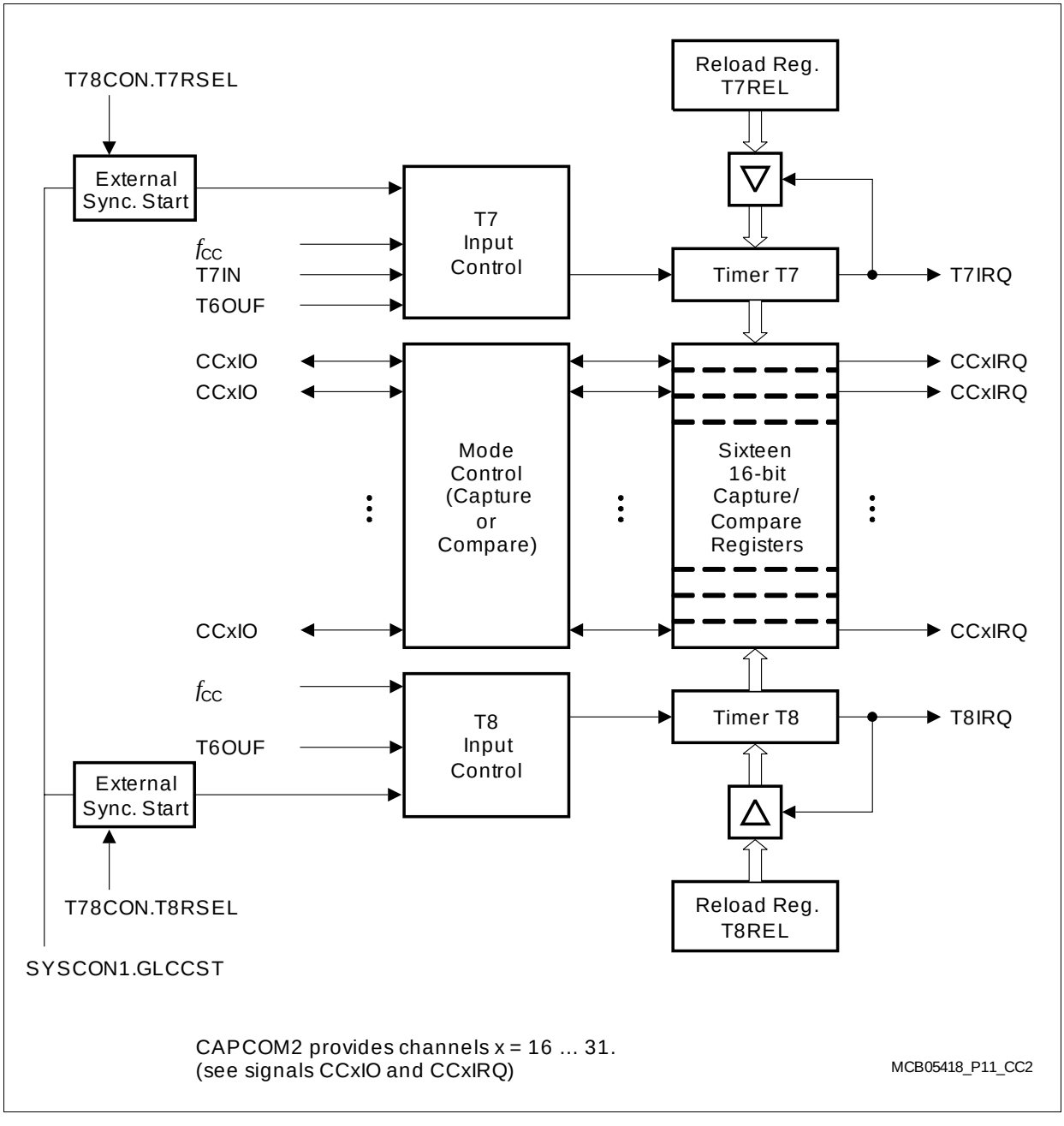


Figure 17-2 CAPCOM Unit Block Diagram

There is a possibility to start both timers T7 and T8 synchronously with the CAPCOM6 timers, by seting the bit in the SYSCON1.GLCCST module.

17.1 The CAPCOM2 Timers

The primary use of the timers T7 and T8 is to provide two independent time bases for the capture/compare channels of each unit. The maximum resolution is 8 t_{CC} in staggered mode, and 1 t_{CC} in non-staggered mode.

The basic structure of the two timers, illustrated in **Figure 17-3**, is identical, except for the input pin (see mark).

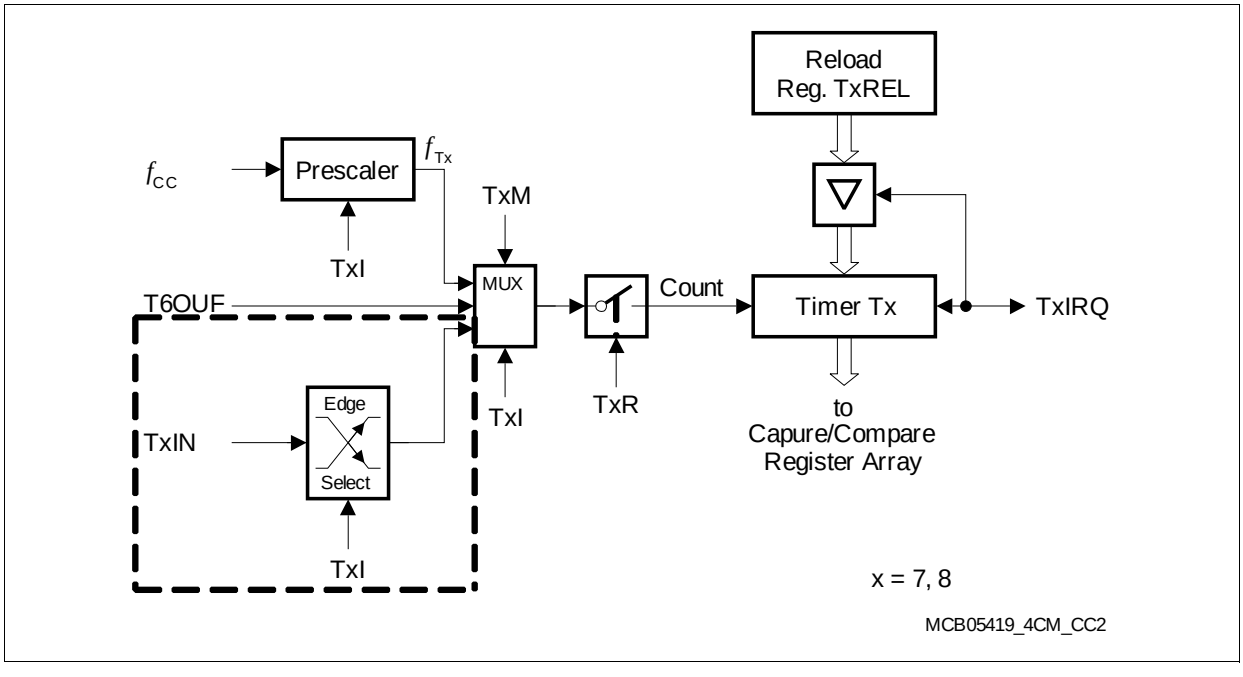


Figure 17-3 Block Diagram of a CAPCOM Timer

The functions of the CAPCOM timers are controlled via the bit-addressable control register T78CON. The high-byte of T78CON controls T8, the low-byte of T78CON controls T7. The control options are identical for all four timers (except for external input).

In all modes, the timers are always counting upward. The current timer values are accessible for the CPU in the timer registers Tx, which are non bit-addressable registers. When the CPU writes to a register Tx in the state immediately before the respective timer increment or reload is to be performed, the CPU write operation has priority and the increment or reload is disabled to guarantee correct timer operation.

CC2_T78CON
Timer 7/8 Control Register **SFR ($FF20_H/90_H$)** **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	T8R	T8RSEL		T8M	T8I			-	T7R	T7RSEL		T7M	T7I		
-	rw	-		rw	rw			-	rw	-		rw	rw		

Field	Bits	Type	Description
TxR	14, 6	rw	**Timer/Counter Tx Run Control** 0 Timer/Counter Tx is disabled 1 Timer/Counter Tx is enabled
TxM	11, 3	rw	**Timer/Counter Tx Mode Selection** 0 Timer Mode 1 Counter Mode
TxI	[10:8], [2:0]	rw	**Timer/Counter Tx Input Selection** **Timer Mode** (TxM = 0): Input frequency $f_{Tx} = f_{CC}/2^{(<TxI>+3)}$ or $f_{CC}/2^{(<TxI>)}$, depending on (non-)staggered mode, see **Table 17-1** **Counter Mode** (TxM = 1): 000 Overflow/Underflow of GPT Timer T6 001 Positive (rising) edge on pin TxIN 010 Negative (falling) edge on pin TxIN 011 Any edge (rising and falling) on pin TxIN 1XX Reserved. Do not use this combination! *Note: For timer T8 the only option in counter mode is 000_B. T8 stop in all other cases.*
T7RSEL	[5:4]	rw	**Timer T7 External Run Selection** Bit field T7RSEL defines the event of signal T7HR that can set the run bit T7R by HW. 00 The external setting of T7R is disabled. 01 Bit T7R is set if a rising edge of signal T7HR is detected. 10 Bit T7R is set if a falling edge of signal T7HR is detected. 11 Bit T7R is set if an edge of signal T7HR is detected.

Field	Bits	Type	Description
T8RSEL	[13:12]	rw	**Timer T8 External Run Selection** Bit field T8RSEL defines the event of signal T8HR that can set the run bit T8R by HW. 00 The external setting of T8R is disabled. 01 Bit T8R is set if a rising edge of signal T8HR is detected. 10 Bit T8R is set if a falling edge of signal T8HR is detected. 11 Bit T8R is set if an edge of signal T8HR is detected.

The timer run flags TxR allow the starting and stopping of the timers. The following description of the timer modes and operation always applies to the enabled state of the timers, i.e. the respective run flag is assumed to be set.

Timer Mode

In Timer Mode (TxM = 0), the input clock for a CAPCOM2 timer is derived from f_{CC}, divided by a programmable prescaler. Each timer has its own individual prescaler, controlled through the individual bitfields TxI in the timer control registers T01CON and T78CON.

The input frequency f_{Tx} for a timer Tx and its resolution r_{Tx} are determined by the following formulas:

Staggered Mode:

$$f_{Tx}[MHz] = \frac{f_{CC}[MHz]}{2^{(<TxI> + 3)}} \qquad r_{Tx}[\mu s] = \frac{2^{(<TxI> + 3)}}{f_{CC}[MHz]} \tag{17.1}$$

Non-Staggered Mode:

When a timer overflows from $FFFF_H$ to 0000_H, it is reloaded with the value stored in its respective reload register TxREL. The reload value determines the period P_{Tx} between two consecutive overflows of Tx as follows:

$$f_{Tx}[MHz] = \frac{f_{CC}[MHz]}{2^{<TxI>}} \qquad r_{Tx}[\mu s] = \frac{2^{<TxI>}}{f_{CC}[MHz]} \tag{17.2}$$

Staggered Mode:

$$P_{Tx}[\mu s] = \frac{(2^{16} - <TxREL>) \times 2^{(<TxI> + 3)}}{f_{CC}[MHz]} \tag{17.3}$$

Non-Staggered Mode:

$$P_{Tx}[\mu s] = \frac{(2^{16} - <TxREL>) \times 2^{<TxI>}}{f_{CC}[MHz]} \quad (17.4)$$

After a timer has been started by setting its run flag (TxR), the first increment will occur within the time interval which is defined by the selected timer resolution. All further increments occur exactly after the time defined by the timer resolution.

Examples for timer input frequencies, resolution and periods, which result from the selected prescaler option in TxI when using a 40 MHz clock, are listed in **Table 17-1** below. The numbers for the timer periods are based on a reload value of 0000_H. Note that some numbers may be rounded.

Table 17-1 Timer Tx Input Clock Selection for Timer Mode, f_{CC} = 40 MHz

TxI	Prescaler	Input Frequency	Resolution	Period
Non-Staggered Mode				
000_B	8	5 MHz	200 ns	13.11 ms
001_B	16	2.5 MHz	400 ns	26.21 ms
010_B	32	1.25 MHz	800 ns	52.43 ms
011_B	64	625 kHz	1.6 µs	104.86 ms
100_B	128	312.5 kHz	3.2 µs	209.72 ms
101_B	256	156.25 kHz	6.4 µs	419.43 ms
110_B	512	78.125 kHz	12.8 µs	838.86 ms
111_B	1024	39.0625 kHz	25.6 µs	1677.72 ms
Non-Staggered Mode				
000_B	1	40 MHz	25 ns	1.6384 ms
001_B	2	20 MHz	50 ns	3.2768 ms
010_B	4	10 MHz	100 ns	6.5536 ms
011_B	8	5 MHz	200 ns	13.11 ms
100_B	16	2.5 MHz	400 ns	26.21 ms
101_B	32	1.25 MHz	800 ns	52.43 ms
110_B	64	625 kHz	1.6 µs	104.86 ms
111_B	128	312.5 kHz	3.2 µs	209.72 ms

Counter Mode

In Counter Mode (TxM = 1), the input clock of a CAPCOM2 timer is either derived from an associated external input pin, T7IN, or from the over-/underflows of GPT timer T6.

Using an external signal connected to pin TxIN as a counting signal is only possible for timer T7. The only counter option for timer T8 is using the over-/underflows of the GPT timer T6 (selected by TxI = 000_B).

Bitfields T7I are used to select either a positive, a negative, or both a positive and a negative transition of the external signal at pin T7IN to trigger an increment of timer T7. Please note that certain criteria must be met for the external signal and the port pin programming for this mode in order to operate properly. These conditions are detailed in **Chapter 17.10**.

Timer Overflow and Reload

When a CAPCOM2 timer contains the value $FFFF_H$ at the time a new count trigger occurs, a timer interrupt request is generated, and the timer is loaded with the contents of its associated reload register TxREL. The timer then resumes incrementing with the next count trigger starting from the reloaded value.

The reload registers TxREL are not bitaddressable. After reset, they contain the value 0000_H.

17.2 CAPCOM2 Timer Interrupts

Upon a timer overflow the corresponding timer interrupt request flag TxIR for the respective timer will be set. This flag can be used to generate an interrupt or trigger a PEC service request, when enabled by the respective interrupt enable bit TxIE.

Each timer has its own bitaddressable interrupt control register and its own interrupt vector. The organization of the interrupt control registers TxIC is identical with the other interrupt control registers.

CC2_T7IC
CAPCOM T7 Intr. Ctrl. Reg. **ESFR ($FF6C_H$/BD_H)** **Reset Value: - - 00_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-							GPX	T7IR	T7IE	ILVL				GLVL	
-	-	-	-	-	-	-	rw	rwh	rw	rw				rw	

CC2_T8IC
CAPCOM T8 Intr. Ctrl. Reg. **ESFR ($FF6E_H$/BE_H)** **Reset Value: - - 00_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-							GPX	T8IR	T8IE	ILVL				GLVL	
-	-	-	-	-	-	-	rw	rwh	rw	rw				rw	

Note: Please refer to the general Interrupt Control Register description for an explanation of the control fields.

17.3 Capture/Compare Channels

The 16-bit capture/compare registers CC0 through CC15 (CC16 through CC31) are used as data registers for capture or compare operations with respect to timers T0/T7 and T1/T8. The capture/compare registers are not bit-addressable.

The functions of the 16 capture/compare registers of a unit are controlled by 4 bit-addressable 16-bit mode control registers, named CC1_M0 … CC1_M3 (CC2_M4 … CC2_M7), which are all organized identically (see description below). Each register contains the bits for mode selection and timer allocation for four capture/comp. registers.

TBD RegisterAdressSpace

17.3.1 Capture/Compare Registers for the CAPCOM2 (CC31 … CC16)

CC2_M4
CAPCOM Mode Ctrl. Reg. 4 **SFR ($FF22_H/91_H$)** **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACC 19	MOD19			ACC 18	MOD18			ACC 17	MOD17			ACC 16	MOD16		
rw	rw			rw	rw			rw	rw			rw	rw		

CC2_M5
CAPCOM Mode Ctrl. Reg. 5 **SFR ($FF24_H/92_H$)** **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACC 23	MOD23			ACC 22	MOD22			ACC 21	MOD21			ACC 20	MOD20		
rw	rw			rw	rw			rw	rw			rw	rw		

CC2_M6
CAPCOM Mode Ctrl. Reg. 6 **SFR ($FF26_H/93_H$)** **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACC 27	MOD27			ACC 26	MOD26			ACC 25	MOD25			ACC 24	MOD24		
rw	rw			rw	rw			rw	rw			rw	rw		

CC2_M7
CAPCOM Mode Ctrl. Reg. 7 **SFR ($FF28_H/94_H$)** **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACC 31	**MOD31**			**ACC 30**	**MOD30**			**ACC 29**	**MOD29**			**ACC 28**	**MOD28**		
rw	rw			rw	rw			rw	rw			rw	rw		

Field	Bits	Type	Description
ACCy	15, 11, 7, 3	rw	**Allocation Bit for CAPCOM Register CCy** 0 CCy allocated to Timer T0 or T7, respectively 1 CCy allocated to Timer T1 or T8, respectively
MODy	[14:12], [10:8], [6:4], [2:0]	rw	**Mode Selection for CAPCOM Register CCy** See **Table 17-2**.

Each of the registers CCy may be individually programmed for capture mode or for one of 4 different compare modes, and may be allocated individually to one of the two timers of the respective CAPCOM unit. A special double-register compare mode combines two registers to act on one common output signal. When capture or compare operations are disabled for one of the CCy registers, it may be used for general purpose variable storage.

Table 17-2 Selection of Capture Modes and Compare Modes

Mode	MODy	Selected Operating Mode
Disabled	**0 0 0**	**Disable Capture and Compare Modes** The respective CAPCOM register may be used for general variable storage.
Capture	**0 0 1**	**Capture on Positive Transition (Rising Edge)** at Pin CCyIO
	0 1 0	**Capture on Negative Transition (Falling Edge)** at Pin CCyIO
	0 1 1	**Capture on Positive and Negative Transition (Both Edges)** at Pin CCyIO

Table 17-2 Selection of Capture Modes and Compare Modes (cont'd)

Mode	MODy	Selected Operating Mode
Compare	**1 0 0**	**Compare Mode 0:** Interrupt Only Several interrupts per timer period. Can enable double-register compare mode for Bank2 registers.
	1 0 1	**Compare Mode 1:** Toggle Output Pin on each Match Several compare events per timer period. Can enable double-register compare mode for Bank1 registers.
	1 1 0	**Compare Mode 2:** Interrupt Only Only one interrupt per timer period.
	1 1 1	**Compare Mode 3:** Set Output Pin on each Match Reset output pin on each timer overflow; only one interrupt per timer period.

The detailed discussion of the capture and compare modes is valid for all the capture/compare channels, so registers, bits and pins are only referenced by a placeholder.

Note: A capture or compare event on channel 31 may be used to trigger a channel injection on the XC2000's A/D converter if enabled.

17.4 Capture Mode Operation

In Capture Mode, the current contents of a CAPCOM timer are latched (captured) into the respective capture/compare register in response to an external event. This is used, for example, to record the time at which an external event has occurred, or to measure the distance between two external events in timer increments.

The event to cause a capture of a timer's contents can be programmed to be either the positive, the negative, or both the positive and the negative transition of the external signal connected to the input pin. This triggering transition is selected by bitfield MODy in the respective mode control register. When the selected external signal transition occurs, the selected timer's contents is latched into the capture/compare register and the respective interrupt request line CCyIRQ is activated. This can cause an interrupt or PEC service request, when enabled.

Note: A capture input can be used as an additional external interrupt input. The capture operation can be disregarded in this case.

Either the contents of timer T0/T7 or T1/T8 can be captured, selected by the timer allocation control bit ACCy in the respective mode control register.

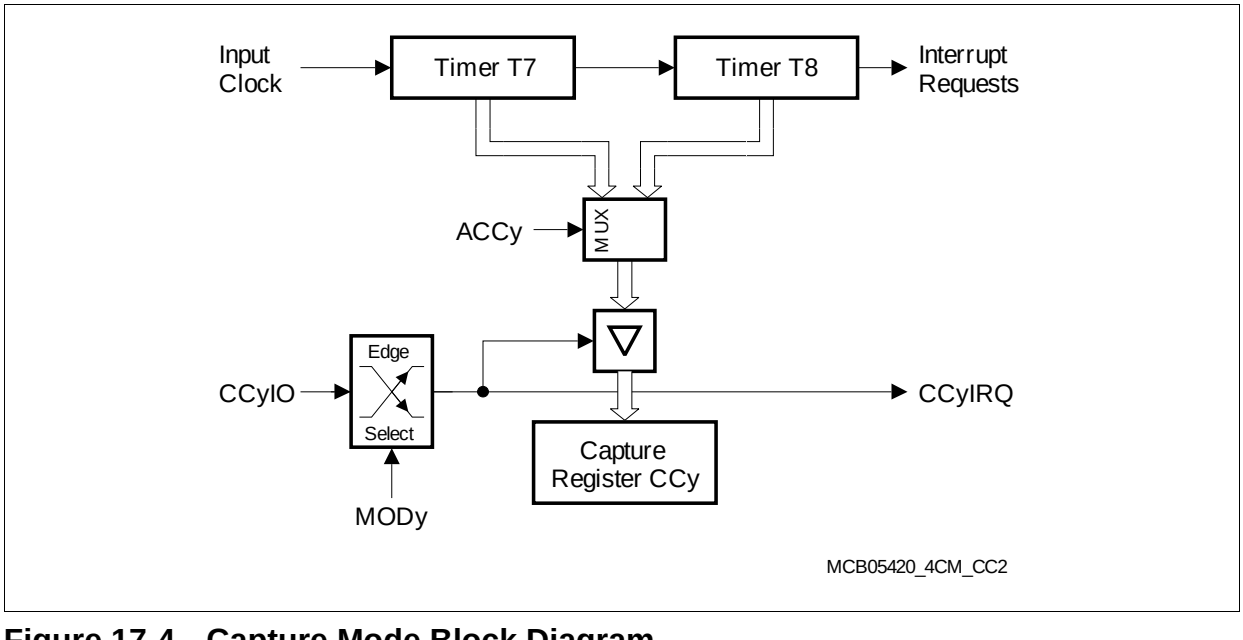


Figure 17-4 Capture Mode Block Diagram

For capture operation, the respective pin must be programmed for input. To ensure that a transition of the input signal is recognized correctly, its level must be held high or low for a minimum number of module clock cycles before it changes. This information can be found in **Section 17.10**.

17.5 Compare Mode Operation

The compare modes allow triggering of events (interrupts and/or output signal transitions) or generation of pulse trains with minimum software overhead. In all compare modes, the 16-bit value stored in a capture/compare register CCy (in the following also referred to as ‘compare value’) is continuously compared with the contents of the allocated timer (T7 or T8). If the current timer contents match the compare value, the interrupt request line associated with register CCy is activated and, depending on the compare mode, an output signal can be generated at the corresponding output pin CCyIO.

Four different compare modes are available, which can be selected individually for each of the capture/compare registers by bitfield MODy in the respective mode control register. Modes 0 and 2 do not influence the output signals. In the following, each mode is described in detail.

In addition to these ‘single-register’ modes, a ‘double-register’ compare mode enables two registers to operate on the same pin. This feature can further reduce software overhead, as two different compare values can be programmed to control a sequence of transitions for a signal. See **Section 17.5.5** for details for this operation.

In all Compare Modes, the comparator performs an ‘equal to’ comparison. This means, a match is only detected when the timer contents are equal to the contents of a compare register. In addition, the comparator is only enabled in the clock cycle directly after the timer was incremented by hardware. This is done to prevent repeated matches if the timer does not operate with the highest possible input clock (either in timer or counter mode). In this case, the timer contents would remain at the same value for several or up to thousands of cycles. This operation has the side-effect, that software modifications of the timer contents will have no effect regarding the comparator. If a timer is set by software to the same value stored in one of the compare registers, no match will be detected. If a compare register is set to a value smaller than the current timer contents, no action will take place.

For the exact operation of the port output function, please see **Section 17.6**.

When two or more compare registers are programmed to the same compare value[1)], their corresponding interrupt request flags will be set and the selected output signals will be generated after the allocated timer is incremented to this compare value. Further compare events on the same compare value are disabled[2)] until the timer is incremented again or written to by software. After a reset, compare events for register CCy will only become enabled, if the allocated timer has been incremented or written to by software and one of the compare modes described in the following has been selected for this register.

1) In staggered mode these interrupts and output signals are generated sequentially (see **Section 17.8**).

2) Even if more compare cycles are executed before the timer increments (lower timer frequency) a given compare value only results in one single compare event.

17.5.1 Compare Mode 0

This is an interrupt-only mode which can be used for software timing purposes. In this mode, the interrupt request line CCyIRQ is activated each time a match is detected between the contents of the compare register CCy and the allocated timer. A match means, the contents of the timer are equal to (‘=’) the contents of the compare register. Several of these compare events are possible within a single timer period, if the compare value in register CCy is updated during the timer period. The corresponding port signal CCyIO is not affected by compare events in this mode and can be used as general purpose IO.

*Note: If compare mode 0 is programmed for one of the bank2 registers the double-register compare mode may be enabled for this register (see **Chapter 17.5.5**).*

17.5.2 Compare Mode 1

This is a compare mode which influences the associated output signal. Besides this, the basic operation is as in compare mode 0. Each time a match is detected between the contents of the compare register CCy and the allocated timer, the interrupt request line CCyIRQ is activated. In addition, the associated output signal is toggled. Several of these compare events are possible within a single timer period, if the compare value in register CCy is updated during the timer period.

*Note: If compare mode 1 is programmed for one of the bank1 registers the double-register compare mode may be enabled for this register (see **Section 17.5.5**).*

For the exact operation of the port output signal, please see **Section 17.6**.

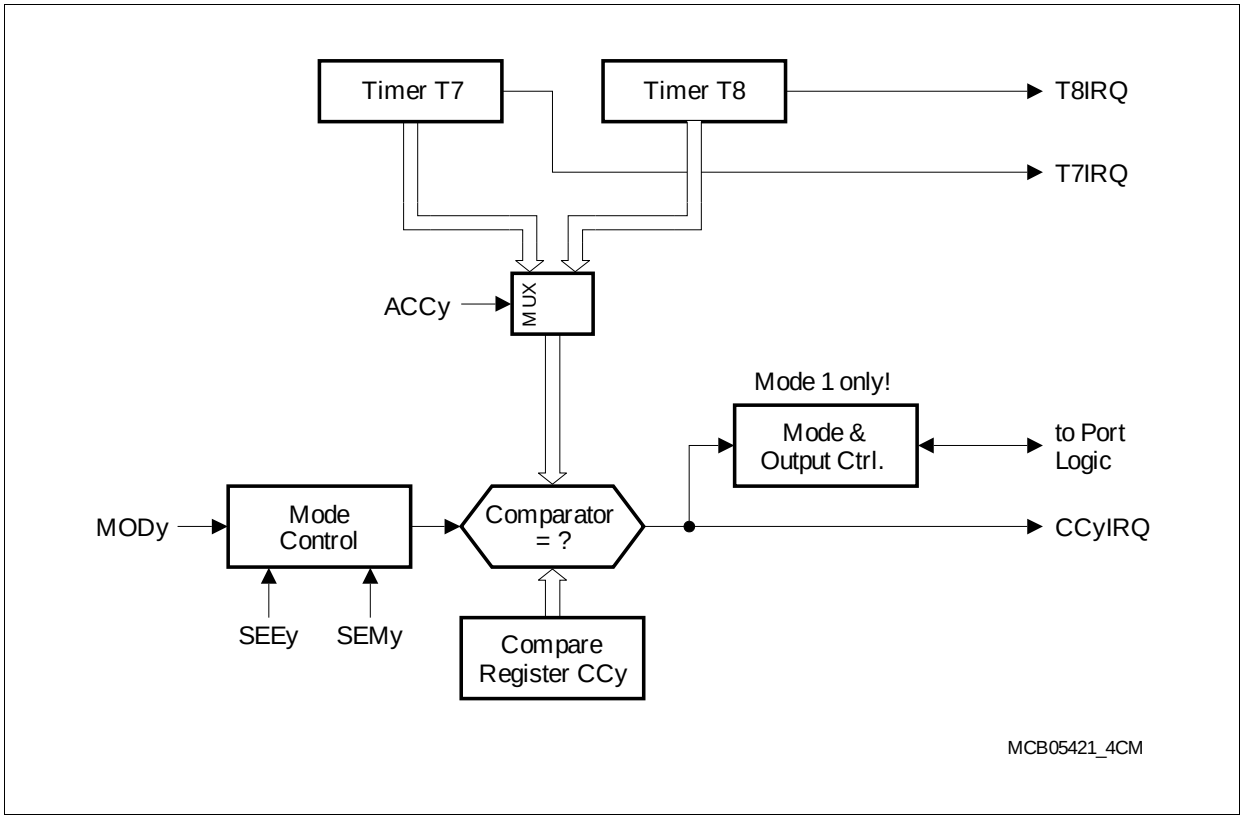


Figure 17-5 Compare Mode 0 and 1 Block Diagram

Note: The signal remains unaffected in compare mode 0.

Figure 17-6 illustrates a few example cases for compare modes 0 and 1.

In all examples, the reload value of the used timer is set to $FFF9_H$. When the timer overflows, it starts counting from this value upwards.

In Case 1, register CCy contains the value $FFFC_H$. When the timer reaches this value, a match is detected, and the interrupt request line CCyIRQ is activated. In compare mode 0, this is all that will happen. In compare mode 1, additionally the associated port output is toggled, causing an inversion of the output signal. If the contents of register CCy are not changed, this operation will take place each time the timer reaches the programmed compare value.

In Case 2, software reloads the compare register CCy with $FFFF_H$ after the first match with $FFFC_H$ has occurred. As the timer continues to count up, it finally reaches this new compare value, and a new match is detected, activating the interrupt request line (both modes) and toggling the output signal (compare mode 1). If then the compare value is left unchanged, the next match will occur when the timer reaches $FFFF_H$ again.
This example illustrates, that further compare matches are possible within the current timer period (this is in contrast to compare modes 2 and 3).

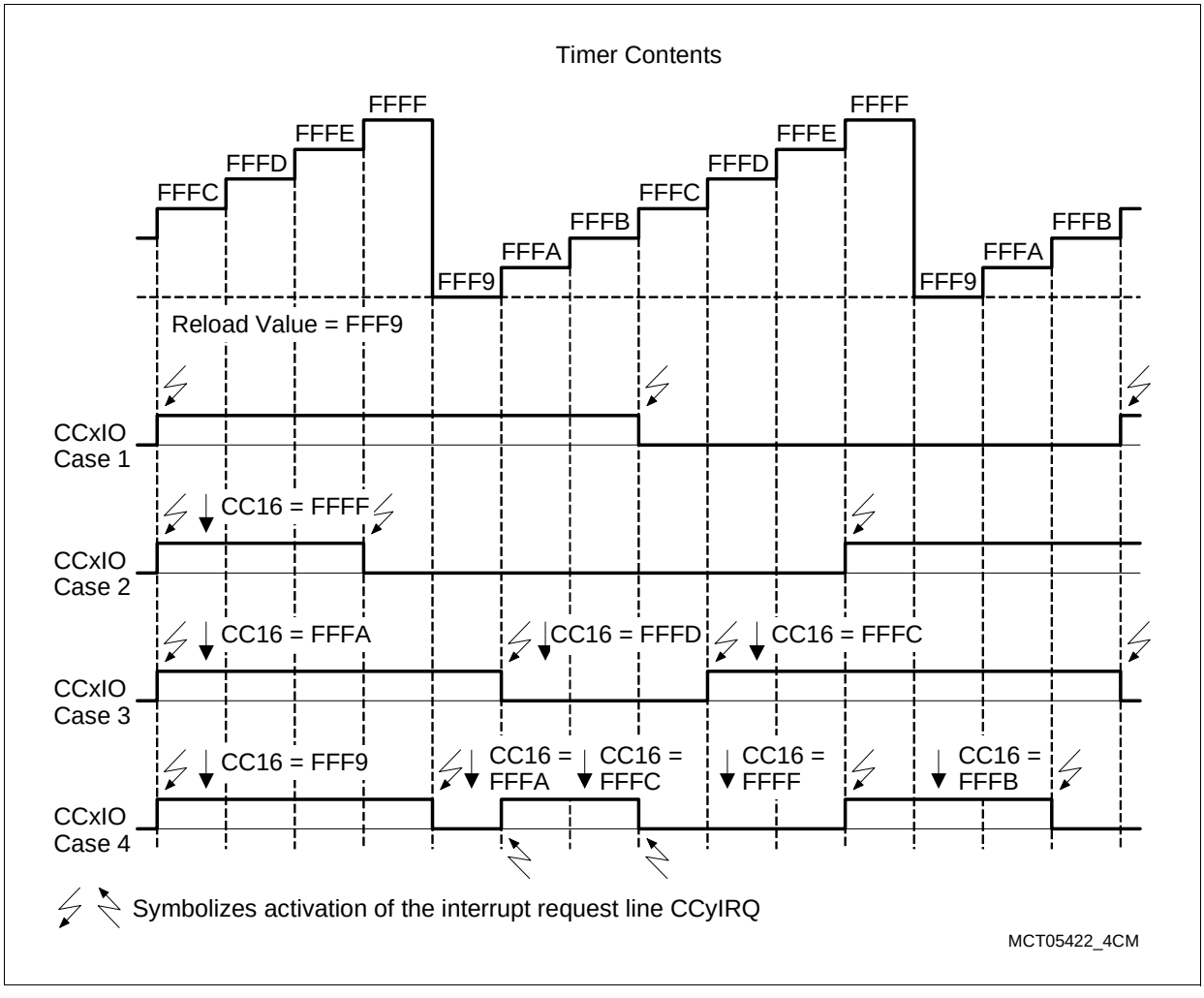


Figure 17-6 Examples for Compare Modes 0 and 1

In Case 3, a new compare value, higher than the current timer contents, causes a new match within the current timer period. The compare register is reloaded with $FFFA_H$ after the first match (at $FFFC_H$). However, the timer has already passed this value. Thus, it will take until the timer reaches $FFFA_H$ in the following timer period to cause the desired compare match. Reloading register CCy now with a value higher than the current timer contents will cause the next match within this period.

In Case 4, the compare values are equal to the timer reload value or to the maximum count value, $FFFF_H$.

17.5.3 Compare Mode 2

Compare mode 2 is an interrupt-only mode similar to compare mode 0. The main difference is that only one compare match, corresponding to one interrupt request, is possible within a given timer period.

When a match is detected in compare mode 2 for the first time within a count period of the allocated timer, the interrupt request line CCyIRQ is activated. In addition, all further compare matches within the current timer period are disabled, even if a new compare value, higher than the current timer contents, would be written to the register. This blocking is only released when the allocated timer overflows. A new compare value written to the compare register after the first match will only go into effect within the following timer period.

17.5.4 Compare Mode 3

Compare mode 3 is based on compare mode 2, but additionally influences the associated port pin. Only one compare event is possible within one timer period.

When a match is detected in compare mode 3 for the first time within a count period of the allocated timer, the interrupt request line CCyIRQ is activated, and the associated output signal is set to 1. In addition, all further compare matches within the current timer period are disabled, even if a new compare value, higher than the current timer contents, would be written to the register. This blocking is only released when the allocated timer overflows. A new compare value written to the compare register after the first match will only go into effect within the following timer period.

The overflow signal is also used to reset the associated output signal to 0.

Special attention has to be paid when the compare value is set equal to the timer reload value. In this case, the compare match signal would try to set the output signal, while the timer overflow tries to reset the output signal. This conflict is avoided such that the state of the output signal is left unchanged in this case.

Note: When the compare value is changed from a value above the current timer contents to a value below the current timer contents, the new value is not recognized before the next timer period.

For the exact operation of the port output signal, please see **Section 17.6**.

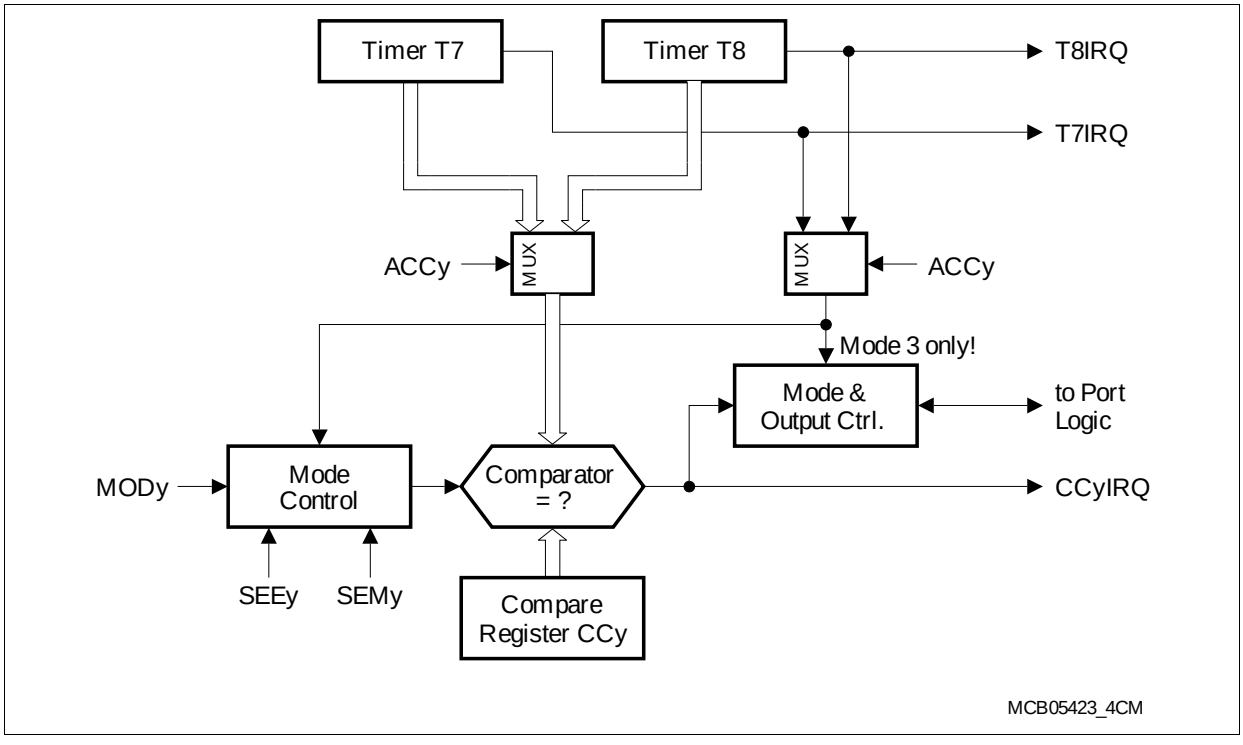


Figure 17-7 Compare Mode 2 and 3 Block Diagram

Note: The port latch and signal remain unaffected in compare mode 2.

Figure 17-8 illustrates a few timing examples for compare modes 2 and 3.

In all examples, the reload value of the used timer is set to $FFF9_H$. When the timer overflows, it starts counting from this value upwards.

In Case 1, register CCy contains the value $FFFC_H$. When the timer reaches this value, a match is detected, and the interrupt request line CCyIRQ is activated. In compare mode 2, this is all that will happen. In compare mode 3, additionally the associated port output is set to 1. The timer continues to count, and finally reaches its overflow. At this point, the port output is reset to 0 again. Note that, although not shown in the diagrams, the overflow signal of the timer also activates the associated interrupt request line TxIRQ. If the contents of register CCy are not changed, the port output will be set again during the following timer period, and reset again when the timer overflows. This operation is ideal for the generation of a pulse width modulated (PWM) signal with a minimum of software overhead. The pulse width is varied by changing the compare value accordingly.

In Case 2, the compare operation is blocked after the first match within a timer period. After the first match at $FFFC_H$, the interrupt request is generated and the port output is set. In addition, further compare matches are disabled. If now a new compare value is written to register CCy, no interrupt request and no port output influence will take place, although the new compare value is higher than the current timer contents. Only after the overflow of the timer, the compare logic is enabled again, and the next match will be

detected at $FFFF_H$. One can see, that this operation is ideal for PWM generation, as software can write a new compare value regardless of whether this value is higher or lower than the current timer contents. It is assured that the new value (usually written to the compare register in the appropriate interrupt service routine) will only go into effect during the following timer period.

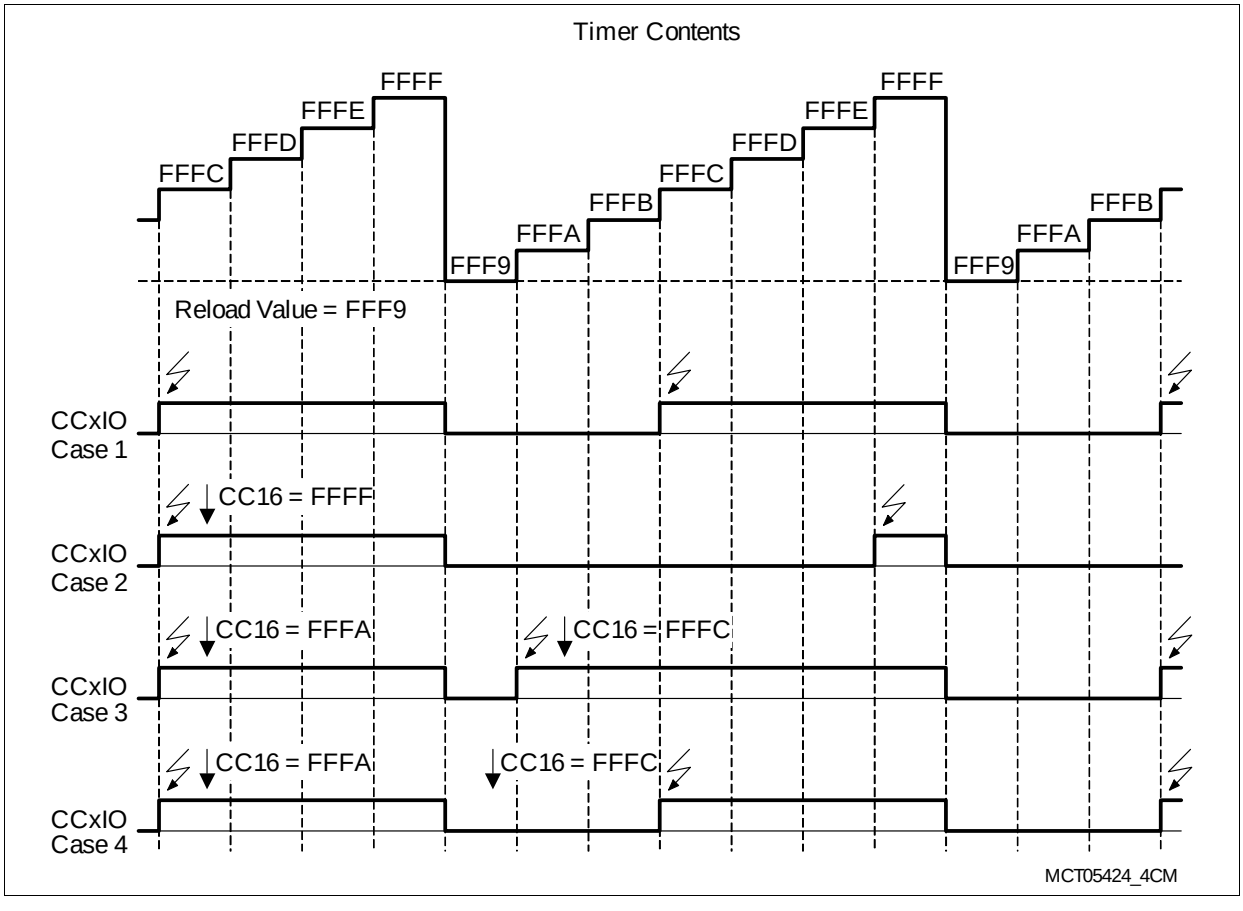


Figure 17-8 Timing Example for Compare Modes 2 and 3

Note: In compare mode 2, only interrupt requests are generated, in mode 3, also the output signals are generated.

In Case 3, further examples for the operation of the compare match blocking are illustrated.

In Case 4, a new compare value is written to a compare register before the first match within the timer period. One can see that, of course, the originally programmed compare match (at $FFFA_H$) will not take place. The first match will be detected at $FFFC_H$. However, it is important to note that the reprogramming of the compare register took place asynchronously - this means, the register was written to without any regard to the current contents of the timer. This is dangerous in the sense that the effect of such an asynchronous reprogramming is not easily predictable. If the timer would have already reached the originally programmed compare value of $FFFA_H$ by the time the software

wrote to the register, a match would have been detected and the reprogramming would go into effect during the next timer period.

The examples in **Figure 17-9** show special cases for compare modes 2 and 3. Case 1 illustrates the effect when the compare value is equal to the reload value of the timer. An interrupt is generated in both modes. In mode 3, the output signal is not affected - it remains at the high level. Setting the compare value equal to the reload value easily enables a 100% duty cycle signal for PWM generation. The important advantage here is that the compare interrupt is still generated and can be used to reload the next compare value. Thus, no special treatment is required for this case (see Case 3).

Cases 2, 4, and 5 show different options for the generation of a 0% duty cycle signal. Case 2 shows an asynchronous reprogramming of the compare value equal to the reload value. At the end of the current timer period, a compare interrupt will be generated, which enables software to set the next compare value. The disadvantage of this method is that at least two timer periods will pass until a new regular compare value can go into effect. The compare match with the reload value $FFF9_H$ will block further compare matches during that timer period. This is additionally illustrated by Case 4.

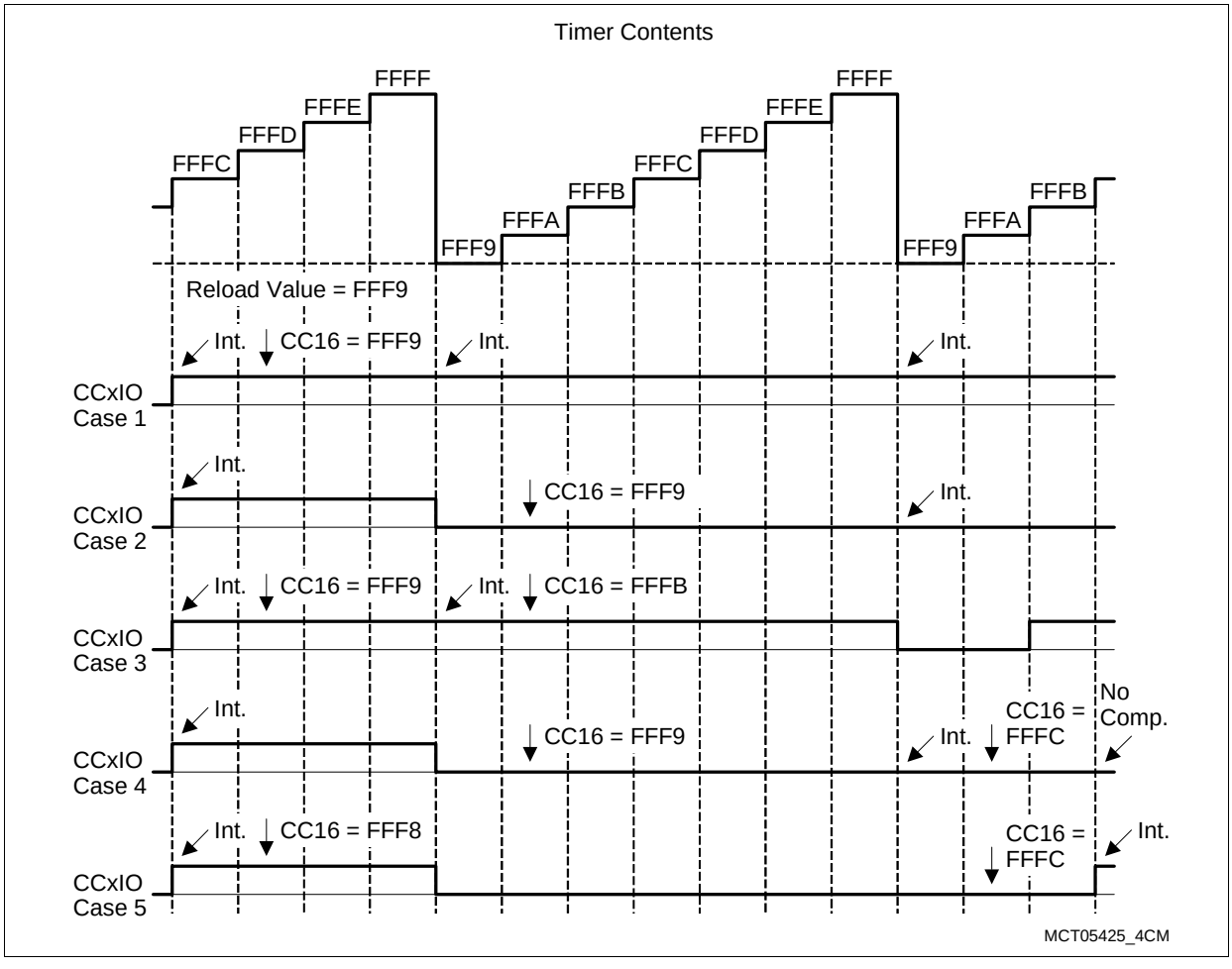


Figure 17-9 Special Cases in Compare Modes 2 and 3

Case 5 shows an option to get around this problem. Here, the compare register is reloaded with $FFF8_H$, a value which is lower than the timer reload value. Thus, the timer will never reach this value, and no compare match will be detected. The output signal will be set to 0 after the first timer overflow. However, after the second overflow, software now reloads the compare register with a regular compare value. As no compare blocking has taken place (since there was no compare match), the newly written compare value will go into effect during the current timer period.

17.5.5 Double-Register Compare Mode

The Double-Register Compare Mode makes it possible to further reduce software overhead for a number of applications. In this mode, two compare registers work together to control one output. This mode is selected via the DRM register, or by a special combination of compare modes for the two registers.

For double-register compare mode, the 16 capture/compare registers of a CAPCOM2 unit are regarded as two banks of 8 registers each. The lower eight registers form bank1, while the upper eight registers form bank2. For double-register mode, a bank1 register and a bank2 register form a register pair. Both registers of this register pair operate on the pin associated with the bank1 register.

The relationship between the bank1 and bank2 register of a pair and the effected output pins for double-register compare mode is listed in **Table 17-3**.

Table 17-3 Register Pairs for Double-Register Compare Mode

CAPCOM2 Unit			
Register Pair		**Used Output Pin**	**Control Bitfield in CC2DRM**
Bank 1	**Bank 2**		
CC16	CC24	CC16IO	DR0M
CC17	CC25	CC17IO	DR1M
CC18	CC26	CC18IO	DR2M
CC19	CC27	CC19IO	DR3M
CC20	CC28	CC20IO	DR4M
CC21	CC29	CC21IO	DR5M
CC22	CC30	CC22IO	DR6M
CC23	CC31	CC23IO	DR7M

The double-register compare mode can be programmed individually for each register pair. Double-register compare mode can be selected via a certain combination of compare modes for the two registers of a pair. The bank1 register must be programmed for mode 1 (with port influence), while the bank2 register must be programmed for mode 0 (interrupt-only).

Double-register compare mode can be controlled (this means, enabled or disabled) for each register pair via the associated control bitfield DRxM in register CC2_DRM.

CC2_DRM
Double-Reg. Cmp. Mode Reg. SFR ($FF2A_H/95_H$) Reset Value: 0000_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DR7M		**DR6M**		**DR5M**		**DR4M**		**DR3M**		**DR2M**		**DR1M**		**DR0M**	
rw		rw		rw		rw		rw		rw		rw		rw	

Field	Bits	Type	Description
DRxM	[1:0], [3:2], [5:4], [7:6], [9:8], [11:10], [13:12], [15:14]	rw	**Double Register x Compare Mode Selection** 00 DRM is controlled via the combination of compare modes 1 and 0 (compatibility mode) 01 DRM disabled regardless of compare modes 10 DRM enabled regardless of compare modes 11 Reserved *Note: “x” indicates the register pair index in a bank.*

Double-register compare mode can be controlled individually for each of the register pairs.

In the block diagram of the double-register compare mode (**Figure 17-10**), a bank2 register will be referred to as CCz, while the corresponding bank1 register will be referred to as CCy.

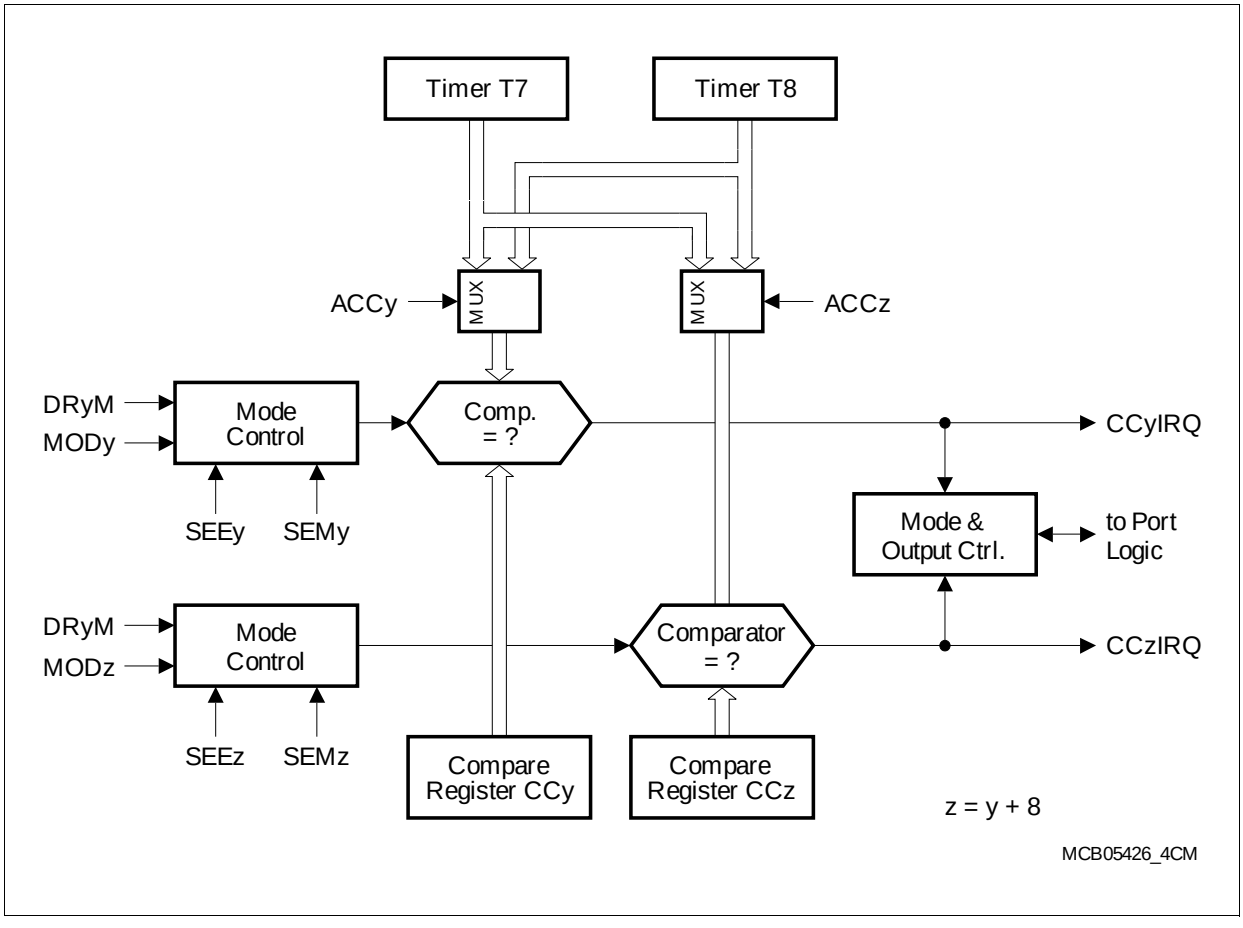


Figure 17-10 Double-Register Compare Mode Block Diagram

When a match is detected for one of the two registers in a register pair (CCy or CCz), the associated interrupt request line (CCyIRQ or CCzIRQ) is activated, and pin CCyIO, corresponding to the bank1 register CCy, is toggled. The generated interrupt always corresponds to the register that caused the match.

Note: If a match occurs simultaneously for both register CCy and register CCz of the register pair, pin CCyIO will be toggled only once, but two separate compare interrupt requests will be generated.

Each of the two registers of a pair can be individually allocated to one of the two timers in the CAPCOM2 unit. This offers a wide variety of applications, as the two timers can run in different modes with different resolution and frequency. However, this might require sophisticated software algorithms to handle the different timer periods.

Note: The signals CCzIO (which do not serve for double-register compare mode) may be used for general purpose IO.

17.6 Compare Output Signal Generation

This section discusses the interaction between the CAPCOM Unit and the Port Logic. The block diagram illustrated in **Figure 17-11** details the logic of the block “Mode & Output Control”, shown in **Figure 17-5**, **Figure 17-7**, and **Figure 17-10**.

Each output signal is latched in its associated bit of the respective output latch register CCx_OUT. The individual bits are updated each time an associated compare event occurs. The bits of these registers are connected to the respective port pins as an alternate output function of a port line.

Compare signals can also directly affect the associated port output latch Px. In this case, the port latch must be selected for the respective pin. The direct port latch option is disabled in non-staggered mode or it can be disabled by setting bit PL in register CCx_IOC.

Register CCx_OUT is always updated in parallel to the update of the port output latch.

CC2_OUT
Compare Output Reg. **SFR ($FF2C_H/96_H$)** **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC15IO	**CC14IO**	**CC13IO**	**CC12IO**	**CC11IO**	**CC10IO**	**CC9IO**	**CC8IO**	**CC7IO**	**CC6IO**	**CC5IO**	**CC4IO**	**CC3IO**	**CC2IO**	**CC1IO**	**CC0IO**
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
CCyIO	15 … 0	rwh	**Compare Output for Channel y** Alternative port output for the associated port pin.

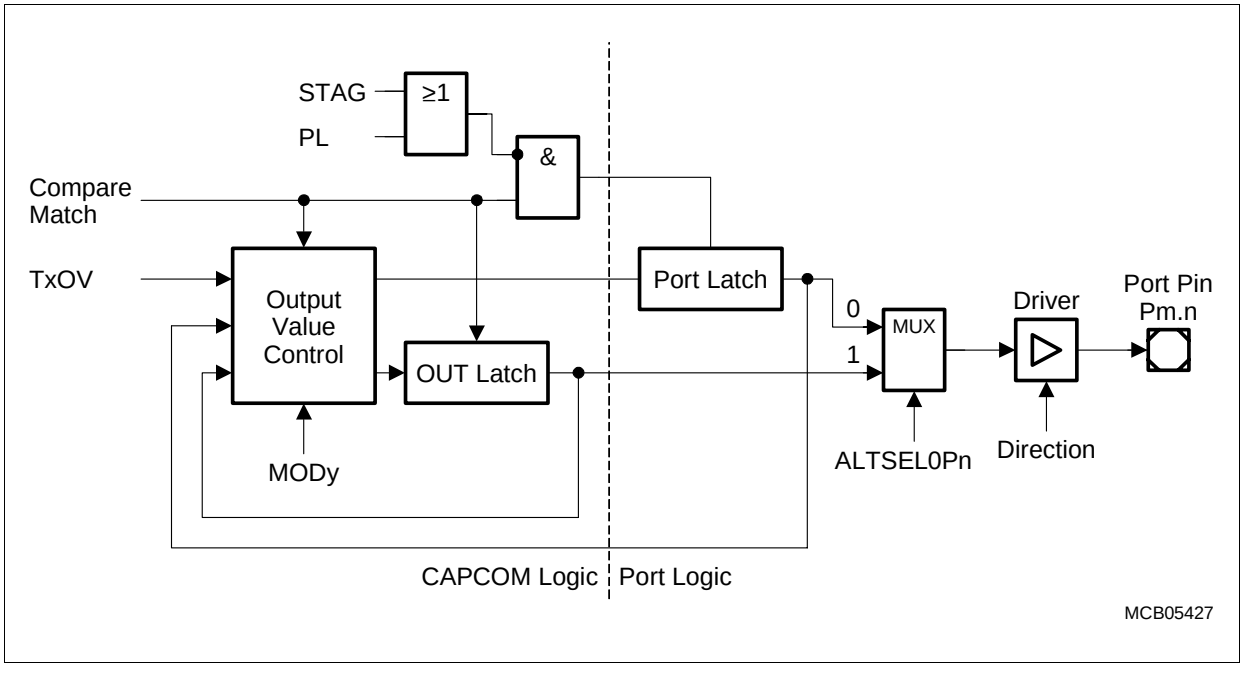


Figure 17-11 Port Output Block Diagram for Compare Modes

Note: A compare output signal is visible at the pin only in compare modes 1 or 3.

The output signal of a compare event can either be a 1, a 0, the complement of the current level, or the previous level. The block ‘Output Value Control’ determines the correct new level based on the compare event, the timer overflow signal, and the current states of the Port and OUT latches. For the output toggle function (e.g. in compare mode 1), the state of the output latch is read, inverted, and then written back.

The associated output pins either drives the port latch signal or the OUT signal, selected by register ALTSEL (see **Figure 17-11**).

Note: If the port output latch is written to by software at the same time it would be altered by a compare event, the software write will have priority. In this case the hardware-triggered change will not become effective.

17.7 Single Event Operation

If an application requires that one and only one compare event needs to take place (within a certain time frame), single event operation helps to reduce software overhead and to eliminate the need for fast reaction upon events.

In order to achieve a single event operation without this feature, software would have to either disable the compare mode or write a new value, which is outside of the count range of the timer, into the compare register, after the programmed compare match has taken place. Thus, usually an interrupt service routine is required to perform this operation. Interrupt response time may be critical if the timer period is very short - the disable operation needs to be completed before the timer would reach the same value again.

The single event operation eliminates the need for software to react after the first compare match. The complete operation can be set up before the event, and no action is required after the event. The hardware takes care of generating only one event, and then disabling all further compare matches.

This option is programmed via the Single Event Mode register CC2_SEM and the Single Event Enable register CC2_SEE. Each register provides one bit for each CCy register of a unit.

CC2_SEM
Single Event Mode Ctrl. Reg. **SFR ($FE28_H/14_H$)** **Reset Value: 0000_H**

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

Field	Bits	Type	Description
SEMy	15 … 0	rw	**Single Event Mode Control** 0 Single Event Mode disabled for channel y 1 Single Event Mode enabled for channel y

CC2_SEE
Single Event Enable Reg. **SFR ($FE2A_H/15_H$)** **Reset Value: 0000_H**

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

Field	Bits	Type	Description
SEEy	15 … 0	rw	**Single Event Enable Control** 0 Single Event disabled for channel y 1 Single Event enabled for channel y *Note: This bit is cleared by hardware after the event.*

To setup a single event operation for a CCy register, software first programs the desired compare operation and compare value, and then sets the respective bit in register CC2_SEM to enable the single event mode. At last, the respective event enable bit in register CC2_SEE is set.

When the programmed compare match occurs, all operations of the selected compare mode take place. In addition, hardware automatically disables all further compare matches and reset the event enable bit in register CC2_SEE to 0. As long as this bit is cleared, any compare operation is disabled. To setup a new event, this bit must first be set again.

17.8 Staggered and Non-Staggered Operation

The CAPCOM2 units can run in one of two basic operation modes: Staggered Mode and Non-Staggered Mode. The selection between these modes is performed via register IOC.

CC2_IOC
I/O Control Register **ESFR ($F066_H/33_H$)** **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-												-	STAG	PL	-
-												-	rw	rw	-

Field	Bits	Type	Description
STAG	2	rw	**Staggered Mode Control** 0 CAPCOM operates in Staggered Mode 1 CAPCOM operates in Non-Staggered Mode
PL	1	rw	**Port Lock Control** 0 Compare output signals affect the associated port output latch 1 Direct influence of the port output latch by the compare output signals is disabled

Note: Whenever Non-Staggered Mode is enabled (STAG = 1) or Port Lock is activated (PL = 1), the port output registers are not changed by the CAPCOM unit.

In staggered mode, a CAPCOM2 operation cycle consists of 8 module clock cycles, and the outputs of the compare events of the different registers are staggered, that is, the outputs for compare matches with the same compare value are not switched at the same time, but with a fixed time delay. This operation helps to reduce noise and peak power consumption caused by simultaneous switching outputs.

In non-staggered Mode, a CAPCOM2 operation cycle is equal to one module clock cycle, and all compare outputs for compare events with the same compare value are switched in the same clock cycle. This mode offers a faster operation and increased resolution of the CAPCOM2 unit, 8 times higher than in staggered mode.

Staggered Mode

Figure 17-12 illustrates the staggered mode operation. In this example, all CCy registers are programmed for compare mode 3.

Registers CC16, CC17, and CC18 are all programmed for a compare value of $FFFE_H$. When the timer increments to $FFFE_H$, the comparator detects a match for all of the three registers. The output CC16IO of register CC16 is switched to 1 one cycle after the comparator match. However, the outputs CC17IO and CC18IO are not switched at the same time, but one, respectively two cycles later. This staggering of the outputs continues for all registers including register CC23. The number of the register indicates the delay of the output signal in clock cycles - the output of register CC23 is switched 7 cycles later than the one of register CC16. In the example, the compare value for register CC13 is set to $FFFD_H$. Thus, the output is switched in the last clock cycle of the CAPCOM2 cycle in which the timer reached $FFFD_H$.

When the timer overflows, all compare outputs are reset to 0 (compare mode 3). Again, the staggering of the output signals can be seen from **Figure 17-12**.

Looking at registers CC24 through CC31 shows that their outputs are switched in parallel to the respective outputs of registers CC16 through CC23. In fact, the staggering is performed in parallel for the upper and the lower register bank. In this way, it is assured, that both compare signals of a register pair in double-register compare mode operate simultaneously.

In staggered mode direct port latch switching (see **Section 17.6**) is possible. However, it is possible to use the alternate output function option of the associated port pins to output the compare signals.

Note: This is a general description and only refers to channels connected to pins.

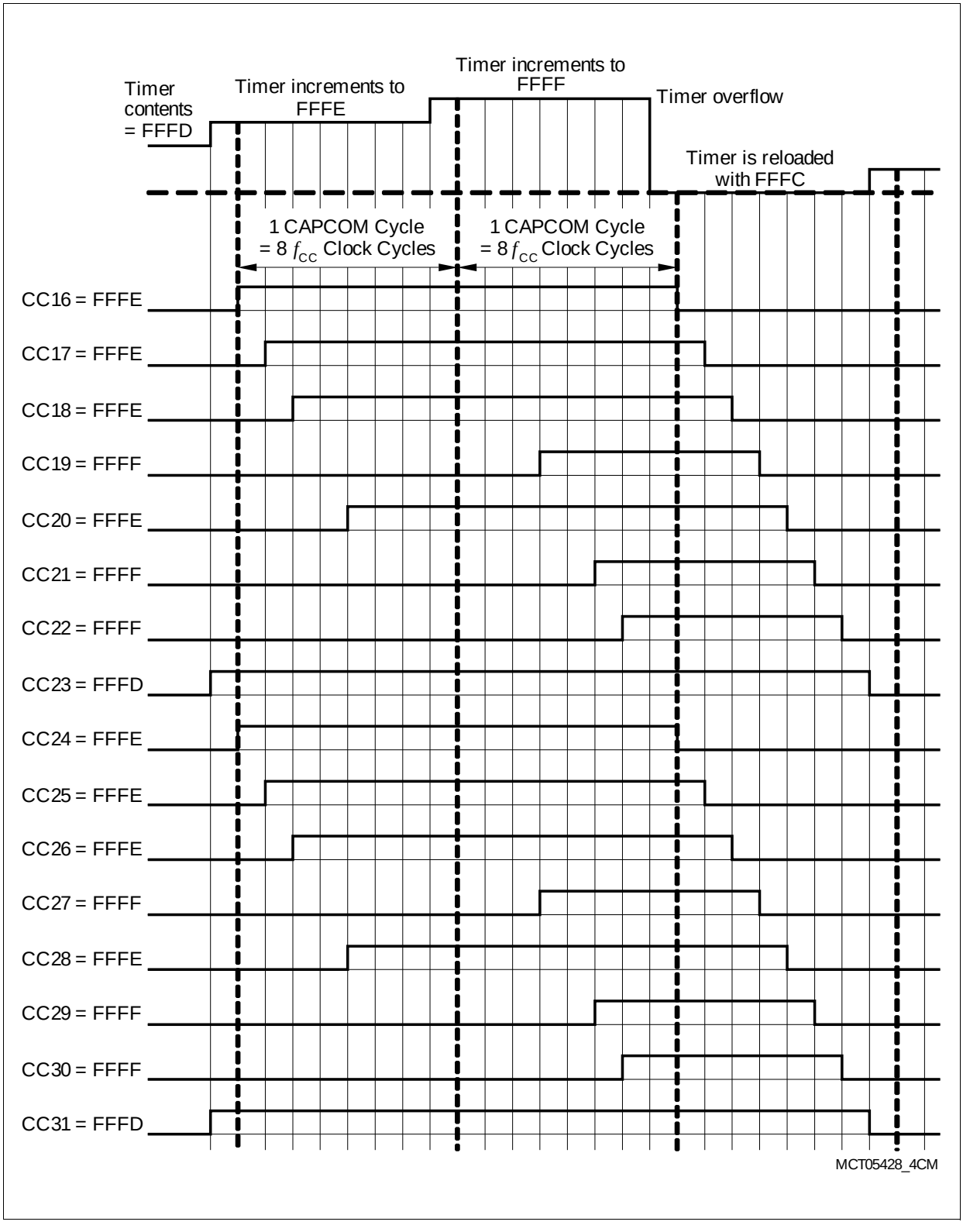


Figure 17-12 Staggered Mode Operation

Non-Staggered Mode

To gain maximum speed and resolution with the CAPCOM2 unit, it can be switched to non-staggered mode. In this mode, one CAPCOM2 operation cycle is equal to one module clock cycle. Timer increment and the comparison of its new contents with the contents of the compare register takes place within one clock cycle. The appropriate output signals are switched in the following clock cycle (in parallel to the next possible timer increment and comparison).

Figure 17-13 illustrates the non-staggered mode. Note that when the timer overflows, it also takes one additional clock cycle to switch the output signals.

Note: In non-staggered mode, direct port latch switching is disabled.
This is a general description and only refers to channels connected to pins.

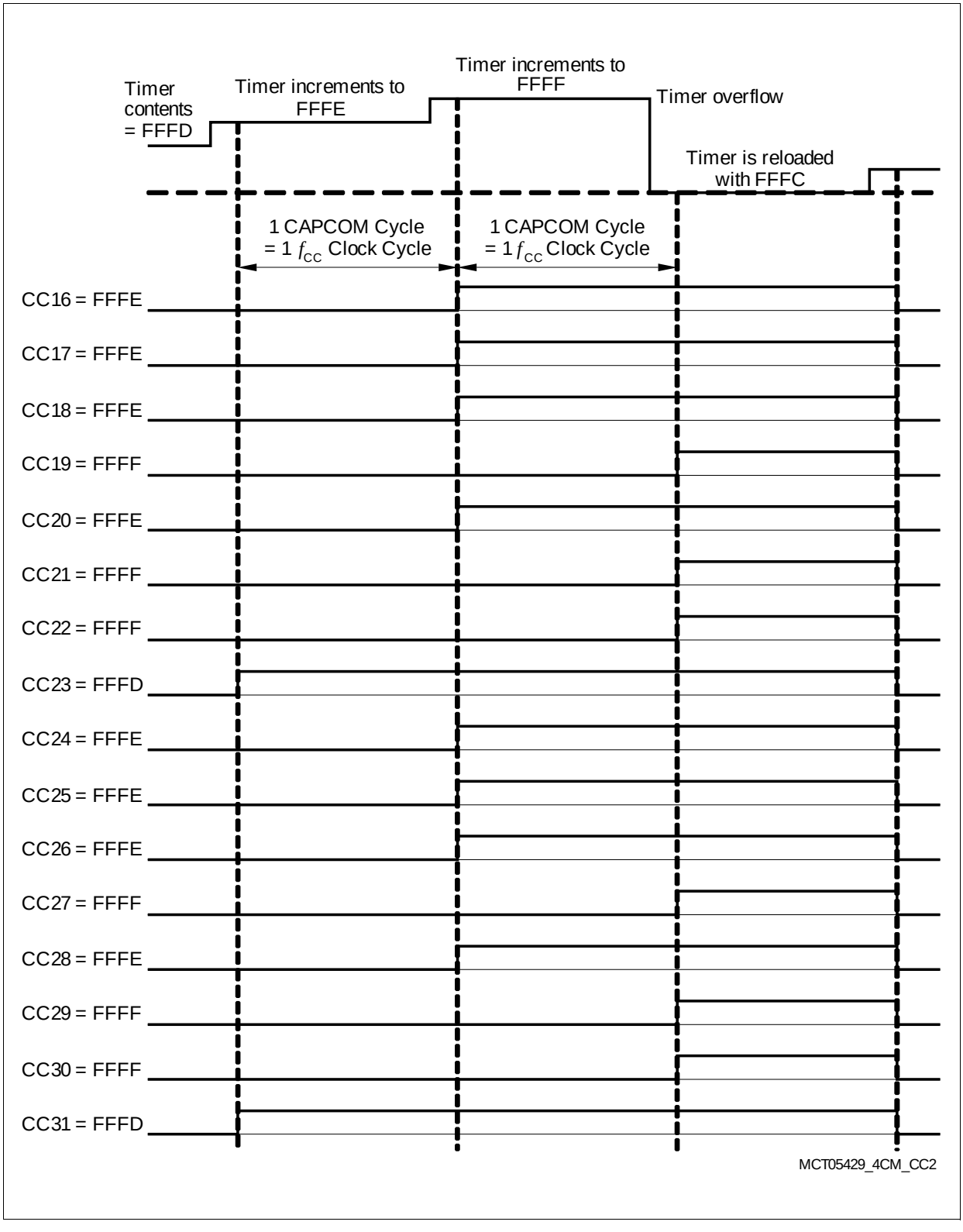


Figure 17-13 Non-Staggered Mode Operation

17.9 CAPCOM2 Interrupts

Upon a capture or compare event, the interrupt request flag CCyIR for the respective capture/compare register CCy is automatically set. This flag can be used to generate an interrupt or trigger a PEC service request when enabled by the interrupt enable bit CCyIE. Capture interrupts can be regarded as external interrupt requests with the additional feature of recording the time at which the triggering event occurred.

Each of the capture/compare registers has its own bitaddressable interrupt control register and its own interrupt vector allocated. These registers are organized in the same way as all other interrupt control registers. The basic register layout is shown below, **Table 17-4** lists the associated addresses.

CC2_CCyIC
CAPCOM Intr. Ctrl. Reg. **(E)SFR (Table 17-4)** **Reset Value: - - 00_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-							GPX	CCy IR	CCy IE	ILVL				GLVL	
-	-	-	-	-	-	-	rw	rwh	rw	rw				rw	

Note: Please refer to the general Interrupt Control Register description for an explanation of the control fields.

Table 17-4 CAPCOM Unit Interrupt Control Register Addresses

CAPCOM2 Unit		
Register Name	**Address**	**Reg. Space**
CC2_CC16IC	$FE60_H/30_H$	ESFR
CC2_CC17IC	$FE62_H/31_H$	ESFR
CC2_CC18IC	$FE64_H/32_H$	ESFR
CC2_CC19IC	$FE66_H/33_H$	ESFR
CC2_CC20IC	$FE68_H/34_H$	ESFR
CC2_CC21IC	$FE6A_H/35_H$	ESFR
CC2_CC22IC	$FE6C_H/36_H$	ESFR
CC2_CC23IC	$FE6E_H/37_H$	ESFR
CC2_CC24IC	$FE70_H/38_H$	ESFR
CC2_CC25IC	$FE72_H/39_H$	ESFR
CC2_CC26IC	$FE74_H/3A_H$	ESFR
CC2_CC27IC	$FE76_H/3B_H$	ESFR
CC2_CC28IC	$FE78_H/3C_H$	ESFR
CC2_CC29IC	$FE7A_H/3D_H$	ESFR
CC2_CC30IC	$FE7C_H/3E_H$	ESFR
CC2_CC31IC	$FE7E_H/3F_H$	ESFR

17.10 External Input Signal Requirements

The external input signals of a CAPCOM2 unit are sampled by the CAPCOM2 logic based on the module clock and the basic operation mode (staggered or non-staggered mode). To assure that a signal level is recognized correctly, its high or low level must be held active for at least one complete sampling period.

The duration of a sampling period is one module clock cycle in non-staggered mode, and 8 module clock cycles in staggered mode. To recognize a signal transition, the signal needs to be sampled twice. If the level of the first sampling is different to the level detected during the second sampling, a transition is recognized. Therefore, a minimum of two sampling periods are required for the sampling of an external input signal. Thus, the maximum frequency of an input signal must not be higher than half the module clock frequency in non-staggered mode, and a $1/16^{th}$ of the module clock frequency in staggered mode.

Table 17-5 summarizes the requirements and limits for external input signals.

Table 17-5 CAPCOM2 External Input Signal Limits

	Non-Staggered Mode	Staggered Mode
Maximum Input Frequency	$f_{CC}/2$	$f_{CC}/16$
Minimum Input Signal Level Duration	$1/f_{CC}$	$8/f_{CC}$

In order to use an external signal as a count or capture input, the port pin to which it is connected must be configured as input.

Note: For example for test purposes a pin used as a count or capture input may be configured as output. Software or an other peripheral may control the respective signal and thus trigger count or capture events.

In order to cause a compare output signal to be seen by the external world, the associated port pin must be configured as output. Compare output signals can either directly switch the port latch, or the output of the CC2_OUT latch is used as an alternate output function of a port.

17.10.1 KSCCFG Register

KSCCFG
Kernel State Configuration Register

SFR(FE24$_H$) **Reset Value: 0000$_H$**

15	14	13 12	11	10	9 8	7	6	5 4	3 2	1	0
BP COM	0	COMCFG	BP SUM	0	SUMCFG	BP NOM	0	NOMCFG	0	BP MOD EN	MOD EN
w	r	rw	w	r	rw	w	r	rw	r	w	rw

Field	Bits	Type	Description
MODEN	0	rw	**Module Enable** This bit enables the module kernel clock and the module functionality. 0_B The module is switched off. It does not react on mode control actions and the module clock is switched off immediately (without stop condition). The module does not react on read accesses and ignores write accesses. 1_B The module is switched on. *Note: This bit is reset by a class 3 reset.*
BPMODE	1	w	**Bit Protection for MODEN** This bit enables the write access to the bit MODEN. It always reads 0. It is only active during the write access cycle. 0_B MODEN is not changed. 1_B MODEN is updated with the written value. *Note: This bit is reset by a class 3 reset.*
NOMCFG	[5:4]	rw	**Normal Operation Mode Configuration** This bit field defines the kernel mode applied in normal operation mode. $0X_B$ The module is switched on. $1X_B$ The module is switched off. This field is taken into account for CR = 00 or 11. *Note: This bit is reset by a class 3 reset.*

Field	Bits	Type	Description
BPNOM	7	w	**Bit Protection for NOMCFG** This bit enables the write access to the bit field NOMCFG. It always reads 0. It is only active during the write access cycle. 0_B NOMCFG is not changed. 1_B NOMCFG is updated with the written value. *Note: This bit is reset by a class 3 reset.*
SUMCFG	[9:8]	rw	**Suspend Mode Configuration** This bit field defines the kernel mode applied in suspend mode. $0X_B$ The module is switched on. $1X_B$ The module is switched off. This field is taken into account for CR = 01. *Note: This bit is reset by a class 1 reset.*
BPSUM	11	w	**Bit Protection for SUMCFG** This bit enables the write access to the bit field SUMCFG. It always reads 0. It is only active during the write access cycle. 0_B SUMCFG is not changed. 1_B SUMCFG is updated with the written value. *Note: This bit is reset by a class 1 reset.*
COMCFG	[13:12]	rw	**Clock Off Mode Configuration** This bit field defines the kernel mode applied in clock off mode. $0X_B$ The module is switched on. $1X_B$ The module is switched off. This field is taken into account for CR = 10. *Note: This bit is reset by a class 3 reset.*
BPCOM	15	w	**Bit Protection for COMCFG** This bit enables the write access to the bit field COMCFG. It always reads 0. It is only active during the write access cycle. 0_B COMCFG is not changed. 1_B COMCFG is updated with the written value. *Note: This bit is reset by a class 3 reset.*
0	[3:2], 6, 10, 14	r	**Reserved;** returns 0 if read; should be written with 0;

17.11 Interfaces of the CAPCOM Units

The CAPCOM2 unit (see **Figure 17-14**) are connected to their environment in different ways.

Internal Connections

The overflow/underflow signal T6OUF of GPT2 timer T6 is connected to the CAPCOM units, providing an optional clock source for the CAPCOM timers.

The 18 interrupt request lines of the CAPCOM2 unit are connected to the interrupt control block.

Note: The upper 6 input lines from PORT1, connected with the CAPCOM2 unit, can also be used as individual external interrupt inputs.

The upper four channels are connected to the External Request Unit (ERU).

The CAPCOM2 module is clocked with the XC2000 system clock, so $f_{CC} = f_{SYS}$.

External Connections

Twelve capture/compare signals of the CAPCOM2 unit are connected with input/output ports of the XC2000. Depending on the selected direction, these ports may provide capture trigger signals from the external system or issue compare output signals to external circuitry..

Note: Capture trigger signals may also be derived from output pins. In this case, software can generate the trigger edges, for example.

Timer T7 can be clocked by an external signal.

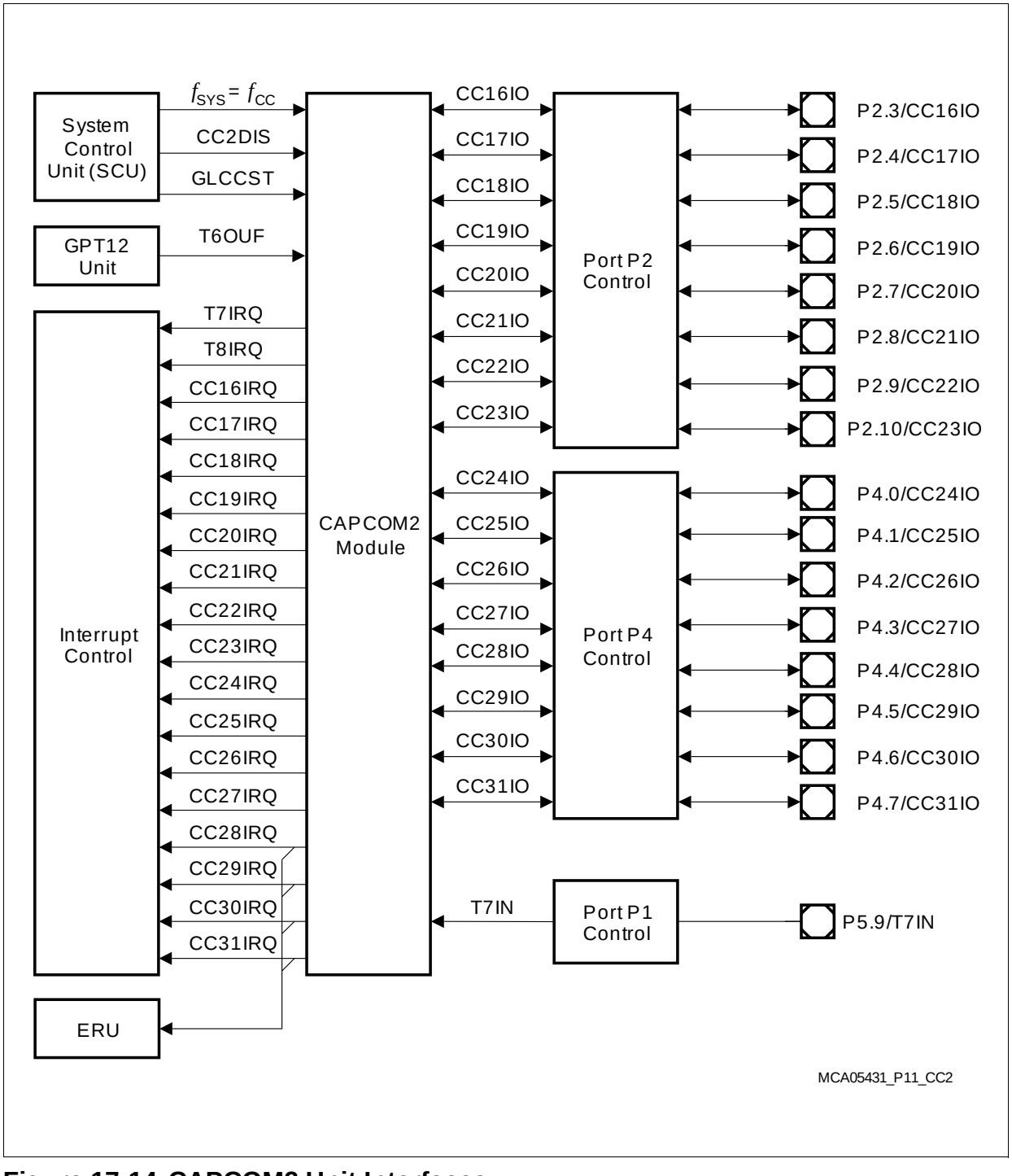


Figure 17-14 CAPCOM2 Unit Interfaces

18 Capture/Compare Unit 6 (CCU6)

The CCU6 is a high-resolution 16-bit capture and compare unit with application specific modes, mainly for AC drive control. Special operating modes support the control of Brushless DC-motors using Hall sensors or Back-EMF detection. Furthermore, block commutation and control mechanisms for multi-phase machines are supported.
It also supports inputs to start several timers synchronously, an important feature in devices with several CCU6 modules.

This chapter is structured as follows:

- Introduction (see **Section 18.1**)
  including the register overview (see **Section 18.1.3**)
- Operating T12 (see **Section 18.2**)
  including T12 related registers (see **Section 18.2.8**)
  and capture/compare control registers (see **Section 18.2.9**)
- Operating T13 (see **Section 18.3**)
  including T13 related registers (see **Section 18.3.6**)
- Trap handling (see **Section 18.4**)
- Multi-Channel mode (see **Section 18.5**)
- Hall sensor mode (see **Section 18.6**)
- Modulation control registers (see **Section 18.7**)
- Interrupt handling (see **Section 18.8**)
  including interrupt registers (see **Section 18.8.2**)
- General module operation (see **Section 18.9**)
  including general registers (see **Section 18.9.3**)
- Module implementation (see **Section 18.10**)

18.1 Introduction

The CCU6 unit is made up of a Timer T12 Block with three capture/compare channels and a Timer T13 Block with one compare channel. The T12 channels can independently generate PWM signals or accept capture triggers, or they can jointly generate control signal patterns to drive AC-motors or inverters.

A rich set of status bits, synchronized updating of parameter values via shadow registers, and flexible generation of interrupt request signals provide means for efficient software-control.

Note: The capture/compare module itself is named CCU6 (capture/compare unit 6).
A capture/compare channel inside this module is named CC6x.

18.1.1 Feature Set Overview

This section gives an overview over the different building blocks and their main features.

Timer 12 Block Features

- Three capture/compare channels, each channel can be used either as capture or as compare channel
- Generation of a three-phase PWM supported (six outputs, individual signals for high-side and low-side switches)
- 16-bit resolution, maximum count frequency = peripheral clock
- Dead-time control for each channel to avoid short-circuits in the power stage
- Concurrent update of T12 registers
- Center-aligned and edge-aligned PWM can be generated
- Single-shot mode supported
- Start can be controlled by external events
- Capability of counting external events
- Many interrupt request sources
- Hysteresis-like control mode

Timer 13 Block Features

- One independent compare channel with one output
- 16-bit resolution, maximum count frequency = peripheral clock
- Concurrent update of T13 registers
- Can be synchronized to T12
- Interrupt generation at period-match and compare-match
- Single-shot mode supported
- Start can be controlled by external events
- Capability of counting external events

Additional Specific Functions

- Block commutation for Brushless DC-drives implemented
- Position detection via Hall-sensor pattern
- Noise filter supported for position input signals
- Automatic rotational speed measurement and commutation control for block commutation
- Integrated error handling
- Fast emergency stop without CPU load via external signal ($\overline{\text{CTRAP}}$)
- Control modes for multi-channel AC-drives
- Output levels can be selected and adapted to the power stage

18.1.2 Block Diagram

The Timer T12 can work in capture and/or compare mode for its three channels. The modes can also be combined (e.g. a channel works in compare mode, whereas another channel works in capture mode). The Timer T13 can work in compare mode only. The multi-channel control unit generates output patterns which can be modulated by T12 and/or T13. The modulation sources can be selected and combined for the signal modulation.

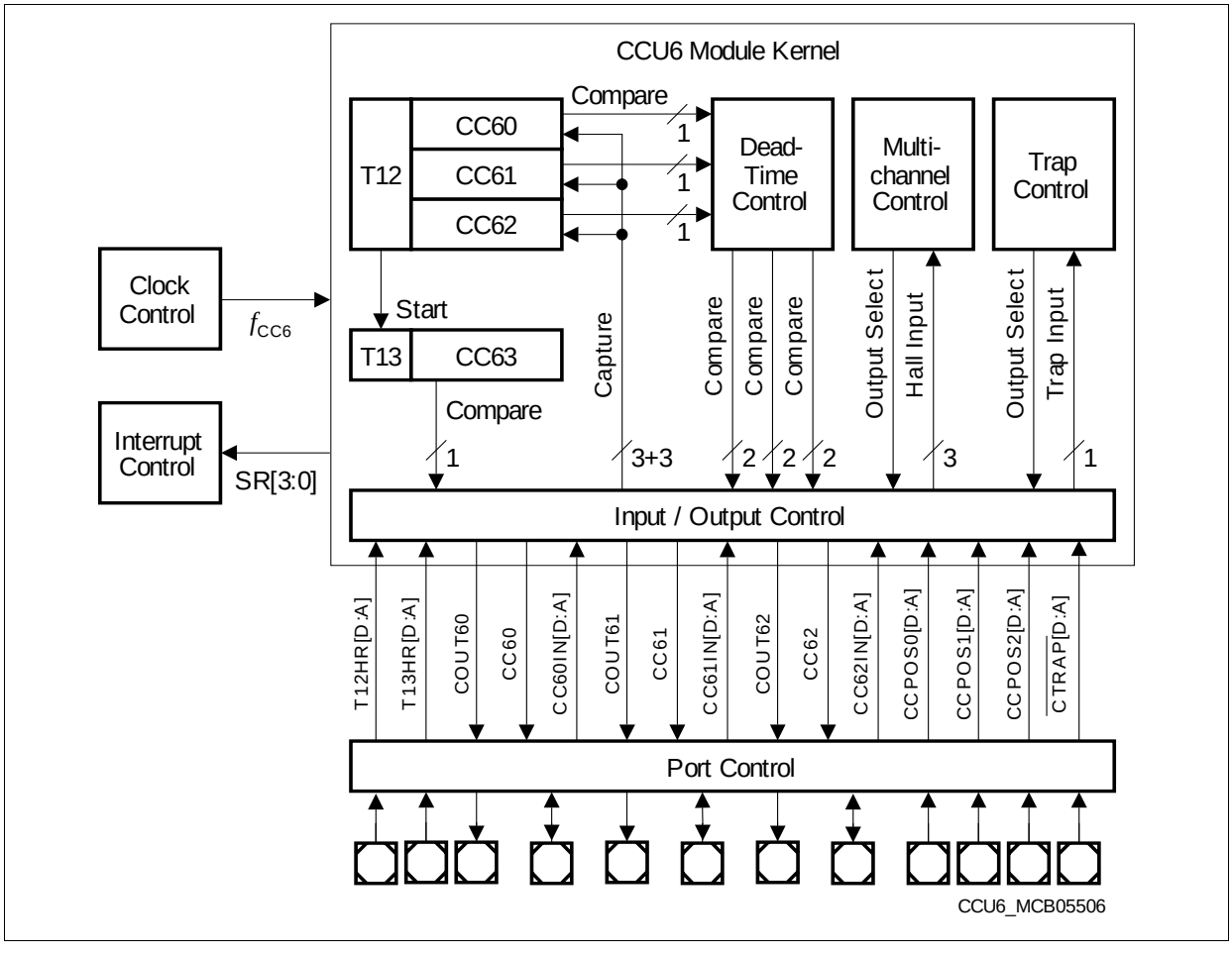


Figure 18-1 CCU6 **Block Diagram**

18.1.3 Register Overview

For the generation of the overall register table, the prefix “CCU6x_” has to be added to the register names in this table to identify the registers of different CCU6 modules that are implemented. In this naming convention, x indicates the module number.

Table 18-1 shows all registers required for programming of a CCU6 module. It summarizes the CCU6 kernel registers and defines their relative addresses and the reset values.
8-bit short addresses are not available for this module.

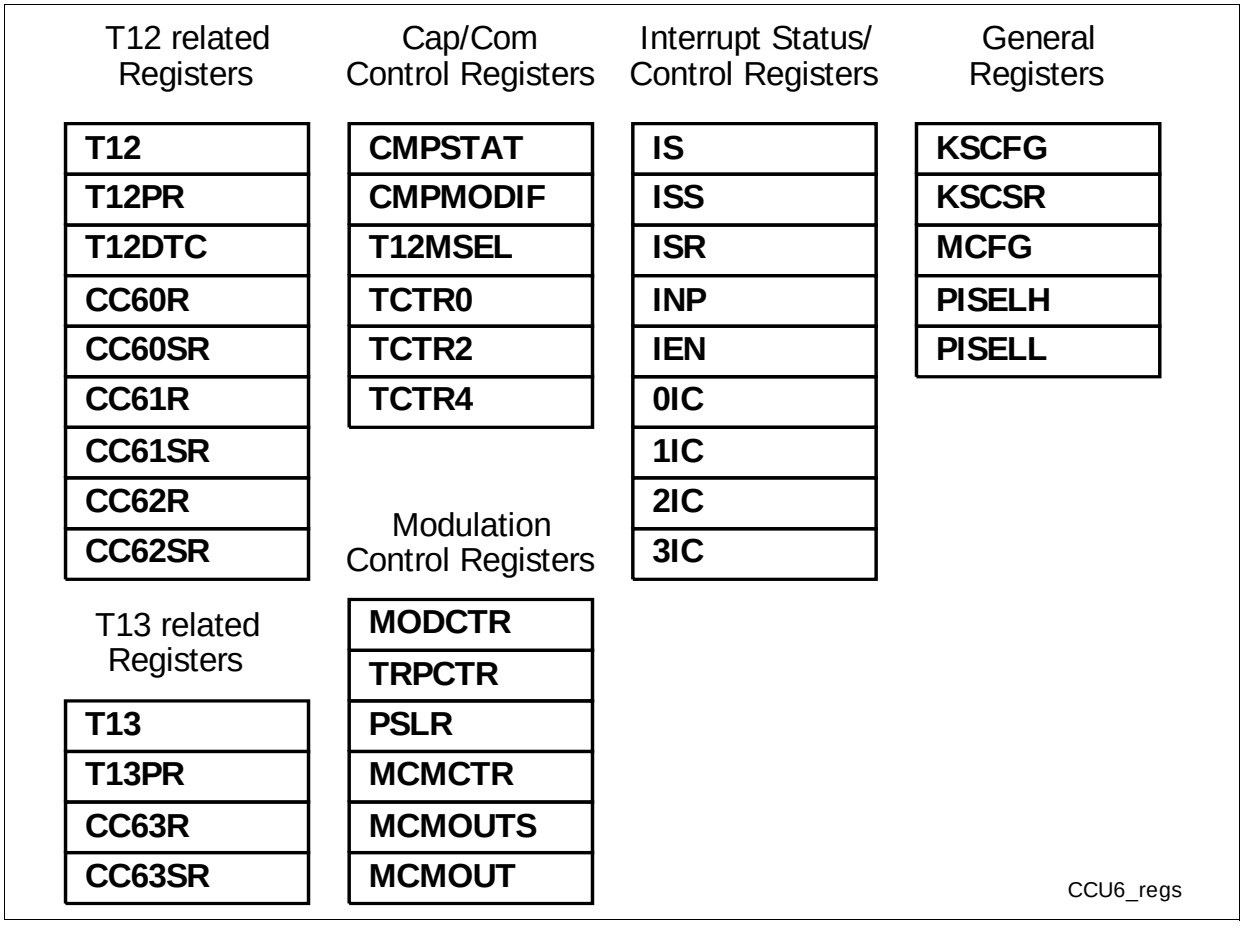


Figure 18-2 CCU6 **Registers**

Table 18-1 CCU6 Module Register Summary

Short Name	Description	Rel. Addr.	Reset Value	See Page
General Registers				
PISELL	Module Port Input Select Register	04_H	0000_H	**Page 18-107**
PISELH	Module Port Input Select Register	06_H	0000_H	**Page 18-109**
KSCFG	Kernel State Configuration Register	00_H	0000_H	**Page 18-111**
KSCSR	Kernel State Control Sensitivity Register	$0E_H$	0000_H	**Page 18-113**
MCFG	Module Configuration Register	$0C_H$	8007_H	**Page 18-114**
Timer T12 related Registers				
T12	Timer 12 Counter Register	10_H	0000_H	**Page 18-33**
T12PR	Timer 12 Period Register	12_H	0000_H	**Page 18-33**
T12DTC	Dead-Time Control Register for Timer T12	14_H	0000_H	**Page 18-36**
CC60R	Capture/Compare Register Channel CC60	18_H	0000_H	**Page 18-34**
CC61R	Capture/Compare Register Channel CC61	$1A_H$	0000_H	**Page 18-34**
CC62R	Capture/Compare Register Channel CC62	$1C_H$	0000_H	**Page 18-34**
CC60SR	Capture/Compare Shadow Register Channel CC60	20_H	0000_H	**Page 18-35**
CC61SR	Capture/Compare Shadow Register Channel CC61	22_H	0000_H	**Page 18-35**
CC62SR	Capture/Compare Shadow Register Channel CC62	24_H	0000_H	**Page 18-35**

Table 18-1 CCU6 Module Register Summary (cont'd)

Short Name	Description	Rel. Addr.	Reset Value	See Page
Capture/Compare Control Registers				
CMPSTAT	Compare State Register	28_H	0000_H	**Page 18-38**
CMPMODIF	Compare State Modification Register	$2A_H$	0000_H	**Page 18-40**
T12MSEL	T12 Capture/Compare Mode Select Register	46_H	0000_H	**Page 18-41**
TCTR0	Timer Control Register 0	$2C_H$	0000_H	**Page 18-42**
TCTR2	Timer Control Register 2	$2E_H$	0000_H	**Page 18-45**
TCTR4	Timer Control Register 4	26_H	0000_H	**Page 18-48**
Timer T13 related Registers				
T13	Timer 13 Counter Register	30_H	0000_H	**Page 18-63**
T13PR	Timer 13 Period Register	32_H	0000_H	**Page 18-64**
CC63R	Compare Register for Timer 13	34_H	0000_H	**Page 18-65**
CC63SR	Compare Shadow Register for Timer 13	36_H	0000_H	**Page 18-65**
Modulation Control Registers				
MODCTR	Modulation Control Register	40_H	0000_H	**Page 18-78**
TRPCTR	Trap Control Register	42_H	0000_H	**Page 18-80**
PSLR	Passive State Level Register	44_H	0000_H	**Page 18-83**
MCMOUTS	Multi-Channel Mode Output Shadow Register	$4A_H$	0000_H	**Page 18-86**

Table 18-1 CCU6 Module Register Summary (cont'd)

Short Name	Description	Rel. Addr.	Reset Value	See Page
MCMOUT	Multi-Channel Mode Output Register	$4C_H$	0000_H	**Page 18-87**
MCMCTR	Multi-Channel Mode Control Register	$4E_H$	0000_H	**Page - HIDDEN**
Interrupt Status and Node Registers				
IS	Interrupt Status Register	50_H	0000_H	**Page 18-91**
ISS	Interrupt Status Set Register	52_H	0000_H	**Page 18-94**
ISR	Interrupt Status Reset Register	54_H	0000_H	**Page 18-96**
INP	Interrupt Node Pointer Register	56_H	3940_H	**Page 18-101**
IEN	Interrupt Node Pointer Register	58_H	0000_H	**Page 18-98**

Note: In the case of a write access to addresses inside the address range (that is covered by the same chip select signal), but that are not the addresses explicitly mentioned for the module, the write access is not taken into account for the module. The same principle is valid for read accesses. In case of a read access to another address, the module does not react.

18.2 Operating Timer T12

The timer T12 block is the main unit to generate the 3-phase PWM signals. A 16-bit counter is connected to 3 channel registers via comparators, that generate a signal when the counter contents match one of the channel register contents. A variety of control functions facilitate the adaptation of the T12 structure to different application needs.
Besides the 3-phase PWM generation, the T12 block offers options for individual compare and capture functions, as well as dead-time control and hysteresis-like compare mode.

This section provides information about:

- T12 overview (see **Section 18.2.1**)
- Counting scheme (see **Section 18.2.2**)
- Compare modes (see **Section 18.2.3**)
- Compare mode output path (see **Section 18.2.4**)
- Capture modes (see **Section 18.2.5**)
- Shadow transfer (see **Section 18.2.6**)
- T12 operating mode selection (see **Section 18.2.7**)
- T12 counter register description (see **Section 18.2.8**)

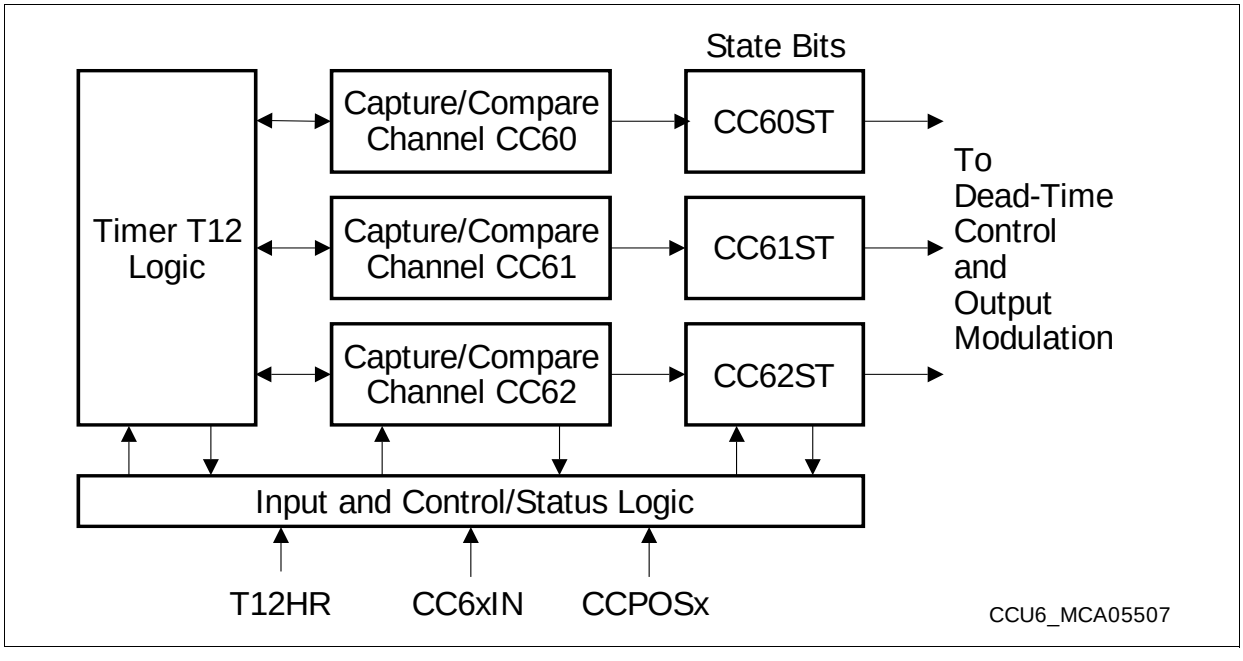


Figure 18-3 Overview Diagram of the Timer T12 Block

18.2.1 T12 Overview

Figure 18-4 shows a detailed block diagram of Timer T12. The functions of the timer T12 block are controlled by bits in registers **TCTR0**, **TCTR2**, and **PISELL**.

Timer T12 receives its input clock (f_{T12}) from the module clock f_{CC6} via a programmable prescaler and an optional 1/256 divider or from an input signal T12HR. These options are controlled via bit fields T12CLK and T12PRE (see **Table 18-2**). T12 can count up or down, depending on the selected operation mode. A direction flag, CDIR, indicates the current counting direction.

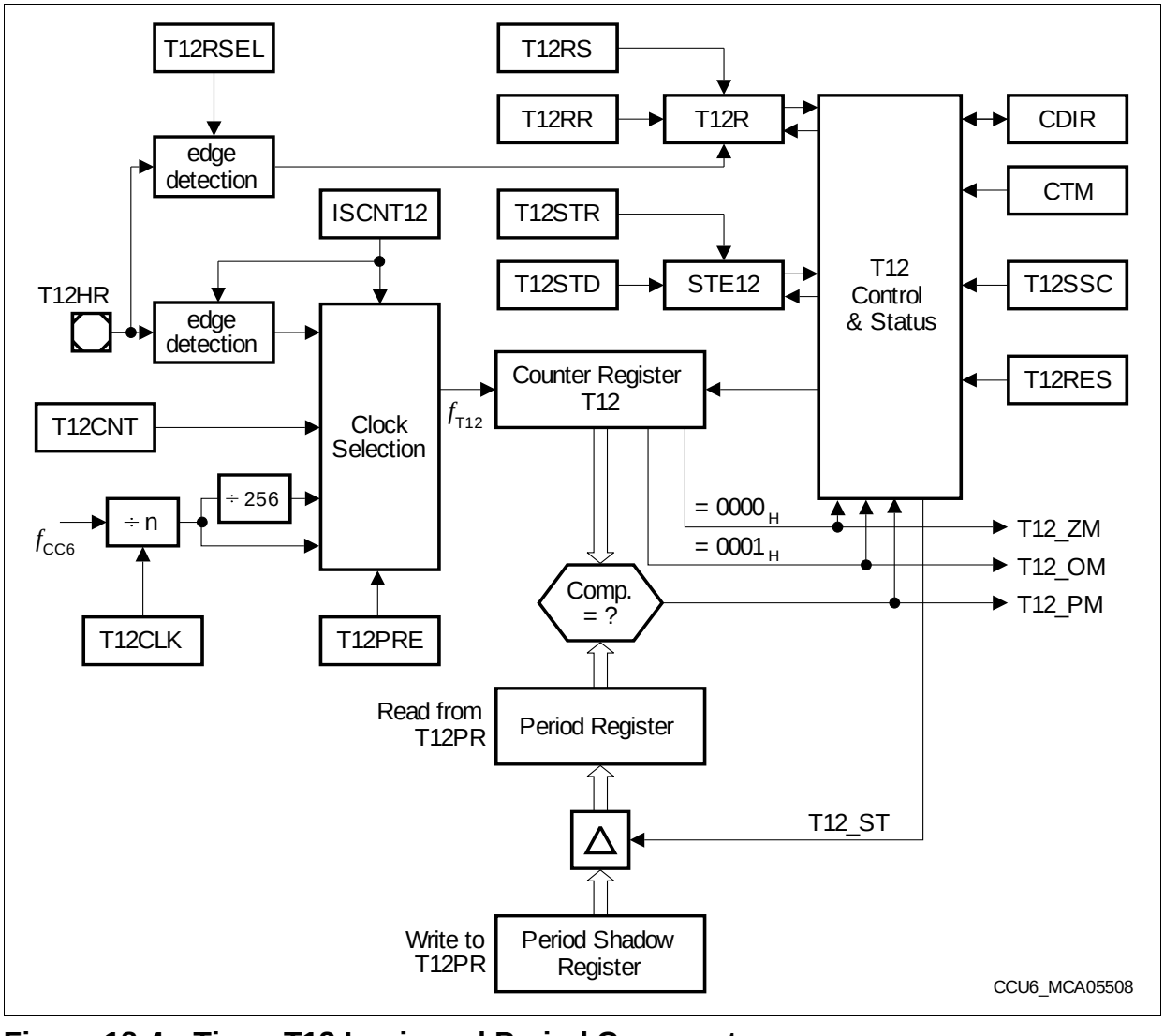


Figure 18-4 Timer T12 Logic and Period Comparators

Via a comparator, the T12 counter register **T12** is connected to a Period Register **T12PR**. This register determines the maximum count value for T12.
In Edge-Aligned mode, T12 is cleared to 0000_H after it has reached the period value defined by T12PR. In Center-Aligned mode, the count direction of T12 is set from ‘up’ to

'down' after it has reached the period value (please note that in this mode, T12 exceeds the period value by one before counting down). In both cases, signal T12_PM (T12 Period Match) is generated. The Period Register receives a new period value from its Shadow Period Register.
A read access to T12PR delivers the current period value at the comparator, whereas a write access targets the Shadow Period Register to prepare another period value. The transfer of a new period value from the Shadow Period Register into the Period Register (see **Section 18.2.6**) is controlled via the 'T12 Shadow Transfer' control signal, T12_ST. The generation of this signal depends on the operating mode and on the shadow transfer enable bit STE12. Providing a shadow register for the period value as well as for other values related to the generation of the PWM signal allows a concurrent update by software for all relevant parameters.
Two further signals indicate whether the counter contents are equal to 0000_H (T12_ZM = zero match) or 0001_H (T12_OM = one match). These signals control the counting and switching behavior of T12.

The basic operating mode of T12, either Edge-Aligned mode (**Figure 18-5**) or Center-Aligned mode (**Figure 18-6**), is selected via bit CTM. A Single-Shot control bit, T12SSC, enables an automatic stop of the timer when the current counting period is finished (see **Figure 18-7** and **Figure 18-8**).

The start or stop of T12 is controlled by the Run bit T12R that can be modified by bits in register **TCTR4**. The run bit can be set/cleared by software via the associated set/clear bits T12RS or T12RR, it can be set by a selectable edge of the input signal T12HR (**TCTR2**.T12RSEL), or it is cleared by hardware according to preselected conditions.
Timer T12 can be cleared via control bit T12RES. Setting this write-only bit does only clear the timer contents, but has no further effects, for example, it does not stop the timer. The timer T12 run bit T12R must not be set while the applied T12 period value is zero.

The generation of the T12 shadow transfer control signal, T12_ST, is enabled via bit STE12. This bit can be set or reset by software indirectly through its associated set/clear control bits T12STR and T12STD.
While Timer T12 is running, write accesses to the count register T12 are not taken into account. If T12 is stopped and the Dead-Time counters are 0, write actions to register T12 are immediately taken into account.

18.2.2 T12 Counting Scheme

This section describes the clocking and counting capabilities of T12.

18.2.2.1 Clock Selection

In **Timer Mode** (**PISELH**.ISCNT12 = 00_B), the input clock f_{T12} of Timer T12 is derived from the internal module clock f_{CC6} through a programmable prescaler and an optional 1/256 divider. The resulting prescaler factors are listed in **Table 18-2**. The prescaler of T12 is cleared while T12 is not running (**TCTR0**.T12R = 0) to ensure reproducible timings and delays.

Table 18-2 Timer T12 Input Frequency Options

T12CLK	Resulting Input Clock f_{T12} Prescaler Off (T12PRE = 0)	Resulting Input Clock f_{T12} Prescaler On (T12PRE = 1)
000_B	f_{CC6}	f_{CC6} / 256
001_B	f_{CC6} / 2	f_{CC6} / 512
010_B	f_{CC6} / 4	f_{CC6} / 1024
011_B	f_{CC6} / 8	f_{CC6} / 2048
100_B	f_{CC6} / 16	f_{CC6} / 4096
101_B	f_{CC6} / 32	f_{CC6} / 8192
110_B	f_{CC6} / 64	f_{CC6} / 16384
111_B	f_{CC6} / 128	f_{CC6} / 32768

In **Counter Mode**, timer T12 counts one step:

- If a 1 is written to **TCTR4**.T12CNT and **PISELH**.ISCNT12 = 01_B
- If a rising edge of input signal T12HR is detected and **PISELH**.ISCNT12 = 10_B
- If a falling edge of input signal T12HR is detected and **PISELH**.ISCNT12 = 11_B

18.2.2.2 Edge-Aligned / Center-Aligned Mode

In **Edge-Aligned Mode** (CTM = 0), timer T12 is always counting upwards (CDIR = 0). When reaching the value given by the period register (period-match T12_PM), the value of T12 is cleared with the next counting step (saw tooth shape).

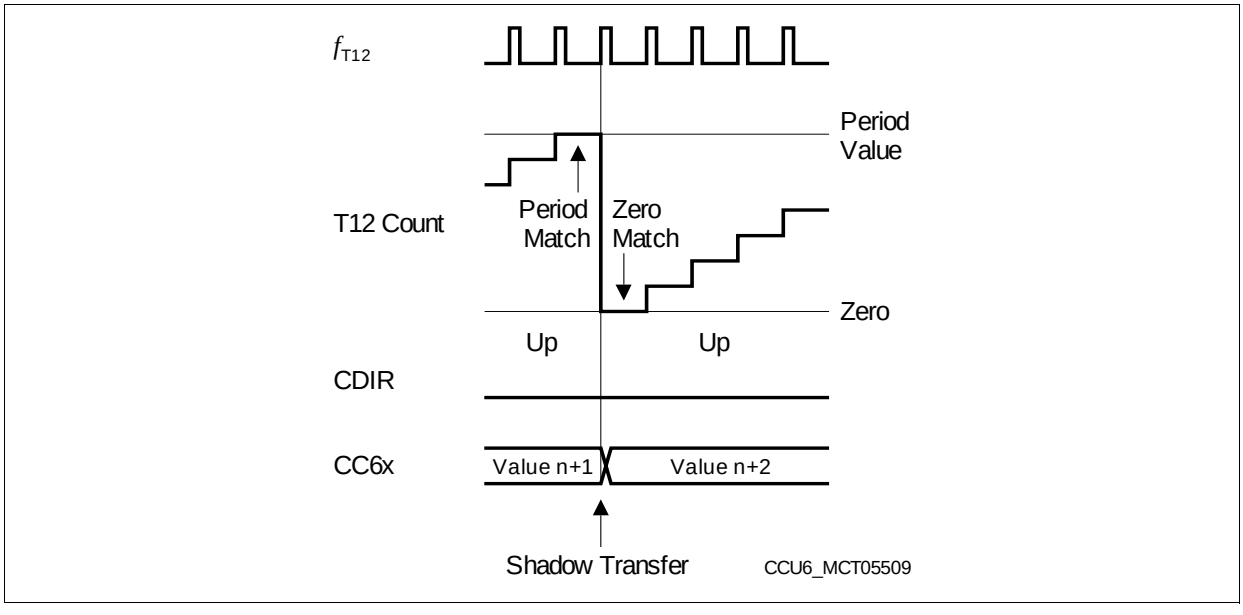


Figure 18-5 T12 Operation in Edge-Aligned Mode

As a result, in Edge-Aligned mode, the timer period is given by:

$T12_{PER}$ = <Period-Value> + 1; in T12 clocks (f_{T12}) (18.1)

In **Center-Aligned Mode** (CTM = 1), timer T12 is counting upwards or downwards (triangular shape). When reaching the value given by the period register (period-match T12_PM) while counting upwards (CDIR = 0), the counting direction control bit CDIR is changed to downwards (CDIR = 1) with the next counting step.
When reaching the value 0001_H (one-match T12_OM) while counting downwards, the counting direction control bit CDIR is changed to upwards with the next counting step.

As a result, in Center.Aligned mode, the timer period is given by:

$T12_{PER}$ = (<Period-Value> + 1) × 2; in T12 clocks (f_{T12}) (18.2)

- With the next clock event of f_{T12} the count direction is set to counting up (CDIR = 0) when the counter reaches 0001_H while counting down.
- With the next clock event of f_{T12} the count direction is set to counting down (CDIR = 1) when the Period-Match is detected while counting up.
- With the next clock event of f_{T12} the counter counts up while CDIR = 0 and it counts down while CDIR = 1.

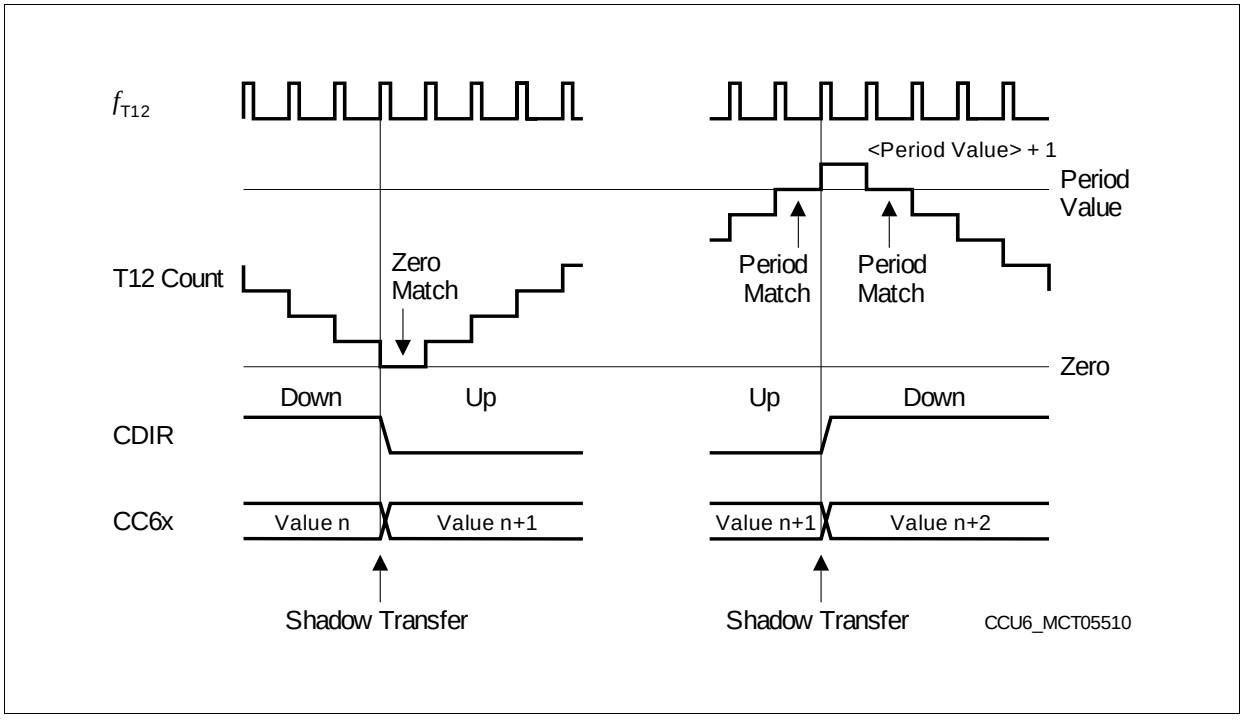


Figure 18-6 T12 Operation in Center-Aligned Mode

*Note: Bit CDIR changes with the next timer clock event after the one-match or the period-match. Therefore, the timer continues counting in the previous direction for one cycle before actually changing its direction (see **Figure 18-6**).*

18.2.2.3 Single-Shot Mode

In Single-Shot Mode, the timer run bit T12R is cleared by hardware. If bit T12SSC = 1, the timer T12 will stop when the current timer period is finished.

In Edge-Aligned mode, T12R is cleared when the timer becomes zero after having reached the period value (see **Figure 18-7**).

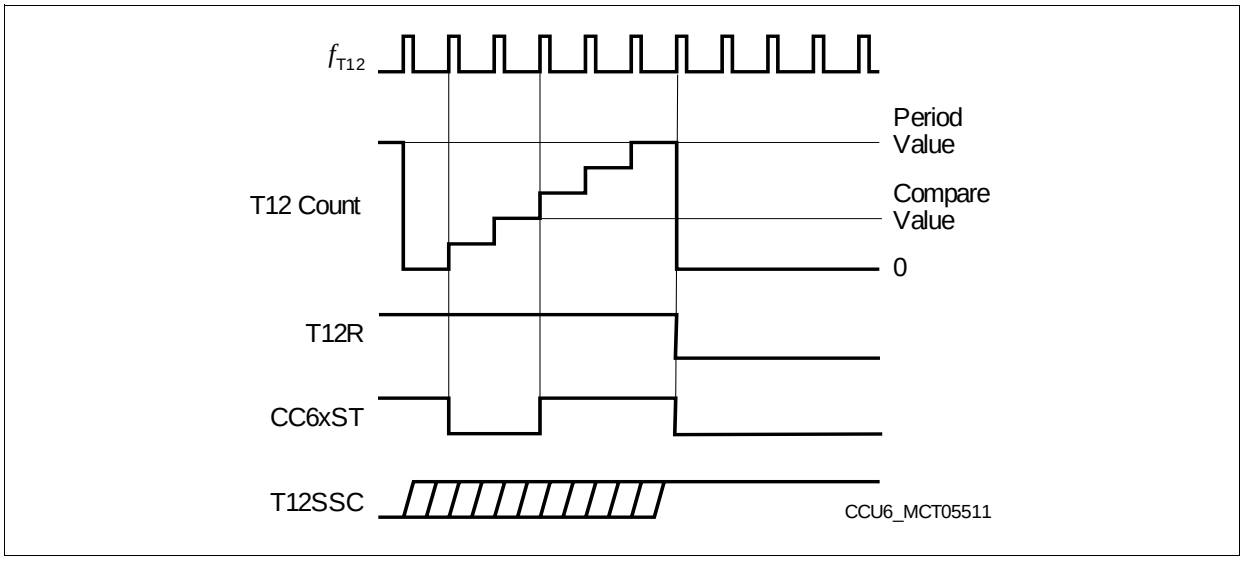


Figure 18-7 Single-Shot Operation in Edge-Aligned Mode

In Center-Aligned mode, the period is finished when the timer has counted down to zero (one clock cycle after the one-match while counting down, see **Figure 18-8**).

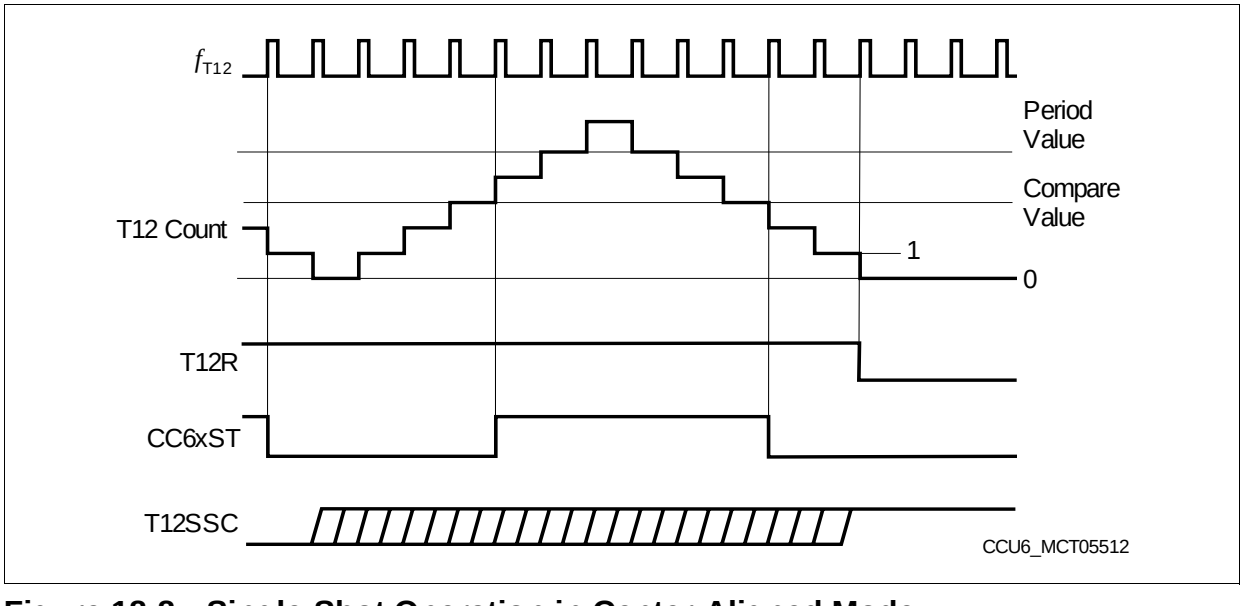


Figure 18-8 Single-Shot Operation in Center-Aligned Mode

18.2.3 T12 Compare Mode

Associated with Timer T12 are three individual capture/compare channels, that can perform compare or capture operations with regard to the contents of the T12 counter. The capture functions are explained in **Section 18.2.5**.

18.2.3.1 Compare Channels

In Compare Mode (see **Figure 18-9**), the three individual compare channels CC60 CC61, and CC62 can generate a three-phase PWM pattern.

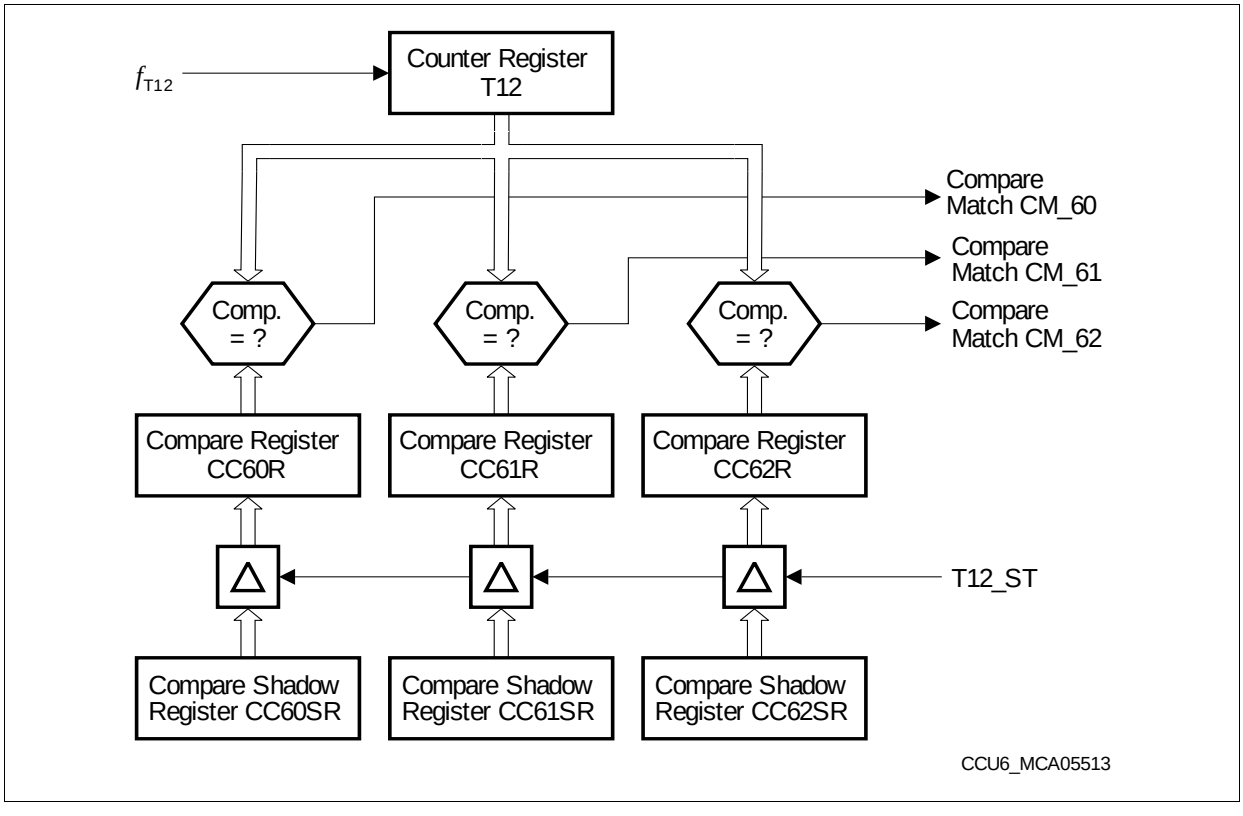


Figure 18-9 T12 Channel Comparators

Each compare channel is connected to the T12 counter register via its individual equal-to comparator, generating a match signal when the contents of the counter matches the contents of the associated compare register. Each channel consists of the comparator and a double register structure - the actual compare register CC6xR, feeding the comparator, and an associated shadow register CC6xSR, that is preloaded by software and transferred into the compare register when signal T12 shadow transfer, T12_ST, gets active. Providing a shadow register for the compare value as well as for other values related to the generation of the PWM signal facilitates a concurrent update by software for all relevant parameters of a three-phase PWM.

18.2.3.2 Channel State Bits

Associated with each (compare) channel is a State Bit, **CMPSTAT**.CC6xST, holding the status of the compare (or capture) operation (see **Figure 18-10**). In compare mode, the State Bits are modified according to a set of switching rules, depending on the current status of timer T12.

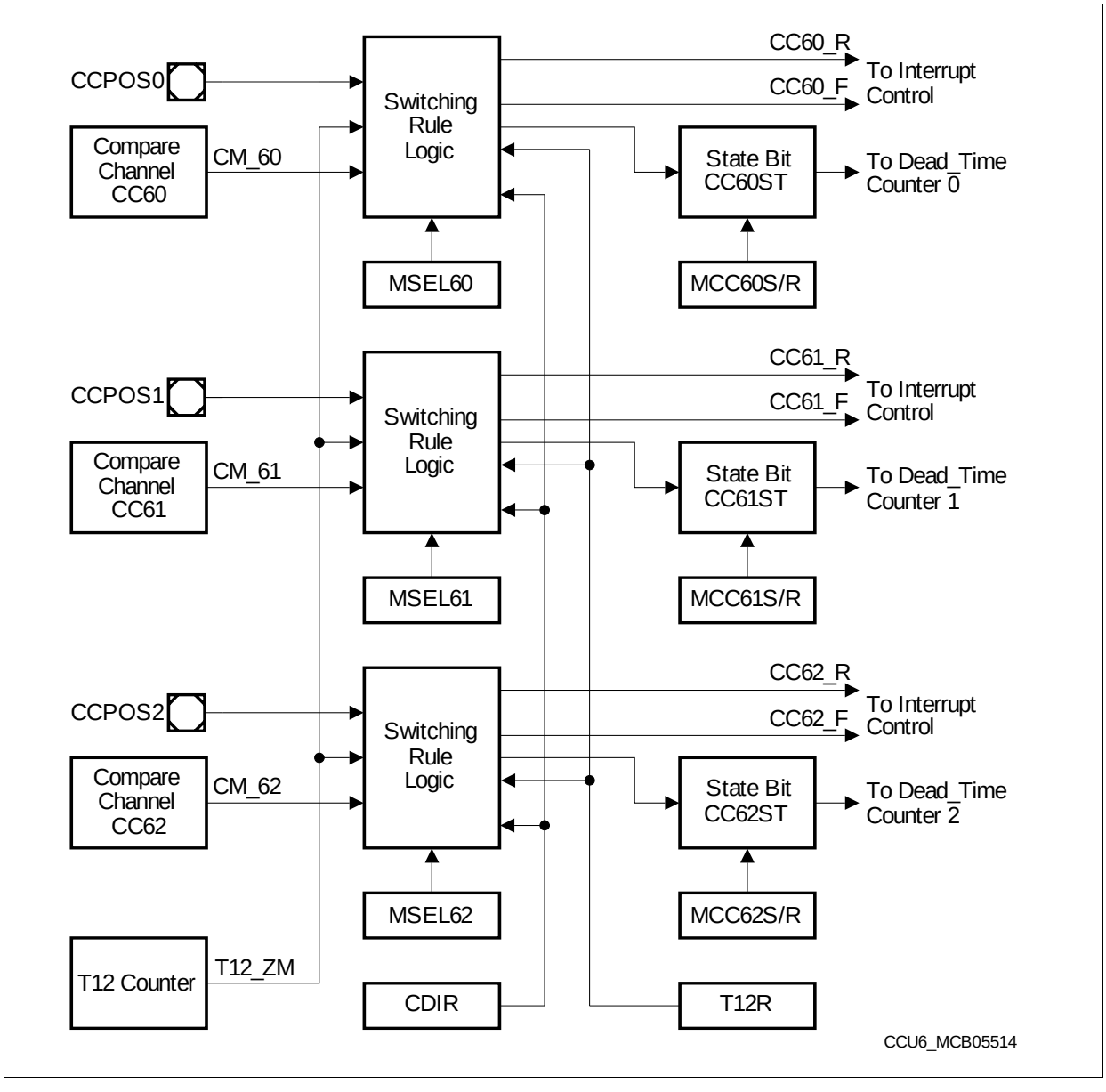


Figure 18-10 Compare State Bits for Compare Mode

The inputs to the switching rule logic for the CC6xST bits are the timer direction (CDIR), the timer run bit (T12R), the timer T12 zero-match signal (T12_ZM), and the actual individual compare-match signals CM_6x as well as the mode control bits, **T12MSEL**.MSEL6x.

In addition, each state bit can be set or cleared by software via the appropriate set and reset bits in register **CMPMODIF**, MCC6xS and MCC6xR. The input signals CCPOSx are used in hysteresis-like compare mode, whereas in normal compare mode, these inputs are ignored.

Note: In Hall Sensor, single shot or capture modes, additional/different rules are taken into account (see related sections).

A compare interrupt event CC6x_R is signaled when a compare match is detected while counting upwards, whereas the compare interrupt event CC6x_F is signaled when a compare match is detected while counting down. The actual setting of a State Bit has no influence on the interrupt generation in compare mode.

A modification of a State Bit CC6xST by the switching rule logic due to a compare action is only possible while Timer T12 is running (T12R = 1). If this is the case, the following switching rules apply for setting and clearing the State Bits in Compare Mode (illustrated in **Figure 18-11** and **Figure 18-12**):

A State Bit **CC6xST is set** to 1:

- with the next T12 clock (f_{T12}) after a compare-match when T12 is counting up (i.e., when the counter is incremented above the compare value);
- with the next T12 clock (f_{T12}) after a zero-match AND a parallel compare-match when T12 is counting up.

A State Bit **CC6xST is cleared** to 0:

- with the next T12 clock (f_{T12}) after a compare-match when T12 is counting down (i.e., when the counter is decremented below the compare value in center-aligned mode);
- with the next T12 clock (f_{T12}) after a zero-match AND NO parallel compare-match when T12 is counting up.

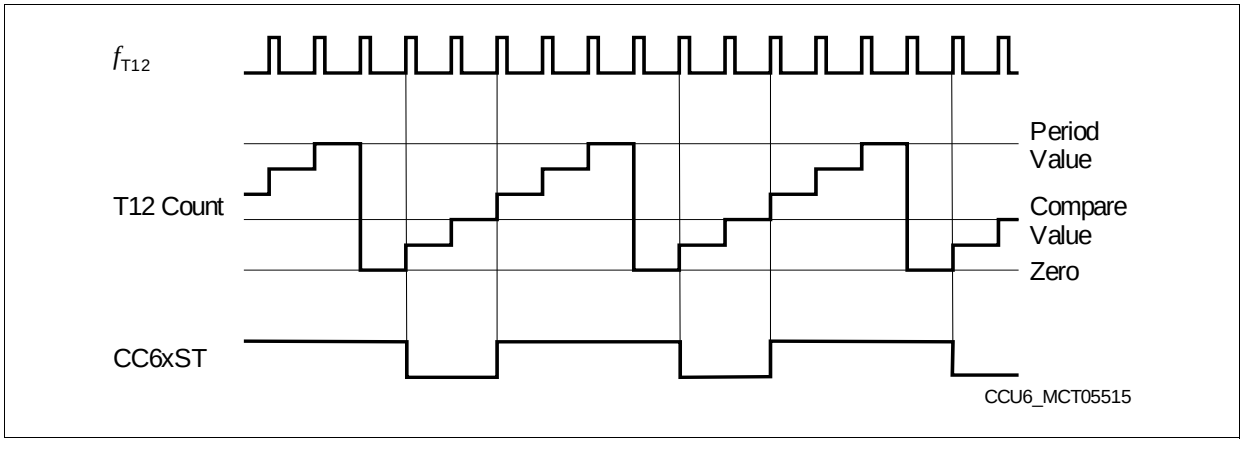


Figure 18-11 Compare Operation, Edge-Aligned Mode

Figure 18-13 illustrates some more examples for compare waveforms. It is important to note that in these examples, it is assumed that some of the compare values are changed

while the timer is running. This change is performed via a software preload of the Shadow Register, CC6xSR. The value is transferred to the actual Compare Register CC6xR with the T12 Shadow Transfer signal, T12_ST, that is assumed to be enabled.

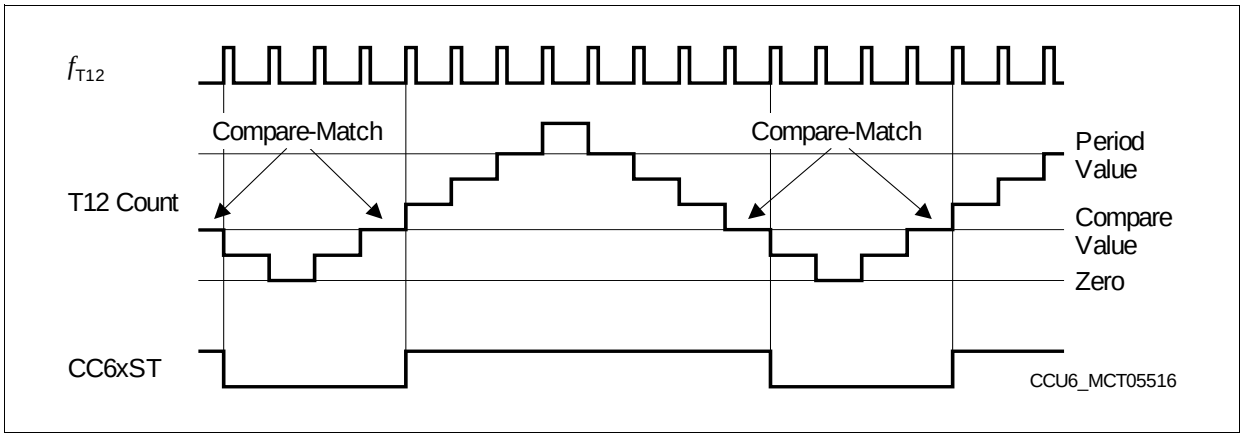


Figure 18-12 Compare Operation, Center-Aligned Mode

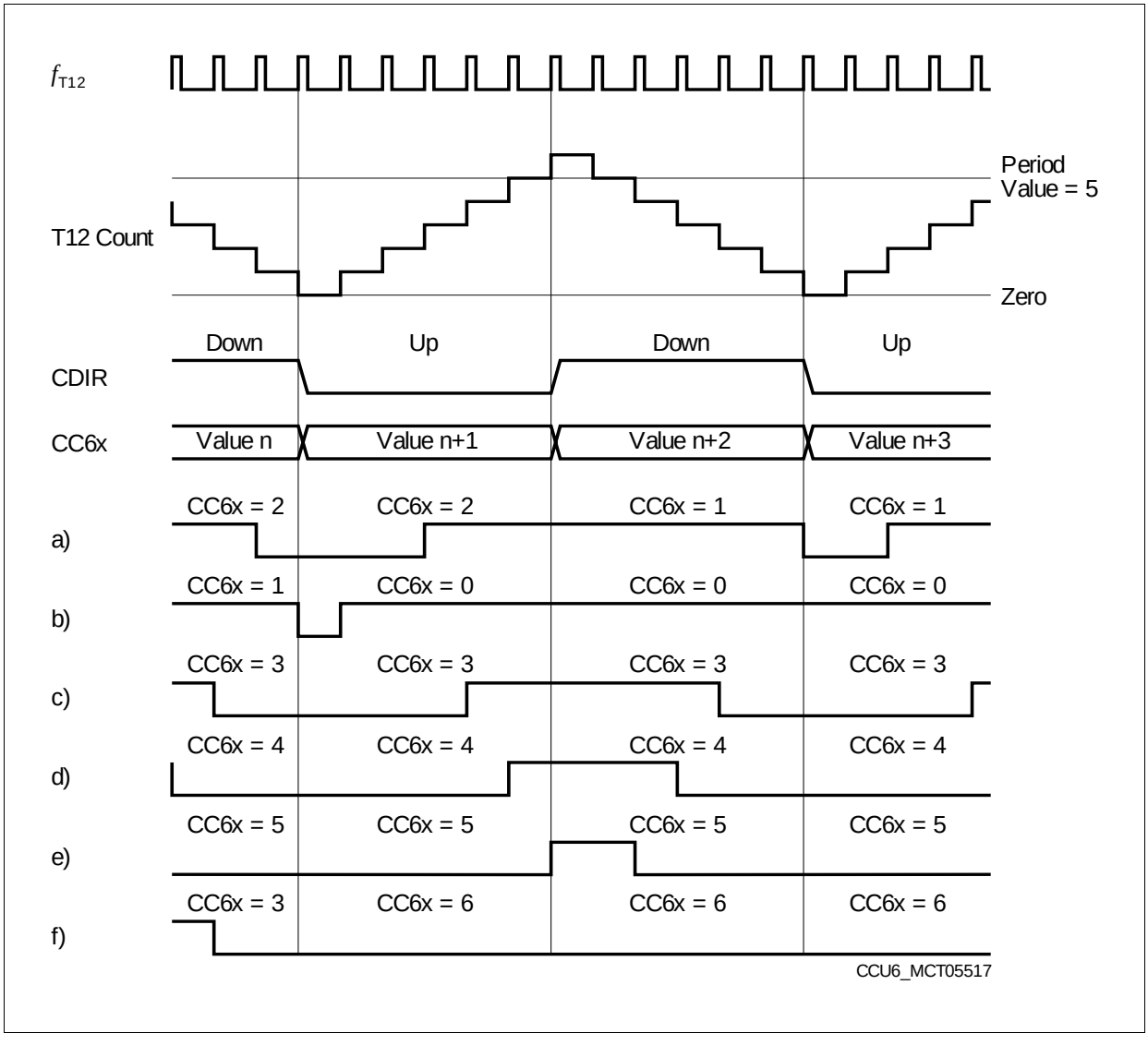


Figure 18-13 Compare Waveform Examples

Example b) illustrates the transition to a duty cycle of 100%. First, a compare value of 0001_H is used, then changed to 0000_H. Please note that a low pulse with the length of one T12 clock is still produced in the cycle where the new value 0000_H is in effect; this pulse originates from the previous value 0001_H. In the following timer cycles, the State Bit CC6xST remains at 1, producing a 100% duty cycle signal. In this case, the compare rule ‘zero-match AND compare-match’ is in effect.

Example f) shows the transition to a duty cycle of 0%. The new compare value is set to <Period-Value> + 1, and the State Bit CC6ST remains cleared.

Figure 18-14 illustrates an example for the waveforms of all three channels. With the appropriate dead-time control and output modulation, a very efficient 3-phase PWM signal can be generated.

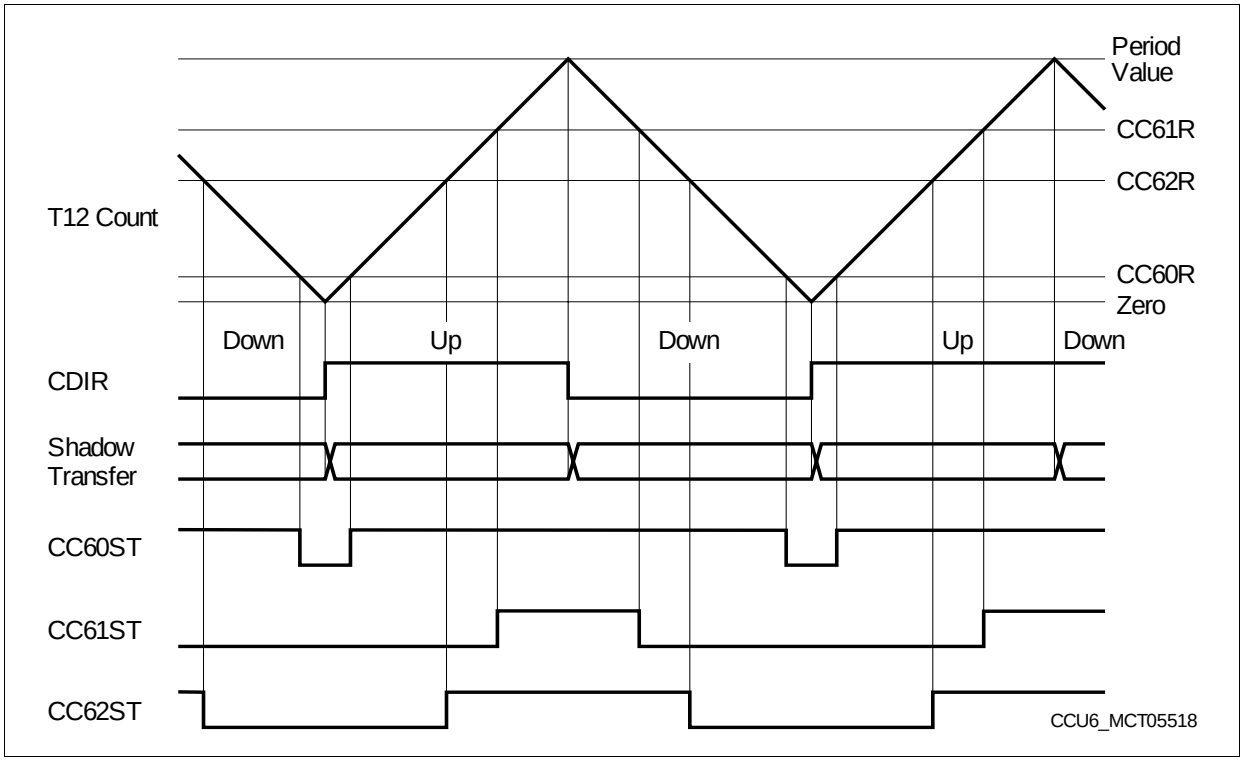


Figure 18-14 Three-Channel Compare Waveforms

18.2.3.3 Hysteresis-Like Control Mode

The hysteresis-like control mode (**T12MSEL**.MSEL6x = 1001_B) offers the possibility to switch off the PWM output if the input CCPOSx becomes 0 by clearing the State Bit CC6xST. This can be used as a simple motor control feature by using a comparator indicating, e.g., overcurrent. While CCPOSx = 0, the PWM outputs of the corresponding channel are driving their passive levels, because the setting of bit CC6xST is only possible while CCPOSx = 1.
As long as input CCPOSx is 0, the corresponding State Bit is held 0. When CCPOSx is at high level, the outputs can be in active state and are determined by bit CC6xST (see **Figure 18-10** for the state bit logic and **Figure 18-15** for the output paths).
The CCPOSx inputs are evaluated with f_{CC6}.

This mode can be used to introduce a timing-related behavior to a hysteresis controller. A standard hysteresis controller detects if a value exceeds a limit and switches its output according to the compare result. Depending on the operating conditions, the switching frequency and the duty cycle are not fixed, but change permanently.

If (outer) time-related control loops based on a hysteresis controller in an inner loop should be implemented, the outer loops show a better behavior if they are synchronized to the inner loops. Therefore, the hysteresis-like mode can be used, that combines timer-related switching with a hysteresis controller behavior. For example, in this mode, an output can be switched on according to a fixed time base, but it is switched off as soon as a falling edge is detected at input CCPOSx.

This mode can also be used for standard PWM with overcurrent protection. As long as there is no low level signal at pin CCPOSx, the output signals are generated in the normal manner as described in the previous sections. Only if input CCPOSx shows a low level, e.g. due to the detection of overcurrent, the outputs are shut off to avoid harmful stress to the system.

18.2.4 Compare Mode Output Path

Figure 18-15 gives an overview on the signal path from a channel State Bit to its output pin in its simplest form. As illustrated, a user has a variety of controls to determine the desired output signal switching behavior in relation to the current state of the State Bit, CC6xST. Please refer to **Section 18.2.4.3** for details on the output modulation.

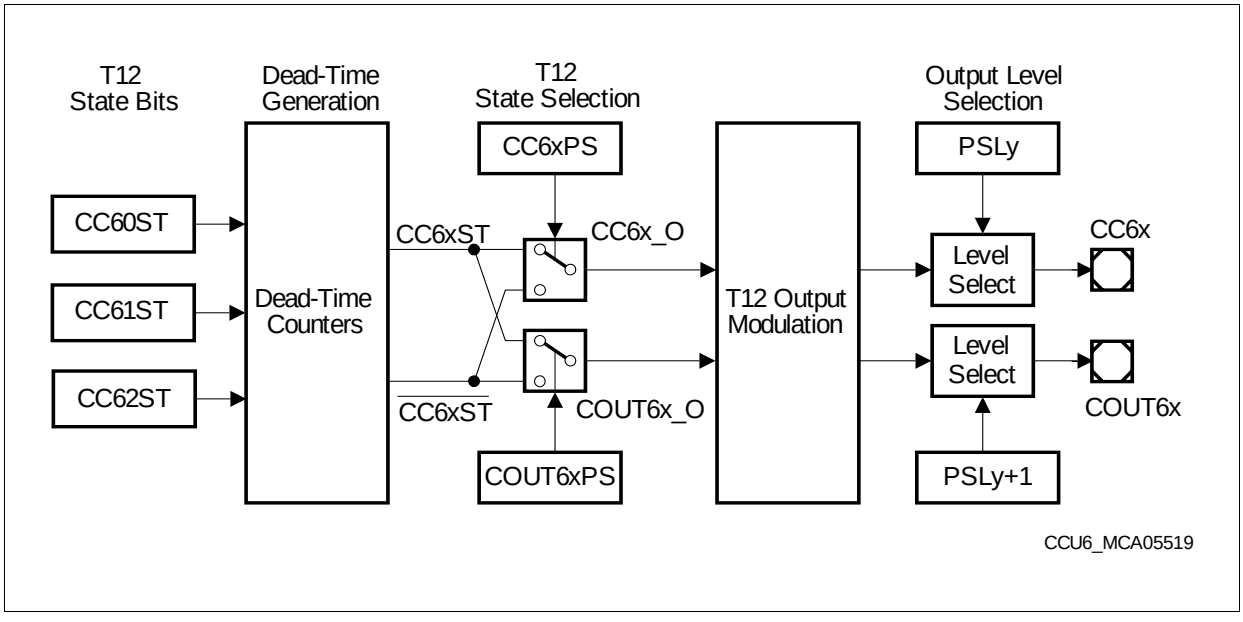


Figure 18-15 Compare Mode Simplified Output Path Diagram

The output path is based on signals that are defined as active or passive. The terms active and passive are not related to output levels, but to internal actions. This mainly applies for the modulation, where T12 and T13 signals are combined with the multi-channel signals and the trap function. The Output level Selection allows the user to define the output level at the output pin for the passive state (inverted level for the active state). It is recommended to configure this block in a way that an external power switch is switched off while the CCU6 delivers an output signal in the passive state.

18.2.4.1 Dead-Time Generation

The generation of (complementary) signals for the high-side and the low-side switches of one power inverter phase is based on the same compare channel. For example, if the high-side switch should be active while the T12 counter value is above the compare value (State Bit = 1), then the low-side switch should be active while the counter value is below the compare value (State Bit = 0).

In most cases, the switching behavior of the connected power switches is not symmetrical concerning the switch-on and switch-off times. A general problem arises if the time for switch-on is smaller than the time for switch-off of the power device. In this case, a short-circuit can occur in the inverter bridge leg, which may damage the complete system. In order to solve this problem by HW, this capture/compare unit

contains a programmable Dead-Time Generation Block, that delays the passive to active edge of the switching signals by a programmable time (the active to passive edge is not delayed).

The Dead-Time Generation Block, illustrated in **Figure 18-16**, is built in a similar way for all three channels of T12. It is controlled by bits in register **T12DTC**. Any change of a CC6xST State Bit activates the corresponding Dead-Time Counter, that is clocked with the same input clock as T12 (f_{T12}). The length of the dead-time can be programmed by bit field DTM. This value is identical for all three channels. Writing **TCTR4**.DTRES = 1 sets all dead-times to passive.

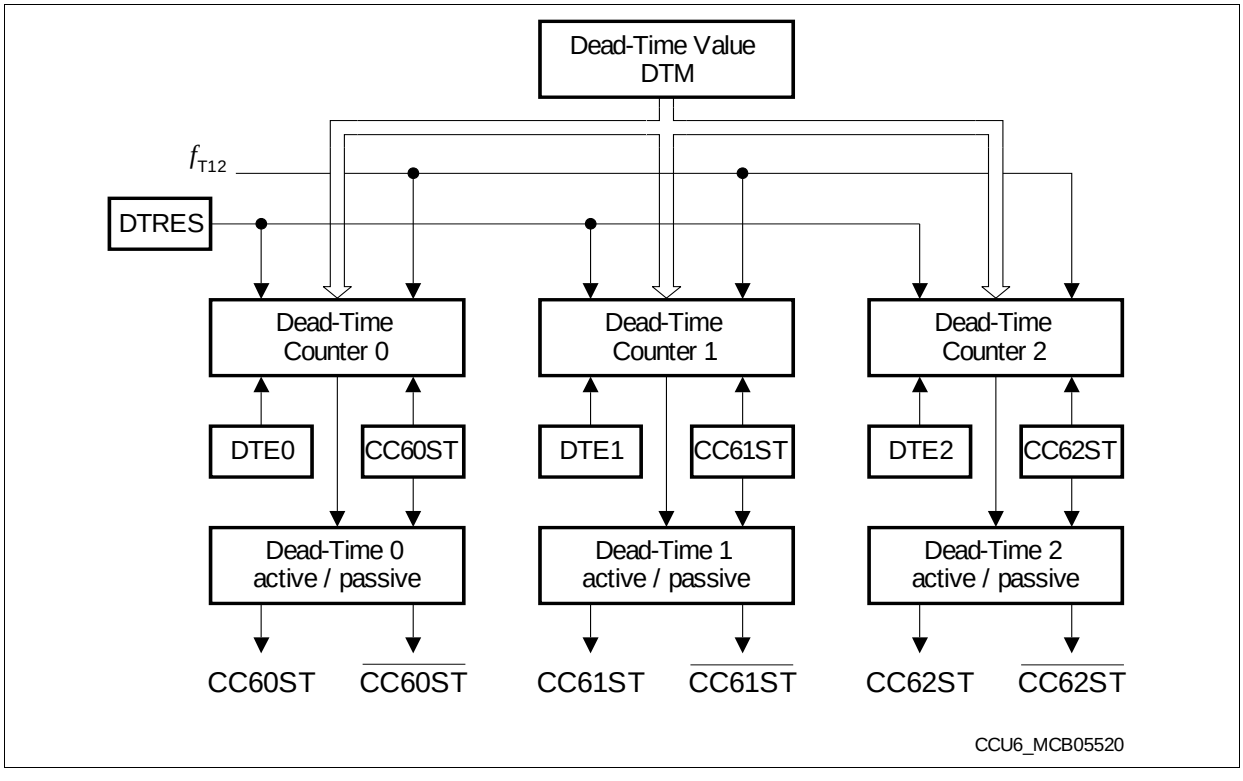


Figure 18-16 Dead-Time Generation Block Diagram

Each of the three dead-time counters has its individual dead-time enable bit, DTEx. An enabled dead-time counter generates a dead-time delaying the passive-to-active edge of the channel output signal. The change in a State Bit CC6xST is not taken into account while the dead-time generation of this channel is currently in progress (active). This avoids an unintentional additional dead-time if a State Bit CC6xST changes too early.
A disabled dead-time counter is always considered as passive and does not delay any edge of CC6xST.
Based on the State Bits CC6xST, the Dead-Time Generation Block outputs a direct signal CC6xST and an inverted signal $\overline{CC6xST}$ for each compare channel, each masked with the effect of the related Dead-Time Counters (waveforms illustrated in **Figure 18-17**).

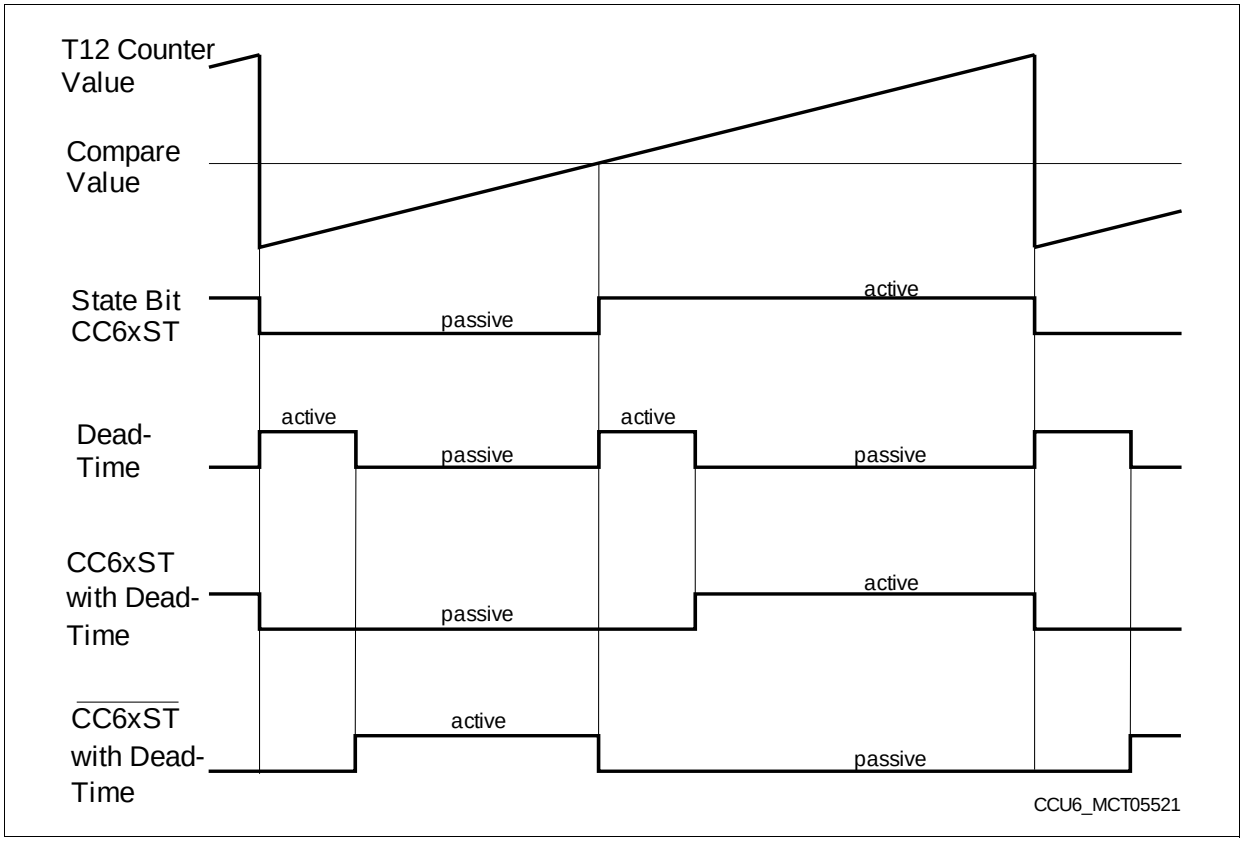


Figure 18-17 Dead-Time Generation Waveforms

18.2.4.2 State Selection

To support a wide range of power switches and drivers, the state selection offers the flexibility to define when a an output can be active and can be modulated, especially useful for **complementary or multi-phase PWM** signals.

The state selection is based on the signals CC6xST and $\overline{\text{CC6xST}}$ delivered by the dead-time generator (see **Figure 18-15**). Both signals are never active at the same time, but can be passive at the same time. This happens during the dead-time of each compare channel after a change of the corresponding State Bit CC6xST.

The user can select independently for each output signal CC6xO and COUT6xO if it should be active before or after the compare value has been reached (see register **CMPSTAT**). With this selection, the active (conducting) phases of complementary power switches in a power inverter bridge leg can be positioned with respect to the compare value (e.g. signal CC6xO can be active before, whereas COUT6xO can be active after the compare value is reached). Like this, the output modulation, the trap logic and the output level selection can be programmed independently for each output signal, although two output signals are referring to the same compare channel.

18.2.4.3 Output Modulation and Level Selection

The last block of the data path is the Output Modulation block. Here, all the modulation sources and the trap functionality are combined and control the actual level of the output pins (controlled by the modulation enable bits T1xMODENy and MCMEN in register **MODCTR**). The following signal sources can be combined here **for each T12 output signal** (see **Figure 18-18** for compare channel CC60):

- A **T12 related compare signal** CC6x_O (for outputs CC6x) or COUT6x_O (for outputs COUT6x) delivered by the T12 block (state selection with dead-time) with an individual enable bit T12MODENy per output signal (y = 0, 2, 4 for outputs CC6x and y = 1, 3, 5 for outputs COUT6x)
- The **T13 related compare signal** CC63_O delivered by the T13 state selection with an individual enable bit T13MODENy per output signal (y = 0, 2, 4 for outputs CC6x and y = 1, 3, 5 for outputs COUT6x)
- A **multi-channel output signal** MCMPy (y = 0, 2, 4 for outputs CC6x and y = 1, 3, 5 for outputs COUT6x) with a common enable bit MCMEN
- The **trap state** TRPS with an individual enable bit TRPENy per output signal (y = 0, 2, 4 for outputs CC6x and y = 1, 3, 5 for outputs COUT6x)

If one of the modulation input signals CC6x_O/COUT6x_O, CC63_O, or MCMPy of an output modulation block is enabled and is at passive state, the modulated is also in passive state, regardless of the state of the other signals that are enabled. Only if all enabled signals are in active state the modulated output shows an active state. If no modulation input is enabled, the output is in passive state.
If the Trap State is active (TRPS = 1), then the outputs that are enabled for the trap signal (by TRPENy = 1) are set to the passive state.

The output of each of the modulation control blocks is connected to a level select block that is configured by register **PSLR**. It offers the option to determine the actual output level of a pin, depending on the state of the output line (decoupling of active/passive state and output polarity) as specified by the Passive State Select bit PSLy. If the modulated output signal is in the passive state, the level specified directly by PSLy is output. If it is in the active state, the inverted level of PSLy is output. This allows the user to adapt the polarity of an active output signal to the connected circuitry.

The PSLy bits have shadow registers to allow for updates without undesired pulses on the output lines. The bits related to CC6x and COUT6x (x = 0, 1, 2) are updated with the T12 shadow transfer signal (T12_ST). A read action returns the actually used values, whereas a write action targets the shadow bits. Providing a shadow register for the PSL value as well as for other values related to the generation of the PWM signal facilitates a concurrent update by software for all relevant parameters.

Figure 18-18 shows the output modulation structure for compare channel CC60 (output signals CC60 and COUT60). A similar structure is implemented for the other two compare channels CC61 and CC62.

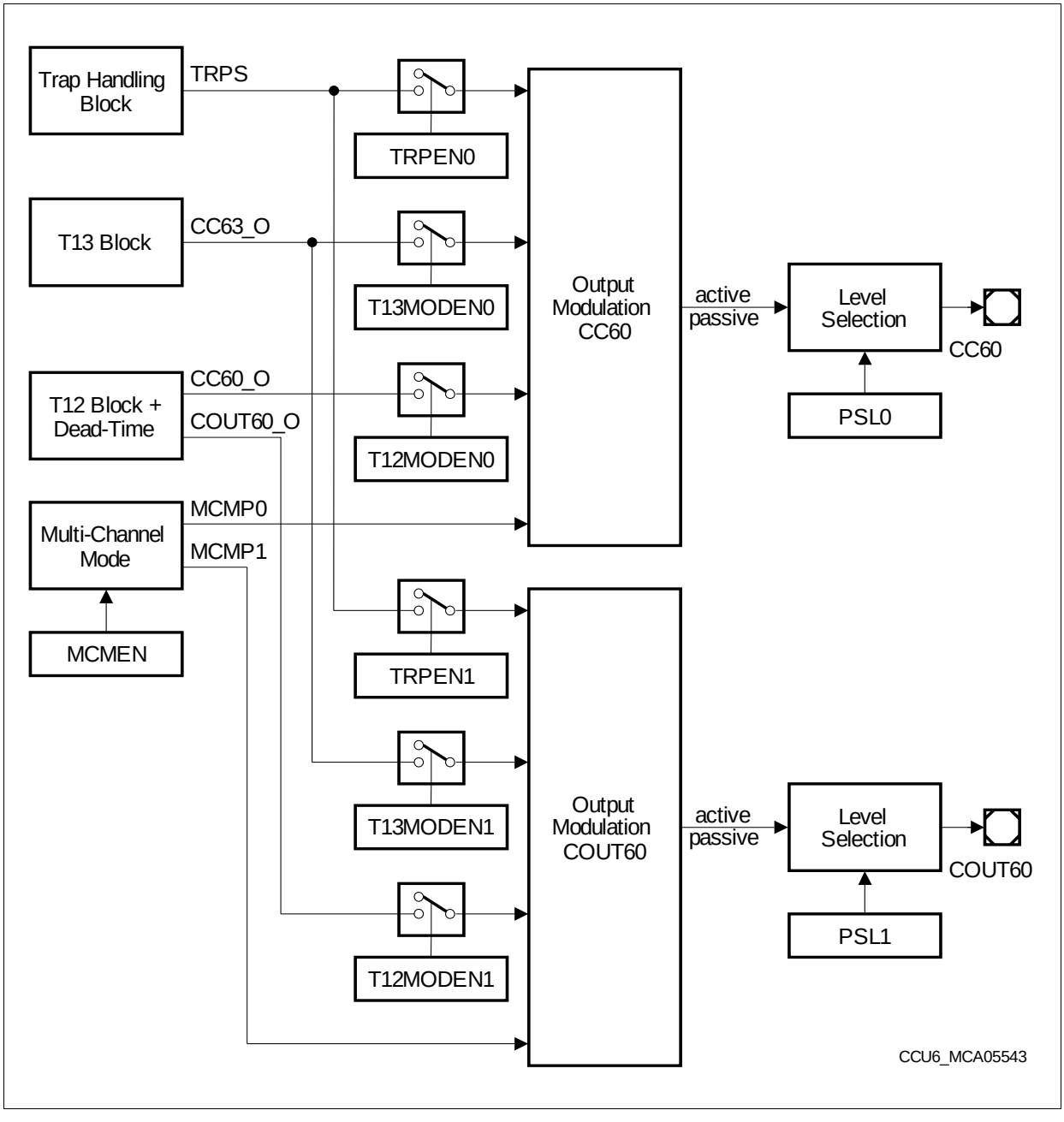


Figure 18-18 Output Modulation for Compare Channel CC60

18.2.5 T12 Capture Modes

Each of the three channels of the T12 Block can also be used to capture T12 time information in response to an external signal CC6xIN.
In capture mode, the interrupt event CC6x_R is detected when a rising edge is detected at the input CC6xIN, whereas the interrupt event CC6x_F is detected when a falling edge is detected.

There are a number of different modes for capture operation. In all modes, both of the registers of a channel are used. The selection of the capture modes is done via the **T12MSEL**.MSEL6x bit fields and can be selected individually for each of the channels.

Table 18-3 Capture Modes Overview

MSEL6x	Mode	Signal	Active Edge	CC6nSR Stored in	T12 Stored in
0100_B	1	CC6xIN	Rising	–	CC6xR
		CC6xIN	Falling	–	CC6xSR
0101_B	2	CC6xIN	Rising	CC6xR	CC6xSR
0110_B	3	CC6xIN	Falling	CC6xR	CC6xSR
0111_B	4	CC6xIN	Any	CC6xR	CC6xSR

Figure 18-19 illustrates **Capture Mode 1**. When a rising edge (0-to-1 transition) is detected at the corresponding input signal CC6xIN, the current contents of Timer T12 are captured into register CC6xR. When a falling edge (1-to-0 transition) is detected at the input signal CC6xIN, the contents of Timer T12 are captured into register CC6xSR.

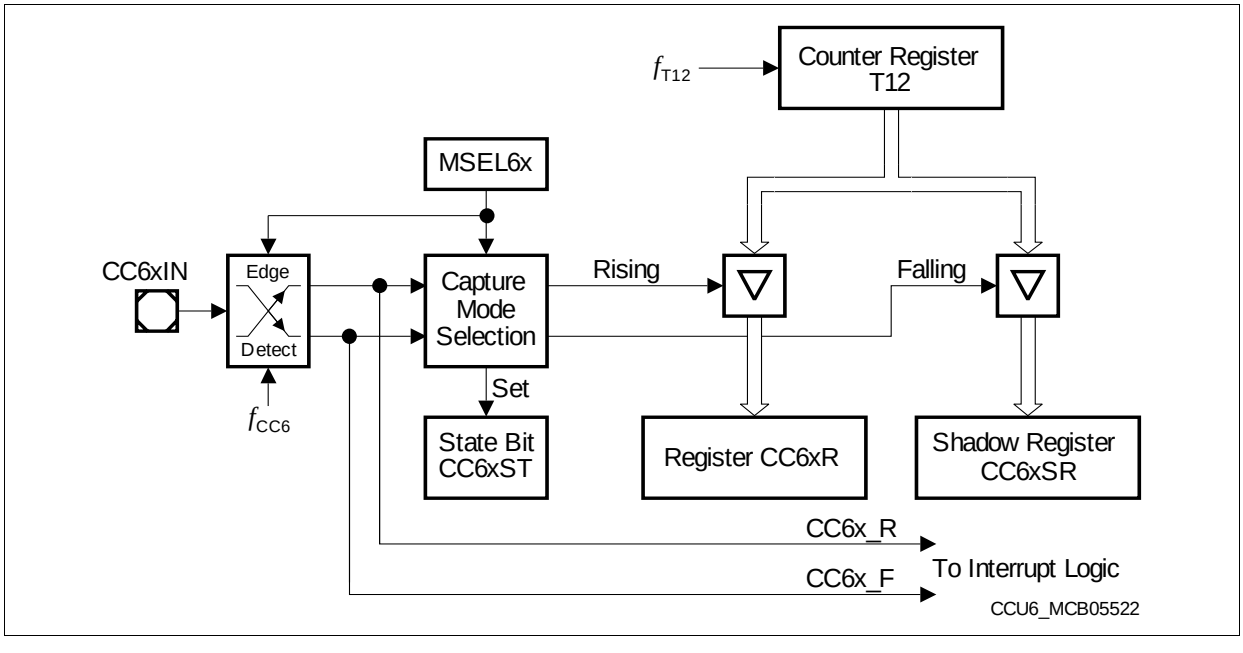


Figure 18-19 Capture Mode 1 Block Diagram

Capture Modes 2, 3 and 4 are shown in **Figure 18-20**. They differ only in the active edge causing the capture operation. In each of the three modes, when the selected edge is detected at the corresponding input signal CC6xIN, the current contents of the shadow register CC6xSR are transferred into register CC6xR, and the current Timer T12 contents are captured in register CC6xSR (simultaneous transfer). The active edge is a rising edge of CC6xIN for Capture Mode 2, a falling edge for Mode 3, and both, a rising or a falling edge for Capture Mode 4, as shown in **Table 18-3**. These capture modes are very useful in cases where there is little time between two consecutive edges of the input signal.

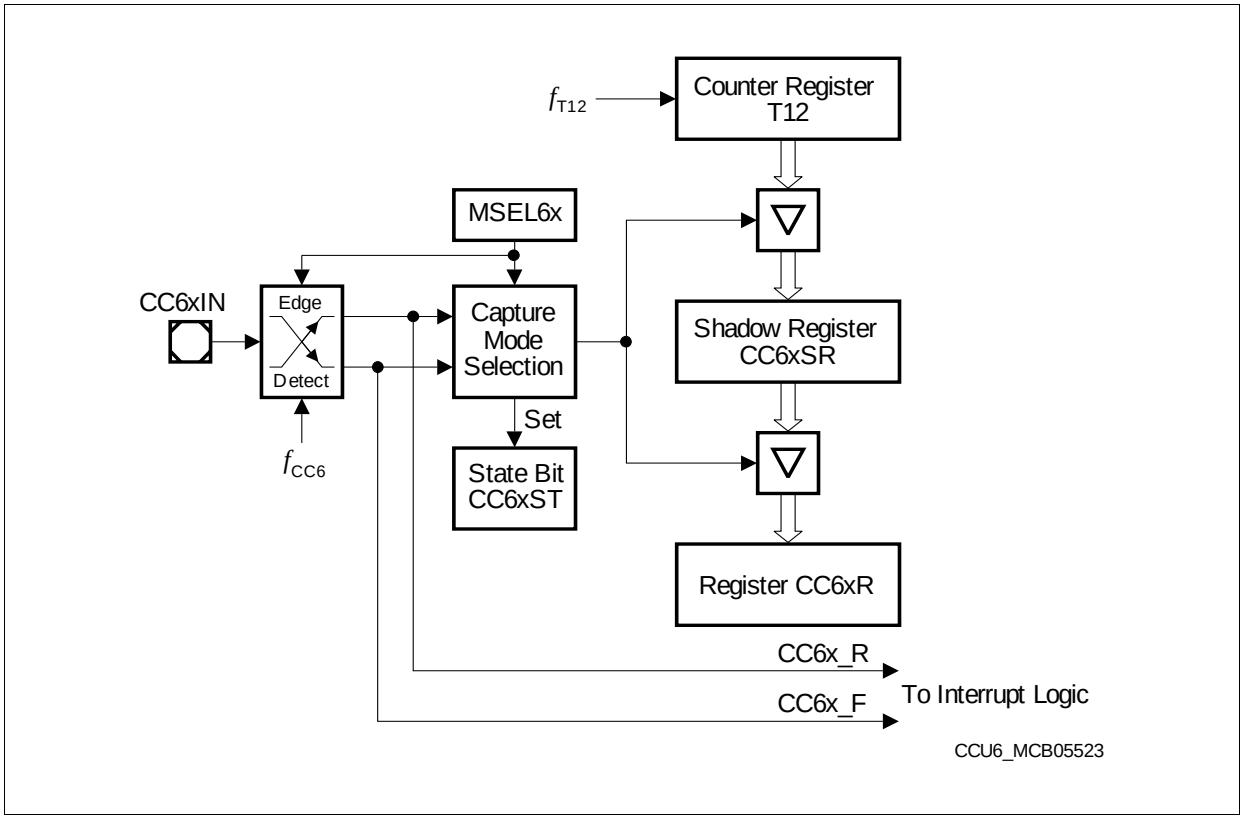


Figure 18-20 Capture Modes 2, 3 and 4 Block Diagram

Five further capture modes are called **Multi-Input Capture Modes**, as they use two different external inputs, signal CC6xIN and signal CCPOSx.

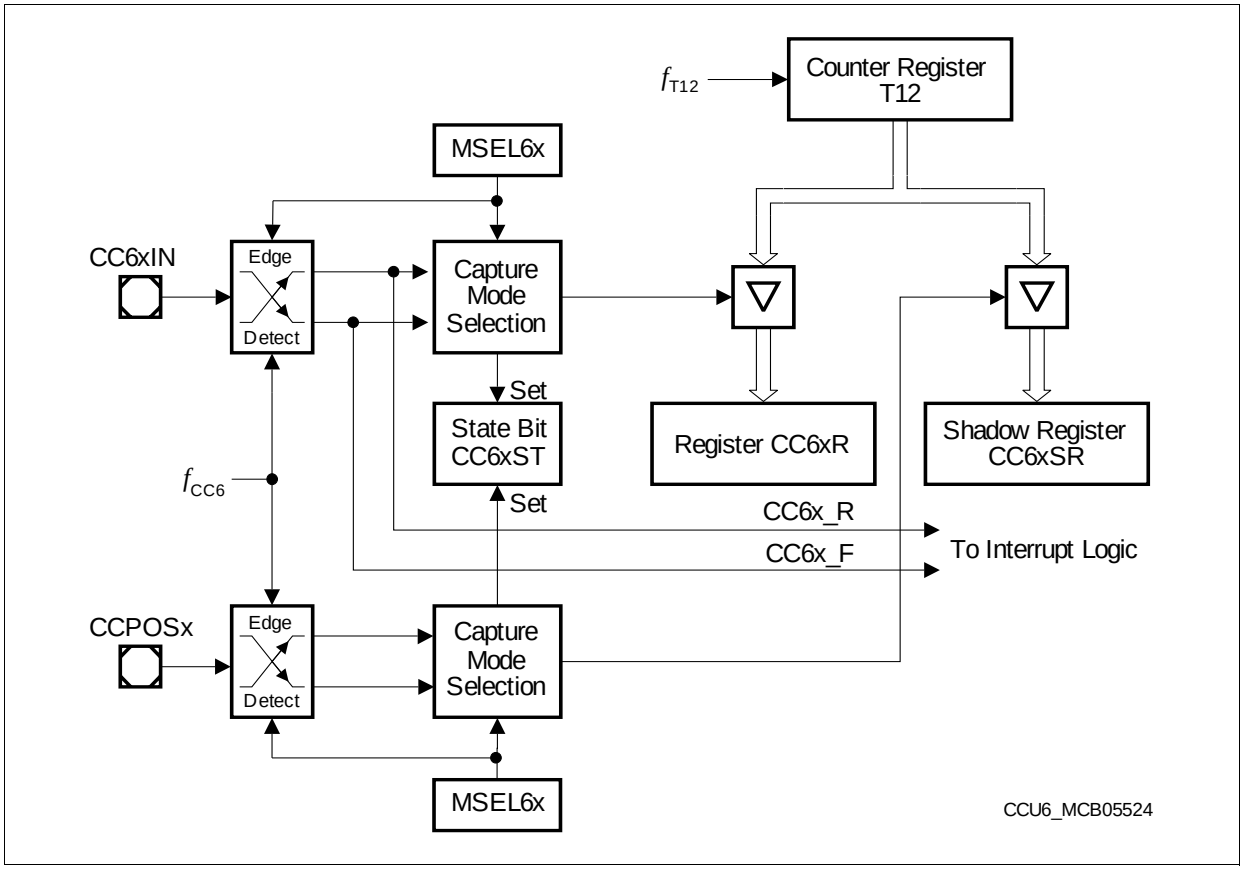


Figure 18-21 Multi-Input Capture Modes Block Diagram

In each of these modes, the current T12 contents are captured in register CC6xR in response to a selected event at signal CC6xIN, and in register CC6xSR in response to a selected event at signal CCPOSx. The possible events can be opposite input transitions, or the same transitions, or any transition at the two inputs. The different options are detailed in **Table 18-4**.

In each of the various capture modes, the Channel State Bit, CC6xST, is set to 1 when the selected capture trigger event at signal CC6xIN or CCPOSx has occurred. The State Bit is not cleared by hardware, but can be cleared by software.

In addition, appropriate signal lines to the interrupt logic are activated, that can generate an interrupt request to the CPU. Regardless of the selected active edge, all edges detected at signal CC6xIN can lead to the activation of the appropriate interrupt request line (see also **Section 18.8**).

Table 18-4 Multi-Input Capture Modes Overview

MSEL6x	Mode	Signal	Active Edge	T12 Stored in
1010_B	5	CC6xIN	Rising	CC6xR
		CCPOSx	Falling	CC6xSR
1011_B	6	CC6xIN	Falling	CC6xR
		CCPOSx	Rising	CC6xSR
1100_B	7	CC6xIN	Rising	CC6xR
		CCPOSx	Rising	CC6xSR
1101_B	8	CC6xIN	Falling	CC6xR
		CCPOSx	Falling	CC6xSR
1110_B	9	CC6xIN	Any	CC6xR
		CCPOSx	Any	CC6xSR
1111_B	–	reserved (no capture or compare action)		

18.2.6 T12 Shadow Register Transfer

A special shadow transfer signal (T12_ST) can be generated to facilitate updating the period and compare values of the compare channels CC60, CC61, and CC62 synchronously to the operation of T12. Providing a shadow register for values defining one PWM period facilitates a concurrent update by software for all relevant parameters. The next PWM period can run with a new set of parameters. The generation of this signal is requested by software via bit **TCTR0**.STE12 (set by writing 1 to the write-only bit **TCTR4**.T12STR, cleared by writing 1 to the write-only bit **TCTR4**.T12STD).

Figure 18-22 shows the shadow register structure and the shadow transfer signals, as well as on the read/write accessibility of the various registers.

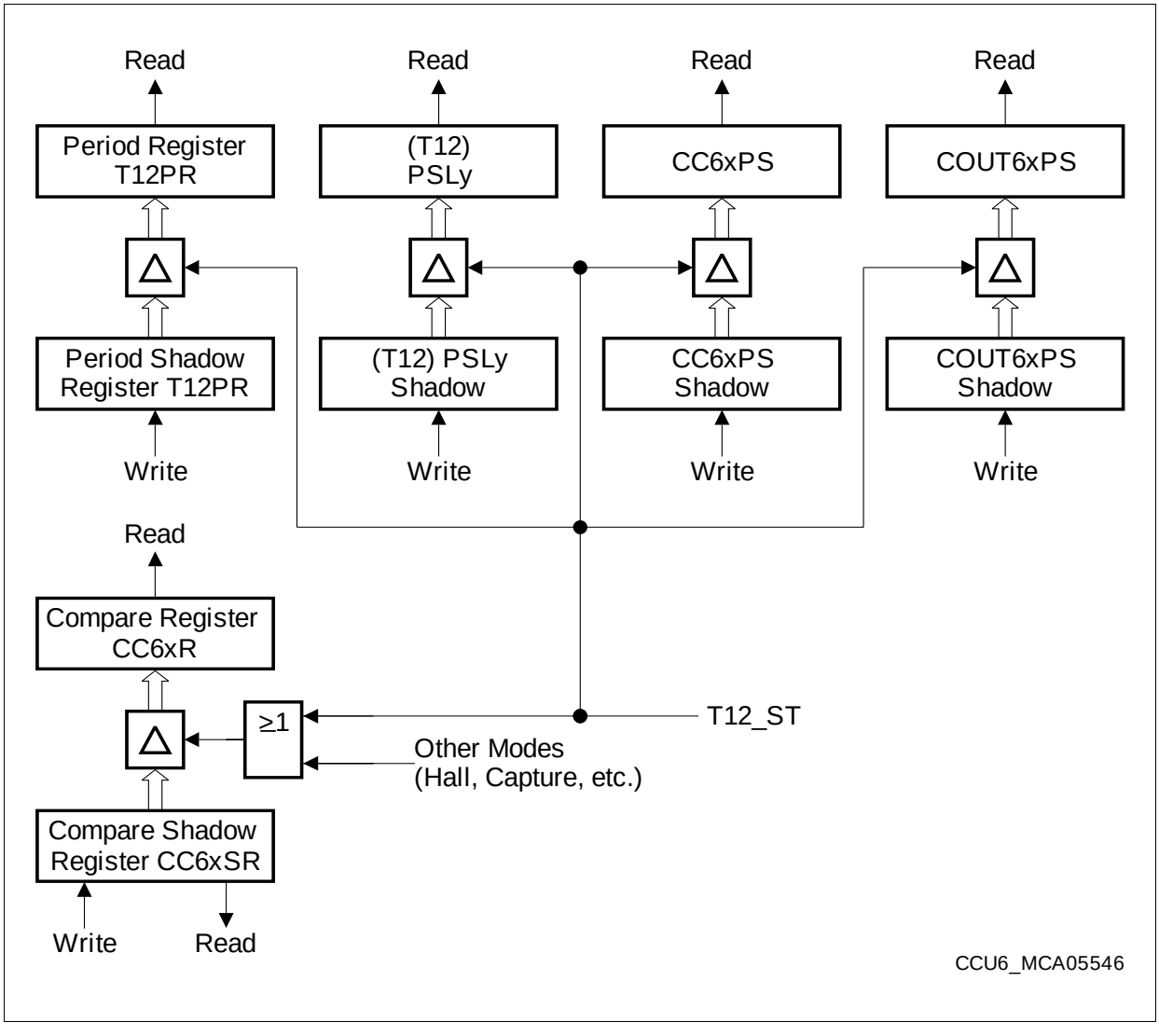


Figure 18-22 T12 Shadow Register Overview

A T12 shadow register transfer takes place (T12_ST active):

- while timer T12 is not running (T12R = 0), or
- STE12 = 1 and a Period-Match is detected while counting up, or
- STE12 = 1 and a One-Match is detected while counting down

When signal T12_ST is active, a shadow register transfer is triggered with the next cycle of the T12 clock. Bit STE12 is automatically cleared with the shadow register transfer.

18.2.7 Timer T12 Operating Mode Selection

The operating mode for the T12 channels are defined by the bit fields **T12MSEL**.MSEL6x.

Table 18-5 T12 Capture/Compare Modes Overview

MSEL6x	Selected Operating Mode
0000_B, 1111_B	Capture/Compare modes switched off
0001_B, 0010_B, 0011_B	Compare mode, see **Section 18.2.3** same behavior for all three codings
$01XX_B$	Double-Register Capture modes, see **Section 18.2.5**
1000_B	Hall Sensor Mode, see **Section 18.6** In order to properly enable this mode, all three MSEL6x fields have to be programmed to Hall Sensor mode.
1001_B	Hysteresis-like compare mode, see **Section 18.2.3.3**
1010_B 1011_B, 1100_B, 1101_B 1110_B	Multi-Input Capture modes, see **Section 18.2.5**

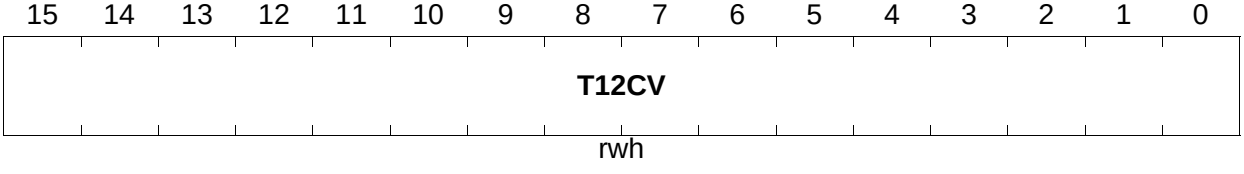
The clocking and counting scheme of the timers are controlled by the timer control registers **TCTR0** and **TCTR2**. Specific actions are triggered by write operations to register **TCTR4**.

18.2.8 T12 related Registers

18.2.8.1 T12 Counter Register

Register T12 represents the counting value of timer T12. It can only be written while the timer T12 is stopped. Write actions while T12 is running are not taken into account. Register T12 can always be read by SW.
In edge-aligned mode, T12 only counts up, whereas in center-aligned mode, T12 can count up and down.

T12
Timer T12 Counter Register **XSFR(10_H)** **Reset Value: 0000_H**



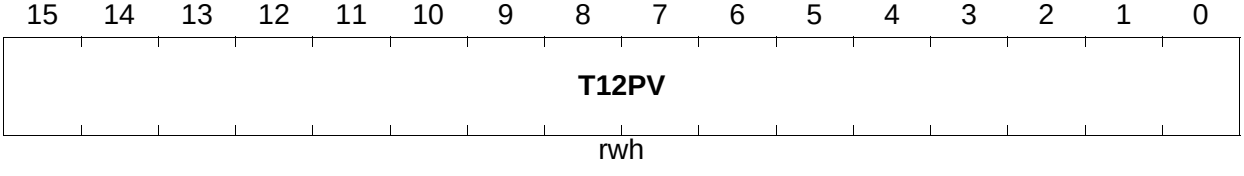
Field	Bits	Type	Description
T12CV	[15:0]	rwh	**Timer 12 Counter Value** This register represents the 16-bit counter value of Timer12.

Note: While timer T12 is stopped, the internal clock divider is reset in order to ensure reproducible timings and delays.

18.2.8.2 Period Register

Register T12PR contains the period value for timer T12. The period value is compared to the actual counter value of T12 and the resulting counter actions depend on the defined counting rules. This register has a shadow register and the shadow transfer is controlled by bit STE12. A read action by SW delivers the value that is currently used for the compare action, whereas the write action targets a shadow register. The shadow register structure allows a concurrent update of all T12-related values.

T12PR
Timer 12 Period Register **XSFR(12_H)** **Reset Value: 0000_H**



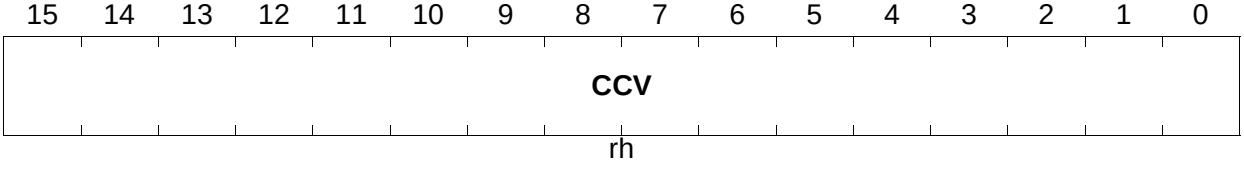
Field	Bits	Type	Description
T12PV	[15:0]	rwh	**T12 Period Value** The value T12PV defines the counter value for T12 leading to a period-match. When reaching this value, the timerT12 is set to zero (edge-aligned mode) or changes its count direction to down counting (center-aligned mode).

18.2.8.3 Capture/Compare Registers

In compare mode, the registers CC6xR (x = 0 - 2) are the actual compare registers for T12. The values stored in CC6xR are compared (all three channels in parallel) to the counter value of T12. In capture mode, the current value of the T12 counter register is captured by registers CC6xR if the corresponding capture event is detected.

CC6xR (x = 0-2)
Capture/Compare Register for Channel CC6x

XSFR(18_H + 2*x) **Reset Value: 0000_H**



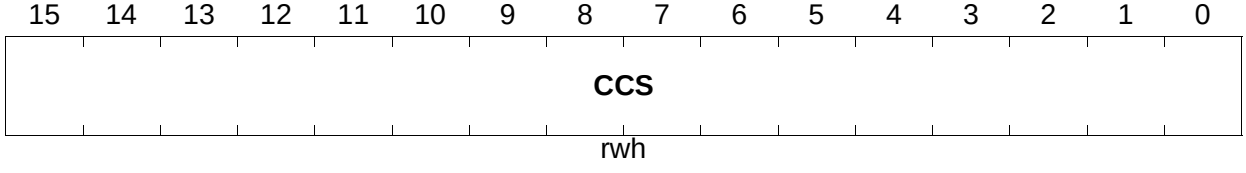
Field	Bits	Type	Description
CCV	[15:0]	rh	**Capture/Compare Value** In compare mode, the bit fields CCV contain the values, that are compared to the T12 counter value. In capture mode, the captured value of T12 can be read from these registers.

18.2.8.4 Capture/Compare Shadow Registers

The registers CC6xR can only be read by SW, the modification of the value is done by a shadow register transfer from register CC6xSR. The corresponding shadow registers CC6xSR can be read and written by SW. In capture mode, the value of the T12 counter register can also be captured by registers CC6xSR if the selected capture event is detected (depending on the selected capture mode).

CC6xSR (x=0-2)
Capture/Compare Shadow Reg. for Channel CC6x

XSFR(20_H+2*x) **Reset Value: 0000_H**



Field	Bits	Type	Description
CCS	[15:0]	rwh	**Shadow Register for Channel x Capture/ Compare Value** In compare mode, the bit fields contents of CCS are transferred to the bit fields CCV for the corresponding channel during a shadow transfer. In capture mode, the captured value of T12 can be read from these registers.

Note: The shadow registers can also be written by SW in capture mode. In this case, the HW capture event wins over the SW write if both happen in the same cycle (the SW write is discarded).

18.2.8.5 Dead-time Control Register

Register T12DTC controls the dead-time generation for the timer T12 compare channels. Each channel can be independently enabled/disabled for dead-time generation. If enabled, the transition from passive state to active state is delayed by the value defined by bit field DTM.
The dead time counters are clocked with the same frequency as T12.
This structure allows symmetrical dead-time generation in center-aligned and in edge-aligned PWM mode. A duty cycle of 50% leads to CC6x, COUT6x switched on for: 0.5 * period - dead time.

Note: The dead-time counters are not reset by bit T12RES, but by bit DTRES.

T12DTC
Dead-Time Control Register for Timer12

XSFR(14_H) **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**DTR2**	**DTR1**	**DTR0**	**0**	**DTE2**	**DTE1**	**DTE0**	**DTM**							
r	rh	rh	rh	r	rw	rw	rw	rw							

Field	Bits	Type	Description
DTM	[7:0]	rw	**Dead-Time** Bit field DTM determines the programmable delay between switching from the passive state to the active state of the selected outputs. The switching from the active state to the passive state is not delayed.
DTE2, **DTE1,** **DTE0**	10, 9, 8	rw	**Dead Time Enable Bits** Bits DTE0..DTE2 enable and disable the dead time generation for each compare channel (0, 1, 2) of timer T12. 0_B Dead-Time Counter x is disabled. The corresponding outputs switch from the passive state to the active state (according to the actual compare status) without any delay. 1_B Dead-Time Counter x is enabled. The corresponding outputs switch from the passive state to the active state (according to the compare status) with the delay programmed in bit field DTM.

Field	Bits	Type	Description
DTR2, DTR1, DTR0	14, 13, 12	rh	**Dead Time Run Indication Bits** Bits DTR0..DTR2 indicate the status of the dead time generation for each compare channel (0, 1, 2) of timer T12. 0_B Dead-Time Counter x is currently in the passive state. 1_B Dead-Time Counter x is currently in the active state.
0	[15:11]	r	**reserved;** returns 0 if read; should be written with 0;

18.2.9 Capture/Compare Control Registers

18.2.9.1 Channel State Bits

The Compare State Register CMPSTAT contains status bits monitoring the current capture and compare state and control bits defining the active/passive state of the compare channels.

CMPSTAT
Compare State Register **XSFR(28_H)** **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T13 IM	**C OUT 63PS**	**C OUT 62PS**	**CC 62PS**	**C OUT 61PS**	**CC 61PS**	**C OUT 60PS**	**CC 60PS**	**0**	**CC 63ST**	**CC POS 62**	**CC POS 61**	**CC POS 60**	**CC 62ST**	**CC 61ST**	**CC 60ST**
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	r	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
CC60ST, CC61ST, CC62ST, CC63ST **1)**	0, 1, 2, 6	rh	**Capture/Compare State Bits** Bits CC6xST monitor the state of the capture/ compare channels. Bits CC6xST (x = 0, 1, 2) are related to T12, bit CC63ST is related to T13. 0_B In compare mode, the timer count is less than the compare value. In capture mode, the selected edge has not yet been detected since the bit has been cleared by SW the last time. 1_B In compare mode, the counter value is greater than or equal to the compare value. In capture mode, the selected edge has been detected.
CCPOS0, CCPOS1, CCPOS2	3, 4, 5	rh	**Sampled Hall Pattern Bits** Bits CCPOSx (x = 0, 1, 2) are indicating the value of the input Hall pattern that has been compared to the current and expected value. The value is sampled when the event HCRDY (Hall Compare Ready) occurs. 0_B The input CCPOSx has been sampled as 0. 1_B The input CCPOSx has been sampled as 1.

Field	Bits	Type	Description
CC60PS, CC61PS, CC62PS, COUT60PS, COUT61PS, COUT62PS, COUT63PS **2)**	8, 10, 12, 9, 11, 13, 14	rwh	**Passive State Select for Compare Outputs** Bits CC6xPS, COUT6xPS select the state of the corresponding compare channel, that is considered to be the passive state. During the passive state, the passive level (defined in register PSLR) is driven by the output pin. Bits CC6xPS, COUT6xPS (x = 0, 1, 2) are related to T12, bit CC63PS is related to T13. 0_B The corresponding compare signal is in passive state while CC6xST is 0. 1_B The corresponding compare signal is in passive state while CC6xST is 1. In capture mode, these bits are not used.
T13IM **3)**	15	rwh	**T13 Inverted Modulation** Bit T13IM inverts the T13 signal for the modulation of the CC6x and COUT6x (x = 0, 1, 2) signals. 0_B T13 output CC63_O is equal to $\overline{\text{CC63ST}}$. 1_B T13 output CC63_O is equal to CC63ST.
0	7	r	**reserved;** returns 0 if read; should be written with 0;

1) These bits are set and cleared according to the T12, T13 switching rules

2) These bits have shadow bits and are updated in parallel to the capture/compare registers of T12, T13 respectively. A read action targets the actually used values, whereas a write action targets the shadow bits.

3) This bit has a shadow bit and is updated in parallel to the compare and period registers of T13. A read action targets the actually used values, whereas a write action targets the shadow bit.

The Compare Status Modification Register CMPMODIF provides software-control (independent set and clear conditions) for the channel state bits CC6xST. This feature enables the user to individually change the status of the output lines by software, for example when the corresponding compare timer is stopped.

CMPMODIF
Compare State Modification Register

XSFR($2A_H$) **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**MCC 63R**		**0**		**MCC 62R**	**MCC 61R**	**MCC 60R**	**0**	**MCC 63S**		**0**		**MCC 62S**	**MCC 61S**	**MCC 60S**
r	w		r		w	w	w	r	w		r		w	w	w

Field	Bits	Type	Description
MCC60S, MCC61S, MCC62S, MCC63S, MCC60R, MCC61R, MCC62R, MCC63R	0, 1, 2, 7, 8, 9, 10, 14	w	**Capture/Compare Status Modification Bits** These bits are used to bits to set (MCC6xS) or to clear (MCC6xR) the corresponding bits CC6xST by SW. This feature allows the user to individually change the status of the output lines by SW, e.g. when the corresponding compare timer is stopped. This allows a bit manipulation of CC6xST-bits by a single data write action. The following functionality of a write access to bits concerning the same capture/compare state bit is provided: [MCC6xR, MCC6xS] = 00_B Bit CC6xST is not changed. 01_B Bit CC6xST is set. 10_B Bit CC6xST is cleared. 11_B reserved
0	[5:3], 7, [13:11], 15	r	**reserved;** returns 0 if read; should be written with 0;

18.2.9.2 T12 Mode Control Register

Register T12MSEL contains control bits to select the capture/compare functionality of the three channels of Timer T12.

T12MSEL
T12 Mode Select Register **XSFR (46_H)** **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DBYP	HSYNC			MSEL62				MSEL61				MSEL60			
rw	rw			rw				rw				rw			

Field	Bits	Type	Description
MSEL60, MSEL61, MSEL62	[3:0], [7:4], [11:8]	rw	**Capture/Compare Mode Selection** These bit fields select the operating mode of the three T12 capture/compare channels. Each channel (x = 0, 1, 2) can be programmed individually for one of these modes (except for Hall Sensor Mode). Coding see **Table 18-5**.
HSYNC	[14:12]	rw	**Hall Synchronization** Bit field HSYNC defines the source for the sampling of the Hall input pattern and the comparison to the current and the expected Hall pattern bit fields. Coding see **Table 18-11**.
DBYP	15	rw	**Delay Bypass** DBYP controls whether the source signal for the sampling of the Hall input pattern (selected by HSYNC) is delayed by the Dead-Time Counter 0. 0 The bypass is not active. Dead-Time Counter 0 is generating a delay after the source signal becomes active. 1 The bypass is active. Dead-Time Counter 0 is not used for a delay.

18.2.9.3 Timer Control Registers

Register TCTR0 controls the basic functionality of both timers, T12 and T13.

Note: A write action to the bit fields T12CLK or T12PRE is only taken into account while the timer T12 is not running (T12R=0). A write action to the bit fields T13CLK or T13PRE is only taken into account while the timer T13 is not running (T13R=0).

TCTR0
Timer Control Register 0 **XSFR($2C_H$)** **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		STE 13	T13R	T13 PRE	T13CLK			CTM	CDIR	STE 12	T12R	T12 PRE	T12CLK		
r		rh	rh	rw	rw			rw	rh	rh	rh	rw	rw		

Field	Bits	Type	Description
T12CLK	[2:0]	rw	**Timer T12 Input Clock Select** Selects the input clock for timer T12 that is derived from the peripheral clock according to the equation $f_{T12} = f_{CC6} / 2^{<T12CLK>}$. 000_B $f_{T12} = f_{CC6}$ 001_B $f_{T12} = f_{CC6} / 2$ 010_B $f_{T12} = f_{CC6} / 4$ 011_B $f_{T12} = f_{CC6} / 8$ 100_B $f_{T12} = f_{CC6} / 16$ 101_B $f_{T12} = f_{CC6} / 32$ 110_B $f_{T12} = f_{CC6} / 64$ 111_B $f_{T12} = f_{CC6} / 128$
T12PRE	3	rw	**Timer T12 Prescaler Bit** In order to support higher clock frequencies, an additional prescaler factor of 1/256 can be enabled for the prescaler for T12. 0_B The additional prescaler for T12 is disabled. 1_B The additional prescaler for T12 is enabled.
T12R	4	rh	**Timer T12 Run Bit[1)]** T12R starts and stops timer T12. It is set/cleared by SW by setting bits T12RR or T12RS or it is cleared by HW according to the function defined by bit field T12SSC. 0_B Timer T12 is stopped. 1_B Timer T12 is running.

Field	Bits	Type	Description
STE12	5	rh	**Timer T12 Shadow Transfer Enable** Bit STE12 enables or disables the shadow transfer of the T12 period value, the compare values and passive state select bits and levels from their shadow registers to the actual registers if a T12 shadow transfer event is detected. Bit STE12 is cleared by hardware after the shadow transfer. A T12 shadow transfer event is a period-match while counting up or a one-match while counting down. 0_B The shadow register transfer is disabled. 1_B The shadow register transfer is enabled.
CDIR	6	rh	**Count Direction of Timer T12** This bit is set/cleared according to the counting rules of T12. 0_B T12 counts up. 1_B T12 counts down.
CTM	7	rw	**T12 Operating Mode** 0_B Edge-aligned Mode: T12 always counts up and continues counting from zero after reaching the period value. 1_B Center-aligned Mode: T12 counts down after detecting a period-match and counts up after detecting a one-match.
T13CLK	[10:8]	rw	**Timer T13 Input Clock Select** Selects the input clock for timer T13 that is derived from the peripheral clock according to the equation $f_{T13} = f_{CC6} / 2^{<T13CLK>}$. 000_B $f_{T13} = f_{CC6}$ 001_B $f_{T13} = f_{CC6} / 2$ 010_B $f_{T13} = f_{CC6} / 4$ 011_B $f_{T13} = f_{CC6} / 8$ 100_B $f_{T13} = f_{CC6} / 16$ 101_B $f_{T13} = f_{CC6} / 32$ 110_B $f_{T13} = f_{CC6} / 64$ 111_B $f_{T13} = f_{CC6} / 128$

Field	Bits	Type	Description
T13PRE	11	rw	**Timer T13 Prescaler Bit** In order to support higher clock frequencies, an additional prescaler factor of 1/256 can be enabled for the prescaler for T13. 0_B The additional prescaler for T13 is disabled. 1_B The additional prescaler for T13 is enabled.
T13R	12	rh	**Timer T13 Run Bit**[2)] T13R starts and stops timer T13. It is set/cleared by SW by setting bits T13RR orT13RS or it is set/cleared by HW according to the function defined by bit fields T13SSC, T13TEC and T13TED. 0_B Timer T13 is stopped. 1_B Timer T13 is running.
STE13	13	rh	**Timer T13 Shadow Transfer Enable** Bit STE13 enables or disables the shadow transfer of the T13 period value, the compare value and passive state select bit and level from their shadow registers to the actual registers if a T13 shadow transfer event is detected. Bit STE13 is cleared by hardware after the shadow transfer. A T13 shadow transfer event is a period-match. 0_B The shadow register transfer is disabled. 1_B The shadow register transfer is enabled.
0	[15: 14]	r	**reserved;** returns 0 if read; should be written with 0;

1) A concurrent set/clear action on T12R (from T12SSC, T12RR or T12RS) will have no effect. The bit T12R will remain unchanged.

2) A concurrent set/cleared action on T13R (from T13SSC, T13TEC, T13RR or T13RS) will have no effect. The bit T12R will remain unchanged.

Register TCTR2 controls the single-shot and the synchronization functionality of both timers T12 and T13. Both timers can run in single-shot mode. In this mode they stop their counting sequence automatically after one counting period with a count value of zero. The single-shot mode and the synchronization feature of T13 to T12 allow the generation of events with a programmable delay after well-defined PWM actions of T12.

TCTR2
Timer Control Register 2 **XSFR($2E_H$)** **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				T13 RSEL		T12 RSEL		0	T13 TED		T13 TEC			T13 SSC	T12 SSC
r				rw		rw		r	rw		rw			rw	rw

Field	Bits	Type	Description
T12SSC	0	rw	**Timer T12 Single Shot Control** This bit controls the single shot-mode of T12. 0_B The single-shot mode is disabled, no HW action on T12R. 1_B The single shot mode is enabled, the bit T12R is cleared by HW if - T12 reaches its period value in edge-aligned mode - T12 reaches the value 1 while down counting in center-aligned mode. In parallel to the clear action of bit T12R, the bits CC6xST (x=0, 1, 2) are cleared.
T13SSC	1	rw	**Timer T13 Single Shot Control** This bit controls the single shot-mode of T13. 0_B No HW action on T13R 1_B The single-shot mode is enabled, the bit T13R is cleared by HW if T13 reaches its period value. In parallel to the clear action of bit T13R, the bit CC63ST is cleared.

Field	Bits	Type	Description
T13TEC	[4:2]	rw	**T13 Trigger Event Control** bit field T13TEC selects the trigger event to start T13 (automatic set of T13R for synchronization to T12 compare signals) according to following combinations: 000_B no action 001_B set T13R on a T12 compare event on channel 0 010_B set T13R on a T12 compare event on channel 1 011_B set T13R on a T12 compare event on channel 2 100_B set T13R on any T12 compare event (ch. 0, 1, 2) 101_B set T13R upon a period-match of T12 110_B set T13R upon a zero-match of T12 (while counting up) 111_B set T13R on any edge of inputs CCPOSx
T13TED	[6:5]	rw	**Timer T13 Trigger Event Direction**[1)] Bit field T13TED delivers additional information to control the automatic set of bit T13R in the case that the trigger action defined by T13TEC is detected. 00_B reserved, no action 01_B while T12 is counting up 10_B while T12 is counting down 11_B independent on the count direction of T12
T12RSEL	[9:8]	rw	**Timer T12 External Run Selection** Bit field T12RSEL defines the event of signal T12HR that can set the run bit T12R by HW. 00_B The external setting of T12R is disabled. 01_B Bit T12R is set if a rising edge of signal T12HR is detected. 10_B Bit T12R is set if a falling edge of signal T12HR is detected. 11_B Bit T12R is set if an edge of signal T12HR is detected.

Field	Bits	Type	Description
T13RSEL	[11:10]	rw	**Timer T13 External Run Selection** Bit field T13RSEL defines the event of signal T13HR that can set the run bit T13R by HW. 00_B The external setting of T13R is disabled. 01_B Bit T13R is set if a rising edge of signal T13HR is detected. 10_B Bit T13R is set if a falling edge of signal T13HR is detected. 11_B Bit T13R is set if an edge of signal T13HR is detected.
0	7, [15: 12]	r	**reserved;** returns 0 if read; should be written with 0;

1) Example:
If the timer T13 is intended to start at any compare event on T12 (T13TEC=100) the trigger event direction can be programmed to
- counting up >> a T12 channel 0, 1, 2 compare match triggers T13R only while T12 is counting up
- counting down >> a T12 channel 0, 1, 2 compare match triggers T13R only while T12 is counting down
- independent from bit CDIR >> each T12 channel 0, 1, 2 compare match triggers T13R

The timer count direction is taken from the value of bit CDIR. As a result, if T12 is running in edge-aligned mode (counting up only), T13 can only be started automatically if bit field T13TED=01 or 11.

Register TCTR4 provides software-control (independent set and clear conditions) for the run bits T12R and T13R. Furthermore, the timers can be reset (while running) and bits STE12 and STE13 can be controlled by software. Reading these bits always returns 0.

TCTR4
Timer Control Register 4 **XSFR(26_H)** **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T13 STD	**T13 STR**	**T13 CNT**	**0**		**T13 RES**	**T13 RS**	**T13 RR**	**T12 STD**	**T12 STR**	**T12 CNT**	**0**	**DT RES**	**T12 RES**	**T12 RS**	**T12 RR**
w	w	w	r		w	w	w	w	w	w	r	w	w	w	w

Field	Bits	Type	Description
T12RR	0	w	**Timer T12 Run Reset** Setting this bit clears the T12R bit. 0_B T12R is not influenced. 1_B T12R is cleared, T12 stops counting.
T12RS	1	w	**Timer T12 Run Set** Setting this bit sets the T12R bit. 0_B T12R is not influenced. 1_B T12R is set, T12 starts counting.
T12RES	2	w	**Timer T12 Reset** 0_B No effect on T12. 1_B The T12 counter register is cleared to zero. The switching of the output signals is according to the switching rules. Setting of T12RES has no impact on bit T12R.
DTRES	3	w	**Dead-Time Counter Reset** 0_B No effect on the dead-time counters. 1_B The three dead-time counter channels are cleared to zero.
T12CNT	5	w	**Timer T12 Count Event** 0_B No action 1_B If enabled (PISELH), timer T12 counts one step.
T12STR	6	w	**Timer T12 Shadow Transfer Request** 0_B No action 1_B STE12 is set, enabling the shadow transfer.

Field	Bits	Type	Description
T12STD	7	w	**Timer T12 Shadow Transfer Disable** 0_B No action 1_B STE12 is cleared without triggering the shadow transfer.
T13RR	8	w	**Timer T13 Run Reset** Setting this bit clears the T13R bit. 0_B T13R is not influenced. 1_B T13R is cleared, T13 stops counting.
T13RS	9	w	**Timer T13 Run Set** Setting this bit sets the T13R bit. 0_B T13R is not influenced. 1_B T13R is set, T13 starts counting.
T13RES	10	w	**Timer T13 Reset** 0_B No effect on T13. 1_B The T13 counter register is cleared to zero. The switching of the output signals is according to the switching rules. Setting of T13RES has no impact on bit T13R.
T13CNT	13	w	**Timer T13 Count Event** 0_B No action 1_B If enabled (PISELH), timer T13 counts one step.
T13STR	14	w	**Timer T13 Shadow Transfer Request** 0_B No action 1_B STE13 is set, enabling the shadow transfer.
T13STD	15	w	**Timer T13 Shadow Transfer Disable** 0_B No action 1_B STE13 is cleared without triggering the shadow transfer.
0	4, [12:11]	r	**reserved;** returns 0 if read; should be written with 0;

Note: A simultaneous write of a 1 to bits that set and clear the same bit will trigger no action. The corresponding bit will remain unchanged.

18.3 Operating Timer T13

Timer T13 is implemented similarly to Timer T12, but only with one channel in compare mode. A 16-bit up-counter is connected to a channel register via a comparator, that generates a signal when the counter contents match the contents of the channel register. A variety of control functions facilitate the adaptation of the T13 structure to different application needs. In addition, T13 can be started synchronously to timer T12 events.

This section provides information about:

- T13 overview (see **Section 18.3.1**)
- Counting scheme (see **Section 18.3.2**)
- Compare mode (see **Section 18.3.3**)
- Compare output path (see **Section 18.3.4**)
- Shadow register transfer (see **Section 18.3.5**)
- T13 counter register description (see **Section 18.3.6**)

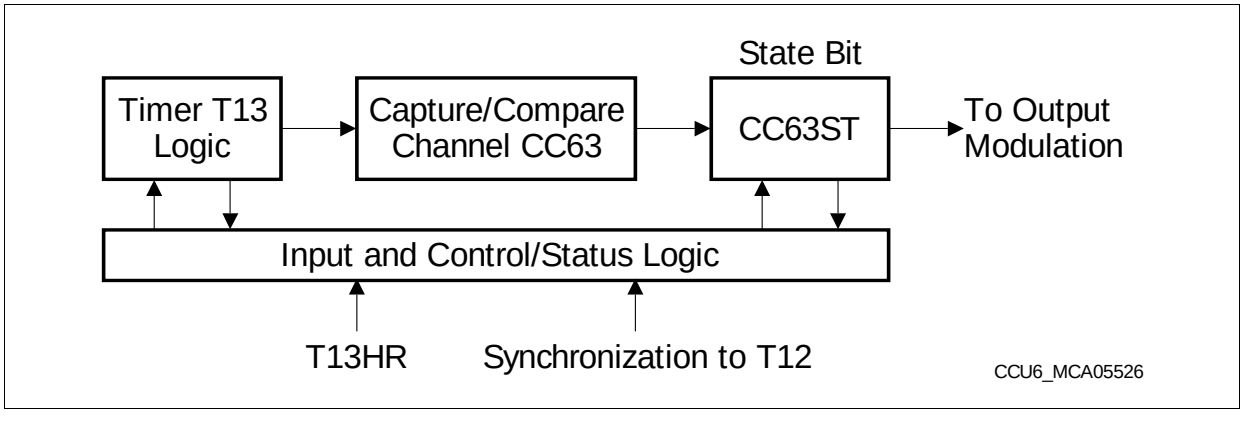


Figure 18-23 Overview Diagram of the Timer T13 Block

18.3.1 T13 Overview

Figure 18-24 shows a detailed block diagram of Timer T13. The functions of the timer T12 block are controlled by bits in registers **TCTR0**, **TCTR2**, and **PISELH**.
Timer T13 receives its input clock, f_{T13}, from the module clock f_{CC6} via a programmable prescaler and an optional 1/256 divider or from an input signal T13HR. T13 can only count up (similar to the Edge-Aligned mode of T12).

Via a comparator, the timer T13 Counter Register **T13** is connected to the Period Register **T13PR**. This register determines the maximum count value for T13. When T13 reaches the period value, signal T13_PM (T13 Period Match) is generated and T13 is cleared to 0000_H with the next T13 clock edge. The Period Register receives a new period value from its Shadow Period Register, T13PS, that is loaded via software. The transfer of a new period value from the shadow register into T13PR is controlled via the ‘T13 Shadow Transfer’ control signal, T13_ST. The generation of this signal depends on the associated control bit STE13. Providing a shadow register for the period value as

well as for other values related to the generation of the PWM signal facilitates a concurrent update by software for all relevant parameters (refer to **Table 18.3.5**).
Another signal indicates whether the counter contents are equal to 0000_H (T13_ZM).

A Single-Shot control bit, T13SSC, enables an automatic stop of the timer when the current counting period is finished (see **Figure 18-26**).

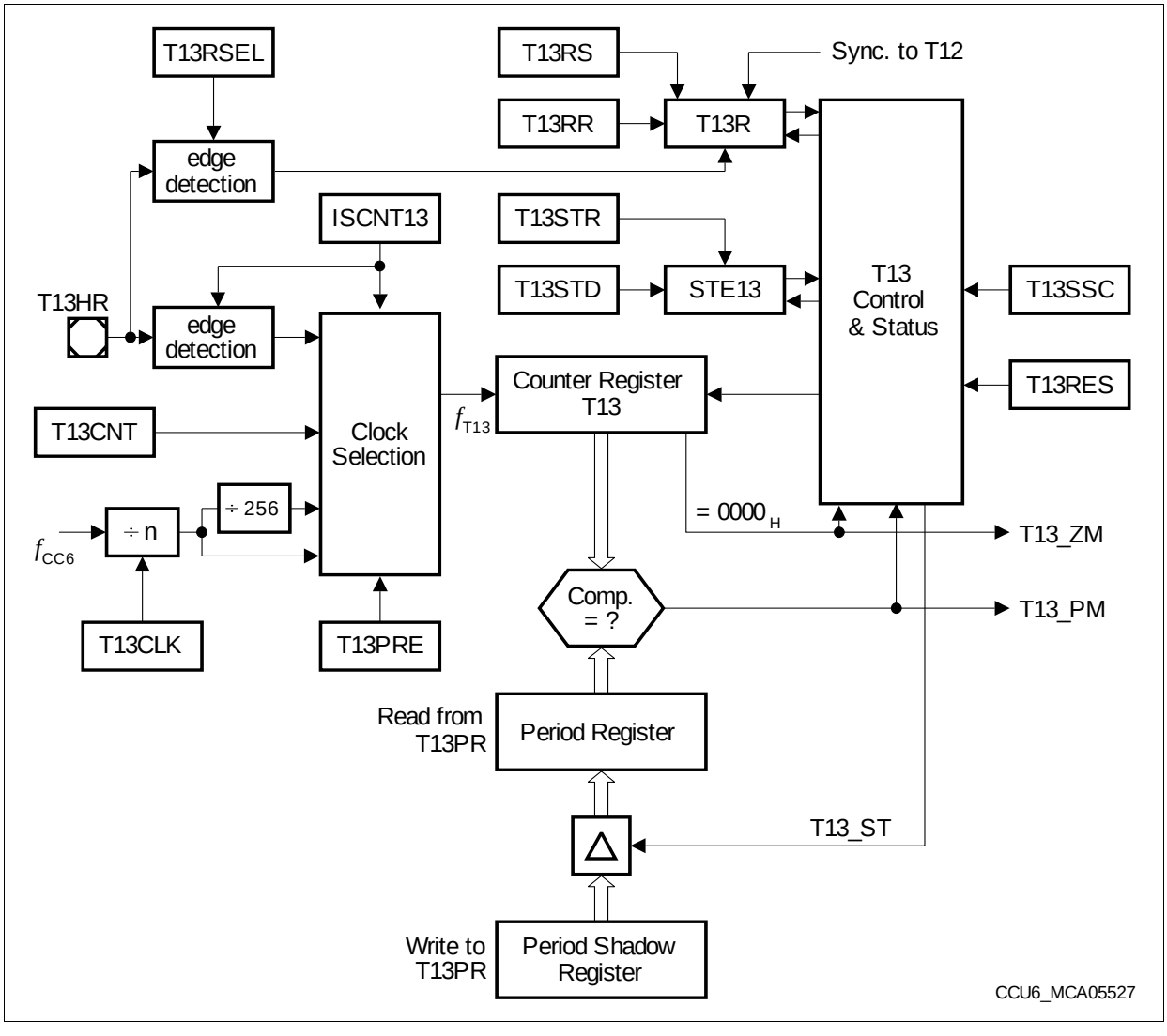


Figure 18-24 T13 Counter Logic and Period Comparators

The start or stop of T13 is controlled by the Run bit, T13R. This control bit can be set by software via the associated set/clear bits T13RS or T13RR in register **TCTR4**, or it is cleared by hardware according to preselected conditions (single-shot mode).
The timer T13 run bit T13R must not be set while the applied T13 period value is zero.
Bit T13R can be set automatically if an event of T12 is detected to synchronize T13 timings to T12 events, e.g. to generate a programmable delay via T13 after an edge of a T12 compare channel before triggering an AD conversion (T13 can trigger ADC

conversions).
Timer T13 can be cleared to 0000_H via control bit T13RES. Setting this write-only bit only clears the timer contents, but has no further effects, e.g., it does not stop the timer.

The generation of the T13 shadow transfer control signal, T13_ST, is enabled via bit STE13. This bit can be set or cleared by software indirectly through its associated set/ reset control bits T13STR and T13STD.

Two bit fields, T13TEC and T13TED, control the synchronization of T13 to Timer T12 events. T13TEC selects the trigger event, while T13TED determines for which T12 count direction the trigger should be active.
While Timer T13 is running, write accesses to the count register T13 are not taken into account. If T13 is stopped, write actions to register T13 are immediately taken into account.

Note: The T13 Period Register and its associated shadow register are located at the same physical address. A write access to this address targets the Shadow Register, while a read access reads from the actual period register.

18.3.2 T13 Counting Scheme

This section describes the clocking and the counting capabilities of T13.

18.3.2.1 Clock Selection

In **Timer Mode** (**PISELH**. ISCNT13 = 00_B), the input clock f_{T13} of Timer T13 is derived from the internal module clock f_{CC6} through a programmable prescaler and an optional 1/256 divider. The resulting prescaler factors are listed in **Table 18-6**. The prescaler of T13 is cleared while T13 is not running (**TCTR0**.T13R = 0) to ensure reproducible timings and delays.

Table 18-6 Timer T13 Input Clock Options

T13CLK	Resulting Input Clock f_{T13} Prescaler Off (T13PRE = 0)	Resulting Input Clock f_{T13} Prescaler On (T13PRE = 1)
000_B	f_{CC6}	f_{CC6} / 256
001_B	f_{CC6} / 2	f_{CC6} / 512
010_B	f_{CC6} / 4	f_{CC6} / 1024
011_B	f_{CC6} / 8	f_{CC6} / 2048
100_B	f_{CC6} / 16	f_{CC6} / 4096
101_B	f_{CC6} / 32	f_{CC6} / 8192
110_B	f_{CC6} / 64	f_{CC6} / 16384
111_B	f_{CC6} / 128	f_{CC6} / 32768

In **Counter Mode**, timer T13 counts one step:

- If a 1 is written to **TCTR4**.T13CNT and **PISELH**.ISCNT13 = 01_B
- If a rising edge of input signal T13HR is detected and **PISELH**.ISCNT13 = 10_B
- If a falling edge of input signal T13HR is detected and **PISELH**.ISCNT13 = 11_B

18.3.2.2 T13 Counting

The period of the timer is determined by the value in the period Register T13PR according to the following formula:

$T13_{PER}$ = <Period-Value> + 1; in T13 clocks (f_{T13}) (18.3)

Timer T13 can only count up, comparable to the Edge-Aligned mode of T12. This leads to very simple ‘counting rule’ for the T13 counter:

- The counter is cleared with the next T13 clock edge if a Period-Match is detected. The counting direction is always upwards.

The behavior of T13 is illustrated in **Figure 18-25**.

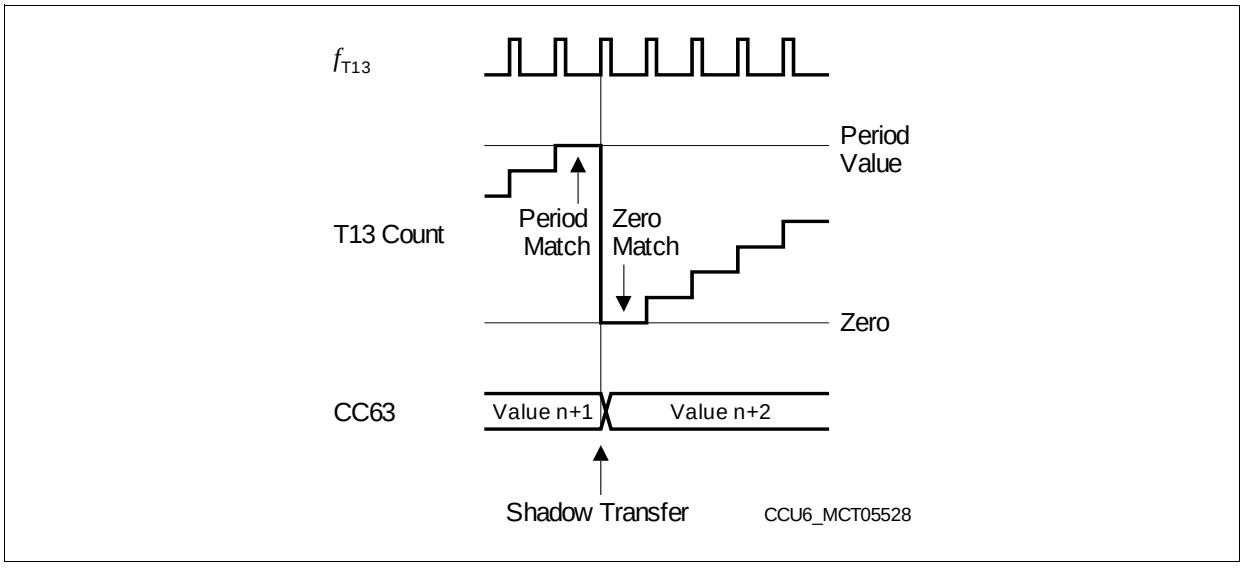


Figure 18-25 T13 Counting Sequence

18.3.2.3 Single-Shot Mode

In Single-Shot Mode, the timer run bit T13R is cleared by hardware. If bit T13SSC = 1, the timer T13 will stop when the current timer period is finished.

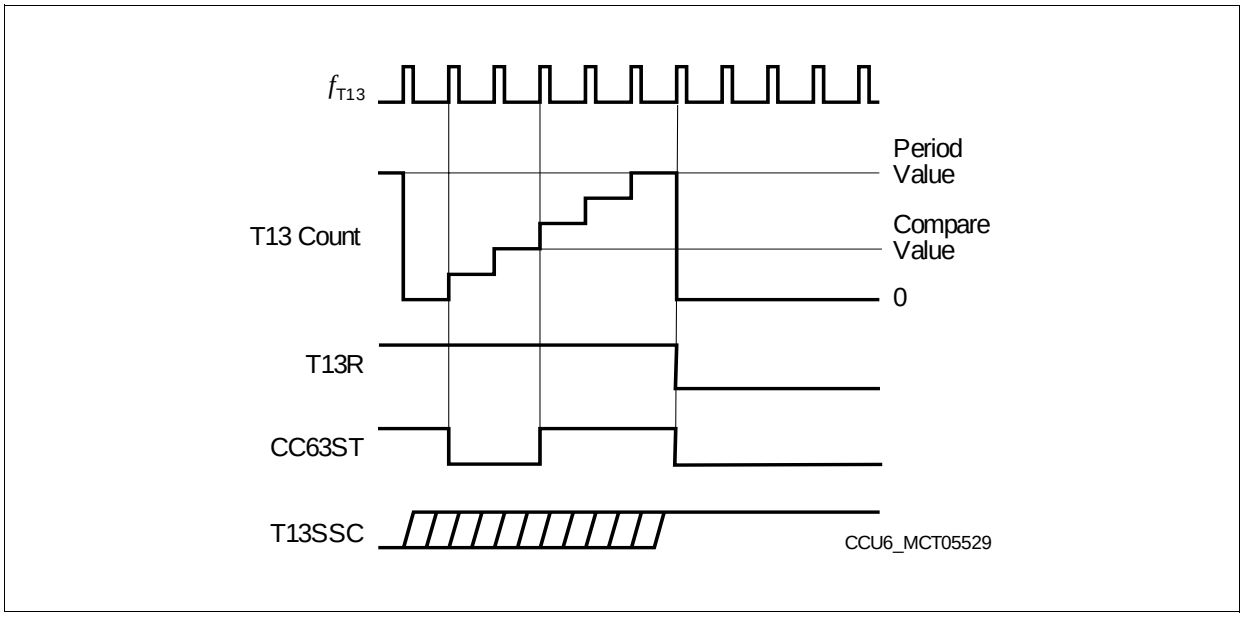


Figure 18-26 Single-Shot Operation of Timer T13

18.3.2.4 Synchronization to T12

Timer T13 can be synchronized to a T12 event. Bit fields T13TEC and T13TED select the event that is used to start Timer T13. The selected event sets bit T13R via HW, and T13 starts counting. Combined with the Single-Shot mode, this feature can be used to generate a programmable delay after a T12 event.

Figure 18-27 shows an example for the synchronization of T13 to a T12 event. Here, the selected event is a compare-match (compare value = 2) while counting up. The clocks of T12 and T13 can be different (other prescaler factor); the figure shows an example in which T13 is clocked with half the frequency of T12.

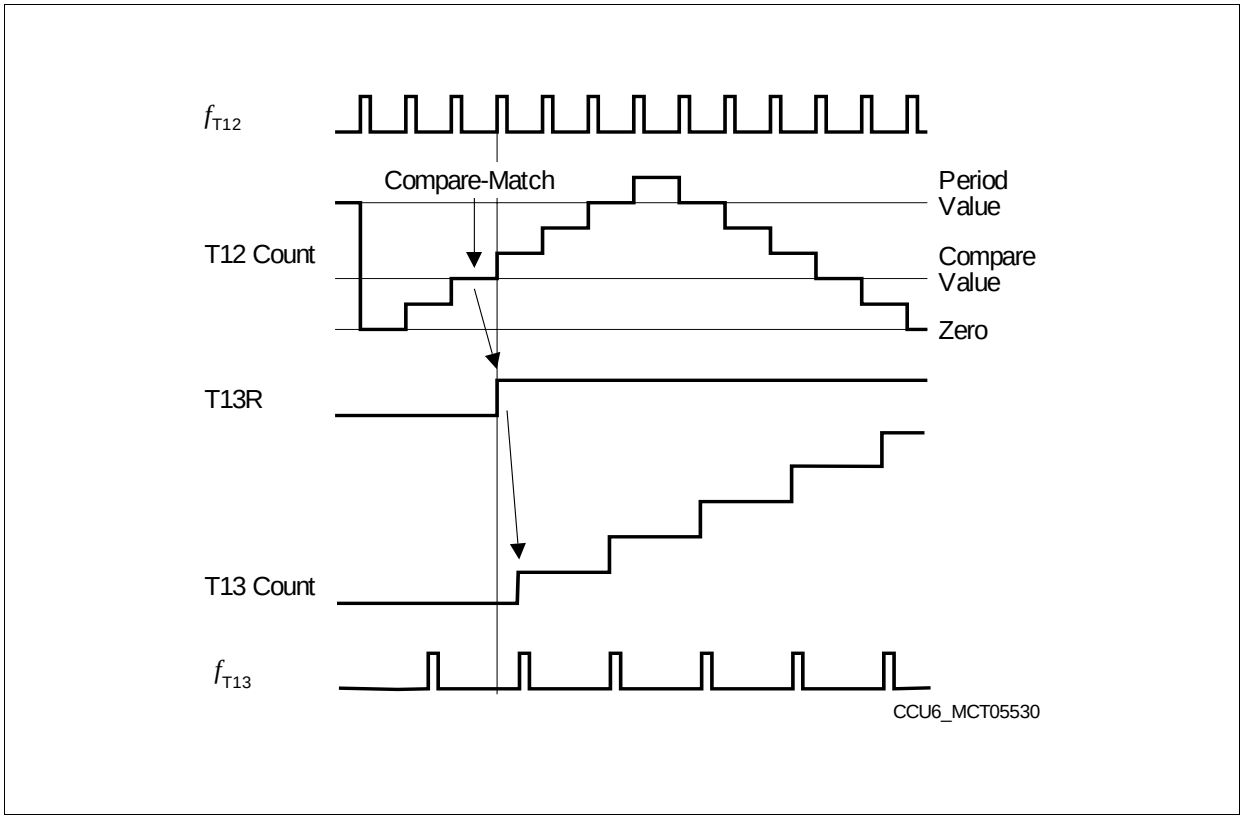


Figure 18-27 Synchronization of T13 to T12 Compare Match

Bit field T13TEC selects the trigger event to start T13 (automatic set of T13R for synchronization to T12 compare signals) according to the combinations shown in **Table 18-7**. Bit field T13TED additionally specifies for which count direction of T12 the selected trigger event should be regarded (see **Table 18-8**).

Table 18-7 T12 Trigger Event Selection

T13TEC	Selected Event
000_B	None
001_B	T12 Compare Event on Channel 0 (CM_CC60)
010_B	T12 Compare Event on Channel 1 (CM_CC61)
011_B	T12 Compare Event on Channel 2 (CM_CC62)
100_B	T12 Compare Event on any Channel (0, 1, 2)
101_B	T12 Period-Match (T12_PM)
110_B	T12 Zero-Match while counting up (T12_ZM and CDIR = 0)
111_B	Any Hall State Change

Table 18-8 T12 Trigger Event Additional Specifier

T13TED	Selected Event Specifier
00_B	Reserved, no action
01_B	Selected event is active while T12 is counting up (CDIR = 0)
10_B	Selected event is active while T12 is counting down (CDIR = 1)
11_B	Selected event is active independently of the count direction of T12

18.3.3 T13 Compare Mode

Associated with Timer T13 is one compare channel, that can perform compare operations with regard to the contents of the T13 counter.

Figure 18-23 gives an overview on the T13 channel in Compare Mode. The channel is connected to the T13 counter register via an equal-to comparator, generating a compare match signal when the contents of the counter matches the contents of the compare register.
The channel consists of the comparator and a double register structure - the actual compare register, **CC63R**, feeding the comparator, and an associated shadow register, **CC63SR**, that is preloaded by software and transferred into the compare register when signal T13 shadow transfer, T13_ST, gets active. Providing a shadow register for the compare value as well as for other values related to the generation of the PWM signal facilitates a concurrent update by software for all relevant parameters.

Associated with the channel is a State Bit, **CMPSTAT**.CC63ST, holding the status of the compare operation. **Figure 18-28** gives an overview on the logic for the State Bit.

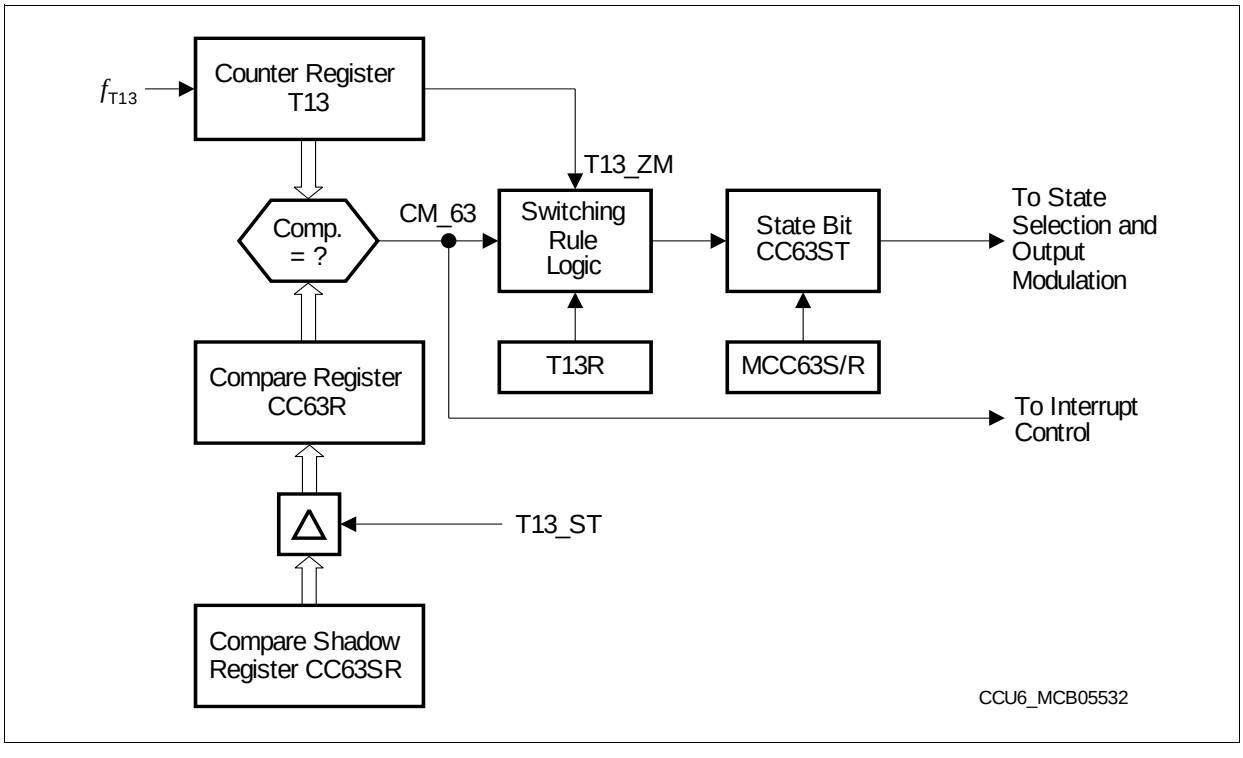


Figure 18-28 T13 State Bit Block Diagram

A compare interrupt event CM_63 is signaled when a compare match is detected. The actual setting of a State Bit has no influence on the interrupt generation.

The inputs to the switching rule logic for the CC63ST bit are the timer run bit (T13R), the timer zero-match signal (T13_ZM), and the actual individual compare-match signal CM_63. In addition, the state bit can be set or cleared by software via bits MCC63S and

MCC63R in register **CMPMODIF**.
A modification of the State Bit CC63ST by hardware is only possible while Timer T13 is running (T13R = 1). If this is the case, the following switching rules apply for setting and resetting the State Bit in Compare Mode:

State Bit **CC63ST is set** to 1

- with the next T13 clock (f_{T13}) after a compare-match (T13 is always counting up) (i.e., when the counter is incremented above the compare value);
- with the next T13 clock (f_{T13}) after a zero-match AND a parallel compare-match.

State Bit **CC63ST is cleared** to 0

- with the next T13 clock (f_{T13}) after a zero-match AND NO parallel compare-match.

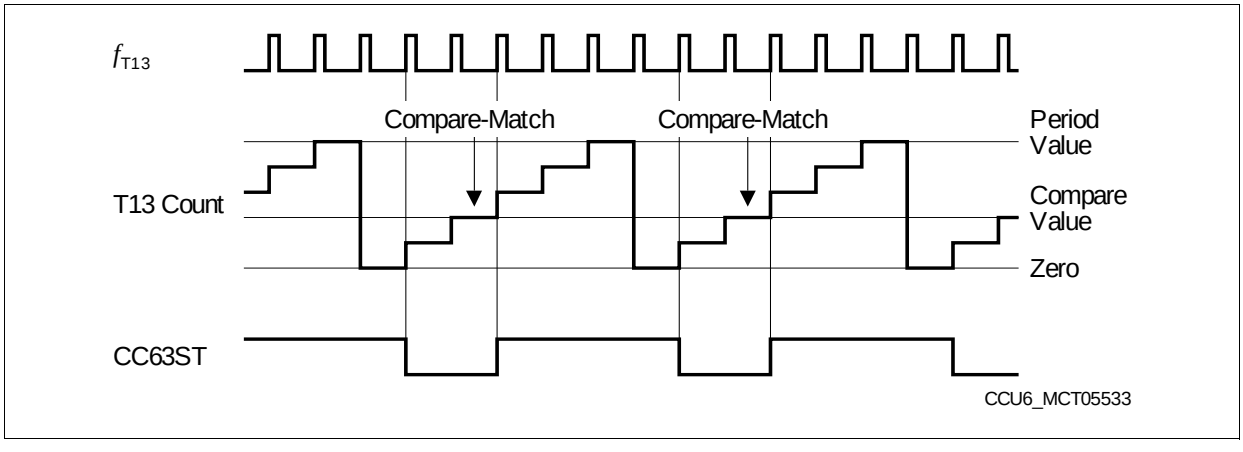


Figure 18-29 T13 Compare Operation

18.3.4 Compare Mode Output Path

Figure 18-30 gives an overview on the signal path from the channel State Bit CC63ST to its output pin COUT63. As illustrated, a user can determine the desired output behavior in relation to the current state of CC63ST. Please refer to **Section 18.2.4.3** for detailed information on the output modulation for T12 signals.

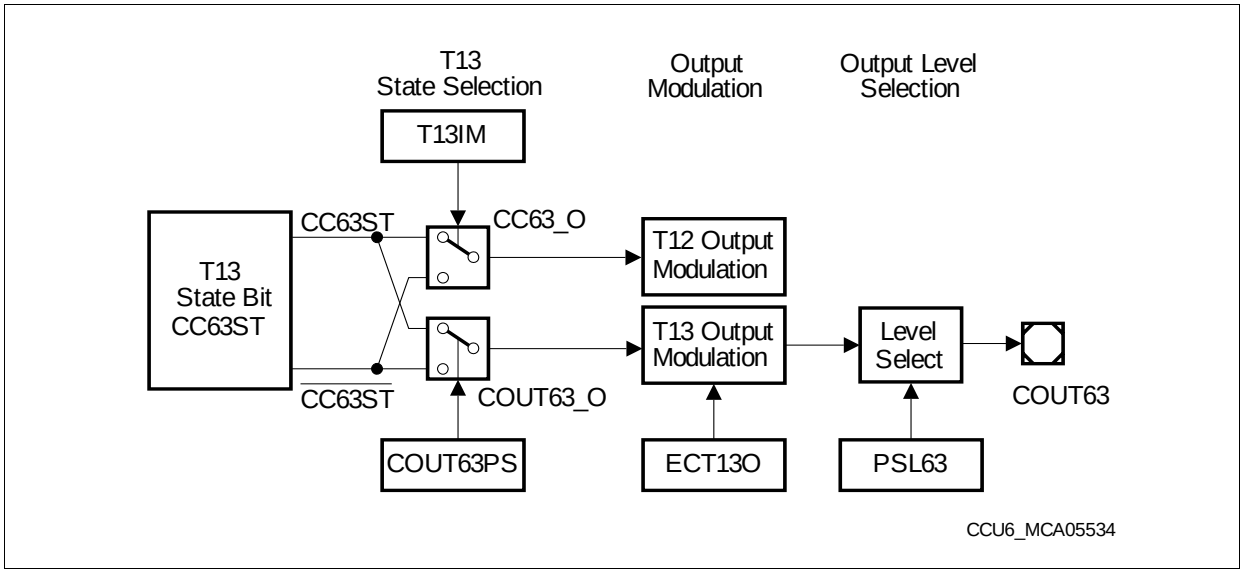


Figure 18-30 Channel 63 Output Path

The output line COUT63_O can generate a T13 PWM at the output pin COUT63. The signal CC63_O can be used to modulate the T12-related output signals with a T13 PWM. In order to decouple COUT63 from the internal modulation, the compare state leading to an active signal can be selected independently by bits T13IM and COUT63PS.

The last block of the data path is the Output Modulation block. Here, the modulation source T13 and the trap functionality are combined and control the actual level of the output pin COUT63 (see **Figure 18-31**):

- The **T13 related compare signal** COUT63_O delivered by the T13 state selection with the enable bit **MODCTR**.ECT13O
- The **trap state** TRPS with an individual enable bit **TRPCTR**.TRPEN13

If the modulation input signal COUT63_O is enabled (ECT13O = 1) and is at passive state, the modulated is also in passive state. If the modulation input is not enabled, the output is in passive state.
If the Trap State is active (TRPS = 1), then the output enabled for the trap signal (by TRPEN13 = 1) is set to the passive state.

The output of the modulation control block is connected to a level select block. It offers the option to determine the actual output level of a pin, depending on the state of the output line (decoupling of active/passive state and output polarity) as specified by the Passive State Select bit **PSLR**.PSL63. If the modulated output signal is in the passive

state, the level specified directly by PSL63 is output. If it is in the active state, the inverted level of PSL63 is output. This allows the user to adapt the polarity of an active output signal to the connected circuitry.

The PSL63 bit has a shadow register to allow for updates with the T13 shadow transfer signal (T13_ST) without undesired pulses on the output lines. A read action returns the actually used value, whereas a write action targets the shadow bit. Providing a shadow register for the PSL value as well as for other values related to the generation of the PWM signal facilitates a concurrent update by software for all relevant parameters.

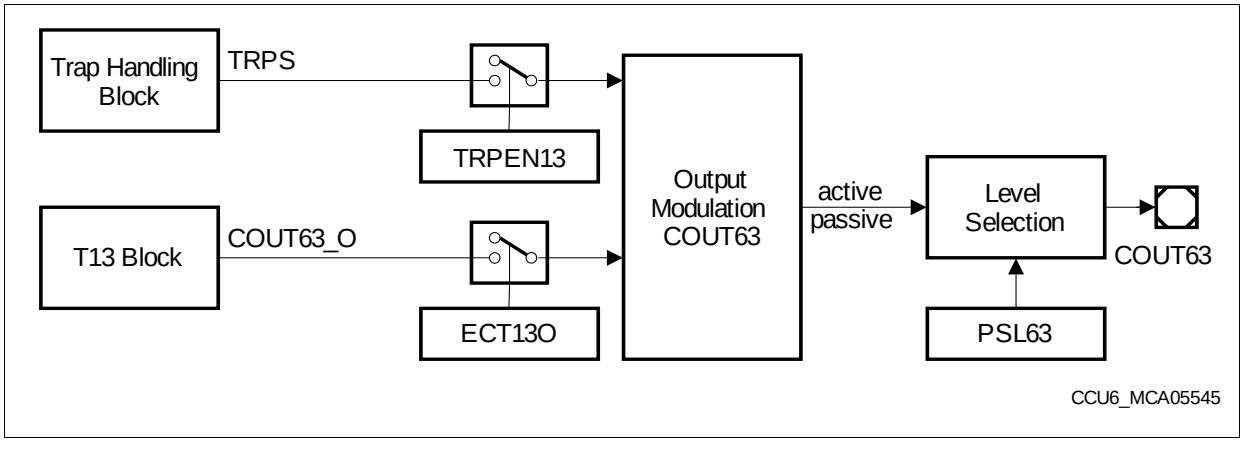


Figure 18-31 T13 Output Modulation

18.3.5 T13 Shadow Register Transfer

A special shadow transfer signal (T13_ST) can be generated to facilitate updating the period and compare values of the compare channel CC63 synchronously to the operation of T13. Providing a shadow register for values defining one PWM period facilitates a concurrent update by software for all relevant parameters. The next PWM period can run with a new set of parameters. The generation of this signal is requested by software via bit **TCTR0**.STE13 (set by writing 1 to the write-only bit **TCTR4**.T13STR, cleared by writing 1 to the write-only bit **TCTR4**.T13STD).
When signal T13_ST is active, a shadow register transfer is triggered with the next cycle of the T13 clock. Bit STE13 is automatically cleared with the shadow register transfer.
A T13 shadow register transfer takes place (T13_ST active):

- while timer T13 is not running (T13R = 0), or
- STE13 = 1 and a Period-Match is detected while T13R = 1

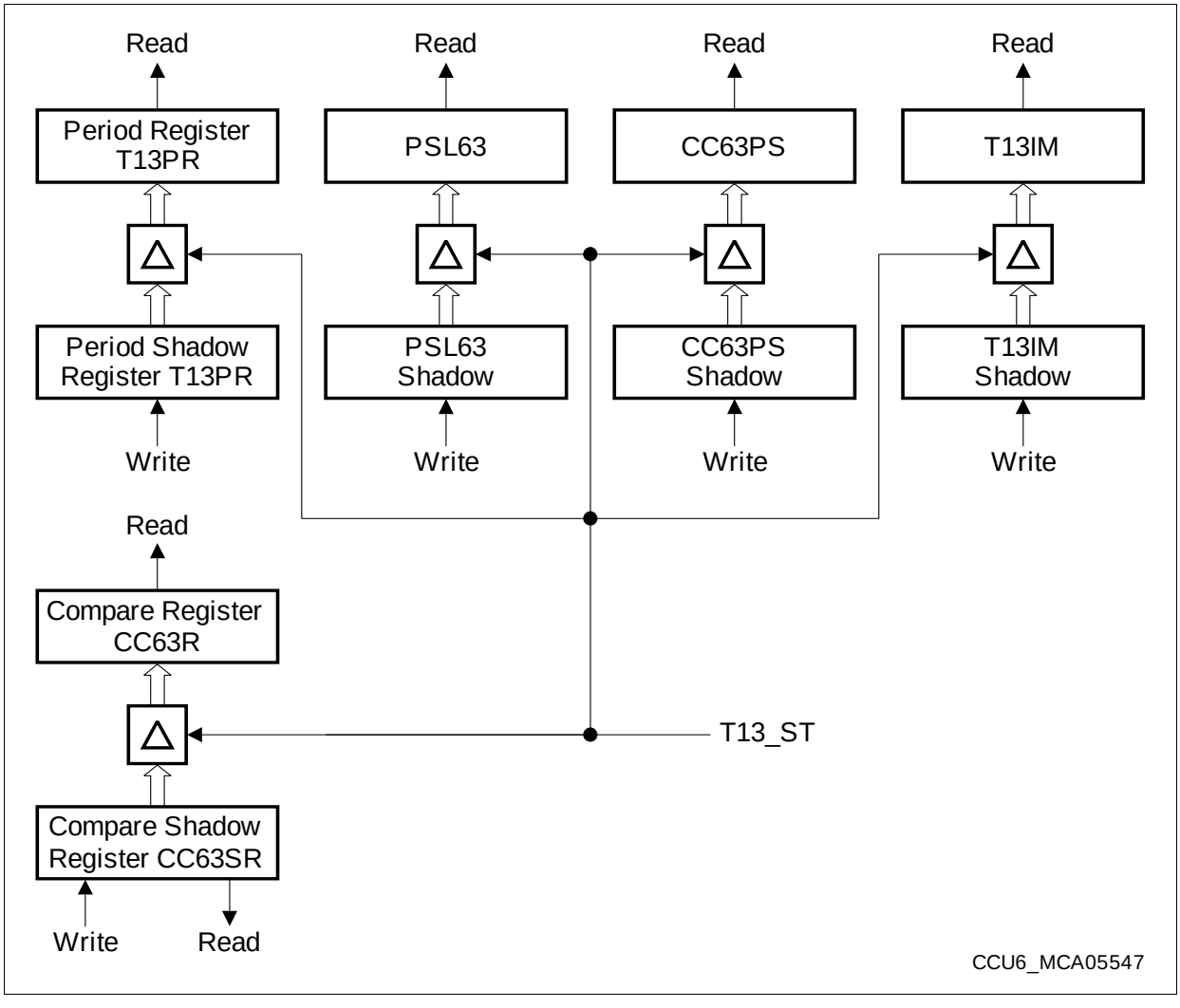


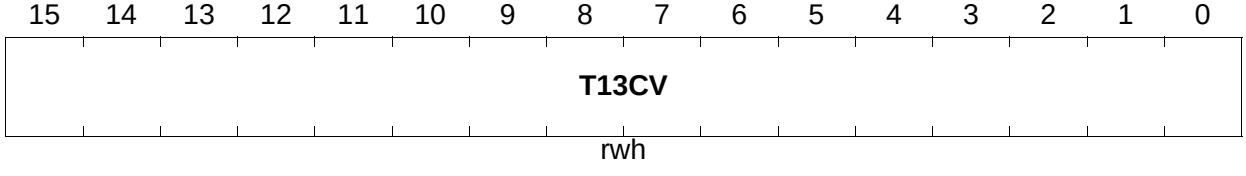
Figure 18-32 T13 Shadow Register Overview

18.3.6 T13 related Registers

18.3.6.1 T13 Counter Register

The generation of the patterns for a single channel pulse width modulation (PWM) is based on timer T13. The registers related to timer T13 can be concurrently updated (with well-defined conditions) in order to ensure consistency of the PWM signal. T13 can be synchronized to several timer T12 events.
Timer T13 only supports compare mode on its compare channel CC63.
Register T13 represents the counting value of timer T13. It can only be written while the timer T13 is stopped. Write actions while T13 is running are not taken into account. Register T13 can always be read by SW.
Timer T13 only supports edge-aligned mode (counting up).

T13
Timer T13 Counter Register **XSFR(30_H)** **Reset Value: 0000_H**



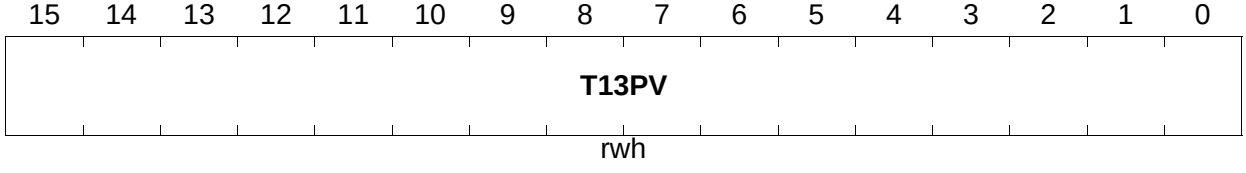
Field	Bits	Type	Description
T13CV	[15:0]	rwh	**Timer 13 Counter Value** This register represents the 16-bit counter value of Timer13.

Note: While timer T13 is stopped, the internal clock divider is reset in order to ensure reproducible timings and delays.

18.3.6.2 Period Register

Register T13PR contains the period value for timer T13. The period value is compared to the actual counter value of T13 and the resulting counter actions depend on the defined counting rules. This register has a shadow register and the shadow transfer is controlled by bit STE13. A read action by SW delivers the value currently used for the compare action, whereas the write action targets a shadow register. The shadow register structure allows a concurrent update of all T13-related values.

T13PR
Timer 13 Period Register **XSFR(32_H)** **Reset Value: 0000_H**

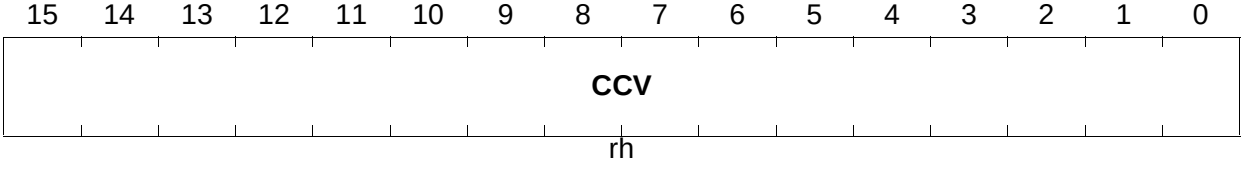


Field	Bits	Type	Description
T13PV	[15:0]	rwh	**T13 Period Value** The value T13PV defines the counter value for T13 leading to a period-match. When reaching this value, the timer T13 is set to zero.

18.3.6.3 Compare Register

Registers CC63R is the actual compare register for T13. The values stored in CC63R is compared to the counter value of T13. The State Bit CC63ST is located in register **CMPSTAT**.

CC63R
Compare Register for T13 **XSFR(34_H)** **Reset Value: 0000_H**

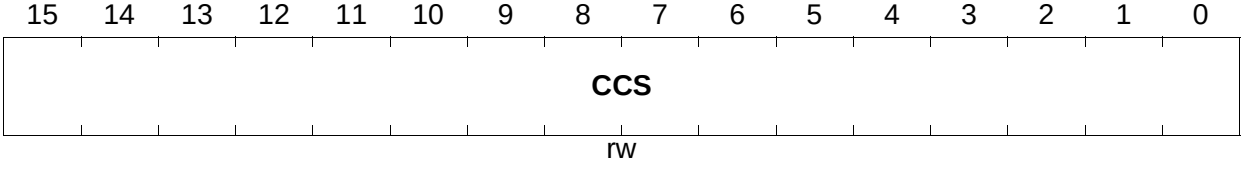


Field	Bits	Type	Description
CCV	[15:0]	rh	**Channel CC63 Compare Value** The bit field CCV contains the value, that is compared to the T13 counter value.

18.3.6.4 Compare Shadow Register

The register CC63R can only be read by SW, the modification of the value is done by a shadow register transfer from register CC63SR. The corresponding shadow register CC63SR can be read and written by SW.

CC63SR
Compare Shadow Register for T13
XSFR(36_H) **Reset Value: 0000_H**



Field	Bits	Type	Description
CCS	[15:0]	rw	**Shadow Register for Channel CC63 Compare Value** The bit field contents of CCS is transferred to the bit field CCV during a shadow transfer.

18.4 Trap Handling

The trap functionality permits the PWM outputs to react on the state of the input signal CTRAP. This functionality can be used to switch off the power devices if the trap input becomes active (e.g. to perform an emergency stop). The trap handling and the effect on the output modulation are controlled by the bits in the trap control register **TRPCTR**. The trap flags TRPF and TRPS are located in register **IS** and can be set/cleared by SW by writing to registers **ISS** and **ISR**.
Figure 18-33 gives an overview on the trap function.

The Trap Flag TRPF monitors the trap input and initiates the entry into the Trap State. The Trap State Bit TRPS determines the effect on the outputs and controls the exit of the Trap State.
When a trap condition is detected ($\overline{CTRAP}$ = 0) and the input is enabled (TRPPEN = 1), both, the Trap Flag TRPF and the Trap State Bit TRPS, are set to 1 (trap state active).
The output of the Trap State Bit TRPS leads to the Output Modulation Blocks (for T12 and for T13) and can there deactivate the outputs (set them to the passive state). Individual enable control bits for each of the six T12-related outputs and the T13-related output facilitate a flexible adaptation to the application needs.

There are a number of different ways to exit the Trap State. This offers SW the option to select the best operation for the application. Exiting the Trap State can be done either immediately when the trap condition is removed ($\overline{CTRAP}$ = 1 or TRPPEN = 0), or under software control, or synchronously to the PWM generated by either Timer T12 or Timer T13.

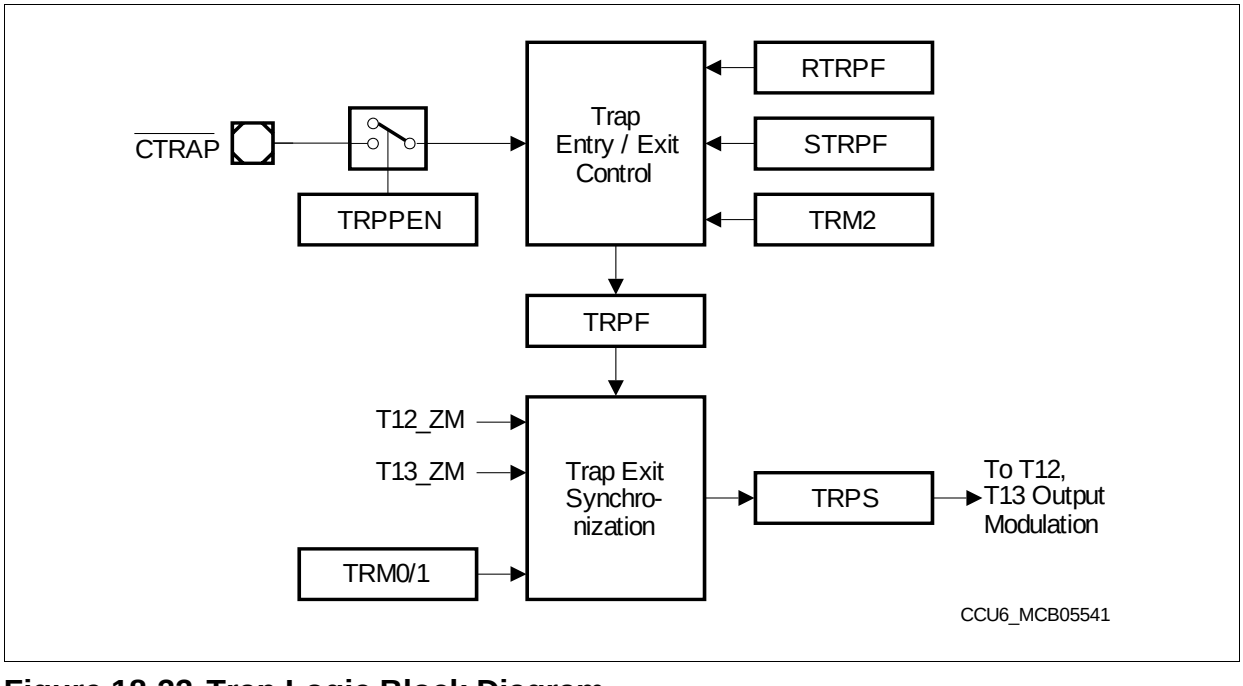


Figure 18-33 Trap Logic Block Diagram

Clearing of TRPF is controlled by the mode control bit TRPM2. If TRPM2 = 0, TRPF is automatically cleared by HW when $\overline{\text{CTRAP}}$ returns to the inactive level ($\overline{\text{CTRAP}}$ = 1) or if the trap input is disabled (TRPPEN = 0). When TRPM2 = 1, TRPF must be reset by SW after $\overline{\text{CTRAP}}$ has become inactive.

Clearing of TRPS is controlled by the mode control bits TRPM1 and TRPM0 (located in the Trap Control Register TRPCTR). A reset of TRPS terminates the Trap State and returns to normal operation. There are three options selected by TRPM1 and TRPM0. One is that the Trap State is left immediately when the Trap Flag TRPF is cleared, without any synchronization to timers T12 or T13. The other two options facilitate the synchronization of the termination of the Trap State to the count periods of either Timer T12 or Timer T13. **Figure 18-34** gives an overview on the associated operation.

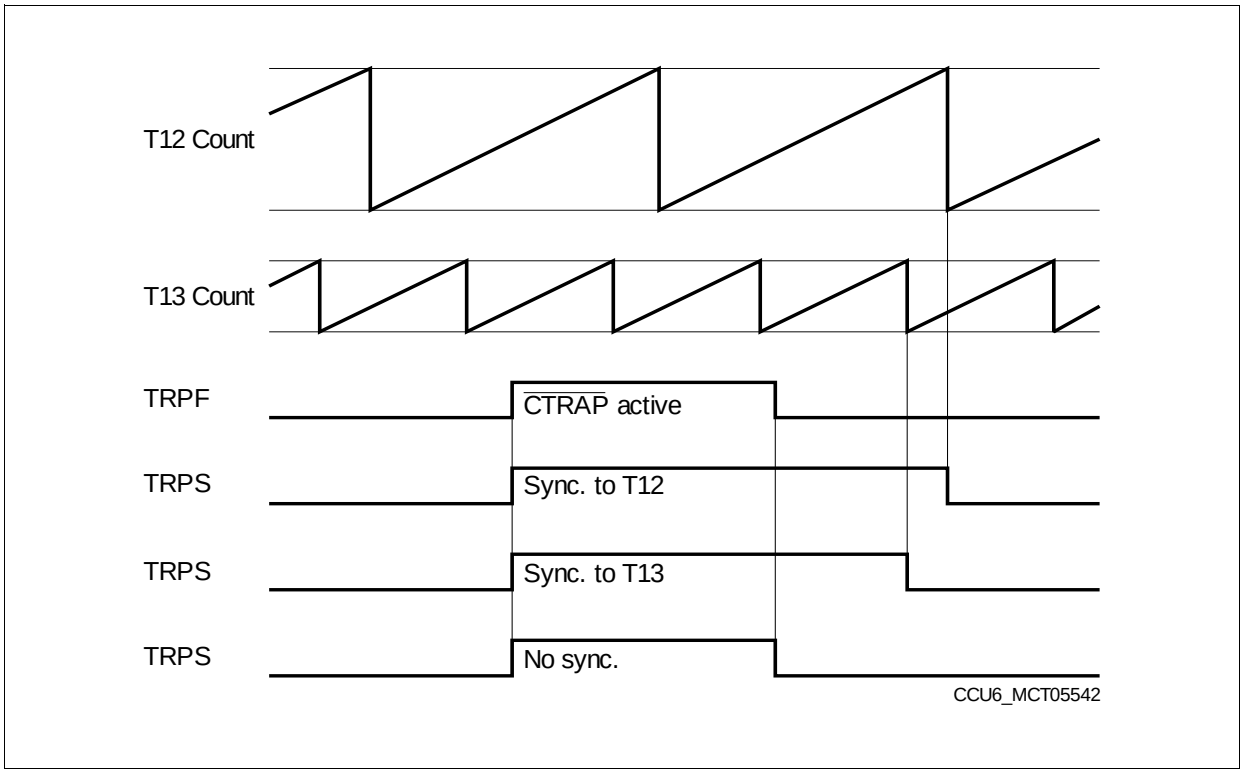


Figure 18-34 Trap State Synchronization (with TRM2 = 0)

18.5 Multi-Channel Mode

The Multi-Channel mode offers the possibility to modulate all six T12-related output signals with one instruction. The bits in bit field **MCMOUT**.MCMP are used to specify the outputs that may become active. If Multi-Channel mode is enabled (bit **MODCTR**.MCMEN = 1), only those outputs may become active, that have a 1 at the corresponding bit position in bit field MCMP.

This bit field has its own shadow bit field **MCMOUTS**.MCMPS, that can be written by software. The transfer of the new value in MCMPS to the bit field MCMP can be triggered by, and synchronized to, T12 or T13 events. This structure permits the software to write the new value, that is then taken into account by the hardware at a well-defined moment and synchronized to a PWM signal. This avoids unintended pulses due to unsynchronized modulation sources.

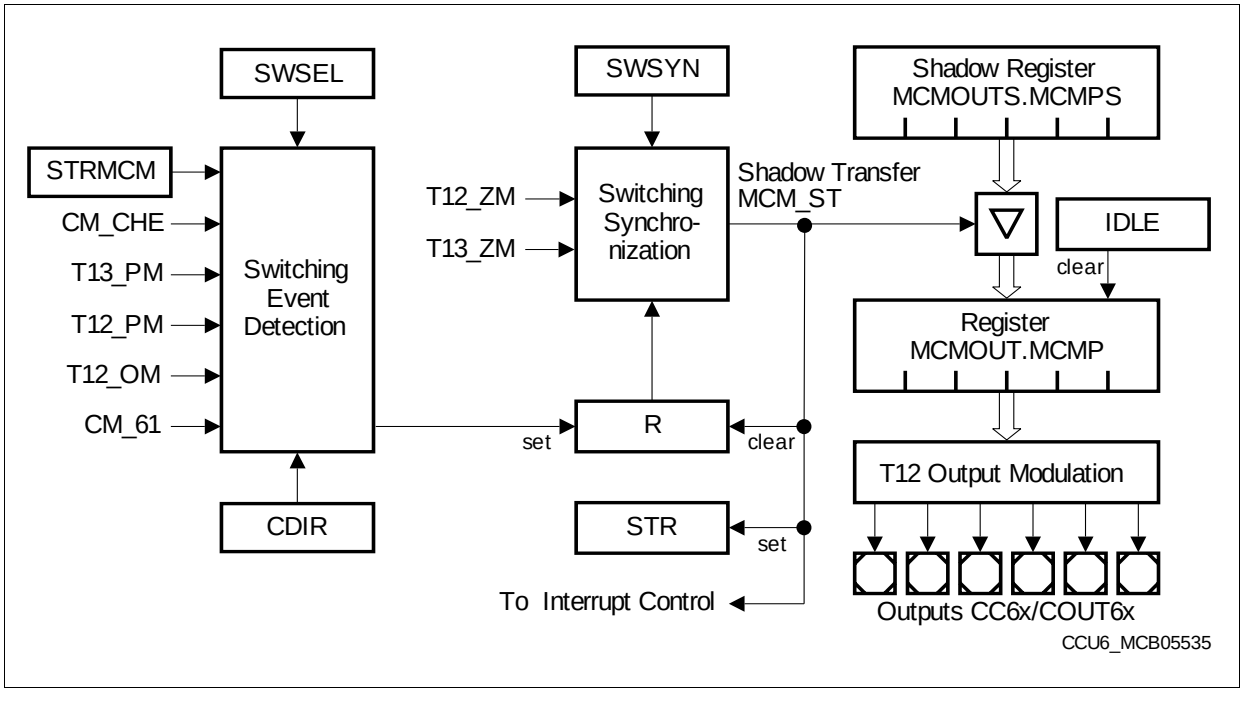


Figure 18-35 Multi-Channel Mode Block Diagram

Figure 18-35 shows the functional blocks for the Multi-Channel operation, controlled by bit fields in register **MCMCTR**. The event that triggers the update of bit field MCMP is chosen by SWSEL. In order to synchronize the update of MCMP to a PWM generated by T12 or T13, bit field SWSYN allows the selection of the synchronization event leading to the transfer from MCMPS to MCMP. Due to this structure, an update takes place with a new PWM period. A reminder flag R is set when the selected switching event occurs (the event is not necessarily synchronous to the modulating PWM), and is cleared when the transfer takes place. This flag can be monitored by software to check for the status of this logic block.

If the shadow transfer from MCMPS to MCMP takes place, bit **IS**.STR becomes set and an interrupt can be generated.

If it is explicitly desired, the update takes place immediately with the occurrence of the selected event when the direct synchronization mode is selected. The update can also be requested by software by writing to bit field MCMPS with the shadow transfer request bit STRMCM = 1. The option to trigger an update by SW is possible for all settings of SWSEL.
By using the direct mode and bit STRMCM = 1, the update takes place completely under software control.
The event selection and synchronization options are summarized in **Table 18-9** and **Table 18-10**.

Table 18-9 Multi-Channel Mode Switching Event Selection

SWSEL	Selected Event (see register MCMCTR)
000_B	No automatic event detection
001_B	Correct Hall Event (CM_CHE) detected at input signals CCPOSx without additional delay
010_B	T13 Period-Match (T13_PM)
011_B	T12 One-Match while counting down (T12_OM and CDIR = 1)
100_B	T12 Compare Channel 1 Event while counting up (CM_61 and CDIR = 0) to support the phase delay function by CC61 for block commutation mode.
101_B	T12 Period-Match while counting up (T12_PM and CDIR = 0)
110_B, 111_B	Reserved, no action

Table 18-10 Multi-Channel Mode Switching Synchronization

SWSYN	Synchronization Event (see register MCMCTR)
00_B	Direct Mode: the trigger event directly causes the shadow transfer
01_B	T13 Zero-Match (T13_ZM), the MCM shadow transfer is synchronized to a T13 PWM
10_B	T12 Zero-Match (T12_ZM), the MCM shadow transfer is synchronized to a T12 PWM
11_B	Reserved, no action

18.6 Hall Sensor Mode

For Brushless DC-Motors in block commutation mode, the Multi-Channel Mode has been introduced to provide efficient means for switching pattern generation. These patterns need to be output in relation to the angular position of the motor. For this, usually Hall sensors or Back-EMF sensing are used to determine the angular rotor position. The CCU6 provides three inputs, CCPOS0, CCPOS1, and CCPOS2, that can be used as inputs for the Hall sensors or the Back-EMF detection signals.

There is a strong correlation between the motor position and the output modulation pattern. When a certain position of the motor has been reached, indicated by the sampled Hall sensor inputs (the Hall pattern), the next, pre-determined Multi-Channel Modulation pattern has to be output. Because of different machine types, the modulation pattern for driving the motor can vary. Therefore, it is wishful to have a wide flexibility in defining the correlation between the Hall pattern and the corresponding Modulation pattern. Furthermore, a hardware mechanism significantly reduces the CPU for block-commutation.

The CCU6 offers the flexibility by having a register containing the currently assumed Hall pattern (CURH), the next expected Hall pattern (EXPH) and the corresponding output pattern (MCMP). A new Modulation pattern is output when the sampled Hall inputs match the expected ones (EXPH). To detect the next rotation phase (segment for block commutation), the CCU6 monitors the Hall inputs for changes. When the next expected Hall pattern is detected, the next corresponding Modulation pattern is output.

To increase for noise immunity (to a certain extend), the CCU6 offers the possibility to introduce a sampling delay for the Hall inputs. Some changes of the Hall inputs are not leading to the expected Hall pattern, because they are only short spikes due to noise. The Hall pattern compare logic compares the Hall inputs to the next expected pattern and also to the currently assumed pattern to filter out spikes.

For the Hall and Modulation output patterns, a double-register structure is implemented. While register **MCMOUT** holds the actually used values, its shadow register **MCMOUTS** can be loaded by software from a pre-defined table, holding the appropriate Hall and Modulation patterns for the given motor control.
A transfer from the shadow register into register MCMOUT can take place when a correct Hall pattern change is detected. Software can then load the next values into register MCMOUTS. It is also possible by software to force a transfer from MCMOUTS into MCMOUT.

Note: The Hall input signals CCPOSx and the CURH and EXPH bit fields are arranged in the following order:
CCPOS0 corresponds to CURH.0 (LSB) and EXPH.0 (LSB)
CCPOS1 corresponds to CURH.1 and EXPH.1
CCPOS2 corresponds to CURH.2 (MSB) and EXPH.2 (MSB)

18.6.1 Hall Pattern Evaluation

The Hall sensor inputs CCPOSx can be permanently monitored via an edge detection block (with the module clock f_{CC6}). In order to suppress spikes on the Hall inputs due to noise in rugged inverter environment, two optional noise filtering methods are supported by the Hall logic (both methods can be combined).

- Noise filtering with delay:
  For this function, the mode control bit fields MSEL6x for all T12 compare channels must be programmed to 1000_B and DBYP = 0. The selected event triggers Dead-Time Counter 0 to generate a programmable delay (defined by bit field DTM). When the delay has elapsed, the evaluation signal HCRDY becomes activated.
  Output modulation with T12 PWM signals is not possible in this mode.
- Noise filtering by synchronization to PWM:
  The Hall inputs are not permanently monitored by the edge detection block, but samples are taken only at defined points in time during a PWM period. This can be used to sample the Hall inputs when the switching noise (due to PWM) does not disturb the Hall input signals.

If neither the delay function of Dead-Time Counter 0 is not used for the Hall pattern evaluation nor the Hall mode for Brushless DC-Drive control is enabled, the timer T12 block is available for PWM generation and output modulation.

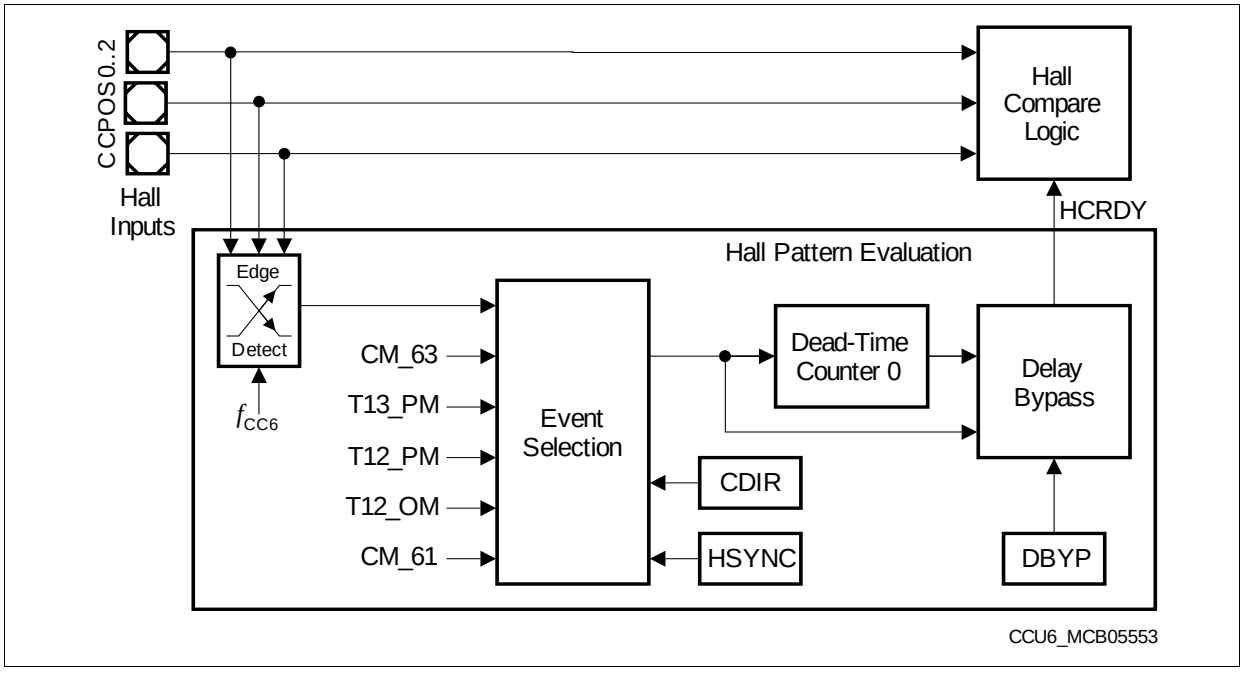


Figure 18-36 Hall Pattern Evaluation

If the evaluation signal HCRDY (Hall Compare Ready, see **Figure 18-37**) becomes activated, the Hall inputs are sampled and the Hall compare logic starts the evaluation of the Hall inputs.

Figure 18-36 illustrates the events for Hall pattern evaluation and the noise filter logic, **Table 18-11** summarizes the selectable trigger input signals.

Table 18-11 Hall Sensor Mode Trigger Event Selection

HSYNC	Selected Event (see register T12MSEL)
000_B	Any edge at any of the inputs CCPOSx, independent from any PWM signal (permanent check).
001_B	A T13 Compare-Match (CM_63).
010_B	A T13 Period-Match (T13_PM).
011_B	Hall sampling triggered by HW sources is switched off.
100_B	A T12 Period-Match while counting up (T12_PM and CDIR = 0).
101_B	A T12 One-Match while counting down (T12_OM and CDIR = 1).
110_B	A T12 Compare-Match of compare channel CC61 while counting up (CM_61 and CDIR = 0).
111_B	A T12 Compare-Match of compare channel CC61 while counting down (CM_61 and CDIR = 1).

18.6.2 Hall Pattern Compare Logic

Figure 18-37 gives an overview on the double-register structure and the pattern compare logic. Software writes the next modulation pattern (MCMPS) and the corresponding current (CURHS) and expected (EXPHS) Hall patterns into the shadow register MCMOUTS. Register MCMOUT holds the actually used values CURH and EXPH. The modulation pattern MCMP is provided to the T12 Output Modulation block. The current (CURH) and expected (EXPH) Hall patterns are compared to the sampled Hall sensor inputs (visible in register **CMPSTAT**). Sampling of the inputs and the evaluation of the comparator outputs is triggered by the evaluation signal HCRDY (Hall Compare Ready), that is detailed in the next section.

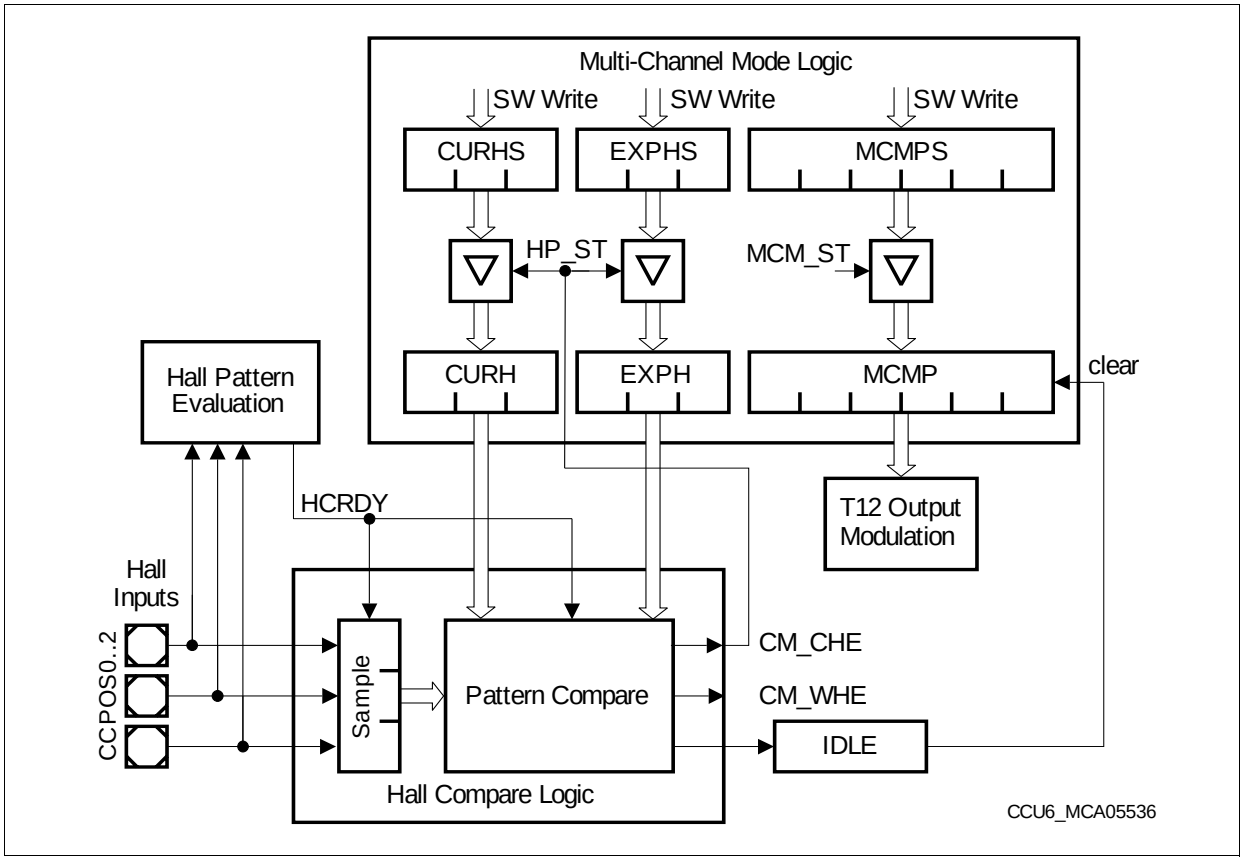


Figure 18-37 Hall Pattern Compare Logic

- If the sampled Hall pattern matches the value programmed in CURH, the detected transition was a spike (no Hall event) and no further actions are necessary.
- If the sampled Hall pattern matches the value programmed in EXPH, the detected transition was the expected event (correct Hall event CM_CHE) and the MCMP value has to change.
- If the sampled Hall pattern matches neither CURH nor EXPH, the transition was due to a major error (wrong Hall event CM_CWE) and can lead to an emergency shut down (IDLE).

At every correct Hall event (CM_CHE), the next Hall patterns are transferred from the shadow register MCMOUTS into MCMOUT (Hall pattern shadow transfer HP_ST), and a new Hall pattern with its corresponding output pattern can be loaded (e.g. from a predefined table in memory) by software into MCMOUTS. For the Modulation patterns, signal MCM_ST is used to trigger the transfer.
Loading this shadow register can also be done by writing MCMOUTS.STRHP = 1 (for EXPH and CURH) or MCMOUTS.STRMCMP = 1 (for MCMP).

18.6.3 Hall Mode Flags

Depending on the Hall pattern compare operation, a number of flags are set in order to indicate the status of the module and to trigger further actions and interrupt requests.

Flag **IS**.CHE (Correct Hall Event) is set by signal CM_CHE when the sampled Hall pattern matches the expected one (EXPH). This flag can also be set by SW by setting bit **ISS**.SCHE = 1. If enabled by bit **IEN**.ENCHE = 1, the set signal for CHE can also generate an interrupt request to the CPU. Bit field **INP**.INPCHE defines which service request output becomes activated in case of an interrupt request.To clear flag CHE, SW needs to write **ISR**.RCHE = 1.
Flag IS.WHE indicates a Wrong Hall Event. Its handling for flag setting and resetting as well as interrupt request generation are similar to the mechanism for flag CHE.
The implementation of flag STR is done in the same way as for CHE and WHE. This flag is set by HW by the shadow transfer signal MCM_ST (see also **Figure 18-35**).

Please note that for flags CHE, WHE, and STR, the interrupt request generation is triggered by the set signal for the flag. That means, a request can be generated even if the flag is already set. There is no need to clear the flag in order to enable further interrupt requests.

The implementation for the IDLE flag is different. It is set by HW through signal CM_WHE if enabled by bit ENIDLE. Software can also set the flag via bit SIDLE. As long as bit IDLE is set, the modulation pattern field MCMP is cleared to force the outputs to the passive state. Flag IDLE must be cleared by software by writing RIDLE = 1 in order to return to normal operation. To fully restart from IDLE mode, the transfer requests for the bit fields in register MCMOUTS to register MCMOUT have to be initiated by software via bits STRMCM and STRHP in register MCMOUTS. In this way, the release from IDLE mode is under software control, but can be performed synchronously to the PWM signal.

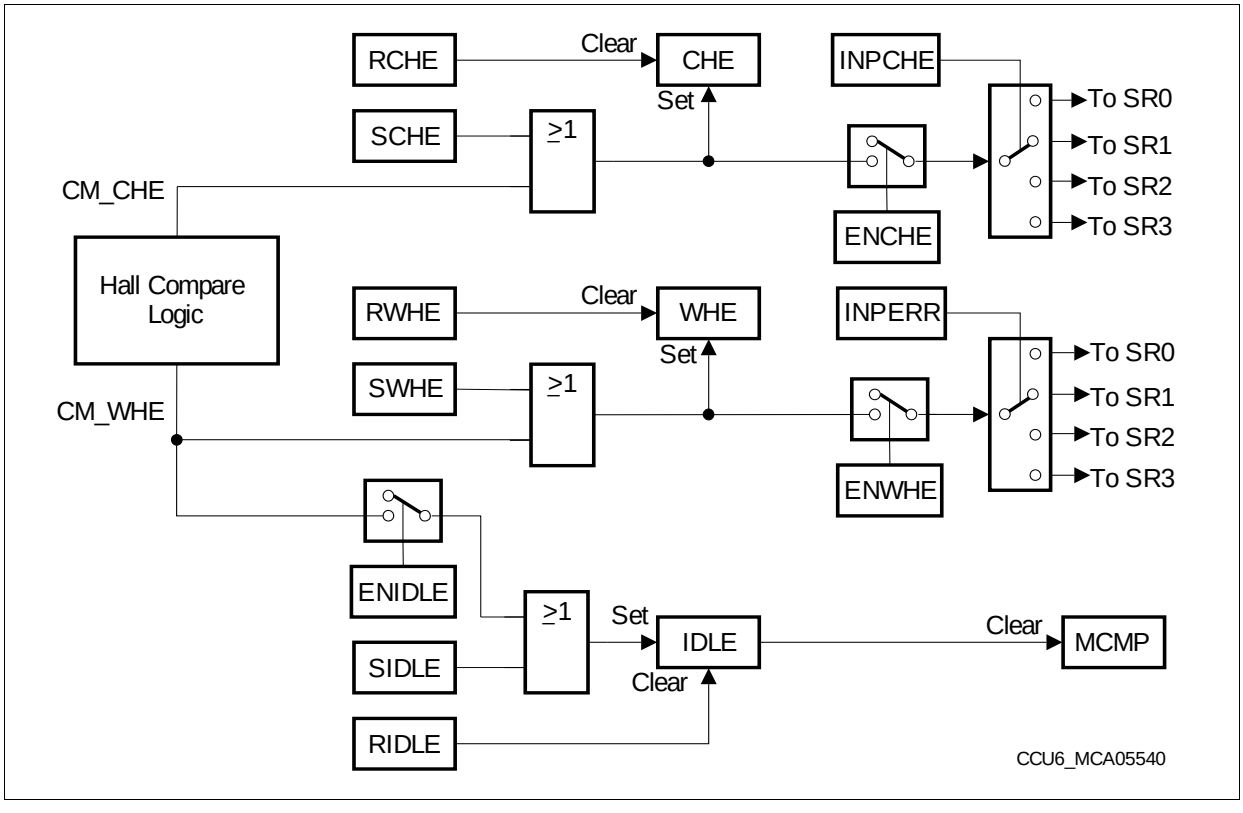


Figure 18-38 Hall Mode Flags

18.6.4 Hall Mode for Brushless DC-Motor Control

The CCU6 provides a mode for the Timer T12 Block especially targeted for convenient control of block commutation patterns for Brushless DC-Motors. This mode is selected by setting all **T12MSEL**.MSEL6x bit fields of the three T12 Channels to 1000_B.
In this mode, illustrated in **Figure 18-39**, channel CC60 is placed in capture mode to measure the time elapsed between the last two correct Hall events, channel CC61 in compare mode to provide a programmable phase delay between the Hall event and the application of a new PWM output pattern, and channel CC62 also in compare mode as first time-out criterion. A second time-out criterion can be built by the T12 period match event.

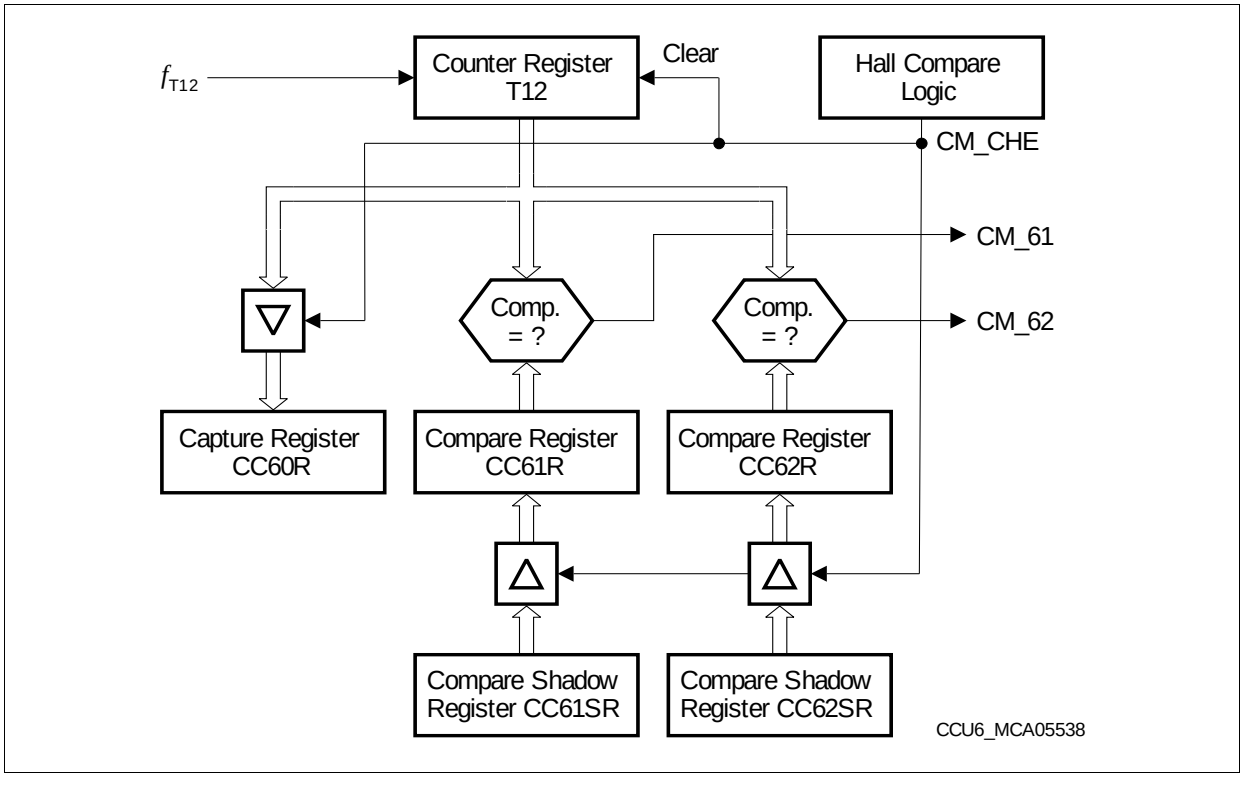


Figure 18-39 T12 Block in Hall Sensor Mode

The signal CM_CHE from the Hall compare logic is used to transfer the new compare values from the shadow registers CC6xSR into the actual compare registers CC6xR, performs the shadow transfer for the T12 period register, to capture the current T12 contents into register CC60R, and to clear T12.

Note: In this mode, the shadow transfer signal T12_ST is not generated. Not all shadow bits, such as the PSLy bits, will be transferred to their main registers. To program the main registers, SW needs to write to these registers while Timer T12 is stopped. In this case, a SW write actualizes both registers.

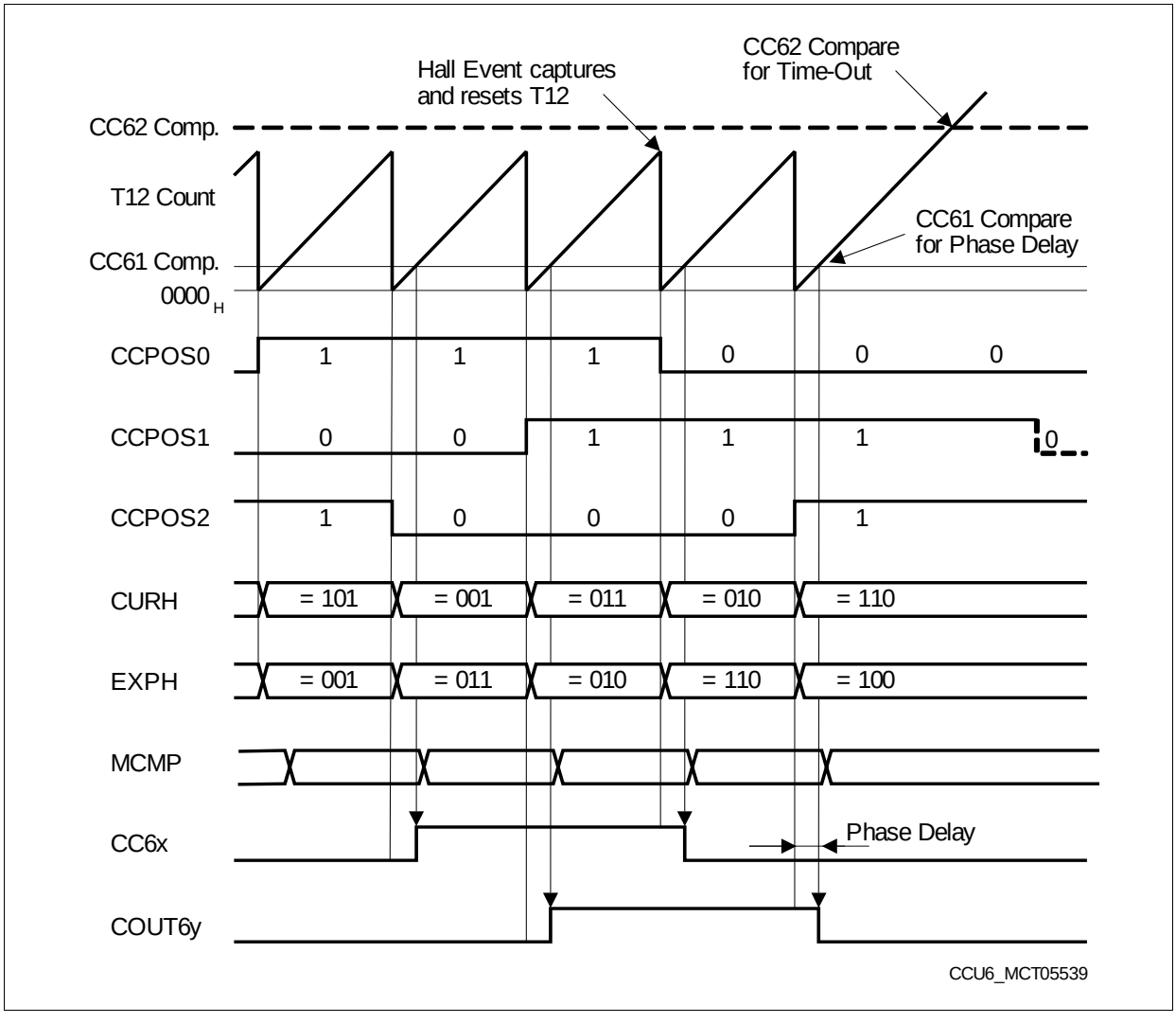


Figure 18-40 Brushless DC-Motor Control Example (all MSEL6x = 1000_B)

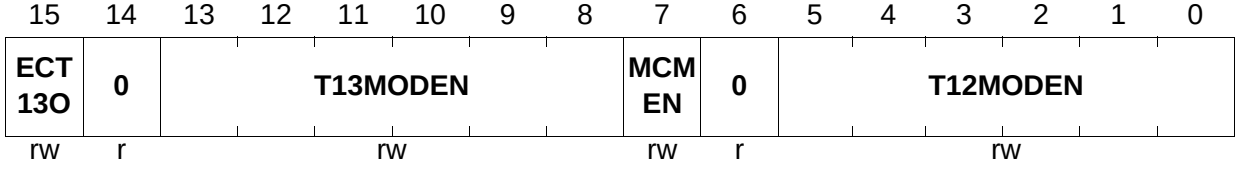
After the detection of an expected Hall pattern (CM_CHE active), the T12 count value is captured into channel CC60 (representing the actual rotor speed by measuring the elapsed time between the last two correct Hall events), and T12 is reset. When the timer reaches the compare value in channel CC61, the next multi-channel state is switched by triggering the shadow transfer of bit field MCMP (if enabled in bit field **SWEN**). This trigger event can be combined with the synchronization of the next multi-channel state to the PWM source (to avoid spikes on the output lines, see **Section 18.5**). This compare function of channel CC61 can be used as a phase delay from the position sensor input signals to the switching of the output signals, that is necessary if a sensorless back-EMF technique or Hall sensors are used. The compare value in channel CC62 can be used as a time-out trigger (interrupt), indicating that the actual motor speed is far below the desired destination value. An abnormal load change can be detected with this feature and PWM generation can be disabled.

18.7 Modulation Control Registers

18.7.1 Modulation Control

This register contains bits enabling the modulation of the corresponding output signal by PWM pattern generated by the timers T12 and T13. Furthermore, the multi-channel mode can be enabled as additional modulation source for the output signals.

MODCTR
Modulation Control Register **XSFR(40_H)** **Reset Value: 0000_H**



Field	Bits	Type	Description
T12MODEN	[5:0]	rw	**T12 Modulation Enable** These bits enable the modulation of the corresponding output signal by a PWM pattern generated by timer T12. T12MODEN0 = MODCTR.0 for output CC60 T12MODEN1 = MODCTR.1 for output COUT60 T12MODEN2 = MODCTR.2 for output CC61 T12MODEN3 = MODCTR.3 for output COUT61 T12MODEN4 = MODCTR.4 for output CC62 T12MODEN5 = MODCTR.5 for output COUT62 0_B The modulation of the corresponding output signal by a T12 PWM pattern is disabled. 1_B The modulation of the corresponding output signal by a T12 PWM pattern is enabled.
MCMEN	7	rw	**Multi-Channel Mode Enable** 0_B The modulation of the corresponding output signal by a multi-channel pattern according to bit field MCMOUT is disabled. 1_B The modulation of the corresponding output signal by a multi-channel pattern according to bit field MCMOUT is enabled.

Field	Bits	Type	Description
T13MODEN	[13:8]	rw	**T13 Modulation Enable** These bits enable the modulation of the corresponding output signal by the PWM pattern CC63_O generated by timer T13. T13MODEN0 = MODCTR.8 for output CC60 T13MODEN1 = MODCTR.9 for output COUT60 T13MODEN2 = MODCTR.10 for output CC61 T13MODEN3 = MODCTR.11 for output COUT61 T13MODEN4 = MODCTR.12 for output CC62 T13MODEN5 = MODCTR.13 for output COUT62 0_B The modulation of the corresponding output signal by a T13 PWM pattern is disabled. 1_B The modulation of the corresponding output signal by a T13 PWM pattern is enabled.
ECT13O	15	rw	**Enable Compare Timer T13 Output** 0_B The output COUT63 is in the passive state. 1_B The output COUT63 is enabled for the PWM signal generated by T13.
0	6, 14	r	**reserved;** returns 0 if read; should be written with 0;

18.7.2 Trap Control Register

The register TRPCTR controls the trap functionality. It contains independent enable bits for each output signal and control bits to select the behavior in case of a trap condition. The trap condition is a low level on the CTRAP input pin, that is monitored (inverted level) by bit IS.TRPF. While TRPF=1 (trap input active), the trap state bit IS.TRPS is set to 1.

TRPCTR
Trap Control Register **XSFR(42_H)** **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRPPEN	**TRPEN13**	**TRPEN**						**0**					**TRPM2**	**TRPM1**	**TRPM0**
rw	rw	rw						r					rw	rw	rw

Field	Bits	Type	Description
TRPM1, TRPM0	1, 0	rw	**Trap Mode Control Bits 1, 0** These two bits define the behavior of the selected outputs when leaving the trap state after the trap condition has become inactive again. A synchronization to the timer driving the PWM pattern avoids unintended pulses when leaving the trap state. The combination [TRPM1, TRPM0] leads to: 00_B The trap state is left (return to normal operation) after TRPF has become 0 again when a zero-match of T12 (while counting up) is detected (synchronization to T12). 01_B The trap state is left (return to normal operation) after TRPF has become 0 again when a zero-match of T13 is detected (synchronization to T13). 10_B reserved 11_B The trap state is left (return to normal operation) immediately after TRPF has become 0 again without any synchronization to T12 or T13.

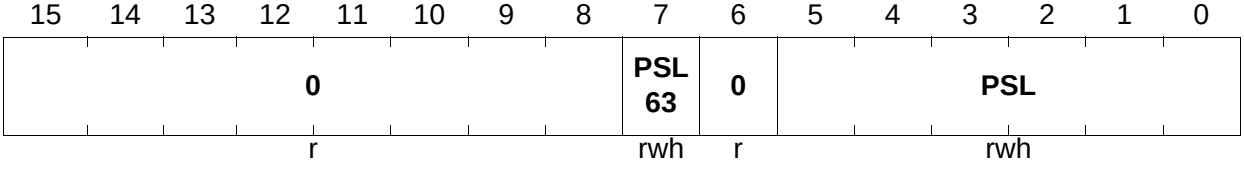
Field	Bits	Type	Description
TRPM2	2	rw	**Trap Mode Control Bit 2** This bit defines how the trap flag TRPF can be cleared after the trap input condition ($\overline{CTRAP}$ = 0 and TRPPEN = 1) is no longer valid (either by $\overline{CTRAP}$ = 1 or by TRPPEN = 0). 0_B Automatic Mode: Bit TRPF is cleared by HW if the trap input condition is no longer valid. 1_B Manual Mode: Bit TRPF stays 0 after the trap input condition is no longer valid. It has to be cleared by SW by writing ISR.RTRPF = 1.
TRPEN	[13:8]	rw	**Trap Enable Control** Setting a bit enables the trap functionality for the following corresponding output signals: TRPEN0 = TRPCTR.8 for output CC60 TRPEN1 = TRPCTR.9 for output COUT60 TRPEN2 = TRPCTR.10 for output CC61 TRPEN3 = TRPCTR.11 for output COUT61 TRPEN4 = TRPCTR.12 for output CC62 TRPEN5 = TRPCTR.13 for output COUT62 0_B The trap functionality of the corresponding output signal is disabled. The output state is independent from bit IS.TRPS. 1_B The trap functionality of the corresponding output signal is enabled. The output state is set to the passive while IS.TRPS=1.
TRPEN13	14	rw	**Trap Enable Control for Timer T13** 0_B The trap functionality for output COUT63 is disabled. The output state is independent from bit IS.TRPS. 1_B The trap functionality for output COUT63 is enabled. The output state is set to the passive while IS.TRPS=1.

Field	Bits	Type	Description
TRPPEN	15	rw	**Trap Pin Enable** This bit enables the input (pin) function for the trap generation. An interrupt can only be generated if a falling edge is detected at pin CTRAP while TRPPEN = 1. 0_B The CCU6 trap functionality based on the input CTRAP is disabled. A CCU6 trap can only be generated by SW by setting bit TRPF. 1_B The CCU6 trap functionality based on the input CTRAP is enabled. A CCU6 trap can be generated by SW by setting bit TRPF or by CTRAP=0.
0	[7:3]	r	**reserved;** returns 0 if read; should be written with 0;

18.7.3 Passive State Level Register

Register PSLR defines the passive state level of the PWM outputs of the module. The passive state level is the value that is driven during the passive state of the output. During the active state, the corresponding output pin drives the active state level, that is the inverted passive state level. The passive state level permits to adapt the driven output levels to the driver polarity (inverted, not inverted) of the connected power stage. The bits in this register have shadow bit fields to permit a concurrent update of all PWM-related parameters (bit field PSL is updated with T12_ST, whereas PSL63 is updated with T13_ST). The actually used values can be read (attribute “rh”), whereas the shadow bits can only be written (attribute “w”).

PSLR
Passive State Level Register **XSFR(44_H)** **Reset Value: 0000_H**



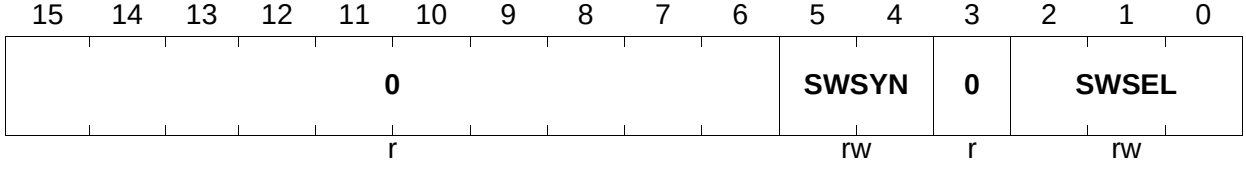
Field	Bits	Type	Description
PSL	[5:0]	rwh	**Compare Outputs Passive State Level** These bits define the passive level driven by the module outputs during the passive state. PSL0 = PSLR.0 for output CC60 PSL1 = PSLR.1 for output COUT60 PSL2 = PSLR.2 for output CC61 PSL3 = PSLR.3 for output COUT61 PSL4 = PSLR.4 for output CC62 PSL5 = PSLR.5 for output COUT62 0_B The passive level is 0. 1_B The passive level is 1.
PSL63	7	rwh	**Passive State Level of Output COUT63** This bit defines the passive level driven by the module output COUT63 during the passive state. 0_B The passive level is 0. 1_B The passive level is 1.
0	6, [15:8]	r	**reserved;** returns 0 if read; should be written with 0;

18.7.4 Multi-Channel Mode Registers

Register MCMCTR contains control bits for the multi-channel functionality.

MCMCTR
Multi-Channel Mode Control Register

XSFR($4E_H$) **Reset Value: 0000_H**



Field	Bits	Type	Description
SWSEL	[2:0]	rw	**Switching Selection** Bit field SWSEL selects one of the following trigger request sources (next multi-channel event) for the shadow transfer MCM_ST from MCMPS to MCMP. The trigger request is stored in the reminder flag R until the shadow transfer is done and flag R is cleared automatically with the shadow transfer. The shadow transfer takes place synchronously with an event selected in bit field SWSYN. 000_B No trigger request will be generated 001_B Correct Hall pattern detected (CM_CHE) 010_B T13 period-match detected (while counting up) 011_B T12 one-match (while counting down) 100_B T12 channel 1 compare-match detected (phase delay function) 101_B T12 period match detected (while counting up) else reserved, no trigger request will be generated

Field	Bits	Type	Description
SWSYN	[5:4]	rw	**Switching Synchronization** Bit field SWSYN defines the synchronization mechanism of the shadow transfer event MCM_ST if it has been requested before (flag R set by an event selected by SWSEL) and if MCMEN = 1. This feature permits the synchronization of the outputs to the PWM source, that is used for modulation (T12 or T13). 00_B Direct; the trigger event immediately leads to the shadow transfer 01_B T13 zero-match triggers the shadow transfer 10_B a T12 zero-match (while counting up) triggers the shadow transfer 11_B reserved; no action
0	3, [15:6]	r	**reserved;** returns 0 if read; should be written with 0;

Register MCMOUTS contains bits used as pattern input for the multi-channel mode and the Hall mode. This register is a shadow register (that can be read and written) for register MCMOUT, indicating the currently active signals.

MCMOUTS
Multi-Channel Mode Output Shadow Register
XSFR($4A_H$) **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STRHP	0	CURHS			EXPHS			STRMCM	0	MCMPS					
w	r	rw			rw			w	r	rw					

Field	Bits	Type	Description
MCMPS	[5:0]	rw	**Multi-Channel PWM Pattern Shadow** Bit field MCMPS is the shadow bit field for bit field MCMP. The multi-channel shadow transfer is triggered by MCM_ST according to the transfer conditions defined by register MCMCTR.
STRMCM	7	w	**Shadow Transfer Request for MCMPS** Writing STRMCM = 1 leads to an immediate activation of MCM_ST to update bit field MCMP by the value of MCMPS. When read, this bit always delivers 0. 0_B No action. 1_B Bit field MCMP is updated.
EXPHS	[10:8]	rw	**Expected Hall Pattern Shadow** Bit field EXPHS is the shadow bit field for bit field EXPH. The shadow transfer takes place when a correct Hall event is detected (CM_CHE).
CURHS	[13:11]	rw	**Current Hall Pattern Shadow** Bit field CURHS is the shadow bit field for bit field CURH. The shadow transfer takes place when a correct Hall event is detected (CM_CHE).

Field	Bits	Type	Description
STRHP	15	w	**Shadow Transfer Request for the Hall Pattern** Writing STRHP = 1 leads to an immediate activation of HP_ST to update bit fields EXPH and CURH by EXPHS and CURHS. When read, this bit always delivers 0. 0_B No action. 1_B Bit fields EXPH and CURH are updated.
0	6, 14	r	**reserved;** returns 0 if read; should be written with 0;

MCMOUT
Multi-Channel Mode Output Register

XSFR($4C_H$) **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		**CURH**			**EXPH**			**0**	**R**	**MCMP**					
r		rh			rh			r	rh	rh					

Field	Bits	Type	Description
MCMP	[5:0]	rh	**Multi-Channel PWM Pattern** Bit field MCMP defines the output pattern for the multi-channel mode. If this mode is enabled by MODCTR.MCMEN = 1, the output state of all T12 related PWM outputs can be modified. This bit field is 0 while IS.IDLE = 1. MCMP0 = MCMOUT.0 for output CC60 MCMP1 = MCMOUT.1 for output COUT60 MCMP2 = MCMOUT.2 for output CC61 MCMP3 = MCMOUT.3 for output COUT61 MCMP4 = MCMOUT.4 for output CC62 MCMP5 = MCMOUT.5 for output COUT62 0_B The output is set to the passive state. A PWM generated by T12 or T13 are not taken into account. 1_B The output can be in the active state, depending on the enabled PWM modulation signals generated by T12, T13 and the trap state.

Field	Bits	Type	Description
R	6	rh	**Reminder Flag** This flag indicates that the shadow transfer from MCMPS to MCMP has been requested by the selected trigger source. It is cleared when the shadow transfer takes place or while MCMEN=0. 0_B A shadow transfer MCM_ST is not requested. 1_B A shadow transfer MCM_ST is requested, but has not yet been executed, because the selected synchronization condition has not yet occurred.
EXPH	[10:8]	rh	**Expected Hall Pattern** Bit field EXPH is updated by a shadow transfer HP_ST from bit field EXPHS. If HCRDY = 1, EXPH is compared to the sampled CCPOSx inputs in order to detect the occurrence of the next desired (=expected) hall pattern or a wrong pattern. If the sampled hall pattern at the hall input pins is equal to bit field EXPH, a correct Hall event has been detected (CM_CHE).
CURH	[13:11]	rh	**Current Hall Pattern** Bit field CURH is updated by a shadow transfer HP_ST from bit field CURHS. If HCRDY = 1, CURH is compared to the sampled CCPOSx inputs in order to detect a spike. If the sampled Hall pattern at the Hall input pins is equal to bit field CURH, no Hall event has been detected. If the sampled Hall input pattern is neither equal to CURH nor equal to EXPH, the Hall event was not the desired one and may be due to a fatal error (e.g. blocked rotor, etc.). In this case, a wrong Hall event has been detected (CM_WHE).
0	7, [15:14]	r	**reserved;** returns 0 if read; should be written with 0;

18.8 Interrupt Handling

This section describes the interrupt handling of the CCU6 module.

18.8.1 Interrupt Structure

The HW interrupt event or the SW setting of the corresponding interrupt set bit (in register ISS) sets the event indication flags (in register IS) and can trigger the interrupt generation. The interrupt pulse is generated independently from the interrupt status flag in register IS (it is not necessary to clear the related status bit to be able to generate another interrupt). The interrupt flag can be cleared by SW by writing to the corresponding bit in register ISR.

If enabled by the related interrupt enable bit in register IEN, an interrupt pulse can be generated on one of the four service request outputs (SR0 to SR3) of the module. If more than one interrupt source is connected to the same interrupt node pointer (in register INP), the requests are logically OR-combined to one common service request output (see **Figure 18-41**).

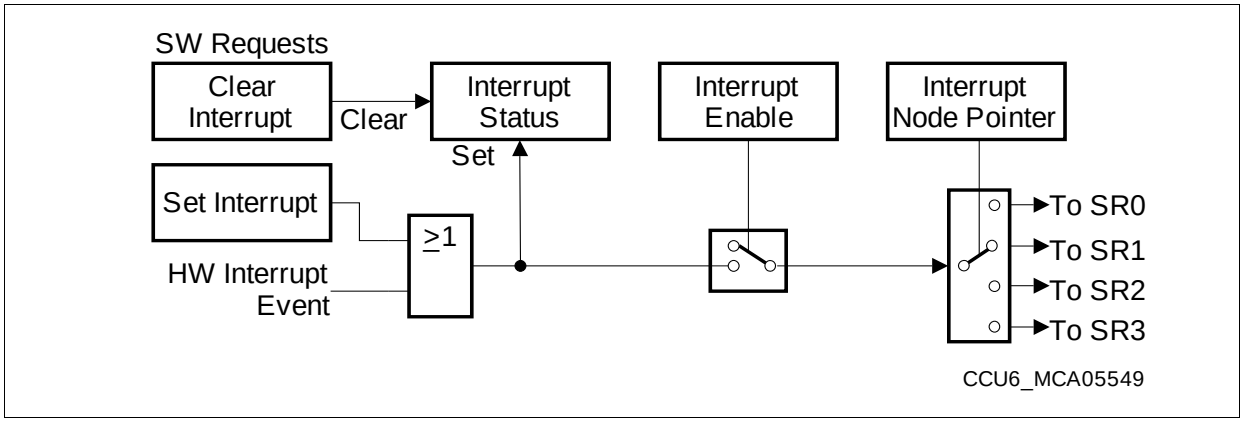


Figure 18-41 General Interrupt Structure

The available interrupt events in the CCU6 are shown in **Figure 18-42**.

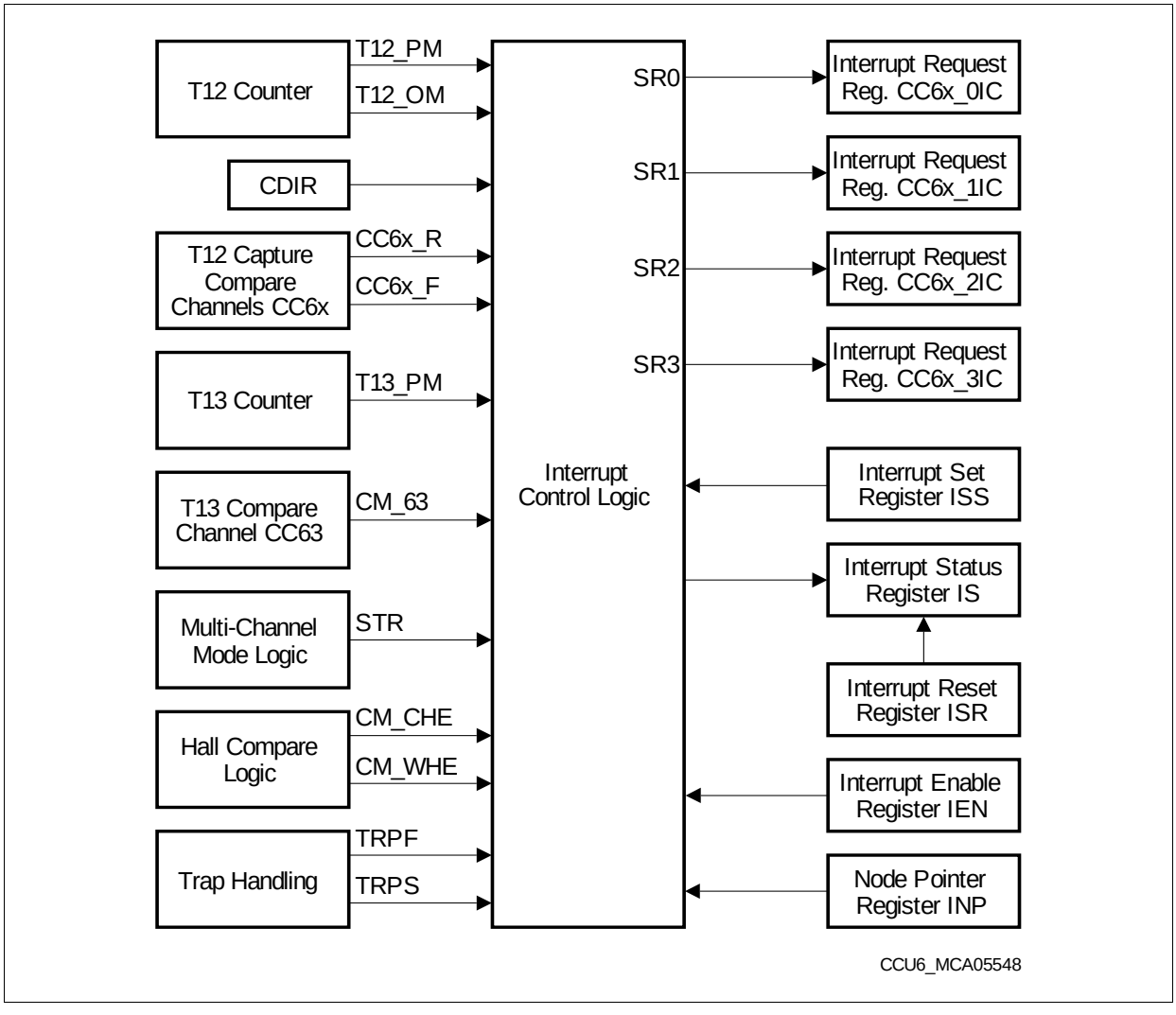


Figure 18-42 Interrupt Sources and Events

18.8.2 Interrupt Registers

18.8.2.1 Interrupt Status Register

Register IS contains the individual interrupt request bits. This register can only be read, write actions have no impact on the contents of this register. The SW can set or clear the bits individually by writing to the registers ISS (to set the bits) or to register ISR (to clear the bits).

The interrupt generation is independent from the value of the bits in register IS, e.g. the interrupt will be generated (if enabled) even if the corresponding bit is already set. The trigger for an interrupt generation is the detection of a set condition (by HW or SW) for the corresponding bit in register IS.

In compare mode (and hall mode), the timer-related interrupts are only generated while the timer is running (T1xR=1). In capture mode, the capture interrupts are also generated while the timer T12 is stopped.

Note: Not all bits in register IS can generate an interrupt. Other status bits have been added, that have a similar structure for their set and clear actions. It is recommended that SW checks the interrupt bits bit-wisely (instead of common OR over the bits).

IS
Interrupt Status Register **XSFR(50_H)** **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STR	**IDLE**	**WHE**	**CHE**	**TRPS**	**TRPF**	**T13PM**	**T13CM**	**T12PM**	**T12OM**	**ICC62F**	**ICC62R**	**ICC61F**	**ICC61R**	**ICC60F**	**ICC60R**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
ICC60R, ICC61R, ICC62R	0, 2, 4	rh	**Capture, Compare-Match Rising Edge Flag** This bit indicates that event CC6x_R has been detected. This event occurs in compare mode when a compare-match is detected while T12 is counting up (CM_6x and CDIR = 0) and in capture mode when a rising edge is detected at the related input CC6xIN. 0_B The event has not yet been detected. 1_B The event has been detected.

Field	Bits	Type	Description
ICC60F, ICC61F, ICC62F	1, 3, 5	rh	**Capture, Compare-Match Falling Edge Flag** This bit indicates that event CC6x_F has been detected. This event occurs in compare mode when a compare-match is detected while T12 is counting down (CM_6x and CDIR = 1) and in capture mode when a falling edge is detected at the related input CC6xIN. 0_B The event has not yet been detected. 1_B The event has been detected.
T12OM	6	rh	**Timer T12 One-Match Flag** This bit indicates that a timer T12 one-match while counting down (T12_OM and CDIR = 1) has been detected. 0_B The event has not yet been detected. 1_B The event has been detected.
T12PM	7	rh	**Timer T12 Period-Match Flag** This bit indicates that a timer T12 period-match while counting up (T12_PM and CDIR = 0) has been detected. 0_B The event has not yet been detected. 1_B The event has been detected.
T13CM	8	rh	**Timer T13 Compare-Match Flag** This bit indicates that a timer T13 compare-match (CM_63) has been detected. 0_B The event has not yet been detected. 1_B The event has been detected.
T13PM	9	rh	**Timer T13 Period-Match Flag** This bit indicates that a timer T13 period-match (T13_PM) has been detected. 0_B The event has not yet been detected. 1_B The event has been detected.
TRPF	10	rh	**Trap Flag** This bit indicates if a trap condition (input $\overline{\text{CTRAP}}$ = 0 or by SW) is / has been detected. If TRM2= 0, it becomes cleared automatically if $\overline{\text{CTRAP}}$ = 1 or TRPPEN = 0, whereas if TRM2 = 1, it has to be cleared by writing RTRPF = 1. 0_B The trap condition has not been detected. 1_B The trap condition is / has been detected.

Field	Bits	Type	Description
TRPS	11	rh	**Trap State**[1)] This bit indicates the actual trap state. It is set if TRPF = 1 and becomes cleared according to the mode selected in register TRPCTR. 0_B The trap state is not active. 1_B The trap state is active.
CHE	12	rh	**Correct Hall Event** This bit indicates that a correct Hall event (CM_CHE) has been detected. 0_B The event has not yet been detected. 1_B The event has been detected.
WHE	13	rh	**Wrong Hall Event** This bit indicates that a wrong Hall event (CM_WHE) has been detected. 0_B The event has not yet been detected. 1_B The event has been detected.
IDLE	14	rh	**IDLE State** If enabled by ENIDLE = 1, this bit is set together with bit WHE and it has to be cleared by SW. 0_B No action. 1_B Bit field MCMP is cleared, the selected outputs are set to passive state.
STR	15	rh	**Multi-Channel Mode Shadow Transfer Request** This bit indicates that a shadow transfer from MCMPS to MCMP (MCM_ST) has taken place. 0_B The event has not yet been detected. 1_B The event has been detected.

1) During the trap state, the selected outputs are set to the passive state. The logic level driven during the passive state is defined by the corresponding bit in register PSLR. Bits TRPS=1 and TRPF=0 can occur if the trap condition is no longer active but the selected synchronization has not yet taken place.

18.8.2.2 Interrupt Status Set Register

Register ISS contains individual interrupt request set bits to generate a CCU6 interrupt request by software. Writing a 1 sets the bit(s) in register IS at the corresponding bit position(s) and can generate an interrupt event (if available and enabled).
All bit positions read as 0.

ISS
Interrupt Status Set Register **XSFR(52_H)** **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
S STR	**S IDLE**	**S WHE**	**S CHE**	**S WHC**	**S TRPF**	**S T13PM**	**S T13CM**	**S T12PM**	**S T12OM**	**S CC62F**	**S CC62R**	**S CC61F**	**S CC61R**	**S CC60F**	**S CC60R**
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
SCC60R SCC61R, SCC62R	0, 2, 4	w	**Set Capture, Compare-Match Rising Edge Flag** 0_B No action 1_B Bit CC6xR will be set.
SCC60F, SCC61F, SCC62F	1, 3, 5	w	**Set Capture, Compare-Match Falling Edge Flag** 0_B No action 1_B Bit CC6xF will be set.
ST12OM	6	w	**Set Timer T12 One-Match Flag** 0_B No action 1_B Bit T12OM will be set.
ST12PM	7	w	**Set Timer T12 Period-Match Flag** 0_B No action 1_B Bit T12PM will be set.
ST13CM	8	w	**Set Timer T13 Compare-Match Flag** 0_B No action 1_B Bit T13CM will be set.
ST13PM	9	w	**Set Timer T13 Period-Match Flag** 0_B No action 1_B Bit T13PM will be set.
STRPF	10	w	**Set Trap Flag** 0_B No action 1_B Bits TRPF and TRPS will be set.

Field	Bits	Type	Description
SWHC	11	w	**Software Hall Compare** 0_B No action 1_B The Hall compare action is triggered.
SCHE	12	w	**Set Correct Hall Event Flag** 0_B No action 1_B Bit CHE will be set.
SWHE	13	w	**Set Wrong Hall Event Flag** 0_B No action 1_B Bit WHE will be set.
SIDLE	14	w	**Set IDLE Flag** 0_B No action 1_B Bit IDLE will be set.
SSTR	15	w	**Set STR Flag** 0_B No action 1_B Bit STR will be set.

18.8.2.3 Status Reset Register

Register ISR contains bits to individually clear the interrupt event flags by software. Writing a 1 clears the bit(s) in register IS at the corresponding bit position(s).
All bit positions read as 0.

ISR
Interrupt Status Reset Register **XSFR(54_H)** **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R STR	**R IDLE**	**R WHE**	**R CHE**	**0**	**R TRPF**	**R T13PM**	**R T13CM**	**R T12PM**	**R T12OM**	**R CC62F**	**R CC62R**	**R CC61F**	**R CC61R**	**R CC60F**	**R CC60R**
w	w	w	w	r	w	w	w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
RCC60R, RCC61R, RCC62R	0, 2, 4	w	**Reset Capture, Compare-Match Rising Edge Flag** 0_B No action 1_B Bit CC6xR will be cleared.
RCC60F, RCC61F, RCC62F	1, 3, 5	w	**Reset Capture, Compare-Match Falling Edge Flag** 0_B No action 1_B Bit CC6xF will be cleared.
RT12OM	6	w	**Reset Timer T12 One-Match Flag** 0_B No action 1_B Bit T12OM will be cleared.
RT12PM	7	w	**Reset Timer T12 Period-Match Flag** 0_B No action 1_B Bit T12PM IS will be cleared.
RT13CM	8	w	**Reset Timer T13 Compare-Match Flag** 0_B No action 1_B Bit T13CM will be cleared.
RT13PM	9	w	**Reset Timer T13 Period-Match Flag** 0_B No action 1_B Bit T13PM will be cleared.
RTRPF	10	w	**Reset Trap Flag** 0_B No action 1_B Bit TRPF will be cleared (not taken into account while input $\overline{\text{CTRAP}}$=0 and TRPPEN=1.

Field	Bits	Type	Description
RCHE	12	w	**Reset Correct Hall Event Flag** 0_B No action 1_B Bit CHE will be cleared.
RWHE	13	w	**Reset Wrong Hall Event Flag** 1_B No action 0_B Bit WHE will be cleared.
RIDLE	14	w	**Reset IDLE Flag** 0_B No action 1_B Bit IDLE will be cleared.
RSTR	15	w	**Reset STR Flag** 0_B No action 1_B Bit STR will be cleared.
0	11	r	**reserved;** returns 0 if read; should be written with 0;

18.8.2.4 Interrupt Enable Register

Register IEN contains the interrupt enable bits and a control bit to enable the automatic idle function in the case of a wrong hall pattern.

IEN
Interrupt Enable Register **XSFR(58_H)** **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENSTR	**ENIDLE**	**ENWHE**	**ENCHE**	**0**	**ENTRPF**	**ENT13PM**	**ENT13CM**	**ENT12PM**	**ENT12OM**	**ENCC62F**	**ENCC62R**	**ENCC61F**	**ENCC61R**	**ENCC60F**	**ENCC60R**
rw	rw	rw	rw	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ENCC60R, ENCC61R, ENCC62R	0, 2, 4	rw	**Capture, Compare-Match Rising Edge Interrupt Enable for Channel CC6x** 0_B No interrupt will be generated if the set condition for bit CC6xR in register IS occurs. 1_B An interrupt will be generated if the set condition for bit CC6xR in register IS occurs. The service request output that will be activated is selected by bit field INPCC6x.
ENCC60F, ENCC61F, ENCC62F	1, 3, 5	rw	**Capture, Compare-Match Falling Edge Interrupt Enable for Channel CC6x** 0_B No interrupt will be generated if the set condition for bit CC6xF in register IS occurs. 1_B An interrupt will be generated if the set condition for bit CC6xF in register IS occurs. The service request output that will be activated is selected by bit field INPCC6x.
ENT12OM	6	rw	**Enable Interrupt for T12 One-Match** 0_B No interrupt will be generated if the set condition for bit T12OM in register IS occurs. 1_B An interrupt will be generated if the set condition for bit T12OM in register IS occurs. The service request output that will be activated is selected by bit field INPT12.

Field	Bits	Type	Description
ENT12PM	7	rw	**Enable Interrupt for T12 Period-Match** 0_B No interrupt will be generated if the set condition for bit T12PM in register IS occurs. 1_B An interrupt will be generated if the set condition for bit T12PM in register IS occurs. The service request output that will be activated is selected by bit field INPT12.
ENT13CM	8	rw	**Enable Interrupt for T13 Compare-Match** 0_B No interrupt will be generated if the set condition for bit T13CM in register IS occurs. 1_B An interrupt will be generated if the set condition for bit T13CM in register IS occurs. The service request output that will be activated is selected by bit field INPT13.
ENT13PM	9	rw	**Enable Interrupt for T13 Period-Match** 0_B No interrupt will be generated if the set condition for bit T13PM in register IS occurs. 1_B An interrupt will be generated if the set condition for bit T13PM in register IS occurs. The service request output that will be activated is selected by bit field INPT13.
ENTRPF	10	rw	**Enable Interrupt for Trap Flag** 0_B No interrupt will be generated if the set condition for bit TRPF in register IS occurs. 1_B An interrupt will be generated if the set condition for bit TRPF in register IS occurs. The service request output that will be activated is selected by bit field INPERR.
ENCHE	12	rw	**Enable Interrupt for Correct Hall Event** 0_B No interrupt will be generated if the set condition for bit CHE in register IS occurs. 1_B An interrupt will be generated if the set condition for bit CHE in register IS occurs. The service request output that will be activated is selected by bit field INPCHE.

Field	Bits	Type	Description
ENWHE	13	rw	**Enable Interrupt for Wrong Hall Event** 0_B No interrupt will be generated if the set condition for bit WHE in register IS occurs. 1_B An interrupt will be generated if the set condition for bit WHE in register IS occurs. The service request output that will be activated is selected by bit field INPERR.
ENIDLE	14	rw	**Enable Idle** This bit enables the automatic entering of the idle state (bit IDLE will be set) after a wrong hall event has been detected (bit WHE is set). During the idle state, the bit field MCMP is automatically cleared. 0_B The bit IDLE is not automatically set when a wrong hall event is detected. 1_B The bit IDLE is automatically set when a wrong hall event is detected.
ENSTR	15	rw	**Enable Multi-Channel Mode Shadow Transfer Interrupt** 0_B No interrupt will be generated if the set condition for bit STR in register IS occurs. 1_B An interrupt will be generated if the set condition for bit STR in register IS occurs. The service request output that will be activated is selected by bit field INPCHE.
0	11	r	**reserved;** returns 0 if read; should be written with 0;

18.8.2.5 Interrupt Node Pointer Register

Register INP contains the interrupt node pointers allowing a flexible interrupt handling. These bit fields define which service request output will be activated if the corresponding interrupt event occurs and the interrupt generation for this event is enabled.

INP
Interrupt Node Pointer Register **XSFR(56_H)** **Reset Value: 3940_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		INP T13		INP T12		INP ERR		INP CHE		INP CC62		INP CC61		INP CC60	
r		rw		rw		rw		rw		rw		rw		rw	

Field	Bits	Type	Description
INPCC60, INPCC61, INPCC62	[1:0], [3:2], [5:4]	rw	**Interrupt Node Pointer for Channel CC6x Interrupts** This bit field defines the service request output activated due to a set condition for bit CC6xR (if enabled by bit ENCC6xR) or for bit CC6xF (if enabled by bit ENCC6xF). 00_B Service request output SR0 is selected. 01_B Service request output SR1 is selected. 10_B Service request output SR2 is selected. 11_B Service request output SR3 is selected.
INPCHE	[7:6]	rw	**Interrupt Node Pointer for the CHE Interrupt** This bit field defines the service request output activated due to a set condition for bit CHE (if enabled by bit ENCHE) of for bit STR (if enabled by bit ENSTR). Coding see INPCC6x.
INPERR	[9:8]	rw	**Interrupt Node Pointer for Error Interrupts** This bit field defines the service request output activated due to a set condition for bit TRPF (if enabled by bit ENTRPF) or for bit WHE (if enabled by bit ENWHE). Coding see INPCC6x.

Field	Bits	Type	Description
INPT12	[11:10]	rw	**Interrupt Node Pointer for Timer12 Interrupts** This bit field defines the service request output activated due to a set condition for bit T12OM (if enabled by bit ENT12OM) or for bit T12PM (if enabled by bit ENT12PM). Coding see INPCC6x.
INPT13	[13:12]	rw	**Interrupt Node Pointer for Timer13 Interrupt** This bit field defines the service request output activated due to a set condition for bit T13CM (if enabled by bit ENT13CM) or for bit T13PM (if enabled by bit ENT13PM). Coding see INPCC6x.
0	[15:14]	r	**reserved;** returns 0 if read; should be written with 0;

18.9 General Module Operation

This section provides information about the:

- Configuration of the behavior of the different device operating modes (see mode control description in **Section 18.9.1**)
- Input selection (see **Section 18.9.2**)
- General register description (see **Section 18.9.3**)

18.9.1 Mode Control

The mode control concept for system control tasks, such as power saving, or suspend request for debugging, allows to program the module behavior under different device operating conditions. The behavior of a CCU6 kernel can be programmed for each of the device operating modes, that are requested by the global state control part of the SCU. Therefore, a CCU6 module provides a kernel state configuration register **KSCFG** defining the behavior in the following device operating modes:

- **Normal operation:**
  This operating mode is the default operating mode when neither a suspend request nor a clock-off request are pending. The module clock is not switched off and the CCU6 registers can be read or written. The kernel behavior is defined by KSCFG.NOMCFG.
- **Suspend mode:**
  This operating mode is requested when a suspend request (issued by a debugger) is pending in the device. The module clock is not switched off and the CCU6 registers can be read or written. The kernel behavior is defined by KSCFG.SUMCFG.
- **Clock-off mode:**
  This operating mode is requested for power saving purposes. The module clock is switched off automatically when all kernels of the CCU6 module reached their specified state in a stop mode. In this case, CCU6 registers can not be accessed. The kernel behavior is defined by KSCFG.COMCFG.

The kernel distinguishes four different blocks (T12, T13, Hall logic, and trap logic). These blocks can be individually enabled for the request of stop mode 0 and stop mode 1 by the sensitivity bits **KSCSR**.SBx. If the request sensitivity is disabled, the block continues normal operation. If the request sensitivity is enabled, the block operates as specified for the selected stop mode.
The complete CCU6 acknowledge is given to the GSC when all four blocks have reached their defined end condition.

Table 18-12 CCU6 Functional Blocks

Block	Function	Sensitivity Bit
0	**Timer T12:** A functional enable is delivered until the specified stop condition is reached. Then, T12 stops counting and the CC6xIN input stages are frozen.	KSCSR.SB0
1	**Timer T13:** A functional enable is delivered until the specified stop condition is reached. Then, T13 stops counting.	KSCSR.SB1
2	**Hall Logic:** The hall logic is stopped immediately and the CCPOSx input stages are frozen.	KSCSR.SB2
3	**Trap Logic:** The trap logic is stopped immediately and the CTRAP input stage is frozen.	KSCSR.SB3

The behavior of the CCU6 kernel can be programmed for each of the device operating modes (normal operation, suspend mode, clock-off mode). Therefore, it supports four kernel modes, as shown in **Table 18-13**.

Table 18-13 CCU6 Kernel Behavior

Kernel Mode	Kernel Behavior	Code
run mode 0	kernel operation as specified, no impact on CCU6 operation (same behavior for run mode 0 and run mode 1)	00_B
run mode 1		01_B

Table 18-13 CCU6 **Kernel Behavior** (cont'd)

Kernel Mode	Kernel Behavior	Code
stop mode 0	The sensitivity bits are taken into account for: **T12 block**: Timer T12 continues normal operation (if running) until they reach the end of the PWM period and then it stops (same stop condition as in single shot mode). When the timer stops, the CC6xIN inputs are frozen. **T13 block**: Timer T13 continues normal operation (if running) until they reach the end of the PWM period and then it stops (same stop condition as in single shot mode). **Hall logic block**: The CCPOSx input values are frozen. **Trap logic block**: The CTRAP input value is frozen.	10_B
stop mode 1	The output lines enabled for the trap condition are set to their passive values (similar to a trap state). The sensitivity bits are taken into account for: **T12 block**: Timer T12 stops immediately and CC6xIN inputs are frozen. **T13 block**: Timer T13 stops. **Hall logic block**: The CCPOSx input values are frozen. **Trap logic block**: The CTRAP input value is frozen.	11_B

Generally, bit field KSCFG.NOMCFG should be configured for run mode 0 as default setting for standard operation. If a CCU6 kernel should not react to a suspend request (and to continue operation as in normal mode), bit field KSCFG.SUMCFG has to be configured with the same value as KSCFG.NOMCFG. If a CCU6 kernel should show a different behavior and stop operation when a specific stop condition is reached, the code for stop mode 0 or stop mode 1 has to be written to KSCFG.SUMCFG.
A similar mechanism applies for the clock-off mode with the possibility to program the desired behavior by bit field KSCFG.COMCFG.

Note: The stop mode selection strongly depends on the application needs and it is very unlikely that different stop modes are required in parallel in the same application. As a result, only one stop mode type (either 0 or 1) should be used in the bit fields in register KSCFG. Do not mix stop mode 0 and stop mode 1 and avoid transitions from stop mode 0 to stop mode 1 (or vice versa) for the CCU6 module.

If the module clock is disabled by KSCFG.MODEN = 0 or in clock-off mode when the stop condition is reached (in stop mode 0 or 1), the module can not be accessed by read or write operations (except register KSCFG that can always be accessed). As a consequence, it can not be configured.
Please note that bit KSCFG.MODEN should only be set by SW while all configuration fields are configured for run mode 0.

18.9.2 Input Selection

Each CCU6 input signal can be selected from a vector of four possible inputs by programming the port input select registers **PISELL** and **PISELH**. This permits to adapt the pin functionality of the device to the application requirements.
The output pins for the module output signals are chosen in the ports.

Naming convention:
The input vector CC60IN[D:A] for input signal CC60IN is composed of the signals CC60INA to CC60IND.

Note: All functional inputs of the CCU6 *are synchronized to f_{CC6} before they affect the module internal logic. The resulting delay of $2/f_{CC6}$ and for asynchronous signals an additional uncertainty of $1/f_{CC6}$ have to be taken into account for precise timing calculation. An edge of an input signal can only be correctly detected if the high phase and the low phase of the input signal are both longer than $1/f_{CC6}$.*

18.9.3 General Registers

18.9.3.1 Port Input Select Registers

Registers PISELL and PISELH contain bit fields selecting the actual input signal for the module inputs.

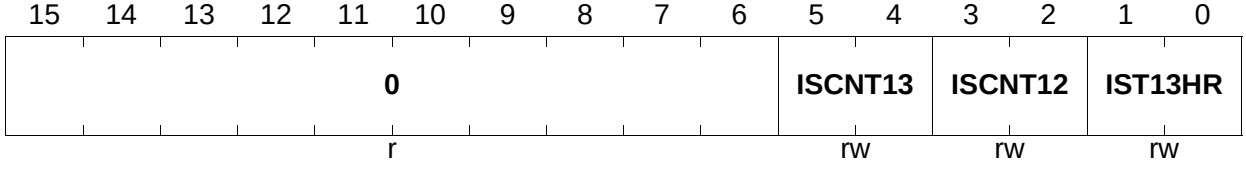
PISELL
Port Input Select Register Low **XSFR(04_H)** **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IST12HR		**ISPOS2**		**ISPOS1**		**ISPOS0**		**ISTRP**		**ISCC62**		**ISCC61**		**ISCC60**	
rw		rw		rw		rw		rw		rw		rw		rw	

Field	Bits	Type	Description
ISCC60	[1:0]	rw	**Input Select for CC60** This bit field defines the input signal used as CC60 capture input. 00_B The signal CC60INA is selected. 01_B The signal CC60INB is selected. 10_B The signal CC60INC is selected. 11_B The signal CC60IND is selected.
ISCC61	[3:2]	rw	**Input Select for CC61** This bit field defines the input signal used as CC61 capture input. 00_B The signal CC61INA is selected. 01_B The signal CC61INB is selected. 10_B The signal CC61INC is selected. 11_B The signal CC61IND is selected.
ISCC62	[5:4]	rw	**Input Select for CC62** This bit field defines the input signal used as CC62 capture input. 00_B The signal CC62INA is selected. 01_B The signal CC62INB is selected. 10_B The signal CC62INC is selected. 11_B The signal CC62IND is selected.

Field	Bits	Type	Description
ISTRP	[7:6]	rw	**Input Select for CTRAP** This bit field defines the input signal used as CTRAP input. 00_B The signal CTRAPA is selected. 01_B The signal CTRAPB is selected. 10_B The signal CTRAPC is selected. 11_B The signal CTRAPD is selected.
ISPOS0	[9:8]	rw	**Input Select for CCPOS0** This bit field defines the input signal used as CCPOS0 input. 00_B The signal CCPOS0A is selected. 01_B The signal CCPOS0B is selected. 10_B The signal CCPOS0C is selected. 11_B The signal CCPOS0D is selected.
ISPOS1	[11:10]	rw	**Input Select for CCPOS1** This bit field defines the input signal used as CCPOS1 input. 00_B The signal CCPOS1A is selected. 01_B The signal CCPOS1B is selected. 10_B The signal CCPOS1C is selected. 11_B The signal CCPOS1D is selected.
ISPOS2	[13:12]	rw	**Input Select for CCPOS2** This bit field defines the input signal used as CCPOS2 input. 00_B The signal CCPOS2A is selected. 01_B The signal CCPOS2B is selected. 10_B The signal CCPOS2C is selected. 11_B The signal CCPOS2D is selected.
IST12HR	[15:14]	rw	**Input Select for T12HR** This bit field defines the input signal used as T12HR input. 00_B The signal T12HRA is selected. 01_B The signal T12HRB is selected. 10_B The signal T12HRC is selected. 11_B The signal T12HRD is selected.

PISELH
Port Input Select Register High **XSFR(06_H)** **Reset Value: 0000_H**



Field	Bits	Type	Description
IST13HR	[1:0]	rw	**Input Select for T13HR** This bit field defines the input signal used as T13HR input. 00_B The signal T13HRA is selected. 01_B The signal T13HRB is selected. 10_B The signal T13HRC is selected. 11_B The signal T13HRD is selected.
ISCNT12	[3:2]	rw	**Input Select for T12 Counting Input** This bit field defines the input event leading to a counting action of T12. 00_B The T12 prescaler generates the counting events. Bit TCTR4.T12CNT is not taken into account. 01_B Bit TCTR4.T12CNT written with 1 is a counting event. The T12 prescaler is not taken into account. 10_B The timer T12 is counting each rising edge detected in the selected T12HR signal. 11_B The timer T12 is counting each falling edge detected in the selected T12HR signal.

Field	Bits	Type	Description
ISCNT13	[5:4]	rw	**Input Select for T13 Counting Input** This bit field defines the input event leading to a counting action of T13. 00_B The T13 prescaler generates the counting events. Bit TCTR4.T13CNT is not taken into account. 01_B Bit TCTR4.T13CNT written with 1 is a counting event. The T13 prescaler is not taken into account. 10_B The timer T13 is counting each rising edge detected in the selected T13HR signal. 11_B The timer T13 is counting each falling edge detected in the selected T13HR signal.
0	[15:6]	r	**reserved;** returns 0 if read; should be written with 0;

18.9.3.2 Kernel State Configuration Register

The kernel state configuration register KSCFG allows the selection of the desired kernel modes for the different device operating modes.
Bit fields KSCFG.NOMCFG and KSCFG.COMCFG are reset by a class 3 reset. Bit field KSCFG.SUMCFG is reset by a class 1 reset.

*Note: The coding of the bit fields NOMCFG, SUMCFG and COMCFG is described in **Table 18-13**.*

KSCFG
Kernel State Configuration Register

XSFR(00_H) **Reset Value: 0000_H**

15	14	13 12	11	10	9 8	7	6	5 4	3 2	1	0
BP COM	**0**	**COMCFG**	**BP SUM**	**0**	**SUMCFG**	**BP NOM**	**0**	**NOMCFG**	**0**	**BP MOD EN**	**MOD EN**
w	r	rw	w	r	rw	w	r	rw	r	w	rw

Field	Bits	Type	Description
MODEN	0	rw	**Module Enable** This bit enables the module kernel clock and the module functionality. 0_B The module is switched off immediately (without respecting a stop condition). It does not react on mode control actions and the module clock is switched off. The module does not react on read accesses and ignores write accesses (except to KSCFG). 1_B The module is switched on and can operate. After writing 1 to MODEN, it is recommended to read register KSCFG to avoid pipeline effects in the control block before accessing other ADC registers.
BPMODEN	1	w	**Bit Protection for MODEN** This bit enables the write access to the bit MODEN. It always reads 0. 0_B MODEN is not changed. 1_B MODEN is updated with the written value.

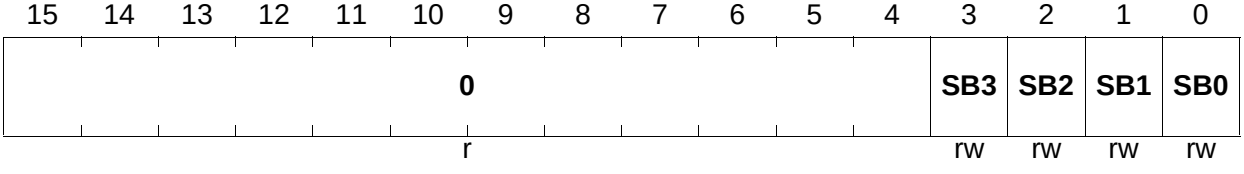
Field	Bits	Type	Description
NOMCFG	[5:4]	rw	**Normal Operation Mode Configuration** This bit field defines the kernel mode applied in normal operation mode. 00_B Run mode 0 is selected. 01_B Run mode 1 is selected. 10_B Stop mode 0 is selected. 11_B Stop mode 1 is selected.
BPNOM	7	w	**Bit Protection for NOMCFG** This bit enables the write access to the bit field NOMCFG. It always reads 0. 0_B NOMCFG is not changed. 1_B NOMCFG is updated with the written value.
SUMCFG	[9:8]	rw	**Suspend Mode Configuration** This bit field defines the kernel mode applied in suspend mode. Coding like NOMCFG.
BPSUM	11	w	**Bit Protection for SUMCFG** This bit enables the write access to the bit field SUMCFG. It always reads 0. 0_B SUMCFG is not changed. 1_B SUMCFG is updated with the written value.
COMCFG	[13:12]	rw	**Clock Off Mode Configuration** This bit field defines the kernel mode applied in clock-off mode. Coding like NOMCFG.
BPCOM	15	w	**Bit Protection for COMCFG** This bit enables the write access to the bit field COMCFG. It always reads 0. 0_B COMCFG is not changed. 1_B COMCFG is updated with the written value.
0	[3:2], 6, 10, 14	r	**Reserved** returns 0 if read; should be written with 0;

Note: The bit protection bits BPxxx allow partly modification of the configuration bits with a single write operation (without the need of a read-modify-write mechanism handled by the CPU).

18.9.3.3 Kernel State Sensitivity Control Register

The kernel state control sensitivity register bits define which internal block is effected by stop modes 0 and 1.

KSCSR
Kernel State Control Sensitivity Register
XSFR($0E_H$) **Reset Value: 0000_H**

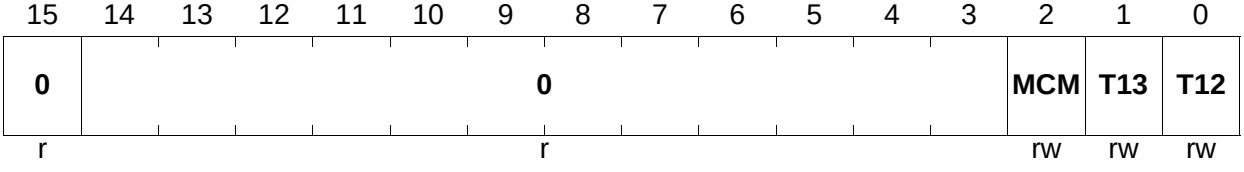


Field	Bits	Type	Description
SB0, **SB1,** **SB2,** **SB3**	0, 1, 2, 3	rw	**Sensitivity Block x** This bit defines if block x of the CCU6 kernel is sensitive to stop mode 0 or stop mode 1. The functional definition of the blocks is given in **Table 18-12**. 0_B Block x is not sensitive to stop mode 0 or stop mode 1 and behaves like in run mode 0. It continues normal operation without respecting the defined stop condition. 1_B Block x is sensitive to stop mode 0 or stop mode 1. It is respecting the defined stop condition.
0	[15:4]	r	**reserved;** returns 0 if read; should be written with 0;

18.9.3.4 Module Configuration Register

The module configuration register contains bits describing the functionality that is available in the CCU6 module.

MCFG
Module Configuration Register **XSFR($0C_H$)** **Reset Value: 0007_H**



Field	Bits	Type	Description
T12	0	rw	**T12 Available** This bit indicates if the T12 block is available. 0_B The T12 block is not available. A write access to T12PR is ignored. 1_B The T12 block is available. A write access to T12PR is executed.
T13	1	rw	**T13 Available** This bit indicates if the T13 block is available. 0_B The T13 block is not available. A write access to T13PR is ignored. 1_B The T13 block is available. A write access to T13PR is executed.
MCM	2	rw	**Multi-Channel Mode Available** This bit indicates if the multi-channel mode functionality is available. 0_B The multi-channel mode functionality is not available. A write access to MCMOUTS is ignored. 1_B The multi-channel mode functionality is available. A write access to MCMOUTS is executed.
0	[15:3]	r	**Reserved;** read as 0; should be written with 0.

18.10 Implementation

This section describes the implementation of the CCU6 modules in the XC2000 device.

- Address map (see **Section 18.10.1**)
- Interrupt control registers (see **Section 18.10.2**)
- Synchronous start (see **Section 18.10.3**)
- Connections of CCU60 (see **Section 18.10.4.1**)
- Connections of CCU61 (see **Section 18.10.4.2**)
- Connections of CCU62 (see **Section 18.10.4.3**)
- Connections of CCU63 (see **Section 18.10.4.4**)

18.10.1 Address Map

The four CCU6x modules (named CCU60 to CCU63) can be accessed in the following address ranges.
The exact register address is given by the relative address of the register (given in **Table 18-1**) plus the kernel base address (given in **Table 18-14**) of the module.

Table 18-14 Registers Address Space

Module	Base Address	End Address	Note
CCU60	$EA00_H$	$EA7E_H$	
CCU61	$EA80_H$	$EAFE_H$	
CCU62	$EB00_H$	$EB7E_H$	
CCU63	$EB80_H$	$EBFE_H$	

18.10.2 Interrupt Control Registers

The interrupt control registers are located in the SFR area. They are described in the general interrupt chapter.

Table 18-15 CCU6 Interrupt Control Registers

Short Name	Description
CCU60_0IC	Interrupt Control Register for SR0 of CCU60
CCU60_1IC	Interrupt Control Register for SR1 of CCU60
CCU60_2IC	Interrupt Control Register for SR2 of CCU60
CCU60_3IC	Interrupt Control Register for SR3 of CCU60
CCU61_0IC	Interrupt Control Register for SR0 of CCU61
CCU61_1IC	Interrupt Control Register for SR1 of CCU61
CCU61_2IC	Interrupt Control Register for SR2 of CCU61
CCU61_3IC	Interrupt Control Register for SR3 of CCU61
CCU62_0IC	Interrupt Control Register for SR0 of CCU62
CCU62_1IC	Interrupt Control Register for SR1 of CCU62
CCU62_2IC	Interrupt Control Register for SR2 of CCU62
CCU62_3IC	Interrupt Control Register for SR3 of CCU62
CCU63_0IC	Interrupt Control Register for SR0 of CCU63
CCU63_1IC	Interrupt Control Register for SR1 of CCU63
CCU63_2IC	Interrupt Control Register for SR2 of CCU63
CCU63_3IC	Interrupt Control Register for SR3 of CCU63

18.10.3 Synchronous Start Feature

Synchronous start is supported by bit SYSCON.GLCCST (global capture/compare start) in the SCU module that is connected to the T12HR and T13HR inputs of all CCU6x modules.

The same signal can also be connected to other capture/compare units in order to allow a synchronous start of the capture/compare timers.

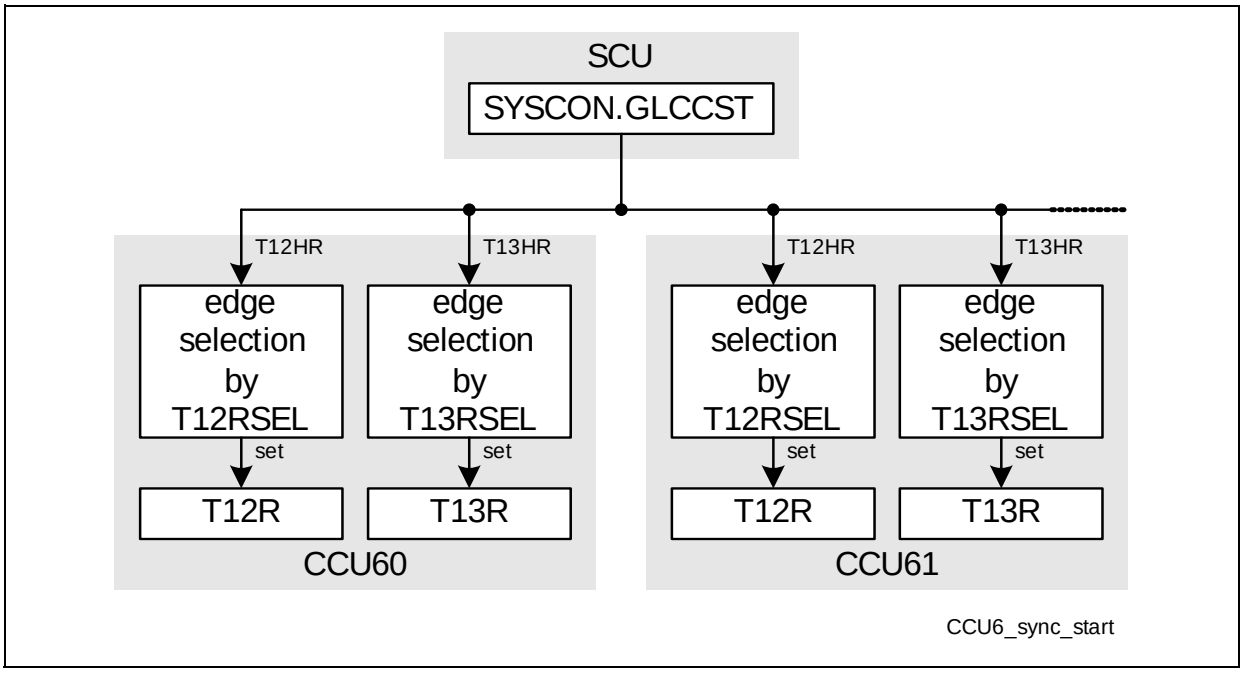


Figure 18-43 Synchronization Concept

18.10.4 Digital Connections

The following tables show the digital connections of the CCU6x modules with other modules or pins in the XC2000 device.
Each input signal can be selected among 4 possible input lines, e.g. the input vector for input signal CC60IN is composed of CC60IN[D:A].

The following sections refer to the interface signals, whereas the connections of the service request outputs SR[3:0] to the interrupt control registers of each CCU6x to the interrupt control registers is given in **Section 18.10.2**.

The CCU6x modules are clocked with the XC2000 system clock, so $f_{CC6} = f_{SYS}$.

Note: All functional inputs of the CCU6 *are synchronized to f_{CC6} before they can affect the module internal logic. The resulting delay of $2/f_{CC6}$ and an uncertainty of $1/f_{CC6}$ have to be taken into account for precise timing calculation.*
An edge of an input signal can only be correctly detected if both, the high phase and the low phase of the input signal are each longer than $1/f_{CC6}$.

18.10.4.1 Connections of CCU60

This table describes the module interconnections of CCU60.

Table 18-16 CCU60 Digital Connections in XC2000

Signal	from/to Module	I/O to CCU60	Can be used to/as
CC60INA	P10.0	I	input signals for capture event on channel CC60
CC60INB	P8.0	I	
CC60INC	0	I	
CC60IND	RTC interrupt	I	
CC61INA	P10.1	I	input signals for capture event on channel CC61
CC61INB	P8.1	I	
CC61INC	0	I	
CC61IND	0	I	
CC62INA	P10.2	I	input signals for capture event on channel CC62
CC62INB	P8.2	I	
CC62INC	0	I	
CC62IND	0	I	

Table 18-16 CCU60 Digital Connections in XC2000 (cont'd)

Signal	from/to Module	I/O to CCU60	Can be used to/as
CTRAPA	P10.6	I	input signals for CTRAP, the ESRx input refers to the synchronized input signal, that can be filtered (if enabled)
CTRAPB	P8.6	I	
CTRAPC	ESR2	I	
CTRAPD	1	I	
CCPOS0A	P10.7	I	input signals for CCPOS0
CCPOS0B	P9.7	I	
CCPOS0C	0	I	
CCPOS0D	0	I	
CCPOS1A	P10.8	I	input signals for CCPOS1
CCPOS1B	P9.6	I	
CCPOS1C	0	I	
CCPOS1D	0	I	
CCPOS2A	P10.9	I	input signals for CCPOS2
CCPOS2B	P9.5	I	
CCPOS2C	0	I	
CCPOS2D	0	I	
T12HRA	CCU63_MCM_ST	I	input signals for T12HR
T12HRB	P5.5	I	
T12HRC	P5.8	I	
T12HRD	SYSCON.GLCCST	I	
T13HRA	EXTCLK (SCU)	I	input signals for T13HR
T13HRB	CCU60_T12_ZM	I	
T13HRC	P5.8	I	
T13HRD	SYSCON.GLCCST	I	
CC60	P10.0 P8.0	O	compare outputs of channel CC60
COUT60	P10.3 P8.3	O	

Table 18-16 CCU60 Digital Connections in XC2000 (cont'd)

Signal	from/to Module	I/O to CCU60	Can be used to/as
CC61	P10.1 P8.1	O	compare outputs of channel CC61
COUT61	P10.4 P8.4	O	
CC62	P10.2 P8.2	O	compare outputs of channel CC62
COUT62	P10.5 P8.5	O	
COUT63	P10.7 P10.10 P8.6	O	compare output of channel CC63
T12_ZM	CCU60_T13HRB	O	T12 zero match
T13_PM	ADC0_REQTR0B ADC0_REQTR1B U0C0_DX2F U0C1_DX2F ERU_OGU02	O	T13 period match
MCM_ST	CCU61_T12HRA ERU_OGU01	O	MCM shadow transfer
SR[3:0]	interrupt controller	O	interrupt output lines (service request)

18.10.4.2 Connections of CCU61

This table describes the module interconnections of CCU61.

Table 18-17 CCU61 Digital Connections in XC2000

Signal	from/to Module	I/O to CC61	Can be used to/as
CC60INA	P0.0	I	input signals for capture event on channel CC60
CC60INB	0	I	
CC60INC	0	I	
CC60IND	0	I	
CC61INA	P0.1	I	input signals for capture event on channel CC61
CC61INB	0	I	
CC61INC	0	I	
CC61IND	0	I	
CC62INA	P0.2	I	input signals for capture event on channel CC62
CC62INB	0	I	
CC62INC	0	I	
CC62IND	0	I	
CTRAPA	P0.6	I	input signals for CTRAP, the ESRx input refers to the synchronized input signal, that can be filtered (if enabled)
CTRAPB	P0.7	I	
CTRAPC	ESR2	I	
CTRAPD	1	I	
CCPOS0A	0	I	input signals for CCPOS0
CCPOS0B	0	I	
CCPOS0C	0	I	
CCPOS0D	0	I	
CCPOS1A	0	I	input signals for CCPOS1
CCPOS1B	0	I	
CCPOS1C	0	I	
CCPOS1D	0	I	

Table 18-17 CCU61 Digital Connections in XC2000 (cont'd)

Signal	from/to Module	I/O to CC61	Can be used to/as
CCPOS2A	0	I	input signals for CCPOS2
CCPOS2B	0	I	
CCPOS2C	0	I	
CCPOS2D	0	I	
T12HRA	CCU60_MCM_ST	I	input signals for T12HR
T12HRB	P1.2	I	
T12HRC	P5.8	I	
T12HRD	SYSCON.GLCCST	I	
T13HRA	0	I	input signals for T13HR
T13HRB	CCU61_T12_ZM	I	
T13HRC	P5.8	I	
T13HRD	SYSCON.GLCCST	I	
CC60	P0.0	O	compare outputs of channel CC60
COUT60	P0.3	O	
CC61	P0.1	O	compare outputs of channel CC61
COUT61	P0.4	O	
CC62	P0.2	O	compare outputs of channel CC62
COUT62	P0.5	O	
COUT63	P0.6	O	compare output of channel CC63
T12_ZM	CCU61_T13HRB	O	T12 zero match
T13_PM	ADC1_REQTR0B ADC1_REQTR1B U1C0_DX2F U1C1_DX2F ERU_OGU12	O	T13 period match
MCM_ST	CCU62_T12HRA ERU_OGU11	O	MCM shadow transfer

18.10.4.3 Connections of CCU62

This table describes the module interconnections of CCU62.

Table 18-18 CCU62 Digital Connections in XC2000

Signal	from/to Module	I/O to CCU62	Can be used to/as
CC60INA	P1.7	I	input signals for capture event on channel CC60
CC60INB	0	I	
CC60INC	0	I	
CC60IND	0	I	
CC61INA	P1.6	I	input signals for capture event on channel CC61
CC61INB	0	I	
CC61INC	0	I	
CC61IND	0	I	
CC62INA	P1.2	I	input signals for capture event on channel CC62
CC62INB	0	I	
CC62INC	0	I	
CC62IND	0	I	
CTRAPA	P7.1	I	input signals for CTRAP, the ESRx input refers to the synchronized input signal, which can be filtered (if enabled)
CTRAPB	P1.0	I	
CTRAPC	ESR2	I	
CTRAPD	1	I	
CCPOS0A	P7.2	I	input signals for CCPOS0
CCPOS0B	0	I	
CCPOS0C	0	I	
CCPOS0D	0	I	
CCPOS1A	P7.3	I	input signals for CCPOS1
CCPOS1B	0	I	
CCPOS1C	0	I	
CCPOS1D	0	I	

Table 18-18 CCU62 Digital Connections in XC2000 (cont'd)

Signal	from/to Module	I/O to CCU62	Can be used to/as
CCPOS2A	P7.4	I	input signals for CCPOS2
CCPOS2B	0	I	
CCPOS2C	0	I	
CCPOS2D	0	I	
T12HRA	CCU61_MCM_ST	I	input signals for T12HR
T12HRB	P1.3	I	
T12HRC	P5.8	I	
T12HRD	SYSCON.GLCCST	I	
T13HRA	CAN_INT_O15	I	input signals for T13HR
T13HRB	CCU62_T12_ZM	I	
T13HRC	P5.8	I	
T13HRD	SYSCON.GLCCST	I	
CC60	P1.7	O	compare outputs of channel CC60
COUT60	P1.5	O	
CC61	P1.6	O	compare outputs of channel CC61
COUT61	P1.4	O	
CC62	P1.2	O	compare outputs of channel CC62
COUT62	P1.1	O	
COUT63	P1.3	O	compare output of channel CC63
T12_ZM	CCU62_T13HRB	O	T12 zero match
T13_PM	ADC0_REQTR2B U2C0_DX2F U2C1_DX2F ERU_OGU22	O	T13 period match
MCM_ST	CCU63_T12HRA ERU_OGU21	O	MCM shadow transfer

18.10.4.4 Connections of CCU63

This table describes the module interconnections of CCU63.

Table 18-19 CCU63 Digital Connections in XC2000

Signal	from/to Module	I/O to CCU63	Can be used to/as
CC60INA	P9.0	I	input signals for capture event on channel CC60
CC60INB	P2.0	I	
CC60INC	0	I	
CC60IND	0	I	
CC61INA	P9.1	I	input signals for capture event on channel CC61
CC61INB	P2.1	I	
CC61INC	0	I	
CC61IND	0	I	
CC62INA	P9.2	I	input signals for capture event on channel CC62
CC62INB	P2.2	I	
CC62INC	0	I	
CC62IND	0	I	
CTRAPA	P9.6	I	input signals for CTRAP, the ESRx input refers to the synchronized input signal, which can be filtered (if enabled)
CTRAPB	P9.7	I	
CTRAPC	ESR2	I	
CTRAPD	1	I	
CCPOS0A	P11.0	I	input signals for CCPOS0
CCPOS0B	0	I	
CCPOS0C	0	I	
CCPOS0D	0	I	
CCPOS1A	P11.1	I	input signals for CCPOS1
CCPOS1B	0	I	
CCPOS1C	0	I	
CCPOS1D	0	I	

Table 18-19 CCU63 Digital Connections in XC2000 (cont'd)

Signal	from/to Module	I/O to CCU63	Can be used to/as
CCPOS2A	P11.2	I	input signals for CCPOS2
CCPOS2B	0	I	
CCPOS2C	0	I	
CCPOS2D	0	I	
T12HRA	CCU62_MCM_ST	I	input signals for T12HR
T12HRB	P5.4	I	
T12HRC	P5.8	I	
T12HRD	SYSCON.GLCCST	I	
T13HRA	CAN_INT_O15	I	input signals for T13HR
T13HRB	CCU63_T12_ZM	I	
T13HRC	P5.8	I	
T13HRD	SYSCON.GLCCST	I	
CC60	P9.0 P2.0	O	compare outputs of channel CC60
COUT60	P9.3	O	
CC61	P9.1 P2.1	O	compare outputs of channel CC61
COUT61	P9.4	O	
CC62	P9.2 P2.2	O	compare outputs of channel CC62
COUT62	P9.5 P9.6	O	
COUT63	P9.6 P2.3	O	compare output of channel CC63
T12_ZM	CCU63_T13HRB	O	T12 zero match
T13_PM	ADC1_REQTR2B ERU_OGU32	O	T13 period match
MCM_ST	CCU60_T12HRA ERU_OGU31	O	MCM shadow transfer

19 Universal Serial Interface Channel

The **U**niversal **S**erial **I**nterface **C**hannel module (USIC) is a flexible interface module covering several serial communication protocols. A USIC module contains two independent communication channels named UxC0 and UxC1, with x being the number of the USIC module (e.g. channel y of USIC module x is referenced as UxCy). The user can program during run-time which protocol will be handled by each communication channel and which pins are used.

This chapter is structured as follows:

- Introduction (see **Section 19.1**)
- Operating the USIC (see **Section 19.2**)
- ASC protocol for UART and LIN (see **Section 19.3**)
- SSC protocol (see **Section 19.4**)
- IIC protocol (see **Section 19.5**)
- IIS protocol (see **Section 19.6**)
- Module implementation in XC2000 (see **Section 19.7**)

19.1 Introduction

This section gives an overview about the feature set of the USIC and introduces the USIC structure. It describes the:

- Feature set overview (see **Section 19.1.1**)
- Channel structure (see **Section 19.1.2**)
- Input stages (see **Section 19.1.3**)
- Output signals (see **Section 19.1.4**)
- Baud rate generator (see **Section 19.1.5**)
- Channel events and interrupts (see **Section 19.1.6**)
- Data shifting and handling (see **Section 19.1.7**)

19.1.1 Feature Set Overview

Each USIC channel can be individually configured to match the application needs, e.g. the protocol can be selected or changed during run time without the need for a reset. The following protocols are supported:

- **UART** (ASC, asynchronous serial channel)
  - module capability: receiver/transmitter with max. baud rate $f_{sys}/4$
  - application target baud rate range: 1.2 kBaud to 3.5 MBaud
  - number of data bits per data frame 1 to 63
  - MSB or LSB first
- **LIN** Support by HW (low-cost network, baud rate up to 20 kBaud)
  - data transfers based on ASC protocol
  - baud rate detection possible by built-in capture event of baud rate generator
  - checksum generation under SW control for higher flexibility
- **SSC/SPI** (synchronous serial channel with or without slave select lines)
  - module capability: slave mode with max. baud rate f_{sys} /2
  - module capability: master mode with max. baud rate f_{sys} /2
  - application target baud rate range: 2 kBaud to 10 MBaud
  - number of data bits per data frame 1 to 63, more with explicit stop condition
  - MSB or LSB first
- **IIC** (Inter-IC Bus)
  - application baud rate 100 kBaud to 400 kBaud
  - 7-bit and 10-bit addressing supported
  - full master and slave device capability
- **IIS** (infotainment audio bus)
  - module capability: receiver with max. baud rate f_{SYS}
  - module capability: transmitter with max. baud rate f_{SYS} /2
  - application target baud rate range: up to 26 MBaud

In addition to the flexible choice of the communication protocol, the USIC structure has been designed to reduce the system load (CPU load) allowing efficient data handling. The following aspects have been considered:

- **Data buffer capability**
  The standard buffer capability includes a double word buffer for receive data and a single word buffer for transmit data. This allows longer CPU reaction times (e.g. interrupt latency).
- **Additional FIFO buffer capability**
  In addition to the standard buffer capability, the received data and the data to be transmitted can be buffered in a FIFO buffer structure. The size of the receive and the transmit FIFO buffer can be programmed independently. Depending on the application needs, a total buffer capability of 64 data words can be assigned to the receive and transmit FIFO buffers of a USIC module (the two channels of the USIC module share the 64 data word buffer).

In addition to the FIFO buffer, a bypass mechanism allows the introduction of high-priority data without flushing the FIFO buffer.

- **Transmit control information**
  For each data word to be transmitted, a 5-bit transmit control information has been added to automatically control some transmission parameters, such as word length, frame length, or the slave select control for the SPI protocol. The transmit control information is generated automatically by analyzing the address where the user SW has written the data word to be transmitted (32 input locations = 2^5 = 5 bit transmit control information).
  This feature allows individual handling of each data word, e.g. the transmit control information associated to the data words stored in a transmit FIFO can automatically modify the slave select outputs to select different communication targets (slave devices) without CPU load. Alternatively, it can be used to control the frame length.
- **Flexible frame length control**
  The number of bits to be transferred within a data frame is independent of the data word length and can be handled in two different ways. The first option allows automatic generation of frames up to 63 bits with a known length. The second option supports longer frames (even unlimited length) or frames with a dynamically controlled length.
- **Interrupt capability**
  The events of each USIC channel can be individually routed to one of 4 service request outputs SR[3:0], depending on the application needs. Furthermore, specific start and end of frame indications are supported in addition to protocol-specific events.
- **Flexible interface routing**
  Each USIC channel offers the choice between several possible input and output pins connections for the communications signals. This allows a flexible assignment of USIC signals to pins that can be changed without resetting the device.
- **Input conditioning**
  Each input signal is handled by a programmable input conditioning stage with programmable filtering and synchronization capability.
- **Baud rate generation**
  Each USIC channel contains an own baud rate generator. The baud rate generation can be based either on the internal module clock or on an external frequency input. This structure allows data transfers with a frequency that can not be generated internally, e.g. to synchronize several communication partners.
- **Transfer trigger capability**
  In master mode, data transfers can be triggered events generated outside the USIC module, e.g. at an input pin or a timer unit (transmit data validation). This feature allows time base related data transmission.
- **Debugger support**
  The USIC offers specific addresses to read out received data without interaction with

the FIFO buffer mechanism. This feature allows debugger accesses without the risk of a corrupted receive data sequence.

To reach a desired baud rate, two criteria have to be respected, the module capability and the application environment. The module capability is defined with respect to the module's input clock frequency f_{SYS}, being the base for the module operation. Although the module's capability being much higher (depending on the module clock and the number of module clock cycles needed to represent a data bit), the reachable baud rate is generally limited by the application environment. In most cases, the application environment limits the maximum reachable baud rate due to driver delays, signal propagation times, or due to EMI reasons.

Note: Depending on the selected additional functions (such as digital filters, input synchronization stages, sample point adjustment, data structure, etc.), the maximum reachable baud rate can be limited. Please also take care about additional delays, such as (internal or external) propagation delays and driver delays (e.g. for collision detection in ASC mode, for IIC, etc.).

19.1.2 Channel Structure

The USIC module contains two independent communication channels, with structure shown in **Figure 19-1**.
The data shift unit and the data buffering of each channel support full-duplex data transfers. The protocol-specific actions are handled by protocol pre-processors (PPP). In order to simplify data handling, an additional FIFO data buffer is optionally available for each USIC module to store transmit and receive data for each channel. This FIFO data buffer is not necessarily available in all devices (please refer to USIC implementation chapter for details).
Due to the independent channel control and baud rate generation, the communication protocol, baud rate and the data format can be independently programmed for each communication channel.

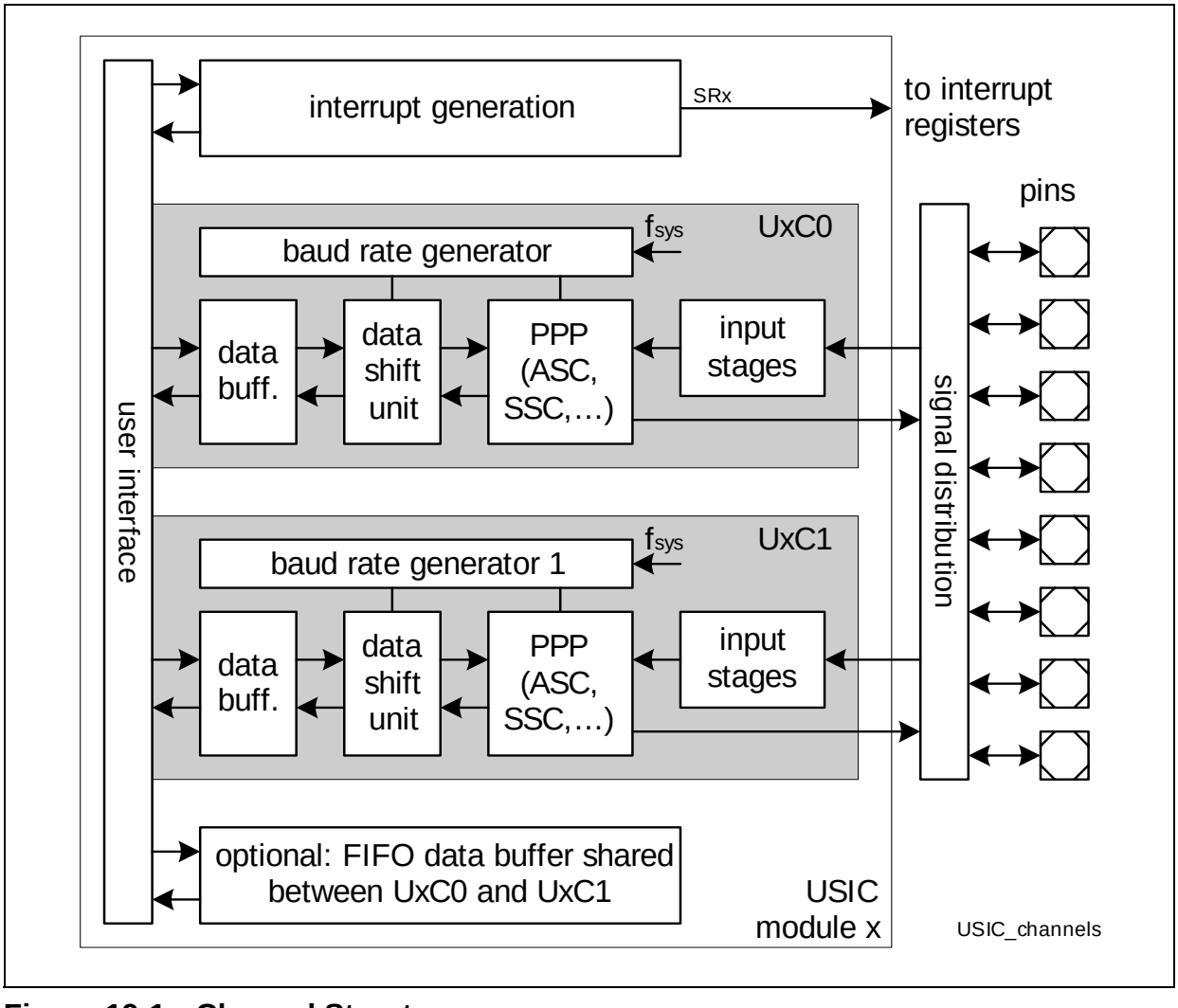


Figure 19-1 Channel Structure

19.1.3 Input Stages

For each protocol up to three input signals are available, the number of actually used inputs depends on the selected protocol. Each input signal is handled by an input stage (named DX0, DX1, DX2) for signal conditioning, such as input selection, polarity control, or a digital input filter. They can be classified according to their meaning for the protocols, see **Table 19-1**.
The inputs marked as “optional” are not needed for the standard function of a protocol and may be used for enhancements. The descriptions of protocol-specific items are given in the related protocol chapters, for the external frequency input please refer to the baud rate generator, and for the transmit data validation to the data handling section.

Table 19-1 Input Signals for Different Protocols

Selected Protocol	Shift Data Input (handled by DX0)	Shift Clock Input (handled by DX1)	Shift Control Input (handled by DX2)
ASC, LIN	RxD	optional: external frequency input or TxD collision detection	optional: transmit data validation
SSC, SPI (master)	DIN (MRST, MISO)	optional: external frequency input or delay compensation	optional: transmit data validation or delay compensation
SSC, SPI (slave)	DIN (MTSR, MOSI)	SCLKIN	SELIN
IIC	SDA	SCL	optional: transmit data validation
IIS (master)	DIN	optional: external frequency input or delay compensation	optional: transmit data validation or delay compensation
IIS (slave)	DIN	SCLKIN	WAIN

Note: To allow a certain flexibility in assigning required USIC input functions to port pins of the device, each input stage can select the desired input location among several posibilities.
*The available USIC signals and their port locations are listed in the implementation chapter, see **Section 19.7.5**.*

19.1.4 Output Signals

For each protocol up to eleven protocol-related output signals are available, the number of actually used outputs depends on the selected protocol. They can be classified according to their meaning for the protocols, see **Table 19-2**.
The outputs marked as “optional” are not needed for the standard function of a protocol and may be used for enhancements. The descriptions of protocol-specific items are given in the related protocol chapters. The MCLKOUT output signal has a stable frequency relation to the shift clock output (the frequency of MCLKOUT can be higher than for SCLKOUT) for synchronization purposes of a slave device to a master device. If the baud rate generator is not needed for a specific protocol (e.g. in SSC slave mode), the SCLKOUT and MCLKOUT signals can be used as clock outputs with 50% duty cycle with a frequency that can be independent from the communication baud rate.

Table 19-2 Output Signals for Different Protocols

Selected Protocol	Shift Data Output DOUT	Shift Clock Output SCLKOUT	Shift Control Outputs SELO[7:0]	Master Clock Output MCLKOUT
ASC, LIN	TxD	not used	not used	optional: master time base
SSC, SPI (master)	DOUT (MTSR, MOSI)	master shift clock	slave select, chip select	optional: master time base
SSC, SPI (slave)	DOUT (MRST, MISO)	optional: independent clock output	not used	optional: independent clock output
IIC	SDA	SCL	not used	optional: master time base
IIS (master)	DOUT	master shift clock	WA	optional: master time base
IIS (slave)	DOUT	optional: independent clock output	not used	optional: independent clock output

Note: To allow a certain flexibility in assigning required USIC output functions to port pins of the device, most output signals are made available on several port pins. The port control itself defines pin-by-pin which signal is used as output signal for a port pin (see port chapter).
*The available USIC signals and their port locations are listed in the implementation chapter, see **Section 19.7.5**.*

19.1.5 Baud Rate Generator

Each USIC Channel contains a baud rate generator structured as shown in **Figure 19-2**. It is based on coupled divider stages, providing the frequencies needed for the different protocols. It contains:

- A fractional divider to generate the input frequency $f_{PIN} = f_{FD}$ for baud rate generation based on the internal system frequency f_{SYS}.
- The DX1 input to generate the input frequency $f_{PIN} = f_{DX1}$ for baud rate generation based on an external signal.
- A protocol-related counter to provide the master clock signal MCLK, the shift clock signal SCLK, and other protocol-related signals. It can also be used for time interval measurement, e.g. baud rate detection.
- A time quanta counter associated to the protocol pre-processor defining protocol-specific timings, such shift control signals or bit timings, based on the input frequency f_{CTQIN}.
- The output signals MCLKOUT and SCLKOUT of the protocol-related divider that can be made available on pins. In order to adapt to different applications, some output characteristics of these signals can be configured.
  For device-specific details about availability of USIC signals on pins please refer to the implementation chapter.

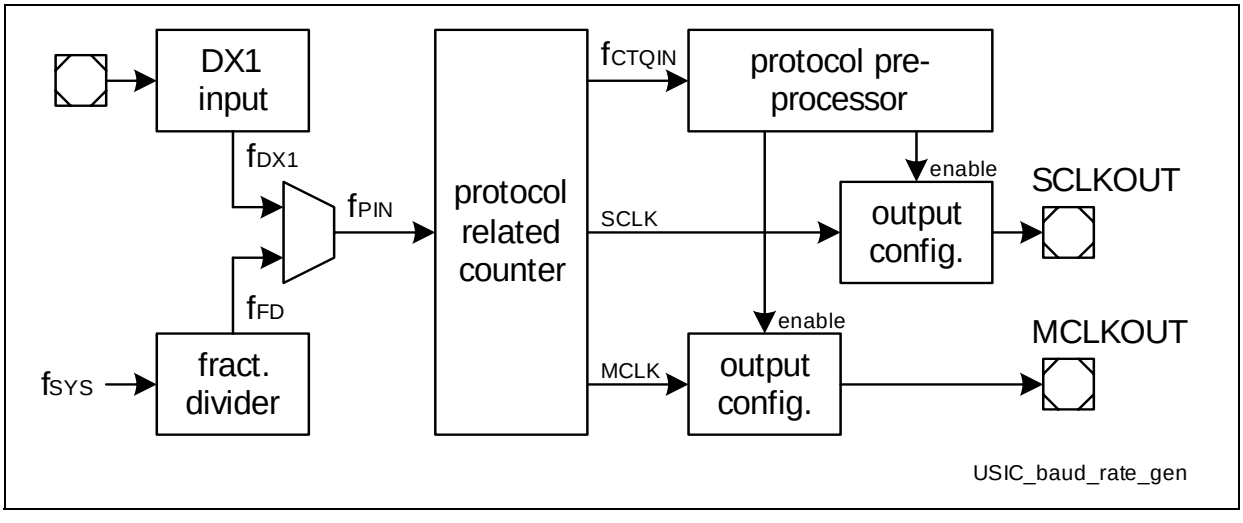


Figure 19-2 Baud Rate Generator

19.1.6 Channel Events and Interrupts

The notification of the user about events occurring during data traffic and data handling is based on:

- Data transfer events related to the transmission or reception of a data word, independent of the selected protocol.
- Protocol-specific events depending on the selected protocol.
- Data buffer events related to data handling by the optional FIFO data buffers.

19.1.7 Data Shifting and Handling

The data handling of the USIC module is based on an independent data shift unit (DSU) and a buffer structure that is similar for the supported protocols. The data shift and buffer registers are 16-bit wide (maximum data word length), but several data words can be concatenated to achieve longer data frames. The DSU inputs are the shift data (handled by input stage DX0), the shift clock (handled by the input stage DX1), and the shift control (handled by the input stage DX2). The signal DOUT represents the shift data output.

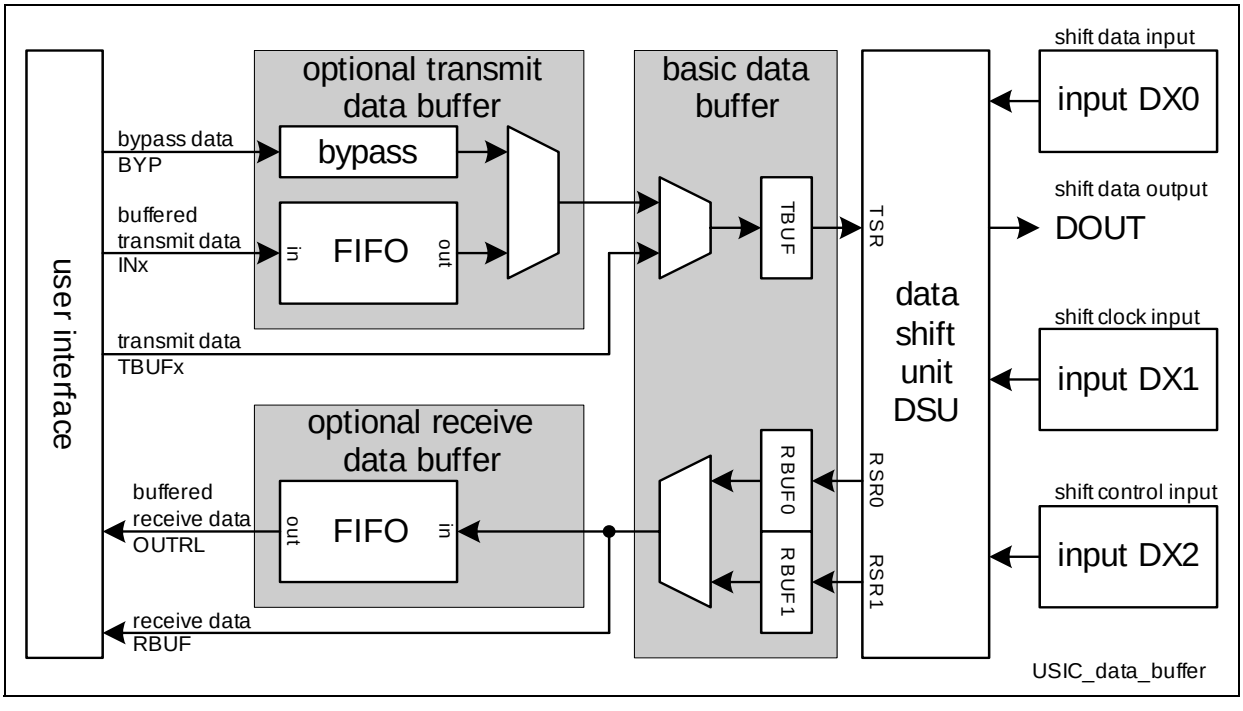


Figure 19-3 Principle of Data Buffering

The principle of data handling comprises:

- A transmitter with a transmit shift register (TSR) in the DSU and a transmit data buffer (TBUF). A data validation scheme allows triggering and gating of data transfers by external events under certain conditions.
- A receiver with two alternating receive shift registers (RSR0 and RSR1) in the DSU and a double receive buffer structure (RBUF0, RBUF1). The alternating receive shift

registers support the reception of data streams and data frames longer than one data word.
- Optional transmit and receive data buffers according to the first-in-first-out principle (FIFO), that are not necessarily available in all devices. For device-specific details about availability of the FIFO buffer please refer to the USIC implementation chapter.
- A user interface to handle data, interrupts, and status and control information.

19.1.7.1 Basic Data Buffer Structure

The read access to received data and the write access of data to be transmitted can be handled by a basic data buffer structure.
The received data stored in the receiver buffers RBUF0/RBUF1 can be read directly from these registers. In this case, the user has to take care about the reception sequence to read these registers in the correct order. To simplify the use of the receive buffer structure, register RBUF has been introduced. A read action from this register delivers the data word received first (oldest data) to respect the reception sequence. With a read access from at least the low byte of RBUF, the data is automatically declared to be no longer new and the next received data word becomes visible in RBUF and can be read out next.

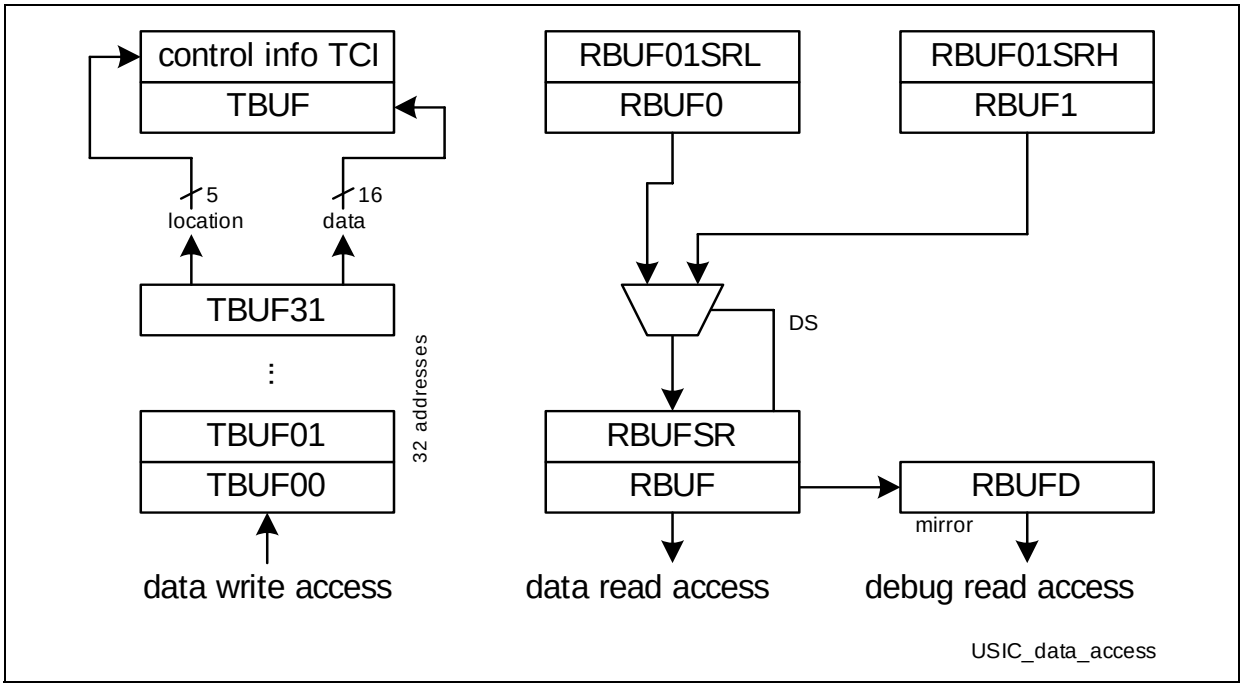


Figure 19-4 Data Access Structure without additional Data Buffer

It is recommended to read the received data words by accesses to RBUF and to avoid handling of RBUF0 and RBUF1. The USIC module also supports the use of debug accesses to receive data words. Debugger read accesses should not disturb the receive data sequence and, as a consequence, should not target RBUF. Therefore, register

RBUFD has been introduced. It contains the same value as RBUF, but a read access from RBUFD does not change the status of the data (same data can be read several times). In addition to the received data, some additional status information about each received data word is available in the receiver buffer status registers RBUF01SRL/H (related to data in RBUF0 and RBUF1) and RBUFSR (related to data in RBUF).

Transmit data can be loaded to TBUF by SW by writing to the transmit buffer input locations TBUFx (x = 00-31), consisting of 32 consecutive addresses. The data written to one of these input locations is stored in the transmit buffer TBUF. Additionally, the address of the written location is evaluated and can be used for additional control purposes. This 5-bit wide information (named transmit control information TCI) can be used for different purposes in different protocols.

19.1.7.2 FIFO Buffer Structure

To allow easier data setup and handling, an additional data buffering mechanism can be optionally supported. The data buffer is based on the first-in-first-out principle (FIFO) that ensures that the sequence of transferred data words is respected.

If a FIFO buffer structure is used, the data handling scheme (data with associated control information) is similar to the one without FIFO. The additional FIFO buffer can be independently enabled/disabled for transmission and reception (e.g. if data FIFO buffers are available for a specific USIC channel, it is possible to configure the transmit data path without and the receive data path with FIFO buffering).

The transmit FIFO buffer can be reached by SW at 32 consecutive address locations INx instead of TBUFx (x = 0 - 31). The number of the written address location is used as additional control information and is transported through the FIFO together with the written data.
The receive FIFO can be read out at two independent addresses, OUTR and OUTDRL instead of RBUF and RBUFD. A read from the OUTR location triggers the next data packet to be available for the next read (general FIFO mechanism). In order to allow non-intrusive debugging (without risk of data loss), a second address location (OUTDRL) has been introduced. A read at this location delivers the same value as OUTR, but without modifying the FIFO contents.

The transmit FIFO also has the capability to bypass the data stream and to load bypass data to TBUF. This can be used to generate high-priority messages or to send an emergency message if the transmit FIFO runs empty. The transmission control of the FIFO buffer can also use the transfer trigger and transfer gating scheme of the transmission logic for data validation (e.g. to trigger data transfers by events).

Note: The available size of a FIFO data buffer for a USIC channel depends on the specific device. Please refer to the implementation chapter for details about available FIFO buffer capability.

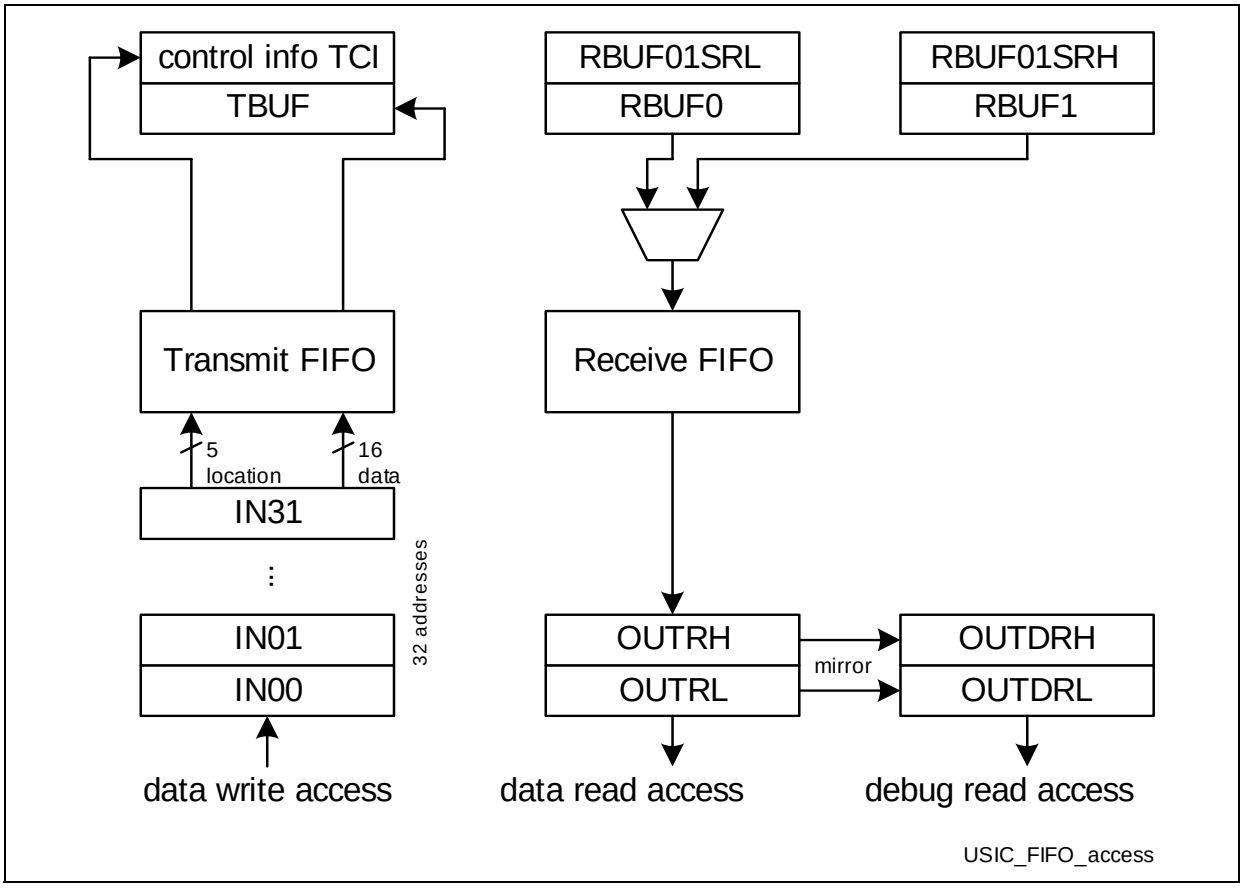


Figure 19-5 Data Access Structure with FIFO

19.2 Operating the USIC

This section describes how to operate the USIC communication channel.
It describes:

- Register Overview (see **Section 19.2.1**)
- General channel operation (see **Section 19.2.2**)
- Channel control and configuration registers (see **Section 19.2.3**)
- Protocol related registers (see **Section 19.2.4**)
- Input stages (see **Section 19.2.5**)
- Input stage control registers (see **Section 19.2.6**)
- Baud rate generation (see **Section 19.2.7**)
- Baud rate and shift control registers (see **Section 19.2.8**)
- Operating the transmit path (see **Section 19.2.9**)
- Operating the receive path (see **Section 19.2.10**)
- Transfer control and status registers (see **Section 19.2.11**)
- Data buffer registers (see **Section 19.2.12**)
- Operating the FIFO data buffer (see **Section 19.2.13**)
- FIFO buffer and bypass registers (see **Section 19.2.14**)

19.2.1 Register Overview

The module itself being 32 bit wide, some registers have been split up in two parts for the 16-bit implementation. Both parts keep the same name as the former 32-bit register, with an additional index. The lower part ends with the index L, whereas the upper (higher) part ends with the index H. Former 32-bit registers consisting of only 16 used bits keep their name (without additional index), because only the used bits appear in the register map.

19.2.1.1 Overview

The table below shows all registers which are required for programming a USIC channel, as well as the FIFO buffer. It summarizes the USIC communication channel registers and defines their relative addresses and the reset values.

19.2.1.2 Register Table

Please note that all registers can be accessed with any access width (8-bit, 16-bit), independent of the described width. Short addressing is not supported.

Table 19-3 USIC Kernel-Related and Kernel Registers

Register Short Name	Register Long Name	Offset Addr.	Reset Value	Description see
Channel Registers (once per USIC channel)				
FDRL	Fractional Divider Register L	04_H	0000_H	Page 19-49
FDRH	Fractional Divider Register H	06_H	0000_H	Page 19-50
KSCFG	Kernel State Configuration Register	$0C_H$	0000_H	Page 19-33
CCR	Channel Control Register	10_H	0000_H	Page 19-28
INPRL	Interrupt Node Pointer Register L	14_H	0000_H	Page 19-35
INPRH	Interrupt Node Pointer Register H	16_H	0000_H	Page 19-36
CCFG	Channel Configuration Register	18_H	$00CF_H$	Page 19-31
BRGL	Baud Rate Generator Register L	$1C_H$	0000_H	Page 19-51
BRGH	Baud Rate Generator Register H	$1E_H$	0000_H	Page 19-53
DX0CR	Input Control Register 0	20_H	0000_H	Page 19-42
DX1CR	Input Control Register 1	24_H	0000_H	Page 19-42
DX2CR	Input Control Register 2	28_H	0000_H	Page 19-42
SCTRL	Shift Control Register L	30_H	0000_H	Page 19-60
SCTRH	Shift Control Register H	32_H	0000_H	Page 19-62
FMRL	Flag Modification Register L	38_H	0000_H	Page 19-70
FMRH	Flag Modification Register H	$3A_H$	0000_H	Page 19-50
TCSRL	Transmit Control/Status Register L	$3C_H$	0000_H	Page 19-63
TCSRH	Transmit Control/Status Register H	$3E_H$	0000_H	Page 19-68
PCRL	Protocol Control Register L	40_H	0000_H	Page 19-37
PCRH	Protocol Control Register H	42_H	0000_H	Page 19-37
PSR	Protocol Status Register	44_H	0000_H	Page 19-38
PSCR	Protocol Status Clear Register	48_H	0000_H	Page 19-39
RBUFD	Receiver Buffer Register for Debugger	$4C_H$	0000_H	Page 19-79
RBUF0	Receiver Buffer 0 Register	50_H	0000_H	Page 19-73
RBUF1	Receiver Buffer 1 Register	54_H	0000_H	Page 19-76
RBUFSR	Receiver Buffer Status Register	58_H	0000_H	Page 19-80

Table 19-3 USIC Kernel-Related and Kernel Registers (cont'd)

Register Short Name	Register Long Name	Offset Addr.	Reset Value	Description see
RBUF	Receiver Buffer Register	$5C_H$	0000_H	Page 19-79
RBUF01SRL	Receiver Buffer 01 Status Register L	60_H	0000_H	Page 19-73
RBUF01SRH	Receiver Buffer 01 Status Register H	62_H	0000_H	Page 19-76
TBUF00 to TBUF31	Transmit Buffer Input Locations	80_H to FF_H	0000_H	Page 19-72
FIFO Buffer Registers (once per USIC channel)				
BYP	Bypass Data Register	100_H	0000_H	Page 19-91
BYPCRL	Bypass Control Register L	104_H	0000_H	Page 19-91
BYPCRH	Bypass Control Register H	106_H	0000_H	Page 19-93
TRBPTRL	Transmit / Receive Buffer Pointer Register L	108_H	0000_H	Page 19-110
TRBPTRH	Transmit / Receive Buffer Pointer Register H	$10A_H$	0000_H	Page 19-110
TBCTRL	Transmit Buffer Control Register L	110_H	0000_H	Page 19-100
TBCTRH	Transmit Buffer Control Register H	112_H	0000_H	Page 19-100
RBCTRL	Receive Buffer Control Register L	114_H	0000_H	Page 19-103
RBCTRH	Receive Buffer Control Register H	116_H	0000_H	Page 19-103
TRBSRL	Transmit / Receive Buffer Status Register L	118_H	0808_H	Page 19-94
TRBSRH	Transmit / Receive Buffer Status Register H	$11A_H$	0000_H	Page 19-97
TRBSCR	Transmit / Receive Buffer Status Clear Register	$11C_H$	0000_H	Page 19-98
OUTRL	Receive Buffer Output Register L	120_H	0000_H	Page 19-108
OUTRH	Receive Buffer Output Register H	122_H	0000_H	Page 19-108
OUTDRL	Receive Buffer Output Register for Debugger L	124_H	0000_H	Page 19-109
OUTDRH	Receive Buffer Output Register for Debugger H	126_H	0000_H	Page 19-109

Table 19-3 USIC Kernel-Related and Kernel Registers (cont'd)

Register Short Name	Register Long Name	Offset Addr.	Reset Value	Description see
IN00 to IN31	Transmit FIFO Buffer Input Locations	180_H to $1FF_H$	0000_H	Page 19-107
-	reserved	$6C_H$	0000_H	-
-	reserved	$6E_H$	0000_H	-

All USIC registers (except bit field KSCFG. SUMCFG) are always reset by a class 3 reset. Bit field KSCFG.SUMCFG is reset by a class 1 reset.

Note: The register bits marked "w" always deliver 0 when read. They are used to modify flipflops in other registers or to trigger internal actions.

19.2.2 Operating the USIC Communication Channel

This section describes how to operate a USIC communication channel, including protocol control and status, mode control and interrupt handling. The following aspects have to be taken into account:

- Enable the USIC module for operation and configure the behavior for the different device operation modes (see **Section 19.2.2.2**).
- Configure the pinning (refer to description in the corresponding protocol section).
- Configure the data structure (shift direction, word length, frame length, polarity, etc.).
- Configure the data buffer structure of the optional FIFO buffer area. A FIFO buffer can only be enabled if the related bit in register CCFG is set.
- Select a protocol by CCR.MODE. A protocol can only be selected if the related bit in register CCFG is set.

19.2.2.1 Protocol Control and Status

The protocol-related status and control information are located in the protocol status register PSR and in the protocol control registers PCRL and PCRH. These registers are shared between all available protocols. As a consequence, the meaning of the bit positions in these registers changes with the selected protocol.

Use of PCRL/H Bits

The signification of the bits in registers PCRL/H is indicated by the alias names for the different protocols.

Table 19-4 Overview of PCRL/H Bits for the ASC Protocol

Bit	Alias	Description
CTR[0]	**SMD**	Sample Mode
CTR[1]	**STPB**	Stop Bits
CTR[2]	**IDM**	Idle Detection Mode
CTR[3]	**SBIEN**	Sync.-Break Interrupt Enable
CTR[4]	**CDEN**	Collision Detection Enable
CTR[5]	**RNIEN**	Receiver Noise Detection Interrupt Enable
CTR[6]	**FEIEN**	Format Error Interrupt Enable
CTR[7]	**FFIEN**	Frame Finished Interrupt Enable
CTR [12:8]	**SP**	Sample Position
CTR [15:13]	**PL**	Pulse Length

Table 19-4 Overview of PCRL/H Bits for the ASC Protocol (cont'd)

Bit	Alias	Description
CTR[31]	**MCLK**	Start-Stop of MCLK
others		Reserved

Table 19-5 Overview of PCRL/H Bits for the SSC Protocol

Bit	Alias	Description
CTR[0]	**MSLSEN**	MSLS Enable
CTR[1]	**SELCTR**	Select Control
CTR[2]	**SELINV**	Select Inversion
CTR[3]	**FEM**	Frame End Mode
CTR [5:4]	**CTQ SEL1**	Input Selection for CTQ
CTR [7:6]	**PCTQ1**	Divider Factor to Generate f_{PDIV}
CTR [12:8]	**DCTQ1**	Divider Factor to Generate $f_{TCTQ / RCTQ}$
CTR[14]	**MSLS IEN**	MSLS Interrupt Enable
CTR[15]	**DX2T IEN**	DX2T Interrupt Enable
CTR [23:16]	**SELO [7:0]**	Select Output
CTR[24]	**TIWEN**	Enable Inter-Word Delay T_{IW}
CTR[31]	**MCLK**	Start Stop of-MCLK
others		Reserved

Table 19-6 Overview of PCRL/H Bits for the IIC Protocol

Bit	Alias	Description
CTR [15:0]	**SLAD [15:0]**	Slave Address
CTR[16]	**ACK00**	Acknowledge 00_H
CTR[17]	**STIM**	Symbol Timing

Table 19-6 Overview of PCRL/H Bits for the IIC Protocol (cont'd)

Bit	Alias	Description
CTR[18]	**SCR IEN**	Start Condition Received Interrupt Enable
CTR[19]	**RSCR IEN**	Repeated Start Condition Received Interrupt Enable
CTR[20]	**PCR IEN**	Stop Condition Received Interrupt Enable
CTR[21]	**NACK IEN**	Non-Acknowledge Interrupt Enable
CTR[22]	**ARL IEN**	Arbitration Lost Interrupt Enable
CTR[23]	**SRR IEN**	Slave Read Request Interrupt Enable
CTR[24]	**ERR IEN**	Error Interrupt Enable
CTR[25]	**SACK DIS**	Slave Acknowledge Disable
CTR [29:26]	**HDEL**	Hardware Delay
CTR[31]	**MCLK**	Start-Stop of MCLK
others		Reserved

Table 19-7 Overview of PCRL/H Bits for the IIS Protocol

Bit	Alias	Description
CTR[0]	**WAGEN**	WA Generation Enable
CTR[1]	**DTEN**	Data Transfers Enable
CTR[2]	**SELINV**	Select Inversion
CTR[4]	**WAFE IEN**	WA Falling Edge Interrupt Enable
CTR[5]	**WARE IEN**	WA Rising Edge Interrupt Enable
CTR[6]	**END IEN**	END Interrupt Enable

Table 19-7 Overview of PCRL/H Bits for the IIS Protocol (cont'd)

Bit	Alias	Description
CTR[15]	**DX2T IEN**	DX2T Interrupt Enable
CTR [21:16]	**TDEL**	Transfer Delay
CTR[31]	**MCLK**	Start-Stop of MCLK
others		Reserved

Use of PSR Flags

The signification of the flags in register PSR is indicated by the alias names for the different protocols. The setting by HW of flags marked "y" in the column "PI" are considered as protocol event and can generate a protocol interrupt (if enabled).

Table 19-8 Overview of PSR Flags for the ASC Protocol

Bit	Alias	Description	PI?
ST[0]	**TXIDLE**	Transmission Idle	-
ST[1]	**RXIDLE**	Reception Idle	-
ST[2]	**SBD**	Synchronization Break Detected	y
ST[3]	**COL**	Collision Detected	y
ST[4]	**RNS**	Receiver Noise Detected	y
ST[5]	**FER0**	Format Error in Stop Bit 0	y
ST[6]	**FER1**	Format Error in Stop Bit 1	y
ST[7]	**RFF**	Receive Frame Finished	y
ST[8]	**TFF**	Transmitter Frame Finished	y

Table 19-9 Overview of PSR Flags for the SSC Protocol

Bit	Alias	Description	PI?
ST[0]	**MSLS**	MSLS Status	-
ST[1]	**DX2S**	DX2S Status	-
ST[2]	**MSLSEV**	MSLS Event Detected	y
ST[3]	**DX2TEV**	DX2T Event Detected	y

Table 19-10 Overview of PSR Flags for the IIC Protocol

Bit	Alias	Description	PI?
ST[0]	**SLSEL**	Slave Select	-
ST[1]	**WTDF**	Wrong TDF Code	y
ST[2]	**SCR**	Start Condition Received	y
ST[3]	**RSCR**	Repeated Start Condition Received	y
ST[4]	**PCR**	Stop Condition Received	y
ST[5]	**NACK**	Non-Acknowledge Received	y
ST[6]	**ARL**	Arbitration Lost	y
ST[7]	**SRR**	Slave Read Request	y
ST[8]	**ERR**	Error	y

Table 19-11 Overview of PSR Flags for the IIS Protocol

Bit	Alias	Description	PI?
ST[0]	**WA**	Word Address	-
ST[1]	**DX2S**	DX2S Status	-
ST[3]	**DX2TEV**	DX2T Event Detected	y
ST[4]	**WAFE**	WA Falling Edge Event Detected	y
ST[5]	**WARE**	WA Rising Edge Event Detected	y
ST[6]	**END**	WA Generation End	y

19.2.2.2 Mode Control

The mode control concept for system control tasks, such as power saving, or suspend request for debugging, allows to program the module behavior under different device operating conditions. The behavior of a communication channel can be programmed for each of the device operating modes, that are requested by the global state control part of the SCU. Therefore, each communication channel has an associated register **KSCFG** defining its behavior in the following operating modes:

- Normal operation:
  This operating mode is the default operating mode when neither a suspend request nor a clock-off request are pending. The module clock is not switched off and the USIC registers can be read or written. The channel behavior is defined by KSCFG.NOMCFG.

- Suspend mode:
  This operating mode is requested when a suspend request is pending in the device. The module clock is not switched off and the USIC registers can be read or written. The channel behavior is defined by KSCFG.SUMCFG.
- Clock-off mode:
  This operating mode is requested for power saving purposes. The module clock is switched off automatically when all channels of the USIC module reached their specified state in a stop mode. In this case, USIC registers can not be accessed. The channel behavior is defined by KSCFG.COMCFG.

The behavior of a USIC communication channel can be programmed for each of the device operating modes (normal operation, suspend mode, clock-off mode). Therefore, the USIC communication channel provides four kernel modes, as shown in **Table 19-12**.

Table 19-12 USIC Communication Channel Behavior

Kernel Mode	Channel Behavior	Code
run mode 0	channel operation as specified, no impact on data transfer	00_B
run mode 1		01_B
stop mode 0	explicit stop condition as described in the protocol chapters	10_B
stop mode 1		11_B

Generally, bit field KSCFG.NOMCFG should be configured for run mode 0 as default setting for standard operation. If a communication channel should not react to a suspend request (and to continue its operation as in normal mode), bit field KSCFG.SUMCFG has to be configured with the same value as KSCFG.NOMCFG. If the communication channel should show a different behavior and stop operation when a specific stop condition is reached, the code for stop mode 0 or stop mode 1 have to be written to KSCFG.SUMCFG.
A similar mechanism applies for the clock-off mode with the possibility to program the desired behavior by bit field KSCFG.COMCFG.
The stop conditions are defined for the selected protocol (see mode control description in the protocol section).

Note: The stop mode selection strongly depends on the application needs and it is very unlikely that different stop modes are required in parallel in the same application. As a result, only one stop mode type (either 0 or 1) should be used in the bit fields in register KSCFG. Do not mix stop mode 0 and stop mode 1 and avoid transitions from stop mode 0 to stop mode 1 (or vice versa) for the same communication channel.

If the module clock is disabled by KSCFG.MODEN = 0 or in clock-off mode when the stop condition is reached (in stop mode 0 or 1), the module can not be accessed by read or write operations (except register KSCFG that can always be accessed).

19.2.2.3 General Channel Events and Interrupts

The general event and interrupt structure is shown in **Figure 19-6**. If a defined condition is met, an event is detected and an event indication flag becomes automatically set. The flag stays set until it is cleared by SW. If enabled, an interrupt can be generated if an event is detected. The actual status of the event indication flag has no influence on the interrupt generation. As a consequence, the event indication flag does not need to be cleared to generate further interrupts.
Additionally, the service request output SRx of the USIC channel that becomes activated in case of an event can be selected by an interrupt node pointer. This structure allows to assign events to interrupts, e.g. depending on the application, several events can share the same interrupt routine (several events activate the same SRx output) or can be handled individually (only one event activates an SRx output).
The SRx outputs are connected to interrupt control registers to handle the CPU reaction to the service requests. This assignment is described in the implementation chapter, see **Section 19.7.4**.

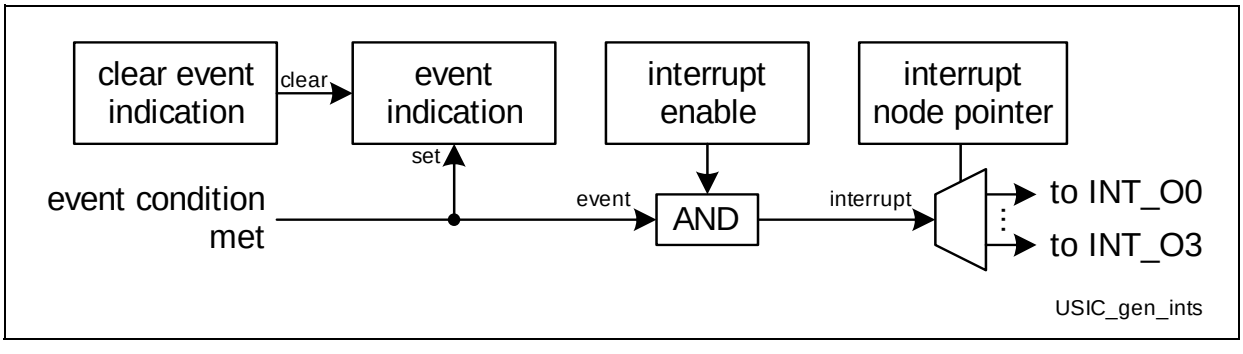


Figure 19-6 General Event and Interrupt Structure

19.2.2.4 Data Transfer Events and Interrupts

The data transfer events are based on the transmission or reception of a data word. The related indication flags are located in register PSR. All events can be individually enabled for interrupt generation.

- Receive event to indicate that a data word has been received:
  If a new received word becomes available in the receive buffer RBUF, either a receive event or an alternative receive event occurs.
  The receive event occurs if bit RBUFSR.PERR = 0. It is indicated by flag PSR.RIF and can lead to a receive interrupt (if enabled).
- Receiver start event to indicate that a data word reception has started:
  When the receive clock edge that shifts in the first bit of a new data word is detected and reception is enabled, a receiver start event occurs. It is indicated by flag PSR.RSIF and can lead to a transmit buffer interrupt (if enabled).
  In full duplex mode, this event follows half a shift clock cycle after the transmit buffer event and indicates when the shift control settings are internally “frozen” for the current data word reception and a new setting can be programmed.
  In SSC and IIS mode, the transmit data valid flag **TCSRL**.TDV is cleared in single shot mode with the receiver start event.
- Alternative receive event to indicate that a specific data word has been received:
  If a new received word becomes available in the receive buffer RBUF, either a receive event or an alternative receive event occurs.
  The alternative receive event occurs if bit RBUFSR.PERR = 1. It is indicated by flag PSR.AIF and can lead to an alternative receive interrupt (if enabled).
  Depending on the selected protocol, bit RBUFSR.PERR is set to indicate a parity error in ASC mode, the reception of the first byte of a new frame in IIC mode, and the WA information about right/left channel in IIS mode, whereas it is always 0 in SSC mode.
- Transmit shift event to indicate that a data word has been transmitted:
  A transmit shift event occurs with the last shift clock edge of a data word. It is indicated by flag PSR.TSIF and can lead to a transmit shift interrupt (if enabled).
- Transmit buffer event to indicate that a data word transmission has been started:
  When a data word from the transmit buffer TBUF has been loaded to the shift register and a new data word can be written to TBUF, a transmit buffer event occurs. This happens with the transmit clock edge that shifts out the first bit of a new data word and transmission is enabled. It is indicated by flag PSR.TBIF and can lead to a transmit buffer interrupt (if enabled).
  This event also indicates when the shift control settings (word length, shift direction, etc.) are internally “frozen” for the current data word transmission.
  In ASC and IIC mode, the transmit data valid flag **TCSRL**.TDV is cleared in single shot mode with the transmit buffer event.
- Data lost event to indicate a loss of the oldest received data word:
  If the data word available in register RBUF (oldest data word from RBUF0 or RBUF1)

has not been read out before it becomes overwritten with new incoming data, this event occurs. It is indicated by flag PSR.DLIF and can lead to a protocol interrupt (if enabled).

The table below shows the registers, bits and bit fields indicating the data transfer events and controlling the interrupts of a USIC channel.

Table 19-13 Data Transfer Events and Interrupt Handling

Event	Indication Flag	Indication cleared by	Interrupt enabled by	SRx Output selected by
Receive event	PSR. RIF	PSCR. CRIF	CCR. RIEN	INPRL. RINP
Receiver start event	PSR. RSIF	PSCR. CRSIF	CCR. RSIEN	INPRL. TBINP
Alternative receive event	PSR. AIF	PSCR. CAIF	CCR. AIEN	INPRL. AINP
Transmit shift event	PSR. TSIF	PSCR. CTSIF	CCR. TSIEN	INPRL. TSINP
Transmit buffer event	PSR. TBIF	PSCR. CTBIF	CCR. TBIEN	INPRL. TBINP
Data lost event	PSR. DLIF	PSCR. CDLIF	CCR. DLIEN	INPRH. PINP

The following figure shows the transmit events and interrupts.

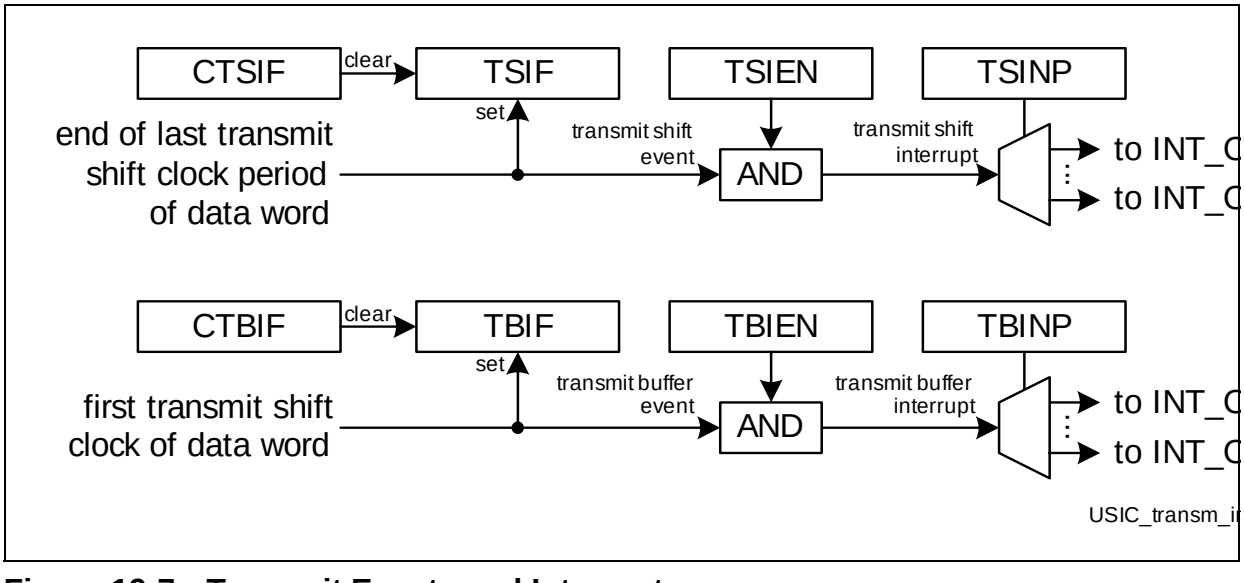


Figure 19-7 Transmit Events and Interrupts

The following figure shows the receive events and interrupts.

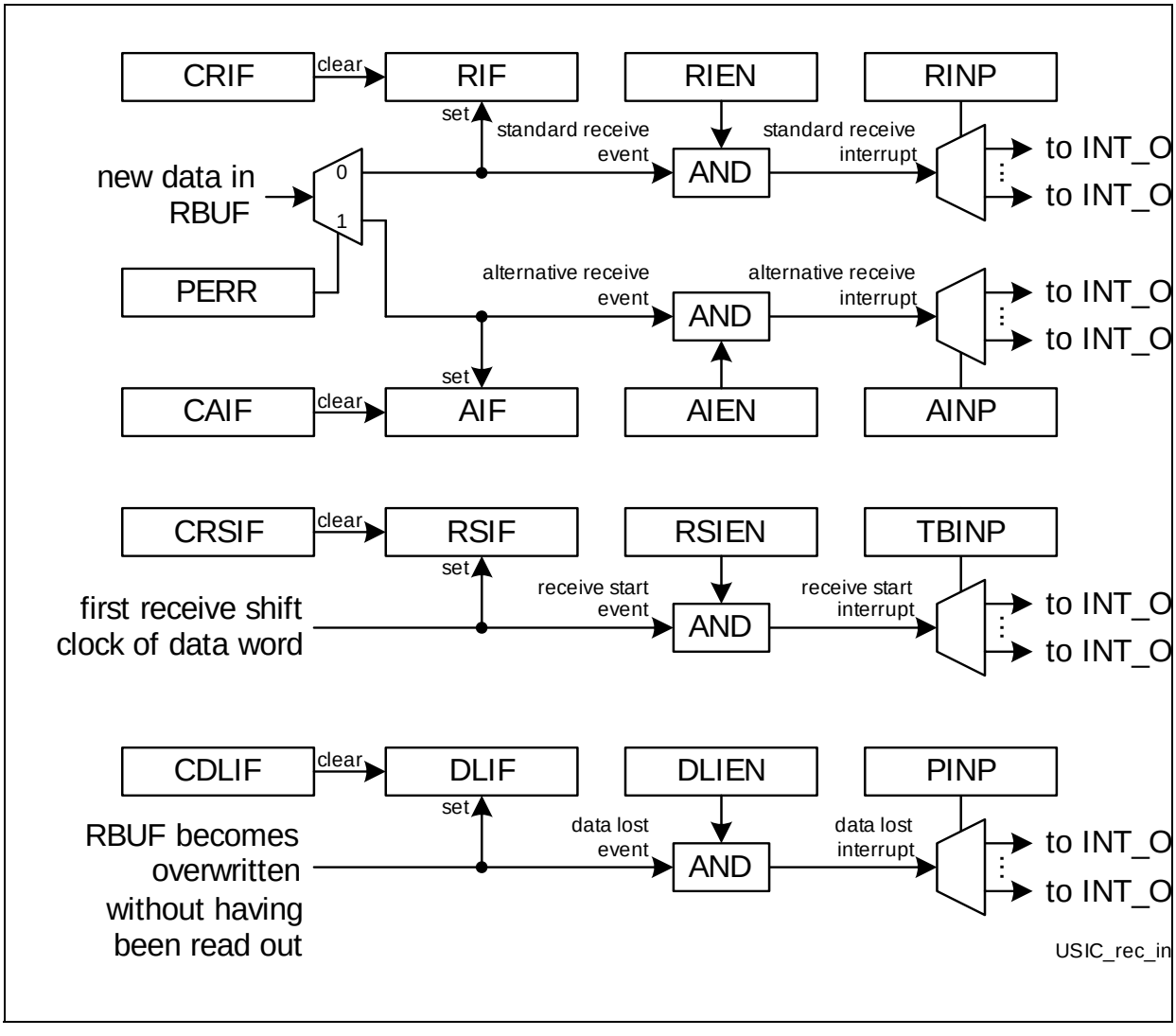


Figure 19-8 Receive Events and Interrupts

19.2.2.5 Protocol-specific Events and Interrupts

These events are related to protocol-specific actions that are described in the corresponding protocol chapters. The related indication flags are located in register PSR. All events can be individually enabled for the generation of the common protocol interrupt.

- Protocol-specific events in ASC mode:
  Synchronization break, data collision on the transmit line, receiver noise, format error in stop bits, receiver frame finished, transmitter frame finished
- Protocol-specific events in SSC mode:
  MSLS event (start-end of frame in master mode), DX2T event (start/end of frame in slave mode), both based on slave select signals
- Protocol-specific events in IIC mode:
  Wrong transmit code (error in frame sequence), start condition received, repeated start condition received, stop condition received, non-acknowledge received, arbitration lost, slave read request, other general errors
- Protocol-specific events in IIS mode:
  DX2T event (change on WA line), WA falling edge or rising edge detected, WA generation finished

Table 19-14 Protocol-specific Events and Interrupt Handling

Event	Indication Flag	Indication cleared by	Interrupt enabled by	SRx Output selected by
Protocol-specific events in ASC mode	PSR. ST[8:2]	PSCR. CST[8:2]	PCRL. CTR[7:3]	INPRH. PINP
Protocol-specific events in SSC mode	PSR. ST[3:2]	PSCR. CST[3:2]	PCRL. CTR[15:14]	INPRH. PINP
Protocol-specific events in IIC mode	PSR. ST[8:1]	PSCR. CST[8:1]	PCRH. CTR[24:18]	INPRH. PINP
Protocol-specific events in IIS mode	PSR. ST[6:3]	PSCR. CST[6:3]	PCRL. CTR[6:4], PCRL. CTR[15]	INPRH. PINP

19.2.3 Channel Control and Configuration Registers

19.2.3.1 Channel Control Register

The channel control register contains the enable/disable bits for interrupt generation on channel events, the control of the parity generation and the protocol selection of a USIC channel.

CCR
Channel Control Register **(10_H)** **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AI EN	RI EN	TBI EN	TSI EN	DLI EN	RSI EN	PM		0				MODE			
rw	rw	rw	rw	rw	rw	rw		r				rw			

Field	Bits	Type	Description
MODE	[3:0]	rw	**Operating Mode** This bit field selects the protocol for this USIC channel. Selecting a protocol that is not available (see register CCFG) or a reserved combination disables the USIC channel. When switching between two protocols, the USIC channel has to be disabled before selecting a new protocol. In this case, registers PCRH, PCRL, and PSR have to be cleared or updated by SW. 00_H The USIC channel is disabled. All protocol-related state machines are set to an idle state. 01_H The SSC (SPI) protocol is selected. 02_H The ASC (SCI, UART) protocol is selected. 03_H The IIS protocol is selected. 04_H The IIC protocol is selected. else reserved

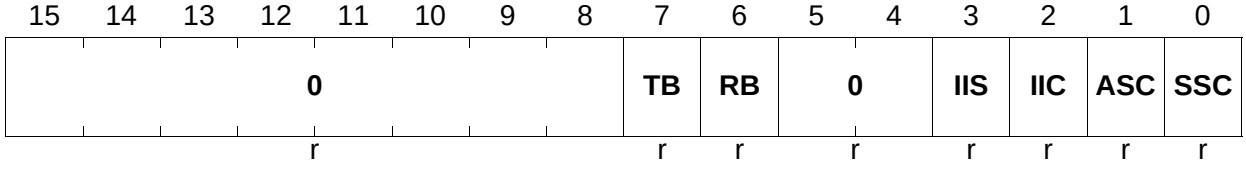
Field	Bits	Type	Description
PM	[9:8]	rw	**Parity Mode** This bit field defines the parity generation of the sampled input values. 00_B The parity generation is disabled. 01_B reserved 10_B Even parity is selected (parity bit = 1 on odd number of 1s in data, parity bit = 0 on even number of 1s in data). 11_B Odd parity is selected (parity bit = 0 on odd number of 1s in data, parity bit = 1 on even number of 1s in data).
RSIEN	10	rw	**Receiver Start Interrupt Enable** This bit enables the interrupt generation in case of a receiver start event. 0_B The receiver start interrupt is disabled. 1_B The receiver start interrupt is enabled. In case of a receiver start event, the service request output SRx indicated by INPRL.TBINP is activated.
DLIEN	11	rw	**Data Lost Interrupt Enable** This bit enables the interrupt generation in case of a data lost event (data received in RBUFx while RDVx = 1). 0_B The data lost interrupt is disabled. 1_B The data lost interrupt is enabled. In case of a data lost event, the service request output SRx indicated by INPRH.PINP is activated.
TSIEN	12	rw	**Transmit Shift Interrupt Enable** This bit enables the interrupt generation in case of a transmit shift event. 0_B The transmit shift interrupt is disabled. 1_B The transmit shift interrupt is enabled. In case of a transmit shift interrupt event, the service request output SRx indicated by INPRL.TSINP is activated.

Field	Bits	Type	Description
TBIEN	13	rw	**Transmit Buffer Interrupt Enable** This bit enables the interrupt generation in case of a transmit buffer event. 0_B The transmit buffer interrupt is disabled. 1_B The transmit buffer interrupt is enabled. In case of a transmit buffer event, the service request output SRx indicated by INPRL.TBINP is activated.
RIEN	14	rw	**Receive Interrupt Enable** This bit enables the interrupt generation in case of a receive event. 0_B The receive interrupt is disabled. 1_B The receive interrupt is enabled. In case of a receive event, the service request output SRx indicated by INPRL.RINP is activated.
AIEN	15	rw	**Alternative Receive Interrupt Enable** This bit enables the interrupt generation in case of a alternative receive event. 0_B The alternative receive interrupt is disabled. 1_B The alternative receive interrupt is enabled. In case of an alternative receive event, the service request output SRx indicated by INPRL.AINP is activated.
0	[7:4]	r	**Reserved** read as 0; should be written with 0.

19.2.3.2 Channel Configuration Register

The channel configuration register contains bits describing the functionality that is available in the USIC channel.

CCFG
Channel Configuration Register (18_H) Reset Value: $00CF_H$



Field	Bits	Type	Description
SSC	0	r	**SSC Protocol Available** This bit indicates if the SSC protocol is available. 0_B The SSC protocol is not available. 1_B The SSC protocol is available.
ASC	1	r	**ASC Protocol Available** This bit indicates if the ASC protocol is available. 0_B The ASC protocol is not available. 1_B The ASC protocol is available.
IIC	2	r	**IIC Protocol Available** This bit indicates if the IIC functionality is available. 0_B The IIC protocol is not available. 1_B The IIC protocol is available.
IIS	3	r	**IIS Protocol Available** This bit indicates if the IIS protocol is available. 0_B The IIS protocol is not available. 1_B The IIS protocol is available.
RB	6	r	**Receive FIFO Buffer Available** This bit indicates if an additional receive FIFO buffer is available. 0_B A receive FIFO buffer is not available. 1_B A receive FIFO buffer is available.
TB	7	r	**Transmit FIFO Buffer Available** This bit indicates if an additional transmit FIFO buffer is available. 0_B A transmit FIFO buffer is not available. 1_B A transmit FIFO buffer is available.

Field	Bits	Type	Description
0	[5:4], [15:8]	r	**Reserved** read as 0; should be written with 0.

19.2.3.3 Kernel State Configuration Register

The kernel state configuration register KSCFG allows the selection of the desired kernel modes for the different device operating modes.

KSCFG
Kernel State Configuration Register (0C$_H$) **Reset Value: 0000$_H$**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BP COM	**0**	**COMCFG**		**BP SUM**	**0**	**SUMCFG**		**BP NOM**	**0**	**NOMCFG**		**0**	**0**	**BP MOD EN**	**MOD EN**
w	r	rw		w	r	rw		w	r	rw		r	r	w	rw

Field	Bits	Type	Description
MODEN	0	rw	**Module Enable** This bit enables the module kernel clock and the module functionality. 0_B The module is switched off immediately (without respecting a stop condition). It does not react on mode control actions and the module clock is switched off. The module does not react on read accesses and ignores write accesses (except to KSCFG). 1_B The module is switched on and can operate. After writing 1 to MODEN, it is recommended to read register KSCFG to avoid pipeline effects in the control block before accessing other ADC registers.
BPMODEN	1	w	**Bit Protection for MODEN** This bit enables the write access to the bit MODEN. It always reads 0. 0_B MODEN is not changed. 1_B MODEN is updated with the written value.
NOMCFG	[5:4]	rw	**Normal Operation Mode Configuration** This bit field defines the kernel mode applied in normal operation mode. 00_B Run mode 0 is selected. 01_B Run mode 1 is selected. 10_B Stop mode 0 is selected. 11_B Stop mode 1 is selected.

Field	Bits	Type	Description
BPNOM	7	w	**Bit Protection for NOMCFG** This bit enables the write access to the bit field NOMCFG. It always reads 0. 0_B NOMCFG is not changed. 1_B NOMCFG is updated with the written value.
SUMCFG	[9:8]	rw	**Suspend Mode Configuration** This bit field defines the kernel mode applied in suspend mode. Coding like NOMCFG.
BPSUM	11	w	**Bit Protection for SUMCFG** This bit enables the write access to the bit field SUMCFG. It always reads 0. 0_B SUMCFG is not changed. 1_B SUMCFG is updated with the written value.
COMCFG	[13:12]	rw	**Clock Off Mode Configuration** This bit field defines the kernel mode applied in clock-off mode. Coding like NOMCFG.
BPCOM	15	w	**Bit Protection for COMCFG** This bit enables the write access to the bit field COMCFG. It always reads 0. 0_B COMCFG is not changed. 1_B COMCFG is updated with the written value.
0	[3:2], 6, 10, 14	r	**Reserved** returns 0 if read; should be written with 0;

19.2.3.4 Interrupt Node Pointer Registers

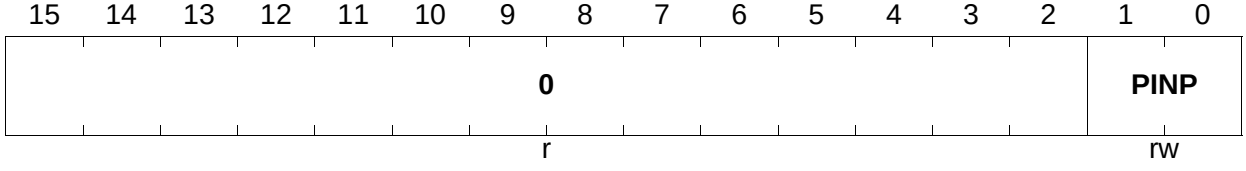
The interrupt node pointer registers define the service request output SRx that is activated if the corresponding event occurs and interrupt generation is enabled.

INPRL
Interrupt Node Pointer Register L (14_H) **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		**AINP**		**0**		**RINP**		**0**		**TBINP**		**0**		**TSINP**	
r		rw		r		rw		r		rw		r		rw	

Field	Bits	Type	Description
TSINP	[1:0]	rw	**Transmit Shift Interrupt Node Pointer** This bit field defines which service request output SRx will be activated in case of a transmit shift interrupt. 00_B Output SR0 will be activated. 01_B Output SR1 will be activated. 10_B Output SR2 will be activated. 11_B Output SR3 will be activated.
TBINP	[5:4]	rw	**Transmit Buffer Interrupt Node Pointer** This bit field defines which service request output SRx will be activated in case of a transmit buffer interrupt or a receive start interrupt. Coding like TSINP.
RINP	[9:8]	rw	**Receive Interrupt Node Pointer** This bit field defines which service request output SRx will be activated in case of a receive interrupt. Coding like TSINP.
AINP	[13:12]	rw	**Alternative Receive Interrupt Node Pointer** This bit field defines which service request output SRx will be activated in case of a alternative receive interrupt. Coding like TSINP.
0	[3:2], [7:6], [11:10], [15:14]	r	**Reserved** read as 0; should be written with 0.

INPRH
Interrupt Node Pointer Register H (16_H) Reset Value: 0000_H



Field	Bits	Type	Description
PINP	[1:0]	rw	**Protocol Interrupt Node Pointer** This bit field defines which service request output SRx will be activated in case of a protocol interrupt. 00_B Output SR0 will be activated. 01_B Output SR1 will be activated. 10_B Output SR2 will be activated. 11_B Output SR3 will be activated.
0	[15:2]	r	**Reserved** read as 0; should be written with 0.

19.2.4 Protocol Related Registers

19.2.4.1 Protocol Control Registers

The bits in the protocol control registers define protocol-specific functions. They have to be configured by SW before enabling a new protocol. Only the bits used for the selected protocol are taken into account, whereas the other bit positions always read as 0. The protocol-specific meaning is described in the related protocol section.

PCRL
Protocol Control Register L **(40_H)** **Reset Value: 0000_H**

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

Field	Bits	Type	Description
CTRx (x = 15-0)	x	rw	**Protocol Control Bit x** This bit is a protocol control bit.

PCRH
Protocol Control Register H **(42_H)** **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTR 31	**CTR 30**	**CTR 29**	**CTR 28**	**CTR 27**	**CTR 26**	**CTR 25**	**CTR 24**	**CTR 23**	**CTR 22**	**CTR 21**	**CTR 20**	**CTR 19**	**CTR 18**	**CTR 17**	**CTR 16**
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
CTRx (x = 30-16)	x - 16	rwh	**Protocol Control Bit x** This bit is a protocol control bit that can be overwritten by protocol-specific information.
CTR31	15	rwh	**Start Stop Control of MCLK** This bit controls the start and the stop of the MCLK signal. 0_B Signal MCLK is not generated (MCLK = 0). 1_B Signal MCLK generation is enabled.

19.2.4.2 Protocol Status Register

The flags in the protocol status register can be cleared by writing a 1 to the corresponding bit position in register PSCR. Writing a 1 to a bit position in PSR sets the corresponding flag, but doesn't lead to further actions (no interrupt generation). Writing a 0 has no effect. These flags should be cleared by SW before enabling a new protocol. The protocol-specific meaning is described in the related protocol section.

PSR
Protocol Status Register **(44_H)** **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AIF	**RIF**	**TBIF**	**TSIF**	**DLIF**	**RSIF**	**ST 9**	**ST 8**	**ST 7**	**ST 6**	**ST 5**	**ST 4**	**ST 3**	**ST 2**	**ST 1**	**ST 0**
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
STx (x = 9-0)	x	rwh	**Protocol Status Flag x** See protocol specific description.
RSIF	10	rwh	**Receiver Start Indication Flag** 0_B A receiver start event has not occurred. 1_B A receiver start event has occurred.
DLIF	11	rwh	**Data Lost Indication Flag** 0_B A data lost event has not occurred. 1_B A data lost event has occurred.
TSIF	12	rwh	**Transmit Shift Indication Flag** 0_B A transmit shift event has not occurred. 1_B A transmit shift event has occurred.
TBIF	13	rwh	**Transmit Buffer Indication Flag** 0_B A transmit buffer event has not occurred. 1_B A transmit buffer event has occurred.
RIF	14	rwh	**Receive Indication Flag** 0_B A receive event has not occurred. 1_B A receive event has occurred.
AIF	15	rwh	**Alternative Receive Indication Flag** 0_B An alternative receive event has not occurred. 1_B An alternative receive event has occurred.

19.2.4.3 Protocol Status Clear Register

Read accesses to this register always deliver 0 at all bit positions.

PSCR
Protocol Status Clear Register (48_H) Reset Value: 0000_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CAIF	**CRIF**	**CTBIF**	**CTSIF**	**CDLIF**	**CRSIF**	**CST9**	**CST8**	**CST7**	**CST6**	**CST5**	**CST4**	**CST3**	**CST2**	**CST1**	**CST0**
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
CSTx (x = 9-0)	x	w	**Clear Status Flag x in PSR** 0_B No action 1_B Flag PSR.STx is cleared.
CRSIF	10	w	**Clear Receiver Start Indication Flag** 0_B No action 1_B Flag PSR.RSIF is cleared.
CDLIF	11	w	**Clear Data Lost Indication Flag** 0_B No action 1_B Flag PSR.DLIF is cleared.
CTSIF	12	w	**Clear Transmit Shift Indication Flag** 0_B No action 1_B Flag PSR.TSIF is cleared.
CTBIF	13	w	**Clear Transmit Buffer Indication Flag** 0_B No action 1_B Flag PSR.TBIF is cleared.
CRIF	14	w	**Clear Receive Indication Flag** 0_B No action 1_B Flag PSR.RIF is cleared.
CAIF	15	w	**Clear Alternative Receive Indication Flag** 0_B No action 1_B Flag PSR.AIF is cleared.

19.2.5 Operating the Input Stages

All three input stages offer the same feature set. They are used for all protocols, because the signal conditioning can be adapted in a very flexible way and the digital filters can be switched on and off separately.

19.2.5.1 General Input Structure

All input stages are built in a similar way as shown in **Figure 19-9**. All enable/disable functions and selections are controlled independently for each input stage by bits in the registers DX0CR, DX1CR, and DX2CR.
The desired input signal can be selected among the input lines DXnA to DXnG and a permanent 1-level by programming bit field DSEL. Please refer to the implementation chapter for the device-specific input signal assignment. Bit DPOL allows a polarity inversion of the selected input signal to adapt the input signal polarity to the internal polarity of the data shift unit and the protocol state machine. For some protocols, the input signals can be directly forwarded to the data shift unit for the data transfers (DSEN = 0, INSW = 1) without any further signal conditioning. In this case, the data path does not contain any delay due to synchronization or filtering.
In the case of noise on the input signals, there is the possibility to synchronize the input signal (signal DXnS is synchronized to f_{SYS}) and additionally to enable a digital noise filter in the signal path. The synchronized input signal (and optionally filtered if DFEN = 1) is taken into account by DSEN = 1. Please note that the synchronization leads to a delay in the signal path of 2-3 times the period of f_{SYS}.

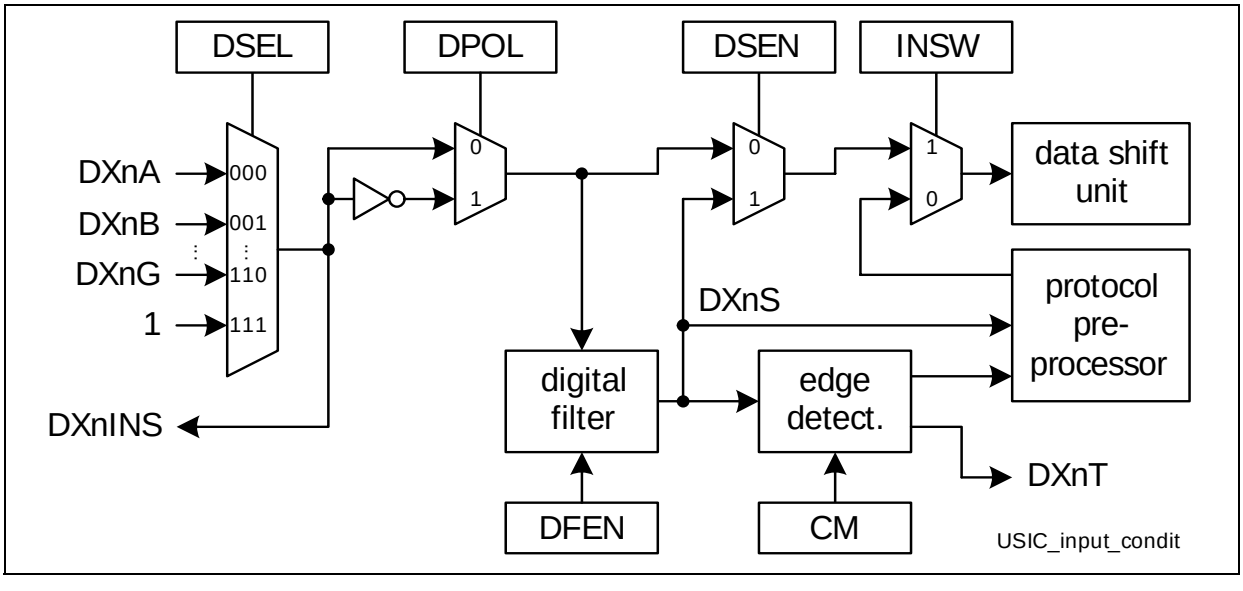


Figure 19-9 Input Conditioning

If the input signals are handled by a protocol pre-processor, the data shift unit is directly connected to the protocol pre-processor by INSW = 0. The protocol pre-processor is

connected to the synchronized input signal DXnS and, depending on the selected protocol, also evaluates the edges.

19.2.5.2 Digital Filter

The digital filter can be enabled to reduce noise on the input signals. Before being filtered, the input signal becomes synchronized to f_{SYS}. If the filter is disabled, signal DXnS corresponds to the synchronized input signal. If the filter is enabled, pulses shorter than one filter sampling period are suppressed in signal DXnS. After an edge of the synchronized input signal, signal DXnS changes to the new value if two consecutive samples of the new value have been detected.
In order to adapt the filter sampling period to different applications, it can be programmed. The first possibility is the system frequency f_{SYS}. Longer pulses can be suppressed if the fractional divider output frequency f_{FD} is selected. This frequency is programmable in a wide range and can also be used to determine the baud rate of the data transfers.
In addition to the synchronization delay of 2-3 periods of f_{SYS}, an enabled filter adds a delay of up to two filter sampling periods between the selected input and signal DXnS.

19.2.5.3 Edge Detection

The synchronized (and optionally filtered) signal DXnS can be used as input to the data shift unit and is also an input to the selected protocol pre-processor. If the protocol pre-processor does not use the DXnS signal for protocol-specific handling, DXnS can be used for other tasks, e.g. to control data transmissions in master mode (a data word can be tagged valid for transmission, see chapter about data buffering).
A programmable edge detection indicates that the desired event has occurred by activating the trigger signal DXnT (introducing a delay of one period of f_{SYS} before a reaction to this event can take place).

19.2.5.4 Selected Input Monitoring

The selected input signal of each input stage has been made available with the signals DX0INS, DX1INS, and DX2INS. These signals can be used in the system to trigger other actions, e.g. to generate interrupts.

19.2.5.5 Loop Back Mode

The USIC transmitter output signals can be connected to the corresponding receiver inputs of the same communication channel in loop back mode. Therefore, the input “G” of the input stages that are needed for the selected protocol have to be selected. In this case, drivers for ASC, SSC, and IIS can be evaluated on-chip without the connections to port pins. Data transferred by the transmitter can be received by the receiver as if it would have been sent by another communication partner.

19.2.6 Input Stage Registers

19.2.6.1 Input Control Registers

The input control registers contain the bits to define the characteristics of the input stages (input stage DX0 is controlled by register DX0CR, etc.).

DX0CR
Input Control Register 0 **(20_H)** **Reset Value: 0000_H**
DX1CR
Input Control Register 1 **(24_H)** **Reset Value: 0000_H**
DX2CR
Input Control Register 2 **(28_H)** **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DXS	0			CM		SF SEL	D POL	0	DS EN	DF EN	IN SW	0	DSEL		
rh	r			rw		rw	rw	r	rw	rw	rw	r	rw		

Field	Bits	Type	Description
DSEL	[2:0]	rw	**Data Selection for Input Signal** This bit field defines the input data signal for the corresponding input line for protocol pre-processor. The selection can be made from the input vector DXn[G:A]. 000_B The data input DXnA is selected. 001_B The data input DXnB is selected. ... 110_B The data input DXnG is selected. 111_B The data input is always 1.
INSW	4	rw	**Input Switch** This bit defines if the data shift unit input is derived from the input data path DXn or from the selected protocol pre-processors. 0_B The input of the data shift unit is controlled by the protocol pre-processor. 1_B The input of the data shift unit is connected to the selected data input line. This setting is used if the signals are directly derived from an input pin without treatment by the protocol pre-processor.

Field	Bits	Type	Description
DFEN	5	rw	**Digital Filter Enable** This bit enables/disables the digital filter for signal DXnS. 0_B The input signal is not digitally filtered. 1_B The input signal is digitally filtered.
DSEN	6	rw	**Data Synchronization Enable** This bit selects if the asynchronous input signal or the synchronized (and optionally filtered) signal DXnS can be used as input for the data shift unit. 0_B The unsynchronized signal can be taken as input for the data shift unit. 1_B The synchronized signal can be taken as input for the data shift unit.
DPOL	8	rw	**Data Polarity for DXn** This bit defines the signal polarity of the input signal. 0_B The input signal is not inverted. 1_B The input signal is inverted.
SFSEL	9	rw	**Sampling Frequency Selection** This bit defines the sampling frequency of the digital filter for the synchronized signal DXnS. 0_B The sampling frequency is f_{SYS}. 1_B The sampling frequency is f_{FD}.
CM	[11:10]	rw	**Combination Mode** This bit field selects which edge of the synchronized (and optionally filtered) signal DXnS actives the trigger output DXnT of the input stage. 00_B The trigger activation is disabled. 01_B A rising edge activates DXnT. 10_B A falling edge activates DXnT. 11_B Both edges activate DXnT.
DXS	15	rh	**Synchronized Data Value** This bit indicates the value of the synchronized (and optionally filtered) input signal. 0_B The current value of DXnS is 0. 1_B The current value of DXnS is 1.
0	3, 7, [14:12]	r	**Reserved** read as 0; should be written with 0.

19.2.7 Operating the Baud Rate Generator

The following blocks can be configured to operate the baud rate generator.

19.2.7.1 Fractional Divider

The fractional divider generates its output frequency f_{FD} by dividing the input frequency f_{SYS} either by an integer factor n or by multiplication by n/1024. It has two operating modes:

- Normal divider mode (FDRL.DM = 01_B):
  In this mode, the output frequency f_{FD} is derived from the input clock f_{SYS} by an integer division by a value between 1 and 1024. The division is based on a counter FDRH.RESULT that is incremented by 1 with f_{SYS}. After reaching the value $3FF_H$, the counter is loaded with FDRL.STEP and then continues counting. In order to achieve $f_{FD} = f_{SYS}$, the value of STEP has to be programmed with $3FF_H$.
  The output frequency in normal divider mode is defined by the equation:

(19.1)

$$f_{FD} = f_{SYS} \times \frac{1}{n} \qquad \text{with} \qquad n = 1024 - STEP$$

- Fractional divider mode (FDRL.DM = 10_B):
  In this mode, the output frequency f_{FD} is derived from the input clock f_{SYS} by a fractional multiplication by n/1024 for a value of n between 0 and 1023. In general, the fractional divider mode allows to program the average output clock frequency with a finer granularity than in normal divider mode. Please note that in fractional divider mode f_{FD} can have a maximum period jitter of one f_{SYS} period. This jitter is not accumulated over several cycles.
  The frequency f_{FD} is generated by an addition of FDRL.STEP to FDRH.RESULT with f_{SYS}. The frequency f_{FD} is based on the overflow of the addition result over $3FF_H$.
  The output frequency in fractional divider mode is defined by the equation:

(19.2)

$$f_{FD} = f_{SYS} \times \frac{n}{1024} \qquad \text{with} \qquad n = STEP$$

The output frequency f_{FD} of the fractional divider is selected for baud rate generation by BRGL.CLKSEL = 00_B.

19.2.7.2 External Frequency Input

The baud rate can be generated referring to an external frequency input (instead of to f_{SYS}) if in the selected protocol the input stage DX1 is not needed (DX1CTR.INSW = 0). In this case, an external frequency input signal at the DX1 input stage can be synchronized and sampled with the system frequency f_{SYS}. It can be optionally filtered by the digital filter in the input stage. This feature allows data transfers with frequencies

that can not be generated by the device itself, e.g. for specific audio frequencies.
If BRGL.CLKSEL = 10_B, the trigger signal DX1T determines f_{DX1}. In this mode, either the rising edge, the falling edge, or both edges of the input signal can be used for baud rate generation, depending on the configuration of the DX1T trigger event by bit field DX1CTR.CM. The signal MCLK toggles with each trigger event of DX1T.
If BRGL.CLKSEL = 11_B, the rising edges of the input signal can be used for baud rate generation. The signal MCLK represents the synchronized input signal DX1S.
Both, the high time and the low time of external input signal must each have a length of minimum 2 periods of f_{SYS} to be used for baud rate generation.

19.2.7.3 Protocol-Related Counter in Divider Mode

In divider mode, the protocol-related counter is used for an integer division delivering the output frequency f_{PDIV}. Additionally, two divider stages with a fixed division by 2 provide the output signals MCLK and SCLK with 50% duty cycle. If the fractional divider mode is used, the maximum fractional jitter of 1 period of f_{SYS} can also appear in these signals. The outputs frequencies of this divider is controlled by registers BRGL and BRGH.
In order to define a frequency ratio between the master clock MCLK and the shift clock SCLK, the divider stage for MCLK is located in front of the divider by PDIV+1, whereas the divider stage for SCLK is located at the output of this divider.

$$f_{MCLK} = \frac{f_{PIN}}{2} \tag{19.3}$$

$$f_{SCLK} = \frac{f_{PDIV}}{2} \tag{19.4}$$

In the case that the master clock is used as reference for external devices (e.g. for IIS components) and a fixed phase relation to SCLK and other timing signals is required, it is recommended to use the MCLK signal as input for the PDIV divider. If the MCLK signal is not used or a fixed phase relation is not necessary, the faster frequency f_{PIN} can be selected as input frequency.

$$f_{PDIV} = \begin{cases} f_{PIN} \times \frac{1}{PDIV + 1} & \text{if} \quad PPPEN = 0 \\ f_{MCLK} \times \frac{1}{PDIV + 1} & \text{if} \quad PPPEN = 1 \end{cases} \tag{19.5}$$

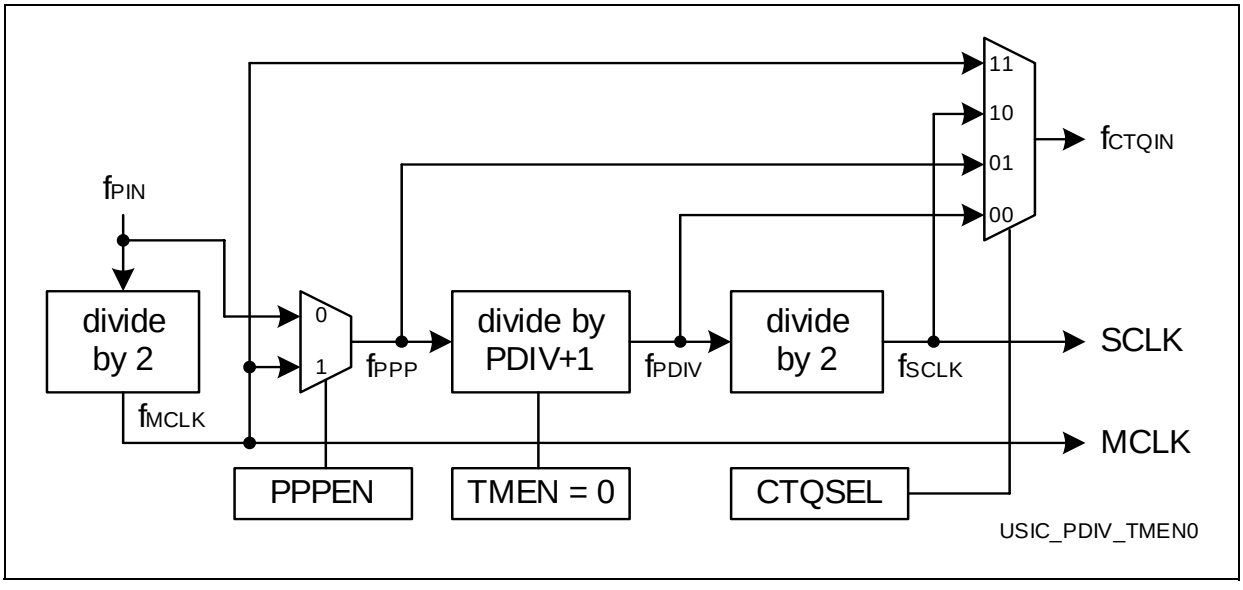


Figure 19-10 Protocol-Related Counter (Divider Mode)

19.2.7.4 Protocol-Related Counter in Capture Mode

In capture mode, the protocol-related counter stage can be used for time interval measurement (BRGL.TMEN = 1). In this case, the frequency division is disabled (reception and transmission are not possible) and the counter is working as capture timer by counting f_{PPP} periods. When reaching its maximum value, the counter stops counting. If an event is indicated by DX0T or DX1T, the actual counter value is captured into bit field BRGH.PDIV and the counter restarts from 0. Additionally, a transmit shift interrupt event is generated (bit PSRL.TSIF becomes set).

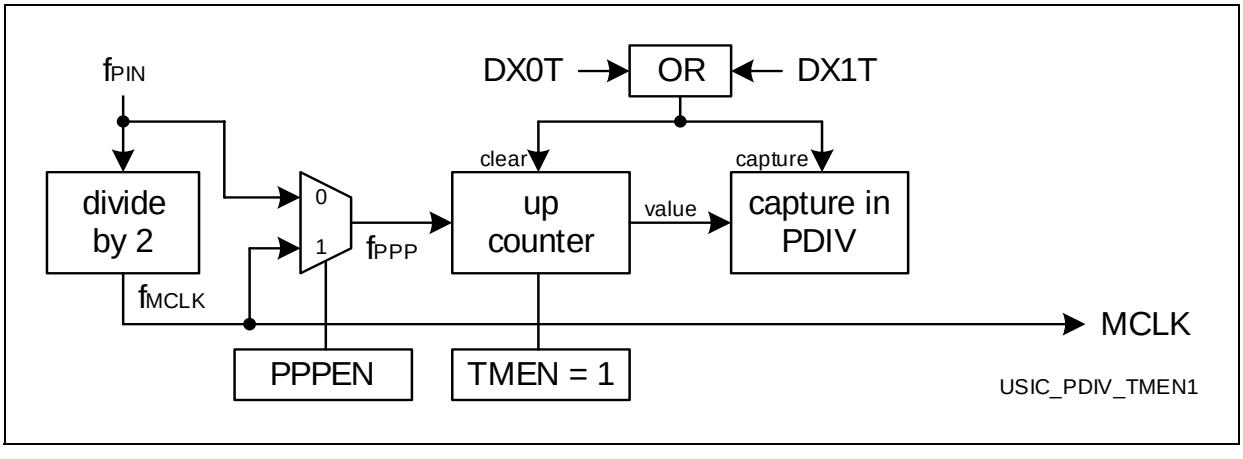


Figure 19-11 Protocol-Related Counter (Capture Mode)

The capture mode can be used to measure the baud rate in slave mode before starting data transfers, e.g. to measure the time between two edges of a data signal (by DX0T)

or of a shift clock signal (by DX1T). The conditions to activate the DXnT trigger signals can be configured in each input stage.

19.2.7.5 Time Quanta Counter

The time quanta counter CTQ associated to the protocol pre-processor allows to generate time intervals for protocol-specific purposes. The length of a time quantum tq is given by the selected input frequency f_{CTQIN} and the programmed pre-divider value. The meaning of the time quanta depend on the selected protocol, please refer to the corresponding chapters for more protocol-specific information.

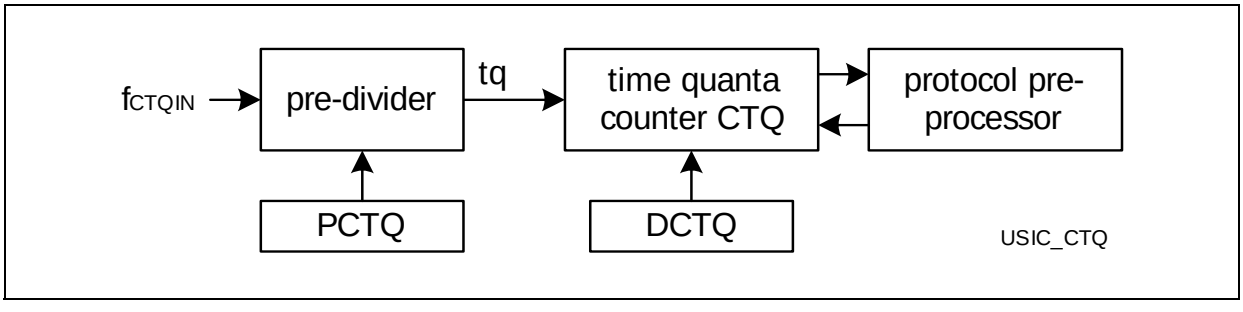


Figure 19-12 Time Quanta Counter

19.2.7.6 Shift Clock Output Configuration

The master clock output signal MCLKOUT available at the corresponding output pin can be configured in polarity. The MCLK signal can be generated for each protocol in order to provide a kind of higher frequency time base compared to the shift clock.
The configuration mechanism of the master clock output signal MCLKOUT ensures that no shortened pulses can occur. Each MCLK period consists of two phases, an active phase, followed by a passive phase. The polarity of the MCLKOUT signal during the active phase is defined by the inverted level of bit BRGH.MCLKCFG, evaluated at the start of the active phase. The polarity of the MCLKOUT signal during the passive phase is defined by bit BRGH.MCLKCFG, evaluated at the start of the passive phase. If bit BRGH.MCLKOUT is programmed with another value, the change is taken into account with the next change between the phases. This mechanism ensures that no shorter pulses than the length of a phase occur at the MCLKOUT output. In the example shown in **Figure 19-13**, the value of BRGH.MCLKCFG is changed from 0 to 1 during the passive phase of MCLK period 2.
The generation of the MCLKOUT signal is enabled/disabled by the protocol pre-processor, based on bit PCRH.MCLK. After this bit has become set, signal MCLKOUT is generated with the next active phase of the MCLK period. If PCRH.MCLK = 0 (MCLKOUT generation disabled), the level for the passive phase is also applied for active phase.

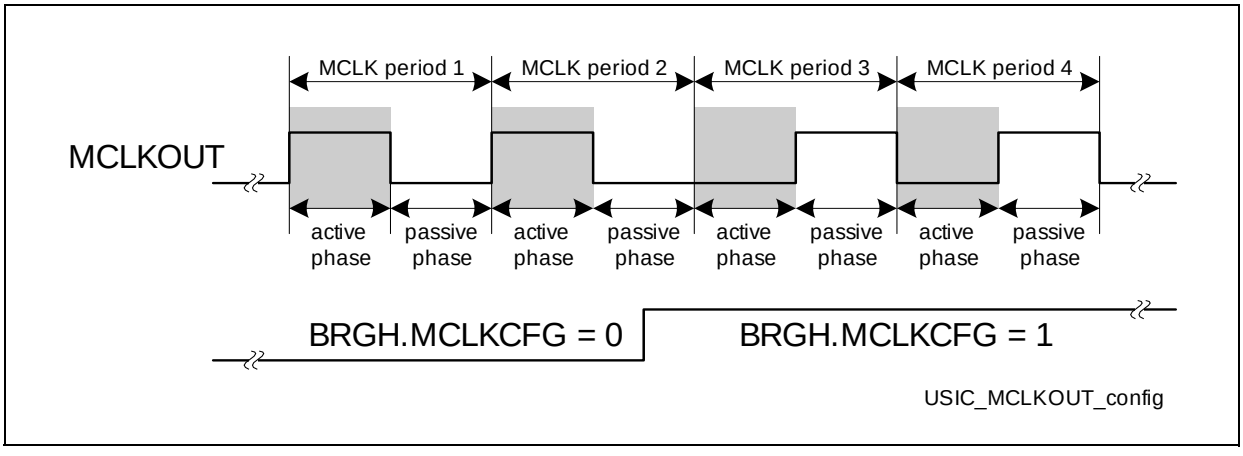


Figure 19-13 Master Clock Output Configuration

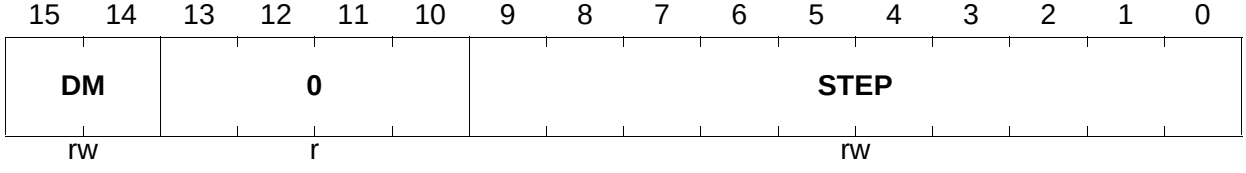
The shift clock output signal SCLKOUT available at the corresponding output pin can be configured in polarity and additionally, a delay of one period of f_{PDIV} (= half SCLK period) can be introduced. The delay allows to adapt the order of the shift clock edges to the application requirements. If the delay is used, it has to be taken into account for the calculation of the signal propagation times and loop delays.
The mechanism for the polarity control of the SCLKOUT signal is similar to the one for MCLKOUT, but based on bit field BRGH.SCLKCFG. The generation of the SCLKOUT signal is enabled/disabled by the protocol pre-processor. Depending on the selected protocol, the protocol pre-processor can control the generation of the SCLKOUT signal independently of the divider chain, e.g. for protocols without the need of a shift clock available at a pin, the SCLKOUT generation is disabled.

19.2.8 Baud Rate Generator Registers

19.2.8.1 Fractional Divider Registers

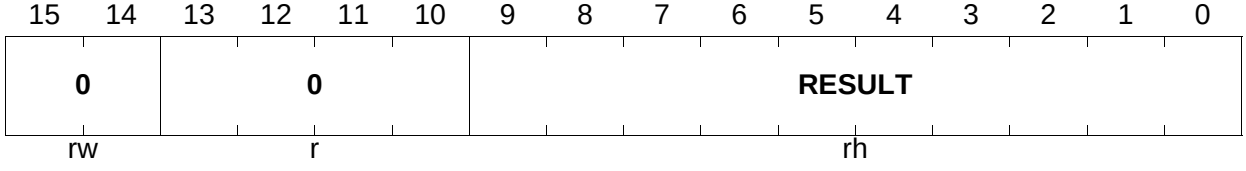
The fractional divider registers FDRL and FDRH allow the generation of the internal frequency f_{FD}, that is derived from the system clock f_{SYS}.

FDRL
Fractional Divider Register L **(04_H)** **Reset Value: 0000_H**



Field	Bits	Type	Description
STEP	[9:0]	rw	**Step Value** In normal divider mode STEP contains the reload value for RESULT after RESULT has reached $3FF_H$. In fractional divider mode STEP defines the value added to RESULT with each input clock cycle.
DM	[15:14]	rw	**Divider Mode** This bit fields defines the functionality of the fractional divider block. 00_B The divider is switched off, $f_{FD} = 0$. 01_B Normal divider mode selected. 10_B Fractional divider mode selected. 11_B The divider is switched off, $f_{FD} = 0$.
0	[13:10]	r	**Reserved** read as 0; should be written with 0.

FDRH
Fractional Divider Register H **(06_H)** **Reset Value: 0000_H**



Field	Bits	Type	Description
RESULT	[9:0]	rh	**Result Value** In normal divider mode this bit field is updated with f_{SYS} according to: RESULT = RESULT + 1 In fractional divider mode this bit field is updated with f_{SYS} according to: RESULT = RESULT + STEP If bit field DM is written with 01_B or 10_B, RESULT is loaded with a start value of $3FF_H$.
0	[15:14]	rw	**Reserved for Future Use** Must be written with 0 to allow correct fractional divider operation.
0	[13:10]	r	**Reserved** read as 0; should be written with 0.

19.2.8.2 Baud Rate Generator Registers

The protocol-related divider for baud rate generation is controlled by the registers BRGL and BRGH.

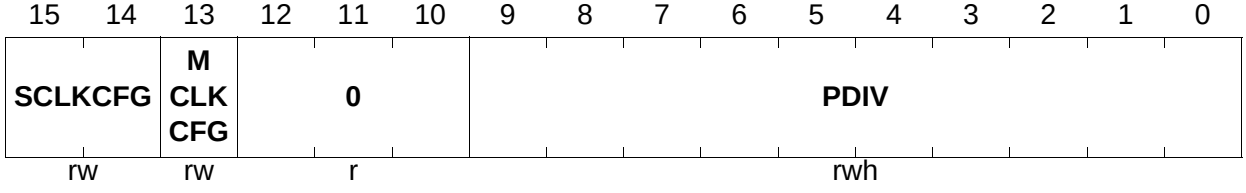
BRGL
Baud Rate Generator Register L (1C$_H$) **Reset Value: 0000$_H$**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**DCTQ**					**PCTQ**		**CTQSEL**		**0**	**PPP EN**	**TM EN**	**0**	**CLKSEL**	
r	rw					rw		rw		r	rw	rw	r	rw	

Field	Bits	Type	Description
CLKSEL	[1:0]	rw	**Clock Selection** This bit field defines the input frequency f_{PIN}. 00_B The fractional divider frequency f_{FD} is selected. 01_B reserved, no action 10_B The trigger signal DX1T defines f_{PIN}. Signal MCLK toggles with f_{PIN}. 11_B Signal MCLK corresponds to the DX1S signal and the frequency f_{PIN} is derived from the rising edges of DX1S.
TMEN	3	rw	**Timing Measurement Enable** This bit defines the functionality of the protocol-related divider. 0_B Divider mode: $f_{PDIV} = f_{PPP} / (PDIV + 1)$ Data transfers are possible and the trigger signals DX0T and DX1T are ignored. 1_B Capture mode: The 10-bit counter is incremented by 1 with f_{PPP} and stops counting when reaching its maximum value. If one of the trigger signals DX0T or DX1T become active, the counter value is captured into bit field PDIV, the counter is cleared and a transmit shift event is generated. Data transfers are not possible.

Field	Bits	Type	Description
PPPEN	4	rw	**Enable 2:1 Divider for f_{PPP}** This bit defines the input frequency f_{PPP}. 0_B The 2:1 divider for f_{PPP} is disabled. $f_{PPP} = f_{PIN}$ 1_B The 2:1 divider for f_{PPP} is enabled. $f_{PPP} = f_{MCLK} = f_{PIN}$ / 2.
CTQSEL	[7:6]	rw	**Input Selection for CTQ** This bit defines the length of a time quantum for the protocol pre-processor. 00_B $f_{CTQIN} = f_{PDIV}$ 01_B $f_{CTQIN} = f_{PPP}$ 10_B $f_{CTQIN} = f_{SCLK}$ 11_B $f_{CTQIN} = f_{MCLK}$
PCTQ	[9:8]	rw	**Pre-Divider for Time Quanta Counter** This bit field defines length of a time quantum tq for the time quanta counter in the protocol pre-processor. $tq = (PCTQ + 1) / f_{CTQIN}$
DCTQ	[14:10]	rw	**Denominator for Time Quanta Counter** This bit field defines the number of time quanta tq taken into account by the time quanta counter in the protocol pre-processor.
0	2, 5, 15	r	**Reserved** read as 0; should be written with 0.

BRGH
Baud Rate Generator Register H (1E$_H$) **Reset Value: 0000$_H$**



Field	Bits	Type	Description
PDIV	[9:0]	rwh	**Divider Mode: Divider Factor to Generate f_{PDIV}** This bit field defines the ratio between the input frequency f_{PPP} and the divider frequency f_{PDIV}. **Capture Mode: Captured Time Interval** The value of the counter is captured into this bit field if one of the trigger signals DX0T or DX1T are activated by the corresponding input stage.
MCLKCFG	13	rw	**Master Clock Configuration** This bit field defines the level of the passive phase of the MCLKOUT signal. 0_B The passive level is 0. 1_B The passive level is 1.
SCLKCFG	[15:14]	rw	**Shift Clock Output Configuration** This bit field defines the level of the passive phase of the SCLKOUT signal and enables/disables a delay of half of a SCLK period. 00_B The passive level is 0 and the delay is disabled. 01_B The passive level is 1 and the delay is disabled. 10_B The passive level is 0 and the delay is enabled. 11_B The passive level is 1 and the delay is enabled.
0	[12:10]	r	**Reserved** read as 0; should be written with 0.

19.2.9 Operating the Transmit Data Path

The transmit data path is based on a 16-bit wide transmit shift register TSR and a transmit buffer TBUF. The data transfer parameters like data word length, data frame length, or the shift direction are controlled commonly for transmission and reception by the shift control registers. Register TCSRL mainly controls the transmit data handling, whereas register TCSRH monitors the transmit status.
A change of the value of the data shift output signal DOUT only happens at the corresponding edge of the shift clock input signal. The level of the last data bit of a data word/frame is held constant at DOUT until the next data word begins with the next corresponding edge of the shift clock.

19.2.9.1 Transmit Buffering

The transmit shift register TSR can not be directly accessed by SW, because it is automatically updated with the value stored in the transmit buffer TBUF if a currently transmitted data word is finished and new data is valid for transmission. Data words can be loaded directly into TBUF by writing to one of the transmit buffer input locations TBUFx (see **Chapter 19.2.9.2**) or, optionally, by a FIFO buffer stage (see **Chapter 19.2.13**).

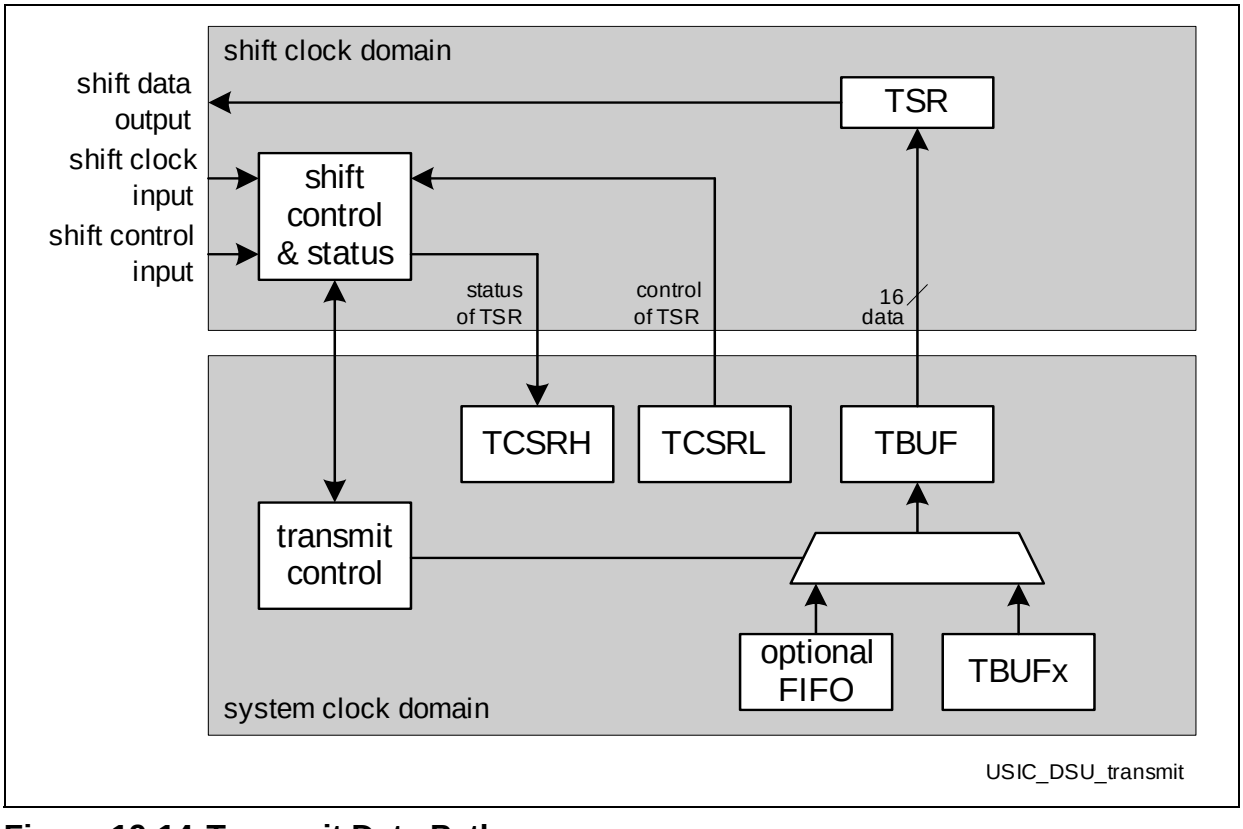


Figure 19-14 Transmit Data Path

19.2.9.2 Transmit Control Information

The transmit control information TCI can be used as additional control parameter for data transfers. The TCI is derived from the address x of the written TBUFx transmit buffer input location.
It can be used to dynamically change the data word length, the data frame length, or other protocol-specific functions (for more details about this topic, please refer to the corresponding protocol chapters). The way how the TCI is used in different applications can be programmed by bits WLEMD, FLEMD, SELMD, and WAMD in register TCSRL. Please note that not all possible settings lead to useful system behavior.

- Word length control:
  If TCSRL.WLEMD = 1, bit field SCTRH.WLE is updated with TCI[3:0] if a transmit buffer input location TBUFx is written. This function can be used in all protocols to dynamically change the data word length between 1 and 16 data bits per data word. Additionally, bit TCSRL.EOF is updated with TCI[4]. This function can be used in SSC master mode to control the slave select generation to finish data frames. It is recommended to program TCSRL.FLEMD = TCSRL.SELMD = 0.
- Frame length control:
  If TCSRL.FLEMD = 1, bit field SCTRH.FLE[4:0] is updated with TCI[4:0] and SCTRH.FLE[5] becomes 0 if a transmit buffer input location TBUFx is written. This function can be used in all protocols to dynamically change the data frame length between 1 and 32 data bits per data frame. It is recommended to program TCSRL.SELMD = TCSRL.WLEMD = TCSRL.WAMD = 0.
- Select output control:
  If TCSRL.SELMD = 1, bit field PCR.CTR[20:16] is updated with TCI[4:0] and PCR.CTR[23:21] becomes 0 if a transmit buffer input location TBUFx is written. This function can be used in SSC master mode to define the targeted slave device(s). It is recommended to program TCSRL.WLEMD = TCSRL.FLEMD = TCSRL.WAMD = 0.
- Word address control:
  If TCSRL.WAMD = 1, bit TCSRL.WA is updated with TCI[4] if a transmit buffer input location TBUFx is written. This function can be used in IIS mode to define if the data word is transmitted on the right or the left channel. It is recommended to program TCSRL.SELMD = TCSRL.FLEMD = 0.

19.2.9.3 Transmit Data Validation

The data word in the transmit buffer TBUF can be tagged valid or invalid for transmission by bit TCSRL.TDV (transmit data valid). A combination of data flow related and event related criteria define whether the data word is considered valid for transmission. A data validation logic checks the start conditions for each data word. Depending on the result of the check, the transmit shift register is loaded with different values, according to the following rules:

- If a USIC channel is the communication master (it defines the start of each data word transfer), a data word transfer can only be started with valid data in the transmit buffer TBUF. In this case, the transmit shift register is loaded with the content of TBUF, that is not changed due to this action.
- If a USIC channel is a communication slave (it can not define the start itself, but has to react), a data word transfer requested by the communication master has to be started independently of the status of the data word in TBUF. If a data word transfer is requested and started by the master, the transmit shift register is loaded at the first corresponding shift clock edge either with the data word in TBUF (if it is valid for transmission) or with the level defined by bit SCTRL.PDL (if the contents of TBUF has not been valid at the transmission start). In both cases, the contents of TBUF is not changed.

The control and status bits for the data validation are located in registers **TCSRL** or **TCSRH**. The data validation is based on the logic blocks shown in **Figure 19-15**.

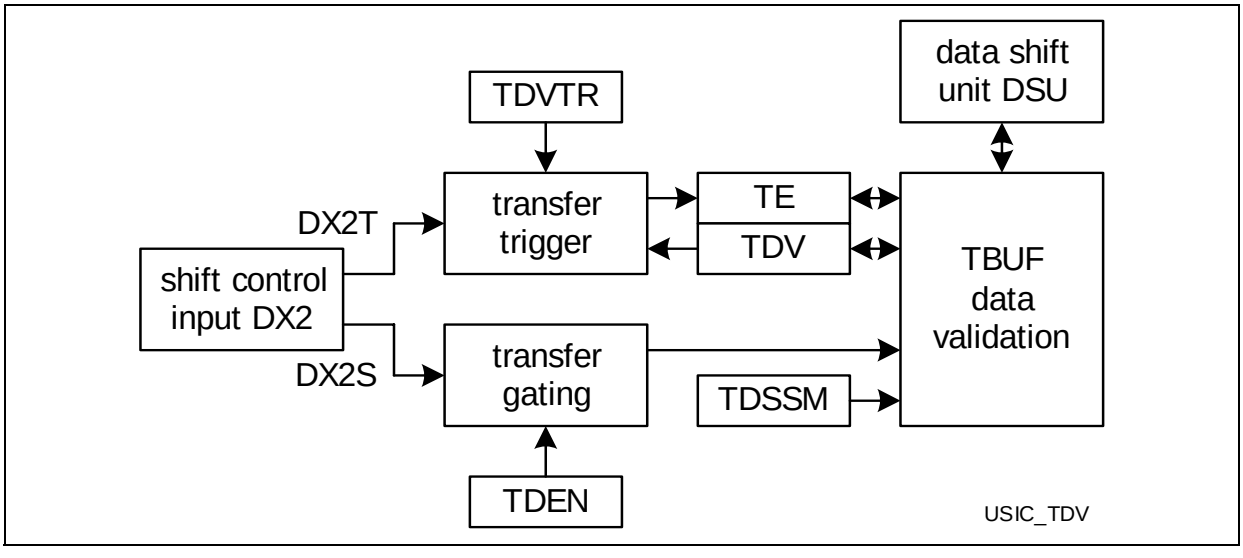


Figure 19-15 Transmit Data Validation

- A transfer gating logic enables or disables the data word transfer from TBUF under SW or under HW control. If the input stage DX2 is not needed for data shifting, signal DX2S can be used for gating purposes. The transfer gating logic is controlled by bit field TCSRL.TDEN.

- A transfer trigger logic supports data word transfers related to events, e.g. timer based or related to an input pin. If the input stage DX2 is not needed for data shifting, signal DX2T can be used for trigger purposes. The transfer trigger logic is controlled by bit TCSRL.TDVTR and the occurrence of a trigger event is indicated by bit TCSRH.TE.
- A data validation logic combining the inputs from the gating logic, the triggering logic and DSU signals. A transmission of the data word located in TBUF can only be started if the gating enables the start, bit TCSRL.TDV = 1, and bit TCSRH.TE = 1.
  The contents of the transmit buffer TBUF should not be overwritten with new data while it is valid for transmission and a new transmission can start. If the contents of TBUF has to be changed, it is recommended to clear bit TCSRL.TDV by writing FMR.MDTV = 10_B before updating the data. Bit TCSRL.TDV becomes automatically set when TBUF is updated with new data. Another possibility are the interrupts TBI (for ASC and IIC) or RSI (for SSC and IIS) indicating that a transmission has started. While a transmission is in progress, TBUF can be loaded with new data. In this case the user has to take care that an update of the TBUF contents takes place before a new transmission starts.

With this structure, the following data transfer functionality can be achieved:

- If bit TCSRL.TDSSM = 0, the contents of the transmit buffer TBUF is always considered as valid for transmission. The transfer trigger mechanism can be used to start the transfer of the same data word based on the selected event (e.g. on a timer base or an edge at a pin) to realize a kind of life-sign mechanism. Furthermore, in slave mode, it is ensured that always a correct data word is transmitted instead of the passive data level.
- Bit TCSRL.TDSSM = 1 has to be programmed to allow word-by-word data transmission with a kind of single-shot mechanism. After each transmission start, a new data word has to be loaded into the transmit buffer TBUF, either by SW write actions to one of the transmit buffer input locations TBUFx or by an optional data buffer (e.g. FIFO buffer). To avoid that data words are sent out several times or to allow data handling with an additional data buffer (e.g. FIFO), bit TCSRL.TDSSM has to be 1.
- Bit TCSRL.TDV becoming automatically set when a new data word is loaded into the transmit buffer TBUF, a transmission start can be requested by a write action of the data to be transmitted to at least the low byte of one of the transmit buffer input locations TBUFx. The additional information TCI can be used to control the data word length or other parameters independently for each data word by a single write access.
- Bit field FMR.MTDV allows SW driven modification (set or clear) of bit TCSRL.TDV. Together with the gating control bit field TCSRL.TDEN, the user can set up the transmit data word without starting the transmission. A possible program sequence could be: clear TCSRL.TDEN = 00_B, write data to TBUFx, clear TCSRL.TDV by

writing FMR.MDTV = 10_B, re-enable the gating with TCSRL.TDEN = 01_B and then set TCSRL.TDV under SW control by writing FMR.MTDV = 01_B.

19.2.10 Operating the Receive Data Path

The receive data path is based on two 16-bit wide receive shift registers RSR0 and RSR1 and a receive buffer for each of them (RBUF0 and RBUF1). The data transfer parameters like data word length, data frame length, or the shift direction are controlled commonly for transmission and reception by the shift control registers.
Register RBUF01SRL monitors the status of RBUF0 and register RBUF01SRH of RBUF1.

19.2.10.1 Receive Buffering

The receive shift registers can not be directly accessed by SW, but their contents are automatically loaded into the receive buffer registers RBUF0 (or RBUF1 respectively) if a complete data word has been received or the frame is finished. The received data words in RBUF0 or RBUF1 can be read out in the correct order directly from register RBUF or, optionally, from a FIFO buffer stage (see **Chapter 19.2.13**).

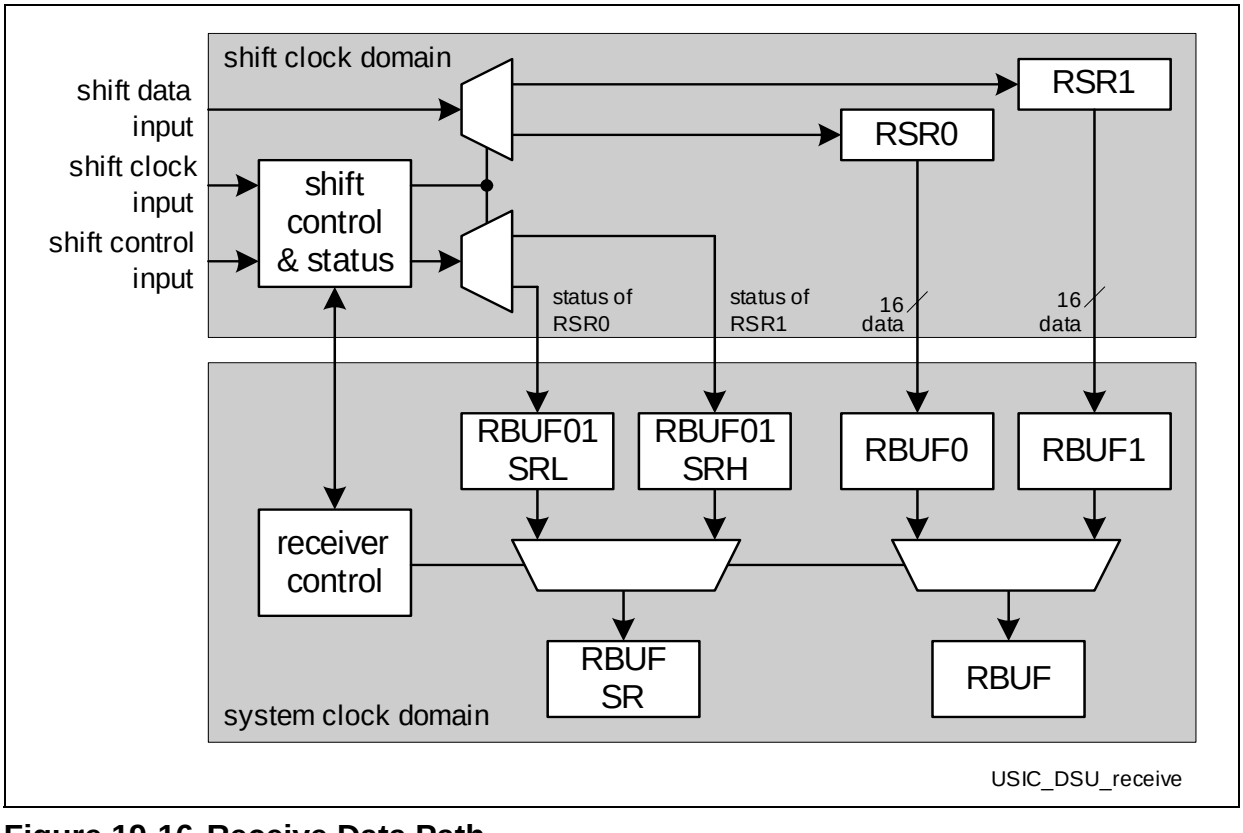


Figure 19-16 Receive Data Path

19.2.10.2 Baud Rate Constraints

The following baud rate constraints have to be respected to ensure correct data reception and buffering. The user has to take care about these restrictions when selecting the baud rate and the data word length with respect to the module clock frequency f_{SYS}.

- A received data word in a receiver shift register RSRx must be held constant for at least 4 periods of f_{SYS} in order to ensure correct loading of the related receiver buffer register RBUFx.
- The shift control signal has to be constant inactive for at least 5 periods of f_{SYS} between two consecutive frames in order to correctly detect the end of a frame.
- The shift control signal has to be constant active for at least 1 period of f_{SYS} in order to correctly detect a frame (shortest frame).
- A minimum setup and hold time of the shift control signal with respect to the shift clock signal of 10ns has to be ensured.

19.2.11 Transfer Control and Status Registers

19.2.11.1 Shift Control Registers

The data shift unit is controlled by the following registers. The values in these registers are applied for data transmission and reception.

Please note that the shift control settings SDIR, WLE, and FLE are shared between transmitter and receiver. They are internally “frozen” for a each data word transfer in the transmitter with the first transmit shift clock edge and with the first receive shift clock edge in the receiver. The SW has to take care that updates of these bit fields by SW are done coherently (e.g. refer to the receiver start event indication PSR.RSIF).

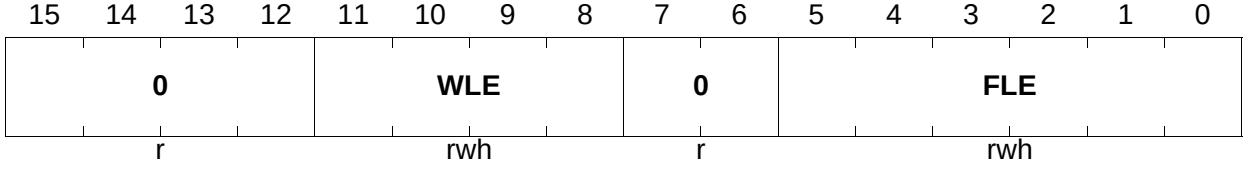
SCTRL
Shift Control Register L **(30_H)** **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0						TRM		DOCFG		0				PDL	SDIR
r						rw		rw		r				rw	rw

Field	Bits	Type	Description
SDIR	0	rw	**Shift Direction** This bit defines the shift direction of the data words for transmission and reception. 0_B Shift LSB first. The first data bit of a data word is located at bit position 0. 1_B Shift MSB first. The first data bit of a data word is located at the bit position given by bit field SCTRH.WLE.
PDL	1	rw	**Passive Data Level** This bit defines the output level at the shift data output signal when no data is available for transmission. The PDL level is output with the first relevant transmit shift clock edge of a data word. 0_B The passive data level is 0. 1_B The passive data level is 1.
DOCFG	[7:6]	rw	**Data Output Configuration** This bit defines the relation between the internal shift data value and the data output signal DOUT. $X0_B$ DOUT = shift data value $X1_B$ DOUT = inverted shift data value

Field	Bits	Type	Description
TRM	[9:8]	rw	**Transmission Mode** This bit field describes how the shift control signal is interpreted by the DSU. Data transfers are only possible while the shift control signal is active. 00_B The shift control signal is considered as inactive and data frame transfers are not possible. 01_B The shift control signal is considered active if it is at 1-level. This is the setting to be programmed to allow data transfers. 10_B The shift control signal is considered active if it is at 0-level. It is recommended to avoid this setting and to use the inversion in the DX2 stage in case of a low-active signal. 11_B The shift control signal is considered active without referring to the actual signal level. Data frame transfer is possible after each edge of the signal.
0	[5:2], [15:10]	r	**Reserved** read as 0; should be written with 0.

SCTRH
Shift Control Register H **(32_H)** **Reset Value: 0000_H**



Field	Bits	Type	Description
FLE	[5:0]	rwh	**Frame Length** This bit field defines how many bits are transferred within a data frame. A data frame can consist of several concatenated data words. If TCSRL.FLEMD = 1, the value can be updated automatically by the data handler.
WLE	[11:8]	rwh	**Word Length** This bit field defines the data word length (amount of bits that are transferred in each data word) for reception and transmission. The data word is always right-aligned in the data buffer at the bit positions [WLE down to 0]. If TCSRL.WLEMD = 1, the value can be updated automatically by the data handler. 00_H The data word contains 1 data bit located at bit position 0. 01_H The data word contains 2 data bits located at bit positions [1:0]. ... $0F_H$ The data word contains 16 data bits located at bit positions [15:0].
0	[7:6], [15:12]	r	**Reserved** read as 0; should be written with 0.

19.2.11.2 Transmission Control and Status Registers

The data transmission is controlled by register TCSRL.

TCSRL
Transmit Control/Status Register L ($3C_H$) Reset Value: 0000_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		WA	TD VTR	TDEN		0	TD SSM	TDV	EOF	SOF	0	WA MD	FLE MD	SEL MD	WLE MD
r		rwh	rw	rw		r	rw	rh	rwh	rwh	r	rw	rw	rw	rw

Field	Bits	Type	Description
WLEMD	0	rw	**WLE Mode** This bit enables the data handler to automatically update the bit field SCTRH.WLE by the transmit control information TCI[3:0] and bit TCSR.EOF by TCI[4] (see **Section 19.2.9.2**). If enabled, an automatic update takes place when new data is loaded to register TBUF, either by writing to one of the transmit buffer input locations TBUFx or by an optional data buffer. 0_B The automatic update of SCTRH.WLE and TCSR.EOF is disabled. 1_B The automatic update of SCTR.WLE and TCSR.EOF is enabled.
SELMD	1	rw	**Select Mode** This bit can be used mainly for the SSC protocol. It enables the data handler to automatically update bit field PCRH.CTR[20:16] by the transmit control information TCI[4:0] and clear bit field PCRH.CTR[23:21] (see **Section 19.2.9.2**). If enabled, an automatic update takes place when new data is loaded to register TBUF, either by writing to one of the transmit buffer input locations TBUFx or by an optional data buffer. 0_B The automatic update of PCRH.CTR[23:16] is disabled. 1_B The automatic update of PCRH.CTR[23:16] is disabled.

Field	Bits	Type	Description
FLEMD	2	rw	**FLE Mode** This bit enables the data handler to automatically update bits SCTRH.FLE[4:0] by the transmit control information TCI[4:0] and to clear bit SCTRH.FLE[5] (see **Section 19.2.9.2**). If enabled, an automatic update takes place when new data is loaded to register TBUF, either by writing to one of the transmit buffer input locations TBUFx or by an optional data buffer. 0_B The automatic update of FLE is disabled. 1_B The automatic update of FLE is enabled.
WAMD	3	rw	**WA Mode** This bit can be used mainly for the IIS protocol. It enables the data handler to automatically update bit TCSRL.WA by the transmit control information TCI[4] (see **Section 19.2.9.2**). If enabled, an automatic update takes place when new data is loaded to register TBUF, either by writing to one of the transmit buffer input locations TBUFx or by an optional data buffer. 0_B The automatic update of bit WA is disabled. 1_B The automatic update of bit WA is enabled.
SOF	5	rw	**Start Of Frame** This bit is only taken into account for the SSC protocol, otherwise it is ignored. It indicates that the data word in TBUF is considered as the first word of a new SSC frame if it is valid for transmission (TCSRL.TDV = 1). This bit becomes cleared when the TBUF data word is transferred to the transmit shift register. 0_B The data word in TBUF is not considered as first word of a frame. 1_B The data word in TBUF is considered as first word of a frame. A currently running frame is finished and MSLS becomes deactivated (respecting the programmed delays).

Field	Bits	Type	Description
EOF	6	rwh	**End Of Frame** This bit is only taken into account for the SSC protocol, otherwise it is ignored. It can be modified automatically by the data handler if bit WLEMD = 1. It indicates that the data word in TBUF is considered as the last word of an SSC frame. If it is the last word, the MSLS signal becomes inactive after the transfer, respecting the programmed delays. This bit becomes cleared when the TBUF data word is transferred to the transmit shift register. 0_B The data word in TBUF is not considered as last word of an SSC frame. 1_B The data word in TBUF is considered as last word of an SSC frame.
TDV	7	rh	**Transmit Data Valid** This bit indicates that the data word in the transmit buffer TBUF can be considered as valid for transmission. The TBUF data word can only be sent out if TDV = 1. It is automatically set when data is moved to TBUF (by writing to one of the transmit buffer input locations TBUFx, or optionally, by the bypass or FIFO mechanism). 0_B The data word in TBUF is not valid for transmission. 1_B The data word in TBUF is valid for transmission and a transmission start is possible. New data should not be written to a TBUFx input location while TDV = 1.

Field	Bits	Type	Description
TDSSM	8	rw	**TBUF Data Single Shot Mode** This bit defines if the data word TBUF data is considered as permanently valid or if the data should only be transferred once. 0_B The data word in TBUF is not considered as invalid after it has been loaded into the transmit shift register. The loading of the TBUF data into the shift register does not clear TDV. 1_B The data word in TBUF is considered as invalid after it has been loaded into the shift register. In ASC and IIC mode, TDV is cleared with the TBI event, whereas in SSC and IIS mode, it is cleared with the RSI event. TDSSM = 1 has to be programmed if an optional data buffer is used.
TDEN	[11:10]	rw	**TBUF Data Enable** This bit field controls the gating of the transmission start of the data word in the transmit buffer TBUF. 00_B A transmission start of the data word in TBUF is disabled. If a transmission is started, the passive data level is sent out. 01_B A transmission of the data word in TBUF can be started if TDV = 1. 10_B A transmission of the data word in TBUF can be started if TDV = 1 while DX2S = 0. 11_B A transmission of the data word in TBUF can be started if TDV = 1 while DX2S = 1.
TDVTR	12	rw	**TBUF Data Valid Trigger** This bit enables the transfer trigger unit to set bit TCSRH.TE if the trigger signal DX2T becomes active for event driven transfer starts, e.g. timer-based or depending on an event at an input pin. Bit TDVTR has to be 0 for protocols where the input stage DX2 is used for data shifting. 0_B Bit TCSRH.TE is permanently set. 1_B Bit TCSRH.TE is set if DX2T becomes active while TDV = 1.

Field	Bits	Type	Description
WA	13	rwh	**Word Address** This bit is only taken into account for the IIS protocol, otherwise it is ignored. It can be modified automatically by the data handler if bit WAMD = 1. Bit WA defines for which channel the data stored in TBUF will be transmitted. 0_B The data word in TBUF will be transmitted after a falling edge of WA has been detected (referring to PSR.WA). 1_B The data word in TBUF will be transmitted after a rising edge of WA has been detected (referring to PSR.WA).
0	4, 9, [15:14]	r	**Reserved** read as 0; should be written with 0.

The data transmission status is monitored by register TCSRH.

TCSRH
Transmit Control/Status Register H ($3E_H$) **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0			TE	TVC	TV	0	TSOF	0							
r			rh	rh	rh	r	rh	r							

Field	Bits	Type	Description
TSOF	8	rh	**Transmitted Start Of Frame** This bit indicates if the latest start of a data word transmission has taken place for the first data word of a new data frame. This bit is updated with the transmission start of each data word. 0_B The latest data word transmission has not been started for the first word of a data frame. 1_B The latest data word transmission has been started for the first word of a data frame.
TV	10	rh	**Transmission Valid** This bit represents the transmit buffer underflow and indicates if the latest start of a data word transmission has taken place with a valid data word from the transmit buffer TBUF. This bit is updated with the transmission start of each data word. 0_B The latest start of a data word transmission has taken place while no valid data was available. As a result, the transmission of a data words with passive level (SCTRL.PDL) has been started. 1_B The latest start of a data word transmission has taken place with valid data from TBUF.
TVC	11	rh	**Transmission Valid Cumulated** This bit cumulates the transmit buffer underflow indication TV. It is cleared automatically together with bit TV and has to be set by writing FMRL.ATVC = 1. 0_B Since TVC has been set, at least one data buffer underflow condition has occurred. 1_B Since TVC has been set, no data buffer underflow condition has occurred.

Field	Bits	Type	Description
TE	12	rh	**Trigger Event** If the transfer trigger mechanism is enabled, this bit indicates that a trigger event has been detected (DX2T = 1) while TCSRL.TDV = 1. If the event trigger mechanism is disabled, the bit TE is permanently set. It is cleared by writing FMRL.MTDV = 10_B or when the data word located in TBUF is loaded into the shift register. 0_B The trigger event has not yet been detected. A transmission of the data word in TBUF can not be started. 1_B The trigger event has been detected (or the trigger mechanism is switched off) and a transmission of the data word in TBUF can not be started.
0	[7:0], 9, [15:13]	r	**Reserved** read as 0; should be written with 0.

19.2.11.3 Flag Modification Registers

The flag modification registers FMRL, FMRH allow the modification of control and status flags related to data handling by using only write accesses. Read accesses to FMRL, FMRH always deliver 0 at all bit positions.

Additionally, the service request outputs of this USIC channel can be activated by SW (the activation is triggered by the write access and is deactivated automatically).

FMRL
Flag Modification Register L **(38_H)** **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRDV1	**CRDV0**	**0**									**ATVC**	**0**		**MTDV**	
w	w	r									w	r		w	

Field	Bits	Type	Description
MTDV	[1:0]	w	**Modify Transmit Data Valid** Writing to this bit field can modify bits TCSRL.TDV and TCSRH.TE to control the start of a data word transmission by SW. 00_B No action. 01_B Bit TDV is set, TE is unchanged. 10_B Bits TDV and TE are cleared. 11_B reserved
ATVC	4	w	**Activate Bit TVC** Writing to this bit can set bit TCSRH.TVC to start a new cumulation of the transmit buffer underflow condition. 0_B No action. 1_B Bit TCSRH.TVC is set.
CRDV0	14	w	**Clear Bits RDV for RBUF0** Writing 1 to this bit clears bits RBUF01SRL.RDV00 and RBUF01SRH.RDV10 to declare the received data in RBUF0 as no longer valid (to emulate a read action). 0_B No action. 1_B Bits RBUF01SRL.RDV00 and RBUF01SRH.RDV10 are cleared.

Field	Bits	Type	Description
CRDV1	15	w	**Clear Bit RDV for RBUF1** Writing 1 to this bit clears bits RBUF01SRL.RDV01 and RBUF01SRH.RDV11 to declare the received data in RBUF1 as no longer valid (to emulate a read action). 0_B No action. 1_B Bits RBUF01SRL.RDV01 and RBUF01SRH.RDV11 are cleared.
0	[3:2], [13:5]	r	**Reserved** read as 0; should be written with 0.

FMRH
Flag Modification Register H **($3A_H$)** **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0												**SIO3**	**SIO2**	**SIO1**	**SIO0**
r												w	w	w	w

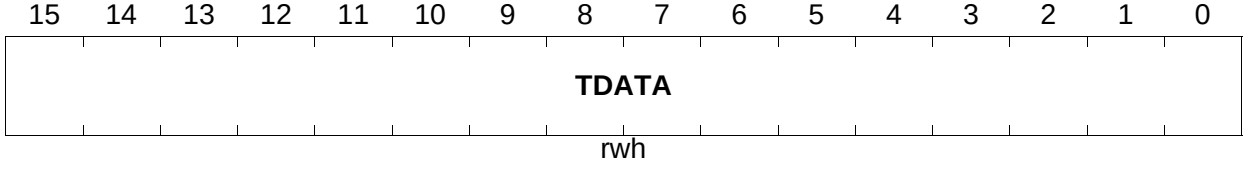
Field	Bits	Type	Description
SIO0, **SIO1,** **SIO2,** **SIO3**	0, 1, 2, 3	w	**Set Interrupt Output SRx** Writing a 1 to this bit field activates the service request output SRx of this USIC channel. It has no impact on service request outputs of other USIC channels. 0_B No action. 1_B The service request output SRx is activated.
0	[15:4]	r	**Reserved** returns 0 if read; should be written with 0;

19.2.12 Data Buffer Registers

19.2.12.1 Transmit Buffer Locations

The 32 independent data input locations TBUF00 to TBUF31 are address locations that can be used as data entry locations for the transmit buffer. Data written to one of these locations will appear in a common register TBUF. Additionally, the 5 bit coding of the number [31:0] of the addressed data input location represents the transmit control information TCI (please refer to the protocol sections for more details).
The internal transmit buffer register TBUF contains the data that will be loaded to the transmit shift register for the next transmission of a data word. It can be read out at all TBUF00 to TBUF31 addresses.

TBUFx (x = 00-31)
Transmit Buffer Input Location x **(80_H + x*4)** **Reset Value: 0000_H**

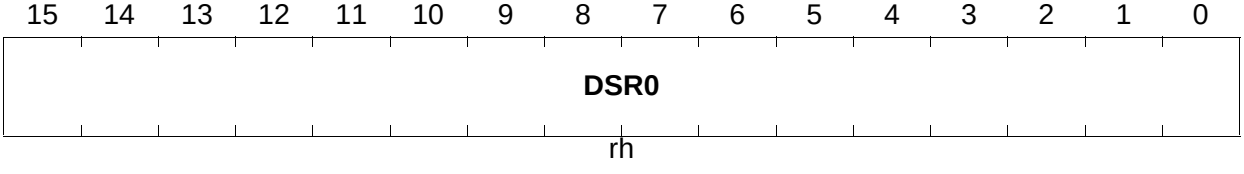


Field	Bits	Type	Description
TDATA	[15:0]	rwh	**Transmit Data** This bit field contains the data to be transmitted (read view). A data write action to at least the low byte of TDATA sets TCSRL.TDV.

19.2.12.2 Receive Buffer Registers RBUF0, RBUF1

The receive buffer register RBUF0 contains the data received from RSR0. A read action does not change the status of the receive data from “not yet read = valid” to “already read = not valid”.

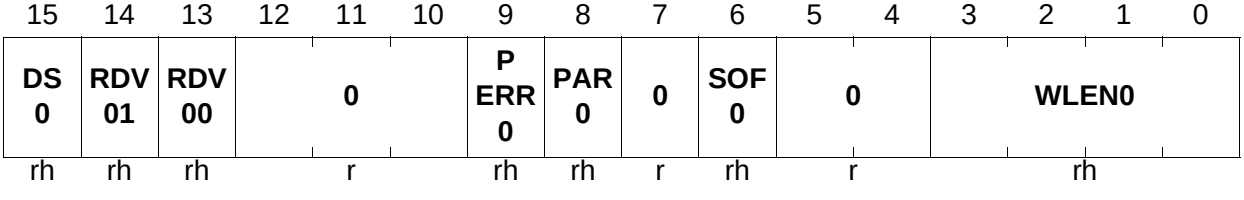
RBUF0
Receiver Buffer Register 0 (50_H) Reset Value: 0000_H



Field	Bits	Type	Description
DSR0	[15:0]	rh	**Data of Shift Register 0**

The receive buffer status register RBUF01SRL provides the status of the data in receive buffer RBUF0.

RBUF01SRL
Receiver Buffer 01 Status Register L (60_H) Reset Value: 0000_H



Field	Bits	Type	Description
WLEN0	[3:0]	rh	**Received Data Word Length in RBUF0** This bit field indicates how many bits have been received within the last data word stored in RBUF0. This number indicates how many data bits have to be considered as receive data, whereas the other bits in RBUF0 have been cleared automatically. The received bits are always right-aligned. 00_H One bit has been received. ... $0F_H$ Sixteen bits have been received.

Field	Bits	Type	Description
SOF0	6	rh	**Start of Frame in RBUF0** This bit indicates whether the data word in RBUF0 has been the first data word of a data frame. 0_B The data in RBUF0 has not been the first data word of a data frame. 1_B The data in RBUF0 has been the first data word of a data frame.
PAR0	8	rh	**Protocol-Related Argument in RBUF0** This bit indicates the value of the protocol-related argument. This value is elaborated depending on the selected protocol and adds additional information to the data word in RBUF0. The meaning of this bit is described in the corresponding protocol chapter.
PERR0	9	rh	**Protocol-related Error in RBUF0** This bit indicates if the value of the protocol-related argument meets an expected value. This value is elaborated depending on the selected protocol and adds additional information to the data word in RBUF0. The meaning of this bit is described in the corresponding protocol chapter. 0_B The received protocol-related argument PAR matches the expected value. The reception of the data word sets bit PSR.RIF and can generate a receive interrupt. 1_B The received protocol-related argument PAR does not match the expected value. The reception of the data word sets bit PSR.AIF and can generate an alternative receive interrupt.

Field	Bits	Type	Description
RDV00	13	rh	**Receive Data Valid in RBUF0** This bit indicates the status of the data content of register RBUF0. This bit is identical to bit RBUF01SRH.RDV10 and allows consisting reading of information for the receive buffer registers. It is set when a new data word is stored in RBUF0 and automatically cleared if it is read out via RBUF. 0_B Register RBUF0 does not contain data that has not yet been read out. 1_B Register RBUF0 contains data that has not yet been read out.
RDV01	14	rh	**Receive Data Valid in RBUF1** This bit indicates the status of the data content of register RBUF1. This bit is identical to bit RBUF01SRH.RDV11 and allows consisting reading of information for the receive buffer registers. It is set when a new data word is stored in RBUF1 and automatically cleared if it is read out via RBUF. 0_B Register RBUF1 does not contain data that has not yet been read out. 1_B Register RBUF1 contains data that has not yet been read out.
DS0	15	rh	**Data Source** This bit indicates which receive buffer register (RBUF0 or RBUF1) is currently visible in registers RBUF(D) and in RBUFSR for the associated status information. It indicates which buffer contains the oldest data (the data that has been received first). This bit is identical to bit RBUF01SRH.DS1 and allows consisting reading of information for the receive buffer registers. 0_B The register RBUF contains the data of RBUF0 (same for associated status information). 1_B The register RBUF contains the data of RBUF1 (same for associated status information).
0	[5:4], 7, [12:10]	r	**Reserved** read as 0; should be written with 0.

The receive buffer register RBUF1 contains the data received from RSR1. A read action does not change the status of the receive data from “not yet read = valid” to “already read = not valid”.

RBUF1
Receiver Buffer Register 1 (54_H) Reset Value: 0000_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DSR1															
rh															

Field	Bits	Type	Description
DSR1	[15:0]	rh	**Data of Shift Register 1**

The receive buffer status register RBUF01SRH provides the status of the data in receive buffer RBUF1.

RBUF01SRH
Receiver Buffer 01 Status Register H (62_H) Reset Value: 0000_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DS1	RDV11	RDV10	0			PERR1	PAR1	0	SOF1	0		WLEN1			
rh	rh	rh	r			rh	rh	r	rh	r		rh			

Field	Bits	Type	Description
WLEN1	[3:0]	rh	**Received Data Word Length in RBUF1** This bit field indicates how many bits have been received within the last data word stored in RBUF1. This number indicates how many data bits have to be considered as receive data, whereas the other bits in RBUF1 have been cleared automatically. The received bits are always right-aligned. 00_H One bit has been received. ... $0F_H$ Sixteen bits have been received.

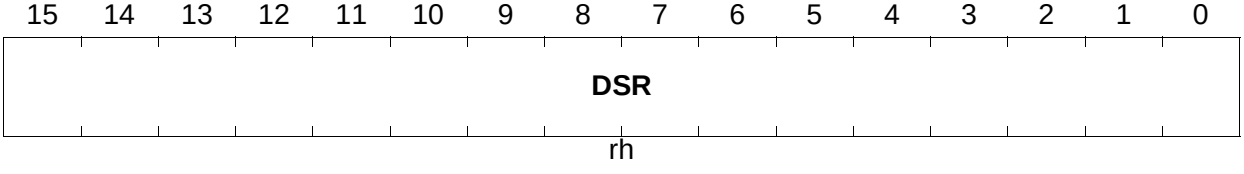
Field	Bits	Type	Description
SOF1	6	rh	**Start of Frame in RBUF1** This bit indicates whether the data word in RBUF1 has been the first data word of a data frame. 0_B The data in RBUF1 has not been the first data word of a data frame. 1_B The data in RBUF1 has been the first data word of a data frame.
PAR1	8	rh	**Protocol-Related Argument in RBUF1** This bit indicates the value of the protocol-related argument. This value is elaborated depending on the selected protocol and adds additional information to the data word in RBUF1. The meaning of this bit is described in the corresponding protocol chapter.
PERR1	9	rh	**Protocol-related Error in RBUF1** This bit indicates if the value of the protocol-related argument meets an expected value. This value is elaborated depending on the selected protocol and adds additional information to the data word in RBUF1. The meaning of this bit is described in the corresponding protocol chapter. 0_B The received protocol-related argument PAR matches the expected value. The reception of the data word sets bit PSR.RIF and can generate a receive interrupt. 1_B The received protocol-related argument PAR does not match the expected value. The reception of the data word sets bit PSR.AIF and can generate an alternative receive interrupt.
RDV10	13	rh	**Receive Data Valid in RBUF0** This bit indicates the status of the data content of register RBUF0. This bit is identical to bit RBUF01SRL.RDV00 and allows consisting reading of information for the receive buffer registers. 0_B Register RBUF0 does not contain data that has not yet been read out. 1_B Register RBUF0 contains data that has not yet been read out.

Field	Bits	Type	Description
RDV11	14	rh	**Receive Data Valid in RBUF1** This bit indicates the status of the data content of register RBUF1. This bit is identical to bit RBUF01SRL.RDV01 and allows consisting reading of information for the receive buffer registers. 0_B Register RBUF1 does not contain data that has not yet been read out. 1_B Register RBUF1 contains data that has not yet been read out.
DS1	15	rh	**Data Source** This bit indicates which receive buffer register (RBUF0 or RBUF1) is currently visible in registers RBUF(D) and in RBUFSR for the associated status information. It indicates which buffer contains the oldest data (the data that has been received first). This bit is identical to bit RBUF01SRL.DS0 and allows consisting reading of information for the receive buffer registers. 0_B The register RBUF contains the data of RBUF0 (same for associated status information). 1_B The register RBUF contains the data of RBUF1 (same for associated status information).
0	[5:4], 7, [12:10]	r	**Reserved** read as 0; should be written with 0.

19.2.12.3 Receive Buffer Registers RBUF, RBUFD

The receiver buffer register RBUF shows the contents of the either RBUF0 or RBUF1, depending on the order of reception. Always the oldest data (the data word that has been received first) from both receive buffers can be read from RBUF. It is recommended to read out the received data from RBUF instead of RBUF0/1. With a read access of at least the low byte of RBUF, the status of the receive data is automatically changed from “not yet read = valid” to “already read = not valid”, the content of RBUF becomes updated, and the next received data word becomes visible in RBUF.

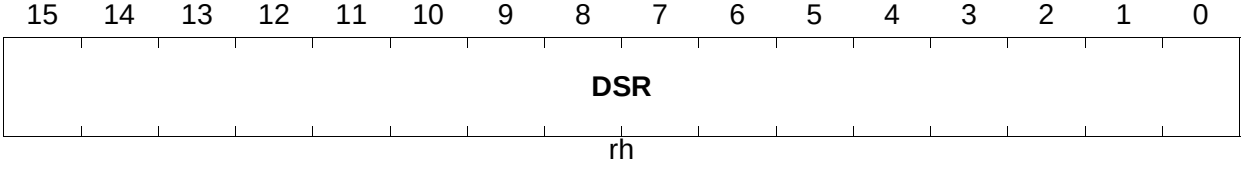
RBUF
Receiver Buffer Register **($5C_H$)** **Reset Value: 0000_H**



Field	Bits	Type	Description
DSR	[15:0]	rh	**Received Data** This bit field monitors the content of either RBUF0 or RBUF1, depending on the reception sequence.

If a debugger should be used to monitor the received data, the automatic update mechanism has to be de-activated to guaranty data consistency. Therefore, the receiver buffer register for debugging RBUFD is available. It is similar to RBUF, but without the automatic update mechanism by a read action. So a debugger (or other monitoring function) can read RBUFD without disturbing the receive sequence.

RBUFD
Receiver Buffer Reg. for Debugger **($4C_H$)** **Reset Value: 0000_H**



Field	Bits	Type	Description
DSR	[15:0]	rh	**Data from Shift Register** Same as RBUF.DSR, but without releasing the buffer after a read action.

The receive buffer status register RBUFSR provides the status of the data in receive buffers RBUF and RBUFD. If bits RBUF01SRL.DS0 (or RBUF01SRH.DS1) are 0, the content of RBUF01SRL is monitored in RBUFSR, otherwise the content of RBUF01SRH is shown.

RBUFSR
Receiver Buffer Status Register (58$_H$) Reset Value: 0000$_H$

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DS	**RDV 1**	**RDV 0**	**0**			**P ERR**	**PAR**	**0**	**SOF**	**0**		**WLEN**			
rh	rh	rh	r			rh	rh	r	rh	r		rh			

Field	Bits	Type	Description
WLEN	[3:0]	rh	**Received Data Word Length in RBUF or RBUFD** Description see RBUF01SRL.WLEN0 or RBUF01SRH.WLEN1.
SOF	6	rh	**Start of Frame in RBUF or RBUFD** Description see RBUF01SRL.SOF0 or RBUF01SRH.SOF1.
PAR	8	rh	**Protocol-Related Argument in RBUF or RBUFD** Description see RBUF01SRL.PAR0 or RBUF01SRH.PAR1.
PERR	9	rh	**Protocol-related Error in RBUF or RBUFD** Description see RBUF01SRL.PERR0 or RBUF01SRH.PERR1.
RDV0	13	rh	**Receive Data Valid in RBUF or RBUFD** Description see RBUF01SRL.RDV00 or RBUF01SRH.RDV10.
RDV1	14	rh	**Receive Data Valid in RBUF or RBUFD** Description see RBUF01SRL.RDV01 or RBUF01SRH.RDV11.

Field	Bits	Type	Description
DS	15	rh	**Data Source of RBUF or RBUFD** Description see RBUF01SRL.DS0 or RBUF01SRH.DS1.
0	[5:4], 7, [12:10]	r	**Reserved** read as 0; should be written with 0.

19.2.13 Operating the FIFO Data Buffer

The FIFO data buffers of a USIC module are built in a similar way, with transmit buffer and receive buffer capability for each channel. Depending on the device, the amount of available FIFO buffer area can vary. In the XC2000, totally 64 buffer entries can be distributed among the transmit or receive FIFO buffers of both channels of the USIC module.

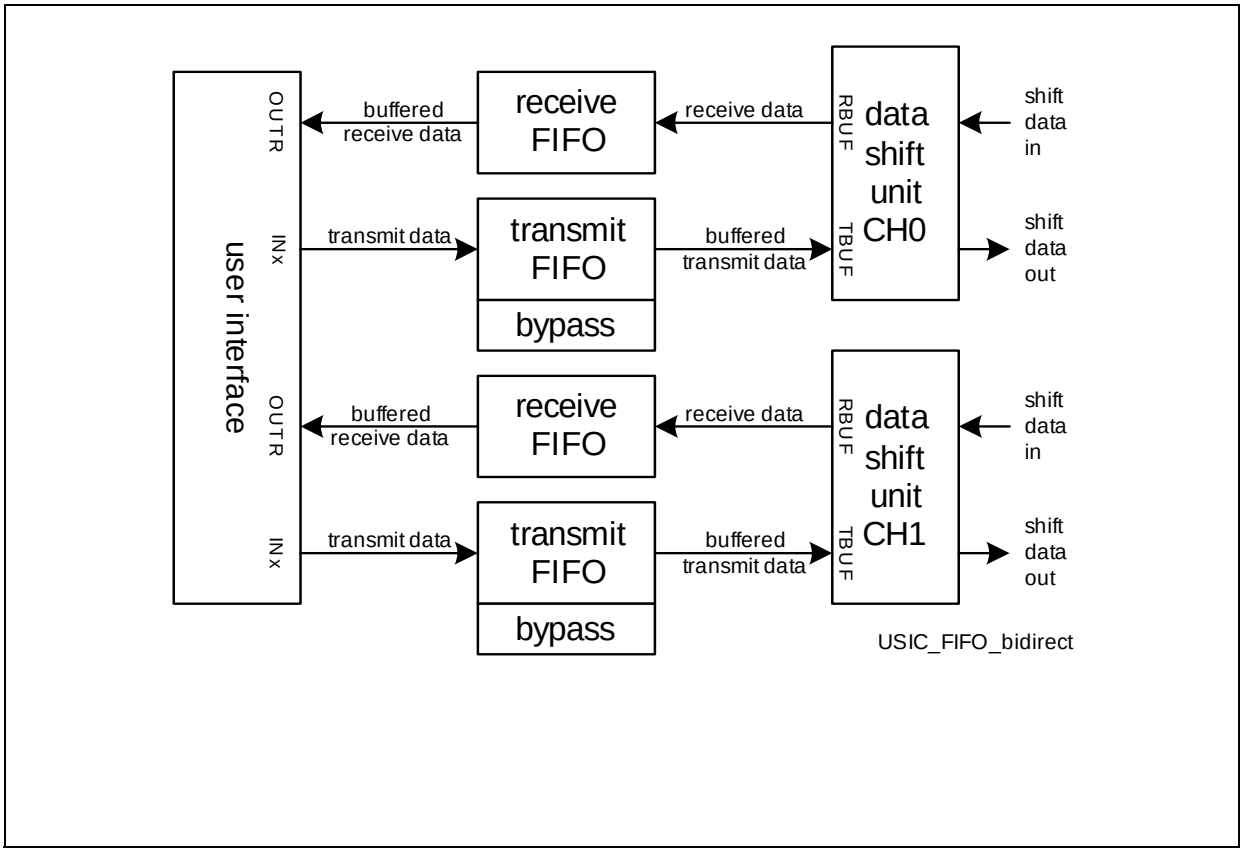


Figure 19-17 FIFO Buffer Overview

In order to operate the FIFO data buffers, the following issues have to be considered:

- FIFO buffer available and selected:
  The transmit FIFO buffer and the bypass structure are only available if CCFG.TB = 1, whereas the receive FIFO buffer is only available if CCFG.RB = 1.
  It is recommended to configure all buffer parameters while there is no data traffic for this USIC channel and the FIFO mechanism is disabled by TBCTRH.SIZE = 0 (for transmit buffer) or RBCTRH.SIZE = 0 (for receive buffer). The allocation of a buffer area by writing TBCTRL or RBCTRL has to be done while the corresponding FIFO buffer is disabled. The FIFO buffer interrupt control bits can be modified independently of data traffic.
- FIFO buffer setup:
  The total amount of available FIFO buffer entries limits the length of the transmit and

receive buffers for each USIC channel. The configuration of the size of each FIFO buffer and their locations in the available buffer range is described in **Section 19.2.13.1**.

- Bypass setup:
  In addition to the transmit FIFO buffer, a bypass can be configured as described in **Section 19.2.13.3**.

19.2.13.1 FIFO Buffer Partitioning

If available, the FIFO buffer area consists of a defined number of FIFO buffer entries, each containing a data part and the associated control information (RCI for receive data, TCI for transmit data). One FIFO buffer entry represents the finest granularity that can be allocated to a receive FIFO buffer or a transmit FIFO buffer. All available FIFO buffer entries of a USIC module are located one after the other in the FIFO buffer area. The overall counting starts with FIFO entry 0, followed by 1, 2, etc.
For each USIC module, a certain number of FIFO entries is available, that can be allocated to the channels of the same USIC module. It is not possible to assign FIFO buffer area to USIC channels that are not located within the same USIC module.
For each USIC channel, the size of the transmit and the receive FIFO buffer can be chosen independently. For example, it is possible to allocate the full amount of available FIFO entries as transmit buffer for one USIC channel. Some possible scenarios of FIFO buffer partitioning are shown in **Figure 19-18**.

Each FIFO buffer consists of a set of consecutive FIFO entries. The size of a FIFO data buffer can only be programmed as a power of 2, starting with 2 entries, then 4 entries, then 8 entries, etc. A FIFO data buffer can only start at a FIFO entry aligned to its size. For example, a FIFO buffer containing n entries can only start with FIFO entry 0, n, 2*n, 3*n, etc. and consists of the FIFO entries [x*n, (x+1)*n-1], with x being an integer number (incl. 0). It is not possible to have “holes” with unused FIFO entries within a FIFO buffer, whereas there can be unused FIFO entries between two FIFO buffers.

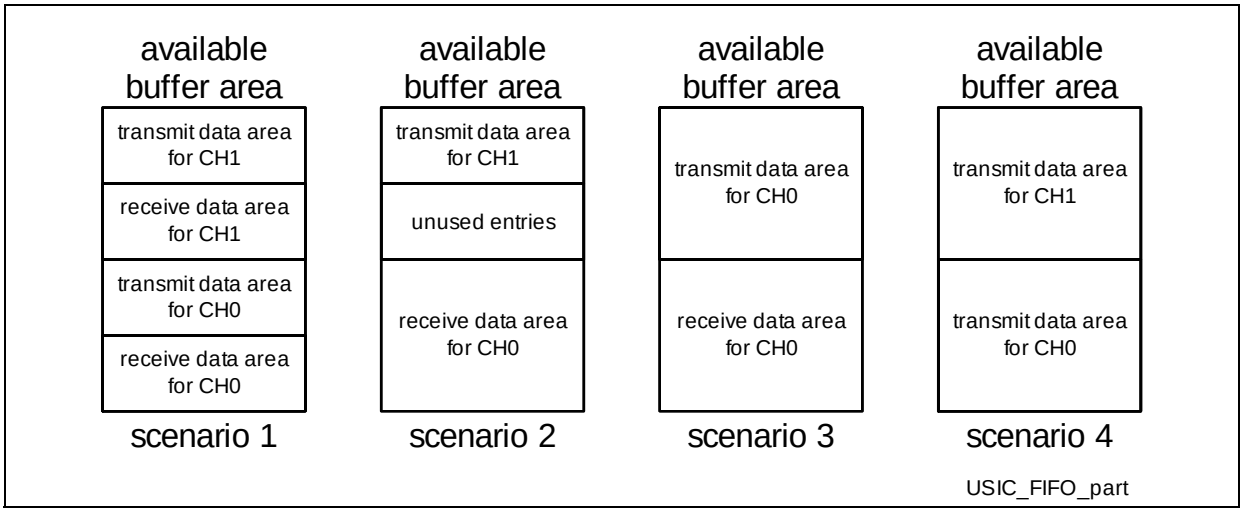


Figure 19-18 FIFO Buffer Partitioning

The data storage inside the FIFO buffers is based on pointers, that are internally updated whenever the data contents of the FIFO buffers have been modified. This happens automatically when new data is put into a FIFO buffer or the oldest data is taken from a FIFO buffer. As a consequence, the user program does not need to modify the pointers for data handling. Only during the initialization phase, the start entry of a FIFO buffer has to be defined by writing the number of the first FIFO buffer entry in the FIFO buffer to the corresponding bit field DPTR in register RBCTRL (for a receive FIFO buffer) or TBCTRL (for a transmit FIFO buffer) while the related bit field RBCTRH.SIZE=0 (or TBCTRH.SIZE = 0, respectively). The assignment of buffer entries to a FIFO buffer (regarding to size and pointers) must not be changed by SW while the related USIC channel is taking part in data traffic.

19.2.13.2 Data Buffer Events and Interrupts

The transmit FIFO buffer mechanism detects the following events, that can lead to interrupts (if enabled).

- Standard transmit buffer event:
  The filling level of the transmit buffer (given by TRBSRH.TBFLVL) exceeds (TBCTRH.LOF = 1) or falls below (TBCTRH.LOF = 0) a programmed limit (TBCTRL.LIMIT). The trigger of this event is the transition from equal to below or bigger, not the fact of being below or above.
  If the standard transmit buffer event is used to indicate that new data has to be written to one of the INx locations, TBCTRH.LOF = 0 should be programmed.
- Transmit buffer error event:
  The SW has written to a full buffer. The written value is ignored.

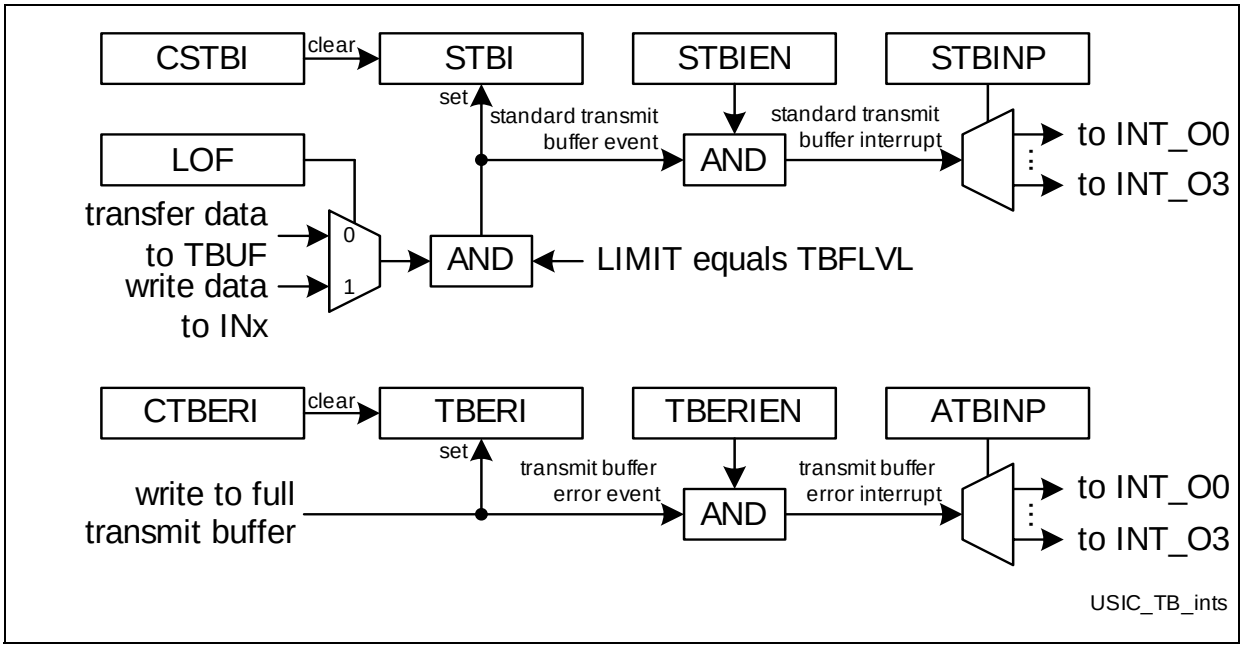


Figure 19-19 Transmit Buffer Events

The receive FIFO buffer mechanism detects the following events, that can lead to an interrupt (if enabled). The standard receive buffer event and the alternative receive buffer event can be programmed to two different modes, one referring to the filling level of the receive buffer, the other one related to a bit position in the receive control information RCI of the data word that becomes available in OUTRL.

If the interrupt generation refers to the filling level of the receive FIFO buffer, only the standard receive buffer event is used, whereas the alternative receive buffer event is not used. This mode can be selected to indicate that a certain amount of data has been received, without regarding the contents of the associated RCI.

If the interrupt generation refers to RCI, the filling level is not taken into account. Each time a new data word becomes available in OUTRL, an event is detected. If bit RCI[4] = 0, a standard receive buffer event is signaled, otherwise an alternative receive buffer device (RCI[4] = 1). Depending on the selected protocol and the setting of RBCTRH.RCIM, the value of RCI[4] can hold different information that can be used for protocol-specific interrupt handling (see protocol sections for more details).

- Standard receive buffer event in filling level mode (RBCTRH.RNM = 0):
  The filling level of the receive buffer (given by TRBSRH.RBFLVL) exceeds (RBCTRH.LOF = 1) or falls below (RBCTRH.LOF = 0) a programmed limit (RBCTRL.LIMIT). The trigger of this event is the transition from equal to below or greater, not the fact of being below or above.
  If the standard receive buffer event is used to indicate that new data has to be read from OUTRL, RBCTRH.LOF = 1 should be programmed.
- Standard receive buffer event in RCI mode (RBCTRH.RNM = 1):
  If the OUTR stage is updated with a new data value with RCI[4] = 0.

- Alternative receive buffer event in filling level mode (RBCTRH.RNM = 0): not used
- Alternative receive buffer event in RCI mode (RBCTRH.RNM = 1):
  If the OUTR stage is updated with a new value with RCI[4] = 1.
- Receive buffer error event:
  The SW reads from an empty buffer. The read data is invalid.

The following figure shows the receiver buffer events and interrupts in filling level mode.

Note: A buffer event in filling level mode occurs only when the filling level transitions away from the threshold value. Transitions starting with a filling level other than the threshold level generate no trigger event.

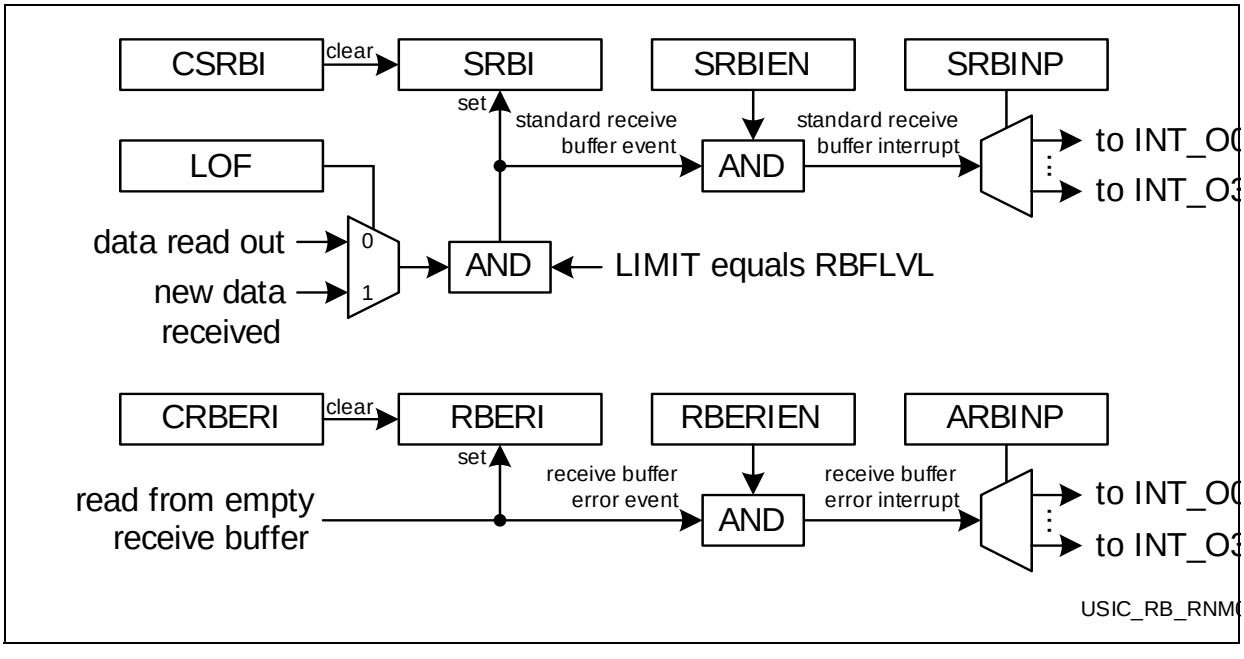


Figure 19-20 Receiver Buffer Events in Filling Level Mode

The following figure shows the receiver buffer events and interrupts in RCI mode.

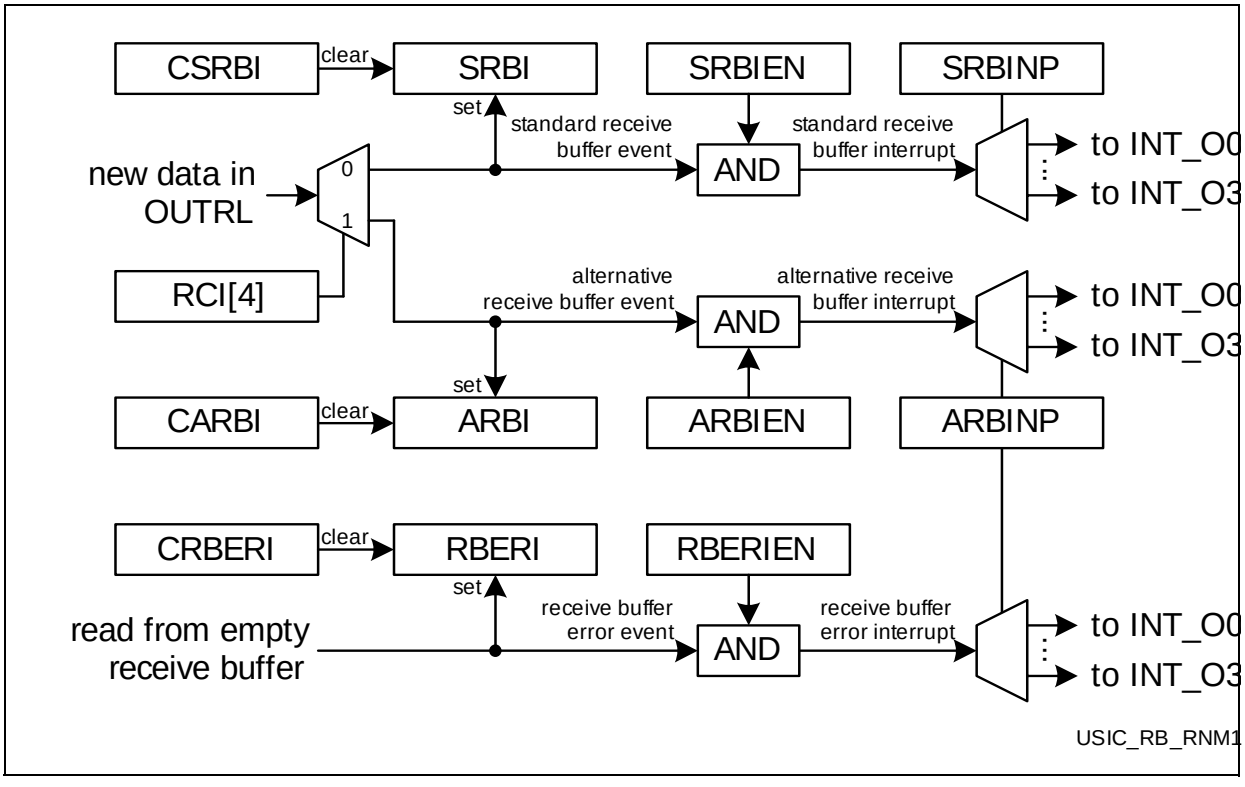


Figure 19-21 Receiver Buffer Events in RCI Mode

The table below shows the registers, bits and bit fields to indicate the buffer events and to control the interrupts related to the FIFO buffers (transmit and the receive) of a USIC channel.

Table 19-15 Buffer Events and Interrupt Handling

Event	Indication Flag	Indication cleared by	Interrupt enabled by	SRx Output selected by
Standard transmit buffer event	TRBSRL. STBI	TRBSCR. CSTBI	TBCTRH. STBIEN	TBCTRH. STBINP
Transmit buffer error event	TRBSRL. TBERI	TRBSCR. CTBERI	TBCTRH. TBERIEN	TBCTRH. ATBINP
Standard receive buffer event	TRBSRL. SRBI	TRBSCR. CSRBI	RBCTRH. SRBIEN	RBCTRH. SRBINP
Alternative receive buffer event	TRBSRL. ARBI	TRBSCR. CARBI	RBCTRH. ARBIEN	RBCTRH. ARBINP
Receive buffer error event	TRBSRL. RBERI	TRBSCR. CRBERI	RBCTRH. RBERIEN	RBCTRH. ARBINP

19.2.13.3 FIFO Buffer Bypass

The data bypass mechanism is part of the transmit FIFO control block. It allows to introduce a data word in the data stream without modifying the transmit FIFO buffer contents, e.g. to send a high-priority message. The bypass structure consists of a bypass data word of maximum 16 bits in register BYP and some associated control information in registers BYPCRL and BYPCRH. For example, these bits define the word length of the bypass data word and configure a transfer trigger and gating mechanism similar to the one for the transmit buffer TBUF.
The bypass data word can be tagged valid or invalid for transmission by bit BYRCRL.BDV (bypass data valid). A combination of data flow related and event related criteria define whether the bypass data word is considered valid for transmission. A data validation logic checks the start conditions for this data word. Depending on the result of the check, the transmit buffer register TBUF is loaded with different values, according to the following rules:

- Data from the transmit FIFO buffer or the bypass data can only be transferred to TBUF if TCSRL.TDV = 0 (TBUF is empty).
- Bypass data can only be transferred to TBUF if the bypass is enabled by BYPCRL.BDEN or the selecting gating condition is met.
- If the bypass data is valid for transmission and has either a higher transmit priority than the FIFO data or if the transmit FIFO is empty, the bypass data is transferred to TBUF.
- If the bypass data is valid for transmission and has a lower transmit priority than the FIFO buffer that contains valid data, the oldest transmit FIFO data is transferred to TBUF.
- If the bypass data is not valid for transmission and the FIFO buffer contains valid data, the oldest FIFO data is transferred to TBUF.
- If neither the bypass data is valid for transmission nor the transmit FIFO buffer contains valid data, TBUF is unchanged.

The bypass data validation is based on the logic blocks shown in **Figure 19-22**.

- A transfer gating logic enables or disables the bypass data word transfer to TBUF under SW or under HW control. If the input stage DX2 is not needed for data shifting, signal DX2S can be used for gating purposes. The transfer gating logic is controlled by bit field BYPCRL.BDEN.
- A transfer trigger logic supports data word transfers related to events, e.g. timer based or related to an input pin. If the input stage DX2 is not needed for data shifting, signal DX2T can be used for trigger purposes. The transfer trigger logic is controlled by bit BYPCRL.BDVTR.
- A bypass data validation logic combining the inputs from the gating logic, the triggering logic and TCSRL.TDV.

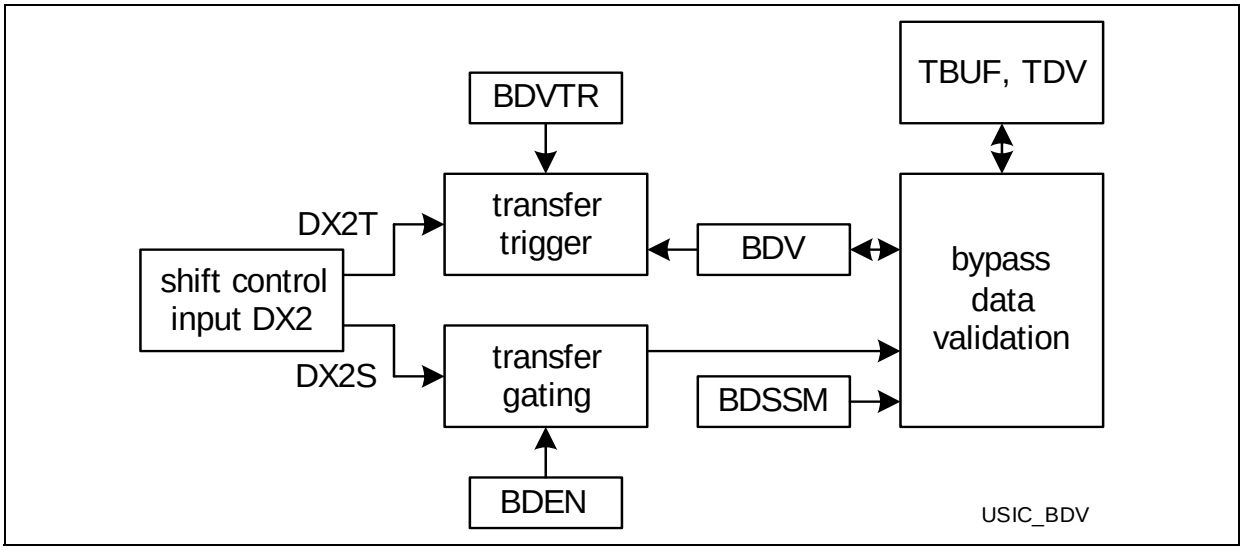


Figure 19-22 Bypass Data Validation

With this structure, the following bypass data transfer functionality can be achieved:

- Bit BYPCRL.BDSSM = 1 has to be programmed for a single-shot mechanism. After each transfer of the bypass data word to TBUF, the bypass data word has to be tagged valid again. This can be achieved either by writing a new bypass data word to BYP or by DX2T if BDVTR = 1 (e.g. trigger on a timer base or an edge at a pin).
- Bit BYPCRL.BDSSM = 0 has to be programmed if the bypass data is permanently valid for transmission (e.g. as alternative data if the data FIFO runs empty).

19.2.13.4 FIFO Access Constraints

The data in the shared FIFO buffer area is accessed by the HW mechanisms for data transfer of each communication channel (for transmission and reception) and by SW to read out received data or to write data to be transmitted. As a consequence, the data delivery rate can be limited by the FIFO mechanism. Each access by HW to the FIFO buffer area has priority over a SW access, that is delayed in case of an access collision. In order to avoid data loss and stalling of the CPU due to delayed SW accesses, the baud rate, the word length and the SW access mechanism have to be taken into account. Each access to the FIFO data buffer area by SW or by HW takes one period of f_{SYS}. Especially a continuous flow of very short, consecutive data words can lead to an access limitation.

19.2.13.5 Handling of FIFO Transmit Control Information

In addition to the transmit data, the transmit control information TCI can be transferred from the transmit FIFO or bypass structure to the USIC channel. Depending on the selected protocol and the enabled update mechanism, some settings of the USIC channel parameters can be modified. The modifications are based on the TCI of the FIFO data word loaded to TBUF or by the bypass control information if the bypass data is loaded into TBUF.

- TCSRL.SELMD = 1: update of PCRH.CTR[20:16] by FIFO TCI or BYPCRH.BSELO with additional clear of PCRH.CTR[23:21]
- TCSRL.WLEMD = 1: update of SCTRH.WLE and TCSRL.EOF by FIFO TCI or BYPCRL.BWLE (if the WLE information is overwritten by TCI or BWLE, the user has to take care that FLE is set accordingly)
- TCSRL.FLEMD = 1: update of SCTRH.FLE[4:0] by FIFO TCI or BYPCRL.BWLE with additional clear of SCTRH.FLE[5]
- TCSRL.WAMD = 1: update of TCSRL.WA by FIFO TCI[4]

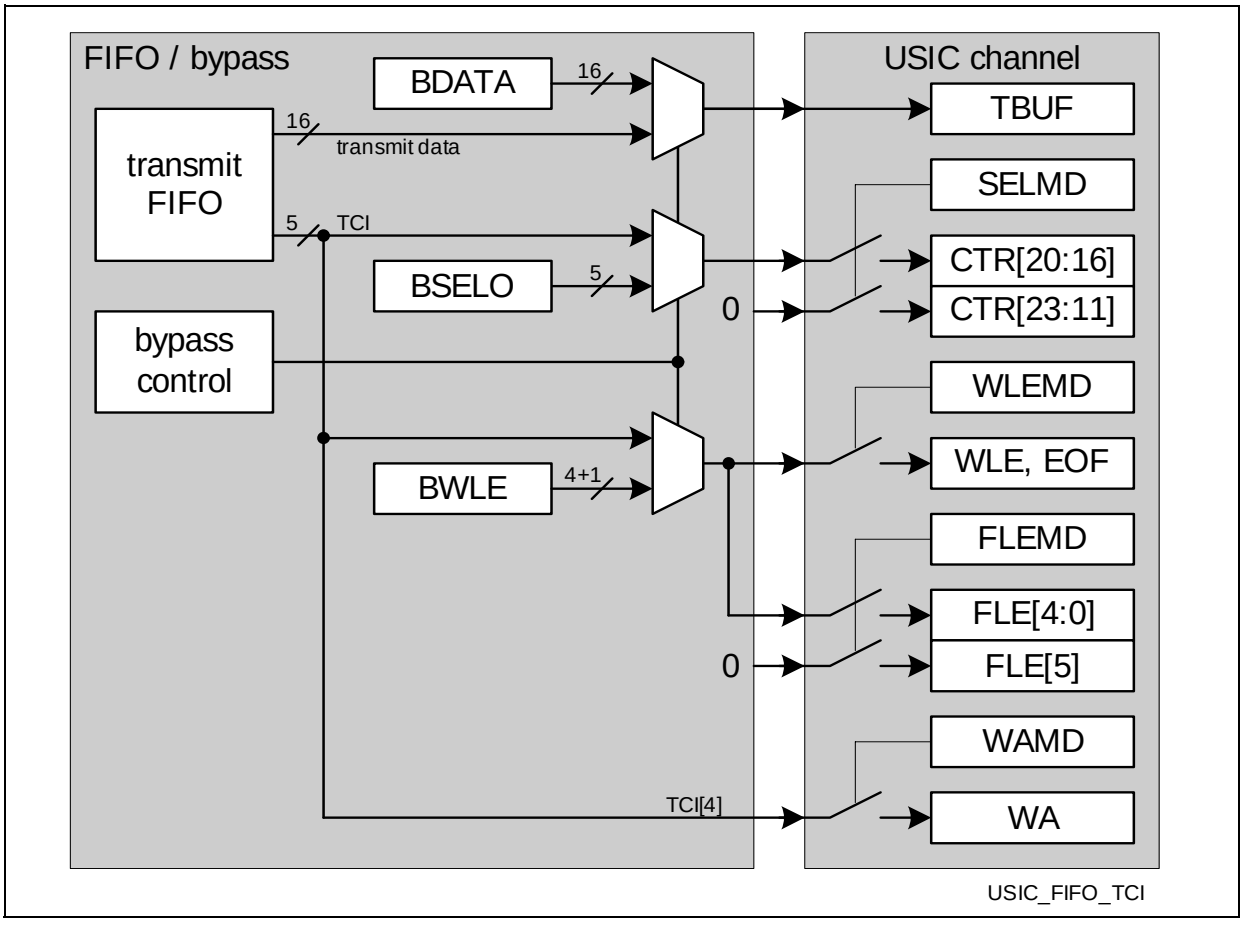


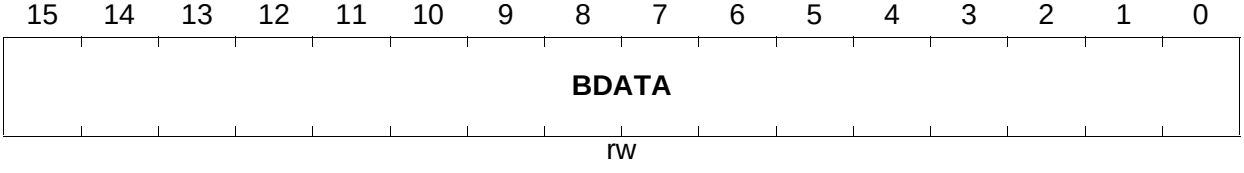
Figure 19-23 TCI Handling with FIFO / Bypass

19.2.14 FIFO Buffer and Bypass Registers

19.2.14.1 Bypass Registers

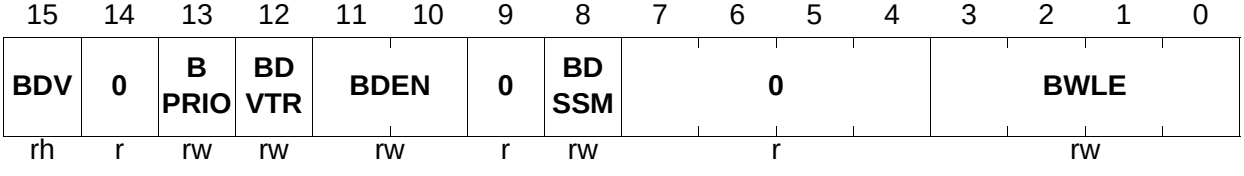
A write action to at least the low byte of the bypass data register sets BYPCRL.BDV = 1 (bypass data tagged valid).

BYP
Bypass Data Register **(100_H)** **Reset Value: 0000_H**



Bit (Field)	Width	Type	Description
BDATA	[15:0]	rw	**Bypass Data** This bit field contains the bypass data.

BYPCRL
Bypass Control Register L **(104_H)** **Reset Value: 0000_H**

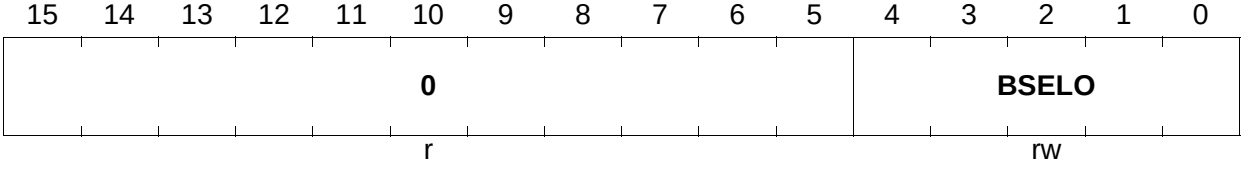


Field	Bits	Type	Description
BWLE	[3:0]	rw	**Bypass Word Length** This bit field defines the word length of the bypass data. The word length is given by BWLE + 1 with the data word being right-aligned in the data buffer at the bit positions [BWLE down to 0]. The bypass data word is always considered as an own frame with the length of BWLE. Same coding as SCTRH.WLE.

Field	Bits	Type	Description
BDSSM	8	rw	**Bypass Data Single Shot Mode** This bit defines if the bypass data is considered as permanently valid or if the bypass data is only transferred once (single shot mode). 0_B The bypass data is still considered as valid after it has been loaded into TBUF. The loading of the data into TBUF does not clear BDV. 1_B The bypass data is considered as invalid after it has been loaded into TBUF. The loading of the data into TBUF clears BDV.
BDEN	[11:10]	rw	**Bypass Data Enable** This bit field defines if and how the transfer of bypass data to TBUF is enabled. 00_B The transfer of bypass data is disabled. 01_B The transfer of bypass data to TBUF is possible. Bypass data will be transferred to TBUF according to its priority if BDV = 1. 10_B Gated bypass data transfer is enabled. Bypass data will be transferred to TBUF according to its priority if BDV = 1 and while DX2S = 0. 11_B Gated bypass data transfer is enabled. Bypass data will be transferred to TBUF according to its priority if BDV = 1 and while DX2S = 1.
BDVTR	12	rw	**Bypass Data Valid Trigger** This bit enables the bypass data for being tagged valid when DX2T is active (for time framing or time-out purposes). 0_B Bit BDV is not influenced by DX2T. 1_B Bit BDV is set if DX2T is active.
BPRIO	13	rw	**Bypass Priority** This bit defines the priority between the bypass data and the transmit FIFO data. 0_B The transmit FIFO data has a higher priority than the bypass data. 1_B The bypass data has a higher priority than the transmit FIFO data.

Field	Bits	Type	Description
BDV	15	rh	**Bypass Data Valid** This bit defines if the bypass data is valid for a transfer to TBUF. This bit is set automatically by a write access to at least the low-byte of register BYP. It can be cleared by SW by writing TRBSCR.CBDV. 0_B The bypass data is not valid. 1_B The bypass data is valid.
0	[7:4], 9, 14	r	**Reserved** read as 0; should be written with 0.

BYPCRH
Bypass Control Register H **(106_H)** **Reset Value: 0000_H**



Field	Bits	Type	Description
BSELO	[4:0]	rw	**Bypass Select Outputs** This bit field contains the value that is written to PCRH.CTR[20:16] if bypass data is transferred to TBUF while TCSRL.SELMD = 1. In the SSC protocol, this bit field can be used to define which SELOx output line will be activated when bypass data is transmitted.
0	[15:5]	r	**Reserved** read as 0; should be written with 0.

19.2.14.2 General FIFO Buffer Control Registers

The transmit and receive FIFO status information of UxCy is given in registers UxCy_TRBSRL/H.
The bits related to the transmitter buffer in this register can only by written if the transmit buffer functionality is enabled by CCFG.TB = 1, otherwise write accesses are ignored. A similar behavior applies for the bits related to the receive buffer referring to CCFG.RB = 1.

The interrupt flags (event flags) in the transmit and receive FIFO status register TRBSRL can be cleared by writing a 1 to the corresponding bit position in register TRBSCR, whereas writing a 0 has to effect on these bits. Writing a 1 by SW to SRBI, RBERI, ARBI, STBI, or TBERI sets the corresponding bit to simulate the detection of a transmit/receive buffer event, but without activating any service request output (therefore, see FMR.SIOx).

Bits TBUS and RBUS have been implemented for testing purposes. They can be ignored by data handling SW. Please note that a read action can deliver either a 0 or a 1 for these bits. It is recommended to treat them as “don’t care”.

TRBSRL
Transmit / Receive Buffer Status Reg. L (118_H) **Reset Value: 0808_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		TBUS	TFULL	TEMPTY	0	TBERI	STBI	0		RBUS	RFULL	REMPTY	ARBI	RBERI	SRBI
r		rh	rh	rh	r	rwh	rwh	r		rh	rh	rh	rwh	rwh	rwh

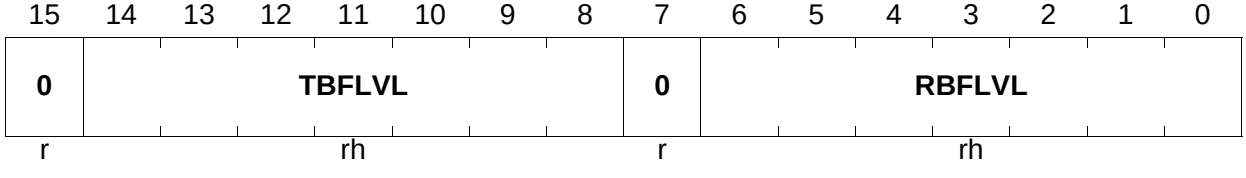
Field	Bits	Type	Description
SRBI	0	rwh	**Standard Receive Buffer Event** This bit indicates that a standard receive buffer event has been detected. It is cleared by writing TRBSCR.CSRBI = 1. If enabled by RBCTRH.SRBIEN, the service request output SRx selected by RBCTRH.SRBINP becomes activated if a standard receive buffer event is detected. 0_B A standard receive buffer event has not been detected. 1_B A standard receive buffer event has been detected.

Field	Bits	Type	Description
RBERI	1	rwh	**Receive Buffer Error Event** This bit indicates that a receive buffer error event has been detected. It is cleared by writing TRBSCR.CRBERI = 1. If enabled by RBCTRH.RBERIEN, the service request output SRx selected by RBCTRH.ARBINP becomes activated if a receive buffer error event is detected. 0_B A receive buffer error event has not been detected. 1_B A receive buffer error event has been detected.
ARBI	2	rwh	**Alternative Receive Buffer Event** This bit indicates that an alternative receive buffer event has been detected. It is cleared by writing TRBSCR.CARBI = 1. If enabled by RBCTRH.ARBIEN, the service request output SRx selected by RBCTRH.ARBINP becomes activated if an alternative receive buffer event is detected. 0_B An alternative receive buffer event has not been detected. 1_B An alternative receive buffer event has been detected.
REMPTY	3	rh	**Receive Buffer Empty** This bit indicates whether the receive buffer is empty. 0_B The receive buffer is not empty. 1_B The receive buffer is empty.
RFULL	4	rh	**Receive Buffer Full** This bit indicates whether the receive buffer is full. 0_B The receive buffer is not full. 1_B The receive buffer is full.

Field	Bits	Type	Description
RBUS	5	rh	**Receive Buffer Busy** This bit indicates whether the receive buffer is currently updated by the FIFO handler. 0_B The receive buffer information has been completely updated. 1_B The OUTRL/H update from the FIFO memory is ongoing. A read from OUTRL/H will be delayed. FIFO pointers from the previous read are not yet updated.
STBI	8	rwh	**Standard Transmit Buffer Event** This bit indicates that a standard transmit buffer event has been detected. It is cleared by writing TRBSCR.CSTBI = 1. If enabled by TBCTRH.STBIEN, the service request output SRx selected by TBCTRH.STBINP becomes activated if a standard transmit buffer event is detected. 0_B A standard transmit buffer event has not been detected. 1_B A standard transmit buffer event has been detected.
TBERI	9	rwh	**Transmit Buffer Error Event** This bit indicates that a transmit buffer error event has been detected. It is cleared by writing TRBSCR.CTBERI = 1. If enabled by TBCTRH.TBERIEN, the service request output SRx selected by TBCTRH.ATBINP becomes activated if a transmit buffer error event is detected. 0_B A transmit buffer error event has not been detected. 1_B A transmit buffer error event has been detected.
TEMPTY	11	rh	**Transmit Buffer Empty** This bit indicates whether the transmit buffer is empty. 0_B The transmit buffer is not empty. 1_B The transmit buffer is empty.

Field	Bits	Type	Description
TFULL	12	rh	**Transmit Buffer Full** This bit indicates whether the transmit buffer is full. 0_B The transmit buffer is not full. 1_B The transmit buffer is full.
TBUS	13	rh	**Transmit Buffer Busy** This bit indicates whether the transmit buffer is currently updated by the FIFO handler. 0_B The transmit buffer information has been completely updated. 1_B The FIFO memory update after write to INx is ongoing. A write to INx will be delayed. FIFO pointers from the previous INx write are not yet updated.
0	[7:6], [15:14]	r	**Reserved** read as 0; should be written with 0.

TRBSRH
Transmit / Receive Buffer Status Reg H ($11A_H$) **Reset Value: 0000_H**



Field	Bits	Type	Description
RBFLVL	[6:0]	rh	**Receive Buffer Filling Level** This bit field indicates the filling level of the receive buffer, starting with 0 for an empty buffer.
TBFLVL	[14:8]	rh	**Transmit Buffer Filling Level** This bit field indicates the filling level of the transmit buffer, starting with 0 for an empty buffer.
0	7, 15	r	**Reserved** read as 0; should be written with 0.

The bits in register TRBSCR are used to clear the notification bits in register TRBSRL or to clear the FIFO mechanism for the transmit or receive buffer. A read action always delivers 0.

TRBSCR

Transmit / Receive Buffer Status Clear Reg ($11C_H$) **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLUSHTB	FLUSHRB	0			CBDV	CTBERI	CSTBI	0					CARBI	CRBERI	CSRBI
w	w	r			w	w	w	r					w	w	w

Field	Bits	Type	Description
CSRBI	0	w	**Clear Standard Receive Buffer Event** 0_B No effect. 1_B Clear TRBSRL.SRBI.
CRBERI	1	w	**Clear Receive Buffer Error Event** 0_B No effect. 1_B Clear TRBSRL.RBERI.
CARBI	2	w	**Clear Alternative Receive Buffer Event** 0_B No effect. 1_B Clear TRBSRL.ARBI.
CSTBI	8	w	**Clear Standard Transmit Buffer Event** 0_B No effect. 1_B Clear TRBSRL.STBI.
CTBERI	9	w	**Clear Transmit Buffer Error Event** 0_B No effect. 1_B Clear TRBSRL.TBERI.
CBDV	10	w	**Clear Bypass Data Valid** 0_B No effect. 1_B Clear BYPCRL.BDV.
FLUSHRB	14	w	**Flush Receive Buffer** 0_B No effect. 1_B The receive FIFO buffer is cleared (filling level is cleared and output pointer is set to input pointer value). Should only be used while the FIFO buffer is not taking part in data traffic.

Field	Bits	Type	Description
FLUSHTB	15	w	**Flush Transmit Buffer** 0_B No effect. 1_B The transmit FIFO buffer is cleared (filling level is cleared and output pointer is set to input pointer value). Should only be used while the FIFO buffer is not taking part in data traffic.
0	[7:3], [13:11]	r	**Reserved** read as 0; should be written with 0.

19.2.14.3 Transmit FIFO Buffer Control Registers

The transmit FIFO buffer is controlled by registers TBCTRL and TBCTRH. These registers can only by written if the transmit buffer functionality is enabled by CCFG.TB = 1, otherwise write accesses are ignored.

TBCTRL
Transmitter Buffer Control Reg. L (110_H) Reset Value: 0000_H

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

Field	Bits	Type	Description
DPTR	[5:0]	w	**Data Pointer** This bit field defines the start value for the transmit buffer pointers when assigning the FIFO entries to the transmit FIFO buffer. A read always delivers 0. When writing DPTR while SIZE = 0, both internal pointers are updated with the written value and the buffer is considered as empty. A write access to DPTR while SIZE > 0 is ignored and does not modify the pointers.
LIMIT	[13:8]	rw	**Limit For Interrupt Generation** This bit field defines the target filling level of the transmit FIFO buffer that is used for the standard transmit buffer event detection.
0	[7:6], [15:14]	r	**Reserved** read as 0; should be written with 0.

TBCTRH
Transmitter Buffer Control Reg. H (112_H) Reset Value: 0000_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TBERIEN	**STBIEN**	**0**	**LOF**	**0**	**SIZE**			**0**			**ATBINP**		**0**	**STBINP**	
rw	rw	r	rw	r	rw			r			rw		r	rw	

Field	Bits	Type	Description
STBINP	[1:0]	rw	**Standard Transmit Buffer Interrupt Node Pointer** This bit field defines which service request output SRx will be activated in case of a standard transmit buffer event. 00_B Output SR0 will be activated. 01_B Output SR1 will be activated. 10_B Output SR2 will be activated. 11_B Output SR3 will be activated.
ATBINP	[4:3]	rw	**Alternative Transmit Buffer Interrupt Node Pointer** This bit field define which service request output SRx will be activated in case of a transmit buffer error event. 00_B Output SR0 will be activated. 01_B Output SR1 will be activated. 10_B Output SR2 will be activated. 11_B Output SR3 will be activated.
SIZE	[10:8]	rw	**Buffer Size** This bit field defines the number of FIFO entries assigned to the transmit FIFO buffer. 000_B The FIFO mechanism is disabled. The buffer does not accept any request for data. 001_B The FIFO buffer contains 2 entries. 010_B The FIFO buffer contains 4 entries. 011_B The FIFO buffer contains 8 entries. 100_B The FIFO buffer contains 16 entries. 101_B The FIFO buffer contains 32 entries. 110_B The FIFO buffer contains 64 entries. 111_B reserved

Field	Bits	Type	Description
LOF	12	rw	**Buffer Event on Limit Overflow** This bit defines which relation between filling level and programmed limit leads to a standard transmit buffer event. 0_B A standard transmit buffer event occurs when the filling level equals the limit value and gets lower due to transmission of a data word. 1_B A standard transmit buffer interrupt event occurs when the filling level equals the limit value and gets bigger due to a write access to a data input location INx.
STBIEN	14	rw	**Standard Transmit Buffer Interrupt Enable** This bit enables/disables the generation of a standard transmit buffer interrupt in case of a standard transmit buffer event. 0_B The standard transmit buffer interrupt generation is disabled. 1_B The standard transmit buffer interrupt generation is enabled.
TBERIEN	15	rw	**Transmit Buffer Error Interrupt Enable** This bit enables/disables the generation of a transmit buffer error interrupt in case of a transmit buffer error event (SW writes to a full transmit buffer). 0_B The transmit buffer error interrupt generation is disabled. 1_B The transmit buffer error interrupt generation is enabled.
0	2, [7:5], 11, 13	r	**Reserved** read as 0; should be written with 0.

19.2.14.4 Receive FIFO Buffer Control Registers

The receive FIFO buffer is controlled by registers RBCTRL and RBCTRH. These registers can only by written if the receive buffer functionality is enabled by CCFG.RB = 1, otherwise write accesses are ignored.

RBCTRL
Receiver Buffer Control Register L (114$_H$) **Reset Value: 0000$_H$**

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

Field	Bits	Type	Description
DPTR	[5:0]	w	**Data Pointer** This bit field defines the start value for the receive buffer pointers when assigning the FIFO entries to the receive FIFO buffer. A read always delivers 0. When writing DPTR while SIZE = 0, both internal pointers are updated with the written value and the buffer is considered as empty. A write access to DPTR while SIZE > 0 is ignored and does not modify the pointers.
LIMIT	[13:8]	rw	**Limit For Interrupt Generation** This bit field defines the target filling level of the receive FIFO buffer that is used for the standard receive buffer event detection.
0	[7:6], [15:14]	r	**Reserved** read as 0; should be written with 0.

RBCTRH
Receiver Buffer Control Register H (116$_H$) **Reset Value: 0000$_H$**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RBERIEN	**SRBIEN**	**ARBIEN**	**LOF**	**RNM**	**SIZE**			**RCIM**		**0**	**ARBINP**		**0**	**SRBINP**	
rw	rw	rw	rw	rw	rw			rw		r	rw		r	rw	

Field	Bits	Type	Description
SRBINP	[1:0]	rw	**Standard Receive Buffer Interrupt Node Pointer** This bit field defines which service request output SRx will be activated in case of a standard receive buffer event. 00_B Output SR0 will be activated. 01_B Output SR1 will be activated. 10_B Output SR2 will be activated. 11_B Output SR3 will be activated.
ARBINP	[4:3]	rw	**Alternative Receive Buffer Interrupt Node Pointer** This bit field defines which service request output SRx will be activated in case of an alternative receive buffer event or a receive buffer error event. 00_B The output SR0 will be activated. 01_B The output SR1 will be activated. 10_B The output SR2 will be activated. 11_B The output SR3 will be activated.
RCIM	[7:6]	rw	**Receiver Control Information Mode** This bit field defines which information from the receiver status register RBUFSR is propagated as 5 bit receiver control information RCI[4:0] to the receive FIFO buffer and can be read out in registers OUT(D)RH. 00_B RCI[4] = PERR, RCI[3:0] = WLEN 01_B RCI[4] = SOF, RCI[3:0] = WLEN 10_B RCI[4] = 0, RCI[3:0] = WLEN 11_B RCI[4] = PERR, RCI[3] = PAR, RCI[2:1] = 00_B, RCI[0] = SOF

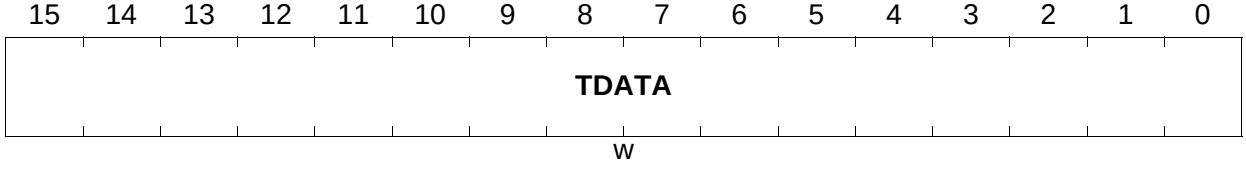
Field	Bits	Type	Description
SIZE	[10:8]	rw	**Buffer Size** This bit field defines the number of FIFO entries assigned to the receive FIFO buffer. 000_B The FIFO mechanism is disabled. The buffer does not accept any request for data. 001_B The FIFO buffer contains 2 entries. 010_B The FIFO buffer contains 4 entries. 011_B The FIFO buffer contains 8 entries. 100_B The FIFO buffer contains 16 entries. 101_B The FIFO buffer contains 32 entries. 110_B The FIFO buffer contains 64 entries. 111_B reserved
RNM	11	rw	**Receiver Notification Mode** This bit defines the receive buffer event mode. The receive buffer error event is not affected by RNM. 0_B Filling level mode: A standard receive buffer event occurs when the filling level equals the limit value and changes, either due to a read access from OUTRL (LOF = 0) or due to a new received data word (LOF = 1). 1_B RCI mode: A standard receive buffer event occurs when register OUTRL is updated with a new value if the corresponding value in OUTRH.RCI[4] = 0. If OUTRH.RCI[4] = 1, an alternative receive buffer event occurs instead of the standard receive buffer event.
LOF	12	rw	**Buffer Event on Limit Overflow** This bit defines which relation between filling level and programmed limit leads to a standard receive buffer event in filling level mode (RNM = 0). In RCI mode (RNM = 1), bit fields LIMIT and LOF are ignored. 0_B A standard receive buffer event occurs when the filling level equals the limit value and gets lower due to a read access from OUTRL. 1_B A standard receive buffer event occurs when the filling level equals the limit value and gets bigger due to the reception of a new data word.

Field	Bits	Type	Description
ARBIEN	13	rw	**Alternative Receive Buffer Interrupt Enable** This bit enables/disables the generation of an alternative receive buffer interrupt in case of an alternative receive buffer event. 0_B The alternative receive buffer interrupt generation is disabled. 1_B The alternative receive buffer interrupt generation is enabled.
SRBIEN	14	rw	**Standard Receive Buffer Interrupt Enable** This bit enables/disables the generation of a standard receive buffer interrupt in case of a standard receive buffer event. 0_B The standard receive buffer interrupt generation is disabled. 1_B The standard receive buffer interrupt generation is enabled.
RBERIEN	15	rw	**Receive Buffer Error Interrupt Enable** This bit enables/disables the generation of a receive buffer error interrupt in case of a receive buffer error event (the SW reads from an empty receive buffer). 0_B The receive buffer error interrupt generation is disabled. 1_B The receive buffer error interrupt generation is enabled.
0	2, 5	r	**Reserved** read as 0; should be written with 0.

19.2.14.5 FIFO Buffer Data Registers

The 32 independent data input locations IN00 to IN31 are addresses that can be used as data entry locations for the transmit FIFO buffer. Data written to one of these locations will be stored in the transmit buffer FIFO. Additionally, the 5 bit coding of the number [31:0] of the addressed data input location represents the transmit control information TCI.
If the FIFO is already full and new data is written to it, the write access is ignored and a transmit buffer error event is signaled.

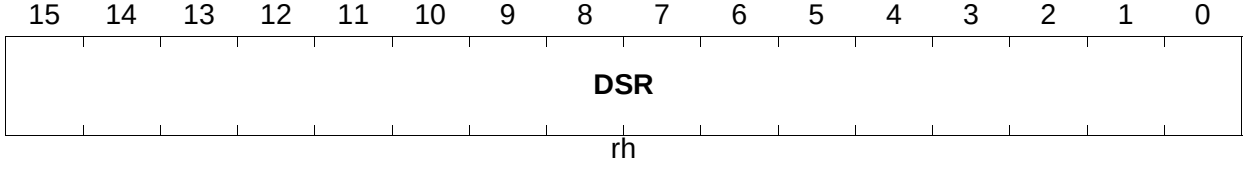
INx (x = 00-31)
Transmit FIFO Buffer Input Location x(180_H + x *4) **Reset Value: 0000_H**



Field	Bits	Type	Description
TDATA	[15:0]	w	**Transmit Data** This bit field contains the data to be transmitted (write view), read actions deliver 0. A write action to at least the low byte of TDATA triggers the data storage in the FIFO.

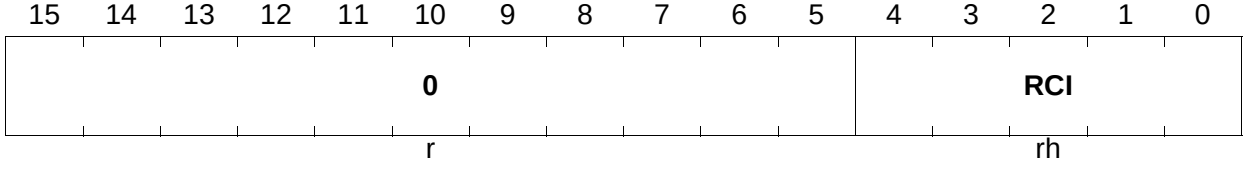
The receiver FIFO buffer output register OUTRL shows the oldest received data word in the FIFO buffer. A read action from this address location delivers the received data. With a read access of at least the low byte, the data is declared to be read and the next entry becomes visible. Register OUTRH contains the receiver control information RCI containing the information selected by RBCTRH.RCIM.
Write accesses are ignored.

OUTRL
Receiver Buffer Output Register L (120$_H$) **Reset Value: 0000$_H$**



Field	Bits	Type	Description
DSR	[15:0]	rh	**Received Data** This bit field monitors the content of the oldest data word in the receive FIFO. Reading at least the low byte releases the buffer entry currently shown in DSR.

OUTRH
Receiver Buffer Output Register H (122$_H$) **Reset Value: 0000$_H$**

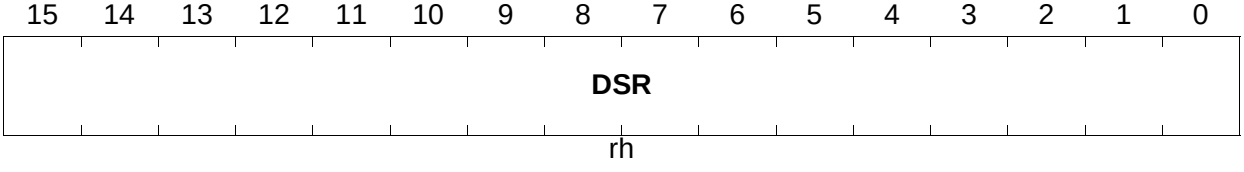


Field	Bits	Type	Description
RCI	[4:0]	rh	**Receiver Control Information** This bit field monitors the receiver control information associated to DSR. The bit structure of RCI depends on bit field RBCTRH.RCIM.
0	[15:5]	r	**Reserved** read as 0; should be written with 0.

If a debugger should be used to monitor the received data in the FIFO buffer, the FIFO mechanism must not be activated in order to guaranty data consistency. Therefore, a second address set is available, named OUTDRL/H (D like debugger), having the same bit fields like the original buffer output register OUTRL/H, but without the FIFO mechanism. A debugger can read here (in order to monitor the receive data flow) without the risk of data corruption. Write accesses are ignored.

OUTDRL

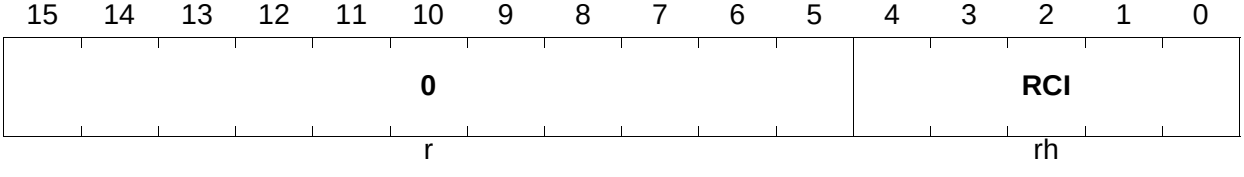
Receiver Buffer Output Register L for Debugger (124_H) **Reset Value: 0000_H**



Field	Bits	Type	Description
DSR	[15:0]	rh	**Data from Shift Register** Same as OUTRL.DSR, but without releasing the buffer after a read action.

OUTDRH

Receiver Buffer Output Register H for Debugger (126_H) **Reset Value: 0000_H**

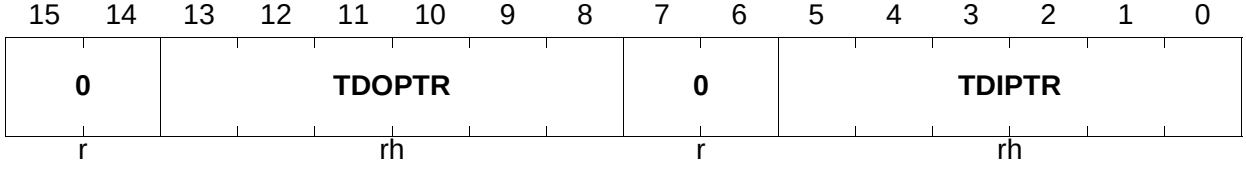


Field	Bits	Type	Description
RCI	[4:0]	rh	**Receive Control Information from Shift Register** Same as OUTRH.RCI.
0	[15:5]	r	**Reserved** read as 0; should be written with 0.

19.2.14.6 FIFO Buffer Pointer Registers

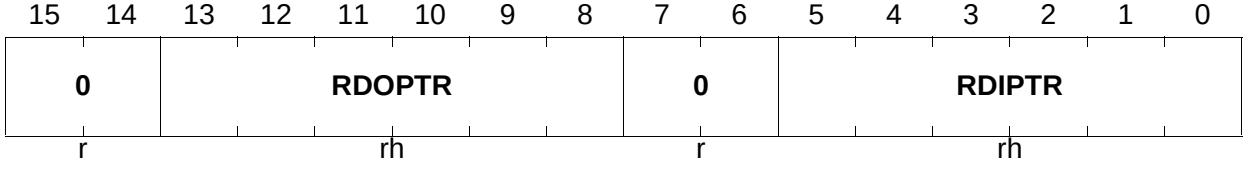
The pointers for FIFO handling of the transmit and receive FIFO buffers are located in registers TRBPTRL (for the transmit buffer) and TRBPTRH (for the receive buffer). The pointers are automatically handled by the FIFO buffer mechanism and do not need to be modified by SW. As a consequence, these registers can only be read by SW (e.g. for verification purposes), whereas write accesses are ignored.

TRBPTRL
Transmit / Receive Buffer Pointer Reg L (108_H) **Reset Value: 0000_H**



Field	Bits	Type	Description
TDIPTR	[5:0]	rh	**Transmitter Data Input Pointer** This bit field indicates the buffer entry that will be used for the next transmit data coming from the INx addresses.
TDOPTR	[13:8]	rh	**Transmitter Data Input Pointer** This bit field indicates the buffer entry that will be used for the next transmit data to be output to TBUF.
0	[7:6], [15:14]	r	**Reserved** read as 0; should be written with 0.

TRBPTRH
Transmit / Receive Buffer Pointer Reg H ($10A_H$) **Reset Value: 0000_H**



Field	Bits	Type	Description
RDIPTR	[5:0]	rh	**Receiver Data Input Pointer** This bit field indicates the buffer entry that will be used for the next receive data coming from RBUF.

Field	Bits	Type	Description
RDOPTR	[13:8]	rh	**Receiver Data Input Pointer** This bit field indicates the buffer entry that will be used for the next receive data to be output at the OUT(D)RL addresses.
0	[7:6], [15:14]	r	**Reserved** read as 0; should be written with 0.

19.3 Asynchronous Serial Channel (ASC = UART)

The asynchronous serial channel ASC covers the reception and the transmission of asynchronous data frames and provides a hardware LIN support. The receiver and transmitter being independent, frames can start at different points in time for transmission and reception.
The ASC mode is selected by CCR.MODE = 0010_B with CCFG.ASC = 1 (ASC mode available).

This chapter contains the following sections:

- Signal description (see **Section 19.2.5**)
- Frame format (see **Section 19.3.2**)
- Bit timing (see **Section 19.3.3.1**)
- Operating the ASC (see **Section 19.3.3**)
- Protocol registers (see **Section 19.3.4**)
- Hardware LIN support (see **Section 19.3.5**)

19.3.1 Signal Description

An ASC connection is characterized by the use of a single connection line between a transmitter and a receiver. The receiver input RxD signal is handled by the input stage DX0.

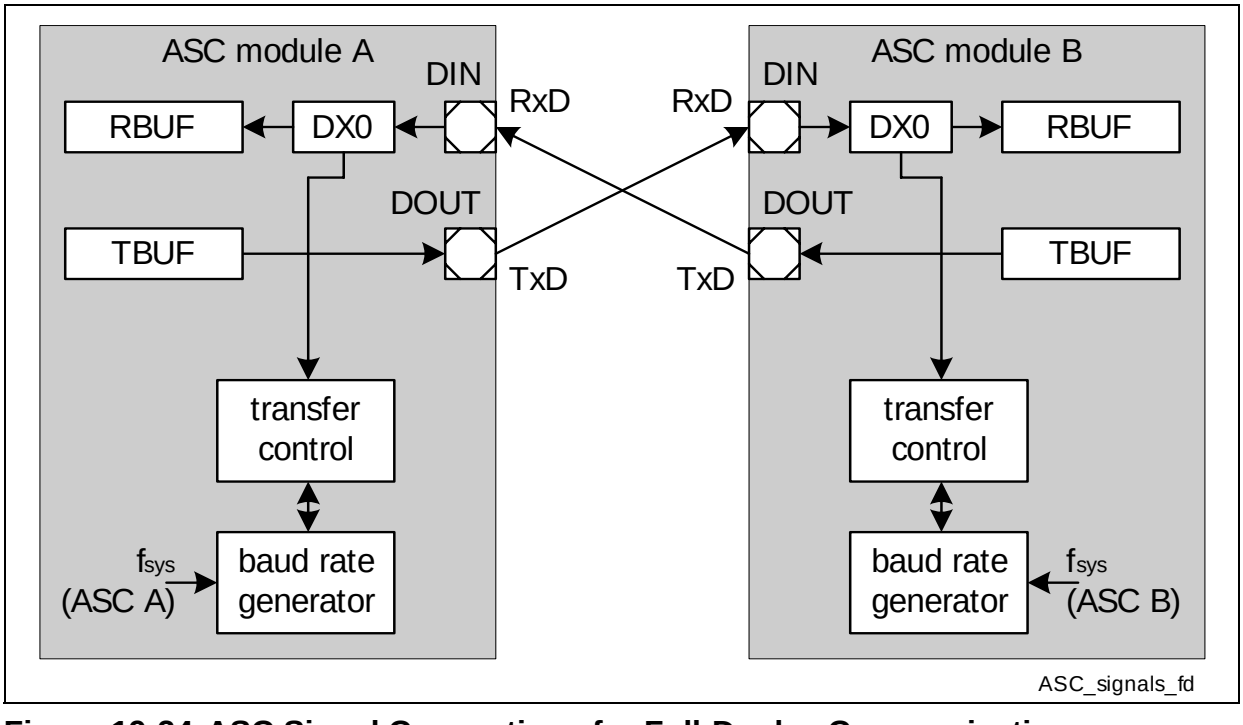


Figure 19-24 ASC Signal Connections for Full-Duplex Communication

For full-duplex communication, an independent communication line is needed for each transfer direction. **Figure 19-24** shows an example with a point-to-point full-duplex

connection between two communication partners ASC A and ASC B.
For half-duplex or multi-transmitter communication, a single communication line is shared between the communication partners. **Figure 19-25** shows an example with a point-to-point half-duplex connection between ASC A and ASC B. In this case, the user has to take care that only one transmitter is active at a time. In order to support transmitter collision detection, the input stage DX1 can be used to monitor the level of the transmit line and to check if the line is in the idle state or if a collision occurred.
There are two possibilities to connect the receiver input DIN to the transmitter output DOUT. Communication partner ASC A uses an internal connection with only the transmit pin TxD, that is delivering its input value as RxD to the DX0 input stage for reception and to DX1 to check for transmitter collisions. Communication partner ASC B uses an external connection between the two pins TxD and RxD.

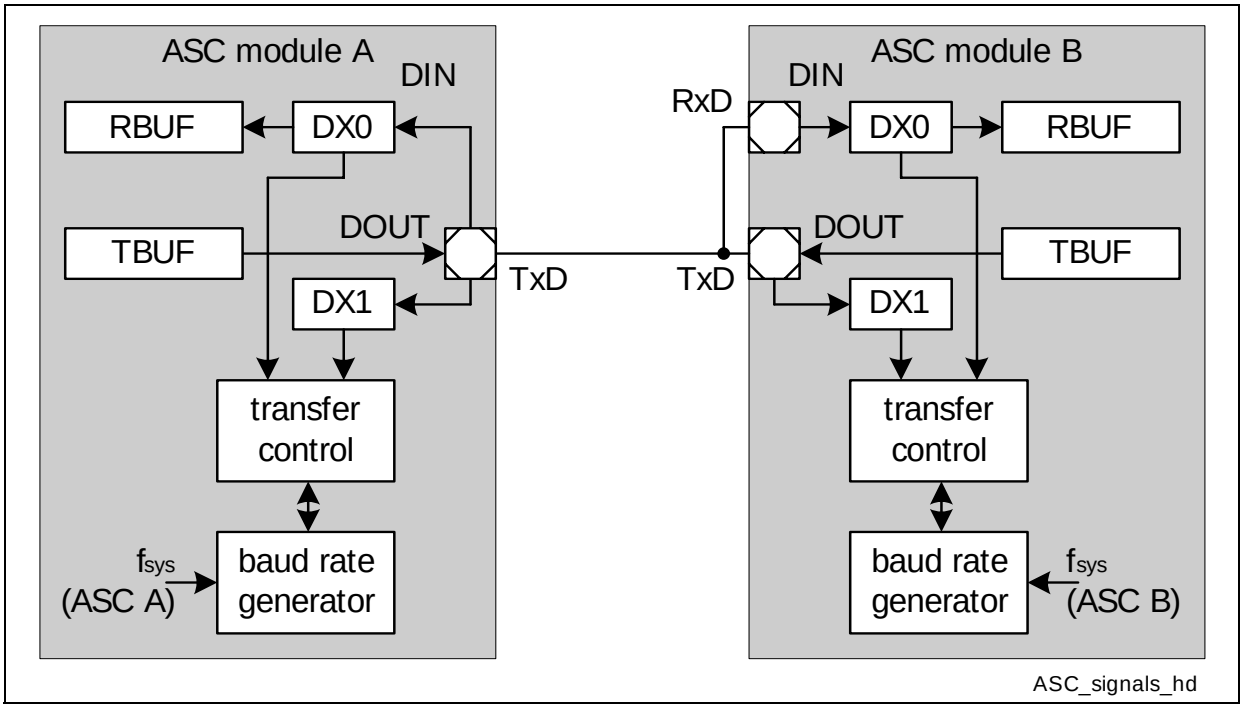


Figure 19-25 ASC Signal Connections for Half-Duplex Communication

19.3.2 Frame Format

A standard ASC frame is shown in **Figure 19-26**. It consists of:

- An idle time with the signal level 1.
- One start of frame bit (SOF) with the signal level 0.
- A data field containing a programmable number of data bits (1-63).
- A parity bit (P), programmable for either even or odd parity. It is optionally possible to handle frames without parity bit.
- One or two stop bits with the signal level 1.

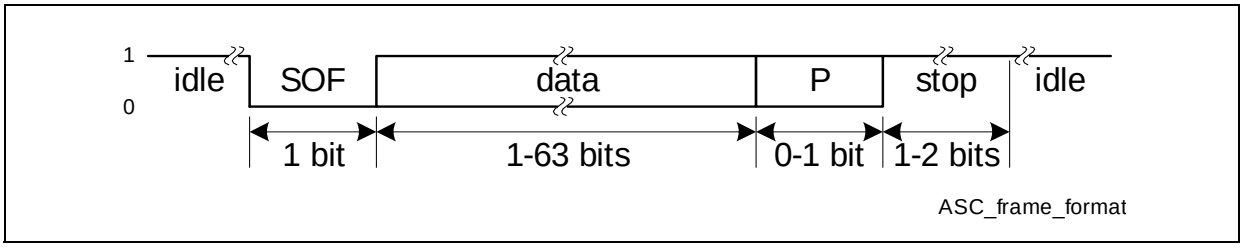


Figure 19-26 Standard ASC Frame Format

The protocol specific bits (SOF, P, stop) are automatically handled by the ASC protocol state machine and do not appear in the data flow via the receive and transmit buffers.

19.3.2.1 Idle Time

The receiver and the transmitter independently check the respective data input lines (DX0, DX1) for being idle. The idle detection ensures that an SOF bit of a recently enabled ASC module does not collide with an already running frame of another ASC module.
In order to start the idle detection, the user SW has to clear bits PSR.RXIDLE and/or PSR.TXIDLE, e.g. before selecting the ASC mode or during operation. If a bit is cleared by SW while a data transfer is in progress, the currently running frame transfer is finished normally before starting the idle detection again. Frame reception is only possible if PSR.RXIDLE = 1 and frame transmission is only possible if PSR.TXIDLE = 1.
The duration of the idle detection depends on the setting of bit PCRL.IDM. In the case that a collision is not possible, the duration can be shortened and the bus can be declared as being idle by setting PCRL.IDM = 0.
In the case that the complete idle detection is enabled by PCRL.IDM = 1, the data input of DX0 is considered as idle (PSR.RXIDLE becomes set) if a certain number of consecutive passive bit times has been detected. The same scheme applies for the transmitter's data input of DX1. Here, bit PSR.TXIDLE becomes set if the idle condition of this input signal has been detected.
The duration of the complete idle detection is given by the number of programmed data bits per frame plus 2 (in the case without parity) or plus 3 (in the case with parity). The counting of consecutive bit times with 1 level restarts from the beginning each time an edge is found, after leaving a stop mode or if ASC mode becomes enabled.
If the idle detection bits PSR.RXIDLE and/or TXIDLE are cleared by SW, the counting scheme is not stopped (no re-start from the beginning). As a result, the cleared bit(s) can become set immediately again if the respective input line still meets the idle criterion.
Please note that the idle time check is based on bit times, so the maximum time can be up to 1 bit time more than programmed value (but not less).

19.3.2.2 Start Bit Detection

The receiver input signal DIN (selected signal of input stage DX0) is checked for a falling edge. An SOF bit is detected if a falling edge occurs while the receiver is idle or after the sampling point of the last stop bit. To increase noise immunity, the SOF bit timing starts with the first falling edge that is detected. If the sampled bit value of the SOF is 1, the previous falling edge is considered to be due to noise and the receiver is considered to be idle again.

19.3.2.3 Data Field

The length of the data field (number of data bits) can be programmed by bit field SCTRH.FLE. It can vary between 1 and 63 data bits, corresponding to values of SCTRH.FLE = 0 to 62 (the value of 63 is reserved and must not be programmed in ASC mode).
The data field can consist of several data words, e.g. a transfer of 12 data bits can be composed of two 8 bit words, with the 12 bits being split into 8 bits of the first word and 4 bits of the second word. The user SW has to take care that the transmit data is available in-time, once a frame has been started. If the transmit buffer runs empty during a running data frame, the passive data level (SCTRL.PDL) is sent out.
The shift direction can be programmed by SCTRL.SDIR. The standard setting for ASC frames with LSB first is achieved with the default setting SDIR = 0.

19.3.2.4 Parity Bit

The ASC allows parity generation for transmission and parity check for reception on frame base. The type of parity can be selected by bit field CCR.PM, common for transmission and reception (no parity, even or odd parity). If the parity handling is disabled, the ASC frame does not contain any parity bit. For consistency reasons, all communication partners have to be programmed to the same parity mode.
After the last data bit of the data field, the transmitter automatically sends out its calculated parity bit if parity generation has been enabled. The receiver interprets this bit as received parity and compares it to its internally calculated one. The received parity bit value and the result of the parity check are monitored in the receiver buffer status registers as receiver buffer status information. These registers contain bits to monitor a protocol-related argument (PAR) and protocol-related error indication (PERR).

19.3.2.5 Stop Bit(s)

Each ASC frame is completed by 1 or 2 of stop bits with the signal level 1 (same level as the idle level). The number of stop bits is programmable by bit PSR.STPB.
A new start bit can transferred directly after the last stop bit.

19.3.3 Operating the ASC

In order to operate the ASC protocol, the following issues have to be considered:

- Select ASC mode:
  It is recommended to configure all parameters of the ASC that do not change during run time while CCR.MODE = 0000_B. Bit field SCTRL.TRM = 01_B has to be programmed. The configuration of the input stages has to be done while CCR.MODE = 0000_B to avoid unintended edges of the input signals and the ASC mode can be enabled by CCR.MODE = 0010_B afterwards.
- Pin connections:
  Establish a connection of input stage DX0 with the selected receive data input pin (signal DIN) with DX0CR.INSW = 0 and configure a transmit data output pin (signal DOUT). For collision or idle detection of the transmitter, the input stage DX1 has to be connected to the selected transmit output pin, also with DX1CR.INSW = 0. Additionally, program DX2CR.INSW = 0.
  Due to the handling of the input data stream by the synchronous protocol handler, the propagation delay of the synchronization in the input stage has to be considered.
- Bit timing configuration:
  The desired baud rate setting has to be selected, comprising the fractional divider, the baud rate generator and the bit timing. Please note that not all feature combinations can be supported by the application at the same time, e.g. due to propagation delays. For example, the length of a frame is limited by the frequency difference of the transmitter and the receiver device. Furthermore, in order to use the average of samples (SMD = 1), the sampling point has to be chosen to respect the signal settling and data propagation times.
- Data format configuration:
  The word length, the frame length, and the shift direction have to be set up according to the application requirements by programming the registers SCTRL and SCTRH. If required by the application, the data input and output signals can be inverted.
  Additionally, the parity mode has to be configured (CCR.PM).

19.3.3.1 Bit Timing

In ASC mode, each bit (incl. protocol bits) is divided into time quanta in order to provide granularity in the sub-bit range to adjust the sample point to the application requirements. The number of time quanta per bit is defined by bit fields BRGL.DCTQ and the length of a time quantum is given by BRGL.PCTQ.
In the example given in **Figure 19-27**, one bit time is composed of 16 time quanta (BRGL.DCTQ = 15). It is not recommended to program less than 4 time quanta per bit time.
Bit field PCRL.SP determines the position of the sampling point for the bit value. The value of PCRL.SP must not be set to a value greater than BRGL.DCTQ. It is possible to sample the bit value only once per bit time or to take the average of samples. Depending on bit PCRL.SMD, either the current input value is directly sampled as bit value, or a majority decision over the input values sampled at the latest three time quanta is taken into account. The standard ASC bit timing consists of 16 time quanta with sampling after 8 or 9 time quanta with majority decision.
The bit timing setup (number of time quanta and the sampling point definition) is common for the transmitter and the receiver. Due two independent bit timing blocks, the receiver and the transmitter can be in different time quanta or bit positions inside their frames. The transmission of a frame is aligned to the time quanta generation.

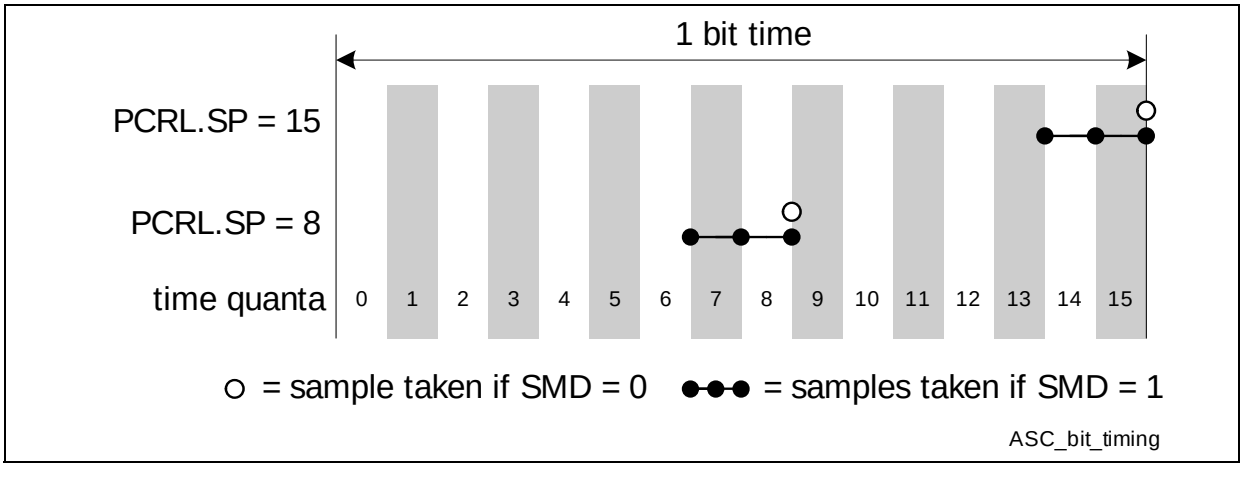


Figure 19-27 ASC Bit Timing

The sample point setting has to be adjusted carefully if collision or idle detection is enabled (via DX1 input signal), because the driver delay and some external delays have to be taken into account. The sample point for the transmit line has to be set to a value where the bit level is stable enough to be evaluated.
If the sample point is located late in the bit time, the signal itself has more time to become stable, but the robustness against differences in the clock frequency of transmitter and receiver decreases.

19.3.3.2 Baud Rate Generation

The baud rate f_{ASC} in ASC mode depends on the number of time quanta per bit time and their timing. The baud rate setting should only be changed while the transmitter and the receiver are idle. The bits in register BRGL define the baud rate setting:

- BRGL.CTQSEL
  to define the input frequency f_{CTQIN} for the time quanta generation
- BRGL.PCTQ
  to define the length of a time quantum (division of f_{CTQIN} by 1, 2, 3, or 4)
- BRGL.DCTQ
  to define the number of time quanta per bit time

The standard setting is given by CTQSEL = 00_B ($f_{CTQIN} = f_{PDIV}$) and PPPEN = 0 ($f_{PPP} = f_{PIN}$). Under these conditions, the baud rate is given by:

(19.6)

$$f_{ASC} = f_{PIN} \times \frac{1}{PDIV + 1} \times \frac{1}{PCTQ + 1} \times \frac{1}{DCTQ + 1}$$

In order to generate slower frequencies, two additional divide-by-2 stages can be selected by CTQSEL = 10_B ($f_{CTQIN} = f_{SCLK}$) and PPPEN = 1 ($f_{PPP} = f_{MCLK}$), leading to:

(19.7)

$$f_{ASC} = \frac{f_{PIN}}{2 \times 2} \times \frac{1}{PDIV + 1} \times \frac{1}{PCTQ + 1} \times \frac{1}{DCTQ + 1}$$

19.3.3.3 Noise Detection

The ASC receiver permanently checks the data input line of the DX0 stage for noise (the check is independent from the setting of bit PCRL.SMD). Bit PSR.RNS (receiver noise) becomes set if the three input samples of the majority decision are not identical at the sample point for the bit value. The information about receiver noise gets accumulated over several bits in bit PSR.RNS (it has to be cleared by SW) and can trigger a protocol interrupt each time noise is detected if enabled by PCRL.RNIEN.

19.3.3.4 Collision Detection

In some applications, such as data transfer over a single data line shared by several sending devices (see **Figure 19-25**), several transmitters have the possibility to send on the same data output line TxD. In order to avoid collisions of transmitters being active at the same time or to allow a kind of arbitration, a collision detection has been implemented.
The data value read at the TxD input at the DX1 stage and the transmitted data bit value are compared after the sampling of each bit value. If enabled by PCRL.CDEN = 1 and a bit sent is not equal to the bit read back, a collision is detected and bit PSR.COL is set.

If enabled, bit PSR.COL = 1 disables the transmitter (the data output lines become 1) and generates a protocol interrupt. The content of the transmit shift register is considered as invalid, so the transmit buffer has to be programmed again.

19.3.3.5 Pulse Shaping

For some applications, the 0 level of transmitted bits with the bit value 0 is not applied at the transmit output during the complete bit time. Instead of driving the original 0 level, only a 0 pulse is generated and the remaining time quanta of the bit time are driven with 1 level. The length of a bit time is not changed by the pulse shaping, only the signalling is changed.
In the standard ASC signalling scheme, the 0 level is signalled during the complete bit time with bit value 0 (ensured by programming PCRH.PL = 000_B). In the case PCRH.PL > 000_B, the transmit output signal becomes 0 for the number of time quanta defined by PCRH.PL. In order to support correct reception with pulse shaping by the transmitter, the sample point has to be adjusted in the receiver according to the applied pulse length.

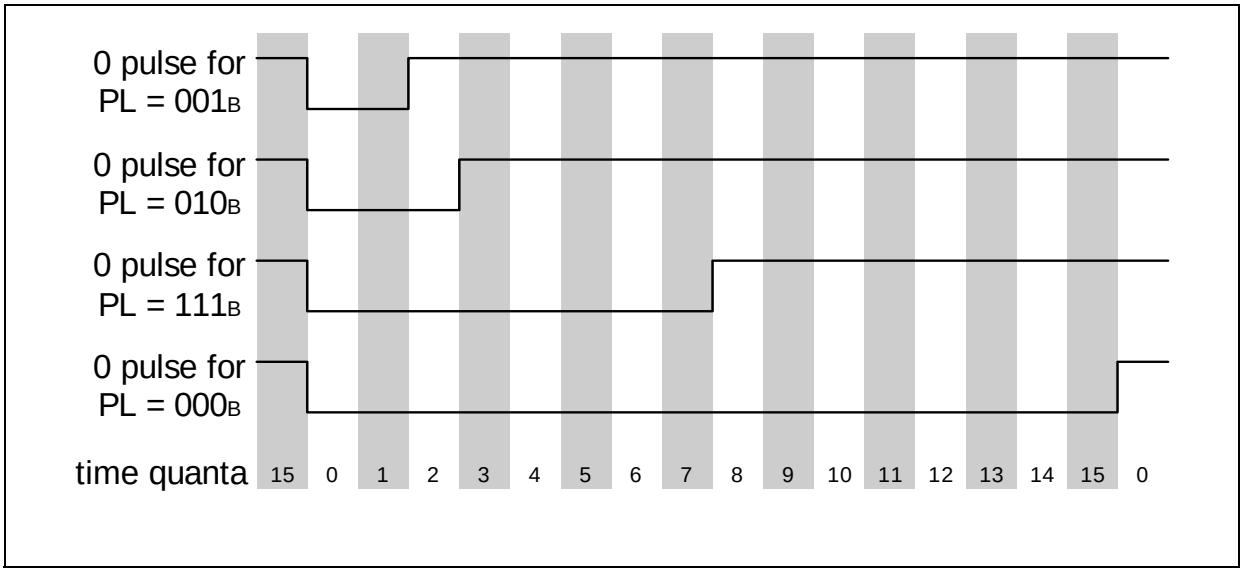


Figure 19-28 Transmitter Pulse Length Control

Figure 19-29 shows an example for the transmission of an 8 bit data word with LSB first and one stop bit (e.g. like for IrDA). The polarity of the transmit output signal has been inverted by SCTRL.DOCFG = 01_B.

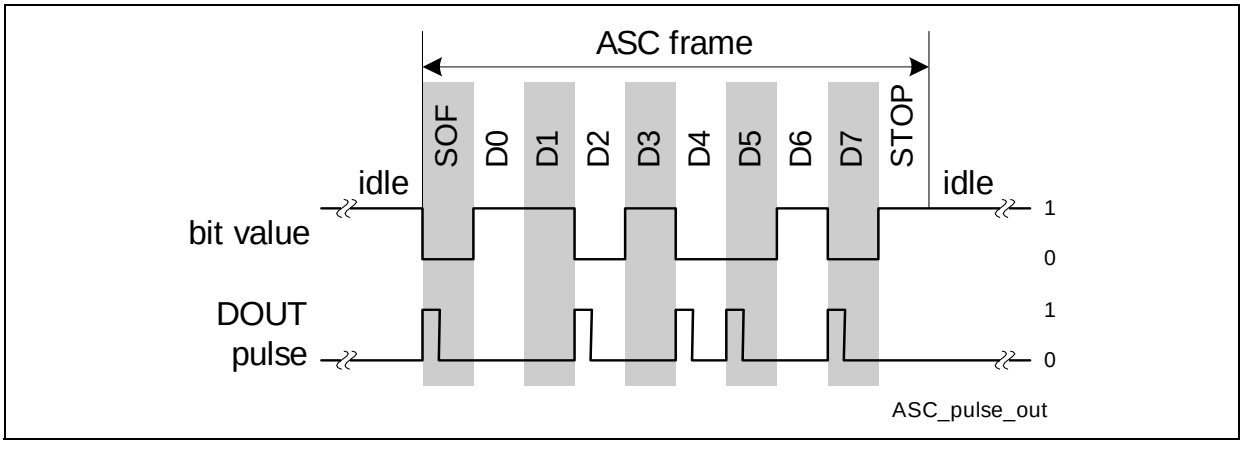


Figure 19-29 Pulse Output Example

19.3.3.6 Automatic Shadow Mechanism

The contents of the protocol control registers PCRL and PCRH, as well as bit field SCTRH.FLE are internally kept constant while a data frame is transferred by an automatic shadow mechanism (shadowing takes place with each frame start). The registers can be programmed all the time with new settings that are taken into account for the next data frame. During a data frame, the applied (shadowed) setting is not changed, although new values have been written after the start of the data frame.
Bit fields SCTRH.WLE and SCTRL.SDIR are shadowed automatically with the start of each data word. As a result, a data frame can consist of data words with a different length. It is recommended to change SCTRL.SDIR only when no data frame is running to avoid interference between HW and SW.
Please note that the starting point of a data word can be different for a transmitter and a receiver. In order to ensure correct handling, it is recommended to modify SCTRH.WLE only while transmitter and receiver are both idle. If the transmitter and the receiver are referring to the same data signal (e.g. in a LIN bus system), SCTRH.WLE can be modified while a data transfer is in progress after the RSI event has been detected.

19.3.3.7 End of Frame Control

The number of bits per ASC frame is defined by bit field SCTRH.FLE. In order to support different frame length settings for consecutively transmitted frames, this bit field can be modified by HW. The automatic update mechanism is enabled by TCSRL.FLEMD = 1 (in this case, bits TCSRL.WLEMD, SELMD, and WAMD have to be cleared).
If enabled, the transmit control information TCI automatically overwrites the bit field TCSRL.FLEMD when the ASC frame is started (leading to frames with 1 to 32 data bits). The TCI value represents the written address location of TBUFxx (without additional data buffer) or INxx (with additional data buffer). With this mechanism, an ASC with 8 data bits is generated by writing a data word to TBUF07 (IN07, respectively).

19.3.3.8 Mode Control Behavior

In ASC mode, the following kernel modes are supported:

- Run Mode 0/1:
  Behavior as programmed, no impact on data transfers.
- Stop Mode 0:
  Bit PSR.TXIDLE is cleared. A new transmission is not started. A current transmission is finished normally. Bit PSR.RXIDLE is not modified. Reception is still possible.
  When leaving stop mode 0, bit TXIDLE is set according to PCR.IDM.
- Stop Mode 1:
  Bit PSR.TXIDLE is cleared. A new transmission is not started. A current transmission is finished normally. Bit PSR.RXIDLE is cleared. A new reception is not possible. A current reception is finished normally.
  When leaving stop mode 1, bits TXIDLE and RXIDLE are set according to PCR.IDM.

19.3.3.9 Disabling ASC Mode

In order to switch off ASC mode without any data corruption, the receiver and the transmitter have to be both idle. This is ensured by requesting Stop Mode 1 in register KSCFG. After waiting for the end of the frame, the ASC mode can be disabled.

19.3.3.10 Protocol Interrupt Events

The following protocol-related events are generated in ASC mode and can lead to a protocol interrupt. The collision detection and the transmitter frame finished events are related to the transmitter, whereas the receiver events are given by the synchronization break detection, the receiver noise detection, the format error checks and the end of the received frame.
Please note that the bits in register PSR are not automatically cleared by HW and have to be cleared by SW in order to monitor new incoming events.

- Collision detection:
  This interrupt indicates that the transmitted value (DOUT) does not match with the input value of the DX1 input stage at the sample point of a bit. For more details refer to **Section 19.3.3.4**.
- Transmitter frame finished:
  This interrupt indicates that the transmitter has completely finished a frame. Bit PSR.TFF becomes set at the end of the last stop bit. The DOUT signal assignment to port pins can be changed while no transmission is in progress.
- Receiver frame finished:
  This interrupt indicates that the receiver has completely finished a frame. Bit PSR.RFF becomes set at the end of the last stop bit. The DIN signal assignment to port pins can be changed while no reception is in progress.

- Synchronization break detection:
  This interrupt can be used in LIN networks to indicate the reception of the synchronization break symbol (at the beginning of a LIN frame).
- Receiver noise detection:
  This interrupt indicates that the input value at the sample point of a bit and at the two time quanta before are not identical.
- Format error:
  The bit value of the stop bit(s) is defined as 1 level for the ASC protocol. A format error is signalled if the sampled bit value of a stop bit is 0.

19.3.3.11 Data Transfer Interrupt Handling

The data transfer interrupts indicate events related to ASC frame handling.

- Transmit buffer interrupt TBI:
  Bit PSR.TBIF is set after the start of first data bit of a data word. This is the earliest point in time when a new data word can be written to TBUF.
  With this event, bit **TCSRL**.TDV is cleared and new data can be loaded to the transmit buffer.
- Transmit shift interrupt TSI:
  Bit PSR.TSIF is set after the start of the last data bit of a data word.
- Receiver start interrupt RSI:
  Bit PSR.RSIF is set after the sample point of the first data bit of a data word.
- Receiver interrupt RI and alternative interrupt AI:
  Bit PSR.RIF is set after the sampling point of the last data bit of a data word if this data word is not directly followed by a parity bit (parity generation disabled or not the last word of a data frame).
  If the data word is directly followed by a parity bit (last data word of a data frame and parity generation enabled), bit PSR.RIF is set after the sampling point of the parity bit if no parity error has been detected. If a parity error has been detected, bit PSR.AIF is set instead of bit PSR.RIF.
  The first data word of a data frame is indicated by RBUFSR.SOF = 1 for the received word.

19.3.3.12 Protocol-Related Argument and Error

The protocol-related argument (RBUFSR.PAR) and the protocol-related error (RBUFSR.PERR) are two flags that are assigned to each received data word in the corresponding receiver buffer status registers.
In ASC mode, the received parity bit is monitored by the protocol-related argument and the result of the parity check by the protocol-related error indication (0 = received parity bit equal to calculated parity value). This information being elaborated only for the last received data word of each data frame, both bit positions are 0 for data words that are not the last data word of a data frame or if the parity generation is disabled.

19.3.3.13 Receive Buffer Handling

If a receive FIFO buffer is available (CCFG.RB = 1) and enabled for data handling (RBCTRH.SIZE > 0), it is recommended to set RBCTRH.RCIM = 11_B in ASC mode. This leads to an indication that the data word has been the first data word of a new data frame if bit OUTRH.RCI[0] = 1, a parity error is indicated by OUTRH.RCI[4] = 1, and the received parity bit value is given by OUTRH.RCI[3].
The standard receive buffer event and the alternative receive buffer event can be used for the following operations in RCI mode (RBCTRH.RNM = 1):

- A standard receive buffer event indicates that a data word can be read from OUTRL that has been received without parity error.
- An alternative receive buffer event indicates that a data word can be read from OUTRL that has been received with parity error.

19.3.3.14 Sync-Break Detection

The receiver permanently checks the DIN signal for a certain number of consecutive bit times with 0 level. The number is given by the number of programmed bits per frame (SCTRH.FLE) plus 2 (in the case without parity) or plus 3 (in the case with parity). If a 0 level is detected at a sample point of a bit after this event has been found, bit PSR.SBD is set and additionally, a protocol interrupt can be generated (if enabled by PCRL.SBD = 1). The counting restarts from 0 each time a falling edge is found at input DIN. This feature can be used for the detection of a synchronization break for slave devices in a LIN bus system (the master doesn't check for sync break).
For example, in a configuration for 8 data bits without parity generation, bit PCRL.SBD is set after at the next sample point at 0 level after 10 complete bit times have elapsed (representing the sample point of the 11th bit time since the first falling edge).

19.3.4 ASC Protocol Registers

In ASC mode, the registers PCRH, PCRL and PSR handle ASC related information.

19.3.4.1 Protocol Control Registers

The following PCRL/PCRH control bits or bit fields are available in ASC mode.

Table 19-16 Signification of PCRL/PCRH Bits for ASC

Bit	Alias	Description
CTR[0]	**SMD**	**Sample Mode** This bit field defines the sample mode of the ASC receiver. The selected data input signal can be sampled only once per bit time or three times (in consecutive time quanta). When sampling three times, the bit value shifted in the receiver shift register is given by a majority decision among the three sampled values. 0_B Only one sample is taken per bit time. The current input value is sampled. 1_B Three samples are taken per bit time and a majority decision is made.
CTR[1]	**STPB**	**Stop Bits** This bit defines the number of stop bits in an ASC frame. 0_B The number of stop bits is 1. 1_B The number of stop bits is 2.
CTR[2]	**IDM**	**Idle Detection Mode** This bit defines if the idle detection is switched off or based on the frame length. 0_B The bus idle detection is switched off and bits PSR.TXIDLE and PSR.RXIDLE are set automatically to enable data transfers without checking the inputs before. 1_B The bus is considered as idle after a number of consecutive passive bit times defined by SCTRH.FLE plus 2 (in the case without parity bit) or plus 3 (in the case with parity bit).
CTR[3]	**SBIEN**	**Synchronization Break Interrupt Enable** This bit enables the generation of a protocol interrupt if a synchronization break is detected. The automatic detection is always active, so bit SBD can be set independently of SBIEN. 0_B The interrupt generation is disabled. 1_B The interrupt generation is enabled.

Table 19-16 Signification of PCRL/PCRH Bits for ASC (cont'd)

Bit	Alias	Description
CTR[4]	**CDEN**	**Collision Detection Enable** This bit enables the reaction of a transmitter to the collision detection. 0_B The collision detection is disabled. 1_B If a collision is detected, the transmitter stops its data transmission, outputs a 1, sets bit PSR.COL and generates a protocol interrupt. In order to allow data transmission again, PSR.COL has to be cleared by SW.
CTR[5]	**RNIEN**	**Receiver Noise Detection Interrupt Enable** This bit enables the generation of a protocol interrupt if receiver noise is detected. The automatic detection is always active, so bit PSR.RNS can be set independently of PCRL.RNIEN. 0_B The interrupt generation is disabled. 1_B The interrupt generation is enabled.
CTR[6]	**FEIEN**	**Format Error Interrupt Enable** This bit enables the generation of a protocol interrupt if a format error is detected. The automatic detection is always active, so bits PSR.FER0/FER1 can be set independently of PCRL.FEIEN. 0_B The interrupt generation is disabled. 1_B The interrupt generation is enabled.
CTR[7]	**FFIEN**	**Frame Finished Interrupt Enable** This bit enables the generation of a protocol interrupt if the receiver or the transmitter reach the end of a frame. The automatic detection is always active, so bits PSR.RFF or PSR.TFF can be set independently of PCRL.FFIEN. 0_B The interrupt generation is disabled. 1_B The interrupt generation is enabled.
CTR [12:8]	**SP**	**Sample Point** This bit field defines the sample point of the bit value. The sample point must not be located outside the programmed bit timing (PCRL.SP ≤ BRGL.DCTQ).

Table 19-16 Signification of PCRL/PCRH Bits for ASC (cont'd)

Bit	Alias	Description
CTR [15:13]	**PL**	**Pulse Length** This bit field defines the length of a 0 data bit, counted in time quanta, starting with the time quantum 0 of each bit time. Each bit value that is a 0 can lead to a 0 pulse that is shorter than a bit time, e.g. for IrDA applications. The length of a bit time is not changed by PL, only the length of the 0 at the output signal. The pulse length must not be longer than the programmed bit timing (PCRH.PL ≤ BRGL.DCTQ). This bit field is only taken into account by the transmitter and is ignored by the receiver. 000_B The pulse length is equal to the bit length (no shortened 0). 001_B The pulse length of a 0 bit is 2 time quanta. 010_B The pulse length of a 0 bit is 3 time quanta. ... 111_B The pulse length of a 0 bit is 8 time quanta.
CTR[31]	**MCLK**	**Master Clock Enable** This bit enables the generation of the master clock MCLK. 0_B The MCLK generation is disabled and the MCLK signal is 0. 1_B The MCLK generation is enabled.
other CTR[x]		**Reserved** returns 0 if read; should be written with 0;

19.3.4.2 Protocol Status Register PSR

The following PSR status bits or bit fields are available in ASC mode. Please note that the bits in register PSR are not cleared by HW.
The row marked PI indicates which status bit can generate a protocol interrupt. The general interrupt status flags are described in the general interrupt chapter.

Table 19-17 Signification of PSR Bits for ASC

Bit	Alias	Description	PI?
ST[0]	**TXIDLE**	**Transmission Idle** This bit shows if the transmit line (DX1) has been idle. A frame transmission can only be started if TXIDLE is set. 0_B The transmitter line has not yet been idle. 1_B The transmitter line has been idle and frame transmission is possible.	n
ST[1]	**RXIDLE**	**Reception Idle** This bit shows if the receive line (DX0) has been idle. A frame reception can only be started if RXIDLE is set. 0_B The receiver line has not yet been idle. 1_B The receiver line has been idle and frame reception is possible.	n
ST[2]	**SBD**	**Synchronization Break Detected** This bit is set if a programmed number of consecutive bit values with level 0 has been detected (called synchronization break, e.g. in a LIN bus system). 0_B A synchronization break has not yet been detected. 1_B A synchronization break has been detected.	y
ST[3]	**COL**	**Collision Detected** This bit is set if a collision has been detected (with PCRL.CDEN = 1). 0_B A collision has not yet been detected and frame transmission is possible. 1_B A collision has been detected and frame transmission is not possible.	y
ST[4]	**RNS**	**Receiver Noise Detected** This bit is set if receiver noise has been detected. 0_B Receiver noise has not been detected. 1_B Receiver noise has been detected.	y

Table 19-17 Signification of PSR Bits for ASC (cont'd)

Bit	Alias	Description	PI?
ST[5]	**FER0**	**Format Error in Stop Bit 0** This bit is set if a 0 has been sampled in the stop bit 0 (called format error 0). 0_B A format error 0 has not been detected. 1_B A format error 0 has been detected.	y
ST[6]	**FER1**	**Format Error in Stop Bit 1** This bit is set if a 0 has been sampled in the stop bit 1 (called format error 1). 0_B A format error 1 has not been detected. 1_B A format error 1 has been detected.	y
ST[7]	**RFF**	**Receive Frame Finished** This bit is set if the receiver has finished the last stop bit. 0_B The received frame is not yet finished. 1_B The received frame is finished.	y
ST[8]	**TFF**	**Transmitter Frame Finished** This bit is set if the transmitter has finished the last stop bit. 0_B The transmitter frame is not yet finished. 1_B The transmitter frame is finished.	y
PSR [15:10]		**General Interrupt Flags**	n

19.3.5 Hardware LIN Support

The Local Interconnect Network (LIN) data exchange protocol contains several symbols that can all be handled in ASC mode. Each single LIN symbol represents a complete ASC frame. The LIN bus is a master-slave bus system with a single master and multiple slaves (for the exact definition please refer to the official LIN specification).
In order to support the LIN protocol, bit TCSRL.FLEMD = 1 should be set for the master. For slave devices, it can be cleared and the fixed number of 8 data bits has to be set (SCTRH.FLE = 7). For both, master and slave devices, the parity generation has to be switched off (CCR.PM = 00_B) and transfers take place with LSB first (SCTRL.SDIR = 0) and 1 stop bit (PCRL.STPB = 0).
A complete LIN frame contains the following symbols:

- Synchronization break:
  The master sends a synchronization break to signal the beginning of a new frame. It contains at least 13 consecutive bit times at 0 level, followed by at least one bit time at 1 level (corresponding to 1 stop bit). Therefore, TBUF11 (or IN11) has to be written with 0 (leading to a frame with SOF followed by 12 data bits at 0 level).
  A slave device shall detect 11 consecutive bit times at 0 level, which done by the synchronization break detection. Bit PSR.SBD is set if such an event is detected and a protocol interrupt can be generated. Additionally, the received data value of 0 appears in the receive buffer and a format error is signaled.
  If the baud rate of the slave has to be adapted to the master, the baud rate measurement has to be enabled for falling edges by setting BRGL.TMEN = 1, DX0CR.CM = 10_H and DX1CR.CM = 00_H before the next symbol starts.
- Synchronization byte:
  The master sends this symbol after writing the data value 55_H to TBUF07 (or IN07).
  A slave device can either receive this symbol without any further action (and can discard it) or it can use the falling edges for baud rate measurement. Bit PSR.TSIF = 1 (with optionally the corresponding interrupt) indicates the detection of a falling edge and the capturing of the elapsed time since the last falling edge in BRGH.PDIV. Valid captured values can be read out after the second, third, fourth and fifth activation of TSIF. After the fifth activation of TSIF within this symbol, the baud rate detection has to be disabled (BRGL.TMEN = 0) and BRGH.PDIV can be programmed with the formerly captured value divided by twice the number of time quanta per bit (assuming BRGL.PCTQ = 00_B).
- Other symbols:
  The other symbols of a LIN frame can be handled with ASC data frames without specific actions.

If LIN frames should be sent out on a frame base by the LIN master, the input DX2 can be connected to external timers to trigger the transmit actions (e.g. the synchronization break symbol has been prepared but is started if a trigger occurs).
Please note that during the baud rate measurement of the ASC receiver, no transmission can take place by the ASC transmitter of the same USIC channel.

19.4 Synchronous Serial Channel (SSC)

The synchronous serial channel SSC covers the data transfer function of an SPI-like module. It can handle reception and transmission of synchronous data frames between a device operating in master mode and at least one device in slave mode. The SSC mode is selected by CCR.MODE = 0001_B with CCFG.SSC = 1 (SSC mode is available).

This chapter contains the following sections:

- Signal description (see **Section 19.4.1**)
- General SSC issues (see **Section 19.4.2**)
- Master mode operation (see **Section 19.4.3**)
- Slave mode operation (see **Section 19.4.4**)
- Protocol registers (see **Section 19.4.5**)
- Timing considerations (see **Section 19.4.6**)

19.4.1 Signal Description

A synchronous SSC data transfer is characterized by a simultaneous transfer of a shift clock signal together with the transmit and/or receive data signal(s) to determine when the data is valid (definition of transmit and sample point).

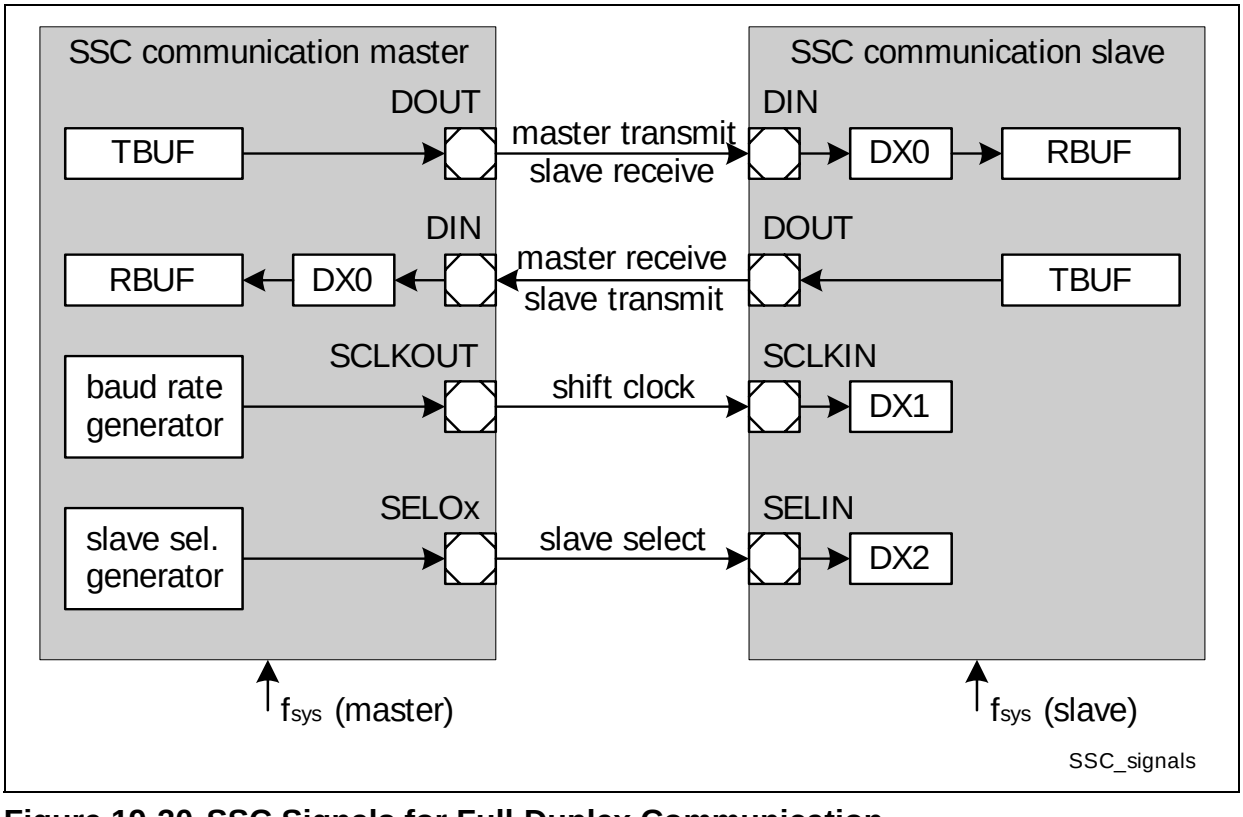


Figure 19-30 SSC Signals for Full-Duplex Communication

In order to explicitly indicate the start and the end of a data transfer and to address more than one slave devices individually, the SSC module supports the handling of slave select signals. They are optional and are not necessarily needed for SSC data transfers. The SSC module supports up to 8 different slave select output signals for master mode operation (named SELOx, with x = 0-7) and 1 slave select input SELIN for slave mode. In most applications, the slave select signals are active low.
A device operating in master mode controls the start and end of a data frame, as well as the generation of the shift clock and slave select signals. This comprises the baud rate setting for the shift clock and the delays between the shift clock and the slave select output signals. If several SSC modules are connected together, there can be only one SSC master at a time, but several slaves. Slave devices receive the shift clock and optionally a slave select signal(s). For the programming of the input stages DX0, DX1, and DX2 please refer to **Section 19.2.5**.

Table 19-18 SSC Communication Signals

SSC Mode	Receive Data	Transmit Data	Shift Clock	Slave Select(s)
master	MRST[1], input DIN, handled by DX0	MTSR[2], output DOUT	output SCLKOUT	output(s) SELOx
slave	MTSR, input DIN, handled by DX0	MRST, output DOUT	input SCLKIN, handled by DX1	input SELIN, handled by DX2

1) MRST = master receive slave transmit, also known as MISO = master in slave out

2) MTSR = master transmit slave receive, also known as MOSI = master out slave in

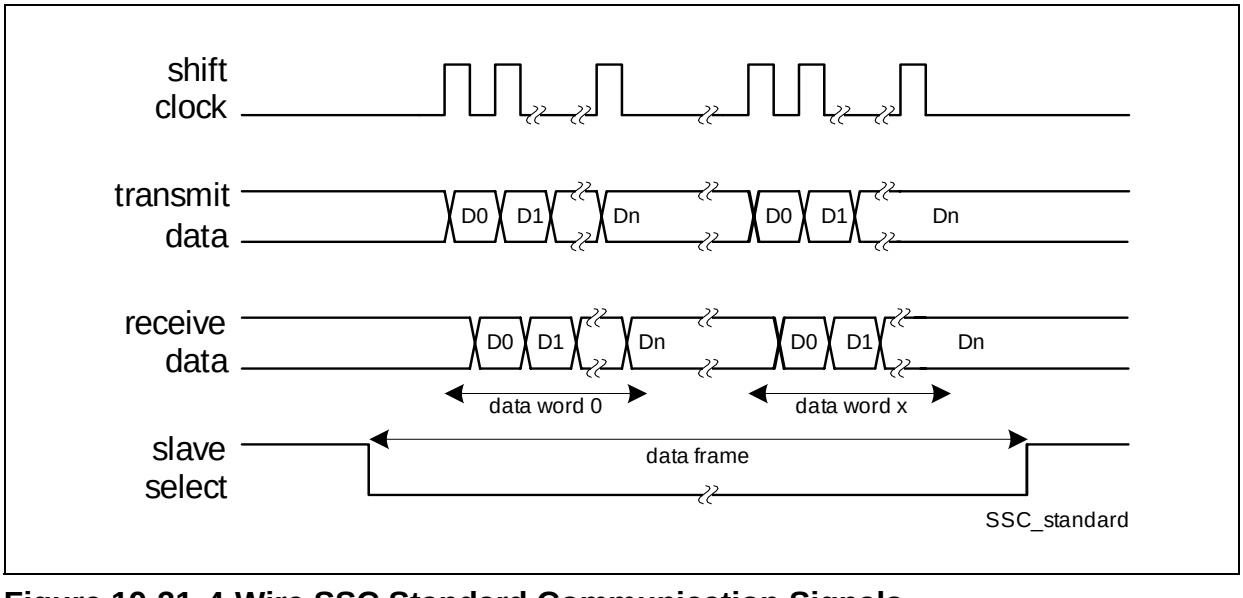


Figure 19-31 4-Wire SSC Standard Communication Signals

19.4.1.1 Transmit and Receive Data Signals

In half-duplex mode, a single data line is used, either for data transfer from the master to a slave or from a slave to the master. In this case, MRST and MTSR are connected together, one signal as input, the other one as output, depending on the data direction. The user SW has to take care about the data direction to avoid data collision (e.g. by preparing dummy data of all 1s for transmission in case of a wired AND connection with open-drain drivers or by enabling/disabling push/pull output drivers). In full-duplex mode, data transfers take place in parallel between the master device and a slave device via two independent data signals MTSR and MRST, as shown in **Figure 19-30**.
The receive data input signal DIN is handled by the input stage DX0. In master mode (referring to MRST) as well as in slave mode (referring to MTSR), the data input signal DIN is taken from an input pin. The signal polarity of DOUT (data output) with respect to the data bit value can be configured in block DOCFG (data output configuration) by bit field SCTRL.DOCFG.

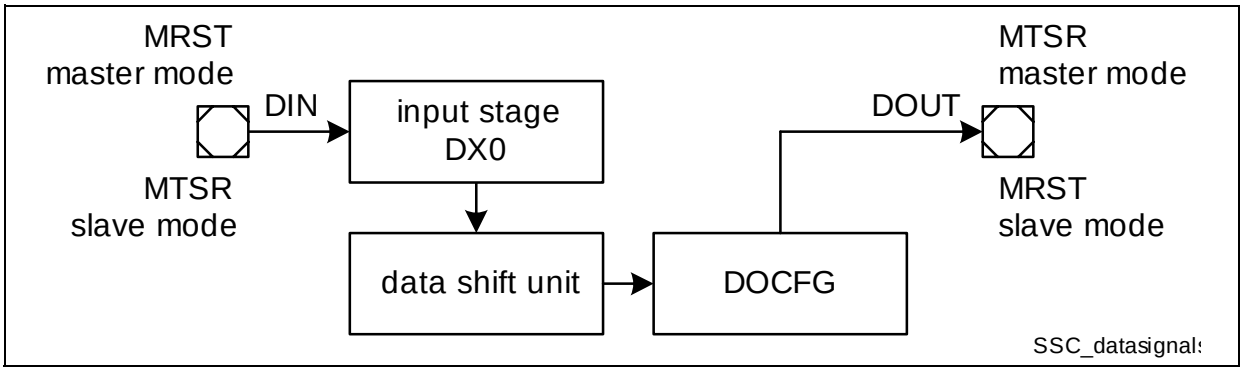


Figure 19-32 SSC Data Signals

19.4.1.2 Shift Clock Signals

The shift clock signal is handled by the input stage DX1. In slave mode, the signal SCLKIN is received from an external master, so the DX1 stage has to be connected to an input pin. The input stage can invert the received input signal to adapt to the polarity of SCLKIN to the function of the data shift unit (data transmission on rising edges, data reception on falling edges).
In master mode, the shift clock is generated by the internal baud rate generator. The output signal SCLK of the baud rate generator is taken as shift clock input for the data shift unit. The internal signal SCLK is made available for external slave devices by signal SCLKOUT.

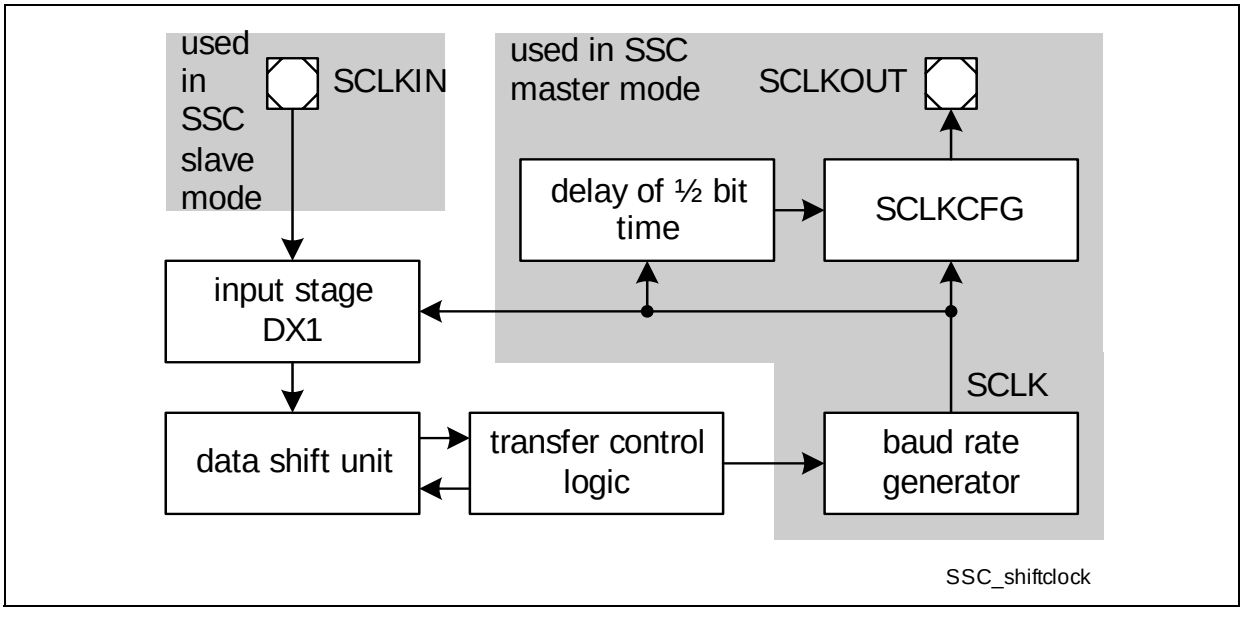


Figure 19-33 SSC Shift Clock Signals

Due to the multitude of different SSC applications, in master mode, there are different ways to configure the shift clock output signal SCLKOUT with respect to SCLK. This is done in the block SCLKCFG (shift clock configuration) by bit field BRGH.SCLKCFG, allowing 4 possible settings, as shown in **Figure 19-34**.

- No delay, no polarity inversion (SCLKCFG = 00_B, SCLKOUT equals SCLK):
  The inactive level of SCLKOUT is 0, while no data frame is transferred. The first data bit of a new data frame is transmitted with the first rising edge of SCLKOUT and the first data bit is received in with the first falling edge of SCLKOUT. The last data bit of a data frame is transmitted with the last rising clock edge of SCLKOUT and the last data bit is received in with the last falling edge of SCLKOUT. This setting can be used in master and in slave mode. It corresponds to the behavior of the internal data shift unit.
- No delay, polarity inversion (SCLKCFG = 01_B):
  The inactive level of SCLKOUT is 1, while no data frame is transferred. The first data

bit of a new data frame is transmitted with the first falling clock edge of SCLKOUT and the first data bit is received with the first rising edge of SCLKOUT. The last data bit of a data frame is transmitted with the last falling edge of SCLKOUT and the last data bit is received with the last rising edge of SCLKOUT. This setting can be used in master and in slave mode.

- SCLKOUT is delayed by 1/2 shift clock period, no polarity inversion (SCLKCFG = 10_B):
  The inactive level of SCLKOUT is 0, while no data frame is transferred.
  The first data bit of a new data frame is transmitted 1/2 shift clock period before the first rising clock edge of SCLKOUT. Due to the delay, the next data bits seem to be transmitted with the falling edges of SCLKOUT. The last data bit of a data frame is transmitted 1/2 period of SCLKOUT before the last rising clock edge of SCLKOUT. The first data bit is received 1/2 shift clock period before the first falling edge of SCLKOUT. Due to the delay, the next data bits seem to be received with the rising edges of SCLKOUT. The last data bit is received 1/2 period of SCLKOUT before the last falling clock edge of SCLKOUT.
  This setting can be used only in master mode and not in slave mode (the connected slave has to provide the first data bit before the first SCLKOUT edge, e.g. as soon as it is addressed by its slave select).
- SCLKOUT is delayed by 1/2 shift clock period, polarity inversion (SCLKCFG = 11_B):
  The inactive level of SCLKOUT is 1, while no data frame is transferred.
  The first data bit of a new data frame is transmitted 1/2 shift clock period before the first falling clock edge of SCLKOUT. Due to the delay, the next data bits seem to be transmitted with the rising edges of SCLKOUT. The last data bit of a data frame is transmitted 1/2 period of SCLKOUT before the last falling clock edge of SCLKOUT. The first data bit is received 1/2 shift clock period before the first rising edge of SCLKOUT. Due to the delay, the next data bits seem to be received with the falling edges of SCLKOUT. The last data bit is received 1/2 period of SCLKOUT before the last rising clock edge of SCLKOUT.
  This setting can be used only in master mode and not in slave mode (the connected slave has to provide the first data bit before the first SCLKOUT edge, e.g. as soon as it is addressed by its slave select).

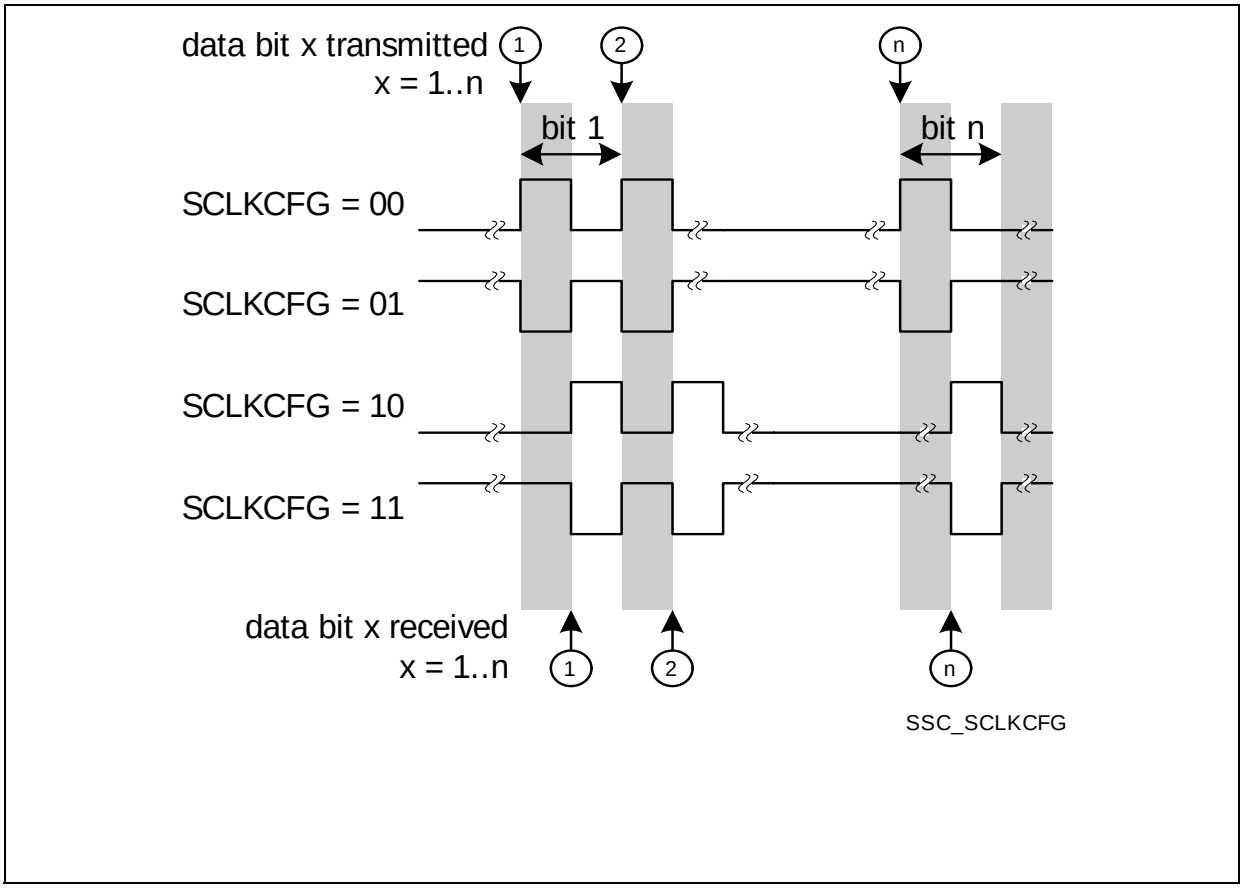


Figure 19-34 SCLKOUT Configuration in SSC Master Mode

Note: If a configuration with delay is selected and a slave select line is used, the slave select delays have to be set up accordingly.

19.4.1.3 Slave Select Signals

The slave select signal is handled by the input stage DX2. In slave mode, the input signal SELIN is received from an external master via an input pin. The input stage can invert the received input signal to adapt the polarity of signal SELIN to the function of the data shift unit (the module internal signals are considered as high active, so a data transfer is only possible while the slave select input of the data shift unit is at 1-level, otherwise, shift clock pulses are ignored and do not lead to data transfers). If an input signal SELIN is low active, it should be inverted in the DX2 input stage.
In master mode, a master slave select signal MSLS is generated by the internal slave select generator. In order to address different external slave devices independently, the internal MSLS signal is made available externally via up to 8 SELOx output signals that can be configured by the block SELCFG (select configuration).

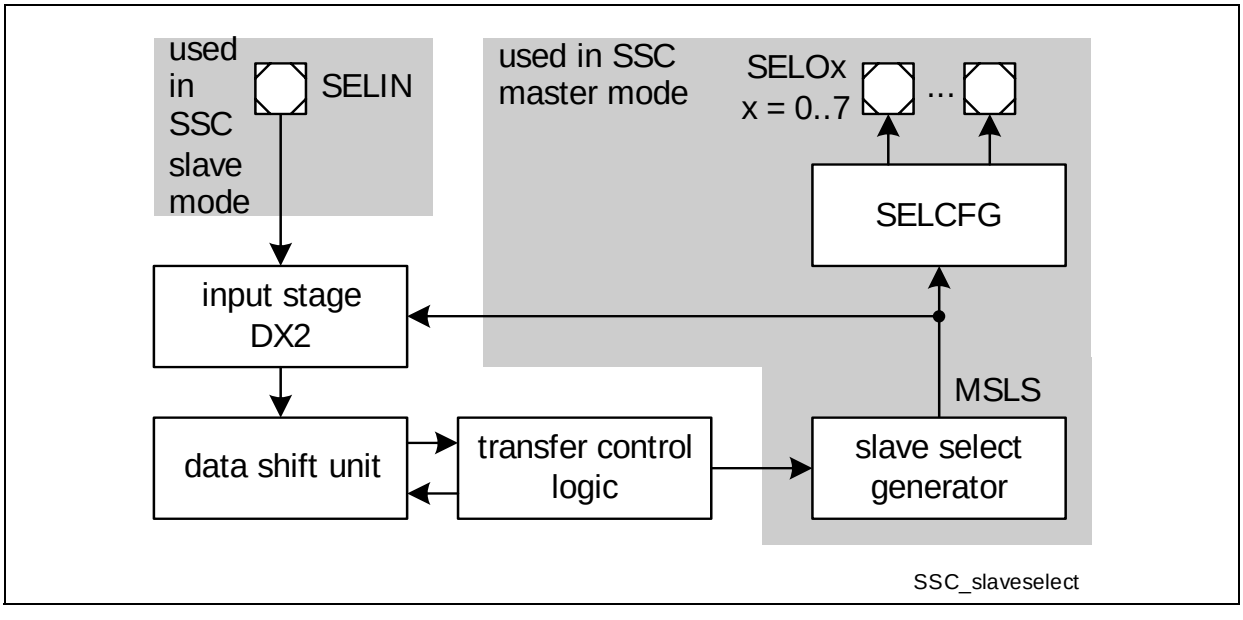


Figure 19-35 SSC Slave Select Signals

The control of the SELCFG block is based on protocol specific bits and bit fields in the protocol control register parts PCRL and PCRH. For the generation of the MSLS signal please refer to **Section 19.4.3.2**.

- PCRL.SELCTR to chose between direct and coded select mode
- PCRL.SELINV to invert the SELOx outputs
- PCRH.SELO[7:0] as individual value for each SELOx line

The SELCFG block supports the following configurations of the SELOx output signals:

- Direct Select Mode (SELCTR = 1):
  Each SELOx line (with x = 0-7) can be directly connected to an external slave device. If bit x in bit field SELO is 0, the SELOx output is permanently inactive. A SELOx output becomes active while the internal signal MSLS is active (see **Section 19.4.3.2**) and bit x in bit field SELO is 1. Several external slave devices can

be addressed in parallel if more than one bit in bit field SELO are set during a data frame. The number of external slave devices that can be addressed individually is limited to the number of available SELOx outputs.

- Coded Select Mode (SELCTR = 0):
  The SELOx lines (with x = 1-7) can be used as addresses for an external address decoder to increase the number of external slave devices. These lines only change with the start of a new data frame and have no other relation to MSLS. Signal SELO0 can be used as enable signal for the external address decoder. It is active while MSLS is active (during a data frame) and bit 0 in bit field SELO is 1. Furthermore, in coded select mode, this output line is delayed by one cycle of f_{SYS} compared to MSLS to allow the other SELOx lines to stabilize before enabling the address decoder.

19.4.2 Operating the SSC

This chapter contains SSC issues, that are of general interest and not directly linked to either master mode or slave mode.

19.4.2.1 Automatic Shadow Mechanism

The contents of the baud rate control registers BRGL and BRGH, bit field SCTRH.FLE as well as the protocol control registers PCRL and PCRH are internally kept constant while a data frame is transferred (= while MSLS is active) by an automatic shadow mechanism. The registers can be programmed all the time with new settings that are taken into account for the next data frame. During a data frame, the applied (shadowed) setting is not changed, although new values have been written after the start of the data frame.
Bit fields SCTRH.WLE and SCTRL.SDIR are shadowed automatically with the start of each data word. As a result, a data frame can consist of data words with a different length. It is recommended to change SCTRL.SDIR only when no data frame is running to avoid interference between HW and SW.
Please note that the starting point of a data word are different for a transmitter (first bit transmitted) and a receiver (first bit received). In order to ensure correct handling, it is recommended to refer to the receive start interrupt RSI before modifying SCTRL.WLE. If TCSRL.WLEMD = 1, it is recommended to update TCSRL and TBUFxx after the receiver start interrupt has been generated.

19.4.2.2 Mode Control Behavior

In SSC mode, the following kernel modes are supported:

- Run Mode 0/1:
  Behavior as programmed, no impact on data transfers.
- Stop Mode 0/1:
  The contents of the transmit buffer is considered as not valid for transmission. Although being considered as 0, bit TCSRL.TDV it is not modified by the stop mode condition.
  In master mode, a currently running word transfer is finished normally, but no new data word is started (the stop condition is not considered as end-of-frame condition).
  In slave mode, a currently running word transfer is finished normally. Passive data will be sent out instead of a valid data word if a data word transfer is started by the external master while the slave device is in stop mode. In order to avoid passive slave transmit data, it is recommended not to program stop mode for an SSC slave device if the master device does not respect the slave device’s stop mode.

19.4.2.3 Disabling SSC Mode

In order to disable SSC mode without any data corruption, the receiver and the transmitter have to be both idle. This is ensured by requesting Stop Mode 1 in register KSCFG. After Stop Mode 1 has been acknowledged by KSCFG.ACK = 1, the SSC mode can be disabled.

19.4.2.4 Data Frame Control

An SSC data frame can consist of several consecutive data words that may be separated by an inter-word delay. Without inter-word delay, the data words seem to form a longer data word, being equivalent to a data frame. The length of the data words are most commonly identical within a data frame, but may also differ from one word to another. The data word length information (defined by SCTRH.WLE) is evaluated for each new data word, whereas the frame length information (defined by SCTRH.FLE) is evaluated at the beginning at each start of a new frame.
The length of an SSC data frame can be defined in two different ways:

- By the number of bits per frame:
  If the number of bits per data frame is defined (frame length FLE), a slave select signal is not necessarily required to indicate the start and the end of a data frame.
  If the programmed number of bits per frame is reached within a data word, the frame is considered as finished and remaining data bits in the last data word are ignored and are not transferred.
  This method can be applied for data frames with up to 63 data bits.
- By the slave select signal:
  If the number of bits per data frame is not known, the start/end information of a data frame is given by a slave select signal. If a deactivation of the slave select signal is detected within a data word, the frame is considered as finished and remaining data bits in the last data word are ignored and are not transferred.
  This method has to be applied for frames with more than 63 data bits (programming limit of FLE). The advantage of slave select signals is the clearly defined start and end condition of data frames in a data stream. Furthermore, slave select signals allow to address slave devices individually.

19.4.2.5 Parity Mode

Parity generation is not supported in SSC mode and bit field CCR.PM = 00_B has to be programmed.

19.4.2.6 Transfer Mode

In SSC mode, bit field SCTRL.TRM = 01_B has to be programmed to allow data transfers. Setting SCTRL.TRM = 00_B disables and stops the data transfer immediately.

19.4.2.7 Data Transfer Interrupt Handling

The data transfer interrupts indicate events related to SSC frame handling.

- Transmit buffer interrupt TBI:
  Bit PSR.TBIF is set after the start of first data bit of a data word.
- Transmit shift interrupt TSI:
  Bit PSR.TSIF is set after the start of the last data bit of a data word.
- Receiver start interrupt RSI:
  Bit PSR.RSIF is set after the reception of the first data bit of a data word.
  With this event, bit **TCSRL**.TDV is cleared and new data can be loaded to the transmit buffer.
- Receiver interrupt RI:
  Bit PSR.RIF is set at after the reception of the last data bit of a data word. Bit RBUFSR.SOF indicates whether the received data word has been the first data word of a new data frame.
- Alternative interrupt AI:
  In SSC mode, this interrupt is not used.

19.4.2.8 Protocol-Related Argument and Error

The protocol-related argument (PAR) and the protocol-related error (PERR) are two flags that are assigned to each received data word in the corresponding receiver buffer status registers.
In SSC mode, these flags are always 0 (parity handling must be disabled).

19.4.2.9 Receive Buffer Handling

If a receive FIFO buffer is available (CCFG.RB = 1) and enabled for data handling (RBCTRH.SIZE > 0), it is recommended to set RBCTRH.RCIM = 01_B in SSC mode. This leads to an indication that the data word has been the first data word of a new data frame if bit OUTRH.RCI[4] = 1, and the word length of the received data is given by OUTRH.RCI[3:0].
The standard receive buffer event and the alternative receive buffer event can be used for the following operation in RCI mode (RBCTRH.RNM = 1):

- A standard receive buffer event indicates that a data word can be read from OUTRL that has not been the first word of a data frame.
- An alternative receive buffer event indicates that the first data word of a new data frame can be read from OUTRL.

19.4.3 Operating the SSC in Master Mode

In order to operate the SSC in master mode, the following issues have to be considered:

- Select SSC mode:
  It is recommended to configure all parameters of the SSC that do not change during run time while CCR.MODE = 0000_B. Bit field SCTRL.TRM = 01_B has to be programmed. The configuration of the input stages has to be done while CCR.MODE = 0000_B to avoid unintended edges of the input signals and the SSC mode can be enabled by CCR.MODE = 0001_B afterwards.
- Pin connections:
  Establish a connection of input stage DX0 with the selected receive data input pin (DIN) with DX0CR.INSW = 1 and configure a transmit data output pin (DOUT).
- Baud rate generation:
  The desired baud rate setting has to be selected, comprising the fractional divider and the baud rate generator. Bit DX1CR.INSW = 0 has to be programmed to use the baud rate generator output SCLK directly as input for the data shift unit. Configure a shift clock output pin (signal SCLKOUT).
- Slave select generation:
  The slave select delay generation has to be enabled by setting PCRL.MSLSEN = 1 and the programming of the time quanta counter setting. Bit DX2CR.INSW = 0 has to be programmed to use the slave select generator output MSLS as input for the data shift unit. Configure slave select output pins (signals SELOx) if needed.
- Data format configuration:
  The word length, the frame length, and the shift direction have to be set up according to the application requirements by programming the registers SCTRL and SCTRH.

Note: The USIC can only receive in master mode if it is transmitting, because the master frame handling refers to bit TDV of the transmitter part.

19.4.3.1 Baud Rate Generation

The baud rate (determining the length of one data bit) of the SSC is defined by the frequency of the SCLK signal (one period of f_{SCLK} represents one data bit). The SSC baud rate generation does not imply any time quanta counter.
In a standard SSC application, the phase relation between the optional MCLK output signal and SCLK is not relevant and can be disabled (BRGL.PPPEN = 0). In this case, the SCLK signal directly derives from the protocol input frequency f_{PIN}. In the exceptional case that a fixed phase relation between the MCLK signal and SCLK is required (e.g. when using MCLK as clock reference for external devices), the additional divider by 2 stage has to be taken into account (BRGL.PPPEN = 1).

The adjustable divider factor is defined by bit field BRGH.PDIV.

(19.8)

$$f_{SCLK} = \begin{cases} \frac{f_{PIN}}{2} \times \frac{1}{PDIV + 1} & \text{if} \quad PPPEN = 0 \\ \frac{f_{PIN}}{2 \times 2} \times \frac{1}{PDIV + 1} & \text{if} \quad PPPEN = 1 \end{cases}$$

19.4.3.2 MSLS Generation

The slave select signals indicate the start and the end of a data frame and are also used by the communication master to individually select the desired slave device. A slave select output of the communication master becomes active a programmable time before a data part of the frame is started (leading delay T_{ld}), necessary to prepare the slave device for the following communication. After the transfer of a data part of the frame, it becomes inactive again a programmable time after the end of the last bit (trailing delay T_{td}) to respect the slave hold time requirements. If data frames are transferred back-to-back one after the other, the minimum time between the deactivation of the slave select and the next activation of a slave select is programmable (next-frame delay T_{nf}). If a data frame consists of more than one data word, an optional delay between the data words can also be programmed (inter-word delay T_{iw}).

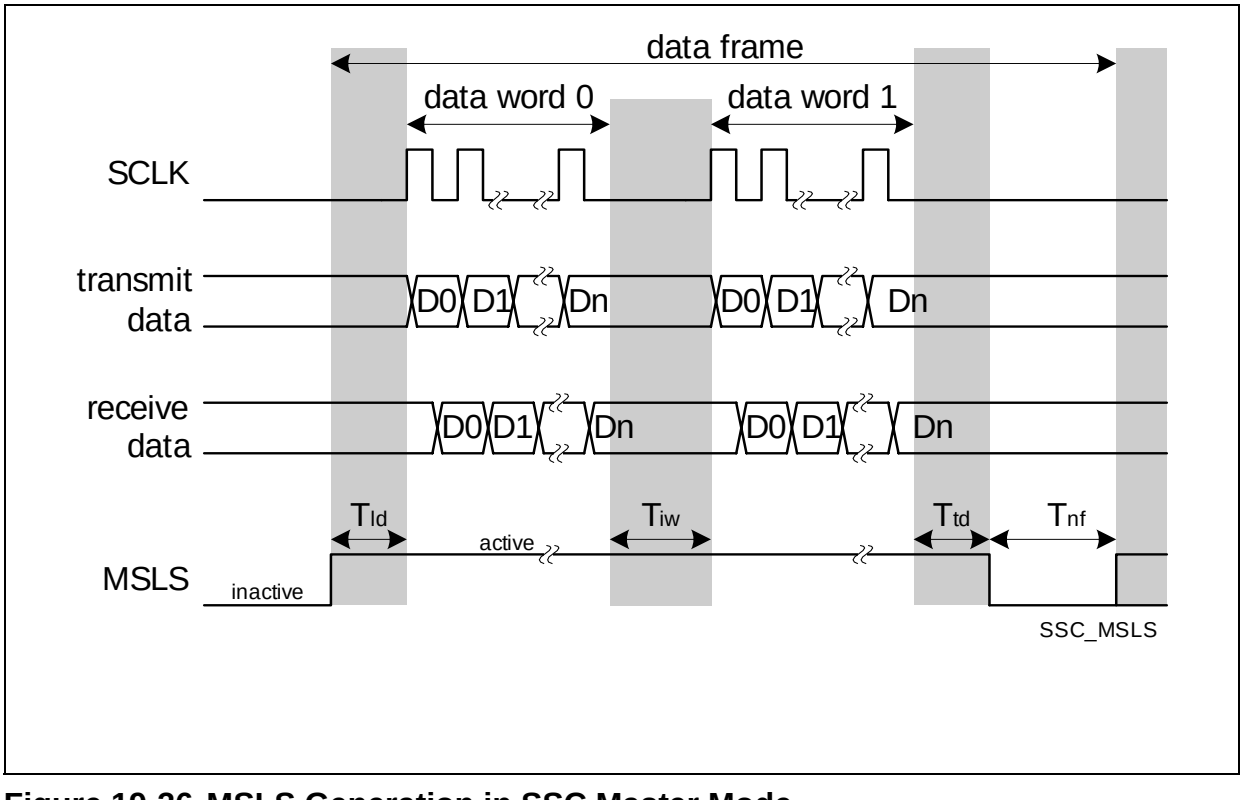


Figure 19-36 MSLS Generation in SSC Master Mode

In SSC master mode, the slave select delays are defined as follows:

- Leading delay T_{ld}:
  The leading delay starts if valid data is available for transmission. The internal signal MSLS becomes active with the start of the leading delay. The first shift clock edge (rising edge) of SCLK is generated by the baud rate generator after the leading delay has elapsed.
- Trailing delay T_{td}
  The trailing delay starts at the end of the last SCLK cycle of a data frame. The internal signal MSLS becomes inactive with the end of the trailing delay.
- Inter-word delay T_{iw}:
  This delay is optional and can be enabled/disabled by PCRH.TIWEN. If the inter-word delay is disabled (TIWEN = 0), the last data bit of a data word is directly followed by the first data bit of the next data word of the same data frame. If enabled (TIWEN = 1), the inter-word delay starts at the end of the last SCLK cycle of a data word. The first SCLK cycle of the following data word of the same data frame is started when the inter-word delay has elapsed. During this time, no shift clock pulses are generated and signal MSLS stays active. The communication partner has time to “digest” the previous data word or to prepare for the next one.
- Next-frame delay T_{nf}:
  The next-frame delay starts at the end of the trailing delay. During this time, no shift clock pulses are generated and signal MSLS stays inactive. A frame is considered as finished after the next-frame delay has elapsed.

19.4.3.3 Automatic Slave Select Update

If the number of bits per SSC frame and the word length are defined by bit fields SCTRH.FLE and SCTRH.WLE, the transmit control information TCI can be used to update the slave select setting PCRH.CTR[23:16] to control the SELOx select outputs. The automatic update mechanism is enabled by TCSRL.SELMD = 1 (bits TCSRL.WLEMD, FLEMD, and WAMD have to be cleared). In this case, the TCI of the first data word of a frame defines the slave select setting of the complete frame due to the automatic shadow mechanism (see **Section 19.3.3.6**).

19.4.3.4 Slave Select Delay Generation

The slave select delay generation is based on time quanta. The length of a time quantum (defined by the period of the f_{CTQIN}) and the number of time quanta per delay can be programmed.
In standard SSC applications, the leading delay T_{ld} and the trailing delay T_{td} are mainly used to ensure stability on the input and output lines as well as to respect setup and hold times of the input stages. These two delays have the same length (in most cases shorter than a bit time) and can be programmed with the same set of bit fields.

- BRGL.CTQSEL
  to define the input frequency f_{CTQIN} for the time quanta generation for T_{ld} and T_{td}
- BRGL.PCTQ
  to define the length of a time quantum (division of f_{CTQIN} by 1, 2, 3, or 4) for T_{ld} and T_{td}
- BRGL.DCTQ
  to define the number of time quanta for the delay generation for T_{ld} and T_{td}

The inter-word delay T_{iw} and the next-frame delay T_{nf} are used to handle received data or to prepare data for the next word or frame. These two delays have the same length (in most cases in the bit time range) and can be programmed with a second, independent set of bit fields.

- PCRL.CTQSEL1
  to define the input frequency f_{CTQIN} for the time quanta generation for T_{nf} and T_{iw}
- PCRL.PCTQ1
  to define the length of a time quantum (division of f_{CTQIN} by 1, 2, 3, or 4) for T_{nf} and T_{iw}
- PCRL.DCTQ1
  to define the number of time quanta for the delay generation for T_{nf} and T_{iw}
- PCRH.TIWEN
  to enable/disable the inter-word delay T_{iw}

Each delay depends on the length of a time quantum and the programmed number of time quanta given by the bit fields CTQSEL/CTQSEL1, PCTQ/DCTQ and PCTQ1/DCTQ1 (the coding of CTQSEL1 is similar to CTQSEL, etc.). To provide a high flexibility in programming the delay length, the input frequencies can be selected between several possibilities (e.g. based on bit times or on the faster inputs of the protocol-related divider). The delay times are defined as follows:

(19.9)

$$T_{ld} = T_{td} = \frac{(PCTQ + 1) \times (DCTQ + 1)}{f_{CTQIN}}$$

$$T_{iw} = T_{nf} = \frac{(PCTQ1 + 1) \times (DCTQ1 + 1)}{f_{CTQIN}}$$

19.4.3.5 Protocol Interrupt Events

The following protocol-related events generated in SSC mode and can lead to a protocol interrupt. They are related to the start and the end of a data frame. After the start of a data frame a new setting could be programmed for the next data frame and after the end of a data frame the SSC connections to pins can be changed.
Please note that the bits in register PSR are not all automatically cleared by HW and have to be cleared by SW in order to monitor new incoming events.

- MSLS Interrupt:
  This interrupt indicates in master mode (MSLS generation enabled) that a data frame has started (activation of MSLS) and has been finished (deactivation of MSLS). Any change of the internal MSLS signal sets bit PSR.MSLSEV and additionally, a protocol interrupt can be generated if PCRL.MSLSIEN = 1. The actual state of the internal MSLS signal can be read out at PSR.MSLS to take appropriate actions when this interrupt has been detected.
- DX2T Interrupt:
  This interrupt monitors edges of the input signal of the DX2 stage (although this signal is not used as slave select input for data transfers).
  A programmable edge detection for the DX2 input signal sets bit PSR.DX2TEV and additionally, a protocol interrupt can be generated if PCRL.DX2TIEN = 1. The actual state of the selected input signal can be read out at PSR.DX2S to take appropriate actions when this interrupt has been detected.

19.4.3.6 End of Frame Control

The information about the frame length is required for the MSLS generator of the master device. In addition to the mechanism based on the number of bits per frame (selected with SCTRH.FLE<63), the following alternative mechanisms for end of frame handling are supported. It is recommended to set SCTRH.FLE = 63 (if several end of frame mechanisms are activated in parallel, the first end condition being found finishes the frame).

- SW-based start of frame indication TCSRL.SOF:
  This mechanism can be used if SW handles the TBUF data without data FIFO. If bit SOF is set, a valid contents of TBUF is considered as first word of a new frame. Bit SOF has to be set before the content of TBUF is transferred to the transmit shift register, so it is recommended to write it before writing data to TBUF. A current data word transfer is finished completely and the slave select delays T_{td} and T_{nf} are applied before starting a new data frame with T_{ld} and the contents of TBUF.
  For SW-handling of bit SOF, bit TCSRL.WLEMD = 0 has to be programmed. In this case, all TBUF[31:0] address locations show an identical behavior (TCI not taken into account for data handling).
- SW-based end of frame indication TCSRL.EOF:
  This mechanism can be used if SW handles the TBUF data without data FIFO. If bit EOF is set, a valid contents of TBUF is considered as last word of a new frame. Bit EOF has to be set before the content of TBUF is transferred to the transmit shift register, so it is recommended to write it before writing data to TBUF. The data word in TBUF is sent out completely and the slave select delays T_{td} and T_{nf} are applied. A new data frame can start with T_{ld} with the next valid TBUF value.
  For SW-handling of bit EOF, bit TCSRL.WLEMD = 0 has to be programmed. In this case, all TBUF[31:0] address locations show an identical behavior (TCI not taken into account for data handling).
- SW-based address related end of frame handling:
  This mechanism can be used if SW handles the TBUF data without data FIFO. If bit TCSRL.WLEMD = 1, the address of the written TBUF[31:0] is used as transmit control information TCI[4:0] to update SCTRH.WLE (= TCI[3:0]) and TCSRL.EOF (= TCI[4]) for each data word. The written TBUF[31:0] address location defines the word length and the end of a frame (locations TBUF[31:16] lead to a frame end).
  For example, writing transmit data to TBUF[07] results in a data word of 8 bit length without finishing the frame, whereas writing transmit data to TBUF[31] leads to a data word length of 16 bits, followed by T_{td}, the deactivation of MSLS and T_{nf}.
  If TCSRL.WLEMD = 1, bits TCSRL.EOF and SOF, as well as SCTRH.WLE must not be written by SW after writing data to a TBUF location. Furthermore, it is recommended to clear bits TCSRL.SELMD, FLEMD and WAMD.
- FIFO-based address related end of frame handling:
  This mechanism can be used if a data FIFO is used to store the transmit data. The general behavior is similar to the SW-based address related end of frame handling,

except that transmit data is not written to the locations TBUF[31:0], but to the FIFO input locations IN[31:0] instead. In this case, SW must not write to any of the TBUF locations.

- TBUF related end of frame handling:
  If bit PCRL.FEM = 0, an end of frame is assumed if the transmit buffer TBUF does not contain valid transmit data at the end of a data word transmission (TCSRL.TDV = 0 or in Stop Mode). In this case, the SW has to take care that TBUF does not run empty during a data frame in Run Mode. If bit PCRL.FEM = 1, signal MSLS stays active while the transmit buffer is waiting for new data (TCSRL.TDV = 1 again) or until Stop Mode is left.
- Explicit end of frame by SW:
  The SW can explicitly stop a frame by clearing bit PSR.MSLS by writing a 1 to the related bit position in register PSCR. This write action immediately clears bit PSR.MSLS, whereas the internal MSLS signal becomes inactive after finishing a currently running word transfer and respecting the slave select delays T_{td} and T_{nf}.

19.4.4 Operating the SSC in Slave Mode

In order to operate the SSC in slave mode, the following issues have to be considered:

- Select SSC mode:
  It is recommended to configure all parameters of the SSC that do not change during run time while CCR.MODE = 0000_B. Bit field SCTRL.TRM = 01_B has to be programmed. The configuration of the input stages has to be done while CCR.MODE = 0000_B to avoid unintended edges of the input signals and the SSC mode can be enabled afterwards by CCR.MODE = 0001_B.
- Pin connections:
  Establish a connection of input stage DX0 with the selected receive data input pin (signal DIN) with DX0CR.INSW = 1 and configure a transmit data output pin (signal DOUT).
  Establish a connection of input stage DX1 with the selected shift clock input pin (signal SCLKIN) with DX1CR.INSW = 1.
  Establish a connection of input stage DX2 with the selected slave select input pin (signal SELIN) with DX2CR.INSW = 1. If no slave select input signal is used, the DX2 stage has to deliver a 1-level to the data shift unit to allow data reception and transmission. If a slave device is not selected (DX2 stage delivers a 0 to the data shift unit) and a shift clock pulse are received, the incoming data is not received and the DOUT signal outputs the passive data level defined by SCTRL.PDL.
- Baud rate generation:
  The baud rate generator is not needed and can be switched off by the fractional divider.
- Slave select generation:
  The slave select delay generation is not needed and can be switched off. The bits and bit fields MSLSEN, SELCTR, SELINV, CTQSEL1, PCTQ1, DCTQ1, MSLSIEN, SELO[7:0], and TIWEN in registers PCRL/PCRH are not necessary and can be programmed to 0.

19.4.4.1 Protocol Interrupts

The following protocol-related events generated in SSC mode and can lead to a protocol interrupt. They are related to the start and the end of a data frame. After the start of a data frame a new setting could be programmed for the next data frame and after the end of a data frame the SSC connections to pins can be changed.
Please note that the bits in register PSR are not all automatically cleared by HW and have to be cleared by SW in order to monitor new incoming events.

- MSLS event:
  The MSLS generation being switched off, this event is not available.
- DX2T event:
  The slave select input signal SELIN is handled by the DX2 stage and the edges of the selected signal can generate a protocol interrupt. This interrupt allows to indicate

that a data frame has started and/or that a data frame has been completely finished. A programmable edge detection for the DX2 input signal activates DX2T, sets bit PSR.DX2TEV and additionally, a protocol interrupt can be generated if PCRL.DX2TIEN = 1. The actual state of the selected input signal can be read out at PSR.DX2S to take appropriate actions when this interrupt has been detected.

19.4.4.2 End of Frame Control

In slave mode, the following possibilities exist to determine the frame length. The slave device either has to refer to an external slave select signal, or to the number of received data bits.

- Frame length known in advance by the slave device, no slave select:
  In this case bit field SCTRH.FLE can be programmed to the known value (if it does not exceed 63 bits). A currently running data word transfer is considered as finished if the programmed frame length is reached.
- Frame length not known by the slave, no slave select:
  In this case, the slave device's SW has to decide on data word base if a frame is finished. Bit field SCTRH.FLE can be either programmed to the word length SCTRH.WLE, or to its maximum value to disable the slave internal frame length evaluation by counting received bits.
- Slave device addressed via slave select signal SELIN:
  If the slave device is addressed by a slave select signal delivered by the communication master, the frame start and end information are given by this signal. In this case, bit field SCTRH.FLE should be programmed to its maximum value to disable the slave internal frame length evaluation.

19.4.5 SSC Protocol Registers

In SSC mode, the bits in the registers PCRL, PCRH and PSR handle SSC related information.

19.4.5.1 Protocol Control Registers

The following PCRL/PCRH control bits or bit fields are available in SSC mode.

Table 19-19 Signification of PCRL/PCRH Bits for SSC

Field	Alias	Description
CTR[0]	**MSLSEN**	**MSLS Enable** This bit enables/disables the generation of the master slave select signal MSLS. If the SSC is a transfer slave, the SLS information is read from a pin and the internal generation is not needed. If the SSC is a transfer master, it has to provide the MSLS signal. 0_B The MSLS generation is disabled (MSLS = 0). This is the setting for SSC slave mode. 1_B The MSLS generation is enabled. This is the setting for SSC master mode.
CTR[1]	**SELCTR**	**Select Control** This bit selects the operating mode for the SELO[7:0] outputs. 0_B The coded select mode is enabled. 1_B The direct select mode is enabled.
CTR[2]	**SELINV**	**Select Inversion** This bit defines if the polarity of the SELO[7:0] outputs in relation to the master slave select signal MSLS. 0_B The SELO outputs have the same polarity as the MSLS signal (active high). 1_B The SELO outputs have the inverted polarity to the MSLS signal (active low).

Table 19-19 Signification of PCRL/PCRH Bits for SSC (cont'd)

Field	Alias	Description
CTR[3]	**FEM**	**Frame End Mode** This bit defines if a transmit buffer content that is not valid for transmission is considered as an end of frame condition for the slave select generation. 0_B The current data frame is considered as finished when the last bit of a data word has been sent out and the transmit buffer TBUF does not contain new data (TDV = 0). 1_B The MSLS signal is kept active also while no new data is available and no other end of frame condition is reached. In this case, the SW can accept delays in delivering the data without automatic deactivation of MSLS in multi-word data frames.
CTR [5:4]	**CTQ SEL1**	**Input Frequency Selection** This bit field defines the input frequency f_{CTQIN} for the generation of the slave select delays T_{iw} and T_{nf}. 00_B $f_{CTQIN} = f_{PDIV}$ 01_B $f_{CTQIN} = f_{PPP}$ 10_B $f_{CTQIN} = f_{SCLK}$ 11_B $f_{CTQIN} = f_{MCLK}$
CTR [7:6]	**PCTQ1**	**Divider Factor PCTQ1 for T_{iw} and T_{nf}** This bit field represents the divider factor PCTQ1 (range = 0 - 3) for the generation of the inter-word delay and the next-frame delay. $T_{iw} = T_{nf} = 1/f_{CTQIN} * (PCTQ1 + 1) * (DCTQ1 + 1)$
CTR [12:8]	**DCTQ1**	**Divider Factor DCTQ1 for T_{iw} and T_{nf}** This bit field represents the divider factor DCTQ1 (range = 0 - 31) for the generation of the inter-word delay and the next-frame delay. $T_{iw} = T_{nf} = 1/f_{CTQIN} * (PCTQ1 + 1) * (DCTQ1 + 1)$
CTR[14]	**MSLS IEN**	**MSLS Interrupt Enable** This bit enables/disables the generation of a protocol interrupt if the state of the MSLS signal changes (indicated by PSR.MSLSEV = 1). 0_B A protocol interrupt is not generated if a change of signal MSLS is detected. 1_B A protocol interrupt is generated if a change of signal MSLS is detected.

Table 19-19 Signification of PCRL/PCRH Bits for SSC (cont'd)

Field	Alias	Description
CTR[15]	**DX2T IEN**	**DX2T Interrupt Enable** This bit enables/disables the generation of a protocol interrupt if the DX2T signal becomes activated (indicated by PSR.DX2TEV = 1). 0_B A protocol interrupt is not generated if DX2T is activated. 1_B A protocol interrupt is generated if DX2T is activated.
CTR [23:16]	**SELO [7:0]**	**Select Output** This bit field defines the setting of the SELO[7:0] output lines. 0_B The corresponding SELOx line can not be activated. 1_B The corresponding SELOx line can be activated (according to the mode selected by SELCTR).
CTR[24]	**TIWEN**	**Enable Inter-Word Delay T_{iw}** This bit enables/disables the inter-word delay T_{iw} after the transmission of a data word. 0_B No delay between data words of the same frame. 1_B The inter-word delay T_{iw} is enabled and introduced between data words of the same frame.
CTR[31]	**MCLK**	**Master Clock Enable** This bit enables/disables the generation of the master clock output signal MCLK, independent from master or slave mode. 0_B The MCLK generation is disabled and output MCLK = 0. 1_B The MCLK generation is enabled.
other CTR[x]		**Reserved** returns 0 if read; should be written with 0;

19.4.5.2 Protocol Status Register PSR

The following PSR status bits or bit fields are available in SSC mode. Please note that the bits in register PSR are not cleared by HW.
The row marked PI indicates which status bit can generate a protocol interrupt in SSC mode. The general interrupt status flags are described in the general interrupt chapter.

Table 19-20 Signification of PSR Bits for SSC

Bit	Alias	Description	PI
ST[0]	**MSLS**	**MSLS Status** This bit indicates the current status of the MSLS signal. It be cleared by SW to stop a running frame. 0 The internal signal MSLS is inactive (0). 1 The internal signal MSLS is active (1).	-
ST[1]	**DX2S**	**DX2S Status** This bit indicates the current status of the DX2S signal that can be used as slave select input SELIN. 0 DX2S is 0. 1 DX2S is 1.	-
ST[2]	**MSLS EV**	**MSLS Event Detected** This bit indicates that the MSLS signal has changed its state since MSLSEV has been cleared. Together with the MSLS status bit, the activation/deactivation of the MSLS signal can be monitored. 0 The MSLS signal has not changed its state. 1 The MSLS signal has changed its state.	y
ST[3]	**DX2T EV**	**DX2T Event Detected** This bit indicates that the DX2T trigger signal has been activated since DX2TEV has been cleared. 0 The DX2T signal has not been activated. 1 The DX2T signal has been activated.	y
other ST[x]		**Reserved** returns 0 if read; not modified in SSC mode	y
PSR [15:10]		**General Interrupt Flags**	-

19.4.6 SSC Timing Considerations

The input and output signals have to respect certain timings in order to ensure correct data reception and transmission. In addition to module internal timings (due to input filters, reaction times on events, etc.), also the timings from the input pin via the input stage (T_{in}) to the module and from the module via the output driver stage to the pin (T_{out}), as well as the signal propagation on the wires (T_{prop}) have to be taken into account.
Please note that there might by additional delays in the DXn input stages, because the digital filter and the synchronization stages lead to systematic delays, that have to be considered if these functions are used.

19.4.6.1 Closed-loop Delay

A system-inherent limiting factor for the baud rate of an SSC connection is the closed-loop delay. In a typical application setup, a communication master device is connected to a slave device in full-duplex mode with independent lines for transmit and receive data. In a general case, all transmitters refer to one shift clock edge for transmission and all receivers refer to the other shift clock edge for reception. The master device's SSC module sends out the transmit data, the shift clock and optionally the slave select signal. Therefore, the baud rate generation (BRG) and slave select generation (SSG) are part of the master device. The frame control is similar for SSC modules in master and slave mode, the main difference is the fact which module generates the shift clock and optionally, the slave select signals.

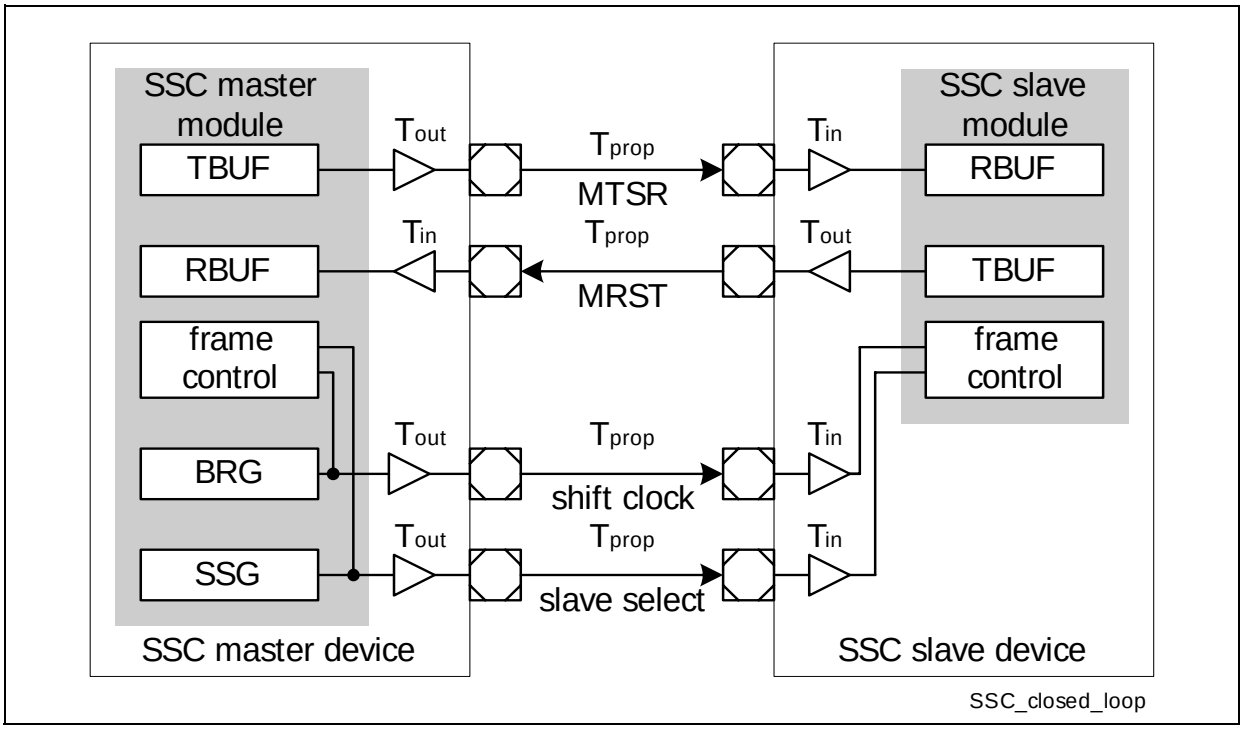


Figure 19-37 SSC Closed-loop Delay

The signal path between the SSC modules of the master and the slave device includes the master's output driver, the wiring to the slave device and the slave device's input stage. With the received shift clock edges, the slave device receives the master's transmit data and transmits its own data back to the master device, passing by a similar signal path in the other direction. The master module receives the slave's transmit data related to its internal shift clock edges. In order to ensure correct data reception in the master device, the slave's transmit data has to be stable (respecting setup and hold times) as master receive data with the next shift clock edge of the master (generally 1/2 shift clock period). To avoid data corruption, the accumulated delays of the input and output stages, the signal propagation on the wiring and the reaction times of the transmitter/receiver have to be carefully considered, especially at high baud rates.
In the given example, the time between the generation of the shift clock signal and the evaluation of the receive data by the master SSC module is given by the sum of T_{out_master} + 2*T_{prop} + T_{in_slave} + T_{out_slave} + T_{in_master} + module reaction times + input setup times.
The input path is characterized by an input delay depending mainly on the input stage characteristics of the pads. The output path delay is determined by the output driver delay and its slew rate, the external load and current capability of the driver. The device specific values for the input/output driver are given in the AC/DC chapter.

19.4.6.2 Delay Compensation in Master Mode

A higher baud rate can be reached by delay compensation in master mode. This compensation is possible if (at least) the shift clock pin is bidirectional.

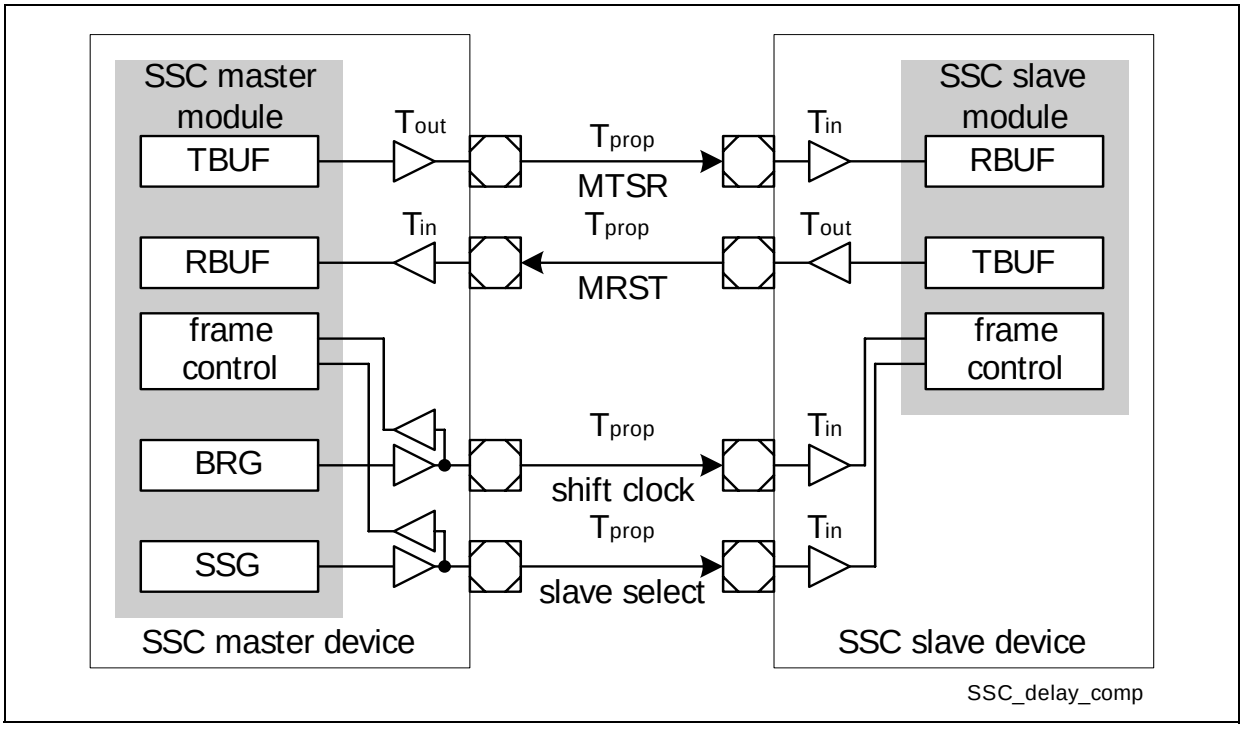


Figure 19-38 SSC Master Mode with Delay Compensation

If the shift clock signal in master mode is directly taken from the input function in parallel to the output signal, the output delay of the master device's shift clock output is compensated and only the difference between the input delays of the master and the slave devices have to be taken into account instead of the complete master's output delay and the slave's input delay of the shift clock path.
In the given example, the time between the evaluation of the shift clock signal and the receive data by the master SSC module is reduced by $T_{in\_master} + T_{out\_master}$.
Although being a master mode, the shift clock input and optionally the slave select signal are not directly connected internally to the data shift unit, but are taken as external signals from input pins (DXnCTR.INSW = 1). The delay compensation does not lead to additional pins for the SSC communication if the shift clock output pin (slave select output pin, respectively) is/are bidirectional. In this case, the input signal is decoupled from other internal signals, because it is related to the signal level at the pin itself.

19.4.6.3 SSC Timing Values

Figure 19-39 shows an example for SSC signal timings on module level. It does not imply the input/output driver delays that have to be additionally taken into consideration.

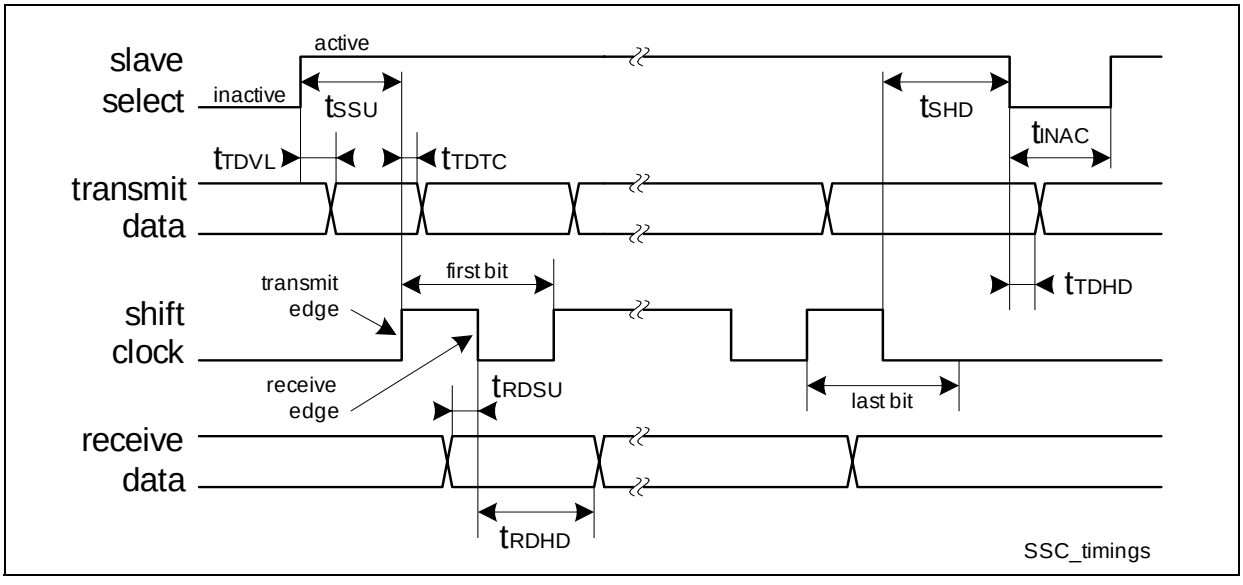


Figure 19-39 SSC Signal Timings

Table 19-21 SSC Signal Timings

Timing Relation	Symbol	Min.	Max.	Unit
slave select setup time (slave select active before first shift clock edge)	t_{SSU}	10	-	ns
slave select hold time (slave select active after last shift clock edge)	t_{SHD}	10	-	ns
slave select inactive time (slave select inactive to slave select active)	t_{INAC}	2	-	t_{SYS} = 1 / f_{SYS}
Transmit data valid (transmit data stable after slave select active)	t_{TDVL}	0	10	ns
Transmit data to clock (transmit data valid after transmit clock edge)	t_{TDTC}	0	10	ns
Transmit data hold (transmit data stable after slave select inactive)	t_{TDHD}	0	10	ns
Receive data setup (receive data stable before receive clock edge)	t_{RDSU}	10	-	ns
Receive data hold (receive data stable after receive clock edge)	t_{RDHD}	10	-	ns

19.5 Inter-IC Bus Protocol (IIC)

The IIC protocol of the USIC refers to the IIC bus specification version 2.1, january 2000 from Philips Semiconductors. Contrary to that specification, the USIC device assumes rise/fall times of the bus signals of max. 300 ns in all modes. Please refer to the pad characteristics in the AC/DC chapter for the driver capability. CBUS mode and HS mode are not supported.
The IIC mode is selected by CCR.MODE = 0100_B with CCFG.IIC = 1 (IIC mode available).

This chapter contains the following sections:

- Introduction (see **Section 19.5.1**)
- Operating the IIC protocol (see **Section 19.5.2**)
- Symbol timing and programming (see **Section 19.5.3**)
- Data flow handling (see **Section 19.5.4**)
- IIC protocol registers (see **Section 19.5.5**)

19.5.1 Introduction

USIC IIC Features:

- Two-wire interface, with one line for shift clock transfer and synchronization (shift clock SCL), the other one for the data transfer (shift data SDA)
- Communication in standard mode (100 kBit/s) or in fast mode (up to 400 kBit/s)
- Support of 7-bit addressing, as well as 10-bit addressing
- Master mode operation,
  where the IIC controls the bus transactions and provides the clock signal.
- Slave mode operation,
  where an external master controls the bus transactions and provides the clock signal.
- Multi-master mode operation,
  where several masters can be connected to the bus and bus arbitration can take place, i.e. the IIC module can be master or slave. The master/slave operation of an IIC bus participant can change from frame to frame.
- Efficient frame handling (low SW effort), also allowing PEC transfers
- Powerful interrupt handling due to multitude of indication flags
- Compensation support for input delays

19.5.1.1 Signal Description

An IIC connection is characterized by two wires (SDA and SCL). The output drivers for these signals must have open-drain characteristics to allow the wired-AND connection of all SDA lines together and all SCL lines together to form the IIC bus system. Due to this structure, a high level driven by an output stage does not necessarily lead immediately to a high level at the corresponding input. Therefore, each SDA or SCL connection has to be input and output at the same time, because the input function always monitors the level of the signal, also while sending.

- Shift data SDA: input handled by DX0 stage, output signal DOUT
- Shift clock SCL: input handled by DX1 stage, output signal SCLKOUT

Figure 19-24 shows a connection of two IIC bus participants (modules IIC A and IIC B) using the USIC. In this example, the pin assignment of module IIC A shows separate pins for the input and output signals for SDA and SCL. This assignment can be used if the application does not provide pins having DOUT and a DX0 stage input for the same pin (similar for SCLKOUT and DX1). The pin assignment of module IIC B shows the connection of DOUT and a DX0 input at the same pin, also for SCLKOUT and a DX1 input.

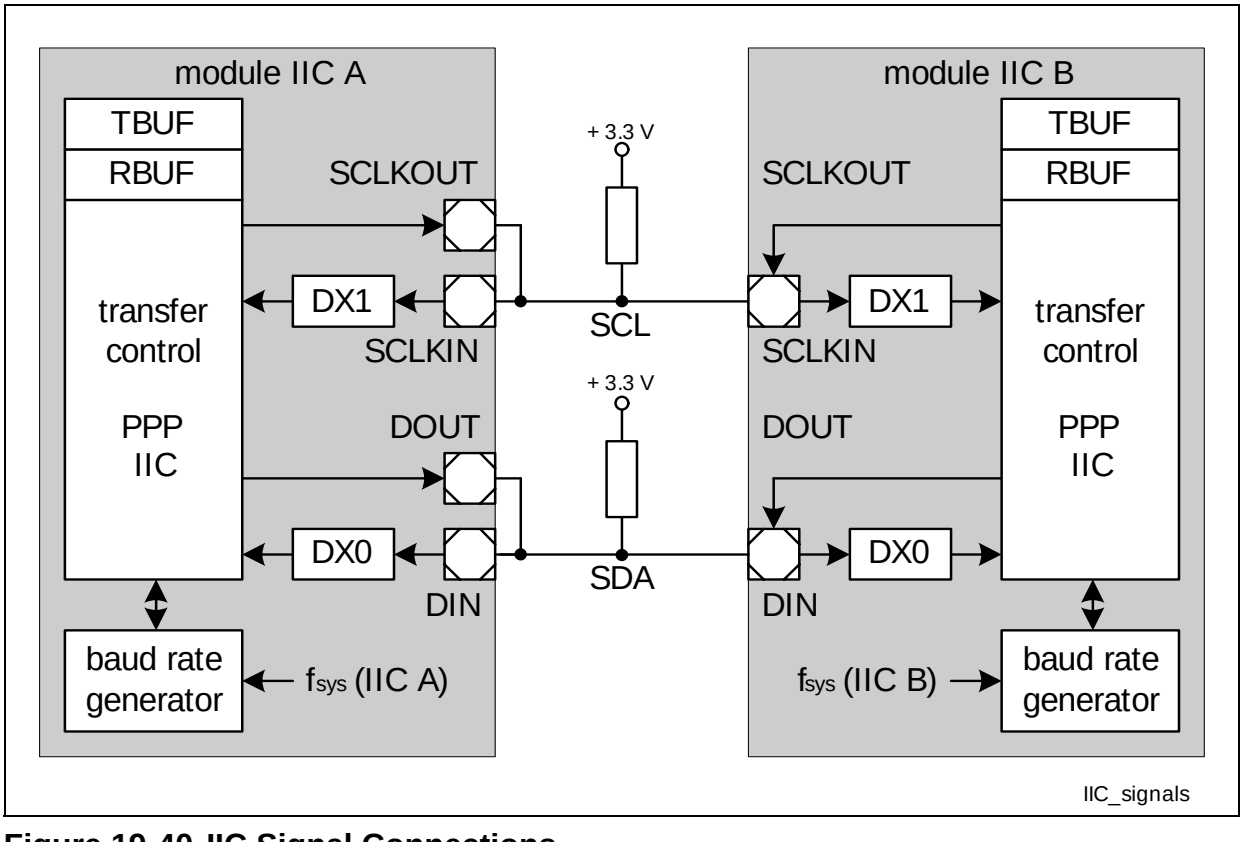


Figure 19-40 IIC Signal Connections

19.5.1.2 Symbols

A symbol is a sequence of edges on the lines SDA and SCL. Symbols contain 10 or 25 time quanta tq, depending on the selected baud rate. The baud rate generator determines the length of the time quanta tq, the sequence of edges in a symbol is handled by the IIC protocol pre-processor, and the sequence of symbols can be programmed by the user according to the application needs.

The following symbols are defined:

- Bus idle:
  SDA and SCL are high. No data transfer takes place currently.
- Data bit symbol:
  SDA stable during the high phase of SCL. SDA then represents the transferred bit value. There is one clock pulse on SCL for each transferred bit of data. During data transfers SDA may only change while SCL is low.
- Start symbol:
  Signal SDA being high followed by a falling edge of SDA while SCL is high indicates a start condition. This start condition initiates a data transfer over the IIC bus after the bus has been idle.
- Repeated start symbol:
  This start condition initiates a data transfer over the bus after a data symbol when the bus has not been idle. Therefore, SDA is set high and SCL low, followed by a start symbol.
- Stop symbol:
  A rising edge on SDA while SCL is high indicates a stop condition. This stop condition terminates a data transfer to release the bus to idle state. Between a start condition and a stop condition an arbitrary number of bytes may be transferred.

19.5.1.3 Frame Format

Data is transferred by the 2-line IIC bus (SDA, SCL) using a protocol that ensures reliable and efficient transfers. The sender of a (data) byte receives and checks the value of the following acknowledge field. The IIC being a wired-AND bus system, a 0 of at least one device leads to a 0 on the bus, which is received by all devices.
A data word consists of 8 data bit symbols for the data value, followed by another data bit symbol for the acknowledge bit. The data word can be interpreted as address information (after a start symbol) or as transferred data (after the address).
In order to be able to receive an acknowledge signal, the sender of the data bits has to release the SDA line by sending a 1 as acknowledge value. Depending on the internal state of the receiver, the acknowledge bit is either sent active or passive.

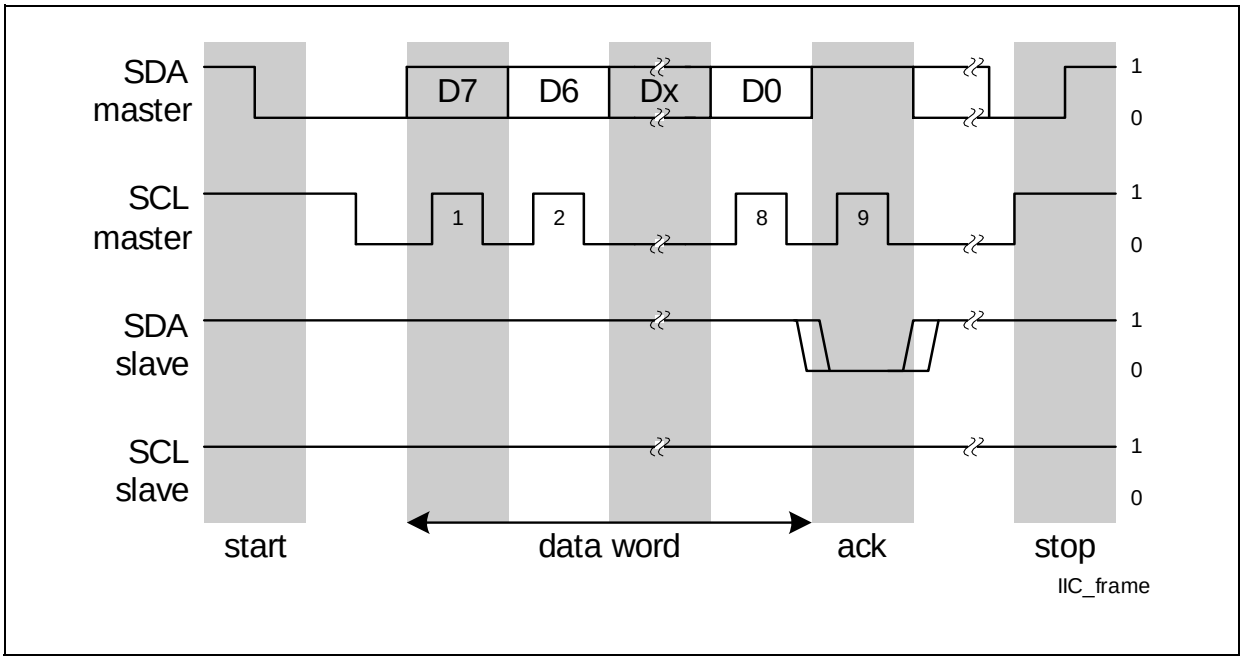


Figure 19-41 IIC Frame Example (simplified)

19.5.2 Operating the IIC

In order to operate the IIC protocol, the following issues have to be considered:

- Select IIC mode:
  It is recommended to configure all parameters of the IIC that do not change during run time while CCR.MODE = 0000_B. Bit field SCTRL.TRM = 11_B should to be programmed. The configuration of the input stages has to be done while CCR.MODE = 0000_B to avoid unintended edges of the input signals and the IIC mode can be enabled by CCR.MODE = 0100_B afterwards.
- Pin connections:
  Establish a connection of input stage DX0 (with DX0CR.DPOL = 0) to the selected shift data pin SDA (signal DIN) with DX0CR.INSW = 0 and configure the transmit data output signal DOUT (with SCTRL.DOCFG = 00_B) to the same pin. If available, this can be the same pin for input and output, or connect the selected input pin and the output pin to form the SDA line.
  The same mechanism applies for the shift clock line SCL. Here, signal SCLKOUT (with BRGH.SCLKCFG = 00_B) and an input of the DX1 stage have to be connected (with DX1CR.DPOL = 0).
  The input stage DX2 is not used for the IIC protocol.
  If the digital input filters are enabled in the DX0/1 stages, their delays have to be taken into account for correct calculation of the signal timings.
  The pins used for SDA and SCL have to be set to open-drain mode to support the wired-AND structure of the IIC bus lines.
- Bit timing configuration:
  In standard mode (100 kBit/s) a minimum module frequency of 2 MHz is necessary, whereas in fast mode (400 kBit/s) a minimum of 10 MHz is required. Additionally, if the digital filter stage should be used to eliminate spikes up to 50 ns, a filter frequency of 20 MHz is necessary.
  There could be an uncertainty in the SCL high phase timing of maximum 1/ f_{PPP} if another IIC participant lengthens the SCL low phase on the bus.
  More details are given in **Section 19.5.3**.
- Data format configuration:
  The data format has to be configured for 8 data bits (SCTRH.WLE = 7), unlimited data flow (SCTRH.FLE = $3FF_H$), and MSB shifted first (SCTRL.SDIR = 1). The parity generation has to be disabled (CCR.PM = 00_B).
- General hints:
  The IIC slave module becomes active (for reception or transmission) if it is selected by the address sent by the master. In the case that the slave sends data to the master, it uses the transmit path. So a master must not request to read data from the slave address defined for its own channel in order to avoid collisions.
  The built-in error detection mechanisms are only activated while the IIC module is taking part in IIC bus traffic.
  If the slave can not deal with too high frequencies, it can lengthen the low phase of

the SCL signal.
For data transfers according to the IIC specification, the shift data line SDA shall only change while SCL = 0 (defined by IIC bus specification).

19.5.2.1 Transmission Chain

The IIC bus protocol requiring a kind of in-bit-response during the arbitration phase and while a slave is transmitting, the resulting loop delay of the transmission chain can limit the reachable maximal baud rate, strongly depending on the bus characteristics (bus load, module frequency, etc.).
Figure 19-24 shows the general signal path and the delays in the case of a slave transmission. The shift clock SCL is generated by the master device, output on the wire, then it passes through the input stage and the input filter. Now, the edges can be detected and the SDA data signal can be generated accordingly. The SDA signal passes through the output stage and the wire to the master receiver part. There, it passes through the input stage and the input filter before it is sampled.
This complete loop has to be finished (including all settling times to obtain stable signal levels) before the SCL signal changes again. The delays in this path have to be taken into account for the calculation of the baud rate as a function of f_{SYS} and f_{PPP}.

19.5.2.2 Byte Stretching

If a device is selected as transceiver and should transmit a data byte but the transmit buffer TBUF does not contain valid data to be transmitted, the device ties down SCL = 0 at the end of the previous acknowledge bit. The waiting period is finished if new valid data has been detected in TBUF.

19.5.2.3 Baud Rate Update

The baud rate setting can be changed from frame to frame. The BRGL/H register setting and PCR.STIM are sampled (shadowed) while the IIC bus is idle. A new setting of these bits can be programmed while a frame is running. The new setting will be taken into account with the start of the next frame. In order to minimize the risk of inconsistencies when changing baud rate setting (several registers have to be updated), it is recommended to avoid baud rate changes while the IIC protocol is enabled, especially for slave devices.

19.5.2.4 Master Arbitration

During the address and data transmission, the master transmitter checks at the rising edge of SCL for each data bit if the value it is sending is equal to the value read on the SDA line. If yes, the next data bit values can be 0. If this is not the case (transmitted value = 1, value read = 0), the master has lost the transmit arbitration. This is indicated by status flag PSR.ARL and can generate a protocol interrupt if enabled by

PCRH.ARLIEN.
When the transmit arbitration has been lost, the SW has to initialize the complete frame again, starting with the first address byte together with the start condition for a new master transmit attempt. Arbitration also takes place for the ACK bit.

19.5.2.5 Release of TBUF

In case of a non-acknowledge or an error, the content of TBUF becomes invalid. In both cases, the SW has to flush the transmit buffer and to set it up again with appropriate values to react on the previous event.

19.5.2.6 Mode Control Behavior

In multi-master mode, only run mode 0 and stop mode 0 are supported, the other modes must not be programmed.

- Run Mode 0:
  Behavior as programmed. If TCSRL.TDV = 0 (no new valid TBUF entry found) when a new TBUF entry needs to be processed, the IIC module waits for TDV becoming set to continue operation.
- Run Mode 1:
  Behavior as programmed. If in master mode, TCSRL.TDV = 0 (no new valid TBUF entry found) when a new TBUF entry needs to be processed, the IIC module sends a stop condition to finish the frame. In slave mode, no difference to run mode 0.
- Stop Mode 0:
  Bit TCSRL.TDV is internally considered as 0 (the bit itself is not modified by the stop mode). A currently running word is finished normally, but no new word is started in case of master mode (wait for TDV active).
  Bit TDV being considered as 0 for master and slave, the slave will force a wait state on the bus if read by an external master, too.
  Additionally, it is not possible to force the generation of a STOP condition out of the wait state. The reason is, that a master read transfer must be finished with a not-acknowledged followed by a STOP condition to allow the slave to release his SDA line. Otherwise the slave may force the SDA line to 0 (first data bit of next byte) making it impossible to generate the STOP condition (rising edge on SDA).
  To continue operation, the mode must be switched to run mode 0
- Stop Mode 1:
  Same as stop mode 0, but additionally, a master sends a STOP condition to finish the frame.
  If stop mode 1 is requested for a master device after the first byte of a 10 bit address, a stop condition will be sent out. In this case, a slave device will issue an error interrupt.

19.5.2.7 IIC Protocol Interrupt Events

The following protocol-related events are generated in IIC mode and can lead to a protocol interrupt.
Please note that the bits in register PSR are not all automatically cleared by HW and have to be cleared by SW in order to monitor new incoming events.

- Transmit buffer event:
  The transmit buffer event indication flag PSR.TBIF is set when the content of the transmit buffer TBUF has been loaded to the transmit shift register, indicating that the action requested by the TBUF entry has started.
  With this event, bit **TCSRL**.TDV is cleared. This interrupt can be used to write the next TBUF entry while the last one is in progress (handled by the transmitter part).
- Receive event:
  This receive event indication flag PSR.RIF indicates that a new data byte has been written to the receive buffer RBUF0/1 (except for the first address byte of a new frame, that is indicated by an alternative receive interrupt). The flag becomes set when the data byte is received (after the falling edge of SCL). This interrupt can be used to read out the received data while a new data byte can be in progress (handled by the receiver part).
- Alternate receive event:
  The alternative receive event indication flag AIF is based on bit RBUFSR[9] (same as RBUF[9]), indicating that the received byte is the first byte of a frame.
- Protocol interrupt events:
  The IIC protocol related interrupt events are either indicating the reception of symbols or the detection of frame errors (common indication PSR.ERR) or unexpected/wrong TDF codes (common indication PSR.WTDF).
  - start condition received at a correct position in a frame (PSR.SCR)
  - repeated start condition received at a correct position in a frame (PSR.RSCR)
  - stop condition transferred at a correct position in a frame (PSR.PCR)
  - master arbitration lost (PSR.ARL)
  - slave read requested (PSR.SRR)
  - non-acknowledge received (PSR.NACK)
  - start condition not at the expected position in a frame (PSR.ERR)
  - stop condition not at the expected position in a frame (PSR.ERR)
  - 10-bit address interrupted by a stop condition after the first address byte (PSR.ERR)
  - TDF slave code in master mode (PSR.WTDF)
  - TDF master code in slave mode (PSR.WTDF)
  - Reserved TDF code found (PSR.WDTF)
  - Start condition code during a running frame in master mode (PSR.WTDF)
  - Data byte transmission code after transfer direction has been changed to reception (master read) in master mode (PSR.WTDF)

If a wrong TDF code is found in TBUF, the error event is active until the TDF value is either corrected or invalidated. If the related interrupt is enabled, the interrupt handler should check PSR.WDTF first and correct or invalidate TBUF, before dealing with the other possible interrupt events.

19.5.2.8 Receiver Address Acknowledge

After a (repeated) start condition, the master sends a slave address to identify the target device of the communication. The start address can comprise one or two address bytes (for 7 bit or for 10 bit addressing schemes). After an address byte, a slave sensitive to the transmitted address has to acknowledge the reception.
Therefore, the slave's address can be programmed in the device, where it is compared to the received address. In case of a match, the slave answers with an acknowledge (SDA = 0). Slaves that are not targeted answer with an non-acknowledge (SDA = 1).
In addition to the match of the programmed address, an other address byte value has to be answered with an acknowledge if the slave is capable to handle the corresponding requests. The address byte 00_H indicates a general call address, that can be acknowledged. The value 01_H stands for a start byte generation, that is not acknowledged (details see IIC specification, chapter 10).
In order to allow selective acknowledges for the different values of the address byte(s), the following control mechanism is implemented:

- The address byte 00_H is acknowledged if bit PCRH.ACK00 is set.
- The address byte 01_H is not acknowledged.
- The first 7 bits of a received first address byte are compared to the programmed slave address (PCR.SLAD[15:9]). If these bits match, the slave sends an acknowledge. In addition to this, if the slave address is programmed to 1111 $0XX_B$, the slave device waits for a second address byte and compares it also to PCR.SLAD[7:0] and sends an acknowledge accordingly to cover the 10 bit addressing mode. The user has to take care about reserved addresses (refer to IIC specification for more detailed description). Only the address 1111 $0XX_B$ is supported.

Under each of these conditions, bit PSR.SLSEL will be set when the addressing delivered a match. This bit is cleared automatically by a (repeated) start condition.

19.5.2.9 Receiver Handling

A selected slave receiver always acknowledges a received data byte. If the receive buffers RBUF0/1 are already full and can not accept more data, the respective register is overwritten (PSR.DLI becomes set in this case and a protocol interrupt can be generated).
An address reception also uses the registers RBUF0/1 to store the address before checking if the device is selected. The received addresses do not set RDV0/1, so the addresses are not handled like received data.

19.5.2.10 Receiver Status Information

In addition to the received data byte, some IIC protocol related information is stored in the 16 bit data word of the receive buffer. The received data byte is available at the bit positions RBUF[7:0], whereas the additional information is monitored at the bit positions RBUF[12:8]. This structure allows to identify the meaning of each received data byte without reading additional registers, also when using a FIFO data buffer.

- RBUF[8]:
  Value of the received acknowledge bit. This information is also available in RBUFSR[8] as protocol argument.
- RBUF[9]:
  A 1 at this bit position indicates that the data byte has been the first byte of a new frame (address) after a (repeated) start condition. A 0 at this bit position indicates further data bytes. This information is also available in RBUFSR[9], allowing different interrupt routines for the address and data handling.
- RBUF[10]:
  A 1 at this bit position indicates that the data byte has been received when the device has been in slave mode, whereas a 1 indicates a reception in master mode.
- RBUF[11]:
  A 1 at this bit position indicates an incomplete/erroneous data byte in the receive buffer caused by a wrong position of a START or STOP condition in the frame. The bit is not identical to the frame error status bit in PSR, because the bit in the PSR has to be cleared by software (“sticky” bit), whereas RBUF[11] is evaluated data byte by data byte. If RBUF[11] = 0, the received data byte has been correct, independent of former errors.
- RBUF[12]:
  A 0 at this bit position indicates that the programmed address has been received. A 1 indicates a general call address.

19.5.3 Symbol Timing

The symbol timing of the IIC is determined by the master stimulating the shift clock line SCL. It is different for standard and fast IIC mode.

- 100 kBaud standard mode (PCRH.STIM = 0):
  The symbol timing is based on 10 time quanta tq per symbol. A minimum module clock frequency f_{SYS} = 2 MHz is required.
- 400 kBaud standard mode (PCRH.STIM = 1):
  The symbol timing is based on 25 time quanta tq per symbol. A minimum module clock frequency f_{SYS} = 10 MHz is required.

The baud rate setting should only be changed while the transmitter and the receiver are idle or CCR.MODE = 0. The bits in register BRGL define the length of a time quantum tq that is given by one period of f_{PCTQ}.

- BRGL.CTQSEL
  to define the input frequency f_{CTQIN} for the time quanta generation
- BRGL.PCTQ
  to define the length of a time quantum (division of f_{CTQIN} by 1, 2, 3, or 4)
- BRGL.DCTQ
  to define the number of time quanta per symbol (number of tq = DCTQ + 1)

The standard setting is given by CTQSEL = 00_B ($f_{CTQIN} = f_{PDIV}$) and PPPEN = 0 ($f_{PPP} = f_{PIN}$). Under these conditions, the frequency f_{PCTQ} is given by:

(19.10)

$$f_{PCTQ} = f_{PIN} \times \frac{1}{PDIV + 1} \times \frac{1}{PCTQ + 1}$$

To respect the specified SDA hold time of 300 ns after a falling edge of signal SCL, a hold delay t_{HDEL} has been introduced. It also prevents an erroneous detection of a start or a stop condition. The length of this delay can be programmed by bit field PCRH.HDEL. If the HDEL value is programmed to 0, a delay of one period of f_{SYS} is introduced. Taking into account the input sampling and output update, bit field HDEL can be programmed according to:

(19.11)

$$HDEL \geq 300ns \cdot f_{PPP} - \left(3 \cdot \frac{f_{PPP}}{f_{SYS}}\right) + 1$$

If the digital input filter is used, HDEL compensates the filter delay of 2 filter periods (f_{PPP} should be used) in case of a spike on the input signal. This ensures that a data bit on the SDA line changing just before the rising edge or behind the falling edge of SCL won't be treated as a start or stop condition.

19.5.3.1 Start Symbol

Figure 19-42 shows the general start symbol timing.

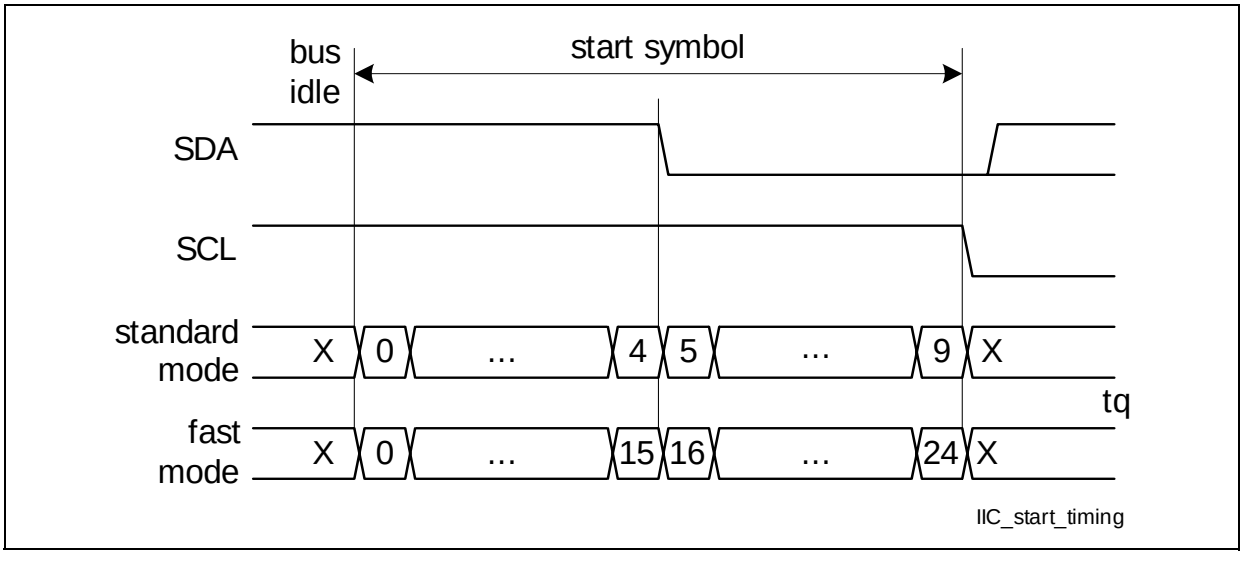


Figure 19-42 Start Symbol Timing

19.5.3.2 Repeated Start Symbol

During the first part of a repeated start symbol, an SCL low value is driven for the specified number of time quanta. Then a high value is output. After the detection of a rising edge at the SCL input, a normal start symbol is generated, as shown in **Figure 19-43**.

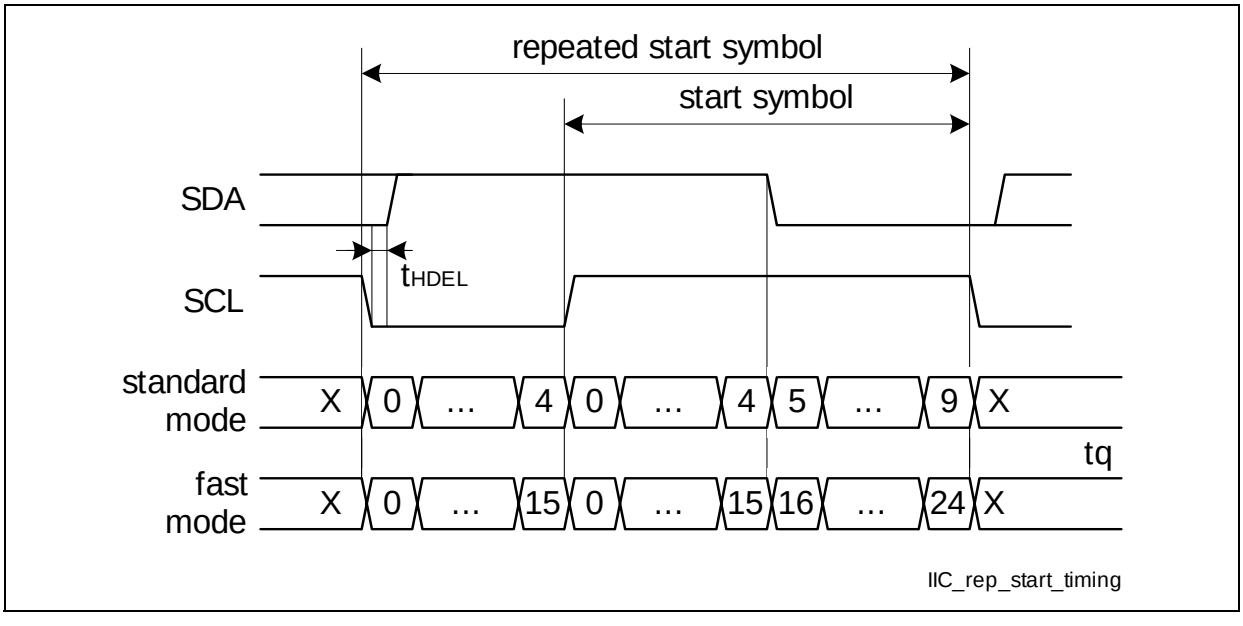


Figure 19-43 Repeated Start Symbol Timing

19.5.3.3 Stop Symbol

Figure 19-44 shows the stop symbol timing.

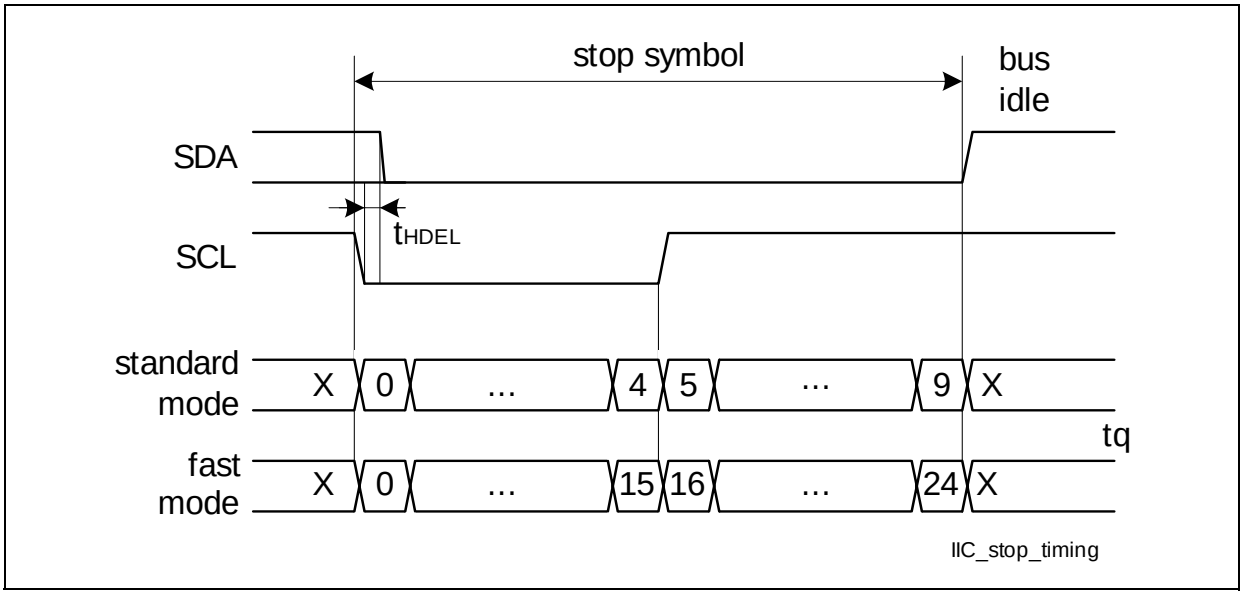


Figure 19-44 Stop Symbol Timing

19.5.3.4 Data Bit Symbol

Figure 19-45 shows the general data bit symbol timing.

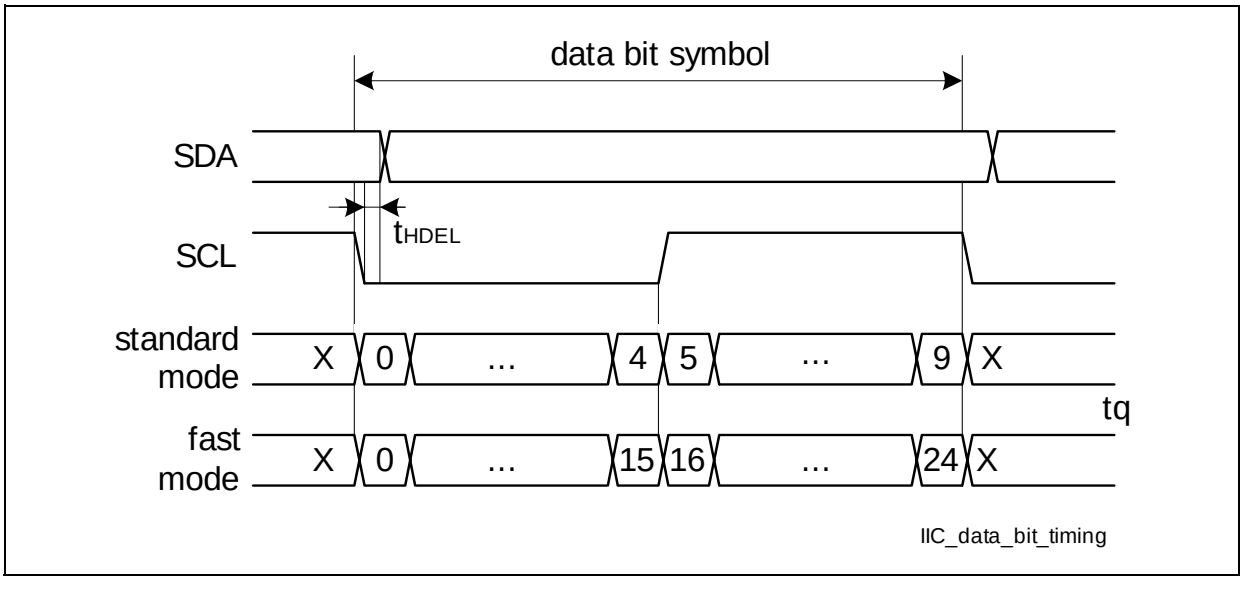


Figure 19-45 Data Bit Symbol

Output SDA changes after the time t_{HDEL} defined by PCRH.HDEL has elapsed if a falling edge is detected at the SCL input to respect the SDA hold time. The value of PCRH.HDEL allows compensation of the delay of the SCL input path (sampling,

filtering).
In the case of an acknowledge transmission, the USIC IIC waits for the receiver indicating that a complete byte has been received. This adds an additional delay of 3 periods of f_{SYS} to the path. The minimum module input frequency has to be selected properly to ensure the SDA setup time to SCL rising edge.

19.5.4 Data Flow Handling

The handling of the data flow and the sequence of the symbols in an IIC frame is controlled by the IIC transmitter part of the USIC communication channel. The IIC bus protocol is byte-oriented, whereas a USIC data buffer word can contain up to 16 data bits. In addition to the data byte to be transmitted (located at TBUF[7:0]), bit field TDF (transmit data format) to control the IIC sequence is located at the bit positions TBUF[10:8]. The TDF code defines for each data byte how it should be transmitted (IIC master or IIC slave), and controls the transmission of (repeated) start and stop symbols. This structure allows the definition of a complete IIC frame for an IIC master device only by writing to TBUFx or by using a FIFO data buffer mechanism, because no other control registers have to be accessed.
If a wrong or unexpected TDF code is encountered (e.g. due to a SW error during setup of the transmit buffer), a stop condition will be sent out by the master. This leads to an abort of the currently running frame. A slave module waits for a valid TDF code and sets SCL = 0. The SW then has to invalidate the unexpected TDF code and write a valid one. Please note that during an arbitration phase in multi-master bus systems an unpredictable bus behavior may occur due to an unexpected stop condition.

19.5.4.1 Transmit Data Formats

The following transmit data formats are available in master mode:

- Send data byte as master (TDF = 000_B):
  This format is used to transmit a data byte from the master to a slave. The transmitter sends its data byte (TBUF[7:0]), receives and checks the acknowledge bit sent by the slave.
- Receive data byte and send acknowledge 0 (TDF = 010_B):
  This format is used by the master to read a data byte from a slave. The master acknowledges the transfer with a 0-level to continue the transfer. The content of TBUF[7:0] is ignored.
- Receive data byte and send acknowledge 1 (TDF = 011_B):
  This format is used by the master to read a data byte from a slave. The master does not acknowledge the transfer with a 1-level to finish the transfer. The content of TBUF[7:0] is ignored.
- Send start condition (TDF = 100_B):
  If TBUF contains this entry while the bus is idle, a start condition will be generated.

The content of TBUF[7:0] is taken as first address byte for the transmission (bits TBUF[7:1] are the address, the LSB is the read/write control).

- Send repeated start condition (TDF = 101_B):
  If TBUF contains this entry and SCL = 0 and a byte transfer is not in progress, a repeated start condition will be sent out if the device is the current master. The current master is defined as the device that has set the start condition (and also won the master arbitration) for the current message. The content of TBUF[7:0] is taken as first address byte for the transmission (bits TBUF[7:1] are the address, the LSB is the read/write control).
- Send stop condition (TDF = 110_B):
  If the current master has finished its last byte transfer (including acknowledge), it sends a stop condition if this format is in TBUF. The content of TBUF[7:0] is ignored.
- TDF = 111_B:
  Reserved and must not be programmed. No additional action except releasing the TBUF entry and setting the error bit in PSR (that can lead to a protocol interrupt).

The following transmit data format is available in slave mode (the symbols in a frame are controlled by the master and the slave only has to send data if it has been “asked” by the master):

- Send data byte as slave (TDF = 001_B):
  This format is used to transmit a data byte from a slave to the master. The transmitter sends its data byte (TBUF[7:0]) plus the acknowledge bit as a 1.

19.5.4.2 Valid Master Transmit Data Formats

Due to the IIC frame formats[1)], only some specific sequences of TDF codes are possible and valid. If the USIC IIC module detects a wrong TDF code in a running frame, the transfer is aborted and flag PCR.WTDF is set. Additionally, an interrupt can be generated if enabled by the user. In case of a wrong TDF code, the frame will be aborted immediately with a STOP condition if the USIC IIC master still owns the SDA line. But if the accessed slave owns the SDA line (read transfer), the master must perform a dummy read with a non-acknowledge so that the slave releases the SDA line before a STOP condition can be sent. The received data byte of the dummy read will be stored in RBUF0/1, but RDV0/1 won't be set. Therefore the dummy read won't generate a receive interrupt and the data byte won't be stored into the receive FIFO.
If the transfer direction has changed in the current frame (master read access), the transmit data request (TDF = 000_B) is not possible and won't be accepted (leading to a wrong TDF Code indication).

Table 19-22 Valid TDF Codes Overview

Frame Position	Valid TDF Codes
first TDF code (master idle)	start (100_B)
read transfer: second TDF code (after start or repeated start)	receive with acknowledge (010_B) or receive with not-acknowledge (011_B)
write transfer: second TDF code (after start or repeated start)	transmit (000_B), repeated start (101_B), or stop (110_B)
read transfer: third and subsequent TDF code after acknowledge	receive with acknowledge (010_B) or receive with not-acknowledge (011_B)
read transfer: third and subsequent TDF code after not-acknowledge	repeated start (101_B) or stop (110_B)
write transfer: third and subsequent TDF code	transmit (000_B), repeated start (101_B), or stop (110_B)

- First TDF code:
  A master transfer starts with the TDF start code (100_B). All other codes are ignored, but no WTDF error will be indicated.
- TDF code after a start (100_B) or repeated start code (101_B) in case of a read access:
  If a master-read transfer is started (determined by the LSB of the address byte = 1), the transfer direction of SDA changes and the slave will actively drive the data line. In this case, only the codes 010_B and 011_B are valid. To abort the transfer in case of

1) Phillips IIC Spec V2.1, chapters 9 and 14.2

a wrong code, a dummy read must be performed by the master before the STOP condition can be generated.

- TDF code after a start (100_B) or repeated start code (101_B) in case of a write access:
  If a master-write transfer is started (determined by the LSB of the address byte = 0), the master still owns the SDA line. In this case, the transmit (000_B), repeated start (101_B) and stop (110_B) codes are valid. The other codes are considered as wrong. To abort the transfer in case of a wrong code, the STOP condition is generated immediately.
- TDF code of the third and subsequent command in case of a read access with acknowledged previous data byte:
  If a master-read transfer is started (determined by the LSB of the address byte), the transfer direction of SDA changes and the slave will actively drive the data line. To force the slave to release the SDA line, the master has to not-acknowledge a byte transfer. In this case, only the receive codes 010_B and 011_B are valid. To abort the transfer in case of a wrong code, a dummy read must be performed by the master before the STOP condition can be generated.
- TDF code of the third and subsequent command in case of a read access with a not-acknowledged previous data byte:
  If a master-read transfer is started (determined by the LSB of the address byte), the transfer direction of SDA changes and the slave will actively drive the data line. To force the slave to release the SDA line, the master has to not-acknowledge a byte transfer. In this case, only the restart (101_B) and stop code (110_B) are valid. To abort the transfer in case of a wrong code, the STOP condition is generated immediately.
- TDF code of the third and subsequent command in case of a write access:
  If a master-write transfer is started (determined by the LSB of the address byte), the master still owns the SDA line. In this case, the transmit (000_B), repeated start (101_B) and stop (110_B) codes are valid. The other codes are considered as wrong. To abort the transfer in case of a wrong code, the STOP condition is generated immediately.
- After a master device has received a non-acknowledge from a slave device, a stop condition will be sent out automatically, except if the following TDF code requests a repeated start condition. In this case, the TDF code is taken into account, whereas all other TDF codes are ignored.

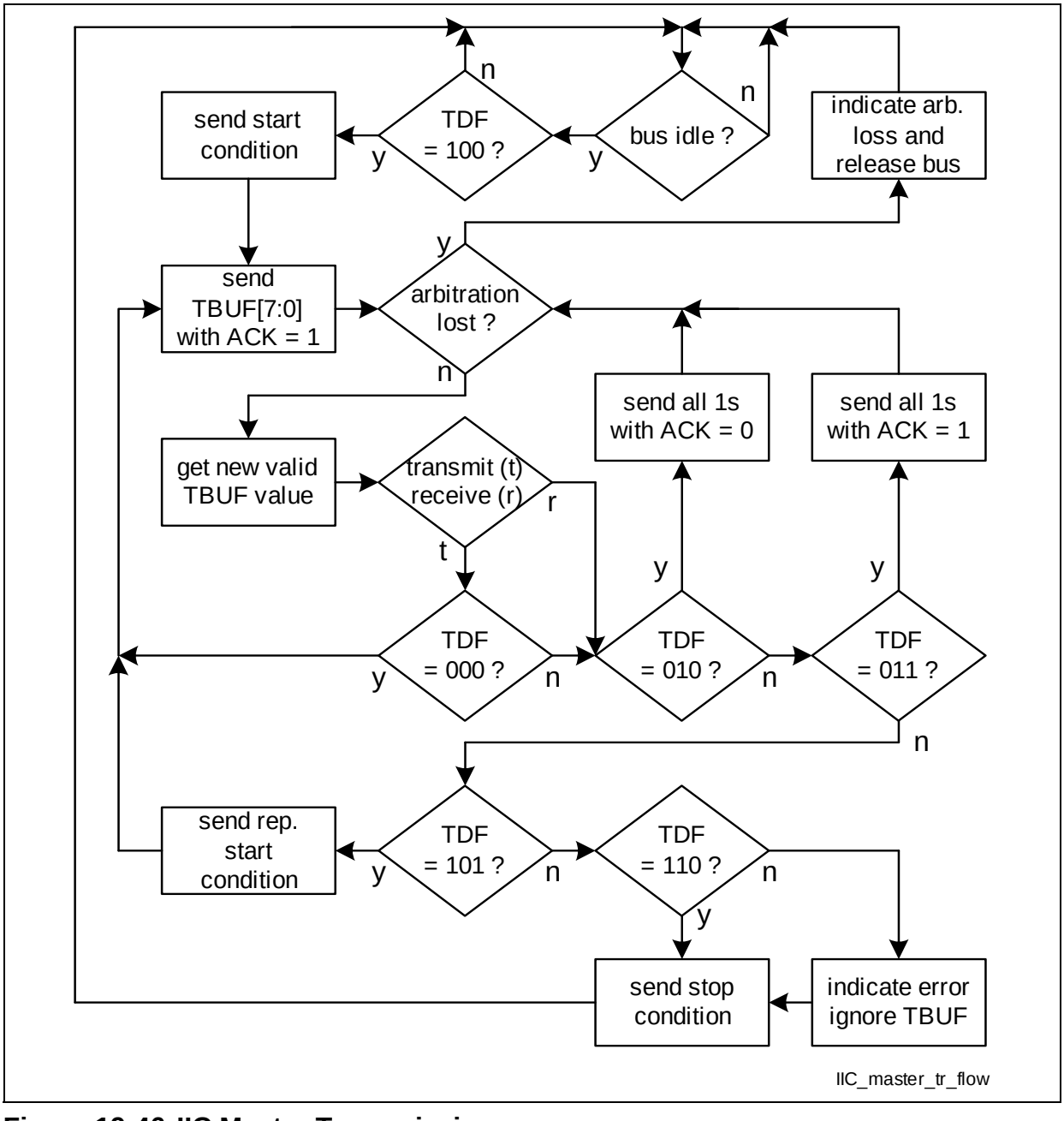


Figure 19-46 IIC Master Transmission

19.5.5 IIC Protocol Registers

In IIC mode, the registers PCRH, PCRL and PSR handle IIC related information.

19.5.5.1 Protocol Control Registers

The following control bits or bit fields are available in the IIC mode.

Table 19-23 Signification of PCRL/H Bits for IIC

Bit	Alias	Description
CTR [15:0]	**SLAD [15:0]**	**Slave Address** This bit field contains the programmed slave address. The corresponding bits in the first received address byte are compared to the bits SLAD[15:9] to check for address match. If SLAD[15:11] = 11110_B, then the second address byte is also compared to SLAD[7:0].
CTR[16]	**ACK00**	**Acknowledge 00_H** This bit defines if a slave device should be sensitive to the slave address 00_H. 0_B The slave device is not sensitive to this address. 1_B The slave device is sensitive to this address.
CTR[17]	**STIM**	**Symbol Timing** This bit defines how many time quanta are used in a symbol. 0_B A symbol contains 10 time quanta. The timing is adapted for standard mode (100 kBaud). 1_B A symbol contains 25 time quanta. The timing is adapted for fast mode (400 kBaud).
CTR[18]	**SCR IEN**	**Start Condition Received Interrupt Enable** This bit enables the generation of a protocol interrupt if a start condition is detected. 0_B The start condition interrupt is disabled. 1_B The start condition interrupt is enabled.
CTR[19]	**RSCR IEN**	**Repeated Start Condition Received Interrupt Enable** This bit enables the generation of a protocol interrupt if a repeated start condition is detected. 0_B The repeated start condition interrupt is disabled. 1_B The repeated start condition interrupt is enabled.

Table 19-23 Signification of PCRL/H Bits for IIC (cont'd)

Bit	Alias	Description
CTR[20]	**PCR IEN**	**Stop Condition Received Interrupt Enable** This bit enables the generation of a protocol interrupt if a stop condition is detected. 0_B The stop condition interrupt is disabled. 1_B The stop condition interrupt is enabled.
CTR[21]	**NACK IEN**	**Non-Acknowledge Interrupt Enable** This bit enables the generation of a protocol interrupt if a non-acknowledge is detected by a master. 0_B The non-acknowledge interrupt is disabled. 1_B The non-acknowledge interrupt is enabled.
CTR[22]	**ARL IEN**	**Arbitration Lost Interrupt Enable** This bit enables the generation of a protocol interrupt if an arbitration lost event is detected. 0_B The arbitration lost interrupt is disabled. 1_B The arbitration lost interrupt is enabled.
CTR[23]	**SRR IEN**	**Slave Read Request Interrupt Enable** This bit enables the generation of a protocol interrupt if a slave read request is detected. 0_B The slave read request interrupt is disabled. 1_B The slave read request interrupt is enabled.
CTR[24]	**ERR IEN**	**Error Interrupt Enable** This bit enables the generation of a protocol interrupt if an IIC error condition is detected (indicated by PSR.ERR or PSR.WTDF). 0_B The error interrupt is disabled. 1_B The error interrupt is enabled.

Table 19-23 Signification of PCRL/H Bits for IIC (cont'd)

Bit	Alias	Description
CTR[25]	**S ACK DIS**	**Slave Acknowledge Disable** This bit disables the generation of an active acknowledge signal for a slave device (active acknowledge = 0 level). Once set by SW, it is automatically cleared with each (repeated) start condition. If this bit is set after a byte has been received (indicated by an interrupt) but before the next acknowledge bit has started, the next acknowledge bit will be sent with passive level. This would indicate that the receiver does not accept more bytes. As a result, a minimum of 2 bytes will be received if the first receive interrupt is used to set this bit. 0_B The generation of an active slave acknowledge is enabled (slave acknowledge with 0 level = more bytes can be received). 1_B The generation of an active slave acknowledge is disabled (slave acknowledge with 1 level = reception stopped).
CTR [29:26]	**HDEL**	**Hardware Delay** This bit field defines the delay used to compensate the internal treatment of the SCL signal (see **Section 19.5.3**) in order to respect the SDA hold time specified for the IIC protocol.
CTR[31]	**MCLK**	**Master Clock Enable** This bit enables generation of the master clock MCLK (not directly used for IIC protocol, can be used as general frequency output). 0_B The MCLK generation is disabled and MCLK is 0. 1_B The MCLK generation is enabled.
others		**Reserved** returns 0 if read; should be written with 0;

19.5.5.2 Protocol Status Register

The following PSR status bits or bit fields are available in IIC mode. Please note that the bits in register PSR are not cleared by HW.
The row marked PI indicates which status bit can generate a protocol interrupt. The general interrupt status flags are described in the general interrupt chapter.

Table 19-24 Signification of PSR Bits for IIC

Bit	Alias	Description	PI?
ST[0]	**SLSEL**	**Slave Select** This bit indicates that this device has been selected as slave. 0_B The device is not selected as slave. 1_B The device is selected as slave.	-
ST[1]	**WTDF**	**Wrong TDF Code Found** This bit indicates that an unexpected/wrong TDF code has been found. A protocol interrupt can be generated if PCRH.ERRIEN = 1. 0_B A wrong TDF code has not been found. 1_B A wrong TDF code has been found.	y
ST[2]	**SCR**	**Start Condition Received** This bit indicates that a start condition has been detected on the IIC bus lines. A protocol interrupt can be generated if PCRH.SCRIEN = 1. 0_B A start condition has not yet been detected. 1_B A start condition has been detected.	y
ST[3]	**RSCR**	**Repeated Start Condition Received** This bit indicates that a repeated start condition has been detected on the IIC bus lines. A protocol interrupt can be generated if PCRH.RSCRIEN = 1. 0_B A repeated start condition has not yet been detected. 1_B A repeated start condition has been detected.	y
ST[4]	**PCR**	**Stop Condition Received** This bit indicates that a stop condition has been detected on the IIC bus lines. A protocol interrupt can be generated if PCRH.PCRIEN = 1. 0_B A stop condition has not yet been detected. 1_B A stop condition has been detected.	y

Table 19-24 Signification of PSR Bits for IIC (cont'd)

Bit	Alias	Description	PI?
ST[5]	**NACK**	**Non-Acknowledge Received** This bit indicates that a non-acknowledge has been received in master mode. This bit is not set in slave mode. A protocol interrupt can be generated if PCRH.NACKIEN = 1. 0_B A non-acknowledge has not been received. 1_B A non-acknowledge has been received.	y
ST[6]	**ARL**	**Arbitration Lost** This bit indicates that an arbitration has been lost. A protocol interrupt can be generated if PCRH.ARLIEN = 1. 0_B An arbitration has not been lost. 1_B An arbitration has been lost.	y
ST[7]	**SRR**	**Slave Read Request** This bit indicates that a slave read request has been detected. It becomes active to request the first data byte to be made available in the transmit buffer. For further consecutive data bytes, the transmit buffer issues more interrupts. For the end of the transfer, the master transmitter sends a stop condition. A protocol interrupt can be generated if PCRH.SRRIEN = 1. 0_B A slave read request has not been detected. 1_B A slave read request has been detected.	y
ST[8]	**ERR**	**Error** This bit indicates that an IIC error (frame format or TDF code) has been detected. A protocol interrupt can be generated if PCRH.ERRIEN = 1. 0_B An IIC error has not been detected. 1_B An IIC error has been detected.	y
others		**Reserved** returns 0 if read; should be written with 0;	-

19.6 IIS Protocol

This chapter describes how the USIC module handles the IIS protocol. This serial protocol can handle reception and transmission of synchronous data frames between a device operating in master mode and a device in slave mode. An IIS connection based on a USIC communication channel supports half-duplex and full-duplex data transfers. The IIS mode is selected by CCR.MODE = 0011_B with CCFG.IIS = 1 (IIS mode is available).

This chapter contains the following sections:

- Introduction (see **Section 19.6.1**)
- General IIS issues (see **Section 19.6.2**)
- Master mode operation (see **Section 19.6.3**)
- Slave mode operation (see **Section 19.6.4**)
- Protocol registers (see **Chapter 19.6.5**)

19.6.1 Introduction

The IIS protocol is a synchronous serial communication protocol mainly for audio and infotainment applications and refers to the Philips specification, 1986, revised June 5, 1996.

19.6.1.1 Signal Description

A connection between an IIS master and an IIS slave is based on the following signals:

- A shift clock signal SCK, generated by the transfer master. It is permanently generated while an IIS connection is established, also while no valid data bits are transferred.
- A word address signal WA (also named WS), generated by the transfer master. It indicates the beginning of a new data word and the targeted audio channel (e.g. left/ right). The word address output signal WA is available on all SELOx outputs if the WA generation is enabled (by PCR.WAGEN = 1 for the transfer master). The WA signal changes synchronously to the falling edges of the shift clock.
- If the transmitter is the IIS master device, it generates a master transmit slave receive data signal. The data changes synchronously to the falling edges of the shift clock.
- If the transmitter is the IIS slave device, it generates a master receive slave transmit data signal. The data changes synchronously to the falling edges of the shift clock.

The transmitter part and the receiver part of the USIC communication channel can be used together to establish a full-duplex data connection between an IIS master and a slave device.

Table 19-25 IIS IO Signals

IIS Mode	Receive Data	Transmit Data	Shift Clock	Word Address
master	input DIN, handled by DX0	output DOUT	output SCLKOUT	output(s) SELOx
slave	input DIN, handled by DX0	output DOUT	input SCLKIN, handled by DX1	input SELIN, handled by DX2

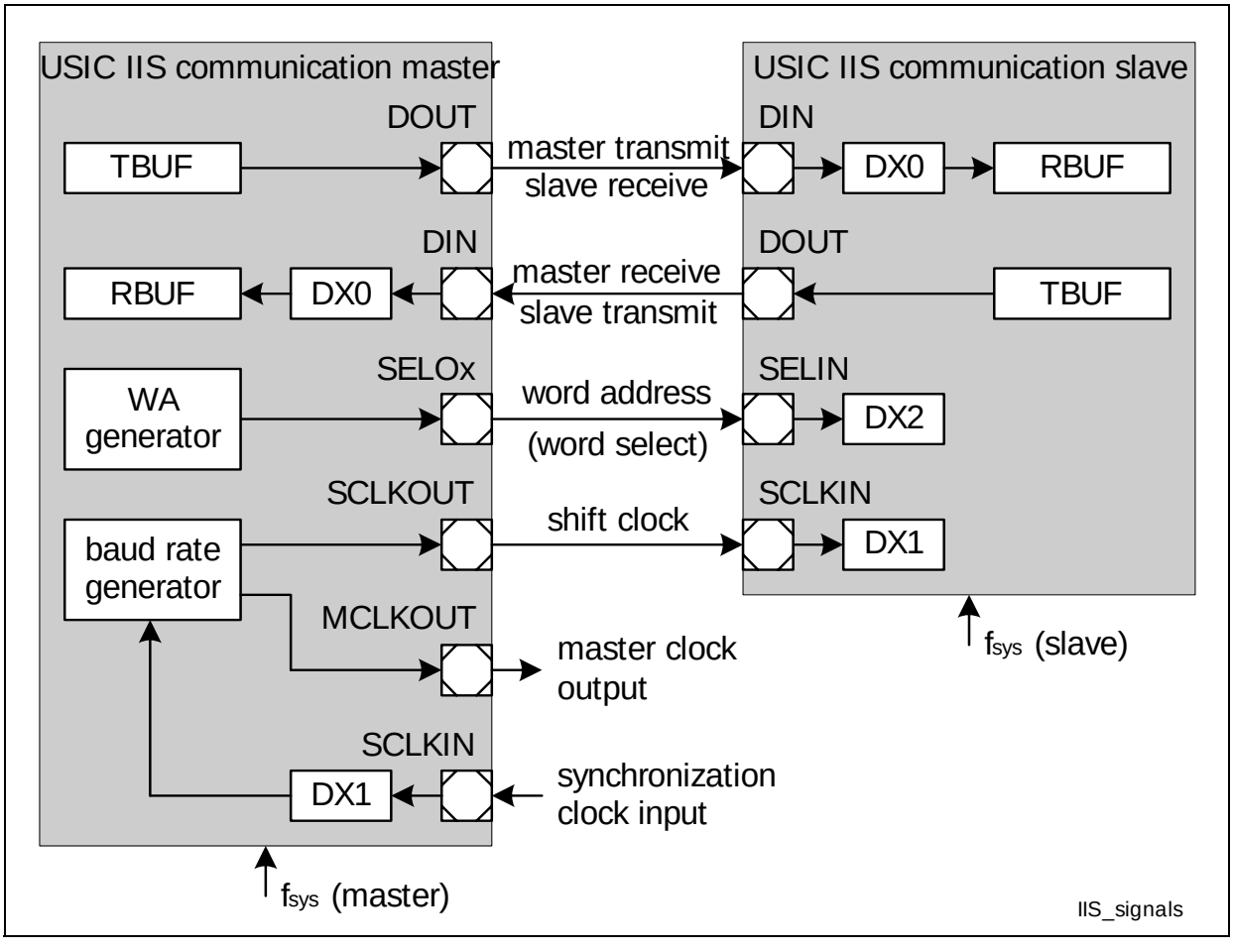


Figure 19-47 IIS Signals

Two additional signals are available for the USIC IIS communication master:

- A master clock output signal MCLKOUT with a fixed phase relation to the shift clock to support oversampling for audio components. It can also be used as master clock output of a communication network with synchronized IIS connections.
- A synchronization clock input SCLKIN for synchronization of the shift clock generation to an external frequency to support audio frequencies that can not be directly derived from the system clock f_{SYS} of the communication master. It can be

used as master clock input of a communication network with synchronized IIS connections.

19.6.1.2 Protocol Overview

An IIS connection supports transfers for two different data frames via the same data line, e.g. a data frames for the left audio channel and a data frame for the right audio channel. The word address signal WA is used to distinguish between the different data frames. Each data frame can consist of several data words.
In a USIC communication channel, data words are tagged for being transmitted for the left or for the right channel. Also the received data words contain a tag identifying the WA state when the data has been received.

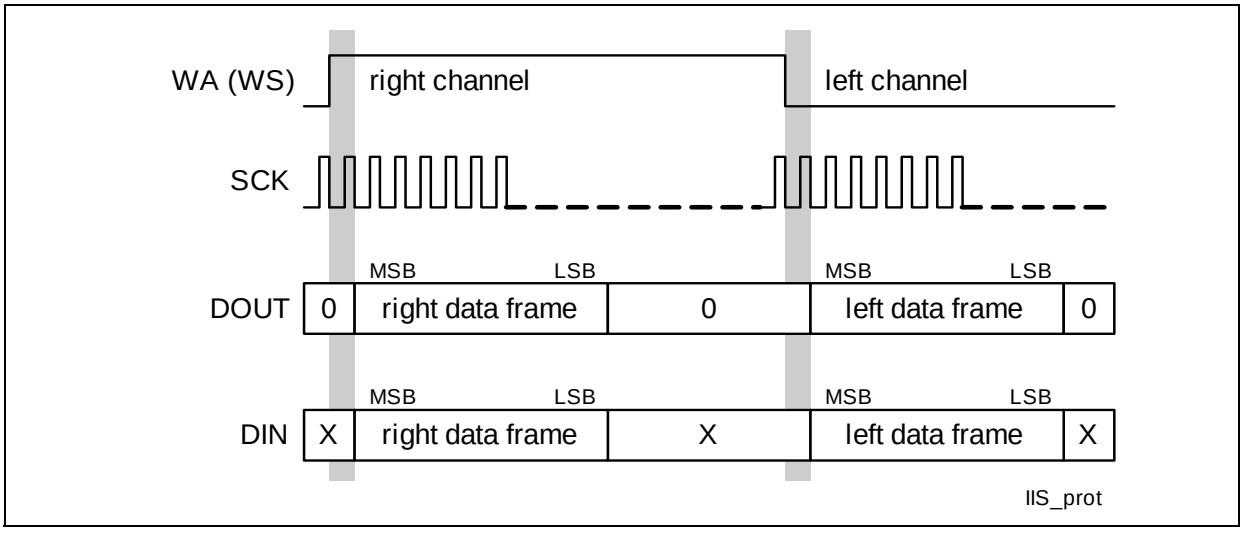


Figure 19-48 Protocol Overview

19.6.1.3 Transfer Delay

The transfer delay feature allows the transfer of data (transmission and reception) with a programmable delay (counted in shift clock periods).

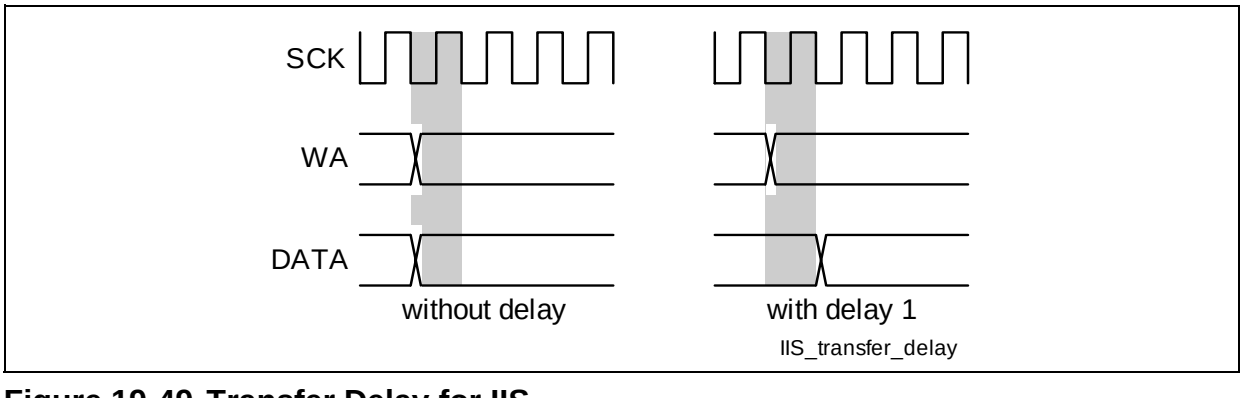


Figure 19-49 Transfer Delay for IIS

19.6.1.4 Connection of External Audio Components

The IIS signals can be used to communicate with external audio devices (such as Codecs) or other audio data sources/destinations.

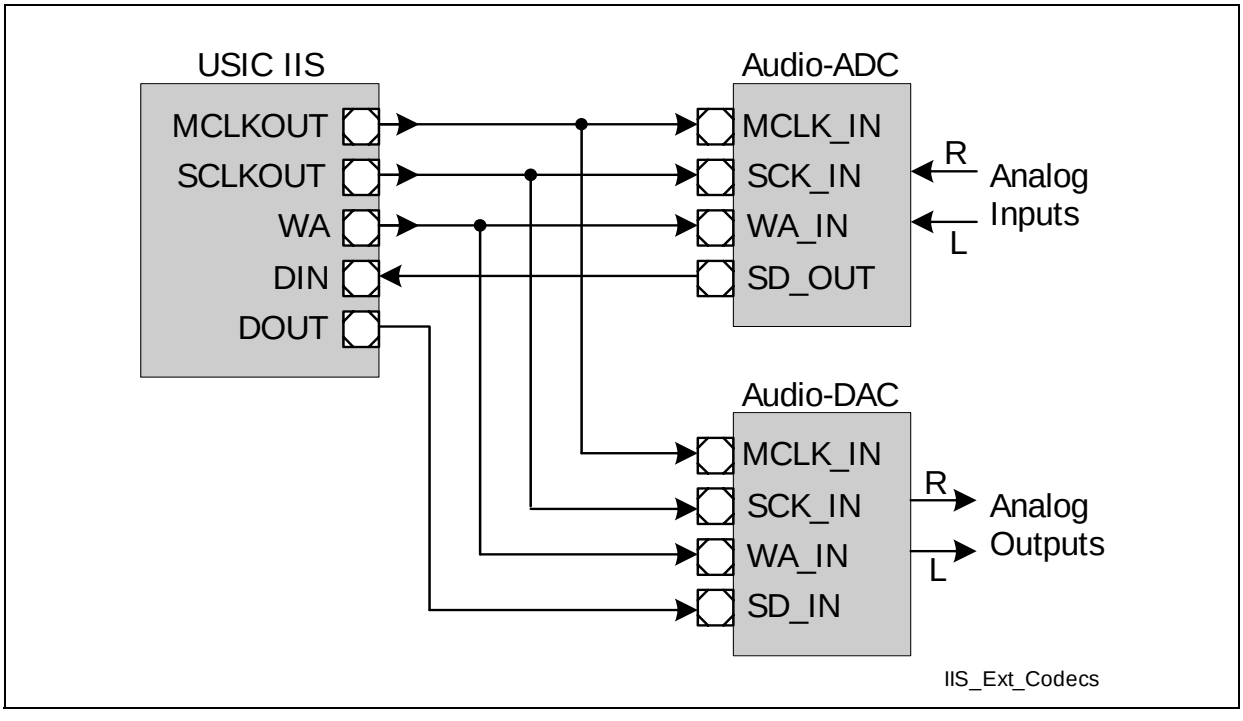


Figure 19-50 Connection of External Audio Devices

In some applications, especially for Audio-ADCs or Audio-DACs, a master clock signal is required with a fixed phase relation to the shift clock signal. The frequency of MCLKOUT is a multiple of the shift frequency SCLKOUT. This factor defines the oversampling factor of the external device (commonly used values: 256 or 384).

19.6.2 Operating the IIS

This chapter contains IIS issues, that are of general interest and not directly linked to master mode or slave mode.

19.6.2.1 Frame Length and Word Length Configuration

After each change of the WA signal, a complete data frame is intended to be transferred (frame length ≤ system word length). The number of data bits transferred after a change of signal WA is defined by SCTRH.FLE. A data frame can consist of several data words with a data word length defined by SCTRH.WLE. The changes of signal WA define the system word length as the number of SCLK cycles between two changes of WA (number of bits available for the right channel and same number available for the left channel).
If the system word length is longer than the frame length defined by SCTRH.FLE, the additional bits are transmitted with passive data level (SCTRL.PDL). If the system word length is smaller than the device frame length, not all LSBs of the transmit data can be transferred.
It is recommended to program bits WLEMD, FLEMD and SELMD in register TCSRL to 0.

19.6.2.2 Automatic Shadow Mechanism

The baud rate and shift control setting are internally kept constant while a data frame is transferred by an automatic shadow mechanism. The registers can be programmed all the time with new settings that are taken into account for the next data frame. During a data frame, the applied (shadowed) setting is not changed, although new values have been written after the start of the data frame. The setting is internally “frozen” with the start of each data frame.
Although this shadow mechanism being implemented, it is recommended to change the baud rate and shift control setting only while the IIS protocol is switched off.

19.6.2.3 Mode Control Behavior

In IIS mode, the following kernel modes are supported:

- Run Mode 0/1:
  Behavior as programmed, no impact on data transfers.
- Stop Mode 0/1:
  Bit PCRL.WAGEN is internally considered as 0 (the bit itself is not changed). If WAGEN = 1, then the current system word cycle is finished and then the WA generation is stopped, but PSR.END is not set. The complete data frame is finished before entering stop mode, including a possible delay due to PCRH.TDEL.
  When leaving a stop mode with WAGEN = 1, the WA generation starts from the beginning.

19.6.2.4 Transfer Delay

The transfer delay can be used to synchronize a data transfer to an event (e.g. a change of the WA signal). This event has to be synchronously generated to the falling edge of the shift clock SCK (like the change of the transmit data), because the input signal for the event is directly sampled in the receiver (as a result, the transmitter can use the detection information with its next edge).
Event signals that are asynchronous to the shift clock while the shift clock is running must not be used. In the example in **Figure 19-49**, the event (change of signal WA) is generated by the transfer master and as a result, is synchronous to the shift clock SCK. With the rising edge of SCK, signal WA is sampled and checked for a change. If a change is detected, a transfer delay counter TDC is automatically loaded with its programmable reload value (PCRH.TDEL), otherwise it is decremented with each rising edge of SCK until it reaches 0, where it stops. The transfer itself is started if the value of TDC has become 0. This can happen under two conditions:

- TDC is reloaded with a PCRH.TDEL = 0 when the event is detected
- TDC has reached 0 while counting down

The transfer delay counter is internal to the IIS protocol pre-processor and can not be observed by SW. The transfer delay in SCK cycles is given by PCRH.TDEL+1.

In the example in **Figure 19-51**, the reload value PCRH.TDEL for TDC is 0. When the samples taken on receiver side show the change of the WA signal, the counter TDC is reloaded. If the reload value is 0, the data transfer starts with 1 shift clock cycle delay compared to the change of WA.

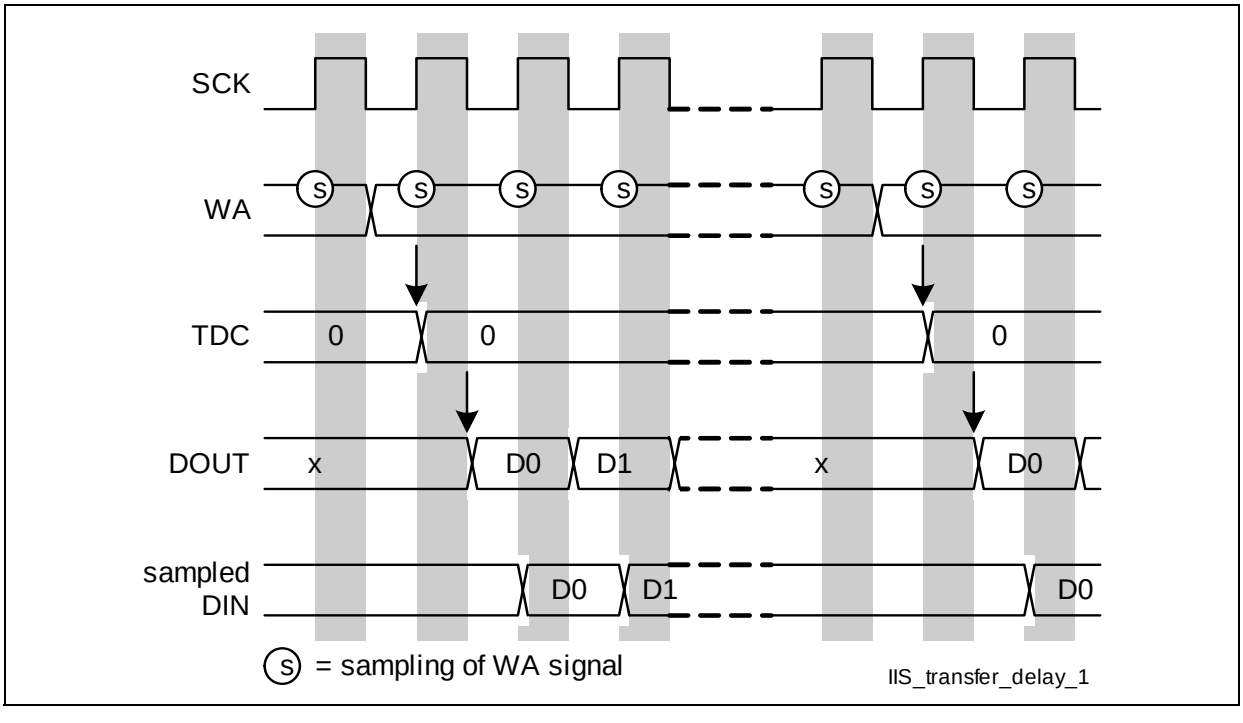


Figure 19-51 Transfer Delay with Delay 1

The ideal case without any transfer delay is shown in **Figure 19-52**. The WA signal changes and the data output value become valid at the same time. This implies that the transmitter “knows” in advance that the event signal will change with the next rising edge of TCLK. This is achieved by delaying the data transmission after the previously detected WA change the system word length minus 1.

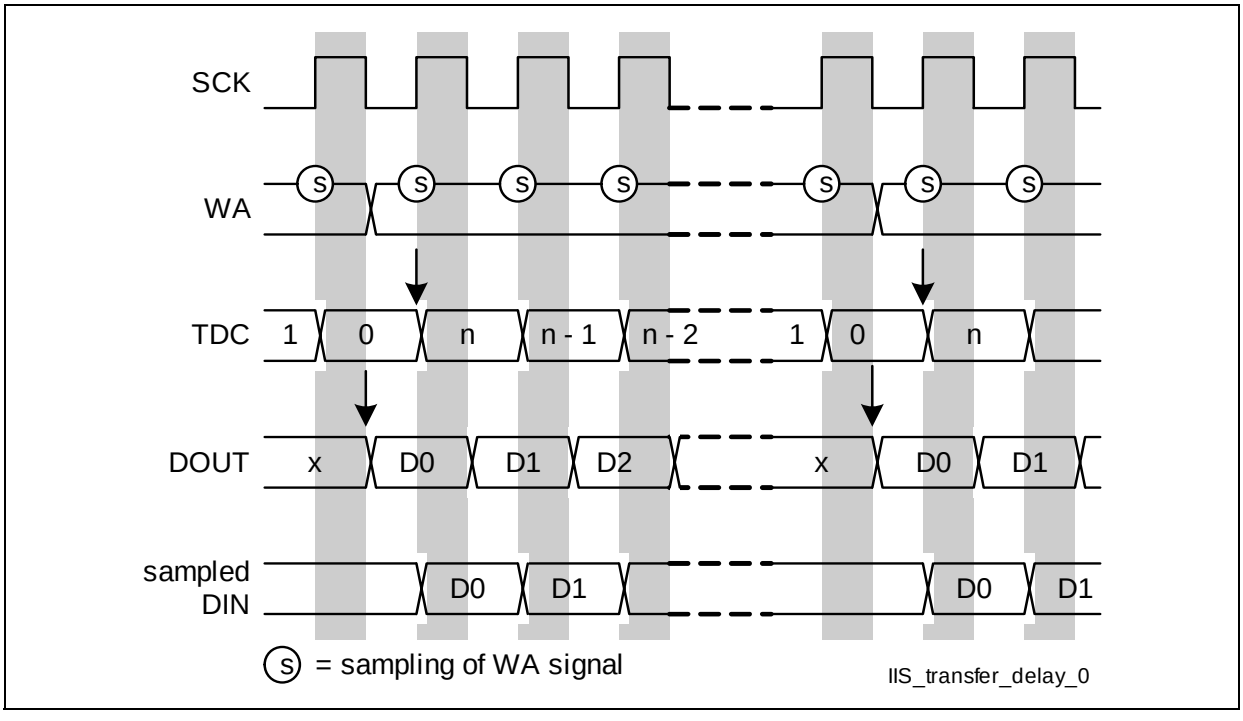


Figure 19-52 Transfer Delay with 0 Delay

If the end of the transfer delay is detected simultaneously to change of WA, the transfer is started and the delay counter is reloaded with PCRH.TDEL. This allows to run the USIC as IIS device without any delay. In this case, internally the delay from the previous event elapses just at the moment when a new event occurs. If PCRH.TDEL is set to a value bigger than the system word length, no transfer takes place.

19.6.2.5 Parity Mode

Parity generation is not supported in IIS mode and bit field CCR.PM = 00_B has to be programmed.

19.6.2.6 Transfer Mode

In IIS mode, bit field SCTRL.TRM = 11_B has to be programmed to allow data transfers. Setting SCTRL.TRM = 00_B disables and stops the data transfer immediately.

19.6.2.7 Data Transfer Interrupt Handling

The data transfer interrupts indicate events related to IIS frame handling.

- Transmit buffer interrupt TBI:
  Bit PSR.TBIF is set after the start of first data bit of a data word.
- Transmit shift interrupt TSI:
  Bit PSR.TSIF is set after the start of the last data bit of a data word.
- Receiver start interrupt RSI:
  Bit PSR.RSIF is set after the reception of the first data bit of a data word.
  With this event, bit **TCSRL**.TDV is cleared and new data can be loaded to the transmit buffer.
- Receiver interrupt RI and alternative interrupt AI:
  Bit PSR.RIF is set at after the reception of the last data bit of a data word with WA = 0.
  Bit RBUFSR.SOF indicates whether the received data word has been the first data word of a new data frame.
- Alternative interrupt AI:
  Bit PSR.AIF is set at after the reception of the last data bit of a data word with WA = 1.
  Bit RBUFSR.SOF indicates whether the received data word has been the first data word of a new data frame.

19.6.2.8 Protocol-Related Argument and Error

In order to distinguish between data words received for the left or the right channel, the IIS protocol pre-processor samples the level of the WA input (just after the WA transition) and propagates it as protocol-related error (although it is not an error, but an indication) to the receive buffer status register at the bit position RBUFSR[9]. This bit position defines if either a standard receive interrupt (if RBUFSR[9] = 0) or an alternative receive interrupt (if RBUFSR[9] = 1) becomes activated when a new data word has been received. Incoming data can be handled by different interrupts or DMA mechanisms for the left and the right channel if the corresponding events are directed to different interrupt nodes. Flag PAR is always 0.

19.6.2.9 Transmit Data Handling

The IIS protocol pre-processor allows to distinguish between the left and the right channel for data transmission. Therefore, bit TCSRL.WA indicates on which channel the data in the buffer will be transmitted. If TCSRL.WA = 0, the data will be transmitted after a falling edge of WA. If TCSRL.WA = 1, the data will be transmitted after a rising edge of WA. The WA value sampled after the WA transition is considered to distinguish between both channels (referring to PSR.WA).
Bit TCSRL.WA can be automatically updated by the transmit control information TCI[4] for each data word if TCSRL.WAMD = 1. In this case, data written to TBUF[15:0] (or IN[15:0] if a FIFO data buffer is used) is considered as left channel data, whereas data

written to TBUF[31:16] (or IN[31:16] if a FIFO data buffer is used) is considered as right channel data.

19.6.2.10 Receive Buffer Handling

If a receive FIFO buffer is available (CCFG.RB = 1) and enabled for data handling (RBCTRH.SIZE > 0), it is recommended to set RBCTRH.RCIM = 11_B in IIS mode. This leads to an indication that the data word has been the first data word of a new data frame if bit OUTRH.RCI[0] = 1, and the channel indication by the sampled WA value is given by OUTRH.RCI[4].
The standard receive buffer event and the alternative receive buffer event can be used for the following operation in RCI mode (RBCTRH.RNM = 1):

- A standard receive buffer event indicates that a data word can be read from OUTRL that belongs to a data frame started when WA = 0.
- An alternative receive buffer event indicates that a data word can be read from OUTRL that belongs to a data frame started when WA = 1.

19.6.2.11 Loop-Delay Compensation

The synchronous signaling mechanism of the IIS protocol being similar to the one of the SSC protocol, the closed-loop delay has to be taken into account for the application setup. In IIS mode, loop-delay compensation in master mode is also possible to achieve higher baud rates.
Please refer to the more detailed description in the SSC chapter.

19.6.3 Operating the IIS in Master Mode

In order to operate the IIS in master mode, the following issues have to be considered:

- Select IIS mode:
  It is recommended to configure all parameters of the IIS that do not change during run time while CCR.MODE = 0000_B. Bit field SCTRL.TRM = 11_B has to be programmed. The configuration of the input stages has to be done while CCR.MODE = 0000_B to avoid unintended edges of the input signals and the IIS mode can be enabled by CCR.MODE = 0011_B afterwards.
- Pin connection for data transfer:
  Establish a connection of input stage DX0 with the selected receive data input pin (DIN) with DX0CR.INSW = 1. Configure a transmit data output pin (DOUT) for a transmitter.
  The data shift unit allowing full-duplex data transfers based on the same WA signal, the values delivered by the DX0 stage are considered as data bits (receive function can not be disabled independently from the transmitter). To receive IIS data, the transmitter does not necessarily need to be configured (no assignment of DOUT signal to a pin).
- Baud rate generation:
  The desired baud rate setting has to be selected, comprising the fractional divider and the baud rate generator. Bit DX1CR.INSW = 0 has to be programmed to use the baud rate generator output SCLK directly as input for the data shift unit. Configure a shift clock output pin with the inverted signal SCLKOUT without additional delay (BRGH.SCLKCFG = 01_B).
- Word address WA generation:
  The WA generation has to be enabled by setting PCRL.WAGEN = 1 and the programming of the number of shift clock cycles between the changes of WA. Bit DX2CR.INSW = 0 has to be programmed to use the WA generator as input for the data shift unit. Configure WA output pin for signal SELOx if needed.
- Data format configuration:
  The word length, the frame length, and the shift direction have to be set up according to the application requirements by programming the registers SCTRL and SCTRH. Generally, the MSB is shifted first (SCTRL.SDIR = 1).
  Bit TCSRL.WAMD can be set to use the transmit control information TCI[4] to distinguish the data words for transmission while WA = 0 or while WA = 1.

19.6.3.1 Baud Rate Generation

The baud rate is defined by the frequency of the SCLK signal (one period of f_{SCLK} represents one data bit).
If the fractional divider mode is used to generate f_{PIN}, there can be an uncertainty of one period of f_{SYS} for f_{PIN}. This uncertainty does not accumulate over several SCLK cycles. As a consequence, the average frequency is reached, whereas the duty cycle of 50% of

the SCLK and MCLK signals can vary by one period of f_{SYS}.
In IIS applications, where the phase relation between the optional MCLK output signal and SCLK is not relevant, SCLK can be based on the frequency f_{PIN} (BRGL.PPPEN = 0).
In the case that a fixed phase relation between the MCLK signal and SCLK is required (e.g. when using MCLK as clock reference for external devices), the additional divider by 2 stage has to be taken into account (BRGL.PPPEN = 1). This division is due to the fact that signal MCLK toggles with each cycle of f_{PIN}. Signal SCLK is then based on signal MCLK, see **Figure 19-53**.

The adjustable integer divider factor is defined by bit field BRGH.PDIV.

(19.12)

$$f_{SCLK} = \begin{cases} \frac{f_{PIN}}{2} \times \frac{1}{PDIV + 1} & \text{if} \quad PPPEN = 0 \\ \frac{f_{PIN}}{2 \times 2} \times \frac{1}{PDIV + 1} & \text{if} \quad PPPEN = 1 \end{cases}$$

Note: In the IIS protocol, the master (unit generating the shift clock and the WA signal) changes the status of its data and WA output line with the falling edge of SCK. The slave transmitter also has to transmit on falling edges. The sampling of the received data is done with the rising edges of SCLK. The input stage DX1 and the SCLKOUT have to be programmed to invert the shift clock signal to fit to the internal signals.

19.6.3.2 WA Generation

The word address (or word select) line WA regularly toggles after N cycles of signal SCLK. The time between the changes of WA is called system word length and can be programmed by using the following bit fields.

In IIS master mode, the system word length is defined by:

- BRGL.CTQSEL = 10_B
  to base the WA toggling on SCLK
- BRGL.PCTQ
  to define the number N of SCLK cycles per system word length
- BRGL.DCTQ
  to define the number N of SCLK cycles per system word length

(19.13)

$$N = (PCTQ + 1) \times (DCTQ + 1)$$

19.6.3.3 Master Clock Output

The master clock signal MCLK can be generated by the master of the IIS transfer (BRGL.PPPEN = 1). It is used especially to connect external Codec devices. It can be configured by bit BRGH.MCLKCFG in its polarity to become the output signal MCLKOUT.

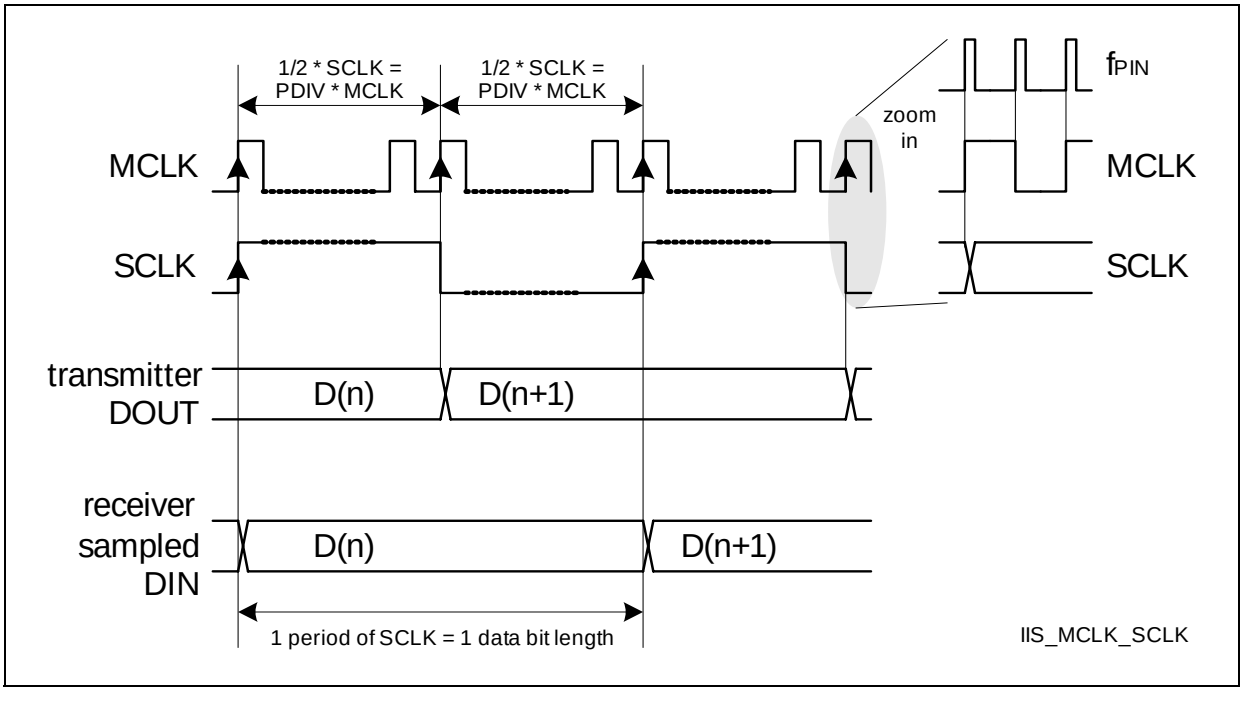


Figure 19-53 MCLK and SCLK for IIS

19.6.3.4 Protocol Interrupt Events

The following protocol-related events are generated in IIS mode and can lead to a protocol interrupt.
Please note that the bits in register PSR are not all automatically cleared by HW and have to be cleared by SW in order to monitor new incoming events.

- WA rising/falling edge events:
  The WA generation block indicates two events that are monitored in register PSR. Flag PSR.WAFE is set with the falling edge, flag PSR.WARE with the rising edge of the WA signal. A protocol interrupt can be generated if PCRL.WAFEIEN = 1 for the falling edge, similar for PCRL.WAREIEN = 1 for a rising edge.
- WA end event:
  The WA generation block also indicates when it has stopped the WA generation after it has been disabled by writing PCRL.WAGEN = 0. A protocol interrupt can be generated if PCRL.ENDIEN = 1.
- DX2T event:
  An activation of the trigger signal DX2T is indicated by PSR.DX2TEV = 1 and can generate a protocol interrupt if PCRL.DX2TIEN = 1. This event can be evaluated instead of the WA rising/falling events if a delay compensation like in SSC mode (for details, refer to corresponding SSC section) is used.

19.6.4 Operating the IIS in Slave Mode

In order to operate the IIS in slave mode, the following issues have to be considered:

- Select IIS mode:
  It is recommended to configure all parameters of the IIS that do not change during run time while CCR.MODE = 0000_B. Bit field SCTRL.TRM = 11_B has to be programmed. The configuration of the input stages has to be done while CCR.MODE = 0000_B to avoid unintended edges of the input signals and the IIS mode can be enabled by CCR.MODE = 0011_B afterwards.
- Pin connection for data transfer:
  Establish a connection of input stage DX0 with the selected receive data input pin (DIN) with DX0CR.INSW = 1. Configure a transmit data output pin (DOUT) for a transmitter.
  The data shift unit allowing full-duplex data transfers based on the same WA signal, the values delivered by the DX0 stage are considered as data bits (receive function can not be disabled independently from the transmitter). To receive IIS data, the transmitter does not necessarily need to be configured (no assignment of DOUT signal to a pin).
- Pin connection for shift clock:
  Establish a connection of input stage DX1 with the selected shift clock input pin (SCLKIN) with DX1CR.INSW = 1 and with inverted polarity (DX1CR.DPOL = 1).
- Pin connection for WA input:
  Establish a connection of input stage DX2 with the WA input pin (SELIN) with DX2CR.INSW = 1.
- Baud rate generation:
  The baud rate generator is not needed and can be switched off by the fractional divider.
- WA generation:
  The WA generation is not needed and can be switched off (PCRL.WAGEN = 0).

19.6.4.1 Protocol Events and Interrupts

The following protocol-related event is generated in IIS mode and can lead to a protocol interrupt.
Please note that the bits in register PSR are not all automatically cleared by HW and have to be cleared by SW in order to monitor new incoming events.

- WA rising/falling/end events:
  The WA generation being switched off, these events are not available.
- DX2T event:
  An activation of the trigger signal DX2T is indicated by PSR.DX2TEV = 1 and can generate a protocol interrupt if PCRL.DX2TIEN = 1.

19.6.5 IIS Protocol Registers

In IIS mode, the bits in the registers PCRL, PCRH and PSR handle IIS related information.

19.6.5.1 Protocol Control Registers

The following PCRL/PCRH control bits or bit fields are available in IIS mode.

Table 19-26 Signification of PCRL/H Bits for IIS

Bit	Alias	Description
CTR[0]	**WAGEN**	**WA Generation Enable** This bit enables/disables the generation of word address control output signal WA. 0_B The IIS can be used as slave. The generation of the word address signal is disabled. The output signal WA is 0. The MCLKO signal generation depends on PCRH.MCLK. 1_B The IIS can be used as master. The generation of the word address signal is enabled. The signal starts with a 0 after being enabled. The generation of MCLK is enabled, independent of PCRH.MCLK. After clearing WAGEN, the USIC module stops the generation of the WA signal within the next 4 WA periods.
CTR[1]	**DTEN**	**Data Transfers Enable** This bit enables/disables the transfer of IIS frames as a reaction to changes of the input word address control line WA. 0_B The changes of the WA input signal are ignored and no transfers take place. 1_B Transfers are enabled.
CTR[2]	**SELINV**	**Select Inversion** This bit defines if the polarity of the SELOx outputs in relation to the internally generated word address signal WA. 0_B The SELOx outputs have the same polarity as the WA signal. 1_B The SELOx outputs have the inverted polarity to the WA signal.

Table 19-26 Signification of PCRL/H Bits for IIS (cont'd)

Bit	Alias	Description
CTR[4]	**WAFE IEN**	**WA Falling Edge Interrupt Enable** This bit enables/disables the activation of a protocol interrupt when a falling edge of WA has been generated. 0_B A protocol interrupt is not activated if a falling edge of WA is generated. 1_B A protocol interrupt is activated if a falling edge of WA is generated.
CTR[5]	**WARE IEN**	**WA Rising Edge Interrupt Enable** This bit enables/disables the activation of a protocol interrupt when a rising edge of WA has been generated. 0_B A protocol interrupt is not activated if a rising edge of WA is generated. 1_B A protocol interrupt is activated if a rising edge of WA is generated.
CTR[6]	**END IEN**	**END Interrupt Enable** This bit enables/disables the activation of a protocol interrupt when the WA generation stops after clearing PCR.WAGEN (complete system word length is processed before stopping). 0_B A protocol interrupt is not activated. 1_B A protocol interrupt is activated.
CTR[15]	**DX2T IEN**	**DX2T Interrupt Enable** This bit enables/disables the generation of a protocol interrupt if the DX2T signal becomes activated (indicated by PSR.DX2TEV = 1). 0_B A protocol interrupt is not generated if DX2T is active. 1_B A protocol interrupt is generated if DX2T is active.
CTR [21:16]	**TDEL**	**Transfer Delay** This bit field defines the transfer delay when an event is detected. If bit field TDEL = 0, the additional delay functionality is switched off and a delay of one shift clock cycle is introduced.
CTR[31]	**MCLK**	**Master Clock Enable** This bit enables generation of the master clock MCLK. 0_B The MCLK generation is disabled and MCLK is 0. 1_B The MCLK generation is enabled.
others		**Reserved** returns 0 if read; should be written with 0;

19.6.5.2 Protocol Status Register PSR

The following PSR status bits or bit fields are available in IIS mode. Please note that the bits in register PSR are not cleared by HW.
The row marked PI indicates which status bit can generate a protocol interrupt in IIS mode. The general interrupt status flags are described in the general interrupt chapter.

Table 19-27 Signification of PSR Bits for IIS

Bit	Alias	Description	PI?
ST[0]	**WA**	**Word Address** This bit indicates the status of the WA input signal, sampled after a transition of WA has been detected. This information is forwarded to the corresponding bit position RBUFSR[9] to distinguish between data received for the right and the left channel. 0_B WA has been sampled 0. 1_B WA has been sampled 1.	-
ST[1]	**DX2S**	**DX2S Status** This bit indicates the current status of the DX2S signal, which is used as word address signal WA. 0_B DX2S is 0. 1_B DX2S is 1.	-
ST[3]	**DX2T EV**	**DX2T Event Detected** This bit indicates that the DX2T signal has been activated. In IIS slave mode, an activation of DX2T generates a protocol interrupt if PCRL.DX2TIEN = 1. 0_B The DX2T signal has not been activated. 1_B The DX2T signal has been activated.	y
ST[4]	**WAFE**	**WA Falling Edge Event** This bit indicates that a falling edge of the WA output signal has been generated. This event generates a protocol interrupt if PCRL.WAFEIEN = 1. 0_B A WA falling edge has not been generated. 1_B A WA falling edge has been generated.	y
ST[5]	**WARE**	**WA Rising Edge Event** This bit indicates that a rising edge of the WA output signal has been generated. This event generates a protocol interrupt if PCRL.WAREIEN = 1. 0_B A WA rising edge has not been generated. 1_B A WA rising edge has been generated.	y

Table 19-27 Signification of PSR Bits for IIS (cont'd)

Bit	Alias	Description	PI?
ST[6]	**END**	**WA Generation End** This bit indicates that the WA generation has ended after clearing PCRL.WAGEN. This bit should be cleared by SW before clearing WAGEN. 0_B The WA generation has not yet ended (if it is running and WAGEN has been cleared). 1_B The WA generation has ended (if it has been running).	y
others		**Reserved**	

19.7 USIC Implementation in XC2000

This section describes the implementation of the USIC channels in the XC2000. It contains details about the:

- Implementation overview (see **Section 19.7.1**)
- Channel features (see **Section 19.7.2**)
- Address map (see **Section 19.7.3**)
- Interrupt control registers (see **Section 19.7.4**)
- Input and output signals of U0C0 (see **Section 19.7.5.1**)
- Input and output signals of U0C1 (see **Section 19.7.5.2**)
- Input and output signals of U1C0 (see **Section 19.7.5.3**)
- Input and output signals of U1C1 (see **Section 19.7.5.4**)
- Input and output signals of U2C0 (see **Section 19.7.5.5**)
- Input and output signals of U2C1 (see **Section 19.7.5.6**)

19.7.1 Implementation Overview

The XC2000 device contains three identical USIC modules (USIC0, USIC1 and USIC2) with 2 communication channels each. The address range between 400_H and $7FF_H$ of each USIC module is reserved.

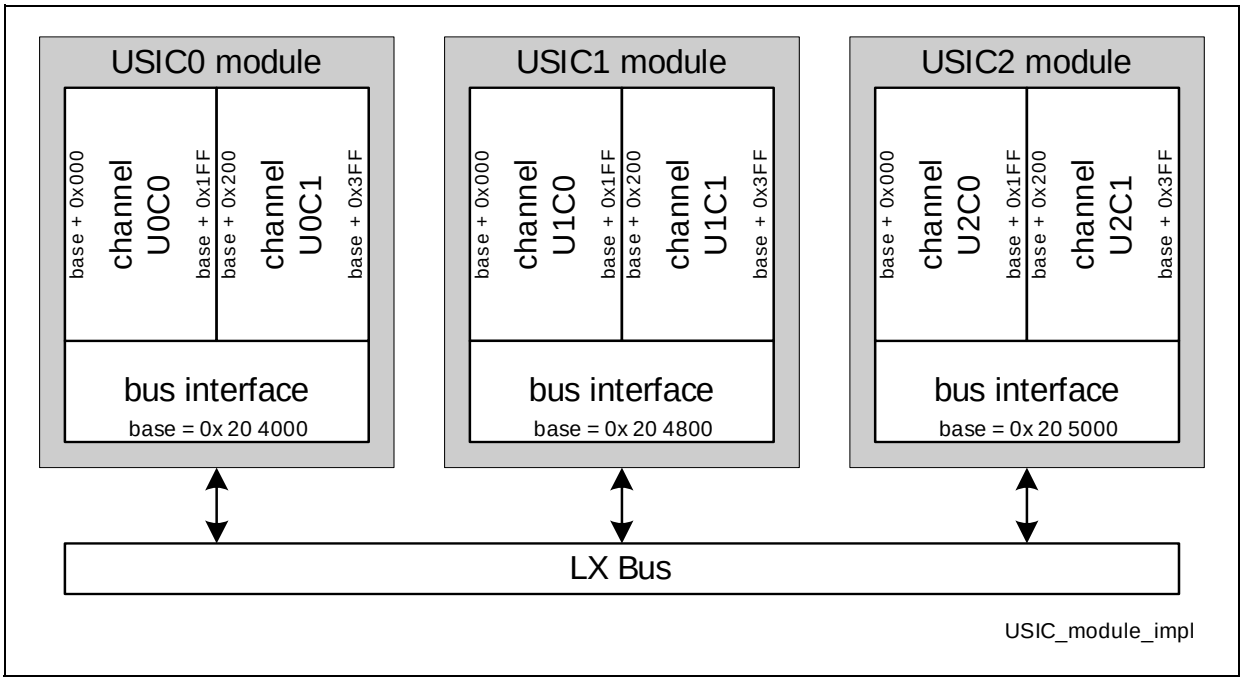


Figure 19-54 USIC Module Structure in XC2000

19.7.2 Channel Features

The USIC channels in the XC2000 support the following functionality:

Table 19-28 USIC Module Feature Set

Channel	ASC Protocol	LIN Support	SSC Protocol	IIC Protocol	IIS Protocol	FIFO Buffer Entries	SELOx[1)]
USIC Module 0							
U0C0	yes	yes	yes	yes	yes	64 shared	8
U0C1	yes	yes	yes	yes	yes		4
USIC Module 1							
U1C0	yes	yes	yes	yes	yes	64 shared	8
U1C1	yes	yes	yes	yes	yes		5
USIC Module 2							
U2C0	yes	yes	yes	yes	yes	64 shared	6
U2C1	yes	yes	yes	yes	yes		2

1) This number refers to the maximum number of signals available in the 144 pin package.

19.7.3 Address Map

The registers of the USIC communication channels are available at the following base addresses. The exact register address is given by the relative address of the register (given in **Table 19-3**) plus the channel base address (given in **Table 19-29**).

Table 19-29 Registers Address Space

Module	Base Address	End Address	Note
U0C0	$20\ 4000_H$	$20\ 41FF_H$	
U0C1	$20\ 4200_H$	$20\ 43FF_H$	
U1C0	$20\ 4800_H$	$20\ 49FF_H$	
U1C1	$20\ 4A00_H$	$20\ 4BFF_H$	
U2C0	$20\ 5000_H$	$20\ 51FF_H$	
U2C1	$20\ 5200_H$	$20\ 53FF_H$	

19.7.4 Interrupt Control Registers

Each USIC channel provides 4 service request outputs SR[3:0] (not all of them are necessarily connected to independent interrupt registers UxCy_nIC). The table below shows the assignment of the interrupt control registers to the service request outputs.
Each USIC communication channel is connected to 3 dedicated interrupt control registers (connected to UxCy_SR[2:0], e.g. one for transmission, one for reception, the third one for protocol or error handling, or for the alternative receive events). A fourth interrupt control register per communication channel (connected to UxCy_SR3) is shared with module CC2.

The interrupt control registers are located in the SFR area. They are described in the general interrupt chapter.

Table 19-30 USIC Interrupt Control Registers

Short Name	Description
U0CO_0IC	Interrupt Control Register for SR0 of USIC0 channel 0
U0CO_1IC	Interrupt Control Register for SR1 of USIC0 channel 0
U0CO_2IC	Interrupt Control Register for SR2 of USIC0 channel 0
	Interrupt Control Register for SR3 of USIC0 channel 0, shared with CC2, see SCU_ISSR.4
U0C1_0IC	Interrupt Control Register for SR0 of USIC0 channel 1
U0C1_1IC	Interrupt Control Register for SR1 of USIC0 channel 1
U0C1_2IC	Interrupt Control Register for SR2 of USIC0 channel 1
	Interrupt Control Register for SR3 of USIC0 channel 1, shared with CC2, see SCU_ISSR.5
U1CO_0IC	Interrupt Control Register for SR0 of USIC1 channel 0
U1CO_1IC	Interrupt Control Register for SR1 of USIC1 channel 0
U1CO_2IC	Interrupt Control Register for SR2 of USIC1 channel 0
	Interrupt Control Register for SR3 of USIC1 channel 0, shared with CC2, see SCU_ISSR.6
U1C1_0IC	Interrupt Control Register for SR0 of USIC1 channel 1
U1C1_1IC	Interrupt Control Register for SR1 of USIC1 channel 1
U1C1_2IC	Interrupt Control Register for SR2 of USIC1 channel 1
	Interrupt Control Register for SR3 of USIC1 channel 1, shared with CC2, see SCU_ISSR.7
U2CO_0IC	Interrupt Control Register for SR0 of USIC2 channel 0
U2CO_1IC	Interrupt Control Register for SR1 of USIC2 channel 0

Table 19-30 USIC Interrupt Control Registers (cont'd)

Short Name	Description
U2CO_2IC	Interrupt Control Register for SR2 of USIC2 channel 0
	Interrupt Control Register for SR3 of USIC2 channel 1, shared with CC2, see SCU_ISSR.12
U2C1_0IC	Interrupt Control Register for SR0 of USIC2 channel 1
U2C1_1IC	Interrupt Control Register for SR1 of USIC2 channel 1
U2C1_2IC	Interrupt Control Register for SR2 of USIC2 channel 1
	Interrupt Control Register for SR3 of USIC2 channel 1, shared with CC2, see SCU_ISSR.13

19.7.5 Input/Output Connections

The following tables show the pin assignment of the USIC channels in the XC2000 device. Naming convention: UxCy refers to USIC module x channel y.

The following sections refer to the communication signals, whereas the connections of the service request outputs SR[3:0] to the interrupt control registers of each communication channel to the interrupt control registers is given in **Section 19.7.4**.

19.7.5.1 USIC Module 0 Channel 0

The signals of USIC module 0 channel 0 have the prefix U0C0_.

Table 19-31 USIC0 Channel 0 (U0C0) I/O Connections

Signal	from/to	I/O[1)]	Can be used to/as
Data Inputs			
DX0A	P10.0	I	shift data input
DX0B	P10.1	I	shift data input
DX0C	P10.6	I	shift data input
DX0D	P7.4	I	shift data input
DX0E	P2.3	I	shift data input
DX0F	P2.4	I	shift data input
DX0G	U0C0_ DOUT	I	loop back data shift input
Clock Inputs			
DX1A	P10.1	I	shift clock input
DX1B	P10.2	I	shift clock input
DX1C	P10.8	I	shift clock input
DX1D	0	I	shift clock input
DX1E	0	I	shift clock input
DX1F	0	I	shift clock input
DX1G	U0C0_ SCLKOUT	I	loop back shift clock input
Control Inputs			
DX2A	P10.3	I	shift control input
DX2B	P10.4	I	shift control input
DX2C	P10.10	I	shift control input

Table 19-31 USIC0 Channel 0 (U0C0) I/O Connections (cont'd)

Signal	from/to	I/O[1)]	Can be used to/as
DX2D	P2.6	I	shift control input
DX2E	CC2_8	I	input for transmit data validation
DX2F	CCU60_ T13_PM	I	input for transmit data validation
DX2G	U0C0_ SELO0	I	loop back shift control input
Data Outputs			
DOUT	P2.3	O	shift data output
DOUT	P7.3	O	shift data output
DOUT	P10.1	O	shift data output
DOUT	P10.6	O	shift data output
Clock Outputs			
MCLKOUT	P10.8	O	master clock output, e.g. for IIS
SCLKOUT	P2.5	O	shift clock output
SCLKOUT	P10.2	O	shift clock output
Control Outputs			
SELO0	P2.6	O	shift control output 0
SELO0	P10.10	O	shift control output 0
SELO1	P2.7	O	shift control output 1
SELO2	P2.11	O	shift control output 2
SELO3	P2.10	O	shift control output 3
SELO3	P10.4	O	shift control output 3
SELO4	P3.4	O	shift control output 4
SELO4	P10.9	O	shift control output 4
SELO4	P2.12	O	shift control output 4
SELO5	P3.5	O	shift control output 5
SELO6	P3.6	O	shift control output 6
SELO7	P3.7	O	shift control output 7
System Related Outputs			
DX0INS	ERU_0A2	O	external interrupt input for ERU (SCU)
DX0INS	U0C0_DX1F	O	single wire ASC collision detection

Table 19-31 USIC0 Channel 0 (U0C0) I/O Connections (cont'd)

Signal	from/to	I/O[1)]	Can be used to/as
DX1INS	-	O	
DX2INS	ERU_0A3	O	external interrupt input for ERU (SCU)
U0C0 Loop Back Outputs			
DOUT	U0C0_DX0G	O	loop back shift data output
SCLKOUT	U0C0_DX1G	O	loop back shift clock output
SELO0	U0C0_DX2G	O	loop back shift control output

1) Input/output for U0C0.

19.7.5.2 USIC Module 0 Channel 1

The signals of USIC module 0 channel 1 have the prefix U0C1_.

Table 19-32 USIC0 Channel 1 (U0C1) I/O Connections

Signal	from/to	I/O[1)]	Can be used to/as
Data Inputs			
DX0A	P10.0	I	shift data input
DX0B	P10.7	I	shift data input
DX0C	P10.14	I	shift data input
DX0D	0	I	shift data input
DX0E	P2.10	I	shift data input
DX0F	P7.3	I	shift data input
DX0G	U0C1_ DOUT	I	loop back data shift input
Clock Inputs			
DX1A	P10.10	I	shift clock input
DX1B	P10.5	I	shift clock input
DX1C	P10.15	I	shift clock input
DX1D	P2.8	I	shift clock input
DX1E	0	I	shift clock input
DX1F	0	I	shift clock input
DX1G	U0C1_ SCLKOUT	I	loop back shift clock input
Control Inputs			
DX2A	P10.3	I	shift control input
DX2B	P10.4	I	shift control input
DX2C	P2.7	I	shift control input
DX2D	0	I	shift control input
DX2E	RTC_ T14INT	I	input for transmit data validation
DX2F	CCU60_ T13_PM	I	input for transmit data validation
DX2G	U0C1_ SELO0	I	loop back shift control input

Table 19-32 USIC0 Channel 1 (U0C1) I/O Connections (cont'd)

Signal	from/to	I/O[1)]	Can be used to/as
Data Outputs			
DOUT	P2.9	O	shift data output
DOUT	P2.10	O	shift data output
DOUT	P7.3	O	shift data output
DOUT	P7.4	O	shift data output
DOUT	P10.0	O	shift data output
DOUT	P10.7	O	shift data output
DOUT	P10.14	O	shift data output
DOUT	P10.15	O	shift data output
Clock Outputs			
MCLKOUT	P10.9	O	master clock output, e.g. for IIS
SCLKOUT	P2.8	O	shift clock output
SCLKOUT	P7.4	O	shift clock output
SCLKOUT	P10.5	O	shift clock output
Control Outputs			
SELO0	P2.7	O	shift control output 0
SELO0	P10.8	O	shift control output 0
SELO1	P2.6	O	shift control output 1
SELO2	P2.11	O	shift control output 2
SELO3	P2.12	O	shift control output 3
SELO4	-	O	shift control output 4
SELO5	-	O	shift control output 5
SELO6	-	O	shift control output 6
SELO7	-	O	shift control output 7
System Related Outputs			
DX0INS	ERU_0B2	O	external interrupt input for ERU (SCU)
DX0INS	U0C1_DX1F	O	single wire ASC collision detection
DX1INS	-	O	
DX2INS	ERU_0B3	O	external interrupt input for ERU (SCU)
U0C1 Loop Back Outputs			

Table 19-32 USIC0 Channel 1 (U0C1) I/O Connections (cont'd)

Signal	from/to	I/O[1)]	Can be used to/as
DOUT	U0C1_DX0G	O	loop back shift data output
SCLKOUT	U0C1_DX1G	O	loop back shift clock output
SELO0	U0C1_DX2G	O	loop back shift control output

1) Input/output for U0C1.

19.7.5.3 USIC Module 1 Channel 0

The signals of USIC module 1 channel 0 have the prefix U1C0_.

Table 19-33 USIC1 Channel 0 (U1C0) I/O Connections

Signal	from/to	I/O[1)]	Can be used to/as
Data Inputs			
DX0A	P0.0	I	shift data input
DX0B	P0.1	I	shift data input
DX0C	P10.12	I	shift data input
DX0D	P10.13	I	shift data input
DX0E	ESR0	I	shift data input
DX0F	ESR1	I	shift data input
DX0G	U1C0_ DOUT	I	loop back data shift input
Clock Inputs			
DX1A	P0.1	I	shift clock input
DX1B	P0.2	I	shift clock input
DX1C	P0.5	I	shift clock input
DX1D	P10.11	I	shift clock input
DX1E	P10.12	I	shift clock input
DX1F	0	I	shift clock input
DX1G	U1C0_ SCLKOUT	I	loop back shift clock input
Control Inputs			
DX2A	P0.3	I	shift control input
DX2B	ESR0	I	shift control input
DX2C	ESR1	I	shift control input
DX2D	P10.6	I	shift control input
DX2E	CC2_9	I	input for transmit data validation
DX2F	CCU61_ T13_PM	I	input for transmit data validation
DX2G	U1C0_ SELO0	I	loop back shift control input
Data Outputs			

Table 19-33 USIC1 Channel 0 (U1C0) I/O Connections (cont'd)

Signal	from/to	I/O[1)]	Can be used to/as
DOUT	P0.0	O	shift data output
DOUT	P0.1	O	shift data output
DOUT	P10.12	O	shift data output
DOUT	P10.13	O	shift data output
DOUT	P10.15	O	shift data output
Clock Outputs			
MCLKOUT	P1.0	O	master clock output, e.g. for IIS
SCLKOUT	P0.2	O	shift clock output
SCLKOUT	P10.11	O	shift clock output
Control Outputs			
SELO0	P0.3	O	shift control output 0
SELO0	P10.6	O	shift control output 0
SELO1	P0.4	O	shift control output 1
SELO1	P10.14	O	shift control output 1
SELO2	P0.5	O	shift control output 2
SELO2	P10.15	O	shift control output 2
SELO3	P0.7	O	shift control output 3
SELO3	P10.13	O	shift control output 3
SELO4	P1.0	O	shift control output 4
SELO5	P1.1	O	shift control output 5
SELO6	P1.2	O	shift control output 6
SELO7	P1.3	O	shift control output 7
System Related Outputs			
DX0INS	ERU_1A2	O	external interrupt input for ERU (SCU)
DX0INS	U1C0_DX1F	O	single wire ASC collision detection
DX1INS	ERU_3B0	O	external interrupt input for ERU (SCU)
DX2INS	ERU_1A3	O	external interrupt input for ERU (SCU)
U1C0 Loop Back Outputs			
DOUT	U1C0_DX0G	O	loop back shift data output

Table 19-33 USIC1 Channel 0 (U1C0) I/O Connections (cont'd)

Signal	from/to	I/O[1)]	Can be used to/as
SCLKOUT	U1C0_DX1G	O	loop back shift clock output
SELO0	U1C0_DX2G	O	loop back shift control output

1) Input/output for U1C0.

19.7.5.4 USIC Module 1 Channel 1

The signals of USIC module 1 channel 1 have the prefix U1C1_.

Table 19-34 USIC1 Channel 1 (U1C1) I/O Connections

Signal	from/to	I/O[1)]	Can be used to/as
Data Inputs			
DX0A	P0.6	I	shift data input
DX0B	P0.7	I	shift data input
DX0C	ESR1	I	shift data input
DX0D	ESR2	I	shift data input
DX0E	P6.0	I	shift data input
DX0F	0	I	shift data input
DX0G	U1C1_ DOUT	I	loop back data shift input
Clock Inputs			
DX1A	P0.5	I	shift clock input
DX1B	P0.6	I	shift clock input
DX1C	P6.2	I	shift clock input
DX1D	0	I	shift clock input
DX1E	0	I	shift clock input
DX1F	0	I	shift clock input
DX1G	U1C1_ SCLKOUT	I	loop back shift clock input
Control Inputs			
DX2A	P0.4	I	shift control input
DX2B	ESR1	I	shift control input
DX2C	ESR2	I	shift control input
DX2D	P6.3	I	shift control input
DX2E	RTC_ T14INT	I	input for transmit data validation
DX2F	CCU61_ T13_PM	I	input for transmit data validation
DX2G	U1C1_ SELO0	I	loop back shift control input

Table 19-34 USIC1 Channel 1 (U1C1) I/O Connections (cont'd)

Signal	from/to	I/O[1)]	Can be used to/as
Data Outputs			
DOUT	P0.6	O	shift data output
DOUT	P0.7	O	shift data output
DOUT	P6.1	O	shift data output
Clock Outputs			
MCLKOUT	P1.7	O	master clock output, e.g. for IIS
SCLKOUT	P0.5	O	shift clock output
SCLKOUT	P6.2	O	shift clock output
Control Outputs			
SELO0	P0.4	O	shift control output 0
SELO0	P6.3	O	shift control output 0
SELO1	P0.3	O	shift control output 1
SELO2	P1.6	O	shift control output 2
SELO3	P1.5	O	shift control output 3
SELO4	P1.4	O	shift control output 4
SELO5	-	O	shift control output 5
SELO6	-	O	shift control output 6
SELO7	-	O	shift control output 7
System Related Outputs			
DX0INS	ERU_1B2	O	external interrupt input for ERU (SCU)
DX0INS	U1C1_DX1F	O	single wire ASC collision detection
DX1INS	-	O	
DX2INS	ERU_1B3	O	external interrupt input for ERU (SCU)
U1C1 Loop Back Outputs			
DOUT	U1C1_DX0G	O	loop back shift data output
SCLKOUT	U1C1_DX1G	O	loop back shift clock output
SELO0	U1C1_DX2G	O	loop back shift control output

1) Input/output for U1C1.

19.7.5.5 USIC Module 2 Channel 0

The signals of USIC module 2 channel 0 have the prefix U2C0_.

Table 19-35 USIC2 Channel 0 (U2C0) I/O Connections

Signal	from/to	I/O[1)]	Can be used to/as
Data Inputs			
DX0A	P3.0	I	shift data input
DX0B	P3.1	I	shift data input
DX0C	P1.5	I	shift data input
DX0D	P1.6	I	shift data input
DX0E	P9.5	I	shift data input
DX0F	0	I	shift data input
DX0G	U2C0_ DOUT	I	loop back data shift input
Clock Inputs			
DX1A	P3.0	I	shift clock input
DX1B	P3.2	I	shift clock input
DX1C	P1.7	I	shift clock input
DX1D	P9.7	I	shift clock input
DX1E	0	I	shift clock input
DX1F	0	I	shift clock input
DX1G	U2C0_ SCLKOUT	I	loop back shift clock input
Control Inputs			
DX2A	P3.3	I	shift control input
DX2B	P1.4	I	shift control input
DX2C	0	I	shift control input
DX2D	0	I	shift control input
DX2E	CC2_10	I	input for transmit data validation
DX2F	CCU62_ T13_PM	I	input for transmit data validation
DX2G	U2C0_ SELO0	I	loop back shift control input
Data Outputs			

Table 19-35 USIC2 Channel 0 (U2C0) I/O Connections (cont'd)

Signal	from/to	I/O[1)]	Can be used to/as
DOUT	P3.0	O	shift data output
DOUT	P3.1	O	shift data output
DOUT	P1.6	O	shift data output
DOUT	P9.4	O	shift data output
DOUT	P9.5	O	shift data output
Clock Outputs			
MCLKOUT	-	O	master clock output, e.g. for IIS
SCLKOUT	P3.2	O	shift clock output
SCLKOUT	P1.7	O	shift clock output
Control Outputs			
SELO0	P3.3	O	shift control output 0
SELO1	P3.4	O	shift control output 1
SELO2	P3.5	O	shift control output 2
SELO3	P3.7	O	shift control output 3
SELO4	P1.3	O	shift control output 4
SELO5	P1.4	O	shift control output 5
SELO6	-	O	shift control output 6
SELO7	-	O	shift control output 7
System Related Outputs			
DX0INS	ERU_2A2	O	external interrupt input for ERU (SCU)
DX0INS	U2C0_DX1F	O	single wire ASC collision detection
DX1INS	ERU_2B0	O	external interrupt input for ERU (SCU)
DX2INS	ERU_2A3	O	external interrupt input for ERU (SCU)
U2C0 Loop Back Outputs			
DOUT	U2C0_DX0G	O	loop back shift data output
SCLKOUT	U2C0_DX1G	O	loop back shift clock output
SELO0	U2C0_DX2G	O	loop back shift control output

1) Input/output for U2C0.

19.7.5.6 USIC Module 2 Channel 1

The signals of USIC module 2 channel 1 have the prefix U2C1_.

Table 19-36 USIC2 Channel 1 (U2C1) I/O Connections

Signal	from/to	I/O[1)]	Can be used to/as
Data Inputs			
DX0A	P3.6	I	shift data input
DX0B	P3.7	I	shift data input
DX0C	P1.1	I	shift data input
DX0D	P1.2	I	shift data input
DX0E	ESR2	I	shift data input
DX0F	0	I	shift data input
DX0G	U2C1_ DOUT	I	loop back data shift input
Clock Inputs			
DX1A	P3.5	I	shift clock input
DX1B	P3.6	I	shift clock input
DX1C	P1.2	I	shift clock input
DX1D	0	I	shift clock input
DX1E	0	I	shift clock input
DX1F	0	I	shift clock input
DX1G	U2C1_ SCLKOUT	I	loop back shift clock input
Control Inputs			
DX2A	P3.4	I	shift control input
DX2B	ESR2	I	shift control input
DX2C	ESR1	I	shift control input
DX2D	0	I	shift control input
DX2E	RTC_ T14INT	I	input for transmit data validation
DX2F	CCU62_ T13_PM	I	input for transmit data validation
DX2G	U2C1_ SELO0	I	loop back shift control input

Table 19-36 USIC2 Channel 1 (U2C1) I/O Connections (cont'd)

Signal	from/to	I/O[1)]	Can be used to/as
Data Outputs			
DOUT	P3.6	O	shift data output
DOUT	P3.7	O	shift data output
DOUT	P1.1	O	shift data output
Clock Outputs			
MCLKOUT	-	O	master clock output, e.g. for IIS
SCLKOUT	P3.5	O	shift clock output
SCLKOUT	P1.2	O	shift clock output
Control Outputs			
SELO0	P3.4	O	shift control output 0
SELO1	P3.3	O	shift control output 1
SELO2	-	O	shift control output 2
SELO3	-	O	shift control output 3
SELO4	-	O	shift control output 4
SELO5	-	O	shift control output 5
SELO6	-	O	shift control output 6
SELO7	-	O	shift control output 7
System Related Outputs			
DX0INS	ERU_2B2	O	external interrupt input for ERU (SCU)
DX0INS	U2C1_DX1F	O	single wire ASC collision detection
DX1INS	-	O	
DX2INS	ERU_2B3	O	external interrupt input for ERU (SCU)
U2C1 Loop Back Outputs			
DOUT	U2C1_DX0G	O	loop back shift data output
SCLKOUT	U2C1_DX1G	O	loop back shift clock output
SELO0	U2C1_DX2G	O	loop back shift control output

1) Input/output for U2C1.

20 Controller Area Network (MultiCAN) Controller

This chapter describes the MultiCAN controller of the XC2000. It contains the following sections:

- Overview of the MultiCAN Kernel (see **Section 20.1**)
- Functional description of the MultiCAN Kernel (see **Section 20.2**)
- XC2000 implementation specific details and registers of the MultiCAN controller (port connections and control, interrupt control, address decoding, clock control, see **Section 20.4**).

Note: The MultiCAN kernel register names described in this chapter will be referenced in the XC2000 User's Manual by the module name prefix "CAN_".

20.1 MultiCAN Short Description

This section describes the serial communication interfaces CAN (Controller Area Network) of the communication module MultiCAN of the XC2000.

20.1.1 Overview

The MultiCAN module contains 5 independent CAN nodes, representing the communication interfaces.

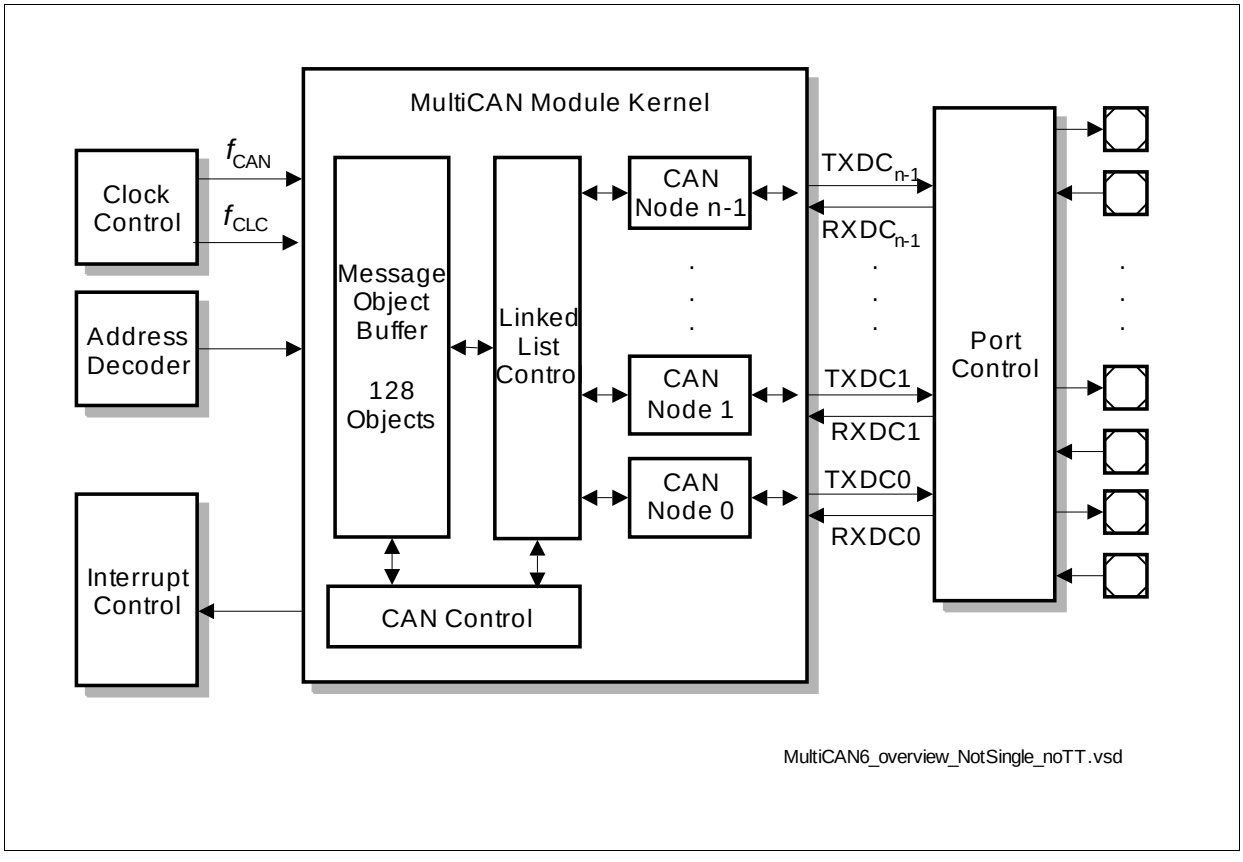


Figure 20-1 Overview of the MultiCAN Module

20.1.2 CAN Features

Several key features contribute to the high performance of the MultiCAN module:

- CAN functionality conforms to CAN specification V2.0 B active for each CAN node (compliant to ISO 11898)
- 5 independent CAN nodes available
- 128 independent message objects (shared by the CAN nodes)
- Dedicated control registers for each CAN node
- Data transfer rate up to 1MBaud, individually programmable for each node
- Flexible and powerful message transfer control and error handling capabilities
- Full-CAN functionality: message objects can be individually
  - Assigned to one of the 5 CAN nodes
  - Configured as transmit or receive object
  - Participate in a message buffer with FIFO algorithm
  - Set up to handle frames with 11-bit or 29-bit identifiers
  - Provided with programmable acceptance mask register for filtering
  - Monitored via a frame counter
  - Configured to Remote Monitoring Mode
- Automatic gateway mode support

- 16 individually programmable interrupt outputs
- CAN Analyzer Mode for bus monitoring
- SRAMs in MultiCAN module optionally parity error protected

20.2 CAN Functional Description

This section describes the core features of the CAN module.

20.2.1 Conventions and Definitions

Table 20-1 defines constants that are used throughout the MultiCAN specification. These are fixed values for a given MultiCAN implementation.

Table 20-1 Fixed Module Constants

Constant	Value	Description
n_objects	128	**Number of Message Objects** n_objects denotes the total amount of message objects available.
n_interrupts	16	**Number of Interrupt Output Lines** n_interrupts denotes the total number of interrupt outputs available.
n_pendings	256	**Number of Message Pending Bits** n_pendings denotes the number of message pending bits available. The number of message pending registers is given by n_pendings/32.
n_lists	8	**Number of Lists** n_lists denotes the total number of lists available for allocation of message number.
n_nodes	5	**Number of CAN Nodes Available** n_nodes denotes the total number of CAN nodes available. As each CAN node has it's own list in addition to the list of un-allocated elements, the relation n_nodes < n_lists is true.

20.2.2 Introduction

The MultiCAN module contains 5 Full-CAN nodes operating independently or exchanging data and remote frames via a gateway function.Transmission and reception of CAN frames is handled in accordance to CAN specification V2.0part B (active). Each CAN node can receive and transmit standard frames with 11-bit identifiers as well as extended frames with 29-bit identifiers.

20.2.2.1 Feature Overview

All CAN nodes share a common set of message objects, where each message object may be individually allocated to one of the CAN nodes. Besides serving as a storage

container for incoming and outgoing frames, message objects may be combined to build gateways between the CAN nodes or to setup a FIFO buffer.

The message objects are organized in double chained lists, where each CAN node has it's own list of message objects. A CAN node stores frames only into message objects that are allocated to the list of the CAN node. It only transmits messages from objects of this list.

A powerful, command driven list controller performs all list operations.

The bit timings for the CAN nodes are derived from the peripheral clock (f_{CAN}) and are programmable up to a data rate of 1 MBaud. A pair of receive and transmit pins connects each CAN node to a bus transceiver.

Features

- Compliant to ISO 11898.
- CAN functionality according to CAN specification V2.0 B active.
- Dedicated control registers are provided for each CAN node.
- A data transfer rate up to 1 MBaud is supported.
- Flexible and powerful message transfer control and error handling capabilities are implemented.
- Advanced CAN bus bit timing analysis and baud rate detection can be performed for each CAN node via the frame counter.
- Full-CAN functionality: A set of 128 message objects can be individually
  - allocated (assigned) to any CAN node
  - configured as transmit or receive object
  - setup to handle frames with 11-bit or 29-bit identifier
  - counted or assigned a timestamp via a frame counter
  - configured to remote monitoring mode
- Advanced Acceptance Filtering:
  - Each message object provides an individual acceptance mask to filter incoming frames.
  - A message object can be configured to accept only standard or only extended frames or to accept both standard and extended frames.
  - Message objects can be grouped into 4 priority classes.
  - The selection of the message to be transmitted first can be performed on the basis of frame identifier, IDE bit and RTR bit according to CAN arbitration rules.
- Advanced Message Object Functionality:
  - Message Objects can be combined to build FIFO message buffers of arbitrary size, which is only limited by the total number of message objects.
  - Message objects can be linked to form a gateway to automatically transfer frames between 2 different CAN buses. A single gateway can link any two CAN nodes. An arbitrary number of gateways may be defined.
- Advanced Data Management:

  - The Message objects are organized in double chained lists.
  - List reorganizations may be performed any time, even during full operation of the CAN nodes.
  - A powerful, command driven list controller manages the organization of the list structure and ensures consistency of the list.
  - Message FIFOs are based on the list structure and can easily be scaled in size during CAN operation.
  - Static Allocation Commands offer compatibility with TwinCAN applications, which are not list based.
- Advanced Interrupt Handling:
  - Up to 16 interrupt output lines are available. Most interrupt requests can be individually routed to one of the 16 interrupt output lines.
  - Message postprocessing notifications can be flexibly aggregated into a dedicated register field of 256 notification bits.

20.2.2.2 Module Structure

Figure 20-2 shows the general structure of the MultiCAN module with 5 CAN nodes.

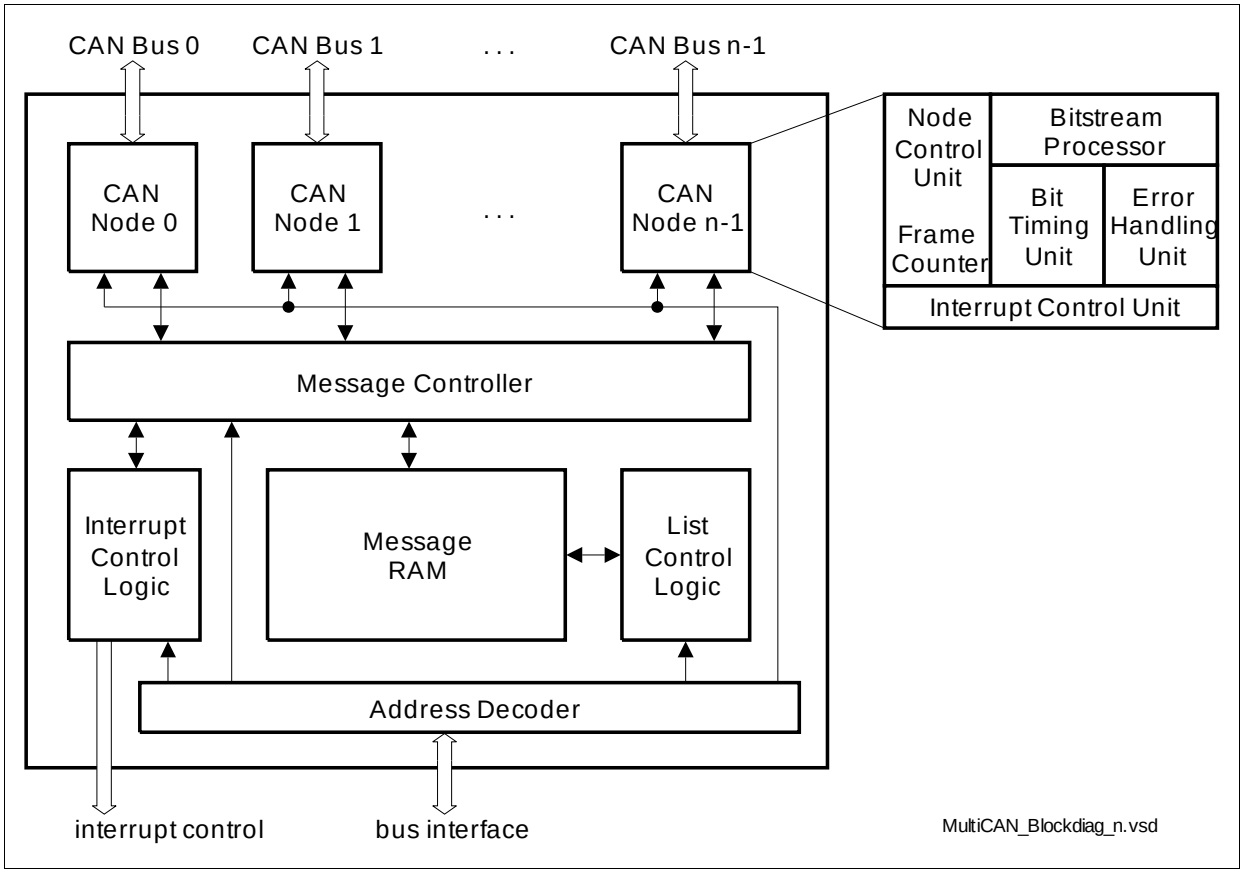


Figure 20-2 MultiCAN Block Diagram with several CAN Nodes

CAN Nodes

Each CAN node consists of several sub-units as described in **Table 20-2**.:

Table 20-2 Subunits of CAN Nodes

Subunit	Description
Bit Stream Processor	The Bit Stream Processor performs data, remote, error and overload frame processing according to the ISO 11898 standard. This includes conversion between the serial data stream and the input/output shift registers.
Bit Timing Unit	The Bit Timing Unit defines the length of a bit time and the location of the sample point according to the user settings, taking into account propagation delays and phase shift errors. The Bit Timing Unit also performs resynchronization.
Error Handling Unit	The Error Handling Unit manages the receive and transmit error counter. According to the contents of both counters the CAN node is set into an “Error Active”, “Error Passive” or “Bus-Off” state.
Node Control Unit	The Node Control Unit coordinates the operation of the CAN node: • Enables/disable CAN transfer of the node • Enable/Disable and generate node specific events that lead to an interrupt request (CAN bus errors, successful frame transfers etc.) • Administration of the Frame Counter

Message Controller

The message controller handles the exchange of CAN frames between the CAN nodes and the message objects which are stored in the Message RAM. It performs:

- Receive Acceptance filtering to determine the correct message object for storing of a received CAN frame.
- Transmit Acceptance Filtering to determine the message object to be transmitted first, individually for each CAN node.
- Content transfer between message objects and the CAN nodes, taking into account the status/control bits of the message objects.
- Handling of the FIFO buffering and Gateway functionality.
- Aggregation of message pending notification bits.

List Controller

The list controller performs all operations that lead to a modification of the double chained message object lists. Only the list controller is allowed to modify the list structure. The allocation/deallocation or reallocation of a message object can be requested via a user command interface (command panel). The list controller state machine then performs the requested command autonomously.

20.2.3 CAN Node Control

Each CAN node may be configured and run independently from the other CAN nodes. To this end each CAN node is equipped with an individual set of SFR registers to control and to monitor the CAN node.

20.2.3.1 Bit Timing

According to ISO 11898 standard, a CAN bit time is subdivided into different segments (**Figure 20-3**). Each segment consists of multiples of a time quantum t_q. The magnitude of t_q is adjusted by the bit field BRP and by bit DIV8, both controlling the baud rate prescaler (see bit timing register NBTR). The baud rate prescaler is driven by the MultiCAN module clock f_{CAN}.

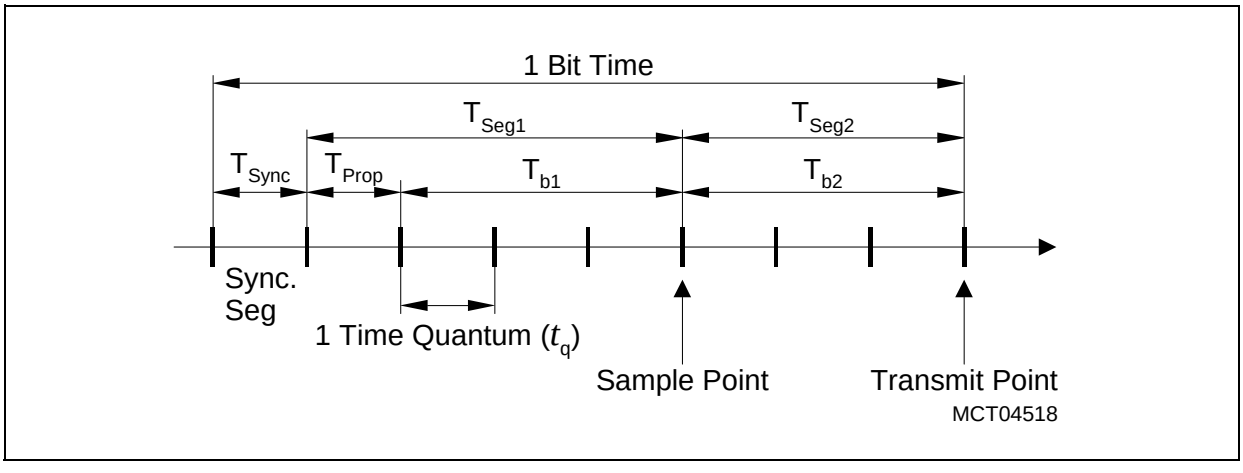


Figure 20-3 CAN Bus Bit Timing Standard

The Synchronization Segment (T_{Sync}) allows a phase synchronization between transmitter and receiver time base. The Synchronization Segment length is always 1 t_q. The Propagation Time Segment (T_{Prop}) takes into account the physical propagation delay in the transmitter output driver, on the CAN bus line and in the transceiver circuit. For a working collision detect mechanism, T_{Prop} has to be two times the sum of all propagation delay quantities rounded up to a multiple of t_q. The Phase Buffer Segments 1 and 2 (T_{b1}, T_{b2}) before and after the signal sample point are used to compensate a mismatch between transmitter and receiver clock phase detected in the synchronization segment.

The maximum number of time quanta allowed for resynchronization is defined by bit field SJW in the CAN Node Bit Timing register NBTR. The Propagation Time Segment and the Phase Buffer Segment 1 are combined to parameter TSeg1, which is defined by the value TSEG1 in the respective CAN Node Bit Timing register NBTR. A minimum of 3 time quanta is requested by the ISO standard. Parameter TSeg2, which is defined by the value of TSEG2 in the CAN Node Bit Timing Register NBTR, covers the Phase Buffer Segment 2. A minimum of 2 time quanta is requested by the ISO standard. According

ISO standard, a CAN bit time, calculated as the sum of $T_{Sync,}$ T_{Seg1} and $T_{Seg2,}$ must not fall below 8 time quanta.

Calculation of the bit time:

t_q	= (BRP+1) / f_{CAN}	if DIV8 = 0
	= 8 × (BRP+1) / f_{CAN}	if DIV8 = 1
T_{Sync}	= 1 t_q	
T_{Seg1}	= (TSEG1 + 1) × t_q	(min. 3 t_q)
T_{Seg2}	= (TSEG2 + 1) × t_q	(min. 2 t_q)
bit time	= T_{Sync} + T_{Seg1} + T_{Seg2}	(min. 8 t_q)

To compensate phase shifts between clocks of different CAN controllers, the CAN controller has to synchronize on any edge from the recessive to the dominant bus level. If the hard synchronization is enabled (at the start of frame), the bit time is restarted at the synchronization segment. Otherwise, the resynchronization jump width T_{SJW} defines the maximum number of time quanta a bit time may be shortened or lengthened by one resynchronization. The value of SJW is programmed in the CAN Node Bit Timing Register.

T_{SJW}	= (SJW + 1) × t_q
T_{Seg1}	≥ T_{SJW} + T_{prop}
T_{Seg2}	≥ T_{SJW}

The maximum relative tolerance for f_{CAN} depends on the Phase Buffer Segments and the resynchronization jump width.

dfCAN	≤ min (Tb1, Tb2) / 2 × (13 × bit time - Tb2)	AND
dfCAN	≤ TSJW / 20 × bit time	

A valid CAN bit timing must be written to the CAN Node Bit Timing Register NBTR before resetting the INIT bit in the Node Control Register, i.e. before enabling the operation of the CAN node.

The Node Bit Timing Register may be written only if bit CCE (Configuration Change Enable) is set in the corresponding Node Control Register.

20.2.3.2 CAN Error Handling

The Error Handling Unit of the CAN node is responsible for the fault confinement of the CAN device. Its two counters, the Receive Error Counter and the Transmit Error Counter (control register NECNT), are incremented and decremented by commands from the Bit Stream Processor. If the Bit Stream Processor itself detects an error while a transmit operation is running, the Transmit Error Counter is incremented by 8. An increment of 1 is used, when the error condition was reported by an external CAN node via an error frame generation. For error analysis, the transfer direction of the disturbed message and the node, recognizing the transfer error, are indicated in the control register NECNT of the respective CAN node. According to the values of the error counters, the CAN node is set into the states "error active", "error passive" and "bus-off".

The CAN node is in error active state, if both error counters are below the error passive limit of 128. It is in error passive state, if at least one of the error counters equals or exceeds 128.

The bus-off state is activated if the Transmit Error Counter equals or exceeds the bus-off limit of 256. This state is reported by flag BOFF in the NSR status register of the CAN node. The device remains in this state, until the bus-off recovery sequence is finished. Additionally, there is the bit EWRN in the NSR status register, which is set, if at least one of the error counters equals or exceeds the error warning limit defined by bit field EWRNLVL in the control registers NECNT of the CAN node. Bit EWRN is reset if both error counters fall below the error warning limit again (see **Page 20-63**).

20.2.3.3 CAN Frame Counter

Each CAN node is equipped with a frame counter which allows to count transmitted/received CAN frames or to obtain information about the time instant when a frame has been started to transmit or being received by the CAN node. CAN frame counting/bit time counting is performed by a 16 bit counter which is controlled by register NFCR of the respective CAN node. Bit field CFSEL of register NFCR defines the operation mode of the frame counter:

- Frame Count Mode: The frame counter is incremented after the successful transmission and/or reception of a CAN frame. The incremented value is copied to the CFC field of the Interrupt Pointer Register of the message object involved in the transfer.
- Time Stamp Mode: The frame counter is incremented with the beginning of a new bit time. When the transmission/reception of a frame starts, the value of the frame counter is captured and stored to the CFC field of register NFCR. After the successful transfer of the frame the captured value is copied to the CFC field of the Interrupt Pointer Register of the message object involved in the transfer.
- Bit Timing Mode: Used for baud rate detection and analysis of the bit timing (**Chapter 20.2.5.3**).

20.2.3.4 CAN Node Interrupts

Each CAN node is equipped with four interrupt sources.

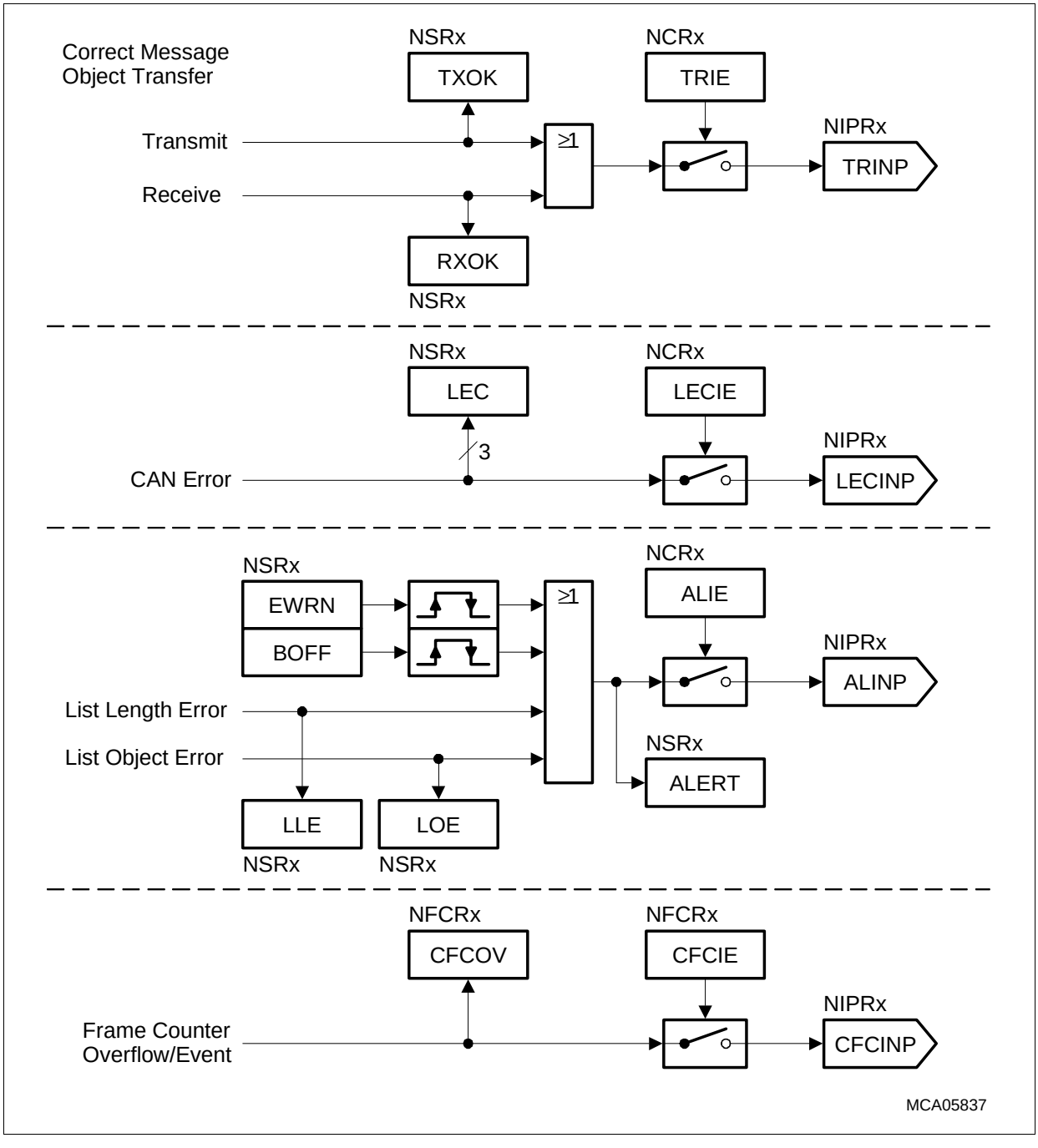


Figure 20-4 CAN Node Interrupts

An interrupt request is generated upon:

- The successful transmission/reception of a frame,
- An overflow of the frame counter (frame count mode/time stamp mode) or a bit timing measurement event (bit timing mode),
- An error related to the CAN node.

20.2.4 Message Object List Structure

The message objects of the MultiCAN module are organized in double chained lists, where each message object has a pointer to the previous message object in the list as well as a pointer to the next message object in the list.

20.2.4.1 Basics

The MultiCAN module provides 16 different lists, where each object is allocated to one of these lists. A 4 bit LIST bit field in the Message Object Control Register indicates the list to which the respective message object is currently allocated. In the example of **Figure 20-5** three message objects are allocated to the list with list index 2.

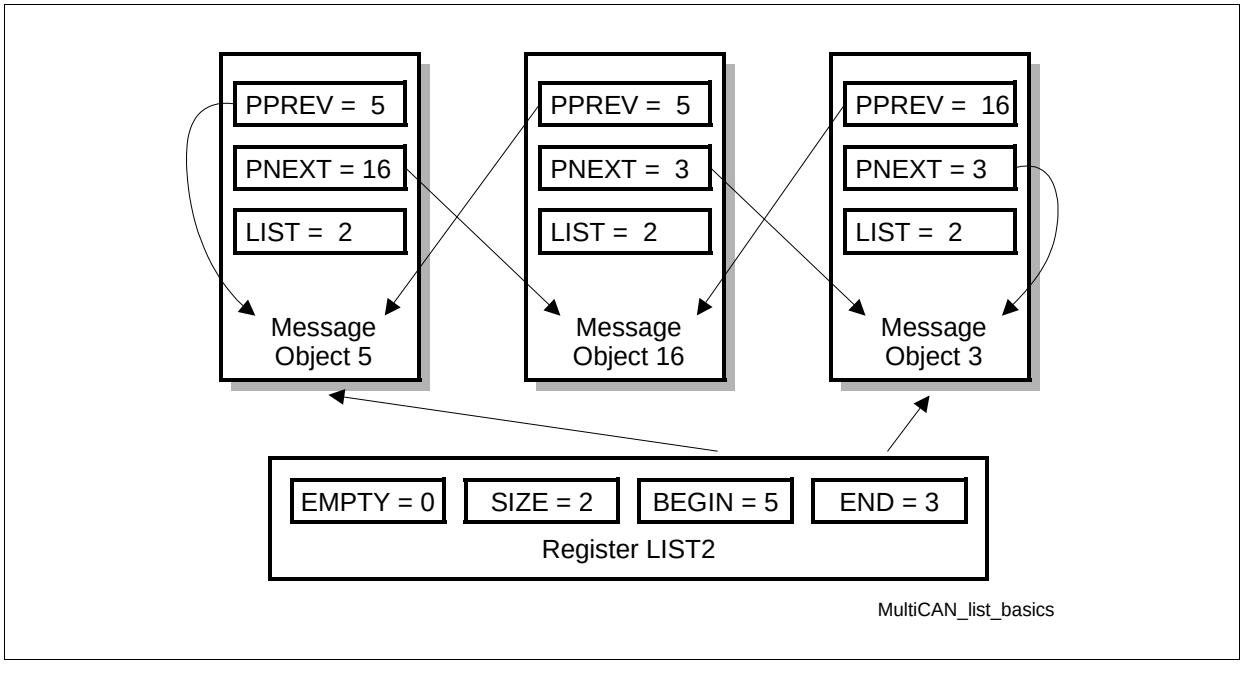


Figure 20-5 Example Allocation of Message Objects to a List

The BEGIN field of the List Register points to the first element in the list (object 5 in the example) whereas the END field points to the last element in the list (object 3 in the example). The number of elements in the list is indicated in the SIZE field of the List Register (#elements = SIZE + 1, thus SIZE = 2 for the 3 elements of the example). The

EMPTY bit indicates a list with no elements (EMPTY = 0 in the example, as the list is not empty).

Each message object has a pointer PNEXT (located in the Message Object Control Register) that points to the next message object in the list and a pointer PREV that points to the previous message object in the list. PPREV of the first message object points to the object itself because the first object has no predecessor (in the example object 5 is the first object, indicated by PPREV = 5). PNEXT of the last message object also points to the object itself because the last element has no successor (in the example object 3 is the last object, indicated by PNEXT = 3).

Each message object also has a 4 bit LIST field (located in the Message Object Control Register) which shows list index of the list to which the object is currently allocated (the objects of the example are allocated to list 2, thus LIST = 2).

20.2.4.2 List of Unallocated Elements

The list with list index 0 has a special meaning: It is the list of all unallocated elements. An element is called unallocated if and only if it belongs to list zero. It is called allocated if and only if it belongs to one of the other lists.

After reset all message objects are unallocated, i.e. belong to the list of unallocated elements. The initial allocation of the message objects within the list of unallocated objects is ordered by message number, i.e. the predecessor of message object n is object n-1 and the successor of object n is object n+1.

20.2.4.3 Connection to the CAN Nodes

One CAN node is linked to exactly one unique list of message objects..

Table 20-3 List Indices

List Index	Description
0	List of unallocated elements
1 to n_nodes	Lists associated to a CAN node. List index i belongs to CAN node i -1.
n_nodes+1 to n_lists-1	Free user lists, which are not associated to a CAN node.

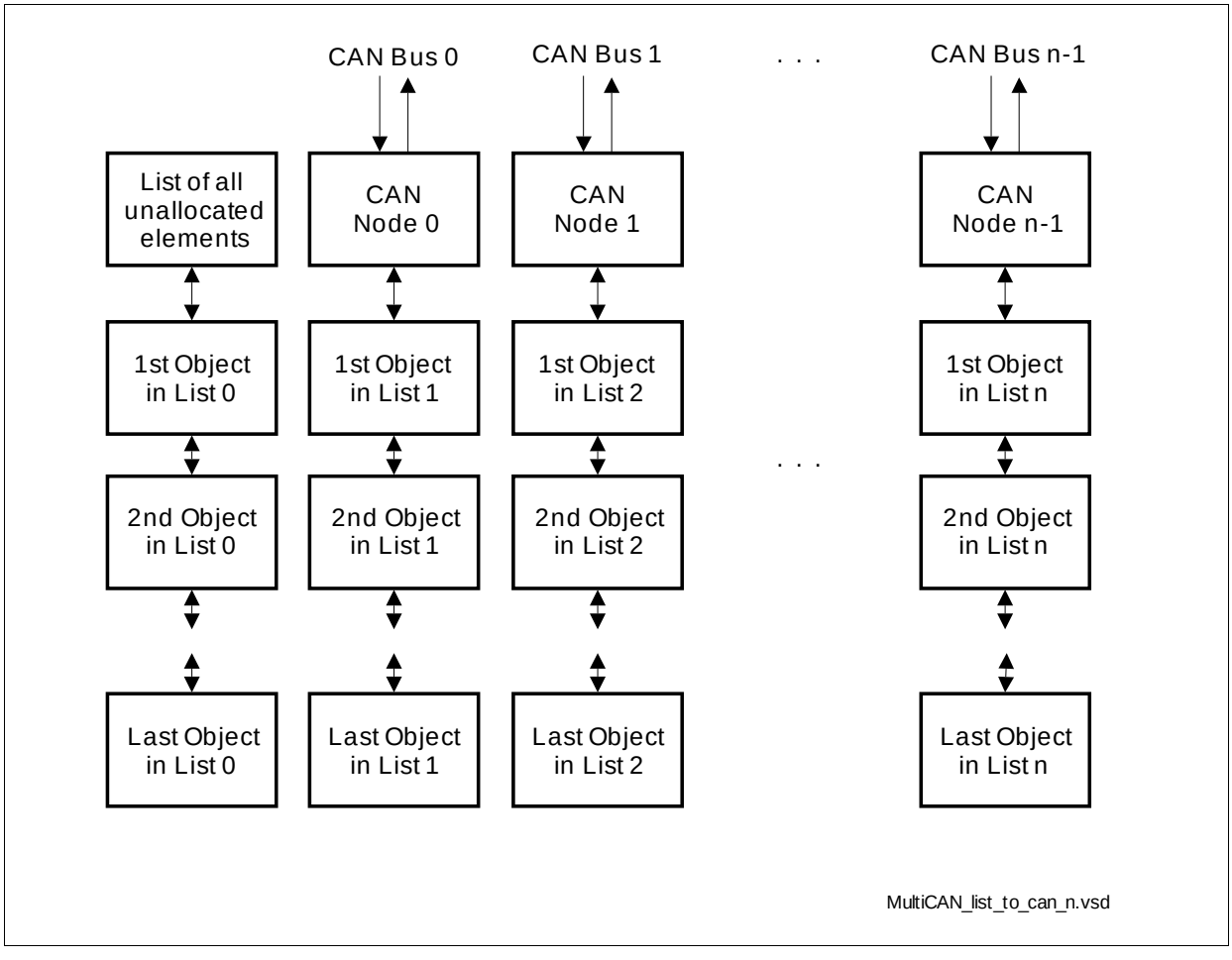


Figure 20-6 Message Objects Linked to several CAN Nodes

20.2.4.4 List Command Panel

The list structure may not be modified directly by means of write accesses to the LIST registers and the PPREV, PNEXT and LIST fields in the message objects as they are read only. The management of the list structure is performed by and limited to the list controller unit inside the MultiCAN module. The list controller is controlled via a command panel which allows the user to issue list allocation commands to the list controller. The list controller basically serves two purposes:

1. Ensure that all operations that modify the list structure result in a consistent list structure.
2. Present maximum comfort and flexibility to the user.

The list controller and the associated command panel allows the programmer to concentrate on the final properties of the list, which are characterized by the allocation of message objects to a CAN node and the ordering relation between objects which are allocated to the same list. The process of list (re-)building is left to the list controller.

A panel command is started by writing the respective command code (see **Table 20-9 “Panel Commands” on Page 20-53**) into to the PANCMD field of the panel control register. The corresponding command arguments must be written to PANAR1 and PANAR2 before writing the command code or latest together with the command code in a single 32 bit write access to the panel control register (only possible within 32 bit system environments).

With the write of a valid command code the BUSY flag in the Panel Control Register becomes active (BUSY = 1) and the control panel registers are locked, which means that write accesses to the Panel Control Register are ignored. The BUSY flag remains active and the control panel remains locked until the execution of the requested command is completed.

When the issued command is a dynamic allocation which takes an element from the list of unallocated objects, then also the RBUSY bit becomes active together with the BUSY bit (RBUSY = BUSY = 1) to indicate that PANAR1 and PANAR2 are going to be updated by the list controller:

1. The message number of the message object taken from the list of unallocated elements is written to PANAR1.
2. An error status is posted to bit 7 of PANAR2 (Bit 7 = ERR). If ERR = 1 then the list of unallocated elements was empty and the command is aborted. If ERR = 0 then the list was not empty and the command will be performed successfully.

The results are written before the list controller starts the actual allocation process. As soon as the results are available, RBUSY becomes inactive (RBUSY = 0) again, while BUSY still remains active until completion of the command. This allows the user to setup the new message object while it is still in the process of list allocation. The access to message objects is not limited during ongoing list operations. However, any access to a

register resource located inside the RAM delays the ongoing allocation process by one access cycle.

As soon as the command is done the BUSY flag becomes inactive (BUSY = 0) and write accesses to the Panel Control Register are enabled again. Also the NOP command code is automatically written to the CMD field of the Panel Control Register. A new command may be started any time during BUSY inactive.

All fields of the Panel Control Register except BUSY and RBUSY may be written by the user. This allows to save and restore the Panel Control Register if the Command Panel shall be used within independent (mutually interruptible) interrupt routines. If this is the case then any task that uses the Command Panel and that may interrupt another task also using the Command Panel should poll the BUSY flag until is becomes inactive and save the whole PANCTR register to a save memory location before issuing a command. At the end it should restore PANCTR from the said memory location.

Before a message object which is allocated to the list of an active CAN node shall be moved to another list or to another position within the same list, bit MSGVAL (“Message Valid”) should be cleared in the Message Object Control Register of the message object.

20.2.5 CAN Node Analysis Features

CAN Analyze Mode allows to monitor the CAN traffic without affecting the logical state of the CAN bus.

20.2.5.1 Analyze Mode

CAN Analyze Mode is selected by setting bit CALM in the Node Control Register. CAN Analyze Mode may be selected for each CAN node individually.

In CAN Analyze Mode the transmit pin of the CAN node is held on recessive level. The CAN node may receive frames (data-, remote-, and error frames) but is not allowed to transmit. Active error frames are sent recessive. Received data/remote frames are not acknowledged (i.e. acknowledge slot is sent recessive), but will be received and stored in matching message objects as long as there is any other node that acknowledges the frame.

All message object functionality is available, but no transmit request will be executed.

20.2.5.2 Loop-back Mode

The MultiCAN module provides a loop-back mode to enable an in-system test of the MultiCAN module as well as the development of CAN driver software without access to an external CAN bus.

The loop-back feature consists of an internal CAN bus (inside the MultiCAN module) and a bus select switch for each CAN node (**Figure 20-7**). With the switch each CAN node can be wired either to the internal CAN bus (loop-back mode activated) or the external CAN bus, i.e. the transmit- and receive pins (normal operation). The CAN bus which is currently not selected is driven recessive, i.e. the transmit pin is held at 1 and the receive pin is ignored by the CAN nodes which are in loop-back mode.

Loop-back Mode is selected individually for each CAN node by setting bit LBM in the respective Node Port Control Register. All CAN nodes that are in loop-back mode may communicate on the internal CAN bus without affecting the normal operation of the other CAN nodes which are not in loop-back mode.

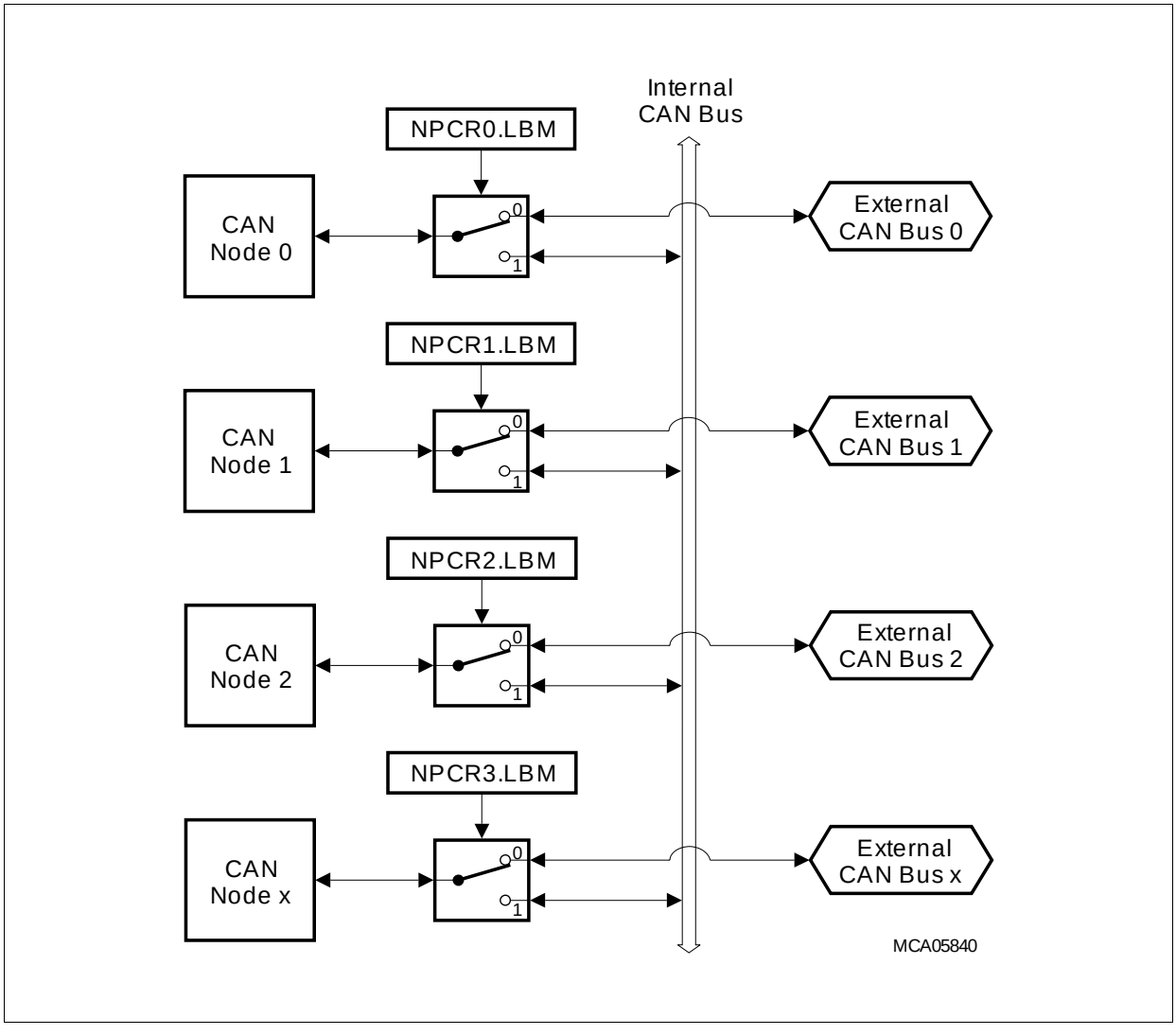


Figure 20-7 Loop-back Mode for several CAN Nodes

20.2.5.3 Bit Timing Analysis

For each CAN node detailed analysis of the bit timing can be performed by means of using dedicated analysis modes of the CAN frame counter. The bit timing analysis functionality of the frame counter may be used for automatic detection of the CAN baud rate as well as for the analysis of the timing of the CAN network.

Bit timing analysis for a CAN node is selected by CFMOD = 10_B (Bit Timing Mode) in the CAN Node Frame Counter Register.

Bit timing analysis does not affect the operation of the CAN node.

The measurement results are written to the CFC field. Whenever CFC is updated in Bit Timing Mode, then also the CFCOV bit is set in order to indicate the update event. If CFCIE is set then also an interrupt request is generated, where for the the CAN node i = 0 to 4 the interrupt request is generated on the interrupt output line i.

Automatic Baud Rate Detection

Automatic baud rate detection requires to measure the time between the observation of subsequent dominant edges on the CAN bus. This measurement is automatically performed if CFSE = 000_B in the CAN Node Frame Counter Register. With each dominant edge monitored on the CAN receive input the time (measured in clock cycles) between this edge and the most recent dominant edge is stored in the CFC field.

Synchronization Analysis

The bit time synchronization is monitored if CFSEL = 010_B. The time between the first dominant edge and the sample point is measured and stored in CFC. The bit timing synchronization offset may be derived from this time as the first edge after the sample point triggers synchronization and there is only one synchronization between consecutive sample points.

Synchronization Analysis may be used to fine tune the baud rate during reception of the first CAN frame with the measured baud rate.

Driver Delay Measurement

The delay between a transmitted edge and the corresponding received edge is measured with CFSEL = 011_B (dominant to dominant) and CFSEL = 001_B (recessive to recessive). These delays indicate the time needed to represent a new bit value on the physical implementation of the CAN bus.

20.2.6 Message Acceptance Filtering

The message acceptance filtering includes receive and transmit filtering.

20.2.6.1 Receive Acceptance Filtering

When a message object is received on a CAN node, then a unique message object is determined in which the received frame will be stored upon successful frame reception. A message object qualifies for the reception of a frame if and only if the following conditions are fulfilled:

1. The message object is allocated to the list of the CAN node on which the frame is received.
2. MSGVAL is set in the Message Control Register
3. RXEN is set in the Message Control Register
4. The DIR bit in the Message Control Register equals the RTR bit of the received frame. If DIR = 1 (transmit object) then the message object only accepts remote frames. If DIR = 0 (receive object) then the message object only accepts data frames.
5. If MIDE = 1 in the Acceptance Mask Register (MOAMR) then the IDE bit of the received frame equals the IDE bit in the Arbitration Register (MOAR). IF MOAR.IDE = 1 then the message object only accepts frames with extended identifier. If MOAR.IDE = 0 then the message object only accepts standard frames. If MOAMR.MIDE = 0 then the IDE bit of the received frame is don’t care, i.e. the message object accepts both standard and extended frames.
6. The identifier of the received frame matches the identifier stored in the Arbitration Register of the message object with respect to the acceptance mask in the MOAMR register. This means that each bit of the received identifier is equal to the corresponding identifier bit in the Acceptance Register, except those bits for which the corresponding mask bits in MOAMR are cleared. These identifier bits are don’t care. **Figure 20-8** illustrates this identifier check.

A priority ordering relation is defined for the message objects:

A message object A has higher receive priority than a message object B if and only if the following conditions are fulfilled:

1. A belongs to a higher priority class than B, i.e. MOAR.PRI of A must be less than or equal to MOAR.PRI of B.
2. If both objects belong to the same priority class (PRI of A = PRI of B) then message object B is a list successor of A, i.e. B can be reached by means of successively stepping forward in the list, starting from A.

Among all messages that fulfill all 6 qualifying criteria the unique message object with highest receive priority wins acceptance filtering, i.e. is selected for storage of the received frame. All other message objects loose receive acceptance filtering.

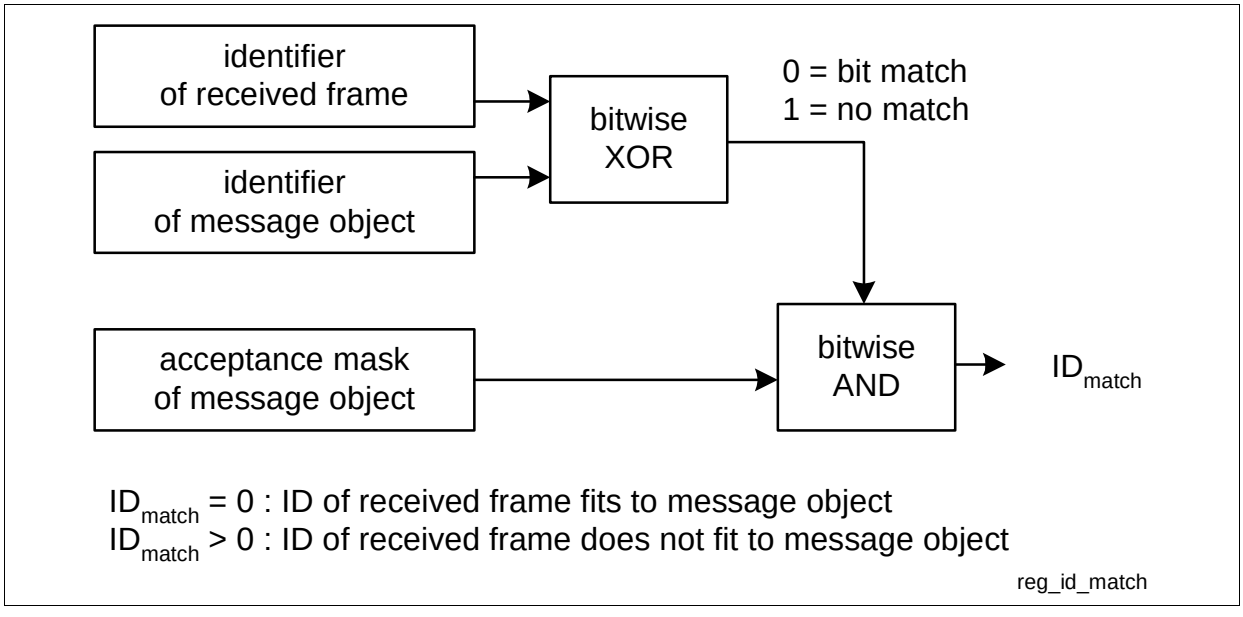


Figure 20-8 Received Message Identifier Acceptance Check

20.2.6.2 Transmit Acceptance Filtering

A message is requested for transmission by means of setting a transmit request in the message object which holds the message. If more than one message object has a valid transmit request for the same CAN node, then a single message object is chosen for actual transmission from the candidates, because only a single message object may be transmitted at the same time on a single CAN bus.

A message object qualifies for transmission on a given CAN node if and only if it meets the following criteria (**Figure 20-9**):

1. The message object is allocated to the list of the CAN node considered.
2. MSGVAL is set in the Message Object Control Register.
3. TXRQ is set in the Message Object Control Register.
4. TXEN0 and TXEN1 are set in the Message Object Control Register.

A priority order relation is defined for all qualifying objects to determine the message to be transmitted first: Let A and B be two message objects qualifying for transmission, where without loss of generality object B is assumed to be is a list successor of A, i.e. B can be reached by means of successively stepping forward in the list, starting from A. For both message objects associated CAN messages CAN_A and CAN_B are defined, where identifier, IDE and RTR bit are taken from MOAR.ID, MOAR.IDE and MOCTR.DIR.

If both message objects belong to a different priority class (different value of bit field PRI in the Message Object Arbitration Register MOAR) then the message object with lower PRI value has higher transmit priority and will be transmitted first.

If both message objects belong to the same priority class (equal value of bit field MOAR.PRI), then message object A has higher transmit priority than object B if and only if one of the following conditions is fulfilled:

1. PRI = 10 and CAN message CAN_A has higher or equal priority than CAN message CAN_B with respect to CAN arbitration rules (see **Table 20-13**).
2. PRI = 01 or PRI = 11 (priority by list order).
3. PRI = 00 is reserved. Transmit objects with PRI = 00 are not taken into account for the transmit acceptance filtering.

The unique message object that qualifies for transmission and has highest transmit priority wins transmit acceptance filtering, i.e. will be transmitted first. All other message objects lose the current transmit acceptance filtering round. They get a new chance in subsequent filtering rounds.

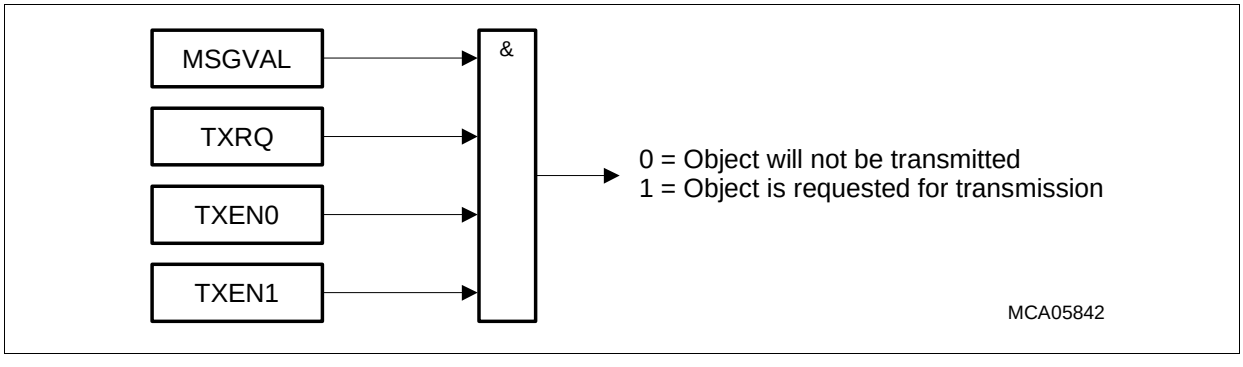


Figure 20-9 Effective Transmit Request of Message Object

20.2.7 Message Postprocessing Interface

When a message object has received or transmitted a frame successfully then the CPU may be notified to perform message postprocessing on the message object. The postprocessing interface of the MultiCAN module consists of two elements:

1. Message Interrupts to trigger postprocessing.
2. Message Pending Registers to aggregate the pending message interrupts into a common structure for postprocessing.

20.2.7.1 Message Interrupts

When the storage of a received frame into a message object or the successful transmission of a frame is completed then a message interrupt my be requested. For each message object both transmit and receive interrupts may be routed individually to one of the available interrupt output lines, as illustrated in **Table 20-10**. A receive interrupt is not restricted to the direct storage of a received frame from the CAN node the message object belongs to. It also occurs upon frame storage induced by FIFO or gateway action. The TXPND and RXPND bits are set whenever a successful transmission/reception takes place, no matter if the respective interrupt is enabled or not.

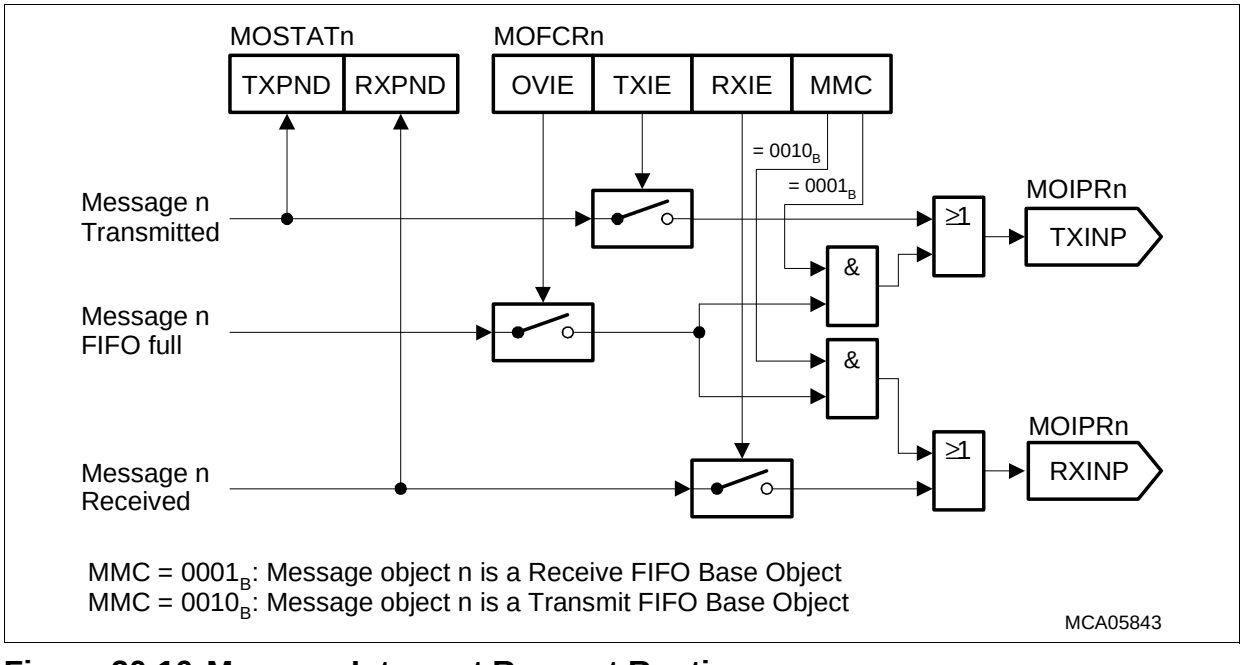


Figure 20-10 Message Interrupt Request Routing

20.2.7.2 Message Pendings

When a message interrupt request is generated then also a message pending bit is set in one of the Message Pending Register. To this end the pending bit selection field MPN is defined in the Message Object Interrupt Pointer Register. The value of MPN is combined with TXINP and RXINP to yield the effective bit position of the Pending bit, as illustrated in **Figure 20-11**. The bit position consists of 2 parts:

1. The high part (bits [7:5]) of the calculated position selects the Message Pending Register in which the pending bit will be set.
2. The low part (bits [4:0]) of the calculated position selects the position (0-31) of the pending bit within the 32 bit Message Pending Register.

The MPSEL bit field in the MultiCAN Control Register allows to include the interrupt request node pointer (RXINP for reception, TXINP for transmission) so as to implement different target location of the pending bit for receive and transmit.

The Message Pending Registers may be written by the CPU, but those bits that are written 1 are left unchanged and only those bits which are written 0 are cleared. This allows to clear individual bits with a single write access instead of a read/modify/write-back access. Thus there is no access conflict when the MultiCAN module sets another pending bit in the same register at the same time.

Each Message Pending Register is linked to an individual Message Index Register which displays the lowest bit position of all set (1) bits in the Message Pending Register. The Message Index Register is read only and is updated immediately when the value of the corresponding Message Pending Register changes.

There is no direct link between the Message Pending Registers and the interrupt request nodes. Such a link may, however, be established by the application. For example, each interrupt request node could be linked to a unique Message Pending Register. The example shown in **Figure 20-12** links message Pending Register n to interrupt output line n (n = 0-7).

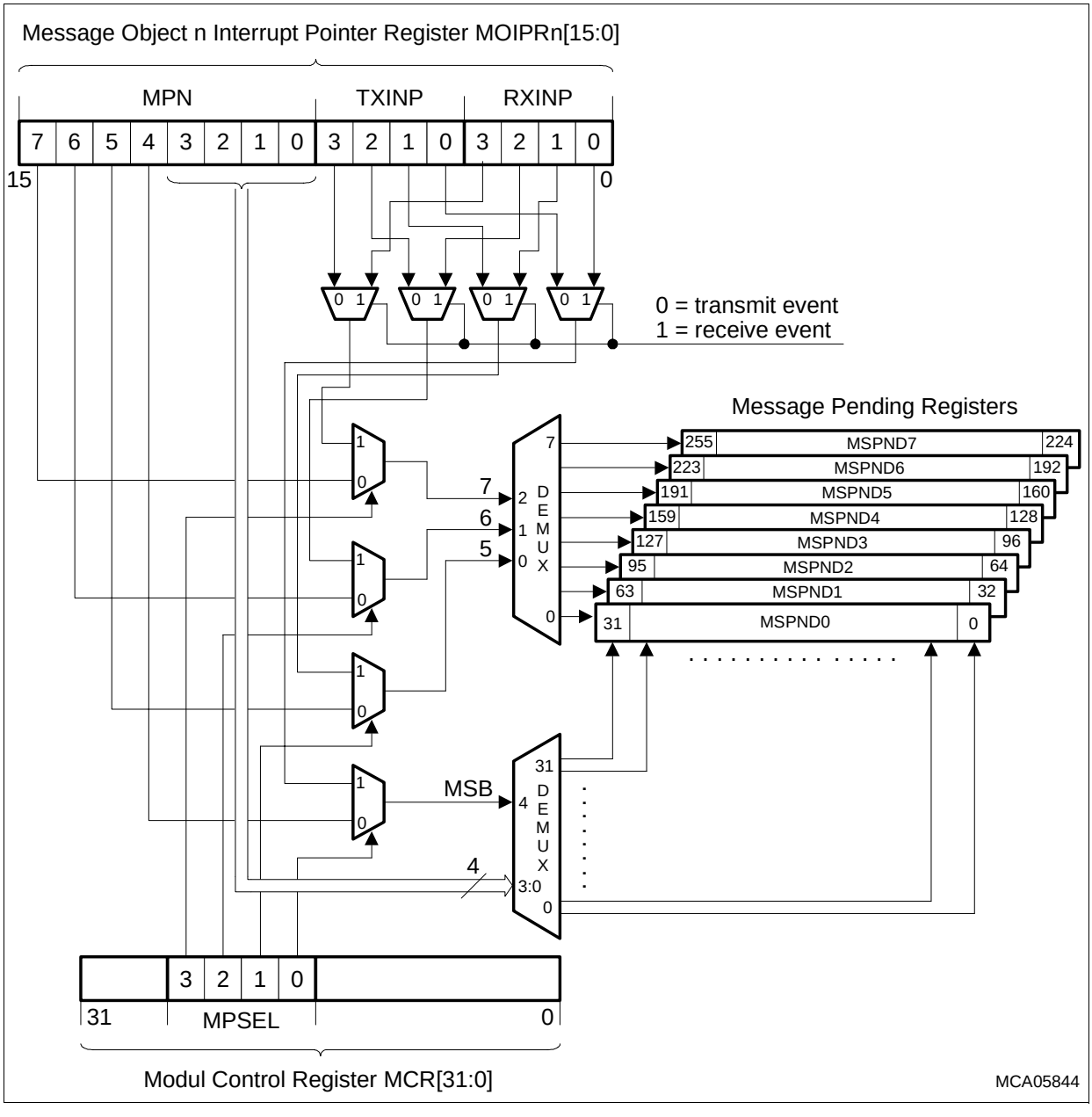


Figure 20-11 Target Location of the Message Pending Bit (Transmit/Receive)

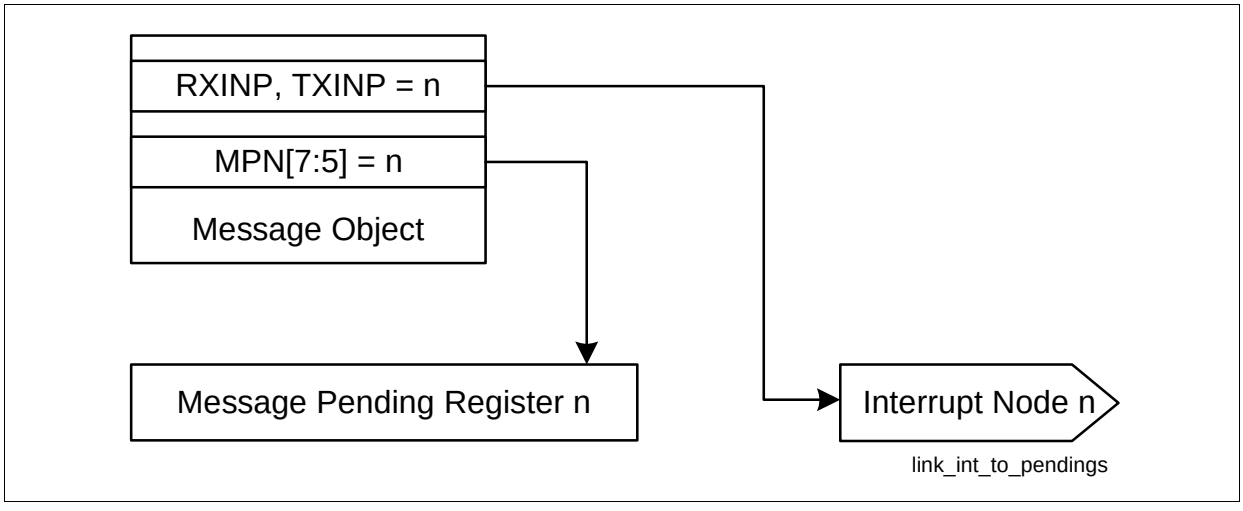


Figure 20-12 Example Link of Message Pending Registers to Interrupt Output Lines

20.2.8 Message Object Data Handling

The follwing section describes the actions taken during a frame reception and during a frame transmition.

20.2.8.1 Frame Reception

When a message is received on the CAN bus then the storage of the message into a message object is prepared and performed according to the scheme shown in **Figure 20-13**. The MultiCAN module not just copies the received data into the message object, but it provides advanced features to enable consistent data exchange between MultiCAN and CPU.

MSGVAL

The MSGVAL (“Message Valid”) bit in the Message Object Control Register is the main switch of the message object. The MultiCAN module only stores information in the message object during the frame reception process when MSGVAL is set (MSGVAL = 1).

Whenever MSGVAL is reset (MSGVAL = 0) by the CPU then the MultiCAN module stops all ongoing write accesses to the message object so that the message object may be reconfigured by the CPU in subsequent write accesses to the message object without being disturbed by the MultiCAN.

RTSEL

When the CPU re-configures a message object (.i.e. clears MSGVAL, modifies the message object and sets MSGVAL again) during CAN operation then the following scenario can occur:

1. The message object wins receive acceptance filtering.
2. The CPU clears MSGVAL to reconfigure the message object.
3. The CPU sets MSGVAL again after reconfiguration.
4. The end of the received frame is reached. As MSGVAL is set, the received data are stored in the message object, a message interrupt request is generated, gateway and fifo actions are processed etc.

The storage of the received data may be undesirable if the context of the message object has changed, because the old message object configuration has been used for acceptance filtering of the message.

Bit MOCTR.RTSEL (“Receive/Transmit Selected”) allows to disconnect a message object from an ongoing frame reception:

When a message object wins receive acceptance filtering then bit RTSEL is set (RTSEL = 1) by the MultiCAN in order to indicate an upcoming frame delivery. The MultiCAN checks RTSEL for value 1 upon successful frame reception in order to verify

that the object is still ready for receiving the frame. The received frame is stored in the message object (along with all subsequent actions such as message interrupts, FIFO & gateway actions, flag updates) only if RTSEL = 1.

When the user invalidates a message object during CAN operation (MSGVAL -> 0) then the user should clear RTSEL before setting MSGVAL again (latest with the same write access that sets MSGVAL) in order to prevent the storage of a frame that belongs to the old context of the message object. Thus message object reconfiguration should consist of the following sequence of steps:

1. Clear MSGVAL.
2. Reconfigure message object while MSGVAL = 0.
3. Clear RTSEL and set MSGVAL.

RXEN

Bit MOCTR.RXEN enables a message object for frame reception. A message object can receive CAN messages from the CAN bus only if RXEN = 1. The MultiCAN evaluates RXEN only during receive acceptance filtering. After receive acceptance filtering RXEN is ignored, i.e. the value of RXEN has no influence on the actual storage of a received message in a message object.

Bit RXEN enables a “soft phase out” of a message object: When the user clears RXEN then a currently received CAN message for which the message object has won acceptance filtering is still stored in the message object, but for subsequent messages the message object no longer wins receive acceptance filtering.

RXUPD, NEWDAT and MSGLST

An ongoing frame storage process is indicated with the RXUPD (“Receive Updating”) bit in the Message Object Control Register. The MultiCAN module sets RXUPD with the start and clears RXUPD with the end of a message object update (which consists of frame storage as well as flag updates).

After storing the received frame (identifier, IDE bit, DLC and for data frames also the data field) in the message object NEWDAT (“New Data”) is set by the MultiCAN. If NEWDAT was already set then also MSGLST (“Message Lost”) is set in order to indicate data loss.

The RXUPD and NEWDAT flags may be used by the CPU to read consistent frame data from the message object during ongoing CAN operation. The following steps are recommended:

1. Clear NEWDAT
2. Read message content (identifier, data etc.) from message object
3. Read Message Object Control Register and check that both NEWDAT and RXUPD are cleared. If this is not the case then goto back to step 1.
4. As step 3 was successful, the read message content is consistent, i.e. has not been updated by the MultiCAN while reading.

The bits RXUPD, NEWDAT and MSGLST work in the same fashion for the reception of data as well as remote frames.

20.2.8.2 Frame Transmission

The process of message object transmission is illustrated in **Figure 20-14**. In addition to copying the message content (identifier, IDE bit, RTR = DIR bit, DLC and for data frames also the data field) to the internal transmit buffer of the CAN node that the message object belongs to, also several status flags are served and monitored in order to enable consistent data handling.

The transmission process (after transmit acceptance filtering) of a given message object makes no difference between remote and data frames.

MSGVAL, TXRQ, TXEN0, TXEN1

For the MSGVAL bit the section **"MSGVAL" on Page 20-29** for frame reception is also valid for transmission.

A message may only be transmitted if all four bits MSGVAL ("Message Valid"), TXRQ ("Transmit Request"), TXEN0 ("Transmit Enable 0"), TXEN1("Transmit Enable 1") of the Message Object Control Register are set (1) (see also **Figure 20-9**). Although these bits are equivalent with respect to the transmission process, they have different semantics:

Table 20-4 Bits to set (1) in MOCTR for message transmission

Bit	Description
MSGVAL	**Message Valid** Main Switch of the Message Object
TXRQ	**Transmit Request** Standard Transmit Request bit. The CPU should set this bit whenever a message object shall be transmitted. TXRQ is cleared automatically at the end of the successful transmission, except when there are new data (indicated by NEWDAT = 1) to be transmitted. When the single transmit trial bit is set (STT = 1) in the Message Object Function Register then TXRQ is already cleared by the MultiCAN when the content of the message object is copied to the transmit frame buffer of the CAN node. A received remote request (i.e. remote frame received on CAN bus) sets bit TXRQ to request the transmission of the corresponding data frame.

Table 20-4 Bits to set (1) in MOCTR for message transmission (cont'd)

Bit	Description
TXEN0	**Transmit Enable 0** This bit may be temporarily cleared by the CPU to suppress the transmission of this object when it writes new content to the data field. This avoids transmission of inconsistent frames which consist of a mixture of old and new data. Remote requests are still accepted during TXEN0 = 0, but transmission of the data frame is suspended until the CPU re-enables transmission (TXEN0 = 1).
TXEN1	**Transmit Enable 1** This bit is used in transmit FIFOs to select the message object which is transmit active within the FIFO structure. For message objects which are not transmit FIFO elements TXEN1 may either be set to 1 permanently or be used as a second, independent transmission enable bit.

RTSEL

When a message object has been identified to be transmitted next (by acceptance filtering) then the MultiCAN set bit MOCTR.RTSEL ("Receive/Transmit Selected").

When the MultiCAN copies the message object to the transmit buffer it checks bit RTSEL and the message is transmitted only if RTSEL = 1.

After the successful transmission of the message bit RTSEL is checked again and message postprocessing is only performed if RTSEL = 1.

A complete reconfiguration of an operating message object should be done by means of the following steps:

1. Clear MSGVAL ("Message Valid").
2. Reconfigure message object while MSGVAL = 0.
3. Clear RTSEL and set MSGVAL.

Here clearing RTSEL ensures that the message object is disconnected from an ongoing/scheduled transmission and no message object processing (copying message to transmit buffer incl. clearing NEWDAT, clearing TXRQ, time stamp update, message interrupt etc.) within the old context of the object may occur after the message object becomes valid again, but within a new context.

NEWDAT

When the content of a message object has been transferred to the transmit buffer of the CAN node then bit NEWDAT ("New Data") is cleared to indicate that the transmit data of the message object are no longer new.

When the CAN transmission of the frame is successful and NEWDAT is still cleared (i.e. no new data have been copied to the message object in the meantime) then TXRQ (“Transmit Request”) is cleared automatically.

If, however, the NEWDAT bit has been set again by the CPU (because a new frame shall be transmitted) then TXRQ is not cleared in order to enable the transmission of the new data.

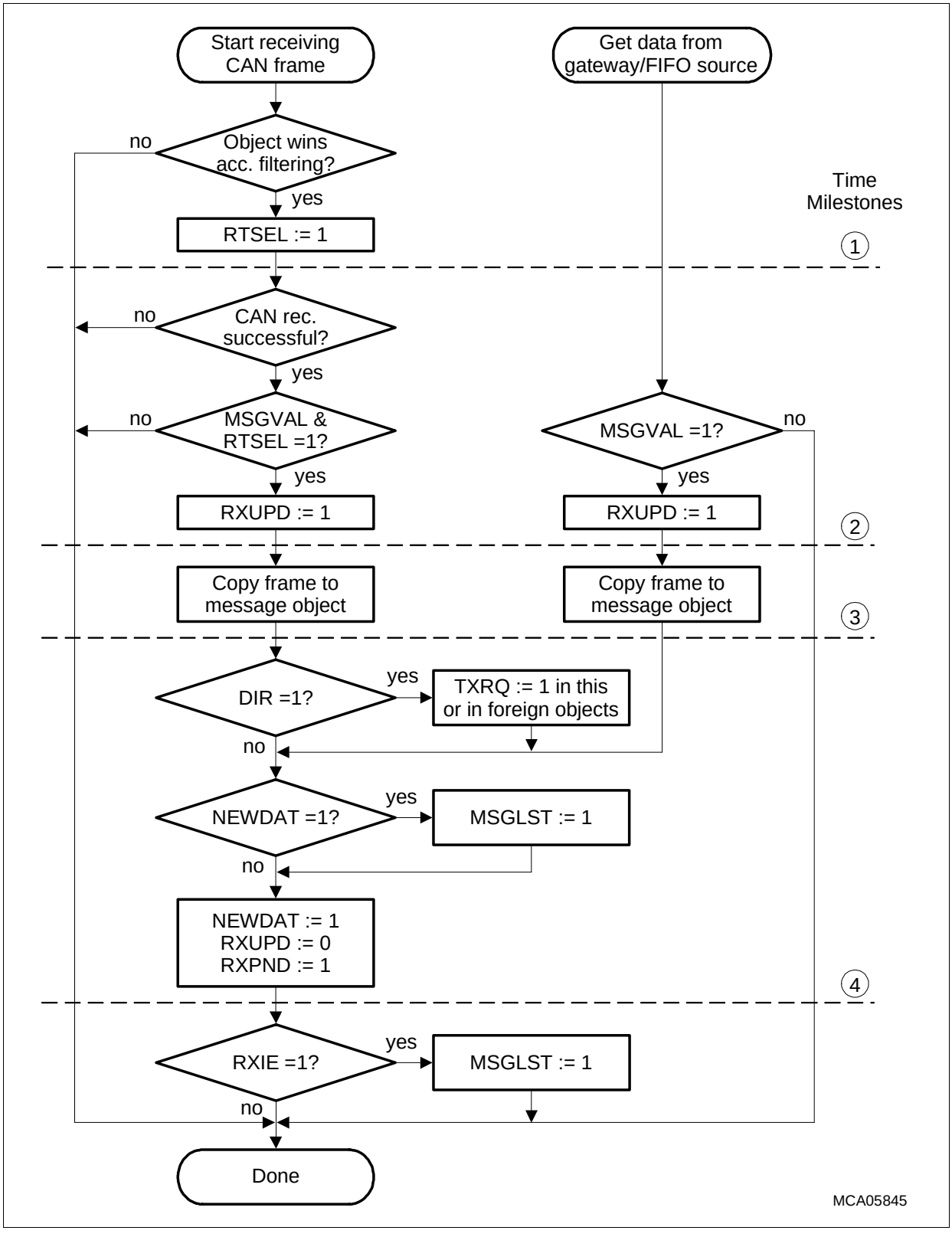


Figure 20-13 Data Delivery to Message Object by MultiCAN Module

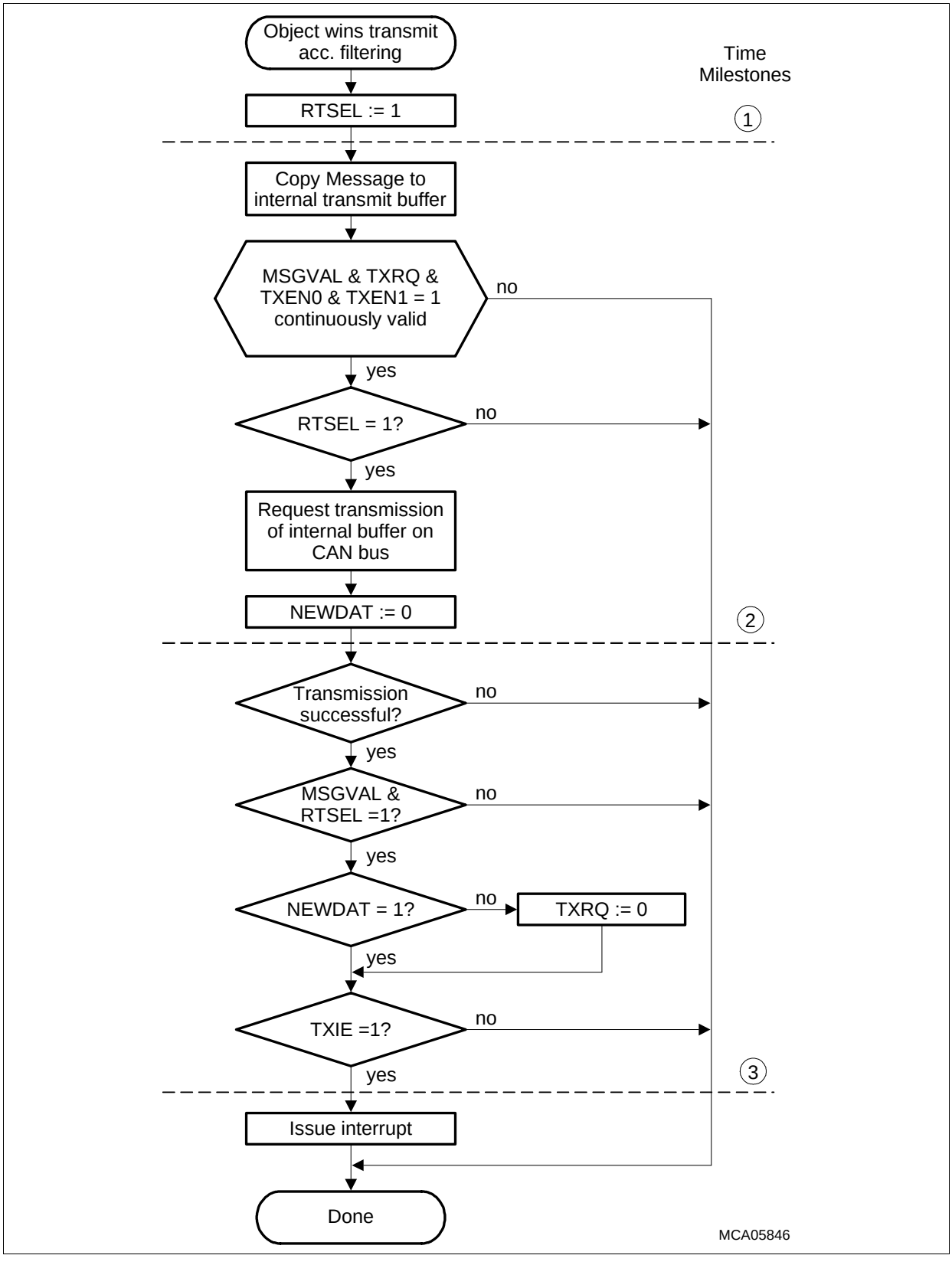


Figure 20-14 Transmission of a Message Object

20.2.9 Message Object Functionality

This section describes the functionality that is related to each individual Message Object.

20.2.9.1 Standard Message Object Mode

Standard message mode is selected via MMC = 0000_B in the Message Object Function Control Register of the message object. In this mode a message object may transmit and receive CAN frames according to the basic rules as described in the previous sections. Additional services such as Single Data Transfer Mode or Single Transmit Trial (see sections below) are available and may be selected individually by the user.

20.2.9.2 Single Data Transfer Mode

Single Data Transfer Mode is a useful feature in order to broadcast data over the CAN bus without unintended doubling of information. Single Data Transfer Mode is selected via bit SDT in the Message Object Function Register of the message object.

Message Reception

When a received message is stored in a message object and further messages are stored in the same message object before the CPU reads the first message object, then the content of the first message gets lost and is replaced with the content of the subsequent messages (indicated by MSGLST = 1).

If SDT = 1 (Single Data Transfer Mode activated) then the MultiCAN controller automatically clears the MSGVAL bit of the message object after the storage of a received data frame to prevent the reception of further messages.

The reception of a remote frame does not lead to the clearance of MSGVAL.

Message Transmission

When a message object receives a series of multiple remote requests then it transmit several data frames in response to the requests. If the data within the message object has not been updated in the time between the transmissions, the same data may be represented more than once on the CAN bus.

In Single Data Transfer Mode (SDT = 1) this is avoided because the MultiCAN controller automatically clears MSGVAL after the successful transmission of a data frame.

The transmission of a remote frame does not clear MSGVAL.

20.2.9.3 Single Transmit Trial

If the bit STT in the message object function register is set (STT = 1) then the transmission request is cleared (TXRQ := 0) when the frame content of the message

object has been copied to the internal transmit buffer of the CAN node. Thus the transmission of the message object is not tried again when it fails due to CAN bus errors.

20.2.9.4 Message Object FIFO Structure

In case of high CPU load it may be difficult to process a series of CAN frames in time. This may happen for the short term reception of multiple messages as well as the transmission of a series with tight due date.

Therefore a FIFO buffer structure has been implemented in order to avoid loss of incoming messages and to minimize the setup time for outgoing messages. The FIFO structure may also be used to automate the reception or transmission of a series of CAN messages and to generate a single message interrupt when the whole series is done.

There may be as many FIFOs in parallel as are required by the application. The number of FIFOs and their size are only limited by the number of message objects available. A FIFO may be installed, resized and deinstalled any time, even during CAN operation.

The basic structure of a FIFO is shown in **Figure 20-15**. A FIFO consists of a single base object (shown on the left side) and several slave objects (shown on the right side). The slave objects are chained together in the same list structure. The base object may be allocated to any list. Although **Figure 20-15** shows the base object as a separate item apart from the slave objects, it is also possible to integrate the base object at any place into the chain of slave objects, so that the base object is slave object, too (not possible for gateways). The FIFO structure fully relies on the list structure. The absolute object numbers of the message objects have no impact on the operation of the FIFO.

The base object needs not be allocated to the same list as the slave objects. Only the slave object must be allocated to a common list (as they are chained together). The BOT, CUR and TOP pointer link the base object to the slave objects, no matter whether the base object is allocated to the same or to another list than the slave objects.

The absolute minimum FIFO would consist of a single message object which is both FIFO base and FIFO slave (not very useful). The biggest possible FIFO would use all message objects of the MultiCAN module. Any sizes between these extremes are possible.

In the FIFO base object the boundaries of the FIFO are defined. The BOT field in the FIFO/Gateway Pointer Register of the base object points to the first (bottom) slave element in the FIFO. The TOP field in the FIFO/Gateway Pointer Register of the base object points to the last (top) slave element.

The CUR field in the FIFO/Gateway Pointer Register of the FIFO base object points to the actual slave object selected by the MultiCAN for message transfer. When a message transfer takes place with this object then CUR is moved to the next position. If CUR has already reached the top of the FIFO (CUR = TOP) then it is wrapped around to the bottom of the FIFO (CUR := BOT). Otherwise CUR is moved to the next message object in the list structure of the slave objects (CUR := PNEXT of current object). This scheme

yields a circular FIFO structure where the fields BOT and TOP just establish the link from the last to the first element, which is missing in the linear structure.

The SEL field in the FIFO/Gateway Pointer register of the base object may be used for monitoring purposes. It allows to select any slave object and to generate a message interrupt if the CUR pointer reaches the value of SEL. Thus SEL offers a convenient way to determine the end of a predefined series of message transfers, or it may be used to issue a warning to the CPU when the FIFO gets full.

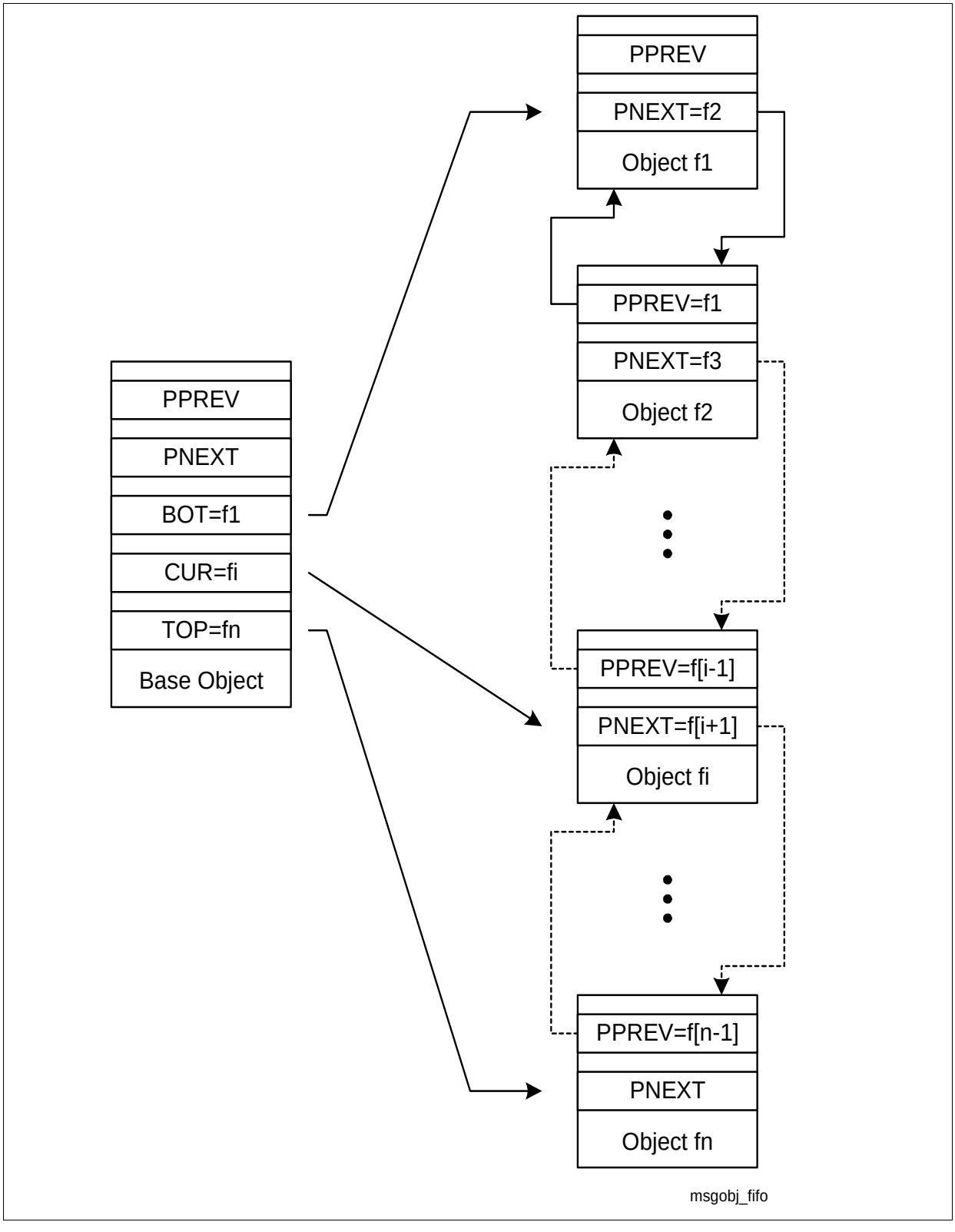


Figure 20-15 FIFO Structure with FIFO Base and n FIFO Destinations (Slaves)

20.2.9.5 Receive FIFO

The Receive FIFO structure is used to buffer incoming (received) remote or data frames.

A Receive FIFO is selected via MMC = 0001_B in the Message Object Function Control Register of the FIFO base object. This MMC code automatically designates the message object as FIFO base object. The message mode of the FIFO slave objects are not relevant for the operation of the Receive FIFO.

When the FIFO base object receives a frame from the CAN node it belongs to, then the frame is not stored in the base object. Instead the message is stored in the message object that is selected by the CUR pointer in the FIFO/Gateway Pointer Register of the FIFO base object.

The message object selected by CUR receives the CAN message as if it were the direct receiver of the message. However, MMC = 0000_B is implicitly assumed for the FIFO slave, i.e. a standard message delivery is performed. The actual message mode (MMC) of the FIFO slave is ignored. There is also no extra acceptance filtering to match the received frame against the identifier, IDE bit and DIR bit of the slave object.

When the FIFO base object receives a CAN frame then the MultiCAN moves the current pointer CUR to the next message object in the FIFO structure, which will then be used to store the next incoming message. The old value of CUR is used for the current transfer.

IF bit OVIE is set in the Message Object Function Register of the FIFO base object and the pointer CUR reaches the value stored in SEL then a FIFO overflow interrupt request is generated. The interrupt request is generated on interrupt output line TXINP (TXINP of the base object) immediately after the storage of the received frame into the slave object. Transmit interrupts are still generated if TXIE is set.

A CAN message is stored in a FIFO slave only if MSGVAL = 1 in both FIFO base and slave object.

In order to avoid direct reception of a message by a slave message object, as if it was an independent message object and not a part of a FIFO, the bit RXEN of each slave object must be cleared. The setting of the bit RXEN is “don’t care” only if the slave object is located in a list not assigned to a CAN node.

20.2.9.6 Transmit FIFO

The Transmit FIFO structure is used to buffer a series of data or remote frames to be transmitted. A transmit FIFO consists of one base message object and one or more slave message objects.

A Transmit FIFO base object is selected via MMC = 0010_B in the Message Object Function Control Register of the FIFO base object. Unlike the Receive FIFO the Transmit FIFO requires the explicit declaration of the FIFO slave objects via MMC = 0011_B. The CUR pointer of all slave objects must point back to the Transmit FIFO Base Object (to be initialized by user).

The TXEN1 bits of all message objects except the one which is selected by the CUR pointer of the base object must be cleared (to be initialized by user). TXEN1 of the message object selected by CUR must be set. CUR may be initialized to any FIFO slave object.

When tagging the message objects of the FIFO valid to start the operation of the FIFO then the base object must be tagged valid (MSGVAL := 1) first.

When a Transmit FIFO shall be deinstalled during operation, then the slave objects must be tagged invalid (MSGVAL := 0) first.

The Transmit FIFO uses the TXEN1 bit in the Message Object Control Register of all FIFO elements to select the actual message for transmission. Transmit acceptance filtering evaluates TXEN1 for each message object and a message object may win transit acceptance filtering only if TXEN1 is set. When a FIFO element has transmitted a message then in addition to standard transmit postprocessing (clear TXRQ, transmit interrupt etc.) the MultiCAN clears TXEN1 in that message object and moves the CUR pointer in the corresponding FIFO base object to the next message object to be transmitted. TXEN1 is set automatically in the next message object. Thus TXEN1 moves along the FIFO structure like a token to select the active element.

IF bit OVIE is set in the Message Object Function Register of the FIFO base object and the pointer CUR reaches the value stored in SEL then a FIFO overflow interrupt request is generated. The interrupt request is generated on interrupt output line as defined by RXINP (RXINP of the base object) when postprocessing of the received frame is done. Receive interrupts are still generated for the Transmit FIFO base object if bit RXIE is set.

20.2.9.7 Gateway Mode

The Gateway Mode allows to establish an automatic information transfer between two independent CAN bus systems without CPU interaction.

The Gateway Mode operates on message object level. In Gateway mode, information is transferred between two message objects, resulting in an information transfer between the two CAN nodes to which the message objects are allocated. A gateway may be established between any pair of CAN nodes and there may be as many gateways as there are message objects available to build the gateway structure.

Gateway Mode is selected via MMC = 0100_B in the Message Object Function Control Register of the gateway source object. The gateway destination object is selected by the CUR pointer in the FIFO/Gateway Pointer Register of the source object. The gateway destination object just needs to be valid (MSGVAL = 1), all other settings are not relevant for the information transfer from the source object to the destination object.

A gateway source object behaves like a standard message objects, but when a CAN frame has been received and stored in the source object, some additional actions are performed by the MultiCAN (**Figure 20-16**):

1. If bit DLCC is set in the Message Object Function Register of the source object, then the DLC code is copied from the source object to the destination object.
2. if bit IDC is set in the Message Object Function Register of the source object, then the identifier and the IDE bit are copied from the source object to the destination object.
3. if bit DATC is set in the Message Object Function Register of the source object, then the data field is copied from the source object to the destination object.
4. If bit GDFS is set in the Message Object Function Register of the source object, then TXRQ is set in the Message Object Control Register of the destination object.
5. RXPND and NEWDAT are set in the Message Object Control Register of the destination object.
6. A message interrupt request is generated for the destination object if RXIE is set in the Message Object Control Register of the destination object.
7. The current pointer CUR in the FIFO/Gateway Pointer Register of the source object is moved to the next destination object according to the FIFO rules as described in **Chapter 20.2.9.4**. A gateway with a single (static) destination object is obtained by means of setting TOP = BOT = CUR = destination object.

The link from the source to the destination object works in the same way as the link from a FIFO source to a FIFO slave. This means that a gateway with an integrated destination FIFO may be created (**Figure 20-15**), where the object on the left in **Figure 20-15** is the gateway source object and the message objects on the right side are the gateway destination objects.

The gateway works in the same way for the reception of data frames (source object is receive object, i.e. DIR = 0) as wall as for the reception of remote frames (source object is transmit object).

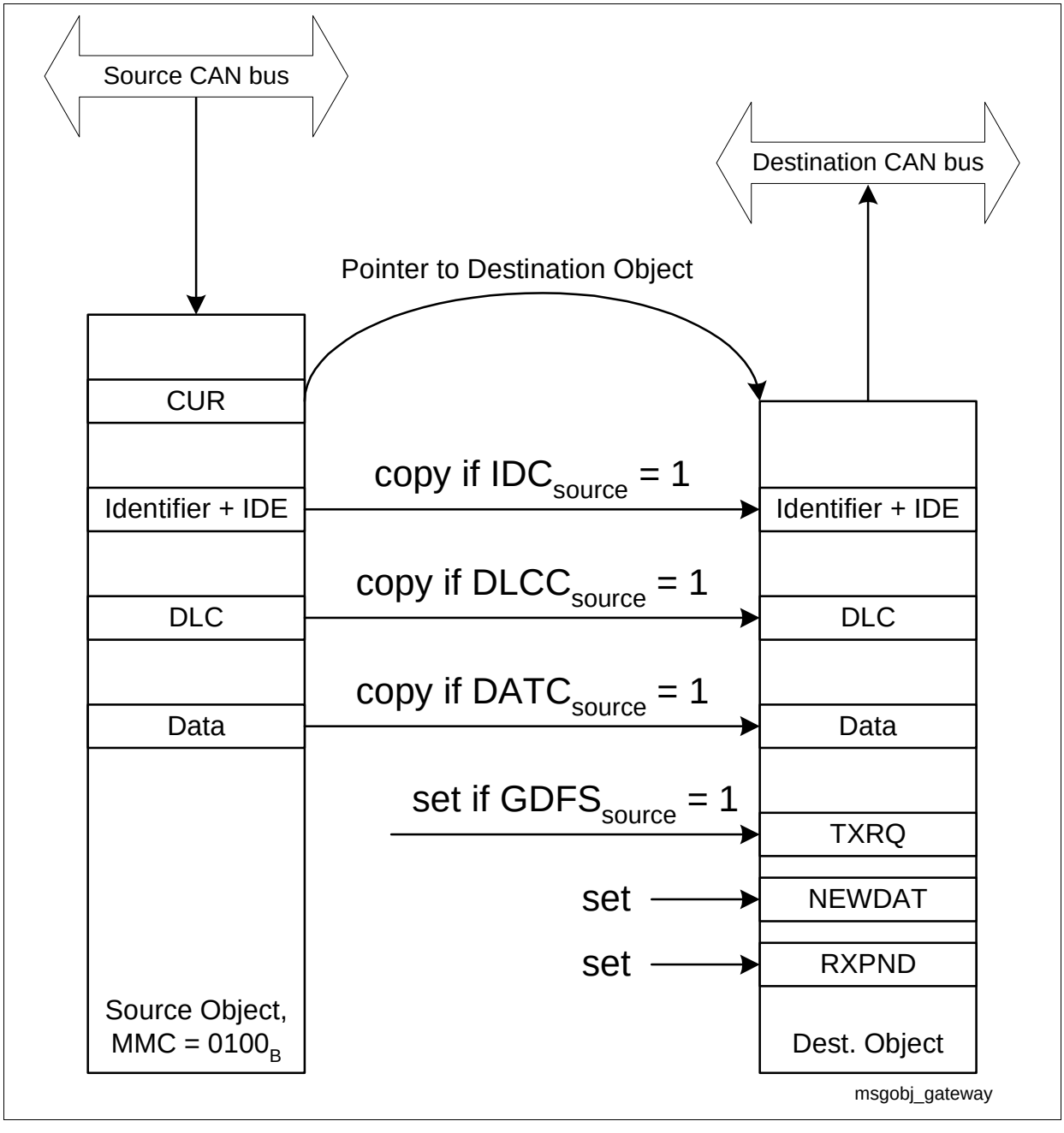


Figure 20-16 Gateway Transfer from Source to Destination

20.2.9.8 Foreign Remote Requests

When a remote frame received on a CAN node is stored in a message object, then a transmit request is set in order to trigger the answer (data frame transmission) to the request or to automatically issue a secondary request. If bit FRREN is cleared (FRREN = 0) in the Function Control register of the message object where the remote request is stored, then TXRQ is set in the Control Register of the same message object.

If bit FRREN is set (FRREN = 1: foreign remote request enabled) then TXRQ is set in the message object that is referenced by pointer CUR in the FIFO/Gateway Pointer Register. The value of CUR is, however, not changed by this feature.

Although the foreign remote request feature works independently from the selected message mode, it is especially useful for gateways to issue a remote request on the source of a gateway upon the reception of a remote request on the gateway destination. According to the setting of FRREN in the gateway destination object there are two ways to handle remote requests that appear on the destination side (assuming that the source object is a receive object and the destination is a transmit object, i.e. DIR_{source} = 0 and $DIR_{destination}$ = 1):

FRREN = 0 in the Gateway Destination Object

1. A remote frame is received by gateway destination.
2. TXRQ is set automatically in the gateway destination object.
3. A data frame with the current data stored in the destination object is transmitted on the destination bus.

FRREN = 1 in the Gateway Destination Object

1. A remote frame is received by gateway destination.
2. TXRQ is set automatically in the gateway source object (must be referenced by CUR pointer of the destination object).
3. A remote request is transmitted by the source object (which is a receive object) on the source CAN bus.
4. The receiver of the remote request responds with a data frame on the source bus.
5. The data frame is stored in the source object.
6. The data frame is copied to the destination object (gateway action).
7. TXRQ is set in the destination object (assuming $GDFS_{source}$ = 1).
8. The new data stored in the destination object is transmitted on the destination bus, as response to the initial remote request on the destination bus.

20.2.10 MultiCAN Kernel Registers

The register set of the MultiCAN module consists of three distinct subsets:

1. The **Global Module Registers** apply to the whole MultiCAN module and exist only once.
2. The **CAN Node Registers** apply to a single CAN node and thus exist once for each CAN node.
3. The collection of **Message Object Registers** defines a single message object and thus exists once for each message object.

20.2.10.1 Register Address Map

Table 20-5 shows the address map of the MultiCAN module with respect to the base address of the MultiCAN module.

Table 20-5 MultiCAN Address Map (Relative to MultiCAN Base Address)

Register Group	Start address	Total Range
Global Module Registers	$+100_H$	$+100_H$ to $+ 1FF_H$
CAN Node Registers for CANx, x = 0 - 4	$+200_H$	$+200_H$ to $+ 2FF_H$
Message Objects n = 0 - 127	$+1000_H$	$+1000_H$ to $+ 13FF_H$

Global Module Registers

The global module registers exist only once. They are listed in **Table 20-6** with their relative address with respect to the start address of the Global Module Registers.

Table 20-6 Relative Addresses of Global Module Registers

Register	Rel. Address	Full Name of Register
LIST0L	100_H	List Registers 0 Low
LIST0H	102_H	List Registers 0 High
LIST1L	104_H	List Registers 1 Low
LIST1H	106_H	List Registers 1 High
LIST2L	108_H	List Registers 2 Low
LIST2H	$10A_H$	List Registers 2 High
LIST3L	$10C_H$	List Registers 3 Low
LIST3H	$10E_H$	List Registers 3High
LIST4L	110_H	List Registers 4 Low
LIST4H	112_H	List Registers 4 High
LIST5L	114_H	List Registers 5 Low
LIST5H	116_H	List Registers 5 High
LIST6L	118_H	List Registers 6 Low
LIST6H	$11A_H$	List Registers 6 High
LIST7L	$11C_H$	List Registers 7 Low
LIST7H	$11E_H$	List Registers 7 High
MSPND0L	140_H	Message Pending Registers 0 Low
MSPND0H	142_H	Message Pending Registers 0 High
MSPND1L	144_H	Message Pending Registers 1 Low
MSPND1H	146_H	Message Pending Registers 1 High
MSPND2L	148_H	Message Pending Registers 2 Low
MSPND2H	$14A_H$	Message Pending Registers 2 High
MSPND3L	$14C_H$	Message Pending Registers 3 Low
MSPND3H	$14E_H$	Message Pending Registers 3 High
MSPND4L	150_H	Message Pending Registers 4 Low
MSPND4H	152_H	Message Pending Registers 4 High
MSPND5L	154_H	Message Pending Registers 5 Low

Table 20-6 Relative Addresses of Global Module Registers

Register	Rel. Address	Full Name of Register
MSPND5H	156_H	Message Pending Registers 5 High
MSPND6L	158_H	Message Pending Registers 6 Low
MSPND6H	$15A_H$	Message Pending Registers 6 High
MSPND7L	$15C_H$	Message Pending Registers 7 Low
MSPND7H	146_H	Message Pending Registers 7 High
MSID0	180_H	Message Index Registers 0
MSID1	184_H	Message Index Registers 1
MSID2	188_H	Message Index Registers 2
MSID3	$18C_H$	Message Index Registers 3
MSID4	190_H	Message Index Registers 4
MSID5	194_H	Message Index Registers 5
MSID6	198_H	Message Index Registers 6
MSID7	$19C_H$	Message Index Registers 7
MSIMASKL	$1C0_H$	Message Index Mask Register Low
MSIMASKH	$1C2_H$	Message Index Mask Register High
PANCTRL	$1C4_H$	Panel Control Register Low
PANCTRH	$1C6_H$	Panel Control Register High
MCR	$1C8_H$	Module Control Register
MITR	$1CC_H$	Module Interrupt Trigger Register
-	$+120_H$ ... $+13E_H$ $+148_H$ ... $+17E_H$ $+188_H$ ... $+1BE_H$ $+1CE_H$ ... $+1FE_H$	Reserved

CAN Node Registers

The registers of a CAN node are located at consecutive 32 bit addresses according to**Table 20-7** which shows the relative address of the 32 bit CAN Node Registers with respect to the base address of CAN node register block. The CAN Node Register block exists once for each CAN node.

Table 20-7 Relative Addresses of CAN Node Registers

Register	Rel. Address	Full Name of Register
NCR	$+00_H$	CAN Node Control Register
NSR	$+04_H$	CAN Node Status Register
NIPR	$+08_H$	CAN Node Interrupt Pointer Register
NPCR	$+0C_H$	CAN Node Port Control Register
NBTRL	$+10_H$	CAN Node Bit Timing Register Low
NBTRH	$+12_H$	CAN Node Bit Timing Register High
NECNTL	$+14_H$	CAN Node Error Counter Register Low
NECNTH	$+16_H$	CAN Node Error Counter Register High
NFCRL	$+18_H$	CAN Node Frame Counter Register Low
NFCRH	$+1A_H$	CAN Node Frame Counter Register High
-	$+1C_H$ to $+FE_H$	Reserved

Message Object Registers

The registers of a message object are located at consecutive 32 bit addresses according to **Table 20-8** which shows the relative address of the 32 bit Message Object Registers with respect to the base address of the Message Object.

Table 20-8 Relative Addresses of Message Object Registers

Register	Rel. Address	Full Name of Register
MOFCRL	$+00_H$	Message Object Function Control Register Low
MOFCRH	$+02_H$	Message Object Function Control Register High
MOFGPRL	$+04_H$	Message Object FIFO/Gateway Pointer Reg. Low
MOFGPRH	$+06_H$	Message Object FIFO/Gateway Pointer Reg. High
MOIPRL	$+08_H$	Message Object Interrupt Pointer Register Low
MOIPRH	$+0A_H$	Message Object Interrupt Pointer Register High
MOAMRL	$+0C_H$	Message Object Acceptance Mask Register Low
MOAMRH	$+0E_H$	Message Object Acceptance Mask Register High
MODATALL	$+10_H$	Message Object Data Register Low Low
MODATALH	$+12_H$	Message Object Data Register Low High
MODATAHL	$+14_H$	Message Object Data Register High Low

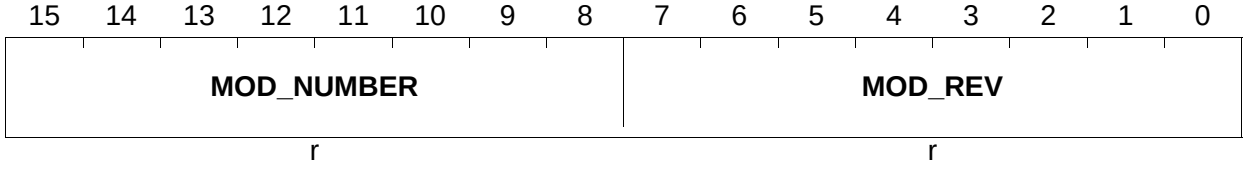
Table 20-8 Relative Addresses of Message Object Registers

Register	Rel. Address	Full Name of Register
MODATAHH	$+16_H$	Message Object Data Register High High
MOARL	$+18_H$	Message Object Arbitration Register Low
MOARH	$+1A_H$	Message Object Arbitration Register High
MOCTRL	$+1C_H$	Message Object Control Register Low
MOCTRH	$+1E_H$	Message Object Control Register High

Registers Description

20.2.10.2 Module Identification Register

ID
Module Identification Register (08_H) Reset Value: 4001_H



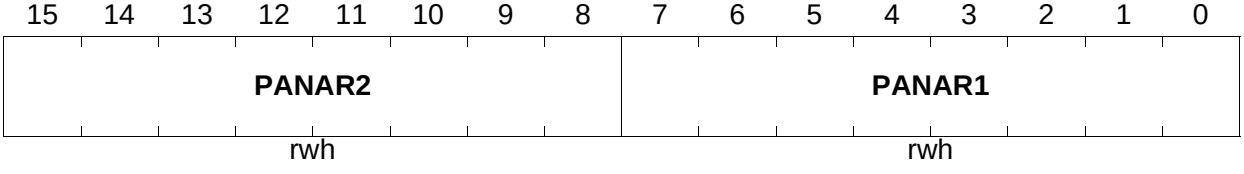
Field	Bits	Type	Description
MOD_REV	[7:0]	r	**Module Revision Number Value** Bits 7-0 bits are used for module revision numbering. The value of the module revision number starts with 01_H (first revision), 02_H, 03_H, ... up to FF_H.
MOD_NUMBER	[15:8]	r	**Module Identification Number Value** Bits 15-8 are used for module identification. The MultiCAN has the module number 40_H.

20.2.10.3 Command Panel

All list operations such as allocation, deallocation and relocation of message objects within the list structure are performed via the Command Panel. It is not possible to modify the list structure directly by means of writing to the message objects and the LIST registers.

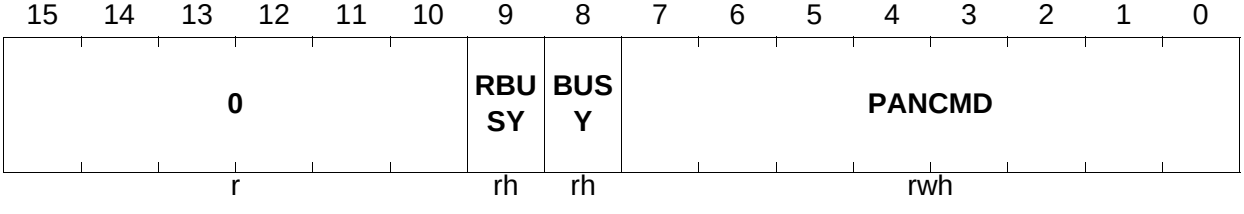
A new command is started by means of writing the command arguments and the command code to the Panel Control Register.

PANCTRH
Panel Control Register **($1C6_H$)** **Reset Value: 0000_H**



Field	Bits	Type	Description
PANAR1	[7:0]	rwh	**Panel Argument 1**
PANAR2	[15:8]	rwh	**Panel Argument 2**

PANCTRL
Panel Control Register **($1C4_H$)** **Reset Value: 0301_H**



Field	Bits	Type	Description
PANCMD	[7:0]	rwh	**Panel Command** A new command is started by means of writing the command number to PANCMD. At the end of a panel command the NOP (no operation) command code is automatically written to PANCMD.

Field	Bits	Type	Description
BUSY	8	rh	**Panel Busy** 0 Panel has finished command and is ready to accept a new command. 1 Panel operation is in progress.
RBUSY	9	rh	**Result Busy** 0No update of PANAR1 and PANAR2 is scheduled by the list controller. 1A list command is running (BUSY = 1) that will write results to PANAR1 and PANAR2, but the results are not yet available.
0	[10:15]	r	**reserved;** returns ’0’ if read; should be written with ’0’;

Panel Commands

A panel operation consists of a command code to be written to PANCMD and up to 2 panel arguments (PANAR1, PANAR2). Commands that have a return value deliver it to the PANAR1 field. Commands that deliver an error flag post it to bit 7 of PANAR2.

Table 20-9 Panel Commands

Code	PANAR2	PANAR1	Command Description
0			**No Operation** Writing value 0 to PANCMD has no effect. No new command is started.
1	**Result:** Bit 7 : ERR Bit 6-0 : undefined		**Initialize Lists** Run the initialization sequence to reset the CTRL and LIST field of all message objects and the list registers LIST[7:0] to their reset values. This results in the deallocation of all message objects. The initialization command requires that bits INIT and CCE are set in the Node Control Register of all CAN nodes 0-4. An ERR bit (bit 7 of PANAR2) reports the success of the operation: 0Success 1Not all INIT and CCE bits are set. Thus no initialization is performed. The initialization command is automatically performed with each reset of the MultiCAN module, but with the exception that all message object registers are reset.
2	**Argument:** List Index	**Argument:** Message Object Number	**Static Allocate** Allocate a given message object to a list. The message object is removed from the list that it currently belongs to and appended to the end of the list. given by PANAR2. This command is also used to deallocate a message object. In this case the target list is the list of unallocated elements. (PANAR2 = 0).

Table 20-9 Panel Commands

Code	PANAR2	PANAR1	Command Description
3	**Argument:** List Index **Result:** Bit 7 : ERR Bit 6-0 : undefined	**Result:** Message Object Number	**Dynamic Allocate** Allocate the first message object of the list of unallocated objects to the selected list. The message object is appended to the end of the list. The message number of the message object is returned in PANAR1. An ERR bit (bit 7 of PANAR2) reports the success of the operation: 0Success. 1The operation has not been performed because the list of unallocated elements was empty.
4	**Argument:** Destination Object Number	**Argument:** Source Object Number	**Static Insert Before** Remove a message object (source object) from the list that it currently belongs to and insert it before a given destination object into the list structure of the destination object. The source object thus becomes the predecessor of the destination object.
5	**Argument:** Destination Object Number **Result:** Bit 7 : ERR Bit 6-0 : undefined	**Result:** Object Number of inserted object	**Dynamic Insert Before** Insert a new message object before a given destination object. The new object is taken from the list of unallocated elements (the first element is chosen). The number of the new object is delivered as result to PANAR1. An ERR bit (bit 7 of PANAR2) reports the success of the operation: 0Success. 1The operation has not been performed because the list of unallocated elements was empty.
6	**Argument:** Destination Object Number	**Argument:** Source Object Number	**Static Insert Behind** Remove a message object (source object) from the list that it currently belongs to and insert it behind a given destination object into the list structure of the destination object. The source object thus becomes the successor of the destination object.

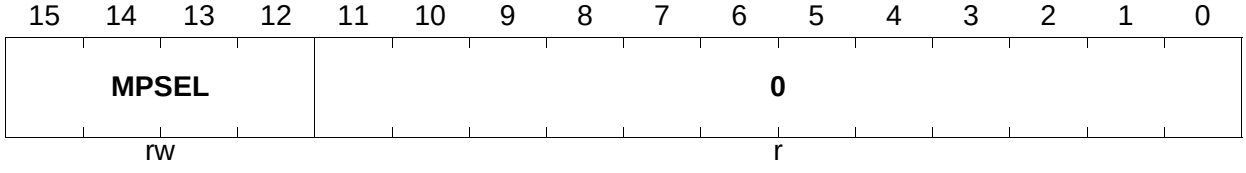
Table 20-9 Panel Commands

Code	PANAR2	PANAR1	Command Description
7	**Argument:** Destination Object Number **Result:** Bit 7 : ERR Bit 6-0 : undefined	**Result:** Object Number of inserted object	**Dynamic Insert Behind** Insert a new message object behind a given destination object. The new object is taken from the list of unallocated elements (the first element is chosen). The number of the new object is delivered as result to PANAR1. An ERR bit (bit 7 of PANAR2) reports the success of the operation: 0Success. 1The operation has not been performed because the list of unallocated elements was empty.
8 - 255	**-**	**-**	**Reserved**

20.2.10.4 Module Setup

The Module Control Register contains basic settings to define the operation of the module.

MCR
Module Control Register **($1C8_H$)** **Reset Value: 0000_H**

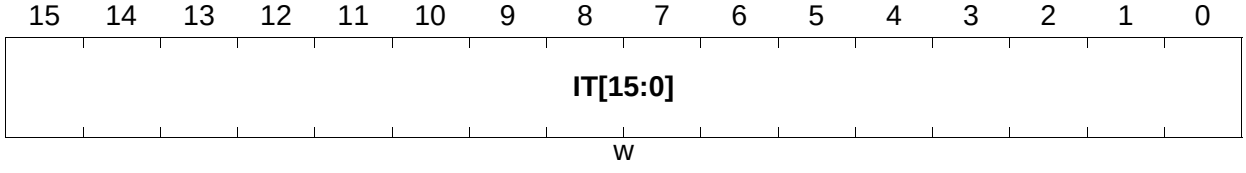


Field	Bits	Type	Description
MPSEL	[15:12]	rw	**Message Pending Selector** MPSEL allows to calculate the bit position of the message pending bit to be set after a message reception/transmission interrupt from a mixture of RXINP,TXINP and MPN (Message Pending Number). With the definitions INP ... RXINP upon message reception, TXINP upon message transmission MPN ... 8 bit message pending number the effective position of the message pending bit is calculated according to the formula POS = ((INP & MPSEL) << 4) | (MPN & (~MPSEL<< 4)) | (MPN & = $0x0F_H$) If MPSEL = 0 then the position is simply given by the message pending number MPN. If MPSEL = 1111_B then the upper 4 bits of the position is given by the interrupt output line pointer INP and the lower 4 bits are taken from MPN.
0	[11:0]	r	**reserved;** returns ’0’ if read; should be written with ’0’;

20.2.10.5 Interrupt Trigger Register ITR

The Interrupt Trigger Register ITR allows to trigger interrupt requests on each interrupt output line by software.

MITR
Module Interrupt Trigger Register **($1CC_H$)** **Reset Value: 0000_H**



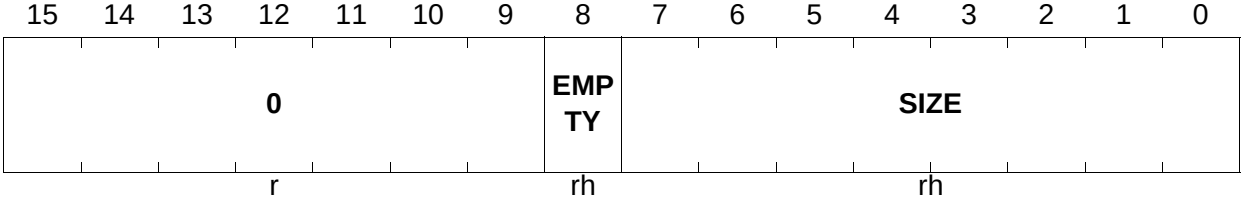
Field	Bits	Type	Description
IT[15:0]	[15:0]	w	**Interrupt Trigger** Writing value 1 to bit n (n = 15-0) generates an interrupt request on interrupt output line n. Writing value 0 has no effect. Reading delivers always 0. More than one interrupt request may be generated at the same time by means of writing 1 to several bit positions of IT with a single write access.

20.2.10.6 List Pointer

Each CAN node has an own list which defines the message objects that are allocated to the respective node. In addition to that there is the list of all unallocated objects and finally a general purpose user list which is not associated to a CAN node. Each list is assigned a list index according to **Table 20-3 “List Indices” on Page 20-15**.

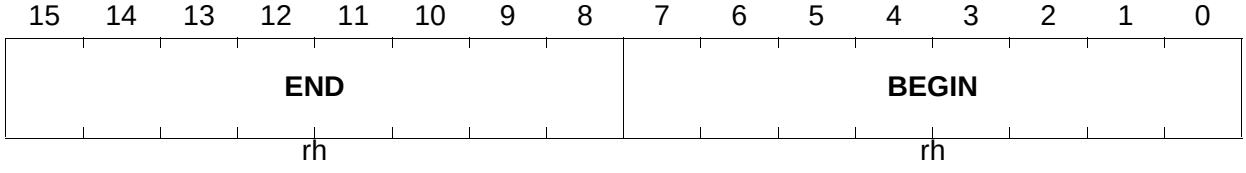
Each list is terminated with a List Register which defines the first and the last element in the list.

LIST0H
List Register 0 High **(102_H)** **Reset Value: $007F_H$**
LISTyH (y = 1-7)
List Register y High **(102_H+y*4)** **Reset Value: 0100_H**



Field	Bits	Type	Description
SIZE	[7:0]	rh	**Size of List** The number of elements in the list l is given by #elements = SIZE + 1, provided the list is not empty. If the list l is empty, the value of SIZE is zero.
EMPTY	8	rh	**List Empty Indication** 0At least one message object is allocated to list l. 1No message object is allocated to the list l.
0	[15:9]	r	**Reserved;** read as 0; should be written with 0.

LIST0L
List Register 0 Low **(100_H)** **Reset Value: $7F00_H$**
LISTxL (x = 1-7)
List Register x Low **(100_H+x*4)** **Reset Value: 0000_H**

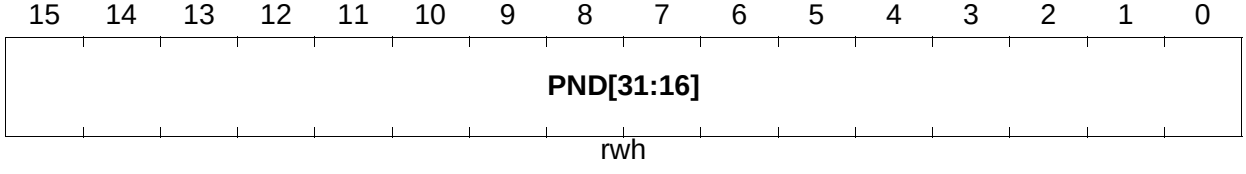


Field	Bits	Type	Description
BEGIN	[7:0]	rh	**List Begin** Pointer to the first message object in the list l.
END	[15:8]	rh	**END Pointer** Pointer to the last message object in the list l.

20.2.10.7 Message Notifications

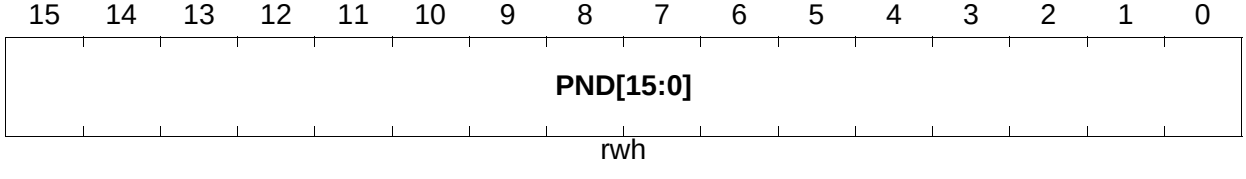
When a message object generates an interrupt request upon the transmission or reception of a message, then the request is routed to the interrupt output line selected by TXINP or RXINP of the message object. As there are more message objects than interrupt output lines, an interrupt routine typically processes requests from more than one message object. Therefore a priority selection mechanism is implemented in the MultiCAN module to select the highest priority object within a collection of message objects.The Message Pending Register contains the interrupt pending.

MSPNDkH (k = 0-7)
Message Pending Register k High (142_H+k*4) **Reset Value: 0000_H**



Field	Bits	Type	Description
PND[31:16]	[15:0]	rwh	**Message Pending** When a message interrupt occurs then the message object sets a bit in one of the MSPND register, where the bit position is given by the MPN[4:0] field of the IPR register of the message object. The register selection n is given by the higher bits of MPN. The register bits may be cleared by SW (write 0), but writing 1 has no effect.

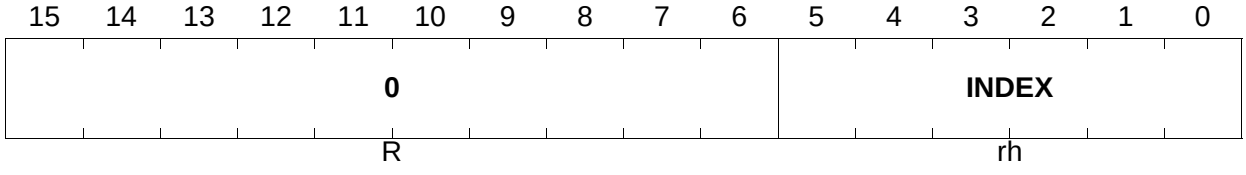
MSPNDkL (k = 0-7)
Message Pending Register k Low (140_H+k*4) **Reset Value: 0000_H**



Field	Bits	Type	Description
PND[15:0]	[15:0]	rwh	**Message Pending** The same as PND[31:16]

Each Message Pending Register has a Message Index Register associated to it. The Message Index Register shows the active (set) pending bit with lowest bit position within groups of pending bits.

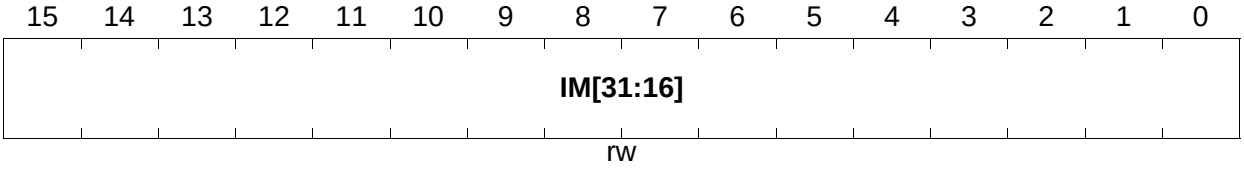
MSIDk (k = 0-7)
Message Index Register k **(180_H+k*4)** **Reset Value: 0020_H**



Field	Bits	Type	Description
INDEX	[5:0]	rh	**Message Pending Index** The value of INDEX is given by the bit position i of the pending bit of MSPNDk with the following properties: 1. MSPNDk[i] & IM[i] = 1 2. i = 0 or MSPNDk[i-1:0] & IM[i-1:0] = 0 If no bit of MSPNDk satisfies these conditions then INDEX reads 100000_B. Thus INDEX shows the position of the first pending bit of MSPNDk, where only those bits of MSPNDk which are selected in the Message Index Mask Register are taken into account.
0	[15:6]	r	**Reserved:** read as 0; should be written with 0.

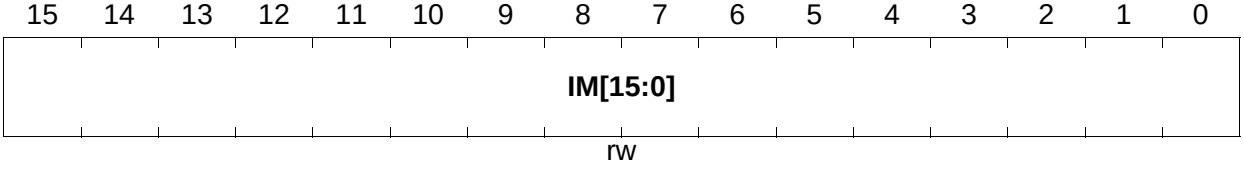
The Message Index Mask Register selects individual bits for the calculation of the Message Pending Index. The Message Index Mask Register is used commonly for all Message Pending registers and their associated Message Index registers.

MSIMASKH
Message Index Mask Register High (1C2$_H$) **Reset Value: 0000$_H$**



Field	Bits	Type	Description
IM[31:16]	[15:0]	rw	**Message Index Mask** Only those bits in MSPNDk for which the corresponding Index Mask bits are set contribute to the calculation of the Message Index.

MSIMASKL
Message Index Mask Register Low (1C0$_H$) **Reset Value: 0000$_H$**



Field	Bits	Type	Description
IM[15:0]	[15:0]	rw	**Message Index Mask** Only those bits in MSPNDk for which the corresponding Index Mask bits are set contribute to the calculation of the Message Index.

20.2.11 CAN Node Specific Registers

The CAN node specific registers exist once for each CAN node of the MultiCAN module.They contain information that is directly related to the operation of the CAN nodes and which may not be shared among the nodes.

The Node Control Register contains basic settings that define the operation of the CAN node and the interaction of the CAN node with the message objects.

NCRx (x = 0-4)
Node x Control Register **(200_H+x*100_H)** **Reset Value: 0001_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0							SUS EN	CAL M	CCE	0	CAN DIS	ALIE	LECI E	TRIE	INIT
r							rw	rw	rw	r	rw	rw	rw	rw	rwh

Field	Bits	Type	Description
INIT	0	rwh	**Node Initialization** 0 Resetting bit INIT enables the participation of the node in the CAN traffic. If the CAN node is in the bus off state then the ongoing bus off recovery (which does not depend on the INIT bit) is continued. With the end of the bus off recovery sequence the CAN node is allowed to take part in the CAN traffic. If the CAN node is not in the bus off state a sequence of 11 consecutive recessive bits must be detected before the node is allowed to take part in the CAN traffic. 1Setting this bit terminates the participation of this node in the CAN traffic. Any ongoing frame transfer is cancelled and the transmit line goes recessive. If the CAN node is in the bus off state then the running bus off recovery sequence is continued. If the INIT bit is still set after the successful completion of the bus off recovery sequence, i.e. after detecting 128 sequences of 11 consecutive recessive bits (11 × 1) then the CAN node leaves the bus off state but remains inactive as long as INIT remains set. Bit INIT is automatically set when the CAN node becomes ‘bus off’ (see **Page 20-12**).
TRIE	1	rw	**Transfer Interrupt Enable** If this bit is set, then an interrupt request is generated upon the successful reception or transmission of a CAN frame. The interrupt output line is selected by TRINP in the CAN Node Interrupt Pointer Register.

Field	Bits	Type	Description
LECIE	2	rw	**LEC indicated Error Interrupt Enable** If this bit is set, then an interrupt request is generated upon each update of the LEC field in the Node Status Register leading to LEC > 0 (CAN protocol error). The interrupt output line is selected by LECINP in the CAN Node Interrupt Pointer Register.
ALIE	3	rw	**Alert Interrupt Enable** If this bit is set then an alert interrupt is generated on one of the following events: 1) A change of bit BOFF in the CAN Node Status Register. 2) A change of bit EWRN in the CAN Node Status Register. 3) A List Length Error, which also sets bit LLE in the CAN Node Status Register. 4) A List Object Error, which also sets bit LOE in the CAN Node Status Register. 5) Bit INIT has been set by the MultiCAN. The interrupt is requested on the interrupt output line selected by ALINP in the CAN Node Interrupt Pointer Register.
CANDIS	4	rw	**CAN Disable** Setting this bit disables the CAN node. The CAN node first waits until it is BUS IDLE or BUS OFF. Then bit INIT is automatically set and an alert interrupt is generated if bit ALIE is set.
CCE	6	rw	**Configuration Change Enable** 0 The Bit Timing Register, the Port Control Register and the Error Counter Register may only be read. All attempts to modify them are ignored. 1 The Bit Timing Register, the Port Control Register and the Error Counter Register may be read and written.

Field	Bits	Type	Description
CALM	7	rw	**Can Analyze Mode** If this bit is set then the CAN node operates in analyze mode. This means that messages may be received, but not transmitted. No acknowledge is sent on the CAN bus upon frame reception. Active error flags are sent recessive instead of dominant. The transmit line is continuously held at recessive (1) level. Bit CALM can be written only while bit INIT is set.
SUSEN	8	rw	**Suspend Enable** This bit allows to set the CAN node into suspend mode via OCDS (on chip debug support): 0 An OCDS suspend trigger is ignored by the CAN node. 1 An OCDS suspend trigger disables the CAN node: As soon as the CAN node becomes BUS IDLE or BUS OFF bit INIT is internally forced to '1' to disable the CAN node. The actual value of bit INIT remains unchanged. Bit SUSEN is reset via OCDS Reset.
0	5, [15:9]	r	**Reserved;** read as 0; should be written with 0.

The Node Status Register reports errors as well as successfully transferred CAN frames.

NSRx (x = 0-4)
Node x Status Register **(204_H+x*100_H)** **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0					SUSACK	LOE	LLE	BOFF	EWRN	ALERT	RXOK	TXOK	LEC		
r					rh	rwh	rwh	rh	rh	rwh	rwh	rwh	rwh		

Field	Bits	Type	Description
LEC	[2:0]	rwh	**Last Error Code** The encoding of this bit field is detailed in **Table 20-10**.
TXOK	3	rwh	**Message Transmitted Successfully** 0 No successful transmission since last flag reset. 1 A message has been transmitted successfully (error free and acknowledged by at least another node). TXOK must be reset by software (write 0). Writing 1 has no effect.
RXOK	4	rwh	**Message Received Successfully** 0 No successful reception since last flag reset. 1 A message has been received successfully. RXOK must be reset by software (write 0). Writing 1 has no effect.

Field	Bits	Type	Description
ALERT	5	rwh	**Alert Warning** The ALERT bit is set upon the occurrence of one of the following events (the same events which also trigger an alert interrupt if ALIE is set): 1) A change of bit BOFF in the CAN Node Status Register. 2) A change of bit EWRN in the CAN Node Status Register. 3) A List Length Error, which also sets bit LLE in the CAN Node Status Register. 4) A List Object Error, which also sets bit LOE in the CAN Node Status Register. 5) Bit INIT has been set by the MultiCAN. ALERT must be reset by software (write 0). Writing 1 has no effect.
EWRN	6	rh	**Error Warning Status** 0 No warning limit exceeded. 1 One of the error counters REC or TEC reached the warning limit EWRNLVL.
BOFF	7	rh	**Bus-off Status** 0CAN controller is not in the bus-off state. 1CAN controller is in the bus-off state.
LLE	8	rwh	**List Length Error** 0 No list length error since last flag reset. 1 A list length error has been detected during message acceptance filtering. The number of elements in the list that belongs to this CAN node differs from the list SIZE given in the list termination pointer. LLE must be reset by software (write 0). Writing 1 has no effect.
LOE	9	rwh	**List Object Error** 0 No list object error since last flag reset. 1 A list object error has been detected during message acceptance filtering. A message object with wrong LIST index entry in the Message Object Control Register has been detected. LOE must be reset by software (write 0). Writing 1 has no effect.

Field	Bits	Type	Description
SUSACK	10	rh	**Suspend Acknowledge** 0 The CAN node is not in suspend mode or a suspend request is pending, but the CAN node has not yet reached BUS IDLE or BUS OFF. 1 The CAN node is in suspend mode: The CAN node is inactive (bit NCR.INIT internally forced to ‘1’) due to an OCDS suspend request.
0	[15:11]	r	**Reserved;** read as 0; should be written with 0.

Encoding of the LEC Bitfield

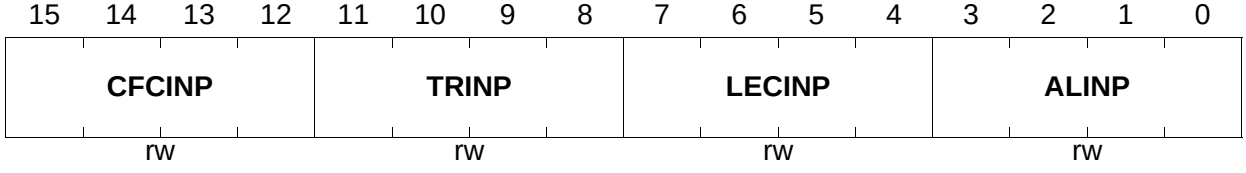
Table 20-10 Encoding of the LEC Bit Field

LEC Value	Signification
000_B	No Error: No error was detected for the last message on the CAN bus.
001_B	Stuff Error: More than 5 equal bits in a sequence have occurred in a part of a received message where this is not allowed.
010_B	Form Error: A ‘fixed format part’ of a received frame has the wrong format.
011_B	Ack Error: The transmitted message was not acknowledged by another node.
100_B	Bit1 Error: During a message transmission the CAN node tried to send a recessive level (1) outside the arbitration field and the acknowledge slot, but the monitored bus value was dominant.
101_B	Bit0 Error: Two different conditions are signalled by this code: a) During transmission of a message (or acknowledge bit, active error flag, overload flag) the CAN node tried to send a dominant level (0), but the monitored bus value was recessive. b) During bus-off recovery this code is set each time a sequence of 11 recessive bits has been monitored. The CPU may use this code as indication that the bus is not continuously disturbed.
110_B	CRC Error: The CRC checksum of the received message was incorrect.
111_B	CPU write to LEC: Whenever the the CPU writes the value 111 to LEC, it takes the value 111. Whenever the CPU writes another value to LEC, the written LEC value is ignored.

The Node Interrupt Pointer Register connects each interrupt request source of the CAN node to one of the up to 16 available interrupt output lines.

NIPRx (x = 0-4)

Node x Interrupt Pointer Register (208_H+x*100_H) **Reset Value: 0000_H**

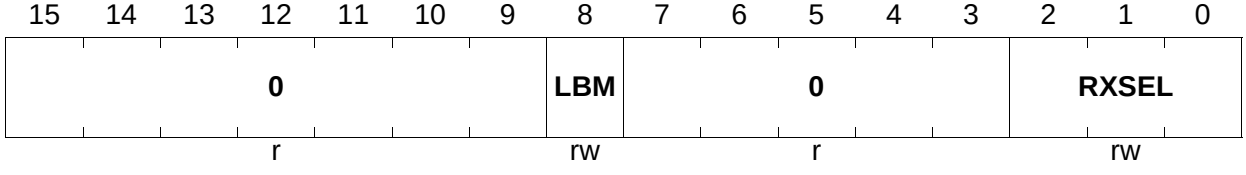


Field	Bits	Type	Description
ALINP	[3:0]	rw	**Alert Interrupt Node Pointer** Number of interrupt output line INT_Om (m=0-15) reporting the “Alert Interrupt Request”, if enabled by ALIE = 1.
LECINP	[7:4]	rw	**Last Error Code Interrupt Node Pointer** Number of interrupt output line INT_Om (m=0-15) reporting the “Last Error Interrupt Request”, if enabled by LECIE = 1.
TRINP	[11:8]	rw	**Transfer OK Interrupt Node Pointer** Number of interrupt output line INT_Om (m=0-15) reporting the “Transfer Interrupt Request”, if enabled by TRIE.
CFCINP	[15:12]	rw	**Frame Counter Interrupt Node Pointer** Number of interrupt output line INT_Om (m=0-15) reporting the “Frame Counter Overflow Interrupt Request”, if enabled by CFCIE = 1.

The Node Port Control Register configures the CAN bus transmit/receive ports. NPCRx may be written only if bit NCRx.CCE is set.

NPCRx (x = 0-4)

Node x Port Control Register **($20C_H$+x*100_H)** **Reset Value: 0000_H**

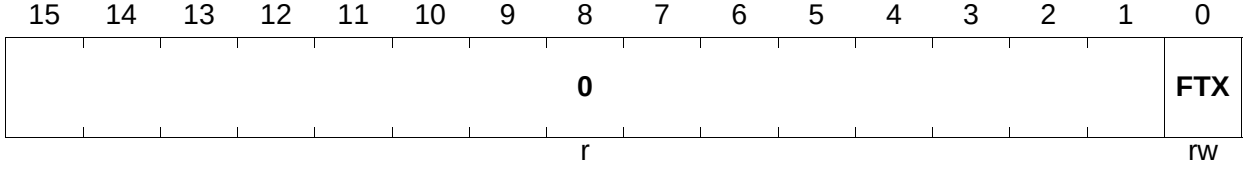


Field	Bits	Type	Description
RXSEL	[2:0]	rw	**Receive Select** RXSEL selects one out of 8 possible receive inputs. CAN traffic is performed through the selected input. The other inputs are ignored. See also “Receive Input Selection” Section
LBM	8	rw	**Loop Back Mode** 0 Loop back mode is disabled. 1 Loop back mode is enabled. This node is connected to an internal (virtual) loop back CAN bus. All CAN nodes which are in loop back mode are connected to this virtual CAN bus so that they can communicate with each other internally. The external transmit line is forced recessive in loop back mode.
0	[7:3], [15:9]	r	**Reserved;** read as 0; should be written with 0.

The Node Bit Timing Register contains all parameters to setup the bit timing for the CAN transfer. NBTRx may be written only if bit NCRx.CCE is set.

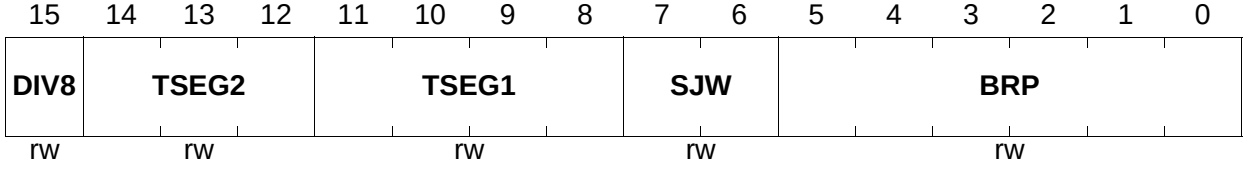
NBTRxH (x = 0-4)

Node x Bit Timing Register High (212_H+x*100_H) **Reset Value: 0000_H**



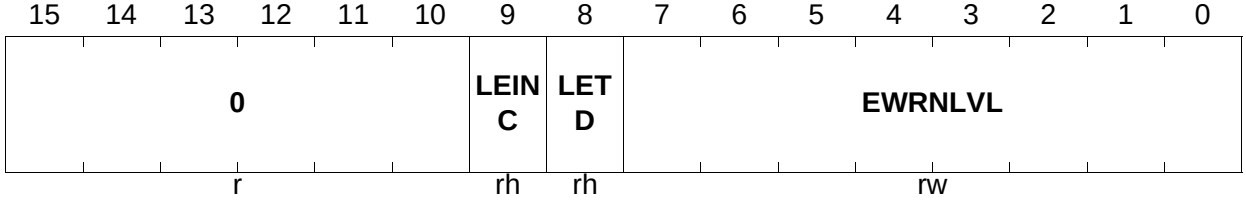
Field	Bits	Type	Description
FTX	0	rw	**Fast Transmit (TTC only)** When a message is requested for transmission on the CAN bus, then the start of frame (SOF) symbol is sent with the beginning of a new bit time. If the CAN bus is in the idle state and bit FTX is set (FTX = 1) then a new bit time is started immediately with the transmit trigger of a new message. This eliminates the variable delay between the transmit trigger of a message and the actual SOF signal on the transmit output. Such a variable delay occurs when transmit triggers occur at different positions within a CAN bit time.
0	[15:1]	r	**reserved;** returns ’0’ if read; should be written with ’0’;

NBTRxL (x = 0-4)
Node x Bit Timing Register Low (210_H+x*100_H) **Reset Value: 0000_H**



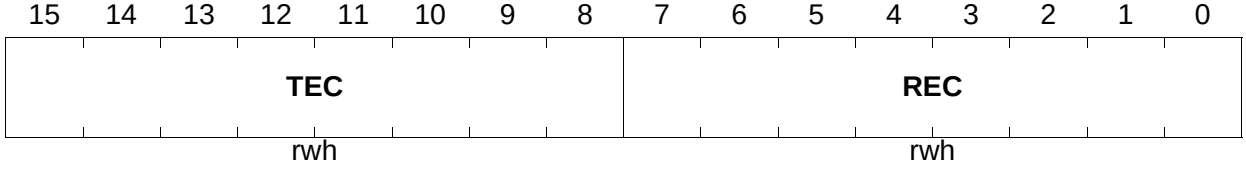
Field	Bits	Type	Description
BRP	[5:0]	rw	**Baud Rate Prescaler** The duration of one time quantum is given by (BRP + 1) clock cycles if DIV8 = 0. The duration of one time quantum is given by 8 × (BRP + 1) clock cycles if DIV8 = 1.
SJW	[7:6]	rw	**(Re)Synchronization Jump Width** (SJW + 1) time quanta are allowed for resynchronization.
TSEG1	[11:8]	rw	**Time Segment Before Sample Point** (TSEG1 + 1) time quanta is the user defined nominal time between the end of the synchronization segment and the sample point. It includes the propagation segment, which takes into account signal propagation delays. The time segment may be lengthened due to resynchronization. Valid values for TSEG1 are 2 to 15.
TSEG2	[14:12]	rw	**Time Segment After Sample Point** (TSEG2 + 1) time quanta is the user defined nominal time between the sample point and the start of the next synchronization segment. It may be shortened due to resynchronization. Valid values for TSEG2 are 1 to 7.
DIV8	15	rw	**Divide Prescaler Clock by 8** 0 A time quantum lasts (BRP+1) clock cycles. 1 A time quantum lasts 8 × (BRP+1) clock cycles.

NECNTxH (x = 0-4)
Node x Error Counter Register High(216_H+x*100_H) **Reset Value: 0060_H**



Field	Bits	Type	Description
EWRNLVL	[7:0]	rw	**Error Warning Level** Bit field EWRNLVL defines the threshold value (warning level, default 96) to be reached in order to set the corresponding error warning bit EWRN.
LETD	8	rh	**Last Error Transfer Direction** 0 The last error occurred while the CAN node was receiver (REC has been incremented). 1 The last error occurred while the CAN node was transmitter (TEC has been incremented).
LEINC	9	rh	**Last Error Increment** 0The last error led to an error counter increment of 1. 1The last error led to an error counter increment of 8.
0	[15:10]	r	**Reserved;** read as 0; should be written with 0.

NECNTxL (x = 0-4)
Node x Error Counter Register Low(214_H+x*100_H) **Reset Value: 0000_H**



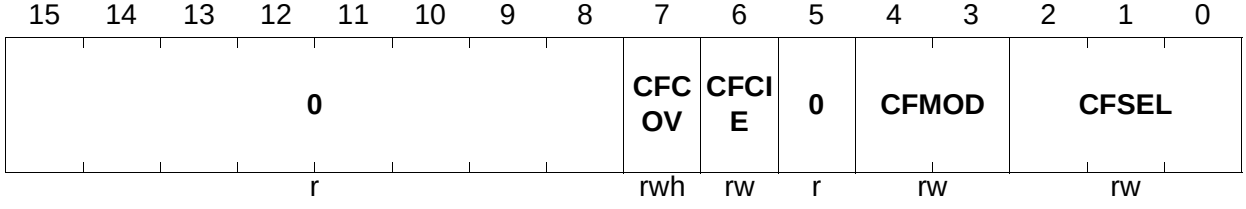
Field	Bits	Type	Description
REC	[7:0]	rwh	**Receive Error Counter** Bit field REC contains the value of the receive error counter of the CAN node.

Field	Bits	Type	Description
TEC	[15:8]	rwh	**Transmit Error Counter** Bit field TEC contains the value of the transmit error counter of the CAN node.

The Node Frame Counter Register contains the actual value of the frame counter as well as control and status bits of the frame counter.

NFCRxH (x = 0-4)

Node x Frame Counter Register High($21A_H$+x*100_H) **Reset Value: 0000_H**

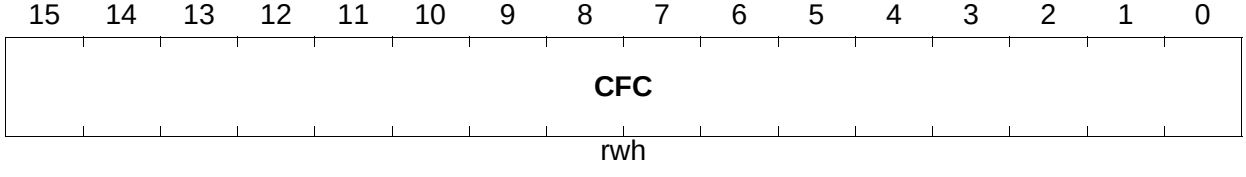


Field	Bits	Type	Description
CFSEL	[2:0]	rw	**CAN Frame Count Selection** This bit field selects the function of the frame counter for the chosen frame count mode. **Frame Count Mode** Bit 0 If Bit 0 of CFSEL is set then CFC is incremented each time a foreign frame (i.e. a frame not matching to a message object) has been received on the CAN bus. Bit 1 If Bit 1 of CFSEL is set then CFC is incremented each time a frame matching to a message object has been received on the CAN bus. Bit 2 If Bit 2 of CFSEL is set then CFC is incremented each time a frame has been transmitted successfully by the node. **Time Stamp Mode** The frame counter is incremented (internally) with the beginning of a new bit time. Its value is permanently sampled in the CFC field while the bus is idle. The value sampled just before the SOF bit of a new frame is detected is written to the corresponding message object. When the treatment of a message object is finished, the sampling continues. **Bit Timing Mode** The available bit timing measurement modes are shown in **Table 20-11**. If CFCIE is set then an interrupt on request node x (where x is the CAN node index) is generated with a CFC update.

Field	Bits	Type	Description
CFMOD	[4:3]	rw	**CAN Frame Counter Mode** This bit field defines the operation mode of the frame counter. 00_B Frame Count Mode: The frame counter is incremented upon the reception and transmission of frames. 01_B Time Stamp Mode: The frame counter is used to count CAN bit times. 10_B Bit Timing Mode: The frame counter is used for analysis of the bit timing.[1)] 11_B reserved
CFCIE	6	rw	**CAN Frame Count Interrupt Enable** 0 CAN Frame Counter Overflow interrupt request is disabled. 1 CAN Frame Counter Overflow interrupt request is enabled.
CFCOV	7	rwh	**CAN Frame Counter Overflow Flag** Flag CFCOV is set upon a frame counter overflow (transition from $FFFF_H$ to 0000_H). In bit timing analysis mode CFCOV is set upon an update of CFC. An interrupt request is generated if CFCIE = 1. 0 No overflow has occurred since last flag reset. 1 An overflow has occurred since last flag reset. CFCOV must be cleared by software.
0	5, [15:8]	r	**reserved;** returns ’0’ if read; should be written with ’0’;

1) For all bit timing analysis modes, the count value of NFCRx.CFC always displays the measured value minus 1. Example: A CFC value of 34 in mode CFSEL = 000 indicates that 35 have been elapsed between the most recent 2 dominat edges on the receive input.

NFCRxL (x = 0-4)
Node x Frame Counter Register Low(218_H+x*100_H) **Reset Value: 0000_H**



Field	Bits	Type	Description
CFC	[15:0]	rwh	**CAN Frame Counter** In Frame Count Mode this bit field contains the frame count value. In TimeStamp Mode this bit field contains the captured bit time count value, captured with the start of a new frame.

Bit Timings Analysis Modes and States

Table 20-11 Bit Timing Analysis Modes (CFMOD = 10)

CFSEL	Measurement
000	Whenever a dominant edge (transition from 1 to 0) is monitored on the receive input the time (measured in clock cycles) between this edge and the most recent dominant edge is stored in CFC.
001	Whenever a recessive edge (transition from 0 to 1) is monitored on the receive input the time (measured in clock cycles) between this edge and the most recent dominant edge is stored in CFC.
010	Whenever a dominant edge is received as a result of a transmitted dominant edge the time (clock cycles) between both edges is stored in CFC.
011	Whenever a recessive edge is received as a result of a transmitted recessive edge the time (clock cycles) between both edges is stored in CFC.
100	Whenever a dominant edge that qualifies for synchronization is monitored on the receive input the time (measured in clock cycles) between this edge and the most recent sample point is stored in CFC.
101	With each sample point, the time (measured in clock cycles) between the start of the new bit time and the start of the previous bit time is stored in CFC[11:0]. Additional information is written to CFC[15:12] at each sample point: CFC[15] : Transmit value of actual bit time CFC[14] : Receive sample value of actual bit time CFC[13:12] : CAN bus information (see **Table 20-12**)
110	reserved
111	reserved

Table 20-12 CAN Bus State Information

CFC[13:12]	CAN bus state
00	**NoBit** The CAN bus is idle, performs bit (de-) stuffing or is in one of the following frame segments: SOF, SRR, CRC, delimiters, first 6 EOF bits, IFS
01	**NewBit** This code represents the first bit of a new frame segment. The current bit is the first bit in one of the following frame segments: bit 10 (MSB) of standard ID (transmit only), RTR, reserved bits, IDE, DLC(MSB), bit 7 (MSB) in each data byte and the first bit of the ID extension
10	**Bit** This code represents a bit inside a frame segment with a length of more than one bit (not the first bit of those frame segments which is indicated by NewBit). The current bit is processed within one of the following frame segments: ID bits (except first bit of standard ID for transmission and first bit of ID extension), DLC (3 LSB) and bits 6-0 in each data byte
11	**Done** The current bit is in one of the following frame segments: Acknowledge slot, last bit of EOF, active/passive error frame, overload frame. Two or more directly consecutive Done codes signal an error frame.

20.2.12 Message Object Registers

The Message Object Control Register contains control bits for the CAN transfer and the message object link pointer. Each control bit has a corresponding bit in the CTRL field. A control bit is set by writing 1 to the corresponding bit in CTRL. It is cleared by writing 1 to the control bit directly. Any other combination leaves the control bit unchanged . After reset initialization the pointer PNEXT (read value of MOCTRnH[15:8]) points to message object n+1 (PNEXT = n+1), except for PNEXT of message object 127, which terminates the initial list (PNEXT = 127). Pointer PREV (read value of MOCTRnH[7:0]) initially points to message object n-1 (PPREV = n-1), except for PPREV of message object 0 which indicates the start of the initial list (PPREV = 0). This reset initialization means that all message objects initially belong to the list of unallocated elements.

MOCTRnH (n = 0-127)
Message Object n Control Register High ($101E_H$+n*20_H) **Reset Value:shiftl8(n+1)+(n-1)[1)]**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				SETDIR	SETTXEN1	SETTXEN0	SETTXRQ	SETRXEN	SETRTSEL	SETMSGVAL	SETMSGLST	SETNEWDAT	SETRXUPD	SETTXPND	SETRXPND
w				w	w	w	w	w	w	w	w	w	w	w	w

1) Exceptions: Message Obj. 0 : Reset val. of PPREV = 0, Message Obj. 127 : Reset val. of PNEXT = 127

Field	Bits	Type	Description
SETRXPND	0	w	**Set Receive Pending** This bit sets the RXPND
SETTXPND	1	w	**Set Transmit Pending** This bit sets the TXPND
SETRXUPD	2	w	**Set Receive Updating** This bit sets the RXUPD
SETNEWDAT	3	w	**Set New Data** This bit sets the NEWDAT
SETMSGLST	4	w	**Set Message Lost** This bit sets the MSGLST
SETMSGVAL	5	w	**Set Message Valid** This bit sets the MSGVAL
SETRTSEL	6	w	**Set Receive/Transmit Selected** This bit sets the RTSEL
SETRXEN	7	w	**Set Receive Enable** This bit sets the RXEN
SETTXRQ	8	w	**Set Transmit Request** This bit sets the TXRQ
SETTXEN0	9	w	**Set Transmit Enable 0** This bit sets the TXEN0
SETTXEN1	10	w	**Set Transmit Enable 1** This bit sets the TXEN1
SETDIR	11	w	**Set Message Direction** This bit sets the DIR

Field	Bits	Type	Description
0	[15:12]	w	**Reserved** Should be written with 0.

MOCTRnL (n = 0-127)
Message Object n Control Register Low($101C_H$+n*20_H) **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				RESDIR	RESTXEN1	RESTXEN0	RESTXRQ	RESRXEN	RESRTSEL	RESMSGVAL	RESMSGLST	RESNEWDAT	RESRXUPD	RESTXPND	RESRXPND
w				w	w	w	w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
RESRXPND	0	w	**Reset Receive Pending** This bit resets the RXPND
RESTXPND	1	w	**Reset Transmit Pending** This bit resets the TXPND
RESRXUPD	2	w	**Reset Receive Updating** This bit resets the RXUPD
RESNEWDAT	3	w	**Reset New Data** This bit resets the NEWDAT
RESMSGLST	4	w	**Reset Message Lost** This bit resets the MSGLST
RESMSGVAL	5	w	**Reset Message Valid** This bit resets the MSGVAL
RESRTSEL	6	w	**Reset Receive/Transmit Selected** This bit resets the RTSEL
RESRXEN	7	w	**Reset Receive Enable** This bit resets the RXEN
RESTXRQ	8	w	**Reset Transmit Request** This bit resets the TXRQ
RESTXEN0	9	w	**Reset Transmit Enable 0** This bit resets the TXEN0
RESTXEN1	10	w	**Reset Transmit Enable 1** This bit resets the TXEN1

Field	Bits	Type	Description
RESDIR	11	w	**Reset Message Direction** This bit resets the DIR
0	[15:12]	w	**Reserved** Should be written with 0.

MOSTATnH (n = 0-127)
Message Object n Status Register High ($101E_H$+n*20_H) **Reset Value:shiftl8(n+1)+(n-1)[1)]**

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

1) Exceptions: Message Obj. 0 : Reset val. of PPREV = 0, Message Obj. 127 : Reset val. of PNEXT = 127

Field	Bits	Type	Description
PNEXT	[15:8]	rh	**Pointer to Previous Message Object** PPREV holds the message object number of the previous message object in a message list structure.
PPREV	[7:0]	rh	**Pointer to Next Message Object** PNEXT holds the message object number of the next message object in a message list structure.

MOSTATnL (n = 0-127)
Message Object n Status Register Low($101C_H$+n*20_H) **Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LIST				**DIR**	**TX EN1**	**TX EN0**	**TX RQ**	**RX EN**	**RT SEL**	**MSG VAL**	**MSG LST**	**NEW DAT**	**RX UPD**	**TX PND**	**RX PND**
rh				rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
RXPND	0	rh	**Receive Pending** 0_B No CAN message has been received. 1_B A CAN message has been received by the message object n, either directly or via gateway copy action. RXPND is not reset by hardware but must be reset by software.
TXPND	1	rh	**Transmit Pending** 0_B No CAN message has been transmitted. 1_B A CAN message from message object n has been transmitted successfully over the CAN bus. TXPND is reset by hardware but must be reset by software.
RXUPD	2	rh	**Receive Updating** 0_B No receive update ongoing. 1_B Message identifier, DLC, and data of the message object are currently updated.
NEWDAT	3	rh	**New Data** 0_B No update of the message object n since last flag reset. 1_B Message object n has been updated. NEWDAT is set by hardware after a received CAN frame has been stored in message object n. NEWDAT is cleared by hardware when a CAN transmission of message object n has been started. NEWDAT should be set by software after the new transmit data has been stored in message object n to prevent the automatic reset of TXRQ at the end of an ongoing transmission.
MSGLST	4	rh	**Message Lost** 0_B No CAN message is lost. 1_B A CAN message is lost because NEWDAT has become set again when it has already been set.

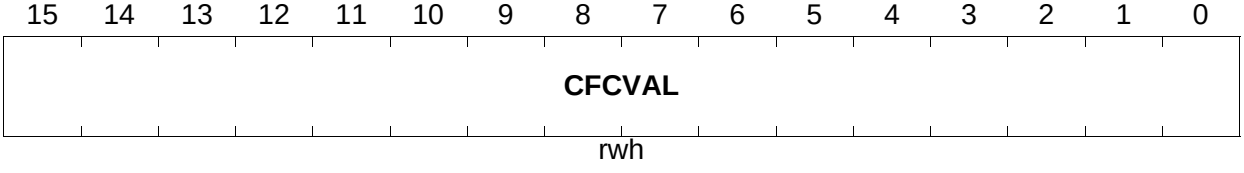
Field	Bits	Type	Description
MSGVAL	5	rh	**Message Valid** 0_B Message object n is not valid. 1_B Message object n is valid. Only a valid message object takes part in CAN transfers.
RTSEL	6	rh	**Receive/Transmit Selected** 0_B Message object n is not selected for receive or transmit operation. 1_B Message object n is selected for receive or transmit operation. **Frame Reception:** RTSEL is set by hardware when message object n has been identified for storage of a CAN frame that is currently received. Before a received frame becomes finally stored in message object n, a check is performed to determine if RTSEL is set. Thus the CPU can suppress a scheduled frame delivery to this message object n by clearing RTSEL by software. **Frame Transmission:** RTSEL is set by hardware when message object n has been identified to be transmitted next. A check is performed to determine if RTSEL is still set before message object n is actually set up for transmission and bit NEWDAT is cleared. It is also checked that RTSEL is still set before its message object n is verified due to the successful transmission of a frame. RTSEL needs to be checked only when the context of message object n changes, and a conflict with an ongoing frame transfer shall be avoided. In all other cases, RTSEL can be ignored. RTSEL has no impact on message acceptance filtering. RTSEL is not cleared by hardware.
RXEN	7	rh	**Receive Enable** 0_B Message object n is not enabled for frame reception. 1_B Message object n is enabled for frame reception. RXEN is evaluated for receive acceptance filtering only.

Field	Bits	Type	Description
TXRQ	8	rh	**Transmit Request** 0_B No transmission of message object n is requested. 1_B Transmission of message object n on the CAN bus is requested. The transmit request becomes valid only if TXRQ, TXEN0, TXEN1 and MSGVAL are set. TXRQ is set by hardware if a matching Remote Frame has been received correctly. TXRQ is reset by hardware if message object n has been transmitted successfully and NEWDAT is not set again by software.
TXEN0	9	rh	**Transmit Enable 0** 0_B Message object n is not enabled for frame transmission. 1_B Message object n is enabled for frame transmission. Message object n can be transmitted only if both bits, TXEN0 and TXEN1, are set. The user may clear TXEN0 in order to inhibit the transmission of a message that is currently updated, or to disable automatic response of Remote Frames.
TXEN1	10	rh	**Transmit Enable 1** 0_B Message object n is not enabled for frame transmission. 1_B Message object n is enabled for frame transmission. Message object n can be transmitted only if both bits, TXEN0 and TXEN1, are set. TXEN1 is used by the MultiCAN module for selecting the active message object in the Transmit FIFOs.

Field	Bits	Type	Description
DIR	11	rh	**Message Direction** 0_B Receive Object selected: With TXRQ = 1, a Remote Frame with the identifier of message object n is scheduled for transmission. On reception of a Data Frame with matching identifier, the message is stored in message object n. 1_B Transmit Object selected: If TXRQ = 1, message object n is scheduled for transmission of a Data Frame. On reception of a Remote Frame with matching identifier, bit TXRQ is set.
LIST	[15:12]	rh	**List Allocation** LIST indicates the number of the message list to which message object n is allocated. LIST is updated by hardware when the list allocation of the object is modified by a panel command.

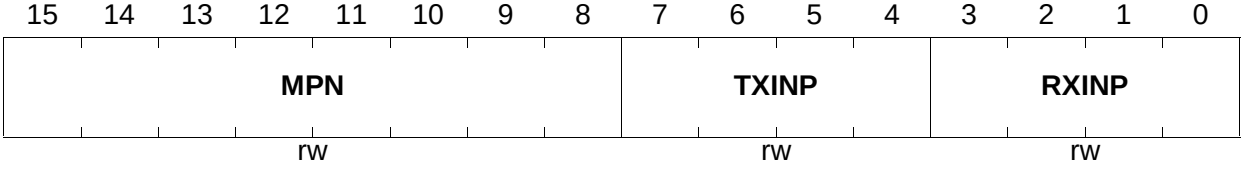
The Message Object Interrupt Pointer Registers MOIPR H/L hold various pointers related to message interrupts as well as the frame counter value.

MOIPRnH (n = 0-127)
Message Object n Interrupt Pointer Register High
($100A_H$+n*20_H) **Reset Value: 0000_H**



Field	Bits	Type	Description
CFCVAL	[15:0]	rwh	**CAN Frame Counter Value** When a message is stored in this message object or this message object has been successfully transmitted then the CAN frame counter value CFC of the CAN Node Frame Counter Register (NFCR) is copied to CFCVAL.

MOIPRnL (n = 0-127)
Message Object n Interrupt Pointer Register Low
(1008_H+n*20_H) **Reset Value: 0000_H**



Field	Bits	Type	Description
RXINP	[3:0]	rw	**Receive Interrupt Node Pointer** Select the interrupt output line INT_Om (m=0-15) for receive interrupts.
TXINP	[7:4]	rw	**Transmit Interrupt Node Pointer** Select the interrupt output line INT_Om (m=0-15) for transmit interrupts.

Field	Bits	Type	Description
MPN	[15:8]	rw	**Message Pending Number** This field selects the bit position of the bit in the message pending register to be set upon a receive/transmit interrupt.

The Message Object Function Control Registers High / Low contain bits to select and to configure the function of the message object. It also holds the CAN data length code.

MOFCRnH (n = 0-127)
Message Object n Function Control Register High
(1002_H+n*20_H) **Reset Value: 0000_H**

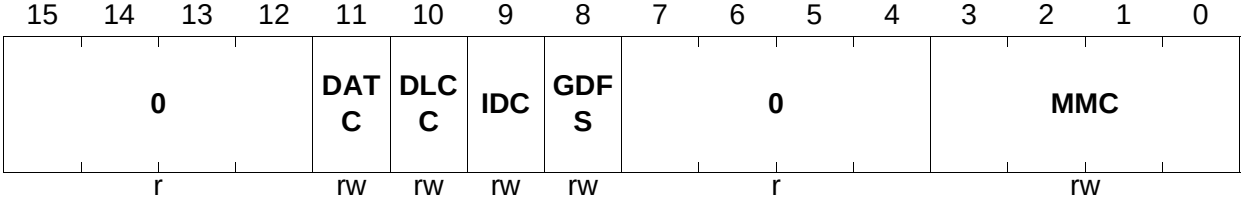
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0MSC				DLC				STT	SDT	RMM	FRR EN	0	OVIE	TXIE	RXIE
rwh				rwh				rw	rw	rw	rw	r	rw	rw	rw

Field	Bits	Type	Description
RXIE	0	rw	**Receive Interrupt Enable** If RXIE is set then a message interrupt request is generated with the reception of a CAN message, no matter whether the CAN message is received directly or indirectly via a gateway action. The interrupt is requested on interrupt output line as defined by RXINP.
TXIE	1	rw	**Transmit Interrupt Enable** If TXIE is set then a message interrupt request is generated when this message object successfully transmitted a message over the CAN bus. The interrupt is requested on interrupt output line as defined by TXINP.
OVIE	2	rw	**Overflow Interrupt Enable** IF OVIE = 1 then a FIFO full interrupt is generated when the pointer to the current object CUR reaches the value of SEL in the FIFO/Gateway Pointer Register. If this object is a receive FIFO base object then the FIFO full interrupt is requested on interrupt ouput line as defined by TXINP. If this object is a transmit FIFO base object then the FIFO full interrupt is requested on interrupt output line as defined by RXINP. For all other message object modes OVIE has no effect.

Field	Bits	Type	Description
FRREN	4	rw	**Foreign Remote Request Enable** Specifies if the TXRQ bit is set in this message object or in a foreign object referenced by the pointer CUR. 0 TXRQ of this message object is set upon the reception of a matching remote frame. 1TXRQ of the message object referenced by the pointer CUR is set upon the reception of a matching remote frame.
RMM	5	rw	**Transmit Object Remote Monitoring** 0 Remote monitoring disabled: The identifier, IDE bit and DLC of the message object remain unchanged upon the reception of a matching remote frame. 1 Remote monitoring enabled: The identifier, DLC and IDE bit of a matching remote frame are copied to this transmit object in order to monitor incoming remote frames. Bit RMM only applies to transmit objects and has no impact on receive objects.
SDT	6	rw	**Single Data Transfer** If SDT = 1 and this object is not a FIFO base object then MSGVAL is reset when this object has taken part in a successful data transfer (receive or transmit). If SDT = 1 and this object is a FIFO base object then MSGVAL is reset when the pointer to the current object CUR reaches the value of SEL in the FIFO/Gateway Pointer Register. With SDT = 0, bit MSGVAL is not affected.
STT	7	rw	**Single Transmit Trial** If this bit is set then TXRQ is cleared upon transmission start of this message object. Thus no transmission retry is performed in case of transmission failure.
DLC	[11:8]	rwh	**Data Length Code** Valid values for the data length are 0 to 8. DLC>8 leads to 8 data bytes, but the DLC code is not truncated upon reception or transmission of CAN frames.

Field	Bits	Type	Description
0	[15:12]	r	**Reserved;** read as 0; should be written with 0.
0	3	r	**Reserved;** read as 0; should be written with 0.

MOFCRnL (n = 0-127)
Message Object n Function Control Register Low
(1000_H+n*20_H) **Reset Value: 0000_H**

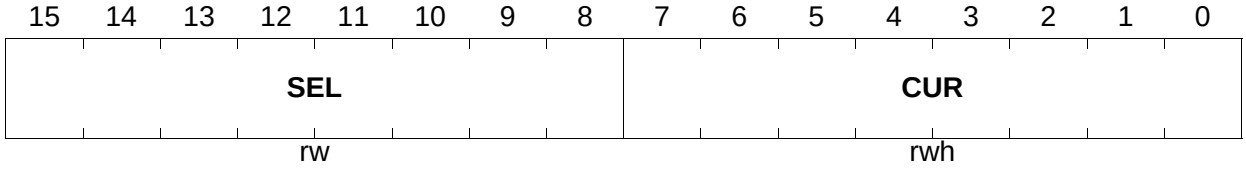


Field	Bits	Type	Description
MMC	[3:0]	rw	**Message Mode Control** Bit field MMC controls the functionality of the message object. 0000 Standard Message Object 0001 Receive FIFO Base Object 0010 Transmit FIFO Base Object 0011 Transmit FIFO Slave Object 0100 Gateway Source Object else Reserved
GDFS	8	rw	**Gateway Data Frame Send** 1 TXRQ is set in the gateway destination object after the transfer of a data frame from the gateway source to the gateway destination. 0 TXRQ is not set in the destination object. Applicable only to Gateway Source Object.
IDC	9	rw	**Identifier Copy** IF IDC = 1 then the identifier of the gateway source object (after storing the received frame in the source) is copied to the gateway destination. Applicable only to Gateway Source Object.

Field	Bits	Type	Description
DLCC	10	rw	**Data Length Code Copy** If DLCC = 1 then the data length code of the gateway source object (after storing the received frame in the source) is copied to the gateway destination. Applicable only to Gateway Source Object.
DATC	11	rw	**Data Copy** If DATC = 1 then the data field (registers MODATA0 and MODATA4) of the gateway source object (after storing the received frame in the source) is copied to the gateway destination. Applicable only to Gateway Source Object.
0	[7:4], [15:12]	r	**reserved;** returns ’0’ if read; should be written with ’0’;

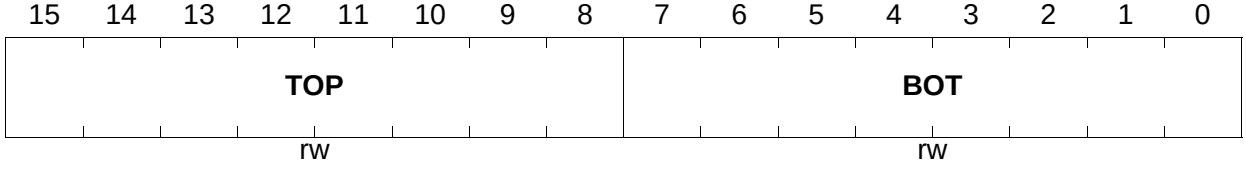
The Message Object FIFO/Gateway Pointer Registers H/L contain a set of message object link pointer used for FIFO and gateway functionality

MOFGPRnH (n = 0-127)
Message Object n FIFO/Gateway Pointer Register High
(1006_H+n*20_H) **Reset Value: 0000_H**



Field	Bits	Type	Description
CUR	[7:0]	rwh	**Current Object Pointer** The Current Object Pointer links to the actual target object within a FIFO/Gateway structure. After a FIFO/gateway operation CUR is updated with the message number of the next message object in the list structure (given by PNEXT of the message control register) until it reaches the FIFO top element (given by TOP) when it is reset to the bottom element (given by BOT).
SEL	[15:8]	rw	**Object Select Pointer** The Object Select Pointer is the second (software) pointer to complement the hardware pointer CUR in the FIFO structure. SEL is used for monitoring purposes only.

MOFGPRnL (n = 0-127)
Message Object n FIFO/Gateway Pointer Register Low
(1004_H+n*20_H) **Reset Value: 0000_H**

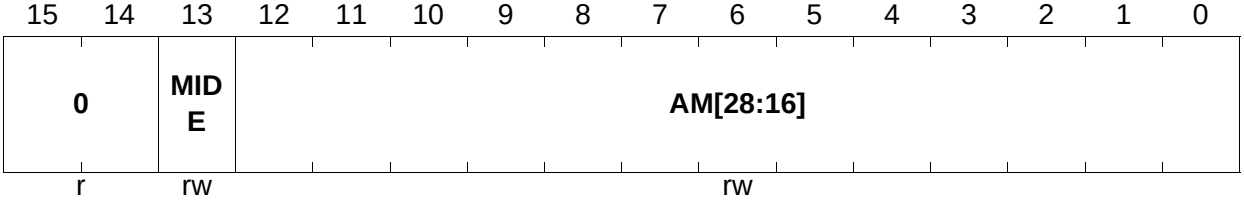


Field	Bits	Type	Description
BOT	[7:0]	rw	**Bottom Pointer** The Bottom Pointer points to the first element in a FIFO structure.
TOP	[15:8]	rw	**Top Pointer** The TOP pointer points to the last element in a FIFO structure.

Note: The pointers in this register must be set to objects assigned to the same CAN node. It is forbidden to refer to objects that are not in the linked list for the same CAN node.

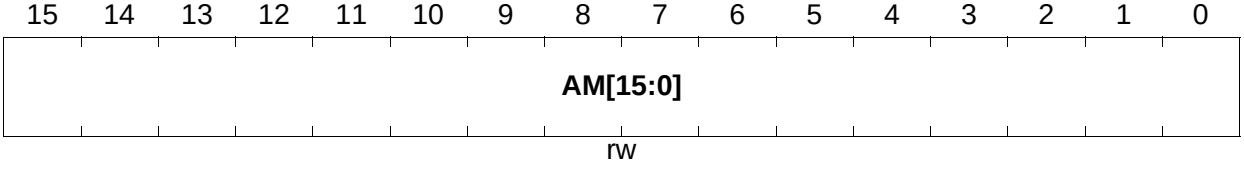
Registers MOAMR H/L contain the mask bits for the acceptance filtering of the mesage object.

MOAMRnH (n = 0-127)
Message Object n Acceptance Mask Register High
($100E_H$+n*20_H) **Reset Value: $3FFF_H$**



Field	Bits	Type	Description
AM[28:16]	[12:0]	rw	**Acceptance Mask for Message Identifier High** see description of MOAMRnL.AM[15:0]
MIDE	13	rw	**Acceptance Mask bit for Message IDE bit** 0 This message objects accepts the reception of both standard and extended frames. 1 This message object only receives frames with matching IDE bit.
0	[15:14]	r	**Reserved;** read as 0; should be written with 0.

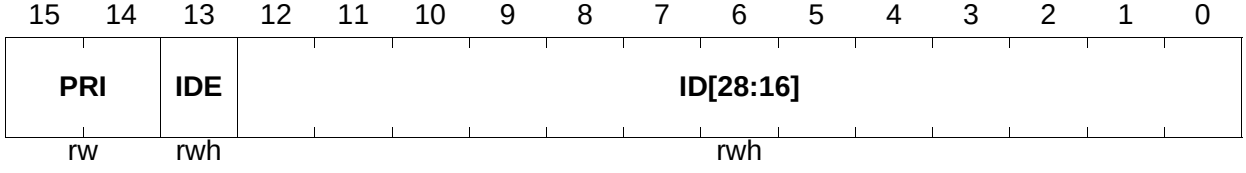
MOAMRnL (n = 0-127)
Message Object n Acceptance Mask Register Low
($100C_H$+n*20_H) **Reset Value: $FFFF_H$**



Field	Bits	Type	Description
AM[15:0]	[15:0]	rw	**Acceptance Mask for Message Identifier** Mask to filter incoming messages with standard identifiers (AM[28:18]) or extended identifiers (AM[28:0]). For standard identifiers bits AM[17:0] are “don’t care”.

Registers MOAR H/L contain the CAN identifier of the message object.

MOARnH (n = 0-127)
Message Object n Arbitration Register High
($101A_H$+n*20_H) **Reset Value: 0000_H**

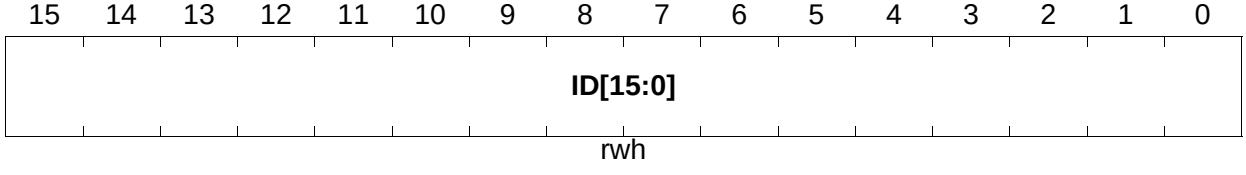


Field	Bits	Type	Description
ID[28:16]	[12:0]	rwh	**CAN Identifier of Message Object** Identifier of a standard message (ID[28:18]) or an extended message (ID[28:0]). For standard identifiers bits ID[17:0] are “don’t care”.
IDE	13	rwh	**CAN IDE bit of Message Object** 0 Standard frame with 11-bit identifier 1Extended frame with 29-bit identifier

Field	Bits	Type	Description
PRI	[15:14]	rw	**Priority Class** PRI assigns one of the four priority classes 0, 1, 2, 3 to the message object, with lower PRI number meaning higher priority. Message objects with lower PRI value always win acceptance filtering for frame reception and transmission over message objects with higher PRI value. Acceptance filtering based on identifier/mask and list position is only performed between message objects of the same priority class. PRI also defines the acceptance filtering method for transmission: 00 Reserved. Transmit objects with PRI = 00 are not taken into account for the transmit acceptance filtering. 01 Transmit acceptance filtering is based on the list order, i.e. this message object is considered for transmission only if there is no other message object with valid transmit request (MSGVAL & TXRQ & TXEN0 & TXEN1 = 1) somewhere before this object in the list. 10Transmit acceptance filtering is based on the CAN identifier, i.e. this message object is considered for transmission only if there is no other message object with higher priority identifier+IDE+DIR (with respect to CAN arbitration rules) somewhere in the list (see **Table 20-13**). 11Transmit acceptance filtering is based on the list order (like PRI = 01).

MOARnL (n = 0-127)
Message Object n Arbitration Register Low

(1018_H+n*20_H) **Reset Value: 0000_H**



Field	Bits	Type	Description
ID[15:0]	[15:0]	rwh	**CAN Identifier of Message Object Low** Identifier of a standard message (ID[28:18]) or an extended message (ID[28:0]). For standard identifiers bits ID[17:0] are “don’t care”.

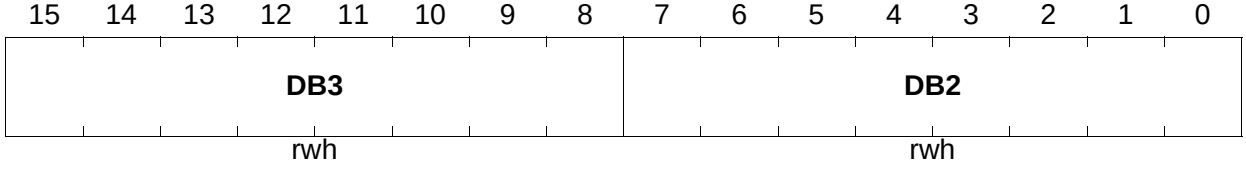
Transmit Priority

Table 20-13 Transmit Priority based on CAN Arbitration Rules

Settings of arbitrarily chosen message objects A and B, where A has higher transmit priority than B	Comment
A.MOAR[28:18] < B.MOAR[28:18] (11 bit standard identifier of A less than 11 bit standard identifier of B)	Messages with lower standard identifier have higher priority than messages with higher standard identifier. MOAR[28] is the most significant bit (MSB) of the standard identifier. MOAR[18] is the least significant bit of the standard identifier.
A.MOAR[28:18] = B.MOAR[28:18] A.MOAR.IDE = 0 (send standard frame) B.MOAR.IDE = 1 (send extended frame)	Standard frames have higher transmit priority than extended frames with equal standard identifier.
A.MOAR[28:18] = B.MOAR[28:18] A.MOAR.IDE = B.MOAR.IDE = 0 A.MOCTR.DIR = 1 (send data frame) B.MOCTR.DIR = 0 (send remote frame)	Standard data frames have higher transmit priority than standard remote frames with equal identifier.
A.MOAR[28:0] = B.MOAR[28:0] A.MOAR.IDE = B.MOAR.IDE = 1 A.MOCTR.DIR = 1 (send data frame) B.MOCTR.DIR = 0 (send remote frame)	Extended data frames have higher transmit priority than extended remote frames with equal identifier.
A.MOAR[28:0] < B.MOAR[28:0] A.MOAR.IDE = B.MOAR.IDE = 1 (29 bit identifier)	Extended frames with lower identifier have higher transmit priority than extended frames with higher identifier. MOAR[28] is the most significant bit (MSB) of the overall identifier (standard identifier MOAR[28:18] and identifier extension MOAR[17:0]). MOAR[0] is the least significant bit (LSB) of the overall identifier.

MODATAnLH (n = 0-127)
Message Object n Data Register Low High

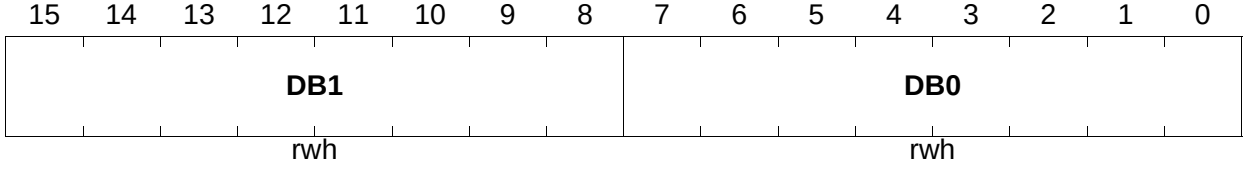
(1012_H+n*20_H) Reset Value: 0000_H



Field	Bits	Type	Description
DB2	[7:0]	rwh	**CAN Data Byte 2**
DB3	[15:8]	rwh	**CAN Data Byte 3**

MODATAnLL (n = 0-127)
Message Object n Data Register Low Low

(1010_H+n*20_H) Reset Value: 0000_H

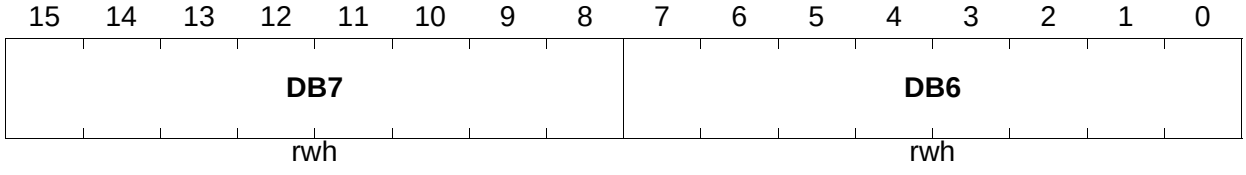


Field	Bits	Type	Description
DB0	[7:0]	rwh	**CAN Data Byte 0**
DB1	[15:8]	rwh	**CAN Data Byte 1**

Registers MODATAH H/L contain the highest four CAN data bytes. Unused data bytes are padded zero upon reception and ignored for transmission .

MODATAnHH (n = 0-127)
Message Object n Data Register High High

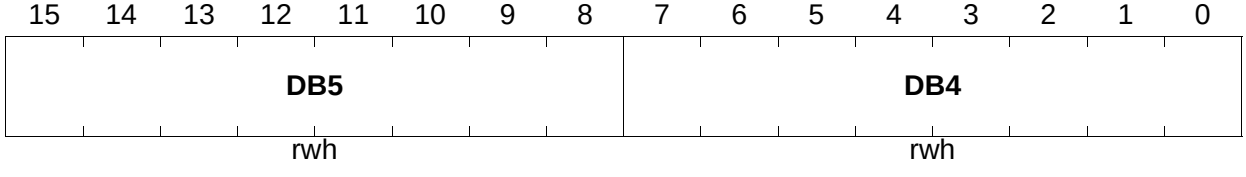
(1016_H+n*20_H) **Reset Value: 0000_H**



Field	Bits	Type	Description
DB6	[7:0]	rwh	**CAN Data Byte 6**
DB7	[15:8]	rwh	**CAN Data Byte 7**

MODATAnHL (n = 0-127)
Message Object n Data Register High Low

(1014_H+n*20_H) **Reset Value: 0000_H**



Field	Bits	Type	Description
DB4	[7:0]	rwh	**CAN Data Byte 4**
DB5	[15:8]	rwh	**CAN Data Byte 5**

20.3 General Control and Status

The following section describes the general clock, debug and interrupt topics.

20.3.1 Clock Control

The CAN clock frequency f_{CAN} of the functional blocks of the MultiCAN module is derived from the system clock f_{SYS} (= clock on the system bus). The fractional divider FDIV in the module is used to generate the CAN clock frequency for the bit timing calculation. This frequency is identical for all CAN nodes. The scheduler itself is in the f_{SYS} domain. The clock generation can be enabled/disabled by the fractional divider control bit field FDR.DM.

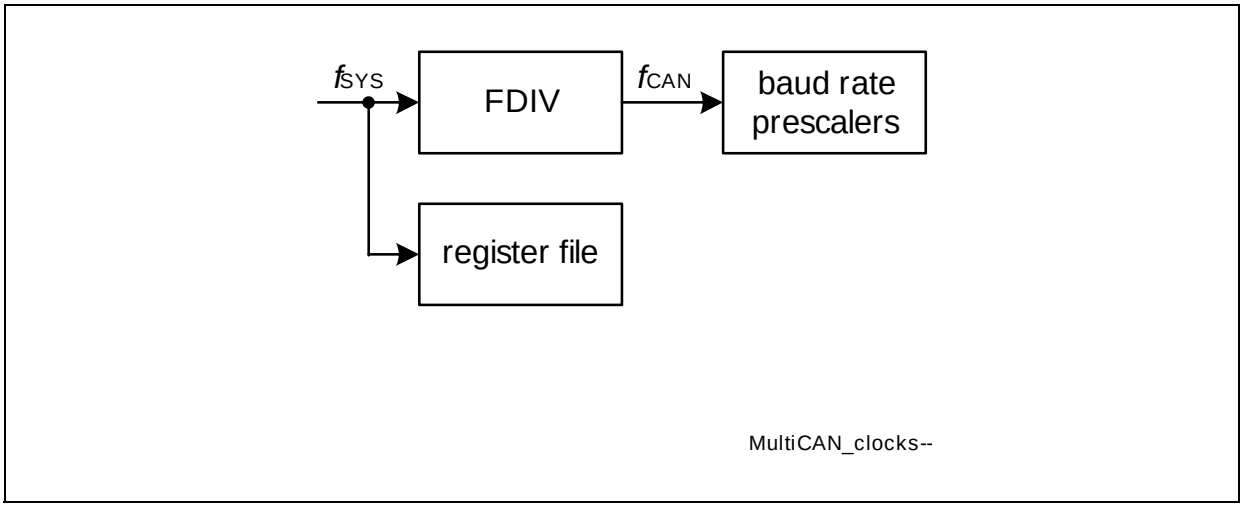


Figure 20-17 MultiCAN Clock Generation

The fractional divider FDIV output f_{CAN} is based on the system clock f_{SYS}, but only every n-th clock pulse is taken. The register file is in the system frequency domain. The suspend signal (coming as acknowledge from the module as answer to the OCDS suspend request) freezes or resets the fractional divider.

Note: The receive input line contains a synchronization stage to ensure stable input data. Together with the internal CAN state machine, this leads to a minimum reaction time of at least 3 clock cycles of f_{SYS} between CAN input and output. The switching delay of the input stages can be generally neglected, whereas the rise/fall times of the port output drivers (programmable values) should be taken into account, especially for higher baud rates.

The table below indicates the minimum frequencies of f_{SYS} in MHz for MultiCAN module operation (acceptance filtering, MO handling, etc.), that are required for a baud rate of 1 Mbit/s for the active CAN nodes (the highest CAN baud rate of the activated CAN nodes has to be taken into account). If less baud rate is desired, the values can be scaled linearly (e.g. for a maximum of 500 kbit/s, 50% of the indicated value are required).

The values imply that the CPU (or PEC) executes a maximum of accesses to the MultiCAN module. The values may contain rounding effects.

Table 20-14 Minimum Operating Frequencies [MHz]

Number of allocated message objects MO[1)]	1 CAN node active	2 CAN nodes active	3 CAN nodes active	4 CAN nodes active	5 CAN nodes active
16 MO	12	19	26	33	40
32 MO	15	23	30	37	44
64 MO	21	28	37	46	53
128 MO	40	45	50	55	61

1) Only those message objects have to be taken into account that are allocated to a CAN node. The unallocated message objects have no influence on the minimum operating frequency.

The baud rate generation of the MultiCAN being based on f_{SYS}, this frequency has to be chosen carefully to allow correct CAN bit timing. The required value of f_{SYS} is given by an integer multiple (n) of the CAN baud rate multiplied by the number of time quanta per CAN bit time. For example, to reach 1 Mbit/s with 20 tq per bit time, possible values of f_{SYS} are given by formula [n × 20] MHz, with n being an integer value, starting at 1.
In order to minimize jitter, it is not recommended to use the fractional divider mode for high baud rates.

20.3.2 Port Input Control

There is the possibility to select the input lines for the RXDCANx inputs for the CAN nodes. The selected input is connected to the CAN node and is also available to wake up the system.

20.3.3 Suspend Mode

The suspend mode can be triggered by the OCDS in order to freeze the state of the module and to have access to the registers (at least for read actions). There are several aspects related to the suspend mode:

- All actions are immediately stopped ("hard suspend"):
  The module clock is switched off as soon as the suspend line becomes active. This mode is supported by the fast switch off feature of the BPI. Write actions to the module are not supported and only combinatorial read actions deliver the desired data (the CAN RAM and the CAN registers can not be accessed).
  In this mode, all further module actions are disabled and there is a very high probability that the communication with other devices is made impossible and that the CAN bus is blocked by the device in hard suspend mode (e.g. if the suspended CAN just sends a dominant level). A normal continuation when the suspend mode is left is not always possible and reset must be activated.
- The current action is finished ("soft suspend"):
  The module functions are stopped (clock is still running!) automatically after internal actions have been finished, for example after a CAN frame has been sent out. Due to this behavior, the communication network is not blocked due to the suspend mode of one communication partner. Furthermore, all registers are accessible for read and write actions. As a result, the debugger can stop the module actions and modify registers. These modifications are taken into account after the suspend mode is left. This mode is designed to be able to modify registers or to read them by the OCDS while the rest of the systems is still running and not corrupted by the suspend mode.

In the MultiCAN module a suspend mechanism is implemented allowing the individual freeze of CAN nodes. The fast switch off feature (hard suspend) of the BPI must not be activated by the user in order to support the soft suspend mode. In order to allow the required flexibility for the system, each CAN node can be individually enabled for the soft suspend mode.

The hard suspend feature can be enabled/disabled for the complete MultiCAN module, whereas the soft suspend feature can be enabled/disabled independently for each CAN node. The fractional divider disables the CAN clock only if all CAN nodes signal that they can be suspended. A CAN node that is not active can always be suspended.

20.3.4 Interrupt Structure

The general interrupt structure is shown in the figure below. The interrupt event can trigger the interrupt generation. The interrupt pulse is generated independently from the interrupt flag in the interrupt status register. The interrupt flag can be reset by SW by writing a 0 to it.
If enabled by the related interrupt enable bit in the interrupt enable register, an interrupt pulse can be generated at one of the 16 interrupt output lines INT_Ox of the module. If more than one interrupt source is connected to the same interrupt node pointer (in the interrupt node pointer register), the requests are combined to one common line.

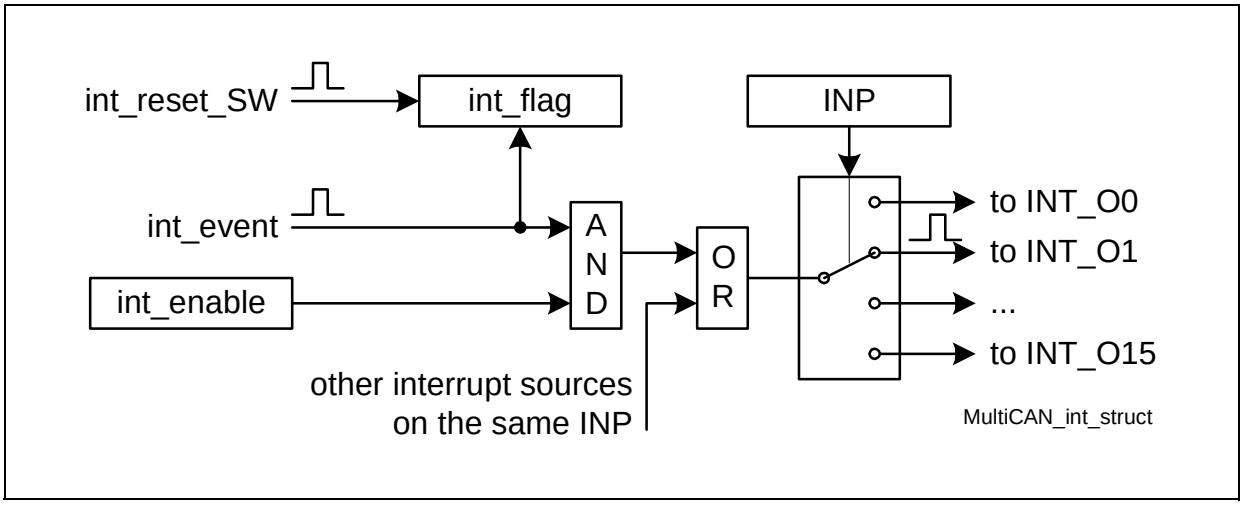


Figure 20-18 General Interrupt Structure

20.4 MultiCAN Module Implementation

This section describes CAN module interfaces with the clock control, port connections, interrupt control, and address decoding.

20.4.1 Interfaces of the CAN Module

Figure 20-19 shows the XC2000 specific implementation details and interconnections of the CAN module. The I/O lines of the CAN module kernel (two I/O lines of each CAN node) are connected to the ports as described in **Table 20-18**. The CAN module is further supplied by clock control, interrupt control, and address decoding logic.

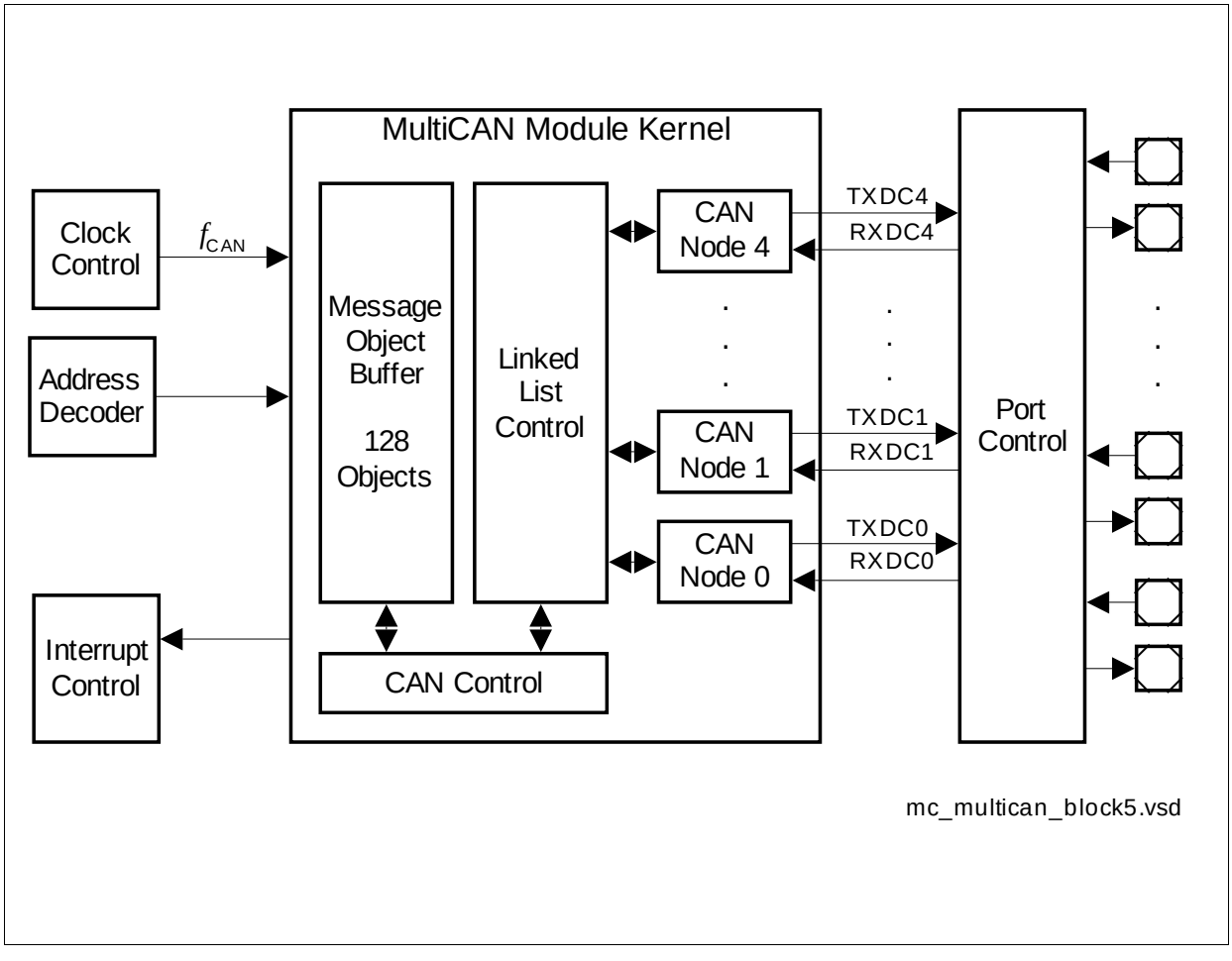


Figure 20-19 CAN Module Implementation and Interconnections

The MultiCAN interrupt control register x is connected to the CAN interrupt output line INT_Ox, with x = 15 - 0. Additionally, the signal INT_O15 can be used to start timers.

20.4.2 Module Clock Generation

As shown in **Figure 20-20**, the clock signals for the MultiCAN module are generated and controlled by a clock generation unit. This clock generation unit is responsible for the enable/disable control, the clock frequency adjustment, and the debug clock control.

The frequency control of the module timer clock f_{CAN} is performed via the CAN_FDR register.

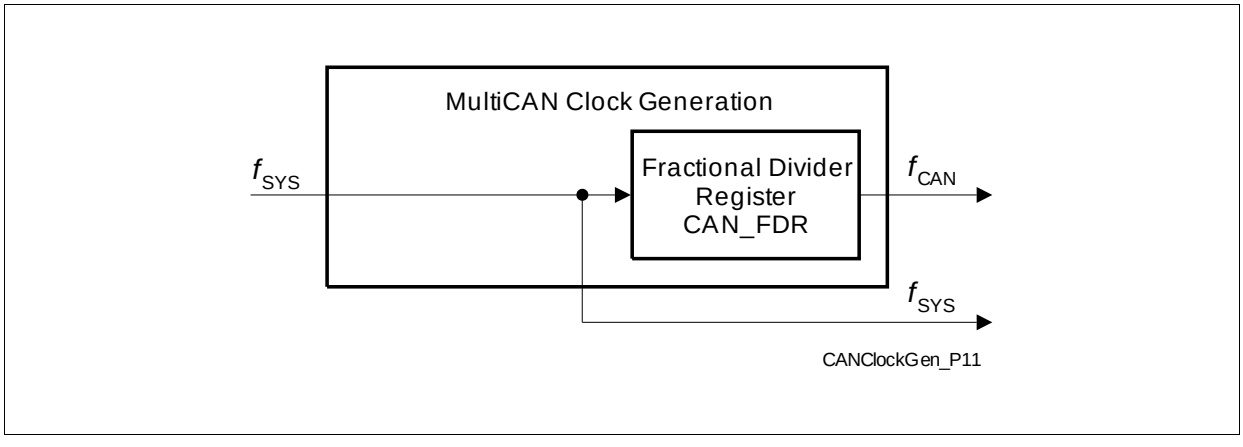


Figure 20-20 MultiCAN Module Clock Generation

The module control clock f_{SYS} is used inside the MultiCAN module kernel for control purposes such as e.g. for clocking of control logic and register operations. The frequency of f_{SYS} is identical to the system clock frequency f_{SYS}.

The module timer clock f_{CAN} is used inside the MultiCAN module kernels as input clock for all timing relevant operations.

The frequency of f_{CAN} is defined by:

$$f_{CAN} = f_{SYS} \times \frac{1}{n} \qquad \text{with n = 1024 - CAN\_FDR.STEP}$$

$$\text{or} \quad f_{CAN} = f_{SYS} \times \frac{n}{1024} \qquad \text{with n = 0-1023}$$

Note: The upper formula applies to normal divider mode of the fractional divider (CAN_FDR.DM = 01_B). The lower formula applies to fractional divider mode (CAN_FDR.DM = 10_B).

Note: Input signal ECEN of the MultiCAN fractional divider is wired to 0.

20.4.2.1 Fractional Divider Overview

The fractional divider allows to generate output clocks from an input clock using a programmable divider. The fractional divider divides an input clock f_{IN} either by the factor 1/n or by a fraction of n/1024 for any value of n from 0 to 1023 and outputs the clock

signals, f_{OUT}. The clock generation can be enabled/disabled by the fractional divider register control bit field FDR.DM.

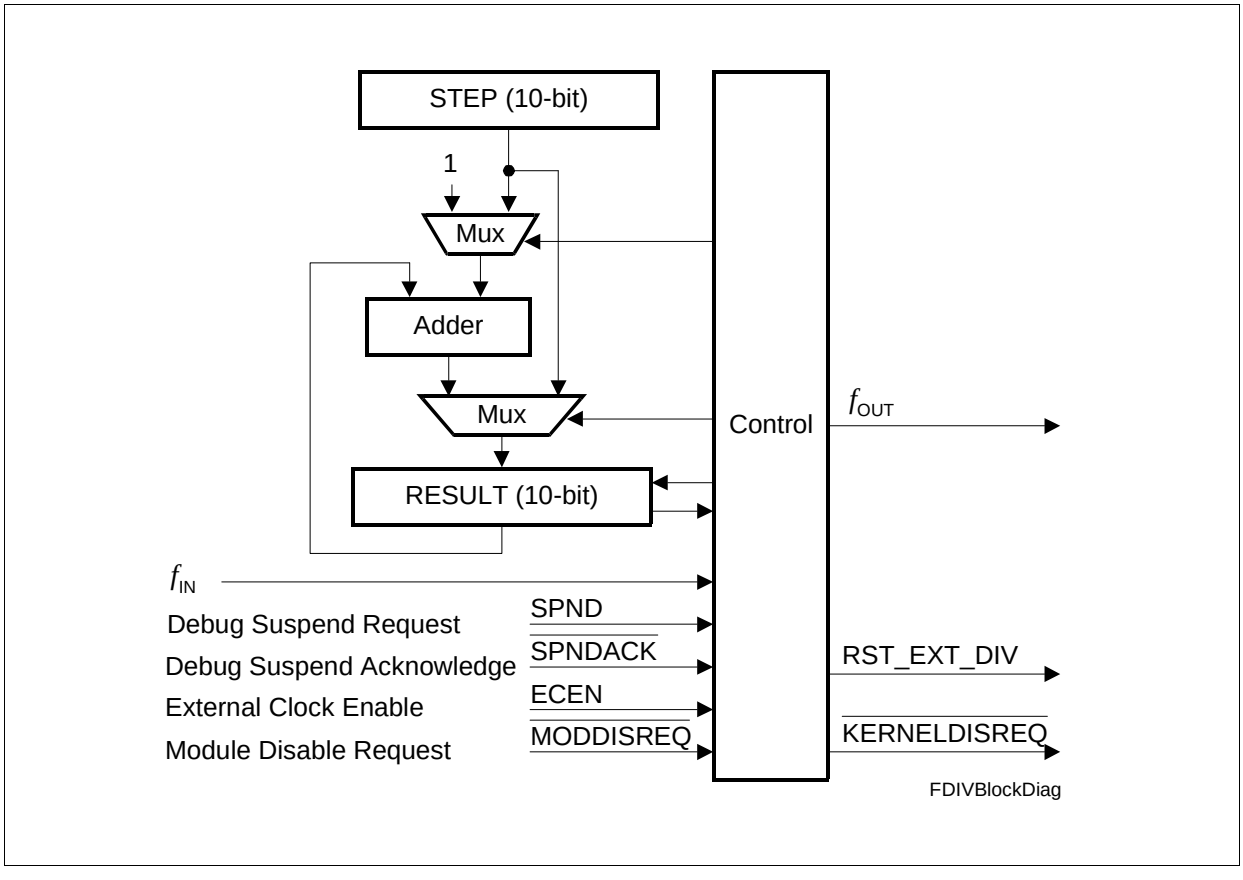


Figure 20-21 Fractional Divider Block Diagram

The clock generation in the fractional divider is further controlled by four input signals.

Table 20-15 Fractional Divider I/O Lines

Signal	I/O	Description
SPND	Input	Suspend Request Input is controlled by the debug system suspend request signal. It becomes active when a general suspend request is issued from the debug system to the on-chip modules.
$\overline{\text{SPNDACK}}$		Suspend Acknowledge Input is driven with the disable acknowledge signal from the module kernel. This signal is activated by the module kernel as a response to a suspend request that has been issued by the fractional divider via $\overline{\text{KERNELDISREQ}}$ = 0.
$\overline{\text{MODDISREQ}}$		Module Disable Request Input is connected to the disable request output from the CLC logic. An active signal at this input results in the activation of output signal $\overline{\text{KERNELDISREQ}}$.
ECEN		External Clock Enable Signal ECEN can be used to synchronize the fractional divider clock generation to external events.
$\overline{\text{KERNELDISREQ}}$	Output	Kernel Disable Request This output signal becomes active when either $\overline{\text{MODDISREQ}}$ is activated or when SPND becomes active.
RST_EXT_DIV		Reset External Divider This output signal allows to control (stop/reset) external divider stages for f_{OUT}.
f_{OUT}		Module Clock Enable Signal fOUT is the enable signal for the module clock. The module clock itself is built by and-ing the f_{OUT} enable signal with f_{IN}. Module clock frequency references mostly refer to the AND combination of f_{OUT} with f_{IN}.

The fractional divider has two operating modes:

- Normal divider mode
- Fractional divider mode

Normal Divider Mode

In normal divider mode (FDR.DM = 01_B) the fractional divider behaves like a reload counter (addition of +1) that generates an output clock pulse at f_{OUT} on the transition from $3FF_H$ to 000_H. FDR.RESULT represents the counter value and FDR.STEP defines the reload value.

The output frequencies in normal divider mode are defined according the following formulas:

$$f_{OUT} = f_{IN} \times \frac{1}{n} \qquad \text{with n = 1024 - STEP}$$

In order to get $f_{OUT} = f_{IN}$ STEP must be programmed with $3FF_H$. **Figure 20-22** shows the operation of the normal divider mode with a reload value of FDR.STEP = $3FD_H$. The clock frequency of f_{OUT} is represented by and-ing the f_{OUT} enable signal with f_{IN}.

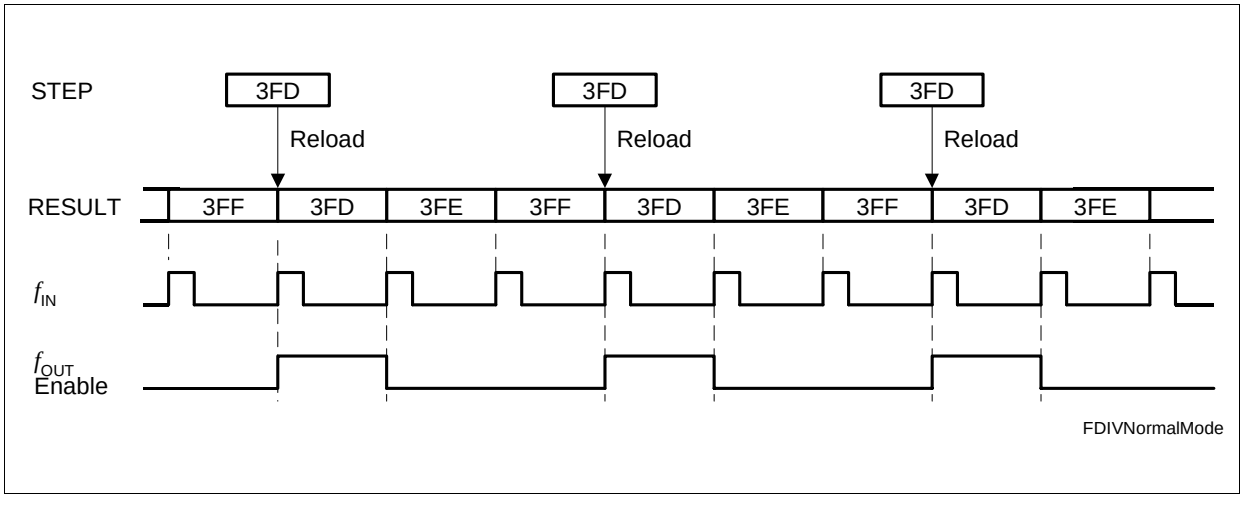


Figure 20-22 Normal Mode Timing

Fractional Divider Mode

When the fractional divider mode is selected (FDR.DM = 10_B), the output clock f_{OUT} is derived from the input clock f_{IN} by division of a fraction of n/1024 for any value of n from 0 to 1023. In general, the fractional divider mode allows to program the average output clock frequency with a higher accuracy than in normal divider mode.

In fractional divider mode an output clock pulse at f_{OUT} is generated dependent on the result of the addition FDR.RESULT + FDR.STEP. If the addition leads to an overflow over $3FF_H$ a pulse is generated at f_{OUT}. Note that in fractional divider mode the clock f_{OUT} can have a maximum period jitter of one f_{IN} clock period.

The output frequencies in fractional divider mode are defined according the following formulas:

$$f_{OUT} = f_{IN} \times \frac{n}{1024} \quad \text{with n = 0-1023}$$

Figure 20-23 shows the operation of the fractional divider mode with a reload value of FDR.STEP = 234_H (= factor 564/1024 = 0.55). The clock frequency of f_{OUT} is represented by and-ing the f_{OUT} enable signal with f_{IN}.

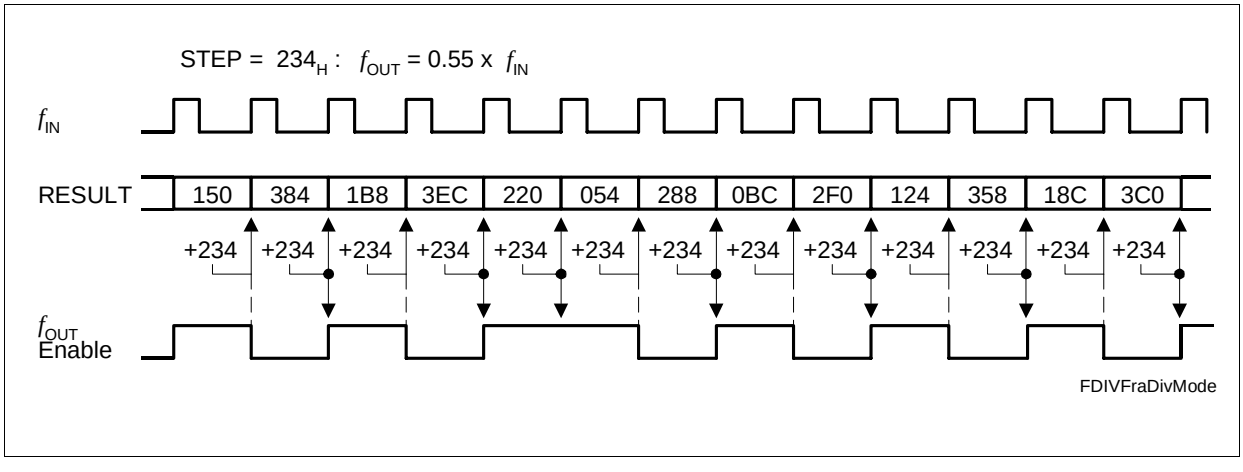


Figure 20-23 Fractional Divider Mode Timing

Suspend Mode Control

The fractional divider allows to control its operation according to the input Suspend Request (SPND). This input is activated in suspend mode by the on-chip debug control logic. In suspend mode, the module registers are accessible for read and write actions, but the other module internal functions are frozen. Suspend mode is requested by SPND = 1. Suspend mode is entered one f_{IN} clock cycle after the suspend mode request has been acknowledged by setting SPNDACK to 0 (granted suspend mode) and

FDR.SC is not equal 00_B (clock output signal disabled). Suspend mode is immediately entered when bit SM is set to 1 and FDR.SC is not equal 00_B (immediate suspend mode

The state of signals SPND and SPNDACK is latched in two status flags of register FDR, SUSREQ and SUSACK. SPND and (SPNDACK or bit SM) must remain set both to maintain the suspend mode.

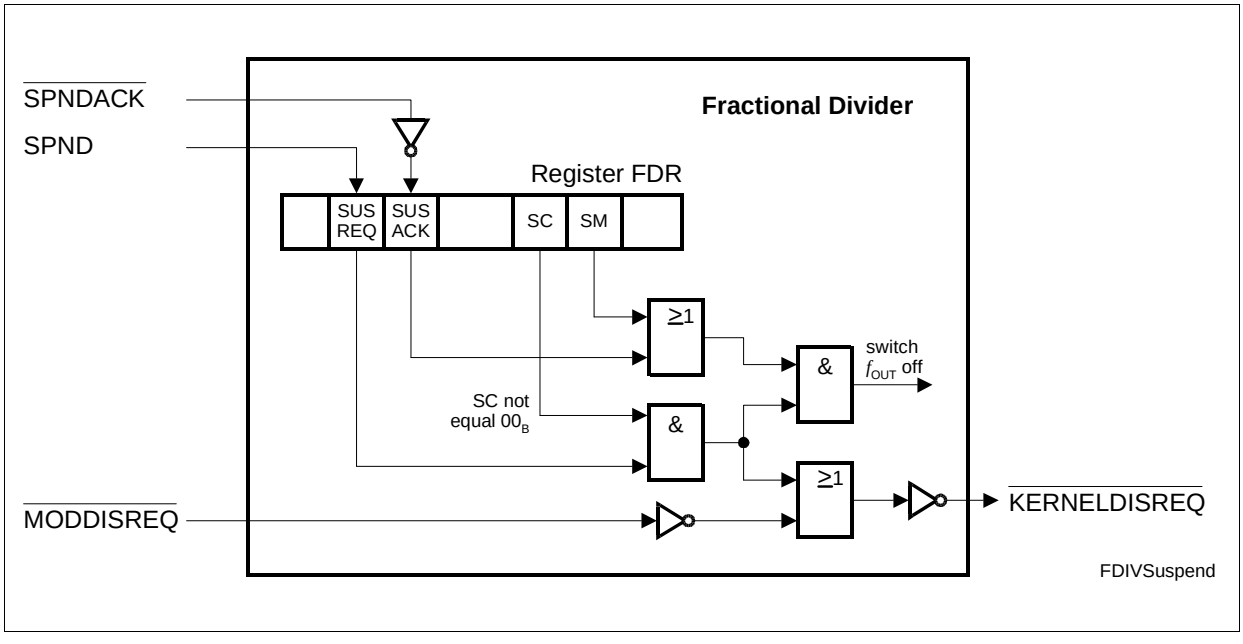


Figure 20-24 Suspend Mode Configuration

The Kernel Disable Request signal KERNELDISREQ becomes always active when MODDISREQ is activated, independently of the suspend mode settings in the fractional divider logic.

External Clock Enable

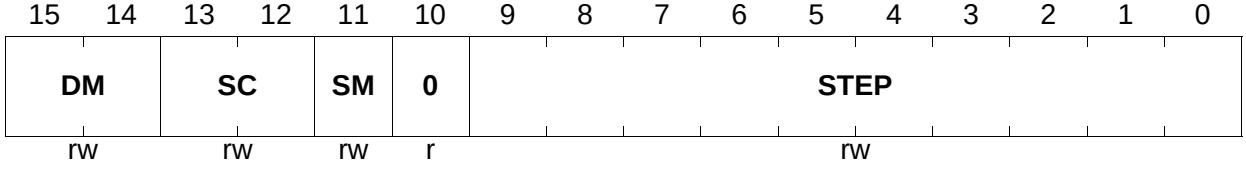
When the module clock generation has been disabled by software (setting FDR.DISCLK = 1), the disable state can be left via input ECEN = 1 (hardware controlled). This feature is enabled when FDR.ENHW = 1. In the MultiCAN module, signal ECEN is tied to 0.

Registers Overview

Fractional Divider Registers

The fractional divider contains two registers, FDRL (lower 16 bits) and FDRH (higher 16 bits).

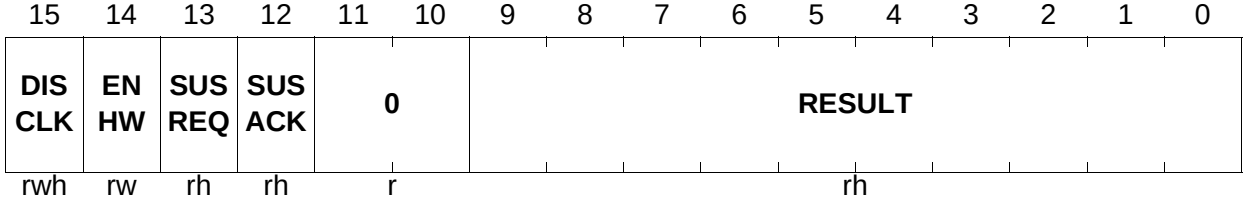
FDRL
Fractional Divider Register L **($0C_H$)** **Reset Value: 0000H**



Field	Bits	Type	Description
STEP	[9:0]	rw	**Step Value** In normal divider mode STEP contains the reload value for RESULT. In fractional divider mode this bit field defines the 10-bit value that is added to the RESULT with each input clock cycle.
SM	11	rw	**Suspend Mode** SM selects between granted or immediate suspend mode. 0 Granted suspend mode selected 1 Immediate suspend mode selected
SC	[13:12]	rw	**Suspend Control** This bit field defines the behavior of the fractional divider in suspend mode (bit SUSREQ and SUSACK set). 00 Clock generation continues. 01 Clock generation is stopped and the clock output signals are not generated. RESULT is not changed except when writing bit field DM with 01B or 10B. 10 Clock generation is stopped and the clock output signals are not generated. RESULT is loaded with $3FF_H$. 11 Same as SC = 10B but RST_EXT_DIV is 1 (independently of bit field DM).

Field	Bits	Type	Description
DM	[15:14]	rw	**Divider Mode** This bit fields defines the functionality of the fractional divider block. 00 Fractional divider is switched off; no output clock is generated. RST_EXT_DIV is 1. RESULT is not updated (default after reset). 01 Normal divider mode selected. 10 Fractional divider mode selected. 11 Fractional divider is switched off; no output clock is generated. RESULT is not updated.
0	others	r	**Reserved** read as 0; should be written with 0.

FDRH
Fractional Divider Register H **($0E_H$)** **Reset Value: 0000_H**



Field	Bits	Type	Description
RESULT	[9:0]	rh	**Result Value** In normal divider mode RESULT acts as reload counter (addition +1). In fractional divider mode this bit field contains the result of the addition RESULT+STEP. If DM is written with 01_B or 10_B, RESULT is loaded with $3FF_H$.
SUSACK	12	rh	**Suspend Mode Acknowledge** 0 Suspend mode is not acknowledged. 1 Suspend mode is acknowledged. Suspend mode is entered when SUSACK and SUSREQ are set.

Field	Bits	Type	Description
SUSREQ	13	rh	**Suspend Mode Request** 0 Suspend mode is not requested. 1 Suspend mode is requested. Suspend mode is entered when SUSREQ and SUSACK are set.
ENHW	14	rw	**Enable Hardware Clock Control** 0 Bit DISCLK cannot be reset by HW by a high level at input signal ECEN. 1 Bit DISCLK is reset by hardware while input signal ECEN is at high level.
DISCLK	15	rwh	**Disable Clock** 0 Clock generation of fOUT is enabled according to the setting of bit field DM. 1 Fractional divider is stopped. The enable signal *f*OUT becomes inactive. No change except when writing bit field DM.
0	others	r	**Reserved** read as 0; should be written with 0.

Fractional Divider Operation Modes

Table 20-16 Fractional Divider Function Table

Mode	SC	DM	RES_EXT_DIV	RESULT	f_{OUT}	Operation of Fractional Divider
Normal Mode	–	00	1	unchanged	inactive	switched off
		01	0	continuously updated[1)]	active	normal divider mode
		10				fractional divider mode
		11		unchanged	inactive	switched off

Table 20-16 Fractional Divider Function Table

Mode	SC	DM	RES_EXT_DIV	RESULT	f_{OUT}	Operation of Fractional Divider
Suspend Mode	00	00	1	unchanged	inactive	switched off
		01	0	continuously updated[1)]	active	normal divider mode
		10				fractional divider mode
		11		unchanged	inactive	switched off
	01	00	1	unchanged	inactive	switched off
		01	0	unchanged[1)]		halted
		10				
		11		unchanged		switched off
	10	00	1	loaded with $3FF_H$	inactive	switched off
		01	0			halted
		10				
		11				switched off
	11	–	1	loaded with $3FF_H$	inactive	switched off

1) Each write operation to FDR with DM = 01_B or 10_B sets RESULT to $3FF_H$.

20.4.3 Mode Control Behavior

The MultiCAN module provides two mechanisms to stop participation in CAN traffic:

- Suspend Mode:
  The suspend mode request is issued by the OCDS block. The sensitivity of a CAN node to a suspend request can be individually enabled/disabled for each CAN node. In suspend mode, a CAN node correctly finishes a running CAN frame, but does not start a new one.
- Immediate Stop Mode:
  The immediate stop mode is entered when a stop mode is requested by the mode control of the device, configured by MCAN_KSCCFG. If an immediate stop is requested, the CAN module immediately stops all CAN activity (even within a running frame) and sets all transmit outputs to 1. In oder to allow CAN operation, bit field NOMCFG has to be set to a run mode. To support suspend mode (see description above), bit field SUMCFG has to be set to run mode to avoid immediate stop mode.

20.4.4 Mode Control

The mode control concept for system control tasks, such as power saving, or suspend request for debugging, allows to program the module behavior under different device operating conditions. The behavior of the MultiCAN kernel can be programmed for each of the device operating modes, that are requested by the global state control part of the SCU. MultiCAN has an associated register **CAN_KSCFG** defining the behavior of the kernel of the module in the following device operating modes:

- **Normal operation:**
  This operating mode is the default operating mode when neither a suspend request nor a clock-off request are pending. The module clock is not switched off and the MultiCAN registers can be read or written. The kernel behavior is defined by KSCFG.NOMCFG.
- **Suspend mode:**
  This operating mode is requested when a suspend request (issued by a debugger) is pending in the device. The module clock is not switched off and the MultiCAN registers can be read or written. The kernel behavior is defined by KSCFG.SUMCFG.
- **Clock-off mode:**
  This operating mode is requested for power saving purposes. The module clock is switched off automatically when all kernels of the MultiCAN module reached their specified state in a stop mode. In this case, MultiCAN registers can not be accessed. The kernel behavior is defined by KSCFG.COMCFG.

For the MultiCAN module, the following internal actions can be influenced by mode control:

- A current transmision of a CAN message:
  If there is a pending request, it can be started. This start has to be enabled by the mode control. If the current kernel mode allows the start (run modes 0 and 1), it will be executed. If the kernel mode does not allow a start (stop modes 0 and 1), the request is not started. The start request is not cancelled, but frozen. A “frozen” request is started as programmed if the kernel mode is changed to a run mode again.

The behavior of the MultiCAN kernel can be programmed for each of the device operating modes (normal operation, suspend mode, clock-off mode). Therefore, the MultiCAN kernel supports four kernel modes, as shown in **Table 20-17**.

Table 20-17 MultiCAN Kernel Behavior

Kernel Mode	Kernel Behavior	Code
run mode 0	kernel operation as specified, no impact on data transfer (same behavior for run mode 0 and run mode 1)	00_B
run mode 1		01_B
stop mode 0	A currently running transfer is completely finished and the result is treated. Pending requests are not taken into account (but not deleted). They restart after entering a run mode as programmed. The arbiter continues as programmed.	10_B
stop mode 1	Like stop mode 0, but the arbiter is stopped after it has finished its arbitration round.	11_B

Generally, bit field KSCFG.NOMCFG should be configured for run mode 0 as default setting for standard operation. If the MultiCAN kernel should not react to a suspend request (and to continue operation as in normal mode), bit field KSCFG.SUMCFG has to be configured with the same value as KSCFG.NOMCFG. If the MultiCAN kernel should show a different behavior and stop operation when a specific stop condition is reached, the code for stop mode 0 or stop mode 1 has to be written to KSCFG.SUMCFG. A similar mechanism applies for the clock-off mode with the possibility to program the desired behavior by bit field KSCFG.COMCFG.

Note: The stop mode selection strongly depends on the application needs and it is very unlikely that different stop modes are required in parallel in the same application. As a result, only one stop mode type (either 0 or 1) should be used in the bit fields in register KSCFG. Do not mix stop mode 0 and stop mode 1 and avoid transitions from stop mode 0 to stop mode 1 (or vice versa) for the MultiCAN module.

Please note that bit KSCFG.MODEN should only be set by SW while all configuration fields are configured for run mode 0.

20.4.5 Mode Control Register Description

20.4.5.1 Kernel State Configuration Register

The kernel state configuration register KSCFG allows the selection of the desired kernel modes for the different device operating modes.
Bit fields KSCFG.NOMCFG and KSCFG.COMCFG are reset by a class 3 reset. Bit field KSCFG.SUMCFG is reset by a class 1 reset.

*Note: The coding of the bit fields NOMCFG, SUMCFG and COMCFG is described in **Table 20-17**.*

CAN_KSCFG
Kernel State Configuration Register

SFR(FE1E$_H$) **Reset Value: 0000$_H$**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BP COM	**0**	**COMCFG**		**BP SUM**	**0**	**SUMCFG**		**BP NOM**	**0**	**NOMCFG**		**0**		**BP MOD EN**	**MOD EN**
w	r	rw		w	r	rw		w	r	rw		r		w	rw

Field	Bits	Type	Description
MODEN	0	rw	**Module Enable** This bit enables the module kernel clock and the module functionality. 0_B The module is switched off immediately (without respecting a stop condition). It does not react on mode control actions and the module clock is switched off. The module does not react on read accesses and ignores write accesses (except to KSCFG). 1_B The module is switched on and can operate. After writing 1 to MODEN, it is recommended to read register KSCFG to avoid pipeline effects in the control block before accessing other ADC registers. *Note: This bit is reset by a class 3 reset.*

Field	Bits	Type	Description
BPMODEN	1	w	**Bit Protection for MODEN** This bit enables the write access to the bit MODEN. It always reads 0. 0_B MODEN is not changed. 1_B MODEN is updated with the written value. *Note: This bit is reset by a class 3 reset.*
NOMCFG	[5:4]	rw	**Normal Operation Mode Configuration** This bit field defines the kernel mode applied in normal operation mode. $0X_B$ The module is switched on. $1X_B$ The module is switched off. This field is taken into account for CR = 00 or 11. *Note: This bit is reset by a class 3 reset.*
BPNOM	7	w	**Bit Protection for NOMCFG** This bit enables the write access to the bit field NOMCFG. It always reads 0. 0_B NOMCFG is not changed. 1_B NOMCFG is updated with the written value. *Note: This bit is reset by a class 3 reset.*
SUMCFG	[9:8]	rw	**Suspend Mode Configuration** This bit field defines the kernel mode applied in suspend mode. $0X_B$ The module is switched on. This is the recommended setting in order to have soft suspend behavior. $1X_B$ The module is switched off. This field is taken into account for CR = 01. *Note: This bit is reset by a class 1 reset.*
BPSUM	11	w	**Bit Protection for SUMCFG** This bit enables the write access to the bit field SUMCFG. It always reads 0. 0_B SUMCFG is not changed. 1_B SUMCFG is updated with the written value. *Note: This bit is reset by a class 1 reset.*

Field	Bits	Type	Description
COMCFG	[13:12]	rw	**Clock Off Mode Configuration** This bit field defines the kernel mode applied in clock off mode. $0X_B$ The module is switched on. $1X_B$ The module is switched off. This field is taken into account for CR = 10. *Note: This bit is reset by a class 3 reset.*
BPCOM	15	w	**Bit Protection for COMCFG** This bit enables the write access to the bit field COMCFG. It always reads 0. 0_B COMCFG is not changed. 1_B COMCFG is updated with the written value. *Note: This bit is reset by a class 3 reset.*
0	[3:2], 6, 10, 14	r	**Reserved** returns 0 if read; should be written with 0;

Note: The bit protection bits BPxxx allow partly modification of the configuration bits with a single write operation (without the need of a read-modify-write mechanism handled by the CPU).

20.4.6 Connection of External Signals

The following table shows the digital connections of the MultiCAN signals with other modules or pins in the XC2000 device.

The selected input signal (selected by bit field RXSEL) for each CAN node is made available by signal CANxINS (CAN node x input signal, with x = 4 - 0). These signals can be used in the SCU for wake-up purposes and to allow two CAN nodes receive the same signal (analysis and safety feature).

Table 20-18 MultiCAN Connections in XC2000

Signal	from/to Module	I/O to CAN	Can be used to/as
MultiCAN Node 0 Signals			
RXDC0A	P2.3	I	receive input A (NPCR0.RXSEL = 000_B)
RXDC0B	P0.3		receive input B (NPCR0.RXSEL = 001_B)
RXDC0C	P2.0		receive input C (NPCR0.RXSEL = 010_B)
RXDC0D	P2.6		receive input D (NPCR0.RXSEL = 011_B)
RXDC0E	ESR1		receive input E (NPCR0.RXSEL = 100_B)
RXDC0 [G:F]	1		receive inputs [G:F] (NPCR0.RXSEL = 101_B to 110_B)
RXDC0H	0		receive input H (NPCR0.RXSEL = 111_B)
TXDC0	P0.1	O	transmit output
	P0.2		
	P2.1		
	P2.4		
	P2.5		
MultiCAN Node 1 Signals			
RXDC1A	P2.4	I	receive input A (NPCR1.RXSEL = 000_B)
RXDC1B	P0.4		receive input B (NPCR1.RXSEL = 001_B)
RXDC1C	P2.7		receive input C (NPCR1.RXSEL = 010_B)
RXDC1D	CAN0INS		receive input D (NPCR1.RXSEL = 011_B)
RXDC1E	ESR2		receive input E (NPCR1.RXSEL = 100_B)
RXDC1 [G:F]	1		receive inputs [G:F] (NPCR1.RXSEL = 101_B to 110_B)
RXDC1H	0		receive input H (NPCR1.RXSEL = 111_B)

Table 20-18 MultiCAN Connections in XC2000 (cont'd)

Signal	from/to Module	I/O to CAN	Can be used to/as
TXDC1	P0.6	O	transmit output
	P2.2		
	P2.9		
MultiCAN Node 2 Signals			
RXDC2A	P4.3	I	receive input A (NPCR2.RXSEL = 000_B)
RXDC2B	P10.11		receive input B (NPCR2.RXSEL = 001_B)
RXDC2C	CAN1INS		receive input C (NPCR2.RXSEL = 010_B)
RXDC2D	ESR0		receive input D (NPCR2.RXSEL = 011_B)
RXDC2 [G:E]	1		receive inputs [G:E] (NPCR2.RXSEL = 100_B to 110_B)
RXDC2H	0		receive input H (NPCR2.RXSEL = 111_B)
TXDC2	P4.1	O	transmit output
	P4.2		
	P10.12		
MultiCAN Node 3 Signals			
RXDC3A	P3.3	I	receive input A (NPCR3.RXSEL = 000_B)
RXDC3B	P3.0		receive input B (NPCR3.RXSEL = 001_B)
RXDC3C	P10.14		receive input C (NPCR3.RXSEL = 010_B)
RXDC3D	CAN2INS		receive input D (NPCR3.RXSEL = 011_B)
RXDC3 [G:E]	1		receive inputs [G:E] (NPCR3.RXSEL = 100_B to 110_B)
RXDC3H	0		receive input H (NPCR3.RXSEL = 111_B)
TXDC3	P3.1	O	transmit output
	P3.2		
	P10.13		
MultiCAN Node 4 Signals			

Table 20-18 MultiCAN Connections in XC2000 (cont'd)

Signal	from/to Module	I/O to CAN	Can be used to/as
RXDC4A	P3.4	I	receive input A (NPCR4.RXSEL = 000_B)
RXDC4B	P7.0		receive input B (NPCR4.RXSEL = 001_B)
RXDC4C	P10.7		receive input C (NPCR4.RXSEL = 010_B)
RXDC4D	CAN3INS		receive input D (NPCR4.RXSEL = 011_B)
RXDC4 [G:E]	1		receive inputs [G:E] (NPCR4.RXSEL = 100_B to 110_B)
RXDC4H	0		receive input H (NPCR4.RXSEL = 111_B)
TXDC4	P3.6	O	transmit output
	P7.1		
	P7.2		
	P10.6		
General MultiCAN Signals			
INT_O[15:0]	interrupt controller	O	interrupt output lines (service requests)
	INT_O15 to CCU62 and CCU63		notification for timer start

20.4.7 MultiCAN Module Register Address Map

In the XC2000, the registers of the MultiCAN module are located in the following address range:

Table 20-19 Registers Address Space

Module	Base Address	End Address	Note
CAN	20 0000_H	20 $3FFF_H$	

Note: he complete and detailed address map of the MultiCAN modules is described in the chapter "Register Overview" of the XC2000 System Units User's Manual.

In the MultiCAN module address range, the following register blocks are located at the offset start addresses, see **Figure 20-25**:

- 0000_H General registers for clock control, fractional divider, ID
- 0100_H Global Module Control registers
- 0200_H CAN node 0 registers
- 1000_H Message object memory (32 bytes for each object)
- $3E00_H$ Scheduler memory (128 * 4 bytes)

The CAN RAM is automatically initialized after reset by the list controller in order to ensure correct list pointers in each message object. The end of this CAN RAM initialization is indicated by bit PANCTR.BUSY becoming inactive. Before the end of the initialization sequence, the CAN module must not be accessed with other instructions than polling for bit PANCTR.BUSY.

The CAN RAM can be optionally enabled for parity detection. The feature is controlled in the SCU.

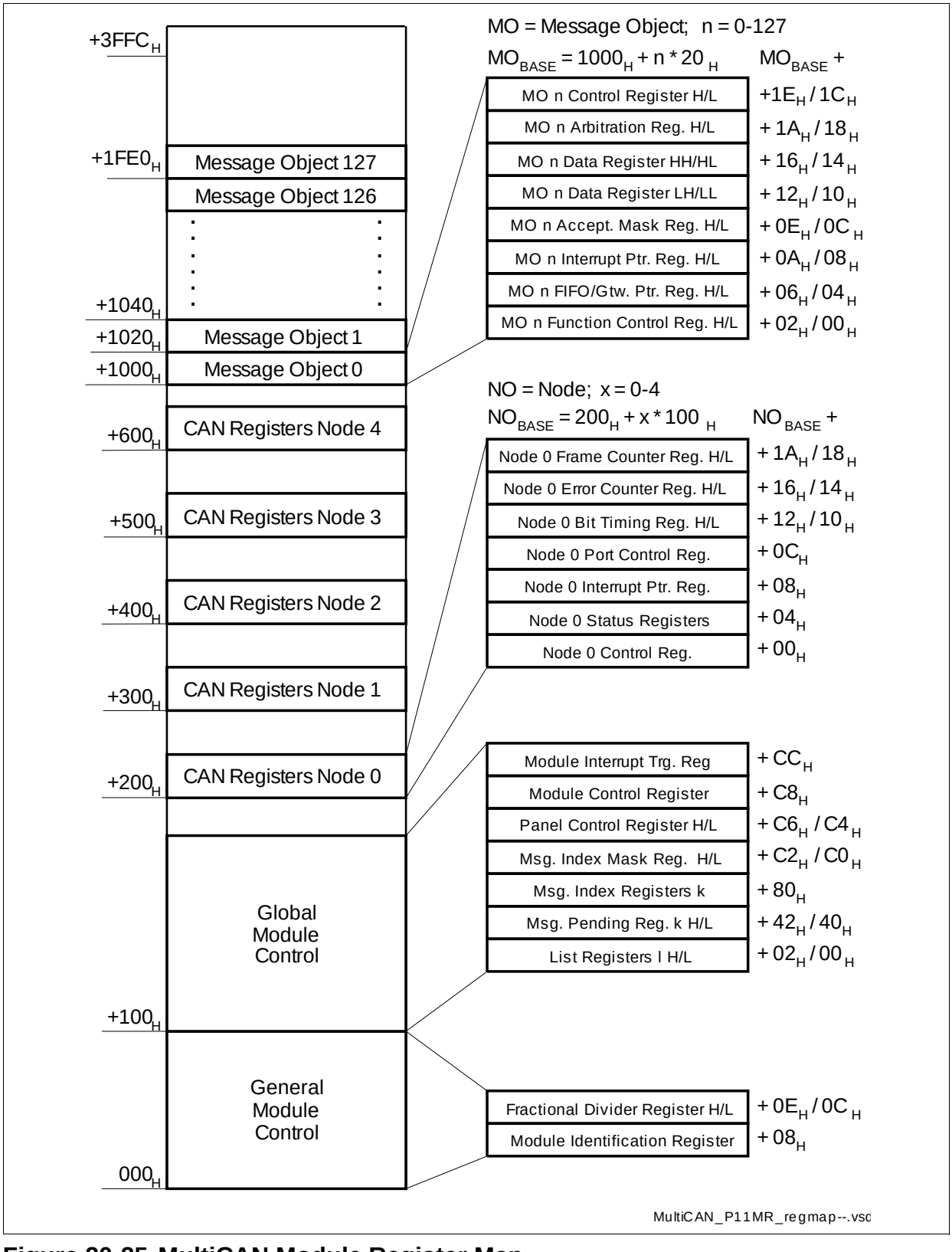


Figure 20-25 MultiCAN Module Register Map

Keyword Index

This section lists a number of keywords which refer to specific details of the XC2000 in terms of its architecture, its functional units or functions. This helps to quickly find the answer to specific questions about the XC2000.

This User's Manual consists of two Volumes, "System Units" and "Peripheral Units". For your convenience this keyword index refers to both volumes, so you can immediately find the reference to the desired section in the corresponding document ([1] or [2]).

*Note: Registers are listed in a separate index: **Register Index**.*

D

E

F

G

H

I

L

M

O

P

R

S

T

W

Register Index

This section lists the registers of the XC2000. This helps to quickly find the reference to the description of the respective register.

This User's Manual consists of two Volumes, "System Units" and "Peripheral Units". For your convenience this register index refers to both volumes, so you can immediately find the reference to the desired section in the corresponding document ([1] or [2]).

*Note: Keywords are listed in a separate index: **Keyword Index**.*

D

E

F

G

I

M

O

P

Q

R

S

T

U

V

X

Z

w w w . i n f i n e o n . c o m

Published by Infineon Technologies AG