User's Manual, V2.0, Feb. 2001

TC1775

Peripheral Units

32-Bit Single-Chip Microcontroller

Microcontrollers

Infineon technologies

Never stop thinking.

Edition 2001-02

Published by Infineon Technologies AG,
St.-Martin-Strasse 53,
D-81541 München, Germany

Attention please!

The information herein is given to describe certain components and shall not be considered as warranted characteristics.

Terms of delivery and rights to technical change reserved.

We hereby disclaim any and all warranties, including but not limited to warranties of non-infringement, regarding circuits, descriptions and charts stated herein.

Infineon Technologies is an approved CECC manufacturer.

Information

For further information on technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies Office in Germany or our Infineon Technologies Representatives worldwide.

Warnings

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

Infineon Technologies Components may only be used in life-support devices or systems 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.

TC1775

Peripheral Units

32-Bit Single-Chip Microcontroller

Microcontrollers

Infineon technologies

Never stop thinking.

TC1775 Peripheral Units User's Manual
Revision History: 2001-02 V2.0

Previous Version: V1.0, 2000-11 (B-Step, PDF only)[1)]
V1.1, 2000-05 (A-Step, PDF and limited print)[2)]
V1.0, 2000-04 (A-Step, PDF only)[2)]

Page	Subjects (major changes since last revision)

1) This version is an intermediate version for the B-Step created for release purposes of the final version V2.0.

2) These versions for the A-Step have been created as an "Advance Information" for key customers and have not been published officially.

We Listen to Your Comments
Any information within this document that you feel is wrong, unclear or missing?
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

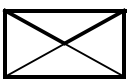


Table of Contents **Page**

Table of Contents **Page**

Table of Contents **Page**

Table of Contents **Page**

Table of Contents **Page**

Table of Contents **Page**

Table of Contents **Page**

1 Introduction

1.1 About This Document

This document is designed to be read primarily by design engineers and software engineers who need a detailed description of the interactions of the TC1775 functional units, registers, instructions, and exceptions.

1.1.1 Related Documentations

A complete description of the TriCore architecture is found in the document titled “TriCore Architecture Manual”. The architecture of the TC1775 is described separately this way because of the configurable nature of the TriCore specification: different versions of the architecture may contain a different mix of systems components. The TriCore architecture, however, remains constant across all derivative designs in order to preserve compatibility.

Additionally to this “TC1775 Peripheral Units User’s Manual”, a second document, the “TC1775 System Units User’s Manual”, is available. These two User’s Manuals together with the “TriCore Architecture Manual” are required for the understanding the complete TC1775 microcontroller functionality.

Implementation-specific details such as electrical characteristics and timing parameters of the TC1775 can be found in the “TC1775 Data Sheet”.

1.1.2 Textual Conventions

This document uses the following textual conventions for named components of the TC1775:

- Functional units of the TC1775 are given in plain UPPER CASE. For example: “The EBU provides an interface to external peripherals.”
- Pins using negative logic are indicated by an overbar. For example: “The $\overline{\text{BYPASS}}$ pin is latched with the rising edge of the $\overline{\text{PORST}}$ pin.”
- Bit fields and bits in registers are in general referenced as “Register name.Bit field” or “Register name.Bit”. For example: “The Current CPU Priority Number bit field ICR.CCPN is cleared.”. Most of the register names contain a module name prefix, separated by a underscore character “_” from the real register name (for example, “ASC0_CON”, where “ASC0” is the module name prefix, and “CON” is the real register name). In chapters describing peripheral modules the real register name is referenced also as kernel register name.
- Variables used to describe sets of processing units or registers appear in mixed-case font. For example, register name “MSGCFGn” refers to multiple “MSGCFG” registers with variable n. The bounds of the variables are always given where the register expression is first used (for example, “n = 31-0”), and is repeated as needed in the rest of the text.

- The default radix is decimal. Hexadecimal constants are suffixed with a subscript letter “H”, as in 100_H. Binary constants are suffixed with a subscript letter “B”, as in: 111_B.
- When the extent of register fields, groups of signals, or groups of pins are collectively named in the body of the document, they are given as “NAME[A:B]“, which defines a range for the named group from B to A. Individual bits, signals, or pins are given as “NAME[C]” where the range of the variable C is given in the text. For example: CLKSEL[2:0], and TOS[0].
- Units are abbreviated as follows:
  - **MHz** = Megahertz
  - **µs** = Microseconds
  - **kBaud, kBit** = 1000 characters/bits per second
  - **MBaud, MBit** = 1,000,000 characters per second
  - **KByte** = 1024 bytes of memory
  - **MByte** = 1048576 bytes of memory

  In general, the *k* prefix scales a unit by 1000 whereas the K prefix scales a unit by 1024. Hence, the KByte unit scales the expression preceding it by 1024. The kBaud unit scales the expression preceding it by 1000. The M prefix scales by 1,000,000 or 1048576, and µ scales by .000001. For example, 1 KByte is 1024 bytes, 1 MByte is 1024 × 1024 bytes, 1 kBaud/kBit are 1000 characters/bits per second, 1 MBaud/MBit are 1000000 characters/bits per second, and 1 MHz is 1,000,000 Hz.
- Data format quantities are defined as follows:
  - **Byte** = 8-bit quantity
  - **Half-word** = 16-bit quantity
  - **Word** = 32-bit quantity
  - **Double-word** = 64-bit quantity

1.1.3 Reserved, Undefined, and Unimplemented Terminology

In tables where register bit fields are defined, the following conventions are used to indicate undefined and unimplemented function. Further, types of bits and bit fields are defined using the abbreviations as shown in **Table 1-1**.

Table 1-1 Bit Function Terminology

Function of Bits	Description
Unimplemented	Register bit fields named **0** indicate unimplemented functions with the following behavior. - Reading these bit fields returns 0. - Writing these bit fields has no effect. These bit fields are reserved. When writing, software should always set such bit fields to 0 in order to preserve compatibility with future products.

Table 1-1 Bit Function Terminology (cont'd)

Function of Bits	Description
Undefined	Certain bit combinations in a bit field can be labeled "Reserved", indicating that the behavior of the TC1775 is undefined for that combination of bits. Setting the register to undefined bit combinations may lead to unpredictable results. Such bit combinations are reserved. When writing, software must always set such bit fields to legal values as given in the tables.
rw	The bit or bit field can be read and written.
r	The bit or bit field can only be read (read-only).
w	The bit or bit field can only be written (write-only).
h	The bit or bit field can also be modified by hardware (such as a status bit). This symbol can be combined with 'rw' or 'r' bits to 'rwh' and 'rh' bits.

1.1.4 Register Access Modes

Read and write access to registers and memory locations are sometimes restricted. In memory and register access tables, the following terms are used.

Table 1-2 Access Terms

Symbol	Description
U	Access permitted in User Mode 0 or 1
SV	Access permitted in Supervisor Mode
R	Read-only register
32	Only 32-bit word accesses are permitted to that register/address range
E	Endinit protected register/address
PW	Password protected register/address
NC	No change, indicated register is not changed
BE	Indicates that an access to this address range generates a Bus Error
nBE	Indicates that no Bus Error is generated when accessing this address range, even though it is either an access to an undefined address or the access does not follow the given rules
nE	Indicates that no Error is generated when accessing this address or address range, even though the access is to an undefined address or address range. True for CPU accesses (MTCR/MFCR) to undefined addresses in the CSFR range
X	Undefined value or bit

1.1.5 Abbreviations

The following acronyms and termini are used within this document:

ADC	Analog-to-Digital Converter
AGPR	Address General Purpose Register
ALE	Address Latch Enable
ALU	Arithmetic and Logic Unit
ASC	Asynchronous/Synchronous Serial Controller
BCU	Bus Control Unit
CAN	Controller Area Network (License Bosch)
CISC	Complex Instruction Set Computing
CPS	CPU Slave Interface Registers
CPU	Central Processing Unit
CSFR	Core Special Function Registers
DGPR	Data General Purpose Register
DMU	Data Memory Unit
EBU	External Bus Unit
FPI	Flexible Peripheral Interconnect (Bus)
GPR	General Purpose Register
GPTA	General Purpose Timer Array
GPTU	General Purpose Timer Unit
ICACHE	Instruction Cache
I/O	Input / Output
NMI	Non-Maskable Interrupt
OCDS	On-Chip Debug Support
OVRAM	Code Overlay Memory
PCP	Peripheral Control Processor
PMU	Program Memory Unit
PLL	Phase Locked Loop
PCODE	PCP Code Memory
PMU	Program Memory Unit
PRAM	PCP Parameter RAM
RAM	Random Access Memory
RISC	Reduced Instruction Set Computing
RTC	Real Time Clock
SCU	System Control Unit
SDLM	Serial Data Link Module (J1850)
SFR	Special Function Register
SPRAM	Scratch-Pad Code Memory
SRAM	Static Data Memory
SSC	Synchronous Serial Controller
STM	System Timer
WDT	Watchdog Timer

1.2 Peripheral Units of the TC1775

The TC1775 microcontroller offers several versatile on-chip peripheral units such as serial controllers, timer units, and Analog-to-Digital converters. Within the TC1775, all these peripheral units are connected to the TriCore CPU/system via the FPI (Flexible Peripheral Interconnect) Bus. Several I/O lines on the TC1775 ports are reserved for these peripheral units to communicate with the external world.

The following peripherals are all described in detail in the chapters of this “TC1775 Peripheral Units’ User’s Manual”:

Peripheral Units of the TC1775:

- Two Asynchronous/Synchronous Serial Channels with baud rate generator, parity, framing, and overrun error detection
- Two High Speed Synchronous Serial Channels with programmable data length and shift direction
- TwinCAN Module with two interconnected CAN nodes for high efficiency data handling via FIFO buffering and gateway data transfer
- Serial Data Link Module compliant to SAE Class B J1850 Specification
- Multifunctional General Purpose Timer Unit with three 32-bit timer/counter
- General Purpose Timer Array with a powerful set of digital signal filtering and timer functionality to realize autonomous and complex Input/Output management
- Two Analog-to-Digital Converter Units with 8-bit, 10-bit, or 12-bit resolution and sixteen analog inputs each

The next sections within this chapter provide an overview of these peripheral units.

1.2.1 Serial Interfaces

The TC1775 includes six serial peripheral interface units:

- Two Asynchronous/Synchronous Serial Interfaces (ASC0 and ASC1)
- Two High-Speed Synchronous Serial Interfaces (SSC0 and SSC1)
- One TwinCAN Interface
- One J1850 Serial Data Link Interface (SDLM)

1.2.1.1 Asynchronous/Synchronous Serial Interfaces

Figure 1-1 shows a global view of the functional blocks of the two Asynchronous/ Synchronous Serial interfaces.

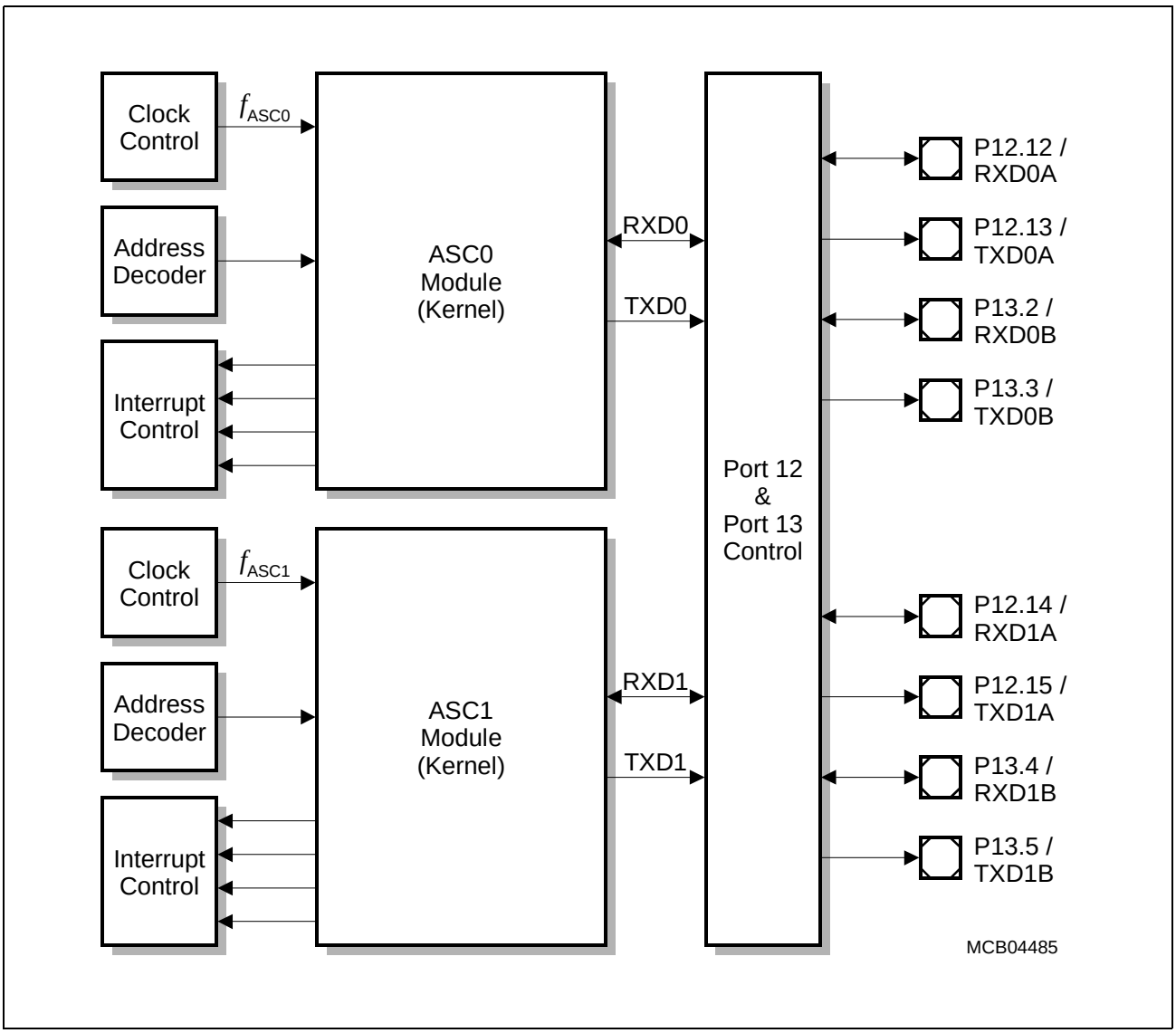


Figure 1-1 General Block Diagram of the ASC Interfaces

Each ASC Module, ASC0 and ASC1, communicates with the external world via two pairs of two I/O lines each. The RXD line is the receive data input signal (in Synchronous Mode also output). TXD is the transmit output signal. Clock control, address decoding, and interrupt service request control are managed outside the ASC Module kernel.

The Asynchronous/Synchronous Serial Interfaces provide serial communication between the TC1775 and other microcontrollers, microprocessors, or external peripherals.

The ASC supports full-duplex asynchronous communication and half-duplex synchronous communication. In Synchronous Mode, data is transmitted or received synchronous to a shift clock which is generated by the ASC internally. In Asynchronous Mode, 8-bit or 9-bit data transfer, parity generation, and the number of stop bits can be selected. Parity, framing, and overrun error detection are provided to increase the reliability of data transfers. Transmission and reception of data are double-buffered. For multiprocessor communication, a mechanism is included to distinguish address bytes from data bytes. Testing is supported by a loop-back option. A 13-bit baud rate generator provides the ASC with a separate serial clock signal that can be very accurately adjusted by a prescaler implemented as a fractional divider.

Features:

- Full-duplex asynchronous operating modes
  - 8- or 9-bit data frames, LSB first
  - Parity bit generation/checking
  - One or two stop bits
  - Baud rate from 2.5 MBaud to 0.6 Baud (@ 40 MHz clock)
  - Multiprocessor Mode for automatic address/data byte detection
  - Loop-back capability
- Half-duplex 8-bit synchronous operating mode
  - Baud rate from 5 MBaud to 406.9 Baud (@ 40 MHz clock)
- Double buffered transmitter/receiver
- Interrupt generation
  - On a transmitter buffer empty condition
  - On a transmit last bit of a frame condition
  - On a receiver buffer full condition
  - On an error condition (frame, parity, overrun error)
- Two-pin pairs RXD/TXD for each ASC available at Port 12 or Port 13

1.2.1.2 High-Speed Synchronous Serial Interfaces

Figure 1-2 shows a global view of the functional blocks of the two High-Speed Synchronous Serial interfaces (SSC).

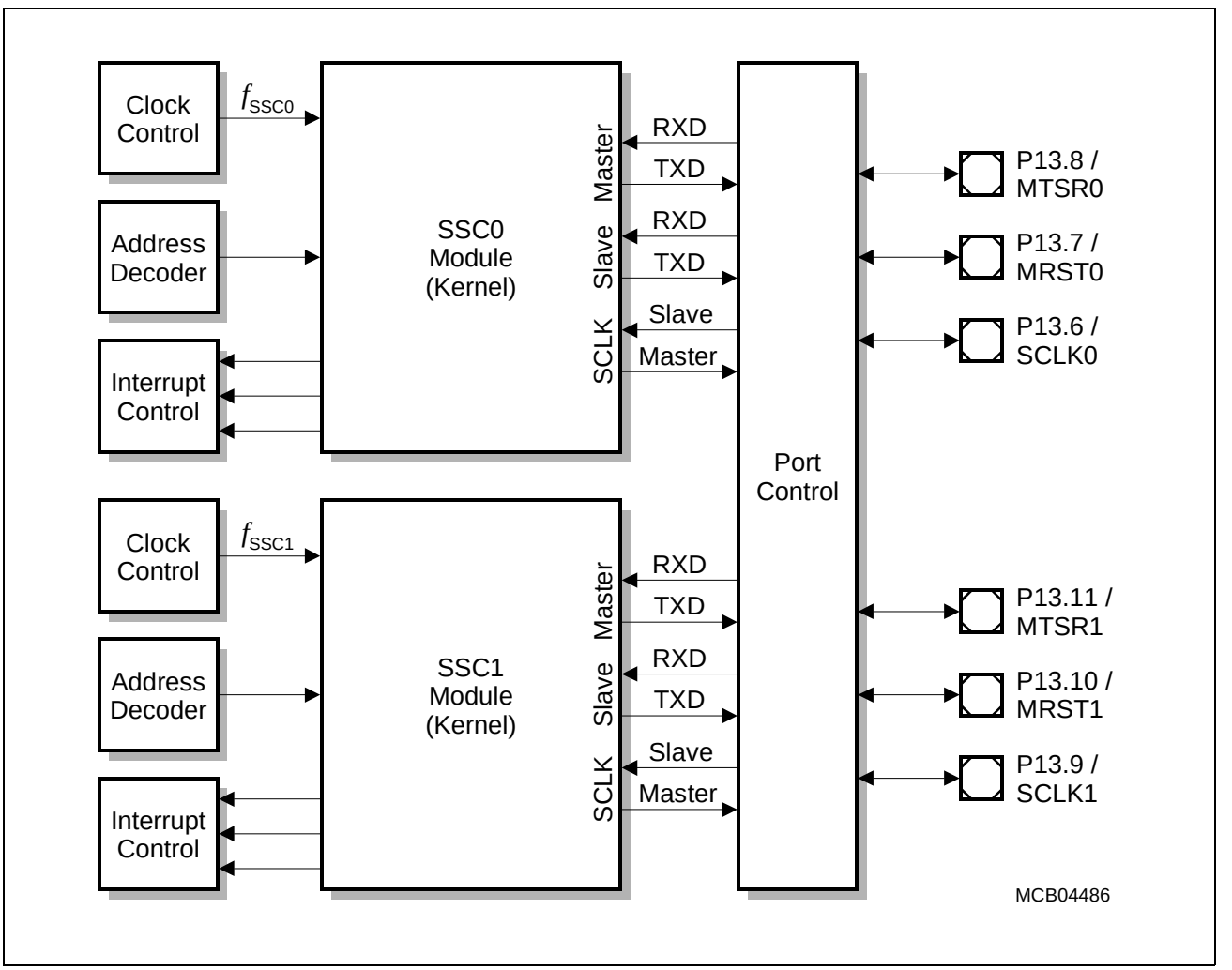


Figure 1-2 General Block Diagram of the SSC Interfaces

Each of the SSC Modules has three I/O lines, located at Port 13. Each of the SSC Modules is further supplied by separate clock control, interrupt control, address decoding, and port control logic.

The SSC supports full-duplex and half-duplex serial synchronous communication up to 20 MBaud (@ 40 MHz module clock). The serial clock signal can be generated by the SSC itself (master mode) or can be received from an external master (slave mode). Data width, shift direction, clock polarity, and phase are programmable. This allows communication with SPI-compatible devices. Transmission and reception of data are double-buffered. A 16-bit baud rate generator provides the SSC with a separate serial clock signal.

Features:

- Master and slave mode operation
  - Full-duplex or half-duplex operation
- Flexible data format
  - Programmable number of data bits: 2-bit to 16-bit
  - Programmable shift direction: LSB or MSB shift first
  - Programmable clock polarity: idle low or high state for the shift clock
  - Programmable clock/data phase data shift with leading or trailing edge of the shift clock
- Baud rate generation from 20 MBaud to 305.18 Baud (@ 40 MHz module clock)
- Interrupt generation
  - On a transmitter empty condition
  - On a receiver full condition
  - On an error condition (receive, phase, baud rate, transmit error)
- Three-pin interface
  - Flexible SSC pin configuration

1.2.1.3 TwinCAN Interface

Figure 1-3 shows a global view of the functional blocks of the TwinCAN Module.

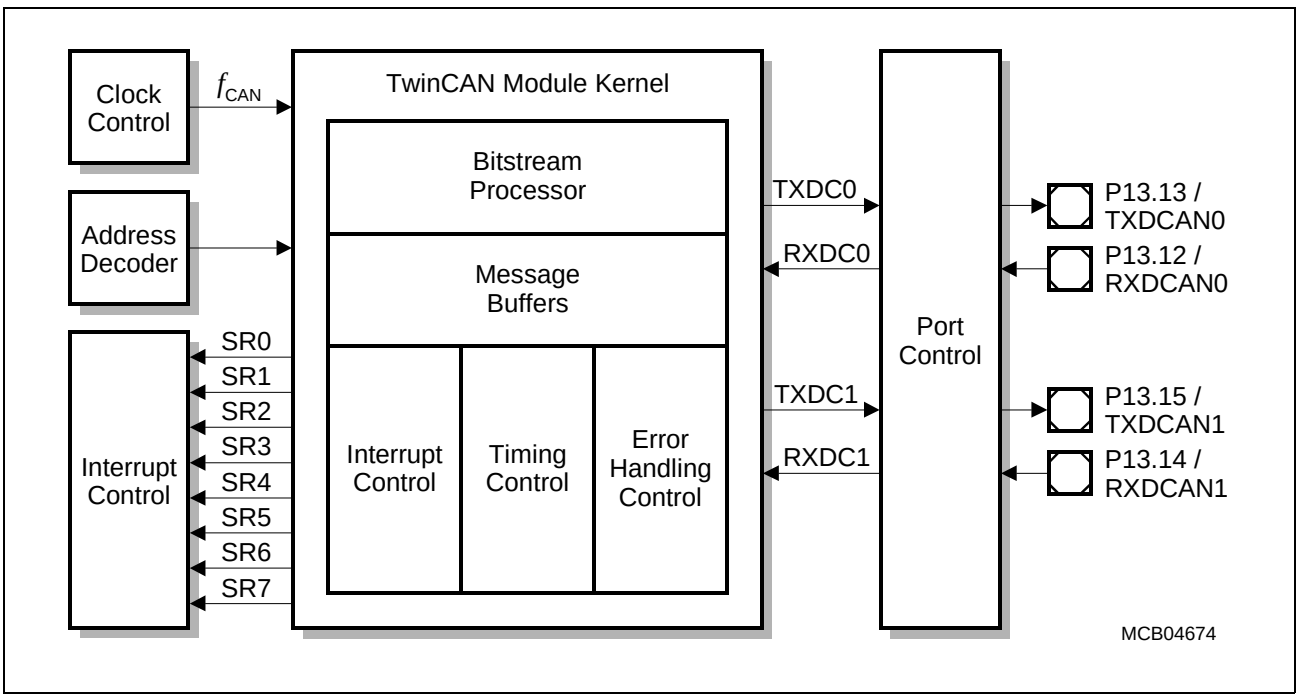


Figure 1-3 General Block Diagram of the TwinCAN Interfaces

The TwinCAN Module has four I/O lines located at Port 13. The TwinCAN Module is further supplied by a clock control, interrupt control, address decoding, and port control logic.

The TwinCAN Module combines two Full-CAN interfaces into one module. Each Full-CAN interface can either operate independently or share the TwinCAN module's resources. Transmission and reception of CAN frames is handled autonomously in accordance to CAN specification V2.0 part B (active). Each of the two Full-CAN interfaces can receive and transmit standard frames with 11-bit identifiers as well as extended frames with 29-bit identifiers.

Both CAN nodes share the TwinCAN module's resources to optimize the CAN bus traffic handling as well as to minimize the CPU load. The flexible combination of Full-CAN functionality and FIFO architecture reduces the efforts to fulfill the real-time requirements of complex embedded control applications. Improved CAN bus monitoring functionality as well as the increased number of message objects permit precise and comfortable CAN bus traffic handling.

Depending on the application, each of the 32 message objects can be individually assigned to one of the two CAN nodes. Gateway functionality allows automatic data exchange between two separate CAN bus systems, which decreases CPU load and improves the real time behavior of the entire system.

The bit timings for both 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 connect each CAN node to a bus transceiver.

Features:

- Full CAN functionality compliant with 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
- Full-CAN functionality: 32 message objects can be individually:
  - Assigned to one of the two CAN nodes
  - Configured as transmit or receive objects
  - Participate in a 2, 4, 8, 16 or 32 message buffer with FIFO algorithm
  - Setup 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
- Up to eight individually programmable interrupt nodes can be used
- CAN Analyzer Mode for bus monitoring is implemented

1.2.1.4 Serial Data Link Interface

Figure 1-4 shows a global view of the functional blocks of the Serial Data Link Interface (SDLM).

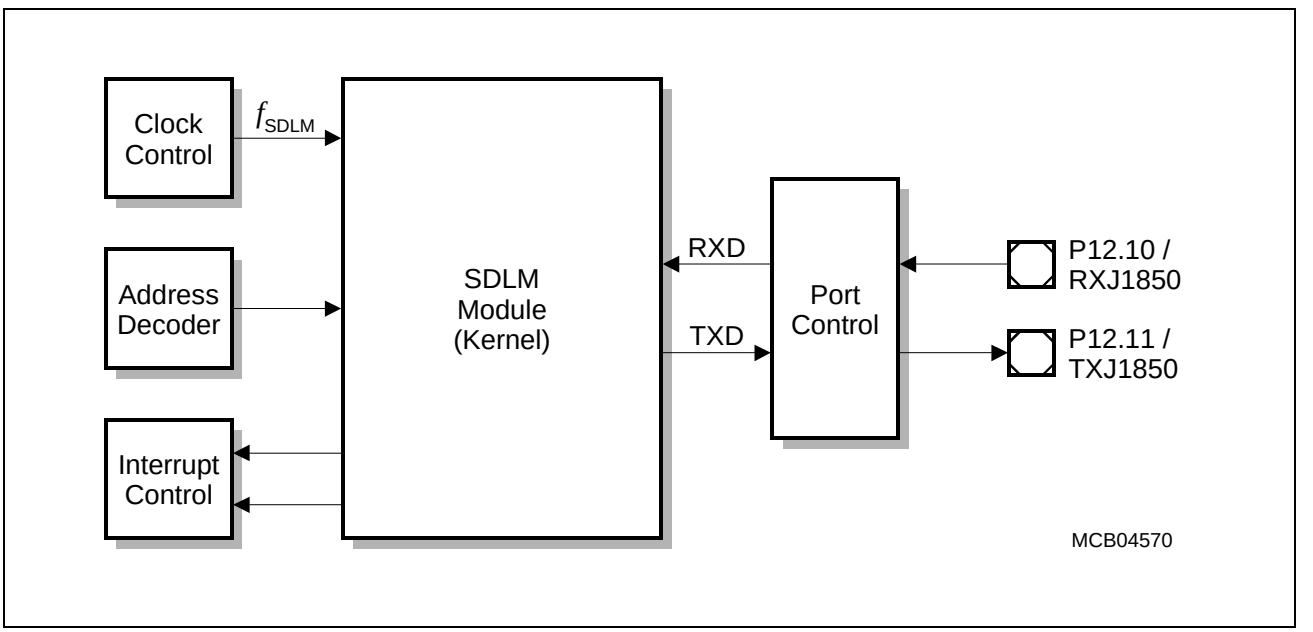


Figure 1-4 General Block Diagram of the SDLM Interface

The SDLM Module communicates with the external world via two I/O lines located at Port 12, the J1850 bus. The RXD line is the receive data input signal and TXD is the transmit data output signal.

The Serial Data Link Module provides serial communication to a J1850 based serial bus. J1850 bus transceivers must be implemented externally in a system. The SDLM Module conforms to the SAE Class B J1850 Specification and is compatible to Class 2 protocol.

General SDLM Features:

- Compliant to SAE Class B J1850 Specification
- Full support of GM Class 2 protocol
- Variable Pulse Width (VPW) format with 10.4 kBaud
- High speed receive/transmit 4x mode with 41.6 kBaud
- Digital noise filter
- Power save mode and automatic wake-up upon bus activity
- Support of single byte headers or consolidated headers
- CRC generation & check
- Support of block mode for receive and transmit

Data Link Operation Features:

- 11-byte transmit buffer
- Double buffered 11-byte receive buffer
- Support of In-Frame Response (IFR) types 1, 2, 3
- Advanced interrupt handling for RX, TX, and error conditions
- All interrupt sources can be enabled/disabled individually
- Support of automatic IFR Transmission for IFR types 1 and 2 for 3-byte consolidated headers

Note: The J1850 module does not support the Pulse Width Modulation (PWM) data format.

1.2.2 Timer Units

The TC1775 includes two timer units:

– General Purpose Timer Unit (GPTU)
– General Purpose Timer Array (GPTA)

1.2.2.1 General Purpose Timer Unit

Figure 1-5 shows a global view of all functional blocks of the General Purpose Timer Unit (GPTU) Module.

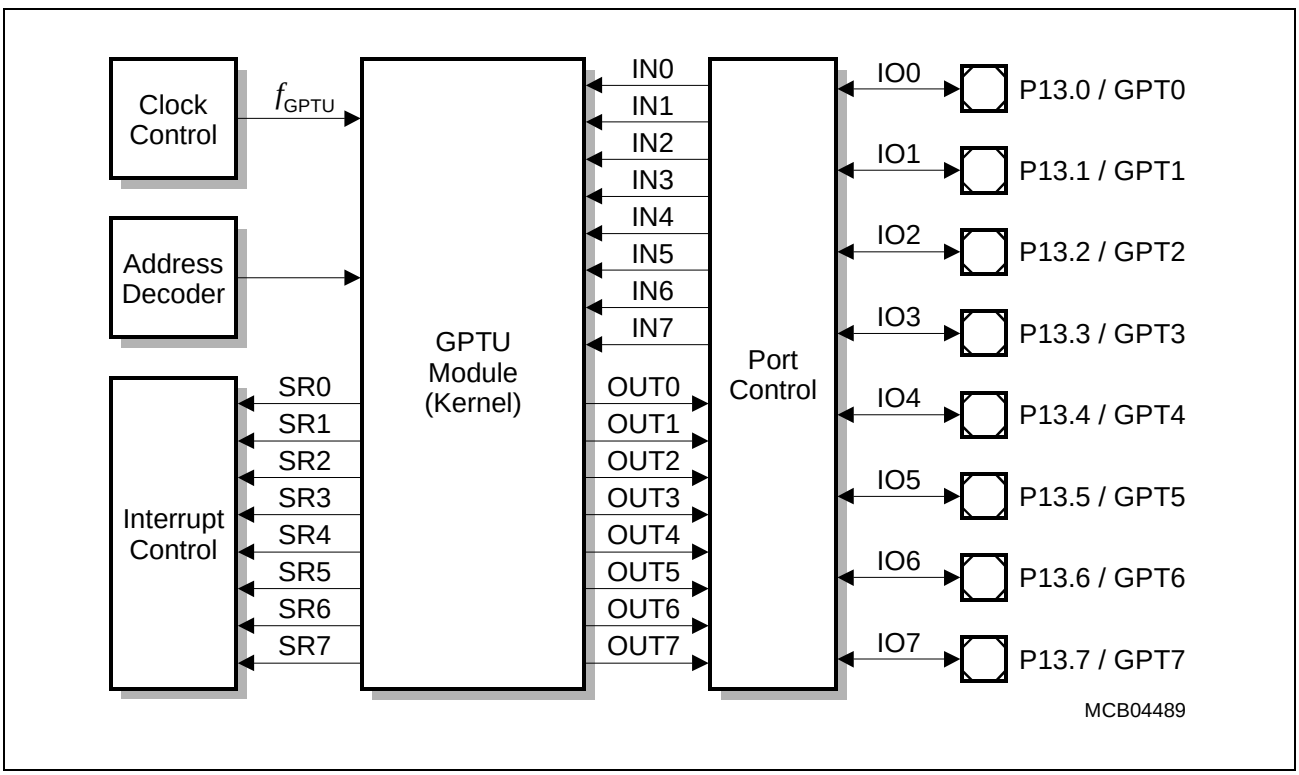


Figure 1-5 General Block Diagram of the GPTU Interface

The GPTU consists of three 32-bit timers designed to solve such application tasks as event timing, event counting, and event recording. The GPTU communicates with the external world via eight inputs and eight outputs located at Port 13.

The three timers of the GPTU Module T0, T1, and T2, can operate independently from each other or can be combined:

General Features:

- All timers are 32-bit precision timers with a maximum input frequency of f_{GPTU}
- Events generated in T0 or T1 can be used to trigger actions in T2
- Timer overflow or underflow in T2 can be used to clock either T0 or T1
- T0 and T1 can be concatenated to form one 64-bit timer

Features of T0 and T1:

- Each timer has a dedicated 32-bit reload register with automatic reload on overflow
- Timers can be split into individual 8-, 16-, or 24-bit timers with individual reload registers
- Overflow signals can be selected to generate service requests, pin output signals, and T2 trigger events
- Two input pins can determine a count option

Features of T2:

- Count up or down is selectable
- Operating modes:
  - Timer
  - Counter
  - Quadrature counter (incremental/phase encoded counter interface)
- Options:
  - External start/stop, one-shot operation, timer clear on external event
  - Count direction control through software or an external event
  - Two 32-bit reload/capture registers
- Reload modes:
  - Reload on overflow or underflow
  - Reload on external event: positive transition, negative transition, or both transitions
- Capture modes:
  - Capture on external event: positive transition, negative transition, or both transitions
  - Capture and clear timer on external event: positive transition, negative transition, or both transitions
- Can be split into two 16-bit counter/timers
- Timer count, reload, capture, and trigger functions can be assigned to input pins. T0 and T1 overflow events can also be assigned to these functions
- Overflow and underflow signals can be used to trigger T0 and/or T1 and to toggle output pins
- T2 events are freely assignable to the service request nodes

1.2.2.2 General Purpose Timer Array

Figure 1-6 shows a global block diagram of the General Purpose Timer Array (GPTA) implementation.

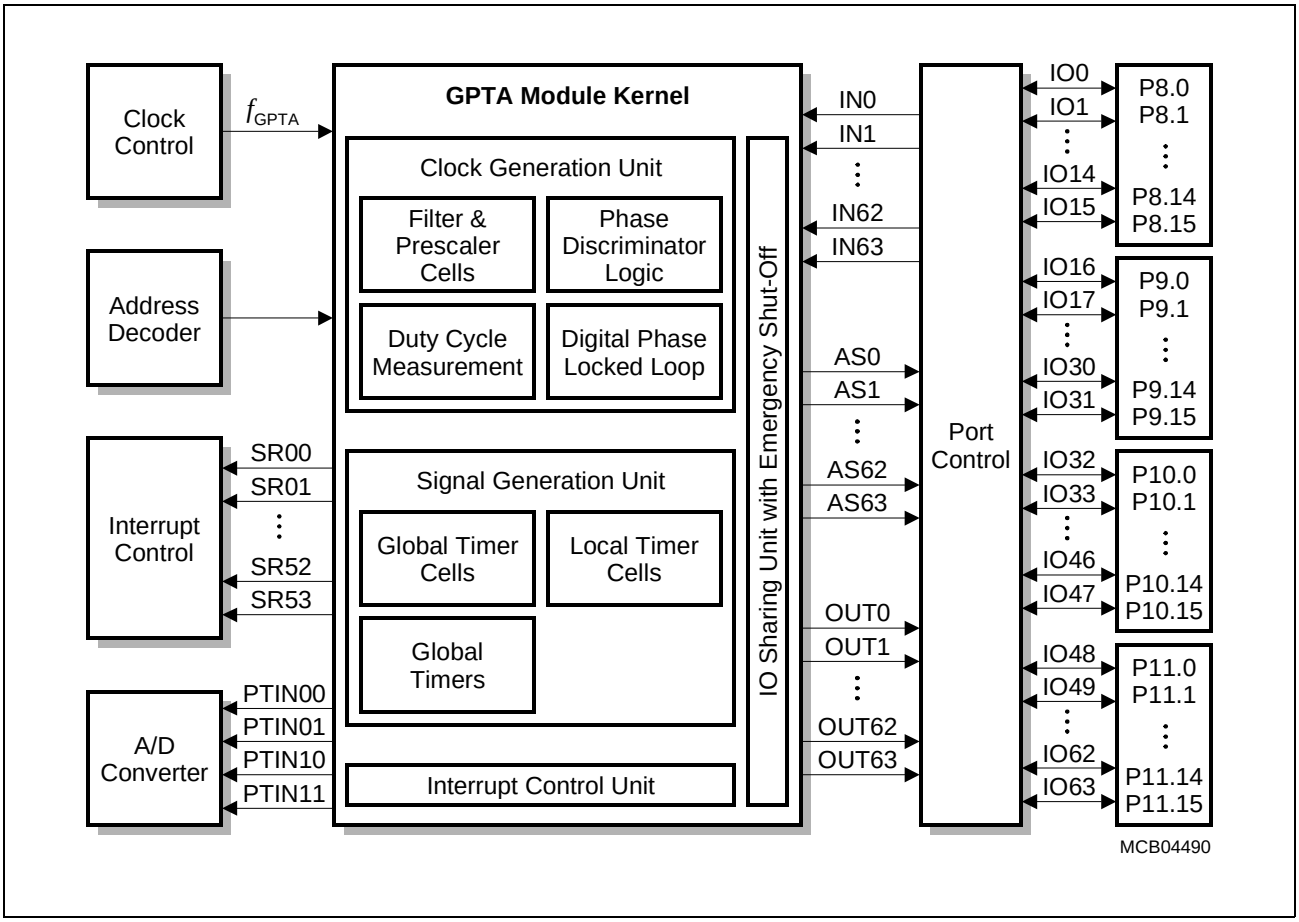


Figure 1-6 GPTA Module Kernel Block Diagram

The GPTA Module has 64 input lines and 64 output lines which are connected with Port 8, Port 9, Port 10, and Port 11.

The General Purpose Timer Array (GPTA) provides important digital signal filtering and timer support whose combination enables autonomous and complex functionalities. This architecture allows easy implementation and easy validation of any kind of timer functions.

The GPTA provides a set of hardware modules required for high speed digital signal processing:

- Filter and Prescaler Cells (FPC) support input noise filtering and prescaler operation.
- Phase Discrimination Logic units (PDL) decode the direction information output by a rotation tracking system.
- Duty Cycle Measurement Cells (DCM) provide pulse width measurement capabilities.
- Digital Phase Locked Loop unit (PLL) generates a programmable number of GPTA module clock ticks during an input signal's period.
- Global Timer units (GT) driven by various clock sources are implemented to operate as a time base for the associated "Global Timer Cells".
- Global Timer Cells (GTC) can be programmed to capture the contents of a Global Timer on an event occurring at an external port pin or at an internal FPC output. A GTC may be also used to control an external port pin with the result of an internal compare operation. GTCs can be logically concatenated to provide a common external port pin with a complex signal waveform.
- Local Timer Cells (LTC) operating in timer, capture, or compare mode may be also tied together logically to drive a common external port pin with a complex signal waveform. LTCs, enabled in timer or capture mode, can be clocked or triggered by
  - a prescaled GPTA module clock,
  - an FPC, PDL, DCM, PLL or GTC output signal line,
  - an external port pin.

Some input lines driven by processor I/O pads may be shared by a LTC and a GTC to trigger their programmed operation simultaneously.

The following list summarizes all blocks supported:

Clock Generation Unit:

- Filter and Prescaler Cell (FPC):
  - Six independent units
  - Three operating modes (Prescaler, Delayed Debounce Filter, Immediate Debounce Filter)
  - f_{GPTA} down-scaling capability
  - $f_{GPTA}/2$ maximum input signal frequency in Filter Mode
- Phase Discriminator Logic (PDL):
  - Two independent units
  - Two operating modes (2 and 3 sensor signals)
  - $f_{GPTA}/4$ maximum input signal frequency in 2-sensor mode, $f_{GPTA}/6$ maximum input signal frequency in 3-sensor mode
- Duty Cycle Measurement (DCM):
  - Four independent units
  - 0 - 100% margin and time-out handling
  - f_{GPTA} maximum resolution

  - f_{GPTA}/2 maximum input signal frequency
- Digital Phase Locked Loop (PLL):
  - One unit
  - Arbitrary multiplication factor between 1 and 65535
  - f_{GPTA} maximum resolution
  - f_{GPTA}/2 maximum input signal frequency

Signal Generation Unit:

- Global Timers (GT):
  - Two independent units
  - Two operating modes (Free Running Timer and Reload Timer)
  - 24-bit data width
  - f_{GPTA} maximum resolution
  - f_{GPTA}/2 maximum input signal frequency
- Global Timer Cell (GTC):
  - 32 independent units
  - Two operating modes (Capture, Compare and Capture after Compare)
  - 24-bit data width
  - f_{GPTA} maximum resolution
  - f_{GPTA}/2 maximum input signal frequency
- Local Timer Cell (LTC):
  - 64 independent units
  - Three operating modes (Timer, Capture and Compare)
  - 16-bit data width
  - f_{GPTA} maximum resolution
  - f_{GPTA}/2 maximum input signal frequency

Interrupt Control Unit:

- 111 interrupt sources generating 54 service requests

I/O Sharing Unit:

- Able to process lines from FPC, GTC and LTC
- Emergency function

1.2.3 Analog-to-Digital Converters

The two on-chip Analog-to-Digital Converter (ADC) Modules of the TC1775 offer 8-bit, 10-bit, or 12-bit resolution including sample-and-hold functionality. The A/D converters operate by the method of the successive approximation. A multiplexer selects among up to 16 analog input channels for each ADC. Conversion requests are generated either under software control or by hardware. An automatic self-calibration adjusts the ADC modules to changing temperatures or process variations.

Features:

The following functions have been implemented in the two on-chip ADC Modules to fulfill the enhanced requirements of embedded control applications:

- 8-bit, 10-bit, 12-bit A/D conversion
- Successive approximation conversion method
- Total Unadjusted Error (TUE) of ± 2 LSB @ 10-bit resolution
- Integrated sample-and-hold functionality
- Sixteen analog input channels
- Dedicated control and status registers for each analog channel
- Flexible conversion request mechanisms
- Selectable reference voltages for each channel
- Programmable sample and conversion timing schemes
- Limit checking
- Broken wire - short circuit detection
- Flexible ADC Module service request control unit
- Synchronization of the two on-chip A/D Converters
- Automatic control of external analog multiplexer
- Equidistant samples initiated by timer
- External trigger inputs for conversion requests
- Two external trigger inputs, connected with the General Purpose Timer Array (GPTA)
- Power reduction and clock control feature

Figure 1-7 shows a global view of the ADC Module kernel with the module specific interface connections.

Each of the ADC Modules communicates with the external world via five digital I/O lines and sixteen analog inputs. Clock control, address decoding, and interrupt service request control are managed outside the ADC Module kernel. Two trigger inputs and a synchronization bridge are used for internal control purposes.

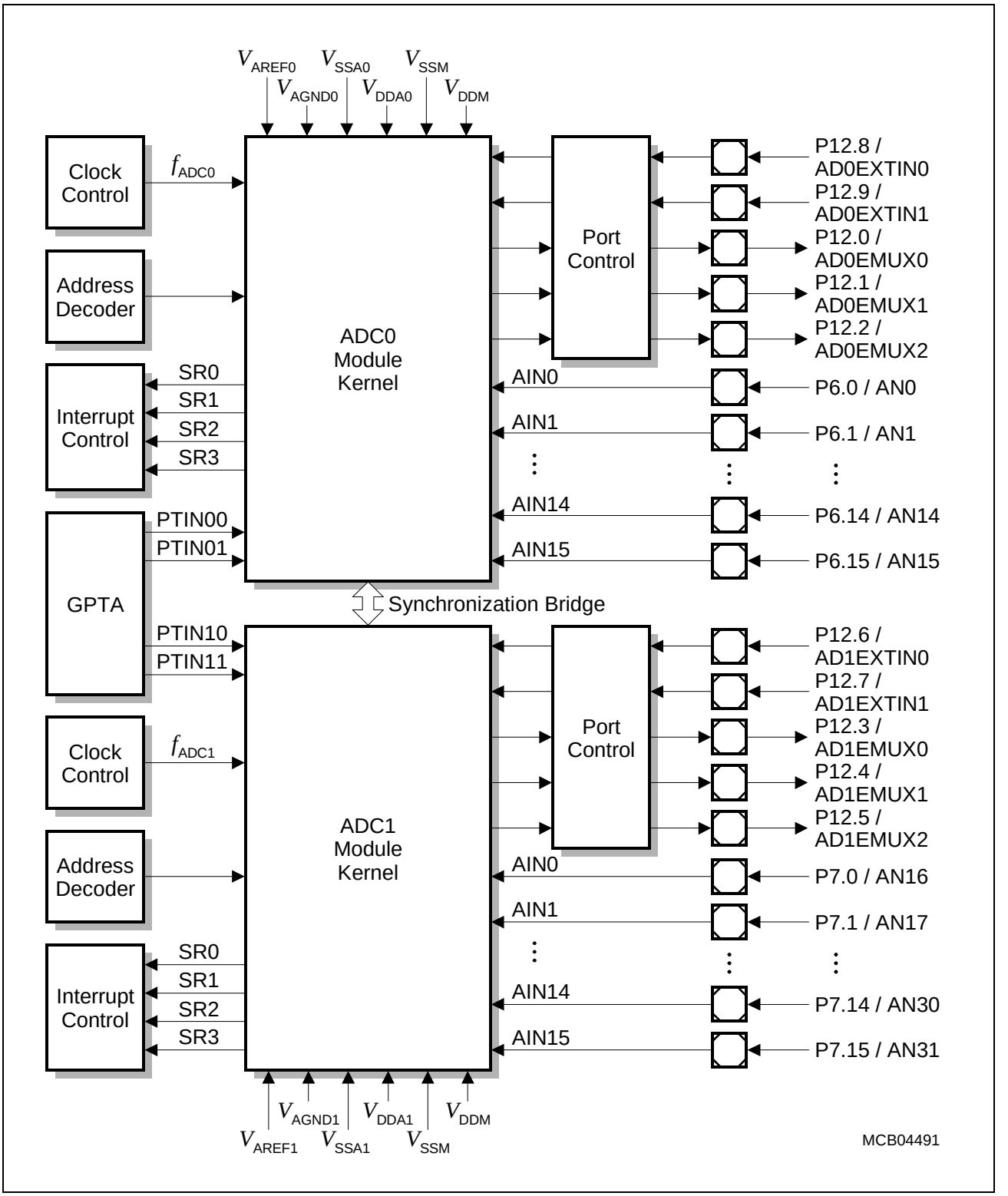


Figure 1-7 General Block Diagram of the ADC Interface

2 Asynchronous/Synchronous Serial Interface (ASC)

This chapter describes the two ASC asynchronous/synchronous serial interfaces ASC0 and ASC1 of the TC1775. It contains the following sections:

- Functional description of the ASC Kernel, valid for ASC0 and ASC1 (see **Section 2.1**)
- ASC kernel register description, describes all ASC Kernel specific registers (see **Section 2.2**)
- TC1775 implementation specific details and registers of the ASC0/ASC1 modules (port connections and control, interrupt control, address decoding, clock control, see **Section 2.3**).

*Note: The ASC kernel register names described in **Section 2.2** will be referenced in the TC1775 User's Manual by the module name prefix "ASC0_" for the ASC0 interface and by "ASC1_" for the ASC1 interface.*

2.1 ASC Kernel Description

Figure 2-1 shows a global view of all functional blocks of the ASC interface.

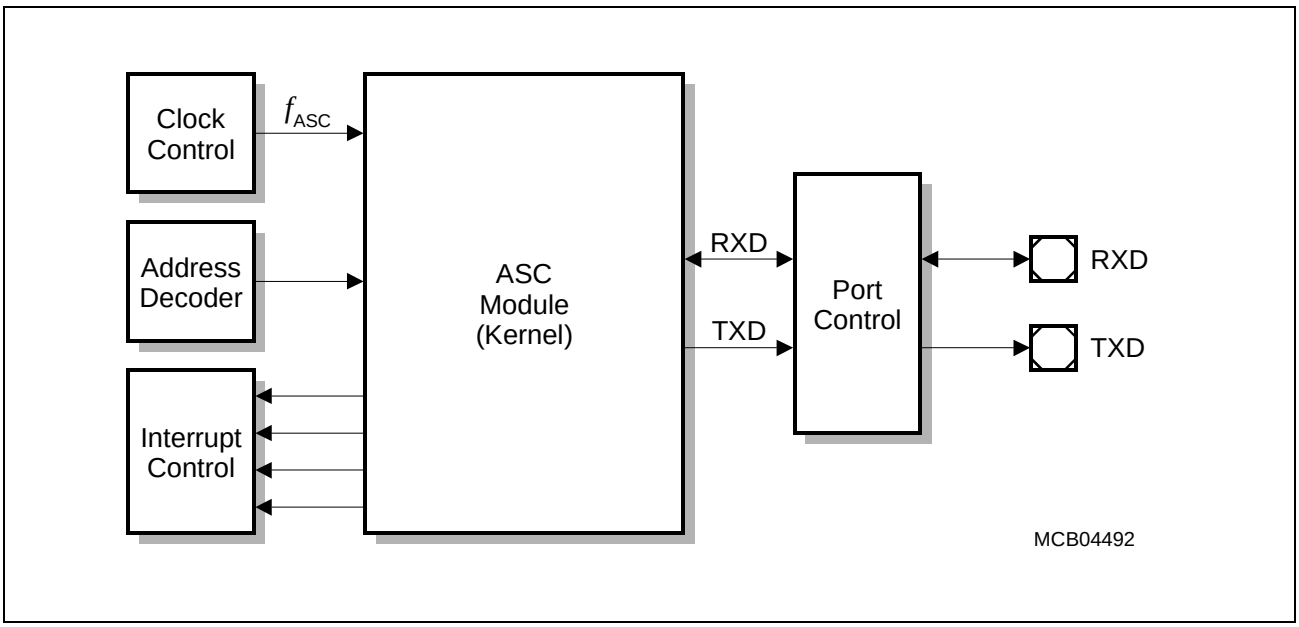


Figure 2-1 General Block Diagram of the ASC Interface

The ASC module communicates with the external world via two I/O lines. The RXD line is the receive data input signal (in synchronous mode also output), and TXD is the transmit output signal.

Clock control, address decoding, and interrupt service request control are managed outside the ASC Module kernel.

2.1.1 Overview

The Asynchronous/Synchronous Serial Interfaces provide serial communication between the TC1775 and other microcontrollers, microprocessors or external peripherals.

The ASC supports full-duplex asynchronous communication and half-duplex synchronous communication. In Synchronous Mode, data is transmitted or received synchronous to a shift clock generated by the ASC internally. In Asynchronous Mode, 8-bit or 9-bit data transfer, parity generation and the number of stop bits can be selected. Parity, framing and overrun error detection are provided to increase the reliability of data transfers. Transmission and reception of data are double-buffered. For multiprocessor communication, a mechanism is included to distinguish address bytes from data bytes. Testing is supported by a loop-back option. A 13-bit baud rate generator provides the ASC with a separate serial clock signal, which can be very accurately adjusted by a prescaler implemented as fractional divider.

Features:

- Full-duplex asynchronous operating modes
  - 8-bit or 9-bit data frames, LSB first
  - Parity bit generation/checking
  - One or two stop bits
  - Baud rate from 2.5 MBaud to 0.6 Baud (@ 40 MHz clock)
  - Multiprocessor mode for automatic address/data byte detection
  - Loop-back capability
- Half-duplex 8-bit synchronous operating mode
  - Baud rate from 5 MBaud to 406.9 Baud (@ 40 MHz clock)
- Double buffered transmitter/receiver
- Interrupt generation
  - On a transmitter buffer empty condition
  - On a transmit last bit of a frame condition
  - On a receiver buffer full condition
  - On an error condition (frame, parity, overrun error)

2.1.2 General Operation

The ASC supports full-duplex asynchronous communication up to 2.5 MBaud and half-duplex synchronous communication up to 5 MBaud (@ 40 MHz module clock). In Synchronous Mode, data are transmitted or received synchronous to a shift clock generated by the microcontroller. In Asynchronous Mode, 8-bit or 9-bit data transfer, parity generation, and the number of stop bits can be selected. Parity, framing, and overrun error detection are provided to increase the reliability of data transfers. Transmission and reception of data are double-buffered. For multiprocessor communication, a mechanism is included to distinguish address bytes from data bytes. Testing is supported by a loop-back option. A 13-bit baud rate timer with a versatile input clock divider circuitry provides the ASC with the serial clock signal.

A transmission is started by writing to the Transmit Buffer register TBUF. Only the number of data bits determined by the selected operating mode will actually be transmitted, that is, bits written to positions 9 through 15 of register TBUF are always insignificant.
Data transmission is double-buffered, so a new character may be written to the transmit buffer register, before the transmission of the previous character is complete. This allows back-to-back transmission of characters without gaps.

Data reception is enabled by the Receiver Enable Bit CON.REN. After reception of a character has been completed, the received data and, if provided by the selected operating mode, the received parity bit can be read from the (read-only) Receive Buffer register RBUF. Bits in the upper half of RBUF not valid in the selected operating mode will be read as zeros.
Data reception is double-buffered, so that reception of a second character may already begin before the previously received character has been read out of the receive buffer register. In all modes, receive buffer overrun error detection can be selected through bit CON.OEN. When enabled, the overrun error status flag CON.OE and the error interrupt request line EIR will be activated when the receive buffer register has not been read by the time reception of a second character is complete. The previously received character in the receive buffer is overwritten.

The Loop-Back option (selected by bit CON.LB) allows the data currently being transmitted to be received simultaneously in the receive buffer. This may be used to test serial communication routines at an early stage without having to provide an external network. In Loop-Back Mode the alternate input/output function of port pins is not required.

2.1.3 Asynchronous Operation

Asynchronous mode supports full-duplex communication, where both transmitter and receiver use the same data frame format and have the same baud rate. Data is transmitted on pin TXD and received on pin RXD. **Figure 2-2** shows the block diagram of the ASC when operating in Asynchronous Mode.

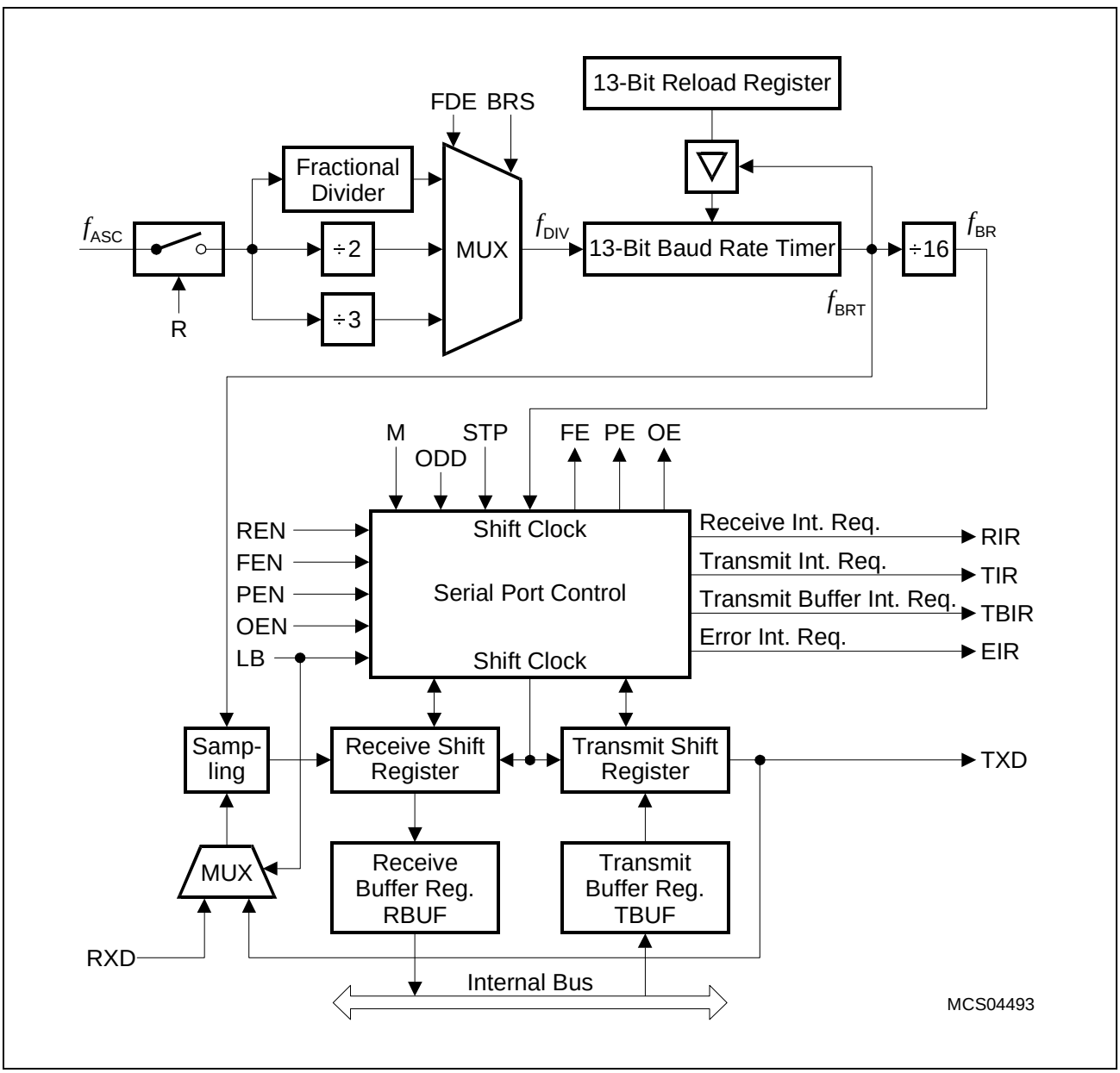


Figure 2-2 Asynchronous Mode of the ASC

2.1.3.1 Asynchronous Data Frames

8-Bit Data Frames

8-bit data frames consist of either eight data bits D7 … D0 (CON.M = 001_B), or of seven data bits D6 … D0 plus an automatically generated parity bit (CON.M = 011_B). Parity may be odd or even, depending on bit CON.ODD. An even parity bit will be set if the modulo-2-sum of the seven data bits is 1. An odd parity bit will be cleared in this case. Parity checking is enabled via bit CON.PEN (always OFF in 8-bit data mode). The parity error flag CON.PE will be set, along with the error interrupt request flag, if a wrong parity bit is received. The parity bit itself will be stored in bit RBUF.7.

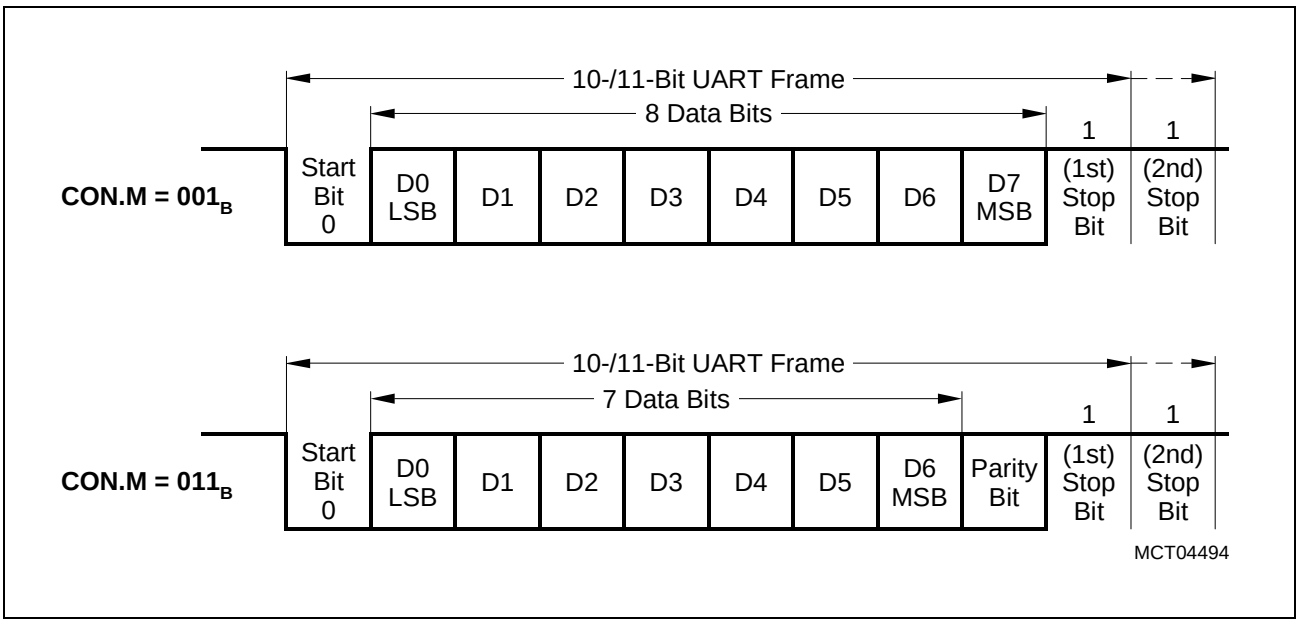


Figure 2-3 Asynchronous 8-Bit Frames

9-Bit Data Frames

9-bit data frames consist of either nine data bits D8 … D0 (CON.M = 100_B), of eight data bits D7 … D0 plus an automatically generated parity bit (CON.M = 111_B) or of eight data bits D7 … D0 plus wake-up bit (CON.M = 101_B). Parity may be odd or even, depending on bit CON.ODD. An even parity bit will be set if the modulo-2-sum of the eight data bits is 1. An odd parity bit will be cleared in this case. Parity checking is enabled via bit CON.PEN (always OFF in 9-bit data and wake-up mode). The parity error flag CON.PE will be set along with the error interrupt request flag, if a wrong parity bit is received. The parity bit itself will be stored in bit RBUF.8.

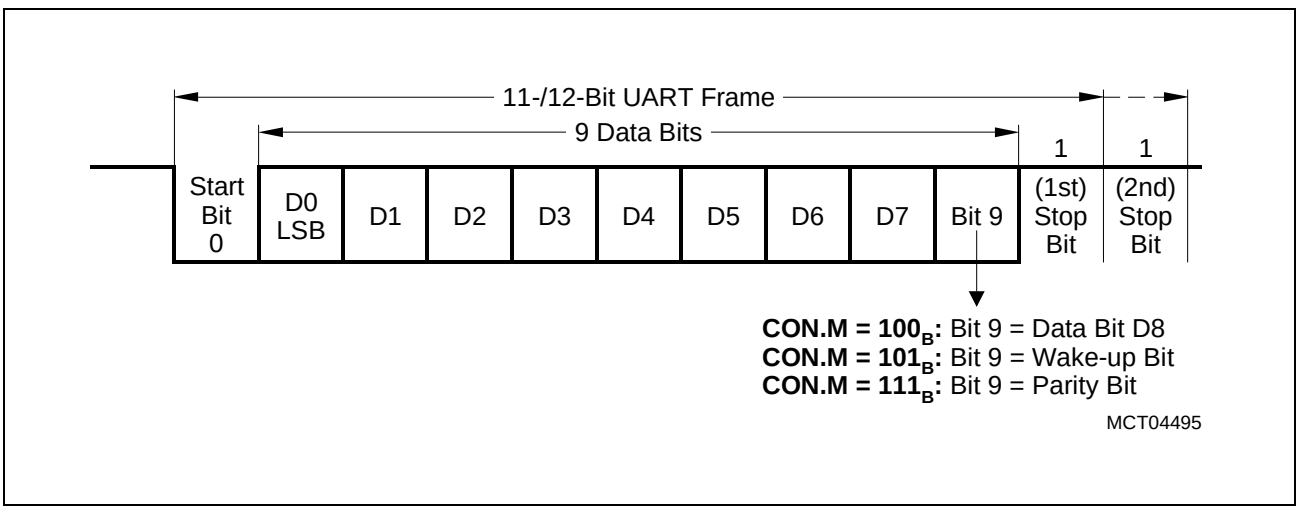


Figure 2-4 Asynchronous 9-Bit Frames

In Wake-up Mode, received frames are transferred to the receive buffer register only if the 9th bit (the wake-up bit) is 1. If this bit is 0, no receive interrupt request will be activated and no data will be transferred.

This feature may be used to control communication in multi-processor systems:
When the master processor wants to transmit a block of data to one of several slaves, it first sends out an address byte that identifies the target slave. An address byte differs from a data byte in that the additional 9th bit is a 1 for an address byte but is a 0 for a data byte, so, no slave will be interrupted by a data ‘byte’. An address ‘byte’ will interrupt all slaves (operating in 8-bit data + wake-up bit mode), so each slave can examine the eight LSBs of the received character (the address). The addressed slave will switch to 9-bit data mode (for example, by clearing bit CON.M.0), which enables it to also receive the data bytes that will be coming (having the wake-up bit cleared). The slaves that were not being addressed remain in 8-bit data + wake-up bit mode, ignoring the following data bytes.

2.1.3.2 Asynchronous Transmission

Asynchronous transmission begins at the next overflow of the divide-by-16 baud rate timer (transition of the baud rate clock f_{BR}), if bit CON.R must be set and data has been loaded into TBUF. The transmitted data frame consists of three basic elements:

- The start bit
- The data field (8 or 9 bits, LSB first, including a parity bit, if selected)
- The delimiter (1 or 2 stop bits)

Data transmission is double buffered. When the transmitter is idle, the transmit data loaded into TBUF is immediately moved to the transmit shift register; thus, freeing TBUF for the next data to be sent. This is indicated by the transmit buffer interrupt request line TBIR being activated. TBUF may now be loaded with the next data, while transmission of the previous one continues.

The transmit interrupt request line TIR will be activated before the last bit of a frame is transmitted, that is, before the first or the second stop bit is shifted out of the transmit shift register.

Note: The transmitter output pin TXD must be configured for alternate data output.

2.1.3.3 Asynchronous Reception

Asynchronous reception is initiated by a falling edge (1-to-0 transition) on pin RXD, provided that bits CON.R and CON.REN are set. The receive data input pin RXD is sampled at sixteen times the rate of the selected baud rate. A majority decision of the 7^{th}, 8^{th} and 9^{th} sample determines the effective bit value. This avoids erroneous results that may be caused by noise.

If the detected value is not a 0 when the start bit is sampled, the receive circuit is reset and waits for the next 1-to-0 transition at pin RXD. If the start bit proves valid, the receive circuit continues sampling and shifts the incoming data frame into the receive shift register.

When the last stop bit has been received, the contents of the receive shift register are transferred to the receive data buffer register RBUF. Simultaneously, the receive interrupt request line RIR is activated after the 9^{th} sample in the last stop bit timeslot (as programmed), regardless whether valid stop bits have been received or not. The receive circuit then waits for the next start bit (1-to-0 transition) at the receive data input pin.

Asynchronous reception is stopped by clearing bit CON.REN. A currently received frame is completed including generation of the receive interrupt request and an error interrupt request, if appropriate. Start bits that follow this frame will not be recognized.

Note: In wake-up mode received frames are transferred to the receive buffer register only if the 9^{th} bit (the wake-up bit) is 1. If this bit is 0, no receive interrupt request will be activated and no data will be transferred.

2.1.4 Synchronous Operation

Synchronous Mode supports half-duplex communication, basically for simple I/O expansion via shift registers. Data is transmitted and received via pin RXD while pin TXD outputs the shift clock. These signals are alternate functions of port pins. Synchronous mode is selected with CON.M = 000_B.

Eight data bits are transmitted or received synchronous to a shift clock generated by the internal baud rate generator. The shift clock is active only as long as data bits are transmitted or received.

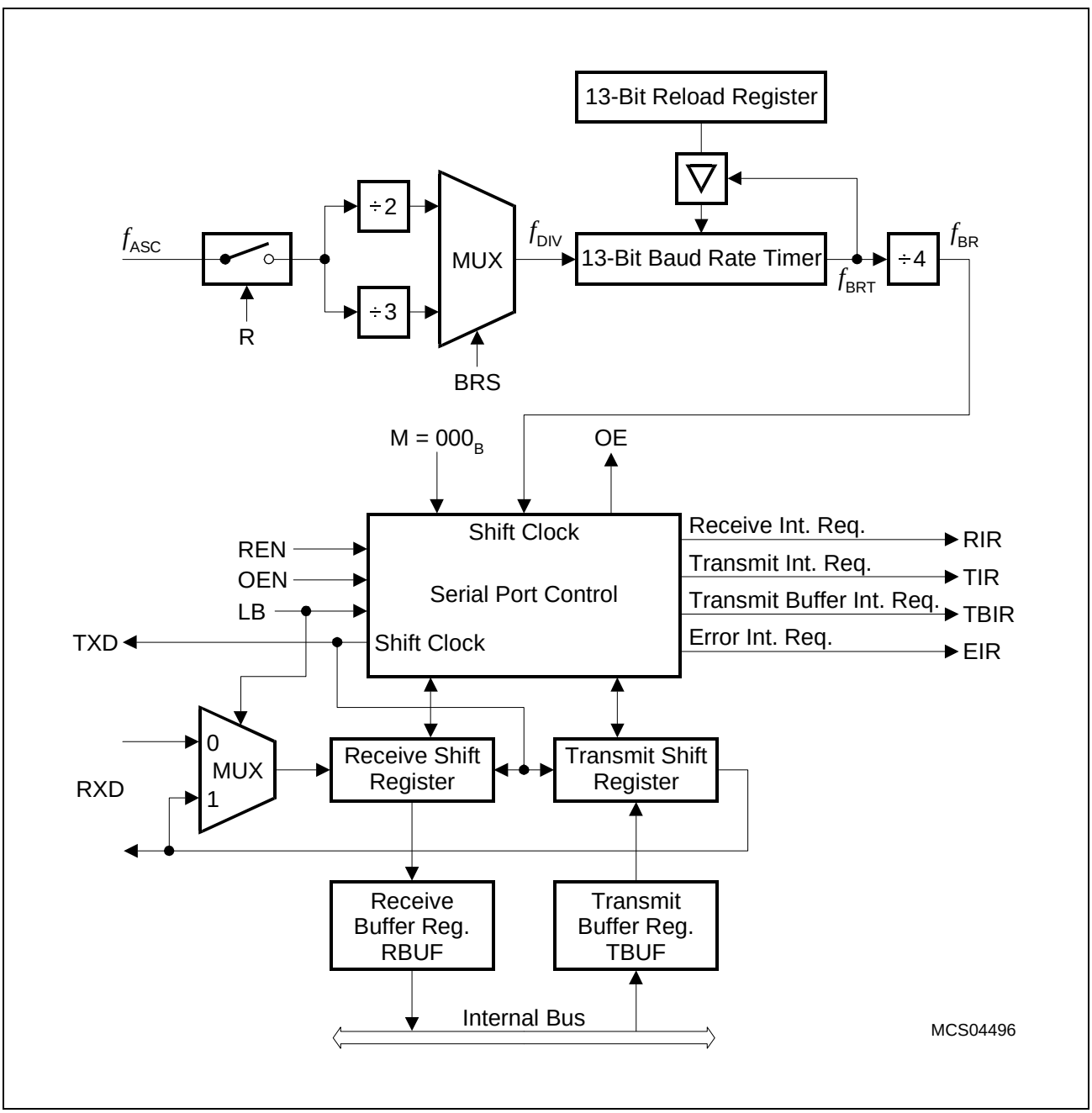


Figure 2-5 Synchronous Mode of Serial Channel ASC

2.1.4.1 Synchronous Transmission

Synchronous transmission begins within four state times after data has been loaded into TBUF, provided that CON.R is set and CON.REN = 0 (half-duplex, no reception). Exception: in Loop-back Mode (bit CON.LB set), CON.REN must be set for reception of the transmitted byte. Data transmission is double buffered. When the transmitter is idle, the transmit data loaded into TBUF is immediately moved to the transmit shift register thus, freeing TBUF for the next data to be sent. This is indicated by the transmit buffer interrupt request line TBIR being activated. TBUF may now be loaded with the next data, while transmission of the previous one continues. The data bits are transmitted synchronous with the shift clock. After the bit time for the 8^{th} data bit, both TXD and RXD will go high, the transmit interrupt request line TIR is activated, and serial data transmission stops.

Pin TXD must be configured for alternate data output in order to provide the shift clock. Pin RXD must also be configured for output during transmission.

2.1.4.2 Synchronous Reception

Synchronous reception is initiated by setting bit CON.REN = 1. If bit CON.R = 1, the data applied at RXD is clocked into the receive shift register synchronous to the clock which is output at pin TXD. After the 8^{th} bit has been shifted in, the contents of the receive shift register are transferred to the receive data buffer RBUF, the receive interrupt request line RIR is activated, the receiver enable bit CON.REN is reset, and serial data reception stops.

Pin TXD must be configured for alternate data output in order to provide the shift clock. Pin RXD must be configured as alternate data input.

Synchronous reception is stopped by clearing bit CON.REN. A currently received byte is completed, including the generation of the receive interrupt request and an error interrupt request, if appropriate. Writing to the transmit buffer register while a reception is in progress has no effect on reception and will not start a transmission.

If a previously received byte has not been read out of the receive buffer register by the time the reception of the next byte is complete, both the error interrupt request line EIR and the overrun error status flag CON.OE will be activated/set, provided that the overrun check has been enabled by bit CON.OEN.

2.1.4.3 Synchronous Timing

Figure 2-6 shows timing diagrams of the ASC Synchronous Mode data reception and data transmission. In Idle State the shift clock is at high level. With the beginning of a synchronous transmission of a data byte, the data is shifted out at RXD with the falling edge of the shift clock. If a data byte is received through RXD, data is latched with the rising edge of the shift clock.

One shift clock cycle (f_{BR}) delay is inserted between two consecutive receive or transmit data bytes.

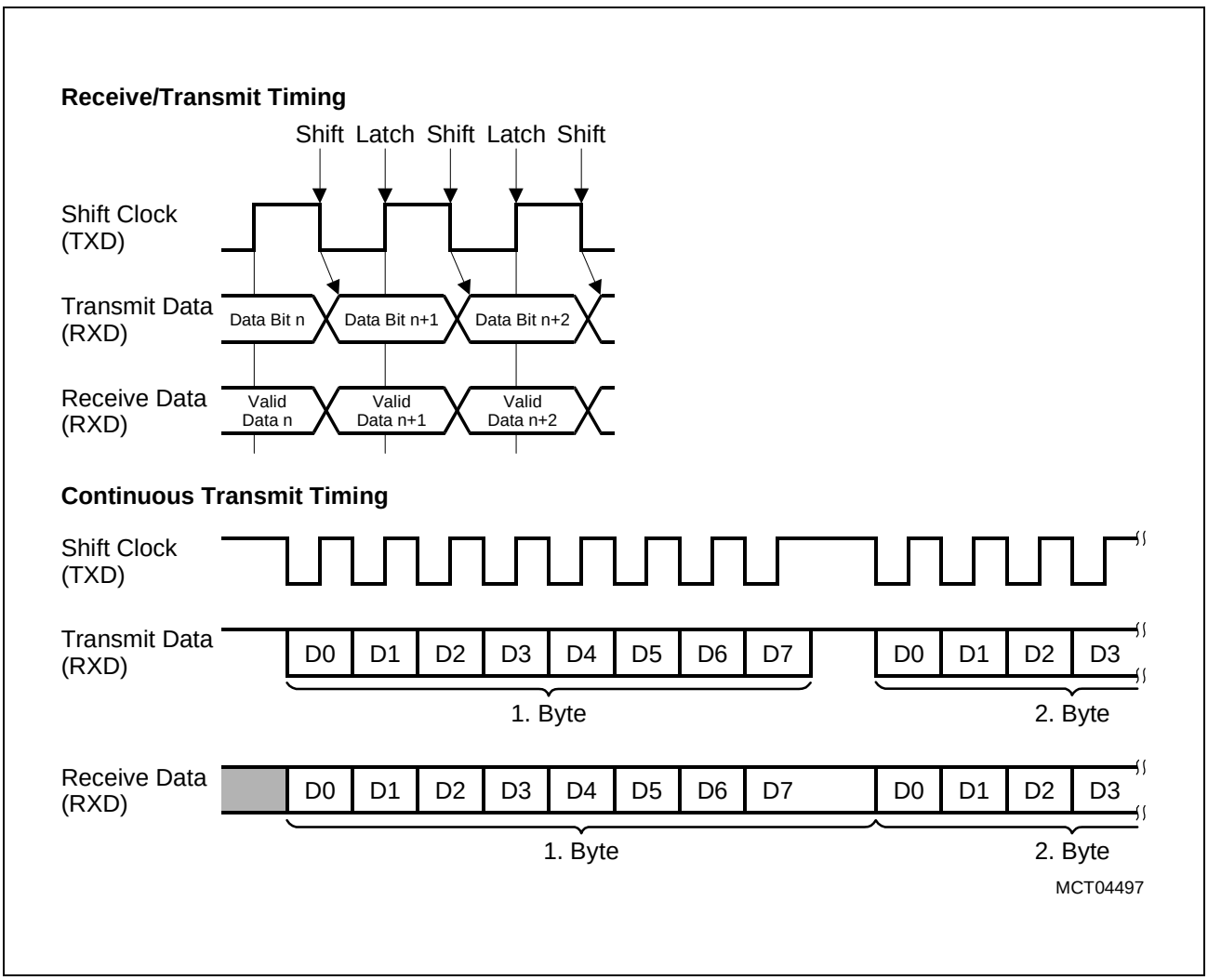


Figure 2-6 ASC Synchronous Mode Waveforms

2.1.5 Baud Rate Generation

The serial channel ASC has its own dedicated 13-bit baud rate generator with 13-bit reload capability, allowing baud rate generation independent of other timers.

The baud rate generator is clocked with a clock (f_{DIV}) which is derived via a prescaler from the ASC input clock f_{ASC}. The baud rate timer is counting downwards and can be started or stopped through the baud rate generator run bit CON.R. Each underflow of the timer provides one clock pulse to the serial channel. The timer is reloaded with the value stored in its 13-bit reload register each time it underflows. The resulting clock f_{BRT} is again divided by a factor for the baud rate clock ($\pm$ 16 in asynchronous modes and $\pm$ 4 in synchronous mode). The prescaler is selected by the bits CON.BRS and CON.FDE. In the asynchronous operating modes, a fractional divider prescaler unit is available (in addition to the two fixed dividers) which allows selection of prescaler divider ratios of n/512 with n = 0-511. Therefore, the baud rate of ASC is determined by the module clock, the content of FDV, the reload value of BG, and the operating mode (asynchronous or synchronous).

Register BG is the dual-function Baud Rate Generator/Reload register. Reading BG returns the contents of the timer BR_VALUE (bits 15 … 13 return zero), while writing to BG always updates the reload register (bits 15 … 13 are insignificant).

An auto-reload of the timer with the contents of the reload register is performed each time BG is written to. However, if CON.R = 0 at the time the write operation to BG is performed, the timer will not be reloaded until the first instruction cycle after CON.R = 1. For a clean baud rate initialization BG should only be written if CON.R = 0. If BG is written with CON.R = 1, an unpredicted behavior of the ASC may occur during running transmit or receive operations.

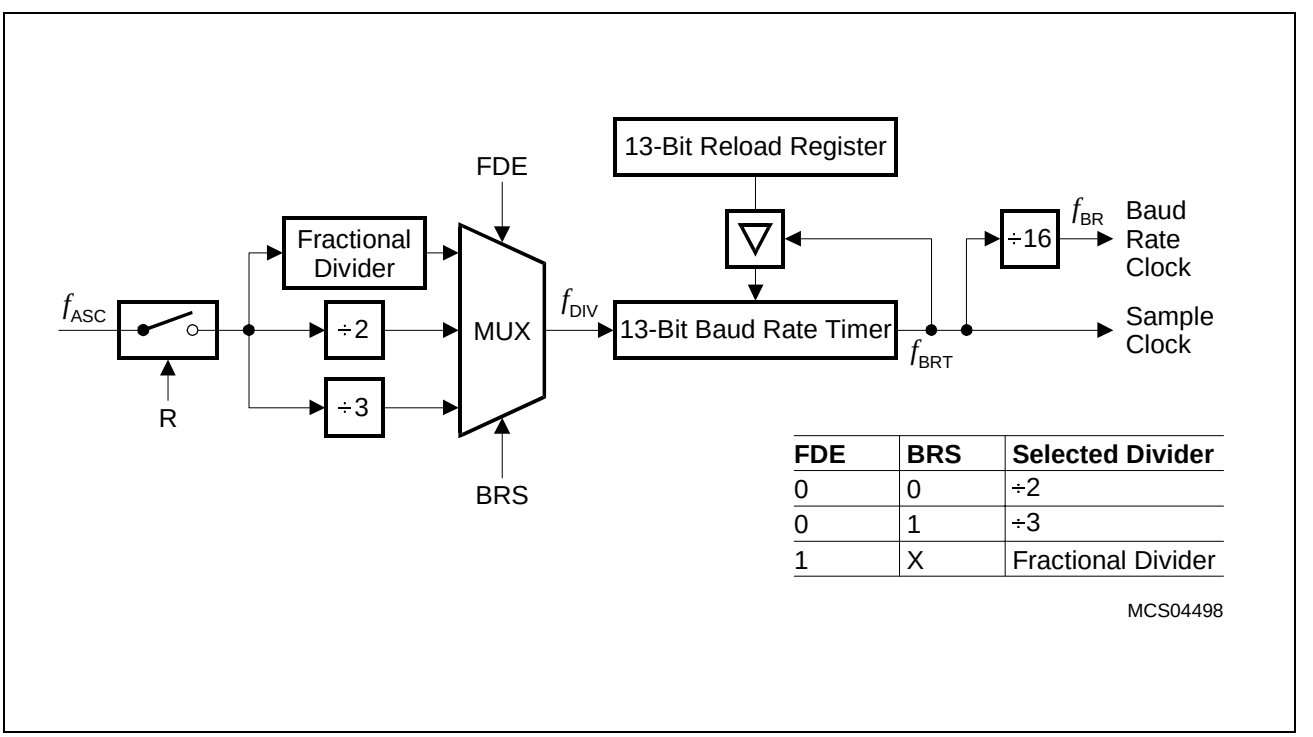
2.1.5.1 Baud Rates in Asynchronous Mode

For asynchronous operation, the baud rate generator provides a clock f_{BRT} with sixteen times the rate of the established baud rate. Every received bit is sampled at the 7th, 8th and 9th cycle of this clock. The clock divider circuitry, which generates the input clock for the 13-bit baud rate timer, is extended by a fractional divider circuitry that allows the adjustment of more accurate baud rates and the extension of the baud rate range.

The baud rate of the baud rate generator depends on the settings of the following bits and register values:

- Input clock f_{ASC}
- Selection of the baud rate timer input clock f_{DIV} by bits CON.FDE and CON.BRS
- If bit CON.FDE = 1 (fractional divider): value of register FDV
- Value of the 13-bit reload register BG

The output clock of the baud rate timer with the reload register is the sample clock in the asynchronous modes of the ASC. For baud rate calculations, this baud rate clock f_{BR} is derived from the sample clock f_{BRT} by a division by sixteen.



FDE	BRS	Selected Divider
0	0	÷2
0	1	÷3
1	X	Fractional Divider

Figure 2-7 ASC Baud Rate Generator Circuitry in Asynchronous Modes

Using the fixed Input Clock Divider

The baud rate for asynchronous operation of the serial channel ASC, when using the fixed input clock divider ratios (CON.FDE = 0) and the required reload value for a given baud rate can be determined by the following formulas:

Table 2-1 Asynchronous Baud Rate Formulas using the Fixed Input Clock Dividers

FDE	BRS	BG	Formula
0	0	0 … 8191	$\text{Baud rate} = \frac{f_{ASC}}{32 \times (BG + 1)}$ $BG = \frac{f_{ASC}}{32 \times \text{Baud rate}} - 1$
	1		$\text{Baud rate} = \frac{f_{ASC}}{48 \times (BG + 1)}$ $BG = \frac{f_{ASC}}{48 \times \text{Baud rate}} - 1$

BG represents the content of the reload register BG (BR_VALUE), taken as unsigned 13-bit integer.

The maximum baud rate that can be achieved for the asynchronous modes when using the two fixed clock dividers and a module clock of 40 MHz is 1.25 MBaud. The table below lists various commonly used baud rates together with the required reload values and the deviation errors compared to the intended baud rate.

Table 2-2 Typical Asynchronous Baud Rates using Fixed Input Clock Dividers

Baud Rate	BRS = 0, f_{ASC} = 40 MHz		BRS = 1, f_{ASC} = 40 MHz	
	Deviation Error	Reload Value	Deviation Error	Reload Value
781.25 kBaud	–	0000_H	–	–
520.8 kBaud	–	–	–	0000_H
19.2 kBaud	+0.2% / -1.4%	0040_H / 0041_H	+0.9% / -1.4%	$002A_H$ / $002B_H$
9600 kBaud	+0.2% / -0.6%	0081_H / 0082_H	+0.9% / -0.2%	0055_H / 0056_H
4800 kBaud	+0.2% / -0.2%	0103_H / 0104_H	+0.4% / -0.2%	$00AC_H$ / $00AD_H$
2400 kBaud	+0.2% / -0.0%	0207_H / 0209_H	+0.1% / -0.1%	$015A_H$ / $015B_H$
1200 Baud	+0.1% / -0.0%	0410_H / 0412_H	+0.1% / -0.1%	$02B5_H$ / $02B6_H$
110 Baud	not possible		+0.0% / -0.0%	$1D96_H$ / $1D97_H$

Note: CON.FDE must be 0 to achieve the baud rates in the table above. The deviation errors given in the table above are rounded. Using a baud rate crystal will provide correct baud rates without deviation errors.

Using the Fractional Divider

When the fractional divider is selected, the input clock f_{DIV} for the baud rate timer is derived from the module clock f_{ASC} by a programmable divider. If CON.FDE = 1, the fractional divider is activated. It divides f_{ASC} by a fraction of n/512 for any value of n from 0 to 511. If n = 0, the divider ratio is 1, which means that $f_{DIV} = f_{ASC}$. In general, the fractional divider allows the baud rate to programmed with a much better accuracy than with the two fixed prescaler divider stages.

Table 2-3 Asynchronous Baud Rate Formulas using the Fractional Input Clock Divider

FDE	BRS	BG	FDV	Formula
1	–	0 … 8191	1 … 511	$\text{Baud rate} = \frac{\text{FDV}}{512} \times \frac{f_{ASC}}{16 \times (\text{BG} + 1)}$
			0	$\text{Baud rate} = \frac{f_{ASC}}{16 \times (\text{BG} + 1)}$

BG represents the contents of the reload register BG (BR_VALUE), taken as an unsigned 13-bit integer. FDV represents the contents of the fractional divider register (FD_VALUE) taken as an unsigned 9-bit integer.

Table 2-4 Typical Asynchronous Baud Rates using the Fractional Input Clock Divider

f_{ASC}	Desired Baud Rate	BG	FDV	Resulting Baud Rate	Deviation
25 MHz	115.2 kBaud	7	302	115.204 kBaud	< 0.01%
	57.6 kBaud	15	302	57.602 kBaud	< 0.01%
	38.4 kBaud	23	302	39.401 kBaud	< 0.01%
	19.2 kBaud	47	302	19.201 kBaud	< 0.01%
40 MHz	115.2 kBaud	16	401	115.117 kBaud	< 0.01%
	57.6 kBaud	38	460	57.592 kBaud	< 0.01%
	38.4 kBaud	36	291	38.403 kBaud	< 0.01%
	19.2 kBaud	58	232	19.200 kBaud	0%

2.1.5.2 Baud Rates in Synchronous Mode

For synchronous operation, the baud rate generator provides a clock with four times the rate of the established baud rate (see **Figure 2-8**).

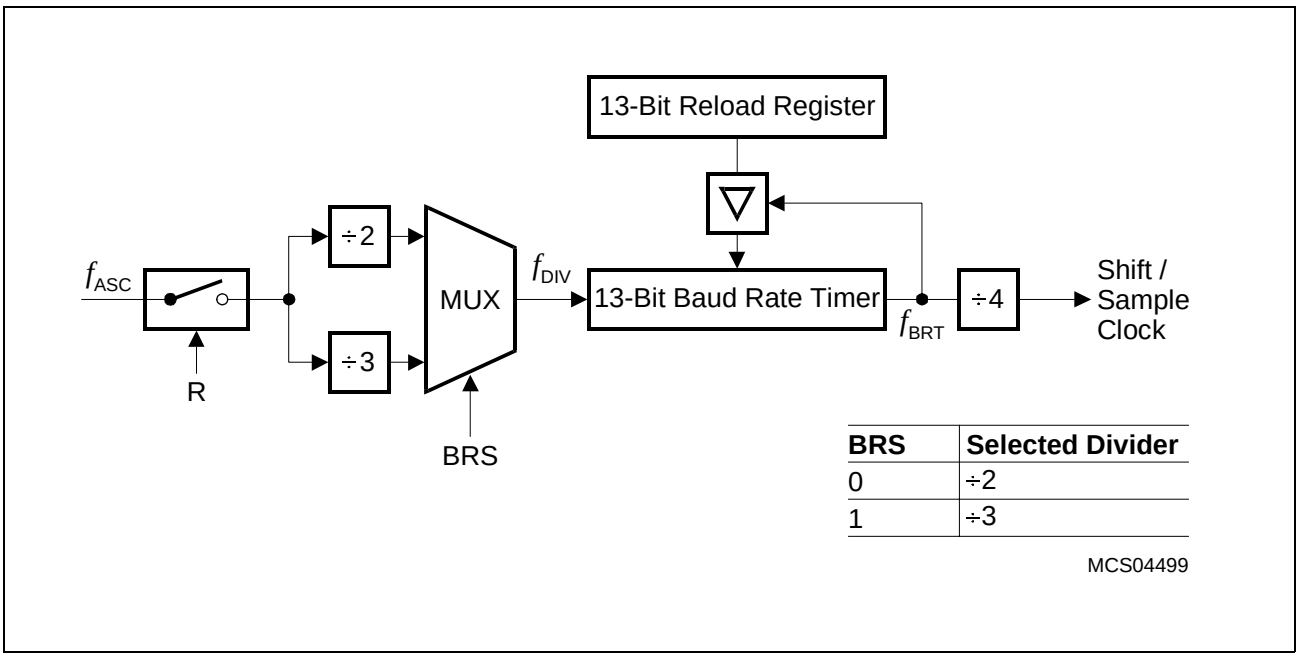


Figure 2-8 ASC Baud Rate Generator Circuitry in Synchronous Mode

The baud rate for synchronous operation of the serial channel ASC can be determined by the formulas as shown in **Table 2-5**.

Table 2-5 Synchronous Baud Rate Formulas

BRS	BG	Formula
0	0 … 8191	$\text{Baud rate} = \frac{f_{ASC}}{8 \times (BG + 1)}$ $\quad BG = \frac{f_{ASC}}{8 \times \text{Baud rate}} - 1$
1		$\text{Baud rate} = \frac{f_{ASC}}{12 \times (BG + 1)}$ $\quad BG = \frac{f_{ASC}}{12 \times \text{Baud rate}} - 1$

BG represents the contents of the reload register (BR_VALUE), taken as unsigned 13-bit integers.

The maximum baud rate that can be achieved in Synchronous Mode when using a module clock of 40 MHz is 5 MBaud.

2.1.6 Hardware Error Detection Capabilities

To improve the reliability of serial data exchange, the serial channel ASC provides an error interrupt request flag that indicates the presence of an error and three (selectable) error status flags in register CON that indicate which error has been detected during reception. Upon completion of a reception, the error interrupt request line EIR will be activated simultaneously with the receive interrupt request line RIR, if one or more of the following conditions are met:

- If the framing error detection enable bit CON.FEN is set and any of the expected stop bits is not high, the framing error flag CON.FE is set, indicating that the error interrupt request is due to a framing error (Asynchronous Mode only).
- If the parity error detection enable bit CON.PEN is set in the modes where a parity bit is received and the parity check on the received data bits proves false, the parity error flag CON.PE is set, indicating that the error interrupt request is due to a parity error (Asynchronous Mode only).
- If the overrun error detection enable bit CON.OEN is set and the last character received was not read out of the receive buffer by software or DMA transfer at the time the reception of a new frame is complete, the overrun error flag CON.OE is set indicating that the error interrupt request is due to an overrun error (asynchronous and synchronous mode).

2.1.7 Interrupts

Four interrupt sources are provided for serial channel ASC. Line TIR indicates a transmit interrupt, TBIR indicates a transmit buffer interrupt, RIR indicates a receive interrupt, and EIR indicates an error interrupt of the serial channel. The interrupt output lines TBIR, TIR, RIR, and EIR are activated (active state) for two periods of the module clock f_{ASC}. The interrupt control unit provides interrupt request flags that are set when these interrupt output lines are activated.

The cause of an error interrupt request EIR (framing, parity, overrun error) can be identified by the error status flags FE, PE, and OE located in control register CON.

Note: In contrary to the error interrupt request line EIR, the error status flags FE/PE/OE are not reset automatically but must be cleared by software.

For normal operation (that is, other than error interrupt) the ASC provides three interrupt requests to control data exchange via this serial channel:

- TBIR is activated when data is moved from TBUF to the transmit shift register.
- TIR is activated before the last bit of an asynchronous frame is transmitted, or after the last bit of a synchronous frame has been transmitted.
- RIR is activated when the received frame is moved to RBUF.

While the task of the receive interrupt handler is quite clear, the transmitter is serviced by two interrupt handlers. This provides advantages for the servicing software.

For single transfers it is sufficient to use the transmitter interrupt (TIR), which indicates that the previously loaded data has been transmitted, except for the last bit of an asynchronous frame.

For multiple back-to-back transfers it is necessary to load the following piece of data at last until the time the last bit of the previous frame has been transmitted. In Asynchronous Mode, this leaves just one bit-time for the handler to respond to the transmitter interrupt request; in Synchronous Mode, it is entirely impossible.

Using the Transmit Buffer Interrupt (TBIR) to reload transmit data provides the time necessary to transmit a complete frame for the service routine, as TBUF may be reloaded while the previous data is still being transmitted.

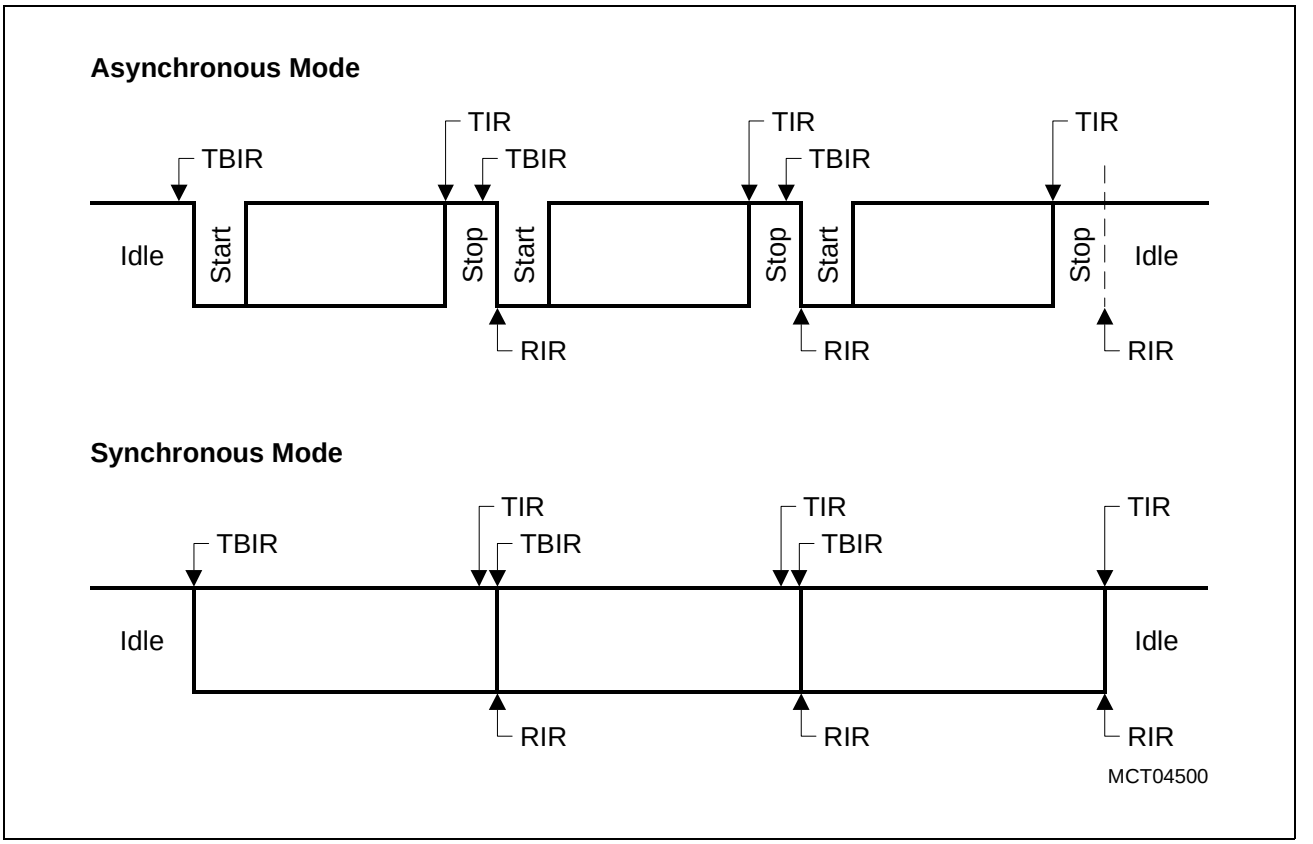


Figure 2-9 ASC Interrupt Generation

As shown in **Figure 2-9** above, TBIR is an early trigger for the reload routine, while TIR indicates the completed transmission. Software using handshake should, therefore, rely on TIR at the end of a data block to ensure that all data has been transmitted.

2.2 ASC Kernel Registers

Figure 2-10 and **Table 2-6** show all registers associated with the ASC Kernel.

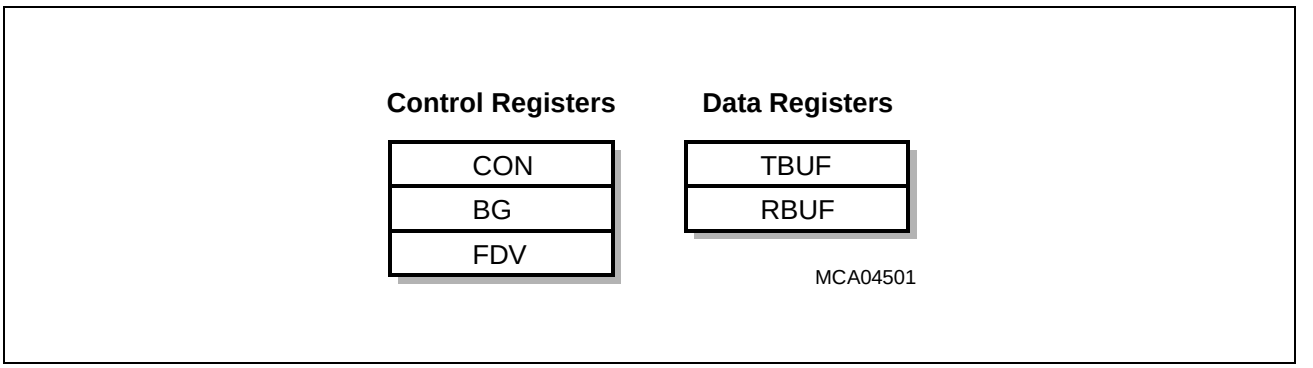


Figure 2-10 ASC Kernel Registers

Table 2-6 ASC Kernel Registers

Register Short Name	Register Long Name	Offset Address	Description see
CON	Control Register	0010_H	**Page 2-22**
BG	Baud Rate Timer Reload Register	0014_H	**Page 2-24**
FDV	Fractional Divider Register	0018_H	**Page 2-24**
TBUF	Transmit Buffer Register	0020_H	**Page 2-25**
RBUF	Receive Buffer Register	0024_H	**Page 2-26**

Note: All ASC kernel register names described in this section will be referenced in other parts of the TC1775 User's Manual with the module name prefix "ASC0_" for the ASC0 interface and "ASC1_" for the ASC1 interface.

The serial operating modes of the ASC module are controlled by its control register CON. This register contains control bits for mode and error check selection, and status flags for error identification.

CON
Control Register **Reset Value: 0000 0000$_H$**

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LB	BRS	ODD	FDE	OE	FE	PE	OEN	FEN	PEN	REN	STP	M		
rw	rw	rw	rw	rw	rwh	rwh	rwh	rw	rw	rw	rwh	rw	rw		

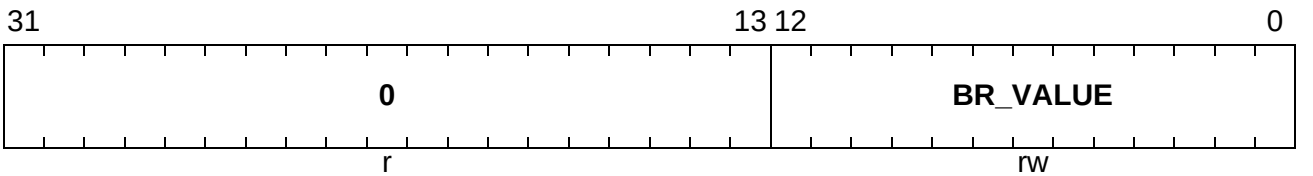
Field	Bits	Type	Description
M	[2:0]	rw	**Mode Selection** 000 8-bit data Synchronous Mode 001 8-bit data Asynchronous Mode 010 Reserved. Do not use this combination! 011 7-bit data + parity Asynchronous Mode 100 9-bit data Asynchronous Mode 101 8-bit data + wake up bit Asynchronous Mode 110 Reserved. Do not use this combination! 111 8-bit data + parity Asynchronous Mode
STP	3	rw	**Number of Stop Bit Selection** 0 One stop bit 1 Two stop bits
REN	4	rwh	**Receiver Enable Control** 0 Receiver disabled 1 Receiver enabled Bit is reset by hardware after reception of byte in Synchronous Mode.
PEN	5	rw	**Parity Check Enable (asynchronous modes only)** 0 Ignore parity 1 Check parity
FEN	6	rw	**Framing Check Enable (asynchronous modes only)** 0 Ignore framing errors 1 Check framing errors

Field	Bits	Type	Description
OEN	7	rw	**Overrun Check Enable** 0 Ignore overrun errors 1 Check overrun errors
PE	8	rwh	**Parity Error Flag** Set by hardware on a parity error (PEN = 1). Must be reset by software.
FE	9	rwh	**Framing Error Flag** Set by hardware on a framing error (FEN = 1). Must be reset by software.
OE	10	rwh	**Overrun Error Flag** Set by hardware on an overrun error (OEN = 1). Must be reset by software.
FDE	11	rw	**Fractional Divider Enable** 0 Fractional divider disabled 1 Fractional divider is enabled and used as prescaler for baud rate timer (bit BRS is don't care)
ODD	12	rw	**Parity Selection** 0 Even parity selected (parity bit set on odd number of 1s in data) 1 Odd parity selected (parity bit set on even number of 1s in data)
BRS	13	rw	**Baud Rate Selection** 0 Baud rate timer prescaler divide-by-2 selected 1 Baud rate timer prescaler divide-by-3 selected BRS is don't care if FDE = 1 (fractional divider enabled)
LB	14	rw	**Loopback Mode Enable** 0 Loop-Back mode disabled 1 Loop-Back mode enabled
R	15	rw	**Baud Rate Generator Run Control** 0 Baud rate generator disabled (ASC inactive) 1 Baud rate generator enabled BG should only be written if R = 0.
0	[31:16]	r	**Reserved**; returns 0 if read; should be written with 0.

Note: Serial data transmission or reception is possible only when the run bit CON.R is set to 1. Otherwise, the serial interface is idle. Do not program the mode control field CON.M to one of the reserved combinations to avoid unpredictable behavior of the serial interface.

The baud rate timer reload register BG of the ASC module contains the 13-bit reload value for the baud rate timer in Asynchronous and Synchronous Mode.

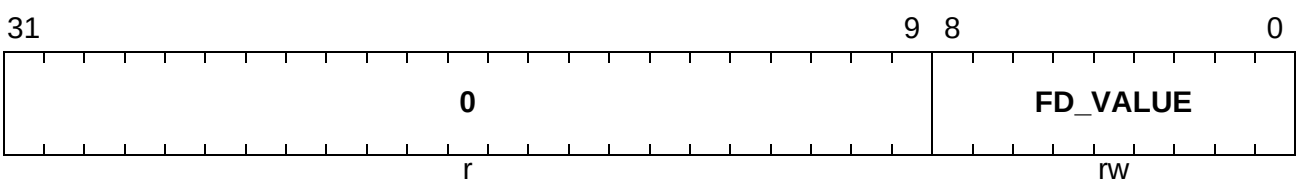
BG
Baud Rate Timer/Reload Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
BR_VALUE	[12:0]	rw	**Baud Rate Timer/Reload Register Value** Reading BG returns the 13-bit content of the baud rate timer. Writing BG loads the baud rate timer reload register. BG should only be written if CON.R = 0.
0	[31:13]	r	**Reserved**; returns 0 if read; should be written with 0.

The fractional divider register FDV of the ASC module contains the 9-bit divider value for the fractional divider (asynchronous mode only).

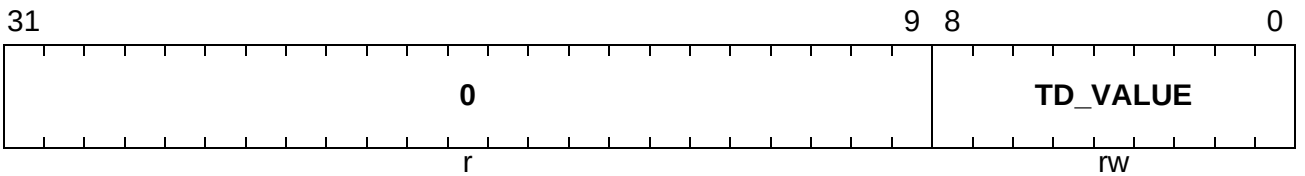
FDV
Fractional Divider Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
FD_VALUE	[8:0]	rw	**Fractional Divider Register Value** FDV contains the 9-bit value n of the fractional divider which defines the fractional divider ratio n/512 (n = 0-511). With n = 0, the fractional divider is switched off (divider ratio = 1).
0	[31:9]	r	**Reserved**; read as 0; should be written with 0.

The transmitter buffer register TBUF of the ASC module contains the transmit data value in Asynchronous and Synchronous Modes.

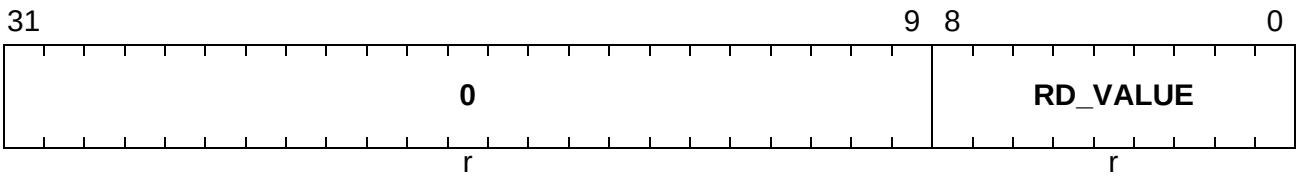
TBUF
Transmitter Buffer Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
TD_VALUE	[8:0]	rw	**Transmit Data Register Value** TBUF contains the data to be transmitted in the asynchronous and synchronous operating modes of the ASC. Data transmission is double buffered; therefore, a new value can be written to TBUF before the transmission of the previous value is complete.
0	[31:9]	r	**Reserved**; read as 0 should be written with 0.

The receiver buffer register RBUF of the ASC module contains the receive data value in Asynchronous and Synchronous Modes.

RBUF
Receiver Buffer Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
RD_VALUE	[8:0]	r	**Receive Data Register Value** RBUF contains the received data bits and, depending on the selected mode, the parity bit in the asynchronous and synchronous operating modes of the ASC. In Asynchronous Mode, with CON.M = 011_B (7-bit data + parity), the received parity bit is written into RBUF.7. In Asynchronous Mode with CON.M = 111_B (8-bit data + parity), the received parity bit is written into RBUF.8.
0	[31:9]	r	**Reserved**; read as 0.

2.3 ASC0/ASC1 Module Implementation

This section describes ASC0/ASC1 module interfaces with the clock control, port connections, interrupt control, and address decoding.

2.3.1 Interfaces of the ASC Modules

Figure 2-11 shows the TC1775 specific implementation details and interconnections of the ASC0/ASC1 modules. Each of the ASC modules has two pairs of RXD/TXD I/O lines located at Port 12 or Port 13. Each of the ASC modules is further supplied by a separate clock control, interrupt control, address decoding, and port control logic.

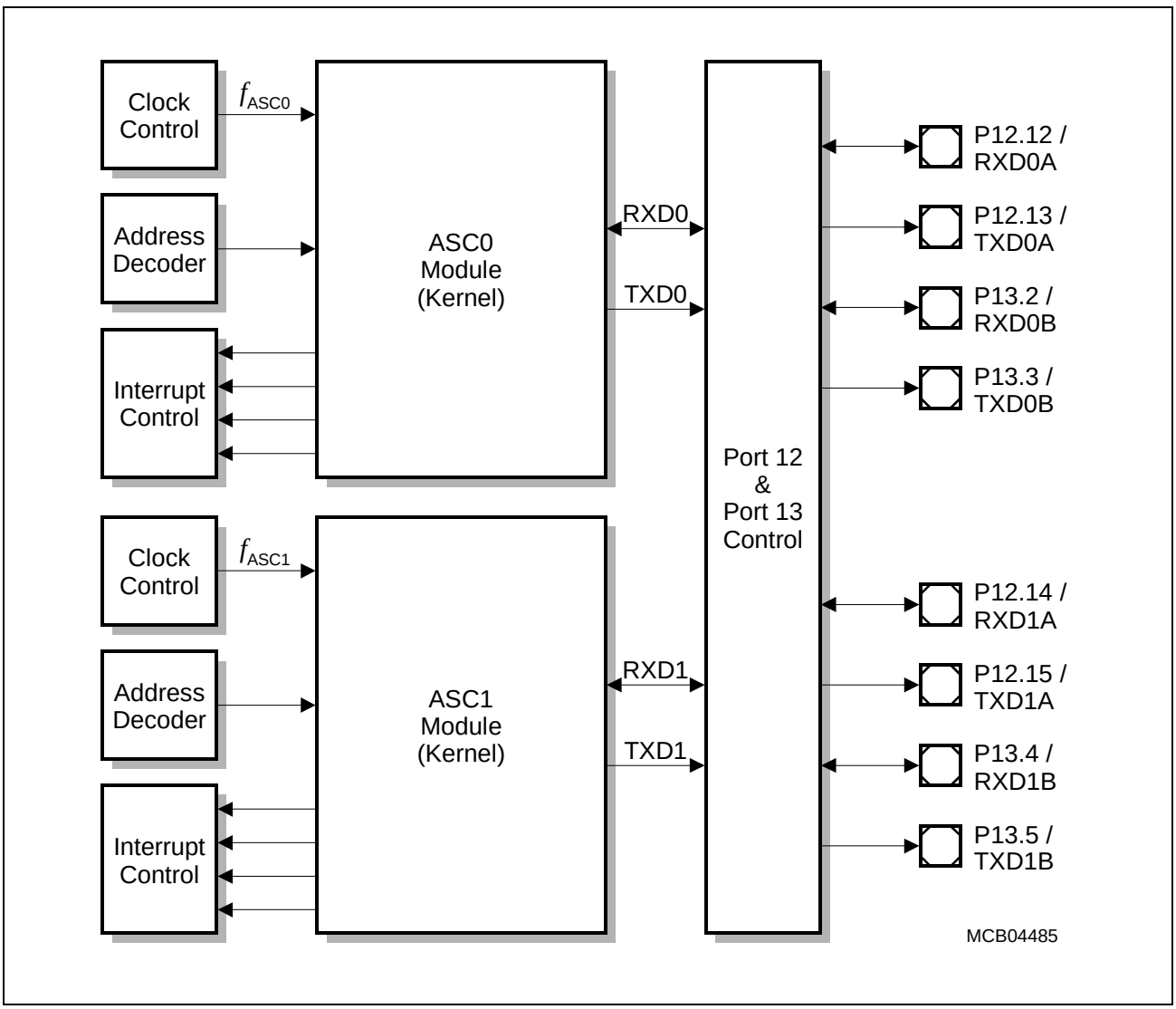


Figure 2-11 ASC0/ASC1 Module Implementation and Interconnections

2.3.2 ASC0/ASC1 Module Related External Registers

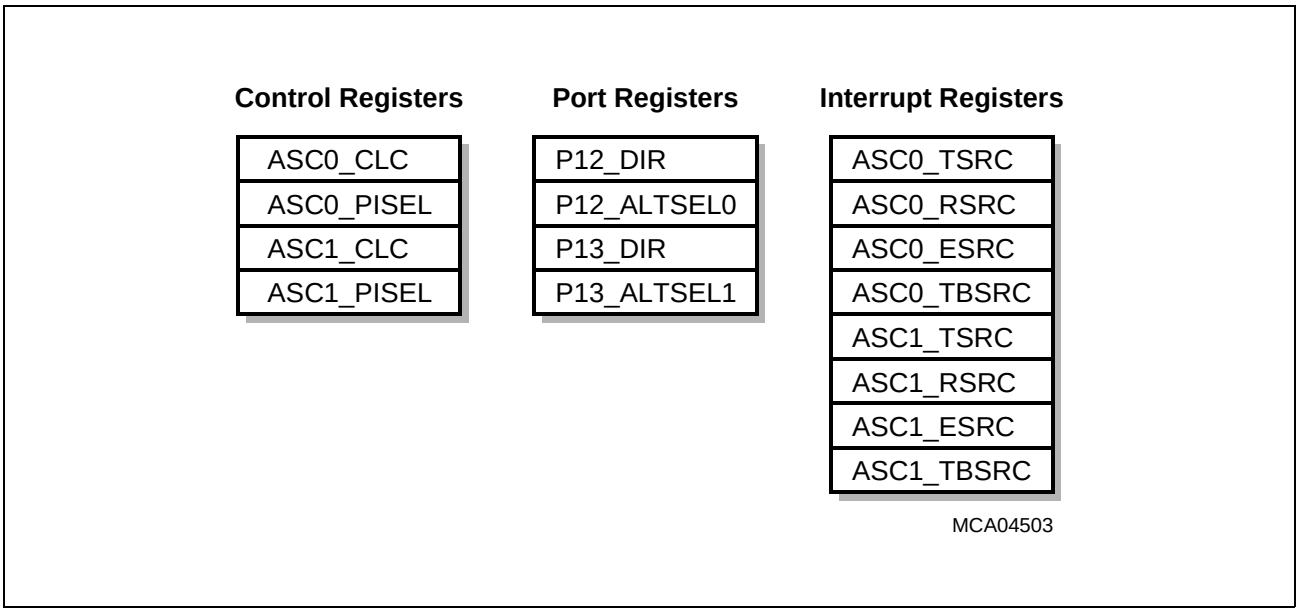
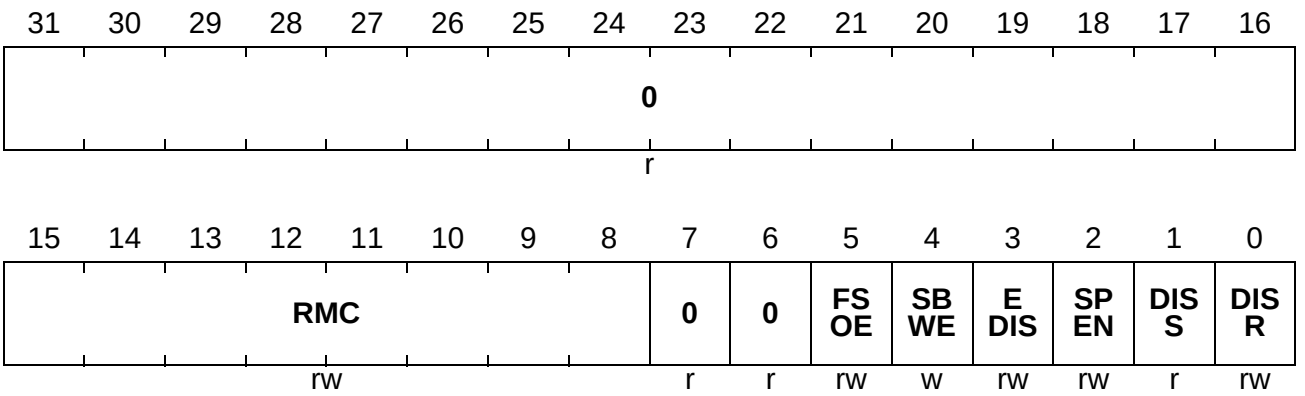


Figure 2-12 ASC0/ASC1 Implementation Specific Special Function Registers

2.3.2.1 Clock Control Registers

The clock control register allows the programmer to adapt the functionality and power consumption of an ASC module to the requirements of the application. The table below shows the clock control register functionality which is implemented for the ASC modules. ASC0_CLC is controlling the f_{ASC0} clock signal and ASC1_CLC is controlling the f_{ASC1} clock signal.

ASC0_CLC
ASC0 Clock Control Register **Reset Value: 0000 0002$_H$**
ASC1_CLC
ASC1 Clock Control Register **Reset Value: 0000 0002$_H$**



Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control of the module.
DISS	1	r	**Module Disable Status Bit** Bit indicates the current status of the module.
SPEN	2	rw	**Module Suspend Enable for OCDS** Used for enabling the suspend mode.
EDIS	3	rw	**External Request Disable** Used for controlling the external clock disable request.
SBWE	4	w	**Module Suspend Bit Write Enable for OCDS** Defines whether SPEN and FSOE are write protected.
FSOE	5	rw	**Fast Switch Off Enable** Used for fast clock switch off in OCDS suspend mode.
RMC	[15:8]	rw	**8-Bit Clock Divider Value in RUN Mode**
0	7, 6, [31:16]	r	**Reserved**; returns 0 if read; should be written with 0.

Note: After a hardware reset operation, the ASC modules are disabled.

2.3.2.2 Peripheral Input Select Registers

The ASC0/ASC1 module kernels provide a Peripheral Input Select Register that can be used to switch the RXD input lines of the ASC0/ASC1 module kernels to either Port 12 or Port 13, as shown in **Figure 2-12**. This feature can be useful if, for example, Port 13 is required for the eight GPTU I/O lines. In this case the lines of Port 12 can be assigned as ASC I/O port (default after reset).

*Note: As shown in **Figure 2-12**, the RXD line of the ASC module can also be an output line in Synchronous Mode. Port line input/output switching is controlled with the port direction registers.*

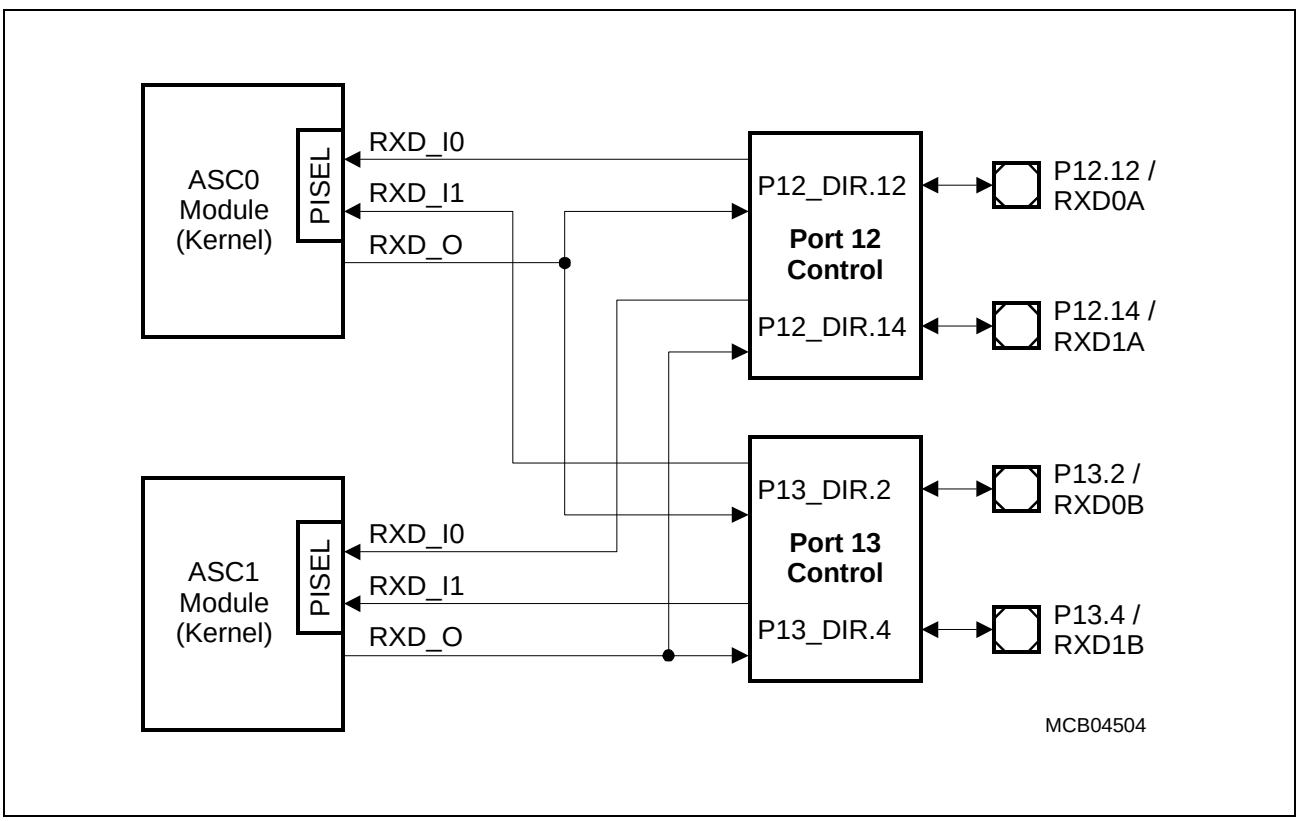
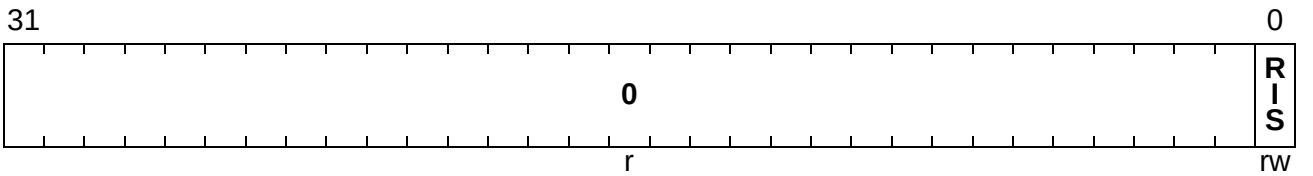


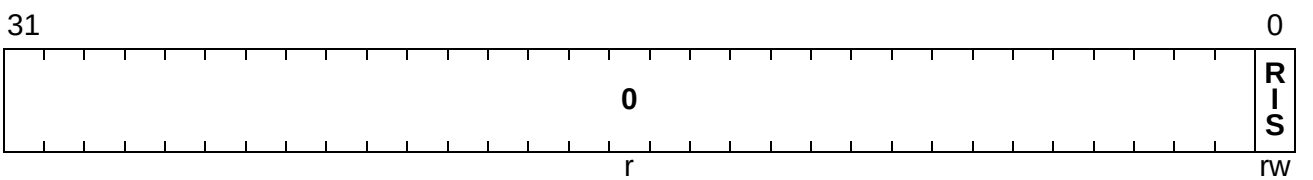
Figure 2-13 RXD Input Line Selection of the ASC Modules

ASC0_PISEL
ASC0 Peripheral Input Select Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
RIS	0	rw	**Receive Input Select** 0 ASC0 receiver input RXD0A (P12.12) selected 1 ASC0 receiver input RXD0B (P13.2) selected
0	[31:1]	0	**Reserved**; returns 0 if read; should be written with 0.

ASC1_PISEL
ASC1 Peripheral Input Select Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
RIS	0	rw	**Receive Input Select** 0 ASC1 receiver input RXD1A (P12.14) selected 1 ASC1 receiver input RXD1B (P13.4) selected
0	[31:1]	0	**Reserved**; returns 0 if read; should be written with 0.

2.3.2.3 Port Registers

The interconnections between the ASC modules and the port I/O lines are controlled in the port logic of Port 12 and Port 13. Two basic selections must be executed:

– Alternate function select by the port alternate select (ALTSEL) registers
– Direction control by the port direction (DIR) registers

The port registers which are related to the ASC I/O lines are the alternate select registers P12_ALTSEL0 and P13_ALTSEL1 and the direction registers P12_DIR and P13_DIR.

P12_ALTSEL0
Port 12 Alternate Select Register 0 **Reset Value: 0000 0000$_H$**

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	X											
rw	rw	rw	rw	rw											

P13_ALTSEL1
Port 13 Alternate Select Register 1 **Reset Value: 0000 0000$_H$**

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X										P5	P4	P3	P2	X	
rw										rw	rw	rw	rw	rw	

The direction registers configure the direction of a port pin and must be set according to the selected ASC operation mode (if direction bit = 0, the pin is set to input; direction bit = 1, the pin is set to output). The ASC I/O lines are connected with Port 12 and Port 13. Therefore, the Port 12 and Port 13 Direction Register P12_DIR/P13_DIR must be set accordingly.

Note: Bits marked with “X” are not relevant for ASC operation.

P12_DIR
Port 12 Direction Register **Reset Value: 0000 0000$_H$**

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	X											
rw	rw	rw	rw	rw											

P13_DIR
Port 13 Direction Register **Reset Value: 0000 0000$_H$**

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X										P5	P4	P3	P2	X	
rw										rw	rw	rw	rw	rw	

Table 2-7 shows which bits must be set/reset depending on the required I/O functionality of the ASC I/O lines. This table also shows the values of the peripheral input select registers.

Table 2-7 ASC0/ASC1 I/O Line Selection and Setup

Module	Port Lines	PISEL Register	Alternate Select Register Bits		Direction Register Bits		I/O
			P12_ ALTSEL0	P13_ ALTSEL1	P12_DIR	P13_DIR	
ASC0	P12.12 / RXD0A	ASC0_ PISEL.RIS = ‘0’	P12 = 1	–	P12 = 0	–	Input
					P12 = 1		Out-put
	P13.2 / RXD0B	ASC0_ PISEL.RIS = 1	–	P2 = 1	–	P2 = 0	Input
						P2 = 1	Out-put
	P12.13 / TXD0A	–	P13 = 1	–	–	–	Out-put
	P13.3 / TXD0B	–	–	P3 = 1	–	–	Out-put
ASC1	P12.14 / RXD1A	ASC1_ PISEL.RIS = 0	P14 = 1	–	P14 = 0	–	Input
					P14 = 1		Out-put
	P13.4 / RXD1B	ASC1_ PISEL.RIS = 1	–	P4 = 1	–	P4 = 0	Input
						P4 = 1	Out-put
	P12.15 / TXD1A	–	P15 = 1	–	–	–	Out-put
	P13.5 / TXD1B	–	–	P5 = 1	–	–	Out-put

Note: In Synchronous Mode the direction of the selected RXD port pin (input or output) is not automatically set by the ASC but must be switched by the user program, depending on the selected mode (receive or transmit data).

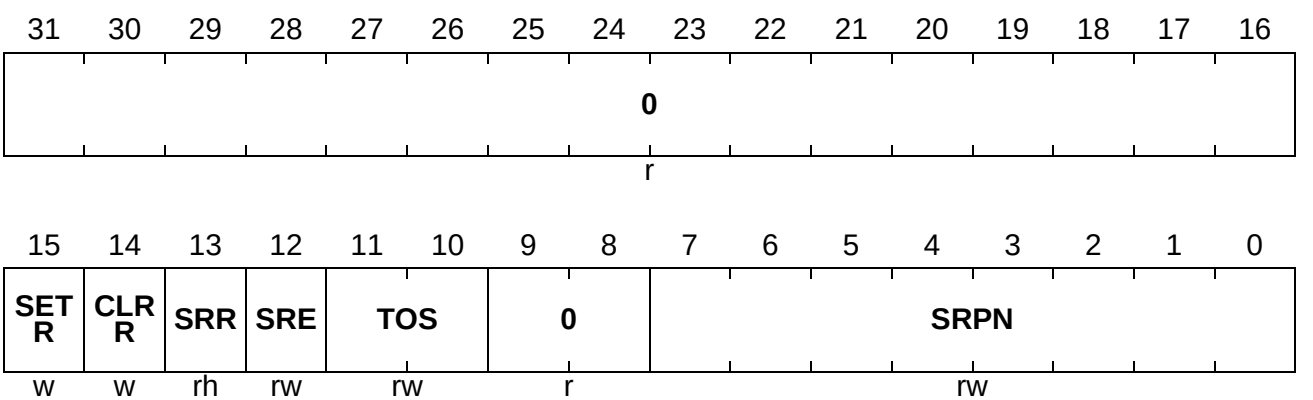
2.3.2.4 Interrupt Registers

The eight interrupts of the ASC0 and ASC1 modules are controlled by the following service request control registers:

- ASC0_TSRC, ASC1_TSRC — controls the transmit interrupts
- ASC0_RSRC, ASC1_RSRC — controls the receive interrupts
- ASC0_ESRC, ASC1_ESRC — controls the error interrupts
- ASC0_TBSRC, ASC1_TBSRC — controls the transmit buffer empty interrupts

ASC0_TSRC
ASC0 Transmit Interrupt Service Request Control Register
ASC0_RSRC
ASC0 Receive Interrupt Service Request Control Register
ASC0_ESRC
ASC0 Error Interrupt Service Request Control Register
ASC0_TBSRC
ASC0 Transmit Buffer Interrupt Service Request Control Register
ASC1_TSRC
ASC1 Transmit Interrupt Service Request Control Register
ASC1_RSRC
ASC1 Receive Interrupt Service Request Control Register
ASC1_ESRC
ASC1 Error Interrupt Service Request Control Register
ASC1_TBSRC
ASC1 Transmit Buffer Interrupt Service Request Control Register

Reset Values: 0000 0000$_H$



Field	Bits	Type	Description
SRPN	[7:0]	rw	**Service Request Priority Number**
TOS	[11:10]	rw	**Type of Service Control**
SRE	12	rw	**Service Request Enable**

Field	Bits	Type	Description
SRR	13	rh	**Service Request Flag**
CLRR	14	w	**Request Clear Bit**
SETR	15	w	**Request Set Bit**
0	[9:8], [31:16]	r	**Reserved**; returns 0 if read; should be written with 0.

Note: Further details on interrupt handling and processing are described in the "Interrupt System" chapter of the TC1775 System Units User's Manual.

2.3.3 ASC0/ASC1 Register Address Ranges

In the TC1775, the registers of the two ASC modules are located in the following address ranges:

- ASC0 module: Module Base Address = F000 0800_H
  Module End Address = F000 $08FF_H$
- ASC1 module: Module Base Address = F000 0900_H
  Module End Address = F000 $09FF_H$
- Absolute Register Address = Module Base Address + Offset Address
  (offset addresses see **Table 2-6**)

3 High-Speed Synchronous Serial Interface (SSC)

This chapter describes the two high-speed synchronous serial interfaces of the TC1775, SSC0 and SSC1. It contains the following sections:

- Functional description of the SSC Kernel, valid for SSC0 and SSC1 (see **Section 3.1**)
- SSC kernel register descriptions of all SSC Kernel specific registers (see **Section 3.2**)
- TC1775 implementation specific details and registers of the SSC0/SSC1 Modules (port connections and control, interrupt control, address decoding, and clock control, see **Section 3.3**) with register address ranges (see **Chapter 3.3.3**)

*Note: The SSC kernel register names described in **Section 3.2** will be referenced in the TC1775 User's Manual by the module name prefix "SSC0_" for the SSC0 interface and by "SSC1_" for the SSC1 interface.*

3.1 SSC Kernel Description

Figure 3-1 shows a global view of all functional blocks of the SSC interface.

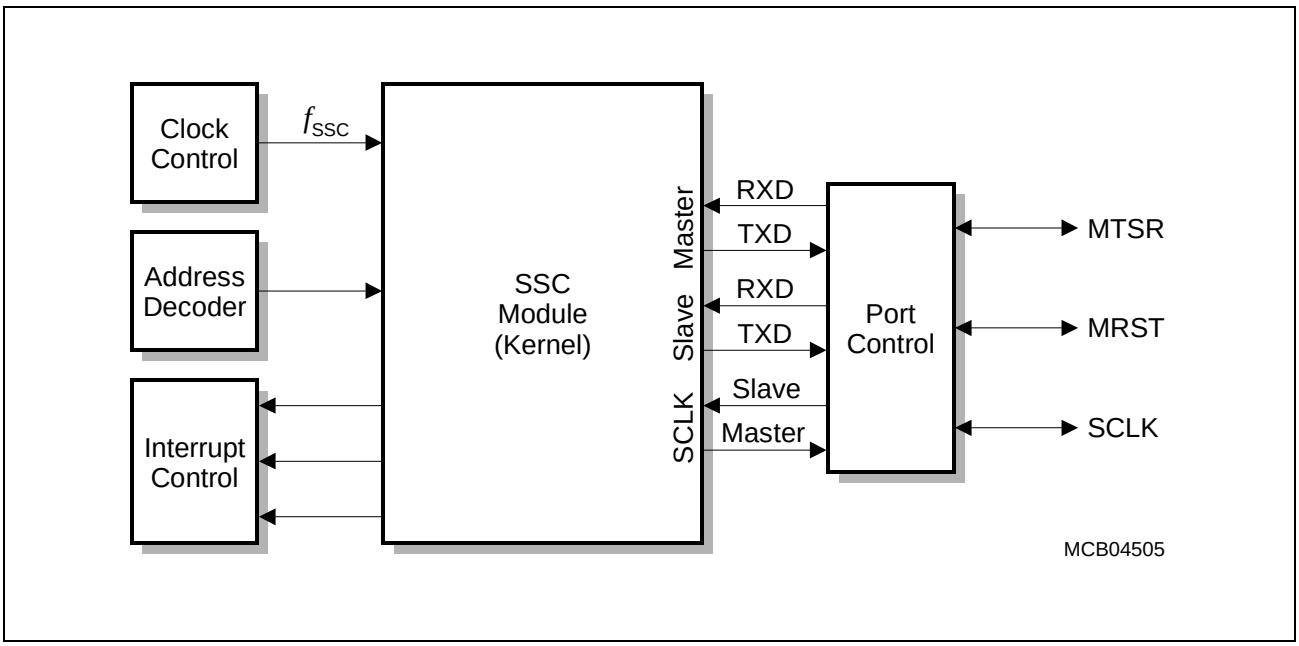


Figure 3-1 General Block Diagram of the SSC Interface

3.1.1 Overview

The SSC supports full-duplex and half-duplex serial synchronous communication up to 20 MBaud (@ 40 MHz module clock). The serial clock signal can be generated by the SSC itself (master mode) or can be received from an external master (slave mode). Data width, shift direction, clock polarity and phase are programmable. This allows communication with SPI-compatible devices. Transmission and reception of data is double-buffered. A 16-bit baud rate generator provides the SSC with a separate serial clock signal.

Features:

- Master and slave mode operation
  - Full-duplex or half-duplex operation
- Flexible data format
  - Programmable number of data bits: 2 to 16 bit
  - Programmable shift direction: LSB or MSB shift first
  - Programmable clock polarity: idle low or high state for the shift clock
  - Programmable clock/data phase: data shift with leading or trailing edge of the shift clock
- Baud rate generation from 20 MBaud to 305.18 Baud (@ 40 MHz module clock)
- Interrupt generation
  - On a transmitter empty condition
  - On a receiver full condition
  - On an error condition (receive, phase, baud rate, transmit error)
- Three-pin interface
  - Flexible SSC pin configuration

3.1.2 General Operation

The SSC supports full-duplex and half-duplex synchronous communication up to 20 MBaud (@ 40 MHz module clock). The serial clock signal can be generated by the SSC itself (master mode) or be received from an external master (slave mode). Data width, shift direction, clock polarity and phase are programmable. This allows communication with SPI-compatible devices. Transmission and reception of data are double-buffered. A 16-bit baud rate generator provides the SSC with a separate serial clock signal.

Configuration of the high-speed synchronous serial interface is very flexible, so it can work with other synchronous serial interfaces, can serve for master/slave or multimaster interconnections, or can operate compatibly with the popular SPI interface. It can be used to communicate with shift registers (I/O expansion), peripherals (e.g. EEPROMs etc.), or other controllers (networking). The SSC supports half-duplex and full-duplex communication. Data is transmitted or received on pins MTSR (Master Transmit / Slave Receive) and MRST (Master Receive/Slave Transmit). The clock signal is output or input via pin SCLK. These pins are alternate functions of port pins.

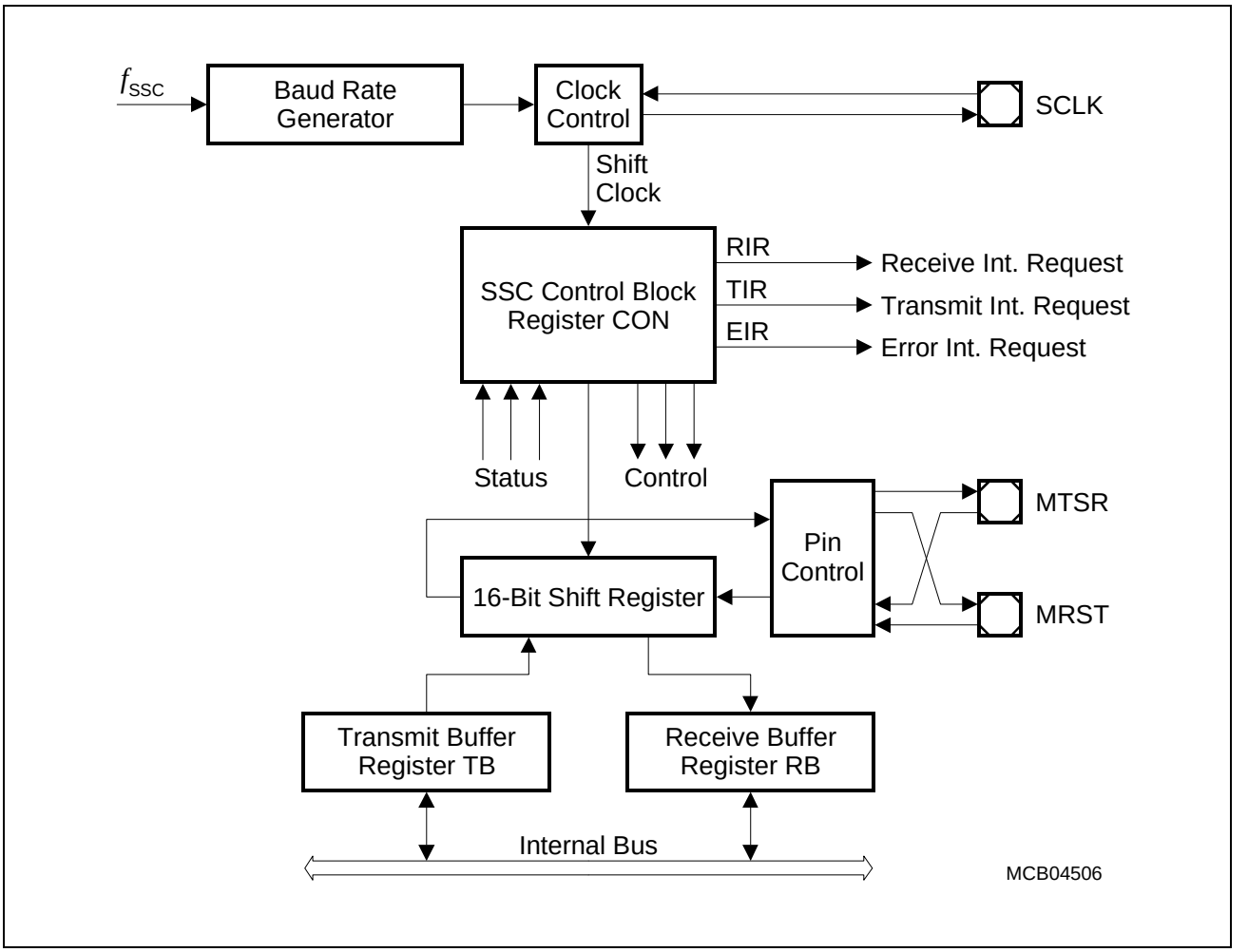


Figure 3-2 Synchronous Serial Channel SSC Block Diagram

3.1.2.1 Operating Mode Selection

The operating mode of the serial channel SSC is controlled by its control register, CON. This register serves two purposes:

- During programming (SSC disabled by CON.EN = 0), it provides access to a set of control bits
- During operation (SSC enabled by CON.EN = 1), it provides access to a set of status flags.

The shift register of the SSC is connected to both the transmit pin and the receive pin via the pin control logic (see block diagram in **Figure 3-2**). Transmission and reception of serial data are synchronized and take place at the same time, that is, the same number of transmitted bits is also received. Transmit data is written into the Transmit Buffer TB. It is moved to the shift register as soon as this is empty. An SSC master (CON.MS = 1) immediately begins transmitting, while an SSC slave (CON.MS = 0) will wait for an active shift clock. When the transfer starts, the busy flag CON.BSY is set and the Transmit Interrupt Request line (TIR) will be activated to indicate that register Transmit Buffer (TB) may be reloaded. When the number of bits (2 to 16, as programmed) have been transferred, the contents of the shift register are moved to the Receive Buffer (RB) and the Receive Interrupt Request line (RIR) will be activated. If no further transfer is to take place (TB is empty), CON.BSY will be cleared at the same time. Software should not modify CON.BSY, as this flag is hardware controlled.

Note: Only one SSC (etc.) can be master at a given time.

The transfer of serial data bits can be programmed in many respects:

- The data width can be selected from 2 bits to 16 bits
- A transfer may start with the LSB or the MSB
- The shift clock may be idle low or idle high
- The data bits may be shifted with the leading or trailing edge of the clock signal
- The baud rate may be set from 305.2 Baud up to 20 MBaud (@ 40 MHz module clock)
- The shift clock can be generated (master) or received (slave)

These features allow the SSC to be adapted to a wide range of applications that require serial data transfer.

The Data Width Selection supports the transfer of frames of any data length from 2-bit “characters” up to 16-bit “characters”. Starting with the LSB (CON.HB = 0) allows communication with such devices as an SSC device in synchronous mode or 8051-like serial interfaces. Starting with the MSB (CON.HB = 1) allows operation compatible with the SPI interface.
Regardless of the data width selected and whether the MSB or the LSB is transmitted first, the transfer data is always right-aligned in registers TB and RB, with the LSB of the transfer data in bit 0 of these registers. The data bits are rearranged for transfer by the

internal shift register logic. The unselected bits of TB are ignored, the unselected bits of RB will not be valid and should be ignored by the receiver service routine.

The Clock Control allows the adaptation of transmit and receive behavior of the SSC to a variety of serial interfaces. A specific clock edge (rising or falling) is used to shift out transmit data, while the other clock edge is used to latch in receive data. Bit CON.PH selects the leading edge or the trailing edge for each function. Bit CON.PO selects the level of the clock line in the idle state. So for an idle-high clock, the leading edge is a falling one, a 1-to-0 transition (see **Figure 3-3**).

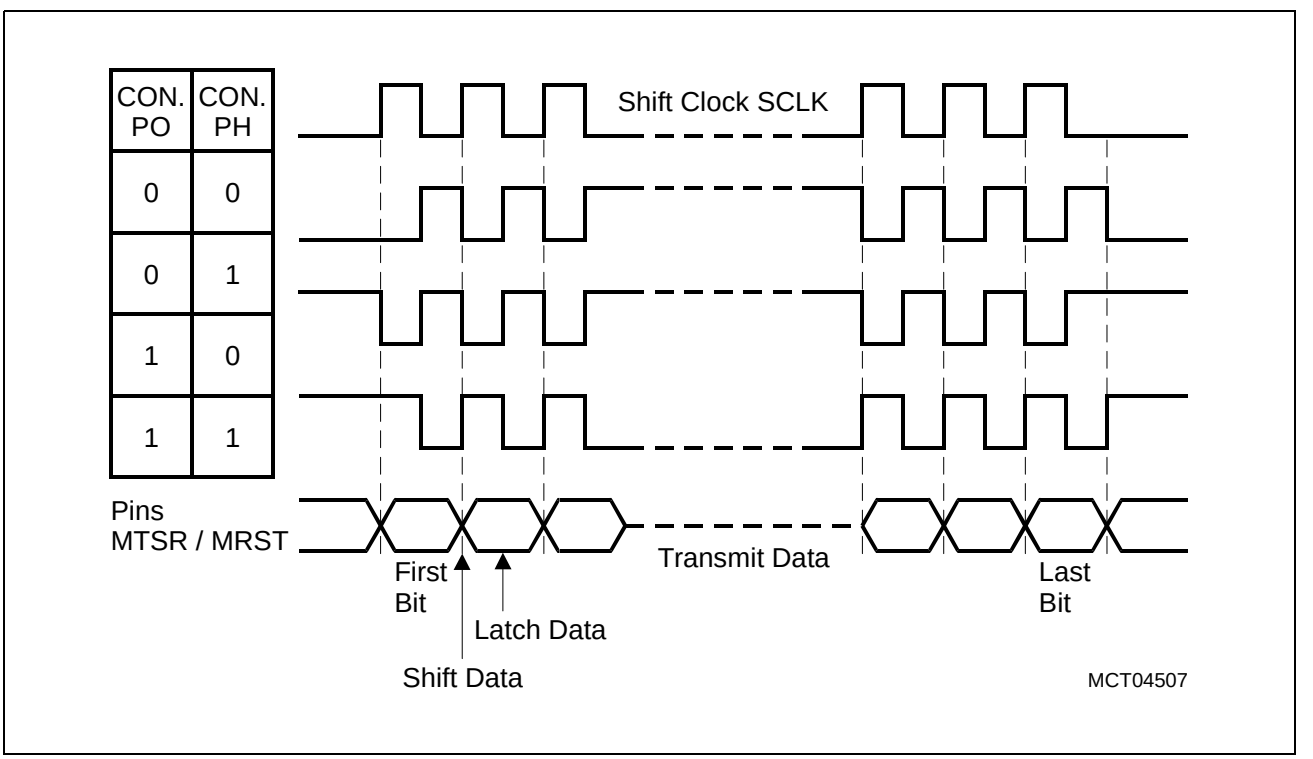


Figure 3-3 Serial Clock Phase and Polarity Options

3.1.2.2 Full-Duplex Operation

The various devices are connected through three lines. The definition of these lines is always determined by the master. The line connected to the master's data output pin MTSR is the transmit line, the receive line is connected to its data input line MRST, and the clock line is connected to pin SCLK. Only the device selected for master operation generates and outputs the serial clock on pin SCLK. All slaves receive this clock, so their pin SCLK must be switched to input mode. The output of the master's shift register is connected to the external transmit line, which in turn is connected to the slaves' shift register input. The output of the slaves' shift register is connected to the external receive line in order to enable the master to receive the data shifted out of the slave. The external connections are hard-wired, with the function and direction of these pins determined by the master or slave operation of the individual device.

*Note: The shift direction shown in **Figure 3-4** applies to both MSB-first and LSB-first operation.*

When initializing the devices in this configuration, one device must be selected for master operation while all other devices must be programmed for slave operation. Initialization includes the operating mode of the device's SSC and also the function of the respective port lines.

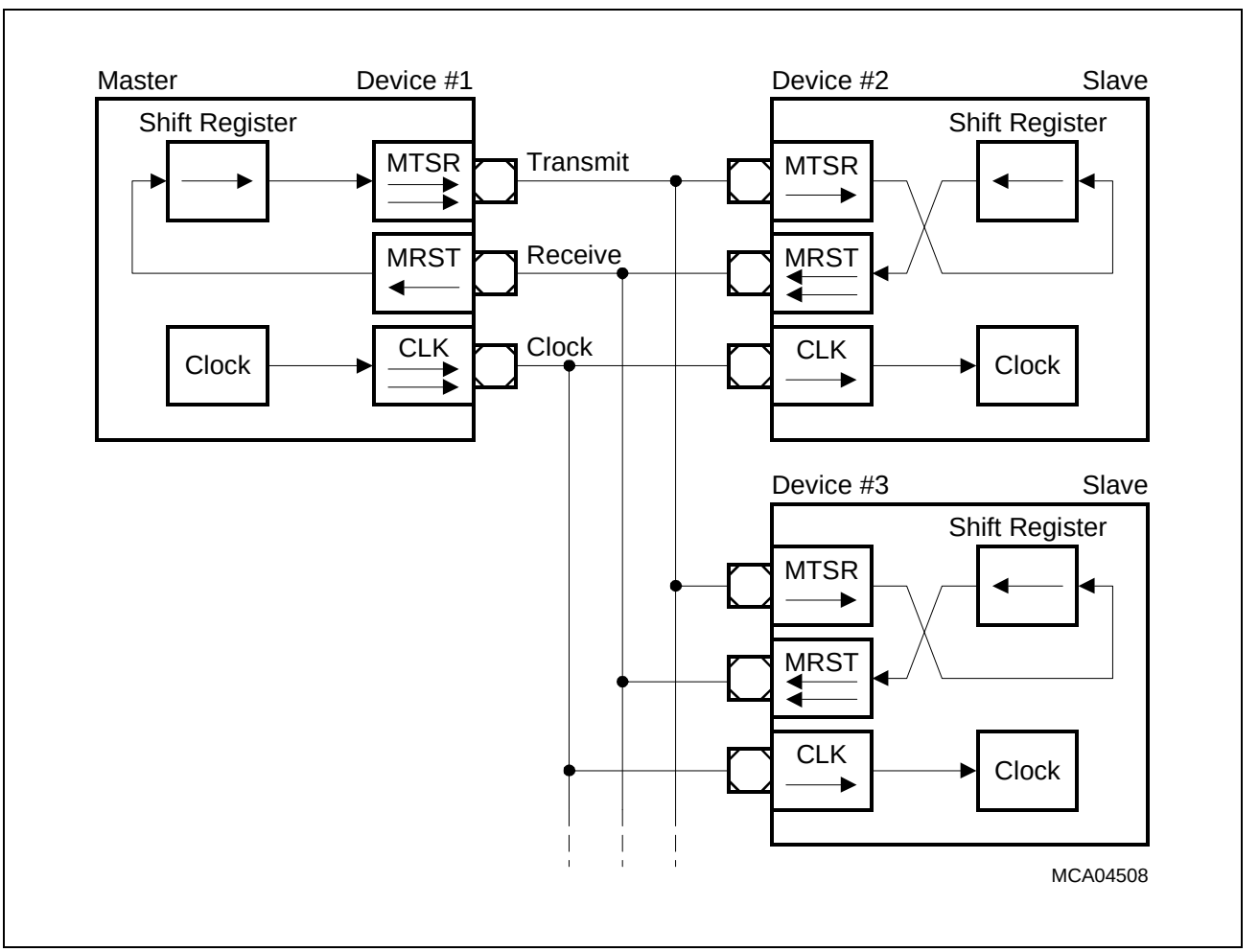


Figure 3-4 SSC Full-Duplex Configuration

The data output pins MRST of all slave devices are connected onto one receive line in this configuration. During a transfer each slave shifts out data from its shift register. There are two ways to avoid collisions on the receive line due to different slave data:

– **Only one slave drives the line** and enables the driver of its MRST pin. All the other slaves must program their MRST pins to input. So, only one slave can put its data onto the master's receive line. Only reception of data from the master is possible. The master selects the slave device from which it expects data either by separate select lines, or by sending a special command to this slave. The selected slave then switches its MRST line to output until it gets a de-selection signal or command.

– **The slaves use open drain output on MRST**. This forms a wired-AND connection. The receive line needs an external pull-up in this case. Corruption of the data on the receive line sent by the selected slave is avoided when all slaves not selected for transmission to the master send only 1s. Since this high level is not actively driven onto the line, but is only held through the pull-up device, the selected slave can pull this line actively to a low level when transmitting a zero bit. The master selects the slave device from which it expects data either by separate select lines, or by sending a special command to this slave.

After performing all necessary initializations of the SSC, the serial interfaces can be enabled. For a master device, the alternate clock line will now go to its programmed polarity. The alternate data line will go to either 0 or 1, until the first transfer will start. After a transfer, the alternate data line will always remain at the logic level of the last transmitted data bit.

When the serial interfaces are enabled, the master device can initiate the first data transfer by writing the transmit data into register TB. This value is copied into the shift register (assumed to be empty at this time), and the selected first bit of the transmit data will be placed onto the MTSR line on the next clock from the baud rate generator (transmission only starts, if CON.EN = 1). Depending on the selected clock phase, also a clock pulse will be generated on the SCLK line. With the opposite clock edge, the master simultaneously latches and shifts in the data detected at its input line MRST. This “exchanges” the transmit data with the receive data. Because the clock line is connected to all slaves, their shift registers will be shifted synchronously with the master’s shift register, shifting out the data contained in the registers, and shifting in the data detected at the input line. After the preprogrammed number of clock pulses (via the data width selection), the data transmitted by the master is contained in all slaves’ shift registers, while the master’s shift register holds the data of the selected slave. In the master and all slaves, the content of the shift register is copied into the Receive Buffer (RB) and the Receive Interrupt Line (RIR) is activated.

A slave device will immediately output the selected first bit (MSB or LSB of the transfer data) at pin MRST when the contents of the transmit buffer are copied into the slave’s shift register. Bit CON.BSY is not set until the first clock edge at SCLK appears. The slave device will not wait for the next clock from the baud rate generator — as the master does — because the first clock edge generated by the master may be already used to clock in the first data bit, depending on the selected clock phase. So the slave’s first data bit must already be valid at this time.

Note: On the SSC a transmission and a reception always takes place at the same time, regardless whether valid data has been transmitted or received.

3.1.2.3 Half-Duplex Operation

In a half-duplex configuration, only one data line is necessary for both receiving and transmitting data. The data exchange line is connected to both pins MTSR and MRST of each device, the clock line is connected to the SCLK pin.

The master device controls the data transfer by generating the shift clock, while the slave devices receive it. Due to the fact that all transmit and receive pins are connected to the one data exchange line, serial data may be moved between arbitrary stations.

Similar to full-duplex mode there are two ways to avoid collisions on the data exchange line:

- Only the transmitting device may enable its transmit pin driver
- The non-transmitting devices use open drain output and only send 1s

Because the data inputs and outputs are connected together, a transmitting device will clock in its own data at the input pin (MRST for a master device, MTSR for a slave). In this way, any corruption is detected on the common data exchange line where the received data is not equal to the transmitted data.

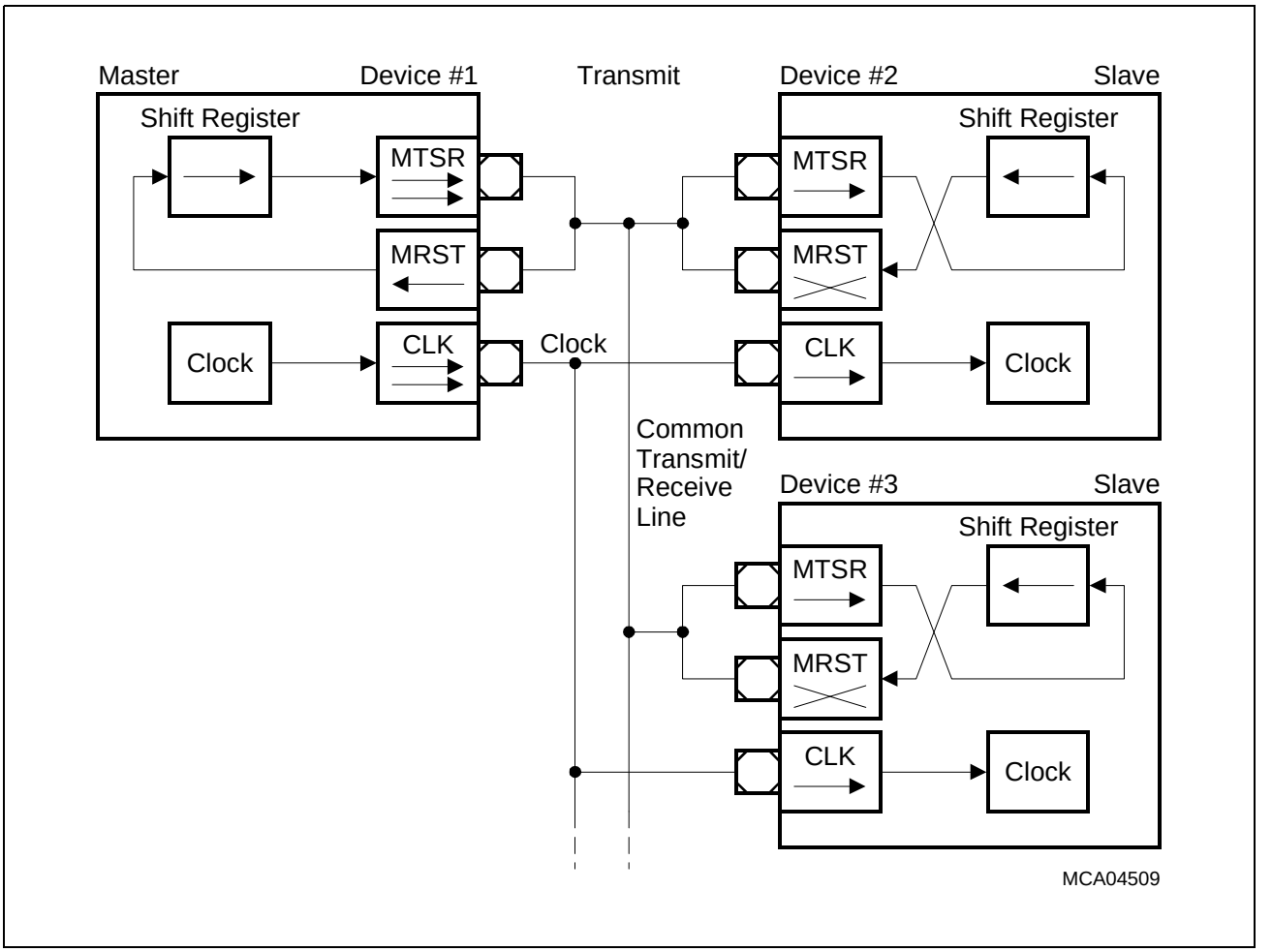


Figure 3-5 SSC Half-Duplex Configuration

3.1.2.4 Continuous Transfers

When the transmit interrupt request flag is set, it indicates that the Transmit Buffer (TB) is empty and is ready to be loaded with the next transmit data. If TB has been reloaded by the time the current transmission is finished, the data is immediately transferred to the shift register and the next transmission will start without any additional delay. On the data line there is no gap between the two successive frames. For example, two byte transfers would look the same as one word transfer. This feature can be used to interface with devices that can operate with or require more than 16 data bits per transfer. It is just a matter for software how long a total data frame length can be. This option can also be used e.g. to interface to byte-wide and word-wide devices on the same serial bus.

Note: Of course, this can only happen in multiples of the selected basic data width, because it would require disabling/enabling of the SSC to reprogram the basic data width on-the-fly.

3.1.2.5 Port Control

The SSC uses three pins to communicate with the external world. Pin SCLK serves as the clock line, while pins MRST (Master Receive/Slave Transmit) and MTSR (Master Transmit/Slave Receive) serve as the serial data input/output lines.

Operation of the SSC pins depends on the selected operating mode (master or slave). The direction of the port lines depends on the operating mode. The SSC will automatically use the correct alternate input or output line of the ports when switching modes. The direction of the port pins, however, must be programmed by the user (see **Section 3.3** for more details on port switching).

Using the open drain output feature of the port lines helps avoid bus contention problems and reduces the need for hard-wired hand-shaking or slave select lines. In this case, it is not always necessary to switch the direction of a port pin.

3.1.2.6 Baud Rate Generation

The serial channel SSC has its own dedicated 16-bit baud rate generator with 16-bit reload capability, allowing baud rate generation independent from the timers. In addition to **Figure 3-2**, **Figure 3-6** shows the baud rate generator of the SSC in more detail.

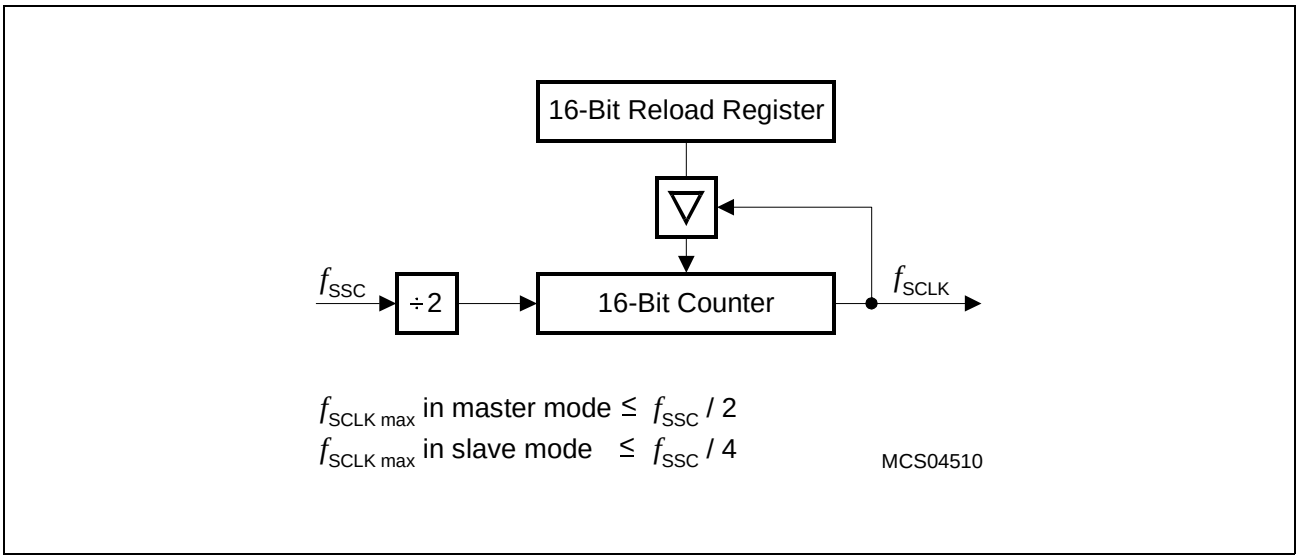


Figure 3-6 SSC Baud Rate Generator

The baud rate generator is clocked with the module clock f_{SSC}. The timer counts downwards. Register BR is the dual-function Baud Rate Generator/Reload register. Reading BR, while the SSC is enabled, returns the contents of the timer. Reading BR, while the SSC is disabled, returns the programmed reload value. In this mode, the desired reload value can be written to BR.

Note: Never write to BR while the SSC is enabled.

The formulas below calculate either the resulting baud rate for a given reload value, or the required reload value for a given baud rate:

$$\text{Baud rate}_{SSC} = \frac{f_{SSC}}{2 \times (
 + 1)} \qquad BR = \frac{f_{SSC}}{2 \times \text{Baud rate}_{SSC}} - 1$$

 represents the content of the reload register, taken as unsigned 16-bit integer while Baud rate$_{SSC}$ is equal to f_{SCLK} as shown in **Figure 3-6**.

The maximum baud rate that can be achieved when using a module clock of 40 MHz is 20 MBaud in master mode (with
 = 0000_H) and 10 MBaud in slave mode (with
 = 0001_H).

Table 3-1 lists some possible baud rates together with the required reload values and the resulting bit times, assuming a module clock of 40 MHz.

Table 3-1 Typical Baud Rates of the SSC (f_{SSC} = 40 MHz)

Reload Value	Baud Rate (= f_{SCLK})	Deviation
0000_H	20 MBaud (only in master mode)	0.0%
0001_H	10 MBaud	0.0%
0013_H	1 MBaud	0.0%
0018_H	800 kBaud	0.0%
0031_H	400 kBaud	0.0%
0063_H	200 kBaud	0.0%
$00C7_H$	100 kBaud	0.0%
$FFFF_H$	305.18 Baud	0.0%

3.1.2.7 Error Detection Mechanisms

The SSC is able to detect four different error conditions. Receive Error and Phase Error are detected in all modes, while Transmit Error and Baud Rate Error apply to slave mode only. When an error is detected, the respective error flag is set and an error interrupt request will be generated by activating the EIR line (see **Figure 3-7**). The error interrupt handler may then check the error flags to determine the cause of the error interrupt. The error flags are not reset automatically, but must be cleared by software after servicing. This allows servicing of some error conditions via interrupt, while others may be polled by software.

Note: The error interrupt handler must clear the associated (enabled) error flag(s) to prevent repeated interrupt requests.

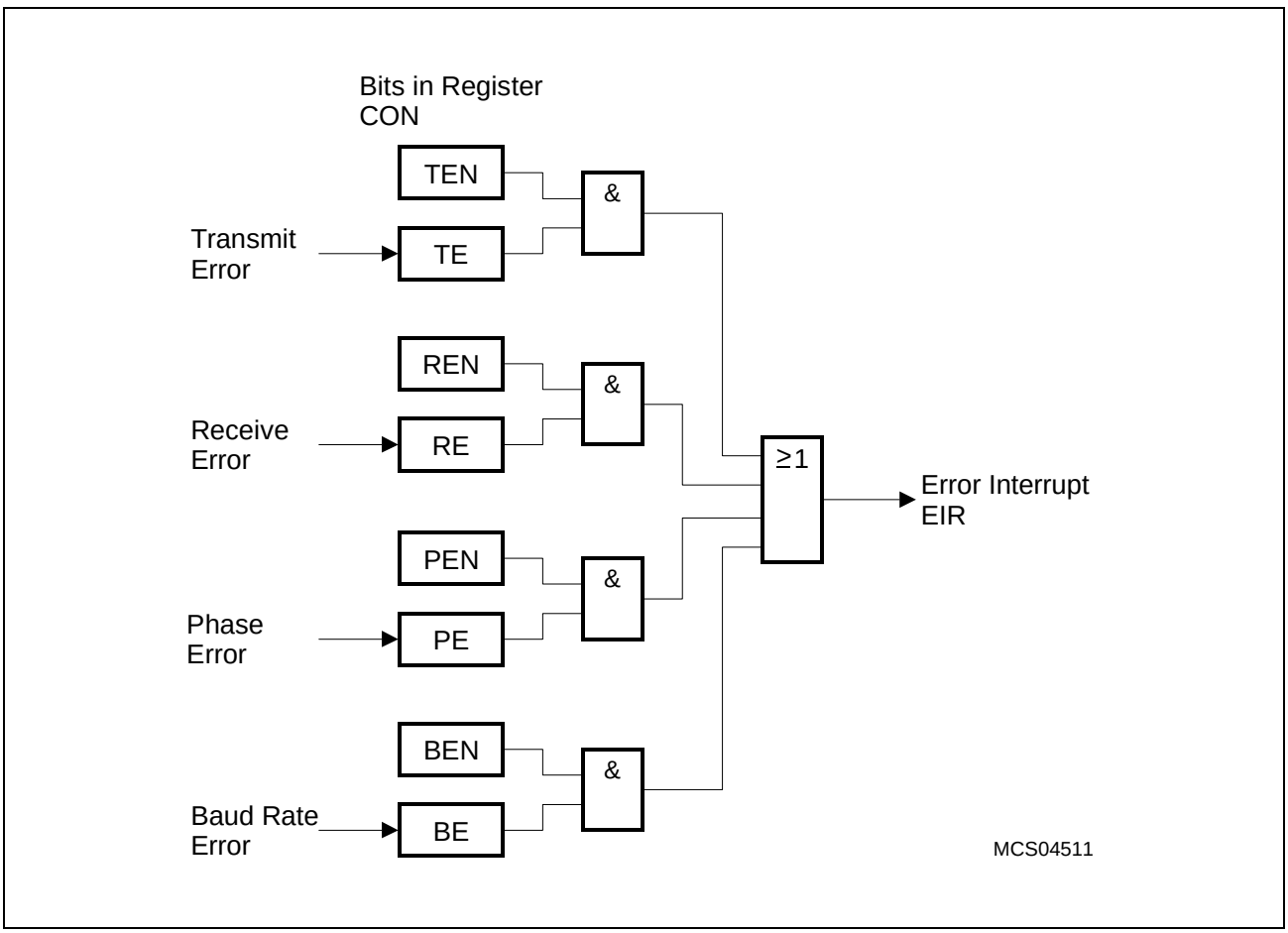


Figure 3-7 SSC Error Interrupt Control

A **Receive Error** (Master or Slave mode) is detected when a new data frame is completely received, but the previous data was not read out of the receive buffer register RB. This condition sets the error flag CON.RE and sets the error interrupt request line EIR, if enabled via CON.REN. The old data in the receive buffer RB will be overwritten with the new value and is unretrievably lost.

A **Phase Error** (Master or Slave mode) is detected when the incoming data at pin MRST (master mode) or MTSR (slave mode), sampled with the same frequency as the module clock, changes between one cycle before and two cycles after the latching edge of the shift clock signal SCLK. This condition sets the error flag CON.PE and the error interrupt request line EIR, if enabled via CON.PEN.

A **Baud Rate Error** (Slave mode) is detected when the incoming clock signal deviates from the programmed baud rate by more than 100%, meaning it is either more than double or less than half the expected baud rate. This condition sets the error flag CON.BE and the error interrupt request line EIR, if enabled via CON.BEN. Using this error detection capability requires that the slave's baud rate generator is programmed to the same baud rate as the master device. This feature detects false additional pulses or missing pulses on the clock line (within a certain frame).

Note: If this error condition occurs and bit CON.REN = 1 an automatic reset of the SSC will be performed in case of this error. This is done to re-initialize the SSC, if too few or too many clock pulses have been detected.

A **Transmit Error** (Slave mode) is detected when a transfer was initiated by the master (shift clock gets active), but the Transmit Buffer (TB) of the slave was not updated since the last transfer. This condition sets the error flag CON.TE and the error interrupt request line EIR, if enabled via CON.TEN. If a transfer starts while the transmit buffer is not updated, the slave will shift out the 'old' contents of the shift register, which is normally the data received during the last transfer. This may lead to the corruption of the data on the transmit/receive line in Half-duplex Mode (open drain configuration) if this slave is not selected for transmission. This mode requires that slaves not selected for transmission only shift out ones, thus, their transmit buffers must be loaded with $FFFF_H$ prior to any transfer.

Note: A slave with push/pull output drivers not selected for transmission, will normally have its output drivers switched. However, to avoid possible conflicts or misinterpretations, it is recommended to always load the slave's transmit buffer prior to any transfer.

The cause of an error interrupt request (receive, phase, baud rate, transmit error) can be identified by the error status flags in control register CON.

Note: In contrast to the error interrupt request line EIR, the error status flags CON.TE, CON.RE, CON.PE, and CON.BE, are not reset automatically upon entry into the error interrupt service routine, but must be cleared by software.

3.2 SSC Kernel Registers

Figure 3-8 shows all registers associated with the SSC Kernel.

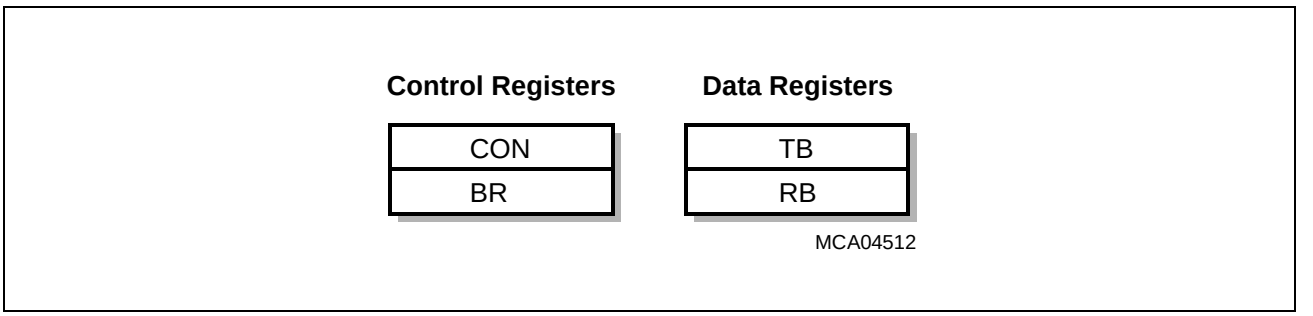


Figure 3-8 SSC Kernel Registers

Table 3-2 SSC Kernel Registers

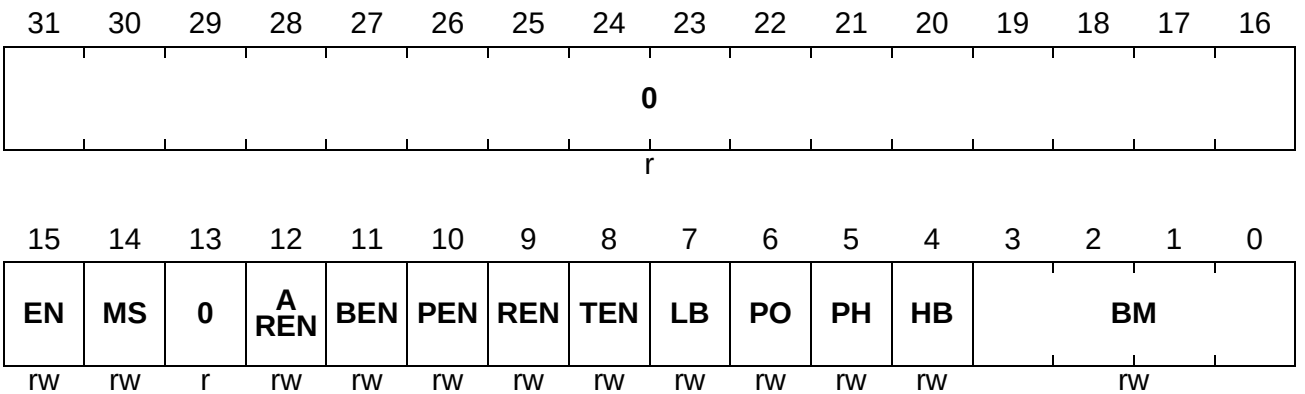
Register Short Name	Register Long Name	Offset Address	Description see
CON	Control Register	0010_H	**Page 3-16** **Page 3-18**
BR	Baud Rate Timer Reload Register	0014_H	**Page 3-20**
TB	Transmit Buffer Register	0020_H	**Page 3-21**
RB	Receive Buffer Register	0024_H	**Page 3-21**

Note: All SSC kernel register names described in this section will be referenced in other parts of the TC1775 User's Manual with the module name prefix "SSC0_" for the SSC0 interface and by "SSC1_" for the SSC1 interface.

The operating mode of the serial channel SSC is controlled by the control register CON. This register contains control bits for mode and error check selection, and status flags for error identification. Depending on bit EN, either control functions are enabled or status flags and master/slave control are enabled.

CON.EN = 0: Programming Mode

CON
Control Register **Reset Value: 0000 0000$_H$**

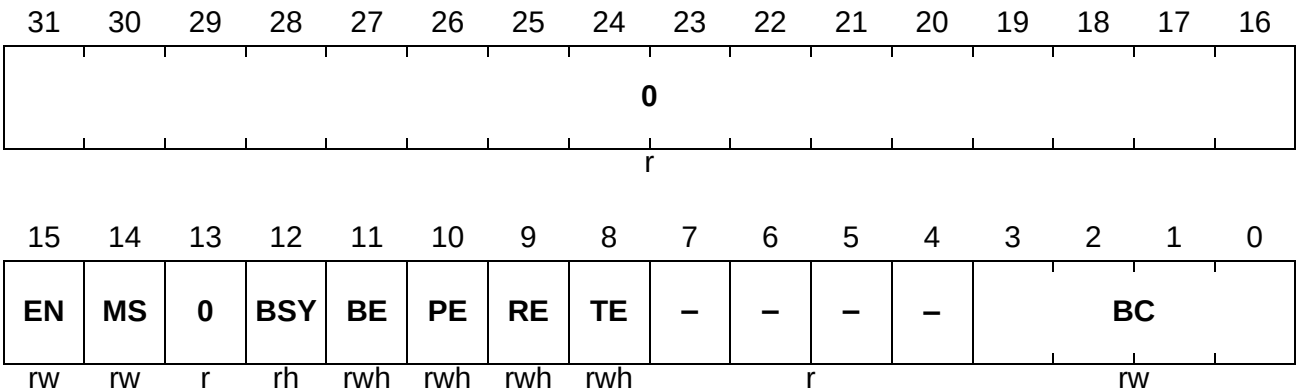


Field	Bits	Type	Description
BM	[3:0]	rw	**Data Width Selection** 0000 Reserved; do not use this combination. 0001 to 1111 Transfer Data Width is 2 … 16 bit (<BM> + 1)
HB	4	rw	**Heading Control** 0 Transmit/Receive LSB First 1 Transmit/Receive MSB First
PH	5	rw	**Clock Phase Control** 0 Shift transmit data on the leading clock edge, latch on trailing edge 1 Latch receive data on leading clock edge, shift on trailing edge
PO	6	rw	**Clock Polarity Control** 0 Idle clock line is low, the leading clock edge is low-to-high transition 1 Idle clock line is high, the leading clock edge is high-to-low transition

Field	Bits	Type	Description
LB	7	rw	**Loop Back Control** 0 Normal output 1 Receive input is connected with transmit output (Half-duplex Mode)
TEN	8	rw	**Transmit Error Enable** 0 Ignore transmit errors 1 Check transmit errors
REN	9	rw	**Receive Error Enable** 0 Ignore receive errors 1 Check receive errors
PEN	10	rw	**Phase Error Enable** 0 Ignore phase errors 1 Check phase errors
BEN	11	rw	**Baud Rate Error Enable** 0 Ignore baud rate errors 1 Check baud rate errors
AREN	12	rw	**Automatic Reset Enable** 0 No additional action upon a baud rate error 1 SSC is automatically reset on a baud rate error
MS	14	rw	**Master Select** 0 Slave Mode. Operate on shift clock received via SCLK 1 Master Mode. Generate shift clock and output it via SCLK
EN	15	rw	**Enable Bit = 0** Transmission and reception disabled. Access to control bits.
0	13, [31:16]	r	**Reserved**; returns 0 if read; should be written with 0.

CON.EN = 1: Operating Mode

CON
Control Register **Reset Value: 0000 0000$_H$**



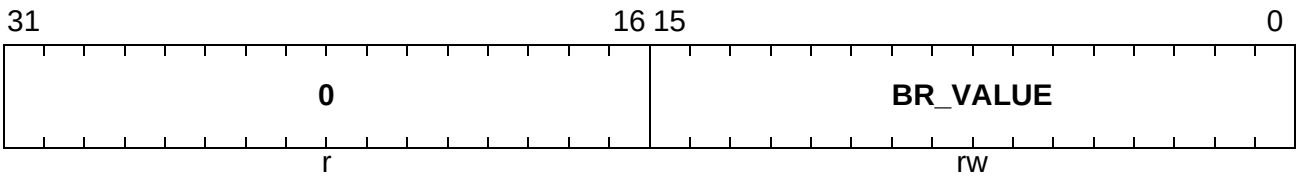
Field	Bits	Type	Description
BC	[3:0]	rh	**Bit Count Field** 0001 - 1111 Shift counter is updated with every shifted bit **Do not write to this field!**
TE	8	rwh	**Transmit Error Flag** 0 No error 1 Transfer starts with the slave's transmit buffer not being updated
RE	9	rwh	**Receive Error Flag** 0 No error 1 Reception completed before the receive buffer was read
PE	19	rwh	**Phase Error Flag** 0 No error 1 Received data changes around the sampling clock edge
BE	11	rwh	**Baud Rate Error Flag** 0 No error 1 More than factor 2 or 0.5 between slave's actual and expected baud rate
BSY	12	rh	**Busy Flag** Set while a transfer is in progress. **Do not write to this bit!**

Field	Bits	Type	Description
MS	14	rw	**Master Select Bit** 0 Slave Mode. Operate on shift clock received via SCLK. 1 Master Mode. Generate shift clock and output it via SCLK.
EN	15	rw	**Enable Bit = 1** Transmission and reception enabled. Access to status flags and M/S control.
–	[7:4]	–	**Reserved**
0	13, [31:16]	r	**Reserved**; returns 0 if read; should be written with 0.

Note: The target of an access to CON (control bits or flags) is determined by the state of CON.EN prior to the access, writing $C057_H$ to CON in programming mode (CON.EN = 0) will initialize the SSC (CON.EN was 0) and then turn it on (CON.EN = 1). When writing to CON, make sure that reserved locations receive all zeros.

The SSC baud rate timer reload register BR contains the 16-bit reload value for the baud rate timer.

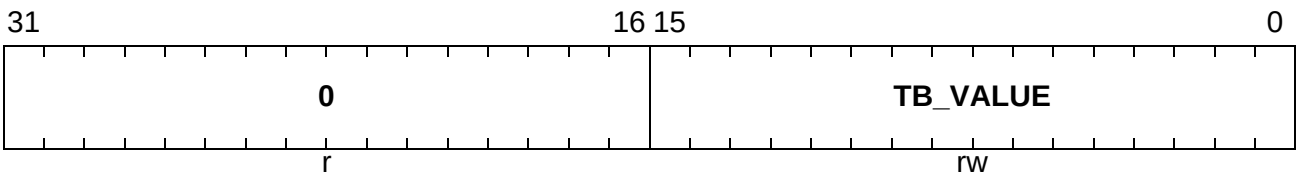
BR
Baud Rate Timer Reload Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
BR_VALUE	[15:0]	rw	**Baud Rate Timer/Reload Register Value** Reading BR returns the 16-bit content of the baud rate timer. Writing BR loads the baud rate timer reload register with BR_VALUE.
0	[31:16]	r	**Reserved**; returns 0 if read; should be written with 0.

The SSC transmitter buffer register TB contains the transmit data value.

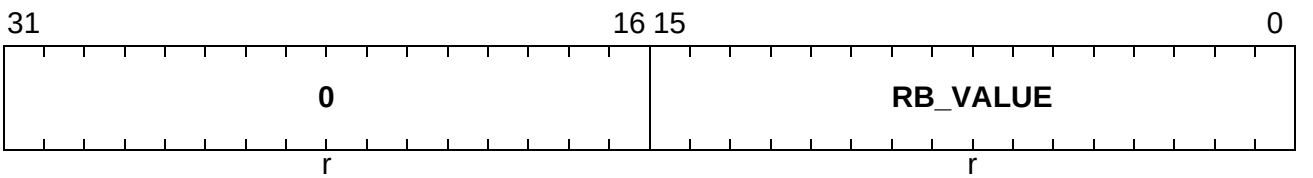
TB
Transmitter Buffer Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
TB_VALUE	[15:0]	rw	**Transmit Data Register Value** TB_VALUE is the data value to be transmitted. Unselected bits of TB are ignored during transmission.
0	[31:16]	r	**Reserved**; returns 0 if read; should be written with 0.

The SSC receiver buffer register RB contains the receive data value.

RB
Receiver Buffer Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
RB_VALUE	[15:0]	r	**Receive Data Register Value** RB contains the received data value RB_VALUE. Unselected bits of RB will be not valid and should be ignored.
0	[31:16]	r	**Reserved**; returns 0 if read; should be written with 0.

3.3 SSC0/SSC1 Module Implementation

This section describes the SSC0/SSC1 Module interfaces with the clock control, port connections, interrupt control, and address decoding.

3.3.1 Interfaces of the SSC Modules

Figure 3-9 shows the TC1775 specific implementation details and interconnections of the SSC0/SSC1 Modules. Each of the SSC Modules has three I/O lines located at Port 13. Each of the SSC Modules is further supplied by a separate clock control, interrupt control, address decoding, and port control logic.

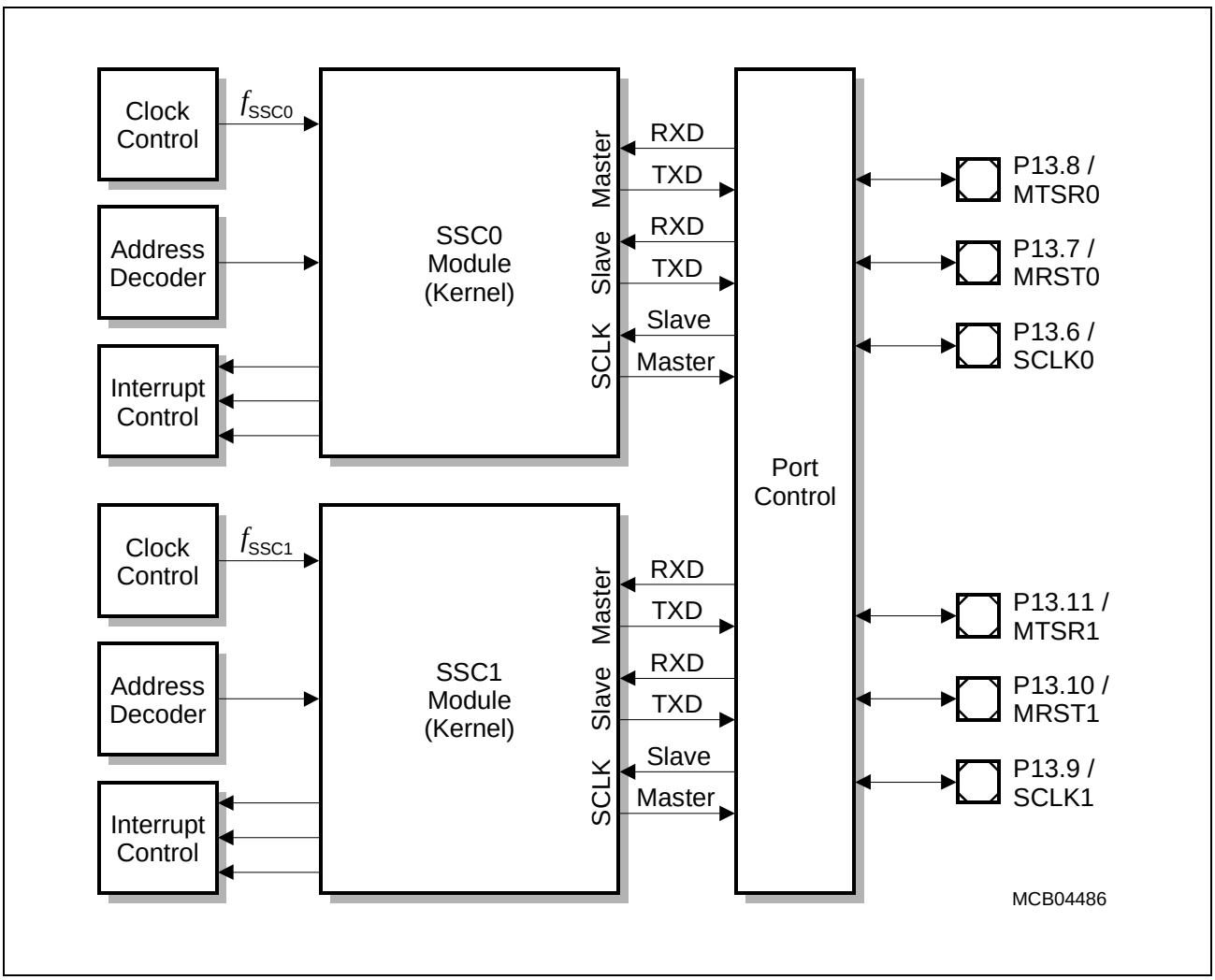


Figure 3-9 SSC0/SSC1 Module Implementation and Interconnections

3.3.2 SSC0/SSC1 Module Related External Registers

The figure below summarizes the module related external registers which are required for SSC0/SSC1 programming (see also **Figure 3-8** for the module kernel specific registers).

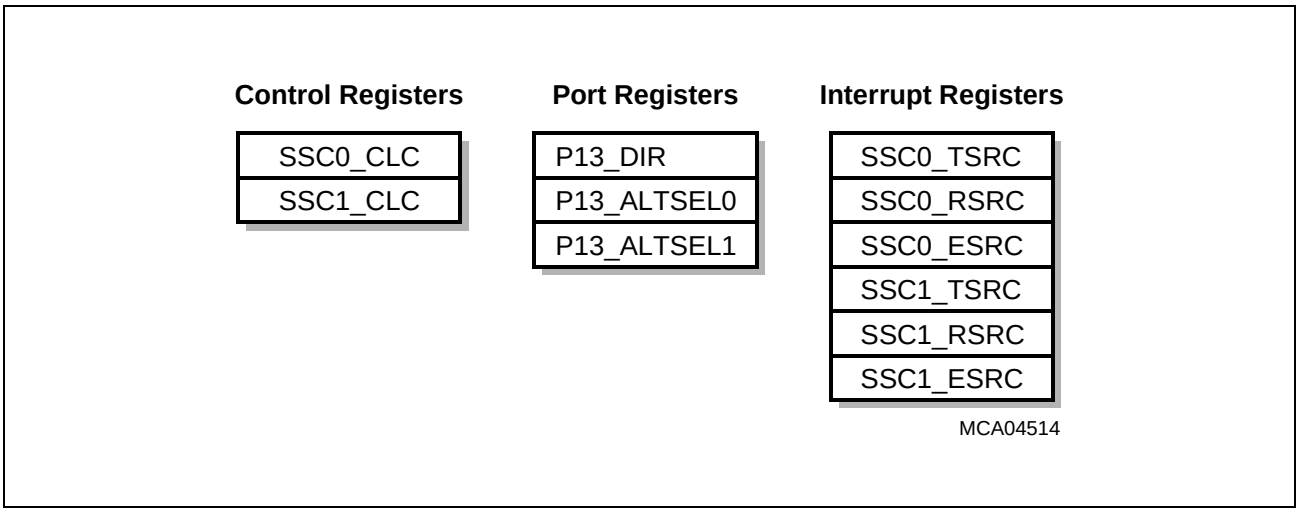


Figure 3-10 SSC0/SSC1 Implementation Specific Special Function Registers

3.3.2.1 Clock Control Registers

The clock control register allows the programmer to adapt the functionality and power consumption of an SSC Module to the requirements of the application. The diagram below shows the clock control register functionality implemented for the SSC Modules. SSC0_CLC is controlling the f_{SSC0} clock signal and SSC1_CLC is controlling the f_{SSC1} clock signal.

SSC0_CLC
SSC0 Clock Control Register **Reset Value: 0000 0002$_H$**
SSC1_CLC
SSC1 Clock Control Register **Reset Value: 0000 0002$_H$**

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RMC								0	0	FSOE	SBWE	EDIS	SPEN	DISS	DISR
rw								r	r	rw	w	rw	rw	r	rw

Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control of the module.
DISS	1	r	**Module Disable Status Bit** Bit indicates the current status of the module.
SPEN	2	rw	**Module Suspend Enable for OCDS** Used for enabling the suspend mode.
EDIS	3	rw	**External Request Disable** Used for controlling the external clock disable request.
SBWE	4	w	**Module Suspend Bit Write Enable for OCDS** Defines whether SPEN and FSOE are write protected.
FSOE	5	rw	**Fast Switch Off Enable** Used for fast clock switch off in OCDS suspend mode.
RMC	[15:8]	rw	**8-Bit Clock Divider Value in RUN Mode**
0	7, 6, [31:16]	r	**Reserved**; returns 0 if read; should be written with 0.

Note: After a hardware reset operation the SSC Modules are disabled.

3.3.2.2 Port Registers

The alternate functions, associated with the SSC0/SSC1 I/O lines, are controlled by the ALTSEL registers located in the ports. Two basic selections must be executed:

– Alternate function select by the port alternate select (ALTSEL) registers
– Direction control by the port direction (DIR) registers

The SSC0/SSC1 I/O lines are connected with Port 13. Therefore, P13_ALTSEL0 and P13_ALTSEL1 must programmed for the Port 13 pins required for the SSC Modules in the specific application.

Note: Bits marked with 'X' are not relevant for SSC operation.

P13_ALTSEL0
Port 13 Alternate Select Register 0 **Reset Value: 0000 0000$_H$**

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

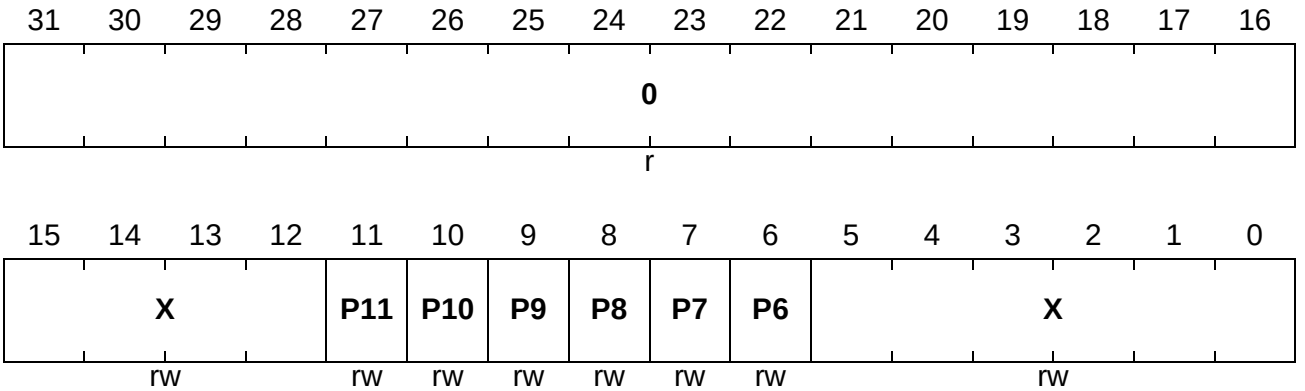
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X				P11	P10	P9	P8	P7	P6	X					
rw				rw	rw	rw	rw	rw	rw	rw					

P13_ALTSEL1
Port 13 Alternate Select Register 1 **Reset Value: 0000 0000$_H$**

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X				P11	P10	P9	P8	P7	P6	X					
rw				rw	rw	rw	rw	rw	rw	rw					

P13_DIR
Port 13 Direction Register **Reset Value: 0000 0000$_H$**



Bit 11-6 of the Port 13 Alternate Select Registers and the Port 13 Direction Register must be set depending on the operating mode, as shown in **Table 3-3**.

Note: The direction control bits are not affected automatically by the SSC0/SSC1 Module. It must be cleared/set by software for the required functionality (slave or master mode, Half-duplex or Full-duplex).

Table 3-3 SSC0/SSC1 I/O Line Selection and Setup

Module	Alternate Function	Port Line	Alternate Select Register Bits		Direction Registers		
			P13_ ALTSEL0	P13_ ALTSEL1	Mode	P13_DIR Bits	I/O
SSC0	SCLK0	P13.6	P6 = X	P6 = 1	Master	P6 = 1	Output
					Slave	P6 = 0	Input
	MRST0	P13.7	P7 = X	P7 = 1	Master	P7 = 0	Input
					Slave	P7 = 1	Output
	MTSR0	P13.8	P8 = 1	P8 = X	Master	P8 = 1	Output
					Slave	P8 = 0	Input
SSC1	SCLK1	P13.9	P9 = 1	P9 = X	Master	P9 = 1	Output
					Slave	P9 = 0	Input
	MRST1	P13.10	P10 = 1	P10 =X	Master	P10 = 0	Input
					Slave	P10 = 1	Output
	MTSR1	P13.11	P11 = 1	P11 = X	Master	P11 = 1	Output
					Slave	P11 = 0	Input

3.3.2.3 Interrupt Registers

The six interrupts of the SSC0 and SSC1 Module are controlled by the following service request control registers:

- SSC0_TSRC, SSC1_TSRC controls the transmit interrupts
- SSC0_RSRC, SSC1_RSRC controls the receive interrupts
- SSC0_ESRC, SSC1_ESRC controls the error interrupts

SSC0_TSRC
SSC0 Transmit Interrupt Service Request Control Register
SSC0_RSRC
SSC0 Receive Interrupt Service Request Control Register
SSC0_ESRC
SSC0 Error Interrupt Service Request Control Register
SSC1_TSRC
SSC1 Transmit Interrupt Service Request Control Register
SSC1_RSRC
SSC1 Receive Interrupt Service Request Control Register
SSC1_ESRC
SSC1 Error Interrupt Service Request Control Register

Reset Values: 0000 0000$_H$

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SETR	CLRR	SRR	SRE	TOS		0		SRPN							
w	w	rh	rw	rw		r		rw							

Field	Bits	Type	Description
SRPN	[7:0]	rw	**Service Request Priority Number**
TOS	[11:10]	rw	**Type of Service Control**
SRE	12	rw	**Service Request Enable**
SRR	13	rh	**Service Request Flag**
CLRR	14	w	**Request Clear Bit**
SETR	15	w	**Request Set Bit**

Field	Bits	Type	Description
0	[9:8], [31:16]	r	**Reserved**; returns 0 if read; should be written with 0.

Note: Further details on interrupt handling and processing are described in the "Interrupt System" chapter of the TC1775 System Units User's Manual.

3.3.3 SSC0/SSC1 Register Address Ranges

In the TC1775, the registers of the two SSC Modules are located in the following address ranges:

- SSC0 Module: Module Base Address = F000 $0A00_H$
  Module End Address = F000 $0AFF_H$
- SSC1 Module: Module Base Address = F000 $0B00_H$
  Module End Address = F000 $0BFF_H$
- Absolute Register Address = Module Base Address + Offset Address (offset addresses see **Table 3-2**)

4 TwinCAN Controller

This chapter describes the Twin Controller Area Network Module (TwinCAN) of the TC1775 in the following sections:

- Functional description of the TwinCAN Kernel (see **Section 4.1**)
- TwinCAN kernel register description of all TwinCAN Kernel specific registers (see **Section 4.2**)
- TC1775 implementation specific details and registers of the TwinCAN module (port connections and control, interrupt control, address decoding, and clock control (see **Section 4.3**).

*Note: The TwinCAN kernel register names described in **Section 4.2** will be referenced in other parts of the TC1775 User's Manual with the module name prefix "CAN_".*

4.1 TwinCAN Kernel Description

4.1.1 Overview

The TwinCAN module contains two Full-CAN nodes operating independently or exchanging data and remote frames via a gateway function. Transmission and reception of CAN frames are handled in accordance to CAN specification V2.0 part B (active). Each of the two Full-CAN nodes can receive and transmit standard frames with 11-bit identifiers as well as with extended frames with 29-bit identifiers.

Both CAN nodes share the TwinCAN module's resources to optimize the CAN bus traffic handling and to minimize the CPU load. The flexible combination of Full-CAN functionality and the FIFO architecture reduces the effort to fulfill the real-time requirements of complex embedded control applications. Improved CAN bus monitoring functionality as well as the increased number of message objects permit precise and convenient CAN bus traffic handling.

Depending on the application, each of the thirty-two message objects can be individually assigned to one of the two CAN nodes. Gateway functionality allows automatic data exchange between two separate CAN bus systems to reduce CPU load and improve the real time behavior of the entire system.

The bit timings for both 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 connect each CAN node to a bus transceiver.

Features

- CAN functionality conforms to CAN specification V2.0 B active.
- Dedicated control registers are provided for each CAN node.
- A data transfer rate up to 1MBaud is supported.
- Flexible and powerful message transfer control and error handling capabilities are implemented.
- Full-CAN functionality: 32 message objects can be individually
  - Assigned to one of the two CAN nodes
  - Configured as transmit or receive object
  - Participate in a 2,4,8,16 or 32 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
- Up to eight individually programmable interrupt nodes can be used.
- CAN Analyzer Mode for bus monitoring is implemented.

Figure 4-1 shows the functional units of the TwinCAN module.

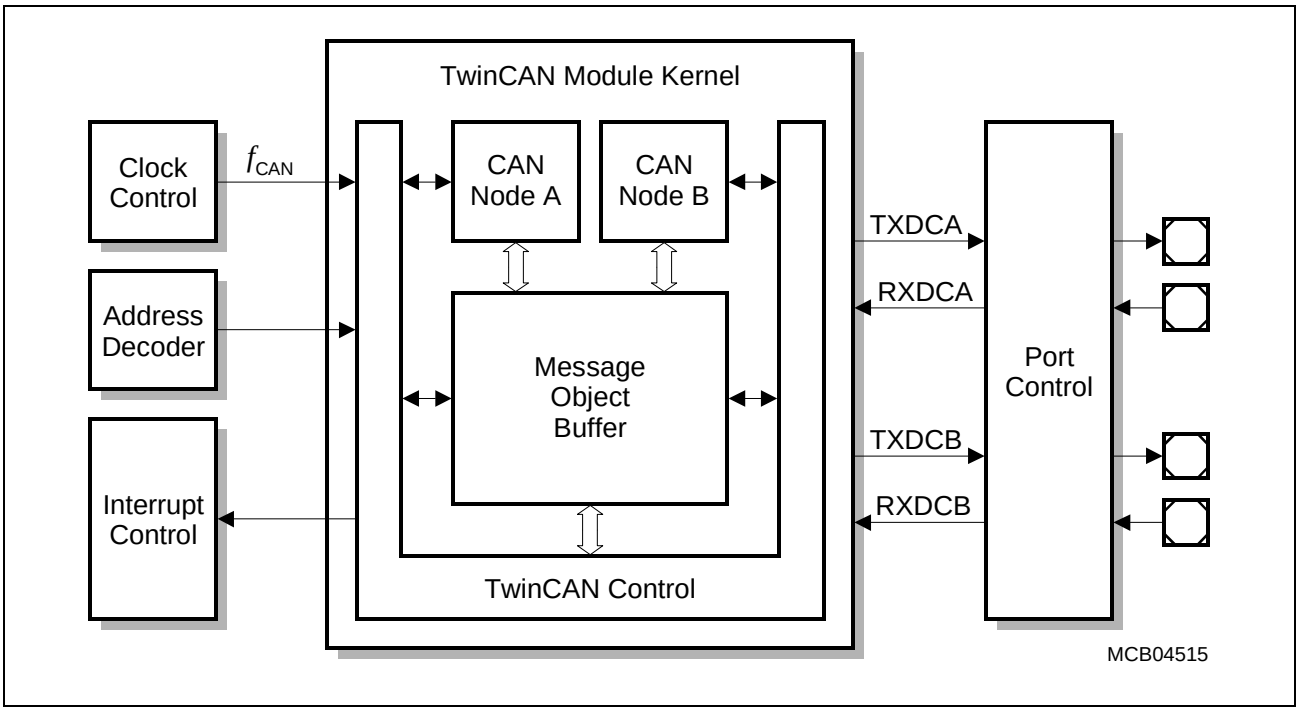


Figure 4-1 General Block Diagram of the TwinCAN Module

The TwinCAN kernel (**Figure 4-2**) is split into

- A global control shell, subdivided into the Initialization Logic, the Global Control and Status Logic and the Interrupt Request Compressor.
  - The Initialization Logic sets up all submodules after power-on or reset. After finishing the initialization of the node control logic and its associated message objects, the respective CAN node is synchronized with the connected CAN bus.
  - The Global Control and Status Logic informs the CPU of pending object transmit and receive interrupts and of recent transfer history.
  - The Interrupt Request Compressor condenses the interrupt requests from 72 sources (belonging to CAN node A and B) down to 8 interrupt nodes.
- A message buffer unit, containing the Message Buffers, the FIFO Buffer Management, the Gateway Control logic, and a message based Interrupt Request Generation unit.
  - The Message Buffer Unit stores up to 32 message objects of 8 bytes maximum data length. Each object has an identifier and its own set of control and status bits. After initialization, the Message Buffer Unit can handle reception and transmission of data without CPU supervision.
  - The FIFO Buffer Management stores the incoming and outgoing messages in a circular buffer and determines the next message to be processed by the CAN controller.
  - The Gateway Control logic transfers a message from CAN node A to CAN node B or vice versa.
  - The Interrupt Request Generation unit indicates the reception or transmission of an object specifically for each message object.

- Two separate CAN nodes, subdivided into a Bitstream Processor, a Bit Timing Control Unit, an Error Handling Control Logic, an Interrupt Request Generation unit and a Node Control Logic:
  - The Bitstream Processor performs data, remote, error and overload frames according to the ISO-DIS 11898 standard. The serial data flow between the CAN bus line, the input/output shift register and the CRC register is controlled as well as the parallel data flow between the I/O shift register and the message buffer unit.
  - The Bit Timing Control unit defines the sampling point in respect to propagation time delays and phase shift errors and performs the resynchronization.
  - The Error Handling Control Logic manages the Receive and the Transmit Error Counter. The CAN controller is set into an “error active”, “error passive” or “bus-off” state, depending on the contents in both timers.
  - The Interrupt Request Generation Unit globally signals the successful end of a message transmit or receive operation, as well as many kinds of transfer problems such as bit stuffing errors, format, acknowledge, CRC or bit state errors, and every change of the “CAN Bus Warning Level” or the “bus-off” state.
  - The Node Control Logic enables and disables the node specific interrupt sources, enters the CAN Analyzer Mode, and manages a global frame counter.

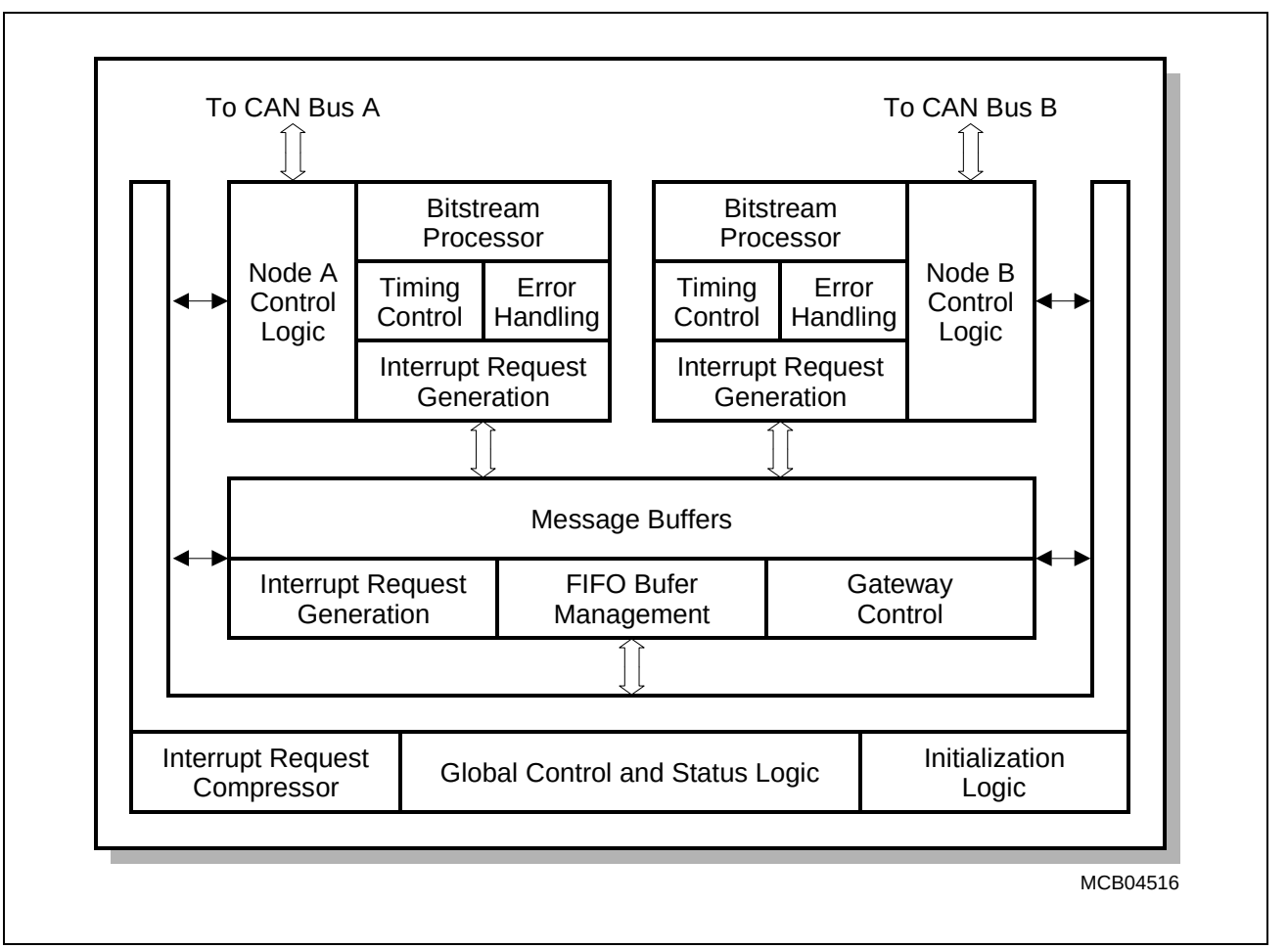


Figure 4-2 Detailed Block Diagram of the TwinCAN Kernel

4.1.2 TwinCAN Control Shell

4.1.2.1 Initialization Processing

After an external hardware reset or the occurrence of a “bus-off” event, the respective CAN controller node is logically disconnected from the associated CAN bus and does not participate in any message transfer. The “Disconnect Mode” is indicated by the ACR/BCR control register bit INIT = 1, which is automatically set in case of a reset or “bus-off” event. Furthermore, the “Disconnect Mode” can be also entered by setting bit INIT to 1 via software. While INIT is active, all message transfers between the affected TwinCAN node controller and its associated CAN bus are stopped and the bus output pin (TXDC) is held on ‘High’ level (recessive state).

After an external hardware reset, all control and message object registers are reset to their associated reset values. Upon an activation of the “bus-off” state or a write access to register ACR/BCR with INIT = 1, all respective control and message object registers hold their current values (except the error counters).

Resetting bit INIT to 0 without being in “bus-off” state starts a connect procedure, which must monitor at least one “Bus Idle” event (11 consecutive ‘recessive’ bits) on the associated CAN bus before the node is allowed to take part in CAN traffic again.

During the bus-off recovery sequence:

- The Receive and Transmit Error Counter within the Error Handling Control Logic are reset.
- 128 “Bus Idle” events (11 consecutive ‘recessive’ bits) must be detected, before the reconnect procedure can be initiated. The monitoring of the bus idle events is immediately started by hardware after entering the “bus-off” state. The number of “Bus Idle” events already detected is counted and indicated by the receive error counter.
- The reconnect procedure tests bit INIT by hardware after 128 “Bus Idle” events. If INIT is still set, the affected TwinCAN node controller waits until INIT is cleared and at least one “Bus Idle” event is detected on the CAN bus, before the node takes part in CAN traffic again. If INIT has been already cleared, the message transfer between the affected TwinCAN node controller and its associated CAN bus is immediately enabled.

4.1.2.2 Interrupt Request Compressor

The TwinCAN module is equipped with 32 × 2 message object specific interrupt request sources and 2 × 4 node control interrupt request sources. A request compressor condenses these 72 sources to 8 interrupt nodes reporting the interrupt requests of the TwinCAN module to the interrupt controller. Each request source is provided with an ‘Interrupt Node Pointer’, selecting the interrupt node to start the associated service routine to increase flexibility in interrupt processing. Each of the 8 interrupt nodes can trigger an independent interrupt routine with its own interrupt vector and its own priority.

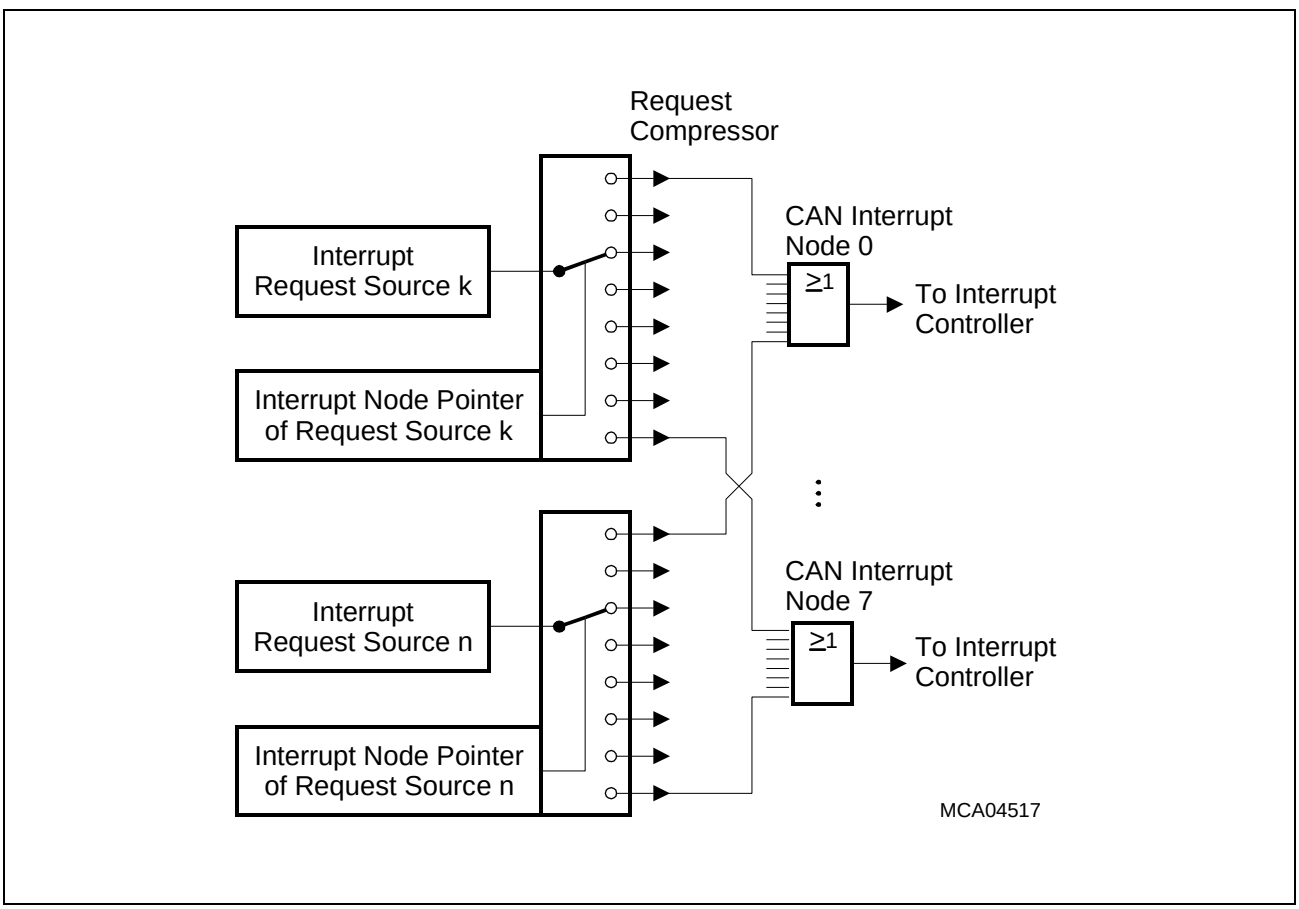


Figure 4-3 Interrupt Node Pointer and Interrupt Request Compressor

4.1.2.3 Global Control and Status Logic

The Receive Interrupt Pending Register (RXIPND) contains 32 individual flags indicating a pending receive interrupt for the associated message objects. Flag bit RXIPNDn is set by hardware if the corresponding message object has received a frame and the correlated interrupt request generation has been enabled by RXIEn = 10_B. RXIPNDn can be cleared by software by resetting bit INTPNDn in the corresponding message object control register MSGCTRn.

The Transmit Interrupt Pending Register (TXIPND) has the same layout as the RXIPND register and provides identical information about pending transmit interrupts.

4.1.3 CAN Node Control Logic

4.1.3.1 Overview

Each node is equipped with its own Node Control Logic to configure the global behavior and providing status information.

Configuration Mode is activated when the ACR/BCR register bit CCE is set to 1. This mode allows CAN bit timing parameters and the error counter registers to be modified.

CAN Analyzer Mode is activated when bit CALM in control register ACR/BCR is set to 1. In this operation mode, data and remote frames are monitored without an active participation in any CAN transfer (CAN transmit pin is held on recessive level). Incoming remote frames are stored in a corresponding ‘transmit message object’, while arriving data frames are saved in a matching ‘receive message object’.

In CAN Analyzer mode, the entire configuration information of the received frame is stored in the corresponding message object and can be evaluated by the CPU concerning their identifier, XTD bit information and data length code (ID and DLC optionally if the Remote Monitoring Mode is active, RMM = 1). Incoming frames are not acknowledged and no error frames are generated. Neither remote frames are answered by the corresponding data frame nor data frames can be transmitted by setting TXRQ, if CAN Analyzer Mode is enabled. Receive interrupts are generated (if enabled) for all error free received frames and the respective remote pending (bit RMTPND) is set in case of received remote frames.

The node specific interrupt configuration is also defined by the Node Control Logic via the ACR/BCR register bits SIE, EIE and LECIE:

- If control bit SIE is set to 1, a status change interrupt occurs when the ASR/BSR register has been updated (by each successfully completed message transfer).
- If control bit EIE is set to 1, an error interrupt is generated when a “bus-off” condition has been recognized or the ‘Error Warning Level’ has been exceeded or underrun.
- If control bit LECIE is set to1, a last error code interrupt is generated when an error code is set in bit field LEC in the status registers ASR or BSR.

The Status Register (ASR/BSR) provides an overview about the current state of the respective TwinCAN node:

- Flag TXOK is set when a message has been transmitted successfully and has been acknowledged by at least one other CAN node.
- Flag RXOK indicates an error-free reception of a CAN bus message.
- Bit field LEC indicates the last error occurred on the CAN bus. Stuff, form, and CRC errors as well as bus arbitration errors (Bit0, Bit1) are reported.
- Bit EWRN is set when at least one of the error counters in the Error Handling Control Logic has reached the error warning limit (default value 96).

- Bit BOFF is set when the transmit error counter has exceeded the error limit of 255 and the respective TwinCAN node controller has been logically disconnected from the associated CAN bus.

The CAN frame counter can be used to check the transfer sequence of message objects or to obtain information about the time instant a frame has been transmitted or received from the associated CAN bus. CAN frame counting is performed by a 16-bit counter, controlled by register AFCR/BFCR. Bit field CFCMD defines the operation mode and the trigger event incrementing the frame counter:

- After correctly transmitted frames
- After correctly received frames
- After a foreign frame on the CAN bus (not transmitted/received by the CAN node itself)
- At beginning of a new bit time

The captured frame counter value is copied to the CFCVAL field of the associated MSGCTRn register at the end of the monitored frame transfer. Flag CFCOV is set on a frame counter overflow condition ($FFFF_H$ to 0000_H) and an interrupt request is generated if bit CFCIE is set to1.

4.1.3.2 Timing Control Unit

According to ISO-DIS 11898 standard, a CAN bit time is subdivided into different segments (**Figure 4-4**). 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 DIV8X, both controlling the baud rate prescaler (see bit timing register ABTR/BBTR). The baud rate prescaler is driven by the TwinCAN module clock f_{CAN}.

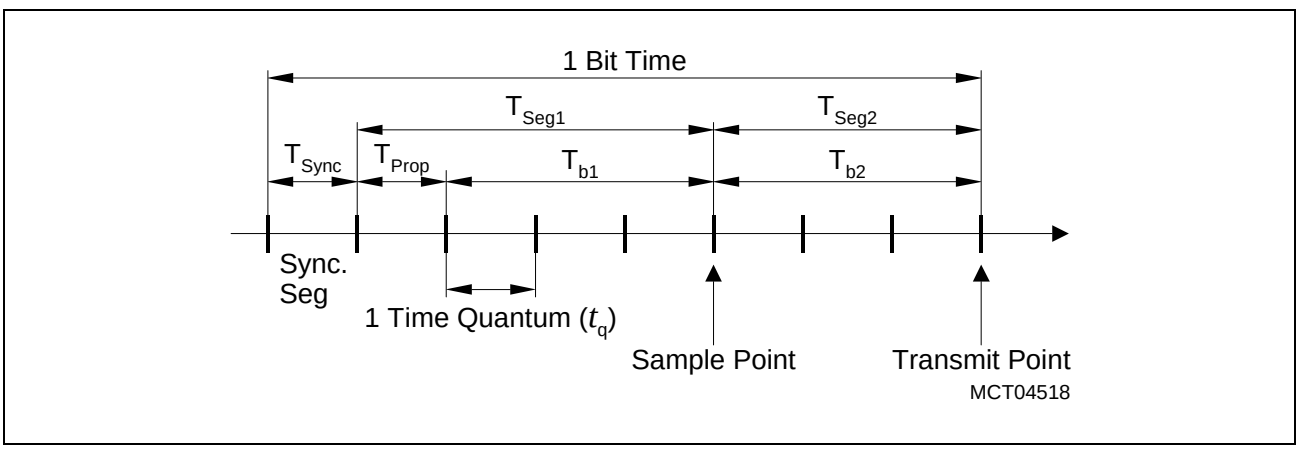


Figure 4-4 CAN Bus Bit Timing Standard

The Synchronization Segment (T_{Sync}) allows a phase synchronization between transmitter and receiver time base. The Synchronization Segment length is always one 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} must 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 bit timing register ABTR/BBTR. The Propagation Time Segment and the Phase Buffer Segment 1 are combined to parameter T_{Seg1}, which is defined by the value TSEG1 in the respective bit timing register ABTR/BBTR. A minimum of 3 time quanta are requested by the ISO standard. Parameter T_{Seg2}, which is defined by the value of TSEG2 in the bit timing register ABTR/BBTR, covers the Phase Buffer Segment 2. A minimum of 2 time quanta are 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 be less than 8 time quanta.

Note: The access to bit timing register ABTR/BBTR is only enabled if bit CCE in control register ACR/BCR is set to 1.

Calculation of the bit time:

t_q	= (BRP + 1) / f_{CAN}	if DIV8X = 0
	= (BRP + 1) / 8 × f_{CAN}	if DIV8X = 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 must 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 which a bit time may be shortened or lengthened by one resynchronization. The value of SJW is programmed in the ABTR/BBTR registers.

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.

df_{CAN}	≤ min (T_{b1}, T_{b2}) / 2 × (13 × bit time - T_{b2})	AND
df_{CAN}	≤ T_{SJW} / 20 × bit time	

4.1.3.3 Bitstream Processor

Based on the objects in the message buffer, the Bitstream Processor generates the remote and data frames to be transmitted via the CAN bus. It controls the CRC generator and adds the checksum information to the new remote or data frame. After including the ‘Start of Frame Bit’ and the ‘End of Frame Field’, the Bitstream Processor starts the CAN bus arbitration procedure and continues with the frame transmission when the bus was found in idle state. While the data transmission is running, the Bitstream Processor monitors continuously the I/O line. If (outside the CAN bus arbitration phase or the acknowledge slot) a mismatch is detected between the voltage level on the I/O line and the logic state of the bit currently sent out by the transmit shift register, a ‘Last Error’ interrupt request is generated and the error code is indicated by bit field LEC in status register ASR/BSR.

An incoming frame is verified by checking the associated CRC field. When an error has been detected, the ‘Last Error’ interrupt request is generated and the associated error code is presented in status register ASR/BSR. Furthermore, an error frame is generated and transmitted on the CAN bus. After decomposing a faultless frame into identifier and data portion, the received information is transferred to the message buffer executing remote and data frame handling, interrupt generation and status processing.

4.1.3.4 Error Handling Logic

The Error Handling Logic is responsible for the fault confinement of the CAN device. Its two counters, the Receive Error Counter and the Transmit Error Counter (control registers AECNT and BECNT), are incremented and decremented by commands from the Bitstream Processor. If the Bitstream 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 registers AECNT, BECNT. According to the values of the error counters, the CAN controller is set into the states “error active”, “error passive” and “bus-off”.

The CAN controller 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 is equal to or exceeds the “bus-off” limit of 256. This state is reported by flag BOFF in the ASR/BSR status register. The device remains in this state until the “bus-off” recovery sequence is finished. Additionally, the bit EWRN in the ASR/BSR status register 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 AECNT and BECNT. Bit EWRN is reset if both error counters fall below the error warning limit again.

4.1.3.5 Node Interrupt Processing

Each CAN node is equipped with 4 interrupt sources supporting the following:

- Global transmit/receive logic,
- CAN frame counter,
- Error reporting system.

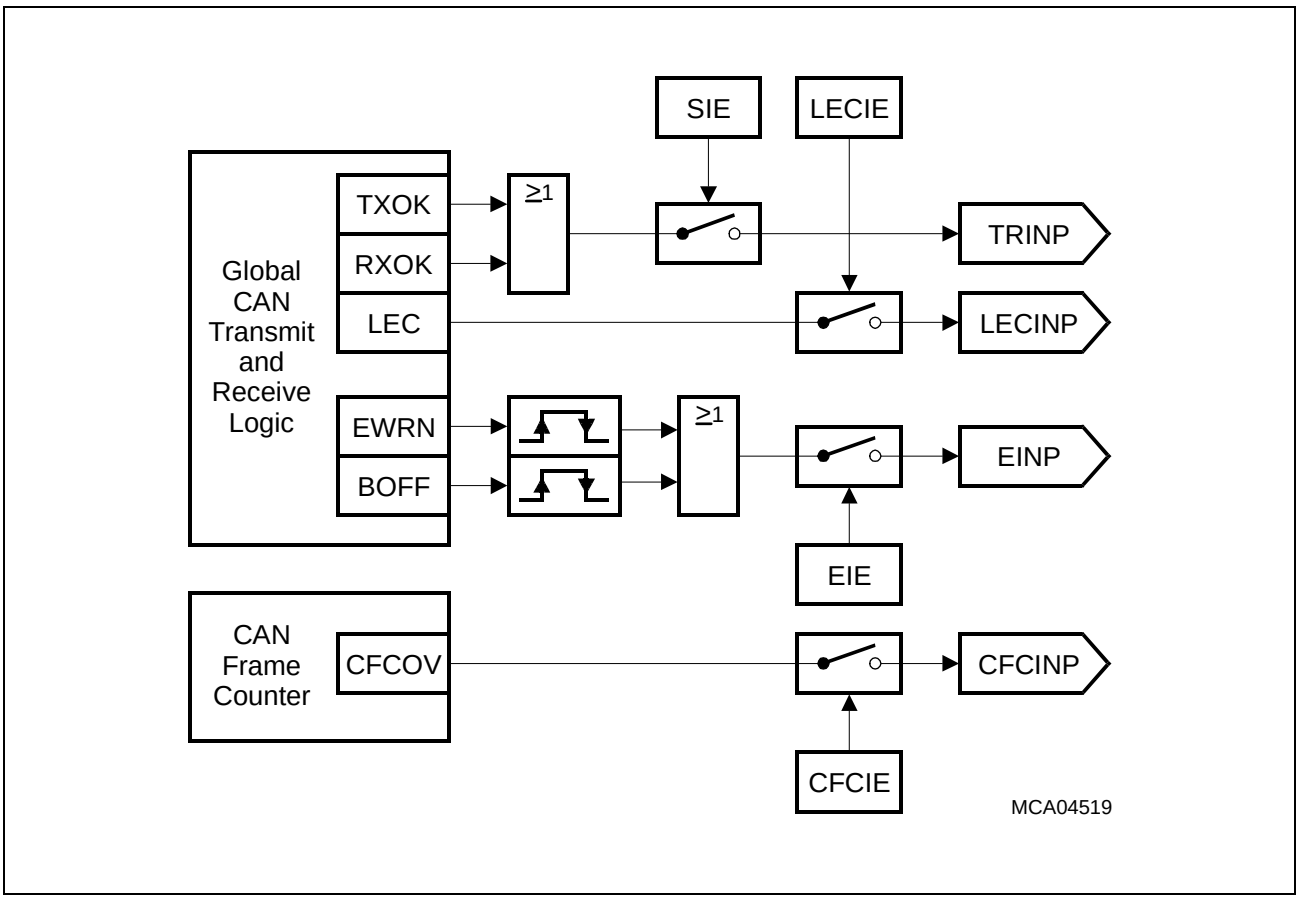


Figure 4-5 Node Specific Interrupt Control

If enabled by bit SIE = 1 in the ACR/BCR register, the global transmit/receive logic generates an interrupt request, if the node status register (ASR/BSR) is updated after finishing a faultless transmission or reception of a message object. The associated interrupt node pointer is defined by bit field TRINP in control register AGINP/BGINP.

An error is reported by a ‘Last Error Code’ interrupt request, if activated by LECIE = 1 in the ACR/BCR register. The corresponding interrupt node pointer is defined by bit field LECINP in control register AGINP/BGINP.

The CAN frame counter creates an interrupt request upon an overflow, when the AFCR/BFCR control register bit CFCIE is set to 1. Bit field CFCINP, located also in the AGINP/BGINP control register, selects the corresponding interrupt node pointer.

The error logic monitors the number of CAN bus errors and sets or resets an ‘Error Warning Bit’ (EWRN) according to the value in the error counters. If bit EIE in control

register ACR/BCR is set to 1, an interrupt request is generated on any modification of bits EWRN and BOFF. The associated interrupt node pointer is defined by bit field EINP in control register AGINP/BGINP.

4.1.3.6 Message Interrupt Processing

Each message object is equipped with two interrupt request sources indicating the successful end of a message transmission or reception.

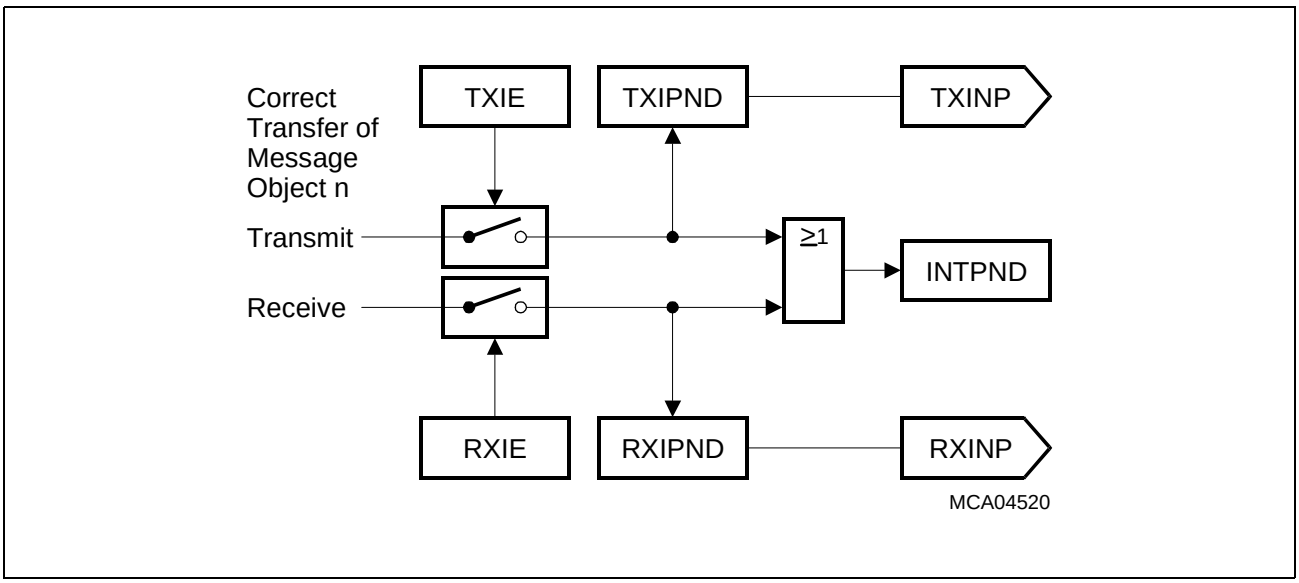


Figure 4-6 Message Specific Interrupt Control

The message-based transfer interrupt sources are enabled, if bit TXIE or RXIE in the associated message control register MSGCTRn are set to 10_B. The associated interrupt node pointers are defined by bit fields RXINP and TXINP in message configuration register MSGCFGn.

4.1.3.7 Interrupt Indication

The AIR/BIR register provides an INTID bit field indicating the source of the pending interrupt request with the highest internal priority (lowest message object number). The type of the monitored interrupt requests considered by bit field INTID can be selected by registers AIMR0/AIMR4 and BIMR0/BIMR4 containing a mask bit for each interrupt source. If no interrupt request is pending, all bits of AIR/BIR are cleared. The interrupt requests INTPNDn must be cleared by software.

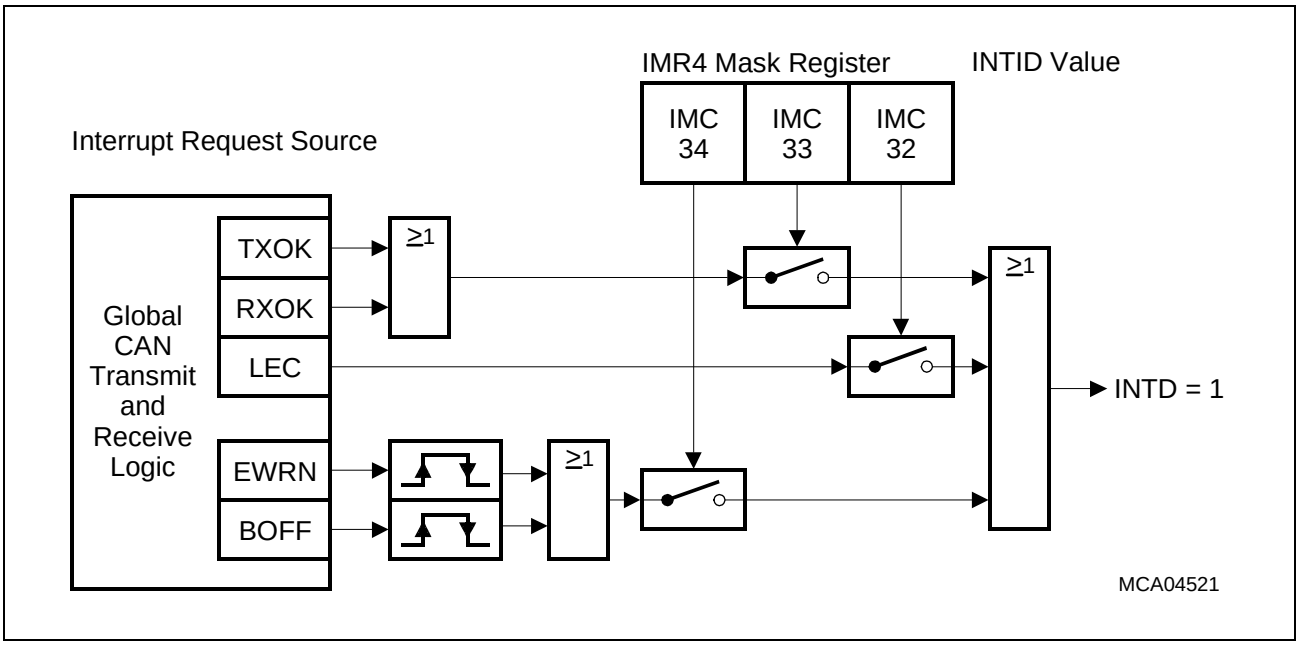


Figure 4-7 INTID Mask for Global Interrupt Request Sources

Registers AIMR0/4 and BIMR0/4 contain a mask bit for each interrupt source (AIMR0/ BIMR0 for message-specific interrupt sources and AIMR4/BIMR4 for the node-specific interrupt sources). If a mask bit is reset, the corresponding interrupt source is not taken into account for the generation of the INTID value.

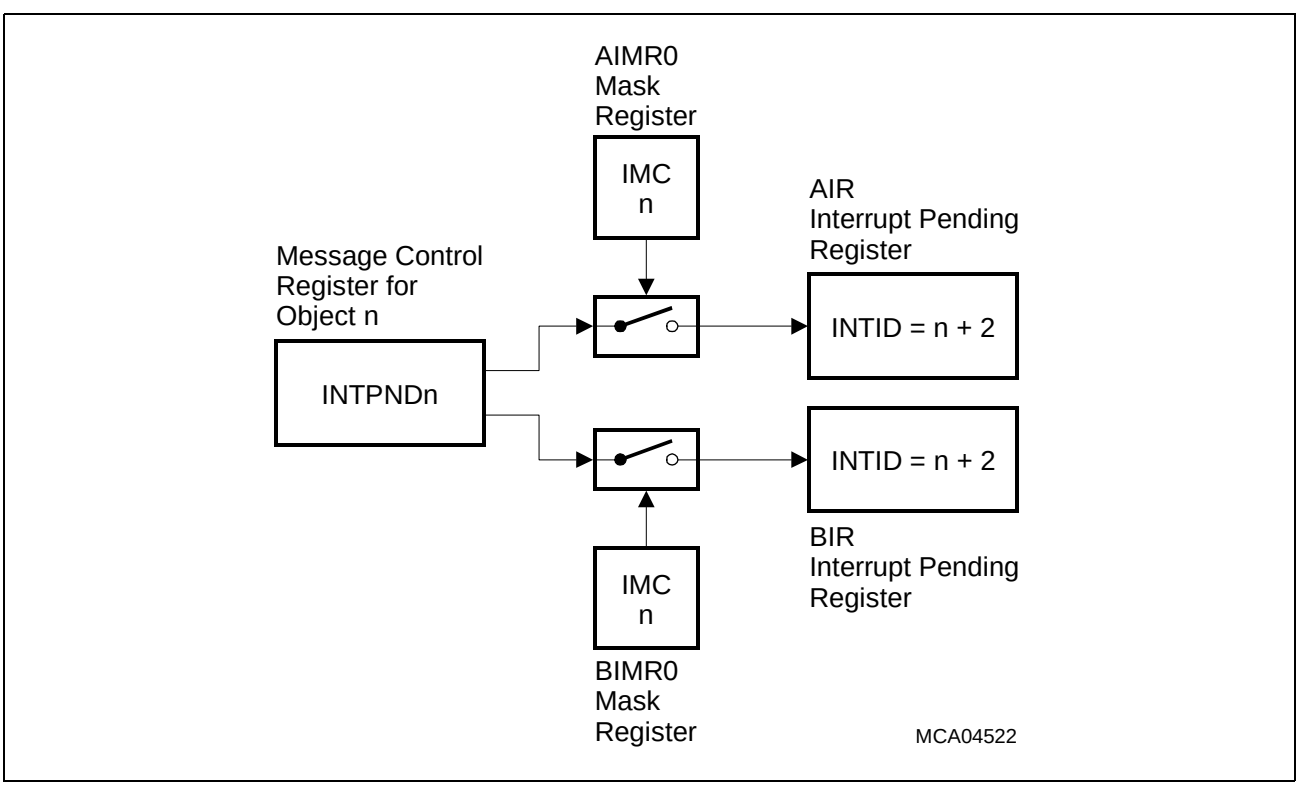


Figure 4-8 INTID Mask for Message Interrupt Request Sources

4.1.4 Message Handling Unit

A ‘Message Object’ is the basic information unit exchanged between the CPU and the CAN controller. Thirty-two message objects are provided by the internal CAN memory. Each of these objects has an identifier, its own set of control and status bits, and a separate data area. Each message object covers 32 bytes of internal memory subdivided into control registers and data storage as illustrated in **Figure 4-9**.

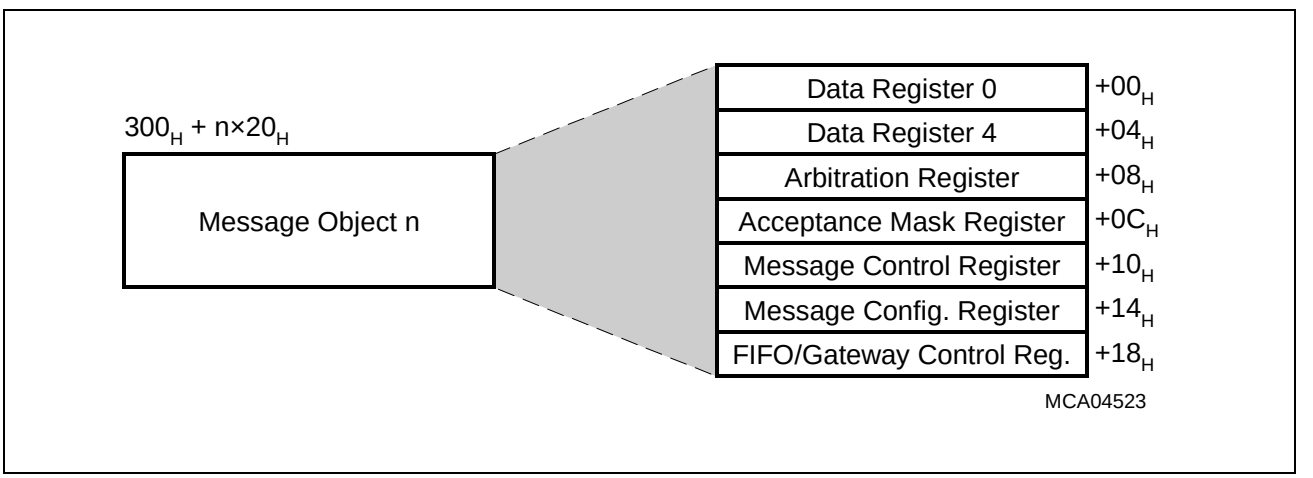


Figure 4-9 Structure of a Message Object

In “Normal Operation Mode”, each message object is associated with one CAN node. Only in “Shared Gateway Mode”, a message object can be accessed by both TwinCAN nodes.

In order to be considered by the respective CAN node control logic, the message object must be declared valid in its associated message control register (bit MSGVAL).

When a message object is initialized by the CPU, bit field MSGVAL in message control register MSGCTRn should be reset, thus, inhibiting a read or write access of the TwinCAN node controller to the associated register and data buffer storage. Afterwards, the message identifier and operation mode (transmit, receive) must be defined. If a successful transmission and/or reception of a message object should be followed by the execution of an interrupt service routine, the respective bit fields TXIE and RXIE must be set and the interrupt pending indicator (bit field INTPND) should be reset.

If the automatic response of an incoming remote frame with matching identifier is not requested, the respective transmission message object should be configured with CPUUPD = 10_B.

As soon as bit field MSGVAL is set to 10_B, the respective message object is operable and can taken into account by the associated TwinCAN node controller.

4.1.4.1 Arbitration and Acceptance Mask Register

The Arbitration Register (MSGARn) is used to filter the incoming messages and to provide the outgoing messages with an identifier. The Acceptance Mask Register (MSGAMRn) may be used to disable some identifier bits of an incoming message for the acceptance test.

The identifier of a received message is compared (bitwise XOR) to the identifiers of all message objects stored in the internal CAN controller memory. The compare operation starts at object 0 and takes into account all objects with:

- A valid message flag (MSGVAL = 10_B),
- A suitable NODE declaration (register MSGCFGn),
- A cleared DIR control bit (receive message object) for data frame reception,
- DIR = 1 (transmit message object) for remote frame reception,
- A matching identifier length declaration (XTD = 1 marks extended 29-bit identifiers, XTD = 0 indicates standard 11-bit identifiers).

The result of the compare operation is bit-by-bit ANDED with the contents of the Acceptance Mask Register (**Figure 4-10**). If concordance is detected, the received message is stored into the CAN controller's message object. The compare operation is finished after analyzing message object 31.

Note: Depending on the allocated identifiers and the corresponding mask register contents, multiple message objects may fulfill the selection criteria described above. In this case, the received frame is stored in the appropriate message object with the lowest message number.

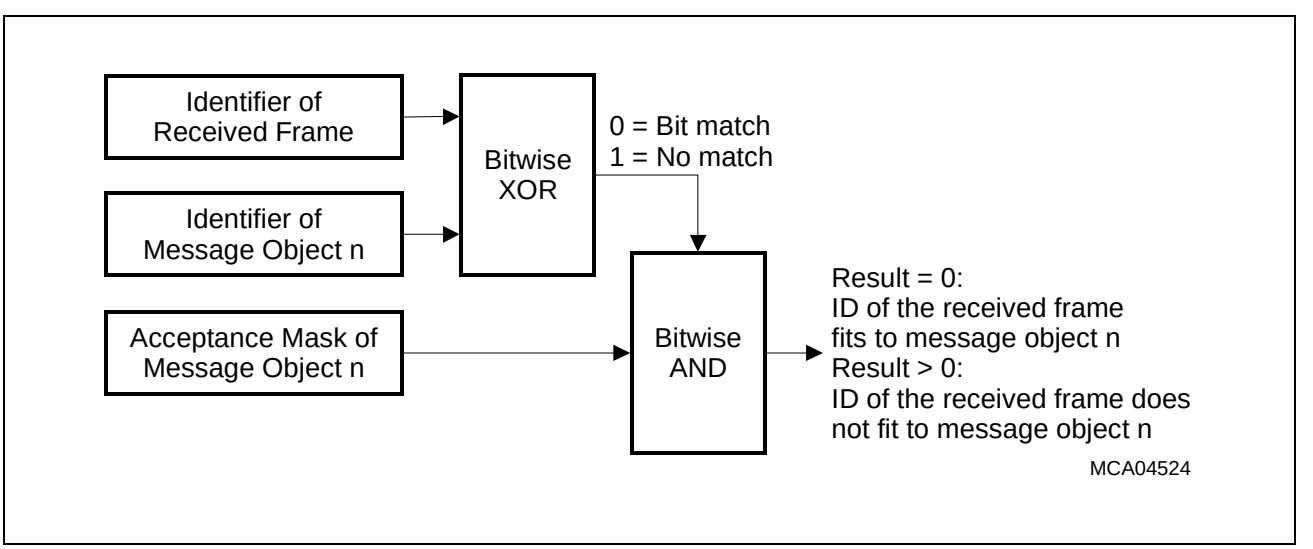


Figure 4-10 Acceptance Filtering for Received Message Identifiers

4.1.4.2 Handling of Remote and Data Frames

Message objects can be set up for transmit or receive operation according to the selected value for control bit DIR. The influence of the message object type on the associated TwinCAN node controller concerning to the generation or reception of remote and data frames is illustrated in **Table 4-1**.

Table 4-1 Handling of Remote and Data Frames

	A transmission request (TXRQ = 10_B) for this message object generates …	**If a data frame with matching identifier is received …**	**If a remote frame with matching identifier is received …**
Receive Object (receives data frames, transmits remote frames, control bit DIR = 0)	… a remote frame. The requested data frame is stored in this message object on reception.	… the data frame is stored in this message object.	… the remote frame is NOT taken into account.
Transmit Object (transmits data frames, receives remote frames, control bit DIR = 1)	… a data frame based upon the information stored in this message object.	… the data frame is NOT stored.	… the remote frame is stored in this message object and RMTPND and TXRQ are set to 10_B. A data frame, based upon the information stored in this message object, is generated automatically if CPUUPD is set to 01_B.

4.1.4.3 Handling of Transmit Message Objects

A message object with direction flag DIR = 1 (message configuration register MSGCFGn) is handled as a transmit object.

All message objects with bit field MSGVAL = 10_B are operable and can taken into account by the TwinCAN node controller operation described below.

During the initialization phase, the ‘transmit request’ bit field (TXRQ), the ‘new information’ bit field (NEWDAT) should be reset to 01_B and the ‘update in progress by CPU’ bit field (CPUUPD) in register MSGCTRn should be reset to 10_B. The message bytes to be transmitted are written into the data partition of the message object (MSGDRn0, MSGDRn4). The number of message bytes to be transmitted must be written to bit field DLC in register MSGCFGn. The selected identifier must be written to register MSGARn. Then, bit field NEWDAT in register MSGCTRn should be set to 10_B and bit field CPUUPD should be reset to 01_B by the CPU.

When ‘Remote Monitoring Mode’ is enabled (RMM = 1 in MSGCFGn), the identifier and the data length code of a received remote frame will be copied to the corresponding transmit message object, if an acceptable identifier was found during the compare and mask operation with all CAN message objects. The copy procedure may change the identifier in the transmit message object if some MSGAMRn mask register bits have been set to 0.

As long as bit field MSGVAL in register MSGCTRn is set to 10_B, the reception of a remote frame with matching identifier automatically sets bit field TXRQ to 10_B. Simultaneously, bit field RMTPND in register MSGCTRn is set to 10_B to indicate the reception of an accepted remote frame. Alternatively, TXRQ may be set by the CPU via a write access to register MSGCTRn. If the transmit request bit field TXRQ is found at 10_B (while MSGVAL = 10_B and CPUUPD = 01_B) by the appropriate CAN controller node, a data frame based upon the information stored in the respective transmit message object is generated and transferred automatically when the associated CAN bus becomes idle.

If bit field CPUUPD in register MSGCTRn is set to 10_B, the automatic transmission of a message object is prohibited and flag TXRQ is not evaluated by the respective TwinCAN node controller. The CPU can release the pending transmission by clearing CPUUPD. This allows the user to listen to the bus and to answer remote frames under software control.

When the data partition of a transmit message object must be updated by the CPU, bit field CPUUPD in message control register MSGCTRn should be set to 10_B, inhibiting a read or write access of the associated TwinCAN node controller. If a remote frame with an accepted identifier arrives during the update of a message object’s data storage, bit fields TXRQ and RMTPND are automatically set to 10_B and the transmission of the corresponding data frame is pending until CPUUPD is reset again.

If several valid message objects with pending transmission request are noticed by the associated TwinCAN node controller, the contents of the message object with the lowest message number is transmitted first.

NEWDAT is internally reset by the respective TwinCAN node controller when the contents of the selected message object's data registers are copied to the bitstream processor. RMTPND and TXRQ are automatically reset when the message object has been successfully transmitted.

The captured value of the frame counter is copied to bit field CFCVAL in register MSGCTRn and a transmit interrupt request is generated (INTPNDn and TXIPNDn are set) if enabled by TXIE = 10_B. Then the Frame Counter is incremented by one if enabled in control register AFCR/BFCR.

When a data frame with matching identifier is received, it is ignored by the respective transmit object and is not indicated by any interrupt request.

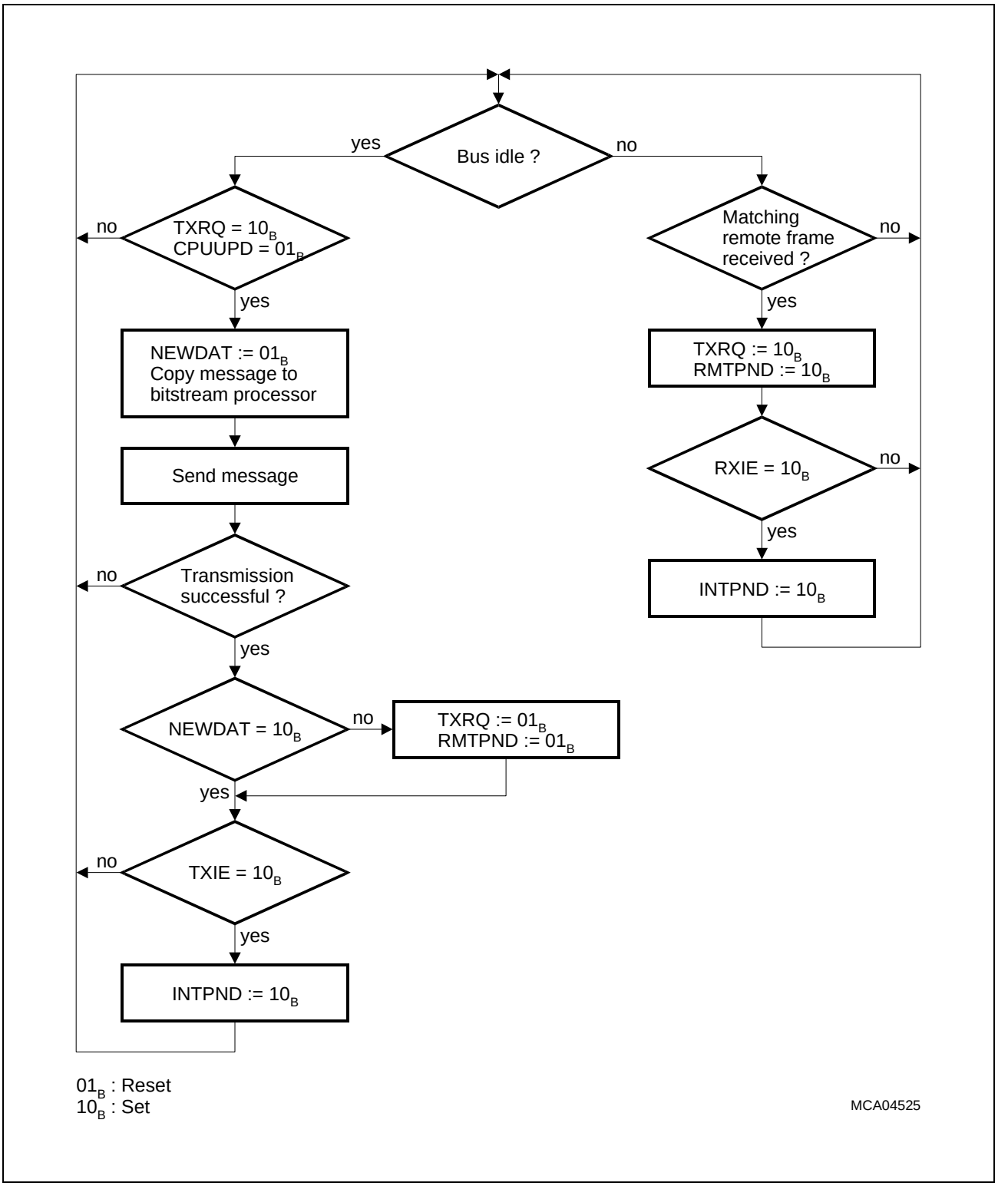


Figure 4-11 Handling of Message Objects with Direction = 1: Transmit by the CAN Controller Node Hardware

4.1.4.4 Handling of Receive Message Objects

A message object with direction flag DIR = 0 (message configuration register MSGCFGn) is handled as receive object.

In the initialization phase, the transmit request bit field (TXRQ), the message lost bit field (MSGLST) and the NEWDAT bit field in register MSGCTR should be reset.

All message objects with bit field MSGVAL = 10_B are operable and taken into account by the TwinCAN node controller operation described below.

When a data frame has been received, the new information is stored in the data partition of the message object (MSGDRn0, MSGDRn4) and the bit field DLC in register MSGCFG is updated with the number of received bytes. Unused message bytes will be overwritten by non-specified values. If the NEWDAT bit field in register MSGCTR is still set, the CAN controller assumes an overwrite of the previously stored message and signals a data loss by setting bit field MSGLST. In any case, bit field NEWDAT is automatically set to 10_B reporting an update of the data register by the CAN controller. The captured value of the frame counter is copied to bit field CFCVAL in register MSGCTRn and a receive interrupt request is generated (INTPNDn and RXIPNDn are set) if enabled by RXIE = 10_B. Then the Frame Counter is incremented by one if enabled in control register AFCR/BFCR.

When a receive object is marked to be transmitted (TXRQ = 10_B), bit MSGLST changes automatically to CPUUPD. If CPUUPD is reset to 01_B, the CAN controller generates a remote frame which is emitted to the other communication partners via CAN bus. In case of CPUUPD = 10_B, the remote frame transfer is prohibited until the CPU releases the pending transmission by resetting CPUUPD to 01_B. RMTPND and TXRQ are automatically reset when the remote frame has been successfully transmitted. Finally, a transmit interrupt request is generated if enabled by TXIE = 10_B.

When a remote frame with matching identifier is received, it is not answered and not indicated by an interrupt request.

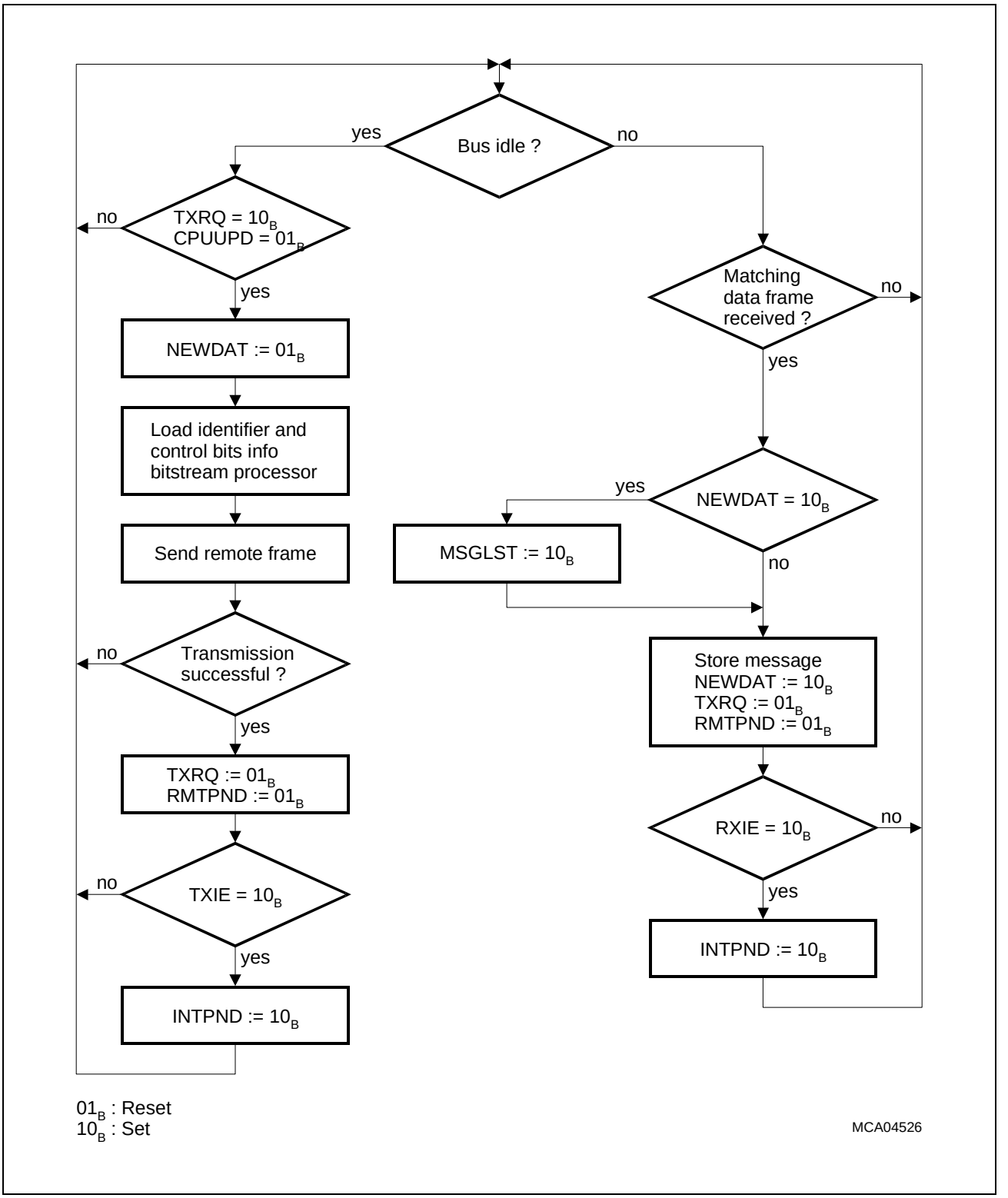


Figure 4-12 Handling of Message Objects with Direction = 0: Receive by the CAN Controller Node Hardware

4.1.4.5 Single Data Transfer Mode

Single Data Transfer Mode is a useful feature to broadcast data over the CAN bus without unintentional duplication of information. Single Data Transfer Mode is selected via bit SDT in the FIFO/Gateway control register MSGFGCRn.

Each received data frame with matching identifier is automatically stored in the corresponding receive message object if MSGVAL is set to 10_B. When data frames addressing the same message object are received within a short time interval, information might get lost (indicated by MSGLST = 10_B), if the CPU has not processed the former message object contents in time.
Each arriving remote frame with matching identifier is answered by a data frame based on the contents of the corresponding message object. This behavior may lead to multiple generation and transmission of identical data frames according to the number of accepted remote requests.

If SDT is set to 1, the TwinCAN node controller automatically resets bit MSGVAL in the addressed message object, when the transmission of the corresponding data frame has been finished successfully. Consequently, all following remote requests concerning the disabled message object are ignored until MSGVAL is set again by the CPU. This feature allows for transmitting of data in a consecutive manner without unintended doubling of any information.

If SDT is cleared, control bit field MSGVAL is not reset by the TwinCAN node controller.

4.1.5 CAN Message Object Buffer (FIFO)

With a high CPU load, it may be difficult to process an incoming data frame before the corresponding message object is overwritten with the next input data stream provided by the TwinCAN node controller. Depending on the application, it could be also necessary to ensure a minimum data frame generation rate to fulfill external real time requirements.

Therefore, a message buffer facility has been implemented to avoid a loss of incoming messages and to minimize the setup time for outgoing messages. Some message objects can be configured as ‘Base Object’ using successive ‘Slave Message Objects’ as individual buffer storage (circular buffer used as message FIFO). The number of base and slave message objects, combined to a buffer, must be a power of two (2, 4, 8 etc.) and the ‘Buffer Base Address’ must be an integer multiple of the buffer length. For example, a buffer containing 8 messages can use object 0, 8, 16 or 24 as the ‘Base Object’ as illustrated in **Table 4-2**.

Table 4-2 Message Objects Providing FIFO Base Functionality

Message Object n FIFO Size	0	2	4	6	8	10	12	14	16	18	…	30
2-stage FIFO	X	X	X	X	X	X	X	X	X	X		X
4-stage FIFO	X		X		X		X		X			
8-stage FIFO	X				X				X			
16-stage FIFO	X								X			
32-stage FIFO	X											

A ‘Base Object’ is defined by setting bit field MMC to 010_B (control register MSGFGCRn); the requested buffer size is determined by selecting an appropriate value for FSIZE. A ‘Slave Object’ is defined by setting bit field MMC to 011_B. Bit field FSIZE must be equal in all FIFO elements.

The identifiers and corresponding acceptance masks must be identical in all FIFO elements belonging to the same buffer in case of a receive FIFO (DIR = 0). For a transmit FIFO (DIR = 1) the identifier of the currently addressed message object is taken into account for transmission.

Each member of a buffer configuration keeps its individual MSGVAL, NEWDAT, CPUUPD or MSGLST, TXRQ and RMTPND flag and its separate interrupt control configuration. Inside a FIFO buffer, all elements must be:

- Assigned to the same CAN node (control bit NODE in register MSGCFGn),
- Programmed for the same transfer direction (control bit DIR),
- Set up to the same identifier length (control bit XTD),
- Programmed to the same FIFO length (bit field FSIZEn) and
- Set up with the same value for the FIFO direction (bit FD in register MSGFGCRn).
- The slave’s CANPTR must point to the FIFO base object.

The CANPTR of the base object must be initialized with the message number of the base object, the CANPTR pointers of the slave objects must be set up with the message number of the base object. The CANPTR of the base object addresses the next FIFO element to be accessed for information transfer and its value is calculated as follows:

CANPTRn(new) := CANPTRn(old) & ~FSIZEn | (CANPTRn(old) + 1) & FSIZEn

Control bit FD defines which transfer action (reception or transmission) leads to an update of the CANPTR bit field. Bit FD works independently from the direction bit DIR of the FIFO elements. The reception of a data frame (DIR = 0) or the reception of a remote frame (DIR = 1) are receive actions leading to an update of CANPTR if FD = 0. The transmission of a data frame (DIR = 1) or the transmission of a remote frame (DIR = 0) are transmit actions initiating an increment of CANPTR if FD = 1.

Note: The overall message object storage size is not affected by the configuration of buffer structures. The available storage size may be used for 32 message objects without buffering or for one message object with a buffer depth of 32 elements. Additionally, any combination of buffered and unbuffered message objects is allowed, according to the FIFO rules, as long as the limit of 32 message objects is not exceeded.

4.1.5.1 Buffer Access by the CAN Controller

Data transfer between the message buffer and the CAN bus is managed by the associated CAN controller. Each buffer is controlled by a FIFO algorithm (First In, First Out = First Overwritten) storing messages, delivered by the CAN controller, in a circular order.

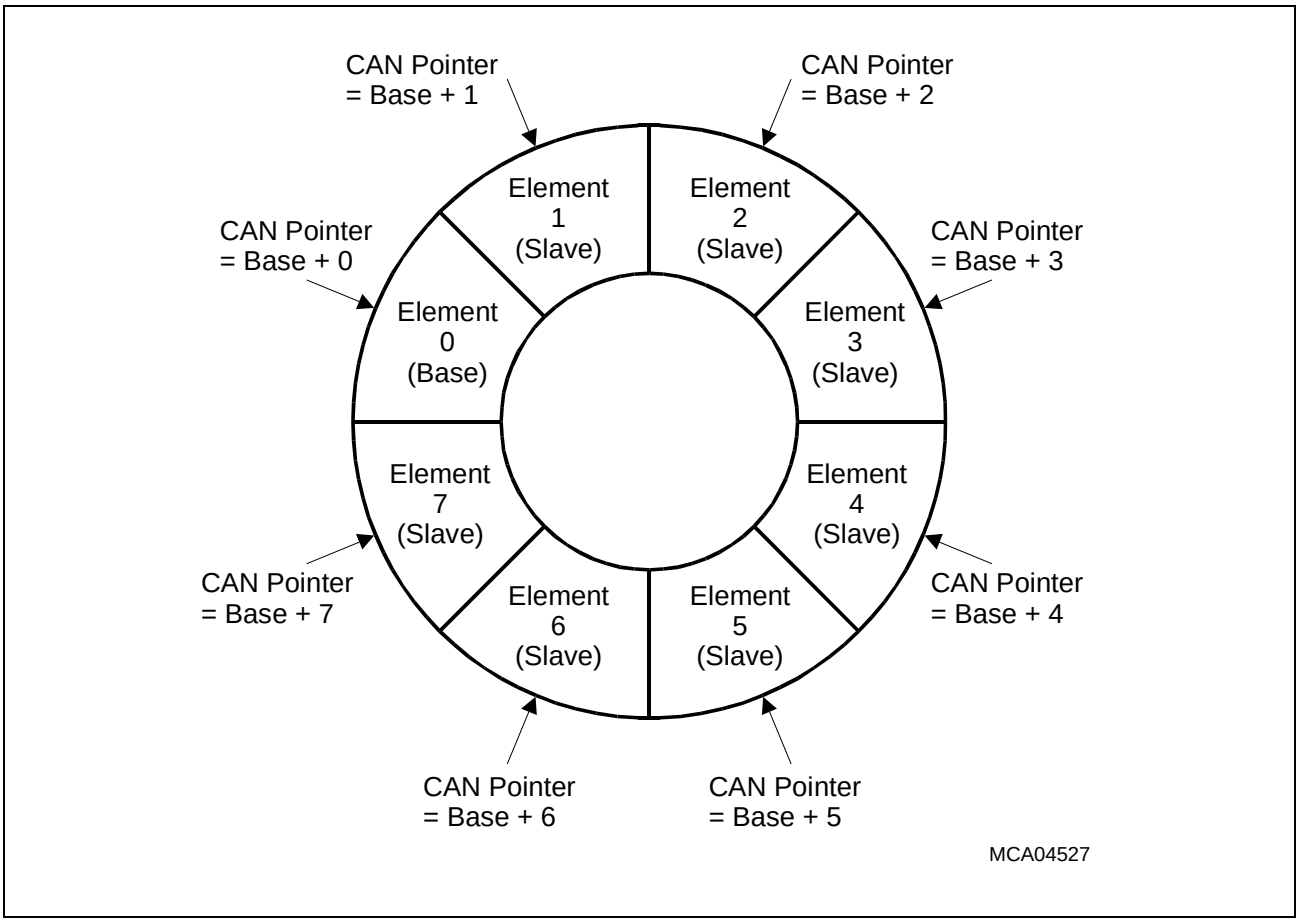


Figure 4-13 FIFO Buffer Structure: One Base Object and Seven Slave Objects

If the FIFO buffer was initialized with receive objects, the first accepted message is stored in the Base Message Object (number n), the second message is written to buffer element (n + 1) and so on. The number of the element, used to store the next input message, is indicated by bit field CANPTR in control register MSGFGCRn of the base object. If the reserved buffer space has been used up, the Base Message Object (followed by the consecutive Slave Objects) is addressed again to store the next incoming message. When a message object was not read out on time by the CPU, the previous message data is overwritten, as indicated by flag MSGLST in the corresponding MSGCTR register.

If the FIFO buffer was initialized with transmit message objects, the CAN controller starts the transfer with the contents of buffer element 0 (FIFO base object) and increments bit field CANPTR in control register MSGFGCRn, pointing to the next element to be transmitted.

If the message object currently addressed by the base object's CANPTR is not valid (MSGVAL = 01_B), the FIFO is not enabled for data transfer. In this case, the MSGVAL bit fields of the other FIFO elements (including the base element if not currently addressed) are not taken into account.

If MSGVAL bit fields are set to 10_B for the FIFO base object and 01_B for the currently addressed FIFO slave object, the data will not be delivered to the slave object, whereas the bit field CANPTR in the FIFO base object is incremented according to FIFO rules.

If the FIFO is set up for the transmission of data frames and a matching remote frame is detected for one of the elements of the FIFO, the transmit request and remote pending bits will be set automatically in the corresponding message object. The transmission of the requested data frame is handled according to the FIFO rules and the value of the CANPTR bit field in the FIFO base object.

4.1.5.2 Buffer Access by the CPU

The message transfer between a buffer and the CPU must be managed by software. All message objects combined to a buffer can be accessed directly by the CPU. Bit field CANPTR in control register MSGFGCRn is not automatically modified by a CPU access to the message object registers.

4.1.6 Gateway Message Handling

The TwinCAN module supports an automatic information transfer between two independent CAN bus systems without CPU interaction.

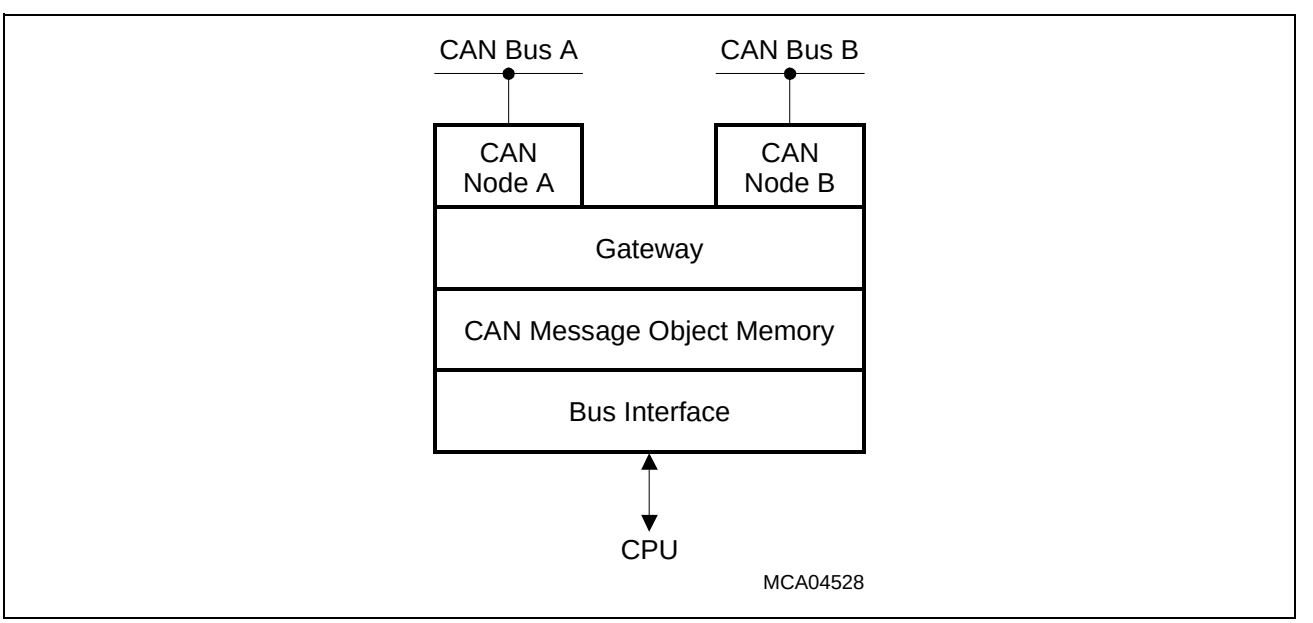


Figure 4-14 TwinCAN Gateway Functionality

The gateway functionality is handled via the CAN message object memory shared by both CAN nodes. Each object stored in the message memory is associated to Node A or to Node B via bit NODE in the message configuration register MSGCFGn. The information exchange between both CAN nodes can be handled by coupling two message objects (Normal Gateway Mode) or by sharing one common message object (Shared Gateway Mode).

In the following sections, the gateway side receiving data frames is named “Source” (indicated by <s>) and the side transmitting data frames that passed the gateway is called “Destination” (indicated by <d>). In accordance with this notation, remote frames passing the gateway are received on the destination side and transmitted on the source side.

The gateway function of a message object and the requested information transfer mode are defined by bit field MMC in the FIFO/Gateway control register MSGFGCRn.

4.1.6.1 Normal Gateway Mode

The ‘Normal Gateway Mode’ consumes two message objects to transfer a message from the source to the destination node. In this mode, different identifiers can be used for the same message data. Details of the message transfer through the ‘Normal Gateway’ are controlled by the respective $MSGFGCR_{<s>}$ and $MSGFGCR_{<d>}$ registers. All eight data bytes from the source object (even if not all bytes are valid) are copied to the destination object.

The object receiving the information from the source node must be configured as receive message object (DIR = 0) and must be associated with the source CAN bus via bit NODE. Register $MSGFGCR_{<s>}$ should be initialized according the following enumeration:

- Bit field $MMC_{<s>}$ must be set to 100_B indicating a ‘Normal Mode Gateway’ for incoming (data) frames.
- Bit field $CANPTR_{<s>}$ must be initialized with the number of the message object used as destination for the data copy process.
- If no FIFO functionality is required on the destination side, bit field $FSIZE_{<s>}$ must be filled with 00000_B. When FIFO capabilities are needed, bit field $FSIZE_{<s>}$ must contain the FIFO buffer length, which must be identical with the content of the FIFO base object’s FSIZE bit field on the destination side.
- When bit $IDC_{<s>}$ is set, the identifier of the source message object is copied to the destination message object. Otherwise, the identifier of the destination message object is not modified.
- If $DLCC_{<s>}$ is set, the ‘Data Length Code’ of the source message is copied to the destination object.
- Bit $GDFS_{<s>}$ decides, whether the transmit request flag on the destination side is set ($TXRQ_{<d>} = 10_B$ if $GDFS_{<s>} = 1$) after finishing the data copy process. An automatic transmission of the copied data frame on the destination side takes place, if control bit $CPUUPD_{<d>}$ is reset to 01_B.

The destination message object, addressed by $CANPTR_{<s>}$, must be configured for transmit operation (DIR = 1). Depending on the required functionality, the destination message object can be set up in three different operating modes:

- With $MMC_{<d>} = 000_B$, the destination message object is declared as standard message object. In this case, data frames, received on the source side, can be

automatically emitted on the destination side if enabled by the respective control bits $CPUUPD_{<d>}$ and $GDFS_{<s>}$. Remote frames, received on the destination side, are not transferred to the source side, but can be directly answered by the destination message object if $CPUUPD_{<d>}$ is reset to 01_B.

- With $MMC_{<d>} = 100_B$, the destination message object is declared as ‘Normal Mode Gateway’ for incoming (remote) frames. Data frames, received on the source side, can be automatically emitted on the destination side if enabled ($CPUUPD_{<d>}$, $GDFS_{<s>}$) and remote frames, received on the destination side, are transmitted on the source side if enabled by $SRREN_{<d>} = 1$.
- With $MMC_{<d>} = 01X_B$, the destination message object is set up as an element of a FIFO buffering the data frames transferred from the source side through the gateway. Remote frames received on the destination side are not transferred to the source side, but can be directly answered by the currently addressed FIFO element if $CPUUPD_{<d>}$ is reset (bits $SRREN_{<d>}$ must be cleared).
- Remote frame handling is completely done on the destination side according to FIFO rules.

$MMC_{<d>} = 000_B$:

Operation with a standard message object on the destination side is illustrated in **Figure 4-15**.

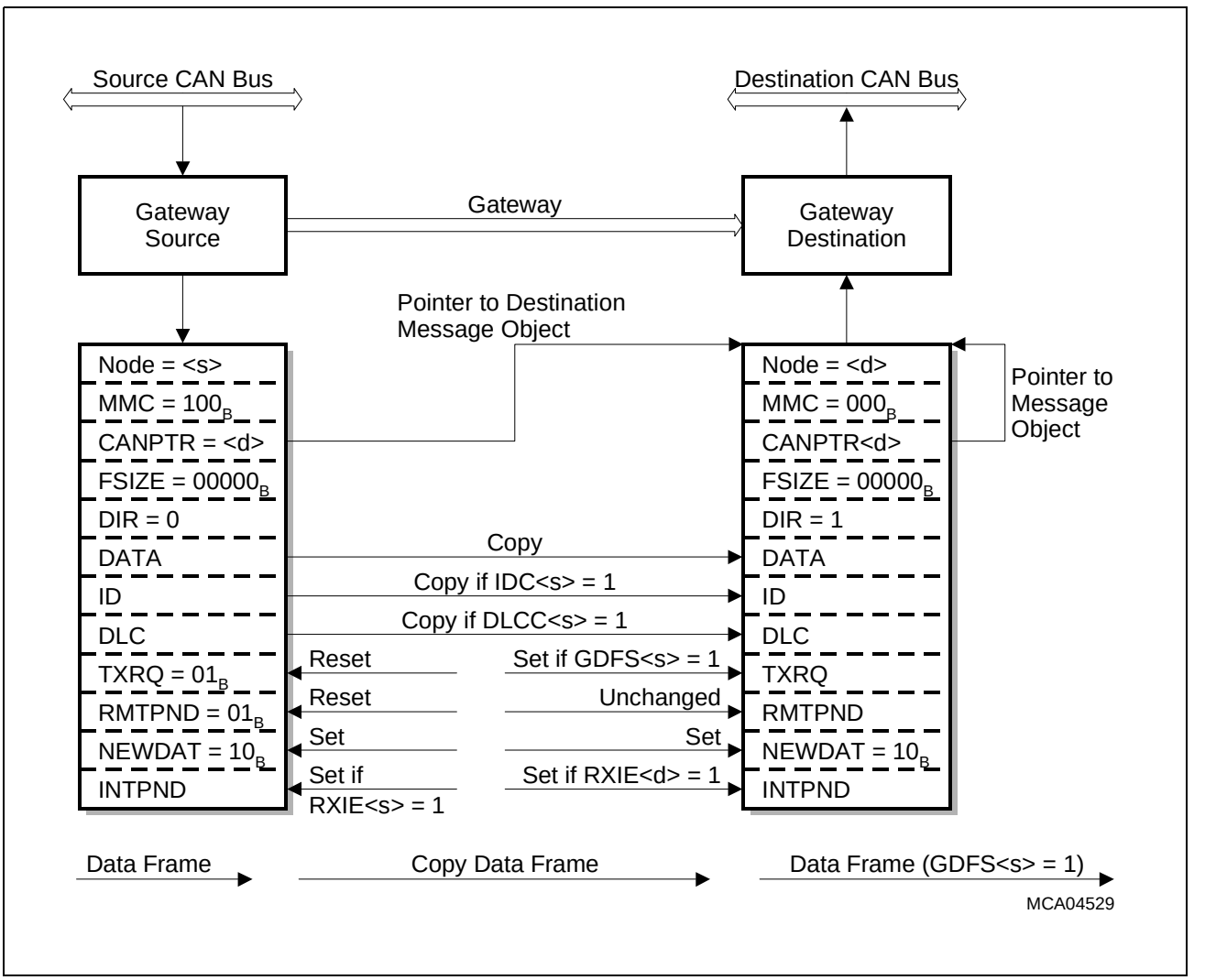


Figure 4-15 Data Frame Reception in Normal Gateway Mode with a Standard Destination Message Object ($MMC_{<d>} = 000_B$)

A matching data frame, arrived at the source node, is automatically copied to the destination node's message object addressed by $CANPTR_{<s>}$. Bit field $CANPTR_{<d>}$ is loaded with the destination message object number. Regardless of control bit $SRREN_{<d>}$, remote frames received on the destination node are not transferred to the source side, but can be directly answered by the destination message object. For this purpose, control bit fields $TXRQ_{<d>}$ and $RMTPND_{<d>}$ are set to 10_B, which immediately initiates a data frame transmission on the destination CAN bus if $CPUUPD_{<d>}$ is reset to 01_B.

$MMC_{<d>} = 100_B$:

The operation with a ‘Normal Mode Gateway’ message object for incoming (remote) frames on the destination side is illustrated in **Figure 4-16**.

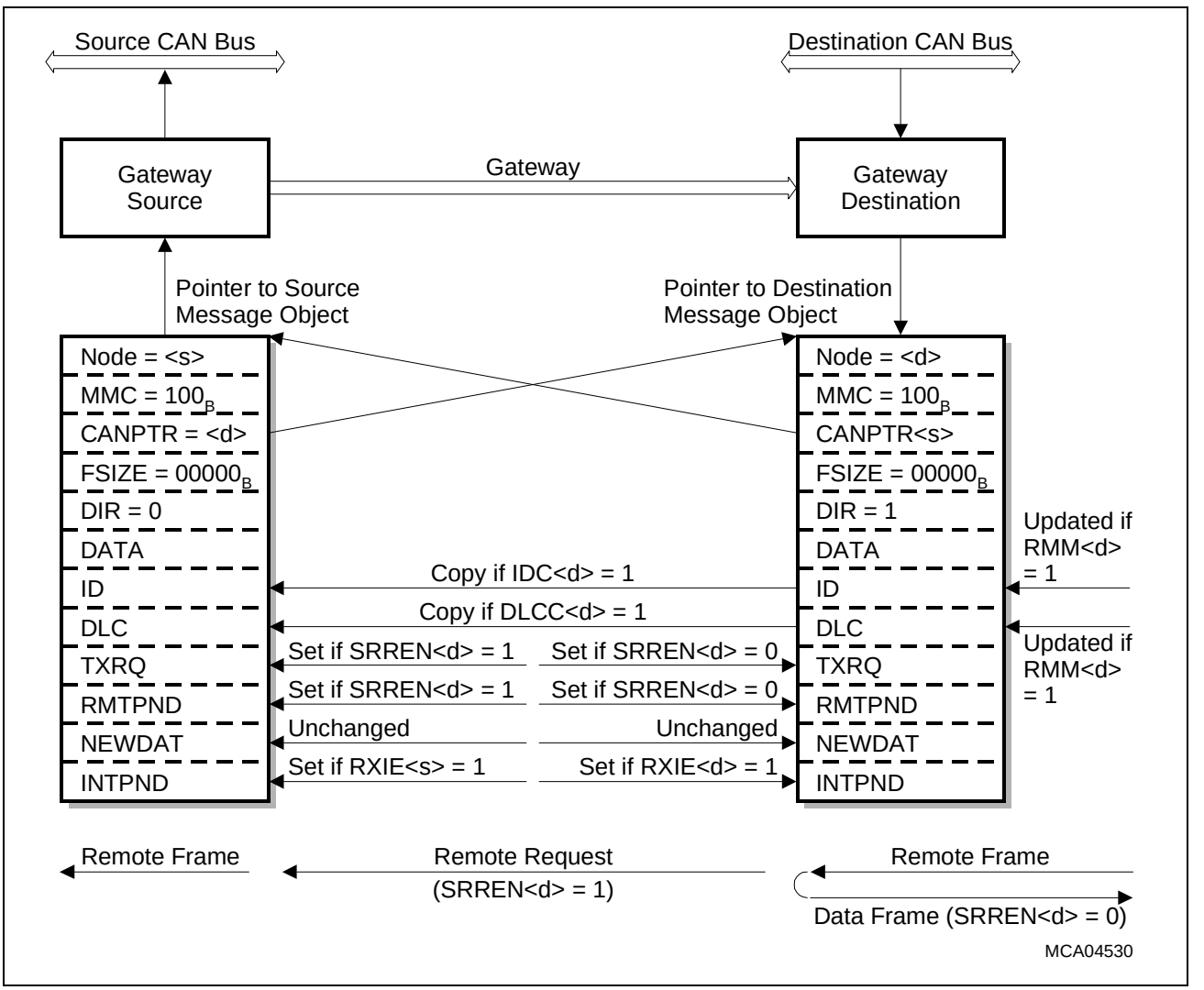


Figure 4-16 Remote Frame Transfer in Normal Gateway Mode, $MMC_{<d>} = 100_B$

The gateway object on the destination side, setup as transmit object, can receive remote frames. If bit $SRREN_{<d>}$ in the associated gateway control register MSGFGCRn is cleared, a remote frame with matching identifier is directly answered by the CAN destination node controller. For this purpose, control bits $TXRQ_{<d>}$ and $RMTPND_{<d>}$ are set to 10_B, which immediately initiates a data frame transmission on the destination CAN bus if $CPUUPD_{<d>}$ is reset. When bit $SRREN_{<d>}$ is set to 1, a remote frame received on the destination side is transferred via the gateway and transmitted again by the CAN source node controller.

A transmit request for the gateway message object on the source side, initiated by the CPU via setting $TXRQ_{<s>}$, always generates a remote frame on the source CAN bus system.

4.1.6.2 Normal Gateway with FIFO Buffering

$MMC_{<d>}$= $01X_B$:

When the gateway destination object is programmed as FIFO buffer, bit field $CANPTR_{<s>}$ is used as the pointer to the FIFO element to be addressed as destination for the next copy process. $CANPTR_{<s>}$ must be initialized with the message object number of the FIFO base element on the destination side. $CANPTR_{<s>}$ is automatically updated according to the FIFO rules when a data frame was copied to the indicated FIFO element on the destination side. Bit $GDFS_{<s>}$ determines if the $TXRQ_{<d>}$ bit in the selected FIFO element is set after reception of a data frame copied from the source side.

The base message object is indicated by <ba>, the slave message objects by <sl>. The number of base and slave message objects, combined to a buffer on the destination side, must be a power of two (2, 4, 8 etc.) and the ‘Buffer Base Address’ must be an integer multiple of the buffer length. Bit field $CANPTR_{<ba>}$ of the FIFO base element and bit field $CANPTR_{<s>}$ must be initialized with the same start value (message object number of the FIFO base element). $CANPTR_{<sl>}$ of all FIFO slave elements must be initialized with the message object number of the FIFO base element. Bit field $FSIZE_{<d>}$ of all FIFO elements must contain the FIFO buffer length and must be identical with the content of $FSIZE_{<s>}$.

Figure 4-17 illustrates the operation of a ‘Normal Gateway’ with a FIFO buffer on the destination side.

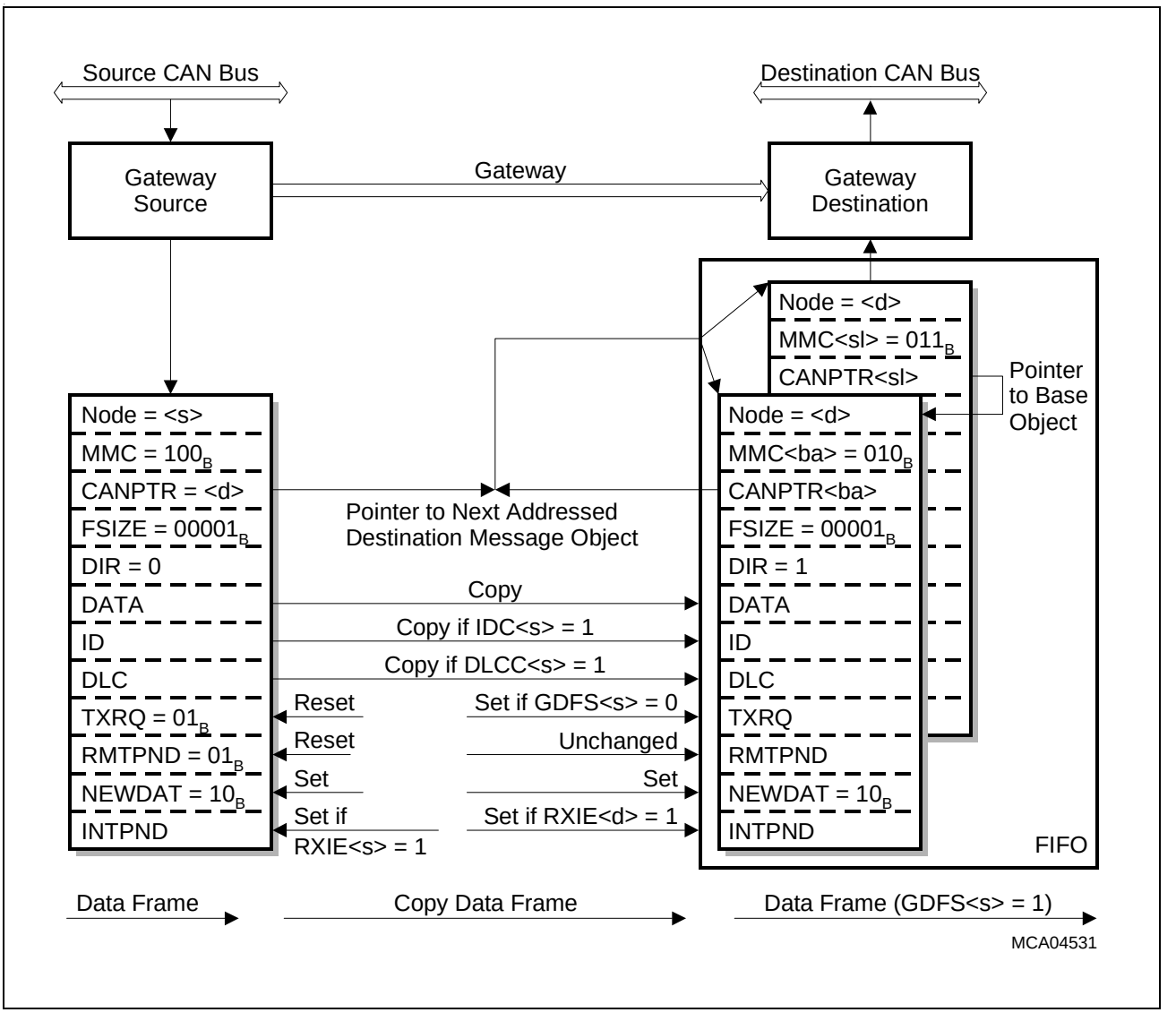


Figure 4-17 Data Frame Transfer in Normal Gateway Mode with a 2 Stage FIFO on the Destination Side ($MMC_{<d>} = 01X_B$)

Remote frames, received on the destination side by a FIFO element, cannot be automatically passed to the source side. Therefore, the $SRREN_{<d>}$ control bits associated to the FIFO elements on the destination side, must be cleared so that incoming remote frames with matching identifiers can be answered directly with appropriate data frames.

Buffered transfers of remote requests from the destination to the source side can be handled by a software routine operating on the FIFO buffered gateway configuration for data frame transfers. The elements of the FIFO buffer on the destination side should be configured as transmit message objects with $CPUUPD_{<d>} = 10_B$. An arriving remote frame with matching identifier should initiate an interrupt service request for the addressed FIFO message object. The associated interrupt service routine may copy the message identifier and the data length code from the received remote frame to a receive

message object linked with the source side CAN node. In any case, TXRQ of the selected receive message object must be set to 10_B initiating the transmission of a remote frame on the source side.

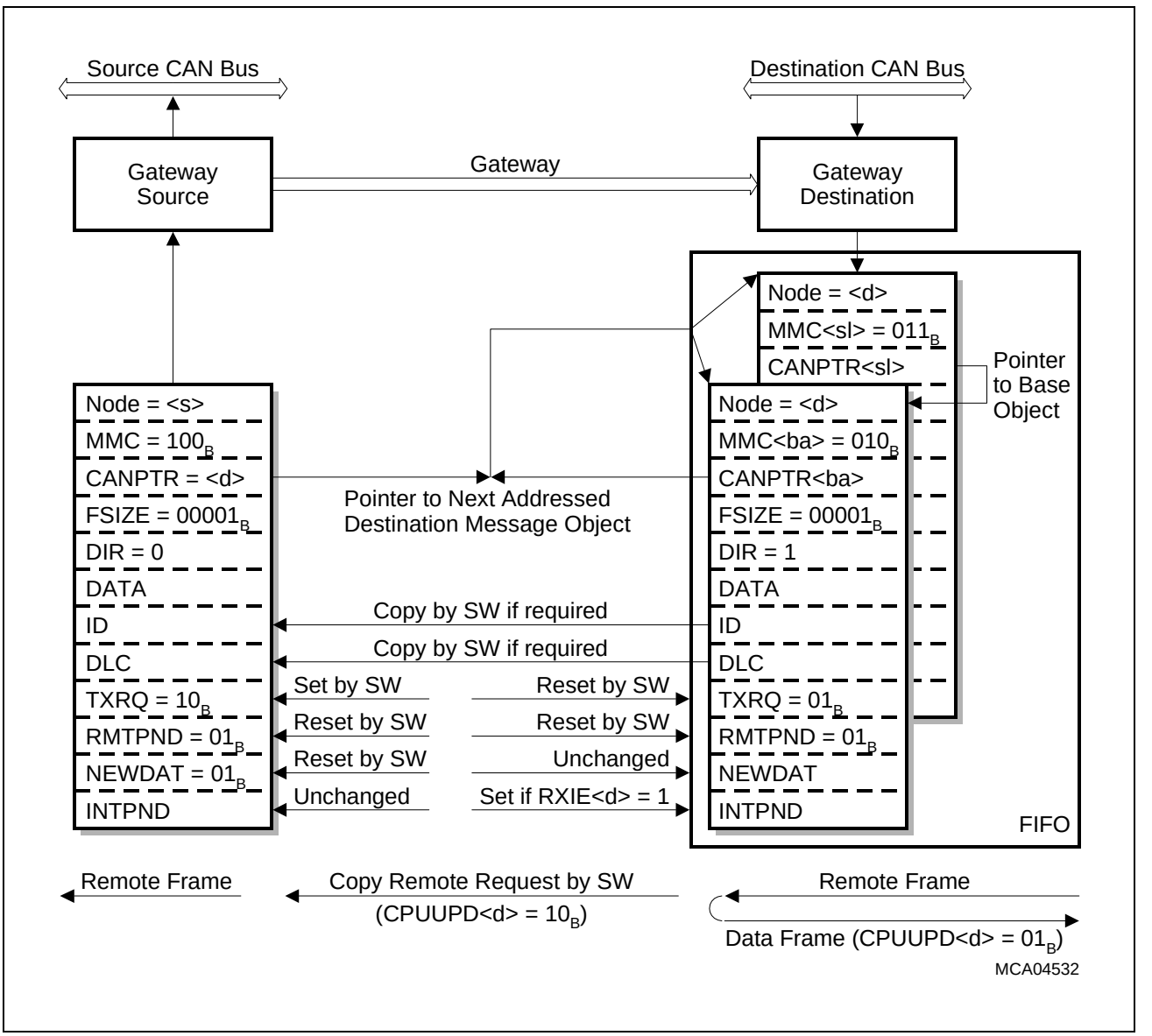


Figure 4-18 Remote Frame Transfer in Normal Gateway Mode with a Two-Stage FIFO on the Destination Side

4.1.6.3 Shared Gateway Mode

In Shared Gateway Mode, only one message object is required to implement a gateway function. The shared gateway object can be considered as normal message object, that is toggled between the source and destination CAN node as illustrated in **Figure 4-19**.

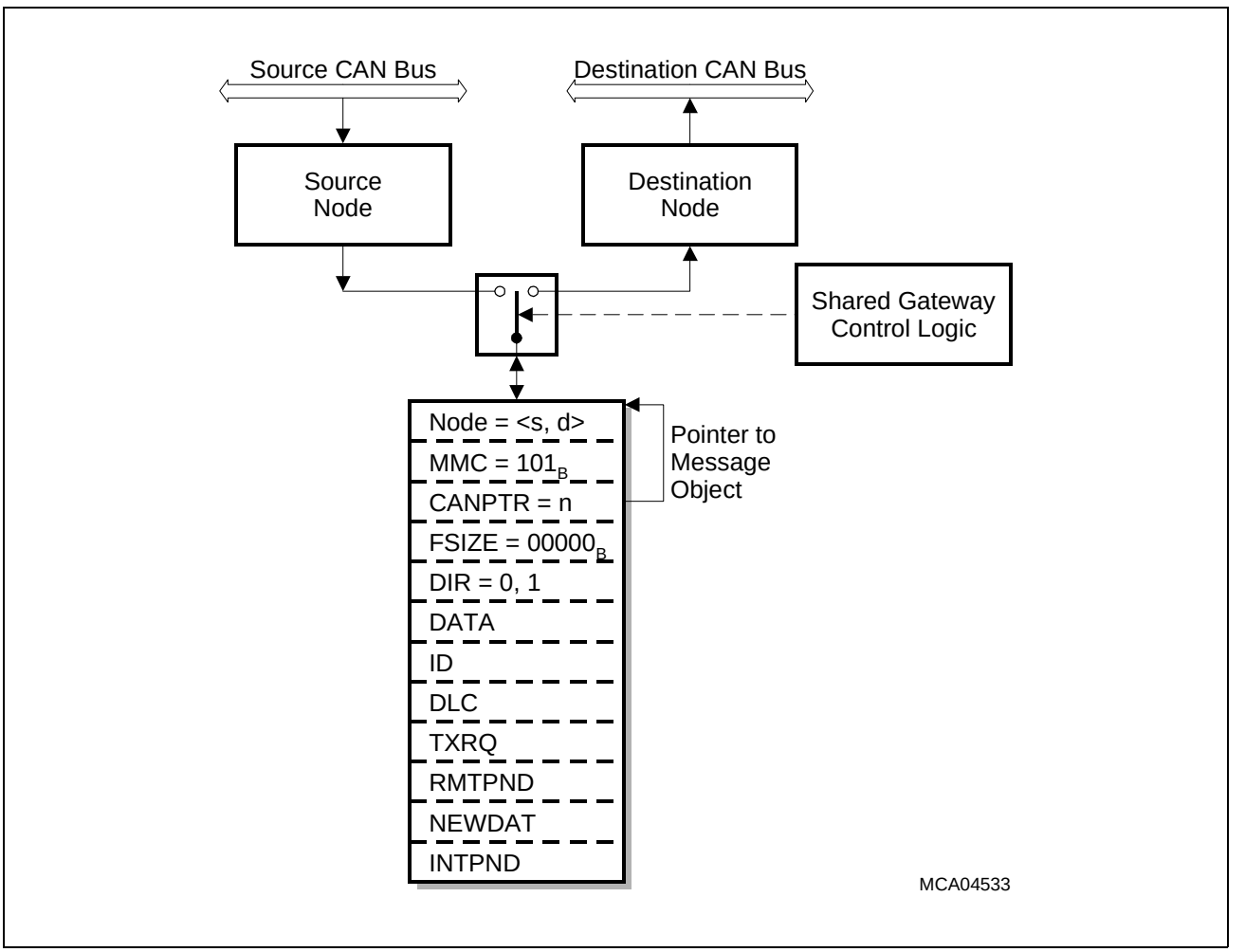


Figure 4-19 Principle of the Shared Gateway Mode

Each message object can be used as a shared gateway by setting MMC in the corresponding MSGFGCRn register to 101_B. When the message configuration bit NODE is cleared, CAN node A is used as source, transferring data frames to destination node B. If NODE is set to 1, CAN node B operates as data source. A bi-directional gateway is achieved by using a second message object, configured to shared gateway mode with a complementary NODE declaration. Bit field CANPTR must be initialized with the shared gateway's message object number, whereas FSIZE, IDC and DLCC must be cleared. Bit GDFS in control register MSGFGCRn determines whether bit TXRQ will be set automatically for any arriving data frame with matching identifier (GDFS = 1).

Bit SRREN determines, whether a remote frame, received on the destination side, is transferred through the gateway to the source node or is answered directly by a data frame generated on the destination side.

The functionality of the shared gateway mode is optimized to support different scenarios:

- A data source, connected with CAN node A continuously transmits data frames, which must be automatically emitted on the destination CAN bus by CAN node B.
  The corresponding transfer state transitions are 1 - 2 - …
- A data source connected with CAN node A continuously transmits data frames, which must be emitted by CAN node B upon a matching remote frame received from the destination CAN bus.
  The corresponding transfer state transitions are 7 - 4 - 2 - …
- A data source connected with CAN node A transmits a data frame upon a matching remote frame that has been triggered by a matching remote frame received by CAN node B. The respective data frame must be emitted again on the destination CAN bus by CAN node B.
  The corresponding transfer state transitions are 5 - 6 - 1 - 3 - …

Depending on the application, the shared gateway message object can be initialized as receive object on the source side or as transmit object on the destination side via an appropriate configuration of NODE, DIR, GDFS, and SRREN. The various transfer states are illustrated in **Figure 4-20**.

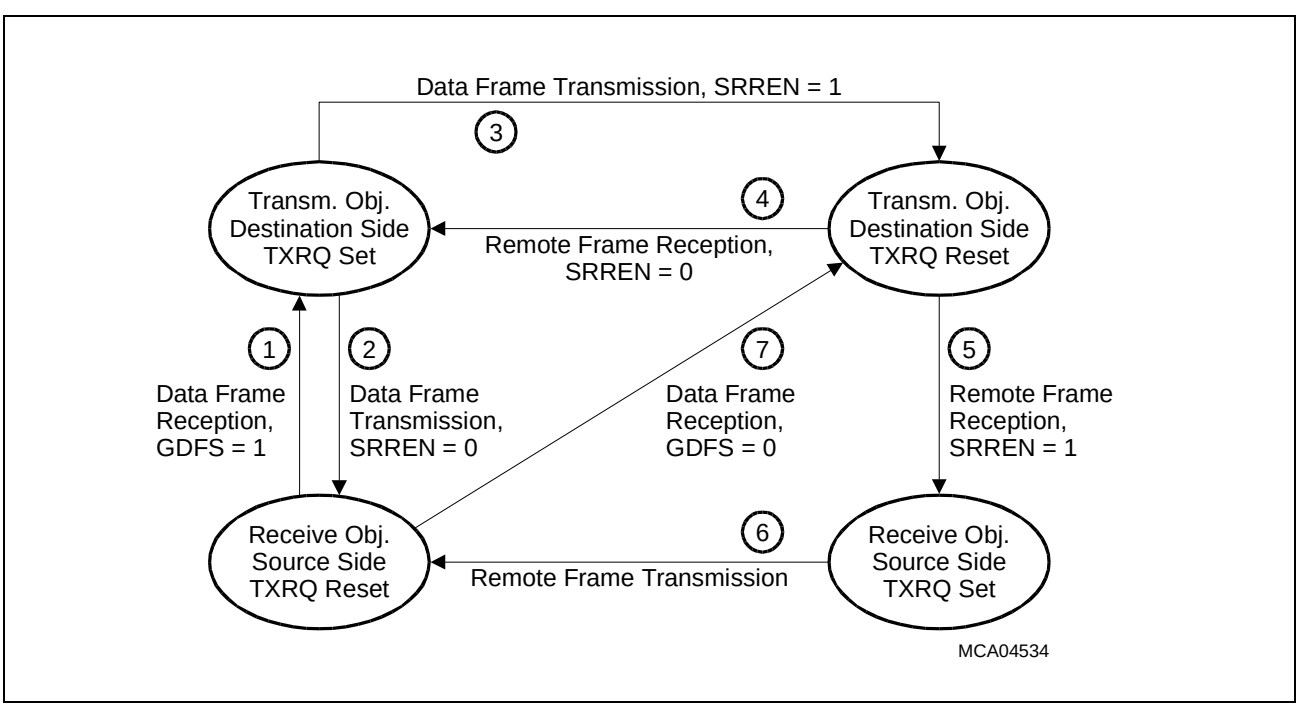


Figure 4-20 Transfer States in Shared Gateway Mode

When a shared gateway message object, set up as receive object on the source side (lower left state bubble in **Figure 4-20**), receives a data frame while GDFS is set to 1, it commutes to a transmission object on the destination side by toggling control bits NODE

and DIR and sends the corresponding data frame without any CPU interaction (upper left state bubble).
Depending on control bit SRREN, the shared gateway message object returns to its initial function as receive object assigned to the source side (SRREN = 0: state transition 2 to the lower left state bubble in **Figure 4-20**) or remains assigned to the destination side waiting for a remote frame with matching identifier (SRREN = 1: state transition 3 to the upper right state bubble).
When the shared gateway message object is assigned as transmit object to the destination side (upper right state bubble), it responds to remote frames received on the destination side. If bit SRREN is cleared, the remote request is answered directly by a data frame based on the contents of the gateway message object (state transition 4 to the upper left state bubble).
If bit SRREN is set and a remote frame is received on the destination side, the shared gateway message object commutes to a receive object on the source side by toggling control bits NODE and DIR and prepares the emission of the received remote frame by setting TXRQ and RMTPND to 10_B (state transition 5 to the lower right state bubble).
Then, the shared gateway message object emits the corresponding remote frame without any CPU interaction (state transition 6 to the lower left state bubble).
The gateway message object remains assigned to the source side until a data frame with matching identifier arrives (lower left state bubble). Then, the shared gateway message object returns to the destination side and, depending on control bit GDFS, transmits immediately the corresponding data frame (GDFS = 1, upper left state bubble) or waits upon an action of the CPU setting TXRQ to 10_B (GDFS = 0: state transition 7 to the upper right state bubble). Alternatively, a remote frame with matching identifier arriving on the destination side may set TXRQ to 10_B and initiate the data frame transmission.

If a data frame arrives on the source side while the shared gateway object with matching identifier is switched to the destination side, the data frame on the source side gets lost. Due to its temporary assignment to the destination node, the shared gateway message object does not notice the data frame on the source node and is not able to report the data loss via control bit field MSGLST = 10_B. The probability for a data loss is enlarged, if the automatic data frame transmission on the destination side is disabled by GDFS = 0. A corresponding behavior must be taken into account for incoming remote frames on the destination bus.

Note: As long as bit field MSGLST is activated, an incoming data frame cannot be transmitted automatically on the destination side. Due to the internal toggling of control bit DIR, the shared gateway object converts from receive to transmit operation and bit field MSGLST is interpreted as CPUUPD = 10_B preventing the automatic transmission of a data frame.

Table 4-3 and Table 4-4 show the impact of the transfer state transitions on the bit fields in the message object in Shared Gateway Mode:

Table 4-3 Shared Gateway State Transitions (Part 1 of 2)

Bit Fields	Transition 1: data frame received, GDFS = 1	Transition 2: data frame transmitted, SRREN = 0	Transition 3: data frame transmitted, SRREN = 1	Transition 4: remote frame received, SRREN = 0
Node	toggled to <d>	toggled to <s>	unchanged	unchanged
DIR	set	reset	unchanged	unchanged
DATA	received	unchanged	unchanged	unchanged
Identifier	received	unchanged	unchanged	received if RMM = 1
DLC	received	unchanged	unchanged	received if RMM = 1
TXRQ	set	reset	reset	set
RMTPND	reset	reset	reset	set
NEWDAT	set	reset	reset	reset
INTPND	set if RXIE = 10_B	set if TXIE = 10_B	set if TXIE = 10_B	set if RXIE = 10_B

Table 4-4 Shared Gateway State Transitions (Part 2 of 2)

Bit Fields	Transition 5: remote frame received, SRREN = 1	Transition 6: remote frame transmitted	Transition 7: data frame received, GDFS = 0
Node	toggled to <s>	unchanged	toggled to <d>
DIR	reset	unchanged	set
DATA	unchanged	unchanged	received
Identifier	received if RMM = 1	unchanged	received
DLC	received if RMM = 1	unchanged	received
TXRQ	set	reset	reset
RMTPND	reset	reset	reset
NEWDAT	unchanged	unchanged	set
INTPND	set if RXIE = 10_B	set if TXIE = 10_B	set if RXIE = 10_B

4.1.7 Programming the TwinCAN Module

Software initialization should be performed by setting bit INIT in the TwinCAN node specific control register ACR/BCR to 1. While bit INIT is set, all message transfers between the CAN controller and the CAN bus are disabled.

The initialization routine should process the following tasks:

- Configuration of the corresponding node,
- Initialization of each associated message object.

4.1.7.1 Configuration of CAN Node A/B

Each CAN node may be individually configured by programming the associated register. Depending on the contents of the ACR/BCR control registers, the “Normal Operation Mode” or the “CAN Analyzer Mode” is activated. Furthermore, various interrupt categories (status change, error, last error) can be enabled or disabled.
The bit timing is defined by programming the ABTR/BBTR register. The prescaler value, the synchronization jump width, and the time segments (arranged before and after the sample point) depend on the characteristic of the CAN bus segment linked to the corresponding CAN node.
The global interrupt node pointer register (AGINP/BGINP) controls the multiplexer connecting an interrupt request source (error, last error, global transmit/receive and frame counter overflow interrupt request) with one of the eight common interrupt nodes. The contents of the INTID mask register (AIMR0/4 and BIMR0/4) determine which interrupt sources may be reported by the AIR/BIR interrupt pending register.

4.1.7.2 Initialization of Message Objects

The message memory space, containing 32 message objects, is shared by both CAN nodes. Each message object must be configured concerning its target node and operation properties. Initialization of the message object properties is always started with disabling the message object via MSGVAL = 01_B.
The CAN node, associated with a message, is defined by bit NODE in register MSGCFGn. The message object can be also defined as gateway, transferring information from CAN node A to B or vice versa. In this case, the FIFO/Gateway control register MSGFGCRn must be programmed to specify the gateway mode (bit field MMC), the target interrupt node, and further details of the information handover.
The identifier, correlated with a message, is set up in register MSGARn. Bit XTD in register MSGCFGn indicates, whether an extended 29-bit or a standard 11-bit identifier is used and must be set accordingly. Incoming messages can be filtered by the mask defined in register MSGAMRn.
The message interrupt handling can be individually configured for transmit and receive direction. The direction specific interrupt is enabled by bits TXIE and RXIE in register MSGCNTn and the target interrupt node is selected by bit fields TXINP and RXINP in

register MSGCFGn.
Message objects can be provided with a FIFO buffer. The buffer size is determined by bit field FSIZE in the FIFO/Gateway control register MSGFGCRn.
For transmit message objects, the object property assignment can be completed by setting MSGVAL to 10_B, before the corresponding data partition has been initialized. If bit field CPUUPD is set to 10_B, an incoming remote frame with matching identifier is kept in mind via setting TXRQ internally, but is not immediately answered by a corresponding data frame. The message data stored in register MSGDRn0/MSGDRn4 can be updated as long as CPUUPD is hold on 10_B. As soon as CPUUPD is reset to 01_B, the respective data frame is transmitted by the associated TwinCAN node controller.

4.1.7.3 Controlling a Message Transfer

Figure 4-21 illustrates the handling of a transmit message object. Initialization of the message object properties is always started by disabling the message object via MSGVAL = 01_B. After resetting some control flags (INTPND, RMTPND, TXRQ and NEWDAT), the transfer direction and the identifier are defined. The message object initialization is completed by setting MSGVAL to 10_B.
An update of a transmit message data partition should be prepared by setting CPUUPD to 10_B followed by a write access to the MSGDRn0/MSGDRn4 register. The data partition update must be indicated by the CPU via setting NEWDAT to 10_B. Afterwards, bit CPUUPD must be reset to 01_B, if an automatic message handling is requested. In this case, the data transmission is started when flag TXRQ in register MSGCTRn has been set to 10_B by software or by the respective CAN node hardware due to a received remote frame with matching identifier. If CPUUPD remains set, the CPU must initiate the data transmission by setting TXRQ to 10_B and disabling CPUUPD. If a remote frame with an accepted identifier arrives during the update of a message object's data storage, bit TXRQ and RMTPND are automatically set to 10_B and the transmission of the corresponding data frame is automatically started by the CAN controller when CPUUPD is reset again.
Figure 4-22 demonstrates the handling of a receive message object. The initialization of the message object properties is embedded between disabling and enabling the message object via MSGVAL as described above. After setting MSGVAL to 10_B, the transmission of a remote frame can be initiated by the CPU via TXRQ = 10_B. The reception of a data frame is indicated by the associated TwinCAN node controller via NEWDAT = 10_B. The processing of the received data frame, stored in register MSGDRn0/MSGDRn4, should be started by the CPU with resetting NEWDAT to 01_B. After scanning flag MSGLST, indicating a loss of the previous message, the received information should be copied to an application data buffer in order to release the message object for a new data frame. Finally, NEWDAT should be checked again to ensure, that the processing was based on a consistent set of data and not on a part of an old message and part of the new message.

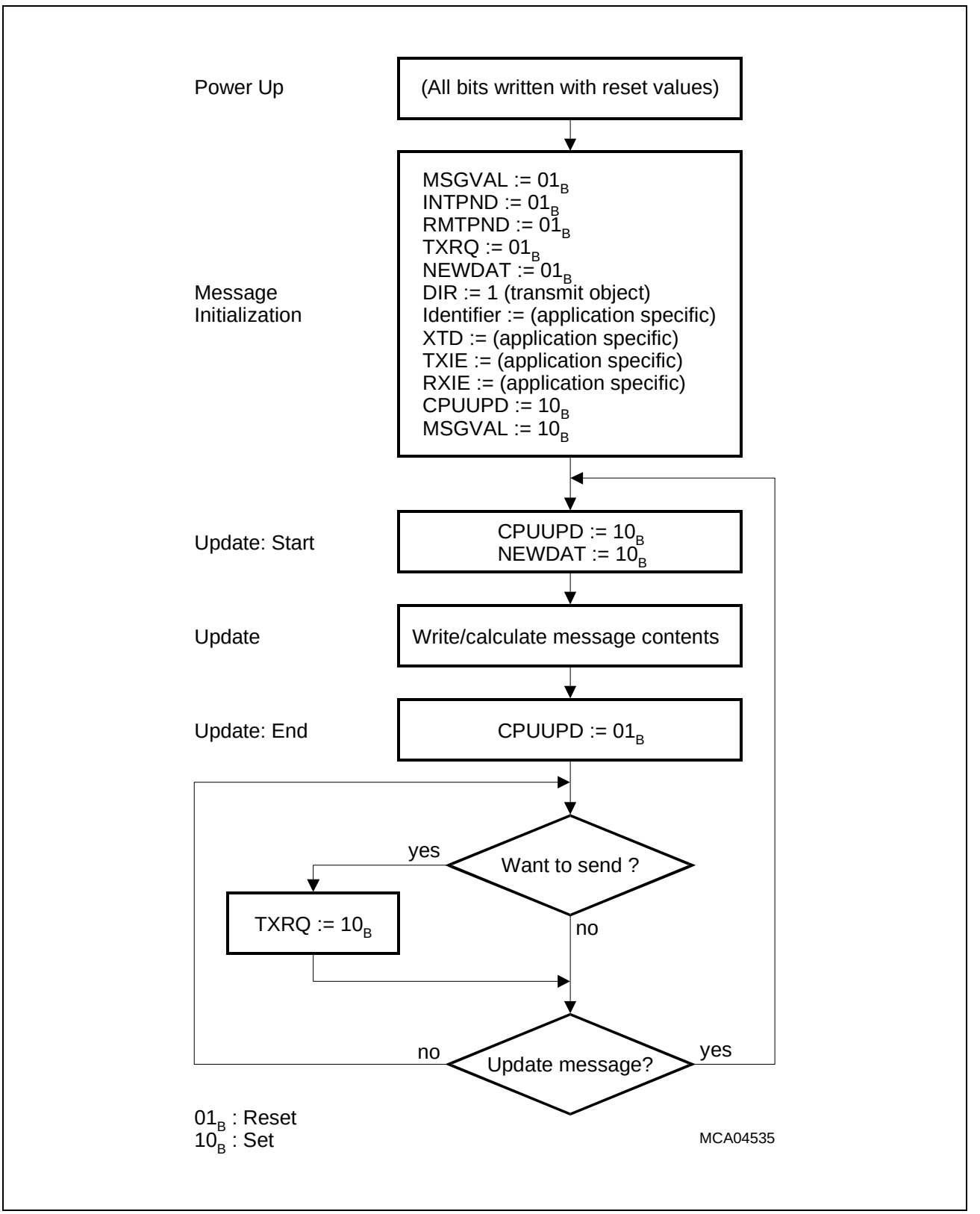


Figure 4-21 CPU Handling of Message Objects with Direction = Transmit

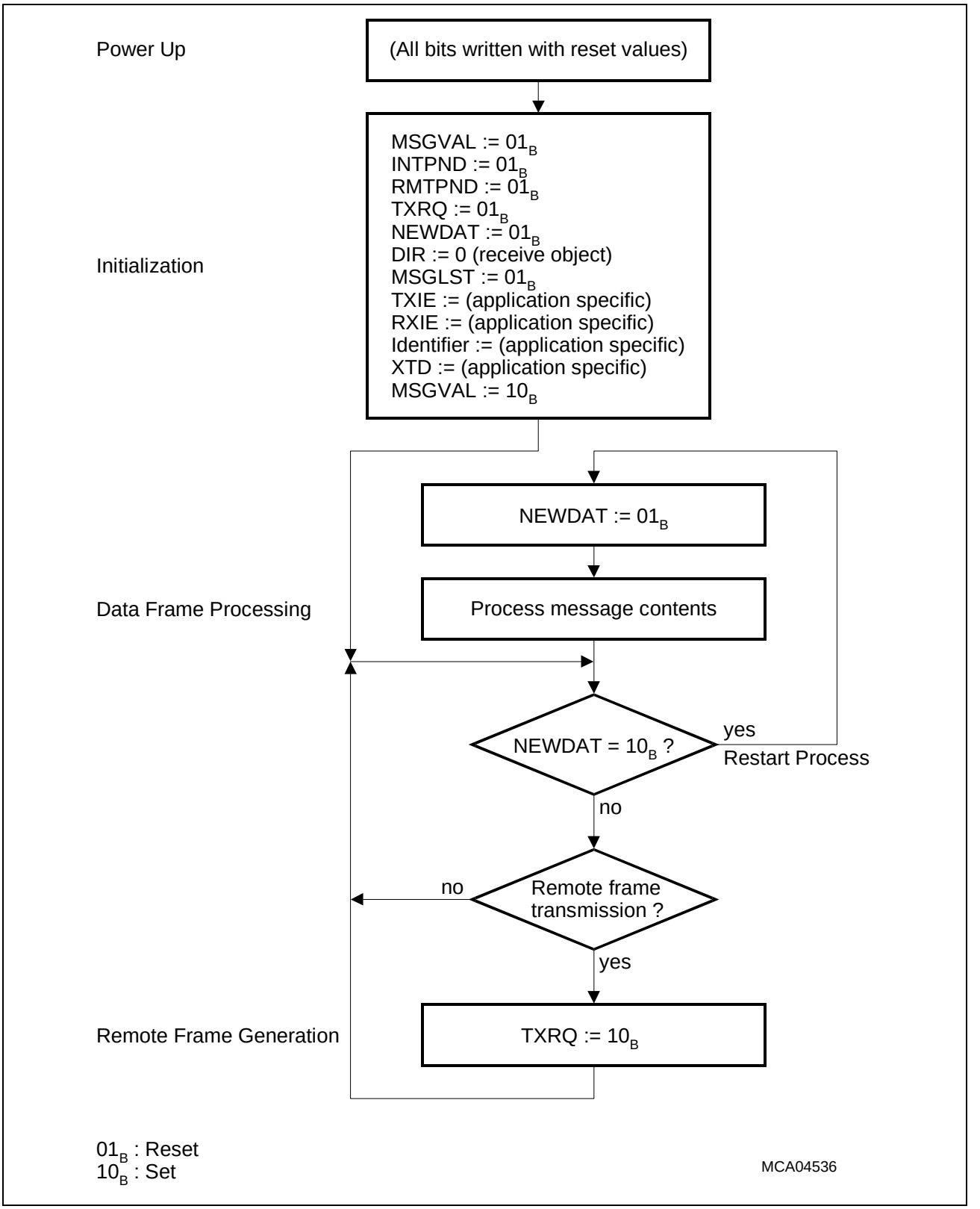


Figure 4-22 CPU Handling of Message Objects with Direction = Receive

4.1.8 Loop-Back Mode

The TwinCAN module's Loop-Back Mode provides the means to internally test the TwinCAN module and CAN driver software. CAN driver software can be developed and tested without being connected to a CAN bus system.

In Loop-Back Mode, the transmit pins deliver recessive signals to the transceiver. The transmit signals are combined together and are connected to the internal receive signals, as shown in **Figure 4-23**. The receive input pins are not taken into account in Loop-Back Mode.

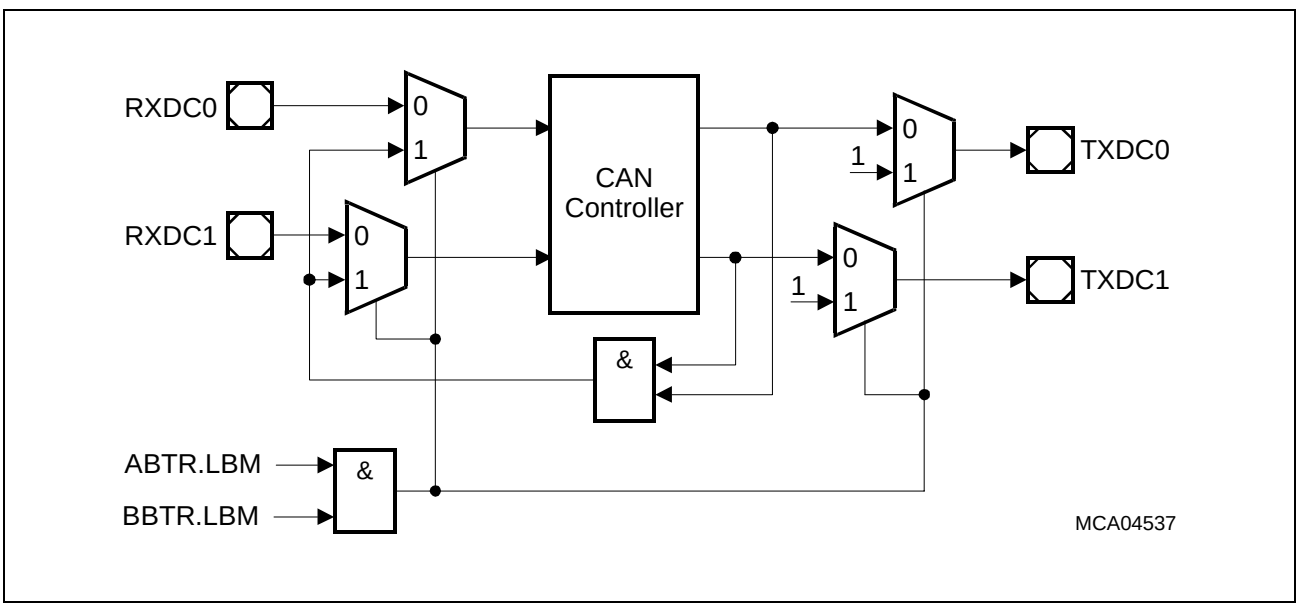


Figure 4-23 Loop-Back Mode

Loop-Back Mode is controlled by bits LBM in the bit timing registers of Node A and Node B according to **Table 4-5**.

Table 4-5 Loop-Back Mode

ABTR.LBM	BBTR.LBM	Description
0	0	Loop-Back Mode is disabled.
0	1	Loop-Back Mode is disabled.
1	0	Loop-Back Mode is disabled.
1	1	Loop-Back Mode is enabled.

4.1.9 Single Transmission Try Functionality

Single transmission try functionality is controlled individually for each message object by bit MSGFGCRn.STT. If the single transmission try functionality is enabled, the transmit request flag MSGCTRn.TXRQ is reset immediately after the transmission of a frame related to this message object has started. Thus, a transmit frame is only transferred once on the CAN bus, even if it has been corrupted by error frames.

Note: A message object must be tagged valid by bit MSGCTRn.MSGVAL in order to enable the transmission of the respective frame.

4.2 TwinCAN Registers

4.2.1 Register Map

Figure 4-24 shows all registers associated with the TwinCAN module kernel.

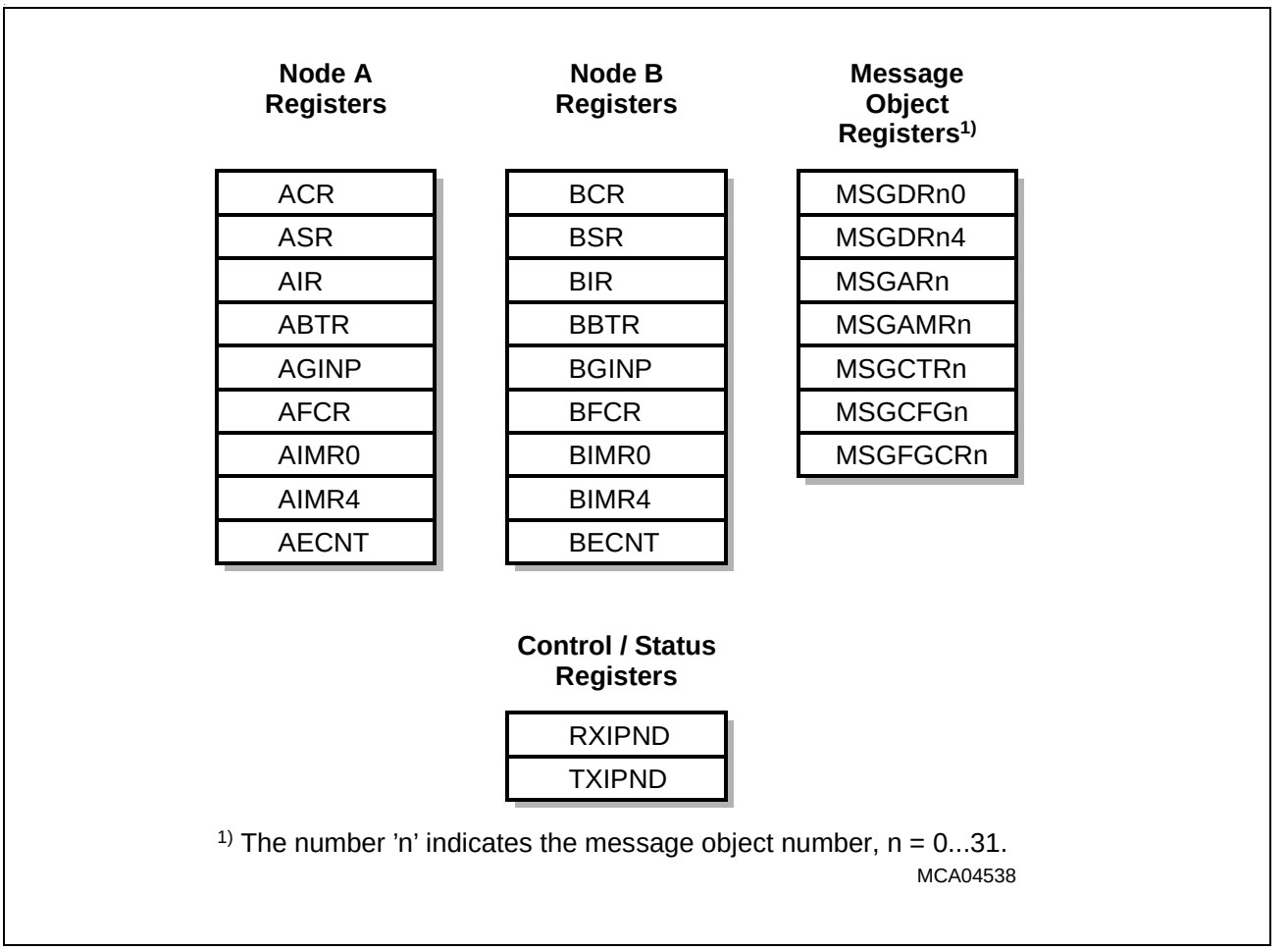


Figure 4-24 TwinCAN Kernel Registers

Table 4-6 TwinCAN Kernel Registers

Register Short Name	Register Long Name	Offset Address	Description see
ACR	Node A Control Register	0200_H	Page 4-49
ASR	Node A Status Register	0204_H	Page 4-51
AIR	Node A Interrupt Pending Register	0208_H	Page 4-54
ABTR	Node A Bit Timing Register	$020C_H$	Page 4-57
AGINP	Node A Global Int. Node Pointer Register	0210_H	Page 4-62
AFCR	Node A Frame Counter Register	0214_H	Page 4-59

Table 4-6 TwinCAN Kernel Registers (cont'd)

Register Short Name	Register Long Name	Offset Address	Description see
AIMR0	Node A INTID Mask Register 0	0218_H	Page 4-64
AIMR4	Node A INTID Mask Register 4	$021C_H$	Page 4-65
AECNT	Node A Error Counter Register	0220_H	Page 4-55
BCR	Node B Control Register	0240_H	Page 4-49
BSR	Node B Status Register	0244_H	Page 4-51
BIR	Node B Interrupt Pending Register	0248_H	Page 4-54
BBTR	Node B Bit Timing Register	$024C_H$	Page 4-57
BGINP	Node B Global Int. Node Pointer Register	0250_H	Page 4-62
BFCR	Node B Frame Counter Register	0254_H	Page 4-59
BIMR0	Node B INTID Mask Register 0	0258_H	Page 4-64
BIMR4	Node B INTID Mask Register 4	$025C_H$	Page 4-65
BECNT	Node B Error Counter Register	0260_H	Page 4-55
RXIPND	Receive Interrupt Pending Register	0284_H	Page 4-80
TXIPND	Transmit Interrupt Pending Register	0288_H	Page 4-81
MSGDRn0	Message Object n Data Register 0 (n = 31-0)	$0300_H + n \times 20_H$	Page 4-66
MSGDRn4	Message Object n Data Register 4 (n = 31-0)	$0304_H + n \times 20_H$	Page 4-66
MSGARn	Message Object n Arbitration Register (n = 31-0)	$0308_H + n \times 20_H$	Page 4-67
MSGAMRn	Message Object n Acceptance Mask Register (n = 31-0)	$030C_H + n \times 20_H$	Page 4-67
MSGCTRn	Message Object n Message Control Register (n = 31-0)	$0310_H + n \times 20_H$	Page 4-68
MSGCFGn	Message Object n Message Configuration Register (n = 31-0)	$0314_H + n \times 20_H$	Page 4-72
MSGFGCRn	Message Object n FIFO/Gateway Control Register (n = 31-0)	$0318_H + n \times 20_H$	Page 4-74

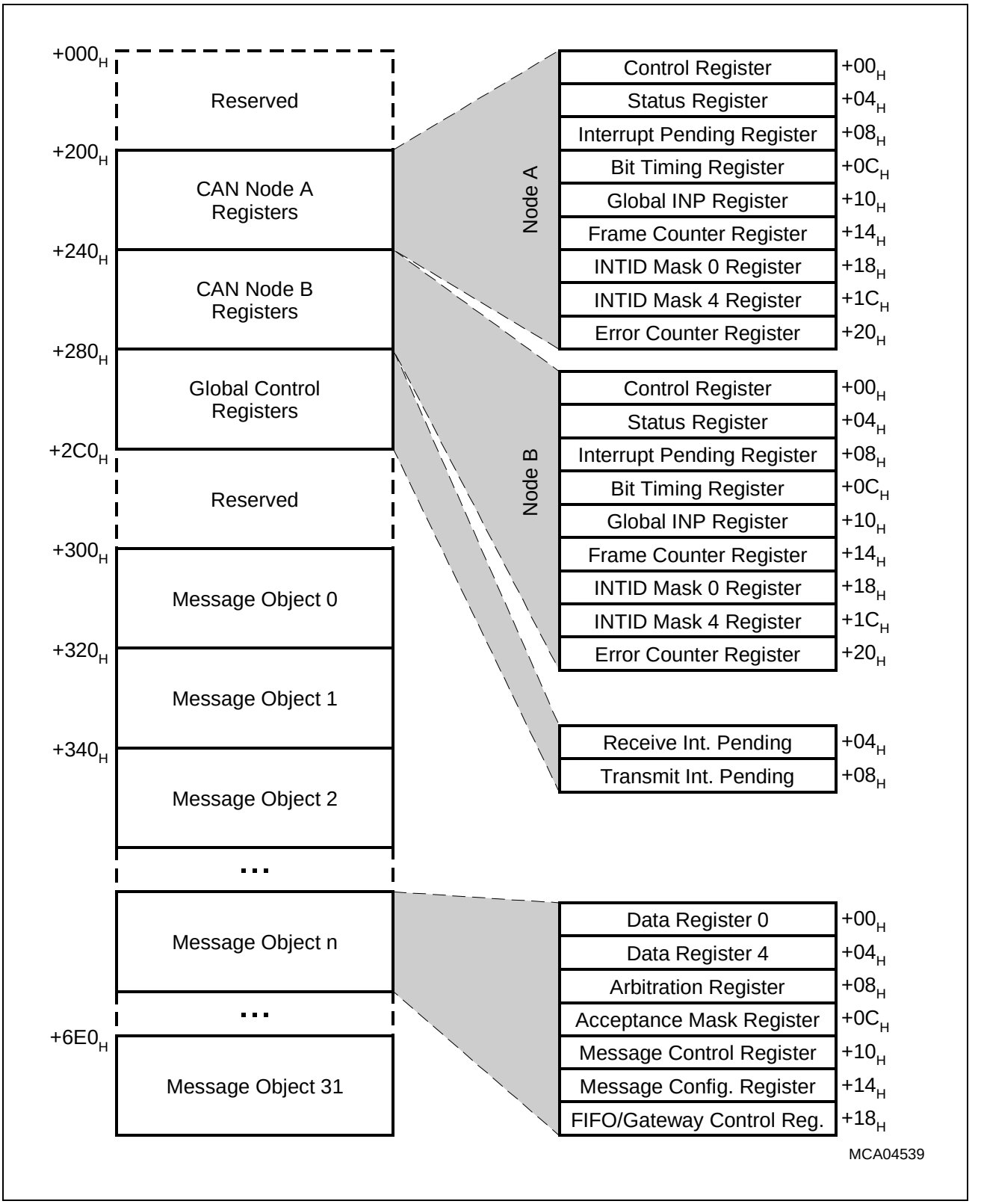
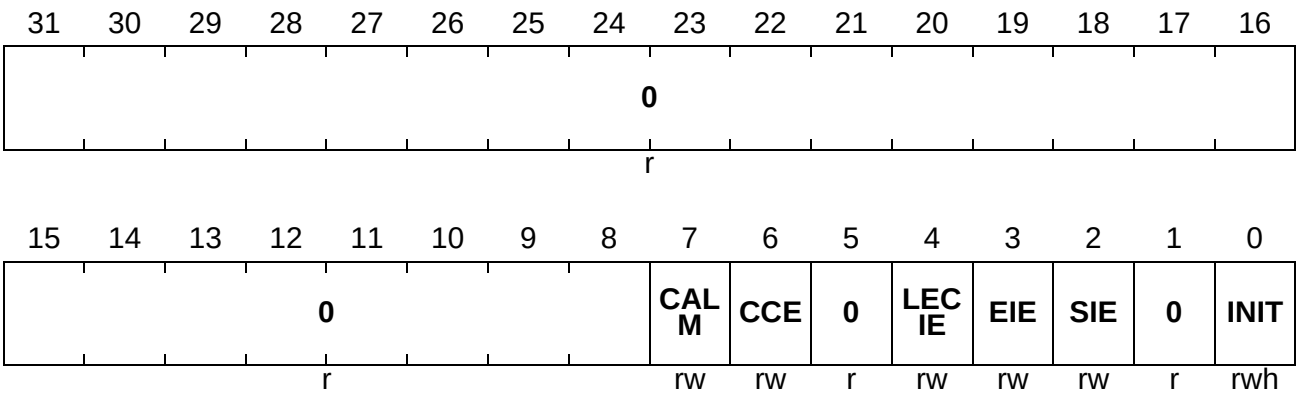


Figure 4-25 TwinCAN Kernel Address Map

4.2.2 CAN Node A / B Registers

The Node Control Register controls the initialization, defines the node specific interrupt handling, and selects an operation mode.

ACR
Node A Control Register **Reset Value: 0000 0001$_H$**
BCR
Node B Control Register **Reset Value: 0000 0001$_H$**



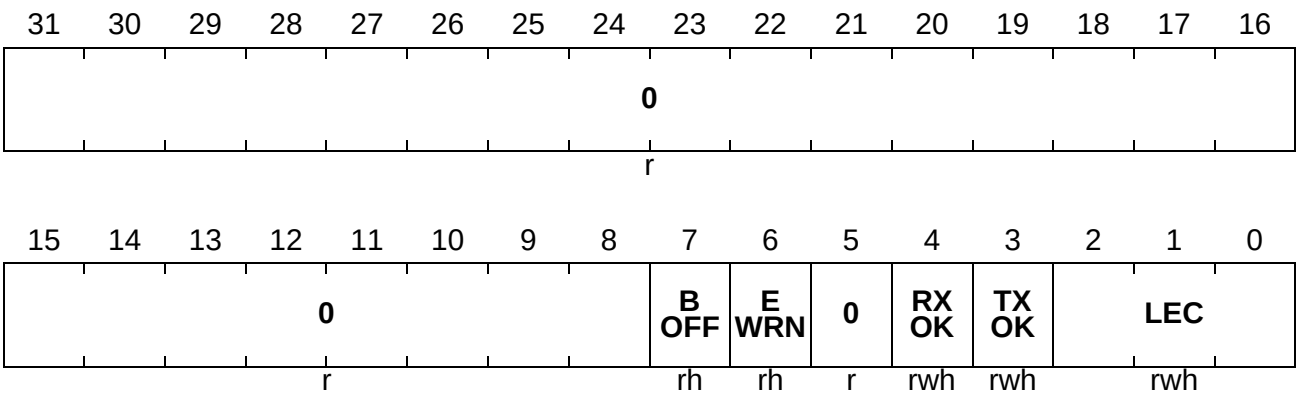
Field	Bits	Type	Description
INIT	0	rwh	**Initialization** 0 Resetting bit INIT starts the synchronization to the CAN bus. After a synchronization procedure[1)], the node takes part in CAN communication. 1 After setting bit INIT, the CAN node stops all CAN bus activities and all registers can be initialized without any influence on the actual CAN bus traffic. Bit INIT is automatically set when the “bus-off” state is entered.
SIE	2	rw	**Status Change Interrupt Enable** A status change interrupt occurs when a message transfer (indicated by the flags TXOK or RXOK in the status registers ASR or BSR) is successfully completed. 0 Status change interrupt is disabled 1 Status change interrupt is enabled

Field	Bits	Type	Description
EIE	3	rw	**Error Interrupt Enable** An error interrupt is generated on a change of bit BOFF or bit EWRN in the status registers ASR or BSR. 0 Error interrupt is disabled 1 Error interrupt is enabled
LECIE	4	rw	**Last Error Code Interrupt Enable** A last error code interrupt is generated when an error code is set in bit field LEC in the status registers ASR or BSR. 0 Last error code interrupt is disabled 1 Last error code interrupt is enabled
CCE	6	rw	**Configuration Change Enable** 0 Access to bit timing register and modification of the error counters are disabled 1 Access to bit timing register and modification of the error counters are enabled
CALM	7	rw	**CAN Analyzer Mode** Bit CALM defines if the message objects of the corresponding node operate in analyzer mode 0 The CAN message objects participate in CAN protocol 1 CAN Analyzer Mode is selected
0	1, 5, [31:8]	r	**Reserved**; returns 0 if read; should be written with 0.

1) After resetting bit INIT by software without being in the “bus-off” state (such as after power-on), a sequence of 11 consecutive recessive bits (11×1) on the bus must be monitored before the module takes part in the CAN traffic.
During a “bus-off” recovery procedure, 128 sequences of 11 consecutive recessive bits (11×1) must be detected. The monitoring of the recessive bit sequences is immediately started by hardware after entering the “bus-off” state. The number of already detected 11×1 sequences is indicated by the receive error counter.
At the end of the “bus-off” recovery sequence, bit INIT is tested by hardware. If INIT is still set, the affected TwinCAN node controller waits until INIT is cleared and 11 consecutive recessive bits (11×1) are detected on the CAN bus, before the node takes part in CAN traffic again. If INIT has been already cleared, the message transfer between the affected TwinCAN node controller and its associated CAN bus is immediately enabled.

The Node Status Register reports error states and successfully ended data transmissions. This register must be read in order to release the status change interrupt request.

ASR
Node A Status Register **Reset Value: 0000 0000$_H$**
BSR
Node B Status Register **Reset Value: 0000 0000$_H$**

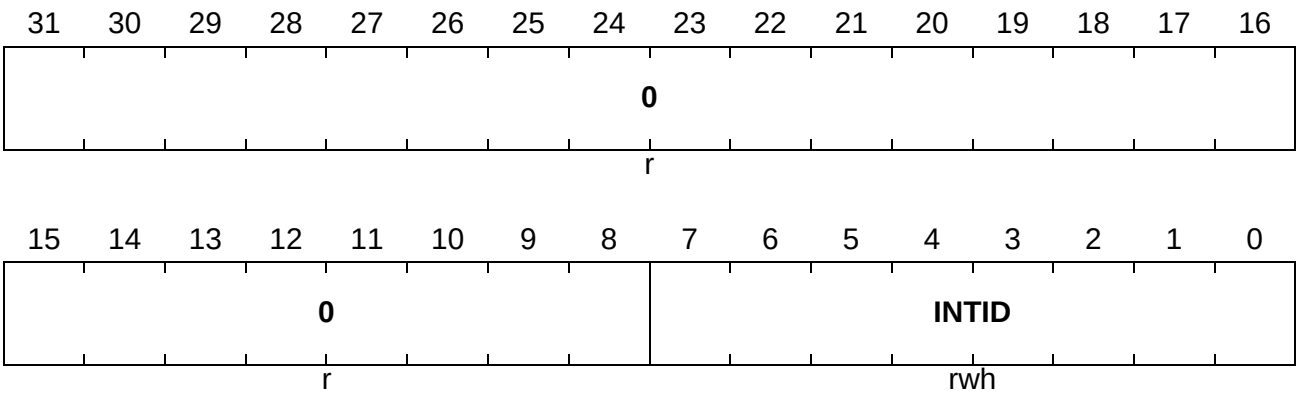


Field	Bits	Type	Description
LEC	[2:0]	rwh	**Last Error Code** 000_B No error 001_B Stuff Error: More than 5 equal bits in a sequence have occurred in a part of a received message where this is not allowed. 010_B Form Error: A ‘fixed format part’ of a received frame has the wrong format. 011 Ack Error: 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), but the monitored bus value was *dominant* (outside the arbitration field and the acknowledge slot). 101_B Bit0 Error: Two different conditions are signaled 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 Reserved
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 one other node). TXOK must be reset by software.
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.

Field	Bits	Type	Description
EWRN	6	rh	**Error Warning Status** 0 No warning limit exceeded 1 One of the error counters in the Error Management Logic reached the error warning limit of 96.
BOFF	7	rh	**Bus-off Status** 0 CAN controller is not in the “bus-off” state 1 CAN controller is in the “bus-off” state
0	5, [31:8]	r	**Reserved**; returns 0 if read; should be written with 0.

The Interrupt Pending Register contains the ID number of the pending interrupt request with the highest priority.

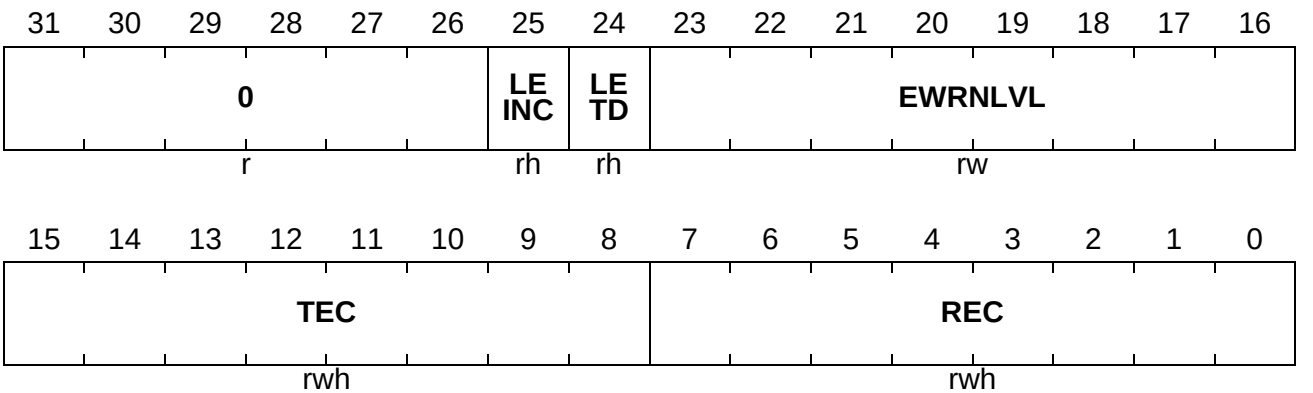
AIR
Node A Interrupt Pending Register **Reset Value: 0000 0000$_H$**
BIR
Node B Interrupt Pending Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
INTID	[7:0]	rwh	**Interrupt Identifier** 00_H No interrupt is pending 01_H LEC, EI, TXOK or RXOK interrupt is pending 02_H RX or TX interrupt of message object 0 is pending 03_H RX or TX interrupt of message object 1 is pending … 21_H RX or TX interrupt of message object 31 is pending Bit field INTID can be written by software to start an update after software actions and to check for changes.
0	[31:8]	r	**Reserved**; returns 0 if read.

Register AECNT/BECNT contains the values of the receive error counter and the transmit error counter. Some additional status/ control bits allow for easier error analysis.

AECNT
Node A Error Counter Register **Reset Value: 0060 0000$_H$**
BECNT
Node B Error Counter Register **Reset Value: 0060 0000$_H$**



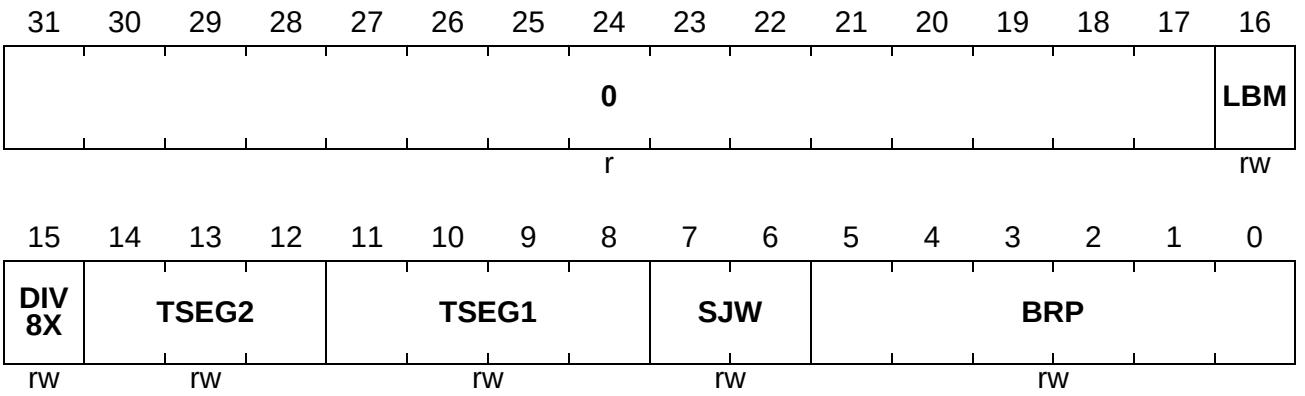
Field	Bits	Type	Description
REC	[7:0]	rwh	**Receive Error Counter** Bit field REC contains the value of the receive error counter for the corresponding node.
TEC	[15:8]	rwh	**Transmit Error Counter** Bit field TEC contains the value of the transmit error counter for the corresponding node.
EWRNLVL	[23:16]	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	24	rh	**Last Error Transfer Direction** 0 The last error occurred while the corresponding CAN node was receiving a message (REC has been incremented). 1 The last error occurred while the corresponding CAN node was transmitting a message (TEC has been incremented). An error during message reception is indicated without regard to the result of the acceptance filtering.

Field	Bits	Type	Description
LEINC	25	rh	**Last Error Increment** 0 The error counter was incremented by 1 due to the error reported by LETD. 1 The error counter was incremented by 8 due to the error reported by LETD.
0	[31:26]	r	**Reserved**; returns 0 if read; should be written with 0.

Note: Modifying the contents of register AECNT/BECNT requires bit CCE = 1 in register ACR/BCR.

The Bit Timing Register contains all parameters to adjust the data transfer baud rate.

ABTR
Node A Bit Timing Register **Reset Value: 0000 0000$_H$**
BBTR
Node B Bit Timing Register **Reset Value: 0000 0000$_H$**



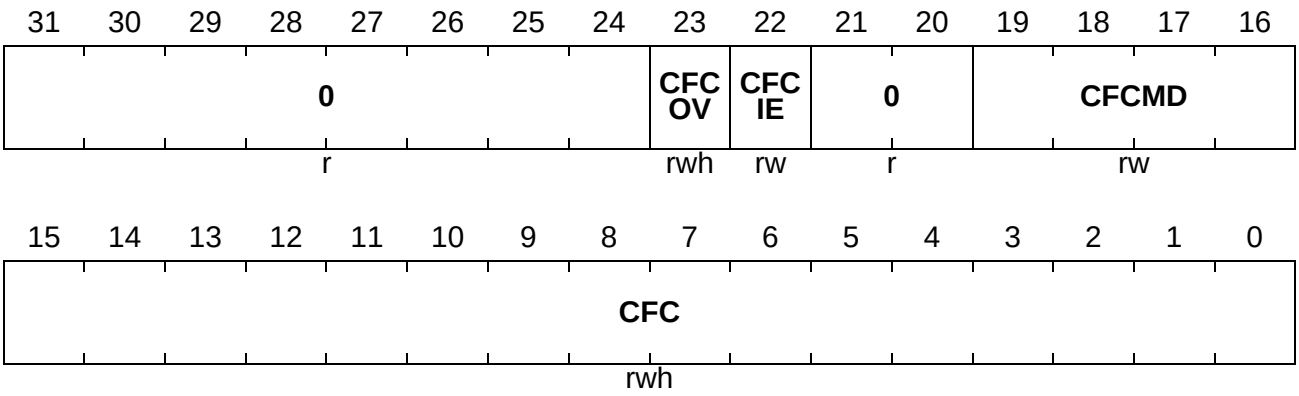
Field	Bits	Type	Description
BRP	[5:0]	rw	**Baud Rate Prescaler** One bit time quantum corresponds to the period length of the external oscillator clock multiplied by (BRP + 1), depending also on bit DIV8X.
SJW	[7:6]	rw	**(Re)Synchronization Jump Width** (SJW + 1) time quanta are allowed for resynchronization.
TSEG1	[11:8]	rw	**Time Segment Before Sample Point** (TSEG1 + 1) time quanta before the sample point take into account the signal propagation delay and compensate for a mismatch between transmitter and receiver clock phase. Valid values for TSEG1 are 2 … 15.
TSEG2	[14:12]	rw	**Time Segment After Sample Point** (TSEG2 + 1) time quanta after the sample point take into account a user defined delay and compensate for a mismatch between transmitter and receiver clock phase. Valid values for TSEG2 are 1 … 7.
DIV8X	15	rw	**Division of Module Clock f_{CAN} by 8** 0 The baud rate prescaler is directly driven by f_{CAN}. 1 The baud rate prescaler is driven by f_{CAN}/8.

Field	Bits	Type	Description
LBM	16	rw	**Loop-Back Mode** 0 Loop-Back Mode is disabled 1 Loop-Back Mode is enabled, if bits LBM are set in the BTR registers of Node A and Node B.
0	[31:17]	r	**Reserved**; read as 0; should be written with 0.

Note: Modifying the contents of register ABTR/BBTR requires bit CCE = 1 in register ACR/BCR.

The Frame Counter Register controls the frame counter and provides status information.

AFCR
Node A Frame Counter Register **Reset Value: 0000 0000$_H$**
BFCR
Node B Frame Counter Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
CFC	[15:0]	rwh	**CAN Frame Counter** This bit field contains the count value of the frame counter. At the end of a correct message transfer, the value of CFC (captured value during SOF bit) is copied to bit field CFCVAL of the corresponding message object control register MSGCTRn.

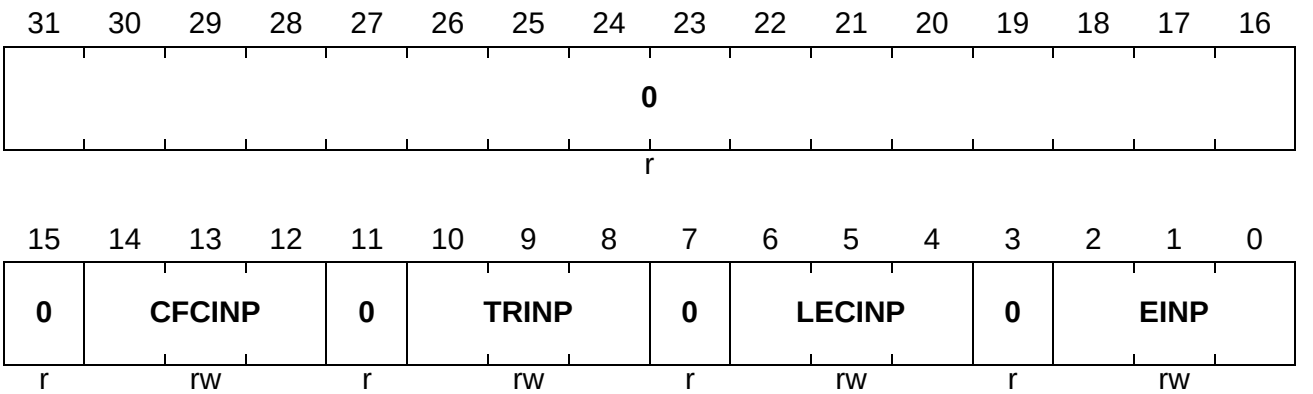
Field	Bits	Type	Description
CFCMD	[19:16]	rw	**Frame Count Mode** This bit field defines the operation mode of the frame counter. This counter can work on frame base (frame count) or on time base (time stamp). $0XXX_B$ **Frame Count:**[1)] $0XX0_B$ The CFC is not incremented after a foreign frame was transferred on the CAN bus. $0XX1_B$ The CFC is incremented each time a foreign frame was transferred correctly on the CAN bus. $0X0X_B$ The CFC is not incremented after a frame was received by the respective CAN node. $0X1X_B$ The CFC is incremented each time a frame was received correctly by the node. $00XX_B$ The CFC is not incremented after a frame was transmitted by the node. $01XX_B$ The CFC is incremented each time a frame was transmitted correctly by the node. $1XXX_B$ **Time Stamp:** 1000_B The CFC is incremented with the beginning of a new bit time. The value is sampled during the SOF bit. 1001_B The CFC is incremented with the beginning of a new bit time. The value is sampled during the last bit of EOF. others Reserved
CFCIE	22	rw	**CAN Frame Count Interrupt Enable** Setting CFCIE enables the CAN Frame Counter Overflow (CFCO) interrupt request. 0 The CFCO interrupt is disabled. 1 The CFCO interrupt is enabled.
CFCOV	23	rwh	**CAN Frame Count Overflow Flag** Flag CFCOV is set on a CFC overflow condition ($FFFF_H$ to 0000_H). An interrupt request is generated if the corresponding interrupt is enabled (CFCIE = 1). 0 An overflow has not yet been detected. 1 An overflow has been detected since the bit has been reset. CFCOV must be reset by software.

Field	Bits	Type	Description
0	[21:20], [31:24]	r	**Reserved**; read as 0; should be written with 0.

1) If the frame counter functionality has been selected (CFCMD.3 = 0), bit CFCMD.0 enables or disables the counting of foreign frames. A foreign frame is a correct frame on the bus that has not been transmitted/received by the node itself. Bit CFCMD.1 enables or disables the counting of frames that have been received correctly by the corresponding CAN node. Bit CFCMD.2 enables or disables the counting of frames that have been transmitted correctly by the corresponding CAN node.

The Global Interrupt Node Pointer Register connects each global interrupt request source with one of the eight available interrupt nodes.

AGINP
Node A Global Interrupt Node Pointer Register **Reset Value: 0000 0000$_H$**
BGINP
Node B Global Interrupt Node Pointer Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
EINP	[2:0]	rw	**Error Interrupt Node Pointer** Number of interrupt node reporting the “Error Interrupt Request”, if enabled by EIE = 1. 000_B CAN interrupt node 0 is selected … 111_B CAN interrupt node 7 is selected
LECINP	[6:4]	rw	**Last Error Code Interrupt Node Pointer** Number of interrupt node reporting the “Last Error Interrupt Request”, if enabled by LECIE = 1. 000_B CAN interrupt node 0 is selected … 111_B CAN interrupt node 7 is selected

Field	Bits	Type	Description
TRINP	[10:8]	rw	**Transmit/Receive OK Interrupt Node Pointer** Number of interrupt node reporting the “Transmit and Receive Interrupt Request”, if enabled by SIE = 1. 000_B CAN interrupt node 0 is selected … 111_B CAN interrupt node 7 is selected
CFCINP	[14:12]	rw	**Frame Counter Interrupt Node Pointer** Number of interrupt node reporting the “Frame Counter Overflow Interrupt Request”, if enabled by CFCIE = 1. 000_B CAN interrupt node 0 is selected … 111_B CAN interrupt node 7 is selected
0	3, 7,11, [31:15]	r	**Reserved**; read as 0; should be written with 0.

The Interrupt ID Mask Registers allow disabling the ID notification of a pending interrupt request in the AIR/BIR register. The Interrupt Mask Registers AIMR0/BIMR0 are used to enable the message specific interrupt sources (correct transmission/ reception) for the generation of the corresponding INTID value.

AIMR0
Node A INTID Mask Register 0 **Reset Value: 0000 0000$_H$**
BIMR0
Node B INTID Mask Register 0 **Reset Value: 0000 0000$_H$**

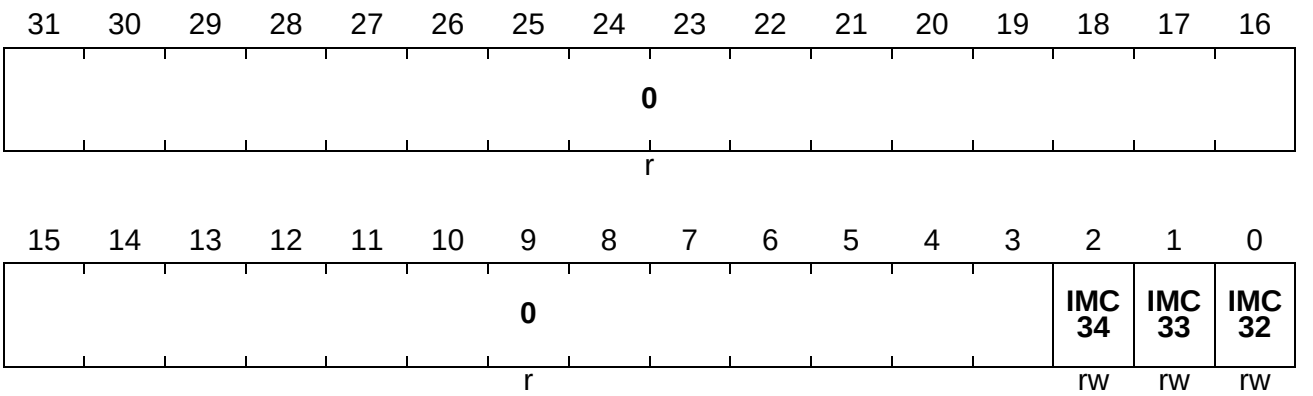
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IMC 31	**IMC 30**	**IMC 20**	**IMC 28**	**IMC 27**	**IMC 26**	**IMC 25**	**IMC 24**	**IMC 23**	**IMC 22**	**IMC 21**	**IMC 20**	**IMC 19**	**IMC 18**	**IMC 17**	**IMC 16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Field	Bits	Type	Description
IMCn (n = 31-0)	n	rw	**Message Object n INTID Mask Control** 0 Message object n is ignored for the generation of the INTID value. 1 The interrupt pending status of message object n is taken into account for the generation of the INTID value.

The Interrupt Mask Registers AIMR4/BIMR4 are used to enable the node specific interrupt sources (last error, correct reception, error warning/bus-off) for the generation of the corresponding INTID value.

AIMR4
Node A INTID Mask Register 4 **Reset Value: 0000 0000$_H$**
BIMR4
Node B INTID Mask Register 4 **Reset Value: 0000 0000$_H$**

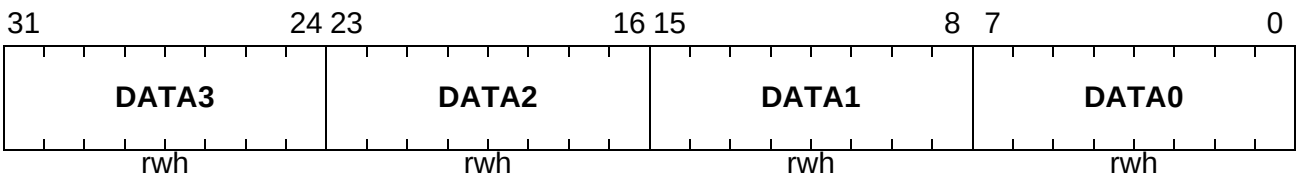


Field	Bits	Type	Description
IMC32	0	rw	**Last Error Interrupt INTID Mask Control** 0 The Last Error Interrupt source is ignored for the generation of the INTID value. 1 The Last Error Interrupt source is taken into account for the generation of the INTID value.
IMC33	1	rw	**TX/RX Interrupt INTID Mask Control** 0 The TX/RX Interrupt source is ignored for the generation of the INTID value. 1 The TX/RX Interrupt pending status is taken into account for the generation of the INTID value.
IMC34	2	rw	**Error Interrupt INTID Mask Control** 0 The Error Interrupt source is ignored for the generation of the INTID value. 1 The Error Interrupt pending status is taken into account for the generation of the INTID value.
0	[31:3]	r	**Reserved**; read as 0; should be written with 0.

4.2.3 CAN Message Object Registers

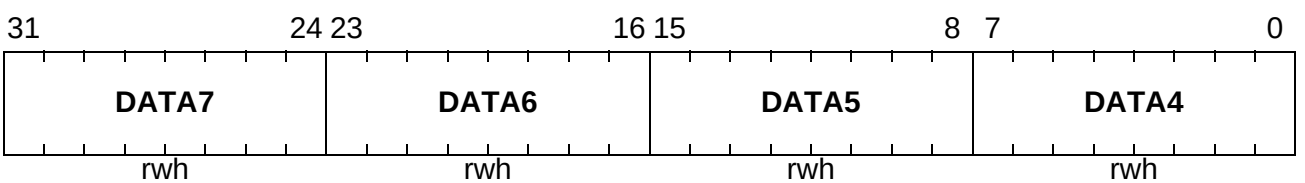
Each message object is provided with a set of control and data register. The corresponding register names are supplemented with a variable n running from 0 to 31 (for example, MSGDRn0 means that data register MSGDR300 is assigned with message object number 30).

MSGDRn0 (n = 31-0)
Message Object n Data Register 0 **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
DATA0	[7:0]	rwh	**Data Byte 0 associated to Message Object n**
DATA1	[15:8]	rwh	**Data Byte 1 associated to Message Object n**
DATA2	[23:16]	rwh	**Data Byte 2 associated to Message Object n**
DATA3	[31:24]	rwh	**Data Byte 3 associated to Message Object n**

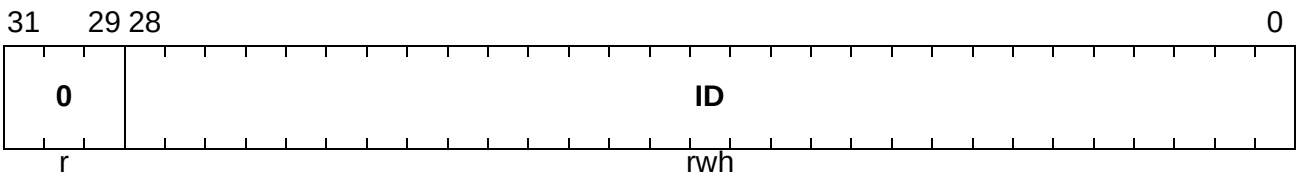
MSGDRn4 (n = 31-0)
Message Object n Data Register 4 **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
DATA4	[7:0]	rwh	**Data Byte 4 associated to Message Object n**
DATA5	[15:8]	rwh	**Data Byte 5 associated to Message Object n**
DATA6	[23:16]	rwh	**Data Byte 6 associated to Message Object n**
DATA7	[31:24]	rwh	**Data Byte 7 associated to Message Object n**

Register MSGARn contains the identifier of message object n.

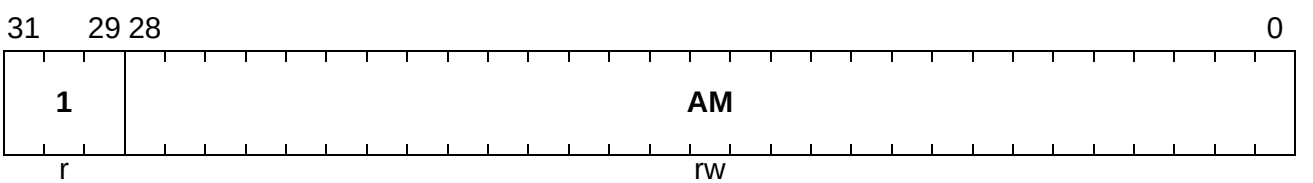
MSGARn (n = 31-0)
Message Object n Arbitration Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
ID	[28:0]	rwh	**Message Identifier** 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”.
0	[31:29]	r	**Reserved**; returns 0 if read; should be written with 0.

Register MSGAMRn contains the mask bits for the acceptance filtering of message object n.

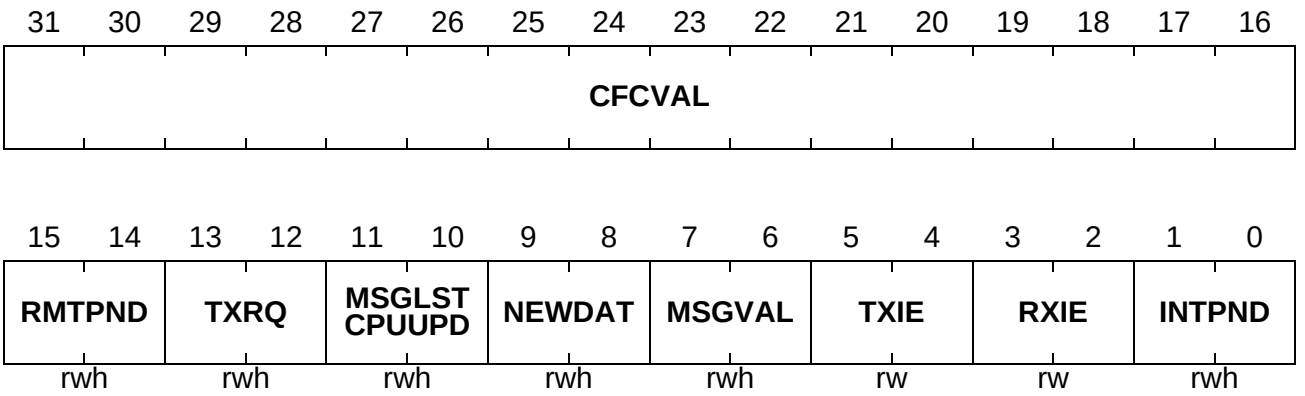
MSGAMRn (n = 31-0)
Message Object n Acceptance Mask Register **Reset Value: FFFF FFFF$_H$**



Field	Bits	Type	Description
AM	[28:0]	rw	**Message Acceptance Mask** 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”. 0 Identifier bit is ignored for acceptance test 1 Identifier bit is taken into account for the acceptance filtering
1	[31:29]	r	**Reserved**; returns 1 if read; should be written with 1.

Register MSGCTRn affects the data transfer between a TwinCAN node controller and the corresponding message object n and provides a bit field to store a snapshot of the frame counter value.

MSGCTRn (n = 31-0)
Message Object n Message Control Register **Reset Value: 0000 5555$_H$**



Field	Bits	Type	Description
INTPND	[1:0]	rwh	**Message Object Interrupt Pending** INTPND is generated by an “OR” operation between the RXIPNDn and TXIPNDn flags (if enabled by TXIE or RXIE). INTPND must be reset by software. Resetting INTPND also resets the corresponding RXIPND and TXIPND flags. 01 No message object interrupt request is pending. 10 The message object has generated an interrupt request.
RXIE	[3:2]	rw	**Message Object Receive Interrupt Enable** 01 Message object receive interrupt is disabled. 10 Message object receive interrupt is enabled. If RXIE is set, bits INTPND and RXIPND are set after successful reception of a frame.
TXIE	[5:4]	rw	**Message Object Transmit Interrupt Enable** 01 Message object transmit interrupt is disabled. 10 Message object transmit interrupt is enabled. If TXIE is set, bits INTPND and TXIPND are set after successful transmission of a frame.

Field	Bits	Type	Description
MSGVAL[1)]	[7:6]	rwh	**Message Object Valid** The CAN controller only operates on valid message objects. Message objects can be tagged invalid while they are changed or if they are not used at all. 01 Message object is invalid 10 Message object is valid
NEWDAT[2)]	[9:8]	rwh	**New Message Object Data Available** 01 No update of message object data occurred 10 New message object data has been updated
MSGLST	[11:10]	rwh	**Message Lost** (for reception only) 01 No message object data is lost 10 The CAN controller has stored a new message into the message object while NEWDAT was still set. The previously stored message is lost MSGLST must be reset by software.
CPUUPD[3)]			**CPU Update** (for transmission only) Indicates that the corresponding message object cannot be transmitted now. The software sets this bit in order to inhibit the transmission of a message that is currently updated by the CPU or to control the automatic response to remote requests. 01 The message object data can be transmitted automatically by the CAN controller. 10 The automatic transmission of the message data is inhibited.
TXRQ[4)]	[13:12]	rwh	**Message Object Transmit Request Flag** 01 No message object data transmission is requested by the CPU or a remote frame. 10 The transmission of the message object data, requested by the CPU or by a remote frame, is pending. Automatic setting of TXRQ by the TwinCAN node controller can be disabled for Gateway Message Objects via control bit GDFS = 0. TXRQ is automatically reset, when the message object has been successfully transmitted. If there are several valid message objects with pending transmit requests, the message object with the lowest message number will be transmitted first.

Field	Bits	Type	Description
RMTPND	[15:14]	rwh	**Remote Pending Flag** (used for transmit-objects) 01 No remote node request for a message object data transmission. 10 Transmission of the message object data has been requested by a remote node but the data has not yet been transmitted. When RMTPND is set, the TwinCAN node controller also sets TXRQ. RMTPND is automatically reset, when the message object data has been successfully transmitted.
CFCVAL	[31:16]	rwh	**Message Object Frame Counter Value** CFCVAL contains a copy of the frame counter content valid at the end of the last data transmission or reception executed for the corresponding message object.

1) MSGVAL has to be set from 01_B to 10_B in order to take into account an update of bits XTD, DIR, NODE and CANPTR.

2) Bit NEWDAT indicates that new data has been written into the data registers of this corresponding message object. For transmit objects, NEWDAT should be set by software and is reset by the respective TwinCAN node controller when the transmission is started.
For receive objects, NEWDAT is set by the respective TwinCAN node controller after receiving a data frame with matching identifier. It has to be reset by software.
When the CAN controller writes new data into the message object, unused message bytes will be overwritten with non-specified values. Usually, the CPU will clear this bit field before working on the data and will verify that the bit field is still cleared once the CPU has finished working to ensure a consistent set of data. For transmit objects, the CPU should set this bit field along with clearing bit field CPUUPD. This will ensure that, if the message is actually being transmitted during the time the message is updated by the CPU, the CAN controller will not reset bit field TXRQ. In this way, TXRQ is only reset once the actual data has been transferred correctly.

3) While bit field MSGVAL is set (10_B) an incoming matching remote frame is taken into account by automatically setting bit fields TXRQ and RMTPND to 10_B (independent from bit field CPUUPD/MSGLST). The transmission of a frame is only possible if CPUUPD is reset (01_B).

4) If a receive object (DIR = 0) is requested for transmission, a remote frame will be sent in order to request a data frame from another node. If a transmit object (DIR = 1) is requested for transmission, a data frame will be sent. Bit field TXRQ will be reset by the CAN controller along with bit field RMTPND after the correct transmission of the data frame if bit field NEWDAT has not been set or after correct transmission of a remote frame.

Note: For transmitting frames (remote frames or data frames), bit field CPUUPD/ MSGLST must be reset.

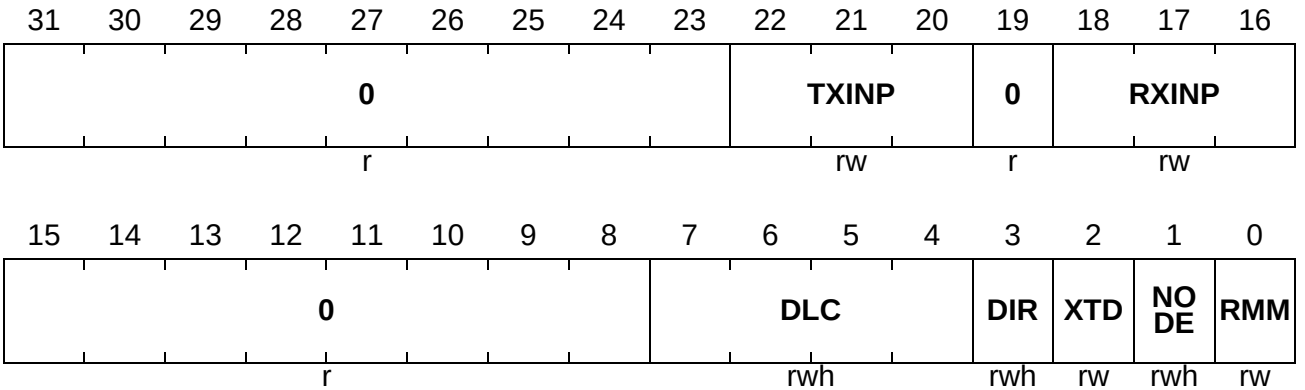
Each control and status element of the MSGCTRn register is implemented with two complementary bits. This special mechanism allows the selective setting or resetting of a specific element (leaving others unchanged) without requiring read-modify-write cycles. **Table 4-7** illustrates how to use these 2-bit bit fields.

Table 4-7 Setting/Resetting the Control and Status Element of the MSGCTRn Register

Value of the 2-bit Bit Field	Function on Write	Meaning on Read
00_B	Reserved	Reserved
01_B	Reset element	Element is reset
10_B	Set element	Element is set
11_B	Leave element unchanged	Reserved

Register MSGCFGn defines the configuration of message object n and the associated interrupt node pointers. Changes of bits XTD, NODE or DIR by software are only taken into account after setting bit field MSGVAL to 10_B. This avoids unintentional modification while the message object is still active by explicitly defining a timing instant for the update. Bits XTD, NODE or DIR can be written while MSGVAL is 01_B or 10_B, the update always takes place by setting MSGVAL to 10_B.

MSGCFGn (n = 31-0)
Message Object n Message Configuration Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
RMM	0	rw	**Transmit Message Object Remote Monitoring Mode** 0 Remote Monitoring mode is disabled. 1 Remote Monitoring mode is enabled for this transmit message object. The identifier and DLC code of a remote frame with matching identifier are copied to this transmit message object in order to monitor incoming remote frames. Bit RMM is only available for transmit objects and has no influence for receive objects.
NODE	1	rwh	**Message Object CAN Node Select** 0 The message object is assigned to CAN node A. 1 The message object is assigned to CAN node B.
XTD	2	rw	**Message Object Extended Identifier** 0 This message object uses a standard 11-bit identifier. 1 This message object uses an extended 29-bit identifier.

Field	Bits	Type	Description
DIR	3	rwh	**Message Object Direction Control** 0 The message object is defined as receive object. If TXRQ = 10_B, a remote frame with the identifier of this message object is transmitted. On reception of a data frame with matching identifier, the message data is stored in the corresponding MSGDRn0/MSGDRn4 registers. 1 The message object is declared as transmit object. If TXRQ = 10_B, the respective data frame is transmitted. On reception of a remote frame with matching identifier, RMTPND and TXRQ are set to 10_B.
DLC[1)]	[7:4]	rwh	**Message Object Data Length Code** 0000_B - $1XXX_B$ DLC contains the number of data bytes associated to the message object. Bit field DLC may be modified by hardware in Remote Monitoring Mode and in Gateway Mode.
RXINP	[18:16]	rw	**Receive Interrupt Node Pointer** Bit field RXINP determines which interrupt node is triggered by a message object receive event, if bit field RXIE in register MSGCTRn is set. 000_B CAN interrupt node 0 is selected … 111_B CAN interrupt node 7 is selected
TXINP	[22:20]	rw	**Transmit Interrupt Node Pointer** Bit field TXINP determines which interrupt node is triggered by a message object transmit event, if bit field TXIE in register MSGCTRn is set. 000_B CAN interrupt node 0 is selected … 111_B CAN interrupt node 7 is selected
0	[15:8], 19, [31:23]	r	**Reserved**; returns 0 if read; should be written with 0.

1) The maximum number of data bytes is eight. A value > 8 written by the CPU, is internally corrected to eight but the content of bit field DLC is not updated.
If a received data frame contains a data length code value > 8, only eight bytes are taken into account. A read access to bit field DLC returns the original value of the DLC bit field of the received data frame.

The FIFO/gateway control register MSGFGCRn contains bits to enable and to control the FIFO functionality, the gateway functionality and the desired transfer actions.

MSGFGCRn (n = 31-0)
CAN FIFO/Gateway Control Register n **Reset Value: 0000 0000$_H$**

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STT	SDT	FD	0	DLCC	IDC	SRREN	GDFS	0			FSIZE				
rw	rw	rw	r	rw	rw	rw	rw	r			rw				

Field	Bits	Type	Description
FSIZE	[4:0]	rw	**FIFO Size Control** Bit field FSIZE determines the number of message objects combined to a FIFO buffer. Even numbered message objects may provide FIFO base or slave functionality, while odd numbered message objects are restricted to slave functionality. In gateway mode, FSIZE determines the length of the FIFO on the destination side. 00000_B Message object n is part of a 1-stage FIFO 00001_B Message object n is part of a 2-stage FIFO 00011_B Message object n is part of a 4-stage FIFO 00111_B Message object n is part of a 8-stage FIFO 01111_B Message object n is part of a 16-stage FIFO 11111_B Message object n is part of a 32-stage FIFO else Reserved FSIZE = 00000_B leads to the behavior of a standard message object (the pointer CANPTR used for this action will not be changed). This value must be written if a gateway transfer to a single message object (no FIFO) as destination is desired. FSIZE is not evaluated for message objects configured in standard mode, shared gateway mode or FIFO slave functionality. In this case, FSIZE should be programmed to 00000_B.

Field	Bits	Type	Description
GDFS	8	rw	**Gateway Data Frame Send** Specifies if a CAN data frame will be automatically generated on the destination side after new data has been transferred via gateway from the source to the destination side. 0 No additional action. TXRQ will not be set on the destination side. 1 The corresponding data frame will be sent automatically (TXRQ of the message object, pointed to by CANPTRn, will be set by hardware). Bit GDFS is only taken into account, if a data frame has been received ($DIR_{<S>}$ = 0).
SRREN	9	rw	**Source Remote Request Enable** Specifies if the transmit request bit is set in message object n itself (to generate a data frame) or in the message object pointed to by CANPTRn (in order to generate a remote frame on the source bus). 0 A remote on the source bus will not be generated, a data frame with the contents of the destination object will be generated on the destination bus, instead (TXRQn will be set). 1 A data frame with the contents of the destination object will not be sent. Instead, a corresponding remote frame will be generated by the message object pointed to by bit field CANPTRn (TXRQ[CANPTRn] will be set). SRREN is restricted to transmit message objects in normal or shared gateway mode (DIR = 1). This bit is only taken into account if a remote frame has been received. Bit SRREN must not be set if message object n is part of a FIFO buffer. In order to generate a remote frame on the source side, CANPTR must point to the source message object.

Field	Bits	Type	Description
IDC	10	rw	**Identifier Copy** IDC controls the identifier handling during a frame transfer through a gateway. 0 The identifier of the receiving object is not copied to the transmitting message object. 1 The identifier of the receiving object is automatically copied to the transmitting message object. Bit field IDC is restricted to message objects configured in normal gateway mode.
DLCC	11	rw	**Data Length Code Copy** DLCC controls the handling of the data length code during a data frame transfer through a gateway. 0 The data length code, provided by the source object, is not copied to the transmitting object. 1 The data length code, valid for the receiving object, is copied automatically to the transmitting object. Bit field DLCC is restricted to message objects configured in normal gateway mode.

Field	Bits	Type	Description
FD	13	rw	**FIFO Direction** FD is only taken into account for a FIFO base object (the FD bits of all FIFO elements should have an identical value). It defines which transfer action (reception or transmission) leads to an update of the FIFO base object's CANPTR. 0 FIFO Reception: The CANPTR (of the FIFO base object) is updated after a correct reception of a data frame (DIR = 0) or a remote frame (DIR = 1) by the currently addressed message object. The CANPTR is left unchanged after any transmission. 1 FIFO Transmission: The CANPTR (of the FIFO base object) is updated after a correct transmission of a data frame (DIR = 1) or a remote frame (DIR = 0) from the currently addressed message object. The CANPTR is left unchanged after any reception. Bit field FD is not correlated with bit DIR.
SDT	14	rw	**Single Data Transfer Mode** This bit is taken into account in any transfer mode (FIFO mode or as standard object, receive and transmit objects). 0 Control bit MSGVAL is not reset when this object has taken part in a successful data transfer (receive or transmit). 1 Control bit MSGVAL is automatically reset after a successful data transfer (receive or transmit) has taken place. Bit SDT is not taken into account for remote frames. Bit SDT must be reset in all message objects belonging to a FIFO buffer.
STT	15	rw	**Single Transmission Try** 0 Single transmission try is disabled. 1 Single transmission try is enabled. The corresponding TXRQ bit is reset immediately after the transmission has started.

Field	Bits	Type	Description
CANPTR	[20:16]	rwh	**CAN Pointer for FIFO/Gateway Functions** Message object is configured in standard mode ($MMC = 000_B$): No influence, CANPTR should be initialized with the respective message object number. Message object is configured as FIFO base object ($MMC = 010_B$): CANPTR contains the number of the message object addressed by the associated CAN controller for the next transmit or receive operation. For initialization, CANPTR should be written with the message number of the respective FIFO base object. Message object is configured as FIFO slave object ($MMC = 011_B$): CANPTR must be initialized with the respective message object number of the FIFO base object. Message object is configured for normal gateway mode ($MMC = 100_B$): CANPTR contains the number of the message object used as gateway destination object. Message object is configured as gateway destination object without FIFO functionality ($MMC = 000_B$): If SRREN is set to 1, CANPTR must be initialized with the number of the message object used as gateway source. The backward pointer is required to transfer remote frames from the destination to the source side. If SRREN is cleared, CANPTR is not evaluated and must be initialized with the respective message object number. Message object is configured for shared gateway mode ($MMC = 101_B$): No influence, CANPTR must be initialized with the respective message object number. For FIFO functionality (or gateway functionality with a FIFO as destination), CANPTRn should not be written by software while FIFO mode is activated and data transfer is in progress. This bit field can be used to reset the FIFO by software.

Field	Bits	Type	Description
MMC	[24:26]	rw	**Message Object Mode Control** Bit field MMC controls the functionality of message object n. 000_B Standard message object functionality 010_B FIFO functionality enabled (base object) 011_B FIFO functionality enabled (slave object) 100_B Normal gateway functionality for incoming frames 101_B Shared gateway functionality for incoming frames others Reserved
0	[7:5], 12, [23:21], [31:27]	r	**Reserved**; returns 0 if read; should be written with 0.

Note: Changes of bit field CANPTR for transmission objects are taken into account only after setting bit field MSGVAL to 10_B. This avoids unintentional modification while the message object is still active by explicitly defining a timing instant for the update. Bit field CANPTR for transmission objects can be written while MSGVAL is 01_B or 10_B; the update always takes place by setting MSGVAL to 10_B. Changes to bit field CANPTR for receive objects are taken into account immediately.

4.2.4 Global CAN Control / Status Registers

The Receive Interrupt Pending Register indicates whether a receive interrupt is pending for message object n.

RXIPND
Receive Interrupt Pending Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXIPND 31	**RXIPND 30**	**RXIPND 20**	**RXIPND 28**	**RXIPND 27**	**RXIPND 26**	**RXIPND 25**	**RXIPND 24**	**RXIPND 23**	**RXIPND 22**	**RXIPND 21**	**RXIPND 20**	**RXIPND 19**	**RXIPND 18**	**RXIPND 17**	**RXIPND 16**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

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

Field	Bits	Type	Description
RXIPNDn (n = 31-0)	n	rh	**Message Object n Receive Interrupt Pending** Bit RXIPNDn is set by hardware if message object n received a frame and bit RXIEn has been set. 0 No receive is pending for message object n 1 Receive is pending for message object n RXIPNDn can be cleared by software via resetting the corresponding bit INTPNDn.

The Transmit Interrupt Pending Register indicates whether a transmit interrupt is pending for message object n.

TXIPND
Transmit Interrupt Pending Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXIPND 31	**TXIPND 30**	**TXIPND 20**	**TXIPND 28**	**TXIPND 27**	**TXIPND 26**	**TXIPND 25**	**TXIPND 24**	**TXIPND 23**	**TXIPND 22**	**TXIPND 21**	**TXIPND 20**	**TXIPND 19**	**TXIPND 18**	**TXIPND 17**	**TXIPND 16**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

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

Field	Bits	Type	Description
TXIPNDn (n = 31-0)	n	rh	**Message Object n Transmit Interrupt Pending** Bit TXIPNDn is set by hardware if message object n transmitted a frame and bit TXIEn has been set. 0 No transmit is pending for message object n 1 Transmit is pending for message object n TXIPNDn can be cleared by software via resetting the corresponding bit INTPNDn.

4.3 TwinCAN Module Implementation

This section describes the TwinCAN module interfaces with the clock control, port connections, interrupt control, and address decoding.

4.3.1 Interfaces of the TwinCAN Module

Figure 4-26 shows the TC1775 specific implementation details and interconnections of the CAN module. The TwinCAN module has four I/O lines, located at Port 13. The TwinCAN modules is further supplied by a clock control, interrupt control, and address decoding logic.

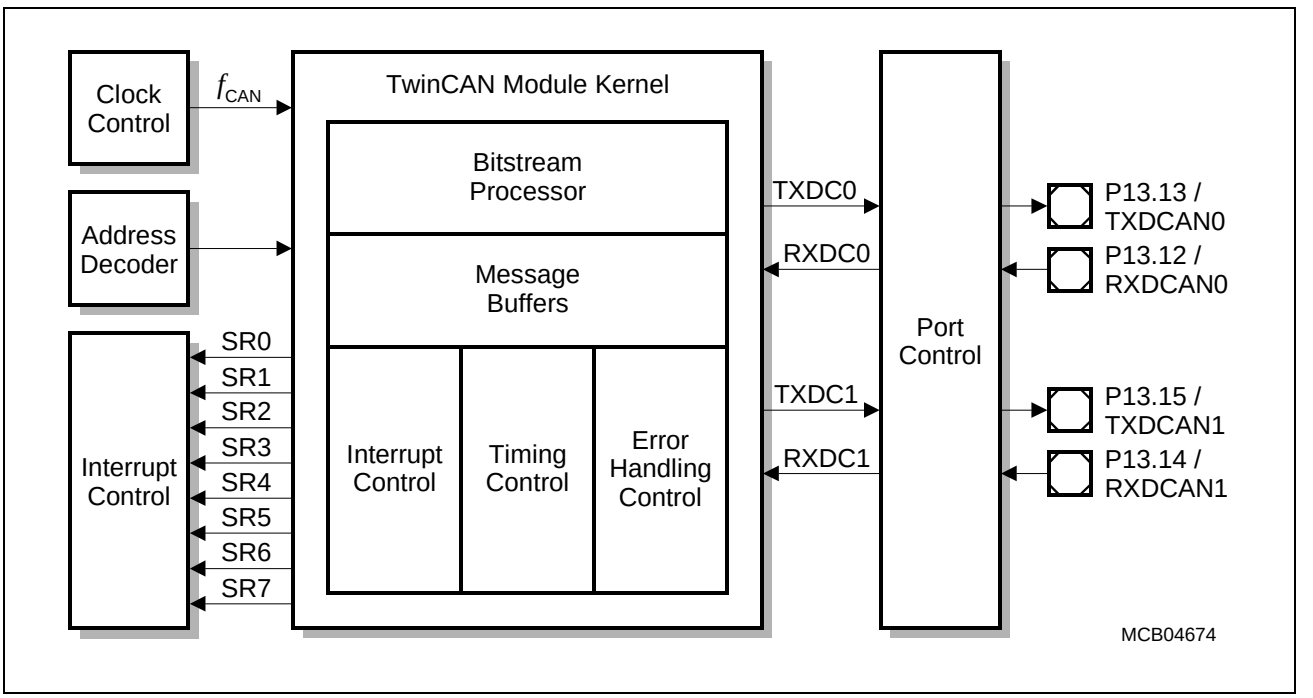


Figure 4-26 CAN Module Implementation and Interconnections

4.3.2 TwinCAN Module Start-Up Operation after Reset

When the TwinCAN module is switched on after a TC1775 reset, the kernel of the TwinCAN module is initialized; this lasts for 1000 CAN clock cycles (f_{CAN}). During this initialization phase, the TwinCAN module kernel register must not be accessed.

4.3.3 External Registers of the TwinCAN Module

The figure below summarizes the module related external registers which are required for CAN programming (see also **Figure 4-25** for the module kernel specific registers).

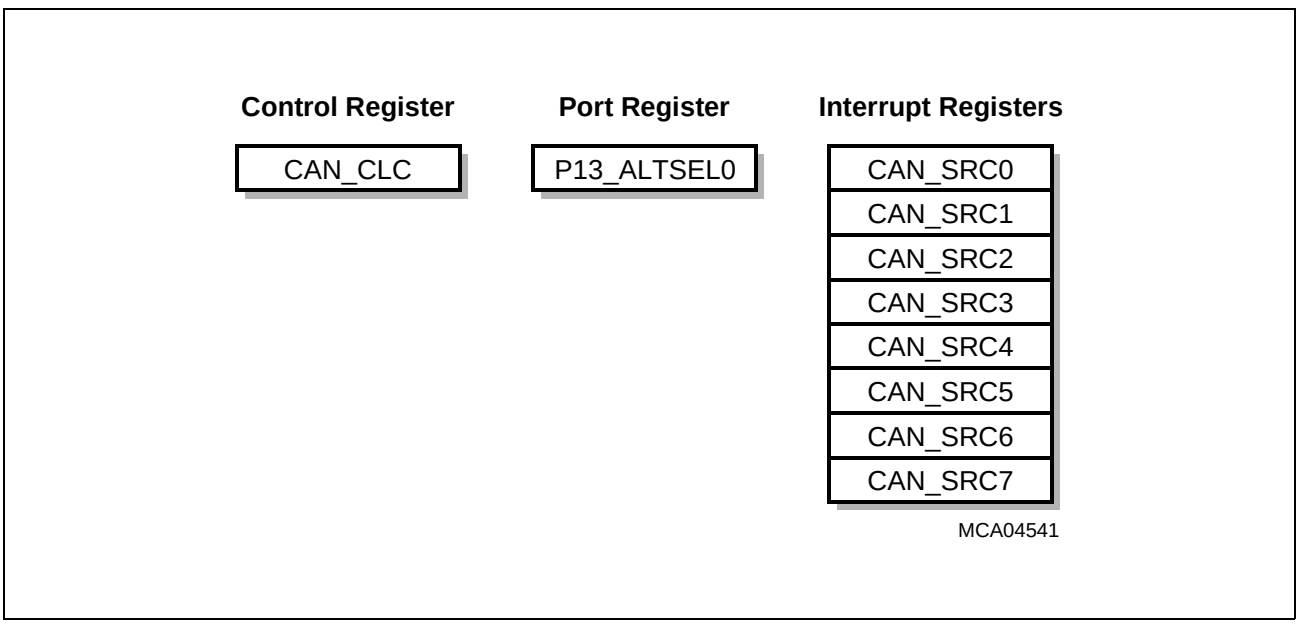
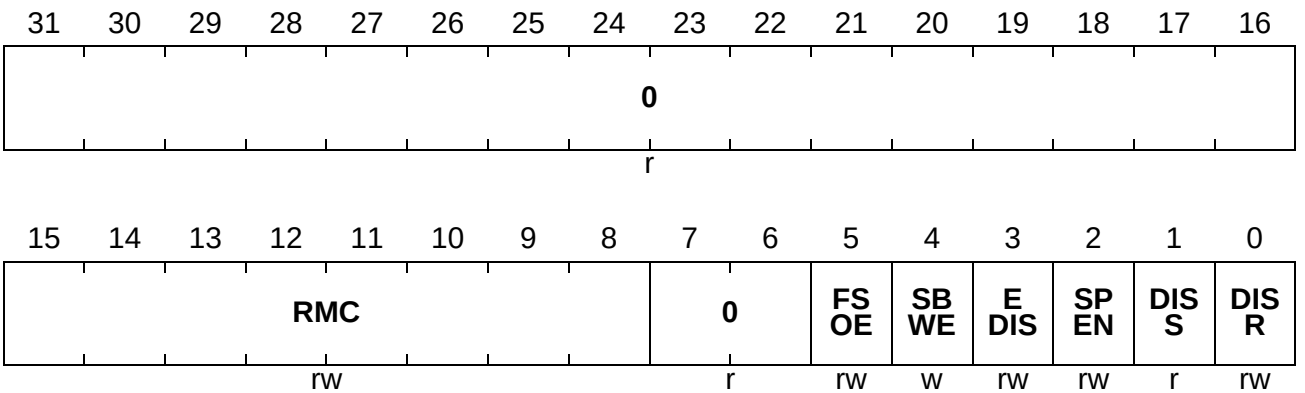


Figure 4-27 CAN Implementation Specific Special Function Registers

4.3.3.1 Clock Control Register

The clock control register allows the programmer to adapt the functionality and power consumption of the TwinCAN module to the requirements of the application. The diagram below shows the clock control register functionality implemented for the TwinCAN module.

CAN_CLC
CAN Clock Control Register **Reset Value: 0000 0002$_H$**



Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control of the module.
DISS	1	r	**Module Disable Status Bit** Bit indicates the current status of the module.
SPEN	2	rw	**Module Suspend Enable for OCDS** Used for enabling the suspend mode.
EDIS	3	rw	**External Request Disable** Used for controlling the external clock disable request.
SBWE	4	w	**Module Suspend Bit Write Enable for OCDS** Defines whether SPEN and FSOE are write protected.
FSOE	5	rw	**Fast Switch Off Enable** Used for fast clock switch off in OCDS suspend mode.
RMC	[15:8]	rw	**8-Bit Clock Divider Value in RUN Mode**
0	[7:6], [31:16]	r	**Reserved**; returns 0 if read; should be written with 0.

Note: The TwinCAN module is disabled after reset processing (DISS = 1).

Note: Careless use of FSOE may corrupt currently transferred messages or lead to drive a dominant level on the CAN bus.

4.3.3.2 Port Registers

The alternate functions associated with the TwinCAN module I/O lines, are controlled by the ALTSEL registers located in the ports. The TwinCAN module I/O lines are connected with Port 13. Therefore, P13_ALTSEL0 must be programmed for the Port 13 pins which are required for the CAN module in the specific application.

Note: Bits marked with 'X' are not relevant for TwinCAN operation.

P13_ALTSEL0
Port 13 Alternate Select Register 0 **Reset Value: 0000 0000$_H$**

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	X											
rw	rw	rw	rw	rw											

Bit P15-P12 of the Port 13 Alternate Select Register 0 must be set according **Table 4-8**.

Table 4-8 CAN I/O Line Selection and Setup

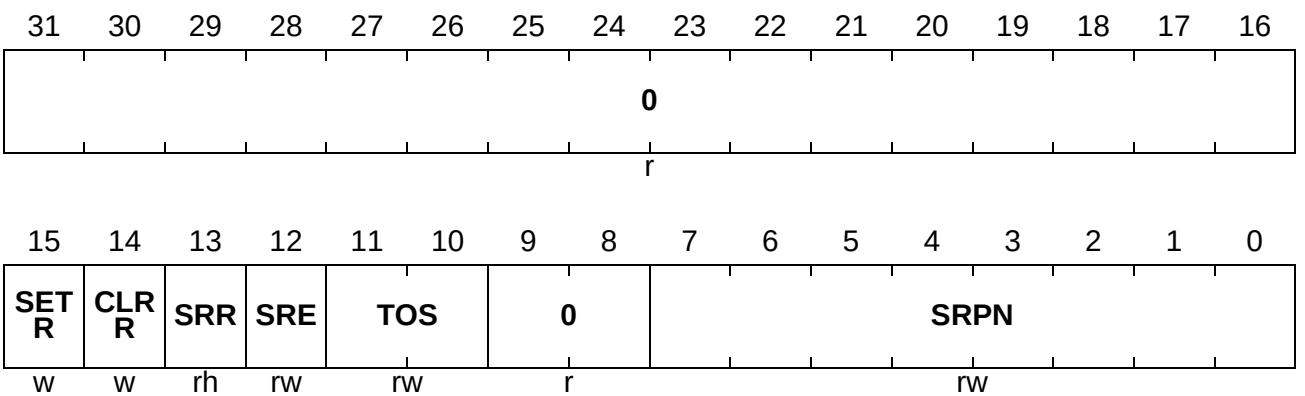
Port Line	Alternate Function	Alternate Select Register P13_ALTSEL0 Bits	I/O Port Line Operation
P13.12	RXDCAN0	P12 = 1	input
P13.13	TXDCAN0	P13 = 1	output
P13.14	RXDCAN1	P14 = 1	input
P13.15	TXDCAN1	P15 = 1	output

4.3.3.3 Service Request Control Registers

Each of the eight interrupts of the TwinCAN module are controlled by its own service request control registers.

CAN_SRC0
CAN Service Request Control Register 0
CAN_SRC1
CAN Service Request Control Register 1
CAN_SRC2
CAN Service Request Control Register 2
CAN_SRC3
CAN Service Request Control Register 3
CAN_SRC4
CAN Service Request Control Register 4
CAN_SRC5
CAN Service Request Control Register 5
CAN_SRC6
CAN Service Request Control Register 6
CAN_SRC7
CAN Service Request Control Register 7

Reset Values: 0000 0000$_H$



Field	Bits	Type	Description
SRPN	[7:0]	rw	**Service Request Priority Number**
TOS	[11:10]	rw	**Type of Service Control**
SRE	12	rw	**Service Request Enable**
SRR	13	rh	**Service Request Flag**
CLRR	14	w	**Request Clear Bit**
SETR	15	w	**Request Set Bit**

Field	Bits	Type	Description
0	[9:8], [31:16]	r	**Reserved**; returns 0 if read; should be written with 0.

For proper operation of a TwinCAN function controlled by an interrupt service routine, the following conditions should be checked:

- An interrupt request can only be serviced if the respective Service Request Enable Bit (CAN_SRC.SRE) is set to 1.
- The exact source of an interrupt request should be identified by analyzing the interrupt pending register AIR/BIR, the receive and transmit interrupt pending register RXIPND/ TXIPND, and the frame counter register AFCR/BFCR.
- The Service Request Priority Number bit field SRPN defines the sequence for the CPU arbitration in case of simultaneously set Interrupt Service Request Flags. That requires careful estimation of the TwinCAN service request priorities depending on the real time characteristic of higher prioritized interrupt sources, the CPU load, and the timing constraints to be matched by a TwinCAN interrupt service routine.

Note: Further details on interrupt handling and processing are described in the “Interrupt System” chapter of the TC1775 System Units User’s Manual.

4.3.4 TwinCAN Register Address Range

In the TC1775, the registers of the TwinCAN module are located within the following address range:

– Module Base Address. $F010\ 0000_H$
  Module End Address. $F010\ 0BFF_H$
– Absolute Register Address = Module Base Address + Offset Address
  (offset addresses see **Table 4-6**)

5 Serial Data Link Module SDLM (J1850)

This chapter describes the Serial Data Link Module (SDLM) of the TC1775. The chapter contains the following sections:

- Functional description of the SDLM Kernel (see **Section 5.1**)
- SDLM kernel register description of all SDLM Kernel specific registers (see **Section 5.2**)
- TC1775 implementation specific details and registers of the SDLM Module (port connections and control, interrupt control, address decoding, clock control, see **Section 5.3**).

*Note: The SDLM kernel register names described in **Section 5.2** will be referenced in other parts of the TC1775 User's Manual with the module name prefix "SDLM_".*

5.1 SDLM Kernel Description

Figure 5-1 provides a global view of all functional blocks of the SDLM interface.

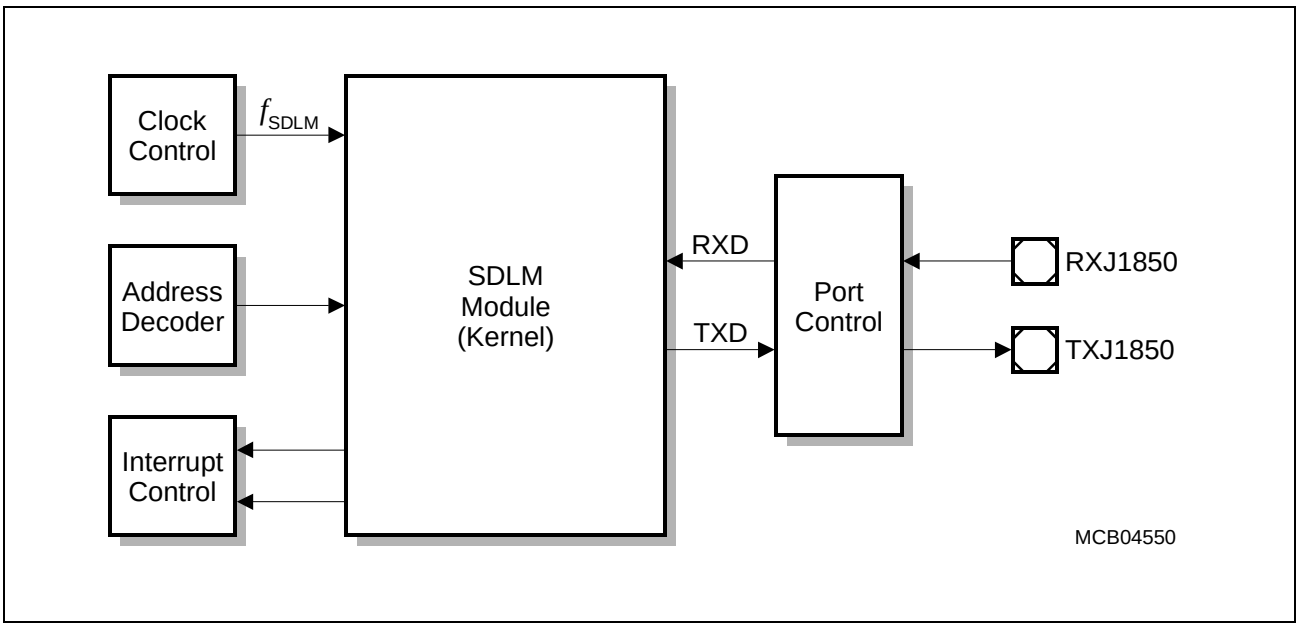


Figure 5-1 General Block Diagram of the SDLM Interface

The SDLM Module communicates with the external world via two I/O lines referred to as the J1850 bus. The RXD line is the receive data input signal and TXD is the transmit data output signal.

Clock control, address decoding, and interrupt service request control are managed outside the SDLM Module kernel.

5.1.1 Overview

The Serial Data Link Module (SDLM) provides serial communication to a J1850 based serial bus. J1850 bus transceivers must be implemented externally in a system. The SDLM Module conforms to the SAE Class B J1850 specification and is compatible with the Class 2 protocol.

In the TC1775 architecture, the SDLM is integrated as a peripheral block communicating with the TriCore CPU via the internal FPI bus.

General SDLM Features

- Compliant to SAE Class B J1850 specification
- Full support of GM Class 2 protocol
- Variable Pulse Width (VPW) format with 10.4 kBaud
- High speed receive/transmit 4x mode with 41.6 kBaud
- Digital noise filter
- Support of single byte headers or consolidated headers
- CRC generation and check
- Support of Block Mode for receive and transmit

Data Link Operation Features

- 11-byte transmit buffer
- Double buffered 11-byte receive buffer
- Support of In-Frame Response (IFR) types 1, 2, 3
- Advanced interrupt handling for RX, TX and error conditions
- All interrupt sources can be enabled/disabled individually
- Support of automatic IFR Transmission for IFR types 1 and 2 for 3-byte consolidated headers

Note: The J1850 module does not support the Pulse Width Modulation (PWM) data format.

5.1.2 SDLM States

The various operating states of SDLM are shown in **Figure 5-2**:

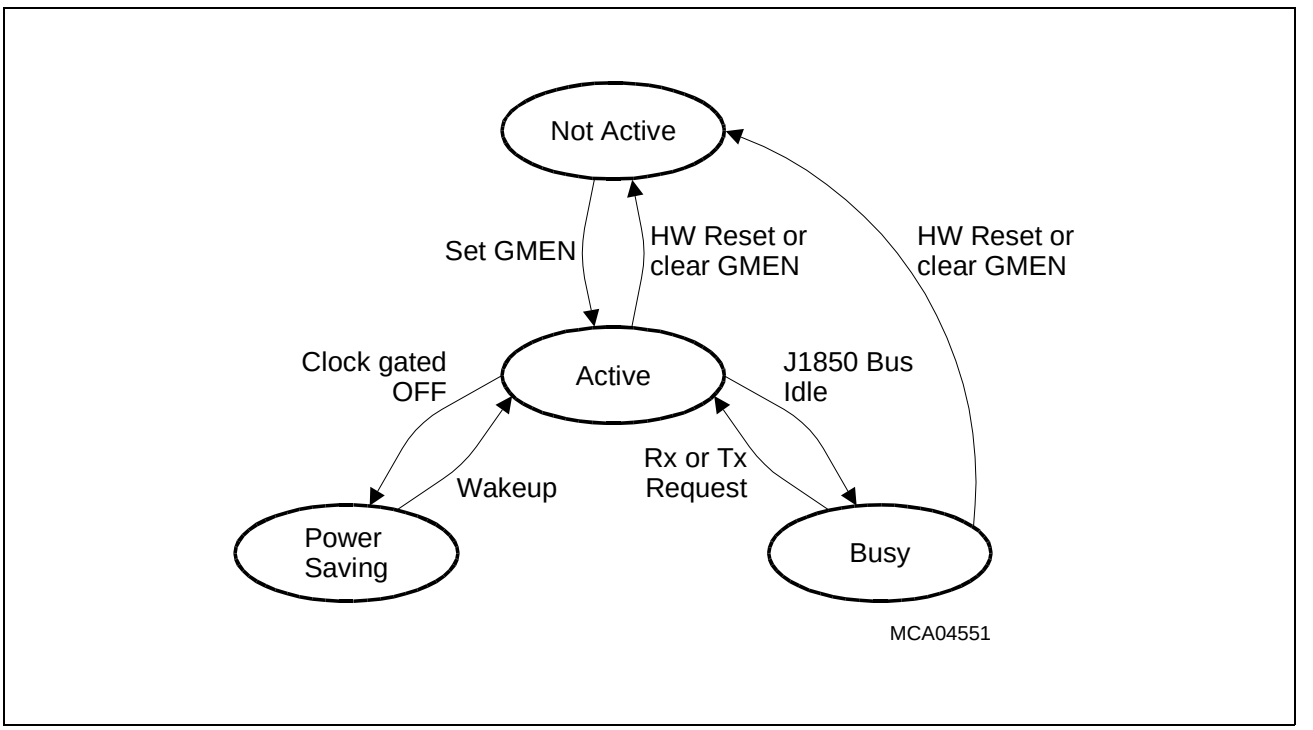


Figure 5-2 SDLM States

5.1.3 J1850 Concept

The SAE Class-B specification establishes the requirements for a serial bus protocol used in automotive and industrial applications. The specification describes the network's characteristics in three layers: physical layer, data link layer, and application layer.

The physical layer handles the frame transfer including bit/symbol encoding and timing. The data link layer defines the J1850 protocol in terms of frame elements, error detection, bus access, frame arbitration, and clock synchronization. Finally, the application layer needs to evaluate message screening/filtering by software and the handling of diagnostic parameters/codes.

J1850 is a multi-master based serial protocol. Each node has a local clock that allows simultaneous access to the J1850 bus.

5.1.4 Frame Format Basics

This section summarizes the basic definitions of the SAE Standard Class B Data Communication Network Interface protocol.

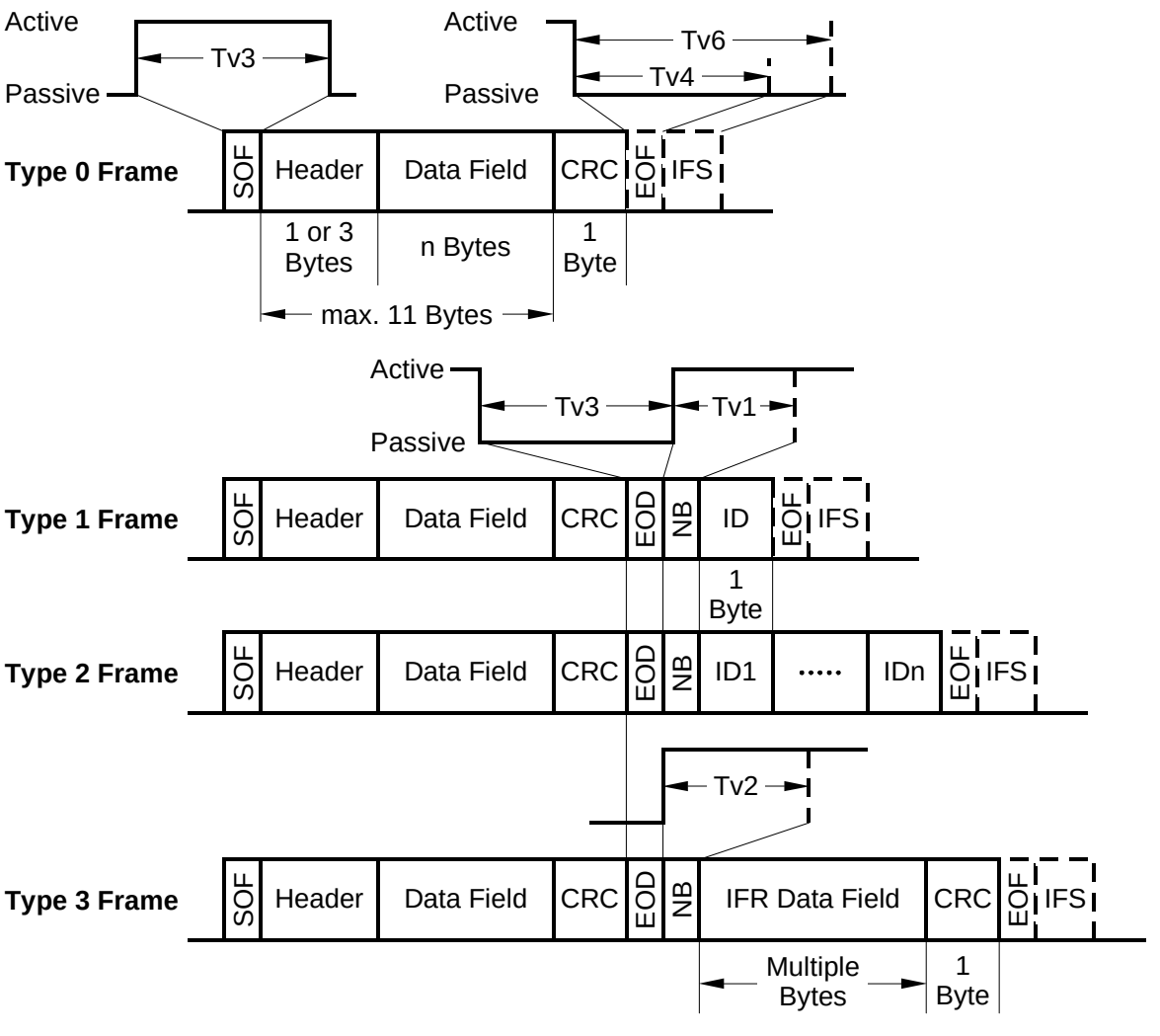
The general J1850 frame format is defined as:

– idle, SOF, DATA_0, …, Data_N, CRC, EOD, NB, IFR_1, …, IFR_N, EOF, IFS, idle

Table 5-1 J1850 Frame Format Elements

Symbol	Name	Description
SOF	Start of Frame	The SOF mark is used to uniquely identify the start of a frame. SOF is not used for CRC error calculation.
DATA_0 - DATA_N	Data bytes	Data bytes (8 bits); starting with MSB first; maximum frame length including header byte(s) but excluding frame delimiters (SOF, EOD, EOF, and IFS) and CRC is 11 bytes.
CRC	CRC byte	Cyclic redundancy check byte; generated at the transmission and checked during reception.
EOD	End of Data	Indicates the end of a transmission by the originator of a frame; Directly after EOD an IFR can be started by the recipient(s) of the frame.
NB	Normalization Bit	Required for 10.4 Kbps mode only; follows after an EOD and before an IFR symbol; defines the start of an in-frame response.
IFR_1 - IFR_N	In-frame response byte(s)	In Frame Response byte(s) (ID) can be sent by receiving devices after the sending device has sent an EOD.
EOF	End of Frame	This symbol defines the end of a frame.
IFS	Inter-Frame Separation	This symbol separates two consecutive frames.
idle	Idle state	The bus is idle if no transfer takes place (occurs before SOF or after IFS).

5.1.4.1 Frame Types



Frequency	Tv1	Tv2	Tv3	Tv4	Tv6
10.4 kbit/s	64 µs	128 µs	200 µs	280 µs	300 µs

MCA04552

Figure 5-3 J1850 Frame Formats

5.1.4.2 J1850 Bits and Symbols

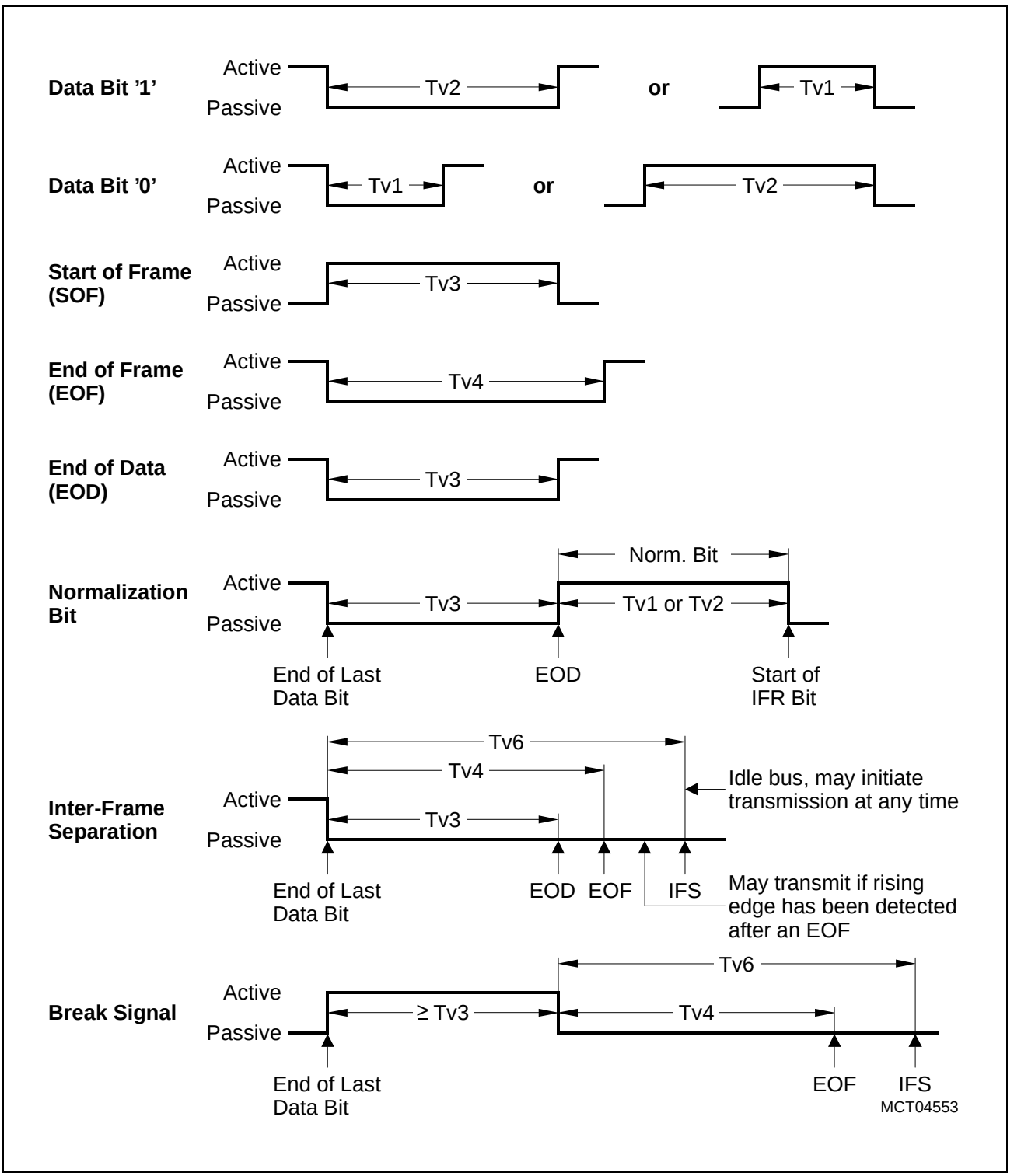


Figure 5-4 J1850 Variable Pulse Width (VPW) Format

5.1.4.3 Frame Arbitration

Frame arbitration in J1850 compatible networks follows the concept of Carrier Sense Multiple Access (CSMA) with non-destructive message arbitration. When two nodes have access to the bus at the same time, the priority decision is made during transmission. The node that has won the arbitration will continue transmission and the other node will stop transmitting. The SDLM Module always receives the current message on the bus in its receive buffer structure, even while transmitting.

The SDLM always stores the current message on the bus into its bus-side receiver buffer, even while transmitting (called self-echo message).

In the example of **Figure 5-5**, two nodes start transmitting simultaneously. The transmitted data is different in the fourth bit location. Node 1 detects that its received bus signal is different from the transmitted signal and aborts its transmission.

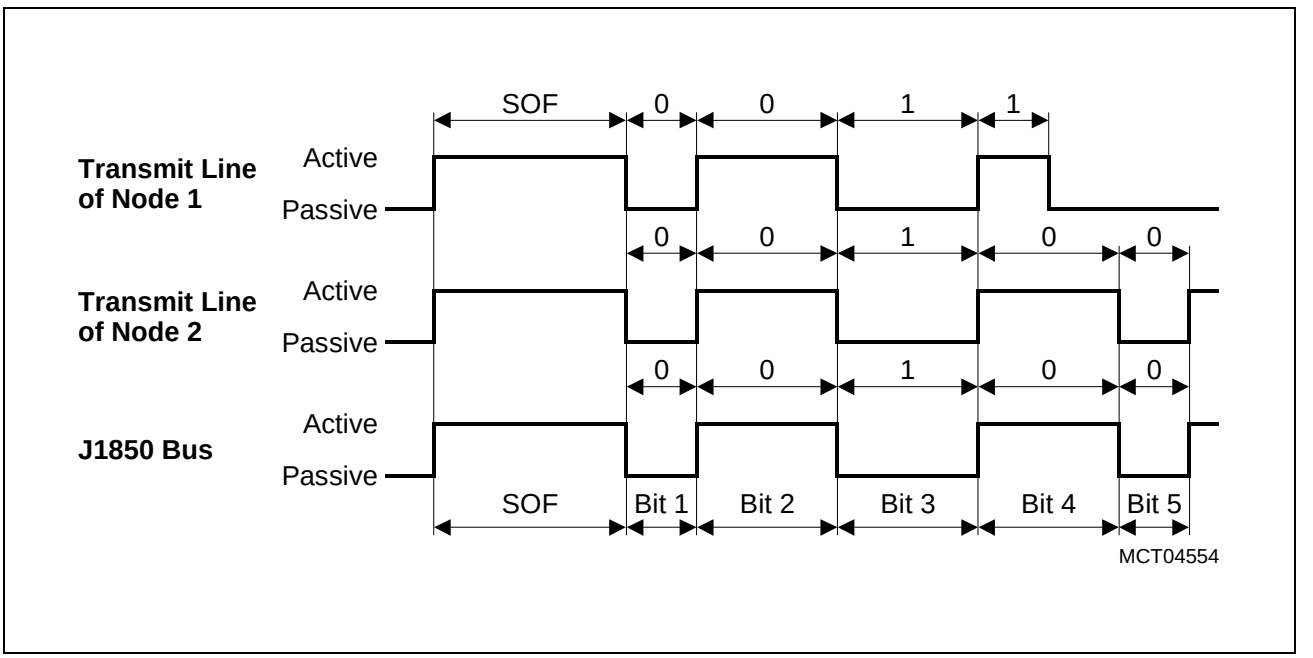


Figure 5-5 J1850 VPW Message Arbitration

5.1.5 Block Diagram

The SDLM Module is built up from two basic blocks as shown in **Figure 5-6**:

- Protocol Controller
- Data Link Controller

The Protocol Controller contains the Bit Stream Processor and the two Shift Registers for the Transmit and the Receive path. The Bit Stream Processor encodes/decodes the Variable Pulse Width (VPW) data stream and translates incoming VPW symbols into data logic levels. The Protocol Controller further has 8-bit wide data interfaces to the Data Link Controller.

The Data Link Controller handles incoming and outgoing data using three 8-bit wide data buffers, the 11-byte Transmit Buffer, and two 11-byte Receive Buffers. Several control tasks (interrupt, timing, and buffer control) are managed by the Data Link Controller.

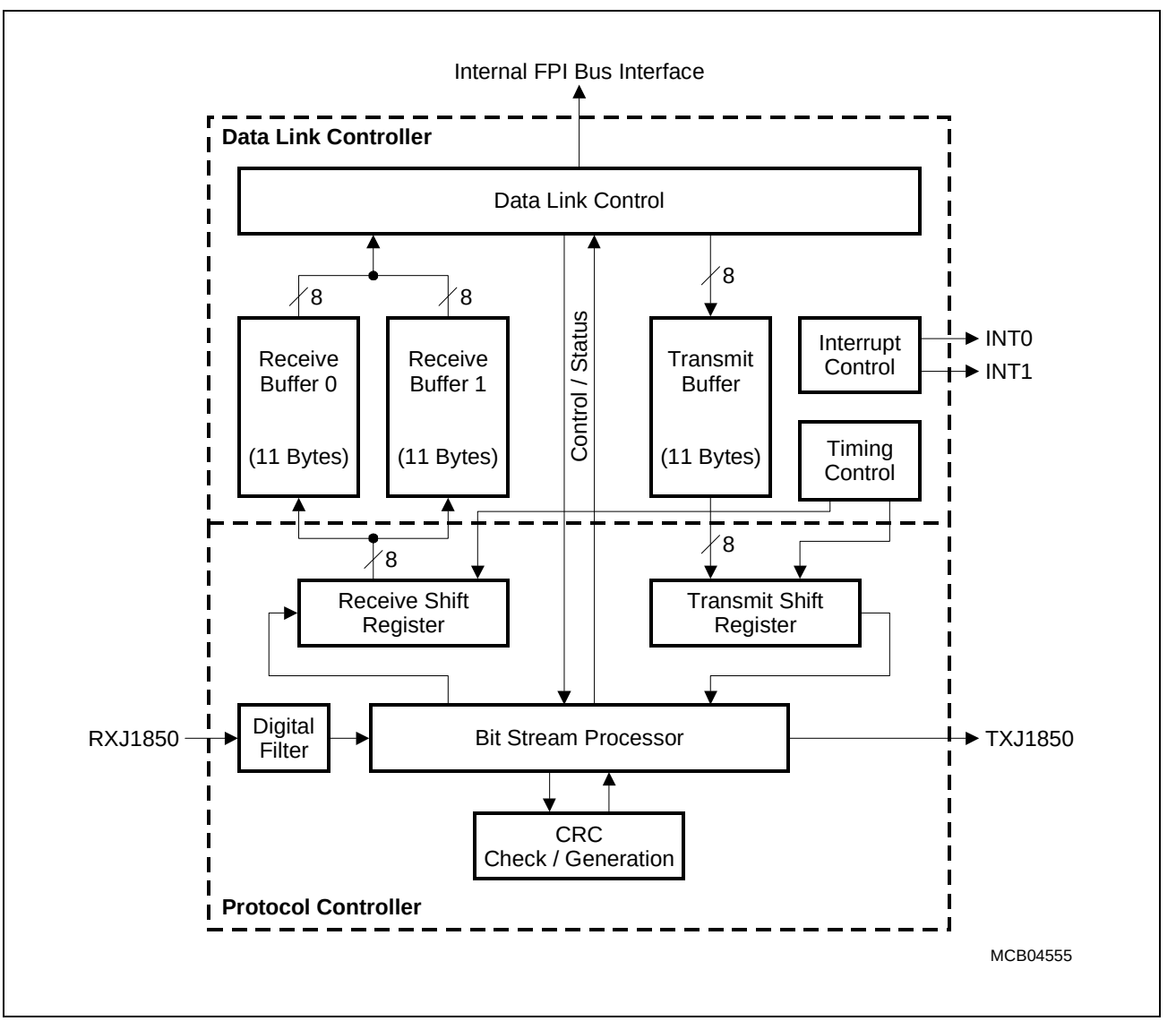


Figure 5-6 SDLM Block Diagram

5.1.6 DLL Global Configuration

The general configuration of the SDLM Module Data Link Layer (DLL) is controlled by two registers, the Global Control Register (CON), and the Timing Register (TMG). The bits within these two registers provide the following functions:

- SDLM enable/disable
- 4x Mode enable/disable
- Block Mode enable/disable
- Header type configuration (single byte or consolidated)
- Normalization bit polarity selection
- Receive buffer overwrite control
- Clock divider for module clock generation, depending on the external frequency
- Compensation of transceiver delay by SDLM
- Receive input inversion selection

5.1.6.1 4x Mode

- When high speed mode is used, all J1850 nodes should be configured to 4x mode when supported
- Nodes that do not support 4x mode need to tolerate high speed operation (no error sign)
- Enable control via bit EN4X
- IFRs are supported in 4x mode
- The transceiver delay should not exceed 4 µs to operate in 4x mode

5.1.6.2 Break Operation

- Break Symbol can occur anytime on the J1850 bus and can generate an interrupt
- Break Symbol terminates the current communication
- All nodes reset to a ‘reset-to-receive’ state (reset status bits by CPU)
- After Break symbol transition, an IFS must follow for re-synchronization
- When break is sent, the current frame is ignored (if any)
- A break transmission can be generated by setting bit BUFCON.SBRK
- Break reception is indicated by the STAT0.BREAK bit set

5.1.7 Interrupt Handling

The SDLM Module supports the generation of two interrupts grouped as:

- Protocol related interrupt conditions (INT0)
  – End of frame, break received, arbitration lost, CRC error, bus error
- Data receive/transmit interrupt conditions (INT1)
  – Message transmitted, message received, header received

The interrupt structure of the SDLM Module is shown in **Figure 5-7**. The interrupt request flags, located in the status registers STAT0 and STAT1, are set by hardware. The interrupt request flags can be reset by software via register FR but some of them are also reset by hardware. **Figure 5-7** shows the assignment of the interrupt enable flags.

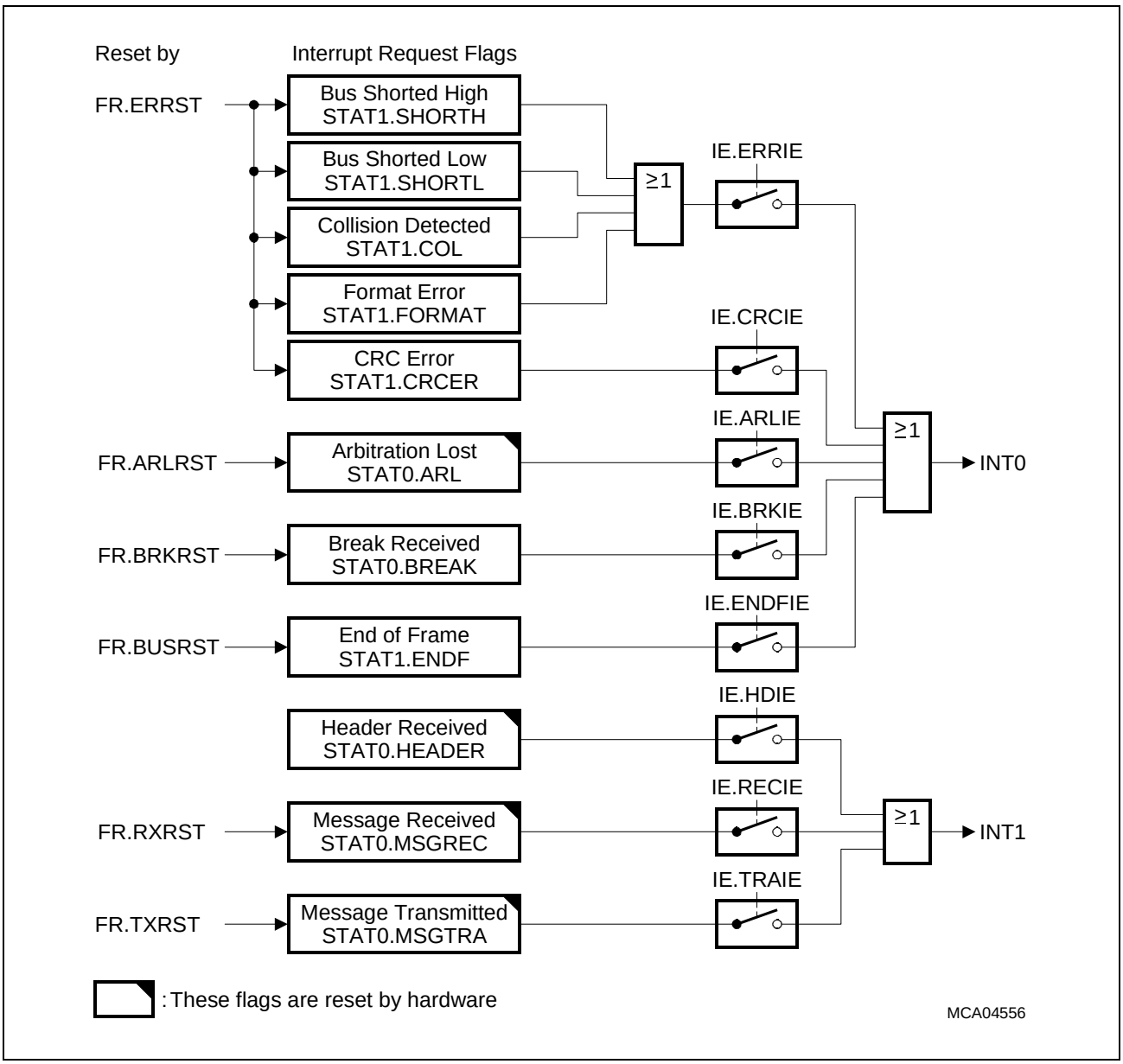


Figure 5-7 Interrupt Structure of SDLM Module

5.1.8 Message Operating Mode

Two 11-byte receive buffers and one 11-byte transmit buffer are available for data transfers. This allows the transfer of a complete J1850 frame without buffer reloading. The data buffer access can be:

- In Random Mode or additionally
- In FIFO Mode

In Random Mode, the data stored in the data buffers can be accessed in parallel either as 8-bit, 16-bit, or 32-bit wide part by their address. In FIFO Mode the data stored in the data buffers can be accessed only sequentially as single bytes (8-bit) by addressing the FIFO base.

In case of a loss of arbitration, an automatic retransmission is started, until the transmit request bit (BUFCON.TXRQ) is reset by the CPU. For correct transmission, the transmit buffer must contain valid data (TXPTR.TXCPU > 0) when BUFCON.TXRQ is set.

5.1.8.1 Receive Operation

The receive buffer structure contains two independent 11-byte receive buffers, one on CPU side and one on bus side. If the receive buffer on CPU side is full (not yet completely read out), it cannot be accessed by the J1850 module. Data reception over the bus is always done via the receive buffer on bus side. In order to release the receive buffer on CPU side, bit BUFCON.DONE must be set. After complete reception of a frame, the buffer on the J1850 (bus) side is declared full. If both buffers are full, the buffer on J1850 side can be overwritten by a new incoming frame, depending on the user-programmable overwrite enable bit (CON.OVWR). It the CPU buffer is empty and the J1850 buffer is full, both buffers are swapped automatically. By this action, the full buffer can be accessed by the CPU and the empty one is available on J1850 side.

The total number of received bytes in the corresponding buffer is indicated by register RXPTR. Register PXPTR.RXCPU indicates how many bytes have already been read out. In FIFO Mode (BUFCON.RXINCE = 1), CPU data read actions take place via registers RXD00 to RXD08 and bit field RXPTR.RXCPU is automatically incremented by 1 after each read action. In Random Mode, the buffer bytes can be directly accessed via their address. In this case, RXPTR.RXCPU is not incremented. All accesses to addresses other than RXD00 do not lead to an increment of RXPTR.RXCPU. It is only incremented after a read operation from RXD00 if RXINCE = 1.

A receive interrupt can be generated after complete reception of the whole frame (STAT0.MSGREC = 1).

5.1.8.2 Receive Control

Register BUFCON provides two bits for receive operation control:

– Receive Buffer Increment Enable (RXINCE):
  This bit enables FIFO Mode in addition to Random Mode: In Random Mode the CPU has access to each receive buffer byte via its address. In FIFO Mode, the RXPTR.RXCPU pointer is incremented upon CPU read access to RXD00 until RXPTR.RXCPU == RXPTR.RXCNT (max. 11). This mode allows easy CPU read transfer from the RXBuffer into RAM location.
– Receive Buffer on CPU Side Read Out Done (DONE)
  Bit DONE declares the receive buffer on CPU side to be empty, resets RXCPU and releases the buffer (reset of STAT0.RBC). If there is a full receive buffer on bus side, the buffers are swapped. This bit is reset after the buffer has been released.

5.1.8.3 Receive Status Information

Register STAT0 contains information about the status of the receive buffer(s):

– Reception in Progress (RIP):
  Bit RIP indicates whether the SDLM Module is currently receiving a J1850 frame.
– Receive Buffer on Bus Side Full (RBB):
  Bit RBB indicates when the receive buffer on bus side is full (not set for a self-echo message).
– Receive Buffer on CPU side Full (RBC):
  Bit RBC indicates when the receive buffer on CPU side is full (not set for a self-echo message).
– Message Lost (MSGLST):
  Bit MSGLST that a data frame has overwritten the bus side buffer or has been discarded.
– Break Received (BREAK):
  Bit BREAK is set when a break symbol has been received on the J1850 bus.

Register STAT0 contains information about the state of a receive operation:

– Header Received (HEADER):
  Bit HEADER indicates that the complete header has been received in the receive buffer on bus side.
– Message Received (MSGREC):
  Bit MSGREC indicates that a complete frame has been received.

5.1.8.4 Receive Buffer Access

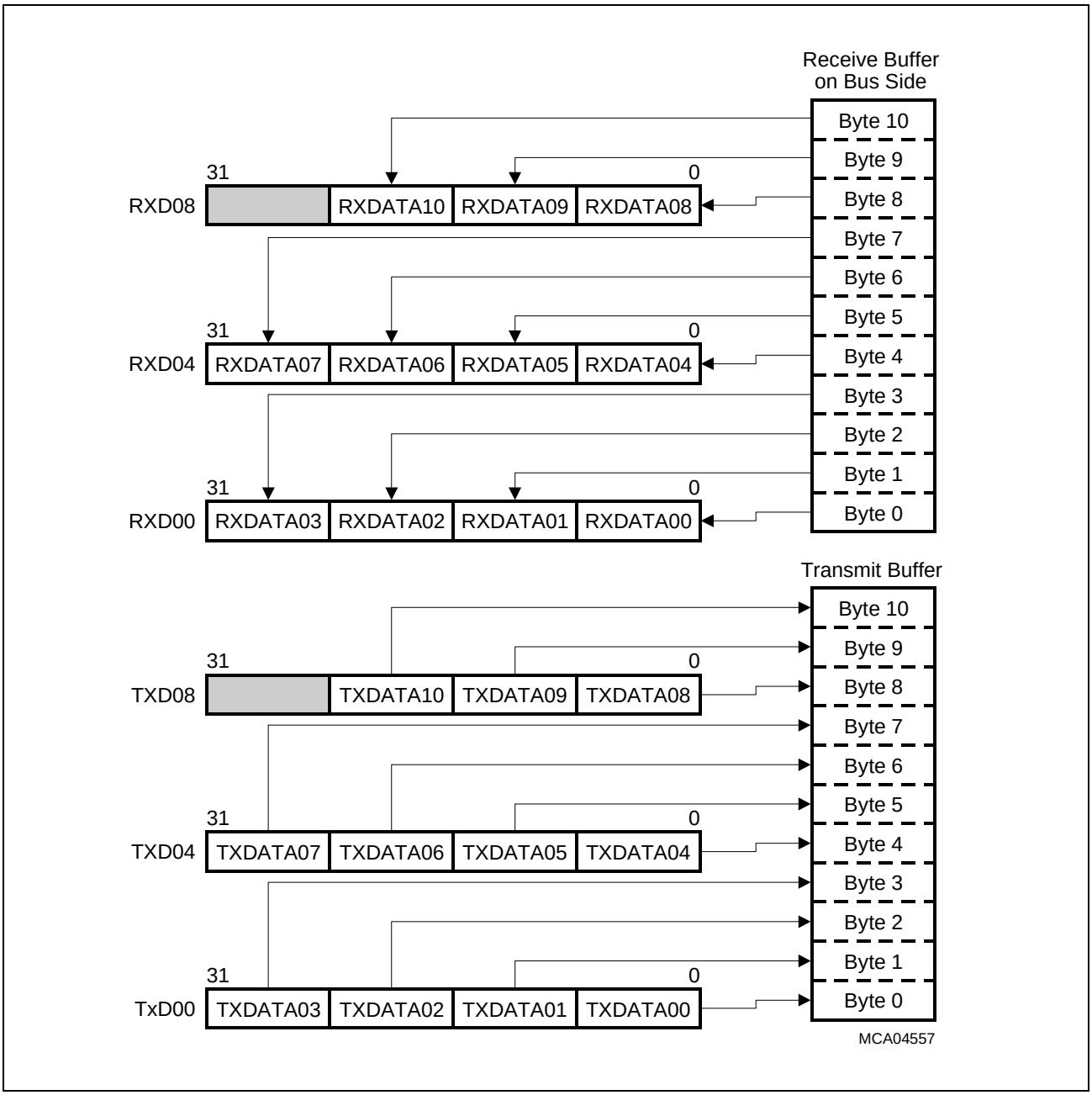


Figure 5-8 Buffer Access in Random Mode

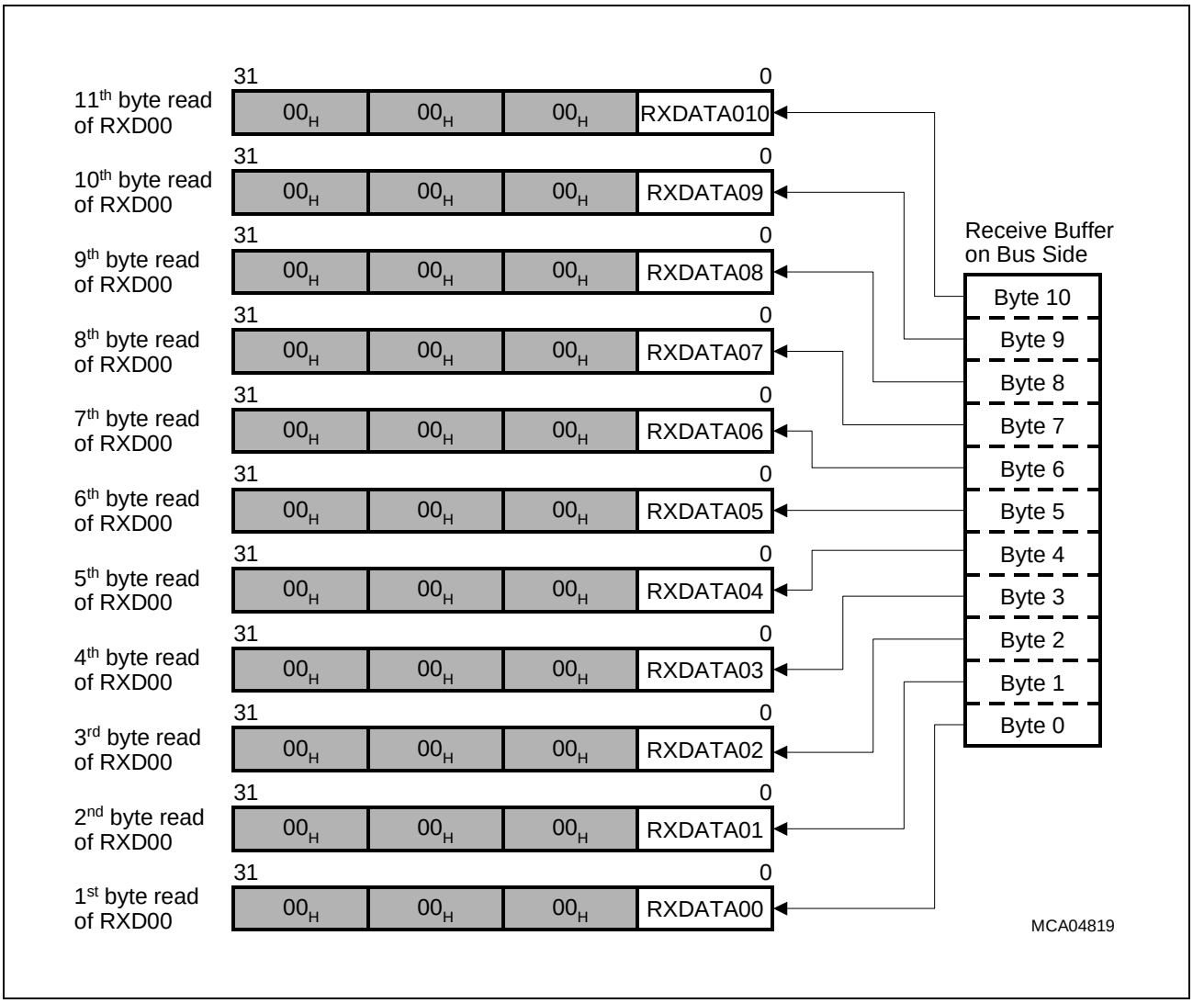


Figure 5-9 FIFO Mode: Buffer Access in Bytes

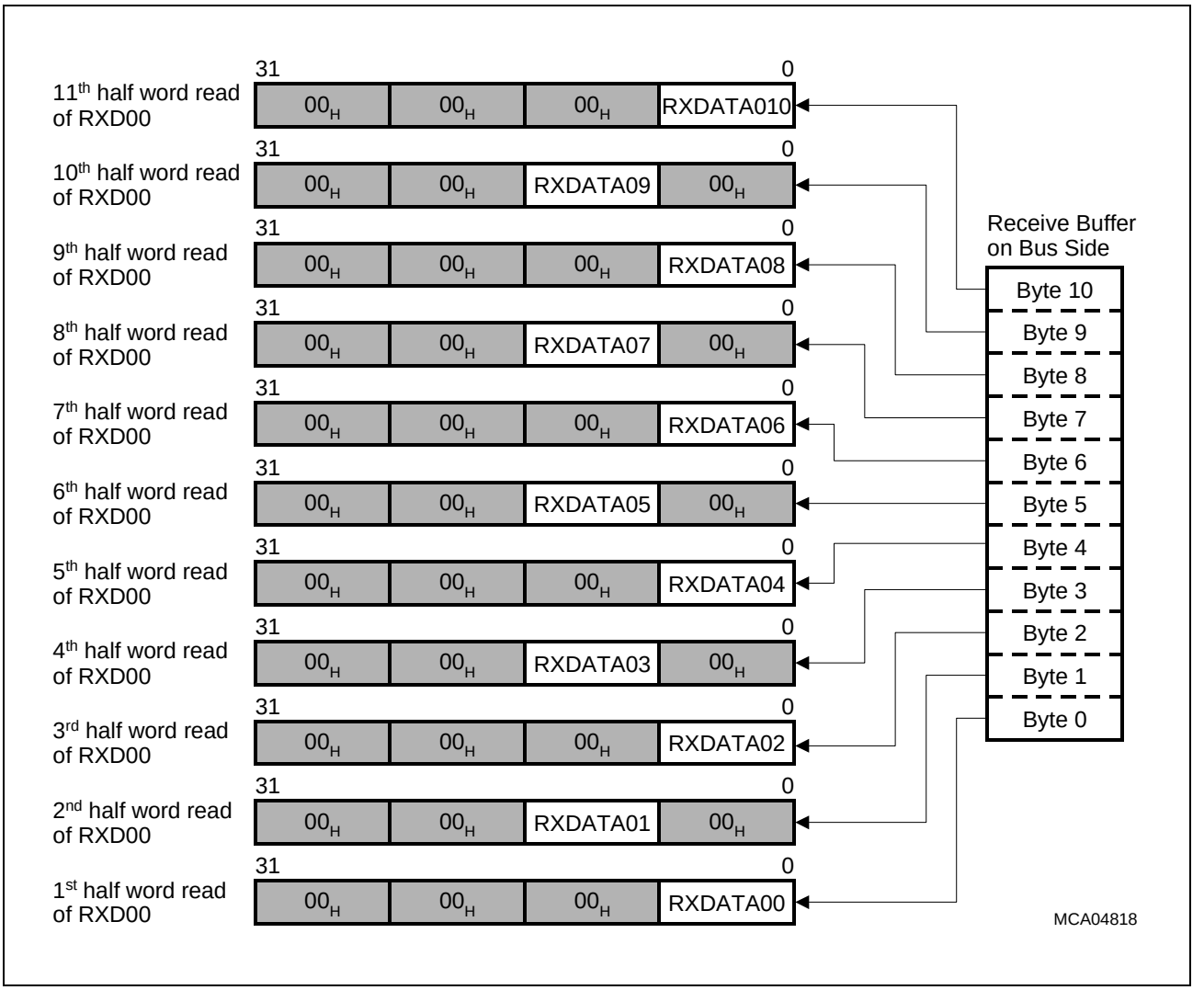


Figure 5-10 FIFO Mode: Buffer Access in Half Words

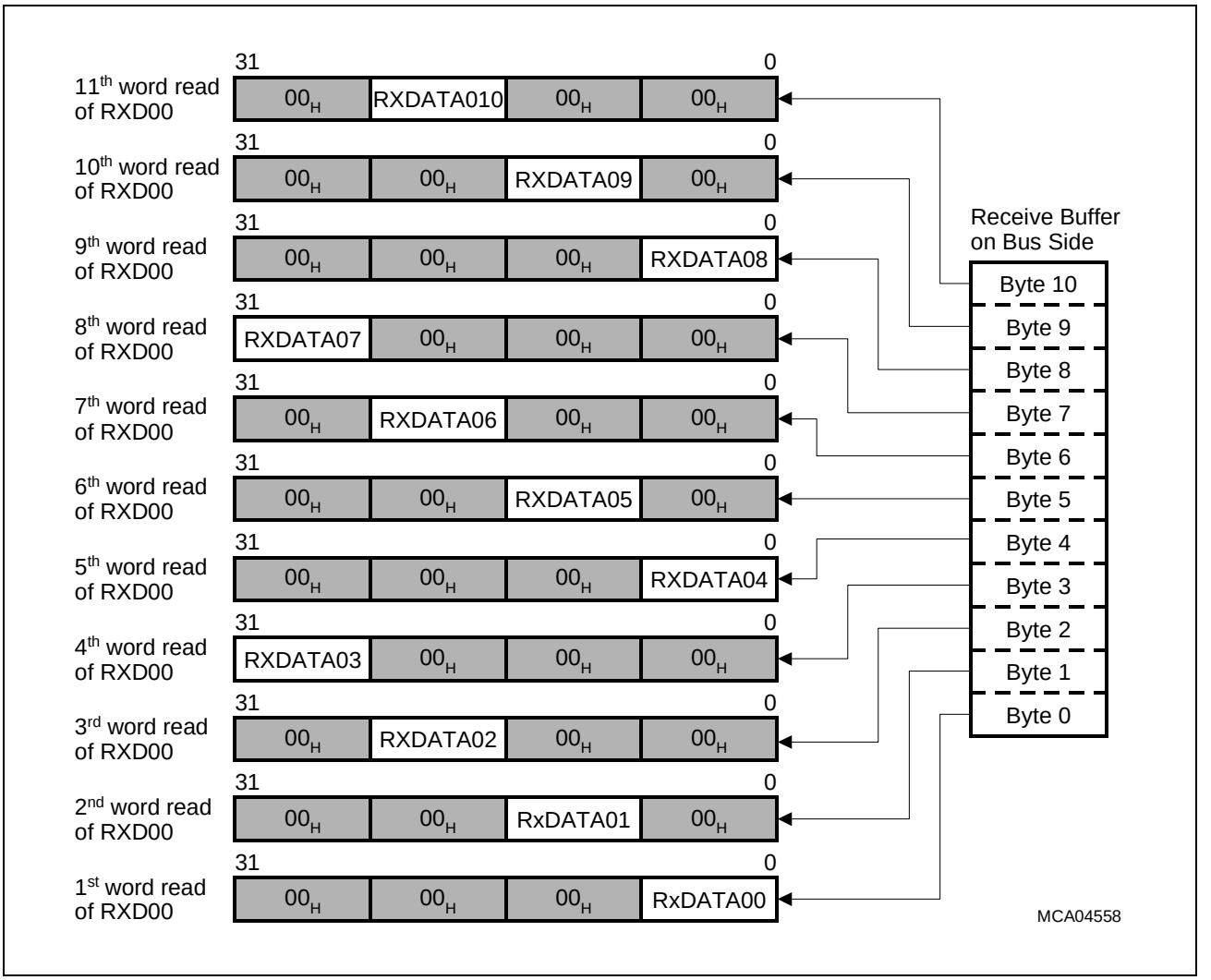


Figure 5-11 FIFO Mode: Buffer Access in Words

5.1.8.5 Transmit Operation

A data transmission is started by setting the transmission request bit BUFCON.TXRQ. Transmission is aborted by resetting bit BUFCON.TXRQ by software. FIFO Mode and Random Mode work the same way as described for data reception. A transmit interrupt can be generated after a successful transmission of the complete frame (when bit STAT0.MSGTRA is set).

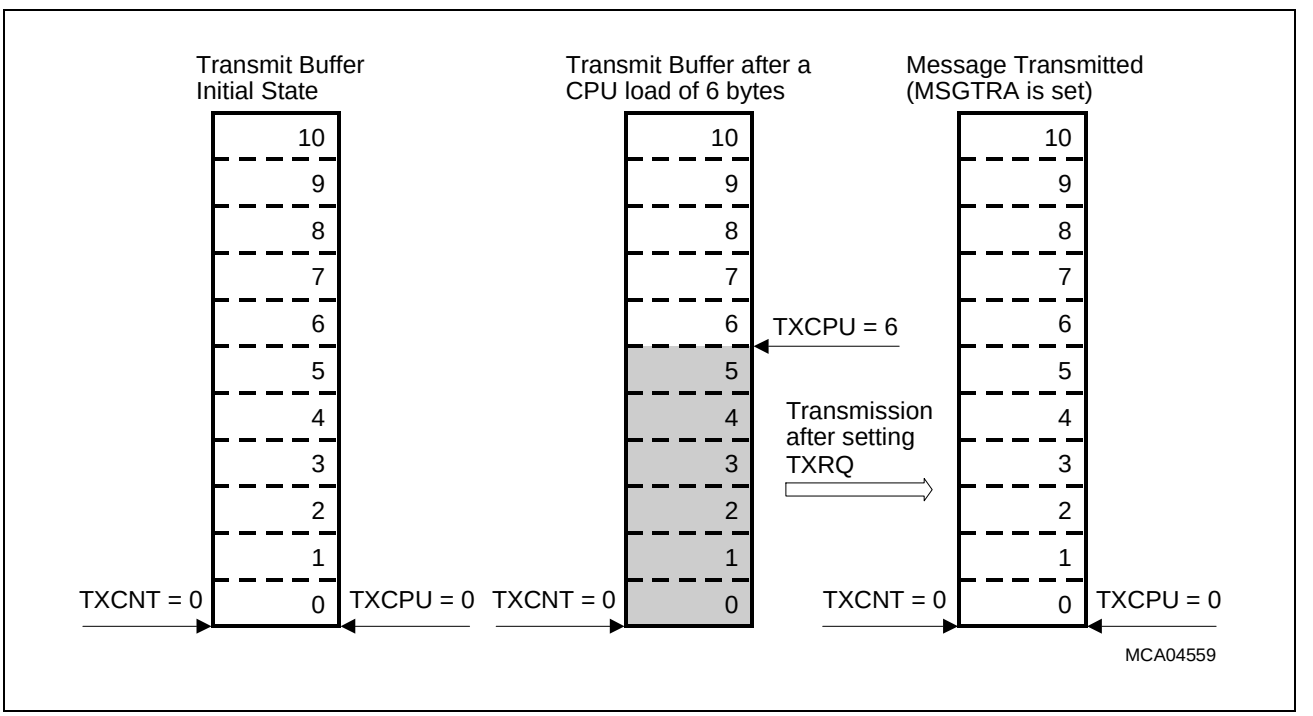


Figure 5-12 Transmit Buffer Operation

5.1.8.6 CPU Transmission Control

BUFCON provides flags controlling the TXBuffer:

- TXINCE enables (analog to RXINCE) FIFO Mode for the transmit buffer
- TXRQ initiates a Frame Transmission to the J1850 bus. In case of a lost arbitration, the module automatically retries transmission until the frame has been correctly sent out or the transmit request has been reset by software.
- CPU can initiate a break transmission by setting SBRK

5.1.8.7 Transmit Status Information

STAT0 contains information about TX and RX operations

- CPU is informed when a message is currently transmitted (Transmission in Progress - TIP)
- MSGTRA indicates a successful frame transmission
- ARL indicates that arbitration has been lost

5.1.9 In-Frame Response (IFR) Operation

The module supports automatic In-Frame Response (IFR) transmission for type 1, 2 IFR for 3-byte consolidated headers (no CPU load required). If the IFRs are handled via the transmit buffer, TXPTR.TXCPU indicates the number of bytes to be transmitted.

- If automatic IFR transmission function is not possible (single byte or 1-byte consolidated headers): If bit BUFCON.IFREN is not set, type 1 and 2 are also handled via register IFR. If BUFCON.IFREN is set, the value stored in IFR.IFRVAL will be transmitted if bit BUFCON.TXIRF is set.
- In case of automatic IFR transmission, register IFRVAL delivers the source ID. The value must be written by the CPU first.
- IFR type 3 transmission can be handled over register IFR only.
- Setting bit BUFCON.TXIFR initiates an IFR transmission (if automatic IFR not possible).
- Normalization symbol can be configured over CON.NB configuration bit.
- If IFR with CRC (Type 3, CRCEN = 1) is used, the STAT1.CRCER indicates CRC error conditions.
- If register IFR is used for transmission, no CRC will be sent out (not depending on BUFCON.CRCEN). If registers TXD0 to TXD8 are used, CRC will be sent out if bit BUFCON.CRCEN is set.
- The STAT0.HEADER bit indicates complete reception of header byte(s) in the receive buffer on bus side in order to prepare the IFR byte(s).

Transmission of type 1 and type 2 IFRs for single byte headers and 1-byte consolidated headers is also accomplished by bit BUFCON.TXIFR, which must be set by software. In case of automatic IFR (for type 1, 2 for 3-byte consolidated headers and BUFCON.IRFEN = 1), bit BUFCON.TXIFR is not needed.

3-byte consolidated headers: If BUFCON.IFREN is set, automatic response to Type 1 and Type 2 IFRs via the IFR register is enabled. Type 3 IFR is sent by writing the IFR byte(s) to the transmit buffer and then setting BUFCON.TXIFR. If BUFCON.IFREN is not set, all IFR bytes are transmitted via TXD0 to TXD8 upon setting of BUFCON.TXIFR.

Single byte headers and 1-byte consolidated headers: As there is no information in the header to indicate if an IFR is required, automatic transmission of Type 1 and 2 IFRs is not possible. If IFRs are used in the system, the header interrupt should be enabled in order to give time to decode the header in the receive buffer on bus side by software to determine if an IFR is required. IFR transmission is always initiated by setting bit BUFCON.TXIFR. Type 3 IFR is always done via TXD0 to TXD8, whereas types 1, 2 are handled via TXD0 (BUFCON.IFREN = 0) or register IFR (BUFCON.IFREN = 1).

5.1.10 Block Mode

In Block Mode, the SDLM Module supports transfers of J1850 frames of unlimited length (application specific). Block Mode is selected by setting bit CON.BMEN. In Block Mode, a 16-byte receive buffer and an 8-byte transmit buffer are available. Both buffers operate in FIFO Mode, independent of bits RXINCE or TXINCE in register CON. Because of the FIFO structure of the buffers in Block Mode, data bytes are always written to the transmit buffer via bit field TXDATA0 (register TXD0[7:0]) and data bytes are read from the receive buffer via bit field RXDATA00 (register RXD00[7:0]), as shown in **Figure 5-9** to **Figure 5-11**.

When a byte has been transmitted a transmit interrupt can be generated when TXPTR.TXCPU = TXPTR.TXCNT. After the reception of a byte, a receive interrupt can be generated if RXPTR.RXCNT0 = RXPTR.RXCPU0.

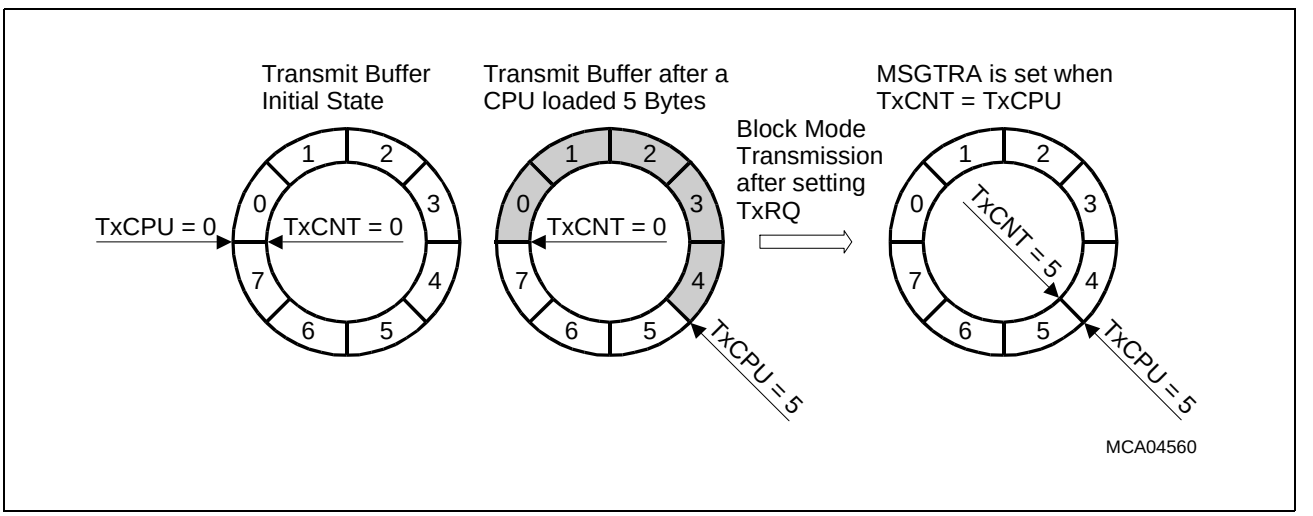


Figure 5-13 Transmit Operation in Block Mode

In order to monitor the status of the bus during transmission, the SDLM always reads on the bus, even while transmitting. As a result, the user can check whether the message sent is equal to the message on the bus (test for arbitration).

The receive buffer in Block Mode can be accessed via RXD00.RXDATA00 on CPU side at the relative FIFO base address 40_H. The elements of the FIFO can always be accessed via their addresses as well. The first eight bytes are located at the relative addresses 40_H to 47_H, the second eight bytes at the relative addresses 50_H to 57_H.

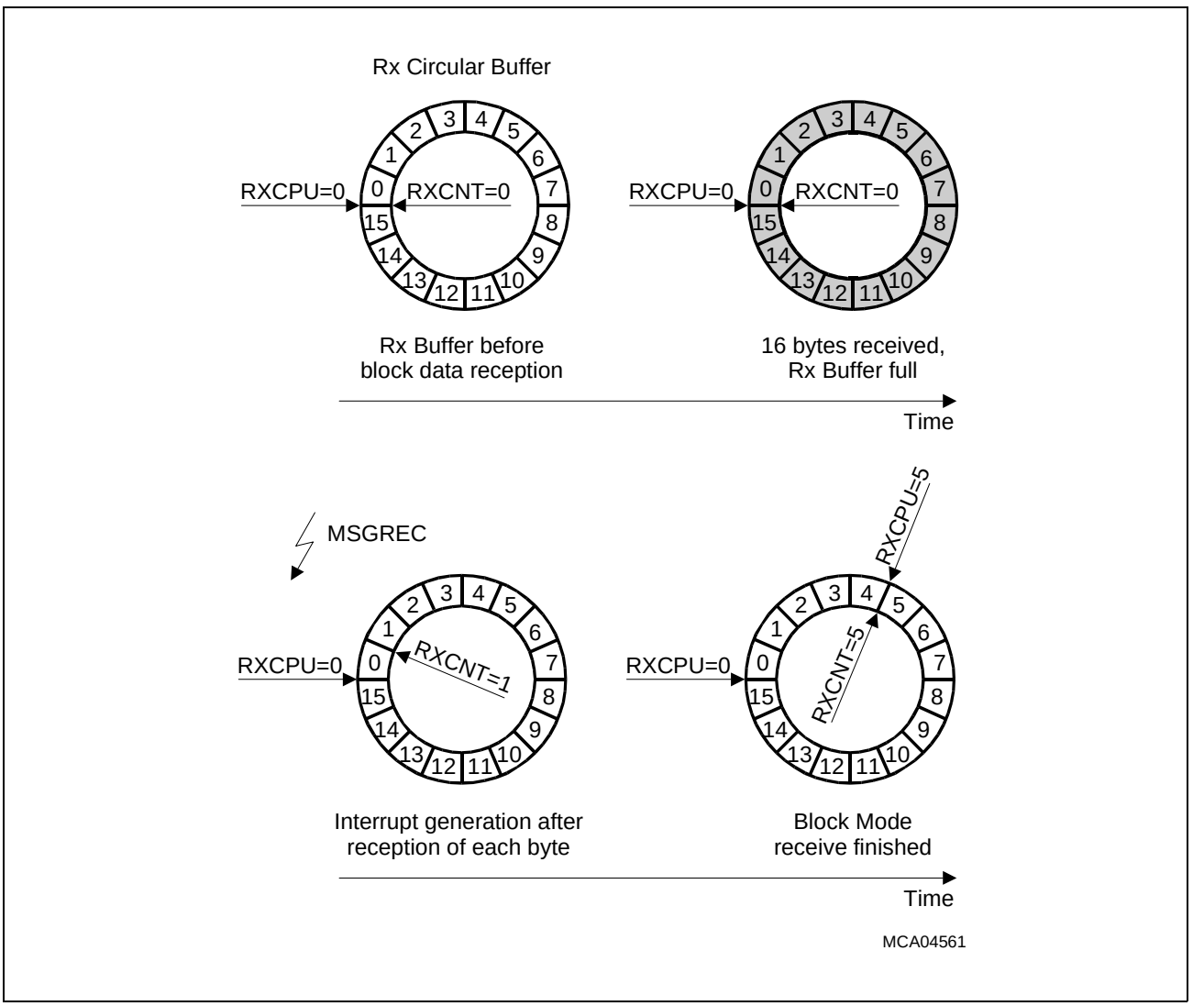


Figure 5-14 Receive Data in Block Mode

In Block Mode, the interrupt request flags MSGREC (reception) and STAT0.MSGTRA (transmission) are automatically reset by hardware upon a read action from RXDATA00, or a write action to TXDATA0 respectively. Bit STAT0.RBB (= STAT0.RBC) is set upon a pointer match after reception of a byte (receive buffer full) and reset by a read action from this buffer. STAT0.MSGLST is set if STAT0.RBB has been set previously and a new data byte is received. STAT0.MSGLST is not automatically reset by hardware. All error flags remain pending (once set) and must be cleared by software. In case an error is detected during transmission, the transmit request bit BUFCON.TXRQ is reset by hardware to abort the transmission (no automatic retry).

5.1.11 Flowcharts for Transmit Operations

5.1.11.1 Overview

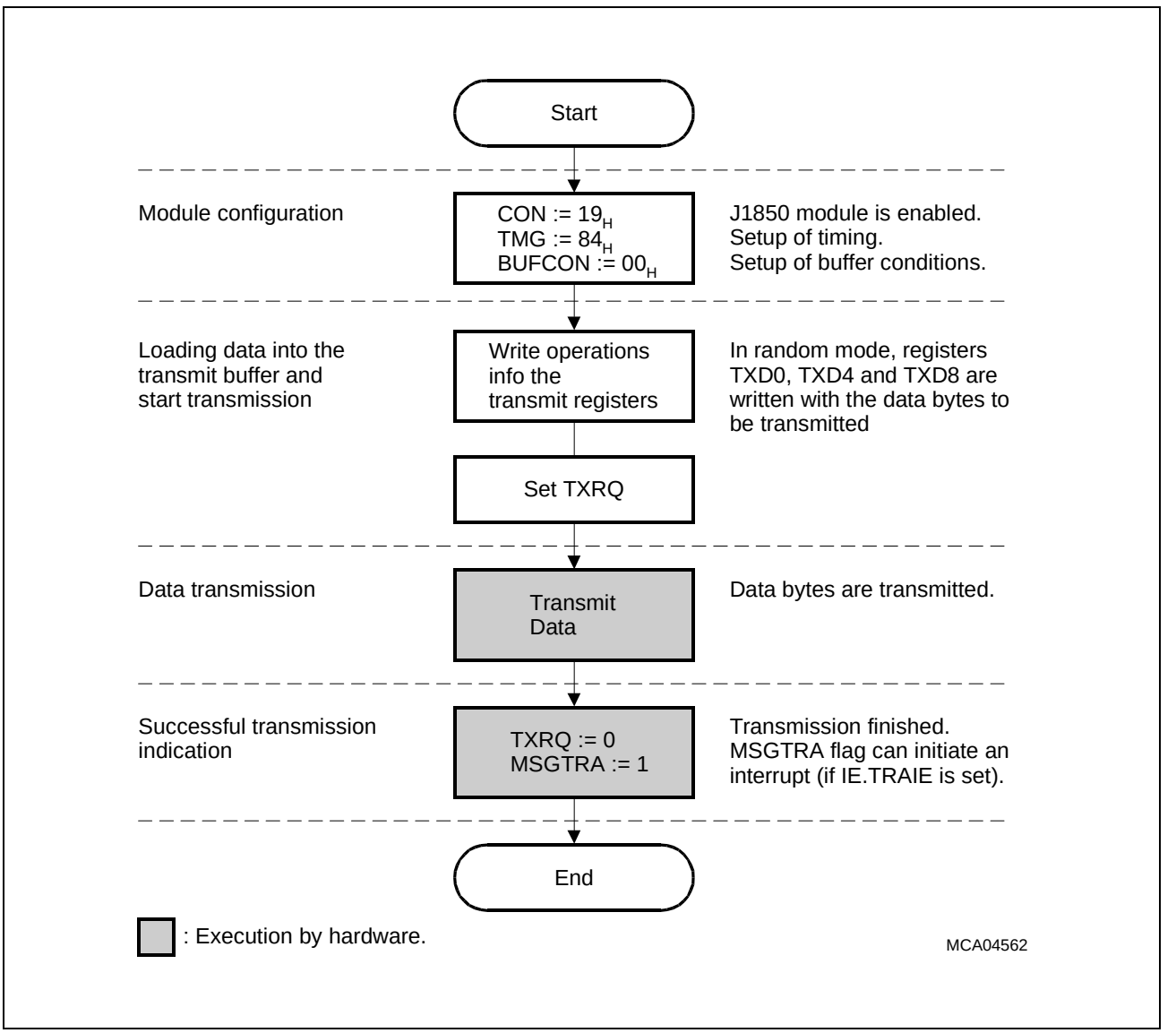


Figure 5-15 Initialization, Data Setup, and Transmission (Random Mode)

Note: To adapt the timing to the transceiver device, also bits 21-16 in register TMG must be configured.

5.1.12 Transmission of a Normal Message in FIFO Mode

Bit BUFCON.TXINCE must be set to provide FIFO functionality. Bit field TXPTR.TXCPU is incremented after each write operation to TXD0.TXDATA0. The transmit buffer is filled by multiple write actions to TXD0.TXDATA0. All other registers of the transmit buffer can always be accessed directly via their addresses without changing TXPTR.TXCPU.

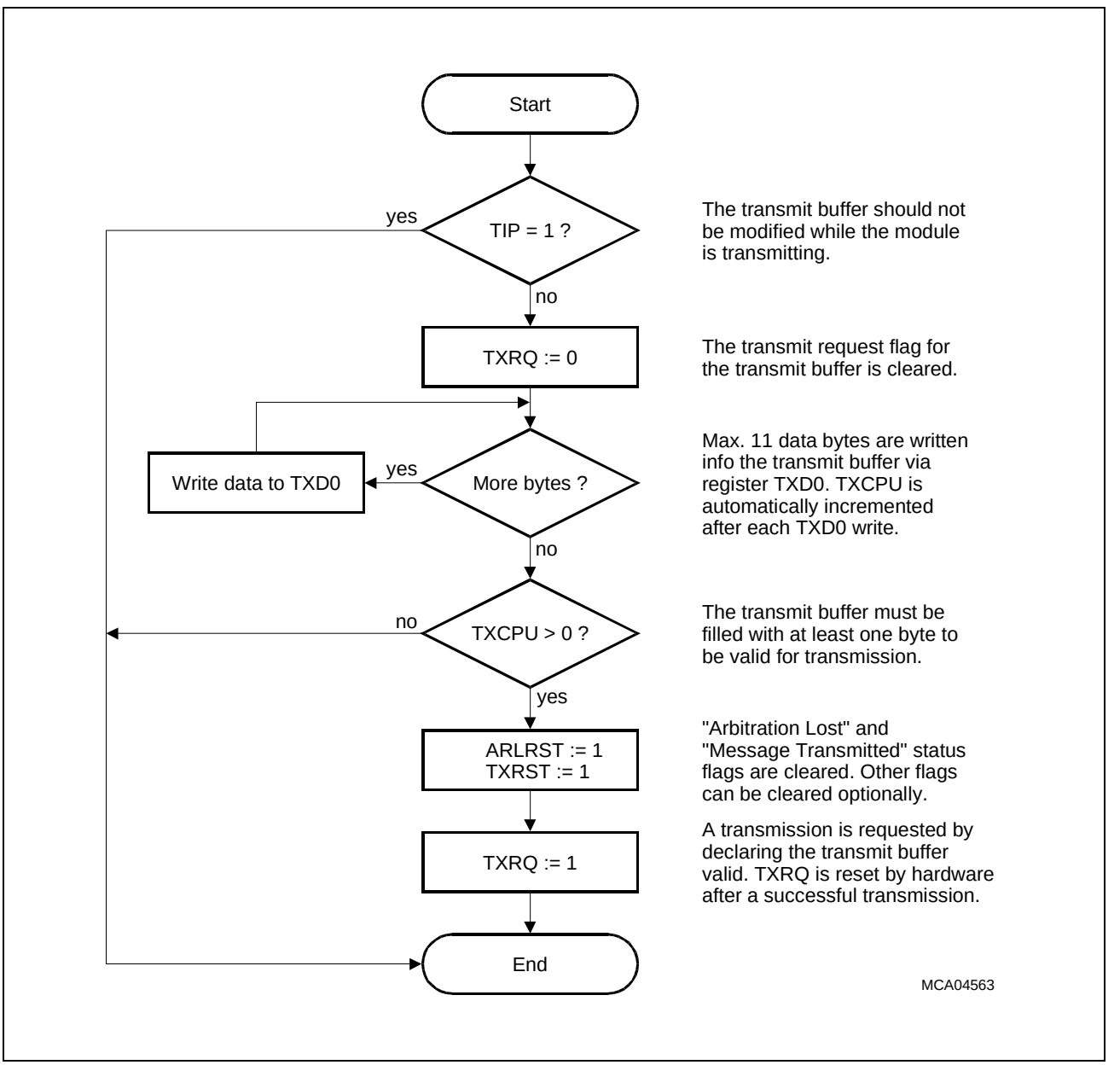


Figure 5-16 Transmission in FIFO Mode

5.1.13 Transmission of a Normal Message in Random Mode

In Random Mode, the data bytes must be written to the transmit buffer at the corresponding addresses, which must be calculated by software. Bit field TXCPU is not incremented automatically. The software must write the number of bytes to be transmitted to bit field TXCPU in order to be able to transmit the message.

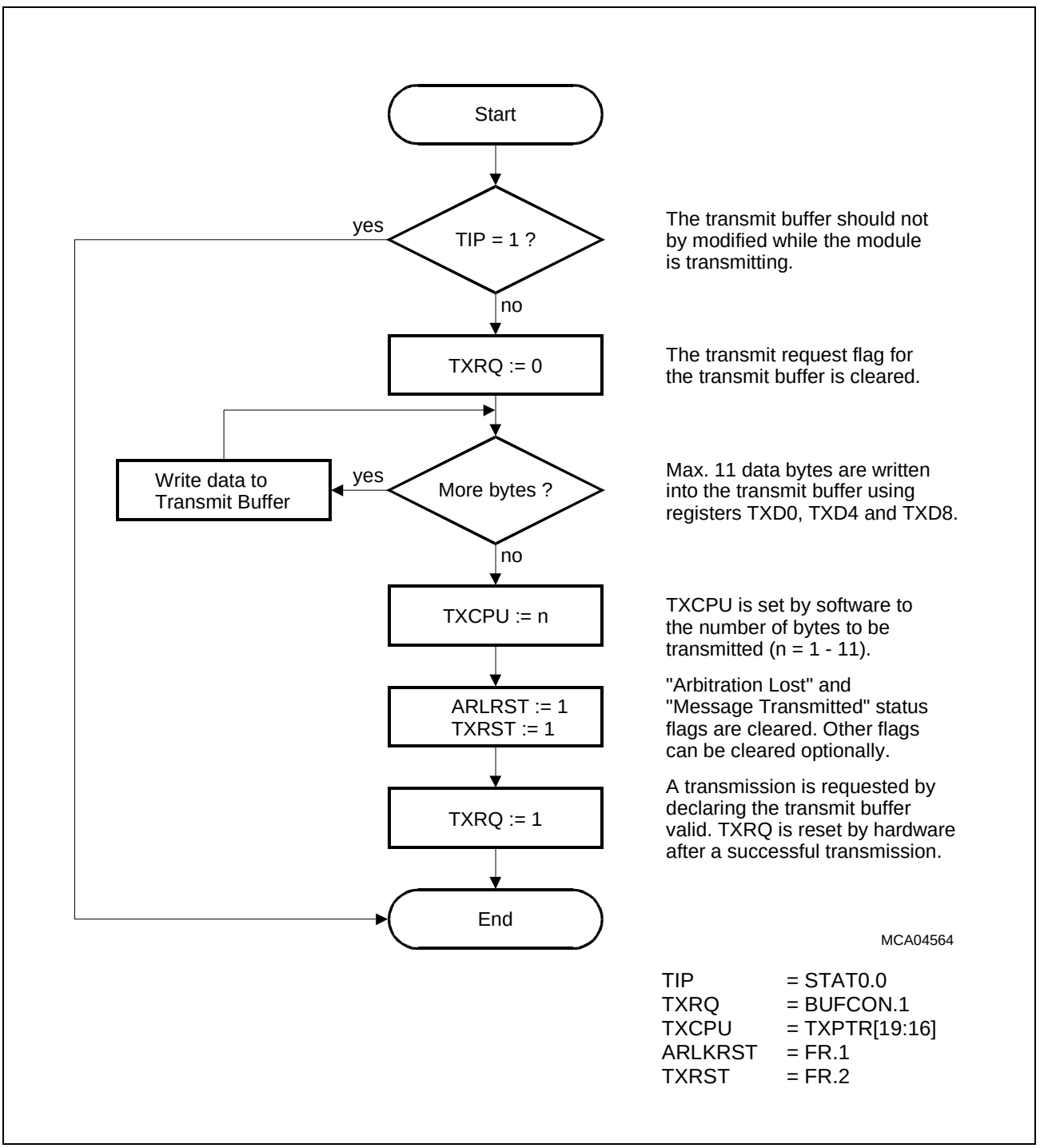


Figure 5-17 Transmission in Random Mode

5.1.14 Transmission in Block Mode

Block mode is selected by setting bit CON.BMEN. In Block Mode, FIFO Mode is always selected automatically (independent od BUFCON.TXINCE). The transmit buffer in Block Mode is 8 bytes long.

In this example, three interrupt sources (that are important for Block Mode transmission) have been enabled and need to be checked. The following flowchart represents a standard interrupt routine for this task. If only the interrupt source STAT0.MSGTRA is enabled, the data transfer can be done by the PCP.

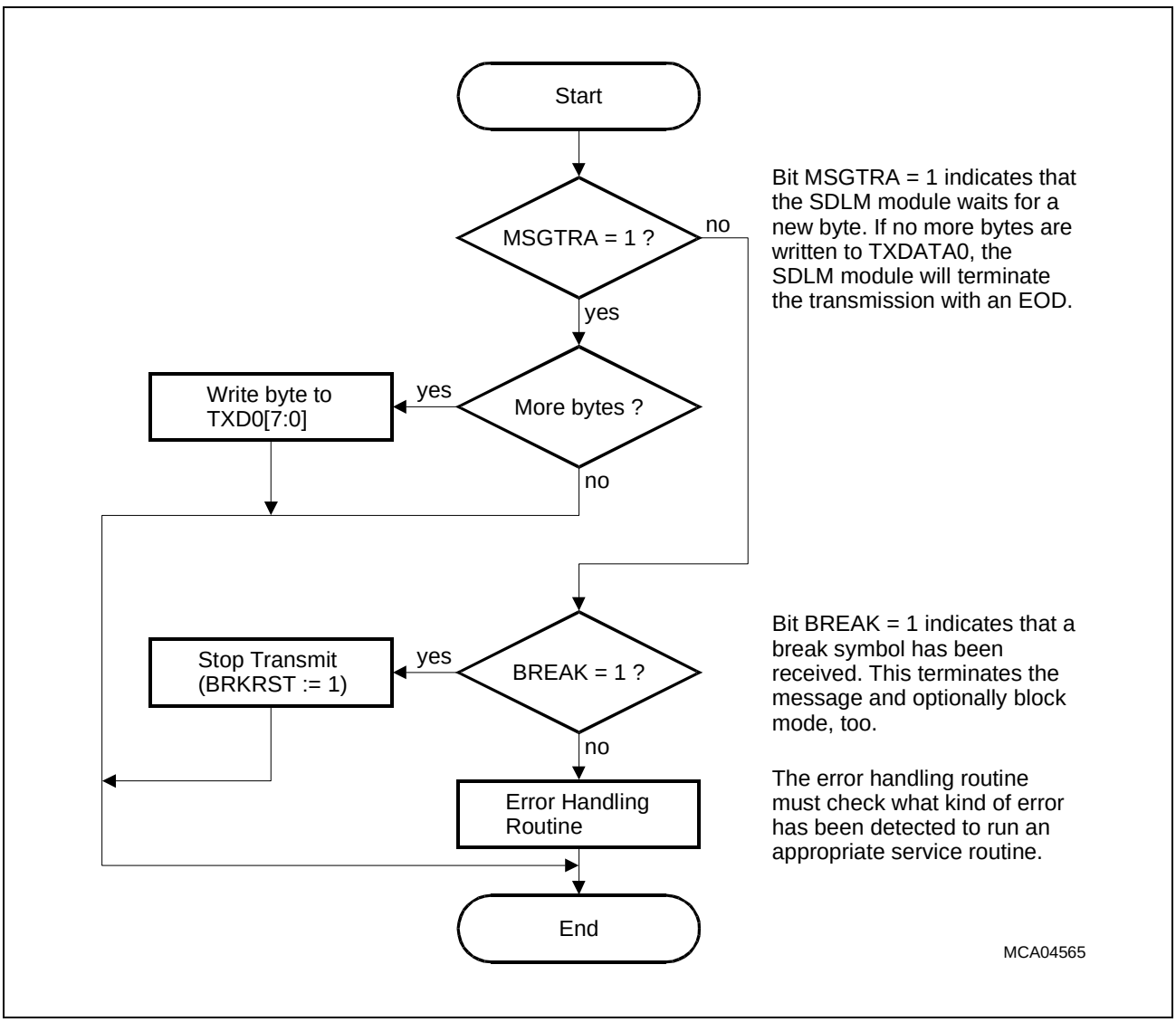


Figure 5-18 Transmission in Block Mode

5.1.15 Read Operations

5.1.15.1 Read of the Receive FIFO in Normal Mode

If bit BUFCON.RXINCE is set, FIFO Mode is enabled. Bit field RXPTR.RXCPU is incremented after each read operation from register RXD00.RXDATA00. The receive buffer is read out by the CPU using multiple read actions from RXD00.RXDATA00.

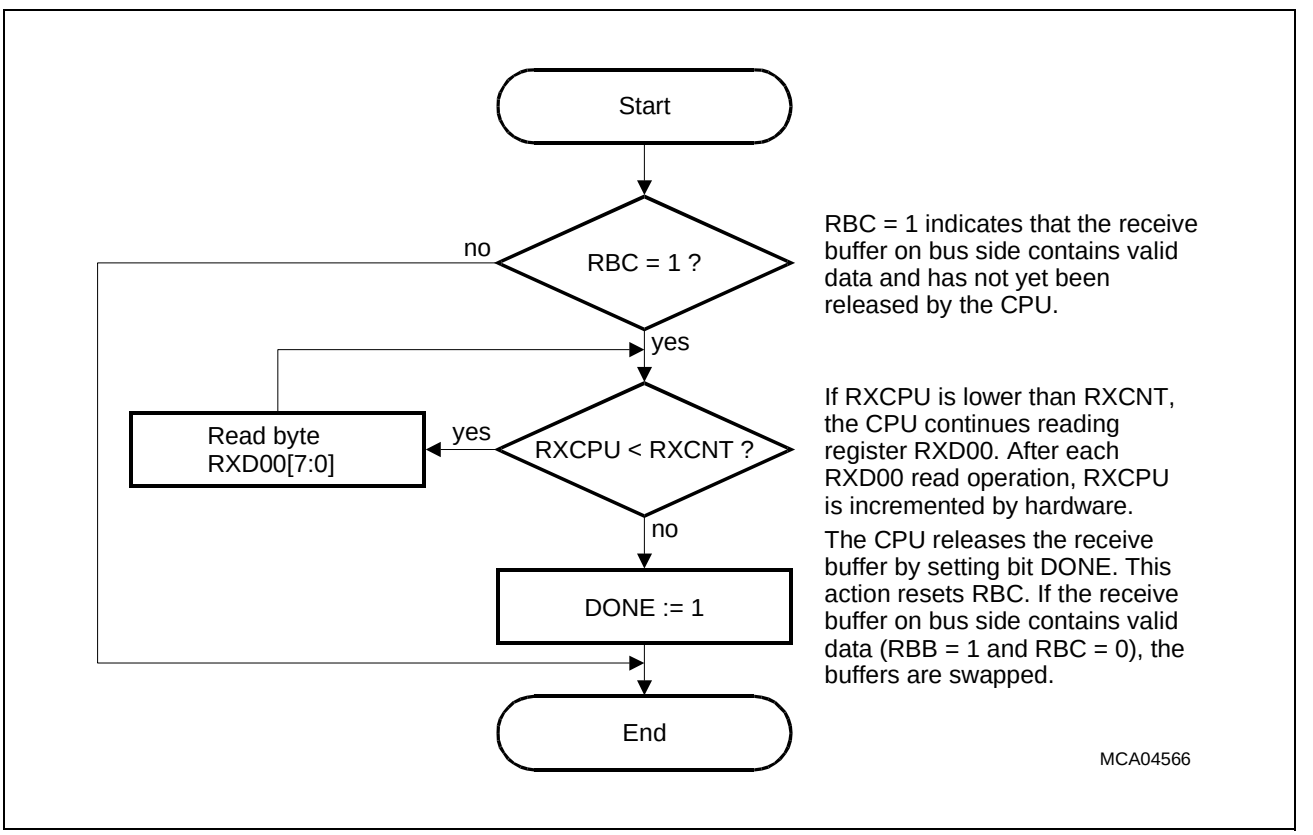


Figure 5-19 Read Operation in FIFO Mode

5.1.15.2 Read Operation in Block Mode

Block Mode is selected by setting bit CON.BMEN. In Block Mode, FIFO Mode is always selected automatically, independent of bit RXINCE. The receive buffer in Block Mode is 16 bytes long. In the example of **Figure 5-20**, four interrupt sources (that are important for Block Mode reception) have been enabled. The flowchart represents a standard interrupt routine for this task.

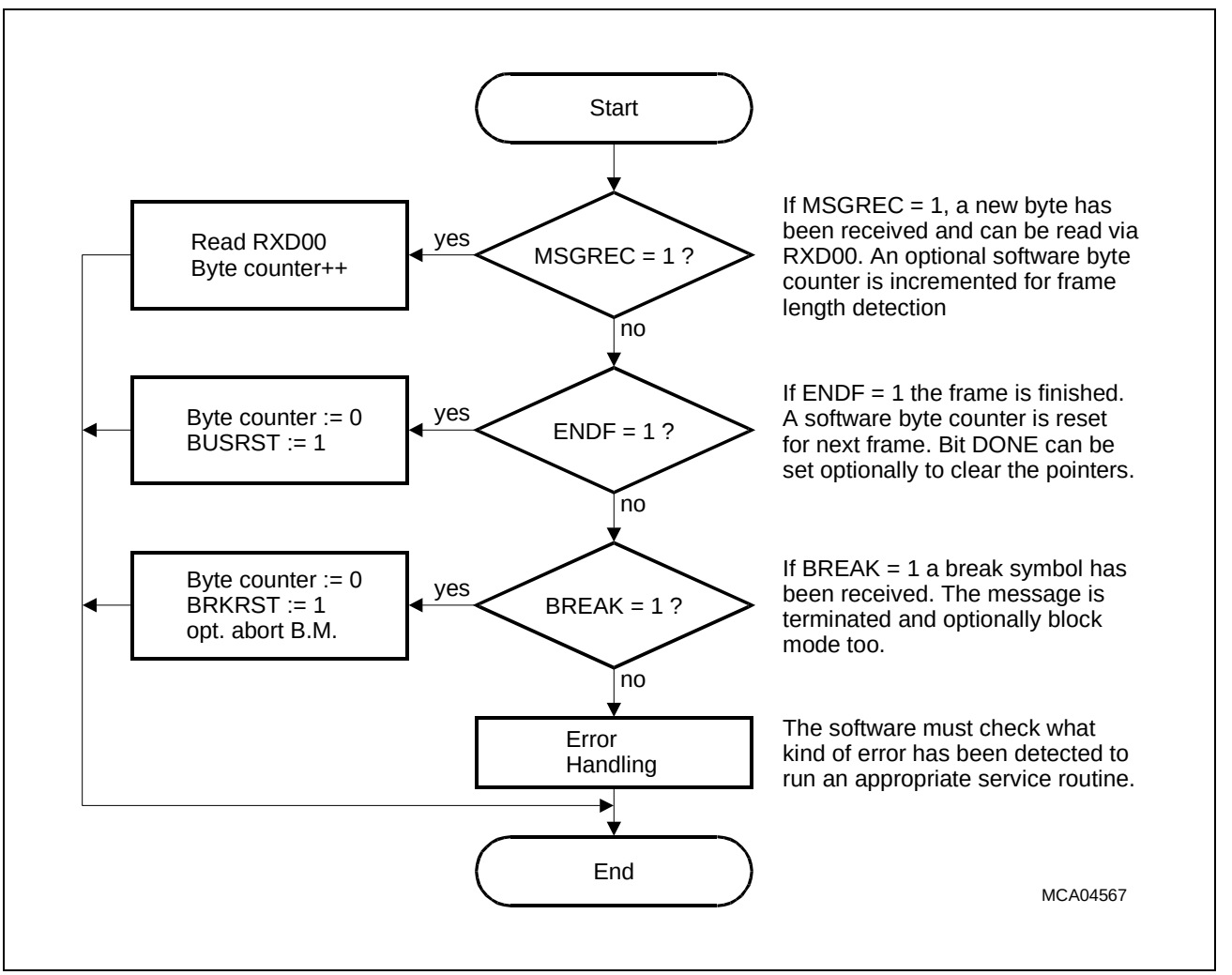


Figure 5-20 Reception in Block Mode

5.1.16 IFR Handling

5.1.16.1 IFR Types 1, 2 via IFRVAL

Bit STAT0.HEADER is automatically set by hardware after reception of the complete header (one or three bytes) in the receive buffer on bus side. This buffer can be accessed at consecutive relative addresses starting at 50_H. In case of 3-byte consolidated headers with the k bit set, an IFR (via IFR.IFRVAL) will be automatically generated if bit BUFCON.IFREN is set. Bit STAT0.HEADER is reset when FR.RXRST has been set by software or after the reception of the complete frame. In case of single byte headers or 1-byte consolidated headers, the flowchart shows a possibility to send IFR.

If an IFR is requested (automatically or by hardware), the IFR byte(s) are sent after the EOD symbol. In case of type 2 IFR, automatic retry after arbitration loss takes place.

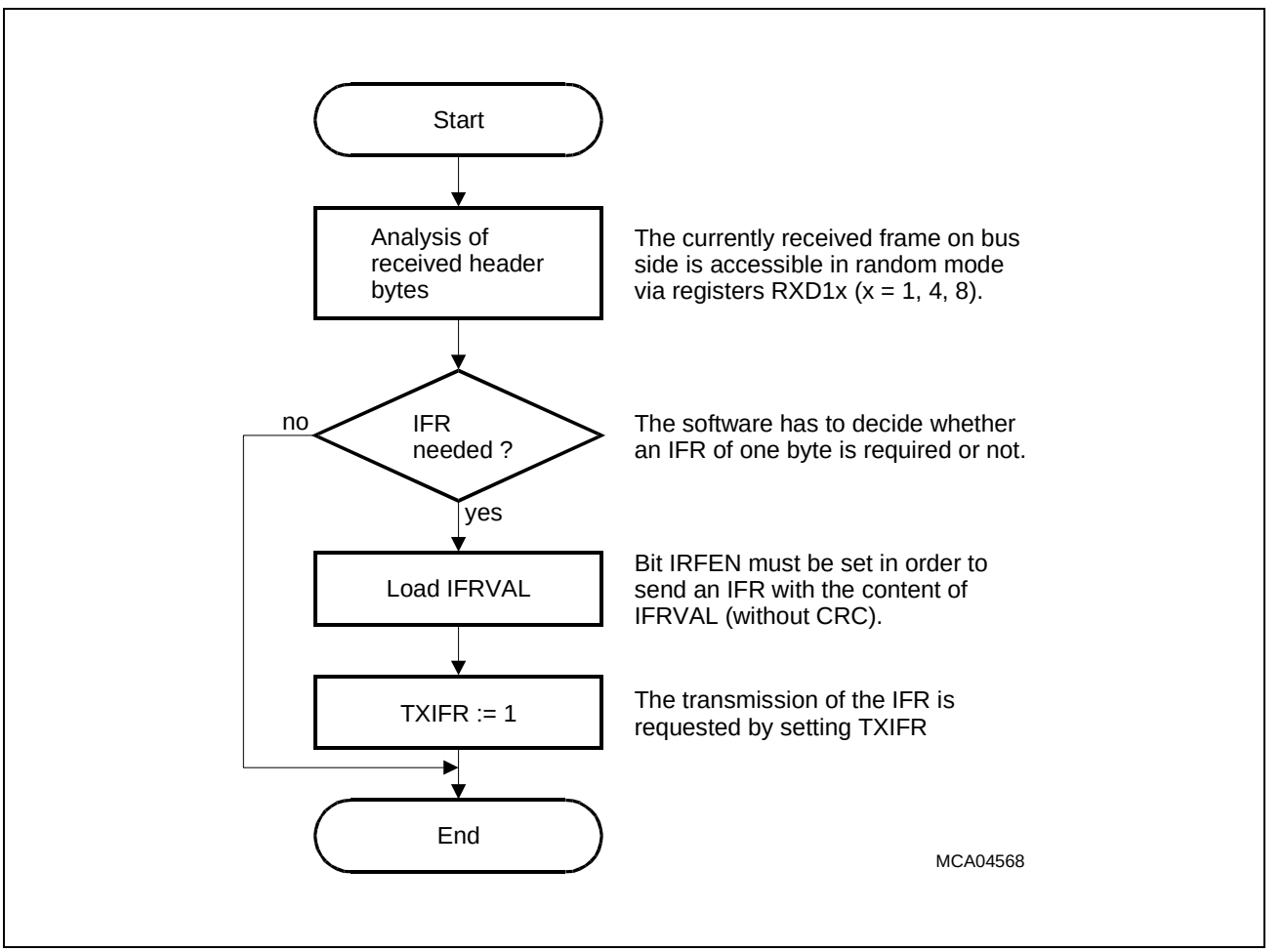


Figure 5-21 IFR Handling via IFRVAL

Note: If IFR type 3 is required, the IFR bytes must be written to the transmit buffer, bit BUFCON.IFREN must be reset and bit BUFCON.TXIFR must be set.

5.2 SDLM Kernel Registers

Figure 5-22 shows all registers associated with the SDLM Module kernel.

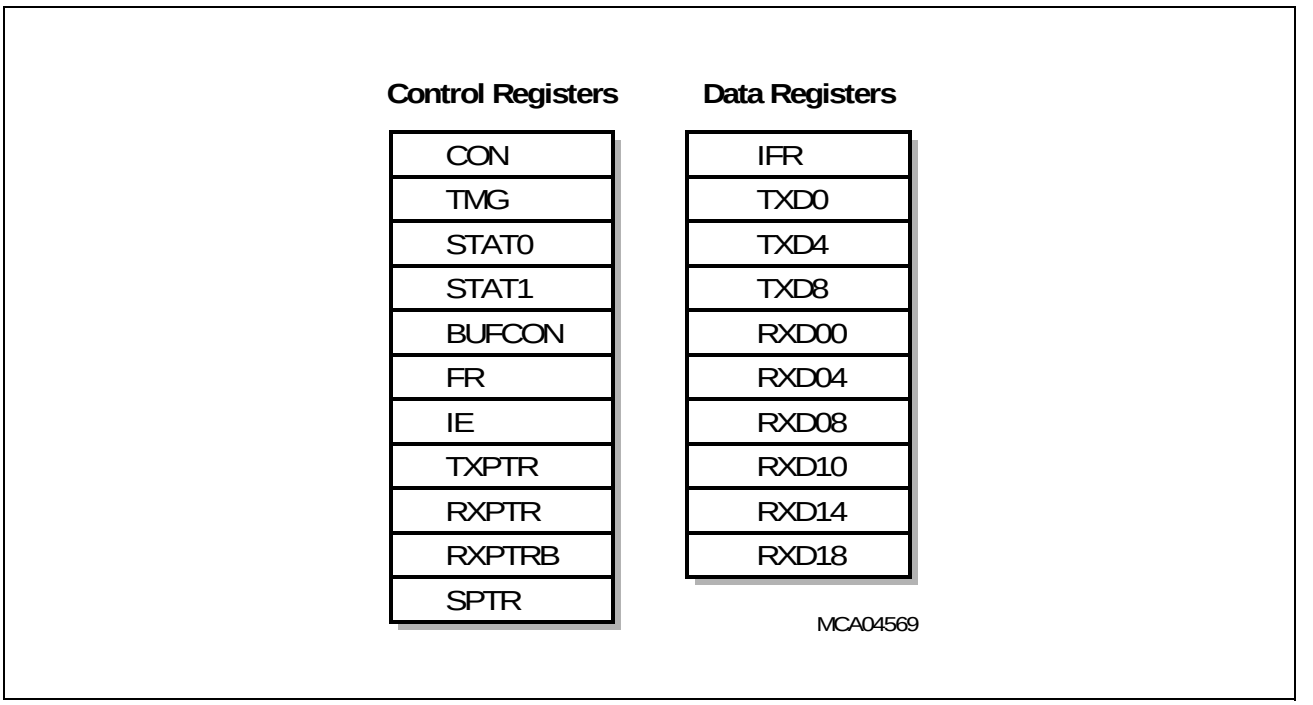


Figure 5-22 SDLM Kernel Registers

Table 5-2 SDLM Kernel Registers

Register Short Name	Register Long Name	Offset Address	Description see
CON	Global Control Register	0010_H	Page 5-31
TMG	Timing Register	0014_H	Page 5-33
IFR	In-Frame Response Value Register	0018_H	Page 5-49
STAT0	Status Register 0	$001C_H$	Page 5-34
STAT1	Status Register 1	0020_H	Page 5-37
BUFCON	Buffer Control Register	0024_H	Page 5-39
FR	Flag Reset Register	0028_H	Page 5-42
IE	Interrupt Control Register	$002C_H$	Page 5-43
TXD0	Transmit Data Register 0	0030_H	Page 5-50
TXD4	Transmit Data Register 4	0034_H	Page 5-50
TXD8	Transmit Data Register 8	0038_H	Page 5-51
TXPTR	Transmission Pointer Register	$003C_H$	Page 5-45

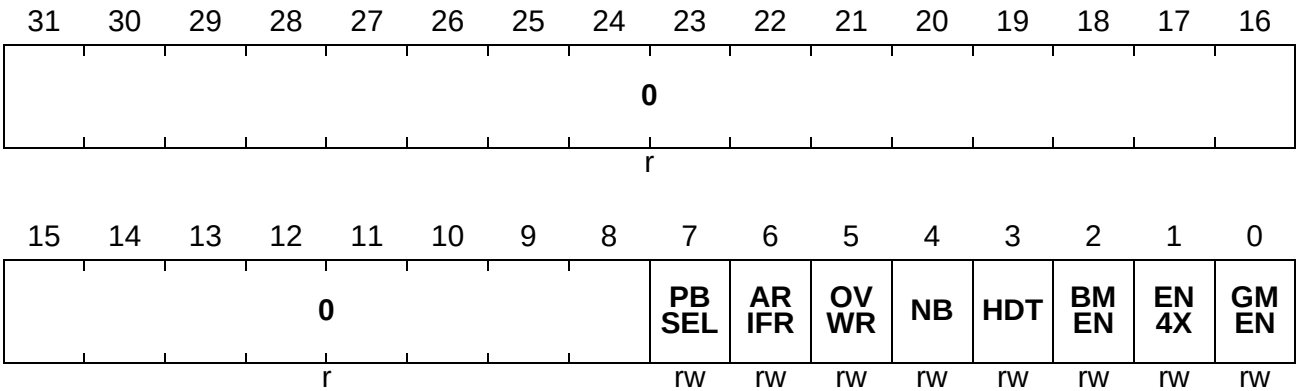
Table 5-2 SDLM Kernel Registers

Register Short Name	Register Long Name	Offset Address	Description see
RXD00	Receive Buffer 0 Data Register 0 on CPU Side	0040_H	Page 5-52
RXD04	Receive Buffer 0 Data Register 4 on CPU Side	0044_H	Page 5-53
RXD08	Receive Buffer 0 Data Register 8 on CPU Side	0048_H	Page 5-53
RXPTR	Receive Pointer Register on CPU Side	$004C_H$	Page 5-46
RXD10	Receive Buffer 1 Data Register 0 on Bus Side	0050_H	Page 5-54
RXD14	Receive Buffer 1 Data Register 4 on Bus Side	0054_H	Page 5-54
RXD18	Receive Buffer 1 Data Register 8 on Bus Side	0058_H	Page 5-55
RXPTRB	Receive Pointer Register on Bus Side	$005C_H$	Page 5-47
SPTR	Start of Frame Pointer Register	0060_H	Page 5-48

Note: All SDLM kernel register names described in this section will be referenced in other parts of the TC1775 User's Manual with the module name prefix "SDLM_".

The Global Control Register contains bits to select different transfer modes and to determine the message handling.

CON
Global Control Register **Reset Value: 0000 0000$_H$**

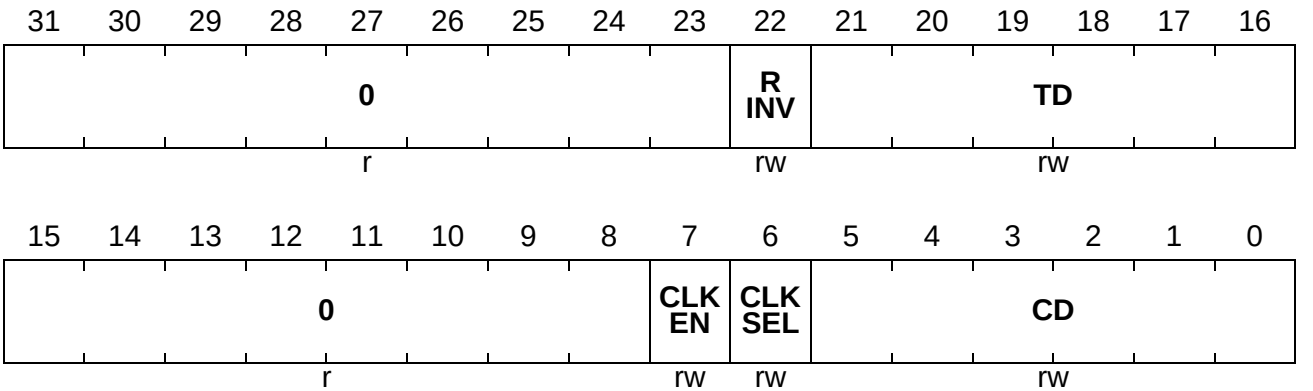


Field	Bits	Type	Description
GMEN	0	rw	**Global Module Enable** 0 The complete SDLM Module is disabled. 1 The SDLM Module is enabled and data transmission via the SDLM serial bus is possible. Resetting GMEN by software from 1 to 0 resets the module, except the timings.
EN4X	1	rw	**High Speed Transfer Enable (4x)** 0 The data transfer rate is 10.4 kbit/s. 1 The data transfer rate is 41.6 kbit/s.
BMEN	2	rw	**Block Mode Enable** 0 The maximum frame length is 11 data bytes (Normal Mode). 1 Transfers of frames longer than 11 data bytes are enabled. The transmit and the receive buffers are organized as circular buffers, that can be accessed in FIFO Mode. Resetting BMEN resets the buffer pointers.
HDT	3	rw	**Header Type** 0 Single byte headers are supported. 1 Consolidated headers (1-byte and 3-byte) are supported.

Field	Bits	Type	Description
NB	4	rw	**Normalization Bit** NB defines the polarity of the normalization bit. 0 Normalization bit is functional 0 1 Normalization bit is functional 1
OVWR	5	rw	**Overwrite Enable** 0 Overwrite action of the receive buffer on bus side in case of an incoming frame and a full receive buffer on bus side (RBB = 1) is disabled. The frame on the bus is not accepted and is lost. 1 The full buffer on bus side is declared empty and then overwritten by the next incoming frame. The formerly stored frame in the receive buffer on bus side is lost.
ARIFR	6	rw	**Automatic Retry of IFR** 0 The module will not retry transmission of IFR byte(s) in case of an arbitration loss (for IFR types 1, 3). The collision detection mechanism is generally enabled during IFR. 1 An automatic retry of IFR transmission in case of arbitration loss is enabled (handling of IFR type 2). The collision detection mechanism is disabled during EOD.
PBSEL	7	rw	**Passive Bits Select** 0 When loosing arbitration during the last bit of a byte (on a byte boundary) during the normal frame, two passive bits are automatically transmitted by the module. The passive bits are never added during IFR. 1 No extra action after the loss of arbitration (the two passive bits are not added).
0	[31:8]	r	**Reserved**; returns 0 if read; should be written with 0.

The Timing Register allows to setup the configuration of the internal timings and the transceiver characteristics.

TMG
Timing Register **Reset Value: 0014 0000$_H$**



Field	Bits	Type	Description
CD	[5:0]	rw	**Clock Divider Bits** This bit field defines the value of the clock divider to generate the module clock (either 1.00 MHz or 1.05 MHz, depending on bit CLKSEL). In order to run the module with different system frequencies, the divider can be programmed from 1 to 64. 000000_B module clock = f_{SDLM} 000001_B module clock = f_{SDLM} / 2 … 111111_B module clock = f_{SDLM} / 64
CLKSEL	6	rw	**Clock Select** 0 1.00 MHz module clock 1 1.05 MHz module clock
CLKEN	7	rw	**Clock Enable** 0 Module clock is gated off (protocol controller not clocked) in order to reduce power consumption. 1 Module is clocked, protocol controller working.
TD	[21:16]	rw	**Transceiver Delay** This bit field defines a transceiver delay that is taken into account by the J1850 bit stream processor. TD defines the number of module clock cycles. The reset value equals 20 µs @ 1.00 MHz module clock or 19 µs @ 1.05 MHz module clock.

Field	Bits	Type	Description
RINV	22	rw	**Receive Input Invert Control** 0 Receive input polarity is not inverted 1 Receive input polarity is inverted
0	[15:8], [31:23]	r	**Reserved**; returns 0 if read; should be written with 0.

Status Register 0 contains transmission-related status flags and monitors three functional bits of the header of the currently received frame.

STAT0
Status Register 0 **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								Y	K	H	ARL	BR EAK	HEA DER	MSG REC	MSG TRA
r								rh	rh	rh	rh	rh	rh	rh	rh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0											RBC	RBB	MSG LST	RIP	TIP
r											rh	rh	rh	rh	rh

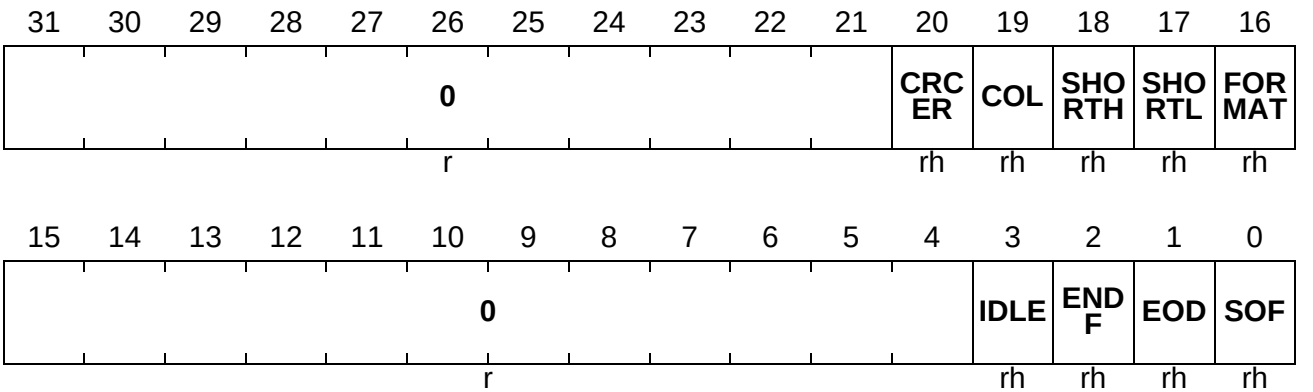
Field	Bits	Type	Description
TIP	0	rh	**Transmission in Progress** 0 The SDLM Module is not currently sending a frame on the bus or has finished its transmission. 1 The SDLM Module is sending the contents of its transmit buffer or IFR.IFRVAL on the bus and has not yet lost arbitration. TIP is reset by hardware if the arbitration is lost or STAT1.EOD or STAT1.ENDF are set.
RIP	1	rh	**Reception in Progress** 0 The SDLM Module is not currently receiving a frame on the bus. 1 The SDLM Module is receiving a frame on the bus. RIP is reset by hardware when STAT1.ENDF is set.

Field	Bits	Type	Description
MSGLST	2	rh	**Message Lost** Bit MSGLST indicates an incoming frame when the receive buffer on bus side is full (RBB = 1, contains received message data and has not yet been swapped). MSGLST is reset when MSGREC is reset in Normal Mode. If overwrite is enabled (CON.OVWR = 1), the receive buffer on bus side will be overwritten. In Block Mode, MSGLST is set if RBB is set and a new byte is received. If CON.OVWR = 0, the new byte is not stored. If CON.OVWR = 1, the new byte is stored. MSGLST must be reset by software.
RBB	3	rh	**Receive Buffer on Bus Side Full** RBB = 1 indicates that the receive buffer on the J1850 bus side is full. In case of a new incoming message, this buffer is declared empty (RBB = 0) and overwritten by the new data if bit CON.OVWR = 1. If CON.OVWR = 0, the buffer status and its contents are not changed (new data is not received). RBB is reset by hardware when the buffer is swapped to CPU side. In Block Mode, (only one buffer, so RBB = RBC) RBB is set upon a pointer match after reception of a byte. It is reset by reading from the receive buffer.
RBC	4	rh	**Receive Buffer on CPU Side Full** RBC = 1 indicates that the receive buffer on CPU side is full (not empty). This buffer remains allocated by the CPU until bit BUFCON.DONE has been set by software.
MSGTRA	16	rh	**Message Transmitted** **Normal Mode:** MSGTRA is set after the complete transmission of the transmit data or IFR.IFRVAL (arbitration won and EOD detected). Bit is reset when bit FR.TXRST is set. **Block Mode:** MSGTRA is set upon a pointer match after transmission of a byte or after end of frame has been detected (STAT1.ENDF set). Bit is reset when the CPU writes to the transmit buffer or when bit FR.TXRST is set.

Field	Bits	Type	Description
MSGREC	17	rh	**Message Received** Normal Mode: MSGREC is set after the reception of a new frame in the receive buffer on bus side (STAT1.ENDF set). Bit is reset by bit FR.RXRST or by hardware if the buffer is overwritten. Block Mode: MSGREC is set if the pointers do not match after reception of a byte (FIFO not empty) or after end of frame has been detected (STAT1.ENDF set). Bit is reset when the CPU reads from the receive buffer or when bit FR.RXRST is set. *Note: MSGREC will not be set upon the reception of a self-echo message.*
HEADER	18	rh	**Header Received** HEADER is set by hardware after reception of the complete header byte(s) in the receive buffer on bus side. It is reset by hardware after reception of the complete frame.
BREAK	19	rh	**Break Received** If BREAK is set, a break symbol has been received on the J1850 bus. BREAK must be reset by software.
ARL	20	rh	**Arbitration Lost** ARL is set after the arbitration for transmission has been lost. It is reset by software or by hardware if the arbitration has been won or the transmission has been aborted.
H	21	rh	**H Bit in Consolidated Headers** This bit monitors the status of Bit 4 of the first byte in a frame on bus side.
K	22	rh	**K Bit in 3-Byte Consolidated Headers** This bit monitors the status of Bit 3 of the first byte in a frame on bus side.
Y	23	rh	**Y Bit in 3-Byte Consolidated Headers** This bit monitors the status of Bit 2 of the first byte in a frame on bus side.
0	[15:5], [31:24]	r	**Reserved**; returns 0 if read.

Status Register 1 contains the error flags and the frame related status bits.

STAT1
Status Register 1 **Reset Value: 0000 0000$_H$**

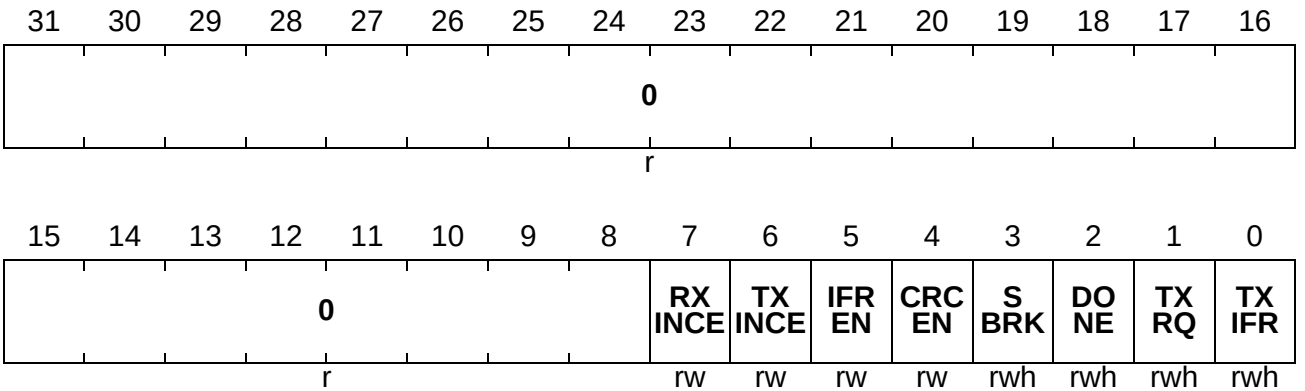


Field	Bits	Type	Description
SOF	0	rh	**Start Of Frame Detected** 0 No start of frame (SOF) detected 1 Start of frame has been detected Bit is reset by setting FR.BUSRST.
EOD	1	rh	**End Of Data Detected** 0 No End of data (EOD) detected 1 End of data detected Bit is reset by setting FR.BUSRST.
ENDF	2	rh	**End Of Frame Detected** 0 No End of frame (EOF) detected 1 End of frame detected Bit is reset by setting FR.BUSRST.
IDLE	3	rh	**Bus Idle** 0 The bus is currently not idle (frame is sent) 1 The bus is currently idle This bit is set by hardware after IFS and reset by hardware with the start of a new frame.
FORMAT	16	rh	**Format Error** 0 No frame error detected 1 Frame error detected. A frame length error or byte length error or symbol timing error or bit timing error has occurred. Bit is reset by setting FR.ERRST.

Field	Bits	Type	Description
SHORTL	17	rh	**Bus Shorted Low** 0 No bus shorted low error detected 1 The device tries to sent data, but permanently reads a 0 on the J1850 bus. Bit is reset by setting FR.ERRST.
SHORTH	18	rh	**Bus Shorted High** 0 No bus shorted high error detected 1 The device reads a 1 on the J1850 bus for more than one second. Bit is reset by setting FR.ERRST.
COL	19	rh	**Collision Detected** 0 No bus collision detected 1 A bus collision has been detected Bit is reset by setting FR.ERRST.
CRCER	20	rh	**CRC Error** 0 No CRC error detected 1 CRC error detected. The calculated CRC value differs from the received CRC value. Bit is reset by setting FR.ERRST.
0	[15:4], [31:21]	r	**Reserved**; returns 0 if read.

The Buffer Control Register contains the transfer-related control bits, including IFR control and FIFO control.

BUFCON
Buffer Control Register **Reset Value: 0000 0000$_H$**



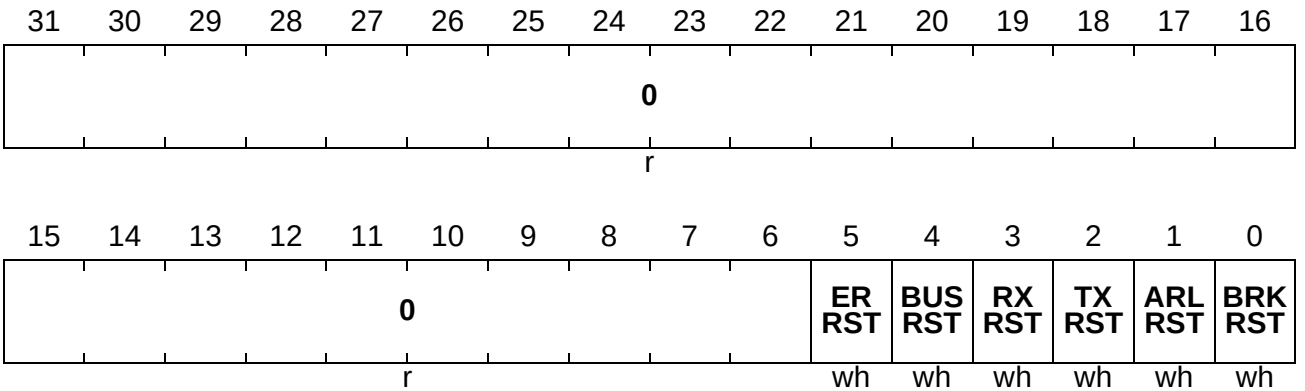
Field	Bits	Type	Description
TXIFR	0	rwh	**Transmit In-Frame Response** Setting bit TXIFR declares the transmit buffer or the IFR register (if IFREN = 1, IFR type 1,2, others than 3-byte consolidated header) to be valid for IFR and initiates its transmission. TXIFR is reset by hardware after successful transmission of the IFR. Resetting TXIFR by software stops the transmission of the IFR.
TXRQ	1	rwh	**Transmit Request** Setting bit TXRQ declares the transmit buffer to be valid and starts its transmission. TXRQ is reset by hardware after successful transmission. Resetting TXRQ by software stops the transmission in Normal Mode and in Block Mode. TXRQ can not be used to start IFR transmission from the transmit buffer. If TXRQ is reset, TXCPU is cleared also.
DONE	2	rwh	**Receive Buffer on CPU Side Read Out Done** Setting bit DONE declares the receive buffer on CPU side to be empty, resets RXCPU and releases the buffer (reset of STAT0.RBC). If there is a full receive buffer on bus side, the buffers are swapped. This bit is reset after the buffer has been released.

Field	Bits	Type	Description
SBRK	3	rwh	**Send Break** Setting bit SBRK initiates the transmission of a break symbol on the J1850 bus. Bit SBRK is reset by hardware after having sent the break symbol.
CRCEN	4	rw	**CRC Enable** 0 No CRC generation for type IFR via TXDATA 1 CRC enabled for IFR via TXDATA If IFR is sent from IFRVAL, no CRC is generated (even if CRCEN = 1). CRC generation is always enabled in Normal Mode and in Block Mode.
IFREN	5	rw	**In-Frame Response Enable** Setting bit IFREN enables the automatic IFR (type 1, 2 for 3-byte consolidated header) of the SDLM Module with the value stored in IFRVAL. If the IFR request cannot be automatically detected (all headers, except see above), IFREN selects the data source for types 1, 2 IFR initiated by TXIFR. 0 Transmit buffer contains IFR data byte(s), IFR types 1, 2, 3 supported, CRC depends on CRCEN. 1 IFRVAL contains data byte for types 1, 2 IFR. Automatic IFR for types 1, 2 for 3-byte consolidated headers is supported. No CRC is used.
TXINCE	6	rw	**Transmit Buffer Increment Enable** Random Mode is always enabled. 0 FIFO Mode is disabled for transmit buffer 1 FIFO Mode is enabled additionally TXPTR.TXCPU is incremented after each CPU write operation to the transmit data register. In Block Mode, FIFO Mode is automatically enabled (independent on TXINCE).

Field	Bits	Type	Description
RXINCE	7	rw	**Receive Buffer Increment Enable** Random Mode is always enabled. 0 FIFO Mode of receive buffer on CPU side is disabled 1 FIFO Mode is enabled additionally RXCPU is incremented after each CPU read operation to RXDATA. In Block Mode, FIFO Mode is automatically enabled (does not depend on RXINCE).
0	[31:8]	r	**Reserved**; returns 0 if read; should be written with 0.

The Flag Reset Register contains the control bits to reset the error flags, the bus-related status flags, and the transfer-related status flags.

FR
Flag Reset Register **Reset Value: 0000 0000$_H$**

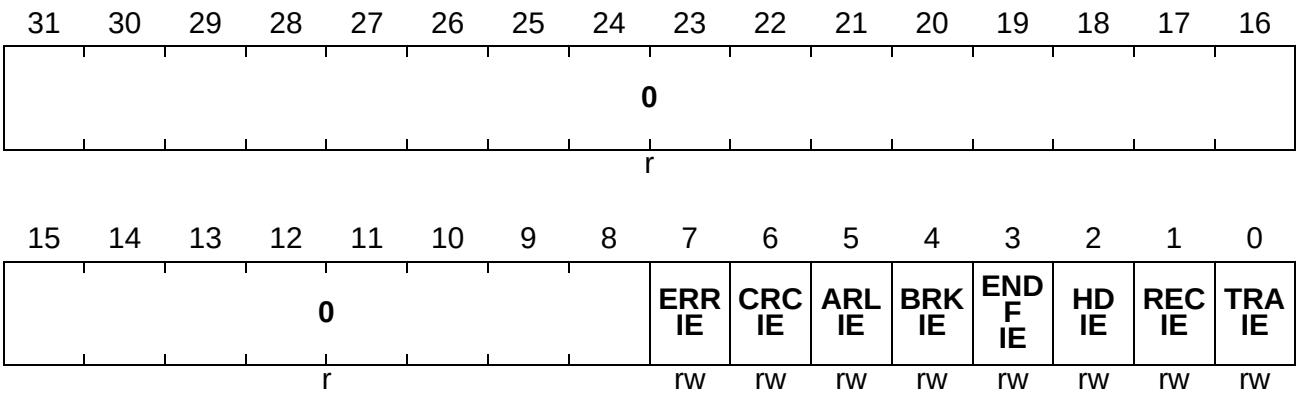


Field	Bits	Type	Description
BRKRST	0	wh	**Reset Break Received Status Flag** 0 No operation 1 Bit BREAK in register STAT0 is reset
ARLRST	1	wh	**Reset Arbitration Lost Status Flag** 0 No operation 1 Bit ARL in register STAT0 is reset
TXRST	2	wh	**Reset Message Transmitted Status Flag** 0 No operation 1 Bit MSGTRA in register STAT0 is reset
RXRST	3	wh	**Reset Message Received and Message Lost Status Flags** 0 No operation 1 Bits MSGREC and MSGLST in register STAT0 are reset
BUSRST	4	wh	**Reset Bus Status Flags** 0 No operation 1 Bits ENDF, EOD, and SOF in register STAT1 are reset
ERRST	5	wh	**Reset Error Flags** Bits SHORTH, SHORTL, COL, CRCER, and FORMAT in register STAT1 are reset.
0	[31:6]	r	**Reserved**; returns 0 if read; should be written with 0.

Note: Bits 5-0 of register FR are reset by hardware after the corresponding bit reset operation is finished. A read operation of these bits always returns a zero.

The Interrupt Control Register contains interrupt enable bits.

IE
Interrupt Control Register **Reset Value: 0000 0000$_H$**

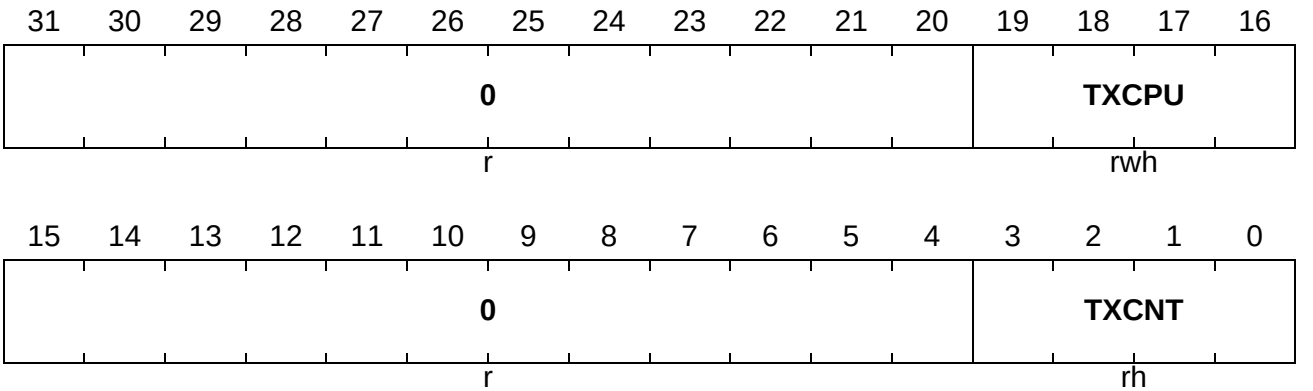


Field	Bits	Type	Description
TRAIE	0	rw	**Enable Transmit Interrupt** 0 The transmission interrupt is disabled. 1 An interrupt is generated if bit MSGTRA in register STAT0 is set.
RECIE	1	rw	**Enable Receive Interrupt** 0 The receive interrupt is disabled. 1 An interrupt is generated if bit MSGREC in register STAT0 is set.
HDIE	2	rw	**Enable Header Received Interrupt** 0 The header interrupt is disabled. 1 An interrupt is generated if bit HEADER in register STAT0 is set.
ENDFIE	3	rw	**Enable End of Frame Detection** 0 The end-of-frame interrupt is disabled. 1 An interrupt is generated if bit ENDF in register STAT1 is set.
BRKIE	4	rw	**Enable Break Received Interrupt** 0 The break interrupt is disabled. 1 An interrupt is generated if bit BREAK in register STAT0 is set.

Field	Bits	Type	Description
ARLIE	5	rw	**Enable Arbitration Lost Interrupt** 0 The arbitration-lost interrupt is disabled. 1 An interrupt is generated if bit ARL in register STAT0 is set.
CRCIE	6	rw	**Enable CRC Error Interrupt** 0 The CRC error interrupt is disabled. 1 An interrupt is generated if bit CRCER in register STAT1 is set.
ERRIE	7	rw	**Enable Error Interrupt** 0 The error interrupt is disabled. 1 An interrupt is generated if at least one of the bits SHORTH, SHORTL, COL, or FORMAT in register STAT1 is set.
0	[31:8]	r	**Reserved**; returns 0 if read; should be written with 0.

The Transmission Pointer Register contains the two pointers for the control of the transmit buffer.

TXPTR
Transmission Pointer Register **Reset Value: 0000 0000$_H$**

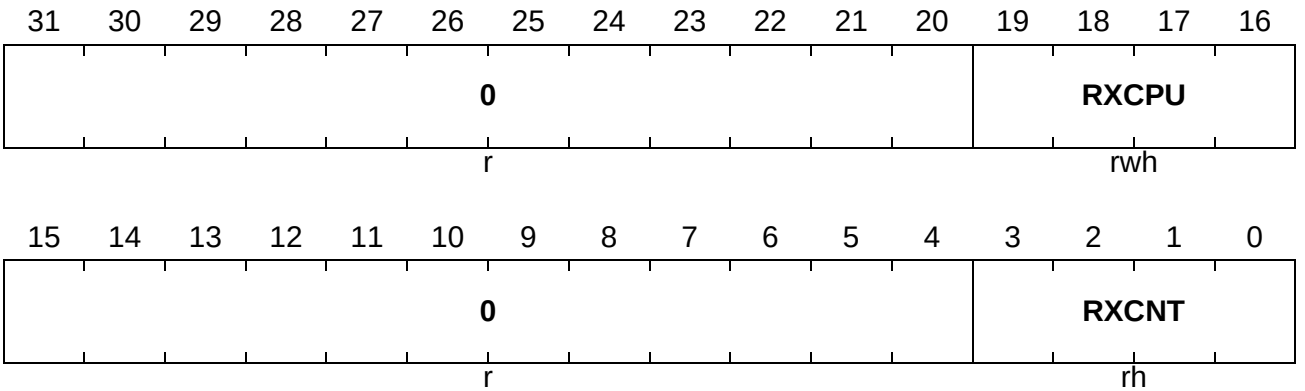


Field	Bits	Type	Description
TXCNT	[3:0]	rh	**Bus Transmit Byte Counter** Bit field TXCNT contains the number of bytes transmitted by the SDLM Module. TXCNT is reset by resetting bit BUFCON.TXRQ. A transmit interrupt can be generated when TXRQ is reset by hardware. If TXCPU is equal to TXCNT and a successful transmission occurred (arbitration not lost), bit TXRQ is reset by hardware in Normal Mode. = pointer for SDLM access to transmit buffer
TXCPU	[19:16]	rwh	**CPU Transmit Byte Counter** Bit field TXCPU contains the number of bytes, which have been written to the transmit buffer by the CPU (0 to 11). In FIFO Mode (TXINCE = 1 or BMEN = 1), TXCPU is incremented by 1 after each CPU write action to register TXDATA. In Random Mode only (TXINCE = 0 and BMEN = 0), TXCPU must be written by software before setting the transmit request bit (TXRQ) in order to define the number of bytes to be sent. TXCPU is reset by resetting bit TXRQ in Normal Mode or resetting BMEN. For more details see TXCNT. = pointer for CPU access to transmit buffer in FIFO Mode

Field	Bits	Type	Description
0	[15:4], [31:20]	r	**Reserved**; returns 0 if read; should be written with 0.

The Receive Pointer Register on CPU Side contains the two pointers for control of the receive buffer on CPU side.

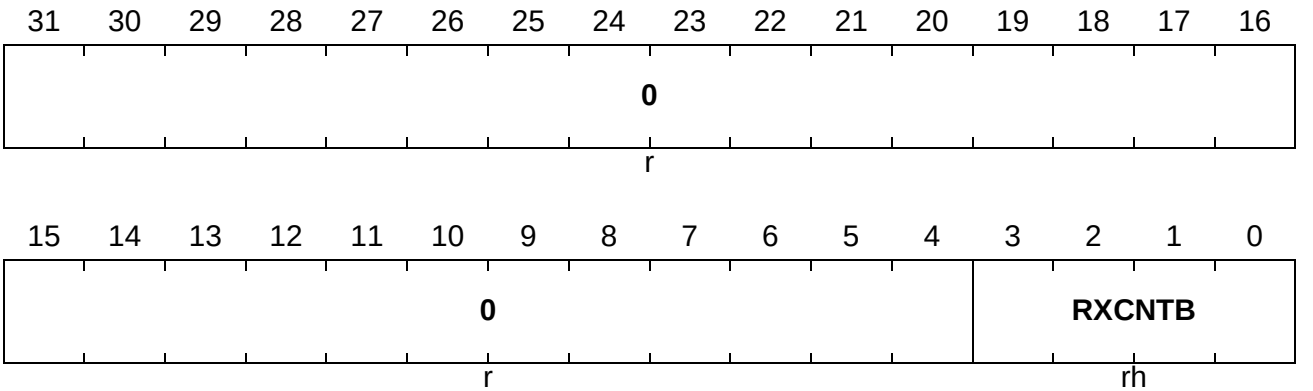
RXPTR
Receive Pointer Register on CPU Side **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
RXCNT	[3:0]	rh	**Bus Receive Byte Counter** Bit field RXCNT contains the number of received bytes in the receive buffer on CPU side. = pointer for SDLM access to receive buffer. RXCNT is reset when the receive buffer on CPU side is released (see DONE).
RXCPU	[19:16]	rwh	**CPU Receive Byte Counter** Bit field RXCPU contains the number of bytes read out by the CPU. In FIFO Mode (RXINCE = 1 or BMEN = 1), RXCPU is incremented by 1 after each CPU read action to register RXD00. In Random Mode (RXINCE = 0 and BMEN = 0), RXCPU is not used and is 0. = pointer for CPU access to receive buffer RXCPU is reset when the receive buffer on CPU side is released.
0	[15:4], [31:20]	r	**Reserved**; returns 0 if read; should be written with 0.

The Receive Pointer Register on Bus Side contains the number of received bytes in the receive buffer on bus side.

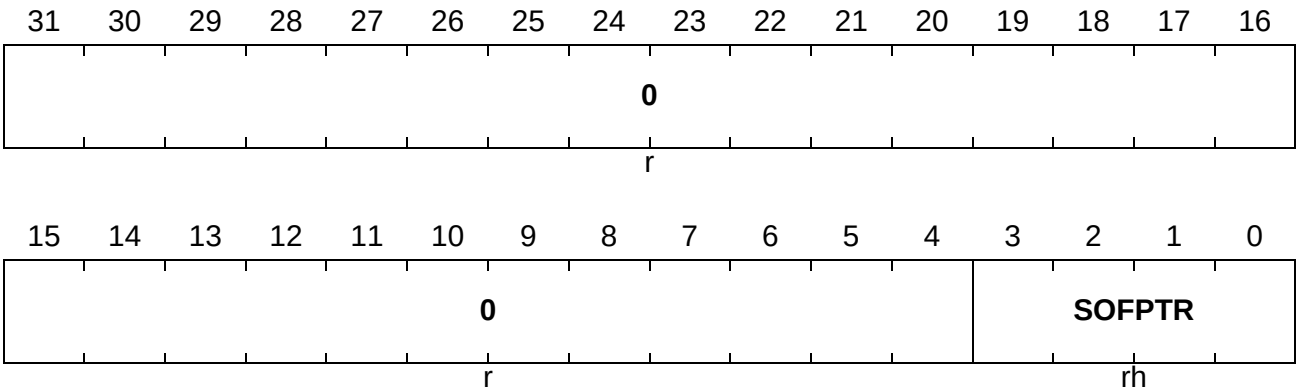
RXPTRB
Receive Pointer Register on Bus Side **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
RXCNTB	[3:0]	rh	**Receive Byte Counter** Bit field RXCNTB contains the number of received bytes in the receive buffer on bus side.
0	[31:4]	r	**Reserved**; returns 0 if read; should be written with 0.

The Start of Frame Register contains the bit field indicating the value RXCNT after the last end of frame detection in Block Mode.

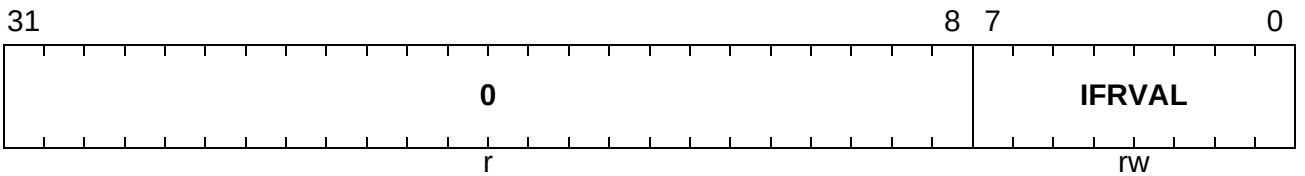
SPTR
Start of Frame Pointer Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
SOFPTR	[3:0]	rh	**Start of Frame Pointer** The value of bit field RXPTR.RXCNT is automatically copied into this bit field if an end of frame symbol is detected. This feature can be used in Block Mode to determine the position of the first new byte of a frame in the 16-byte receive buffer.
0	[31:4]	r	**Reserved**; returns 0 if read; should be written with 0.

The In-Frame Response Value Register contains a bit field that can be sent out as source ID in case of an 1-byte IFR (automatic transmission or triggered by software).

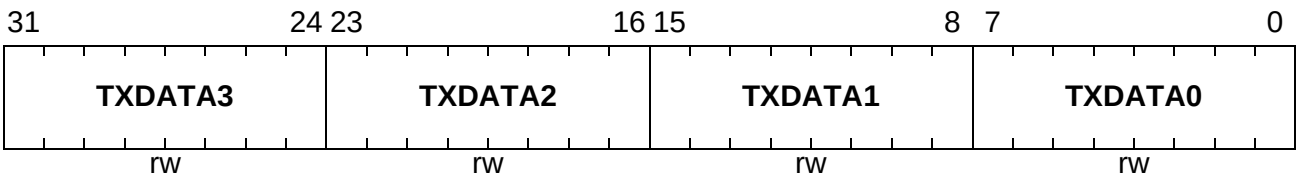
IFR
In-Frame Response Value Register **Reset Value: 0000 00000$_H$**



Field	Bits	Type	Description
IFRVAL	[7:0]	rw	**In-Frame Response Value** Bit field IFRVAL contains the value to be transmitted in case of requested in-frame response (type 1, 2 IFR, 1 byte response). It must be enabled by BUFCON.IFREN and initialized by the CPU.
0	[31:8]	r	**Reserved**; returns 0 if read; should be written with 0.

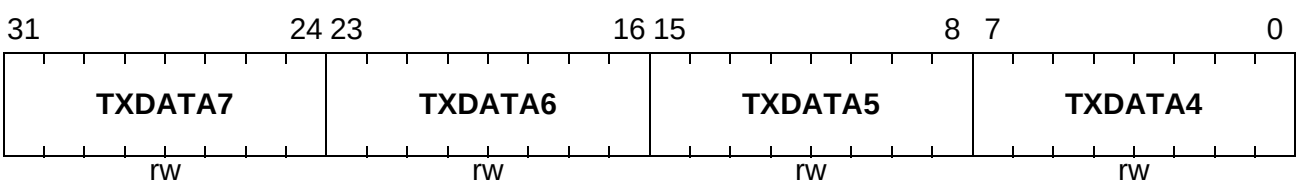
The Transmit Data Registers contain the data bytes in the transmit buffer. In Random Mode, all data bytes can be directly accessed via their addresses, whereas in FIFO Mode, only TXD0 should be used.

TXD0
Transmit Data Register 0 **Reset Value: 0000 00000$_H$**



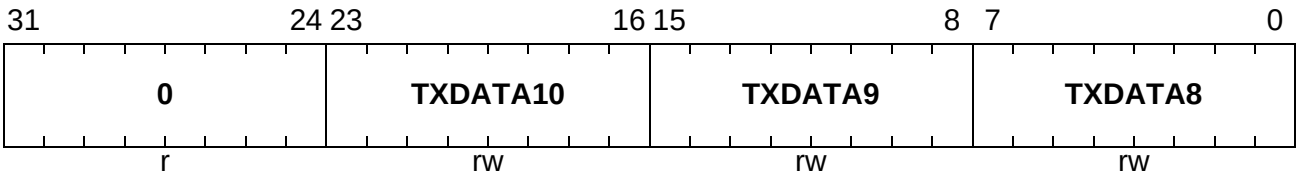
Field	Bits	Type	Description
TXDATA0	[7:0]	rw	Transmit Data Byte 0
TXDATA1	[15:8]	rw	Transmit Data Byte 1
TXDATA2	[23:16]	rw	Transmit Data Byte 2
TXDATA3	[31:24]	rw	Transmit Data Byte 3

TXD4
Transmit Data Register 4 **Reset Value: 0000 00000$_H$**



Field	Bits	Type	Description
TXDATA4	[7:0]	rw	Transmit Data Byte 4
TXDATA5	[15:8]	rw	Transmit Data Byte 5
TXDATA6	[23:16]	rw	Transmit Data Byte 6
TXDATA7	[31:24]	rw	Transmit Data Byte 7

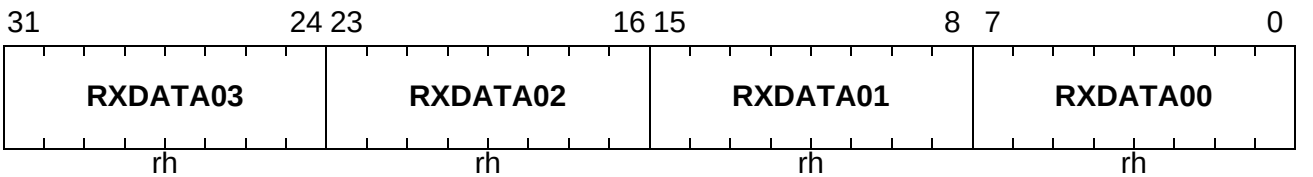
TXD8
Transmit Data Register 8 **Reset Value: 0000 00000$_H$**



Field	Bits	Type	Description
TXDATA8	[7:0]	rw	Transmit Data Byte 8
TXDATA9	[15:8]	rw	Transmit Data Byte 9
TXDATA10	[23:16]	rw	Transmit Data Byte 10
0	[31:24]	r	**Reserved**; returns 0 if read; should be written with 0.

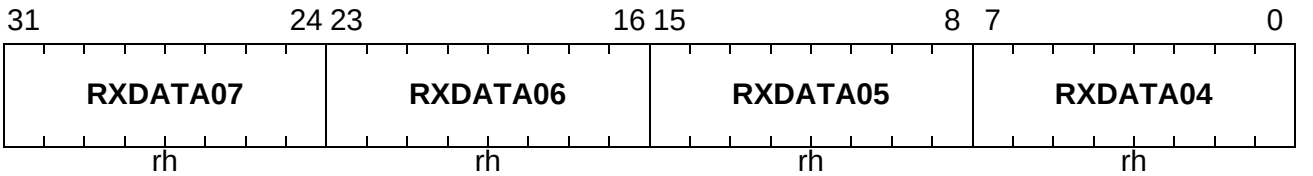
The Receive Data Registers contain the data bytes of the receive buffers. Random Mode, all data bytes can be directly accessed via their addresses, whereas in FIFO Mode, only register RXD00 should be used. Registers RXD0x are assigned to the receive buffer on CPU side and registers RXD1x are assigned to the receive buffer on bus side. In Block Mode, the 16-byte receive buffer is built by RXD00, RXD04 and RXD10, RXD14.

RXD00
Receive Buffer 0 Data Register 0 on CPU Side **Reset Value: 0000 00000$_H$**



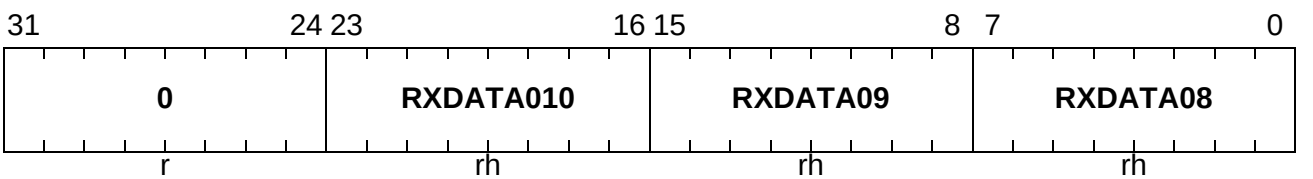
Field	Bits	Type	Description
RXDATA00	[7:0]	rh	**Receive Buffer 0 Data Byte 0** Byte 0 in the receive buffer on CPU side.
RXDATA01	[15:8]	rh	**Receive Buffer 0 Data Byte 1** Byte 1 in the receive buffer on CPU side.
RXDATA02	[23:16]	rh	**Receive Buffer 0 Data Byte 2** Byte 2 in the receive buffer on CPU side.
RXDATA03	[31:24]	rh	**Receive Buffer 0 Data Byte 3** Byte 3 in the receive buffer on CPU side.

RXD04
Receive Buffer 0 Data Register 4 on CPU Side **Reset Value: 0000 00000$_H$**



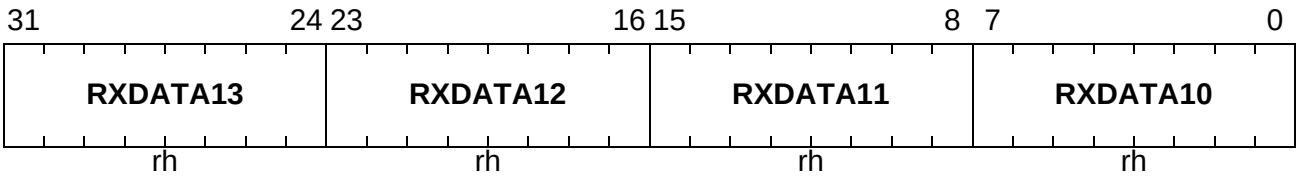
Field	Bits	Type	Description
RXDATA04	[7:0]	rh	**Receive Buffer 0 Data Byte 4** Byte 4 in the receive buffer on CPU side.
RXDATA05	[15:8]	rh	**Receive Buffer 0 Data Byte 5** Byte 5 in the receive buffer on CPU side.
RXDATA06	[23:16]	rh	**Receive Buffer 0 Data Byte 6** Byte 6 in the receive buffer on CPU side.
RXDATA07	[31:24]	rh	**Receive Buffer 0 Data Byte 7** Byte 7 in the receive buffer on CPU side.

RXD08
Receive Buffer 0 Data Register 8 on CPU Side **Reset Value: 0000 00000$_H$**



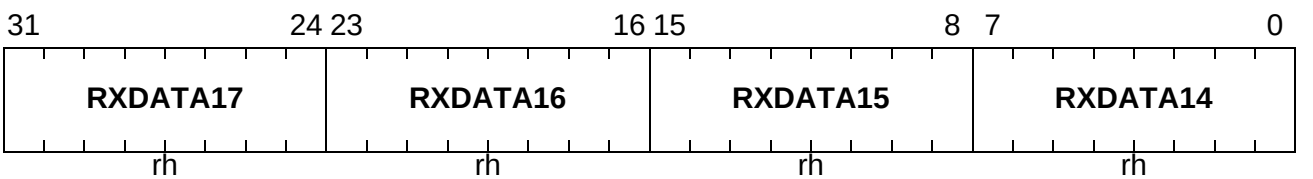
Field	Bits	Type	Description
RXDATA08	[7:0]	rh	**Receive Buffer 0 Data Byte 8** Byte 8 in the receive buffer on CPU side.
RXDATA09	[15:8]	rh	**Receive Buffer 0 Data Byte 9** Byte 9 in the receive buffer on CPU side.
RXDATA010	[23:16]	rh	**Receive Buffer 0 Data Byte 10** Byte 10 in the receive buffer on CPU side.
0	[31:24]	r	**Reserved**; returns 0 if read; should be written with 0.

RXD10
Receive Buffer 1 Data Register 0 on Bus Side **Reset Value: 0000 00000$_H$**



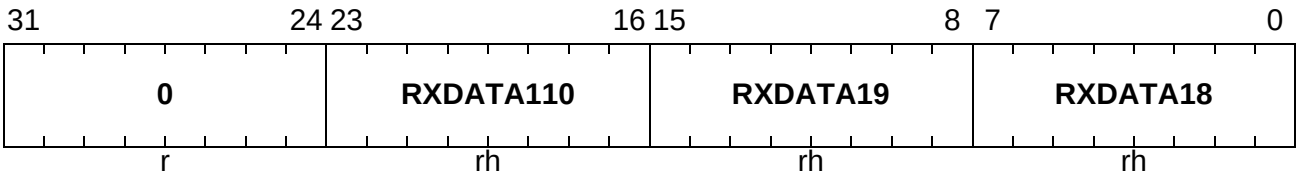
Field	Bits	Type	Description
RXDATA10	[7:0]	rh	**Receive Buffer 1 Data Byte 0** Byte 0 in the receive buffer on bus side.
RXDATA11	[15:8]	rh	**Receive Buffer 1 Data Byte 1** Byte 1 in the receive buffer on bus side.
RXDATA12	[23:16]	rh	**Receive Buffer 1 Data Byte 2** Byte 2 in the receive buffer on bus side.
RXDATA13	[31:24]	rh	**Receive Buffer 1 Data Byte 3** Byte 3 in the receive buffer on bus side.

RXD14
Receive Buffer 1 Data Register 4 on Bus Side **Reset Value: 0000 00000$_H$**



Field	Bits	Type	Description
RXDATA14	[7:0]	rh	**Receive Buffer 1 Data Byte 4** Byte 4 in the receive buffer on bus side.
RXDATA15	[15:8]	rh	**Receive Buffer 1 Data Byte 5** Byte 5 in the receive buffer on bus side.
RXDATA16	[23:16]	rh	**Receive Buffer 1 Data Byte 6** Byte 6 in the receive buffer on bus side.
RXDATA17	[31:24]	rh	**Receive Buffer 1 Data Byte 7** Byte 7 in the receive buffer on bus side.

RXD18
Receive Buffer 1 Data Register 8 on Bus Side **Reset Value: 0000 00000$_H$**



Field	Bits	Type	Description
RXDATA18	[7:0]	rh	**Receive Buffer 1 Data Byte 8** Byte 8 in the receive buffer on bus side.
RXDATA19	[15:8]	rh	**Receive Buffer 1 Data Byte 9** Byte 9 in the receive buffer on bus side.
RXDATA110	[23:16]	rh	**Receive Buffer 1 Data Byte 10** Byte 10 in the receive buffer on bus side.
0	[31:24]	r	**Reserved**; returns 0 if read; should be written with 0.

5.3 SDLM Module Implementation

This section describes the SDLM Module related external functions such as port connections, interrupt control, address decoding, and clock control.

5.3.1 Interfaces of the SDLM Module

Figure 5-23 shows the TC1775 specific implementation details and interconnections of the SDLM Module. The SDLM Module has two I/O lines which are located at Port 12. The SDLM Module is further supplied by a separate clock control, interrupt control, and address decoding logic.

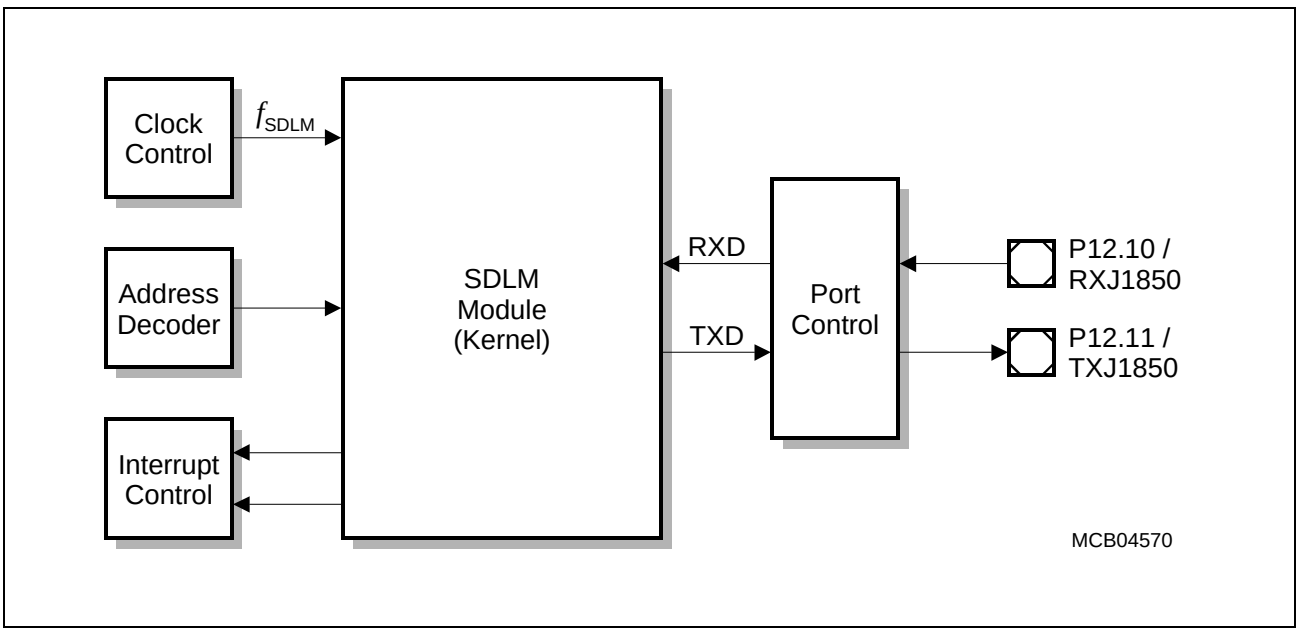


Figure 5-23 SDLM Module Implementation and Interconnections

5.3.2 SDLM Module Related External Registers

Figure 5-24 summarizes the module related external registers which are required for SDLM0/SDLM1 programming (see also **Figure 5-22** for the module kernel specific registers).

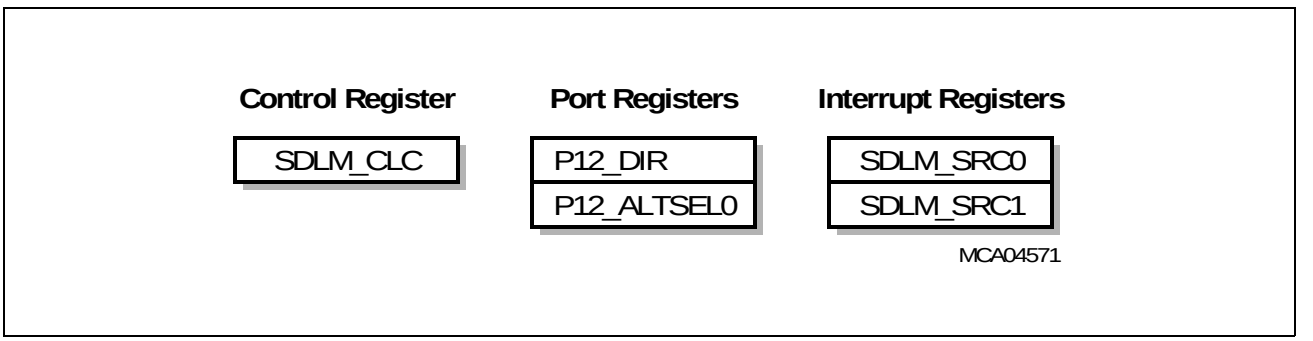
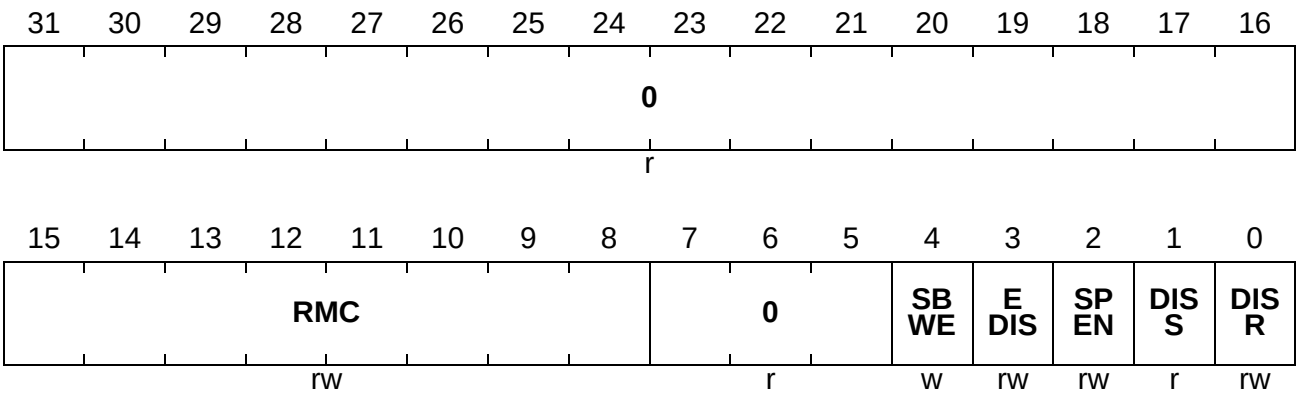


Figure 5-24 SDLM Implementation Specific Special Function Registers

5.3.2.1 Clock Control Register

The clock control register allows the programmer to adapt the functionality and power consumption of an SDLM Module to the requirements of the application. The diagram below shows the clock control register functionality which is implemented for the SDLM Module.

SDLM_CLC
SDLM Clock Control Register **Reset Value: 0000 0002$_H$**



Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control of the module.
DISS	1	r	**Module Disable Status Bit** Bit indicates the current status of the module.
SPEN	2	rw	**Module Suspend Enable for OCDS** Used for enabling the suspend mode.
EDIS	3	rw	**External Request Disable** Used for controlling the external clock disable request.
SBWE	4	w	**Module Suspend Bit Write Enable for OCDS** Defines whether SPEN and FSOE are write protected.
RMC	[15:8]	rw	**8-Bit Clock Divider Value in RUN Mode**
0	[7:5], [31:16]	r	**Reserved**; returns 0 if read; should be written with 0.

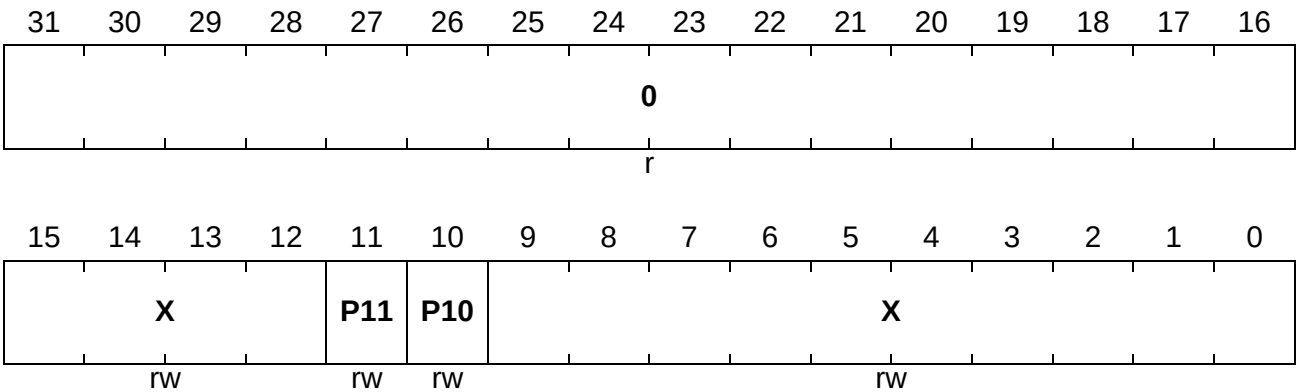
Note: After a hardware reset operation, the SDLM Module is disabled.

5.3.2.2 Port Registers

The alternate functions, associated with the two I/O lines, are controlled by the ALTSEL registers located in the ports. The SDLM I/O lines are connected with Port 12. Therefore, P12_ALTSEL0 must programmed for the port's two SDLM pins.

Note: Bits marked with 'X' are not relevant for SDLM operation.

P12_ALTSEL0
Port 12 Alternate Select Register 0 **Reset Value: 0000 0000$_H$**



Bit 10 and 11 of the Port 12 Alternate Select Register 0 must be set according **Table 5-3**.

Table 5-3 SDLM I/O Line Selection and Setup

Port Line	Alternate Function	Alternate Select Register P12_ALTSEL0 Bits	I/O Port Line Operation
P12.10	RXJ1850	P10 = 1	Input
P12.11	TXJ1850	P11 = 1	Output

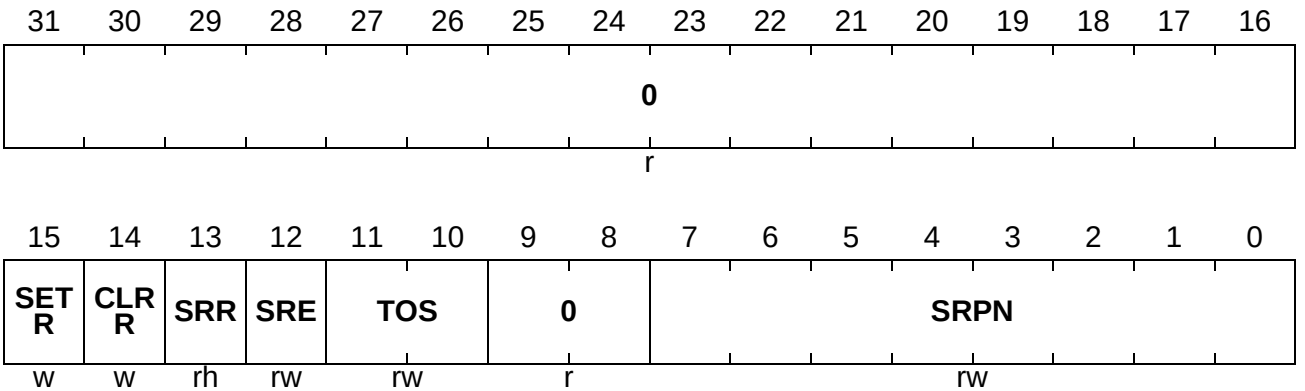
5.3.2.3 Interrupt Registers

The two interrupts of the SDLM Module are controlled by the following service request control registers:

– SDLM_SRC1: controls the data receive/transmit interrupts (INT1)
– SDLM_SRC0: controls the protocol related interrupts (INT0)

SDLM_SRC1
SDLM Service Request Control Register 1
SDLM_SRC0
SDLM Service Request Control Register 0

Reset Values: 0000 0000$_H$



Field	Bits	Type	Description
SRPN	[7:0]	rw	**Service Request Priority Number**
TOS	[11:10]	rw	**Type of Service Control**
SRE	12	rw	**Service Request Enable**
SRR	13	rh	**Service Request Flag**
CLRR	14	w	**Request Clear Bit**
SETR	15	w	**Request Set Bit**
0	[9:8], [31:16]	r	**Reserved**; returns 0 if read; should be written with 0.

Note: Further details on interrupt handling and processing are described in the “Interrupt System” chapter of the TC1775 System Units User’s Manual.

5.3.3 SDLM Register Address Range

In the TC1775, the registers of the SDLM Module are located in the following address range:

– Module Base Address. F000 2600_H
  Module End Address. F000 $26FF_H$
– Absolute Register Address = Module Base Address + Offset Address
  (offset addresses see **Table 5-2**)

6 General Purpose Timer Unit (GPTU)

This chapter describes the General Purpose Timer Unit (GPTU) of the TC1775. The information is presented in the following sections:

- Functional description of the GPTU Kernel (see **Section 6.1**)
- Register descriptions of all GPTU Kernel specific registers (see **Section 6.2**)
- TC1775 implementation specific details and registers of the GPTU (port connections and control, interrupt control, address decoding, clock control, see **Section 6.3**) with register address range (see **Section 6.3.3**).

*Note: The GPTU kernel register names described in **Section 6.2** will be referenced in other parts of the TC1775 User's Manual with the module name prefix "GPTU_".*

6.1 GPTU Kernel Description

Figure 6-1 shows a global view of all functional blocks of the GPTU kernel and its interfaces.

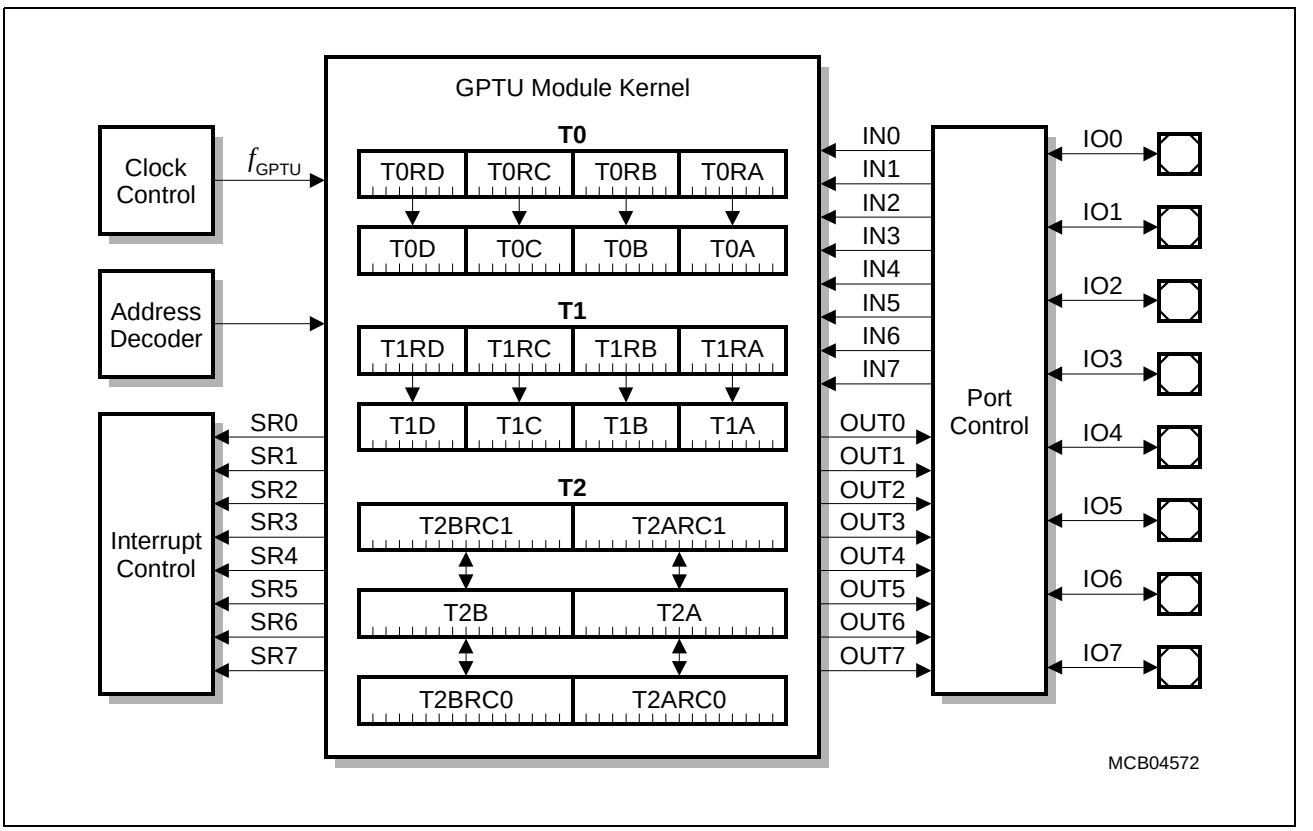


Figure 6-1 General Block Diagram of the GPTU Interface

The GPTU consists of three 32-bit timers designed to solve such application tasks as event timing, event counting, and event recording. The GPTU communicates with the external world via eight inputs and eight outputs concatenated in the port control logic to eight I/O pins IO[7:0]. The input signals coming from the port logic are named IN[7:0], and the output signals going to the port logic are named OUT[7:0]. These signals are used in the further descriptions of the timers. Further, the GPTU can generate eight service requests SR[7:0] within the TC1775.

Clock control, address decoding, and interrupt service request control are managed outside the GPTU Module kernel.

6.1.1 General Operation

The I/O has three timers (T0, T1, and T2) can operate independently from each other, or can be combined:

- All timers are 32-bit precision timers with a maximum input frequency of f_{GPTU}
- Events generated in T0 or T1 can be used to trigger actions in T2
- Timer overflow or underflow in T2 can be used to clock either T0 or T1
- T0 and T1 can be concatenated to form one 64-bit timer

Additional features of timers T0 and T1 include:

- Each timer has a dedicated 32-bit reload register with automatic reload on overflow
- Timers can be split into individual 8-, 16-, or 24-bit timers with individual reload registers
- Overflow signals can be selected to generate service requests, pin output signals, and T2 trigger events
- Two input pins can define a count option

Features of Timer T2 include:

- Count up or down is selectable
- Operating modes:
  - Timer
  - Counter
  - Quadrature counter (incremental/phase encoded counter interface)
- Options:
  - External start/stop, one-shot operation, timer clear on external event
  - Count direction control through software or an external event
  - Two 32-bit reload/capture registers
- Reload modes:
  - Reload on overflow or underflow
  - Reload on external event: positive transition, negative transition, or both transitions
- Capture modes:
  - Capture on external event: positive transition, negative transition, or both transitions
  - Capture and clear timer on external event: positive transition, negative transition, or both transitions
- Can be split into two 16-bit counter/timers
- Timer count, reload, capture, and trigger functions can be assigned to input pins. T0 and T1 overflow events can also be assigned to these functions
- Overflow and underflow signals can be used to trigger T0 and/or T1 and to toggle output pins
- T2 events are freely assignable to the service request nodes

6.1.2 Timers T0 and T1

Figure 6-2 and **Figure 6-3** show detailed block diagrams of Timers T0 and T1. Both, T0 and T1, consist of four 8-bit timer blocks named TxA, TxB, TxC, and TxD (x = 0, 1). Each eight-bit timer block contains a count register and a reload register. These blocks can be configured to run independently as 8-bit timers, or can be concatenated to form wider timers (16-bit, 24-bit, or 32-bit). A cross-connection between T0 and T1 extends these options to permit creation of a 64-bit timer.

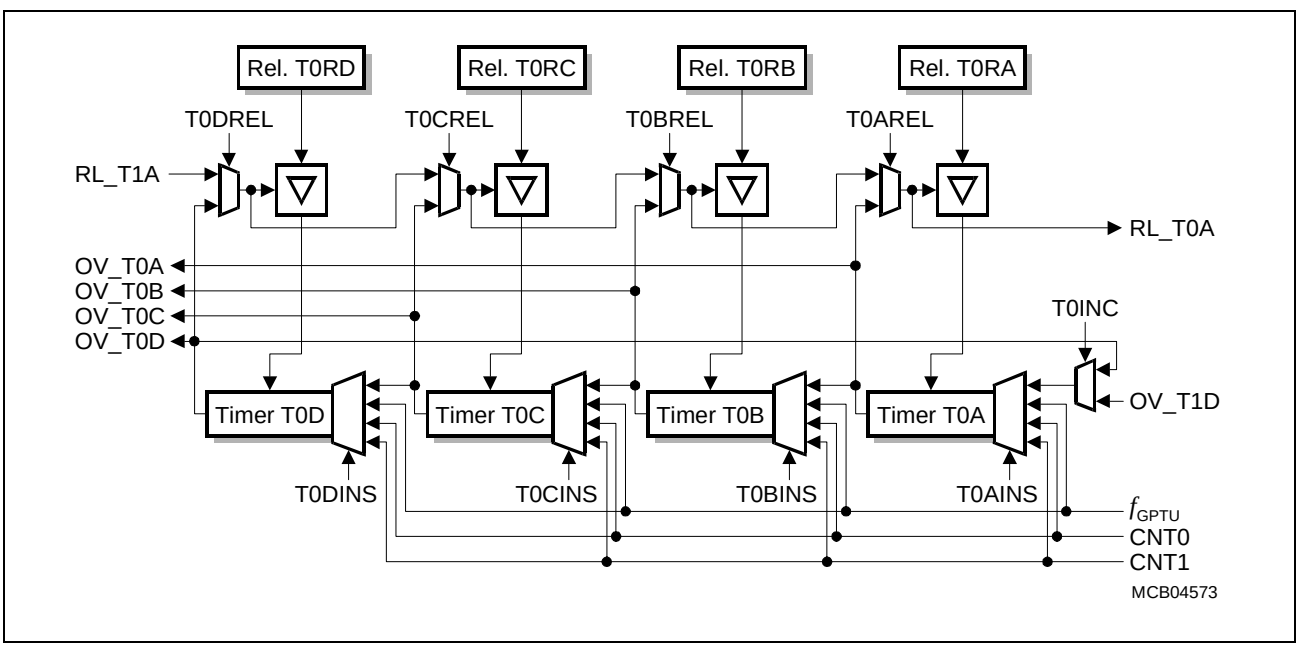


Figure 6-2 Detailed Block Diagram of T0

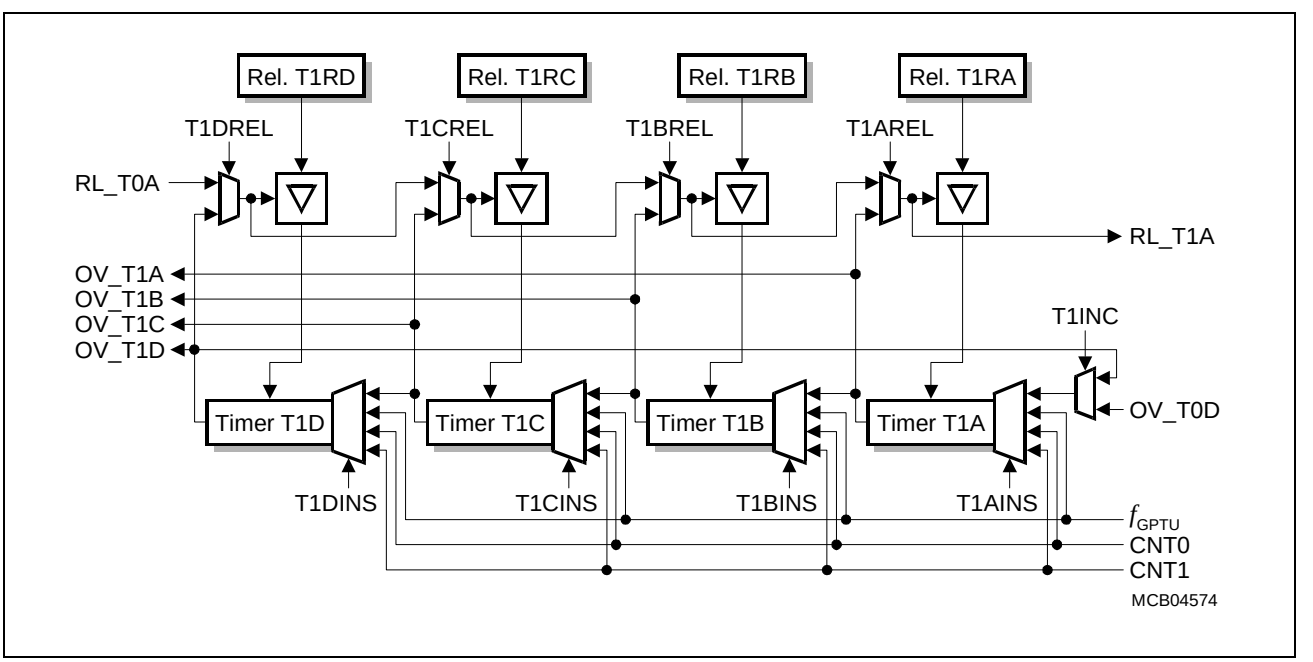


Figure 6-3 Detailed Block Diagram of T1

6.1.2.1 Input Selection

Each 8-bit timer block can select one of three possible inputs:

- The overflow of the previous timer (handled specially for T0A and T1A)
- An input frequency f_{GPTU} derived from the system clock
- One of two count inputs (CNT0, CNT1)

As shown in **Figure 6-2** and **Figure 6-3**, each of the four 8-bit timer blocks within T0 and T1 receives an overflow from the previous 8-bit timer block. Additionally, the A blocks of both timers can be separately configured to receive overflow either from its own D block, or the other's D block (by way of T0INC and T1INC). The two selectable configurations are:

1. The A blocks receive the overflow of their own D-block timer (T0A input is T0D overflow, and T1A input is T1D overflow).
2. The A blocks receive the overflow of the other's D-block timer (T0A input is T1D overflow, and T1A input is T0D overflow).

When configuration 1 is selected, T0 and T1 operate independently. Both timers can be set up individually as 8-bit, 16-bit, 24-bit, or 32-bit timers.

When configuration 2 is selected T0 and T1 inter-operate, and can be concatenated to form wider timers. For 40-bit, 48-bit, 56-bit or 64-bit operation, the timer not receiving overflow from the other timer must be driven by either the module clock, CNT0, or CNT1. Additionally, the overflow selection of the other 8-bit timers within T0 and T1 must all be configured appropriately to source overflow from its previous timer.

The source for the two count inputs (CNT0 or CNT1) can be either an external input or a trigger signal from T2 (by way of T2 overflow signals, OUV_T2A and OUV_T2B). **Figure 6-4** shows these options.

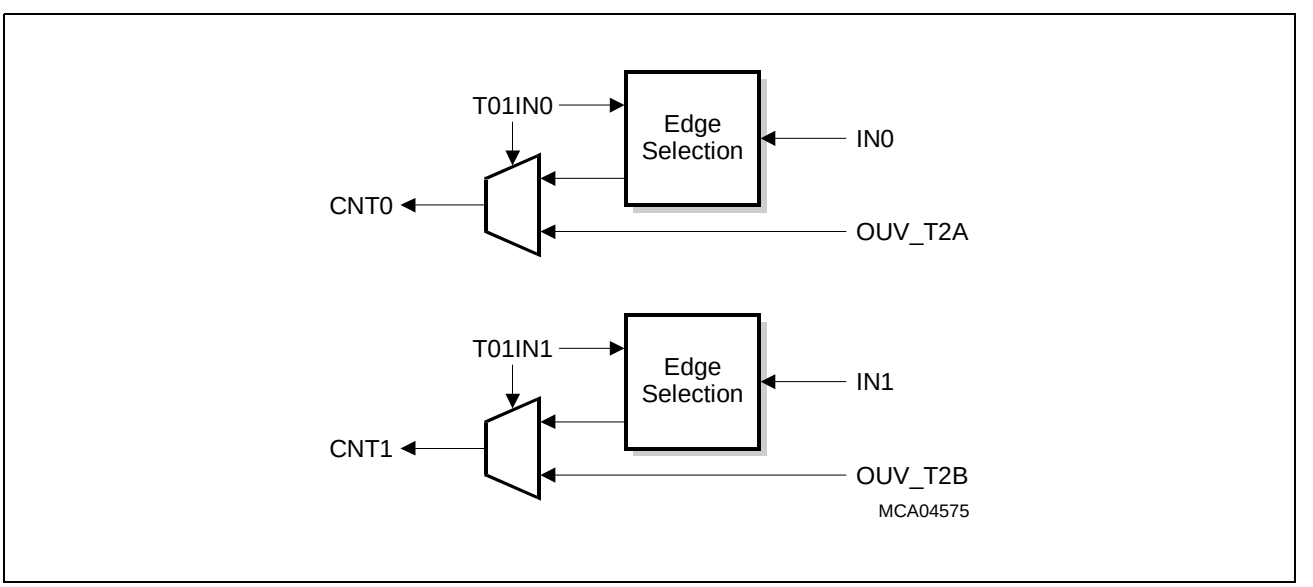


Figure 6-4 Timer T0 and T1 Global Input Control

Access to Timer T0 and T1 Count and Reload Registers

Two address locations are provided for each of the count and reload registers, which enable access to the appropriate registers even for a 24-bit timer configuration. The first address location provides all four bytes of a timer count/reload register. The symbolic name for this address indicates that all four parts, D..A, are accessible. Registers TxDCBA provide access to the count registers and registers TxRDCBA provide access to the reload registers. Individual access to the single bytes, combined 16-bit half-word-aligned combination, or full 32-bit combination, is possible in this way. The second address location provides the lower three bytes of a timer count/reload register; the most significant byte is not connected. The second address location enables access to a timer count/reload register in a 24-bit combination without corrupting the upper byte of the timer count/reload register. The symbolic name for this second address location is TxCBA (for the count registers) and TxRCBA (for the reload registers). These locations provide access only to the lower three parts, C..A, of the timer count and reload registers. **Table 6-1** gives an overview on the different access options to the individual combinations of T0 and T1.

Table 6-1 Access Options to T0/T1

Register	Access Width	Least Significant Address Bits	TxD TxRD	TxC TxRC	TxB TxRB	TxA TxRA
TxDCBA TxRDCBA	Byte	000				x
	Byte	001			x	
	Byte	010		x		
	Byte	011	x			
	Half-word	000			x	
	Half-word	010	x			
	Word	000	x			
TxCBA TxRCBA	Byte	100				x
	Byte	101			x	
	Byte	110		x		
	Byte	111	0			
	Half-word	100			x	
	Half-word	110	0	x		
	Word	100	0	x		

Reading and writing to the individual byte or half-word parts of a timer is performed on the first address location using byte or half-word load/store operations. The entire 32-bit

timer is accessed with word load/store operations. Reading from the second address location with a word load operation provides the contents of the lower three bytes of the timer count/reload register, with the most significant byte returning 0. Writing to it with a word store operation affects only the lower three bytes. The value of the most significant byte is not stored. It is recommended that software always writes 0 to the most significant byte. The second address location can also be accessed with byte or half-word load/ store operations.

Note: Access to a 16-bit half-word that crosses a half-word boundary (for example, the combination of T0C and T0B as one 16-bit timer) and access to a 24-bit combination using the upper three bytes (for example, T0D, T0C, and T0B) are not provided. Because it is always possible to align 16-bit timers on half-word boundaries, and right-align a 24-bit timer, these combinations are not required.

6.1.2.2 Reload Selection

As shown in **Figure 6-2** and **Figure 6-3**, the reload trigger signals for the reload registers are controlled independently from timer concatenation. The independent control provides the option of concatenating timers while giving each timer its own reload period. Reload selection is controlled by T0xREL and T1xREL so that each eight-bit timer can be triggered by either:

– The overflow of its own counter
– The reload event of one of the higher-order timer(s)

6.1.2.3 Service Requests, Output Signals, and Trigger Signals

Overflow signals from T0 and T1 can be used to generate service requests, output signals, or trigger signals for T2.

The four overflow signals from each 8-bit timer in T0 and T1 can trigger two service requests, two output signals, and two trigger signals. These options are shown in **Figure 6-5** for T0 and **Figure 6-6** for T1.

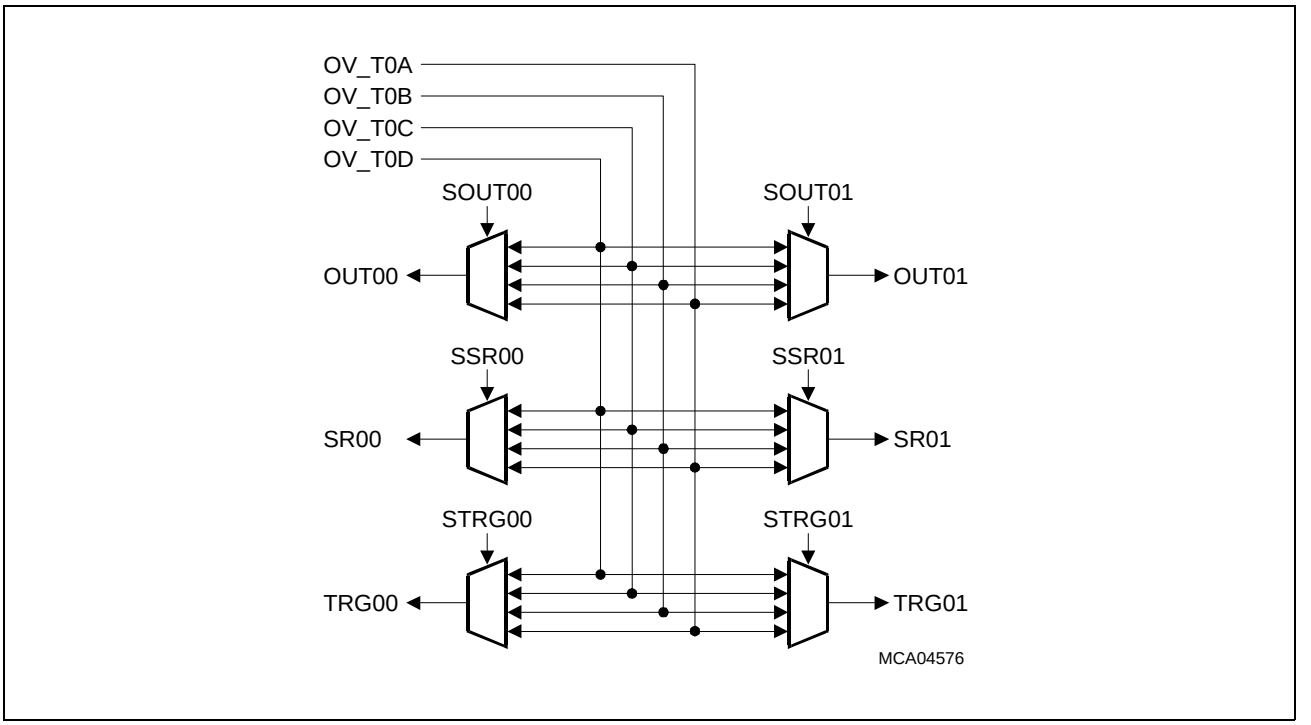


Figure 6-5 Timer T0 Output, Trigger, and Service Request Selection Control

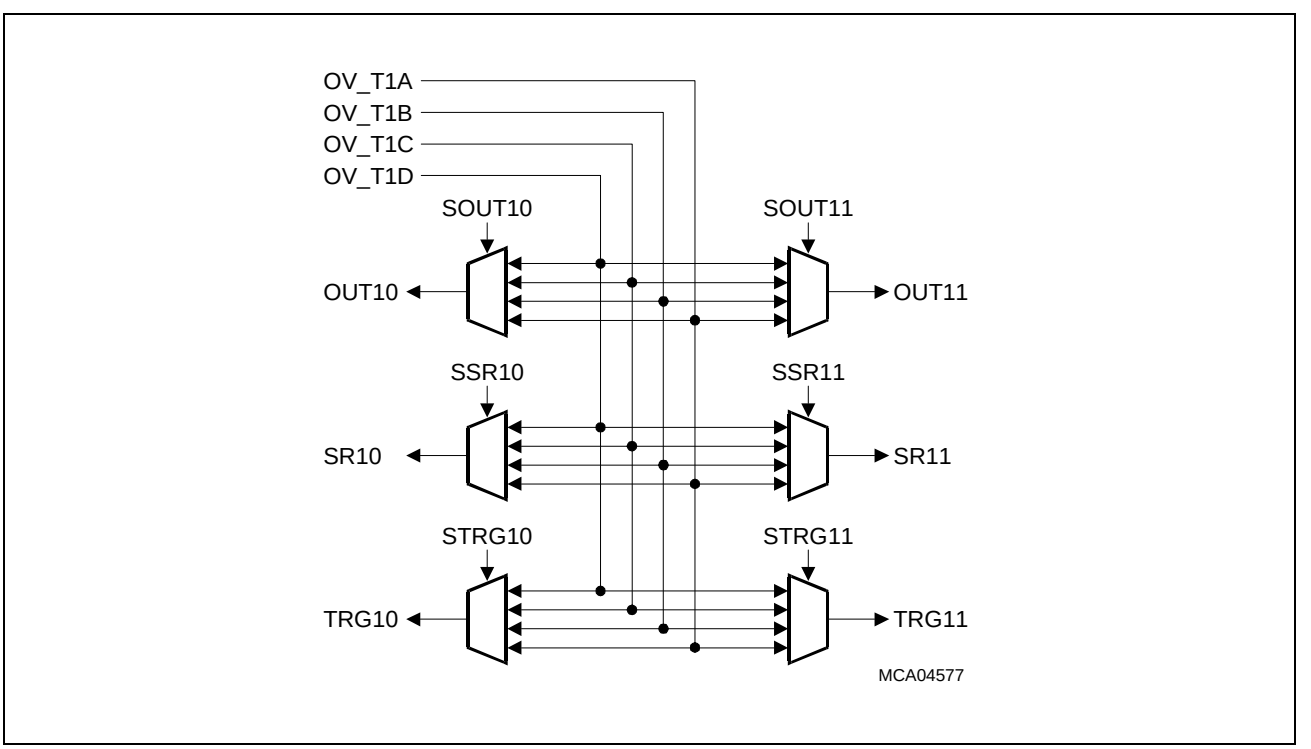


Figure 6-6 Timer T1 Output, Trigger, and Service Request Selection Control

6.1.2.4 Timers T0 and T1 Configuration Limitations

Due to timing delays of the internal circuitry, there are certain special cases and restrictions associated with the configuration possibilities of Timers T0 and T1.

In the following cases, one additional GPTU clock pulse is inserted into the count or reload signal:

- Overflow of T0D is used as count input to T0A
- Overflow of T1D is used as count input to T1A
- Overflow of T1D is used as count input to T0A
- Reload trigger of T0A/T0RA is used as reload trigger for T1D/T1RD

These combinations should either be avoided or the additional clock pulse needs to be taken into account. In the first three cases, if the timer producing the overflow is used as a prescaler for the following timer, the effect of the additional clock pulse is usually irrelevant. The prescaler just needs to be started such that the timer contents are one count higher than the reload value. This avoids a longer initial period due to pulse delay.

It is recommended that the fourth case is always avoided. This case would occur if T0 and T1 (or parts of them) are concatenated such that T1D is the less significant and T0A is the more significant part of this timer combination. The overflow of T1D would be used as count input to T0A would experience a clock delay. The reload trigger line from T0A back to T1D would experience another clock delay, resulting in a total delay of two GPTU clocks from T1D overflow to its reload event. Because T1D continues counting after its overflow, its contents will be overwritten by the reload two clock cycles later, resulting in the loss of two counts.

Concatenating T0 and T1 such that T0 contains the less significant part of the combined timer does not present a problem. The overflow of T0D to T1A and the reload trigger signal from T1A back to T0D do not have this extra delay.

Due to the high flexibility of the configuration options for Timers T0 and T1, it is almost never required to use one of the cases described above.

6.1.3 Timer T2

Timer T2 consists of two 16-bit timer blocks, T2A and T2B. Each 16-bit timer block contains a count register and two reload/capture registers. These blocks can be configured to form one 32-bit timer as shown in **Figure 6-7**, or to run independently as two 16-bit timers as shown in **Figure 6-8**. This basic configuration of Timer T2 is controlled by the T2CON.T2SPLIT control bit.

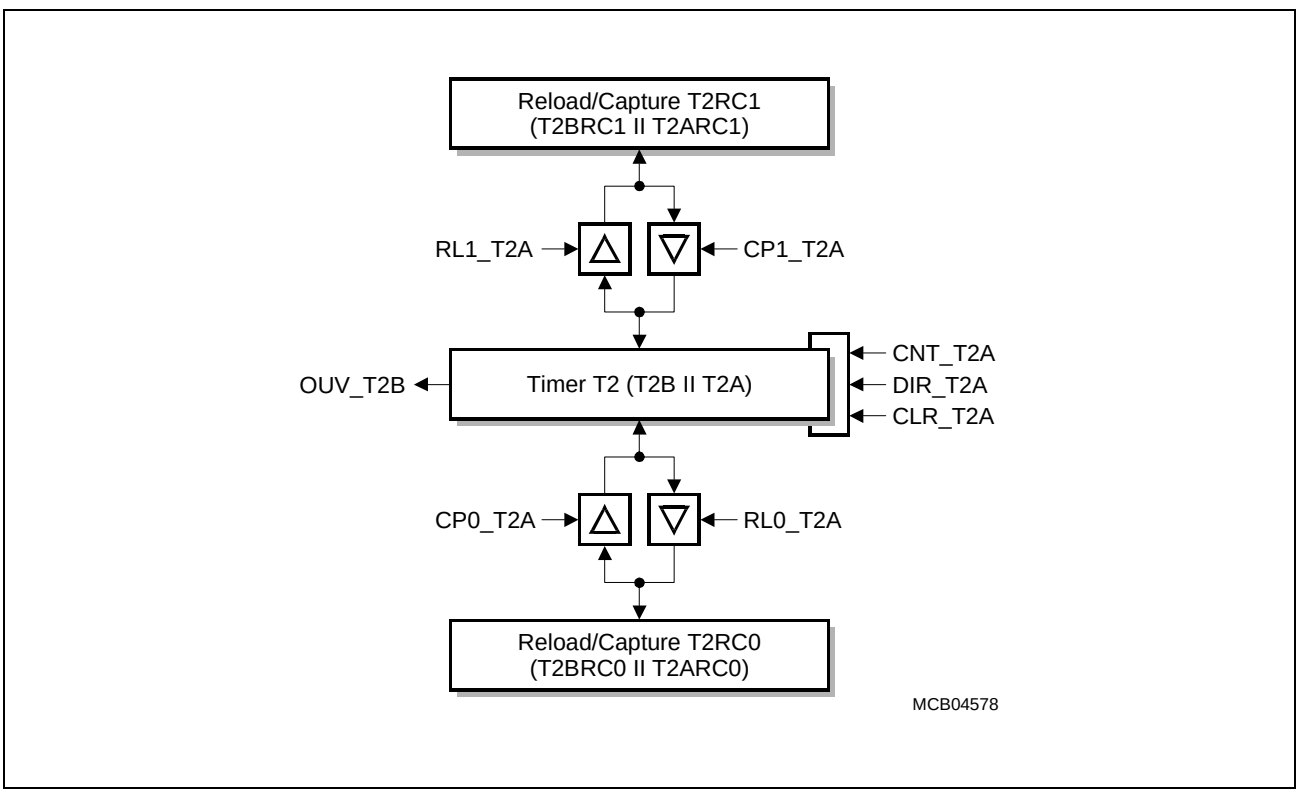


Figure 6-7 Block Diagram of Timer 2 in 32-Bit Mode

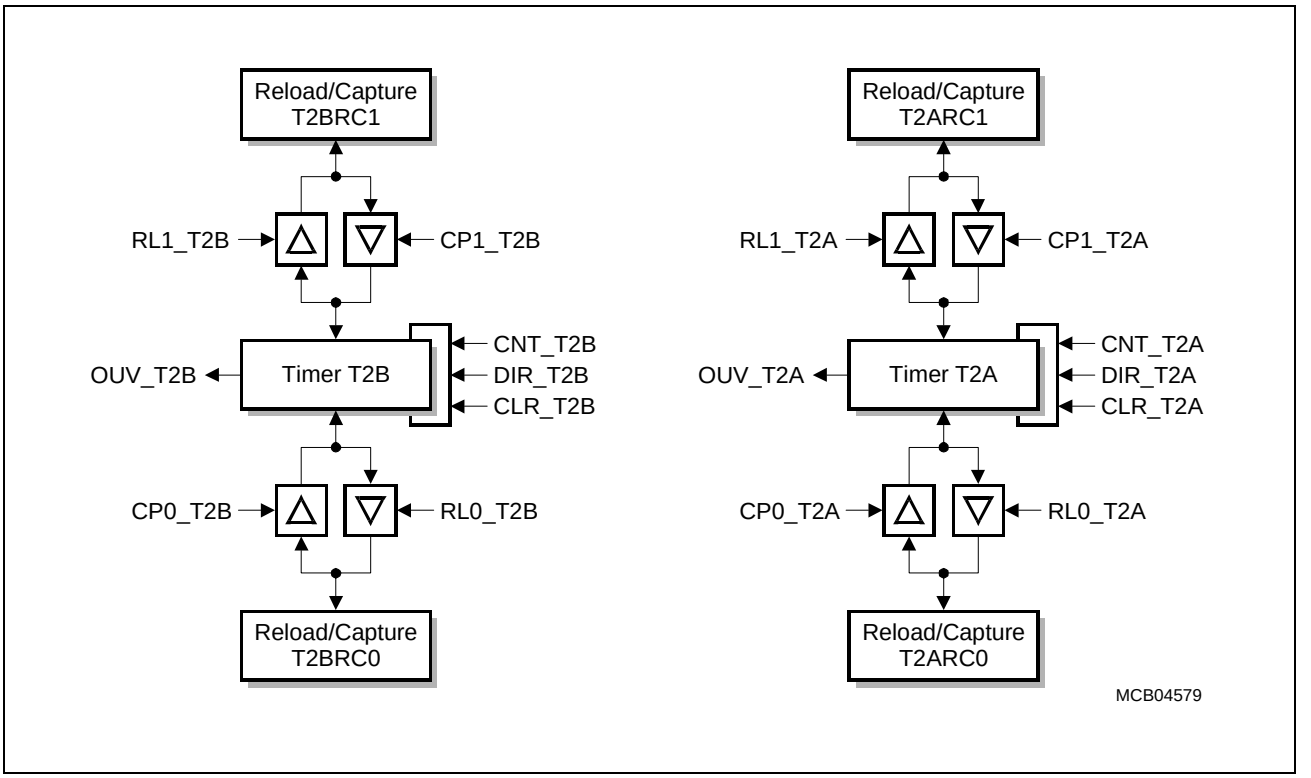


Figure 6-8 Block Diagram of Timer 2 in Split Mode

As shown in **Figure 6-9**, any of the eight GPTU input lines can be assigned to trigger any of the functions performed by T2, including count, start, stop, change direction, clear, reload/capture, and service request. Each of these functions can be selectively triggered on a positive edge, a negative edge, or both edges of the input signal. In addition to these external inputs, signals from Timers T0 and T1 can be used to trigger functions in T2.

All external inputs can be assigned to any of the input functions of T2A and T2B, whether they are split or concatenated. When concatenated, all functions in T2A and T2B are controlled by the T2A mode control block. When split, T2A and T2B are controlled by their individual mode control blocks.

Three registers select the input line and the triggering edge for a specific function. The first register, T2AIS, selects the inputs for either T2 in 32-bit mode or T2A in Split Mode. Register T2BIS does the same for T2B in Split Mode. The third register, T2ES, provides the means to select which edge of the selected external signal causes a trigger of the associated function.

Most of these input signals can be used to generate a service request, independent of whether they are used to trigger Timer T2 functions or not.

Two registers control the mode of operation for the timer and the reload/capture registers. They also provide status information. Register T2CON controls the operation of the timer itself and holds the status information. Register T2RCCON controls the operation of the two reload/capture registers.

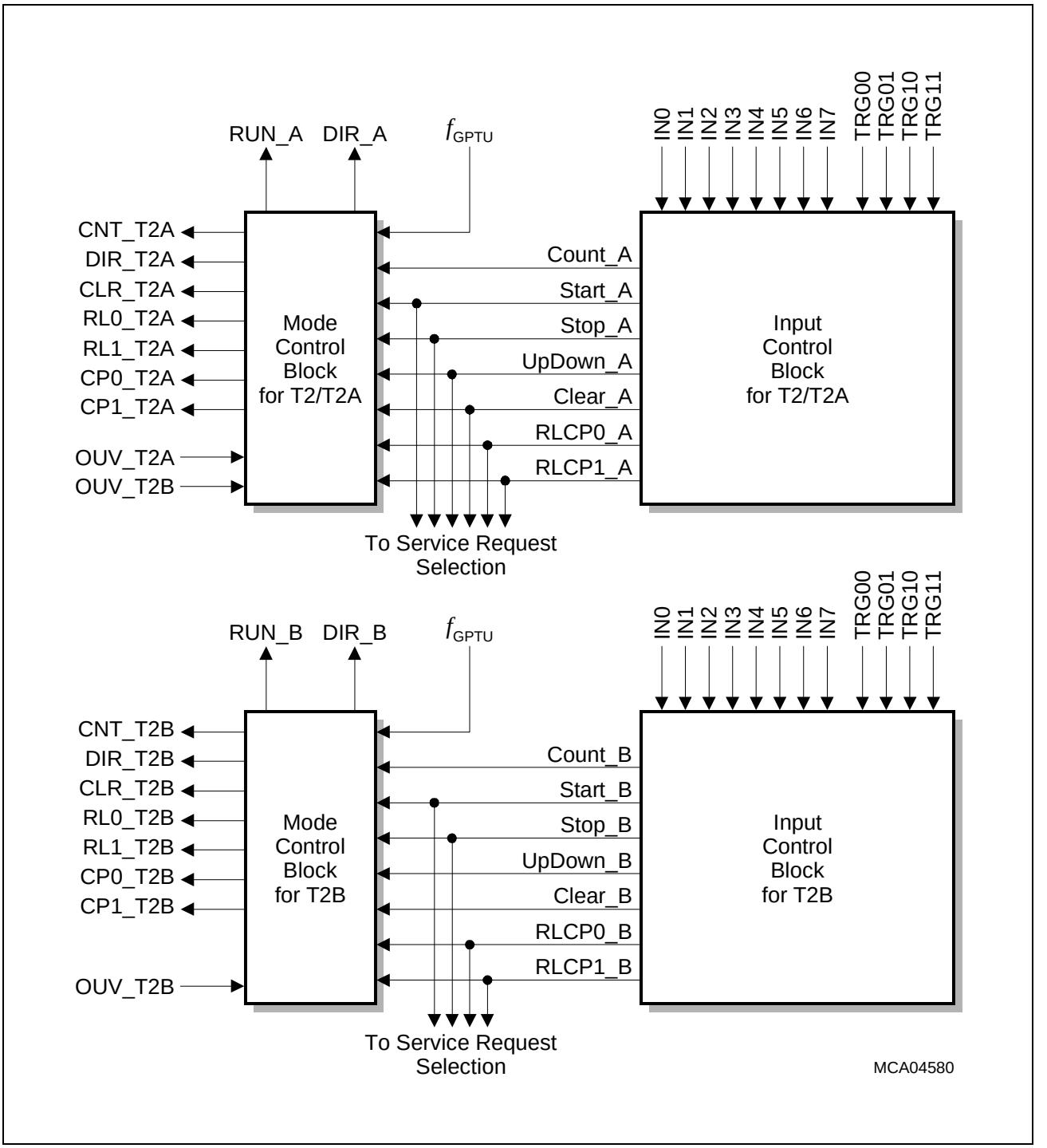


Figure 6-9 Timer 2 Input and Mode Control Blocks

Figure 6-10 and **Figure 6-11** show how T2 control signals are determined. This information is summarized as follows.

- Count control CNT_T2x
  - Clock Source Control, T2CON.T2xCSRC, determines the clocking trigger. Input can be the module clock (f_{GPTU}), or an external trigger source, Count_x. In Quadrature Counter Mode, count input sources are the two inputs, Count_x and UpDown_x.
  - External clocking trigger, Count_x, is determined by T2xIS.T2xICNT. Trigger source can be either an external input, INy, or a trigger signal, TRGxx, from Timer T0 or Timer T1. Bit T2ES.T2xECNT determines the active clock edge.
  - Starting and stopping of the timer can be controlled either by software via setting or clearing the run bit T012RUN.T2xRUN (software modifications of this bit are performed through the run bit set and clear bits, T012RUN.T2*x*SETR and T012RUN.T2xCLRR, respectively), or through the signals Start_x and Stop_x, selected by T2xIS.T2xISTR and T2xIS.T2xISTP, respectively. Any external input INy can be selected for this purpose. T2ES.T2xESTR and T2ES.T2xESTP determine the active clock edges for these sources, respectively.
    Additionally, in one-shot mode, the timer is stopped in response to its own overflow, OUV_T2*x*.
  - The running/stopped status of T2A and T2B can be examined via the T012RUN.T2xRUN status bits.
- Count direction control DIR_T2x
  - Input source control, T2CON.T2xCDIR, selects whether the count direction is up or down, or whether it is determined from an external input.
  - External input selection is controlled by T2xIS.T2xIUD, which selects any of the INy input signals. T2ES.T2xEUD determines the active clock edge. In Quadrature Counter Mode, up/down count information is derived from the two input sources, Count_*x* and UpDown_*x*.
- Clear control CLR_T2x
  - T2CON.T2xCCLR selects whether to clear the timer to 0 on an external event (Clear_*x*), or to clear the timer on capture 0 event (CP0_T2*x*), or to clear timer on capture 1 event (CP1_T2*x*).
  - Selection of the external trigger is determined by T2xIS.T2xICLR, which selects any of the INy input signals. T2ES.T2xECLR determines the active clock edge.
- Reload/capture RL0_T2x, RL1_T2x, and CP0_T2x, CP1_T2x
  - There are two reload/capture registers each in T2A and T2B which can be programmed independently.
  - Controls T2RCCON.T2xMRC0 and T2RCCON.T2xMRC1 determine reload/capture modes. Modes include disabled, capture on external event, reload on overflow or underflow, reload on external event, reload on overflow only, reload on underflow only, reload on external event if count direction is up (if T2CON.T2xDIR = 0), reload on external event if count direction is down (T2CON.T2*x*DIR = 1).

– Selection of external trigger source for RLCP0_x and RLCP1_x is determined by T2*x*IS.T2xIRC0 and T2xIS.T2xIRC1. Trigger source can be either an external input, GPTU*x*_IN*y*, or a trigger signal, TRGxx, from T0 or T1. T2ES.T2xERC0 and T2ES.T2xERC1 determine the active edge of the trigger signal.

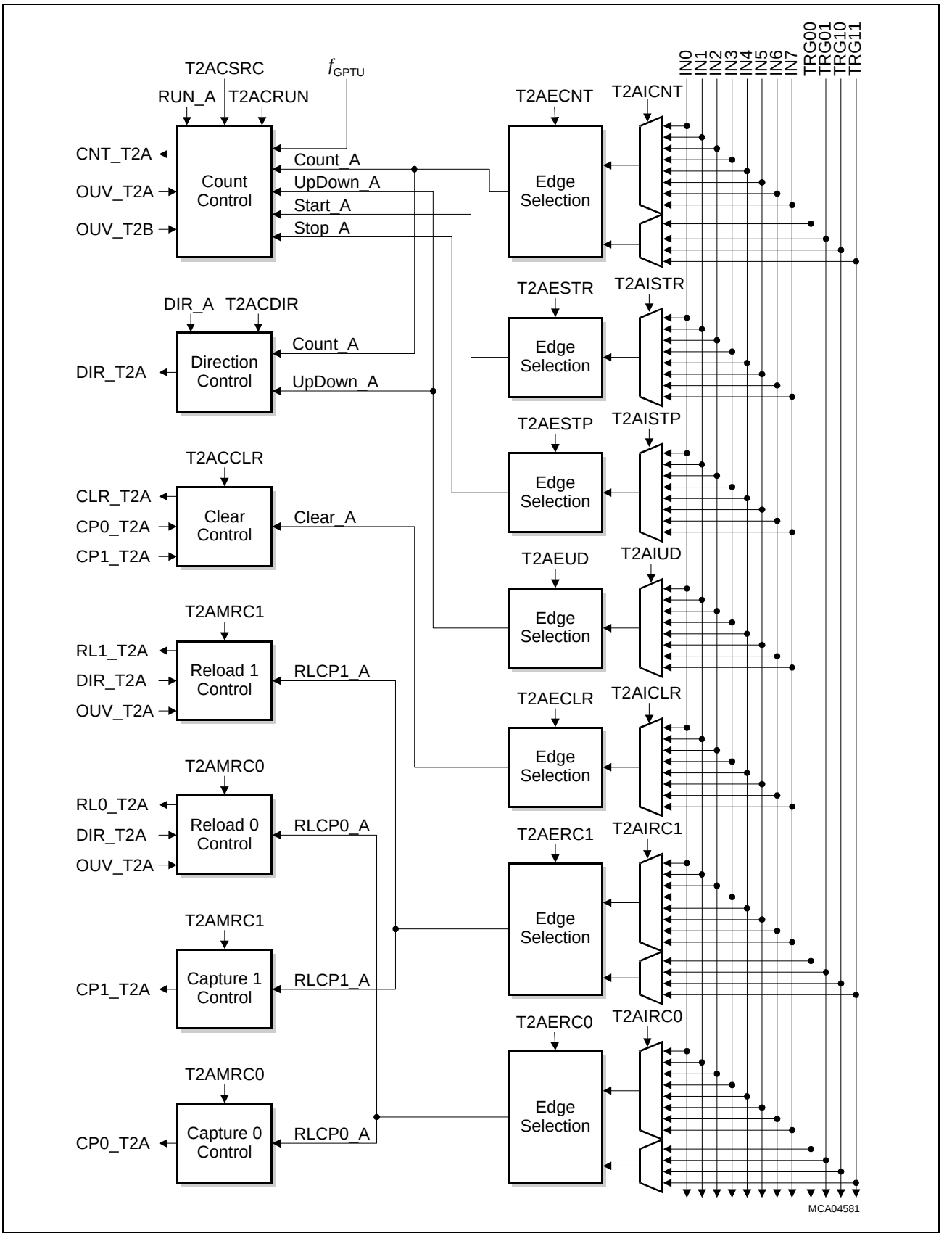


Figure 6-10 Timer T2/T2A Input and Mode Control Details

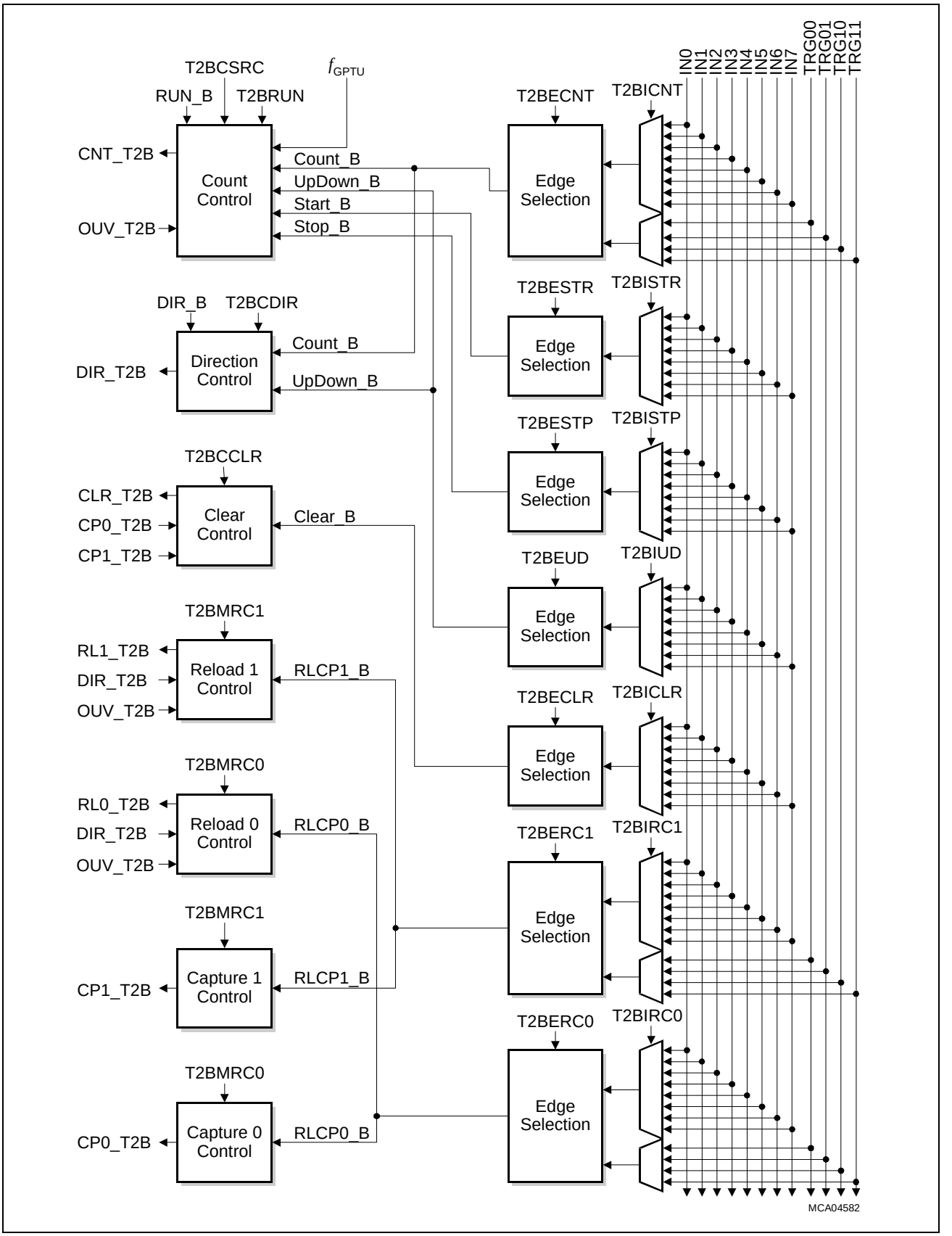


Figure 6-11 Timer T2B Input and Mode Control Details

6.1.4 Quadrature Counting Mode

Position tracking can be performed with Timer T2 in Quadrature Counting Mode, sometimes referred to as incremental or phase encoded interface. The standard way of tracking positions is to use two phase-shifted input signals. These provide the counting and direction information necessary for this task. As shown in **Figure 6-12**, the edges of the signals provide the count signal, while the phase relation between the two signals provides the direction information.

To operate Timer T2 in this mode, the two signals are connected such that they trigger the Count_A/Count_B and the UpDown_A/UpDown_B inputs of the timer block.

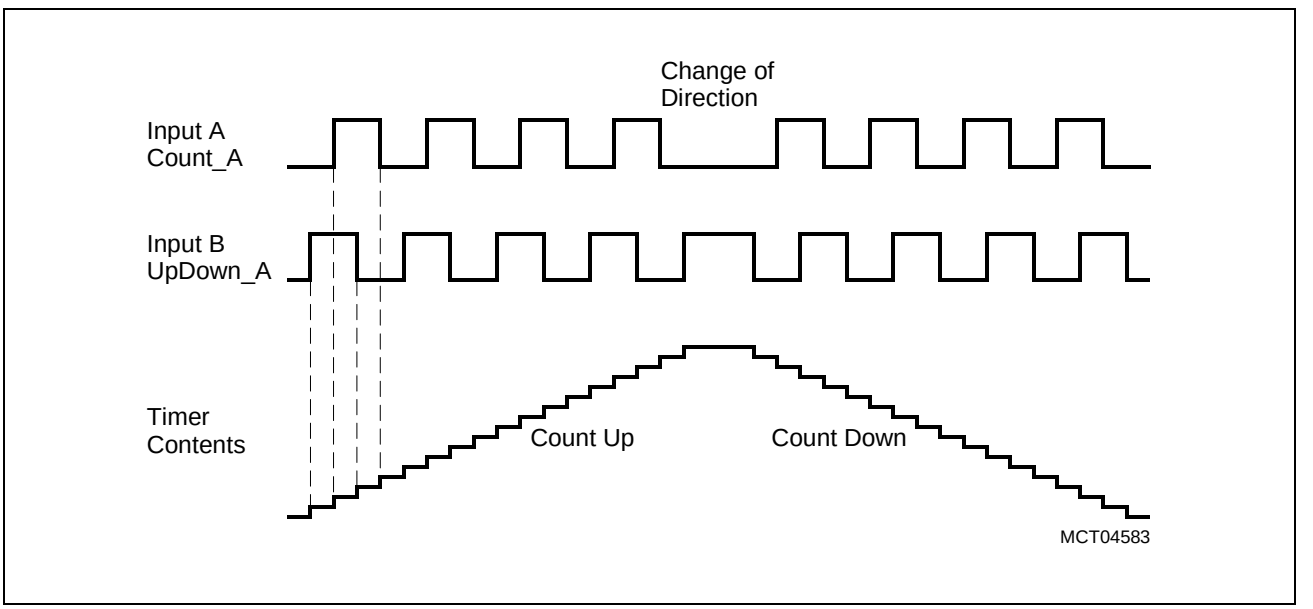


Figure 6-12 Quadrature Counting Operation

6.1.5 Global GPTU Controls

This section describes global control of the GPTU. Global controls are provided for the outputs and interrupt service requests.

6.1.5.1 Output Control

The register OUT has eight bits OUTx (x = 0-7) which store the output signals from the GPTU. The bits in register OUT can also be set or cleared via software. The connection of timer signals to these output bits is determined by eight bit fields in register OSEL, named SOx (x = 0-7). Each output bit in register OUT is connected to a GPTU output line, which connects to the Parallel Ports.

Six signals from Timers T0, T1, and T2 can be selected to generate outputs from the GPTU timers to the Parallel Ports. For each of the eight GPTU output signals, OUT[7:0], the user can select which of the timer signals, OUT00, OUT01, OUT10, OUT11, OUV_T2A, or OUV_T2B, activates the selected output line.

OUT00 and OUT01 can be any T0 timer overflow, OUT10, OUT11 can be any T1 timer overflow. OUV_T2A and OUV_T2B are the timer overflows of T2A and T2B.

Figure 6-13 provides an overview of the output options.

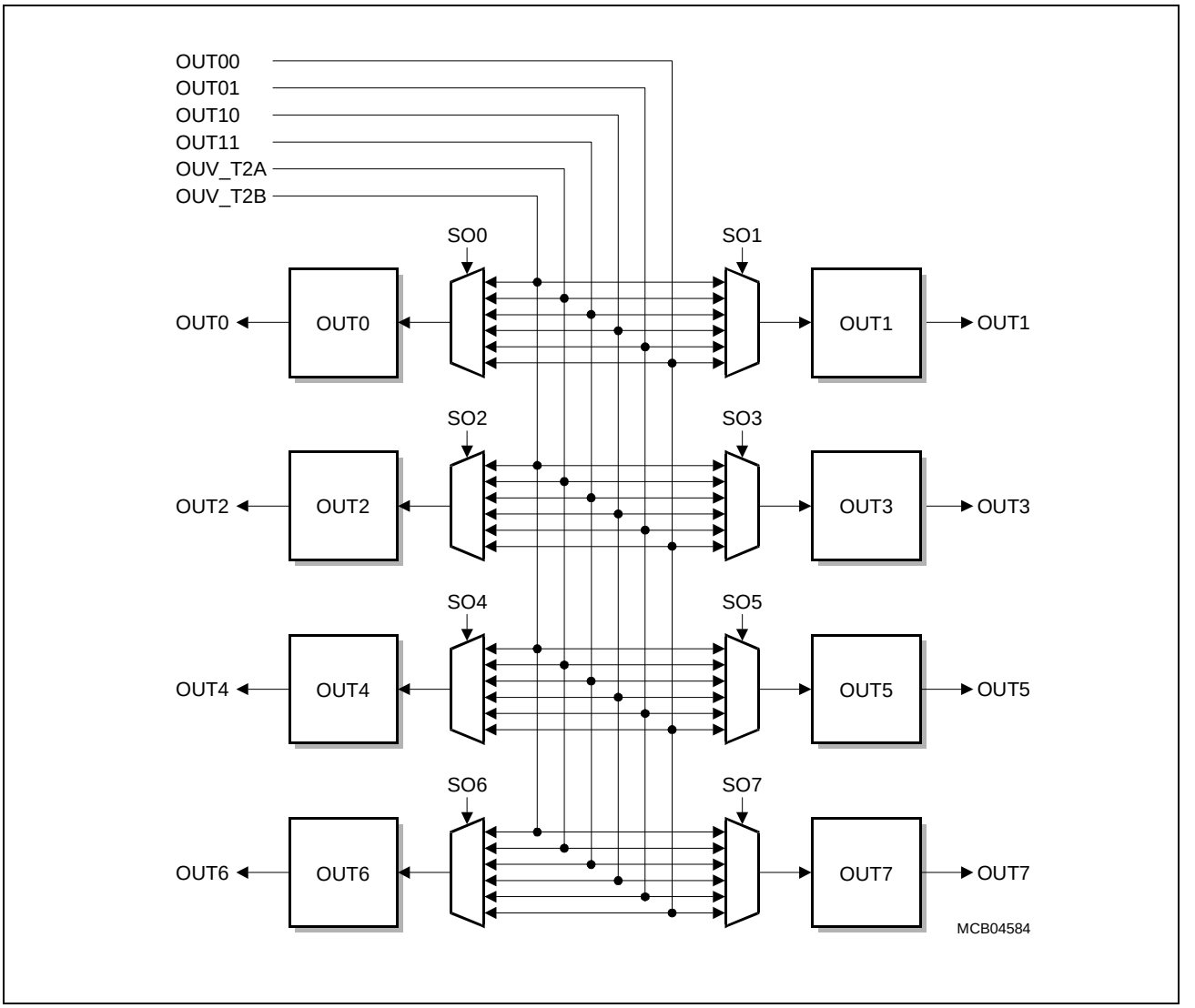


Figure 6-13 Output Control Block Diagram

6.1.5.2 Service Request Control

Sixteen events in T0, T1, and T2 can be selected to generate a service request to either the CPU or the PCP. Eight service request outputs (nodes), SR[7:0], are provided for the GPTU; they can be freely assigned to any of the GPTU events.

Timer T2 events which can be selected include Start_x, Stop_x, UpDown_A, Clear_A (signals UpDown_B and Clear_B are not available for service request generation), RLCP0_x, RLCP1_x, OUV_T2x. Timer T0 overflow events (SR00, SR01) and Timer T1 overflow events (SR10, and SR11) can also be selected. **Figure 6-14** shows these options. Please note that the signals Start_x, Stop_x, UpDown_A, Clear_A, RLCP0_x, and RLCP1_x are the signals coming out of the input selection block, before these lines go into the Timer T2 control logic (see **Figure 6-9**). This has the advantage, that an input line can be used to generate a service request only; it may or may not be used to also trigger a T2 function. In this way, all of the GPTU input lines connected to parallel port pins can be configured as external interrupt inputs.

Because Timers T0 and T1 can generate triggers for Timer T2 signals (such as Count_x, RLCP0_x, and RLCP1_x), it is possible to use these signals for service request generation (whether or not they are also used to trigger functions of T2). This gives additional service requests to Timers T0 and T1.

Because of the flexibility in selecting service requests, more than one service request can be generated by the same event. This option can be used to split the CPU or PCP service routine for an event into several pieces with different priorities.

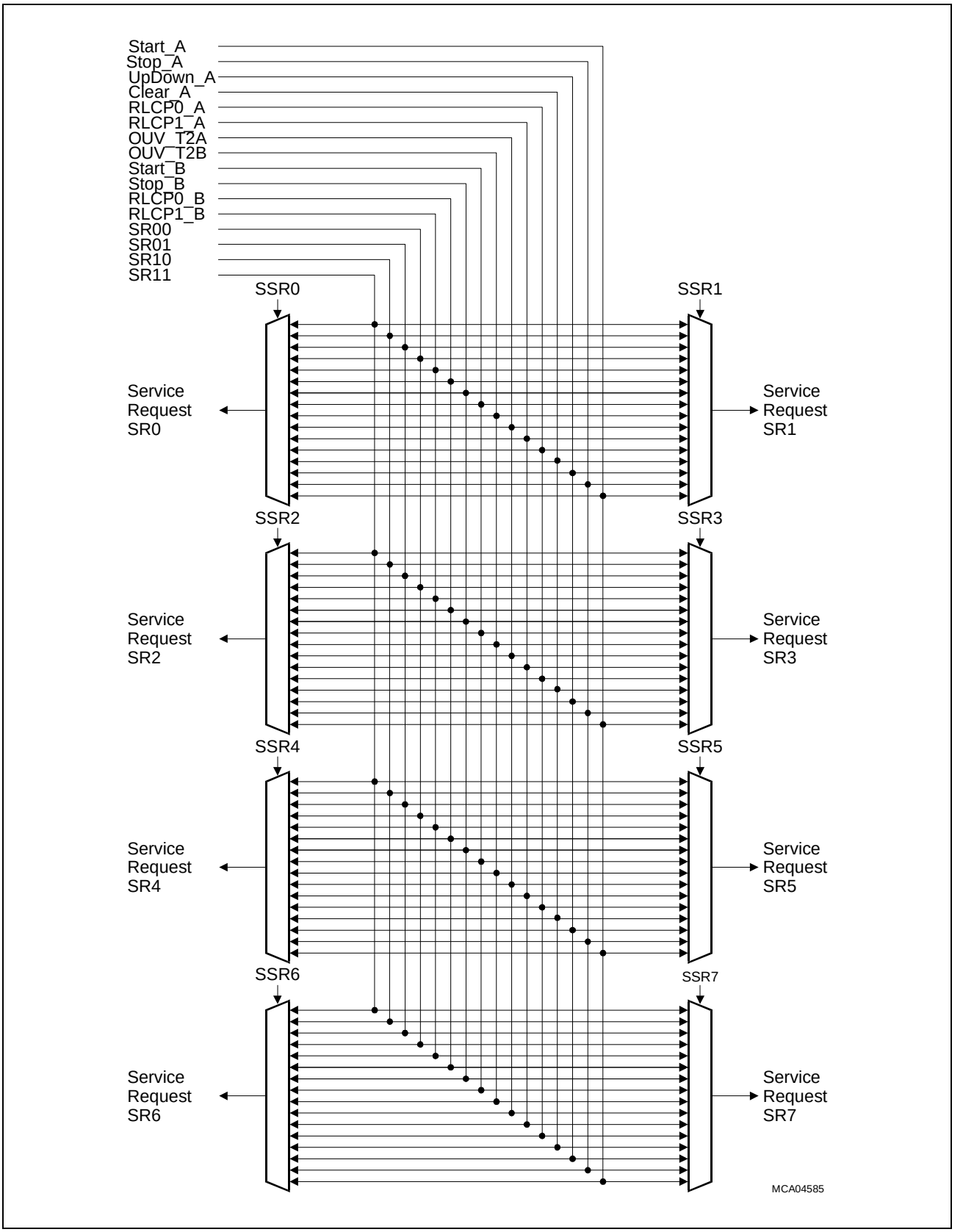


Figure 6-14 Service Request Selection

6.2 GPTU Kernel Registers

Figure 6-15 shows all registers associated with the GPTU Kernel.

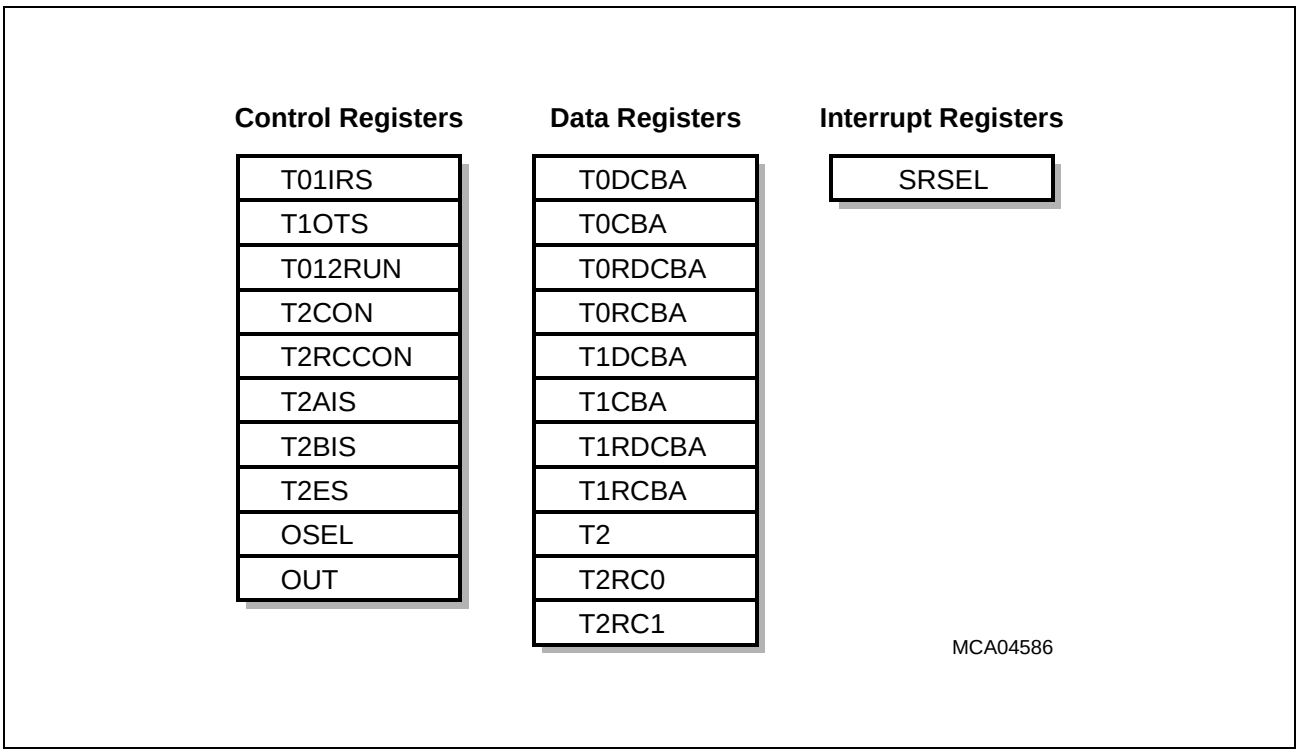


Figure 6-15 GPTU Kernel Registers

Table 6-2 GPTU Kernel Registers

Register Short Name	Register Long Name	Offset Address	Description see
T01IRS	Timer T0 and T1 Input and Reload Source Selection Register	0010_H	Page 6-25
T01OTS	Timer T0 and T1 Output, Trigger and Service Request Register	0014_H	Page 6-28
T2CON	Timer T2 Control Register	0018_H	Page 6-39
T2RCCON	Timer T2 Reload/Capture Control Register	$001C_H$	Page 6-44
T2AIS	Timer T2/T2A Ext. Input Selection Register	0020_H	Page 6-34
T2BIS	Timer T2B External Input Selection Register	0024_H	Page 6-36
T2ES	Timer T2 External Input Edge Selection Reg.	0028_H	Page 6-37
OSEL	Output Source Selection Register	$002C_H$	Page 6-48
OUT	Output Register	0030_H	Page 6-50

Table 6-2 GPTU Kernel Registers (cont'd)

Register Short Name	Register Long Name	Offset Address	Description see
T0DCBA	Timer T0 Count Register (T0D, T0C, T0B, T0A)	0034_H	Page 6-30
T0CBA	Timer T0 Count Register (T0C, T0B, T0A)	0038_H	Page 6-30
T0RDCBA	Timer T0 Reload Register (T0RD, T0RC, T0RB, T0RA)	$003C_H$	Page 6-31
T0RCBA	Timer T0 Reload Register (T0RC, T0RB, T0RA)	0040_H	Page 6-31
T1DCBA	Timer T1 Count Register (T1D, T1C, T1B, T1A)	0044_H	Page 6-32
T1CBA	Timer T1 Count Register (T1C, T1B, T1A)	0048_H	Page 6-32
T1RDCBA	Timer T1 Reload Register (T1RD, T1RC, T1RB, T1RA)	$004C_H$	Page 6-32
T1RCBA	Timer T1 Reload Register (T1RC, T1RB, T1RA)	0050_H	Page 6-33
T2	Timer T2 Count Register	0054_H	Page 6-46
T2RC0	Timer T2 Reload/Capture Register 0	0058_H	Page 6-47
T2RC1	Timer T2 Reload/Capture Register 1	$005C_H$	Page 6-47
T012RUN	Timers T0, T1, T2 Run Control Register	0060_H	Page 6-42
SRSEL	Service Request Source Select Reg.	$00DC_H$	Page 6-52

Note: All GPTU kernel register names described in this section will be referenced in other parts of the TC1775 User's Manual with the module name prefix "GPTU_".

6.2.1 Timer T0/T1 Registers

This section describes the registers related to Timers T0 and T1. Note that register T012RUN is shared between all three timers and is described in **Section 6.2.2.3**.

6.2.1.1 Timer T0/T1 Input & Reload Source Selection Register

The T01IRS register contains the individual controls for the count input and the reload trigger selections for the individual parts of T0 and T1. This register also contains the control for the global input signals CNT0 and CNT1.

T01IRS
Timer T0 and T1 Input and Reload Source Selection Register

Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
T01 IN1		**T01 IN0**		**0**	**0**	**T1 INC**	**T0 INC**	**T1D REL**	**T1C REL**	**T1B REL**	**T1A REL**	**T0D REL**	**T0C REL**	**T0B REL**	**T0A REL**
rw		rw		r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T1D INS		**T1C INS**		**T1B INS**		**T1A INS**		**T0D INS**		**T0C INS**		**T0B INS**		**T0A INS**	
rw		rw		rw		rw		rw		rw		rw		rw	

Field	Bits	Type	Description
T0AINS	[1:0]	rw	**T0A Input Selection** 00 Clock input f_{GPTU} 01 Global input CNT0 10 Global input CNT1 11 Carry input (concatenation)
T0BINS	[3:2]	rw	**T0B Input Selection**; coding as T0AINS
T0CINS	[5:4]	rw	**T0C Input Selection**; coding as T0AINS
T0DINS	[7:6]	rw	**T0D Input Selection**; coding as T0AINS
T1AINS	[9,8]	rw	**T1A Input Selection;** coding as T0AINS
T1BINS	[11:10]	rw	**T1B Input Selection**; coding as T0AINS
T1CINS	[13:12]	rw	**T1C Input Selection**; coding as T0AINS
T1DINS	[15:14]	rw	**T1D Input Selection**; coding as T0AINS

Field	Bits	Type	Description
T0AREL	16	rw	**T0A Reload Source Selection** 0 Reload on overflow of timer T0A 1 Concatenation with T0RB
T0BREL	17	rw	**T0B Reload Source Selection** 0 Reload on overflow of timer T0B 1 Concatenation with T0RC
T0CREL	18	rw	**T0C Reload Source Selection** 0 Reload on overflow of timer T0C 1 Concatenation with T0RD
T0DREL	19	rw	**T0D Reload Source Selection** 0 Reload on overflow of timer T0D 1 Reload on signal T1RA
T1AREL	20	rw	**T1A Reload Source Selection** 0 Reload on overflow of timer T1A 1 Concatenation with T1RB
T1BREL	21	rw	**T1B Reload Source Selection** 0 Reload on overflow of timer T1B 1 Concatenation with T1RC
T1CREL	22	rw	**T1C Reload Source Selection** 0 Reload on overflow of timer T1C 1 Concatenation with T1RD
T1DREL	23	rw	**T1D Reload Source Selection** 0 Reload on overflow of timer T1D 1 Concatenation with T0RA
T0INC	24	rw	**T0 Carry Input Selection** 0 T0A carry in is T0D carry out 1 T0A carry in is T1D carry out
T1INC	25	rw	**T1 Carry Input Selection** 0 T1A carry in is T1D carry out 1 T1A carry in is T0D carry out
0	[27:26]	r	**Reserved**; read as 0; writing to these bit positions has no effect.

Field	Bits	Type	Description
T01IN0	[29:28]	rw	**T0 and T1 Global Input CNT0 Selection** 00 Timer T2A overflow/underflow OUV_T2A 01 Positive edge of IN0 10 Negative edge of IN0 11 Both edges of IN0
T01IN1	[31:30]	rw	**T0 and T1 Global Input CNT1 Selection** 00 Timer T2A overflow/underflow OUV_T2B 01 Positive edge of IN1 10 Negative edge of IN1 11 Both edges of IN1

6.2.1.2 Timer T0/T1 Output, Trigger, and Service Req. Selection Register

T01OTS performs the selections for the output, service request, and trigger signals of the individual parts of both Timers T0 and T1.

T01OTS
Timer T0 and T1 Output, Trigger and Service Request Selection Register
Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0				SSR11		SSR10		STRG11		STRG10		SOUT11		SOUT10	
r				rw		rw		rw		rw		rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				SSR01		SSR00		STRG01		STRG00		SOUT01		SOUT00	
r				rw		rw		rw		rw		rw		rw	

Field	Bits	Type	Description
SOUT00	[1:0]	rw	**T0 Output 0 Source Selection** encoding see **Table 6-3**
SOUT01	[3:2]	rw	**T0 Output 1 Source Selection** encoding see **Table 6-3**
STRG00	[5:4]	rw	**T0 Trigger Output 0 Source Selection** encoding see **Table 6-3**
STRG01	[7:6]	rw	**T0 Trigger Output 1 Source Selection** encoding see **Table 6-3**
SSR00	[9:8]	rw	**T0 Service Request 0 Source Selection** encoding see **Table 6-3**
SSR01	[11:10]	rw	**T0 Service Request 1 Source Selection** encoding see **Table 6-3**
SOUT10	[17:16]	rw	**T1 Output 0 Source Selection** encoding see **Table 6-3**
SOUT11	[19:18]	rw	**T1 Output 1 Source Selection** encoding see **Table 6-3**
STRG10	[21:20]	rw	**T1 Trigger Output 0 Source Selection** encoding see **Table 6-3**

Field	Bits	Type	Description
STRG11	[23:22]	rw	**T1 Trigger Output 1 Source Selection** encoding see **Table 6-3**
SSR10	25, 24]	rw	**T1 Service Request 0 Source Selection** encoding see **Table 6-3**
SSR11	27, 26]	rw	**T1 Service Request 1 Source Selection** encoding see **Table 6-3**
0	[15:12] [31:28]	r	**Reserved**; read as 0; writing to these bit positions has no effect.

Table 6-3 T0/T1 Overflow Source Selection (x, y = 0, 1)

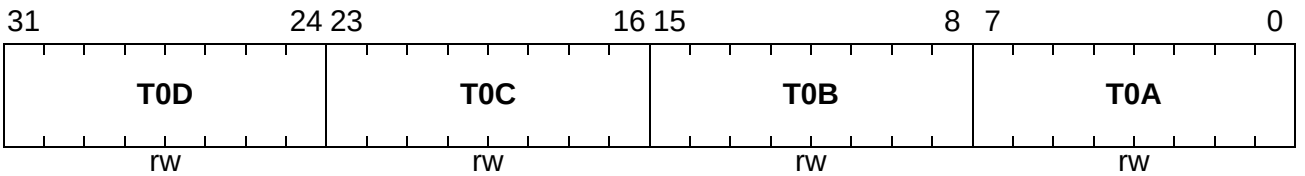
Service Request Selection SSRxy	Trigger Output Selection STRGxy	Output Source Selection SOUTxy	Selected Overflow Signal
00	00	00	TxA overflow
01	01	01	TxB overflow
10	10	10	TxC overflow
11	11	11	TxD overflow

6.2.1.3 Timer T0 and T1 Count and Reload Registers

Timer T0 Count Register T0DCBA (T0D, T0C, T0B, T0A)

This register provides read/write access to all four parts of Timer T0.

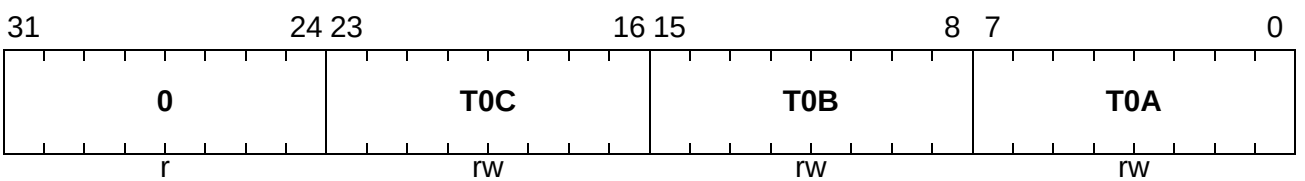
T0DCBA
Timer T0 Count Register (T0D, T0C, T0B, T0A) **Reset Value: 0000 0000$_H$**



Timer T0 Count Register T0CBA (T0C, T0B, T0A)

This register provides read/write access to the lower three parts of Timer T0. The upper byte is always read as 0; writes to it have no effect and are not stored. This register needs to be used if parts A, B, and C of Timer T0 are configured as a 24-bit timer. Part D of Timer T0 will not be affected when writing to this register.

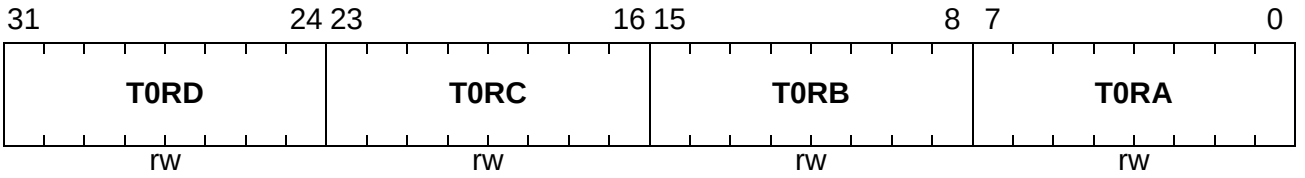
T0CBA
Timer T0 Count Register (T0C, T0B, T0A) **Reset Value: 0000 0000$_H$**



Timer T0 Reload Register T0RDCBA (T0RD, T0RC, T0RB, T0RA)

This register provides read/write access to all four parts of the reload register of Timer T0.

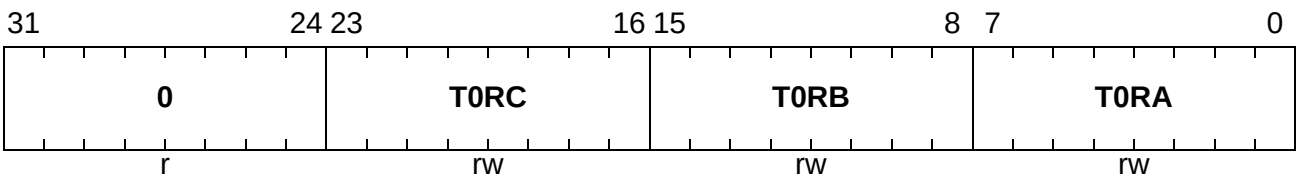
T0RDCBA
Timer T0 Reload Register (T0RD, T0RC, T0RB, T0RA) **Reset Value: 0000 0000$_H$**



T0RCBA, Timer T0 Reload Register (T0RC, T0RB, T0RA)

This register provides read/write access to the lower three parts of the reload register of Timer T0. The upper byte is always read as 0; writes to it have no effect and are not stored. This reload register needs to be used if parts A, B, and C of Timer T0 are configured as a 24-bit timer. Part D of the reload register will not be affected when writing to this register.

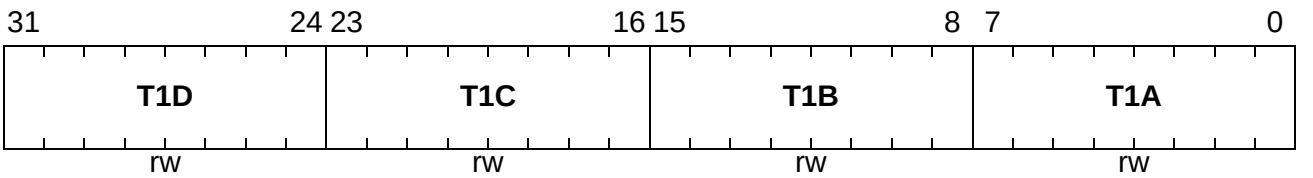
T0RCBA
Timer T0 Reload Register (T0RC, T0RB, T0RA) **Reset Value: 0000 0000$_H$**



Timer T1 Count Register T1DCBA (T1D, T1C, T1B, T1A)

This register provides read/write access to all four parts of Timer T1.

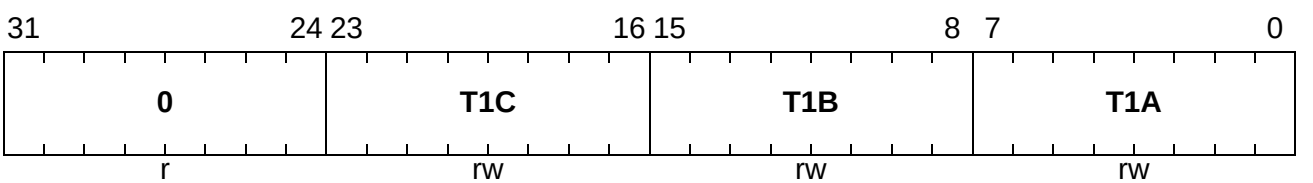
T1DCBA
Timer T1 Count Register (T1D, T1C, T1B, T1A)Reset Value: 0000 0000$_H$



Timer T1 Count Register T1CBA (T1C, T1B, T1A)

This register provides read/write access to the lower three parts of Timer T1. The upper byte is always read as 0; writes to it have no effect and are not stored. This register needs to be used if parts A, B, and C of Timer T1 are configured as a 24-bit timer. Part D of Timer T1 will not be affected when writing to this register.

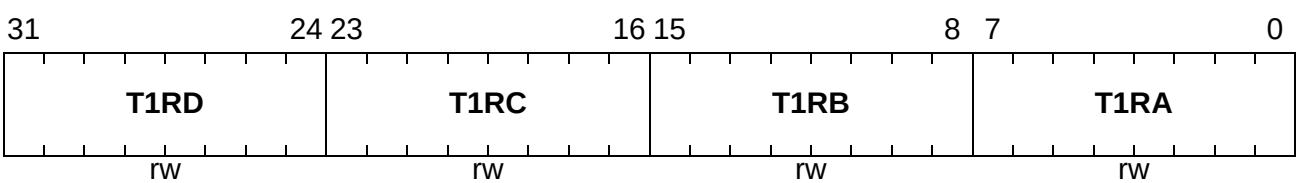
T1CBA
Timer T1 Count Register (T1C, T1B, T1A) **Reset Value: 0000 0000$_H$**



Timer T1 Reload Register T1RDCBA (T1RD, T1RC, T1RB, T1RA)

This register provides read/write access to all four parts of the reload register of Timer T1.

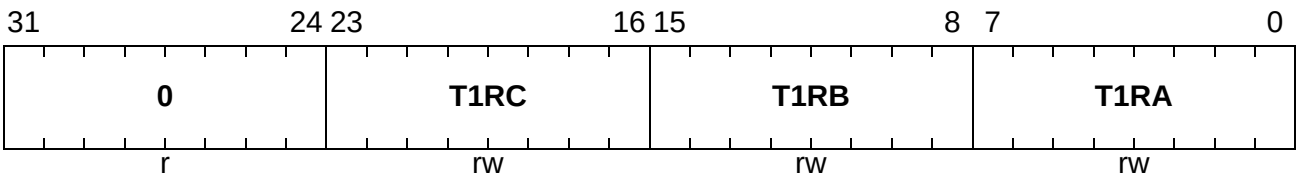
T1RDCBA
Timer T1 Reload Register (T1RD, T1RC, T1RB, T1RA) **Reset Value: 0000 0000$_H$**



Timer T1 Reload Register T1RCBA (T1RC, T1RB, T1RA)

This register provides read/write access to the lower three parts of the reload register of Timer T1. The upper byte is always read as 0; writes to it have no effect and are not stored. This reload register needs to be used if parts A, B, and C of Timer T1 are configured as a 24-bit timer. Part D of the reload register will not be affected when writing to this register.

T1RCBA
Timer T1 Reload Register (T1RC, T1RB, T1RA) **Reset Value: 0000 0000$_H$**



6.2.2 Timer T2 Registers

This section describes the Timer T2 registers.

6.2.2.1 Input Control Registers

Three registers select the input line and the triggering edge for a specific function. The first register, T2AIS, selects the inputs for either Timer T2 in 32-bit mode or Timer T2A in Split Mode. Register T2BIS does the same for Timer T2B in Split Mode. The third register, T2ES, provides the means to select which edge of the selected external signal causes a trigger of the associated function.

Most of these input signals can be used to generate a service request, independent of whether they are used to trigger Timer T2 functions or not.

Timer T2/T2A External Input Selection Register T2AIS

The T2AIS register selects which of the eight external inputs or trigger events from Timer T0/T1 is to be used for the various input functions for Timer T2A. It controls the input selection for Timer T2A in Split Mode and for the entire Timer T2 in 32-bit mode.

T2AIS
Timer T2/T2A External Input Selection Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0					T2AIRC1			0	T2AIRC0			0	T2AICLR		
r					rw			r	rw			r	rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	T2AIUD			0	T2AISTP			0	T2AISTR			0	T2AICNT		
r	rw			r	rw			r	rw			r	rw		

Field	Bits	Type	Description
T2AICNT	[2:0]	rw	**Timer T2A External Count Input Selection** encoding see **Table 6-4**
T2AISTR	[6:4]	rw	**Timer T2A External Start Input Selection** encoding see **Table 6-4**
T2AISTP	[10:8]	rw	**Timer T2A External Stop Input Selection** encoding see **Table 6-4**
T2AIUD	[14:12]	rw	**Timer T2A External Up/Down Input Selection** encoding see **Table 6-4**

Field	Bits	Type	Description
T2AICLR	[18:16]	rw	**Timer T2A External Clear Input Selection** encoding see **Table 6-4**
T2AIRC0	[22:20]	rw	**Timer T2A External Reload/Capture 0 Input** encoding see **Table 6-4**
T2AIRC1	[26:24]	rw	**Timer T2A External Reload/Capture 1 Input** encoding see **Table 6-4**
0	3, 7, 11,15, 19, 23, [31:27]	r	**Reserved**; read as 0; writing to these bit positions has no effect.

Table 6-4 T2 Input Source Selection (x, y = 0, 1)

Value	Selected External Input	In Parallel Selected Input for T2AICNT, T2AIRC1, and T2AIRC0; and T2BICNT, T2BIRC1, and T2BIRC0
000	Input IN0	T0/T1 Trigger Input Signal TRG00
001	Input IN1	T0/T1 Trigger Input Signal TRG01
010	Input IN2	T0/T1 Trigger Input Signal TRG10
011	Input IN3	T0/T1 Trigger Input Signal TRG11
100	Input IN4	T0/T1 Trigger Input Signal TRG00
101	Input IN5	T0/T1 Trigger Input Signal TRG01
110	Input IN6	T0/T1 Trigger Input Signal TRG10
111	Input IN7	T0/T1 Trigger Input Signal TRG11

*Note: Selection between the input lines and TRGxy is done via the edge selection control (register T2ES, encoding see **Table 6-5**).*

Timer T2B External Input Selection Register T2BIS

The T2BIS register selects which of the external pins or trigger events from Timer T0/T1 is to be used for the various input functions for Timer T2B. This register is used only to select the inputs for Timer T2B in Split Mode; it is inactive in 32-bit mode. The selection is the same as for Timer T2A.

T2BIS
Timer T2B External Input Selection Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0					T2BIRC1			0	T2BIRC0			0	T2BICLR		
r					rw			r	rw			r	rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	T2BIUD			0	T2BISTP			0	T2BISTR			0	T2BICNT		
r	rw			r	rw			r	rw			r	rw		

Field	Bits	Type	Description
T2BICNT	[2:0]	rw	**Timer T2B External Count Input Selection** encoding see **Table 6-4**
T2BISTR	[6:4]	rw	**Timer T2B External Start Input Selection** encoding see **Table 6-4**
T2BISTP	[10:8]	rw	**Timer T2B External Stop Input Selection** encoding see **Table 6-4**
T2BIUD	[14:12]	rw	**Timer T2B External Up/Down Input Selection** encoding see **Table 6-4**
T2BICLR	[18:16]	rw	**Timer T2B External Clear Input Selection** encoding see **Table 6-4**
T2BIRC0	[22:20]	rw	**Timer T2B External Reload/Capture 0 Input** encoding see **Table 6-4**
T2BIRC1	[26:24]	rw	**Timer T2B External Reload/Capture 1 Input** encoding see **Table 6-4**
0	3, 7, 11,15, 19, 23, [31:27]	r	**Reserved**; read as 0; writing to these bit positions has no effect.

Timer T2 External Input Edge Selection Register T2ES

This register selects the active edge of the external pin input for both Timer T2A and Timer T2B. **Table 6-5** lists the truth table for the edge selection bit fields.

T2ES
Timer 2 External Input Edge Selection Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0		T2BERC1		T2BERC0		T2BECLR		T2BEUD		T2BESTP		T2BESTR		T2BECNT	
r		rw		rw		rw		rw		rw		rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		T2AERC1		T2AERC0		T2AECLR		T2AEUD		T2AESTP		T2AESTR		T2AECNT	
r		rw		rw		rw		rw		rw		rw		rw	

Field	Bits	Type	Description
T2AECNT	[1:0]	rw	**Timer T2A External Count Input Active Edge Selection** (encoding see **Table 6-5**)
T2AESTR	[3:2]	rw	**Timer T2A External Start Input Active Edge Selection** (encoding see **Table 6-5**)
T2AESTP	[5:4]	rw	**Timer T2A External Stop Input Active Edge Selection** (encoding see **Table 6-5**)
T2AEUD	[7:6]	rw	**Timer T2A External Up/Down Input Active Edge Selection** (encoding see **Table 6-5**)
T2AECLR	[9:8]	rw	**Timer T2A External Clear Input Active Edge Selection** (encoding see **Table 6-5**)
T2AERC0	[11:10]	rw	**Timer T2A External Reload/Capture 0 Input Active Edge Selection** (encoding see **Table 6-5**)
T2AERC1	[13:12]	rw	**Timer T2A External Reload/Capture 1 Input Active Edge Selection** (encoding see **Table 6-5**)
T2BECNT	[17:16]	rw	**Timer T2B External Count Input Active Edge Selection** (encoding see **Table 6-5**)
T2BESTR	[19:18]	rw	**Timer T2B External Start Input Active Edge Selection** (encoding see **Table 6-5**)
T2BESTP	[21:20]	rw	**Timer T2B External Stop Input Active Edge Selection** (encoding see **Table 6-5**)

Field	Bits	Type	Description
T2BEUD	[23:22]	rw	**Timer T2B External Up/Down Input Active Edge Selection** (encoding see **Table 6-5**)
T2BECLR	[25:24]	rw	**Timer T2B External Clear Input Active Edge Selection** (encoding see **Table 6-5**)
T2BERC0	[27:26]	rw	**Timer T2B External Reload/Capture 0 Input Active Edge Selection** (encoding see **Table 6-5**)
T2BERC1	[29:28]	rw	**Timer T2B External Reload/Capture 1 Input Active Edge Selection** (encoding see **Table 6-5**)
0	[15:14], [31:30]	r	**Reserved**; read as 0; writing to these bit positions has no effect.

Table 6-5 T2 Input Source Active Edge Selection

Value	Selected Active Edge	Selected Active Input for T2AECNT, T2AERC1, T2AIRC0; and T2BECNT, T2BERC1, and T2BERC0
00	None.	Input is connected to T0/T1 Trigger Input Signal TRGxy as selected in T2xIS
01	Positive edge	–
10	Negative edge	–
11	Both edges	–

6.2.2.2 Mode Control and Status Register

Two registers control the mode of operation for the timer and the reload/capture registers. They also provide status information. The first register, T2CON, controls the operation of the timer itself and holds the status information, while the second register, T2RCCON, controls the operation of the two reload/capture registers.

The T2CON register controls the operating mode of Timer T2. The control bits and functions are the same for Timer T2A and Timer T2B.

T2CON
Timer 2 Mode Control and Status Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0			T2BDIR	0			T2BCOS	T2BCOV		T2BCCLR		T2BCDIR		T2BCSRC	
r			rw	r			rw	rw		rw		rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T2SPLIT	0		T2ADIR	0			T2ACOS	T2ACOV		T2ACCLR		T2ACDIR		T2ACSRC	
rw	r		rw	r			rw	rw		rw		rw		rw	

Field	Bits	Type	Description
T2ACSRC	[1:0]	rw	**Timer T2A Count Input Source Control** encoding see **Table 6-9**
T2ACDIR	[3:2]	rw	**Timer T2A Direction Control** encoding see **Table 6-8**
T2ACCLR	[5:4]	rw	**Timer T2A Clear Control** encoding see **Table 6-7**
T2ACOV	[7:6]	rw	**Timer T2A Overflow/Underflow Generation Control** (encoding see **Table 6-6**)
T2ACOS	8	rw	**Timer T2A One-Shot Control** 0 T2A continues to run after overflow or underflow 1 T2A stops after the first overflow or underflow
T2ADIR	12	rw	**Timer T2A Direction Status Flag** 0 T2A Direction is up-counting 1 T2A Direction is down-counting

Field	Bits	Type	Description
T2SPLIT	15	rw	**Timer T2 Split Control** 0 Timer T2 operates as one 32-bit timer, controlled via T2A controls 1 Timer T2 operates as two independent 16-bit timers T2A and T2B
T2BCSRC	[17:16]	rw	**Timer T2B Count Input Source Control** encoding see **Table 6-9**
T2BCDIR	[19:18]	rw	**Timer T2B Direction Control** encoding see **Table 6-8**
T2BCCLR	[21:20]	rw	**Timer T2B Clear Control** encoding see **Table 6-7**
T2BCOV	[23:22]	rw	**Timer T2B Overflow/Underflow Generation Control** (encoding see **Table 6-6**)
T2BCOS	24	rw	**Timer T2B One-Shot Control** 0 T2B continues to run after overflow or underflow 1 T2B stops after the first overflow or underflow
T2BDIR	28	rw	**Timer T2B Direction Status Flag** 0 T2B direction is up-counting 1 T2B direction is down-counting
0	[11:9], 14,13, [27:25], [31:29]	r	**Reserved**; read as 0; writing to these bit positions has no effect.

Table 6-6 T2 Overflow/Underflow Generation Control

T2BCOV T2ACOV	Selected Function
00	Overflow is generated for FF..FF$_H$ -> 00..00$_H$; Underflow is generated for 00..00$_H$ -> FF..FF$_H$
01	Overflow is generated for FF..F**E**$_H$ -> FF..FF$_H$,; underflow is generated for 00..00$_H$ -> FF..FF$_H$
10	Overflow is generated for FF..FF$_H$ -> 00..00$_H$; underflow is generated for 00..0**1**$_H$ -> 00..00$_H$
11	Overflow is generated for FF..F**E**$_H$ -> FF..FF$_H$; Underflow is generated for 00..0**1**$_H$ -> 00..00$_H$

Table 6-7 T2 Clear Control

T2BCCLR T2ACCLR[1)]	Selected Function
00	Clear timer to $00..00_H$ on external event (Clear_B/Clear_A)
01	Clear timer on capture 0 event (CP0_T2B/CP0_T2A)
10	Clear timer on capture 1 event (CP1_T2B/CP1_T2A)
11	Reserved. Do not use this combination.

1) In Clear-on-Capture mode, the timer contents are first captured, then the timer is cleared.

Table 6-8 T2 Direction Control

T2BCDIR T2ACDIR	$Selected\ Function_B$[1)]
00	Count direction is count up (software controlled).
01	Count direction is count down (software controlled).
10[2)]	Count direction controlled through external signal (UpDown_B / UpDown_A). Count up if external signal is 1, else count down.
11[2)]	Count direction controlled through external signal (UpDown_B / UpDown_A). Count down if external signal is 1, else count up.

1) If Quadrature Counting is selected, the count direction is controlled through the relation of the two signals Count_A/B and Up/Down A/B; the bit fields T2ACDIR/T2BCDIR have no effect in this case.

2) The last two options have an extra line going from the input selection to the direction control representing the state of the input (not shown in the diagrams). The edge selection has no effect on the direction control; however, it can be used to generate a service request (UpDown_A only).

Table 6-9 T2 Count Input Source Control

T2BCSRC T2ACSRC	Selected Function
00	Count input source is the module clock f_{GPTU}.
01	Count input source is external count input Count_x.
10	Quadrature Counter Mode. Count input sources are the two inputs Count_x and UpDown_x.
11	Reserved. Do not use this combination.

6.2.2.3 Timer T0/T1/T2 Run Control Register

The run control bits of the individual parts of timers T0, T1, and T2 are all contained in register T012RUN. This register allows synchronous starting or stopping of several or all timers with one instruction.

T012RUN
Timer T0, T1, and T2 Run Control Register **Reset Value: 0000 0000$_H$**

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	T2B CLR R	T2B SET R	T2B RUN	0	T2A CLR R	T2A SET R	T2A RUN	T1D RUN	T1C RUN	T1B RUN	T1A RUN	T0D RUN	T0C RUN	T0B RUN	T0A RUN
r	w	w	rh	r	w	w	rh	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
T0ARUN	0	rw	**Timer T0A Run Control** 0 Stop T0A 1 Start T0A
T0BRUN	1	rw	**Timer T0B Run Control** 0 Stop T0B 1 Start T0B
T0CRUN	2	rw	**Timer T0C Run Control** 0 Stop T0C 1 Start T0C
T0DRUN	3	rw	**Timer T0D Run Control** 0 Stop T0D 1 Start T0D
T1ARUN	4	rw	**Timer T1A Run Control** 0 Stop T1A 1 Start T1A
T1BRUN	5	rw	**Timer T1B Run Control** 0 Stop T1B 1 Start T1B

Field	Bits	Type	Description
T1CRUN	6	rw	**Timer T1C Run Control** 0 Stop T1C 1 Start T1C
T1DRUN	7	rw	**Timer T1D Run Control** 0 Stop T1D 1 Start T1D
T2ARUN	8	rh	**Timer T2A Run Status Flag** 0 T2A is stopped 1 T2A is running This bit indicates the running/stopped status of Timer T2A. This status bit can be directly set or reset by hardware depending on the selections and external events causing a start or a stop of the timer. It can only be affected by software through the set and clear bits T2ASETR and T2ACLRR, respectively. Writing directly to this bit via software has no effect.
T2ASETR	9	w	**Timer T2A Run Set Bit** Writing a 1 to this bit causes the run bit T2ARUN to be set to 1, thus starting Timer T2A. Possible hardware modifications of T2ARUN that occurred during read-modify-write instructions (for example, bit set, bit clear instructions) are lost; the software modification has priority. The value written to T2ASETR is not stored. Writing a 0 to this bit has no effect. This bit always returns 0 when read. If both T2ASETR and T2ACLRR are set, T2ARUN is not affected.
T2ACLRR	10	w	**Timer T2A Run Clear Bit** Writing a 1 to this bit causes the run bit T2ARUN to be cleared, thus stopping timer T2A. Possible hardware modifications of T2ARUN that occurred during read-modify-write instructions (for example, bit set, bit clear instructions) are lost; the software modification has priority. The value written to T2ACLRR is not stored. Writing a 0 to this bit has no effect. This bit always returns 0 when read. If both T2ASETR and T2ACLRR are set, T2ARUN is not affected.

Field	Bits	Type	Description
T2BRUN	12	rh	**Timer T2B Run Status Flag** 0 T2B is stopped 1 T2B is running More details see description for T2ARUN.
T2BSETR	13	w	**Timer T2B Run Set Bit** More details see description for T2ASETR.
T2BCLRR	14	w	**Timer T2B Run Clear Bit** More details see description for T2ACLRR.
0	11, [31:15]	r	**Reserved**; read as 0; writing to these bit positions has no effect.

6.2.2.4 T2 Reload/Capture Mode Control Register

This register selects the reload/capture mode operation for the reload/capture registers T2ARC0, T2ARC1, T2BRC0, and T2BRC1.

T2RCCON
Timer 2 Reload/Capture Mode Control Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0									**T2BMRC1**			**0**	**T2BMRC0**		
r									rw			r	rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0									**T2AMRC1**			**0**	**T2AMRC0**		
r									rw			r	rw		

Field	Bits	Type	Description
T2AMRC0	[2:0]	rw	Timer T2A Reload/Capture 0 Mode Control encoding see **Table 6-10**
T2AMRC1	[6:4]	rw	Timer T2A Reload/Capture 1 Mode Control encoding see **Table 6-10**

Field	Bits	Type	Description
T2BMRC0	[18:16]	rw	Timer T2B Reload/Capture 0 Mode Control encoding see **Table 6-10**
T2BMRC1	[22:20]	rw	Timer T2B Reload/Capture 1 Mode Control encoding see **Table 6-10**
0	3, 19, [15:7], [31:23]	r	**Reserved**; read as 0; writing to these bit positions has no effect.

Table 6-10 T2 Capture/Reload Mode Selection

T2AMRCx T2BMRCx	Selected Operation for T2ARC0/T2BRC0	Selected Operation for T2ARC1/T2BRC1
000	Disabled	
001	Reserved. Do not use this combination.	
010	Reserved. Do not use this combination.	
011	Capture on external event	
100	Reload on overflow or underflow	
101	Reload on external event	
110	Reload on overflow only	Reload on underflow only
111	Reload on external event if count direction is up (if T2ADIR/T2BDIR = 0)	Reload on external event if count direction is down (T2ADIR/T2BDIR = 1)

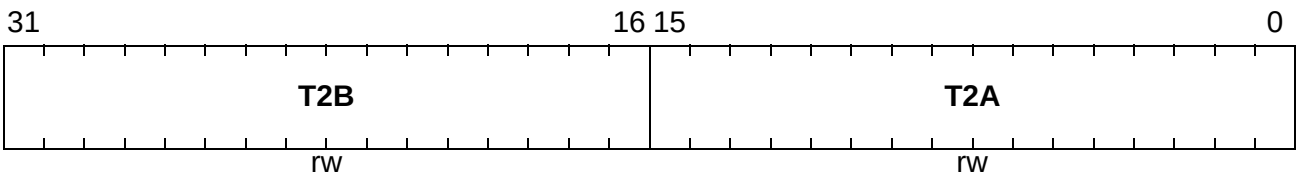
Note: If a capture event for one register and a reload event for the other register occur at the same time, the timer contents are captured first; then, the timer is reloaded. If both reload/capture registers are set up for reload and the trigger events occur at the same time for both, only the reload from the higher numbered register (T2ARC1/T2BRC1) is performed.

6.2.2.5 Timer T2 Count and Reload/Capture Registers

Timer T2 Count Register

Register T2 holds the actual count value of Timer T2. In Split Mode, the lower half-word of this register represents the contents of Timer T2A, while the upper half-word represents the contents of Timer T2B. Proper load/store instructions must be used depending on whether the timer is operated in full 32-bit or in Split Mode.

T2
Timer T2 Count Register **Reset Value: 0000 0000$_H$**

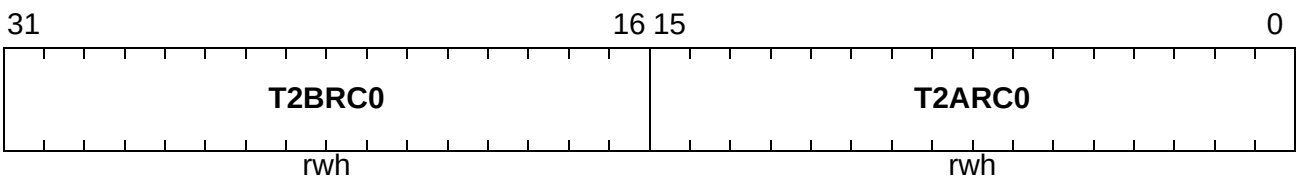


Field	Bits	Type	Description
T2A	[15:0]	rwh	**T2A Contents** (in Split Mode)
T2B	[31:16]	rwh	**T2B Contents** (in Split Mode)

Timer T2 Reload/Capture Registers

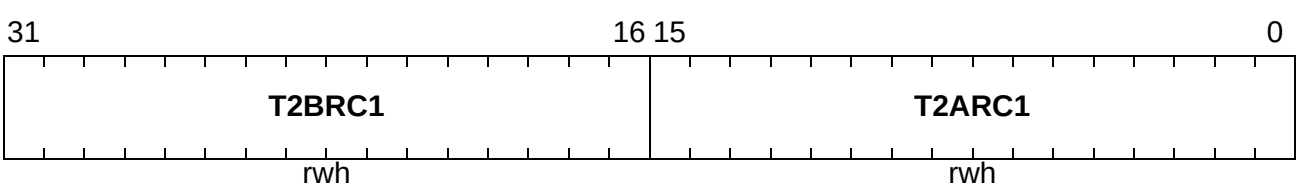
The two reload/capture values for Timer T2 are held in registers T2RC0 and T2RC1, respectively. In Split Mode, the lower half-word of these registers represent the respective Timer T2A reload/capture values (T2ARC0, T2ARC1), while the upper half-word is used for the Timer T2B reload/capture values (T2BRC0, T2BRC1). The same access mechanisms apply here as for the timer count register Timer T2.

T2RC0
Timer T2 Reload/Capture Register 0 **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
T2ARC0	[15.0]	rwh	**T2A Reload/Capture Value** (in Split Mode) In Capture Mode, the register contents are also affected by hardware.
T2BRC0	[31:16]	rwh	**T2B Reload/Capture Value** (in Split Mode) In Capture Mode, the register contents are also affected by hardware.

T2RC1
Timer T2 Reload/Capture Register 1 **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
T2ARC1	[15:0]	rwh	**T2A Reload/Capture Value** (in Split Mode) In Capture Mode, the register contents are also affected by hardware.
T2BRC1	[31:16]	rwh	**T2B Reload/Capture Value** (in Split Mode) In Capture Mode, the register contents are also affected by hardware.

6.2.3 Global Control Registers

Output Source Selection Register OSEL

This register selects the output source function for the output state bits.

OSEL
Output Source Selection Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**SO7**			**0**	**SO6**			**0**	**SO5**			**0**	**SO4**		
r	rw			r	rw			r	rw			r	rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**SO3**			**0**	**SO2**			**0**	**SO1**			**0**	**SO0**		
r	rw			r	rw			r	rw			r	rw		

Field	Bits	Type	Description
SO0	[2:0]	rw	**Output 0 Source Selection** see **Table 6-11** for encoding
SO1	[6:4]	rw	**Output 1 Source Selection** encoding see **Table 6-11**
SO2	[10:8]	rw	**Output 2 Source Selection** encoding see **Table 6-11**
SO3	[14:12]	rw	**Output 3 Source Selection** encoding see **Table 6-11**
SO4	[18:16]	rw	**Output 4 Source Selection** encoding see **Table 6-11**
SO5	[22:20]	rw	**Output 5 Source Selection** encoding see **Table 6-11**
SO6	[26:24]	rw	**Output 6 Source Selection** encoding see **Table 6-11**
SO7	[30:28]	rw	**Output 7 Source Selection** encoding see **Table 6-11**
0	3, 7, 11, 15, 19, 23, 27, 31	r	**Reserved**; read as 0; writing to these bit positions has no effect.

Table 6-11 T2 Output Signal Source Selection

Value	Selected Source
000	OUT00
001	OUT01
010	OUT10
011	OUT11
100	OUV_T2A
101	OUV_T2B
110	Reserved. Do not use these combinations.
111	

Output Register OUT

Each output has an output state bit, OUTx. These bits toggle each time a trigger signal occurs. The state of these bits can be made available at the respective output pins through the alternate function selections at these pins. The output state bits and the enable bits are contained in the output control register OUT.

The output state bits can also be modified by software. Individual set and clear bits are provided for each of the output state bits. Software can update a state bit via these separate bits only.

OUT
Output Register **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								SETO7	SETO6	SETO5	SETO4	SETO3	SETO2	SETO1	SETO0
r								w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLRO7	CLRO6	CLRO5	CLRO4	CLRO3	CLRO2	CLRO1	CLRO0	OUT7	OUT6	OUT5	OUT4	OUT3	OUT2	OUT1	OUT0
w	w	w	w	w	w	w	w	rh	rh	rh	rh	rh	rh	rh	rh

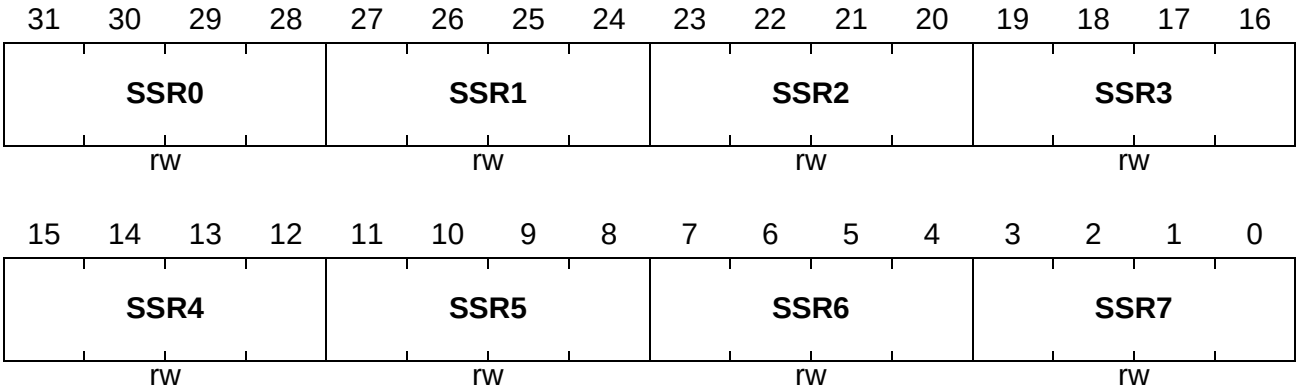
Field	Bits	Type	Description
OUTx (x = 7-0)	[7:0]	rh	**Output x Status Bit** This status bit can be directly set or reset by the associated trigger event. It can be set or reset only by software via writing a 1 to either bit SETOx or bit CLROx, respectively. Writing directly to this bit via software has no effect.
CLROx (x = 7-0)	[15:8]	w	**Output x Clear Bit** Writing a 1 to this bit causes the output bit OUTx to be cleared. Possible hardware modifications of OUTx that occurred during read-modify-write instructions (for example, bit set, bit clear instructions) are lost; the software modification has priority. The value written to CLROx is not stored. Writing a 0 to this bit has no effect. This bit always returns 0 when read. If both SETOx and CLROx are set, OUTx is not affected.

Field	Bits	Type	Description
SETOx (x = 7-0)	[23:16]	w	**Output x Set Bit** Writing a 1 to this bit causes the output bit OUTx to be set to 1. Possible hardware modifications of OUTx that occurred during read-modify-write instructions (for example, bit set, bit clear instructions) are lost; the software modification has priority. The value written to SETOx is not stored. Writing a 0 to this bit has no effect. This bit always returns 0 when read. If both SETOx and CLROx are set, OUTx is not affected.
0	[31:24]	r	**Reserved**; read as 0; writing to these bit positions has no effect.

Service Request Source Selection Register

This register selects which of the various events in the Timer T0, T1, and T2 blocks generate one of the eight service requests.

SRSEL
Service Request Source Selection Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
SSR7	[3:0]	rw	**Service Request Node 7 Source Selection** encoding see **Table 6-12**
SSR6	[7:4]	rw	**Service Request Node 6 Source Selection** encoding see **Table 6-12**
SSR5	[11:8]	rw	**Service Request Node 5 Source Selection** encoding see **Table 6-12**
SSR4	[15:12]	rw	**Service Request Node 4 Source Selection** encoding see **Table 6-12**
SSR3	[19:16]	rw	**Service Request Node 3 Source Selection** encoding see **Table 6-12**
SSR2	[23:20]	rw	**Service Request Node 2 Source Selection** encoding see **Table 6-12**
SSR1	[27:24]	rw	**Service Request Node 1 Source Selection** encoding see **Table 6-12**
SSR0	[31:28]	rw	**Service Request Node 0 Source Selection** encoding see **Table 6-12**

Table 6-12 T2 Service Request Source Selection

Value	Selected Source
0000	Start_A
0001	Stop_A
0010	UpDown_A
0011	Clear_A
0100	RLCP0_A
0101	RLCP1_A
0110	OUV_T2A
0111	OUV_T2B
1000	Start_B
1001	Stop_B
1010	RLCP0_B
1011	RLCP1_B
1100	SR00
1101	SR01
1110	SR10
1111	SR11

6.3 GPTU Module Implementation

This section describes the GPTU Module interfaces with the clock control, port connections, interrupt control, and address decoding.

6.3.1 Interfaces of the GPTU Module

Figure 6-16 shows the TC1775 specific implementation details and interconnections of the GPTU Module. The GPTU Module has eight I/O lines located at Port 13. Further, the GPTU Module is supplied by a separate clock control, interrupt control, and address decoding logic.

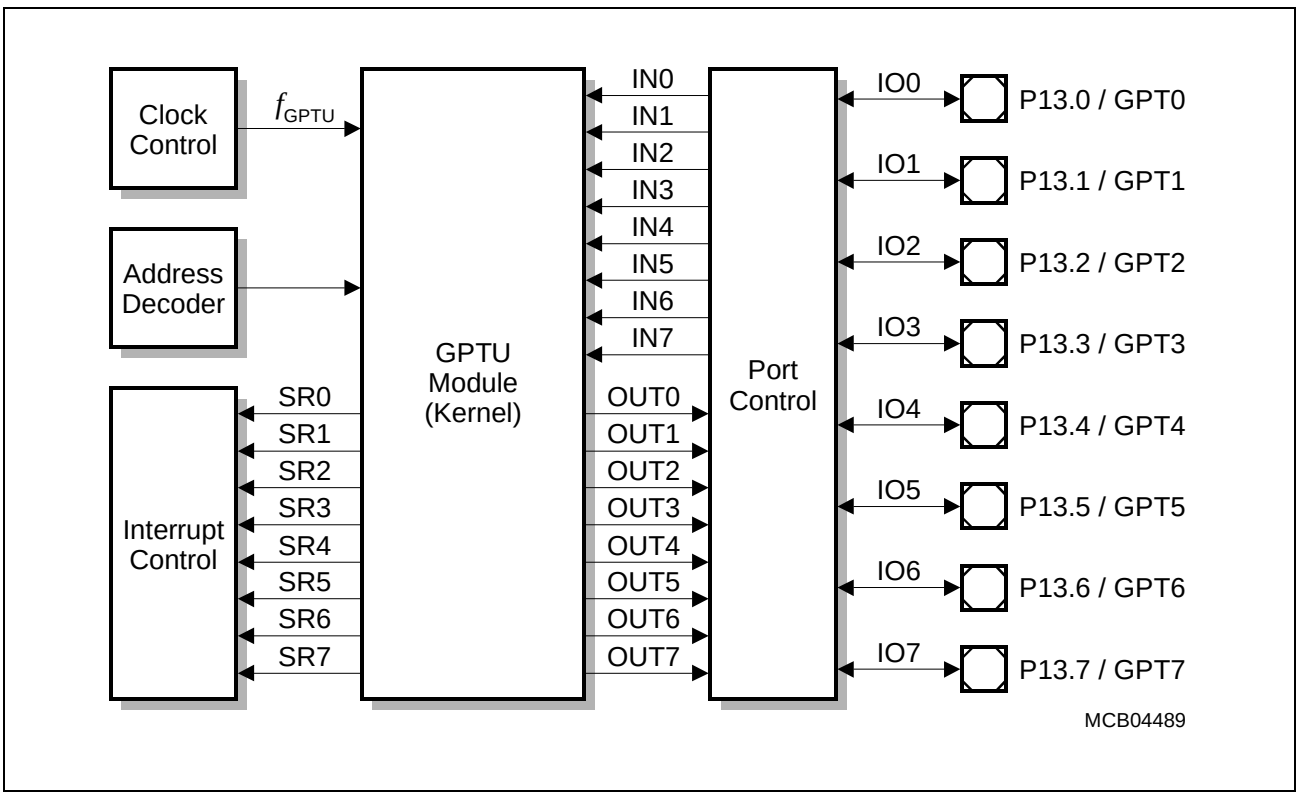


Figure 6-16 GPTU Module Implementation and Interconnections

6.3.2 External GPTU Module Registers

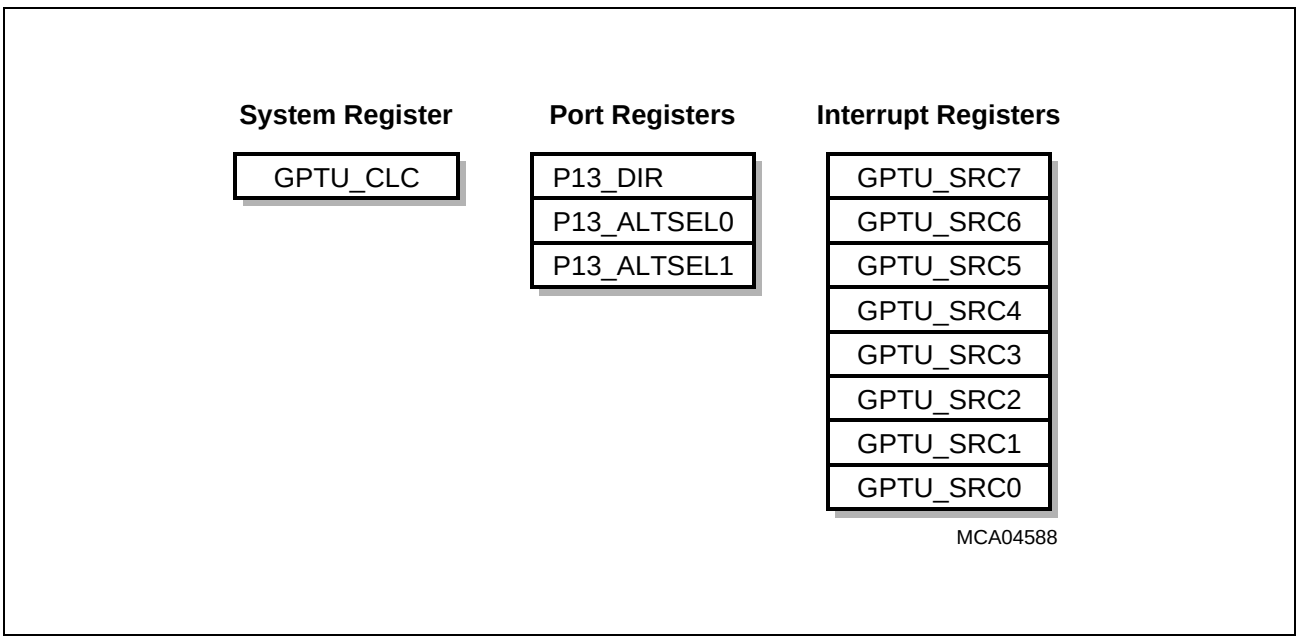
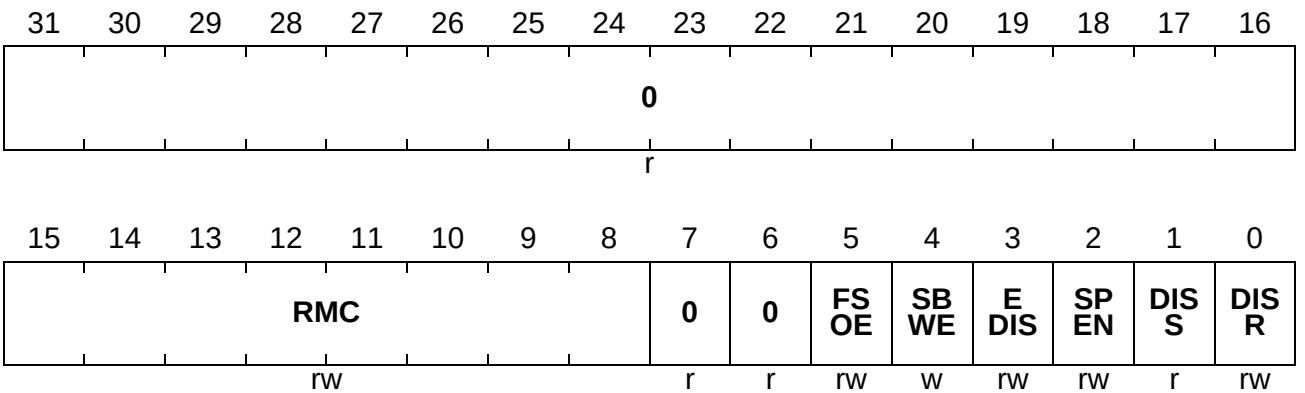


Figure 6-17 GPTU Implementation Specific Special Function Registers

6.3.2.1 Clock Control Register

The clock control register allows the programmer to adapt the functionality and power consumption of the GPTU Module to the requirements of the application. The diagram below shows the clock control register functionality as implemented for the GPTU Module.

GPTU_CLC
GPTU Clock Control Register **Reset Value: 0000 0002$_H$**



Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control of the module.
DISS	1	r	**Module Disable Status Bit** Bit indicates the current status of the module.
SPEN	2	rw	**Module Suspend Enable for OCDS** Used for enabling the suspend mode.
EDIS	3	rw	**External Request Disable** Used for controlling the external clock disable request.
SBWE	4	w	**Module Suspend Bit Write Enable for OCDS** Defines whether SPEN and FSOE are write protected.
FSOE	5	rw	**Fast Switch Off Enable** Used for fast clock switch off in OCDS suspend mode.
RMC	[15:8]	rw	**8-Bit Clock Divider Value in RUN Mode**
0	7, 6, [31:16]	r	**Reserved**; returns 0 if read; should be written with 0.

Note: After a hardware reset operation, the GPTU Module is disabled.

6.3.2.2 Port Registers

The alternate functions, associated with the GPTU I/O lines, are controlled by the ALTSEL registers located in the ports. The GPTU I/O lines are connected with Port 13. Therefore, P13_ALTSEL0 and P13_ALTSEL1 must be programmed for the Port 13 pins which are required for the GPTU Module in the specific application.

Note: Bits marked with 'X' are not relevant for GPTU operation.

P13_ALTSEL0
Port 13 Alternate Select Register 0 **Reset Value: 0000 0000$_H$**

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X								P7	P6	P5	P4	P3	P2	P1	P0
rw								rw	rw	rw	rw	rw	rw	rw	rw

P13_ALTSEL1
Port 13 Alternate Select Register 1 **Reset Value: 0000 0000$_H$**

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X								P7	P6	P5	P4	P3	P2	P1	P0
rw								rw	rw	rw	rw	rw	rw	rw	rw

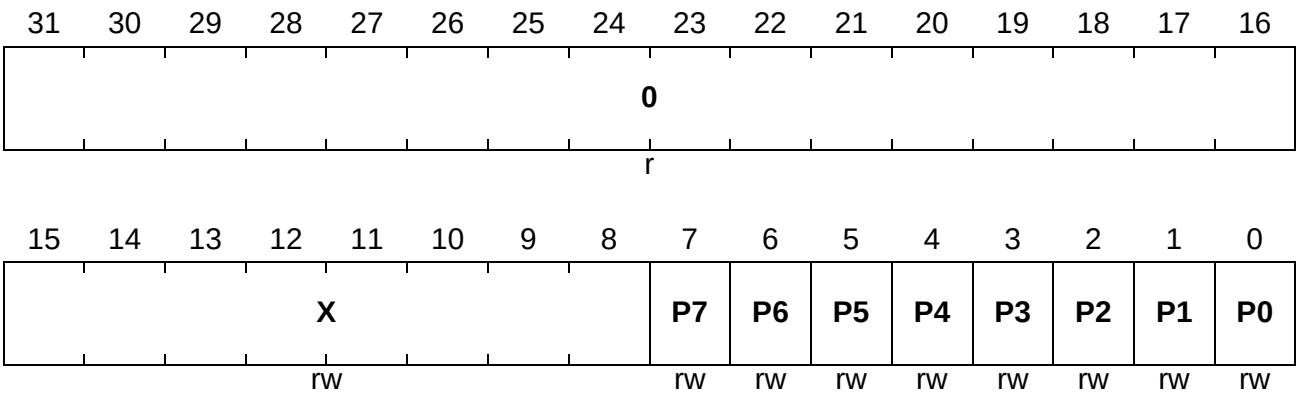
An I/O pin n (n = 7-0) of Port 13 is assigned as GPTU I/O line if the following condition is met;

- P13_ALTSEL0.Pn = 1 and
- P13_ALTSEL1.Pn = 0

Note: P13.n/GPTUn (n = 7-0) is assigned to the INn/OUTn input and output lines of the GPTU. Bits 7-0 of P13_ALTSEL0 and P13_ALTSEL1 must be set to the values defined above only for the lines of Port 13 that are used by the GPTU in a specific application. The unused GPTU port lines of Port 13 can be assigned for general purpose I/O or for other alternate I/O functions.

The direction register configures the direction of a port pin and must be set according to the selected GPTU operation mode (direction bit = 0: pin is set to input, direction bit = 1: pin is set to output). The GPTU I/O lines are connected with Port 13. Therefore, the Port 13 Direction Register P13_DIR must be set accordingly.

P13_DIR
Port 13 Direction Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
Pn **(n = 7-0)**	[7:0]	rw	**GPTUn I/O Lines Direction Select** The bits of P13_DIR select the direction of the I/O lines assigned to the GPTU. 0 P13 port pin n is selected as input 1 P13 port pin n is selected as output
X	[15:8]	rw	**Reserved** for other Port 13 direction selections.
0	[31:16]	r	**Reserved**; read as 0; should be written with 0.

Note: P13.n/GPTUn (n = 7-0) is assigned to the INn/OUTnx input and output lines of the GPTU. Depending on the operating modes of the GPTU timers, bits 7-0 of P13_DIR must be set to the appropriate value.

6.3.2.3 Interrupt Registers

The eight interrupt outputs SR7 - SR0 of the GPTU Module are controlled by the service request control registers GPTU_SRC7 to GPTU_SRC0:

GPTU_SRC0
GPTU Interrupt Service Request Control Register 0
GPTU_SRC1
GPTU Interrupt Service Request Control Register 1
GPTU_SRC2
GPTU Interrupt Service Request Control Register 2
GPTU_SRC3
GPTU Interrupt Service Request Control Register 3
GPTU_SRC4
GPTU Interrupt Service Request Control Register 4
GPTU_SRC5
GPTU Interrupt Service Request Control Register 5
GPTU_SRC6
GPTU Interrupt Service Request Control Register 6
GPTU_SRC7
GPTU Interrupt Service Request Control Register 7

Reset Values: 0000 0000$_H$

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SETR	CLRR	SRR	SRE	TOS		0		SRPN							
w	w	rh	rw	rw		r		rw							

Field	Bits	Type	Description
SRPN	[7:0]	rw	**Service Request Priority Number**
TOS	[11:10]	rw	**Type of Service Control**
SRE	12	rw	**Service Request Enable**
SRR	13	rh	**Service Request Flag**
CLRR	14	w	**Request Clear Bit**
SETR	15	w	**Request Set Bit**

Field	Bits	Type	Description
0	[9:8], [31:16]	r	**Reserved**; returns 0 if read; should be written with 0.

Note: Further details on interrupt handling and processing are described in the "Interrupt System" chapter of the TC1775 System Units User's Manual.

6.3.3 GPTU Register Address Range

In the TC1775, the registers of the two GPTU Modules are located in the following address ranges:

- Module Base Address = $F000\ 0700_H$
- Module End Address = $F000\ 07FF_H$
- Absolute Register Address = Module Base Address + Offset Address (offset addresses see **Table 6-2**)

7 General Purpose Timer Array (GPTA)

This chapter describes the General Purpose Timer Array (GPTA) of the TC1775. This chapter contains the following sections:

- Functional description of the GPTA Kernel (see **Section 7.1**)
- Register descriptions of all GPTA Kernel specific registers (see **Section 7.2**)
- TC1775 implementation specific details and registers of the GPTA Module, including port connections and control, interrupt control, address decoding, and clock control (see **Section 7.3**).

7.1 GPTA Kernel Description

7.1.1 Introduction

The General Purpose Timer Array (GPTA) provides a set of hardware modules required for high speed digital signal processing:

- Filter and Prescaler Cells (FPC) support input noise filtering and prescaler operation.
- Phase Discrimination Logic units (PDL) decode the direction information output by a rotation tracking system.
- Duty Cycle Measurement Cells (DCM) provide pulse width measurement capabilities.
- A Digital Phase Locked Loop unit (PLL) generates a programmable number of GPTA module clock ticks during an input signal's period.
- Global Timer units (GT) driven by various clock sources are implemented to operate as a time base for the associated "Global Timer Cells".
- Global Timer Cells (GTC) can be programmed to capture the contents of a Global Timer on an event that occurred at an external port pin or at an internal FPC output. A GTC may be also used to control an external port pin with the result of an internal compare operation. GTCs can be logically concatenated to provide a common external port pin with a complex signal waveform.
- Local Timer Cells (LTC) operating in Timer, Capture, or Compare Mode may be also logically tied together to drive a common external port pin with a complex signal waveform. LTCs — enabled in Timer Mode or Capture Mode — can be clocked or triggered by
  - A prescaled GPTA module clock,
  - An FPC, PDL, DCM, PLL, or GTC output signal line,
  - An external port pin.

Some input lines driven by processor I/O pads may be shared by a LTC and a GTC cell to trigger their programmed operation simultaneously.

The following list summarizes all blocks supported:

Clock Generation Unit:

- Filter and Prescaler Cell (FPC):
  - Six independent units.
  - Three operating modes (Prescaler, Delayed Debounce Filter, Immediate Debounce Filter).
  - f_{GPTA} down-scaling capability.
  - $f_{GPTA}/2$ maximum input signal frequency in Filter Mode.
- Phase Discriminator Logic (PDL):
  - Two independent units.
  - Two operating modes (2 and 3 sensor signals).
  - $f_{GPTA}/4$ maximum input signal frequency in 2-sensor mode, $f_{GPTA}/6$ maximum input signal frequency in 3-sensor mode.

- Duty Cycle Measurement (DCM):
  - Four independent units.
  - 0 - 100% margin and time-out handling.
  - f_{GPTA} maximum resolution.
  - $f_{GPTA}/2$ maximum input signal frequency.
- Digital Phase Locked Loop (PLL):
  - One unit.
  - Arbitrary multiplication factor between 1 and 65535.
  - f_{GPTA} maximum resolution.
  - $f_{GPTA}/2$ maximum input signal frequency.

Signal Generation Unit:

- Global Timers (GT):
  - Two independent units.
  - Two operating modes (Free Running Timer and Reload Timer).
  - 24-bit data width.
  - f_{GPTA} maximum resolution.
  - $f_{GPTA}/2$ maximum input signal frequency.
- Global Timer Cell (GTC):
  - 32 independent units.
  - Two operating modes (Capture, Compare and Capture after Compare).
  - 24-bit data width.
  - f_{GPTA} maximum resolution.
  - $f_{GPTA}/2$ maximum input signal frequency.
- Local Timer Cell (LTC):
  - 64 independent units.
  - Three operating modes (Timer, Capture and Compare).
  - 16-bit data width.
  - f_{GPTA} maximum resolution.
  - $f_{GPTA}/2$ maximum input signal frequency.

Interrupt Control Unit:

- 111 interrupt sources generating 54 service requests.

I/O Sharing Unit:

- Able to process lines from FPC, GTC and LTC.
- Emergency function.

The General Purpose Timer Array is partitioned into three parts:

- The **Kernel** contains all processing units including auxiliary hardware for interconnecting and concatenating the submodules.
- The **Shell** acts as an interface between the kernel and port area and allows to adapt the GPTA to a specific product environment. For this purpose, the Shell contains a clock control function, an address decoding mechanism, input selection capabilities, and an interrupt processing unit.
- The **Port Area** holds all logic to select a port pin function and the port pin characteristics (such as direction control and electrical operating condition control).

Figure 7-1 shows a global block diagram of GPTA implementation.

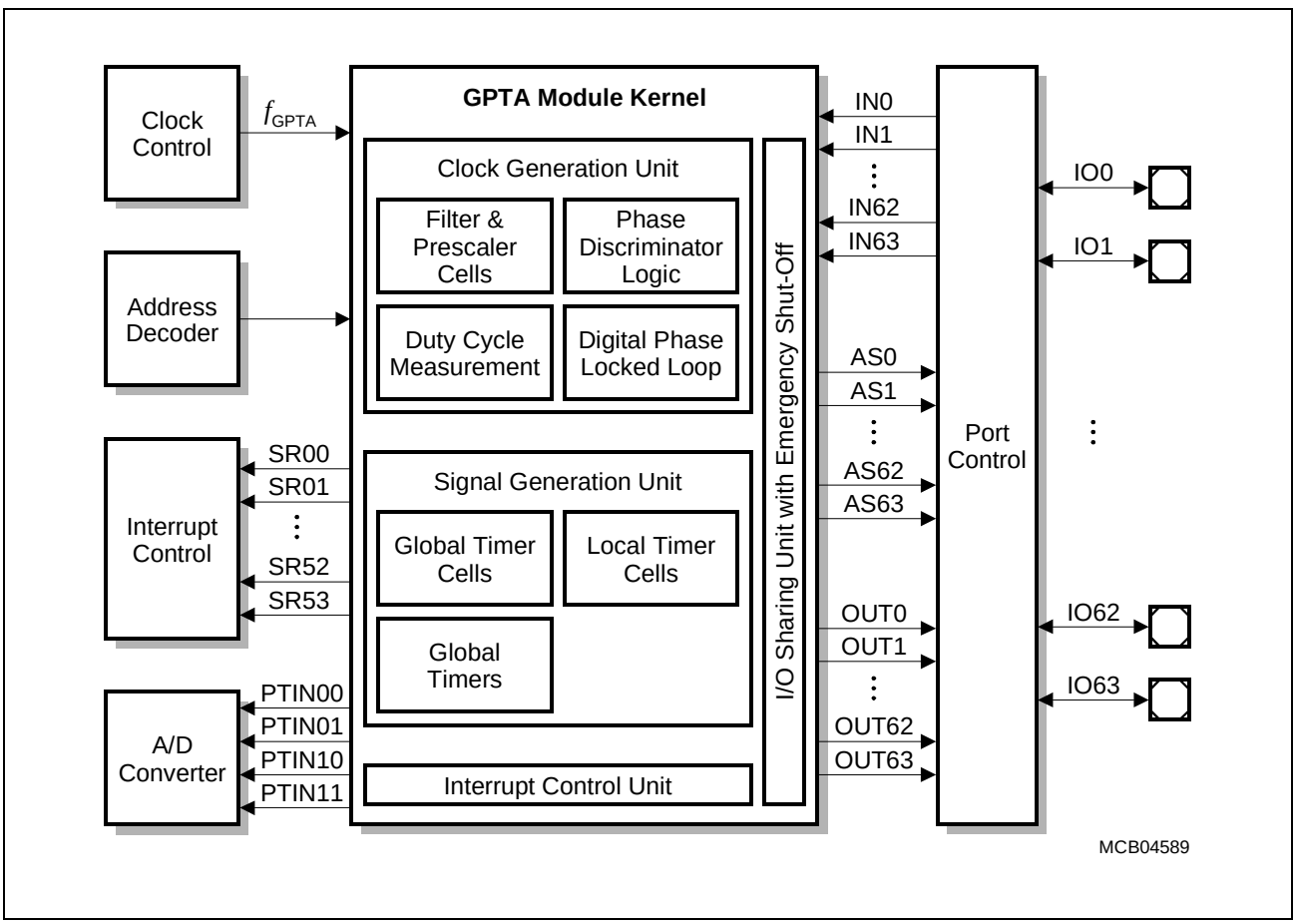


Figure 7-1 General Block Diagram of GPTA Unit

7.1.2 GTPA Units

The General Purpose Timer Array (**Figure 7-2**) is split into a Clock Generation Unit (CGU) and a Signal Generation Unit (SGU):

- The **Clock Generation Unit** (see **Section 7.1.3**) allows a preprocessing of the input signals using filter, timer, capture, compare and enhanced digital PLL Modules:
  - The **Filter and Prescaler Cells** (FPC) provide input noise filtering (high and low pass) and may also work as prescalers for the GPTA clock and external signals.
  - The **Phase Discrimination Logic** (PDL) may take the outputs of the FPCs to decode phase encoded signals from a position and rotation direction sensor system.
  - The **Duty Cycle Measurement Cells** (DCM) provide signal measurement capabilities (timer plus capture register, single and double capture on rising and falling edges or both) as well as missing pulse detection/reconstruction features.
  - The **Digital Phase Locked Loop** (Digital PLL) is intended to generate a higher resolution clock out of the values measured by DCM cells. Any arbitrary multiplication factor between 1 and 65535 is supported and may be changed from input clock period to input clock period.

The original signals and all outputs of the preprocessing units are distributed to the Global Timers and Local Timer Cells via the clock bus.

- The **Signal Generation Unit** (see **Section 7.1.4**) provides a set of timers, capture and compare units:
  - Both 24 bit **Global Timers** (GT) may be individually configured as free running counters or as reload counters starting at a programmable value from 0_H to $FFFFFF_H$. Each GT is equipped with a scalable greater-or-equal comparator, the number of bits to be compared is selectable.
  - The **Global Timer Cell** registers (GTC) are 24 bit wide. GTCs may be used as comparators (modifying the logical state of a related output port pin), or as capture units, storing the current GT0 or GT1 value on rising, falling or both signal edges detected on a related input port pin. Several adjacent GTCs may be connected to logical units operating on the same pin, allowing complex functions to be implemented.
  - The **Local Timer Cell** registers (LTC) are 16 bit wide. LTC may be configured to operate in one of four different modes: free running or resetable counter, capture or compare unit. Adjacent cells may be combined to operate on the same pin, thus generating complex waveforms.

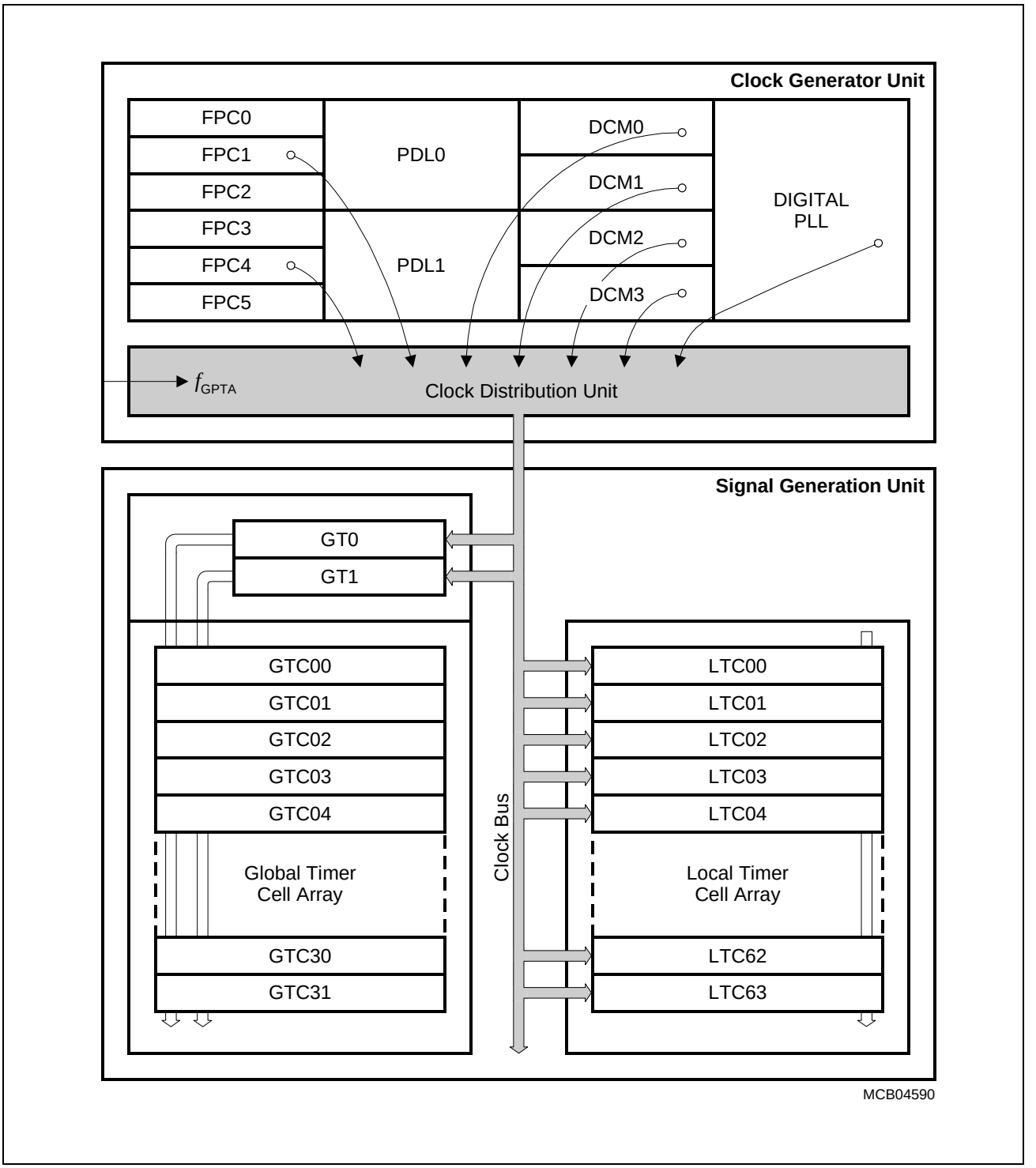


Figure 7-2 GPTA Block Diagram

7.1.3 Clock Generation Unit

The Clock Generation Unit provides five signal preprocessing modules explained in detail in the following sections.

7.1.3.1 Filter and Prescaler Cell (FPC)

The GPTA contains six filter and prescaler cells (FPC5 to FPC0) driven by signal lines coming from an I/O port (**Section 7.1.5**).

Each FPC is equipped with an input signal multiplexer, an input signal sampling unit, an edge detect unit, a 16 bit timer, a 16 bit compare register, a 16 bit comparator and, and an FPC control unit (**Figure 7-3**).

Two output lines are provided by each FPC Module as follows:

- An event output signal, reporting a falling or rising signal edge on the FPC input by a single f_{GPTA} clock pulse,
- A level output signal, indicating the direction of the detected signal transition.

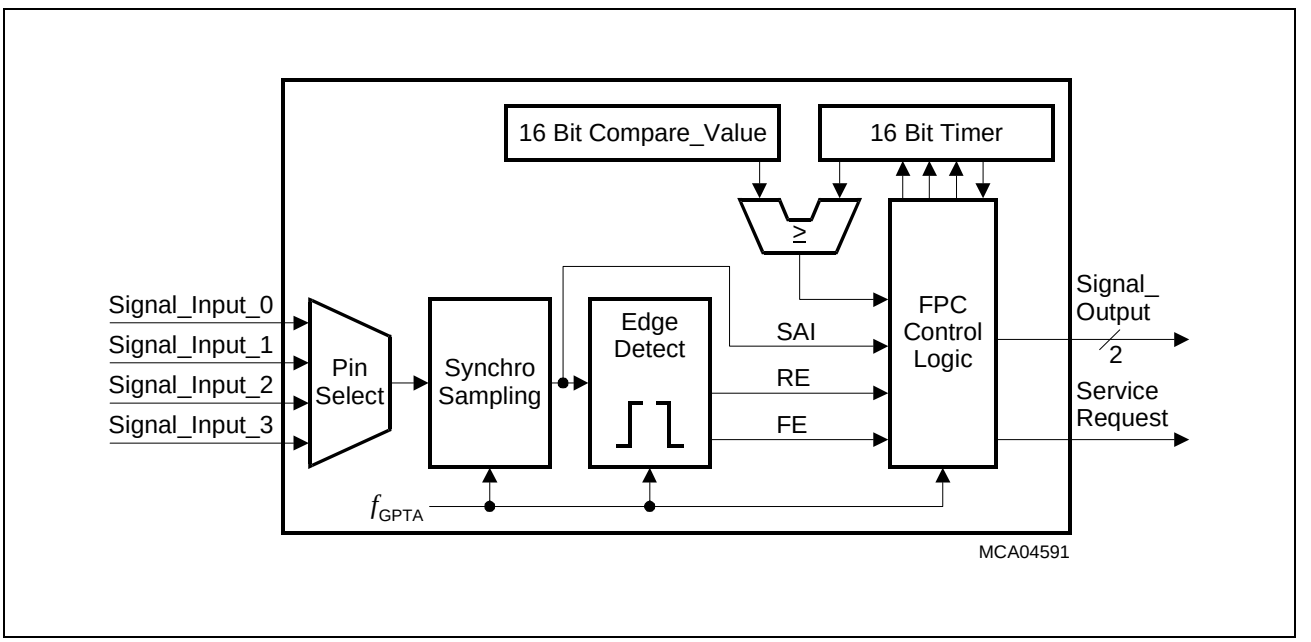


Figure 7-3 Filter and Prescaler Cell Architecture

*Note: FPC inputs connection are described in **Section 7.1.5**.*

Each filter and prescaler cell can be individually configured to operate in one of four modes. Whether a filter or prescaler operation is performed depends on the configuration bit fields MOD0k in control register FPCCTR2.

In **Delayed Debounce Filter Mode**, the input signal is filtered from *all* signal transitions and glitches with a width smaller than the GPTA module clock period length multiplied by the compare register value (**Figure 7-4**).

The input signal is sampled with the GPTA module clock rate. The FPC Control Unit analyses each sampling value. If the state of the input sample differs from the current output signal value, the 16 bit timer is incremented by one. When the timer register FPCTIMk is not in its idle state (0_H) and the state of the input sample matches the current output signal value, the 16 bit timer is decremented by one and the glitch record flag GRCk of the respective FPC in FPCCTR1 control register is set. When the timer matches the value stored in the 16 bit FPCCOMk compare register (timer threshold), the level output signal line is updated with the current state of the input line, the event output signal line is driven by a GPTA module clock pulse and the timer is reset to its idle state. The glitch record bit GRCk must be reset by software.

The filter is by-passed, if the compare register is programmed to zero (0_H). In this case the input signal is directly copied to the output signal.

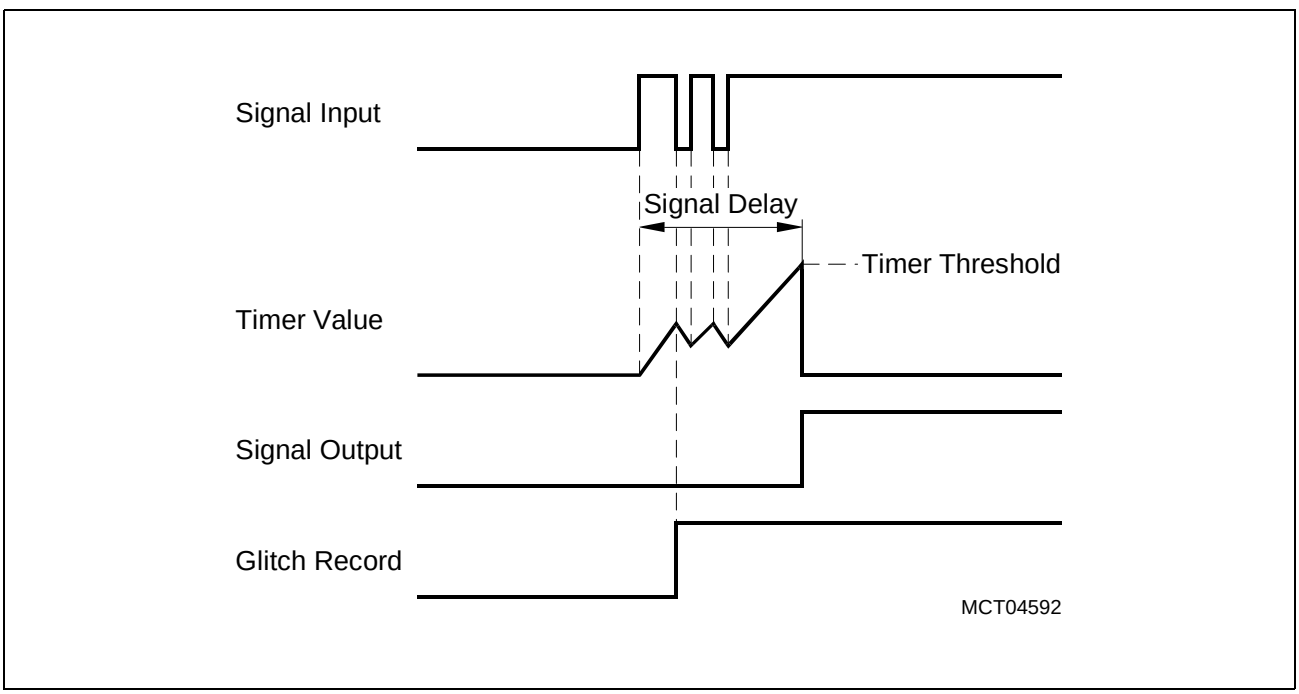


Figure 7-4 FPC Delayed Debounce Filter Algorithm

Due to the sample and hold unit, the maximum FPC input signal frequency must not exceed the Nyquist limit (one-half the GPTA module clock rate). The signal delay from input to output depends on the programmed compare register value and the number of high-frequency pulses (glitches) during the filter operating time.

In the **Immediate Debounce Filter Mode**, the input signal is filtered from signal transitions and glitches arriving a programmable time after an input signal edge detection (**Figure 7-5**).

The input signal is sampled with the GPTA module clock rate. As long as the timer is reset, the FPC Control Unit copies the sampled input value directly to the output signal line. When a rising or falling input signal edge occurs, and the value in the FPCCOMk register is not zero, the timer is enabled to be incremented by the GPTA module clock and the copy mechanism is disabled. When the timer value matches the contents of the 16-bit compare register FPCCOMk, it is reset and the copy mechanism is enabled again. A rising or falling edge, occurring on the input signal line while the timer is greater than zero but less than the compare value, triggers the glitch record flag of the respective FPC in Control Register 1. The glitch record bit must be reset by software.

The filter is by-passed if the compare register is programmed to zero (0_H). In this case, the input signal is directly copied to the output signal without any disable periods.

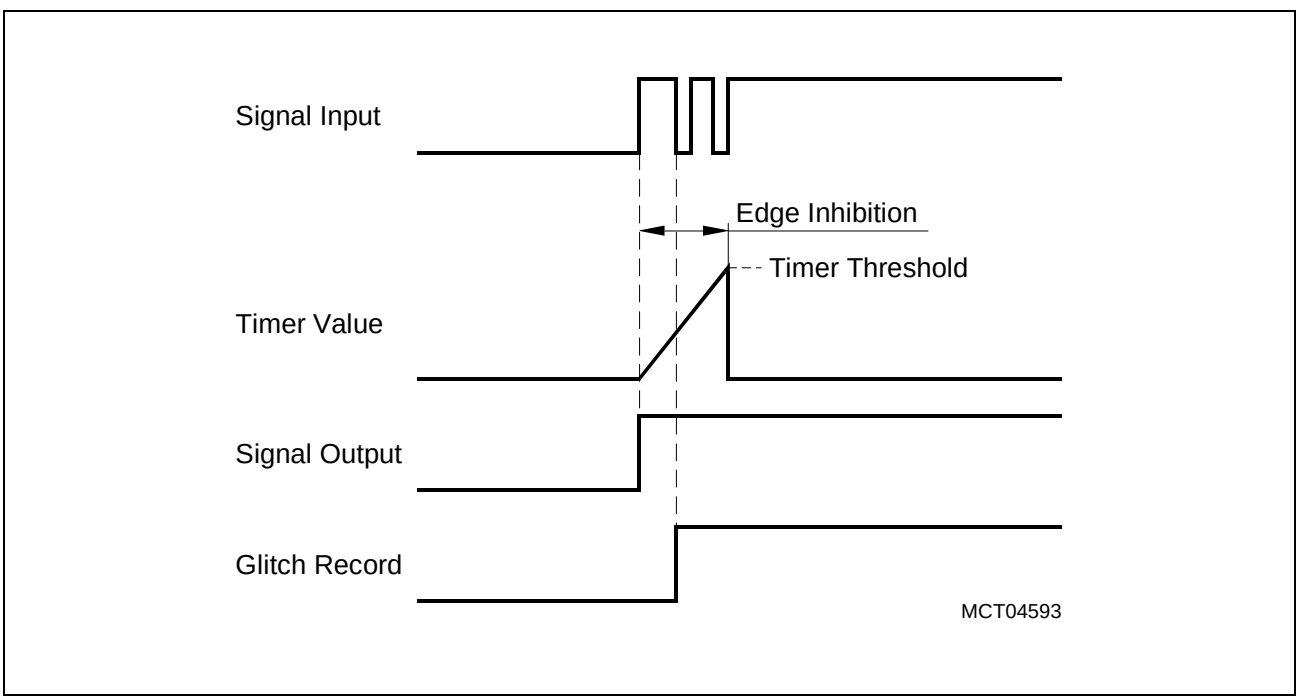


Figure 7-5 FPC Immediate Debounce Filter Algorithm

In **Prescaler Mode**, the FPC Control Unit counts each rising (or falling) edge of the sampled input signal. When the timer value matches the contents of the compare register:

- the level output signal is equal to the event output signal,
- the event output signal is provided with one GPTA module clock pulse,
- and the timer is reset to zero.

An extension to the Prescaler Rising Edge Mode has been implemented to provide an FPC output signal directly derived from the GPTA module clock. For that, the Prescaler Rising Edge Mode and the input signal 0 must be selected. In this mode, the FPC output provides a signal generated by a GPTA clock division with a divisor value stored in the compare register.

Due to the sample and hold unit, the maximum FPC input signal frequency must not exceed the Nyquist limit (one-half the sampling rate).

Signal Output Information Splitting

The FPC output is split into a level and an event signal providing all following PDL and DCM cells with the information about an input signal transition at the same f_{GPTA} clock cycle. This implementation avoids cascading a one clock delay per edge detection unit implemented at the input of each following cell.

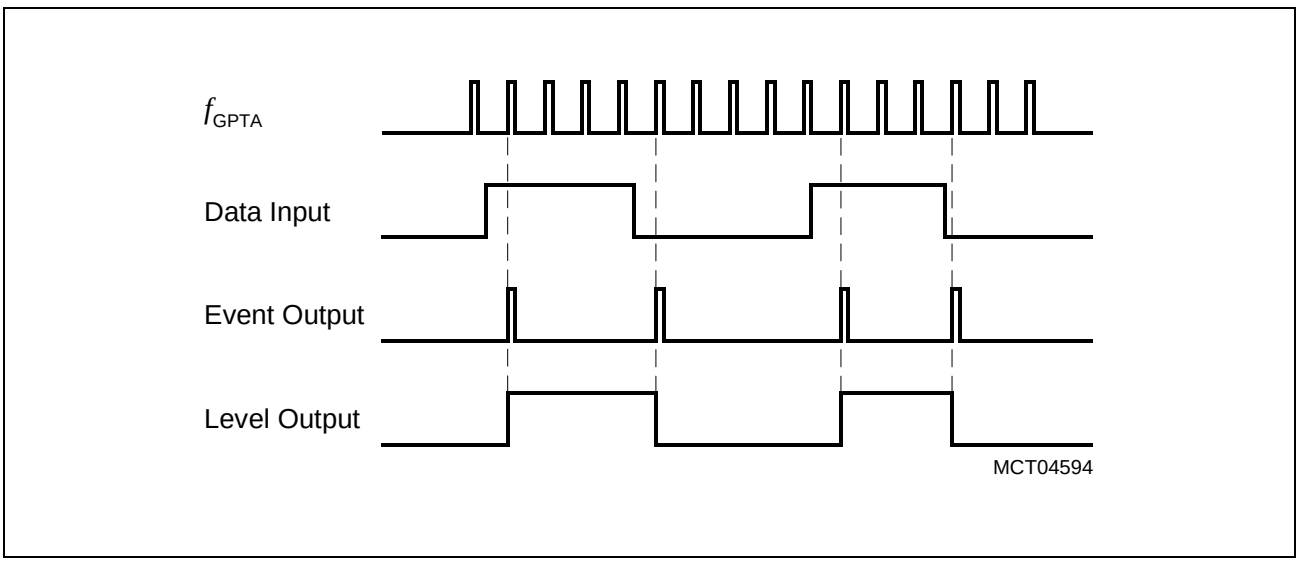


Figure 7-6 FPC Output Split into Level and Event Information

7.1.3.2 Phase Discrimination Logic (PDL)

The GPTA provides two Phase Discrimination Logic modules (PDL0, PDL1) driven by two signal lines coming from an FPC:

- an event input signal and
- a level input signal.

Each PDL is equipped with an edge detection unit, a phase detection unit, a PDL control unit, and an output multiplexer (**Figure 7-7**).

Six output lines are provided by each PDL Module:

- The “Forward” output signal (F0, F1) is driven for one f_{GPTA} clock pulse, if an input signal edge is interpreted as forward rotation. These output lines are directly connected to Local Timer Cells (LTC00, LTC02).
- The “Backward” output signal (B0, B1) is driven by one f_{GPTA} clock pulse, if an input signal edge is recognized as backward rotation. These output lines are directly connected to LTC01 and LTC03.
- Two pairs of output signals, carrying the bypassed input level and event information from the driving FPCs or the angular velocity and error information provided by the PDL function. These output lines are directly connected to the adjacent Duty Cycle Measurement Cells DCM0, DCM1, DCM2 and DCM3.

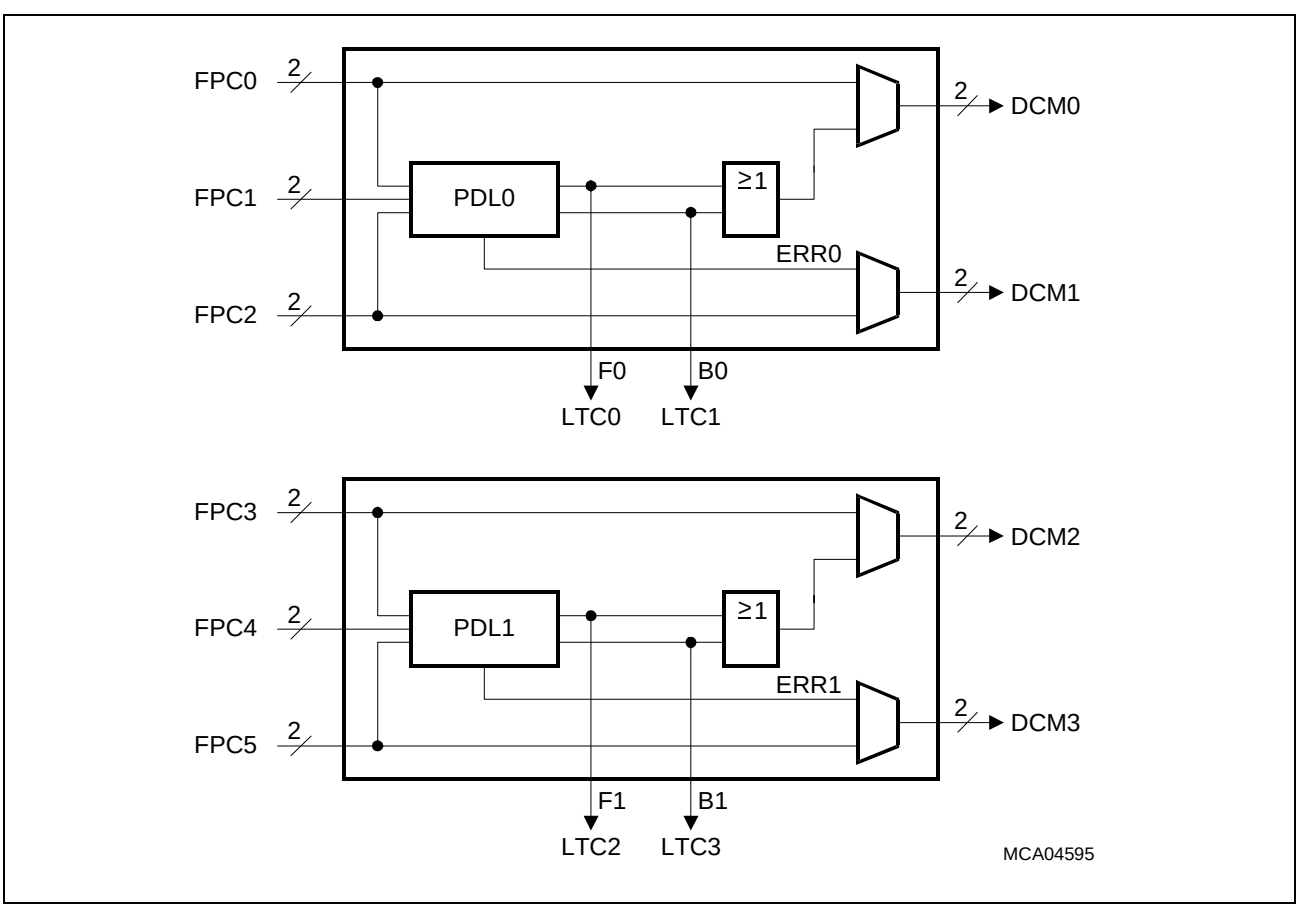


Figure 7-7 Block Diagram of Phase Discrimination Logic Unit

The PDL processes the output signal of a 2-sensor or 3-sensors' positioning system. Bit TSEx in control register PDLCTR=1 configures a 3-sensor system execution and provides the DCM1 and/or DCM3 cell with information concerning erroneous states in the signal input. When TSEx is set to 0, a 2-sensor system is selected and DCM1 and/or DCM3 are supplied with the input event and level information from the driving FPC2 and/or FPC5 cells.

The rotation direction, monitored by the connected sensors, is automatically derived from the sequence in which the input signals change. Each edge detected on an input signal line generates a pulse on the F0, F1 forward output lines or on the B0, B1 backward output lines. Input jitter, which might occur if a sensor rests near to one of its switching points, is compensated. When bit MUXx in control register PDLCTR is set to 1, the associated DCM0 and/or DCM2 cells are provided with the angular velocity information generated by a boolean 'OR' operation on the "Forward" and "Backward" signal. If bit MUXx in control register PDLCTR is reset, the DCM0 and/or DCM2 cells are supplied with the input event and level information from the driving FPC0 and/or FPC3 cells.

To calculate the sensor's current position, the associated Local Timer Cells should be clocked with the PDL "Forward" and "Backward" output pulses. A software operation, subtracting the "Backward" counter contents from the "Forward" counter contents, provides the absolute position. Dynamic information (speed, acceleration, deceleration) may be obtained by analyzing the angular velocity signal periods with the associated DCM cell.

The maximum input frequency is $f_{GPTA}/4$ for a 2-sensor positioning system and $f_{GPTA}/6$ for a 3-sensor positioning system. To ensure that a transition of any input signal is correctly recognized, its level should be held high or low for at least two f_{GPTA} cycles before it changes (three f_{GPTA} cycles for a 3-sensor positioning system).

Positioning System With Two Sensors

The "2-Sensor Mode" is enabled when bit TSEx in control register PDLCTR is reset. The sensors are mounted at a 90° angle to each other (**Figure 7-8**). The third sensor input of the PDL Module is internally disabled.

This configuration can measure an absolute position with an accuracy of 90°.

!	means not
Re	means rising edge
Fe	means falling edge
Forward	ReS1*!S2 + S1*ReS2 + FeS1*S2 + !S1*FeS2
Backward	ReS1*S2 + !S1*ReS2 + FeS1*!S2 + S1*FeS2
Position	Forward_counter - Backward_counter

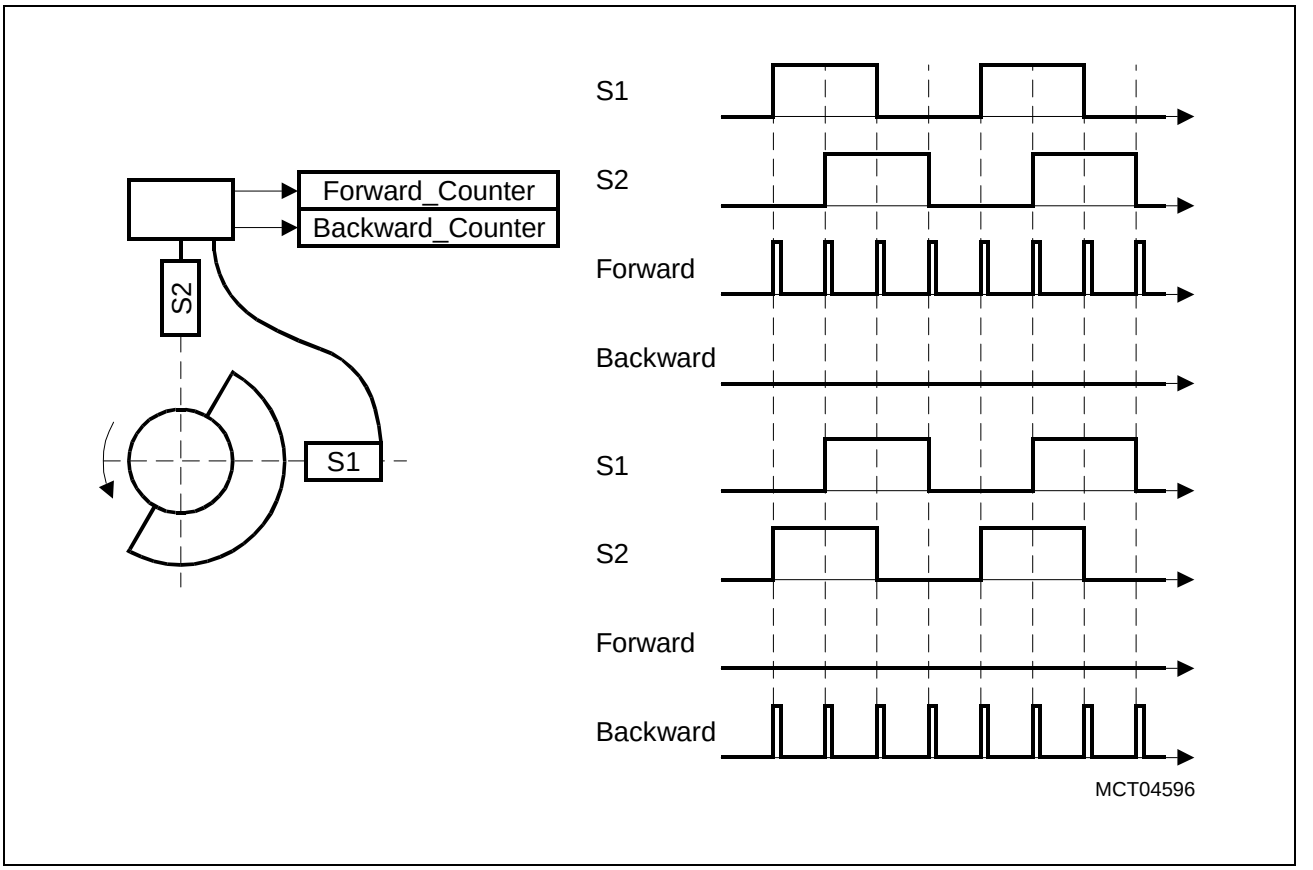


Figure 7-8 Interface Signals of a PDL in a 2-Sensor Positioning System

Figure 7-9 illustrates how the output signals of a “2-Sensor System” superimposed with noise are processed by the PDL unit. Jitter pulses are completely compensated if they do not occur on both signal lines simultaneously.

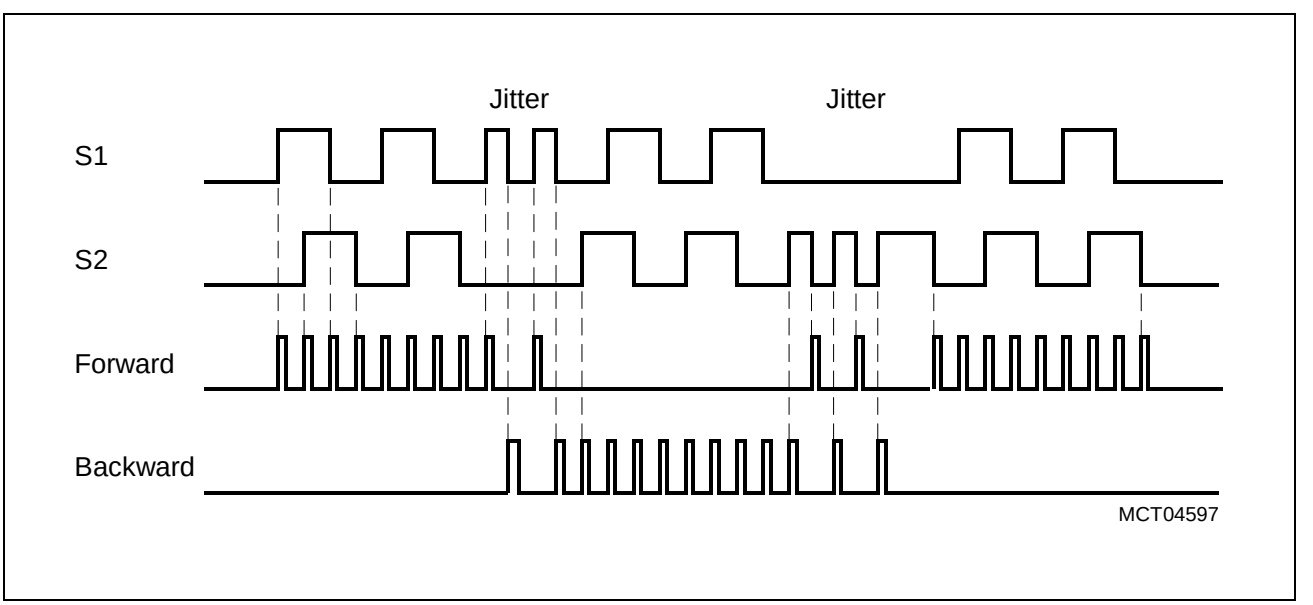


Figure 7-9 Compensation of Input Jitter

Positioning System with Three Sensors

The “3-Sensor Mode” is enabled when bit TSEx in control register PDLCTR is set to 1. The sensors are mounted at an 120° angle to each other (**Figure 7-10**).

This configuration can measure an absolute position with an accuracy of 60°.

Input signal combinations that are not allowed in a properly-working positioning system (such as all inputs low or all inputs high) cause the following to occur:

- an error signal is generated, driving the Duty Cycle Measurement Cells DCM1 and/or DCM3,
- the error flag ERRx in control register PDLCTR is set
- and no forward or backward pulses are generated.

!	means not
Re	means rising edge
Fe	means falling edge
Forward	ReS1*!S2*S3 + FeS3*S1*!S2 + ReS2*S1*!S3 + FeS1*S2*!S3 + ReS3*!S1*S2 + FeS2*!S1*S3
Backward	ReS1*S2*!S3 + FeS3*!S1*S2 + ReS2*!S1*S3 + FeS1*!S2*S3 + ReS3*S1*!S2 + FeS2*S1*!S3
Error	The input signal states S1*S2*S3 and !S1*!S2*!S3 are not allowed
Position	Forward_counter - Backward_counter

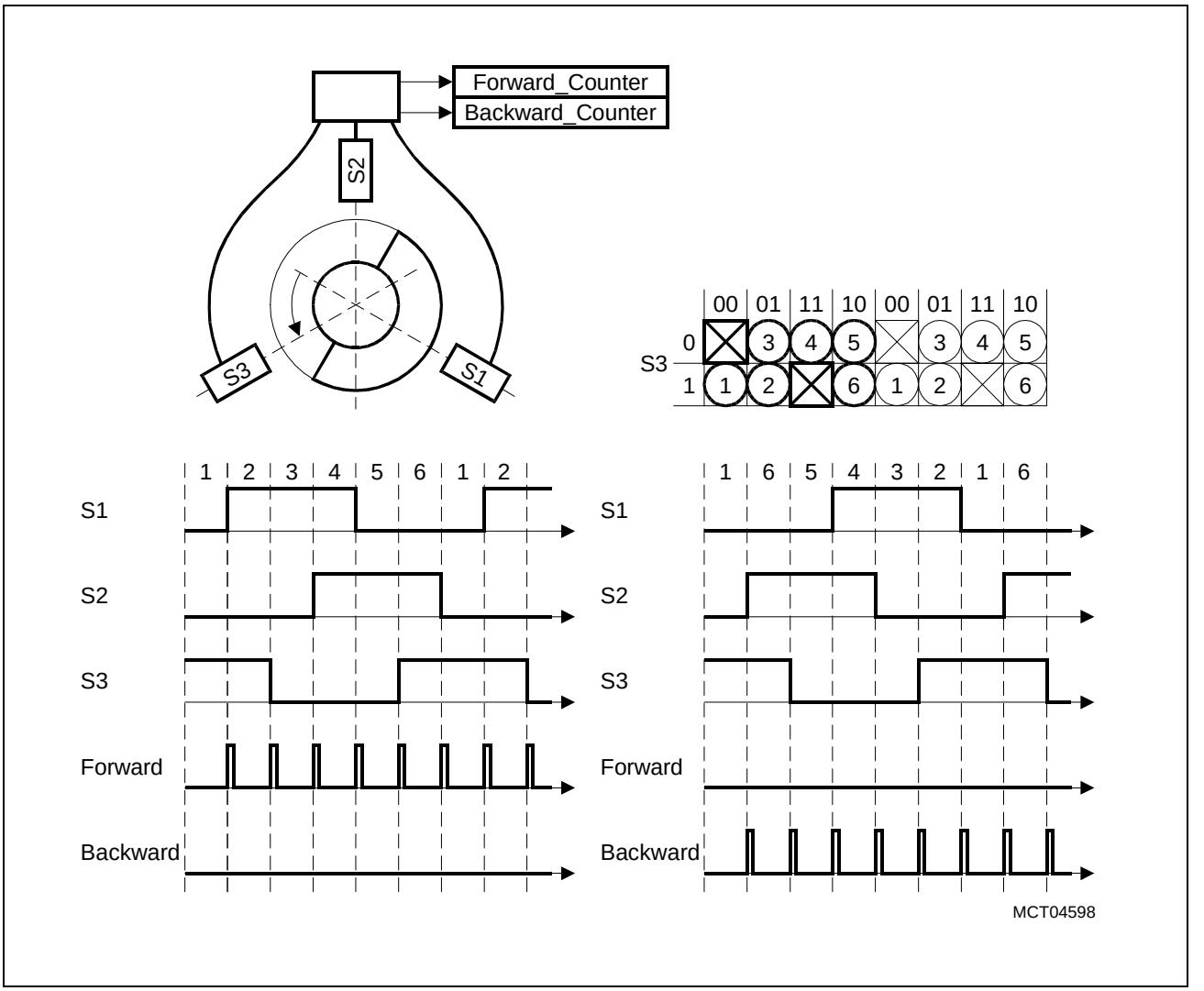


Figure 7-10 Interface Signals of a PDL in a 3-Sensor Positioning System

Jitter pulses are completely compensated as illustrated in **Figure 7-9**.

7.1.3.3 Duty Cycle Measurement Unit (DCM)

The GPTA contains four DCM Modules (DCM3 to DCM0). Each DCM has two inputs for the signal to be analyzed:

- An event input
- And a signal level input

Each DCM is locally equipped with a 24-bit timer, a 24-bit capture register, a 24-bit capture/compare register, a 24-bit comparator and a DCM control unit (**Figure 7-11**).

Each DCM Module has four outputs:

- An event output line
- An interrupt line triggered by an input rising edge detection
- An interrupt line triggered by an input falling edge detection
- An interrupt line triggered by a compare event

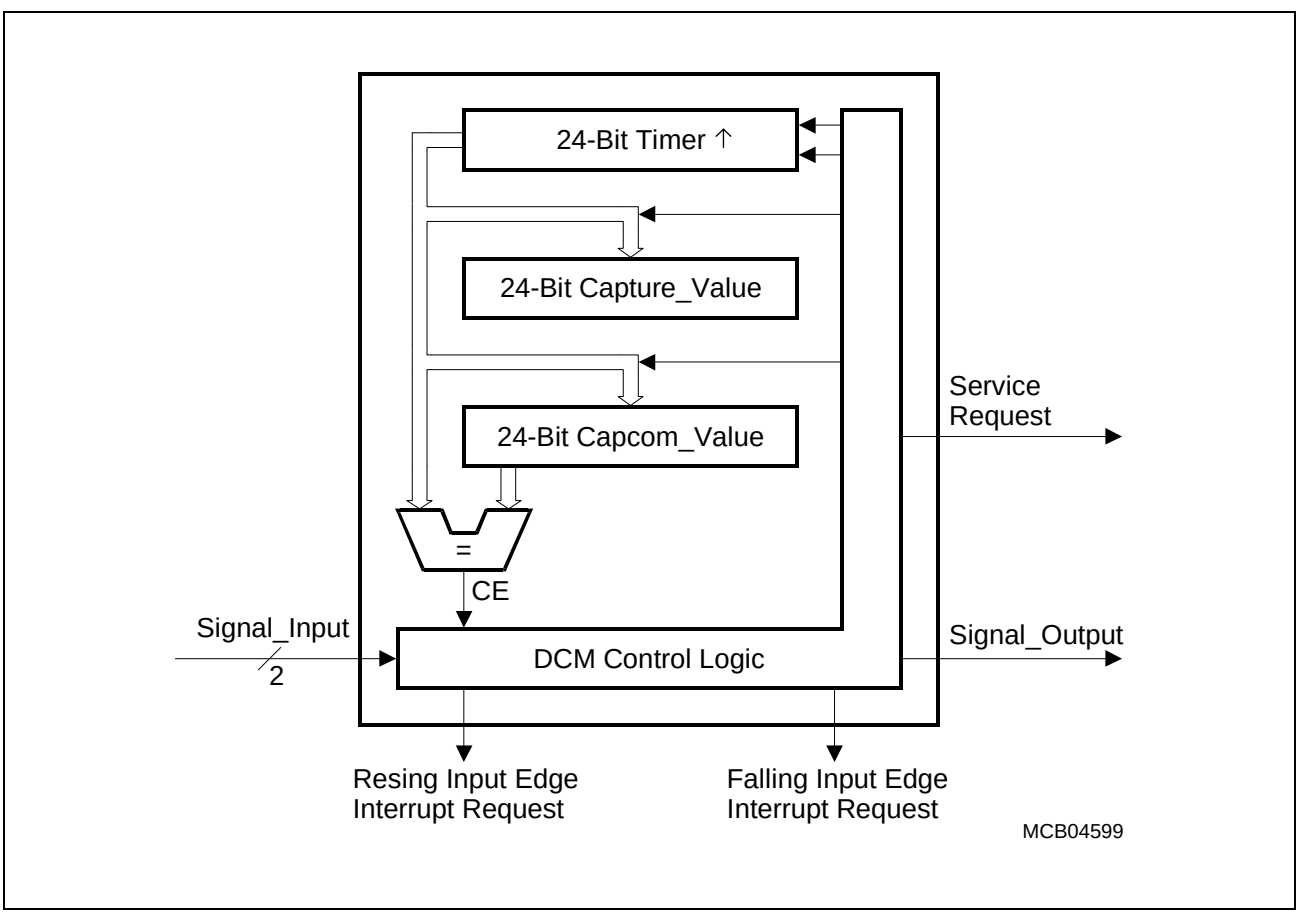


Figure 7-11 Block Diagram of Duty Cycle Measurement Module

The DCM Module supports duty cycle measurement on the input signal being hard-wired to a PDL cell output. Depending on the configuration of the associated PDL cell, some DCM Modules may be also driven by a FPC directly (**Figure 7-7**):

- DCM0 is driven by FPC0 or PDL0 angular velocity signal,
- DCM1 is driven by FPC2 or PDL0 error signal,
- DCM2 is driven by FPC3 or PDL1 angular velocity signal,
- DCM3 is driven by FPC5 or PDL1 error signal.

When the driving FPCs and PDL cells are programmed to feed-through mode, the external port pin signal selected by the FPC input multiplexer may be processed directly by a DCM cell.

The duty cycle of the input signal may be determined by measuring its period length and the width of its ‘Low’ or ‘High’ state. For this purpose, several operations can be started on an input signal edge:

- **Reset**
  The local counter may be reset on rising, falling, or both edges of the signal input line (bits RZE = 1 and/or FZE = 1 in control register DCMCTR). After a reset event, the counter is continuously incremented by the GPTA module clock until the next reset condition occurs. If no reset event is enabled, the counter operates in free running mode, repeatedly increasing from its lower (0_H) to its upper ($FFFFFF_H$) limit.
- **Counter Value Sample**
  The current counter value is stored in the capture register DCMCAV on the rising (RCA = 1) or falling edge (RCA = 0) of the signal input line.
  The current counter value is stored in the capture/compare register DCMCOV when enabled by bit OCA = 1 in control register DCMCTR. This action is triggered on the opposite signal edge selected with RCA.
- **Interrupt Request**
  An interrupt request is generated on a rising input signal edge if enabled by control register bit RRE = 1. An interrupt request generation triggered on the falling input signal edge is selected by FRE = 1. Both edges of the signal input line initiate an interrupt request when both control register bits (FRE, RRE) are set. The interrupt on input signal edges is disabled if both control register bits (FRE, RRE) are cleared. An interrupt request on compare event is generated when the timer content matches the value currently stored in capcom register DCMCOV if enabled by control register bit CRE = 1.
- **Hardware generated Output Pulses**
  A single f_{GPTA} clock pulse on the DCM output line is generated if enabled by control register bit RCK and/or FCK. The additional clock pulse may be triggered by a rising input signal (RCK = 1) and/or by a falling input signal edge (FCK = 1).

The ‘0% or 100% duty cycle’ exception, if no edge or only one edge has been detected, may be handled by a **limit checking** option. The expected input signal’s maximum period length (measured in f_{GPTA} clock ticks) may be loaded into the capcom register

DCMCOV that is continuously compared with the counter value. When the compare interrupt request is enabled (control register bit CRE = 1) and the counter is incremented up to the limit stored in capcom register, the interrupt request is generated and the software must decide what to do.

If the **software** intends to compensate an input pulse backlog, the control register bit QCK should be set to 1 to trigger a single **clock pulse generation** on the DCM output signal line immediately.

7.1.3.4 Digital Phase Locked Loop Cell (PLL)

The GPTA provides a digital Phase Locked Loop Module (PLL) with four trigger inputs hard-wired to the signal output lines of the DCM Modules.

The PLL is locally equipped with a 4 channel input multiplexer, a 16-bit timer, a 16-bit step register, a 24-bit reload register, a 24-bit adder, a 24-bit multiplexer, a 25-bit delta register extended by one sign bit and a PLL control unit (**Figure 7-12**).

Two outputs are available on the PLL Module:

- A signal output line
- An interrupt line triggered by a zero counter value of the 16-bit microtick counter

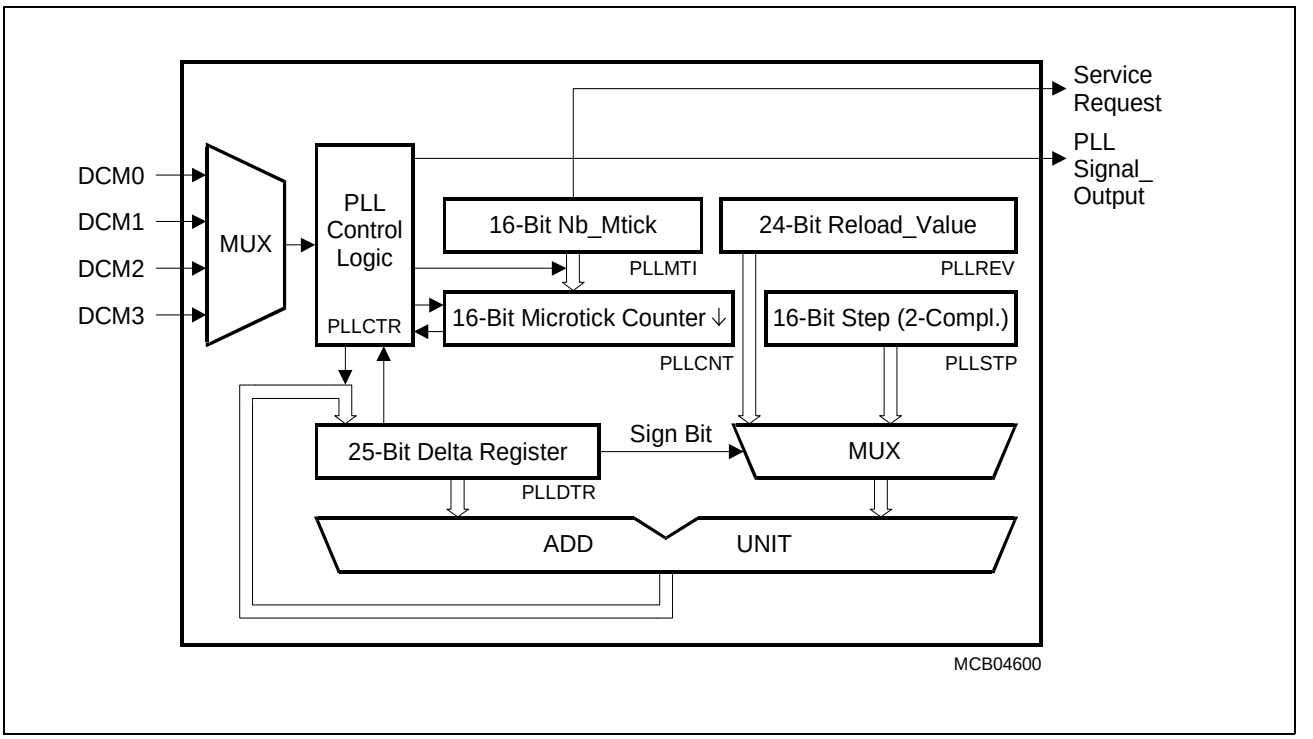


Figure 7-12 Block Diagram of Digital PLL Cell

The PLL Module provides a frequency multiplier function to be applied on the input signal. An input signal edge is used as trigger to generate a programmable number of GPTA clocks on the output signal line. The PLL control unit distributes the desired number of GPTA clocks in regular time intervals over the input signal period length. The PLL can automatically follow an acceleration or deceleration of the input signal. Alternatively, an external software routine may handle the input signal's period length variation.

The desired input signal channel is selected by programming bit field MUX in control register PLLCTR. The number of output pulses to be generated within one input signal period must be stored in the microtick counter and (coded in 2-complement data format) in the step counter. The PLLREV register must be programmed with a reload value. This value is calculated by subtracting the number of output pulses to be generated within one

input signal period from the input signal's period length (measured in number of f_{GPTA} clocks). An automatic compensation of an input signal acceleration or deceleration is enabled by setting the PLL control register bit AEN = 1 (Automatic End Mode). After disabling the Automatic End Mode, the PLL continuously generates output pulses without synchronization to an input signal edge. An interrupt is generated if the counter for the number of remaining output signal pulses decrements to zero and the control register bit REN is set.

Steady Input Signal Example

In the following example, the input signal's period length is 13 GPTA clock periods, which should be subdivided into three equally spaced sections. The reload value to be stored in PLLREV register is calculated to $0A_H$ (10 = 13 - 3). The PLLMTI register (number of output pulses) is loaded with 03_H and its 2-complement representation ($FFFD_H$) is written into the PLLSTP register.

After a reset, a state machine driven by the GPTA module clock, updates the Delta register with the reload value. Afterwards, the PLLSTP register's contents are continuously added to the Delta register value (**Figure 7-13**). In fact, the difference between both values is computed and stored in the PLLDTR register again, because the PLLSTP register has been loaded with a negative value (2-complement data format). When the PLLDTR register has been decremented to a negative value, the reload register contents are added to Delta register's current contents.

A rising edge detected on the selected input signal triggers the microtick counter to load the number of requested output pulses (PLLMTI register contents). When a negative contents of the PLLDTR register is detected, the microtick counter is decremented by one. In 'Automatic Mode' (AEN = 1), the output pulse generation is stopped when the microtick counter reaches zero.

The period length of a single output pulse varies between four and five f_{GPTA} clocks; the maximum period length variation of output pulses is restricted to one f_{GPTA} clock. The total period length of all three output pulses, generated by one PLL loop, corresponds to the input signal period width (5 + 4 + 4 = 13 f_{GPTA} clocks).

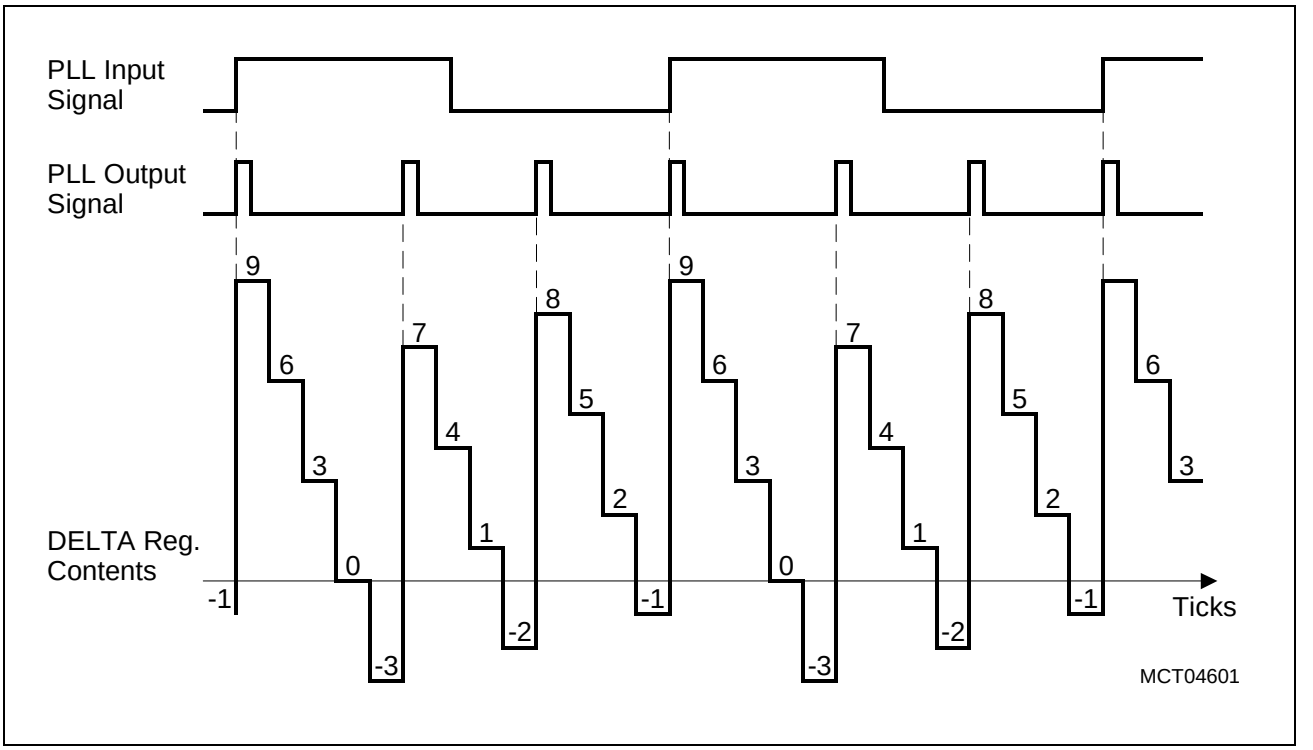


Figure 7-13 Digital PLL Steady State Simulation

This implementation presents a valuable advantage compared to classic PLL implementation. Indeed, the generated microticks are equally distributed. The division reminder is distributed to several clocks instead of adding this reminder to the last pulse clock of the period.

The diagram below illustrates this advantage. Considering a period of 15 clock pulses to be divided by a factor of 4, it gives a result of 3 with a reminder equal to 3. The reload value is calculated to $0B_H$ (11 = 15 - 4). The number of output pulses is equal to 4 and its 2-complement representation ($FFFC_H$) is written into the step register.

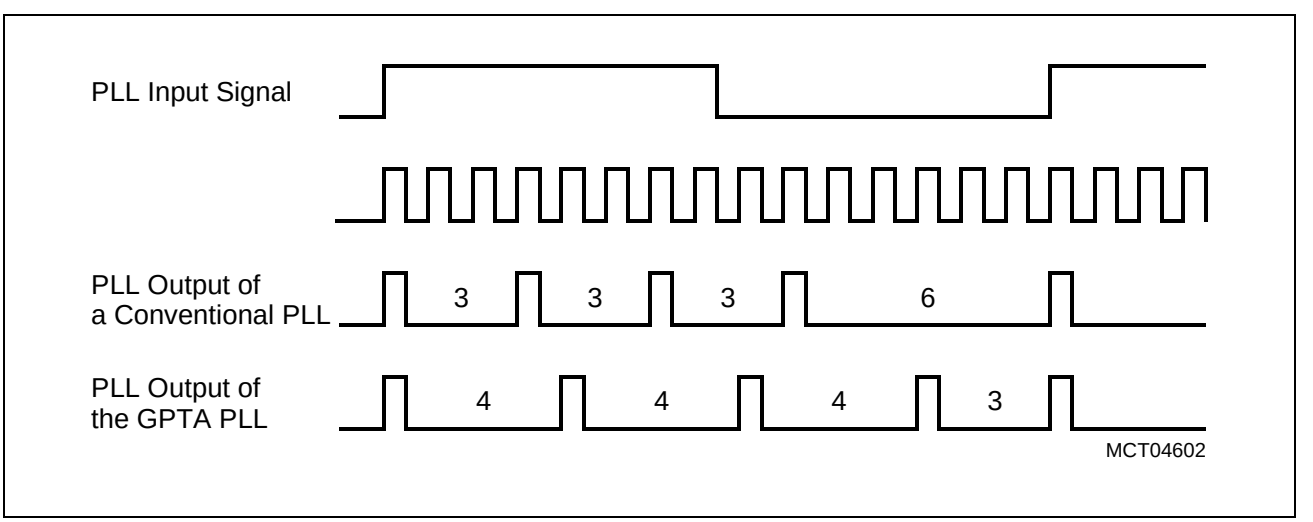


Figure 7-14 Advantage of the GPTA PLL

Input Signal Acceleration and Deceleration

The consequence of an input signal acceleration or deceleration can be compensated either automatically or by an external software routine. It detects an input signal's period length variation by comparing the current period length (measured in the associated DCM cell) with the expected period length used as calculation base for the REV register contents.

- Compensation of input signal deceleration
  - Compensation by PLL Automatic End Mode
    If Automatic End Mode is enabled by setting control register bit AEN to 1, the PLL stops at the calculated end of the current input signal period. Due to the deceleration, the rising edge of the following input signal period is delayed, starting the next PLL operation later than expected. A gap occurs between the last output pulse of the current input signal period and the first pulse of the following one (**Figure 7-15**).
  - Compensation by Software
    After disabling the Automatic End Mode, the PLL generates output pulses without synchronization to an input signal edge. In case of a deceleration, more output pulses than calculated are generated during one input signal period. Several algorithms can be implemented to compensate the surplus of generated output pulses:
    The length of the current input signal period has been underestimated by a certain number of f_{GPTA} clock periods. This deficit could be added to the calculated length of the next input signal period.
    The PLL can continue to operate with the old input signal period length estimation, but the number of output pulses to be generated during the next input clock period may be decreased by the surplus of output pulses initiated during the last signal period.

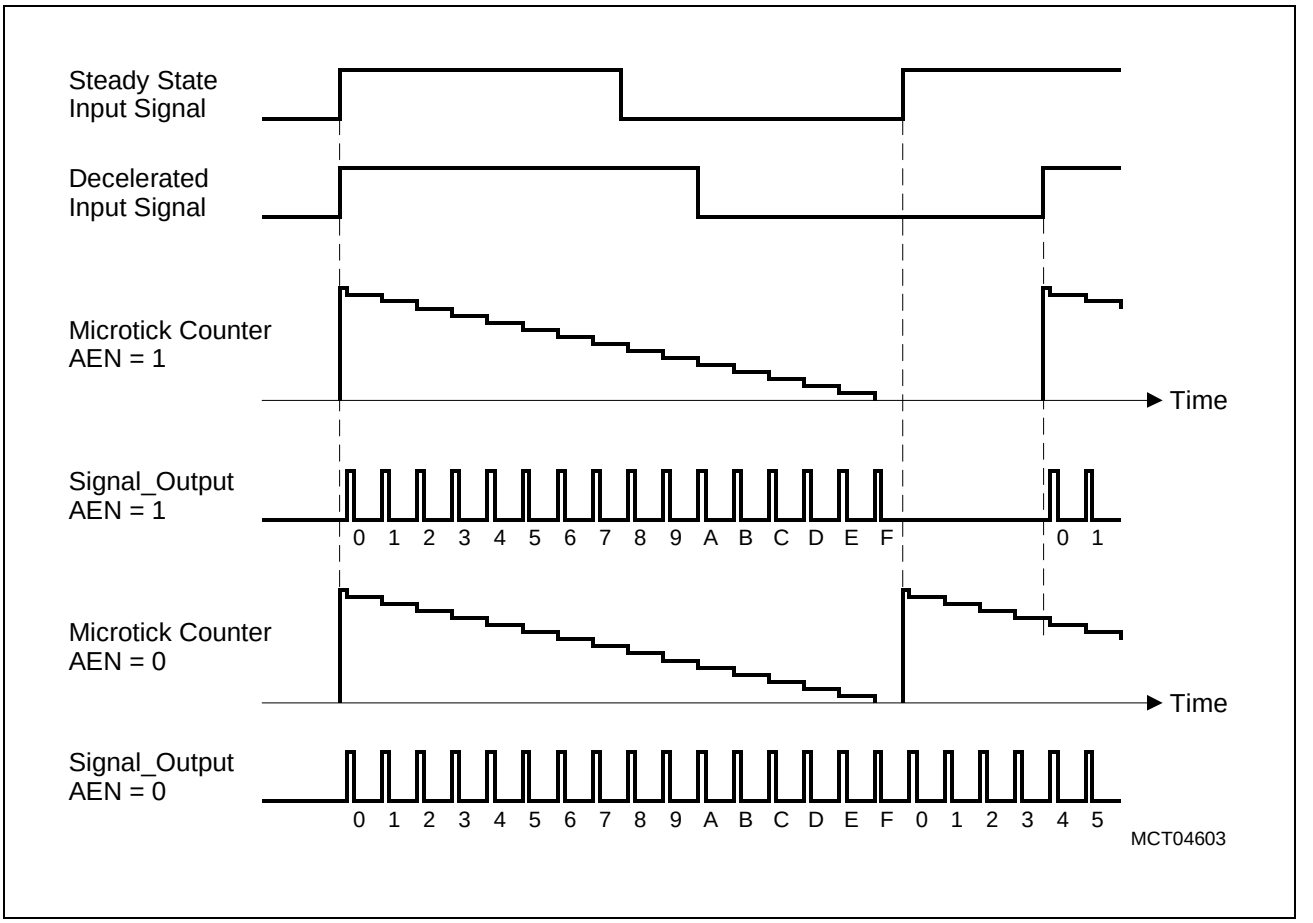


Figure 7-15 Compensation of Input Signal Deceleration

- Compensation of input signal acceleration
  - Compensation by PLL Automatic End Mode
    The next rising edge of the input signal arrives while the counter has not been decremented to zero. The PLL performs all remaining output signal pulses at full speed (f_{GPTA}), when control register bit AEN is set to 1. Afterwards, counter and Delta register are reloaded with their calculated values and the PLL operates at normal speed (**Figure 7-16**).
  - Compensation by Software
    After disabling the Automatic End Mode, the PLL generates fewer output pulses than calculated during one input signal period. Several algorithm can be implemented to compensate for the lack of generated output pulses:
    The length of the current input signal period has been overestimated by a certain number of f_{GPTA} clock periods. This deficit should be subtracted from the calculated length of the next input signal period.
    The PLL can continue to operate with the old input signal period length estimation, but the number of output pulses to be generated during the next input clock period may be increased by the lack of output pulses initiated during the last signal period.

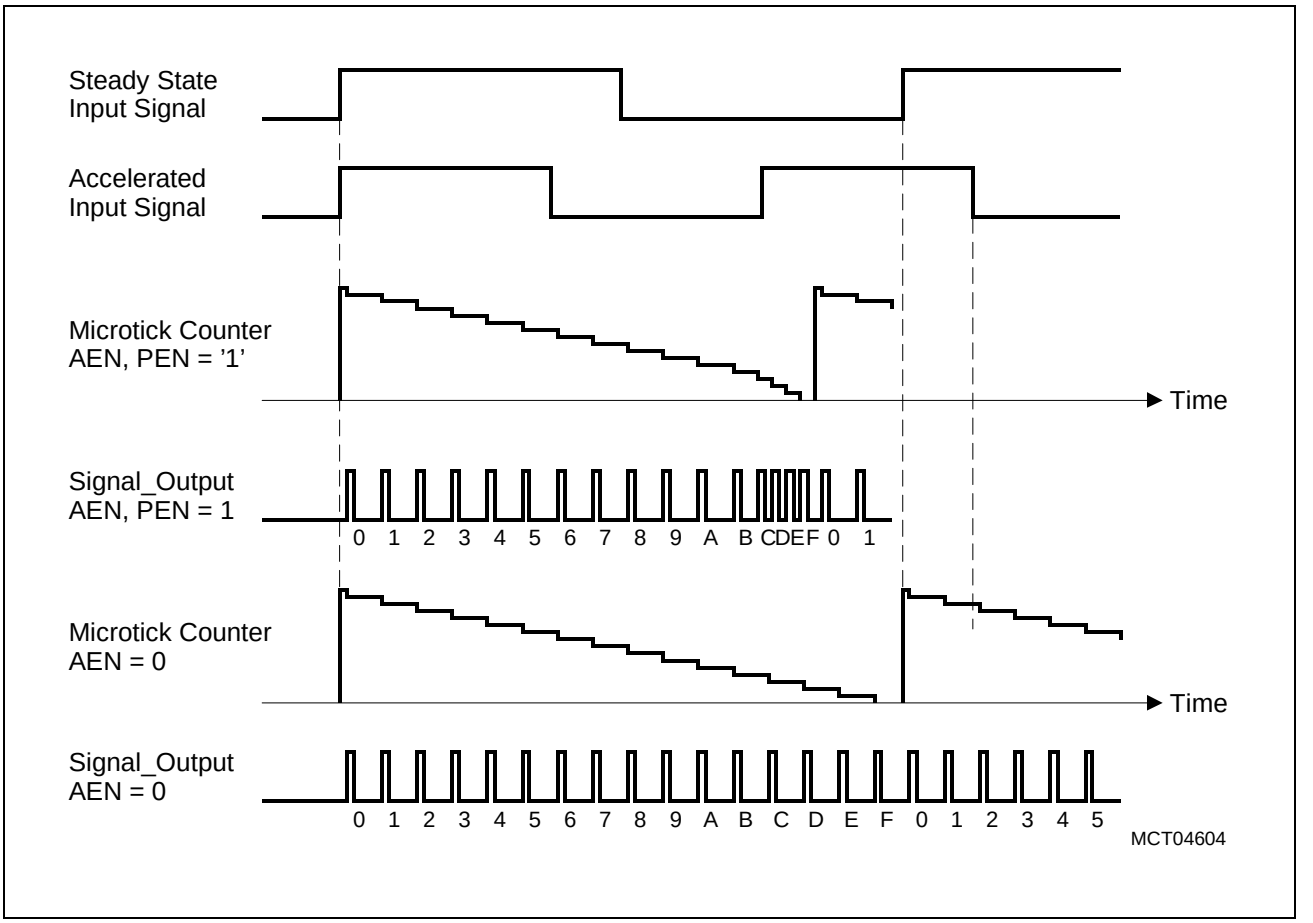


Figure 7-16 Compensation of Input Signal Acceleration

7.1.3.5 Clock Distribution Module (CDM)

The Clock Distribution Module (CDM) provides all local and global timer cells with a variety of different clock signals. The CDM has eight input signals to be distributed:

- Filter and Prescaler Cell FPC1 event output line
- Filter and Prescaler Cell FPC4 event output line
- Duty Cycle Measurement Units DCM0 - DCM3 output signal lines
- PLL output signal line
- GPTA module clock

The CDM is locally equipped with four GPTA clock prescalers and four multiplexers supporting alternate clock sources (**Figure 7-17**).

An 8-bit wide clock bus is implemented as output of the CDM Module:

- CLK0 carries the GPTA module clock,
- CLK1 is directly linked to the PLL output signal,
- CLK2 is driven by a prescaled GPTA module clock or by DCM3 output line,
- CLK3 is directly hooked to DCM2,
- CLK4 is supplied by a prescaled GPTA module clock or by DCM1 output line,
- CLK5 is directly linked to DCM0,
- CLK6 is driven by a prescaled GPTA module clock or by FPC1 event output line,
- CLK7 is supplied by a prescaled GPTA module clock or by a FPC4 event output line.

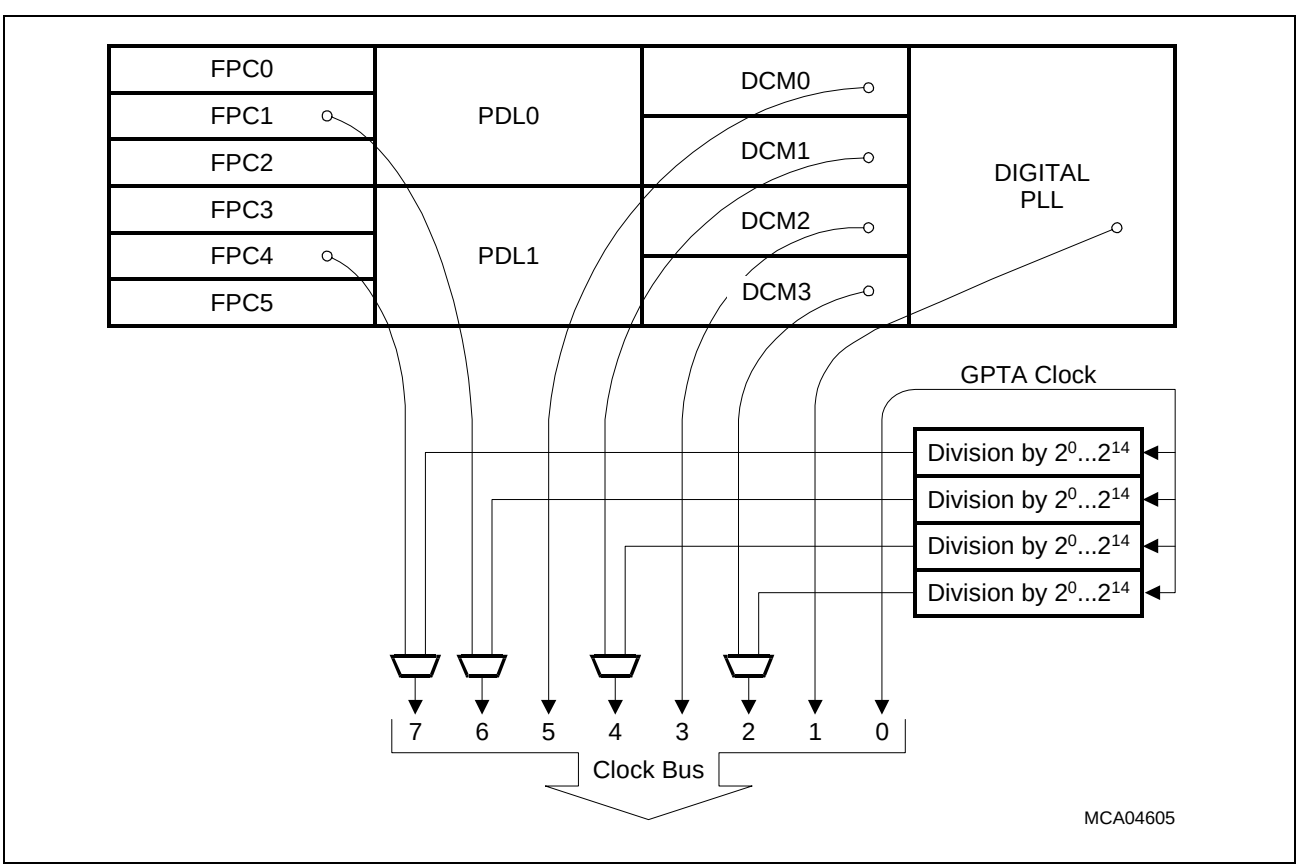


Figure 7-17 Clock Bus Sources

The prescaler divides the GPTA module clock by a programmable modulus 2. The exponent n of the division factor 2^n must be set up in bit fields DFA02, DFA04, DFA06 and DFA07 of control register CDUCTR. An exponent value of ‘1111’ disables the related prescaler and selects the DCM3, DCM1, FPC1 and/or FPC4 output lines as alternate sources of clock lines 2, 4, 6 and 7.

7.1.4 Signal Generation Unit

The Signal Generation Unit contains three types of modules. They are explained in detail in the next chapters.

- Global Timers (GT)
- Global Timer Cell (GTC)
- Local Timer Cell (LTC)

7.1.4.1 Global Timers (GT)

The GPTA provides two global 24-bit timers (GT) connected to the 8-bit wide clock bus.

A GT is locally equipped with one multiplexer selecting a clock source, a 24-bit up-counter, a 24-bit reload register, and a 24-bit greater/equal comparator (**Figure 7-18**).

A 24-bit wide local timer value bus and three flag lines are implemented as output of the GT Module:

- the local timer value bus, carrying the current GT counter value, is fed-through to all 32 global timer capture/compare cells (GTCs),
- a TEV line indicates a counter update,
- a TGE line flags the result of a compare operation,
- SQT is used as interrupt line triggered by a timer overflow.

The global timer may be initialized with a start value, written by an external software routine to the GTTIM register. The GT is incremented by a clock signal selected from the 8-bit clock bus via bit field MUX in control register GTCTR. When the timer overflows, the contents of the GTREV reload register is copied to GTTIM and an interrupt is generated, if bit REN is set in control register GTCTR. A free running timer may be achieved by programming the GTREV register to zero.

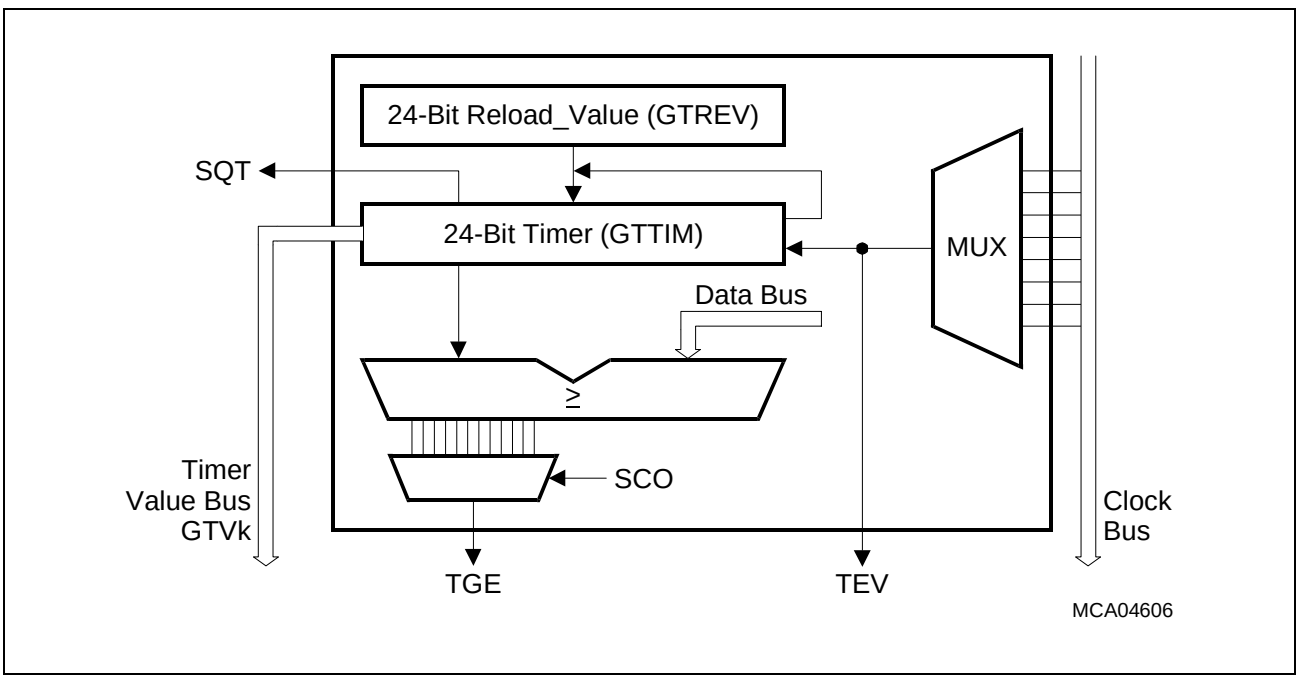


Figure 7-18 Block Diagram of Global Timer (GT)

The flag “Timer Event” (TEV) is set if the timer value changes due to a clock edge, a reload operation or a software write access. TEV is used to trigger a compare operation within all related GTCs, re-checking the equality of their compare register contents and the updated timer value.

Scalable Signed Greater or Equal Compare

Introduction

This chapter explains the **classical timer update problem**, and the solutions supported by the GPTA.

The two Global Timers (GT) embedded into the GPTA include a 24-bit greater/equal comparator. This comparator unit performs compare operations between the GT timer contents and the data value found on the GPTA-internal data bus (coming from a GTC compare register update). The goal of this comparator is to be able to perform an action immediately if the compare cell is updated with a new threshold but the timer has already passed this value (**Figure 7-19**).

A timer is running and a new threshold (value T) is set.

The different points Px represent different cases of present time. When at P1 or P2, the moment represented by T lies in the future and no action is yet required. When at P3 or P4, the moment represented by T lies in the past, and an action is required immediately.

The problem is then to determine if the threshold T has been passed or not.

Considering an **infinite counter**, the situation is simple. The evaluation consists in determining if point P is before or after T.

Considering a **reloaded counter**, as the timer rolls over at its maximum value, the situation is more complex.

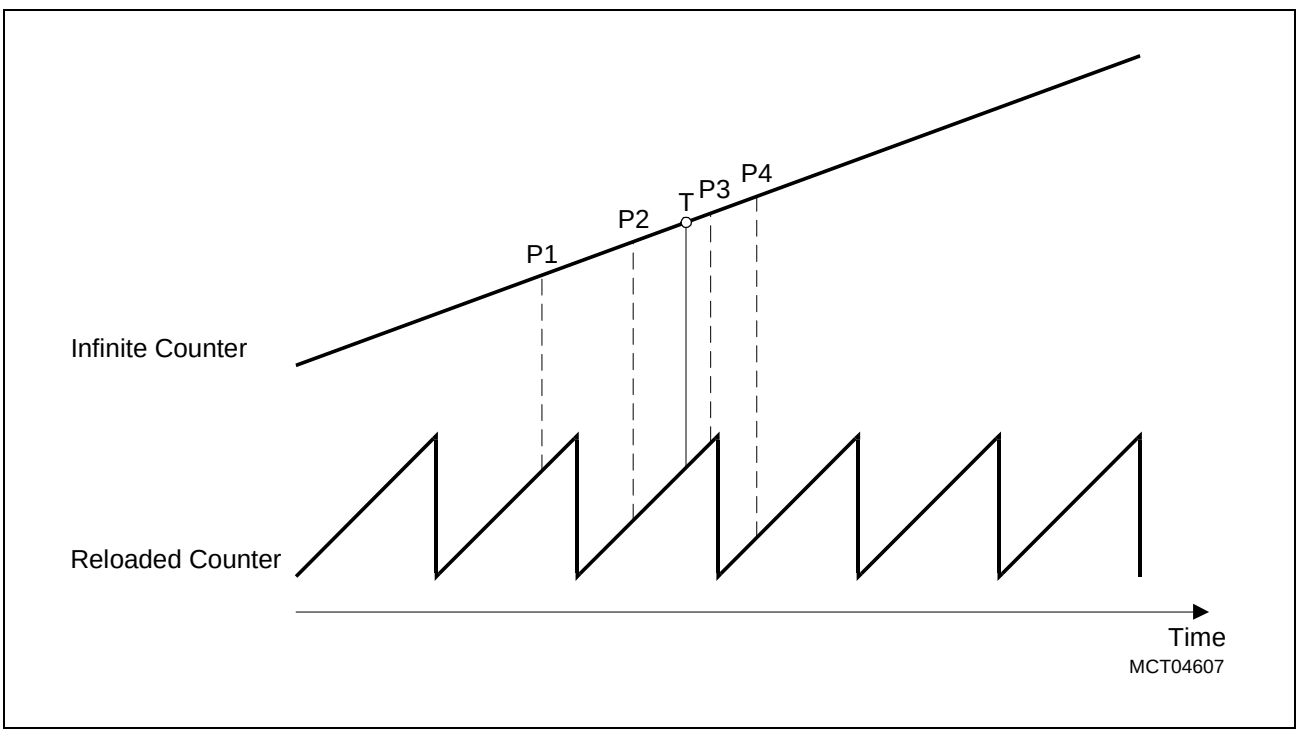


Figure 7-19 Greater/Equal Concept

The **observation window** defines the space in time where writing the value T to the comparator will lead to correct observation (meaning, there is an event if “After”; there is no event if “Before”). Considering an observation window, an event (threshold T) is programmed and then the window is split into two windows, the “After” window and the “Before” window (**Figure 7-20**). If the timer lies in the “After” window at the time of programming the threshold, the event is performed immediately. If it lies in the “Before” window, the event will happen later when the timer reaches the threshold T. The “Before” window refers to a “prediction range”, and the “After” window refers to the “history buffer”.

From a practical point of view, once the value T is determined, it is necessary to calculate the observation window (position and width). Before updating the value T, the application must assure that the observation window was entered but has not yet been left.

The width of the **observation window** cannot exceed the timer period. To support reloaded counters where the overflow can occur within the observation window, a **signed** comparison is performed.

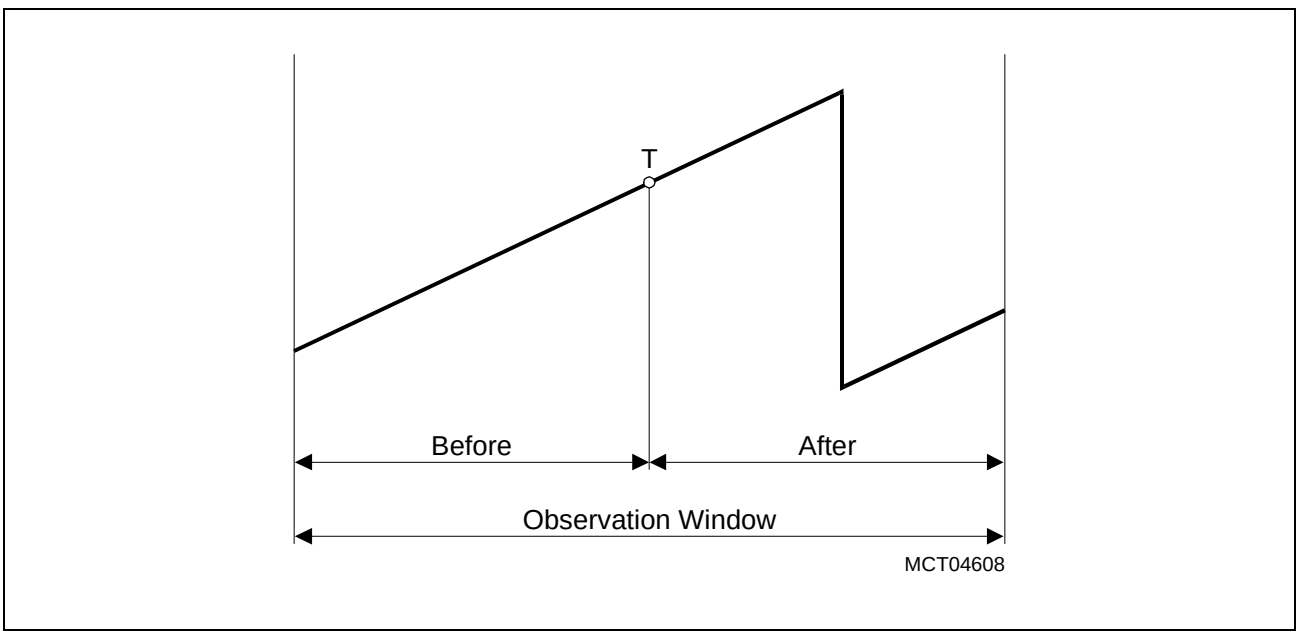


Figure 7-20 Before and After Windows

Comparison Between Unsigned and Signed Compare

To be able to support different timer periods and to support correct observation even beyond timer overflow, the GPTA embeds the **scalable** and **signed** greater/equal comparator. Using a signed comparison allows one overflow of the timer to occur within the observation window. This is illustrated in **Figure 7-21**.

Using a signed compare in order to take into account the timer overflow, the comparator window is introduced. The comparator window is centered to the point T and its width can be selected by the user.

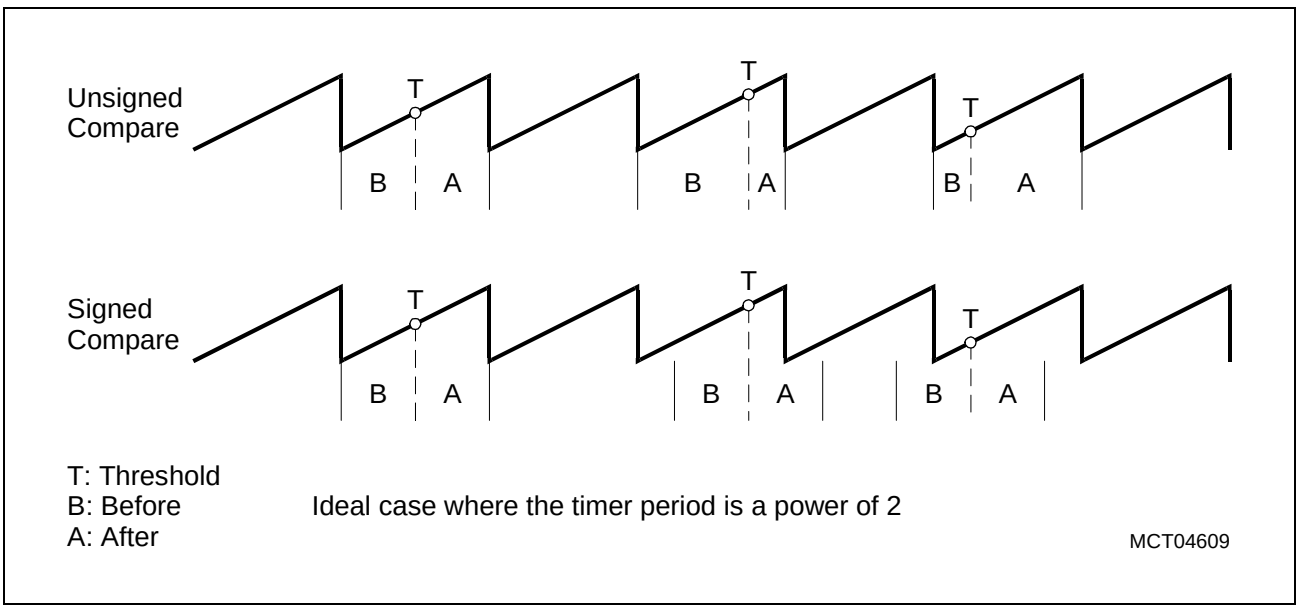


Figure 7-21 Unsigned Versus Signed Compare

When the timer range is a multiple of 2 and since the comparator is scalable, the observation window and the comparator window are identical. See **Figure 7-22**.

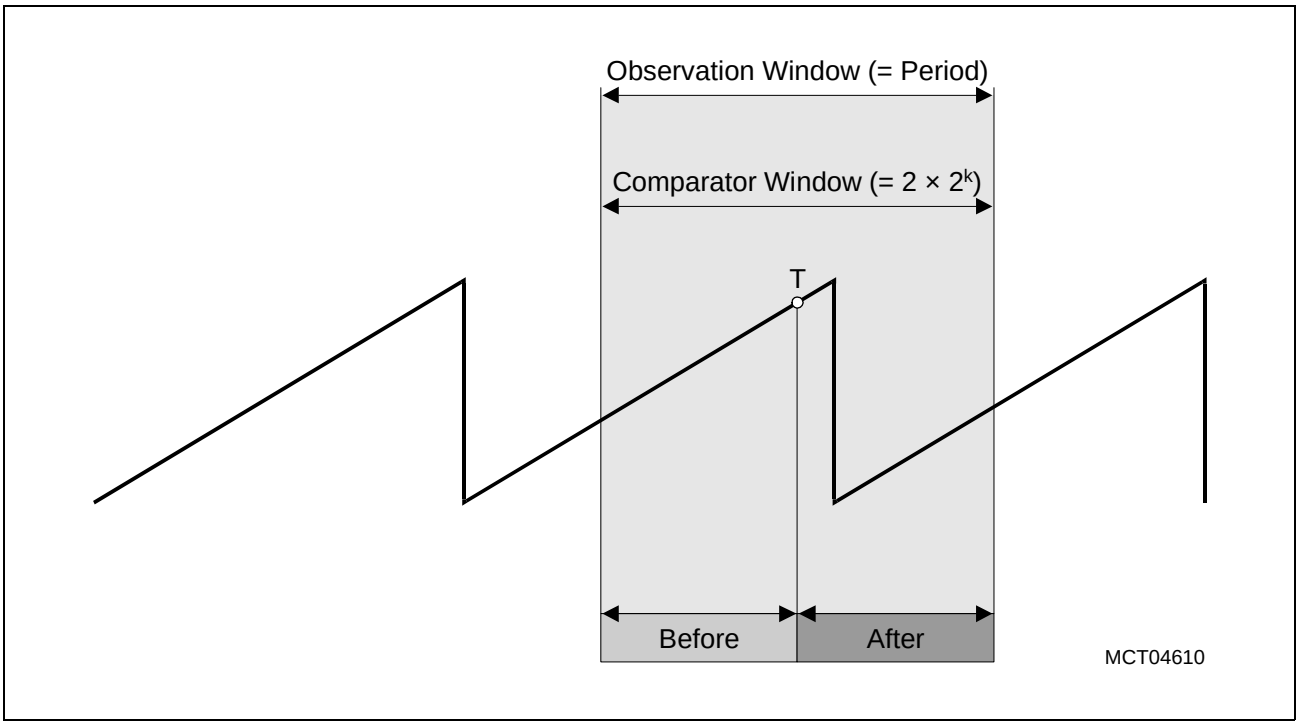


Figure 7-22 Observation and Comparator Windows (timer is a multiple of 2)

The scalable and signed greater/equal comparator scheme leads to a limitation that must be considered when programming the GPTA Module. If the timer range is not a power of 2, the comparator window (always a power of 2) will no longer match the timer period. This will impact the observation window as described in the following paragraph.

Observation window for reloaded timers (period is not a multiple of 2).

In that case, the comparator window must exceed the timer period. The user must find the comparator window (by selecting the scale factor k) which fits best the timer period.

The following equation must apply:

$$2^k < \text{Period} \leq 2 \times 2^k \qquad [7\text{-}1]$$

Figure 7-23 and **Figure 7-24** show that one part of the comparator window must be discarded in order to avoid inconsistency, resulting in the observation window.

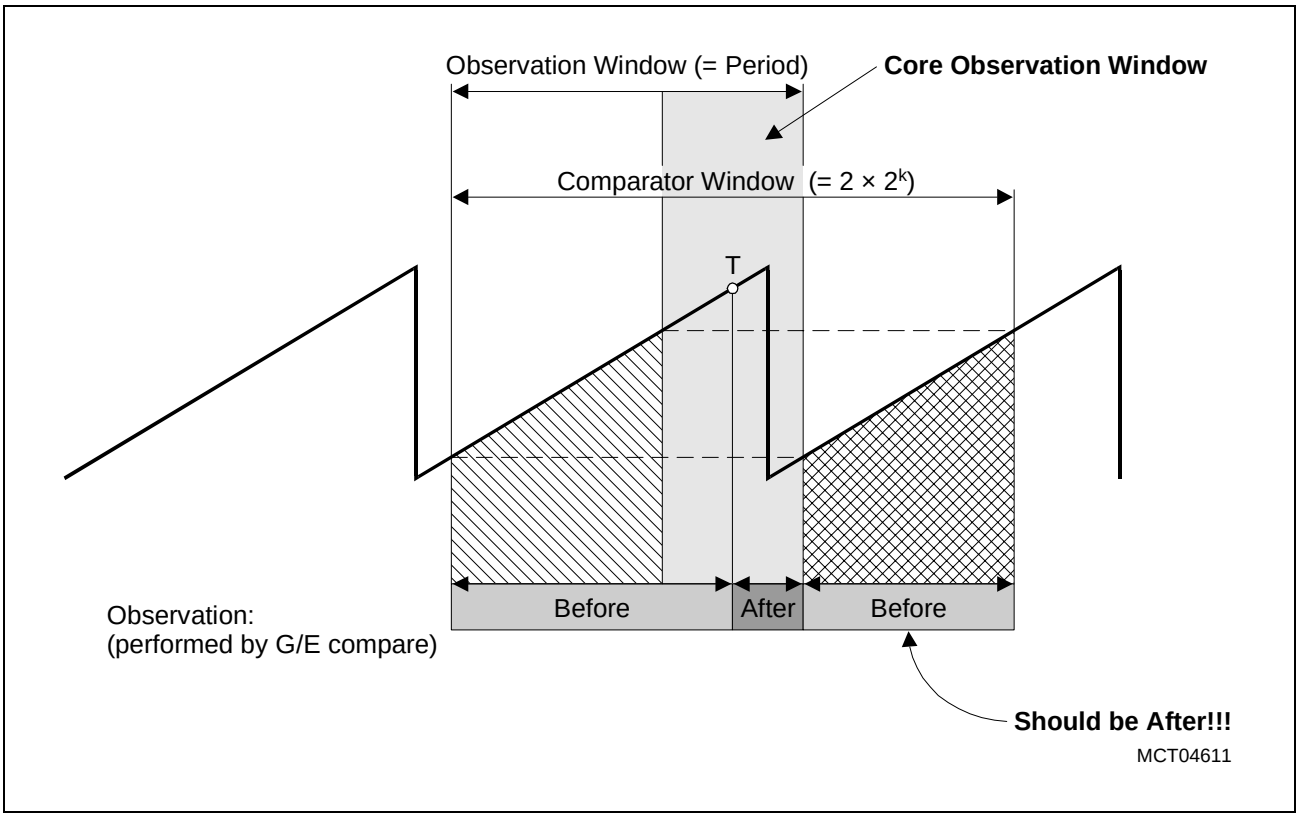


Figure 7-23 Observation Window when Threshold T is High

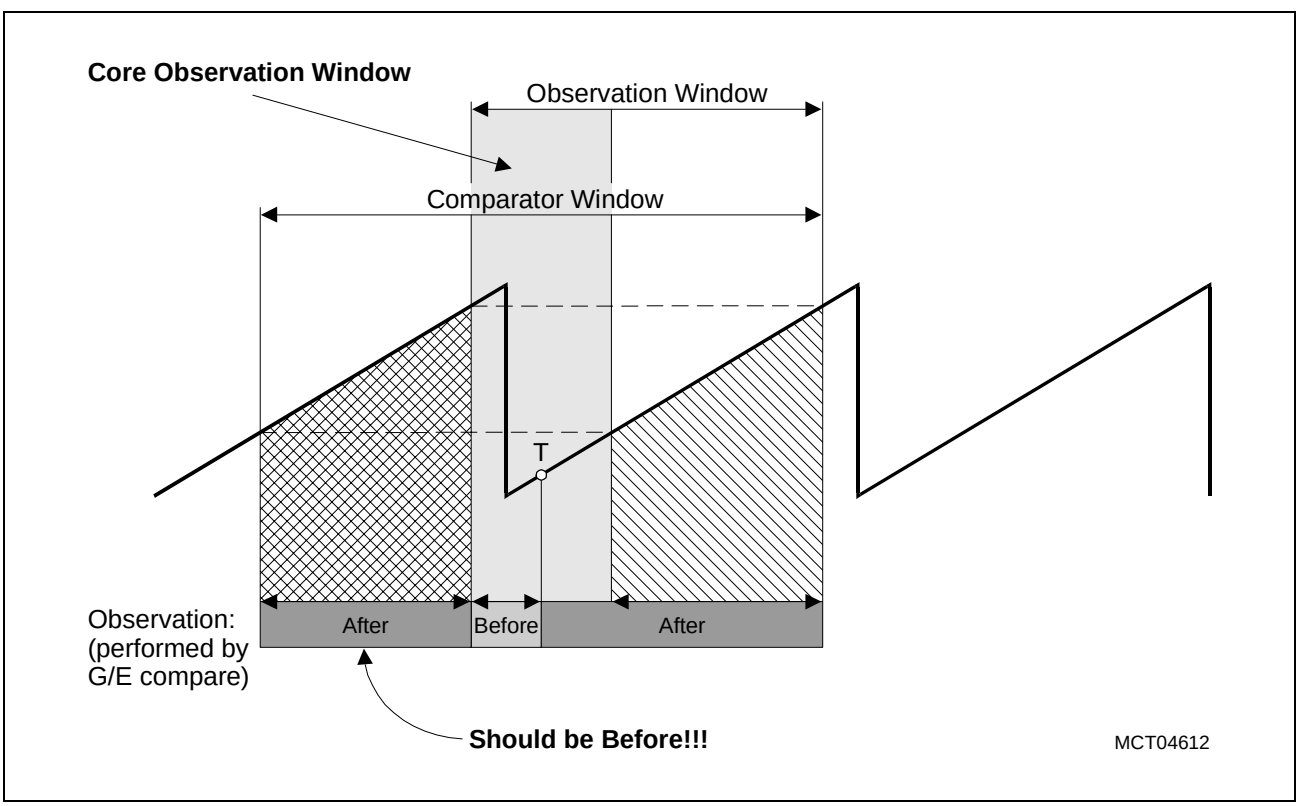


Figure 7-24 Observation Window when Threshold T is Low

A comparison of the previous diagrams shows that the position of the observation window with respect to T is dependent on the value of T itself. That means the user, before updating the comparator with T, needs to calculate the observation window as a function of T. To avoid this calculation, a **core observation window** can be defined that is independent of T. It will always be centered on T whatever its value. However, one particularity exists when using the core observation window: the size of the core observation window varies depending on two static values: the timer period and the comparator window's sizes. In particular, the core observation window reduces as the value of the timer period is just after a multiple of 2. This is shown in **Figure 7-26** below.

For any timer period (whatever the range) and any threshold position, a symmetrical core observation window of a statically defined size can be determined.

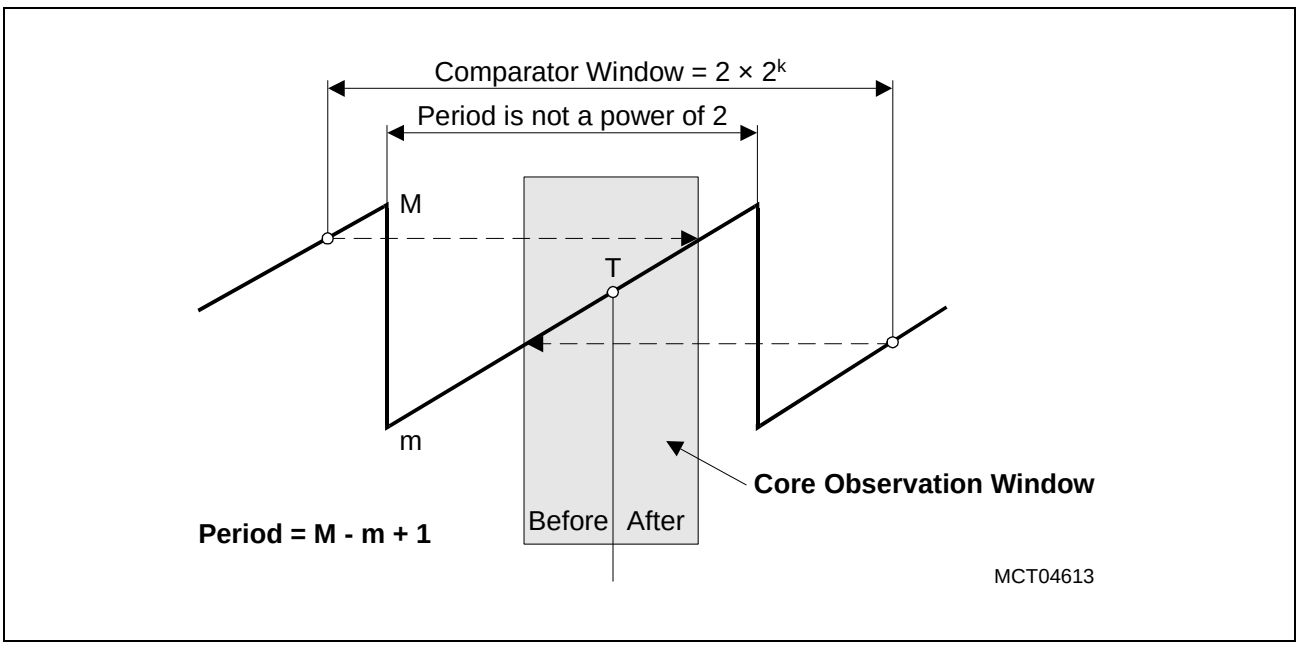


Figure 7-25 The Core Observation Window

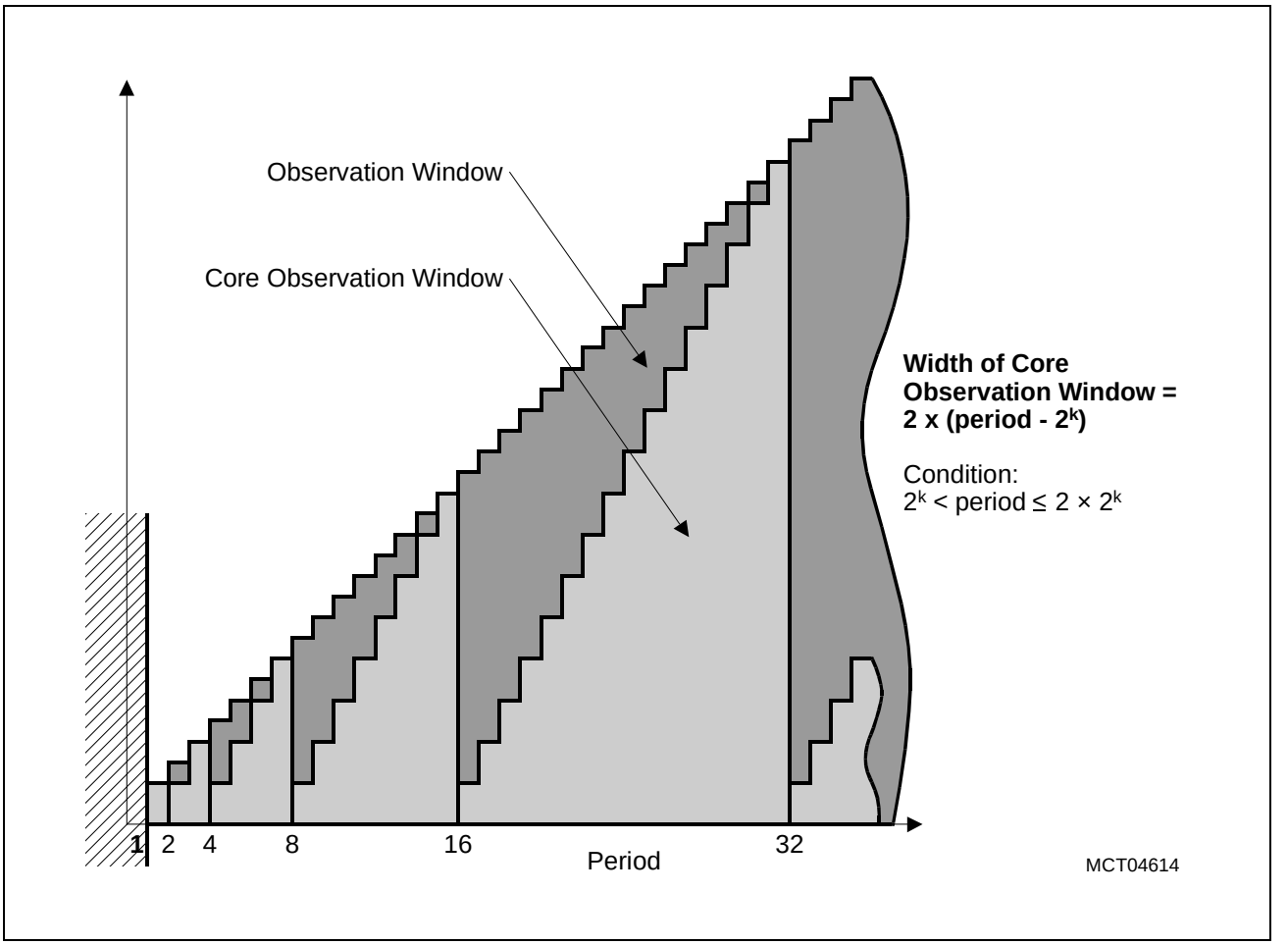


Figure 7-26 Core Observation Window Sizes Versus Period Sizes

Implementation

The hardware implementation of the **scalable and Signed/Unsigned** Greater/Equal compare is illustrated in **Figure 7-27**. The function consists of subtracting the threshold T from the GT timer value. The result is in 2s complement format. The result's sign bit and the 15 most significant bits are at disposal for observation. One of those bits is selected according to the mode of operation (Unsigned or Signed) and the period length (bit field SCO in GTCTR register). This bit drives the TGE (Timer Greater Equal) flag.

Unsigned compare: Select Sign bit (SCO = $0F_H$)

Signed compare: Select one of the 15 most significant result bits (SCO = 00_H to $0E_H$)

Note: How to choose one of the 15 bits is explained later.

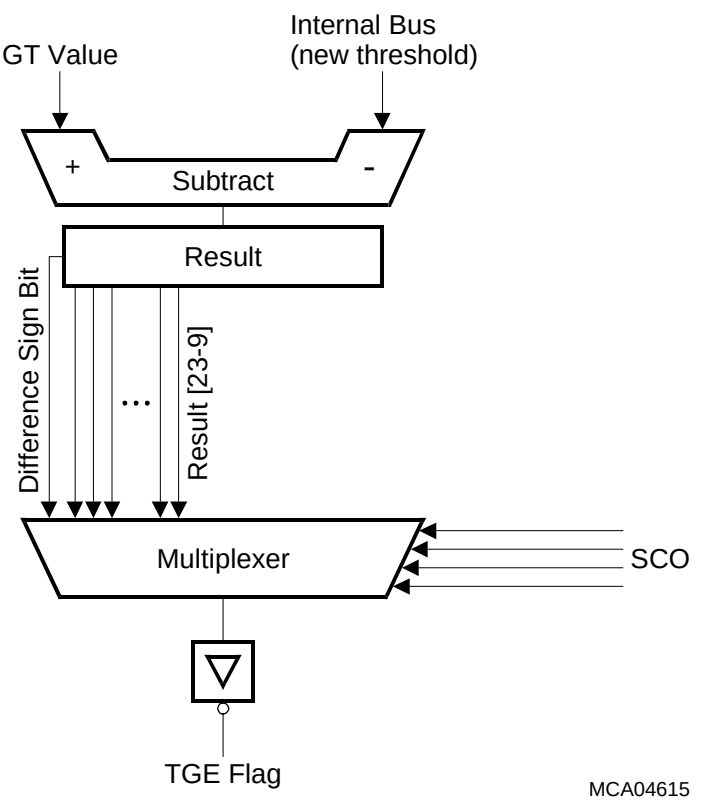


Figure 7-27 Comparator Implemented by a Subtraction Unit

The interpretation of the selected result bit is provided in the following simple example: For a 4-bit timer, the subtraction of the threshold T from the timer value, leads to a 4-bit signed result, as illustrated in **Figure 7-28**. This example is selected for simplicity although 4-bit periods are not covered by the implementation.

When using Unsigned compare, the sign bit S is selected. If it equals 0, the result is positive, indicating that the timer is greater or equal the threshold, and hence **After**. If it equals 1, the result is negative, and the observation indicates **Before**.

When using Signed compare, the result bit R_3 can be selected and interpreted, provided that the timer period is at least 9. Here, the range of the result can be split into four sub-ranges. Because the result is in 2s complement format, a value of 0 for R_3 is interpreted as **After**, and a value of 1 is interpreted as **Before**. A comparison of **Figure 7-28** and **Figure 7-29** shows why this proceeding leads to correct interpretation within the observation window. **Figure 7-29** shows the case of a period equal a multiple of 2.

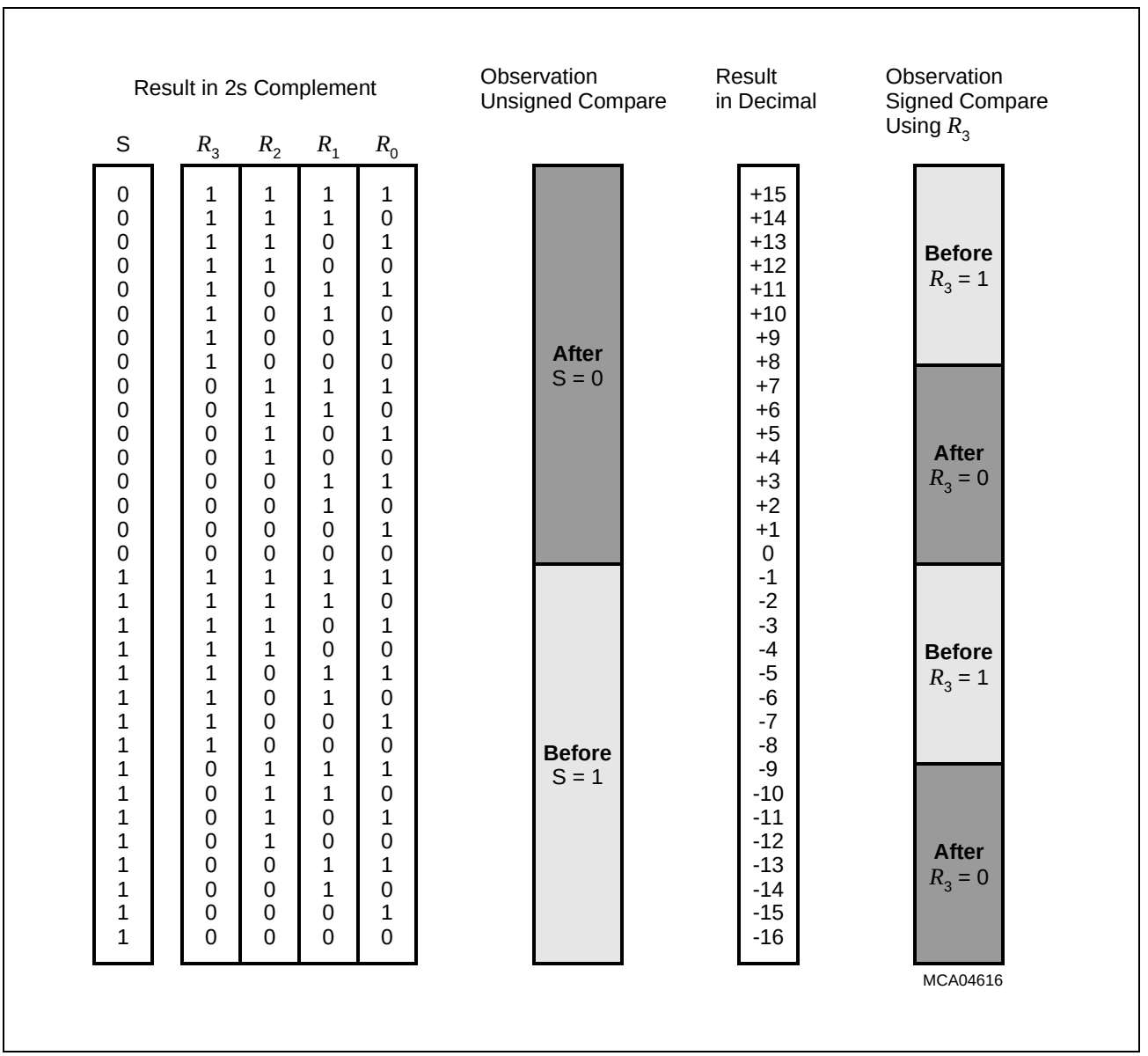


Figure 7-28 Result and Observation for a 4-Bit Timer

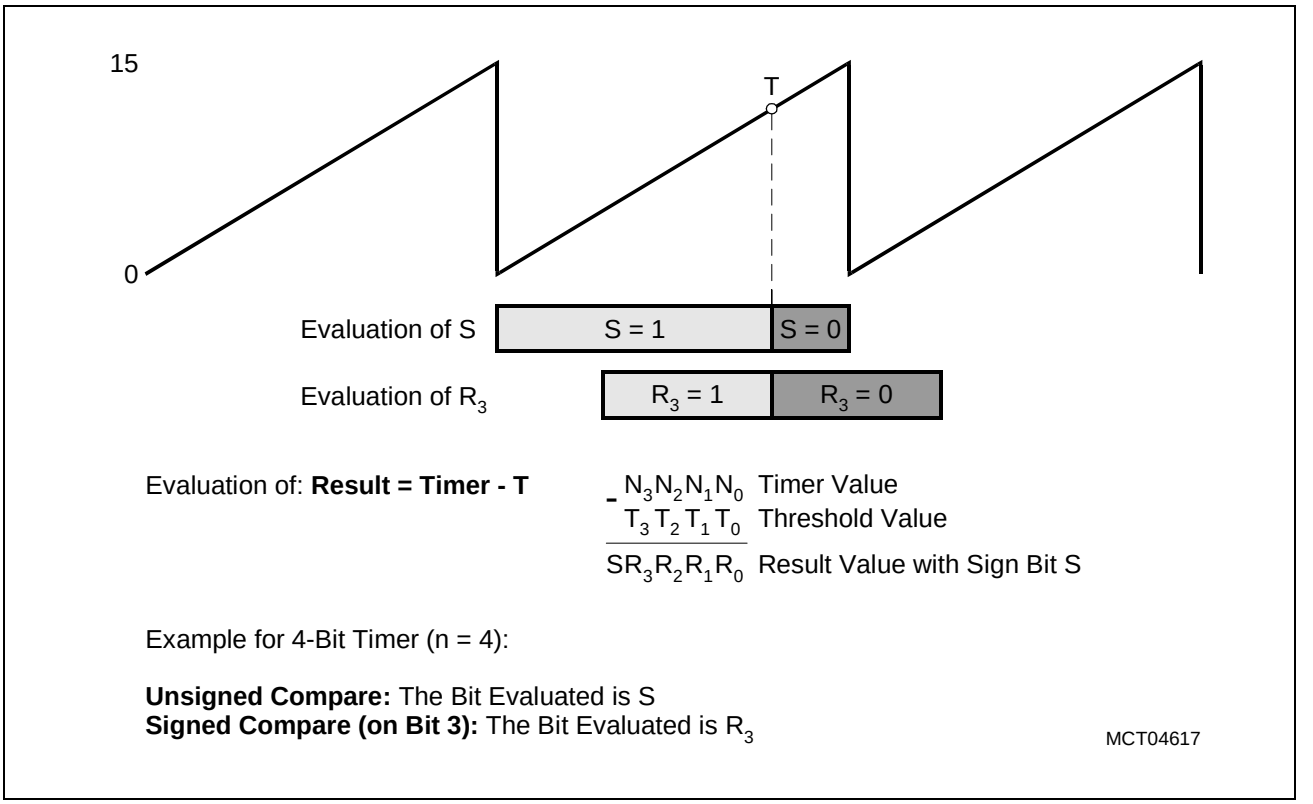


Figure 7-29 Result and Observation (Period = 16)

Figure 7-30 shows the case of a period of 12 which is not a power of 2. Here again the Table in **Figure 7-28** applies.

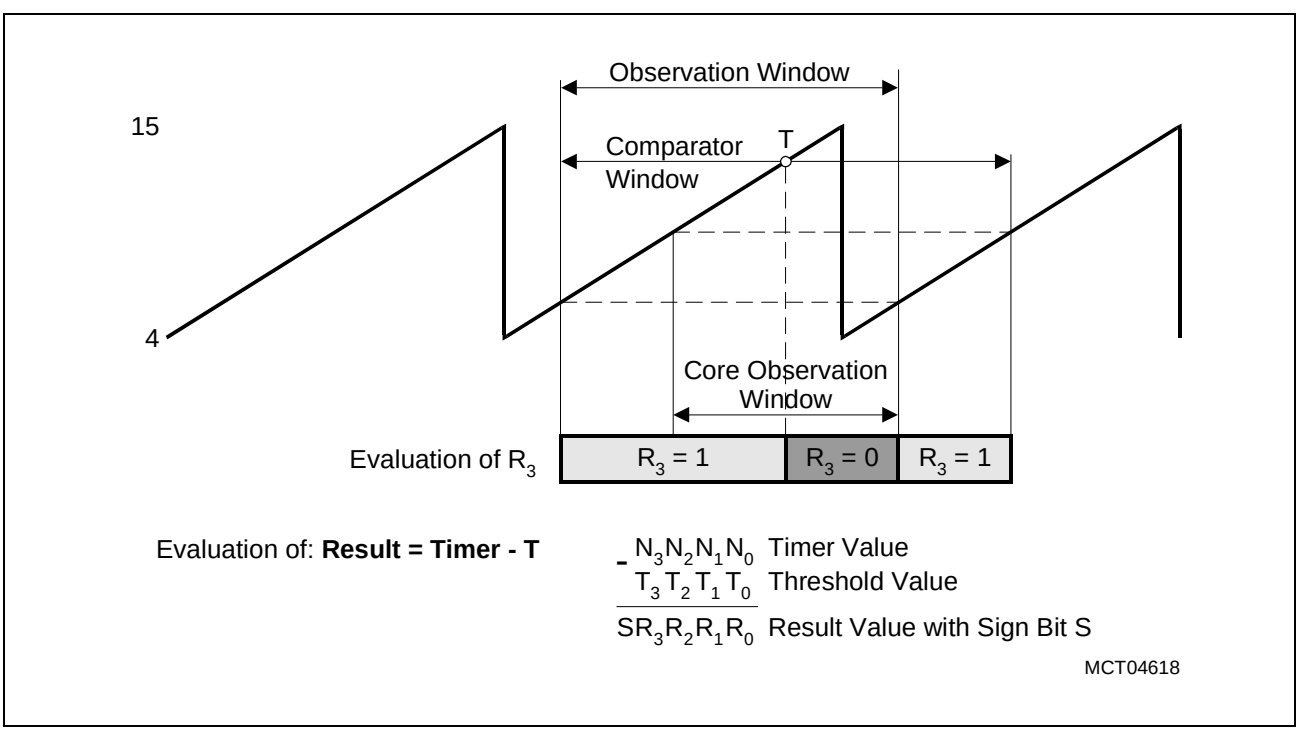


Figure 7-30 Result and Observation (Period = 12)

The previous examples show that the result bit to select for observation (R_3) corresponds to the comparator window's size (k = 3).

Considering the case in which the period is not a multiple of 2, choose a comparator window whose width is between 1 and 2 times the timer period:

$$2^k < \text{Period} \leq 2 \times 2^k \quad [7\text{-}2]$$

In no case the comparator window may be equal to or greater than twice the period.

k represents the Result bit to select.

How to proceed:

a) Unsigned greater/equal compare:

SCO bit field = $0F_H$ (15_d)

Thereby, the sign bit of the result is selected to drive TGE flag.

This setting is valid for all possible periods. The observation window always matches the period.

b) Signed greater/equal compare:

Depending on the period, the appropriate k is selected, so that:

$$\text{Period} = M - m + 1 \ (= \text{Max} - \text{Min} + 1) \quad [7\text{-}3]$$

$$2^k < \text{Period} \leq 2 \times 2^k \quad [7\text{-}4]$$

SCO bit field = 0 to $0E_H$ (0 to 14_d)

Thereby, the result bit R_k is selected to drive TGE flag.

This setting is possible for periods greater than 512.

Table 7-1 Period Range Depending on Selected k

$2^k < \text{Period} \leq 2 \times 2^k$	k	SCO Bit Field (decimal)
0 < period ≤ 512	Not covered by implementation	
512 < period ≤ 1024	9	0
1024 < period ≤ 2048	10	1
2048 < period ≤ 4096	11	2
4096 < period ≤ 8192	12	3
8192 < period ≤ 16384	13	4
16384 < period ≤ 32768	14	5
32768 < period ≤ 65536	15	6
65536 < period ≤ 131072	16	7
131072 < period ≤ 262144	17	8
262144 < period ≤ 524288	18	9
524288 < period ≤ 1048576	19	10
1048576 < period ≤ 2097152	20	11
2097152 < period ≤ 4194304	21	12
4194304 < period ≤ 8388608	22	13
8388608 < period ≤ 16777216	23	14

The width of the core observation window is defined by:

$$2 \times (\text{period} - 2^k) \qquad [7\text{-}5]$$

As a consequence, the width of the “Before” window within the core observation window is (period - 2^k) and the width of the “After” window within the core observation window is (period - 2^k), including the value T.

Additional Information: Illustration on the General Case

The previous section illustrated the G/E compare for the particular case of a 4-bit timer. The purpose of this section is to describe the implementation from a general point of view, that is, for a timer period equal M - m + 1.

In the following figures, the X axis indicates the timer value (elapsing time) and the Y axis indicates the threshold value T. The 45° line starting at (m,m) represents the position in time of T. The graphic shows the observation performed by the hardware for all cases of T ($m \leq T \leq M$).

Figure 7-31 illustrates the Unsigned compare. A particular case is shown in which, for a higher value of T, the observation indicates “Before” at the beginning of the period, and until the timer reaches the value T. Thereafter, the observation switches to “After” and remains there until the timer exits the period.

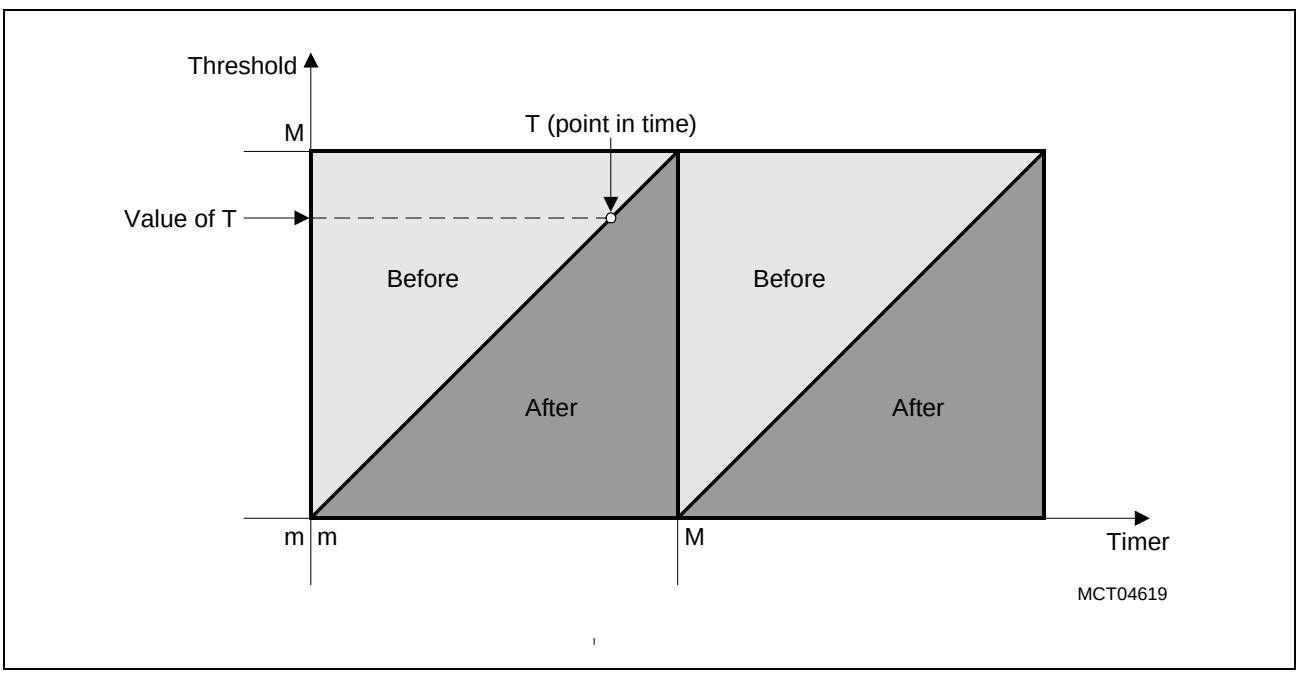


Figure 7-31 Graphical Representation of Unsigned Compare

Figure 7-32 illustrates the Signed compare where the period equals a multiple of 2 (that means M - m + 1 = 2×2^k). In this case, for a higher value of T, the observation indicates “After” at the beginning of the period (not yet inside the observation window). When entering the observation window, “Before” is indicated until the timer reaches the value

T. Thereafter, the observation switches to “After” and remains there until the timer exits the observation window. This graphic can be related to **Table 7-29** where the comparator window equals the period, and the observation window is always centered on the threshold T.

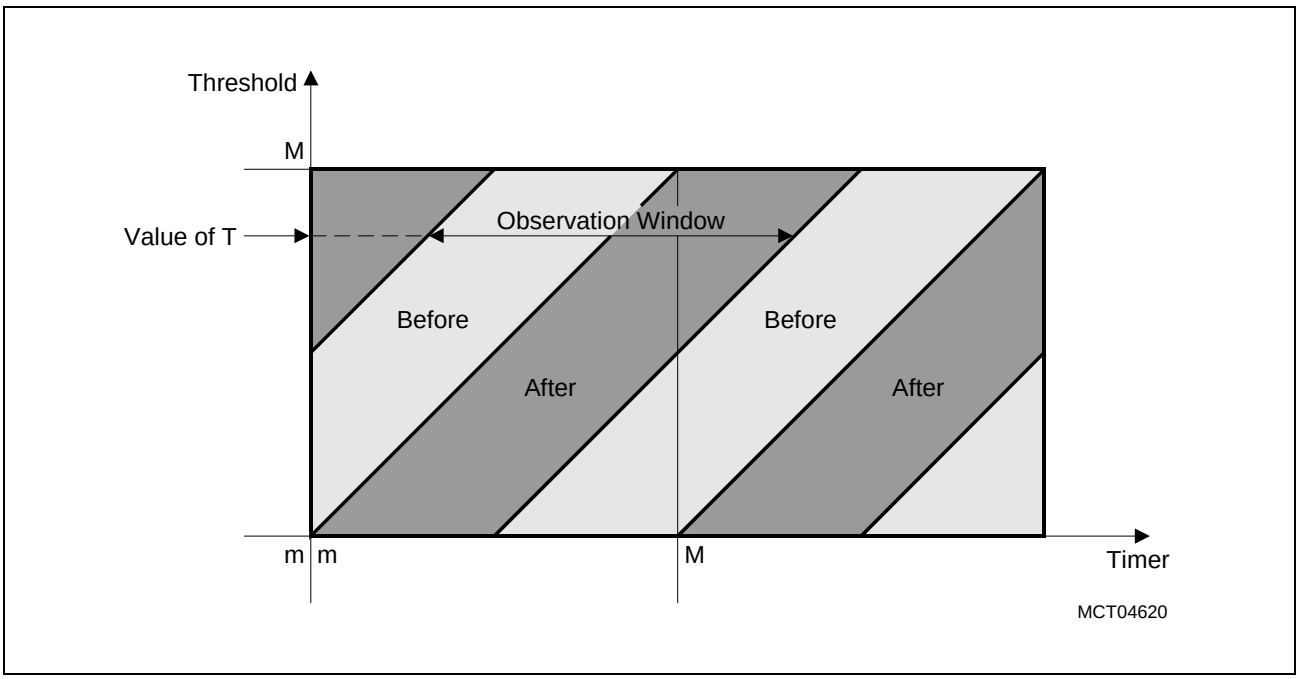


Figure 7-32 Graphical Representation of Signed Compare (Period = 2×2^k)

The **Figure 7-33** illustrates the Signed compare where the period may also be unequal a multiple of 2. The graphical representation of this general case is analogous to the one described in **Figure 7-23**.

If the period is not a multiple of 2, the graphical representation of the Signed compare shows a discontinuity in the “Before” and “After” ranges. Indeed, the widths of the “Before” and “After” windows are not constant, as they depend on the value T. As a consequence, the observation window is not centered on T. The result is that the position of the observation window would have to be re-evaluated for each value T (i.e. determining the widths of the “After” and the “Before” window). For this calculation, the principal characteristic is shown in **Table 7-33** (2×2^k - period = comparator window - period).

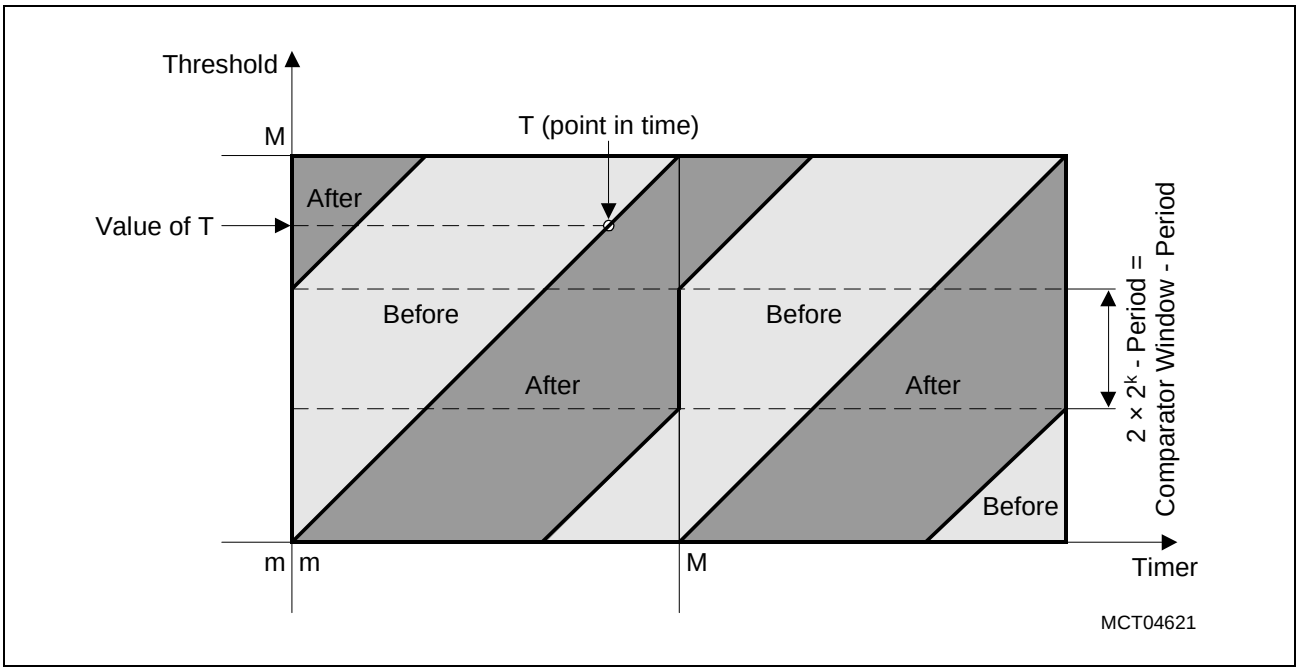


Figure 7-33 Graphical Representation of Signed Compare (2^k < Period ≤ 2×2^k)

Figure 7-34 shows how the observation window is positioned with respect to T. It also shows the **core observation window** that is always centered on T and which has a constant width.

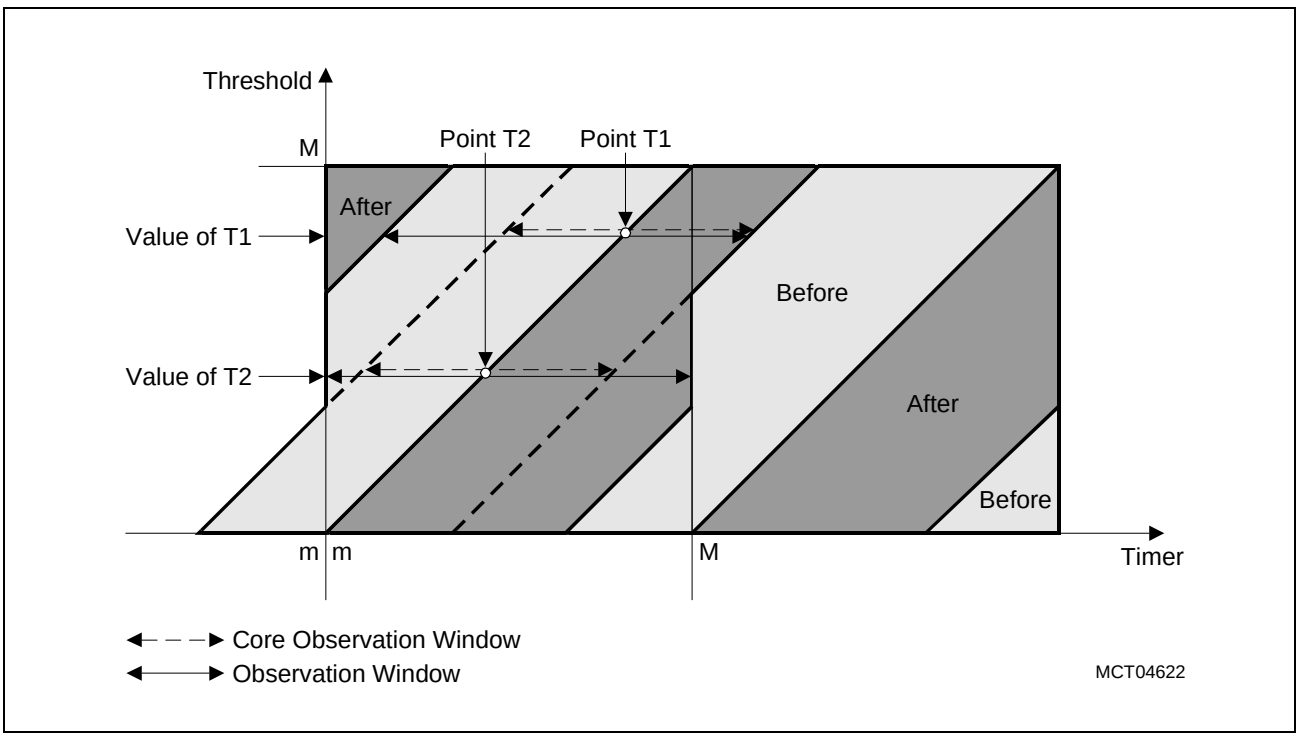


Figure 7-34 Core Observation Window in the Graphic

7.1.4.2 Global Timer Cell (GTC)

Features

- **24-bit based cells** related to two Global Timers GT0 and GT1.
- **Capture Mode** on rising, falling or both edges. The GTC can trigger an interrupt and perform an output manipulation (set, reset, toggle an output pin).
- **Compare Mode** on equal compare, or greater or equal compare. The GTC can trigger an interrupt and perform an output manipulation (set, reset, toggle an output pin). One additional feature is when the compare matches, to **capture after compare** the value of the selected global timer or the opposite global timer.
- **One Shot Mode** allows to stop the selected (capture or compare) mode after the first event.
- **Flexible mechanism** to link pin actions and allow complex combination of cells. (A cell has the ability to propagate actions over adjacent cells with higher number, in order to perform complex waveforms such as PWMs).

Architecture

The GPTA provides 32 global timer capture/compare cells (GTC00 to GTC31) with the following inputs:

- Two local timer value buses carrying GT0 and GT1 timer values,
- Two TEV flags reporting GT0 and GT1 timer value updates,
- Two TGE flags presenting the result of GT0 and GT1 compare operations,
- Four trigger input lines connected to external pins or FPCk output lines (**Section 7.1.5**),
- Two action mode lines (M0I, M1I) coming from the adjacent GTC with lower order number.

The GTC is locally equipped with a multiplexer selecting an external trigger line, three multiplexers accessing GT0 or GT1 output as data source, a 24-bit capture/compare register and a 24-bit equal comparator (**Figure 7-35**).

One signal and three flag lines are implemented as output of the GTC Module:

- One data line linked to an external pin (**Section 7.1.5**),
- Two action mode lines (M0O, M1O) going to the adjacent GTC with higher order number,
- One interrupt line (SQS) triggered by a capture/compare event.

GTCCTR control register bit field MOD initiates the GTC to operate in Capture Mode or Compare Mode hooked to GT0 or GT1.

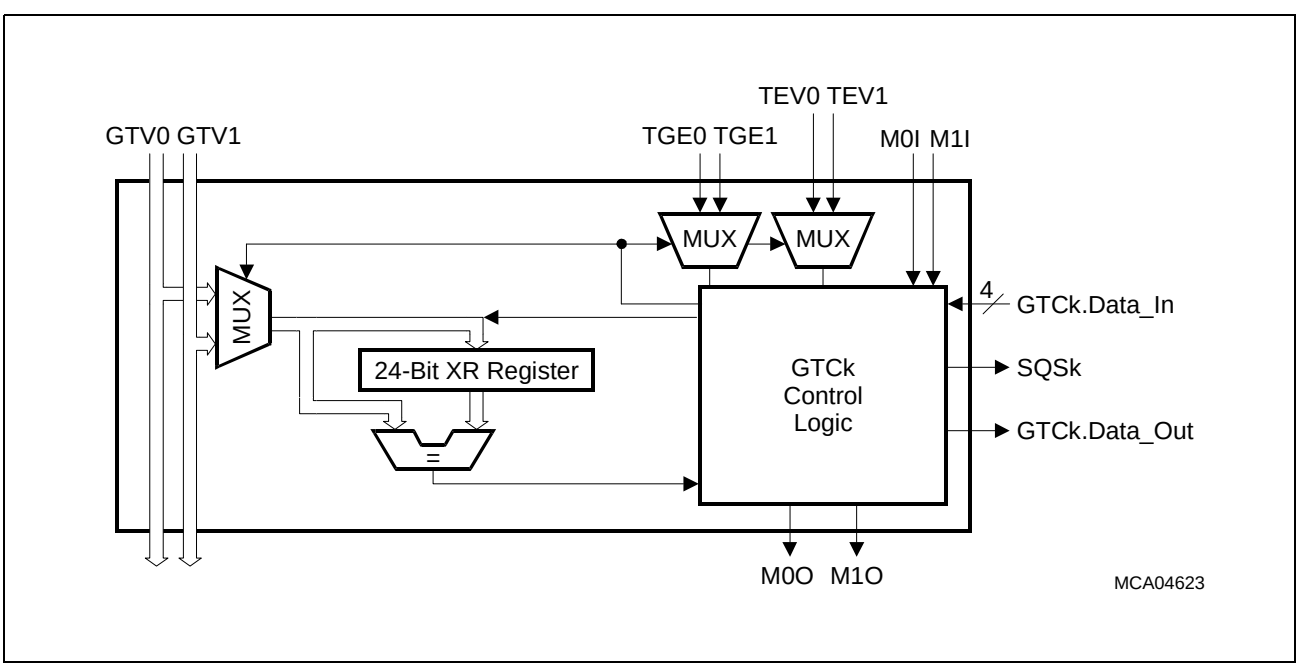


Figure 7-35 Architecture of Global Timer Cell

Capture Mode

The capture function can be performed on a rising edge (RED = 1), a falling edge (FED = 1) or both edges of the selected trigger line driven by an external pin or FPCk output (bit field PMU). On the requested event, the GTC:

- copies the contents of the selected timer to the capture/compare register,
- activates the interrupt line if control register bit REN is set to 1,
- performs an output signal line manipulation like “Set”, “Reset”, “Toggle” or “No Alteration” depending on the control register bit field OCM,
- transfers an action request, generated by an internal event or received on the M1I, M0I input lines, to the M1O, M0O output lines.

Note: When a GTC is in Capture Mode with no edge triggering the capture, the cell has no effect even if it is enabled.

Compare Mode

Several functions may be performed when the value of the selected timer matches and/or exceeds the capture/compare register contents (GTCXR). GES = 0 selects an “Equal Compare”, GES = 1 provides a “Greater Equal Compare”:

- the interrupt request line is activated if control register bit REN is set to 1 and
- an output signal line manipulation, such as “Set”, “Reset”, “Toggle” or “No Alteration” is performed depending on the control register bit field OCM,
- an action request, generated by an internal event or received on the M1I, M0I input lines, is transferred to the M1O, M0O output lines.

If a greater or equal compare is selected, this condition is evaluated when a write to the compare value is performed. The user should then assure that the GTC is already enabled so that the evaluation can take place. Otherwise, the GTC should be enabled first, and the compare value written after.

Compare Mode: Capture after Compare Select

When control register bit CAC is set and a compare event has occurred, the capture/compare register is loaded with one of the following:

- contents of the associated global timer selected by control register bit field MOD (CAT = 0),
- contents of the alternate global timer (CAT = 1). If a greater or equal compare has been evaluated, the GTC should be in One Shot Mode in order to prevent double capturing.

One Shot Mode

When control register bit OSM is set to 1, a self-disable is executed after each GTC event. The disable state is cleared by the next write access to control register $GTCCTR_k$. The current state of a GTC may be evaluated by reading the control register flag bit CEN.

Note: The contents of GTC capture/compare register (GTCXR) is write protected for capture_after_compare in Single-Shot Mode. Write protection is activated when the compare value is reached and released after a software access to register GTCXR.

Data Output Line Control

The data output line can be controlled by the local GTC and adjacent GTCs with a lower order number. For this purpose a communication link is implemented connecting all GTCs via their M0I, M1I inputs and M0O, M1O outputs respectively (**Figure 7-36**).

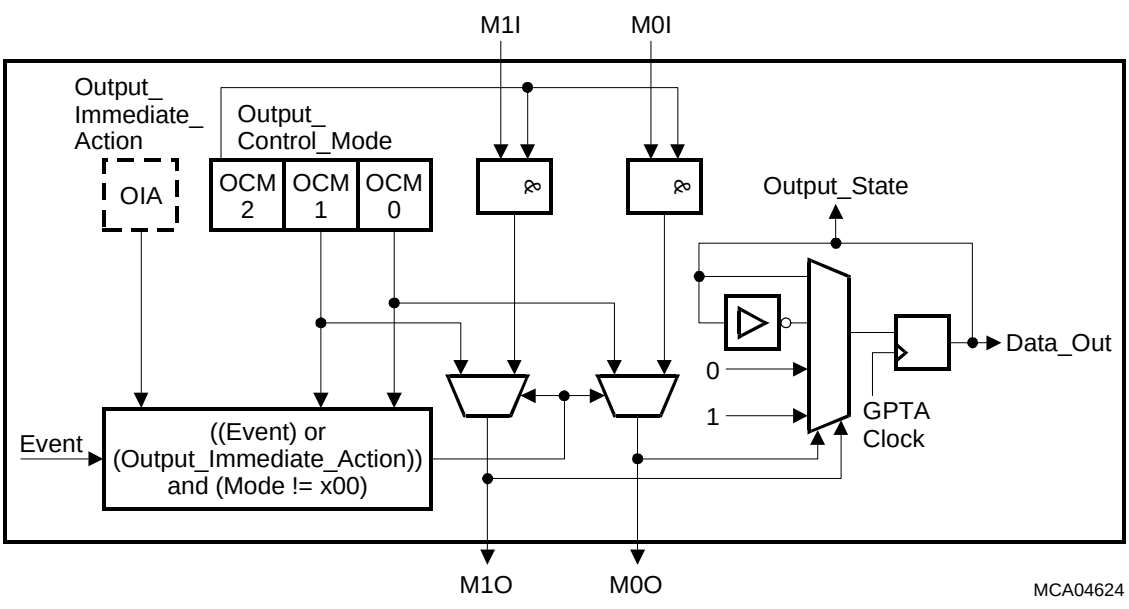


Figure 7-36 GTC Output Operation and Action Transfer

When control register bit OCM2 is reset, the data output line is controlled only by the local GTC. A set, reset, toggle, or hold operation may be performed depending on control register bits OCM1 and OCM0 (**Table 7-2**).

When control register bit OCM2 is set, the data output line is affected either by the local OCM1 and OCM0 bits or by the M0I, M1I input lines coming from the adjacent GTC. An enabled event in the local GTC superimposes an action request generated simultaneously by the M0I, M1I inputs.

Table 7-2 Selection of GTC Output Operations and Action Transfer Modes

Bit Field OCM2/OCM1/OCM0	Local GTC Capture or Compare Event	M1O, M0O	State of local Data Output Line
0 0 0	not occurred occurred	unchanged 0 0	not modified not modified
0 0 1	not occurred occurred	unchanged 0 1	not modified inverted
0 1 0	not occurred occurred	unchanged 1 0	not modified 0
0 1 1	not occurred occurred	unchanged 1 1	not modified 1
1 0 0	not occurred occurred	M1I M0I 0 0	modified according M1I, M0I not modified
1 0 1	not occurred occurred	M1I M0I 0 1	modified according M1I, M0I inverted
1 1 0	not occurred occurred	M1I M0I 1 0	modified according M1I, M0I 0
1 1 1	not occurred occurred	M1I M0I 1 1	modified according M1I, M0I 1

The GTC data output line, controlled by M0O, M1O signals, is linked to an external port line (**Figure 7-36**). The data output line may be updated immediately (control register bit OIA = 1) or upon a capture/compare event within the local or adjacent GTC. The current state of the data output line can be evaluated by reading control register bit OUT.

Logical Operating Units

The inter-cell communication architecture allows implementation of a complex waveform generation to be distributed over several GTCs controlling a common port pin.

For example, one GTC may be configured in Capture Mode triggered by a rising edge detected on the associated input pin line. The related interrupt service routine may increment the captured timer value by a delay offset and store the result in the GTCXR register of the adjacent GTC configured in Compare Mode. Upon a compare event in the second GTC, the output port line of a third GTC may be set via M0O, M1O interface lines. When the GTCXR register of the third cell is loaded with another compare value by the interrupt service routine related to the second GTC, the output port line may be reset by the next compare event within GTC3.

This logical operating unit provides an output signal with programmable pulse width and configurable delay with minimal software overhead.

7.1.4.3 Local Timer Cell (LTC)

Features

- **16-bit based cells** providing capture, compare, and timer functions.
- **Capture Mode** on rising, falling or both edges of the defined pin, or on a clock signal coming from the clock bus. The LTC can trigger an interrupt and perform an output manipulation (set, reset, toggle an output pin).
- **Compare Mode** on equal compare of the last timer. The LTC can trigger an interrupt and perform an output manipulation (set, reset, toggle an output pin).
- **Timer Mode** incremented either on clock signal coming from the clock bus, or on edges of the defined pin. An event is generated at overflow. The LTC can trigger an interrupt and perform an output manipulation (set, reset, toggle an output pin).
- **Reset Timer Mode** allowing the selected cell to be reset by an adjacent cell. Coherent update capability of adjacent LTCs for PWM management.
- **One Shot Mode** allows the selected (capture, compare, timer or reset timer) mode to stop after the first event.
- **Flexible mechanism** to link pin actions and allow complex combination of cells. (A cell has the ability to propagate actions over adjacent cells with higher number, in order to perform complex waveforms such as PWMs).

Architecture

The GPTA provides 64 local timer capture/compare cells (LTC00 to LTC63) with the following inputs:

- A local data bus (YI) carrying the timer value of the adjacent LTC with lower order number,
- A TI flag reporting a timer value update of the adjacent LTC with lower order number,
- A SI flag used as enable line when LTC operates in Compare Mode,
- Two mode lines (M0I, M1I) coming from the adjacent LTC cell with lower order number,
- An EI flag reporting an event generated by the adjacent LTC with higher order number,
- Four trigger lines hooked to external port pins or PDLn (n = 0, 1) and GTCk (k = 00-31) output lines (**Section 7.1.5**),
- Four clock lines derived from the GPTA internal clock bus (**Section 7.1.5**).

The LTC is locally equipped with a multiplexer selecting an external trigger line, a multiplexer accessing a clock source, a 16-bit capture/compare register and a 16-bit equal comparator (**Figure 7-37**).

One signal and five flag lines are implemented as output of the LTC Module:

- One data line linked to an external pin (**Section 7.1.5**),
- An interrupt line (SQT) triggered by a capture/compare event,
- A local data bus (YO) carrying the local timer value to the adjacent LTC with higher order number,
- A TO flag reporting a local timer value update of the adjacent LTC with higher order number,
- A SO flag enabling the compare function in the adjacent LTC with higher order number,
- An EO flag reporting a local event to the adjacent LTC with lower order number,
- Two mode lines (M0O, M1O) going to the adjacent LTC with higher order number.

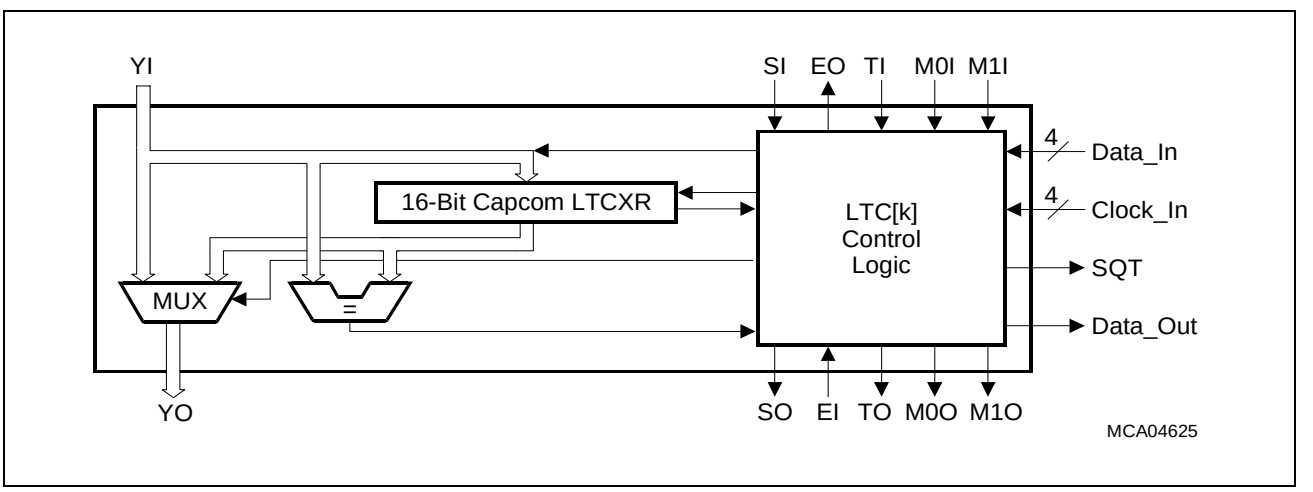


Figure 7-37 Architecture of Local Timer Cell

LTCCTR control register bit field MOD initiates the LTC to operate in “Free Running Timer”, “Reset Timer”, “Capture,” or “Compare” mode.

Free Running Timer Mode

The contents of the local timer cell register LTCXR are initialized by a software write access. The timer register may be incremented by a clock signal selected from the clock bus, or by a signal edge derived from an associated input pin. The trigger source is defined by the most significant bit of control register’s bit field PMU (PMU2 = 0 enables the clock bus input, PMU2 = 1 selects a pin input), while the required input channel is defined by bits PMU1/PMU0. The timer can be incremented on a rising (RED = 1), falling (FED = 1) or both edges of the selected trigger line. Every alteration of the timer register (increment, reset or write access) is indicated by the output signal state TO = 1. When the timer reaches its overflow value ($FFFF_H$),

- An interrupt may be generated (control register bit REN = 1).
- The data output line may be altered by a set, reset or toggle operation, depending on control register bit field OCM.
- An action request is transferred to subsequent LTCs with higher order numbers via M1O, M0O output lines.
- The event output line EO is set to ‘High’ for one clock cycle.

The event output line EO is also set to ‘High’ by a software reset writing $FFFF_H$ to register LTCXR.

Reset Timer Mode

An LTC, configured in “Reset Timer Mode”, provides the same functionality like the free running timer extended by two additional features:

- It can be reset by the EI flag, which may have been set by an event that occurred in the adjacent LTC with higher order number.
- If control register bit CUD has been set to 1, the EI reset event also toggles the logic state of the SO output line before it clears register bit CUD automatically.

Capture Mode

The capture function can be performed on a rising edge (RED = 1), falling edge (FED = 1) or both edges of the selected trigger source. The trigger source is defined by the most significant bit of the control register's bit field PMU (PMU2 = 0 selects the input pins, PMU2 = 1 selects the clocks), while the required input channel is defined by bits PMU1/PMU0. On the requested event, the LTC:

- copies the state of the timer data input bus (YI) to the capture/compare register LTCXR,
- activates the interrupt request line, if control register bit REN is set to 1,
- performs an output signal line manipulation like "Set", "Reset", "Toggle" or "No Alteration", depending on control register bit field OCM and the M1I, M0I input line state,
- generates and/or passes an action request to subsequent LTCs with higher order numbers via M1O, M0O output lines
- and the event output line EO is set to 'High' for one clock cycle.

Compare Mode

The compare function can be enabled on a "Low", "High" or both levels of the select line input SI (SOL = 1 SOH = 1). The current state of the select line input may be obtained by reading the control register bit field SLL. When the value of the timer data input bus (YI) matches the capture/compare register contents (LTCXR)

- an output signal line manipulation is performed ("Set", "Reset", "Toggle", or "No Alteration") depending on control register bit field OCM,
- the interrupt request line is activated if control register bit REN is set to 1,
- an action request is generated and/or passed to subsequent LTCs with higher order numbers via M1O, M0O output lines
- and the event output line EO is set to 'High' for one clock cycle.

Note: To enable the compare function in all cases (on every timer or compare register update caused by a software write access, a reset event or a compare match), the bits SOL and SOH must be set to 1.

When the compare function is disabled (SOL = 0 and SOH = 0), the state of the event input line EI is directly copied to the event output line EO.

An inactive cell (SOL = SOH = 0, or SI doesn't match the programmed value) will mirror the state of the event input line EI to the event output line EO.

One Shot Operation

When control register bit OSM is set to 1, a self-disable is executed after each LTC event. The disable state is cleared by the next write access to control register LTCCTRk. The current state of a LTC may be evaluated by scanning the control register flag bit CEN.

Note: The contents of GTC capture/compare register (GTCXR) are write protected for capture_after_compare in single-shot mode. Write protection is activated when the compare value is reached and is released after a software access to register GTCXR.

Data Output Line Control

The data output line can be controlled by the local LTC and adjacent LTCs with lower order number. For this purpose, a communication link is implemented connecting all LTCs via their M0I, M1I inputs and M0O, M1O outputs respectively (**Figure 7-38**).

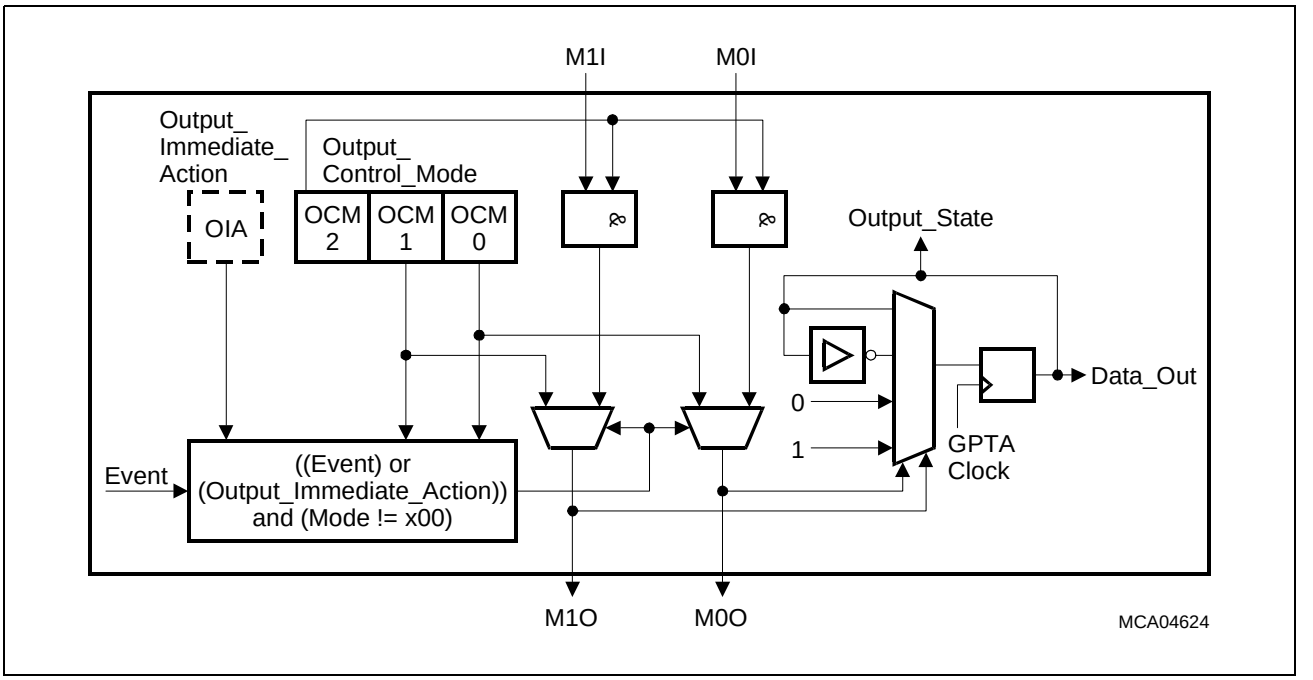


Figure 7-38 LTC Output Operation and Action Transfer

When control register bit OCM2 is reset, the data output line is controlled only by the local LTC. A set, reset, toggle, or hold operation may be performed depending on control register bits OCM1 and OCM0 (**Table 7-3**).

When control register bit OCM2 is set, the data output line is affected by either the local OCM1 and OCM0 bits or by the M0I, M1I input lines coming from the adjacent LTC. An enabled event in the local LTC superimposes an action request generated simultaneously by the M0I, M1I inputs.

Table 7-3 Selection of LTC Output Operations and Action Transfer Modes

Bit Field OCM OCM2/OCM1/OCM0	Local LTC Capture or Compare Event	M1O, M0O	State of local Data Output Line
0 0 0	not occurred occurred	unchanged 0 0	not modified not modified
0 0 1	not occurred occurred	unchanged 0 1	not modified inverted
0 1 0	not occurred occurred	unchanged 1 0	not modified 0
0 1 1	not occurred occurred	unchanged 1 1	not modified 1
1 0 0	not occurred occurred	M1I M0I 0 0	modified according M1I, M0I not modified
1 0 1	not occurred occurred	M1I M0I 0 1	modified according M1I, M0I inverted
1 1 0	not occurred occurred	M1I M0I 1 0	modified according M1I, M0I 0
1 1 1	not occurred occurred	M1I M0I 1 1	modified according M1I, M0I 1

The LTC data output line, controlled by M0O, M1O signals, is linked to an external port line (**Figure 7-36**). The data output line may be updated immediately (control register bit OIA = 1) or upon a timer, capture or compare event within the local or adjacent LTC. The current state of the data output line can be evaluated by reading control register bit OUT.

Logical Operating Units

The inter-cell communication architecture allows concatenation of several LTCs to a logical unit. A logical unit contains any number of LTCs communicating via M1 and M0 lines and ends at a LTC disabled for action input or transfer (such as LTC configured as reset timer or LTC initiated with OCM2 = 0).

Therefore, the LTC with the lowest order number should be configured as reset-timer providing all other LTCs of the logical unit with a time base (YO) and a compare enable signal (SO). Another LTC of the same logical unit may be initiated in Compare Mode to reset the LTC via event output line EO, when a programmed threshold value is reached (register LTCXR) and the current state of the select line input SI matches the condition selected by control register bits SOH/SOL. Additional LTCs, belonging to the same logical unit, may operate in Capture Mode triggered by a rising edge, falling edge or both edges of an input port line or a clock line selected from the clock bus. On the generated event, these LTCs capture the current contents of the timer cell, may generate an

interrupt request, can perform a manipulation of an output port line (set, reset or toggle), and may also reset the LTC via the event output line EO.

LTC Application Example

A logical unit, containing five LTCs, may be used to generate a PWM signal with a programmable duty cycle and period length, and fully coherent update of the period and duty cycle (**Figure 7-39**).

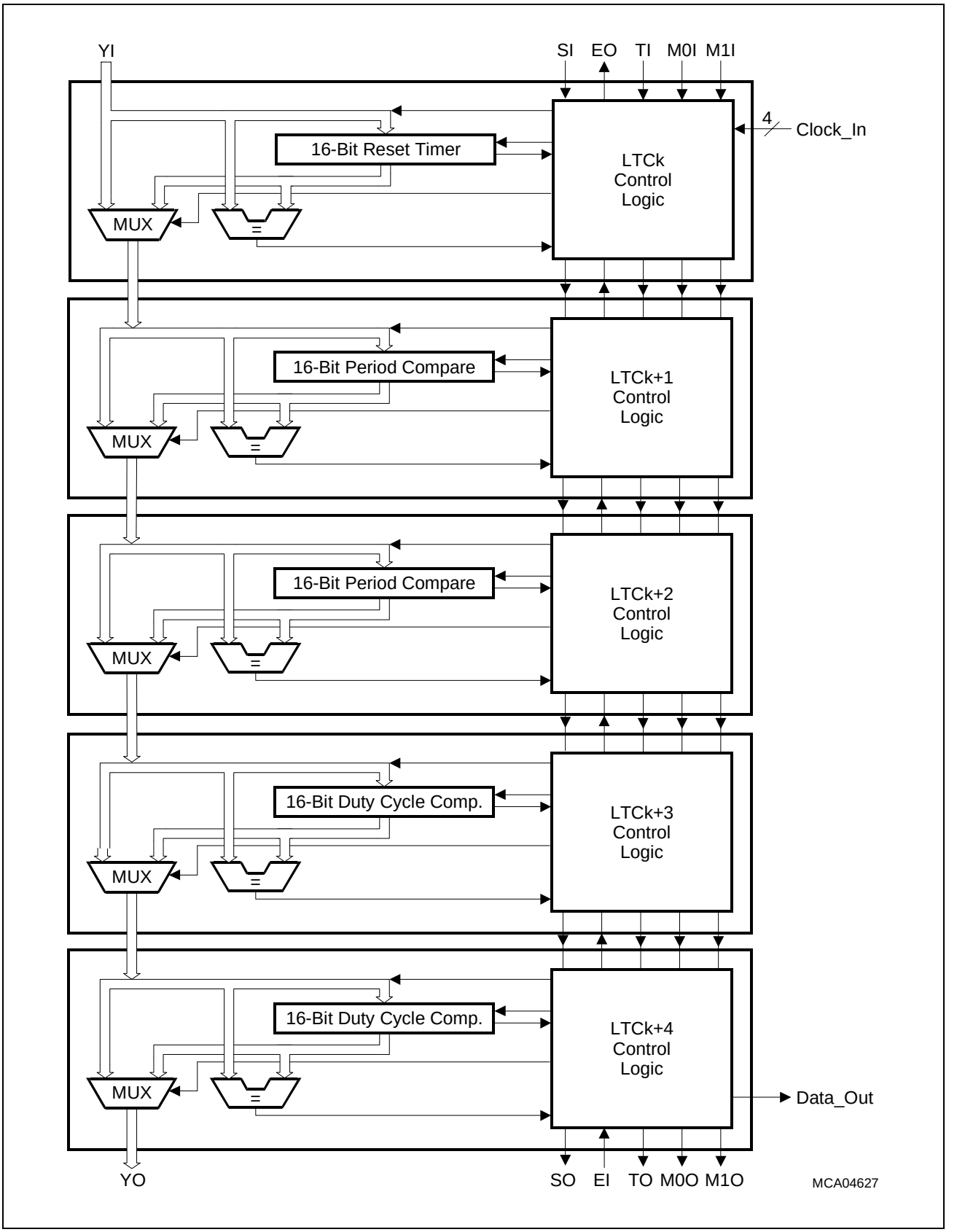


Figure 7-39 PWM Signal Generation with a Logical LTC Unit

LTCk is configured as “Reset Timer” providing all subsequent cells with a time base. The OCM bit field in the LTCk control register has been programmed to ‘011’, setting the data output line to 1 in case of an incoming reset request (event input line EI).
LTCk + 3, initialized in “Compare Mode” and loaded with a “Duty Cycle Threshold”, is enabled when the select line input SI is cleared. The OCM bit field in the LTCk + 3 control register has been programmed to ‘110’, which resets the data output line to as soon as the LTCk timer register contents has been incremented to the duty cycle threshold value. The interrupt service routine, initiated by the LTCk + 3 compare event, may activate the “Coherent Update Enable Flag” (CUD) to toggle the current state of the select line output SO at the next LTCk timer overflow.

The LTCk + 1 also operates in “Compare Mode” and is enabled, if the select line input SI is cleared. The OCM bit field in the LTCk + 1 control register has been set to ‘100’ passing the action request, generated by the reset timer, to the subsequent LTCs. When the timer LTCk has been incremented to the value stored in register LTCk + 1 (Period Threshold), the timer cell LTCk is reset via the event output line EO and the select line input SI toggles if CUD has been set to 1.

A toggle operation disables the LTCk + 1 and LTCk + 3 and activates LTCk + 2 and LTCk + 4. In this case, the same procedure is started with a different parameter block.
LTCk + 4 operates also in “Compare Mode”, but has been loaded with a different “Duty Cycle Threshold”. Its OCM bit field has been also programmed to ‘110’. LTCk + 4 is enabled when the select line input SI is set to 1. LTCk + 4 resets the data output line to 0 as soon as the LTCk timer register contents has been incremented to the duty cycle threshold value.
LTCk + 2, initialized in “Compare Mode” and loaded with a different “Period Threshold”, is enabled if the select line input SI is set to 1. The OCM bit field in the LTCk + 2 control register has been set to ‘100’ passing the action request, generated by the reset timer or LTCk + 1, to the subsequent LTCs. When the timer LTCk has been incremented to the value stored in register LTCk + 2 (Period Threshold), the timer cell LTCk is reset via the event output line EO and the select line output SI toggles if CUD has been set to 1.

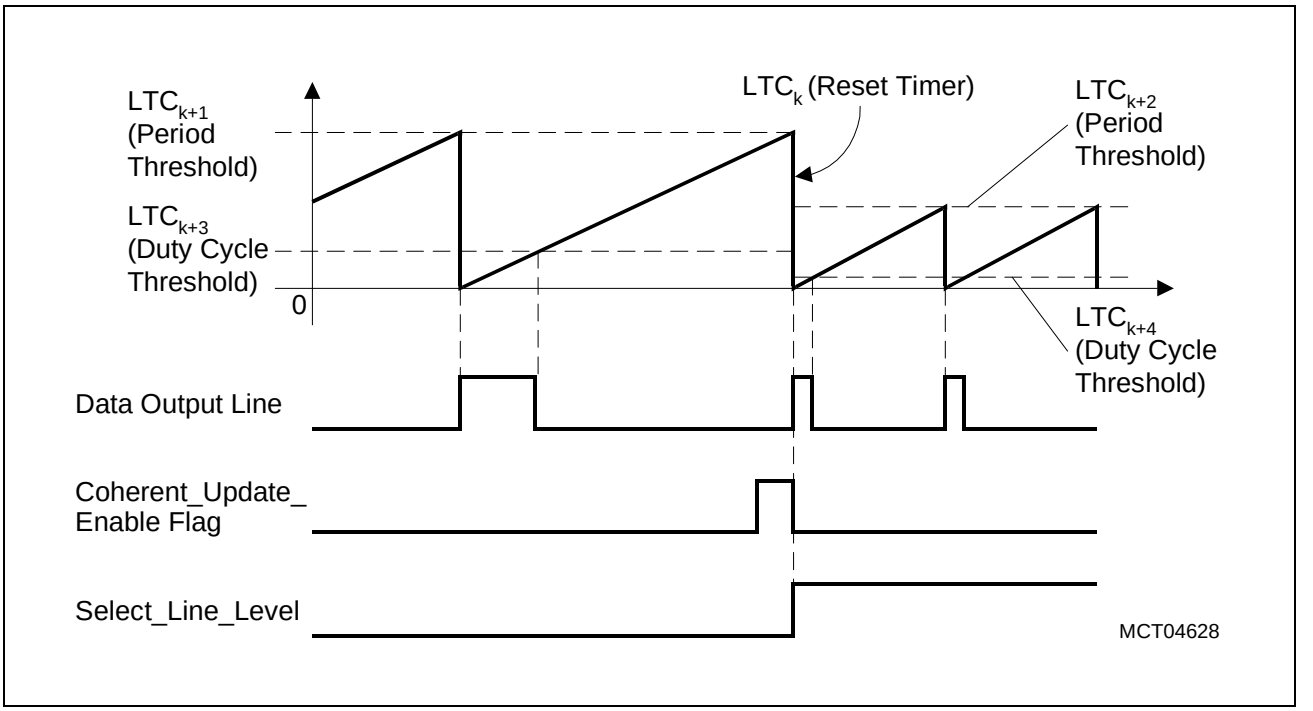


Figure 7-40 Internal Signal States of the Logical LTC Unit generating a PWM Signal

To get a 0% output signal, the duty compare of the active cells must be set to -1 ($FFFF_H$ = -1). The timer attempts to set the data output line, but the dominating duty compare cell resets the data output line. The result is that the data output line remains low.

A duty cycle of 100% can be obtained by setting the duty cycle threshold above the period threshold value. In this case, no reset event for the data line can be generated.

7.1.5 GTC and LTC Input/Output Line Sharing Unit (IOLS)

32 GTCs, each equipped with four input channels and one output line, must be connected to external port lines. 64 LTC units, each provided with three input channels reserved for external signals and one output line, must be linked to external port lines. Also, six FPCs with four input channels each must be connected to external port lines.

Providing an individual port line for every input and output channel of a GTC, LTC and FPC would consume 312 port pins. Therefore, an input and output line sharing unit is implemented to reduce the number of required port pins and to support parallel processing of input signals by GTCs and LTCs.

There are connections to the ADC Module.

Input Line Sharing

Figure 7-41 presents the network implemented for GTCs and LTCs sharing common input port pins. It also demonstrates how the output signal line of each GTC is directly linked to the input multiplexer of two LTCs.

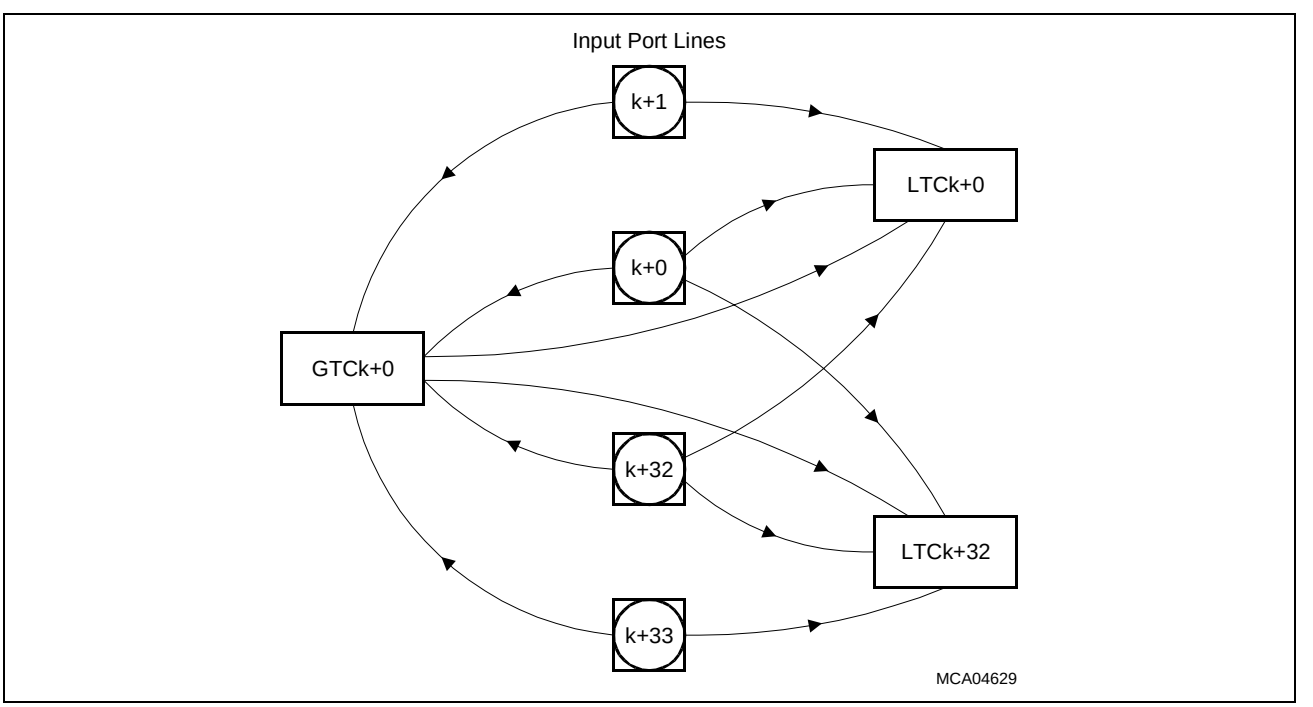


Figure 7-41 GTC and LTC Input Network (k = 00 - 31)

An exception has been made to process the output information of both PDL units; LTC00 up to LTC03 may be driven by PDL0 and PDL1 forward and backward rotation signals instead of GTC00 … GTC03 outputs.

A second exception has been made on the input multiplexer channel 0 of cells GTC00, GTC02; GTC04; GTC06; GTC08; GTC10 which are driven by signal output lines of FPC0, FPC1, FPC2, FPC3, FPC4 and FPC5 units.

The exact assignment of input port lines (referenced by their line numbers) to GTC and LTC input multiplexer channels is presented in **Table 7-4** sorted by cell numbers.

Output Line Sharing

Figure 7-42 presents the network implemented for GTCs and LTCs sharing common output port pins.

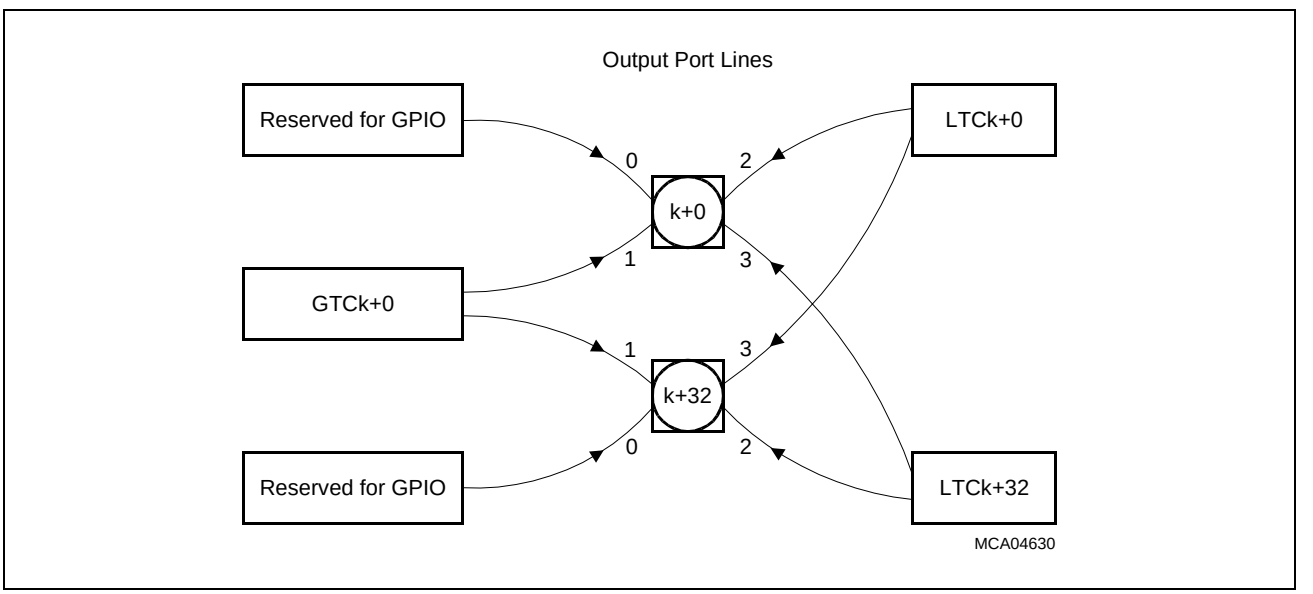


Figure 7-42 GTC and LTC Output Port Line Sharing Network (k = 00 - 31)

Each output port line is linked to the associated GTC and LTC output lines via an integrated multiplexer (**Figure 7-43**) individually controlled by one of four multiplexer control register. The value ‘00’ of the respective control register bit field is reserved to drive a specific port control line (AltEnablek). Depending on the microcontroller specific port implementation, the AltEnablek line may be used to disconnect the associated GTC and LTC output lines from the port pin and to link the freed pin to a general purpose I/O register or another logic module output.

All GTC and LTC output lines may be connected to two different port lines. The exact assignment of GTC and LTC output signal lines to output port lines is presented in **Table 7-4** sorted by cell numbers.

Emergency Function

An emergency function has been implemented to allow a fast response to an external event without any software overhead or clock edge delay. A low state, detected on the emergency input line connected to a specific external port pin, pulls down all AltEnablek lines which have been enabled for this kind of exception handling by the associated bits in the emergency control register EMGCTR. If supported by the microcontroller specific port implementation, the port pins may be provided with the contents of general purpose I/O register loaded with exception handling values.

In case of an emergency situation, such as power supply failure or emergency switch activation, this mechanism may be used to shut down all external devices controlled by the GPTA Module.

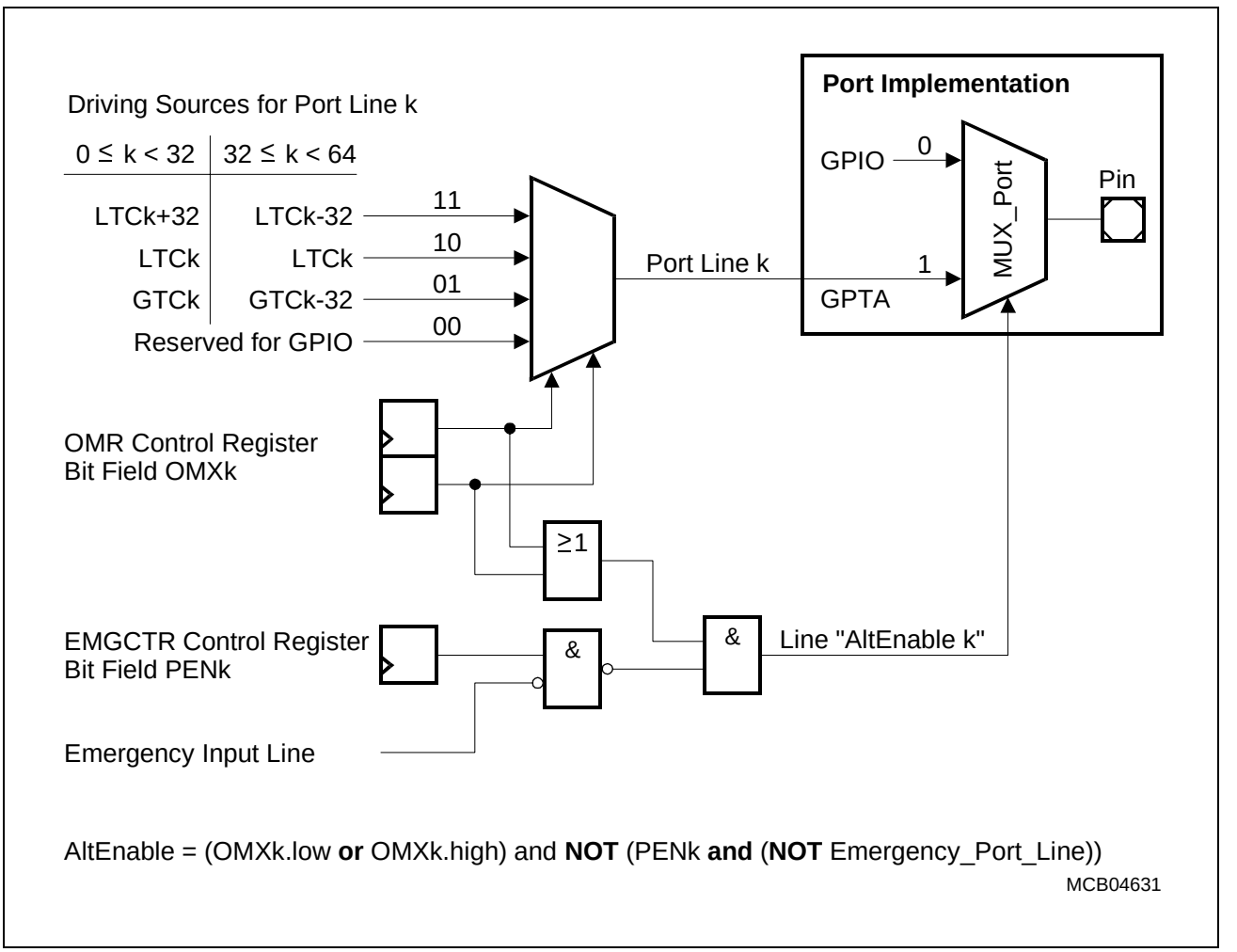


Figure 7-43 Output Sharing Unit with Emergency Function (k = 00 - 63)

In summary, **input selection** is done in the control register of the corresponding cell (LTC or GTC). The PMU bit field is used for that purpose (bit 7-6 in GTC control register and bit 8-6 in LTC control register).

Output selection of the corresponding cells is performed in the Output Port Line Multiplex register (OMR0 to OMR3 registers). The OMX bit field is used for that purpose.

Additionally, the selection between GPTA and GPIO function is performed in a separate control register which is not part of the GPTA Kernel. See **Chapter 7.3**.

An example of a GTC with its Input/Output connections is given in **Figure 7-44**.

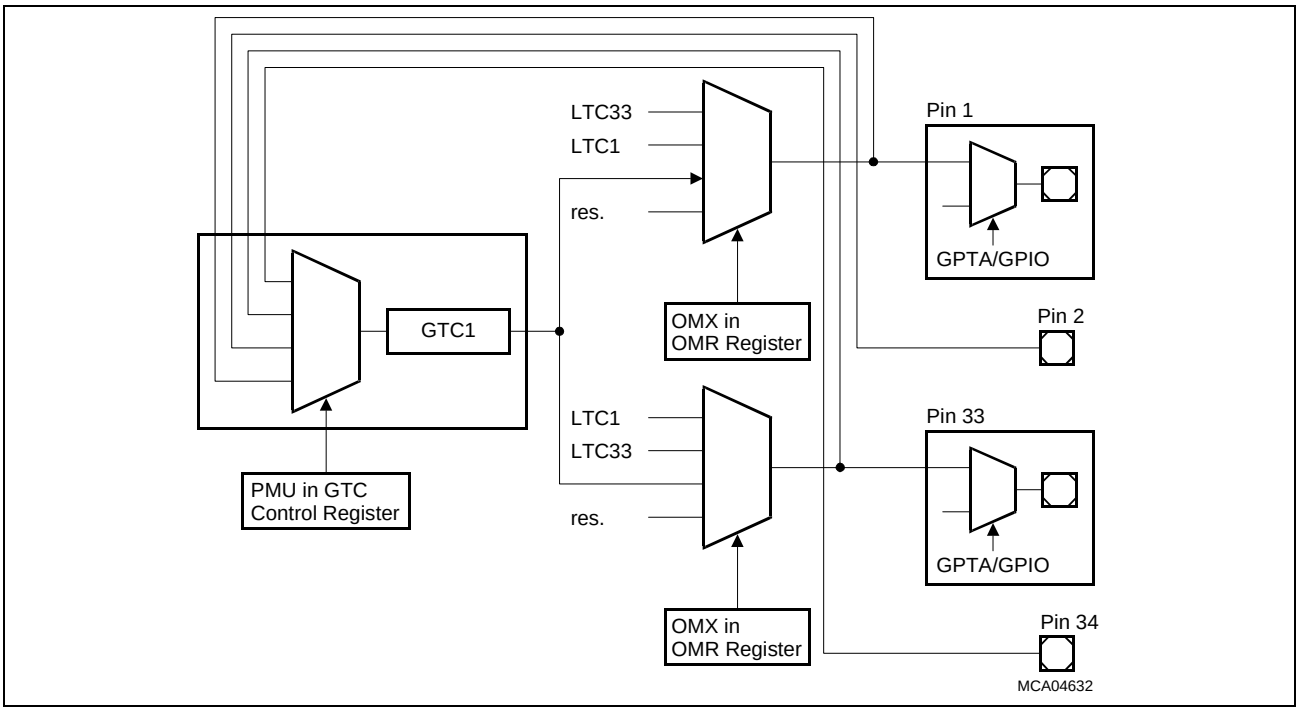


Figure 7-44 Example of a GTC Input/Output Connection (GTC1)

Table 7-4 Input/Output Assignment to GTC, LTC and FPC Cells

Cells	Inputs								Outputs	
	(clock select for LTC only); LTCk.Input_pin_mux = 1xx				FPCk.Input_source GTCk.Input_pin_mux LTCk.Input_pin_mux = X_H				Port lines k linked to the cell	
PMUk:	4	5	6	7	0	1	2	3	–	–
GTC00	–	–	–	–	FPC0	20_H	01_H	21_H	00_H	20_H
GTC01	–	–	–	–	01_H	21_H	02_H	22_H	01_H	21_H
GTC02	–	–	–	–	FPC1	22_H	03_H	23_H	02_H	22_H
GTC03	–	–	–	–	03_H	23_H	04_H	24_H	03_H	23_H
GTC04	–	–	–	–	FPC2	24_H	05_H	25_H	04_H	24_H
GTC05	–	–	–	–	05_H	25_H	06_H	26_H	05_H	25_H
GTC06	–	–	–	–	FPC3	26_H	07_H	27_H	06_H	26_H
GTC07	–	–	–	–	07_H	27_H	08_H	28_H	07_H	27_H
GTC08	–	–	–	–	FPC4	28_H	09_H	29_H	08_H	28_H
GTC09	–	–	–	–	09_H	29_H	$0A_H$	$2A_H$	09_H	29_H
GTC10	–	–	–	–	FPC5	$2A_H$	$0B_H$	$2B_H$	$0A_H$	$2A_H$
GTC11	–	–	–	–	$0B_H$	$2B_H$	$0C_H$	$2C_H$	$0B_H$	$2B_H$

Table 7-4 Input/Output Assignment to GTC, LTC and FPC Cells (cont'd)

Cells	Inputs								Outputs	
GTC12	–	–	–	–	$0C_H$	$2C_H$	$0D_H$	$2D_H$	$0C_H$	$2C_H$
GTC13	–	–	–	–	$0D_H$	$2D_H$	$0E_H$	$2E_H$	$0D_H$	$2D_H$
GTC14	–	–	–	–	$0E_H$	$2E_H$	$0F_H$	$2F_H$	$0E_H$	$2E_H$
GTC15	–	–	–	–	$0F_H$	$2F_H$	10_H	30_H	$0F_H$	$2F_H$
GTC16	–	–	–	–	10_H	30_H	11_H	31_H	10_H	30_H
GTC17	–	–	–	–	11_H	31_H	12_H	32_H	11_H	31_H
GTC18	–	–	–	–	12_H	32_H	13_H	33_H	12_H	32_H
GTC19	–	–	–	–	13_H	33_H	14_H	34_H	13_H	33_H
GTC20	–	–	–	–	14_H	34_H	15_H	35_H	14_H	34_H
GTC21	–	–	–	–	15_H	35_H	16_H	36_H	15_H	35_H
GTC22	–	–	–	–	16_H	36_H	17_H	37_H	16_H	36_H
GTC23	–	–	–	–	17_H	37_H	18_H	38_H	17_H	37_H
GTC24	–	–	–	–	18_H	38_H	19_H	39_H	18_H	38_H
GTC25	–	–	–	–	19_H	39_H	$1A_H$	$3A_H$	19_H	39_H
GTC26	–	–	–	–	$1A_H$	$3A_H$	$1B_H$	$3B_H$	$1A_H$	$3A_H$
GTC27	–	–	–	–	$1B_H$	$3B_H$	$1C_H$	$3C_H$	$1B_H$	$3B_H$
GTC28	–	–	–	–	$1C_H$	$3C_H$	$1D_H$	$3D_H$	$1C_H$	$3C_H$
GTC29	–	–	–	–	$1D_H$	$3D_H$	$1E_H$	$3E_H$	$1D_H$	$3D_H$
GTC30	–	–	–	–	$1E_H$	$3E_H$	$1F_H$	$3F_H$	$1E_H$	$3E_H$
GTC31	–	–	–	–	$1F_H$	$3F_H$	20_H	00_H	$1F_H$	$3F_H$
LTC00	7	6	1	2	00_H	20_H	01_H	PDL0.F	00_H	20_H
LTC01	7	6	1	3	01_H	21_H	02_H	PDL0.B	01_H	21_H
LTC02	7	6	1	4	02_H	22_H	03_H	PDL1.F	02_H	22_H
LTC03	7	6	1	5	03_H	23_H	04_H	PDL1.B	03_H	23_H
LTC04	7	6	2	3	04_H	24_H	05_H	GTC04	04_H	24_H
LTC05	7	6	2	4	05_H	25_H	06_H	GTC05	05_H	25_H
LTC06	7	6	2	5	06_H	26_H	07_H	GTC06	06_H	26_H
LTC07	7	6	3	4	07_H	27_H	08_H	GTC07	07_H	27_H
LTC08	7	6	3	5	08_H	28_H	09_H	GTC08	08_H	28_H
LTC09	7	6	4	5	09_H	29_H	$0A_H$	GTC09	09_H	29_H

Table 7-4 Input/Output Assignment to GTC, LTC and FPC Cells (cont'd)

Cells	Inputs								Outputs	
LTC10	7	6	1	2	$0A_H$	$2A_H$	$0B_H$	GTC10	$0A_H$	$2A_H$
LTC11	7	6	1	3	$0B_H$	$2B_H$	$0C_H$	GTC11	$0B_H$	$2B_H$
LTC12	7	6	1	4	$0C_H$	$2C_H$	$0D_H$	GTC12	$0C_H$	$2C_H$
LTC13	7	6	1	5	$0D_H$	$2D_H$	$0E_H$	GTC13	$0D_H$	$2D_H$
LTC14	7	6	2	3	$0E_H$	$2E_H$	$0F_H$	GTC14	$0E_H$	$2E_H$
LTC15	7	6	2	4	$0F_H$	$2F_H$	10_H	GTC15	$0F_H$	$2F_H$
LTC16	7	6	2	5	10_H	30_H	11_H	GTC16	10_H	30_H
LTC17	7	6	3	4	11_H	31_H	12_H	GTC17	11_H	31_H
LTC18	7	6	3	5	12_H	32_H	13_H	GTC18	12_H	32_H
LTC19	7	6	4	5	13_H	33_H	14_H	GTC19	13_H	33_H
LTC20	7	6	1	2	14_H	34_H	15_H	GTC20	14_H	34_H
LTC21	7	6	1	3	15_H	35_H	16_H	GTC21	15_H	35_H
LTC22	7	6	1	4	16_H	36_H	17_H	GTC22	16_H	36_H
LTC23	7	6	1	5	17_H	37_H	18_H	GTC23	17_H	37_H
LTC24	7	6	2	3	18_H	38_H	19_H	GTC24	18_H	38_H
LTC25	7	6	2	4	19_H	39_H	$1A_H$	GTC25	19_H	39_H
LTC26	7	6	2	5	$1A_H$	$3A_H$	$1B_H$	GTC26	$1A_H$	$3A_H$
LTC27	7	6	3	4	$1B_H$	$3B_H$	$1C_H$	GTC27	$1B_H$	$3B_H$
LTC28	7	6	3	5	$1C_H$	$3C_H$	$1D_H$	GTC28	$1C_H$	$3C_H$
LTC29	7	6	4	5	$1D_H$	$3D_H$	$1E_H$	GTC29	$1D_H$	$3D_H$
LTC30	7	6	1	2	$1E_H$	$3E_H$	$1F_H$	GTC30	$1E_H$	$3E_H$
LTC31	7	6	1	3	$1F_H$	$3F_H$	20_H	GTC31	$1F_H$	$3F_H$
LTC32	7	6	1	4	20_H	00_H	21_H	GTC00	20_H	00_H
LTC33	7	6	1	5	21_H	01_H	22_H	GTC01	21_H	01_H
LTC34	7	6	2	3	22_H	02_H	23_H	GTC02	22_H	02_H
LTC35	7	6	2	4	23_H	03_H	24_H	GTC03	23_H	03_H
LTC36	7	6	2	5	24_H	04_H	25_H	GTC04	24_H	04_H
LTC37	7	6	3	4	25_H	05_H	26_H	GTC05	25_H	05_H
LTC38	7	6	3	5	26_H	06_H	27_H	GTC06	26_H	06_H
LTC39	7	6	4	5	27_H	07_H	28_H	GTC07	27_H	07_H

Table 7-4 Input/Output Assignment to GTC, LTC and FPC Cells (cont'd)

Cells	Inputs								Outputs	
LTC40	7	6	1	2	28_H	08_H	29_H	GTC08	28_H	08_H
LTC41	7	6	1	3	29_H	09_H	$2A_H$	GTC09	29_H	09_H
LTC42	7	6	1	4	$2A_H$	$0A_H$	$2B_H$	GTC10	$2A_H$	$0A_H$
LTC43	7	6	1	5	$2B_H$	$0B_H$	$2C_H$	GTC11	$2B_H$	$0B_H$
LTC44	7	6	2	3	$2C_H$	$0C_H$	$2D_H$	GTC12	$2C_H$	$0C_H$
LTC45	7	6	2	4	$2D_H$	$0D_H$	$2E_H$	GTC13	$2D_H$	$0D_H$
LTC46	7	6	2	5	$2E_H$	$0E_H$	$2F_H$	GTC14	$2E_H$	$0E_H$
LTC47	7	6	3	4	$2F_H$	$0F_H$	30_H	GTC15	$2F_H$	$0F_H$
LTC48	7	6	3	5	30_H	10_H	31_H	GTC16	30_H	10_H
LTC49	7	6	4	5	31_H	11_H	32_H	GTC17	31_H	11_H
LTC50	7	6	1	2	32_H	12_H	33_H	GTC18	32_H	12_H
LTC51	7	6	1	3	33_H	13_H	34_H	GTC19	33_H	13_H
LTC52	7	6	1	4	34_H	14_H	35_H	GTC20	34_H	14_H
LTC53	7	6	1	5	35_H	15_H	36_H	GTC21	35_H	15_H
LTC54	7	6	2	3	36_H	16_H	37_H	GTC22	36_H	16_H
LTC55	7	6	2	4	37_H	17_H	38_H	GTC23	37_H	17_H
LTC56	7	6	2	5	38_H	18_H	39_H	GTC24	38_H	18_H
LTC57	7	6	3	4	39_H	19_H	$3A_H$	GTC25	39_H	19_H
LTC58	7	6	3	5	$3A_H$	$1A_H$	$3B_H$	GTC26	$3A_H$	$1A_H$
LTC59	7	6	4	5	$3B_H$	$1B_H$	$3C_H$	GTC27	$3B_H$	$1B_H$
LTC60	7	6	1	2	$3C_H$	$1C_H$	$3D_H$	GTC28	$3C_H$	$1C_H$
LTC61	7	6	1	3	$3D_H$	$1D_H$	$3E_H$	GTC29	$3D_H$	$1D_H$
LTC62	7	6	1	4	$3E_H$	$1E_H$	$3F_H$	GTC30	$3E_H$	$1E_H$
LTC63	7	6	1	5	$3F_H$	$1F_H$	00_H	GTC31	$3F_H$	$1F_H$
FPC0	–	–	–	–	00_H	10_H	20_H	30_H	–	–
FPC1	–	–	–	–	02_H	12_H	22_H	32_H	–	–
FPC2	–	–	–	–	04_H	14_H	24_H	34_H	–	–
FPC3	–	–	–	–	06_H	16_H	26_H	36_H	–	–
FPC4	–	–	–	–	08_H	18_H	28_H	38_H	–	–
FPC5	–	–	–	–	$0A_H$	$1A_H$	$2A_H$	$3A_H$	–	–

7.1.6 ADC Connections

The GPTA can trigger an Analog-to-Digital Conversion. The Service_request_trigger coming from the GTC and LTC are selected by mux. Selection is done via the ADCCTR Multiplex Control Register (see **Section 7.2.10**). Two triggers are implemented for each ADC, ADC0 and ADC1. The GPTA holds the trigger signal as long as the trigger is not acknowledged by the ADC Module. The ADC synchronizes the trigger signal from the GPTA and returns the synchronized trigger signal as an acknowledged signal back to the GPTA.

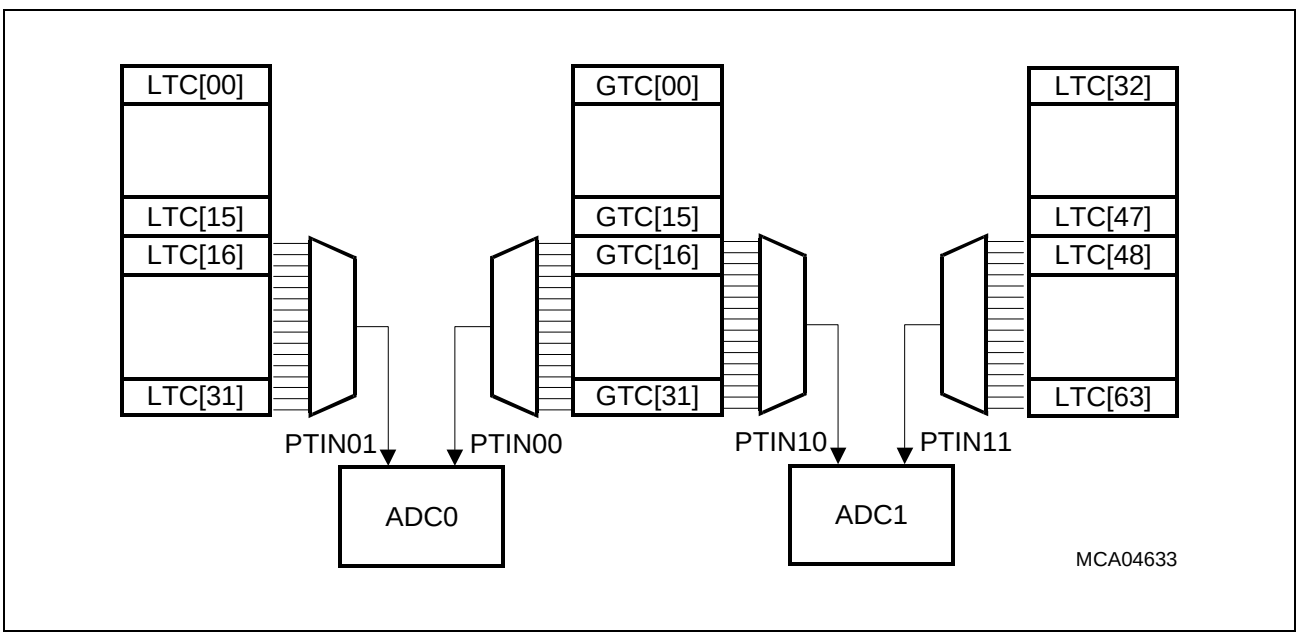


Figure 7-45 ADC Connections

Table 7-5 ADC Trigger Signal Selection Table

Bit Field MUXxx in Register ADCCTR	Signal Source			
	for PTIN00	for PTIN01	for PTIN10	for PTIN11
0	GTC16	LTC16	GTC16	LTC48
1	GTC17	LTC17	GTC17	LTC49
2	GTC18	LTC18	GTC18	LTC50
3	GTC19	LTC19	GTC19	LTC51
4	GTC20	LTC20	GTC20	LTC52
5	GTC21	LTC21	GTC21	LTC53
6	GTC22	LTC22	GTC22	LTC54
7	GTC23	LTC23	GTC23	LTC55
8	GTC24	LTC24	GTC24	LTC56

Table 7-5 ADC Trigger Signal Selection Table (cont'd)

Bit Field MUXxx in Register ADCCTR	Signal Source			
	for PTIN00	**for PTIN01**	**for PTIN10**	**for PTIN11**
9	GTC25	LTC25	GTC25	LTC57
10	GTC26	LTC26	GTC26	LTC58
11	GTC27	LTC27	GTC27	LTC59
12	GTC28	LTC28	GTC28	LTC60
13	GTC29	LTC29	GTC29	LTC61
14	GTC30	LTC30	GTC30	LTC62
15	GTC31	LTC31	GTC31	LTC63

7.1.7 Interrupt Sharing Unit (IS)

Service Requests

The GPTA Module contains 111 interrupt sources implemented in five cell types, as shown in **Table 7-6**.

Table 7-6 GPTA Interrupt Sources

Cell Type	Number of Cells	Number of Interrupt Request Sources/Cell	Total Number of Sources
DCM	4	3	12
PLL	1	1	1
GT	2	1	2
GTC	32	1	32
LTC	64	1	64

To reduce hardware and software overhead, some GPTA interrupt sources belonging to the same cell type are summarized in service request groups enumerated by **Table 7-7**. A Service_request will be the output of each group and will drive a standard interrupt node (**Figure 7-46**).

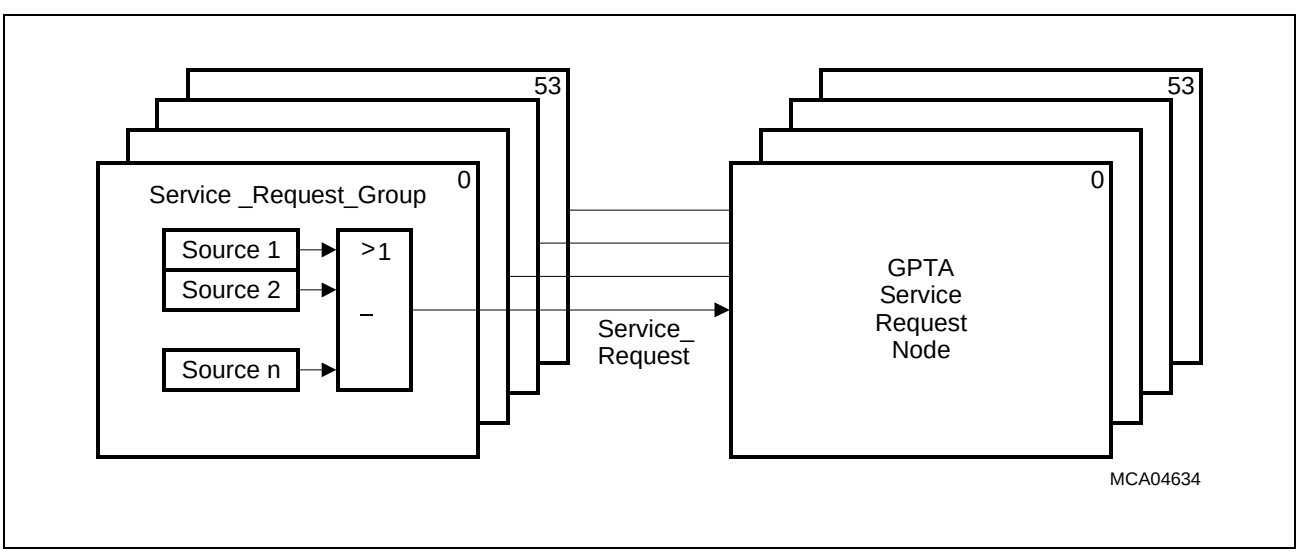


Figure 7-46 Service Request Group

Table 7-7 GPTA Interrupt Service Request Groups

Service Request Group Number	Source 1	Source 2	Source 3
00	DCM0 rising edge	DCM0 falling edge	DCM0 compare
01	DCM1 rising edge	DCM1 falling edge	DCM1 compare
02	DCM2 rising edge	DCM2 falling edge	DCM2 compare
03	DCM3 rising edge	DCM3 falling edge	DCM3 compare
04	PLL	–	–
05	GT0	GT1	–
06	GTC00	GTC01	–
07	GTC02	GTC03	–
08	GTC04	GTC05	–
09	GTC06	GTC07	–
10	GTC08	GTC09	–
11	GTC10	GTC11	–
12	GTC12	GTC13	–
13	GTC14	GTC15	–
14	GTC16	GTC17	–
15	GTC18	GTC19	–
16	GTC20	GTC21	–
17	GTC22	GTC23	–
18	GTC24	GTC25	–
19	GTC26	GTC27	–
20	GTC28	GTC29	–
21	GTC30	GTC31	–
22	LTC00	LTC01	–
23	LTC02	LTC03	–
24	LTC04	LTC05	–
25	LTC06	LTC07	–
26	LTC08	LTC09	–
27	LTC10	LTC11	–
28	LTC12	LTC13	–

Table 7-7 GPTA Interrupt Service Request Groups (cont'd)

Service Request Group Number	Source 1	Source 2	Source 3
29	LTC14	LTC15	–
30	LTC16	LTC17	–
31	LTC18	LTC19	–
32	LTC20	LTC21	–
33	LTC22	LTC23	–
34	LTC24	LTC25	–
35	LTC26	LTC27	–
36	LTC28	LTC29	–
37	LTC30	LTC31	–
38	LTC32	LTC33	–
39	LTC34	LTC35	–
40	LTC36	LTC37	–
41	LTC38	LTC39	–
42	LTC40	LTC41	–
43	LTC42	LTC43	–
44	LTC44	LTC45	–
45	LTC46	LTC47	–
46	LTC48	LTC49	–
47	LTC50	LTC51	–
48	LTC52	LTC53	–
49	LTC54	LTC55	–
50	LTC56	LTC57	–
51	LTC58	LTC59	–
52	LTC60	LTC61	–
53	LTC62	LTC63	–

The source of an interrupt generated by a GPTA service request group may be identified by scanning the service request state register SRS0 … SRS3. Each GPTA service request source is represented by an individual flag set by either a hardware trigger or by a software write access to SRSx. The associated flag is reset by software only. See **Figure 7-47**.

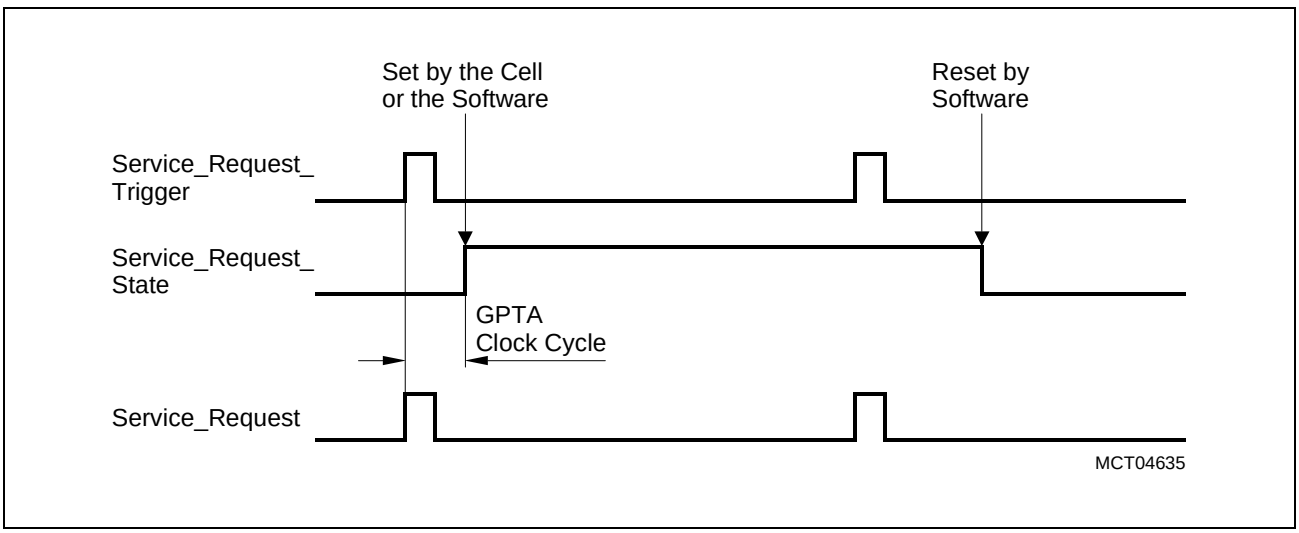


Figure 7-47 Service Request, Trigger, Set, Reset State

A GPTA service request is indicated by an interrupt flag in the associated interrupt control register, but the interrupt processing is started only when the corresponding interrupt enable flag has been set to 1.

7.1.8 PseudoCode Description of GPTA Kernel Functionality

7.1.8.1 FPC-Filter-Algorithm

FPCk_control_logic() “to be performed every GPTA clock”

```
if (FPCk.Mode == PRESCALER_RISING) then
   FPCk.Signal_output.level = FPCk.Signal_output.transition
   if ( (FPCk.Input_source == 0) or (FPCk.Rising_edge) ) then //
      if (FPCk.Timer >= FPCk.Compare_value) then
         perform(FPCk.Signal_output.transition)
         FPCk.Timer = 0
      else
         FPCk.Timer ++
      endif
   endif
endif
```

```
if (FPCk.Mode == PRESCALER_FALLING)
then FPCk.Signal_output.level = FPCk.Signal_output.transition
,    if (FPCk.Falling_edge)
     then if (FPCk.Timer >= FPCk.Compare_value)
        then perform(FPCk.Signal_output.transition)
             FPCk.Timer = 0
        else FPCk.Timer ++
        endif
     endif
endif
```

```
if (FPCk.Mode == LOW_PASS_FILTER) then
  if (FPCk.Timer >= FPCk.Compare_value) then
    if (FPCk.Compare_value == 0)  then            // by-pass
      FPCk.Signal_output = FPCk.Sampled_input
    else
      FPCk.Signal_output = ! FPCk.Signal_output
    endif
    FPCk.Timer = 0                          // timer reset
  else
    if ( ( (FPCk.Signal_output == 1) and (FPCk.Sampled_input == 0) )   or
      ( (FPCk.Signal_output == 0) and (FPCk.Sampled_input == 1) )  ) then
      FPCk.Timer ++
    else
      if (FPCk.Timer <> 0) then
        FPCk.Timer --
        FPCk.Glitch_record = 1
      endif
    endif
  endif
endif
```

```
if (FPCk.Mode == HIGH_PASS_FILTER) then
  if (FPCk.Timer == 0) then
    FPCk.Signal_output = FPCk.Sampled_input
    if ( ( (FPCk.Rising_edge) or (FPCk.Falling_edge) ) and
      (FPCk.Compare_value <> 0) ) then            // by-pass filtering"
      FPCk.Timer ++
    endif
  else
    if (FPCk.Timer >= FPCk.Compare_value) then
      FPCk.Timer = 0                        // Timer Reset
    else
      FPCk.Timer ++
    endif
    if ( (FPCk.Rising_edge) or (FPCk.Falling_edge) ) then
      FPCk.Glitch_record = 1
    endif
  endif
endif
```

Variables

Input, Local, Output variables of the cell (I, L, O)

Name k = [0, 1, 2, 3, 4, 5] m = [0, 1, 2, 3]	Short Name (*)FCP	Used (ILO)	Comment
FPCk.Signal_input[m]	*SINm	I	Input of the cell selected by the mux
FPCk.Rising_edge	*RE	L	Signal coming from the edge detect
FPCk.Falling_edge	*FE	L	Signal coming from the edge detect
FPCk.Sampled_input	*SAI	L	Is the image of the Signal_input sampled at the GPTA clock rate
FPCk.Signal_output.Transition FPCk.Signal_output.Level	*SOTk *SOLk	O	This information is made with the level and the transition of the signal, it is initialized to level 0 at reset

Global variables

Name k = [0, 1, 2, 3, 4, 5]	Short Name (*)FCP	Size (bits)	Function
FPCk.Mode	*MODk	2	00 “=” LOW_PASS_FILTER“=” 01 “=” HIGH_PASS_FILTER 10 “=” PRESCALER_RISING 11 “=” PRESCALER_FALLING
FPCk.Input_source	*IPSk	2	00 “=” pin (not for prescaler rising) 01 “=” pin, 10 “=” pin, 11 “=” pin
FPCk.Glitch_record	*GRCk	1	bit is set when glitch occurs during filtering
FPCk.Timer	*TIMk	16	Value of the timer
FPCk.Compare_value	*COMk	16	Compare value of the timer

Low Pass Filter Mode corresponds to the delayed debounce filter mode.

High Pass Filter Mode corresponds to the immediate debounce filter mode.

7.1.8.2 PDL-Algorithm

pdl0_control_logic()

```
if ((FPC0.Signal_output.Level) xor (FPC1.Signal_output.Level)) then
    if (FPC0.Signal_output.Transition) then
        Make a pulse on PDL0.Forward
    endif
    if (FPC1.Signal_output.Transition) then
       Make a pulse on PDL0.Backward
    endif
else
    if (FPC0.Signal_output.Transition) then
       Make a pulse on PDL0.Backward
    endif
    if (FPC1.Signal_output.Transition) then
       Make a pulse on PDL0.Forward
    endif
endif

if ((FPC2.Signal_output.Transition) and (PDL0.Three_sensors_enable))
then if ((FPC1.Signal_output.Level) xor (FPC2.Signal_output.Level))
    then Make a pulse on PDL0.Backward
    else Make a pulse on PDL0.Forward
    endif
endif

if ( (    (FPC0.Signal_output.Level)
    and (FPC1.Signal_output.Level)
    and (FPC2.Signal_output.Level))
  or(   !(FPC0.Signal_output.Level)
    and!(FPC1.Signal_output.Level)
    and!(FPC2.Signal_output.Level)))
then PDL0.Error =1
else PDL0.Error = 0
endif
```

pdl1_control_logic()

After replacing
PDL0 by PDL1
FPC0 by FPC3
FPC1 by FPC4
FPC2 by FPC5
the flow chart written for pdl0_control_logic() can be used.

Variables

Input, Local, Output variables of the cell (I, L, O)

Name k = [0, 1, 2, 3, 4, 5] m = [0, 1, 2, 3]	**Short Name (*)PDL**	**Used (ILO)**	**Comment**
FPCk.Signal_output.Transition FPCk.Signal_output.Level	SOTk SOLk	I	Signal coming from FPC
DCMm.Signal_input.Transition DCMm.Signal_input.Level	SITk SILk	O	Signal going to DCM
PDL0.Forward	*F0	O	This signal will go to an output pin and will be counted by LTC
PDL0.Backward	*B0	O	This signal will go to an output pin and will be counted by LTC
PDL1.Forward	*F1	O	This signal will go to an output pin and will be counted by LTC
PDL1.Backward	*B1	O	This signal will go to an output pin and will be counted by LTC

Global variables

Name	Short Name (*)PDL	Size (bits)	Function
PDL0.Mux	*MUX0	1	Provides DCM0.Signal_input from 0 “=” FPC0.Signal_output 1 “=” PDL0.Forward or Backward
PDL0.Three_sensors_enable	*TSE0	1	Selects the 3-sensors option and provides DCM1.Signal_input from 0 “=” FPC2.Signal_output 1 “=” PDL0.Error
PDL0.Error	*ERR0	1	Allows the software to read the error
PDL1.Mux	*MUX1	1	Provides DCM2.Signal_input from 0 “=” FPC3.Signal_output 1 “=” PDL1.Forward or Backward
PDL1.Three_sensors_enable	*TSE0	1	Selects the 3-sensors option and provides DCM3.Signal_input from 0 “=” FPC5.Signal_output 1 “=” PDL1.Error
PDL1.Error	*ERR1	1	Allows the software to read the error

7.1.8.3 DCM-Algorithm

DCMk_control_logic() "to be performed every GPTA clock"

```
if (( DCMk.Signal_input.Transition) and ( DCMk.Signal_input.Level)) then
    trig ( DCMk.Service_request_trig_rising)
    if ( DCMk.Capture_on_rising_edge) then
        DCMk.Capture_value =  DCMk.Timer
    else
       if ( DCMk.Capcom_opposite) then
           DCMk.Capcom_value =  DCMk.Timer
       endif
    endif
    if ( DCMk.Clear_on_rising_edge) then
        DCMk.Timer = 0
    endif
    if ( DCMk.Clock_on_rising_edge) then
       Generate  DCMk.Signal_output
    endif
endif
```

```
if (( DCMk.Signal_input.Transition) and (! DCMk.Signal_input.Level)) then
    trig( DCMk.Service_request_trig_falling)
    if (! DCMk.Capture_on_rising_edge) then
        DCMk.Capture_value = DCMk.Timer
    else
       if ( DCMk.Capcom_opposite) then
           DCMk.Capcom_value = DCMk.Timer
       endif
    endif
    if ( DCMk.Clear_on_falling_edge) then
        DCMk.Timer = 0
    endif
    if ( DCMk.Clock_on_falling_edge) then
       Generate  DCMk.Signal_output
    endif
endif
```

DCMk_control_logic()	“to be performed every GPTA clock”

if (DCMk.Compare_event) then
trig(DCMk.Service_request_trig_compare)
endif	

if (DCMk.Clock_request) then
Generate DCMk.Signal_output
DCMk.Clock_request = 0
endif	

Variables

Input, Local, Output variables of the cell (I, L, O)

Name	Short Name	Used (ILO)	Comment
DCMk.Signal_input.Transition DCMk.Signal_input.Level	*SITk *SILk	I	Is the input of the cell
DCMk.Compare_event	CE	L	Is set when Timer = Capcom_value
DCMk.Signal_output	*SOk	O	Is the output of the cell
DCMk.Service_request_trig_rising	*RTQk	O	Trigger on rising edge
DCMk.Service_request_trig_falling	*FTQk	O	Trigger on falling edge
DCMk.Service_request_trig_compare	*CTQk	O	Trigger on compare event

Global variables

Name k = [0, 1, 2, 3]	Short Name (*)DCM	Size (bits)	Function
DCMk.Capture_on_rising_edge	*RCAk	1	Capture into Capture_value 0 “=” then capture on falling edge 1 “=” then capture on rising edge
DCMk.Capcom_opposite	*OCAk	1	0 “=” no capture into Capcom_value 1 “=” capture into Capcom_value on the opposite edge defined by RCAk
DCMk.Clear_on_rising_edge	*RZEk	1	1 “=” then Timer = 0 on rising edge
DCMk.Clear_on_falling_edge	*FZEk	1	1 “=” then Timer = 0 on falling edge
DCMk.Clock_on_rising_edge	*RCKk	1	1 “=” then generates a single clock pulse on rising edge
DCMk.Clock_on_falling_edge	*FCKk	1	1 “=” then generates a single clock pulse on falling edge
DCMk.Clock_request	*QCKk	1	1 “=” generates one extra clock 0 “=” doesn’t have any effect will always be read as 0
DCMk.Req_enable_on_rising_edge	*RREk	1	Request enable on rising edge
DCMk.Req_enable_on_falling_edge	*FREk	1	Request enable on falling edge
DCMk.Req_enable_on_compare	*CREk	1	Request enable on compare
DCMk.Timer	*TIMk	24	Timer value
DCMk.Capture_value	*CAVk	24	Capture value
DCMk.Capcom_value	*COVk	24	Capture and compare value

7.1.8.4 PLL-Algorithm

pll_control_logic() "to be performed every GPTA clock"

```
if ((Pll.Automatic_end) and (Pll.Event)) then
   Pll.Perform_end = 1
endif
if ((Pll.Counter == 0) and (Pll.Perform_end)) then
   Pll.Counter = Pll.Nb_mtick
   Pll.Perfom_end = 0
endif
if ((Pll.Counter != 0) and ((Pll.Perform_end) or (bit 24 of Delta))) then
   Clock on Pll.Signal_output
   Pll.Counter --
   if (Pll.Counter == 0) then
      trig(Pll.Service_request_trigger)
   endif
endif
if (bit 24 of Delta) then
   Pll.Delta = Pll.Delta + Pll.Reload_value
else
   Pll.Delta = Pll.Delta + (0xFFFF0000 or Pll.Step)
endif
```

Variables

Input, Local, Output variables of the cell (I, L, O)

Name k = [0, 3]	**Short Name (*)PLL**	**Used (ILO)**	**Comment**
DCMk.Signal_output	SOk	I	Is the input of the cell from DCM
Pll.Event	*EVE	L	Is the selected input by the mux
Pll.Signal_output	*SO	O	Is the output of the cell
Pll.Service_request_trigger	*SQT	O	Trigger when Counter reaches zero

Name	Short Name (*)PLL	Size (bits)	Function
Pll.mux	*MUX	2	Selects the Signal_input of the PLL 00 “=” DCM0.Signal.output 01 “=” DCM1.Signal.output 10 “=” DCM2.Signal.output 11 “=” DCM3.Signal.output
Pll.Automatic_end	*AEN	1	Performs the accel decel correction
Pll.Perform_end	*PEN	1	Allows to decrement the Counter full speed
Pll.Request_enable	*REN	1	Allows a request when Counter reach zero
Pll.Nb_mtick	*MTI	16	Is the reload of the Counter
Pll.Counter	*CNT	16	Counter of microticks
Pll.Step	*STP	16	Number of steps per period
Pll.Reload_value	*REV	24	Is the Period - Step
Pll.Delta	*DTA	25	Is the reminder of the PLL

7.1.8.5 GT-Algorithm

Timer increment

```
if (Event on selected clock from the clock bus) then
    GTm.Timer = GTm.Timer + 1
    if (Overflow of the timer) then
      GTm.Timer = GTm.Reload_timer
      trig(GTm.Service_request_trigger)
    endif
endif
```

Variables

Input, Local, Output variables of the cell (I, L, O)

Name m = [0, 1] p = [0, 7]	**Short Name (*)GT**	**Used (ILO)**	**Comment**
GTm.Clock_in[p]	*CINkp	I	Inputs coming from pad bus
GTm.Timer_greater_equal_comp	*TGEm	O	Is set when greater or equal is true
GTm.Timer_event	*TEVm	O	Is set when the timer changes
GTm.Service_request_trigger	*SQTm	O	Is set when the timer overflows

Global variables

Name m = [0, 1]	**Short Name (*)GT**	**Size (bits)**	**Function**
GTm.Scale_compare	*SCOm	4	Selects the compare flag
GTm.Clock_mux	*MUXm	3	Selects the clock from the clock bus
GTm.Request_enable	*RENm	1	Allows a request when the timer overflows
GTm.Timer	*TIMm	24	Value of the timer
GTm.Reload_value	*REVm	24	Reloads value when the timer overflows

7.1.8.6 GTC-Algorithm

GTCk_control_logic()

```
if (GTCk.Cell_enable) then
   switch (GTCk.Mode)
      case CAPTURE_T0:
         capture (0)
         break
      case CAPTURE_T1:
         capture (1)
         break
      case COMPARE_T0:
         compare (0)
         break
      case COMPARE_T1:
         compare (1)
   endswitch

   if ( (GTCk.One_shot_mode) and (GTCk.Event) ) then
      GTCk.Cell_enable = 0
   endif
endif
```

capture(m)

```
if (GTCk.Signal_input) then
   trig(GTCk.Service_request_trigger)
   GTCk.X = GTm.Timer
   GTCk.Event  = 1
else
   GTCk.Event  = 0
endif
```

compare(m)

```
if   (
    (GTCk.X == GTm.Timer)
              and
    ( (GTCk.X_write_access) or (GTm.Timer_event) )
              or
    ( (GTCk.Greater_equal_select) and (GTCk.X_write_access) and
                                        (GTm.Timer_greater_equal_comp))
    ) then

    if (GTCk.Capture_after_compare) then
      if (GTCk.Capture_opposite_timer) then
        GTCk.X = GTm.Timer
      else
        GTCk.X = GTm.Timer
      endif
    endif
    trig(GTCk.Service_request_trigger)
    GTCk.Event  = 1
else
    GTCk.Event  = 0
endif
```

Variables

Input, Local, Output variables of the cell (I, L, O)

Name k = [0, 31] m = [0, 1] p = [0, 3]	Short Name (*)GTC	Used (ILO)	Comment
GTm.Timer_greater_equal_comp	TGEm	I	Input coming from GT
GTm.Timer_event	TEVm	I	Input coming from GT
GTm.Timer	TIMm	I	Input coming from GT
GTCk.Data_in[p]	*DINkp	I	Inputs coming from pad bus
GTCk.Output_mode_0_in	*M0Ik	I	Signal coming from the previous cell
GTCk.Output_mode_1_in	*M1Ik	I	Signal coming from the previous cell
GTCk.X_write_access	*XWAk	L	Internal value set to indicate that GTCk.X was modified and the compare function needs to be recomputed
GTCk.Event	*EVE	L	Internal signal
GTCk.Signal_input	*INS	L	Internal signal
GTCk.Service_request_trigger	*SQSk	O	Trigger when an event occurs
GTCk.Data_out	*DOUk	O	Output going to pad bus
GTCk.Output_mode_0_out	*M0Ok	O	Signal going to the next cell
GTCk.Output_mode_1_out	*M1Ok	O	Signal going to the next cell

Gobal variables

Name k = [0, 31]	Short Name (*)GTC	Size (bits)	Comment
GTCk.Mode	*MODk	2	00 “=” CAPTURE_T0 01 “=” CAPTURE_T1 10 “=” COMPARE_T0 11 “=” COMPARE_T1
GTCk.One_shot_mode	*OSMk	1	Mode is stopped after the first event 0 “=” normal mode 1 “=” one shot mode

Gobal variables (cont'd)

Name k = [0, 31]	Short Name (*)GTC	Size (bits)	Comment
GTCk.Request_enable	*RENk	1	Allows a request on event
GTCk.Input_rising_edge_select # GTCk.Greater_equal_select	*REDk #GESk	1 #1	Selects the rising edge of the input pin This bit selects also the Compare Mode 0 "=" compare 1 "≥" compare
GTCk.Input_falling_edge_select # GTCk.Capture_after_compare	*FEDk #CACk	1 #1	Selects the falling edge of the input pin This bits also selects the capture after 0 "=" no capture after compare 1 "=" capture after compare
GTCk.Input_pin_mux GTCk.Capture_opposite_timer	*PMUk #COTk	2 #1	Selects the input on Timer Mode or Capture Mode Capture the opposite global timer after a greater or equal compare 0 "=" selected global timer 1 "=" opposite global timer
GTCk.Cell_enable	*CENk	1	0 "=" cell disabled 1 "=" cell enabled Will be set at each register write access Will be reset in OSM at the first event
GTCk.Output_control_mode	*OCMk	3	Output update
GTCk.Output_immediate_action	*OIAk	1	Read 0: always Write 0: no effect Write 1: force the action to occur
GTCk.Output_state	*OUTk	1	Read the value of Data_out
GTCk.X	*Xk	24	Value of X

(#) second definition in Compare Mode

7.1.8.7 LTC-Algorithm

LTCk_control_logic()

```
if (LTCk.Cell_enable) then
   switch (LTCk.Mode)
          case CAPTURE:
                capture ()
                break
          case COMPARE:
                compare ()
                break
          case TIMER_FREE_RUN:
                timer()
                break
          case TIMER_RESET:
                if (LTCk.Event_in) then
                   LTCk.Reset_timer = 1
                endif
               timer()
               break;
   endswitch
   if ((LTCk.One_shot_mode) and (LTCk.Event)) then
      LTCk.Cell_enable = 0
   endif
endif
manage_mux()
```

capture()

```
if (LTCk.Signal_input) then
      trig(LTCk.Service_request_trigger)
      LTCk.X = LTCk.Y_in
      LTCk.Event  = 1
else LTCk.Event  = 0
endif
LTCk.Event_out = LTCk.Event
```

compare()

```
if (  (LTCk.Select_in == LTCk.Select_on_high_level)
  or (LTCk.Select_in == !LTCk.Select_on_low_level)) then
       if ((LTCk.X == LTCk.Y_in) and ((LTCk.X_write_access) or
(LTCk.Timer_event_in))) then
            trig(LTCk.Service_request_trigger)
            LTCk.Event  = 1
       else LTCk.Event  = 0
       endif
       LTCk.Event_out = LTCk.Event
else LTCk.Event_out = LTCk.Event_in
endif
```

timer()

```
if ((LTCk.X == 0xFFFF) and (LTCk.X_write_access)) then
    // Above condition is also true for timer overflow or software reset  //
    trig(LTCk.Service_request_trigger)
    LTCk.Event  = 1
else
    LTCk.Event  = 0
endif
if (LTCk.Signal_input) then
    if (LTCk.Reset_timer) then
       LTCk.Reset_timer = 0
       LTCk.X = 0xFFFF
       if (LTCk.Coherent_update_enable) then
            LTCk.Select_line_value = !LTCk.Select_line_value
            LTCk.Coherent_update_enable = 0
       endif                          // free running Timer Mode //
    else
       LTCk.X ++
    endif
endif
LTCk.Event_out = LTCk.Event
```

manage_mux()

```
if ( (LTCk.Mode == TIMER_FREE_RUN) or (LTCk.Mode == TIMER_RESET) ) then
   LTCk.Y_out = LTCk.X
   if (the timer has been modified) then          // increment, reset, software overwrite //
      LTCk.Timer_event_out = 1
   else
      LTCk.Timer_event_out = 0
   endif
   LTCk.Select_out = LTCk.Select_line_value
else
   LTCk.Y_out = LTCk.Y_in
   LTCk.Timer_event_out = LTCk.Timer_event_in
   LTCk.Select_line_level = LTCk.Select_in
   LTCk.Select_out = LTCk.Select_in
endif
```

Variables

Input, Local, Output variables of the cell (I, L, O)Control Register Specification

Name	Short Name (*)LTC	Used (ILO)	Comment
LTCk.Clock_in[p]	*CINkp	I	Inputs coming from clock bus
LTCk.Data_in[p]	*DINkp	I	Inputs coming from pad bus
LTCk.Y_in	*YIk	I	Timer coming from the previous cell
LTCk.Output_mode_0_in	*M0Ik	I	Signal coming from the previous cell
LTCk.Output_mode_1_in	*M1Ik	I	Signal coming from the previous cell
LTCk.Timer_event_in	*TIk	I	Signal coming from the previous cell
LTCk.Event_in	*EIk	I	Signal coming from the following cell
LTCk.Select_in	*SI	I	Signal coming from the previous cell
LTCk.X_write_access	*XWA	L	Internal value set to indicate the LTCk.X was modified and the compare function must be recomputed
LTCk.Select_line_value	*SLV	L	Internal value for the select line reset value: 0
LTCk.Signal_input	*INS	L	Signal clocking for Timer Mode
LTCk.Reset_timer	*RTM	L	Flipflop to reset the timer on the next clock
LTCk.Data_out	*DOUk	O	Is the output going to pad bus
LTCk.Service_request_trigger	*SQTk	O	Trigger when an event occurs
LTCk.Y_out	*YOk	O	Timer going to the next cell
LTCk.Output_mode_0_out	*M0Ok	O	Signal going to the next cell
LTCk.Output_mode_1_out	*M1Ok	O	Signal going to the next cell
LTCk.Timer_event_out	*TOk	O	Signal going to the next cell
LTCk.Select_out	*SO	O	Signal going to the next cell
LTCk.Event_out also (Event)	*EOk	O	Signal used by the cell and going to the previous cell

7.1.9 Programming of the GPTA Unit

A hierarchical top-down design approach may be used to implement a complex signal processing unit as follows:

- Partition the complex signal processing unit into simple function units,
- Implement each simple function unit by configuring LTC and /or GTC cells which may be tied together realizing a common signal operation.
- Implement necessary signal preprocessing tasks by configuring the FPC, PDL, DCM and PLL cells accordingly.
- Define and configure all input/output port pins required as clock source, trigger input or signal output.

Figure 7-8 summarizes all software tasks to be implemented for getting a GPTA unit into operation.

Table 7-8 Software Tasks Controlling a GPTA Unit

Port Initialization	
Definition of Electrical Port Characteristic	–
Configuration of Port Pin Direction (Input or Output)	–
GPTA Shell Initialization	
GPTA Module Clock Enable (f_{GPTA}) Must be the very first register of GPTA to be programmed	–
Configuration of Interrupt Handling	–
GPTA Kernel Initialization	
FPC:	**PDL:**
Selection of Operating Mode (Prescaler, Filter or Feed-Through)	Selection of Operating Mode (Phase Discriminator or Feed-Through)
Input Channel Selection	2- or 3-Sensor Mode Selection
Configuration of Prescaler Factor or Debounce Mode	**PLL:**
DCM:	Selection of Input Channel
Selection of Reset Event for Timer	Estimation of Input Signal Period Width
Selection of Trigger Source for Capture Event	Configuration of Output Signal Frequency

Table 7-8 Software Tasks Controlling a GPTA Unit (cont'd)

Selection of Trigger Source for Capcom Register Update	Handling of Input Signal Period Length Variation
Interrupt Request Enable on Input Edge or Compare Event	Interrupt Request Enable on End of Output Pulse Generation
CKB:	
Selection and Configuration of 8 Clock Sources for	GT, GTC and LTC Cells
GT:	**GTC:**
Selection of Timer Clock Source	Selection of Operating Mode (Capture or Compare) and Time Base (GT0 or GT1)
Configuration of Timer Width (Reload Value, TGE Flag)	Configuration of Trigger Events for Capture Mode or Selection of a Relational Operator for Compare Mode
Interrupt Request Enable on Timer Overflow	Interrupt Request Enable on Capture or Compare Event
–	Configuration of Data Output triggered by a GTC Event
LTC:	**IOSU:**
Selection of Operating Mode (Timer, Capture or Compare)	Configuration of Output Line Multiplexer to link selected GTC and LTC data outputs to external Port Pins
Selection of Trigger Source for Timer, Capture or Compare Mode	–
Configuration of Trigger Event for Timer, Capture or Compare Mode	–
Interrupt Request Enable on Timer, Capture or Compare Event	–
Configuration of Data Output triggered by an LTC Event	–

7.2 GPTA Kernel Registers

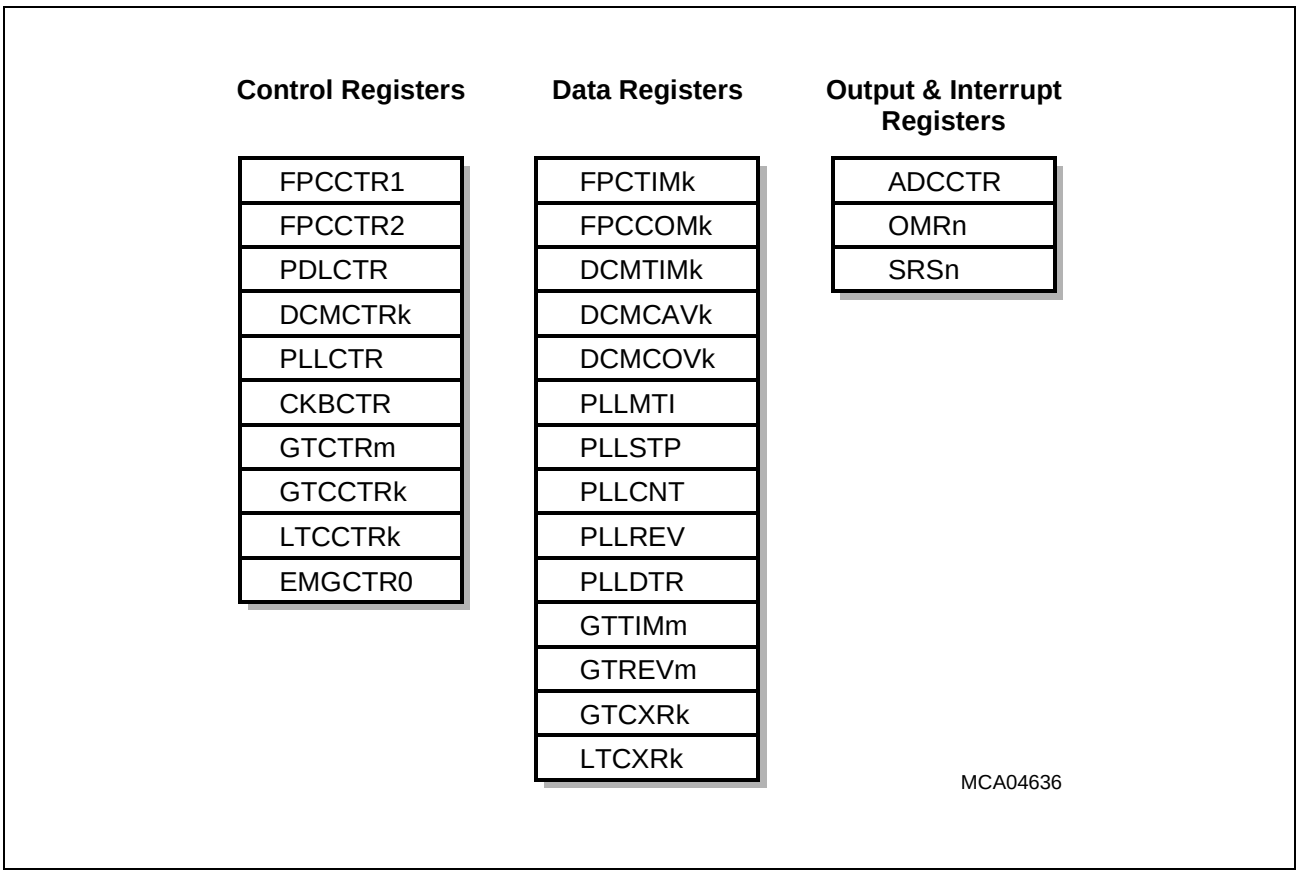


Figure 7-48 GPTA Kernel Registers

Table 7-9 GPTA Kernel Registers

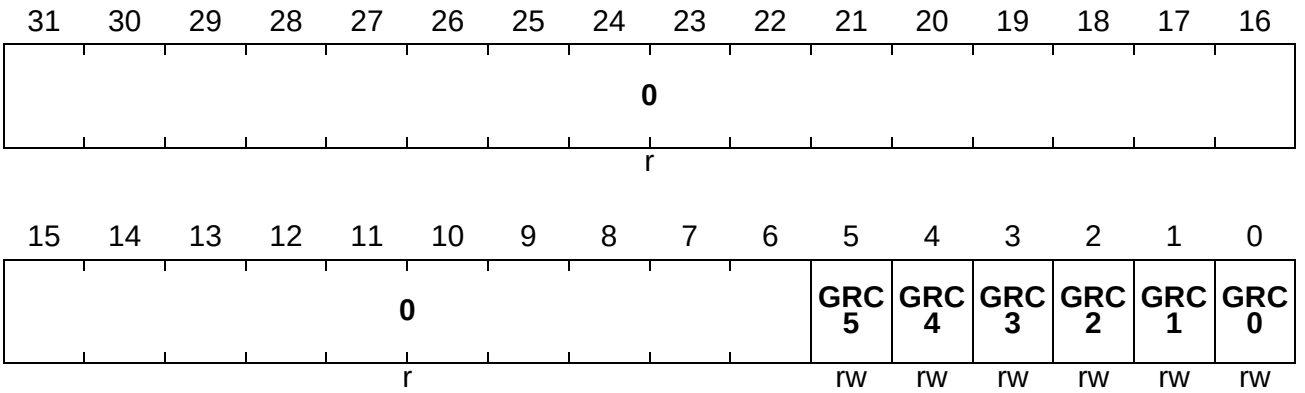
Register Short Name	Register Long Name	Offset Address	Description see
SRS0	Service Request State Register 0	0010_H	Page 7-124
SRS1	Service Request State Register 1	0014_H	Page 7-125
SRS2	Service Request State Register 2	0018_H	Page 7-126
SRS3	Service Request State Register 3	$001C_H$	Page 7-126
ADCCTR	ADC Multiplex Control Register	0030_H	Page 7-123
EMGCTR0	Emergency Control Register 0	0034_H	Page 7-121
EMGCTR1	Emergency Control Register 1	0038_H	Page 7-121
OMR0	Output Port Line Multiplex Register 0	$003C_H$	Page 7-119
OMR1	Output Port Line Multiplex Register 1	0040_H	Page 7-119
OMR2	Output Port Line Multiplex Register 2	0044_H	Page 7-119
OMR3	Output Port Line Multiplex Register 3	0048_H	Page 7-120
FPCCTR1	Filter and Prescaler Cell Control Register 1	0050_H	Page 7-96
FPCCTR2	Filter and Prescaler Cell Control Register 2	0054_H	Page 7-97
FPCTIMk	Filter and Prescaler Cell Timer Register k (k = 0-5)	$0058_H + k \times 8$	Page 7-98
FPCCOMk	Filter and Prescaler Cell Compare Register k (k = 0-5)	$005C_H + k \times 8 + 4$	Page 7-98
PDLCTR	Phase Discrimination Logic Control Register	0088_H	Page 7-99
DCMCTRk	Duty Cycle Measurement Control Register k (k = 0-3)	$008C_H + k \times 16$	Page 7-101
DCMTIMk	Duty Cycle Measurement Timer Register k (k = 0-3)	$0090_H + k \times 16 + 4$	Page 7-103
DCMCAVk	Duty Cycle Measurement Capture Register k (k = 0-3)	$0094_H + k \times 16 + 8$	Page 7-103
DCMCOVk	Duty Cycle Measurement Capture/Compare Register k (k = 0-3)	$0098_H + k \times 16 + 12$	Page 7-103
PLLCTR	Phase Locked Loop Control Register	$00CC_H$	Page 7-104
PLLMT	Phase Locked Loop Micro Tick Register	$00D0_H$	Page 7-105
PLLCNT	Phase Locked Loop Counter Register	$00D4_H$	Page 7-106
PLLSTP	Phase Locked Loop Step Register	$00D8_H$	Page 7-105

Table 7-9 GPTA Kernel Registers (cont'd)

Register Short Name	Register Long Name	Offset Address	Description see
PLLREV	Phase Locked Loop Reload Register	$00DC_H$	Page 7-106
PLLDTR	Phase Locked Loop Delta Register	$00E0_H$	Page 7-107
CKBCTR	Clock Bus Control Register	$00E4_H$	Page 7-108
GTCTR0	Global Timer Control Register 0	$00E8_H$	Page 7-109
GTREV0	Global Timer Reload Value Register 0	$00EC_H$	Page 7-110
GTTIM0	Global Timer Register 0	$00F0_H$	Page 7-110
GTCTR1	Global Timer Control Register 1	$00F4_H$	Page 7-109
GTREV1	Global Timer Reload Value Register 1	$00F8_H$	Page 7-110
GTTIM1	Global Timer Register 1	$00FC_H$	Page 7-110
GTCCTRk	Global Timer Cell Control Register k (k = 00-31)	$0100_H + k \times 8$	Page 7-111
GTCXRk	Global Timer Cell X Register k (k = 00-31)	$0100_H + k \times 8 + 4$	Page 7-114
LTCCTRk	Local Timer Cell Control Register k (k = 00-63)	$0200_H + k \times 8$	Page 7-115
LTCXRk	Local TImer Cell X Reg. k (k = 00-63)	$0200_H + k \times 8 + 4$	Page 7-118

7.2.1 FPC Registers

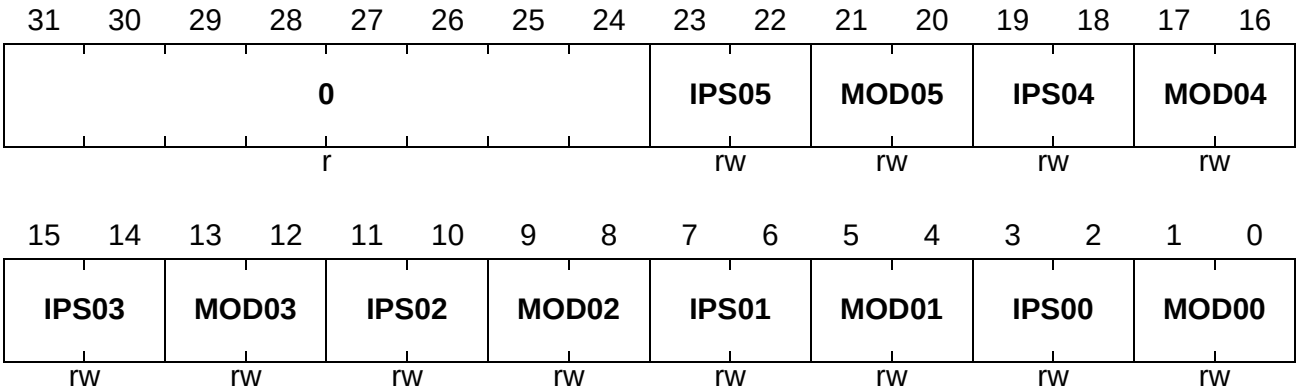
FPCCTR1
Filter and Prescaler Cell Control Register 1 **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
GRCk (k = 0-5)	k	rw	**Input Glitch Flag for FPCk** 0 No glitch occurred during filtering 1 Glitch detected during filtering Bit protection is implemented to allow “read modify write” instructions
0	[31:6]	r	**Reserved**; read as 0, should be written with 0.

FPCCTR2
Filter and Prescaler Cell Control Register 2

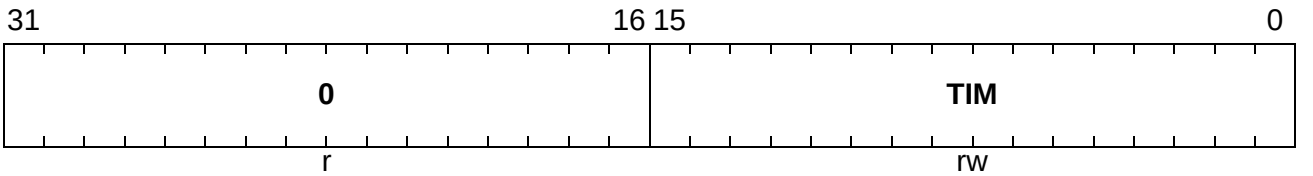
Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
MOD0k (k = 0-5)	[1:0] [5:4] [9:8] [13:12] [17:16] [21:20]	rw	**Operation Mode Selection for FPCk** 00 Low Pass Filter Mode 01 High Pass Filter Mode 10 Prescaler Mode (triggered on rising edge) 11 Prescaler Mode (triggered on falling edge)
IPS0k (k = 0-5)	[3:2] [7:6] [11:10] [15:14] [19:18] [23:22]	rw	**Input Line Selection for FPCk** 00 Signal line input 0 or GPTA module clock select 01 Signal line input 1 selected 10 Signal line input 2 selected 11 Signal line input 3 selected
0	[31:6]	r	**Reserved**; read as 0; should be written with 0.

FPCTIMk (k = 0-5)
Filter and Prescaler Cell Timer Register k

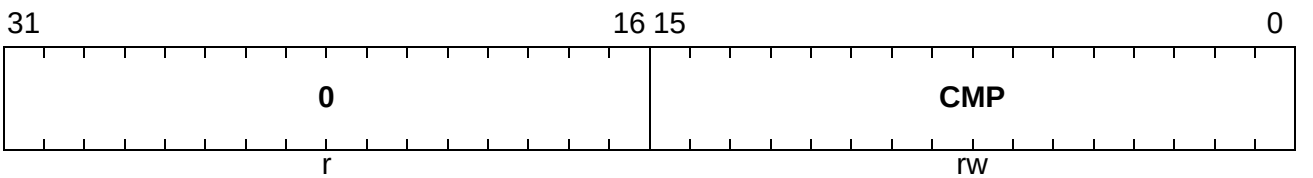
Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
TIM	[15:0]	rw	**Timer Value of Filter and Prescaler Cell k**
0	[31:16]	r	**Reserved**; read as 0; should be written with 0.

FPCCOMk (k = 0-5)
Filter and Prescaler Cell Compare Register k

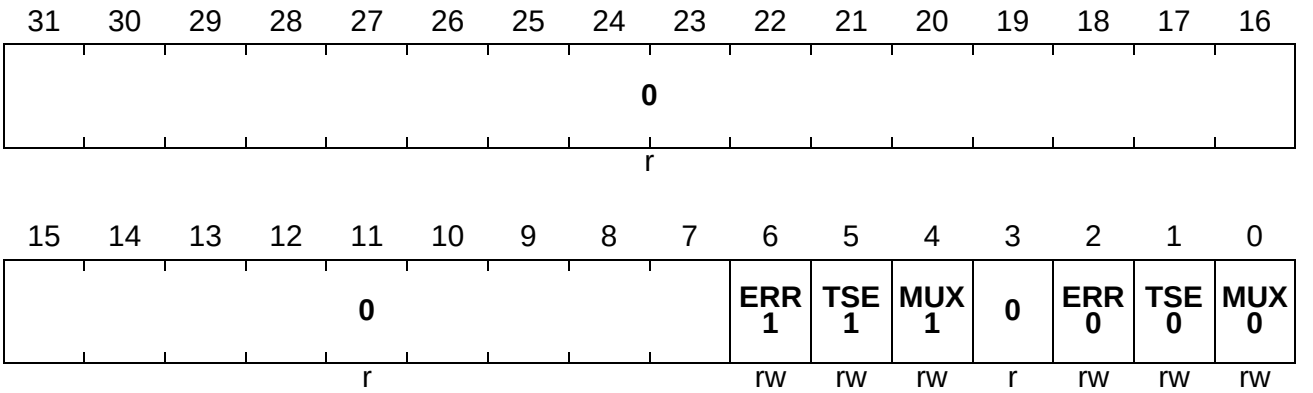
Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
CMP	[15:0]	rw	**Threshold value of Filter and Prescaler Cell k to be compared with timer value**
0	[31:16]	r	**Reserved**; read as 0; should be written with 0.

7.2.2 Phase Discriminator Logic Register

PDLCTR
Phase Discrimination Logic Control Register **Reset Value: 0000 0000$_H$**

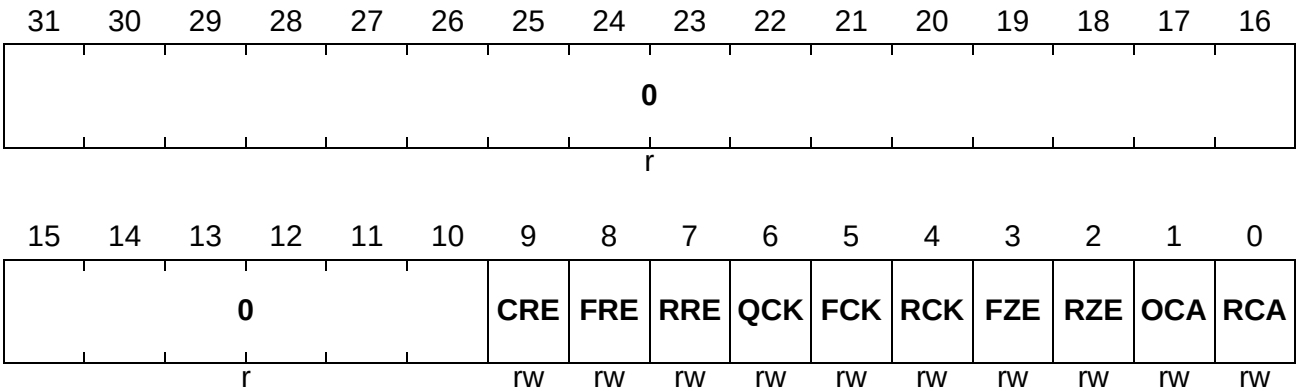


Field	Bits	Type	Description
MUX0	0	rw	**Output Signal Source Selection for PDL0** 0 DCM0 cell input is driven by fed-through PFC0 output lines 1 DCM0 cell input is provided with PDL0 “Forward” and “Backward” pulses
TSE0	1	rw	**3-Sensor Mode Enable for PDL0** 0 PDL0 operates in “2-Sensor Mode” and DCM1 cell input is driven by fed-through PFC2 output lines 1 PDL0 operates in “3-Sensor Mode” and DCM1 cell input is provided with PDL0 error information
ERR0	2	rw	**Error Flag for PDL0** 0 No error is occurred 1 Error detected in “3-Sensor Mode”: all PDL0 input signals are simultaneously provided with high or low level.
MUX1	4	rw	**Output Signal Source Selection for PDL1** 0 DCM2 cell input is driven by fed-through PFC3 output lines 1 DCM2 cell input is provided with PDL1 “Forward” and “Backward” pulses

Field	Bits	Type	Description
TSE1	5	rw	**3-Sensor Mode Enable for PDL1** 0 PDL1 operates in “2-Sensor Mode” and DCM3 cell input is driven by fed-through PFC5 output lines. 1 PDL1 operates in “3-Sensor Mode” and DCM3 cell input is provided with PDL1 error information.
ERR1	6	rw	**Error Flag for PDL1** 0 No error is occurred 1 Error detected in “3-Sensor Mode”: all PDL1 input signals are simultaneously provided with high or low level.
0	3, [31:7]	r	**Reserved**; read as 0; should be written with 0.

7.2.3 Duty Cycle Measurement Register

DCMCTk (k = 0-3)
Duty Cycle Measurement Control Register k **Reset Value: 0000 0000$_H$**

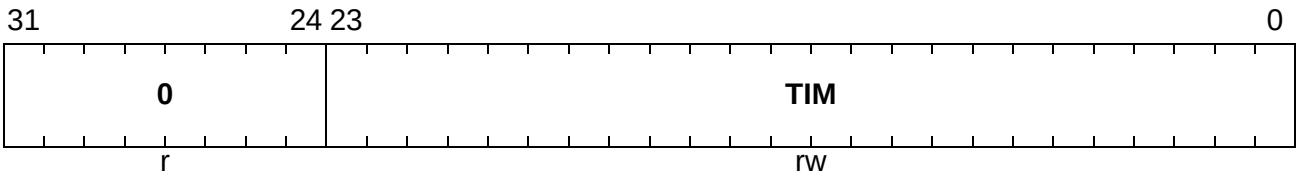


Field	Bits	Type	Description
RCA	0	rw	**Trigger Source Selection for Capture Event** 0 Timer contents are copied to DCMCAVk capture register on a falling input signal edge 1 Timer contents are copied to capture register on a rising input signal edge
OCAk	1	rw	**Trigger Source for Capcom Register Update** 0 Capcom register is not affected 1 Timer contents are copied to DCMCOVk capcom register on the opposite edge selected by RCAk
RZE	2	rw	**Timer Reset on Rising Edge** 0 Timer is not affected 1 Timer is reset on a rising input signal edge
FZE	3	rw	**Timer Reset on Falling Edge** 0 Timer is not affected 1 Timer is reset on a falling input signal edge
RCK	4	rw	**Output Pulse on Rising Edge** 0 DCM output line is not affected 1 DCM output line is provided with a single clock pulse generated on a rising input signal edge
FCK	5	rw	**Output Pulse on Falling Edge** 0 DCM output line is not affected 1 DCM output line is provided with a single clock pulse generated on a falling input signal edge

Field	Bits	Type	Description
QCK	6	rw	**Additional Output Pulse Generation** 0 DCM output line is not affected 1 DCM output line is immediately provided with a single clock pulse This bit is reset automatically
RRE	7	rw	**Interrupt Request on Rising Edge** 0 Interrupt request is not affected 1 Interrupt request is set on rising input signal edge
FRE	8	rw	**Interrupt Request on Falling Edge** 0 Interrupt request is not affected 1 Interrupt request is set on falling input signal edge
CRE	9	rw	**Interrupt Request on Compare Event** 0 No interrupt request enabled on a compare event 1 An interrupt request is generated when the timer contents matches the value currently stored in capcom register DCMCOVk
0	[31:10]	r	**Reserved**; read as 0; should be written with 0.

DCMTIMk (k = 0-3)
Duty Cycle Measurement Timer Register k

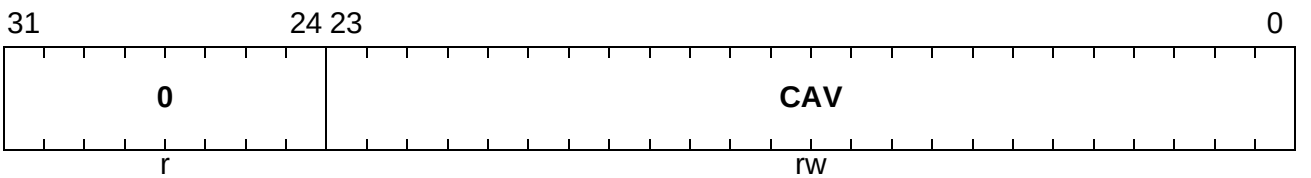
Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
TIM	[23:0]	rw	**Timer Value of DCMk**
ß	[31:24]	r	**Reserved**; read as 0; should be written with 0.

DCMCAVk (k = 0-3)
Duty Cycle Measurement Capture Register k

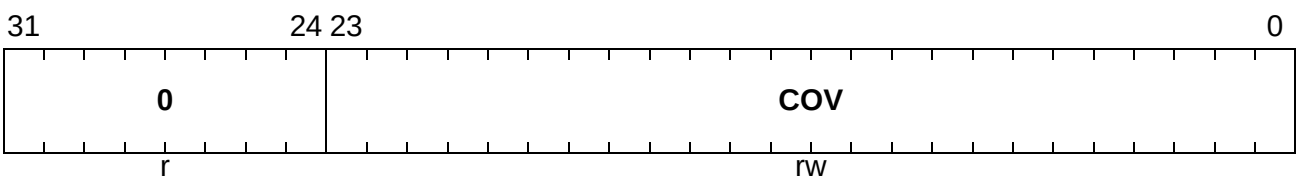
Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
CAV	[23:0]	rw	**Compare Value of DCMk**
0	[31:24]	r	**Reserved**; read as 0; should be written with 0.

DCMCOVk (k = 0-3)
Duty Cycle Measurement Capture/Compare Register k

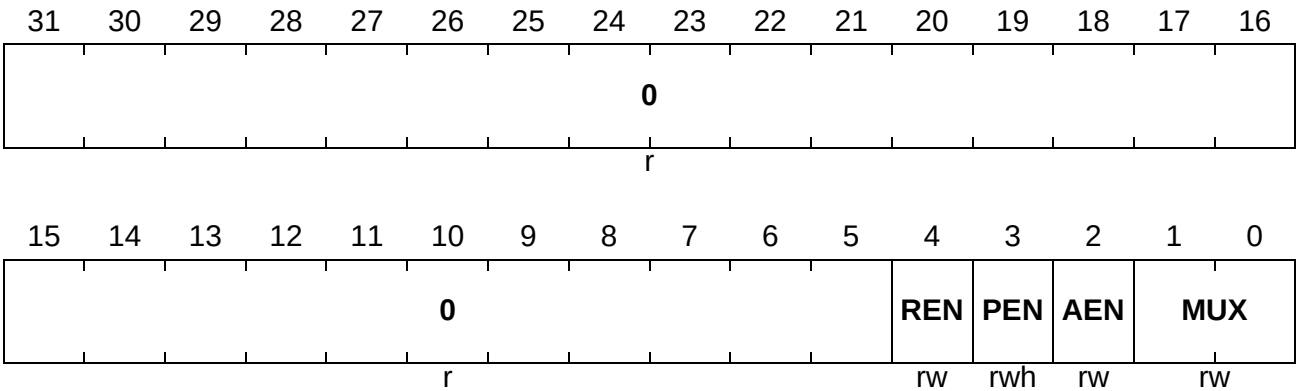
Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
COV	[23:0]	rw	**Capture/Compare Register Value of DCMk**
0	[31:24]	r	**Reserved**; read as 0; should be written with 0.

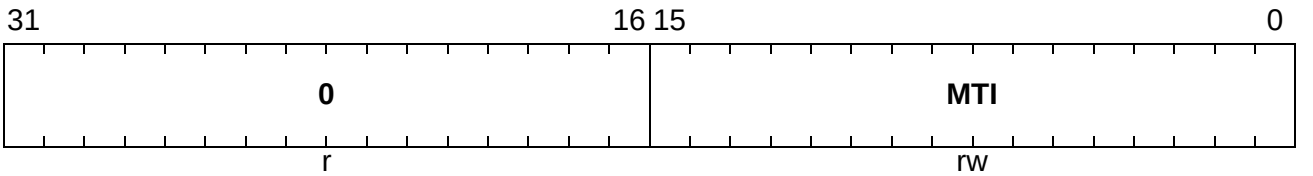
7.2.4 Digital Phase Locked Loop Register

PLLCTR
Phase Locked Loop Control Register **Reset Value: 0000 0000$_H$**



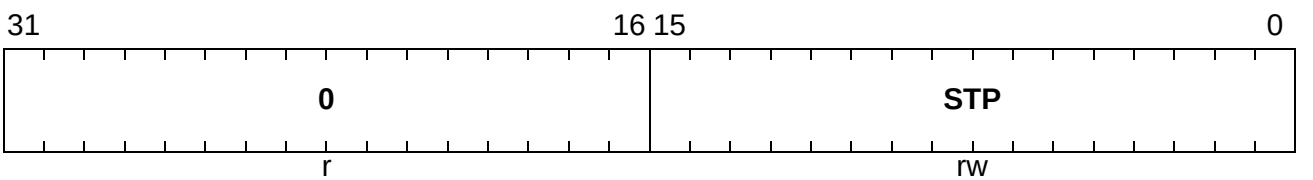
Field	Bits	Type	Description
MUX	[1:0]	rw	**Trigger Input Channel Selection** 00 DCM0 output is selected as PLL input 01 DCM1 output is selected as PLL input 10 DCM2 output is selected as PLL input 11 DCM3 output is selected as PLL input
AEN	2	rw	**Compensation of Input Period Length Variation** 0 Compensation of input signal's period length variation is disabled 1 Compensation of input signal's period length variation (acceleration, deceleration) is requested
PEN	3	rwh	**Unexpected Period End Behavior** 0 Counter decrements with constant frequency 1 Counter is allowed to decrement with f_{GPTA} frequency in case of an input signal period length' reduction Programming PEN to 1 immediately changes the microtick counter to decrement with f_{GPTA} frequency.
REN	4	rw	**Interrupt Service Request Enable** 0 Interrupt request is disabled 1 An interrupt request is set when the number of remaining output pulses to be generated reaches zero
0	[31:5]	r	**Reserved**; read as 0; should be written with 0.

PLLMTI
Phase Locked Loop Microtick Register **Reset Value: 0000 0000$_H$**



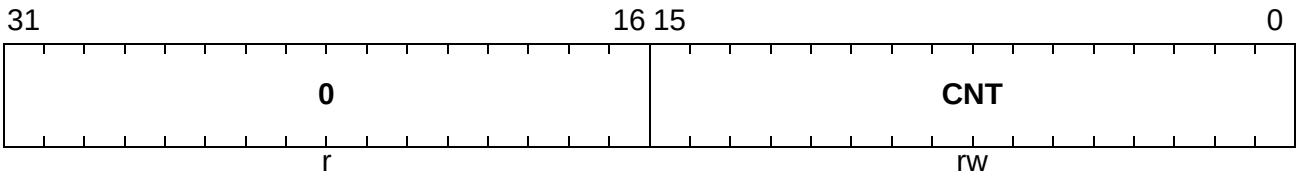
Field	Bits	Type	Description
MTI	[15:0]	rw	**Microtick Value** Number of output pulses to be generated within one input signal period.
0	[31:16]	r	**Reserved**; read as 0; should be written with 0.

PLLSTP
Phase Locked Loop Step Register **Reset Value: 0000 0000$_H$**



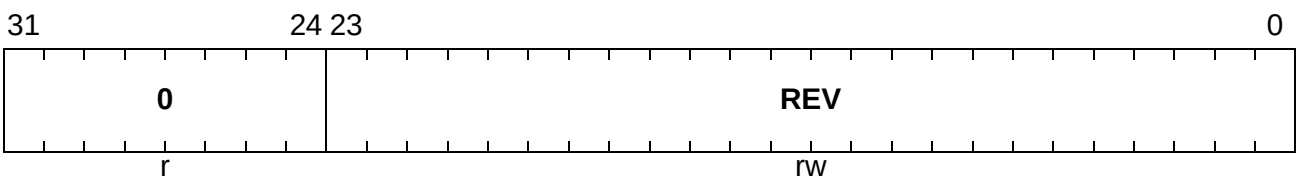
Field	Bits	Type	Description
STP	[15:0]	rw	**Step Value** Number of output pulses to be generated within one input signal period (2-complement data format).
0	[31:16]	r	**Reserved**; read as 0; should be written with 0.

PLLCNT
Phase Locked Loop Counter Register **Reset Value: 0000 0000$_H$**



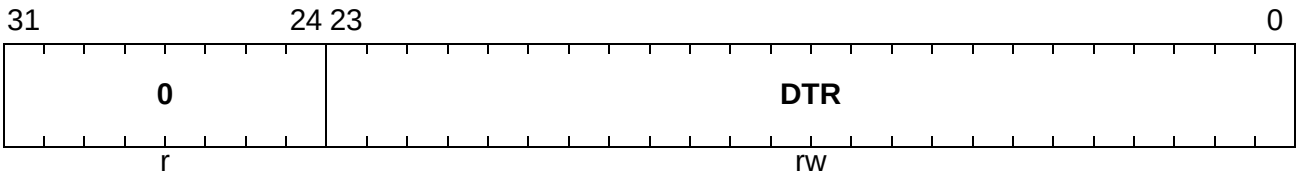
Field	Bits	Type	Description
CNT	[15:0]	rw	**Pulse Counter** Counter for the number of remaining output pulses to be generated.
0	[31:16]	r	**Reserved**; read as 0; should be written with 0.

PLLREV
Phase Locked Loop Reload Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
REV	[23:0]	rw	**Reload Value** Reload value calculated by a subtraction of the number of output pulses to be generated within one input signal period from the input signal's period length (measured in number of GPTA clocks).
0	[31:24]	r	**Reserved**; read as 0; should be written with 0.

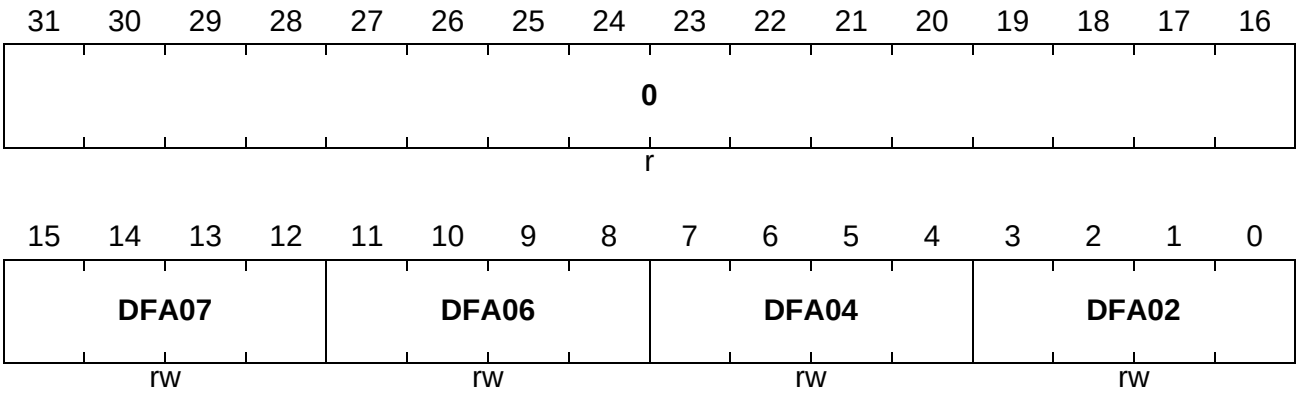
PLLDTR
Phase Locked Loop Delta Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
DTR	[23:0]	rw	**Delta Register Value** Internal register used to store intermediate results for output pulse generation. **Do not write to while PLL is running!**
0	[31:24]	r	**Reserved**; read as 0; should be written with 0.

7.2.5 Clock Bus Register

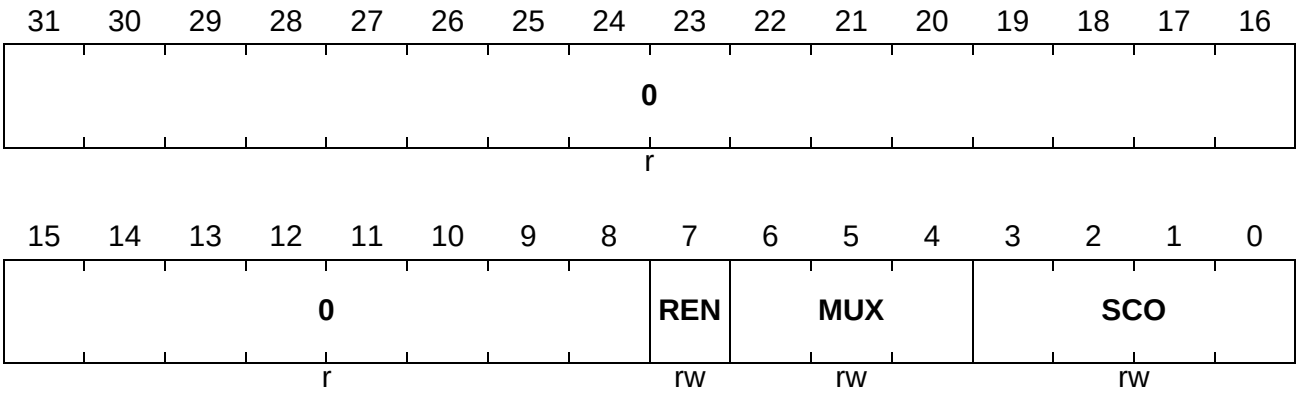
CKBCTR
Clock Bus Control Register **Reset Value: 0000 FFFF$_H$**



Field	Bits	Type	Description
DFA02	[3:0]	rw	**Clock Line 2 Driving Source Selection** CLK2 is provided with the GPTA module clock f_{GPTA} divided by 2^{DFA02} ($0 \le DFA02 \le 14$; DFA02 = 15 selects a different driving source). CLK2 is driven by DCM3 output
DFA04	[7:4]	rw	**Clock Line 4 Driving Source Selection** CLK4 is provided with the GPTA module clock f_{GPTA} divided by 2^{DFA04} ($0 \le DFA04 \le 14$; DFA04 = 15 selects a different driving source). CLK4 is driven by DCM1 output
DFA06	[11-8]	rw	**Clock Line 6 Driving Source Selection** CLK6 is provided with the GPTA module clock f_{GPTA} divided by 2^{DFA06} ($0 \le DFA06 \le 14$) DFA06 = 15 selects a different driving source). Clock6 is driven by FPC1 output
DFA07	[15:12]	rw	**Clock Line 7 Driving Source Selection** CLK7 is provided with the GPTA module clock f_{GPTA} divided by 2^{DFA07} ($0 \le DFA07 \le 14$; DFA07 = 15 selects a different driving source). CLK7 is driven by FPC4 output
0	[31:16]	r	**Reserved**; read as 0; should be written with 0.

7.2.6 Global Timer Register

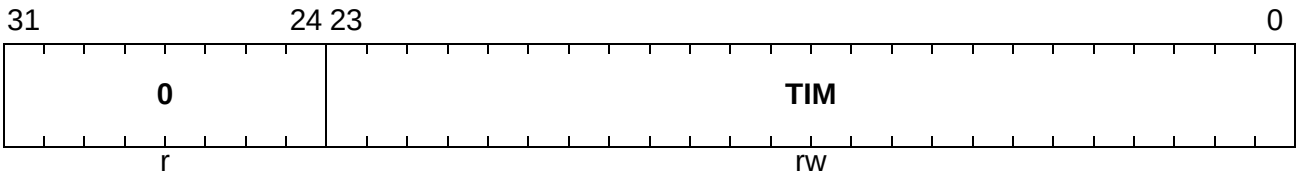
GTCTRm (m = 0, 1)
Global Timer Control Register m **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
SCO	[3:0]	rw	**TGE Flag Source Selection** The **9th bit** of the operation result “GTm timer value - data bus value” is used as the TGE flag. (SCO = 0000_B) The **25th bit** of the subtraction result (sign bit) is addressed to be used as the TGE flag. (SCO = 1111_B)
MUX	[6:4]	rw	**Timer Clock Selection** One of eight available clock bus lines is selected as the timer GTm clock.
REN	7	rw	**Interrupt Request Enable** 0 The interrupt request is disabled 1 An interrupt request is generated when timer GTm overflows
0	[31:8]	r	**Reserved**; read as 0; should be written with 0.

GTTIMm (m = 0, 1)
Global Timer Register

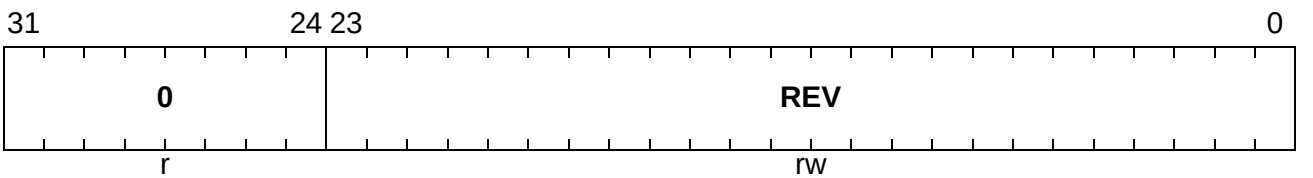
Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
TIM	[23:0]	rw	**Timer Value of Global Timer m**
0	[31:24]	r	**Reserved**; read as 0; should be written with 0.

GTREVm (m = 0, 1)
Global Timer Reload Value Register m

Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
REV	[23:0]	rw	**Reload Value of Global Timer m** Reload value for timer GTm after an overflow
0	[31:24]	r	**Reserved**; read as 0; should be written with 0.

7.2.7 Global Timer Cell Register

Shaded bits represent differences between Capture and Compare Modes.

GTCCTRk (k = 00-31)
Global Timer Cell Control Register k (in Capture Mode) Reset Value: 0000 0000$_H$

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OUT	OIA	OCM			CEN	0		PMU		FED	RED	REN	OSM	MOD	
r	rw	rw			r	r		rw		rw	rw	rw	rw	rw	

GTCCTRk (k = 00-31)
Global Timer Cell Control Register k (in Compare Mode) Reset Value: 0000 0000$_H$

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

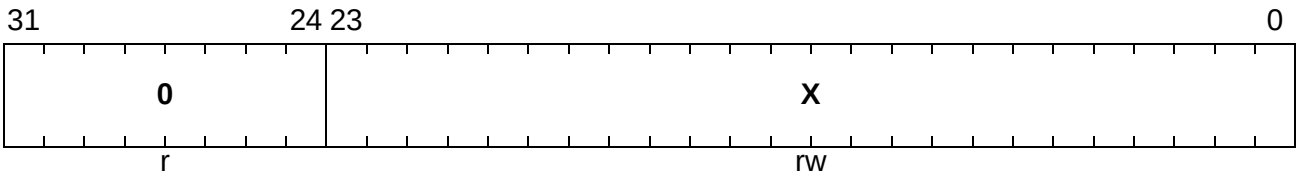
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OUT	OIA	OCM			CEN	0		NE	CAT	CAC	GES	REN	OSM	MOD	
r	rw	rw			r	r		rw	rw	rw	rw	rw	rw	rw	

Field	Bits	Type	Description
MOD	[1:0]	rw	**Mode Control Bits** 00 GTCk operates in Capture Mode hooked to GT0 01 GTCk operates in Capture Mode hooked to GT1 10 GTCk operates in Compare Mode hooked to GT0 11 GTCk operates in Compare Mode hooked to GT1
OSM	2	rw	**One Shot Mode Enable** 0 GTCk is continuously enabled 1 GTCk is enabled for one event only

Field	Bits	Type	Description
REN	3	rw	**Interrupt Request Enable** 0 The interrupt request is disabled 1 An interrupt request is generated when a capture or compare event has occurred
RED	4	rw	**Capture Mode: Input Rising Edge Select** 0 Capture event is not triggered by a rising edge 1 Capture event is triggered by a rising edge on the associated port pin line
GES			**Compare Mode: Greater Equal Select** 0 An “equal” compare is selected 1 A “greater equal” compare is required
FED	5	rw	**Capture Mode: Input Falling Edge Select** 0 Capture event is not triggered by a falling edge 1 Capture event is triggered by a falling edge on the associated port pin line
CAC			**Compare Mode: Capture after Compare Select** 0 Capture after compare is disabled 1 After a compare event, the contents of the associated global timer selected by control register bit field MOD or (depending on control bit CAT) the contents of the alternate global timer are copied to the capture/compare register
PMU	[7:6]	rw	**Capture Mode: External Input Select** One of four available external input lines is selected as trigger source for the capture event. 00 Input line 0 is selected as event trigger line 01 Input line 1 is selected as event trigger line 10 Input line 2 is selected as event trigger line 11 Input line 3 is selected as event trigger line
CAT	6	rw	**Compare Mode: Capture Alternate Timer** 0 The global timer selected by bit field MOD is captured if enabled by control bit CAC = 1 1 The alternate global timer is captured
NE	7	rw	**Compare Mode: Not Effective** Reserved
CEN	10	r	**Cell Enable** 0 GTCk is currently disabled 1 GTCk is currently enabled

Field	Bits	Type	Description
OCM	[13:11]	rw	**Output Control Mode Select** $X00_B$ Current state of GTCk data output line is hold $X01_B$ Current state of GTCk data output line is toggled $X10_B$ GTCk data output line is forced with 0 $X11_B$ GTCk data output line is forced with 1 $0XX_B$ Data output line state is set by an internal GTCk event only $1XX_B$ Data output line state is affected by an internal GTCk event and/or by an operation occurred in an adjacent GTCk + 1 (reported by M1I, M0I interface lines)
OIA	14	rw	**Output Immediate Action** 0 No immediate action required 1 Action defined by bit field OCM must be performed immediately Reading bit OIA returns always 0
OUT	15	r	**Output State** 0 GTCk data output line is 0 1 GTCk data output line is 1
0	[9:8,] [31:16]	r	**Reserved**; read as 0; should be written with 0.

GTCXRk (k = 00-31)
Global Timer Cell X Register k **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
X	[23:0]	rw	**Capture/Compare Register Contents of GTCk**
0	[31:24]	r	**Reserved**; read as 0; should be written with 0.

Note: GTCXRk is write protected when control bits CAC and OSM are set to 1 (“capture after compare” in Single-shot Mode). Write protection is activated, when the value of the selected GT timer matches and/or exceeds the capture/compare register contents. Write protection is released after a software access to register GTCXRk.

7.2.8 Local Timer Cell Register

Shaded bits represent differences between Timer Mode, Capture Mode, and Compare Mode

LTCCTRk (k = 00-63)
Local Timer Cell Control Register k (in Timer Mode) **Reset Value: 0000 0000$_H$**

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OUT	OIA	OCM			CEN	CUD	PMU			FED	RED	REN	OSM	MOD	
r	rw	rw			r	rw	rw			rw	rw	rw	rw	rw	

LTCCTRk (k = 00-63)
Local Timer Control Register k (in Capture Mode) **Reset Value: 0000 0000$_H$**

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OUT	OIA	OCM			CEN	SLL	PMU			FED	RED	REN	OSM	MOD	
r	rw	rw			r	r	rw			rw	rw	rw	rw	rw	

LTCCTRk (k = 00-63)
Local Timer Control Register (in Compare Mode) **Reset Value: 0000 0000$_H$**

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OUT	OIA	OCM			CEN	SLL	PMU			SOH	SOL	REN	OSM	MOD	
r	rw	rw			r	r	rw			rw	rw	rw	rw	rw	

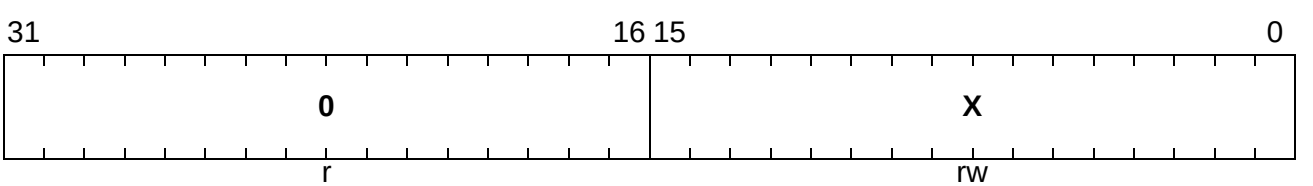
Field	Bits	Type	Description
MOD	[1:0]	rw	**Mode Control bits** 00 LTCk operates in Capture Mode 01 LTCk operates in Compare Mode 10 LTCk operates in free running Timer Mode 11 LTCk operates in reset Timer Mode
OSM	2	rw	**One Shot Mode Enable** 0 LTCk is continuously enabled 1 LTCk is enabled for one event only
REN	3	rw	**Request Enable** 0 The interrupt request is disabled 1 An interrupt request is generated when a - capture event has occurred - compare event has occurred - timer overflow has happened depending on the operation mode selected by bit field MOD
RED	4	rw	**Timer Mode: Input Rising Edge Select** 0 Timer is not affected by a rising edge 1 Timer is updated by a rising edge on the trigger line selected by control bit field PMU
RED			**Capture Mode: Input Rising Edge Select** 0 Capture operation is not affected by a rising edge 1 Capture operation is triggered by a rising edge on the input line selected by control bit field PMU
SOL			**Compare Mode: Select Output Low** 0 Compare deactivated or on high level 1 Compare operation is enabled by a low level on select line input SI[1)]

Field	Bits	Type	Description
FED	5	rw	**Timer Mode: Input Falling Edge Select** 0 Timer is not affected by a falling edge 1 Timer is updated on a falling edge of the trigger line selected by control bit field PMU
FED			**Capture Mode: Input Falling Edge Select** 0 Capture operation is not affected by a falling edge 1 Capture operation is triggered by a falling edge on the input line selected by control bit field PMU
SOH			**Compare Mode: Select Output High** 0 Compare is deactivated or on low level 1 Compare operation is enabled by a high level on select line input SI[1)]
PMU	[8:6]	rw	**Timer Mode: Input Line Multiplexer** **Capture Mode: Input Line Multiplexer** $X00_B$ Input channel 0 is selected as event trigger line $X01_B$ Input channel 1 is selected as event trigger line $X10_B$ Input channel 2 is selected as event trigger line $X11_B$ Input channel 3 is selected as event trigger line $0XX_B$ A port pin is required as trigger source $1XX_B$ A line from the internal clock bus is used as trigger source
CUD	9	rw	**Timer Reset Mode: Coherent Update Enable** 0 Select line output SO is not toggled on timer reset overflow 1 Select line output SO is toggled on timer reset overflow *Note: CUD is automatically cleared after LTCk reset event. Reading bit CUD returns always 0.*
SLL		r	**Capture Mode: Select Line Level** **Compare Mode: Select Line Level** 0 Current state of select line input SI is 0 1 Current state of select line input SI is 1
CEN	10	r	**Cell Enable** 0 LTCk is currently disabled 1 LTCk is currently enabled

Field	Bits	Type	Description
OCM	[13:11]	rw	**Output Control Mode Select** $X00_B$ Current LTCk data output line state is hold $X01_B$ Current LTCk data output line state is toggled $X10_B$ LTCk data output line is forced with 0 $X11_B$ LTCk data output line is forced with 0 $0XX_B$ Data output line state is set by an internal LTCk event only $1XX_B$ Data output line state is affected by an internal LTCk event and/or by an operation occurred in an adjacent $LTCk_{-n}$ cell (reported by M1I, M0I interface lines)
OIA	14	rw	**Output Immediate Action** 0 No immediate action required 1 Action defined by bit field OCM must be performed immediately Reading bit OIA returns always 0
OUT	15	r	**Output State** 0 LTCk data output line is 0 1 LTCk data output line is 1
0	[31:16]	r	**Reserved**; read as 0; should be written with 0.

1) To enable Compare Mode in all cases, SOL and SOH bits must be set to 1.

LTCXRk (k = 00-63)
Local Timer Cell X Register k **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
X	[15:0]	rw	**Local Timer Data Register Value**
0	[31:16]	r	**Reserved**; read a 0; should be written with 0.

7.2.9 Output Port Line Select Register

One GTC and two LTCs share a common GPTA output port line hooked to an external pin. Bit field OMXk controls a multiplexer to select the required signal source for port line k.

OMR0
Output Port Line Multiplex Register 0 **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OMX15		OMX14		OMX13		OMX12		OMX11		OMX10		OMX9		OMX8	
rw		rw		rw		rw		rw		rw		rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OMX7		OMX6		OMX5		OMX4		OMX3		OMX2		OMX1		OMX0	
rw		rw		rw		rw		rw		rw		rw		rw	

OMR1
Output Port Line Multiplex Register 1 **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OMX31		OMX30		OMX29		OMX28		OMX27		OMX26		OMX25		OMX24	
rw		rw		rw		rw		rw		rw		rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OMX23		OMX22		OMX21		OMX20		OMX19		OMX18		OMX17		OMX16	
rw		rw		rw		rw		rw		rw		rw		rw	

OMR2
Output Port Line Multiplex Register 2 **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OMX47		OMX46		OMX45		OMX44		OMX43		OMX42		OMX41		OMX40	
rw		rw		rw		rw		rw		rw		rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OMX39		OMX38		OMX37		OMX36		OMX35		OMX34		OMX33		OMX32	
rw		rw		rw		rw		rw		rw		rw		rw	

OMR3
Output Port Line Multiplex Register 3

Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OMX63		OMX62		OMX61		OMX60		OMX59		OMX58		OMX57		OMX56	
rw		rw		rw		rw		rw		rw		rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OMX55		OMX54		OMX53		OMX52		OMX51		OMX50		OMX49		OMX48	
rw		rw		rw		rw		rw		rw		rw		rw	

Field	Bits	Type	Description
OMXk (k = 0-15)	[31:0]	rw	**Output Multiplexer Control Fields 0-15** 00 Reserved 01 Port line k is driven by GTCk output 10 Port line k is driven by LTCk output 11 Port line k is driven by LTCk + 32 output
OMXk (k = 16-31)	[31:0]	rw	**Output Multiplexer Control Fields 16-31** 00 Reserved 01 Line k is driven by GTCk output 10 Port line k is driven by LTCk output 11 Port line k is driven by LTCk + 32 output
OMXk (k = 32-47)	[31:0]	rw	**Output Multiplexer Control Fields 32-47** 00 Reserved 01 Port line k is driven by GTCk + 32 output 10 Port line k is driven by LTCk output 11 Port line k is driven by LTCk + 32 output
OMXk (k = 48-63)	[31:0]	rw	**Output Multiplexer Control Fields 48-63** 00 Reserved 01 Port line k is driven by GTCk + 32 output 10 Port line k is driven by LTCk output 11 Port line k is driven by LTCk + 32 output

A low state detected on the emergency input line being connected to a specific external port pin pulls down all “alternate pin function select lines” enabled by associated bits in the emergency control register. The “alternate pin function select lines” may be used to change the function of a port pin from GPTA output to General Purpose I/O Register output.

EMGCTR0
Emergency Control Register 0 **Reset Value: 0000 0000$_H$**

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PEN 15	PEN 14	PEN 13	PEN 12	PEN 11	PEN 10	PEN 09	PEN 08	PEN 07	PEN 06	PEN 05	PEN 04	PEN 03	PEN 02	PEN 01	PEN 00
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

EMGCTR1
Emergency Control Register 1 **Reset Value: 0000 0000$_H$**

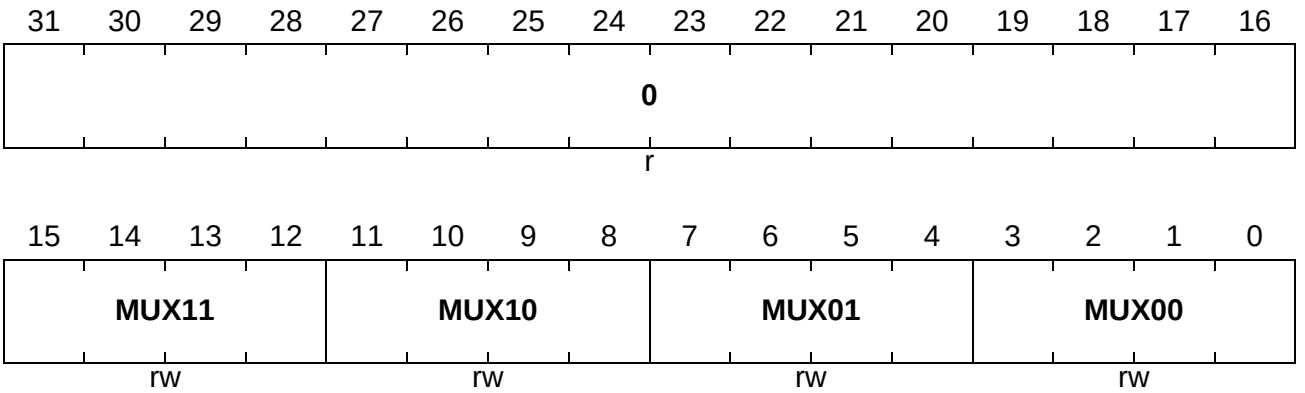
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	PEN 62	PEN 61	PEN 60	PEN 59	PEN 58	PEN 57	PEN 56	PEN 55	PEN 54	PEN 53	PEN 52	PEN 51	PEN 50	PEN 49	PEN 48
r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PEN 47	PEN 46	PEN 45	PEN 44	PEN 43	PEN 42	PEN 41	PEN 40	PEN 39	PEN 38	PEN 37	PEN 36	PEN 35	PEN 34	PEN 33	PEN 32
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
PENk (k = 00-31)	k	rw	**Emergency Control Bits 31-0** 0 The port pin driven by port line k is disconnected from emergency output function 1 The port pin driven by port line k is enabled for emergency output function
PENk (k = 32-62)	k	rw	**Emergency Control Bits 62-32** 0 The port pin driven by port line k is disconnected from emergency output function 1 The port pin driven by port line k is enabled for emergency output function
0	31	r	**Reserved**; read as 0; should be written with 0.

7.2.10 ADC Connections Control Register

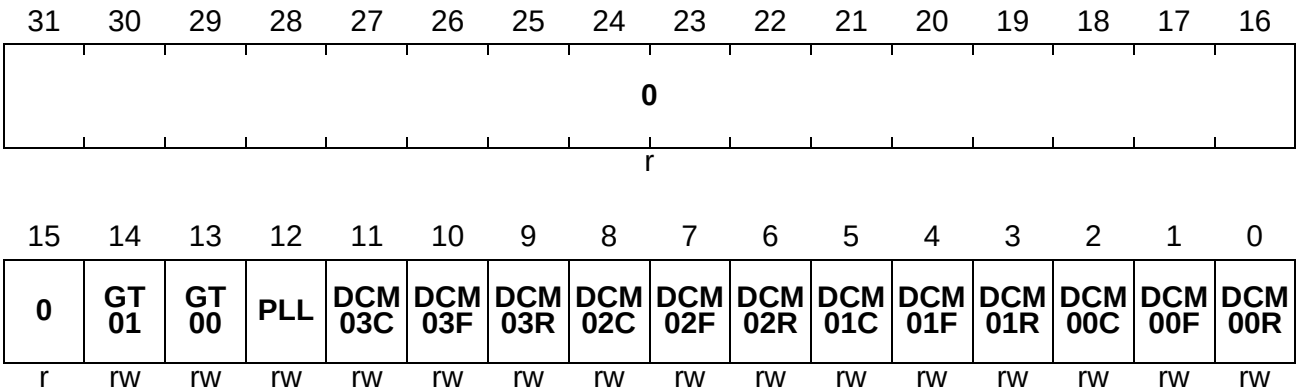
ADCCTR
ADC Multiplex Control Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
MUX00	[3:0]	rw	**ADC0 Trigger Signal 0 Source Selection** Defines the trigger source for the PTIN00 AD conversion start signal to AD converter 0 (bit field definition see **Table 7-5**).
MUX01	[7:4]	rw	**ADC0 Trigger Signal 1 Source Selection** Defines the trigger source for the PTIN01 AD conversion start signal to AD converter 0 (bit field definition see **Table 7-5**).
MUX10	[11:8]	rw	**ADC1 Trigger Signal 0 Source Selection** Defines the trigger source for the PTIN10 AD conversion start signal to AD converter 1 (bit field definition see **Table 7-5**).
MUX11	[15:12]	rw	**ADC1 Trigger Signal 1 Source Selection** Defines the trigger source for the PTIN11 AD conversion start signal to AD converter 1 (bit field definition see **Table 7-5**).
0	[31:16]	r	**Reserved**; read as 0; should be written with 0.

7.2.11 Service Request State Register

SRS0
Service Request State Register 0 **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
DCM00R **DCM01R** **DCM02R** **DCM03R**	0 3 6 9	rw	**Rising Edge Service Request State for DCMk**[1)] 0 No interrupt service is requested 1 Interrupt service is requested due to a rising edge detected on DCMk input signal line
DCM00F **DCM01F** **DCM02F** **DCM03F**	1 4 7 10	rw	**Falling Edge Service Request State for DCMk**[1)] 0 No interrupt service is requested 1 Interrupt service is requested due to a falling edge detected on DCMk input signal line
DCM00C **DCM01C** **DCM02C** **DCM03C**	2 5 8 11	rw	**Compare Service Request State for DCMk**[1)] 0 No interrupt service is requested 1 Interrupt service is requested due to a compare event occurred in DCMk cell
PLL	12	rw	**Counter Service Request State for PLL** 0 No interrupt service is requested 1 Interrupt service is requested because the counter for the number remaining output pulses decremented to 0
GT00	13	rw	**Timer Service Request State for GT0** 0 No interrupt service is requested 1 Interrupt service is requested due to a timer overflow

1) k = 3-0; k = 0 refers to DCM00R, DCM00P, or DCM00C;
k = 1 refers to DCM01R, DCM01P, or DCM01C;
k = 2 refers to DCM02R, DCM02P, or DCM02C;
k = 3 refers to DCM03R, DCM03P, or DCM03C.

Field	Bits	Type	Description
GT01	14	rw	**Timer Service Request State for GT1** 0 No interrupt service is requested 1 Interrupt service is requested due to a timer overflow
0	[31:15]	r	**Reserved**; read as 0; should be written with 0.

SRS1
Service Request State Register 1 **Reset Value: 0000 0000$_H$**

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GTC 15	**GTC 14**	**GTC 13**	**GTC 12**	**GTC 11**	**GTC 10**	**GTC 09**	**GTC 08**	**GTC 07**	**GTC 06**	**GTC 05**	**GTC 04**	**GTC 03**	**GTC 02**	**GTC 01**	**GTC 00**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
GTCk (k = 00-31)	k	rw	**Capt./Comp. Service Request State for GTCk** 0 No interrupt service is requested 1 Interrupt service is requested due to a capture or compare event occurred in GTCk

SRS2
Service Request State Register 2 **Reset Value: 0000 0000$_H$**

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LTC 15	LTC 14	LTC 13	LTC 12	LTC 11	LTC 10	LTC 09	LTC 08	LTC 07	LTC 06	LTC 05	LTC 04	LTC 03	LTC 02	LTC 01	LTC 00
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

SRS3
Service Request State Register 3 **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LTC 63	LTC 62	LTC 61	LTC 60	LTC 59	LTC 58	LTC 57	LTC 56	LTC 55	LTC 54	LTC 53	LTC 52	LTC 51	LTC 50	LTC 49	LTC 48
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LTC 47	LTC 46	LTC 45	LTC 44	LTC 43	LTC 42	LTC 41	LTC 40	LTC 39	LTC 38	LTC 37	LTC 36	LTC 35	LTC 34	LTC 33	LTC 32
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
LTCk (k = 00-31)	k	rw	**Timer/Capture/Compare Service Request State for LTCk** 0 No interrupt service is requested 1 Interrupt service is requested due to a timer overflow, capture that or compare event that occurred in LTCk
LTCk (k = 32-63)	k	rw	**Timer/Capture/Compare Service Request State for LTCk** 0 No interrupt service is requested 1 Interrupt service is requested due to a timer overflow, capture, or compare event that occurred in LTCk

7.3 GPTA Module Implementation

This section describes the GPTA Module interfaces with the clock control, port connections, interrupt control, and address decoding.

7.3.1 Interfaces of the GPTA Module

Figure 7-49 shows the TC1775 specific implementation details and interconnections of the GPTA Module. The GPTA Module has 64 input lines and 64 output lines that are connected with Port 8, Port 9, Port 10, and Port 11. Additionally, the GPTA Module is supplied by separate clock control, interrupt control, and address decoding logic.

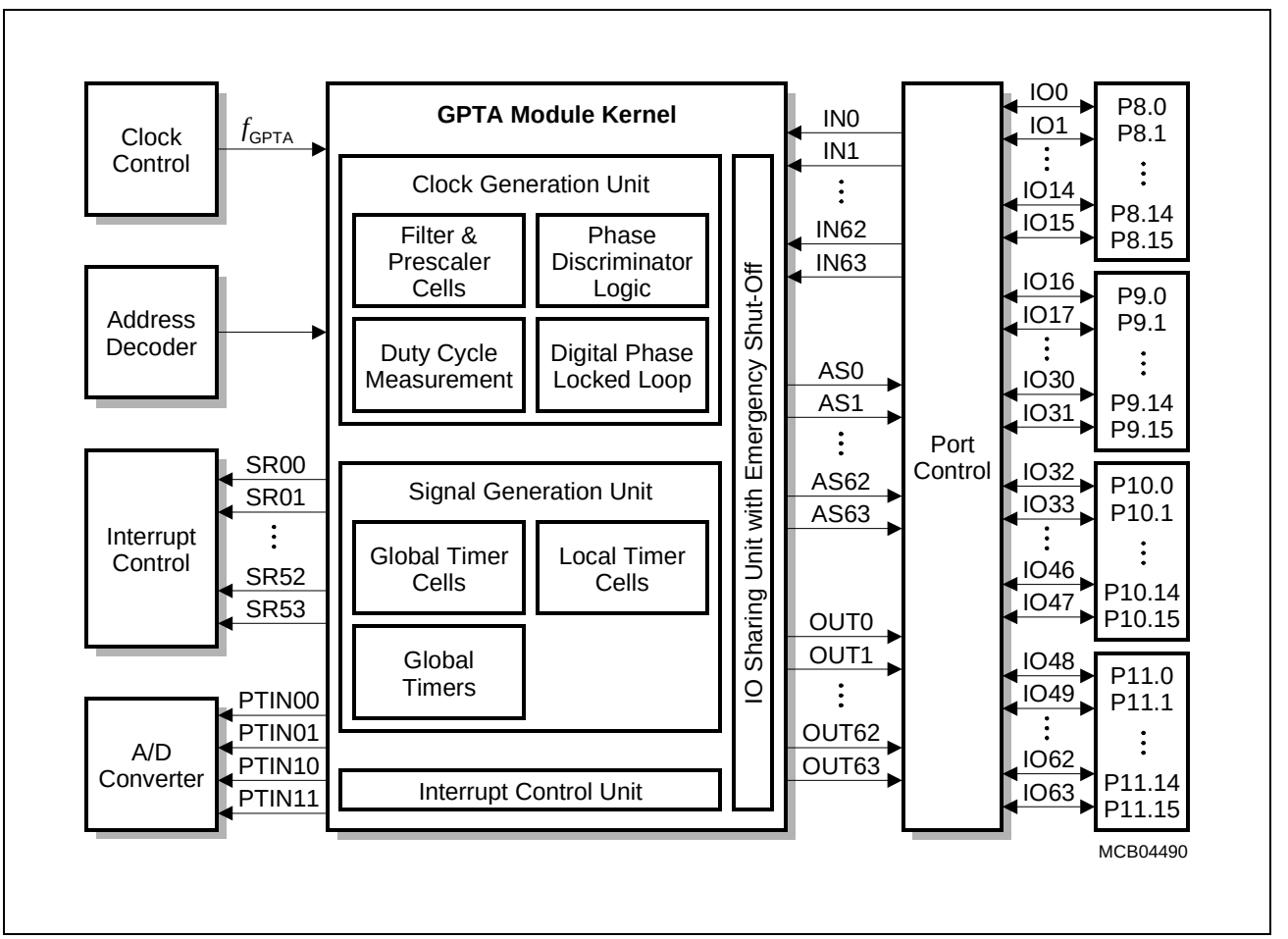


Figure 7-49 GPTA Block Diagram for TC1775

7.3.2 External GPTA Module Registers

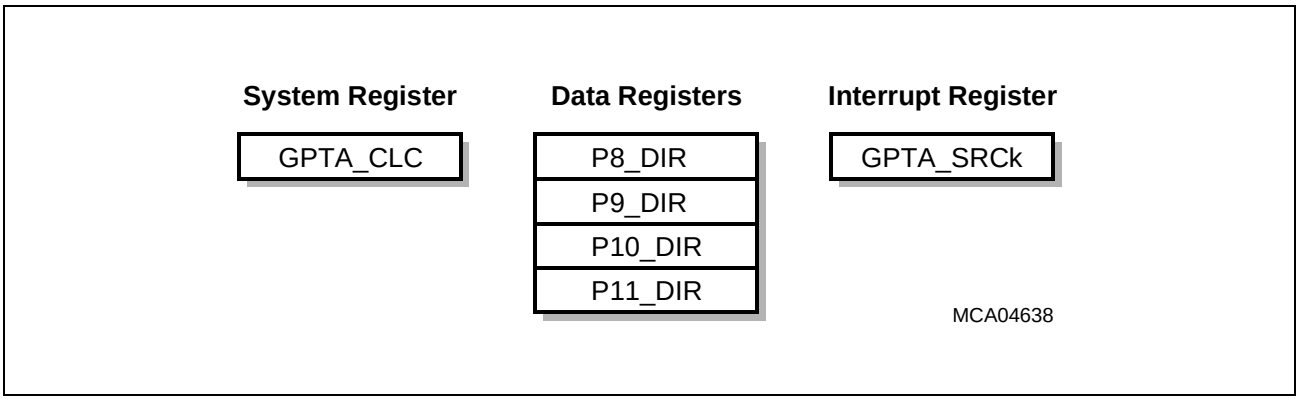
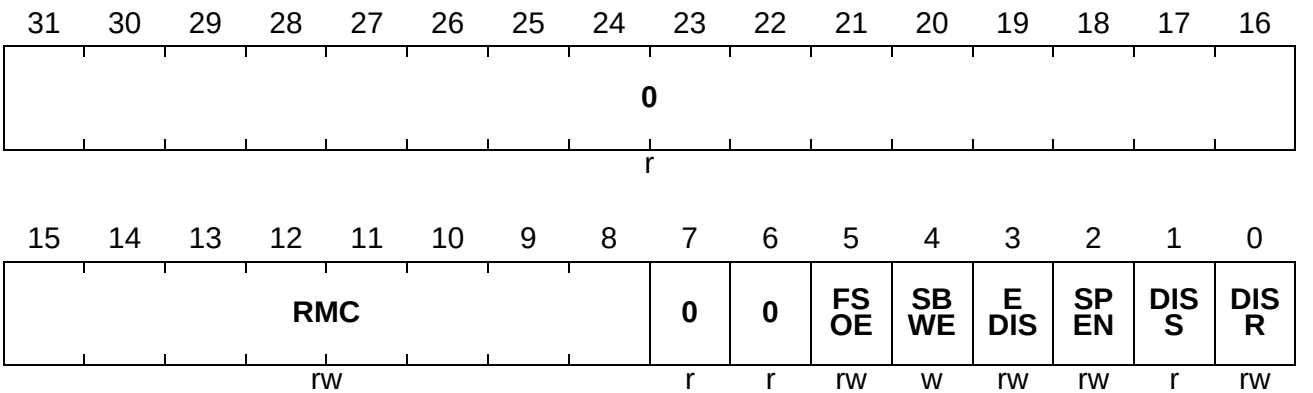


Figure 7-50 GPTA Implementation Specific Special Function Registers

7.3.2.1 Clock Control Register

The clock control register allows the programmer to adapt the functionality and power consumption of the GPTA Module to the requirements of the application. The diagram below shows the clock control register functionality as is implemented in the TC1775 for the GPTA Module.

GPTA_CLC
GPTA Clock Control Register **Reset Value: 0000 0002$_H$**



Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control of the module.
DISS	1	r	**Module Disable Status Bit** Bit indicates the current status of the module.
SPEN	2	rw	**Module Suspend Enable for OCDS** Used for enabling the suspend mode.
EDIS	3	rw	**External Request Disable** Used for controlling the external clock disable request.
SBWE	4	w	**Module Suspend Bit Write Enable for OCDS** Defines whether SPEN and FSOE are write protected.
FSOE	5	rw	**Fast Switch Off Enable** Used for fast clock switch off in OCDS suspend mode.
RMC	[15:8]	rw	**8-Bit Clock Divider Value in RUN Mode**
0	7, 6, [31:16]	r	**Reserved**; returns 0 if read; should be written with 0.

Note: After a hardware reset operation the GPTA Module is disabled.

7.3.2.2 Port Control Registers

The GPTA Module input and output lines provided by the I/O Sharing Unit are connected to four 16-bit TC1775 ports: Port 8, Port 9, Port 10, and Port 11. The port line assignments are shown in **Table 7-10**.

Table 7-10 GPTA Port Line Assignment

Port	Assigned GPTA I/O lines	Port	Assigned GPTA I/O lines
P8.0	IN0 / OUT0 / AS0	P10.0	IN32 / OUT32 / AS32
P8.1	IN1 / OUT1 / AS1	P10.1	IN33 / OUT33 / AS33
P8.2	IN2 / OUT2 / AS2	P10.2	IN34 / OUT34 / AS34
P8.3	IN3 / OUT3 / AS3	P10.3	IN35 / OUT35 / AS35
P8.4	IN4 / OUT4 / AS4	P10.4	IN36 / OUT36 / AS36
P8.5	IN5 / OUT5 / AS5	P10.5	IN37 / OUT37 / AS37
P8.6	IN6 / OUT6 / AS6	P10.6	IN38 / OUT38 / AS38
P8.7	IN7 / OUT7 / AS7	P10.7	IN39 / OUT39 / AS39
P8.8	IN8 / OUT8 / AS8	P10.8	IN40 / OUT40 / AS40
P8.9	IN9 / OUT9 / AS9	P10.9	IN41 / OUT41 / AS41
P8.10	IN10 / OUT10 / AS10	P10.10	IN42 / OUT42 / AS42
P8.11	IN11 / OUT11 / AS11	P10.11	IN43 / OUT43 / AS43
P8.12	IN12 / OUT12 / AS12	P10.12	IN44 / OUT44 / AS44
P8.13	IN13 / OUT13 / AS13	P10.13	IN45 / OUT45 / AS45
P8.14	IN14 / OUT14 / AS14	P10.14	IN46 / OUT46 / AS46
P8.15	IN15 / OUT15 / AS15	P10.15	IN47 / OUT47 / AS47
P9.0	IN16 / OUT16 / AS16	P11.0	IN48 / OUT48 / AS48
P9.1	IN17 / OUT17 / AS17	P11.1	IN49 / OUT49 / AS49
P9.2	IN18 / OUT18 / AS18	P11.2	IN50 / OUT50 / AS50
P9.3	IN19 / OUT19 / AS19	P11.3	IN51 / OUT51 / AS51
P9.4	IN20 / OUT20 / AS20	P11.4	IN52 / OUT52 / AS52
P9.5	IN21 / OUT21 / AS21	P11.5	IN53 / OUT53 / AS53
P9.6	IN22 / OUT22 / AS22	P11.6	IN54 / OUT54 / AS54
P9.7	IN23 / OUT23 / AS23	P11.7	IN55 / OUT55 / AS55
P9.8	IN24 / OUT24 / AS24	P11.8	IN56 / OUT56 / AS56
P9.9	IN25 / OUT25 / AS25	P11.9	IN57 / OUT57 / AS57

Table 7-10 GPTA Port Line Assignment (cont'd)

Port	Assigned GPTA I/O lines	Port	Assigned GPTA I/O lines
P9.10	IN26 / OUT26 / AS26	P11.10	IN58 / OUT58 / AS58
P9.11	IN27 / OUT27 / AS27	P11.11	IN59 / OUT59 / AS59
P9.12	IN28 / OUT28 / AS28	P11.12	IN60 / OUT60 / AS60
P9.13	IN29 / OUT29 / AS29	P11.13	IN61 / OUT61 / AS61
P9.14	IN30 / OUT30 / AS30	P11.14	IN62 / OUT62 / AS62
P9.15	IN31 / OUT31 / AS31	P11.15	IN63 / OUT63 / AS63

Alternate Function Select Control

The alternate function, associated with a port line for GPTA I/O purposes, are controlled by the corresponding ASx (x = 00-63) alternate function select output line of the GPTA Module. The contents of the GPTA output multiplex register GPTA_OMRk (k = 0-3) define whether a port pin is used by the GPTA Module for I/O purposes or not.

- Register GPTA_OMR0 controls the GPTA I/O lines for port 8
- Register GPTA_OMR1 controls the GPTA I/O lines for port 9
- Register GPTA_OMR2 controls the GPTA I/O lines for port 10
- Register GPTA_OMR3 controls the GPTA I/O lines for port 11

Direction Register Control

Each port pin to be used as a GPTA input or output must be prepared by programming the corresponding direction register.

The P8_DIR, P9_DIR, P10_DIR, P11_DIR registers configure the direction of port pins required for GPTA Module input and output. The control bit for a port pin used for GPTA I/O must be set to:

– 0 for input function
– 1 for output function

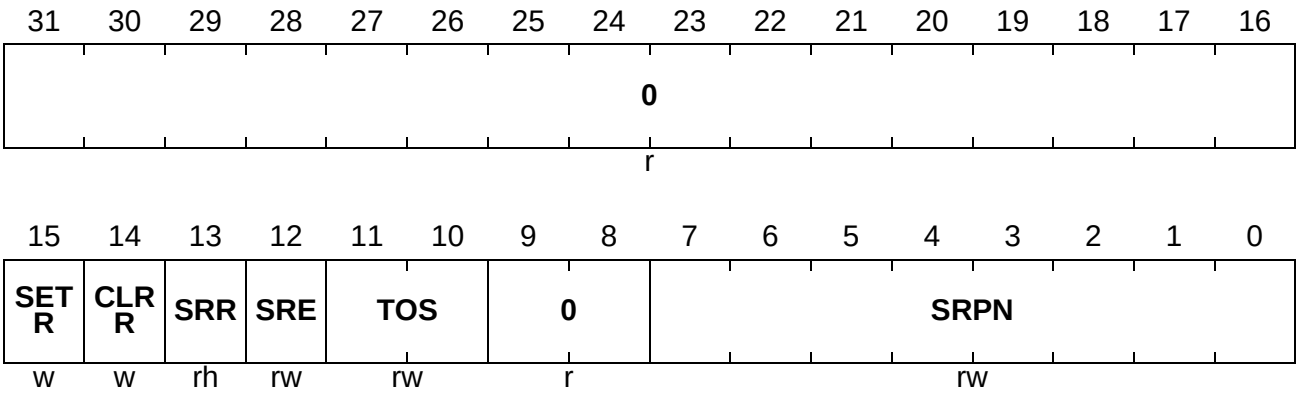
Emergency Input Control

The emergency input signal is connected to P11.15. Therefore, P11.15 must be programmed as input if the emergency input function is required.

7.3.2.3 Interrupt Registers

The 54 interrupt outputs SR00 - SR53 of the GPTA Module are controlled by the service request control registers GPTA_SRC00 to GPTA_SRC53.

GPTA_SRCk (k = 00-53)
GPTA Interrupt Service Request Control Register k **Reset Values: 0000 0000$_H$**



Field	Bits	Type	Description
SRPN	[7:0]	rw	**Service Request Priority Number**
TOS	[11:10]	rw	**Type of Service Control**
SRE	12	rw	**Service Request Enable**
SRR	13	rh	**Service Request Flag**
CLRR	14	w	**Request Clear Bit**
SETR	15	w	**Request Set Bit**
0	[9:8], [31:16]	r	**Reserved;** returns 0 if read; should be written with 0.

Note: Further details on interrupt handling and processing are described in the “Interrupt System” chapter of the TC1775 System Units User’s Manual.

7.3.3 AD Converter Control Outputs

The GPTA Module provides four output lines for AD converter control. These four output lines are assigned in the following way:

- ADC0 is controlled by the PTIN00 and PTIN01 output lines
- ADC1 is controlled by the PTIN10 and PTIN11 output lines

The four GPTA output lines for AD converter control are controlled by the ADC Multiplex Control Register GPTA_ADCCTR. See **Section 7.1.6**.

7.3.4 GPTA Register Address Range

In the TC1775, the registers of the GPTA Module are located in the following address range:

– Module Base Address: F000 1800_H
  Module End Address: F000 $1FFF_H$
– Absolute Register Address = Module Base Address + Offset Address
  (offset addresses see **Table 7-9**)

8 Analog-to-Digital Converter (ADC)

This chapter describes the two ADC Analog-to-Digital converters (ADC0 and ADC1) of the TC1775. This chapter contains the following sections:

- Functional description of the ADC Kernel for ADC0 and ADC1 (see **Section 8.1**)
- Register descriptions for all ADC Kernel specific registers (see **Section 8.2**)
- TC1775 implementation specific details and registers of the ADC0/ADC1 Modules, including port connections and control, interrupt control, address decoding, and clock control (see **Section 8.3**).

*Note: The ADC Kernel register names described in **Section 8.2** will be referenced in the TC1775 User's Manual with the module name prefix "ADC0_" for the ADC0 interface and "ADC1_" for the ADC1 interface.*

8.1 ADC Kernel Description

The ADC provides 8-bit, 10-bit, or 12-bit resolution, including sample & hold functionality. The ADC operates using the method of the successive approximation. A multiplexer selects between up to sixteen analog input channels. Conversion requests are generated either under software control or by hardware. An automatic self-calibration adjusts the ADC Modules to changing temperatures or process variations.

Features

- 8-bit, 10-bit, 12-bit A/D Conversion
- Successive approximation conversion method
- Total Unadjusted Error (TUE) of ±2 LSB @ 10-bit resolution
- Integrated sample and hold functionality
- 16 analog input channels
- Dedicated control and status registers for each analog channel
- Powerful conversion request sources
- Selectable reference voltages for each channel
- Programmable sample and conversion timing schemes
- Limit checking
- Broken wire - short circuit detection
- Flexible service request generation
- Synchronization of two on-chip A/D Converters
- Automatic control of external analog multiplexer
- Equidistant samples initiated by timer
- External trigger inputs for conversion requests
- Two external trigger inputs, connected with an on-chip peripheral
- Power reduction and clock control

Figure 8-1 shows a global view of the ADC Module kernel with the module specific interface connections.

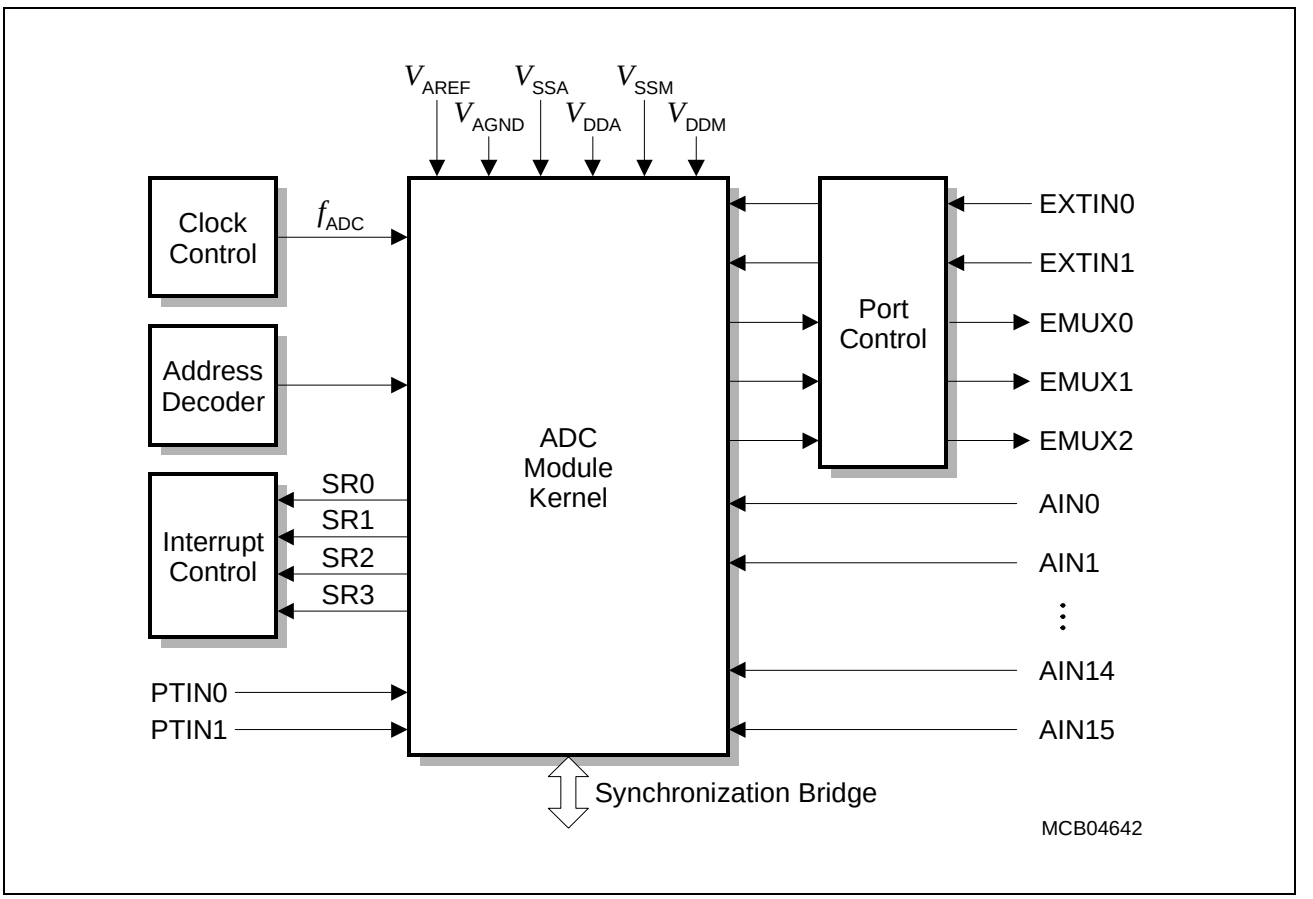


Figure 8-1 General Block Diagram of the ADC Interface

The ADC Module communicates with the external world via five digital I/O lines and sixteen analog inputs. Clock control, address decoding, digital I/O port control, and service request generation is managed outside the ADC Module kernel. Two trigger inputs and a synchronization bridge are used for internal control purposes.

Figure 8-2 shows a more detailed block diagram of the ADC kernel with its main functional units.

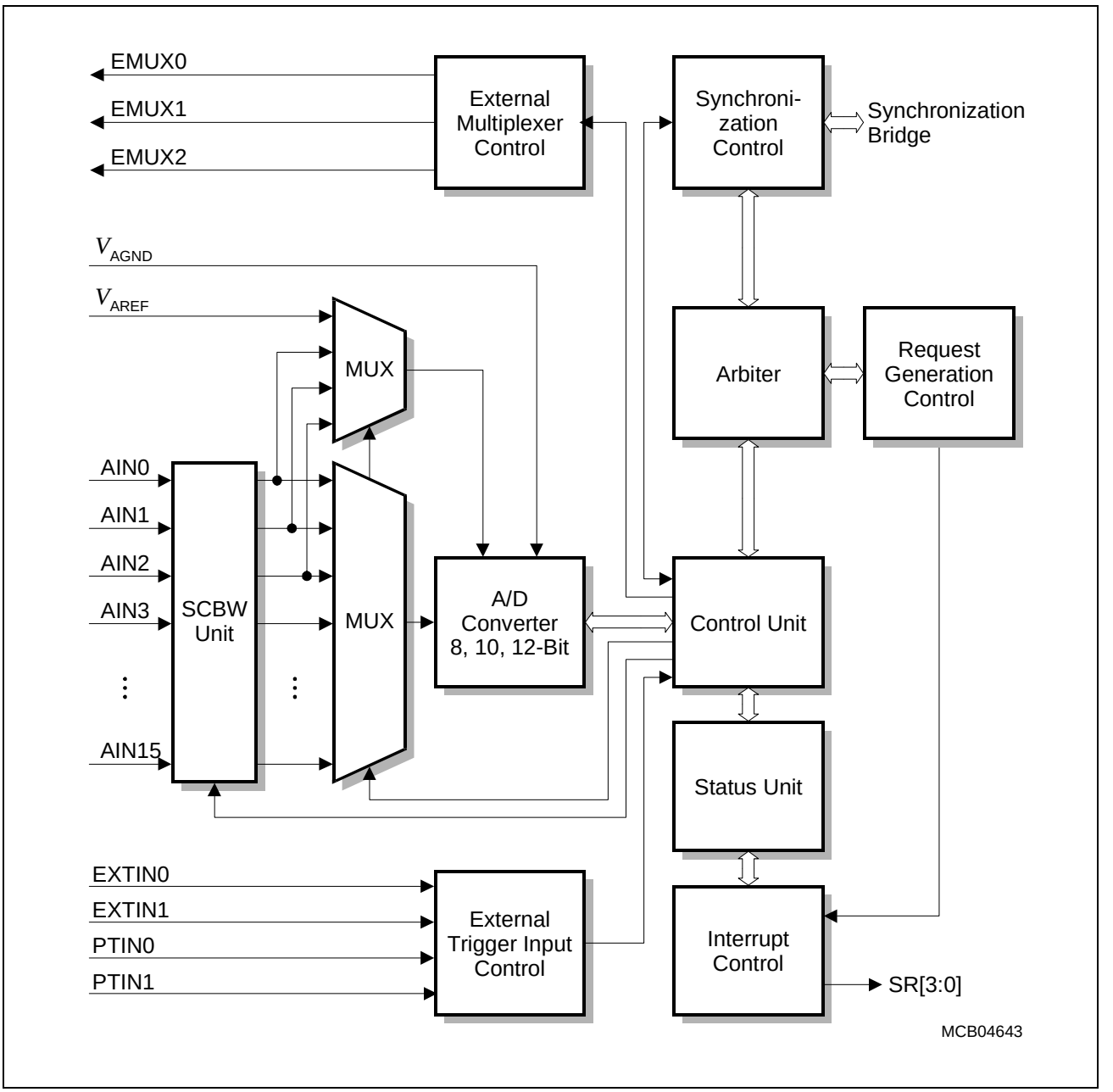


Figure 8-2 Block Diagram of the ADC Kernel

8.1.1 Conversion Request Sources

The ADC Module control logic provides extraordinarily effective methods to request and arbitrate conversions. Conversion requests for one or more analog channels can be triggered by hardware as well as by software to provide maximum flexibility in requesting analog-to-digital conversions. Up to six individual configurable conversion request sources are implemented to issue analog-to-digital conversion requests.

In principle, the conversion request sources can be assigned either to the group of parallel conversion request sources or the group of sequential conversion request sources. A global overview of parallel and sequential conversion request sources and detailed descriptions of each source are provided in the following sections.

8.1.1.1 Parallel Conversion Request Sources

Parallel conversion request sources generate one or more conversion request at a time for an analog channel. **Table 8-1** shows the available parallel conversion request sources including the associated control and status blocks.

Table 8-1 Parallel Conversion Request Sources

Source	Conversion Req. Control Register	Conversion Req. Pending Register	Arbitration Participation Flag	Source Arbitration Level
Timer	TTC	TCRP	AP.TP	SAL.SALT
External Event	EXTC0 EXTC1	EXCRP EXCRP	AP.EXP AP.EXP	SAL.SALEX SAL.SALEX
Software	REQ0	SW0CRP	AP.SW0P	SAL.SALSW0
Auto-Scan	SCN	ASCRP	AP.ASP	SAL.SALAS

A parallel conversion request source consists of a conversion request register, a conversion request pending register, an arbitration participation flag, and the source arbitration level.

Each conversion request register is 16 bits wide and each bit within this register represents an analog channel for which a conversion request can be generated. The contents of the conversion request register are loaded into the conversion request pending register on source specific trigger events. If at least one bit is set in the conversion request pending register, the arbitration participation flag is set for this source. This informs the arbiter to include this parallel conversion request source into arbitration.

If this source is the arbitration winner, a conversion is started for the conversion request within the conversion request register with the highest channel number. Starting a conversion causes the conversion request bit to be reset in the conversion request

pending register by the arbiter. If a currently running conversion initiated by the parallel source is cancelled, the arbiter restores the corresponding conversion request bit in the conversion request pending registers for this channel. If all pending conversion requests are processed, the arbiter resets the arbitration participation flag of this parallel source.

The contents of the conversion request pending register can be reset globally under software control by resetting the arbitration participation flag for this source.

8.1.1.2 Sequential Conversion Request Sources

Sequential conversion request sources generate only one conversion request at a time for an analog channel. The settings of the ADC's resolution and the external multiplexer are derived from the request register of the sequential source. **Table 8-2** shows the available sequential conversion request sources including the associated control and status blocks.

Table 8-2 Sequential Conversion Request Sources

Source	Conversion Request Control Register	Back-Up Register	Arbitration Participation Flag	Source Arbitration Level
Channel Injection	CHIN	not accessible via Bus	AP.CHP	SAL.SALCHIN
Queue	QUEUE0	not accessible via Bus	AP.QP	SAL.SALQ

A sequential conversion request source consists of a conversion request control register, a back-up register, an arbitration participation flag and the source arbitration level.

The request register contains a conversion request bit, the channel number to be converted, control information for external multiplexer settings and control information to select the resolution of the ADC. Setting the conversion request bit causes the arbitration participation flag to be set. This informs the arbiter to include the sequential conversion request source into arbitration. If this sequential source is the arbitration winner, a conversion is started for the analog channel specified within the request register. The settings of the external multiplexer and the resolution of the ADC are also derived from this conversion request control register.

Starting a conversion causes the conversion request bit to be reset by the arbiter. The arbitration participation flag is automatically reset if the conversion request register and the back-up register contain no valid request.

If a currently running conversion initiated by a sequential source is cancelled, the arbiter restores the conversion information in the back-up for this channel. Conversion information means to the conversion request bit, the setting for the external multiplexer,

and the settings of the resolution. If the back-up register contains valid conversion information, the arbiter reads from the back-up register instead from the conversion request control register. Thus, the previously cancelled conversion participates in arbitration again. A new conversion request generated in the meantime via the conversion request register will be performed after the request in the back-up register is served.

The request bit of the request register and the back-up register can be cancelled under software control. Resetting the arbitration participation bit clears either the request bit in the request register (the back-up register contains no request) or the request bit in the back-up register (the back-up register contains a valid request).

8.1.1.3 Conversion Request Source “Timer”

Periodic samples can be achieved by timer generated conversion requests. An individual programmable timer is integrated in the ADC Module to serve as a trigger source. It provides interrupt generation logic as well as the arbitration lock mechanism to ensure periodical sampling without jitter. A block diagram of the timer and its control and status blocks is shown in **Figure 8-3**.

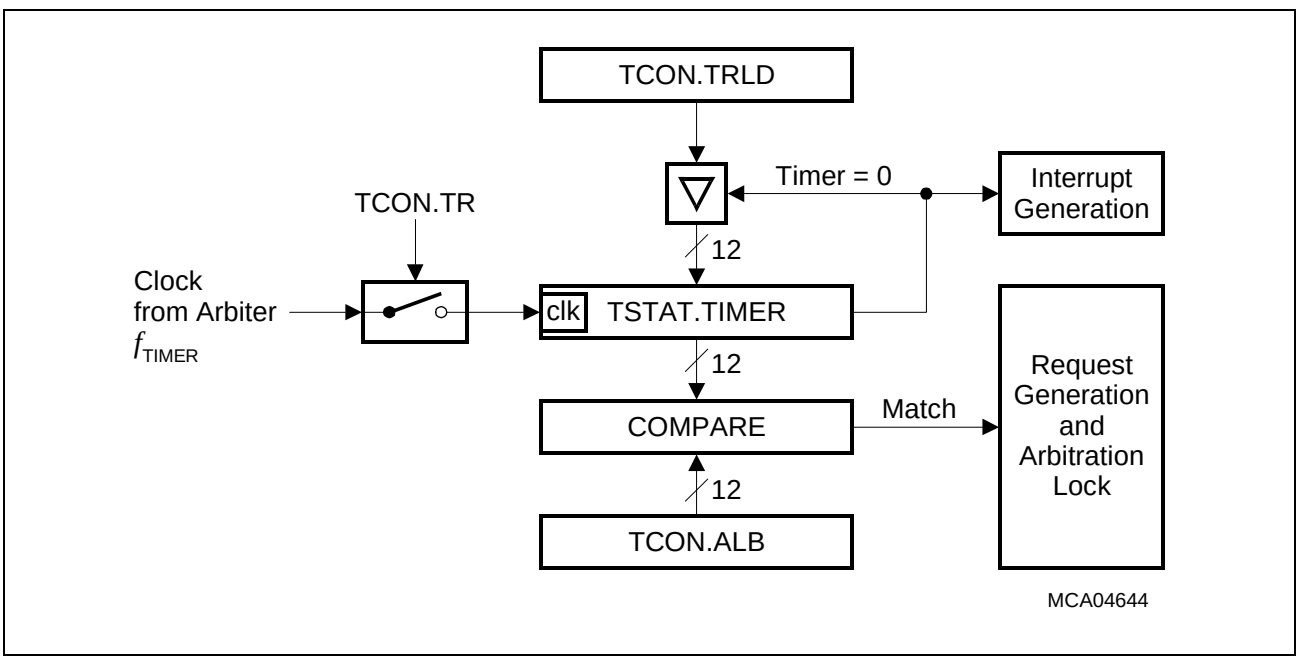


Figure 8-3 Block Diagram of Conversion Request Source “Timer”

Setting the timer run bit clocks the timer with f_{TIMER}, which is derived from the arbiter. This synchronizes the timer on the arbiter for jitter-free sampling. If the timer run bit is set, the timer register bit field STAT.TIMER is loaded with the timer reload value TCON.TRLD. With each clock cycle, the timer register is decremented and compared to the arbitration-lock-boundary value TCON.ALB. If the value of the timer register is equal to or below the value of the arbitration-lock-boundary, the arbitration is locked and the arbitration lock bit STAT.AL is set. This arbitration-lock mechanism can be used to generate samples without being delayed by a currently running conversion. On a timer underflow, the arbitration is unlocked, the timer register is reloaded, the arbitration lock bit is cleared, the timer related service request status flag (MSS1.MSRT) is set, and a trigger pulse is sent to the conversion request source “Timer”.

The timer period $t_{TPERIOD}$ can be specified within the range from microseconds up to milliseconds according to the following equation.

$$t_{TPERIOD} = (TRLD + 1) \times \frac{20}{f_{ADC}}$$

Figure 8-4 shows the control and status blocks of the conversion request source “Timer”.

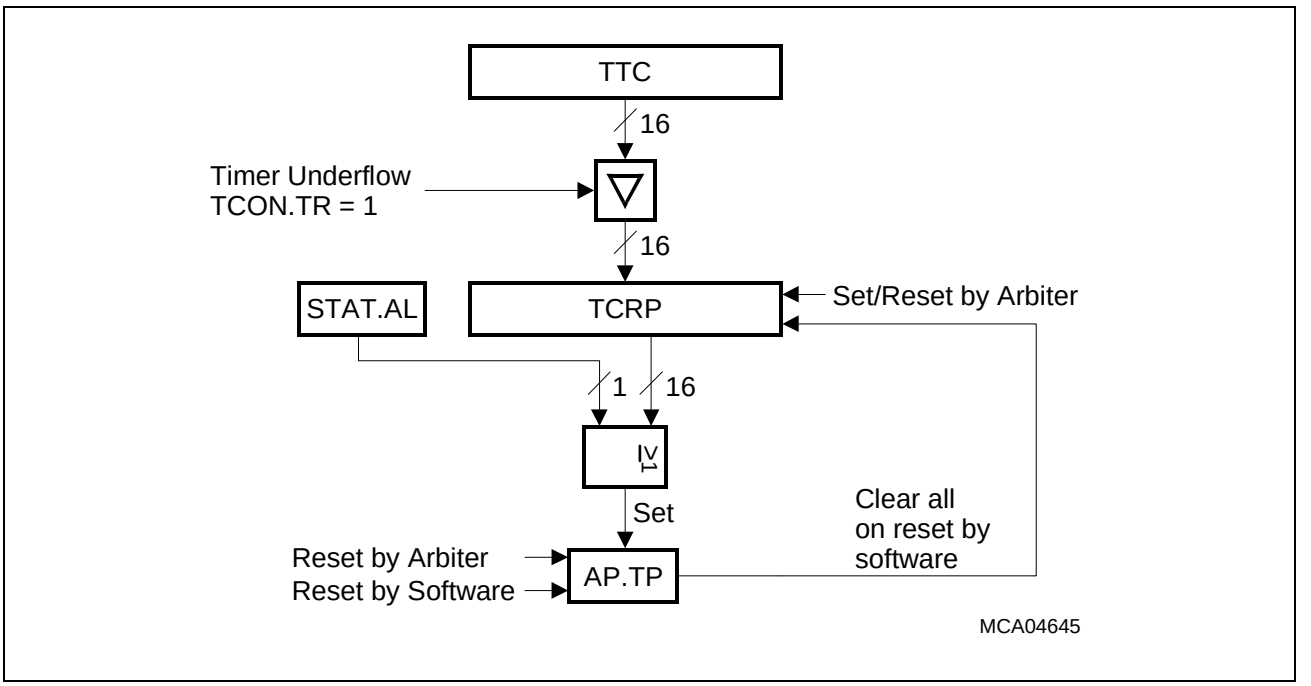


Figure 8-4 Conversion Request Source “Timer”

Up to sixteen individual selectable analog input channels can be allocated to the conversion request source “Timer”. Setting request bit(s) in the timer trigger control register enables the generation of a conversion request for this analog input channel(s) by the timer. On timer underflow, the contents of the timer trigger control register TTC are loaded into the timer conversion request pending register TCRP. This triggers conversion requests for the selected channel(s).

The contents of the timer conversion request pending register and the arbitration lock bit are logically or’ed. If bit STAT.AL or at least one bit is set in the timer conversion request pending register, the arbitration participation flag AP.TP is set. This informs the arbiter to include the conversion request source “Timer” in the arbitration.

If “Timer” is the arbitration winner, a conversion is started for the conversion request within register TCRP with the highest channel number. Starting a conversion causes the conversion request bit to be reset in register TCRP by the arbiter. If a currently running timer initiated conversion is cancelled, the arbiter sets the corresponding conversion request bit in registers TCRP for this channel. If all pending conversion requests are processed, the arbiter resets the arbitration participation flag AP.TP.

The contents of register TCRP can be cleared globally under software control by resetting the timer arbitration participation flag.

The arbitration-lock mechanism provides the means to start timer triggered conversion requests without being delayed by a currently running conversion. **Figure 8-5** shows this method in detail.

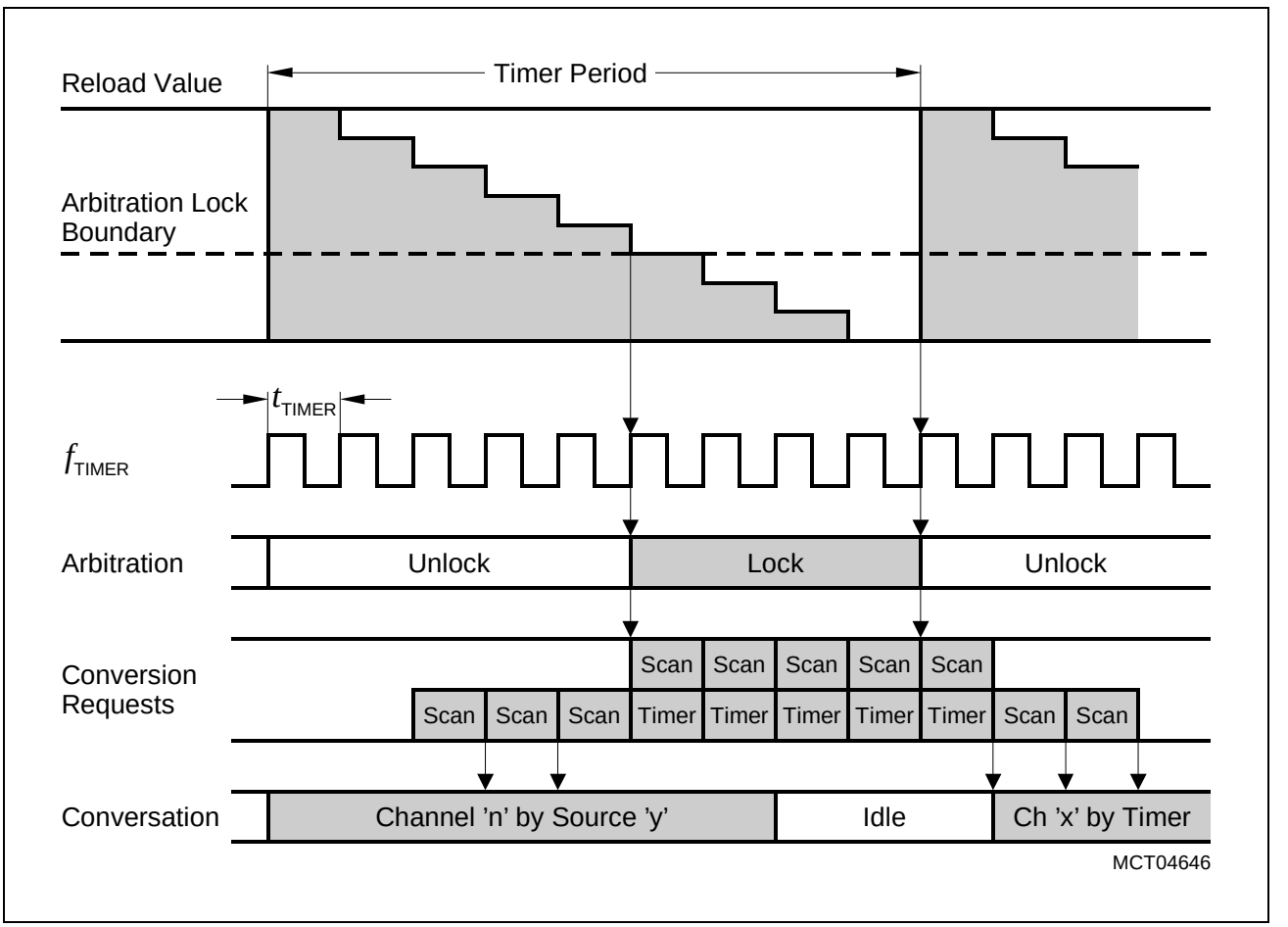


Figure 8-5 Arbitration Lock Mechanism

The arbitration must be locked a special time before the timer underflow occurs, in order to insure that the running conversion has been finished and no new conversion will be started in the meantime. While the arbitration is locked, lower prioritized conversion request source than the “Timer” are blocked from performing requested conversions.

See **Figure 8-5**, in which the conversion request source “Auto Scan” has triggered conversion request(s) that are not served (dashed arrows) according to a currently running conversion and the locked arbitration. On timer underflow, the arbitration is unlocked and the timer is detected to be the arbitration winner (it is assumed that the “Timer” is programmed to a higher priority than the “Auto Scan”).

Arbitration Lock Mode is enabled by setting bit field TCON.ALB to any value greater zero. The value of the arbitration lock boundary is also used to specify the time t_{lock} for which the arbitration is locked. Running in Arbitration Lock Mode, the current value of the timer register is compared to the arbitration lock boundary. Note that the arbitration will always be locked if the reload value is selected to be equal to or less than the arbitration

lock boundary. On a compare match, the arbitration logic is locked (STAT.AL = 1, while an timer underflow removes the arbitration lock. Bit STAT.AL is either reset on timer underflow or after resetting bit TCON.TR.

8.1.1.4 Conversion Request Source “External Event”

Externally triggered conversion requests are mandatory for a multitude of microcontroller based control applications. The conversion request source “External Event” receives triggers from the external world via the port module as well as from on-chip peripherals. **Figure 8-6** shows the two external trigger selection blocks and the conversion request source “External Event”.

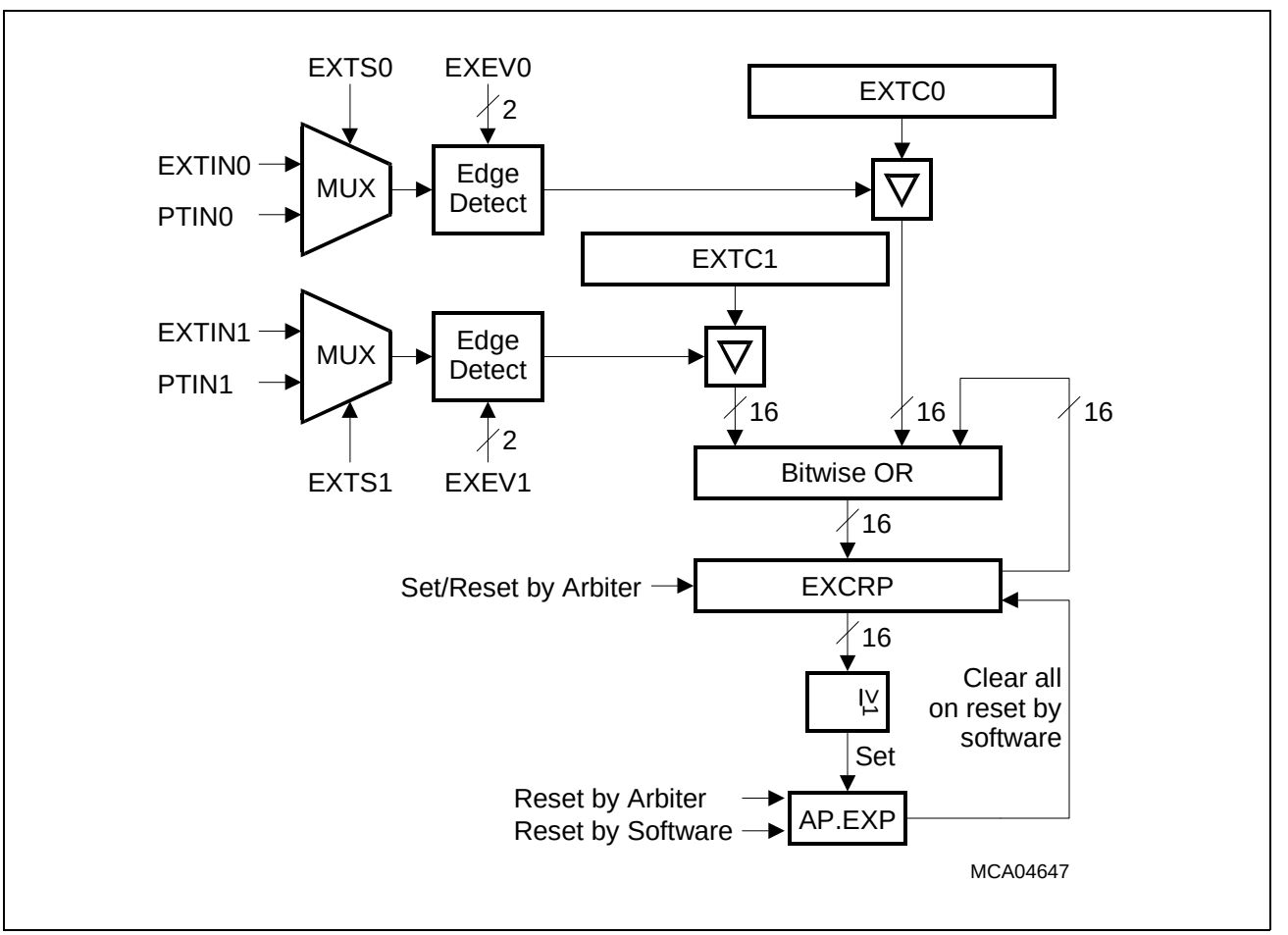


Figure 8-6 Conversion Request Source “External Event”

Up to sixteen individually selectable analog input channels per external trigger control register EXTCn can be allocated to the conversion request source “External Event”. Setting request bit(s) in the external trigger control register enables the generation of a conversion request for the analog input channel(s) on external events. A trigger issued on an external event loads the contents of the corresponding external trigger control register into the external conversion request pending register EXCRP. This triggers conversion requests for the selected channel(s).

External events can be derived from the external world via EXTINn as well as from the on-chip peripheral via PTINn. The external trigger select bits EXTSn determine for each pair of inputs, whether triggers are derived from the external world or from the on-chip peripherals. The edge detection logic can be programmed for each pair of inputs to detect none, rising, falling or both edges. Detecting “none” edges disables the conversion request source “External Event” from generating conversion requests.

If an external event is detected by an external trigger selection block, the contents of the corresponding external trigger control register are loaded into the external conversion request pending register. “Load” means that the outputs of the external trigger control registers and the external conversion request pending register are bitwise or’ed, as shown in **Figure 8-6**.

If at least one bit is set in the conversion request pending register, the arbitration participation flag AP.EXP is set. This informs the arbiter to include the conversion request source “External Event” into arbitration. If “External Event” is the arbitration winner, a conversion is started for the conversion request within register EXCRP with the highest channel number. Starting a conversion causes the conversion request bit to be reset in register EXCRP by the arbiter. If a currently running “External Event” initiated conversion is cancelled, the arbiter sets the corresponding conversion request bit in registers EXCRP for this channel. If all pending conversion requests are processed, the arbiter resets the arbitration participation flag AP.EXP.

The contents of register EXCRP can be reset globally under software control by resetting the “External Event” arbitration participation flag. Note that conversion requests caused by external trigger pulses are lost if the flag for this channel is already set in the external conversion request pending register. Trigger pulses from the external world via the port must have a duration of at least one peripheral clock cycle in order to be detected.

8.1.1.5 Conversion Request Source “Software”

The conversion request source “Software” provides the means to generate conversion request under software control, as shown in **Figure 8-7**.

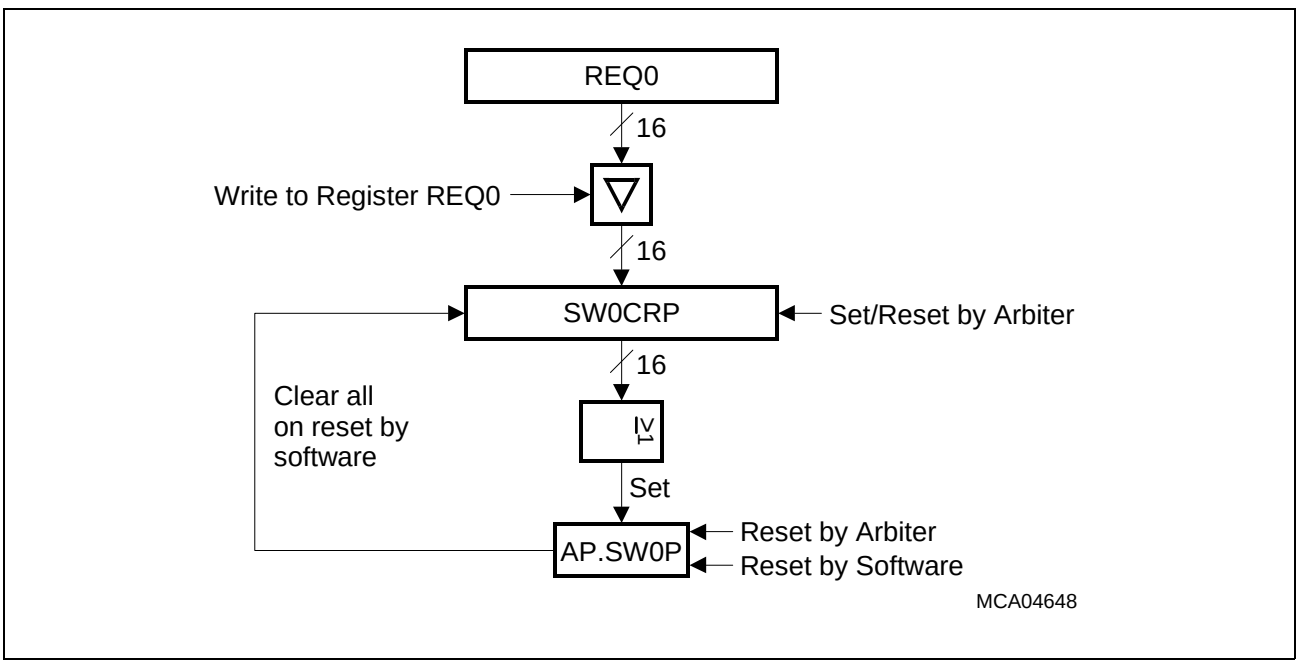


Figure 8-7 Conversion Request Source “Software”

An unique conversion request bit is associated with each analog channel. One or more request bits can be set at a time by software, resulting in a conversion request for the designated analog channel(s). Writing to the software conversion request register REQ0 automatically loads its contents to the software conversion request pending register SW0CRP. The contents of the software conversion request register remain unchanged after a load operation.

If at least one bit is set in the software conversion request pending register, the arbitration participation flag AP.SW0P is set. This informs the arbiter to include the conversion request source “Software” into the arbitration. If “Software” is the arbitration winner, a conversion is started for the conversion request within register SW0CRP with the highest channel number. Starting a conversion causes the conversion request bit to be reset in register SW0CRP by the arbiter. If a currently running “Software” initiated conversion is cancelled, the arbiter sets the corresponding conversion request bit in registers SW0CRP for this channel. If all pending conversion requests are processed, the arbiter resets the arbitration participation flag AP.SW0P.

The contents of register SW0CRP can be reset under software control either bitwise by writing a 0 to the corresponding bit position in register REQ0 or globally by resetting the “Software” arbitration participation flag.

8.1.1.6 Conversion Request Source “Auto-Scan”

The conversion request source “Auto-scan” allows continuous conversions of a selectable group of analog channels with almost zero software effort in generating and controlling these conversion requests. Auto-scan provides a single conversion sequence mode as well as continuous conversion sequence mode. Each analog channel can individually be configured to participate in an auto-scan sequence.

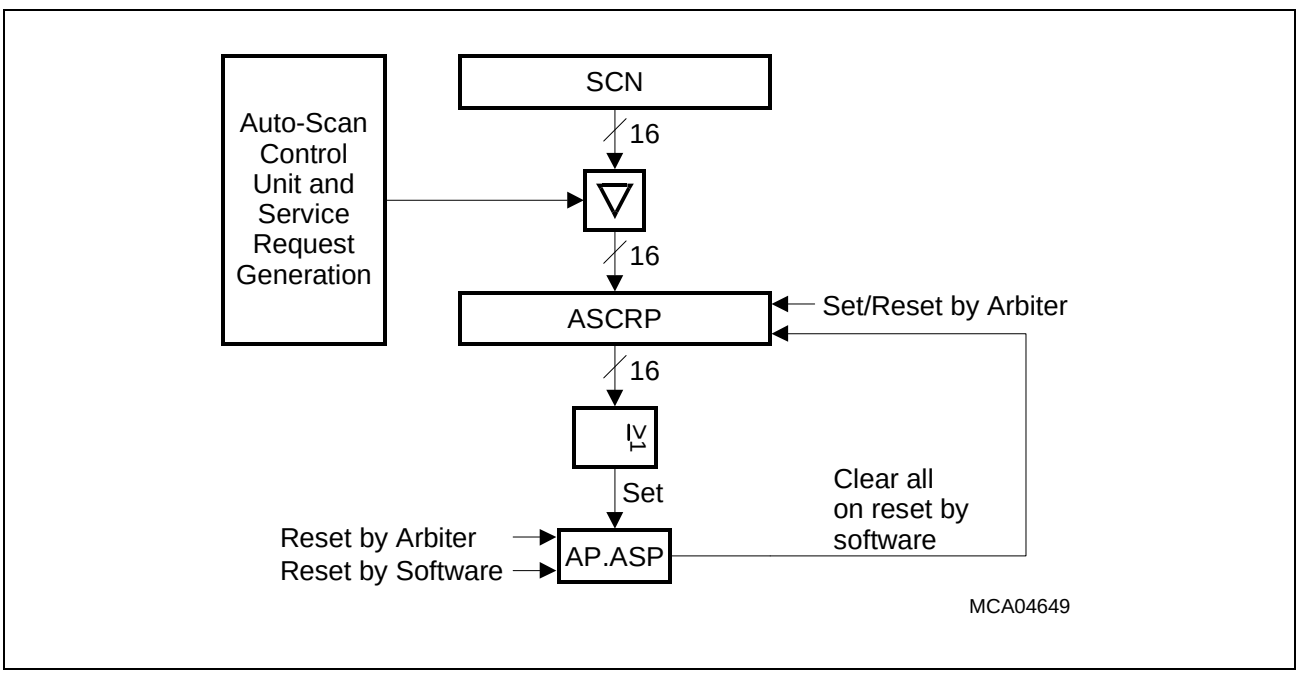


Figure 8-8 Conversion Request Source “Auto Scan”

The group of analog channels to be auto-scanned is specified in the auto-scan control register SCN by setting the corresponding channel request flags SCN.SRQn. The auto-scan sequence is started by selecting an auto-scan mode via bit field CON.SCNM. Selecting an auto-scan mode loads the contents of the auto-scan control register into the auto-scan conversion request pending register ASCRP.

If at least one bit is set in the auto-scan conversion request pending register, the arbitration participation flag AP.ASP is set. This informs the arbiter to include the conversion request source “Auto-Scan” into the arbitration. If “Auto-Scan” is the arbitration winner, a conversion is started for the conversion request within register ASCRP with the highest channel number. Therefore, pending conversion requests for the auto-scan channels are processed in the sequence from the highest to the lowest channel number. Starting a conversion causes the conversion request bit to be reset in register ASCRP by the arbiter.

The auto-scan sequence is complete if the channel with the lowest number selected to be auto-scanned has been converted (all bits of ASCRP are reset). **In single conversion sequence mode**, the bit field CON.SCNM is automatically reset and the conversion request source “Auto-Scan” enters the idle state. **In continuous conversion**

sequence mode, the conversion request source “Auto-Scan” automatically requests a new auto-scan sequence. Results previously stored in the specific channel status register(s) will be overwritten. Continuous auto-scan sequence is performed until auto-scan is stopped under software control.

If a currently running “Auto-Scan” initiated conversion is cancelled, the arbiter sets the corresponding conversion request bit in registers ASCRP for this channel.

The source service request flag MSS1.MSRAS is set after the conversion of the last channel within an auto-scan sequence was finished. Service requests can be generated only if the service request node pointer destination (SRNP.PAS) is configured and enabled (SRNP.ENPAS).

The auto-scan control functionality is described in the following tables. This includes the actions to be performed on changes in the auto-scan mode or the channels to be auto-scanned as well as resetting the auto-scan arbitration participation flag.

Table 8-3 describes the action to be performed on a change of bit field CON.SCNM.

Table 8-3 Change of Auto-Scan Mode

Value of CON.SCNM		Action
Current Value of CON.SCNM	**Value after Write Action to CON.SCNM**	
00	00	No action
00	01	Load SCN contents to register ASCRP, set bit AP.ASP, and start single auto-scan sequence if at least one channel is specified in register SCN to participate in auto-scan mode; otherwise, reset bit field CON.SCNM.
00	10	Load SCN contents to register ASCRP, set bit AP.ASP, and start continuous auto-scan sequence if at least one channel is specified in register SCN to participate in auto-scan mode; otherwise, reset bit field CON.SCNM.
00	11	Reset bit field CON.SCNM
01	00	Finish currently performed auto-scan sequence and generate a service request (if enabled) at the end of the sequence.
01	01	Continue to perform auto-scan sequence and generate a service request (if enabled) at the end of the sequence.

Table 8-3 Change of Auto-Scan Mode (cont'd)

Value of CON.SCNM		Action
Current Value of CON.SCNM	**Value after Write Action to CON.SCNM**	
01	10	Finish currently performed auto-scan conversion, generate a service request (if enabled) if this is the last channel of the auto-scan sequence. Load SCN contents in register ASCRP and start a continuous auto-scan sequence.
01	11	Reset bit field CON.SCNM, finish auto-scan sequence, and generate service request (if enabled) at the end of the sequence.
10	00	Finish auto-scan sequence and generate service request (if enabled) at the end of the sequence.
10	01	Finish currently performed auto-scan conversion and generate a service request (if enabled) at the end of the conversion if this was the last channel of the sequence. Load SCN contents to register ASCRP and start single auto-scan sequence.
10	10	Continue to perform continuous auto-scan sequence and generate a service request (if enabled) at the end of the sequence. Load SCN contents to register ASCRP and start continuous auto-scan sequence.
10	11	Reset bit field CON.SCNM and finish auto-scan sequence. Generate a service request (if enabled) at the end of the sequence.

Table 8-4 shows the actions to be taken on a change of the auto-scan control register SCN.

Table 8-4 Change of the Auto-Scan Control Register

Value of SCN		Action
Current Value of SCN	**Value after Write Action to SCN**	
<>0	0000_H	Bit field CON.SCNM and register ASCRP are reset independently from the auto-scan mode. Finish currently performed auto-scan sequence and generate a service request (if enabled) if this was the last channel of the sequence. No new auto-scan sequence is started.
<>0	<>0	**In case of CON.SCNM = 00_B, 01_B, or 11_B:** Reset bit field CON.SCNM and register ASCRP. Finish currently performed auto-scan sequence and generate a service request (if enabled) if this was the last channel of the sequence. No new auto-scan sequence is started.
<>0	<>0	**In case of CON.SCNM = 10_B:** Finish the currently performed auto-scan conversion and generate a service request is (if enabled) if this was the last channel of the sequence. Start a new continuous auto-scan sequence.

The following table shows the actions to be taken on a change of the auto-scan arbitration participation flag.

Table 8-5 Change of the Auto-Scan Arbitration Participation Flag

Current Value of ASP	Write to ASP	Action
0	0	No action
0	1	No action
1	0	**In case of bit field CON.SCNM = 00_B, 01_B, or 11_B:** Bit field SCNM and register ASCRP are reset. Finish currently performed auto-scan conversion. Generate a service request (if enabled) if this was the last channel of auto-scan sequence.
1	0	**In case of bit field CON.SCNM = 10_B:** Finish currently performed auto-scan conversion and generate a service request (if enabled) if this was the last channel of auto-scan sequence. Start new continuous auto-scan sequence
1	1	Don’t care

8.1.1.7 Conversion Request Source “Channel Injection”

The conversion request source “Channel Injection” generates sequential conversion requests for analog channels either with Wait-Inject or Cancel-Inject-Repeat functionality. “Channel Injection” consists of the Channel Injection control register, the back-up register, and the channel injection arbitration participation flag.

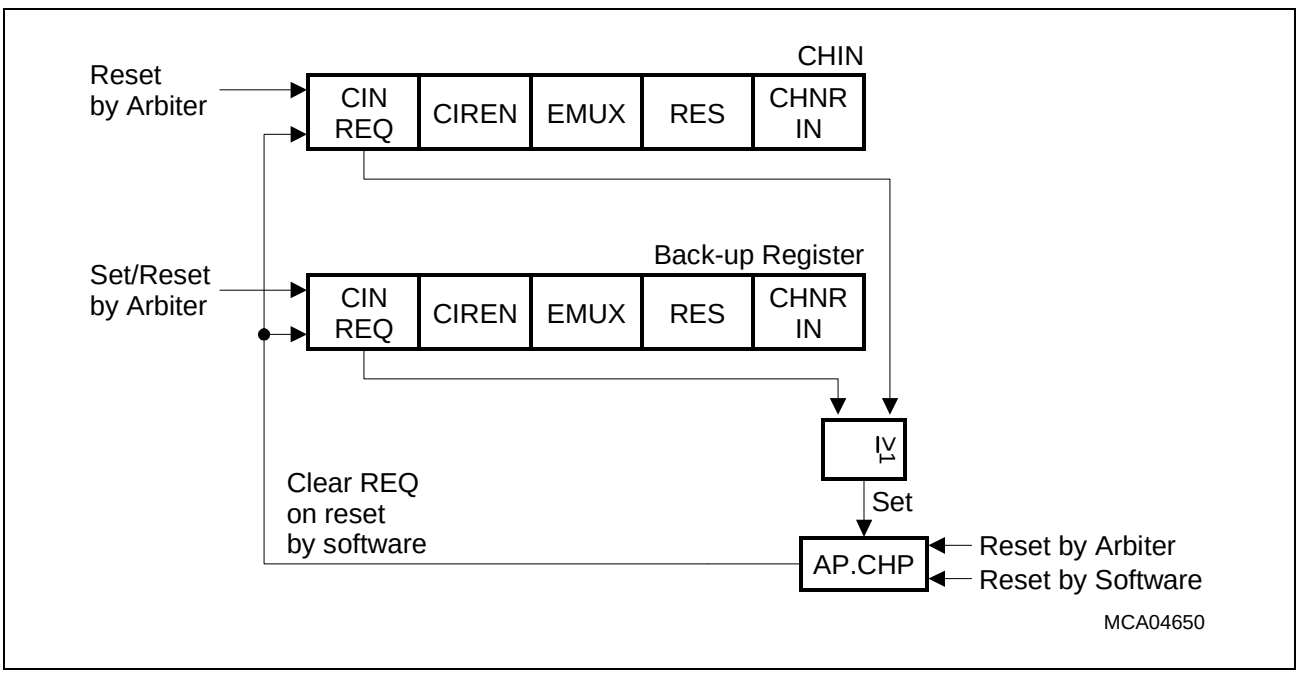


Figure 8-9 Conversion Request Source “Channel Injection”

The Channel Injection request control register CHIN contains a conversion request bit CHIN.CINREQ, controls for selecting the cancel inject repeat feature CHIN.CIREN, control information for external multiplexer settings CHIN.EMUX and control information for selecting the resolution of the ADC CHIN.RES and the channel number to be converted CHIN.CHNRIN.

Setting the channel injection request bit causes the arbitration participation flag to be set. This informs the arbiter to include the conversion request source “Channel Injection” into arbitration. If “Channel Injection” is the arbitration winner, a conversion is started for the analog channel specified within the conversion request control register. The settings of the external multiplexer and the resolution of the ADC are also derived from this register.

Starting a conversion causes the channel injection request bit to be reset by the arbiter. The channel injection arbitration participation flag is automatically reset if the channel injection control register and the back-up register contain no valid request.

If a currently running conversion initiated by “Channel Injection” is cancelled, the arbiter restores the conversion information in the back-up for this channel. In this context, conversion information refers to the conversion request bit, the setting for the external multiplexer, and the settings of the ADC’s resolution. If the back-up register contains valid conversion information, the arbiter reads from the back-up register instead from the

channel injection control register. Thus, the previously cancelled conversion participates in arbitration once again. A new conversion requested via the conversion request control register will be performed after the request in the back-up register is served.

The request bit of the channel injection control register and the backup register can be cancelled under software control. Resetting the arbitration participation bit clears either the request bit in the request register (the back-up register contains no request) or the request bit in the back-up register (the back-up register contains a valid request).

As mentioned previously, “Channel Injection” generates sequential conversion requests for analog channels either with the Inject-Wait or the Cancel-Inject-Repeat functionality.

- **Channel Injection with Inject-Wait** provides the means to wait until all pending conversions with higher priority are finished before the requested conversion is injected. The Inject-Wait feature is selected by default after initialization.
  Channel Injection with Cancel-Inject-Repeat “Cancels” a currently performed conversion, “Injects” the requested conversion, and finally “Repeats” the previous cancelled conversion. The Cancel-Inject-Repeat feature is enabled if bit CHIN.CIREN is set. Using this feature, the currently performed conversion is cancelled if its source arbitration level is lower than the source arbitration level of Channel Injection. If a currently performed conversion is cancelled, a new request is generated for this conversion. Thus, the previously cancelled conversion participates in the arbitration once again.

The following examples give an overview on the behavior of the conversion request source “Channel Injection”.

Figure 8-10 shows the functionality of conversion requests generated by “Channel Injection” with Inject-Wait feature. The conversion requested with a source-arbitration-level of ‘L3’ waits until the currently performed conversion with a source-arbitration-level of ‘L1’ is finished. The second channel injection request is delayed until both conversions requested with a source-arbitration-level of ‘L2’ are finished.

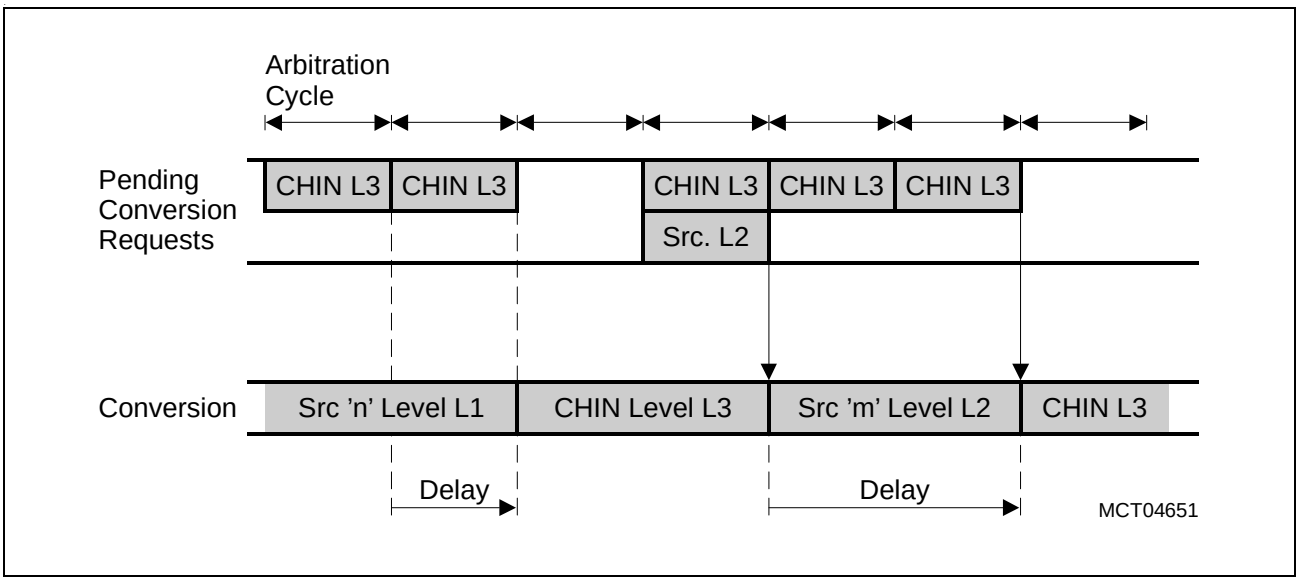


Figure 8-10 Channel Injection with Inject-Wait

Figure 8-11 shows the behavior of conversion requests generated by “Channel Injection” using the Cancel-Inject-Repeat feature. In the first case, the currently performed conversion is cancelled, since its source arbitration level of ‘L2’ is below the source arbitration level of ‘L1’ of “Channel Injection”. A new conversion request is generated for the cancelled conversion in order to restart this cancelled conversion later. This new request participates in arbitration and will be selected for repetition due to its priority level. The second injection request with a source arbitration level of ‘L4’ is delayed, even if the Cancel-Inject-Repeat feature is enabled.

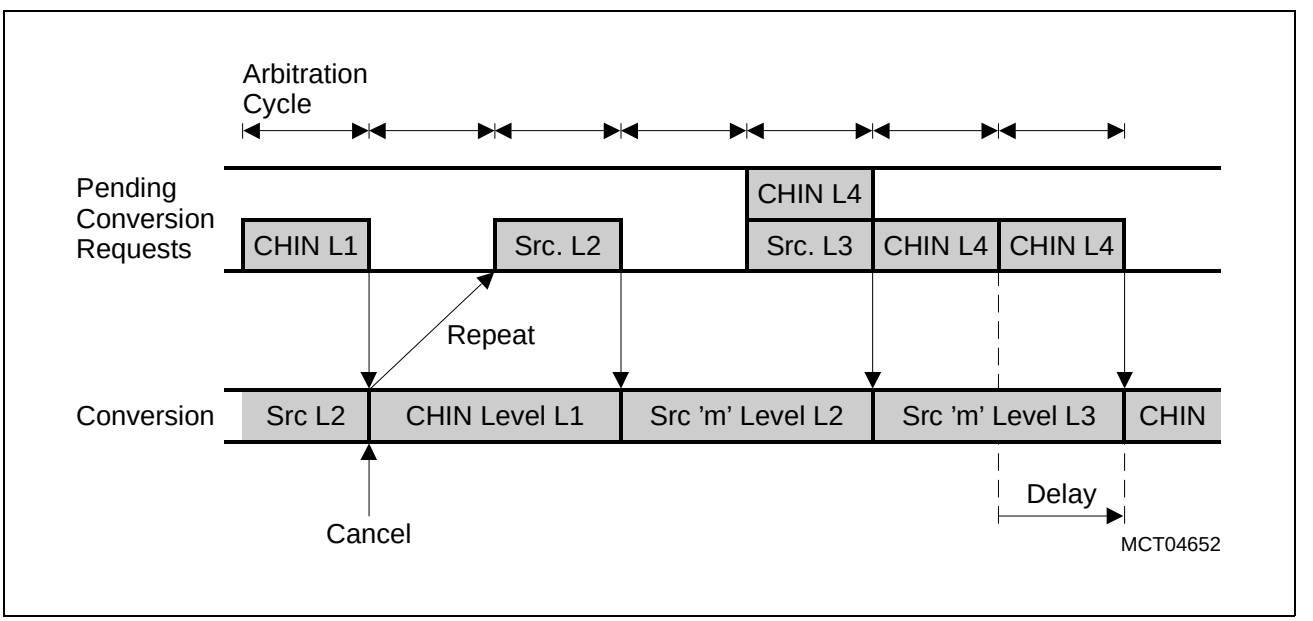


Figure 8-11 Channel Injection with Cancel-Inject-Repeat Feature

Figure 8-12 shows the teamwork of conversions requested by “Channel Injection” and conversions triggered by “Timer” running in Arbitration Lock Mode. First, a conversion is requested by “Channel Injection” with a source arbitration level of ‘L3' using the Cancel-Inject-Repeat feature, during which the arbitration is locked by the timer. This request is delayed until the timer triggered conversion is finished or until “Channel Injection” is programmed to a higher priority than the timer. Second, a conversion is requested by “Channel Injection” with a source arbitration level of ‘L1’ with the Cancel-Inject-Repeat feature selected, during which the arbitration is locked by the timer. In this case the arbitration lock is not taken into account because the timer was programmed on source arbitration level ‘L2’. Even a currently running timer triggered conversion would have been cancelled and participates in arbitration anew.

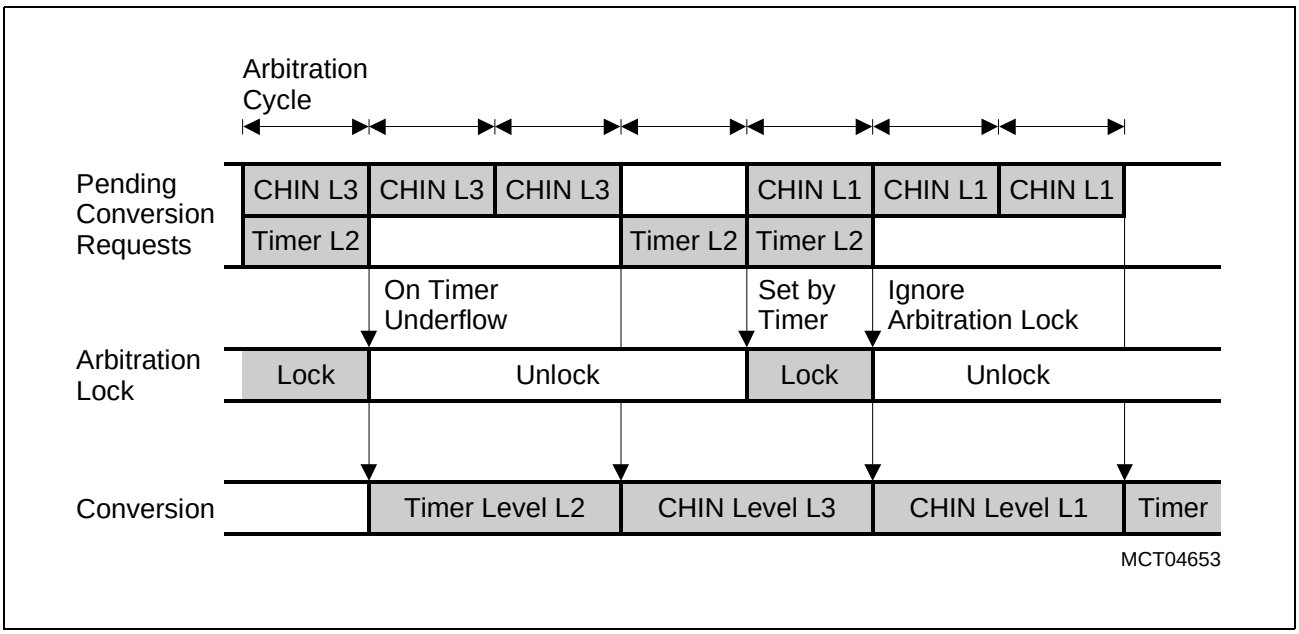


Figure 8-12 Channel Injection and Timer Triggered Conversion

8.1.1.8 Conversion Request Source “Queue”

The conversion request source “Queue” with its queue storage block is designed to handle and store burst transfers of conversion request. Dedicated queue filling-state control logic can be used to request the next burst transfer of data while the queue’s filling level is below a predefined level.

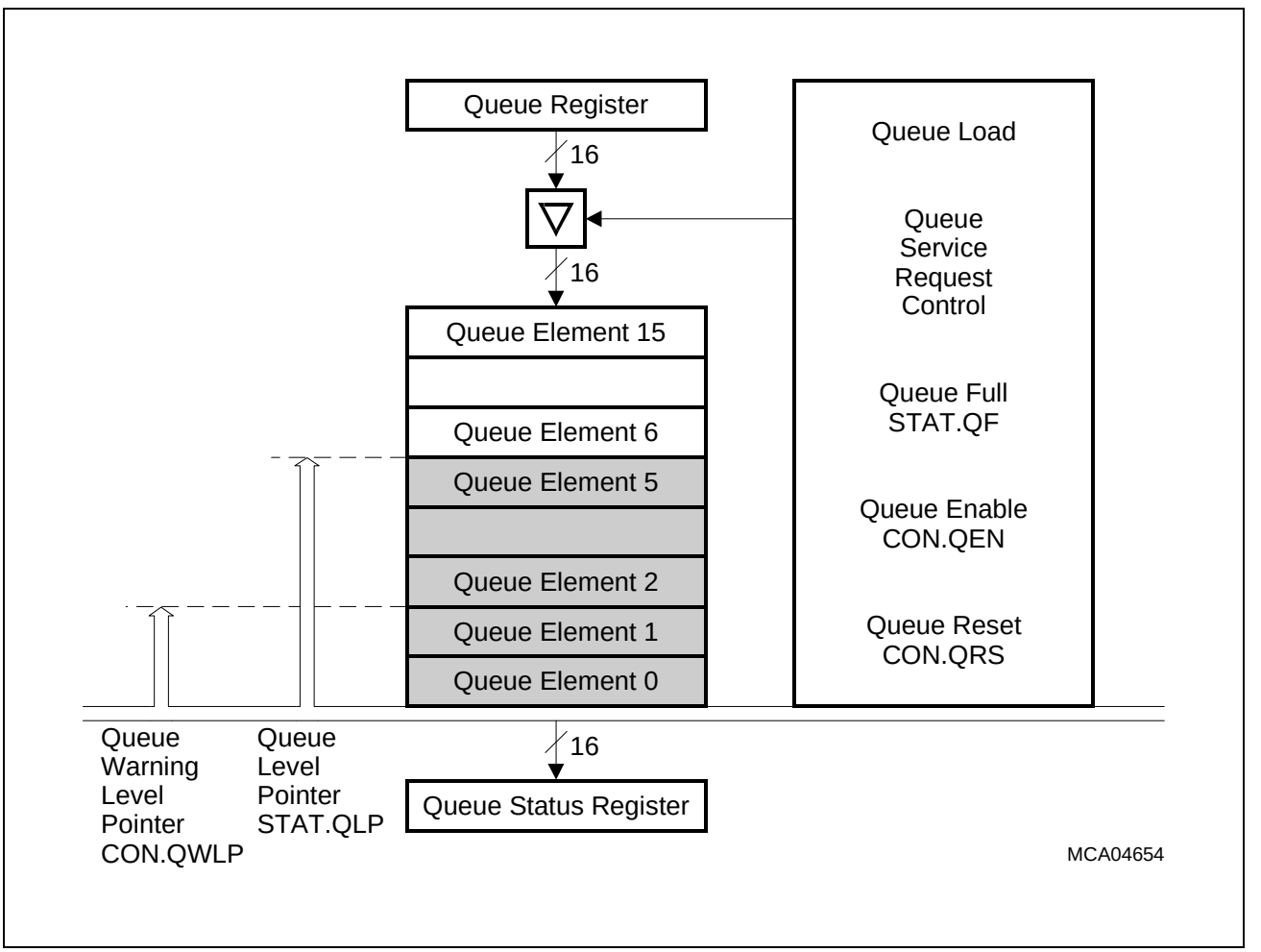


Figure 8-13 Queue Storage Block Diagram

The queue consists of a queue register QR, sixteen queue elements, queue status register QUEUE0, and the queue control logic, as shown in **Figure 8-13**.

The queue control logic includes the queue load logic, a queue level pointer, a queue warning limit pointer, the queue based service request control block, as well as control and status flags to monitor and control the queue state.

The queue register, the queue status register, and each of the sixteen queue elements contains a valid bit (v-bit), external multiplexer control bits (EMUX), A/D Converter’s resolution control bits (RES), and the channel number for which an conversion should be started (CHNR).

The queue is automatically filled by writing valid data to the queue register. Valid data means that at least the V-bit is set, while “zero” is a valid option for the external multiplexer setting, the resolution control bit field and the channel number. Valid data in the queue register (QR.V-bit, QR.EMUX, QR.RES and QR.CHNR data) is then copied to the next empty queue element determined by the queue level pointer STAT.QLP. The queue load operation causes the valid bit in the queue register to be reset automatically. Any software access to the queue register is denied during this copy operation. No queue load is performed if the queue state is full (STAT.QF is set) and the queue register contains valid data.

As shown in **Figure 8-13**, queue elements zero to five contain valid data; therefore, the queue register’s contents are copied to queue element six.

The queue level pointer indicates the number of valid queue elements. It is incremented after a queue load operation. It is decremented after a queue based conversion is started, or after the queue participation flag is reset. The queue level pointer is cleared after a queue reset operation by setting the queue reset bit. Note that there are sixteen valid queue elements in the queue if the queue level pointer is $0F_H$ and the queue full bit is set.

The queue warning limit pointer CON.QWLP can be used to generate service requests, based on a queue element state change. The value of the queue warning limit pointer must be programmed with a value “n” in order to focus on a state change from valid to invalid of queue element “n”. A queue based service request can be triggered in this case, thus requesting the next burst transfer of data to the queue. If the state change of queue element “n” occurs, the module service request flag MSS1.MSRQR is automatically set. The service request destination node pointer (PQR) must be configured and enabled (ENPQR) in order to trigger a service request node assigned to the queue.

The conversion request source “Queue” consists of the queue status register QUEUE0, a backup register, and a queue arbitration participation flag AP.QP, as shown in **Figure 8-14**. The contents of queue element number zero are represented in the queue status register QUEUE0. Therefore, set/reset actions of the valid-bit of the queue status register QUEUE0 are also performed on queue element zero.

If at least one queue element contains valid data, this (these) valid bit(s) cause(s) the queue arbitration participation flag to be set. This informs the arbiter to include the conversion request source “Queue” into arbitration. If “Queue” is the arbitration winner, a conversion is started for the analog channel specified within the queue status register. The settings of the external multiplexer and the resolution of the A/D Converter are also derived from this register.

Starting a queue based conversion causes the valid bit of the queue status register QUEUE0 to be reset by the arbiter. The contents of all queue elements containing valid data slide one step down. For example, queue element one contains valid data, this data “slides down” to queue element zero. Queue based conversion requests are generated

for the control information of register QUEUE0, if the queue is enabled (bit CON.QEN = 1) and the queue status register contains valid data (QUEUE0.V-bit is set). The arbitration participation flag is automatically reset if all queue elements, the queue status register (remember: QUEUE0 represents the contents of queue element number zero) contains and the back-up register contain no valid request.

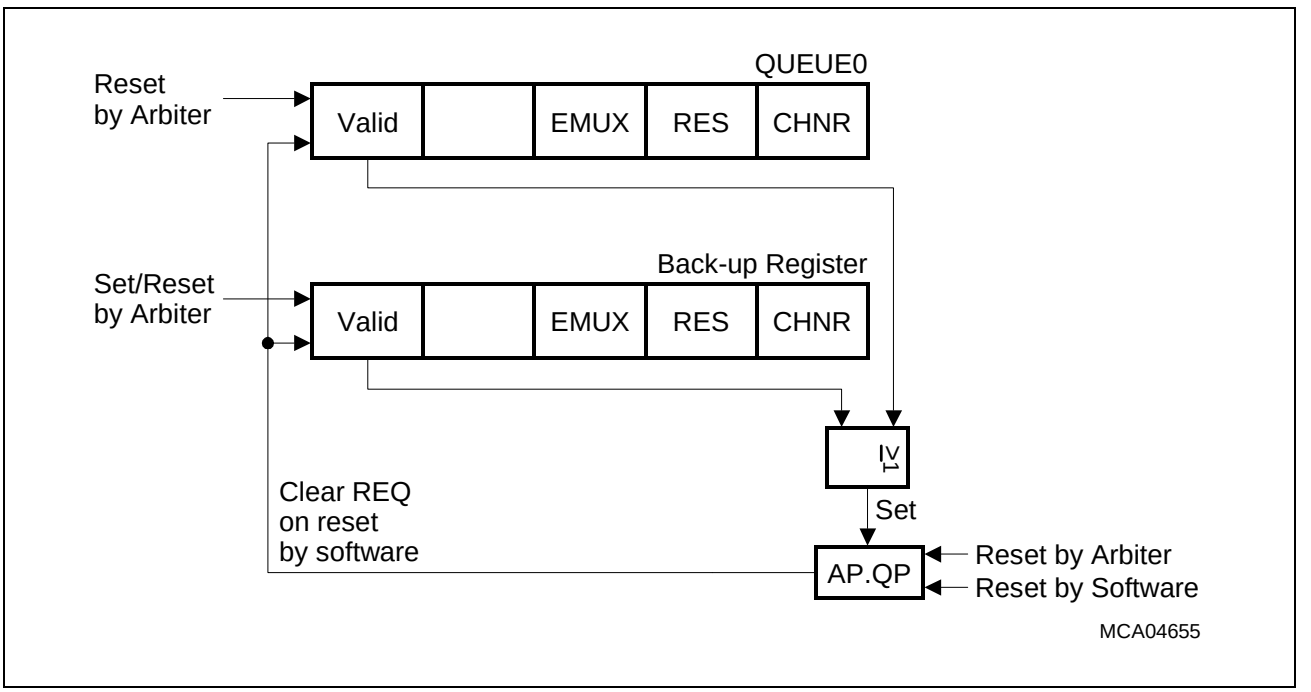


Figure 8-14 Conversion Request Source “Queue”

If a currently running conversion initiated by “Queue” is cancelled, the arbiter restores the conversion information in the back-up for this channel. In this context, conversion information refers to the conversion request bit, the setting for the external multiplexer and the settings of the A/D Converter’s resolution. If the back-up register contains valid conversion information, the arbiter reads from the back-up register instead of the queue status register. Thus, the previously cancelled conversion participates in arbitration once again. A conversion requested via the queue storage block (register QUEUE0) will be performed after the request in the back-up register is served.

The valid bit (V-bit) of the queue status register and the back-up register can be cancelled under software control. Resetting the queue arbitration participation bit clears either the valid bit in the queue status register (the back-up register contains no request) or the request bit in the back-up register (the back-up register contains a valid request). If the valid bit of the queue status register is cleared, a slide operation is performed equal to the slide operation after starting a queue based conversion.

8.1.2 Arbitration

Since several conversion request sources can generate conversion requests at the same time, an arbitration mechanism is implemented in order to detected the conversion request source and channel with the highest priority. **Figure 8-15** shows the arbitration scheme with the associated controls.

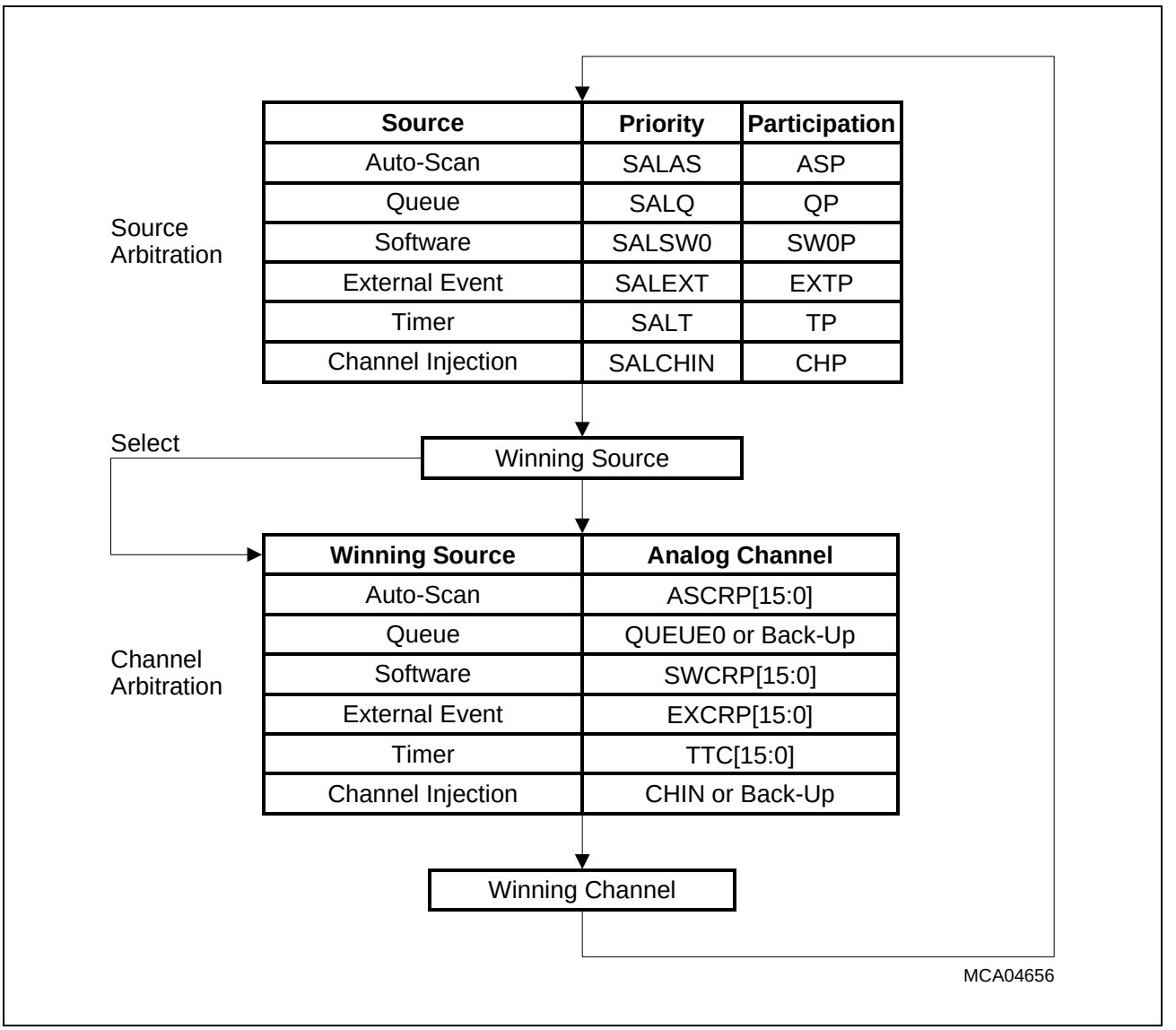


Figure 8-15 Arbitration

Arbitration of pending conversion requests is performed according to the following two stage prioritization algorithm:

- *Source arbitration* is the first stage in the arbitration algorithm. Starting with the conversion request source “Auto-Scan” up to “Channel Injection”, each source is checked if its arbitration participation flag is set. If the participation flag is set and its priority is higher than the priority of the previous selected source, that source is the momentary winner of the arbitration. Ultimately, a winning source is detected.

- *Channel arbitration* follows after source arbitration. For the prioritized source (winning source), channel arbitration is performed. Within the second stage of the arbitration algorithm, the pending conversion request with the highest priority is detected. If a parallel source is the winning source, the flag representing the highest channel number within the conversion request pending register is determined. If a sequential source is the winning source, the channel in the request register or in the back-up register is determined. Note that a pending request in the back-up register is preferred. Ultimately, a channel is detected.

The arbitration result consists of the winning source and channel number. A start of conversion can occur, if the A/D Converter is idle or if the arbitration winner has permission to cancel a currently running conversion. After the conversion has started, the corresponding pending conversion request is automatically reset. Attempt to start a conversion for this arbitration result will be repeated until either the start is successful or a new result (source and channel number) was arbitrated.

8.1.2.1 Source Arbitration Level

The priority of each conversion request source can be programmed individually in the corresponding bit fields of the source arbitration level register SAL. The priority of a source is named as source-arbitration-level and it determines the order in which pending conversion requests from different sources are performed. A low number of the source-arbitration-level represents a high priority and vice versa.

After initialization, each source is assigned an individual source-arbitration-level. “Channel Injection” has the highest priority, while “Auto-Scan” has the lowest priority. These predefined priority levels can be reprogrammed to adapt the ADC’s functionality to the requirements of the application.

It is recommended that source-arbitration-levels be reprogrammed while no conversion request is pending, as any modification of source arbitration level register immediately affects the arbitration scheme. Each source should have assigned an individual priority level. Nevertheless, if several conversion request sources have been programmed to the same priority level, the first detected source within this group of identical levels is taken into account.

8.1.2.2 Arbitration Participation Flags

Each source has assigned an arbitration participation flag located in the arbitration participation register AP. An arbitration participation flag set to 1 indicates, that at least one conversion request is generated by this source and that this source participates in the arbitration.

The arbitration participation flag is automatically reset by the arbiter if no conversion request is pending for this source (all requested conversions have been started).

The arbitration participation flag can also be reset under software control. Writing a 0 to the corresponding flag resets the arbitration participation flag. All bits in the corresponding conversion request pending register are reset if a participation flag of a parallel source is reset under software control. If a participation flag of a sequential source is reset, the following action is performed:

- **ONLY** the request bit of the backup register is reset, **if the backup register contains valid data**. The request bit of the corresponding conversion request register (CHIN or QUEUE0) is **not** reset in this case.
- **OR** the request bit of the corresponding conversion request register (CHIN or QUEUE0) is reset, **if the backup register contains invalid data**.

Note: Writing a 1 to a participation bit is not taken into account.

8.1.2.3 Cancel Functionality

Channel Injection and Synchronized Injection have the ability to cancel a currently running conversion. If a conversion is cancelled, the following action is performed:

- If a conversion initiated by a parallel source is cancelled, the conversion request flag is automatically set again in the corresponding conversion request pending register.
- If a conversion initiated by a sequential source is cancelled, the control information (such as resolution, external multiplexer information, etc.) of the cancelled conversion is rescued into the backup register (for example: queue based conversion is cancelled, so the queue backup register receives the control information of the cancelled conversion).

Thus, the request participates in the arbitration anew and will be served according to its source-arbitration level.

8.1.2.4 Clear of Pending Conversion Requests

This feature can be used to save conversion time by handling more than one conversion request at the same time.

Clear of pending conversion requests in parallel sources: If several conversion requests are pending for the same analog channel and a conversion for this analog channel has been started, all pending conversion requests of **parallel** sources are cancelled for this analog channel by the arbiter (for example: timer, software and auto-scan triggered each a conversion request for the same analog channel. Thus, only one conversion is started for this analog channel. The other two pending conversion requests will be automatically cancelled by the arbiter. Note that the conversion will be started for the arbitration winner, the source with the highest priority). The conversion result is correct for all parallel sources which requested this channel. A service request is generated only for the source that caused the processed conversion.

Individual clear of pending conversion requests: If several conversion requests are pending for the same analog channel, this channel will be converted several times until all pending conversion requests are performed. This is the default setting after reset.

8.1.2.5 Arbitration and Synchronized Injection

The master of a Synchronized Injection provides no separated source for this feature. The behavior of a Synchronized Injection is specified by the original requesting source.

In the slave, a request for a Synchronized Injection **always** has the highest priority. A request for a synchronized request in a slave did not participate in the arbitration cycle. This synchronized request is immediately set as the arbitration winner. This request remains until it is served or it is redrawn by the master.

8.1.2.6 Arbitration Lock

If the timer runs in Arbitration Lock Mode and the current timer value TSTAT.TIMER is equal to or below the arbitration lock boundary the arbitration lock bit STAT.AL is set. Setting the arbitration lock bit also sets the timer participation flag. In this way the timer source can participate in the arbitration cycle without any pending request. Such an arbitration participation by the timer without a pending request denies all currently pending sources that have a source-arbitration-level below the timer source as arbitration winner. All sources with a source-arbitration-level greater than the timer source keep their possibility to win the arbitration. If the timer wins the arbitration without a pending request, no conversion will be started for this arbitration winner. This case can occur if bit AP.TP is set while no bit is set in register TCRP. This feature can be used to guarantee that no conversions can be started for lower prioritized sources.

Note: The timer participation flag is also set by any pending timer conversion request in register TCRP.

Note: If any source has the same source-arbitration-level as the timer source, the result of the arbitration cycle depends on the position of this source compared to the timer source. If this source is checked before the timer source, this can be the arbitration winner. If this source is checked after the timer source, this source can’t be the arbitration winner.

8.1.3 Clock Circuit

The clock divider blocks shown in **Figure 8-16** determine the clock frequencies in the A/D Converter Module and therefore the conversion and sample timing.

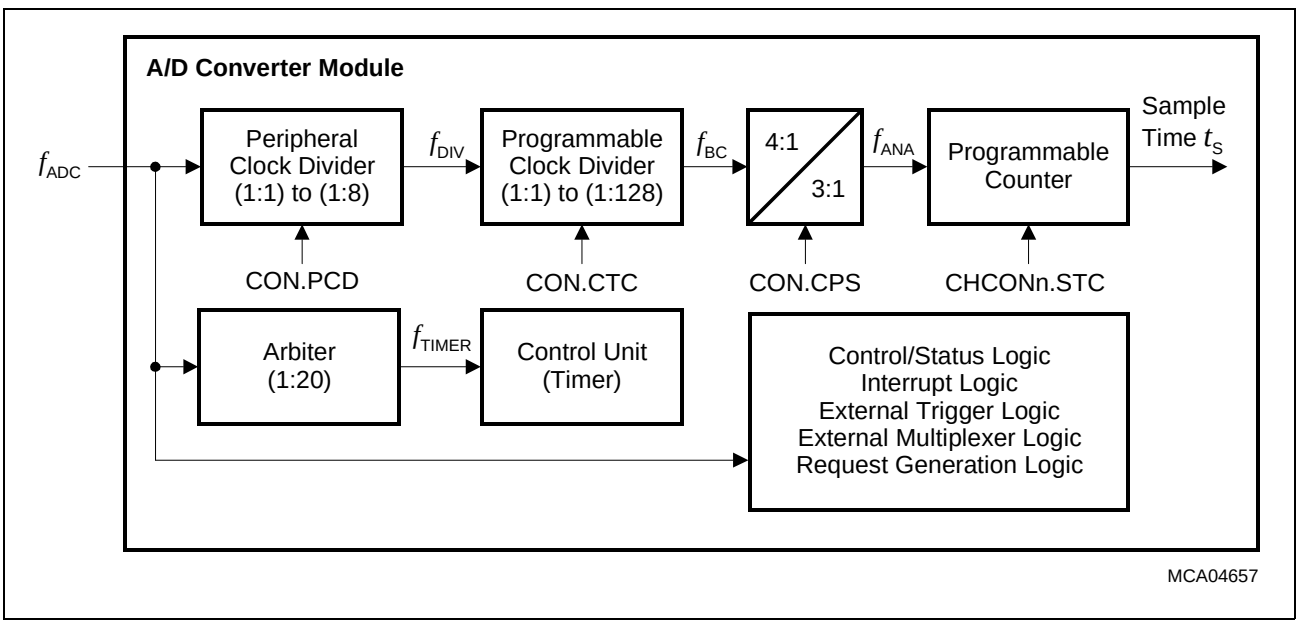


Figure 8-16 Clock Control Structure

The following definitions for the A/D Converter clocks are used in this chapter:

- f_{ADC}: Peripheral clock
- f_{DIV}: Divided peripheral clock
- f_{BC}: Basic operating clock
- f_{ANA}: Internal A/D Converter clock
- f_{TIMER}: Arbiter clock

The conversion time is composed of the sample time, the time for the successive approximation and the calibration time. **Table 8-6** shows the conversion time t_C based on the sample time t_S, basic operating clock frequency f_{BC} and the divided module clock f_{DIV} ($t_{BC} = 1 / f_{BC}$, $t_{DIV} = 1 / f_{DIV}$).

Table 8-6 Conversion Time t_C

A/D Converter Resolution	Clock Divider (CON.CPS)	Conversion Time t_C
8 - bit	0	$t_S + 30\ t_{BC} + 2\ t_{DIV}$
	1	$t_S + 34\ t_{BC} + 2\ t_{DIV}$
10 - bit	0	$t_S + 36\ t_{BC} + 2\ t_{DIV}$
	1	$t_S + 40\ t_{BC} + 2\ t_{DIV}$
12 - bit	0	$t_S + 42\ t_{BC} + 2\ t_{DIV}$
	1	$t_S + 46\ t_{BC} + 2\ t_{DIV}$

Note: The TC1775 basic operating clock frequency f_{BC} influences the maximum allowable internal resistance of the used reference voltage supply.

8.1.3.1 Conversion Principles

After reset, a power-up calibration is automatically performed in order to correct gain and offset errors of the A/D Converter. The ongoing power-up calibration is indicated in the A/D Converter status register by an activated calibrate bit STAT.CAL. The power-up calibration is interruptible and will be continued after the inserted conversion is finished. Note that this inserted conversion will be performed with less accuracy. To achieve best calibration results, the reference voltages as well as the supply voltages must be stable during the power-up calibration.

When a conversion is started, first the capacitances of the converter are loaded via the respective analog input channel to the analog input voltage. The time to load the capacitances is referred to as sample time t_S. The sample phase is indicated by an activated status bit STAT.SMPL in the A/D Converter status register. Next, the sampled voltage is converted to a digital value. Finally an internal self calibration adapts the analog converter module to changing temperatures and device tolerances. The conversion and calibration phase is indicated by the busy signal STAT.BUSY, which goes inactive at the end of the calibration phase.

8.1.3.2 Peripheral Clock Divider

The peripheral clock divider is automatically activated with a divide factor of 4 after reset and can be configured under software by setting bit field CON.PCD.

The following equation shows the dependency of the divided peripheral clock f_{DIV} from f_{ADC}.

$$f_{DIV} = \frac{f_{ADC}}{2^{PCD}}$$

Note: The maximum basic operating clock f_{BC} must not exceed 6 MHz when using a dividing factor of 3, or 8 MHz when using a dividing factor of 4 (see next chapter for details).

8.1.3.3 Conversion Timing Control

The A/D Converter basic operating clock frequency f_{BC} is derived from f_{DIV} via the programmable clock divider, which provides 128 dividing factors from (1:1 to 1:128). The basic operating clock is related to f_{DIV} according to the following equation:

$$f_{BC} = \frac{f_{DIV}}{CTC + 1}$$

The A/D Converter basic operating clock frequency f_{BC} must not exceed 6 MHz when using a dividing factor of 3 (CON.CPS = 0), or must not exceed 8 MHz when using a dividing factor of 4 (CON.CPS = 1). The basic operating clock must also not drop below 0.5 MHz.

The internal A/D Converter clock frequency f_{ANA} is either a third or a quarter of the TC1775 basic operating clock frequency f_{BC}, based on the state of the control bit CON.CPS. The internal A/D Converter clock is related to f_{DIV} according to the following equation:

$$f_{ANA} = \frac{f_{BC}}{CPS + 1} = \frac{1}{CPS + 1} \times \frac{f_{DIV}}{CTC + 1}$$

With the clock control bit fields CON.CTC and CON.CPS, the internal A/D Converter clock f_{ANA} can be adjusted to different peripheral clock frequencies f_{ADC} in order to optimize the performance of the TC1775. Note that CON.CTC and CON.CPS may be changed during a conversion, but will be evaluated after the currently performed conversion is finished.

Table 8-7 Conversion Timing Control

CON.CTC	CON.CPS	f_{BC}	t_{BC}	f_{ANA}	t_{ANA}
0000000_B	0	f_{DIV} / 1	1 / f_{DIV}	f_{DIV} / 3	3 / f_{DIV}
	1			f_{DIV} / 4	4 / f_{DIV}
0000001_B	0	f_{DIV} / 2	2 / f_{DIV}	f_{DIV} / 6	6 / f_{DIV}
	1			f_{DIV} / 8	8 / f_{DIV}
0000010_B	0	f_{DIV} / 3	3 / f_{DIV}	f_{DIV} / 9	9 / f_{DIV}
	1			f_{DIV} / 12	12 / f_{DIV}
0000011_B	0	f_{DIV} / 4	4 / f_{DIV}	f_{DIV} / 12	12 / f_{DIV}
	1			f_{DIV} / 16	16 / f_{DIV}
:	:	:			
1111111_B	0	f_{DIV} / 128	128 / f_{DIV}	f_{DIV} / 388	388 / f_{DIV}
	1			f_{DIV} / 512	512 / f_{DIV}

8.1.3.4 Sample Timing Control (STC)

The sample time control defines the duration of the sample phase of a conversion, that is, the period during which the channel input capacitance is charged/discharged by the selected analog signal source. The duration of the sample phase is programmed individually for each channel via sample time control bit field CHCONn.STC. Any modification of CHCONn.STC will be evaluated after the currently performed conversion is terminated.

The sample time t_S depends on the ADC basic operating clock f_{BC} and the programmable value of bit field CHCONn.STC. The sample time t_S is selected in periods of $t_{BC} = 1 / f_{BC}$ within the range from $6 \times t_{BC}$ up to $1028 \times t_{BC}$.

The sample time t_S is calculated according to the following equation:

$$t_S = (3 + CPS) \times (STC + 2) \times t_{BC}$$

The following table shows the selectable values of CON.CPS and CON.STC and the resulting ADC basic operating clock f_{BC} and sample time t_S .

Table 8-8 Sample Time Control

CHCONn.STC	CON.CPS	t_S
00000000_B	0	$6 \times t_{BC}$
	1	$8 \times t_{BC}$
00000001_B	0	$9 \times t_{BC}$
	1	$12 \times t_{BC}$
00000010_B	0	$12 \times t_{BC}$
	1	$16 \times t_{BC}$
00000011_B	0	$15 \times t_{BC}$
	1	$20 \times t_{BC}$
:	:	:
11111111_B	0	$771 \times t_{BC}$
	1	$1028 \times t_{BC}$

Note: The duration of the sample phase influences the maximum allowable internal resistance of the respective analog input signal source.

8.1.3.5 Power-Up Calibration Time

The power-up calibration takes 3328 × t_{ANA}. As an example, with the reset values of the A/D Converter registers the power-up calibration is calculated as follows:

- f_{SYS} = 40 MHz
- f_{ADC} = f_{SYS} / 1 = 40 MHz (ADCx_CLC.RMC = 00000001_B)
- f_{DIV} = f_{ADC} / 4 = 10 MHz (CON.PDC = 10_B)
- f_{BC} = f_{DIV} / 4 = 2.5 MHz (CON.CTC = 0000011_B)
- f_{ANA} = f_{BC} / 4 = 0,625 MHz (CON.CPS = 1)

These values result in a power-up calibration:

f_{ANA} = 0.625 MHz or t_{ANA} = 1.60 µs

Power-up calibration time = 3328 × t_{ANA} = 5.325 ms

After reset the A/D Converter must be enabled by software by writing register ADCx_CLC. By default after reset, the A/D Converter is disabled.

Note: The time for power-up calibration can be reduced by setting the above described clock divider registers to values providing the ADC with a faster clock rate. This can be done during a running power-up calibration.

8.1.4 Reference Voltages (V_{AREF} and V_{AGND})

The digital result of a conversion represents the analog input as a fraction of the reference (V_{AREF} - V_{AGND}) in steps of 2^{-n} by n-bit resolution. The ADC Module offers the choice of four selectable reference voltages $V_{AREF}[0]$ to $V_{AREF}[3]$. The reference voltage can independently be selected for each analog channel via the respective bit field CHCONn.REF. $V_{AREF}[0]$ corresponds to the positive reference voltage V_{AREF} and is used for self calibration of the A/D Converter. Therefore, it must be stable during all conversions, even for those which use another reference voltage.

The reference voltages must fulfill the following specifications:

$$V_{AREF}[0] \leq V_{DDA} + 0.1\ V;\ V_{DDA} \leq 5\ V$$

$$V_{AREF}[1] \text{ to } V_{AREF}[3] \leq V_{AREF}[0]$$

A conversion with low reference voltage affects the accuracy of the A/D Converter. The TUE of an A/D Converter that is operated at a reduced positive reference voltage can be evaluated according to the following equations:

$$TUE|_A \rightarrow TUE|_B = K \times TUE|_A, (K \geq 1)$$

$$V_{AREF}|_A \rightarrow V_{AREF}|_B = \frac{1}{K} \times V_{AREF}|_A$$

where $V_{AREF}|_A$: minimum positive reference voltage range is specified for $0\ V \leq V_{AREF} \leq V_{DDM} + 0.1\ V$

$V_{AREF}|_B$: positive reference voltage, which is below the specified range;

$TUE|_A$: total unadjusted error for reference voltages within the specified range;

$TUE|_B$: total unadjusted error for reference voltages below the specified range.

Note: All unused analog input pins must be connected to a fixed potential either V_{AGND} or $V_{AREF}[0]$ to avoid disturbance of active analog inputs.

8.1.5 Error through Overload Conditions

An additional error can occur when overloading an analog input (such as channel D). In this case, an additional leakage current exist between the analog input D and the adjacent analog inputs D ± 1, affecting the conversion result of an analog input channel D by an additional error based on the additional sampled voltage V_{AEL} (Analog Error Leakage).

$$V_{AEL}\big|_{D\pm1} = R_{AIN}\big|_{D} \times \left|I_{OV}\right|\big|_{D} \times k_A$$

The coupling factor k_A defines the physical relation of two adjacent analog inputs. The resulting TUE_L out of this behavior is given by

$$TUE_L\big|_{D\pm1} = \frac{R_{AIN}\big|_{D} \times \left|I_{OV}\right|\big|_{D} \times k_A}{V_{AREF}}$$

where V_{ARef}: reference voltage for conversion;
$R_{AIN}|_D$: resistance of the analog input channel D;
$I_{OV}|_D$: overload current of the analog input D;
$TUE|_L$: total unadjusted error caused by a leakage current;
k_A: coupling factor for the analog input D

8.1.6 Limit Checking

Limit checking provides the means to check conversion results on exceeding or becoming lower than a defined limit. The checking parameters can be configured individually for each analog channel. Service requests can be generated for each analog channel on limit checking results such as on a limit violation or on successful limit checks.

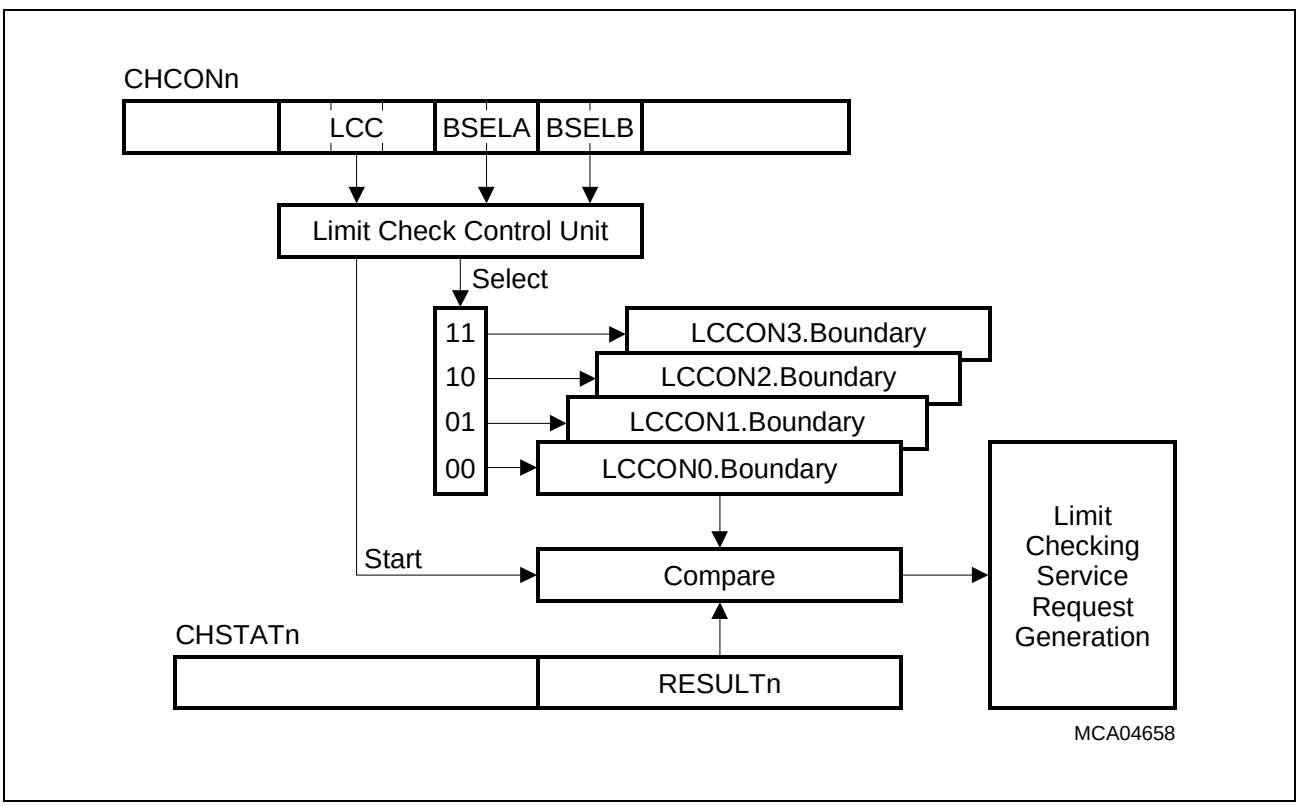


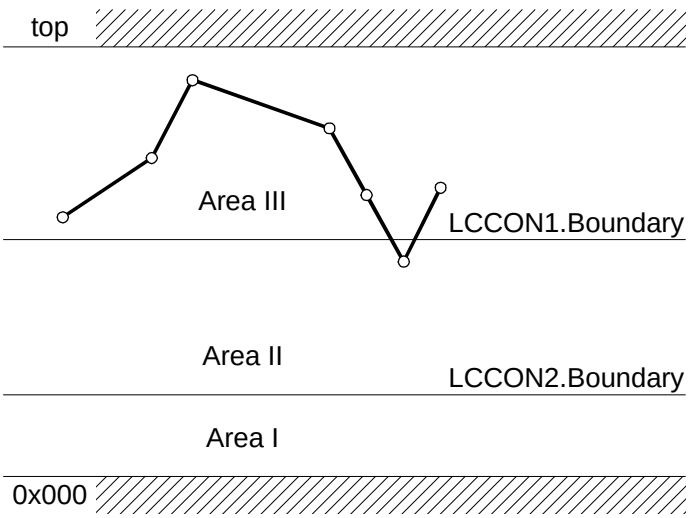
Figure 8-17 Limit Check Unit

A limit check is performed for the conversion result stored in a specific channel status register. For limit checking, the A/D Converter's measuring range is divided into three areas in order to check whether the conversion result meets the specified range or not. Two boundaries out of four must be selected and programmed per limit check. The boundaries are selected for each analog channel via the bit fields CHCONn.BSELA and CHCONn.BSELB, n = 15-0. Four boundaries can individually be set in the limit check control register LCCON0/1/2/3.

The limit check control bit field specifies whether or not a limit check is performed for the current conversion result and specifies which area must be met or avoided by the current conversion result. See **Figure 8-18**.

Depending on the selected limit check control parameter CHCONn.LCC, n = 15-0 the service request flag is not set, is set if the selected area is hit, or is set if the selected area is missed for the related conversion result A service request is only generated if the service request destination node pointer is configured and enabled.

Figure 8-18 shows the selectable parameters for limit check.



LCC	Description
0 0 0	Neither Limit Check nor Interrupt
0 0 1	In Area I
0 1 0	In Area II
0 1 1	In Area III
1 0 0	Interrupt on Write Conv. Result
1 0 1	Not in Area I
1 1 0	Not in Area II
1 1 1	Not in Area III

Figure 8-18 Limit Checking

The A/D Converter's measuring range is divided into the following three areas:

- Area I: From 000_H to (including) the lower boundary
- Area II: excluding the lower boundary to (including) the upper boundary
- Area III: excluding the upper boundary to top (top due to the selected resolution)

The value stored in LCCONn.Boundary represents a boundary, that is selected by the channel-specific bit fields CHCONn.BSELA/B. Neither boundary A (selected by CHCONn.BSELA) nor boundary B (selected by CHCONn.BSELB) is fixed in its assignment as a lower or upper one. The boundary's value specifies, whether it is assumed to be the upper or lower one.

In this example, channel number 5 is configured for limit checking. CHCON5.BSELA is set to 10_B and selects the boundary stored in LCCON2.Boundary. CHCON5.BSELB is configured to 01_B and selects the boundary stored in LCCON1.Boundary. Since the value of LCCON1.Boundary is above than the value of LCCON2.Boundary, it is assumed to be the upper one while the boundary stored in LCCON2.Boundary is the lower one.

8.1.7 Short Circuit and Broken Wire Detection

Short circuit and broken wire detection are essential for embedded control applications. The implemented short circuit or broken wire detection function indicates whether the impedance of an external sensor falls below or exceeds a defined value. A broken wire condition is defined as the interruption of a signal line providing a high-impedance to a fixed potential (R_i > 400 kΩ). A short circuit is defined as a low-impedance connection to a fixed potential (R_i < 500 Ω).

Figure 8-19 shows the ADC Module, an external sensor, and a short circuit and broken wire detection unit for one analog channel. Each analog channel has been assigned such a unit, which consists of two current sources CS0/CS1 and two switches. A typical sensor as shown below, has a source impedance R_i within the range of 1 kΩ to 200 kΩ, while the output-capacitance C_{OUT} is within the range of 10 nF to 10 µF.

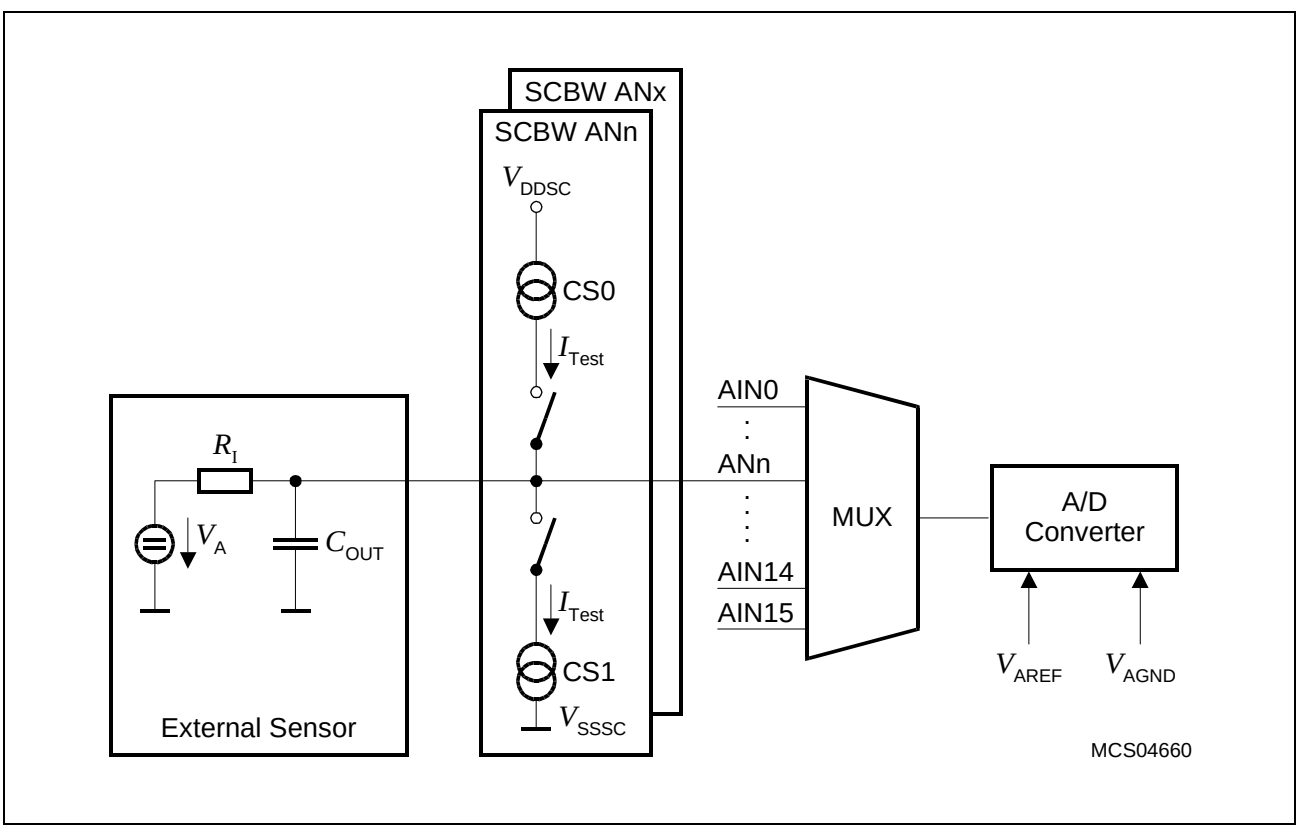


Figure 8-19 Short Circuit and Broken Wire Detection Unit

Short circuit detection and broken wire detection are controlled individually for each analog channel via the specific channel control register CHCONn. Bit CHCON.SBENn enables the short circuit and broken wire detection functionality. The current source to be activated is controlled by bit CHCON.SBCSSn, while one out of four measuring ranges is selected via bit field CHCON.SBMRn.

8.1.7.1 Performing a Short Circuit and Broken Wire Detection

A short circuit or broken wire condition is detected by performing the following steps:

1. Measure the analog voltage V_{AIN}(t = 0) at the pad without any additional test-current I_{TEST}.
2. Apply I_{TEST} due to the previously measured analog input voltage V_{AIN}(t = 0).
3. Measure the analog voltage V_{AIN}(t = 00) at the pad with additional test-current I_{TEST}, after five time-constants $t = R_i \times C_{OUT}$ have been elapsed.
4. Evaluate the previously measured analog voltages V_{AIN}(t = 0), V_{AIN}(t = 00) and the test-current I_{TEST} in order to receive the short circuit and broken wire impedance R_i according to the following equation:

$$R_i = \frac{|V_{AIN}(t = 0) - V_{AIN}(t = 00)|}{|I_{TEST}|}$$

5. The decision criteria for a **broken wire condition is R_i > 400 kΩ**, while the decision criteria for a **short circuit condition is R_i < 500 Ω**. These decision criteria are based on the tolerance of the current source.

8.1.7.2 Test Current I_{TEST} and Measurement Area Settings

Each analog channel has two associated current sources as shown in **Figure 8-19**. If activated, current source zero (CS 0) provides current source functionality, while current source one (CS 1) provides current sink functionality.

During step 2 of the broken wire and short circuit detection, CS 0 or CS 1 is activated according to the analog input voltage V_{AIN}. CS 0 must be activated if V_{AIN} is below V_{AREF}/2, while CS 1 must be activated if V_{AIN} is greater than or equal to V_{AREF}/2. V_{AREF}/2 is used to decide which current source must be enabled, because it is assumed that $V_{AREF} = V_{DDA}$ and $V_{AGnd} = V_{SSA}$.

The measuring range is selected according to the impedance R_i of the external sensor, as shown in **Table 8-9**.

Table 8-9 Measuring Area and Test Current I_{TEST} Settings

Measuring Area	Sensor's Source Impedance R_i [Ω]	Test Current I_{TEST} [mA][1)]
4	0 - 6.000	0.2666
3	0 - 36.000	0.0444
2	0 - 216.000	0.0074
1	0 - 1.296.000	0.0012

1) This column contains nominal values that can differ from the real values (see DC specification of the ADC).

8.1.8 Expansion of Analog Channels

The number of analog inputs can be expanded in a very flexible and powerful way to satisfy the increased needs for analog inputs. In principle, an external analog multiplexer might be connected to each analog channel if the following items are considered:

- Inverse current injection (overload) behavior
- ON resistance of the external multiplexer and load capacitance
- Timing of the external multiplexer
- Noise due to adjacent digital input pins

Note: The characteristics of the external multiplexers influence the accuracy of the A/D Converters. An accuracy of ± 2 LSB @ 10-bit resolution is no longer guaranteed.

Three control lines are provided to drive external multiplexer, if the external channel expansion feature is enabled via bit CON.EMUXEN, as shown in **Figure 8-20**.

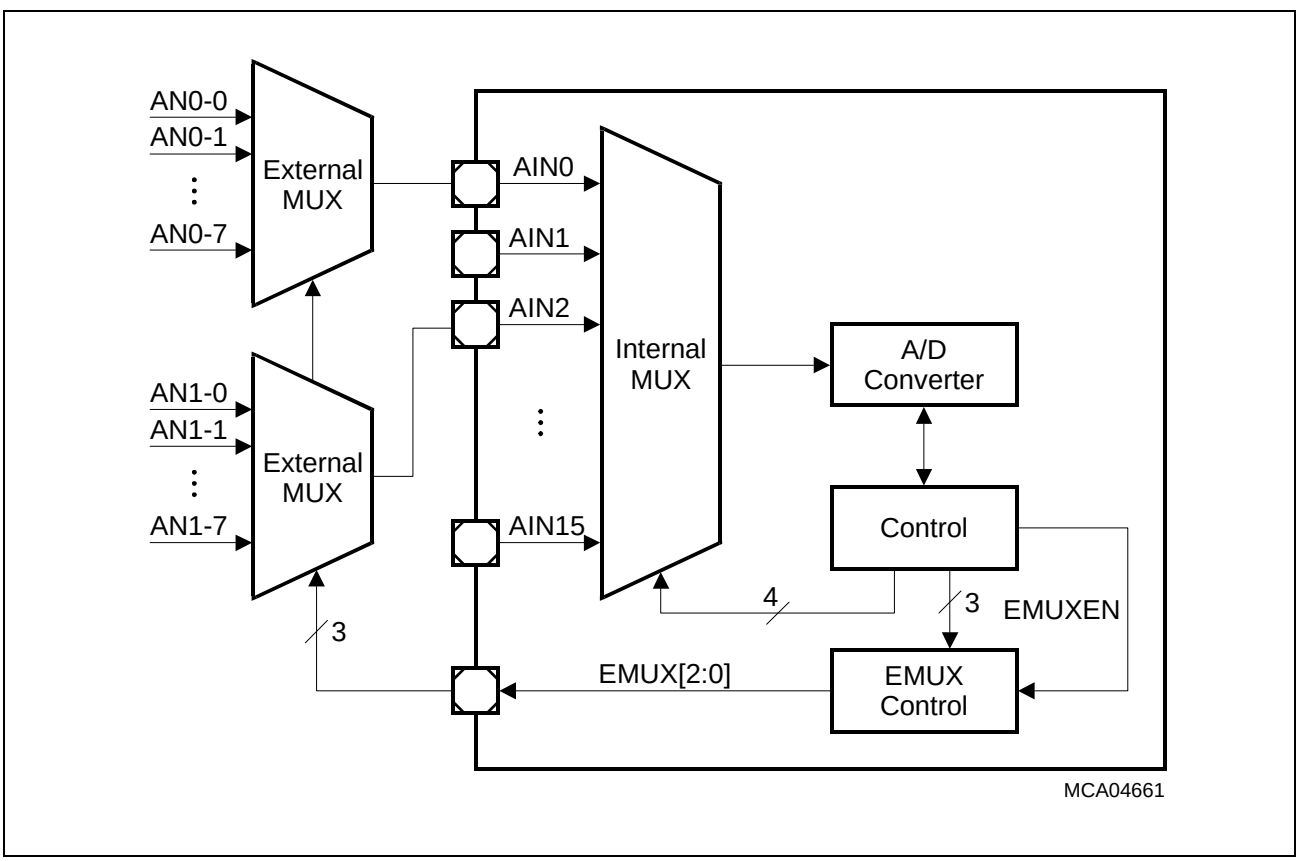


Figure 8-20 External Expansion of Analog Channels

Parallel sources receive the information to drive the external multiplexer from the channel-specific control register individually for each analog channel. Sequential sources derive the external multiplexer control information from the conversion request control register (register CHIN and QUEUE0). The external multiplexer controls from the channel-specific control registers are not taken into account for sequential sources.

8.1.8.1 Inverse Current Injection (Overload) Behavior

An overload condition occurs when the analog input voltage is above or below the supply range. An overload condition at a channel connected to an external multiplexer (such as AN0-0 in **Figure 8-20**) can affect the conversion of another channel connected to the same external multiplexer (such as AN0-1 to AN0-3). This depends on the overload capability of the external multiplexer. In case of an overload condition at one channel while another channel of the same external multiplexer is sampled by the A/D Converter, an even higher conversion error must be expected.

Note: The overload behavior of every channel that is directly connected to the internal multiplexer or through another external multiplexer does not change.

8.1.8.2 On Resistance of the External Multiplexer

If an external multiplexer is connected to an analog input channel, a typical application might add RC filter before the external multiplexer to each additional external analog inputs (for example, each of the external analog inputs AN0-0 to AN0-3 in **Figure 8-20** is adapted by a RC filter). In this case the resistance of the external multiplexer reduces the efficiency of the external capacitors of the RC filter. An additional blocking capacitor between the external multiplexer and the pad of the analog input could improve the noise suppression capability. However, in this case, the capacitance, that must be charged, would be increased by the size of the blocking capacitor.

8.1.8.3 Timing of the External Multiplexer

An analog input channel of an external analog multiplexer is selected after the arbitration round is finished. Therefore, the information to drive an external multiplexer is available before the sample time begins.

8.1.8.4 Load Capacitance

Because each analog input of the external multiplexer might be applied by different analog voltages (e.g. AN0-0 = 4 V, AN0-1 = 1 V), the total input capacitance of the A/D Converter must be recharged within the sample time, each time that an analog input channel of an external multiplexer is measured.

For analog input channels, that are directly applied to the analog input pin of the A/D Converter (such as AN2 to AN15 of **Figure 8-20**) the input capacitance does not change. The analog voltage source of such channels must solely recharge the switched input capacitance of the A/D Converter.

8.1.9 Service Request Processing

A fully configurable and very flexible service request control structure is implemented in the A/D Converter Module. The main part of the service request structure are the service request sources, the module service request status flags (MSS-Flag), the service request node pointer with destination bit field and enable bit, and the four A/D Converter module specific Service Request Nodes. **Table 8-10** lists all A/D Converter module based service request sources and its control and status blocks.

The A/D Converter Module provides the service request compressor logic to assign service request sources to Service Request Nodes, as described in **Chapter 8.1.9.1**.

Table 8-10 Service Request Control Structure

ADC Module based Service Request Source (SR-Source)	Module Service Request Status Flag (MSS-Flag)	SRN-Pointer Destination Bit field	SRN-Pointer Enable Bits/Bit field
Write result to CHSTAT0 or Limit checking of channel 0	MSS0.MSRCH0	CHCON0.PCH	CHCON0.LCC
	:	:	:
Write result to CHSTAT15 or Limit checking of channel 15	MSS0.MSRCH15	CHCON15.PCH	CHCON15.LCC
Timer	MSS1.MSRT	SRNP.PT	SRNP.ENPT
Queue	MSS1.MSRQR	SRNP.PQR	SRNP.ENPQR
Auto-scan	MSS1.MSRAS	SRNP.PAS	SRNP.ENPAS
Synchronization Injection	MSS1.MSRSY	SRNP.PSY	SRNP.ENPSY

8.1.9.1 Service Request Compressor

The A/D Converter Module is equipped with 20 service request sources and four Service Request Nodes. Each service request source can be allocated independently to one of the four module specific Service Request Nodes. A request compressor condenses these 20 sources to the four Service Request Nodes reporting the service requests of the A/D Converter module to the interrupt controller.

Each request source is provided with an “Service Request Node Pointer” and an “Enable Bit”, in order to increase flexibility in interrupt processing. The destination bit field determines which A/D Converter module related service request node is triggered by the associated service request source, while the enable bit is used to enable/disable this connection. Each of the four Service Request Nodes can trigger an independent service request routine with its own service request vector and its own priority.

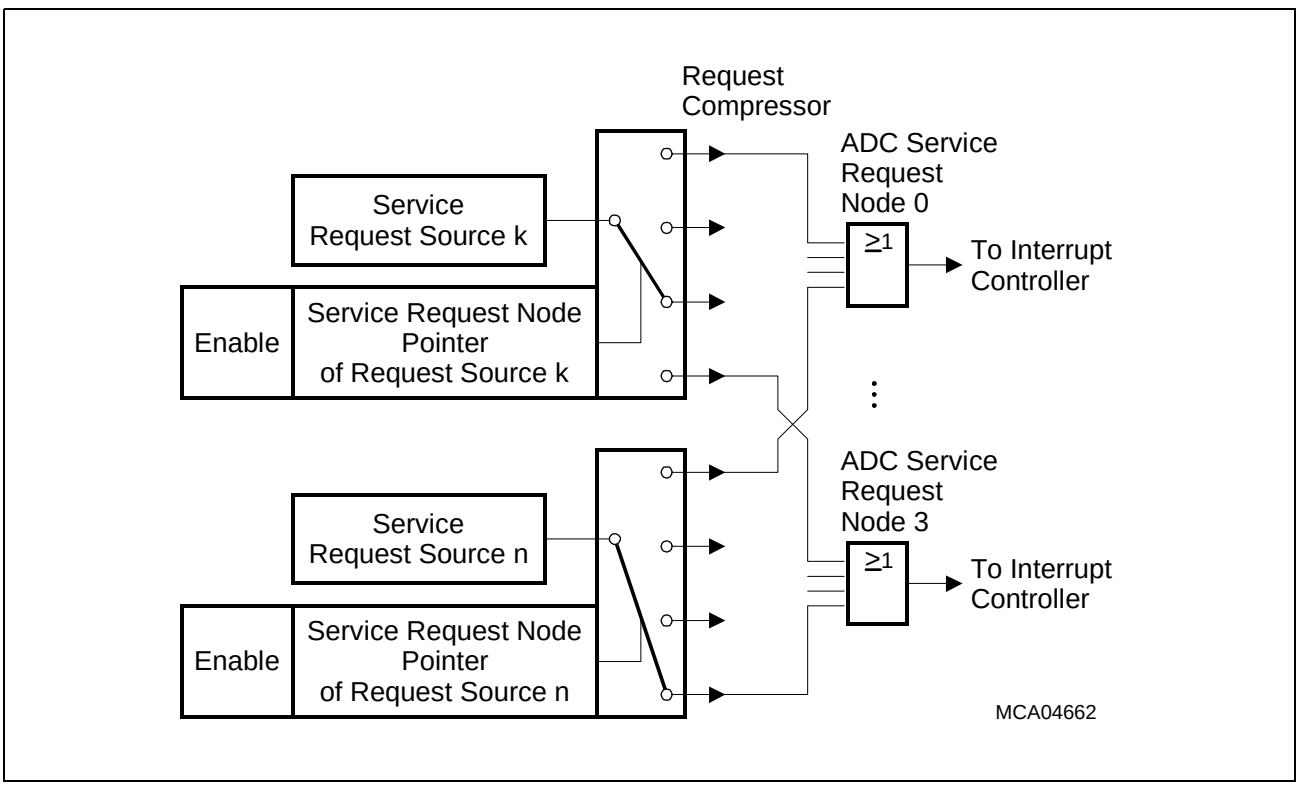


Figure 8-21 Service Request Node Pointer and Compressor

8.1.9.2 Module Service Request Status Flags

Figure 8-22 shows the analog channel-specific control logic, which is responsible to select the trigger cause to set the associated module service request flag MSS0.MSRCHn, n = 15-0.

Bit field CHCONn.LCC selects whether flag MSS0.MSRCHn

- is never set on any action to CHSTATn.RESULT
- is set on a limit violation
- is set on a write action to CHSTATn.RESULT (no limit checking performed)

Flag MSS0.MSRCHn can be used for polling on channel-specific actions. If polling functionality is required, the interrupt node pointer must be disabled by bit CHCONn.ENPCH.

The interrupt node pointer associated with the analog channels are controlled by enable bit CHCONn.ENPCH and selection bit field CHCONn.PCH.

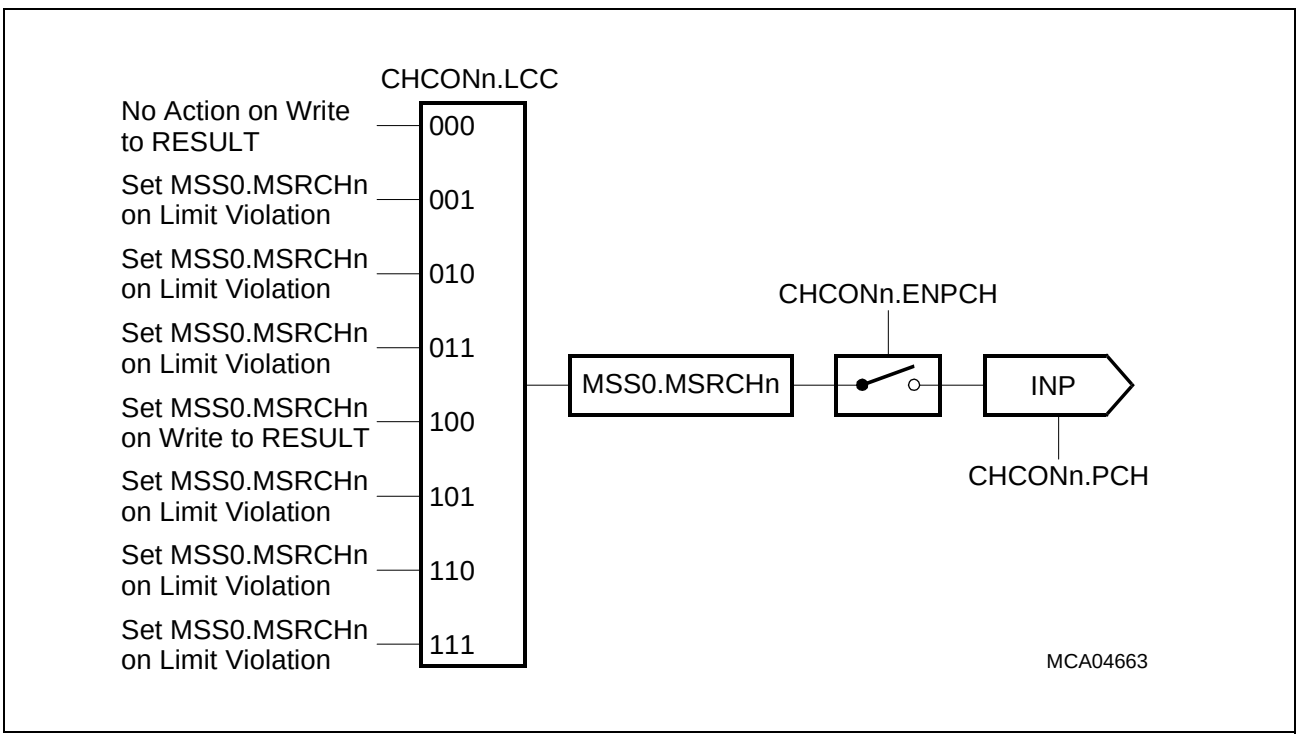


Figure 8-22 Module Service Request Status Flags

8.1.9.3 Service Request Source and Service Request Test Mode

Each event generated by a service request source sets the corresponding MSS-Flag and also sends a trigger to the service request compressor. The module service request status flags are collected in the global module service request status registers MSS0/ MSS1. Because the state of all module service request status flags are combined within these two registers, exceptions can be handled by software via poling on registers MSS0/MSS1 while the generation of service request is disabled. **Figure 8-23** shows the scheme of a service request source.

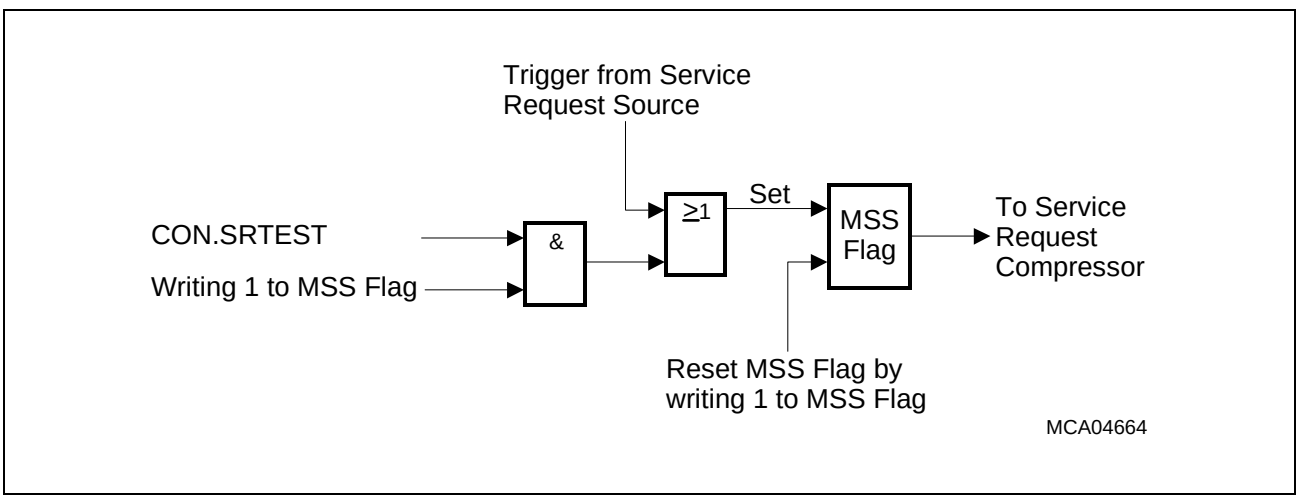


Figure 8-23 Concept of Service Request Sources

A MSS-Flag must be reset under software control by writing a 1 to the bit position in the corresponding MSS0/MSS1 register. This write action is taken into account only if the MSS-Flag is set. If a MSS-Flag is set and a reset condition occurs in the same clock cycle as an new set condition, a new service request is generated and the MSS-Flag remains set.

In Service Request Test Mode, service requests can be triggered under software control additionally to the hardware trigger input. In Test Mode, MSS-Flags can additionally be set if bit CON.SRTEST is set and 1 is written to a MSS-Flag in consecutive. After a write action is performed to register MSS0/MSS1 (writing to a MSS-Flag), bit CON.SRTEST is automatically reset.

The following table summarizes the actions to be performed after a write action on a MSS-Flag depending on the service request test mode.

Table 8-11 Module Service Request Status Flags

SR-Test Mode CON.SRTEST	MSS-Flag current value	Write action to MSS-Flag	Result of write action MSS-Flag	Comment
0	0	0	0	no action
	0	1	0	write not permitted (CON.SRTEST = 0)
	1	0	1	no action
	1	1	0	reset MSS-Flag
1	0	0	0	no action
	0	1	1	set MSS-Flag by software, testmode
	1	0	1	no action
	1	1	1	no new service request generated

8.1.10 Synchronization of Two ADC Modules

To synchronize conversions in two ADC Modules, a synchronization logic is implemented in each module. A handshake mechanism guarantees the synchronization between both ADCs without additional CPU load. As shown in **Figure 8-24**, both modules have an identical structure. Neither Module 0 nor Module 1 has a fixed assignment as master or slave. Because each module can request to be master, an ingenious synchronization and handshake mechanism guarantees a proper master-slave coordination.

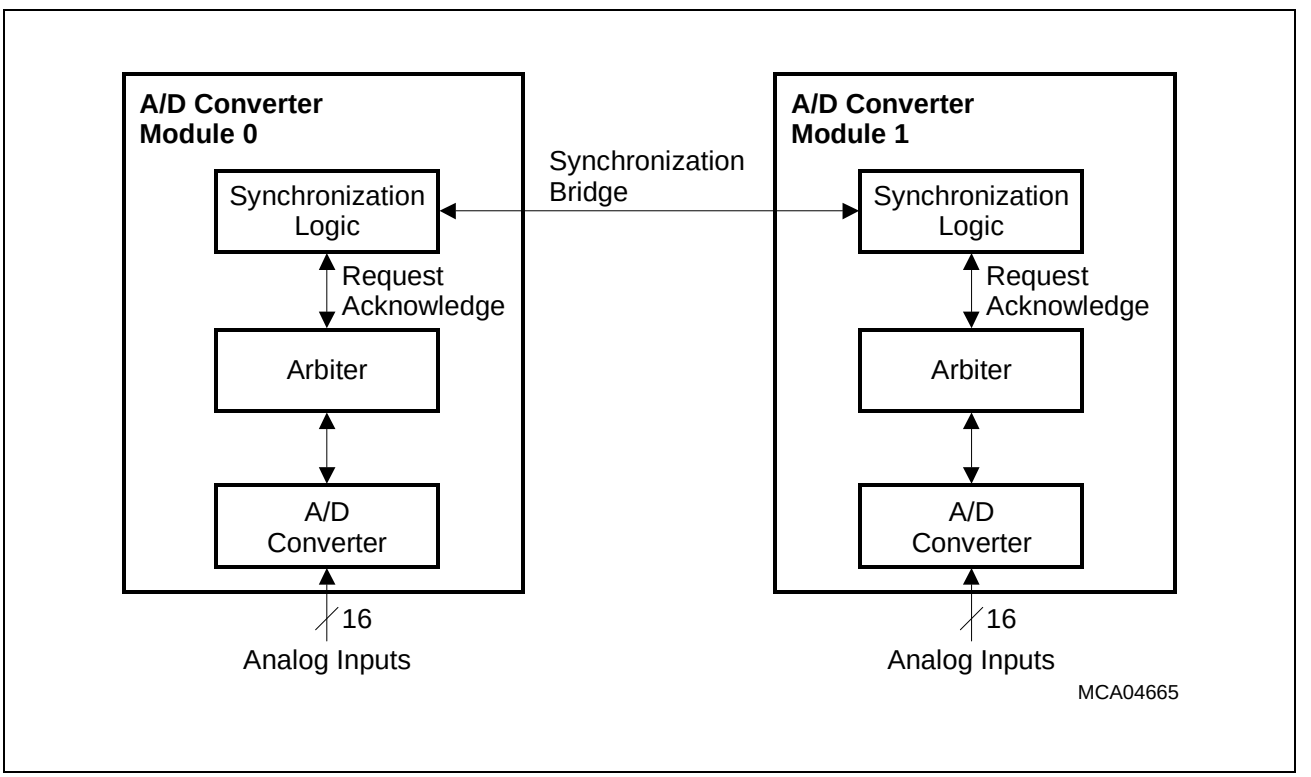


Figure 8-24 Synchronization of Two A/D-Converter

Each ADC Module provides a synchronized injection status register SYSTAT. The conversion request and the control information for a synchronized conversion is always driven by the initiating ADC Module, which is referred to as master. Because the master transfers all control information necessary for the synchronized conversion in the slave, the channel number, the A/D Converter resolution, the external multiplexer information and the cancel-synchronize-repeat information are identical in both modules. The timing information as well as the service request generation can be different in both modules (for instance, a synchronized conversion is started in both ADC Modules for channel number CH5, 12-bit resolution, the identical external multiplexer information). Note that the cancel-synchronize-repeat information is needed only in the slave module in order to determine whether or not a currently running conversion is cancelled.

The control bits of register SYSTAT retain their values if the synchronized conversion is started except that the request bit is automatically reset on a start of the synchronized

conversion. The contents of register SYSTAT will be overwritten if a new synchronized conversion is requested. Status flags indicate the state (master and/or slave) of the ADC Module during the synchronized conversion.

8.1.10.1 Synchronized Injection Mode (SYM)

The Synchronized Injection Mode is controlled by bit field CHCONn.SYM in the ADC Module channel-specific control register. A synchronized conversion is always initiated by an analog channel operating in Synchronized Injection Mode instead of a normal request. The initiating ADC Module is named as master, while the participating ADC Module is named as slave.

To initiate a synchronized conversion in both ADC Modules, the analog channel configured for Synchronized Injection Mode must be triggered by any source. If it wins the arbitration, a synchronized conversion is initiated in both ADC Modules. This means that the control information needed for a synchronized conversion is transferred from the master to the slave (for instance, channel number CH2 of ADC Module 0 is configured for Synchronized Injection Mode with sync-wait feature selected; then, each time this channel is triggered and wins the arbitration, a synchronized conversion is requested. Thus, ADC Module 0 is assumed to be the master and the control information needed for a synchronized conversion is transferred to ADC Module 1 the slave).

Note: A Channel Injection request with an active cancel-inject-repeat feature that is requesting a Synchronized Injection doesn't cancel a running conversion in the master. A Channel Injection request with an active cancel-inject-repeat feature doesn't automatically set the cancel-sync-repeat mode in the Synchronized Injection.

The Synchronized Injection Mode provides two features:

- Sync-wait (CHCONn.SYM = 01_B), the synchronized start of the conversion is delayed until the currently performed conversion in the partner (slave) and the initiating ADC Module (master) is terminated.
- Cancel-sync-repeat (CHCONn.SYM = 10_B), this feature provides the ability to cancel a conversion that is currently performed in the partner ADC Module (slave). Thus, the synchronized conversion is immediately started after a currently performed conversion in the initiating ADC Module is terminated. Because a conversion is cancelled in the partner ADC Module (slave), the control information is restored and will participate in arbitration again.

8.1.10.2 Status Information During Synchronized Conversion

Each ADC Module provides three specific status bits in register STAT that display the status of the ADC Module during a Synchronized Injection.

Master Status: Bit STAT.REQSY is set in the initiating ADC (master) module during a synchronized conversion. It is set at the start of the synchronized conversion and is reset after this synchronized conversion is finished.

Slave Status: Bit STAT.PARSY is set in the partner ADC Module (slave) during a synchronized conversion. It is set at the start of the synchronized conversion and is reset after this synchronized conversion is finished.

Master/Slave Status: Bit STAT.SYMS is set in **both** ADC Modules, to indicate that both ADC Modules requested a synchronized conversion at the same time with identical channel number. Bit STAT.SYMS is automatically reset at the generation of the synchronized service request.

8.1.10.3 Master-Slave Functionality for Synchronized Injection

Each ADC Module can operate either as master or slave or both. The ADC Module operating functionality for Synchronized Injection (master, slave or master/slave functionality) is automatically detected. All associated controls for synchronized conversion are shown in the table below:

Table 8-12 Master-Slave Functionality and Control

Functionality during Sync. Conversion	Controls	Description
Master	CHCONn.SYM	Selects either sync-wait or cancel-sync-repeat feature
	STAT.REQSY	Status bit indicating master functionality
	STAT.IENREQ	Status bit is driven by master to indicate that the master finished its synchronized conversion
	STAT.IENPAR	Status bit is driven by slave to indicate that the slave finished its synchronized conversion

Table 8-12 Master-Slave Functionality and Control (cont'd)

Functionality during Sync. Conversion	Controls	Description
Slave	SYSTAT.SYREQ	Status bit is driven by master to request the slave for a synchronized conversion
	SYSTAT.CHNRSY	Status bit field is driven by master to indicate the channel to be converted for a synchronized conversion
	SYSTAT.RES	Status bit field is driven by master to indicate the resolution for a synchronized conversion
	SYSTAT.EMUX	Status bit field is driven by master to indicate the external multiplexer control info for a synchronized conversion
	SYSTAT.CSREN	Status bit is driven by master to indicate whether sync-wait or cancel-sync-repeat feature was selected in the master
Master/Slave	STAT.SYMS	Status bit to indicate that both modules requested a synchronized conversion at the same time for the same channel

Master Functionality

After an arbitration winner is detected, the Synchronized Injection Mode bit field CHCONn.SYM in the corresponding channel-specific control register is evaluated. If this bit field is configured either for sync-wait (CHCONn_SYM = 01_B) or cancel-sync-repeat (CHCONn.SYM = 10_B) functionality, a synchronized-request is generated for the partner (slave) ADC Module. A synchronized request means setting bit SYSTAT.SYREQ in the slave's register.

In addition to this synchronized-request, the channel number (SYSTAT.CHNRSY), the resolution (SYSTAT.RES), the external multiplexer information (SYSTAT.EMUX), and the cancel-sync-repeat information (SYSTAT.CSREN) is transferred to the slave.

Then the master ADC Module waits for the acknowledge of the slave. This indicates that both ADC Modules are ready to start their synchronized conversion. At reception of this acknowledge, the synchronized conversion is started and bit STAT.REQSY is set. Bit STAT.REQSY indicates that a synchronized conversion is currently performed and this ADC Module provides master functionality. After the currently performed synchronized conversion is completely finished, bit STAT.REQSY is reset and bit STAT.IENREQ is set.

Bits STAT.IENREQ and STAT.IENPAR are used for service request generation in the master ADC Module. In order to generate a service request after both ADC Modules have finished their synchronized conversion, the master checks bit STAT.IENPAR, which is driven by the slave. In case both ADC Modules have finished their conversion, bit STAT.IENREQ and STAT.IENPAR are set. This generates a service request (bit MSS1.MSRSY is set). As well as setting bit MSS1.MSRSY, both bits STAT.IENREQ and STAT.IENPAR are automatically reset.

Slave Functionality

On reception of the synchronized request (bit SYSTAT.SYREQ is set), the channel number (SYSTAT.CHNRSY), the resolution (SYSTAT.RES), the external multiplexer information (SYSTAT.EMUX), as well as the cancel-sync-repeat information (SYSTAT.CSREN) are driven by the master. Beside this synchronized request derived from the master, **the evaluation of an arbitration result of the slave is disabled**. Thus, the slave itself cannot generate a request.

Then, the cancel-sync-repeat enable bit is evaluated. This bit specifies whether a conversion is cancelled (SYSTAT.CSREN = 1) or not (SYSTAT.CSREN = 0) that is currently performed in the slave. Note that a synchronized conversion cannot be cancelled by another synchronized conversion. Bit STAT.REQSY is set to indicate that this module is the partner (slave) in a synchronized conversion.

The handshake guarantees that the master and the slave are ready to start a synchronized conversion if the synchronized request is still active (bit SYSTAT.SYREQ is set in the slave). In the case that bit SYSTAT.SYREQ is reset in the meantime by the master, no synchronized conversion must be started and the ADC Module continues to with normal behavior.

Beside the start of conversion, the synchronized request (bit SYSTAT.SYREQ) is reset, bit STAT.PARSY is set, and the write to the arbitration result is enabled anew. At the end of the synchronized conversion, master’s status bit STAT.IENPAR is driven by the slave and the slave’s status bit STAT.PARSY is reset.

Master/Slave Functionality

The special master/slave mode is entered, if both ADC Modules requested to be master at the same time **and** both ADC Modules requested a synchronized conversion for the same channel. In this case, each ADC Module compares the received channel number from the synchronization bridge with the channel number stored in their arbitration result. Three cases must be treated:

1. SYSTAT.CHNRSY < channel number in arbitration result register
   ADC Module behaves as master.
   Reset the synchronized conversion request (bit SYSTAT.SYREQ) because this is the master (see description on master functionality).

2. SYSTAT.CHNRSY = channel number in arbitration result
   ADC Module provide master/slave functionality.
3. SYSTAT.CHNRSY > channel number in arbitration result
   ADC Module behaves as slave and bit SYSTAT.SYREQ remains set.
   (see description on slave functionality)

In case that this ADC Module provides master/slave functionality, bit STAT.SYMS is set and any write action to the arbitration result is disabled. This means that the synchronized conversion is started next in the slave.

From this point, the behavior is similar to the one of a master until the synchronized conversion is finished. At the end of the synchronized conversion, bit STAT.SYMS is reset and bit MSS1.MSRSY is set for each ADC Module.

8.1.10.4 Conversion Timing during Synchronized Conversion

The settings for the conversion and sample timing can be selected individually for each ADC Module. Thus, the conversions are started synchronous but the master can finish its synchronized conversion at a different time than the slave.

8.1.10.5 Service Request Generation in Synchronized Injection

The Synchronized Injection based service request is automatically generated either in the master ADC Module or in each ADC Module while each provides master/slave functionality.

In the case that both ADC Modules have finished their conversion, bit STAT.IENREQ and STAT.IENPAR are set in the master ADC Module. This generates a service request (bit STAT.SRSY is set). Beside setting bit STAT.SRSY, bits STAT.IENREQ and STAT.IENPAR are automatically reset.

A service request can be generated in both ADC Modules for the converted channel if the channel-specific service request node pointer is configured and enabled.

8.1.10.6 Example for Synchronized Injection

Figure 8-25 shows the Synchronized Injection Mode with sync-wait functionality. The start of the synchronized conversion is always delayed until the currently performed conversion in the slave ADC Module is finished.

In this example, channel 5 is the arbitration winner. Its CHCON5.SYM bit field is configured for Synchronized Injection with sync-wait functionality (CHCON5.SYM = 01_B). Thus, a synchronized request is transferred to the slave, causing the slave's bit SYSTAT.SYREQ to be set. This immediately locks the slaves's arbiter until the synchronized conversion is started. Any pending conversion requests in the slave (in this case the request by source "i") are served after the synchronized conversion is finished.

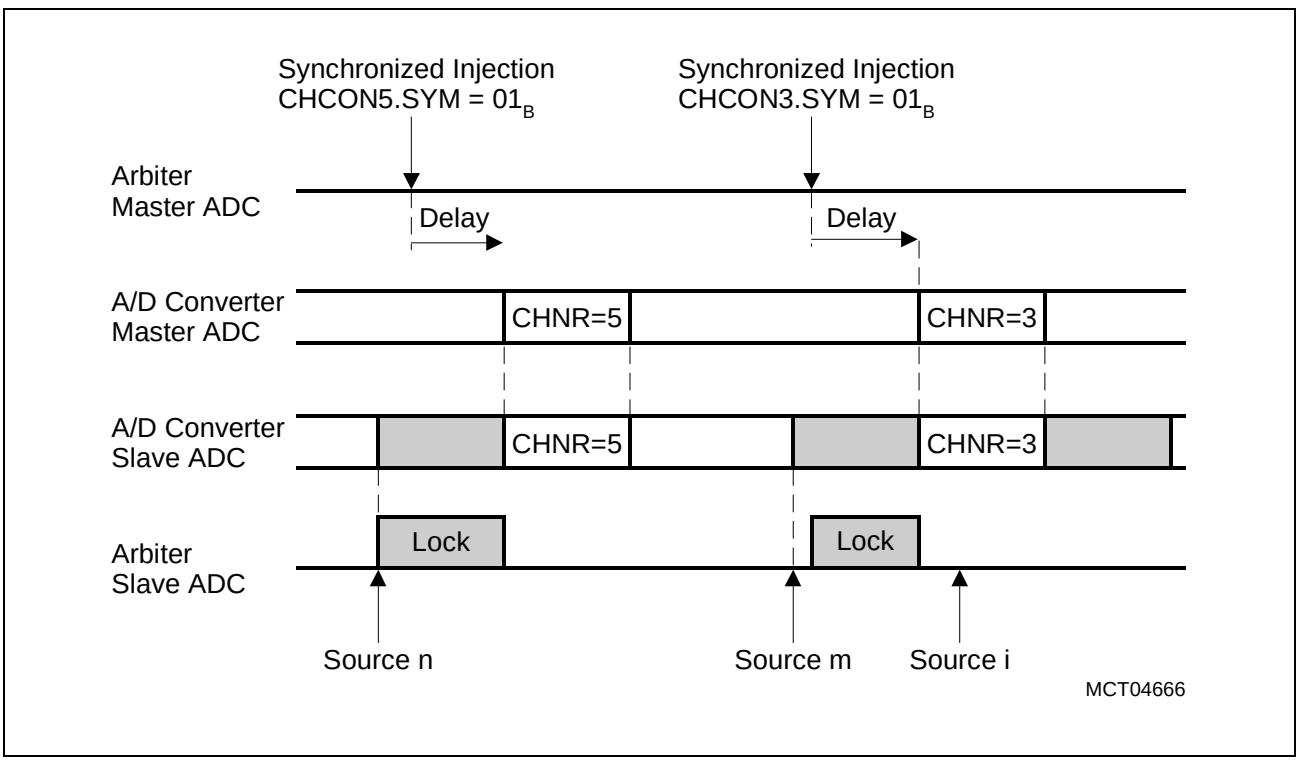


Figure 8-25 Synchronized Injection with Sync-Wait Functionality

Figure 8-26 shows the Synchronized Injection Mode with cancel-sync-repeat functionality. Currently performed conversions in the slave will always be cancelled, independent to their source arbitration levels. Note that a currently running synchronized conversion cannot be cancelled by any other source, not even by a new request for synchronized conversion. Thus, a request for a synchronized conversion will be delayed until the currently running synchronized conversion is finished.

In this example, channel 5 is the arbitration winner. Its CHCON5.SYM bit field is configured for Synchronized Injection with cancel-sync-repeat functionality (CHCON5.SYM = 10_B). Thus, a synchronized request is transferred to the slave and the currently performed conversion is immediately cancelled.

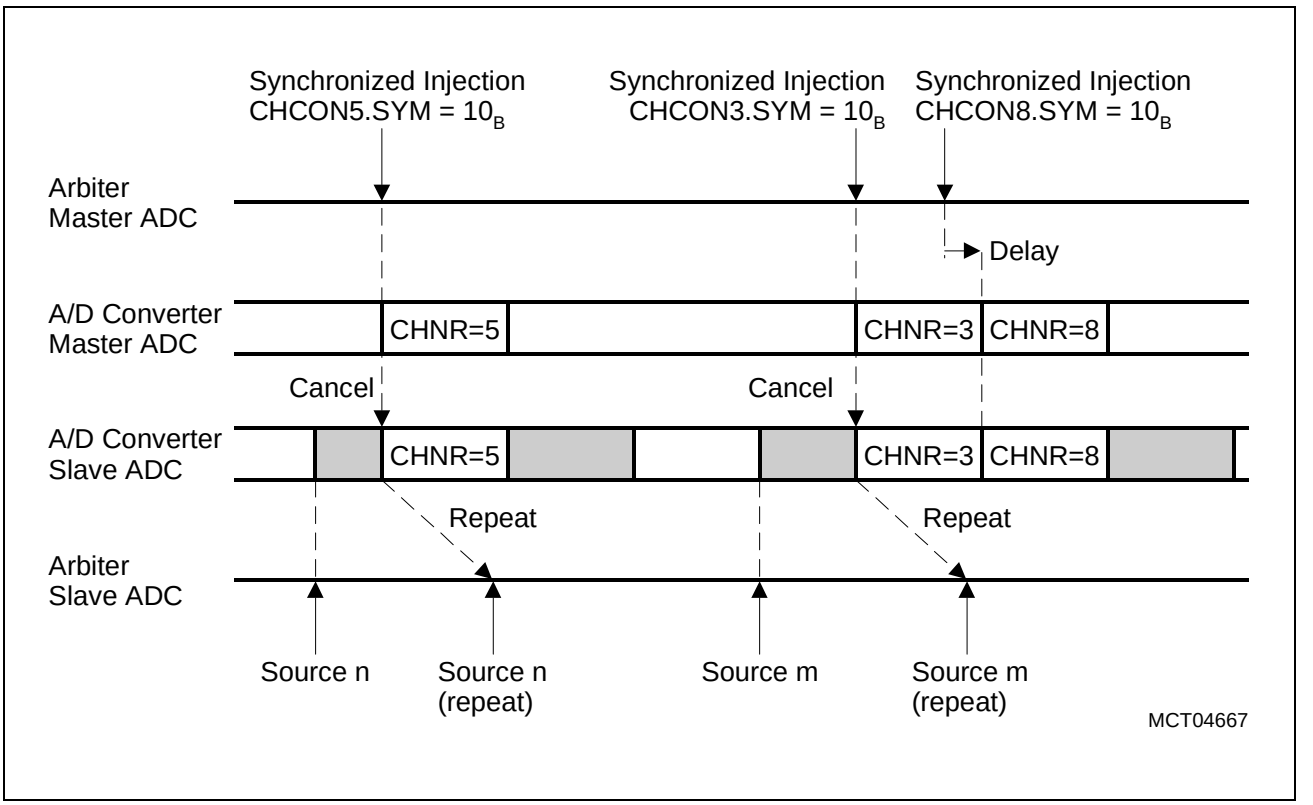


Figure 8-26 Synchronized Conversions with Cancel-Sync-Repeat Functionality

8.2 ADC Kernel Registers

The ADC kernel registers can be divided into two types of register, as shown in Figure 8-27.

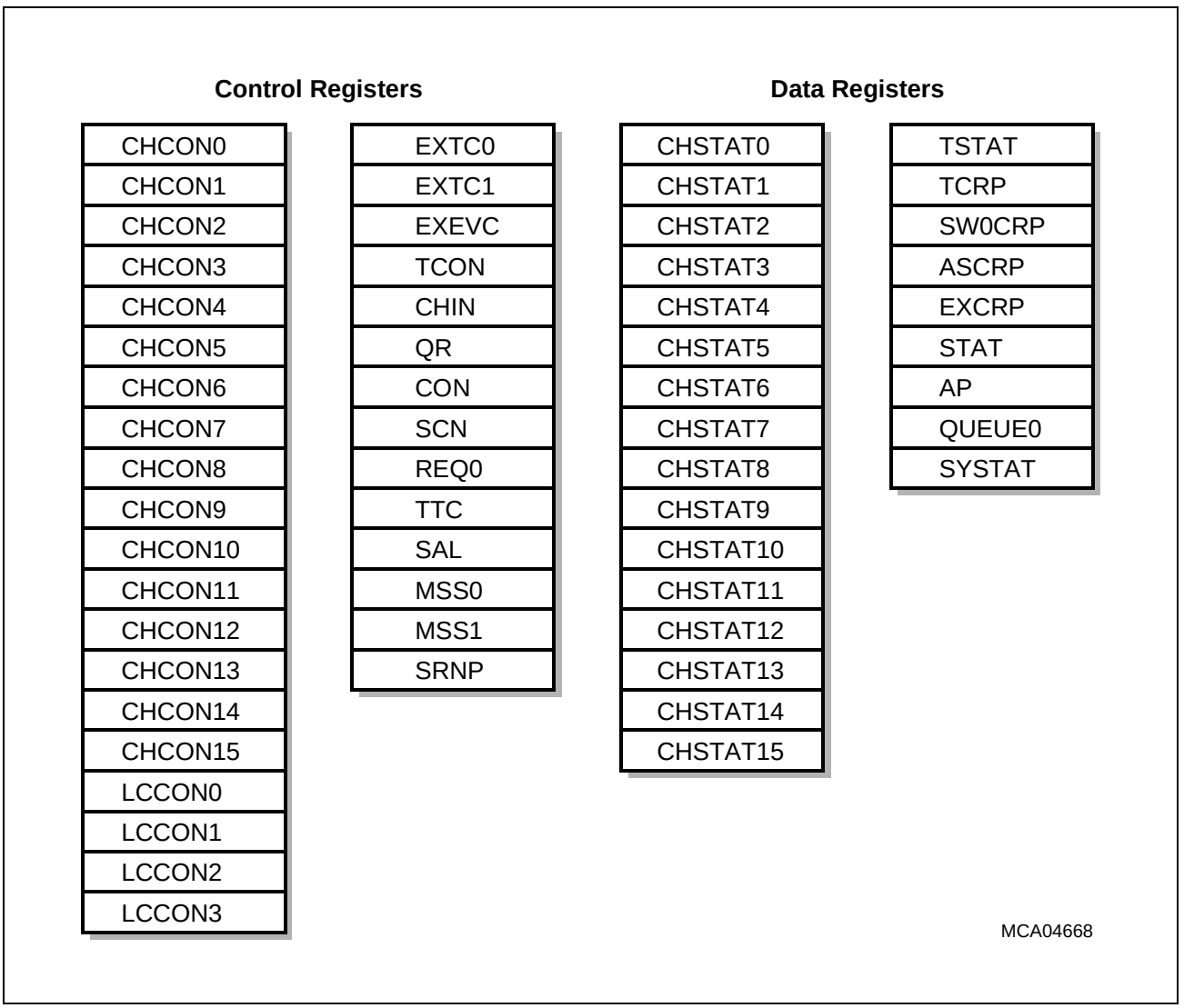


Figure 8-27 SFRs associated with the ADC

Table 8-13 ADC Kernel Registers

Register Short Name	Register Long Name	Offset Address	Description see
CHCONn	Channel Control Register n (n = 0-15)	$0010_H + n \times 4_H$	Page 8-57
EXEVC	External Event Control Register	0080_H	Page 8-60
AP	Arbitration Participation Register	0084_H	Page 8-61
SAL	Source Arbitration Level Register	0088_H	Page 8-62
TTC	Timer Trigger Control Register	$008C_H$	Page 8-63
EXTC0	External Trigger Control Register 0	0090_H	Page 8-65
EXTC1	External Trigger Control Register 1	0094_H	Page 8-65
LCCON0	Limit Check Control Register 0	0100_H	Page 8-64
LCCON1	Limit Check Control Register 1	0104_H	Page 8-64
LCCON2	Limit Check Control Register 2	0108_H	Page 8-64
LCCON3	Limit Check Control Register 3	$010C_H$	Page 8-64
TCON	Timer Control Register	0114_H	Page 8-66
CHIN	Channel Injection Register	0118_H	Page 8-67
QR	Queue Register	$011C_H$	Page 8-69
CON	Converter Control Register	0120_H	Page 8-70
SCN	Auto Scan Control Register	0124_H	Page 8-72
REQ0	Conversion Request Register SW0	0128_H	Page 8-73
CHSTATn	Channel Status Register n (n = 0-15)	$0130_H + n \times 4_H$	Page 8-74
QUEUE0	Queue Status Register	0170_H	Page 8-75
SW0CRP	Software SW0 Conv. Req. Pending Register	0180_H	Page 8-76
ASCRP	Auto Scan Conversion Req. Pending Register	0188_H	Page 8-77
SYSTAT	Synchronization Status Register	0190_H	Page 8-78
TSTAT	Timer Status Register	$01B0_H$	Page 8-79
STAT	Converter Status Register	$01B4_H$	Page 8-80
TCRP	Timer Conversion Req. Pending Register	$01B8_H$	Page 8-83
EXCRP	External Conversion Req. Pending Register	$01BC_H$	Page 8-84
MSS0	ADC0 Service Request Status Register 0	$01D0_H$	Page 8-85

Table 8-13 ADC Kernel Registers (cont'd)

Register Short Name	Register Long Name	Offset Address	Description see
MSS1	ADC0 Service Request Status Register 1	$01D4_H$	Page 8-86
SRNP	ADC0 Service Request Node Pointer Register	$01DC_H$	Page 8-87

CHCONn, n = 15-0
Channel Control Register **Reset Value: XXXX XXXX$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PCH		SYM		SB EN	SB CSS	SBMR		EN PCH	LCC			BSELA		BSELB	
rw		rw		rw	rw	rw		rw	rw			rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	EMUX			RES		REF		STC							
r	rw			rw		rw		rw							

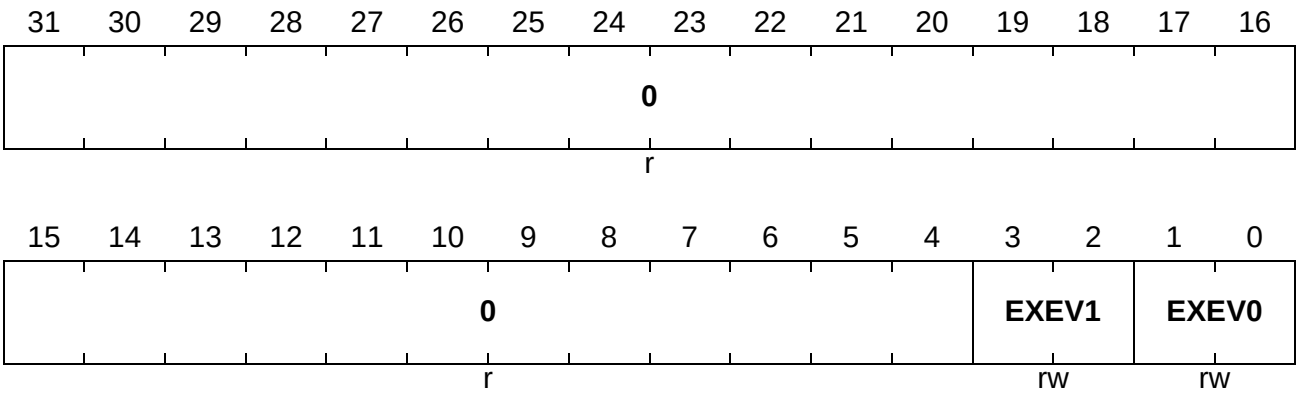
Field	Bits	Type	Description
STC	[7:0]	rw	**Sample Time Control** Defines the duration of the sample phase for this channel. Any modification of this bit field is taken into account after the currently running conversion is finished.
REF	[9:8]	rw	**Analog Reference Voltage Control** Defines the reference voltage for this channel. 00 V_{AREF}[0] == V_{AREF} 01 V_{AREF}[1] == AIN0 10 V_{AREF}[2] == AIN1 11 V_{AREF}[3] == AIN2
RES	[11:10]	rw	**Conversion Resolution Control** Controls the resolution of the A/D Converter for the conversion of this channel. Any modification of this bit field is taken into account after the currently running conversion is finished. 00 10-bit resolution 01 12-bit resolution 10 8-bit resolution 11 Reserved

Field	Bits	Type	Description
EMUX	[14:12]	rw	**External Multiplexer Control** Drives an external multiplexer connected to this analog input channel. *Note: See also the external multiplexer enable bit CON.EMUXEN.*
0	15	–	**Reserved**; read as 0; should be written with 0.
BSELA **BSELB**	[17:16] [19:18]	rw	**Boundary Select Control** Selects two limit check control registers for limit checking. 00 LCCON0 (BOUNDARY0) is selected 01 LCCON1 (BOUNDARY1) is selected 10 LCCON2 (BOUNDARY2) is selected 11 LCCON3 (BOUNDARY3) is selected *Note: see also LCC.*
LCC	[22:20]	rw	**Limit Check Control** 000_B Neither limit check is performed nor an interrupt is generated on write of conversion result to bit field STAT.RESULT. 001_B Check if conversion result is in area I and generate service request in case of limit violation. 010_B Check if conversion result is in area II and generate service request in case of limit violation. 011_B Check if conversion result is in area III and generate service request in case of limit violation. 100_B Generate interrupt on write of conversion result to bit field STAT.RESULT. 101_B Check if conversion result is **not** in area I and generate service request in case of limit violation. 110_B Check if conversion result is **not** in area II and generate service request in case of limit violation. 111_B Check if conversion result is **not** in area III and generate service request in case of limit violation.
ENPCH	23	rw	**Service Request Node Pointer Enable** 0 Service Request Node Pointer is disabled 1 Service Request Node Pointer is enabled

Field	Bits	Type	Description
SBMR	[25:24]	rw	**Short Circuit and Broken Wire Measuring Range Selection** 00 Measuring Range 1 is selected 01 Measuring Range 2 is selected 10 Measuring Range 3 is selected 11 Measuring Range 4 is selected
SBCSS	26	rw	**Short Circuit and Broken Wire Current Source Selection** 0 Current source CS0 is activated 1 Current source CS1 is activated
SBEN	27	rw	**Short Circuit and Broken Wire Detection Enable** 0 Short circuit and broken wire detection is disabled 1 Short circuit and broken wire detection is enabled
SYM	[29:28]	rw	**Synchronized Injection Mode** 00 Synchronized conversion are disabled for this analog channel 01 Synchronized conversions and sync-wait functionality is selected for this channel 10 Synchronized conversion and cancel-sync-repeat functionality is selected for this channel 11 Reserved
PCH	[31:30]	rw	**Service Request Node Pointer Destination** Directs a Service Request Source trigger to one out of four Service Request Nodes. 00 Directed to Service Request Node Pointer 0 01 Directed to Service Request Node Pointer 1 10 Directed to Service Request Node Pointer 2 11 Directed to Service Request Node Pointer 3

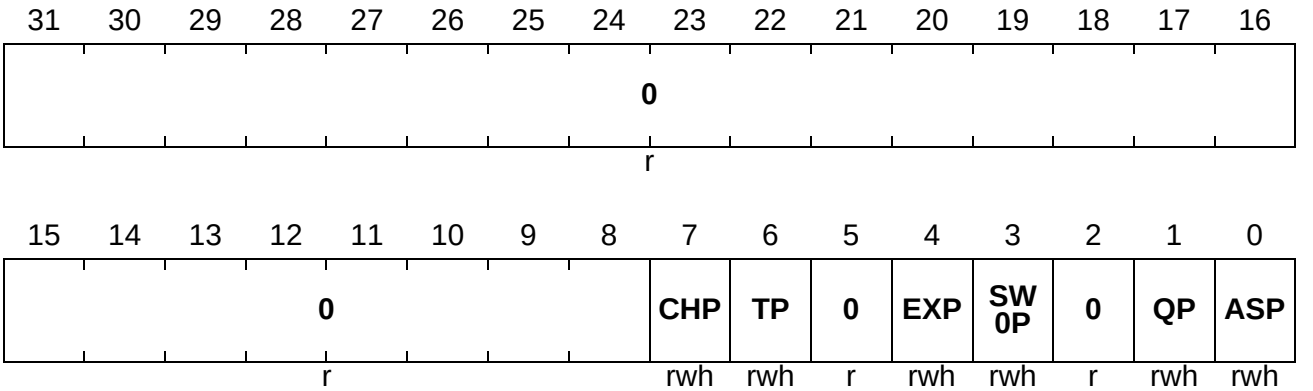
EXEVC
External Event Control Register

Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
EXEV1 **EXEV0**	[3:2] [1:0]	rw	**External Event Trigger Control** 00 Edge detection disabled 01 Detection of falling edges 10 Detection of rising edges 11 Detection of falling and rising edges
0	[31:4]	r	**Reserved**; read as 0; should be written with 0.

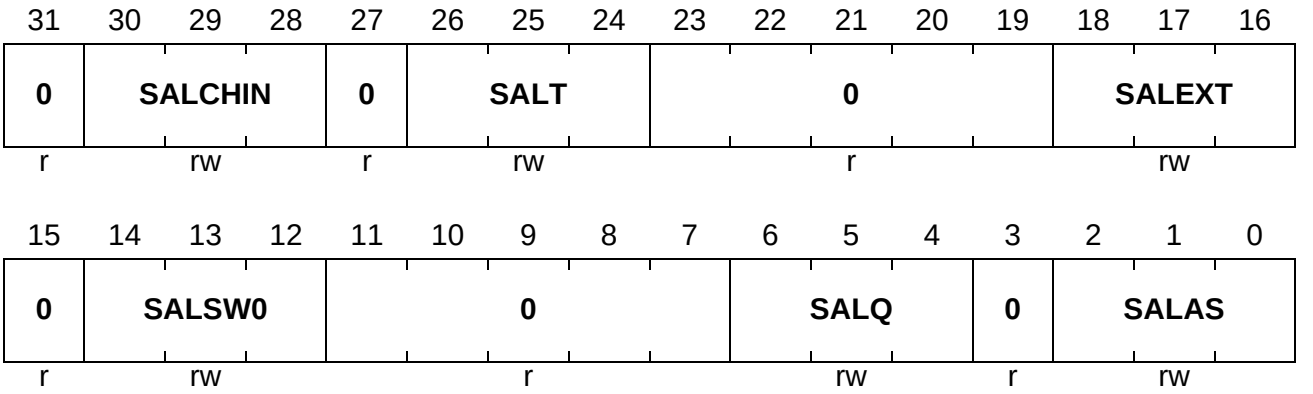
AP
Arbitration Participation Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
ASP	0	rwh	**Auto-Scan Arbitration Participation** 0 Source does not participate in arbitration 1 Source participates in arbitration
QP	1	rwh	**Queue Arbitration Participation** 0 Source does not participate in arbitration 1 Source participates in arbitration
SW0P	3	rwh	**Software SW0 Arbitration Participation Flag** 0 Source does not participate in arbitration 1 Source participates in arbitration
EXP	4	rwh	**External Event Arbitration Participation Flag** 0 Source does not participate in arbitration 1 Source participates in arbitration
TP	6	rwh	**Timer Arbitration Participation Flag** 0 Source does not participate in arbitration 1 Source participates in arbitration
CHP	7	rwh	**Channel Injection Arbitration Participation Flag** 0 Source does not participate in arbitration 1 Source participates in arbitration
0	2, 5, [31:8]	r	**Reserved**; read as 0; should be written with 0.

SAL
Source Arbitration Level Register

Reset Value: 0103 4067$_H$



Field	Bits	Type	Description
SALAS	[2:0]	rw	**Auto-Scan Source Arbitration Level** 000_B Highest priority for arbitration 111_B Lowest priority for arbitration
SALQ	[6:4]	rw	**Queue Source Arbitration Level** 000_B Highest priority for arbitration 111_B Lowest priority for arbitration
SALSW0	[14:12]	rw	**Software Source Arbitration Level** 000_B Highest priority for arbitration 111_B Lowest priority for arbitration
SALEXT	[18:16]	rw	**External Event Source Arbitration Level** 000_B Highest priority for arbitration 111_B Lowest priority for arbitration
SALT	[26:24]	rw	**Timer Source Arbitration Level** 000_B Highest priority for arbitration 111_B Lowest priority for arbitration
SALCHIN	[30:28]	rw	**Channel Injection Source Arbitration Level** 000_B Highest priority for arbitration 111_B Lowest priority for arbitration
0	3, 15, 27,31, [11:7], [23:19]	r	**Reserved**; read as 0; should be written with 0.

Note: see also ***Section 8.1.2.1***

TTC
Time Trigger Control Register

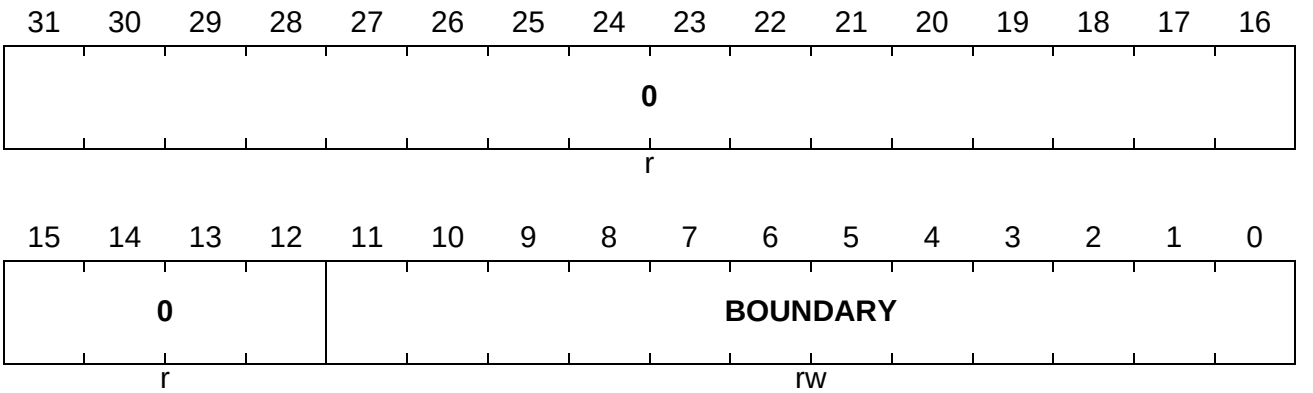
Reset Value: XXXX XXXX$_H$

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

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

Field	Bits	Type	Description
TTCCHn, n = 15-0	[15:0]	rw	**Timer Trigger Control for Channel n** Specifies whether or not a conversion request is triggered for this channel n on timer underflow. 0 No conversion request is triggered 1 A conversion request is triggered
0	[31:16]	r	**Reserved**; read as 0; should be written with 0.

LCCONm, m = 3-0
Limit Check Control Register **Reset Value: XXXX XXXX$_H$**



Field	Bits	Type	Description
BOUNDARY	[11:0]	rw	**Boundary for Limit Checks** Alignment of boundaries in 8-bit, 10-bit, 12-bit resolution of the A/D Converter: 8-bit: LCCONm[11:4] 10-bit: LCCONm[11:2] 12-bit: LCCONm[11:0]
0	[31:12]	r	**Reserved**; read as 0; should be written with 0.

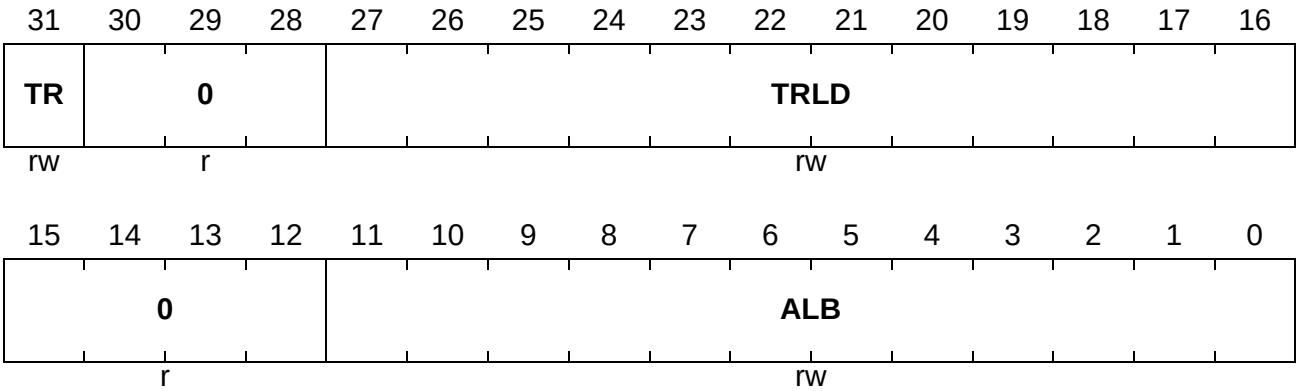
EXTCk, k = 1-0
External Trigger Control Register **Reset Value: XXXX XXXX$_H$**

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETCCH15	ETCCH14	ETCCH13	ETCCH12	ETCCH11	ETCCH10	ETCCH9	ETCCH8	ETCCH7	ETCCH6	ETCCH5	ETCCH4	ETCCH3	ETCCH2	ETCCH1	ETCCH0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ETCHn, n = 15-0	[15:0]	rw	**External Trigger Control for Channel n** Specifies if a conversion request is triggered on a pulse on the external/peripheral trigger input for this channel n. 0 No conversion request is triggered for channel n 1 A conversion request is triggered for channel n *Note: see also bit external event trigger control.*
0	[31:16]	r	**Reserved**; read as 0; should be written with 0.

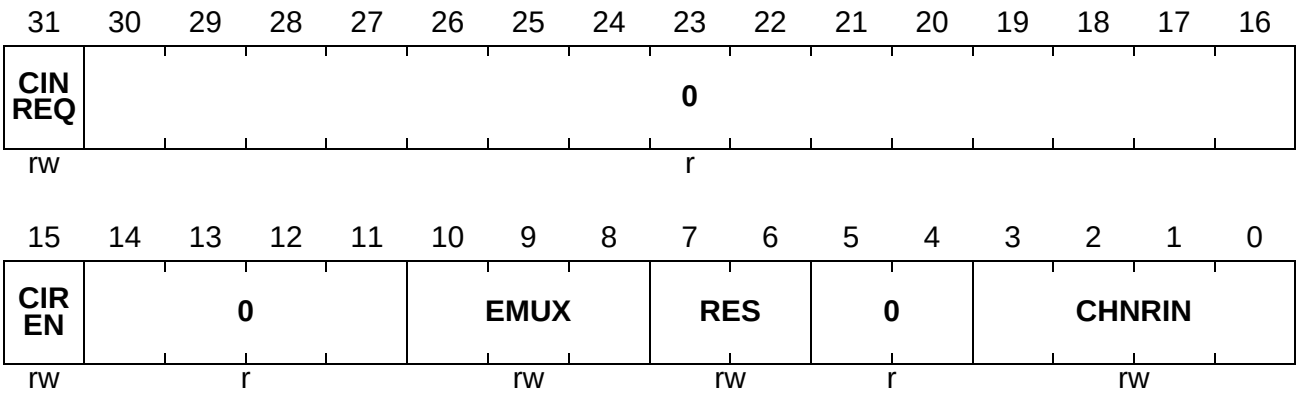
TCON
Timer Control Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
ALB	[11:0]	rw	**Arbitration Lock Boundary** The arbitration lock boundary is used specify the arbitration lock time t_{LOCK}. Arbitration Lock Mode is automatically enabled if any value greater than zero is written to ALB. *Note: The arbitration is locked if the value of ALB is above than TRLD.*
TRLD	[27:16]	rw	**Timer Reload Value** The timer reload value is reloaded into the timer register on timer underflow. *Note: If the timer reload value is zero, timer lock is always active and a service request can be generated for each timer clock.*
TR	31	rw	**Timer Run Control** 0 Timer is stopped 1 Timer register is decremented due to the f_{Timer} *Note: Resetting bit TR causes the arbitration lock to be removed, if it is set.*
0	[15:12], [30:28]	r	**Reserved**; read as 0; should be written with 0.

CHIN
Channel Injection Control Register

Reset Value: XXXX XXXX$_H$

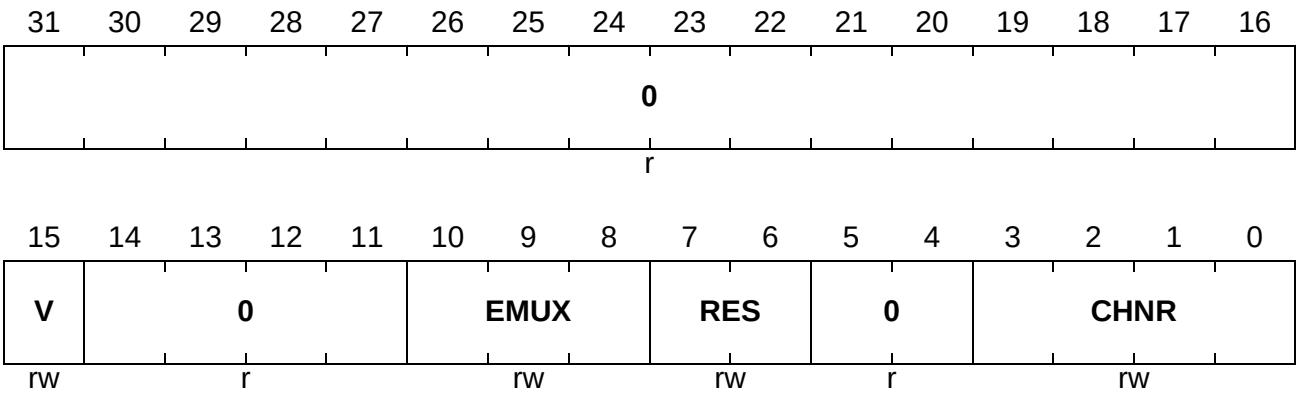


Field	Bits	Type	Description
CHNRIN	[3:0]	rw	**Channel Number to be Injected**
RES	[7:6]	rw	**Conversion Resolution Control** Controls the resolution of the A/D Converter for the conversion of this channel. Any modification of this bit field is taken into account after the currently running conversion is finished. 00 10-bit resolution 01 12-bit resolution 10 8-bit resolution 11 Reserved
EMUX	[10:8]	rw	**External Multiplexer Control** Drives an external multiplexer connected to this analog input channel. *Note: See also the external multiplexer enable bit CON.EMUXEN.*
CIREN	15	rw	**Cancel, Inject, and Repeat Enable** 0 Cancel, Inject, and Repeat feature is disabled 1 Cancel, Inject, and Repeat feature is enabled

Field	Bits	Type	Description
CINREQ	31	rw	**Channel Injection Request** Request bit for Channel Injection and is automatically reset after the requested conversion is injected. 0 No Channel Injection request 1 Channel Injection request *Note: Resetting bit AP_CHP causes bit CHIN.CINREQ to be reset.*
0	[5:4], [14:11], [30:16]	r	**Reserved**; read as 0; should be written with 0.

QR
Queue Register

Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
CHNR	[3:0]	rw	**Channel to be converted**
RES	[7:6]	rw	**Conversion Resolution Control** Controls the resolution of the A/D Converter for the conversion of this channel. Any modification of this bit field is taken into account after the currently running conversion is finished. 00 10-bit resolution 01 12-bit resolution 10 8-bit resolution 11 Reserved
EMUX	[10:8]	rw	**External Multiplexer Control** Drives an external multiplexer connected to this analog input channel. See also the external multiplexer enable bit CON.EMUXEN.
V	15	rw	**Valid Control** Indicates whether the information of register QR is valid or invalid. 0 CHNR, RES, and EMUX are invalid 1 CHNR, RES, and EMUX are valid
0	[5:4], [14:11], [31:16]	r	**Reserved**; read as 0; should be written with 0.

CON
AD Converter Control Register **Reset Value: 4000 0007$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SRTEST	PCD		CPR	0								QWLP			
rw	rw		rw	r								rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	QRS	QEN	EMUXEN	EXTS1	EXTS0	SCNM		CTC							CPS
r	rw	rw	rw	rw	rw	rw		rw							rw

Field	Bits	Type	Description
CPS	0	rw	**Clock Prescaler Select** Defines whether the ADC basic operating clock is divided by 3 or 4. Any modification of this bit is taken into account after the currently performed conversion is finished. 0 ADC basic operating clock f_{BC} is divided by 3 1 ADC basic operating clock f_{BC} is divided by 4
CTC	[7:1]	rw	**Conversion Time Control** Defines the period of the ADC basic operating clock f_{BC}. Any modification of this bit field is taken into account after the currently performed conversion is finished.
SCNM	[9:8]	rw	**Auto-Scan Mode** 00 Auto-scan mode disabled 01 Auto-scan single sequence mode enabled 10 Auto-scan continuous sequence mode enabled 11 Reserved
EXTS0 **EXTS1**	10 11	rw	**External Trigger Select** Selects whether external trigger events are derived from input EXTINk, k = 0, 1 or peripheral interconnect input PTINk, k = 0, 1. 0 Input EXTINk, k = 0, 1 is selected 1 Input PTINk, k = 0, 1 is selected

Field	Bits	Type	Description
EMUXEN	12	rw	**External Multiplexer Enable Control** Enables or disables the external channel expansion feature. 0 External channel expansion feature is disabled 1 External channel expansion feature is enabled
QEN	13	rw	**Queue Enable** Specifies whether or not queue is enabled/disabled and queue based conversion requests are generated or not. 0 Queue is disabled 1 Queue is enabled *Note: The queue load is not affected by a queue disable condition.*
QRS	14	rw	**Queue Reset** Setting bit QRS tags all queue elements invalid (resets V-bit of each queue element), clears bit STAT.QF and STAT.QLP. QRS is automatically reset after all queue elements have been tagged invalid. A read action on QRS shows always zero.
QWLP	[19:16]	rw	**Queue Warning Limit Pointer** The value of the queue warning limit pointer specifies the queue element to be watched.
CPR	28	rw	**Clear of Pending Conversion Requests in Parallel Sources by Arbiter** Bit CPR defines, whether all pending conversion requests for an AD channel, indicated by STAT.CHNRCC, are cancelled by the arbiter or not, when the conversion for this channel has been started. 0 The individual clear by arbiter is enabled. 1 The global clear by arbiter is enabled.
PCD	[30:29]	rw	**Peripheral Clock Divider** The peripheral clock divider is used to divide the input clock f_{ADC} of the analog part of the ADC Module. 00 1:1 clock divider selected 01 2:1 clock divider selected 10 4:1 clock divider selected (default after reset) 11 8:1 clock divider selected

Field	Bits	Type	Description
SRTEST	31	rw	**Service Request Test Mode** Used to set a source service request flag under software control. *Note: See also the chapter on the service request scheme.*
0	15, [27:20]	r	**Reserved**; read as 0; should be written with 0.

SCN
Auto-Scan Conversion Request Register **Reset Value: XXXX XXXX$_H$**

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

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

Field	Bits	Type	Description
SRQn, n = 15-0	[15:0]	rw	**Auto-Scan Request for Channel n** 0 Channel n does not participate in an auto-scan sequence 1 Channel n participates in an auto-scan sequence *Note: Bits SRQn maintain their values after auto-scan control bit field CON.SCNM is cleared.*
0	[31:16]	r	**Reserved**; read as 0; should be written with 0.

REQ0
Software SW0 Conversion Request Register

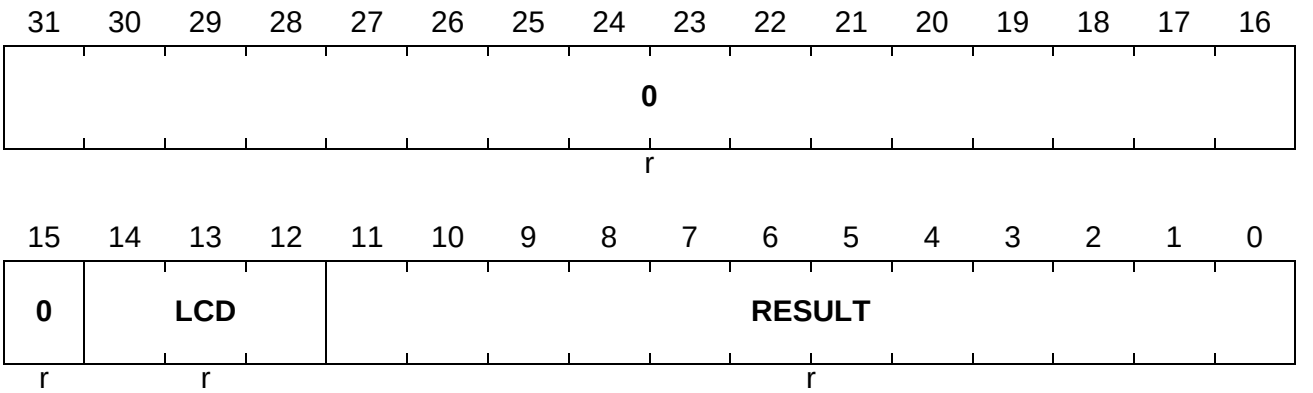
Reset Value: XXXX XXXX$_H$

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REQ0 15	REQ0 14	REQ0 13	REQ0 12	REQ0 11	REQ0 10	REQ0 9	REQ0 8	REQ0 7	REQ0 6	REQ0 5	REQ0 4	REQ0 3	REQ0 2	REQ0 1	REQ0 0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
REQ0n, n = 15-0	[15:0]	rw	**Software SW0 Conversion Request for Channel n** 0 No conversion is requested for channel n 1 A conversion is requested for channel n
0	[31:16]	r	**Reserved**; read as 0; should be written with 0.

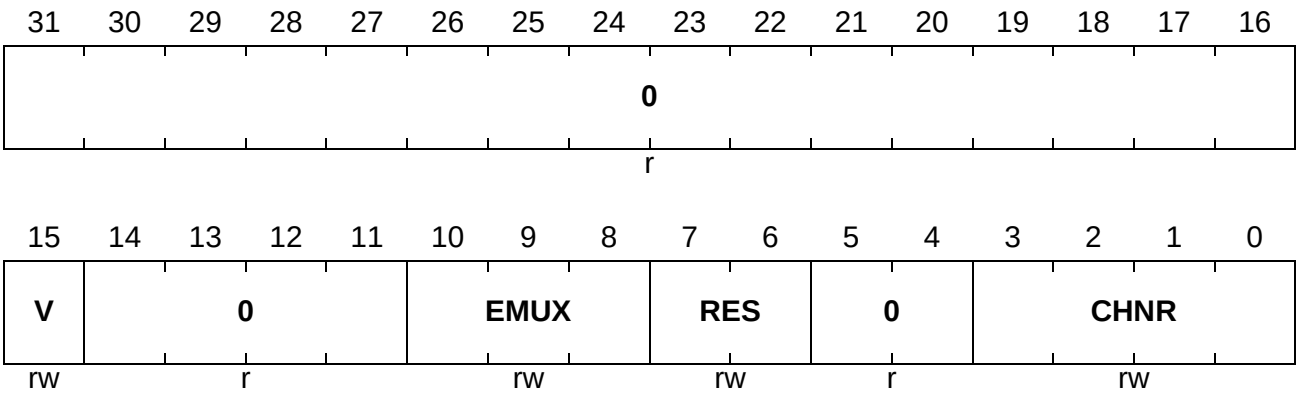
CHSTATn, n = 15-0
Channel Status Register **Reset Value: XXXX XXXX$_H$**



Field	Bits	Type	Description
RESULT	[11:0]	r	**Result of the Last Conversion** Alignment of 8-bit, 10-bit, 12-bit conversion result. 8-bit: CHSTATn[11:4] 10-bit: CHSTATn[11:2] 12-bit: CHSTATn[11:0]
LCD	[14:12]	r	**Last Conversion Data** Indicates the origin of the conversion result, stored in bit field RESULT. 000_B Channel Injection 001_B Timer 010_B Synchronized Injection 011_B External event 100_B Software SW0 101_B Reserved 110_B Queue 111_B Auto-Scan In case that the external multiplexer functionality is enabled, each odd numbered channel-specific status register (CHSTATn, n = 15, 13, 11, … 1) contains in odd numbered bit fields CHSTATn.LCD (n = 15, 13, 11, …1) the external multiplexer information of the last conversion result.
0	[31:15]	r	**Reserved**; read as 0.

QUEUE0
Queue Status Register

Reset Value: XXXX XXXX$_H$



Field	Bits	Type	Description
CHNR	[3:0]	rw	**Channel to be converted**
RES	[7:6]	rw	**Conversion Resolution Control** Controls the resolution of the A/D Converter for the conversion of this channel. Any modification of this bit field is taken into account after the currently running conversion is finished. 00 10-bit resolution 01 12-bit resolution 10 8-bit resolution 11 Reserved
EMUX	[10:8]	rw	**External Multiplexer Control** Drives an external multiplexer connected to this analog input channel. See also the external multiplexer enable bit CON.EMUXEN.
V	15	rw	**Valid Control** Indicates whether the information of register QR is valid or invalid. 0 QR_CHNR, QR_RES, and QR_EMUX are invalid 1 QR_CHNR, QR_RES, and QR_EMUX are valid
0	[5:4], [14:11], [31:16]	r	**Reserved**; read as 0; should be written with 0.

SW0CRP
Software SW0 Conversion Request Pending Register Reset Value: 0000 0000$_H$

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SW0CRP15	SW0CRP14	SW0CRP13	SW0CRP12	SW0CRP11	SW0CRP10	SW0CRP9	SW0CRP8	SW0CRP7	SW0CRP6	SW0CRP5	SW0CRP4	SW0CRP3	SW0CRP2	SW0CRP1	SW0CRP0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Field	Bits	Type	Description
SW0CRPn, n = 15-0	[15:0]	r	**Software SW0 Conversion Request Pending Flag for Channel n** The pending flag is set each time a conversion request is generated for this specific channel n by SW0, which could not be serviced immediately. A start of conversion of the pending request leads automatically to a reset of the pending flag. All pending request flags can also be reset under software control, if bit AP.SW0P is reset. 0 No SW0 based conversion request is pending 1 A SW0 based conversion request is pending
0	[31:16]	r	**Reserved**; read as 0; should be written with 0.

ASCRP
Auto-Scan Conversion Request Pending Register

Reset Value: 0000 0000$_H$

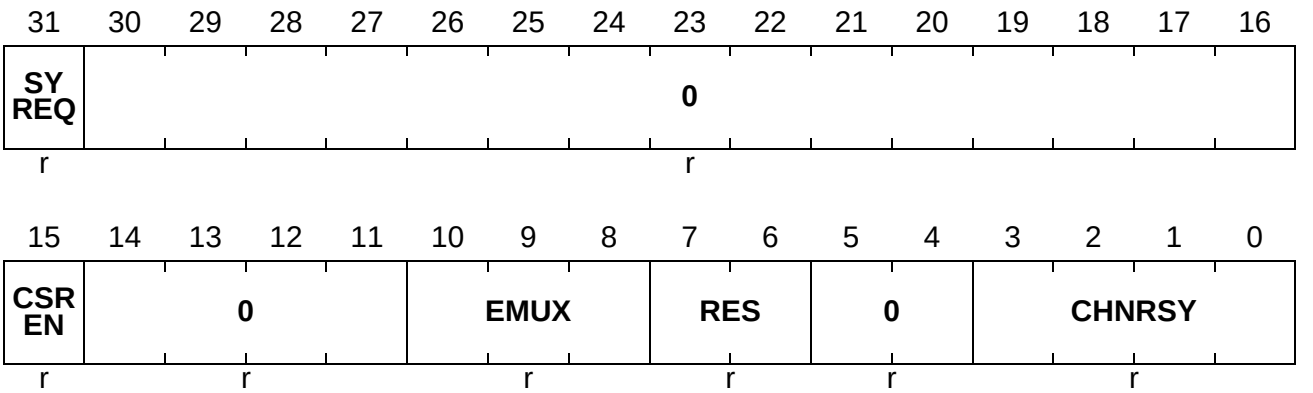
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ASCRP15	ASCRP14	ASCRP13	ASCRP12	ASCRP11	ASCRP10	ASCRP9	ASCRP8	ASCRP7	ASCRP6	ASCRP5	ASCRP4	ASCRP3	ASCRP2	ASCRP1	ASCRP0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Field	Bits	Type	Description
ASCRPn, n = 15-0	[15:0]	r	**Auto-Scan Conversion Request Pending Flag for Channel n** The pending flag is set each time a conversion request is generated for this specific channel n by auto-scan that could not be serviced immediately. A start of conversion of the pending request leads automatically to a reset of the pending flag. All pending request flags can also be reset under software control, if bit AP.ASP is reset. 0 No auto-scan based conversion request is pending 1 An auto-scan based conversion request is pending
0	[31:16]	r	**Reserved**; read as 0; should be written with 0.

SYSTAT
Synchronized Injection Status Register

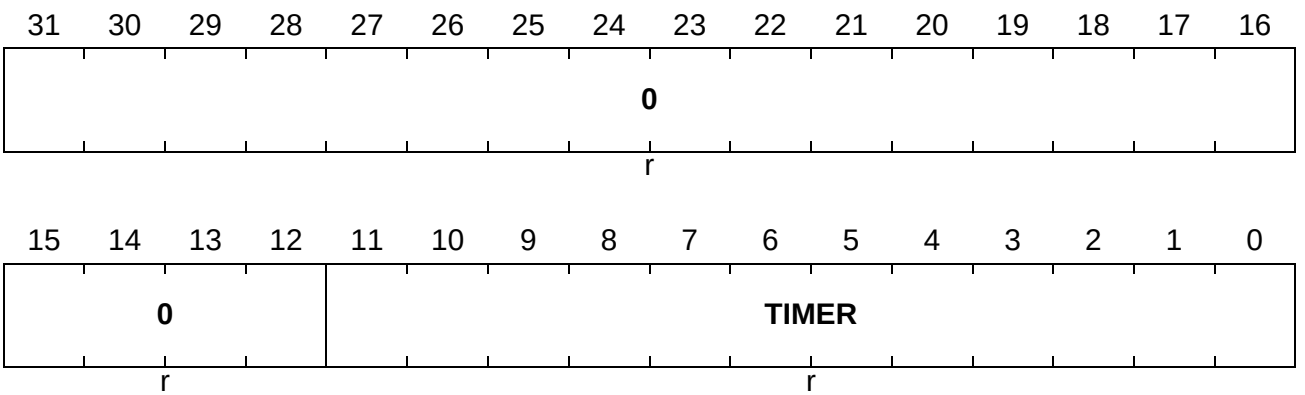
Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
CHNRSY	[3:0]	r	**Channel to be Converted in Synchronized Conversion** This bit field indicates the channel number of the analog channel which is converted in synchronized mode.
RES	[7:6]	r	**Conversion Resolution Status** Indicates the resolution of the A/D Converter for the conversion of this channel. Any update of this bit field is taken into account after the currently running conversion is finished. 00 10-bit resolution 01 12-bit resolution 10 8-bit resolution 11 Reserved
EMUX	[10:8]	r	**External Multiplexer Status** Shows the external multiplexer selection that is used during an AD conversion for the analog input channel defined by CHNRSY. *Note: See also the external multiplexer enable bit CON.EMUXEN.*
CSREN	15	r	**Cancel, Synchronize, and Repeat State** Indicates whether the Cancel, Synchronize and Repeat feature is enabled or disabled for the analog input channel defined by CHNRSY. 0 Cancel, Synchronize, and Repeat is disabled 1 Cancel, Synchronize, and Repeat is enabled

Field	Bits	Type	Description
SYREQ	31	r	**Synchronized Injection Request State** Indicates whether a synchronized conversion is requested for the analog input channel defined by CHNRSY. 0 No synchronized conversion is requested 1 A synchronized conversion is requested
0	[5:4], [14:11] [30:16]	r	**Reserved**; read as 0.

TSTAT
Timer Status Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
TIMER	[11:0]	r	**Timer Register** Contains the current value of the timer.
0	[31:12]	r	**Reserved**; read as 0.

STAT
AD Converter Status Register

Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0			SYMS	IENPAR	IENREQ	PARSY	REQSY	0			QF	QLP			
r			r	r	r	r	r	r			r	r			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BUSY	SMPL	CAL	AL	0	CHTSCC			0				CHNRCC			
r	r	r	r	r	r			r				r			

Field	Bits	Type	Description
CHNRCC	[3:0]	r	**Number of Channel Currently Converted** 0000_B Channel 0 is currently converted … … 1111_B Channel 15 is currently converted
CHTSCC	[10:8]	r	**Trigger Source of Channel Currently Converted** Indicates the origin of a conversion request that triggered the channel currently converted. 000_B Channel Injection 001_B Timer 010_B Synchronization injection mode 011_B External events 100_B Software SW0 101_B Reserved 110_B Queue 111_B Auto-scan
AL	12	r	**Arbitration Lock** This bit is set, if the timer running in Arbitration Lock Mode meets the value specified in TCON.ALB, while it is reset on timer underflow. 0 Arbitration Lock Mode is inactive 1 Arbitration Lock Mode is active
CAL	13	r	**Power-Up Calibration Status** 0 Power-up calibration is finished 1 The ADC is in power-up calibration phase

Field	Bits	Type	Description
SMPL	14	r	**Sample Phase Status** 0 The ADC is currently not in the sample phase. 1 The ADC currently samples the analog input voltage (sample phase).
BUSY	15	r	**Busy Status** 0 The ADC is currently idle 1 The ADC currently performs a conversion
QLP	[19:16]	r	**Queue Level Pointer** This bit field points to an empty queue element with the lowest queue element number. It is incremented on a queue load operation; it is decremented after a queue based conversion is started.
QF	20	r	**Queue Full Status** This bit is set on a write action to the last empty queue element. It is reset if at least one queue element is empty. 0 At least on queue element is empty 1 Queue is full
REQSY	24	r	**Requestor of Synchronized Conversion** This bit is set during a synchronized conversion in the case that this ADC Module is the master in the synchronized conversion. 0 No synchronized conversion is performed **or** this ADC Module provides no master functionality in the synchronized conversion. 1 A synchronized conversion is performed **and** this ADC Module provides master functionality.
PARSY	25	r	**Partner in Synchronized Conversion** This bit is set during a synchronized conversion in the case that this ADC Module is the slave in the synchronized conversion. 0 No synchronized conversion is performed **or** this ADC Module provides no slave functionality in the synchronized conversion. 1 A synchronized conversion is performed **and** this ADC Module provides slave functionality.

Field	Bits	Type	Description
IENREQ	26	r	**Interrupt Enable by Requestor** This bit is set **in the master** ADC Module after the master finished its synchronized conversion. 0 The master does not finish the synchronized conversion, if any was requested. 1 The master finished its synchronized conversion.
IENPAR	27	r	**Interrupt Enable by Requestor** This bit is set **in the master** ADC Module after the slave finished its synchronized conversion.In master/slave mode, bit IENPAR is driven by the opposite ADC Module after the synchronized conversion is finished. 0 The slave doesn't finish the synchronized conversion, if any was requested. 1 The slave finished its synchronized conversion.
SYMS	28	r	**Synchronized Master/Slave Functionality** Is set if this ADC Module enters the master/slave mode. It is reset after the service request of synchronization mode is generated. 0 This ADC Module does not finish the synchronized conversion in master/slave mode, if any was requested. 1 Synchronized conversion in master/slave mode.
0	11, [7:4], [23:21], [31:29]	r	**Reserved**; read as 0.

TCRP
Timer Conversion Request Pending Register

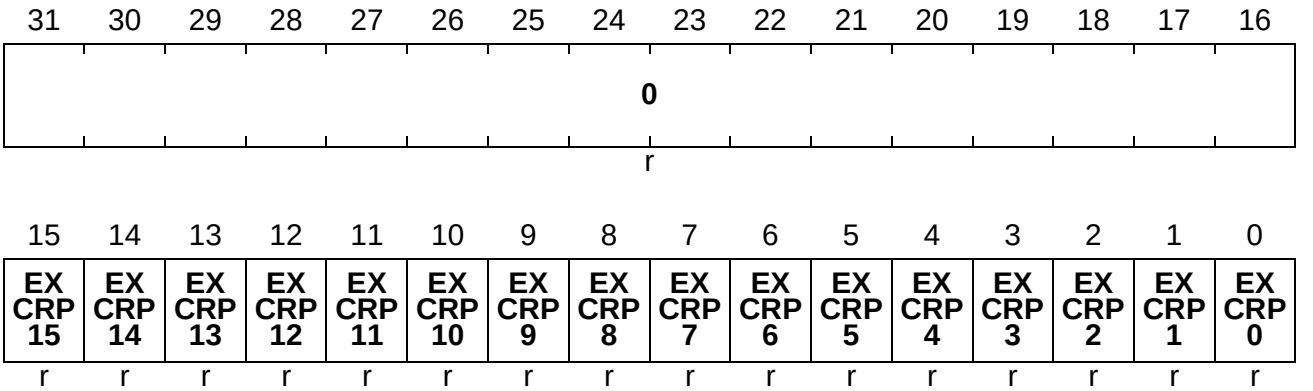
Reset Value: 0000 0000$_H$

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

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

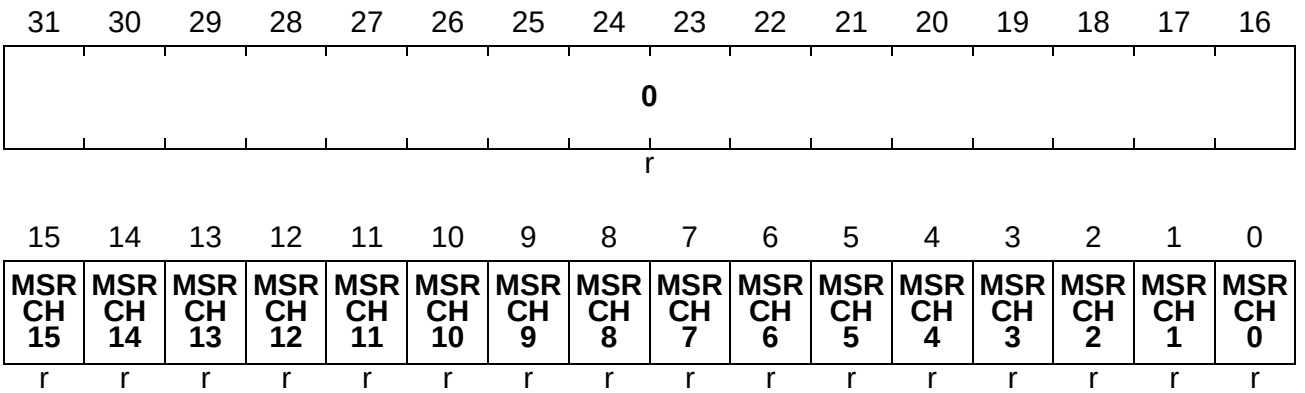
Field	Bits	Type	Description
TRPn, n = 15-0	[15:0]	r	**Timer Conversion Request Pending Flag for Channel n** The pending flag is set each time a conversion request is generated for channel n on timer underflow that could not be serviced immediately.A start of conversion of the pending request leads automatically to a reset of the pending flag. All pending request flags can also be reset under software control, if bit AP.TP is reset. 0 No timer based conversion request is pending 1 A timer based conversion request is pending
0	[31:16]	r	**Reserved**; read as 0; should be written with 0.

EXCRP
External Event Conversion Request Pending Register Reset Value: 0000 0000$_H$



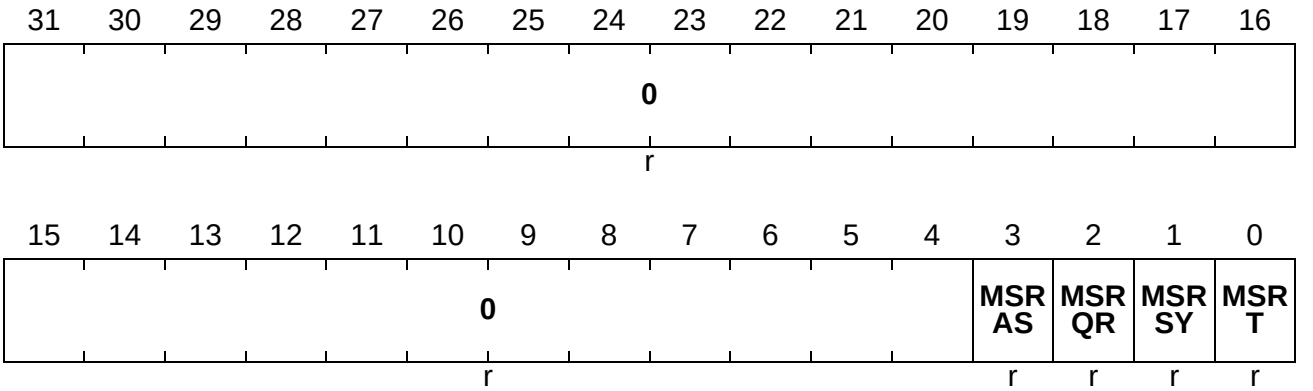
Field	Bits	Type	Description
EXCRPn, n = 15-0	[15:0]	r	**External Event Conversion Request Pending Flag for Channel n** The pending flag is set each time a conversion request is generated for channel n by an external event that could not be serviced immediately. A start of conversion of the pending request leads automatically to a reset of the pending flag. All pending request flags can also be reset under software control, if bit AP.EXP is reset. 0 No external event based conversion request is pending 1 An external event based conversion request is pending
0	[31:16]	r	**Reserved**; read as 0; should be written with 0.

MSS0
Module Service Request Status Register 0 **Reset Value: 0000 0000$_H$**



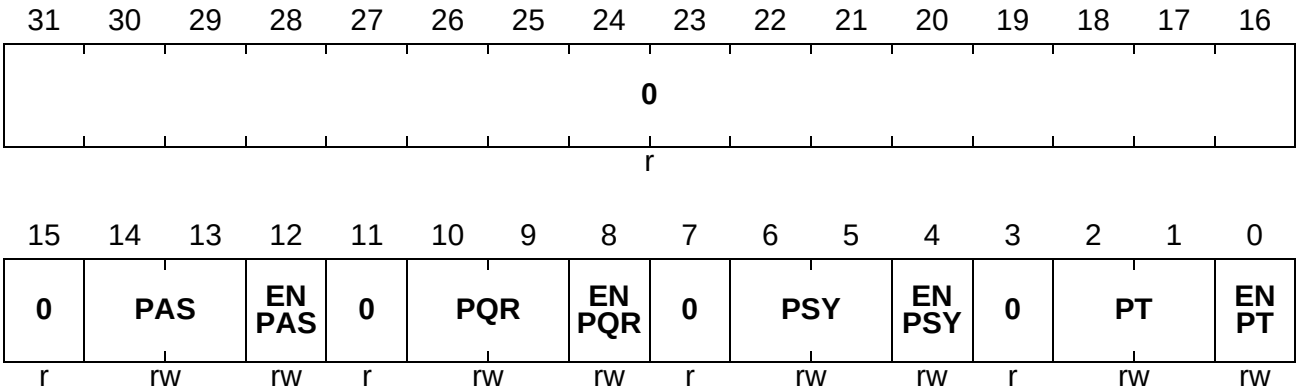
Field	Bits	Type	Description
MSRCHn, n = 15-0	[15:0]	r	**Module Service Request Status for Channel n** Specifies whether or not a source service request is generated by channel n. 0 No source service request is generated 1 A source service request is generated
0	[31:16]	r	**Reserved**; read as 0; should be written with 0.

MSS1
Module Service Request Status Register 1 **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
MSRT	0	r	**Module Service Request Status for Source Timer** Specifies whether or not a timer source service request is generated. 0 No timer source service request is generated 1 A timer source service request is generated
MSRSY	1	r	**Module Service Request Status for Source Synchronized Injection** 0 No Synchronized Injection source service request is generated 1 A Synchronized Injection source service request is generated
MSRQR	2	r	**Module Service Request Status for Source Queue** 0 No queue source service request is generated 1 A queue source service request is generated
MSRAS	3	r	**Module Service Request Status for Source Auto-Scan** 0 No auto-scan source service request is generated 1 A auto-scan source service request is generated
0	[31:4]	r	**Reserved**; read as 0; should be written with 0.

SRNP
Service Request Node Pointer Register **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
PT **PSY** **PQR** **PAS**	[2:1] [6:5] [10:9] [14:13]	rw	**Service Request Node Pointer Destination** Directs a Service Request Source trigger to one out of four Service Request Nodes. 00 Directed to Service Request Node Pointer 0 01 Directed to Service Request Node Pointer 1 10 Directed to Service Request Node Pointer 2 11 Directed to Service Request Node Pointer 3
ENPT **ENPSY** **ENPQR** **ENPAS**	0 4 8 12	rw	**Service Request Node Pointer Enable** 0 Service Request Node Pointer is disabled 1 Service Request Node Pointer is enabled
0	3, 7, 11, [31:15]	r	**Reserved**; read as 0; should be written with 0.

8.3 ADC0/ADC1 Module Implementation

This section describes the ADC0/ADC1 module related external functions such as port connections, interrupt control, address decoding, and clock control.

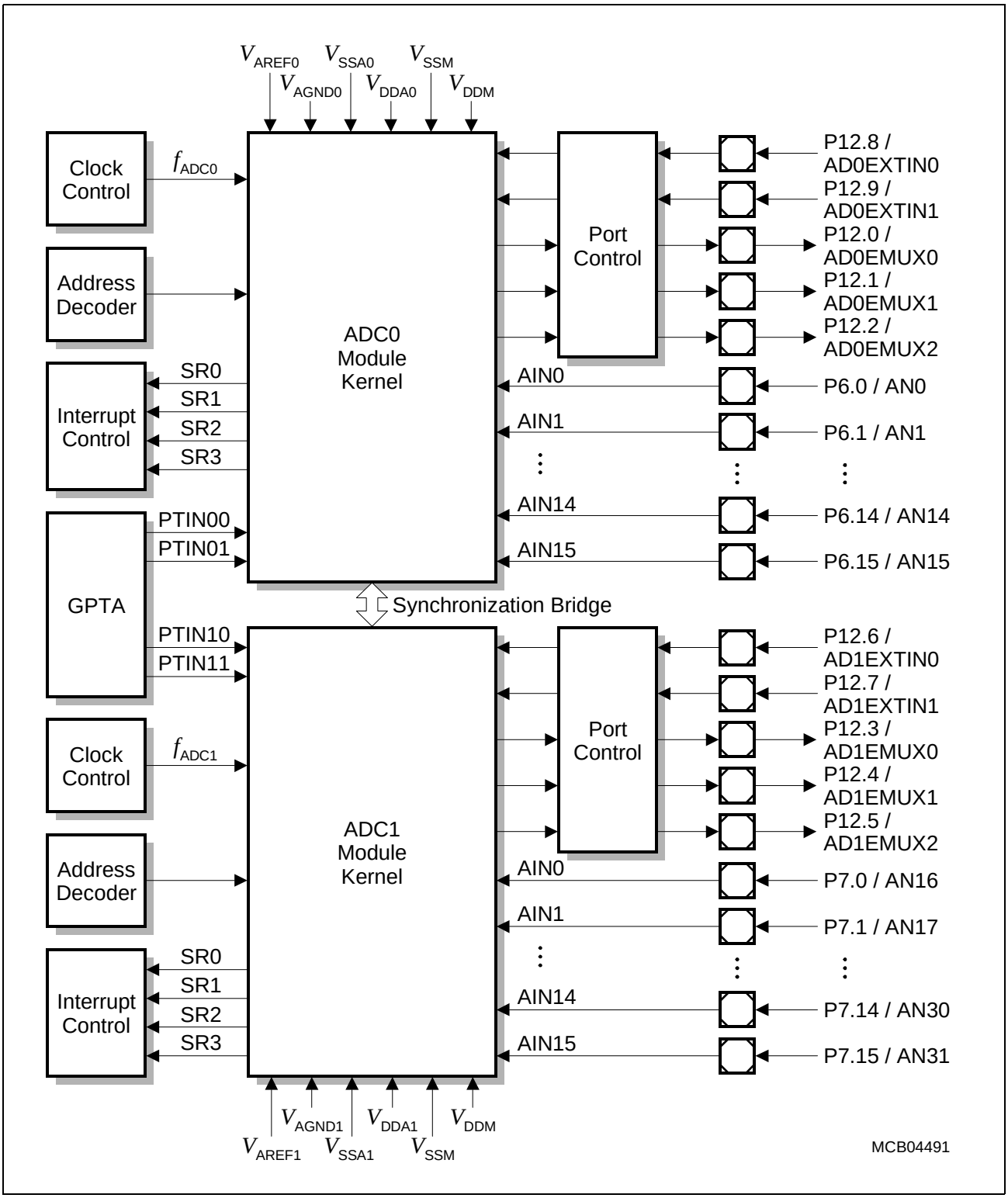


Figure 8-28 ADC0/ADC1 Module Implementation and Interconnections

8.3.1 ADC0/ADC1 Module Related External Registers

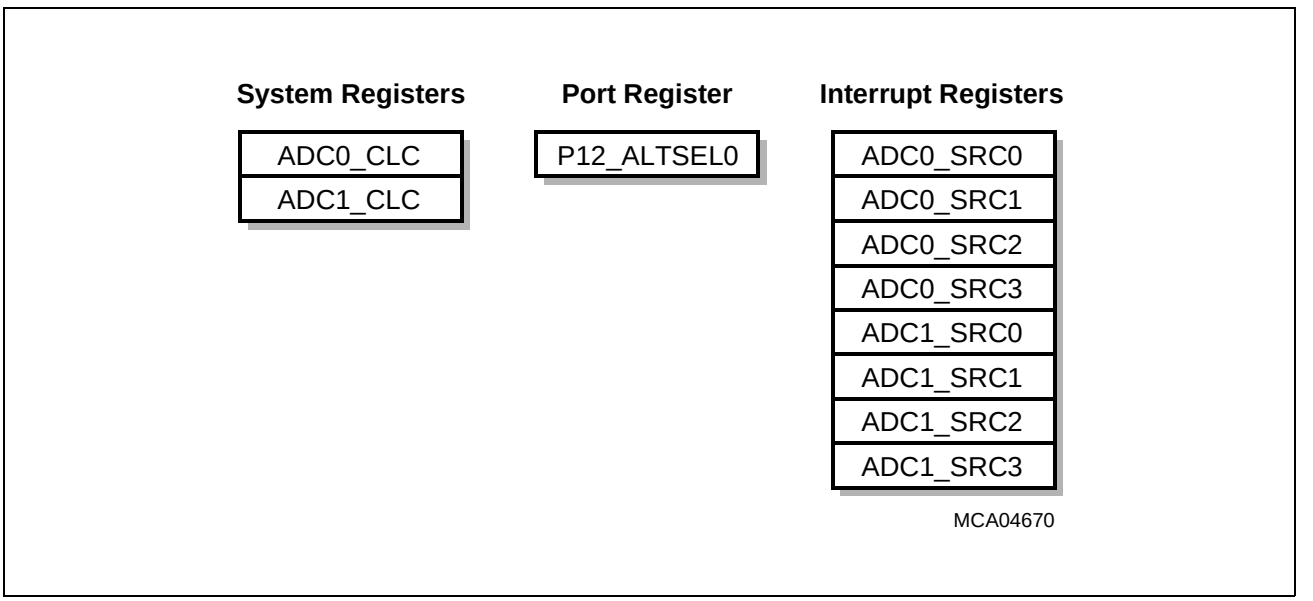
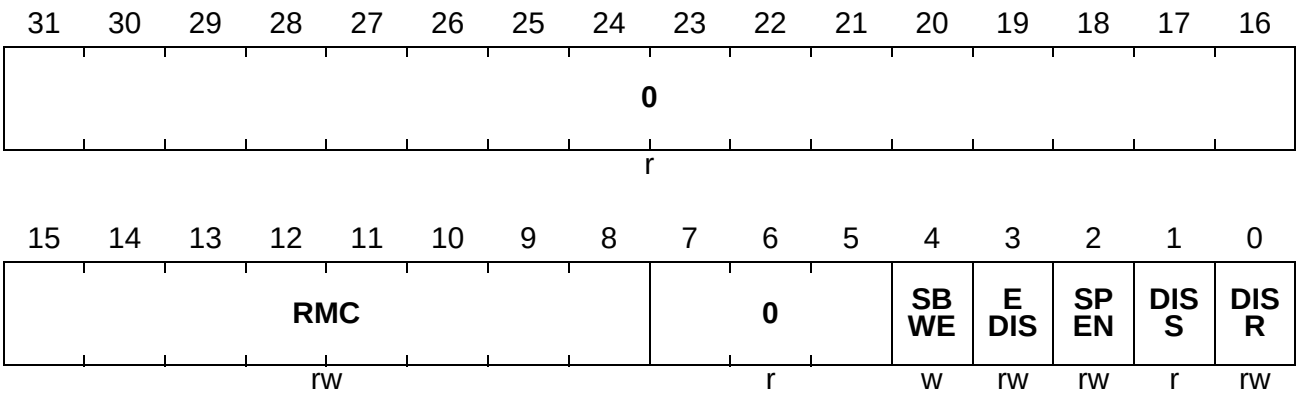


Figure 8-29 ADC0/ADC1 Implementation Specific Special Function Registers

8.3.1.1 Clock Control Registers

The clock control register allows the programmer to adapt the functionality and power consumption of an ADC Module to the requirements of the application. The diagram below shows the clock control register functionality as is implemented for the ADC Modules. ADC0_CLC controls the f_{ADC0} clock signal and ADC1_CLC controls the f_{ADC1} clock signal.

ADC0_CLC
ADC0 Clock Control Register **Reset Value: 0000 0002$_H$**
ADC1_CLC
ADC1 Clock Control Register **Reset Value: 0000 0002$_H$**



Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control of the module.
DISS	1	r	**Module Disable Status Bit** Bit indicates the current status of the module.
SPEN	2	rw	**Module Suspend Enable for OCDS** Used for enabling the suspend mode.
EDIS	3	rw	**External Request Disable** Used for controlling the external clock disable request.
SBWE	4	w	**Module Suspend Bit Write Enable for OCDS** Defines whether SPEN and FSOE are write protected.
RMC	[15:8]	rw	**8-Bit Clock Divider Value in RUN Mode**
0	[7:5], [31:16]	r	**Reserved**; returns 0 if read; should be written with 0.

Note: After a hardware reset operation, the ADC Modules are disabled.

8.3.1.2 Port Registers

The external digital I/O lines of the ADC Modules are connected with Port 12 and can be enabled as alternate function of Port 12. This alternate function is controlled by register P12_ALTSEL0.

Note: Bits marked with 'X' are not relevant for ADC operation.

P12_ALTSEL0
Port 12 Alternate Select Register 0 **Reset Value: 0000 0000$_H$**

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X						P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rw						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Table 8-14 shows the bits of P12_ALTSEL0 that must be set to enable the required I/O functionality of the ADC I/O lines.

Table 8-14 ADC0/ADC1 I/O Line Selection and Setup

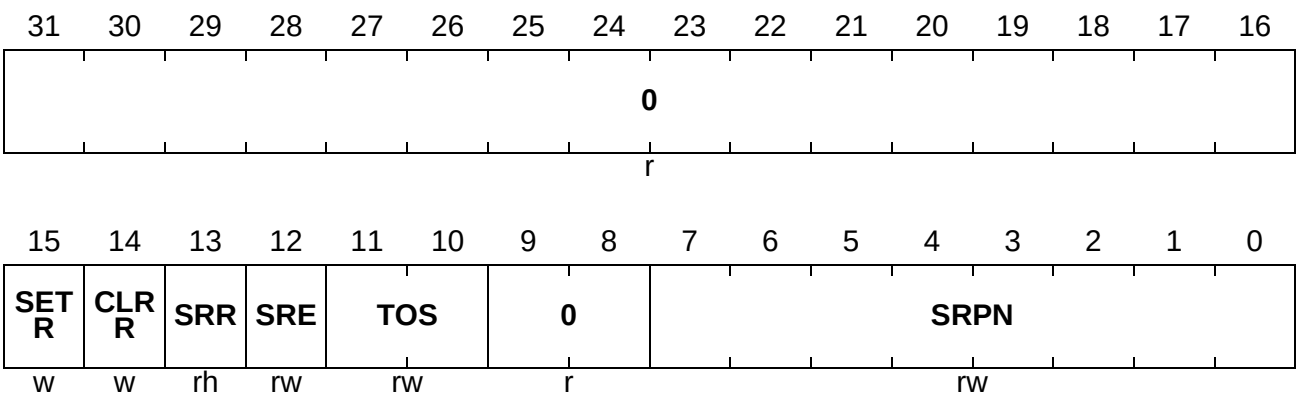
Module	Port Lines	P12_ALTSEL0 Bits	I/O
ADC0	P12.8 / AD0EXTIN0	P12_ALTSEL0.P8 = 0	Input
	P12.9 / AD0EXTIN1	P12_ALTSEL0.P9 = 0	Input
	P12.0 / AD0EMUX0	P12_ALTSEL0.P0 = 1	Output
	P12.1 / AD0EMUX1	P12_ALTSEL0.P1 = 1	Output
	P12.2 / AD0EMUX2	P12_ALTSEL0.P2 = 1	Output
ADC1	P12.6 / AD1EXTIN0	P12_ALTSEL0.P6 = 1	Input
	P12.7 / AD1EXTIN1	P12_ALTSEL0.P7 = 1	Input
	P12.3 / AD1EMUX0	P12_ALTSEL0.P3 = 1	Output
	P12.4 / AD1EMUX1	P12_ALTSEL0.P4 = 1	Output
	P12.5 / AD1EMUX2	P12_ALTSEL0.P5 = 1	Output

8.3.1.3 Interrupt Registers

The eight interrupts of ADC0 and ADC1 are controlled by the following service request control registers:

ADC0_SRC0
ADC0 Service Request Control Register 0
ADC0_SRC1
ADC0 Service Request Control Register 1
ADC0_SRC2
ADC0 Service Request Control Register 2
ADC0_SRC3
ADC0 Service Request Control Register 3
ADC1_SRC0
ADC1 Service Request Control Register 0
ADC1_SRC1
ADC1 Service Request Control Register 1
ADC1_SRC2
ADC1 Service Request Control Register 2
ADC1_SRC3
ADC1 Service Request Control Register 3

Reset Values: 0000 0000$_H$



Field	Bits	Type	Description
SRPN	[7:0]	rw	**Service Request Priority Number**
TOS	[11:10]	rw	**Type of Service Control**
SRE	12	rw	**Service Request Enable**
SRR	13	rh	**Service Request Flag**
CLRR	14	w	**Request Clear Bit**
SETR	15	w	**Request Set Bit**

Field	Bits	Type	Description
0	[9:8], [31:16]	r	**Reserved**; returns 0 if read; should be written with 0.

Note: Further details on interrupt handling and processing are described in the “Interrupt System” chapter of the TC1775 System Units User’s Manual.

8.3.2 ADC0/ADC1 Register Address Ranges

In the TC1775, the registers of the two ADC Modules are located in the following address ranges:

- ADC0 module: Module Base Address = F000 2200_H
  Module End Address = F000 $23FF_H$
- ADC1 module: Module Base Address = F000 2400_H
  Module End Address = F000 $25FF_H$
- Absolute Register Address = Module Base Address + Offset Address (offset addresses see **Table 8-13**)

9 Index

9.1 Keyword Index

This section lists a number of keywords which refer to specific details of the TC1775 in terms of its architecture, its functional units, or functions. Bold page number entries identify the main definition material for a topic.

P

R

S

9.2 Register Index

This section lists the references to the Special Function Registers of the TC1775.

Infineon goes for Business Excellence

“Business excellence means intelligent approaches and clearly defined processes, which are both constantly under review and ultimately lead to good operating results.
Better operating results and business excellence mean less idleness and wastefulness for all of us, more professional success, more accurate information, a better overview and, thereby, less frustration and more satisfaction.”

Dr. Ulrich Schumacher

http://www.infineon.com

Published by Infineon Technologies AG