XMC1100

Microcontroller Series
for Industrial Applications

XMC1000 Family

ARM® Cortex™-M0
32-bit processor core

Reference Manual

V1.0 2013-03

Microcontrollers

Edition 2013-03

Published by
Infineon Technologies AG
81726 Munich, Germany

Legal Disclaimer

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

Information

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

Warnings

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

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

XMC1100

Microcontroller Series
for Industrial Applications

XMC1000 Family

ARM® Cortex™-M0
32-bit processor core

Reference Manual

V1.0 2013-03

Microcontrollers

XMC1100 Reference Manual

Revision History: V1.0 2013-03

Previous Versions:
none

Page	Subjects

Trademarks

C166™, TriCore™ and DAVE™ are trademarks of Infineon Technologies AG.

ARM®, ARM Powered® and AMBA® are registered trademarks of ARM, Limited.

Cortex™, CoreSight™, ETM™, Embedded Trace Macrocell™ and Embedded Trace Buffer™ are trademarks of ARM, Limited.

We Listen to Your Comments
Is there any information in 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

About this Document

This Reference Manual is addressed to embedded hardware and software developers. It provides the reader with detailed descriptions about the behavior of the XMC1100 series functional units and their interaction.

The manual describes the functionality of the superset device of the XMC1100 microcontroller series. For the available functionality (features) of a specific XMC1100 derivative (derivative device), please refer to the respective Data Sheet. For simplicity, the various device types are referenced by the collective term XMC1100 throughout this manual.

XMC1000 Family User Documentation

The set of user documentation includes:

- **Reference Manual**
  – decribes the functionality of the superset device.
- **Data Sheets**
  – list the complete ordering information, available features and electrical characteristics of derivative devices.
- **Errata Sheets**
  – list deviations from the specifications given in the related Reference Manual or Data Sheets. Errata Sheets are provided for the superset of devices.

Attention: Please consult all parts of the documentation set to attain consolidated knowledge about your device.

Application related guidance is provided by **Users Guides** and **Application Notes**.

Please refer to **http://www.infineon.com/xmc1000** to get access to the latest versions of those documents.

Related Documentations

The following documents are referenced:

- ARM® Cortex M0
  – Technical Reference Manual
  – User Guide, Reference Material
- ARM®v6-M Architecture Reference Manual
- AMBA® 3 AHB-Lite Protocol Specification
- AMBA® 3 APB Protocol Specification

Copyright Notice

Text Conventions

This document uses the following naming conventions:

- Functional units of the XMC1100 are given in plain UPPER CASE. For example: “The USIC0 unit supports…”.
- Pins using negative logic are indicated by an overline. For example: “The $\overline{\text{WAIT}}$ input has…”.
- Bit fields and bits in registers are in general referenced as “Module_RegisterName.BitField” or “Module_RegisterName.Bit”. For example: “The USIC0_PCR.MCLK bit enables the…”. Most of the register names contain a module name prefix, separated by an underscore character “_” from the actual register name (for example, “USIC0_PCR”, where “USIC0” is the module name prefix, and “PCR” is the kernel register name). In chapters describing the kernels of the peripheral modules, the registers are mainly referenced with their kernel register names. The peripheral module implementation sections mainly refer to the actual register names with module prefixes.
- Variables used to describe sets of processing units or registers appear in mixed upper and lower cases. For example, register name “MOFCRn” refers to multiple “MOFCR” registers with variable n. The bounds of the variables are always given where the register expression is first used (for example, “n = 0-31”), and are 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 register bits, or groups of pins are collectively named in the body of the document, they are represented 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: CFG[2:0] and SRPN[0].
- Units are abbreviated as follows:
  - **MHz** = Megahertz
  - **µs** = Microseconds
  - **kBaud, kbit** = 1000 characters/bits per second
  - **MBaud, Mbit** = 1,000,000 characters/bits per second
  - **Kbyte, KB** = 1024 bytes of memory
  - **Mbyte**, MB= 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. 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

Bit Function Terminology

In tables where register bits or bit fields are defined, the following conventions are used to indicate the access types.

Table 1 Bit Function Terminology

Bit Function	Description
rw	The bit or bit field can be read and written.
rwh	As rw, but bit or bit field can be also set or reset by hardware. If not otherwise documented the software takes priority in case of a write conflict between software and hardware.
r	The bit or bit field can only be read (read-only).
w	The bit or bit field can only be written (write-only). A read to this register will always give a default value back.
rh	This bit or bit field can be modified by hardware (read-hardware, typical example: status flags). A read of this bit or bit field give the actual status of this bit or bit field back. Writing to this bit or bit field has no effect to the setting of this bit or bit field.

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 2 Register Access Modes

Symbol	Description
PV (SV), U	Access permitted in Privileged (Supervisor) Mode. ***Note: ARM® Cortex M0 processor does not support different privilege levels. Only Privileged (Supervisor) Mode is supported in XMC1000 Family. Symbol "U" and Symbol "PV" can be used to represent the access permitted in this mode.***
BP	Indicates that this register can only be access when the bit protection is disabled. See detailed description of Bit Protection Scheme in the Memory Organisation chapter.

Table 2 Register Access Modes (cont'd)

Symbol	Description
32	Only 32-bit word accesses are permitted to this register/address range.
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.

Reserved Bits

Register bit fields named **Reserved** or **0** indicate unimplemented functions with the following behavior.

- Reading these bit fields returns 0.
- These bit fields should be written with 0 if the bit field is defined as r or rh.
- These bit fields have to be written with 0 if the bit field is defined as rw.

These bit fields are reserved. The detailed description of these bit fields can be found in the register descriptions.

Abbreviations and Acronyms

The following acronyms and terms are used in this document:

ACMP	Analog Comparator
AHB	Advanced High-performance Bus
AMBA	Advanced Microcontroller Bus Architecture
ANACTRL	Analog Control Unit
APB	Advanced Peripheral Bus
ASC	Asynchronous Serial Channel
BCCU	Brightness and Colour Control Unit
BMI	Boot Mode Index
CMSIS	Cortex Microcontroller Software Interface Standard
CPU	Central Processing Unit
CCU4	Capture Compare Unit 4
CCU8	Capture Compare Unit 8
CRC	Cyclic Redundancy Code
DCO	Digitally Controlled Oscillator
ECC	Error Correction Code

ERU	Event Request Unit
EVR	Embedded Voltage Regulator
FPU	Floating Point Unit
GPIO	General Purpose Input/Output
HMI	Human-Machine Interface
IIC	Inter Integrated Circuit (also known as I2C)
IIS	Inter-IC Sound Interface
I/O	Input / Output
JTAG	Joint Test Action Group = IEEE1149.1
LED	Light Emitting Diode
LEDTS	LED and Touch Sense Control Unit
MSB	Most Significant Bit
NC	Not Connected
NMI	Non-Maskable Interrupt
NVIC	Nested Vectored Interrupt Controller
ORC	Out of Range Comparator
PAU	Peripheral Access Unit
POSIF	Position Interface
PRNG	Pseudo Random Number Generator
ROM	Read-Only Memory
RAM	Random Access Memory
RTC	Real Time Clock
SCU	System Control Unit
SFR	Special Function Register
SHS	Sample and Hold Sequencer
SPI	Serial Peripheral Interface
SRAM	Static RAM
SR	Service Request
SSC	Synchronous Serial Channel
SSW	Start-up Software
TSE	Temperature Sensor
UART	Universal Asynchronous Receiver Transmitter

USIC	Universal Serial Interface Channel
VADC	Versatile Analog-to-Digital Converter
WDT	Watchdog Timer

Introduction

1 Introduction

The XMC1100 series belongs to the XMC1000 Family of industrial microcontrollers based on the ARM Cortex-M0 processor core. The XMC1100 series devices are designed for general purpose applications.

Increasing complexity and demand for computing power of embedded control applications requires microcontrollers to have a significant CPU performance, integrated peripheral functionality and rapid development environment enabling short time-to-market, without compromising cost efficiency. Nonetheless the architecture of the XMC1100 microcontroller pursue successful hardware and software concepts, which have been established in Infineon microcontroller families.

1.1 Overview

The XMC1100 series devices combine the extended functionality and performance of the Cortex-M0 core with powerful on-chip peripheral subsystems and on-chip memory units. The following key features are available within the range of XMC1100 series devices:

CPU Subsystem

- CPU Core
  - High Performance 32-bit Cortex-M0 CPU
  - Most of 16-bit Thumb instruction set
  - Subset of 32-bit Thumb2 instruction set
  - High code density with 32-bit performance
  - Single cycle 32-bit hardware multiplier
  - System timer (SysTick) for Operating System support
  - Ultra low power consumption
- Nested Vectored Interrupt Controller (NVIC)
- Event Request Unit (ERU) for programmable processing of external and internal service requests

On-Chip Memories

- 8 kbytes on-chip ROM
- 16 kbytes on-chip high-speed SRAM
- up to 64 kbytes on-chip Flash program and data memory

On-Chip Peripheral Subsystems

- Universal Serial Interface Channels (USIC), usable as UART, double-SPI, quad-SPI, IIC, IIS and LIN interfaces
- A/D Converters, up to 12 channels, includes a 12-bit analog to digital converter
- Capture/Compare Units 4 (CCU4) for use as general purpose timers
- Window Watchdog Timer (WDT) for safety sensitive applications
- Real Time Clock module with alarm support (RTC)
- System Control Unit (SCU) for system configuration and control
- Pseudo random number generator (PRNG), provides random data with fast generation times

Input/Output Lines With Individual Bit Controllability

- Tri-stated in input mode
- Push/pull or open drain output mode
- Configurable pad hysteresis

Debug System

- Access through the standard ARM serial wire debug (SWD) or the single pin debug (SPD) interface
- A breakpoint unit (BPU) supporting up to 4 hardware breakpoints
- A watchpoint unit (DWT) supporting up to 2 watchpoints

Packages Information

- PG-TSSOP-38
- PG-TSSOP-16

Note: For details about package availability for a particular derivative please check the datasheet.

1.1.1 Block Diagram

The diagram below shows the functional blocks and their basic connectivity within the XMC1100 System.

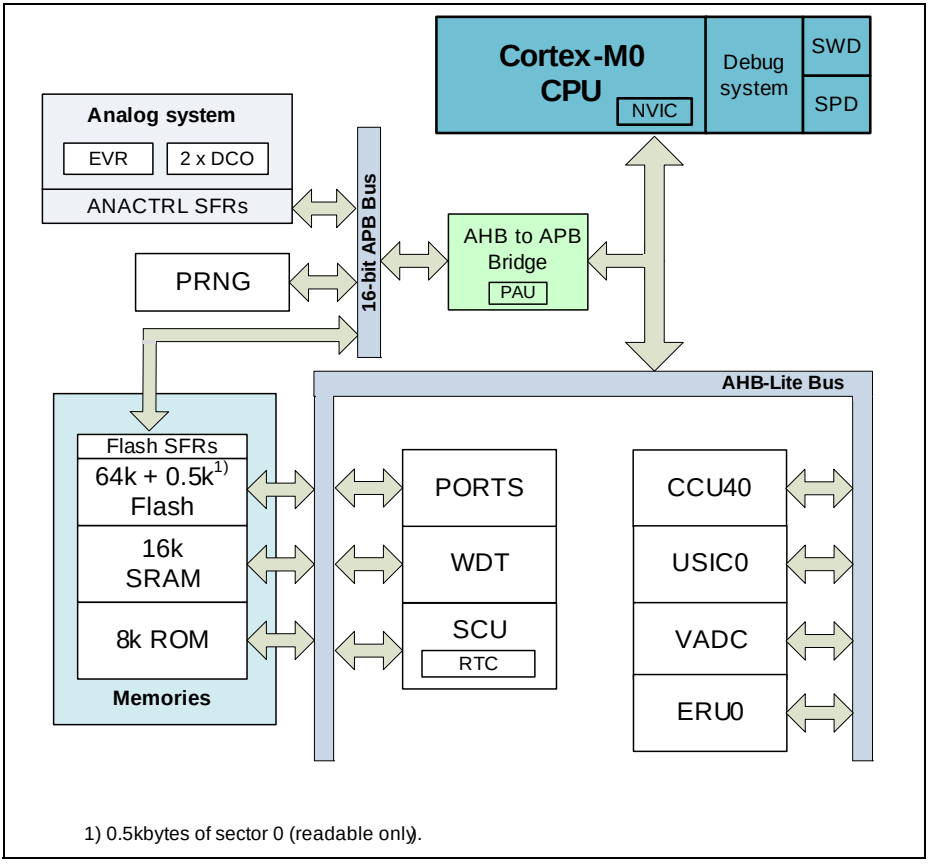


Figure 1-1 XMC1100 Functional diagram

1.2 Core Processing Units

The XMC1100 system core consists of the CPU and the memory interface blocks for memories.

1.2.1 Central Processing Unit (CPU)

The ARM Cortex-M0 processor is built on a highly area and power optimized 32-bit processor core, with a 3-stage pipeline von Neumann architecture. The processor delivers exceptional energy efficiency through a small but powerful instruction set and extensively optimized design, providing high-end processing hardware including a single cycle multiplier.

The instruction set is based on the 16-bit Thumb instrcution set and includes Thumb-2 technology. This provides the exceptional perfomance expected of a modern 32-bit architecture, with a higher code density than other 8-bit and 16-bit microcontrollers.

1.2.2 Programmable Multiple Priority Interrupt System (NVIC)

The XMC1100 provides separate interrupt nodes that may be assigned to 4 interrupt priority levels. Most interrupt sources are connected to a dedicated interrupt node. In some cases, multi-source interrupt nodes are incorporated for efficient use of system resources. These nodes can be activated by several source requests and are controlled via interrupt subnode control registers.

1.3 System Units

The XMC1100 controllers provide a number of system resources designed around the CPU.

1.3.1 Memories

8 kbytes of ROM (ROM) memory for boot code execution and exception vector table. The ROM contains system basic initialization sequence code and is executed immediately after reset release. The Bootstrap Loaders(BSL) and User Routines are also stored in the ROM.

Up to 64 kbytes of on-chip Flash memory store code or constant data. Dynamic error correction provides high read data security for all read accesses.

16 kbytes of on-chip code RAM (SRAM) are provided to store user code or data, as well as system variables such as system stack. The SRAM is accessed via the AHB and provides zero-waitstate access for CPU code execution.

1.3.2 Watchdog Timer (WDT)

The main purpose of the Window Watchdog Timer is to improve the system integrity. WDT triggers the system reset or other corrective action like e.g. an interrupt if the main program, due to some fault condition, neglects to regularly service the watchdog. The intention is to bring the system back from the unresponsive state into normal operation.

1.3.3 Real Timer Clock (RTC)

Real-time clock (RTC) is a clock that keeps track of the current time. RTCs are present in almost any electronic device which needs to keep accurate time in a digital format for clock displays and computer systems..

1.3.4 System Control unit (SCU)

The System Control Unit (SCU) handles all system cotnrol tasks besides the debug related tasks. All functions are tightly coupled and thus, they are conveniently handled by one unit, SCU. It consists of the Power Control Unit (PCU), Reset Control Unit, Clock Control Unit (CCU) and the Miscellaneous Control Unit (GCU).

The CCU generates the Main clock (MCLK) and the fast Peripheral clock (PCLK) using the 64MHz DCO1 oscillator. The PCU has a Embedded Voltage Regulator (EVR) that is used to generate the core voltage. It also provides voltage monitoring detectors to secure system performance under critical condition (eg. brownout).

1.3.5 Pseudo Random Bit Generator (PRNG)

The pseudo random bit generator (PRNG) provides random data with fast generation times.

1.4 Peripherals Units

XMC1100 offers a set of on-chip peripherals to support industrial applications.

Universal Serial Interface Channel (USIC)

The USIC is a flexible interface module covering several serial communication protocols such as ASC, LIN, SSC, I2C, I2S. A USIC module contains two independent communication channels which can be used in parallel. A FIFO allows transmit and result buffering for relaxing real-time conditions. Multiple chip select signals are available for communication with multiple devices on the same channel.

Analog to Digital Converter (VADC)

The Versatile Analog-to-Digital Converter module consists of an independent kernels which operate according to the successive approximation principle (SAR). The resolution is programmable from 8 to 12bit.

The kernel provides a versatile state machine allowing complex measurement sequences. The kernels can be synchronized and conversions may run completely in background. Multiple trigger events can be prioritized and allow the exact measurement of time critical signals. The result buffering and handling avoids data loss and ensures consistency. Self-test mechanisms can be used for plausibility checks.

The basic structure supports a clean software architecture where tasks may only read valid results and do not need to care for starting conversions.

Capture/Compare Unit 4 (CCU4)

The CCU4 peripheral is a major component for systems that need general purpose timers for signal monitoring/conditioning and Pulse Width Modulation (PWM) signal generation. Power electronic control systems like switched mode power supplies or uninterruptible power supplies can easily be implemented with the functions inside the CCU4 peripheral.

The internal modularity of CCU4 translates into a software friendly system for fast code development and portability between applications.

General Purpose I/O Ports (PORTS)

The Ports provide a generic and very flexible software and hardware interface for all standard digital I/Os. Each Port slice has individual interfaces for the operation as General Purpose I/O and it further provides the connectivity to the on-chip periphery and the control for the pad characteristics.

1.5 Debug Unit

The on-chip debug system based on the ARM Cortex-M0™ debug system provides a broad range of debug and emulation features built into the XMC1100. The user software running on the XMC1100 can thus be debugged within the target system environment.

The Debug unit is controlled by an external debugging tool via the debug interface. The debugger controls the Debug unit via a set of dedicated registers accessible via the debug interface. Additionally, the Debug unit can be controlled by the CPU, e.g. by a monitor program.

Multiple breakpoints can be triggered by on-chip hardware or by software. Single stepping is supported as well as the injection of arbitrary instructions and read/write access to the complete internal address space. A breakpoint trigger can be answered with a CPU-halt, a monitor call or a data transfer.

The data transferred at a watchpoint (see above) can be obtained via the debug interface for increased performance.

CPU Subsystem

2 Central Processing Unit (CPU)

XMC1100 features the ARM Cortex-M0 processor. An entry-level 32-bit ARM Cortex processor designed for a broad range of embedded applications. This CPU offers significant benefits to users, including:

- a simple architecture that is easy to learn and program
- ultra-low power, energy efficient operation
- excellent code density
- deterministic, high-performance interrupt handling
- upward compatibility with Cortex-M processor family

References to ARM Documentation

The following documents can be accessed through **http://infocenter.arm.com**

[1] Cortex™-M0 Devices, Generic User Guide (ARM DUI 0467B)

[2] ARMv6-M Architecture Reference Manual (ARM DDI 0419)

[3] Cortex Microcontroller Software Interface Standard (CMSIS)

References to ARM Figures

[4] **http://www.arm.com**

2.1 Overview

The Cortex-M0 processor is built on a highly area and power optimized 32-bit processor core, with a 3-stage pipeline von Neumann architecture. The processor delivers exceptional energy efficiency through a small but powerful instruction set and extensively optimized design, providing high-end processing hardware including a single-cycle multiplier.

The Cortex-M0 processor implements the ARMv6-M architecture, which is based on the 16-bit Thumb® instruction set and includes Thumb-2 technology. The Cortex-M0 instruction set provides exceptional performance expected of a modern 32-bit architecture, with a higher code density than other 8-bit and 16-bit microcontrollers.

The Cortex-M0 processor closely integrates a configurable NVIC, to deliver industry leading interrupt performance. The NVIC provides 4 interrupt priority levels. The tight integration of the processor core and NVIC provides fast execution of interrupt service routines (ISRs), dramatically reducing the interrupt latency. This is achieved through the hardware stacking of registers, and the ability to abandon and restart load-multiple and store-multiple operations. Interrupt handlers do not require any assembler wrapper code, removing any code overhead from the ISRs. Tail-chaining optimization also significantly reduces the overhead when switching from one ISR to another.

To optimize low-power designs, the NVIC integrates with the sleep modes, that include a deep sleep function that enables the entire device to be rapidly powered down.

2.1.1 Features

The CPU provides the following functionality:

- Thumb instruction set combines high code density with 32-bit performance
- integrated sleep modes for low power consumption
- fast code execution permits slower processor clock or increases sleep mode time
- single cycle 32-bit hardware multiplier
- high-performance interrupt handling for time-critical applications
- extensive debug capabilities:
  - Serial Wire Debug and Single Pin Debug reduce the number of pins required for debugging.

2.1.2 Block Diagram

The Cortex-M0 core components comprise of:

Processor Core

The CPU provides most 16-bit Thumb instruction set and subset of 32-bit Thumb2 instruction set.

Nested Vectored Interrupt Controller

The NVIC is an embedded interrupt controller that supports low latency interrupt processing.

Debug Solution

The XMC1100 implements a complete hardware debug solution.

- Single Pin Debug (SPD) or 2-pin Serial Wire Debug (SWD)
- Extensive hardware breakpoint and watchpoint options

This provides high system control and visibility of the processor and memory even in small package devices.

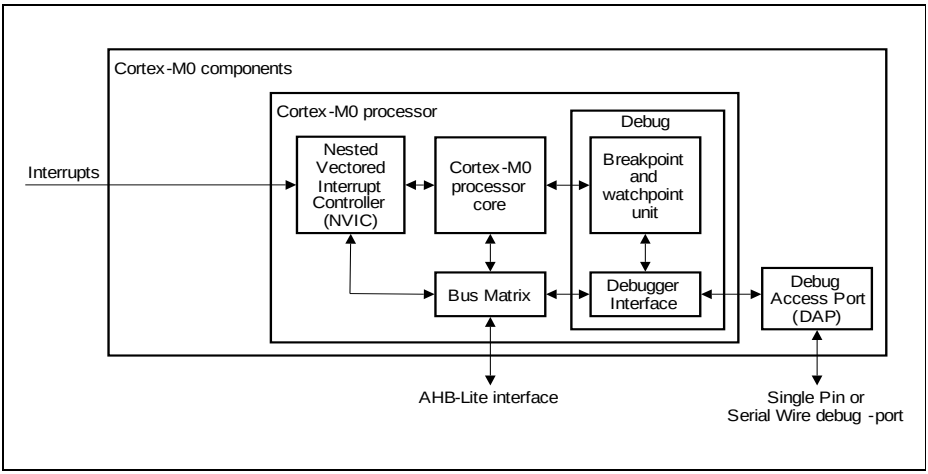


Figure 2-1 Cortex-M0 Block Diagram

System Level Interface

The Cortex-M0 processor provides a single system-level interface using AMBA® technology to provide high speed, low latency memory accesses.

2.2 Programmers Model

This section describes the Cortex-M0 programmers model. In addition to the individual core register descriptions, it contains information about the processor modes and stacks.

2.2.1 Processor Mode

The processor modes are:

- **Thread mode**
  Used to execute application software. The processor enters Thread mode when it comes out of reset.
- **Handler mode**
  Used to handle exceptions. The processor returns to Thread mode when it has finished all exception processing.

2.2.2 Stacks

The processor uses a full descending stack. This means the stack pointer holds the address of the last stacked item in memory. When the processor pushes a new item onto the stack, it decrements the stack pointer and then writes the item to the new memory

location. The processor implements two stacks, the main stack and the process stack, with a pointer for each held in independent registers, see **Stack Pointer**.

In Thread mode, the CONTROL register controls whether the processor uses the main stack or the process stack, see **CONTROL Register**. In Handler mode, the processor always uses the main stack. The options for processor operations are:

Table 2-1 Summary of processor mode, execution, and stack use options

Processor mode	Used to execute	Stack used
Thread	Applications	Main stack or process stack[1)]
Handler	Exception handlers	Main stack

1) See **CONTROL Register**.

2.2.3 Core Registers

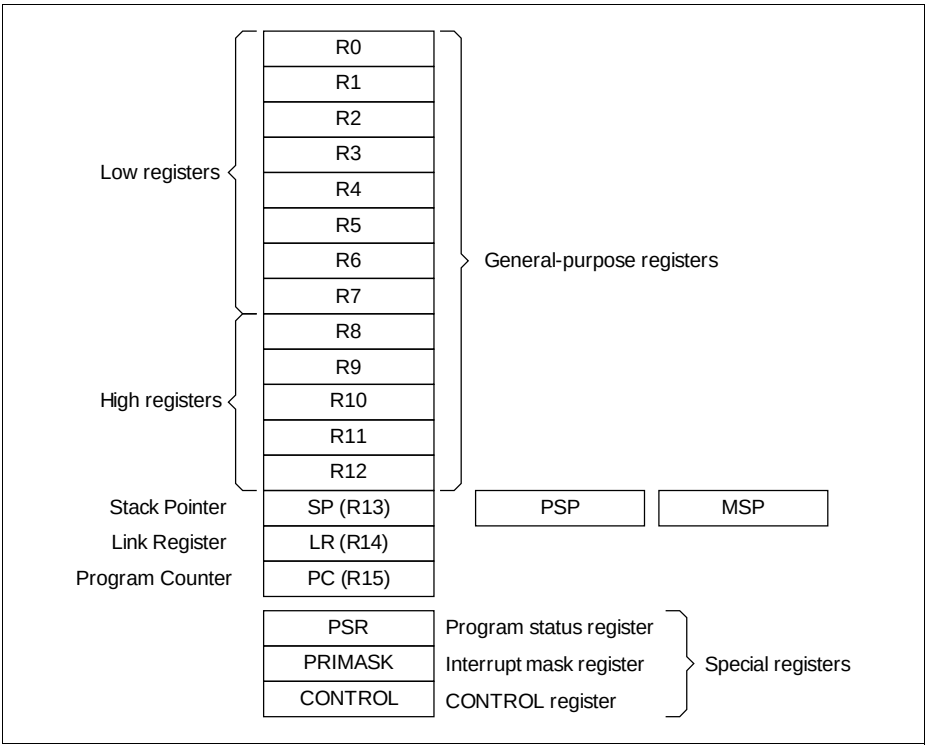


Figure 2-2 Core registers

The processor core registers are:

Table 2-2 Core register set summary

Name	Type[1)]	Reset value	Description
R0-R12	rw	Unknown	General-purpose registers on **Page 2-6**
MSP	rw	See description	Stack Pointer on **Page 2-6**
PSP	rw	Unknown	Stack Pointer on **Page 2-6**
LR	rw	Unknown	Link Register on **Page 2-7**
PC	rw	See description	Program Counter on **Page 2-8**
PSR	rw	Unknown	Program Status Register on **Page 2-8**

Table 2-2 Core register set summary (cont'd)

Name	Type[1)]	Reset value	Description
APSR	rw	Unknown	Application Program Status Register on **Page 2-9**
IPSR	r	00000000_H	Interrupt Program Status Register on **Page 2-10**
EPSR	r	Unknown	Execution Program Status Register on **Page 2-12**
PRIMASK	rw	00000000_H	Priority Mask Register on **Page 2-13**
CONTROL	rw	00000000_H	CONTROL Register on **Page 2-14**

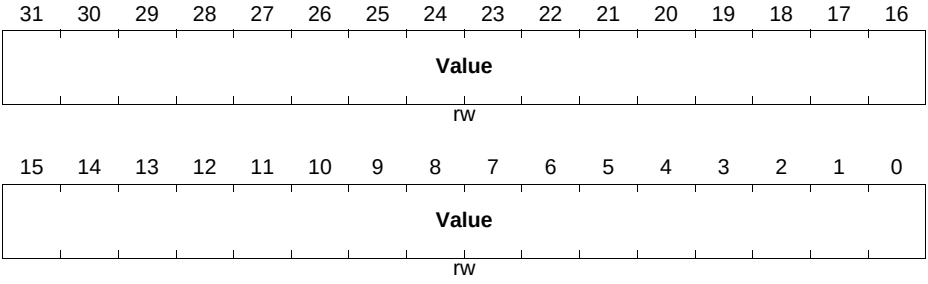
1) Describes access type during program execution in thread mode and handler mode. Debug access can differ.

General-purpose registers

R0-R12 are 32-bit general-purpose registers for data operations.

*Note: For information on how to program the core registers, please refer to the ARMv6-M Architecture Reference Manual **[2]**.*

Rx (x=0-12)
General-purpose register Rx **Reset Value: $XXXXXXXX_H$**



Field	Bits	Type	Description
Value	[31:0]	rw	**Content of Register**

Stack Pointer

The Stack Pointer (SP) is register R13. In Thread mode, bit[1] of the CONTROL register indicates the stack pointer to use:

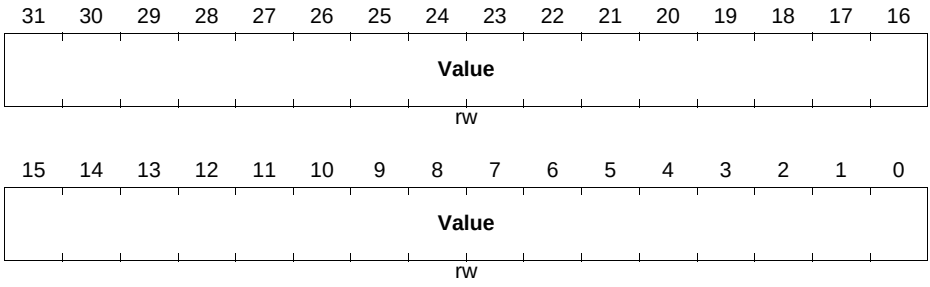
- 0 = Main Stack Pointer (MSP). This is the reset value.

- 1 = Process Stack Pointer (PSP).

On reset, the processor loads the MSP with the value from address 00000000_H.

*Note: For information on how to program the core registers, please refer to the ARMv6-M Architecture Reference Manual **[2]**.*

SP
Stack Pointer **Reset Value: 00000000_H**



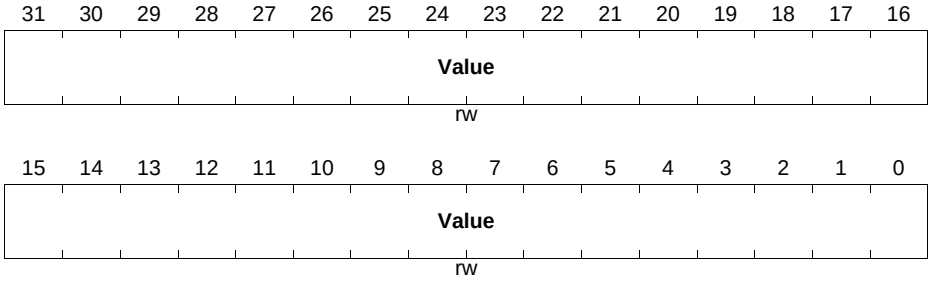
Field	Bits	Type	Description
Value	[31:0]	rw	**Content of Register**

Link Register

The Link Register (LR) is register R14. It stores the return information for subroutines, function calls, and exceptions. On reset, the LR value is unknown.

*Note: For information on how to program the core registers, please refer to the ARMv6-M Architecture Reference Manual **[2]**.*

LR
Link Register **Reset Value: $XXXXXXXX_H$**



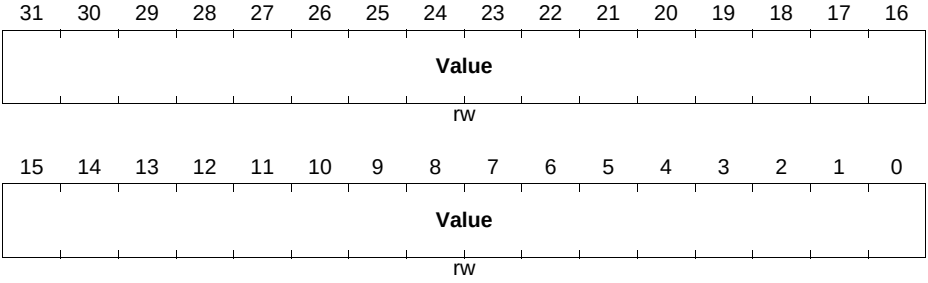
Field	Bits	Type	Description
Value	[31:0]	rw	**Content of Register**

Program Counter

The Program Counter (PC) is register R15. It contains the current program address. On reset, the processor loads the PC with the value of the reset vector, which is at address 00000004_H. Bit [0] of the value is loaded into the EPSR T-bit at reset and must be 1.

Note: For information on how to program the core registers, please refer to the ARMv6-M Architecture Reference Manual [2].

PC
Program Counter **Reset Value: 00000004_H**



Field	Bits	Type	Description
Value	[31:0]	rw	**Content of Register**

Program Status Register

The Program Status Register (PSR) combines:

- Application Program Status Register (APSR)
- Interrupt Program Status Register (IPSR)
- Execution Program Status Register (EPSR)

These registers are mutually exclusive bit fields in the 32-bit PSR.

Access these registers individually or as a combination of any two or all three registers, using the register name as an argument to the MSR or MRS instructions. For example:

- read all of the registers using PSR with the MRS instruction
- write to the APSR N, Z, C, and V bits using APSR with the MSR instruction

The PSR combinations and attributes are:

Table 2-3 PSR register combinations

Register	Type	Combination
PSR	rw[1)2)]	APSR, EPSR, and IPSR
IEPSR	r	EPSR and IPSR
IAPSR	rw[1)]	APSR and IPSR
EAPSR	rw[2)]	APSR and EPSR

1) The processor ignores writes to the IPSR bits.

2) Reads of the EPSR bits return zero, and the processor ignores writes to the these bits

Application Program Status Register

The APSR contains the current state of the condition flags from previous instruction executions. See the register summary in **Table 2-2** for its attributes.

APSR
Application Program Status Register **Reset Value: XXXXXXXX$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
N	**Z**	**C**	**V**	**0**											
rw	rw	rw	rw	rw											

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

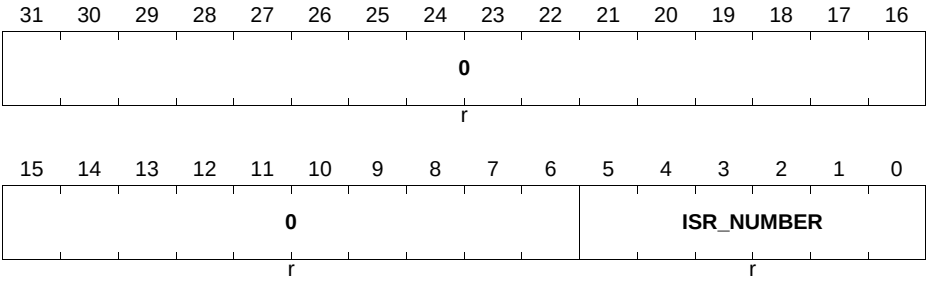
Field	Bits	Type	Description
N	31	rw	**Negative flag**
Z	30	rw	**Zero flag**
C	29	rw	**Carry or borrow flag**
V	28	rw	**Overflow flag**
0	[27:0]	r	**Reserved**

Interrupt Program Status Register

The IPSR contains the exception type number of the current Interrupt Service Routine (ISR). See the register summary in **Table 2-2** for its attributes.

IPSR

Interrupt Program Status Register **Reset Value: 00000000_H**



Field	Bits	Type	Description
0	[31:6]	r	**Reserved**

Field	Bits	Type	Description
ISR_NUMBER	[5:0]	r	**Number of the current exception** 0_D Thread mode 1_D Reserved 2_D Reserved 3_D HardFault 4_D Reserved 5_D Reserved 6_D Reserved 7_D Reserved 8_D Reserved 9_D Reserved 10_D Reserved 11_D SVCall 12_D Reserved 13_D Reserved 14_D PendSV 15_D SysTick 16_D IRQ0 ... 47_D IRQ31 48_D-63_D Reserved See Exception types in **Section 2.5.2** for more information.

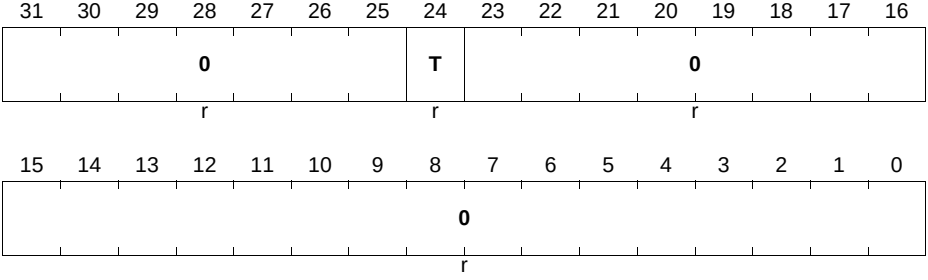
Execution Program Status Register

The EPSR contains the Thumb state bit.

See the register summary in **Table 2-2** for the EPSR attributes.

Attempts to read the EPSR directly through application software using the MSR instruction always return zero. Attempts to write the EPSR using the MSR instruction in application software are ignored. Fault handlers can examine the EPSR value in the stacked PSR to determine the cause of the fault. See Exception Entry and Return in **Section 2.5.6**.

EPSR
Execution Program Status Register **Reset Value: XXXXXXXX$_H$**



Field	Bits	Type	Description
0	[31:25]	r	**Reserved**
T	24	r	**Thumb state bit** See Thumb state.
0	[23:0]	r	**Reserved**

Interruptible-restartable instructions

When an interrupt occurs during the execution of an LDM, STM, PUSH, POP instruction, the processor abandons execution of the instruction.

After servicing the interrupt, the processor restarts execution of the instruction from the beginning.

Thumb state

The Cortex-M0 processor only supports execution of instructions in Thumb state. The following can clear the T bit to 0:

- instructions BLX, BX and POP{PC}

- restoration from the stacked xPSR value on an exception return
- bit[0] of the vector value on an exception entry.

Attempting to execute instructions when the T bit is 0 results in a HardFault or lockup. See Lockup in **Section 2.6.1** for more information.

Exception mask registers

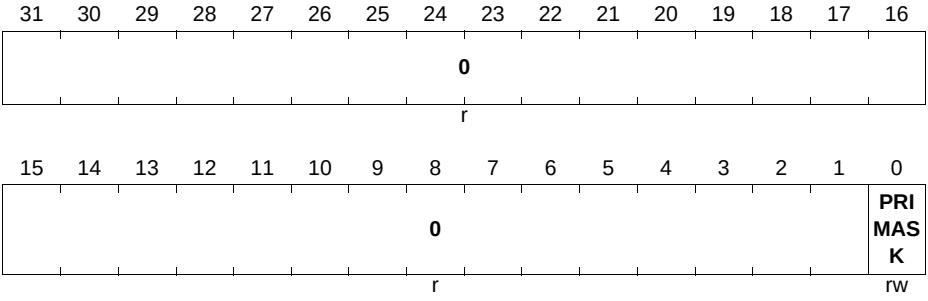
The exception mask registers disable the handling of exceptions by the processor. Disable exceptions where they might impact on timing critical tasks or code sequences requiring atomicity.

Exceptions can be disabled or re-enabled by the MSR and MRS instructions, or the CPS instruction, to change the value of PRIMASK or FAULTMASK.

Priority Mask Register

The PRIMASK register prevents activation of all exceptions with configurable priority. See the register summary in **Table 2-2** for its attributes.

PRIMASK
Priority Mask Register **Reset Value: 00000000_H**

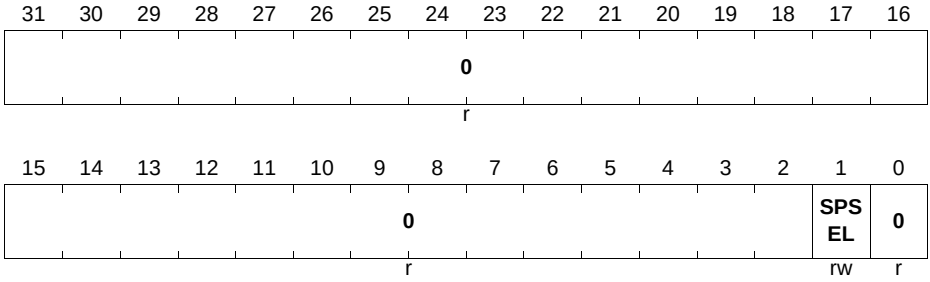


Field	Bits	Type	Description
0	[31:1]	r	**Reserved**
PRIMASK	0	rw	**Priority Mask** 0_B No effect. 1_B Prevents the activation of all exceptions with configurable priority.

CONTROL Register

The CONTROL register controls the stack used when the processor is in Thread mode. See the register summary in **Table 2-2** for its attributes.

CONTROL
CONTROL Register **Reset Value: 00000000_H**



Field	Bits	Type	Description
0	[31:2]	r	**Reserved**
SPSEL	1	rw	**Active stack pointer** This bit defines the current stack. In Handler mode, this bit reads as zero and ignores writes. 0_B MSP is the current stack pointer 1_B PSP is the current stack pointer
0	0	r	**Reserved**

Handler mode always uses the MSP, so the processor ignores explicit writes to the active stack pointer bit of the CONTROL register when in Handler mode. The exception entry and return mechanisms automatically update the CONTROL register.

In an OS environment, it is recommended that threads running in Thread mode use the process stack and the kernel and exception handlers use the main stack.

By default, Thread mode uses the MSP. To switch the stack pointer used in Thread mode to the PSP, use the MSR instruction to set the Active stack pointer bit to 1.

Note: When changing the stack pointer, software must use an ISB instruction immediately after the MSR instruction. This ensures that instructions after the ISB instruction execute using the new stack pointer.

2.2.4 Exceptions and Interrupts

The Cortex-M0 processor supports interrupts and system exceptions. The processor and the NVIC prioritize and handle all exceptions. An interrupt or exception changes the normal flow of software control. The processor uses handler mode to handle all exceptions except for reset. See Exception entry on **Section 2.5.6.1** and Exception return on **Section 2.5.6.2** for more information.

The NVIC registers control interrupt handling. See Interrupt System chapter for more information.

2.2.5 Data Types

The processor:

- supports the following data types:
  - 32-bit words
  - 16-bit halfwords
  - 8-bit bytes
- manages all data memory accesses as little-endian. See Memory regions, types and attributes in **Section 2.3.1** for more information.

2.2.6 The Cortex Microcontroller Software Interface Standard

For a Cortex-M0 microcontroller system, the Cortex Microcontroller Software Interface Standard (CMSIS) **[3]** defines:

- a common way to:
  - access peripheral registers
  - define exception vectors
- the names of:
  - the registers of the core peripherals
  - the core exception vectors
- a device-independent interface for RTOS kernels.

The CMSIS includes address definitions and data structures for the core peripherals in the Cortex-M0 processor.

CMSIS simplifies software development by enabling the reuse of template code and the combination of CMSIS-compliant software components from various middleware vendors. Software vendors can expand the CMSIS to include their peripheral definitions and access functions for those peripherals.

This document includes the register names defined by the CMSIS, and gives short descriptions of the CMSIS functions that address the processor core and the core peripherals.

Note: This document uses the register short names defined by the CMSIS. In a few cases these differ from the architectural short names that might be used in other documents.

The following sections give more information about the CMSIS:

- Power Management Programming Hints in **Section 2.7.3**
- CMSIS Functions in **Section 2.2.7**
- Accessing CPU Registers using CMSIS in Interrupt System chapter
- NVIC programming hints in Interrupt System chapter

For additional information please refer to **http://www.onarm.com/cmsis**

2.2.7 CMSIS Functions

ISO/IEC C code cannot directly access some Cortex-M0 instructions. This section describes intrinsic functions that can generate these instructions, provided by the CMSIS and that might be provided by a C compiler. If a C compiler does not support an appropriate intrinsic function, an inline assembler may be used to access the relevant instruction.

The CMSIS provides the following intrinsic functions to generate instructions that ISO/IEC C code cannot directly access:

Table 2-4 CMSIS functions to generate some Cortex-M0 instructions

Instruction	CMSIS intrinsic function
CPSIE i	void __enable_irq (void)
CPSID i	void __disable_irq (void)
ISB	void __ISB (void)
DSB	void __DSB (void)
DMB	void __DMB (void)
NOP	void __NOP (void)
REV	uint32_t __REV (uint32_t int value)
REV16	uint32_t __REV16 (uint32_t int value)
REVSH	uint32_t __REVSH (uint32_t int value)
WFE	void __WFE (void)
WFI	void __WFI (void)

The CMSIS also provides a number of functions for accessing the special registers using MRS and MSR instructions:

Table 2-5 CMSIS functions to access the special registers

Special register	Access	CMSIS function
PRIMASK	Read	uint32_t __get_PRIMASK (void)
	Write	void __set_PRIMASK (uint32_t value)
CONTROL	Read	uint32_t __get_CONTROL (void)
	Write	void __set_CONTROL (uint32_t value)
MSP	Read	uint32_t __get_MSP (void)
	Write	void __set_MSP (uint32_t TopOfMainStack)
PSP	Read	uint32_t __get_PSP (void)
	Write	void __set_PSP (uint32_t TopOfMainStack)

2.3 Memory Model

This section describes the processor memory map and the behavior of memory accesses. The processor has a fixed default memory map that provides up to 4GB of addressable memory. The memory map is:

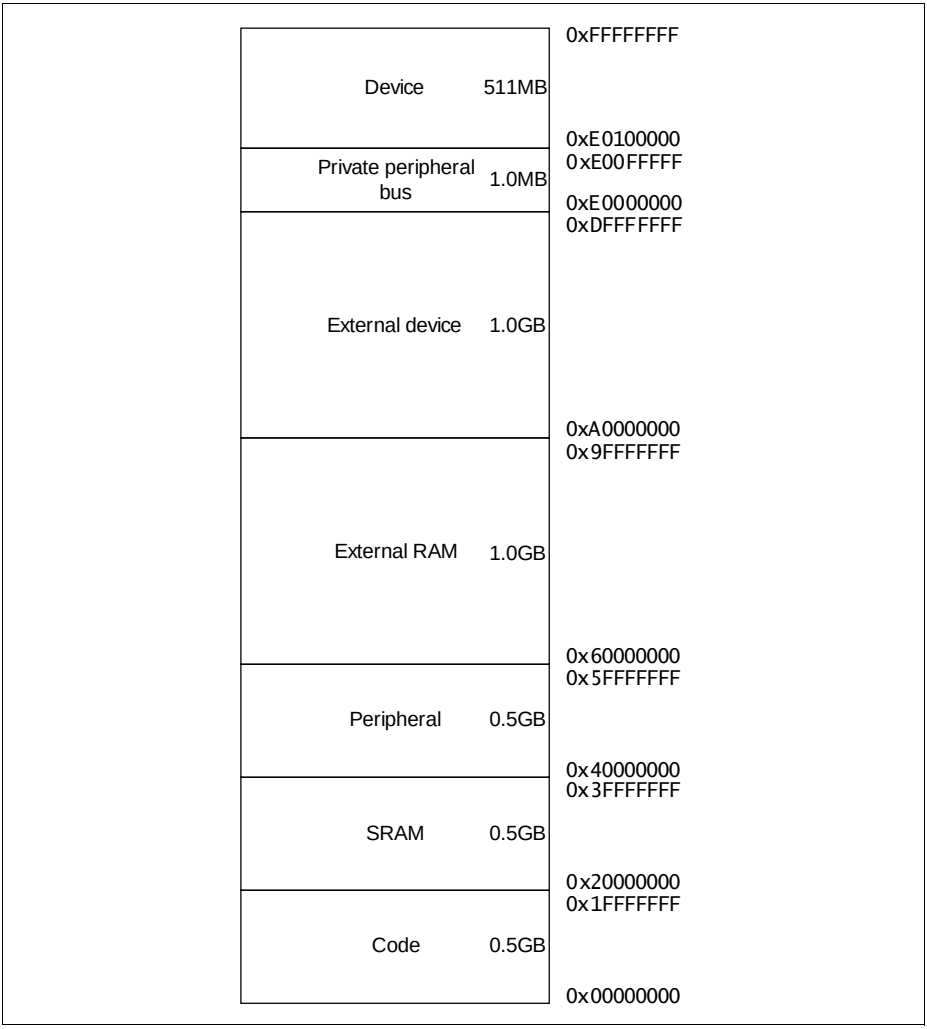


Figure 2-3 Memory map

The processor reserves regions of the Private peripheral bus (PPB) address range for core peripheral registers, see About the Private Peripherals in **Section 2.8.1**.

2.3.1 Memory Regions, Types and Attributes

The memory map is split into regions. Each region has a defined memory type, and some regions have additional memory attributes. The memory type and attributes determine the behavior of accesses to the region.

The memory types are:

Normal	The processor can re-order transactions for efficiency, or perform speculative reads.
Device	The processor preserves transaction order relative to other transactions to Device or Strongly-ordered memory.
Strongly-ordered	The processor preserves transaction order relative to all other transactions.

The different ordering requirements for Device and Strongly-ordered memory mean that the memory system can buffer a write to Device memory, but must not buffer a write to Strongly-ordered memory.

The additional memory attributes include:

Execute Never (XN)	Means the processor prevents instruction accesses. A HardFault exception is generated on execution of an instruction fetched from an XN region of memory.

2.3.2 Memory System Ordering of Memory Accesses

For most memory accesses caused by explicit memory access instructions, the memory system does not guarantee that the order in which the accesses complete matches the program order of the instructions, providing any re-ordering does not affect the behavior of the instruction sequence. Normally, if correct program execution depends on two memory accesses completing in program order, software must insert a memory barrier instruction between the memory access instructions, see Software ordering of memory accesses in **Section 2.3.4**.

However, the memory system does guarantee some ordering of accesses to Device and Strongly-ordered memory. For two memory access instructions A1 and A2, if A1 occurs before A2 in program order, the ordering of the memory accesses caused by two instructions is described in **Figure 2-4**.

A1 \ A2	Normal access	Device access	Strongly-ordered access
Normal access	-	-	-
Device access	-	<	<
Strongly-ordered access	-	<	<

Figure 2-4 Memory system ordering

Where:

- "-" Means that the memory system does not guarentee the ordering of the accesses.
- "<" Means that accesses are observed in program order, that is, A1 is always observed before A2.

2.3.3 Behavior of Memory Accesses

The behavior of accesses to each region in the memory map is:

Table 2-6 Memory access behavior

Address range	Memory region	Memory type[1)]	XN[1)]	Description
0x00000000-0x1FFFFFFF	Code	Normal	-	Executable region for program code. Data can be placed here.
0x20000000-0x3FFFFFFF	SRAM	Normal	-	Executable region for data. Code can be placed here.
0x40000000-0x5FFFFFFF	Peripheral	Device	XN	Peripherals region.
0x60000000-0x9FFFFFFF	External RAM	Normal	-	Executable region for data.
0xA0000000-0xDFFFFFFF	External device	Device	XN	External device memory.

Table 2-6 Memory access behavior (cont'd)

Address range	Memory region	Memory type[1)]	XN[1)]	Description
0xE0000000-0xE00FFFFF	Private Peripheral Bus	Strongly-ordered	XN	This region includes the NVIC, system timer, and system control block. Only word accesses can be used in this region.
0xE0100000-0xFFFFFFFF	Device	Device	XN	Vendor specific

1) See Memory regions, types and attributes in **Section 2.3.1** for more information.

The Code, SRAM, and external RAM regions can hold programs.

2.3.4 Software Ordering of Memory Accesses

The order of instructions in the program flow does not always guarantee the order of the corresponding memory transactions. This is because:

- the processor can reorder some memory accesses to improve efficiency, providing this does not affect the behavior of the instruction sequence.
- memory or devices in the memory map have different wait states
- some memory accesses are buffered or speculative.

Memory system ordering of memory accesses in **Section 2.3.2** describes the cases where the memory system guarantees the order of memory accesses. Otherwise, if the order of memory accesses is critical, software must include memory barrier instructions to force that ordering. The processor provides the following memory barrier instructions:

DMB The Data Memory Barrier (DMB) instruction ensures that outstanding memory transactions complete before subsequent memory transactions.

DSB The Data Synchronization Barrier (DSB) instruction ensures that outstanding memory transactions complete before subsequent instructions execute.

ISB The Instruction Synchronization Barrier (ISB) ensures that the effect of all completed memory transactions is recognizable by subsequent instructions.

2.3.5 Memory Endianness

The processor views memory as a linear collection of bytes numbered in ascending order from zero. For example, bytes 0-3 hold the first stored word, and bytes 4-7 hold the second stored word. **Section 2.3.5.1** describes how words of data are stored in memory.

2.3.5.1 Little-endian format

In little-endian format, the processor stores the least significant byte (lsbyte) of a word at the lowest-numbered byte, and the most significant byte (msbyte) at the highest-numbered byte. An example of the little-endian format is described in **Figure 2-5**.

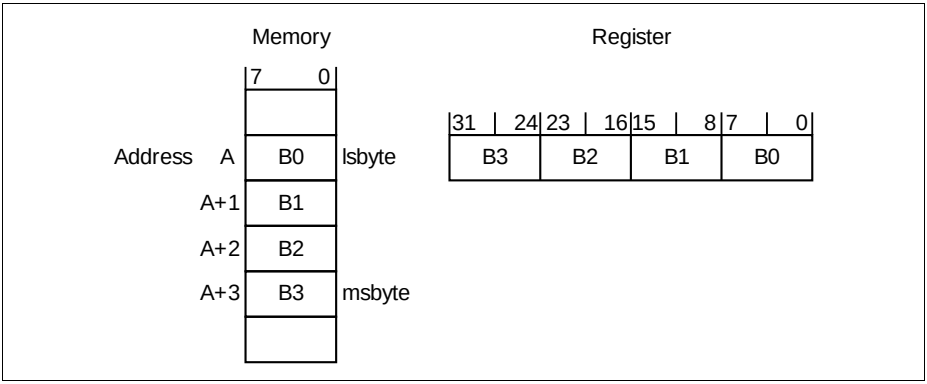


Figure 2-5 Little-endian format (Example)

2.4 Instruction Set

Table 2-7 lists the supported Cortex-M0 instructions. For more information on the instructions and operands, please refer to the Cortex™-M0 Devices, Generic User Guide available through [1].

Table 2-7 Cortex-M0 instructions

Mnemonic	Operands	Brief description	Flags
ADCS	{Rd,} Rn, Rm	Add with carry	N,Z,C,V
ADD{S}	{Rd,} Rn, <Rm|#imm>	Add	N,Z,C,V
ADR	Rd, label	PC-relative Address to Register	-
ANDS	{Rd,} Rn, Rm	Bitwise AND	N,Z
ASRS	{Rd,} Rm, <Rs|#imm>	Arithmetic Shift Right	N,Z,C
B{cc}	label	Branch {conditionally}	-
BICS	{Rd,} Rn, Rm	Bit Clear	N,Z
BKPT	#imm	Breakpoint	-
BL	label	Branch with Link	-
BLX	Rm	Branch indirect with Link	-
BX	Rm	Branch indirect	-
CMN	Rn, Rm	Compare Negative	N,Z,C,V
CMP	Rn, <Rm|#imm>	Compare	N,Z,C,V
CPSID	i	Change Processor State, Disable Interrupts	-
CPSIE	i	Change Processor State, Enable Interrupts	-
DMB	-	Data Memory Barrier	-
DSB	-	Data Synchronization Barrier	-
EORS	{Rd,} Rn, Rm	Exclusive OR	N,Z
ISB	-	Instruction Synchronization Barrier	-
LDM	Rn{!}, reglist	Load Multiple registers, increment after	-

Table 2-7 Cortex-M0 instructions (cont'd)

Mnemonic	Operands	Brief description	Flags
LDR	Rt, label	Load Register from PC-relative address	-
LDR	Rt, [Rn, <Rm|#imm>]	Load Register with word	-
LDRB	Rt, [Rn, <Rm|#imm>]	Load Reigster with byte	-
LDRH	Rt, [Rn, <Rm|#imm>]	Load Reigster with halfword	-
LDRSB	Rt, [Rn, <Rm|#imm>]	Load Reigster with signed byte	-
LDRSH	Rt, [Rn, <Rm|#imm>]	Load Reigster with signed halfword	-
LSLS	{Rd,} Rn, <Rs|#imm>	Logical Shift Left	N,Z,C
LSRS	{Rd,} Rn, <Rs|#imm>	Logical Shift Right	N,Z,C
MOV{S}	Rd, Rm	Move	N,Z
MRS	Rd, spec_reg	Move to general register from special register	-
MSR	spec_reg, Rm	Move to special register from general register	N,Z,C,V
MULS	Rd, Rn, Rm	Multiply, 32-bit result	N,Z
MVNS	Rd, Rm	Bitwise NOT	N,Z
NOP	-	No Operation	-
ORRS	{Rd,} Rn, Rm	Logical OR	N,Z
POP	reglist	Pop registers from stack	-
PUSH	reglist	Push registers onto stack	-
REV	Rd, Rm	Byte-Reverse word	-
REV16	Rd, Rm	Byte-Reverse packed halfwords	-
REVSH	Rd, Rm	Byte-Reverse signed halfword	-
RORS	{Rd,} Rn, Rs	Rotate Right	N,Z,C
RSBS	{Rd,} Rn, #0	Reverse Subtract	N,Z,C,V

Table 2-7 Cortex-M0 instructions (cont'd)

Mnemonic	Operands	Brief description	Flags
SBCS	{Rd,} Rn, Rm	Subtract with Carry	N,Z,C,V
STM	Rn!, reglist	Store Multiple registers, increment after	-
STR	Rt, [Rn, <Rm|#imm>]	Store Register as word	-
STRB	Rt, [Rn, <Rm|#imm>]	Store Register as byte	-
STRH	Rt, [Rn, <Rm|#imm>]	Store Register as halfword	-
SUB{S}	{Rd,} Rn, <Rm|#imm>	Subtract	N,Z,C,V
SVC	#imm	Supervisor Call	-
SXTB	Rd, Rm	Sign extend byte	-
SXTH	Rd, Rm	Sign extend halfword	-
TST	Rn, Rm	Logical AND based test	N,Z
UXTB	Rd, Rm	Zero extend a byte	-
UXTH	Rd, Rm	Zero extend a halfword	-
WFE	-	Wait for Event	-
WFI	-	Wait for Interrupt	-

2.4.1 Intrinsic Functions

ISO/IEC C code cannot directly access some Cortex-M0 instructions. The intrinsic functions that can generate these instructions, provided by the CMSIS and might be provided by a C compiler are described in **Section 2.2.7**.

2.5 Exception Model

This section describes the exception model. It describes:

- Exception states (**Section 2.5.1**)
- Exception types (**Section 2.5.2**)
- Exception handlers (**Section 2.5.3**)
- Vector table (**Section 2.5.4**)
- Exception priorities (**Section 2.5.5**)
- Exception entry and return (**Section 2.5.6**)

2.5.1 Exception States

Each exception is in one of the following states:

Inactive	The exception is not active and not pending.
Pending	The exception is waiting to be serviced by the processor. An interrupt request from a peripheral or from software can change the state of the corresponding interrupt to pending.
Active	An exception that is being serviced by the processor but has not completed. *Note: An exception handler can interrupt the execution of another exception handler. In this case both exceptions are in the active state.*
Active and pending	The exception is being serviced by the processor and there is a pending exception from the same source.

2.5.2 Exception Types

The exception types are described in **Table 2-8**.

Table 2-8 Exception types

Exception Types	Descriptions
Reset	Reset is invoked on power up or a warm reset. The exception model treats reset as a special form of exception. When reset is asserted, the operation of the processor stops, potentially at any point in an instruction. When reset is deasserted, execution restarts from the address provided by the reset entry in the vector table. Execution restarts in Thread mode.
HardFault	A HardFault is an exception that occurs because of an error during normal or exception processing. HardFaults have a fixed priority of -1, meaning they have higher priority than any exception with configurable priority.
SVCall	A supervisor call (SVC) is an exception that is triggered by the SVC instruction. In an OS environment, applications can use SVC instructions to access OS kernel functions and device drivers.
PendSV	PendSV is an interrupt-driven request for system-level service. In an OS environment, use PendSV for context switching when no other exception is active.
SysTick	A SysTick exception is an exception the system timer generates when it reaches zero. Software can also generate a SysTick exception. In an OS environment, the processor can use this exception as system tick.
Interrupt (IRQ)	A interrupt, or IRQ, is an exception signalled by a peripheral, or generated by a software request. All interrupts are asynchronous to instruction execution. In the system, peripherals use interrupts to communicate with the processor.

Table 2-9 Properties of the different exception types

Exception number[1)]	IRQ number[1)]	Exception type	Priority	Vector address or offset[2)]	Activation
1	-	Reset	-3, the highest	0x00000004	Asynchronous
2	-	Reserved	-	-	-
3	-13	HardFault	-1	0x0000000C	Synchronous

Table 2-9 Properties of the different exception types (cont'd)

Exception number[1)]	IRQ number[1)]	Exception type	Priority	Vector address or offset[2)]	Activation
4-10	-	Reserved	-	-	-
11	-5	SVCall	Configurable[3)]	0x0000002C	Synchronous
12-13	-	Reserved	-	-	-
14	-2	PendSV	Configurable[3)]	0x00000038	Asynchronous
15	-1	SysTick	Configurable[3)]	0x0000003C	Asynchronous
16 and above	0 and above	Interrupt (IRQ)	Configurable[3)]	0x00000040 and above[4)]	Asynchronous

1) To simplify the software layer, the CMSIS only uses IRQ numbers and therefore uses negative values for exceptions other than interrupts. The IPSR returns the Exception number, see **Interrupt Program Status Register**.

2) See Vector table in **Section 2.5.4** for more information.

3) See Interrupt Priority Registers in Interrupt System chapter.

4) Increasing in steps of 4.

For an asynchronous exception, other than reset, the processor can execute additional instructions between when the exception is triggered and when the processor enters the exception handler.

Software can disable the exceptions in **Table 2-9** which have configurable priority, see Interrupt Clear-enable Register in Interrupt System chapter.

For more information about HardFaults, see Fault handling in **Section 2.6**.

2.5.3 Exception Handlers

The processor handles exceptions using:

Interrupt Service Routines (ISRs) Interrupts IRQ0 to IRQ31 are the exceptions handled by ISRs.

Fault handlers HardFault is the only exception handled by the fault handler.

System handlers PendSV, SVCall, SysTick, and the HardFault are all system exceptions that are handled by system handlers.

2.5.4 Vector Table

The vector table contains the reset value of the stack pointer, and the start addresses, also called exception vectors, for all exception handlers. **Figure 2-6** shows the order of

the exception vectors in the vector table. The least-significant bit of each vector must be 1, indicating that the exception handler is written in Thumb code.

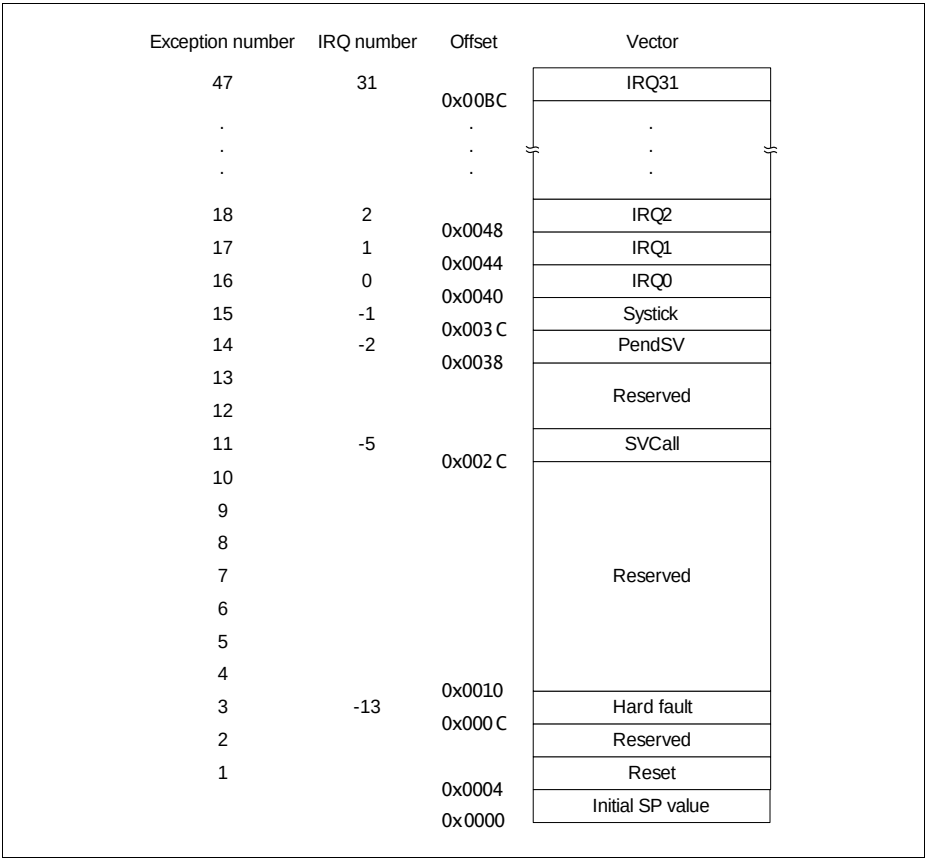


Figure 2-6 Vector table

The vector table is fixed at address 0x00000000.

2.5.4.1 Vector Table Remap

In XMC1100, the vector table is located inside the ROM. Therefore, the vector table is remapped to the SRAM based on the mapping shown in **Table 2-10**. The user application uses these locations as entry points for the actual exception and interrupt handlers. This is done by placing the code for these handlers or having the branch instruction to the handlers there.

For example, upon an exception entry due to IRQ0, the processor reads the intermediate handler start address $2000'0040_H$ (fixed in ROM) from the vector table and starts execution from there. If the actual handler is located in another address location due to size contraints, the address $2000'0040_H$ should trigger a load and a branch instruction to jump to this new location.

Note: The user application needs to reserve the SRAM addresses $2000'000C_H$ - $2000'00BF_H$ for the remapped vector table if all vectors are used.

Table 2-10 Remapped Vector Table

Exception Number	IRQ Number	Vector	Default Vector Address	Remapped Vector Address
-	-	Initial SP Value	$0000'0000_H$	$1000'1000_H$
1	-	Reset	$0000'0004_H$	$1000'1004_H$ 1)
3	-13	HardFault	$0000'000C_H$	$2000'000C_H$
11	-5	SVCall	$0000'002C_H$	$2000'002C_H$
14	-2	PendSV	$0000'0038_H$	$2000'0038_H$
15	-1	SysTick	$0000'003C_H$	$2000'003C_H$
16-47	0-31	IRQn (n=0-31)	$0000'0040_H$ + (n*4)	$2000'0040_H$ + (n*4)

1) The remapped reset vector address refers to the location (start of the Flash memory) that the startup software jumps to upon exiting the startup sequence in user mode.

2.5.5 Exception Priorities

Table 2-9 shows that all exceptions have an associated priority, with:

- a lower priority value indicating a higher priority
- configurable priorities for all exceptions except Reset and HardFault.

If software does not configure any priorities, then all exceptions with a configurable priority have a priority of 0. For information about configuring exception priorities see

- System Handler Priority Registers **SHPR2**, **SHPR3**.
- Interrupt Priority Registers in Interrupt System chapter.

Note: Configurable priority values are in the range 0-192, in steps of 64. This means that the Reset and HardFault exceptions, with fixed negative priority values, always have higher priority than any other exception.

For example, assigning a higher priority value to IRQ[0] and a lower priority value to IRQ[1] means that IRQ[1] has higher priority than IRQ[0]. If both IRQ[1] and IRQ[0] are asserted, IRQ[1] is processed before IRQ[0].

If multiple pending exceptions have the same priority, the pending exception with the lowest exception number takes precedence. For example, if both IRQ[0] and IRQ[1] are pending and have the same priority, then IRQ[0] is processed before IRQ[1].

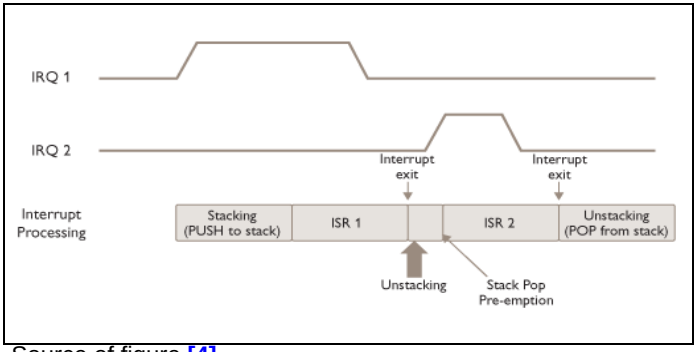
When the processor is executing an exception handler, the exception handler is preempted if a higher priority exception occurs. If an exception occurs with the same priority as the exception being handled, the handler is not preempted, irrespective of the exception number. However, the status of the new interrupt changes to pending.

2.5.6 Exception Entry and Return

Exception handling can be described using the following terms:

Preemption

When the processor is executing an exception handler, an exception can preempt the exception handler if its priority is higher than the priority of the exception being handled. When one exception preempts another, the exceptions are called nested exceptions. See Exception entry in **Section 2.5.6.1** for more information.



Source of figure **[4]**.

Return

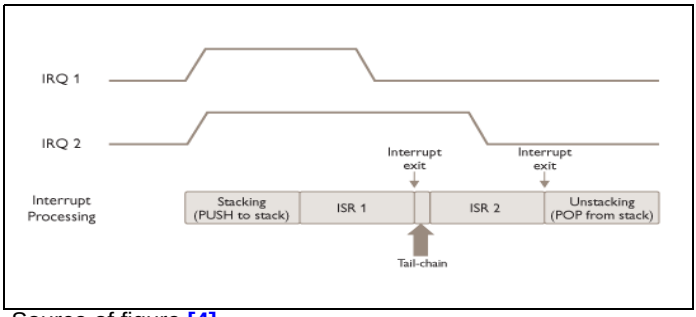
This occurs when the exception handler is completed, and:

- there is no pending exception with sufficient priority to be serviced
- the completed exception handler was not handling a late-arriving exception.

The processor pops the stack and restores the processor state to the state it had before the interrupt occurred. See Exception return in **Section 2.5.6.2** for more information.

Tail-chaining

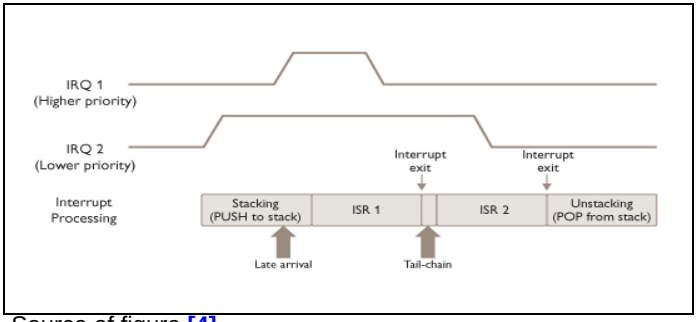
This mechanism speeds up exception servicing. On completion of an exception handler, if there is a pending exception that meets the requirements for exception entry, the stack pop is skipped and control transfers to the new exception handler.



Source of figure [4].

Late-arriving

This mechanism speeds up preemption. If a higher priority exception occurs during state saving for a previous exception, the processor switches to handle the higher priority exception and initiates the vector fetch for that exception. State saving is not affected by late arrival because the state saved is the same for both exceptions. On return from the exception handler of the late-arriving exception, the normal tail-chaining rules apply.



Source of figure [4].

2.5.6.1 Exception entry

Exception entry occurs when there is a pending exception with sufficient priority and either:

- the processor is in Thread mode
- the new exception is of higher priority than the exception being handled, in which case the new exception preempts the original exception.

When one exception preempts another, the exceptions are nested.

Sufficient priority means the exception has greater priority than any limits set by the mask register, see **Exception mask registers**. An exception with less priority than this is pending but is not handled by the processor.

When the processor takes an exception, unless the exception is a tail-chained or a late-arriving exception, the processor pushes information onto the current stack. This operation is referred to as stacking and the structure of eight data words is referred as a stack frame. The stack frame contains the following information, as illustrated in **Figure 2-7**.

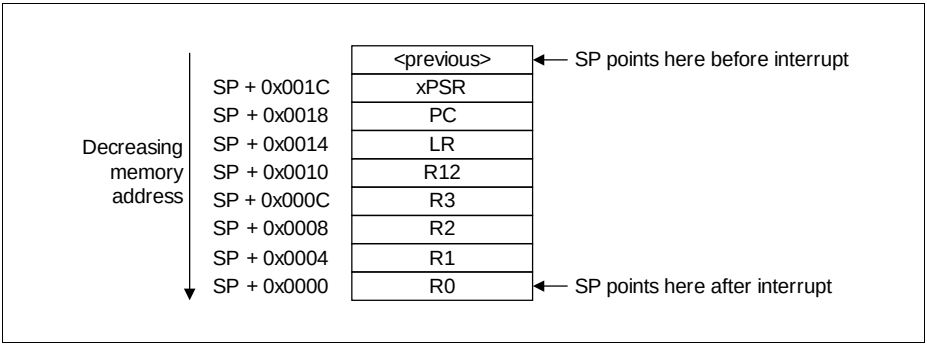


Figure 2-7 Exception stack frame

Immediately after stacking, the stack pointer indicates the lowest address in the stack frame. The stack frame is aligned to a double-word address.

The stack frame includes the return address. This is the address of the next instruction in the interrupted program. This value is restored to the PC at exception return so that the interrupted program resumes.

The processor performs a vector fetch that reads the exception handler start address from the vector table. When stacking is complete, the processor starts executing the exception handler. At the same time, the processor writes an EXC_RETURN value to the LR. This indicates which stack pointer corresponds to the stack frame and what operation mode the processor was in before the entry occurred.

If no higher priority exception occurs during exception entry, the processor starts executing the exception handler and automatically changes the status of the corresponding pending interrupt to active.

If another higher priority exception occurs during exception entry, the processor starts executing the exception handler for this exception and does not change the pending status of the earlier exception. This is the late arrival case.

2.5.6.2 Exception return

Exception return occurs when the processor is in Handler mode and execution of one of the following instructions attempts to set the PC to an EXC_RETURN value:

- a POP instruction that loads the PC
- a BX instruction using any register.

The processor saves an EXC_RETURN value to the LR on exception entry. The exception mechanism relies on this value to detect when the processor has completed an exception handler. Bits [31:4] of an EXC_RETURN value are set to 1. When this value is loaded into the PC, the processor detects that the exception is complete, and starts the exception return sequence. Bits [3:0] of the EXC_RETURN value indicate the required return stack and processor mode. **Table 2-11** shows the EXC_RETURN values with description of the exception return behavior.

Table 2-11 Exception return behavior

EXC_RETURN[31:0]	Description
0xFFFFFFF1	Return to Handler mode. Exception return gets state from the main stack. Execution uses MSP after return.
0xFFFFFFF9	Return to Thread mode. Exception return gets state from MSP. Execution uses MSP after return.
0xFFFFFFFD	Return to Thread mode. Exception return gets state from the PSP. Execution uses PSP after return.
All other values	Reserved.

2.6 Fault Handling

Faults are a subset of the exceptions, see Exception model in **Section 2.5**. All faults result in the HardFault exception being taken or cause lockup if they occur in the HardFault handler. The faults are:

- execution of an SVC instruction at a priority equal or higher than SVCall
- execution of a BKPT instruction without a debugger attached
- a system-generated bus error on a load or store
- execution of an instruction from an XN memory address
- execution of an instruction from a location for which the system generates a bus fault
- a system-generated bus error on a vector fetch
- execution of an undefined instruction
- execution of an instruction when not in Thumb-State as a result of the T-bit being previously cleared to 0
- an attempted load or store to an unaligned address

Note: Only Resetcan preempt the fixed priority HardFault handler. A HardFault can preempt any exception other than Reset, or another hard fault.

2.6.1 Lockup

The processor enters a lockup state if a fault occurs when executing the HardFault handlers, or if the system generates a bus error when unstacking the PSR on an exception return using the MSP. When the processor is in lockup state it does not execute any instructions. The processor remains in lockup state until one of the following occurs:

- it is reset
- it is halted by a debugger

2.7 Power Management

The Cortex-M0 processor sleep modes reduce power consumption:

- Sleep mode
- Deep sleep mode

The SLEEPDEEP bit of the SCR selects which sleep mode is used, see System Control Register **SCR**.

This section describes the mechanisms for entering sleep mode, and the conditions for waking up from sleep mode.

2.7.1 Entering Sleep Mode

This section describes the mechanisms software can use to put the processor into sleep mode.

The system can generate spurious wakeup events, for example a debug operation wakes up the processor. Therefore software must be able to put the processor back into sleep mode after such an event. A program might have an idle loop to put the processor back to sleep mode.

Wait for interrupt

The wait for interrupt instruction, WFI, causes immediate entry to sleep mode. When the processor executes a WFI instruction it stops executing instructions and enters sleep mode.

Wait for event

The wait for event instruction, WFE, causes entry to sleep mode depending on the value of a one-bit event register. When the processor executes a WFE instruction, it checks the value of the event register:

0 The processor stops executing instructions and enters sleep mode.
1 The processor clears the register to 0 and continues executing instructions without entering sleep mode.

If the event register is 1, this indicate that the processor must not enter sleep mode on execution of a WFE instruction. Typically, this is because an external event is asserted. Software cannot access this register directly.

Sleep-on-exit

If the SLEEPONEXIT bit of the SCR is set to 1, when the processor completes the execution of an exception handler and returns to Thread mode, it immediately enters sleep mode. This mechanism is used in applications that only require the processor to run when an interrupt occurs.

2.7.2 Wakeup from Sleep Mode

The conditions for the processor to wakeup depend on the mechanism that caused it to enter sleep mode.

Wakeup from WFI or Sleep-on-exit

The following events are WFI wake-up events:

- reset event
- debug event, if debug is enabled
- exception at a priority that would preempt any currently active exceptions, if PRIMASK was set to 0

*Note: If PRIMASK is set to 1, an interrupt or exception that has a higher priority than the current exception priority will cause the processor to wake up. Interrupt handler is not executed until the processor sets PRIMASK to 0. For more information about PRIMASK, see **Exception mask registers**.*

Wakeup from WFE

The following events are WFE wake-up events:

- reset event
- exception or interrupt with sufficient priority to cause exception entry
- exception or interrupt entering pending state, if SEVONPEND is set to 1 (See **SCR**)
- debug event, if debug is enabled

Note: External events are not supported in XMC1100. Nevertheless, SEV instruction will set the event register.

2.7.3 Power Management Programming Hints

ISO/IEC C cannot directly generate the WFI and WFE instructions. The CMSIS provides the following functions for these instructions:

```
void __WFE(void)  // Wait for Event
void __WFI(void)   // Wait for Interrupt
```

2.8 Private Peripherals

The following sections are the reference material for the ARM Cortex-M0 core peripherals.

2.8.1 About the Private Peripherals

The address map of the Private Peripheral Bus (PPB) is:

Table 2-12 Core peripheral register regions

Address	Core peripheral	Description
0xE000E008-0xE000E00F	System Control Block	See **Section 2.8.2** and **Section 2.9.1**
0xE000E010-0xE000E01F	System timer	See **Section 2.8.3** and **Section 2.9.2**
0xE000E100-0xE000E4EF	Nested Vectored Interrupt Controller	See Interrupt System chapter
0xE000ED00-0xE000ED3F	System Control Block	See **Section 2.8.2** and **Section 2.9.1**
0xE000EF00-0xE000EF03	Nested Vectored Interrupt Controller	See Interrupt System chapter

2.8.2 System control block

The System Control Block (SCB) provides system implementation information, and system control. This includes configuration, control, and reporting of the system exceptions.

2.8.2.1 System control block usage hints and tips

Ensure software uses aligned 32-bit word size transactions to access all the system control block registers.

2.8.3 System timer, SysTick

The processor has a 24-bit system timer, SysTick, that counts down from the reload value to zero, reloads, that is wraps to, the value in the SYST_RVR register on the next clock cycle, then counts down on subsequent clock cycles.

Note: When the processor is halted for debugging the counter does not decrement.

2.8.3.1 SysTick usage hints and tips

The interrupt controller clock updates the SysTick count. When processor clock is selected and the clock signal is stopped for low power mode, the SysTick counter stops. When external clock is selected, the clock continues to run in low power mode and SysTick can be used as a wakeup source.

Ensure software uses aligned word accesses to access the SysTick registers.

If the SysTick counter reload and current value are undefined at reset, the correct initialization sequence for the SysTick counter is:

1. Program reload value.
2. Clear current value.
3. Program Control and Status register.

2.9 PPB Registers

The CPU private peripherals registers base address is $E000E000_H$.

Table 2-13 Register Overview

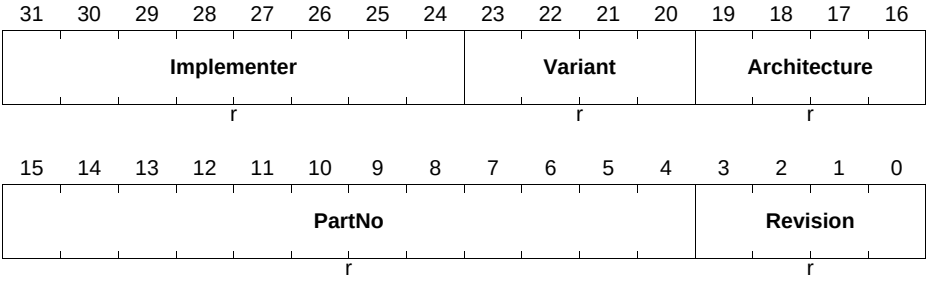
Short Name	Description	Offset Address	Access Mode		Description See
			Read	Write	
System Control Space (SCS)					
CPUID	CPUID Base Register	$D00_H$	PV, 32	PV, 32	**Page 2-41**
ICSR	Interrupt Control and State Register	$D04_H$	PV, 32	PV, 32	**Page 2-42**
AIRCR	Application Interrupt and Reset Control Register	$D0C_H$	PV, 32	PV, 32	**Page 2-45**
SCR	System Control Register	$D10_H$	PV, 32	PV, 32	**Page 2-46**
CCR	Configuration and Control Register	$D14_H$	PV, 32	PV, 32	**Page 2-48**
SHPR2	System Handler Priority Register 2	$D1C_H$	PV, 32	PV, 32	**Page 2-49**
SHPR3	System Handler Priority Register 3	$D20_H$	PV, 32	PV, 32	**Page 2-50**
SHCSR	System Handler Control and State Register	$D24_H$	PV, 32	PV, 32	**Page 2-51**
System Timer (SysTick)					
SYST_CSR	SysTick Control and Status Register	010_H	PV, 32	PV, 32	**Page 2-52**
SYST_RVR	SysTick Reload Value Register	014_H	PV, 32	PV, 32	**Page 2-54**
SYST_CVR	SysTick Current Value Register	018_H	PV, 32	PV, 32	**Page 2-55**
SYST_CALIB	SysTick Calibration Value Register	$01C_H$	PV, 32	-	**Page 2-56**

2.9.1 SCS Registers

CPUID

The CPUID register contains the processor part number, version, and implementation information.

CPUID
CPUID Base Register **(E000ED00$_H$)** **Reset Value: 410CC200$_H$**



Field	Bits	Type	Description
Revision	[3:0]	r	**Revision Number** 0_H Patch 0
PartNo	[15:4]	r	**Part Number of the Processor** $C20_H$ Cortex-M0
Architecture	[19:16]	r	**Architecture** C_H ARMv6-M
Variant	[23:20]	r	**Variant Number** 0_H Revision 0
Implementer	[31:24]	r	**Implementer Code** 41_H ARM

ICSR

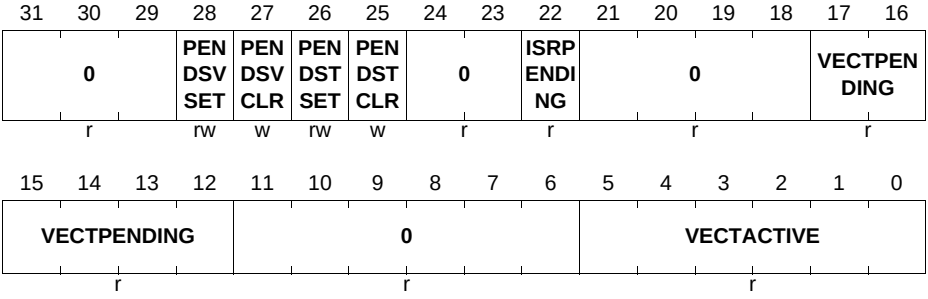
The ICSR:

- provides:
  - set-pending and clear-pending bits for the PendSV and SysTick exceptions
- indicates:
  - the exception number of the exception being processed
  - whether there are preempted active exceptions
  - the exception number of the highest priority pending exception
  - whether any interrupts are pending.

ICSR

Interrupt Control and State Register

(E000ED04$_H$) **Reset Value: 00000000$_H$**



Field	Bits	Type	Description
VECTACTIVE[1)]	[5:0]	r	**Active Exception Number** 00_H Thread mode Non-zero value The exception number of the currently active exception. *Note: Subtract 16 from this value to obtain the CMSIS IRQ number required to index into the Interrupt Clear-Enable, Set-Enable, Clear-Pending, Set-Pending, or Priority Registers, see **Interrupt Program Status Register**.*
0	[11:6]	r	**Reserved** Read as 0; should be written with 0.

Field	Bits	Type	Description
VECTPENDING	[17:12]	r	**Pending Exception Number** Indicates the exception number of the highest priority pending enabled exception. 0_H No pending exceptions Non-zero value: The exception number of the highest priority pending enabled exception.
0	[21:18]	r	**Reserved** Read as 0; should be written with 0.
ISRPENDING	22	r	**Interrupt Pending Flag** This bit sets the interrupt pending flag, excluding faults. 0_B Interrupt not pending 1_B Interrupt pending.
0	[24:23]	r	**Reserved** Read as 0; should be written with 0.
PENDSTCLR	25	w	**SysTick Exception Clear-pending** 0_B No effect 1_B removes the pending state from the SysTick exception. This bit is write-only. On a register read, this value is unknown.
PENDSTSET	26	rw	**SysTick Exception Set-pending** 0_D SysTick exception is not pending 1_D SysTick exception is pending. A write of 0 to the bit has no effect.
PENDSVCLR	27	w	**PendSV Clear Pending** This bit clears a pending PendSV exception. 0_B Do not clear. 1_B Removes pending state from PendSV exception.
PENDSVSET	28	rw	**PendSV Set Pending** This bit sets a pending PendSV exception or reads back the current state. 0_B PendSV exception is not pending. 1_B PendSV excepton is pending. *Note: Writing 1 to this bit is the only way to set the PendSV exception state to pending.* A software write of 0 to the bit has no effect.

Field	Bits	Type	Description
0	[31:29]	r	**Reserved** Read as 0; should be written with 0.

1) This is the same value as IPSR bits[5:0], see **Interrupt Program Status Register**.

Note: The result is unpredictable if:

1. *Both PENDSVSET and PENDSVCLR bits are set to 1.*
2. *Both PENDSTSET and PENDSTCLR bits are set to 1.*

AIRCR

The AIRCR register provides endian status for data accesses and reset control of the system. To write to this register, you must write 0x5FA to the VECTKEY field, otherwise the processor ignores the write.

AIRCR
Application Interrupt and Reset Control Register

(E000ED0C$_H$) **Reset Value: FA050000$_H$**

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

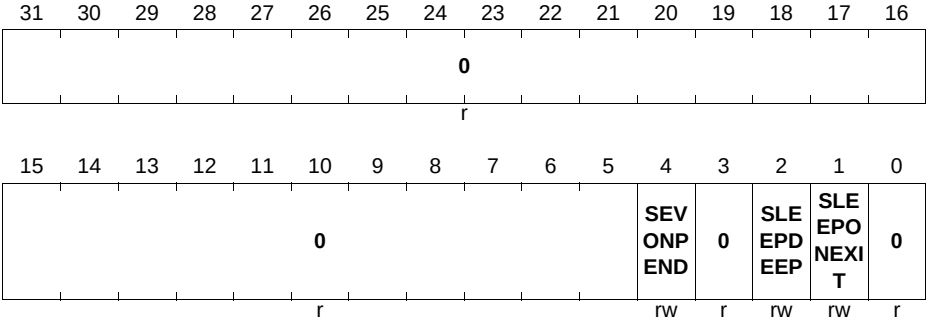
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENDIANNESS	0												SYSRESETREQ	0	0
r	r												w	w	r

Field	Bits	Type	Description
0	0	r	**Reserved** Read as 0; should be written with 0.
0	1	w	**Reserved** Must be written with 0.
SYSRESETREQ	2	w	**System Reset Request** 0_B No effect. 1_B Requests a system level reset. This bit is read as 0.
0	[14:3]	r	**Reserved** Read as 0; should be written with 0.
ENDIANNESS	15	r	**Data Endianness** 0_B Little-endian
VECTKEY	[31:16]	rw	**Register Key** Reads as unknown. On writes, write 0x5FA to VECTKEY, otherwise the write is ignored.

SCR

The SCR controls features of entry to and exit from low power state.

SCR
System Control Register (E000ED10$_H$) Reset Value: 00000000$_H$



Field	Bits	Type	Description
0	0	r	**Reserved** Read as 0; should be written with 0.
SLEEPONEXIT	1	rw	**Sleep-on-exit** This bit indicates sleep-on-exit when returning from Handler mode to Thread mode. 0_B Do not sleep when returning to Thread mode. 1_B Enter sleep, or deep sleep, on return from an ISR to Thread mode. Setting this bit to 1 enables an interrupt driven application to avoid returning to an empty main application.
SLEEPDEEP	2	rw	**Low Power Sleep Mode** This bit controls whether the processor uses sleep or deep sleep as its low power mode. 0_B Sleep 1_B Deep sleep
0	3	r	**Reserved** Read as 0; should be written with 0.

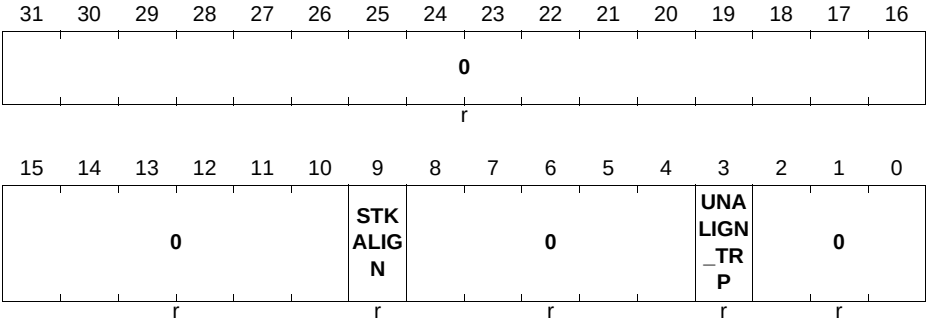
Field	Bits	Type	Description
SEVONPEND	4	rw	**Send Event on Pending bit** 0_B Only enabled interrupts or events can wakeup the processor, disabled interrupts are excluded. 1_B Enabled events and all interrupts, including disabled interrupts, can wakeup the processor. When an event or interrupt enters pending state, the event signal wakes up the processor from WFE. If the processor is not waiting for an event, the event is registered and affects the next WFE.
0	[31:5]	r	**Reserved** Read as 0; should be written with 0.

CCR

The CCR is a read-only register and it indicates some aspects of the behavior of the Cortex-M0 processor.

CCR
Configuration and Control Register

(E000ED14$_H$) **Reset Value: 00000208$_H$**



Field	Bits	Type	Description
0	[2:0]	r	**Reserved** Read as 0; should be written with 0.
UNALIGN_TRP	3	r	**Unaligned Access Traps** This bit always reads as 1, indicates that all unaligned accesses generate a HardFault.
0	[8:4]	r	**Reserved** Read as 0; should be written with 0.
STKALIGN	9	r	**Stack Alignment** This bit always reads as 1, indicates 8-byte stack alignment on exception entry. On exception entry, the processor uses bit [9] of the stacked PSR to indicate the stack alignment. On return from the exception, it uses this stacked bit to restore the correct stack alignment.
0	[31:10]	r	**Reserved** Read as 0; should be written with 0.

System Handler Priority Registers

The SHPR2-SHPR3 registers set the priority level, 0 to 192, of the exception handlers that have configurable priority.

SHPR2-SHPR3 are word accessible. To access to the system exception priority level using CMSIS, the following CMSIS functions are used:

- `uint32_t NVIC_GetPriority(IRQn_Type IRQn)`
- `void NVIC_SetPriority(IRQn_Type IRQn, uint32_t priority)`

The system fault handlers, the priority field and register for each handler are:

Table 2-14 System fault handler priority fields

Handler	Field	Register description
SVCall	PRI_11	System Handler Priority Register 2 on **Page 2-49**
PendSV	PRI_14	System Handler Priority Register 3 on **Page 2-50**
SysTick	PRI_15	

Each PRI_N field is 8 bits wide, but the XMC1100 implements only bits [7:6] of each field, and bits [5:0] read as zero and ignore writes.

SHPR2

The SHPR2 register sets the priority level for the SVCall handler.

SHPR2
System Handler Priority Register 2

(E000ED1C$_H$) **Reset Value: 00000000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRI_11								0							
rw								r							

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

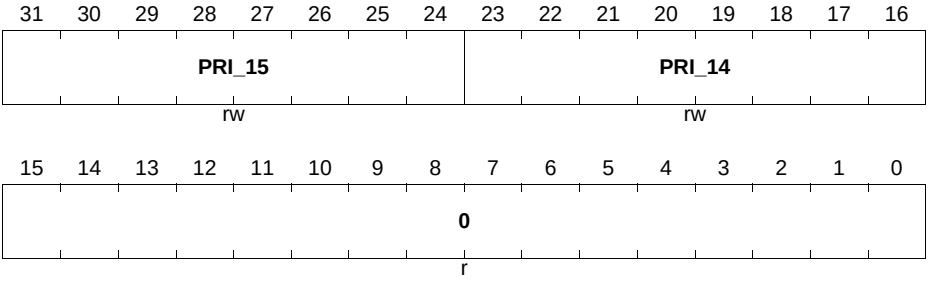
Field	Bits	Type	Description
0	[23:0]	r	**Reserved** Read as 0; should be written with 0.
PRI_11	[31:24]	rw	**Priority of System Handler 11** SVCall.

SHPR3

The SHPR3 register sets the priority level for the PendSV and SysTick handlers.

SHPR3
System Handler Priority Register 3

(E000ED20$_H$) **Reset Value: 00000000$_H$**



Field	Bits	Type	Description
0	[15:0]	r	**Reserved** Read as 0; should be written with 0.
PRI_14	[23:16]	rw	**Priority of System Handler 14** PendSV.
PRI_15	[31:24]	rw	**Priority of System Handler 15** SysTick exception.

SHCSR

The SHCSR register controls and provides the status of system handlers.

SHCSR
System Handler Control and State Register

(E000ED24$_H$) **Reset Value: 00000000$_H$**

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
SVCALLPENDED								0							
rw								r							

Field	Bits	Type	Description
0	[14:0]	r	**Reserved** Read as 0; should be written with 0.
SVCALLPENDED	15	rw	**SVCall Pending bit** This bit reflects the pending state on a read, and updates the pending state, to the value written, on a write. 0_B SVCall is not pending. 1_B SVCall is pending[1].
0	[31:16]	r	**Reserved** Read as 0; should be written with 0.

1) Pending state bits are set to 1 when an exception occurs, and are cleared to 0 when an exception becomes active.

2.9.2 SysTick Registers

SYST_CSR

The SYST_CSR register enables the SysTick features.

Reading SYST_CSR clears the COUNTFLAG bit to 0.

SYST_CSR

SysTick Control and Status Register

(E000E010$_H$) **Reset Value: 00000000$_H$**

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0													CLKSOURCE	TICKINT	ENABLE
r													rw	rw	rw

Field	Bits	Type	Description
ENABLE	0	rw	**Counter Enable** This bit enables the counter. 0_B Counter disabled. 1_B Counter enabled.
TICKINT	1	rw	**SysTick Exception Request** This bit enables the SysTick exception request. 0_B Counting down to zero does not assert the SysTick exception request. 1_B Counting down to zero to assert the SysTick exception request. In software, COUNTFLAG bit can be used to determine if SysTick has counted to zero.
CLKSOURCE	2	rw	**Clock Source** This bit selects the SysTick timer clock source. 0_B External clock[1)]. 1_B Processor clock.

Field	Bits	Type	Description
0	[15:3]	r	**Reserved** Read as 0; should be written with 0.
COUNTFLAG	16	rw	**Counter Flag** This bit returns 1 if timer counted to 0 since the last read of this register.
0	[31:17]	r	**Reserved** Read as 0; should be written with 0.

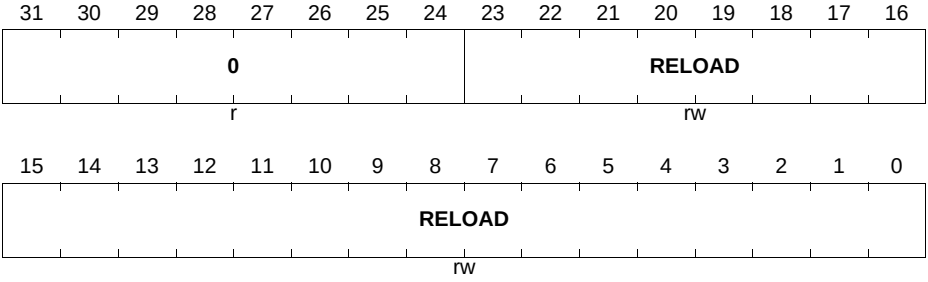
1) In XMC1100, the external clock refers to the on-chip 32 kHz standby clock.

When ENABLE is set to 1, the counter loads the RELOAD value from the SYST_RVR register and then counts down. On reaching 0, it sets the COUNTFLAG to 1 and optionally asserts the SysTick depending on the value of TICKINT. It then loads the RELOAD value again, and begins counting.

SYST_RVR

The SYST_RVR register specifies the start value to load into the SYST_CVR register.

SYST_RVR
SysTick Reload Value Register (E000E014$_H$) Reset Value: XXXXXXXX$_H$



Field	Bits	Type	Description
RELOAD	[23:0]	rw	**Reload Value** This field sets the value to load into the SYST_CVR register when the counter is enabled and when it reaches 0.
0	[31:24]	r	**Reserved** Read as 0; should be written with 0.

Notes on calculating the RELOAD value

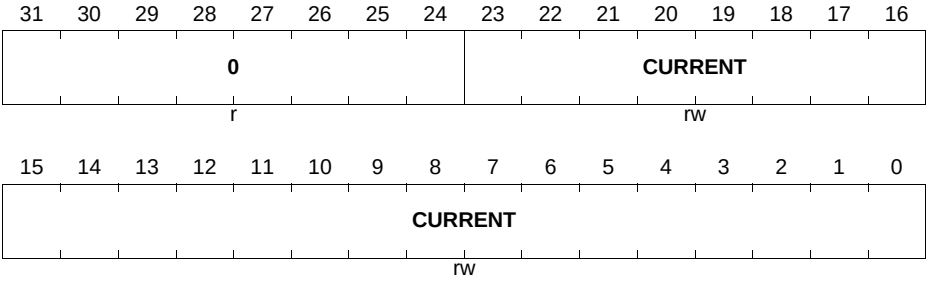
1. *The RELOAD value can be any value in the range 0x00000001-0x00FFFFFF. A start value of 0 is possible, but this has no effect because the SysTick exception request and COUNTFLAG are activated when counting from 1 to 0.*
2. *The RELOAD value is calculated according to its use. For example, to generate a multi-shot timer with a period of N processor clock cycles, use a RELOAD value of N-1. If the SysTick interrupt is required every 100 clock pulses, set RELOAD to 99.*

SYST_CVR

The SYST_CVR register contains the current value of the SysTick counter.

Writing to the SYST_CVR clears the register and the COUNTFLAG status bit to 0. The write does not trigger the SysTick exception logic. Reading the register returns its value at the time it is accessed.

SYST_CVR
SysTick Current Value Register (E000E018$_H$) Reset Value: XXXXXXXX$_H$



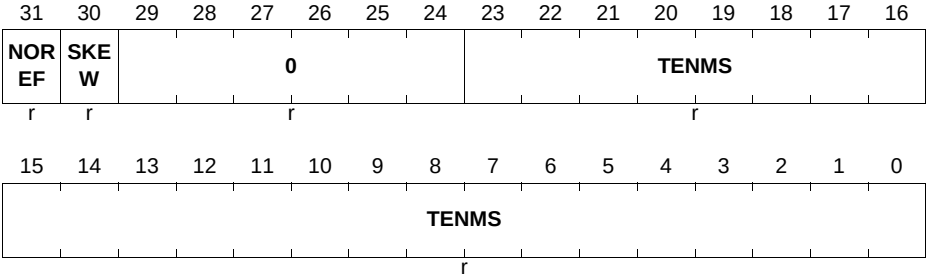
Field	Bits	Type	Description
CURRENT	[23:0]	rw	**SysTick Counter Current Value** When read, it returns the current value of the SysTick counter. A write of any value clears the field to 0, and also clears the SYST_CSR.COUNTFLAG bit to 0.
0	[31:24]	r	**Reserved** Read as 0; should be written with 0.

SYST_CALIB

The SYST_CALIB register indicates the SysTick calibration properties.

SYST_CALIB

SysTick Calibration Value Register(E000E01C$_H$) **Reset Value: 40000147$_H$**



Field	Bits	Type	Description
TENMS	[23:0]	r	**10 Milliseconds** The reload value for 10ms timing is subject to system clock skew errors. The default value of TENMS is 0x000147.
0	[29:24]	r	**Reserved** Read as 0; should be written with 0.
SKEW	30	r	**Clock Skew** This bit is read as 1. It indicates that 10ms calibration value is inexact, because of the clock frequency.
NOREF	31	r	**Reference Clock** This bit is read as 0. It indicates that external reference clock is provided.

3 Bus System

The single master bus system in XMC1100 consists of a high-performance system bus based on the industry AMBA 3 AHB-Lite Protocol standard for memories and high-bandwidth on-chip peripherals and a narrower APB for low-bandwidth on-chip peripherals.

3.1 Bus Interfaces

This chapter describes the features for the two kinds of interfaces.

- Memory Interface
- Peripheral Interface

All on-chip modules implement Little Endian data organization.

Memory Interface

The on-chip memories are capable to accept a transfer request with each bus clock cycle.

The memory interface data bus width is 32-bit. Flash memory supports only 32-bit accesses while SRAM allows 32-bit, 16-bit and 8-bit write accesses. Read accesses to SRAM is always 32-bit wide.

Peripheral Interface

Each slave on the AHB-Lite supports 32-bit accesses. Additionally:

- USIC0 supports 8-bit and 16-bit accesses

Each slave on the APB supports only 16-bit accesses.

Note: Unaligned memory accesses to memory or peripheral slaves result in a HardFault exception.

4 Service Request Processing

A hardware pulse is called Service Request (SR) in an XMC1100 system. Service Requests are the fastest way to send trigger “messages” between connected on-chip resources.

An SR can generate any of the following requests

- Interrupt
- Peripheral action

This chapter describes the available Service Requests and the different ways to select and process them.

Table 4-1 Abbreviations

ERU	Event Request Unit
NVIC	Nested Vectored Interrupt Controller
SR	Service Request

4.1 Overview

Efficient Service Request Processing is based on the interconnect between the request sources and the request processing units. XMC1100 provides both fixed and programmable interconnect.

4.1.1 Features

The following features are provided for Service Request processing:

- Connectivity matrix between Service Requests and request processing units
  - Fixed connections
  - Programmable connections using ERU

4.1.2 Block Diagram

Figure 4-1 shows a representation of the interaction between the request sources and the request processing units.

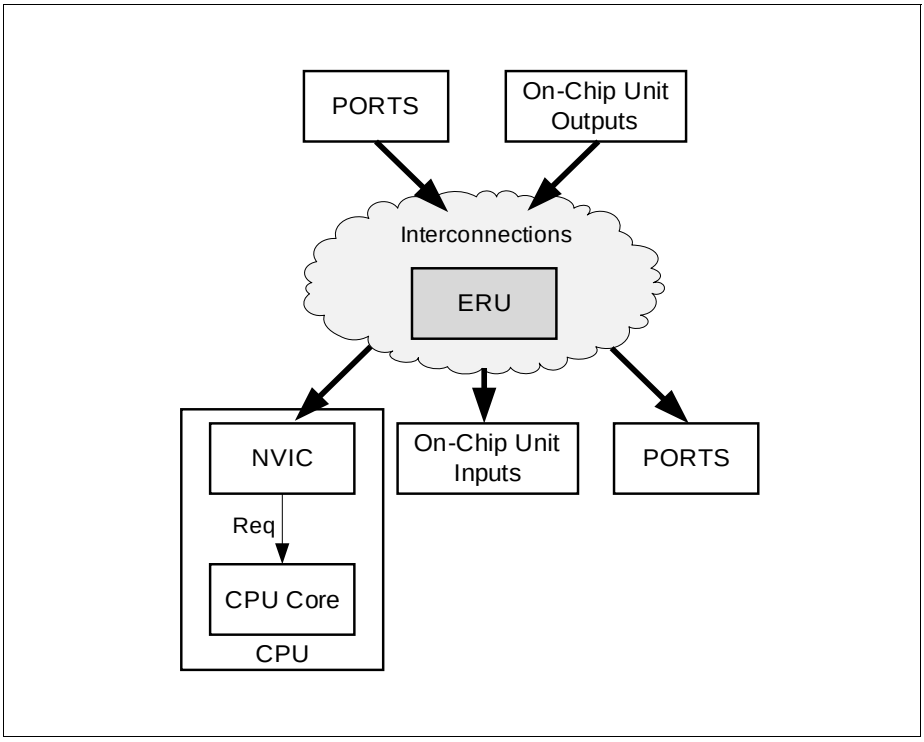


Figure 4-1 Block Diagram on Service Request Processing

4.2 Service Request Distribution

Figure 4-2 shows an example of how a service request can be distributed concurrently. To support the concurrent distribution to multiple receivers, the receiving modules are capable to enable/disable incoming requests.

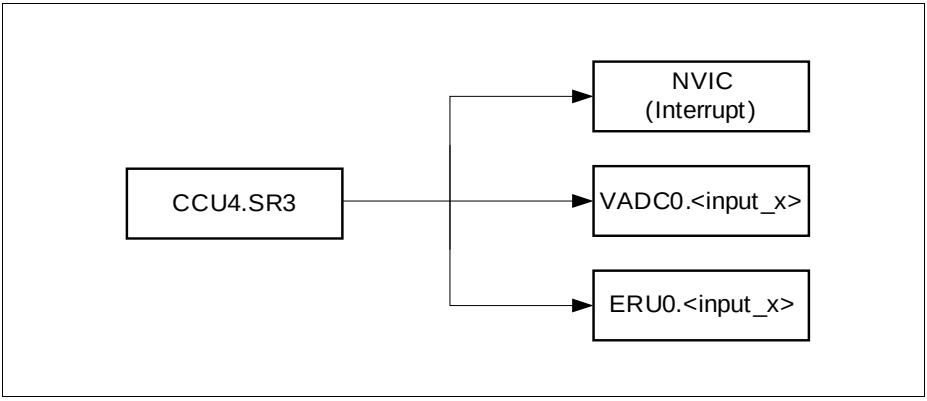


Figure 4-2 Example for Service Request Distribution

The units involved in Service Request distribution can be subdivided into

- Embedded real time services
- Interrupt services

Embedded real time services

Connectivity between On-Chip Units and PORTS is real time application and also chip package dependant. Related connectivity and availability of pins can be looked up in the

- “Interconnects” Section of the respective module(s) chapters
- “Parallel Ports” chapter and Data Sheet for PORTS
- “Event Request Unit” chapter

Interrupt services

The following table gives an overview on the number of service requests per module and how the service requests are assigned to NVIC Interrupt service provider.

Service Requests are always of of type “Pulse” in XMC1100.

Table 4-2 Interrupt services per module

Modules	Request Sources	NVIC	Type
VADC	4	2	Pulse
CCU40	4	4	Pulse
USIC0	6	6	Pulse
SCU	2	2	Pulse
ERU0	4	4	Pulse
Total	20	18	-

5 Interrupt Subsystem

The interrupt Subsystem in XMC1100 consists of the Nested Vectored Interrupt Controller (NVIC) and the respective modules' interrupt generation blocks.

Note: The CPU exception model is described in the CPU chapter.

5.1 Nested Vectored Interrupt Controller (NVIC)

The NVIC is an integral part of the Cortex M0 processor unit. Due to a tight coupling with the CPU, it provides the lowest interrupt latency and efficient processing of late arriving interrupts.

5.1.1 Features

The NVIC supports the following features:

- 32 interrupt nodes
- 4 programmable priority levels for each interrupt node
- Support for interrupt tail-chaining and late-arrival
- Software interrupt generation

5.1.2 Interrupt Node Assignment

Table 5-1 lists the service request sources per peripheral and their assignment to NVIC interrupt nodes. For calculation of the vector routine address, please refer to the section on Vector Table in the CPU chapter.

Table 5-1 Interrupt Node assignment

Service Request	Node ID	Description
SCU.SR0 - SCU.SR2	0...1	System Control SR0 is the system critical request SR1 is the common SCU request
NC	2	Reserved
ERU0.SR0 - ERU0.SR3	3...6	External Request Unit 0
NC	7...8	Reserved
USIC0.SR0 - USIC0.SR5	9...14	Universal Serial Interface Channel (Module 0)
VADC0.C0SR0 - VADC0.C0SR1	15...16	Analog to Digital Converter (Common)
NC	17...20	Reserved

Table 5-1 Interrupt Node assignment (cont'd)

Service Request	Node ID	Description
CCU40.SR0 - CCU40.SR3	21...24	Capture Compare Unit 4 (Module 0)
NC	25...31	Reserved

5.1.3 Interrupt Signal Generation

In XMC1100, all peripherals support only the generation of pulse interrupts. Pulse interrupts are also described as edge-triggered interrupts.

A pulse interrupt is an interrupt signal sampled synchronously on the rising edge of the processor clock (MCLK). To ensure the NVIC detects the interrupt, the peripheral asserts the interrupt signal for at least one MCLK clock cycle, during which the NVIC detects the pulse and latches the interrupt.

When the processor enters the ISR, it automatically removes the pending state from the interrupt, see **Hardware and software control of interrupts**.

The processor automatically stacks its state on exception entry and unstacks this state on exception exit, with no instruction overhead. This provides low latency exception handling.

Hardware and software control of interrupts

The Cortex-M0 latches all interrupts. A peripheral interrupt becomes pending for one of the following reasons:

- the NVIC detects that the interrupt signal is active and the interrupt is not active
- the NVIC detects a rising edge on the interrupt signal
- software writes to the corresponding interrupt set-pending register bit, see Interrupt Set-pending Register NVIC_ISPR.

A pending interrupt remains pending until one of the following:

- The processor enters the ISR for the interrupt. This changes the state of the interrupt from pending to active. Then:
  - The NVIC continues to monitor the interrupt signal, and If this is pulsed the state of the interrupt changes to pending and active. In this case, when the processor returns from the ISR the state of the interrupt changes to pending, which might cause the processor to immediately re-enter the ISR.
    If the interrupt signal is not pulsed while the processor is in the ISR, the state of the interrupt changes to inactive when the processor returns from the ISR.
- Software writes to the corresponding interrupt clear-pending register bit.
  - The state of the interrupt changes to inactive, if the state was pending; or active, if the state was active and pending.

5.1.4 NVIC design hints and tips

An interrupt node can enter pending state even if it is disabled. Disabling an interrupt node only prevents the processor from taking interrupts from that node.

NVIC programming hints

Software uses the CPSIE i and CPSID i instructions to enable and disable interrupts. The CMSIS provides the following intrinsic functions for these instructions:

```
void __disable_irq(void)  // Disable Interrupts
void __enable_irq(void)  // Enable Interrupts
```

In addition, the CMSIS provides a number of functions for NVIC control, including:

Table 5-2 CMSIS functions for NVIC control

CMSIS interrupt control function	Description
void NVIC_EnableIRQ (IRQn_t IRQn)	Enable IRQn
void NVIC_DisableIRQ (IRQn_t IRQn)	Disable IRQn
uint32_t NVIC_GetPendingIRQ (IRQn_t IRQn)	Return true (1) if IRQn is pending
void NVIC_SetPendingIRQ (IRQn_t IRQn)	Set IRQn pending
void NVIC_ClearPendingIRQ (IRQn_t IRQn)	Clear IRQn pending status
void NVIC_SetPriority (IRQn_t IRQn, uint32_t priority)	Set priority for IRQn
uint32_t NVIC_GetPriority (IRQn_t IRQn)	Read priority of IRQn
void NVIC_SystemReset (void)	Reset the system

The input parameter IRQn is the IRQ number. For more information about these functions, please refer to the CMSIS documentation.

5.1.5 Accessing CPU Registers using CMSIS

CMSIS functions enable software portability between different Cortex-M profile processors. To access the NVIC registers when using CMSIS, use the following functions:

Table 5-3 CMSIS access NVIC functions

CMSIS function	Description
void NVIC_EnableIRQ (IRQn_Type IRQn)[1)]	Enables an interrupt or exception.
void NVIC_DisableIRQ (IRQn_Type IRQn)[1)]	Disables an interrupt or exception.

Table 5-3 CMSIS access NVIC functions (cont'd)

CMSIS function	Description
void NVIC_SetPendingIRQ (IRQn_Type IRQn)[1)]	Sets the pending status of interrupt or exception to 1.
void NVIC_ClearPendingIRQ (IRQn_Type IRQn)[1)]	Clears the pending status of interrupt or exception to 0.
uint32_t NVIC_GetPendingIRQ (IRQn_Type IRQn)[1)]	Reads the pending status of interrupt or exception. This function returns non-zero value if the pending status is set to 1.
void NVIC_SetPriority(IRQn_Type IRQn, uint32_t priority)[1)]	Sets the priority of an interrupt or exception with configurable priority level to 1.
uint32_t NVIC_GetPriority(IRQn_Type IRQn)[1)]	Reads the priority of an interrupt or exception with configurable priority level. This function return the current priority level.

1) The input parameter IRQn is the IRQ number.

5.1.6 Interrupt Priority

An interrupt node can be assigned one of four priority levels. The levels are in steps of 64, from 0 to 192, and defined in an 8-bit priority field in the Interrupt Priority Register x (IPRx). A higher level corresponds to a lower priority, so level 0 is the highest interrupt priority.

Since there are four priority fields in each IPRx register and each field corresponds to one interrupt node, altogether 8 IPRx registers (IPR0...IPR7) are needed as shown in **Figure 5-1**.

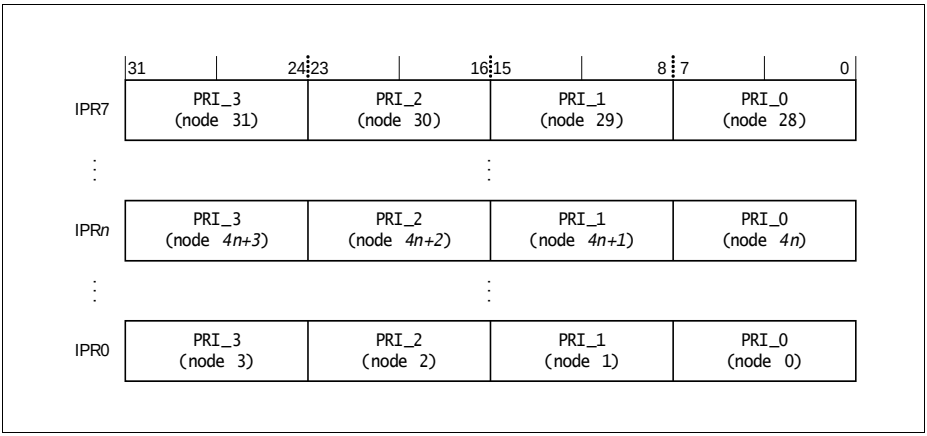


Figure 5-1 Interrupt Priority Register

The IPR number and byte offset for interrupt node m (0...31) can be found as follows:

- the corresponding IPR number n is given by
  n = m DIV 4
- the byte offset of the required Priority field in this register is m MOD 4, where:
  – byte offset 0 refers to register bits [7:0]
  – byte offset 1 refers to register bits [15:8]
  – byte offset 2 refers to register bits [23:16]
  – byte offset 3 refers to register bits [31:24]
- for example, Priority field of interrupt node 21 is located at IPR5.[15:8], since
  – n = 21 DIV 4 = 5
  – byte offset = 21 MOD 4 = 1

Note: IPRx registers are only word-accessible.

Refer to **Table 5-2** for more information on the access to the interrupt node priority array, which provides the software view of the interrupt node priorities.

5.1.7 Interrupt Response Time

The interrupt response time, defined as the time from detection of the generated pulse and latching of the interrupt by NVIC to execution of the first instruction at the interrupt handler, is typically 21 MCLK cycles as shown in **Figure 5-2**.

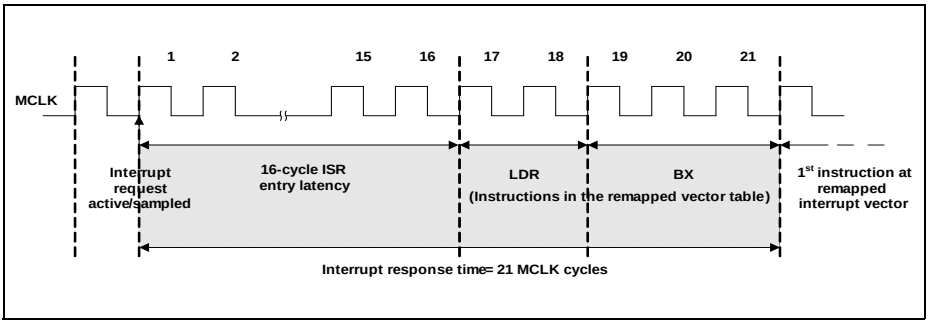


Figure 5-2 Typical Interrupt Response Time

This assumes the following conditions:

- Interrupt generation is enabled
- No occurrence of interrupt pre-emption, late-arrival or tail-chaining
- Delays due to memory wait states are not taken into account

5.2 General Module Interrupt Structure

A module might have multiple interrupt sources. Each interrupt source has typically the following structure (see **Figure 5-3**):

- An interrupt source status flag
- A set bit to allow software to set the flag to 1
- A clear bit to allow software to reset the flag to 0
- An enable bit to trigger interrupt when the hardware event occurs or status flag set bit is set (i.e. software triggered interrupt)

Note: If a flag set event (due to a peripheral HW event) occurs in the same clock cycle as a flag clear event (due to SW Setting of the Clear bit), the set has higher priority over the clear.

Note: Setting of the status flag by a hardware event or software writing to status flag set bit, is independent of interrupt generation enabled/disabled. Similarly, interrupt generation is independent of the level of the status flag.

Additionally, some modules might have more interrupt sources than interrupt lines. Therefore, they include an interrupt routing management block, which maps the interrupt sources to the interrupt lines.

For further details and exceptions to the above general structure, refer to the respective module chapters. An overview of all XMC1100 interrupt sources is given at the end of the chapter.

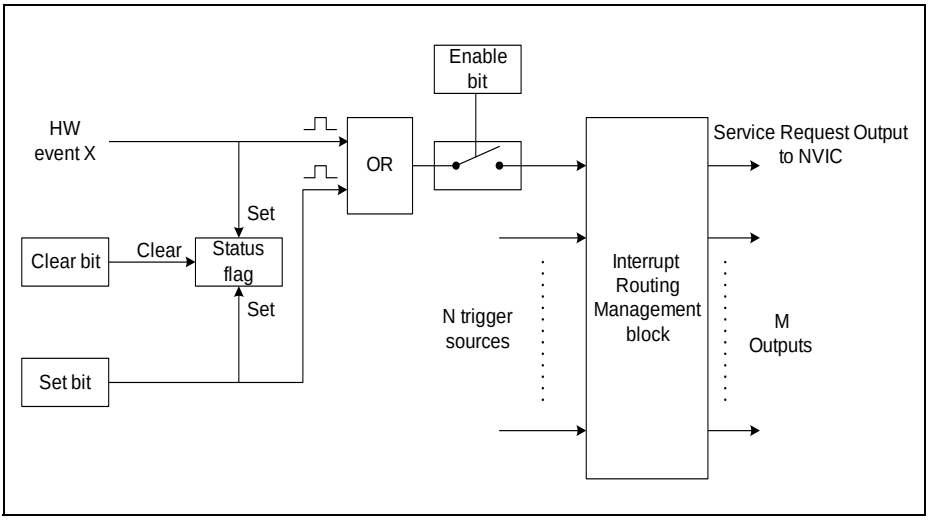


Figure 5-3 Typical Module Interrupt Structure

To enable a module HW event for interrupt generation, SW has to:

- Enable the interrupt node that is allocated to the module in the NVIC, through the NVIC_ISER register.
- If the module has an interrupt routing management block, select an available service request output through which the interrupt will be generated to the NVIC. This is usually done by configuring a interrupt node pointer register in the module.
- Finally, set the interrupt enable bit of the module HW event for interrupt generation.

5.3 Registers

Table 5-4 Registers Address Space

Module	Base Address	End Address	Note
CPU PPB: System Control Space (SCS)	$E000E000_H$	$E000EFFF_H$	

Registers Overview

The absolute register address is calculated by adding:

Module Base Address + Offset Address

Table 5-5 Register Overview

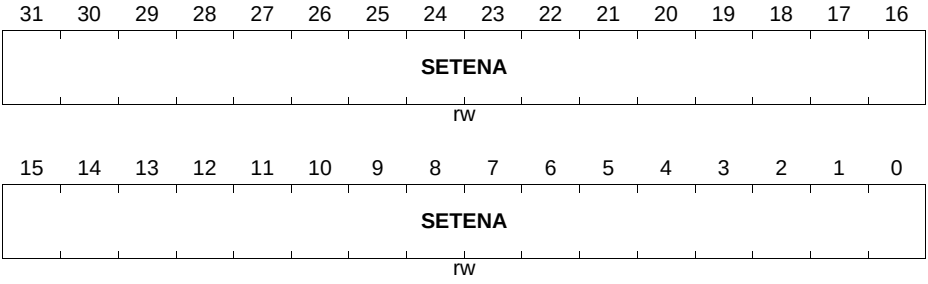
Short Name	Description	Offset Address	Access Mode		Description See
			Read	Write	
Nested Vectored Interrupt Controller (NVIC)					
NVIC_ISER	Interrupt Set-enable Registers	100_H	U, PV	U, PV	Page 5-9
NVIC_ICER	Interrupt Clear-enable Registers	180_H	U, PV	U, PV	Page 5-10
NVIC_ISPR	Interrupt Set-pending Registers	200_H	U, PV	U, PV	Page 5-11
NVIC_ICPR	Interrupt Clear-pending Registers	280_H	U, PV	U, PV	Page 5-12
NVIC_IPR0 - NVIC_IPR7	Interrupt Priority Registers	400_H-$41C_H$	U, PV	U, PV	Page 5-13

5.3.1 NVIC Registers

NVIC_ISER

The ISER register enables interrupt nodes, and shows which interrupt nodes are enabled.

NVIC_ISER
Interrupt Set-enable Register (E000E100$_H$) Reset Value: 00000000$_H$



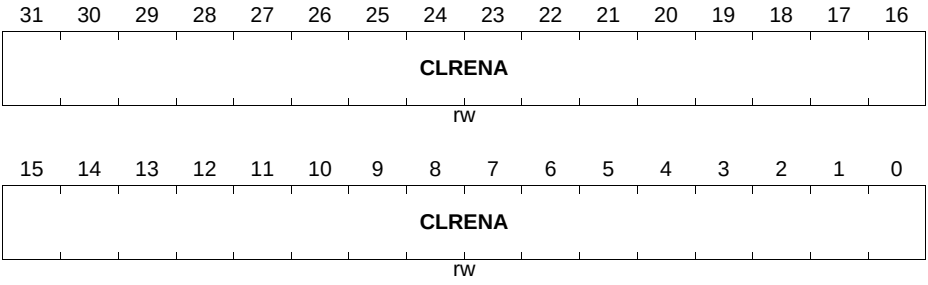
Field	Bits	Type	Description
SETENA	[31:0]	rw	**Interrupt Node Set-enable** 0_B Read: Interrupt node disabled. Write: No effect. 1_B Read: Interrupt node enabled. Write: Enable interrupt node

If a pending interrupt is enabled, the NVIC activates the interrupt based on its priority. If an interrupt is not enabled, asserting its interrupt signal changes the interrupt state to pending, but the NVIC never activates the interrupt, regardless of its priority.

NVIC_ICER

The ICER register disables interrupt nodes, and shows which interrupt nodes are enabled.

NVIC_ICER
IInterrupt Clear-enable Register (E000E180$_H$) **Reset Value: 00000000$_H$**



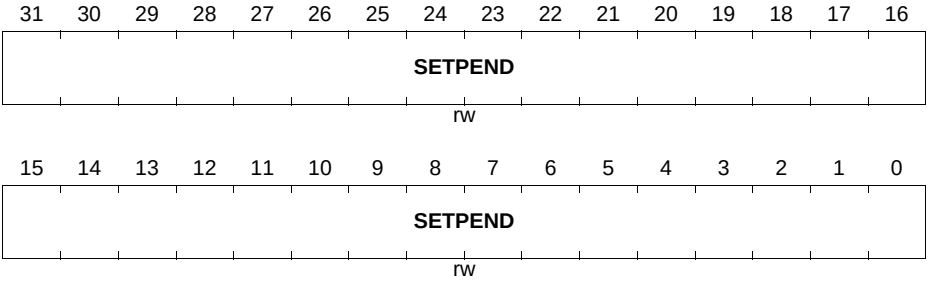
Field	Bits	Type	Description
CLRENA	[31:0]	rw	**Interrupt Node Clear-enable** 0_B Read: Interrupt node disabled. Write: No effect 1_B Read: Interrupt node enabled. Write: Disable interrupt node.

NVIC_ISPR

The ISPR register forces interrupt nodes into the pending state, and shows which interrupt nodes are pending.

NVIC_ISPR
Interrupt Set-pending Register

(E000E200$_H$) **Reset Value: 00000000$_H$**



Field	Bits	Type	Description
SETPEND	[31:0]	rw	**Interrupt Node Set-pending** 0_B Read: Interrupt node is not pending. Write: No effect 1_B Read: Interrupt node is pending. Write: Change interrupt state to pending.

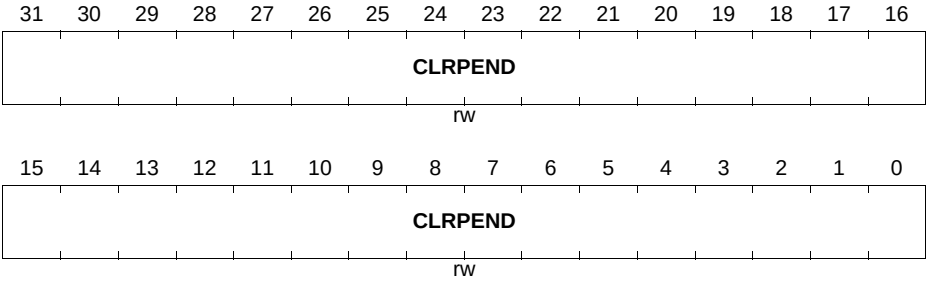
Note: Writing 1 to the ISPR bit corresponding to:
- an interrupt node that is pending has no effect
- a disabled interrupt node sets the state of that interrupt node to pending

NVIC_ICPR

The ICPR register removes the pending state from interrupt nodes, and shows which interrupt nodes are pending.

NVIC_ICPR
Interrupt Clear-pending Register

(E000E280$_H$) **Reset Value: 00000000$_H$**



Field	Bits	Type	Description
CLRPEND	[31:0]	rw	**Interrupt Node Clear-pending** 0_B Read: Interrupt node is not pending. Write: No effect. 1_B Read: Interrupt node is pending. Write: Remove interrupt state from pending.

Note: Writing 1 to an ICPR bit does not affect the active state of the corresponding interrupt node.

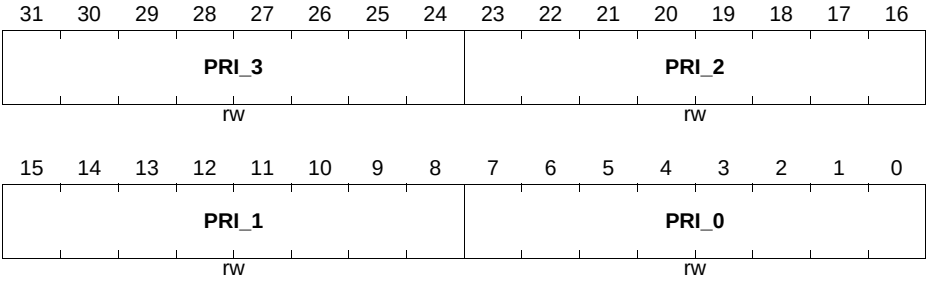
NVIC_IPRx (x=0-7)

The IPR0-IPR7 registers provide a 8-bit priority field for each interrupt node. Each register holds four priority fields.

Each priority field holds a priority value, 0-192. The lower the value, the greater the priority of the corresponding interrupt node. The processor implements only bits [7:6] of each field, bits [5:0] reads as 0 and ignores writes. This means writing 255 to a priority register saves value 192 to the register.

NVIC_IPRx (x=0-7)
Interrupt Priority Register x

($E000E400_H$ + 4*x) **Reset Value: 00000000_H**



Field	Bits	Type	Description
PRI_3	[31:24]	rw	**Priority, Byte Offset 3**
PRI_2	[23:16]	rw	**Priority, Byte Offset 2**
PRI_1	[15:8]	rw	**Priority, Byte Offset 1**
PRI_0	[7:0]	rw	**Priority, Byte Offset 0**

5.4 Interrupt Request Source Overview

An overview of all XMC1100 interrupt sources and related register bits are shown in the next few pages.

Table 5-6 Interrupt Source Overview

IRQ	Interrupt Node	Interrupt Source	Status Flag		Interrupt Enable		Set Flag		Clear Flag		Node Pointer	
			Register	Bit	Register	Bit	Register	Bit	Register	Bit	Register	Bit
0	SCU.SR0	Flash double bit ECC[1)]	SCU_ SRRAW	FLECC2I	SCU_ SRMSK	FLECC2I	SCU_ SRSET	FLECC2I	SCU_ SRCLR	FLECC2I	-	-
			NVM_NVM STATUS	ECC2READ	-	-	-	-	NVM_NVM PROG	RSTECC	-	-
		Flash operation complete	SCU_ SRRAW	FLCMPLTI	NVM_ NVMCONF	INT_ON	SCU_ SRSET	FLCMPLTI	SCU_ SRCLR	FLCMPLTI	-	-
		SRAM parity error	SCU_ SRRAW	PESRAMI	SCU_ SRMSK	PESRAMI	SCU_ SRSET	PESRAMI	SCU_ SRCLR	PESRAMI	-	-
		USIC RAM parity error	SCU_ SRRAW	PEU0I	SCU_ SRMSK	PEU0I	SCU_ SRSET	PEU0I	SCU_ SRCLR	PEU0I	-	-
		Loss of clock	SCU_ SRRAW	LOCI	SCU_ SRMSK	LOCI	SCU_ SRSET	LOCI	SCU_ SRCLR	LOCI	-	-

Table 5-6 Interrupt Source Overview

IRQ	Interrupt Node	Interrupt Source	Status Flag		Interrupt Enable		Set Flag		Clear Flag		Node Pointer	
			Register	Bit	Register	Bit	Register	Bit	Register	Bit	Register	Bit
1	SCU.SR1	Standby clock failure	SCU_ SRRAW	SBYCLKFI	SCU_ SRMSK	SBYCLKFI	SCU_ SRSET	SBYCLKFI	SCU_ SRCLR	SBYCLKFI	-	-
		VDDP pre-warning	SCU_ SRRAW	VDDPI	SCU_ SRMSK	VDDPI	SCU_ SRSET	VDDPI	SCU_ SRCLR	VDDPI	-	-
		VDDC drops below VDROP	SCU_ SRRAW	VDROPI	SCU_ SRMSK	VDROPI	SCU_ SRSET	VDROPI	SCU_ SRCLR	VDROPI	-	-
		VDDC rises above VCLIP	SCU_ SRRAW	VCLIPI	SCU_ SRMSK	VCLIPI	SCU_ SRSET	VCLIPI	SCU_ SRCLR	VCLIPI	-	-
		WDT pre-warning	SCU_ SRRAW	PRWARN	SCU_ SRMSK	PRWARN	SCU_ SRSET	PRWARN	SCU_ SRCLR	PRWARN	-	-
		RTC periodic event	SCU_ SRRAW	PI	-	-	SCU_ SRSET	PI	SCU_ SRCLR	PI	-	-
		RTC alarm	SCU_ SRRAW	AI	-	-	SCU_ SRSET	AI	SCU_ SRCLR	AI	-	-
		RTC CTR Mirror Register updated	SCU_ SRRAW	RTC_CTR	SCU_ SRMSK	RTC_CTR	SCU_ SRSET	RTC_CTR	SCU_ SRCLR	RTC_CTR	-	-
		RTC ATIM0 Mirror Register updated	SCU_ SRRAW	RTC_ ATIM0	SCU_ SRMSK	RTC_ ATIM0	SCU_ SRSET	RTC_ ATIM0	SCU_ SRCLR	RTC_ ATIM0	-	-
		RTC ATIM1 Mirror Register updated	SCU_ SRRAW	RTC_ ATIM1	SCU_ SRMSK	RTC_ ATIM1	SCU_ SRSET	RTC_ ATIM1	SCU_ SRCLR	RTC_ ATIM1	-	-
		RTC TIM0 Mirror Register updated	SCU_ SRRAW	RTC_TIM0	SCU_ SRMSK	RTC_TIM0	SCU_ SRSET	RTC_TIM0	SCU_ SRCLR	RTC_TIM0	-	-
		RTC TIM1 Mirror Register updated	SCU_ SRRAW	RTC_TIM1	SCU_ SRMSK	RTC_TIM1	SCU_ SRSET	RTC_TIM1	SCU_ SRCLR	RTC_TIM1	-	-
2	Reserved											
3, 4, 5, 6	ERU0. SR[3:0]	ERU0_IOUTx (x=0-3)	See section on ERU0 for details.									
7, 8	Reserved											

Table 5-6 Interrupt Source Overview

IRQ	Interrupt Node	Interrupt Source	Status Flag		Interrupt Enable		Set Flag		Clear Flag		Node Pointer	
			Register	Bit	Register	Bit	Register	Bit	Register	Bit	Register	Bit
9, 10, 11, 12, 13, 14	USIC0_SR[5:0]	USIC: Standard receive event	USIC0_PSR	RIF	USIC0_CCR	RIEN	-	-	USIC0_PSCR	CRIF	USIC0_INPR	RINP
		USIC: Receive start event	USIC0_PSR	RSIF	USIC0_CCR	RSIEN	-	-	USIC0_PSCR	CRSIF	USIC0_INPR	TBINP
		USIC: Alternate receive event	USIC0_PSR	AIF	USIC0_CCR	AIEN	-	-	USIC0_PSCR	CAIF	USIC0_INPR	AINP
		USIC: Transmit shift event	USIC0_PSR	TSIF	USIC0_CCR	TSIEN	-	-	USIC0_PSCR	CTSIF	USIC0_INPR	TSINP
		USIC: Transmit buffer event	USIC0_PSR	TBIF	USIC0_CCR	TBIEN	-	-	USIC0_PSCR	CTBIF	USIC0_INPR	TBINP
		USIC: Data lost event	USIC0_PSR	DLIF	USIC0_CCR	DLIEN	-	-	USIC0_PSCR	CDLIF	USIC0_INPR	PINP
		USIC: BRG event	USIC0_PSR	BRGIF	USIC0_CCR	BRGIEN	-	-	USIC0_PSCR	CBRGIF	USIC0_INPR	PINP

Table 5-6 Interrupt Source Overview

IRQ	Interrupt Node	Interrupt Source	Status Flag		Interrupt Enable		Set Flag		Clear Flag		Node Pointer	
			Register	Bit	Register	Bit	Register	Bit	Register	Bit	Register	Bit
9, 10, 11, 12, 13, 14	USIC0_SR[5:0]	USIC: Standard transmit buffer event	USIC0_TRBSR	STBI	USIC0_TBCTR	STBIEN	-	-	USIC0_TRBSCR	CSTBI	USIC0_TBCTR	STBINP
		USIC: Standard transmit buffer event	USIC0_TRBSR	STBT	USIC0_TBCTR	STBIEN	-	-	-	-	USIC0_TBCTR	STBINP
		USIC: Transmit Buffer error event	USIC0_TRBSR	TBERI	USIC0_TBCTR	TBERIEN	-	-	USIC0_TRBSCR	CTBERI	USIC0_TBCTR	ATBINP
		USIC: Standard receive buffer event	USIC0_TRBSR	SRBI	USIC0_RBCTR	SRBIEN	-	-	USIC0_TRBSCR	CSRBI	USIC0_RBCTR	SRBINP
		USIC: Standard receive buffer event	USIC0_TRBSR	SRBT	USIC0_RBCTR	SRBIEN	-	-	-	-	USIC0_RBCTR	SRBINP
		USIC: Alternate receive buffer event	USIC0_TRBSR	ARBI	USIC0_RBCTR	ARBIEN	-	-	USIC0_TRBSCR	CARBI	USIC0_RBCTR	ARBINP
		USIC: Receive buffer error event	USIC0_TRBSR	RBERI	USIC0_RBCTR	RBERIEN	-	-	USIC0_TRBSCR	CRBERI	USIC0_RBCTR	ARBINP
		ASC: Synchronisation break detected	USIC0_PSR	SBD	USIC0_PCR	SBDIEN	-	-	USIC0_PSCR	CSBD	USIC0_INPR	PINP
		ASC: Collision detected	USIC0_PSR	COL	USIC0_PCR	CDIEN	-	-	USIC0_PSCR	CCOL	USIC0_INPR	PINP
		ASC: Receiver noise detected	USIC0_PSR	RNS	USIC0_PCR	RNIEN	-	-	USIC0_PSCR	CRNS	USIC0_INPR	PINP
		ASC: Format error in stop bit 0	USIC0_PSR	FER0	USIC0_PCR	FEIEN	-	-	USIC0_PSCR	CFER0	USIC0_INPR	PINP
		ASC: Format error in stop bit 1	USIC0_PSR	FER1	USIC0_PCR	FEIEN	-	-	USIC0_PSCR	CFER1	USIC0_INPR	PINP
		ASC: Receive frame finished	USIC0_PSR	RFF	USIC0_PCR	FFIEN	-	-	USIC0_PSCR	CRFF	USIC0_INPR	PINP
		ASC: Transmit frame finished	USIC0_PSR	TFF	USIC0_PCR	FFIEN	-	-	USIC0_PSCR	CTFF	USIC0_INPR	PINP

Table 5-6 Interrupt Source Overview

IRQ	Interrupt Node	Interrupt Source	Status Flag		Interrupt Enable		Set Flag		Clear Flag		Node Pointer	
			Register	Bit	Register	Bit	Register	Bit	Register	Bit	Register	Bit
9, 10, 11, 12, 13, 14	USIC0_SR[5:0]	SSC: MSLS event detected	USIC0_PSR	MSLSEV	USIC0_PCR	MSLSIEN	-	-	USIC0_PSCR	CMSLSEV	USIC0_INPR	PINP
		SSC: Parity error detected	USIC0_PSR	PAERR	USIC0_PCR	PARIEN	-	-	USIC0_PSCR	CPAERR	USIC0_INPR	PINP
		SSC: DX2T event detected	USIC0_PSR	DX2TEV	USIC0_PCR	DX2TIEN	-	-	USIC0_PSCR	CDX2TEV	USIC0_INPR	PINP
		IIC: Wrong TDF code detected	USIC0_PSR	WTDF	USIC0_PCR	ERRIEN	-	-	USIC0_PSCR	CWTDF	USIC0_INPR	PINP
		IIC: Start condition received	USIC0_PSR	SCR	USIC0_PCR	SCRIEN	-	-	USIC0_PSCR	CSCR	USIC0_INPR	PINP
		IIC: Repeated start condition received	USIC0_PSR	RSCR	USIC0_PCR	RSCRIEN	-	-	USIC0_PSCR	CRSCR	USIC0_INPR	PINP
		IIC: Stop condition received	USIC0_PSR	PCR	USIC0_PCR	PCRIEN	-	-	USIC0_PSCR	CPCR	USIC0_INPR	PINP
		IIC: NACK received	USIC0_PSR	NACK	USIC0_PCR	NACKIEN	-	-	USIC0_PSCR	CNACK	USIC0_INPR	PINP
		IIC: Arbitration lost	USIC0_PSR	ARL	USIC0_PCR	ARLIEN	-	-	USIC0_PSCR	CARL	USIC0_INPR	PINP
		IIC: Slave read request	USIC0_PSR	SRR	USIC0_PCR	SRRIEN	-	-	USIC0_PSCR	CSRR	USIC0_INPR	PINP
		IIC: Error detected	USIC0_PSR	ERR	USIC0_PCR	ERRIEN	-	-	USIC0_PSCR	CERR	USIC0_INPR	PINP
		IIC: ACK received	USIC0_PSR	ACK	USIC0_PCR	ACKIEN	-	-	USIC0_PSCR	CACK	USIC0_INPR	PINP
		IIS: DX2T event detected	USIC0_PSR	DX2TEV	USIC0_PCR	DX2TIEN	-	-	USIC0_PSCR	CDX2TEV	USIC0_INPR	PINP
		IIS: WA falling edge event	USIC0_PSR	WAFE	USIC0_PCR	WAFEIEN	-	-	USIC0_PSCR	CWAFE	USIC0_INPR	PINP
		IIS: WA rising edge event	USIC0_PSR	WARE	USIC0_PCR	WAREIEN	-	-	USIC0_PSCR	CWARE	USIC0_INPR	PINP
		IIS: WA generation end	USIC0_PSR	END	USIC0_PCR	ENDIEN	-	-	USIC0_PSCR	CEND	USIC0_INPR	PINP

Table 5-6 Interrupt Source Overview

IRQ	Interrupt Node	Interrupt Source	Status Flag		Interrupt Enable		Set Flag		Clear Flag		Node Pointer	
			Register	Bit	Register	Bit	Register	Bit	Register	Bit	Register	Bit
15, 16,	VADC0_C0SR[1:0]	Source Event 0	VADC0_GxSEFLAG	SEV0	VADC0_GxQINR0	ENSI	VADC0_GxSEFLAG	SEV0	VADC0_GxSEFCLR	SEV0	VADC0_GxSEVNP	SEV0NP
		Source Event 1	VADC0_GxSEFLAG	SEV1	VADC0_GxASMR	ENSI	VADC0_GxSEFLAG	SEV1	VADC0_GxSEFCLR	SEV1	VADC0_GxSEVNP	SEV1NP
		Channel Event y (y=0-7)	VADC0_GxCEFLAG	CEVy	VADC0_GxCHCTRy	CHEVMODE	VADC0_GxCEFLAG	CEVy	VADC0_GxCEFCLR	CEVy	VADC0_GxCEVNP	CEVyINP
		Result Event y (y=0-7)	VADC0_GxREFLAG	REVy	VADC0_GxRCRy	SRGEN	VADC0_GxREFLAG	REVy	VADC0_GxREFCLR	REVy	VADC0_GxREVNP0	REVyNP
		Result Event y (y=8-15)	VADC0_GxREFLAG	REVy	VADC0_GxRCRy	SRGEN	VADC0_GxREFLAG	REVy	VADC0_GxREFCLR	REVy	VADC0_GxREVNP1	REVyNP
		Global Source Event	VADC0_GLOBEFLAG	SEVGLB	VADC0_BRSMR	ENSI	VADC0_GLOBEFLAG	SEVGLB	VADC0_GLOBEFLAG	SEVGLBCLR	VADC0_GLOBEVNP	REV0NP
		Global Result Event	VADC0_GLOBEFLAG	REVGLB	VADC0_GLOBERCR	SRGEN	VADC0_GLOBEFLAG	REVGLB	VADC0_GLOBEFLAG	REVGLBCLR	VADC0_GLOBEVNP	SEV0NP
17, 18, 19, 20	Reserved											

Table 5-6 Interrupt Source Overview

IRQ	Interrupt Node	Interrupt Source	Status Flag		Interrupt Enable		Set Flag		Clear Flag		Node Pointer	
			Register	Bit	Register	Bit	Register	Bit	Register	Bit	Register	Bit
21, 22, 23, 24	CCU40_ SR[3:0]	Event 0 edge(s) information from event selector	CCU40_ CC4yINTS	E0AS	CCU40_ CC4yINTE	E0AE	CCU40_ CC4ySWS	SE0A	CCU40_ CC4ySWR	RE0A	CCU40_ CC4ySRS	E0SR
		Event 1 edge(s) information from event selector	CCU40_ CC4yINTS	E1AS	CCU40_ CC4yINTE	E1AE	CCU40_ CC4ySWS	SE1A	CCU40_ CC4ySWR	RE1A	CCU40_ CC4ySRS	E1SR
		Event 2 edge(s) information from event selector	CCU40_ CC4yINTS	E2AS	CCU40_ CC4yINTE	E2AE	CCU40_ CC4ySWS	SE2A	CCU40_ CC4ySWR	RE2A	CCU40_ CC4ySRS	E2SR
		Period Match while counting up	CCU40_ CC4yINTS	PMUS	CCU40_ CC4yINTE	PME	CCU40_ CC4ySWS	SPM	CCU40_ CC4ySWR	RPM	CCU40_ CC4ySRS	POSR
		Compare Match while counting up	CCU40_ CC4yINTS	CMUS	CCU40_ CC4yINTE	CMUE	CCU40_ CC4ySWS	SCMU	CCU40_ CC4ySWR	RCMU	CCU40_ CC4ySRS	CMSR
		Compare Match while counting down	CCU40_ CC4yINTS	CMDS	CCU40_ CC4yINTE	CMDE	CCU40_ CC4ySWS	SCMD	CCU40_ CC4ySWR	RCMD	CCU40_ CC4ySRS	CMSR
		One Match while counting down	CCU40_ CC4yINTS	OMDS	CCU40_ CC4yINTE	OME	CCU40_ CC4ySWS	SOM	CCU40_ CC4ySWR	ROM	CCU40_ CC4ySRS	POSR
		Entering Trap State	CCU40_ CC4yINTS	TRPF	CCU40_ CC4yINTE	E2AE	CCU40_ CC4ySWS	STRPF	CCU40_ CC4ySWR	RTRPF	CCU40_ CC4ySRS	E2SR
25, 26, 27, 28, 29, 30, 31	Reserved											

1) Flash ECC double bit error has two status flags, each having its own clear bit. It is sufficient to use only one of the status flag and ignore the other.

6 Event Request Unit (ERU)

As described in the Service Request Processing chapter, XMC1100 uses the Event Request Unit (ERU) to support the programmable interconnection for the processing of service requests.

6.1 Features

The ERU supports these features:

- Flexible processing of external and internal service requests
- Programmable for edge and/or level triggering
- Multiple inputs per channel
- Triggers combinable from multiple inputs
- Input and output gating

6.2 Overview

The Event Request Unit (ERU) is a versatile multiple input event detection and processing unit.

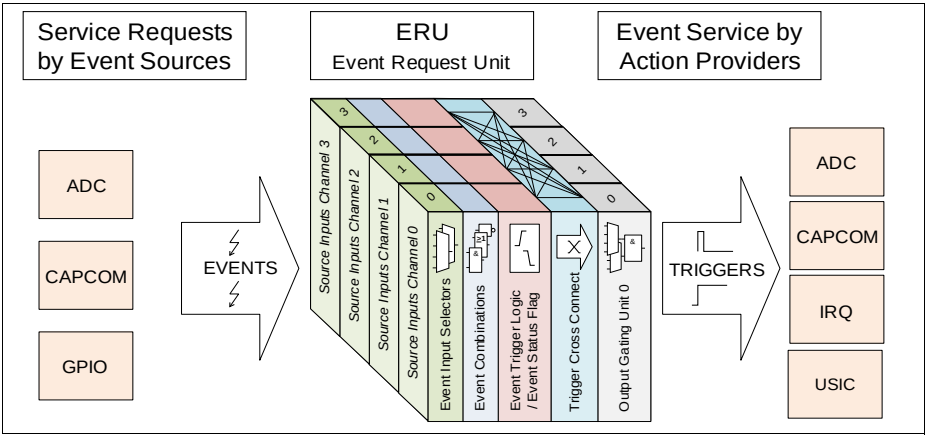


Figure 6-1 Event Request Unit Overview

Each ERU unit consists of the following blocks:

- An **Event Request Select (ERS)** unit.
  - Event Input Selectors allow the selection of one out of two inputs. For each of these two inputs, an vector of 4 possible signals is available.
  - Event Combinations allow a logical combination of two input signals to a common trigger.

- An **Event Trigger Logic (ETL)** per Input Channel allows the definition of the transition (edge selection, or by software) that lead to a trigger event and can also store this status. Here, the input levels of the selected signals are translated into events.
- The Trigger **Cross Connect Matrix** distributes the events and status flags to the Output Channels. Additionally, trigger signals from other modules are made available and can be combined with the local triggers.
- An **Output Gating Unit (OGU)** combines the trigger events and status information and gates the Output depending on a gating signal.

Note: An event of one Input can lead to reactions on several Outputs, or also events on several Inputs can be combined to a reaction on one Output.

6.3 Event Request Select Unit (ERS)

For each Input Channel x (x = 0-3), an ERSx unit handles the input selection for the associated ETLx unit. Each ERSx performs a logical combination of two signals (Ax, Bx) to provide one combined output signal ERSxO to the associated ETLx. Input Ax can be selected from 4 options of the input vector ERU_xA[3:0] and can be optionally inverted. A similar structure exists for input Bx (selection from ERU_xB[3:0]).

In addition to the direct choice of either input Ax or Bx or their inverted values, the possible logical combinations for two selected inputs are a logical AND or a logical OR.

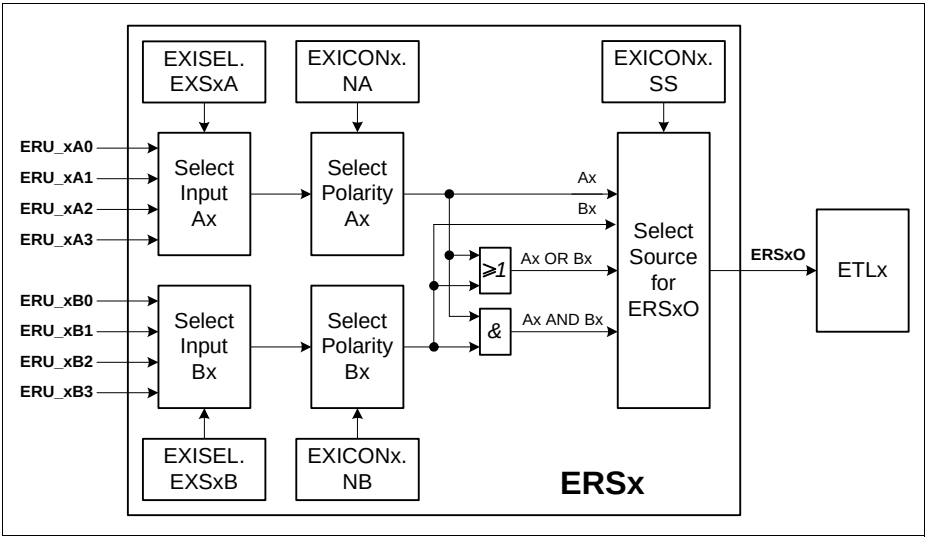


Figure 6-2 Event Request Select Unit Overview

The ERS units are controlled via register **ERU0_EXISEL** (one register for all four ERSx units) and registers EXICONx (one register for each ERSx and associated ETLx unit, e.g. **ERU0_EXICONx (x=0-3)** for Input Channel 0).

6.4 Event Trigger Logic (ETLx)

For each Input Channel x (x = 0-3), an event trigger logic ETLx derives a trigger event and related status information from the input ERSxO. Each ETLx is based on an edge detection block, where the detection of a rising or a falling edge can be individually enabled. Both edges lead to a trigger event if both enable bits are set (e.g. to handle a toggling input).

Each of the four ETLx units has an associated EXICONx register, that controls all options of an ETLx (the register also holds control bits for the associated ERSx unit, e.g. **ERU0_EXICONx (x=0-3)** to control ERS0 and ETL0).

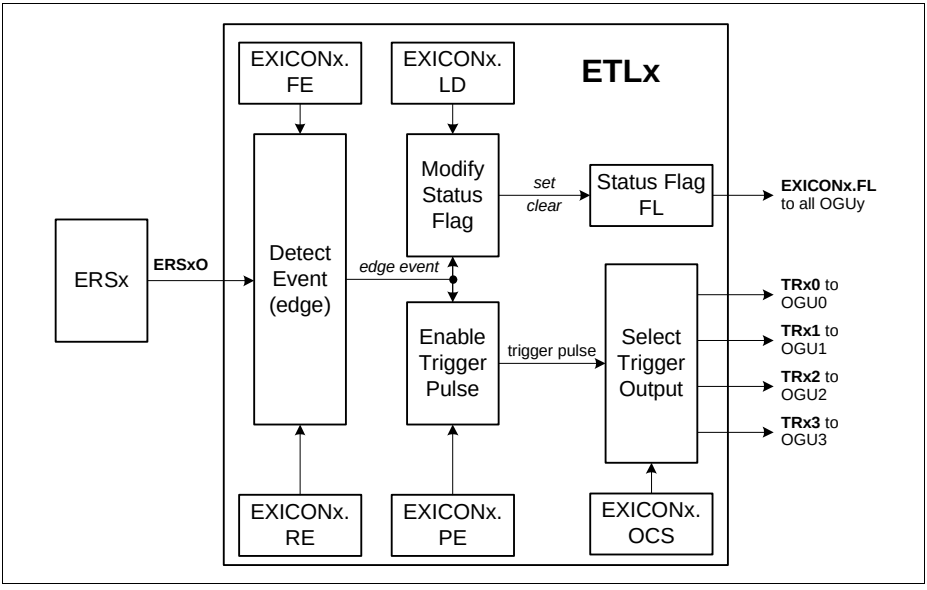


Figure 6-3 Event Trigger Logic Overview

When the selected event (edge) is detected, the status flag EXICONx.FL becomes set. This flag can also be modified by software. Two different operating modes are supported by this status flag.

It can be used as “sticky” flag, which is set by hardware when the desired event has been detected and has to be cleared by software. In this operating mode, it indicates that the event has taken place, but without indicating the actual status of the input.

In the second operating mode, it is cleared automatically if the “opposite” event is detected. For example, if only the falling edge detection is enabled to set the status flag, it is cleared when the rising edge is detected. In this mode, it can be used for pattern detection where the actual status of the input is important (enabling both edge detections is not useful in this mode).

The output of the status flag is connected to all following Output Gating Units (OGUy) in parallel (see **Figure 6-4**) to provide **pattern detection capability of all OGUy** units based on different or the same status flags.

In addition to the modification of the status flag, a trigger pulse output TRxy of ETLx can be enabled (by bit EXICONx.PE) and selected to **trigger actions in one of the OGUy** units. The target OGUy for the trigger is selected by bit field EXICON.OCS.

The trigger becomes active when the selected edge event is detected, independently from the status flag EXICONx.FL.

6.5 Cross Connect Matrix

The matrix shown in **Figure 6-4** distributes the trigger signals (TRxy) and status signals (EXICONx.FL) from the different ETLx units between the OGUy units. In addition, it receives peripheral trigger signals that can be OR-combined with the ETLx trigger signals in the OGUy units.

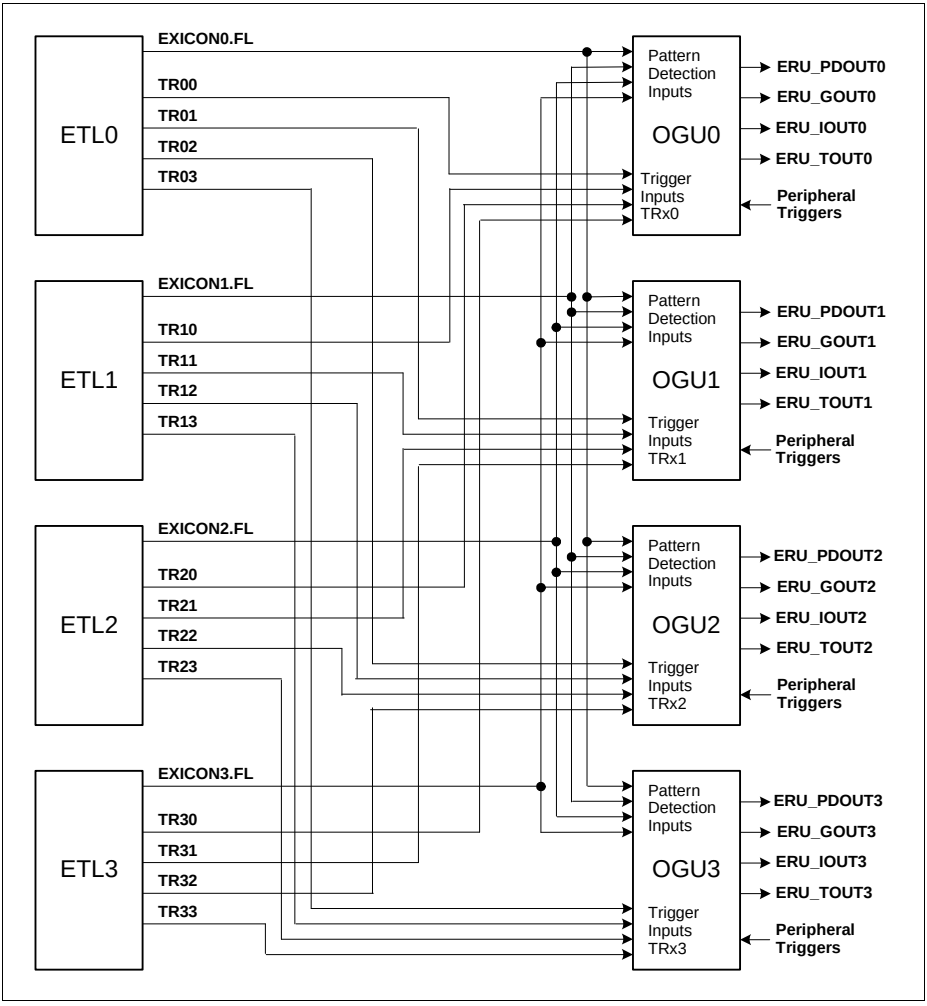


Figure 6-4 ERU Cross Connect Matrix

6.6 Output Gating Unit (OGUy)

Each OGUy (y = 0-3) unit combines the available trigger events and status flags from the Input Channels and distributes the results to the system. **Figure 6-5** illustrates the logic blocks within an OGUy unit. All functions of an OGUy unit are controlled by its associated

EXOCONy register, e.g. **ERU0_EXOCONx (x=0-3)** for OGU0. The function of an OGUy unit can be split into two parts:

- **Trigger Combination**:
  All trigger signals TRxy from the Input Channels that are enabled and directed to OGUy, a selected peripheral-related trigger event, and a pattern change event (if enabled) are logically OR-combined.
- **Pattern Detection**:
  The status flags EXICONx.FL of the Input Channels can be enabled to take part in the pattern detection. A pattern match is detected while all enabled status flags are set.

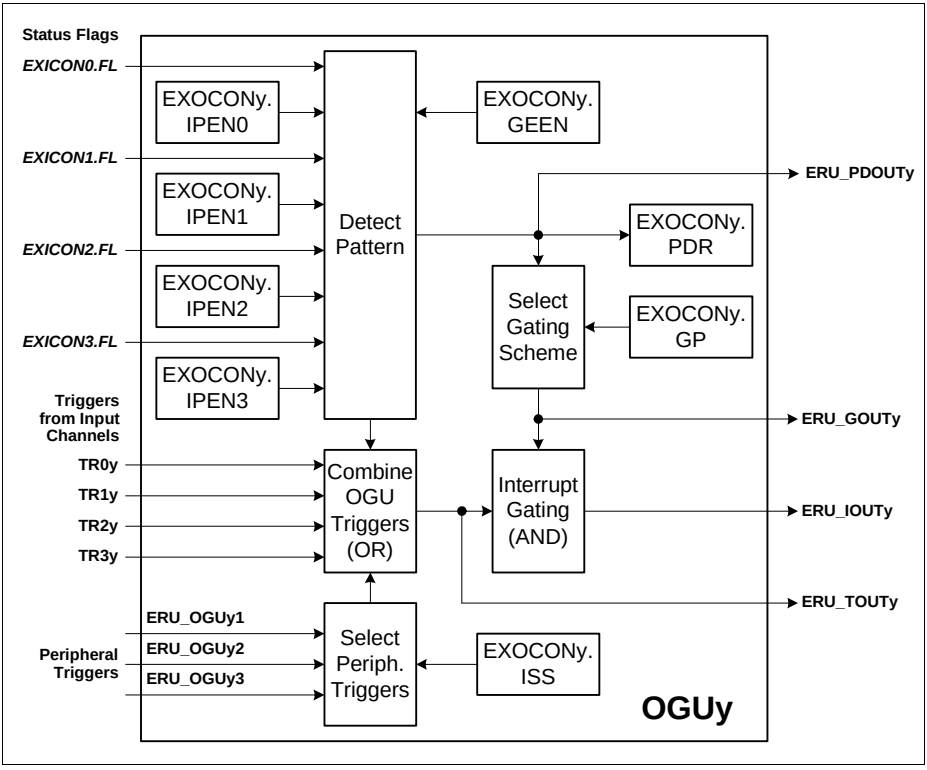


Figure 6-5 Output Gating Unit for Output Channel y

Each OGUy unit generates 4 output signals that are distributed to the system (not all of them are necessarily used):

- **ERU_PDOUTy** to directly output the pattern match information for gating purposes in other modules (pattern match = 1).

- **ERU_GOUTy** to output the pattern match or pattern miss information (inverted pattern match), or a permanent 0 or 1 under software control for gating purposes in other modules.
- **ERU_TOUTy** as combination of a peripheral trigger, a pattern detection result change event, or the ETLx trigger outputs TRxy to trigger actions in other modules.
- **ERU_IOUTy** as gated trigger output (ERU_GOUTy logical AND-combined with ERU_TOUTy) to trigger service requests (e.g. the service request generation can be gated to allow service request activation during a certain time window).

Trigger Combination

The trigger combination logically OR-combines different trigger inputs to form a common trigger ERU_TOUTy. Possible trigger inputs are:

- In each ETLx unit of the **Input Channels**, the trigger output TRxy can be enabled and the trigger event can be directed to one of the OGUy units.
- One out of three **peripheral trigger** signals per OGUy can be selected as additional trigger source. These peripheral triggers are generated by on-chip peripheral modules, such as capture/compare or timer units. The selection is done by bit field EXOCONy.ISS.
- In the case that at least one **pattern detection** input is enabled (EXOCONy.IPENx) and a change of the pattern detection result from pattern match to pattern miss (or vice-versa) is detected, a trigger event is generated to indicate a pattern detection result event (if enabled by ECOCONy.GEEN).

The trigger combination offers the possibility to program different trigger criteria for several input signals (independently for each Input Channel) or peripheral signals, and to combine their effects to a single output, e.g. to generate an service request or to start an ADC conversion. This combination capability allows the generation of a service request per OGU that can be triggered by several inputs (multitude of request sources results in one reaction).

The selection is defined by the bit fields ISS in registers **ERU0_EXOCONx (x=0-3)**.

Pattern Detection

The pattern detection logic allows the combination of the status flags of all ETLx units. Each status flag can be individually included or excluded from the pattern detection for each OGUy, via control bits EXOCONy.IPENx. The pattern detection block outputs the following pattern detection results:

- **Pattern match** (EXOCONy.PDR = 1 and ERU_PDOUTy = 1):
  A pattern match is indicated while all status flags FL that are included in the pattern detection are 1.
- **Pattern miss** (EXOCONy.PDR = 0 and ERU_PDOUTy = 0):
  A pattern miss is indicated while at least one of the status flags FL that are included in the pattern detection is 0.

In addition, the pattern detection can deliver a trigger event if the pattern detection result changes from match to miss or vice-versa (if enabled by EXOCONy.GEEN = 1). The pattern result change event is logically OR-combined with the other enabled trigger events to support service request generation or to trigger other module functions (e.g. in the ADC). The event is indicated when the pattern detection result changes and EXOCONy.PDR becomes updated.

The service request generation in the OGUy is based on the trigger ERU_TOUTy that can be gated (masked) with the pattern detection result ERU_PDOUTy. This allows an automatic and reproducible generation of service requests during a certain time window, where the request event is elaborated by the trigger combination block and the time window information (gating) is given by the pattern detection. For example, service requests can be issued on a regular time base (peripheral trigger input from capture/compare unit is selected) while a combination of input signals occurs (pattern detection based on ETLx status bits).

A programmable gating scheme introduces flexibility to adapt to application requirements and allows the generation of service requests ERU_IOUTy under different conditions:

- **Pattern match** (EXOCONy.GP = 10_B):
  A service request is issued when a trigger event occurs while the pattern detection shows a pattern match.
- **Pattern miss** (EXOCONy.GP = 11_B):
  A service request is issued when the trigger event occurs while the pattern detection shows a pattern miss.
- **Independent** of pattern detection (EXOCONy.GP = 01_B):
  In this mode, each occurring trigger event leads to a service request. The pattern detection output can be used independently from the trigger combination for gating purposes of other peripherals (independent use of ERU_TOUTy and ERU_PDOUTy with service requests on trigger events).
- **No service requests** (EXOCONy.GP = 00_B, default setting)
  In this mode, an occurring trigger event does not lead to a service request. The pattern detection output can be used independently from the trigger combination for gating purposes of other peripherals (independent use of ERU_TOUTy and ERU_PDOUTy without service requests on trigger events).

6.7 Power, Reset and Clock

ERU is running on the main clock, MCLK. It is consuming power in all operating modes as long as the MCLK continues to run.

6.8 Initialization and System Dependencies

Service Requests must always be enabled at the source and at the destination. Additionally it must be checked whether it is necessary to program the ERU0 process and route a request.

Enabling Peripheral SRx Outputs

- Peripherals' SRx outputs must be selectively enabled. This procedure depends on the individual peripheral. Please look up the section "Service Request Generation" within a peripheral chapter for details.
- Optionally ERU0 must be programmed to process and route the request

Enabling External Requests

- Selected PORTS must be programmed for input
- ERU0 must be programmed to process and route the external request

Note: The number of external service request inputs may be limited by the package used.

6.9 Registers

Table 6-1 Registers Address Space

Module	Base Address	End Address	Note
ERU0	$4001\ 0600_H$	$4001\ 06FF_H$	

Registers Overview

The absolute register address is calculated by adding:

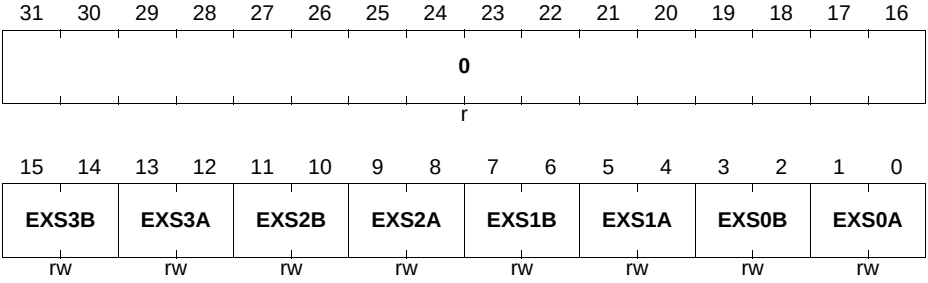
Module Base Address + Offset Address

Table 6-2 Register Overview

Short Name	Description	Offset Address	Access Mode		Description See
			Read	Write	
EXISEL	ERU External Input Control Selection	0000_H	U, PV	U, PV	Page 6-11
EXICON0	ERU External Input Control Selection	0010_H	U, PV	U, PV	Page 6-13
EXICON1	ERU External Input Control Selection	0014_H	U, PV	U, PV	Page 6-13
EXICON2	ERU External Input Control Selection	0018_H	U, PV	U, PV	Page 6-13
EXICON3	ERU External Input Control Selection	$001C_H$	U, PV	U, PV	Page 6-13
EXOCON0	ERU Output Control Register	0020_H	U, PV	U, PV	Page 6-15
EXOCON1	ERU Output Control Register	0024_H	U, PV	U, PV	Page 6-15
EXOCON2	ERU Output Control Register	0028_H	U, PV	U, PV	Page 6-15
EXOCON3	ERU Output Control Register	$002C_H$	U, PV	U, PV	Page 6-15

6.9.1 ERU Registers

ERU0_EXISEL
Event Input Select **(00_H)** **Reset Value: 0000 0000_H**

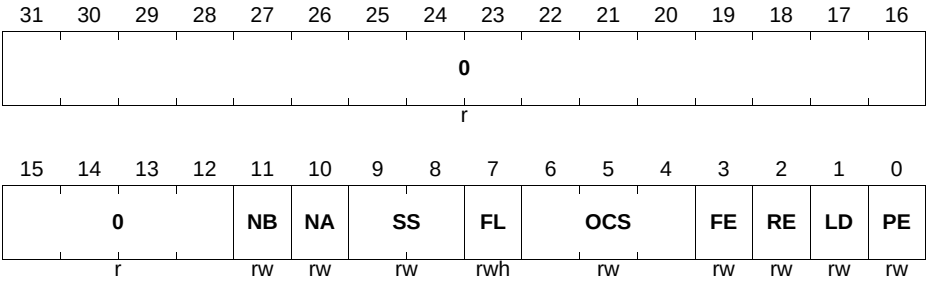


Field	Bits	Type	Description
EXS0A	[1:0]	rw	**Event Source Select for A0 (ERS0)** This bit field defines which input is selected for A0. 00_B Input ERU_0A0 is selected 01_B Input ERU_0A1 is selected 10_B Input ERU_0A2 is selected 11_B Input ERU_0A3 is selected
EXS0B	[3:2]	rw	**Event Source Select for B0 (ERS0)** This bit field defines which input is selected for B0. 00_B Input ERU_0B0 is selected 01_B Input ERU_0B1 is selected 10_B Input ERU_0B2 is selected 11_B Input ERU_0B3 is selected
EXS1A	[5:4]	rw	**Event Source Select for A1 (ERS1)** This bit field defines which input is selected for A1. 00_B Input ERU_1A0 is selected 01_B Input ERU_1A1 is selected 10_B Input ERU_1A2 is selected 11_B Input ERU_1A3 is selected
EXS1B	[7:6]	rw	**Event Source Select for B1 (ERS1)** This bit field defines which input is selected for B1. 00_B Input ERU_1B0 is selected 01_B Input ERU_1B1 is selected 10_B Input ERU_1B2 is selected 11_B Input ERU_1B3 is selected

Field	Bits	Type	Description
EXS2A	[9:8]	rw	**Event Source Select for A2 (ERS2)** This bit field defines which input is selected for A2. 00_B Input ERU_2A0 is selected 01_B Input ERU_2A1 is selected 10_B Input ERU_2A2 is selected 11_B Input ERU_2A3 is selected
EXS2B	[11:10]	rw	**Event Source Select for B2 (ERS2)** This bit field defines which input is selected for B2. 00_B Input ERU_2B0 is selected 01_B Input ERU_2B1 is selected 10_B Input ERU_2B2 is selected 11_B Input ERU_2B3 is selected
EXS3A	[13:12]	rw	**Event Source Select for A3 (ERS3)** This bit field defines which input is selected for A3. 00_B Input ERU_3A0 is selected 01_B Input ERU_3A1 is selected 10_B Input ERU_3A2 is selected 11_B Input ERU_3A3 is selected
EXS3B	[15:14]	rw	**Event Source Select for B3 (ERS3)** This bit field defines which input is selected for B3. 00_B Input ERU_3B0 is selected 01_B Input ERU_3B1 is selected 10_B Input ERU_3B2 is selected 11_B Input ERU_3B3 is selected
0	[31:16]	r	**Reserved** Read as 0; should be written with 0.

ERU0_EXICONx (x=0-3)
Event Input Control x

(10_H + 4*x) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
PE	0	rw	**Output Trigger Pulse Enable for ETLx** This bit enables the generation of an output trigger pulse at TRxy when the selected edge is detected (set condition for the status flag FL). 0_B The trigger pulse generation is disabled 1_B The trigger pulse generation is enabled
LD	1	rw	**Rebuild Level Detection for Status Flag for ETLx** This bit selects if the status flag FL is used as “sticky” bit or if it rebuilds the result of a level detection. 0_B The status flag FL is not cleared by hardware and is used as “sticky” bit. Once set, it is not influenced by any edge until it becomes cleared by software. 1_B The status flag FL rebuilds a level detection of the desired event. It becomes automatically set with a rising edge if RE = 1 or with a falling edge if FE = 1. It becomes automatically cleared with a rising edge if RE = 0 or with a falling edge if FE = 0.
RE	2	rw	**Rising Edge Detection Enable ETLx** This bit enables/disables the rising edge event as edge event as set condition for the status flag FL or as possible trigger pulse for TRxy. 0_B A rising edge is not considered as edge event 1_B A rising edge is considered as edge event

Field	Bits	Type	Description
FE	3	rw	**Falling Edge Detection Enable ETLx** This bit enables/disables the falling edge event as edge event as set condition for the status flag FL or as possible trigger pulse for TRxy. 0_B A falling edge is not considered as edge event 1_B A falling edge is considered as edge event
OCS	[6:4]	rw	**Output Channel Select for ETLx Output Trigger Pulse** This bit field defines which Output Channel OGUy is targeted by an enabled trigger pulse TRxy. 000_B Trigger pulses are sent to OGU0 001_B Trigger pulses are sent to OGU1 010_B Trigger pulses are sent to OGU2 011_B Trigger pulses are sent to OGU3 Others: Reserved, do not use this combination
FL	7	rwh	**Status Flag for ETLx** This bit represents the status flag that becomes set or cleared by the edge detection. 0_B The enabled edge event has not been detected 1_B The enabled edge event has been detected
SS	[9:8]	rw	**Input Source Select for ERSx** This bit field defines which logical combination is taken into account as ERSxO. 00_B Input A without additional combination 01_B Input B without additional combination 10_B Input A OR input B 11_B Input A AND input B
NA	10	rw	**Input A Negation Select for ERSx** This bit selects the polarity for the input A. 0_B Input A is used directly 1_B Input A is inverted
NB	11	rw	**Input B Negation Select for ERSx** This bit selects the polarity for the input B. 0_B Input B is used directly 1_B Input B is inverted
0	[31:12]	r	**Reserved** Read as 0; should be written with 0.

ERU0_EXOCONx (x=0-3)
Event Output Trigger Control x

(20_H + 4*x) **Reset Value: 0000 0008$_H$**

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
IPEN3	IPEN2	IPEN1	IPEN0	0						GP		PDR	GEEN	ISS	
rw	rw	rw	rw	r						rw		rh	rw	rw	

Field	Bits	Type	Description
ISS	[1:0]	rw	**Internal Trigger Source Selection** This bit field defines which input is selected as peripheral trigger input for OGUy. 00_B The peripheral trigger function is disabled 01_B Input ERU_OGUy1 is selected 10_B Input ERU_OGUy2 is selected 11_B Input ERU_OGUy3 is selected
GEEN	2	rw	**Gating Event Enable** Bit GEEN enables the generation of a trigger event when the result of the pattern detection changes from match to miss or vice-versa. 0_B The event detection is disabled 1_B The event detection is enabled
PDR	3	rh	**Pattern Detection Result Flag** This bit represents the pattern detection result. 0_B A pattern miss is detected 1_B A pattern match is detected

Field	Bits	Type	Description
GP	[5:4]	rw	**Gating Selection for Pattern Detection Result** This bit field defines the gating scheme for the service request generation (relation between the OGU output ERU_PDOUTy and ERU_GOUTy). 00_B ERU_GOUTy is always disabled and ERU_IOUTy can not be activated 01_B ERU_GOUTy is always enabled and ERU_IOUTy becomes activated with each activation of ERU_TOUTy 10_B ERU_GOUTy is equal to ERU_PDOUTy and ERU_IOUTy becomes activated with an activation of ERU_TOUTy while the desired pattern is detected (pattern match PDR = 1) 11_B ERU_GOUTy is inverted to ERU_PDOUTy and ERU_IOUTy becomes activated with an activation of ERU_TOUTy while the desired pattern is not detected (pattern miss PDR = 0)
IPENx (x = 0-3)	12+x	rw	**Pattern Detection Enable for ETLx** Bit IPENx defines whether the trigger event status flag EXICONx.FL of ETLx takes part in the pattern detection of OGUy. 0_B Flag EXICONx.FL is excluded from the pattern detection 1_B Flag EXICONx.FL is included in the pattern detection
0	[31:16], [11:6]	r	**Reserved** Read as 0; should be written with 0.

6.10 Interconnects

This section describes how the ERU0 module is connected within the XMC1100 system.

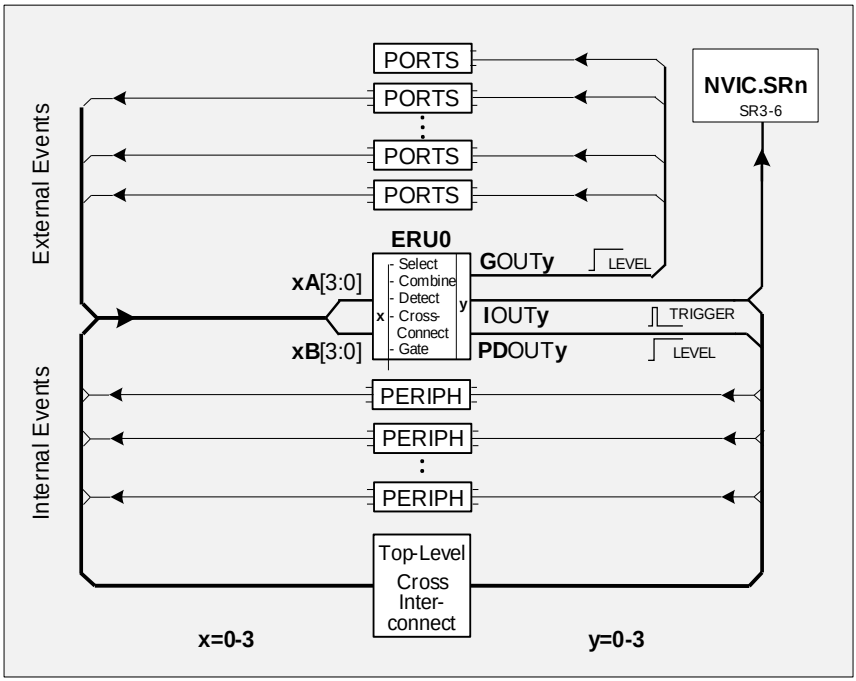


Figure 6-6 ERU Interconnects Overview

6.10.1 ERU0 Connections

The following table shows the ERU0 connections.

Table 6-3 ERU0 Pin Connections

Global Inputs/Outputs	Connected To	I/O	Description
ERU0.0A0	reserved	I	
ERU0.0A1	P2.4	I	
ERU0.0A2	reserved	I	
ERU0.0A3	VADC0.G0BFLOUT0	I	from ADC boundary flag
ERU0.0B0	P2.0	I	
ERU0.0B1	P2.2	I	

Table 6-3 ERU0 Pin Connections

Global Inputs/Outputs	Connected To	I/O	Description
ERU0.0B2	reserved	I	
ERU0.0B3	VADC0.G1BFLOUT0	I	from ADC boundary flag
ERU0.1A0	reserved	I	
ERU0.1A1	P2.5	I	
ERU0.1A2	reserved	I	
ERU0.1A3	VADC0.G0BFLOUT1	I	from ADC boundary flag
ERU0.1B0	P2.1	I	
ERU0.1B1	P2.3	I	
ERU0.1B2	reserved	I	
ERU0.1B3	VADC0.G1BFLOUT1	I	from ADC boundary flag
ERU0.2A0	reserved	I	
ERU0.2A1	P2.6	I	
ERU0.2A2	reserved	I	
ERU0.2A3	VADC0.G0BFLOUT2	I	from ADC boundary flag
ERU0.2B0	P2.10	I	
ERU0.2B1	P2.11	I	
ERU0.2B2	reserved	I	
ERU0.2B3	VADC0.G1BFLOUT2	I	from ADC boundary flag
ERU0.3A0	reserved	I	
ERU0.3A1	P2.7	I	
ERU0.3A2	reserved	I	
ERU0.3A3	VADC0.G0BFLOUT3	I	from ADC boundary flag
ERU0.3B0	P2.9	I	
ERU0.3B1	P2.8	I	
ERU0.3B2	reserved	I	
ERU0.3B3	VADC0.G1BFLOUT3	I	from ADC boundary flag
ERU0.OGU01	CCU40.SR0	I	
ERU0.OGU02	VADC0.C0SR2	I	
ERU0.OGU03	reserved	I	
ERU0.OGU11	CCU40.SR1	I	

Table 6-3 ERU0 Pin Connections

Global Inputs/Outputs	Connected To	I/O	Description
ERU0.OGU12	VADC0.C0SR2	I	
ERU0.OGU13	reserved	I	
ERU0.OGU21	CCU40.SR2	I	
ERU0.OGU22	VADC0.C0SR3	I	
ERU0.OGU23	reserved	I	
ERU0.OGU31	CCU40.SR3	I	
ERU0.OGU32	VADC0.C0SR3	I	
ERU0.OGU33	reserved	I	
ERU0.PDOUT0	VADC0.BGREQGTO VADC0.G0REQGTO VADC0.G1REQGTO CCU40.IN1D CCU40.IN0J reserved USIC0_CH1.HWIN0 P0.0 P2.11	O	
ERU0.GOUT0	P0.0 P2.11	O	
ERU0.TOUT0	not connected	O	
ERU0.IOUT0	NVIC.ERU0.SR0 VADC0.BGREQTRM VADC0.G0REQTRM VADC0.G1REQTRM CCU40.CLKB CCU40.IN0K	O	
ERU0.PDOUT1	VADC0.BGREQGTP VADC0.G0REQGTP VADC0.G1REQGTP CCU40.IN0D CCU40.IN1J USIC0_CH1.HWIN2 P0.1 P2.10	O	

Table 6-3 ERU0 Pin Connections

Global Inputs/Outputs	Connected To	I/O	Description
ERU0.GOUT1	P0.1 P2.10	O	
ERU0.TOUT1	not connected	O	
ERU0.IOUT1	NVIC.ERU0.SR1 VADC0.BGREQTRN VADC0.G0REQTRN VADC0.G1REQTRN CCU40.CLKC CCU40.IN1K	O	
ERU0.PDOUT2	VADC0.BGREQGTK VADC0.G0REQGTK VADC0.G1REQGTK CCU40.IN3D CCU40.IN2J P0.2 P2.1	O	
ERU0.GOUT2	P0.2 P2.1	O	
ERU0.TOUT2	not connected	O	
ERU0.IOUT2	NVIC.ERU0.SR2 VADC0.BGREQTRG VADC0.G0REQTRG VADC0.G1REQTRG CCU40.IN2K	O	
ERU0.PDOUT3	VADC0.BGREQGTL VADC0.G0REQGTL VADC0.G1REQGTL CCU40.IN3J CCU40.IN2D P0.3 P2.0	O	
ERU0.GOUT3	P0.3 P2.0	O	

Table 6-3 ERU0 Pin Connections

Global Inputs/Outputs	Connected To	I/O	Description
ERU0.TOUT3	not connected	O	
ERU0.IOUT3	NVIC.ERU0.SR3 VADC0.BGREQTRH VADC0.G0REQTRH VADC0.G1REQTRH CCU40.IN3K	O	

On-Chip Memories

7 Memory Organization

This chapter provides description of the system memory organization, memory accesses and memory protection strategy.

References

[5] Cortex®-M0 User Guide, ARM DUI 0497A (ID112109)

7.1 Overview

The Memory Map in XMC1100 is based on standard ARM Cortex-M0 system memory map.

7.1.1 Features

The Memory Map implements the following features:

- Compatibility with standard ARM Cortex-M0 CPU **[5]**
- Full compatibility across entire XMC1000 Family

7.2 Memory Map

Table 7-1 defines detailed system memory map of XMC1100 where each individual peripheral or memory instance implement its own address spaces. For detailed register description of the system components and peripherals, please refer to respective chapters of this document.

Note: Depending on the device variant, not all peripherals and memory address ranges may be available.

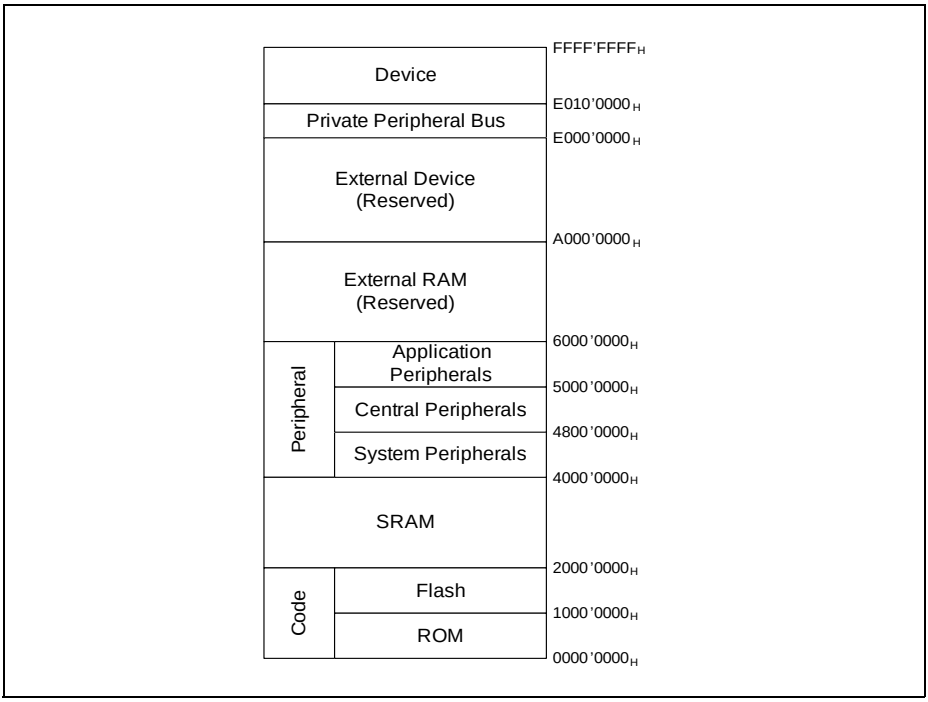


Figure 7-1 XMC1100 Address Space

Table 7-1 Memory Map

Address space	Address Range	Description	Access Type[1]	
			Read	Write
Code	00000000_H - $00000AFF_H$	ROM (user-readable)	U, PV	nBE
	$00000B00_H$ - $00001FFF_H$	ROM (non-user-readable)	BE	BE
	00002000_H - $0FFFFFFF_H$	reserved	BE	BE
	10000000_H - $10000DFF_H$	Flash Sector 0 (non-user-readable)	nBE	nBE
	$10000E00_H$ - $10000FFF_H$	Flash Sector 0 (user-readable)	U, PV	nBE
	10001000_H - $10010FFF_H$	Flash (64 Kbytes)	U, PV	U, PV
	10011000_H - $1FFFFFFF_H$	reserved	BE	BE
SRAM[2]	20000000_H - $20000FFF_H$	SRAM Block 0	U, PV	U, PV
	20001000_H - $20001FFF_H$	SRAM Block 1	U, PV	U, PV
	20002000_H - $20002FFF_H$	SRAM Block 2	U, PV	U, PV
	20003000_H - $20003FFF_H$	SRAM Block 3	U, PV	U, PV
	20004000_H - $3FFFFFFF_H$	reserved	BE	BE

Table 7-1 Memory Map (cont'd)

Address space	Address Range	Description	Access Type[1)]	
			Read	**Write**
System Peripherals	40000000_H - $400007FF_H$	Memory Control	U, PV	U, PV
	40000800_H - $4000FFFF_H$	reserved	BE	BE
	40010000_H - $40010FFF_H$	SCU (including RTC)	U, PV	U, PV
	40011000_H - $4001107F_H$	ANACTRL	U, PV	U, PV
	40011080_H - $4001FFFF_H$	reserved	BE	BE
	40020000_H - $4002001F_H$	WDT	U, PV	U, PV
	40020020_H - $4003FFFF_H$	reserved	BE	BE

Table 7-1 Memory Map (cont'd)

Address space	Address Range	Description	Access Type[1)]	
			Read	Write
System Peripherals (cont'd)	40040000_H - $4004007F_H$	Port 0	U, PV	U, PV
	40040080_H - $400400FF_H$	reserved	BE	BE
	40040100_H - $4004017F_H$	Port 1	U, PV	U, PV
	40040180_H - $400401FF_H$	reserved	BE	BE
	40040200_H - $4004027F_H$	Port 2	U, PV	U, PV
	40040280_H - $4004FFFF_H$	reserved	BE	BE
	40050000_H - $400500DF_H$	Flash Registers	U, PV	U, PV
	$400500E0_H$ - $47FFFFFF_H$	reserved	BE	BE
Central Peripherals	48000000_H - $480001FF_H$	USIC0 Channel 0	U, PV	U, PV
	48000200_H - $480003FF_H$	USIC0 Channel 1	U, PV	U, PV
	48000400_H - $480007FF_H$	USIC0 RAM	nBE	BE
	48000800_H - $4801FFFF_H$	reserved	BE	BE
	48020000_H - $4802000F_H$	PRNG	U, PV	U, PV
	48020010_H - $4802FFFF_H$	reserved	BE	BE

Table 7-1 Memory Map (cont'd)

Address space	Address Range	Description	Access Type[1]	
			Read	Write
Central Peripherals (cont'd)	48030000_H - $480303FF_H$	VADC0 General and Global Registers	U, PV	U, PV
	$48030C00_H$ - $48033FFF_H$	reserved	BE	BE
	48034000_H - $480341FF_H$	SHS0	U, PV	U, PV
	48034200_H - $4803FFFF_H$	reserved	BE	BE
	48040000_H - $480401FF_H$	CCU40 CC40 and Kernel Registers	U, PV	U, PV
	48040200_H - $480402FF_H$	CCU40 CC41	U, PV	U, PV
	48040300_H - $480403FF_H$	CCU40 CC42	U, PV	U, PV
	48040400_H - $480404FF_H$	CCU40 CC43	U, PV	U, PV
	48040500_H - $4FFFFFFF_H$	reserved	BE	BE
Peripheral	50000000_H - $5FFFFFFF_H$	reserved	BE	BE
External SRAM	60000000_H - $9FFFFFFF_H$	reserved	BE	BE
External Device	$A0000000_H$ - $DFFFFFFF_H$	reserved	BE	BE
Private Peripheral Bus	$E0000000_H$ - $E00FFFFF_H$	NVIC, System timer, System Control Block	U, PV	U, PV
Vendor specific 1	$E0100000_H$ - $EFFFFFFF_H$	reserved	BE	BE
Vendor specific 2	$F0000000_H$ - $F0000FFF_H$	System ROM Table	U, PV	nBE
	$F0001000_H$ - $FFFFFFFF_H$	reserved	BE	BE

1) For address ranges taken up by peripherals, the access type for each address in the range may differ from that shown in the table. Refer to respective chapters for details.
2) The address range $2000'0000_H$ to $2000'01FF_H$ will be overwritten by start-up software during device start-up.

7.3 Memory Access

This section describes the memory accesses to the different type of memories in XMC1100.

7.3.1 Flash Memory Access

The XMC1100 provides up to 64 Kbytes of Flash memory for instruction code or constant data, starting at address $1000'1000_H$. This excludes Flash sector 0, which is used to store system information and is always read only.

For details of Flash memory access, refer to the Flash Architecture chapter.

7.3.2 SRAM Access

The XMC1100 provides 16 Kbytes of SRAM for instruction code or constant data, as well as system variables such as the system stack, starting at address $2000'0000_H$.

The SRAM supports 8-bit, 16-bit and 32-bit writes, and generates one parity bit for each 8 bits of written data. A read operation will check for parity errors on the 32-bit read data. Accesses to the SRAM require no wait states.

The 16 Kbytes of SRAM is logically divided into four blocks of 4 Kbytes each. Accesses to blocks 1, 2 and 3 can be disabled and enabled again during run-time with the peripheral privilege access scheme. See PAU chapter for details.

Note: The address range $2000'0000_H$ to $2000'01FF_H$ will be overwritten by the start-up software during device start-up. Therefore, these addresses should not be targeted during the download of code/data by the bootstrap loader into the SRAM nor should they store critical data that are still needed by the application after a system reset or SW master reset.

7.3.3 ROM Access

The XMC1100 provides 8 Kbytes of ROM, which contains the startup software, vector table and user routines.

Read accesses to the ROM require no wait states.

7.4 Memory Protection Strategy

Two aspects of memory protection are considered:

1. Intellectual Property (IP) Protection
2. Memory Access Protection during Run-time

The memory protection measures available in XMC1100 are listed in **Table 7-2**.

Table 7-2 Memory Protection Measures

Protection Aspects	Protection Measures	Protection Target
IP protection	Blocking of unauthorized external access	Flash memory contents
Memory access protection	Bit protection scheme	Specific system-critical registers/bit fields
	Peripheral privilege access control	Specific address ranges; each range can be controlled independently

7.4.1 Intellectual Property (IP) Protection

IP protection refers to the prevention against unauthorized read out of critical data and user IP from Flash memory.

7.4.1.1 Blocking of Unauthorized External Access

In XMC1100, the Boot Mode Index (BMI) is used to control the boot options such that once the BMI is programmed to enter user mode (productive), it is not allowed to enter the other boot modes without an erase of the complete user Flash (including sector 0).

Therefore, boot options that load and execute external code, including unauthorized code that might read out the Flash memory contents, will be blocked and only user code originating from the Flash memory can be executed.

7.4.2 Memory Access Protection during Run-time

Memory access protection refers to the prevention against unintended write access on a memory address space during run-time.

7.4.2.1 Bit Protection Scheme

The bit protection scheme prevents direct software writing of selected register bits (i.e., protected bits) using the PASSWD register in the SCU module. When the bit field MODE is 11_B, writing 10011_B to the bit field PASS opens access to writing of all protected bits, and writing 10101_B to the bit field PASS closes access to writing of all protected bits. In both cases, the value of the bit field MODE is not changed even if PASSWD register is written with 98_H or $A8_H$. It can only be changed when bit field PASS is written with 11000_B, for example, writing $0000'00C0_H$ to PASSWD register disables the bit protection scheme.

Access is opened for maximum 32 MCLKs if the “close access” password is not written. If “open access” password is written again before the end of 32 MCLK cycles, there will be a recount of 32 MCLK cycles.

Table 7-3 shows the list of protected bit in XMC1100.

Table 7-3 List of Protected Register Bit Fields

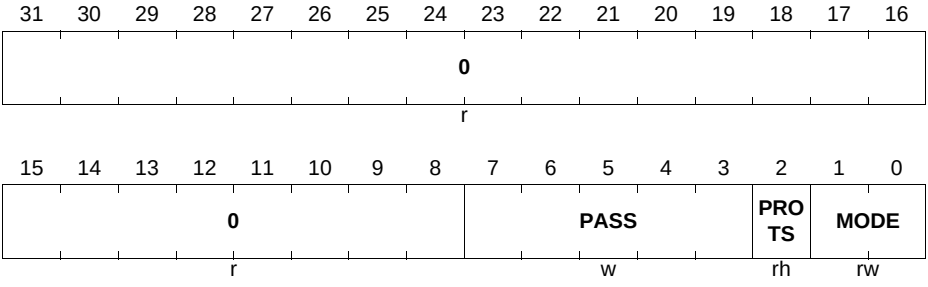
Register	Bit Field
SCU_CLKCR	FDIV, IDIV, PCLKSEL, RTCLKSEL
SCU_CGATSET0	All bits
SCU_CGATCLR0	All bits
VADC0_ACCPROT0	All bits
VADC0_ACCPROT1	All bits

Write to all protected registers except VADC0_ACCPROT[1:0], without opening access through bit field PASS, will be ignored. No bus error will be generated. User should read back the register value to ensure the write has taken place.

Write to VADC0_ACCPROT[1:0], without opening access through bit field PASS, will trigger a hard fault. If an interrupt occurs immediately after the access is opened and the interrupt service routine requires more than 32 MCLK cycles, the write to the register will happen after the access is closed and thereby, triggering a hard fault. To avoid this, interrupts should be disabled before writing to these two registers.

The PASSWD register is also described in the SCU chapter.

SCU_PASSWD
Password Register **(4001 0024$_H$)** **Reset Value: 0000 0007$_H$**



Field	Bits	Type	Description
MODE	[1:0]	rw	**Bit Protection Scheme Control Bits** 00_B Scheme disabled - direct access to the protected bits is allowed. 11_B Scheme enabled - the bit field PASS has to be written with the passwords to open and close the access to the protected bits. (Default) Others: Scheme enabled, similar to the setting for MODE = 11_B. These two bits cannot be written directly. To change the value between 11_B and 00_B, the bit field PASS must be written with 11000_B. Only then will the MODE bit field be registered.
PROTS	2	rh	**Bit Protection Signal Status Bit** This bit shows the status of the protection. 0_B Software is able to write to all protected bits. 1_B Software is unable to write to any of the protected bits.
PASS	[7:3]	w	**Password Bits** This bit protection scheme only recognizes the following three passwords: 11000_BEnables writing of the bit field MODE. 10011_BOpens access to writing of all protected bits. 10101_BCloses access to writing of all protected bits.
0	[31:8]	r	**Reserved**

7.4.2.2 Peripheral Privilege Access Control

All CPU accesses are privileged accesses. In XMC1100, a separate scheme called Peripheral Privilege Access Control, which allows a peripheral's memory address space to be disabled to block any unintended write or read access, is provided. The address space can be re-enabled when necessary.

Refer to the PAU chapter for details.

8 Flash Architecture

This chapter describes the non volatile memory (NVM) module. The NVM has the following features:

- Reading by word, writing by block and erasing by page (256 bytes).
- Fast personalization support.
- Automatic verify support.
- Up to 50,000 erase cycles per page.
- Minimum data retention of 10 years at 25°C in cells that were never previously programmed.
- Flash programming voltage generated on-chip.
- Configurable erase and write protection.
- Power saving sleep mode.
- Incremental write without erase for semaphores.
- Redundancy sector for yield improvement.

8.1 Definitions

Throughout the document the terms NVM (Non Volatile Memory) and Flash are used as synonyms, disregarding the fact that NVM describes a broader class of memories, where the described Flash is only a special case.

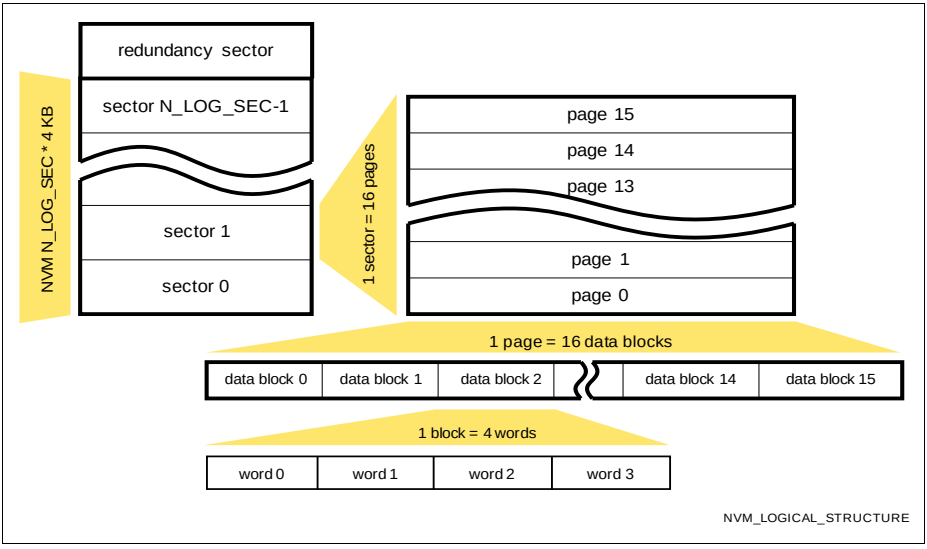


Figure 8-1 Logical structure of the NVM module

8.1.1 Logical and Physical States

Erasing

The erased state of a cell is ‘1’. Forcing an NVM cell to this state is called erasing. Erasing is possible with a granularity of a page (see below).

Writing

The written state of a cell is ‘0’. Changing an erased cell to this state is called writing. Writing is possible with a granularity of a block (see below).

Programming

The combination of erasing and writing is called programming. Programming often means also writing a previously erased page.

*Note: The termini **write** and **writing** are also used for accessing special function registers. The meaning depends therefore on the context.*

8.1.2 Data Portions

Word

A word consists of 32 bits. A word represents the data size which is read from or written to the NVM module within one access cycle.

Block

A block consists of 4 words (128 bit data, extended by 4 bit parity, and 6 bit ECC). A block represents the smallest data portion that can be written.

Page

A page consists of 16 blocks.

Sector

A sector consists of 16 pages.

8.1.3 Address Types

Physical Address

Address of the CPU system.

Base Address or Memory Base Address

Physical address of the lower boundary for memory accesses of an NVM module.

Logical Address

Memory address offset inside the NVM module: If the memory is addressed, the logical address is the physical address subtracted by the base address of the module.

Sector Address

Module specific part of the logical address specifying the sector, for calculation see **Section 8.2.1**.

Page Address

Module specific part of the logical address, for calculation see **Section 8.2.1**.

8.1.4 Module Specific Definitions

The following table defines NVM specific constants used throughout this chapter.

Table 8-1 Module Specific Definitions

Module size [KB]	**204**	
Constant name	**Value**	**Comment**
N_BLOCKS	16	Number of blocks per page
N_PAGES	16	Number of pages per sector
N_SECTORS	52	Number of sectors including redundancy
N_LOG_SEC	51	Number of sectors excluding redundancy

8.2 Module Components

8.2.1 Memory Cell Array

The non-volatile memory cells are organized in sectors, which consist of pages, which on their part are structured in blocks.

A logical memory address addr has the following structure:

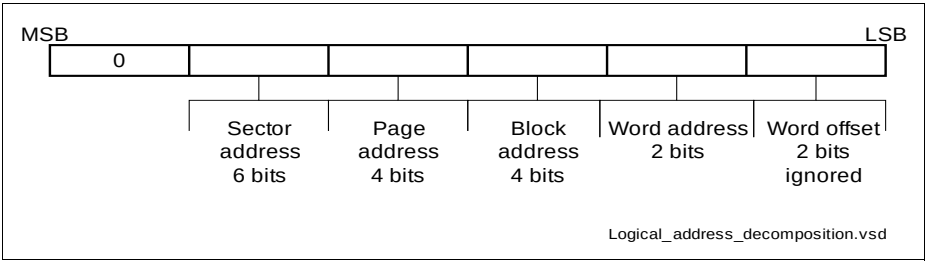


Figure 8-2 Decomposition of Logical Address

- addr[1:0] = word offset, for a memory address undefined and ignored
- addr[3:2] = word address
- addr[7:4] = block address
- addr[11:8] = page address
- addr[17:12] = sector address.

The memory subsystem always accesses whole words, therefore the word offset is ignored by the NVM module.

8.2.1.1 Page

Each page consists of 16 data blocks of 138 bits each (including parity bits and ECC bits).

A page is the granularity of data that can be erased in the cell array.

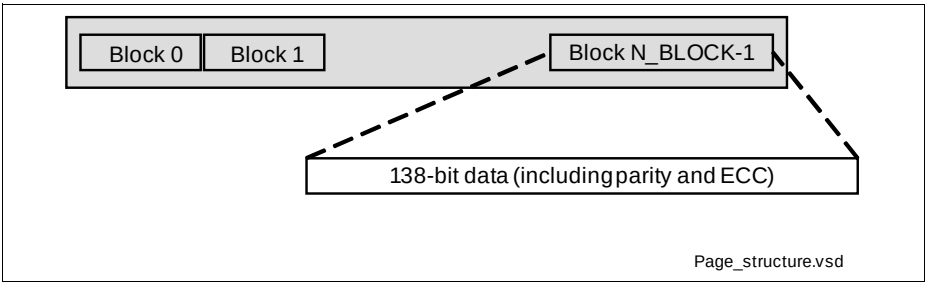


Figure 8-3 Structure of a Page

8.2.1.2 Sector

16 pages form a sector.

The whole cell array is build of 52 sectors.

8.3 Functional Description

The NVM module supports read and write accesses to the memory and to the special function registers (SFRs). No read-modify-write mechanism is supported for SFRs.

The main tasks of the NVM module are reading and writing from/to the memory array.

8.3.1 SFR Accesses

Reading the special function registers is possible in every mode of the NVM module.

Register write is not possible while the state machine is busy. The write is stalled in this case. For other exceptions see **Section 8.5.2**.

8.3.2 Memory Read

The NVM memory can be read with a minimum granularity of a word provided that the word is in the memory address range of the module.

If the word is not within the memory address range of the NVM module, the module does not react at all and a different memory module may handle the access.

Memory read accesses are only possible while no FSM procedure (erase, write, verify, sleep or wake-up) is in progress. A memory read access while the FSM is busy is stalled until the FSM is idle again. Then the access is carried out. Such a stall will also stop the CPU from executing code.

The NVM will insert waitstates during memory read automatically if needed.

8.3.3 Memory Write

From the user's viewpoint data is written directly to the memory array. Module internally one block is buffered and the ECC bits are calculated. Then a finite state machine is started that writes the ECC and data bits to the memory field.

Writing does not support wrap-around, i.e. writing of a block has always to start with word 0 of the selected block and then automatically continues in ascending order up to word 3.

Since a block has to be written in four word writing steps, a special procedure has to be followed to transfer complete block data to the NVM: Writing of a block has always to start with word 0 of the addressed block. The following three writes need to address the other three words of the same block in ascending order. If this rule is not observed, the already provided words are discarded. If in this case the last provided word is addressing a word 0, this word is already accepted as the first word of a new block write procedure. Intermediate read operations do not influence the write data transfer. Intermediate read operations targeting the same address as the interrupted write operation are served from the memory array, i.e. do not read back the already transferred write data.

To write to the memory array, the NVM module has to be set to one-shot or continuous write mode by setting the SFR **NVMPROG** to the corresponding value. Data to be written

can now be written to the desired address. It is not checked, if the addressed block was erased before. Writing to a non erased block leads to corrupted data since writing can only clear bits but not set any bits. Reading such a block will lead most probably to an ECC fault.

A write access to the NVM when not in write mode does not trigger an exception or interrupt. The data is lost. Writes to the NVM are also used to trigger other operations which are described in the following subsections. Similarly, a write access to the NVM module has no effect and the data is lost when a protected sector is addressed (see **Section 8.3.6**).

Depending on the settings of **NVMPROG**, an automatic verify is performed (see **Section 8.3.5**).

In case of a one-shot write, write mode is automatically left. In case of continuous write mode, further write operations to new addresses can follow, until the write mode is explicitly left. Continuous write operations can target blocks within the complete memory without any restriction regarding sector or page borders.

8.3.4 Memory Erase

Only whole pages can be physically erased, i.e. all bits of the addressed page are physically set to '1'.

To erase a page in the memory field, the NVM module has to be set to one-shot or continuous page erase mode by setting the SFR **NVMPROG** to the corresponding value. A write access to a memory location specifies the address of the page to be erased and triggers the erase.

A write access to the NVM when not in page erase mode does not trigger an exception or interrupt because they are also used to trigger other operations; as described in this section. Similarly, no erase is triggered when a protected sector is addressed (see **Section 8.3.6**).

In case of a one-shot page erase, page erase mode is automatically left. In case of continuous page erase mode, further page erase operations can follow, until the page erase mode is explicitly left.

8.3.5 Verify

The data written as described in **Section 8.3.3** can be verified automatically. The written data in the cell array can be automatically compared to the data still available in the module internal buffer. This is automatically performed two times with hardread written and hardread erased. These hardread levels provide some margin compared to the normal read level to ensure that the data is really programmed with suitably distinct levels for written and erased bits. The verification result can be read at SFR **NVMSTATUS**.

Stand-alone verify operations can also be started by setting the SFR **NVMPROG** to the corresponding value. In this case, data written to a memory location is compared with the content of the memory field at the specified address. The comparison here is performed just once with a read level chosen before from normal read, hardread written and hardread erased.

A write access to the NVM when not in verification mode does not trigger an exception or interrupt because it is also used to trigger other operations; as described in this section. Similarly, a write access to the NVM module has no effect, when a protected sector is addressed (see **Section 8.3.6**).

In case of a one-shot verify, verify mode is automatically left. In case of continuous verify mode, further verify operations can follow, until the verify mode is explicitly left.

Note: A continuous write operation without automatic verification, followed by two stand-alone verifications with 'hardread written' and 'hardread erased', is faster than a write operation with continuous automatic verification, since in the second case for every block write the hardread level has to be changed twice, whereas in the first case this change is performed only two times for the complete write data.
On the other hand, for the continuous automatic verification the reference data for the verification is directly available within the NVM module, whereas for the stand-alone verification the reference data needs to be provided again by the CPU.

8.3.6 Erase-Protection and Write-Protection

By setting **NVMCONF**.SECPROT, a configurable number of sectors can be selected to be protected from any modification, i.e. a range of sectors starting with sector 0 can be defined to be erase-protected and write-protected.

8.4 Redundancy

To increase the production yield, the NVM module contains redundancy. Redundancy is transparent to the user.

8.5 Power Saving Modes

The NVM module supports the following power saving modes. The modes differ in power consumption and in the time to restart the memory.

8.5.1 Idle Mode

Idle mode is entered after reset after charging the pumps. In idle mode the power consumption is less than during a memory read access or write access.

Any operation of the NVM module (SFR access or memory access) can be started from idle mode without time delay.

8.5.2 Sleep Mode

Entering and leaving NVM sleep mode requires special state machine sequences, which need some time. Therefore, besides SFR accesses, the NVM module cannot be used immediately after wake-up from sleep mode, as indicated by **NVMSTATUS**.BUSY.

NVM sleep mode is entered via WFE/WFI when Flash Power down is activeted, see SCU chapter. Alternatively, sleep mode for the NVM module can be triggered by writing the respective bit in SFR **NVMCONF**.

In NVM sleep mode only the SFR **NVMCONF** can be read and written, all other SFRs can only be read. No memory access is possible in sleep mode.

8.6 Properties and Implementation of Error Correcting Code (ECC)

The error correcting code (ECC) for the data blocks is a one-bit error correction and a (partial) double-bit error detection for every data block of 128 bit, which uses 4 parity bits and 6 ECC bits in addition to the protected 128 data bits. The correct ECC bits for every block are generated automatically when the data is written.

8.7 NVM SFRs

Table 8-3 shows a complete list of the NVM special function registers.

Table 8-2 Registers Address Space

Module	Base Address	End Address	Note
NVM	$4005\ 0000_H$	$4005\ 00FF_H$	

Table 8-3 Registers Overview

Register Short Name	Register Long Name	Offset Address	Page Number
NVM SFRs, Register Descriptions			
NVMSTATUS	NVM Status Register	0000_H	**Page 8-10**
NVMPROG	NVM Programming Control Register	0004_H	**Page 8-12**
NVMCONF	NVM Configuration Register	0008_H	**Page 8-15**

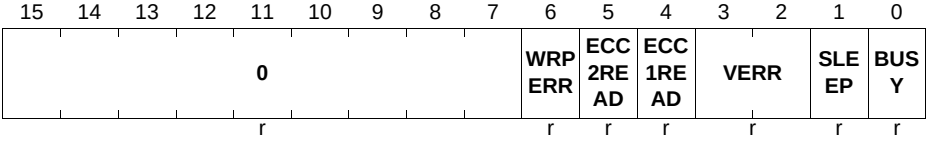
The register is addressed wordwise.

Reading an SFR is not blocked, while the NVM module is busy. If the SFR value depends on the completion of an NVM sequence, **NVMSTATUS**.BUSY must be polled for 0_B, before the SFR is read. Otherwise, reading the SFR might yield a value that is not updated yet.

8.7.1 Register Descriptions

NVM Status Register

NVMSTATUS
NVM Status Register **(0000_H)** **Reset Value: 0002_H**



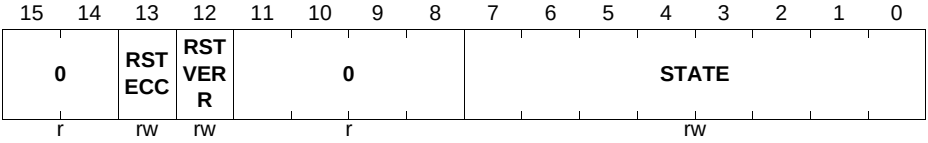
Field	Bits	Type	Description
0	15:7	r	**Reserved** Read as 0.
WRPERR	6	r	**Write Protocol Error** The flag accumulates write protocol violations during the last 4-word write operations (for write or verify). It is also set when a triggered operation was ignored because of write protection. The correct protocol is defined in **Section 8.3.3**. It is reset by hardware when **NVMPROG**.RSTECC is written. 0_B **WRPROTOK**, No write protocol failure occurred. 1_B **WRPROTFAIL**, At least one write protocol failure was detected.
ECC2READ	5	r	**ECC2 Read**[1)] The flag accumulates ECC two bit failure during memory read operations. It is reset by hardware when **NVMPROG**.RSTECC is written. 0_B **ECC2RDOK**, No ECC two bit failure during memory read operations. 1_B **ECC2RDFAIL**, At least one ECC two bit failure was detected.

Field	Bits	Type	Description
ECC1READ	4	r	**ECC1 Read**[1)] The flag accumulates ECC single bit failure during the last memory read operations. It is reset by hardware when **NVMPROG**.RSTECC is written. 0_B **ECC1RDOK**, No ECC single bit failure occurred. 1_B **ECC1RDFAIL**, At least one ECC single bit failure was detected and corrected.
VERR	3:2	r	**Verify Error** The flag is reset by hardware, when **NVMPROG**.RSTVERR is written. The flag is also reset, when write mode or verify-only mode are entered, i.e. **NVMPROG**.ACTION.OPTYPE = 0001_B or **NVMPROG**.ACTION.VERIFY = 11_B. The flag accumulates, i.e. VERR is updated every time a value higher than the current value is required, until write mode or verify-only mode are left (automatically in case of a one-shot write operation). Information on number of fail bits during verify procedure(s): 00_B **NOFAIL**, No fail bit. 01_B **ONEFAIL**, One fail bit in one data block. 10_B **TWOFAIL**, Two fail bits in two different data blocks. 11_B **MOREFAIL**, Two or more fail bits in one data block, or three or more fail bits overall.
SLEEP	1	r	**Sleep Mode** 0_B **READY**, NVM not in sleep mode, and no sleep or wake up procedure in progress. 1_B **SLEEP**, NVM in sleep mode, or busy due to a sleep or wake up procedure.
BUSY	0	r	**Busy** 0_B **READY**, The NVM is not busy. Memory reads from the cell array and register write accesses are possible. 1_B **BUSY**, The NVM is busy. Memory reads and register write accesses are not possible.

1) If a block is read from the field that contains two or more parity errors ECC1 and ECC2 errors may be flagged simultaneously.

NVM Programming Control Register

NVMPROG
NVM Programming Control Register (0004_H) **Reset Value: 0000_H**



Field	Bits	Type	Description
0	15:14	r	**Reserved** Read as 0; should by written with 0.
RSTECC	13	rw	**Reset ECC** Can only be set by software, is reset automatically by hardware. 0_B **NOP**, No action. 1_B **RESET**, Reset of **NVMSTATUS**.ECCxREAD and NVMSTATUS.WRPERR.
RSTVERR	12	rw	**Reset Verify Error** Can only be set by software, is reset automatically by hardware. 0_B **NOP**, No action. 1_B **RESET**, Reset of **NVMSTATUS**.VERR.
0	11:8	r	**Reserved** Read as 0; should be written with 0.

Field	Bits	Type	Description
ACTION	7:0	rw	**ACTION: [VERIFY, ONE_SHOT, OPTYPE]** This field selects an erase, write, or verify operation. See also **More details on ACTION**. ACTION is a concatenation of three bit fields: ACTION[7:6] = VERIFY, ACTION[5:4] = ONE_SHOT and ACTION[3:0] = OPTYPE. **OPTYPE** defines the following operations: 0000_B: idle or verify-only, that depends on the setting of VERIFY; 0001_B: write; 0010_B: page erase. **ONE_SHOT** is a parameter of OPTYPE with the following values: 01_B: once or 10_B: continuously. In case of 01_B, ACTION is automatically reset to idle mode after operation has been performed. **VERIFY** defines a second parameter of OPTYPE: 01_B: verification of written data with hardread levels after every write operation, 10_B: no verification, 11_B: verification of array content. The following operations are defined, other values are interpreted as 00_H 00_H , Idle state, no action triggered. Writing 00_H exits current mode. 51_H , Start one-shot write operation with automatic verify. 91_H , Start one-shot write operation without verify. 61_H , Start continuous write operation with automatic verify of every write. $A1_H$, Start continuous write operation without verify. 92_H , Start one-shot page erase operation. $A2_H$, Start continuous page erase operation. $D0_H$, Start one-shot verify-only: Written data is compared to array content. $E0_H$, Start continuous verify-only: Written data is compared to array content.

More details on ACTION

The operation selected by ACTION is performed, when a write to the NVM address range is performed, which defines the address (and block data) for the operation. Afterwards, in case of a one-shot operation, ACTION is automatically reset.

Verification results can be read at **NVMSTATUS**.VERR.

ACTION can only be changed when the current value of ACTION is 00_H, otherwise ACTION is set to 00_H. Once ACTION is not idle, ACTION can only be written again with its current value; any other value leads to a reset of ACTION.

Only write operations can be automatically verified. Page erase operations cannot use automatic verify, they must be started with ACTION.VERIFY = 10_B.

If the correct erasing of a block or page is to be verified, a separate sequence using ACTION.VERIFY = 11_B has to be started.

A verify operation requires the provision of the data for a complete block and always includes the ECC bits. The ECC bits for every block are generated automatically when the data is written.

A verify operation started with ACTION.VERIFY = 01_B is automatically performed with both hardread written and hardread erased levels.
A verify operation started with ACTION.VERIFY = 11_B is performed with the single hardread level defined by **NVMCONF**.HRLEV at this starting time.

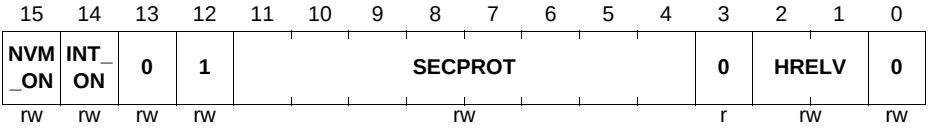
A sequence started by setting ACTION will set **NVMSTATUS**.BUSY only after the transfer of the address (and block data) is performed. The end of the sequence can be detected either by polling NVMSTATUS.BUSY or by waiting for an NVM interrupt.

Entering sleep mode resets ACTION.

NVM Configuration Register

NVMCONF is the only SFR that can be written while the module is in sleep mode. This is necessary to enable the use of NVM_ON = 0_B to go to sleep mode and of NVM_ON = 1_B to wake-up again.

NVMCONF
NVM Configuration Register **(0008_H)** **Reset Value: 9000_H**



Field	Bits	Type	Description
NVM_ON	15	rw	**NVM On** When cleared, no software code can be executed anymore from the NVM, until it is set again. I.e., already the software code that initiates the change in NVM_ON itself may not reside in the NVM, otherwise the software is stalled forever. 0_B **SLEEP**, NVM is switched to or stays in sleep mode. 1_B **NORM**, NVM is switched to or stays in normal mode.
INT_ON	14	rw	**Interrupt On** When enabled the completion of a sequence started by setting **NVMPROG**.ACTION (write, erase or verify-only sequence) will be indicated by NVM interrupt. The same is true for the wake-up sequence. 0_B **INTOFF**, No NVM ready interrupts are generated. 1_B **INTON**, NVM ready interrupts are generated.
0	13	rw	**Reserved for Future Use** Must be written with 0 to allow correct operation.
1	12	rw	**Reserved for Future Use** Must be written with 1 to allow correct operation.
SECPROT	11:4	rw	**Sector Protection**[1)] This field defines the number of write, erase, verify protected sectors, starting with physical sector 0.
0	3	r	**Reserved** Read as 0; should be written with 0.

Field	Bits	Type	Description
HRLEV	2:1	rw	**Hardread Level**[2)] Defines single hardread level for verification with NVMPROG.ACTION.VERIFY = 11_B: 00_B **NR**, Normal read 01_B **HRW**, Hardread written 10_B **HRE**, Hardread erased 11_B **RFU**, Reserved for Future Use
0	0	rw	**Reserved for Future Use** Must be written with 0 to allow correct operation.

1) For SECPROT > 0, SECPROT defines the number of protected sectors. The sectors 0 to SECPROT-1 cannot be written, erased, or verified. All writes that target the protected sectors are accepted, but are internally ignored.

2) HRLEV defines the hardread level for a stand-alone verification sequence started with NVMPROG.ACTION.VERIFY = 11_B. This hardread level is used until the end of the verification sequence. HRLEV may not be changed in between.

8.8 Example Sequences

This section presents some low-level programming examples.

In all following operations the protection defined by **NVMCONF**.SECPROT needs to be taken into account.

8.8.1 Writing to Memory

8.8.1.1 Writing a Single Block

This sequence requires that the target block is already erased.

Additional assumption: **NVMPROG**.ACTION = 00_H.

1. Start a one-shot write operation:
   Write **NVMPROG**.ACTION = 51_H or 91_H, respectively, if an automatic verification of the written data is to be performed or not.
2. Write data for one block to the physical address.
3. Poll flag **NVMSTATUS**.BUSY until write sequence has finished, or wait for NVMready interrupt (if enabled).
4. If an automatic verification of the written data was requested in the first step, read **NVMSTATUS**.VERR for the verification result.

If no verification of the written data was requested in step 1, and thus no read access to the NVM SFR is required in step 4, step 3 may be omitted.

The next access to the NVM module or the next write access to an NVM SFR (which ever comes first) will be automatically stalled, until the operation started in step 2 is finished.

8.8.1.2 Writing Blocks

This sequence requires the target blocks are already erased.

Additional assumption: **NVMPROG**.ACTION = 00_H.

1. Start a continuous write operation:
   Write **NVMPROG**.ACTION = 61_H or $A1_H$, respectively, if an automatic verification of the written data is to be performed or not.
2. Write data for one block to the physical address.
3. Poll flag **NVMSTATUS**.BUSY until write sequence has finished, or wait for NVMready interrupt (if enabled).
   Optionally, step 3 may be omitted: The next access to the NVM module or the next write access to an NVM SFR (which ever comes first) will be automatically stalled, until the operation started in step 2 is finished.
4. Jump to step 2 unless all data is written.

5. Stop continuous write operation:
   Write **NVMPROG**.ACTION = 00_H.
6. If an automatic verification of the written data was requested in the first step, read **NVMSTATUS**.VERR for the verification result.

8.8.2 Erasing Memory

8.8.2.1 Erasing a Single Page

Assumption: **NVMPROG**.ACTION = 00_H.

1. Start a one-shot page erase operation:
   Write **NVMPROG**.ACTION = 92_H.
2. Write dummy data for one word to one arbitrary physical address of the page to be erased.
3. Poll flag **NVMSTATUS**.BUSY until page erase sequence has finished, or wait for NVMready interrupt (if enabled).

Optionally, step 3 may be omitted: The next access to the NVM module or the next write access to an NVM SFR (which ever comes first) will be automatically stalled, until the operation started in step 2 is finished.

8.8.2.2 Erasing Pages

Assumption: **NVMPROG**.ACTION = 00_H.

1. Start a continuous page erase operation:
   Write **NVMPROG**.ACTION = $A2_H$.
2. Write dummy data for one word to one arbitrary physical address of the page to be erased.
3. Poll flag **NVMSTATUS**.BUSY until page erase sequence has finished, or wait for NVMready interrupt (if enabled).
   Optionally, step 3 may be omitted: The next access to the NVM module or the next write access to an NVM SFR (which ever comes first) will be automatically stalled, until the operation started in step 2 is finished.
4. Jump to step 2 unless all pages are erased.
5. Stop continuous page erase operation:
   Write **NVMPROG**.ACTION = 00_H.

8.8.3 Verifying Memory

8.8.3.1 Verifying a Single Block

Assumption: **NVMPROG**.ACTION = 00_H.

1. Choose desired hardread level by setting NVMCONF.HRLEV.
2. Start a one-shot verify operation:
   Write **NVMPROG**.ACTION = $D0_H$.
3. Write reference data for one block to the physical address.
4. Poll flag **NVMSTATUS**.BUSY until verify sequence has finished, or wait for NVMready interrupt (if enabled).
5. Read **NVMSTATUS**.VERR for the verification result.

8.8.3.2 Verifying Blocks

Assumption: **NVMPROG**.ACTION = 00_H.

1. Choose desired hardread level by setting NVMCONF.HRLEV.
2. Start a continuous verify operation:
   Write **NVMPROG**.ACTION = $E0_H$.
3. Write reference data for one block to the physical address.
4. Poll flag **NVMSTATUS**.BUSY until verify sequence has finished, or wait for NVMready interrupt (if enabled).
   Optionally, step 4 may be omitted: The next access to the NVM module or the next write access to an NVM SFR (which ever comes first) will be automatically stalled, until the operation started in step 3 is finished.
5. Jump to step 3 unless all blocks are verified.
6. Stop continuous verify operation:
   Write **NVMPROG**.ACTION = 00_H.
7. Read **NVMSTATUS**.VERR for the verification result.

8.8.4 Writing to an Already Written Block

Normally, writing additional bits in an already written block is not feasible, since the already written ECC bits and the parity bits would need an update, too, which in most cases would require to erase some bits, which is not possible. The result would be an ECC-faulty block.

For special purposes a repeated update of specially constructed data is possible, e.g. to store some marker bits for use in a power loss recovery SW implementation.

Starting with an erased block, and making use of the special structure of the ECC, it is possible to repeatedly add two newly written bits in identical bit positions to two arbitrary words of the block. This principle is shown in the exemplary writing scheme of **Table 8-4**, where a block is updated 32 times without corruption of the ECC, i.e. the data stays ECC protected throughout the procedure.

Table 8-4 Incremental Update of a Block with Specially Constructed Data

Operation	Block Write Data	Resulting Block Content[1)]
Erase block		FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
Add 1st value	FFFFFFFF FFFFFFFC FFFFFFFF FFFFFFFC	FFFFFFFF FFFFFFFC FFFFFFFF FFFFFFFC
Add 2nd value	FFFFFFFF FFFFFFF3 FFFFFFFF FFFFFFF3	FFFFFFFF FFFFFFF0 FFFFFFFF FFFFFFF0
Add 3rd value	FFFFFFFF FFFFFFCF FFFFFFFF FFFFFFCF	FFFFFFFF FFFFFFC0 FFFFFFFF FFFFFFC0
Add 4th value	FFFFFFFF FFFFFF3F FFFFFFFF FFFFFF3F	FFFFFFFF FFFFFF00 FFFFFFFF FFFFFF00
Add 5th value	FFFFFFFF FFFFFCFF FFFFFFFF FFFFFCFF	FFFFFFFF FFFFFC00 FFFFFFFF FFFFFC00
Add 6th value	FFFFFFFF FFFFF3FF FFFFFFFF FFFFF3FF	FFFFFFFF FFFFF000 FFFFFFFF FFFFF000
...	...	...
15th value	FFFFFFFF CFFFFFFF FFFFFFFF CFFFFFFF	FFFFFFFF C0000000 FFFFFFFF C0000000
16th value	FFFFFFFF 3FFFFFFF FFFFFFFF 3FFFFFFF	FFFFFFFF 00000000 FFFFFFFF 00000000
17th value	FFFFFFFC FFFFFFFF FFFFFFFC FFFFFFFF	FFFFFFFC 00000000 FFFFFFFC 00000000
...	...	...
30th value	F3FFFFFF FFFFFFFF F3FFFFFF FFFFFFFF	F0000000 00000000 F0000000 00000000
31st value	CFFFFFFF FFFFFFFF CFFFFFFF FFFFFFFF	C0000000 00000000 C0000000 00000000
32nd value	3FFFFFFF FFFFFFFF 3FFFFFFF FFFFFFFF	00000000 00000000 00000000 00000000
Add to invalidate written values[2)]	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE	Same as before, but ECC2-faulty

1) ECC-clean and parity-clean (ECC bits and parity bits all stay erased), except where indicated otherwise.
2) This write data can be written on any of the states above (except the completely erased state), always resulting in an unchanged data content, but now with an ECC2 error (three ECC bits and one parity bit are written).

Other combinations of write values are possible, too, as long as the basic “add two identical bits in each of two words” is adhered to. In **Table 8-4** it is also shown that an invalidation of the data by deliberately creating an ECC2 error is possible.

To minimize the write times and also to minimize the cycle load of the block, it is important to only write the new bits of the block as shown in **Table 8-4**. In principle it is possible to repeatedly write also the already written bits again, but this would unnecessarily increase the writing times and would also increase the cycle load.

8.8.5 Sleep Mode

Assumption: Active mode (**NVMSTATUS**.SLEEP = 0_B) and **NVMSTATUS**.BUSY = 0_B.

To Enter Sleep Mode

1. Execute WFE/WFI or **NVMCONF**.NVM_ON = 0_B.
2. NVMSTATUS.BUSY = 1_B and NVMSTATUS.SLEEP = 1_B until sleep mode is reached.
3. NVMSTATUS.BUSY = 0_B and NVMSTATUS.SLEEP = 1_B while in sleep mode.

To Wake-up from Sleep Mode

1. Any wake-up event and NVMCONF.NVM_ON = 1_B.
2. NVMSTATUS.BUSY = 1_B and NVMSTATUS.SLEEP = 1_B until active mode is reached.
3. NVMSTATUS.BUSY = 0_B and NVMSTATUS.SLEEP = 0_B while in active mode.

A wake-up event while the goto sleep sequence has not finished yet, does not shorten this sequence, but only directly starts the wake-up sequence afterwards.

8.8.6 Timing

This state diagram shows the transitions and timings of all possible sequences.

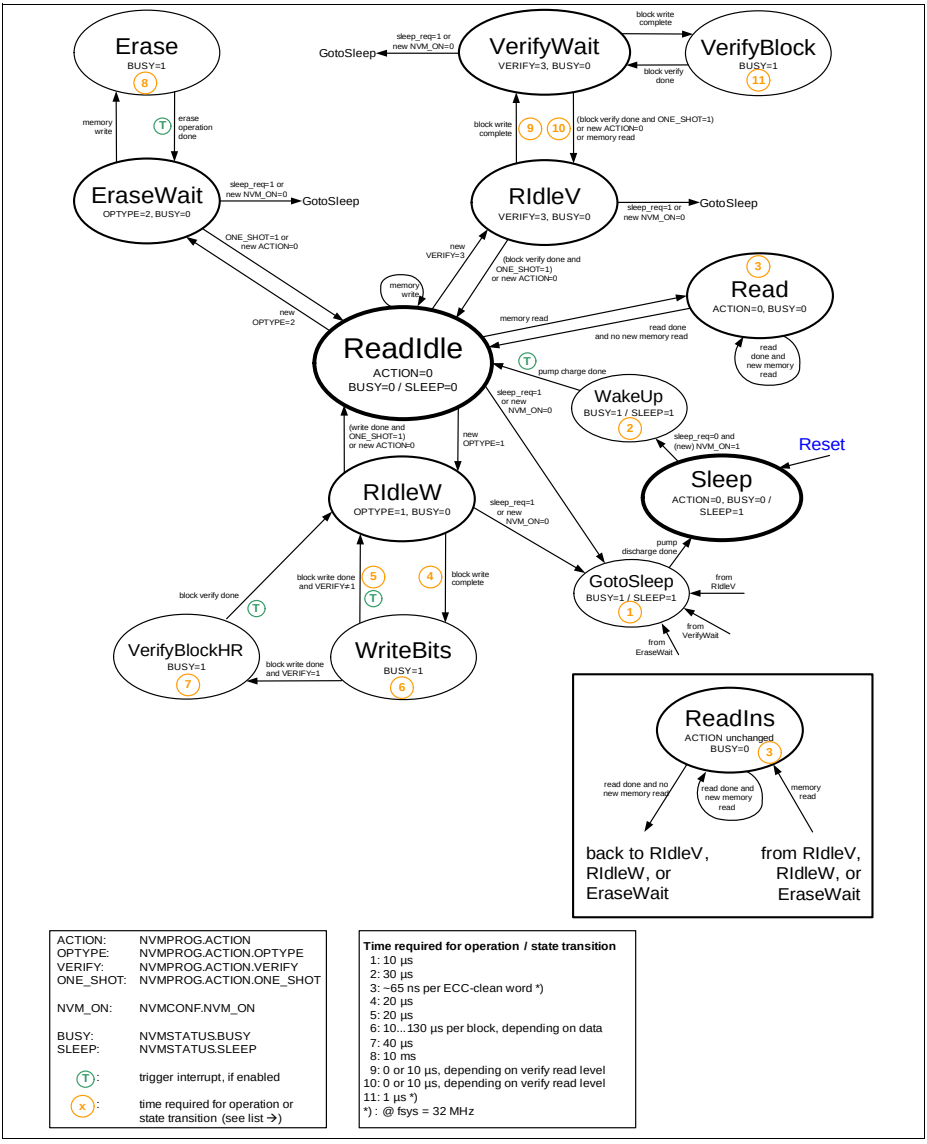


Figure 8-4 State Diagram of the NVM Module Timings are preliminary

9 Peripheral Access Unit (PAU)

The Peripheral Access Unit (PAU) supports access control of memories and peripherals in a central place.

9.1 Features

The PAU provides the following features:

- Allows user application to enable/disable the access to the registers of a peripheral
- Generates a HardFault exception when there is an access to a disabled or unassigned address location
- Provides information on availability of peripherals and size of memories

9.2 Peripheral Privilege Access Control

The user application can use the Peripheral Privilege Access Registers, PRIVDISn, to disable access to a peripheral. When the PDISx bit corresponding to the peripheral is set, the memory address space mapped to the peripheral is rendered invalid. An access to such an invalid address causes a HardFault exception.

The application can clear the same bit to enable accesses to the peripheral again.

Table 9-1 shows the peripherals and their assigned PDISx bits. Peripherals without a PDISx bit are accessible at all times.

Table 9-1 Peripherals Availability and Privilege Access Control

Peripheral	Address Grouping	AVAILn.AVAILx bit	PRIVDIS.PDISx bit
Flash	Flash SFRs	-	PRIVDIS0.2
SRAM	RAM Block 1	AVAIL0.5	PRIVDIS0.5
	RAM Block 2	AVAIL0.6	PRIVDIS0.6
	RAM Block 3	AVAIL0.7	PRIVDIS0.7
WDT	WDT	-	PRIVDIS0.19
Ports	Port 0	AVAIL0.22	PRIVDIS0.22
	Port 1	AVAIL0.23	PRIVDIS0.23
	Port 2	AVAIL0.24	PRIVDIS0.24
USIC0	USIC0_CH0	AVAIL1.0	PRIVDIS1.0
	USIC0_CH1	AVAIL1.1	PRIVDIS1.1
PRNG	PRNG	AVAIL1.4	-
VADC0	VADC0 Basic SFRs	AVAIL1.5	PRIVDIS1.5
SHS0	SHS0	AVAIL1.8	PRIVDIS1.8

Table 9-1 Peripherals Availability and Privilege Access Control

Peripheral	Address Grouping	AVAILn.AVAILx bit	PRIVDIS.PDISx bit
CCU4	CC40 and CCU40 Kernel SFRs	AVAIL1.9	PRIVDIS1.9
	CC41	AVAIL1.10	PRIVDIS1.10
	CC42	AVAIL1.11	PRIVDIS1.11
	CC43	AVAIL1.12	PRIVDIS1.12

9.3 Peripheral Availability and Memory Size

The availability of peripherals and memory sizes varies according to product variants. The user application can read the Peripheral Availability Registers, AVAILn, to check for the availability of peripherals in a product variant. Refer to **Table 9-1** for the bit assignement. Similarly, the Memory Size Registers (e.g. FLSIZE for Flash memory) can be used to check for the size of the available memories.

9.4 PAU Registers

Registers Overview

The absolute register address is calculated by adding:

Module Base Address + Offset Address

Table 9-2 Registers Address Space

Module	Base Address	End Address	Note
PAU	4000 0000$_H$	4000 FFFF$_H$	

Table 9-3 Register Overview

Short Name	Description	Offset Addr.	Access Mode		Description See
			Read	Write	
Reserved	Reserved	0000$_H$ - 003C$_H$	nBE	nBE	
AVAIL0	Peripheral Availability Register 0	0040$_H$	U, PV	BE	Page 9-6
AVAIL1	Peripheral Availability Register 1	0044$_H$	U, PV	BE	Page 9-7
AVAIL2	Peripheral Availability Register 2	0048$_H$	U, PV	BE	Page 9-9
Reserved	Reserved	004C$_H$ - 007C$_H$	nBE	nBE	
PRIVDIS0	Peripheral Privilege Access Register 0	0080$_H$	U, PV	U, PV	Page 9-4
PRIVDIS1	Peripheral Privilege Access Register 1	0084$_H$	U, PV	U, PV	Page 9-5
Reserved	Reserved	0088$_H$ - 03FC$_H$	nBE	nBE	
ROMSIZE	ROM Size Register	0400$_H$	U, PV	BE	Page 9-9
FLSIZE	Flash Size Register	0404$_H$	U, PV	BE	Page 9-10
RAM0SIZE	RAM0 Size Register	0410$_H$	U, PV	BE	Page 9-11

9.4.1 Peripheral Privilege Access Registers (PRIVDISn)

The PRIVDISn registers provide the bit to enable and disable access to a peripheral during runtime. When disabled, an access to the peripheral throws a BusFault.

PRIVDIS0
Peripheral Privilege Access Register 0 (0080_H) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0							PDIS24	PDIS23	PDIS22	0		PDIS19	0		
r							rw	rw	rw	r		rw	r		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								PDIS7	PDIS6	PDIS5	0		PDIS2	0	
r								rw	rw	rw	r		rw	r	

Field	Bits	Type	Description
PDIS2	2	rw	**Flash SFRs Privilege Disable Flag** 0_B Flash SFRs are accessible. 1_B Flash SFRs are not accessible.
PDIS5	5	rw	**RAM Block 1 Privilege Disable Flag** 0_B RAM Block 1 is accessible. 1_B RAM Block 1 is not accessible.
PDIS6	6	rw	**RAM Block 2 Privilege Disable Flag** 0_B RAM Block 2 is accessible. 1_B RAM Block 2 is not accessible.
PDIS7	7	rw	**RAM Block 3 Privilege Disable Flag** 0_B RAM Block 3 is accessible. 1_B RAM Block 3 is not accessible.
PDIS19	19	rw	**WDT Privilege Disable Flag** 0_B WDT is accessible. 1_B WDT is not accessible.
PDIS22	22	rw	**Port 0 Privilege Disable Flag** 0_B Port 0 is accessible. 1_B Port 0 is not accessible.
PDIS23	23	rw	**Port 1 Privilege Disable Flag** 0_B Port 1 is accessible. 1_B Port 1 is not accessible.

Field	Bits	Type	Description
PDIS24	24	rw	**Port 2 Privilege Disable Flag** 0_B Port 2 is accessible. 1_B Port 2 is not accessible.
0	[31:25], [21:20], [18:8], [4:3], [1:0]	r	**Reserved**

PRIVDIS1

Peripheral Privilege Access Register 1 (0084_H) **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			**PDIS 12**	**PDIS 11**	**PDIS 10**	**PDIS 9**	**PDIS 8**	**0**		**PDIS 5**	**0**			**PDIS 1**	**PDIS 0**
r			rw	rw	rw	rw	rw	r		rw	r			rw	rw

Field	Bits	Type	Description
PDIS0	0	rw	**USIC0 Channel 0 Privilege Disable Flag** 0_B USIC0 Channel 0 is accessible. 1_B USIC0 Channel 0 is not accessible.
PDIS1	1	rw	**USIC0 Channel 1 Privilege Disable Flag** 0_B USIC0 Channel 1 is accessible. 1_B USIC0 Channel 1 is not accessible.
PDIS5	5	rw	**VADC0 Basic SFRs Privilege Disable Flag** 0_B VADC0 Basic SFRs are accessible. 1_B VADC0 Basic SFRs are not accessible.
PDIS8	8	rw	**SHS0 Privilege Disable Flag** 0_B SHS0 is accessible. 1_B SHS0 is not accessible.

Field	Bits	Type	Description
PDIS9	9	rw	**CC40 and CCU40 Kernel SFRs Privilege Disable Flag** 0_B CC40 and CCU40 Kernel SFRs are accessible. 1_B CC40 and CCU40 Kernel SFRs are not accessible.
PDIS10	10	rw	**CC41 Privilege Disable Flag** 0_B CC41 is accessible. 1_B CC41 is not accessible.
PDIS11	11	rw	**CC42 Privilege Disable Flag** 0_B CC42 is accessible. 1_B CC42 is not accessible.
PDIS12	12	rw	**CC43 Privilege Disable Flag** 0_B CC43 is accessible. 1_B CC43 is not accessible.
0	[31:13], [7:6], [4:2]	r	**Reserved**

9.4.2 Peripheral Availability Registers (AVAILn)

The AVAILn registers indicate the available peripherals for the particular device variant.

Note: The reset values of AVAILn registers show the configuration with all peripherals available. Actual values might differ depending on the product variant.

AVAIL0

Peripheral Availability Register 0 (0040_H) Reset Value: 01CF 00FF$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0							AVAIL24	AVAIL23	AVAIL22	0		1			
r							r	r	r	r		r			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								AVAIL7	AVAIL6	AVAIL5	1				
r								r	r	r			r		

Field	Bits	Type	Description
AVAIL5	5	r	**RAM Block 1 Availability Flag** 0_B RAM block 1 is not available. 1_B RAM block 1 is available.
AVAIL6	6	r	**RAM Block 2 Availability Flag** 0_B RAM block 2 is not available. 1_B RAM block 2 is available.
AVAIL7	7	r	**RAM Block 3 Availability Flag** 0_B RAM block 3 is not available. 1_B RAM block 3 is available.
AVAIL22	22	r	**Port 0 Availability Flag** 0_B Port 0 is not available. 1_B Port 0 is available.
AVAIL23	23	r	**Port 1 Availability Flag** 0_B Port 1 is not available. 1_B Port 1 is available.
AVAIL24	24	r	**Port 2 Availability Flag** 0_B Port 2 is not available. 1_B Port 2 is available.
1	[19:16], [4:0]	r	**Reserved**
0	[31:25], [21:20], [15:8]	r	**Reserved**

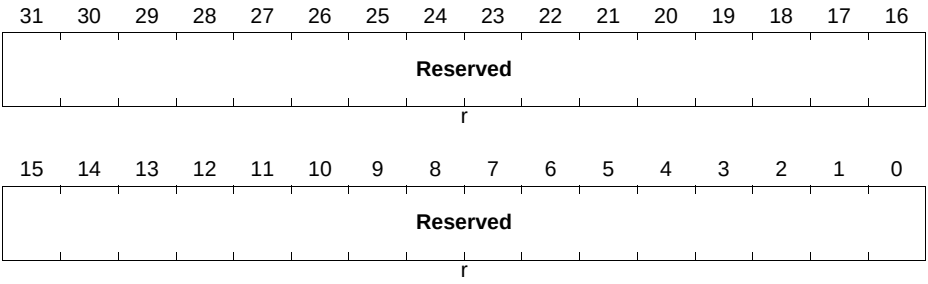
AVAIL1
Peripheral Availability Register 1 (0044_H) **Reset Value: 0000 1F37$_H$**

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			AVAIL12	AVAIL11	AVAIL10	AVAIL9	AVAIL8	0		AVAIL5	AVAIL4	0	1	AVAIL1	AVAIL0
r			r	r	r	r	r	r		r	r	r	r	r	r

Field	Bits	Type	Description
AVAIL0	0	r	**USIC0 Channel 0 Availability Flag** 0_B USIC0 Channel 0 is not available. 1_B USIC0 Channel 0 is available.
AVAIL1	1	r	**USIC0 Channel 1 Availability Flag** 0_B USIC0 Channel 1 is not available. 1_B USIC0 Channel 1 is available.
AVAIL4	4	r	**PRNG Availability Flag** 0_B PRNG is not available. 1_B PRNG is available.
AVAIL5	5	r	**VADC0 Basic SFRs Availability Flag** 0_B VADC0 Basic SFRs are not available. 1_B VADC0 Basic SFRs are available.
AVAIL8	8	r	**SHS0 Availability Flag** 0_B SHS0 is not available. 1_B SHS0 is available.
AVAIL9	9	r	**CC40 Availability Flag** 0_B CC40 is not available. 1_B CC40 is available.
AVAIL10	10	r	**CC41 Availability Flag** 0_B CC41 is not available. 1_B CC41 is available.
AVAIL11	11	r	**CC42 Availability Flag** 0_B CC42 is not available. 1_B CC42 is available.
AVAIL12	12	r	**CC43 Availability Flag** 0_B CC43 is not available. 1_B CC43 is available.
1	2	r	**Reserved**
0	[31:13], [7:6], 3	r	**Reserved**

AVAIL2
Peripheral Availability Register 2 (0048_H) **Reset Value: $0000\ 0000_H$**



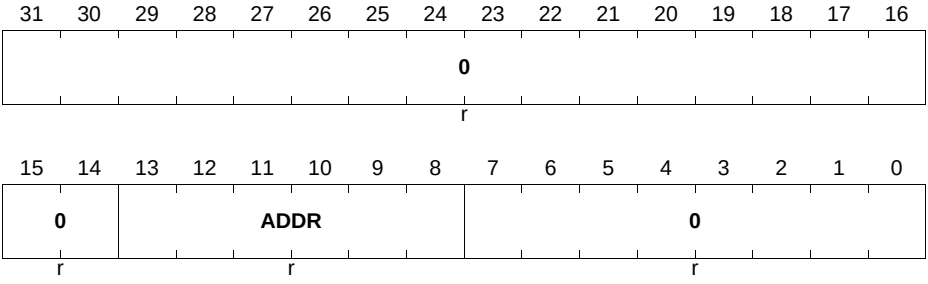
Field	Bits	Type	Description
Reserved	[31:0]	r	**Reserved** Read as 0.

9.4.3 Memory Size Registers

Memory size registers are available for the ROM, SRAM and Flash memories. They are used to indicate the available size of these memories in the device.

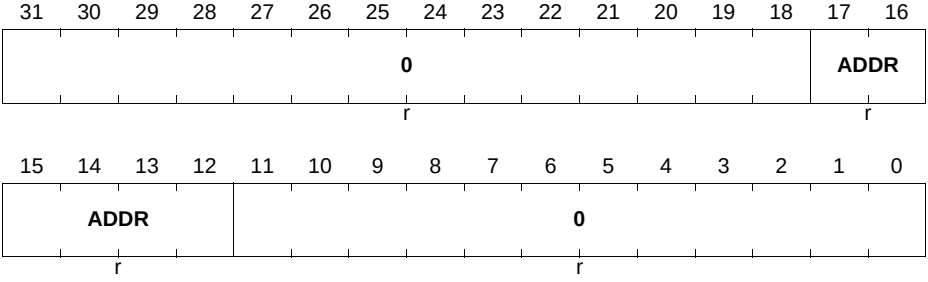
Note: The reset values of the size registers show the configuration of the superset device. Actual values might differ depending on the product variant.

ROMSIZE
ROM Size Register (0400_H) **Reset Value: $0000\ 0B00_H$**



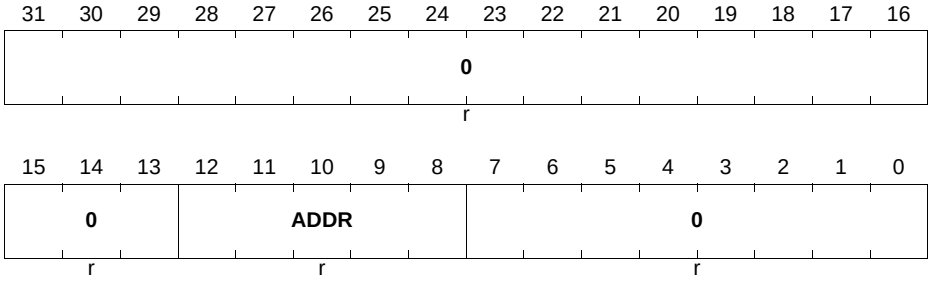
Field	Bits	Type	Description
ADDR	[13:8]	r	**ROM Size** Size of user-readable ROM in bytes = ADDR * 256
0	[31:14], [7:0]	r	**Reserved**

FLSIZE
Flash Size Register **(0404_H)** **Reset Value: $0001\ 1000_H$**



Field	Bits	Type	Description
ADDR	[17:12]	r	**Flash Size** Size of the Flash (excluding Flash sector 0) in Kbytes = (ADDR - 1) * 4
0	[31:18], [11:0]	r	**Reserved**

RAM0SIZE
RAM0 Size Register **(0410_H)** **Reset Value: $0000\ 1000_H$**



Field	Bits	Type	Description
ADDR	[12:8]	r	**RAM0 Size** Size of RAM block 0 in bytes = ADDR * 256 For total RAM size, RAM blocks 1 to 3 have to be also taken into consideration.
0	[31:13], [7:0]	r	**Reserved**

On-chip Peripherals

10 Window Watchdog Timer (WDT)

Purpose of the Window Watchdog Timer module is improvement of system integrity. WDT triggers the system reset or other corrective action like e.g. an interrupt if the main program, due to some fault condition, neglects to regularly service the watchdog (also referred to as “kicking the dog”, “petting the dog”, “feeding the watchdog” or “waking the watchdog”). The intention is to bring the system back from the unresponsive state into normal operation.

References

[6] Cortex-M0 User Guide, ARM DUI 0467B (ID081709)

10.1 Overview

A successful servicing of the WDT results in a pulse on the signal wdt_service. The signal is offered also as an alternate function output can be used to show to an external watchdog that the system is alive.

The WDT timer is a 32-bit counter, which counts up from 0_H. It can be serviced while the counter value is within the window boundary, i.e. between the lower and the upper boundary value. Correct servicing results in a reset of the counter to 0_H. A so called “Bad Service” attempt results in the system reset request.

The timer block is running on the f_{WDT} clock which is independent from the bus clock. The timer value is updated in the corresponding AHB register **TIM**. This mechanism enables immediate response on a read access from the bus.

10.1.1 Features

The watchdog timer (WDT) is an independent window watchdog timer.

The features are:

- Triggers system reset when not serviced on time or serviced in a wrong way
- Servicing restricted to be within boundaries of refresh window
- Can run from an independent clock
- Provides service indication to an external pin
- Can be suspended in halting mode
- Provides optional pre-warning alarm before reset

Table 10-1 Application Features

Feature	Purpose/Application
System reset upon Bad Servicing	Triggered to restore system stable operation and ensure system integrity
Servicing restricted to be within defined boundaries of refresh window	Allows to consider minimum and maximum software timing
Independent clocks	To ensure that WDT counts even in case of the system clock failure
Service indication on external pin	For dual-channel watchdog solution, additional external control of system integrity
Suspending in HALT mode	Enables safe debugging with productive code
Pre-warning alarm	Software recovery to allow corrective action via software recovery routine bringing system back from the unresponsive state into normal operation

10.1.2 Block Diagram

The WDT block diagram is shown in **Figure 10-1**.

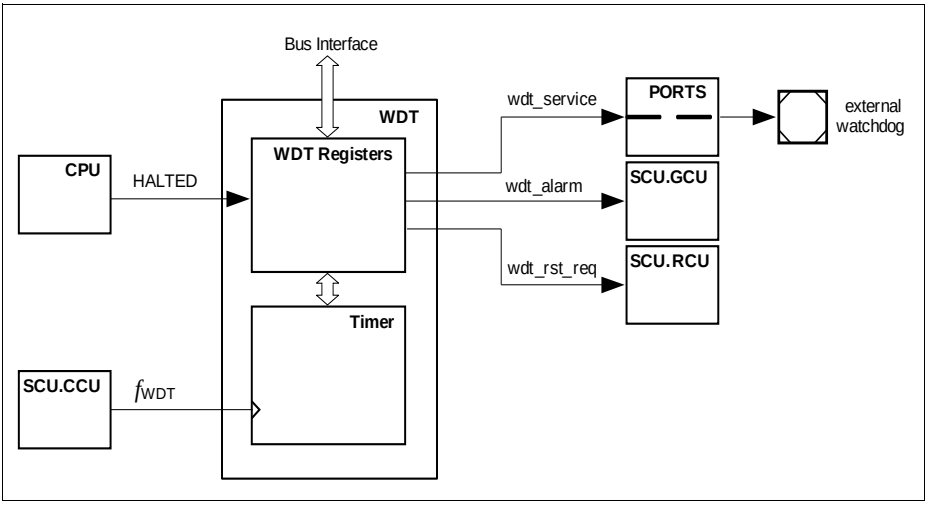


Figure 10-1 Watchdog Timer Block Diagram

10.2 Time-Out Mode

An overflow results in an immediate reset request going to the RCU of the SCU via the signal wdt_rst_req whenever the counter crosses the upper boundary it triggers an overflow event pre-warning is not enabled with **CTR** register. A successful servicing performed with writing a unique value, referred to as “Magic Word” to the **SRV** register of the WDT within the valid servicing window, results in a pulse on the signal wdt_service and reset of the timer counter.

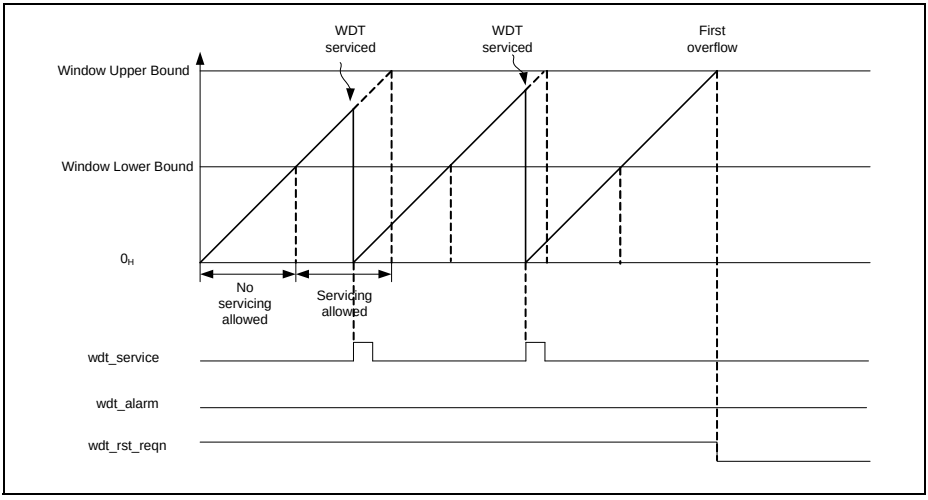


Figure 10-2 Reset without pre-warning

The example scenario depicted in **Figure 10-2** shows two consecutive service pulses generated from WDT module as the result of successful servicing within valid time windows. The situation where no service has been performed immediately triggers generation of reset request on the wdt_rst_req output after the counter value has exceeded window upper bound value.

10.3 Pre-warning Mode

While in pre-warning mode the effect of the overflow event is different with and without pre-warning enabled. The first crossing of the upper bound triggers the outgoing alarm signal wdt_alarm when pre-warning is enabled. Only the next overflow results a reset request. The alarm status is shown via register **WDTSTS** and can be cleared via register **WDTCLR**. A clear of the alarm status will bring the WDT back to normal state.

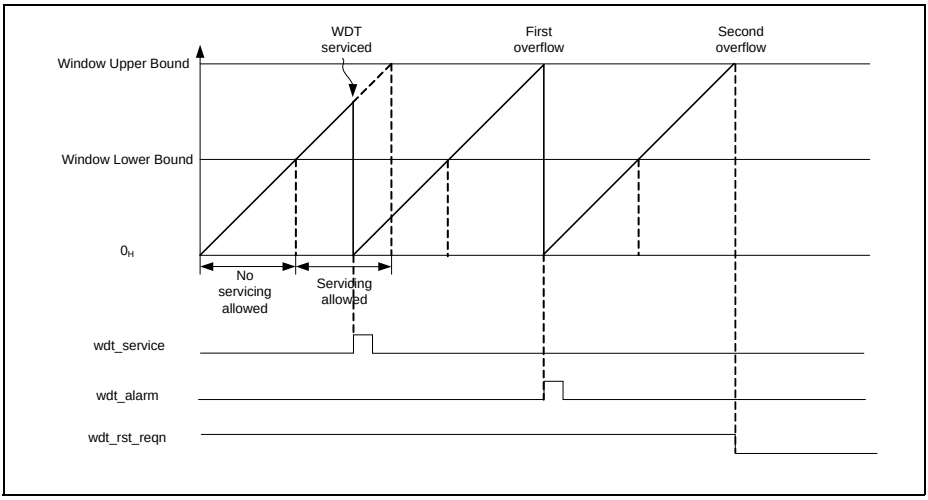


Figure 10-3 Reset after pre-warning

The example scenario depicted in **Figure 10-3** shows service pulse generated from WDT module as the result of successful servicing within valid time window. WDT generates alarm pulse on wdt_alarm upon first missing servicing. The alarm signal is routed as interrupt request to the SCU. Within this alarm service request the user can clear the WDT status bit and give a proper WDT service before it overflows next time. Otherwise WDT generates reset request on wdt_rstn upon the second missing service.

10.4 Bad Service Operation

A bad service attempt results in a reset request. A bad service attempt can be due to servicing outside the window boundaries or servicing with an invalid Magic Word.

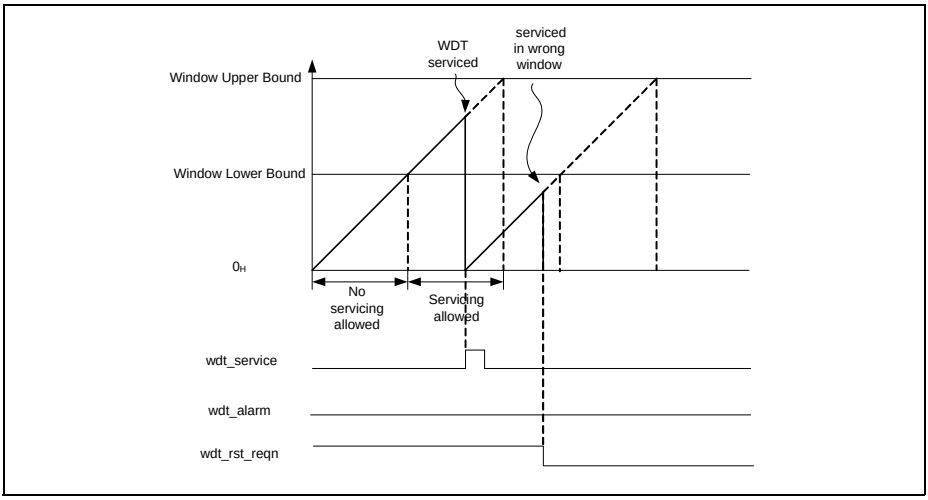


Figure 10-4 Reset upon servicing in a wrong window

The example in **Figure 10-4** shows servicing performed outside of valid servicing window. Attempt to service WDT while counter value remains below the window lower bound results in immediate reset request on wdt_rst_req signal.

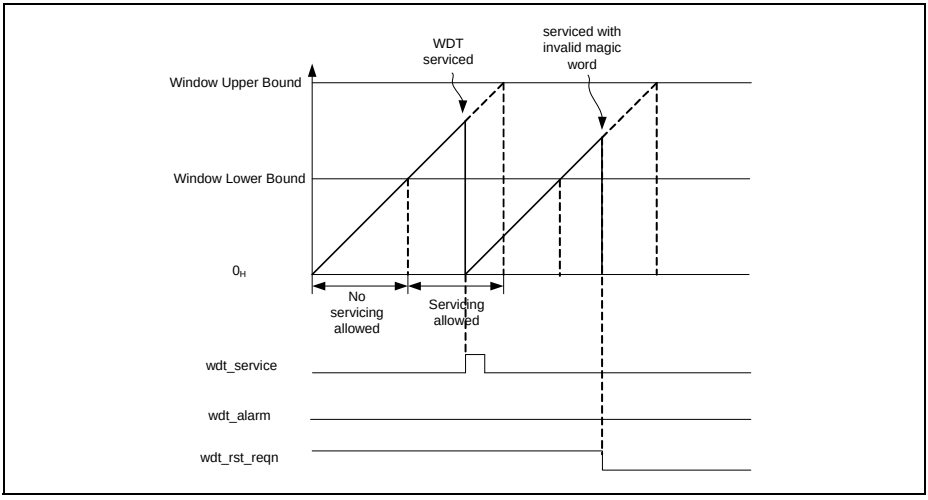


Figure 10-5 Reset upon servicing with a wrong magic word

The example in **Figure 10-5** shows servicing performed within a valid servicing window but with an invalid Magic Word. Attempt to write a wrong word to the **SRV** register results in immediate reset request on wdt_rst_req signal.

10.5 Service Request Processing

The WDT generates watchdog alarm service requests via wdt_alarm output signal upon first counter overflow over watchdog upper bound when pre-warning mode is enabled. The alarm service request is serviced in SCU.

Service requests can be disabled respectively by service request mask register in SCU.

10.6 Debug Behavior

The WDT function can be suspended when the CPU enters HALT mode. WDT debug function is controlled with DSP bit field in **CTR** register and it is set to be suspended by default.

10.7 Power, Reset and Clock

The WDT module is a part of the core domain and supplied with V_{DDC} voltage.

All WDT registers get reset with the system reset.

A sticky bit in the Reset Status Register, RSTSTAT, of SCU/RCU module indicates whether the last system reset has been triggered by the WDT module. This bit does not get reset with system reset.

The input clock of the WDT counter is provided by the internal 32kHz standby clock from SCU/CCU module, independently from the AHB interface clock.

The WDT module clock is default disabled and can be enabled via the SCU_CGATCLR0 register. Enabling and disabling the module clock could cause load change and clock blanking could happen as explained in the CCU (Clock Gating Control) section of the SCU chapter. It is strongly recommended to setup the module clock in the user initialisation code to avoid clock blanking during runtime.

10.8 Initialization and Control Sequence

Programming model of the WDT module assumes several scenarios where different control sequences apply.

Note: Some of the scenarios described in this chapter require operations on system level the that are not in the scope of the WDT module description, therefore for detailed information please refer to relevant chapters of this document.

10.8.1 Initialization & Start of Operation

Complete WDT module initialization is required upon system reset.

- check reason for last system reset in order to determine power state
  - read out SCU_RSTSTAT.RSTSTAT register bit field to determine last system reset cause and clear this bit using bit SCU_RSTCLR.RSCLR
  - perform appropriate operations dependent on the last system reset cause
- WDT software initialization sequence
  - enable WDT clock with SCU_CGATCLR0.WDT register bit field
  - set lower window bound with WDT_WLB register
  - set upper window bound with WDT_WUB register
  - configure external watchdog service indication (optional, please refer to PORT chapter)
  - enable interrupt for pre-warning alarm on system level with SCU_SRMSK register (optional, used in WDT pre-warning mode only)
- software start sequence
  - select mode (Time-Out or Pre-warning) and enable WDT module with WDT_CTR register
- service the watchdog
  - write magic word to WDT_SRV register within valid time window

10.8.2 Software Stop & Resume Operation

The WDT module can be stopped and re-started at any point of time for e.g. debug purpose using software sequence.

- software stop sequence
  - disable WDT module with WDT_CTR register
- perform any user operations
- software start (resume) sequence
  - enable WDT module with WDT_CTR register with WDT_CTR register
- service the watchdog
  - write magic word to WDT_SRV register within valid time window

10.8.3 Enter Sleep/Deep-Sleep & Resume Operation

The WDT counter clock can be configured to stop while in sleep or deep-sleep mode. No direct software interaction with the WDT is required in those modes and no watchdog time-out will fire if the WDT clock is configured to stop while CPU is sleeping.

- software configuration sequence for sleep/deep-sleep mode
  - configure WDT behavior with SCU_CGATx register
- enter sleep/deep-sleep mode software sequence

  - select sleep or deep-sleep mode in CPU (for details please refer to Cortex-M0 documentation [6])
  - enter selected mode (for details please refer to Cortex-M0 documentation [6])
- wait for a wake-up event (no software interaction, CPU stopped)
- resume operation (CPU clock restarted automatically on an event)
- service the watchdog
  - write magic word to WDT_SRV register within valid time window

10.8.4 Pre-warning Alarm Handling

The WDT will fire pre-warning alarm before requesting system reset while in pre-warning mode and not serviced within valid time window. The WDT status register indicating alarm must be cleared before the timer counter value crosses the upper bound for the second time after firing the alarm. After clearing of the alarm status regular watchdog servicing must be performed within valid time window.

- alarm event
  - exception routine (service request) clearing WDT_WDTSTAT register with WDT_WDTCLR register
- service the watchdog
  - write magic word to WDT_SRV register within valid time window

10.9 WDT Registers

Registers Overview

The absolute register address is calculated by adding:

Module Base Address + Offset Address

Table 10-2 Registers Address Space

Module	Base Address	End Address	Note
WDT	$4002\ 0000_H$	$4002\ FFFF_H$	Watchdog Timer Registers

Table 10-3 Register Overview

Short Name	Description	Offset Addr.	Access Mode		Description See
			Read	Write	
WDT Kernel Registers					
ID	Module ID Register	00_H	U, PV	PV	Page 10-9
CTR	Control Register	04_H	U, PV	PV	Page 10-10
SRV	Service Register	08_H	BE	PV	Page 10-11
TIM	Timer Register	$0C_H$	U, PV	BE	Page 10-13
WLB	Window Lower Bound	10_H	U, PV	PV	Page 10-13
WUB	Window Upper Bound	14_H	U, PV	PV	Page 10-14
WDTSTS	Watchdog Status Register	18_H	U, PV	PV	Page 10-14
WDTCLR	Watchdog Status Clear Register	$1C_H$	U, PV	PV	Page 10-15

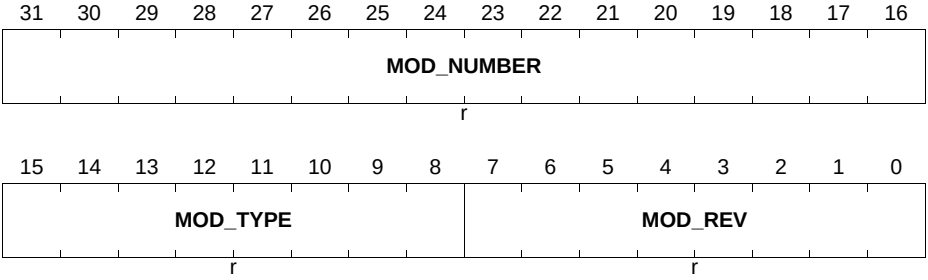
10.9.1 Registers Description

ID

The module unique ID register.

ID
WDT Module ID Register

(0000_H) Reset Value: $00AD\ C0XX_H$

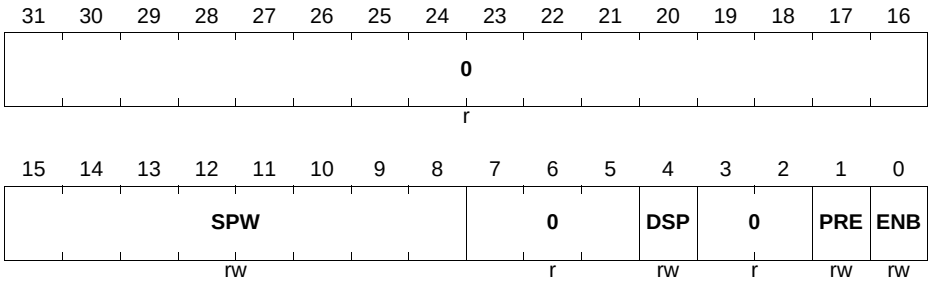


Field	Bits	Type	Description
MOD_REV	[7:0]	r	**Module Revision Number** Indicates the revision number of the implementation. The value of a module revision starts with 01_H (first revision).
MOD_TYPE	[15:8]	r	**Module Type** This bit field is fixed to $C0_H$.
MOD_NUMBER	[31:16]	r	**Module Number Value** This bit field defines the module identification number.

CTR

The operation mode control register.

CTR
WDT Control Register **(04_H)** **Reset Value: 0000 0000$_H$**



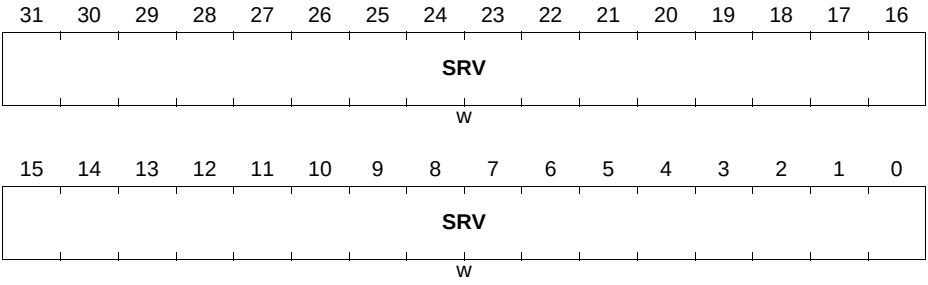
Field	Bits	Type	Description
ENB	0	rw	**Enable** 0_B disables watchdog timer, 1_B enables watchdog timer
PRE	1	rw	**Pre-warning** 0_B disables pre-warning 1_B enables pre-warning,
DSP	4	rw	**Debug Suspend** 0_B watchdog timer is stopped during debug halting mode 1_B watchdog timer is not stopped during debug halting mode
SPW	[15:8]	rw	**Service Indication Pulse Width** Pulse width (SPW+1) of service indication in clk_wdt cycles
0	[3:2], [7:5], [31:16]	r	**reserved**

SRV

The WDT service register. Software must write a magic word while the timer value is within the valid window boundary. Writing the magic word while the timer value is within the window boundary will service the watchdog and result a reload of the timer with 0H.

Upon writing data different than the magic word within valid time window or writing even correct Magic Word but outside of the valid time window no servicing will be performed. Instead will request an immediate system reset request.

SRV
WDT Service Register **(08_H)** **Reset Value: $0000\ 0000_H$**

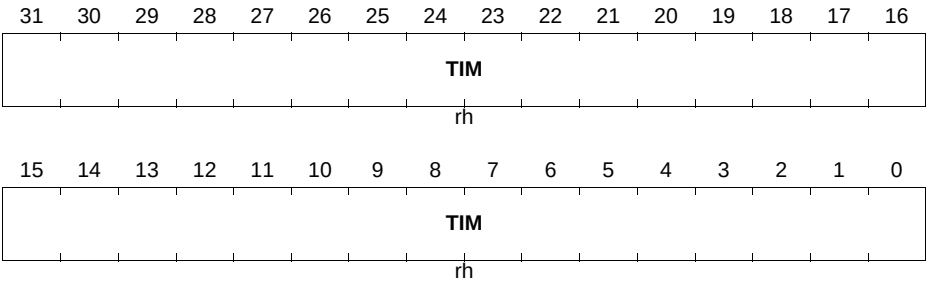


Field	Bits	Type	Description
SRV	[31:0]	w	**Service** Writing the magic word $ABADCAFE_H$ while the timer value is within the window boundary will service the watchdog.

TIM

The actual watchdog timer register count value. This register can be read by software in order to determine current position in the WDT time window.

TIM
WDT Timer Register (0C$_H$) Reset Value: 0000 0000$_H$

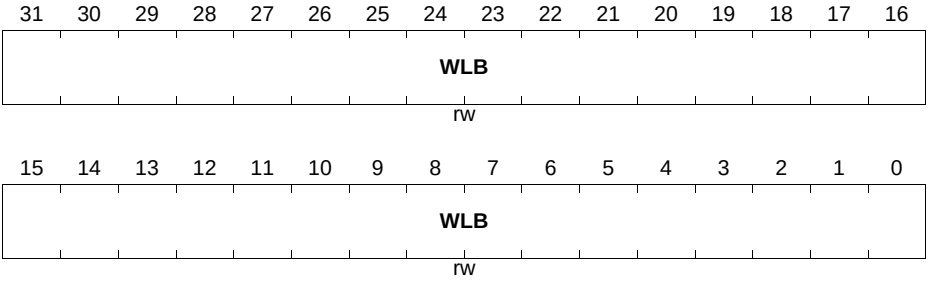


Field	Bits	Type	Description
TIM	[31:0]	rh	**Timer Value** Actual value of watchdog timer value.

WLB

The Window Lower Bound register defines the lower bound for servicing window. Servicing of the watchdog has only effect within the window boundary

WLB
WDT Window Lower Bound Register (10$_H$) Reset Value: 0000 0000$_H$



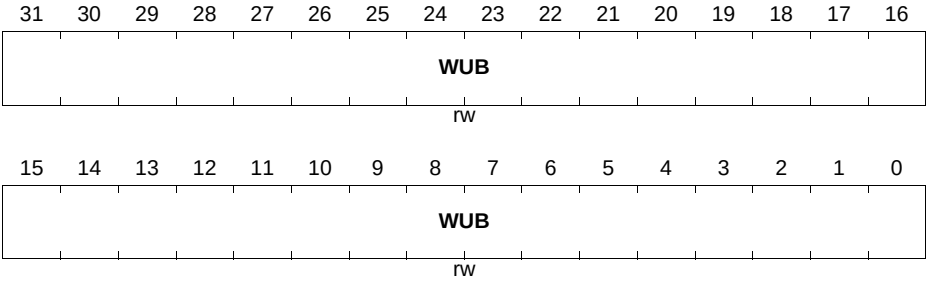
Field	Bits	Type	Description
WLB	[31:0]	rw	**Window Lower Bound** Lower bound for servicing window. Setting the lower bound to 0_H disables the window mechanism.

WUB

The Window Upper Bound register defines the upper bound for servicing window. Servicing of the watchdog has only effect within the window boundary.

WUB

WDT Window Upper Bound Register (14_H) **Reset Value: FFFF FFFF$_H$**

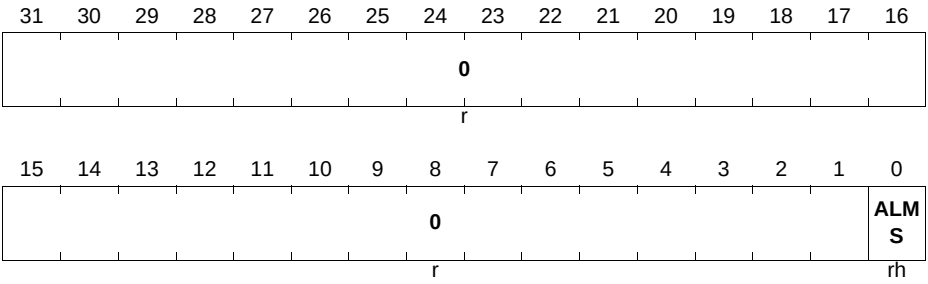


Field	Bits	Type	Description
WUB	[31:0]	rw	**Window Upper Bound** Upper Bound for servicing window. The WDT triggers an reset request when the timer is crossing the upper bound value without pre-warning enabled. With pre-warning enabled the first crossing triggers a watchdog alarm and the second crossing triggers a system reset.

WDTSTS

The status register contains sticky bit indicating occurrence of alarm condition.

WDTSTS
WDT Status Register **(0018_H)** **Reset Value: 00000000_H**

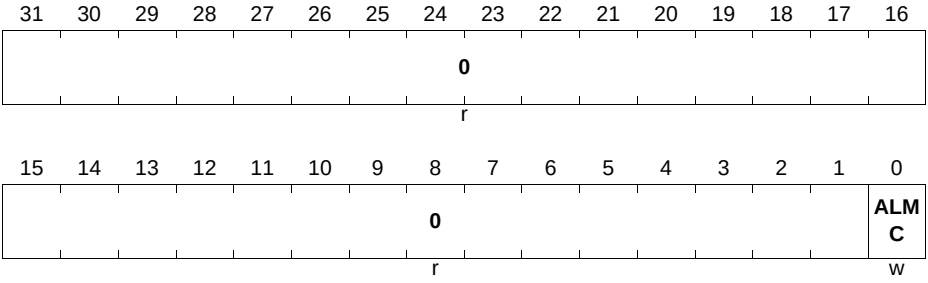


Field	Bits	Type	Description
ALMS	0	rh	**Pre-warning Alarm** 1_B pre-warning alarm occurred, 0_B no pre-warning alarm occurred
0	[31:1]	r	**reserved**

WDTCLR

The status register contains sticky bitfield indicating occurrence of alarm condition.

WDTCLR
WDT Clear Register **($001C_H$)** **Reset Value: 00000000_H**



Field	Bits	Type	Description
ALMC	0	w	**Pre-warning Alarm** 1_B clears pre-warning alarm 0_B no-action
0	[31:1]	r	**reserved**

10.10 Interconnects

Table 10-4 Pin Table

Input/Output	I/O	Connected To	Description
Clock and Reset Signals			
f_{WDT}	I	SCU.CCU	timer clock
Timer Signals			
wdt_service	O	PORTS	service indication to external watchdog
HALTED	I	CPU	In halting mode debug. HALTED remains asserted while the core is in debug.
Service Request Connectivity			
wdt_alarm	O	SCU.GCU	pre-warning alarm
wdt_rst_req	O	SCU.RCU	reset request

11 Real Time Clock (RTC)

Real-time clock (RTC) is a clock that keeps track of the current time. RTCs are present in almost any electronic device which needs to keep accurate time in a digital format for clock displays and computer systems.

11.1 Overview

The RTC module tracks time with separate registers for hours, minutes, and seconds. The calendar registers track date, day of the week, month and year with automatic leap year correction[1]. The clock of RTC is selectable via bit SCU_CLKCR.RTCCLKSEL.

The timer may remains operational during sleep or deep sleep mode.

11.1.1 Features

The features of the Real Time Clock (RTC) module are:

- Real time keeping with
  - 32.768 kHz internal clock
- Periodic time-based interrupt
- Programmable alarm interrupt on time match
- Supports wake-up mechanism from sleep or deep sleep mode

Table 11-1 Application Features

Feature	Purpose/Application
Precise real time keeping	Reduced need for time adjustments
Periodic time-based interrupt	Scheduling of operations performed on precisely defined intervals
Programmable alarm interrupt on time match	Scheduling of operations performed on precisely defined times
Supports wake-up mechanism from sleep or deep sleep mode	Autonomous wakeups from sleep or deep sleep mode for system state control and maintaintenance routine operations

11.1.2 Block Diagram

The RTC block diagram is shown in **Figure 11-1**.

1) The automatic leap year correction is performed when the year is divisible by 4.

The main building blocks of the RTC is Time Counter implementing real time counter and RTC Registers containing multi-field registers for the time counter and alarm programming register where dedicated fields represent a separate values for elapsing second, minutes, hours, days, days of week, months and years.

The RTC module is controlled directly from SCU module and shares system bus interface with other sub-modulesof SCU.

Access to the RTC registers is performed via Register Mirror updated over serial interface.

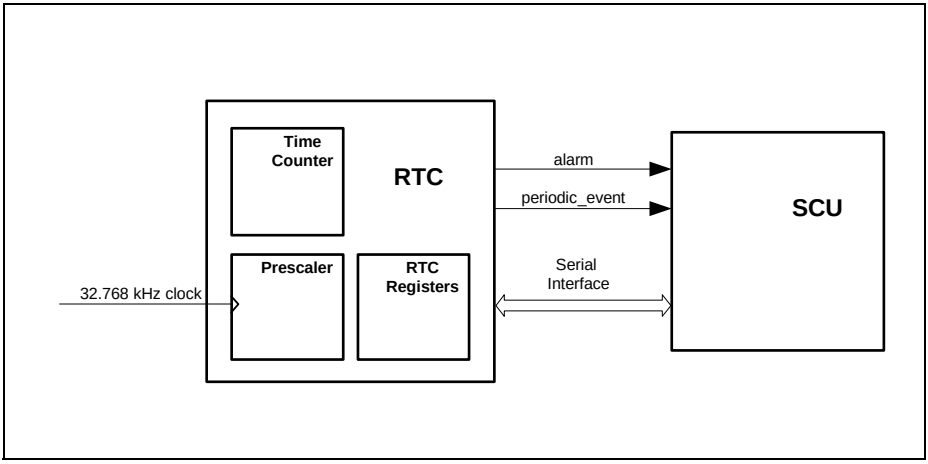


Figure 11-1 Real Time Clock Block Diagram Structure

11.2 RTC Operation

The RTC timer counts seconds, minutes, hours, days, days of week, months and years the time in separate fields (see **Figure 11-2**). Individual bit fields of the RTC counter can be programmed and read with software over serial interface via mirror registers in SCU module.

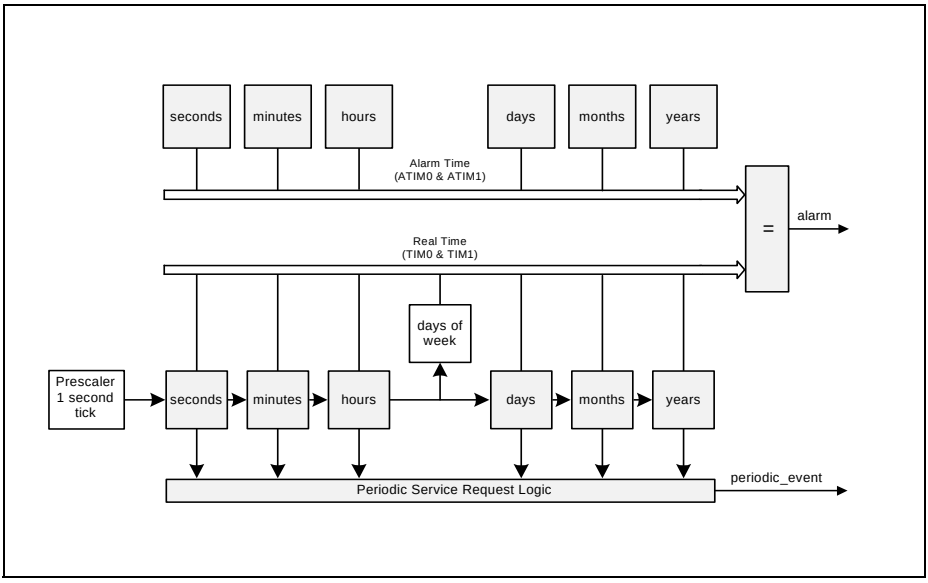


Figure 11-2 Block Diagram of RTC Time Counter

Occurrence of an internal timer event is stored in the service request raw status register **RAWSTAT**. The values of the status register **RAWSTAT** drive the outgoing service request lines alarm and periodic_event.

11.3 Register Access Operations

The RTC module is a part of SCU from programmming model perepcetive and shares register address space for configuration with other sub-modules of SCU. RTC registers are instantiated in the RTC module and mirrored in SCU. The registers get updated in both clock domains over serial interface running at 32kHz clock rate.

Any update of the registers is performed with some delay required for data to propagate to and from the mirror registers over serial interface. Accesses to the RTC registers in core domain must not block the bus interface of SCU module. For details of the register mirror and serial communication handling please refer to SCU chapter.

For consistent write to the timer registers **TIM0** and **TIM1**, the register **TIM0** has to be written before the register **TIM1**.

For consistent read-out of the timer registers **TIM0** and **TIM1**, the register **TIM0** has to be read before the register **TIM1**. The value of **TIM1** is stored in a shadow register upon each read of **TIM0** before they get copied to the mirror register in core domain.

11.4 Service Request Processing

The RTC generates service requests upon:

- periodic timer events
- configured alarm condition

The service requests can be processed in the core domain as regular service requests or as wake-up triggers from sleep or deep sleep mode.

11.4.1 Periodic Service Request

The periodic timer service request is raised whenever a non-masked field of the timer counter gets updated. Masking of the bits is performed using **MSKSR** register. Periodic Service requests can be disabled with **MSKSR**.

11.4.2 Timer Alarm Service Request

The alarm interrupt is triggered when **TIM0** and **TIM1** bit fields values match all corresponding bit fields values of **ATIM0**, **ATIM1** registers. The Timer Alarm Service requests can be enabled/disabled with the **MSKSR** register .

11.5 Debug Behavior

The RTC function can be suspended when the CPU enters HALT mode. RTC debug function is controlled with SUS bit field in **CTR** register.

Note: In XMC1100, bit CTR.SUS is reset to its default value by any reset. Hence, if suspend function is required during debugging, it is recommended to enable the debug suspend function in the user initialisation code. Before programming the register, the module clock has to be enabled and special care need to be handled while enabling the module clock as described in the CCU (Clock Gating Control) section of the SCU chapter.

11.6 Power, Reset and Clock

RTC can be programmed to remains powered up in sleep and deep sleep mode.

The RTC module remains in reset state after initial power up until reset released.

The RTC timer is running from an internal 32.768 kHz clock.The prescaler setting of $7FFF_H$ results in an once per second update of the RTC timer.

The RTC module clock is default disabled and can be enabled via the SCU_CGATCLR0 register.

11.7 Initialization and Control Sequence

Programming model of the RTC module assumes several scenarios where different control sequences apply.

Note: Some of the scenarios described in this chapter require operations on system level the that are not in the scope of the RTC module description, therefore for detailed information please refer to relevant chapters of this document.

11.7.1 Initialization & Start of Operation

Complete RTC module initialization is required upon reset. Accesses to RTC registers are performed via dedicated mirror registers (for more details please refer to SCU chapter)

- enable the clock to RTC module
  – write one to SCU_CGATCLR0.RTC
- program RTC_TIM0 and RTC_TIM1 registers with current time
  – check SCU_MIRRSTS to ensure that no transfer over serial interface is pending to the RTC_TIM0 and RTC_TIM1 registers
  – write a new value to the RTC_TIM0 and RTC_TIM1 registers
- enable RTC module to start counting time
  – write one to RTC_CTR.ENB

Note: To ensure a successful transfer over serial interface, RTC_TIM0 and RTC_TIM1 can only be written once for each transfer. In addition, these registers must be written with a 32-bit data width. Individual bit access will not start the serial transfer operation.

11.7.2 Configure and Enable Periodic Event

The RTC periodic event configuration require programming in order to enable interrupt request generation out upon a change of value in the corresponding bit fields.

- enable service request for periodic timer events in RTC module
  – set respective bit field (MPSE, MPMI, MPHO, MPDA, MPMO, MPYE) in RTC_MSKSR register to enable the individual periodic timer events

11.7.3 Configure and Enable Timer Event

The RTC alarm event configuration require programming in order to enable interrupt request generation out upon compare match of values in the corresponding bit fields of TIM0 and TIM1 against ATIM0 and ATIM1 respectively.

- program compare values in individual bit fields of ATIM0 and ATIM1 in RTC module

  - check SCU_MIRRSTS to ensure that no transfer over serial interface is pending to the RTC_ATIM0 and RTC_ATIM1 registers
  - write to RTC_ATIM0 and RTC_ATIM1 registers
- enable service request for timer alarm events in RTC module
  - set MAI bit field of RTC_MSKSR register in order enable individual periodic timer events

Note: To ensure a successful transfer over serial interface, RTC_ATIM0 and RTC_ATIM1 can only be written once for each transfer. In addition, these registers must be written in 32-bit data width. Individual bit access will not start the serial transfer operation.

11.8 RTC Registers

Registers Overview

The absolute register address is calculated by adding:

Module Base Address + Offset Address

Table 11-2 Registers Address Space

Module	Base Address	End Address	Note
RTC	$4001\ 0A00_H$	$4001\ 0AFF_H$	

Table 11-3 Register Overview

Short Name	Description	Offset Addr.	Access Mode		Description See
			Read	Write	
RTC Kernel Registers					
ID	ID Register	0000_H	U, PV	BE	Page 11-7
CTR	Control Register	0004_H	U, PV	PV	Page 11-8
RAWSTAT	Raw Service Request Register	0008_H	U, PV	BE	Page 11-8
STSSR	Status Service Request Register	$000C_H$	U, PV	BE	Page 11-9
MSKSR	Mask Service Request Register	0010_H	U, PV	PV	Page 11-10
CLRSR	Clear Service Request Register	0014_H	U, PV	PV	Page 11-12
ATIM0	Alarm Time Register 0	0018_H	U, PV	PV	Page 11-13

Table 11-3 Register Overview (cont'd)

Short Name	Description	Offset Addr.	Access Mode		Description See
			Read	Write	
ATIM1	Alarm Time Register 1	$001C_H$	U, PV	PV	Page 11-14
TIM0	Time Register 0	0020_H	U, PV	PV	Page 11-15
TIM1	Time Register 1	0024_H	U, PV	PV	Page 11-16

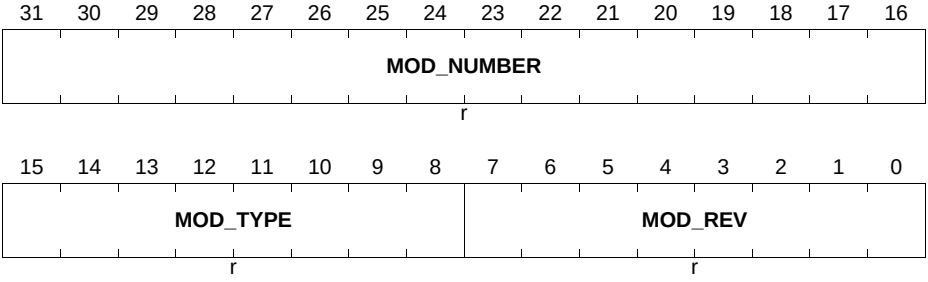
11.8.1 Registers Description

ID

Read-only ID register of the RTC module containing unique identification code of the RTC module.

ID
RTC Module ID Register

(00_H) **Reset Value: 00A3 C0XX$_H$**

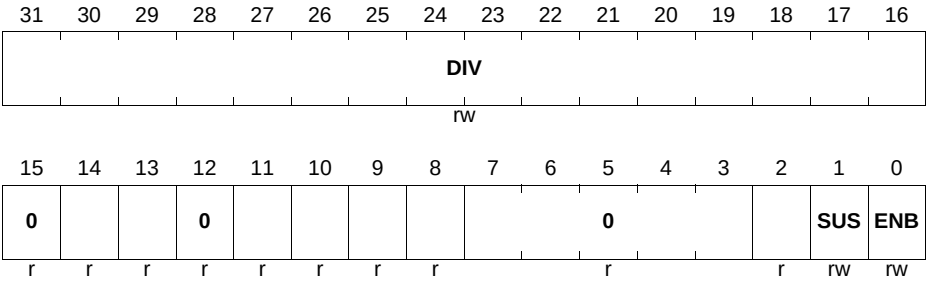


Field	Bits	Type	Description
MOD_REV	[7:0]	r	**Module Revision Number** Indicates the revision number of the implementation. The value of a module revision starts with 01_H (first revision).
MOD_TYPE	[15:8]	r	**Module Type** This bit field is fixed to $C0_H$.
MOD_NUMBER	[31:16]	r	**Module Number Value** This bit field defines the module identification number.

CTR

RTC Control Register providing control means of the operation mode of the module.

CTR
RTC Control Register **(04_H)** **Reset Value: $7FFF\ 0000_H$**



Field	Bits	Type	Description
ENB	0	rw	**RTC Module Enable** 0_B disables RTC module 1_B enables RTC module
SUS	1	rw	**Debug Suspend Control** 0_B RTC is not stopped during halting mode debug 1_B RTC is stopped during halting mode debug
DIV	[31:16]	rw	**Divider Value** reload value of RTC prescaler. Clock is divided by DIV+1. $7FFF_H$ is default value for RTC mode with 32.768 kHz crystal
0	[15:2]	r	**Reserved** Read as 0; should be written with 0

RAWSTAT

RTC Raw Service Request Register contains raw status info i.e. before status mask takes effect on generation of service requests or iterrupts. This register serves debug purpose but can be aslo used for polling of the status withou generating serice requests.

RAWSTAT
RTC Raw Service Request Register (08$_H$) **Reset Value: 0000 0000$_H$**

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

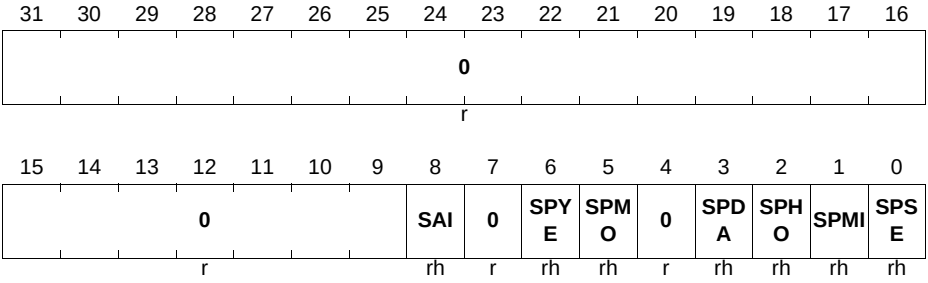
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0							RAI	0	RPYE	RPMO	0	RPDA	RPHO	RPMI	RPSE
r							rh	r	rh	rh	r	rh	rh	rh	rh

Field	Bits	Type	Description
RPSE	0	rh	**Raw Periodic Seconds Service Request** Set whenever seconds count increments
RPMI	1	rh	**Raw Periodic Minutes Service Request** Set whenever minutes count increments
RPHO	2	rh	**Raw Periodic Hours Service Request** Set whenever hours count increments
RPDA	3	rh	**Raw Periodic Days Service Request** Set whenever days count increments
RPMO	5	rh	**Raw Periodic Months Service Request** Set whenever months count increments
RPYE	6	rh	**Raw Periodic Years Service Request** Set whenever years count increments
RAI	8	rh	**Alarm Service Request** Set whenever count value matches compare value
0	4, 7, [31:9]	r	**Reserved**

STSSR

RTC Service Request Status Register contains status info reflecting status mask effect on generation of service requests or iterrupts. This register needs to be accessed by software in order to determine the actual cause of an event.

STSSR
RTC Service Request Status Register ($0C_H$) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
SPSE	0	rh	**Periodic Seconds Service Request Status after masking**
SPMI	1	rh	**Periodic Minutes Service Request Status after masking**
SPHO	2	rh	**Periodic Hours Service Request Status after masking**
SPDA	3	rh	**Periodic Days Service Request Status after masking**
SPMO	5	rh	**Periodic Months Service Request Status after masking**
SPYE	6	rh	**Periodic Years Service Request Status after masking**
SAI	8	rh	**Alarm Service Request Status after masking**
0	4, 7, [31:9]	r	**reserved**

MSKSR

RTC Service Request Mask Register contains masking value for generation control of service requests or iterrupts.

MSKSR
RTC Service Request Mask Register (10$_H$) **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							MAI	0	MPYE	MPMO	0	MPDA	MPHO	MPMI	MPSE
r							rw	r	rw	rw	r	rw	rw	rw	rw

Field	Bits	Type	Description
MPSE	0	rw	**Periodic Seconds Interrupt Mask** 0_B disable interrupt 1_B enable interrupt
MPMI	1	rw	**Periodic Minutes Interrupt Mask** 0_B disable interrupt 1_B enable interrupt
MPHO	2	rw	**Periodic Hours Interrupt Mask** 0_B disable interrupt 1_B enable interrupt
MPDA	3	rw	**Periodic Days Interrupt Mask** 0_B disable interrupt 1_B enable interrupt
MPMO	5	rw	**Periodic Months Interrupt Mask** 0_B disable interrupt 1_B enable interrupt
MPYE	6	rw	**Periodic Years Interrupt Mask** 0_B disable interrupt 1_B enable interrupt
MAI	8	rw	**Alarm Interrupt Mask** 0_B disable interrupt 1_B enable interrupt
0	4, 7, [31:9]	r	**Reserved**

CLRSR

RTC Clear Service Request Register serves purpose of clearing sticky bits of **RAWSTAT** and **STSSR** registers. Write one to a bit in order to clear the status bit. Writing zero has no effect on the set nor reset bits.

CLRSR
RTC Clear Service Request Register (14$_H$) **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							RAI	0	RPYE	RPMO	0	RPDA	RPHO	RPMI	RPSE
r							w	r	w	w	r	w	w	w	w

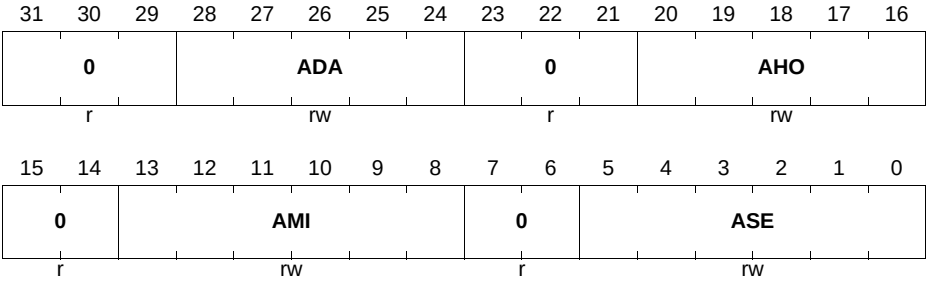
Field	Bits	Type	Description
RPSE	0	w	**Raw Periodic Seconds Interrupt Clear** 0_B no effect 1_B clear status bit
RPMI	1	w	**Raw Periodic Minutes Interrupt Clear** 0_B no effect 1_B clear status bit
RPHO	2	w	**Raw Periodic Hours Interrupt Clear** 0_B no effect 1_B clear status bit
RPDA	3	w	**Raw Periodic Days Interrupt Clear** 0_B no effect 1_B clear status bit
RPMO	5	w	**Raw Periodic Months Interrupt Clear** 0_B no effect 1_B clear status bit
RPYE	6	w	**Raw Periodic Years Interrupt Clear** 0_B no effect 1_B clear status bit

Field	Bits	Type	Description
RAI	8	w	**Raw Alarm Interrupt Clear** 0_B no effect 1_B clear status bit
0	4, 7, [31:9]	r	**Reserved**

ATIM0

RTC Alarm Time Register 0 serves purpose of programming single alarm time at a desired point of time reflecting comparison against **TIM0** register. The ATM0 register contains portion of bit fields for seconds, minutes, hours and days. Upon attempts to write an invalid value to a bit bield e.g. exceeding maximum value a default value gets programmed as described for each individual bit fields.

ATIM0
RTC Alarm Time Register 0 (18_H) Reset Value: 0000 0000$_H$



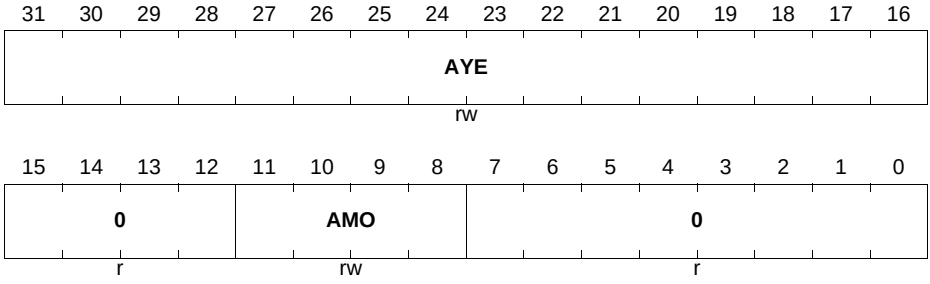
Field	Bits	Type	Description
ASE	[5:0]	rw	**Alarm Seconds Compare Value** Match of seconds timer count to this value triggers alarm seconds interrupt. Setting value equal or above $3C_H$ results in setting the field value to 0_H
AMI	[13:8]	rw	**Alarm Minutes Compare Value** Match of minutes timer count to this value triggers alarm minutes interrupt. Setting value equal or above $3C_H$ results in setting the field value to 0_H

Field	Bits	Type	Description
AHO	[20:16]	rw	**Alarm Hours Compare Value** Match of hours timer count to this value triggers alarm hours interrupt. Setting value equal or above 18_H results in setting the field value to 0_H
ADA	[28:24]	rw	**Alarm Days Compare Value** Match of days timer count to this value triggers alarm days interrupt. Setting valueequal above $1F_H$ results in setting the field value to 0_H
0	[7:6], [15:14], [23:21], [31:29]	r	**Reserved**

ATIM1

RTC Alarm Time Register 1 serves purpose of programming single alarm time at a desired point of time reflecting comparison against **TIM1** register. The ATM1 register contains portion of bit fields for months and years. Upon attempts to write an invalid value to a bit bield e.g. exceeding maximum value a default value gets programmed as described for each individual bit fields.

ATIM1
RTC Alarm Time Register 1 ($1C_H$) Reset Value: $0000\ 0000_H$



Field	Bits	Type	Description
AMO	[11:8]	rw	**Alarm Month Compare Value** Match of months timer count to this value triggers alarm month interrupt. Setting value equal or above the number of days of the actual month count results in setting the field value to 0_H
AYE	[31:16]	rw	**Alarm Year Compare Value** Match of years timer count to this value triggers alarm years interrupt.
0	[7:0], [15:12]	r	**Reserved**

TIM0

RTC Time Register 0 contains current time value for seconds, minutes, hours and days. The bit fields get updated in intervals corresponding with their meaning accordingly. The register needs to be programmed to reflect actual time after initial power up and will continue counting time also while in sleep or deep sleep mode if enabled. Upon attempts to write an invalid value to a bit bield e.g. exceeding maximum value a default value gets programmed as described for each individual bit fields.

TIM0

RTC Time Register 0 **(20_H)** **Reset Value: 0000 0000$_H$**

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

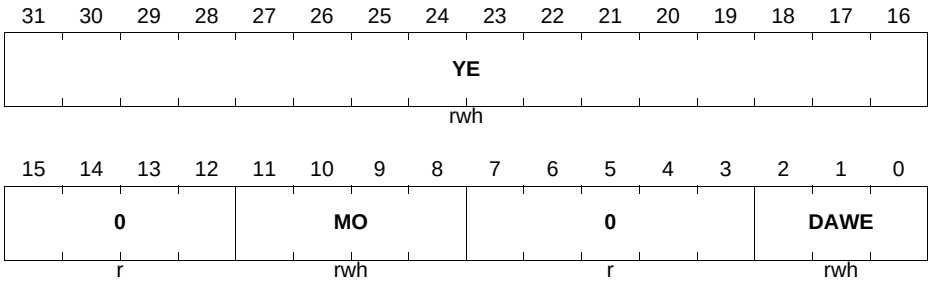
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0			**MI**					**0**		**SE**					
r			rwh					r		rwh					

Field	Bits	Type	Description
SE	[5:0]	rwh	**Seconds Time Value** Setting value equal or above $3C_H$ results in setting the field value to 0_H. Value can only be written, when RTC is disabled via bit CTR.ENB.
MI	[13:8]	rwh	**Minutes Time Value** Setting value equal or above $3C_H$ results in setting the field value to 0_H. Value can only be written, when RTC is disabled via bit CTR.ENB.
HO	[20:16]	rwh	**Hours Time Value** Setting value equal or above 18_H results in setting the field value to 0_H Value can only be written, when RTC is disabled via bit CTR.ENB.
DA	[28:24]	rwh	**Days Time Value** Setting value equal or above the number of days of the actual month count results in setting the field value to 0_H Value can only be written, when RTC is disabled via bit CTR.ENB. Days counter starts with value 0 for the first day of month.
0	[7:6], [15:14], [23:21], [31:29]	r	**Reserved**

TIM1

RTC Time Register 1 contains current time value for days of week, months and years. The bit fields get updated in intervals corresponding with their meaning accordingly. The register needs to be programmed to reflect actual time after initial power up and will continue counting time also while in sleep or deep sleep if enabled. Upon attempts to write an invalid value to a bit bield e.g. exceeding maximum value a default value gets programmed as described for each individual bit fields.

TIM1
RTC Time Register 1 **(24_H)** **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
DAWE	[2:0]	rwh	**Days of Week Time Value** Setting value above 6_H results in setting the field value to 0_H Value can only be written, when RTC is disabled via bit CTR.ENB. Days counter starts with value 0 for the first day of week.
MO	[11:8]	rwh	**Month Time Value** Setting value equal or above C_H results in setting the field value to 0_H. Value can only be written, when RTC is disabled via bit CTR.ENB. Months counter starts with value 0 for the first month of year.
YE	[31:16]	rwh	**Year Time Value** Value can only be written, when RTC is disabled.
0	[7:3], [15:12]	r	**Reserved**

11.9 Interconnects

Table 11-4 Pin Connections

Input/Output	I/O	Connected To	Description
Clock Signals			
f_{RTC}	I	SCU.CCU	32.768 kHz clock selected
Debug Signals			
HALTED	I	CPU	Indicates the processor is in debug state and halted
Service Request Connectivity			
periodic_event	O	SCU.GCU	Timer periodic service request
alarm	O	SCU.GCU	Alarm service request

12 System Control Unit (SCU)

The SCU is the SoC power, reset and a clock manager with additional responsibility of providing system stability protection and other auxiliary functions.

12.1 Overview

The functionality of the SCU described in this chapter is organized in the following sub-chapters, representing different aspects of system control:

- Miscellaneous control functions, GCU **Chapter 12.2**
- Power control, PCU **Chapter 12.3**
- Reset operation, RCU **Chapter 12.4**
- Clock Control, CCU **Chapter 12.5**

12.1.1 Features

The following features are provided for monitoring and controlling the system:

- General Control
  - Start-up Software (SSW) and Boot Mode Support
  - Memory Content Protection
  - Interrupt Handling
- Power control
  - On-chip core supply generation via EVR
  - Power Validation
  - Supply Watchdog
  - Voltage Monitoring
  - Load Change Handling
- Reset Control
  - Reset assertion on various reset request sources
  - System Reset Generation
  - Inspection of reset sources after reset
- Clock Control
  - Clock Generation
  - Clock Supervision
  - Individual Peripheral Clock Gating
  - Clock Blanking Support

12.1.2 Block Diagram

Figure 12-1 shows the following sub-units:

- Power Control Unit (PCU)
- Reset Control Unit (RCU)
- Clock Control Unit (CCU)
- General Control Unit (GCU)

All the SFRs in SCU module are accessible via the AHB and the 16-bit APB bus interface as shown in **Figure 12-1**.The APB bus interface is used to access the group of SFR called ANACTRL register. These registers are used to configure the analog modules in the system, namely, the embedded voltage regulator (EVR) and the digitally controlled oscillators (DCO1 and DCO2). The other group of SFR called SCU register are accesible via the AHB bus interface.

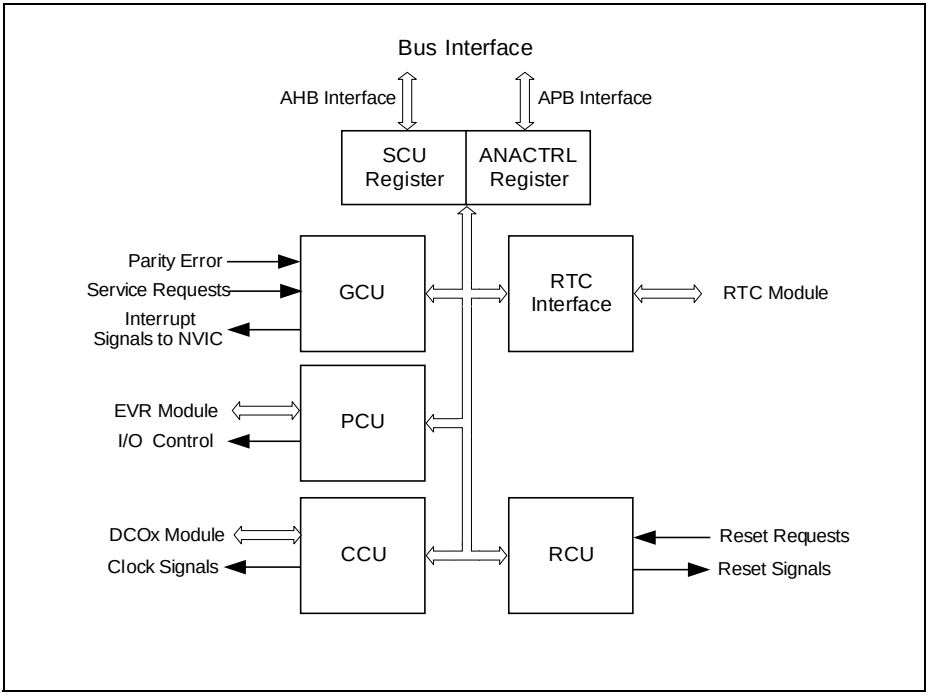


Figure 12-1 SCU Block Diagram

Interface of General Control Unit

The General Control Unit GCU has a memory fault interface to the memory validation logic of each on-chip SRAM and the Flash to receive memory fault events, as parity errors.

Interface of Power Control Unit

The Power Control Unit PCU has an interface to the Embedded Voltage Regulator (EVR) and an interface to the CCU module. The PCU related signals are described in more detail in **Chapter 12.3**.

Interface of Reset Control Unit

The Reset Control Unit RCU has an interface to the Embedded Voltage Regulator (EVR). The RCU receives the power-on reset and the brownout reset information from the EVR. Reset requests are coming to the unit from the watchdog, the CPU, the GCU and the clock control unit (CCU). The RCU is providing the reset signals to all other units of the chip in the core power domain. The RCU related signals are described in more detail in **Chapter 12.4**.

Interface of Clock Control Unit

The Clock Control Unit (CCU) receives the clock source from the on-chip Digitally Controlled Oscillator (DCO). The CCU provides the clock signals to all other units of the chip.

Interface of RTC

Access to the RTC module is served over a serial interface. The interface provides mirror registers updated via the serial interface to the RTC module registers. Update of the mirror registers over the serial is controlled using **MIRRSTS** registers. End of update can also trigger service requests via **SRRAW** register. Refresh of the registers in the register mirror are performed continuously, as fast as possible in order to instantly reflect any register state change on both sides.

The RTC module functionality is described in separate RTC chapter.

12.2 Miscellaneous Control Functions (GCU)

System Control implements system management functions accessible via GCU registers. General system control including various auxillary function is performed in the General Control Unit (GCU).

12.2.1 Service Requests Handling

Service request events listed in **Table 12-1** can result in assertion of an interrupt. Please refer to **SRMSK** a register description.

The interrupt structure is shown in **Figure 12-2**. The interrupt request or the corresponding interrupt set bit (in register **SRSET**) can trigger the interrupt generation at the selected interrupt node x. The service request pulse is generated independently from the interrupt flag in register **SRRAW**. The interrupt flag can be cleared by software by writing to the corresponding bit in register **SRCLR**.

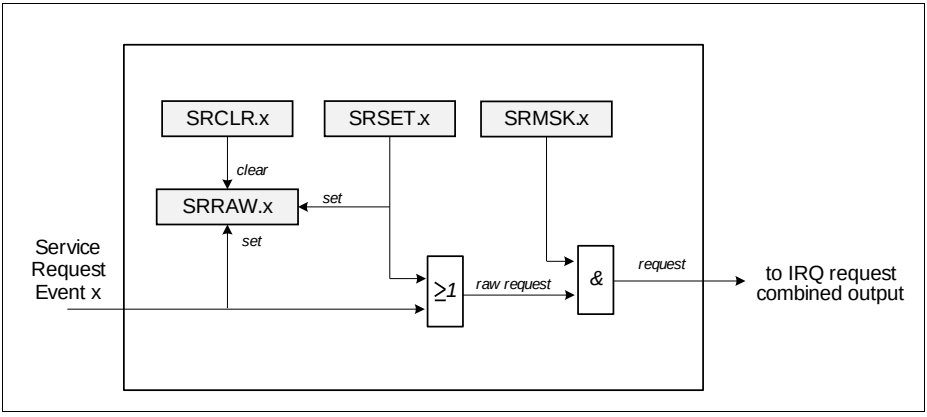


Figure 12-2 Service Request Handling

The flag in register **SRRAW** can be cleared by software by writing to the corresponding bit in register **SRCLR**. All trap requests are combined to one common line and connected to a regular interrupt node of NVIC.

Note: When servicing an SCU service request, make sure that all related request flags are cleared after the identified request has been handled.

12.2.1.1 Service Request Sources

The SCU supports service request sources listed in **Table 12-1** and reflected in the **SRRAW**, **SRMSK**, **SRCLR** and **SRSET** registers. The events that trigger these service

requests are described in the respective module chapters or in the various sections within SCU.

Table 12-1 Service Requests

Modules	Service Request Name	Service Request Short Name	SCU.SRx
NVM	Flash Double Bit ECC Event	FLECC2I	SR0
	Flash Operation Complete Event	FLCMPLTI	
	16kbytes SRAM Parity Error Event	PESRAMI	
	USIC0 SRAM Parity Error Event	PEU0I	
SCU:CCU	Loss of Clock Event	LOCI	
	Standby Clock Failure Event	SBYCLKFI	SR1
SCU:PCU	VDDP Pre-warning Event	VDDPI	
	VDROP Event	VDROPI	
	VCLIP Event	VCLIPI	
WDT	WDT pre-warning	PRWARN	
RTC	RTC Periodic Event	PI	
	RTC Alarm	AI	
	RTC CTR Mirror Register Updated	RTC_CTR	
	RTC ATIM0 Mirror Register Updated	RTC_ATIM0	
	RTC ATIM1 Mirror Register Updated	RTC_ATIM1	
	RTC TIM0 Mirror Register Updated	RTC_TIM0	
	RTC TIM1 Mirror Register Updated	RTC_TIM1	

12.2.2 SRAM Memory Content Protection

For supervising the content of the on-chip SRAM memories, the following mechanism is provided:

All on-chip SRAMs provide protection of content via parity checking. The parity logic generates additional parity bits which are stored along with each data word at a write operation. A read operation implies checking of the previous stored parity information.

An occurrence of a parity error is observable at the **SRRAW** status register. It is configurable via **SRMSK** whether a memory error should trigger an interrupt. It can also trigger a system reset when **RSTCON**.SPERSTEN or **RSTCON**.U0PERSTEN is set to 1.

A parity control software test, such as to support in-system testing to fulfill Class B requirements, can be enabled with bit **PMTSR**.MTENS for the 16 kbytes SRAM memory individually. Once this bit is set, an inverted parity bit is generated during a write

operation. When a read operation is performed on this SRAM address, a parity error shall be detected.

Note: Test software should be located in a different memory space.

12.2.3 Summary of ID

This section describes the various ID in XMC1100.

Module Identification

The module identification register indicates the function and the design step of each peripherals. Register SCU_**ID** is used for SCU module.

System ROM Table ID

The PID values in the system ROM table are defined in the **Table 12-2**. The XMC1100 system ROM table is located at F000 0000_H. Cortex-M0 ROM table is described in the debug chapter.

Table 12-2 PID Values of XMC1100 System ROM Table

Name	Offset	Reference	Values
PID0	$FE0_H$	XMC1100 Part Number [7:0]	ED_H
PID1	$FE4_H$	bits [7:4] JEP106 ID code [3:0] bits [3:0] XMC1100 Part Number [11:8]	11_H
PID2	$FE8_H$	bits [7:4] XMC1100 Revision bit [3] == 1: JEDEC assigned ID fields bits [2:0] JEP106 ID code [6:4]	$1C_H$
PID3	FEC_H	bits [7:4] RevAnd, minor revision field bits [3:0] if non-zero indicate a customer-modified block	00_H
PID4	$FD0_H$	bits [7:4] 4KB count bits [3:0] JEP106 continuation code	00_H

Chip Identification Number

The chip identification number is a 8 word length number. It consists of the values in register DBGROMID, IDCHIP, register PAU_FLSIZE, register PAU_RAM0SIZE and register PAU_AVAILn(n=0-2) respectively. This number is for easy identification of product variants information of the device, example, package type, the temperature profile, flash size, RAM size and the peripheral availability.

12.3 Power Management (PCU)

Power management control is performed in the Power Control Unit (PCU).

12.3.1 Functional Description

The XMC1100 is running from a single external power supply of 1.8 - 5.5V (V_{DDP}). The main supply voltage is supervised by a supply watchdog.

The I/Os is running directly from the external supply voltage. The core voltage (V_{DDC}) is generated by an on-chip Embedded Voltage Regulator (EVR). The safe voltage range of the core voltage is supervised by a power validation circuit, which is part of the EVR.

12.3.2 System States

The system has the following general system states:

- Off
- Active
- Sleep
- Deep-Sleep

Figure 12-3 shows the state diagram and the transitions between the states.

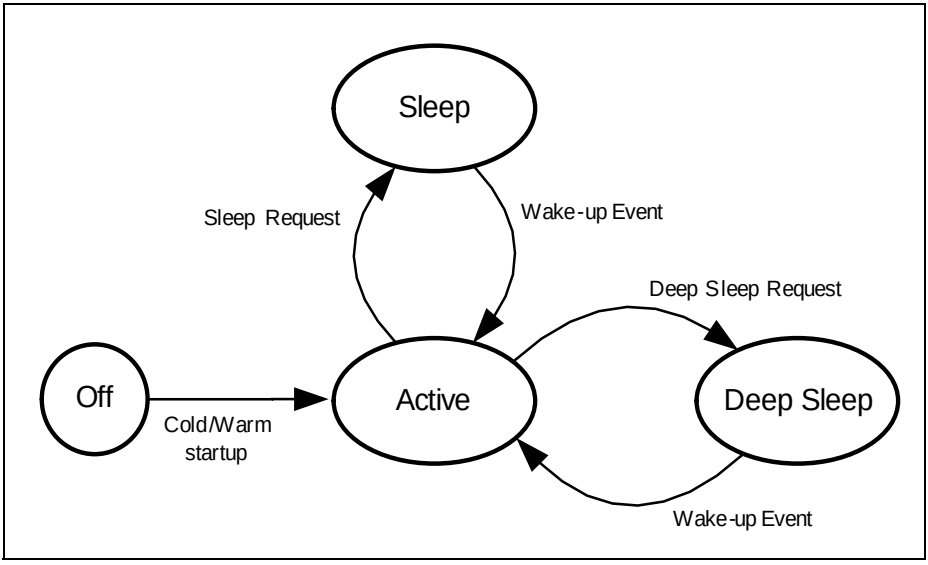


Figure 12-3 System States Diagram

Active State

The active state is the normal operation state. The system is fully powered. The CPU is usually running from a high-speed clock. Depending on the application, the system clock might be slowed down. Unused peripherals might be stopped by gating the clock to these peripherals.

Sleep State

The sleep state of the system corresponds to the sleep state of the CPU. The state is entered via WFI or WFE instruction of the CPU. In this state, the clock to the CPU is stopped. To save power, the clock of the peripherals that are not needed during sleep state can be gated by register **CGATSET0** before entering sleep state.

The Flash can be put into shutdown mode during active state to achieve a further power reduction before entering sleep state via bit NVMCONF.NVM_ON. However, user code would have to be executed in SRAM before entering sleep state and after waking up from sleep state.

To avoid the switching of code execution to SRAM due to the shutdown of flash, register **PWRSVCR** can be used. When FPD bit is set, flash is shutdown only when the device has entered sleep state. The shut down operation is performed after the core has executed WFI/WFE instruction. After a wake-up event is detected, the system will resumed to the previous state i.e. the flash is operable again before the CPU can continue to fetch and execute the code. In this case, user code can be executed in Flash and no switching to SRAM in needed. In his approach, the wake-up time is longer because of the time needed for flash to reach the active state.

Peripherals can continue to operate unaffected and eventually generate an event to wake-up the CPU. In User with Debug mode (UMD) or User with Debug mode and HAR (UMHAR), a Debug HALT request is also able to wake-up the CPU. Any accordingly configured interrupt will bring the CPU back to operation via the NVIC or the M0 debug system.

Deep-Sleep State

The deep-sleep state is entered on the same mechanism as the sleep state with the addition that user code has enabled the deep-sleep state in system control register. This state is similiar to sleep state, except that in deep-sleep state, the PCLK and MCLK will be switched to a slow standby clock and DCO1 will be put into power-down mode.

The shutting down of flash via NVMCONF.NVM_ON or PWRSVCR.FPD as explained in above section is applicable for deep-sleep mode.

Peripherals that continue to operate will run using the slow standby clock and can eventually generate an event to wake-up the CPU. In User with Debug mode (UMD) or User with Debug mode and HAR (UMHAR), a Debug HALT request is also able to wake-up the CPU. Any accordingly configured interrupt will bring the CPU back to operation

via the NVIC or the M0 debug system. The clock system is restored to the previous configuration for active state upon wake-up. Peripherals that are active will run with restored clock configuration.

The SRAM content is preserved in the deep-sleep state.

Note: It is recommended to slow down the PCLK and MCLK before entering deep sleep mode to prevent a sudden load change that could cause a brownout reset.

12.3.3 Embedded Voltage Regulator (EVR)

The EVR generates the core voltage V_{DDC} out of the external supplied voltage V_{DDP}. The EVR provides 2 supply monitoring detectors for the input voltage V_{DDP}. The generated core voltage V_{DDC} is monitored by a power validation circuit (PV).

12.3.4 Power-on Reset

The EVR starts operation as soon as V_{DDP} is above defined minimum level. It releases the reset, when the external voltage V_{DDP} and the generated voltage V_{DDC} are above the reset thresholds and reaching the nominal values.

12.3.5 Power Validation

A power validation circuit monitors the internal core supply voltage,V_{DDC}. It monitors that the core voltage is above the voltage threshold V_{DDCBO} which guarantees safe operation. Whenever the voltage falls below the threshold level, a brownout reset is generated.

12.3.6 Supply Voltage Monitoring

There are 2 detectors, namely, External voltage detector (VDEL) and External brownout detector (BDE) in the EVR that are used to monitor the V_{DDP}.

VDEL detector compares the supply voltage against a pre-warning threshold voltage. The threshold level is programmable via register ANAVDEL.VDEL_SELECT. An interrupt if enabled, will be triggered if a level below this threshold is detected and the flag, VDDPI, in SRRAW register bit is set. An indication bit, VDESR.VDDPPW, shows the output of the detector.

BDE detector is used to trigger a brownout reset when the V_{DDP} supply voltage drops below the defined threshold. Similiarly, it is also used to ensure a proper startup when the V_{DDP} is above the defined threshold during the power-up phase.

The Data Sheet defines the nominal value and applied hysteresis

12.3.7 V_{DDC} Response During Load Change

In XMC1100, the core voltage level, V_{DDC}, drops below the typical threshold when there is an increase in the load and rises when there is a decrease in the load. 2 detectors,

VDROP and VCLIP detectors are used to montior the lower limits and the upper limits of the core voltage level respectively (detectors details are described in the next section). A VDROP event happens when VDDC drops below the VDROP threshold voltage. A VCLIP event happens when VDDC rises above the VCLIP threshold voltage. Each of these events can triggered it's dedicated interrupt if enabled via SRMSK register and the event status can be monitior via SRRAW register.

In XMC1100, the following scenarios may trigger a VDROP/VCLIP event due to load change:

- Changing the MCLK and PCLK frequency via CLKCR register
- Enabling/Gating the peripheral clock via the CGATSET0/CGATCLR0 registers

When a sudden load change happens (< 4*baseload or > 0.25*baseload)(TBC), irregardless of an increase or decrease in load, the EVR needs time (15 usec) to regulate the core voltage back to a stable nominal voltage. During this period of time, the system is expected to maintain in the current load and no load change is allowed. Status bit VDDC2LOW and VDDC2HIGH in CRCLK register are used to indicate whether the voltage is stable.

Note: It is not recommended to increase the load more than 4 times of the baseload or decrease the load to less than 0.25 times of the baseload. If a bigger than the specified load change is required, the recommendation is to change the load in steps that each steps are within the limits. For example, a final load of 16mA from the current baseload of 1mA needs at least 2 steps. A step from 1mA to 4mA followed by another step from 4mA to 16mA

The VDDC2LOW and VDDC2HIGH status bit is generated by a 10-bit counter using DCO1, 64MHz clock as the clock input. It is implemented to count the 16 usec (default) that is needed to have a stable V_{DDC} after a VDROP or VCLIP event happens. The length of this counter can be changed depending on the amount of load change via bit CLKCR.CNTADJ. The larger is the load change, the more is the time that user need to wait for a stable clock. Refer to datasheet for a guideline of the current consumption of each modules.

Using the example above, after programming some configuration that causes a change in load from 1mA to 4mA, user can poll for VDDC2LOW (CNTADJ=$3FF_H$) to ensure the 16 usec(max) needed to regulate EVR. After VDDC2LOW is set to 0, another load change from 4mA to 16mA can be performed and the cycle repeats for each step of load change.

During a VDROP event, clock blanking happens and the detail description is documented in **"Clock Blanking" on Page 12-16**. CPU clock and peripheral clocks continue to run during VCLIP event.

Note: When overflow event happens while the VDROP=1 (time to overflow based on the CNTADJ value is shorter than the time the device stays in a vdrop event), the 10-bit counter will automatically be restarted with the CNTADJ value.

Note: VDROP and VCLIP detectors are disabled during deep sleep mode.

12.3.8 Flash Power Control

The Flash module can be switched off to reduce static power. In sleep or deep-sleep state, it is dependent on the setting of register **PWRSVCR** whether the Flash module will be put to sleep or not in this state. The user has to evaluate the reduced leakage current against the longer startup time. In addition, the Flash can also be put to sleep using NVMCONF.NVM_ON before entering these power save modes.

12.4 Reset Control (RCU)

Reset Control Unit performs control of all reset related functionality including:

- Reset assertion on various reset request sources
- Inspection of reset sources after reset
- Selective reset of peripheral

12.4.1 Functional Description

The XMC1100 has the following reset types for the system:

- Master Reset, $\overline{\text{MRESET}}$
- System Reset, $\overline{\text{SYSRESET}}$

Master Reset, $\overline{\text{MRESET}}$

Master reset is triggered by:

- Power-on reset (PORST)
- V_{DDP} or V_{DDC} undervoltage reset (also known as brownout reset)
- SW master reset via setting bit RSTCON.MRSTEN to 1

A complete reset to the device is executed by a master reset. Master reset is triggered by a power-on reset upon power-up. Whenever the supply V_{DDP} is ramped-up and crossing the V_{DDP} and V_{DDC} voltage threshold, the power-on reset is released. A power-on reset (also known as brown-out reset) is asserted again whenever the V_{DDP} voltage or the V_{DDC} voltage falls below reset thresholds. In additional, a master reset can be triggered by setting bit RSTCON.MRSTEN.

The sources that trigger a master reset also triggers a system reset $\overline{\text{SYSRESET}}$.

System Reset, $\overline{\text{SYSRESET}}$

System reset is triggered by:

- SW reset via Cortex M0 Application Interrupt and Reset Control Register (AIRCR)
- Lockup signal from Cortex M0 when enabled at RCU
- Watchdog reset
- Memory parity error when enabled
- Flash ECC double bit error when enabled
- Loss of clock when enabled
- sources that trigger a master reset

A system reset affects almost all logics. The only exceptions are RCU registers and Debug system when debug probe is present. The reset gets extended to the length defined by implementation requirements.

The debug system is reset by System Reset in normal operation mode when debug probe is not present. When debug probe is present, System Reset must not affect the Debug system.

12.4.2 Reset Status

The EVR provides the cause of a power on reset to the RCU. The reset cause can be inspected after resuming operation by reading register **RSTSTAT**. This register also indicates the source of event that causes the triggering of a system reset.

All registers of the RCU undergo reset only by a master reset except for RSTSTAT and RSTCON register. RSTSTAT register is only reset by a power-on reset and RSTCON is reset by any reset type.

*Note: Clearing of the reset status via register bit **RSTCLR**.RSCLR is strongly recommended to ensure a clear indication of the cause of next reset.*

Table 12-3 shows an overview of the reset signals their source and effects on the various parts of the system.

Table 12-3 Reset Overview

Module/Function	Power-on Reset	Master Reset by SW bit	System Reset
CPU Core	yes	yes	yes
SCU	yes	yes	yes, except reset indication bit
Peripherals	yes	yes	yes
Debug System	yes	yes	see footnote[1)2)]
Port Control	yes	yes	yes
SRAM	Affected,unreliable	Not affected	Not affected
Flash	yes	yes	yes
EVR	yes	no[3)]	no
Clock System	yes	yes	yes

1) Debug system will be reset only if debug probe is not present.

2) Access to debug interface is disabled after every reset even when debug probe is present. See Warm Reset section of Debug System chapter for more details.

3) The supply to EVR will not be affected and hence a complete reset to EVR is not possible. However, it will be partially affected by the reset in the ANACTRL module.

12.5 Clock Control (CCU)

12.5.1 Features

The clock control unit CCU has the following functionality:

- Dedicated RTC and standby clock
- Clock Supervisory
  - Oscillator watchdog
- Wide range of frequency scaling of system frequencies
- Individual peripheral clock gating

12.5.2 Clock System and Control

Figure 12-4 shows the block diagram of the clock system in XMC1100. It consists of two oscillator (DCO1 and DCO2) and a clock control unit (CCU). DCO1 has a clock output, dco1_dclk running at 64MHz. DCO2 is used to generate the standby clock running at 32.768kHz. The main clock, MCLK, and fast peripherial clock, PCLK, are generated from dco1_dclk. PCLK is running in either the same frequency as MCLK or double the frequency of MCLK. It is selectable via CLKCR.PCLKSEL.

Figure 12-4 shows the list of peripherals that are running in the PCLK domain. The rest of the peripherals except RTC and WDT are clocked by MCLK which is the same as the core and bus system. RTC and WDT is running at a frequency of 32.768kHz from the standby clock which is asychronous to the MCLK and PCLK clock.

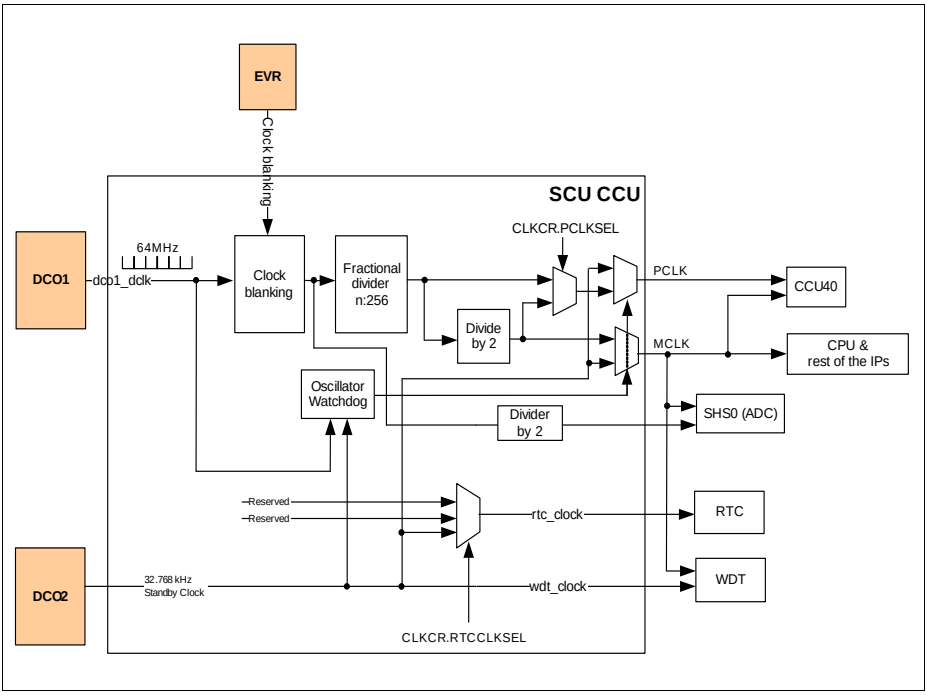


Figure 12-4 Clock System Block Diagram

Note: SHS(ADC) clock will not switched to standby clock source when there is a loss of clock event.

Fractional Divider

The frequency of MCLK and PCLK are programmable through a fractional divider. PCLK has a range of frequency from 125kHz to 64MHz and MCLK has a range from 125kHz to 32MHz.

The following formula calculate the MCLK clock frequency.

(12.1)

$$\left(MCLK = \frac{dco\_dclk}{(2) \times \left(IDIV + \frac{FDIV}{256} \right)} \right) \text{for IDIV} > 0$$

The following formula calculate the PCLK clock frequency when CLKCR.PCLKSEL is set to 1 and it is double the frequency of MCLK.

(12.2)

$$\left(PCLK = \frac{dco\_dclk}{\left(IDIV + \frac{FDIV}{256} \right)} \right) \text{for IDIV} > 0$$

IDIV represents an unsigned 8-bit integer from the bit field CLKCR.IDIV and FDIV/256 defines the fractional divider selection in CLKCR.FDIV. Changing of the MCLK and PCLK can be done within 2 clock cycles. While changing the frequency of MCLK clock and PCLK clock, it is recommended to disable all interrupts to prevent any access to flash that may results in an unsuccessful flash operation.

*Note: Changing the MCLK and PCLK frequency may result in a load change that causes clock blanking to happen. Refer to **Clock Blanking** and **VDDC Response During Load Change** for more details.*

Clock Blanking

To prevent a brown-out reset in case of a sudden positive load change, the clock blanking circuitry is used. It freezed the clock input to the fractional divider to regulate the

load change. The clock blanking only causes a small jitter if it is considered over a longer time period.

The clock blanking is activated when V_{DDC} is detected to be below the VDROP threshold. Once the V_{DDC} is above this threshold, the enabled clocking is resume. This voltage drop detector is part of the EVR and it is activated by default upon any reset.

To monitor clock blanking activities, user can enable interrupt but can only enter ISR after the clock resume.

In addition to the use of clock blanking function to prevent a brown-out reset, the system is also expected to maintain in the current load and no further load change is allowed for 16 usec (max). Detail description of the core voltage behaviors during load change are described in **"VDDC Response During Load Change" on Page 12-9**.

As described in **"VDDC Response During Load Change" on Page 12-9**, the status bit CLKCR.VDDC2LOW is used to indicate to user that the core voltage, V_{DDC}, is below the nominal voltage and EVR is in the process of regulating it. During this period, the clock could be also not running in the selected speed and it is recommended to poll this bit before continuing with any large change to the current load.

Note: It is recommended to slow down the PCLK and MCLK before entering deep sleep mode to prevent a sudden load change that could cause a brownout reset when entering .

12.5.2.1 Oscillator Watchdog

The oscillator watchdog (OWD) monitors the DCO1 clock frequency using the standby clock as the reference clock. It can be disabled via OSCCSR.OWDEN. By setting bit OSCCSR.OWDRES[1)], the detection for DCO1 clock frequency can be restarted. The detection status output is only valid after some cycles of the standby clock frequency. When the OWD is disabled, the detection status will be reset and no detection is possible.

If the OWD is enabled before entering deep sleep mode, it will be disabled automatically by hardware when it enters deep sleep mode and re-enabled again after exiting deep sleep mode. The detection is reset and restarted after device wake-up from deep sleep mode.

12.5.2.2 Loss of Clock Detection and Recovery

Loss of clock happens when the oscillator watchdog (OWD) detects a DCO1 frequency that is less than 50 MHz or more than 68.5 MHz during normal operation. In this case,

1) It is recommended to clear the status bit SRRAW.LOCI and SRRAW.SBYCLKFI before restarting the detection.

an interrupt will be generated if it is enabled. Concurrently, the oscillator status flag, OSCCSR.OSC2L or OSCCSR.OSC2H, is set to 1 and the system clock will be provided by the standby clock of 32.768kHz. Emergency routines can be executed to safely shut down the system. Beside triggering an interrupt when the loss of clock happens, a system reset can also be triggered if it is enabled by **RSTCON**.LOCRSTEN.

*Note: Switching to standby clock during loss of clock event happens only if DCO2 is still running (>0MHz). Bit **SRRAW**.SBYCLKFI indicates the fail status of the standby clock.*

The XMC1100 remains in this loss of clock state until the next reset or after a successful clock recovery has been performed. A clock recovery could be carried out by restarting the detection by setting bit OSCCSR.OWDRES. Upon detecting a stable oscillator frequency of more than 50 MHz and lower than 68.5MHz, OSC2L and OSC2H will be set to 0 and the MCLK will switched automatically to the DCO1 clock source.

12.5.2.3 Standby Clock Failure

The standby clock failure event happens when the OWD detects a failure in standby clock where the clock stops running (~0kHz). Bit SRRAW.SBYCLKFI is set to 1 and trigger an interrupt via SCU_SR1 service request if the interrupt is enabled in register SRMSK.

12.5.2.4 Startup Control for System Clock

When the XMC1100 starts up after reset, system frequency is provided by the DCO1 oscillator. After reset, CPU runs in 8MHz, default frequency. User can change the clock frequency that is used to execute the SSW by defining it in the flash memory location $1000\ 1010_H$. Refer to Boot and Startup chapter for more details.

12.5.3 Clock Gating Control

The clock to peripherals can be individually gated and parts of the system can be stopped by registers **CGATSET0**. After a master reset, only core, memories, SCU and PORT peripheral are not clock gated. The rest of the peripherals are default clock gated. User can select the clock of individual modules to be enabled by SSW after reset by defining it in the flash memory location $1000\ 1014_H$. Refer to Boot and Startup chapter for more details.

Load change during module clock enabling or gating

Enabling or gating the clock to peripherals could result in a load change that could cause clock blanking to happen. In addition, a load change of more than 4 times the current load would required the system to maintain in the current load and no further load change

is allowed for 16 usec (max). See **VDDC Response During Load Change** for more details.

Module clock gating in Sleep and Deep-Sleep modes

It is recommended to gate the clock using registers **CGATSET0** for module that is not needed during sleep mode or deep-sleep mode. These modules must be disabled before entering sleep or deep-sleep mode. In addition, the PCLK and MCLK will be switched to a slow standby clock and DCO1 will be put into power-down mode in deep-sleep mode.

12.6 Service Request Generation

The SCU module provides 2 service request outputs SR[1:0]. SR0 is for system critical request, such as loss of clock event. SR1 is for the common SCU request such as WDT pre-warning request. The service request outputs SR[1:0] are connected to interrupt nodes in the Nested Vectored Interrupt Controller (NVIC).

Refer to **Section 12.2.1** for more details.

12.7 Debug Behavior

The SCU module does not get affected with the HALTED signal from SCU upon debug activities performed using external debug probe.

12.8 Power, Reset and Clock

The SCU module implements functions that involve varoius types of modules controlled directly or via dedicated interfaces that are instantiated in different power, clock and reset domains. These modules are functionally considered parts of the SCU and therefore SCU is also considered a multi domain circuit in this sense.

Power domains:

Power domains get separated with appropriate power separation cells.

- Core domain supplied with V_{DDC} voltage
- Pad domain supplied with V_{DDP} voltage

Clock domains:

All cross-domain interfaces implement signal synchchronization.

- internal SCU clock is MCLK, always identical to the CPU clock
- RTC and register mirror interface clock is 32.786 kHz clock generated from DCO2 oscillator

Reset domains:

All resets get internally synchronized to respective clocks.

- System Reset ($\overline{\text{SYSRESET}}$) resets most of the logics in SCU and can be triggered from various sources (please refer to **Reset Control (RCU)** section for more details)
- Master Reset ($\overline{\text{MRESET}}$) contributes in generation of the System Reset and gets triggered upon power-up sequence of the Core domain

12.9 Registers

This section describes the registers of SCU which some of them resides in the ANACTRL module. Most of the registers are reset by SYSRESET reset signal but some of the registers can be reset only with power-on reset. SCU registers are accessible via the AHB-lite bus. ANACTRL registers are accessible via the APB bus. ANACTRL registers have name starting with “ANA”.

Table 12-4 Base Addresses of sub-sections of SCU registers

Short Name	Description	Offset Addr.[1)]
GCU Registers	Offset address of General Control Unit	0000_H
PCU Registers	Offset address of Power Control Unit	0200_H
CCU Registers	Offset address of Clock Control Unit	0300_H
RCU Registers	Offset address of Reset Control Unit	0400_H
RTC Registers	Offset address of Real Time Clock Module	$0A00_H$
ANACTRL Registers	Offset address of ANACTRL registers	1000_H

1) The absolute register address is calculated as follows:
Module Base Address + Sub-Module Offset Address (shown in this column) + Register Offset Address

Following access to SCU/ANACTRL SFRs result in an AHB/APB error response:

- Read or write access to undefined address
- Write access to read-only registers
- Write access to startup protected registers

Table 12-5 Registers Address Space

Module	Base Address	End Address	Note
SCU	$4001\ 0000_H$	$4001\ FFFF_H$	System Control Unit Registers

Table 12-6 Registers Overview

Short Name	Description	Offset Addr.[1]	Access Mode		Description See
			Read	Write	
PCU Registers (ANACTRL)					
ANAVDEL	Voltage Detector Control register	1050_H	U, PV	U, PV	Page 12-23
PCU Registers (SCU)					
VDESR	Voltage Detector Status Register	0000_H	U, PV	U, PV	Page 12-24
CCU Registers (SCU)					
CLKCR	Clock Control	0000_H	U, PV	U, PV, BP	Page 12-25
PWRSVCR	Power Save Control Register	0004_H	U, PV	U, PV	Page 12-27
CGATSTAT0	Clock Gating Status for Peripherals 0	0008_H	U, PV	U, PV	Page 12-27
CGATSET0	Clock Gating Set for Peripherals 0	$000C_H$	U, PV	U, PV, BP	Page 12-28
CGATCLR0	Clock Gating Clear for Peripherals 0	0010_H	U, PV	U, PV, BP	Page 12-29
OSCCSR	Oscillator Control and Status Register	0014_H	U, PV	U, PV	Page 12-31
RCU Registers (SCU)					
RSTSTAT	Reset Status	0000_H	U, PV	BE	Page 12-32
RSTSET	Reset Set Register	0004_H	U, PV	U, PV	Page 12-33
RSTCLR	Reset Clear Register	0008_H	U, PV	U, PV	Page 12-34
RSTCON	Reset Control Register	$000C_H$	U, PV	U, PV	Page 12-35
GCU Registers (SCU)					
ID	Module Identification Register	0008_H	U, PV	BE	Page 12-36
IDCHIP	Chip ID	0004_H	U, PV	SP	Page 12-37
DBGROMID	DBGROMID	0000_H	U, PV	SP	Page 12-37
SSW0	SSW Support Register	0014_H	U, PV	U, PV	Page 12-38

Table 12-6 Registers Overview (cont'd)

Short Name	Description	Offset Addr.[1]	Access Mode		Description See
			Read	Write	
CCUCON	CCUx Global Start Control Register	0030_H	U, PV	U, PV	Page 12-38
SRRAW	RAW Service Request Status	0038_H	U, PV	BE	Page 12-39
SRMSK	Service Request Mask	$003C_H$	U, PV	U, PV	Page 12-41
SRCLR	Service Request Clear	0040_H	U, PV	U, PV	Page 12-43
SRSET	Service Request Set	0044_H	U, PV	U, PV	Page 12-45
PASSWD	Bit protection Register	0024_H	U, PV	U, PV	Page 12-48
MIRRSTS	Mirror Update Status Register	0048_H	U, PV	BE	Page 12-49
PMTSR	Parity Memory Test Select Register	0054_H	U, PV	U, PV	Page 12-50

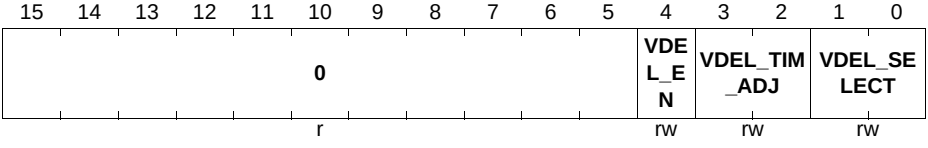
1) The absolute register address is calculated as follows:
Module Base Address + Sub-Module Offset Address + Offset Address (shown in this column)

12.9.1 PCU Registers (ANACTRL)

ANAVDEL

Voltage Detector Control register.

ANAVDEL
Voltage Detector Control Register (1050_H) **Reset Value:$001C_H$**



Field	Bits	Type	Description
VDEL_SELECT	1:0	rw	**VDEL Range Select** With these bits the VDDP range is set. 00B 2.25V 01B 3.0V 10B 4.4V

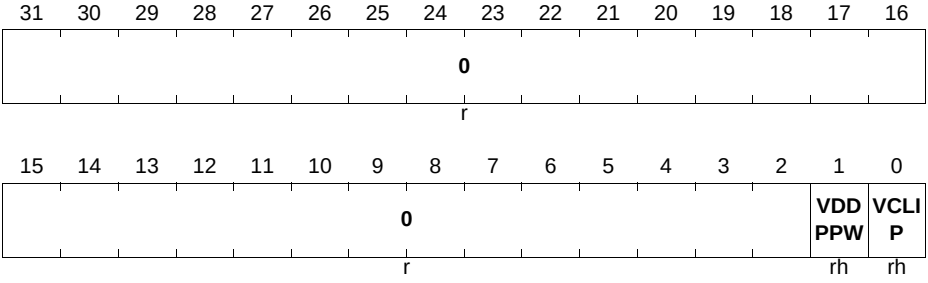
Field	Bits	Type	Description
VDEL_TIM_ADJ	3:2	rw	**VDEL Timing Setting** These bits control the reaction speed of the VDEL. The value is determined by characterisation. 00B typ 1µs - slowest response time 01B typ 500n 10B typ 250n 11B no delay - fastest response time.
VDEL_EN	4	rw	**VDEL unit Enable** 0B VDEL is disabled 1B VDEL is active
0	15:5	r	**Reserved** Read as 0; should be written with 0.

12.9.2 PCU Registers (SCU)

VDESR

Voltage Detector status register.

VDESR
Voltage Detector Status Register (0200_H) Reset Value: $0000\ 0000_H$



Field	Bits	Type	Description
VCLIP	0	rh	**VCLIP Indication** VCLIP monitoring bit. 0_B VCLIP is not active 1_B VCLIP is active

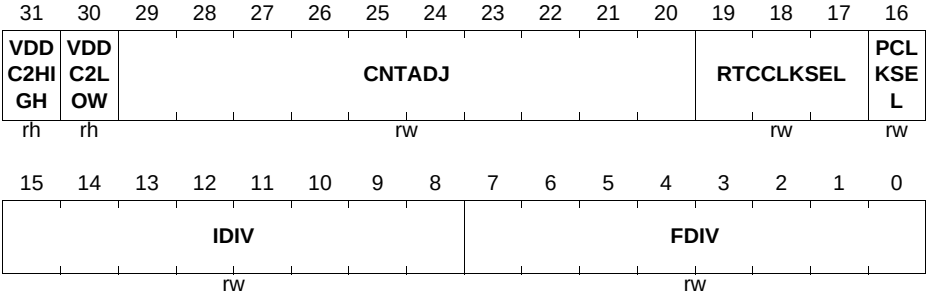
Field	Bits	Type	Description
VDDPPW	1	rh	**VDDPPW Indication** 0_B VDDP is above pre-warning threshold 1_B VDDP is below pre-warningthreshold
0	[31:2]	r	**Reserved** Read as 0; should be written with 0.

12.9.3 CCU Registers (SCU)

CLKCR

Clock control register.

CLKCR
Clock Control Register **(0300_H)** **Reset Value: 3FF0 0400_H**



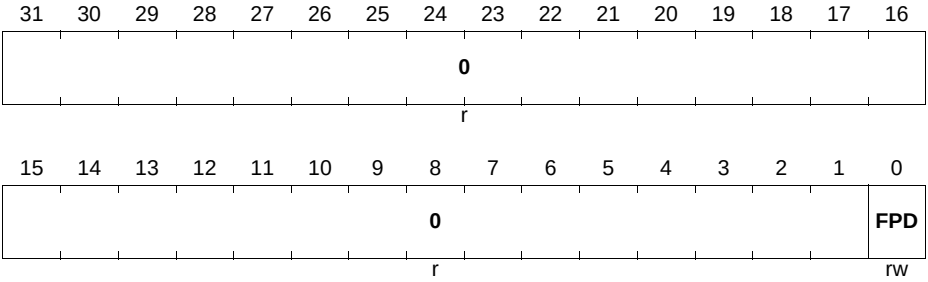
Field	Bits	Type	Function
FDIV	[7:0]	rw	**Fractional Divider Selection** Selects the fractional divider to be n/256, where n is the value of FDIV and is in the range of 0 to 255. For example, writing 0001_B to FDIV selects the fractional divider to be 1/256. This bit is protected by the bit protection scheme as described in Memory Organization chapter *Note: Fractional divider has no effect if IDIV = 00_H .*

Field	Bits	Type	Function
IDIV	[15:8]	rw	**Divider Selection** 00_H Divider is bypassed. 01_H 1; MCLK = 32 MHz 02_H 2; MCLK = 16 MHz 03_H 3; MCLK = 10.67 MHz 04_H 4; MCLK = 8 MHz FE_H 254; MCLK = 126 kHz FF_H 255; MCLK = 125.5 kHz This bit is protected by the bit protection scheme as described in Memory Organization chapter
PCLKSEL	16	rw	**PCLK Clock Select** 0_B PCLK = MCLK 1_B PCLK = 2 x MCLK This bit is protected by the bit protection scheme as described in Memory Organization chapter
RTCCLKSEL	[19:17]	rw	**RTC Clock Select** 000_B 32.768kHz standby clock Others Reserved This bit is protected by the bit protection scheme as described in Memory Organization chapter
CNTADJ	[29:20]	rw	**Counter Adjustment** 000_H 1 clock cycles of the DCO1, 64MHz clock 001_H 2 clock cycles of the DCO1, 64MHz clock 002_H 3 clock cycles of the DCO1, 64MHz clock 003_H 4 clock cycles of the DCO1, 64MHz clock 004_H 5 clock cycles of the DCO1, 64MHz clock $3FE_H$ 1023 clock cycles of the DCO1, 64MHz clock $3FF_H$ 1024 clock cycles of the DCO1, 64MHz clock
VDDC2LOW	30	rh	**VDDC too low** 0_B VDDC is not too low and the fractional divider input clock is running at the targeted frequency 1_B VDDC is too low and the fractional divider input clock is not running at the targeted frequency
VDDC2HIGH	31	rh	**VDDC too high** 0_B VDDC is not too high 1_B VDDC is too high

PWRSVCR

Configuration register that defines some system behaviour aspects while in deep-sleep mode or sleep mode. The original system state gets restored upon wakeup from sleep mode or deep-sleep mode.

PWRSVCR
Power Save Control Register (0304_H) Reset Value: $0000\ 0000_H$



Field	Bits	Type	Description
FPD	0	rw	**Flash Power Down** 0_B no effect 1_B Flash power down when entering power save mode. Upon wake-up, CPU is able to fetch code from flash.
0	[31:1]	r	**Reserved** Read as 0.

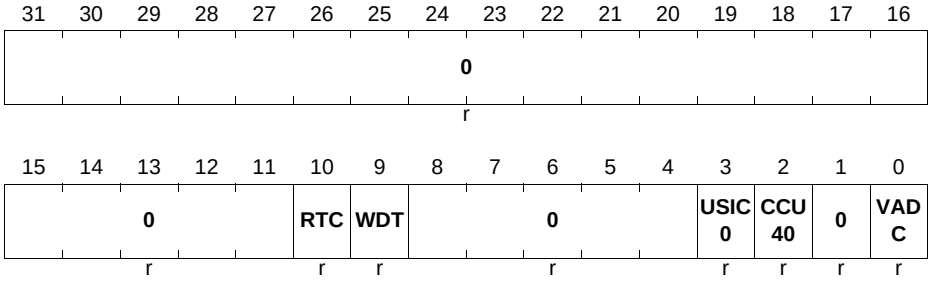
CGATSTAT0

Clock gating status for XMC1100 peripherals. After reset, all peripherals as listed in the registers are not running. Their module clock are gated.

Every bit in this register is protected by the bit protection scheme as described in Memory Organization chapter.

CGATSTAT0

Peripheral 0 Clock Gating Status **(0308_H)** **Reset Value: 0000 07FF$_H$**



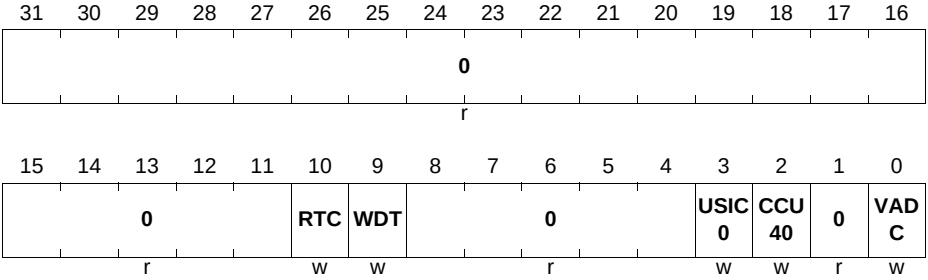
Field	Bits	Type	Description
VADC	0	r	**VADC and SHS Gating Status** 0_B gating de-asserted 1_B gating asserted
CCU40	2	r	**CCU40 Gating Status** 0_B gating de-asserted 1_B gating asserted
USIC0	3	r	**USIC0 Gating Status** 0_B gating de-asserted 1_B gating asserted
WDT	9	r	**WDT Gating Status** 0_B gating de-asserted 1_B gating asserted
RTC	10	r	**RTC Gating Status** 0_B gating de-asserted 1_B gating asserted
0	1, [8:4], [31:11]	r	**Reserved**

CGATSET0

Clock gating enable for XMC1100 peripherals. Write one to selected bit to enable gating of corresponding clock, writing zeros has no effect.

Every bit in this register is protected by the bit protection scheme as described in Memory Organization chapter.

CGATSET0
Peripheral 0 Clock Gating Set **($030C_H$)** **Reset Value: $0000\ 0000_H$**

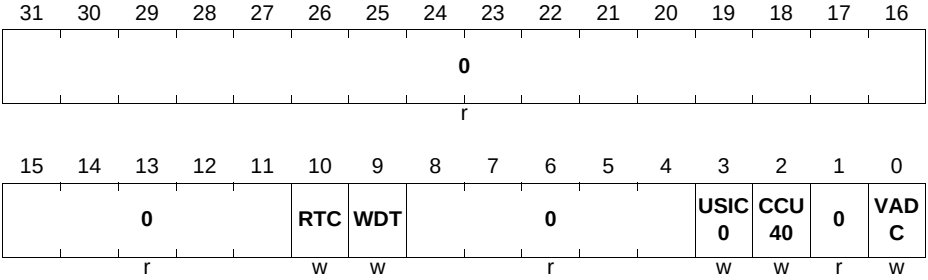


Field	Bits	Type	Description
VADC	0	w	**VADC and SHS Gating Set** 0_B no effect 1_B enable gating
CCU40	2	w	**CCU40 Gating Set** 0_B no effect 1_B enable gating
USIC0	3	w	**USIC0 Gating Set** 0_B no effect 1_B enable gating
WDT	9	w	**WDT Gating Set** 0_B no effect 1_B enable gating
RTC	10	w	**RTC Gating Set** 0_B no effect 1_B enable gating
0	1, [8:4], [31:11]	r	**Reserved**

CGATCLR0

Clock gating disable for XMC1100 peripherals. Write one to selected bit to disable gating of corresponding clock, writing zeros has no effect.

CGATCLR0
Peripheral 0 Clock Gating Clear **(0310_H)** **Reset Value: $0000\ 0000_H$**



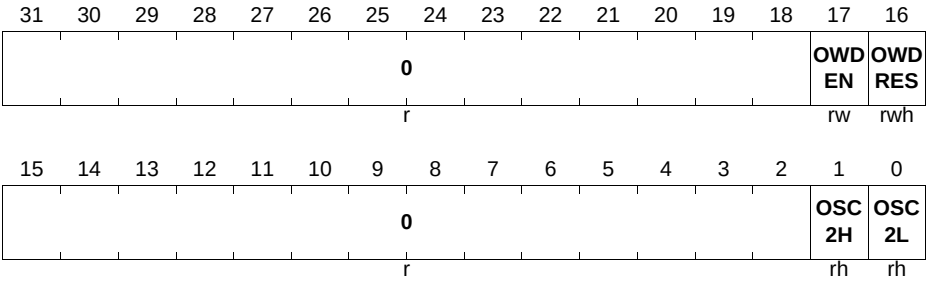
Field	Bits	Type	Description
VADC	0	w	**VADC and SHS Gating Clear** 0_B no effect 1_B disable gating
CCU40	2	w	**CCU40 Gating Clear** 0_B no effect 1_B disable gating
USIC0	3	w	**USIC0 Gating Clear** 0_B no effect 1_B disable gating
WDT	9	w	**WDT Gating Clear** 0_B no effect 1_B disable gating
RTC	10	w	**RTC Gating Clear** 0_B no effect 1_B disable gating
0	1, [8:4], [31:11]	r	**Reserved**

OSCCSR

Oscillator Control and Status Register.

OSCCSR
Oscillator Control and Status Register

(0314_H) **Reset Value: 0000 000X_H**



Field	Bits	Type	Description
OSC2L	0	rh	**Oscillator Valid Low Status Bit** This bit indicates if the frequency output of OSC is usable. This is checked by the Oscillator Watchdog 0_B The OSC frequency is usable 1_B The OSC frequency is not usable. Frequency is too low.
OSC2H	1	rh	**Oscillator Valid High Status Bit** This bit indicates if the frequency output of OSC is usable. This is checked by the Oscillator Watchdog. 0_B The OSC frequency is usable 1_B The OSC frequency is not usable. Frequency is too high.
OWDRES	16	rwh	**Oscillator Watchdog Reset** Setting this bit will restart the oscillator detection. This bit will be automatically reset to 0 after OWD is reset which takes 2 standby clock cycles due to synchronisation. 0_B The Oscillator Watchdog is not cleared and remains active 1_B The Oscillator Watchdog is cleared and restarted. The OSC2L and OSC2H flag will be held in the last value until it is updated after 3 standby clock cycles.

Field	Bits	Type	Description
OWDEN	17	rw	**Oscillator Watchdog Enable** 0_B The Oscillator Watchdog is disabled 1_B The Oscillator Watchdog is enabled *Note: OSC2H and OSC2L will be cleared to 0 when OWD is disabled.*
0	[15:2], [31:18]	r	**Reserved** Read as 0.

12.9.4 RCU Registers (SCU)

RSTSTAT

Reset status register. This register needs to be checked after system startup in order to determine last reset reason. User should clear this register after reading it to ensure a clear status when the next reset happen. This register is reset by a power-on reset.

RSTSTAT
RCU Reset Status **(0400_H)** **Reset Value: 0000 0XXX$_H$**

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

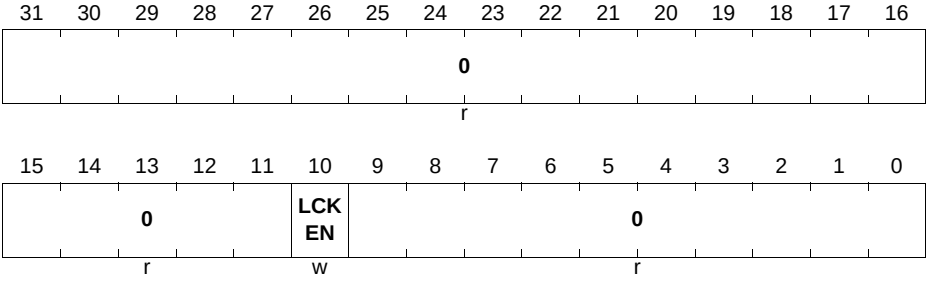
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0					LCK EN	RSTSTAT									
r					r	rh									

Field	Bits	Type	Description
RSTSTAT	[9:0]	rh	**Reset Status Information** Provides reason of last reset 0000000001_B Power on reset or Brownout reset $XXXXXXXX1X_B$ Master reset via bit RSTCON.MRSTEN $XXXXXXX1XX_B$ CPU system reset request $XXXXXX1XXX_B$ CPU lockup reset $XXXXX1XXXX_B$ Flash ECC reset $XXXX1XXXXX_B$ WDT reset $XXX1XXXXXX_B$ Loss of clock reset $XX1XXXXXXX_B$ Parity Error reset
LCKEN	10	r	**Enable Lockup Status** 0_B Reset by Lockup disabled 1_B Reset by Lockup enabled
0	[31:11]	r	**Reserved**

RSTSET

Selective configuration of reset behaviour in the system. Write one to set selected bit, writing zeros has no effect.

RSTSET
RCU Reset Set Register **(0404_H)** **Reset Value: $0000\ 0000_H$**



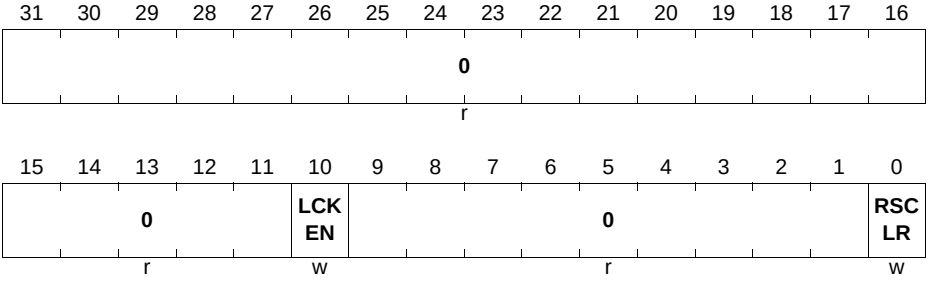
Field	Bits	Type	Description
LCKEN	10	w	**Enable Lockup Reset** 0_B no effect 1_B Enable reset when Lockup gets asserted

Field	Bits	Type	Description
0	[9:0], [31:11]	r	**Reserved**

RSTCLR

Selective configuration of reset behaviour in the system. Write one to clear selected bit, writing zeros has no effect.

RSTCLR
RCU Reset Clear Register **(0408_H)** **Reset Value: $0000\ 0000_H$**

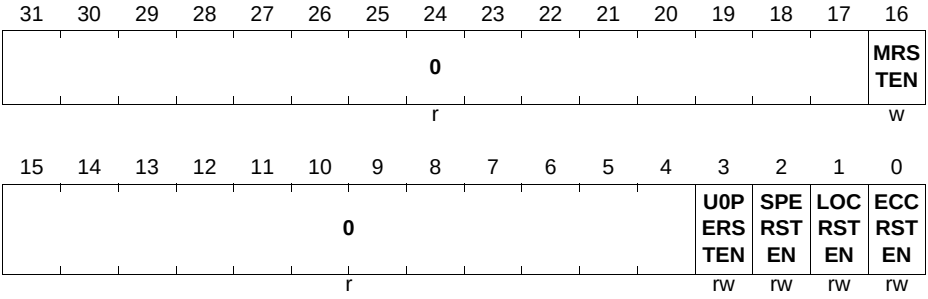


Field	Bits	Type	Description
RSCLR	0	w	**Clear Reset Status** 0_B no effect 1_B Clears field **RSTSTAT.RSTSTAT**
LCKEN	10	w	**Enable Lockup Reset** 0_B no effect 1_B Disable reset when Lockup gets asserted
0	[9:1], [31:11]	r	**Reserved**

RSTCON

Enabling of reset triggered by critical events. It is reset by any reset type.

RSTCON
RCU Reset Control Register **($040C_H$)** **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
ECCRSTEN	0	rw	**Enable ECC Error Reset** 0_B No reset when ECC double bit error occur 1_B Reset when ECC double bit error occur
LOCRSTEN	1	rw	**Enable Loss of Clock Reset** 0_B No reset when loss of clock occur 1_B Reset when loss of clock occur
SPERSTEN	2	rw	**Enable 16kbytes SRAM Parity Error Reset** 0_B No reset when SRAM parity error occur 1_B Reset when SRAM parity error occur
U0PERSTEN	3	rw	**Enable USIC0 SRAM Parity Error Reset** 0_B No reset when USIC0 memory parity error occur 1_B Reset when USIC0 memory parity error occur
MRSTEN	16	w	**Enable Master Reset** 0_B No effect 1_B Triggered Master reset
0	[15:4], [31:17]	r	**Reserved**

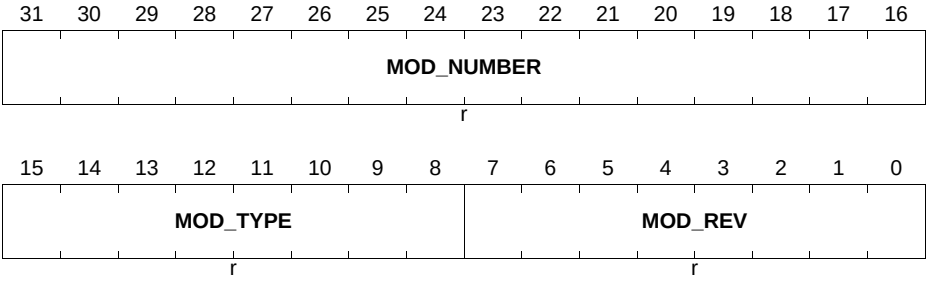
12.9.5 GCU Registers (SCU)

ID

Register containing uniques ID of the module.

ID
SCU Module ID Register

(0008_H) **Reset Value: 00F1 C0XX$_H$**

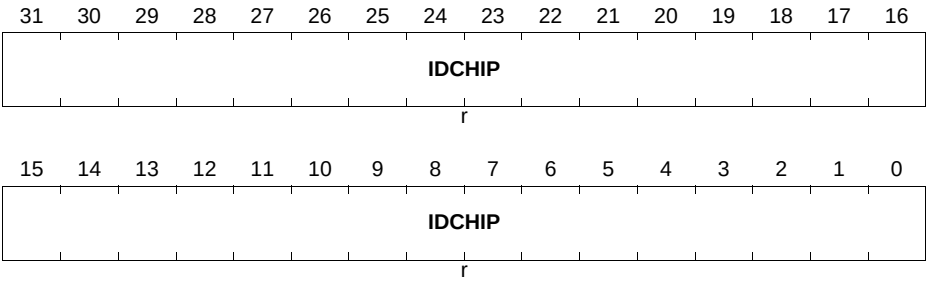


Field	Bits	Type	Description
MOD_REV	[7:0]	r	**Module Revision Number** MOD_REV defines the revision number. The value of a module revision starts with 01_H (first revision).
MOD_TYPE	[15:8]	r	**Module Type** This bit field is $C0_H$. It defines the module as a 32-bit module.
MOD_NUMBER	[31:16]	r	**Module Number Value** This bit field defines the module identification number.

IDCHIP

Register containing a unique ID of the chip in the XMC family. The value of this register formed part of the chip identification number as described in **Chip Identification Number**.

IDCHIP
Chip ID Register **(0004_H)** **Reset Value: 0000 0000_H**

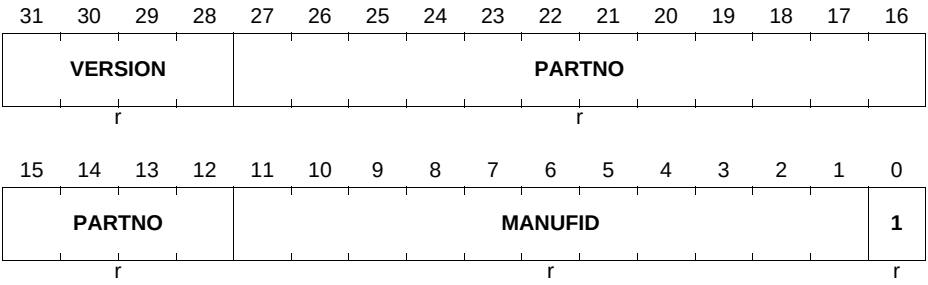


Field	Bits	Type	Description
IDCHIP	[31:0]	r	**CHIP ID** 0001 $1XXX_H$ XCM1100 0001 $XXX2_H$ temperature : -40 - 85 °C 0001 $XXX3_H$ temperature : -40 - 105 °C 0001 $XX1X_H$ TSSOP38 pin package 0001 $XX2X_H$ TSSOP28 pin package 0001 $XX3X_H$ TSSOP16 pin package Others Reserved

DBGROMID

Register containing unique manufactory ID, part number and the design stepping code of the chip.

DBGROMID
Debug System ROM ID Register **(0000_H)** **Reset Value: 101E D083$_H$**



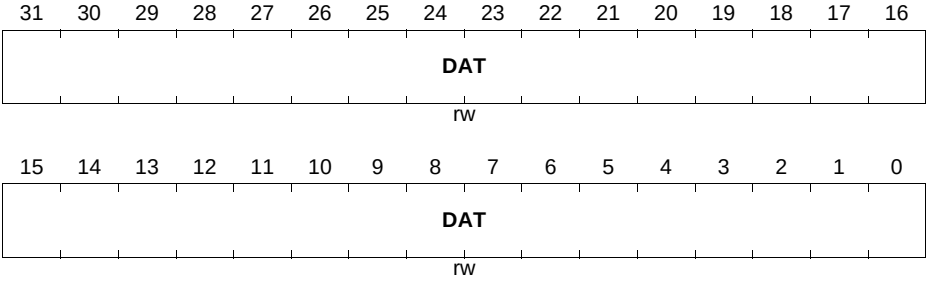
Field	Bits	Type	Description
MANUFID	[11:1]	r	**Manufactory Identity**
PARTNO	[27:12]	r	**Part Number**
VERSION	[31:28]	r	**Product version**
1	0	r	**Reserved** Read as 0; should be written with 0.

SSW0

Software support registers.

SSW0 is used to change the BMI value.

SSW0
SSW Register 0 **(0014$_H$)** **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
DAT	[31:0]	rw	**SSW Data** *Note: SSW registers can be reset with master reset only*

CCUCON

CAPCOM module control register.

CCUCON
CCU Control Register **(0030_H)** **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															GSC40
r															rw

Field	Bits	Type	Description
GSC40	0	rw	**Global Start Control CCU40** 0_B disable 1_B enable Note: Outgoing signal has to be generated with clk_cc
0	[31:1]	r	**Reserved** Read as 0; should be written with 0.

SRRAW

Service request status without masking. Write one to a bit in SRCLR register to clear a bit or SRSET to set a bit. Writing zero has no effect.

SRRAW
SCU Raw Service Request Status **(0038_H)** **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0			RTC_TIM1	RTC_TIM0	RTC_ATIM1	RTC_ATIM0	RTC_CTR	0	SBYCLKFI	VCLIPI	FLCMPLTI	FLECC2I	PEU0I	PESRAMI	LOCI
r			rh	rh	rh	rh	rh	r	rh	rh	rh	rh	rh	rh	rh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								VDROPI	0			VDDPI	AI	PI	PRWARN
r								rh	r			rh	rh	rh	rh

Field	Bits	Type	Description
PRWARN	0	rh	**WDT pre-warning Event Status Before Masking** 0_B Event has not occurred 1_B Event has occurred
PI	1	rh	**RTC Raw Periodic Event Status Before Masking** 0_B Event has not occurred 1_B Event has occurred
AI	2	rh	**RTC Raw Alarm Event Status Before Masking** 0_B Event has not occurred 1_B Event has occurred
VDDPI	3	rh	**VDDP pre-warning Event Status Before Masking** 0_B Event has not occurred 1_B Event has occurred
VDROPI	7	rh	**VDROP Event Status Before Masking** 0_B Event has not occurred 1_B Event has occurred
LOCI	16	rh	**Loss of Clock Event Status Before Masking** 0_B Event has not occurred 1_B Event has occurred
PESRAMI	17	rh	**16kbytes SRAM Parity Error Event Status Before Masking** 0_B Event has not occurred 1_B Event has occurred
PEU0I	18	rh	**USIC0 SRAM Parity Error Event Status Before Masking** 0_B Event has not occurred 1_B Event has occurred
FLECC2I	19	rh	**Flash Double Bit ECC Event Status Before Masking** 0_B Event has not occurred 1_B Event has occurred
FLCMPLTI	20	rh	**Flash Operation Complete Event Status Before Masking** 0_B Event has not occurred 1_B Event has occurred

Field	Bits	Type	Description
VCLIPI	21	rh	**VCLIP Event Status Before Masking** 0_B Event has not occurred 1_B Event has occurred
SBYCLKFI	22	rh	**Standby Clock Failure Event Status Before Masking** 0_B No standby clock failure has occurred 1_B Standby clock failure has occurred
RTC_CTR	24	rh	**RTC CTR Mirror Register Update Status Before Masking** 0_B not updated 1_B update completed
RTC_ATIM0	25	rh	**RTC ATIM0 Mirror Register Update Status Before Masking** 0_B not updated 1_B update completed
RTC_ATIM1	26	rh	**RTC ATIM1 Mirror Register Update Status Before Masking** 0_B not updated 1_B update completed
RTC_TIM0	27	rh	**RTC TIM0 Mirror Register Update Before Masking** 0_B not updated 1_B update completed
RTC_TIM1	28	rh	**RTC TIM1 Mirror Register Update Status Before Masking** 0_B not updated 1_B update completed
0	[6:4], [15:8], 23, [31:29]	r	**Reserved**

SRMSK

Service request mask used to mask outputs of RAW register. When the bit is set to 1, an interrupt or service request will be triggered when the event happens.

SRMSK
SCU Service Request Mask **($003C_H$)** **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0			RTC_TIM1	RTC_TIM0	RTC_ATIM1	RTC_ATIM0	RTC_CTR	0	SBYCLKFI	VCLIPI	0	FLECC2I	PEU0I	PESRAMI	LOCI
r			rw	rw	rw	rw	rw	r	rw	rw	r	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								VDROPI	0			VDDPI	0		PRWARN
r								rw	r			rw	r		rw

Field	Bits	Type	Description
PRWARN	0	rw	**WDT pre-warning Interrupt Mask** 0_B disable interrupt 1_B enable interrupt
VDDPI	3	rw	**VDDP pre-warning Interrupt Mask** 0_B disable interrupt 1_B enable interrupt
VDROPI	7	rw	**VDROP Interrupt Mask** 0_B disable interrupt 1_B enable interrupt
LOCI	16	rw	**Loss of Clock Interrupt Mask** 0_B disable interrupt 1_B enable interrupt
PESRAMI	17	rw	**16kbytes SRAM Parity Error Interrupt Mask** 0_B disable interrupt 1_B enable interrupt
PEU0I	18	rw	**USIC0 SRAM Parity Error Interrupt Mask** 0_B disable interrupt 1_B enable interrupt
FLECC2I	19	rw	**Flash Double Bit ECC Interrupt Mask** 0_B disable interrupt 1_B enable interrupt
VCLIPI	21	rw	**VCLIP Interrupt Mask** 0_B disable interrupt 1_B enable interrupt

Field	Bits	Type	Description
SBYCLKFI	22	rw	**Standby Clock Failure Interrupt Mask** 0_B disable interrupt 1_B enable interrupt
RTC_CTR	24	rw	**RTC CTR Mirror Register Update Mask** 0_B disable interrupt 1_B enable interrupt
RTC_ATIM0	25	rw	**RTC ATIM0 Mirror Register Update Mask** 0_B disable interrupt 1_B enable interrupt
RTC_ATIM1	26	rw	**RTC ATIM1 Mirror Register Update Mask** 0_B disable interrupt 1_B enable interrupt
RTC_TIM0	27	rw	**RTC TIM0 Mirror Register Update Mask** 0_B disable interrupt 1_B enable interrupt
RTC_TIM1	28	rw	**RTC TIM1 Mirror Register Update Mask** 0_B disable interrupt 1_B enable interrupt
0	[2:1], [6:4], [15:8], 20, 23, [31:29]	r	**Reserved**

SRCLR

Clear service request bits of register SRRAW. Write one to clear corresponding bits. Writing zeros has no effect.

SRCLR
SCU Service Request Clear **(0040_H)** **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0			RTC_TIM1	RTC_TIM0	RTC_ATIM1	RTC_ATIM0	RTC_CTR	0	SBYCLKFI	VCLIPI	FLCMPLTI	FLECC2I	PEU0I	PESRAMI	LOCI
r			w	w	w	w	w	r	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								VDROPI	0			VDDPI	AI	PI	PRWARN
r								w	r			w	w	w	w

Field	Bits	Type	Description
PRWARN	0	w	**WDT pre-warning Interrupt Clear** 0_B no effect 1_B clear status bit in the raw status register
PI	1	w	**RTC Periodic Interrupt Clear** 0_B no effect 1_B clear status bit in the raw status register
AI	2	w	**RTC Alarm Interrupt Clear** 0_B no effect 1_B clear status bit in the raw status register
VDDPI	3	w	**VDDP pre-warning Interrupt Clear** 0_B no effect 1_B clear status bit in the raw status register
VDROPI	7	w	**VDROP Interrupt Clear** 0_B no effect 1_B clear status bit in the raw status register
LOCI	16	w	**Loss of Clock Interrupt Clear** 0_B no effect 1_B clear status bit in the raw status register
PESRAMI	17	w	**16kbytes SRAM Parity Error Interrupt Clear** 0_B no effect 1_B clear status bit in the raw status register
PEU0I	18	w	**USIC0 SRAM Parity Error Interrupt Clear** 0_B no effect 1_B clear status bit in the raw status register

Field	Bits	Type	Description
FLECC2I	19	w	**Flash Double Bit ECC Interrupt Clear** 0_B no effect 1_B clear status bit in the raw status register
FLCMPLTI	20	w	**Flash Operation Complete Interrupt Clear** 0_B no effect 1_B clear status bit in the raw status register
VCLIPI	21	w	**VCLIP Interrupt Clear** 0_B no effect 1_B clear status bit in the raw status register
SBYCLKFI	22	w	**Standby Clock Failure Interrupt Clear** 0_B no effect 1_B clear status bit in the raw status register
RTC_CTR	24	w	**RTC CTR Mirror Register Update Clear** 0_B no effect 1_B clear status bit in the raw status register
RTC_ATIM0	25	w	**RTC ATIM0 Mirror Register Update Clear** 0_B no effect 1_B clear status bit in the raw status register
RTC_ATIM1	26	w	**RTC ATIM1 Mirror Register Update Clear** 0_B no effect 1_B clear status bit in the raw status register
RTC_TIM0	27	w	**RTC TIM0 Mirror Register Update Clear** 0_B no effect 1_B clear status bit in the raw status register
RTC_TIM1	28	w	**RTC TIM1 Mirror Register Update Clear** 0_B no effect 1_B clear status bit in the raw status register
0	[6:4], [15:8], 23, [31:29]	r	**Reserved**

SRSET

Set service request fits of register SRRAW. Write one to set corresponding bits. Writing zeros has no effect.

SRSET
SCU Service Request Set **(0044_H)** **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0			RTC_TIM1	RTC_TIM0	RTC_ATIM1	RTC_ATIM0	RTC_CTR	0	SBYCLKFI	VCLIPI	FLCMPLTI	FLECC2I	PEU0I	PESRAMI	LOCI
r			w	w	w	w	w	r	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								VDROPI	0			VDDPI	AI	PI	PRWARN
r								w	r			w	w	w	w

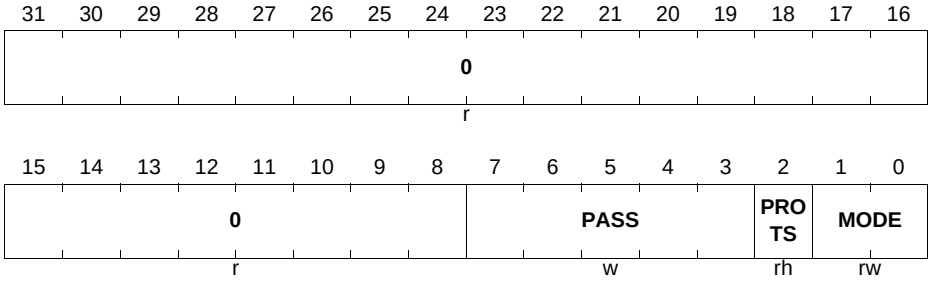
Field	Bits	Type	Description
PRWARN	0	w	**WDT pre-warning Interrupt Set** 0_B no effect 1_B set status bit in the raw status register
PI	1	w	**RTC Periodic Interrupt Set** 0_B no effect 1_B set status bit in the raw status register
AI	2	w	**RTC Alarm Interrupt Set** 0_B no effect 1_B set status bit in the raw status register
VDDPI	3	w	**VDDP pre-warning Interrupt Set** 0_B no effect 1_B set status bit in the raw status register
VDROPI	7	w	**VDROP Interrupt Set** 0_B no effect 1_B set status bit in the raw status register
LOCI	16	w	**Loss of Clock Interrupt Set** 0_B no effect 1_B set status bit in the raw status register
PESRAMI	17	w	**16kbytes SRAM Parity Error Interrupt Set** 0_B no effect 1_B set status bit in the raw status register
PEU0I	18	w	**USIC0 SRAM Parity Error Interrupt Set** 0_B no effect 1_B set status bit in the raw status register

Field	Bits	Type	Description
FLECC2I	19	w	**Flash Double Bit ECC Interrupt Set** 0_B no effect 1_B set status bit in the raw status register
FLCMPLTI	20	w	**Flash Operation Complete Interrupt Set** 0_B no effect 1_B set status bit in the raw status register
VCLIPI	21	w	**VCLIP Interrupt Set** 0_B no effect 1_B set status bit in the raw status register
SBYCLKFI	22	w	**Standby Clock Failure Interrupt Set** 0_B no effect 1_B set status bit in the raw status register
RTC_CTR	24	w	**RTC CTR Mirror Register Update Set** 0_B no effect 1_B set status bit in the raw status register
RTC_ATIM0	25	w	**RTC ATIM0 Mirror Register Update Set** 0_B no effect 1_B set status bit in the raw status register
RTC_ATIM1	26	w	**RTC ATIM1 Mirror Register Update Set** 0_B no effect 1_B set status bit in the raw status register
RTC_TIM0	27	w	**RTC TIM0 Mirror Register Update Set** 0_B no effect 1_B set status bit in the raw status register
RTC_TIM1	28	w	**RTC TIM1 Mirror Register Update Set** 0_B no effect 1_B set status bit in the raw status register
0	[6:4], [15:8], 23, [31:29]	r	**Reserved**

PASSWD

The PASSWD register is used to control the bit protection scheme.

PASSWD
Password Register **(0024_H)** **Reset Value: 0000 0007$_H$**

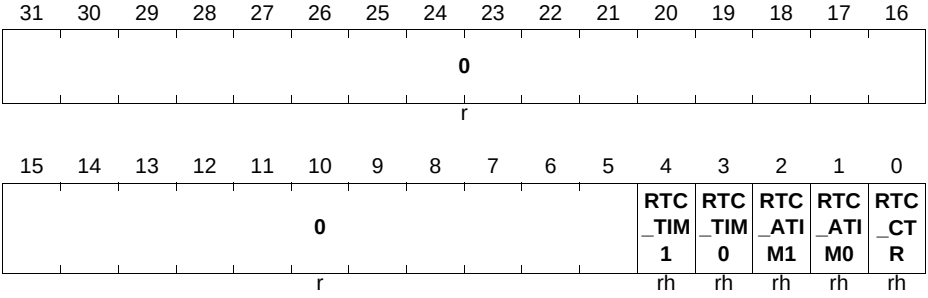


Field	Bits	Type	Description
MODE	[1:0]	rw	**Bit Protection Scheme Control Bits** 00_B Scheme disabled - direct access to the protected bits is allowed. 11_B Scheme enabled - the bit field PASS has to be written with the passwords to open and close the access to the protected bits. (Default) Others: Scheme enabled, similar to the setting for MODE = 11_B. These two bits cannot be written directly. To change the value between 11_B and 00_B, the bit field PASS must be written with 11000_B. Only then will the MODE bit field be registered.
PROTS	2	rh	**Bit Protection Signal Status Bit** This bit shows the status of the protection. 0_B Software is able to write to all protected bits. 1_B Software is unable to write to any of the protected bits.
PASS	[7:3]	w	**Password Bits** This bit protection scheme only recognizes the following three passwords: 11000_BEnables writing of the bit field MODE. 10011_BOpens access to writing of all protected bits. 10101_BCloses access to writing of all protected bits.
0	[31:8]	r	**Reserved**

MIRRSTS

Mirror status register for control of communication between SCU and RTC.

MIRRSTS
Mirror Update Status Register **(0048_H)** **Reset Value: 0000 0000$_H$**

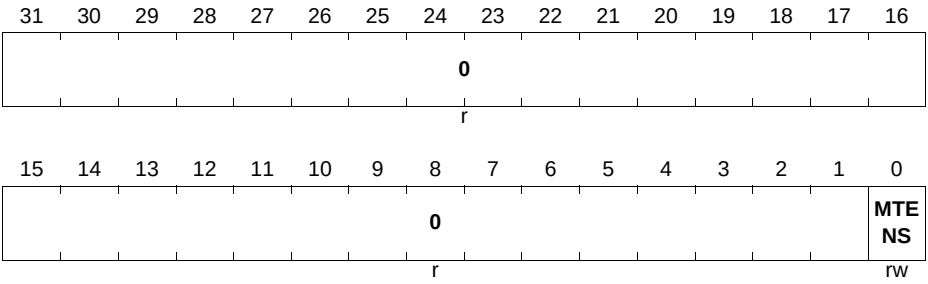


Field	Bits	Type	Function
RTC_CTR	0	rh	**RTC CTR Mirror Register Update Status** 0_B no update pending 1_B update pending
RTC_ATIM0	1	rh	**RTC ATIM0 Mirror Register Update Status** 0_B no update pending 1_B update pending
RTC_ATIM1	2	rh	**RTC ATIM1 Mirror Register Update Status** 0_B no update pending 1_B update pending
RTC_TIM0	3	rh	**RTC TIM0 Mirror Register Update Status** 0_B no update pending 1_B update pending
RTC_TIM1	4	rh	**RTC TIM1 Mirror Register Update Status** 0_B no update pending 1_B update pending
0	[31:14]	r	**Reserved** Read as 0; should be written with 0.

PMTSR

This register selects parity test output from a memory instance.

PMTSR
Parity Memory Test Select Register (0054$_H$) **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
MTENS	0	rw	**Parity Test Enable Control for 16kbytes SRAM** Controls the test multiplexer for the 16kbytes SRAM. 0_B standard operation 1_B generate an inverted parity bit during a write operation
0	31:1]	r	**Reserved** Should be written with 0.

13 Pseudo Random Number Generator

This chapter describes the Pseudo Random Number Generator (PRNG) module. It contains the following sections:

13.1 Introduction

The pseudo random bit generator (PRNG) provides random data with fast generation times.

13.2 Description of Operation Modes

13.2.1 Key Loading Mode

Before the PRNG can be used it has to be initialized by the user software.

The key (seed) k of the PRNG is a bit string $k = (k_{n-1}, k_{n-2}, ..., k_2, k_1, k_0)$ of length n. A key length of 80 bits is recommended, although smaller or larger key lengths are possible. It is recommended to use chip individual seed values.

The initialization of the PRNG consists of two basic phases:

1. Key loading
2. Warm-up

Key loading is initialized by setting the bit **PRNG_CTRL**.KLD to "1". In the key loading mode, **PRNG_WORD** acts always as a 16 bit destination register. The p partial words W_i (0 <= i < p) of the key k = (W_{p-1}, ..., W_1, W_0), with W_i = (k_{15}, ..., k_1, k_0), are sequentially written to **PRNG_WORD** in the order W_0, W_1, ..., W_{p-1}. A useful seed size is 80 bits (i.e., 5 data words, p=5). Because the bits of the partial key word are sequentially loaded into the internal state of the PRNG, loading of a key word will take 16 clock cycles. The **PRNG_CHK**.RDV flag is set to "0" while loading is in progress. A flag value of "1" indicates that the next partial key word can be written to **PRNG_WORD**.

After the complete key has been loaded, the **PRNG_CTRL**.KLD flag must be set to "0" to prepare the following warm-up phase. This operation takes one clock cycle.

The warm-up phase provides a thorough diffusion of the key bits. For this purpose the user must read and discard 64 random bits from the register **PRNG_WORD**. The random data output block must be set to either b = 8 or 16 bits. This is achieved by setting the corresponding value of the **PRNG_CTRL**.RDBS field.

The flag **PRNG_CHK**.RDV set to "1" indicates that the next random data block of width b can be read from **PRNG_WORD**.

If, for any reason, **PRNG_CTRL**.RDBS is reset to the default value of 00_B, the PRNG must be initialized once more – i.e., a key must be loaded and a warm-up phase carried out.

13.2.2 Streaming Mode

The flag **PRNG_CHK**.RDV set to 1_B indicates that the next random data block can be read from **PRNG_WORD**. After a word has been read the flag **PRNG_CHK**.RDV is reset to 0_B by the hardware and generation of new random bits starts. The PRNG requires 17–18 clock cycles to generate a 16 random bits and 9–10 clock cycles to generate eight random bits. From a software point of view it is not necessary to poll the **PRNG_CHK**.RDV flag. Consecutive read accesses to **PRNG_WORD** will be delayed automatically by hardware as long as **PRNG_CHK**.RDV is "0".

The width of the output data block is changed by setting the value of **PRNG_CTRL**.RDBS. This should be done before entering streaming mode, otherwise if the change is made during streaming, the new setting will not come into effect until the next generation cycle is started. The hardware checks that the selected number of bits are available and the flag **PRNG_CHK**.RDV is set when this condition is true.

In order to avoid reading a duplicate random byte when switching from 8-bit to 16-bit operation mode the last byte generated in the 8-bit mode should first be discarded before switching to 16-bit mode.

Note: PRNG_WORD should be accessed as 16 bit register, the upper 16 bits (of a 32 bit access) are ignored on a write and zeroes on a read and therefore contains no random data.

13.2.3 Refreshing and Restarting a Random Bit Stream

The random bit sequence can be refreshed with a new key. In this case the entropy contained in the last internal state is not cleared, but instead the new key is mixed into the current PRNG state. This is referred to as refreshing.

Refreshing is a means of introducing additional entropy into the generation of the random bit sequence. Without refreshing, the entire entropy rests in the inital key (or seed) that was used for initializing the PRNG during first key loading. Thereafter, the generation of the random bit sequence is purely deterministic. The deterministic process is broken whenever refreshing is performed. Refreshing implies that it is not possible to reproduce the same random result using the same key for data generation.

Since the internal state of the PRNG cannot be read and set directly, a sequence cannot be restored from any given state. A random bit sequence based on a certain initial key can, however, be continued. To this means, a segment s of the output sequence is stored and this segment is used as a initial key later on. The segment s = (s_{n-1}, s_{n-2}, …, s_2, s_1, s_0) of length **n** should be fresh – i.e., generated after the last bits used in the application. The length of **s** should be at least n = 80 bits. This way a pseudorandom bit sequence can be continued after a system reset without requiring new key material. This process is known as restarting.

Note: Integration in a multi-application/multitasking environment together with the requirement to obtain a reproducible pseudorandom bit sequence would necessitate a state saving and restoring feature. Hence the OS must be able to save the PRNG state before giving control to application 2 and restore the state before returning control to application 1. This is not possible with the PRNG module.

13.3 Debug Behavior

The PRNG does not support a suspend mode while the system is halted by the debugger. That means that the PRNG continues its operation during debug halt.

13.4 PRNG Registers

The interface of the pseudorandom number generator comprises the registers PRNG_WORD, PRNG_CHK and PRNG_CTRL.

Table 13-1 Registers Address Space

Module	Base Address	End Address	Note
PRNG	4802 0000$_H$	4802 000F$_H$	

Table 13-2 Registers Overview

Register Short Name	Register Long Name	Offset Address	Page Number
PRNG Registers, Data SFRs			
PRNG_WORD	Pseudo RNG word register	00$_H$	**Page 13-4**
PRNG_CHK	Pseudo RNG status check register	04$_H$	**Page 13-5**
PRNG Registers, Control SFRs			
PRNG_CTRL	Pseudo RNG control register	0C$_H$	**Page 13-7**

The register is addressed wordwise.

13.4.1 Data SFRs

Pseudo RNG Word Register

PRNG_WORD
Pseudo RNG Word Register (00$_H$) Reset Value: 0000$_H$

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

Field	Bits	Type	Description
RDATA	15:0	rw	**Random Data** Random bit block or key to load. In the streaming mode the range of valid random bits is defined by the settings given by PRNG_CTRL.RDSB and the value of the flag PRNG_CHK.RDV. In the key loading mode the seed value (key) is written via this register. In this case, the key is written in units of 16 bits.

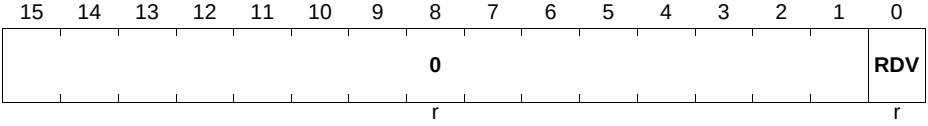
Additional Information

Notes

1. *Reading PRNG_WORD while the PRNG is running in key loading mode (configured by setting PRNG_CTRL.KLD) will return the last value written to the register, whereas write accesses to the register while the PRNG is running in streaming mode (PRNG_CTRL.KLD = '0') will be ignored.*
2. *Write access to SFR PRNG_WORD:*
   It is strongly recommended to wait until the SFR bit PRNG_CHK.RDV indicates that the register PRNG_WORD is ready to receive (new) data (which will be used to seed the PRNG.)
3. *Read access to SFR PRNG_WORD:*
   It is strongly recommended to check the SFR bit PRNG_CHK.RDV before reading SFR PRNG_WORD.

Pseudo RNG Status Check Register

PRNG_CHK
Pseudo RNG Status Check Register (04_H) **Reset Value: 0000_H**



Field	Bits	Type	Description
RDV	0	r	**Random Data / Key Valid Flag** 0_B **INV**, New random data block is not yet ready to be read. In **"Key Loading Mode" on Page 13-2**) this flag is set to 0_B while loading is in progress. 1_B **VAL**, Random data block is valid. In key loading mode this value indicates that the next partial key word can be written to **PRNG_WORD**.
0	15:1	r	**Reserved**

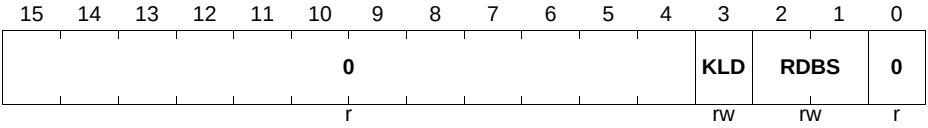
Additional Information

*Note: If a word or byte is read from **PRNG_WORD** although the data is not ready (**PRNG_CHK**.RDV = 0_B), the system stalls until the data becomes ready.*

13.4.2 Control SFRs

Pseudo RNG Control Register

PRNG_CTRL
Pseudo RNG Control Register **($0C_H$)** **Reset Value: 0000_H**



Field	Bits	Type	Description
KLD	3	rw	**Key Load Operation Mode** 0_B **STRM**, Streaming mode (default) 1_B **KLD**, Key loading mode
RDBS	2:1	rw	**Random Data Block Size** Set random data block size for read access (16 bits always used for key loading) 00_B **RES**, Reset state (no random data block size defined)[1], value of **PRNG_WORD** is undefined. 01_B **BYTE**, 8 bits in PRNG_WORD.RDATA[7:0] 10_B **WORD**, 16 bits in PRNG_WORD.RDATA[15:0] 11_B **RFU**, Reserved for future use, value of **PRNG_WORD** is undefined.
0	0	r	**Reserved**
0	15:4	r	**Reserved**

1) Once the reset value for RDBS (00_B) is set, the PRNG must be fully initialized before the functionality can be used. Initialization requires key loading, followed by a warm-up phase.

Additional Information

*Note: It is recommended not to change the **PRNG_CTRL**.KLD flag during key load. Otherwise the distribution of the bits of the PRNG will not be equal.*

14 Universal Serial Interface Channel (USIC)

The **U**niversal **S**erial **I**nterface **C**hannel module (USIC) is a flexible interface module covering several serial communication protocols. A USIC module contains two independent communication channels named USICx_CH0 and USICx_CH1, with x being the number of the USIC module (e.g. channel 0 of USIC module 0 is referenced as USIC0_CH0). The user can program during run-time which protocol will be handled by each communication channel and which pins are used.

References

The following documents are referenced for further information

[7] IIC Bus Specification (Philips Semiconductors v2.1)

[8] IIS Bus Specification (Philips Semiconductors June 5 1996 revision)

14.1 Overview

This section gives an overview about the feature set of the USIC.

14.1.1 Features

Each USIC channel can be individually configured to match the application needs, e.g. the protocol can be selected or changed during run time without the need for a reset. The following protocols are supported:

- **UART** (ASC, asynchronous serial channel)
  - Module capability: receiver/transmitter with max. baud rate f_{PB} / 4
  - Wide baud rate range down to single-digit baud rates
  - Number of data bits per data frame: 1 to 63
  - MSB or LSB first
- **LIN** Support by hardware (Local Interconnect Network)
  - Data transfers based on ASC protocol
  - Baud rate detection possible by built-in capture event of baud rate generator
  - Checksum generation under software control for higher flexibility
- **SSC/SPI** (synchronous serial channel with or without slave select lines)
  - Standard, Dual and Quad SPI format supported
  - Module capability: maximum baud rate f_{PB} / 2, limited by loop delay
  - Number of data bits per data frame 1 to 63, more with explicit stop condition
  - Parity bit generation supported
  - MSB or LSB first
- **IIC** (Inter-IC Bus)
  - Application baud rate 100 kbit/s to 400 kbit/s
  - 7-bit and 10-bit addressing supported
  - Full master and slave device capability

- **IIS** (infotainment audio bus)
  – Module capability: maximum baud rate f_{PB} / 2

Note: The real baud rates that can be achieved in a real application depend on the operating frequency of the device, timing parameters as described in the Data Sheet, signal delays on the PCB and timings of the peer device.

In addition to the flexible choice of the communication protocol, the USIC structure has been designed to reduce the system load (CPU load) allowing efficient data handling. The following aspects have been considered:

- **Data buffer capability**
  The standard buffer capability includes a double word buffer for receive data and a single word buffer for transmit data. This allows longer CPU reaction times (e.g. interrupt latency).
- **Additional FIFO buffer capability**
  In addition to the standard buffer capability, the received data and the data to be transmitted can be buffered in a FIFO buffer structure. The size of the receive and the transmit FIFO buffer can be programmed independently. Depending on the application needs, a total buffer capability of 64 data words can be assigned to the receive and transmit FIFO buffers of a USIC module (the two channels of the USIC module share the 64 data word buffer).
  In addition to the FIFO buffer, a bypass mechanism allows the introduction of high-priority data without flushing the FIFO buffer.
- **Transmit control information**
  For each data word to be transmitted, a 5-bit transmit control information has been added to automatically control some transmission parameters, such as word length, frame length, or the slave select control for the SPI protocol. The transmit control information is generated automatically by analyzing the address where the user software has written the data word to be transmitted (32 input locations = 2^5 = 5 bit transmit control information).
  This feature allows individual handling of each data word, e.g. the transmit control information associated to the data words stored in a transmit FIFO can automatically modify the slave select outputs to select different communication targets (slave devices) without CPU load. Alternatively, it can be used to control the frame length.
- **Flexible frame length control**
  The number of bits to be transferred within a data frame is independent of the data word length and can be handled in two different ways. The first option allows automatic generation of frames up to 63 bits with a known length. The second option supports longer frames (even unlimited length) or frames with a dynamically controlled length.
- **Interrupt capability**
  The events of each USIC channel can be individually routed to one of 6 service request outputs SR[5:0] available for each USIC module, depending on the

application needs. Furthermore, specific start and end of frame indications are supported in addition to protocol-specific events.

- **Flexible interface routing**
  Each USIC channel offers the choice between several possible input and output pins connections for the communications signals. This allows a flexible assignment of USIC signals to pins that can be changed without resetting the device.
- **Input conditioning**
  Each input signal is handled by a programmable input conditioning stage with programmable filtering and synchronization capability.
- **Baud rate generation**
  Each USIC channel contains its own baud rate generator. The baud rate generation can be based either on the internal module clock or on an external frequency input. This structure allows data transfers with a frequency that can not be generated internally, e.g. to synchronize several communication partners.
- **Transfer trigger capability**
  In master mode, data transfers can be triggered by events generated outside the USIC module, e.g. by an input pin or a timer unit (transmit data validation). This feature allows time base related data transmission.
- **Debugger support**
  The USIC offers specific addresses to read out received data without interaction with the FIFO buffer mechanism. This feature allows debugger accesses without the risk of a corrupted receive data sequence.

To reach a desired baud rate, two criteria have to be respected, the module capability and the application environment. The module capability is defined with respect to the module's input clock frequency, being the base for the module operation. Although the module's capability being much higher (depending on the module clock and the number of module clock cycles needed to represent a data bit), the reachable baud rate is generally limited by the application environment. In most cases, the application environment limits the maximum reachable baud rate due to driver delays, signal propagation times, or due to EMI reasons.

Note: Depending on the selected additional functions (such as digital filters, input synchronization stages, sample point adjustment, data structure, etc.), the maximum reachable baud rate can be limited. Please also take care about additional delays, such as (internal or external) propagation delays and driver delays (e.g. for collision detection in ASC mode, for IIC, etc.).

A block diagram of the USIC module/channel structure is shown in **Figure 14-1**.

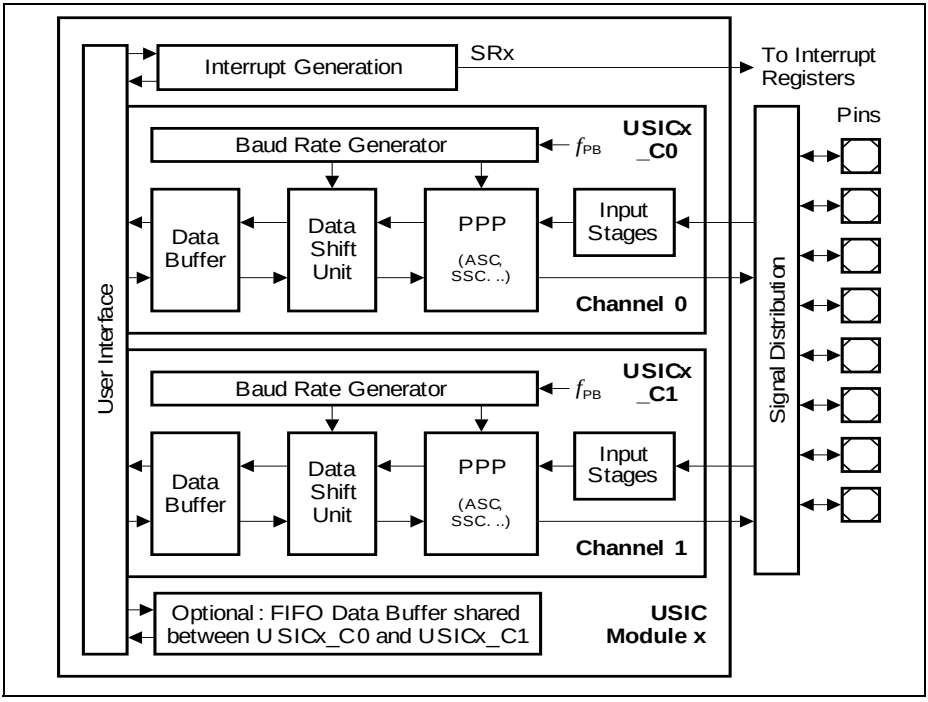


Figure 14-1 USIC Module/Channel Structure

14.2 Operating the USIC

This section describes how to operate the USIC communication channel.

14.2.1 USIC Structure Overview

This section introduces the USIC structure.

14.2.1.1 Channel Structure

The USIC module contains two independent communication channels, with a structure as shown in **Figure 14-1**.

The data shift unit and the data buffering of each channel support full-duplex data transfers. The protocol-specific actions are handled by the protocol pre-processors (PPP). In order to simplify data handling, an additional FIFO data buffer is optionally available for each USIC module to store transmit and receive data for each channel.

Due to the independent channel control and baud rate generation, the communication protocol, baud rate and the data format can be independently programmed for each communication channel.

14.2.1.2 Input Stages

For each protocol, the number of input signals used depends on the selected protocol. Each input signal is handled by an input stage (called DXn, where n=0-5) for signal conditioning, such as input selection, polarity control, or a digital input filter. They can be classified according to their meaning for the protocols, see **Table 14-1**.

The inputs marked as “optional” are not needed for the standard function of a protocol and may be used for enhancements. The descriptions of protocol-specific items are given in the related protocol chapters. For the external frequency input, please refer to the baud rate generator section, and for the transmit data validation, to the data handling section.

Table 14-1 Input Signals for Different Protocols

Selected Protocol	Shift Data Input(s) (handled by DX0, DX3, DX4 and DX5)[1)]	Shift Clock Input (handled by DX1)	Shift Control Input (handled by DX2)
ASC, LIN	RXD	optional: external frequency input or TXD collision detection	optional: transmit data validation
Standard SSC, SPI (Master)	DIN0 (MRST, MISO)	optional: external frequency input or delay compensation	optional: transmit data validation or delay compensation
Standard SSC, SPI (Slave)	DIN0 (MTSR, MOSI)	SCLKIN	SELIN
Dual-SSC, SPI (Master)	DIN[1:0] (MRST[1:0], MISO[1:0])	optional: external frequency input or delay compensation	optional: transmit data validation or delay compensation
Dual-SSC, SPI (Slave)	DIN[1:0] (MTSR[1:0], MOSI[1:0])	SCLKIN	SELIN
Quad-SSC, SPI (Master)	DIN[3:0] (MRST[3:0], MISO[3:0])	optional: external frequency input or delay compensation	optional: transmit data validation or delay compensation
Quad-SSC, SPI (Slave)	DIN[3:0] (MTSR[3:0], MOSI[3:0])	SCLKIN	SELIN
IIC	SDA	SCL	optional: transmit data validation
IIS (Master)	DIN0	optional: external frequency input or delay compensation	optional: transmit data validation or delay compensation
IIS (Slave)	DIN0	SCLKIN	WAIN

1) ASC, IIC, IIS and standard SSC protocols use only DX0 as the shift data input.

Note: To allow a certain flexibility in assigning required USIC input functions to port pins of the device, each input stage can select the desired input location among several possibilities.
The available USIC signals and their port locations are listed in the interconnects section, see ***Page 14-226****.*

14.2.1.3 Output Signals

For each protocol, up to 14 protocol-related output signals are available. The number of actually used outputs depends on the selected protocol. They can be classified according to their meaning for the protocols, see **Table 14-2**.

The outputs marked as “optional” are not needed for the standard function of a protocol and may be used for enhancements. The descriptions of protocol-specific items are given in the related protocol chapters. The MCLKOUT output signal has a stable frequency relation to the shift clock output (the frequency of MCLKOUT can be higher than for SCLKOUT) for synchronization purposes of a slave device to a master device. If the baud rate generator is not needed for a specific protocol (e.g. in SSC slave mode), the SCLKOUT and MCLKOUT signals can be used as clock outputs with 50% duty cycle with a frequency that can be independent from the communication baud rate.

Table 14-2 Output Signals for Different Protocols

Selected Protocol	Shift Data Output(s) DOUT[3:0]	Shift Clock Output SCLKOUT	Shift Control Outputs SELO[7:0]	Master Clock Output MCLKOUT
ASC, LIN	TXD	not used	not used	optional: master time base
Standard SSC, SPI (Master)	DOUT0 (MTSR, MOSI)	master shift clock	slave select, chip select	optional: master time base
Standard SSC, SPI (Slave)	DOUT0 (MRST, MISO)	optional: independent clock output	not used	optional: independent clock output
Dual-SSC, SPI (Master)	DOUT[1:0] (MTSR[1:0], MOSI[1:0])	master shift clock	slave select, chip select	optional: master time base
Dual-SSC, SPI (Slave)	DOUT[1:0] (MRST[1:0], MISO[1:0])	optional: independent clock output	not used	optional: independent clock output

Table 14-2 Output Signals for Different Protocols (cont'd)

Selected Protocol	Shift Data Output(s) DOUT[3:0]	Shift Clock Output SCLKOUT	Shift Control Outputs SELO[7:0]	Master Clock Output MCLKOUT
Quad-SSC, SPI (Master)	DOUT[3:0] (MTSR[3:0], MOSI[3:0])	master shift clock	slave select, chip select	optional: master time base
Quad-SSC, SPI (Slave)	DOUT[3:0] (MRST[3:0], MISO[3:0])	optional: independent clock output	not used	optional: independent clock output
IIC	SDA	SCL	not used	optional: master time base
IIS (master)	DOUT0	master shift clock	WA	optional: master time base
IIS (slave)	DOUT0	optional: independent clock output	not used	optional: independent clock output

Note: To allow a certain flexibility in assigning required USIC output functions to port pins of the device, most output signals are made available on several port pins. The port control itself defines pin-by-pin which signal is used as output signal for a port pin (see port chapter).
The available USIC signals and their port locations are listed in the interconnects section, see ***Page 14-226****.*

14.2.1.4 Baud Rate Generator

Each USIC Channel contains a baud rate generator structured as shown in **Figure 14-2**. It is based on coupled divider stages, providing the frequencies needed for the different protocols. It contains:

- A fractional divider to generate the input frequency $f_{PIN} = f_{FD}$ for baud rate generation based on the internal system frequency f_{PB}.
- The DX1 input to generate the input frequency $f_{PIN} = f_{DX1}$ for baud rate generation based on an external signal.
- Two protocol-related counters: the divider mode counter to provide the master clock signal MCLK, the shift clock signal SCLK, and other protocol-related signals; and the capture mode timer for time interval measurement, e.g. baud rate detection.
- A time quanta counter associated to the protocol pre-processor defining protocol-specific timings, such shift control signals or bit timings, based on the input frequency f_{CTQIN}.

- The output signals MCLKOUT and SCLKOUT of the protocol-related divider that can be made available on pins. In order to adapt to different applications, some output characteristics of these signals can be configured.
  For device-specific details about availability of USIC signals on pins please refer to the interconnects section.

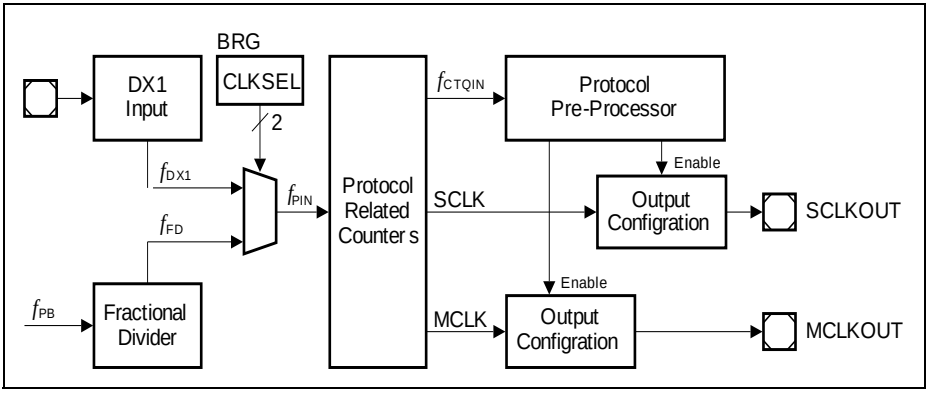


Figure 14-2 Baud Rate Generator

14.2.1.5 Channel Events and Interrupts

The notification of the user about events occurring during data traffic and data handling is based on:

- Data transfer events related to the transmission or reception of a data word, independent of the selected protocol.
- Protocol-specific events depending on the selected protocol.
- Data buffer events related to data handling by the optional FIFO data buffers.

14.2.1.6 Data Shifting and Handling

The data handling of the USIC module is based on an independent data shift unit (DSU) and a buffer structure that is similar for the supported protocols. The data shift and buffer registers are 16-bit wide (maximum data word length), but several data words can be concatenated to achieve longer data frames. The DSU inputs are the shift data (handled by input stage DX0, DX3, DX4 and DX5), the shift clock (handled by the input stage DX1), and the shift control (handled by the input stage DX2). The signal DOUT[3:0] represents the shift data outputs.

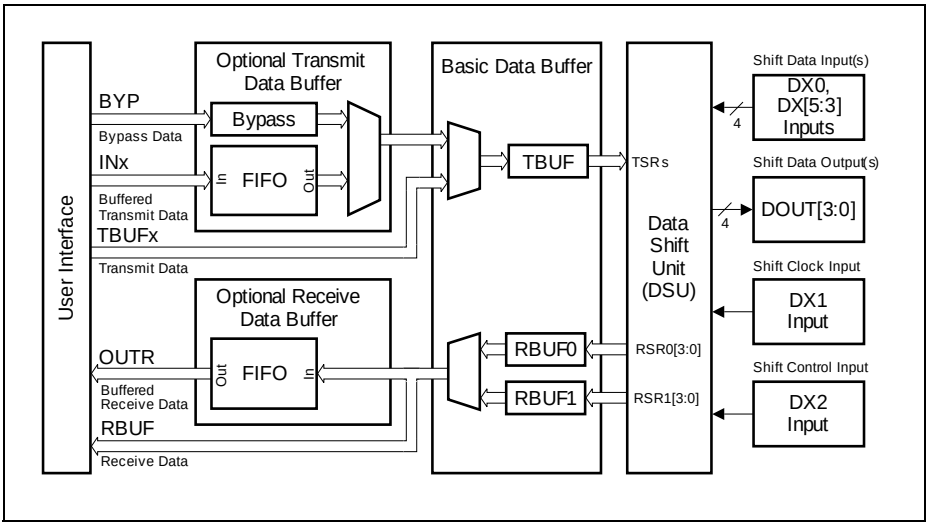


Figure 14-3 Principle of Data Buffering

The principle of data handling comprises:

- A transmitter with transmit shift registers (TSR and TSR[3:0]) in the DSU and a transmit data buffer (TBUF). A data validation scheme allows triggering and gating of data transfers by external events under certain conditions.
- A receiver with two alternating sets of receive shift registers (RSR0[3:0] and RSR1[3:0]) in the DSU and a double receive buffer structure (RBUF0, RBUF1). The alternating receive shift registers support the reception of data streams and data frames longer than one data word.
- A user interface to handle data, interrupts, and status and control information.

Basic Data Buffer Structure

The read access to received data and the write access of data to be transmitted can be handled by a basic data buffer structure.

The received data stored in the receiver buffers RBUF0/RBUF1 can be read directly from these registers. In this case, the user has to take care about the reception sequence to read these registers in the correct order. To simplify the use of the receive buffer structure, register RBUF has been introduced. A read action from this register delivers the data word received first (oldest data) to respect the reception sequence. With a read access from at least the low byte of RBUF, the data is automatically declared to be no longer new and the next received data word becomes visible in RBUF and can be read out next.

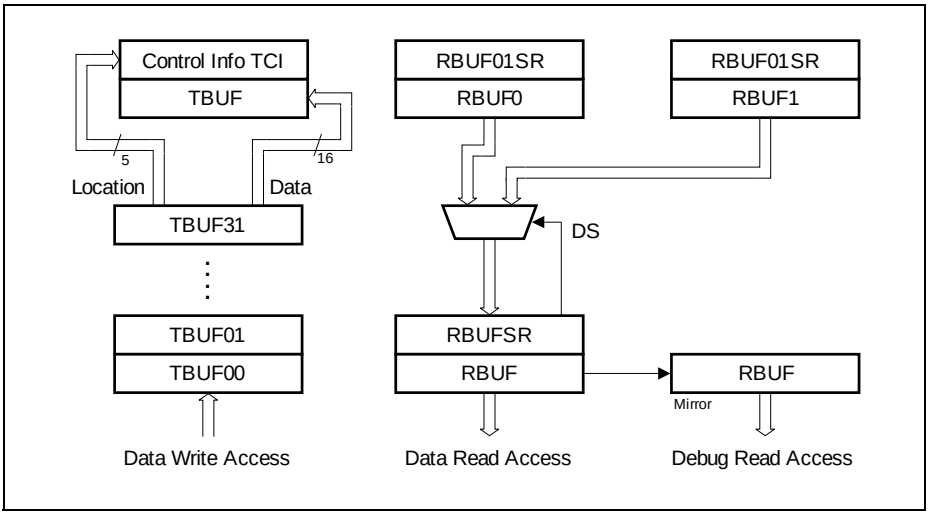


Figure 14-4 Data Access Structure without additional Data Buffer

It is recommended to read the received data words by accesses to RBUF and to avoid handling of RBUF0 and RBUF1. The USIC module also supports the use of debug accesses to receive data words. Debugger read accesses should not disturb the receive data sequence and, as a consequence, should not target RBUF. Therefore, register RBUFD has been introduced. It contains the same value as RBUF, but a read access from RBUFD does not change the status of the data (same data can be read several times). In addition to the received data, some additional status information about each received data word is available in the receiver buffer status register RBUF01SR (related to data in RBUF0 and RBUF1) and RBUFSR (related to data in RBUF).

Transmit data can be loaded to TBUF by software by writing to the transmit buffer input locations TBUFx (x = 00-31), consisting of 32 consecutive addresses. The data written to one of these input locations is stored in the transmit buffer TBUF. Additionally, the address of the written location is evaluated and can be used for additional control purposes. This 5-bit wide information (named **T**ransmit **C**ontrol **I**nformation TCI) can be used for different purposes in different protocols.

FIFO Buffer Structure

To allow easier data setup and handling, an additional data buffering mechanism can be optionally supported. The data buffer is based on the first-in-first-out principle (FIFO) that ensures that the sequence of transferred data words is respected.

If a FIFO buffer structure is used, the data handling scheme (data with associated control information) is similar to the one without FIFO. The additional FIFO buffer can be

independently enabled/disabled for transmission and reception (e.g. if data FIFO buffers are available for a specific USIC channel, it is possible to configure the transmit data path without and the receive data path with FIFO buffering).

The transmit FIFO buffer is addressed by using 32 consecutive address locations for INx instead of TBUFx (x=00-31) regardless of the FIFO depth. The 32 addresses are used to store the 5-bit TCI (together with the written data) associated with each FIFO entry.

The receive FIFO can be read out at two independent addresses, OUTR and OUTDR instead of RBUF and RBUFD. A read from the OUTR location triggers the next data packet to be available for the next read (general FIFO mechanism). In order to allow non-intrusive debugging (without risk of data loss), a second address location (OUTDR) has been introduced. A read at this location delivers the same value as OUTR, but without modifying the FIFO contents.

The transmit FIFO also has the capability to bypass the data stream and to load bypass data to TBUF. This can be used to generate high-priority messages or to send an emergency message if the transmit FIFO runs empty. The transmission control of the FIFO buffer can also use the transfer trigger and transfer gating scheme of the transmission logic for data validation (e.g. to trigger data transfers by events).

Note: The available size of a FIFO data buffer for a USIC channel depends on the specific device. Please refer to the implementation chapter for details about available FIFO buffer capability.

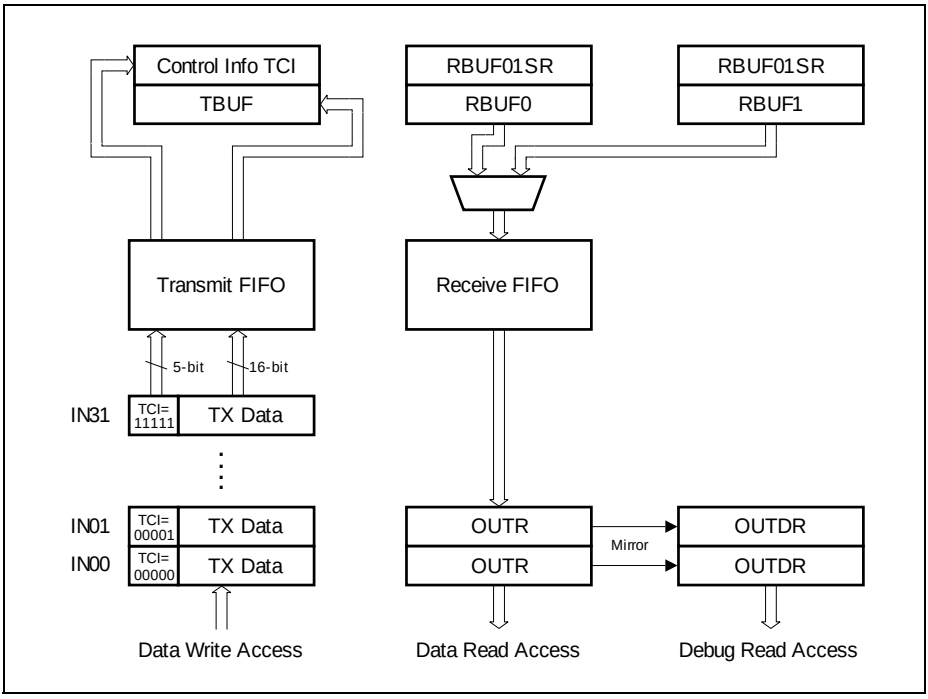


Figure 14-5 Data Access Structure with FIFO

14.2.2 Operating the USIC Communication Channel

This section describes how to operate a USIC communication channel, including protocol control and status, mode control and interrupt handling. The following aspects have to be taken into account:

- Enable the USIC module for operation and configure the behavior for the different device operation modes (see **Page 14-15**).
- Configure the pinning (refer to description in the corresponding protocol section).
- Configure the data structure (shift direction, word length, frame length, polarity, etc.).
- Configure the data buffer structure of the optional FIFO buffer area. A FIFO buffer can only be enabled if the related bit in register CCFG is set.
- Select a protocol by CCR.MODE. A protocol can only be selected if the related bit in register CCFG is set.

14.2.2.1 Protocol Control and Status

The protocol-related control and status information are located in the protocol control register PCR and in the protocol status register PSR. These registers are shared between the available protocols. As a consequence, the meaning of the bit positions in these registers is different within the protocols.

Use of PCR Bits

The signification of the bits in register PCR is indicated by the protocol-related alias names for the different protocols.

- PCR for the ASC protocol (see **Page 14-65**)
- PCR for the SSC protocol (see **Page 14-97**)
- PCR for the IIC protocol (see **Page 14-128**)
- PCR for the IIS protocol (see **Page 14-147**)

Use of PSR Flags

The signification of the flags in register PSR is indicated by the protocol-related alias names for the different protocols.

- PSR flags for the ASC protocol (see **Page 14-68**)
- PSR flags for the SSC protocol (see **Page 14-101**)
- PSR flags for the IIC protocol (see **Page 14-131**)
- PSR flags for the IIS protocol (see **Page 14-149**)

14.2.2.2 Mode Control

The mode control concept for system control tasks, such as suspend request for debugging, allows to program the module behavior under different device operating conditions. The behavior of a communication channel can be programmed for each of the device operating modes (normal operation, suspend mode). Therefore, each communication channel has an associated kernel state configuration register KSCFG defining its behavior in the following operating modes:

- **Normal operation**:
  This operating mode is the default operating mode when no suspend request is pending. The module clock is not switched off and the USIC registers can be read or written. The channel behavior is defined by KSCFG.NOMCFG.
- **Suspend mode**:
  This operating mode is requested when a suspend request is pending in the device. The module clock is not switched off and the USIC registers can be read or written. The channel behavior is defined by KSCFG.SUMCFG.

The four kernel modes defined by the register KSCFG are shown in **Table 14-3**.

Table 14-3 USIC Communication Channel Behavior

Kernel Mode	Channel Behavior	KSCFG. NOMCFG
Run mode 0	Channel operation as specified, no impact on data transfer	00_B
Run mode 1		01_B
Stop mode 0	Explicit stop condition as described in the protocol chapters	10_B
Stop mode 1		11_B

Generally, bit field KSCFG.NOMCFG should be configured for run mode 0 as default setting for standard operation. If a communication channel should not react to a suspend request (and to continue its operation as in normal mode), bit field KSCFG.SUMCFG has to be configured with the same value as KSCFG.NOMCFG. If the communication channel should show a different behavior and stop operation when a specific stop condition is reached, the code for stop mode 0 or stop mode 1 have to be written to KSCFG.SUMCFG.

The stop conditions are defined for the selected protocol (see mode control description in the protocol section).

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

from stop mode 0 to stop mode 1 (or vice versa) for the same communication channel.

Note: In XMC1100, bit field KSCFG.SUMCFG is reset to its default value by any reset. If the suspend function is required during debugging, it is recommended that it is enabled in the user initialization code. Before programming the bit field, the module clock has to be enabled and special care needs to be taken while enabling the module clock as described in the CCU (Clock Gating Control) section of the SCU chapter.

14.2.2.3 General Channel Events and Interrupts

The general event and interrupt structure is shown in **Figure 14-6**. If a defined condition is met, an event is detected and an event indication flag becomes automatically set. The flag stays set until it is cleared by software. If enabled, an interrupt can be generated if an event is detected. The actual status of the event indication flag has no influence on the interrupt generation. As a consequence, the event indication flag does not need to be cleared to generate further interrupts.

Additionally, the service request output SRx of the USIC channel that becomes activated in case of an event condition can be selected by an interrupt node pointer. This structure allows to assign events to interrupts, e.g. depending on the application, several events can share the same interrupt routine (several events activate the same SRx output) or can be handled individually (only one event activates one SRx output).

The SRx outputs are connected to interrupt control registers to handle the CPU reaction to the service requests. This assignment is described in the implementation section on **Page 14-153**.

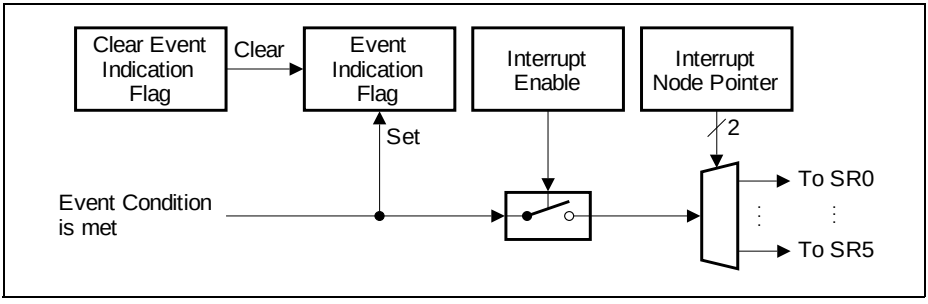


Figure 14-6 General Event and Interrupt Structure

14.2.2.4 Data Transfer Events and Interrupts

The data transfer events are based on the transmission or reception of a data word. The related indication flags are located in register PSR. All events can be individually enabled for interrupt generation.

- Receive event to indicate that a data word has been received:
  If a new received word becomes available in the receive buffer RBUF0 or RBUF1, either a receive event or an alternative receive event occurs.
  The receive event occurs if bit RBUFSR.PERR = 0. It is indicated by flag PSR.RIF and, if enabled, leads to receive interrupt.
- Receiver start event to indicate that a data word reception has started:
  When the receive clock edge that shifts in the first bit of a new data word is detected and reception is enabled, a receiver start event occurs. It is indicated by flag PSR.RSIF and, if enabled, leads to transmit buffer interrupt.
  In full duplex mode, this event follows half a shift clock cycle after the transmit buffer event and indicates when the shift control settings are internally “frozen” for the current data word reception and a new setting can be programmed.
  In SSC and IIS mode, the transmit data valid flag TCSR.TDV is cleared in single shot mode with the receiver start event.
- Alternative receive event to indicate that a specific data word has been received:
  If a new received word becomes available in the receive buffer RBUF0 or RBUF1, either a receive event or an alternative receive event occurs.
  The alternative receive event occurs if bit RBUFSR.PERR = 1. It is indicated by flag PSR.AIF and, if enabled, leads to alternative receive interrupt.
  Depending on the selected protocol, bit RBUFSR.PERR is set to indicate a parity error in ASC mode, the reception of the first byte of a new frame in IIC mode, and the WA information about right/left channel in IIS mode. In SSC mode, it is used as indication if the received word is the first data word, and is set if first and reset if not.
- Transmit shift event to indicate that a data word has been transmitted:
  A transmit shift event occurs with the last shift clock edge of a data word. It is indicated by flag PSR.TSIF and, if enabled, leads to transmit shift interrupt.
- Transmit buffer event to indicate that a data word transmission has been started:
  When a data word from the transmit buffer TBUF has been loaded to the shift register and a new data word can be written to TBUF, a transmit buffer event occurs. This happens with the transmit clock edge that shifts out the first bit of a new data word and transmission is enabled. It is indicated by flag PSR.TBIF and, if enabled, leads to transmit buffer interrupt.
  This event also indicates when the shift control settings (word length, shift direction, etc.) are internally “frozen” for the current data word transmission.
  In ASC and IIC mode, the transmit data valid flag TCSR.TDV is cleared in single shot mode with the transmit buffer event.
- Data lost event to indicate a loss of the oldest received data word:
  If the data word available in register RBUF (oldest data word from RBUF0 or RBUF1)

has not been read out before it becomes overwritten with new incoming data, this event occurs. It is indicated by flag PSR.DLIF and, if enabled, leads to a protocol interrupt.

Table 14-4 shows the registers, bits and bit fields indicating the data transfer events and controlling the interrupts of a USIC channel.

Table 14-4 Data Transfer Events and Interrupt Handling

Event	Indication Flag	Indication cleared by	Interrupt enabled by	SRx Output selected by
Standard receive event	PSR.RIF	PSCR.CRIF	CCR.RIEN	INPR.RINP
Receive start event	PSR.RSIF	PSCR.CRSIF	CCR.RSIEN	INPR.TBINP
Alternative receive event	PSR.AIF	PSCR.CAIF	CCR.AIEN	INPR.AINP
Transmit shift event	PSR.TSIF	PSCR.CTSIF	CCR.TSIEN	INPR.TSINP
Transmit buffer event	PSR.TBIF	PSCR.CTBIF	CCR.TBIEN	INPR.TBINP
Data lost event	PSR.DLIF	PSCR.CDLIF	CCR.DLIEN	INPR.PINP

Figure 14-7 shows the two transmit events and interrupts.

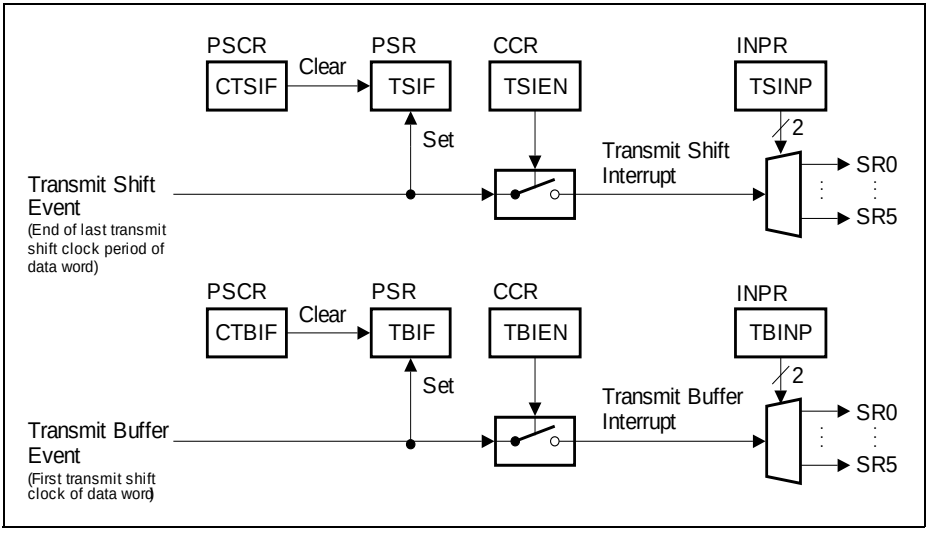


Figure 14-7 Transmit Events and Interrupts

Figure 14-8 shows the receive events and interrupts.

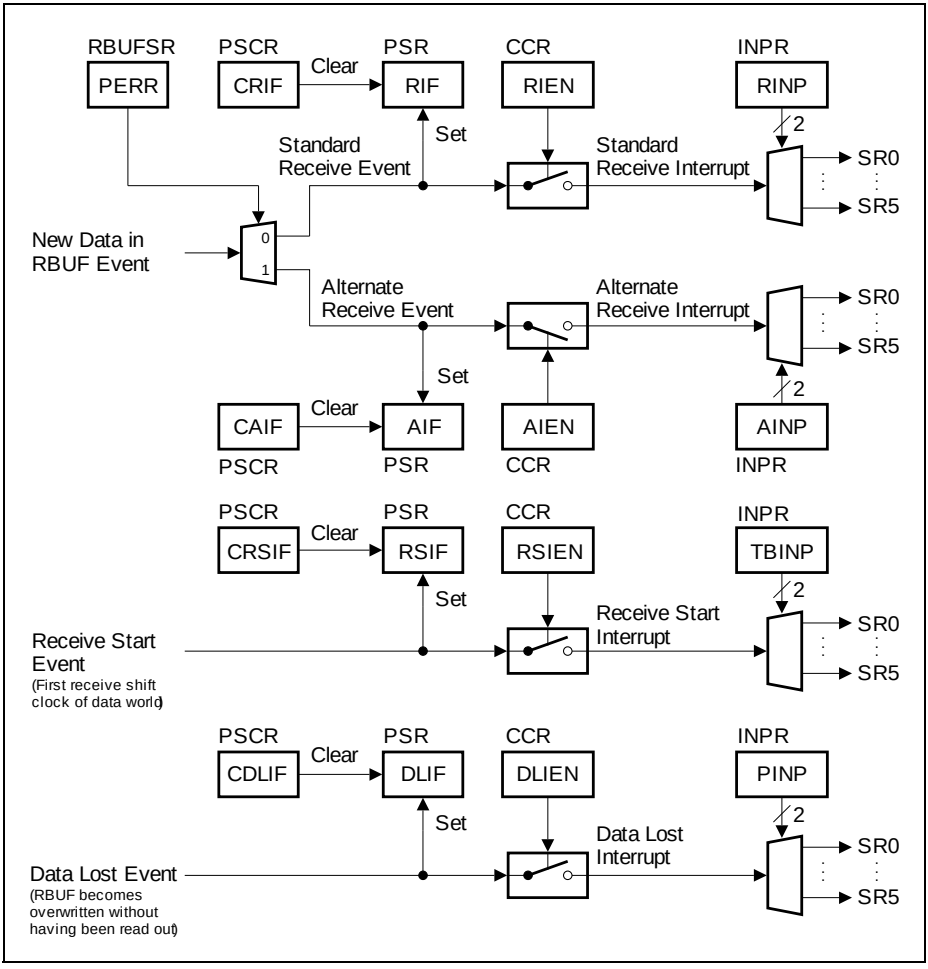


Figure 14-8 Receive Events and Interrupts

14.2.2.5 Baud Rate Generator Event and Interrupt

The baud rate generator event is based on the capture mode timer reaching its maximum value. It is indicated by flag PSR.BRGIF and, if enabled, leads to a protocol interrupt.

Table 14-5 shows the registers, bits and bit fields indicating the baud rate generator event and controlling the interrupt of a USIC channel.

Table 14-5 Baud Rate Generator Event and Interrupt Handling

Event	Indication Flag	Indication cleared by	Interrupt enabled by	SRx Output selected by
Baud rate generator event	PSR. BRGIF	PSCR. CBRGIF	CCR. BRGIEN	INPR.PINP

Figure 14-9 shows the baud rate generator event and interrupt.

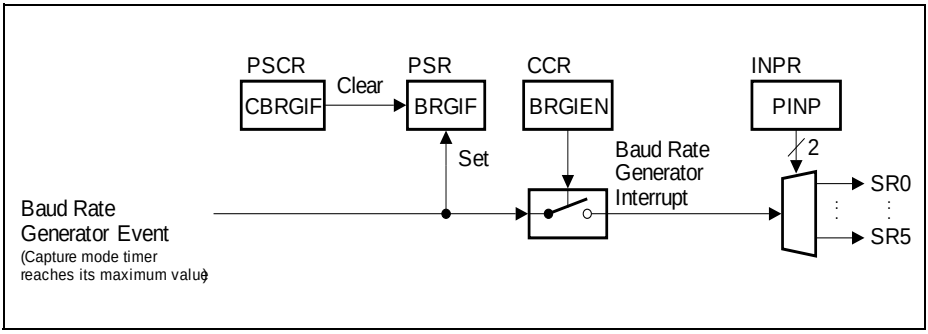


Figure 14-9 Baud Rate Generator Event and Interrupt

14.2.2.6 Protocol-specific Events and Interrupts

These events are related to protocol-specific actions that are described in the corresponding protocol chapters. The related indication flags are located in register PSR. All events can be individually enabled for the generation of the common protocol interrupt.

- Protocol-specific events in ASC mode:
  Synchronization break, data collision on the transmit line, receiver noise, format error in stop bits, receiver frame finished, transmitter frame finished
- Protocol-specific events in SSC mode:
  MSLS event (start-end of frame in master mode), DX2T event (start/end of frame in slave mode), both based on slave select signals, parity error
- Protocol-specific events in IIC mode:
  Wrong transmit code (error in frame sequence), start condition received, repeated start condition received, stop condition received, non-acknowledge received, arbitration lost, slave read request, other general errors
- Protocol-specific events in IIS mode:
  DX2T event (change on WA line), WA falling edge or rising edge detected, WA generation finished

Table 14-6 Protocol-specific Events and Interrupt Handling

Event	Indication Flag	Indication cleared by	Interrupt enabled by	SRx Output selected by
Protocol-specific events in ASC mode	PSR.ST[8:2]	PSCR.CST[8:2]	PCR.CTR[7:3]]	INPR.PINP
Protocol-specific events in SSC mode	PSR.ST[3:2]	PSCR.CST[3:2]	PCR.CTR[15:14]	INPR.PINP
Protocol-specific events in IIC mode	PSR.ST[8:1]	PSCR.CST[8:1]	PCR.CTR[24:18]	INPR.PINP
Protocol-specific events in IIS mode	PSR.ST[6:3]	PSCR.CST[6:3]	PCR.CTR[6:4], PCR.CTR[15]	INPR.PINP

14.2.3 Operating the Input Stages

All input stages offer the same feature set. They are used for all protocols, because the signal conditioning can be adapted in a very flexible way and the digital filters can be switched on and off separately.

14.2.3.1 General Input Structure

There are generally two types of input stages, one for the data input stages DX0, DX[5:3] and the other for non-data input stages DX[2:1], as shown in **Figure 14-10** and **Figure 14-11**. The difference is that for the data input stages, the input signal can be additionally selected from the port signal HWINn if hardware port control is enabled through CCR.HPCEN bit. All other enable/disable functions and selections are controlled independently for each input stage by bits in the registers DXnCR.

The desired input signal can be selected among the input lines DXnA to DXnG and a permanent 1-level by programming bit field DSEL (for the data input stages, hardware port control must be disabled for DSEL to take effect). Please refer to the interconnects section (**Section 14.12**) for the device-specific input signal assignment. Bit DPOL allows a polarity inversion of the selected input signal to adapt the input signal polarity to the internal polarity of the data shift unit and the protocol state machine. For some protocols, the input signals can be directly forwarded to the data shift unit for the data transfers (DSEN = 0, INSW = 1) without any further signal conditioning. In this case, the data path does not contain any delay due to synchronization or filtering.

In the case of noise on the input signals, there is the possibility to synchronize the input signal (signal DXnS is synchronized to f_{PB}) and additionally to enable a digital noise filter in the signal path. The synchronized input signal (and optionally filtered if DFEN = 1) is taken into account by DSEN = 1. Please note that the synchronization leads to a delay in the signal path of 2-3 times the period of f_{PB}.

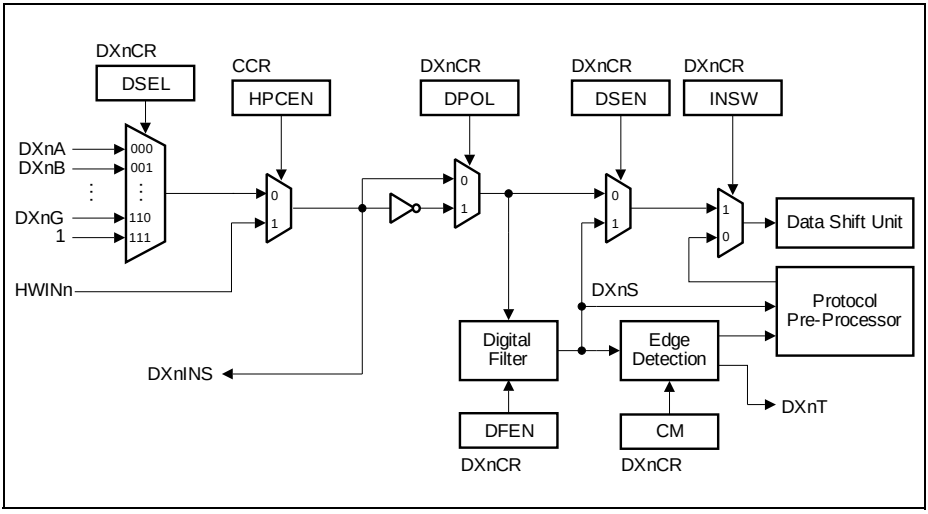


Figure 14-10 Input Conditioning for DX0 and DX[5:3]

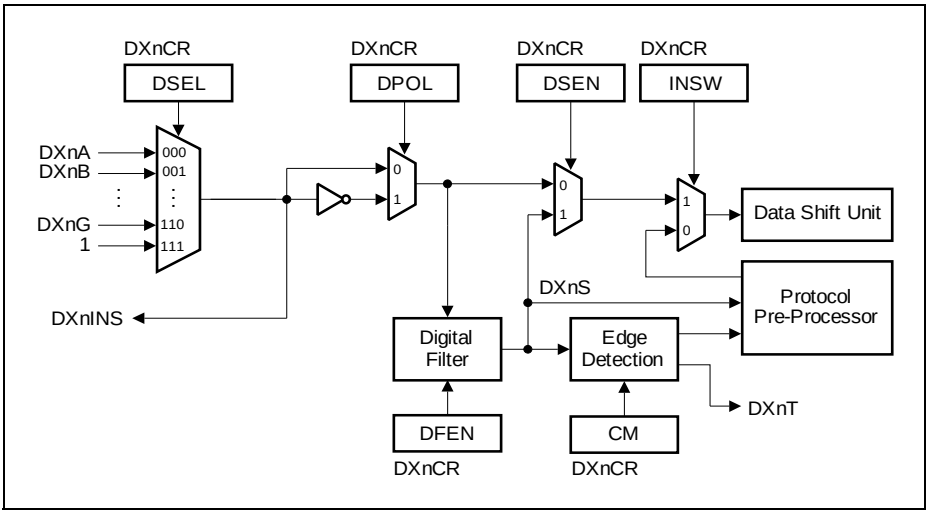


Figure 14-11 Input Conditioning for DX[2:1]

If the input signals are handled by a protocol pre-processor, the data shift unit is directly connected to the protocol pre-processor by INSW = 0. The protocol pre-processor is connected to the synchronized input signal DXnS and, depending on the selected protocol, also evaluates the edges.

To support delay compensation in SSC and IIS protocols, the DX1 input stage additionally allows the receive shift clock to be controlled independently from the transmit shift clock through the bit DCEN. When DCEN = 0, the shift clock source is selected by INSW and is the same for both receive and transmit. When DCEN = 1, the receive shift clock is derived from the selected input line as shown in **Figure 14-12**.

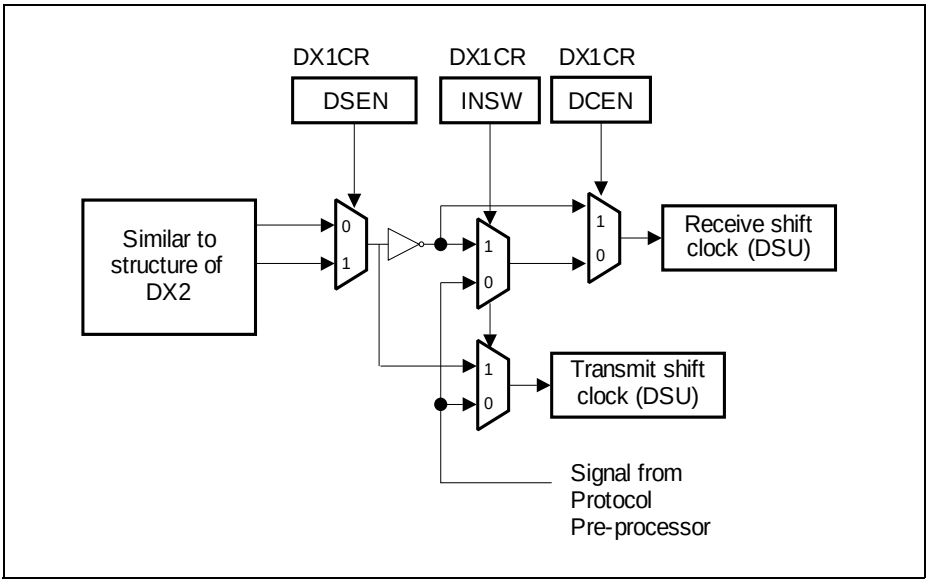


Figure 14-12 Delay Compensation Enable in DX1

14.2.3.2 Digital Filter

The digital filter can be enabled to reduce noise on the input signals. Before being filtered, the input signal becomes synchronized to f_{PB}. If the filter is disabled, signal DXnS corresponds to the synchronized input signal. If the filter is enabled, pulses shorter than one filter sampling period are suppressed in signal DXnS. After an edge of the synchronized input signal, signal DXnS changes to the new value if two consecutive samples of the new value have been detected.

In order to adapt the filter sampling period to different applications, it can be programmed. The first possibility is the system frequency f_{PB}. Longer pulses can be suppressed if the fractional divider output frequency f_{FD} is selected. This frequency is programmable in a wide range and can also be used to determine the baud rate of the data transfers.

In addition to the synchronization delay of 2-3 periods of f_{PB}, an enabled filter adds a delay of up to two filter sampling periods between the selected input and signal DXnS.

14.2.3.3 Edge Detection

The synchronized (and optionally filtered) signal DXnS can be used as input to the data shift unit and is also an input to the selected protocol pre-processor. If the protocol pre-processor does not use the DXnS signal for protocol-specific handling, DXnS can be

used for other tasks, e.g. to control data transmissions in master mode (a data word can be tagged valid for transmission, see chapter about data buffering).

A programmable edge detection indicates that the desired event has occurred by activating the trigger signal DXnT (introducing a delay of one period of f_{PB} before a reaction to this event can take place).

14.2.3.4 Selected Input Monitoring

The selected input signal of each input stage has been made available with the signals DXnINS. These signals can be used in the system to trigger other actions, e.g. to generate interrupts.

14.2.3.5 Loop Back Mode

The USIC transmitter output signals can be connected to the corresponding receiver inputs of the same communication channel in loop back mode. Therefore, the input “G” of the input stages that are needed for the selected protocol have to be selected. In this case, drivers for ASC, SSC, and IIS can be evaluated on-chip without the connections to port pins. Data transferred by the transmitter can be received by the receiver as if it would have been sent by another communication partner.

14.2.4 Operating the Baud Rate Generator

The following blocks can be configured to operate the baud rate generator, see also **Figure 14-2**.

14.2.4.1 Fractional Divider

The fractional divider generates its output frequency f_{FD} by either dividing the input frequency f_{PB} by an integer factor n or by multiplication of n/1024. It has two operating modes:

- Normal divider mode (FDR.DM = 01_B):
  In this mode, the output frequency f_{FD} is derived from the input clock f_{PB} by an integer division by a value between 1 and 1024. The division is based on a counter FDR.RESULT that is incremented by 1 with f_{PB}. After reaching the value $3FF_H$, the counter is loaded with FDR.STEP and then continues counting. In order to achieve $f_{FD} = f_{PB}$, the value of STEP has to be programmed with $3FF_H$.
  The output frequency in normal divider mode is defined by the equation:

$$f_{FD} = f_{PB} \times \frac{1}{n} \qquad \text{with } n = 1024 - \text{STEP} \tag{14.1}$$

- Fractional divider mode (FDR.DM = 10_B):
  In this mode, the output frequency f_{FD} is derived from the input clock f_{PB} by a

fractional multiplication of n/1024 for a value of n between 0 and 1023. In general, the fractional divider mode allows to program the average output clock frequency with a finer granularity than in normal divider mode. Please note that in fractional divider mode f_{FD} can have a maximum period jitter of one f_{PB} period. This jitter is not accumulated over several cycles.
The frequency f_{FD} is generated by an addition of FDR.STEP to FDR.RESULT with f_{PB}. The frequency f_{FD} is based on the overflow of the addition result over $3FF_H$.
The output frequency in fractional divider mode is defined by the equation:

$$f_{FD} = f_{PB} \times \frac{n}{1024} \qquad \text{with n = STEP} \tag{14.2}$$

The output frequency f_{FD} of the fractional divider is selected for baud rate generation by BRG.CLKSEL = 00_B.

14.2.4.2 External Frequency Input

The baud rate can be generated referring to an external frequency input (instead of to f_{PB}) if in the selected protocol the input stage DX1 is not needed (DX1CTR.INSW = 0). In this case, an external frequency input signal at the DX1 input stage can be synchronized and sampled with the system frequency f_{PB}. It can be optionally filtered by the digital filter in the input stage. This feature allows data transfers with frequencies that can not be generated by the device itself, e.g. for specific audio frequencies.

If BRG.CLKSEL = 10_B, the trigger signal DX1T determines f_{DX1}. In this mode, either the rising edge, the falling edge, or both edges of the input signal can be used for baud rate generation, depending on the configuration of the DX1T trigger event by bit field DX1CTR.CM. The signal MCLK toggles with each trigger event of DX1T.

If BRG.CLKSEL = 11_B, the rising edges of the input signal can be used for baud rate generation. The signal MCLK represents the synchronized input signal DX1S.

Both, the high time and the low time of external input signal must each have a length of minimum 2 periods of f_{PB} to be used for baud rate generation.

14.2.4.3 Divider Mode Counter

The divider mode counter is used for an integer division delivering the output frequency f_{PDIV}. Additionally, two divider stages with a fixed division by 2 provide the output signals MCLK and SCLK with 50% duty cycle. If the fractional divider mode is used, the maximum fractional jitter of 1 period of f_{PB} can also appear in these signals. The output frequencies of this divider is controlled by register BRG.

In order to define a frequency ratio between the master clock MCLK and the shift clock SCLK, the divider stage for MCLK is located in front of the divider by PDIV+1, whereas the divider stage for SCLK is located at the output of this divider.

$$f_{MCLK} = \frac{f_{PIN}}{2} \tag{14.3}$$

$$f_{SCLK} = \frac{f_{PDIV}}{2} \tag{14.4}$$

In the case that the master clock is used as reference for external devices (e.g. for IIS components) and a fixed phase relation to SCLK and other timing signals is required, it is recommended to use the MCLK signal as input for the PDIV divider. If the MCLK signal is not used or a fixed phase relation is not necessary, the faster frequency f_{PIN} can be selected as input frequency.

$$\begin{aligned} f_{PDIV} &= f_{PIN} \times \frac{1}{PDIV + 1} && \text{if PPPEN = 0} \\ f_{PDIV} &= f_{MCLK} \times \frac{1}{PDIV + 1} && \text{if PPPEN = 1} \end{aligned} \tag{14.5}$$

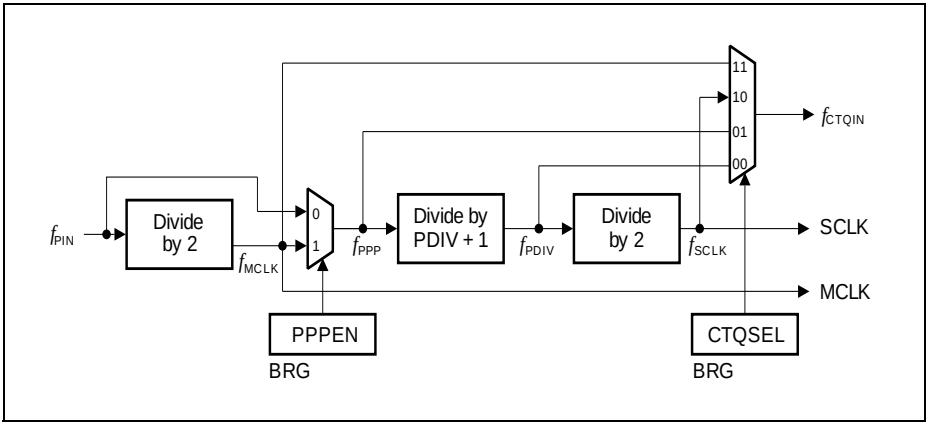


Figure 14-13 Divider Mode Counter

14.2.4.4 Capture Mode Timer

The capture mode timer is used for time interval measurement and is enabled by BRG.TMEN = 1. The timer works independently from the divider mode counter. Therefore, any serial data reception or transmission can continue while the timer is performing timing measurements. The timer counts f_{PPP} periods and stops counting

when it reaches its maximum value. Additionally, a baud rate generator interrupt event is generated (bit PSR.BRGIF becomes set).

If an event is indicated by DX0T or DX1T, the actual timer value is captured into bit field CMTR.CTV and the timer restarts from 0. Additionally, a transmit shift interrupt event is generated (bit PSR.TSIF becomes set).

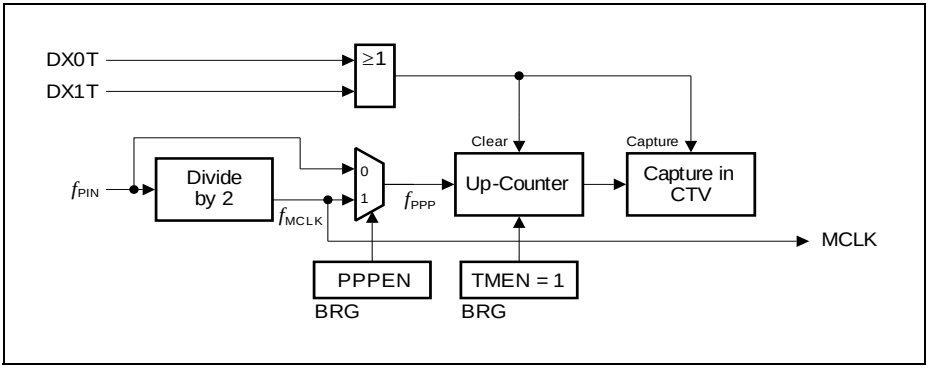


Figure 14-14 Protocol-Related Counter (Capture Mode)

The capture mode timer can be used to measure the baud rate in slave mode before starting or during data transfers, e.g. to measure the time between two edges of a data signal (by DX0T) or of a shift clock signal (by DX1T). The conditions to activate the DXnT trigger signals can be configured in each input stage.

14.2.4.5 Time Quanta Counter

The time quanta counter CTQ associated to the protocol pre-processor allows to generate time intervals for protocol-specific purposes. The length of a time quantum t_q is given by the selected input frequency f_{CTQIN} and the programmed pre-divider value.
The meaning of the time quanta depend on the selected protocol, please refer to the corresponding chapters for more protocol-specific information.

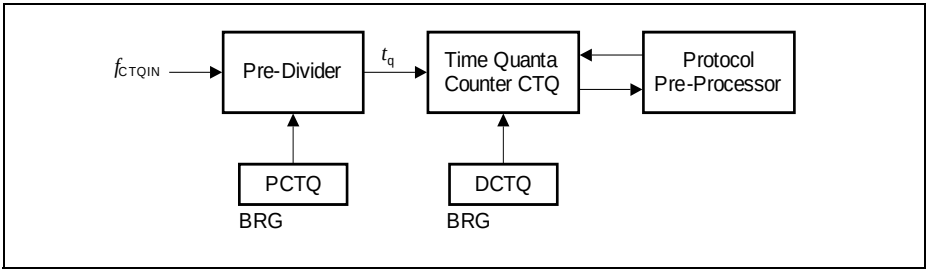


Figure 14-15 Time Quanta Counter

14.2.4.6 Master and Shift Clock Output Configuration

The master clock output signal MCLKOUT available at the corresponding output pin can be configured in polarity. The MCLK signal can be generated for each protocol in order to provide a kind of higher frequency time base compared to the shift clock.

The configuration mechanism of the master clock output signal MCLKOUT ensures that no shortened pulses can occur. Each MCLK period consists of two phases, an active phase, followed by a passive phase. The polarity of the MCLKOUT signal during the active phase is defined by the inverted level of bit BRG.MCLKCFG, evaluated at the start of the active phase. The polarity of the MCLKOUT signal during the passive phase is defined by bit BRG.MCLKCFG, evaluated at the start of the passive phase. If bit BRG.MCLKOUT is programmed with another value, the change is taken into account with the next change between the phases. This mechanism ensures that no shorter pulses than the length of a phase occur at the MCLKOUT output. In the example shown in **Figure 14-16**, the value of BRG.MCLKCFG is changed from 0 to 1 during the passive phase of MCLK period 2.

The generation of the MCLKOUT signal is enabled/disabled by the protocol pre-processor, based on bit PCR.MCLK. After this bit has become set, signal MCLKOUT is generated with the next active phase of the MCLK period. If PCR.MCLK = 0 (MCLKOUT generation disabled), the level for the passive phase is also applied for active phase.

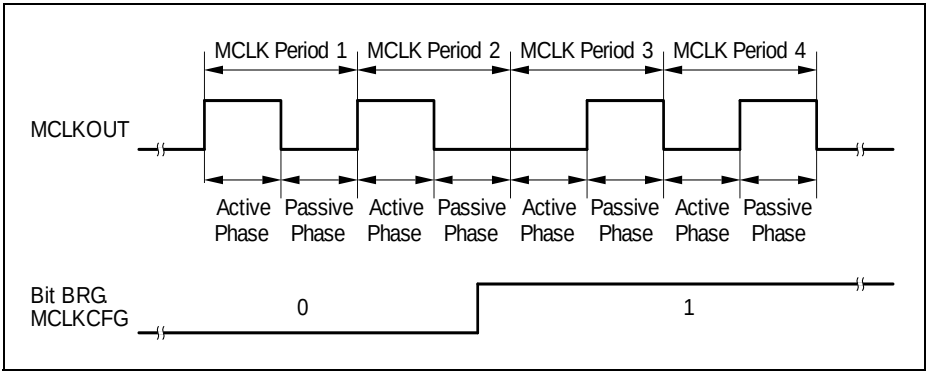


Figure 14-16 Master Clock Output Configuration

The shift clock output signal SCLKOUT available at the corresponding output pin can be configured in polarity and additionally, a delay of one period of f_{PDIV} (= half SCLK period) can be introduced. The delay allows to adapt the order of the shift clock edges to the application requirements. If the delay is used, it has to be taken into account for the calculation of the signal propagation times and loop delays.

The mechanism for the polarity control of the SCLKOUT signal is similar to the one for MCLKOUT, but based on bit field BRG.SCLKCFG. The generation of the SCLKOUT

signal is enabled/disabled by the protocol pre-processor. Depending on the selected protocol, the protocol pre-processor can control the generation of the SCLKOUT signal independently of the divider chain, e.g. for protocols without the need of a shift clock available at a pin, the SCLKOUT generation is disabled.

14.2.5 Operating the Transmit Data Path

The transmit data path is based on 16-bit wide transmit shift registers (TSR and TSR[3:0]) and a transmit buffer TBUF. The data transfer parameters like data word length, data frame length, or the shift direction are controlled commonly for transmission and reception by the shift control register SCTR. The transmit control and status register TCSR controls the transmit data handling and monitors the transmit status.

A change of the value of the data shift output signal DOUTx only happens at the corresponding edge of the shift clock input signal. The level of the last data bit of a data word/frame is held constant at DOUTx until the next data word begins with the next corresponding edge of the shift clock.

14.2.5.1 Transmit Buffering

The transmit shift registers can not be directly accessed by software, because they are automatically updated with the value stored in the transmit buffer TBUF if a currently transmitted data word is finished and new data is valid for transmission. Data words can be loaded directly into TBUF by writing to one of the transmit buffer input locations TBUFx (see **Page 14-32**) or, optionally, by a FIFO buffer stage (see **Page 14-38**).

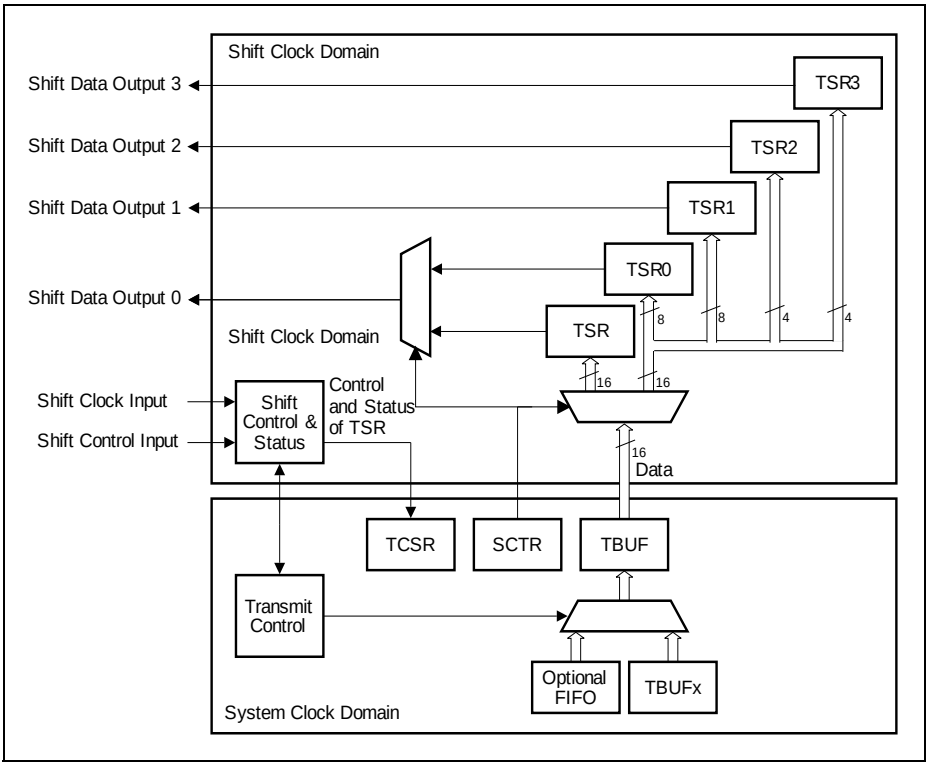


Figure 14-17 Transmit Data Path

14.2.5.2 Transmit Data Shift Mode

The transmit shift data can be selected to be shifted out one, two or four bits at a time through the corresponding number of output lines. This option allows the USIC to support protocols such as the Dual- and Quad-SSC. The selection is done through the bit field DSM in the shift control register SCTR.

Note: The bit field SCTR.DSM controls the data shift mode for both the transmit and receive paths to allow the transmission and reception of data through one to four data lines.

For the shift mode with two or four parallel data outputs, the data word and frame length must be in multiples of two or four respectively. The number of data shifts required to output a specific data word or data frame length is thus reduced by the factor of the number of parallel data output lines. For example, to transmit a 16-bit data word through four output lines, only four shifts are required.

Depending on the shift mode, different transmit shift registers with different bit composition are used as shown in **Table 14-7**. Note that the 'n' in the table denotes the shift number less one, i.e. for the first data shift n = 0, the second data shift n = 1 and continues until the total number of shifts less one is reached.

For all transmit shift registers, whether the first bit shifted out is the MSB or LSB depends on the setting of SCTR.SDIR.

Table 14-7 Transmit Shift Register Composition

Transmit Shift Registers	Single Data Output (SCTR.DSM = 00_B)	Two Data Outputs (SCTR.DSM = 10_B)	Four Data Outputs (SCTR.DSM = 11_B)
TSR	All data bits	Not used	Not used
TSR0	Not used	Bit n*2	Bit n*4
TSR1	Not used	Bit n*2 + 1	Bit n*4 + 1
TSR2	Not used	Not used	Bit n*4 + 2
TSR3	Not used	Not used	Bit n*4 + 3

14.2.5.3 Transmit Control Information

The transmit control information TCI is a 5-bit value derived from the address x of the written TBUFx or INx input location. For example, writing to TBUF31 generates a TCI of 11111_B.

The TCI can be used as an additional control parameter for data transfers to dynamically change the data word length, the data frame length, or other protocol-specific functions (for more details about this topic, please refer to the corresponding protocol chapters). The way how the TCI is used in different applications can be programmed by the bits WLEMD, FLEMD, SELMD, WAMD and HPCMD in register TCSR. Please note that not all possible settings lead to useful system behavior.

- Word length control:
  If TCSR.WLEMD = 1, bit field SCTR.WLE is updated with TCI[3:0] if a transmit buffer input location TBUFx is written. This function can be used in all protocols to dynamically change the data word length between 1 and 16 data bits per data word. Additionally, bit TCSR.EOF is updated with TCI[4]. This function can be used in SSC master mode to control the slave select generation to finish data frames. It is recommended to program TCSR.FLEMD = TCSR.SELMD = TCSR.WAMD = TCSR.HPCMD = 0.
- Frame length control:
  If TCSR.FLEMD = 1, bit field SCTR.FLE[4:0] is updated with TCI[4:0] and SCTR.FLE[5] becomes 0 if a transmit buffer input location TBUFx is written. This function can be used in all protocols to dynamically change the data frame length

between 1 and 32 data bits per data frame. It is recommended to program TCSR.SELMD = TCSR.WLEMD = TCSR.WAMD = TCSR.HPCMD = 0.

- Select output control:
  If TCSR.SELMD = 1, bit field PCR.CTR[20:16] is updated with TCI[4:0] and PCR.CTR[23:21] becomes 0 if a transmit buffer input location TBUFx is written. This function can be used in SSC master mode to define the targeted slave device(s). It is recommended to program TCSR.WLEMD = TCSR.FLEMD = TCSR.WAMD = TCSR.HPCMD = 0.
- Word address control:
  If TCSR.WAMD = 1, bit TCSR.WA is updated with TCI[4] if a transmit buffer input location TBUFx is written. This function can be used in IIS mode to define if the data word is transmitted on the right or the left channel. It is recommended to program TCSR.WLEMD = TCSR.FLEMD = TCSR.SELMD = TCSR.HPCMD = 0.
- Hardware Port control:
  If TCSR.HPCMD = 1, bit field SCTR.DSM is updated with TCI[1:0] if a transmit buffer input location TBUFx is written. This function can be used in SSC protocols to dynamically change the number of data input and output lines to set up for standard, dual and quad SSC formats.
  Additionally, bit TCSR.HPCDIR is updated with TCI[2]. This function can be used in SSC protocols to control the pin(s) direction when the hardware port control function is enabled through CCR.HPCEN = 1. It is recommended to program TCSR.FLEMD = TCSR.WLEMD = TCSR.SELMD = TCSR.WAMD = 0.

14.2.5.4 Transmit Data Validation

The data word in the transmit buffer TBUF can be tagged valid or invalid for transmission by bit TCSR.TDV (transmit data valid). A combination of data flow related and event related criteria define whether the data word is considered valid for transmission. A data validation logic checks the start conditions for each data word. Depending on the result of the check, the transmit shift register is loaded with different values, according to the following rules:

- If a USIC channel is the communication master (it defines the start of each data word transfer), a data word transfer can only be started with valid data in the transmit buffer TBUF. In this case, the transmit shift register is loaded with the content of TBUF, that is not changed due to this action.
- If a USIC channel is a communication slave (it can not define the start itself, but has to react), a data word transfer requested by the communication master has to be started independently of the status of the data word in TBUF. If a data word transfer is requested and started by the master, the transmit shift register is loaded at the first corresponding shift clock edge either with the data word in TBUF (if it is valid for transmission) or with the level defined by bit SCTR.PDL (if the content of TBUF has not been valid at the transmission start). In both cases, the content of TBUF is not changed.

The control and status bits for the data validation are located in register TCSR. The data validation is based on the logic blocks shown in **Figure 14-18**.

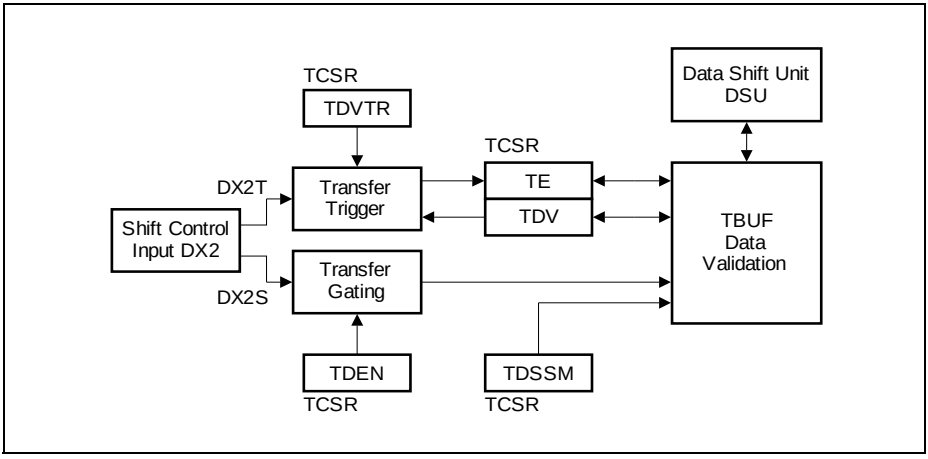


Figure 14-18 Transmit Data Validation

- A transfer gating logic enables or disables the data word transfer from TBUF under software or under hardware control. If the input stage DX2 is not needed for data shifting, signal DX2S can be used for gating purposes. The transfer gating logic is controlled by bit field TCSR.TDEN.
- A transfer trigger logic supports data word transfers related to events, e.g. timer based or related to an input pin. If the input stage DX2 is not needed for data shifting, signal DX2T can be used for trigger purposes. The transfer trigger logic is controlled by bit TCSR.TDVTR and the occurrence of a trigger event is indicated by bit TCSR.TE. For example, this can be used for triggering the data transfer upon receiving the Clear to Send (CTS) signal at DX2 in the RS-232 protocol.
- A data validation logic combining the inputs from the gating logic, the triggering logic and DSU signals. A transmission of the data word located in TBUF can only be started if the gating enables the start, bit TCSR.TDV = 1, and bit TCSR.TE = 1.
  The content of the transmit buffer TBUF should not be overwritten with new data while it is valid for transmission and a new transmission can start. If the content of TBUF has to be changed, it is recommended to clear bit TCSR.TDV by writing FMR.MTDV = 10_B before updating the data. Bit TCSR.TDV becomes automatically set when TBUF is updated with new data. Another possibility are the interrupts TBI (for ASC and IIC) or RSI (for SSC and IIS) indicating that a transmission has started. While a transmission is in progress, TBUF can be loaded with new data. In this case the user has to take care that an update of the TBUF content takes place before a new transmission starts.

With this structure, the following data transfer functionality can be achieved:

- If bit TCSR.TDSSM = 0, the content of the transmit buffer TBUF is always considered as valid for transmission. The transfer trigger mechanism can be used to start the transfer of the same data word based on the selected event (e.g. on a timer base or an edge at a pin) to realize a kind of life-sign mechanism. Furthermore, in slave mode, it is ensured that always a correct data word is transmitted instead of the passive data level.
- Bit TCSR.TDSSM = 1 has to be programmed to allow word-by-word data transmission with a kind of single-shot mechanism. After each transmission start, a new data word has to be loaded into the transmit buffer TBUF, either by software write actions to one of the transmit buffer input locations TBUFx or by an optional data buffer (e.g. FIFO buffer). To avoid that data words are sent out several times or to allow data handling with an additional data buffer (e.g. FIFO), bit TCSR.TDSSM has to be 1.
- Bit TCSR.TDV becoming automatically set when a new data word is loaded into the transmit buffer TBUF, a transmission start can be requested by a write action of the data to be transmitted to at least the low byte of one of the transmit buffer input locations TBUFx. The additional information TCI can be used to control the data word length or other parameters independently for each data word by a single write access.
- Bit field FMR.MTDV allows software driven modification (set or clear) of bit TCSR.TDV. Together with the gating control bit field TCSR.TDEN, the user can set up the transmit data word without starting the transmission. A possible program sequence could be: clear TCSR.TDEN = 00_B, write data to TBUFx, clear TCSR.TDV by writing FMR.MTDV = 10_B, re-enable the gating with TCSR.TDEN = 01_B and then set TCSR.TDV under software control by writing FMR.MTDV = 01_B.

14.2.6 Operating the Receive Data Path

The receive data path is based on two sets of 16-bit wide receive shift registers RSR0[3:0] and RSR1[3:0] and a receive buffer for each of the set (RBUF0 and RBUF1). The data transfer parameters like data word length, data frame length, or the shift direction are controlled commonly for transmission and reception by the shift control registers.

Register RBUF01SR monitors the status of RBUF0 and RBUF1.

14.2.6.1 Receive Buffering

The receive shift registers cannot be directly accessed by software, but their contents are automatically loaded into the receive buffer registers RBUF0 (or RBUF1 respectively) if a complete data word has been received or the frame is finished. The received data words in RBUF0 or RBUF1 can be read out in the correct order directly from register RBUF or, optionally, from a FIFO buffer stage (see **Page 14-38**).

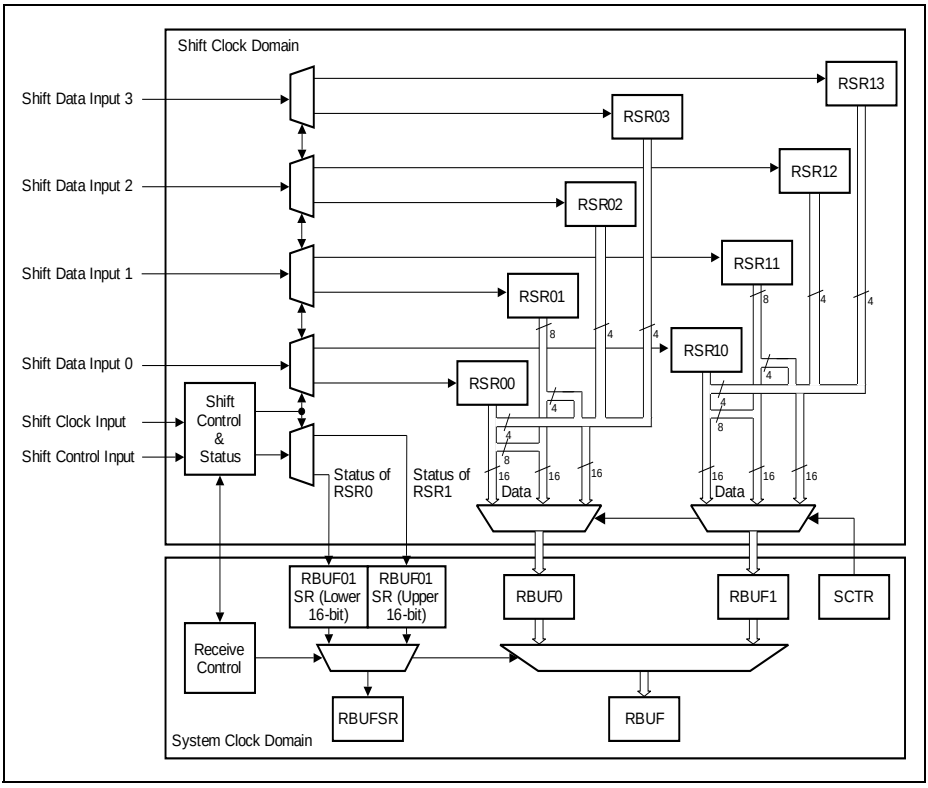


Figure 14-19 Receive Data Path

14.2.6.2 Receive Data Shift Mode

Receive data can be selected to be shifted in one, two or four bits at a time through the corresponding number of input stages and data input lines. This option allows the USIC to support protocols such as the Dual- and Quad-SSC. The selection is done through the bit field DSM in the shift control register SCTR.

Note: The bit field SCTR.DSM controls the data shift mode for both the transmit and receive paths to allow the transmission and reception of data through one to four data lines.

For the shift mode with two or four parallel data inputs, the data word and frame length must be in multiples of two or four respectively. The number of data shifts required to input a specific data word or data frame length is thus reduced by the factor of the

number of parallel data input lines. For example, to receive a 16-bit data word through four input lines, only four shifts are required.

Depending on the shift mode, different receive shift registers with different bit composition are used as shown in **Table 14-7**. Note that the 'n' in the table denotes the shift number less one, i.e. for the first data shift n = 0, the second data shift n = 1 and continues until the total number of shifts less one is reached.

For all receive shift registers, whether the first bit shifted in is the MSB or LSB depends on the setting of SCTR.SDIR.

Table 14-8 Receive Shift Register Composition

Receive Shift Registers	Input stage used	Single Data Input (SCTR.DSM = 00_B)	Two Data Inputs (SCTR.DSM = 10_B)	Four Data Inputs (SCTR.DSM = 11_B)
RSRx0	DX0	All data bits	Bit n*2	Bit n*4
RSRx1	DX3	Not used	Bit n*2 + 1	Bit n*4 + 1
RSRx2	DX4	Not used	Not used	Bit n*4 + 2
RSRx3	DX5	Not used	Not used	Bit n*4 + 3

14.2.6.3 Baud Rate Constraints

The following baud rate constraints have to be respected to ensure correct data reception and buffering. The user has to take care about these restrictions when selecting the baud rate and the data word length with respect to the module clock frequency f_{PB}.

- A received data word in a receiver shift registers RSRx[3:0] must be held constant for at least 4 periods of f_{PB} in order to ensure correct loading of the related receiver buffer register RBUFx.
- The shift control signal has to be constant inactive for at least 5 periods of f_{PB} between two consecutive frames in order to correctly detect the end of a frame.
- The shift control signal has to be constant active for at least 1 period of f_{PB} in order to correctly detect a frame (shortest frame).
- A minimum setup and hold time of the shift control signal with respect to the shift clock signal has to be ensured.

14.2.7 Hardware Port Control

Hardware port control is intended for SSC protocols with half-duplex configurations, where a single port pin is used for both input and output data functions, to control the pin direction through a dedicated hardware interface. All settings in Pn_IOCRy.PCx, except for the input pull device selection and output driver type (open drain or push-pull), are overruled by the hardware port control.

Input pull device selection is done through the Pn_IOCRy.PCx as before, while the output driver is fixed to push-pull-only in this mode.

One, two or four port pins can be selected with the hardware port control to support SSC protocols with multiple bi-directional data lines, such as dual- and quad-SSC. This selection and the enable/disable of the hardware port control is done through CCR.HPCEN. The direction of all selected pins is controlled through a single bit SCTR.HPCDIR.

SCTR.HPCDIR is automatically shadowed with the start of each data word to prevent changing of the pin direction in the middle of a data word transfer.

14.2.8 Operating the FIFO Data Buffer

The FIFO data buffers of a USIC module are built in a similar way, with transmit buffer and receive buffer capability for each channel. Depending on the device, the amount of available FIFO buffer area can vary. In the XMC1100, totally 64 buffer entries can be distributed among the transmit or receive FIFO buffers of both channels of the USIC module.

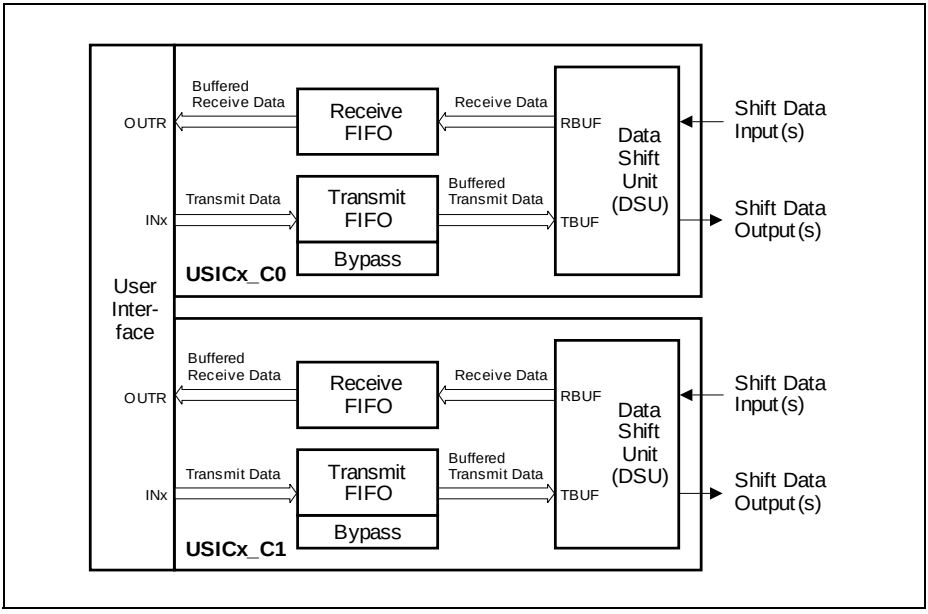


Figure 14-20 FIFO Buffer Overview

In order to operate the FIFO data buffers, the following issues have to be considered:

- FIFO buffer available and selected:
  The transmit FIFO buffer and the bypass structure are only available if CCFG.TB = 1, whereas the receive FIFO buffer is only available if CCFG.RB = 1.
  It is recommended to configure all buffer parameters while there is no data traffic for this USIC channel and the FIFO mechanism is disabled by TBCTR.SIZE = 0 (for transmit buffer) or RBCTR.SIZE = 0 (for receive buffer). The allocation of a buffer area by writing TBCTR or RBCTR has to be done while the corresponding FIFO buffer is disabled. The FIFO buffer interrupt control bits can be modified independently of data traffic.
- FIFO buffer setup:
  The total amount of available FIFO buffer entries limits the length of the transmit and receive buffers for each USIC channel.
- Bypass setup:
  In addition to the transmit FIFO buffer, a bypass can be configured as described on **Page 14-49**.

14.2.8.1 FIFO Buffer Partitioning

If available, the FIFO buffer area consists of a defined number of FIFO buffer entries, each containing a data part and the associated control information (RCI for receive data, TCI for transmit data). One FIFO buffer entry represents the finest granularity that can be allocated to a receive FIFO buffer or a transmit FIFO buffer. All available FIFO buffer entries of a USIC module are located one after the other in the FIFO buffer area. The overall counting starts with FIFO entry 0, followed by 1, 2, etc.

For each USIC module, a certain number of FIFO entries is available, that can be allocated to the channels of the same USIC module. It is not possible to assign FIFO buffer area to USIC channels that are not located within the same USIC module.
For each USIC channel, the size of the transmit and the receive FIFO buffer can be chosen independently. For example, it is possible to allocate the full amount of available FIFO entries as transmit buffer for one USIC channel. Some possible scenarios of FIFO buffer partitioning are shown in **Figure 14-21**.

Each FIFO buffer consists of a set of consecutive FIFO entries. The size of a FIFO data buffer can only be programmed as a power of 2, starting with 2 entries, then 4 entries, then 8 entries, etc. A FIFO data buffer can only start at a FIFO entry aligned to its size. For example, a FIFO buffer containing n entries can only start with FIFO entry 0, n, 2*n, 3*n, etc. and consists of the FIFO entries [x*n, (x+1)*n-1], with x being an integer number (incl. 0). It is not possible to have “holes” with unused FIFO entries within a FIFO buffer, whereas there can be unused FIFO entries between two FIFO buffers.

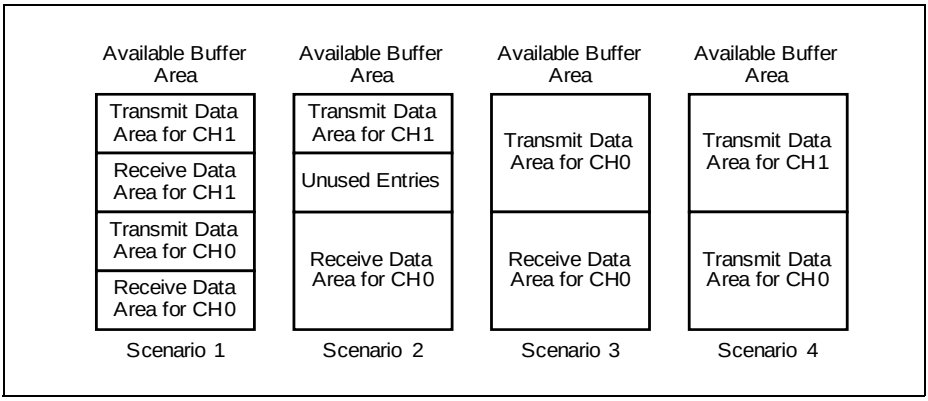


Figure 14-21 FIFO Buffer Partitioning

The data storage inside the FIFO buffers is based on pointers, that are internally updated whenever the data contents of the FIFO buffers have been modified. This happens automatically when new data is put into a FIFO buffer or the oldest data is taken from a FIFO buffer. As a consequence, the user program does not need to modify the pointers for data handling. Only during the initialization phase, the start entry of a FIFO buffer has to be defined by writing the number of the first FIFO buffer entry in the FIFO buffer to the corresponding bit field DPTR in register RBCTR (for a receive FIFO buffer) or TBCTR (for a transmit FIFO buffer) while the related bit field RBCTR.SIZE=0 (or TBCTR.SIZE = 0, respectively). The assignment of buffer entries to a FIFO buffer (regarding to size and pointers) must not be changed by software while the related USIC channel is taking part in data traffic.

14.2.8.2 Transmit Buffer Events and Interrupts

The transmit FIFO buffer mechanism detects the following events, that can lead to interrupts (if enabled):

- Standard transmit buffer event
- Transmit buffer error event

Standard Transmit Buffer Event

The standard transmit buffer event is triggered by the filling level of the transmit buffer (given by TRBSR.TBFLVL) exceeding (TBCTR.LOF = 1) or falling below (TBCTR.LOF = 0)[1)] a programmed limit (TBCTR.LIMIT).

1) If the standard transmit buffer event is used to indicate that new data has to be written to one of the INx locations, TBCTR.LOF = 0 should be programmed.

If the event trigger with TRBSR.STBT feature is disabled (TBCTR.STBTEN = 0), the trigger of the standard transmit buffer event is based on the transition of the fill level from equal to below or above the limit, not the fact of being below or above.

If TBCTR.STBTEN = 1, the transition of the fill level below or above the programmed limit additionally sets TRBSR.STBT. This bit triggers also the standard transmit buffer event whenever there is a transfer data to TBUF event or write data to INx event, depending on TBCTR.LOF setting.

The way TRBSR.STBT is cleared depends on the trigger mode (selected by TBCTR.STBTM). If TBCTR.STBTM = 0, TRBSR.STBT is cleared by hardware when the buffer fill level equals the programmed limit again (TRBSR.TBFLVL = TBCTR.LIMIT). If TBCTR.STBTM = 1, TRBSR.STBT is cleared by hardware when the buffer fill level equals the buffer size (TRBSR.TBFLVL = TBCTR.SIZE).

Note: The flag TRBSR.STBI is set only when the transmit buffer fill level exceeds or falls below the programmed limit (depending on TBCTR.LOF setting). Standard transmit buffer events triggered by TRBSR.STBT does not set the flag.

Figure 14-22 shows examples of the standard transmit buffer event with the different TBCTR.STBTEN and TBCTR.STBTM settings. These examples are meant to illustrate the hardware behaviour and might not always represent real application use cases.

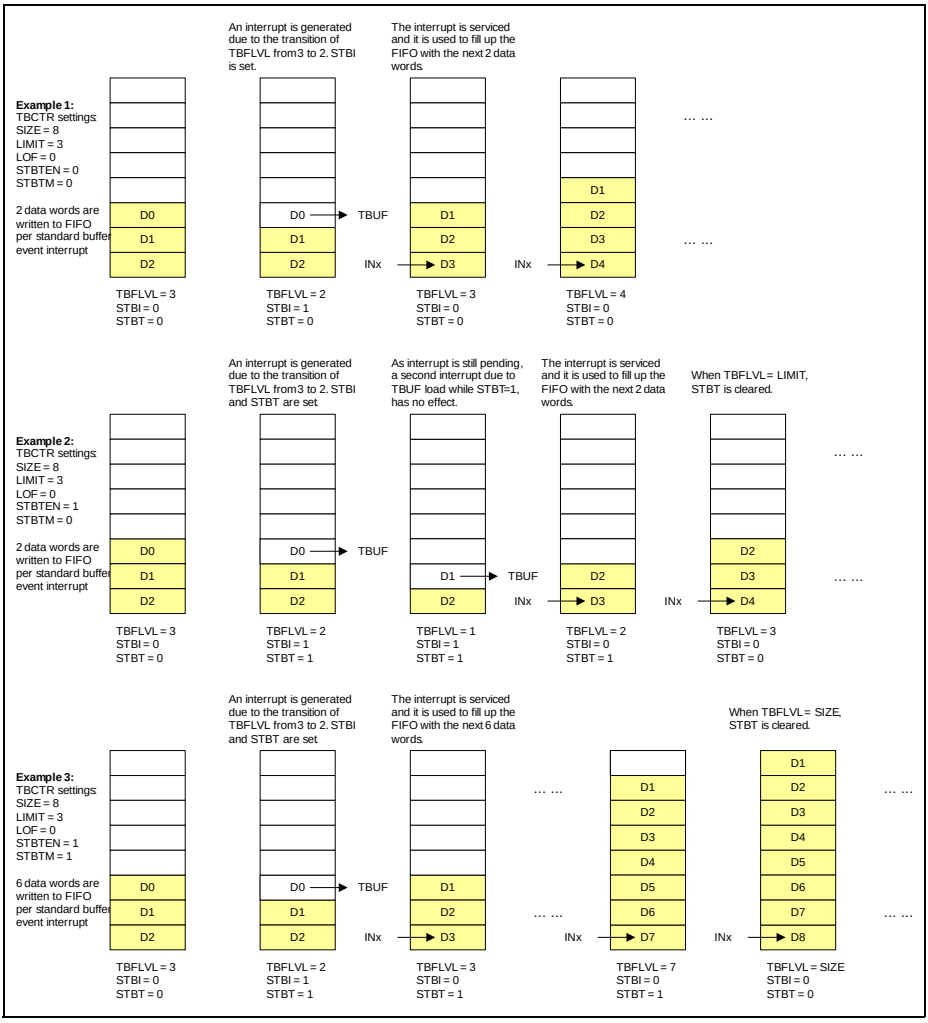


Figure 14-22 Standard Transmit Buffer Event Examples

Transmit Buffer Error Event

The transmit buffer error event is triggered when software has written to a full buffer. The written value is ignored.

Transmit Buffer Events and Interrupt Handling

Figure 14-23 shows the transmit buffer events and interrupts.

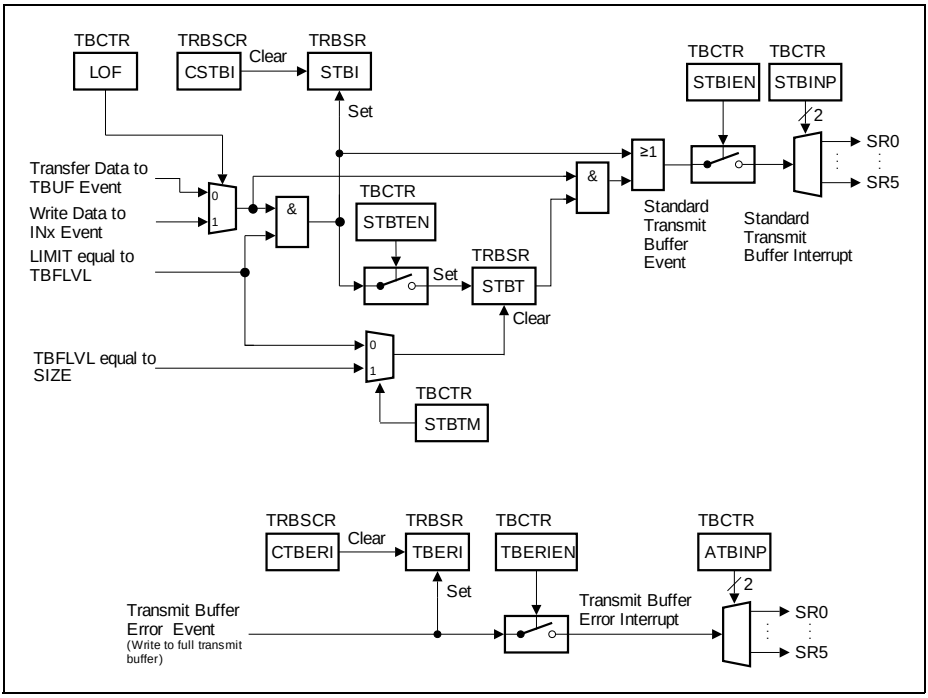


Figure 14-23 Transmit Buffer Events

Table 14-9 shows the registers, bits and bit fields to indicate the transmit buffer events and to control the interrupts related to the transmit FIFO buffers of a USIC channel.

Table 14-9 Transmit Buffer Events and Interrupt Handling

Event	Indication Flag	Indication cleared by	Interrupt enabled by	SRx Output selected by
Standard transmit buffer event	TRBSR.STBI	TRBSCR.CSTBI	TBCTR.STBIEN	TBCTR.STBINP
	TRBSR.STBT	Cleared by hardware		
Transmit buffer error event	TRBSR.TBERI	TRBSCR.CTBERI	TBCTR.TBERIEN	TBCTR.ATBINP

14.2.8.3 Receive Buffer Events and Interrupts

The receive FIFO buffer mechanism detects the following events, that can lead to an interrupt (if enabled):

- Standard receive buffer event
- Alternative receive buffer event
- Receive buffer error event

The standard receive buffer event and the alternative receive buffer event can be programmed to two different modes, one referring to the filling level of the receive buffer, the other one related to a bit position in the receive control information RCI of the data word that becomes available in OUTR.

If the interrupt generation refers to the filling level of the receive FIFO buffer, only the standard receive buffer event is used, whereas the alternative receive buffer event is not used. This mode can be selected to indicate that a certain amount of data has been received, without regarding the content of the associated RCI.

If the interrupt generation refers to RCI, the filling level is not taken into account. Each time a new data word becomes available in OUTR, an event is detected. If bit RCI[4] = 0, a standard receive buffer event is signaled, otherwise an alternative receive buffer device (RCI[4] = 1). Depending on the selected protocol and the setting of RBCTR.RCIM, the value of RCI[4] can hold different information that can be used for protocol-specific interrupt handling (see protocol sections for more details).

Standard Receive Buffer Event in Filling Level Mode

In filling level mode (RBCTR.RNM = 0), the standard receive buffer event is triggered by the filling level of the receive buffer (given by TRBSR.RBFLVL) exceeding (RBCTR.LOF = 1) or falling below (RBCTR.LOF = 0) a programmed limit (RBCTR.LIMIT).[1)]

If the event trigger with bit TRBSR.SRBT feature is disabled (RBCTR.SRBTEN = 0), the trigger of the standard receive buffer event is based on the transition of the fill level from equal to below or above the limit, not the fact of being below or above.

If RBCTR.SRBTEN = 1, the transition of the fill level below or above the programmed limit additionally sets the bit TRBSR.SRBT. This bit also triggers the standard receive buffer event each time there is a data read out event or new data received event, depending on RBCTR.LOF setting.

The way TRBSR.SRBT is cleared depends on the trigger mode (selected by RBCTR.SRBTM). If RBCTR.SRBTM = 0, TRBSR.SRBT is cleared by hardware when the buffer fill level equals the programmed limit again (TRBSR.RBFLVL =

1) If the standard receive buffer event is used to indicate that new data has to be read from OUTR, RBCTR.LOF = 1 should be programmed.

RBCTR.LIMIT). If RBCTR.SRBTM = 1, TRBSR.SRBT is cleared by hardware when the buffer fill level equals 0 (TRBSR.RBFLVL = 0).

Note: The flag TRBSR.SRBI is set only when the receive buffer fill level exceeds or falls below the programmed limit (depending on RBCTR.LOF setting). Standard receive buffer events triggered by TRBSR.SRBT does not set the flag.

Figure 14-24 shows examples of the standard receive buffer event with the different RBCTR.SRBTEN and RBCTR.SRBTM settings. These examples are meant to illustrate the hardware behaviour and might not always represent real application use cases.

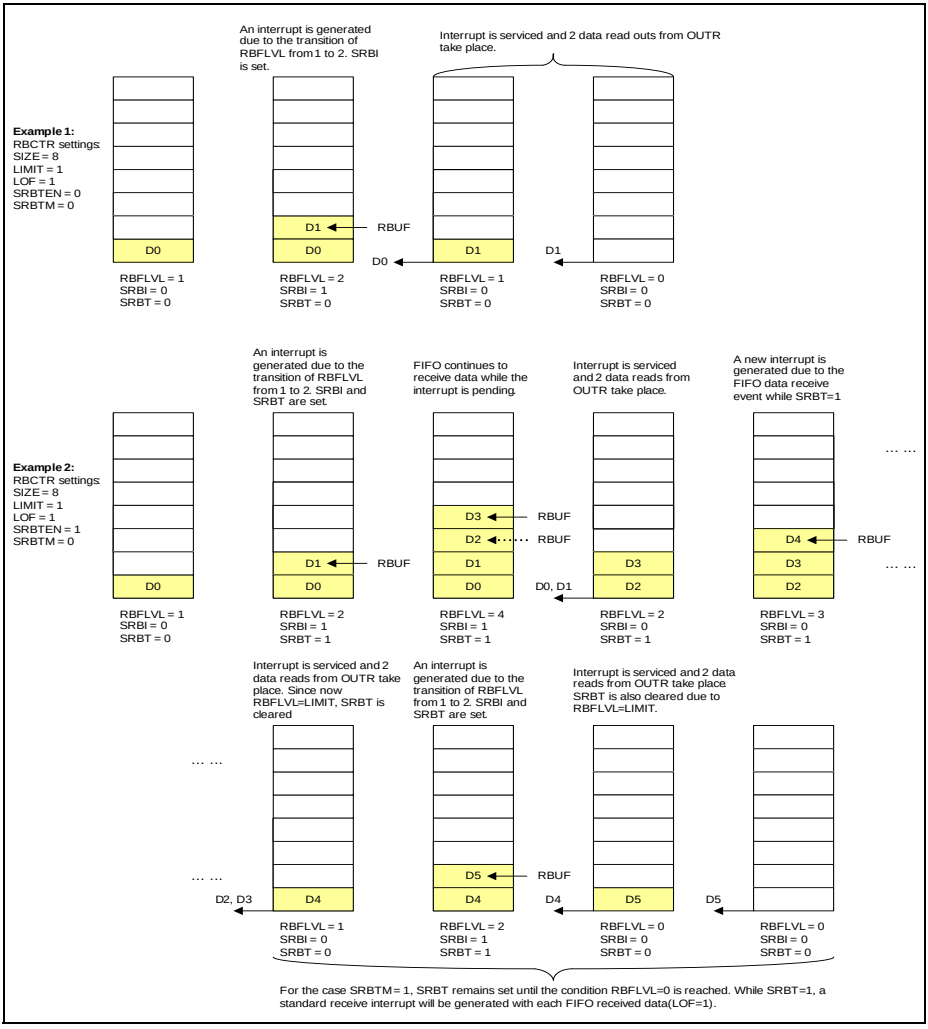


Figure 14-24 Standard Receive Buffer Event Examples

Standard and Alternate Receive Buffer Events in RCI Mode

In RCI mode (RBCTR.RNM = 1), the standard receive buffer event is triggered when the OUTR stage is updated with a new data value with RCI[4] = 0.

If the OUTR stage is updated with a new data value with RCI[4] = 1, an alternate receive buffer event is triggered instead.

Receive Buffer Error Event

The receive buffer error event is triggered if the software reads from an empty buffer, regardless of RBCTR.RNM value. The read data is invalid.

Receive Buffer Events and Interrupt Handling

Figure 14-25 shows the receiver buffer events and interrupts in filling level mode.

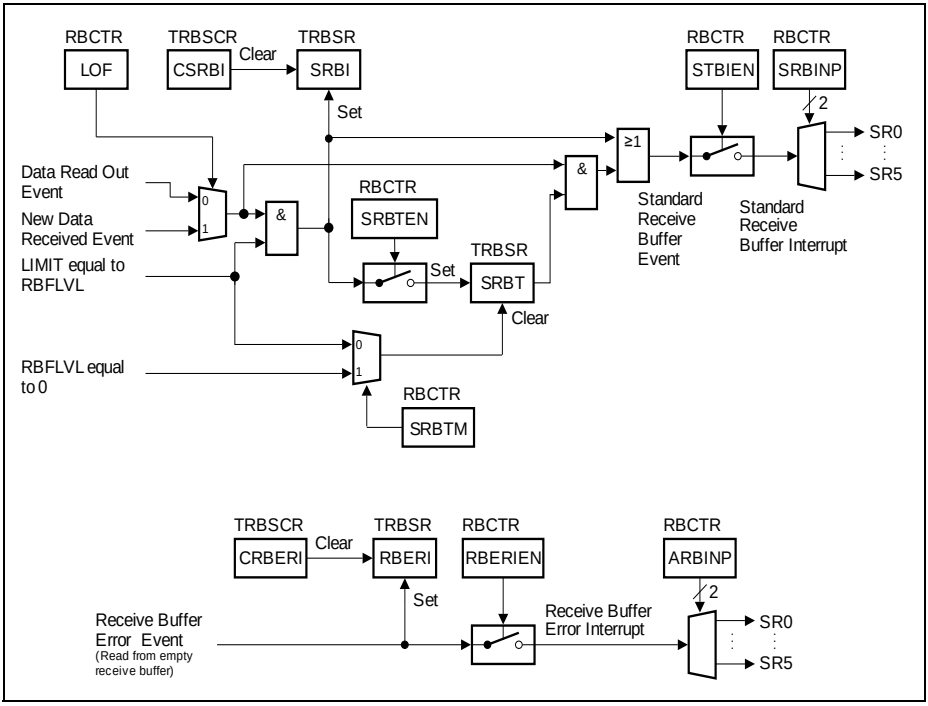


Figure 14-25 Receiver Buffer Events in Filling Level Mode

Figure 14-26 shows the receiver buffer events and interrupts in RCI mode.

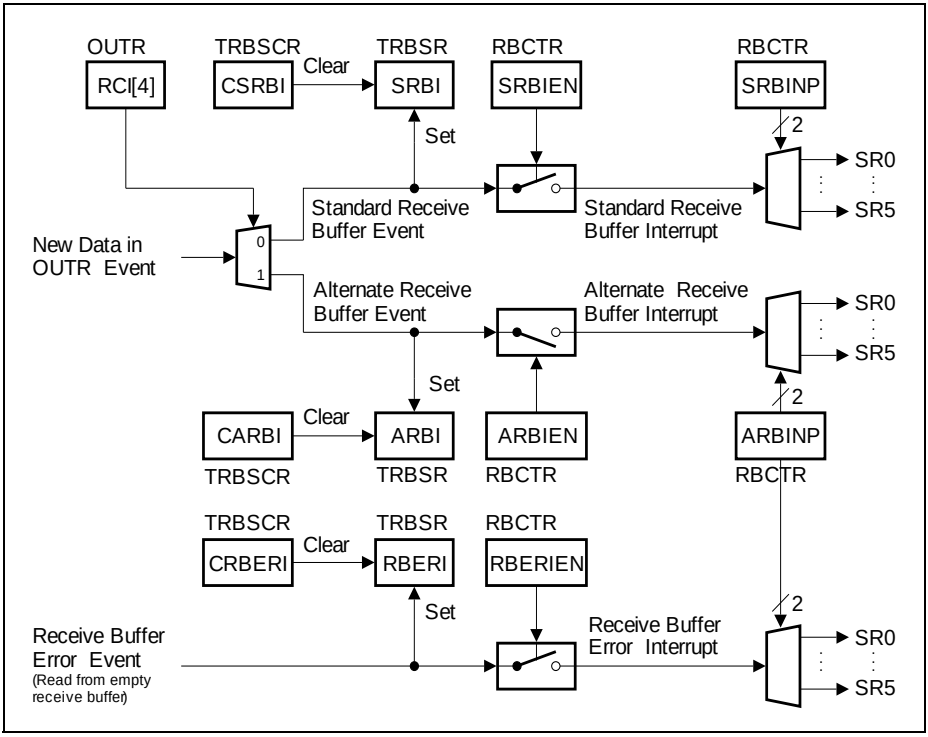


Figure 14-26 Receiver Buffer Events in RCI Mode

Table 14-10 shows the registers, bits and bit fields to indicate the receive buffer events and to control the interrupts related to the receive FIFO buffers of a USIC channel.

Table 14-10 Receive Buffer Events and Interrupt Handling

Event	Indication Flag	Indication cleared by	Interrupt enabled by	SRx Output selected by
Standard receive buffer event	TRBSR. SRBI	TRBSCR. CSRBI	RBCTR. SRBIEN	RBCTR. SRBINP
	TRBSR. SRBT	Cleared by hardware		

Table 14-10 Receive Buffer Events and Interrupt Handling (cont'd)

Event	Indication Flag	Indication cleared by	Interrupt enabled by	SRx Output selected by
Alternative receive buffer event	TRBSR. ARBI	TRBSCR. CARBI	RBCTR. ARBIEN	RBCTR. ARBINP
Receive buffer error event	TRBSR. RBERI	TRBSCR. CRBERI	RBCTR. RBERIEN	RBCTR. ARBINTXDP

14.2.8.4 FIFO Buffer Bypass

The data bypass mechanism is part of the transmit FIFO control block. It allows to introduce a data word in the data stream without modifying the transmit FIFO buffer contents, e.g. to send a high-priority message. The bypass structure consists of a bypass data word of maximum 16 bits in register BYP and some associated control information in register BYPCR. For example, these bits define the word length of the bypass data word and configure a transfer trigger and gating mechanism similar to the one for the transmit buffer TBUF.

The bypass data word can be tagged valid or invalid for transmission by bit BYRCR.BDV (bypass data valid). A combination of data flow related and event related criteria define whether the bypass data word is considered valid for transmission. A data validation logic checks the start conditions for this data word. Depending on the result of the check, the transmit buffer register TBUF is loaded with different values, according to the following rules:

- Data from the transmit FIFO buffer or the bypass data can only be transferred to TBUF if TCSR.TDV = 0 (TBUF is empty).
- Bypass data can only be transferred to TBUF if the bypass is enabled by BYPCR.BDEN or the selecting gating condition is met.
- If the bypass data is valid for transmission and has either a higher transmit priority than the FIFO data or if the transmit FIFO is empty, the bypass data is transferred to TBUF.
- If the bypass data is valid for transmission and has a lower transmit priority than the FIFO buffer that contains valid data, the oldest transmit FIFO data is transferred to TBUF.
- If the bypass data is not valid for transmission and the FIFO buffer contains valid data, the oldest FIFO data is transferred to TBUF.
- If neither the bypass data is valid for transmission nor the transmit FIFO buffer contains valid data, TBUF is unchanged.

The bypass data validation is based on the logic blocks shown in **Figure 14-27**.

- A transfer gating logic enables or disables the bypass data word transfer to TBUF under software or under hardware control. If the input stage DX2 is not needed for

data shifting, signal DX2S can be used for gating purposes. The transfer gating logic is controlled by bit field BYPCR.BDEN.

- A transfer trigger logic supports data word transfers related to events, e.g. timer based or related to an input pin. If the input stage DX2 is not needed for data shifting, signal DX2T can be used for trigger purposes. The transfer trigger logic is controlled by bit BYPCR.BDVTR.
- A bypass data validation logic combining the inputs from the gating logic, the triggering logic and TCSR.TDV.

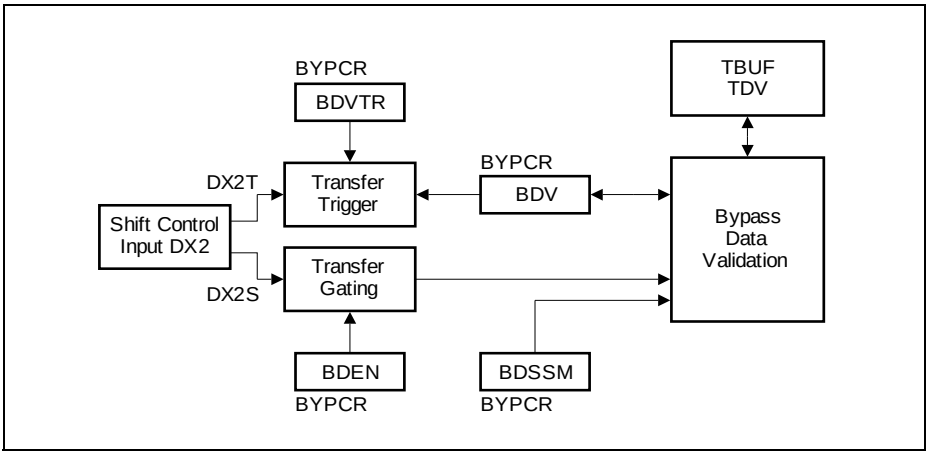


Figure 14-27 Bypass Data Validation

With this structure, the following bypass data transfer functionality can be achieved:

- Bit BYPCR.BDSSM = 1 has to be programmed for a single-shot mechanism. After each transfer of the bypass data word to TBUF, the bypass data word has to be tagged valid again. This can be achieved either by writing a new bypass data word to BYP or by DX2T if BDVTR = 1 (e.g. trigger on a timer base or an edge at a pin).
- Bit BYPCR.BDSSM = 0 has to be programmed if the bypass data is permanently valid for transmission (e.g. as alternative data if the data FIFO runs empty).

14.2.8.5 FIFO Access Constraints

The data in the shared FIFO buffer area is accessed by the hardware mechanisms for data transfer of each communication channel (for transmission and reception) and by software to read out received data or to write data to be transmitted. As a consequence, the data delivery rate can be limited by the FIFO mechanism. Each access by hardware to the FIFO buffer area has priority over a software access, that is delayed in case of an access collision.

In order to avoid data loss and stalling of the CPU due to delayed software accesses, the

baud rate, the word length and the software access mechanism have to be taken into account. Each access to the FIFO data buffer area by software or by hardware takes one period of f_{PB}. Especially a continuous flow of very short, consecutive data words can lead to an access limitation.

14.2.8.6 Handling of FIFO Transmit Control Information

In addition to the transmit data, the transmit control information TCI can be transferred from the transmit FIFO or bypass structure to the USIC channel. Depending on the selected protocol and the enabled update mechanism, some settings of the USIC channel parameters can be modified. The modifications are based on the TCI of the FIFO data word loaded to TBUF or by the bypass control information if the bypass data is loaded into TBUF.

- TCSR.SELMD = 1: update of PCR.CTR[20:16] by FIFO TCI or BYPCR.BSELO with additional clear of PCR.CTR[23:21]
- TCSR.WLEMD = 1: update of SCTR.WLE and TCSR.EOF by FIFO TCI or BYPCR.BWLE (if the WLE information is overwritten by TCI or BWLE, the user has to take care that FLE is set accordingly)
- TCSR.FLEMD = 1: update of SCTR.FLE[4:0] by FIFO TCI or BYPCR.BWLE with additional clear of SCTR.FLE[5]
- TCSR.HPCMD = 1: update of SCTR.DSM and SCTR.HPCDIR by FIFO TCI or BYPCR.BHPC
- TCSR.WAMD = 1: update of TCSR.WA by FIFO TCI[4]

See **Section 14.2.5.3** for more details on TCI.

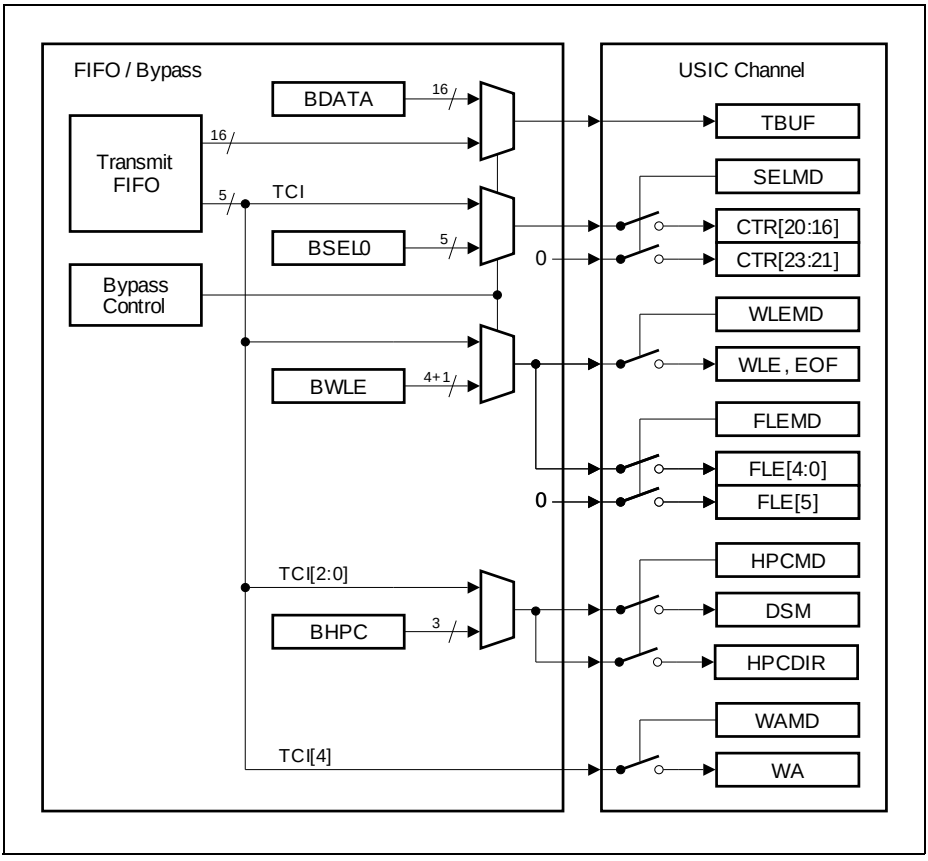


Figure 14-28 TCI Handling with FIFO / Bypass

14.3 Asynchronous Serial Channel (ASC = UART)

The asynchronous serial channel ASC covers the reception and the transmission of asynchronous data frames and provides a hardware LIN support. The receiver and transmitter being independent, frames can start at different points in time for transmission and reception. The ASC mode is selected by CCR.MODE = 0010_B with CCFG.ASC = 1 (ASC mode available).

14.3.1 Signal Description

An ASC connection is characterized by the use of a single connection line between a transmitter and a receiver. The receiver input RXD signal is handled by the input stage DX0.

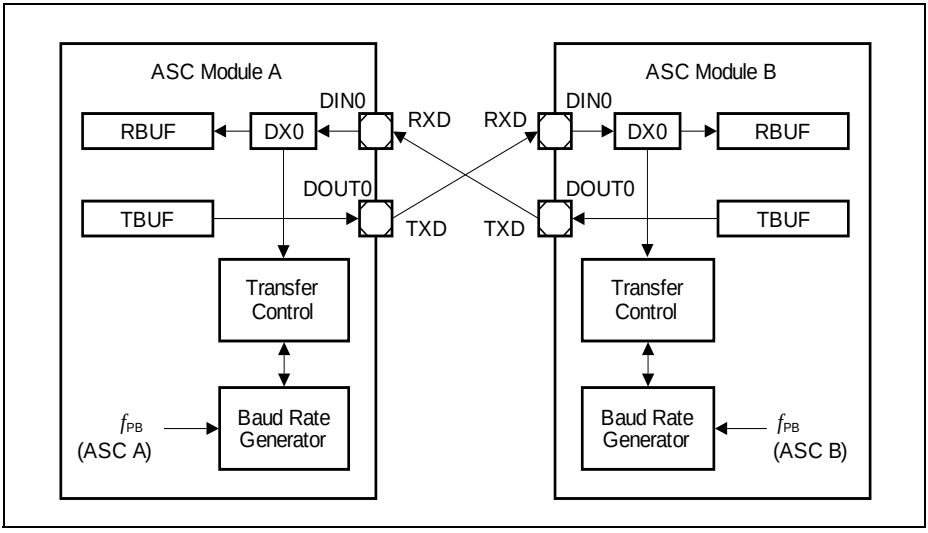


Figure 14-29 ASC Signal Connections for Full-Duplex Communication

For full-duplex communication, an independent communication line is needed for each transfer direction. **Figure 14-29** shows an example with a point-to-point full-duplex connection between two communication partners ASC A and ASC B.

For half-duplex or multi-transmitter communication, a single communication line is shared between the communication partners. **Figure 14-30** shows an example with a point-to-point half-duplex connection between ASC A and ASC B. In this case, the user has to take care that only one transmitter is active at a time. In order to support transmitter collision detection, the input stage DX1 can be used to monitor the level of the transmit line and to check if the line is in the idle state or if a collision occurred.
There are two possibilities to connect the receiver input DIN0 to the transmitter output

DOUT0. Communication partner ASC A uses an internal connection with only the transmit pin TXD, that is delivering its input value as RXD to the DX0 input stage for reception and to DX1 to check for transmitter collisions. Communication partner ASC B uses an external connection between the two pins TXD and RXD.

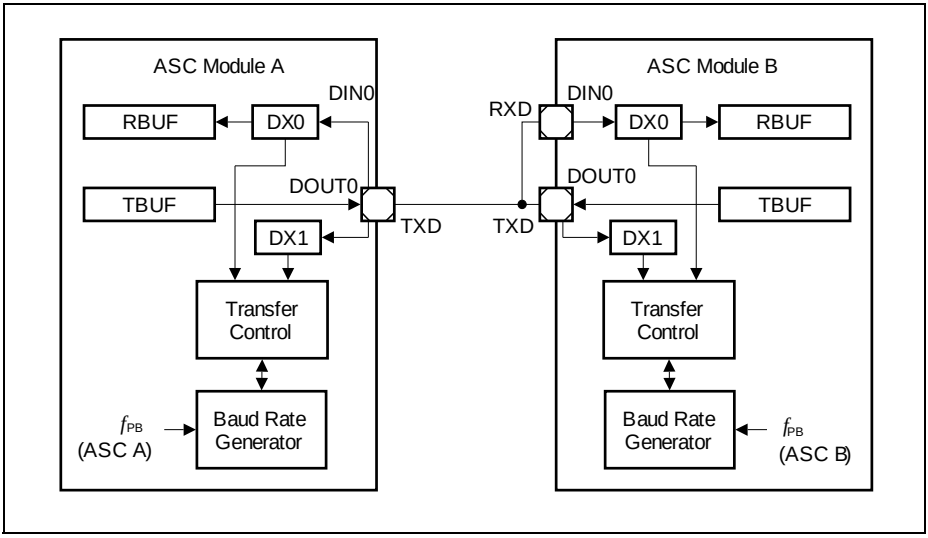


Figure 14-30 ASC Signal Connections for Half-Duplex Communication

14.3.2 Frame Format

A standard ASC frame is shown in **Figure 14-31**. It consists of:

- An idle time with the signal level 1.
- One start of frame bit (SOF) with the signal level 0.
- A data field containing a programmable number of data bits (1-63).
- A parity bit (P), programmable for either even or odd parity. It is optionally possible to handle frames without parity bit.
- One or two stop bits with the signal level 1.

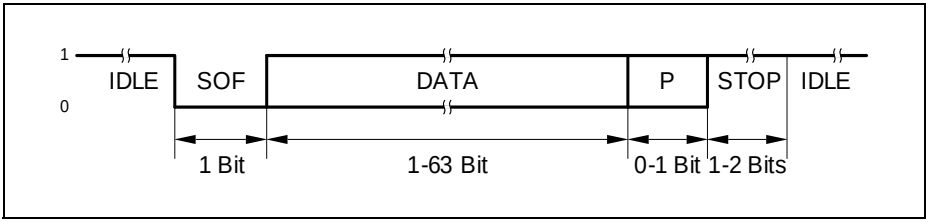


Figure 14-31 Standard ASC Frame Format

The protocol specific bits (SOF, P, STOP) are automatically handled by the ASC protocol state machine and do not appear in the data flow via the receive and transmit buffers.

14.3.2.1 Idle Time

The receiver and the transmitter independently check the respective data input lines (DX0, DX1) for being idle. The idle detection ensures that an SOF bit of a recently enabled ASC module does not collide with an already running frame of another ASC module.

In order to start the idle detection, the user software has to clear bits PSR.RXIDLE and/or PSR.TXIDLE, e.g. before selecting the ASC mode or during operation. If a bit is cleared by software while a data transfer is in progress, the currently running frame transfer is finished normally before starting the idle detection again. Frame reception is only possible if PSR.RXIDLE = 1 and frame transmission is only possible if PSR.TXIDLE = 1. The duration of the idle detection depends on the setting of bit PCR.IDM. In the case that a collision is not possible, the duration can be shortened and the bus can be declared as being idle by setting PCR.IDM = 0.

In the case that the complete idle detection is enabled by PCR.IDM = 1, the data input of DX0 is considered as idle (PSR.RXIDLE becomes set) if a certain number of consecutive passive bit times has been detected. The same scheme applies for the transmitter's data input of DX1. Here, bit PSR.TXIDLE becomes set if the idle condition of this input signal has been detected.

The duration of the complete idle detection is given by the number of programmed data bits per frame plus 2 (in the case without parity) or plus 3 (in the case with parity). The counting of consecutive bit times with 1 level restarts from the beginning each time an edge is found, after leaving a stop mode or if ASC mode becomes enabled.

If the idle detection bits PSR.RXIDLE and/or TXIDLE are cleared by software, the counting scheme is not stopped (no re-start from the beginning). As a result, the cleared bit(s) can become set immediately again if the respective input line still meets the idle criterion.

Please note that the idle time check is based on bit times, so the maximum time can be up to 1 bit time more than programmed value (but not less).

14.3.2.2 Start Bit Detection

The receiver input signal DIN0 (selected signal of input stage DX0) is checked for a falling edge. An SOF bit is detected when a falling edge occurs while the receiver is idle or after the sampling point of the last stop bit. To increase noise immunity, the SOF bit timing starts with the first falling edge that is detected. If the sampled bit value of the SOF is 1, the previous falling edge is considered to be due to noise and the receiver is considered to be idle again.

14.3.2.3 Data Field

The length of the data field (number of data bits) can be programmed by bit field SCTR.FLE. It can vary between 1 and 63 data bits, corresponding to values of SCTR.FLE = 0 to 62 (the value of 63 is reserved and must not be programmed in ASC mode).

The data field can consist of several data words, e.g. a transfer of 12 data bits can be composed of two 8-bit words, with the 12 bits being split into 8-bits of the first word and 4 bits of the second word. The user software has to take care that the transmit data is available in-time, once a frame has been started. If the transmit buffer runs empty during a running data frame, the passive data level (SCTR.PDL) is sent out.

The shift direction can be programmed by SCTR.SDIR. The standard setting for ASC frames with LSB first is achieved with the default setting SDIR = 0.

14.3.2.4 Parity Bit

The ASC allows parity generation for transmission and parity check for reception on frame base. The type of parity can be selected by bit field CCR.PM, common for transmission and reception (no parity, even or odd parity). If the parity handling is disabled, the ASC frame does not contain any parity bit. For consistency reasons, all communication partners have to be programmed to the same parity mode.

After the last data bit of the data field, the transmitter automatically sends out its calculated parity bit if parity generation has been enabled. The receiver interprets this bit as received parity and compares it to its internally calculated one. The received parity bit value and the result of the parity check are monitored in the receiver buffer status registers, RBUFSR and RBUF01SR, as receiver buffer status information. These registers contain bits to monitor a protocol-related argument (PAR) and protocol-related error indication (PERR).

14.3.2.5 Stop Bit(s)

Each ASC frame is completed by 1 or 2 of stop bits with the signal level 1 (same level as the idle level). The number of stop bits is programmable by bit PSR.STPB. A new start bit can be transferred directly after the last stop bit.

14.3.3 Operating the ASC

In order to operate the ASC protocol, the following issues have to be considered:

- Select ASC mode:
  It is recommended to configure all parameters of the ASC that do not change during run time while CCR.MODE = 0000_B. Bit field SCTR.TRM = 01_B has to be programmed. The configuration of the input stages has to be done while CCR.MODE = 0000_B to avoid unintended edges of the input signals and the ASC mode can be enabled by CCR.MODE = 0010_B afterwards.
- Pin connections:
  Establish a connection of input stage DX0 with the selected receive data input pin (signal DIN0) with DX0CR.INSW = 0 and configure a transmit data output pin (signal DOUT0). For collision or idle detection of the transmitter, the input stage DX1 has to be connected to the selected transmit output pin, also with DX1CR.INSW = 0. Additionally, program DX2CR.INSW = 0.
  Due to the handling of the input data stream by the synchronous protocol handler, the propagation delay of the synchronization in the input stage has to be considered.
  Note that the step to enable the alternate output port functions should only be done after the ASC mode is enabled, to avoided unintended spikes on the output.
- Bit timing configuration:
  The desired baud rate setting has to be selected, comprising the fractional divider, the baud rate generator and the bit timing. Please note that not all feature combinations can be supported by the application at the same time, e.g. due to propagation delays. For example, the length of a frame is limited by the frequency difference of the transmitter and the receiver device. Furthermore, in order to use the average of samples (SMD = 1), the sampling point has to be chosen to respect the signal settling and data propagation times.
- Data format configuration:
  The word length, the frame length, and the shift direction have to be set up according to the application requirements by programming the register SCTR. If required by the application, the data input and output signals can be inverted.
  Additionally, the parity mode has to be configured (CCR.PM).

14.3.3.1 Bit Timing

In ASC mode, each bit (incl. protocol bits) is divided into time quanta in order to provide granularity in the sub-bit range to adjust the sample point to the application requirements. The number of time quanta per bit is defined by bit fields BRG.DCTQ and the length of a time quantum is given by BRG.PCTQ.

In the example given in **Figure 14-32**, one bit time is composed of 16 time quanta (BRG.DCTQ = 15). It is not recommended to program less than 4 time quanta per bit time.

Bit field PCR.SP determines the position of the sampling point for the bit value. The value of PCR.SP must not be set to a value greater than BRG.DCTQ. It is possible to sample the bit value only once per bit time or to take the average of samples. Depending on bit PCR.SMD, either the current input value is directly sampled as bit value, or a majority decision over the input values sampled at the latest three time quanta is taken into account. The standard ASC bit timing consists of 16 time quanta with sampling after 8 or 9 time quanta with majority decision.

The bit timing setup (number of time quanta and the sampling point definition) is common for the transmitter and the receiver. Due to independent bit timing blocks, the receiver and the transmitter can be in different time quanta or bit positions inside their frames. The transmission of a frame is aligned to the time quanta generation.

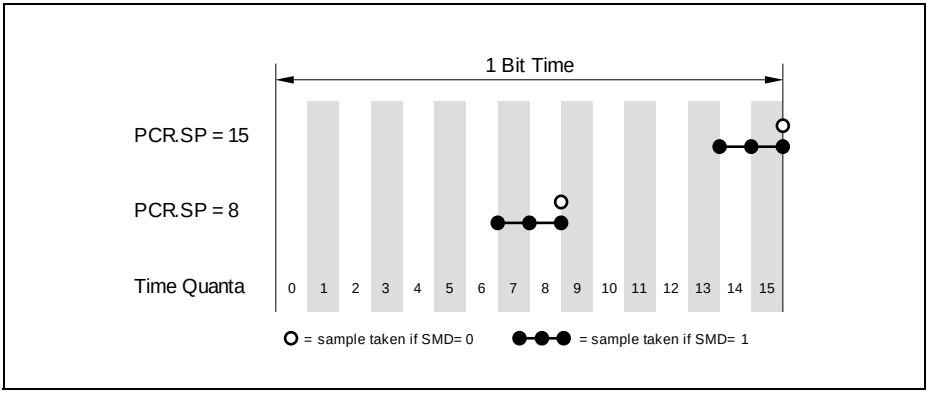


Figure 14-32 ASC Bit Timing

The sample point setting has to be adjusted carefully if collision or idle detection is enabled (via DX1 input signal), because the driver delay and some external delays have to be taken into account. The sample point for the transmit line has to be set to a value where the bit level is stable enough to be evaluated.
If the sample point is located late in the bit time, the signal itself has more time to become stable, but the robustness against differences in the clock frequency of transmitter and receiver decreases.

14.3.3.2 Baud Rate Generation

The baud rate f_{ASC} in ASC mode depends on the number of time quanta per bit time and their timing. The baud rate setting should only be changed while the transmitter and the receiver are idle. The bits in register BRG define the baud rate setting:

- BRG.CTQSEL
  to define the input frequency f_{CTQIN} for the time quanta generation

- BRG.PCTQ
  to define the length of a time quantum (division of f_{CTQIN} by 1, 2, 3, or 4)
- BRG.DCTQ
  to define the number of time quanta per bit time

The standard setting is given by CTQSEL = 00_B ($f_{CTQIN} = f_{PDIV}$) and PPPEN = 0 ($f_{PPP} = f_{PIN}$). Under these conditions, the baud rate is given by:

$$f_{ASC} = f_{PIN} \times \frac{1}{PDIV + 1} \times \frac{1}{PCTQ + 1} \times \frac{1}{DCTQ + 1} \quad (14.6)$$

In order to generate slower frequencies, two additional divide-by-2 stages can be selected by CTQSEL = 10_B ($f_{CTQIN} = f_{SCLK}$) and PPPEN = 1 ($f_{PPP} = f_{MCLK}$), leading to:

$$f_{ASC} = \frac{f_{PIN}}{2 \times 2} \times \frac{1}{PDIV + 1} \times \frac{1}{PCTQ + 1} \times \frac{1}{DCTQ + 1} \quad (14.7)$$

14.3.3.3 Noise Detection

The ASC receiver permanently checks the data input line of the DX0 stage for noise (the check is independent from the setting of bit PCR.SMD). Bit PSR.RNS (receiver noise) becomes set if the three input samples of the majority decision are not identical at the sample point for the bit value. The information about receiver noise gets accumulated over several bits in bit PSR.RNS (it has to be cleared by software) and can trigger a protocol interrupt each time noise is detected if enabled by PCR.RNIEN.

14.3.3.4 Collision Detection

In some applications, such as data transfer over a single data line shared by several sending devices (see **Figure 14-30**), several transmitters have the possibility to send on the same data output line TXD. In order to avoid collisions of transmitters being active at the same time or to allow a kind of arbitration, a collision detection has been implemented.

The data value read at the TXD input at the DX1 stage and the transmitted data bit value are compared after the sampling of each bit value. If enabled by PCR.CDEN = 1 and a bit sent is not equal to the bit read back, a collision is detected and bit PSR.COL is set. If enabled, bit PSR.COL = 1 disables the transmitter (the data output lines become 1) and generates a protocol interrupt. The content of the transmit shift register is considered as invalid, so the transmit buffer has to be programmed again.

14.3.3.5 Pulse Shaping

For some applications, the 0 level of transmitted bits with the bit value 0 is not applied at the transmit output during the complete bit time. Instead of driving the original 0 level,

only a 0 pulse is generated and the remaining time quanta of the bit time are driven with 1 level. The length of a bit time is not changed by the pulse shaping, only the signalling is changed.

In the standard ASC signalling scheme, the 0 level is signalled during the complete bit time with bit value 0 (ensured by programming PCR.PL = 000_B). In the case PCR.PL > 000_B, the transmit output signal becomes 0 for the number of time quanta defined by PCR.PL. In order to support correct reception with pulse shaping by the transmitter, the sample point has to be adjusted in the receiver according to the applied pulse length.

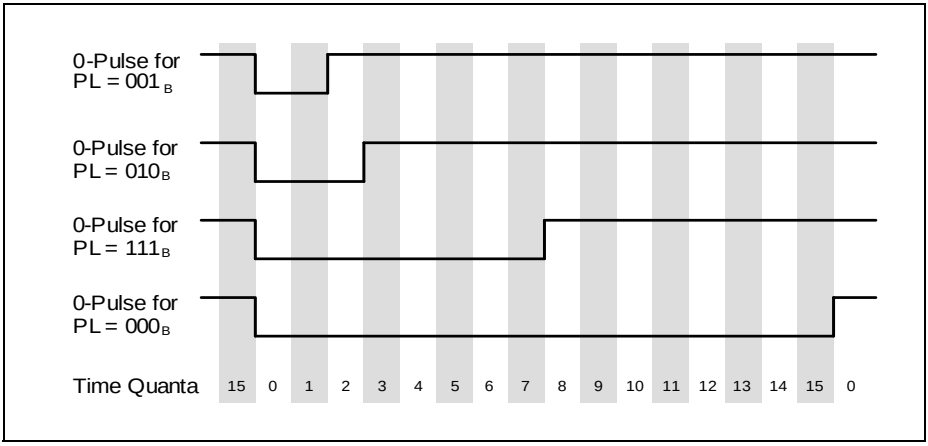


Figure 14-33 Transmitter Pulse Length Control

Figure 14-34 shows an example for the transmission of an 8-bit data word with LSB first and one stop bit (e.g. like for IrDA). The polarity of the transmit output signal has been inverted by SCTR.DOCFG = 01_B.

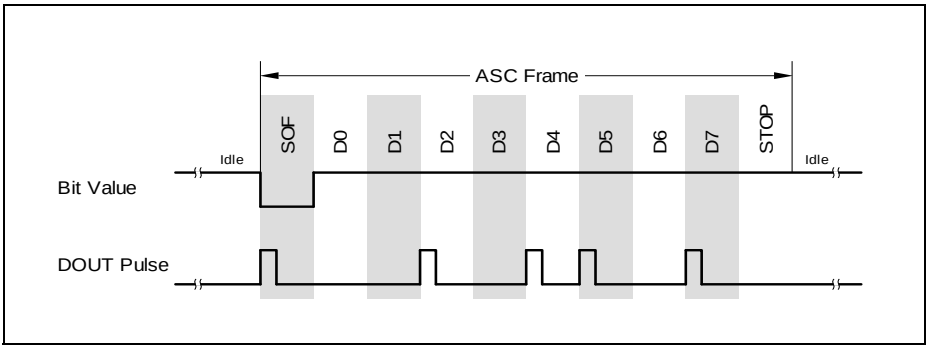


Figure 14-34 Pulse Output Example

14.3.3.6 Automatic Shadow Mechanism

The contents of the protocol control register PCR, as well as bit field SCTR.FLE are internally kept constant while a data frame is transferred by an automatic shadow mechanism (shadowing takes place with each frame start). The registers can be programmed all the time with new settings that are taken into account for the next data frame. During a data frame, the applied (shadowed) setting is not changed, although new values have been written after the start of the data frame.
Bit fields SCTR.WLE and SCTR.SDIR are shadowed automatically with the start of each data word. As a result, a data frame can consist of data words with a different length. It is recommended to change SCTR.SDIR only when no data frame is running to avoid interference between hardware and software.

Please note that the starting point of a data word can be different for a transmitter and a receiver. In order to ensure correct handling, it is recommended to modify SCTR.WLE only while transmitter and receiver are both idle. If the transmitter and the receiver are referring to the same data signal (e.g. in a LIN bus system), SCTR.WLE can be modified while a data transfer is in progress after the RSI event has been detected.

14.3.3.7 End of Frame Control

The number of bits per ASC frame is defined by bit field SCTR.FLE. In order to support different frame length settings for consecutively transmitted frames, this bit field can be modified by hardware. The automatic update mechanism is enabled by TCSR.FLEMD = 1 (in this case, bits TCSR.WLEMD, SELMD, WAMD and HPCMD have to be written with 0).

If enabled, the transmit control information TCI automatically overwrites the bit field TCSR.FLEMD when the ASC frame is started (leading to frames with 1 to 32 data bits). The TCI value represents the written address location of TBUFxx (without additional data

buffer) or INxx (with additional data buffer). With this mechanism, an ASC with 8 data bits is generated by writing a data word to TBUF07 (IN07, respectively).

14.3.3.8 Mode Control Behavior

In ASC mode, the following kernel modes are supported:

- Run Mode 0/1:
  Behavior as programmed, no impact on data transfers.
- Stop Mode 0:
  Bit PSR.TXIDLE is cleared. A new transmission is not started. A current transmission is finished normally. Bit PSR.RXIDLE is not modified. Reception is still possible.
  When leaving stop mode 0, bit TXIDLE is set according to PCR.IDM.
- Stop Mode 1:
  Bit PSR.TXIDLE is cleared. A new transmission is not started. A current transmission is finished normally. Bit PSR.RXIDLE is cleared. A new reception is not possible. A current reception is finished normally.
  When leaving stop mode 1, bits TXIDLE and RXIDLE are set according to PCR.IDM.

14.3.3.9 Disabling ASC Mode

In order to switch off ASC mode without any data corruption, the receiver and the transmitter have to be both idle. This is ensured by requesting Stop Mode 1 in register KSCFG. After waiting for the end of the frame, the ASC mode can be disabled.

14.3.3.10 Protocol Interrupt Events

The following protocol-related events are generated in ASC mode and can lead to a protocol interrupt. The collision detection and the transmitter frame finished events are related to the transmitter, whereas the receiver events are given by the synchronization break detection, the receiver noise detection, the format error checks and the end of the received frame.

Please note that the bits in register PSR are not automatically cleared by hardware and have to be cleared by software in order to monitor new incoming events.

- Collision detection:
  This interrupt indicates that the transmitted value (DOUT0) does not match with the input value of the DX1 input stage at the sample point of a bit. For more details refer to **Page 14-59**.
- Transmitter frame finished:
  This interrupt indicates that the transmitter has completely finished a frame. Bit PSR.TFF becomes set at the end of the last stop bit. The DOUT0 signal assignment to port pins can be changed while no transmission is in progress.
- Receiver frame finished:
  This interrupt indicates that the receiver has completely finished a frame. Bit

PSR.RFF becomes set at the end of the last stop bit. The DIN0 signal assignment to port pins can be changed while no reception is in progress.

- Synchronization break detection:
  This interrupt can be used in LIN networks to indicate the reception of the synchronization break symbol (at the beginning of a LIN frame).
- Receiver noise detection:
  This interrupt indicates that the input value at the sample point of a bit and at the two time quanta before are not identical.
- Format error:
  The bit value of the stop bit(s) is defined as 1 level for the ASC protocol. A format error is signalled if the sampled bit value of a stop bit is 0.

14.3.3.11 Data Transfer Interrupt Handling

The data transfer interrupts indicate events related to ASC frame handling.

- Transmit buffer interrupt TBI:
  Bit PSR.TBIF is set after the start of first data bit of a data word. This is the earliest point in time when a new data word can be written to TBUF.
  With this event, bit TCSR.TDV is cleared and new data can be loaded to the transmit buffer.
- Transmit shift interrupt TSI:
  Bit PSR.TSIF is set after the start of the last data bit of a data word.
- Receiver start interrupt RSI:
  Bit PSR.RSIF is set after the sample point of the first data bit of a data word.
- Receiver interrupt RI and alternative interrupt AI:
  Bit PSR.RIF is set after the sampling point of the last data bit of a data word if this data word is not directly followed by a parity bit (parity generation disabled or not the last word of a data frame).
  If the data word is directly followed by a parity bit (last data word of a data frame and parity generation enabled), bit PSR.RIF is set after the sampling point of the parity bit if no parity error has been detected. If a parity error has been detected, bit PSR.AIF is set instead of bit PSR.RIF.
  The first data word of a data frame is indicated by RBUFSR.SOF = 1 for the received word.
  Bit PSR.RIF is set for a receiver interrupt RI with WA = 0. Bit PSR.AIF is set for a alternative interrupt AI with WA = 1.

14.3.3.12 Baud Rate Generator Interrupt Handling

The baud rate generator interrupt indicate that the capture mode timer has reached its maximum value. With this event, the bit PSR.BRGIF is set.

14.3.3.13 Protocol-Related Argument and Error

The protocol-related argument (RBUFSR.PAR) and the protocol-related error (RBUFSR.PERR) are two flags that are assigned to each received data word in the corresponding receiver buffer status registers.

In ASC mode, the received parity bit is monitored by the protocol-related argument and the result of the parity check by the protocol-related error indication (0 = received parity bit equal to calculated parity value). This information being elaborated only for the last received data word of each data frame, both bit positions are 0 for data words that are not the last data word of a data frame or if the parity generation is disabled.

14.3.3.14 Receive Buffer Handling

If a receive FIFO buffer is available (CCFG.RB = 1) and enabled for data handling (RBCTR.SIZE > 0), it is recommended to set RBCTR.RCIM = 11_B in ASC mode. This leads to an indication that the data word has been the first data word of a new data frame if bit OUTR.RCI[0] = 1, a parity error is indicated by OUTR.RCI[4] = 1, and the received parity bit value is given by OUTR.RCI[3].

The standard receive buffer event and the alternative receive buffer event can be used for the following operations in RCI mode (RBCTR.RNM = 1):

- A standard receive buffer event indicates that a data word can be read from OUTR that has been received without parity error.
- An alternative receive buffer event indicates that a data word can be read from OUTR that has been received with parity error.

14.3.3.15 Sync-Break Detection

The receiver permanently checks the DIN0 signal for a certain number of consecutive bit times with 0 level. The number is given by the number of programmed bits per frame (SCTR.FLE) plus 2 (in the case without parity) or plus 3 (in the case with parity). If a 0 level is detected at a sample point of a bit after this event has been found, bit PSR.SBD is set and additionally, a protocol interrupt can be generated (if enabled by PCR.SBD = 1). The counting restarts from 0 each time a falling edge is found at input DIN0. This feature can be used for the detection of a synchronization break for slave devices in a LIN bus system (the master does not check for sync break).
For example, in a configuration for 8 data bits without parity generation, bit PCR.SBD is set after at the next sample point at 0 level after 10 complete bit times have elapsed (representing the sample point of the 11th bit time since the first falling edge).

14.3.3.16 Transfer Status Indication

The receiver status can be monitored by flag PSR[9] = BUSY if bit PCR.CTR[16] (receiver status enable RSTEN) is set. In this case, bit BUSY is set during a complete frame reception from the beginning of the start of frame bit to the end of the last stop bit.

The transmitter status can be monitored by flag PSR[9] = BUSY if bit PCR.CTR[17] (transmitter status enable TSTEN) is set. In this case, bit BUSY is set during a complete frame reception from the beginning of the start of frame bit to the end of the last stop bit. If both bits RSTEN and TSTEN are set, flag BUSY indicates the logical OR-combination of the receiver and the transmitter status. If both bits are cleared, flag BUSY is not modified depending on the transfer status (status changes are ignored).

14.3.4 ASC Protocol Registers

In ASC mode, the registers PCR and PSR handle ASC related information.

14.3.4.1 ASC Protocol Control Register

In ASC mode, the PCR register bits or bit fields are defined as described in this section.

PCR
Protocol Control Register [ASC Mode]

($3C_H$) **Reset Value: $0000\ 0000_H$**

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PL			SP					FFIEN	FEIEN	RNIEN	CDEN	SBIEN	IDM	STPB	SMD
rw			rw					rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
SMD	0	rw	**Sample Mode** This bit field defines the sample mode of the ASC receiver. The selected data input signal can be sampled only once per bit time or three times (in consecutive time quanta). When sampling three times, the bit value shifted in the receiver shift register is given by a majority decision among the three sampled values. 0_B Only one sample is taken per bit time. The current input value is sampled. 1_B Three samples are taken per bit time and a majority decision is made.

Field	Bits	Type	Description
STPB	1	rw	**Stop Bits** This bit defines the number of stop bits in an ASC frame. 0_B The number of stop bits is 1. 1_B The number of stop bits is 2.
IDM	2	rw	**Idle Detection Mode** This bit defines if the idle detection is switched off or based on the frame length. 0_B The bus idle detection is switched off and bits PSR.TXIDLE and PSR.RXIDLE are set automatically to enable data transfers without checking the inputs before. 1_B The bus is considered as idle after a number of consecutive passive bit times defined by SCTR.FLE plus 2 (in the case without parity bit) or plus 3 (in the case with parity bit).
SBIEN	3	rw	**Synchronization Break Interrupt Enable** This bit enables the generation of a protocol interrupt if a synchronization break is detected. The automatic detection is always active, so bit SBD can be set independently of SBIEN. 0_B The interrupt generation is disabled. 1_B The interrupt generation is enabled.
CDEN	4	rw	**Collision Detection Enable** This bit enables the reaction of a transmitter to the collision detection. 0_B The collision detection is disabled. 1_B If a collision is detected, the transmitter stops its data transmission, outputs a 1, sets bit PSR.COL and generates a protocol interrupt. In order to allow data transmission again, PSR.COL has to be cleared by software.
RNIEN	5	rw	**Receiver Noise Detection Interrupt Enable** This bit enables the generation of a protocol interrupt if receiver noise is detected. The automatic detection is always active, so bit PSR.RNS can be set independently of PCR.RNIEN. 0_B The interrupt generation is disabled. 1_B The interrupt generation is enabled.

Field	Bits	Type	Description
FEIEN	6	rw	**Format Error Interrupt Enable** This bit enables the generation of a protocol interrupt if a format error is detected. The automatic detection is always active, so bits PSR.FER0/FER1 can be set independently of PCR.FEIEN. 0_B The interrupt generation is disabled. 1_B The interrupt generation is enabled.
FFIEN	7	rw	**Frame Finished Interrupt Enable** This bit enables the generation of a protocol interrupt if the receiver or the transmitter reach the end of a frame. The automatic detection is always active, so bits PSR.RFF or PSR.TFF can be set independently of PCR.FFIEN. 0_B The interrupt generation is disabled. 1_B The interrupt generation is enabled.
SP	[12:8]	rw	**Sample Point** This bit field defines the sample point of the bit value. The sample point must not be located outside the programmed bit timing (PCR.SP ≤ BRG.DCTQ).
PL	[15:13]	rw	**Pulse Length** This bit field defines the length of a 0 data bit, counted in time quanta, starting with the time quantum 0 of each bit time. Each bit value that is a 0 can lead to a 0 pulse that is shorter than a bit time, e.g. for IrDA applications. The length of a bit time is not changed by PL, only the length of the 0 at the output signal. The pulse length must not be longer than the programmed bit timing (PCR.PL ≤ BRG.DCTQ). This bit field is only taken into account by the transmitter and is ignored by the receiver. 000_B The pulse length is equal to the bit length (no shortened 0). 001_B The pulse length of a 0 bit is 2 time quanta. 010_B The pulse length of a 0 bit is 3 time quanta. ... 111_B The pulse length of a 0 bit is 8 time quanta.

Field	Bits	Type	Description
RSTEN	16	rw	**Receiver Status Enable** This bit enables the modification of flag PSR[9] = BUSY according to the receiver status. 0_B Flag PSR[9] is not modified depending on the receiver status. 1_B Flag PSR[9] is set during the complete reception of a frame.
TSTEN	17	rw	**Transmitter Status Enable** This bit enables the modification of flag PSR[9] = BUSY according to the transmitter status. 0_B Flag PSR[9] is not modified depending on the transmitter status. 1_B Flag PSR[9] is set during the complete transmission of a frame.
MCLK	31	rw	**Master Clock Enable** This bit enables the generation of the master clock MCLK. 0_B The MCLK generation is disabled and the MCLK signal is 0. 1_B The MCLK generation is enabled.
0	[30:18]	r	**Reserved** Returns 0 if read; should be written with 0.

14.3.4.2 ASC Protocol Status Register

In ASC mode, the PSR register bits or bit fields are defined as described in this section. The bits and bit fields in register PSR are not cleared by hardware.

The flags in the PSR register can be cleared by writing a 1 to the corresponding bit position in register PSCR. Writing a 1 to a bit position in PSR sets the corresponding flag, but does not lead to further actions (no interrupt generation). Writing a 0 has no effect. The PSR flags should be cleared by software before enabling a new protocol.

PSR
Protocol Status Register [ASC Mode] (48_H) **Reset Value: $0000\ 0000_H$**

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AIF	RIF	TBIF	TSIF	DLIF	RSIF	BUSY	TFF	RFF	FER1	FER0	RNS	COL	SBD	RXIDLE	TXIDLE
rwh	rwh	rwh	rwh	rwh	rwh	r	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
TXIDLE	0	rwh	**Transmission Idle** This bit shows if the transmit line (DX1) has been idle. A frame transmission can only be started if TXIDLE is set. 0_B The transmitter line has not yet been idle. 1_B The transmitter line has been idle and frame transmission is possible.
RXIDLE	1	rwh	**Reception Idle** This bit shows if the receive line (DX0) has been idle. A frame reception can only be started if RXIDLE is set. 0_B The receiver line has not yet been idle. 1_B The receiver line has been idle and frame reception is possible.
SBD	2	rwh	**Synchronization Break Detected**[1)] This bit is set if a programmed number of consecutive bit values with level 0 has been detected (called synchronization break, e.g. in a LIN bus system). 0_B A synchronization break has not yet been detected. 1_B A synchronization break has been detected.
COL	3	rwh	**Collision Detected**[1)] This bit is set if a collision has been detected (with PCR.CDEN = 1). 0_B A collision has not yet been detected and frame transmission is possible. 1_B A collision has been detected and frame transmission is not possible.

Field	Bits	Type	Description
RNS	4	rwh	**Receiver Noise Detected**[1)] This bit is set if receiver noise has been detected. 0_B Receiver noise has not been detected. 1_B Receiver noise has been detected.
FER0	5	rwh	**Format Error in Stop Bit 0**[1)] This bit is set if a 0 has been sampled in the stop bit 0 (called format error 0). 0_B A format error 0 has not been detected. 1_B A format error 0 has been detected.
FER1	6	rwh	**Format Error in Stop Bit 1**[1)] This bit is set if a 0 has been sampled in the stop bit 1 (called format error 1). 0_B A format error 1 has not been detected. 1_B A format error 1 has been detected.
RFF	7	rwh	**Receive Frame Finished**[1)] This bit is set if the receiver has finished the last stop bit. 0_B The received frame is not yet finished. 1_B The received frame is finished.
TFF	8	rwh	**Transmitter Frame Finished**[1)] This bit is set if the transmitter has finished the last stop bit. 0_B The transmitter frame is not yet finished. 1_B The transmitter frame is finished.
BUSY	9	r	**Transfer Status BUSY** This bit indicates the receiver status (if PCR.RSTEN = 1) or the transmitter status (if PCR.TSTEN = 1) or the logical OR combination of both (if PCR.RSTEN = PCR.TSTEN = 1). 0_B A data transfer does not take place. 1_B A data transfer currently takes place.
RSIF	10	rwh	**Receiver Start Indication Flag** 0_B A receiver start event has not occurred. 1_B A receiver start event has occurred.
DLIF	11	rwh	**Data Lost Indication Flag** 0_B A data lost event has not occurred. 1_B A data lost event has occurred.
TSIF	12	rwh	**Transmit Shift Indication Flag** 0_B A transmit shift event has not occurred. 1_B A transmit shift event has occurred.

Field	Bits	Type	Description
TBIF	13	rwh	**Transmit Buffer Indication Flag** 0_B A transmit buffer event has not occurred. 1_B A transmit buffer event has occurred.
RIF	14	rwh	**Receive Indication Flag** 0_B A receive event has not occurred. 1_B A receive event has occurred.
AIF	15	rwh	**Alternative Receive Indication Flag** 0_B An alternative receive event has not occurred. 1_B An alternative receive event has occurred.
BRGIF	16	rwh	**Baud Rate Generator Indication Flag** 0_B A baud rate generator event has not occurred. 1_B A baud rate generator event has occurred.
0	[31:17]	r	**Reserved** Returns 0 if read; should be written with 0.

1) This status bit can generate a protocol interrupt (see **Page 14-21**). The general interrupt status flags are described in the general interrupt chapter.

14.3.5 Hardware LIN Support

In order to support the LIN protocol, bit TCSR.FLEMD = 1 should be set for the master. For slave devices, it can be cleared and the fixed number of 8 data bits has to be set (SCTR.FLE = 7_H). For both, master and slave devices, the parity generation has to be switched off (CCR.PM = 00_B) and transfers take place with LSB first (SCTR.SDIR = 0) and 1 stop bit (PCR.STPB = 0).

The Local Interconnect Network (LIN) data exchange protocol contains several symbols that can all be handled in ASC mode. Each single LIN symbol represents a complete ASC frame. The LIN bus is a master-slave bus system with a single master and multiple slaves (for the exact definition please refer to the official LIN specification).

A complete LIN frame contains the following symbols:

- Synchronization break:
  The master sends a synchronization break to signal the beginning of a new frame. It contains at least 13 consecutive bit times at 0 level, followed by at least one bit time at 1 level (corresponding to 1 stop bit). Therefore, TBUF11 if the transmit buffer is used, (or IN11 if the FIFO buffer is used) has to be written with 0 (leading to a frame with SOF followed by 12 data bits at 0 level).
  A slave device shall detect 11 consecutive bit times at 0 level, which done by the synchronization break detection. Bit PSR.SBD is set if such an event is detected and a protocol interrupt can be generated. Additionally, the received data value of 0 appears in the receive buffer and a format error is signaled.

If the baud rate of the slave has to be adapted to the master, the baud rate measurement has to be enabled for falling edges by setting BRG.TMEN = 1, DX0CR.CM = 10_H and DX1CR.CM = 00_H before the next symbol starts.

- Synchronization byte:
  The master sends this symbol after writing the data value 55_H to TBUF07 (or IN07). A slave device can either receive this symbol without any further action (and can discard it) or it can use the falling edges for baud rate measurement. Bit PSR.TSIF = 1 (with optionally the corresponding interrupt) indicates the detection of a falling edge and the capturing of the elapsed time since the last falling edge in CMTR.CTV. Valid captured values can be read out after the second, third, fourth and fifth activation of TSIF. After the fifth activation of TSIF within this symbol, the baud rate detection can be disabled (BRG.TMEN = 0) and BRG.PDIV can be programmed with the captured CMTR.CTV value divided by twice the number of time quanta per bit (assuming BRG.PCTQ = 00_B).
- Other symbols:
  The other symbols of a LIN frame can be handled with ASC data frames without specific actions.

If LIN frames should be sent out on a frame base by the LIN master, the input DX2 can be connected to external timers to trigger the transmit actions (e.g. the synchronization break symbol has been prepared but is started if a trigger occurs). Please note that during the baud rate measurement of the ASC receiver, the ASC transmitter of the same USIC channel can still perform a transmission.

14.4 Synchronous Serial Channel (SSC)

The synchronous serial channel SSC covers the data transfer function of an SPI-like module. It can handle reception and transmission of synchronous data frames between a device operating in master mode and at least one device in slave mode. Besides the standard SSC protocol consisting of one input and one output data line, SSC protocols with two (Dual-SSC) or four (Quad-SSC) input/output data lines are also supported. The SSC mode is selected by CCR.MODE = 0001_B with CCFG.SSC = 1 (SSC mode is available).

14.4.1 Signal Description

A synchronous SSC data transfer is characterized by a simultaneous transfer of a shift clock signal together with the transmit and/or receive data signal(s) to determine when the data is valid (definition of transmit and sample point).

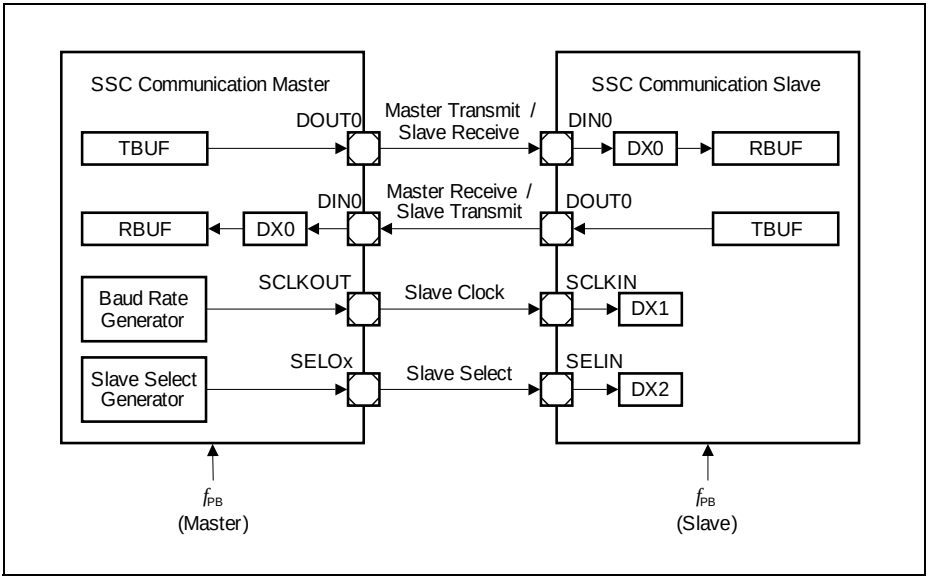


Figure 14-35 SSC Signals for Standard Full-Duplex Communication

In order to explicitly indicate the start and the end of a data transfer and to address more than one slave devices individually, the SSC module supports the handling of slave select signals. They are optional and are not necessarily needed for SSC data transfers. The SSC module supports up to 8 different slave select output signals for master mode operation (named SELOx, with x = 0-7) and 1 slave select input SELIN for slave mode. In most applications, the slave select signals are active low.

A device operating in master mode controls the start and end of a data frame, as well as the generation of the shift clock and slave select signals. This comprises the baud rate setting for the shift clock and the delays between the shift clock and the slave select output signals. If several SSC modules are connected together, there can be only one SSC master at a time, but several slaves. Slave devices receive the shift clock and optionally a slave select signal(s). For the programming of the input stages DXn please refer to **Page 14-21**.

Table 14-11 SSC Communication Signals

SSC Mode	Receive Data	Transmit Data	Shift Clock	Slave Select(s)
Standard SSC Master	MRST[1)], input DIN0, handled by DX0	MTSR[2)], Output DOUT0	Output SCLKOUT	Output(s) SELOx
Standard SSC Slave	MTSR, input DIN0, handled by DX0	MRST, Output DOUT0	Input SCLKIN, handled by DX1	input SELIN, handled by DX2
Dual-SSC Master	MRST[1:0], input DIN[1:0], handled by DX0 and DX3	MTSR[1:0], Output DOUT[1:0]	Output SCLKOUT	Output(s) SELOx
Dual-SSC Slave	MTSR[1:0], input DIN[1:0], handled by DX0 and DX3	MRST[1:0], Output DOUT[1:0]	Input SCLKIN, handled by DX1	input SELIN, handled by DX2
Quad-SSC Master	MRST[3:0], input DIN[3:0], handled by DX0, DX3, DX4 and DX5	MTSR[3:0], Output DOUT[3:0]	Output SCLKOUT	Output(s) SELOx
Quad-SSC Slave	MTSR[3:0], input DIN[3:0], handled by DX0, DX3, DX4 and DX5	MRST[3:0], Output DOUT[3:0]	Input SCLKIN, handled by DX1	input SELIN, handled by DX2

1) MRST = master receive slave transmit, also known as MISO = master in slave out

2) MTSR = master transmit slave receive, also known as MOSI = master out slave in

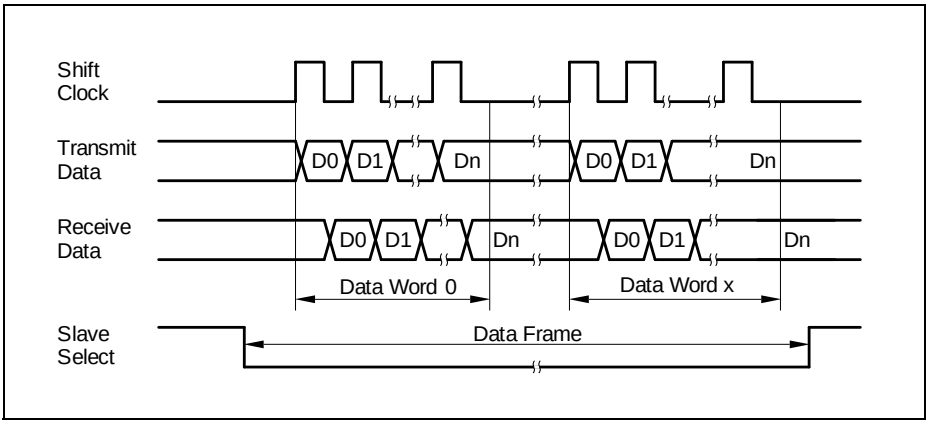


Figure 14-36 4-Wire SSC Standard Communication Signals

14.4.1.1 Transmit and Receive Data Signals

In standard SSC half-duplex mode, a single data line is used, either for data transfer from the master to a slave or from a slave to the master. In this case, MRST and MTSR are connected together, one signal as input, the other one as output, depending on the data direction. The user software has to take care about the data direction to avoid data collision (e.g. by preparing dummy data of all 1s for transmission in case of a wired AND connection with open-drain drivers, by enabling/disabling push/pull output drivers or by switching pin direction with hardware port control enabled). In full-duplex mode, data transfers take place in parallel between the master device and a slave device via two independent data signals MTSR and MRST, as shown in **Figure 14-35**.

The receive data input signal DIN0 is handled by the input stage DX0. In master mode (referring to MRST) as well as in slave mode (referring to MTSR), the data input signal DIN0 is taken from an input pin. The signal polarity of DOUT0 (data output) with respect to the data bit value can be configured in block DOCFG (data output configuration) by bit field SCTR.DOCFG.

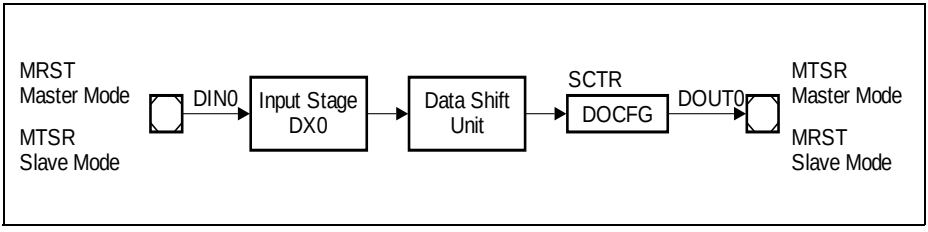


Figure 14-37 SSC Data Signals

For dual- and quad-SSC modes that require multiple input and output data lines to be used, additional input stages, DINx and DOUTx signals need to be set up.

14.4.1.2 Shift Clock Signals

The shift clock signal is handled by the input stage DX1. In slave mode, the signal SCLKIN is received from an external master, so the DX1 stage has to be connected to an input pin. The input stage can invert the received input signal to adapt to the polarity of SCLKIN to the function of the data shift unit (data transmission on rising edges, data reception on falling edges).

In master mode, the shift clock is generated by the internal baud rate generator. The output signal SCLK of the baud rate generator is taken as shift clock input for the data shift unit. The internal signal SCLK is made available for external slave devices by signal SCLKOUT. For complete closed loop delay compensation in a slave mode, SCLKOUT can also take the transmit shift clock from the input stage DX1. The selection is done through the bit BRG.SCLKOSEL. See **Section 14.4.6.3**.

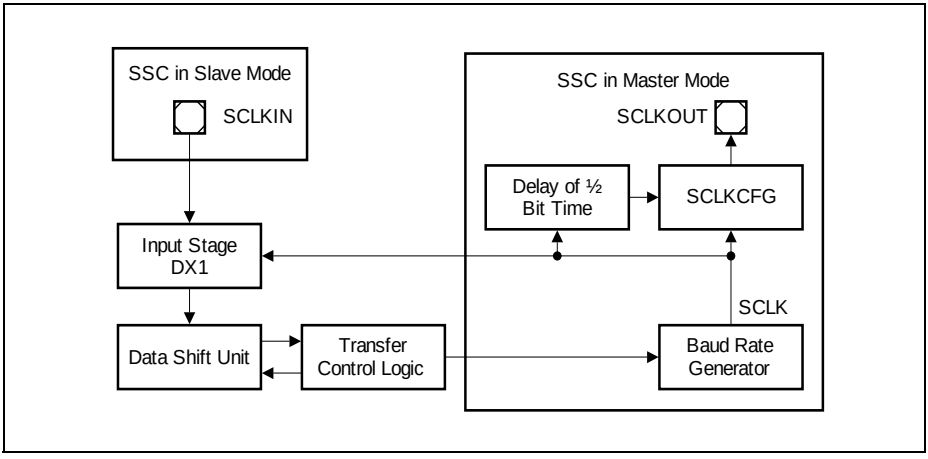


Figure 14-38 SSC Shift Clock Signals

Due to the multitude of different SSC applications, in master mode, there are different ways to configure the shift clock output signal SCLKOUT with respect to SCLK. This is done in the block SCLKCFG (shift clock configuration) by bit field BRG.SCLKCFG, allowing 4 possible settings, as shown in **Figure 14-39**.

- No delay, no polarity inversion (SCLKCFG = 00_B, SCLKOUT equals SCLK):
  The inactive level of SCLKOUT is 0, while no data frame is transferred. The first data bit of a new data frame is transmitted with the first rising edge of SCLKOUT and the first data bit is received in with the first falling edge of SCLKOUT. The last data bit of a data frame is transmitted with the last rising clock edge of SCLKOUT and the last

data bit is received in with the last falling edge of SCLKOUT. This setting can be used in master and in slave mode. It corresponds to the behavior of the internal data shift unit.

- No delay, polarity inversion (SCLKCFG = 01_B):
  The inactive level of SCLKOUT is 1, while no data frame is transferred. The first data bit of a new data frame is transmitted with the first falling clock edge of SCLKOUT and the first data bit is received with the first rising edge of SCLKOUT. The last data bit of a data frame is transmitted with the last falling edge of SCLKOUT and the last data bit is received with the last rising edge of SCLKOUT. This setting can be used in master and in slave mode.
- SCLKOUT is delayed by 1/2 shift clock period, no polarity inversion (SCLKCFG = 10_B):
  The inactive level of SCLKOUT is 0, while no data frame is transferred.
  The first data bit of a new data frame is transmitted 1/2 shift clock period before the first rising clock edge of SCLKOUT. Due to the delay, the next data bits seem to be transmitted with the falling edges of SCLKOUT. The last data bit of a data frame is transmitted 1/2 period of SCLKOUT before the last rising clock edge of SCLKOUT. The first data bit is received 1/2 shift clock period before the first falling edge of SCLKOUT. Due to the delay, the next data bits seem to be received with the rising edges of SCLKOUT. The last data bit is received 1/2 period of SCLKOUT before the last falling clock edge of SCLKOUT.
  This setting can be used only in master mode and not in slave mode (the connected slave has to provide the first data bit before the first SCLKOUT edge, e.g. as soon as it is addressed by its slave select).
- SCLKOUT is delayed by 1/2 shift clock period, polarity inversion (SCLKCFG = 11_B):
  The inactive level of SCLKOUT is 1, while no data frame is transferred.
  The first data bit of a new data frame is transmitted 1/2 shift clock period before the first falling clock edge of SCLKOUT. Due to the delay, the next data bits seem to be transmitted with the rising edges of SCLKOUT. The last data bit of a data frame is transmitted 1/2 period of SCLKOUT before the last falling clock edge of SCLKOUT. The first data bit is received 1/2 shift clock period before the first rising edge of SCLKOUT. Due to the delay, the next data bits seem to be received with the falling edges of SCLKOUT. The last data bit is received 1/2 period of SCLKOUT before the last rising clock edge of SCLKOUT.
  This setting can be used only in master mode and not in slave mode (the connected slave has to provide the first data bit before the first SCLKOUT edge, e.g. as soon as it is addressed by its slave select).

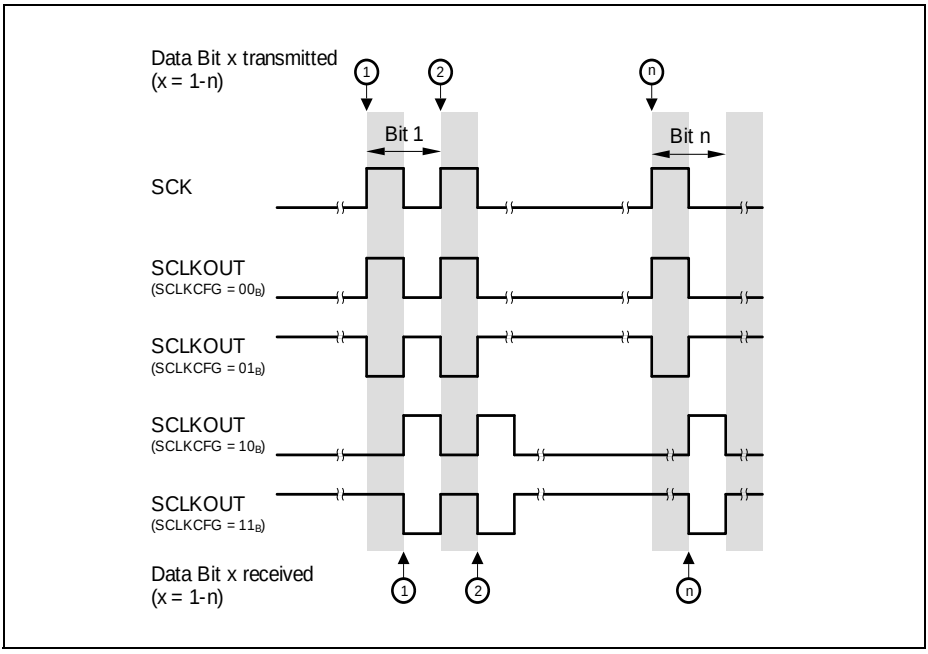


Figure 14-39 SCLKOUT Configuration in SSC Master Mode

Note: If a configuration with delay is selected and a slave select line is used, the slave select delays have to be set up accordingly.

In SSC slave mode, the bit PCR.SLPHSEL can be used to configure the clock phase of the data shift.

- When SLPHSEL = 0_B, the slave SSC transmits data bits with each leading edge of the selected shift clock input (SCLKIN) and receives data bits with each trailing edge of SCLKIN
- When SLPHSEL = 1_B, the slave SSC transmits the first data bit once the selected slave select input (SELIN) becomes active. If SELIN is not used, the DX2 stage has to deliver a 1-level to the data shift unit to shift out the first bit. Subsequent data bits are then transmitted with each trailing edge of SCLKIN. The SSC slave receives all data bits with each leading edge of SCLKIN.

For both settings, the clock polarity is determined by bit 0 of SCLKCFG.

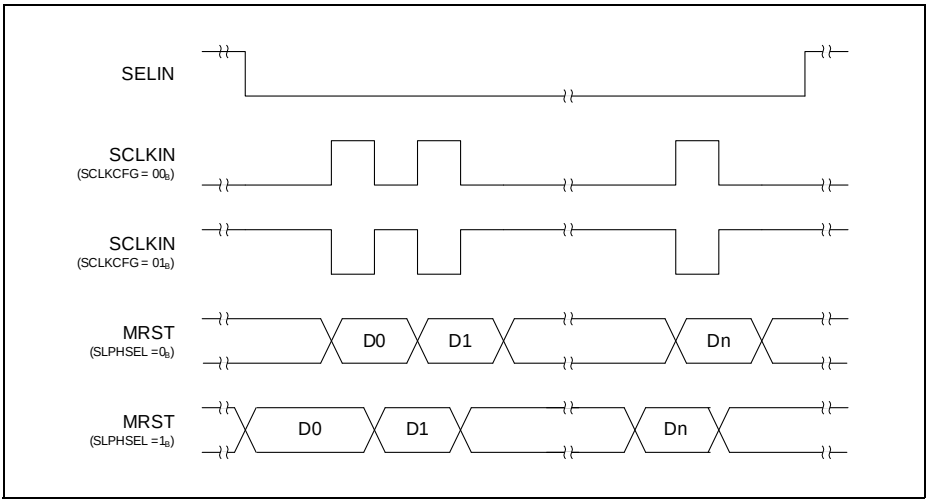


Figure 14-40 SLPHSEL Configuration in SSC Slave Mode

14.4.1.3 Slave Select Signals

The slave select signal is handled by the input stage DX2. In slave mode, the input signal SELIN is received from an external master via an input pin. The input stage can invert the received input signal to adapt the polarity of signal SELIN to the function of the data shift unit (the module internal signals are considered as high active, so a data transfer is only possible while the slave select input of the data shift unit is at 1-level, otherwise, shift clock pulses are ignored and do not lead to data transfers). If an input signal SELIN is low active, it should be inverted in the DX2 input stage.

In master mode, a master slave select signal MSLS is generated by the internal slave select generator. In order to address different external slave devices independently, the internal MSLS signal is made available externally via up to 8 SELOx output signals that can be configured by the block SELCFG (select configuration).

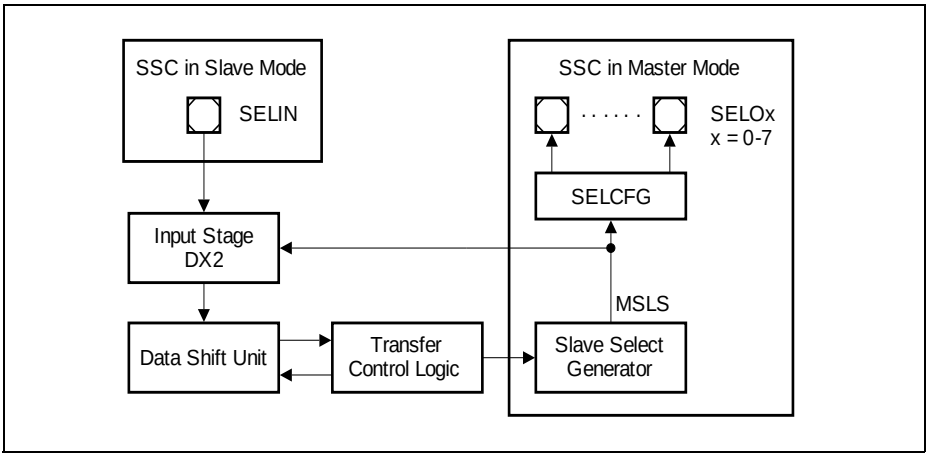


Figure 14-41 SSC Slave Select Signals

The control of the SELCFG block is based on protocol specific bits and bit fields in the protocol control register PCR. For the generation of the MSLS signal please refer to **Section 14.4.3.2**.

- PCR.SELCTR to chose between direct and coded select mode
- PCR.SELINV to invert the SELOx outputs
- PCR.SELO[7:0] as individual value for each SELOx line

The SELCFG block supports the following configurations of the SELOx output signals:

- Direct Select Mode (SELCTR = 1):
  Each SELOx line (with x = 0-7) can be directly connected to an external slave device. If bit x in bit field SELO is 0, the SELOx output is permanently inactive. A SELOx output becomes active while the internal signal MSLS is active (see **Section 14.4.3.2**) and bit x in bit field SELO is 1. Several external slave devices can be addressed in parallel if more than one bit in bit field SELO are set during a data frame. The number of external slave devices that can be addressed individually is limited to the number of available SELOx outputs.
- Coded Select Mode (SELCTR = 0):
  The SELOx lines (with x = 1-7) can be used as addresses for an external address decoder to increase the number of external slave devices. These lines only change with the start of a new data frame and have no other relation to MSLS. Signal SELO0 can be used as enable signal for the external address decoder. It is active while MSLS is active (during a data frame) and bit 0 in bit field SELO is 1. Furthermore, in coded select mode, this output line is delayed by one cycle of f_{PB} compared to MSLS to allow the other SELOx lines to stabilize before enabling the address decoder.

14.4.2 Operating the SSC

This chapter contains SSC issues, that are of general interest and not directly linked to either master mode or slave mode.

14.4.2.1 Automatic Shadow Mechanism

The contents of the baud rate control register BRG, bit fields SCTR.FLE as well as the protocol control register PCR are internally kept constant while a data frame is transferred (= while MSLS is active) by an automatic shadow mechanism. The registers can be programmed all the time with new settings that are taken into account for the next data frame. During a data frame, the applied (shadowed) setting is not changed, although new values have been written after the start of the data frame.

Bit fields SCTR.WLE, SCTR.DSM, SCTR.HPCDIR and SCTR.SDIR are shadowed automatically with the start of each data word. As a result, a data frame can consist of data words with a different length, or data words that are transmitted or received through different number of data lines. It is recommended to change SCTR.SDIR only when no data frame is running to avoid interference between hardware and software.

Please note that the starting point of a data word are different for a transmitter (first bit transmitted) and a receiver (first bit received). In order to ensure correct handling, it is recommended to refer to the receive start interrupt RSI before modifying SCTR.WLE. If TCSR.WLEMD = 1, it is recommended to update TCSR and TBUFxx after the receiver start interrupt has been generated.

14.4.2.2 Mode Control Behavior

In SSC mode, the following kernel modes are supported:

- Run Mode 0/1:
  Behavior as programmed, no impact on data transfers.
- Stop Mode 0/1:
  The content of the transmit buffer is considered as not valid for transmission. Although being considered as 0, bit TCSR.TDV it is not modified by the stop mode condition.
  In master mode, a currently running word transfer is finished normally, but no new data word is started (the stop condition is not considered as end-of-frame condition).
  In slave mode, a currently running word transfer is finished normally. Passive data will be sent out instead of a valid data word if a data word transfer is started by the external master while the slave device is in stop mode. In order to avoid passive slave transmit data, it is recommended not to program stop mode for an SSC slave device if the master device does not respect the slave device's stop mode.

14.4.2.3 Disabling SSC Mode

In order to disable SSC mode without any data corruption, the receiver and the transmitter have to be both idle. This is ensured by requesting Stop Mode 1 in register KSCFG. After Stop Mode 1 has been acknowledged by KSCFG.2 = 1, the SSC mode can be disabled.

14.4.2.4 Data Frame Control

An SSC data frame can consist of several consecutive data words that may be separated by an inter-word delay. Without inter-word delay, the data words seem to form a longer data word, being equivalent to a data frame. The length of the data words are most commonly identical within a data frame, but may also differ from one word to another. The data word length information (defined by SCTR.WLE) is evaluated for each new data word, whereas the frame length information (defined by SCTR.FLE) is evaluated at the beginning at each start of a new frame.
The length of an SSC data frame can be defined in two different ways:

- By the number of bits per frame:
  If the number of bits per data frame is defined (frame length FLE), a slave select signal is not necessarily required to indicate the start and the end of a data frame.
  If the programmed number of bits per frame is reached within a data word, the frame is considered as finished and remaining data bits in the last data word are ignored and are not transferred.
  This method can be applied for data frames with up to 63 data bits.
- By the slave select signal:
  If the number of bits per data frame is not known, the start/end information of a data frame is given by a slave select signal. If a deactivation of the slave select signal is detected within a data word, the frame is considered as finished and remaining data bits in the last data word are ignored and are not transferred.
  This method has to be applied for frames with more than 63 data bits (programming limit of FLE). The advantage of slave select signals is the clearly defined start and end condition of data frames in a data stream. Furthermore, slave select signals allow to address slave devices individually.

14.4.2.5 Parity Mode

The SSC allows parity generation for transmission and parity check for reception on frame base. The type of parity can be selected by bit field CCR.PM, common for transmission and reception (no parity, even or odd parity). If the parity handling is disabled, the SSC frame does not contain any parity bit. For consistency reasons, all communication partners have to be programmed to the same parity mode.

If parity generation has been enabled, the transmitter automatically extends the clock by one cycle after the last data word of the data frame, and sends out its calculated parity bit in this cycle.

Figure 14-42 shows how a parity bit is added to the transmitted data bits of a frame. The number of the transmitted bits of a complete frame with parity is always one more than that without parity. The parity bit is transmitted as the last bit of a frame, following the data bits, independent of the shift direction (SCTR.SDIR).

Note: For dual and quad SSC protocols, the parity bit will be transmitted and received only on DOUT0 and DX0 respectively in the extended clock cycle.

Figure 14-42 Data Frames without/with Parity

Similarly, after the receiver receives the last word of a data frame as defined by FLE, it expects an additional one clock cycle, which will contain the parity bit. The receiver interprets this bit as received parity and separates it from the received data. The received parity bit value is instead monitored in the protocol-related argument (PAR) of the receiver buffer status registers as receiver buffer status information. The receiver compares the bit to its internally calculated parity and the result of the parity check is indicated by the flag PSR.PARERR. The parity error event generates a protocol interrupt if PCR.PARIEN = 1.

Parity bit generation and detection is not supported for the following cases:

- When frame length is 64 data bits or greater, i.e. FLE = 63_H;
- When in slave mode, the end of frame occurs before the number of data bits defined by FLE is reached.

14.4.2.6 Transfer Mode

In SSC mode, bit field SCTR.TRM = 01_B has to be programmed to allow data transfers. Setting SCTR.TRM = 00_B disables and stops the data transfer immediately.

14.4.2.7 Data Transfer Interrupt Handling

The data transfer interrupts indicate events related to SSC frame handling.

- Transmit buffer interrupt TBI:
  Bit PSR.TBIF is set after the start of first data bit of a data word.
- Transmit shift interrupt TSI:
  Bit PSR.TSIF is set after the start of the last data bit of a data word.
- Receiver start interrupt RSI:
  Bit PSR.RSIF is set after the reception of the first data bit of a data word.
  With this event, bit TCSR.TDV is cleared and new data can be loaded to the transmit buffer.
- Receiver interrupt RI:
  The reception of the second, third, and all subsequent words in a multi-word frame is always indicated by RBUFSR.SOF = 0. Bit PSR.RIF is set after the reception of the last data bit of a data word if RBUFSR.SOF = 0.
  Bit RBUFSR.SOF indicates whether the received data word has been the first data word of a multi-word frame or some subsequent word. In SSC mode, it decides if alternative interrupt or receive interrupt is generated.
- Alternative interrupt AI:
  The reception of the first word in a frame is always indicated by RBUFSR.SOF = 1. This is true both in case of reception of multi-word frames and single-word frames. In SSC mode, this results in setting PSR.AIF.

14.4.2.8 Baud Rate Generator Interrupt Handling

The baud rate generator interrupt indicate that the capture mode timer has reached its maximum value. With this event, the bit PSR.BRGIF is set.

14.4.2.9 Protocol-Related Argument and Error

The protocol-related argument (RBUFSR.PAR) and the protocol-related error (RBUFSR.PERR) are two flags that are assigned to each received data word in the corresponding receiver buffer status registers.

In SSC mode, the received parity bit is monitored by the protocol-related argument. The received start of frame indication is monitored by the protocol-related error indication (0 = received word is not the first word of a frame, 1 = received word is the first word of a new frame).

Note: For SSC, the parity error event indication bit is located in the PSR register.

14.4.2.10 Receive Buffer Handling

If a receive FIFO buffer is available (CCFG.RB = 1) and enabled for data handling (RBCTR.SIZE > 0), it is recommended to set RBCTR.RCIM = 01_B in SSC mode. This leads to an indication that the data word has been the first data word of a new data frame if bit OUTR.RCI[4] = 1, and the word length of the received data is given by OUTR.RCI[3:0].

The standard receive buffer event and the alternative receive buffer event can be used for the following operation in RCI mode (RBCTR.RNM = 1):

- A standard receive buffer event indicates that a data word can be read from OUTR that has not been the first word of a data frame.
- An alternative receive buffer event indicates that the first data word of a new data frame can be read from OUTR.

14.4.2.11 Multi-IO SSC Protocols

The SSC implements the following three features to support multiple data input/output SSC protocols, such as the dual- and quad-SSC:

1. Data Shift Mode (**Section 14.2.5.2**)
   Configures the data for transmission and reception using one, two or four data lines in parallel, through the bit field SCTR.DSM.
2. Hardware Port Control (**Section 14.2.7**)
   Sets up a dedicated hardware interface to control the direction of the pins overlaid with both DINx and DOUTx functions, through the bit SCTR.HPCDIR.
3. Transmit Control Information (**Section 14.2.5.3**)
   Allows the dynamic control of both the shift mode and pin direction during data transfers by writing to SCTR.DSM and SCTR.HPCDIR with TCI.

Figure 14-43 shows an example of a Quad-SSC protocol, which requires the master SSC to first transmit a command byte (to request a quad output read from the slave) and a dummy byte through a single data line. At the end of the dummy byte, both master and slave SSC switches to quad data lines, and with the roles of transmitter and receiver reversed. The master SSC then receives the data four bits per shift clock from the slave through the MRST[3:0] lines.

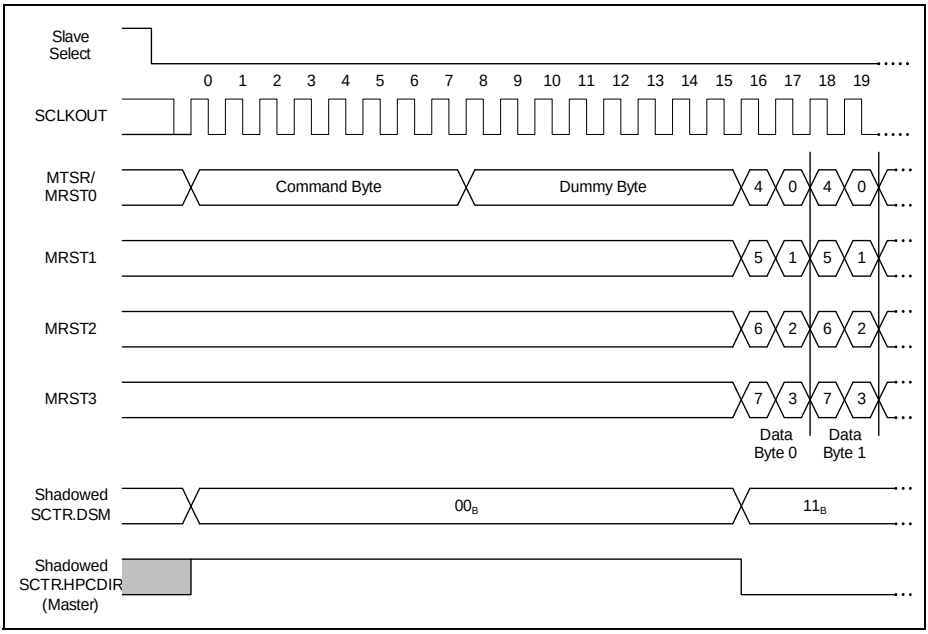


Figure 14-43 Quad-SSC Example

To work with the quad-SSC protocol in the given example, the following issues have to be additionally considered on top of those defined in **Section 14.4.3** and **Section 14.4.4**:

- During the initialization phase:
  - Set CCR.HPCEN to 11_B to enable the dedicated hardware interface to the DX0/DOUT0, DX3/DOUT1, DX4/DOUT2 and DX5/DOUT3 pins.
  - Set TCSR.[4:0] to 10_H to enable hardware port control in TCI
- To start the data transfer:
  - For the master SSC, write the command and dummy bytes into TBUF04 to select a single data line in output mode and initiate the data transfer.
  - For the slave SSC, dummy data can be preloaded into TBUF00 to select a single data line in input mode.
- To switch to quad data lines and pin direction:

– For the master SSC, write subsequent dummy data to TBUF03 to select quad data lines in input mode to read in valid slave data.
– For the slave SSC, write valid data to TBUF07 for transmission through quad data lines in output mode.

Figure 14-44 shows the connections for the Quad-SSC example.

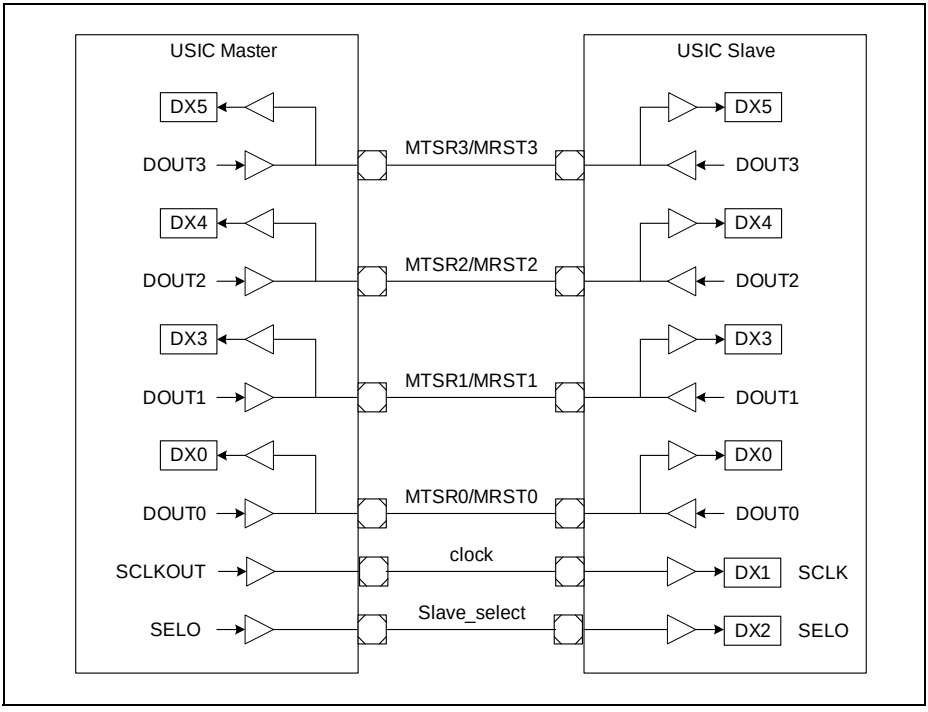


Figure 14-44 Connections for Quad-SSC Example

14.4.3 Operating the SSC in Master Mode

In order to operate the SSC in master mode, the following issues have to be considered:

- Select SSC mode:
  It is recommended to configure all parameters of the SSC that do not change during run time while CCR.MODE = 0000_B. Bit field SCTR.TRM = 01_B has to be programmed. The configuration of the input stages has to be done while CCR.MODE = 0000_B to avoid unintended edges of the input signals and the SSC mode can be enabled by CCR.MODE = 0001_B afterwards.
- Pin connections:
  Establish a connection of the input stage (DX0, DX3, DX4, DX5) with the selected

receive data input pin (DIN[3:0]) with DXnCR.INSW = 1 and configure the transmit data output pin (DOUT[3:0]). One, two or four such connections may be needed depending on the protocol. For half-duplex configurations, hardware port control can be also used to establish the required connections.

- Baud rate generation:
  The desired baud rate setting has to be selected, comprising the fractional divider and the baud rate generator. Bit DX1CR.INSW = 0 has to be programmed to use the baud rate generator output SCLK directly as input for the data shift unit. Configure a shift clock output pin (signal SCLKOUT).
- Slave select generation:
  The slave select delay generation has to be enabled by setting PCR.MSLSEN = 1 and the programming of the time quanta counter setting. Bit DX2CR.INSW = 0 has to be programmed to use the slave select generator output MSLS as input for the data shift unit. Configure slave select output pins (signals SELOx) if needed.
- Data format configuration:
  The word length, the frame length, the shift direction and shift mode have to be set up according to the application requirements by programming the register SCTR.

Note: The USIC can only receive in master mode if it is transmitting, because the master frame handling refers to bit TDV of the transmitter part.

Note: The step to enable the alternate output port functions should only be done after the SSC mode is enabled, to avoided unintended spikes on the output.

14.4.3.1 Baud Rate Generation

The baud rate (determining the length of one data bit) of the SSC is defined by the frequency of the SCLK signal (one period of f_{SCLK} represents one data bit). The SSC baud rate generation does not imply any time quanta counter.

In a standard SSC application, the phase relation between the optional MCLK output signal and SCLK is not relevant and can be disabled (BRG.PPPEN = 0). In this case, the SCLK signal directly derives from the protocol input frequency f_{PIN}. In the exceptional case that a fixed phase relation between the MCLK signal and SCLK is required (e.g. when using MCLK as clock reference for external devices), the additional divider by 2 stage has to be taken into account (BRG.PPPEN = 1).

The adjustable divider factor is defined by bit field BRG.PDIV.

$$f_{SCLK} = \frac{f_{PIN}}{2} \times \frac{1}{PDIV + 1} \quad \text{if PPPEN = 0}$$
$$f_{SCLK} = \frac{f_{PIN}}{2 \times 2} \times \frac{1}{PDIV + 1} \quad \text{if PPPEN = 1} \qquad (14.8)$$

14.4.3.2 MSLS Generation

The slave select signals indicate the start and the end of a data frame and are also used by the communication master to individually select the desired slave device. A slave select output of the communication master becomes active a programmable time before a data part of the frame is started (leading delay T_{ld}), necessary to prepare the slave device for the following communication. After the transfer of a data part of the frame, it becomes inactive again a programmable time after the end of the last bit (trailing delay T_{td}) to respect the slave hold time requirements. If data frames are transferred back-to-back one after the other, the minimum time between the deactivation of the slave select and the next activation of a slave select is programmable (next-frame delay T_{nf}). If a data frame consists of more than one data word, an optional delay between the data words can also be programmed (inter-word delay T_{iw}).

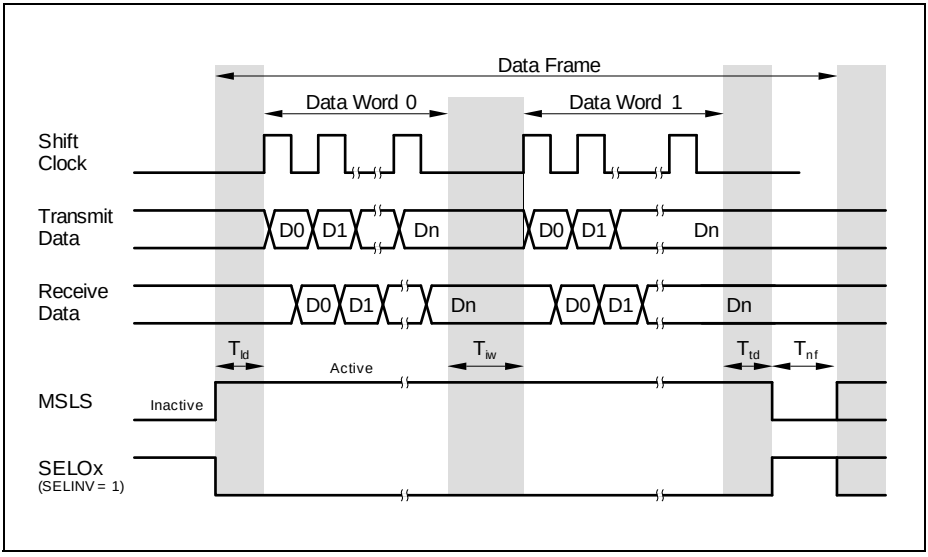


Figure 14-45 MSLS Generation in SSC Master Mode

In SSC master mode, the slave select delays are defined as follows:

- Leading delay T_{ld}:
  The leading delay starts if valid data is available for transmission. The internal signal MSLS becomes active with the start of the leading delay. The first shift clock edge (rising edge) of SCLK is generated by the baud rate generator after the leading delay has elapsed.
- Trailing delay T_{td}
  The trailing delay starts at the end of the last SCLK cycle of a data frame. The internal signal MSLS becomes inactive with the end of the trailing delay.

- Inter-word delay T_{iw}:
  This delay is optional and can be enabled/disabled by PCR.TIWEN. If the inter-word delay is disabled (TIWEN = 0), the last data bit of a data word is directly followed by the first data bit of the next data word of the same data frame. If enabled (TIWEN = 1), the inter-word delay starts at the end of the last SCLK cycle of a data word. The first SCLK cycle of the following data word of the same data frame is started when the inter-word delay has elapsed. During this time, no shift clock pulses are generated and signal MSLS stays active. The communication partner has time to “digest” the previous data word or to prepare for the next one.
- Next-frame delay T_{nf}:
  The next-frame delay starts at the end of the trailing delay. During this time, no shift clock pulses are generated and signal MSLS stays inactive. A frame is considered as finished after the next-frame delay has elapsed.

14.4.3.3 Automatic Slave Select Update

If the number of bits per SSC frame and the word length are defined by bit fields SCTR.FLE and SCTR.WLE, the transmit control information TCI can be used to update the slave select setting PCR.CTR[23:16] to control the SELOx select outputs. The automatic update mechanism is enabled by TCSR.SELMD = 1 (bits TCSR.WLEMD, FLEMD, and WAMD have to be cleared). In this case, the TCI of the first data word of a frame defines the slave select setting of the complete frame due to the automatic shadow mechanism (see **Page 14-61**).

14.4.3.4 Slave Select Delay Generation

The slave select delay generation is based on time quanta. The length of a time quantum (defined by the period of the f_{CTQIN}) and the number of time quanta per delay can be programmed.

In standard SSC applications, the leading delay T_{ld} and the trailing delay T_{td} are mainly used to ensure stability on the input and output lines as well as to respect setup and hold times of the input stages. These two delays have the same length (in most cases shorter than a bit time) and can be programmed with the same set of bit fields.

- BRG.CTQSEL
  to define the input frequency f_{CTQIN} for the time quanta generation for T_{ld} and T_{td}
- BRG.PCTQ
  to define the length of a time quantum (division of f_{CTQIN} by 1, 2, 3, or 4) for T_{ld} and T_{td}
- BRG.DCTQ
  to define the number of time quanta for the delay generation for T_{ld} and T_{td}

The inter-word delay T_{iw} and the next-frame delay T_{nf} are used to handle received data or to prepare data for the next word or frame. These two delays have the same length (in most cases in the bit time range) and can be programmed with a second, independent set of bit fields.

- PCR.CTQSEL1
  to define the input frequency f_{CTQIN} for the time quanta generation for T_{nf} and T_{iw}
- PCR.PCTQ1
  to define the length of a time quantum (division of f_{CTQIN} by 1, 2, 3, or 4) for T_{nf} and T_{iw}
- PCR.DCTQ1
  to define the number of time quanta for the delay generation for T_{nf} and T_{iw}
- PCR.TIWEN
  to enable/disable the inter-word delay T_{iw}

Each delay depends on the length of a time quantum and the programmed number of time quanta given by the bit fields CTQSEL/CTQSEL1, PCTQ/DCTQ and PCTQ1/DCTQ1 (the coding of CTQSEL1 is similar to CTQSEL, etc.). To provide a high flexibility in programming the delay length, the input frequencies can be selected between several possibilities (e.g. based on bit times or on the faster inputs of the protocol-related divider). The delay times are defined as follows:

$$T_{ld} = T_{td} = \frac{(PCTQ + 1) \times (DCTQ + 1)}{f_{CTQIN}}$$
$$T_{iw} = T_{nf} = \frac{(PCTQ1 + 1) \times (DCTQ1 + 1)}{f_{CTQIN}} \quad (14.9)$$

14.4.3.5 Protocol Interrupt Events

The following protocol-related events generated in SSC mode and can lead to a protocol interrupt. They are related to the start and the end of a data frame. After the start of a data frame a new setting could be programmed for the next data frame and after the end of a data frame the SSC connections to pins can be changed.

Please note that the bits in register PSR are not all automatically cleared by hardware and have to be cleared by software in order to monitor new incoming events.

- MSLS Interrupt:
  This interrupt indicates in master mode (MSLS generation enabled) that a data frame has started (activation of MSLS) and has been finished (deactivation of MSLS). Any change of the internal MSLS signal sets bit PSR.MSLSEV and additionally, a protocol interrupt can be generated if PCR.MSLSIEN = 1. The actual state of the internal MSLS signal can be read out at PSR.MSLS to take appropriate actions when this interrupt has been detected.
- DX2T Interrupt:
  This interrupt monitors edges of the input signal of the DX2 stage (although this signal is not used as slave select input for data transfers).
  A programmable edge detection for the DX2 input signal sets bit PSR.DX2TEV and additionally, a protocol interrupt can be generated if PCR.DX2TIEN = 1. The actual state of the selected input signal can be read out at PSR.DX2S to take appropriate actions when this interrupt has been detected.
- Parity Error Interrupt:
  This interrupt indicates that there is a mismatch in the received parity bit (in RBUFSR.PAR) with the calculated parity bit of the last received word of a data frame.

14.4.3.6 End-of-Frame Control

The information about the frame length is required for the MSLS generator of the master device. In addition to the mechanism based on the number of bits per frame (selected with SCTR.FLE < 63), the following alternative mechanisms for end of frame handling are supported. It is recommended to set SCTRFLE = 63 (if several end of frame mechanisms are activated in parallel, the first end condition being found finishes the frame).

- Software-based start of frame indication TCSR.SOF:
  This mechanism can be used if software handles the TBUF data without data FIFO. If bit SOF is set, a valid content of TBUF is considered as first word of a new frame. Bit SOF has to be set before the content of TBUF is transferred to the transmit shift register, so it is recommended to write it before writing data to TBUF. A current data word transfer is finished completely and the slave select delays T_{td} and T_{nf} are applied before starting a new data frame with T_{ld} and the content of TBUF.
  For software-handling of bit SOF, bit TCSR.WLEMD = 0 has to be programmed. In this case, all TBUF[31:0] address locations show an identical behavior (TCI not taken into account for data handling).
- Software-based end of frame indication TCSR.EOF:
  This mechanism can be used if software handles the TBUF data without data FIFO. If bit EOF is set, a valid content of TBUF is considered as last word of a new frame. Bit EOF has to be set before the content of TBUF is transferred to the transmit shift register, so it is recommended to write it before writing data to TBUF. The data word in TBUF is sent out completely and the slave select delays T_{td} and T_{nf} are applied. A new data frame can start with T_{ld} with the next valid TBUF value.
  For software-handling of bit EOF, bit TCSR.WLEMD = 0 has to be programmed. In this case, all TBUF[31:0] address locations show an identical behavior (TCI not taken into account for data handling).
- Software-based address related end of frame handling:
  This mechanism can be used if software handles the TBUF data without data FIFO. If bit TCSR.WLEMD = 1, the address of the written TBUF[31:0] is used as transmit control information TCI[4:0] to update SCTR.WLE (= TCI[3:0]) and TCSR.EOF (= TCI[4]) for each data word. The written TBUF[31:0] address location defines the word length and the end of a frame (locations TBUF[31:16] lead to a frame end).
  For example, writing transmit data to TBUF[07] results in a data word of 8-bit length without finishing the frame, whereas writing transmit data to TBUF[31] leads to a data word length of 16 bits, followed by T_{td}, the deactivation of MSLS and T_{nf}.
  If TCSR.WLEMD = 1, bits TCSR.EOF and SOF, as well as SCTR.WLE must not be written by software after writing data to a TBUF location. Furthermore, it is recommended to clear bits TCSR.SELMD, FLEMD and WAMD.
- FIFO-based address related end of frame handling:
  This mechanism can be used if a data FIFO is used to store the transmit data. The general behavior is similar to the software-based address related end of frame

handling, except that transmit data is not written to the locations TBUF[31:0], but to the FIFO input locations IN[31:0] instead. In this case, software must not write to any of the TBUF locations.

- TBUF related end of frame handling:
  If bit PCR.FEM = 0, an end of frame is assumed if the transmit buffer TBUF does not contain valid transmit data at the end of a data word transmission (TCSR.TDV = 0 or in Stop Mode). In this case, the software has to take care that TBUF does not run empty during a data frame in Run Mode. If bit PCR.FEM = 1, signal MSLS stays active while the transmit buffer is waiting for new data (TCSR.TDV = 1 again) or until Stop Mode is left.
- Explicit end of frame by software:
  The software can explicitly stop a frame by clearing bit PSR.MSLS by writing a 1 to the related bit position in register PSCR. This write action immediately clears bit PSR.MSLS, whereas the internal MSLS signal becomes inactive after finishing a currently running word transfer and respecting the slave select delays T_{td} and T_{nf}.

14.4.4 Operating the SSC in Slave Mode

In order to operate the SSC in slave mode, the following issues have to be considered:

- Select SSC mode:
  It is recommended to configure all parameters of the SSC that do not change during run time while CCR.MODE = 0000_B. Bit field SCTR.TRM = 01_B has to be programmed. The configuration of the input stages has to be done while CCR.MODE = 0000_B to avoid unintended edges of the input signals and the SSC mode can be enabled afterwards by CCR.MODE = 0001_B.
- Pin connections:
  Establish the connection of the input stage (DX0, DX3, DX4, DX5) with the selected receive data input pin (DIN[3:0]) with DXnCR.INSW = 1 and configure the transmit data output pin (DOUT[3:0]). One, two or four such connections may be needed depending on the protocol. For half-duplex configurations, hardware port control can be also used to establish the required connections.
  Establish a connection of input stage DX1 with the selected shift clock input pin (signal SCLKIN) with DX1CR.INSW = 1.
  Establish a connection of input stage DX2 with the selected slave select input pin (signal SELIN) with DX2CR.INSW = 1. If no slave select input signal is used, the DX2 stage has to deliver a 1-level to the data shift unit to allow data reception and transmission. If a slave device is not selected (DX2 stage delivers a 0 to the data shift unit) and a shift clock pulse are received, the incoming data is not received and the DOUTx signal outputs the passive data level defined by SCTR.PDL.
  Note that the step to enable the alternate output port functions should only be done after the SSC mode is enabled, to avoided unintended spikes on the output.
- Baud rate generation:
  The baud rate generator is not needed and can be switched off by the fractional divider.
- Data format configuration:
  If required, the shift mode can be set up for reception and/or transmission of two or four data bits at one time by programming the register SCTR.
- Slave select generation:
  The slave select delay generation is not needed and can be switched off. The bits and bit fields MSLSEN, SELCTR, SELINV, CTQSEL1, PCTQ1, DCTQ1, MSLSIEN, SELO[7:0], and TIWEN in register PCR are not necessary and can be programmed to 0.

14.4.4.1 Protocol Interrupts

The following protocol-related events generated in SSC mode and can lead to a protocol interrupt. They are related to the start and the end of a data frame. After the start of a data frame a new setting could be programmed for the next data frame and after the end of a data frame the SSC connections to pins can be changed.

Please note that the bits in register PSR are not all automatically cleared by hardware and have to be cleared by software in order to monitor new incoming events.

- MSLS event:
  The MSLS generation being switched off, this event is not available.
- DX2T event:
  The slave select input signal SELIN is handled by the DX2 stage and the edges of the selected signal can generate a protocol interrupt. This interrupt allows to indicate that a data frame has started and/or that a data frame has been completely finished. A programmable edge detection for the DX2 input signal activates DX2T, sets bit PSR.DX2TEV and additionally, a protocol interrupt can be generated if PCR.DX2TIEN = 1. The actual state of the selected input signal can be read out at PSR.DX2S to take appropriate actions when this interrupt has been detected.
- Parity Error Interrupt:
  This interrupt indicates that there is a mismatch in the received parity bit (in RBUFSR.PAR) with the calculated parity bit of the last received word of a data frame.

14.4.4.2 End-of-Frame Control

In slave mode, the following possibilities exist to determine the frame length. The slave device either has to refer to an external slave select signal, or to the number of received data bits.

- Frame length known in advance by the slave device, no slave select:
  In this case bit field SCTR.FLE can be programmed to the known value (if it does not exceed 63 bits). A currently running data word transfer is considered as finished if the programmed frame length is reached.
- Frame length not known by the slave, no slave select:
  In this case, the slave device's software has to decide on data word base if a frame is finished. Bit field SCTR.FLE can be either programmed to the word length SCTR.WLE, or to its maximum value to disable the slave internal frame length evaluation by counting received bits.
- Slave device addressed via slave select signal SELIN:
  If the slave device is addressed by a slave select signal delivered by the communication master, the frame start and end information are given by this signal. In this case, bit field SCTR.FLE should be programmed to its maximum value to disable the slave internal frame length evaluation.

14.4.5 SSC Protocol Registers

In SSC mode, the registers PCR and PSR handle SSC related information.

14.4.5.1 SSC Protocol Control Registers

In SSC mode, the PCR register bits or bit fields are defined as described in this section.

PCR
Protocol Control Register [SSC Mode]

($3C_H$) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MCLK	0					SLPHSEL	TIWEN	SELO							
rw	rw					rw	rw	rw							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DX2TIEN	MSLSIEN	PARIEN	DCTQ1					PCTQ1		CTQSEL1		FEM	SELINV	SELCTR	MSLSEN
rw	rw	rw	rw					rw		rw		rw	rw	rw	rw

Field	Bits	Type	Description
MSLSEN	0	rw	**MSLS Enable** This bit enables/disables the generation of the master slave select signal MSLS. If the SSC is a transfer slave, the SLS information is read from a pin and the internal generation is not needed. If the SSC is a transfer master, it has to provide the MSLS signal. 0_B The MSLS generation is disabled (MSLS = 0). This is the setting for SSC slave mode. 1_B The MSLS generation is enabled. This is the setting for SSC master mode.
SELCTR	1	rw	**Select Control** This bit selects the operating mode for the SELO[7:0] outputs. 0_B The coded select mode is enabled. 1_B The direct select mode is enabled.

Field	Bits	Type	Description
SELINV	2	rw	**Select Inversion** This bit defines if the polarity of the SELO[7:0] outputs in relation to the master slave select signal MSLS. 0_B The SELO outputs have the same polarity as the MSLS signal (active high). 1_B The SELO outputs have the inverted polarity to the MSLS signal (active low).
FEM	3	rw	**Frame End Mode** This bit defines if a transmit buffer content that is not valid for transmission is considered as an end of frame condition for the slave select generation. 0_B The current data frame is considered as finished when the last bit of a data word has been sent out and the transmit buffer TBUF does not contain new data (TDV = 0). 1_B The MSLS signal is kept active also while no new data is available and no other end of frame condition is reached. In this case, the software can accept delays in delivering the data without automatic deactivation of MSLS in multi-word data frames.
CTQSEL1	[5:4]	rw	**Input Frequency Selection** This bit field defines the input frequency f_{CTQIN} for the generation of the slave select delays T_{iw} and T_{nf}. 00_B $f_{CTQIN} = f_{PDIV}$ 01_B $f_{CTQIN} = f_{PPP}$ 10_B $f_{CTQIN} = f_{SCLK}$ 11_B $f_{CTQIN} = f_{MCLK}$
PCTQ1	[7:6]	rw	**Divider Factor PCTQ1 for T_{iw} and T_{nf}** This bit field represents the divider factor PCTQ1 (range = 0 - 3) for the generation of the inter-word delay and the next-frame delay. $T_{iw} = T_{nf} = 1/f_{CTQIN}$ x (PCTQ1 + 1) x (DCTQ1 + 1)
DCTQ1	[12:8]	rw	**Divider Factor DCTQ1 for T_{iw} and T_{nf}** This bit field represents the divider factor DCTQ1 (range = 0 - 31) for the generation of the inter-word delay and the next-frame delay. $T_{iw} = T_{nf} = 1/f_{CTQIN}$ x (PCTQ1 + 1) x (DCTQ1 + 1)

Field	Bits	Type	Description
PARIEN	13	rw	**Parity Error Interrupt Enable** This bit enables/disables the generation of a protocol interrupt with the detection of a parity error. 0_B A protocol interrupt is not generated with the detection of a parity error. 1_B A protocol interrupt is generated with the detection of a parity error.
MSLSIEN	14	rw	**MSLS Interrupt Enable** This bit enables/disables the generation of a protocol interrupt if the state of the MSLS signal changes (indicated by PSR.MSLSEV = 1). 0_B A protocol interrupt is not generated if a change of signal MSLS is detected. 1_B A protocol interrupt is generated if a change of signal MSLS is detected.
DX2TIEN	15	rw	**DX2T Interrupt Enable** This bit enables/disables the generation of a protocol interrupt if the DX2T signal becomes activated (indicated by PSR.DX2TEV = 1). 0_B A protocol interrupt is not generated if DX2T is activated. 1_B A protocol interrupt is generated if DX2T is activated.
SELO	[23:16]	rw	**Select Output** This bit field defines the setting of the SELO[7:0] output lines. 0_B The corresponding SELOx line cannot be activated. 1_B The corresponding SELOx line can be activated (according to the mode selected by SELCTR).
TIWEN	24	rw	**Enable Inter-Word Delay T_{iw}** This bit enables/disables the inter-word delay T_{iw} after the transmission of a data word. 0_B No delay between data words of the same frame. 1_B The inter-word delay T_{iw} is enabled and introduced between data words of the same frame.

Field	Bits	Type	Description
SLPHSEL	25	rw	**Slave Mode Clock Phase Select** This bit selects the clock phase for the data shifting in slave mode. 0_B Data bits are shifted out with the leading edge of the shift clock signal and latched in with the trailing edge. 1_B The first data bit is shifted out when the data shift unit receives a low to high transition from the DX2 stage. Subsequent bits are shifted out with the trailing edge of the shift clock signal. Data bits are always latched in with the leading edge.
MCLK	31	rw	**Master Clock Enable** This bit enables/disables the generation of the master clock output signal MCLK, independent from master or slave mode. 0_B The MCLK generation is disabled and output MCLK = 0. 1_B The MCLK generation is enabled.
0	[30:26]	rw	**Reserved** Returns 0 if read; should be written with 0.

14.4.5.2 SSC Protocol Status Register

In SSC mode, the PSR register bits or bit fields are defined as described in this section. The bits and bit fields in register PSR are not cleared by hardware.

The flags in the PSR register can be cleared by writing a 1 to the corresponding bit position in register PSCR. Writing a 1 to a bit position in PSR sets the corresponding flag, but does not lead to further actions (no interrupt generation). Writing a 0 has no effect. The PSR flags should be cleared by software before enabling a new protocol.

PSR
Protocol Status Register [SSC Mode] (48_H) **Reset Value: $0000\ 0000_H$**

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AIF	RIF	TBIF	TSIF	DLIF	RSIF	0					PARERR	DX2TEV	MSLSEV	DX2S	MSLS
rwh	rwh	rwh	rwh	rwh	rwh	r					rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
MSLS	0	rwh	**MSLS Status** This bit indicates the current status of the MSLS signal. It must be cleared by software to stop a running frame. 0_B The internal signal MSLS is inactive (0). 1_B The internal signal MSLS is active (1).
DX2S	1	rwh	**DX2S Status** This bit indicates the current status of the DX2S signal that can be used as slave select input SELIN. 0_B DX2S is 0. 1_B DX2S is 1.
MSLSEV	2	rwh	**MSLS Event Detected**[1)] This bit indicates that the MSLS signal has changed its state since MSLSEV has been cleared. Together with the MSLS status bit, the activation/deactivation of the MSLS signal can be monitored. 0_B The MSLS signal has not changed its state. 1_B The MSLS signal has changed its state.

Field	Bits	Type	Description
DX2TEV	3	rwh	**DX2T Event Detected**[1)] This bit indicates that the DX2T trigger signal has been activated since DX2TEV has been cleared. 0_B The DX2T signal has not been activated. 1_B The DX2T signal has been activated.
PARERR	4	rwh	**Parity Error Event Detected**[1)] This bit indicates that there is a mismatch in the received parity bit (in RBUFSR.PAR) with the calculated parity bit of the last received word of the data frame. 0_B A parity error event has not been activated. 1_B A parity error event has been activated.
RSIF	10	rwh	**Receiver Start Indication Flag** 0_B A receiver start event has not occurred. 1_B A receiver start event has occurred.
DLIF	11	rwh	**Data Lost Indication Flag** 0_B A data lost event has not occurred. 1_B A data lost event has occurred.
TSIF	12	rwh	**Transmit Shift Indication Flag** 0_B A transmit shift event has not occurred. 1_B A transmit shift event has occurred.
TBIF	13	rwh	**Transmit Buffer Indication Flag** 0_B A transmit buffer event has not occurred. 1_B A transmit buffer event has occurred.
RIF	14	rwh	**Receive Indication Flag** 0_B A receive event has not occurred. 1_B A receive event has occurred.
AIF	15	rwh	**Alternative Receive Indication Flag** 0_B An alternative receive event has not occurred. 1_B An alternative receive event has occurred.
BRGIF	16	rwh	**Baud Rate Generator Indication Flag** 0_B A baud rate generator event has not occurred. 1_B A baud rate generator event has occurred.
0	[9:5], [31:17]	r	**Reserved** Returns 0 if read; not modified in SSC mode.

1) This status bit can generate a protocol interrupt in SSC mode (see **Page 14-21**). The general interrupt status flags are described in the general interrupt chapter.

14.4.6 SSC Timing Considerations

The input and output signals have to respect certain timings in order to ensure correct data reception and transmission. In addition to module internal timings (due to input filters, reaction times on events, etc.), also the timings from the input pin via the input stage (T_{in}) to the module and from the module via the output driver stage to the pin (T_{out}), as well as the signal propagation on the wires (T_{prop}) have to be taken into account.

Please note that there might be additional delays in the DXn input stages, because the digital filter and the synchronization stages lead to systematic delays, that have to be considered if these functions are used.

14.4.6.1 Closed-loop Delay

A system-inherent limiting factor for the baud rate of an SSC connection is the closed-loop delay. In a typical application setup, a communication master device is connected to a slave device in full-duplex mode with independent lines for transmit and receive data. In a general case, all transmitters refer to one shift clock edge for transmission and all receivers refer to the other shift clock edge for reception. The master device's SSC module sends out the transmit data, the shift clock and optionally the slave select signal. Therefore, the baud rate generation (BRG) and slave select generation (SSG) are part of the master device. The frame control is similar for SSC modules in master and slave mode, the main difference is the fact which module generates the shift clock and optionally, the slave select signals.

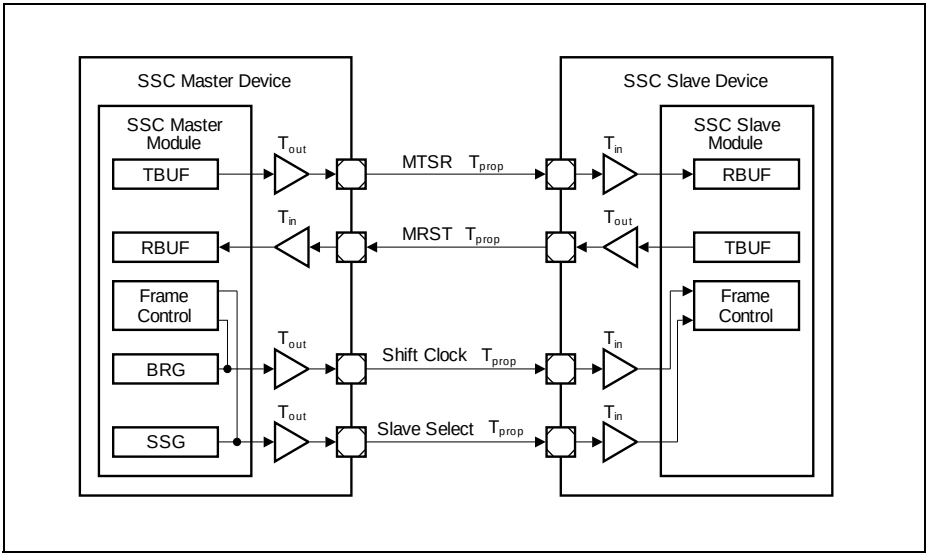


Figure 14-46 SSC Closed-loop Delay

The signal path between the SSC modules of the master and the slave device includes the master's output driver, the wiring to the slave device and the slave device's input stage. With the received shift clock edges, the slave device receives the master's transmit data and transmits its own data back to the master device, passing by a similar signal path in the other direction. The master module receives the slave's transmit data related to its internal shift clock edges. In order to ensure correct data reception in the master device, the slave's transmit data has to be stable (respecting setup and hold times) as master receive data with the next shift clock edge of the master (generally 1/2 shift clock period). To avoid data corruption, the accumulated delays of the input and output stages, the signal propagation on the wiring and the reaction times of the transmitter/receiver have to be carefully considered, especially at high baud rates.
In the given example, the time between the generation of the shift clock signal and the evaluation of the receive data by the master SSC module is given by the sum of T_{out_master} + 2 x T_{prop} + T_{in_slave} + T_{out_slave} + T_{in_master} + module reaction times + input setup times.
The input path is characterized by an input delay depending mainly on the input stage characteristics of the pads. The output path delay is determined by the output driver delay and its slew rate, the external load and current capability of the driver. The device specific values for the input/output driver are given in the Data Sheet.

Figure 14-47 describes graphically the closed-loop delay and the effect of two delay compensation options discussed in **Section 14.4.6.2** and **Section 14.4.6.3**.

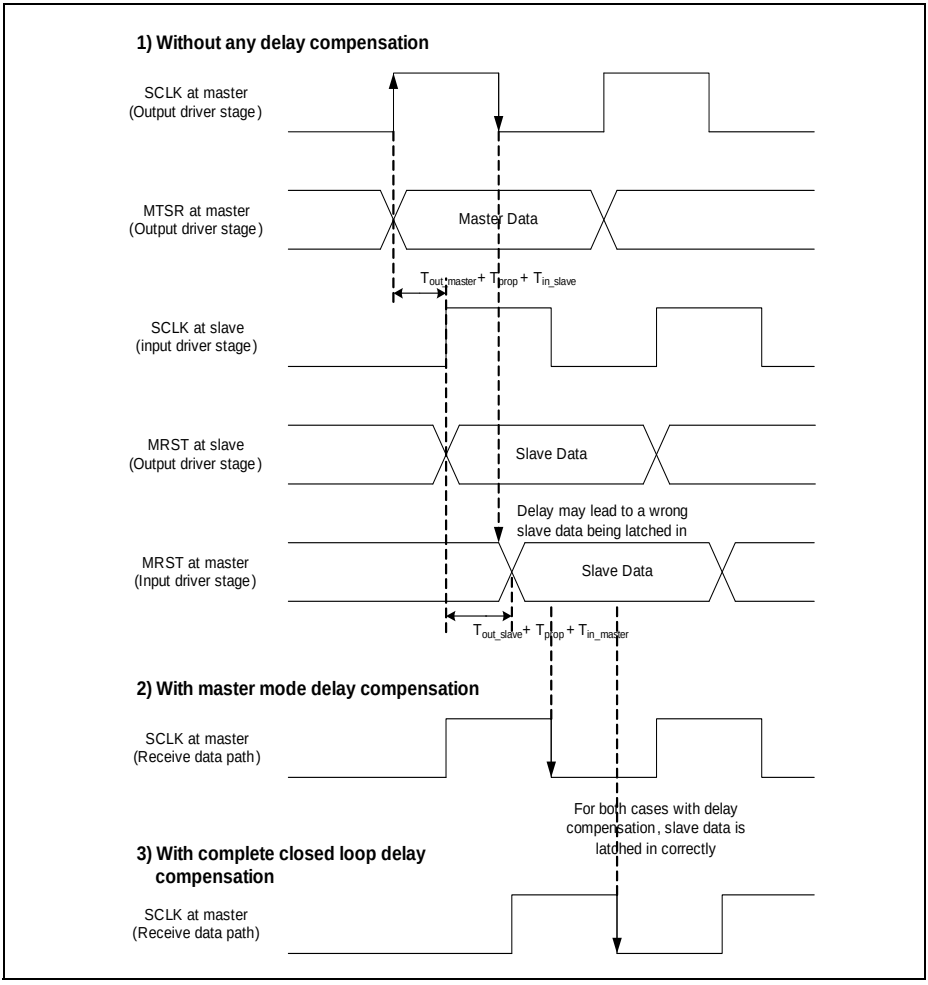


Figure 14-47 SSC Closed-loop Delay Timing Waveform

14.4.6.2 Delay Compensation in Master Mode

A higher baud rate can be reached by delay compensation in master mode. This compensation is possible if (at least) the shift clock pin is bidirectional.

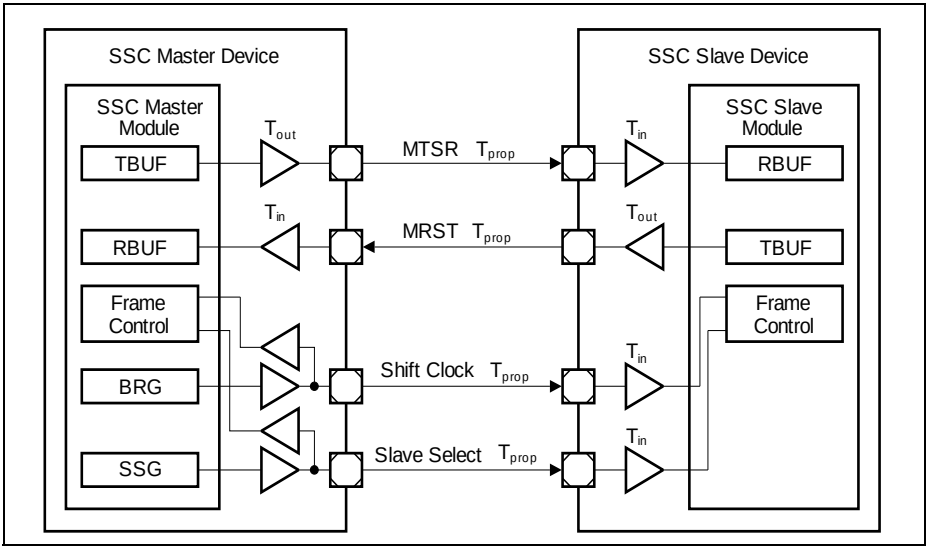


Figure 14-48 SSC Master Mode with Delay Compensation

If the receive shift clock signal in master mode is directly taken from the input function in parallel to the output signal, the output delay of the master device's shift clock output is compensated and only the difference between the input delays of the master and the slave devices have to be taken into account instead of the complete master's output delay and the slave's input delay of the shift clock path. The delay compensation is enabled with DX1CR.DCEN = 1 while DX1CR.INSW = 0 (transmit shift clock is taken from the baud-rate generator).

In the given example, the time between the evaluation of the shift clock signal and the receive data by the master SSC module is reduced by $T_{in\_master} + T_{out\_master}$.

Although being a master mode, the shift clock input and optionally the slave select signal are not directly connected internally to the data shift unit, but are taken as external signals from input pins. The delay compensation does not lead to additional pins for the SSC communication if the shift clock output pin (slave select output pin, respectively) is/are bidirectional. In this case, the input signal is decoupled from other internal signals, because it is related to the signal level at the pin itself.

14.4.6.3 Complete Closed-loop Delay Compensation

Alternatively, the complete closed-loop delay can be compensated by using one additional pin on both the SSC master and slave devices for the SSC communication.

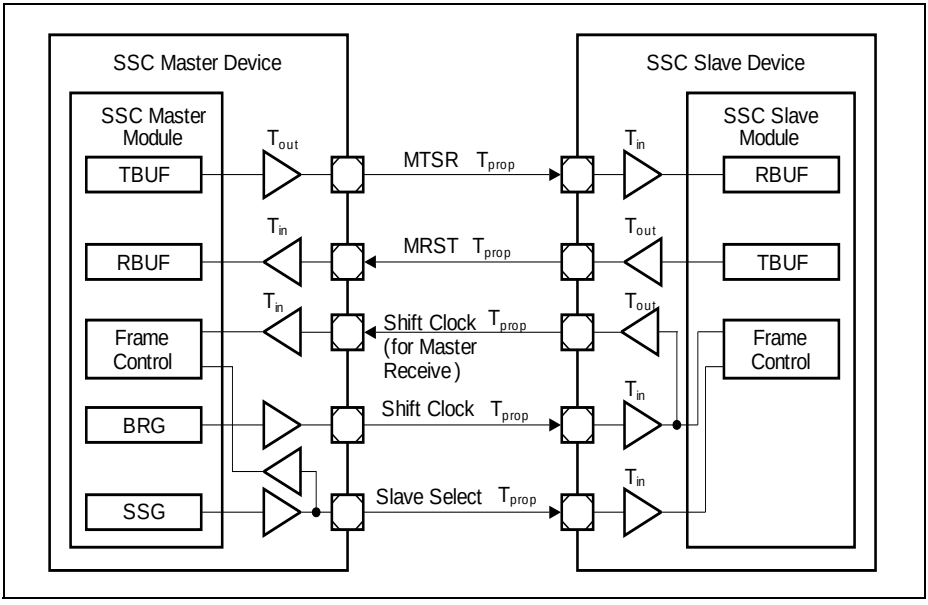


Figure 14-49 SSC Complete Closed-loop Delay Compensation

The principle behind this delay compensation method is to have the slave feedback the shift clock back to the master, which uses it as the receive shift clock. By going through a complete closed-loop signal path, the receive shift clock is thus fully compensated.

The slave has to setup the SCLKOUT pin function to output the shift clock by setting the bit BRG.SCLKOSEL to 1, while the master has to setup the DX1 pin function to receive the shift clock from the slave and enable the delay compensation with DX1CR.DCEN = 1 and DX1CR.INSW = 0.

14.5 Inter-IC Bus Protocol (IIC)

The IIC protocol of the USIC refers to the IIC bus specification **[7]**. Contrary to that specification, the USIC device assumes rise/fall times of the bus signals of max. 300 ns in all modes. Please refer to the pad characteristics in the AC/DC chapter for the driver capability. CBUS mode and HS mode are not supported.
The IIC mode is selected by CCR.MODE = 0100_B with CCFG.IIC = 1 (IIC mode available).

14.5.1 Introduction

USIC IIC Features:

- Two-wire interface, with one line for shift clock transfer and synchronization (shift clock SCL), the other one for the data transfer (shift data SDA)
- Communication in standard mode (100 kBit/s) or in fast mode (up to 400 kBit/s)
- Support of 7-bit addressing, as well as 10-bit addressing
- Master mode operation,
  where the IIC controls the bus transactions and provides the clock signal.
- Slave mode operation,
  where an external master controls the bus transactions and provides the clock signal.
- Multi-master mode operation,
  where several masters can be connected to the bus and bus arbitration can take place, i.e. the IIC module can be master or slave. The master/slave operation of an IIC bus participant can change from frame to frame.
- Efficient frame handling (low software effort)
- Powerful interrupt handling due to multitude of indication flags
- Compensation support for input delays

14.5.1.1 Signal Description

An IIC connection is characterized by two wires (SDA and SCL). The output drivers for these signals must have open-drain characteristics to allow the wired-AND connection of all SDA lines together and all SCL lines together to form the IIC bus system. Due to this structure, a high level driven by an output stage does not necessarily lead immediately to a high level at the corresponding input. Therefore, each SDA or SCL connection has to be input and output at the same time, because the input function always monitors the level of the signal, also while sending.

- Shift data SDA: input handled by DX0 stage, output signal DOUT0
- Shift clock SCL: input handled by DX1 stage, output signal SCLKOUT

Figure 14-29 shows a connection of two IIC bus participants (modules IIC A and IIC B) using the USIC. In this example, the pin assignment of module IIC A shows separate pins for the input and output signals for SDA and SCL. This assignment can be used if the application does not provide pins having DOUT0 and a DX0 stage input for the same pin

(similar for SCLKOUT and DX1). The pin assignment of module IIC B shows the connection of DOUT0 and a DX0 input at the same pin, also for SCLKOUT and a DX1 input.

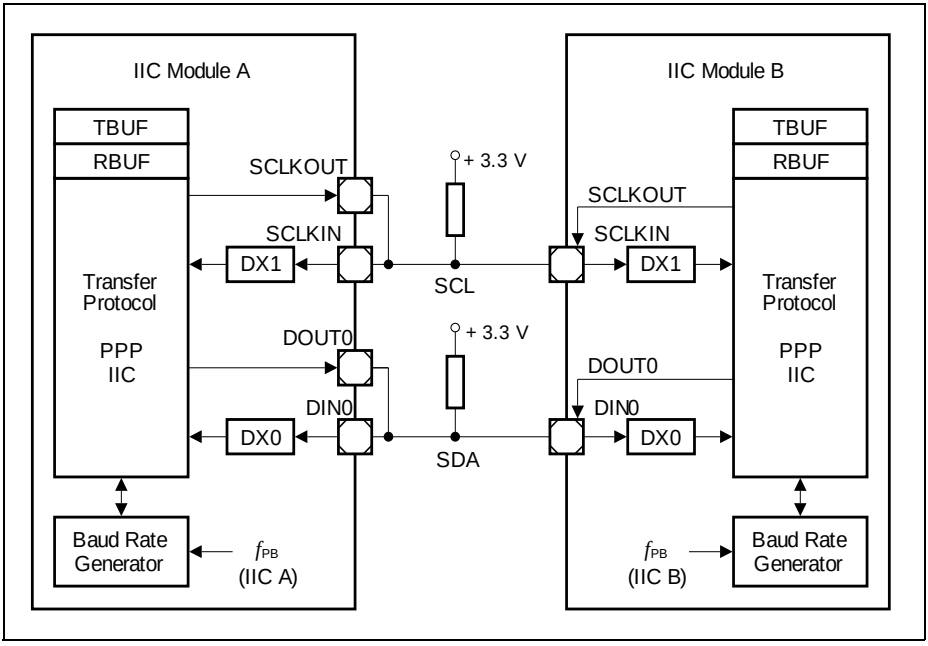


Figure 14-50 IIC Signal Connections

14.5.1.2 Symbols

A symbol is a sequence of edges on the lines SDA and SCL. Symbols contain 10 or 25 time quanta t_q, depending on the selected baud rate. The baud rate generator determines the length of the time quanta t_q, the sequence of edges in a symbol is handled by the IIC protocol pre-processor, and the sequence of symbols can be programmed by the user according to the application needs.

The following symbols are defined:

- Bus idle:
  SDA and SCL are high. No data transfer takes place currently.
- Data bit symbol:
  SDA stable during the high phase of SCL. SDA then represents the transferred bit value. There is one clock pulse on SCL for each transferred bit of data. During data transfers SDA may only change while SCL is low.

- Start symbol:
  Signal SDA being high followed by a falling edge of SDA while SCL is high indicates a start condition. This start condition initiates a data transfer over the IIC bus after the bus has been idle.
- Repeated start symbol:
  This start condition initiates a data transfer over the bus after a data symbol when the bus has not been idle. Therefore, SDA is set high and SCL low, followed by a start symbol.
- Stop symbol:
  A rising edge on SDA while SCL is high indicates a stop condition. This stop condition terminates a data transfer to release the bus to idle state. Between a start condition and a stop condition an arbitrary number of bytes may be transferred.

14.5.1.3 Frame Format

Data is transferred by the 2-line IIC bus (SDA, SCL) using a protocol that ensures reliable and efficient transfers. The sender of a (data) byte receives and checks the value of the following acknowledge field. The IIC being a wired-AND bus system, a 0 of at least one device leads to a 0 on the bus, which is received by all devices.

A data word consists of 8 data bit symbols for the data value, followed by another data bit symbol for the acknowledge bit. The data word can be interpreted as address information (after a start symbol) or as transferred data (after the address).

In order to be able to receive an acknowledge signal, the sender of the data bits has to release the SDA line by sending a 1 as acknowledge value. Depending on the internal state of the receiver, the acknowledge bit is either sent active or passive.

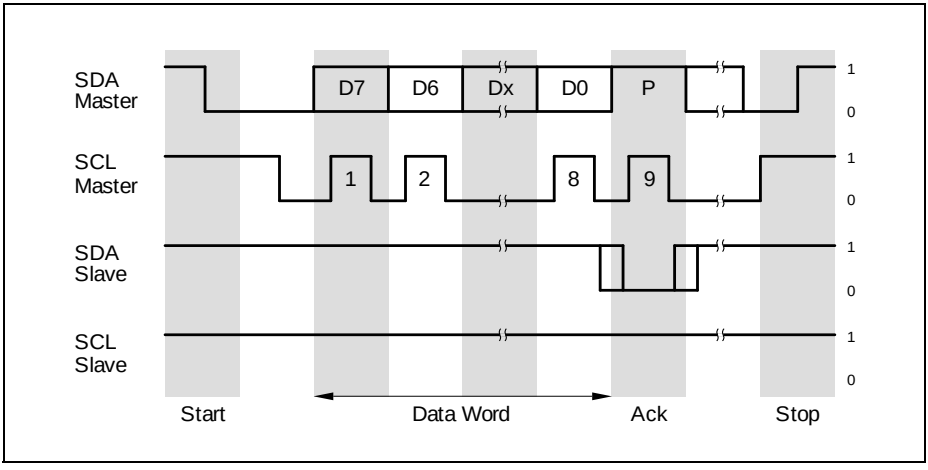


Figure 14-51 IIC Frame Example (simplified)

14.5.2 Operating the IIC

In order to operate the IIC protocol, the following issues have to be considered:

- Select IIC mode:
  It is recommended to configure all parameters of the IIC that do not change during run time while CCR.MODE = 0000_B. Bit field SCTR.TRM = 11_B should be programmed. The configuration of the input stages has to be done while CCR.MODE = 0000_B to avoid unintended edges of the input signals and the IIC mode can be enabled by CCR.MODE = 0100_B afterwards.
- Pin connections:
  Establish a connection of input stage DX0 (with DX0CR.DPOL = 0) to the selected shift data pin SDA (signal DIN0) with DX0CR.INSW = 0 and configure the transmit data output signal DOUT0 (with SCTR.DOCFG = 00_B) to the same pin. If available, this can be the same pin for input and output, or connect the selected input pin and the output pin to form the SDA line.
  The same mechanism applies for the shift clock line SCL. Here, signal SCLKOUT (with BRG.SCLKCFG = 00_B) and an input of the DX1 stage have to be connected (with DX1CR.DPOL = 0).
  The input stage DX2 is not used for the IIC protocol.
  If the digital input filters are enabled in the DX0/1 stages, their delays have to be taken into account for correct calculation of the signal timings.
  The pins used for SDA and SCL have to be set to open-drain mode to support the wired-AND structure of the IIC bus lines.
  Note that the step to enable the alternate output port functions should only be done after the IIC mode is enabled, to avoided unintended spikes on the output.
- Bit timing configuration:
  In standard mode (100 kBit/s) a minimum module frequency of 2 MHz is necessary, whereas in fast mode (400 kBit/s) a minimum of 10 MHz is required. Additionally, if the digital filter stage should be used to eliminate spikes up to 50 ns, a filter frequency of 20 MHz is necessary.
  There could be an uncertainty in the SCL high phase timing of maximum $1/f_{PPP}$ if another IIC participant lengthens the SCL low phase on the bus.
  More details are given in **Section 14.5.3**.
- Data format configuration:
  The data format has to be configured for 8 data bits (SCTR.WLE = 7), unlimited data flow (SCTR.FLE = $3F_H$), and MSB shifted first (SCTR.SDIR = 1). The parity generation has to be disabled (CCR.PM = 00_B).
- General hints:
  The IIC slave module becomes active (for reception or transmission) if it is selected by the address sent by the master. In the case that the slave sends data to the master, it uses the transmit path. So a master must not request to read data from the slave address defined for its own channel in order to avoid collisions.
  The built-in error detection mechanisms are only activated while the IIC module is

taking part in IIC bus traffic.
If the slave can not deal with too high frequencies, it can lengthen the low phase of the SCL signal.
For data transfers according to the IIC specification, the shift data line SDA shall only change while SCL = 0 (defined by IIC bus specification).

14.5.2.1 Transmission Chain

The IIC bus protocol requiring a kind of in-bit-response during the arbitration phase and while a slave is transmitting, the resulting loop delay of the transmission chain can limit the reachable maximal baud rate, strongly depending on the bus characteristics (bus load, module frequency, etc.).

Figure 14-50 shows the general signal path and the delays in the case of a slave transmission. The shift clock SCL is generated by the master device, output on the wire, then it passes through the input stage and the input filter. Now, the edges can be detected and the SDA data signal can be generated accordingly. The SDA signal passes through the output stage and the wire to the master receiver part. There, it passes through the input stage and the input filter before it is sampled.

This complete loop has to be finished (including all settling times to obtain stable signal levels) before the SCL signal changes again. The delays in this path have to be taken into account for the calculation of the baud rate as a function of f_{PB} and f_{PPP}.

14.5.2.2 Byte Stretching

If a device is selected as transceiver and should transmit a data byte but the transmit buffer TBUF does not contain valid data to be transmitted, the device ties down SCL = 0 at the end of the previous acknowledge bit. The waiting period is finished if new valid data has been detected in TBUF.

14.5.2.3 Master Arbitration

During the address and data transmission, the master transmitter checks at the rising edge of SCL for each data bit if the value it is sending is equal to the value read on the SDA line. If yes, the next data bit values can be 0. If this is not the case (transmitted value = 1, value read = 0), the master has lost the transmit arbitration. This is indicated by status flag PSR.ARL and can generate a protocol interrupt if enabled by PCR.ARLIEN.
When the transmit arbitration has been lost, the software has to initialize the complete frame again, starting with the first address byte together with the start condition for a new master transmit attempt. Arbitration also takes place for the ACK bit.

14.5.2.4 Non-Acknowledge and Error Conditions

In case of a non-acknowledge or an error, the TCSR.TDV flag remains set, but no further transmission will take place. User software must invalidate the transmit buffer and disable transmissions (by writing FMRL.MTDV = 10_B), before configuring the transmission (by writing TBUF) again with appropriate values to react on the previous event. In the case the FIFO data buffer is used, additionally the FIFO buffer needs to be flushed and filled again.

14.5.2.5 Mode Control Behavior

In multi-master mode, only run mode 0 and stop mode 0 are supported, the other modes must not be programmed.

- Run Mode 0:
  Behavior as programmed. If TCSR.TDV = 0 (no new valid TBUF entry found) when a new TBUF entry needs to be processed, the IIC module waits for TDV becoming set to continue operation.
- Run Mode 1:
  Behavior as programmed. If in master mode, TCSR.TDV = 0 (no new valid TBUF entry found) when a new TBUF entry needs to be processed, the IIC module sends a stop condition to finish the frame. In slave mode, no difference to run mode 0.
- Stop Mode 0:
  Bit TCSR.TDV is internally considered as 0 (the bit itself is not modified by the stop mode). A currently running word is finished normally, but no new word is started in case of master mode (wait for TDV active).
  Bit TDV being considered as 0 for master and slave, the slave will force a wait state on the bus if read by an external master, too.
  Additionally, it is not possible to force the generation of a STOP condition out of the wait state. The reason is, that a master read transfer must be finished with a not-acknowledged followed by a STOP condition to allow the slave to release his SDA line. Otherwise the slave may force the SDA line to 0 (first data bit of next byte) making it impossible to generate the STOP condition (rising edge on SDA).
  To continue operation, the mode must be switched to run mode 0
- Stop Mode 1:
  Same as stop mode 0, but additionally, a master sends a STOP condition to finish the frame.
  If stop mode 1 is requested for a master device after the first byte of a 10 bit address, a stop condition will be sent out. In this case, a slave device will issue an error interrupt.

14.5.2.6 Data Transfer Interrupt Handling

The data transfer interrupts indicate events related to IIC frame handling. As the data input and output pins are the same in IIC protocol, a IIC transmitter also receives the

output data at its input pin. However, no receive related interrupts will be generated in this case.

- Transmit buffer event:
  The transmit buffer event indication flag PSR.TBIF is set when the content of the transmit buffer TBUF has been loaded to the transmit shift register, indicating that the action requested by the TBUF entry has started.
  With this event, bit TCSR.TDV is cleared. This interrupt can be used to write the next TBUF entry while the last one is in progress (handled by the transmitter part).
- Receive event:
  This receive event indication flag PSR.RIF indicates that a new data byte has been written to the receive buffer RBUF0/1 (except for the first data byte of a new frame, that is indicated by an alternative receive interrupt). The flag becomes set when the data byte is received (after the falling edge of SCL). This interrupt can be used to read out the received data while a new data byte can be in progress (handled by the receiver part).
- Alternate receive event:
  The alternative receive event indication flag AIF is based on bit RBUFSR[9] (same as RBUF[9]), indicating that the received data word has been the first data word of a new data frame.
- Transmit shift event:
  The transmit shift event indication flag TSIF is set after the start of the last data bit of a data byte.
- Receive start event:
  The receive start event indication flag RSIF is set after the sample point of the first data bit of a data byte.

Note: The transmit shift and receive start events can be ignored if the application does not require them during the IIC data transfer.

14.5.2.7 IIC Protocol Interrupt Events

The following protocol-related events are generated in IIC mode and can lead to a protocol interrupt.

Please note that the bits in register PSR are not all automatically cleared by hardware and have to be cleared by software in order to monitor new incoming events.

- start condition received at a correct position in a frame (PSR.SCR)
- repeated start condition received at a correct position in a frame (PSR.RSCR)
- stop condition transferred at a correct position in a frame (PSR.PCR)
- master arbitration lost (PSR.ARL)
- slave read requested (PSR.SRR)
- acknowledge received (PSR.ACK)
- non-acknowledge received (PSR.NACK)
- start condition not at the expected position in a frame (PSR.ERR)

- stop condition not at the expected position in a frame (PSR.ERR)
- as slave, 10-bit address interrupted by a stop condition after the first address byte (PSR.ERR)
- TDF slave code in master mode (PSR.WTDF)
- TDF master code in slave mode (PSR.WTDF)
- Reserved TDF code found (PSR.WDTF)
- Start condition code during a running frame in master mode (PSR.WTDF)
- Data byte transmission code after transfer direction has been changed to reception (master read) in master mode (PSR.WTDF)

If a wrong TDF code is found in TBUF, the error event is active until the TDF value is either corrected or invalidated. If the related interrupt is enabled, the interrupt handler should check PSR.WDTF first and correct or invalidate TBUF, before dealing with the other possible interrupt events.

14.5.2.8 Baud Rate Generator Interrupt Handling

The baud rate generator interrupt indicate that the capture mode timer has reached its maximum value. With this event, the bit PSR.BRGIF is set.

14.5.2.9 Receiver Address Acknowledge

After a (repeated) start condition, the master sends a slave address to identify the target device of the communication. The start address can comprise one or two address bytes (for 7 bit or for 10 bit addressing schemes). After an address byte, a slave sensitive to the transmitted address has to acknowledge the reception.

Therefore, the slave's address can be programmed in the device, where it is compared to the received address. In case of a match, the slave answers with an acknowledge (SDA = 0). Slaves that are not targeted answer with an non-acknowledge (SDA = 1).
In addition to the match of the programmed address, another address byte value has to be answered with an acknowledge if the slave is capable to handle the corresponding requests. The address byte 00_H indicates a general call address, that can be acknowledged. The value 01_H stands for a start byte generation, that is not acknowledged

In order to allow selective acknowledges for the different values of the address byte(s), the following control mechanism is implemented:

- The address byte 00_H is acknowledged if bit PCR.ACK00 is set.
- The address byte 01_H is not acknowledged.
- The first 7 bits of a received first address byte are compared to the programmed slave address (PCR.SLAD[15:9]). If these bits match, the slave sends an acknowledge. In addition to this, if the slave address is programmed to 1111 $0XX_B$, the slave device waits for a second address byte and compares it also to PCR.SLAD[7:0] and sends an acknowledge accordingly to cover the 10 bit addressing mode. The user has to

take care about reserved addresses (refer to IIC specification for more detailed description). Only the address 1111 $0XX_B$ is supported.

Under each of these conditions, bit PSR.SLSEL will be set when the addressing delivered a match. This bit is cleared automatically by a (repeated) start condition.

14.5.2.10 Receiver Handling

A selected slave receiver always acknowledges a received data byte. If the receive buffers RBUF0/1 are already full and can not accept more data, the respective register is overwritten (PSR.DLI becomes set in this case and a protocol interrupt can be generated).

An address reception also uses the registers RBUF0/1 to store the address before checking if the device is selected. The received addresses do not set RDV0/1, so the addresses are not handled like received data.

14.5.2.11 Receiver Status Information

In addition to the received data byte, some IIC protocol related information is stored in the 16-bit data word of the receive buffer. The received data byte is available at the bit positions RBUF[7:0], whereas the additional information is monitored at the bit positions RBUF[12:8]. This structure allows to identify the meaning of each received data byte without reading additional registers, also when using a FIFO data buffer.

- RBUF[8]:
  Value of the received acknowledge bit. This information is also available in RBUFSR[8] as protocol argument.
- RBUF[9]:
  A 1 at this bit position indicates that after a (repeated) start condition followed by the address reception the first data byte of a new frame has been received. A 0 at this bit position indicates further data bytes. This information is also available in RBUFSR[9], allowing different interrupt routines for the address and data handling.
- RBUF[10]:
  A 0 at this bit position indicates that the data byte has been received when the device has been in slave mode, whereas a 1 indicates a reception in master mode.
- RBUF[11]:
  A 1 at this bit position indicates an incomplete/erroneous data byte in the receive buffer caused by a wrong position of a START or STOP condition in the frame. The bit is not identical to the frame error status bit in PSR, because the bit in the PSR has to be cleared by software (“sticky” bit), whereas RBUF[11] is evaluated data byte by data byte. If RBUF[11] = 0, the received data byte has been correct, independent of former errors.
- RBUF[12]:
  A 0 at this bit position indicates that the programmed address has been received. A 1 indicates a general call address.

14.5.3 Symbol Timing

The symbol timing of the IIC is determined by the master stimulating the shift clock line SCL. It is different for standard and fast IIC mode.

- 100 kBaud standard mode (PCR.STIM = 0):
  The symbol timing is based on 10 time quanta t_q per symbol. A minimum module clock frequency f_{PB} = 2 MHz is required.
- 400 kBaud standard mode (PCR.STIM = 1):
  The symbol timing is based on 25 time quanta t_q per symbol. A minimum module clock frequency f_{PB} = 10 MHz is required.

The baud rate setting should only be changed while the transmitter and the receiver are idle or CCR.MODE = 0. The bits in register BRG define the length of a time quantum t_q that is given by one period of f_{PCTQ}.

- BRG.CTQSEL
  to define the input frequency f_{CTQIN} for the time quanta generation
- BRG.PCTQ
  to define the length of a time quantum (division of f_{CTQIN} by 1, 2, 3, or 4)
- BRG.DCTQ
  to define the number of time quanta per symbol (number of t_q = DCTQ + 1)

The standard setting is given by CTQSEL = 00_B ($f_{CTQIN} = f_{PDIV}$) and PPPEN = 0 ($f_{PPP} = f_{IN}$). Under these conditions, the frequency f_{PCTQ} is given by:

$$f_{PCTQ} = f_{PIN} \times \frac{1}{PDIV + 1} \times \frac{1}{PCTQ + 1} \quad (14.10)$$

To respect the specified SDA hold time of 300 ns after a falling edge of signal SCL, a hold delay t_{HDEL} has been introduced. It also prevents an erroneous detection of a start or a stop condition. The length of this delay can be programmed by bit field PCR.HDEL. Taking into account the input sampling and output update, bit field HDEL can be programmed according to:

$$\begin{aligned} HDEL &\geq 300\ ns \times f_{PPP} - \left(3 \times \frac{f_{PPP}}{f_{PB}}\right) + 1 \quad \text{with digital filter and } HDEL_{min} = 2 \\ HDEL &\geq 300\ ns \times f_{PPP} - \left(3 \times \frac{f_{PPP}}{f_{PB}}\right) + 2 \quad \text{without digital filter and } HDEL_{min} = 1 \end{aligned} \quad (14.11)$$

If the digital input filter is used, HDEL compensates the filter delay of 2 filter periods (f_{PPP} should be used) in case of a spike on the input signal. This ensures that a data bit on the SDA line changing just before the rising edge or behind the falling edge of SCL will not be treated as a start or stop condition.

14.5.3.1 Start Symbol

Figure 14-52 shows the general start symbol timing.

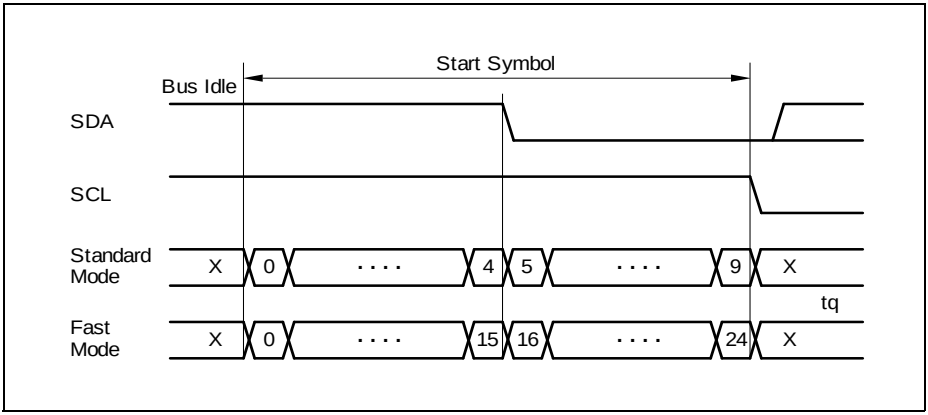


Figure 14-52 Start Symbol Timing

14.5.3.2 Repeated Start Symbol

During the first part of a repeated start symbol, an SCL low value is driven for the specified number of time quanta. Then a high value is output. After the detection of a rising edge at the SCL input, a normal start symbol is generated, as shown in **Figure 14-53**.

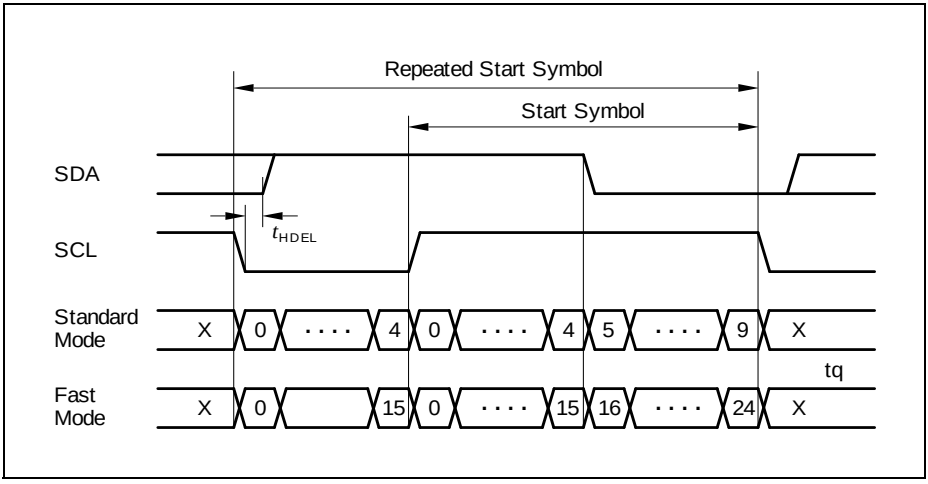


Figure 14-53 Repeated Start Symbol Timing

14.5.3.3 Stop Symbol

Figure 14-54 shows the stop symbol timing.

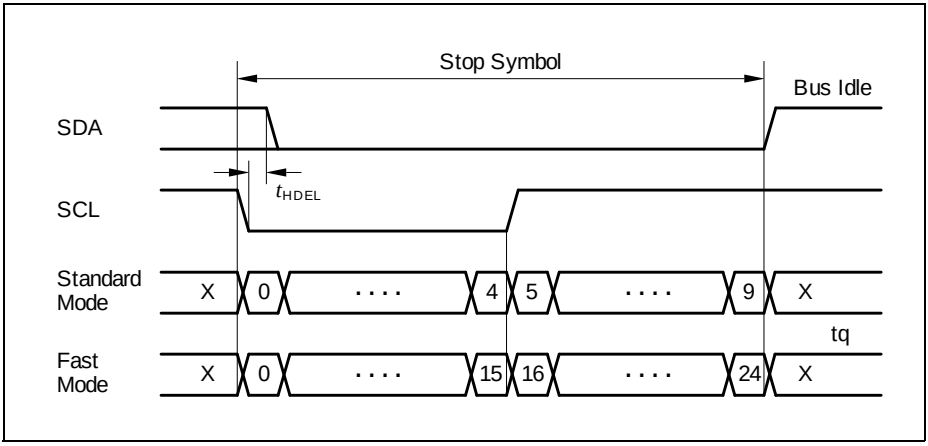


Figure 14-54 Stop Symbol Timing

14.5.3.4 Data Bit Symbol

Figure 14-55 shows the general data bit symbol timing.

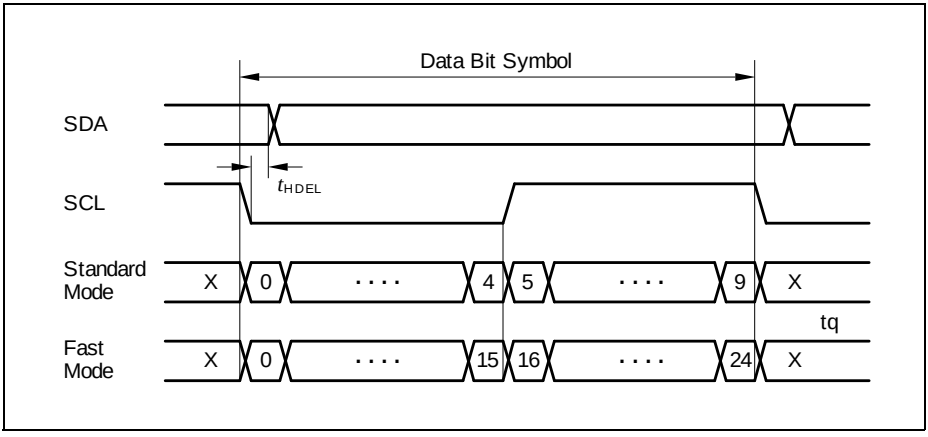


Figure 14-55 Data Bit Symbol

Output SDA changes after the time t_{HDEL} defined by PCR.HDEL has elapsed if a falling edge is detected at the SCL input to respect the SDA hold time. The value of PCR.HDEL allows compensation of the delay of the SCL input path (sampling, filtering).

In the case of an acknowledge transmission, the USIC IIC waits for the receiver indicating that a complete byte has been received. This adds an additional delay of 3 periods of f_{PB} to the path. The minimum module input frequency has to be selected properly to ensure the SDA setup time to SCL rising edge.

14.5.4 Data Flow Handling

The handling of the data flow and the sequence of the symbols in an IIC frame is controlled by the IIC transmitter part of the USIC communication channel. The IIC bus protocol is byte-oriented, whereas a USIC data buffer word can contain up to 16 data bits. In addition to the data byte to be transmitted (located at TBUF[7:0]), bit field TDF (transmit data format) to control the IIC sequence is located at the bit positions TBUF[10:8]. The TDF code defines for each data byte how it should be transmitted (IIC master or IIC slave), and controls the transmission of (repeated) start and stop symbols. This structure allows the definition of a complete IIC frame for an IIC master device only by writing to TBUFx or by using a FIFO data buffer mechanism, because no other control registers have to be accessed. Alternatively, polling of the ACK and NACK bits in PSR register can be performed, and the next data byte is transmitted only after an ACK is received.

If a wrong or unexpected TDF code is encountered (e.g. due to a software error during setup of the transmit buffer), a stop condition will be sent out by the master. This leads to an abort of the currently running frame. A slave module waits for a valid TDF code and sets SCL = 0. The software then has to invalidate the unexpected TDF code and write a valid one.

Please note that during an arbitration phase in multi-master bus systems an unpredictable bus behavior may occur due to an unexpected stop condition.

14.5.4.1 Transmit Data Formats

The following transmit data formats are available in master mode:

Table 14-12 Master Transmit Data Formats

TDF Code	Description
000_B	**Send data byte as master** This format is used to transmit a data byte from the master to a slave. The transmitter sends its data byte (TBUF[7:0]), receives and checks the acknowledge bit sent by the slave.
010_B	**Receive data byte and send acknowledge** This format is used by the master to read a data byte from a slave. The master acknowledges the transfer with a 0-level to continue the transfer. The content of TBUF[7:0] is ignored.

Table 14-12 Master Transmit Data Formats (cont'd)

TDF Code	Description
011_B	**Receive data byte and send not-acknowledge** This format is used by the master to read a data byte from a slave. The master does not acknowledge the transfer with a 1-level to finish the transfer. The content of TBUF[7:0] is ignored.
100_B	**Send start condition** If TBUF contains this entry while the bus is idle, a start condition will be generated. The content of TBUF[7:0] is taken as first address byte for the transmission (bits TBUF[7:1] are the address, the LSB is the read/write control).
101_B	**Send repeated start condition** If TBUF contains this entry and SCL = 0 and a byte transfer is not in progress, a repeated start condition will be sent out if the device is the current master. The current master is defined as the device that has set the start condition (and also won the master arbitration) for the current message. The content of TBUF[7:0] is taken as first address byte for the transmission (bits TBUF[7:1] are the address, the LSB is the read/write control).
110_B	**Send stop condition** If the current master has finished its last byte transfer (including acknowledge), it sends a stop condition if this format is in TBUF. The content of TBUF[7:0] is ignored.
111_B	**Reserved** This code must not be programmed. No additional action except releasing the TBUF entry and setting the error bit in PSR (that can lead to a protocol interrupt).

The following transmit data format is available in slave mode (the symbols in a frame are controlled by the master and the slave only has to send data if it has been "asked" by the master):

Table 14-13 Slave Transmit Data Format

TDF Code	Description
001_B	**Send data byte as slave** This format is used to transmit a data byte from a slave to the master. The transmitter sends its data byte (TBUF[7:0]) plus the acknowledge bit as a 1.

14.5.4.2 Valid Master Transmit Data Formats

Due to the IIC frame format definitions, only some specific sequences of TDF codes are possible and valid. If the USIC IIC module detects a wrong TDF code in a running frame, the transfer is aborted and flag PCR.WTDF is set. Additionally, an interrupt can be generated if enabled by the user. In case of a wrong TDF code, the frame will be aborted immediately with a STOP condition if the USIC IIC master still owns the SDA line. But if the accessed slave owns the SDA line (read transfer), the master must perform a dummy read with a non-acknowledge so that the slave releases the SDA line before a STOP condition can be sent. The received data byte of the dummy read will be stored in RBUF0/1, but RDV0/1 will not be set. Therefore the dummy read will not generate a receive interrupt and the data byte will not be stored into the receive FIFO.

If the transfer direction has changed in the current frame (master read access), the transmit data request (TDF = 000_B) is not possible and won't be accepted (leading to a wrong TDF Code indication).

Table 14-14 Valid TDF Codes Overview

Frame Position	Valid TDF Codes
First TDF code (master idle)	Start (100_B)
Read transfer: second TDF code (after start or repeated start)	Receive with acknowledge (010_B) or receive with not-acknowledge (011_B)
Write transfer: second TDF code (after start or repeated start)	Transmit (000_B), repeated start (101_B), or stop (110_B)
Read transfer: third and subsequent TDF code after acknowledge	Receive with acknowledge (010_B) or receive with not-acknowledge (011_B)
Read transfer: third and subsequent TDF code after not-acknowledge	Repeated start (101_B) or stop (110_B)
Write transfer: third and subsequent TDF code	Transmit (000_B), repeated start (101_B), or stop (110_B)

- First TDF code:
  A master transfer starts with the TDF start code (100_B). All other codes are ignored, but no WTDF error will be indicated.
- TDF code after a start (100_B) or repeated start code (101_B) in case of a read access:
  If a master-read transfer is started (determined by the LSB of the address byte = 1), the transfer direction of SDA changes and the slave will actively drive the data line. In this case, only the codes 010_B and 011_B are valid. To abort the transfer in case of a wrong code, a dummy read must be performed by the master before the STOP condition can be generated.

- TDF code after a start (100_B) or repeated start code (101_B) in case of a write access:
  If a master-write transfer is started (determined by the LSB of the address byte = 0), the master still owns the SDA line. In this case, the transmit (000_B), repeated start (101_B) and stop (110_B) codes are valid. The other codes are considered as wrong. To abort the transfer in case of a wrong code, the STOP condition is generated immediately.
- TDF code of the third and subsequent command in case of a read access with acknowledged previous data byte:
  If a master-read transfer is started (determined by the LSB of the address byte), the transfer direction of SDA changes and the slave will actively drive the data line. To force the slave to release the SDA line, the master has to not-acknowledge a byte transfer. In this case, only the receive codes 010_B and 011_B are valid. To abort the transfer in case of a wrong code, a dummy read must be performed by the master before the STOP condition can be generated.
- TDF code of the third and subsequent command in case of a read access with a not-acknowledged previous data byte:
  If a master-read transfer is started (determined by the LSB of the address byte), the transfer direction of SDA changes and the slave will actively drive the data line. To force the slave to release the SDA line, the master has to not-acknowledge a byte transfer. In this case, only the restart (101_B) and stop code (110_B) are valid. To abort the transfer in case of a wrong code, the STOP condition is generated immediately.
- TDF code of the third and subsequent command in case of a write access:
  If a master-write transfer is started (determined by the LSB of the address byte), the master still owns the SDA line. In this case, the transmit (000_B), repeated start (101_B) and stop (110_B) codes are valid. The other codes are considered as wrong. To abort the transfer in case of a wrong code, the STOP condition is generated immediately.
- After a master device has received a non-acknowledge from a slave device, a stop condition will be sent out automatically, except if the following TDF code requests a repeated start condition. In this case, the TDF code is taken into account, whereas all other TDF codes are ignored.

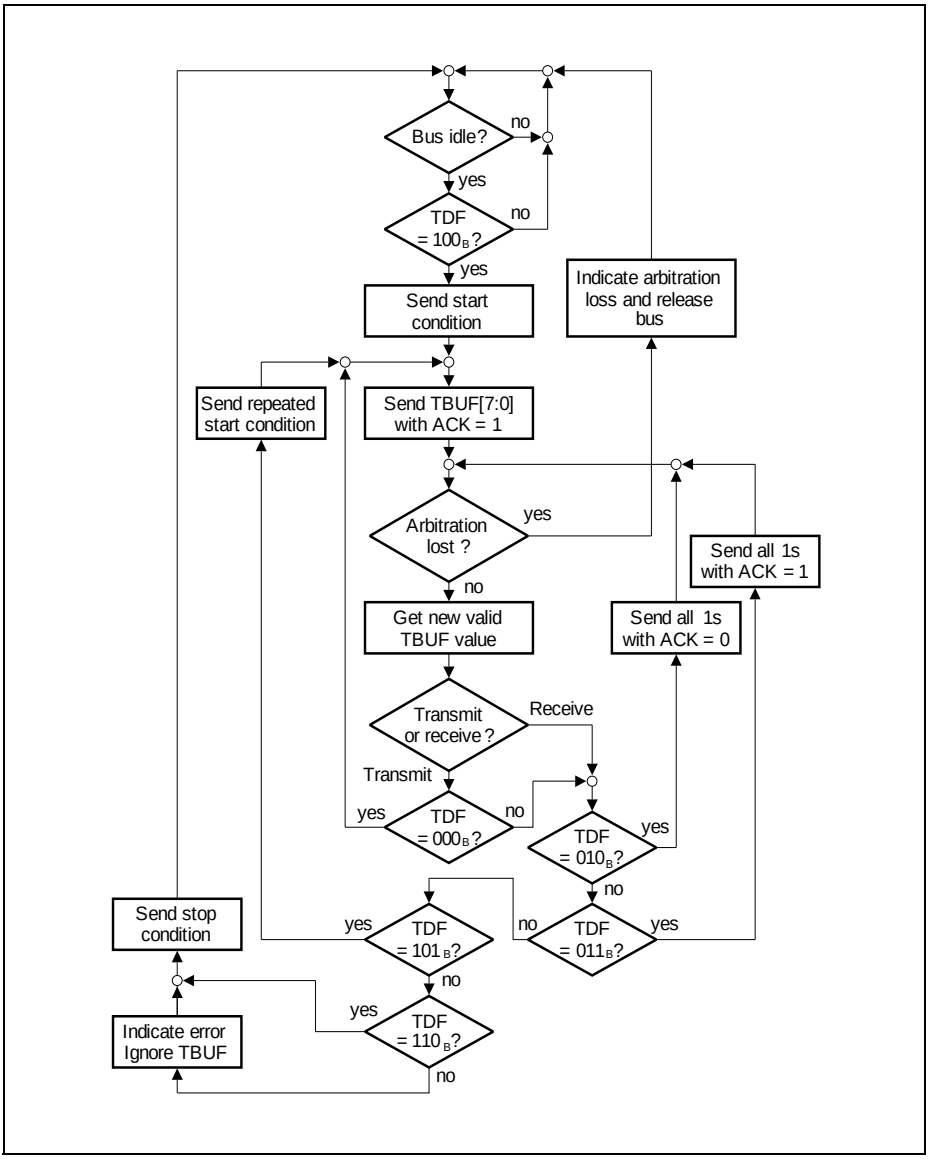


Figure 14-56 IIC Master Transmission

14.5.4.3 Master Transmit/Receive Modes

In master transmit mode, the IIC sends a number of data bytes to a slave receiver. The TDF code sequence for the master transmit mode is shown in **Table 14-15**.

Table 14-15 TDF Code Sequence for Master Transmit

TDF Code Sequence	TBUF[10:8] (TDF Code)	TBUF[7:0]	IIC Response	Interrupt Events
1st code	100_B	Slave address + write bit	Send START condition, slave address and write bit	SCR: Indicates a START condition is detected TBIF: Next word can be written to TBUF
2nd code	000_B	Data or 2nd slave address byte	Send data or 2nd slave address byte	TBIF: Next word can be written to TBUF
Subsequent codes for data transmit	000_B	Data	Send data	TBIF: Next word can be written to TBUF
Last code	110_B	Don't care	Send STOP condition	PCR: Indicates a STOP condition is detected

In master receive mode, the IIC receives a number of data bytes from a slave transmitter. The TDF code sequence for the master receive 7-bit and 10-bit addressing modes are shown in **Table 14-16** and **Table 14-17**.

Table 14-16 TDF Code Sequence for Master Receive (7-bit Addressing Mode)

TDF Code Sequence	TBUF[10:8] (TDF Code)	TBUF[7:0]	IIC Response	Interrupt Events
1st code	100_B	Slave address + read bit	Send START condition, slave address and read bit	SCR: Indicates a START condition is detected TBIF: Next word can be written to TBUF
2nd code	010_B	Don't care	Receive data and send ACK bit	TBIF: Next word can be written to TBUF AIF: First data received can be read

Table 14-16 TDF Code Sequence for Master Receive (7-bit Addressing Mode)

TDF Code Sequence	TBUF[10:8] (TDF Code)	TBUF[7:0]	IIC Response	Interrupt Events
Subsequent codes for data receive	010_B	Don't care	Receive data and send ACK bit	TBIF: Next word can be written to TBUF RIF: Subsequent data received can be read
Code for last data to be received	011_B	Don't care	Receive data and send NACK bit	TBIF: Next word can be written to TBUF RIF: Last data received can be read
Last code	110_B	Don't care	Send STOP condition	PCR: Indicates a STOP condition is detected

Table 14-17 TDF Code Sequence for Master Receive (10-bit Addressing Mode)

TDF Code Sequence	TBUF[10:8] (TDF Code)	TBUF[7:0]	IIC Response	Interrupt Events
1st code	100_B	Slave address (1st byte) + write bit	Send START condition, slave address (1st byte) and write bit	SCR: Indicates a START condition is detected TBIF: Next word can be written to TBUF
2nd code	000_B	Slave address (2nd byte)	Send address (2nd byte)	TBIF: Next word can be written to TBUF
3rd code	101_B	1st slave address + read bit	Send repeated START condition, slave address (1st byte) and read bit	RSCR: Indicates a repeated START condition is detected TBIF: Next word can be written to TBUF
4th code	010_B	Don't care	Receive data and send ACK bit	TBIF: Next word can be written to TBUF AIF: First data received can be read
Subsequent codes for data receive	010_B	Don't care	Receive data and send ACK bit	TBIF: Next word can be written to TBUF RIF: Subsequent data received can be read

Table 14-17 TDF Code Sequence for Master Receive (10-bit Addressing Mode)

TDF Code Sequence	TBUF[10:8] (TDF Code)	TBUF[7:0]	IIC Response	Interrupt Events
Code for last data to be received	011_B	Don't care	Receive data and send NACK bit	TBIF: Next word can be written to TBUF RIF: Last data received from slave can be read
Last code	110_B	Don't care	Send STOP condition	PCR: Indicates a STOP condition is detected

Figure 14-57 shows the interrupt events during the master transmit-slave receive and master receive/slave transmit sequences.

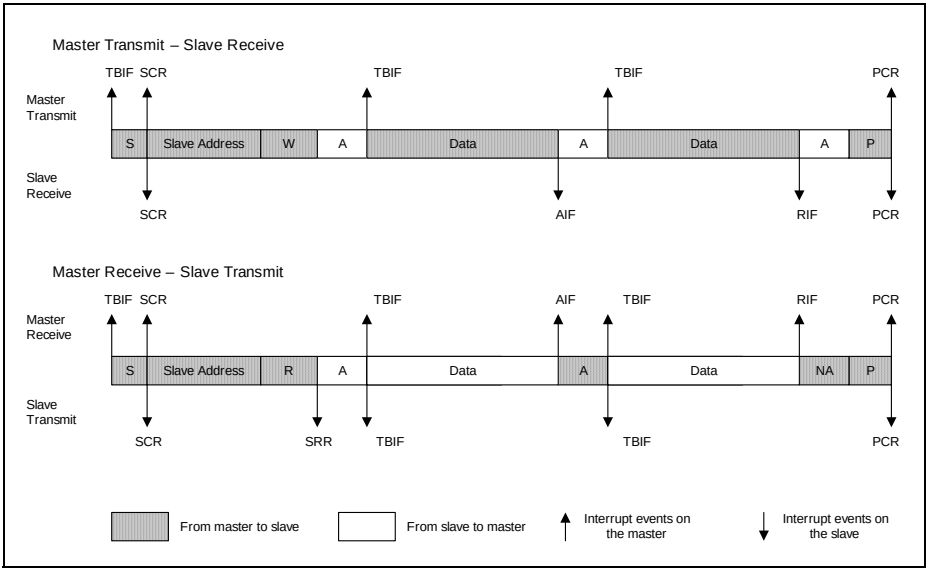


Figure 14-57 Interrupt Events on Data Transfers

14.5.4.4 Slave Transmit/Receive Modes

In slave receive mode, no TDF code needs to be written and data reception is indicated by the alternate receive (AIF) or receive (RIF) events.

In slave transmit mode, upon receiving its own slave address or general call address if this option is enabled, a slave read request event (SRR) will be triggered. The slave IIC then writes the TDF code 001_B and the requested data to TBUF to transmit the data to

the master. The slave does not check if the master reply with an ACK or NACK to the transmitted data.

In both cases, the data transfer is terminated by the master sending a STOP condition, which is indicated by a PCR event. See also **Figure 14-57**.

14.5.5 IIC Protocol Registers

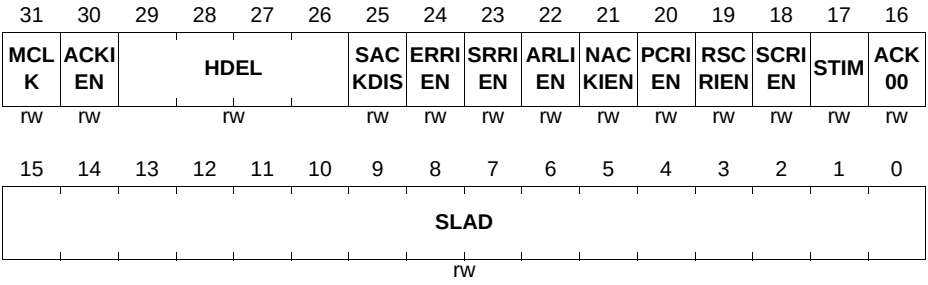
In IIC mode, the registers PCR and PSR handle IIC related information.

14.5.5.1 IIC Protocol Control Registers

In IIC mode, the PCR register bits or bit fields are defined as described in this section.

PCR
Protocol Control Register [IIC Mode]

($3C_H$) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
SLAD	[15:0]	rw	**Slave Address** This bit field contains the programmed slave address. The corresponding bits in the first received address byte are compared to the bits SLAD[15:9] to check for address match. If SLAD[15:11] = 11110_B, then the second address byte is also compared to SLAD[7:0].
ACK00	16	rw	**Acknowledge 00_H** This bit defines if a slave device should be sensitive to the slave address 00_H. 0_B The slave device is not sensitive to this address. 1_B The slave device is sensitive to this address.

Field	Bits	Type	Description
STIM	17	rw	**Symbol Timing** This bit defines how many time quanta are used in a symbol. 0_B A symbol contains 10 time quanta. The timing is adapted for standard mode (100 kBaud). 1_B A symbol contains 25 time quanta. The timing is adapted for fast mode (400 kBaud).
SCRIEN	18	rw	**Start Condition Received Interrupt Enable** This bit enables the generation of a protocol interrupt if a start condition is detected. 0_B The start condition interrupt is disabled. 1_B The start condition interrupt is enabled.
RSCRIEN	19	rw	**Repeated Start Condition Received Interrupt Enable** This bit enables the generation of a protocol interrupt if a repeated start condition is detected. 0_B The repeated start condition interrupt is disabled. 1_B The repeated start condition interrupt is enabled.
PCRIEN	20	rw	**Stop Condition Received Interrupt Enable** This bit enables the generation of a protocol interrupt if a stop condition is detected. 0_B The stop condition interrupt is disabled. 1_B The stop condition interrupt is enabled.
NACKIEN	21	rw	**Non-Acknowledge Interrupt Enable** This bit enables the generation of a protocol interrupt if a non-acknowledge is detected by a master. 0_B The non-acknowledge interrupt is disabled. 1_B The non-acknowledge interrupt is enabled.
ARLIEN	22	rw	**Arbitration Lost Interrupt Enable** This bit enables the generation of a protocol interrupt if an arbitration lost event is detected. 0_B The arbitration lost interrupt is disabled. 1_B The arbitration lost interrupt is enabled.
SRRIEN	23	rw	**Slave Read Request Interrupt Enable** This bit enables the generation of a protocol interrupt if a slave read request is detected. 0_B The slave read request interrupt is disabled. 1_B The slave read request interrupt is enabled.

Field	Bits	Type	Description
ERRIEN	24	rw	**Error Interrupt Enable** This bit enables the generation of a protocol interrupt if an IIC error condition is detected (indicated by PSR.ERR or PSR.WTDF). 0_B The error interrupt is disabled. 1_B The error interrupt is enabled.
SACKDIS	25	rw	**Slave Acknowledge Disable** This bit disables the generation of an active acknowledge signal for a slave device (active acknowledge = 0 level). Once set by software, it is automatically cleared with each (repeated) start condition. If this bit is set after a byte has been received (indicated by an interrupt) but before the next acknowledge bit has started, the next acknowledge bit will be sent with passive level. This would indicate that the receiver does not accept more bytes. As a result, a minimum of 2 bytes will be received if the first receive interrupt is used to set this bit. 0_B The generation of an active slave acknowledge is enabled (slave acknowledge with 0 level = more bytes can be received). 1_B The generation of an active slave acknowledge is disabled (slave acknowledge with 1 level = reception stopped).
HDEL	[29:26]	rw	**Hardware Delay** This bit field defines the delay used to compensate the internal treatment of the SCL signal (see **Page 14-117**) in order to respect the SDA hold time specified for the IIC protocol.
ACKIEN	30	rw	**Acknowledge Interrupt Enable** This bit enables the generation of a protocol interrupt if an acknowledge is detected by a master. 0_B The acknowledge interrupt is disabled. 1_B The acknowledge interrupt is enabled.
MCLK	31	rw	**Master Clock Enable** This bit enables generation of the master clock MCLK (not directly used for IIC protocol, can be used as general frequency output). 0_B The MCLK generation is disabled and MCLK is 0. 1_B The MCLK generation is enabled.

14.5.5.2 IIC Protocol Status Register

The following PSR status bits or bit fields are available in IIC mode. Please note that the bits in register PSR are not cleared by hardware.

The flags in the PSR register can be cleared by writing a 1 to the corresponding bit position in register PSCR. Writing a 1 to a bit position in PSR sets the corresponding flag, but does not lead to further actions (no interrupt generation). Writing a 0 has no effect. These flags should be cleared by software before enabling a new protocol.

PSR
Protocol Status Register [IIC Mode] (48_H) **Reset Value: $0000\ 0000_H$**

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AIF	RIF	TBIF	TSIF	DLIF	RSIF	ACK	ERR	SRR	ARL	NACK	PCR	RSCR	SCR	WTDF	SLSEL
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
SLSEL	0	rwh	**Slave Select** This bit indicates that this device has been selected as slave. 0_B The device is not selected as slave. 1_B The device is selected as slave.
WTDF	1	rwh	**Wrong TDF Code Found**[1)] This bit indicates that an unexpected/wrong TDF code has been found. A protocol interrupt can be generated if PCR.ERRIEN = 1. 0_B A wrong TDF code has not been found. 1_B A wrong TDF code has been found.
SCR	2	rwh	**Start Condition Received**[1)] This bit indicates that a start condition has been detected on the IIC bus lines.A protocol interrupt can be generated if PCR.SCRIEN = 1. 0_B A start condition has not yet been detected. 1_B A start condition has been detected.

Field	Bits	Type	Description
RSCR	3	rwh	**Repeated Start Condition Received**[1)] This bit indicates that a repeated start condition has been detected on the IIC bus lines. A protocol interrupt can be generated if PCR.RSCRIEN = 1. 0_B A repeated start condition has not yet been detected. 1_B A repeated start condition has been detected.
PCR	4	rwh	**Stop Condition Received**[1)] This bit indicates that a stop condition has been detected on the IIC bus lines. A protocol interrupt can be generated if PCR.PCRIEN = 1. 0_B A stop condition has not yet been detected. 1_B A stop condition has been detected.
NACK	5	rwh	**Non-Acknowledge Received**[1)] This bit indicates that a non-acknowledge has been received in master mode. This bit is not set in slave mode. A protocol interrupt can be generated if PCR.NACKIEN = 1. 0_B A non-acknowledge has not been received. 1_B A non-acknowledge has been received.
ARL	6	rwh	**Arbitration Lost**[1)] This bit indicates that an arbitration has been lost. A protocol interrupt can be generated if PCR.ARLIEN = 1. 0_B An arbitration has not been lost. 1_B An arbitration has been lost.
SRR	7	rwh	**Slave Read Request**[1)] This bit indicates that a slave read request has been detected. It becomes active to request the first data byte to be made available in the transmit buffer. For further consecutive data bytes, the transmit buffer issues more interrupts. For the end of the transfer, the master transmitter sends a stop condition. A protocol interrupt can be generated if PCR.SRRIEN = 1. 0_B A slave read request has not been detected. 1_B A slave read request has been detected.

Field	Bits	Type	Description
ERR	8	rwh	**Error**[1)] This bit indicates that an IIC error (frame format or TDF code) has been detected. A protocol interrupt can be generated if PCR.ERRIEN = 1. 0_B An IIC error has not been detected. 1_B An IIC error has been detected.
ACK	9	rwh	**Acknowledge Received**[1)] This bit indicates that an acknowledge has been received in master mode. This bit is not set in slave mode. A protocol interrupt can be generated if PCR.ACKIEN = 1. 0_B An acknowledge has not been received. 1_B An acknowledge has been received.
RSIF	10	rwh	**Receiver Start Indication Flag** 0_B A receiver start event has not occurred. 1_B A receiver start event has occurred.
DLIF	11	rwh	**Data Lost Indication Flag** 0_B A data lost event has not occurred. 1_B A data lost event has occurred.
TSIF	12	rwh	**Transmit Shift Indication Flag** 0_B A transmit shift event has not occurred. 1_B A transmit shift event has occurred.
TBIF	13	rwh	**Transmit Buffer Indication Flag** 0_B A transmit buffer event has not occurred. 1_B A transmit buffer event has occurred.
RIF	14	rwh	**Receive Indication Flag** 0_B A receive event has not occurred. 1_B A receive event has occurred.
AIF	15	rwh	**Alternative Receive Indication Flag** 0_B An alternative receive event has not occurred. 1_B An alternative receive event has occurred.
BRGIF	16	rwh	**Baud Rate Generator Indication Flag** 0_B A baud rate generator event has not occurred. 1_B A baud rate generator event has occurred.
0	[31:17]	r	**Reserved** Returns 0 if read; not modified in IIC mode.

1) This status bit can generate a protocol interrupt (see **Page 14-21**). The general interrupt status flags are described in the general interrupt chapter.

14.6 Inter-IC Sound Bus Protocol (IIS)

This chapter describes how the USIC module handles the IIS protocol. This serial protocol can handle reception and transmission of synchronous data frames between a device operating in master mode and a device in slave mode. An IIS connection based on a USIC communication channel supports half-duplex and full-duplex data transfers. The IIS mode is selected by CCR.MODE = 0011_B with CCFG.IIS = 1 (IIS mode is available).

14.6.1 Introduction

The IIS protocol is a synchronous serial communication protocol mainly for audio and infotainment applications **[8]**.

14.6.1.1 Signal Description

A connection between an IIS master and an IIS slave is based on the following signals:

- A shift clock signal SCK, generated by the transfer master. It is permanently generated while an IIS connection is established, also while no valid data bits are transferred.
- A word address signal WA (also named WS), generated by the transfer master. It indicates the beginning of a new data word and the targeted audio channel (e.g. left/right). The word address output signal WA is available on all SELOx outputs if the WA generation is enabled (by PCR.WAGEN = 1 for the transfer master). The WA signal changes synchronously to the falling edges of the shift clock.
- If the transmitter is the IIS master device, it generates a master transmit slave receive data signal. The data changes synchronously to the falling edges of the shift clock.
- If the transmitter is the IIS slave device, it generates a master receive slave transmit data signal. The data changes synchronously to the falling edges of the shift clock.

The transmitter part and the receiver part of the USIC communication channel can be used together to establish a full-duplex data connection between an IIS master and a slave device.

Table 14-18 IIS IO Signals

IIS Mode	Receive Data	Transmit Data	Shift Clock	Word Address
Master	Input DIN0, handled by DX0	Output DOUT0	Output SCLKOUT	Output(s) SELOx
Slave	Input DIN0, handled by DX0	Output DOUT0	Input SCLKIN, handled by DX1	Input SELIN, handled by DX2

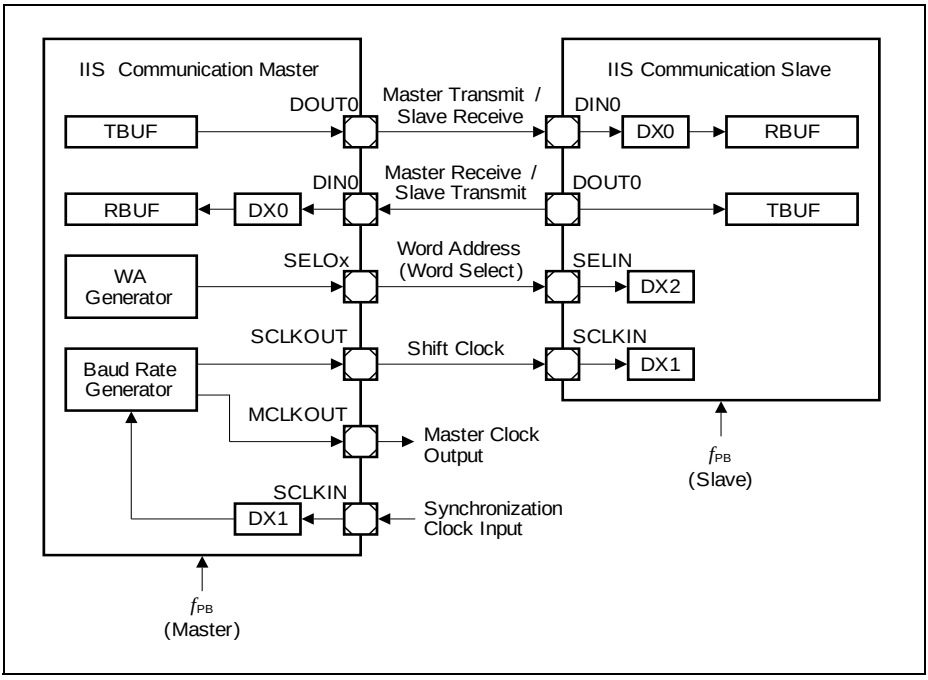


Figure 14-58 IIS Signals

Two additional signals are available for the USIC IIS communication master:

- A master clock output signal MCLKOUT with a fixed phase relation to the shift clock to support oversampling for audio components. It can also be used as master clock output of a communication network with synchronized IIS connections.
- A synchronization clock input SCLKIN for synchronization of the shift clock generation to an external frequency to support audio frequencies that can not be directly derived from the system clock f_{PB} of the communication master. It can be used as master clock input of a communication network with synchronized IIS connections.

14.6.1.2 Protocol Overview

An IIS connection supports transfers for two different data frames via the same data line, e.g. a data frames for the left audio channel and a data frame for the right audio channel. The word address signal WA is used to distinguish between the different data frames. Each data frame can consist of several data words.

In a USIC communication channel, data words are tagged for being transmitted for the left or for the right channel. Also the received data words contain a tag identifying the WA state when the data has been received.

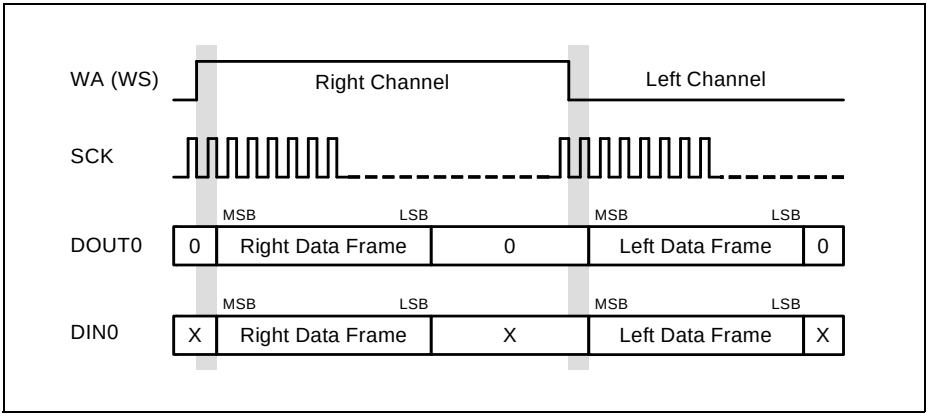


Figure 14-59 Protocol Overview

14.6.1.3 Transfer Delay

The transfer delay feature allows the transfer of data (transmission and reception) with a programmable delay (counted in shift clock periods).

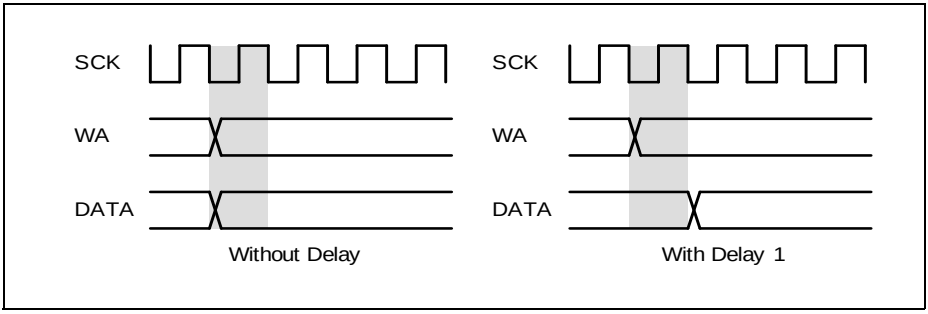


Figure 14-60 Transfer Delay for IIS

14.6.1.4 Connection of External Audio Components

The IIS signals can be used to communicate with external audio devices (such as Codecs) or other audio data sources/destinations.

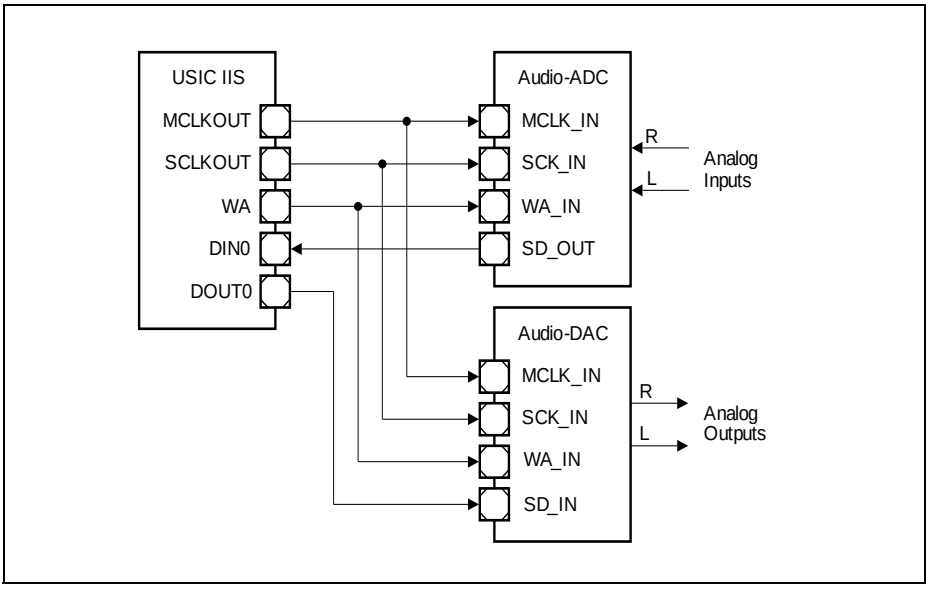


Figure 14-61 Connection of External Audio Devices

In some applications, especially for Audio-ADCs or Audio-DACs, a master clock signal is required with a fixed phase relation to the shift clock signal. The frequency of MCLKOUT is a multiple of the shift frequency SCLKOUT. This factor defines the oversampling factor of the external device (commonly used values: 256 or 384).

14.6.2 Operating the IIS

This chapter contains IIS issues, that are of general interest and not directly linked to master mode or slave mode.

14.6.2.1 Frame Length and Word Length Configuration

After each change of the WA signal, a complete data frame is intended to be transferred (frame length ≤ system word length). The number of data bits transferred after a change of signal WA is defined by SCTR.FLE. A data frame can consist of several data words with a data word length defined by SCTR.WLE. The changes of signal WA define the system word length as the number of SCLK cycles between two changes of WA (number of bits available for the right channel and same number available for the left channel).

If the system word length is longer than the frame length defined by SCTR.FLE, the additional bits are transmitted with passive data level (SCTR.PDL). If the system word

length is smaller than the device frame length, not all LSBs of the transmit data can be transferred.

It is recommended to program bits WLEMD, FLEMD and SELMD in register TCSR to 0.

14.6.2.2 Automatic Shadow Mechanism

The baud rate and shift control setting are internally kept constant while a data frame is transferred by an automatic shadow mechanism. The registers can be programmed all the time with new settings that are taken into account for the next data frame. During a data frame, the applied (shadowed) setting is not changed, although new values have been written after the start of the data frame. The setting is internally “frozen” with the start of each data frame.

Although this shadow mechanism being implemented, it is recommended to change the baud rate and shift control setting only while the IIS protocol is switched off.

14.6.2.3 Mode Control Behavior

In IIS mode, the following kernel modes are supported:

- Run Mode 0/1:
  Behavior as programmed, no impact on data transfers.
- Stop Mode 0/1:
  Bit PCR.WAGEN is internally considered as 0 (the bit itself is not changed). If WAGEN = 1, then the current system word cycle is finished and then the WA generation is stopped, but PSR.END is not set. The complete data frame is finished before entering stop mode, including a possible delay due to PCR.TDEL.
  When leaving a stop mode with WAGEN = 1, the WA generation starts from the beginning.

14.6.2.4 Transfer Delay

The transfer delay can be used to synchronize a data transfer to an event (e.g. a change of the WA signal). This event has to be synchronously generated to the falling edge of the shift clock SCK (like the change of the transmit data), because the input signal for the event is directly sampled in the receiver (as a result, the transmitter can use the detection information with its next edge).

Event signals that are asynchronous to the shift clock while the shift clock is running must not be used. In the example in **Figure 14-60**, the event (change of signal WA) is generated by the transfer master and as a result, is synchronous to the shift clock SCK. With the rising edge of SCK, signal WA is sampled and checked for a change. If a change is detected, a transfer delay counter TDC is automatically loaded with its programmable reload value (PCR.TDEL), otherwise it is decremented with each rising edge of SCK until it reaches 0, where it stops. The transfer itself is started if the value of TDC has become 0. This can happen under two conditions:

- TDC is reloaded with a PCR.TDEL = 0 when the event is detected
- TDC has reached 0 while counting down

The transfer delay counter is internal to the IIS protocol pre-processor and can not be observed by software. The transfer delay in SCK cycles is given by PCR.TDEL+1.

In the example in **Figure 14-62**, the reload value PCR.TDEL for TDC is 0. When the samples taken on receiver side show the change of the WA signal, the counter TDC is reloaded. If the reload value is 0, the data transfer starts with 1 shift clock cycle delay compared to the change of WA.

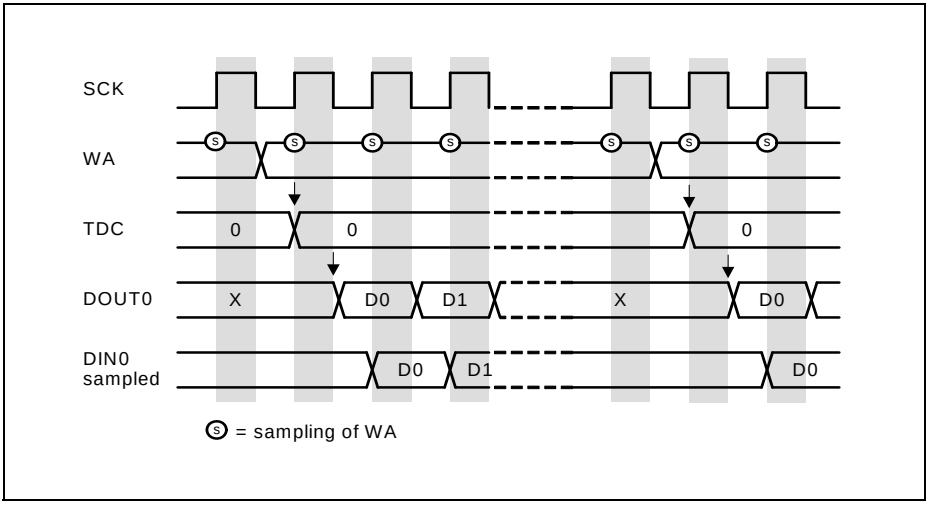


Figure 14-62 Transfer Delay with Delay 1

The ideal case without any transfer delay is shown in **Figure 14-63**. The WA signal changes and the data output value become valid at the same time. This implies that the transmitter “knows” in advance that the event signal will change with the next rising edge of TCLK. This is achieved by delaying the data transmission after the previously detected WA change the system word length minus 1.

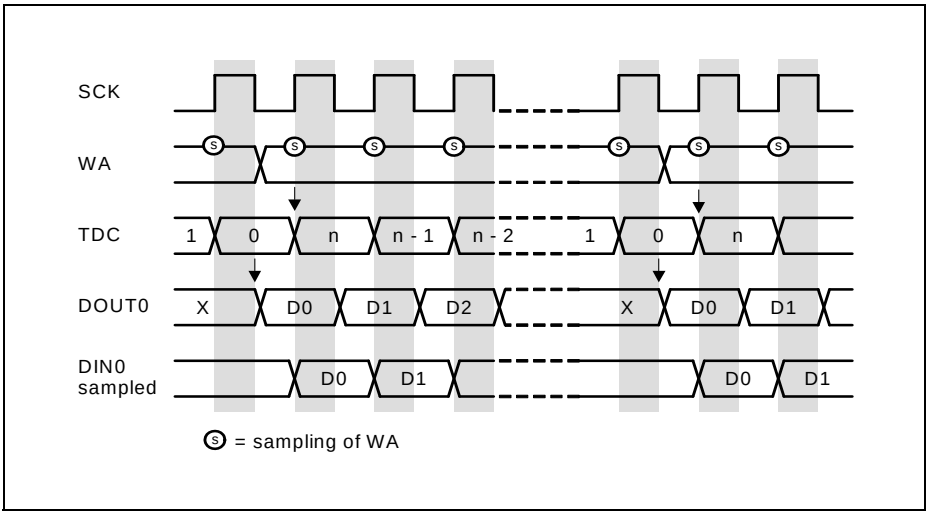


Figure 14-63 No Transfer Delay

If the end of the transfer delay is detected simultaneously to change of WA, the transfer is started and the delay counter is reloaded with PCR.TDEL. This allows to run the USIC as IIS device without any delay. In this case, internally the delay from the previous event elapses just at the moment when a new event occurs. If PCR.TDEL is set to a value bigger than the system word length, no transfer takes place.

14.6.2.5 Parity Mode

Parity generation is not supported in IIS mode and bit field CCR.PM = 00_B has to be programmed.

14.6.2.6 Transfer Mode

In IIS mode, bit field SCTR.TRM = 11_B has to be programmed to allow data transfers. Setting SCTR.TRM = 00_B disables and stops the data transfer immediately.

14.6.2.7 Data Transfer Interrupt Handling

The data transfer interrupts indicate events related to IIS frame handling.

- Transmit buffer interrupt TBI:
  Bit PSR.TBIF is set after the start of first data bit of a data word.
- Transmit shift interrupt TSI:
  Bit PSR.TSIF is set after the start of the last data bit of a data word.

- Receiver start interrupt RSI:
  Bit PSR.RSIF is set after the reception of the first data bit of a data word.
  With this event, bit TCSR.TDV is cleared and new data can be loaded to the transmit buffer.
- Receiver interrupt RI and alternative interrupt AI:
  Bit PSR.RIF is set at after the reception of the last data bit of a data word with WA = 0.
  Bit RBUFSR.SOF indicates whether the received data word has been the first data word of a new data frame.
  Bit PSR.AIF is set at after the reception of the last data bit of a data word with WA = 1.
  Bit RBUFSR.SOF indicates whether the received data word has been the first data word of a new data frame.

14.6.2.8 Baud Rate Generator Interrupt Handling

The baud rate generator interrupt indicate that the capture mode timer has reached its maximum value. With this event, the bit PSR.BRGIF is set.

14.6.2.9 Protocol-Related Argument and Error

In order to distinguish between data words received for the left or the right channel, the IIS protocol pre-processor samples the level of the WA input (just after the WA transition) and propagates it as protocol-related error (although it is not an error, but an indication) to the receive buffer status register at the bit position RBUFSR[9]. This bit position defines if either a standard receive interrupt (if RBUFSR[9] = 0) or an alternative receive interrupt (if RBUFSR[9] = 1) becomes activated when a new data word has been received. Incoming data can be handled by different interrupts for the left and the right channel if the corresponding events are directed to different interrupt nodes. Flag PAR is always 0.

14.6.2.10 Transmit Data Handling

The IIS protocol pre-processor allows to distinguish between the left and the right channel for data transmission. Therefore, bit TCSR.WA indicates on which channel the data in the buffer will be transmitted. If TCSR.WA = 0, the data will be transmitted after a falling edge of WA. If TCSR.WA = 1, the data will be transmitted after a rising edge of WA. The WA value sampled after the WA transition is considered to distinguish between both channels (referring to PSR.WA).

Bit TCSR.WA can be automatically updated by the transmit control information TCI[4] for each data word if TCSR.WAMD = 1. In this case, data written to TBUF[15:0] (or IN[15:0] if a FIFO data buffer is used) is considered as left channel data, whereas data written to TBUF[31:16] (or IN[31:16] if a FIFO data buffer is used) is considered as right channel data.

14.6.2.11 Receive Buffer Handling

If a receive FIFO buffer is available (CCFG.RB = 1) and enabled for data handling (RBCTR.SIZE > 0), it is recommended to set RBCTR.RCIM = 11_B in IIS mode. This leads to an indication that the data word has been the first data word of a new data frame if bit OUTR.RCI[0] = 1, and the channel indication by the sampled WA value is given by OUTR.RCI[4].

The standard receive buffer event and the alternative receive buffer event can be used for the following operation in RCI mode (RBCTR.RNM = 1):

- A standard receive buffer event indicates that a data word can be read from OUTR that belongs to a data frame started when WA = 0.
- An alternative receive buffer event indicates that a data word can be read from OUTR that belongs to a data frame started when WA = 1.

14.6.2.12 Loop-Delay Compensation

The synchronous signaling mechanism of the IIS protocol being similar to the one of the SSC protocol, the closed-loop delay has to be taken into account for the application setup. In IIS mode, loop-delay compensation in master mode is also possible to achieve higher baud rates.

Please refer to the more detailed description in the SSC chapter.

14.6.3 Operating the IIS in Master Mode

In order to operate the IIS in master mode, the following issues have to be considered:

- Select IIS mode:
  It is recommended to configure all parameters of the IIS that do not change during run time while CCR.MODE = 0000_B. Bit field SCTR.TRM = 11_B has to be programmed. The configuration of the input stages has to be done while CCR.MODE = 0000_B to avoid unintended edges of the input signals and the IIS mode can be enabled by CCR.MODE = 0011_B afterwards.
- Pin connection for data transfer:
  Establish a connection of input stage DX0 with the selected receive data input pin (DIN0) with DX0CR.INSW = 1. Configure a transmit data output pin (DOUT0) for a transmitter.
  The data shift unit allowing full-duplex data transfers based on the same WA signal, the values delivered by the DX0 stage are considered as data bits (receive function can not be disabled independently from the transmitter). To receive IIS data, the transmitter does not necessarily need to be configured (no assignment of DOUT0 signal to a pin).
- Baud rate generation:
  The desired baud rate setting has to be selected, comprising the fractional divider and the baud rate generator. Bit DX1CR.INSW = 0 has to be programmed to use the

baud rate generator output SCLK directly as input for the data shift unit. Configure a shift clock output pin with the inverted signal SCLKOUT without additional delay (BRG.SCLKCFG = 01_B).

- Word address WA generation:
  The WA generation has to be enabled by setting PCR.WAGEN = 1 and the programming of the number of shift clock cycles between the changes of WA. Bit DX2CR.INSW = 0 has to be programmed to use the WA generator as input for the data shift unit. Configure WA output pin for signal SELOx if needed.
- Data format configuration:
  The word length, the frame length, and the shift direction have to be set up according to the application requirements by programming the register SCTR. Generally, the MSB is shifted first (SCTR.SDIR = 1).
  Bit TCSR.WAMD can be set to use the transmit control information TCI[4] to distinguish the data words for transmission while WA = 0 or while WA = 1.

Note: The step to enable the alternate output port functions should only be done after the IIS mode is enabled, to avoided unintended spikes on the output.

14.6.3.1 Baud Rate Generation

The baud rate is defined by the frequency of the SCLK signal (one period of f_{SCLK} represents one data bit).

If the fractional divider mode is used to generate f_{PIN}, there can be an uncertainty of one period of f_{PB} for f_{PIN}. This uncertainty does not accumulate over several SCLK cycles. As a consequence, the average frequency is reached, whereas the duty cycle of 50% of the SCLK and MCLK signals can vary by one period of f_{PB}.

In IIS applications, where the phase relation between the optional MCLK output signal and SCLK is not relevant, SCLK can be based on the frequency f_{PIN} (BRG.PPPEN = 0). In the case that a fixed phase relation between the MCLK signal and SCLK is required (e.g. when using MCLK as clock reference for external devices), the additional divider by 2 stage has to be taken into account (BRG.PPPEN = 1). This division is due to the fact that signal MCLK toggles with each cycle of f_{PIN}. Signal SCLK is then based on signal MCLK, see **Figure 14-64**.

The adjustable integer divider factor is defined by bit field BRG.PDIV.

$$f_{SCLK} = \frac{f_{PIN}}{2} \times \frac{1}{PDIV + 1} \quad \text{if PPPEN = 0}$$
$$f_{SCLK} = \frac{f_{PIN}}{2 \times 2} \times \frac{1}{PDIV + 1} \quad \text{if PPPEN = 1} \qquad (14.12)$$

Note: In the IIS protocol, the master (unit generating the shift clock and the WA signal) changes the status of its data and WA output line with the falling edge of SCK. The slave transmitter also has to transmit on falling edges. The sampling of the

received data is done with the rising edges of SCLK. The input stage DX1 and the SCLKOUT have to be programmed to invert the shift clock signal to fit to the internal signals.

14.6.3.2 WA Generation

The word address (or word select) line WA regularly toggles after N cycles of signal SCLK. The time between the changes of WA is called system word length and can be programmed by using the following bit fields.

In IIS master mode, the system word length is defined by:

- BRG.CTQSEL = 10_B
  to base the WA toggling on SCLK
- BRG.PCTQ
  to define the number N of SCLK cycles per system word length
- BRG.DCTQ
  to define the number N of SCLK cycles per system word length

$$N = (PCTQ + 1) \times (DCTQ + 1) \quad (14.13)$$

14.6.3.3 Master Clock Output

The master clock signal MCLK can be generated by the master of the IIS transfer (BRG.PPPEN = 1). It is used especially to connect external Codec devices. It can be configured by bit BRG.MCLKCFG in its polarity to become the output signal MCLKOUT.

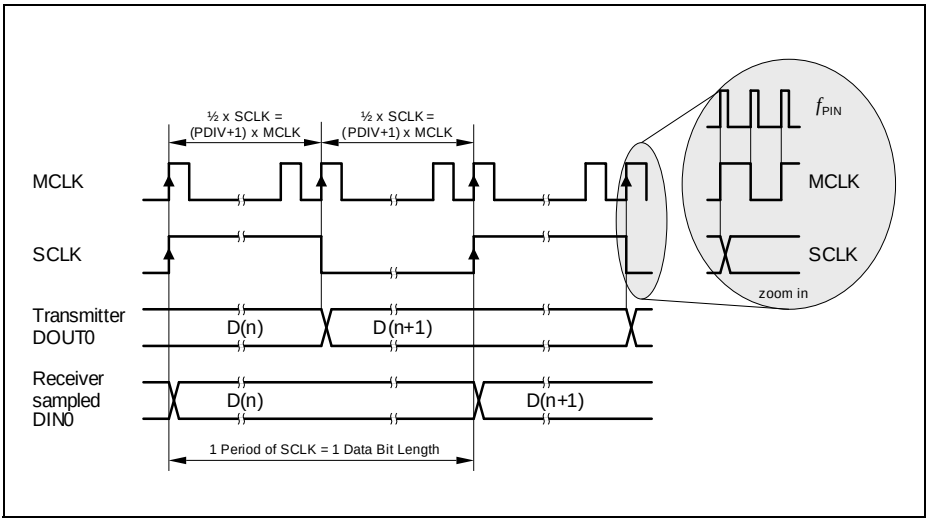


Figure 14-64 MCLK and SCLK for IIS

14.6.3.4 Protocol Interrupt Events

The following protocol-related events are generated in IIS mode and can lead to a protocol interrupt.

Please note that the bits in register PSR are not all automatically cleared by hardware and have to be cleared by software in order to monitor new incoming events.

- WA rising/falling edge events:
  The WA generation block indicates two events that are monitored in register PSR. Flag PSR.WAFE is set with the falling edge, flag PSR.WARE with the rising edge of the WA signal. A protocol interrupt can be generated if PCR.WAFEIEN = 1 for the falling edge, similar for PCR.WAREIEN = 1 for a rising edge.
- WA end event:
  The WA generation block also indicates when it has stopped the WA generation after it has been disabled by writing PCR.WAGEN = 0. A protocol interrupt can be generated if PCR.ENDIEN = 1.
- DX2T event:
  An activation of the trigger signal DX2T is indicated by PSR.DX2TEV = 1 and can generate a protocol interrupt if PCR.DX2TIEN = 1. This event can be evaluated instead of the WA rising/falling events if a delay compensation like in SSC mode (for details, refer to corresponding SSC section) is used.

14.6.4 Operating the IIS in Slave Mode

In order to operate the IIS in slave mode, the following issues have to be considered:

- Select IIS mode:
  It is recommended to configure all parameters of the IIS that do not change during run time while CCR.MODE = 0000_B. Bit field SCTR.TRM = 11_B has to be programmed. The configuration of the input stages has to be done while CCR.MODE = 0000_B to avoid unintended edges of the input signals and the IIS mode can be enabled by CCR.MODE = 0011_B afterwards.
- Pin connection for data transfer:
  Establish a connection of input stage DX0 with the selected receive data input pin (DIN0) with DX0CR.INSW = 1. Configure a transmit data output pin (DOUT0) for a transmitter.
  The data shift unit allowing full-duplex data transfers based on the same WA signal, the values delivered by the DX0 stage are considered as data bits (receive function can not be disabled independently from the transmitter). To receive IIS data, the transmitter does not necessarily need to be configured (no assignment of DOUT0 signal to a pin).
  Note that the step to enable the alternate output port functions should only be done after the IIS mode is enabled, to avoided unintended spikes on the output.
- Pin connection for shift clock:
  Establish a connection of input stage DX1 with the selected shift clock input pin (SCLKIN) with DX1CR.INSW = 1 and with inverted polarity (DX1CR.DPOL = 1).
- Pin connection for WA input:
  Establish a connection of input stage DX2 with the WA input pin (SELIN) with DX2CR.INSW = 1.
- Baud rate generation:
  The baud rate generator is not needed and can be switched off by the fractional divider.
- WA generation:
  The WA generation is not needed and can be switched off (PCR.WAGEN = 0).

14.6.4.1 Protocol Events and Interrupts

The following protocol-related event is generated in IIS mode and can lead to a protocol interrupt.

Please note that the bits in register PSR are not all automatically cleared by hardware and have to be cleared by software in order to monitor new incoming events.

- WA rising/falling/end events:
  The WA generation being switched off, these events are not available.
- DX2T event:
  An activation of the trigger signal DX2T is indicated by PSR.DX2TEV = 1 and can generate a protocol interrupt if PCR.DX2TIEN = 1.

14.6.5 IIS Protocol Registers

In IIS mode, the registers PCR and PSR handle IIS related information.

14.6.5.1 IIS Protocol Control Registers

In IIS mode, the PCR register bits or bit fields are defined as described in this section.

PCR
Protocol Control Register [IIS Mode]

($3C_H$) **Reset Value: $0000\ 0000_H$**

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DX2TIEN	0								ENDIEN	WAREIEN	WAFEIEN	0	SELINV	DTEN	WAGEN
rw	rw								rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
WAGEN	0	rw	**WA Generation Enable** This bit enables/disables the generation of word address control output signal WA. 0_B The IIS can be used as slave. The generation of the word address signal is disabled. The output signal WA is 0. The MCLKO signal generation depends on PCR.MCLK. 1_B The IIS can be used as master. The generation of the word address signal is enabled. The signal starts with a 0 after being enabled. The generation of MCLK is enabled, independent of PCR.MCLK. After clearing WAGEN, the USIC module stops the generation of the WA signal within the next 4 WA periods.

Field	Bits	Type	Description
DTEN	1	rw	**Data Transfers Enable** This bit enables/disables the transfer of IIS frames as a reaction to changes of the input word address control line WA. 0_B The changes of the WA input signal are ignored and no transfers take place. 1_B Transfers are enabled.
SELINV	2	rw	**Select Inversion** This bit defines if the polarity of the SELOx outputs in relation to the internally generated word address signal WA. 0_B The SELOx outputs have the same polarity as the WA signal. 1_B The SELOx outputs have the inverted polarity to the WA signal.
WAFEIEN	4	rw	**WA Falling Edge Interrupt Enable** This bit enables/disables the activation of a protocol interrupt when a falling edge of WA has been generated. 0_B A protocol interrupt is not activated if a falling edge of WA is generated. 1_B A protocol interrupt is activated if a falling edge of WA is generated.
WAREIEN	5	rw	**WA Rising Edge Interrupt Enable** This bit enables/disables the activation of a protocol interrupt when a rising edge of WA has been generated. 0_B A protocol interrupt is not activated if a rising edge of WA is generated. 1_B A protocol interrupt is activated if a rising edge of WA is generated.
ENDIEN	6	rw	**END Interrupt Enable** This bit enables/disables the activation of a protocol interrupt when the WA generation stops after clearing PCR.WAGEN (complete system word length is processed before stopping). 0_B A protocol interrupt is not activated. 1_B A protocol interrupt is activated.

Field	Bits	Type	Description
DX2TIEN	15	rw	**DX2T Interrupt Enable** This bit enables/disables the generation of a protocol interrupt if the DX2T signal becomes activated (indicated by PSR.DX2TEV = 1). 0_B A protocol interrupt is not generated if DX2T is active. 1_B A protocol interrupt is generated if DX2T is active.
TDEL	[21:16]	rw	**Transfer Delay** This bit field defines the transfer delay when an event is detected. If bit field TDEL = 0, the additional delay functionality is switched off and a delay of one shift clock cycle is introduced.
MCLK	31	rw	**Master Clock Enable** This bit enables generation of the master clock MCLK (not directly used for IIC protocol, can be used as general frequency output). 0_B The MCLK generation is disabled and MCLK is 0. 1_B The MCLK generation is enabled.
0	3, [14:7], [30:22]	rw	**Reserved** Returns 0 if read; should be written with 0;

14.6.5.2 IIS Protocol Status Register

The following PSR status bits or bit fields are available in IIS mode. Please note that the bits in register PSR are not cleared by hardware.

The flags in the PSR register can be cleared by writing a 1 to the corresponding bit position in register PSCR. Writing a 1 to a bit position in PSR sets the corresponding flag, but does not lead to further actions (no interrupt generation). Writing a 0 has no effect. These flags should be cleared by software before enabling a new protocol.

PSR
Protocol Status Register [IIS Mode] (48_H) **Reset Value: 0000 0000$_H$**

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AIF	RIF	TBIF	TSIF	DLIF	RSIF	0			END	WARE	WAFE	DX2TEV	0	DX2S	WA
rwh	rwh	rwh	rwh	rwh	rwh	r			rwh	rwh	rwh	rwh	r	rwh	rwh

Field	Bits	Type	Description
WA	0	rwh	**Word Address** This bit indicates the status of the WA input signal, sampled after a transition of WA has been detected. This information is forwarded to the corresponding bit position RBUFSR[9] to distinguish between data received for the right and the left channel. 0_B WA has been sampled 0. 1_B WA has been sampled 1.
DX2S	1	rwh	**DX2S Status** This bit indicates the current status of the DX2S signal, which is used as word address signal WA. 0_B DX2S is 0. 1_B DX2S is 1.
DX2TEV	3	rwh	**DX2T Event Detected**[1)] This bit indicates that the DX2T signal has been activated. In IIS slave mode, an activation of DX2T generates a protocol interrupt if PCR.DX2TIEN = 1. 0_B The DX2T signal has not been activated. 1_B The DX2T signal has been activated.
WAFE	4	rwh	**WA Falling Edge Event**[1)] This bit indicates that a falling edge of the WA output signal has been generated. This event generates a protocol interrupt if PCR.WAFEIEN = 1. 0_B A WA falling edge has not been generated. 1_B A WA falling edge has been generated.

Field	Bits	Type	Description
WARE	5	rwh	**WA Rising Edge Event[1)]** This bit indicates that a rising edge of the WA output signal has been generated. This event generates a protocol interrupt if PCR.WAREIEN = 1. 0_B A WA rising edge has not been generated. 1_B A WA rising edge has been generated.
END	6	rwh	**WA Generation End[1)]** This bit indicates that the WA generation has ended after clearing PCR.WAGEN. This bit should be cleared by software before clearing WAGEN. 0_B The WA generation has not yet ended (if it is running and WAGEN has been cleared). 1_B The WA generation has ended (if it has been running).
RSIF	10	rwh	**Receiver Start Indication Flag** 0_B A receiver start event has not occurred. 1_B A receiver start event has occurred.
DLIF	11	rwh	**Data Lost Indication Flag** 0_B A data lost event has not occurred. 1_B A data lost event has occurred.
TSIF	12	rwh	**Transmit Shift Indication Flag** 0_B A transmit shift event has not occurred. 1_B A transmit shift event has occurred.
TBIF	13	rwh	**Transmit Buffer Indication Flag** 0_B A transmit buffer event has not occurred. 1_B A transmit buffer event has occurred.
RIF	14	rwh	**Receive Indication Flag** 0_B A receive event has not occurred. 1_B A receive event has occurred.
AIF	15	rwh	**Alternative Receive Indication Flag** 0_B An alternative receive event has not occurred. 1_B An alternative receive event has occurred.
BRGIF	16	rwh	**Baud Rate Generator Indication Flag** 0_B A baud rate generator event has not occurred. 1_B A baud rate generator event has occurred.
0	2, [9:7], [31:17]	r	**Reserved** Returns 0 if read; not modified in IIS mode.

1) This status bit can generate a protocol interrupt (see **Page 14-21**). The general interrupt status flags are described in the general interrupt chapter.

14.7 Service Request Generation

The USIC module provides 6 service request outputs SR[5:0] to be shared between two channels. The service request outputs SR[5:0] are connected to interrupt nodes in the Nested Vectored Interrupt Controller (NVIC).

Each USIC communication channel can be connected to up to 6 service request handlers (connected to USICx.SR[5:0], though 3 or 4 are normally used, e.g. one for transmission, one for reception, one or two for protocol or error handling, or for the alternative receive events).

14.8 Debug Behaviour

Each USIC communication channel can be pre-configured to enter one of four kernel modes, when the program execution of the CPU is halted by the debugger.

Refer to **Section 14.2.2.2** for details.

14.9 Power, Reset and Clock

The USIC module is located in the core power domain. The module can be reset to its default state by a system reset.

The USIC module is clocked by the main clock, MCLK, from SCU. MCLK is disabled by default and can be enabled via the SCU_CGATCLR0 register. Enabling and disabling the module clock could cause a load change and clock blanking could occur as described in the CCU (Clock Gating Control) section of the SCU chapter. It is strongly recommended to set up the module clock in the user initialization code to avoid clock blanking during runtime.

Note: To differentiate from the USIC baud rate generator output, master clock (MCLK), the SCU MCLK is referenced throughout the USIC chapter as f_{PB}.

14.10 Initialization and System Dependencies

The application has to apply the following initialization sequence before operating the USIC module:

- Enable the module by writing 1s to the MODEN and BPMODEN bits in KSCFG register.

14.11 Registers

Table 14-19 shows all registers which are required for programming a USIC channel, as well as the FIFO buffer. It summarizes the USIC communication channel registers and defines the relative addresses and the reset values.

Please note that all registers can be accessed with any access width (8-bit, 16-bit, 32-bit), independent of the described width.

All USIC registers (except bit field KSCFG.SUMCFG) are always reset by a system reset. Bit field KSCFG.SUMCFG is reset by a debug reset.

Note: The register bits marked "w" always deliver 0 when read. They are used to modify flip-flops in other registers or to trigger internal actions.

Figure 14-65 shows the register types of the USIC module registers and channel registers. In a specific microcontroller, module registers of USIC module "x" are marked by the module prefix "USICx_". Channel registers of USIC module "x" are marked by the channel prefix "USICx_CH0_" and "USICx_CH1_".

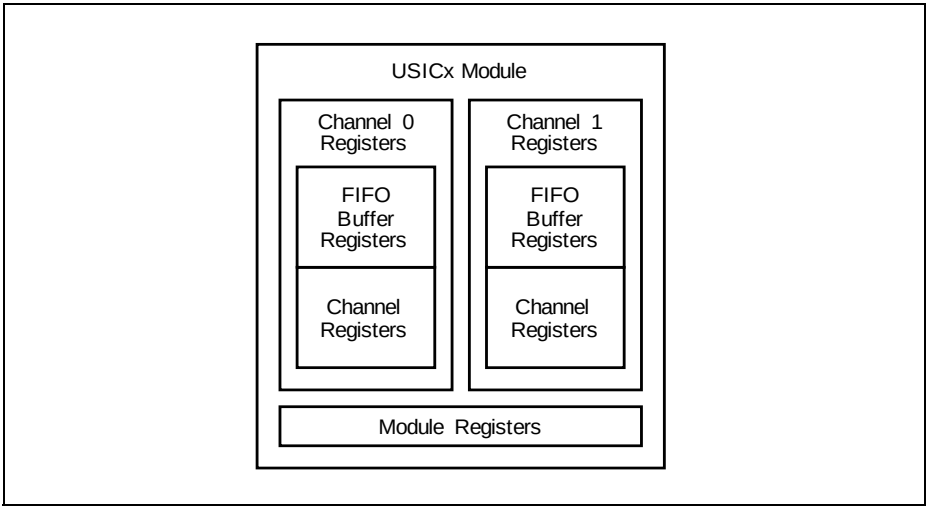


Figure 14-65 USIC Module and Channel Registers

Table 14-19 USIC Kernel-Related and Kernel Registers

Register Short Name	Register Long Name	Offset Addr.	Access Mode		Description see
			Read	Write	
Module Registers[1)]					
ID	Module Identification Register	008_H	U, PV	U, PV	**Page 14-158**
Channel Registers					
–	reserved	000_H	BE	BE	–
CCFG	Channel Configuration Register	004_H	U, PV	U, PV	**Page 14-163**
KSCFG	Kernel State Configuration Register	$00C_H$	U, PV	U, PV	**Page 14-164**
FDR	Fractional Divider Register	010_H	U, PV	PV	**Page 14-177**

Table 14-19 USIC Kernel-Related and Kernel Registers (cont'd)

Register Short Name	Register Long Name	Offset Addr.	Access Mode		Description see
			Read	Write	
BRG	Baud Rate Generator Register	014_H	U, PV	PV	Page 14-178
INPR	Interrupt Node Pointer Register	018_H	U, PV	U, PV	Page 14-167
DX0CR	Input Control Register 0	$01C_H$	U, PV	U, PV	Page 14-172
DX1CR	Input Control Register 1	020_H	U, PV	U, PV	Page 14-174
DX2CR	Input Control Register 2	024_H	U, PV	U, PV	Page 14-172
DX3CR	Input Control Register 3	028_H	U, PV	U, PV	
DX4CR	Input Control Register 4	$02C_H$	U, PV	U, PV	
DX5CR	Input Control Register 5	030_H	U, PV	U, PV	
SCTR	Shift Control Register	034_H	U, PV	U, PV	Page 14-182
TCSR	Transmit Control/Status Register	038_H	U, PV	U, PV	Page 14-185
PCR	Protocol Control Register	$03C_H$	U, PV	U, PV	Page 14-168 2)
			U, PV	U, PV	Page 14-65 3)
			U, PV	U, PV	Page 14-97 4)
			U, PV	U, PV	Page 14-128 5)
			U, PV	U, PV	Page 14-147 6)
CCR	Channel Control Register	040_H	U, PV	PV	Page 14-159
CMTR	Capture Mode Timer Register	044_H	U, PV	U, PV	Page 14-181
PSR	Protocol Status Register	048_H	U, PV	U, PV	Page 14-169 2)
			U, PV	U, PV	Page 14-69 3)
			U, PV	U, PV	Page 14-101 4)
			U, PV	U, PV	Page 14-131 5)
			U, PV	U, PV	Page 14-150 6)
PSCR	Protocol Status Clear Register	$04C_H$	U, PV	U, PV	Page 14-170

Table 14-19 USIC Kernel-Related and Kernel Registers (cont'd)

Register Short Name	Register Long Name	Offset Addr.	Access Mode		Description see
			Read	Write	
RBUFSR	Receiver Buffer Status Register	050_H	U, PV	U, PV	Page 14-203
RBUF	Receiver Buffer Register	054_H	U, PV	U, PV	Page 14-201
RBUFD	Receiver Buffer Register for Debugger	058_H	U, PV	U, PV	Page 14-202
RBUF0	Receiver Buffer Register 0	$05C_H$	U, PV	U, PV	Page 14-194
RBUF1	Receiver Buffer Register 1	060_H	U, PV	U, PV	Page 14-195
RBUF01SR	Receiver Buffer 01 Status Register	064_H	U, PV	U, PV	Page 14-196
FMR	Flag Modification Register	068_H	U, PV	U, PV	Page 14-192
–	reserved; do not access this location	$06C_H$	U, PV	BE	–
–	reserved	070_H - $07C_H$	BE	BE	–
TBUFx	Transmit Buffer Input Location x (x = 00-31)	080_H + x*4	U, PV	U, PV	Page 14-194
FIFO Buffer Registers					
BYP	Bypass Data Register	100_H	U, PV	U, PV	Page 14-204
BYPCR	Bypass Control Register	104_H	U, PV	U, PV	Page 14-205
TBCTR	Transmit Buffer Control Register	108_H	U, PV	U, PV	Page 14-213
RBCTR	Receive Buffer Control Register	$10C_H$	U, PV	U, PV	Page 14-217
TRBPTR	Transmit/Receive Buffer Pointer Register	110_H	U, PV	U, PV	Page 14-225
TRBSR	Transmit/Receive Buffer Status Register	114_H	U, PV	U, PV	Page 14-208
TRBSCR	Transmit/Receive Buffer Status Clear Register	118_H	U, PV	U, PV	Page 14-212
OUTR	Receive Buffer Output Register	$11C_H$	U, PV	U, PV	Page 14-223
OUTDR	Receive Buffer Output Register for Debugger	120_H	U, PV	U, PV	Page 14-224

Table 14-19 USIC Kernel-Related and Kernel Registers (cont'd)

Register Short Name	Register Long Name	Offset Addr.	Access Mode		Description see
			Read	Write	
–	reserved	124_H - $17C_H$	BE	BE	–
INx	Transmit FIFO Buffer Input Location x (x = 00-31)	180_H + x*4	U, PV	U, PV	Page 14-222

1) Details of the module identification registers are described in the implementation section (see Page 14-158).
2) This page shows the general register layout.
3) This page shows the register layout in ASC mode.
4) This page shows the register layout in SSC mode.
5) This page shows the register layout in IIC mode.
6) This page shows the register layout in IIS mode.

14.11.1 Address Map

The registers of the USIC communication channel are available at the following base addresses. The exact register address is given by the relative address of the register (given in Table 14-19) plus the channel base address (given in Table 14-20).

Table 14-20 Registers Address Space

Module	Base Address	End Address	Note
USIC0_CH0	48000000_H	$480001FF_H$	–
USIC0_CH1	48000200_H	$480003FF_H$	–

Table 14-21 FIFO and Reserved Address Space

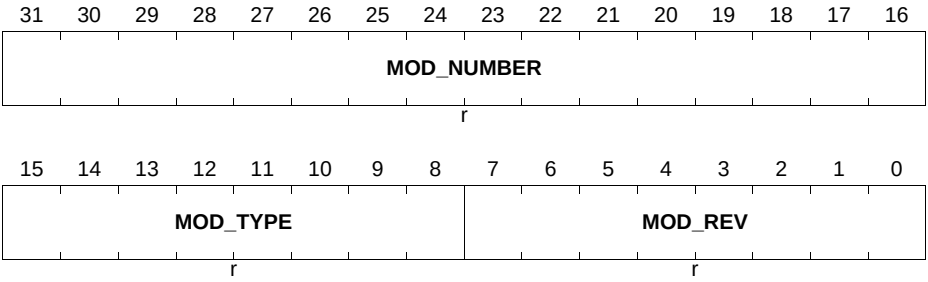
Module	Base Address	End Address	Access Mode		Note
			Read	Write	
USIC0	48000400_H	$480007FF_H$	nBE	nBE if in direct RAM test mode; otherwise BE	USIC0 RAM area, shared between USIC0_CH0 and USIC0_CH1
reserved	48000800_H	$4800FFFF_H$	BE	BE	–

14.11.2 Module Identification Registers

The module identification registers indicate the function and the design step of the USIC modules.

USIC0_ID
Module Identification Register

(4800 0008$_H$) **Reset Value: 00AA C0XX$_H$**



Field	Bits	Type	Description
MOD_REV	[7:0]	r	**Module Revision Number** MOD_REV defines the revision number. The value of a module revision starts with 01$_H$ (first revision).
MOD_TYPE	[15:8]	r	**Module Type** This bit field is C0$_H$. It defines the module as a 32-bit module.
MOD_NUMBER	[31:16]	r	**Module Number Value** This bit field defines the USIC module identification number (00AA$_H$ = USIC).

14.11.3 Channel Control and Configuration Registers

14.11.3.1 Channel Control Register

The channel control register contains the enable/disable bits for hardware port control and interrupt generation on channel events, the control of the parity generation and the protocol selection of a USIC channel.

FDR can be written only with a privilege mode access.

CCR
Channel Control Register **(40_H)** **Reset Value: 0000 0000$_H$**

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AIEN	RIEN	TBIEN	TSIEN	DLIEN	RSIEN	PM		HPCEN		0		MODE			
rw	rw	rw	rw	rw	rw	rw		rw		r		rw			

Field	Bits	Type	Description
MODE	[3:0]	rw	**Operating Mode** This bit field selects the protocol for this USIC channel. Selecting a protocol that is not available (see register CCFG) or a reserved combination disables the USIC channel. When switching between two protocols, the USIC channel has to be disabled before selecting a new protocol. In this case, registers PCR and PSR have to be cleared or updated by software. 0_H The USIC channel is disabled. All protocol-related state machines are set to an idle state. 1_H The SSC (SPI) protocol is selected. 2_H The ASC (SCI, UART) protocol is selected. 3_H The IIS protocol is selected. 4_H The IIC protocol is selected. Other bit combinations are reserved.

Field	Bits	Type	Description
HPCEN	[7:6]	rw	**Hardware Port Control Enable** This bit enables the hardware port control for the specified set of DX[3:0] and DOUT[3:0] pins. 00_B The hardware port control is disabled. 01_B The hardware port control is enabled for DX0 and DOUT0. 10_B The hardware port control is enabled for DX3, DX0 and DOUT[1:0]. 11_B The hardware port control is enabled for DX0, DX[5:3] and DOUT[3:0]. *Note: The hardware port control feature is useful only for SSC protocols in half-duplex configurations, such as dual- and quad-SSC. For all other protocols HPCEN must always be written with 00_B.*
PM	[9:8]	rw	**Parity Mode** This bit field defines the parity generation of the sampled input values. 00_B The parity generation is disabled. 01_B Reserved 10_B Even parity is selected (parity bit = 1 on odd number of 1s in data, parity bit = 0 on even number of 1s in data). 11_B Odd parity is selected (parity bit = 0 on odd number of 1s in data, parity bit = 1 on even number of 1s in data).
RSIEN	10	rw	**Receiver Start Interrupt Enable** This bit enables the interrupt generation in case of a receiver start event. 0_B The receiver start interrupt is disabled. 1_B The receiver start interrupt is enabled. In case of a receiver start event, the service request output SRx indicated by INPR.TBINP is activated.

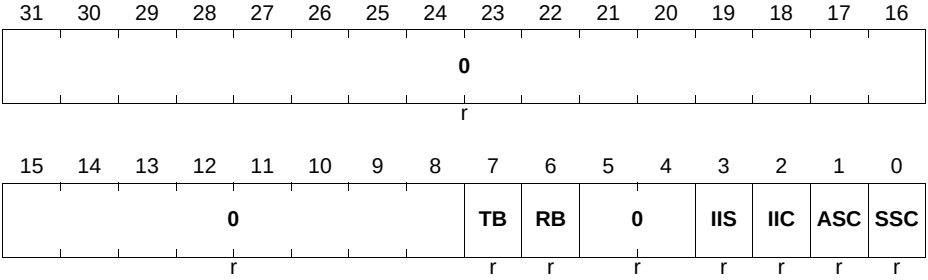
Field	Bits	Type	Description
DLIEN	11	rw	**Data Lost Interrupt Enable** This bit enables the interrupt generation in case of a data lost event (data received in RBUFx while RDVx = 1). 0_B The data lost interrupt is disabled. 1_B The data lost interrupt is enabled. In case of a data lost event, the service request output SRx indicated by INPR.PINP is activated.
TSIEN	12	rw	**Transmit Shift Interrupt Enable** This bit enables the interrupt generation in case of a transmit shift event. 0_B The transmit shift interrupt is disabled. 1_B The transmit shift interrupt is enabled. In case of a transmit shift interrupt event, the service request output SRx indicated by INPR.TSINP is activated.
TBIEN	13	rw	**Transmit Buffer Interrupt Enable** This bit enables the interrupt generation in case of a transmit buffer event. 0_B The transmit buffer interrupt is disabled. 1_B The transmit buffer interrupt is enabled. In case of a transmit buffer event, the service request output SRx indicated by INPR.TBINP is activated.
RIEN	14	rw	**Receive Interrupt Enable** This bit enables the interrupt generation in case of a receive event. 0_B The receive interrupt is disabled. 1_B The receive interrupt is enabled. In case of a receive event, the service request output SRx indicated by INPR.RINP is activated.
AIEN	15	rw	**Alternative Receive Interrupt Enable** This bit enables the interrupt generation in case of a alternative receive event. 0_B The alternative receive interrupt is disabled. 1_B The alternative receive interrupt is enabled. In case of an alternative receive event, the service request output SRx indicated by INPR.AINP is activated.

Field	Bits	Type	Description
BRGIEN	16	rw	**Baud Rate Generator Interrupt Enable** This bit enables the interrupt generation in case of a baud rate generator event. 0_B The baud rate generator interrupt is disabled. 1_B The baud rate generator interrupt is enabled. In case of a baud rate generator event, the service request output SRx indicated by INPR.PINP is activated.
0	[5:4], [31:17]	r	**Reserved** Read as 0; should be written with 0.

14.11.3.2 Channel Configuration Register

The channel configuration register contains indicates the functionality that is available in the USIC channel.

CCFG
Channel Configuration Register (04_H) Reset Value: 0000 00CF$_H$



Field	Bits	Type	Description
SSC	0	r	**SSC Protocol Available** This bit indicates if the SSC protocol is available. 0_B The SSC protocol is not available. 1_B The SSC protocol is available.
ASC	1	r	**ASC Protocol Available** This bit indicates if the ASC protocol is available. 0_B The ASC protocol is not available. 1_B The ASC protocol is available.
IIC	2	r	**IIC Protocol Available** This bit indicates if the IIC functionality is available. 0_B The IIC protocol is not available. 1_B The IIC protocol is available.
IIS	3	r	**IIS Protocol Available** This bit indicates if the IIS protocol is available. 0_B The IIS protocol is not available. 1_B The IIS protocol is available.
RB	6	r	**Receive FIFO Buffer Available** This bit indicates if an additional receive FIFO buffer is available. 0_B A receive FIFO buffer is not available. 1_B A receive FIFO buffer is available.

Field	Bits	Type	Description
TB	7	r	**Transmit FIFO Buffer Available** This bit indicates if an additional transmit FIFO buffer is available. 0_B A transmit FIFO buffer is not available. 1_B A transmit FIFO buffer is available.
0	[5:4], [15:8], [31:16]	r	**Reserved** Read as 0; should be written with 0.

14.11.3.3 Kernel State Configuration Register

The kernel state configuration register KSCFG allows the selection of the desired kernel modes for the different device operating modes.

KSCFG
Kernel State Configuration Register ($0C_H$) **Reset Value: $0000\ 0000_H$**

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				BPSUM	0	SUMCFG		BPNOM	0	NOMCFG		0		BPMODEN	MODEN
r				w	r	rw		w	r	rw		r		w	rw

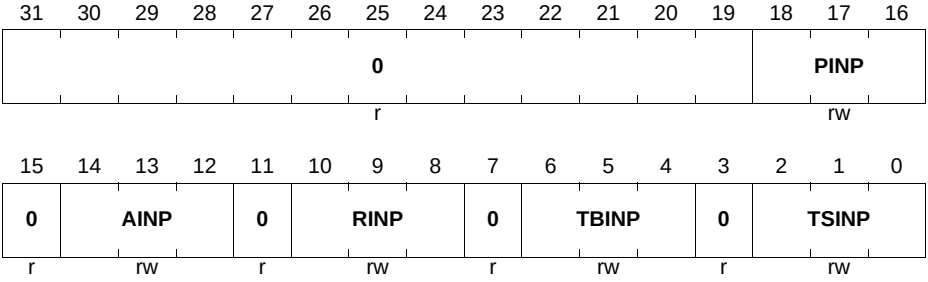
Field	Bits	Type	Description
MODEN	0	rw	**Module Enable** This bit enables the module kernel clock and the module functionality. 0_B The module is switched off immediately (without respecting a stop condition). It does not react on mode control actions and the module clock is switched off. The module does not react on read accesses and ignores write accesses (except to KSCFG). 1_B The module is switched on and can operate. After writing 1 to MODEN, it is recommended to read register KSCFG to avoid pipeline effects in the control block before accessing other USIC registers.
BPMODEN	1	w	**Bit Protection for MODEN** This bit enables the write access to the bit MODEN. It always reads 0. 0_B MODEN is not changed. 1_B MODEN is updated with the written value.
NOMCFG	[5:4]	rw	**Normal Operation Mode Configuration** This bit field defines the kernel mode applied in normal operation mode. 00_B Run mode 0 is selected. 01_B Run mode 1 is selected. 10_B Stop mode 0 is selected. 11_B Stop mode 1 is selected.
BPNOM	7	w	**Bit Protection for NOMCFG** This bit enables the write access to the bit field NOMCFG. It always reads 0. 0_B NOMCFG is not changed. 1_B NOMCFG is updated with the written value.
SUMCFG	[9:8]	rw	**Suspend Mode Configuration** This bit field defines the kernel mode applied in suspend mode. Coding like NOMCFG.

Field	Bits	Type	Description
BPSUM	11	w	**Bit Protection for SUMCFG** This bit enables the write access to the bit field SUMCFG. It always reads 0. 0_B SUMCFG is not changed. 1_B SUMCFG is updated with the written value.
0	[3:2], 6, 10, [31:12]	r	**Reserved** Read as 0; should be written with 0. Bit 2 can read as 1 after BootROM exit (but can be ignored).

14.11.3.4 Interrupt Node Pointer Register

The interrupt node pointer register defines the service request output SRx that is activated if the corresponding event occurs and interrupt generation is enabled.

INPR
Interrupt Node Pointer Register (18_H) Reset Value: $0000\ 0000_H$



Field	Bits	Type	Description
TSINP	[2:0]	rw	**Transmit Shift Interrupt Node Pointer** This bit field defines which service request output SRx becomes activated in case of a transmit shift interrupt. 000_B Output SR0 becomes activated. 001_B Output SR1 becomes activated. 010_B Output SR2 becomes activated. 011_B Output SR3 becomes activated. 100_B Output SR4 becomes activated. 101_B Output SR5 becomes activated. *Note: All other settings of the bit field are reserved.*
TBINP	[6:4]	rw	**Transmit Buffer Interrupt Node Pointer** This bit field defines which service request output SRx will be activated in case of a transmit buffer interrupt or a receive start interrupt. Coding like TSINP.
RINP	[10:8]	rw	**Receive Interrupt Node Pointer** This bit field defines which service request output SRx will be activated in case of a receive interrupt. Coding like TSINP.

Field	Bits	Type	Description
AINP	[14:12]	rw	**Alternative Receive Interrupt Node Pointer** This bit field defines which service request output SRx will be activated in case of a alternative receive interrupt. Coding like TSINP.
PINP	[18:16]	rw	**Protocol Interrupt Node Pointer** This bit field defines which service request output SRx becomes activated in case of a protocol interrupt. Coding like TSINP.
0	3, 7, 11, 15, [31:19]	r	**Reserved** Read as 0; should be written with 0.

14.11.4 Protocol Related Registers

14.11.4.1 Protocol Control Registers

The bits in the protocol control register define protocol-specific functions. They have to be configured by software before enabling a new protocol. Only the bits used for the selected protocol are taken into account, whereas the other bit positions always read as 0. The protocol-specific meaning is described in the related protocol section.

PCR
Protocol Control Register **($3C_H$)** **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CTR 31	**CTR 30**	**CTR 29**	**CTR 28**	**CTR 27**	**CTR 26**	**CTR 25**	**CTR 24**	**CTR 23**	**CTR 22**	**CTR 21**	**CTR 20**	**CTR 19**	**CTR 18**	**CTR 17**	**CTR 16**
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
CTR 15	**CTR 14**	**CTR 13**	**CTR 12**	**CTR 11**	**CTR 10**	**CTR 9**	**CTR 8**	**CTR 7**	**CTR 6**	**CTR 5**	**CTR 4**	**CTR 3**	**CTR 2**	**CTR 1**	**CTR 0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
CTRx (x = 0-31)	x	rw	**Protocol Control Bit x** This bit is a protocol control bit.

14.11.4.2 Protocol Status Register

The flags in the protocol status register can be cleared by writing a 1 to the corresponding bit position in register PSCR. Writing a 1 to a bit position in PSR sets the corresponding flag, but does not lead to further actions (no interrupt generation). Writing a 0 has no effect. These flags should be cleared by software before enabling a new protocol. The protocol-specific meaning is described in the related protocol section.

PSR
Protocol Status Register **(48_H)** **Reset Value: $0000\ 0000_H$**

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AIF	RIF	TBIF	TSIF	DLIF	RSIF	ST9	ST8	ST7	ST6	ST5	ST4	ST3	ST2	ST1	ST0
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
STx (x = 0-9)	x	rwh	**Protocol Status Flag x** See protocol specific description.
RSIF	10	rwh	**Receiver Start Indication Flag** 0_B A receiver start event has not occurred. 1_B A receiver start event has occurred.
DLIF	11	rwh	**Data Lost Indication Flag** 0_B A data lost event has not occurred. 1_B A data lost event has occurred.
TSIF	12	rwh	**Transmit Shift Indication Flag** 0_B A transmit shift event has not occurred. 1_B A transmit shift event has occurred.
TBIF	13	rwh	**Transmit Buffer Indication Flag** 0_B A transmit buffer event has not occurred. 1_B A transmit buffer event has occurred.
RIF	14	rwh	**Receive Indication Flag** 0_B A receive event has not occurred. 1_B A receive event has occurred.

Field	Bits	Type	Description
AIF	15	rwh	**Alternative Receive Indication Flag** 0_B An alternative receive event has not occurred. 1_B An alternative receive event has occurred.
BRGIF	16	rwh	**Baud Rate Generator Indication Flag** 0_B A baud rate generator event has not occurred. 1_B A baud rate generator event has occurred.
0	[31:17]	r	**Reserved;** read as 0; should be written with 0;

14.11.4.3 Protocol Status Clear Register

Read accesses to this register always deliver 0 at all bit positions.

PSCR
Protocol Status Clear Register **($4C_H$)** **Reset Value: 0000 0000$_H$**

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CAIF	CRIF	CTBIF	CTSIF	CDLIF	CRSIF	CST9	CST8	CST7	CST6	CST5	CST4	CST3	CST2	CST1	CST0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
CSTx (x = 0-9)	x	w	**Clear Status Flag x in PSR** 0_B No action 1_B Flag PSR.STx is cleared.
CRSIF	10	w	**Clear Receiver Start Indication Flag** 0_B No action 1_B Flag PSR.RSIF is cleared.
CDLIF	11	w	**Clear Data Lost Indication Flag** 0_B No action 1_B Flag PSR.DLIF is cleared.

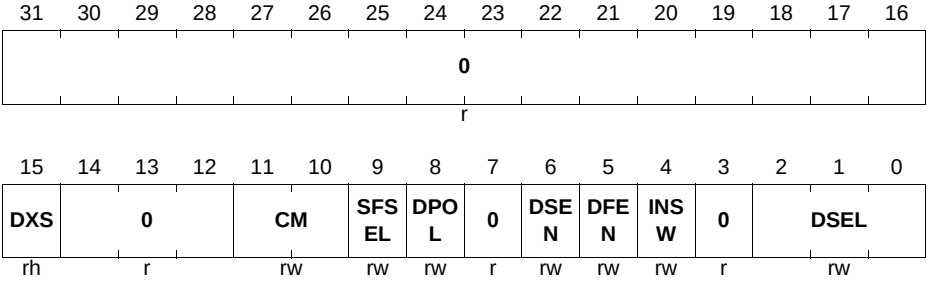
Field	Bits	Type	Description
CTSIF	12	w	**Clear Transmit Shift Indication Flag** 0_B No action 1_B Flag PSR.TSIF is cleared.
CTBIF	13	w	**Clear Transmit Buffer Indication Flag** 0_B No action 1_B Flag PSR.TBIF is cleared.
CRIF	14	w	**Clear Receive Indication Flag** 0_B No action 1_B Flag PSR.RIF is cleared.
CAIF	15	w	**Clear Alternative Receive Indication Flag** 0_B No action 1_B Flag PSR.AIF is cleared.
CBRGIF	16	w	**Clear Baud Rate Generator Indication Flag** 0_B No action 1_B Flag PSR.BRGIF is cleared.
0	[31:17]	r	**Reserved;** read as 0; should be written with 0;

14.11.5 Input Stage Register

14.11.5.1 Input Control Registers

The input control registers contain the bits to define the characteristics of the input stages (input stage DX0 is controlled by register DX0CR, etc.).

DX0CR
Input Control Register 0 **($1C_H$)** **Reset Value: $0000\ 0000_H$**
DX2CR
Input Control Register 2 **(24_H)** **Reset Value: $0000\ 0000_H$**
DX3CR
Input Control Register 3 **(28_H)** **Reset Value: $0000\ 0000_H$**
DX4CR
Input Control Register 4 **($2C_H$)** **Reset Value: $0000\ 0000_H$**
DX5CR
Input Control Register 5 **(30_H)** **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
DSEL	[2:0]	rw	**Data Selection for Input Signal** This bit field defines the input data signal for the corresponding input line for protocol pre-processor. The selection can be made from the input vector DXn[G:A]. 000_B The data input DXnA is selected. 001_B The data input DXnB is selected. 010_B The data input DXnC is selected. 011_B The data input DXnD is selected. 100_B The data input DXnE is selected. 101_B The data input DXnF is selected. 110_B The data input DXnG is selected. 111_B The data input is always 1.

Field	Bits	Type	Description
INSW	4	rw	**Input Switch** This bit defines if the data shift unit input is derived from the input data path DXn or from the selected protocol pre-processors. 0_B The input of the data shift unit is controlled by the protocol pre-processor. 1_B The input of the data shift unit is connected to the selected data input line. This setting is used if the signals are directly derived from an input pin without treatment by the protocol pre-processor.
DFEN	5	rw	**Digital Filter Enable** This bit enables/disables the digital filter for signal DXnS. 0_B The input signal is not digitally filtered. 1_B The input signal is digitally filtered.
DSEN	6	rw	**Data Synchronization Enable** This bit selects if the asynchronous input signal or the synchronized (and optionally filtered) signal DXnS can be used as input for the data shift unit. 0_B The un-synchronized signal can be taken as input for the data shift unit. 1_B The synchronized signal can be taken as input for the data shift unit.
DPOL	8	rw	**Data Polarity for DXn** This bit defines the signal polarity of the input signal. 0_B The input signal is not inverted. 1_B The input signal is inverted.
SFSEL	9	rw	**Sampling Frequency Selection** This bit defines the sampling frequency of the digital filter for the synchronized signal DXnS. 0_B The sampling frequency is f_{PB}. 1_B The sampling frequency is f_{FD}.

Field	Bits	Type	Description
CM	[11:10]	rw	**Combination Mode** This bit field selects which edge of the synchronized (and optionally filtered) signal DXnS actives the trigger output DXnT of the input stage. 00_B The trigger activation is disabled. 01_B A rising edge activates DXnT. 10_B A falling edge activates DXnT. 11_B Both edges activate DXnT.
DXS	15	rh	**Synchronized Data Value** This bit indicates the value of the synchronized (and optionally filtered) input signal. 0_B The current value of DXnS is 0. 1_B The current value of DXnS is 1.
0	3, 7, [14:12], [31:16]	r	**Reserved** Read as 0; should be written with 0.

DX1CR
Input Control Register 1 **(20_H)** **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
DXS	**0**			**CM**		**SFSEL**	**DPOL**	**0**	**DSEN**	**DFEN**	**INSW**	**DCEN**	**DSEL**		
rh	r			rw		rw	rw	r	rw	rw	rw	rw	rw		

Field	Bits	Type	Description
DSEL	[2:0]	rw	**Data Selection for Input Signal** This bit field defines the input data signal for the corresponding input line for protocol pre-processor. The selection can be made from the input vector DX1[G:A]. 000_B The data input DX1A is selected. 001_B The data input DX1B is selected. 010_B The data input DX1C is selected. 011_B The data input DX1D is selected. 100_B The data input DX1E is selected. 101_B The data input DX1F is selected. 110_B The data input DX1G is selected. 111_B The data input is always 1.
DCEN	3	rw	**Delay Compensation Enable** This bit selects if the receive shift clock is controlled by INSW or derived from the input data path DX1. 0_B The receive shift clock is dependent on INSW selection. 1_B The receive shift clock is connected to the selected data input line. This setting is used if delay compensation is required in SSC and IIS protocols, else DCEN should always be 0.
INSW	4	rw	**Input Switch** This bit defines if the data shift unit input is derived from the input data path DX1 or from the selected protocol pre-processors. 0_B The input of the data shift unit is controlled by the protocol pre-processor. 1_B The input of the data shift unit is connected to the selected data input line. This setting is used if the signals are directly derived from an input pin without treatment by the protocol pre-processor.
DFEN	5	rw	**Digital Filter Enable** This bit enables/disables the digital filter for signal DX1S. 0_B The input signal is not digitally filtered. 1_B The input signal is digitally filtered.

Field	Bits	Type	Description
DSEN	6	rw	**Data Synchronization Enable** This bit selects if the asynchronous input signal or the synchronized (and optionally filtered) signal DX1S can be used as input for the data shift unit. 0_B The un-synchronized signal can be taken as input for the data shift unit. 1_B The synchronized signal can be taken as input for the data shift unit.
DPOL	8	rw	**Data Polarity for DXn** This bit defines the signal polarity of the input signal. 0_B The input signal is not inverted. 1_B The input signal is inverted.
SFSEL	9	rw	**Sampling Frequency Selection** This bit defines the sampling frequency of the digital filter for the synchronized signal DX1S. 0_B The sampling frequency is f_{PB}. 1_B The sampling frequency is f_{FD}.
CM	[11:10]	rw	**Combination Mode** This bit field selects which edge of the synchronized (and optionally filtered) signal DX1S actives the trigger output DX1T of the input stage. 00_B The trigger activation is disabled. 01_B A rising edge activates DX1T. 10_B A falling edge activates DX1T. 11_B Both edges activate DX1T.
DXS	15	rh	**Synchronized Data Value** This bit indicates the value of the synchronized (and optionally filtered) input signal. 0_B The current value of DX1S is 0. 1_B The current value of DX1S is 1.
0	7, [14:12], [31:16]	r	**Reserved** Read as 0; should be written with 0.

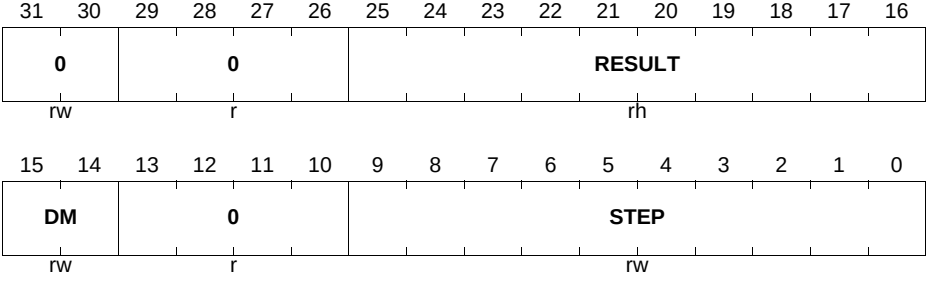
14.11.6 Baud Rate Generator Registers

14.11.6.1 Fractional Divider Register

The fractional divider register FDR allows the generation of the internal frequency f_{FD}, that is derived from the system clock f_{PB}.

FDR can be written only with a privilege mode access.

FDR
Fractional Divider Register (10_H) Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
STEP	[9:0]	rw	**Step Value** In normal divider mode STEP contains the reload value for RESULT after RESULT has reached 3FF$_H$. In fractional divider mode STEP defines the value added to RESULT with each input clock cycle.
DM	[15:14]	rw	**Divider Mode** This bit fields defines the functionality of the fractional divider block. 00_B The divider is switched off, f_{FD} = 0. 01_B Normal divider mode selected. 10_B Fractional divider mode selected. 11_B The divider is switched off, f_{FD} = 0.

Field	Bits	Type	Description
RESULT	[25:16]	rh	**Result Value** In normal divider mode this bit field is updated with f_{PB} according to: RESULT = RESULT + 1 In fractional divider mode this bit field is updated with f_{PB} according to: RESULT = RESULT + STEP If bit field DM is written with 01_B or 10_B, RESULT is loaded with a start value of $3FF_H$.
0	[31:30]	rw	**Reserved for Future Use** Must be written with 0 to allow correct fractional divider operation.
0	[13:10], [29:26]	r	**Reserved** Read as 0; should be written with 0.

14.11.6.2 Baud Rate Generator Register

The protocol-related counters for baud rate generation and timing measurement are controlled by the register BRG.

FDR can be written only with a privilege mode access.

BRG
Baud Rate Generator Register (14_H) Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SCLKCFG		MCLKCFG	SCLKOSEL	0		PDIV									
rw		rw	rw	r		rw									

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	DCTQ					PCTQ		CTQSEL		0	PPPEN	TMEN	0	CLKSEL	
r	rw					rw		rw		r	rw	rw	r	rw	

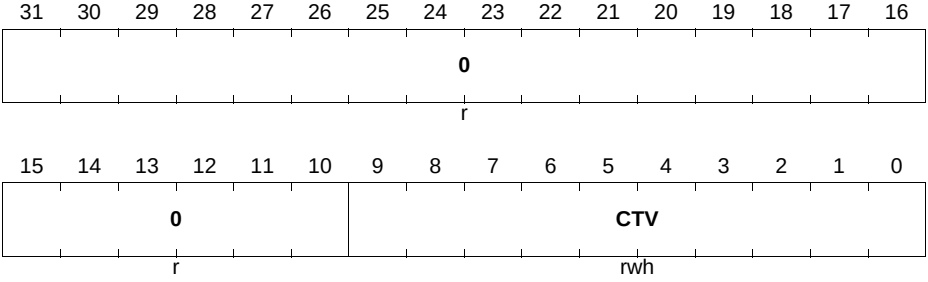
Field	Bits	Type	Description
CLKSEL	[1:0]	rw	**Clock Selection** This bit field defines the input frequency f_{PIN} 00_B The fractional divider frequency f_{FD} is selected. 01_B Reserved, no action 10_B The trigger signal DX1T defines f_{PIN}. Signal MCLK toggles with f_{PIN}. 11_B Signal MCLK corresponds to the DX1S signal and the frequency f_{PIN} is derived from the rising edges of DX1S.
TMEN	3	rw	**Timing Measurement Enable** This bit enables the timing measurement of the capture mode timer. 0_B Timing measurement is disabled: The trigger signals DX0T and DX1T are ignored. 1_B Timing measurement is enabled: The 10-bit counter is incremented by 1 with f_{PPP} and stops counting when reaching its maximum value. If one of the trigger signals DX0T or DX1T become active, the counter value is captured into bit field CTV, the counter is cleared and a transmit shift event is generated.
PPPEN	4	rw	**Enable 2:1 Divider for f_{PPP}** This bit defines the input frequency f_{PPP}. 0_B The 2:1 divider for f_{PPP} is disabled. $f_{PPP} = f_{PIN}$ 1_B The 2:1 divider for f_{PPP} is enabled. $f_{PPP} = f_{MCLK} = f_{PIN} / 2$.
CTQSEL	[7:6]	rw	**Input Selection for CTQ** This bit defines the length of a time quantum for the protocol pre-processor. 00_B $f_{CTQIN} = f_{PDIV}$ 01_B $f_{CTQIN} = f_{PPP}$ 10_B $f_{CTQIN} = f_{SCLK}$ 11_B $f_{CTQIN} = f_{MCLK}$

Field	Bits	Type	Description
PCTQ	[9:8]	rw	**Pre-Divider for Time Quanta Counter** This bit field defines length of a time quantum tq for the time quanta counter in the protocol pre-processor. t_Q = (PCTQ + 1) / f_{CTQIN}
DCTQ	[14:10]	rw	**Denominator for Time Quanta Counter** This bit field defines the number of time quanta t_q taken into account by the time quanta counter in the protocol pre-processor.
PDIV	[25:16]	rw	**Divider Mode: Divider Factor to Generate f_{PDIV}** This bit field defines the ratio between the input frequency f_{PPP} and the divider frequency f_{PDIV}.
SCLKOSEL	28	rw	**Shift Clock Output Select** This bit field selects the input source for the SCLKOUT signal. 0_B SCLK from the baud rate generator is selected as the SCLKOUT input source. 1_B The transmit shift clock from DX1 input stage is selected as the SCLKOUT input source. *Note: The setting SCLKOSEL = 1 is used only when complete closed loop delay compensation is required for a slave SSC/IIS. The default setting of SCLKOSEL = 0 should be always used for all other cases.*
MCLKCFG	29	rw	**Master Clock Configuration** This bit field defines the level of the passive phase of the MCLKOUT signal. 0_B The passive level is 0. 1_B The passive level is 1.
SCLKCFG	[31:30]	rw	**Shift Clock Output Configuration** This bit field defines the level of the passive phase of the SCLKOUT signal and enables/disables a delay of half of a SCLK period. 00_B The passive level is 0 and the delay is disabled. 01_B The passive level is 1 and the delay is disabled. 10_B The passive level is 0 and the delay is enabled. 11_B The passive level is 1 and the delay is enabled.
0	2, 5, 15, [27:26]	r	**Reserved** Read as 0; should be written with 0.

14.11.6.3 Capture Mode Timer Register

The captured timer value is provided by the register CMTR.

CMTR
Capture Mode Timer Register **(44_H)** **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
CTV	[9:0]	rwh	**Captured Timer Value** The value of the counter is captured into this bit field if one of the trigger signals DX0T or DX1T are activated by the corresponding input stage.
0	[31:10]	r	**Reserved** Read as 0; should be written with 0.

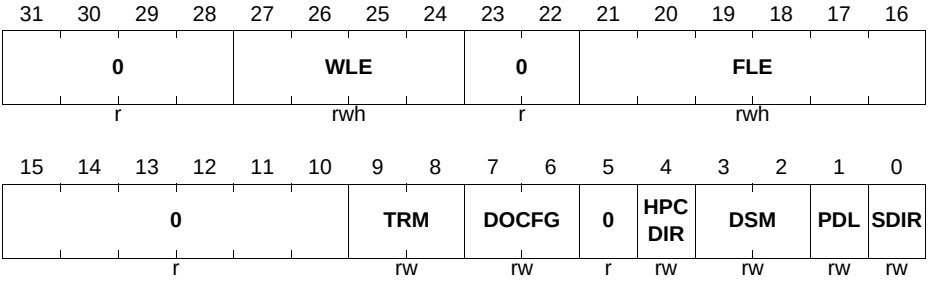
14.11.7 Transfer Control and Status Registers

14.11.7.1 Shift Control Register

The data shift unit is controlled by the register SCTR. The values in this register are applied for data transmission and reception.

Please note that the shift control settings SDIR, WLE, FLE, DSM and HPCDIR are shared between transmitter and receiver. They are internally “frozen” for a each data word transfer in the transmitter with the first transmit shift clock edge and with the first receive shift clock edge in the receiver. The software has to take care that updates of these bit fields by software are done coherently (e.g. refer to the receiver start event indication PSR.RSIF).

SCTR
Shift Control Register **(34_H)** **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
SDIR	0	rw	**Shift Direction** This bit defines the shift direction of the data words for transmission and reception. 0_B Shift LSB first. The first data bit of a data word is located at bit position 0. 1_B Shift MSB first. The first data bit of a data word is located at the bit position given by bit field SCTR.WLE.
PDL	1	rw	**Passive Data Level** This bit defines the output level at the shift data output signal when no data is available for transmission. The PDL level is output with the first relevant transmit shift clock edge of a data word. 0_B The passive data level is 0. 1_B The passive data level is 1.

Field	Bits	Type	Description
DSM	[3:2]	rw	**Data Shift Mode** This bit field describes how the receive and transmit data is shifted in and out. 00_B Receive and transmit data is shifted in and out one bit at a time through DX0 and DOUT0. 01_B Reserved. 10_B Receive and transmit data is shifted in and out two bits at a time through two input stages (DX0 and DX3) and DOUT[1:0] respectively. 11_B Receive and transmit data is shifted in and out four bits at a time through four input stages (DX0, DX[5:3]) and DOUT[3:0] respectively. *Note: Dual- and Quad-output modes are used only by the SSC protocol. For all other protocols DSM must always be written with 00_B.*
HPCDIR	4	rw	**Port Control Direction** This bit defines the direction of the port pin(s) which allows hardware pin control (CCR.PCEN = 1). 0_B The pin(s) with hardware pin control enabled are selected to be in input mode. 1_B The pin(s) with hardware pin control enabled are selected to be in output mode.
DOCFG	[7:6]	rw	**Data Output Configuration** This bit defines the relation between the internal shift data value and the data output signal DOUTx. $X0_B$ DOUTx = shift data value $X1_B$ DOUTx = inverted shift data value

Field	Bits	Type	Description
TRM	[9:8]	rw	**Transmission Mode** This bit field describes how the shift control signal is interpreted by the DSU. Data transfers are only possible while the shift control signal is active. 00_B The shift control signal is considered as inactive and data frame transfers are not possible. 01_B The shift control signal is considered active if it is at 1-level. This is the setting to be programmed to allow data transfers. 10_B The shift control signal is considered active if it is at 0-level. It is recommended to avoid this setting and to use the inversion in the DX2 stage in case of a low-active signal. 11_B The shift control signal is considered active without referring to the actual signal level. Data frame transfer is possible after each edge of the signal.
FLE	[21:16]	rwh	**Frame Length** This bit field defines how many bits are transferred within a data frame. A data frame can consist of several concatenated data words. If TCSR.FLEMD = 1, the value can be updated automatically by the data handler.
WLE	[27:24]	rwh	**Word Length** This bit field defines the data word length (amount of bits that are transferred in each data word) for reception and transmission. The data word is always right-aligned in the data buffer at the bit positions [WLE down to 0]. If TCSR.WLEMD = 1, the value can be updated automatically by the data handler. 0_H The data word contains 1 data bit located at bit position 0. 1_H The data word contains 2 data bits located at bit positions [1:0]. ... E_H The data word contains 15 data bits located at bit positions [14:0]. F_H The data word contains 16 data bits located at bit positions [15:0].

Field	Bits	Type	Description
0	5, [15:10], [23:22], [31:28]	r	**Reserved** Read as 0; should be written with 0.

14.11.7.2 Transmission Control and Status Register

The data transmission is controlled and monitored by register TCSR.

TCSR
Transmit Control/Status Register (38$_H$) Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0			**TE**	**TVC**	**TV**	**0**	**TSOF**	**0**							
r			rh	rh	rh	r	rh	r							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		**WA**	**TDVTR**	**TDEN**		**0**	**TDSSM**	**TDV**	**EOF**	**SOF**	**HPCMD**	**WAMD**	**FLEMD**	**SELMD**	**WLEMD**
r		rwh	rw	rw		r	rw	rh	rwh	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
WLEMD	0	rw	**WLE Mode** This bit enables the data handler to automatically update the bit field SCTR.WLE by the transmit control information TCI[3:0] and bit TCSR.EOF by TCI[4] (see **Page 14-32**). If enabled, an automatic update takes place when new data is loaded to register TBUF, either by writing to one of the transmit buffer input locations TBUFx or by an optional data buffer. 0_B The automatic update of SCTR.WLE and TCSR.EOF is disabled. 1_B The automatic update of SCTR.WLE and TCSR.EOF is enabled.

Field	Bits	Type	Description
SELMD	1	rw	**Select Mode** This bit can be used mainly for the SSC protocol. It enables the data handler to automatically update bit field PCR.CTR[20:16] by the transmit control information TCI[4:0] and clear bit field PCR.CTR[23:21] (see **Page 14-32**). If enabled, an automatic update takes place when new data is loaded to register TBUF, either by writing to one of the transmit buffer input locations TBUFx or by an optional data buffer. 0_B The automatic update of PCR.CTR[23:16] is disabled. 1_B The automatic update of PCR.CTR[23:16] is disabled.
FLEMD	2	rw	**FLE Mode** This bit enables the data handler to automatically update bits SCTR.FLE[4:0] by the transmit control information TCI[4:0] and to clear bit SCTR.FLE[5] (see **Page 14-32**). If enabled, an automatic update takes place when new data is loaded to register TBUF, either by writing to one of the transmit buffer input locations TBUFx or by an optional data buffer. 0_B The automatic update of FLE is disabled. 1_B The automatic update of FLE is enabled.
WAMD	3	rw	**WA Mode** This bit can be used mainly for the IIS protocol. It enables the data handler to automatically update bit TCSR.WA by the transmit control information TCI[4] (see **Page 14-32**). If enabled, an automatic update takes place when new data is loaded to register TBUF, either by writing to one of the transmit buffer input locations TBUFx or by an optional data buffer. 0_B The automatic update of bit WA is disabled. 1_B The automatic update of bit WA is enabled.

Field	Bits	Type	Description
HPCMD	4	rw	**Hardware Port Control Mode** This bit can be used mainly for the dual and quad SSC protocol. It enables the data handler to automatically update bit SCTR.DSM by the transmit control information TCI[1:0] and bit SCTR.HPCDIR by TCI[2] (see **Page 14-32**). If enabled, an automatic update takes place when new data is loaded to register TBUF, either by writing to one of the transmit buffer input locations TBUFx or by an optional data buffer. 0_B The automatic update of bits SCTR.DSM and SCTR.HPCDIR is disabled. 1_B The automatic update of bits SCTR.DSM and SCTR.HPCDIR is enabled.
SOF	5	rw	**Start Of Frame** This bit is only taken into account for the SSC protocol, otherwise it is ignored. It indicates that the data word in TBUF is considered as the first word of a new SSC frame if it is valid for transmission (TCSR.TDV = 1). This bit becomes cleared when the TBUF data word is transferred to the transmit shift register. 0_B The data word in TBUF is not considered as first word of a frame. 1_B The data word in TBUF is considered as first word of a frame. A currently running frame is finished and MSLS becomes deactivated (respecting the programmed delays).

Field	Bits	Type	Description
EOF	6	rwh	**End Of Frame** This bit is only taken into account for the SSC protocol, otherwise it is ignored. It can be modified automatically by the data handler if bit WLEMD = 1. It indicates that the data word in TBUF is considered as the last word of an SSC frame. If it is the last word, the MSLS signal becomes inactive after the transfer, respecting the programmed delays. This bit becomes cleared when the TBUF data word is transferred to the transmit shift register. 0_B The data word in TBUF is not considered as last word of an SSC frame. 1_B The data word in TBUF is considered as last word of an SSC frame.
TDV	7	rh	**Transmit Data Valid** This bit indicates that the data word in the transmit buffer TBUF can be considered as valid for transmission. The TBUF data word can only be sent out if TDV = 1. It is automatically set when data is moved to TBUF (by writing to one of the transmit buffer input locations TBUFx, or optionally, by the bypass or FIFO mechanism). 0_B The data word in TBUF is not valid for transmission. 1_B The data word in TBUF is valid for transmission and a transmission start is possible. New data should not be written to a TBUFx input location while TDV = 1.

Field	Bits	Type	Description
TDSSM	8	rw	**TBUF Data Single Shot Mode** This bit defines if the data word TBUF data is considered as permanently valid or if the data should only be transferred once. 0_B The data word in TBUF is not considered as invalid after it has been loaded into the transmit shift register. The loading of the TBUF data into the shift register does not clear TDV. 1_B The data word in TBUF is considered as invalid after it has been loaded into the shift register. In ASC and IIC mode, TDV is cleared with the TBI event, whereas in SSC and IIS mode, it is cleared with the RSI event. TDSSM = 1 has to be programmed if an optional data buffer is used.
TDEN	[11:10]	rw	**TBUF Data Enable** This bit field controls the gating of the transmission start of the data word in the transmit buffer TBUF. 00_B A transmission start of the data word in TBUF is disabled. If a transmission is started, the passive data level is sent out. 01_B A transmission of the data word in TBUF can be started if TDV = 1. 10_B A transmission of the data word in TBUF can be started if TDV = 1 while DX2S = 0. 11_B A transmission of the data word in TBUF can be started if TDV = 1 while DX2S = 1.
TDVTR	12	rw	**TBUF Data Valid Trigger** This bit enables the transfer trigger unit to set bit TCSR.TE if the trigger signal DX2T becomes active for event driven transfer starts, e.g. timer-based or depending on an event at an input pin. Bit TDVTR has to be 0 for protocols where the input stage DX2 is used for data shifting. 0_B Bit TCSR.TE is permanently set. 1_B Bit TCSR.TE is set if DX2T becomes active while TDV = 1.

Field	Bits	Type	Description
WA	13	rwh	**Word Address** This bit is only taken into account for the IIS protocol, otherwise it is ignored. It can be modified automatically by the data handler if bit WAMD = 1. Bit WA defines for which channel the data stored in TBUF will be transmitted. 0_B The data word in TBUF will be transmitted after a falling edge of WA has been detected (referring to PSR.WA). 1_B The data word in TBUF will be transmitted after a rising edge of WA has been detected (referring to PSR.WA).
TSOF	24	rh	**Transmitted Start Of Frame** This bit indicates if the latest start of a data word transmission has taken place for the first data word of a new data frame. This bit is updated with the transmission start of each data word. 0_B The latest data word transmission has not been started for the first word of a data frame. 1_B The latest data word transmission has been started for the first word of a data frame.
TV	26	rh	**Transmission Valid** This bit represents the transmit buffer underflow and indicates if the latest start of a data word transmission has taken place with a valid data word from the transmit buffer TBUF. This bit is updated with the transmission start of each data word. 0_B The latest start of a data word transmission has taken place while no valid data was available. As a result, the transmission of a data words with passive level (SCTR.PDL) has been started. 1_B The latest start of a data word transmission has taken place with valid data from TBUF.

Field	Bits	Type	Description
TVC	27	rh	**Transmission Valid Cumulated** This bit cumulates the transmit buffer underflow indication TV. It is cleared automatically together with bit TV and has to be set by writing FMR.ATVC = 1. 0_B Since TVC has been set, at least one data buffer underflow condition has occurred. 1_B Since TVC has been set, no data buffer underflow condition has occurred.
TE	28	rh	**Trigger Event** If the transfer trigger mechanism is enabled, this bit indicates that a trigger event has been detected (DX2T = 1) while TCSR.TDV = 1. If the event trigger mechanism is disabled, the bit TE is permanently set. It is cleared by writing FMR.MTDV = 10_B or when the data word located in TBUF is loaded into the shift register. 0_B The trigger event has not yet been detected. A transmission of the data word in TBUF can not be started. 1_B The trigger event has been detected (or the trigger mechanism is switched off) and a transmission of the data word in TBUF can be started.
0	9, [23:14], 25, [31:29]	r	**Reserved** Read as 0; should be written with 0.

14.11.7.3 Flag Modification Registers

The flag modification register FMR allows the modification of control and status flags related to data handling by using only write accesses. Read accesses to FMR always deliver 0 at all bit positions.

Additionally, the service request outputs of this USIC channel can be activated by software (the activation is triggered by the write access and is deactivated automatically).

FMR
Flag Modification Register **(68_H)** **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0										SIO5	SIO4	SIO3	SIO2	SIO1	SIO0
r										w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRDV1	CRDV0	0									ATVC	0		MTDV	
w	w	r									w	r		w	

Field	Bits	Type	Description
MTDV	[1:0]	w	**Modify Transmit Data Valid** Writing to this bit field can modify bits TCSR.TDV and TCSR.TE to control the start of a data word transmission by software. 00_B No action. 01_B Bit TDV is set, TE is unchanged. 10_B Bits TDV and TE are cleared. 11_B Reserved
ATVC	4	w	**Activate Bit TVC** Writing to this bit can set bit TCSR.TVC to start a new cumulation of the transmit buffer underflow condition. 0_B No action. 1_B Bit TCSR.TVC is set.
CRDV0	14	w	**Clear Bits RDV for RBUF0** Writing 1 to this bit clears bits RBUF01SR.RDV00 and RBUF01SR.RDV10 to declare the received data in RBUF0 as no longer valid (to emulate a read action). 0_B No action. 1_B Bits RBUF01SR.RDV00 and RBUF01SR.RDV10 are cleared.

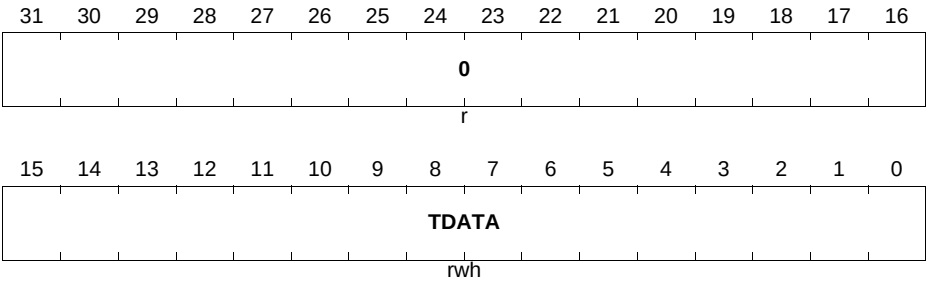
Field	Bits	Type	Description
CRDV1	15	w	**Clear Bit RDV for RBUF1** Writing 1 to this bit clears bits RBUF01SR.RDV01 and RBUF01SR.RDV11 to declare the received data in RBUF1 as no longer valid (to emulate a read action). 0_B No action. 1_B Bits RBUF01SR.RDV01 and RBUF01SR.RDV11 are cleared.
SIO0, **SIO1,** **SIO2,** **SIO3,** **SIO4,** **SIO5**	16, 17, 18, 19, 20, 21	w	**Set Interrupt Output SRx** Writing a 1 to this bit field activates the service request output SRx of this USIC channel. It has no impact on service request outputs of other USIC channels. 0_B No action. 1_B The service request output SRx is activated.
0	[3:2], [13:5], [31:22]	r	**Reserved** Read as 0; should be written with 0.

14.11.8 Data Buffer Registers

14.11.8.1 Transmit Buffer Locations

The 32 independent data input locations TBUF00 to TBUF31 are address locations that can be used as data entry locations for the transmit buffer. Data written to one of these locations will appear in a common register TBUF. Additionally, the 5 bit coding of the number [31:0] of the addressed data input location represents the transmit control information TCI (please refer to the protocol sections for more details).
The internal transmit buffer register TBUF contains the data that will be loaded to the transmit shift register for the next transmission of a data word. It can be read out at all TBUF00 to TBUF31 addresses.

TBUFx (x = 00-31)
Transmit Buffer Input Location x (80_H + x*4) Reset Value: 0000 0000_H

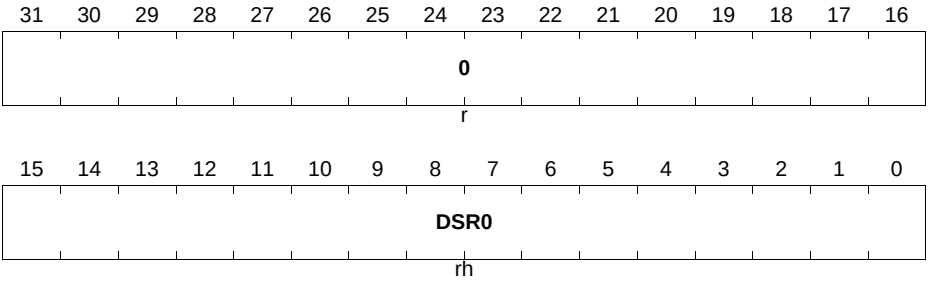


Field	Bits	Type	Description
TDATA	[15:0]	rwh	**Transmit Data** This bit field contains the data to be transmitted (read view). A data write action to at least the low byte of TDATA sets TCSR.TDV.
0	[31:16]	r	**Reserved** Read as 0; should be written with 0.

14.11.8.2 Receive Buffer Registers RBUF0, RBUF1

The receive buffer register RBUF0 contains the data received from RSR0[3:0]. A read action does not change the status of the receive data from “not yet read = valid” to “already read = not valid”.

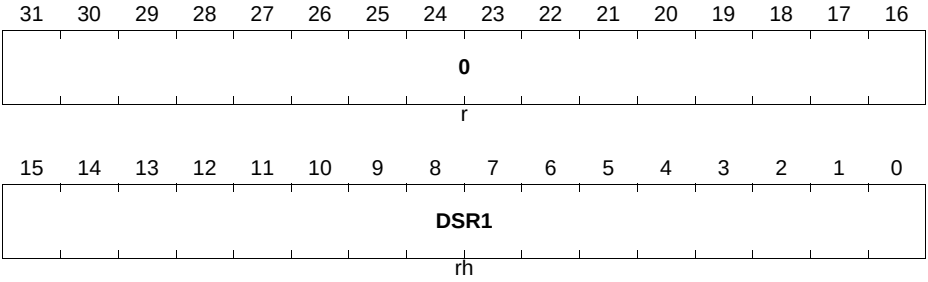
RBUF0
Receiver Buffer Register 0 ($5C_H$) Reset Value: 0000 0000_H



Field	Bits	Type	Description
DSR0	[15:0]	rh	**Data of Shift Registers 0[3:0]**
0	[31:16]	r	**Reserved** Read as 0; should be written with 0.

The receive buffer register RBUF1 contains the data received from RSR1[3:0]. A read action does not change the status of the receive data from “not yet read = valid” to “already read = not valid”.

RBUF1
Receiver Buffer Register 1 **(60_H)** **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
DSR1	[15:0]	rh	**Data of Shift Registers 1[3:0]**
0	[31:16]	r	**Reserved** Read as 0; should be written with 0.

The receive buffer status register RBUF01SR provides the status of the data in receive buffers RBUF0 and RBUF1.

RBUF01SR
Receiver Buffer 01 Status Register (64$_H$) **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DS1	RDV 11	RDV 10	0			PER R1	PAR 1	0	SOF 1	0		WLEN1			
rh	rh	rh	r			rh	rh	r	rh	r		rh			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DS0	RDV 01	RDV 00	0			PER R0	PAR 0	0	SOF 0	0		WLEN0			
rh	rh	rh	r			rh	rh	r	rh	r		rh			

Field	Bits	Type	Description
WLEN0	[3:0]	rh	**Received Data Word Length in RBUF0** This bit field indicates how many bits have been received within the last data word stored in RBUF0. This number indicates how many data bits have to be considered as receive data, whereas the other bits in RBUF0 have been cleared automatically. The received bits are always right-aligned. For all protocol modes besides dual and quad SSC, Received data word length = WLEN0 + 1 For dual SSC mode, Received data word length = WLEN0 + 2 For quad SSC mode, Received data word length = WLEN0 + 4
SOF0	6	rh	**Start of Frame in RBUF0** This bit indicates whether the data word in RBUF0 has been the first data word of a data frame. 0_B The data in RBUF0 has not been the first data word of a data frame. 1_B The data in RBUF0 has been the first data word of a data frame.

Field	Bits	Type	Description
PAR0	8	rh	**Protocol-Related Argument in RBUF0** This bit indicates the value of the protocol-related argument. This value is elaborated depending on the selected protocol and adds additional information to the data word in RBUF0. The meaning of this bit is described in the corresponding protocol chapter.
PERR0	9	rh	**Protocol-related Error in RBUF0** This bit indicates if the value of the protocol-related argument meets an expected value. This value is elaborated depending on the selected protocol and adds additional information to the data word in RBUF0. The meaning of this bit is described in the corresponding protocol chapter. 0_B The received protocol-related argument PAR matches the expected value. The reception of the data word sets bit PSR.RIF and can generate a receive interrupt. 1_B The received protocol-related argument PAR does not match the expected value. The reception of the data word sets bit PSR.AIF and can generate an alternative receive interrupt.
RDV00	13	rh	**Receive Data Valid in RBUF0** This bit indicates the status of the data content of register RBUF0. This bit is identical to bit RBUF01SR.RDV10 and allows consisting reading of information for the receive buffer registers. It is set when a new data word is stored in RBUF0 and automatically cleared if it is read out via RBUF. 0_B Register RBUF0 does not contain data that has not yet been read out. 1_B Register RBUF0 contains data that has not yet been read out.

Field	Bits	Type	Description
RDV01	14	rh	**Receive Data Valid in RBUF1** This bit indicates the status of the data content of register RBUF1. This bit is identical to bit RBUF01SR.RDV11 and allows consisting reading of information for the receive buffer registers. It is set when a new data word is stored in RBUF1 and automatically cleared if it is read out via RBUF. 0_B Register RBUF1 does not contain data that has not yet been read out. 1_B Register RBUF1 contains data that has not yet been read out.
DS0	15	rh	**Data Source** This bit indicates which receive buffer register (RBUF0 or RBUF1) is currently visible in registers RBUF(D) and in RBUFSR for the associated status information. It indicates which buffer contains the oldest data (the data that has been received first). This bit is identical to bit RBUF01SR.DS1 and allows consisting reading of information for the receive buffer registers. 0_B The register RBUF contains the data of RBUF0 (same for associated status information). 1_B The register RBUF contains the data of RBUF1 (same for associated status information).
WLEN1	[19:16]	rh	**Received Data Word Length in RBUF1** This bit field indicates how many bits have been received within the last data word stored in RBUF1. This number indicates how many data bits have to be considered as receive data, whereas the other bits in RBUF1 have been cleared automatically. The received bits are always right-aligned. For all protocol modes besides dual and quad SSC, Received data word length = WLEN1 + 1 For dual SSC mode, Received data word length = WLEN1 + 2 For quad SSC mode, Received data word length = WLEN1 + 4

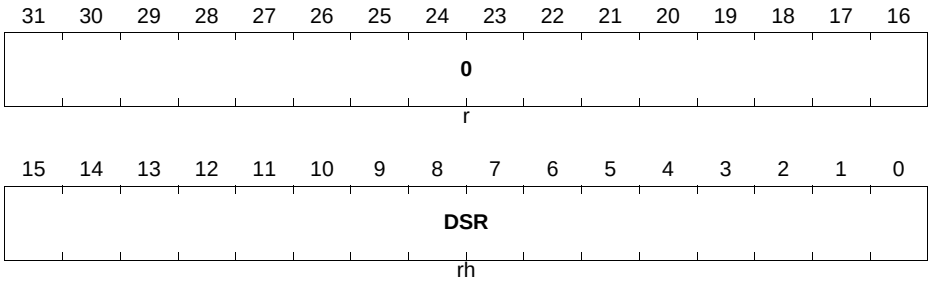
Field	Bits	Type	Description
SOF1	22	rh	**Start of Frame in RBUF1** This bit indicates whether the data word in RBUF1 has been the first data word of a data frame. 0_B The data in RBUF1 has not been the first data word of a data frame. 1_B The data in RBUF1 has been the first data word of a data frame.
PAR1	24	rh	**Protocol-Related Argument in RBUF1** This bit indicates the value of the protocol-related argument. This value is elaborated depending on the selected protocol and adds additional information to the data word in RBUF1. The meaning of this bit is described in the corresponding protocol chapter.
PERR1	25	rh	**Protocol-related Error in RBUF1** This bit indicates if the value of the protocol-related argument meets an expected value. This value is elaborated depending on the selected protocol and adds additional information to the data word in RBUF1. The meaning of this bit is described in the corresponding protocol chapter. 0_B The received protocol-related argument PAR matches the expected value. The reception of the data word sets bit PSR.RIF and can generate a receive interrupt. 1_B The received protocol-related argument PAR does not match the expected value. The reception of the data word sets bit PSR.AIF and can generate an alternative receive interrupt.
RDV10	29	rh	**Receive Data Valid in RBUF0** This bit indicates the status of the data content of register RBUF0. This bit is identical to bit RBUF01SR.RDV00 and allows consisting reading of information for the receive buffer registers. 0_B Register RBUF0 does not contain data that has not yet been read out. 1_B Register RBUF0 contains data that has not yet been read out.

Field	Bits	Type	Description
RDV11	30	rh	**Receive Data Valid in RBUF1** This bit indicates the status of the data content of register RBUF1. This bit is identical to bit RBUF01SR.RDV01 and allows consisting reading of information for the receive buffer registers. 0_B Register RBUF1 does not contain data that has not yet been read out. 1_B Register RBUF1 contains data that has not yet been read out.
DS1	31	rh	**Data Source** This bit indicates which receive buffer register (RBUF0 or RBUF1) is currently visible in registers RBUF(D) and in RBUFSR for the associated status information. It indicates which buffer contains the oldest data (the data that has been received first). This bit is identical to bit RBUF01SR.DS0 and allows consisting reading of information for the receive buffer registers. 0_B The register RBUF contains the data of RBUF0 (same for associated status information). 1_B The register RBUF contains the data of RBUF1 (same for associated status information).
0	[5:4], 7, [12:10], [21:20], 23, [28:26]	r	**Reserved** Read as 0; should be written with 0.

14.11.8.3 Receive Buffer Registers RBUF, RBUFD, RBUFSR

The receiver buffer register RBUF shows the content of the either RBUF0 or RBUF1, depending on the order of reception. Always the oldest data (the data word that has been received first) from both receive buffers can be read from RBUF. It is recommended to read out the received data from RBUF instead of RBUF0/1. With a read access of at least the low byte of RBUF, the status of the receive data is automatically changed from “not yet read = valid” to “already read = not valid”, the content of RBUF becomes updated, and the next received data word becomes visible in RBUF.

RBUF
Receiver Buffer Register **(54_H)** **Reset Value: 0000 0000$_H$**

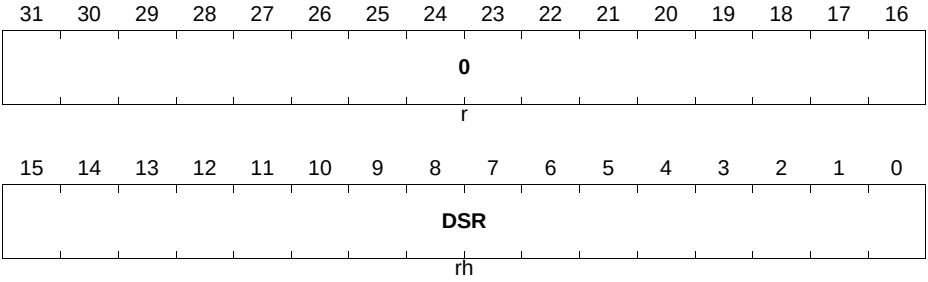


Field	Bits	Type	Description
DSR	[15:0]	rh	**Received Data** This bit field monitors the content of either RBUF0 or RBUF1, depending on the reception sequence.
0	[31:16]	r	**Reserved** Read as 0; should be written with 0.

If a debugger should be used to monitor the received data, the automatic update mechanism has to be de-activated to guaranty data consistency. Therefore, the receiver buffer register for debugging RBUFD is available. It is similar to RBUF, but without the automatic update mechanism by a read action. So a debugger (or other monitoring function) can read RBUFD without disturbing the receive sequence.

RBUFD

Receiver Buffer Register for Debugger(58_H) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
DSR	[15:0]	rh	**Data from Shift Register** Same as RBUF.DSR, but without releasing the buffer after a read action.
0	[31:16]	r	**Reserved** Read as 0; should be written with 0.

The receive buffer status register RBUFSR provides the status of the data in receive buffers RBUF and RBUFD. If bits RBUF01SR.DS0 (or RBUF01SR.DS1) are 0, the lower 16-bit content of RBUF01SR is monitored in RBUFSR, otherwise the upper 16-bit content of RBUF01SR is shown.

RBUFSR
Receiver Buffer Status Register **(50_H)** **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
DS	RDV1	RDV0	0			PERR	PAR	0	SOF	0		WLEN			
rh	rh	rh	r			rh	rh	r	rh	r		rh			

Field	Bits	Type	Description
WLEN	[3:0]	rh	**Received Data Word Length in RBUF or RBUFD** Description see RBUF01SR.WLEN0 or RBUF01SR.WLEN1.
SOF	6	rh	**Start of Frame in RBUF or RBUFD** Description see RBUF01SR.SOF0 or RBUF01SR.SOF1.
PAR	8	rh	**Protocol-Related Argument in RBUF or RBUFD** Description see RBUF01SR.PAR0 or RBUF01SR.PAR1.
PERR	9	rh	**Protocol-related Error in RBUF or RBUFD** Description see RBUF01SR.PERR0 or RBUF01SR.PERR1.
RDV0	13	rh	**Receive Data Valid in RBUF or RBUFD** Description see RBUF01SR.RDV00 or RBUF01SR.RDV10.
RDV1	14	rh	**Receive Data Valid in RBUF or RBUFD** Description see RBUF01SR.RDV01 or RBUF01SR.RDV11.

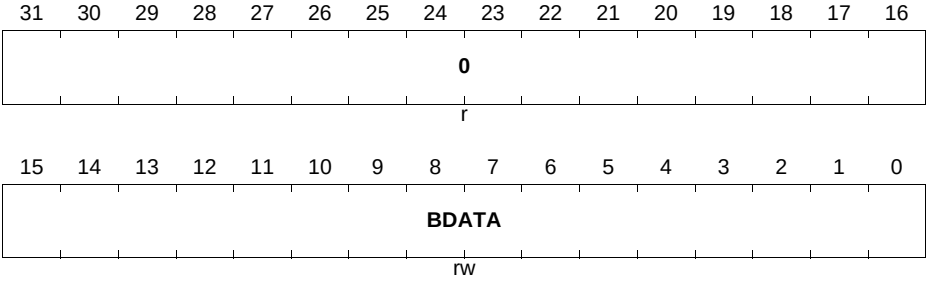
Field	Bits	Type	Description
DS	15	rh	**Data Source of RBUF or RBUFD** Description see RBUF01SR.DS0 or RBUF01SR.DS1.
0	[5:4], 7, [12:10], [31:16]	r	**Reserved** Read as 0; should be written with 0.

14.11.9 FIFO Buffer and Bypass Registers

14.11.9.1 Bypass Registers

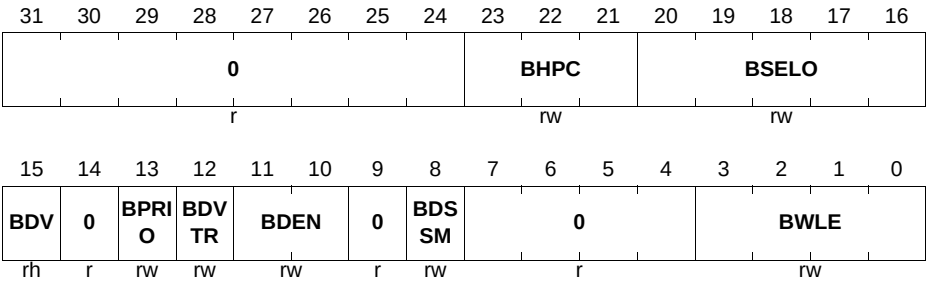
A write action to at least the low byte of the bypass data register sets BYPCR.BDV = 1 (bypass data tagged valid).

BYP
Bypass Data Register **(100_H)** **Reset Value: 0000 0000_H**



Bit (Field)	Width	Type	Description
BDATA	[15:0]	rw	**Bypass Data** This bit field contains the bypass data.
0	[31:16]	r	**Reserved** Read as 0; should be written with 0.

BYPCR
Bypass Control Register **(104_H)** **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
BWLE	[3:0]	rw	**Bypass Word Length** This bit field defines the word length of the bypass data. The word length is given by BWLE + 1 with the data word being right-aligned in the data buffer at the bit positions [BWLE down to 0]. The bypass data word is always considered as an own frame with the length of BWLE. Same coding as SCTR.WLE.
BDSSM	8	rw	**Bypass Data Single Shot Mode** This bit defines if the bypass data is considered as permanently valid or if the bypass data is only transferred once (single shot mode). 0_B The bypass data is still considered as valid after it has been loaded into TBUF. The loading of the data into TBUF does not clear BDV. 1_B The bypass data is considered as invalid after it has been loaded into TBUF. The loading of the data into TBUF clears BDV.

Field	Bits	Type	Description
BDEN	[11:10]	rw	**Bypass Data Enable** This bit field defines if and how the transfer of bypass data to TBUF is enabled. 00_B The transfer of bypass data is disabled. 01_B The transfer of bypass data to TBUF is possible. Bypass data will be transferred to TBUF according to its priority if BDV = 1. 10_B Gated bypass data transfer is enabled. Bypass data will be transferred to TBUF according to its priority if BDV = 1 and while DX2S = 0. 11_B Gated bypass data transfer is enabled. Bypass data will be transferred to TBUF according to its priority if BDV = 1 and while DX2S = 1.
BDVTR	12	rw	**Bypass Data Valid Trigger** This bit enables the bypass data for being tagged valid when DX2T is active (for time framing or time-out purposes). 0_B Bit BDV is not influenced by DX2T. 1_B Bit BDV is set if DX2T is active.
BPRIO	13	rw	**Bypass Priority** This bit defines the priority between the bypass data and the transmit FIFO data. 0_B The transmit FIFO data has a higher priority than the bypass data. 1_B The bypass data has a higher priority than the transmit FIFO data.
BDV	15	rh	**Bypass Data Valid** This bit defines if the bypass data is valid for a transfer to TBUF. This bit is set automatically by a write access to at least the low-byte of register BYP. It can be cleared by software by writing TRBSCR.CBDV. 0_B The bypass data is not valid. 1_B The bypass data is valid.
BSELO	[20:16]	rw	**Bypass Select Outputs** This bit field contains the value that is written to PCR.CTR[20:16] if bypass data is transferred to TBUF while TCSR.SELMD = 1. In the SSC protocol, this bit field can be used to define which SELOx output line will be activated when bypass data is transmitted.

Field	Bits	Type	Description
BHPC	[23:21]	rw	**Bypass Hardware Port Control** This bit field contains the value that is written to SCTR[4:2] if bypass data is transferred to TBUF while TCSR.HPCMD = 1. In the SSC protocol, this bit field can be used to define the data shift mode and if hardware port control is enabled through CCR.HPCEN = 1, the pin direction when bypass data is transmitted.
0	[7:4], 9, 14, [31:24]	r	**Reserved** Read as 0; should be written with 0.

14.11.9.2 General FIFO Buffer Control Registers

The transmit and receive FIFO status information of USICx_CHy is given in registers USICx_CHy.TRBSR.

The bits related to the transmitter buffer in this register can only by written if the transmit buffer functionality is enabled by CCFG.TB = 1, otherwise write accesses are ignored. A similar behavior applies for the bits related to the receive buffer referring to CCFG.RB = 1.

The interrupt flags (event flags) in the transmit and receive FIFO status register TRBSR can be cleared by writing a 1 to the corresponding bit position in register TRBSCR, whereas writing a 0 has to effect on these bits. Writing a 1 by software to SRBI, RBERI, ARBI, STBI, or TBERI sets the corresponding bit to simulate the detection of a transmit/receive buffer event, but without activating any service request output (therefore, see FMR.SIOx).

Bits TBUS and RBUS have been implemented for testing purposes. They can be ignored by data handling software. Please note that a read action can deliver either a 0 or a 1 for these bits. It is recommended to treat them as “don’t care”.

TRBSR
Transmit/Receive Buffer Status Register

(114_H) **Reset Value: $0000\ 0808_H$**

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	STBT	TBUS	TFULL	TEMPTY	0	TBERI	STBI	0	SRBT	RBUS	RFULL	REMPTY	ARBI	RBERI	SRBI
r	rh	rh	rh	rh	r	rwh	rwh	r	rh	rh	rh	rh	rwh	rwh	rwh

Field	Bits	Type	Description
SRBI	0	rwh	**Standard Receive Buffer Event** This bit indicates that a standard receive buffer event has been detected. It is cleared by writing TRBSCR.CSRBI = 1. If enabled by RBCTR.SRBIEN, the service request output SRx selected by RBCTR.SRBINP becomes activated if a standard receive buffer event is detected. 0_B A standard receive buffer event has not been detected. 1_B A standard receive buffer event has been detected.
RBERI	1	rwh	**Receive Buffer Error Event** This bit indicates that a receive buffer error event has been detected. It is cleared by writing TRBSCR.CRBERI = 1. If enabled by RBCTR.RBERIEN, the service request output SRx selected by RBCTR.ARBINP becomes activated if a receive buffer error event is detected. 0_B A receive buffer error event has not been detected. 1_B A receive buffer error event has been detected.

Field	Bits	Type	Description
ARBI	2	rwh	**Alternative Receive Buffer Event** This bit indicates that an alternative receive buffer event has been detected. It is cleared by writing TRBSCR.CARBI = 1. If enabled by RBCTR.ARBIEN, the service request output SRx selected by RBCTR.ARBINP becomes activated if an alternative receive buffer event is detected. 0_B An alternative receive buffer event has not been detected. 1_B An alternative receive buffer event has been detected.
REMPTY	3	rh	**Receive Buffer Empty** This bit indicates whether the receive buffer is empty. 0_B The receive buffer is not empty. 1_B The receive buffer is empty.
RFULL	4	rh	**Receive Buffer Full** This bit indicates whether the receive buffer is full. 0_B The receive buffer is not full. 1_B The receive buffer is full.
RBUS	5	rh	**Receive Buffer Busy** This bit indicates whether the receive buffer is currently updated by the FIFO handler. 0_B The receive buffer information has been completely updated. 1_B The OUTR update from the FIFO memory is ongoing. A read from OUTR will be delayed. FIFO pointers from the previous read are not yet updated.
SRBT	6	rh	**Standard Receive Buffer Event Trigger** This bit triggers a standard receive buffer event when set. If enabled by RBCTR.SRBIEN, the service request output SRx selected by RBCTR.SRBINP becomes activated until the bit is cleared. 0_B A standard receive buffer event is not triggered using this bit. 1_B A standard receive buffer event is triggered using this bit.

Field	Bits	Type	Description
STBI	8	rwh	**Standard Transmit Buffer Event** This bit indicates that a standard transmit buffer event has been detected. It is cleared by writing TRBSCR.CSTBI = 1. If enabled by TBCTR.STBIEN, the service request output SRx selected by TBCTR.STBINP becomes activated if a standard transmit buffer event is detected. 0_B A standard transmit buffer event has not been detected. 1_B A standard transmit buffer event has been detected.
TBERI	9	rwh	**Transmit Buffer Error Event** This bit indicates that a transmit buffer error event has been detected. It is cleared by writing TRBSCR.CTBERI = 1. If enabled by TBCTR.TBERIEN, the service request output SRx selected by TBCTR.ATBINP becomes activated if a transmit buffer error event is detected. 0_B A transmit buffer error event has not been detected. 1_B A transmit buffer error event has been detected.
TEMPTY	11	rh	**Transmit Buffer Empty** This bit indicates whether the transmit buffer is empty. 0_B The transmit buffer is not empty. 1_B The transmit buffer is empty.
TFULL	12	rh	**Transmit Buffer Full** This bit indicates whether the transmit buffer is full. 0_B The transmit buffer is not full. 1_B The transmit buffer is full.

Field	Bits	Type	Description
TBUS	13	rh	**Transmit Buffer Busy** This bit indicates whether the transmit buffer is currently updated by the FIFO handler. 0_B The transmit buffer information has been completely updated. 1_B The FIFO memory update after write to INx is ongoing. A write to INx will be delayed. FIFO pointers from the previous INx write are not yet updated.
STBT	14	rh	**Standard Transmit Buffer Event Trigger** This bit triggers a standard transmit buffer event when set. If enabled by TBCTR.STBIEN, the service request output SRx selected by TBCTR.STBINP becomes activated until the bit is cleared. 0_B A standard transmit buffer event is not triggered using this bit. 1_B A standard transmit buffer event is triggered using this bit.
RBFLVL	[22:16]	rh	**Receive Buffer Filling Level** This bit field indicates the filling level of the receive buffer, starting with 0 for an empty buffer.
TBFLVL	[30:24]	rh	**Transmit Buffer Filling Level** This bit field indicates the filling level of the transmit buffer, starting with 0 for an empty buffer.
0	7, 10, 15, 23, 31	r	**Reserved** Read as 0; should be written with 0.

The bits in register TRBSCR are used to clear the notification bits in register TRBSR or to clear the FIFO mechanism for the transmit or receive buffer. A read action always delivers 0.

TRBSCR
Transmit/Receive Buffer Status Clear Register
(118_H) **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
FLUSHTB	FLUSHRB	0			CBDV	CTBERI	CSTBI	0					CARBI	CRBERI	CSRBI
w	w	r			w	w	w	r					w	w	w

Field	Bits	Type	Description
CSRBI	0	w	**Clear Standard Receive Buffer Event** 0_B No effect. 1_B Clear TRBSR.SRBI.
CRBERI	1	w	**Clear Receive Buffer Error Event** 0_B No effect. 1_B Clear TRBSR.RBERI.
CARBI	2	w	**Clear Alternative Receive Buffer Event** 0_B No effect. 1_B Clear TRBSR.ARBI.
CSTBI	8	w	**Clear Standard Transmit Buffer Event** 0_B No effect. 1_B Clear TRBSR.STBI.
CTBERI	9	w	**Clear Transmit Buffer Error Event** 0_B No effect. 1_B Clear TRBSR.TBERI.
CBDV	10	w	**Clear Bypass Data Valid** 0_B No effect. 1_B Clear BYPCR.BDV.

Field	Bits	Type	Description
FLUSHRB	14	w	**Flush Receive Buffer** 0_B No effect. 1_B The receive FIFO buffer is cleared (filling level is cleared and output pointer is set to input pointer value). Should only be used while the FIFO buffer is not taking part in data traffic.
FLUSHTB	15	w	**Flush Transmit Buffer** 0_B No effect. 1_B The transmit FIFO buffer is cleared (filling level is cleared and output pointer is set to input pointer value). Should only be used while the FIFO buffer is not taking part in data traffic.
0	[7:3], [13:11], [31:16]	r	**Reserved** Read as 0; should be written with 0.

14.11.9.3 Transmit FIFO Buffer Control Registers

The transmit FIFO buffer is controlled by register TBCTR. TBCTR can only by written if the transmit buffer functionality is enabled by CCFG.TB = 1, otherwise write accesses are ignored.

TBCTR
Transmitter Buffer Control Register (108_H) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TBERIEN	STBIEN	0	LOF	0	SIZE			0		ATBINP			STBINP		
rw	rw	r	rw	r	rw			r		rw			rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STBTEN	STBTM	LIMIT						0		DPTR					
rw	rw	rw						r		w					

Field	Bits	Type	Description
DPTR	[5:0]	w	**Data Pointer** This bit field defines the start value for the transmit buffer pointers when assigning the FIFO entries to the transmit FIFO buffer. A read always delivers 0. When writing DPTR while SIZE = 0, both transmitter pointers TDIPTR and RTDOPTR in register TRBPTR are updated with the written value and the buffer is considered as empty. A write access to DPTR while SIZE > 0 is ignored and does not modify the pointers.
LIMIT	[13:8]	rw	**Limit For Interrupt Generation** This bit field defines the target filling level of the transmit FIFO buffer that is used for the standard transmit buffer event detection.
STBTM	14	rw	**Standard Transmit Buffer Trigger Mode** This bit selects the standard transmit buffer event trigger mode. 0_B Trigger mode 0: While TRBSR.STBT=1, a standard buffer event will be generated whenever there is a data transfer to TBUF or data write to INx (depending on TBCTR.LOF setting). STBT is cleared when TRBSR.TBFLVL=TBCTR.LIMIT. 1_B Trigger mode 1: While TRBSR.STBT=1, a standard buffer event will be generated whenever there is a data transfer to TBUF or data write to INx (depending on TBCTR.LOF setting). STBT is cleared when TRBSR.TBFLVL=TBCTR.SIZE.
STBTEN	15	rw	**Standard Transmit Buffer Trigger Enable** This bit enables/disables triggering of the standard transmit buffer event through bit TRBSR.STBT. 0_B The standard transmit buffer event trigger through bit TRBSR.STBT is disabled. 1_B The standard transmit buffer event trigger through bit TRBSR.STBT is enabled.

Field	Bits	Type	Description
STBINP	[18:16]	rw	**Standard Transmit Buffer Interrupt Node Pointer** This bit field defines which service request output SRx becomes activated in case of a standard transmit buffer event. 000_B Output SR0 becomes activated. 001_B Output SR1 becomes activated. 010_B Output SR2 becomes activated. 011_B Output SR3 becomes activated. 100_B Output SR4 becomes activated. 101_B Output SR5 becomes activated. *Note: All other settings of the bit field are reserved.*
ATBINP	[21:19]	rw	**Alternative Transmit Buffer Interrupt Node Pointer** This bit field define which service request output SRx will be activated in case of a transmit buffer error event. 000_B Output SR0 becomes activated. 001_B Output SR1 becomes activated. 010_B Output SR2 becomes activated. 011_B Output SR3 becomes activated. 100_B Output SR4 becomes activated. 101_B Output SR5 becomes activated. *Note: All other settings of the bit field are reserved.*
SIZE	[26:24]	rw	**Buffer Size** This bit field defines the number of FIFO entries assigned to the transmit FIFO buffer. 000_B The FIFO mechanism is disabled. The buffer does not accept any request for data. 001_B The FIFO buffer contains 2 entries. 010_B The FIFO buffer contains 4 entries. 011_B The FIFO buffer contains 8 entries. 100_B The FIFO buffer contains 16 entries. 101_B The FIFO buffer contains 32 entries. 110_B The FIFO buffer contains 64 entries. 111_B Reserved

Field	Bits	Type	Description
LOF	28	rw	**Buffer Event on Limit Overflow** This bit defines which relation between filling level and programmed limit leads to a standard transmit buffer event. 0_B A standard transmit buffer event occurs when the filling level equals the limit value and gets lower due to transmission of a data word. 1_B A standard transmit buffer interrupt event occurs when the filling level equals the limit value and gets bigger due to a write access to a data input location INx.
STBIEN	30	rw	**Standard Transmit Buffer Interrupt Enable** This bit enables/disables the generation of a standard transmit buffer interrupt in case of a standard transmit buffer event. 0_B The standard transmit buffer interrupt generation is disabled. 1_B The standard transmit buffer interrupt generation is enabled.
TBERIEN	31	rw	**Transmit Buffer Error Interrupt Enable** This bit enables/disables the generation of a transmit buffer error interrupt in case of a transmit buffer error event (software writes to a full transmit buffer). 0_B The transmit buffer error interrupt generation is disabled. 1_B The transmit buffer error interrupt generation is enabled.
0	[7:6], [23:22], 27, 29	r	**Reserved** Read as 0; should be written with 0.

14.11.9.4 Receive FIFO Buffer Control Registers

The receive FIFO buffer is controlled by register RBCTR. This register can only be written if the receive buffer functionality is enabled by CCFG.RB = 1, otherwise write accesses are ignored.

RBCTR
Receiver Buffer Control Register ($10C_H$) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RBERIEN	SRBIEN	ARBIEN	LOF	RNM	SIZE			RCIM		ARBINP			SRBINP		
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
SRBTEN	SRBTM	LIMIT						0		DPTR					
rw	rw	rw						r		w					

Field	Bits	Type	Description
DPTR	[5:0]	w	**Data Pointer** This bit field defines the start value for the receive buffer pointers when assigning the FIFO entries to the receive FIFO buffer. A read always delivers 0. When writing DPTR while SIZE = 0, both receiver pointers RDIPTR and RDOPTR in register TRBPTR are updated with the written value and the buffer is considered as empty. A write access to DPTR while SIZE > 0 is ignored and does not modify the pointers.
LIMIT	[13:8]	rw	**Limit For Interrupt Generation** This bit field defines the target filling level of the receive FIFO buffer that is used for the standard receive buffer event detection.

Field	Bits	Type	Description
SRBTM	14	rw	**Standard Receive Buffer Trigger Mode** This bit selects the standard receive buffer event trigger mode. 0_B Trigger mode 0: While TRBSR.SRBT=1, a standard receive buffer event will be generated whenever there is a new data received or data read out (depending on RBCTR.LOF setting). SRBT is cleared when TRBSR.RBFLVL=RBCTR.LIMIT. 1_B Trigger mode 1: While TRBSR.SRBT=1, a standard receive buffer event will be generated whenever there is a new data received or data read out (depending on RBCTR.LOF setting). SRBT is cleared when TRBSR.RBFLVL=0.
SRBTEN	15	rw	**Standard Receive Buffer Trigger Enable** This bit enables/disables triggering of the standard receive buffer event through bit TRBSR.SRBT. 0_B The standard receive buffer event trigger through bit TRBSR.SRBT is disabled. 1_B The standard receive buffer event trigger through bit TRBSR.SRBT is enabled.
SRBINP	[18:16]	rw	**Standard Receive Buffer Interrupt Node Pointer** This bit field defines which service request output SRx becomes activated in case of a standard receive buffer event. 000_B Output SR0 becomes activated. 001_B Output SR1 becomes activated. 010_B Output SR2 becomes activated. 011_B Output SR3 becomes activated. 100_B Output SR4 becomes activated. 101_B Output SR5 becomes activated. *Note: All other settings of the bit field are reserved.*

Field	Bits	Type	Description
ARBINP	[21:19]	rw	**Alternative Receive Buffer Interrupt Node Pointer** This bit field defines which service request output SRx becomes activated in case of an alternative receive buffer event or a receive buffer error event. 000_B Output SR0 becomes activated. 001_B Output SR1 becomes activated. 010_B Output SR2 becomes activated. 011_B Output SR3 becomes activated. 100_B Output SR4 becomes activated. 101_B Output SR5 becomes activated. *Note: All other settings of the bit field are reserved.*
RCIM	[23:22]	rw	**Receiver Control Information Mode** This bit field defines which information from the receiver status register RBUFSR is propagated as 5 bit receiver control information RCI[4:0] to the receive FIFO buffer and can be read out in registers OUT(D)R. 00_B RCI[4] = PERR, RCI[3:0] = WLEN 01_B RCI[4] = SOF, RCI[3:0] = WLEN 10_B RCI[4] = 0, RCI[3:0] = WLEN 11_B RCI[4] = PERR, RCI[3] = PAR, RCI[2:1] = 00_B, RCI[0] = SOF
SIZE	[26:24]	rw	**Buffer Size** This bit field defines the number of FIFO entries assigned to the receive FIFO buffer. 000_B The FIFO mechanism is disabled. The buffer does not accept any request for data. 001_B The FIFO buffer contains 2 entries. 010_B The FIFO buffer contains 4 entries. 011_B The FIFO buffer contains 8 entries. 100_B The FIFO buffer contains 16 entries. 101_B The FIFO buffer contains 32 entries. 110_B The FIFO buffer contains 64 entries. 111_B Reserved

Field	Bits	Type	Description
RNM	27	rw	**Receiver Notification Mode** This bit defines the receive buffer event mode. The receive buffer error event is not affected by RNM. 0_B Filling level mode: A standard receive buffer event occurs when the filling level equals the limit value and changes, either due to a read access from OUTR (LOF = 0) or due to a new received data word (LOF = 1). 1_B RCI mode: A standard receive buffer event occurs when register OUTR is updated with a new value if the corresponding value in OUTR.RCI[4] = 0. If OUTR.RCI[4] = 1, an alternative receive buffer event occurs instead of the standard receive buffer event.
LOF	28	rw	**Buffer Event on Limit Overflow** This bit defines which relation between filling level and programmed limit leads to a standard receive buffer event in filling level mode (RNM = 0). In RCI mode (RNM = 1), bit fields LIMIT and LOF are ignored. 0_B A standard receive buffer event occurs when the filling level equals the limit value and gets lower due to a read access from OUTR. 1_B A standard receive buffer event occurs when the filling level equals the limit value and gets bigger due to the reception of a new data word.
ARBIEN	29	rw	**Alternative Receive Buffer Interrupt Enable** This bit enables/disables the generation of an alternative receive buffer interrupt in case of an alternative receive buffer event. 0_B The alternative receive buffer interrupt generation is disabled. 1_B The alternative receive buffer interrupt generation is enabled.

Field	Bits	Type	Description
SRBIEN	30	rw	**Standard Receive Buffer Interrupt Enable** This bit enables/disables the generation of a standard receive buffer interrupt in case of a standard receive buffer event. 0_B The standard receive buffer interrupt generation is disabled. 1_B The standard receive buffer interrupt generation is enabled.
RBERIEN	31	rw	**Receive Buffer Error Interrupt Enable** This bit enables/disables the generation of a receive buffer error interrupt in case of a receive buffer error event (the software reads from an empty receive buffer). 0_B The receive buffer error interrupt generation is disabled. 1_B The receive buffer error interrupt generation is enabled.
0	[7:6]	r	**Reserved** Read as 0; should be written with 0.

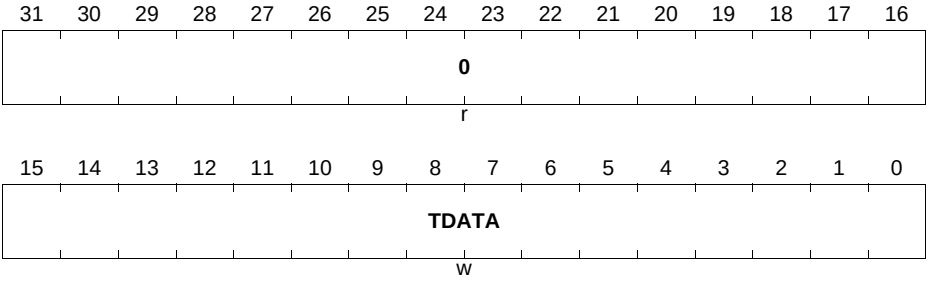
14.11.9.5 FIFO Buffer Data Registers

The 32 independent data input locations IN00 to IN31 are addresses that can be used as data entry locations for the transmit FIFO buffer. Data written to one of these locations will be stored in the transmit buffer FIFO. Additionally, the 5-bit coding of the number [31:0] of the addressed data input location represents the transmit control information TCI.

If the FIFO is already full and new data is written to it, the write access is ignored and a transmit buffer error event is signaled.

INx (x = 00-31)
Transmit FIFO Buffer Input Location x

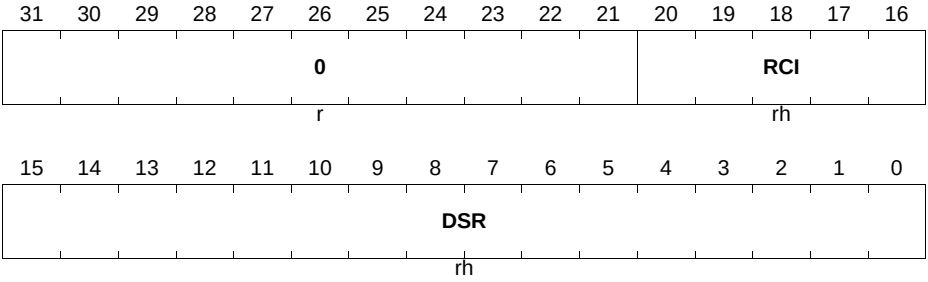
(180_H + x *4) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
TDATA	[15:0]	w	**Transmit Data** This bit field contains the data to be transmitted (write view), read actions deliver 0. A write action to at least the low byte of TDATA triggers the data storage in the FIFO.
0	[31:16]	r	**Reserved** Read as 0; should be written with 0.

The receiver FIFO buffer output register OUTR shows the oldest received data word in the FIFO buffer and contains the receiver control information RCI containing the information selected by RBCTR.RCIM. A read action from this address location delivers the received data. With a read access of at least the low byte, the data is declared to be read and the next entry becomes visible. Write accesses to OUTR are ignored.

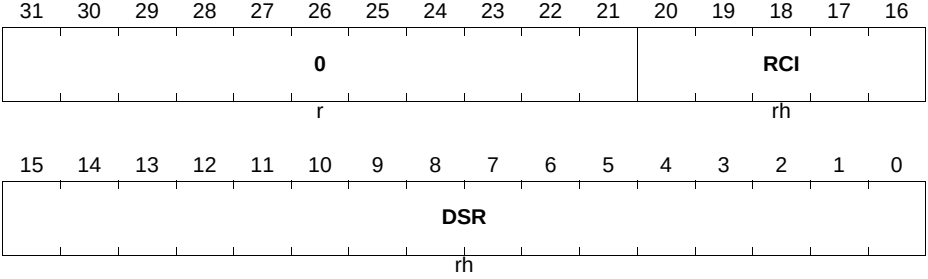
OUTR
Receiver Buffer Output Register (11C$_H$) **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
DSR	[15:0]	rh	**Received Data** This bit field monitors the content of the oldest data word in the receive FIFO. Reading at least the low byte releases the buffer entry currently shown in DSR.
RCI	[20:16]	rh	**Receiver Control Information** This bit field monitors the receiver control information associated to DSR. The bit structure of RCI depends on bit field RBCTR.RCIM.
0	[31:21]	r	**Reserved** Read as 0; should be written with 0.

If a debugger should be used to monitor the received data in the FIFO buffer, the FIFO mechanism must not be activated in order to guaranty data consistency. Therefore, a second address set is available, named OUTDR (D like debugger), having the same bit fields like the original buffer output register OUTR, but without the FIFO mechanism. A debugger can read here (in order to monitor the receive data flow) without the risk of data corruption. Write accesses to OUTDR are ignored.

OUTDR
Receiver Buffer Output Register L for Debugger
(120_H) **Reset Value: $0000\ 0000_H$**



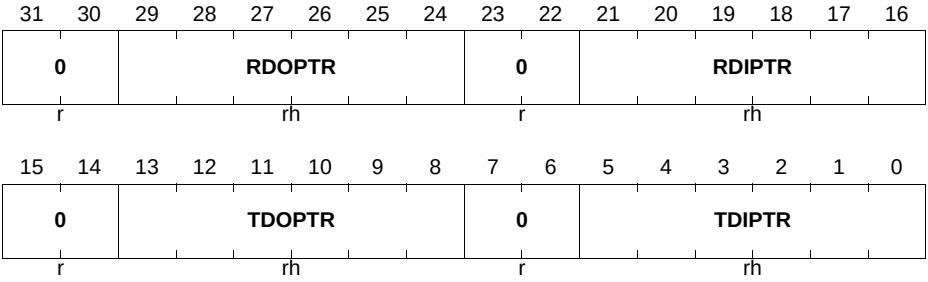
Field	Bits	Type	Description
DSR	[15:0]	rh	**Data from Shift Register** Same as OUTR.DSR, but without releasing the buffer after a read action.
RCI	[20:16]	rh	**Receive Control Information from Shift Register** Same as OUTR.RCI.
0	[31:21]	r	**Reserved** Read as 0; should be written with 0.

14.11.9.6 FIFO Buffer Pointer Registers

The pointers for FIFO handling of the transmit and receive FIFO buffers are located in register TRBPTR. The pointers are automatically handled by the FIFO buffer mechanism and do not need to be modified by software. As a consequence, these registers can only be read by software (e.g. for verification purposes), whereas write accesses are ignored.

TRBPTR
Transmit/Receive Buffer Pointer Register

(110_H) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
TDIPTR	[5:0]	rh	**Transmitter Data Input Pointer** This bit field indicates the buffer entry that will be used for the next transmit data coming from the INx addresses.
TDOPTR	[13:8]	rh	**Transmitter Data Output Pointer** This bit field indicates the buffer entry that will be used for the next transmit data to be output to TBUF.
RDIPTR	[21:16]	rh	**Receiver Data Input Pointer** This bit field indicates the buffer entry that will be used for the next receive data coming from RBUF.
RDOPTR	[29:24]	rh	**Receiver Data Output Pointer** This bit field indicates the buffer entry that will be used for the next receive data to be output at the OUT(D)R addresses.
0	[7:6], [15:14], [23:22], [31:30]	r	**Reserved** Read as 0; should be written with 0.

14.12 Interconnects

The XMC1100 device contains one USIC module (USIC0) with 2 communication channels.

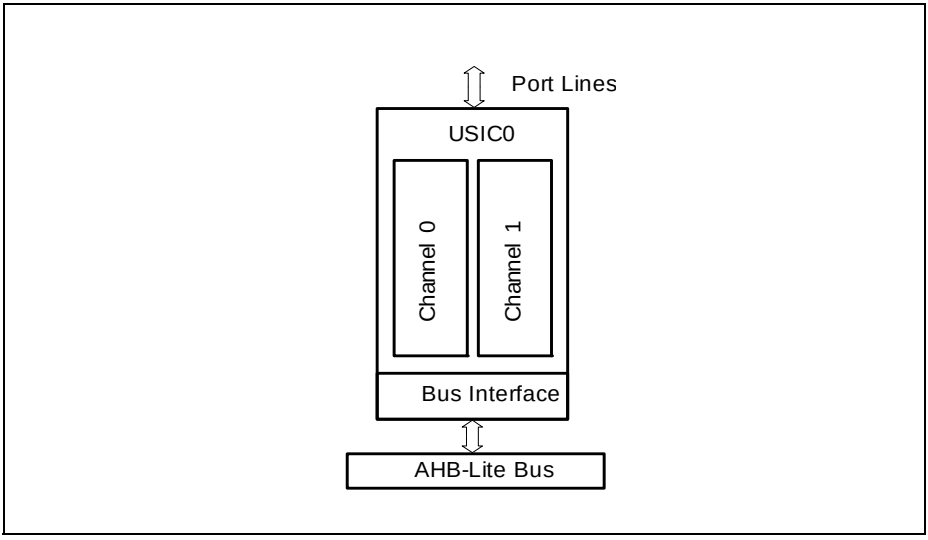


Figure 14-66 USIC Module Structure in XMC1100

Figure 14-67 shows the I/O lines of one USIC channel. The tables in this section define the pin assignments and internal connections of the USIC channels I/O lines in the XMC1100 device. Naming convention: USICx_CHy refers to USIC module x channel y.

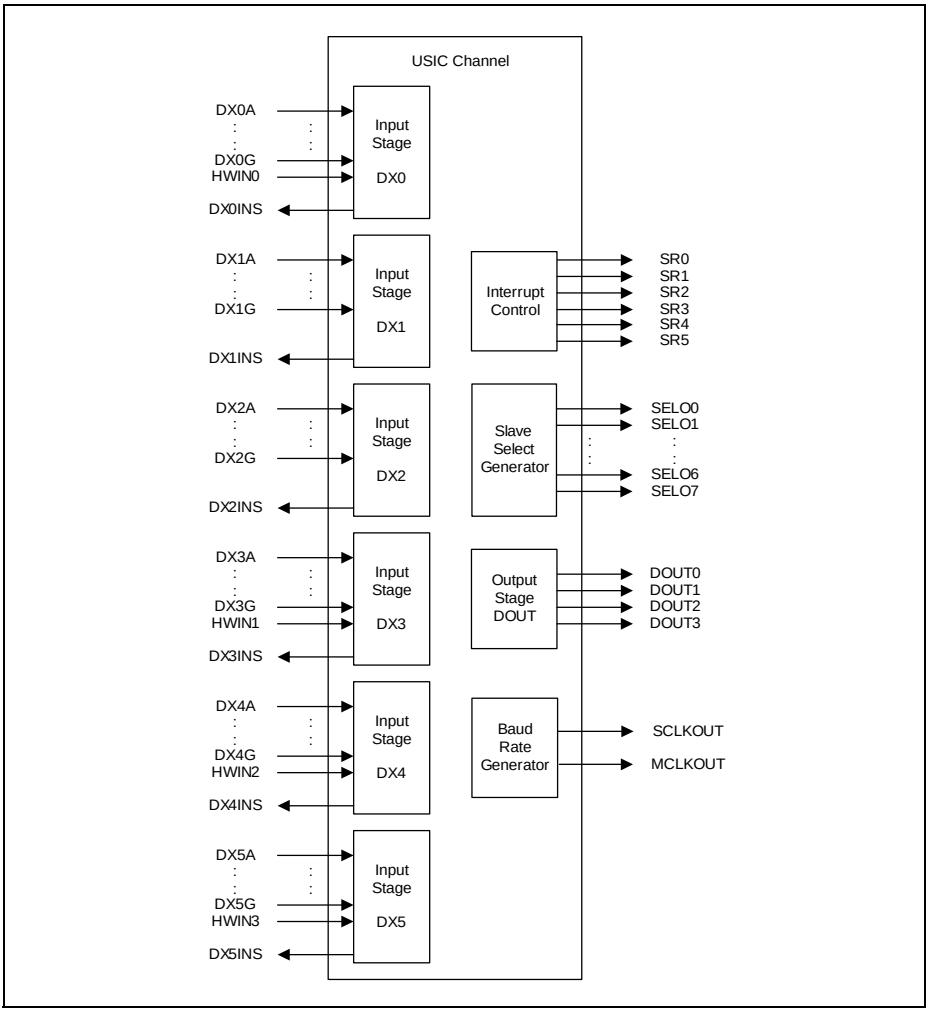


Figure 14-67 USIC Channel I/O Lines

The service request outputs SR[5:0] of one USIC channel is combined with those of the other channel within the module. Therefore, only 6 service request outputs are available per module.

14.12.1 USIC Module 0 Interconnects

The interconnects of USIC module 0 is grouped into the following categories:

- **USIC Module 0 Channel 0 Interconnects** (**Table 14-22**)
- **USIC Module 0 Channel 1 Interconnects** (**Table 14-23**)
- **USIC Module 0 Module Interconnects** (**Table 14-24**)

Table 14-22 USIC Module 0 Channel 0 Interconnects

Input/Output	I/O	Connected To	Description
Data Inputs (DX0)			
USIC0_CH0.DX0A	I	P0.14	Shift data input
USIC0_CH0.DX0B	I	P0.15	Shift data input
USIC0_CH0.DX0C	I	P1.0	Shift data input
USIC0_CH0.DX0D	I	P1.1	Shift data input
USIC0_CH0.DX0E	I	P2.0	Shift data input
USIC0_CH0.DX0F	I	P2.1	Shift data input
USIC0_CH0.DX0G	I	USIC0_CH0.DX3INS	Shift data input
USIC0_CH0.HWIN0	I	P1.0	HW controlled shift data input
Clock Inputs			
USIC0_CH0.DX1A	I	P0.14	Shift clock input
USIC0_CH0.DX1B	I	P0.8	Shift clock input
USIC0_CH0.DX1C	I	P0.7	Shift clock input
USIC0_CH0.DX1D	I	P1.1	Shift clock input
USIC0_CH0.DX1E	I	P2.0	Shift clock input
USIC0_CH0.DX1F	I	USIC0_CH0.DX0INS	Shift clock input
USIC0_CH0.DX1G	I	USIC0_CH0.DX4INS	Loop back shift clock input
Control Inputs			
USIC0_CH0.DX2A	I	P0.0	Shift control input
USIC0_CH0.DX2B	I	P0.9	Shift control input
USIC0_CH0.DX2C	I	P0.10	Shift control input
USIC0_CH0.DX2D	I	P0.11	Shift control input
USIC0_CH0.DX2E	I	P0.12	Shift control input
USIC0_CH0.DX2F	I	P0.13	Shift control input
USIC0_CH0.DX2G	I	USIC0_CH0.DX5INS	Loop back shift control input
Data Inputs (DX3)			
USIC0_CH0.DX3A	I	P2.2	Shift data input

Table 14-22 USIC Module 0 Channel 0 Interconnects (cont'd)

Input/Output	I/O	Connected To	Description
USIC0_CH0.DX3B	I	P2.4	Shift data input
USIC0_CH0.DX3C	I	P2.10	Shift data input
USIC0_CH0.DX3D	I	P2.8	Shift data input
USIC0_CH0.DX3E	I	P2.6	Shift data input
USIC0_CH0.DX3F	I	USIC0_CH0.DX5INS	Shift data input
USIC0_CH0.DX3G	I	USIC0_CH0.DOUT0	Shift data input
USIC0_CH0.HWIN1	I	P1.1	HW controlled shift data input
Data Inputs (DX4)			
USIC0_CH0.DX4A	I	P2.2	Shift data input
USIC0_CH0.DX4B	I	P2.4	Shift data input
USIC0_CH0.DX4C	I	P2.10	Shift data input
USIC0_CH0.DX4D	I	P2.8	Shift data input
USIC0_CH0.DX4E	I	P2.6	Shift data input
USIC0_CH0.DX4F	I	USIC0_CH0.DX5INS	Shift data input
USIC0_CH0.DX4G	I	USIC0_CH0.SCLKOUT	Shift data input
USIC0_CH0.HWIN2	I	P1.2	HW controlled shift data input
Data Inputs (DX5)			
USIC0_CH0.DX5A	I	P2.9	Shift data input
USIC0_CH0.DX5B	I	P2.3	Shift data input
USIC0_CH0.DX5C	I	P2.7	Shift data input
USIC0_CH0.DX5D	I	P2.5	Shift data input
USIC0_CH0.DX5E	I	P1.4	Shift data input
USIC0_CH0.DX5F	I	0	Shift data input
USIC0_CH0.DX5G	I	USIC0_CH0.SELO0	Shift data input
USIC0_CH0.HWIN3	I	P1.3	HW controlled shift data input
Data Outputs			

Table 14-22 USIC Module 0 Channel 0 Interconnects (cont'd)

Input/Output	I/O	Connected To	Description
USIC0_CH0.DOUT0	O	P0.14 P0.15 P1.0 P1.0 (HW1_OUT) P1.1 P1.5 P2.0 P2.1 USIC0_CH0.DX3G	Shift data output
USIC0_CH0.DOUT1	O	P1.1 (HW1_OUT)	Shift data output
USIC0_CH0.DOUT2	O	P1.2 (HW1_OUT)	Shift data output
USIC0_CH0.DOUT3	O	P1.3 (HW1_OUT)	Shift data output
Clock Outputs			
USIC0_CH0.MCLKOUT	O	P0.11	Master clock output
USIC0_CH0.SCLKOUT	O	P0.7 P0.8 P0.14 P2.0 USIC0_CH0.DX4G	Shift clock output
Control Outputs			
USIC0_CH0.SELO0	O	P0.0 P0.9 P1.4 USIC0_CH0.DX5G	Shift control output
USIC0_CH0.SELO1	O	P0.10 P1.5	Shift control output
USIC0_CH0.SELO2	O	P0.11	Shift control output
USIC0_CH0.SELO3	O	P0.12	Shift control output
USIC0_CH0.SELO4	O	P0.13	Shift control output
USIC0_CH0.SELO5	O	not connected	Shift control output
USIC0_CH0.SELO6	O	not connected	Shift control output
USIC0_CH0.SELO7	O	not connected	Shift control output
System Related Outputs			
USIC0_CH0.DX0INS	O	USIC0_CH0.DX1F	Selected DX0 input signal

Table 14-22 USIC Module 0 Channel 0 Interconnects (cont'd)

Input/Output	I/O	Connected To	Description
USIC0_CH0.DX1INS	O	not connected	Selected DX1 input signal
USIC0_CH0.DX2INS	O	CCU40.IN0L	Selected DX2 input signal
USIC0_CH0.DX3INS	O	USIC0_CH0.DX0G	Selected DX3 input signal
USIC0_CH0.DX4INS	O	USIC0_CH0.DX1G	Selected DX4 input signal
USIC0_CH0.DX5INS	O	USIC0_CH0.DX2G USIC0_CH0.DX3F USIC0_CH0.DX4F	Selected DX5 input signal

Table 14-23 USIC Module 0 Channel 1 Interconnects

Input/Output	I/O	Connected To	Description
Data Inputs (DX0)			
USIC0_CH1.DX0A	I	P1.3	Shift data input
USIC0_CH1.DX0B	I	P1.2	Shift data input
USIC0_CH1.DX0C	I	P0.6	Shift data input
USIC0_CH1.DX0D	I	P0.7	Shift data input
USIC0_CH1.DX0E	I	P2.11	Shift data input
USIC0_CH1.DX0F	I	P2.10	Shift data input
USIC0_CH1.DX0G	I	USIC0_CH1.DX3INS	Shift data input
USIC0_CH1.HWIN0	I	ERU0.PDOUT0	HW controlled shift data input
Clock Inputs			
USIC0_CH1.DX1A	I	P1.3	Shift clock input
USIC0_CH1.DX1B	I	P0.8	Shift clock input
USIC0_CH1.DX1C	I	P0.7	Shift clock input
USIC0_CH1.DX1D	I	0	Shift clock input
USIC0_CH1.DX1E	I	P2.11	Shift clock input
USIC0_CH1.DX1F	I	USIC0_CH1.DX0INS	Shift clock input
USIC0_CH1.DX1G	I	USIC0_CH1.DX4INS	Loop back shift clock input
Control Inputs			
USIC0_CH1.DX2A	I	P0.0	Shift control input
USIC0_CH1.DX2B	I	P0.9	Shift control input
USIC0_CH1.DX2C	I	P0.10	Shift control input

Table 14-23 USIC Module 0 Channel 1 Interconnects (cont'd)

Input/Output	I/O	Connected To	Description
USIC0_CH1.DX2D	I	P0.11	Shift control input
USIC0_CH1.DX2E	I	P1.1	Shift control input
USIC0_CH1.DX2F	I	P2.0	Shift control input
USIC0_CH1.DX2G	I	USIC0_CH1.DX5INS	Loop back shift control input
Data Inputs (DX3)			
USIC0_CH1.DX3A	I	P2.1	Shift data input
USIC0_CH1.DX3B	I	P2.9	Shift data input
USIC0_CH1.DX3C	I	P2.3	Shift data input
USIC0_CH1.DX3D	I	P2.7	Shift data input
USIC0_CH1.DX3E	I	P2.5	Shift data input
USIC0_CH1.DX3F	I	USIC0_CH1.DX5INS	Shift data input
USIC0_CH1.DX3G	I	USIC0_CH1.DOUT0	Shift data input
USIC0_CH1.HWIN1	I	0	HW controlled shift data input
Data Inputs (DX4)			
USIC0_CH1.DX4A	I	P2.1	Shift data input
USIC0_CH1.DX4B	I	P2.9	Shift data input
USIC0_CH1.DX4C	I	P2.3	Shift data input
USIC0_CH1.DX4D	I	P2.7	Shift data input
USIC0_CH1.DX4E	I	P2.5	Shift data input
USIC0_CH1.DX4F	I	USIC0_CH1.DX5INS	Shift data input
USIC0_CH1.DX4G	I	USIC0_CH1.SCLKOUT	Shift data input
USIC0_CH1.HWIN2	I	ERU0.PDOUT1	HW controlled shift data input
Data Inputs (DX5)			
USIC0_CH1.DX5A	I	P2.2	Shift data input
USIC0_CH1.DX5B	I	P2.4	Shift data input
USIC0_CH1.DX5C	I	P2.8	Shift data input
USIC0_CH1.DX5D	I	P2.6	Shift data input
USIC0_CH1.DX5E	I	P1.4	Shift data input
USIC0_CH1.DX5F	I	P1.5	Shift data input
USIC0_CH1.DX5G	I	USIC0.SR0	Shift data input

Table 14-23 USIC Module 0 Channel 1 Interconnects (cont'd)

Input/Output	I/O	Connected To	Description
USIC0_CH1.HWIN3	I	USIC0_CH1.DOUT0	HW controlled shift data input
Data Outputs			
USIC0_CH1.DOUT0	O	P0.6 P0.7 P1.2 P1.3 P2.10 P2.11 USIC0_CH1.DX3G USIC0_CH1.HWIN3	Shift data output
USIC0_CH1.DOUT1	O	not connected	Shift data output
USIC0_CH1.DOUT2	O	not connected	Shift data output
USIC0_CH1.DOUT3	O	not connected	Shift data output
Clock Outputs			
USIC0_CH1.MCLKOUT	O	P0.6 P0.15	Master clock output
USIC0_CH1.SCLKOUT	O	P0.8 P1.3 P1.4 P2.1 P2.11 USIC0_CH1.DX4G	Shift clock output
Control Outputs			
USIC0_CH1.SELO0	O	P0.0 P0.9 P1.1	Shift control output
USIC0_CH1.SELO1	O	P0.10 P1.4	Shift control output
USIC0_CH1.SELO2	O	P0.11 P1.5	Shift control output
USIC0_CH1.SELO3	O	not connected	Shift control output
USIC0_CH1.SELO4	O	not connected	Shift control output
USIC0_CH1.SELO5	O	not connected	Shift control output
USIC0_CH1.SELO6	O	not connected	Shift control output

Table 14-23 USIC Module 0 Channel 1 Interconnects (cont'd)

Input/Output	I/O	Connected To	Description
USIC0_CH1.SELO7	O	not connected	Shift control output
System Related Outputs			
USIC0_CH1.DX0INS	O	USIC0_CH1.DX1F	Selected DX0 input signal
USIC0_CH1.DX1INS	O	not connected	Selected DX1 input signal
USIC0_CH1.DX2INS	O	CCU40.IN1L	Selected DX2 input signal
USIC0_CH1.DX3INS	O	USIC0_CH1.DX0G	Selected DX3 input signal
USIC0_CH1.DX4INS	O	USIC0_CH1.DX1G	Selected DX4 input signal
USIC0_CH1.DX5INS	O	USIC0_CH1.DX2G USIC0_CH1.DX3F USIC0_CH1.DX4F	Selected DX5 input signal

Table 14-24 USIC Module 0 Module Interconnects

Input/Output	I/O	Connected To	Description
USIC0_SR0	O	NVIC USIC0_CH1.DX5G	interrupt output lines (service requests SRx)
USIC0_SR[5:1]	O	NVIC	interrupt output lines (service requests SRx)

15 Analog-to-Digital Converter (VADC)

The analog-to-digital converter (ADC) in XMC1100 series provides a subset of functions compared to the feature-rich VADC (versatile analog-to-digital converter) which is implemented in XMC1200 and XMC1300 series.

The XMC1100 provides a series of analog input channels combined into two groups. The ADC is based on a high-speed 12-Bit converter clocked at 32 MHz using the successive approximation register (SAR) principle to convert analog input values (voltages) to discrete digital values. Each analog input channel can be configured to be amplified by an adjustable gain factor. Conversion mode and sampling time can be configured for each group in order to meet the requirements of the application.

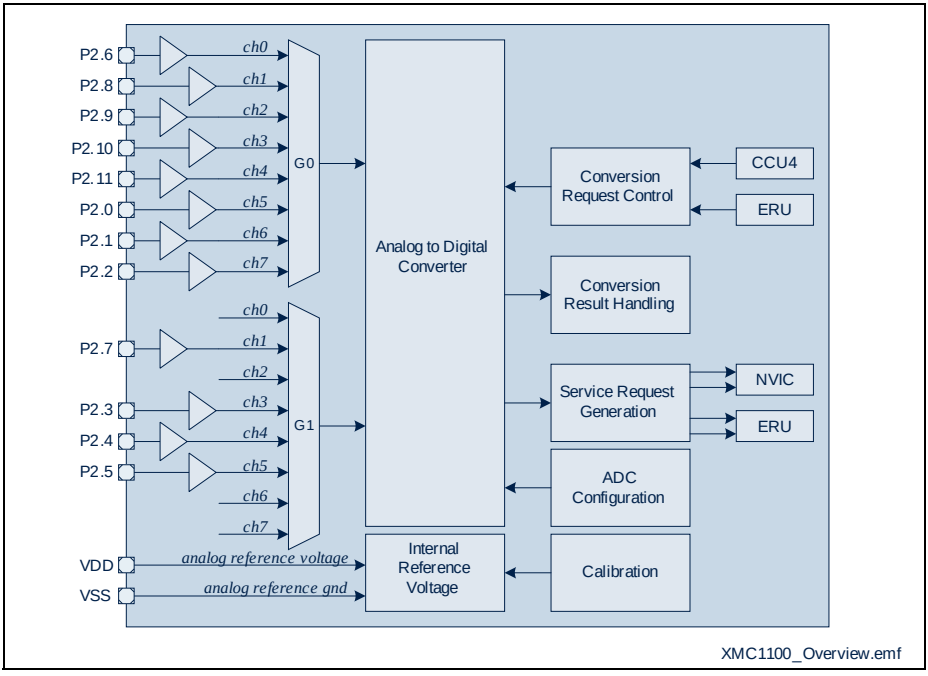


Figure 15-1

The reference voltage of the ADC is internally generated and can be automatically calibrated to the supply voltage of the device. If required, additional calibration steps can be automatically perfomed after each conversion.

The ADC offers two first-order sigma-delta loops which can be assigned to any of the input channels (**Section 15.1.3**). They hold the quantization error of the previous conversion in order to automatically consider this tiny amount of charge in the next

conversion. This will increase the Effective Number Of Bits (ENOB) while implementing oversampling algorithms.

The Conversion Request Control Unit (**Section 15.2**) controls the sampling and conversion of the analog input signal. Conversions of single channels as well as a scan of selected channels can be configured. The start of the conversion sequence is either a software command or a hardware trigger from an interconnected peripheral or a signal at a port pin.

The conversion result is provided in a result register with a wait-for-read option. The wait-for-read mode avoids data loss due to premature result overwrite, by blocking a conversion until the previous result has been read. Automatic accumulation of consecutive measurements can be performed by the data reduction filter which is described in **Section 15.4.4**.

Flexible service request generation is based on selectable events which can be routed to two interrupts.

- **Source events** indicate the completion of a conversion sequence in the corresponding request source. This event can be used to trigger the setup of a new sequence.
- **Result events** indicate the availability of new result data in the corresponding result register. If data reduction mode is active, events are generated only after a complete accumulation sequence.

Each event can be assigned to one of four service request nodes. This allows grouping the requests according to the requirements of the application.

15.1 Analog Module Activation and Control

The analog converter of the ADC is the functional block that generates the digital result values from the selected input voltage. It draws a permanent current during its operation and can be deactivated between conversions to reduce the consumed overall energy.

Note: After reset, the analog converters are off. They must be enabled before triggering any action involving the converter.

15.1.1 Analog Converter Control

Bit ANOFF in register **SHS0_SHSCFG** forces the analog part into power-down mode, without changing the configuration of the associated groups. While ANOFF = 0, the analog part is enabled.

Wakeup Time from Analog Powerdown

When the converter is activated (ANOFF = 0), it needs a certain wakeup time to settle before a conversion can be properly executed. This wakeup time can be established by waiting the required period before starting a conversion.

15.1.2 Calibration

Calibration compensates deviations caused by process, temperature, and voltage variations. This ensures accurate results throughout the operation time.

Several different calibration cycles are executed for offset and gain calibration. These calibration cycles are executed alternating after a conversion.

An initial startup calibration is required once after a reset. First, the converter must be enabled (ANOFF = 0_B), then the startup calibration can be initiated by setting bit SUCAL in register **GLOBCFG**. Conversions may be started after the initial calibration sequence. This is indicated by parameter **SHS0_SHSCFG**.STATE which can be polled by software.

The start-up calibration phase takes 1920 cycles (1920 × 31.25 ns = 60 µs).

Note: Since the ADC error depends on the temperature, the calibration can be repeated periodically.

15.1.3 Sigma-Delta-Loop Function

Each standard analog-to-digital conversion incurs a quantization error due to the limited number of steps i.e. discrete result values. By activating the sigma-delta-loop function this residual quantization error can be forwarded to the next conversion. The residual charge is stored and added to the next sampled charge. Averaging successive conversion results (with the loop enabled) can, therefore, reduce the quantization error.

*Note: The data accumulation function (see **Section 15.4.4**) supports this averaging.*

The loop control bitfields are available in pairs in register **SHS0_LOOP**.

15.2 Conversion Request Generation

The conversion request unit of a group autonomously handles the generation of conversion requests.

- **Software triggers**
  directly activate the request source and requests a conversion.
- **External triggers**
  an external signal can act as conversion trigger request. As a result it synchronizes the request source activation with external events, such as a trigger pulse from a timer generating a PWM signal, or from a port pin.
- **External gating**
  an external signal can block a conversion request.

Application software selects the trigger type and source (**BRSCTRL**), the gating mechanism (**BRSMR**) and the channel(s) to be converted (**BRSSELx (x = 0 - 1)**). The request source can also be activated directly by software (**BRSMR**.LDEV = 1) without requiring an external trigger. The conversion request is then forwarded to the converter to start the sampling and conversion of the requested channel. **Figure 15-2** gives an overview to the conversion request source.

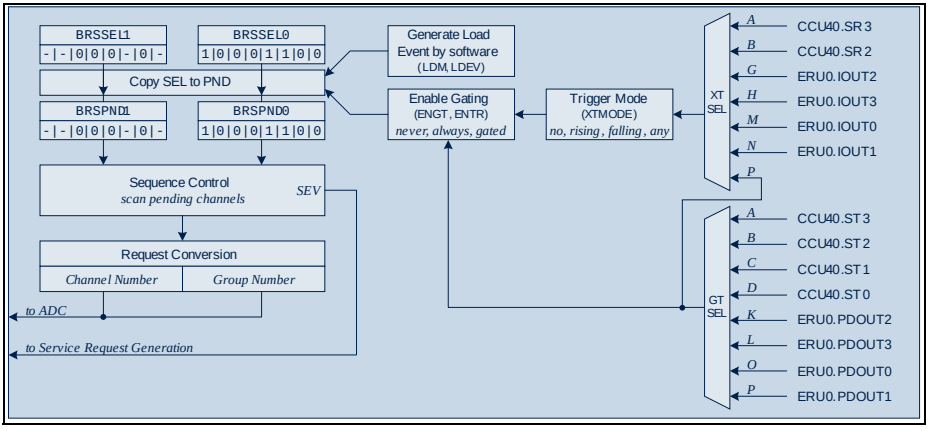


Figure 15-2 Conversion Request Source

The request source can operate in single-shot (scan mode) or in continuous mode (autoscan mode):

- **In single-shot mode**,
  the programmed conversion (sequence) is requested once after being triggered. A subsequent conversion (sequence) must be triggered again.
- **In continuous mode**,
  the programmed conversion (sequence) is automatically requested repeatedly after being triggered once.

External triggers are generated from one of 15 selectable trigger inputs (REQTRx[P:A]) and from one of 16 selectable gating inputs (REQGTx[P:A]). The available trigger signals for the XMC1100 are listed in **Section 15.7.3**.

15.2.1 Channel Scan Request Source Handling

Each analog input channel can be included in or excluded from the scan sequence by setting or clearing the corresponding channel select bit in register **BRSSELx (x = 0 - 1)**. The programmed register value remains unchanged by an ongoing scan sequence. The scan sequence starts with the highest enabled channel number and continues towards lower channel numbers.

Upon a load event which is triggered by hardware signals or software, the request pattern is transferred to the pending bits in register **BRSPNDx (x = 0 - 1)**. The pending conversion requests indicate which input channels are to be converted in an ongoing scan sequence. Each conversion start that was triggered by the scan request source, automatically clears the corresponding pending bit. If the last conversion triggered by the scan source is finished and all pending bits are cleared, the current scan sequence is considered finished and a request source event (REV) is generated.

The trigger and gating unit generates load events from the selected external (outside the ADC) trigger and gating signals. For example, a timer unit can issue a request signal to synchronize conversions to PWM events.

Load events start a scan sequence and can be generated either via software or via the selected hardware triggers. The request source event can also generate an automatic load event, so the programmed sequence is automatically repeated.

Scan Source Operation

Configure the scan request source by executing the following actions:

- Select the input channels for the sequence by programming **BRSSELx (x = 0 - 1)**
- If hardware trigger or gating is desired, select the appropriate trigger and gating inputs and the proper signal transitions by programming **BRSCTRL**. Enable the trigger and select the gating mode by programming **BRSMR**.[1)]
- Define the load event operation (handling of pending bits, autoscan mode) by programming **BRSMR**.
  A load event with bit LDM = 0 copies the content of **BRSSELx (x = 0 - 1)** to **BRSPNDx (x = 0 - 1)** (overwrite mode). This starts a new scan sequence and aborts any pending conversions from a previous scan sequence.
  A load event with bit LDM = 1 OR-combines the content of **BRSSELx (x = 0 - 1)** to

1) If PDOUT signals from the ERU are used, initialize the ERU accordingly before enabling the gate inputs to avoid unexpected signal transitions.

BRSPNDx (x = 0 - 1) (combine mode). This starts a scan sequence that includes pending conversions from a previous scan sequence.

Start a channel scan sequence by generating a load event:

- If a hardware trigger is selected and enabled, generate the configured transition at the selected input signal, e.g. from a timer or an input pin.
- or: Generate a software load event by setting LDEV = 1 (**BRSMR**)
- or: Generate a load event by writing the scan pattern directly to the pending bits in **BRSPNDx (x = 0 - 1)**. The pattern is copied to **BRSSELx (x = 0 - 1)** and a load event is generated automatically.
  In this case, a scan sequence can be defined and started with a single data write action (provided that the pattern fits into one register).

Note: If autoscan is enabled, a load event is generated automatically each time a request source event occurs when the scan sequence has finished. This permanently repeats the defined scan sequence (autoscan).

Stop or abort an ongoing scan sequence by executing one of the following actions:

- If external gating is enabled, switch the gating signal to the defined inactive level. This does not modify the conversion pending bits, but only prevents issuing conversion requests to the ADC.
- Disable the channel scan source by clearing bitfield ENGT = 00_B. Clear the pending request bits by setting bit CLRPND = 1 (**BRSMR**).

Scan Request Source Events and Interrupt Service Requests

A request source event of a scan source occurs when the last conversion of a scan sequence is finished (all pending bits = 0). A request source event interrupt can be generated based on a request source event. If a request source event is detected, it sets the corresponding indication flag in register **GLOBEFLAG**. This flag can also be set by writing 1 to the corresponding bit position, whereas writing 0 has no effect.

The service request output SRx that is selected by the request source event interrupt node pointer bitfields in register **GLOBEVNP** becomes activated each time the related request source event is detected (and enabled by ENSI) or the related bit position in register **GLOBEFLAG** is written with 1 (this write action simulates a request source event).

The indication flags can be cleared by SW by writing 1 to the corresponding bit position in register **GLOBEFLAG**.[1)]

1) Please refer to **"Service Request Generation" on Page 15-13**.

15.3 Analog Input Channel Configuration

The analog input channels are assigned to one of two groups. For each group a number of parameters can be configured in register **GLOBICLASSy (y = 0 - 1)** that control the conversion of the channels.

15.3.1 Conversion Modes

A conversion can be executed in several ways. The conversion mode is selected according to the requested resolution of the digital result and according to the acceptable conversion time (**Section 15.3.2**).
Use bitfield CMS in register **GLOBICLASSy (y = 0 - 1)** to select a mode.

Standard Conversions

A standard conversion returns a result value with a predefined resolution. 8-bit, 10-bit, and 12-bit resolution can be selected.

These result values can be accumulated.

Fast Compare Mode

In Fast Compare Mode, the selected input voltage is directly compared with a digital value that is stored in the result register. This compare operation returns a binary result indicating the compared input voltage is above or below the given reference value. This result is generated quickly and thus supports monitoring of boundary values.

Fast Compare Mode uses a 10-bit compare value stored left-aligned at bit position 11.

15.3.2 Conversion Timing

The total time required for a conversion comprises the time from the start of the sample phase[1)] until the availability of the result.

The sample rate at which consecutive conversions are triggered also depends on several configurable factors:

- The conversion time, according to the selected conversion mode.
- The frequency of external trigger signals, if enabled.

The sample time can be adjusted according to the application requirements. Use bitfiled STC in register **GLOBICLASSy (y = 0 - 1)** to adjust the sample time.

The conversion time is the sum of sample time and conversion time. It can be computed with the following formula:

$t_{CN} = (2 + STC + N + 2) \times t_{ADC}$

Timing Examples

System assumptions:
f_{ADC} = 32 MHz i.e. t_{ADC} = 31.25 ns

According to the given formula the following minimum conversion times can be achieved:

12-bit conversion:
$t_{CN12C} = (2 + 12 + 2) \times t_{ADC} = 16 \times 31.25$ ns = 500 ns

10-bit conversion:
$t_{CN10C} = (2 + 10) \times t_{ADC} + 2 \times t_{ADC} = 14 \times 31.25$ ns = 437.5 ns

1) The time from the trigger event that requests the corresponding conversion until the start of the sample phase depends on propagation delays of the selected trigger signal.

15.4 Conversion Result Handling

The A/D converters can preprocess the conversion result data to a certain extent before storing them for retrieval by the CPU. This supports the subsequent handling of result data by the application software.

Conversion result handling comprises the following functions:

- **Storage of Conversion Results** and **Data Alignment**
- **Wait-for-Read Mode** to avoid loss of data
- **Result Event Generation**
- **Data Modification**

15.4.1 Storage of Conversion Results

The conversion result values are stored in the result register (**GLOBRES**).

The result register has an individual data valid flag (VF) associated with it. This flag indicates when “new” valid data has been stored in the result register and can be read out.

For standard conversions, result values are available in bitfield RESULT. Conversions in Fast Compare Mode use bitfield RESULT for the reference value, so the result of the operation is stored in bit FCR.

The result register can be read via two different views. These views use different addresses but access the same register data:

- When the result register is read via the **application view** (**GLOBRES**), the corresponding valid flag is automatically cleared when the result is read. This provides an easy handshake between result generation and retrieval. This also supports wait-for-read mode.
- When the result register is read via the **debug view (GLOBRESD)**, the corresponding valid flag remains unchanged when the result is read. This supports debugging by delivering the result value without disturbing the handshake with the application.

15.4.1.1 Data Alignment

The position of a conversion result value within the selected result register depends on two configurations:

- The selected result width (12/10/8 bits, see **Section 15.3.1**)
- The selected data accumulation mode (data reduction, see **Section 15.4.4**)

These options provide the conversion results in a way that minimizes data handling for the application software.

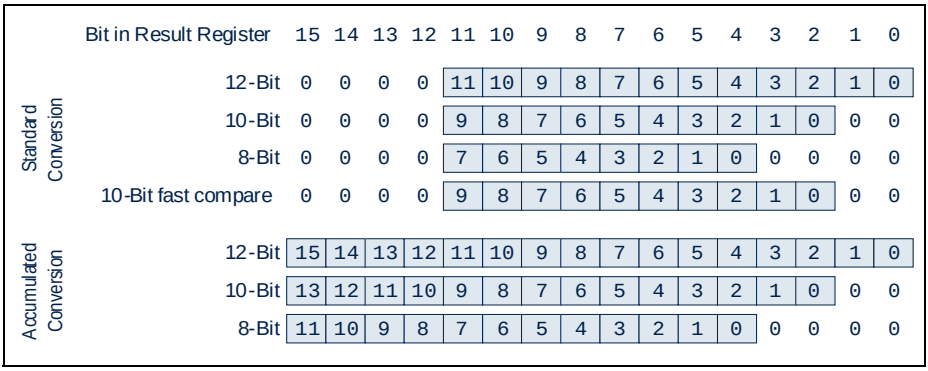


Figure 15-3 Result Register Data Alignment

Bitfield RESULT of register **GLOBRES** must be written by SW with the wanted reference comparison value for Fast Compare Mode. In this mode, bits 11-2 are evaluated, the other bits are ignored.

15.4.2 Wait-for-Read Mode

The wait-for-read mode (**GLOBRCR**.WFR = 1) prevents data loss due to overwriting the result register with a new conversion result before the CPU has read the previous data. Since the results come from different input channels, an overwrite may destroy the result from the previous conversion.

Wait-for-read mode automatically suspends the start of a conversion until the current result has been read. As a result, a conversion or a conversion sequence can be requested by a hardware or software trigger, while each conversion is only started after the result of the previous one has been read. This automatically aligns the conversion sequence with the CPU capability to read the formerly converted result (latency).

15.4.3 Result Event Generation

A result event can be generated when a new value is stored in the result register. Result events can be restricted due to data accumulation and can be generated only if the accumulation is complete.

Result events can also be suppressed completely.

15.4.4 Data Modification

The data resulting from conversions can be automatically modified before being used by an application.

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

The result register can be configured for data reduction, controlled by bitfield **GLOBRCR**.DRCTR.

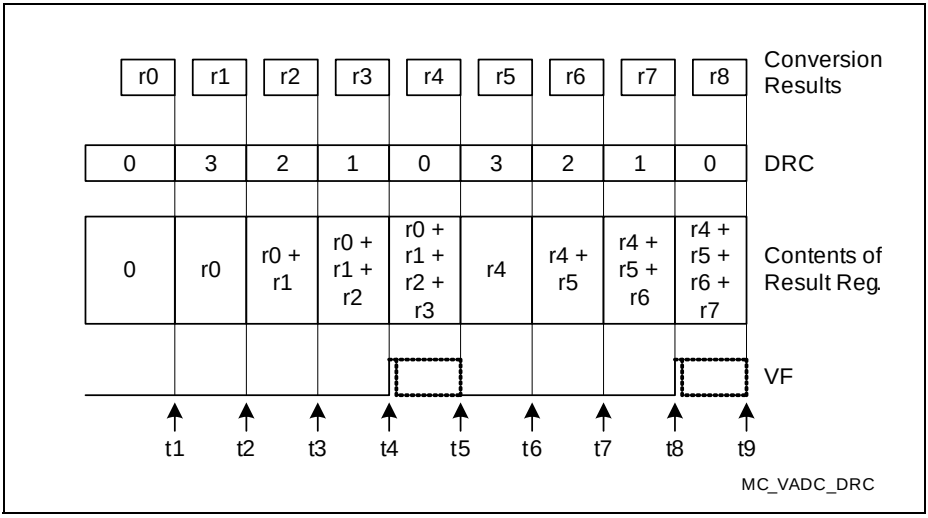


Figure 15-4 Standard Data Reduction Filter

This example shows a data reduction sequence of 4 accumulated conversion results. Eight conversion results (r0 … r7) are accumulated and produce 2 final results.

When a conversion is complete, data is stored in the result register with data reduction mode enabled, and the data handling is controlled by the data reduction counter DRC:

- If DRC = 0 (t1, t5, t9 in the example), the conversion result is stored to the result register. DRC is loaded with the contents of bitfield DRCTR (i.e. the accumulation begins).

- If DRC > 0 (t2, t3, t4 and t6, t7, t8 in the example), the conversion result is added to the value in the result register.
  DRC is decremented by 1.
- If DRC becomes 0, either decremented from 1 (t4 and t8 in the example) or loaded from DRCTR, the valid bit for the respective result register is set and a result register event occurs.
  The final result must be read before the next data reduction sequence starts (before t5 or t9 in the example). This automatically clears the valid flag.

Note: The data reduction filter may be used with a single channel only unless averaging of several channels is intended.

15.5 Service Request Generation

Each A/D Converter group can activate up to 4 shared service request output signals. 2 shared request signals can issue an interrupt, see **Table 15-5 “Digital Connections in the XMC1100” on Page 15-40**.

Several events can be assigned to each service request output. Service requests can be generated by three types of events:

- **Request source events (SEV):** indicate that the request source completed the requested conversion sequence and the application software can initiate further actions. The event is generated when the complete defined set of channels (pending bits) has been converted.
- **Result events (REV):** indicate a new valid result in the result register. Usually, this triggers a read action by the CPU. Optionally, result events can be generated only at a reduced rate if data reduction is active.

Each ADC event is indicated by a dedicated flag that can be cleared by software. If a service request is enabled for a certain event, the service request is generated for each event, independent of the status of the corresponding event indication flag. This ensures that the ADC event can generate a service request without the need to clear the indication flag.

Event flag registers indicate all types of events that occur during the ADC’s operation. Software can set/clear each flag by writing a 1 to the respective position in register **GLOBEFLAG** to trigger/clear an event. If enabled, service requests are generated for each occurrance of an event, even if the associated flag remains set.

Node Pointer Registers

Requests from each event source can be directed to a set of service request nodes via associated node pointers. Requests from several sources can be directed to the same node; in this case, they are ORed to the service request output signal.

15.6 Registers

The register map of the ADC is built from two blocks, the VADC0 and the SHS0. The corresponding registers, therefore, have an individual offset assigned (see **Table 15-2**). The exact register location is obtained by adding the respective register offset to the base address (see **Table 15-1**) of the corresponding block.

Table 15-1 Registers Address Space

Module	Base Address	End Address	Note
VADC0	$4803\ 0000_H$	$4803\ 03FF_H$	
SHS0	$4803\ 4000_H$	$4803\ 41FF_H$	

Table 15-2 Registers Overview

Register Short Name	Register Long Name	Offset Addr.	Access Mode		Page Num.
			Read	Write	
VADC0 Register					
ID	Module Identification Register	0008_H	U, PV	BE	15-16
CLC	Clock Control Register	0000_H	U, PV	PV	15-17
OCS	OCDS Control and Status Register	0028_H	U, PV	PV	15-18
GLOBCFG	Global Configuration Register	0080_H	U, PV	U, PV	15-20
BRSCTRL	Request Source Control Register	0200_H	U, PV	U, PV	15-23
BRSMR	Request Source Mode Register	0204_H	U, PV	U, PV	15-25
BRSSEL0	Request Source Channel Select Register, Group 0	0180_H	U, PV	U, PV	15-27
BRSSEL1	Request Source Channel Select Register, Group 1	0184_H	U, PV	U, PV	15-27
BRSPND0	Request Source Channel Pending Register, Group 0	$01C0_H$	U, PV	U, PV	15-28
BRSPND1	Request Source Channel Pending Register, Group 1	$01C4_H$	U, PV	U, PV	15-28
GLOBICLASS0	Input Class Register 0	$00A0_H$	U, PV	U, PV	15-29
GLOBICLASS1	Input Class Register 1	$00A4_H$	U, PV	U, PV	15-29
GLOBRCR	Result Control Register	0280_H	U, PV	U, PV	15-31
GLOBRES	Result Register	0300_H	U, PV	U, PV	15-32
GLOBRESD	Result Register (debug view)	0380_H	U, PV	U, PV	15-32

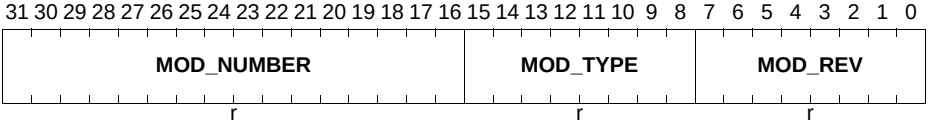
Table 15-2 Registers Overview (cont'd)

Register Short Name	Register Long Name	Offset Addr.	Access Mode		Page Num.
			Read	Write	
GLOBEFLAG	Event Flag Register	$00E0_H$	U, PV	U, PV	15-36
GLOBEVNP	Event Node Pointer Register	0140_H	U, PV	U, PV	15-37
SHS0 Register					
SHS0_ID	SHS Module Identification Register	0008_H	U, SV	U, SV	15-16
SHSCFG	SHS Configuration Register	0040_H	U, SV	U, SV	15-21
GNCTR00	Gain Control Register 0, Group 0	0180_H	U, SV	U, SV	15-33
GNCTR10	Gain Control Register 0, Group 1	0190_H	U, SV	U, SV	15-33
LOOP	SD Loop Control Register	0050_H	U, SV	U, SV	15-35

15.6.1 Module Identification

The module identification register indicates the version of the ADC module that is used in the XMC1100.

ID
Module Identification Register **(0008_H)** **Reset Value: $00C5\ C0XX_H$**
SHS0_ID
Module Identification Register **($4803\ 4008_H$)** **Reset Value: $0099\ C0XX_H$**



Field	Bits	Type	Description
MOD_REV	[7:0]	r	**Module Revision** Indicates the revision number of the implementation. This information depends on the design step.
MOD_TYPE	[15:8]	r	**Module Type** This internal marker is fixed to $C0_H$.
MOD_NUMBER	[31:16]	r	**Module Number** Indicates the module identification number ($00C5_H$ = SARADC, 0099_H = SHS).

15.6.2 System Registers

A set of standardized registers provides general access to the module and controls basic system functions.

The Clock Control Register **CLC** allows the programmer to adapt the functionality and power consumption of the module to the requirements of the application. Register **CLC** controls the module clock signal and the reactivity to the sleep mode signal.

CLC
Clock Control Register **(0000_H)** **Reset Value: 0000 0003_H**

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

Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control of the module. Also the analog section is disabled by clearing ANONS. 0_B On request: enable the module clock 1_B Off request: stop the module clock
DISS	1	r	**Module Disable Status Bit** 0_B Module clock is enabled 1_B Off: module is not clocked
0	2	r	**Reserved, write 0, read as 0**
EDIS	3	rw	**Sleep Mode Enable Control** Used to control module's reaction to sleep mode. 0_B Sleep mode request is enabled and functional 1_B Module disregards the sleep mode control signal
0	[31:4]	r	**Reserved, write 0, read as 0**

The OCDS control and status register OCS controls the module's behavior in suspend mode (used for debugging).

The OCDS Control and Status (OCS) register is cleared by Debug Reset.
The OCS register can only be written when the OCDS is enabled.
If OCDS is being disabled, the OCS register value will not change.
When OCDS is disabled the OCS suspend control is ineffective.
Write access is 32 bit wide only and requires Supervisor Mode.

OCS
OCDS Control and Status Register (0028_H) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**SUS STA**	**SUS _P**	**SUS**				**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
r	r	rh	w	rw				r	r	r	r	r	r	r	r

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

Field	Bits	Type	Description
0	[23:0]	r	**Reserved, write 0, read as 0**
SUS	[27:24]	rw	**OCDS Suspend Control** Controls the sensitivity to the suspend signal coming from the OCDS Trigger Switch (OTGS) 0000_B Will not suspend 0001_B Hard suspend: Clock is switched off immediately. 0010_B Soft suspend mode 0: Stop conversions after the currently running one is completed and its result has been stored. No change for the arbiter. 0011_B Soft suspend mode 1: Stop conversions after the currently running one is completed and its result has been stored. Stop arbiter after the current arbitration round. others: Reserved
SUS_P	28	w	**SUS Write Protection** SUS is only written when SUS_P is 1, otherwise unchanged. Read as 0.

Field	Bits	Type	Description
SUSSTA	29	rh	**Suspend State** 0_B Module is not (yet) suspended 1_B Module is suspended
0	[31:30]	r	**Reserved, write 0, read as 0**

15.6.3 General Registers

The global configuration register provides global control and configuration options that are valid for all converters of the cluster.

GLOBCFG
Global Configuration Register (0080_H) Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SUCAL	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**DPCAL1**	**DPCAL0**
w	r	r	r	r	r	r	r	r	r	r	r	r	r	rw	rw

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

Field	Bits	Type	Description
0	[15:0]	r	**Reserved, write 0, read as 0**
DPCALx (x = 0 - 1)	x+16	rw	**Disable Post-Calibration** 0_B Automatic post-calibration after each conversion of group x 1_B No post-calibration *Note: This bit is only valid for the calibrated converter.*
0	[30:18]	r	**Reserved, write 0, read as 0**
SUCAL	31	w	**Start-Up Calibration** The 0-1 transition of bit SUCAL initiates the start-up calibration phase of all calibrated analog converters. 0_B No action 1_B Initiate the start-up calibration phase (indication in SHS0_SHSCFG.STATE)

SHS0_SHSCFG
SHS Configuration Register **(4803 4040$_H$)** **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STATE				0	0	0	0	0	0	0	0	0	0	SP1	SP0
rh				r	r	r	r	r	r	r	r	r	r	rh	rh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SCWC	ANRDY	0	ANOFF	AREF		0	0	0	0	0	0	0	0	0	0
w	rh	r	rw	rw		r	r	r	r	r	r	r	r	r	r

Field	Bits	Type	Description
0	[9:0]	r	**Reserved, write 0, read as 0**
AREF	[11:10]	rw	**Analog Reference Voltage Selection** 00_B External reference, upper supply range 01_B Reserved 10_B Internal reference, upper supply range 11_B Internal reference, lower supply range
ANOFF	12	rw	**Analog Converter Power Down Force**[1)] 0_B Converter controlled by bitfields ANONS (digital control block) 1_B Converter is permanently off
0	13	r	**Reserved, write 0, read as 0**
ANRDY	14	rh	**Analog Converter Ready** 0_B Converter is in power-down mode 1_B Converter is operable
SCWC	15	w	**Write Control for SHS Configuration** 0_B No write access to SHS configuration 1_B Bitfields ANOFF, AREF can be written
SP0, SP1	16, 17	rh	**Sample Pending on Group x** 0_B No sample pending 1_B Group x has finished the sample phase
0	[27:18]	r	**Reserved, write 0, read as 0**

Field	Bits	Type	Description
STATE	[31:28]	rh	**Current State of Sequencer** 0000_B Idle 0001_B Offset calibration active 0010_B Gain calibration active 0011_B Startup calibration active Others: Normal operation

1) See also **"Analog Converter Control" on Page 15-3**.

15.6.4 Source Registers

Registers of Conversion Request Control

The control register of the request source selects the external gate and/or trigger signals. Write control bits allow separate control of each function with a simple write access.

BRSCTRL
Request Source Control Register

(0200_H) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	GT WC	0	0	GT LVL	GT SEL			
r	r	r	r	r	r	r	r	w	r	r	rh	rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XT WC	XT MODE		XT LVL	XT SEL				0	0	0	0	0	0	0	0
w	rw		rh	rw				r	r	r	r	r	r	r	r

Field	Bits	Type	Description
0	[7:0]	r	**Reserved, write 0, read as 0**
XTSEL	[11:8]	rw	**External Trigger Input Selection** The connected trigger input signals are listed in **Table 15-5 “Digital Connections in the XMC1100” on Page 15-40** *Note: XTSEL = 1111_B uses the selected gate input as trigger source (ENGT must be $0X_B$).*
XTLVL	12	rh	**External Trigger Level** Current level of the selected trigger input
XTMODE	[14:13]	rw	**Trigger Operating Mode** 00_B No external trigger 01_B Trigger event upon a falling edge 10_B Trigger event upon a rising edge 11_B Trigger event upon any edge
XTWC	15	w	**Write Control for Trigger Configuration** 0_B No write access to trigger configuration 1_B Bitfields XTMODE and XTSEL can be written

Field	Bits	Type	Description
GTSEL	[19:16]	rw	**Gate Input Selection** The connected gate input signals are listed in **Table 15-5 “Digital Connections in the XMC1100” on Page 15-40**
GTLVL	20	rh	**Gate Input Level** Current level of the selected gate input
0	[22:21]	r	**Reserved, write 0, read as 0**
GTWC	23	w	**Write Control for Gate Configuration** 0_B No write access to gate configuration 1_B Bitfield GTSEL can be written
0	[31:24]	r	**Reserved, write 0, read as 0**

The Conversion Request Mode Register configures the operating mode of the background request source.

BRSMR
Request Source Mode Register

(0204_H) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
r	r	r	r	r	r	r	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
0	**0**	**0**	**0**	**0**	**0**	**LD EV**	**CLR PND**	**REQ GT**	**0**	**LDM**	**SCA N**	**EN SI**	**EN TR**	**ENGT**	
r	r	r	r	r	r	w	w	rh	r	rw	rw	rw	rw	rw	

Field	Bits	Type	Description
ENGT	[1:0]	rw	**Enable Gate** Selects the gating functionality for source 1. 00_B No conversion requests are issued 01_B Conversion requests are issued if at least one pending bit is set 10_B Conversion requests are issued if at least one pending bit is set and REQGTx = 1. 11_B Conversion requests are issued if at least one pending bit is set and REQGTx = 0. *Note: REQGTx is the selected gating signal.*
ENTR	2	rw	**Enable External Trigger** 0_B External trigger disabled 1_B The selected edge at the selected trigger input signal REQTR generates the load event
ENSI	3	rw	**Enable Source Interrupt** 0_B No request source interrupt 1_B A request source interrupt is generated upon a request source event (last pending conversion is finished)

Field	Bits	Type	Description
SCAN	4	rw	**Autoscan Enable** 0_B No autoscan 1_B Autoscan functionality enabled: a request source event automatically generates a load event
LDM	5	rw	**Autoscan Source Load Event Mode** 0_B Overwrite mode: Copy all bits from the select registers to the pending registers upon a load event 1_B Combine mode: Set all pending bits that are set in the select registers upon a load event (logic OR)
0	6	r	**Reserved, write 0, read as 0**
REQGT	7	rh	**Request Gate Level** Monitors the level at the selected REQGT input. 0_B The gate input is low 1_B The gate input is high
CLRPND	8	w	**Clear Pending Bits** 0_B No action 1_B The bits in registers BRSPNDx are cleared
LDEV	9	w	**Generate Load Event** 0_B No action 1_B A load event is generated
0	[31:10]	r	**Reserved, write 0, read as 0**

The Channel Select Registers select the channels to be converted by the request source. Its bits are used to update the pending registers, when a load event occurs.

The number of valid channel bits depends on the channels available in the respective product type (please refer to **“Product-Specific Configuration” on Page 15-39**).

BRSSELx (x = 0 - 1)
Request Source Channel Select Register, Group x

(0180_H + x * 0004_H) **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
r	r	r	r	r	r	r	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
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**CHSELG7**	**CHSELG6**	**CHSELG5**	**CHSELG4**	**CHSELG3**	**CHSELG2**	**CHSELG1**	**CHSELG0**
r	r	r	r	r	r	r	r	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
CHSELGy (y = 0 - 7)	y	rwh	**Channel Selection Group x** Each bit (when set) enables the corresponding input channel of the respective group to take part in the background scan sequence. 0_B Ignore this channel 1_B This channel is part of the scan sequence
0	[31:8]	r	**Reserved, write 0, read as 0**

The Channel Pending Registers indicate the channels to be converted in the current conversion sequence. They are updated from the select registers, when a load event occurs.

BRSPNDx (x = 0 - 1)
Request Source Pending Register, Group x

($01C0_H$ + x * 0004_H) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
r	r	r	r	r	r	r	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
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**CHPNDG7**	**CHPNDG6**	**CHPNDG5**	**CHPNDG4**	**CHPNDG3**	**CHPNDG2**	**CHPNDG1**	**CHPNDG0**
r	r	r	r	r	r	r	r	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
CHPNDGy (y = 0 - 7)	y	rwh	**Channels Pending Group x** Each bit (when set) request the conversion of the corresponding input channel of the respective group. 0_B Ignore this channel 1_B Request conversion of this channel
0	[31:8]	r	**Reserved, write 0, read as 0**

Note: Writing to any of registers BRSPNDx automatically updates the corresponding register BRSSELx and generates a load event that copies all bits from all registers BRSSELx to BRSPNDx.
Use this shortcut only when writing the last word of the request pattern.

15.6.5 Channel Control Registers

GLOBICLASSy (y = 0 - 1)
Input Class Register y

($00A0_H$ + y * 0004_H) **Reset Value: $0000\ 0000_H$**

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

Field	Bits	Type	Description
STCS	[4:0]	rw	**Sample Time Control for Standard Conversions** Number of additional clock cycles to be added to the minimum sample phase of 2 analog clock cycles: Coding and resulting sample time see **Table 15-3**.
0	[7:5]	r	**Reserved, write 0, read as 0**
CMS	[10:8]	rw	**Conversion Mode for Standard Conversions** 000_B 12-bit conversion 001_B 10-bit conversion 010_B 8-bit conversion 011_B Reserved 100_B Reserved 101_B 10-bit fast compare mode 110_B Reserved 111_B Reserved
0	[31:11]	r	**Reserved, write 0, read as 0**

Table 15-3 Sample Time Coding

STCS	Additional Clock Cycles	Sample Time
$0\ 0000_B$	0	2 / f_{ADCI}
$0\ 0001_B$	1	3 / f_{ADCI}
…	…	…

Table 15-3 Sample Time Coding (cont'd)

STCS	Additional Clock Cycles	Sample Time
$0\ 1111_B$	15	17 / f_{ADCI}
$1\ 0000_B$	16	18 / f_{ADCI}
$1\ 0001_B$	32	34 / f_{ADCI}
…	…	…
$1\ 1110_B$	240	242 / f_{ADCI}
$1\ 1111_B$	256	258 / f_{ADCI}

15.6.6 Result Register

The result control register selects the behavior of the result register.

GLOBRCR
Result Control Register **(0280_H)** **Reset Value: $0000\ 0000_H$**

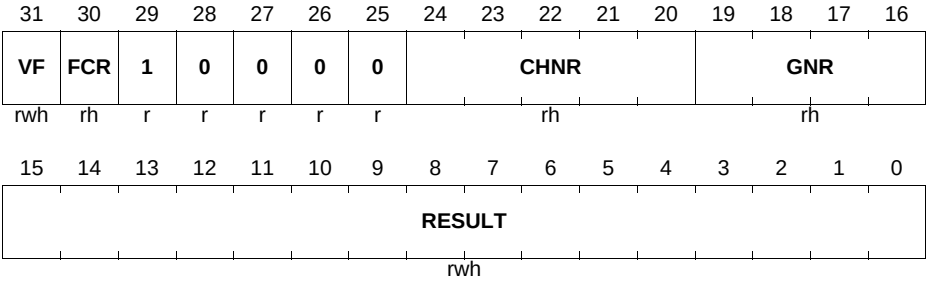
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SRGEN	**0**	**0**	**0**	**0**	**0**	**0**	**WFR**	**0**	**0**	**0**	**0**	**DRCTR**			
rw	r	r	r	r	r	r	rw	r	r	r	r	rw			

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

Field	Bits	Type	Description
0	[15:0]	r	**Reserved, write 0, read as 0**
DRCTR	[19:16]	rw	**Data Reduction Control** Defines how result values are stored/accumulated in this register for the final result. The data reduction counter DRC can be loaded from this bitfield. 0000_BData reduction disabled 0001_BAccumulate 2 result values 0010_BAccumulate 3 result values 0011_BAccumulate 4 result values others: Reserved
0	[23:20]	r	**Reserved, write 0, read as 0**
WFR	24	rw	**Wait-for-Read Mode Enable** 0_B Overwrite mode 1_B Wait-for-read mode enabled for this register
0	[30:25]	r	**Reserved, write 0, read as 0**
SRGEN	31	rw	**Service Request Generation Enable** 0_B No service request 1_B Service request after a result event

The result register provides a common storage location for all channels of all groups.

GLOBRES
Result Register, Application view **(0300_H)** **Reset Value: 0000 0000_H**
GLOBRESD
Result Register, Debug view **(0380_H)** **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
RESULT	[15:0]	rwh	**Result of most recent conversion** The position of the result bits within this bitfield depends on the configured operating mode.[1)] Please, refer to **Section 15.4.1.1**.
GNR	[19:16]	rh	**Group Number** Indicates the group to which the channel number in bitfield CHNR refers.
CHNR	[24:20]	rh	**Channel Number** Indicates the channel number corresponding to the value in bitfield RESULT.
0	[28:25]	r	**Reserved, read as 0**
1	29	r	**Reserved, read as 1**
FCR	30	rh	**Fast Compare Result** Indicates the result of an operation in Fast Compare Mode. 0_B Signal level was below compare value 1_B Signal level was above compare value

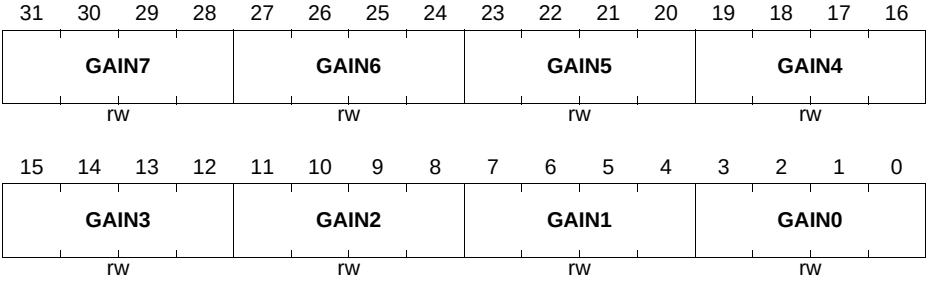
Field	Bits	Type	Description
VF	31	rwh	**Valid Flag** Indicates a new result in bitfield RESULT or bit FCR. 0_B Read access: No new valid data available Write access: No effect 1_B Read access: Bitfield RESULT contains valid data and has not yet been read, or bit FCR has been updated Write access: Clear this valid flag and the data reduction counter (overrides a hardware set action)[1)]

1) Only writable in register GLOBRES, not in register GLOBRESD.

15.6.7 Gain Control Register

The gain control registers GNCTRx0 selects the gain level for each input.

SHS0_GNCTR00
Gain Control Register 00 **(4803 4180$_H$)** **Reset Value: 0000 0000$_H$**
SHS0_GNCTR10
Gain Control Register 10 **(4803 4190$_H$)** **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
GAINz **(z = 0 - 7)**	[z*4+3: z*4]	rw	**Gain Control z** 0000_BGain factor = 1 0001_BGain factor = 3 0010_BGain factor = 6 0011_BGain factor = 12 Others: Reserved

Note: The first index (x0) indicates the associated group.
The channel number is z.

15.6.8 Miscellaneous Registers

The sigma-delta-loop control register LOOP configures the functionality of the sigma-delta-loop(s).

SHS0_LOOP
Loop Control Register **(4803 4050$_H$)** **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LP EN1	**0**	**0**	**0**	**0**	**0**	**0**	**LP SH1**	**0**	**0**	**0**	**LPCH1**				
rw	r	r	r	r	r	r	rw	r	r	r	rw				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LP EN0	**0**	**0**	**0**	**0**	**0**	**0**	**LP SH0**	**0**	**0**	**0**	**LPCH0**				
rw	r	r	r	r	r	r	rw	r	r	r	rw				

Field	Bits	Type	Description
LPCH0, LPCH1	[4:0], [20:16]	rw	**Loop y Channel** Selects the input channel, for which the sigma-delta-loop function shall be enabled.
0	[7:5]	r	**Reserved, write 0, read as 0**
LPSH0, LPSH1	8, 24	rw	**Loop y Group Select** Selects the group, to which the indicated channel is assigned.
0	[14:9]	r	**Reserved, write 0, read as 0**
LPEN0, LPEN1	15, 31	rw	**Loop y Enable** 0_H Off: standard operation 1_H ON: sigma-delta-loop is active
0	[23:21]	r	**Reserved, write 0, read as 0**
0	[30:25]	r	**Reserved, write 0, read as 0**

Note: Bitfields LPSHy and LPCHy together select one individual input channel that is associated with the corresponding sigma-delta loop.

15.6.9 Service Request Registers

GLOBEFLAG
Event Flag Register **($00E0_H$)** **Reset Value: $0000\ 0000_H$**

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

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

Field	Bits	Type	Description
SEVGLB	0	rwh	**Source Event** 0_B No source event 1_B A source event has occurred
0	[7:1]	r	**Reserved, write 0, read as 0**
REVGLB	8	rwh	**Result Event** 0_B No result event 1_B New result was stored in register GLOBRES
0	[15:9]	r	**Reserved, write 0, read as 0**
SEVGLBCLR	16	w	**Clear Source Event** 0_B No action 1_B Clear the source event flag SEVGLB
0	[23:17]	r	**Reserved, write 0, read as 0**
REVGLBCLR	24	w	**Clear Result Event** 0_B No action 1_B Clear the result event flag REVGLB
0	[31:25]	r	**Reserved, write 0, read as 0**

Note: Software can set flags REVGLB and SEVGLB and trigger the corresponding event by writing 1 to the respective bit. Writing 0 has no effect.
Software can clear these flags by writing 1 to bit REVGLBCLR and SECGLBCLR, respectively.

Setting both bits (e.g. SEVGLB and SEVGLBCLR) simultaneously clears the corresponding flag (e.g. SEVGLB).

GLOBEVNP
Event Node Pointer Register

(0140$_H$) **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**REV0NP**			
r	r	r	r	r	r	r	r	r	r	r	r	rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**SEV0NP**			
r	r	r	r	r	r	r	r	r	r	r	r	rw			

Field	Bits	Type	Description
SEV0NP	[3:0]	rw	**Service Request Node Pointer Source Event** Routes the corresponding event trigger to one of the service request lines (nodes). 0000_B Select shared service request line 0 of common service request group 0 … 0011_B Select shared service request line 3 of common service request group 0 others: Reserved *Note: For shared service request lines see **Table 15-5**.*
0	[15:4]	r	**Reserved, write 0, read as 0**

Field	Bits	Type	Description
REV0NP	[19:16]	rw	**Service Request Node Pointer Result Event** Routes the corresponding event trigger to one of the service request lines (nodes). 0000_B Select shared service request line 0 of common service request group 0 … 0011_B Select shared service request line 3 of common service request group 0 others: Reserved *Note: For shared service request lines see **Table 15-5**.*
0	[31:20]	r	**Reserved, write 0, read as 0**

15.7 Interconnects

This section describes the actual implementation of the VADC module into the XMC1100, i.e. the incorporation into the microcontroller system.

15.7.1 Product-Specific Configuration

The functional description describes the features and operating modes of the A/D Converters in a general way. This section summarizes the configuration that is available in this product (XMC1100).

15.7.2 Analog Module Connections in the XMC1100

The VADC module accepts a number of analog input signals. The analog input multiplexers select the input channels from the signals available in this product.

The exact number of analog input channels and the available connection to port pins depend on the employed product type. A summary of channels enclosing all versions of the XMC1100 can be found in **Table 15-4**.

Table 15-4 Analog Connections in the XMC1100

Signal	Dir.	Source/Destin.	Description
Reference Voltage Pins			
V_{AREF}	I	VDD	positive analog reference
V_{AGND}	I	VSS	negative analog reference
Group 0 Analog Inputs, controlled by input class 0			
G0CH0	I	P2.6	analog input channel 0 of group 0
G0CH1	I	P2.8	analog input channel 1 of group 0
G0CH2	I	P2.9	analog input channel 2 of group 0
G0CH3	I	P2.10	analog input channel 3 of group 0
G0CH4	I	P2.11	analog input channel 4 of group 0
G0CH5	I	P2.0	analog input channel 5 of group 0
G0CH6	I	P2.1	analog input channel 6 of group 0
G0CH7	I	P2.2	analog input channel 7 of group 0
Group 1 Analog Inputs, controlled by input class 1			
G1CH1	I	P2.7	analog input channel 1 of group 1
G1CH5	I	P2.3	analog input channel 5 of group 1
G1CH6	I	P2.4	analog input channel 6 of group 1
G1CH7	I	P2.5	analog input channel 7 of group 1

15.7.3 Digital Module Connections in the XMC1100

The VADC module accepts a number of digital input signals and generates a number of output signals. This section summarizes the connection of these signals to other on-chip modules or to external resources via port pins.

Note: The control bitfields for triggers and gates select the corresponding multiplexer input. Values 0000_B … 1111_B select inputs with suffix -A … -P.

Table 15-5 Digital Connections in the XMC1100

Signal	Dir.	Source/Destin.	Description
Gate Inputs			
BGREQGTA	I	CCU40.ST3	Gating input A
BGREQGTB	I	CCU40.ST2	Gating input B
BGREQGTC	I	CCU40.ST1	Gating input C
BGREQGTD	I	CCU40.ST0	Gating input D
BGREQGTE	I	0	Gating input E
BGREQGTF	I	0	Gating input F
BGREQGTG	I	0	Gating input G
BGREQGTH	I	0	Gating input H
BGREQGTI	I	0	Gating input I
BGREQGTJ	I	0	Gating input J
BGREQGTK	I	ERU0.PDOUT2	Gating input K
BGREQGTL	I	ERU0.PDOUT3	Gating input L
BGREQGTM	I	0	Gating input M
BGREQGTN	I	0	Gating input N
BGREQGTO	I	ERU0.PDOUT0	Gating input O
BGREQGTP	I	ERU0.PDOUT1	Gating input E
BGREQGTSEL	O	BGREQTRP[1)]	Selected gating signal
Trigger Inputs			
BGREQTRA	I	CCU40.SR2	Trigger input A
BGREQTRB	I	CCU40.SR3	Trigger input B
BGREQTRC	I	0	Trigger input C
BGREQTRD	I	0	Trigger input D
BGREQTRE	I	0	Trigger input E

Table 15-5 Digital Connections in the XMC1100 (cont'd)

Signal	Dir.	Source/Destin.	Description
BGREQTRF	I	0	Trigger input F
BGREQTRG	I	ERU0.IOUT2	Trigger input G
BGREQTRH	I	ERU0.IOUT3	Trigger input H
BGREQTRI	I	0	Trigger input I
BGREQTRJ	I	0	Trigger input J
BGREQTRK	I	0	Trigger input K
BGREQTRL	I	0	Trigger input L
BGREQTRM	I	ERU0.IOUT0	Trigger input M
BGREQTRN	I	ERU0.IOUT1	Trigger input N
BGREQTRO	I	0	Trigger input O
BGREQTRP	I	BGREQGTSEL[1)]	Extend triggers to selected gating input
System-Internal Connections			
C0SR0	O	NVIC (15)	Service request 0 of common block 0
C0SR1	O	NVIC (16)	Service request 1 of common block 0
C0SR2	O	ERU0.OGU02 ERU0.OGU12	Service request 2 of common block 0
C0SR3	O	ERU0.OGU22 ERU0.OGU32	Service request 3 of common block 0

1) Internal signal connection.

16 Capture/Compare Unit 4 (CCU4)

The CCU4 peripheral is a major component for systems that need general purpose timers for signal monitoring/conditioning and Pulse Width Modulation (PWM) signal generation. Power electronic control systems like switched mode power supplies or uninterruptible power supplies, can easily be implemented with the functions inside the CCU4 peripheral.

The internal modularity of CCU4, translates into a software friendly system for fast code development and portability between applications.

Table 16-1 Abbreviations table

PWM	Pulse Width Modulation
CCU4x	Capture/Compare Unit 4 module instance x
CC4y	Capture/Compare Unit 4 Timer Slice instance y
ADC	Analog to Digital Converter
POSIF	Position Interface peripheral
SCU	System Control Unit
f_{ccu4}	CCU4 module clock frequency
f_{tclk}	CC4y timer clock frequency

Note: A small “y” or “x” letter in a register indicates an index

16.1 Overview

Each CCU4 module is comprised of four identical 16 bit Capture/Compare Timer slices, CC4y. Each timer slice can work in compare mode or in capture mode. In compare mode one compare channel is available while in capture mode, up to four capture registers can be used in parallel.

Each CCU4 module has four service request lines and each timer slice contains a dedicated output signal, enabling the generation of up to four independent PWM signals.

Straightforward timer slice concatenation is also possible, enabling up to 64 bit timing operations. This offers a flexible frequency measurement, frequency multiplication and pulse width modulation scheme.

A programmable function input selector for each timer slice, that offers up to nine functions, discards the need of complete resource mapping due to input ports availability.

A built-in link between the CCU4 and several other modules enable flexible digital motor control loops implementation, e.g. with Hall Sensor monitoring or direct coupling with Encoders.

16.1.1 Features

CCU4 module features

Each CCU4 represents a combination of four timer slices, that can work independently in compare or capture mode. Each timer slice has a dedicated output for PWM signal generation.

All four CCU4 timer slices, CC4y, are identical in terms of available functions and operating modes. Avoiding this way the need of implementing different software routines, depending on which resource of CCU4 is used.

A built-in link between the four timer slices is also available, enabling this way a simplified timer concatenation and sequential operations.

General Features

- 16 bit timer cells
- capture and compare mode for each timer slice
  - four capture registers in capture mode
  - one compare channel in compare mode
- programmable low pass filter for the inputs
- built-in timer concatenation
  - 32, 48 or 64 bit width
- shadow transfer for the period and compare values
- programmable clock prescaler
- normal timer mode
- gated timer mode
- three counting schemes
  - center aligned
  - edge aligned
  - single shot
- PWM generation
- TRAP function
- start/stop can be controlled by external events
- counting external events
- four dedicated service request lines per CCU4

Additional features

- external modulation function
- load controlled by external events
- dithering PWM
- floating point pre scaler
- output state override by an external event
- suitable and flexible connectivity to several modules:
  - motor and power conversion applications
  - high number of signal conditioning possibilities

CCU4 features vs. applications

On **Table 16-2** a summary of the major features of the CCU4 unit mapped with the most common applications.

Table 16-2 Applications summary

Feature	Applications
Four independent timer cells	Independent PWM generation: • Multiple buck/boost converter control (with independent frequencies) • Different modes of operation for each timer, increasing the resource optimization • Up to 2 Half-Bridges control • multiple Zero Voltage Switch (ZVS) converter control with easy link to the ADC channels.
Concatenated timer cells	Easy to configure timer extension up to 64 bit: • High dynamic trigger capturing • High dynamic signal measurement
Dithering PWM	Generating a fractional PWM frequency or duty cycle: • To avoid big steps on frequency or duty cycle adjustment in slow control loop applications • Increase the PWM signal resolution over time
Floating prescaler	Automated control signal measurement: • decrease SW activity for monitoring signals with high or unknown dynamics • emulating more than a 16 bit timer for system control
Up to 9 functions via external signals for each timer	Flexible resource optimization: • The complete set of external functions is always available • Several arrangements can be done inside a CCU4, e.g., one timer working in capture mode and one working in compare

Table 16-2 Applications summary (cont'd)

Feature	Applications
4 dedicated service request lines	Specially developed for: • generating interrupts for the microprocessor • flexible connectivity between peripherals, e.g. ADC triggering.
Linking with other modules	Flexible profiles for: • Hall Sensor feedback/monitoring • Motor Encoders feedback/monitoring • PWM parallel modulation • Flexible signal conditioning

16.1.2 Block Diagram

Each CCU4 timer slice can operate independently from the other slices for all the available modes. Each timer slice contains a dedicated input selector for functions linked with external events and has a dedicated compare output signal, for PWM signal generation.

The built-in timer concatenation is only possible with adjacent slices, e.g. CC40/CC41. Combinations for slice concatenations like, CC40/CC42 or CC40/CC43 are not possible.

The individual service requests for each timer slice (four per slice) are multiplexed into four module service requests lines, **Figure 16-1**.

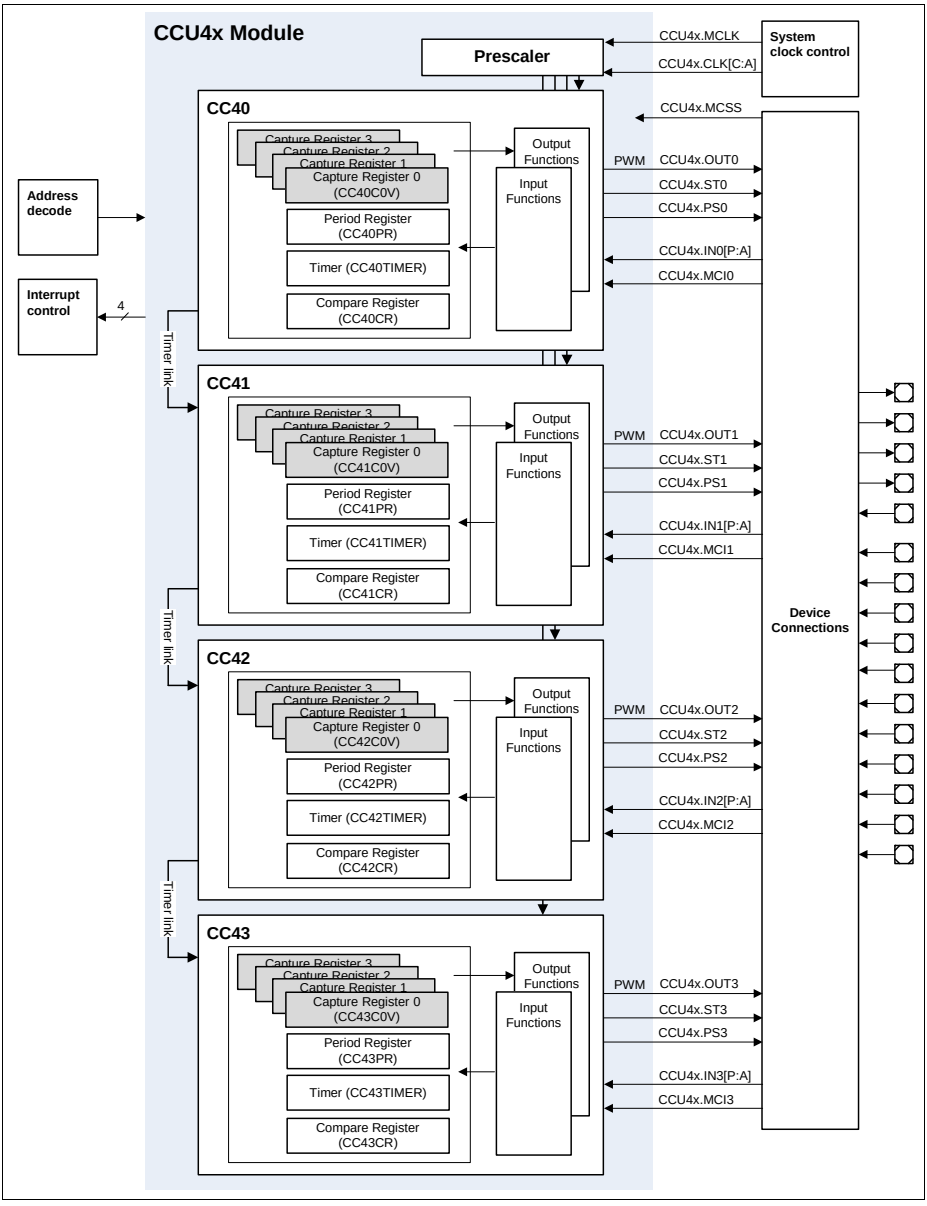


Figure 16-1 CCU4 block diagram

16.2 Functional Description

16.2.1 CC4y Overview

The input path of a CCU4 slice is comprised of a selector (**Section 16.2.2**) and a connection matrix unit (**Section 16.2.3**). The output path contains a service request control unit, a timer concatenation unit and two units that control directly the state of the output signal for each specific slice (for TRAP and modulation handling), see **Figure 16-2**.

The timer core is built of a 16 bit counter one period and one compare register in compare mode, or up to four capture registers in capture mode.

In compare mode the period register sets the maximum counting value while the compare channel is controlling the ACTIVE/PASSIVE state of the dedicated comparison slice output.

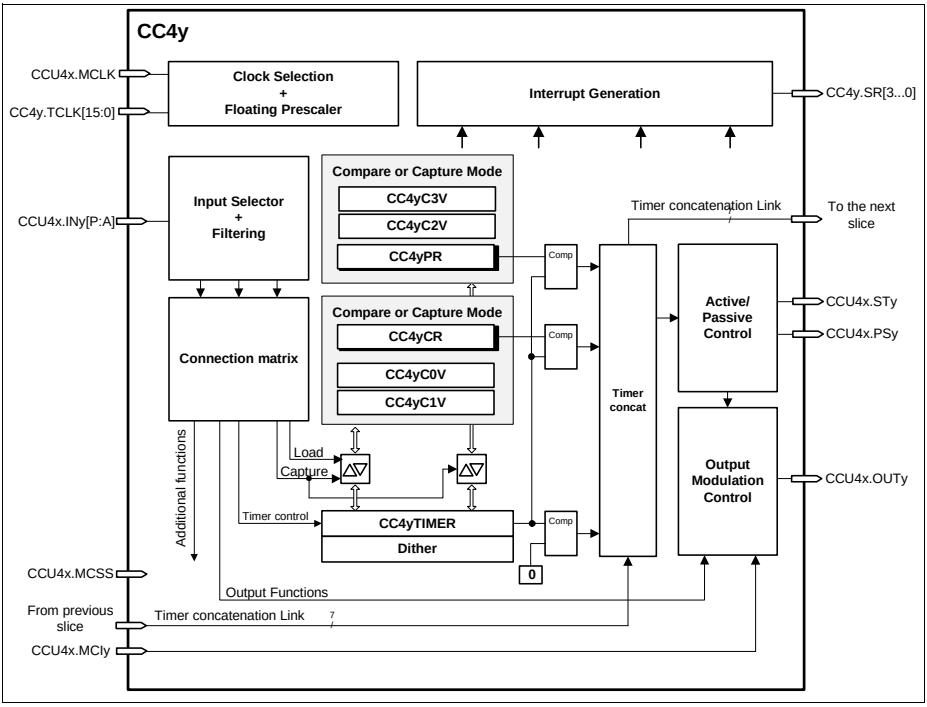


Figure 16-2 CCU4 slice block diagram

Each CCU4 slice, with the exception of the first, contains six dedicated inputs outputs that are used for the built-in timer concatenation functionality.

Inputs and outputs that are not seen at the CCU4 boundaries have a nomenclature of CC4y.<name>, whilst CCU4 module inputs and outputs are described as CCU4x.<signal_name>y (indicating the variable y the object slice).

Table 16-3 CCU4 slice pin description

Pin	I/O	Description
CCU4x.MCLK	I	Module clock
CC4y.TCLK[15:0]	I	Clocks from the pre scaler
CCU4x.INy[P:A]	I	Slice functional inputs (used to control the functionality throughout slice external events)
CCU4x.MCIy	I	Multi Channel mode input
CCU4x.MCSS	I	Multi Channel shadow transfer trigger
CC4y.SR[3...0]	O	Slice service request lines
CC4x.STy	O	Slice comparison status value
CCU4x.PSy	O	Multi channel pattern update trigger
CCU4x.OUTy	O	Slice dedicated output pin

Note:

4. *The status bit outputs of the Kernel, CCU4x.STy, are extended for one more kernel clock cycle.*
5. *The Service Request signals at the output of the kernel are extended for one more kernel clock cycle.*
6. *The maximum output signal frequency of the CCU4x.STy outputs is module clock divided by 4.*

The slice timer, can count up or down depending on the selected operating mode. A direction flag holds the actual counting direction.

The timer is connected to two stand alone comparators, one for the period match and one for a compare match. The registers used for period match and comparison match can be programmed to serve as capture registers enabling sequential capture capabilities on external events.

In normal edge aligned counting scheme, the counter is cleared to 0000_H each time that matches the period value defined in the period register. In center aligned mode, the counter direction changes from ‘up counting’ to ‘down counting’ after reaching the period

value. Both period and compare registers have an aggregated shadow register, which enables the update of the PWM period and duty cycle on the fly.

A single shot mode is also available, where the counter stops after it reaches the value set in the period register.

The start and stop of the counter can be controlled via software access or by a programmable input pin.

Functions like, load, counting direction (up/down), TRAP and output modulation can also be controlled with external events, see **Section 16.2.3**.

16.2.2 Input Selector

The first unit of the slice input path, is used to select which inputs are used to control the available external functions.

Inside this block the user also has the possibility to perform a low pass filtering of the signals and selecting the active edge(s) or level of the external event, see **Figure 16-3**.

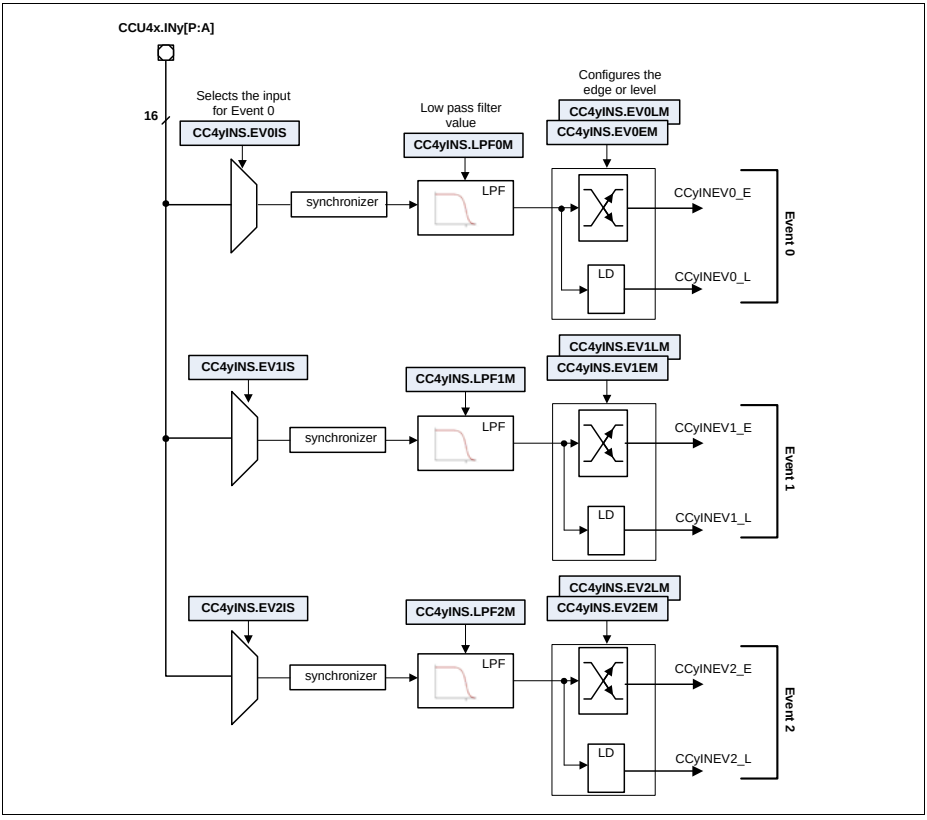


Figure 16-3 Slice input selector diagram

The user has the possibility of selecting any of the CCU4x.INy[P:A] inputs as the source of an event.

At the output of this unit we have a user selection of three events, that were configured to be active at rising, falling or both edges, or level active. These selected events can then be mapped to several functions.

Notice that each decoded event contains two outputs, one edge active and one level active, due to the fact that some functions like counting, capture or load are edge sensitive events while, timer gating or up down counting selection are level active.

16.2.3 Connection Matrix

The connection matrix maps the events coming from the input selector to several user configured functions, **Figure 16-4**.The following functions can be enabled on the connection matrix:

Table 16-4 Connection matrix available functions

Function	Brief description	Map to figure Figure 16-4
Start	Edge signal to start the timer	CCystrt
Stop	Edge signal to stop the timer	CCystp
Count	Edge signal used for counting events	CCycnt
Up/down	Level signal used to select up or down counting direction	CCyupd
Capture 0	Edge signal that triggers a capture into the capture registers 0 and 1	CCycapt0
Capture 1	Edge signal that triggers a capture into the capture register 2 and 3	CCycapt1
Gate	Level signal used to gate the timer clock	CCygate
Load	Edge signal that loads the timer with the value present at the compare register	CCyload
TRAP	Level signal used for fail-safe operation	CCytrap
Modulation	Level signal used to modulate/clear the output	CCymod
Status bit override	Status bit is going to be overridden with an input value	CCyoval for the value CCyoset for the trigger

Inside the connection matrix we also have a unit that performs the built-in timer concatenation. This concatenation enables a completely synchronized operation between the concatenated slices for timing operations and also for capture and load actions. The timer slice concatenation is done via the **CC4yCMC**.TCE bitfield. For a complete description of the concatenation function, please address **Section 16.2.9**.

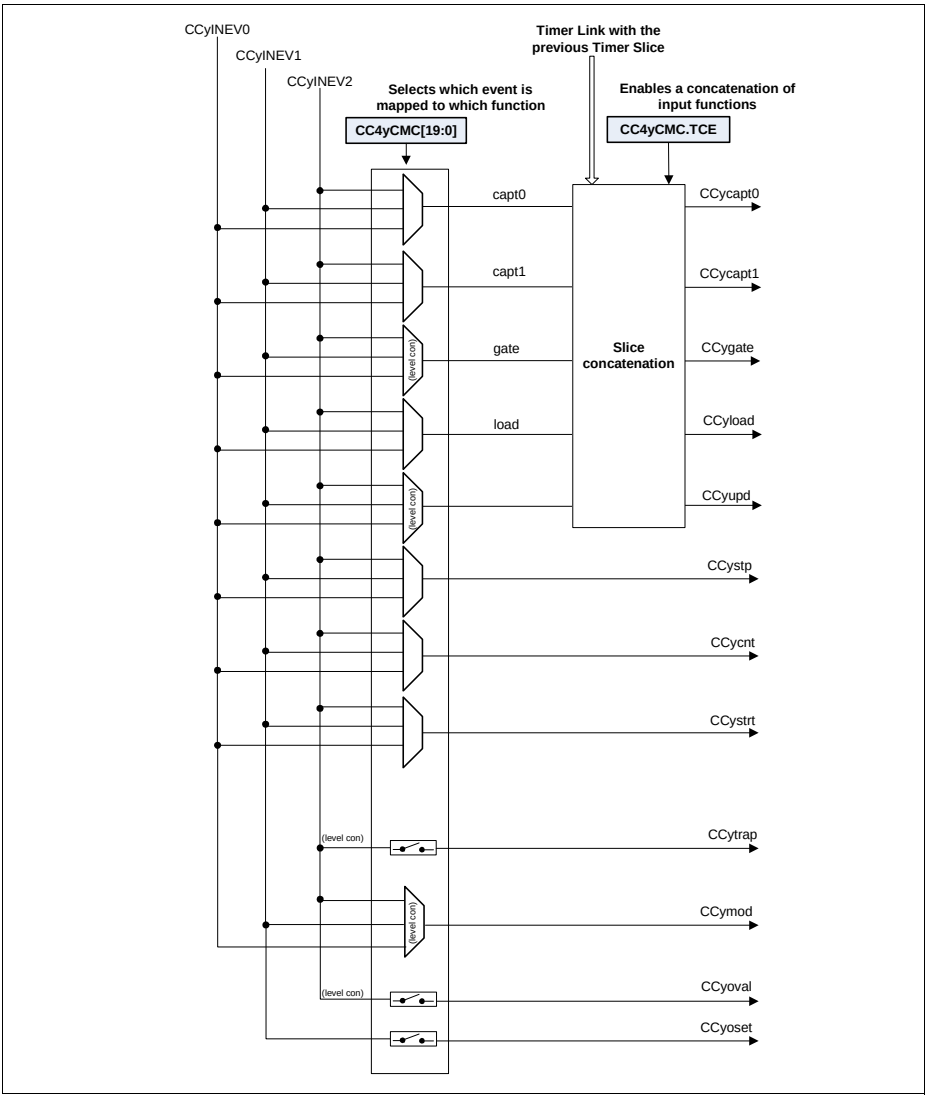


Figure 16-4 Slice connection matrix diagram

16.2.4 Starting/Stopping the Timer

Each timer slice contains a run bit register that indicates the actual status of the timer, **CC4yTCST**.TRB. The start and stop of the timer can be done via software access or can be controlled directly by external events, see **Figure 16-5**.

Selecting an external signal that acts as a start trigger does not force the user to use an external stop trigger and vice versa.

Selecting the single shot mode, imposes that after the counter reaches the period value the run bit, **CC4yTCST**.TRB, is going to be cleared and therefore the timer is stopped.

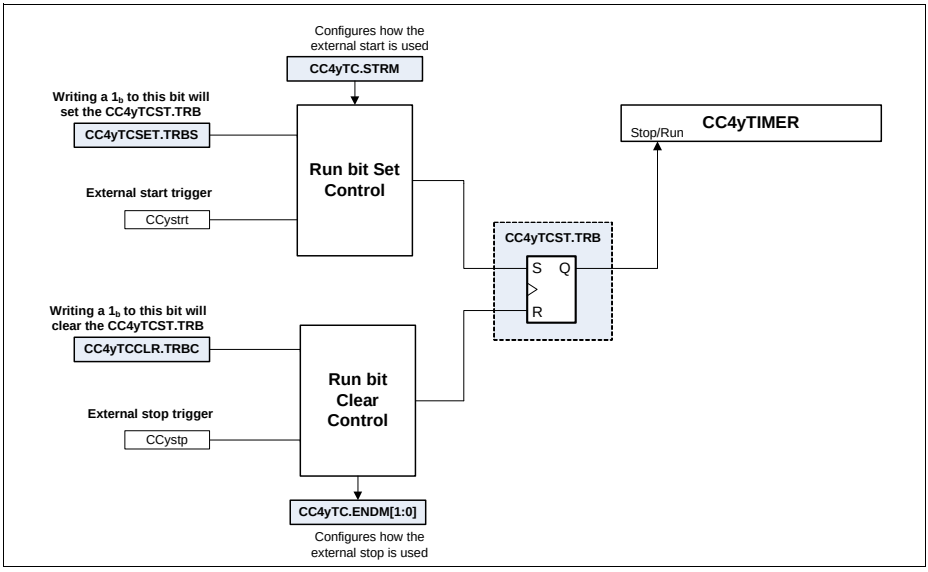


Figure 16-5 Timer start/stop control diagram

One can use the external stop signal to perform the following functions (configuration via **CC4yTC**.ENDM):

- Clear the run bit (stops the timer) - default
- Clear the timer (to 0000_H) but it does not clear the run bit (timer still running)
- Clear the timer and the run bit

One can use the external start to perform the following functions (configuration via **CC4yTC**.STRM):

- Start the timer (resume operation)
- Clear and start the timer

The set (start the timer) of the timer run bit, always has priority over a clear (stop the timer).

To start multiple CCU4 timers at the same time/synchronously one should use a dedicated input as external start (see **Section 16.2.7.1** for a description how to configure an input as start function). This input should be connected to all the Timers that need to started synchronously (see **Section 16.8** for a complete list of module connections), **Figure 16-6**.

For starting the timers synchronously via software there is a dedicated input signal, controlled by the SCU (System Control Unit), that is connected to all the CCU4 timers. This signal should then be configured as an external start signal (see **Section 16.2.7.1**) and then the software must writea 1_B to the specific bitfield of the CCUCON register (this register is described on the SCU chapter).

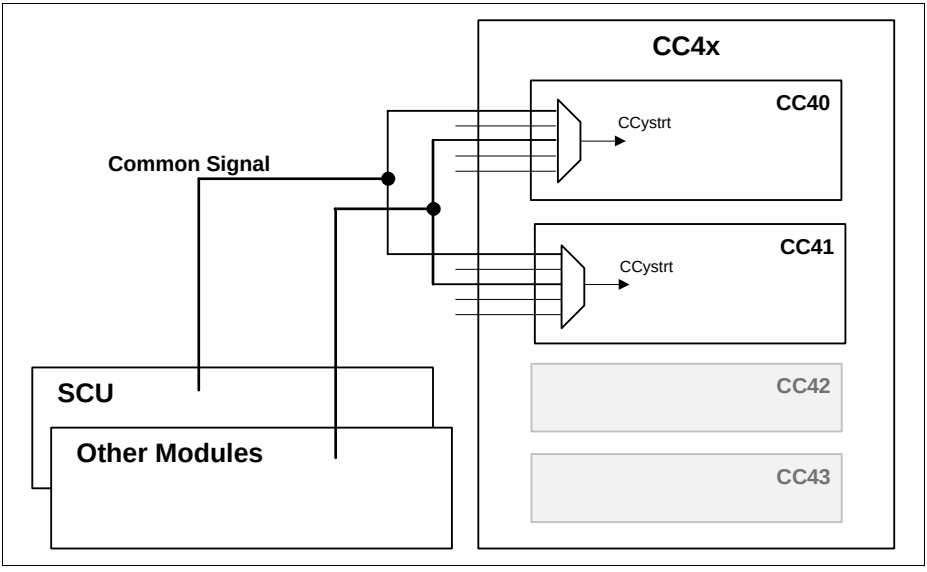


Figure 16-6 Starting multiple timers synchronously

16.2.5 Counting Modes

Each CC4y timer slice can be programmed into three different counting schemes:

- Edge aligned (default)
- Center aligned
- Single shot (can be edge or center aligned)

These three counting schemes can be used as stand alone without the need of selecting any inputs as external event sources. Nevertheless it is also possible to control the counting operation via external events like, timer gating, counting trigger, external stop, external start, etc.

For all the counting modes, it is possible to update on the fly the values for the timer period and compare channel. This enables a cycle by cycle update of the PWM frequency and duty cycle.

The compare channel of each CC4y Timer Slice, has an associated Status Bit (**GCST**.CC4yST), that indicates the active or passive state of the channel, **Figure 16-7**. The set and clear of the status bit and the respective PWM signal generation is dictated by the timer period, compare value and the current counting mode. See the different counting mode descriptions, **Section 16.2.5.3** to **Section 16.2.5.5** to understand how this bit is set and cleared.

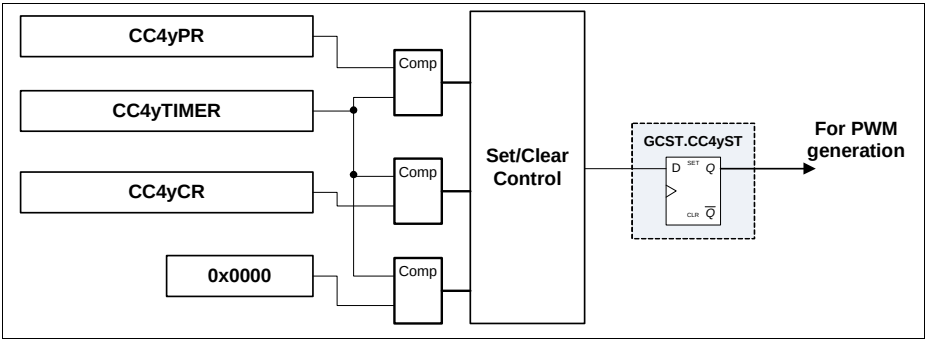


Figure 16-7 CC4y Status Bit

16.2.5.1 Calculating the PWM Period and Duty Cycle

The period of the timer is determined by the value in the period register, **CC4yPR** and by the timer mode.

The base for the PWM signal frequency and duty cycle, is always related to the clock frequency of the timer itself and not to the frequency of the module clock (due to the fact that the timer clock can be a scaled version of the module clock).

In Edge Aligned Mode, the timer period is:

$$T_{per} = \text{<Period-Value>} + 1; \text{ in } f_{tclk} \quad (16.1)$$

In Center Aligned Mode, the timer period is:

$$T_{per} = (\text{<Period-Value>} + 1) \times 2; \text{ in } f_{tclk} \quad (16.2)$$

For each of these counting schemes, the duty cycle of generated PWM signal is dictated by the value programmed into the **CC4yCR** register.

In Edge Aligned and Center Aligned Mode, the PWM duty cycle is:

$$DC = 1 - \text{<Compare-Value>}/(\text{<Period-Value>} + 1) \quad (16.3)$$

Both **CC4yPR** and **CC4yCR** can be updated on the fly via software, enabling a glitch free transition between different period and duty cycle values for the generated PWM signal, **Section 16.2.5.2**

16.2.5.2 Updating the Period and Duty Cycle

Each CCU4 timer slice provides an associated shadow register for the period and compare values. This facilitates a concurrent update by software for these two parameters, with the objective of modifying during run time the PWM signal period and duty cycle.

In addition to the shadow registers for the period and compare values, one also has available shadow registers for the floating prescaler and dither functions, **CC4yFPCS** and **CC4yDITS** respectively (please address **Section 16.2.11** and **Section 16.2.10** for a complete description of these functions).

The structure of the shadow registers can be seen in **Figure 16-8**.

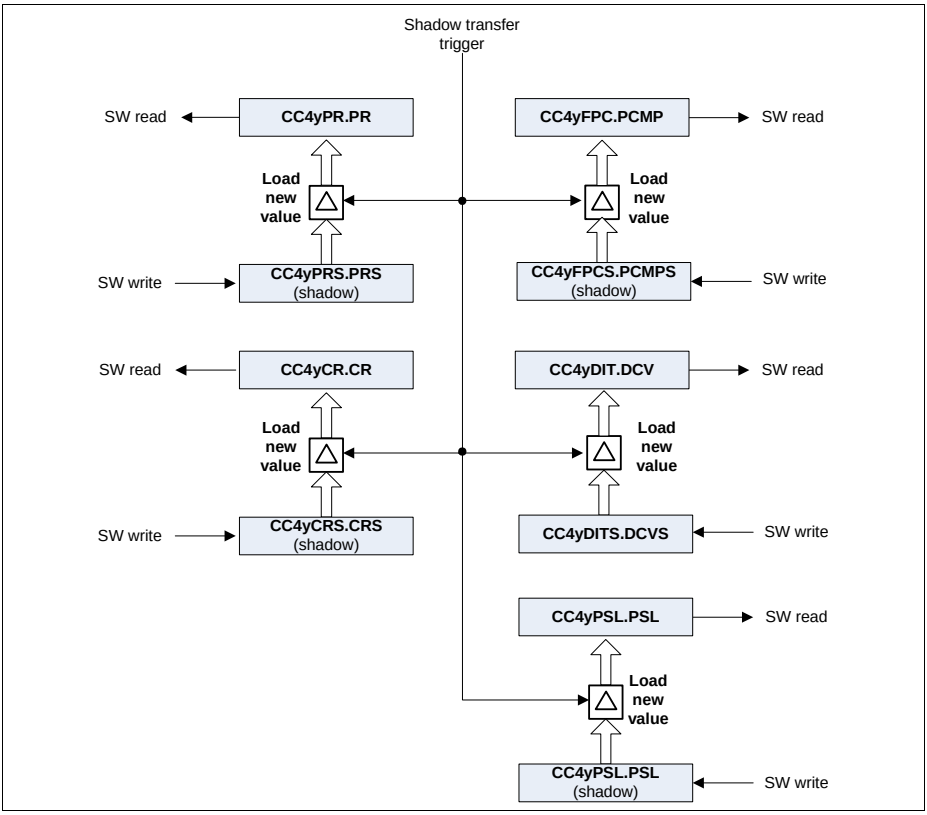


Figure 16-8 Shadow registers overview

The update of these registers can only be done by writing a new value into the associated shadow register and wait for a shadow transfer to occur.

Each group of shadow registers have an individual shadow transfer enable bit, **Figure 16-9**. The software must set this enable bit to 1_B, whenever an update of the values is needed. These bits are automatically cleared by the hardware, whenever an update of the values if finished. Therefore every time that an update of the registers is needed the software must set again the specific bit(s).

Nevertheless it is also possible to clear the enable bit via software. This can be used in the case that an update of the values needs to be cancelled (after the enable bit has already been set).

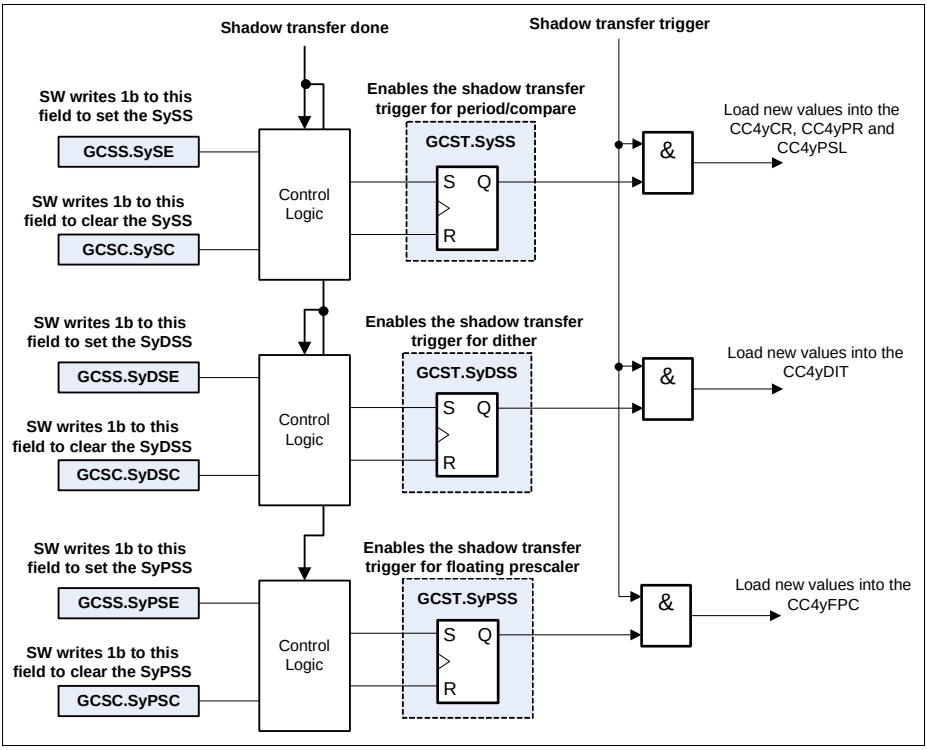


Figure 16-9 Shadow transfer enable logic

The shadow transfer operation is going to be done in the immediately next occurrence of a shadow transfer trigger, after the shadow transfer enable is set (**GCST**.SySS, **GCST**.SyDSS, **GCST**.SyPSS set to 1_B).

The occurrence of the shadow transfer trigger is imposed by the timer counting scheme (edge aligned or center aligned). Therefore the slots when the values are updated can be:

- in the next clock cycle after a Period Match while counting up
- in the next clock cycle after an One Match while counting down
- immediately, if the timer is stopped and the shadow transfer enable bit(s) is set

Figure 16-10 shows an example of the shadow transfer control when the timer slice has been configured into center aligned mode. For a complete description of all the timer slice counting modes, please address **Section 16.2.5.3**, **Section 16.2.5.4** and **Section 16.2.5.5**.

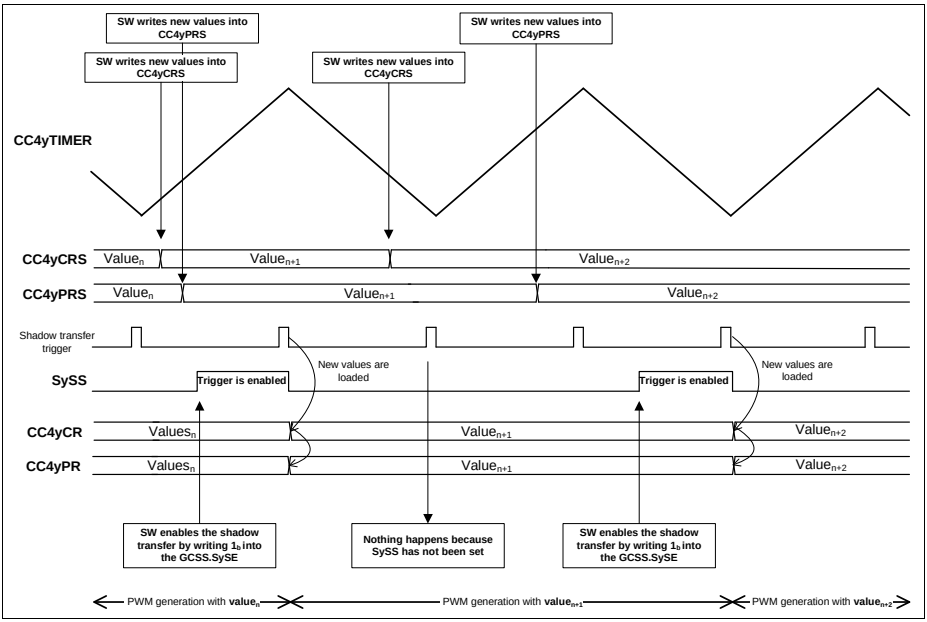


Figure 16-10 Shadow transfer timing example - center aligned mode

In some application cases it may be necessary to request shadow transfers not by software but by hardware. To perform this action each CCU4 contains a dedicated input that can be used to request a shadow transfer by hardware, the CCU4x.MCSS.

This input, when enabled, is used to set the shadow transfer enable bitfields (**GCST**.SySS, **GCST**.SyDSS and **GCST**.SyPSS) of the specific slice. It is possible to select which slice is using this input to perform the synchronization via the **GCTRL**.MSEy bit field. It is also possible to enable the usage of this signal for the three different shadow transfer signals: compare and period values, dither compare value and prescaler compare value. This can be configured on the **GCTRL**.MSDE field.

The structure for using the CCU4x.MCSS input signal can be seen in **Figure 16-9**. The usage of this signal is just an add on to the shadow transfer control and therefore all the previous described functions are still available.

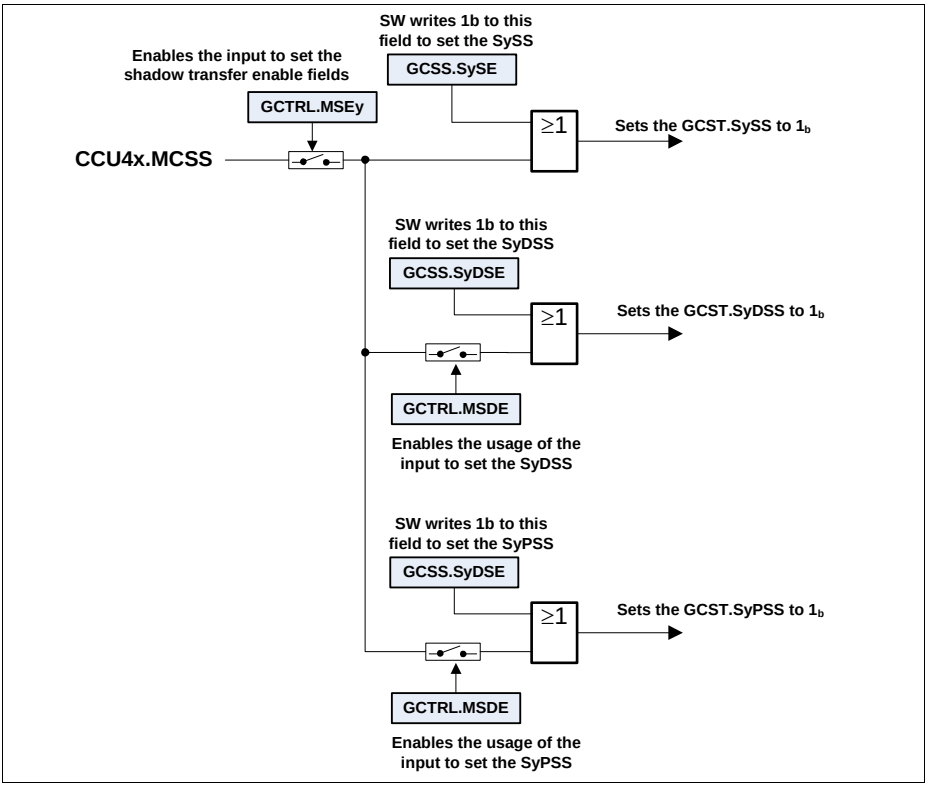


Figure 16-11 Usage of the CCU4x.MCSS input

16.2.5.3 Edge Aligned Mode

Edge aligned mode is the default counting scheme. In this mode, the timer is incremented until it matches the value programmed in the period register, **CC4yPR**. When period match is detected the timer is cleared to 0000_H and continues to be incremented.

In this mode, the value of the period register and compare register are updated with the value written by software into the correspondent shadow register, every time that an overflow occurs (period match), see **Figure 16-12**.

In edge aligned mode, the status bit of the comparison (CC4yST) is set one clock cycle after the timer hits the value programmed into the compare register. The clear of the status bit is done one clock cycle after the timer reaches 0000_H.

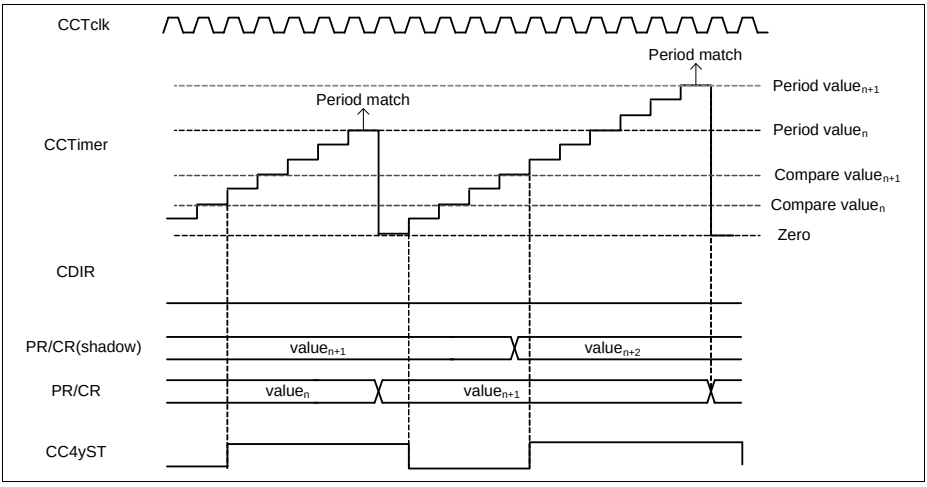


Figure 16-12 Edge aligned mode, CC4yTC.TCM = 0_B

16.2.5.4 Center Aligned Mode

In center aligned mode, the timer is counting up or down with respect to the following rules:

- The counter counts up while **CC4yTCST**.CDIR = 0_B and it counts down while **CC4yTCST**.CDIR = 1_B.
- Within the next clock cycle, the count direction is set to counting up (**CC4yTCST**.CDIR = 0_B) when the counter reaches 0001_H while counting down.
- Within the next clock cycle, the count direction is set to counting down (**CC4yTCST**.CDIR = 1_B), when the period match is detected while counting up.

The status bit (CC4yST) is always 1_B when the counter value is equal or greater than the compare value and 0_B otherwise.

While in edge aligned mode, the shadow transfer for compare and period registers is executed once per period. It is executed twice in center aligned mode as follows

- Within the next clock cycle after the counter reaches the period value, while counting up (**CC4yTCST**.CDIR = 0_B).
- Within the next clock cycle after the counter reaches 0001_H, while counting down (**CC4yTCST**.CDIR = 1_B).

*Note: Bit **CC4yTCST**.CDIR changes within the next timer clock after the one-match or the period-match, which means that the timer continues counting in the previous direction for one more cycle before changing the direction.*

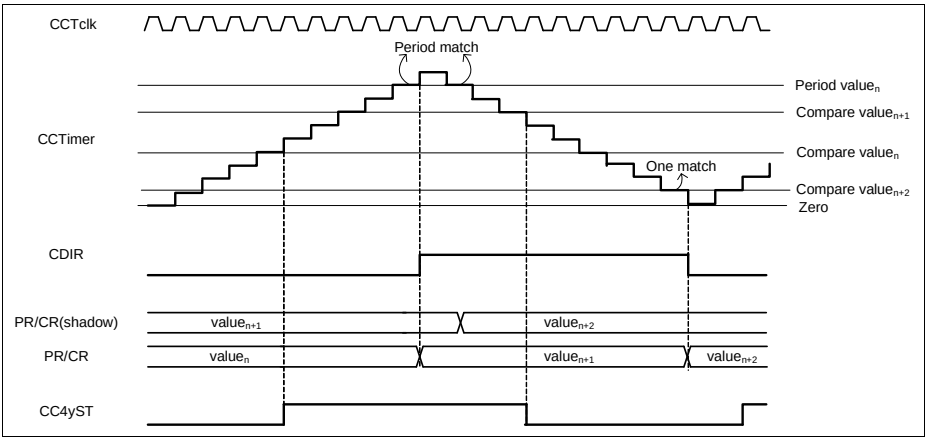


Figure 16-13 Center aligned mode, CC4yTC.TCM = 1_B

16.2.5.5 Single Shot Mode

In single shot mode, the timer is stopped after the current timer period is finished. This mode can be used with center or edge aligned scheme.

In edge aligned mode, **Figure 16-14**, the timer is stopped when it is cleared to 0000_H after having reached the period value. In center aligned mode, **Figure 16-15**, the period is finished when the timer has counted down to 0000_H.

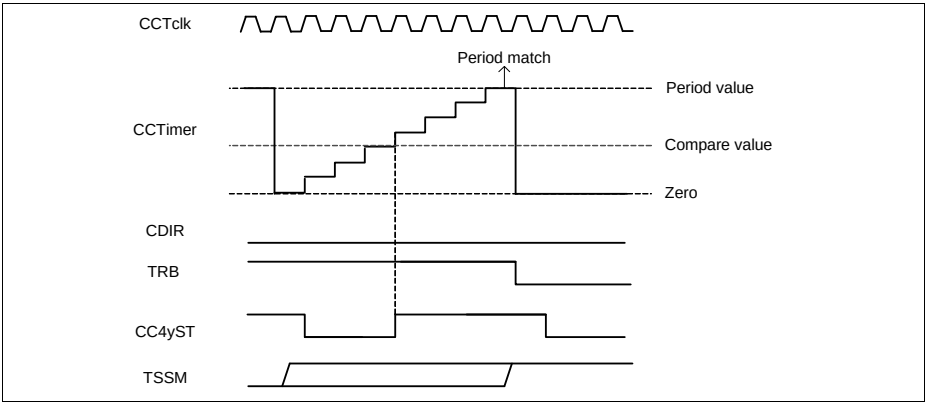


Figure 16-14 Single shot edge aligned - CC4yTC.TSSM = 1_B, CC4yTC.TCM = 0_B

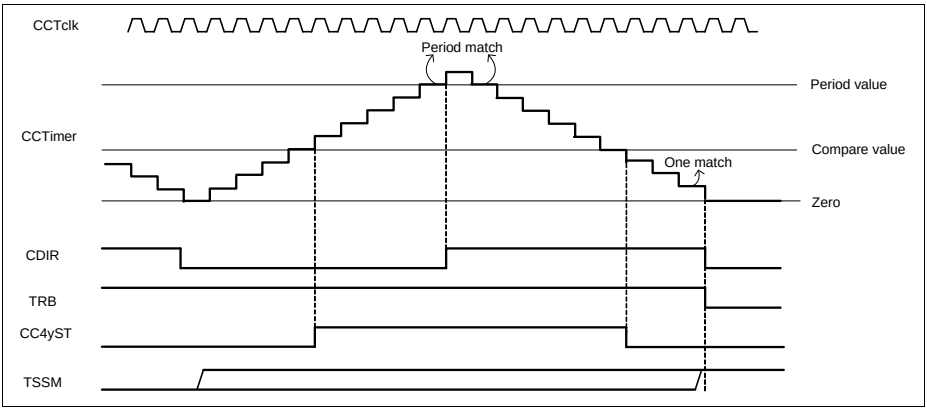


Figure 16-15 Single shot center aligned - CC4yTC.TSSM = 1_B, CC4yTC.TCM = 1_B

16.2.6 Active/Passive Rules

The general rules that set or clear the associated timer slice status bit (CC4yST), can be generalized independently of the timer counting mode.

The following events set the Status bit (CC4yST) to Active:

- in the next f_{tclk} cycle after a compare match while counting up
- in the next f_{tclk} cycle after a zero match while counting down

The following events set the Status bit (CC4yST) to Inactive:

- in the next f_{tclk} cycle after a zero match (and not compare match) while counting up
- in the next f_{tclk} cycle after a compare match while counting down

If external events are being used to control the timer operation, these rules are still applicable.

The status bit state can only be ‘override’ via software or by the external status bit override function, **Section 16.2.7.10**.

The software can at any time write a 1_B into the **GCSS**.SySTS bitfield, which will set the status bit **GCST**.CC4yST of the specific timer slice. Writing a 1_B into the **GCSC**.SySTC bitfield will clear the specific status bit.

16.2.7 External Events Control

Each CCU4 timer slice has the possibility of using up to three different input events, see **Section 16.2.2**. These three events can then be mapped to Timer Slice functions (the full set of available functions is described at **Section 16.2.3**)

These events can be mapped to any of the CCU4x.INy[P...A] inputs and there isn’t any imposition that an event cannot be used to perform several functions, or that an input

cannot be mapped to several events (e.g. input X triggers event 0 with rising edge and triggers event 1 with the falling edge).

16.2.7.1 External Start/Stop

To select an external start function, one should map one of the events (output of the input selector) to a specific input signal, by setting the required value in the **CC4yINS**.EVxIS field and indicating the active edge of the signal on the **CC4yINS**.EVxEM field.

This event should be then mapped to the start or stop functionality by setting the **CC4yCMC**.STRTS (for the start) or the **CC4yTC**.ENDM (for the stop) with the proper value.

Notice that both start and stop functions are edge and not level active and therefore the active/passive configuration is set only by the **CC4yINS**.EVxEM.

The external stop by default just clears the run bit (**CC4yTCST**.TRB), while the start functions does the opposite. Nevertheless one can select an extended subset of functions for the external start and stop. This subset is controlled by the registers **CC4yTC**.ENDM (for the stop) and **CC4yTC**.STRM (for the start).

For the start subset (**CC4yTC**.STRM):

- sets the run bit/starts the timer (resume operation)
- clears the timer, sets the run bit/starts the timer (flush and start)

For the stop subset (**CC4yTC**.ENDM):

- clears the run/stops the timer (stop)
- clears the timer (flush)
- clears the timer, clears the run bit/stops the timer (flush and stop)

If in conjunction with an external start/stop (configured also/only as flush) and external up/down signal is used, during the flush operation the timer is going to be set to 0000_H if the actual counting direction is up or set with the value of the period register if the counting direction is down.

Figure 16-16 to **Figure 16-19** shows the usage of two signals to perform the start/stop functions in all the previously mentioned subsets. External Signal(1) acts as an active HIGH start signal, while External Signal(2) is used as an active HIGH stop function.

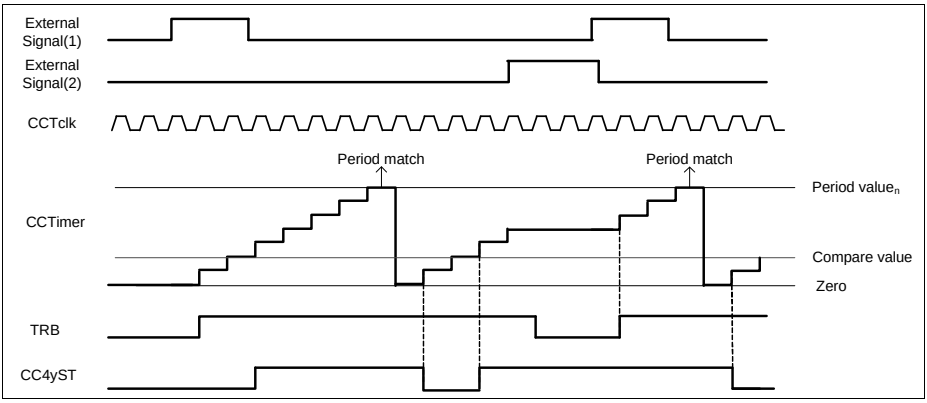


Figure 16-16 Start (as start)/ stop (as stop) - CC4yTC.STRM = 0_B, CC4yTC.ENDM = 00_B

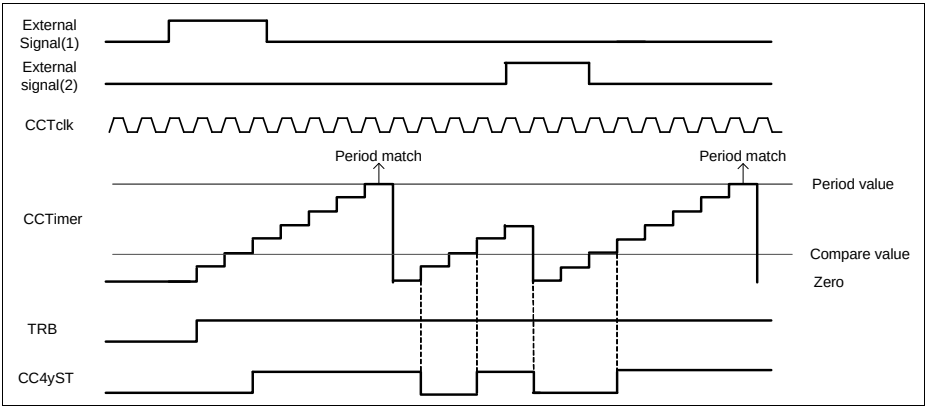


Figure 16-17 Start (as start)/ stop (as flush) - CC4yTC.STRM = 0_B, CC4yTC.ENDM = 01_B

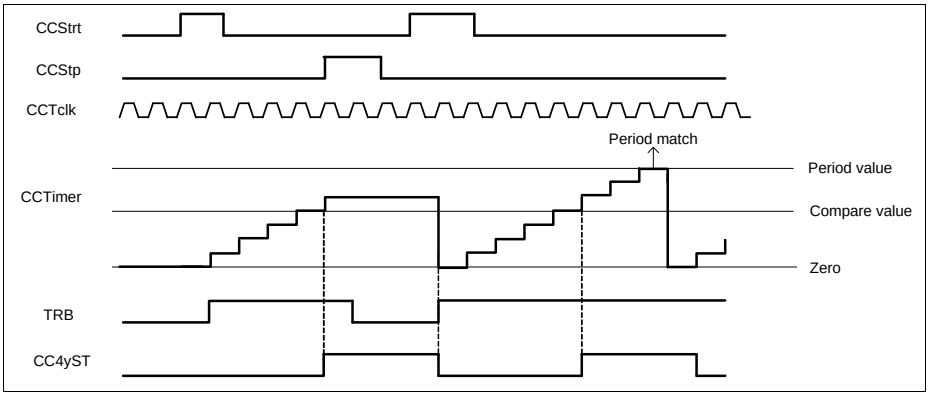


Figure 16-18 Start (as flush and start)/ stop (as stop) - CC4yTC.STRM = 1_B, CC4yTC.ENDM = 00_B

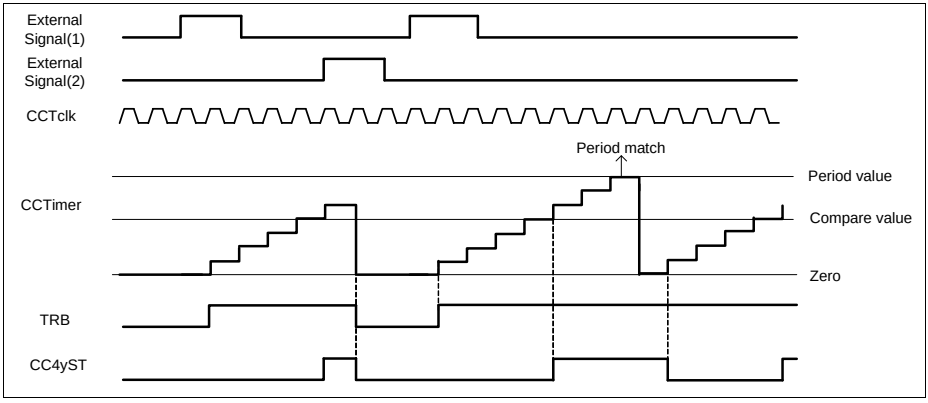


Figure 16-19 Start (as start)/ stop (as flush and stop) - CC4yTC.STRM = 0_B, CC4yTC.ENDM = 10_B

16.2.7.2 External Counting Direction

There is the possibility of selecting an input signal to act as increment/decrement control.

To select an external up/down control, one should map one of the events (output of the input selector) to a specific input signal, by setting the required value in the **CC4yINS**.EVxIS field and indicating the active level of the signal on the **CC4yINS**.EVxLM. This event should be then mapped to the up/down functionality by setting **CC4yCMC**.UDS with the proper value.

Notice that the up/down function is level active and therefore the active/passive configuration is set only by the **CC4yINS**.EVxLM.

The status bit of the slice (CC4yST) is always set when the timer value is equal or greater than the value stored in the compare register, see **Section 16.2.6**.

The update of the period and compare register values is done when:

- with the next clock after a period match, while counting up (**CC4yTCST**.CDIR = 0_B)
- with the next clock after a one match, while counting down (**CC4yTCST**.CDIR = 1_B)

The value of the **CC4yTCST**.CDIR register is updated accordingly with the changes on the decoded event. The Up/Down direction is always understood as **CC4yTCST**.CDIR = 1_B when counting down and **CC4yTCST**.CDIR = 0_B when counting up. Using an external signal to perform the up/down counting function and configuring the event as active HIGH means that the timer is counting up when the signal is HIGH and counting down when LOW.

Figure 16-20 shows an external signal being used to control the counting direction of the time. This signal was selected as active HIGH, which means that the timer is counting down while the signal is HIGH and counting up when the signal is LOW.

*Note: For a signal that should impose an increment when LOW and a decrement when HIGH, the user needs to set the **CC4yINS**.EVxLM = 0_B. When the operation is switched, then the user should set **CC4yINS**.EVxLM = 1_B.*

Note: Using an external counting direction control, sets the slice in edge aligned mode.

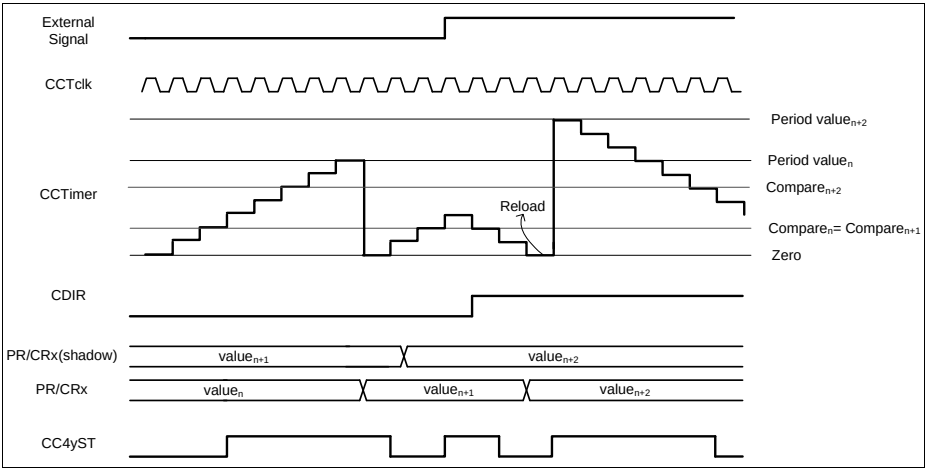


Figure 16-20 External counting direction

16.2.7.3 External Gating Signal

For pulse measurement, the user has the possibility of selecting an input signal that operates as counting gating.

To select an external gating control, one should map one of the events (output of the input selector) to a specific input signal, by setting the required value in the **CC4yINS**.EVxIS register and indicating the active level of the signal on the **CC4yINS**.EVxLM register. This event should be then mapped to the gating functionality by setting the **CC4yCMC**.GATES with the proper value.

Notice that the gating function is level active and therefore the active/passive configuration is set only by the **CC4yINS**.EVxLM.

The status bit during an external gating signal continues to be asserted when the compare value is reached and deasserted when the counter reaches 0000_H. One should note that the counter continues to use the period register to identify the wrap around condition. **Figure 16-21** shows the usage of an external signal for gating the slice counter. The signal was set as active LOW, which means the counter gating functionality is active when the external value is zero.

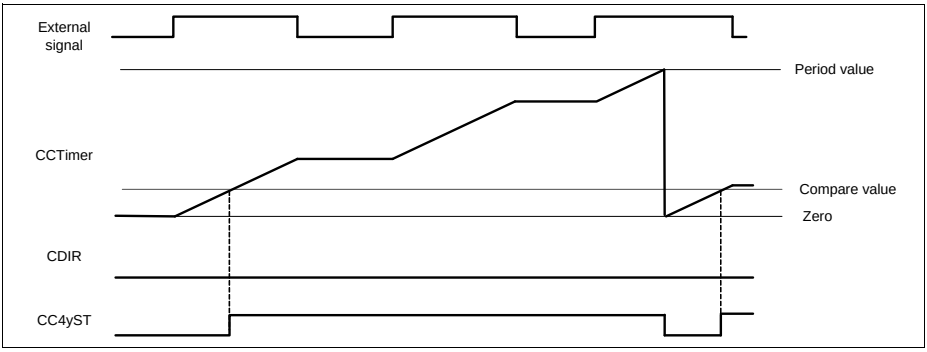


Figure 16-21 External gating

For any type of usage of the external gating function, the specific rung bit of the Timer Slice, **CC4yTCST**.TRB, needs to be set. This can be done via an additional external signal or directly via software.

16.2.7.4 External Count Signal

There is also the possibility of selecting an external signal to act as the counting event.

To select an external counting, one should map one of the events (output of the input selector) to a specific input signal, by setting the required value in the **CC4yINS**.EVxIS register and indicating the active edge of the signal on the **CC4yINS**.EVxEM register.

This event should be then mapped to the counting functionality by setting the **CC4yCMC**.CNTS with the proper value.

Notice that the counting function is edge active and therefore the active/passive configuration is set only by the **CC4yINS**.EVxEM.

One can select just a the rising, falling or both edges to perform a count. On **Figure 16-22**, the external signal was selected as a counter event for both falling and rising edges. Wrap around condition is still applied with a comparison with the period register.

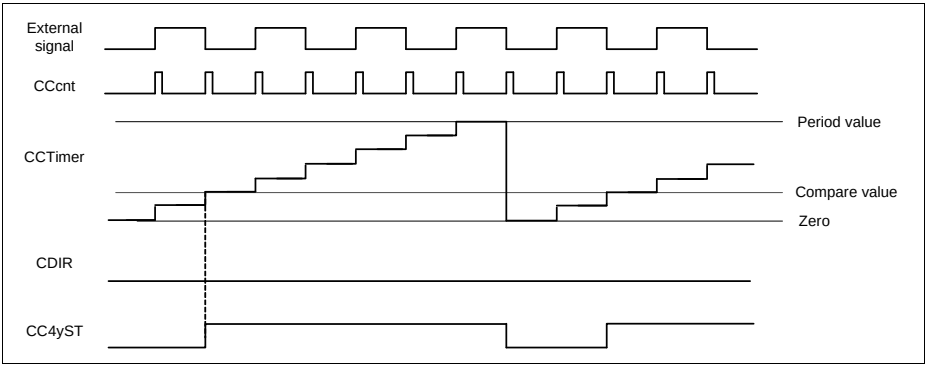


Figure 16-22 External count

For any type of usage of the external gating function, the specific rung bit of the Timer Slice, **CC4yTCST**.TRB, needs to be set. This can be done via an additional external signal or directly via software.

16.2.7.5 External Load

Each slice of the CCU4 also has a functionality that enables the user to select an external signal as trigger for reloading the value of the timer with the current value of the compare register (if **CC4yTCST**.CDIR = 0_B) or with the value of the period register (if **CC4yTCST**.CDIR = 1_B).

To select an external load signal, one should map one of the events (output of the input selector) to a specific input signal, by setting the required value in the **CC4yINS**.EVxIS register and indicating the active edge of the signal on the **CC4yINS**.EVxEM register. This event should be then mapped to the load functionality by setting the **CC4yCMC**.LDS with the proper value.

Notice that the load function is edge active and therefore the active/passive configuration is set only by the **CC4yINS**.EVxEM.

On figure **Figure 16-23**, the external signal (1) was used to act as a load trigger, active on the rising edge. Every time that a rising edge on external signal (1) is detected, the

timer value is loaded with the value present on the compare register. If an external signal is being used to control the counting direction, up or down, the timer value can be loaded also with the value set in the period register. The External signal (2) represents the counting direction control (active HIGH). If at the moment that a load trigger is detected, the signal controlling the counting direction is imposing a decrement, then the value set in the timer is the period value.

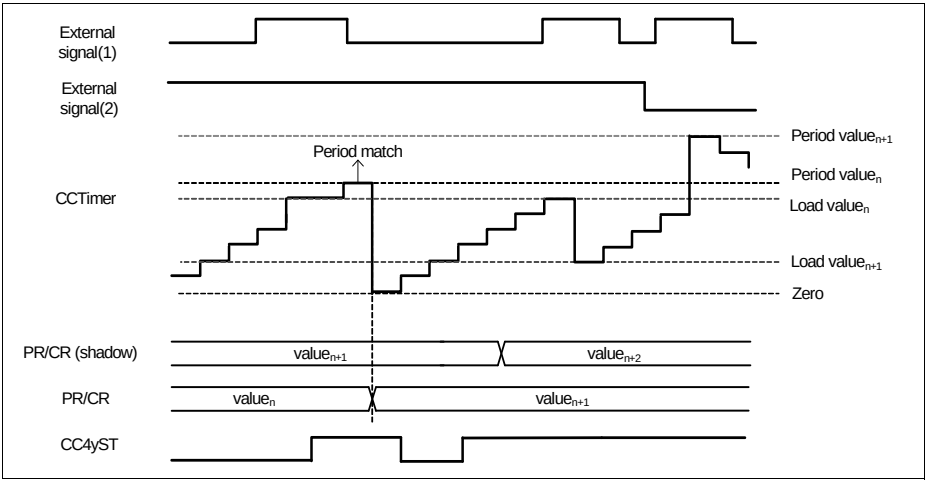


Figure 16-23 External load

16.2.7.6 External Capture

When selecting an external signal to be used as a capture trigger (if **CC4yCMC**.CAP0S or **CC4yCMC**.CAP1S are different from 0_H), the user is automatically setting the specific slice into capture mode.

In capture mode the user can have up to four capture registers, see **Figure 16-26**: capture register 0 (**CC4yC0V**), capture register 1 (**CC4yC1V**), capture register 2 (**CC4yC2V**) and capture register 3 (**CC4yC3V**).

These registers are shared between compare and capture modes which imposes:

- if **CC4yC0V** and **CC4yC1V** are used for capturing, the compare registers **CC4yCR** and **CC4yCRS** are not available (no compare channel)
- if **CC4yC2V** and **CC4yC3V** are used for capturing, the period registers **CC4yPR** and **CC4yPRS** are not available (no period control)

To select an external capture signal, one should map one of the events (output of the input selector) to a specific input signal, by setting the required value in the **CC4yINS**.EVxIS register and indicating the active edge of the signal on the

CC4yINS.EVxEM register. This event should be then mapped to the capture functionality by setting the **CC4yCMC**.CAP0S/**CC4yCMC**.CAP1S with the proper value.

Notice that the capture function is edge active and therefore the active/passive configuration is set only by the **CC4yINS**.EVxEM.

The user has the possibility of selecting the following capture schemes:

- Different capture events for **CC4yC0V**/**CC4yC1V** and **CC4yC2V**/**CC4yC3V**
- The same capture event for **CC4yC0V**/**CC4yC1V** and **CC4yC2V**/**CC4yC3V** with the same capture edge. For this capture scheme, only the CCcapt1 functionality needs to be programmed. To enable this scheme, the field **CC4yTC**.SCE needs to be set to 1.

Different Capture Events (SCE = 0_B)

Every time that a capture trigger 1 occurs, CCcapt1, the actual value of the timer is captured into the capture register 3 and the previous value stored in this register is transferred into capture register 2.

Every time that a capture trigger 0 occurs, CCcapt0, the actual value of the timer is captured into the capture register 1 and the previous value stored in this register is transferred into capture register 0.

Every time that a capture procedure into one of the registers occurs, the respective full flag is set. This flag is cleared automatically by HW when the SW reads back the value of the capture register (by reading the specific capture register or by reading the extended capture read value, see **Section 16.2.7.7**).

The capture of a new value into a specific capture registers is dictated by the status of the full flag as follows:

$$CC4yC1V_{capt} = \text{NOT}(CC4yC1V_{full\_flag} \text{ AND } CC4yC0V_{full\_flag}) \quad (16.4)$$

$$CC4yC0V_{capt} = CC4yC1V_{full\_flag} \text{ AND NOT}(CC4yC0V_{full\_flag}) \quad (16.5)$$

It is also possible to disable the effect of the full flags reset by setting the **CC4yTC**.CCS = 1_B. This enables a continuous capturing independent if the values captured have been read or not.

*Note: When using the period registers for capturing, **CC4yCMC**.CAP1S different from 00_B, the counter always uses its full 16 bit width as period value.*

On **Figure 16-24**, an external signal was selected as an event for capturing the timer value into the **CC4yC0V**/**CC4yC1V** registers. The status bit, CC4yST, during capture mode is asserted whenever a capture trigger is detected and deasserted when the counter matches 0000_H.

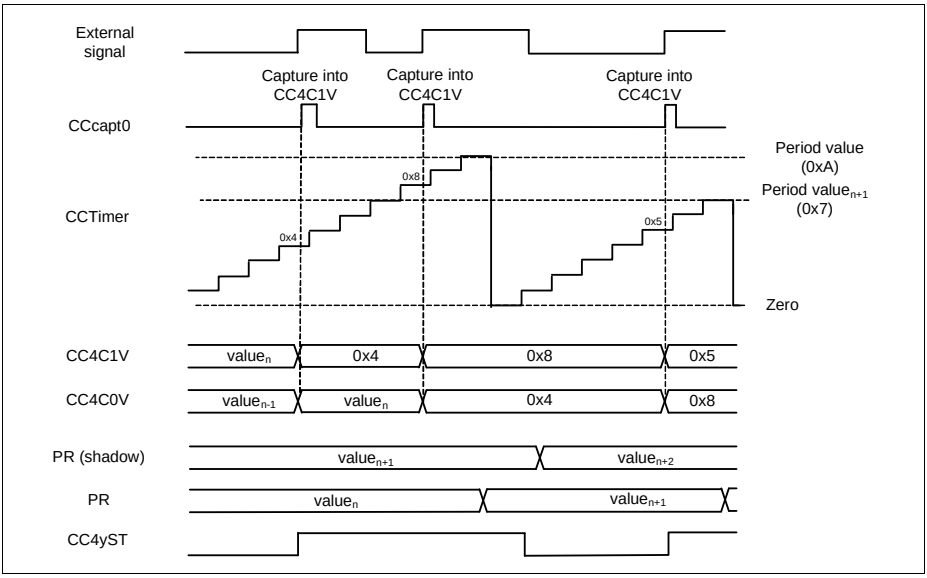


Figure 16-24 External capture - CC4yCMC.CAP0S != 00_B, CC4yCMC.CAP1S = 00_B

On **Figure 16-25**, two different signals were used as source for capturing the timer value into the **CC4yC0V**/**CC4yC1V** and **CC4yC2V**/**CC4yC3V** registers.

External signal(1) was selected as rising edge active capture source for **CC4yC0V**/**CC4yC1V**. External signal(2) was selected has the capture source for **CC4yC2V**/**CC4yC3V**, but as opposite to the external signal(1), the active edge was selected has falling.

See **Section 16.2.12.4**, for the complete capture mode usage description.

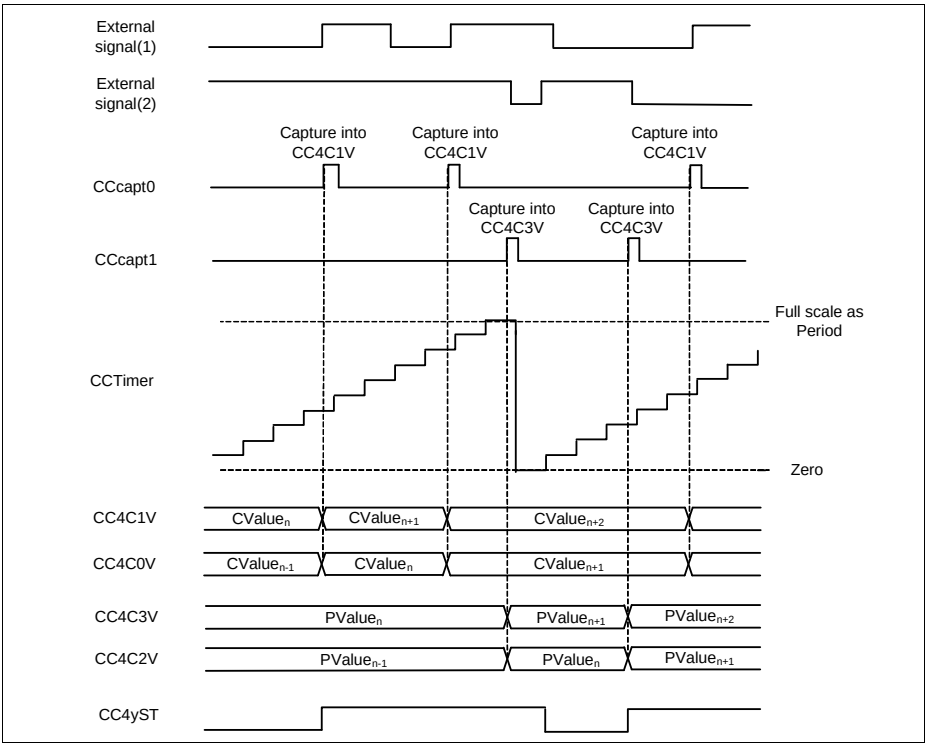


Figure 16-25 External capture - CC4yCMC.CAP0S != 00_B, CC4yCMC.CAP1S != 00_B

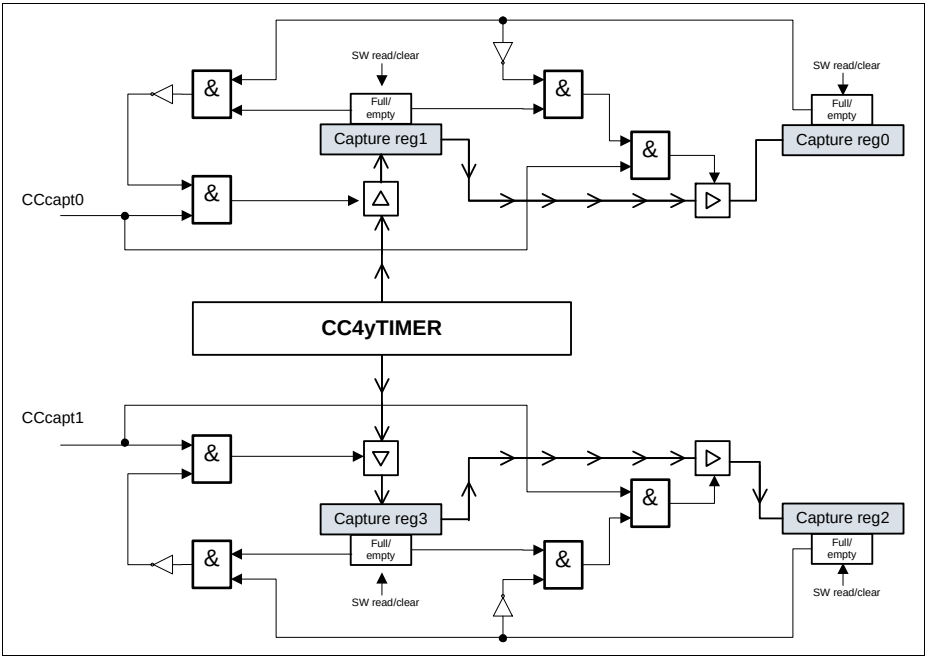


Figure 16-26 Slice capture logic

Same Capture Event (SCE = 1_B)

Setting the field **CC4yTC**.SCE = 1_B, enables the possibility of having 4 capture registers linked with the same capture event, **Figure 16-28**. The function that controls the capture is the CCcapt1.

The capture logic follows the same structure shown in **Figure 16-26** but extended to a four register chain, see **Figure 16-27**. The same full flag lock rules are applied to the four register chain (it also can be disabled by setting the **CC4yTC**.CCS = 1_B):

$$CC4yC3V_{capt} = \text{NOT}(CC4yC3V_{full\_flag} \text{ AND } CC4yC2V_{full\_flag} \text{ AND } CC4yC2V_{full\_flag} \text{ AND } CC4yC1V_{full\_flag}) \quad (16.6)$$

$$CC4yC2V_{capt} = CC4yC3V_{full\_flag} \text{ AND NOT}(CC4yC2V_{full\_flag} \text{ AND } CC4yC1V_{full\_flag} \text{ AND } CC4yC0V_{full\_flag}) \quad (16.7)$$

$$CC4yC1V_{capt} = CC4yC2V_{full\_flag} \text{ AND NOT}(CC4yC1V_{full\_flag} \text{ AND } CC4yC0V_{full\_flag}) \quad (16.8)$$

$$CC4yC0V_{capt} = CC4yC1V_{full\_flag} \text{ AND NOT}(CC4yC0V_{full\_flag}) \quad (16.9)$$

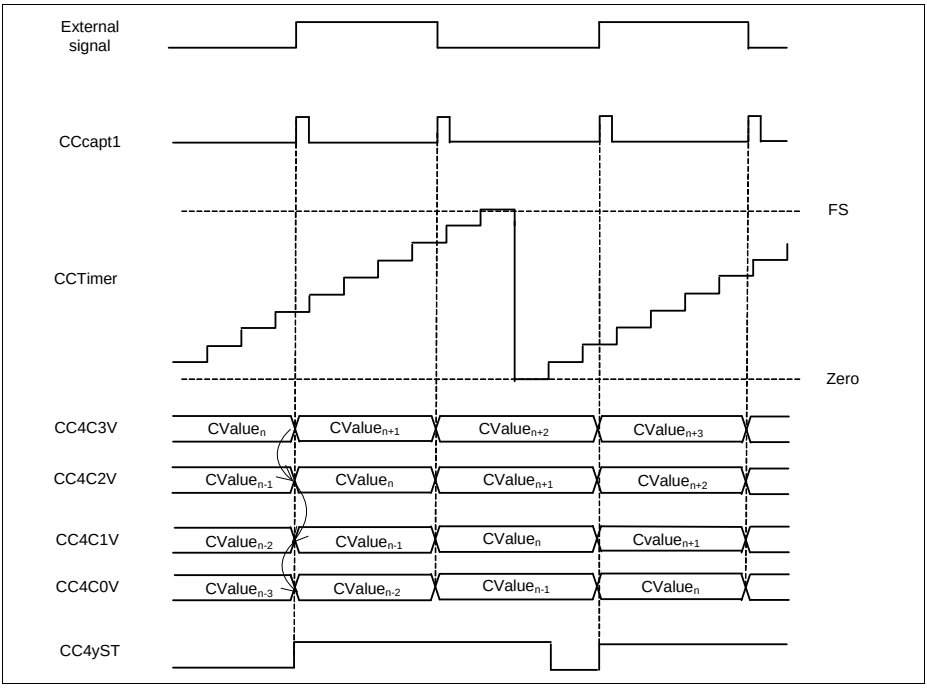


Figure 16-27 External Capture - CC4yTC.SCE = 1_B

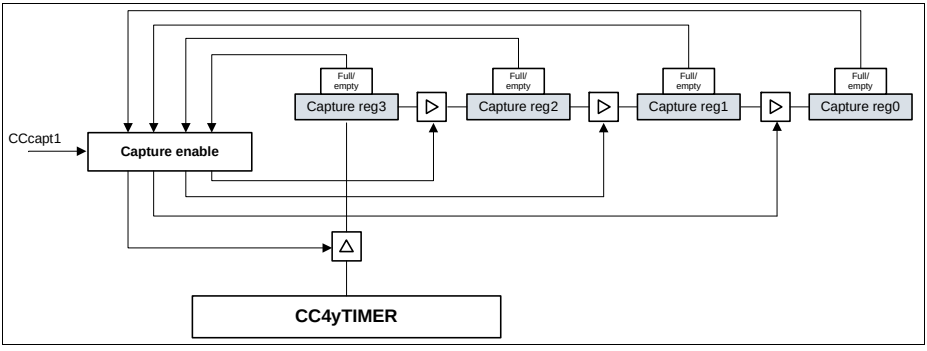


Figure 16-28 Slice Capture Logic - CC4yTC.SCE = 1_B

16.2.7.7 Capture Extended Read Back Mode

Each Timer Slice capture logic can operate in a FIFO read back mode. This mode can be enabled by setting the **CC4yTC**.ECM = 1_B. This Extended Read back mode allows the software to read back the capture data always from the same address (**CC4yECRD0** for the structure linked with the cpature trigger 0 or **CC4yECRD1** for the one linked with capture trigger 1). This read back will always return the oldest captured value, enabling an easy software routine implementation for reconstructing the capture data.

This function allows the usage of a FIFO structure for each capturing trigger. This relaxes the software read back routine when multiple capture triggers are present, and the software is not fast enough to perform a read operation in each capture event.

This FIFO read back function is present for a depth-4 and depth-2 FIFO structure.

The read back data contains also a lost value bitfield, that indicates if a capture trigger was lost due to the fact that the FIFO structure was full. This bitfield is set whenever a capture event was sensed and the FIFO was full (regardless if the continuos capture mode was enabled or not). This bitfield is cleared automatically by HW whenever the next read of the **CC4yECRD0**/**CC4yECRD1** register occurs. This bitfield does not indicate how many capture events were lost, it just indicates that between two ECRD reads at least a capture event was lost (this can help the SW evaluate which part of the data read, can be used for calculation).

*Note: When the ECM bitfield is set, reading the individual capture registers is still possible. Nevertheless the full flags can only be cleared by the HW when a read back is done via the **CC4yECRD0**/**CC4yECRD1** address.*

Depth 4 Structure

The FIFO depth-4 structure is present in the hardware when the capture trigger 1 is enabled and the **CC4yTC**.SCE = 1_B (same capture event), **Figure 16-29**.

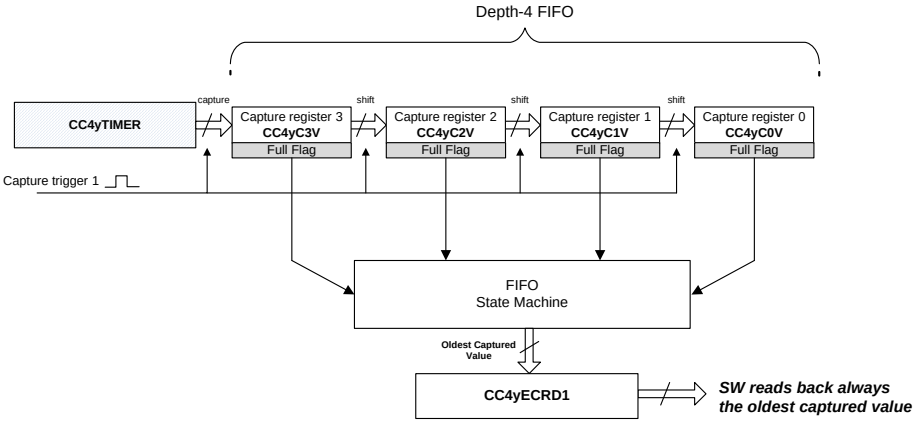


Figure 16-29 Capture Extended Read Back - Depth 4

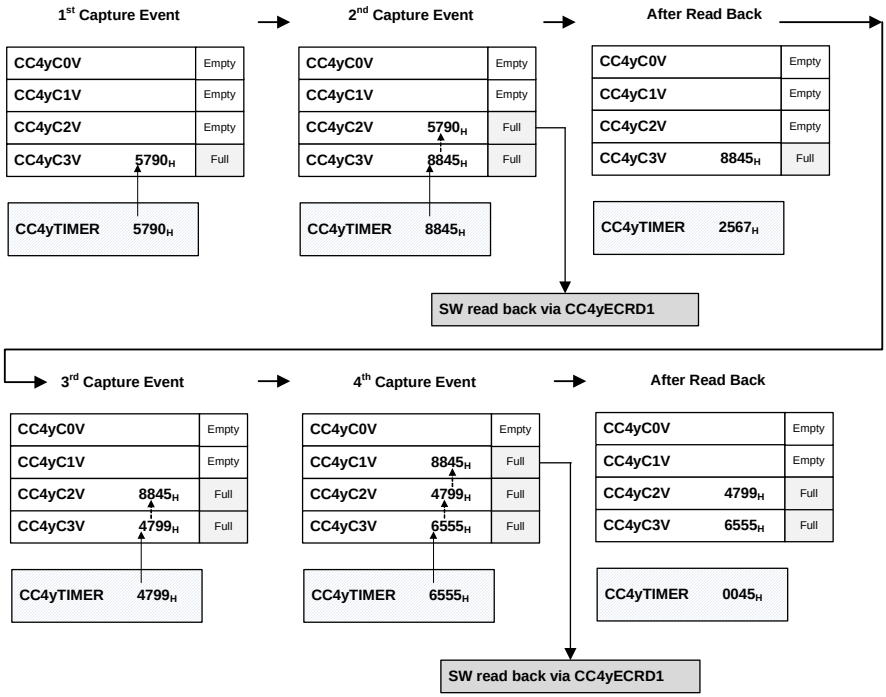


Figure 16-30 Depth 4 software access example

Depth 2 Structure

Each Timer Slice can have two capture structures of depth-2: one used with capture trigger 0 and another with capture trigger 1.

The one linked with capture trigger 0, is accessed via the **CC4yECRD0** while the one linked with the capture trigger 1 is accessed via the **CC4yECRD1**, **Figure 16-31**.

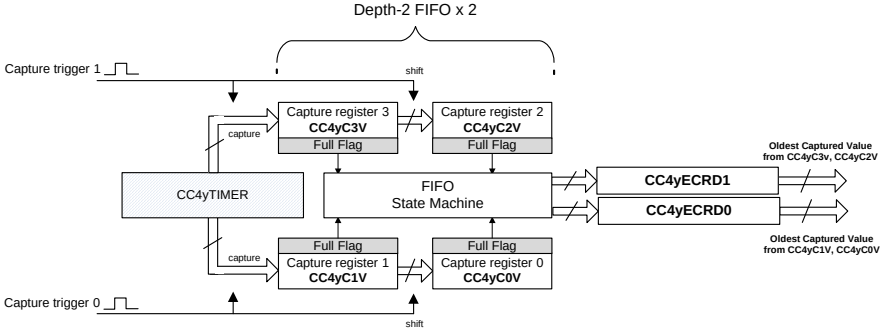


Figure 16-31 Capture Extended Read Back - Depth 2

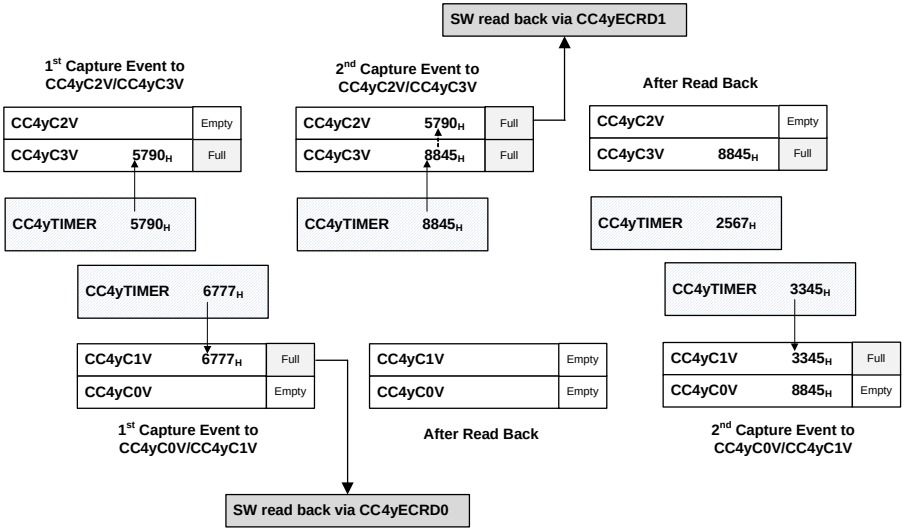


Figure 16-32 Depth 2 software access example

16.2.7.8 External Modulation

An external signal can be used to perform a modulation at the output of each timer slice.

To select an external modulation signal, one should map one of the input signals to one of the events, by setting the required value in the **CC4yINS**.EVxIS register and indicating the active level of the signal on the **CC4yINS**.EVxLM register. This event should be then

mapped to the modulation functionality by setting the **CC4yCMC**.MOS = 01_B if event 0 is being used, **CC4yCMC**.MOS = 10_B if event 1 or **CC4yCMC**.MOS = 11_B if event 2.

Notice that the modulation function is level active and therefore the active/passive configuration is set only by the **CC4yINS**.EVxLM.

The modulation has two modes of operation:

- modulation event is used to clear the CC4yST bit - **CC4yTC**.EMT = 0_B
- modulation event is used to gate the outputs - **CC4yTC**.EMT = 1_B

On **Figure 16-33**, we have a external signal configured to act as modulation source that clears the CC4yST bit, **CC4yTC**.EMT = 0_B. It was programmed to be an active LOW event and therefore, when this signal is LOW the output value follows the normal ACTIVE/PASSIVE rules.

When the signal is HIGH (inactive state), then the CC4yST bit is cleared and the output is forced into the PASSIVE state. Notice that the values of the status bit, CC4yST and the specific output CCU4x.OUTy are not linked together. One can choose for the output to be active LOW or HIGH through the PSL bit.

The exit of the external modulation inactive state is synchronized with the PWM signal due to the fact that the CC4yST bit is cleared and cannot be set while the modulation signal is inactive.

The entering into inactive state also can be synchronized with the PWM signal, by setting **CC4yTC**.EMS = 1_B. With this all possible glitches at the output are avoided, see **Figure 16-34**.

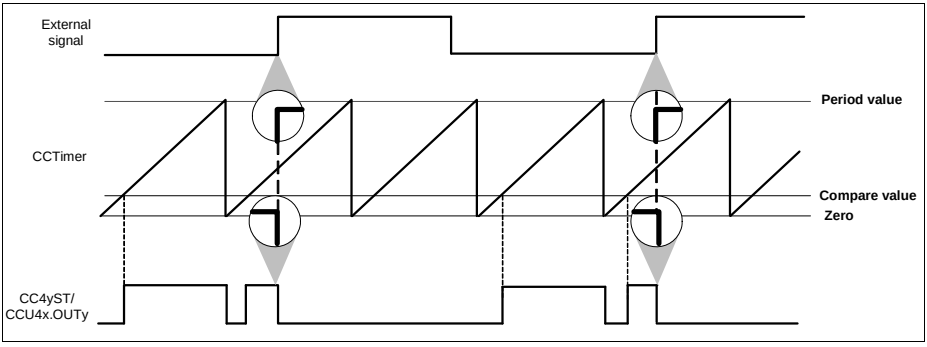


Figure 16-33 External modulation clearing the ST bit - CC4yTC.EMT = 0_B

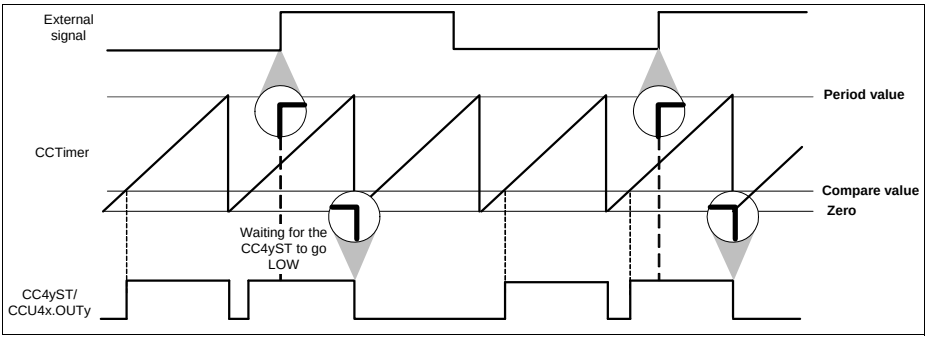


Figure 16-34 External modulation clearing the ST bit - CC4yTC.EMT = 0_B, CC4yTC.EMS = 1_B

On **Figure 16-35**, the external modulation event was used as gating signal of the outputs, **CC4yTC**.EMT = 1_B. The external signal was configured to be active HIGH, **CC4yINS**.EVxLM = 0_B, which means that when the external signal is HIGH the outputs are set to the PASSIVE state.In this mode, the gating event can also be synchronized with the PWM signal by setting the **CC4yTC**.EMS = 1_B.

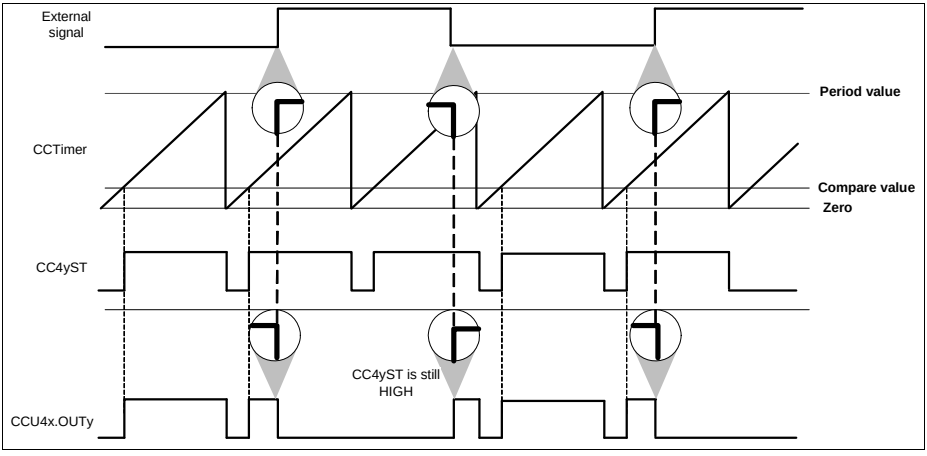


Figure 16-35 External modulation gating the output - CC4yTC.EMT = 1_B

16.2.7.9 TRAP Function

The TRAP functionality allows the PWM outputs to react on the state of an input pin. This functionality can be used to switch off the power devices if the TRAP input becomes active.

To select the TRAP functionality, one should map one of the input signals to event number 2, by setting the required value in the **CC4yINS**.EV2IS register and indicating the active level of the signal on the **CC4yINS**.EV2LM register. This event should be then mapped to the trap functionality by setting the **CC4yCMC**.TS = 1_B.

Notice that the trap function is level active and therefore the active/passive configuration is set only by the **CC4yINTS**.EV2LM.

There are two bitfields that can be monitored via software to crosscheck the TRAP function, **Figure 16-36**:

- The TRAP state bit, **CC4yINTS**.E2AS. This bitfield if the TRAP is currently active or not. This bitfield is therefore setting the specific Timer Slice output, into ACTIVE or PASSIVE state.
- The TRAP Flag, **CC4yINTS**.TRPF. This bitfield is used as a remainder in the case that the TRAP condition is cleared automatically via hardware. This field needs to be cleared by the software.

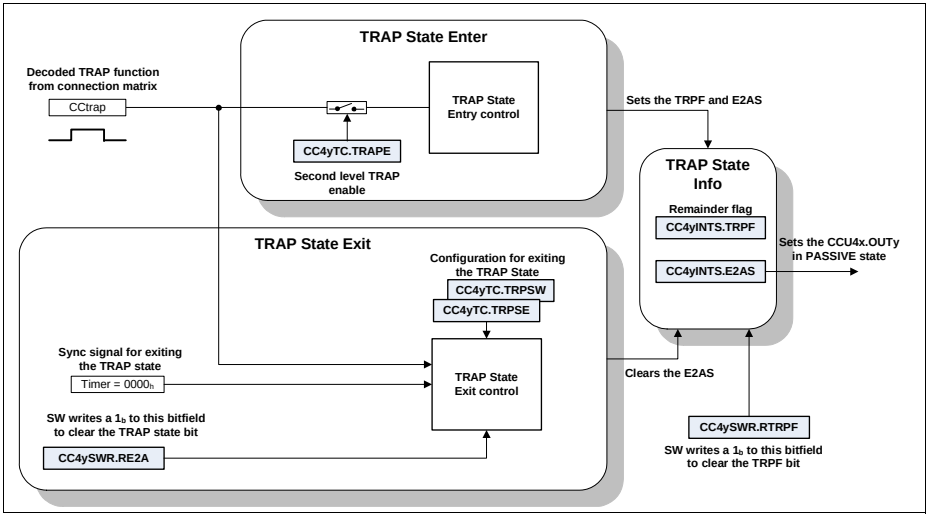


Figure 16-36 Trap control diagram

When a TRAP condition is detected at the selected input pin, both the Trap Flag and the Trap State bit are set to 1_B. The Trap State is entered immediately, by setting the CCU4xOUTy into the programmed PASSIVE state, **Figure 16-37**.

Exiting the Trap State can be done in two ways (**CC4yTC**.TRPSW register):

- automatically via HW, when the TRAP signal becomes inactive - **CC4yTC**.TRPSW = 0_B
- by SW only, by clearing the **CC4yINTS**.E2AS.The clearing is only possible if the input TRAP signal is in inactive state - **CC4yTC**.TRPSW = 1_B

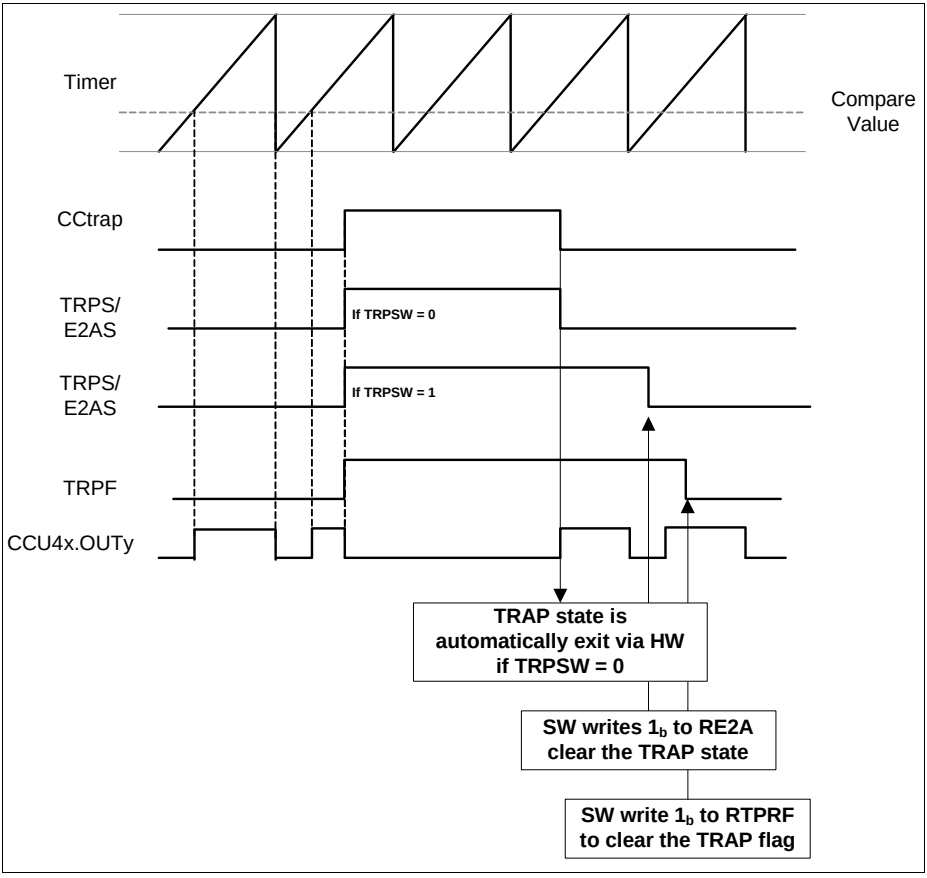


Figure 16-37 Trap timing diagram, CC4yPSL.PSL = 0_B (output passive level is 0_B)

It is also possible to synchronize the exiting of the TRAP state with the PWM signal, **Figure 16-38**. This function is enabled when the bitfield **CC4yTC**.TRPSE = 1_B.

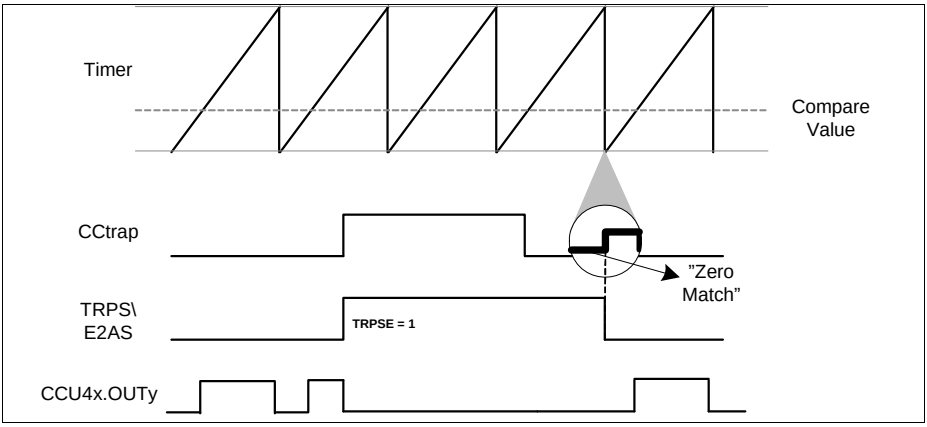


Figure 16-38 Trap synchronization with the PWM signal, CC4yTC.TRPSE = 1_B

16.2.7.10 Status Bit Override

For complex timed output control, each Timer Slice has a functionality that enables the override of the status bit (CC4yST) with a value passed trough an external signal.

The override of the status bit, can then lead to a change on the output pin, CCU4xOUTy (from inactive to active or vice versa).

To enable this functionality, two signals are needed:

- One signal that acts as a trigger to override the status bit (edge active)
- One signal that contains the value to be set in the status bit (level active)

To use the status bit override functionality, one should map the signal that acts as trigger to the event number 1, by setting the required value in the **CC4yINS**.EV1IS register and indicating the active edge of the signal on the **CC4yINS**.EV1EM register.

The signal that carries the value to be set on the status bit, needs to be mapped to the event number 2, by setting the required value in the **CC4yINS**.EV2IS register. The **CC4yINS**.EV2LM register should be set to 0_B if no inversion on the signal is needed and to 1_B otherwise.

The events should be then mapped to the status bit functionality by setting the **CC4yCMC**.OFS = 1_B.

Figure 16-39 shows the functionality of the status bit override, when the external signal(1) was selected as trigger source (rising edge active) and the external signal(2) was selected as override value.

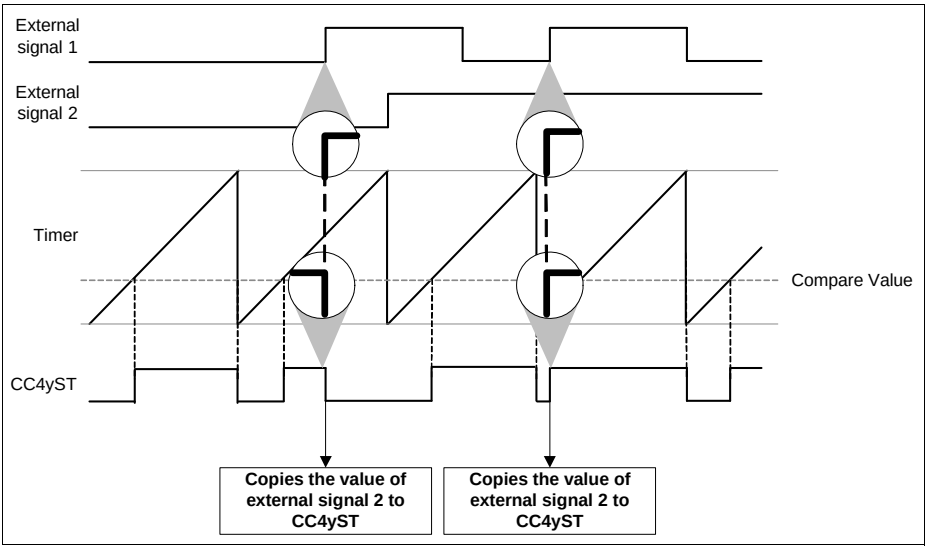


Figure 16-39 Status bit override

16.2.8 Multi-Channel Control

The multi channel control mode is selected individually in each slice by setting the **CC4yTC**.MCME = 1_B.

Within this mode, the output state of the Timer Slices (the ones set in multi channel mode) can be controlled in parallel by a single pattern.

The pattern is controlled via the CCU4 inputs, CCU4x.MCI0, CCU4x.MCI1, CCU4x.MCI2 and CCU4x.MCI3. Each of these inputs is connected directly to the associated slice input, e.g. CCU4x.MCI0 to CC40MCI, CCU4x.MCI1 to CC41MCI.

This pattern can be controlled directly by other module. The connectivity of each device may allow different control possibilities therefore one should address **Section 16.8** to check what modules are connected to these inputs.

When using the Multi Channel support of the CCU4, one can achieve a complete synchronicity between the output state update, CCU4x.OUTy, and the update of a new pattern, **Figure 16-40**. This synchronocity feature can be enabled by using some specific modules and therefore one should address **Section 16.8** to check which module is controlling the CCU4x.MCIy inputs.

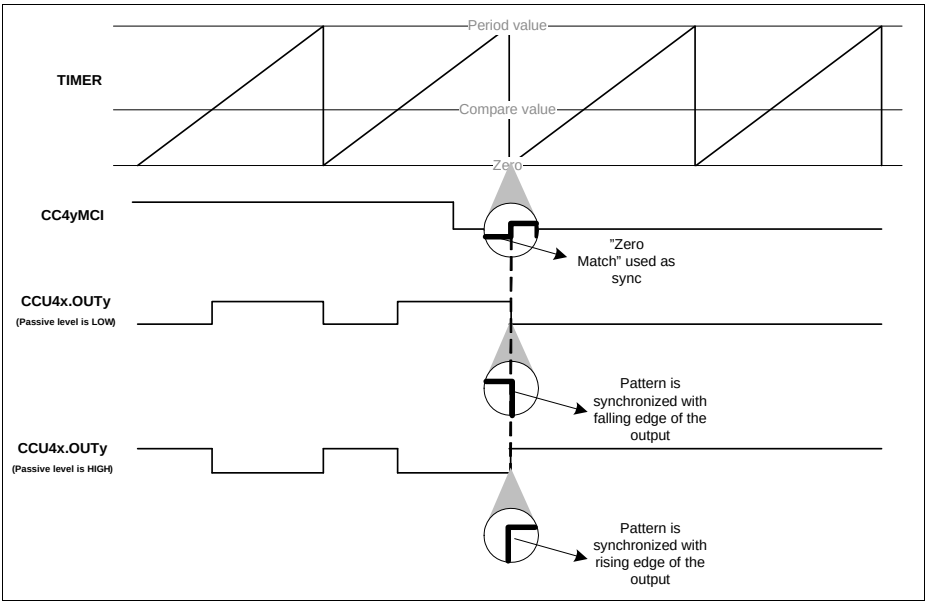


Figure 16-40 Multi channel pattern synchronization

Figure 16-41 shows the usage of the multi channel mode in conjunction with all four Timer Slices inside the CCU4.

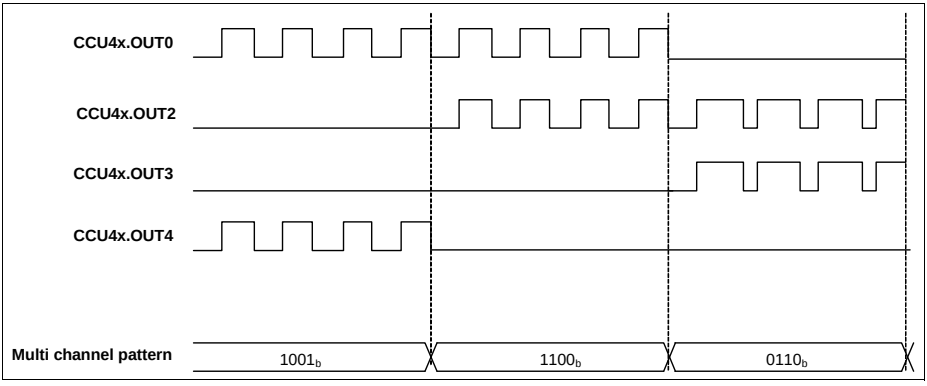


Figure 16-41 Multi Channel mode for multiple Timer Slices

The synchronization between the CCU4 and the module controlling the multi-channel pattern is achieved, by adding a 3 cycle delay on the output path of each Timer Slice

(between the status bit, CC4yST and the direct control of the output pin). This path is only selected when **CC4yINS**.MCME = 1_B, see **Figure 16-42**.

The multi pattern input synchronization can be seen on **Figure 16-43**. To achieve a synchronization between the update of the status bit, the sampling of a new multi channel pattern input is controlled by the period match or one match signal.

In a straightforward utilization of this synchronization feature, the module controlling the multi channel pattern signals, receives a sync signal from the CCU4, the CCU4x.PSy. This signal is then used by this module to update the multi-channel pattern. Due to the structure of the synchronization scheme inside the CCU4, the module controlling the multi-channel pattern needs to update this pattern, within 4 clock cycles after the CCU4x.PSy signal is asserted, **Figure 16-43**.

In a normal operation, where no external signal is used to control the counting direction, the signal used to enable the sampling of the pattern is always the period match when in edge aligned and the one match when in center aligned mode. When an external signal is used to control the counting direction, depending if the counter is counting up or counting down, the period match or the one match signal is used, respectively.

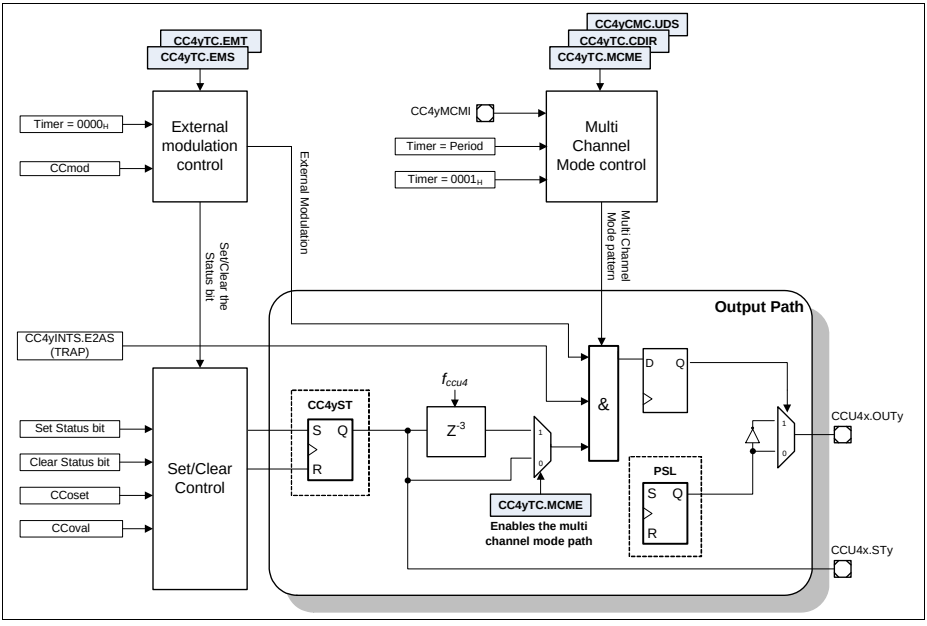


Figure 16-42 CC4y Status bit and Output Path

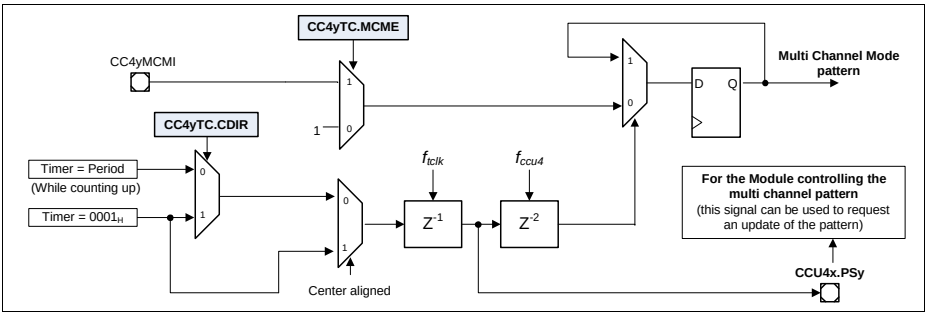


Figure 16-43 Multi Channel Mode Control Logic

16.2.9 Timer Concatenation

The CCU4 offers a very easy mechanism to perform a synchronous timer concatenation. This functionality can be used by setting the **CC4yTC**.TCE = 1_B. By doing this the user is doing a concatenation of the actual CCU4 slice with the previous one, see **Figure 16-44**.

Notice that is not possible to perform concatenation with non adjacent slices and that timer concatenation automatically sets the slice mode into Edge Aligned. It is not possible to perform timer concatenation in Center Aligned mode.

To enable a 64 bit timer, one should set the **CC4yTC**.TCE = 1_B in all the slices (with the exception of the CC40 due to the fact that it doesn't contain this control register).

To enable a 48 bit timer, one should set the **CC4yTC**.TCE = 1_B in two adjacent slices and to enable a 32 bit timer, the **CC4yTC**.TCE is set to 1_B in the slice containing the MSBs. Notice that the timer slice containing the LSBs should always have the TCE bitfield set to 0_B.

Several combinations for timer concatenation can be made inside a CCU4 module:

- one 64 bit timer
- one 48 bit timer plus a 16 timer
- two 32 bit timers
- one 32 bit timer plus two 16 bit timers

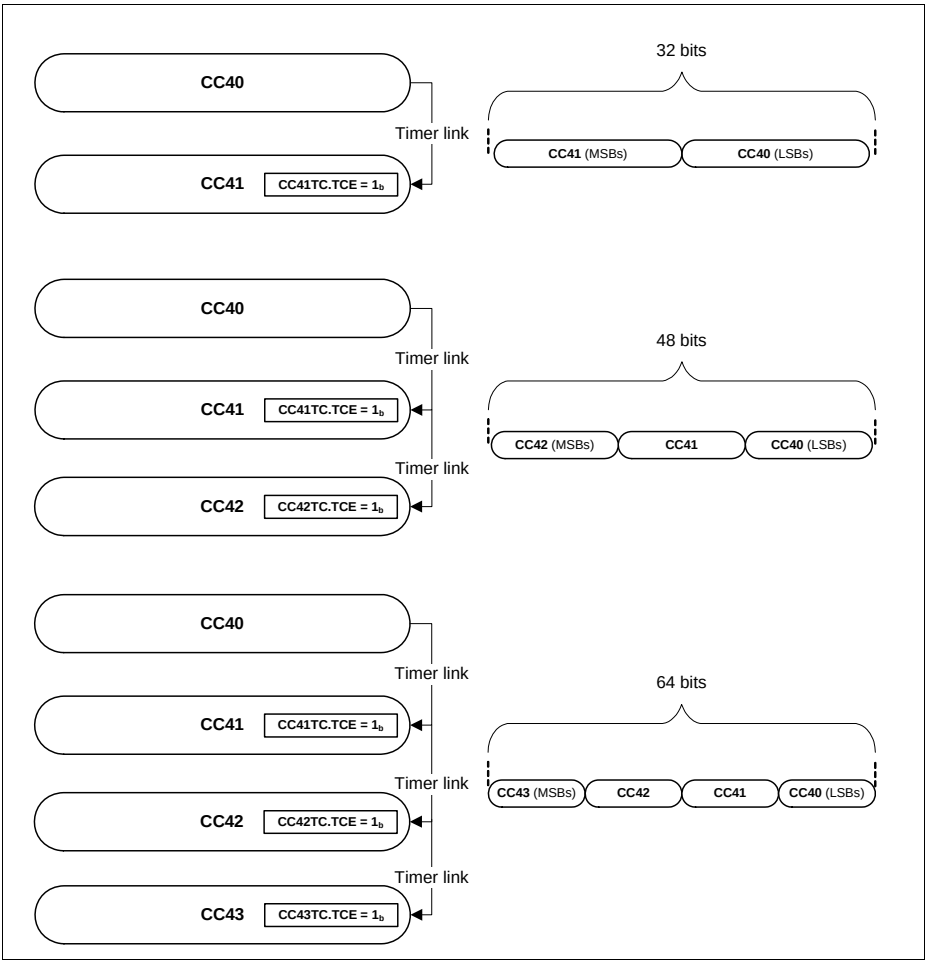


Figure 16-44 Timer Concatenation Example

Each Timer Slice is connected to the adjacent Timer Slices via a dedicated concatenation interface. This interface allows the concatenation of not only the Timer counting operation, but also a synchronous input trigger handling for capturing and loading operations, **Figure 16-45**.

Note: For all cases CC40 and CC43 are not considered adjacent slices

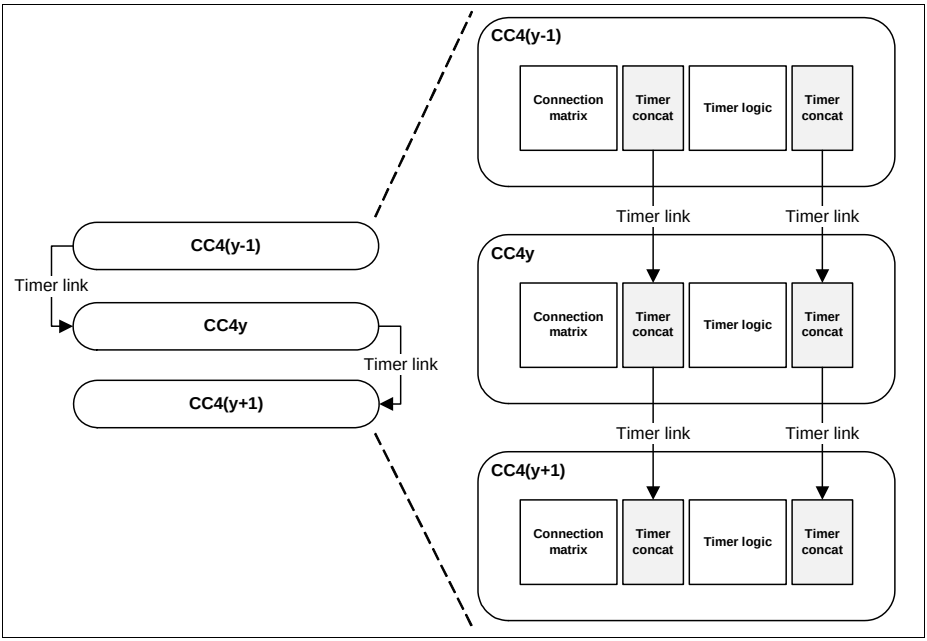


Figure 16-45 Timer Concatenation Link

Seven signals are present in the timer concatenation interface:

- Timer Period match (CC4yPM)
- Timer Zero match (CC4yZM)
- Timer Compare match (CC4yCM)
- Timer counting direction function (CCupd)
- Timer load function (CCload)
- Timer capture function for CC4yC0V and CC4yC1V registers (CCcap0)
- Timer capture function for CC4yC2V and CC4yC3V registers (CCcap1)

The first four signals are used to perform the synchronous timing concatenation at the output of the Timer Logic, like it is seen in **Figure 16-45**. With this link, the timer length can be easily adjusted to 32, 48 or 64 bits (counting up or counting down).

The last three signals are used to perform a synchronous link between the capture and load functions, for the concatenated timer system. This means that the user can have a capture or load function programmed in the first Timer Slice, and propagate this capture or load trigger synchronously from the LSBs until the MSBs, **Figure 16-46**.

The capture or load function only needs to be configured in the first Timer Slice (the one holding the LSBs). From the moment that **CC4yTC**.TCE is set to 1_B, in the following Timer Slices, the link between these functions is done automatically by the hardware.

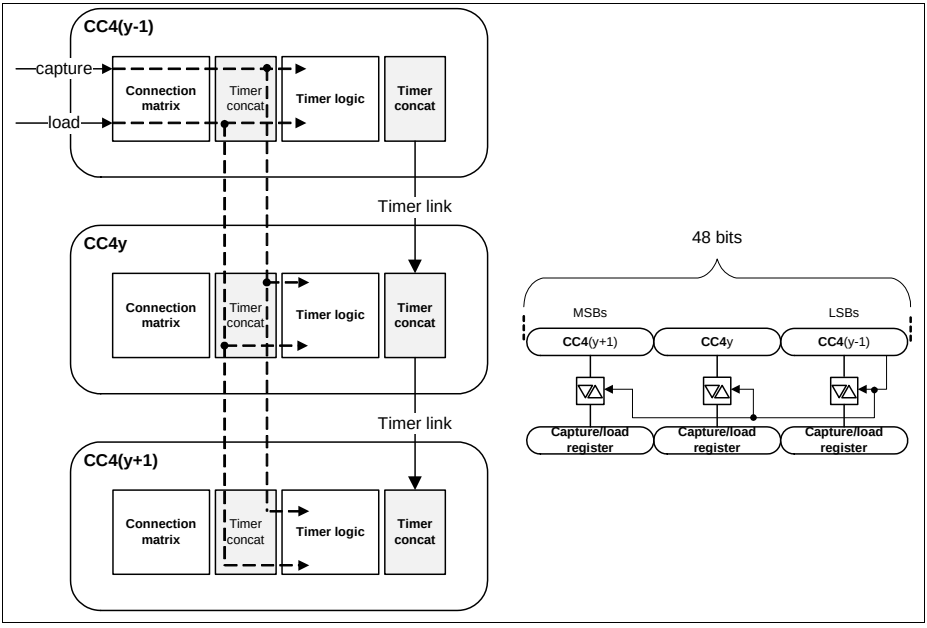


Figure 16-46 Capture/Load Timer Concatenation

The period match (CC4yPM) or zero match (CC4yZM) from the previous Timer Slice (with the immediately next lower index) are used in concatenated mode, as gating signal for the counter. This means that the counting operation of the MSBs only happens when a wrap around condition is detected, avoiding additional DSP operations to extract the counting value.

With the same methodology, the compare match (CC4yCM), zero match and period match are gated with the specific signals from the previous slice. This means that the timing information is propagated throughout all the slices, enabling a completely synchronous match between the LSB and MSB count, see **Figure 16-47**.

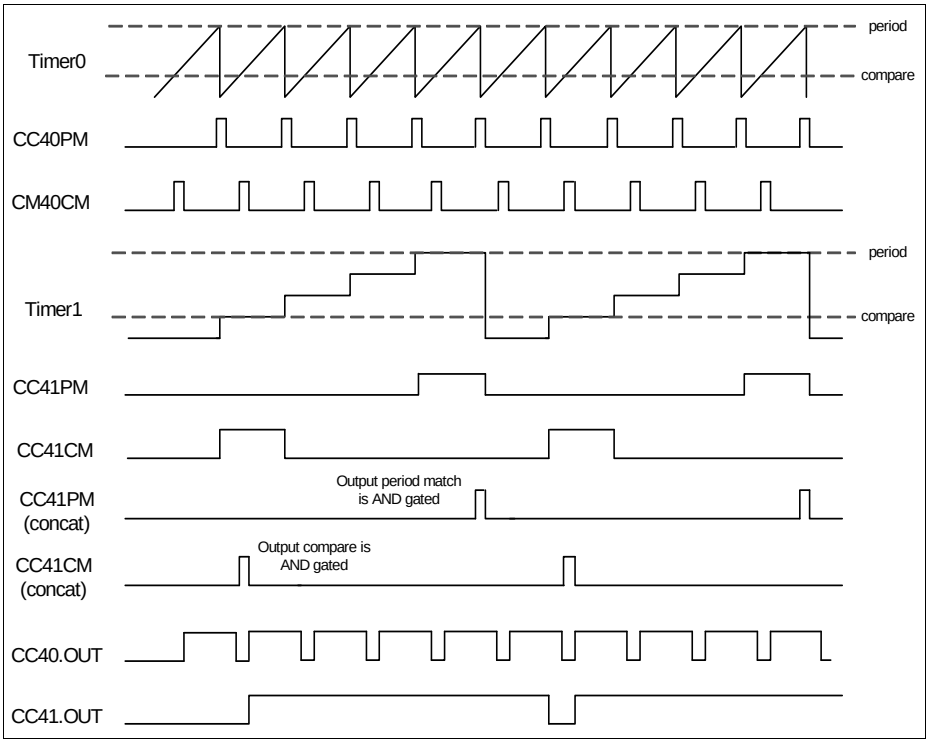


Figure 16-47 32 bit concatenation timing diagram

Note: The counting direction of the concatenated timer needs to be fixed. The timer can count up or count down, but the direction cannot be updated on the fly.

Figure 16-48 gives an overview of the timer concatenation logic. Notice that all the mechanism is controlled solely by the **CC4yTC**.TCE bitfield.

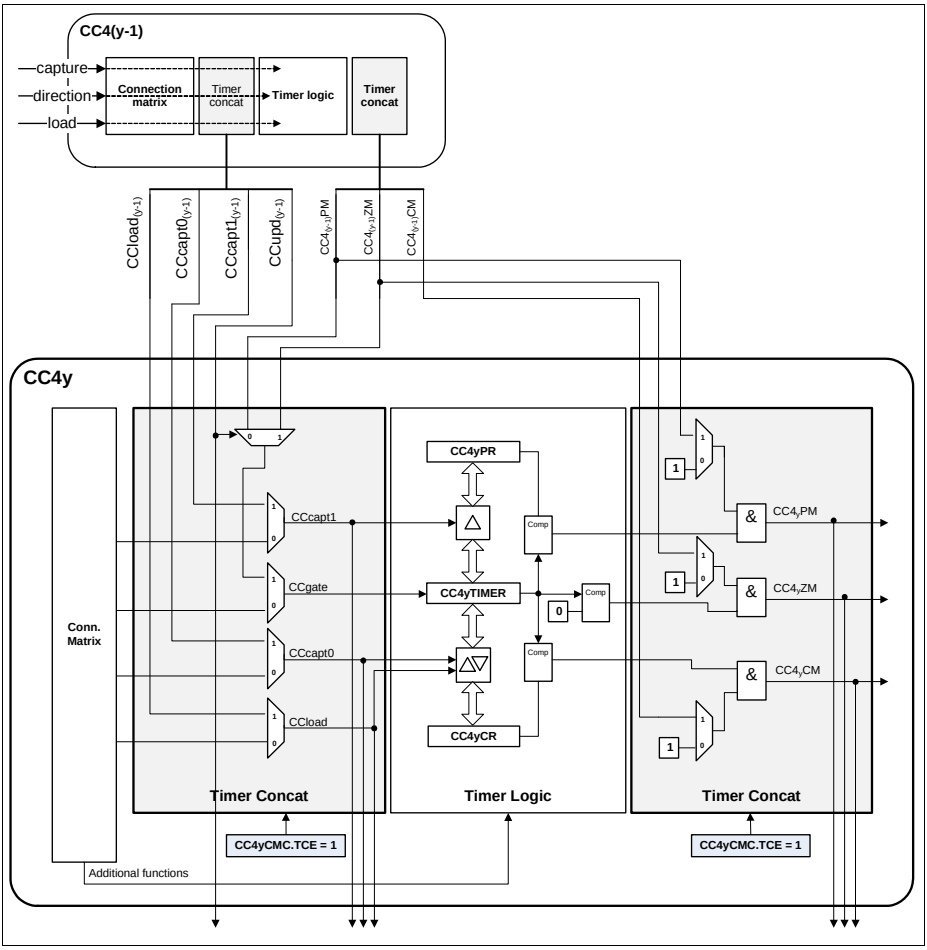


Figure 16-48 Timer concatenation control logic

16.2.10 PWM Dithering

The CCU4 has an automatic PWM dithering insertion function. This functionality can be used with very slow control loops that cannot update the period/compare values in a fast manner, and by that fact the loop can lose precision on long runs. By introducing dither on the PWM signal, the average frequency/duty cycle is then compensated against that error.

Each slice contains a dither control unit, see **Figure 16-49**.

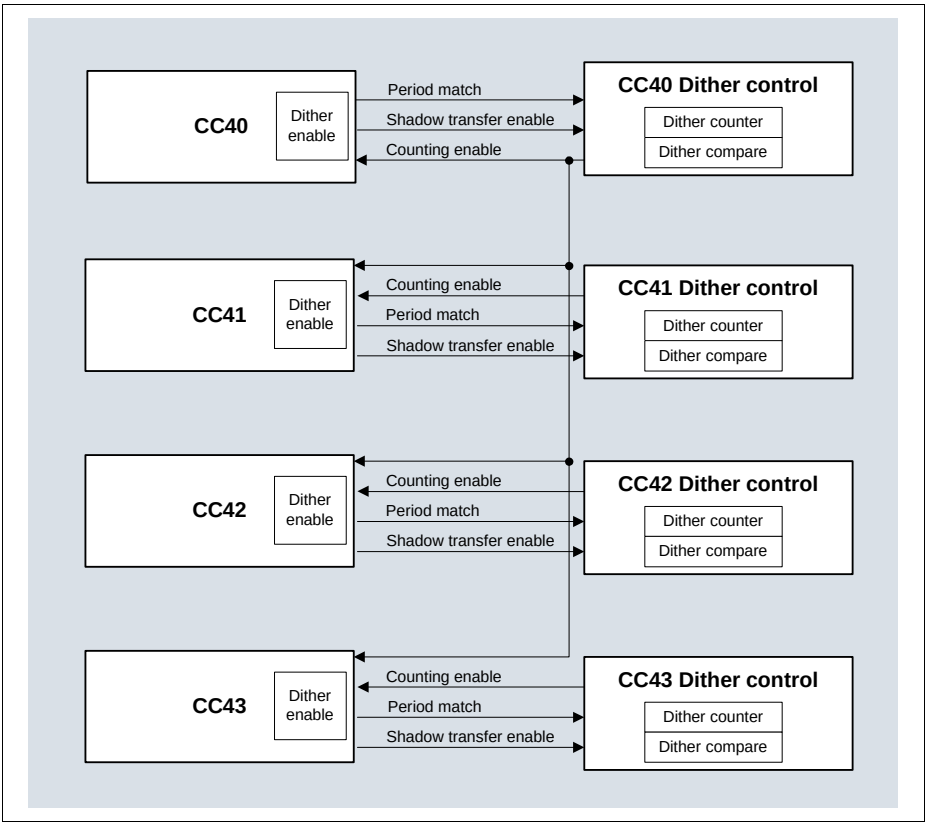


Figure 16-49 Dither structure overview

The dither control unit contains a 4 bit counter and a compare value. The four bit counter is incremented every time that a period match occurs. The counter works in a bit reverse mode so the distribution of increments stays uniform over 16 counter periods, see **Table 16-5**.

Table 16-5 Dither bit reverse counter

counter[3]	counter[2]	counter[1]	counter[0]
0	0	0	0
1	0	0	0
0	1	0	0
1	1	0	0
0	0	1	0
1	0	1	0
0	1	1	0
1	1	1	0
0	0	0	1
1	0	0	1
0	1	0	1
1	1	0	1
0	0	1	1
1	0	1	1
0	1	1	1
1	1	1	1

The counter is then compared against a programmed value, **CC4yDIT**.DCV. If the counter value is smaller than the programmed value, a gating signal is generated that can be used to extend the period, to delay the compare or both (controlled by the **CC4yTC**.DITHE field, see **Table 16-6**) for one clock cycle.

Table 16-6 Dither modes

DITHE[1]	DITH[0]	Mode
0	0	Dither is disabled
0	1	Period is increased by 1 cycle
1	0	Compare match is delayed by 1 cycle
1	1	Period is increased by 1 cycle and compare is delayed by 1 cycle

The dither compare value also has an associated shadow register that enables concurrent update with the period/compare register of CC4y. The control logic for the dithering unit is represented on **Figure 16-50**.

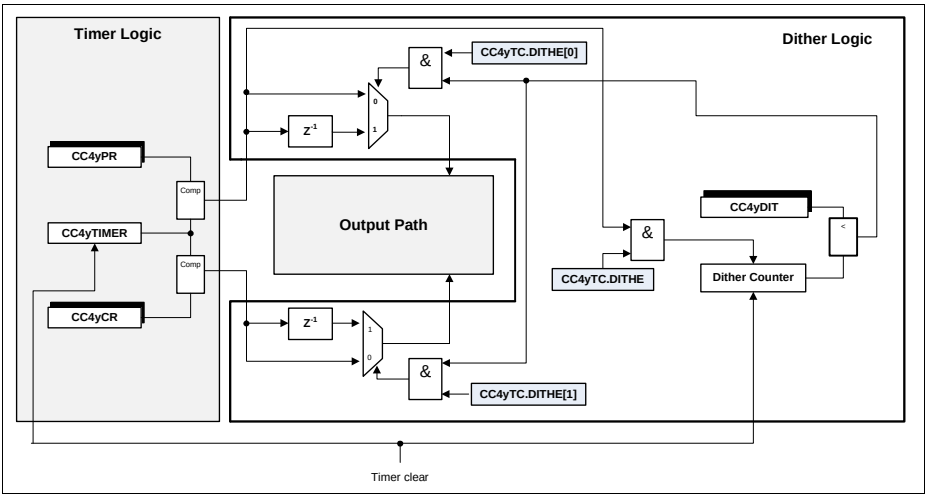


Figure 16-50 Dither control logic

Figure 16-51 to **Figure 16-56** show the effect of the different configurations for the dithering function, **CC4yTC**.DITHE, for both counting schemes, Edge and Center Aligned mode. In each figure, the bit reverse scheme is represented for the dither counter and the compare value was programmed with the value 8_H. In each figure, the variable *T*, represents the period of the counter, while the variable *d* indicates the duty cycle (status bit is set HIGH).

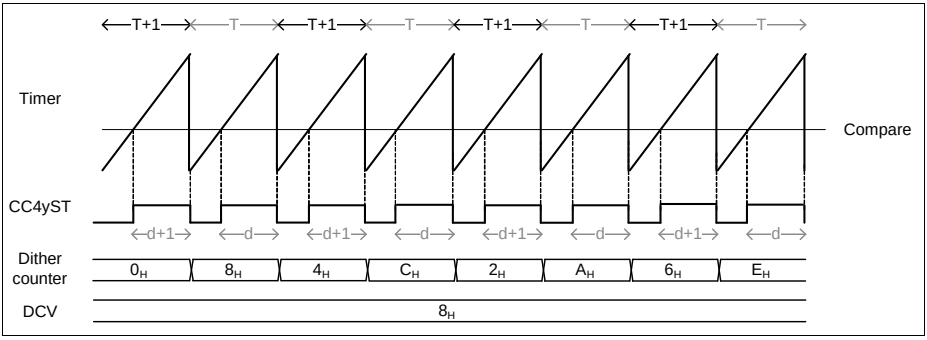


Figure 16-51 Dither timing diagram in edge aligned - CC4yTC.DITHE = 01_B

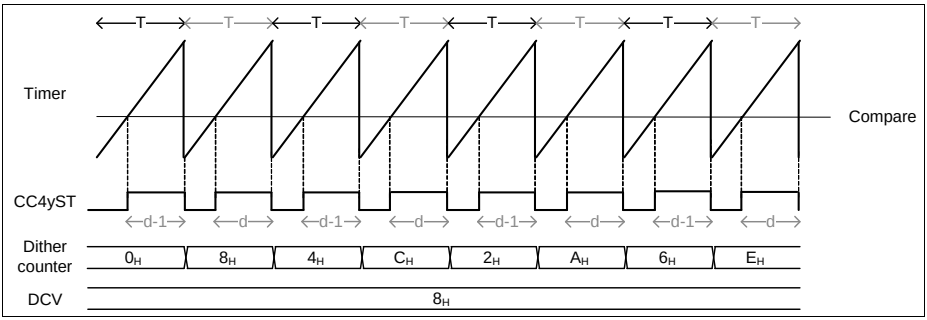


Figure 16-52 Dither timing diagram in edge aligned - CC4yTC.DITHE = 10_B

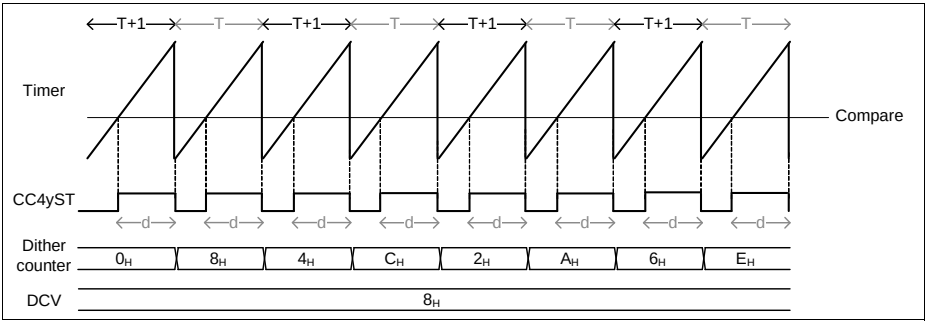


Figure 16-53 Dither timing diagram in edge aligned - CC4yTC.DITHE = 11_B

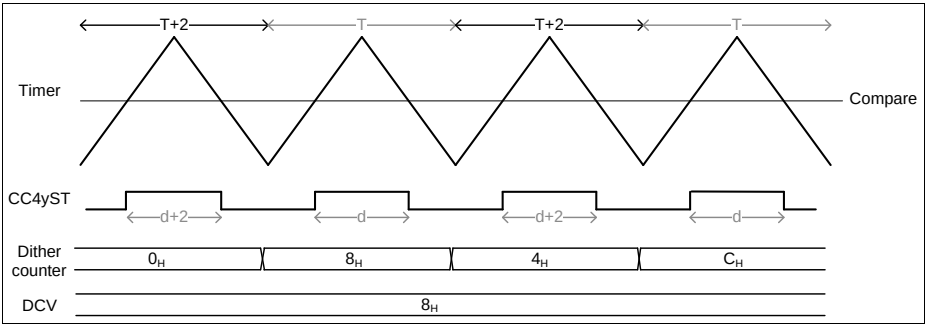


Figure 16-54 Dither timing diagram in center aligned - CC4yTC.DITHE = 01_B

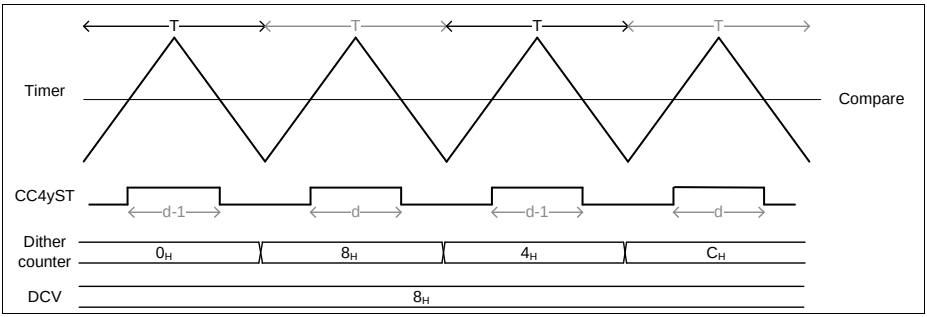


Figure 16-55 Dither timing diagram in center aligned - CC4yTC.DITHE = 10_B

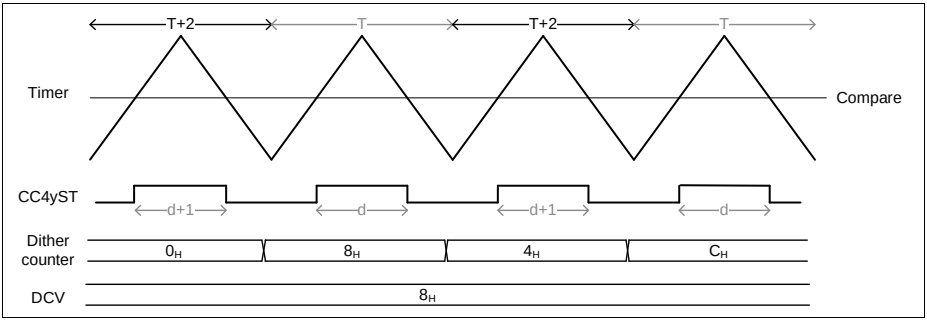


Figure 16-56 Dither timing diagram in center aligned - CC4yTC.DITHE = 11_B

Note: When using the dither, is not possible to select a period value of FS when in edge aligned mode. In center aligned mode, the period value must be at least FS - 2.

16.2.11 Prescaler

The CCU4 contains a 4 bit prescaler that can be used in two operating modes for each individual slice:

- normal prescaler mode
- floating prescaler mode

The run bit of the prescaler can be set/cleared by SW by writing into the registers, **GIDLC**.SPRB and **GIDLS**.CPRB respectively, and it can also be cleared by the run bit of a specific slice. With the last mechanism, the run bit of the prescaler is cleared one clock cycle after the clear of the run bit of the selected slice. To select which slice can perform this action, one should program the **GCTRL**.PRBC register.

16.2.11.1 Normal Prescaler Mode

In Normal prescaler mode the clock fed to the CC4y counter is a normal fixed division by N, accordingly to the value set in the **CC4yPSC**.PSIV register. The values for the possible division values are listed in **Table 16-7**. The **CC4yPSC**.PSIV value is only modified by a SW access. Notice that each slice has a dedicated prescaler value selector (**CC4yPSC**.PSIV), which means that the user can select different counter clocks for each Timer Slice (CC4y).

Table 16-7 Timer clock division options

CC4yPSC.PSIV	**Resulting clock**
0000_B	f_{ccu4}
0001_B	$f_{ccu4}/2$
0010_B	$f_{ccu4}/4$
0011_B	$f_{ccu4}/8$
0100_B	$f_{ccu4}/16$
0101_B	$f_{ccu4}/32$
0110_B	$f_{ccu4}/64$
0111_B	$f_{ccu4}/128$
1000_B	$f_{ccu4}/256$
1001_B	$f_{ccu4}/512$
1010_B	$f_{ccu4}/1024$
1011_B	$f_{ccu4}/2048$
1100_B	$f_{ccu4}/4096$
1101_B	$f_{ccu4}/8192$
1110_B	$f_{ccu4}/16384$
1111_B	$f_{ccu4}/32768$

16.2.11.2 Floating Prescaler Mode

The floating prescaler mode can be used individually in each slice by setting the register **CC4yTC**.FPE = 1_B. With this mode, the user can not only achieve a better precision on the counter clock for compare operations but also reduce the SW read access for the capture mode.

The floating prescaler mode contains additionally to the initial configuration value register, **CC4yPSC**.PSIV, a compare register, **CC4yFPC**.PCMP with an associated shadow register mechanism.

Figure 16-57 shows the structure of the prescaler in floating mode when the specific slice is in compare mode (no external signal is used for capture). In this mode, the value of the clock division is increment by 1_D every time that a timer overflow/underflow (overflow if in Edge Aligned Mode, underflow if in Center Aligned Mode) occurs.

In this mode, the Compare Match from the timer is AND gated with the Compare Match of the prescaler and every time that this event occurs, the value of the clock division is updated with the **CC4yPSC**.PSIV value in the immediately next timer overflow/underflow event.

The shadow transfer of the floating prescaler compare value, **CC4yFPC**.PCMP, is done following the same rules described on **Section 16.2.5.2**.

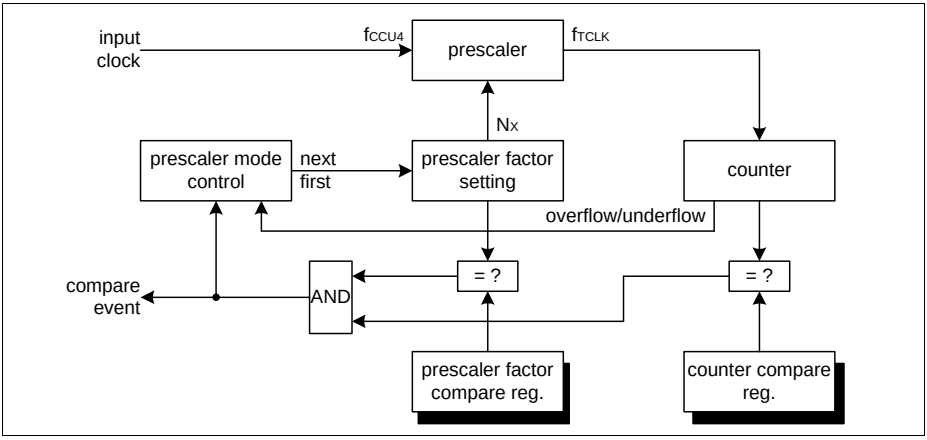


Figure 16-57 Floating prescaler in compare mode overview

When the specific CC4y is operating in capture mode (when at least one external signal is decoded as capture functionality), the actual value of the clock division also needs to be stored every time that a capture event occurs. The floating prescaler can have up to 4 capture registers (the maximum number of capture registers is dictated by the number of capture registers used in the specific slice).

The clock division value continues to be increment by 1_D every time that a timer overflow (in capture mode, the slice is always operating in Edge Aligned Mode) occurs and it is loaded with the PSIV value every time that a capture triggers is detected.

See the **Section 16.2.12.2** for a full description of the usage of the floating prescaler mode in conjunction with compare and capture modes.

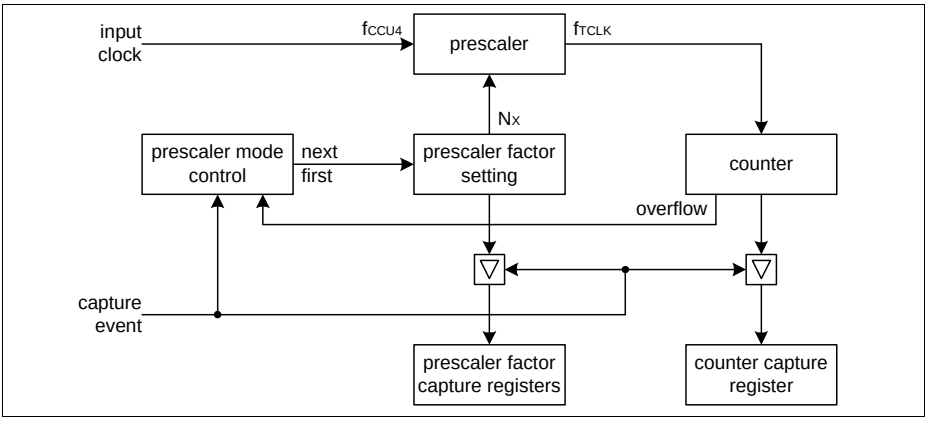


Figure 16-58 Floating Prescaler in capture mode overview

16.2.12 CCU4 Usage

16.2.12.1 PWM Signal Generation

The CCU4 offers a very flexible range in duty cycle configurations. This range is comprised between 0 to 100%.

To generate a PWM signal with a 100% duty cycle in Edge Aligned Mode, one should program the compare value, **CC4yCR**.CR, to 0000_H, see **Figure 16-59**.

In the same manner a 100% duty cycle signal can be generated in Center Aligned Mode, see **Figure 16-60**.

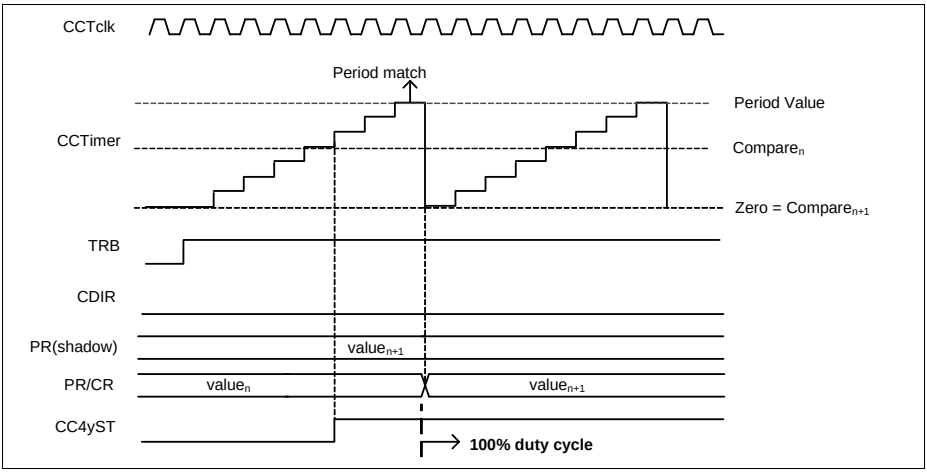


Figure 16-59 PWM with 100% duty cycle - Edge Aligned Mode

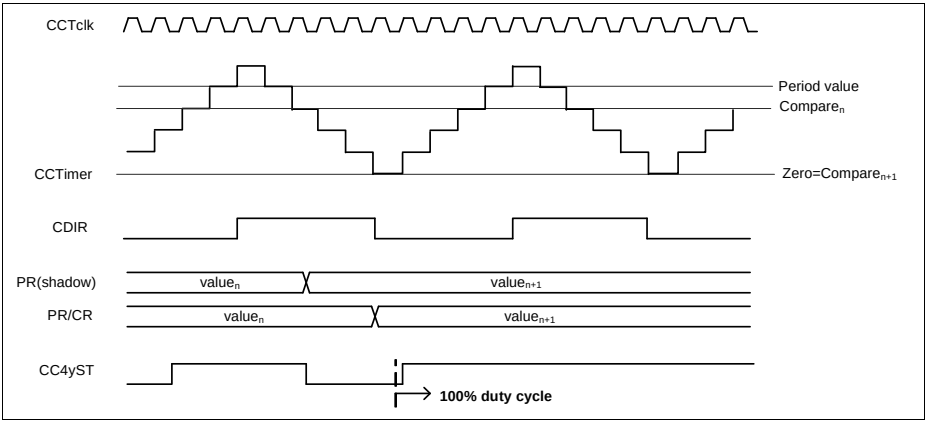


Figure 16-60 PWM with 100% duty cycle - Center Aligned Mode

To generate a PWM signal with 0% duty cycle in Edge Aligned Mode, the compare register should be set with the value programmed into the period value plus 1. In the case that the timer is being used with the full 16 bit capability (counting from 0 to 65535), setting a value bigger than the period value into the compare register is not possible and therefore the smallest duty cycle that can be achieved is 1/FS, see **Figure 16-61**.

In Center Aligned Mode, the counter is never running from 0 to 65535_D, due to the fact that it has to overshoot for one clock cycle the value set in the period register. Therefore the user never has a FS counter, which means that generating a 0% duty cycle signal is

always possible by setting a value in the compare register bigger than the one programmed into the period register, see **Figure 16-62**.

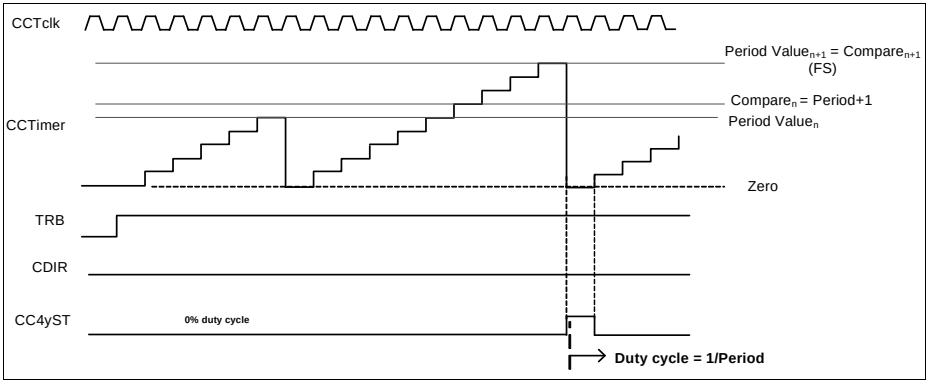


Figure 16-61 PWM with 0% duty cycle - Edge Aligned Mode

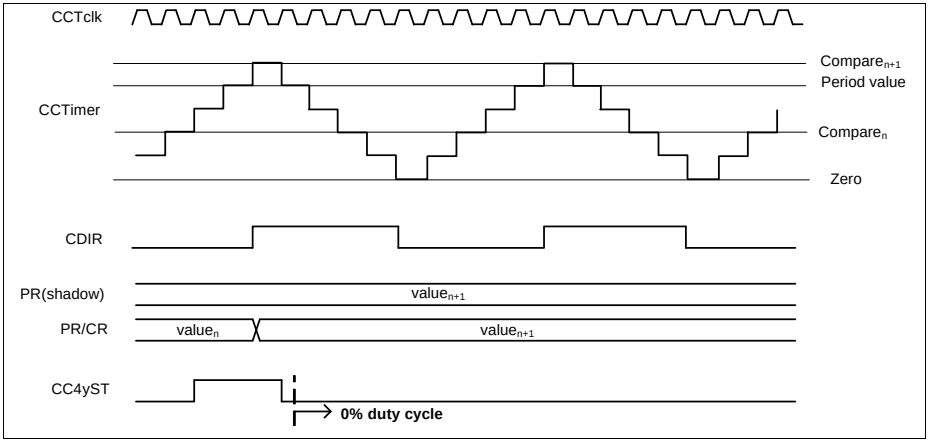


Figure 16-62 PWM with 0% duty cycle - Center Aligned Mode

16.2.12.2 Prescaler Usage

In Normal Prescaler Mode, the frequency of the f_{tclk} fed to the specific CC4y is chosen from the **Table 16-7**, by setting the **CC4yPSC**.PSIV with the required value.

In Floating Prescaler Mode, the frequency of the f_{tclk} can be modified over a selected timeframe, within the values specified in **Table 16-7**. This mechanism is specially useful if, when in capture mode, the dynamic of the capture triggers is very slow or unknown.

In Capture Mode, the Floating Prescaler value is incremented by 1 every time that a timer overflow happens and it is set with the initial programmed value when a capture event happens, see **Figure 16-63**.

When using the Floating Prescaler Mode in Capture Mode, the timer should be cleared each time that a capture event happens, **CC4yTC**.CAPC = 11_B. By operating the Capture mode in conjunction with the Floating Prescaler, even for capture signals that have a periodicity bigger that 16 bits, it is possible to use just a single CCU4 Timer Slice without monitoring the interrupt event of the timer overflow, cycle by cycle. For this the user just needs to know what is the timer captured value and the actual prescaler configuration at the time that the capture event occurred. These values are contained in each CC4yCxV register.

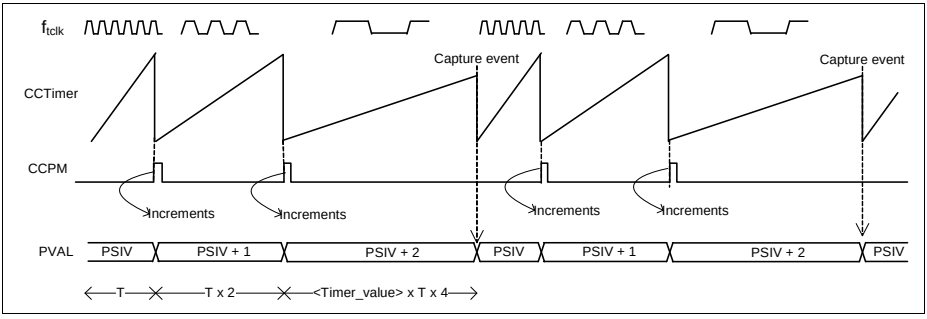


Figure 16-63 Floating Prescaler capture mode usage

When in Compare Mode, the Floating Prescaler function may be used to achieve a fractional PWM frequency or to perform some frequency modulation.

The same incrementing by 1_D mechanism is done every time that a overflow/underflow of the Timer occurs and the actual Prescaler value, doesn't match the one programmed into the **CC4yFPC**.PCMP register.

When a Compare Match from the Timer occurs and the actual Prescaler value is equal to the one programmed on the **CC4yFPC**.PCMP register, then the Prescaler value is set with the initial value, **CC4yPSC**.PSIV, when the next occurrence of a timer overflow/underflow.

In **Figure 16-64**, the Compare value of the Floating Prescaler was set to PSIV + 2. Every time that a timer overflow occurs, the value of the Prescaler is incremented by 1, which means that if we give f_{tclk} as the reference frequency for the **CC4yPSC**.PSIV value, we have $f_{tclk}/2$ for **CC4yPSC**.PSIV + 1 and $f_{tclk}/4$ for **CC4yPSC**.PSIV + 2.

The period over time of the counter becomes:

$$Period = (1/f_{tclk} + 2/f_{tclk} + 4/f_{tclk}) / 3 \quad (16.10)$$

The same mechanism is used in Center Aligned Mode, but to keep the rising arcade and falling arcade always symmetrical, instead of the overflow of the timer, the underflow is used, see **Figure 16-65**.

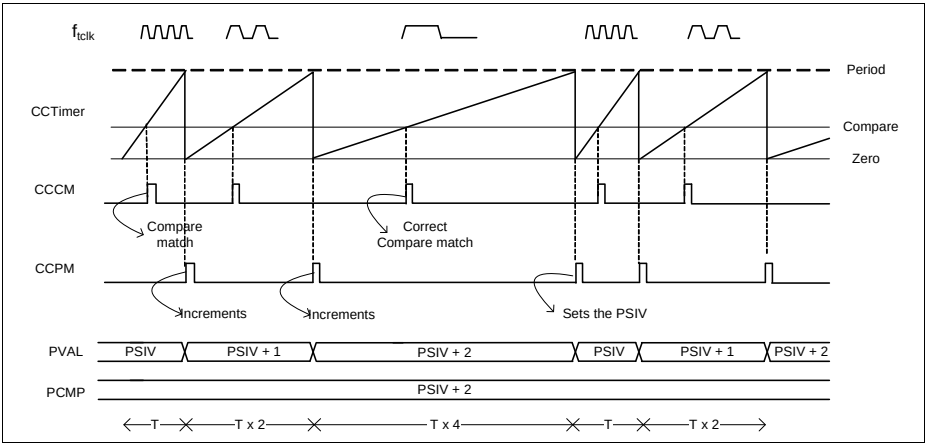


Figure 16-64 Floating Prescaler compare mode usage - Edge Aligned

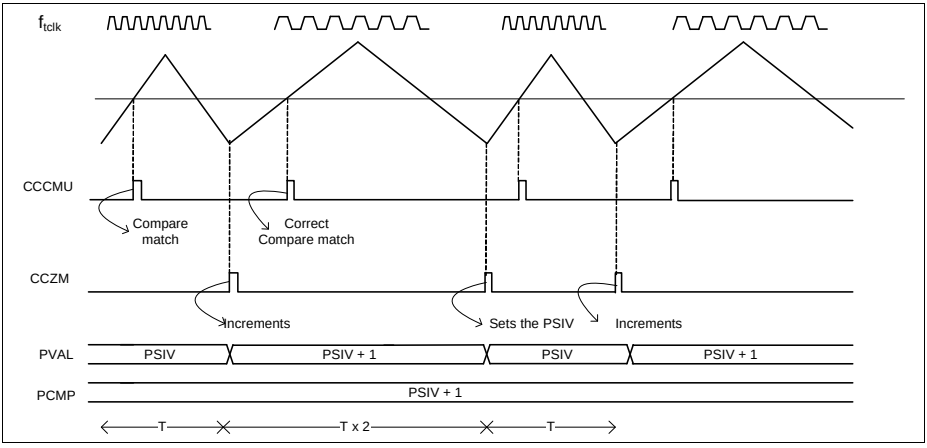


Figure 16-65 Floating Prescaler compare mode usage - Center Aligned

16.2.12.3 PWM Dither

The Dither functionality can be used to achieve a very fine precision on the periodicity of the output state in compare mode. The value set in the dither compare register, **CC4yDIT**.DCV is crosschecked against the actual value of the dither counter and every

time that the dither counter is smaller than the comparison value one of the follows actions is taken:

- The period is extended for 1 clock cycle - **CC4yTC**.DITHE = 01_B; in edge aligned mode
- The period is extended for 2 clock cycles - **CC4yTC**.DITHE = 01_B; in center aligned mode
- The comparison match while counting up (**CC4yTCST**.CDIR = 0_B) is delayed (this means that the status bit is going to stay in the SET state 1 cycle less) for 1 clock cycle - **CC4yTC**.DITHE = 10_B;
- The period is extended for 1 clock cycle and the comparison match while counting up is delayed for 1 clock cycle - **CC4yTC**.DITHE = 11_B; in edge aligned mode
- The period is extended for 2 clock cycles and the comparison match while counting up is delayed for 1 clock cycle; center aligned mode

The bit reverse counter distributes the number programmed in the **CC4yDIT**.DCV throughout a window of 16 timer periods.

Table 16-8 describes the bit reverse distribution versus the programmed value on the **CC4yDIT**.DCV field. The fields marked as '0' indicate that in that counter period, one of the above described actions, is going to be performed.

Table 16-8 Bit reverse distribution

	DCV															
Dither counter	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
8	1	1	1	1	1	1	1	1	1	**0**	**0**	**0**	**0**	**0**	**0**	**0**
4	1	1	1	1	1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
C	1	1	1	1	1	1	1	1	1	1	1	1	1	**0**	**0**	**0**
2	1	1	1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
A	1	1	1	1	1	1	1	1	1	1	1	**0**	**0**	**0**	**0**	**0**
6	1	1	1	1	1	1	1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
E	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**0**
1	1	1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
9	1	1	1	1	1	1	1	1	1	1	**0**	**0**	**0**	**0**	**0**	**0**
5	1	1	1	1	1	1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
D	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**0**	**0**
3	1	1	1	1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
B	1	1	1	1	1	1	1	1	1	1	1	1	**0**	**0**	**0**	**0**

Table 16-8 Bit reverse distribution (cont'd)

	DCV															
Dither counter	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
7	1	1	1	1	1	1	1	1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
F	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

The bit reverse distribution versus the programmed **CC4yDIT**.DCV value results in the following values for the Period and duty cycle:

DITHE = 01_B

$Period = [(16 - DCV) \times T + DCV \times (T + 1)]/16$; in Edge Aligned Mode (16.11)

$Duty\ cycle = [(16 - DCV) \times d/T + DCV \times (d+1)/(T + 1)]/16$; in Edge Aligned Mode (16.12)

$Period = [(16 - DCV) \times T + DCV \times (T + 2)]/16$; in Center Aligned Mode (16.13)

$Duty\ cycle = [(16 - DCV) \times d/T + DCV \times (d+2)/(T + 2)]/16$; in Center Aligned Mode (16.14)

DITHE = 10_B

$Period = T$; in Edge Aligned Mode (16.15)

$Duty\ cycle = [(16 - DCV) \times d/T + DCV \times (d-1)/T\ ]/16$; in Edge Aligned Mode (16.16)

$Period = T$; in Center Aligned Mode (16.17)

$Duty\ cycle = [(16 - DCV) \times d/T + DCV \times (d-1)/T]/16$; in Center Aligned Mode (16.18)

DITHE = 11_B

$Period = [(16 - DCV) \times T + DCV \times (T + 1)]/16$; in Edge Aligned Mode (16.19)

$Duty\ cycle = [(16 - DCV) \times d/T + DCV \times d/(T + 1)]/16$; in Edge Aligned Mode (16.20)

$Period = [(16 - DCV) \times T + DCV \times (T + 2)]/16$; in Center Aligned Mode (16.21)

$Duty\ cycle = [(16 - DCV) \times d/T + DCV \times (d+1)/(T + 2)]/16$; in Center Aligned Mode (16.22)

where:

T - Original period of the signal, see **Section 16.2.5.1**

d - Original duty cycle of the signal, see **Section 16.2.5.1**

16.2.12.4 Capture Mode Usage

Each Timer Slice can make use of 2 or 4 capture registers. Using only 2 capture registers means that only 1 Event was linked to a captured trigger. To use the four capture registers, both capture triggers need to be mapped into an Event (it can be the same signal with different edges selected or two different signals) or the **CC4yTC**.SCE field needs to be set to 1, which enables the linking of the 4 capture registers.

The internal slice mechanism for capturing is the same for the capture trigger 1 or capture trigger 0.

Different Capture Events - SCE = 0_B

Capture trigger 1 (CCcapt1) is appointed to the capture register 2, **CC4yC2V** and capture register 3, **CC4yC3V**, while trigger 0 (CCcapt0) is appointed to capture register 1, **CC4yC1V** and 0, **CC4yC0V**.

In each CCcapt0 event, the timer value is stored into **CC4yC1V** and the value of the **CC4yC1V** is transferred into the **CC4yC0V**.

In each CCcapt1 event, the timer value is stored into capture register **CC4yC3V** and the value of the capture register **CC4yC3V** is transferred into **CC4yC2V**.

The capture/transfer mechanism only happens if the specific register is not full. A capture register becomes full when receives a new value and becomes empty after the SW has read back the value.

The full flag is cleared every time that the SW reads back the **CC4yC0V**, **CC4yC1V**, **CC4yC2V** or **CC4yC3V** register. The SW can be informed of a new capture trigger by enabling the interrupt source linked to the specific Event. This means that every time that a capture is made an interrupt pulse is generated.

In the case that the Floating Prescaler Mode is being used, the actual value of the clock division is also stored in the capture register (CC4yCxV).

Figure 16-66 shows an example of how the capture/transfer may be used in a Timer Slice that is using a external signal as count function (to measure the velocity of a rotating device), and an equidistant capture trigger that is used to dictate the timestamp for the velocity calculation (two Timer waveforms are plotted, one that exemplifies the clearing of the timer in each capture event and another without the clearing function active).

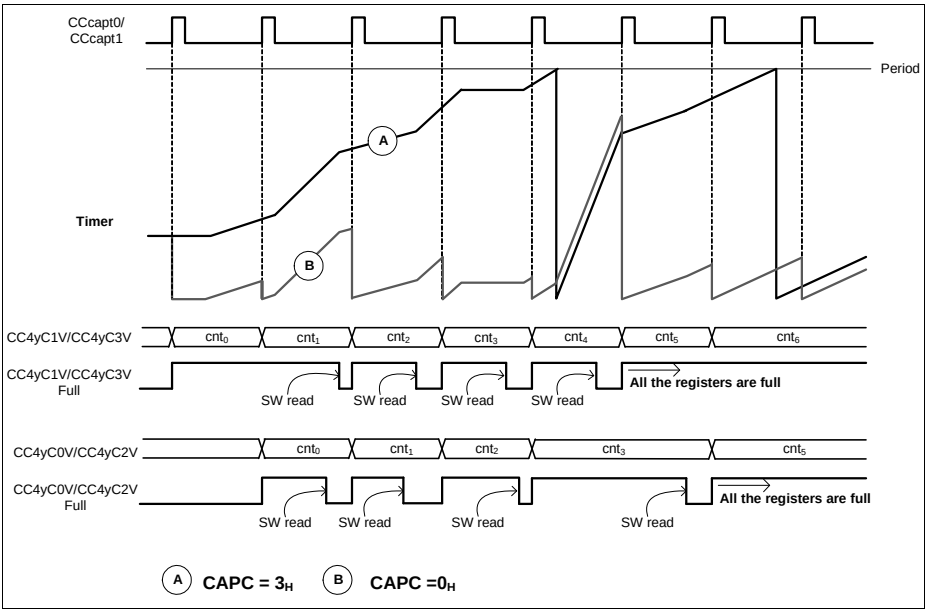


Figure 16-66 Capture mode usage - single channel

Same Capture Event - SCE = 1_B

If the **CC4yTC**.SCE is set to 1_B, all the four capture registers are chained together, emulating a fifo with a depth of 4. In this case, only the capture trigger 1, CCcapt1, is used to perform a capture event.

As an example for this mode, one can consider the case where one Timer Slice is being used in capture mode with SCE = 1_B, with another external signal that controls the counting. This timer slice can be incremented at different speeds, depending on the frequency of the counting signal.

An additional Timer Slice is used to control the capture trigger, dictating the time stamp for the capturing.

A simple scheme for this can be seen in **Figure 16-67**. The CC40ST output of slice 0 was used as capture trigger in the CC41 slice (active on rising and falling edge). The CC40ST output is used as known timebase marker, while the slice timer used for capture is being controlled by external events, e.g. external count.

Due to the fact that we have available 4 capture registers, every time that the SW reads back the complete set of values, 3 speed profiles can be measured.

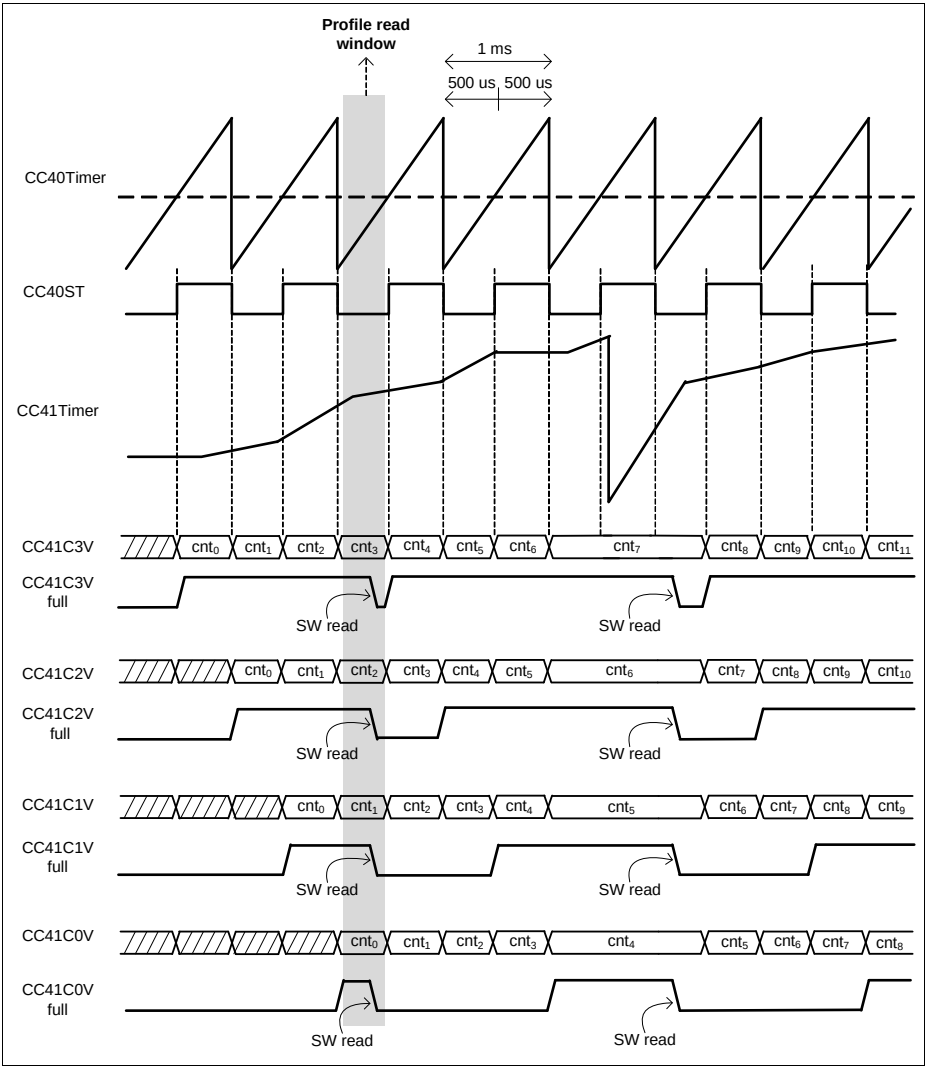


Figure 16-67 Three Capture profiles - CC4yTC.SCE = 1_B

To calculate the three different profiles in **Figure 16-67**, the 4 capture registers need to be read during the pointed read window. After that, the profile calculation is done:

Profile 1 = $CC41C1V_{info}$ - $CC41C0V_{info}$

Profile 2 = $CC41C2V_{info}$ - $CC41C1V_{info}$

Profile 3= $CC41C3V_{info}$ - $CC41C2V_{info}$

Note: This is an example and therefore several Timer Slice configurations and software loops can be implemented.

High Dynamics Capturing

In some cases the dynamics of the capture trigger(s) may vary greatly over time. This will impose that the software needs to be prepared for the worst case scenario, where the frequency of the capture triggers may be very high. In applications where cycle-by-cycle calculation is needed (calculation in each capture trigger), then this constraints needs to be met by the software. Nevertheless for applications where a cycle-by-cycle calculation is not needed, the software can read back the FIFO data register in a periodic base and fetch all the data that has been captured so far, **Figure 16-68**.

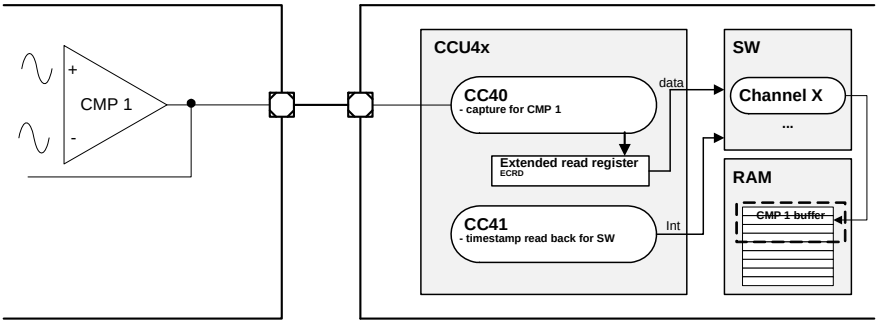


Figure 16-68 High dynamics capturing with software controlled timestamp

In this scenario, the software/CPU will read back the complete set of capture registers (2 or 4 depending on the chosen configuration), every time that an interrupt is triggered from the timestamp timer (the periodicity of this timer can also be adjusted on-the-fly).

Due to the fact that every capture register offers a full flag status bit, the software/CPU can always read back the complete set of registers. At the time of the data processing, this full flag is then checked, indicating if this value needs to be processed or not.

This FIFO read back functionality can also be used for applications that impose a heavy load on the system, which may not guarantee fixed access times to read back the captured data.

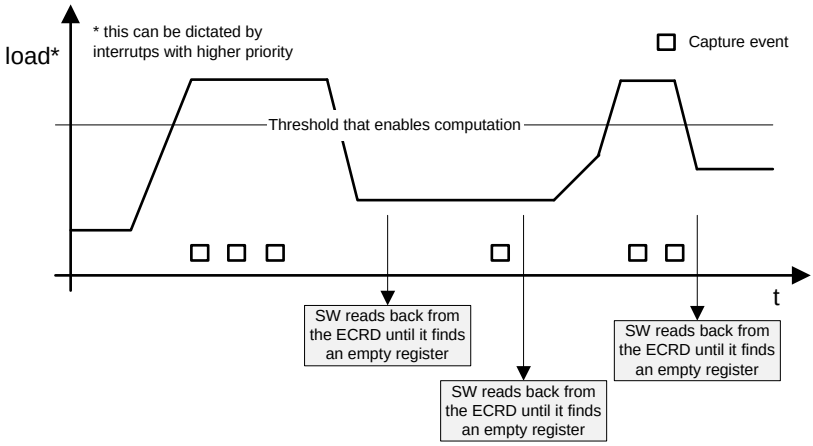


Figure 16-69 Extended read back during high load

Capture Grouping

In applications where multiple capture Timers are needed and the priority of the capture routines, does not imply that a cycle-by-cycle calculation needs to be done for every event, it may be suitable to group all the timers in the same CCU4x unit, **Figure 16-70**.

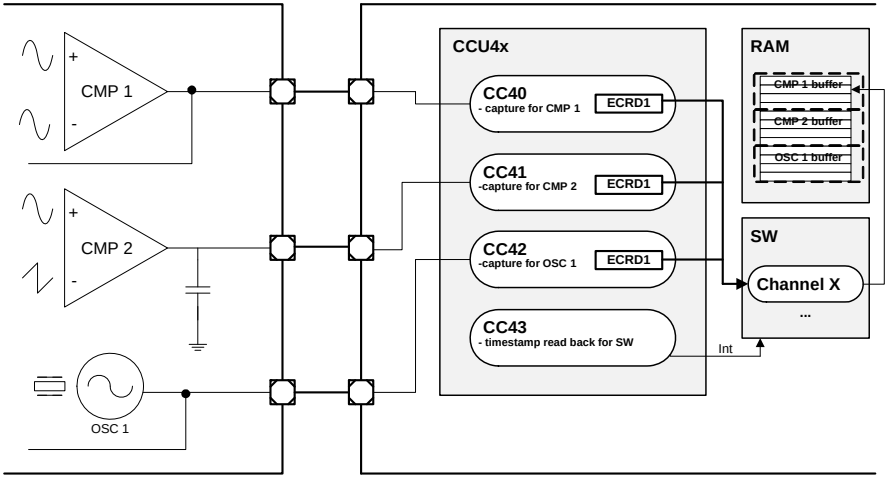


Figure 16-70 Capture grouping with extended read back

By setting the ECM bitfield for the TImer Slices used for capturing, the extended read back mode enables the reading back of data always in the proper capture order (from oldest to newest data). A timestamp timer is then used to trigger the Software/CPU to read back all the capture data present in the Timer Slices.

Every time that the interrupt is sensed, the Software/CPU (in this example) reads back the complete set of capture registers (via the ECRD address) for all Timer Slices. Due to the fact that each data read has a full flag indicator, the Software/CPU can read back the complete set of capture registers from all timers. This allows a fixed memory allocation that is as big as the number of captured registers, **Figure 16-71** (in this example 4 capture registers for each Timer Slice are being used).

The additional lost value bitfield (LCV) on the header of each ECRD data, will indicate if any capture trigger was lost between read operations (this can happen if the capture triggers are faster than the routine that is reading back the values).

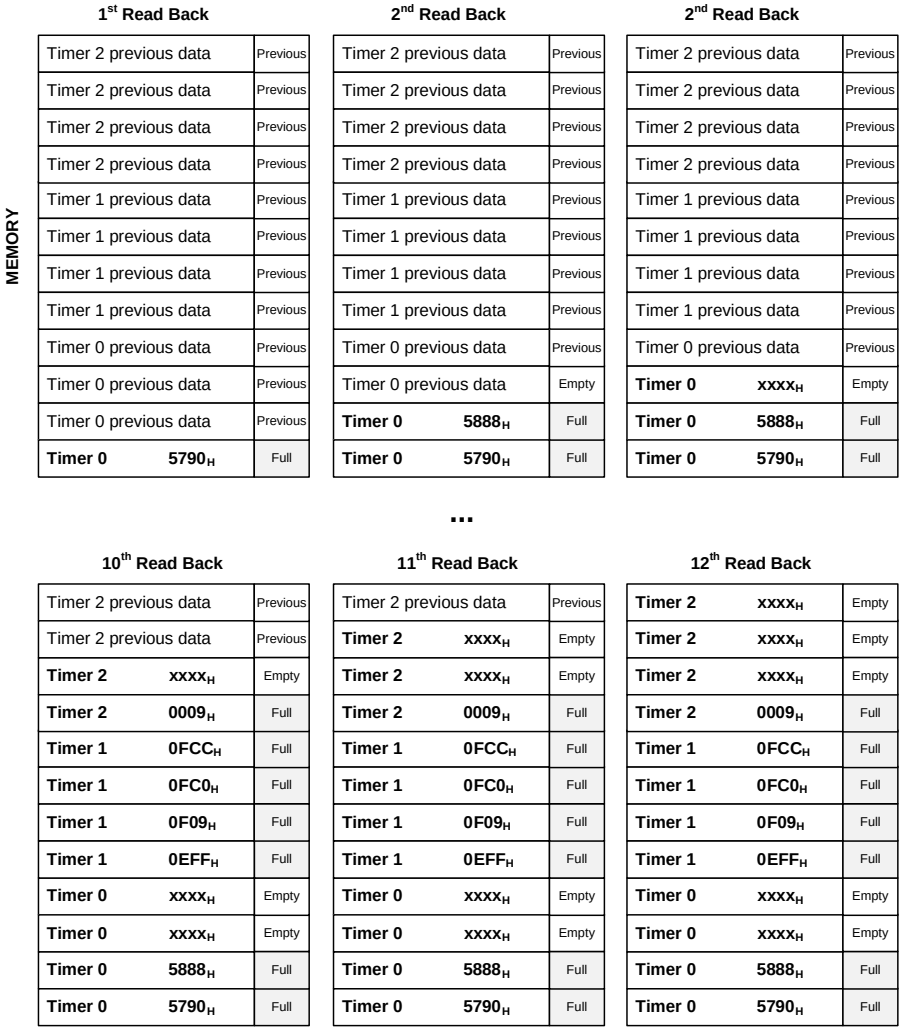


Figure 16-71 Memory structure for extended read back

16.3 Service Request Generation

Each CCU4 slice has an interrupt structure as the one in **Figure 16-72**. The register **CC4yINTS** is the status register for the interrupt sources. Each dedicated interrupt

source can be set or cleared by SW, by writing into the specific bit in the **CC4ySWS** and **CC4ySWR** registers respectively.

Each interrupt source can be enabled/disabled via the **CC4yINTE** register. An enabled interrupt source will always generate a pulse on the service request line even if the specific status bit was not cleared. **Table 16-9** describes the interrupt sources of each CCU4 slice.

The interrupt sources, Period Match while counting up and one Match while counting down are ORed together. The same mechanism is applied to the Compare Match while counting up and Compare Match while counting down.

The interrupt sources for the external events are directly linked with the configuration set on the **CC4yINS**.EVxEM. If an event is programmed to be active on both edges, that means that service request pulse is going to be generated when any transition on the external signal is detected. If the event is linked with a level function, the **CC4yINS**.EVxEM still can be programmed to enable a service request pulse. The TRAP event doesn't need any of extra configuration for generating the service request pulse when the slice enters the TRAP state.

Table 16-9 Interrupt sources

Signal	Description
CCINEV0_E	Event 0 edge(s) information from event selector. Used when an external signal should trigger an interrupt.
CCINEV1_E	Event 1 edge(s) information from event selector. Used when an external signal should trigger an interrupt.
CCINEV2_E	Event 2 edge(s) information from event selector. Used when an external signal should trigger an interrupt.
CCPM_U	Period Match while counting up
CCCM_U	Compare Match while counting up
CCCM_D	Compare Match while counting down
CCOM_D	One Match while counting down
Trap state set	Entering Trap State. Will set the E2AS

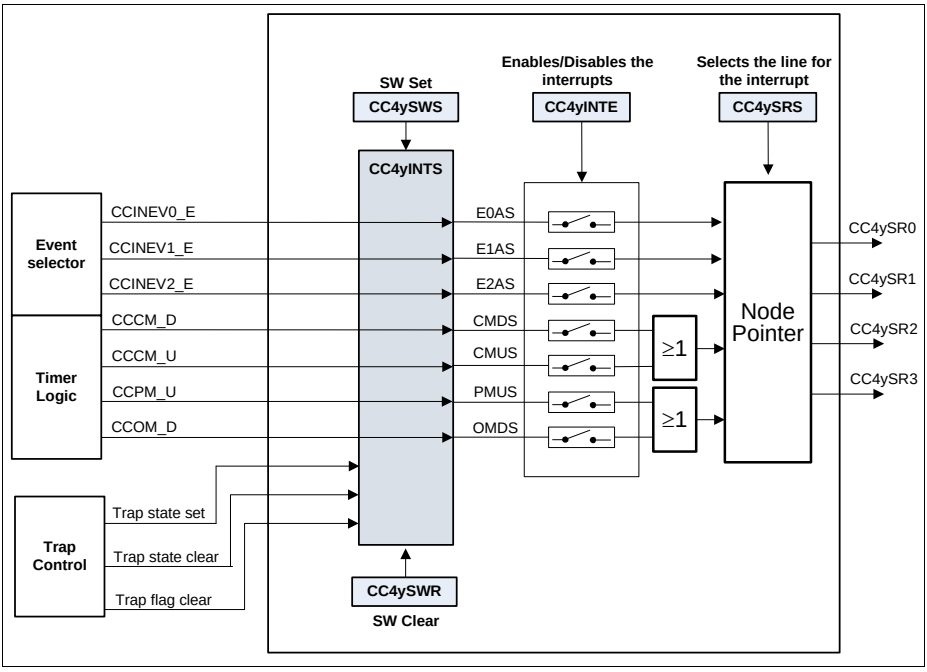


Figure 16-72 Slice interrupt structure overview

Each of the interrupt events can then be forwarded to one of the slice's four service request lines, **Figure 16-73**. The value set on the **CC4ySRS** controls which interrupt event is mapped into which service request line.

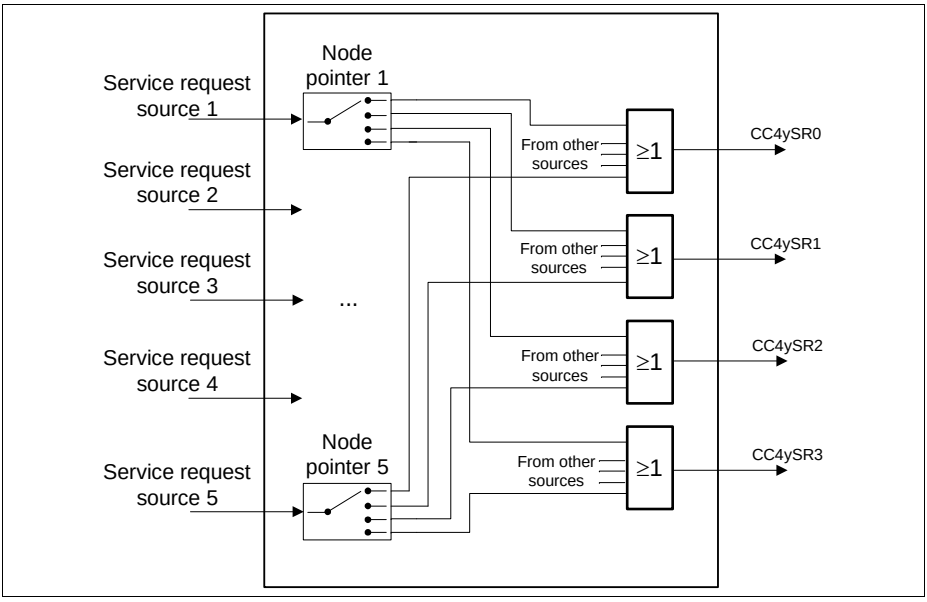


Figure 16-73 Slice Interrupt Node Pointer overview

The four service request lines of each slice are OR together inside the kernel of the CCU4, see **Figure 16-74**. This means that there are only four service request lines per CCU4, that can have in each line interrupt requests coming from different slices.

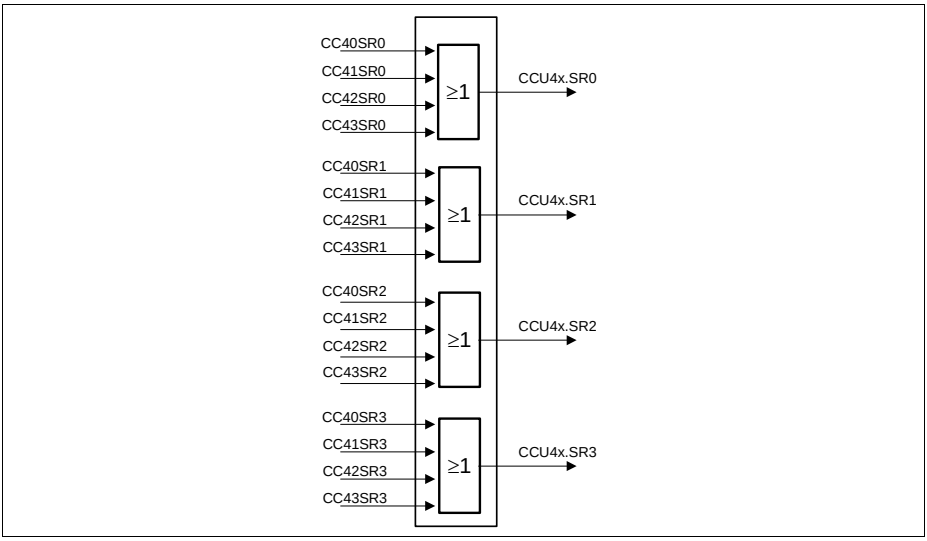


Figure 16-74 CCU4 service request overview

16.4 Debug Behavior

In suspend mode, the functional clocks for all slices as well the prescaler are stopped. The registers can still be accessed by the CPU (read only). This mode is useful for debugging purposes, e.g. where the current device status should be frozen in order to get a snapshot of the internal values. In suspend mode, all the slice timers are stopped. The suspend mode is non-intrusive concerning the register bits. This means register bits are not modified by hardware when entering or leaving the suspend mode.

Entry into suspend mode can be configured at the kernel level by means of the field **GCTRL**.SUSCFG.

The module is only functional after the suspend signal becomes inactive.

16.5 Power, Reset and Clock

The following sections describe the operating conditions, characteristics and timing requirements for the CCU4. All the timing information is related to the module clock, f_{ccu4}.

16.5.1 Clocks

Module Clock

The module clock of the CCU4 module is described in the SCU chapter as f_{PCLK}.

The bus interface clock of the CCU4 module is described in the SCU chapter as f_{MCLK}.

It is possible to disable the module clock for the CCU4 via the **GSTAT** register, nevertheless, there may be a dependency of the f_{ccu4} through the different CCU4 instances. One should address the SCU Chapter for a complete description of the product clock scheme.

If module clock dependencies exist through different IP instances, then one can disable the module clock internally inside the specific CCU4, by disabling the prescaler (**GSTAT**.PRB = 0_B).

External Clock

It is possible to use an external clock as source for the prescaler, and consequently for all the timer Slices, CC4y. This external source can be connected to one of the CCU4x.CLK[C...A] inputs.

This external source is nevertheless synchronized against f_{ccu4}.

Table 16-10 External clock operating conditions

Parameter	Symbol	Values			Unit	Note / Test Condition
		Min.	Typ.	Max.		
Frequency	f_{eclk}	–	–	$f_{ccu4}/4$	MHz	
ON time	ton_{eclk}	$2T_{ccu4}$ 1)2)	–	–	ns	
OFF time	$toff_{eclk}$	$2T_{ccu4}$ 1)2)	–	–	ns	Only the rising edge is used

1) Only valid if the signal was not previously synchronized/generated with the fccu4 clock (or a synchronous clock)

2) 50% duty cycle is not obligatory

16.5.2 Power

The CCU4 is inside the power core domain, therefore no special considerations about power up or power down sequences need to be taken. For an explanation about the different power domains, please address the SCU (System Control Unit) chapter.

An internal power down mode for the CCU4, can be achieved by disabling the clock inside the CCU4 itself. For this one should set the **GSTAT** register with the default reset value (via the idle mode set register, **GIDLS**).

16.6 Initialization and System Dependencies

16.6.1 Initialization Sequence

The initialization sequence for an application that is using the CCU4, should be the following:

1st Step: Enable the CCU4 clock via the specific SCU register, CGATCLR0.

2nd Step: Enable the prescaler block, by writing 1_B to the **GIDLC**.SPRB field.

3rd Step: Configure the global CCU4 register **GCTRL**

4th Step: Configure all the registers related to the required Timer Slice(s) functions, including the interrupt/service request configuration.

5th Step: If needed, configure the startup value for a specific Compare Channel Status, of a Timer Slice, by writing 1_B to the specific **GCSS**.SyTS.

6th Step: Enable the specific timer slice(s), CC4y, by writing 1_B to the specific **GIDLC**.CSyI.

7th Step: For all the Timer Slices that should be started synchronously via SW, the specific system register localized in the SCU, CCUCON, that enables a synchronous timer start should be addressed. The SCU.GLCSTxx input signal needs to be configured previously as a start function, see **Section 16.2.7.1**.

16.6.2 System Dependencies

Each CCU4 may have different dependencies regarding module and bus clock frequencies. This dependencies should be addressed in the SCU and System Architecture Chapters.

Dependencies between several peripherals, regarding different clock operating frequencies may also exist. This should be addressed before configuring the connectivity between the CCU4 and some other peripheral.

The following topics must be taken into consideration for good CCU4 and system operation:

- CCU4 module clock must be at maximum two times faster than the module bus interface clock
- Module input triggers for the CCU4 must not exceed the module clock frequency (if the triggers are generated internally in the device)
- Module input triggers for the CCU4 must not exceed the frequency dictated in **Section 16.5.1**
- Frequency of the CCU4 outputs used as triggers/functions on other modules, must be crosschecked on the end point
- Applying and removing CCU4 from reset, can cause unwanted operations in other modules. This can occur if the modules are using CCU4 outputs as triggers/functions.

16.7 Registers

Registers Overview

The absolute register address is calculated by adding:

Module Base Address + Offset Address

Table 16-11 Registers Address Space

Module	Base Address	End Address	Note
CCU40	48040000_H	$4804FFFF_H$	

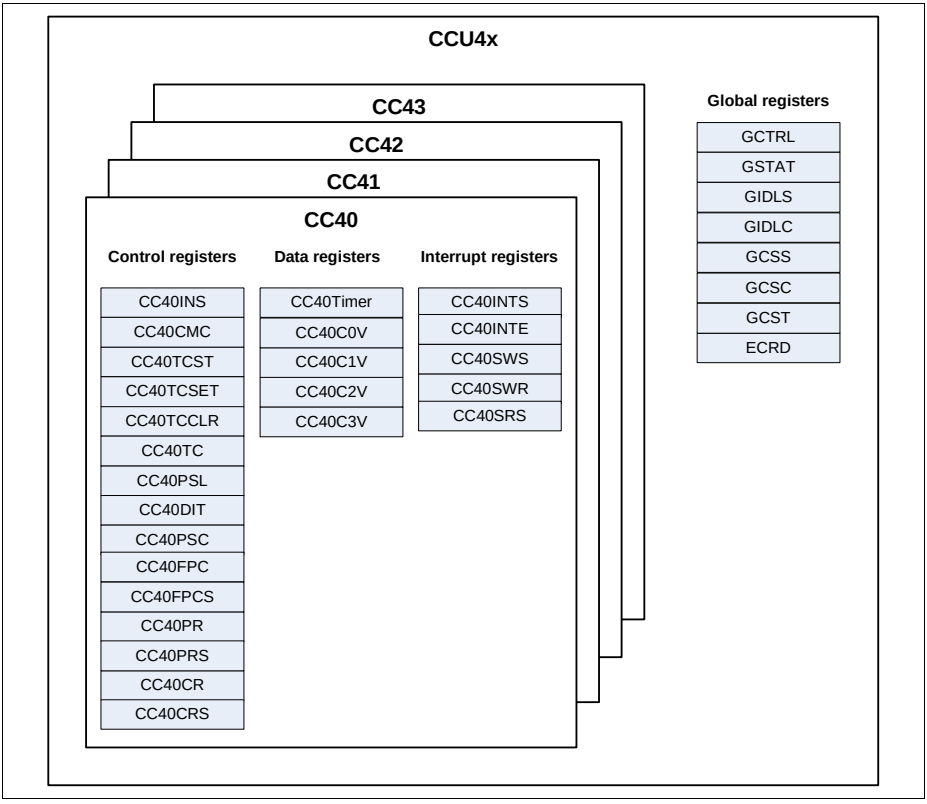


Figure 16-75 CCU4 registers overview

Table 16-12 Register Overview of CCU4

Short Name	Description	Offset Addr.[1)]	Access Mode		Description See
			Read	Write	
CCU4 Global Registers					
GCTRL	Module General Control Register	0000_H	U, PV	U, PV	Page 16-86
GSTAT	General Slice Status Register	0004_H	U, PV	BE	Page 16-89
GIDLS	General Idle Enable Register	0008_H	U, PV	U, PV	Page 16-90
GIDLC	General Idle Disable Register	$000C_H$	U, PV	U, PV	Page 16-92
GCSS	General Channel Set Register	0010_H	U, PV	U, PV	Page 16-93
GCSC	General Channel Clear Register	0014_H	U, PV	U, PV	Page 16-95
GCST	General Channel Status Register	0018_H	U, PV	BE	Page 16-98
MIDR	Module Identification Register	0080_H	U, PV	BE	Page 16-101
CC40 Registers					
CC40INS	Input Selector Unit Configuration	0100_H	U, PV	U, PV	Page 16-101
CC40CMC	Connection Matrix Configuration	0104_H	U, PV	U, PV	Page 16-103
CC40TST	Timer Run Status	0108_H	U, PV	BE	Page 16-106
CC40TCSET	Timer Run Set	$010C_H$	U, PV	U,PV	Page 16-107
CC40TCCLR	Timer Run Clear	0110_H	U, PV	U, PV	Page 16-108
CC40TC	General Timer Configuration	0114_H	U, PV	U, PV	Page 16-109
CC40PSL	Output Passive Level Configuration	0118_H	U, PV	U, PV	Page 16-114
CC40DIT	Dither Configuration	$011C_H$	U, PV	BE	Page 16-114
CC40DITS	Dither Shadow Register	0120_H	U, PV	U, PV	Page 16-115
CC40PSC	Prescaler Configuration	0124_H	U, PV	U, PV	Page 16-116
CC40FPC	Prescaler Compare Value	0128_H	U, PV	U, PV	Page 16-117

Table 16-12 Register Overview of CCU4 (cont'd)

Short Name	Description	Offset Addr.[1)]	Access Mode		Description See
			Read	Write	
CC40FPCS	Prescaler Shadow Compare Value	$012C_H$	U, PV	U, PV	Page 16-118
CC40PR	Timer Period Value	0130_H	U, PV	BE	Page 16-118
CC40PRS	Timer Period Shadow Value	0134_H	U, PV	U, PV	Page 16-119
CC40CR	Timer Compare Value	0138_H	U, PV	BE	Page 16-120
CC40CRS	Timer Compare Shadow Value	$013C_H$	U, PV	U, PV	Page 16-121
CC40TIMER	Timer Current Value	0170_H	U, PV	U, PV	Page 16-122
CC40C0V	Capture Register 0 Value	0174_H	U, PV	BE	Page 16-122
CC40C1V	Capture Register 1Value	0178_H	U, PV	BE	Page 16-123
CC40C2V	Capture Register 2 Value	$017C_H$	U, PV	BE	Page 16-124
CC40C3V	Capture Register 3 Value	0180_H	U, PV	BE	Page 16-125
CC40INTS	Interrupt Status	$01A0_H$	U, PV	BE	Page 16-126
CC40INTE	Interrupt Enable	$01A4_H$	U, PV	U, PV	Page 16-128
CC40SRS	Interrupt Configuration	$01A8_H$	U, PV	U, PV	Page 16-130
CC40SWS	Interrupt Status Set	$01AC_H$	U, PV	U, PV	Page 16-131
CC40SWR	Interrupt Status Clear	$01B0_H$	U, PV	U, PV	Page 16-133
CC40ECRD0	Extended Read Back 0	$01B8_H$	U, PV	BE	Page 16-134
CC40ECRD1	Extended Read Back 1	$01BC_H$	U, PV	BE	Page 16-136
CC41 Registers					
CC41INS	Input Selector Unit Configuration	0200_H	U, PV	U, PV	Page 16-101
CC41CMC	Connection Matrix Configuration	0204_H	U, PV	U, PV	Page 16-103
CC41TST	Timer Run Status	0208_H	U, PV	BE	Page 16-106
CC41TCSET	Timer Run Set	$020C_H$	U, PV	U,PV	Page 16-107
CC41TCCLR	Timer Run Clear	0210_H	U, PV	U, PV	Page 16-108
CC41TC	General Timer Configuration	0214_H	U, PV	U, PV	Page 16-109
CC41PSL	Output Passive Level Configuration	0218_H	U, PV	U, PV	Page 16-114

Table 16-12 Register Overview of CCU4 (cont'd)

Short Name	Description	Offset Addr.[1)]	Access Mode		Description See
			Read	Write	
CC41DIT	Dither Configuration	$021C_H$	U, PV	BE	Page 16-114
CC41DITS	Dither Shadow Register	0220_H	U, PV	U, PV	Page 16-115
CC41PSC	Prescaler Configuration	0224_H	U, PV	U, PV	Page 16-116
CC41FPC	Prescaler Compare Value	0228_H	U, PV	U, PV	Page 16-117
CC41FPCS	Prescaler Shadow Compare Value	$022C_H$	U, PV	U, PV	Page 16-118
CC41PR	Timer Period Value	0230_H	U, PV	BE	Page 16-118
CC41PRS	Timer Period Shadow Value	0234_H	U, PV	U, PV	Page 16-119
CC41CR	Timer Compare Value	0238_H	U, PV	BE	Page 16-120
CC41CRS	Timer Compare Shadow Value	$023C_H$	U, PV	U, PV	Page 16-121
CC41TIMER	Timer Current Value	0270_H	U, PV	U, PV	Page 16-122
CC41C0V	Capture Register 0 Value	0274_H	U, PV	BE	Page 16-122
CC41C1V	Capture Register 1Value	0278_H	U, PV	BE	Page 16-123
CC41C2V	Capture Register 2 Value	$027C_H$	U, PV	BE	Page 16-124
CC41C3V	Capture Register 3 Value	0280_H	U, PV	BE	Page 16-125
CC41INTS	Interrupt Status	$02A0_H$	U, PV	BE	Page 16-126
CC41INTE	Interrupt Enable	$02A4_H$	U, PV	U, PV	Page 16-128
CC41SRS	Interrupt Configuration	$02A8_H$	U, PV	U, PV	Page 16-130
CC41SWS	Interrupt Status Set	$02AC_H$	U, PV	U, PV	Page 16-131
CC41SWR	Interrupt Status Clear	$02B0_H$	U, PV	U, PV	Page 16-133
CC41ECRD0	Extended Read Back 0	$02B8_H$	U, PV	BE	Page 16-134
CC41ECRD1	Extended Read Back 1	$02BC_H$	U, PV	BE	Page 16-136
CC42 Registers					
CC42INS	Input Selector Unit Configuration	0300_H	U, PV	U, PV	Page 16-101
CC42CMC	Connection Matrix Configuration	0304_H	U, PV	U, PV	Page 16-103
CC42TST	Timer Run Status	0308_H	U, PV	BE	Page 16-106
CC42TCSET	Timer Run Set	$030C_H$	U, PV	U,PV	Page 16-107

Table 16-12 Register Overview of CCU4 (cont'd)

Short Name	Description	Offset Addr.[1)]	Access Mode		Description See
			Read	Write	
CC42TCCLR	Timer Run Clear	0310_H	U, PV	U, PV	Page 16-108
CC42TC	General Timer Configuration	0314_H	U, PV	U, PV	Page 16-109
CC42PSL	Output Passive Level Configuration	0318_H	U, PV	U, PV	Page 16-114
CC42DIT	Dither Configuration	$031C_H$	U, PV	BE	Page 16-114
CC42DITS	Dither Shadow Register	0320_H	U, PV	U, PV	Page 16-115
CC42PSC	Prescaler Configuration	0324_H	U, PV	U, PV	Page 16-116
CC42FPC	Prescaler Compare Value	0328_H	U, PV	U, PV	Page 16-117
CC42FPCS	Prescaler Shadow Compare Value	$032C_H$	U, PV	U, PV	Page 16-118
CC42PR	Timer Period Value	0330_H	U, PV	BE	Page 16-118
CC42PRS	Timer Period Shadow Value	0334_H	U, PV	U, PV	Page 16-119
CC42CR	Timer Compare Value	0338_H	U, PV	BE	Page 16-120
CC42CRS	Timer Compare Shadow Value	$033C_H$	U, PV	U, PV	Page 16-121
CC42TIMER	Timer Current Value	0370_H	U, PV	U, PV	Page 16-122
CC42C0V	Capture Register 0 Value	0374_H	U, PV	BE	Page 16-122
CC42C1V	Capture Register 1Value	0378_H	U, PV	BE	Page 16-123
CC42C2V	Capture Register 2 Value	$037C_H$	U, PV	BE	Page 16-124
CC42C3V	Capture Register 3 Value	0380_H	U, PV	BE	Page 16-125
CC42INTS	Interrupt Status	$03A0_H$	U, PV	BE	Page 16-126
CC42INTE	Interrupt Enable	$03A4_H$	U, PV	U, PV	Page 16-128
CC42SRS	Interrupt Configuration	$03A8_H$	U, PV	U, PV	Page 16-130
CC42SWS	Interrupt Status Set	$03AC_H$	U, PV	U, PV	Page 16-131
CC42SWR	Interrupt Status Clear	$03B0_H$	U, PV	U, PV	Page 16-133
CC42ECRD0	Extended Read Back 0	$03B8_H$	U, PV	BE	Page 16-134
CC42ECRD1	Extended Read Back 1	$03BC_H$	U, PV	BE	Page 16-136
CC43 Registers					
CC43INS	Input Selector Unit Configuration	0400_H	U, PV	U, PV	Page 16-101

Table 16-12 Register Overview of CCU4 (cont'd)

Short Name	Description	Offset Addr.[1)]	Access Mode		Description See
			Read	Write	
CC43CMC	Connection Matrix Configuration	0404_H	U, PV	U, PV	Page 16-103
CC43TST	Timer Run Status	0408_H	U, PV	BE	Page 16-106
CC43TCSET	Timer Run Set	$040C_H$	U, PV	U,PV	Page 16-107
CC43TCCLR	Timer Run Clear	0410_H	U, PV	U, PV	Page 16-108
CC43TC	General Timer Configuration	0414_H	U, PV	U, PV	Page 16-109
CC43PSL	Output Passive Level Configuration	0418_H	U, PV	U, PV	Page 16-114
CC43DIT	Dither Configuration	$041C_H$	U, PV	BE	Page 16-114
CC43DITS	Dither Shadow Register	0420_H	U, PV	U, PV	Page 16-115
CC43PSC	Prescaler Configuration	0424_H	U, PV	U, PV	Page 16-116
CC43FPC	Prescaler Compare Value	0428_H	U, PV	U, PV	Page 16-117
CC43FPCS	Prescaler Shadow Compare Value	$042C_H$	U, PV	U, PV	Page 16-118
CC43PR	Timer Period Value	0430_H	U, PV	BE	Page 16-118
CC43PRS	Timer Period Shadow Value	0434_H	U, PV	U, PV	Page 16-119
CC43CR	Timer Compare Value	0438_H	U, PV	BE	Page 16-120
CC43CRS	Timer Compare Shadow Value	$043C_H$	U, PV	U, PV	Page 16-121
CC43TIMER	Timer Current Value	0470_H	U, PV	U, PV	Page 16-122
CC43C0V	Capture Register 0 Value	0474_H	U, PV	BE	Page 16-122
CC43C1V	Capture Register 1Value	0478_H	U, PV	BE	Page 16-123
CC43C2V	Capture Register 2 Value	$047C_H$	U, PV	BE	Page 16-124
CC43C3V	Capture Register 3 Value	0480_H	U, PV	BE	Page 16-125
CC43INTS	Interrupt Status	$04A0_H$	U, PV	BE	Page 16-126
CC43INTE	Interrupt Enable	$04A4_H$	U, PV	U, PV	Page 16-128
CC43SRS	Interrupt Configuration	$04A8_H$	U, PV	U, PV	Page 16-130
CC43SWS	Interrupt Status Set	$04AC_H$	U, PV	U, PV	Page 16-131
CC43SWR	Interrupt Status Clear	$04B0_H$	U, PV	U, PV	Page 16-133

Table 16-12 Register Overview of CCU4 (cont'd)

Short Name	Description	Offset Addr.[1]	Access Mode		Description See
			Read	Write	
CC43ECRD0	Extended Read Back 0	$04B8_H$	U, PV	BE	Page 16-134
CC43ECRD1	Extended Read Back 1	$04BC_H$	U, PV	BE	Page 16-136

1) The absolute register address is calculated as follows:
Module Base Address + Offset Address (shown in this column)

16.7.1 Global Registers

GCTRL

The register contains the global configuration fields that affect all the timer slices inside CCU4.

GCTRL
Global Control Register **(0000_H)** **Reset Value: 00000000_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
MSDE		MSE3	MSE2	MSE1	MSE0	SUSCFG		0		PCIS		0	PRBC		
rw		rw	rw	rw	rw	rw		r		rw		r	rw		

Field	Bits	Type	Description
PRBC	[2:0]	rw	**Prescaler Clear Configuration** This register controls how the prescaler Run Bit and internal registers are cleared. 000_B SW only 001_B **GSTAT**.PRB and prescaler registers are cleared when the Run Bit of CC40 is cleared. 010_B **GSTAT**.PRB and prescaler registers are cleared when the Run Bit of CC41 is cleared. 011_B **GSTAT**.PRB and prescaler registers are cleared when the Run Bit of CC42 is cleared. 100_B **GSTAT**.PRB and prescaler registers are cleared when the Run Bit of CC43 is cleared.
PCIS	[5:4]	rw	**Prescaler Input Clock Selection** 00_B Module clock 01_B CCU4x.ECLKA 10_B CCU4x.ECLKB 11_B CCU4x.ECLKC
SUSCFG	[9:8]	rw	**Suspend Mode Configuration** This field controls the entering in suspend mode for all the CAPCOM4 slices. 00_B Suspend request ignored. The module never enters in suspend 01_B Stops all the running slices immediately. Safe stop is not applied. 10_B Stops the block immediately and clamps all the outputs to PASSIVE state. Safe stop is applied. 11_B Waits for the roll over of each slice to stop and clamp the slices outputs. Safe stop is applied.
MSE0	10	rw	**Slice 0 Multi Channel shadow transfer enable** When this field is set, a shadow transfer of slice 0 can be requested not only by SW but also via the CCU4x.MCSS input. 0_B Shadow transfer can only be requested by SW 1_B Shadow transfer can be requested via SW and via the CCU4x.MCSS input.

Field	Bits	Type	Description
MSE1	11	rw	**Slice 1 Multi Channel shadow transfer enable** When this field is set, a shadow transfer of slice 1 can be requested not only by SW but also via the CCU4x.MCSS input. 0_B Shadow transfer can only be requested by SW 1_B Shadow transfer can be requested via SW and via the CCU4x.MCSS input.
MSE2	12	rw	**Slice 2 Multi Channel shadow transfer enable** When this field is set, a shadow transfer of slice 2 can be requested not only by SW but also via the CCU4x.MCSS input. 0_B Shadow transfer can only be requested by SW 1_B Shadow transfer can be requested via SW and via the CCU4x.MCSS input.
MSE3	13	rw	**Slice 3 Multi Channel shadow transfer enable** When this field is set, a shadow transfer of slice 3 can be requested not only by SW but also via the CCU4x.MCSS input. 0_B Shadow transfer can only be requested by SW 1_B Shadow transfer can be requested via SW and via the CCU4x.MCSS input.
MSDE	[15:14]	rw	**Multi Channel shadow transfer request configuration** This field configures the type of shadow transfer requested via the CCU4x.MCSS input. The field **CC4yTC**.MSEy needs to be set in order for this configuration to have any effect. 00_B Only the shadow transfer for period and compare values is requested 01_B Shadow transfer for the compare, period and prescaler compare values is requested 10_B Reserved 11_B Shadow transfer for the compare, period, prescaler and dither compare values is requested

Field	Bits	Type	Description
0	3, [7:6], [31:16]	r	**Reserved** A read always returns 0.

GSTAT

The register contains the status of the prescaler and each timer slice (idle mode or running).

GSTAT
Global Status Register **(0004_H)** **Reset Value: $0000000F_H$**

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							PRB	0				S3I	S2I	S1I	S0I
r							rh	r				r	r	r	r

Field	Bits	Type	Description
S0I	0	r	**CC40 IDLE status** This bit indicates if the CC40 slice is in IDLE mode or not. In IDLE mode the clocks for the CC40 slice are stopped. 0_B Running 1_B Idle
S1I	1	r	**CC41 IDLE status** This bit indicates if the CC41 slice is in IDLE mode or not. In IDLE mode the clocks for the CC41 slice are stopped. 0_B Running 1_B Idle

Field	Bits	Type	Description
S2I	2	r	**CC42 IDLE status** This bit indicates if the CC42 slice is in IDLE mode or not. In IDLE mode the clocks for the CC42 slice are stopped. 0_B Running 1_B Idle
S3I	3	r	**CC43 IDLE status** This bit indicates if the CC43 slice is in IDLE mode or not. In IDLE mode the clocks for the CC43 slice are stopped. 0_B Running 1_B Idle
PRB	8	rh	**Prescaler Run Bit** 0_B Prescaler is stopped 1_B Prescaler is running
0	[7:4], [31:9]	r	**Reserved** Read always returns 0.

GIDLS

Through this register one can set the prescaler and the specific timer slices into idle mode.

GIDLS
Global Idle Set **(0008_H)** **Reset Value: 00000000_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						PSIC	CPRB	0				SS3I	SS2I	SS1I	SS0I
r						w	w	r				w	w	w	w

Field	Bits	Type	Description
SS0I	0	w	**CC40 IDLE mode set** Writing a 1_B to this bit sets the CC40 slice in IDLE mode. The clocks for the slice are stopped when in IDLE mode. When entering IDLE, the internal slice registers are not cleared. A read access always returns 0.
SS1I	1	w	**CC41 IDLE mode set** Writing a 1_B to this bit sets the CC41 slice in IDLE mode. The clocks for the slice are stopped when in IDLE mode. When entering IDLE, the internal slice registers are not cleared. A read access always returns 0.
SS2I	2	w	**CC42 IDLE mode set** Writing a 1_B to this bit sets the CC42 slice in IDLE mode. The clocks for the slice are stopped when in IDLE mode. When entering IDLE, the internal slice registers are not cleared. A read access always returns 0.
SS3I	3	w	**CC43 IDLE mode set** Writing a 1_B to this bit sets the CC43 slice in IDLE mode. The clocks for the slice are stopped when in IDLE mode. When entering IDLE, the internal slice registers are not cleared. A read access always returns 0.
CPRB	8	w	**Prescaler Run Bit Clear** Writing a 1_B into this register clears the Run Bit of the prescaler. Prescaler internal registers are not cleared. A read always returns 0.
PSIC	9	w	**Prescaler clear** Writing a 1_B to this register clears the prescaler counter. It also loads the PSIV into the PVAL field for all Timer Slices. This performs a re alignment of the timer clock for all Slices. The Run Bit of the prescaler is not cleared. A read always returns 0.
0	[7:4], [31:10]	r	**Reserved** Read always returns 0.

GIDLC

Through this register one can remove the prescaler and the specific timer slices from idle mode.

GIDLC
Global Idle Clear **($000C_H$)** **Reset Value: 00000000_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							SPRB	0				CS3I	CS2I	CS1I	CS0I
r							w	r				w	w	w	w

Field	Bits	Type	Description
CS0I	0	w	**CC40 IDLE mode clear** Writing a 1_B to this bit removes the CC40 from IDLE mode. A read access always returns 0.
CS1I	1	w	**CC41 IDLE mode clear** Writing a 1_B to this bit removes the CC41 from IDLE mode. A read access always returns 0.
CS2I	2	w	**CC42 IDLE mode clear** Writing a 1_B to this bit removes the CC42 from IDLE mode. A read access always returns 0.
CS3I	3	w	**CC43 IDLE mode clear** Writing a 1_B to this bit removes the CC43 from IDLE mode. A read access always returns 0.
SPRB	8	w	**Prescaler Run Bit Set** Writing a 1_B into this register sets the Run Bit of the prescaler. A read always returns 0.
0	[7:4], [31:9]	r	**Reserved** Read always returns 0.

GCSS

Through this register one can request a shadow transfer for the specific timer slice(s) and set the status bit for each of the compare channels.

GCSS
Global Channel Set **(0010_H)** **Reset Value: 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0												S3STS	S2STS	S1STS	S0STS
r												w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	S3PSE	S3DSE	S3SE	0	S2PSE	S2DSE	S2SE	0	S1PSE	S1DSE	S1SE	0	S0PSE	S0DSE	S0SE
r	w	w	w	r	w	w	w	r	w	w	w	r	w	w	w

Field	Bits	Type	Description
S0SE	0	w	**Slice 0 shadow transfer set enable** Writing a 1_B to this bit will set the **GCST**.S0SS field, enabling then a shadow transfer for the Period, Compare and Passive level values. A read always returns 0.
S0DSE	1	w	**Slice 0 Dither shadow transfer set enable** Writing a 1_B to this bit will set the **GCST**.S0DSS field, enabling then a shadow transfer for the Dither compare value. A read always returns 0.
S0PSE	2	w	**Slice 0 Prescaler shadow transfer set enable** Writing a 1_B to this bit will set the **GCST**.S0PSS field, enabling then a shadow transfer for the prescaler compare value. A read always returns 0.
S1SE	4	w	**Slice 1 shadow transfer set enable** Writing a 1_B to this bit will set the **GCST**.S1SS field, enabling then a shadow transfer for the Period, Compare and Passive level values. A read always returns 0.
S1DSE	5	w	**Slice 1 Dither shadow transfer set enable** Writing a 1_B to this bit will set the **GCST**.S1DSS field, enabling then a shadow transfer for the Dither compare value. A read always returns 0.

Field	Bits	Type	Description
S1PSE	6	w	**Slice 1 Prescaler shadow transfer set enable** Writing a 1_B to this bit will set the **GCST**.S1PSS field, enabling then a shadow transfer for the prescaler compare value. A read always returns 0.
S2SE	8	w	**Slice 2 shadow transfer set enable** Writing a 1_B to this bit will set the **GCST**.S2SS field, enabling then a shadow transfer for the Period, Compare and Passive level values. A read always returns 0.
S2DSE	9	w	**Slice 2 Dither shadow transfer set enable** Writing a 1_B to this bit will set the **GCST**.S2DSS field, enabling then a shadow transfer for the Dither compare value. A read always returns 0.
S2PSE	10	w	**Slice 2 Prescaler shadow transfer set enable** Writing a 1_B to this bit will set the **GCST**.S2PSS field, enabling then a shadow transfer for the prescaler compare value. A read always returns 0.
S3SE	12	w	**Slice 3 shadow transfer set enable** Writing a 1_B to this bit will set the **GCST**.S3SS field, enabling then a shadow transfer for the Period, Compare and Passive level values. A read always returns 0.
S3DSE	13	w	**Slice 3 Dither shadow transfer set enable** Writing a 1_B to this bit will set the **GCST**.S3DSS field, enabling then a shadow transfer for the Dither compare value. A read always returns 0.
S3PSE	14	w	**Slice 3 Prescaler shadow transfer set enable** Writing a 1_B to this bit will set the **GCST**.S3PSS field, enabling then a shadow transfer for the prescaler compare value. A read always returns 0.
S0STS	16	w	**Slice 0 status bit set** Writing a 1_B into this field sets the status bit of slice 0 (**GCST**.CC40ST) to 1_B. A read always returns 0.

Field	Bits	Type	Description
S1STS	17	w	**Slice 1 status bit set** Writing a 1_B into this field sets the status bit of slice 1 (**GCST**.CC41ST) to 1_B. A read always returns 0.
S2STS	18	w	**Slice 2 status bit set** Writing a 1_B into this field sets the status bit of slice 2 (**GCST**.CC42ST) to 1_B. A read always returns 0.
S3STS	19	w	**Slice 3 status bit set** Writing a 1_B into this field sets the status bit of slice 3 (**GCST**.CC43ST) to 1_B. A read always returns 0.
0	3, 7, 11, 15, [31:20]	r	**Reserved** Read always returns 0.

GCSC

Through this register one can reset a shadow transfer request for the specific timer slice and clear the status bit for each the compare channels.

GCSC
Global Channel Clear **(0014_H)** **Reset Value: 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0												S3STC	S2STC	S1STC	S0STC
r												w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	S3PSC	S3DSC	S3SC	0	S2PSC	S2DSC	S2SC	0	S1PSC	S1DSC	S1SC	0	S0PSC	S0DSC	S0SC
r	w	w	w	r	w	w	w	r	w	w	w	r	w	w	w

Field	Bits	Type	Description
S0SC	0	w	**Slice 0 shadow transfer clear** Writing a 1_B to this bit will clear the **GCST**.S0SS field, canceling any pending shadow transfer for the Period, Compare and Passive level values. A read always returns 0.
S0DSC	1	w	**Slice 0 Dither shadow transfer clear** Writing a 1_B to this bit will clear the **GCST**.S0DSS field, canceling any pending shadow transfer for the Dither compare value. A read always returns 0.
S0PSC	2	w	**Slice 0 Prescaler shadow transfer clear** Writing a 1_B to this bit will clear the **GCST**.S0PSS field, canceling any pending shadow transfer for the prescaler compare value. A read always returns 0.
S1SC	4	w	**Slice 1 shadow transfer clear** Writing a 1_B to this bit will clear the **GCST**.S1SS field, canceling any pending shadow transfer for the Period, Compare and Passive level values. A read always returns 0.
S1DSC	5	w	**Slice 1 Dither shadow transfer clear** Writing a 1_B to this bit will clear the **GCST**.S1DSS field, canceling any pending shadow transfer for the Dither compare value. A read always returns 0.
S1PSC	6	w	**Slice 1 Prescaler shadow transfer clear** Writing a 1_B to this bit will clear the **GCST**.S1PSS field, canceling any pending shadow transfer for the prescaler compare value. A read always returns 0.
S2SC	8	w	**Slice 2 shadow transfer clear** Writing a 1_B to this bit will clear the **GCST**.S2SS field, canceling any pending shadow transfer for the Period, Compare and Passive level values. A read always returns 0.

Field	Bits	Type	Description
S2DSC	9	w	**Slice 2 Dither shadow transfer clear** Writing a 1_B to this bit will clear the **GCST**.S2DSS field, canceling any pending shadow transfer for the Dither compare value. A read always returns 0.
S2PSC	10	w	**Slice 2 Prescaler shadow transfer clear** Writing a 1_B to this bit will clear the **GCST**.S2PSS field, canceling any pending shadow transfer for the prescaler compare value. A read always returns 0.
S3SC	12	w	**Slice 3 shadow transfer clear** Writing a 1_B to this bit will clear the **GCST**.S3SS field, canceling any pending shadow transfer for the Period, Compare and Passive level values. A read always returns 0.
S3DSC	13	w	**Slice 3 Dither shadow transfer clear** Writing a 1_B to this bit will clear the **GCST**.S3DSS field, canceling any pending shadow transfer for the Dither compare value. A read always returns 0.
S3PSC	14	w	**Slice 3 Prescaler shadow transfer clear** Writing a 1_B to this bit will clear the **GCST**.S3PSS field, canceling any pending shadow transfer for the prescaler compare value. A read always returns 0.
S0STC	16	w	**Slice 0 status bit clear** Writing a 1_B into this field clears the status bit of slice 0 (**GCST**.CC40ST) to 0_B. A read always returns 0.
S1STC	17	w	**Slice 1 status bit clear** Writing a 1_B into this field clears the status bit of slice 1 (**GCST**.CC41ST) to 0_B. A read always returns 0.
S2STC	18	w	**Slice 2 status bit clear** Writing a 1_B into this field clears the status bit of slice 2 (**GCST**.CC42ST) to 0_B. A read always returns 0.

Field	Bits	Type	Description
S3STC	19	w	**Slice 3 status bit clear** Writing a 1_B into this field clears the status bit of slice 3 (**GCST**.CC43ST) to 0_B. A read always returns 0.
0	3, 7, 11, 15, [31:20]	r	**Reserved** Read always returns 0.

GCST

This register holds the information of the shadow transfer requests and of each timer slice status bit.

GCST
Global Channel Status **(0018_H)** **Reset Value: 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0												CC43ST	CC42ST	CC41ST	CC40ST
r												rh	rh	rh	rh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	S3PSS	S3DSS	S3SS	0	S2PSS	S2DSS	S2SS	0	S1PSS	S1DSS	S1SS	0	S0PSS	S0DSS	S0SS
r	rh	rh	rh	r	rh	rh	rh	r	rh	rh	rh	r	rh	rh	rh

Field	Bits	Type	Description
S0SS	0	rh	**Slice 0 shadow transfer status** 0_B Shadow transfer has not been requested 1_B Shadow transfer has been requested This field is cleared by HW after the requested shadow transfer has been executed.
S0DSS	1	rh	**Slice 0 Dither shadow transfer status** 0_B Dither shadow transfer has not been requested 1_B Dither shadow transfer has been requested This field is cleared by HW after the requested shadow transfer has been executed.

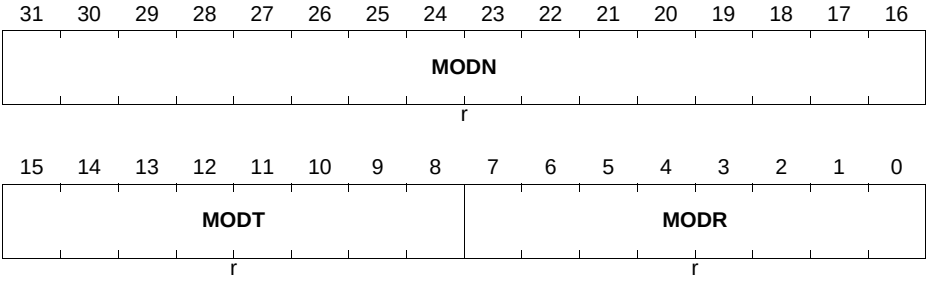
Field	Bits	Type	Description
S0PSS	2	rh	**Slice 0 Prescaler shadow transfer status** 0_B Prescaler shadow transfer has not been requested 1_B Prescaler shadow transfer has been requested This field is cleared by HW after the requested shadow transfer has been executed.
S1SS	4	rh	**Slice 1 shadow transfer status** 0_B Shadow transfer has not been requested 1_B Shadow transfer has been requested This field is cleared by HW after the requested shadow transfer has been executed.
S1DSS	5	rh	**Slice 1 Dither shadow transfer status** 0_B Dither shadow transfer has not been requested 1_B Dither shadow transfer has been requested This field is cleared by HW after the requested shadow transfer has been executed.
S1PSS	6	rh	**Slice 1 Prescaler shadow transfer status** 0_B Prescaler shadow transfer has not been requested 1_B Prescaler shadow transfer has been requested This field is cleared by HW after the requested shadow transfer has been executed.
S2SS	8	rh	**Slice 2 shadow transfer status** 0_B Shadow transfer has not been requested 1_B Shadow transfer has been requested This field is cleared by HW after the requested shadow transfer has been executed.
S2DSS	9	rh	**Slice 2 Dither shadow transfer status** 0_B Dither shadow transfer has not been requested 1_B Dither shadow transfer has been requested This field is cleared by HW after the requested shadow transfer has been executed.

Field	Bits	Type	Description
S2PSS	10	rh	**Slice 2 Prescaler shadow transfer status** 0_B Prescaler shadow transfer has not been requested 1_B Prescaler shadow transfer has been requested This field is cleared by HW after the requested shadow transfer has been executed.
S3SS	12	rh	**Slice 3 shadow transfer status** 0_B Shadow transfer has not been requested 1_B Shadow transfer has been requested This field is cleared by HW after the requested shadow transfer has been executed.
S3DSS	13	rh	**Slice 3 Dither shadow transfer status** 0_B Dither shadow transfer has not been requested 1_B Dither shadow transfer has been requested This field is cleared by HW after the requested shadow transfer has been executed.
S3PSS	14	rh	**Slice 3 Prescaler shadow transfer status** 0_B Prescaler shadow transfer has not been requested 1_B Prescaler shadow transfer has been requested This field is cleared by HW after the requested shadow transfer has been executed.
CC40ST	16	rh	**Slice 0 status bit**
CC41ST	17	rh	**Slice 1 status bit**
CC42ST	18	rh	**Slice 2 status bit**
CC43ST	19	rh	**Slice 3 status bit**
0	3, 7, 11, 15, [31:20]	r	**Reserved** Read always returns 0.

MIDR

This register contains the module identification number.

MIDR
Module Identification **(0080_H)** **Reset Value: $00A6C0XX_H$**



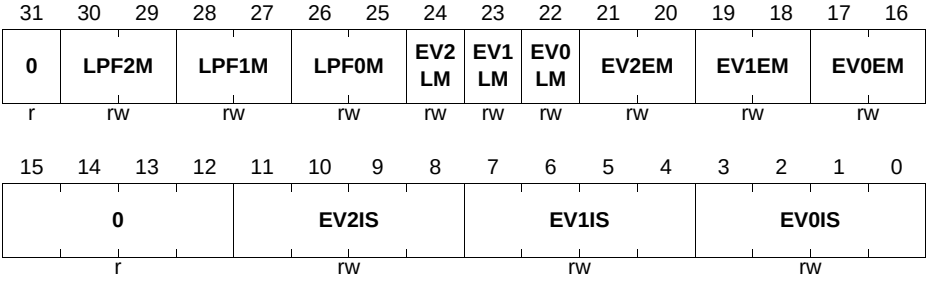
Field	Bits	Type	Description
MODR	[7:0]	r	**Module Revision** This bit field indicates the revision number of the module implementation (depending on the design step). The given value of 00_H is a placeholder for the actual number.
MODT	[15:8]	r	**Module Type**
MODN	[31:16]	r	**Module Number**

16.7.2 Slice (CC4y) Registers

CC4yINS

The register contains the configuration for the input selector.

CC4yINS (y = 0 - 3)
Input Selector Configuration **($0100_H + 0100_H * y$)** **Reset Value: 00000000_H**



Field	Bits	Type	Description
EV0IS	[3:0]	rw	**Event 0 signal selection** This field selects which pins is used for the event 0. 0000_B CCU4x.INyA 0001_B CCU4x.INyB 0010_B CCU4x.INyC 0011_B CCU4x.INyD 0100_B CCU4x.INyE 0101_B CCU4x.INyF 0110_B CCU4x.INyG 0111_B CCU4x.INyH 1000_B CCU4x.INyI 1001_B CCU4x.INyJ 1010_B CCU4x.INyK 1011_B CCU4x.INyL 1100_B CCU4x.INyM 1101_B CCU4x.INyN 1110_B CCU4x.INyO 1111_B CCU4x.INyP
EV1IS	[7:4]	rw	**Event 1 signal selection** Same as EV0IS description
EV2IS	[11:8]	rw	**Event 2 signal selection** Same as EV0IS description
EV0EM	[17:16]	rw	**Event 0 Edge Selection** 00_B No action 01_B Signal active on rising edge 10_B Signal active on falling edge 11_B Signal active on both edges
EV1EM	[19:18]	rw	**Event 1 Edge Selection** Same as EV0EM description
EV2EM	[21:20]	rw	**Event 2 Edge Selection** Same as EV0EM description
EV0LM	22	rw	**Event 0 Level Selection** 0_B Active on HIGH level 1_B Active on LOW level
EV1LM	23	rw	**Event 1 Level Selection** Same as EV0LM description

Field	Bits	Type	Description
EV2LM	24	rw	**Event 2 Level Selection** Same as EV0LM description
LPF0M	[26:25]	rw	**Event 0 Low Pass Filter Configuration** This field sets the number of consecutive counts for the Low Pass Filter of Event 0. The input signal value needs to remain stable for this number of counts (f_{CCU4}), so that a level/transition is accepted. 00_B LPF is disabled 01_B 3 clock cycles of f_{CCU4} 10_B 5 clock cycles of f_{CCU4} 11_B 7 clock cycles of f_{CCU4}
LPF1M	[28:27]	rw	**Event 1 Low Pass Filter Configuration** Same description as LPF0M
LPF2M	[30:29]	rw	**Event 2 Low Pass Filter Configuration** Same description as LPF0M
0	[15:12], 31	r	**Reserved** Read always returns 0.

CC4yCMC

The register contains the configuration for the connection matrix.

CC4yCMC (y = 0 - 3)
Connection Matrix Control **($0104_H + 0100_H * y$)** **Reset Value: 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0											TCE	MOS		TS	OFS
r											rw	rw		rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNTS		LDS		UDS		GATES		CAP1S		CAP0S		ENDS		STRTS	
rw		rw		rw		rw		rw		rw		rw		rw	

Field	Bits	Type	Description
STRTS	[1:0]	rw	**External Start Functionality Selector** Selects the Event that is going to be linked with the external start functionality. 00_B External Start Function deactivated 01_B External Start Function triggered by Event 0 10_B External Start Function triggered by Event 1 11_B External Start Function triggered by Event 2
ENDS	[3:2]	rw	**External Stop Functionality Selector** Selects the Event that is going to be linked with the external stop functionality. 00_B External Stop Function deactivated 01_B External Stop Function triggered by Event 0 10_B External Stop Function triggered by Event 1 11_B External Stop Function triggered by Event 2
CAP0S	[5:4]	rw	**External Capture 0 Functionality Selector** Selects the Event that is going to be linked with the external capture for capture registers number 1 and 0. 00_B External Capture 0 Function deactivated 01_B External Capture 0 Function triggered by Event 0 10_B External Capture 0 Function triggered by Event 1 11_B External Capture 0 Function triggered by Event 2
CAP1S	[7:6]	rw	**External Capture 1 Functionality Selector** Selects the Event that is going to be linked with the external capture for capture registers number 3 and 2. 00_B External Capture 1 Function deactivated 01_B External Capture 1 Function triggered by Event 0 10_B External Capture 1 Function triggered by Event 1 11_B External Capture 1 Function triggered by Event 2

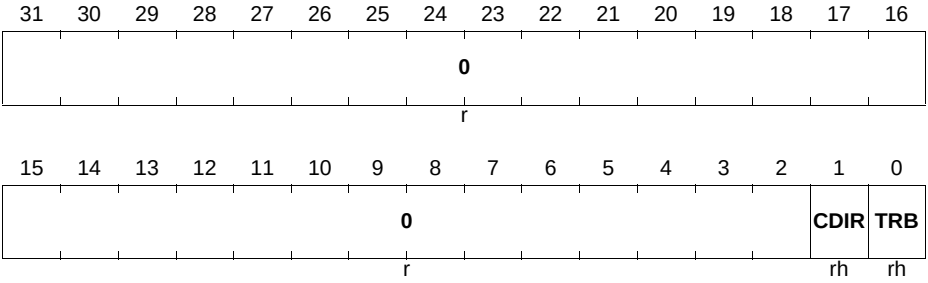
Field	Bits	Type	Description
GATES	[9:8]	rw	**External Gate Functionality Selector** Selects the Event that is going to be linked with the counter gating function. This function is used to gate the timer increment/decrement procedure. 00_B External Gating Function deactivated 01_B External Gating Function triggered by Event 0 10_B External Gating Function triggered by Event 1 11_B External Gating Function triggered by Event 2
UDS	[11:10]	rw	**External Up/Down Functionality Selector** Selects the Event that is going to be linked with the Up/Down counting direction control. 00_B External Up/Down Function deactivated 01_B External Up/Down Function triggered by Event 0 10_B External Up/Down Function triggered by Event 1 11_B External Up/Down Function triggered by Event 2
LDS	[13:12]	rw	**External Timer Load Functionality Selector** Selects the Event that is going to be linked with the timer load function. 00_B - External Load Function deactivated 01_B - External Load Function triggered by Event 0 10_B - External Load Function triggered by Event 1 11_B - External Load Function triggered by Event 2
CNTS	[15:14]	rw	**External Count Selector** Selects the Event that is going to be linked with the count function. The counter is going to be increment/decremented each time that a specific transition on the event is detected. 00_B External Count Function deactivated 01_B External Count Function triggered by Event 0 10_B External Count Function triggered by Event 1 11_B External Count Function triggered by Event 2
OFS	16	rw	**Override Function Selector** This field enables the ST bit override functionality. 0_B Override functionality disabled 1_B Status bit trigger override connected to Event 1; Status bit value override connected to Event 2

Field	Bits	Type	Description
TS	17	rw	**Trap Function Selector** This field enables the trap functionality. 0_B Trap function disabled 1_B TRAP function connected to Event 2
MOS	[19:18]	rw	**External Modulation Functionality Selector** Selects the Event that is going to be linked with the external modulation function. 00_B - Modulation Function deactivated 01_B - Modulation Function triggered by Event 0 10_B - Modulation Function triggered by Event 1 11_B - Modulation Function triggered by Event 2
TCE	20	rw	**Timer Concatenation Enable** This bit enables the timer concatenation with the previous slice. 0_B Timer concatenation is disabled 1_B Timer concatenation is enabled *Note: In CC40 this field doesn't exist. This is a read only reserved field. Read access always returns 0.*
0	[31:21]	r	**Reserved** A read always returns 0

CC4yTCST

The register holds the status of the timer (running/stopped) and the information about the counting direction (up/down).

CC4yTCST (y = 0 - 3)
Slice Timer Status **($0108_H + 0100_H * y$)** **Reset Value: 00000000_H**

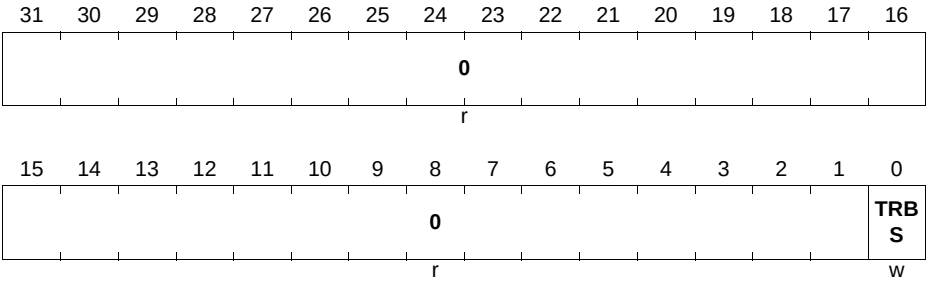


Field	Bits	Type	Description
TRB	0	rh	**Timer Run Bit** This field indicates if the timer is running. 0_B Timer is stopped 1_B Timer is running
CDIR	1	rh	**Timer Counting Direction** This filed indicates if the timer is being increment or decremented 0_B Timer is counting up 1_B Timer is counting down
0	[31:2]	r	**Reserved** Read always returns 0

CC4yTCSET

Through this register it is possible to start the timer.

CC4yTCSET (y = 0 - 3)
Slice Timer Run Set **($010C_H + 0100_H * y$)** **Reset Value: 00000000_H**

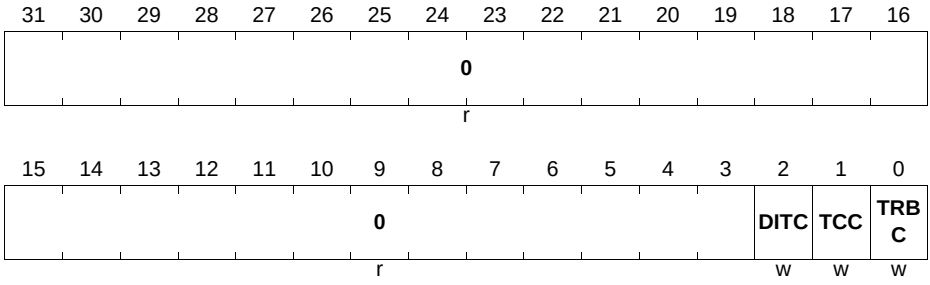


Field	Bits	Type	Description
TRBS	0	w	**Timer Run Bit set** Writing a 1_B into this field sets the run bit of the timer. Read always returns 0.
0	[31:1]	r	**Reserved** Read always returns 0

CC4yTCCLR

Through this register it is possible to stop and clear the timer, and clearing also the dither counter

CC4yTCCLR (y = 0 - 3)
Slice Timer Clear **(0110_H + 0100_H * y)** **Reset Value: 00000000_H**



Field	Bits	Type	Description
TRBC	0	w	**Timer Run Bit Clear** Writing a 1_B into this field clears the run bit of the timer. The timer is not cleared. Read always returns 0.
TCC	1	w	**Timer Clear** Writing a 1_B into this field clears the timer to 0000_H. Read always returns 0.
DITC	2	w	**Dither Counter Clear** Writing a 1_B into this field clears the dither counter to 0_H. Read always returns 0.
0	[31:3]	r	**Reserved** Read always returns 0

CC4yTC

This register holds the several possible configurations for the timer operation.

CC4yTC (y = 0 - 3)
Slice Timer Control **($0114_H + 0100_H * y$)** **Reset Value: 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0						MCME	EMT	EMS	TRPSW	TRPSE	0			TRAPE	FPE
r						rw	rw	rw	rw	rw	r			rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIM	DITHE		CCS	SCE	STRM	ENDM		0	CAPC		ECM	CMOD	CLST	TSSM	TCM
rw	rw		rw	rw	rw	rw		r	rw		rw	rh	rw	rw	rw

Field	Bits	Type	Description
TCM	0	rw	**Timer Counting Mode** This field controls the actual counting scheme of the timer. 0_B Edge aligned mode 1_B Center aligned mode *Note: When using an external signal to control the counting direction, the counting scheme is always edge aligned.*
TSSM	1	rw	**Timer Single Shot Mode** This field controls the single shot mode. This is applicable in edge and center aligned modes. 0_B Single shot mode is disabled 1_B Single shot mode is enabled
CLST	2	rw	**Shadow Transfer on Clear** Setting this bit to 1_B enables a shadow transfer when a timer clearing action is performed. Notice that the shadow transfer enable bitfields on the **GCST** register still need to be set to 1_B via software.

Field	Bits	Type	Description
CMOD	3	rh	**Capture Compare Mode** This field indicates in which mode the slice is operating. The default value is compare mode. The capture mode is automatically set by the HW when an external signal is mapped to a capture trigger. 0_B Compare Mode 1_B Capture Mode
ECM	4	rw	**Extended Capture Mode** This field control the Capture mode of the specific slice. It only has effect if the CMOD bit is 1_B. 0_B Normal Capture Mode. Clear of the Full Flag of each capture register is done by accessing the registers individually only. 1_B Extended Capture Mode. Clear of the Full Flag of each capture register is done not only by accessing the individual registers but also by accessing the ECRD register. When reading the ECRD register, only the capture register register full flag pointed by the ECRD.VPTR is cleared.
CAPC	[6:5]	rw	**Clear on Capture Control** 00_B Timer is never cleared on a capture event 01_B Timer is cleared on a capture event into capture registers 2 and 3. (When SCE = 1_B, Timer is always cleared in a capture event) 10_B Timer is cleared on a capture event into capture registers 0 and 1. (When SCE = 1_B, Timer is always cleared in a capture event) 11_B Timer is always cleared in a capture event.

Field	Bits	Type	Description
ENDM	[9:8]	rw	**Extended Stop Function Control** This field controls the extended functions of the external Stop signal. 00_B Clears the timer run bit only (default stop) 01_B Clears the timer only (flush) 10_B Clears the timer and run bit (flush/stop) 11_B Reserved *Note: When using an external up/down signal the flush operation sets the timer with zero if the counter is counting up and with the Period value if the counter is being decremented.*
STRM	10	rw	**Extended Start Function Control** This field controls the extended functions of the external Start signal. 0_B Sets run bit only (default start) 1_B Clears the timer and sets run bit (flush/start) *Note: When using an external up/down signal the flush operation sets the timer with zero if the counter is being incremented and with the Period value if the counter is being decremented.*
SCE	11	rw	**Equal Capture Event enable** 0_B Capture into **CC4yC0V**/**CC4yC1V** registers control by CCycapt0 and capture into **CC4yC3V**/**CC4yC2V** control by CCycapt1 1_B Capture into **CC4yC0V**/**CC4yC1V** and **CC4yC3V**/**CC4yC2V** control by CCycapt1
CCS	12	rw	**Continuous Capture Enable** 0_B The capture into a specific capture register is done with the rules linked with the full flags, described at **Section 16.2.7.6**. 1_B The capture into the capture registers is always done regardless of the full flag status (even if the register has not been read back).

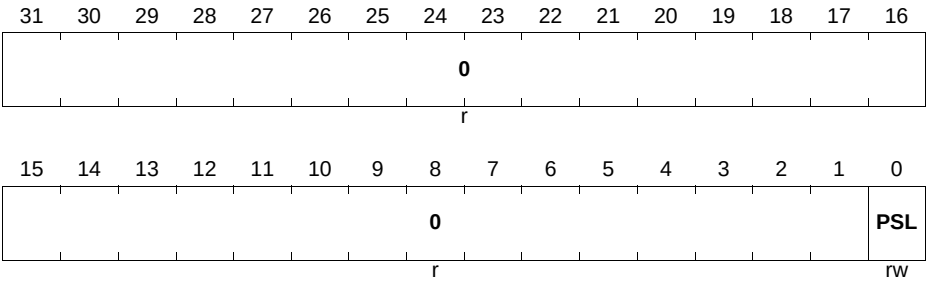
Field	Bits	Type	Description
DITHE	[14:13]	rw	**Dither Enable** This field controls the dither mode for the slice. See **Section 16.2.10**. 00_B Dither is disabled 01_B Dither is applied to the Period 10_B Dither is applied to the Compare 11_B Dither is applied to the Period and Compare
DIM	15	rw	**Dither input selector** This fields selects if the dither control signal is connected to the dither logic of the specific slice of is connected to the dither logic of slice 0. Notice that even if this field is set to 1_B, the field DITHE still needs to be programmed. 0_B Slice is using its own dither unit 1_B Slice is connected to the dither unit of slice 0.
FPE	16	rw	**Floating Prescaler enable** Setting this bit to 1_B enables the floating prescaler mode. 0_B Floating prescaler mode is disabled 1_B Floating prescaler mode is enabled
TRAPE	17	rw	**TRAP enable** Setting this bit to 1_B enables the TRAP action at the output pin. After mapping an external signal to the TRAP functionality, the user must set this field to 1_B to activate the effect of the TRAP on the output pin. Writing a 0_B into this field disables the effect of the TRAP function regardless of the state of the input signal. 0_B TRAP functionality has no effect on the output 1_B TRAP functionality affects the output
TRPSE	21	rw	**TRAP Synchronization Enable** Writing a 1_B into this bit enables a synchronous exiting with the PWM signal of the trap state. 0_B Exiting from TRAP state isn't synchronized with the PWM signal 1_B Exiting from TRAP state is synchronized with the PWM signal

Field	Bits	Type	Description
TRPSW	22	rw	**TRAP State Clear Control** 0_B The slice exits the TRAP state automatically when the TRAP condition is not present 1_B The TRAP state can only be exited by a SW request.
EMS	23	rw	**External Modulation Synchronization** Setting this bit to 1_B enables the synchronization of the external modulation functionality with the PWM period. 0_B External Modulation functionality is not synchronized with the PWM signal 1_B External Modulation functionality is synchronized with the PWM signal
EMT	24	rw	**External Modulation Type** This field selects if the external modulation event is clearing the CC4yST bit or if is gating the outputs. 0_B External Modulation functionality is clearing the CC4yST bit. 1_B External Modulation functionality is gating the outputs.
MCME	25	rw	**Multi Channel Mode Enable** 0_B Multi Channel Mode is disabled 1_B Multi Channel Mode is enabled
0	7, [20:18], [31:26]	r	**Reserved** Read always returns 0

CC4yPSL

This register holds the configuration for the output passive level control.

CC4yPSL (y = 0 - 3)
Passive Level Config **($0118_H + 0100_H * y$)** **Reset Value: 00000000_H**

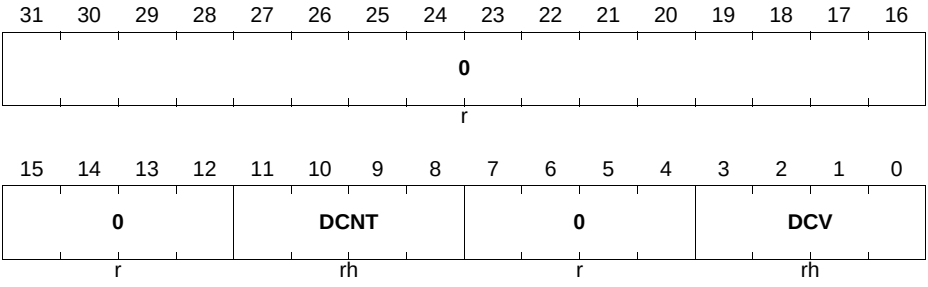


Field	Bits	Type	Description
PSL	0	rw	**Output Passive Level** This field controls the passive level of the output pin. 0_B Passive Level is LOW 1_B Passive Level is HIGH A write always addresses the shadow register, while a read always returns the current used value.
0	[31:1]	r	**Reserved** A read access always returns 0

CC4yDIT

This register holds the current dither compare and dither counter values.

CC4yDIT (y = 0 - 3)
Dither Config **($011C_H + 0100_H * y$)** **Reset Value: 00000000_H**

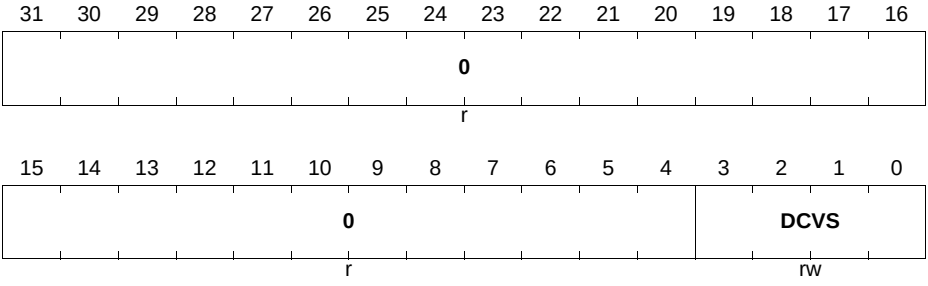


Field	Bits	Type	Description
DCV	[3:0]	rh	**Dither compare Value** This field contains the value used for the dither comparison. This value is updated when a shadow transfer occurs with the **CC4yDITS**.DCVS.
DCNT	[11:8]	rh	**Dither counter actual value**
0	[7:4], [31:12]	r	**Reserved** Read always returns 0.

CC4yDITS

This register contains the value that is going to be loaded into the **CC4yDIT**.DCV when the next shadow transfer occurs.

CC4yDITS (y = 0 - 3)
Dither Shadow Register **($0120_H + 0100_H * y$)** **Reset Value: 00000000_H**

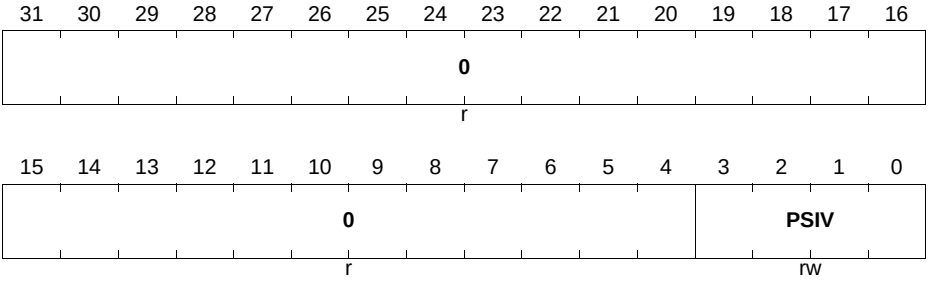


Field	Bits	Type	Description
DCVS	[3:0]	rw	**Dither Shadow Compare Value** This field contains the value that is going to be set on the dither compare value, **CC4yDIT**.DCV, within the next shadow transfer.
0	[31:4]	r	**Reserved** Read always returns 0.

CC4yPSC

This register contains the value that is loaded into the prescaler during restart.

CC4yPSC (y = 0 - 3)
Prescaler Control **($0124_H + 0100_H * y$)** **Reset Value: 00000000_H**

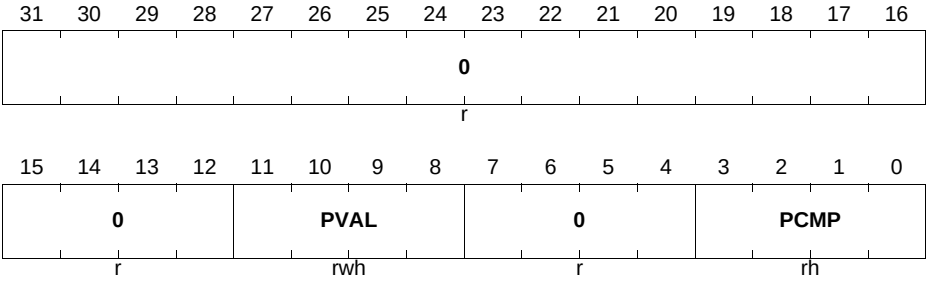


Field	Bits	Type	Description
PSIV	[3:0]	rw	**Prescaler Initial Value** This field contains the value that is applied to the Prescaler at startup. When floating prescaler mode is used, this value is applied when a timer compare match AND prescaler compare match occurs or when a capture event is triggered.
0	[31:4]	r	**Reserved** Read always returns 0.

CC4yFPC

This register contains the value used for the floating prescaler compare and the actual prescaler division value.

CC4yFPC (y = 0 - 3)
Floating Prescaler Control **(0128_H + 0100_H * y)** **Reset Value: 00000000_H**

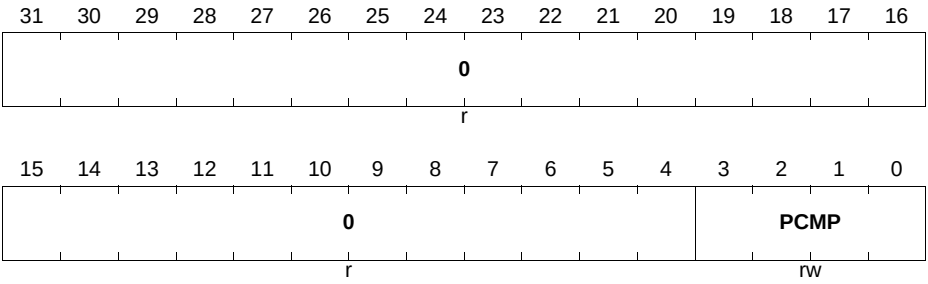


Field	Bits	Type	Description
PCMP	[3:0]	rh	**Floating Prescaler Compare Value** This field contains comparison value used in floating prescaler mode. The comparison is triggered by the Timer Compare match event. See **Section 16.2.11.2**.
PVAL	[11:8]	rwh	**Actual Prescaler Value** See **Table 16-7**. Writing into this register is only possible when the prescaler is stopped. When the floating prescaler mode is not used, this value is equal to the **CC4yPSC**.PSIV.
0	[7:4], [15:12], [31:16]	r	**Reserved** Read always returns 0.

CC4yFPCS

This register contains the value that is going to be transferred to the **CC4yFPC**.PCMP field within the next shadow transfer update.

CC4yFPCS (y = 0 - 3)
Floating Prescaler Shadow **($012C_H + 0100_H * y$)** **Reset Value: 00000000_H**

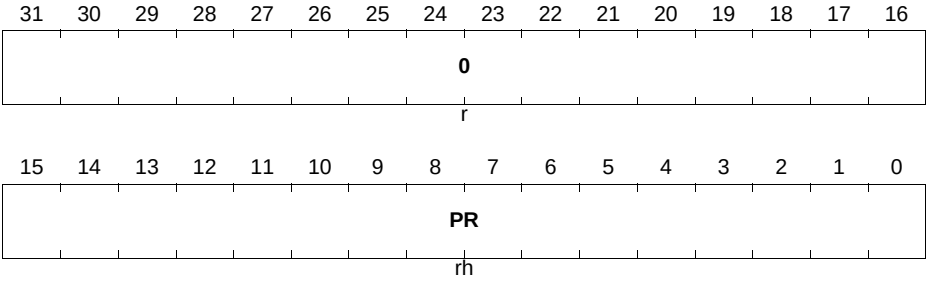


Field	Bits	Type	Description
PCMP	[3:0]	rw	**Floating Prescaler Shadow Compare Value** This field contains the value that is going to be set on the **CC4yFPC**.PCMP within the next shadow transfer. See **Table 16-7**.
0	[31:4]	r	**Reserved** Read always returns 0.

CC4yPR

This register contains the actual value for the timer period.

CC4yPR (y = 0 - 3)
Timer Period Value **($0130_H + 0100_H * y$)** **Reset Value: 00000000_H**

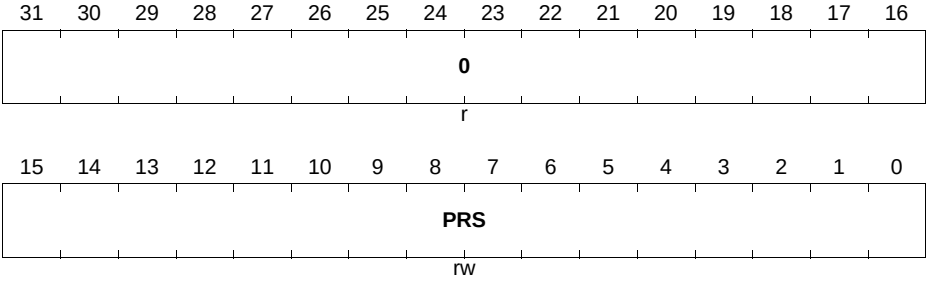


Field	Bits	Type	Description
PR	[15:0]	rh	**Period Register** Contains the value of the timer period. *Note: In Capture Mode when a external signal is selected for capturing the timer value into the capture registers 2 and 3, PR is not accessible for writing. A read always returns 0.*
0	[31:16]	r	**Reserved** A read always returns 0.

CC4yPRS

This register contains the value for the timer period that is going to be transferred into the **CC4yPR**.PR field when the next shadow transfer occurs.

CC4yPRS (y = 0 - 3)
Timer Shadow Period Value **($0134_H + 0100_H * y$)** **Reset Value: 00000000_H**



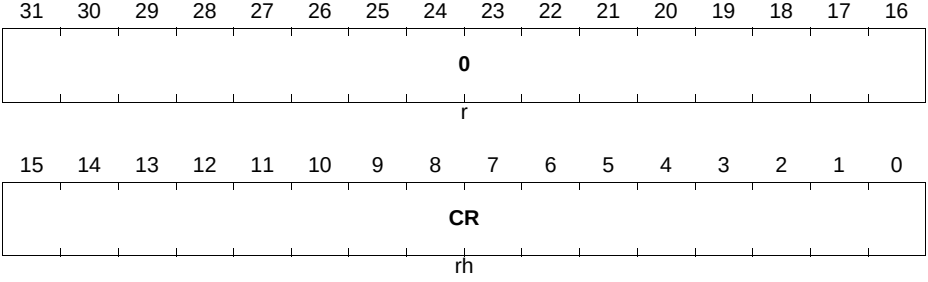
Field	Bits	Type	Description
PRS	[15:0]	rw	**Period Register** Contains the value of the timer period, that is going to be passed into the **CC4yPR**.PR field when the next shadow transfer occurs. *Note: In Capture Mode when a external signal is selected for capturing the timer value into the capture registers 2 and 3, the PRS is not accessible for writing. A read always returns 0.*

Field	Bits	Type	Description
0	[31:16]	r	**Reserved** A read always returns 0.

CC4yCR

This register contains the value for the timer comparison.

CC4yCR (y = 0 - 3)
Timer Compare Value **($0138_H + 0100_H * y$)** **Reset Value: 00000000_H**

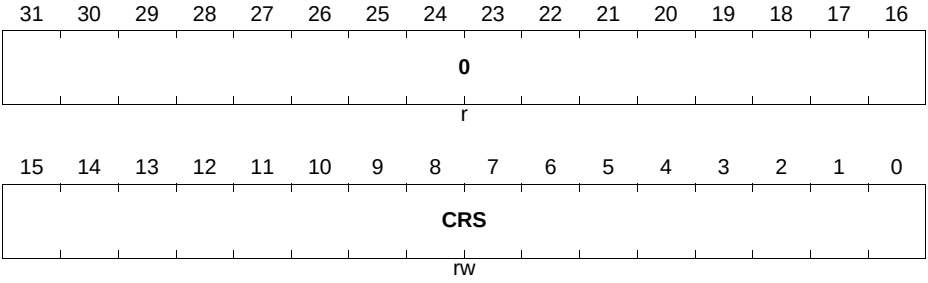


Field	Bits	Type	Description
CR	[15:0]	rh	**Compare Register** Contains the value for the timer comparison. *Note: In Capture Mode when a external signal is selected for capturing the timer value into the capture registers 0 and 1, a read always returns 0.*
0	[31:16]	r	**Reserved** A read always returns 0.

CC4yCRS

This register contains the value that is going to be loaded into the **CC4yCR**.CR field when the next shadow transfer occurs.

CC4yCRS (y = 0 - 3)
Timer Shadow Compare Value ($013C_H$ + 0100_H * y) **Reset Value: 00000000_H**

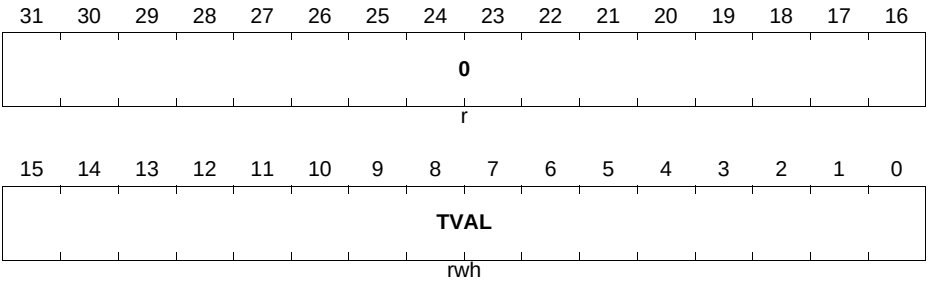


Field	Bits	Type	Description
CRS	[15:0]	rw	**Compare Register** Contains the value for the timer comparison, that is going to be passed into the **CC4yCR**.CR field when the next shadow transfer occurs. *Note: In Capture Mode when a external signal is selected for capturing the timer value into the capture registers 0 and 1, a read always returns 0.*
0	[31:16]	r	**Reserved** A read always returns 0.

CC4yTIMER

This register contains the current value of the timer.

CC4yTIMER (y = 0 - 3)
Timer Value **($0170_H + 0100_H * y$)** **Reset Value: 00000000_H**

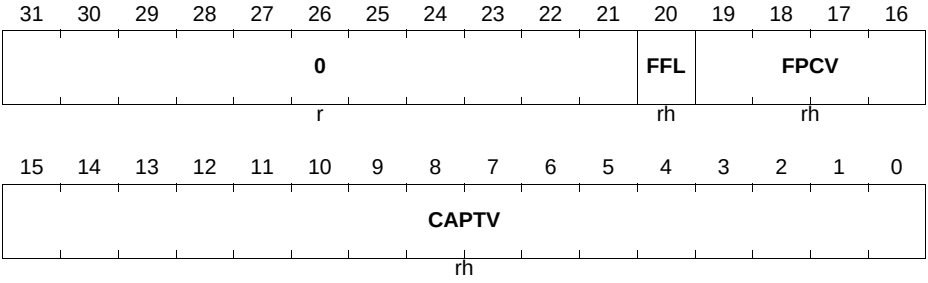


Field	Bits	Type	Description
TVAL	[15:0]	rwh	**Timer Value** This field contains the actual value of the timer. A write access is only possible when the timer is stopped.
0	[31:16]	r	**Reserved** A read access always returns 0

CC4yC0V

This register contains the values associated with the Capture 0 field.

CC4yC0V (y = 0 - 3)
Capture Register 0 **($0174_H + 0100_H * y$)** **Reset Value: 00000000_H**

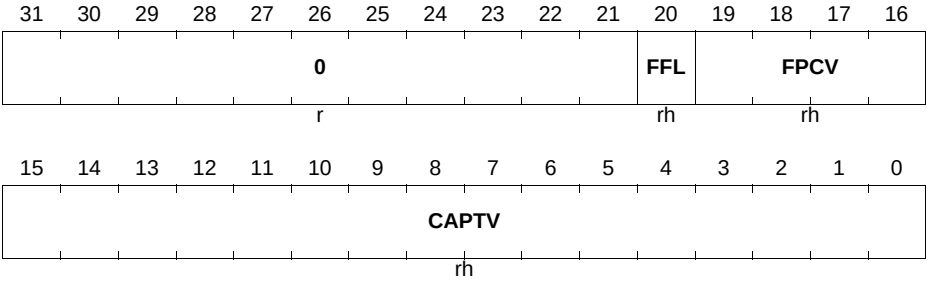


Field	Bits	Type	Description
CAPTV	[15:0]	rh	**Capture Value** This field contains the capture register 0 value. See **Figure 16-26**. In compare mode a read access always returns 0.
FPCV	[19:16]	rh	**Prescaler Value** This field contains the prescaler value at the time of the capture event into the capture register 0. In compare mode a read access always returns 0.
FFL	20	rh	**Full Flag** This bit indicates if a new value was capture into the capture register 0 after the last read access. See **Figure 16-26**. In compare mode a read access always returns 0. 0_B No new value was captured into the specific capture register 1_B A new value was captured into the specific register
0	[31:21]	r	**Reserved** A read always returns 0

CC4yC1V

This register contains the values associated with the Capture 1 field.

CC4yC1V (y = 0 - 3)
Capture Register 1 **($0178_H + 0100_H * y$)** **Reset Value: 00000000_H**

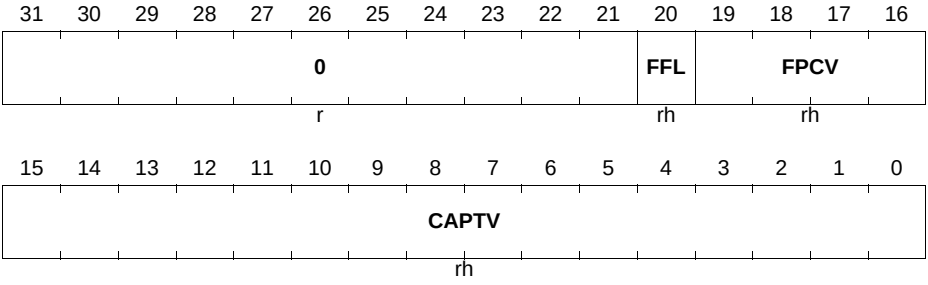


Field	Bits	Type	Description
CAPTV	[15:0]	rh	**Capture Value** This field contains the capture register 1 value. See Figure 16-26. In compare mode a read access always returns 0.
FPCV	[19:16]	rh	**Prescaler Value** This field contains the prescaler value at the time of the capture event into the capture register 1. In compare mode a read access always returns 0.
FFL	20	rh	**Full Flag** This bit indicates if a new value was capture into the capture register 1 after the last read access. See Figure 16-26. In compare mode a read access always returns 0. 0_B No new value was captured into the specific capture register 1_B A new value was captured into the specific register
0	[31:21]	r	**Reserved** A read always returns 0

CC4yC2V

This register contains the values associated with the Capture 2 field.

CC4yC2V (y = 0 - 3)
Capture Register 2 **($017C_H + 0100_H * y$)** **Reset Value: 00000000_H**

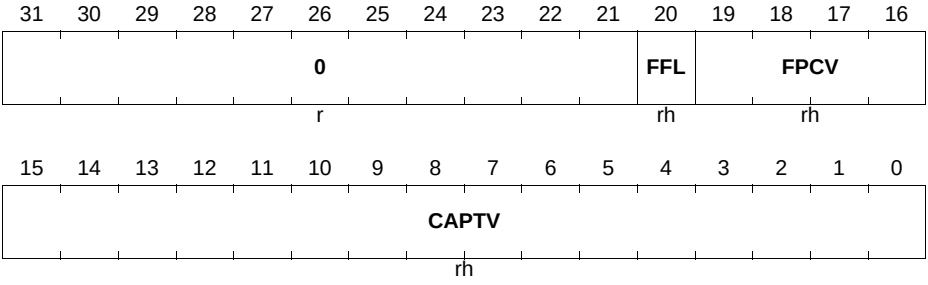


Field	Bits	Type	Description
CAPTV	[15:0]	rh	**Capture Value** This field contains the capture register 2 value. See **Figure 16-26**. In compare mode a read access always returns 0.
FPCV	[19:16]	rh	**Prescaler Value** This field contains the prescaler value at the time of the capture event into the capture register 2. In compare mode a read access always returns 0.
FFL	20	rh	**Full Flag** This bit indicates if a new value was capture into the capture register 2 after the last read access. See **Figure 16-26**. In compare mode a read access always returns 0. 0_B No new value was captured into the specific capture register 1_B A new value was captured into the specific register
0	[31:21]	r	**Reserved** A read always returns 0

CC4yC3V

This register contains the values associated with the Capture 3 field.

CC4yC3V (y = 0 - 3)
Capture Register 3 **(0180_H + 0100_H * y)** **Reset Value: 00000000_H**



Field	Bits	Type	Description
CAPTV	[15:0]	rh	**Capture Value** This field contains the capture register 3 value. See **Figure 16-26**. In compare mode a read access always returns 0.
FPCV	[19:16]	rh	**Prescaler Value** This field contains the prescaler value at the time of the capture event into the capture register 3. In compare mode a read access always returns 0.
FFL	20	rh	**Full Flag** This bit indicates if a new value was capture into the capture register 3 after the last read access. See **Figure 16-26**. In compare mode a read access always returns 0. 0_B No new value was captured into the specific capture register 1_B A new value was captured into the specific register
0	[31:21]	r	**Reserved** A read always returns 0

CC4yINTS

This register contains the status of all interrupt sources.

CC4yINTS (y = 0 - 3)
Interrupt Status **($01A0_H + 0100_H * y$)** **Reset Value: 00000000_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				TRPF	E2AS	E1AS	E0AS	0				CMDS	CMUS	OMDS	PMUS
r				rh	rh	rh	rh	r				rh	rh	rh	rh

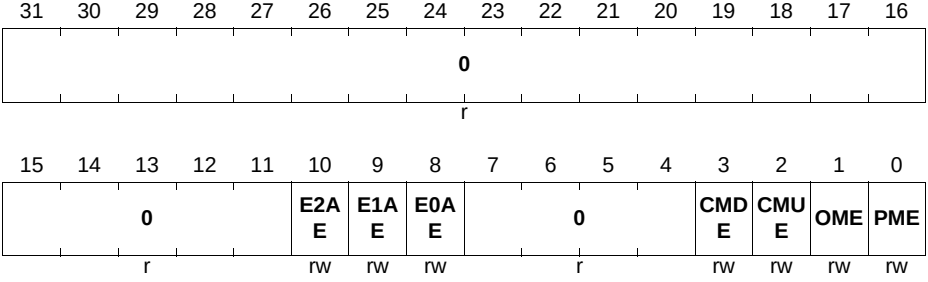
Field	Bits	Type	Description
PMUS	0	rh	**Period Match while Counting Up** 0_B Period match while counting up not detected 1_B Period match while counting up detected
OMDS	1	rh	**One Match while Counting Down** 0_B One match while counting down not detected 1_B One match while counting down detected
CMUS	2	rh	**Compare Match while Counting Up** 0_B Compare match while counting up not detected 1_B Compare match while counting up detected
CMDS	3	rh	**Compare Match while Counting Down** 0_B Compare match while counting down not detected 1_B Compare match while counting down detected
E0AS	8	rh	**Event 0 Detection Status** Depending on the user selection on the **CC4yINS**.EV0EM, this bit can be set when a rising, falling or both transitions are detected. 0_B Event 0 not detected 1_B Event 0 detected
E1AS	9	rh	**Event 1 Detection Status** Depending on the user selection on the **CC4yINS**.EV1EM, this bit can be set when a rising, falling or both transitions are detected. 0_B Event 1 not detected 1_B Event 1 detected
E2AS	10	rh	**Event 2 Detection Status** Depending on the user selection on the **CC4yINS**.EV1EM, this bit can be set when a rising, falling or both transitions are detected. 0_B Event 2 not detected 1_B Event 2 detected *Note: If this event is linked with the TRAP function, this field is automatically cleared when the slice exits the Trap State.*
TRPF	11	rh	**Trap Flag Status** This field contains the status of the Trap Flag.

Field	Bits	Type	Description
0	[7:4], [31:12]	r	**Reserved** A read always returns 0.

CC4yINTE

Through this register it is possible to enable or disable the specific interrupt source(s).

CC4yINTE (y = 0 - 3)
Interrupt Enable Control **($01A4_H + 0100_H * y$)** **Reset Value: 00000000_H**



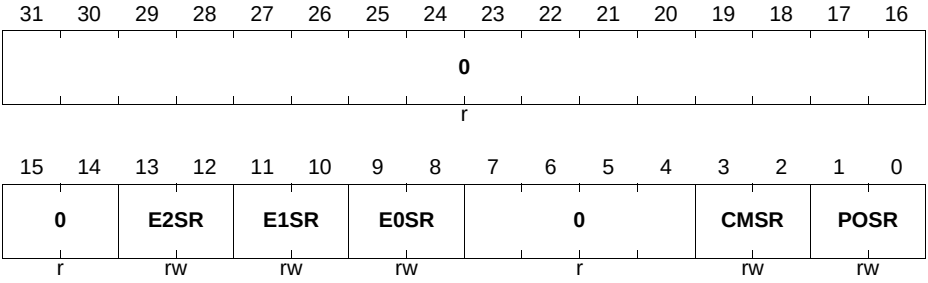
Field	Bits	Type	Description
PME	0	rw	**Period match while counting up enable** Setting this bit to 1_B enables the generation of an interrupt pulse every time a period match while counting up occurs. 0_B Period Match interrupt is disabled 1_B Period Match interrupt is enabled
OME	1	rw	**One match while counting down enable** Setting this bit to 1_B enables the generation of an interrupt pulse every time an one match while counting down occurs. 0_B One Match interrupt is disabled 1_B One Match interrupt is enabled

Field	Bits	Type	Description
CMUE	2	rw	**Compare match while counting up enable** Setting this bit to 1_B enables the generation of an interrupt pulse every time a compare match while counting up occurs. 0_B Compare Match while counting up interrupt is disabled 1_B Compare Match while counting up interrupt is enabled
CMDE	3	rw	**Compare match while counting down enable** Setting this bit to 1_B enables the generation of an interrupt pulse every time a compare match while counting down occurs. 0_B Compare Match while counting down interrupt is disabled 1_B Compare Match while counting down interrupt is enabled
E0AE	8	rw	**Event 0 interrupt enable** Setting this bit to 1_B enables the generation of an interrupt pulse every time that Event 0 is detected. 0_B Event 0 detection interrupt is disabled 1_B Event 0 detection interrupt is enabled
E1AE	9	rw	**Event 1 interrupt enable** Setting this bit to 1_B enables the generation of an interrupt pulse every time that Event 1 is detected. 0_B Event 1 detection interrupt is disabled 1_B Event 1 detection interrupt is enabled
E2AE	10	rw	**Event 2 interrupt enable** Setting this bit to 1_B enables the generation of an interrupt pulse every time that Event 2 is detected. 0_B Event 2 detection interrupt is disabled 1_B Event 2 detection interrupt is enabled
0	[7:4], [31:11]	r	**Reserved** A read always returns 0

CC4ySRS

Through this register it is possible to select to which service request line each interrupt source is forwarded.

CC4ySRS (y = 0 - 3)
Service Request Selector **($01A8_H$ + 0100_H * y)** **Reset Value: 00000000_H**



Field	Bits	Type	Description
POSR	[1:0]	rw	**Period/One match Service request selector** This field selects to which slice Service request line, the interrupt(s) generated by the Period match while counting up and One match while counting down are going to be forward. 00_B Forward to CC4ySR0 01_B Forward to CC4ySR1 10_B Forward to CC4ySR2 11_B Forward to CC4ySR3
CMSR	[3:2]	rw	**Compare match Service request selector** This field selects to which slice Service request line, the interrupt(s) generated by the Compare match while counting up and Compare match while counting down are going to be forward. 00_B Forward to CC4ySR0 01_B Forward to CC4ySR1 10_B Forward to CC4ySR2 11_B Forward to CC4ySR3
E0SR	[9:8]	rw	**Event 0 Service request selector** This field selects to which slice Service request line, the interrupt generated by the Event 0 detection is going to be forward. 00_B Forward to CC4ySR0 01_B Forward to CC4ySR1 10_B Forward to CC4ySR2 11_B Forward to CC4ySR3

Field	Bits	Type	Description
E1SR	[11:10]	rw	**Event 1 Service request selector** This field selects to which slice Service request line, the interrupt generated by the Event 1detection is going to be forward. 00_B Forward to CC4ySR0 01_B Forward to CC4ySR1 10_B Forward to CC4ySR2 11_B Forward to CC4ySR3
E2SR	[13:12]	rw	**Event 2 Service request selector** This field selects to which slice Service request line, the interrupt generated by the Event 2 detection is going to be forward. 00_B Forward to CC4ySR0 01_B Forward to CC4ySR1 10_B Forward to CC4ySR2 11_B Forward to CC4ySR3
0	[7:4], [31:14]	r	**Reserved** Read always returns 0.

CC4ySWS

Through this register it is possible for the SW to set a specific interrupt status flag.

CC4ySWS (y = 0 - 3)
Interrupt Status Set **($01AC_H + 0100_H * y$)** **Reset Value: 00000000_H**

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

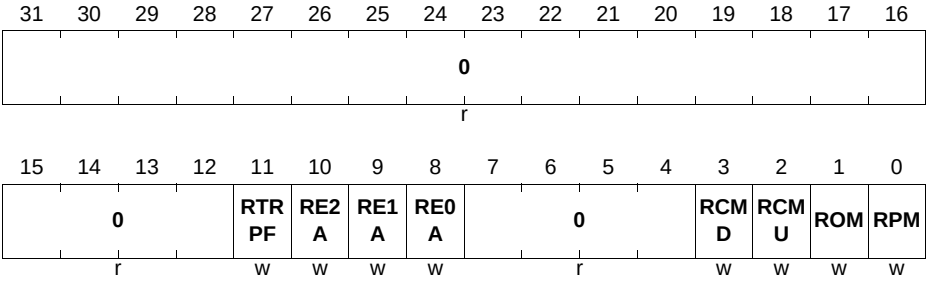
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				STRPF	SE2A	SE1A	SE0A	0				SCMD	SCMU	SOM	SPM
r				w	w	w	w	r				w	w	w	w

Field	Bits	Type	Description
SPM	0	w	**Period match while counting up set** Writing a 1_B into this field sets the **CC4yINTS**.PMUS bit. An interrupt pulse is generated if the source is enabled. A read always returns 0.
SOM	1	w	**One match while counting down set** Writing a 1_B into this bit sets the **CC4yINTS**.OMDS bit. An interrupt pulse is generated if the source is enabled. A read always returns 0.
SCMU	2	w	**Compare match while counting up set** Writing a 1_B into this field sets the **CC4yINTS**.CMUS bit. An interrupt pulse is generated if the source is enabled. A read always returns 0.
SCMD	3	w	**Compare match while counting down set** Writing a 1_B into this bit sets the **CC4yINTS**.CMDS bit. An interrupt pulse is generated if the source is enabled. A read always returns 0.
SE0A	8	w	**Event 0 detection set** Writing a 1_B into this bit sets the **CC4yINTS**.E0AS bit. An interrupt pulse is generated if the source is enabled. A read always returns 0.
SE1A	9	w	**Event 1 detection set** Writing a 1_B into this bit sets the **CC4yINTS**.E1AS bit. An interrupt pulse is generated if the source is enabled. A read always returns 0.
SE2A	10	w	**Event 2 detection set** Writing a 1_B into this bit sets the **CC4yINTS**.E2AS bit. An interrupt pulse is generated if the source is enabled. A read always returns 0.
STRPF	11	w	**Trap Flag status set** Writing a 1_B into this bit sets the **CC4yINTS**.TRPF bit. A read always returns 0.
0	[7:4], [31:12]	r	**Reserved** Read always returns 0

CC4ySWR

Through this register it is possible for the SW to clear a specific interrupt status flag.

CC4ySWR (y = 0 - 3)
Interrupt Status Clear **($01B0_H + 0100_H * y$)** **Reset Value: 00000000_H**



Field	Bits	Type	Description
RPM	0	w	**Period match while counting up clear** Writing a 1_B into this field clears the **CC4yINTS**.PMUS bit. A read always returns 0.
ROM	1	w	**One match while counting down clear** Writing a 1_B into this bit clears the **CC4yINTS**.OMDS bit. A read always returns 0.
RCMU	2	w	**Compare match while counting up clear** Writing a 1_B into this field clears the **CC4yINTS**.CMUS bit. A read always returns 0.
RCMD	3	w	**Compare match while counting down clear** Writing a 1_B into this bit clears the **CC4yINTS**.CMDS bit. A read always returns 0.
RE0A	8	w	**Event 0 detection clear** Writing a 1_B into this bit clears the **CC4yINTS**.E0AS bit. A read always returns 0.
RE1A	9	w	**Event 1 detection clear** Writing a 1_B into this bit clears the **CC4yINTS**.E1AS bit. A read always returns 0.
RE2A	10	w	**Event 2 detection clear** Writing a 1_B into this bit clears the **CC4yINTS**.E2AS bit. A read always returns 0.

Field	Bits	Type	Description
RTRPF	11	w	**Trap Flag status clear** Writing a 1_B into this bit clears the **CC4yINTS**.TRPF bit. Not valid if **CC4yTC**.TRPEN = 1_B and the Trap State is still active. A read always returns 0.
0	[7:4], [31:12]	r	**Reserved** Read always returns 0

CC4yECRD0

Through this register it is possible to read back the FIFO structure of the capture function that is linked with the capture trigger 0. The read back is only valid if the **CC4yTC**.ECM = 1_B.

CC4yECRD0 (y = 0 - 3)
Extended Read Back 0 **($01B8_H + 0100_H * y$)** **Reset Value: 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0						LCV	FFL	VPTR		SPTR		FPCV			
r						rh	rh	rh		rh		rh			

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

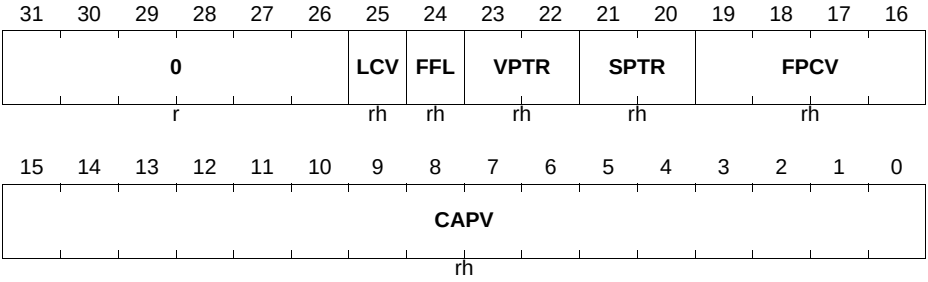
Field	Bits	Type	Description
CAPV	[15:0]	rh	**Timer Capture Value** This field contains the timer captured value
FPCV	[19:16]	rh	**Prescaler Capture value** This field contains the value of the prescaler clock division associated with the specific CAPV field

Field	Bits	Type	Description
SPTR	[21:20]	rh	**Slice pointer** This field indicates the slice index in which the value was captured. 00_B CC40 01_B CC41 10_B CC42 11_B CC43
VPTR	[23:22]	rh	**Capture register pointer** This field indicates the capture register index in which the value was captured. 00_B Capture register 0 01_B Capture register 1 10_B Capture register 2 11_B Capture register 3
FFL	24	rh	**Full Flag** This bit indicates if the associated capture register contains a new value. 0_B No new value was captured into this register 1_B A new value has been captured into this register
LCV	25	rh	**Lost Capture Value** This field indicates if between two reads of the ECRD0 a capture trigger occured while the FIFO structure was full. If a capture trigger occured between two reads than a capture value was lost. This field is automatically cleared by the HW whenever a read to the ECRD occurs. 0_B No capture was lost 1_B A capture was lost
0	[31:26]	r	**Reserved** Read always returns 0

CC4yECRD1

Through this register it is possible to read back the FIFO structure of the capture function that is linked with the capture trigger 1. The read back is only valid if the **CC4yTC**.ECM = 1_B.

CC4yECRD1 (y = 0 - 3)
Extended Read Back 1 **($01BC_H + 0100_H * y$)** **Reset Value: 00000000_H**



Field	Bits	Type	Description
CAPV	[15:0]	rh	**Timer Capture Value** This field contains the timer captured value
FPCV	[19:16]	rh	**Prescaler Capture value** This field contains the value of the prescaler clock division associated with the specific CAPV field
SPTR	[21:20]	rh	**Slice pointer** This field indicates the slice index in which the value was captured. 00_B CC40 01_B CC41 10_B CC42 11_B CC43
VPTR	[23:22]	rh	**Capture register pointer** This field indicates the capture register index in which the value was captured. 00_B Capture register 0 01_B Capture register 1 10_B Capture register 2 11_B Capture register 3
FFL	24	rh	**Full Flag** This bit indicates if the associated capture register contains a new value. 0_B No new value was captured into this register 1_B A new value has been captured into this register

Field	Bits	Type	Description
LCV	25	rh	**Lost Capture Value** This field indicates if between two reads of the ECRD0 a capture trigger occured while the FIFO structure was full. If a capture trigger occured between two reads than a capture value was lost. This field is automatically cleared by the HW whenever a read to the ECRD occurs. 0_B No capture was lost 1_B A capture was lost
0	[31:26]	r	**Reserved** Read always returns 0

16.8 Interconnects

The tables that refer to the “global pins” are the ones that contain the inputs/outputs of each module that are common to all slices.

The GPIO connections are available at the Ports chapter.

16.8.1 CCU40 pins

Table 16-13 CCU40 Pin Connections

Global Inputs/Outputs	I/O	Connected To	Description
CCU40.MCLK	I	PCLK	Kernel clock
CCU40.CLKA	I	not connected	another count source for the prescaler
CCU40.CLKB	I	ERU0.IOUT0	another count source for the prescaler
CCU40.CLKC	I	ERU0.IOUT1	another count source for the prescaler
CCU40.MCSS	I	reserved	Multi pattern sync with shadow transfer trigger
CCU40.SR0	O	NVIC; ERU0.OGU01;	Service request line
CCU40.SR1	O	NVIC; ERU0.OGU11;	Service request line

Table 16-13 CCU40 Pin Connections (cont'd)

Global Inputs/Outputs	I/O	Connected To	Description
CCU40.SR2	O	NVIC; VADC0.BGREQTRA; VADC0.G0REQTRA; VADC0.G1REQTRA; ERU0.OGU21;	Service request line
CCU40.SR3	O	NVIC; VADC0.BGREQTRB; VADC0.G0REQTRB; VADC0.G1REQTRB; ERU0.OGU31;	Service request line

Table 16-14 CCU40 - CC40 Pin Connections

Input/Output	I/O	Connected To	Description
CCU40.IN0A	I	P0.12	General purpose function
CCU40.IN0B	I	P0.6	General purpose function
CCU40.IN0C	I	P0.0	General purpose function
CCU40.IN0D	I	ERU0.PDOUT1	General purpose function
CCU40.IN0E	I	reserved	General purpose function
CCU40.IN0F	I	reserved	General purpose function
CCU40.IN0G	I	reserved	General purpose function
CCU40.IN0H	I	reserved	General purpose function
CCU40.IN0I	I	SCU.GSC40	General purpose function
CCU40.IN0J	I	ERU0.PDOUT0	General purpose function
CCU40.IN0K	I	ERU0.IOUT0	General purpose function
CCU40.IN0L	I	USIC0_CH0.DX2INS	General purpose function
CCU40.IN0M	I	CCU40.GP10	General purpose function
CCU40.IN0N	I	CCU40.ST1	General purpose function
CCU40.IN0O	I	CCU40.ST2	General purpose function
CCU40.IN0P	I	CCU40.ST3	General purpose function
CCU40.MCI0	I	reserved	Multi Channel pattern input

Table 16-14 CCU40 - CC40 Pin Connections (cont'd)

Input/Output	I/O	Connected To	Description
CCU40.OUT0	O	P0.0; P0.5; P0.6; P1.0; P2.0; P2.0.HW1 direction control	Slice compare output
CCU40.GP00	O	CCU40.IN3M	Selected signal for event 0
CCU40.GP01	O	not connected	Selected signal for event 1
CCU40.GP02	O	reserved	Selected signal for event 2
CCU40.ST0	O	CCU40.IN1N; CCU40.IN2N; CCU40.IN3N; VADC0.BGREQGTD; VADC0.G0REQGTD; VADC0.G1REQGTD;	Slice status bit
CCU40.PS0	O	not connected	Multi channel pattern sync trigger: PM when counting UP (edge aligned) or OM when counting DOWN (center aligned)

Table 16-15 CCU40 - CC41 Pin Connections

Input/Output	I/O	Connected To	Description
CCU40.IN1A	I	P0.12	General purpose function
CCU40.IN1B	I	P0.7	General purpose function
CCU40.IN1C	I	P0.1	General purpose function
CCU40.IN1D	I	ERU0.PDOUT0	General purpose function
CCU40.IN1E	I	reserved	General purpose function
CCU40.IN1F	I	reserved	General purpose function
CCU40.IN1G	I	reserved	General purpose function
CCU40.IN1H	I	reserved	General purpose function
CCU40.IN1I	I	SCU.GSC40	General purpose function
CCU40.IN1J	I	ERU0.PDOUT1	General purpose function
CCU40.IN1K	I	ERU0.IOUT1	General purpose function

Table 16-15 CCU40 - CC41 Pin Connections (cont'd)

Input/Output	I/O	Connected To	Description
CCU40.IN1L	I	USIC0_CH1.DX2INS	General purpose function
CCU40.IN1M	I	CCU40.GP20	General purpose function
CCU40.IN1N	I	CCU40.ST0	General purpose function
CCU40.IN1O	I	CCU40.ST2	General purpose function
CCU40.IN1P	I	CCU40.ST3	General purpose function
CCU40.MCI1	I	reserved	Multi Channel pattern input
CCU40.OUT1	O	P0.1: P0.4; P0.7; P1.1; P2.1; P2.1.HW1 direction control	Slice compare output
CCU40.GP10	O	CCU40.IN0M	Selected signal for event 0
CCU40.GP11	O	not connected	Selected signal for event 1
CCU40.GP12	O	reserved	Selected signal for event 2
CCU40.ST1	O	CCU40.IN0N; CCU40.IN2O; CCU40.IN3O; VADC0.BGREQGTC; VADC0.G0REQGTC; VADC0.G1REQGTC;	Slice status bit
CCU40.PS1	O	reserved	Multi channel pattern sync trigger: PM when counting UP (edge aligned) or OM when counting DOWN (center aligned)

Table 16-16 CCU40 - CC42 Pin Connections

Input/Output	I/O	Connected To	Description
CCU40.IN2A	I	P0.12	General purpose function
CCU40.IN2B	I	P0.8	General purpose function
CCU40.IN2C	I	P0.2	General purpose function
CCU40.IN2D	I	ERU0.PDOUT3	General purpose function
CCU40.IN2E	I	reserved	General purpose function

Table 16-16 CCU40 - CC42 Pin Connections (cont'd)

Input/Output	I/O	Connected To	Description
CCU40.IN2F	I	reserved	General purpose function
CCU40.IN2G	I	reserved	General purpose function
CCU40.IN2H	I	reserved	General purpose function
CCU40.IN2I	I	SCU.GSC40	General purpose function
CCU40.IN2J	I	ERU0.PDOUT2	General purpose function
CCU40.IN2K	I	ERU0.IOUT2	General purpose function
CCU40.IN2L	I	reserved	General purpose function
CCU40.IN2M	I	CCU40.GP30	General purpose function
CCU40.IN2N	I	CCU40.ST0	General purpose function
CCU40.IN2O	I	CCU40.ST1	General purpose function
CCU40.IN2P	I	CCU40.ST3	General purpose function
CCU40.MCI2	I	reserved	Multi Channel pattern input
CCU40.OUT2	O	P0.2; P0.8; P1.2; P2.10; P2.8.HW1 pull control P2.9.HW1 pull control P2.10.HW1 direction control	Slice compare output
CCU40.GP20	O	CCU40.IN1M	Selected signal for event 0
CCU40.GP21	O	not connected	Selected signal for event 1
CCU40.GP22	O	reserved	Selected signal for event 2
CCU40.ST2	O	CCU40.IN0O; CCU40.IN1O; CCU40.IN3P; VADC0.BGREQGTB; VADC0.G0REQGTB; VADC0.G1REQGTB;	Slice status bit
CCU40.PS0	O	not connected	Multi channel pattern sync trigger: PM when counting UP (edge aligned) or OM when counting DOWN (center aligned)

Table 16-17 CCU40 - CC43 Pin Connections

Input/Output	I/O	Connected To	Description
CCU40.IN3A	I	P0.12	General purpose function
CCU40.IN3B	I	P0.9	General purpose function
CCU40.IN3C	I	P0.3	General purpose function
CCU40.IN3D	I	ERU0.PDOUT2	General purpose function
CCU40.IN3E	I	reserved	General purpose function
CCU40.IN3F	I	reserved	General purpose function
CCU40.IN3G	I	reserved	General purpose function
CCU40.IN3H	I	reserved	General purpose function
CCU40.IN3I	I	SCU.GSC40	General purpose function
CCU40.IN3J	I	ERU0.PDOUT3	General purpose function
CCU40.IN3K	I	ERU0.IOUT3	General purpose function
CCU40.IN3L	I	reserved	General purpose function
CCU40.IN3M	I	CCU40.GP00	General purpose function
CCU40.IN3N	I	CCU40.ST0	General purpose function
CCU40.IN3O	I	CCU40.ST1	General purpose function
CCU40.IN3P	I	CCU40.ST2	General purpose function
CCU40.MCI3	I	reserved	Multi Channel pattern input
CCU40.OUT3	O	P0.3; P0.9; P1.3; P2.11; P2.2.HW1 pull control P2.6.HW1 pull control P2.7.HW1 pull control P2.11.HW1 direction control	Slice compare output
CCU40.GP30	O	CCU40.IN2M	Selected signal for event 0
CCU40.GP31	O	not connected	Selected signal for event 1
CCU40.GP32	O	reserved	Selected signal for event 2

Table 16-17 CCU40 - CC43 Pin Connections (cont'd)

Input/Output	I/O	Connected To	Description
CCU40.ST3	O	CCU40.IN0P; CCU40.IN1P; CCU40.IN2P; VADC0.BGREQGTA; VADC0.G0REQGTA; VADC0.G1REQGTA;	Slice status bit
CCU40.PS3	O	not connected	Multi channel pattern sync trigger: PM when counting UP (edge aligned) or OM when counting DOWN (center aligned)

17 General Purpose I/O Ports (Ports)

The XMC1100 has 34 digital General Purpose Input/Output (GPIO) port lines which are connected to the on-chip peripheral units.

17.1 Overview

The Ports provide a generic and very flexible software and hardware interface for all standard digital I/Os. Each Port slice has individual interfaces for the operation as General Purpose I/O and it further provides the connectivity to the on-chip periphery and the control for the pad characteristics. **Table 17-1** gives an overview of the available PORTS and other pins in the different packages of the XMC1100:

Table 17-1 Port/Pin Overview

Port	TSSOP-38	TSSOP-16	Note
P0	16	8	
P1	6	-	High current bi-directional pad
P2	12	6	Analog/Digital input and bi-directional pad
Supply VDD, ADC Reference Voltage	1	1	VDD
Supply GND, ADC Reference Ground	1	1	VSS
I/O Port Supply	1	-	VDDP
I/O Port Ground	1	-	VSSP

17.1.1 Features

This is a list of the main features of the Ports:

- same generic register interface for each port pin, **Chapter 17.8**
- simple and robust software access for general purpose I/O functionality, **Chapter 17.2**
- direct input connections to on-chip peripherals, **Chapter 17.2.1**
- dedicated hardware interface for CCU and USIC with select option, **Chapter 17.3**
- defined power-up/power-fail behavior, **Chapter 17.6**.
- up to seven alternate output paths from peripherals selectable, **Chapter 17.8.1**
- programmable open-drain or push-pull output driver stage, **Chapter 17.8.1**
- programmable weak pull-up and pull-down devices, **Chapter 17.8.1**
- programmable input inverter, **Chapter 17.8.1**
- programmable pad hysteresis, **Chapter 17.8.2**

- disabling of digital input stage on shared analog inputs, **Chapter 17.8.3**
- separate set and clear output control to avoid read-modify-write operations, **Chapter 17.8.5**
- programmable power-save behavior in Deep-Sleep mode, **Chapter 17.8.7**
- Privileged Mode restricted access to configuration registers to avoid accidential modification

17.1.2 Block Diagram

Below is a figure with the generic structure of a digital port pin, split into the port slice with the control logic and the pad with the pull devices and the input and output stages, **Figure 17-1**.

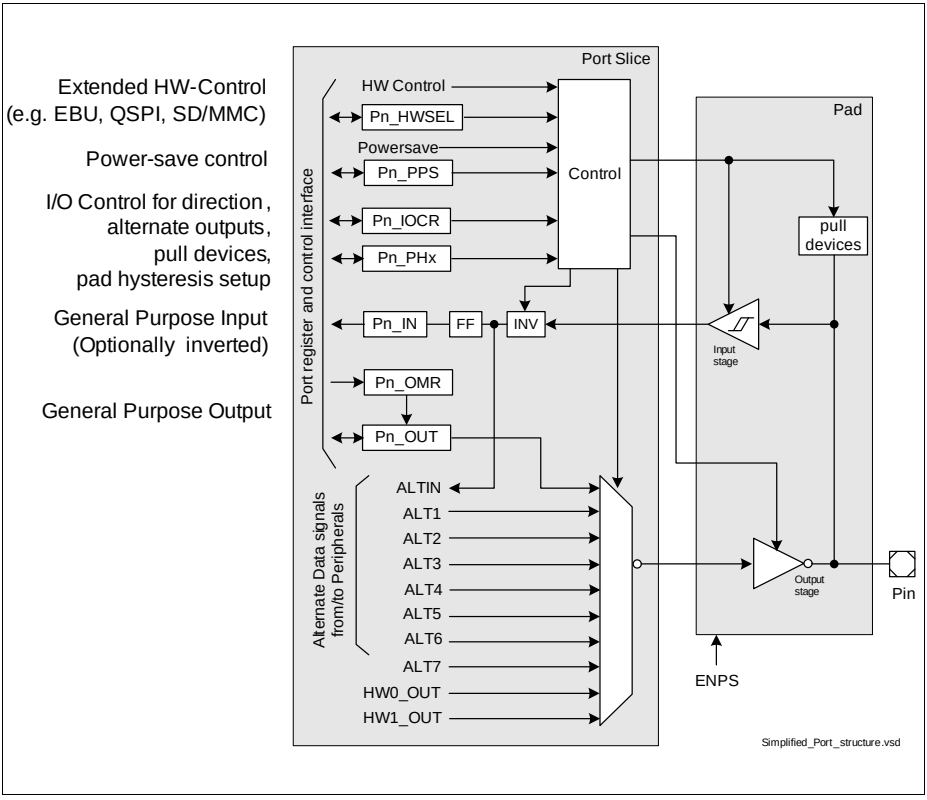


Figure 17-1 General Structure of a digital Port Pin

17.1.3 Definition of Terms

Some specific terms are used throughout this chaper:

- **Pin/Ball:** External connection of the device to the PCB.
- **Dedicated Pin:** A Pin with a dedicated function that is not under the control of the port logic (i.e. supply pins).
- **Port Pin:** A pin under the control of the port logic (P0.1).
- **Port:** A group of up to 16 Port Pins sharing the same generic register set (P0).
- **Port Slice:** The “sum” of register bits and control logic used to control a port pin.
- **Pad:** Analog component containing the output driver, pull devices and input Schmitt-Trigger. Also interfaces the internal logic operating on V_{DDC} to the pad supply domain V_{DDP}.
- **GPIO:** General Purpose Input/Output. A port pin with the input and/or output function controlled by the application software.
- **Alternate Function:** Direct connection of a port pin with an on-chip peripheral.

17.2 GPIO and Alternate Functions

The Ports can be operated as General Purpose Input/Output (GPIO) and with Alternate Functions of the on-chip periphery, configured by the Port Input/Output Control Register (Pn_IOCR, **Chapter 17.8.1**). It selects between

- Direct or Inverted Input
  - with or without pull device
- Push-pull or Open-Drain Output driven by
  - Pn_OUT (GPIO)
  - selected peripheral output connections.

As GPIO the port pin is controlled by the application software, reading the input value by the Port Input register Pn_IN (**Chapter 17.8.6**) and/or defining the output value by the Output Modification Register Pn_OMR (**Chapter 17.8.5**). Output modification by Pn_OMR is preferred over the direct change of the output value with the Output register Pn_OUT (**Chapter 17.8.4**), as Pn_OMR allows the manipulation of individual port pins in a single access without “disturbing” other pins controlled by the same Pn_OUT register. If an application uses a GPIO as a bi-directional I/O line, register Pn_IOCR has to be written to switch between input and output functionality.

For the operation with Alternate Functions, the port pins are directly connected to input or output functions of the on-chip periphery. This allows the peripheral to directly evaluate the input value or drive the output value of the port pin without further application software interaction after the initial configuration. The connection of alternate functions is used for control and communication interfaces, like a PWM from a CAPCOM unit or a SPI communication of a USIC channel. A detailed connectivity list of the peripherals to the port pins is given in the **Port I/O Function Description** chapter. For specific functions, certain peripherals may also take direct control of “their” port pins, see **Hardware Controlled I/Os**.

17.2.1 Input Operation

As an input, the actual voltage level at the port pin is translated into a logical 0_B or 1_B via a Schmitt-Trigger device within the pad. The resulting input value can be optionally inverted. As general purpose input, the signal is synchronized and can be read with the Input register (Pn_IN, **Chapter 17.8.6**). Alternatively, the input can be connected to the multiple on-chip peripherals via the ALTIN signal. Where neccessary, these peripherals have internal controls to select the appropriate port pin with an input multiplexer stage, and will take care of synchronization and further processing of the input signals. (See respective peripheral chapters for more details on the input selection and handling). With the Pn_IOCR register (**Chapter 17.8.1**), it is also possible to activate an internal weak pull-up or pull-down device in the pad.
The input register Pn_IN and ALTIN signal always represent the state of the input, independent whether the port pin is configured as input or output. This means that even if the port is in output mode, the level of the pin can be read by software via Pn_IN and/or a peripheral can use the pin level as an input.

Pad Hysteresis Control

The pad hysteresis can be configured according to the application needs via the Pad Hysteresis Control register (Pn_PHCR, **Chapter 17.8.2**). Selecting the appropriate pad hysteresis allows optimized pad oscillation behaviour for touch-sensing applications.

17.2.2 Output Operation

In output mode, the output driver is activated and drives the value supplied through the multiplexer to the port pin. Switching between input and output mode is accomplished through the Pn_IOCR register (**Chapter 17.8.1**), which

- enables or disables the output driver,
- selects between open-drain and push-pull mode,
- selects the general purpose or alternate function outputs.

The output multiplexer selects the signal source of the output with

- Pn_IOCR
  - general purpose output (Pn_OUT, **Chapter 17.8.4**)
  - alternate peripheral functions, ALT1...ALT7
- hardware control, Pn_HWSEL
  - HW0_OUT
  - HW1_OUT

Note: It is recommended to complete the Port and peripheral configuration with respect to operating mode and initial values before the port pin is switched to output mode.

The output function is exclusive, meaning that only one peripheral has control of the output path at any one time.

Used as general purpose output, software can directly modify the content of Pn_OUT to define the output value on the pin. A write operation to Pn_OUT updates all port pins of that port (e.g. P0) that are configured as general purpose output. Updating just one or a selected few general purpose output pins via Pn_OUT requires a masked read-modify-write operation to avoid disturbing pins that shall not be changed. Direct writes to Pn_OUT will also affect Pn_OUT bits configured for use with the Pin Power-save function, **Chapter 17.4**.

Because of that, it is preferred to modify Pn_OUT bits by the Output Modification Register Pn_OMR (**Chapter 17.8.5**). The bits in Pn_OMR allow to individually set, clear or toggle the bits in the Pn_OUT register and only update the “addressed” Pn_OUT bits.

The data written by software into the output register Pn_OUT can also be used as input data to an on-chip peripheral. This enables, for example, peripheral tests and simulation via software without external circuitry.

Output lines of on-chip peripherals can directly control the output value of the output driver if selected via ALT1 to ALT7 as well as HW0_OUT and HW1_OUT. After initialization, this allows the connected peripherals to directly drive complex control and communication patterns without further software interaction with the ports.

The actual logic level at the pin can be examined through reading Pn_IN and compared against the applied output level (either applied by the output register Pn_OUT, or via an alternate output function of a peripheral unit). This can be used to detect some electrical failures at the pin caused by external circuitry. In addition, software-supported arbitration schemes between different “masters” can be implemented in this way, using the open-drain configuration and an external wired-AND circuitry. Collisions on the external communication lines can be detected when a high level (1_B) is output, but a low level (0_B) is seen when reading the pin value via the input register Pn_IN or directly by a peripheral (via ALTIN, for example a USIC channel in IIC mode).

There are two pad types in XMC1100 providing different drive strength:

- Standard pad
- High Current pad

The assignment of each port pin to one of these pad types is listed in the Package Pin Summary table. Further details about pad properties are summarized in the Data Sheet.

17.3 Hardware Controlled I/Os

Some ports pins are overlaid with peripheral functions for which the connected peripheral needs direct hardware control, e.g. for the direction of a bi-directional data bus. There is a dedicated hardware control interface for these functions. As multiple peripherals need access to this interface, the Pn_HWSEL register (**Chapter 17.8.8**) allows to select between the hardware “masters”.

Depending on the operating mode, the peripheral can take control of various functions:

- Pin direction, input or output, e.g. for bi-directional signals

- Driver type, open-drain or push-pull
- Pull devices under peripheral control or under standard control via Pn_IOCR

Some configurations remain under control by the standard configuration interface, the pad hysteresis by Pn_PHCR and the direct or inverted input path by Pn_IOCR.

Pn_HWSEL.HWx just pre-assigns the hardware-control of the pin to a certain peripheral, but the peripheral itself decides when to take control over it. As long as the peripheral does not take control of a given pin via HWx_EN, the configuration of this pin is still defined by the configuration registers and it is available as GPIO or for other alternate functions. This might be because the selected peripheral has controls to just activate a subset of its pins, or because the peripheral is not active at all.

This mechanism can also be used to prohibit the hardware control of certain pins to a peripheral, in case the application does not need the respective functionality and the peripheral has no controls to disable the hardware control selectively.

The default hardware input configuration and the pull devices are controlled by Pn_IOCR.

Note: Do not enable the Pin Power Save function for pins configured for Hardware Control (Pn_HWSEL.HWx != 00_B). Doing so may result in an undefined behavior of the pin when the device enters the Deep Sleep state.

17.4 Power Saving Mode Operation

In Deep-Sleep mode, the behavior of a pin depends on the setting of the Pin Power save register (Pn_PPS, **Chapter 17.8.7**). Basically, each pin can be configured to react to the Power Save Mode Request or to ignore it. In case a pin is configured to react to a Power Save Mode Request, the output driver is switched to tri-state, the input Schmitt-Trigger and the pull devices are switched off (see **Figure 17-2**). The input signal to the the on-chip peripherals is optionally driven statically high or low, software-defined by a value stored in Pn_OUT or by the last input value sampled to the Pn_OUT register during normal operation. The actual reaction is configured with the Pn_IOCR register under power save conditions, see **Table 17-8**.

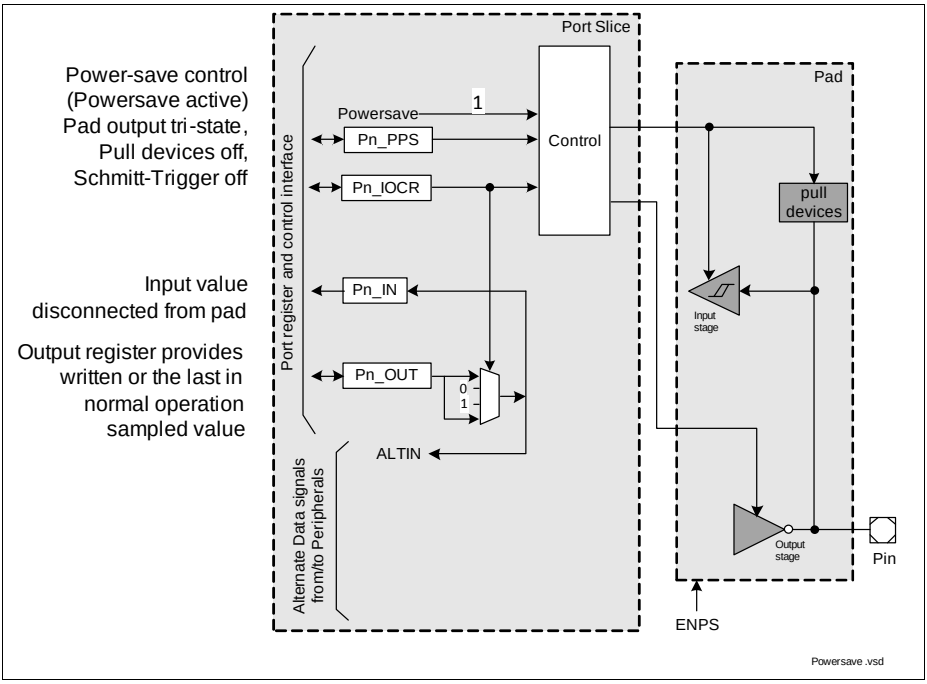


Figure 17-2 Port Pin in Power Save State

Note: Do not enable the Pin Power Save function for pins configured for Hardware Control (Pn_HWSEL.HWx != 00_B). Doing so may result in an undefined behavior of the pin when the device enters the Deep Sleep state.

17.5 Analog Ports

P2.2 - P2.9 is the analog and digital input port with a simplified port and pad structure, see **Figure 17-3**. The analog pads have no output drivers and the digital input Schmitt-Trigger can be controlled by the Pn_PDISC (**Chapter 17.8.3**) register. Accordingly, the port control interface is reduced in its functionality. The Pn_IOCR register controls the pull devices, the optional input inversion and the input source in power-save mode. The Pn_OUT has only its power-save functionality, as described in **Chapter 17.4**.

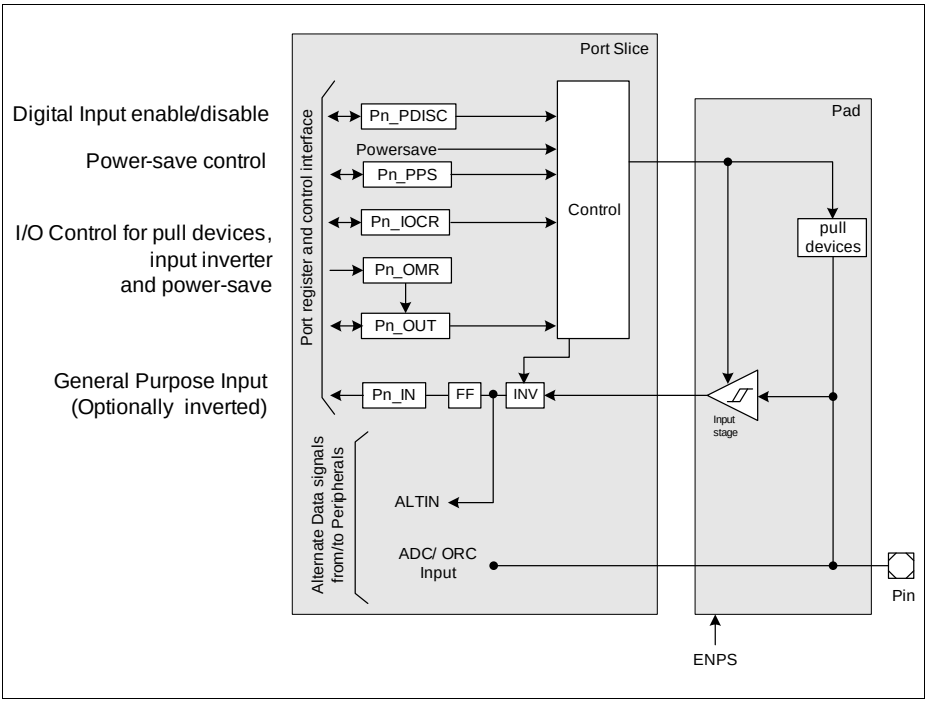


Figure 17-3 Analog Port Structure

17.6 Power, Reset and Clock

All digital I/O pads are held in a defined state, tristate, with output driver disabled and no pull devices active when one of the following occurs:

- During power-up, until V_{DDC} and V_{DDP} voltage levels are stable and within limits
- During power-fail, one or more voltage levels are outside the limits

Refer to the EVR section in the SCU chapter and Data Sheet for details on the power-up, supply monitoring and voltage limits.

All Port registers are reset with the System Reset (see Reset Control Unit chapter in the System Control Unit). The standard reset values are defined such that the port pins are configured as tri-state inputs, output driver disabled and no pull devices active. Exceptions from these standard values are related to special interfaces or the analog input channels.

All registers of the Ports are clocked with f_{MCLK}.

17.7 Initialization and System Dependencies

It is recommended to follow pre-defined routines for the initialization of the port pins.

Input

When a peripheral shall use a port pin as input, the actual pin levels may immediately trigger an unexpected peripheral event (e.g. clock edge at SPI). This can be avoided by forcing the "passive" level via pull-up/down programming.

The following steps are required to configure a port pin as an input:

- Pn_IOCR
  input configuration with pull device and/or power-save mode configuration
- Pn_PHCR
  pad hysteresis configuration (if applicable)
- Hardware Control (if applicable)
  – Pn_HWSEL
    switch hardware control to peripheral
- Pin Power Save (if applicable)
  – Pn_OMR/Pn_OUT
    default value in power save mode (if applicable)
  – Pn_PPS
    enable power save control

Output

When a port pin is configured as output for an on-chip peripheral, it is important that the peripheral is configured before the port switches the control to the peripheral in order to avoid spikes on the output.
The following steps are required to configure a port pin as an output:

- Pn_OMR/Pn_OUT
  Initial output value (as general purpose output)
- GPIO or Alternate Output
  – Pn_IOCR
    Output multiplexer select
    Push-pull or open-drain output driver mode
    Activates the output driver!
- Hardware Control
  – Pn_IOCR
    depending on the hardware function Pn_IOCR can enable the internal pull devices
  – Pn_HWSEL
    Switch hardware control to peripheral

Transitions

If a port pin is used for different functions that require a reconfiguration of the port registers, it is recommended to do this transition via an intermediate “neutral” tri-state input configuration.

- Pn_HWSEL
  disable hardware selection; can be omitted if no hardware control is used on the port pin
- Pn_PPS
  disable power save mode control of the pin; can be ommited if no power save configuration is used on the port pin
- Pn_IOCR
  tri-state input and no pull device active

17.8 Registers

Registers Overview

The absolute register address is calculated by adding:

Module Base Address + Offset Address

Table 17-2 Registers Address Space

Module	Base Address	End Address	Note
P0	$4004\ 0000_H$	$4004\ 00FF_H$	
P1	$4004\ 0100_H$	$4004\ 01FF_H$	High current bi-directional pad
P2	$4004\ 0200_H$	$4004\ 02FF_H$	Analog/Digital input and bi-directional pad

Table 17-3 Register Overview

Short Name	Description	Offset Addr.	Access Mode		Description See
			Read	Write	
Pn_OUT	Port n Output Register	0000_H	U, PV	U, PV	Page 17-28
Pn_OMR	Port n Output Modification Register	0004_H	U, PV	U, PV	Page 17-31
–	Reserved	0008_H-$000C_H$	BE	BE	–
Pn_IOCR0	Port n Input/Output Control Register 0	0010_H	U, PV	PV	Page 17-14
Pn_IOCR4	Port n Input/Output Control Register 4	0014_H	U, PV	PV	Page 17-15
Pn_IOCR8	Port n Input/Output Control Register 8	0018_H	U, PV	PV	Page 17-16
Pn_IOCR12	Port n Input/Output Control Register 12	$001C_H$	U, PV	PV	Page 17-16
–	Reserved	0020_H	BE	BE	–
Pn_IN	Port n Input Register	0024_H	U, PV	R	Page 17-34
–	Reserved	0028_H-$003C_H$	BE	BE	–

Table 17-3 Register Overview (cont'd)

Short Name	Description	Offset Addr.	Access Mode		Description See
			Read	Write	
Pn_PHCR0	Port n Pad Hysteresis Control Register 0	0040_H	U, PV	PV	Page 17-20
Pn_PHCR1	Port n Pad Hysteresis Control Register 1	0044_H	U, PV	PV	Page 17-21
–	Reserved	0048_H-$005C_H$	BE	BE	–
P0_PDISC P1_PDISC	Port n Pin Function Decision Control Register (non-ADC ports)	0060_H	U, PV	BE	Page 17-23
P2_PDISC	Port n Pin Function Decision Control Register (ADC ports)	0060_H	U, PV	PV	Page 17-24
–	Reserved	0064_H-$006C_H$	BE	BE	–
Pn_PPS	Port n Pin Power Save Register	0070_H	U, PV	PV	Page 17-36
Pn_HWSEL	Port n Hardware Select Register	0074_H	U, PV	PV	Page 17-39
–	Reserved	0078_H-$00FC_H$	BE	BE	–

Table 17-4 Registers Access Rights and Reset Classes

Register Short Name	Access Rights		Reset Class
	Read	Write	
Pn_IN	U, PV	R	System Reset
Pn_OUT		U, PV	
Pn_OMR			
Pn_IOCR0		PV	
Pn_IOCR4			
Pn_IOCR8			
Pn_IOCR12			
Pn_PDISC (ADC ports)			
Pn_PH0			
Pn_PH1			
Pn_PPS			
Pn_PDISC (non-ADC ports)		BE	

17.8.1 Port Input/Output Control Registers

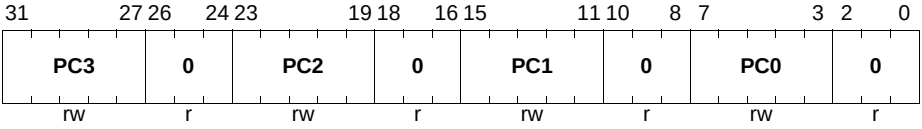
The port input/output control registers select the digital output and input driver functionality and characteristics of a GPIO port pin. Port direction (input or output), pull-up or pull-down devices for inputs, and push-pull or open-drain functionality for outputs can be selected by the corresponding bit fields PCx (x = 0-15). Each 32-bit wide port input/output control register controls four GPIO port lines:

Register Pn_IOCR0 controls the Pn.[3:0] port lines
Register Pn_IOCR4 controls the Pn.[7:4] port lines
Register Pn_IOCR8 controls the Pn.[11:8] port lines
Register Pn_IOCR12 controls the Pn.[15:12] port lines

The diagrams below show the register layouts of the port input/output control registers with the PCx bit fields. One PCx bit field controls exactly one port line Pn.x.

Pn_IOCR0 (n=0-1)
Port n Input/Output Control Register 0
($4004\ 0010_H + n*100_H$) **Reset Value: $0000\ 0000_H$**

P2_IOCR0
Port 2 Input/Output Control Register 0
(0010_H) **Reset Value: $0000\ 0000_H$**



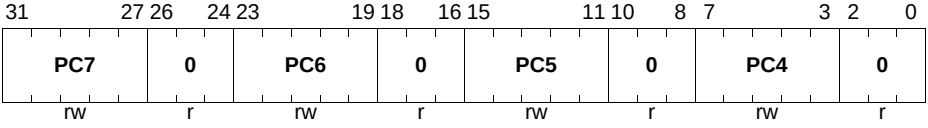
Field	Bits	Type	Description
PC0, PC1, PC2, PC3	[7:3], [15:11], [23:19], [31:27]	rw	**Port Control for Port n Pin 0 to 3** This bit field determines the Port n line x functionality (x = 0-3) according to the coding table (see **Table 17-5**).
0	[2:0], [10:8], [18:16], [26:24]	r	**Reserved** Read as 0; should be written with 0.

P0_IOCR4
Port 0 Input/Output Control Register 4

(0014_H) **Reset Value: $0000\ 0000_H$**

P2_IOCR4
Port 2 Input/Output Control Register 4

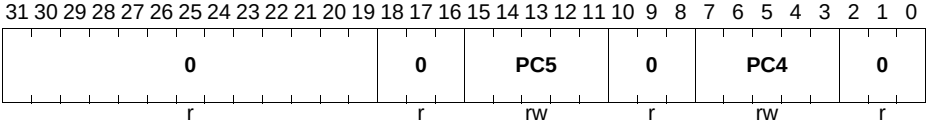
(0014_H) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
PC4, PC5, PC6, PC7	[7:3], [15:11], [23:19], [31:27]	rw	**Port Control for Port n Pin 4 to 7** This bit field determines the Port n line x functionality (x = 4-7) according to the coding table (see **Table 17-5**).
0	[2:0], [10:8], [18:16], [26:24]	r	**Reserved** Read as 0; should be written with 0.

P1_IOCR4
Port 1 Input/Output Control Register 4

(0014_H) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
PC4, PC5	[7:3], [15:11]	rw	**Port Control for Port n Pin 4 to 5** This bit field determines the Port n line x functionality (x = 4-5) according to the coding table (see **Table 17-5**).

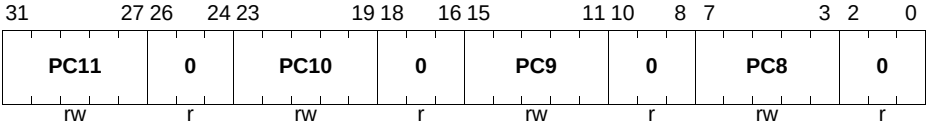
Field	Bits	Type	Description
0	[2:0], [10:8], [18:16], [31:19]	r	**Reserved** Read as 0; should be written with 0.

P0_IOCR8
Port 0 Input/Output Control Register 8

(0018_H) **Reset Value: 0000 0000$_H$ 1)**

P2_IOCR8
Port 2 Input/Output Control Register 8

(0018_H) **Reset Value: 0000 0000$_H$**

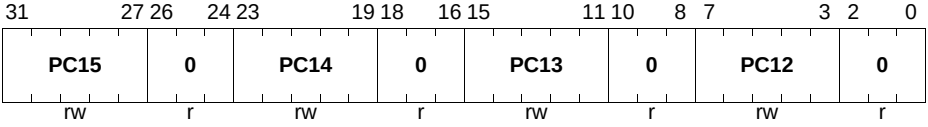


1) Upon reset, the value of PC8 (P0.8) is 00000_B. The Startup Software (SSW) will change the PC8 value to input pull-up device active, 00010_B. Refer to the Startup chapter for more information.

Field	Bits	Type	Description
PC8, PC9, PC10, PC11	[7:3], [15:11], [23:19], [31:27]	rw	**Port Control for Port n Pin 8 to 11** This bit field determines the Port n line x functionality (x = 8-11) according to the coding table (see **Table 17-5**).
0	[2:0], [10:8], [18:16], [26:24]	r	**Reserved** Read as 0; should be written with 0.

P0_IOCR12
Port 0 Input/Output Control Register 12

($001C_H$) **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
PC12, PC13, PC14, PC15	[7:3], [15:11], [23:19], [31:27]	rw	**Port Control for Port n Pin 12 to 15** This bit field determines the Port n line x functionality (x = 12-15) according to the coding table (see **Table 17-5**).
0	[2:0], [10:8], [18:16], [26:24]	r	**Reserved** Read as 0; should be written with 0.

Depending on the GPIO port functionality (number of GPIO lines of a port), not all of the port input/output control registers are implemented.

The structure with one control bit field for each port pin located in different register bytes offers the possibility to configure the port pin functionality of a single pin with byte-oriented accesses without accessing the other PCx bit fields.

Port Control Coding

Table 17-5 describes the coding of the PCx bit fields that determine the port line functionality.

The Pn_IOCRy PCx bit field is also used to contol the pin behavior in Deep-Sleep mode if the Pin Power Save option is enabled, see **Chapter 17.8.7**.

Table 17-5 Standard PCx Coding[1)]

PCx[4:0]	I/O	Output Characteristics	Selected Pull-up / Pull-down / Selected Output Function
$0X000_B$	Direct Input	–	No internal pull device active
$0X001_B$			Internal pull-down device active
$0X010_B$			Internal pull-up device active
$0X011_B$			No internal pull device active; Pn_OUTx continuously samples the input value
$0X100_B$	Inverted Input	–	No internal pull device active
$0X101_B$			Internal pull-down device active
$0X110_B$			Internal pull-up device active
$0X111_B$			No internal pull device active; Pn_OUTx continuously samples the input value

Table 17-5 Standard PCx Coding[1)] (cont'd)

PCx[4:0]	I/O	Output Characteristics	Selected Pull-up / Pull-down / Selected Output Function
10000_B	Output (Direct Input)	Push-pull	General-purpose output
10001_B			Alternate output function 1
10010_B			Alternate output function 2
10011_B			Alternate output function 3
10100_B			Alternate output function 4
10101_B			Alternate output function 5
10110_B			Alternate output function 6
10111_B			Alternate output function 7
11000_B		Open-drain	General-purpose output
11001_B			Alternate output function 1
11010_B			Alternate output function 2
11011_B			Alternate output function 3
11100_B			Alternate output function 4
11101_B			Alternate output function 5
11110_B			Alternate output function 6
11111_B			Alternate output function 7

1) For the analog and digital input port P2.2 - P2.9, the combinations with PCx[4]=1_B is reserved.

17.8.2 Pad Hysteresis Control Register

The pad structure of the XMC1100 GPIO lines offers the possibility to select the pad hysteresis. These two parameters are controlled by the bit fields in the pad hysteresis control registers Pn_PHCR0/1, independently from input/output and pull-up/pull-down control functionality as programmed in the Pn_IOCR register. Pn_PHCR0 and Pn_PHCR1 registers are assigned to each port.

The 1-bit pad hysteresis selection bit field PHx in the pad hysteresis control registers Pn_PHCR make it possible to select the port line functionality as shown in **Table 17-6**. Note that the pad hysteresis control registers are specific for each port.

Table 17-6 Pad Hysteresis Selection

PHx	Functionality
0	Standard hysteresis
1	Large hysteresis

Note: Refer to Input/Output Characteristics table in the XMC1100 Data Sheet for hysteresis parameter values.

Pad Hysteresis Control Registers

This is the general description of the PHCR registers. Each port contains its own specific PHCR registers, described additionally at each port, that can contain between one and eight PHx fields for PHCR0 and PHCR1 registers, respectively. Each field controls 1 pin. For coding of PHx, see **Page 17-18**.

P0_PHCR0
Port 0 Pad Hysteresis Control Register 0
(0040_H) **Reset Value: 0000 0000_H**

P2_PHCR0
Port 2 Pad Hysteresis Control Register 0
(0040_H) **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**PH7**	**0**			**PH6**	**0**			**PH5**	**0**			**PH4**	**0**	
r	rw	r			rw	r			rw	r			rw	r	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**PH3**	**0**			**PH2**	**0**			**PH1**	**0**			**PH0**	**0**	
r	rw	r			rw	r			rw	r			rw	r	

Field	Bits	Type	Description
PH0	2	rw	**Pad Hysteresis for Pn.0**
PH1	6	rw	**Pad Hysteresis for Pn.1**
PH2	10	rw	**Pad Hysteresis for Pn.2**
PH3	14	rw	**Pad Hysteresis for Pn.3**
PH4	18	rw	**Pad Hysteresis for Pn.4**
PH5	22	rw	**Pad Hysteresis for Pn.5**
PH6	26	rw	**Pad Hysteresis for Pn.6**
PH7	30	rw	**Pad Hysteresis for Pn.7**

Field	Bits	Type	Description
0	[1:0], [5:3], [9:7], [13:11], [17:15], [21:19], [25:23], [29:27], 31	r	**Reserved** Read as 0; should be written with 0.

P1_PHCR0
Port 1 Pad Hysteresis Control Register 0

(0040_H) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0						**0**			**PH5**	**0**			**PH4**	**0**	
r						r			rw	r			rw	r	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**PH3**	**0**			**PH2**	**0**			**PH1**	**0**			**PH0**	**0**	
r	rw	r			rw	r			rw	r			rw	r	

Field	Bits	Type	Description
PH0	2	rw	**Pad Hysteresis for P1.0**
PH1	6	rw	**Pad Hysteresis for P1.1**
PH2	10	rw	**Pad Hysteresis for P1.2**
PH3	14	rw	**Pad Hysteresis for P1.3**
PH4	18	rw	**Pad Hysteresis for P1.4**
PH5	22	rw	**Pad Hysteresis for P1.5**

Field	Bits	Type	Description
0	[1:0], [5:3], [9:7], [13:11], [17:15], [21:19], [25:23], [31:26]	r	**Reserved** Read as 0; should be written with 0.

P0_PHCR1
Port 0 Pad Hysteresis Control Register 1

(0044_H) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**PH15**	**0**			**PH14**	**0**			**PH13**	**0**			**PH12**	**0**	
r	rw	r			rw	r			rw	r			rw	r	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**PH11**	**0**			**PH10**	**0**			**PH9**	**0**			**PH8**	**0**	
r	rw	r			rw	r			rw	r			rw	r	

Field	Bits	Type	Description
PH8	2	rw	**Pad Hysteresis for P0.8**
PH9	6	rw	**Pad Hysteresis for P0.9**
PH10	10	rw	**Pad Hysteresis for P0.10**
PH11	14	rw	**Pad Hysteresis for P0.11**
PH12	18	rw	**Pad Hysteresis for P0.12**
PH13	22	rw	**Pad Hysteresis for P0.13**
PH14	26	rw	**Pad Hysteresis for P0.14**
PH15	30	rw	**Pad Hysteresis for P0.15**

Field	Bits	Type	Description
0	[1:0], [5:3], [9:7], [13:11], [17:15], [21:19], [25:23], [29:27], 31	r	**Reserved** Read as 0; should be written with 0.

P2_PHCR1
Port 2 Pad Hysteresis Control Register 1

(0044$_H$) **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	**PH11**	**0**			**PH10**	**0**			**PH9**	**0**			**PH8**	**0**	
r	rw	r			rw	r			rw	r			rw	r	

Field	Bits	Type	Description
PH8	2	rw	**Pad Hysteresis for P2.8**
PH9	6	rw	**Pad Hysteresis for P2.9**
PH10	10	rw	**Pad Hysteresis for P2.10**
PH11	14	rw	**Pad Hysteresis for P2.11**
0	[1:0], [5:3], [9:7], [13:11], [31:15]	r	**Reserved** Read as 0; should be written with 0.

17.8.3 Pin Function Decision Control Register

Pin Function Decision Control Register

The primary use for this register is to disable/enable the digital pad structure in shared analog and digital ports, see the dedicated description for **P2_PDISC**.

For “normal” digital I/O ports (P0-P1) this register is read-only and the read value corresponds to the available pins in the given package.

P0_PDISC
Port 0 Pin Function Decision Control Register

(0060_H) **Reset Value: 0000 XXXX$_H$[1)]**

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

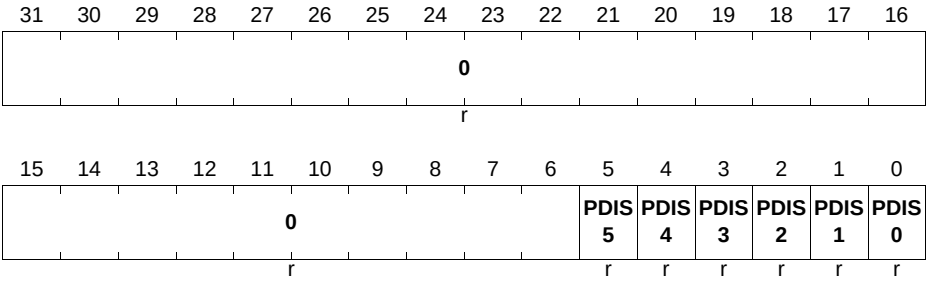
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PDIS 15	**PDIS 14**	**PDIS 13**	**PDIS 12**	**PDIS 11**	**PDIS 10**	**PDIS 9**	**PDIS 8**	**PDIS 7**	**PDIS 6**	**PDIS 5**	**PDIS 4**	**PDIS 3**	**PDIS 2**	**PDIS 1**	**PDIS 0**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

1) The reset value is package dependent.

Field	Bits	Type	Description
PDISx (x = 0-15)	x	r	**Pad Disable for Port 0 Pin x** 0_B Pad P0.x is enabled. 1_B Pad P0.x is disabled.
0	[31:16]	r	**Reserved** Read as 0; should be written with 0.

P1_PDISC
Port 1 Pin Function Decision Control Register

(0060_H) **Reset Value: 0000 00XX$_H$[1)]**

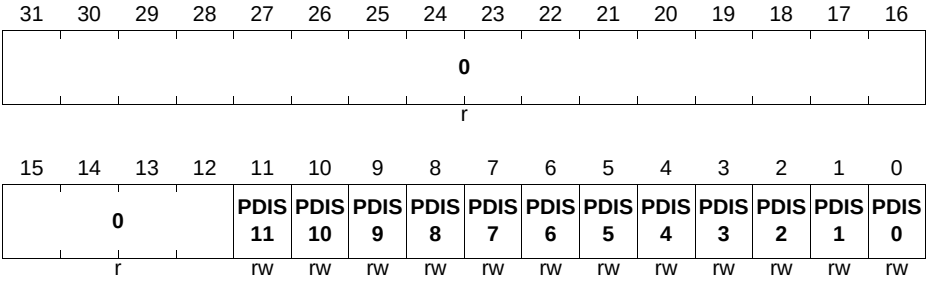


1) The reset value is package dependent.

Field	Bits	Type	Description
PDISx (x = 0-5)	x	r	**Pad Disable for Port 1 Pin x** 0_B Pad P1.x is enabled. 1_B Pad P1.x is disabled.
0	[31:6]	r	**Reserved** Read as 0; should be written with 0.

P2_PDISC
Port 2 Pin Function Decision Control Register

(0060_H) **Reset Value: 0000 0XXX$_H$[1)]**



1) The reset value is package dependent.

Field	Bits	Type	Description
PDIS0	0	rw	**Pad Disable for Port 2 Pin 0** This bit disables or enables the digital pad function. The disabled (default) state of the pad selects the ADC analog input. Upon enabling, the functionality switches to the digital input/output. 0_B Pad is enabled, digital input/output selected. 1_B Pad is disabled, ADC S&H 0 analog input 5 selected. (default)
PDIS1	1	rw	**Pad Disable for Port 2 Pin 1** This bit disables or enables the digital pad function. The disabled (default) state of the pad selects the ADC analog input. Upon enabling, the functionality switches to the digital input/output. 0_B Pad is enabled, digital input/output selected. 1_B Pad is disabled, ADC S&H 0 analog input 6 selected. (default)
PDIS2	2	rw	**Pad Disable for Port 2 Pin 2** This bit disables or enables the digital pad function. The disabled (default) state of the pad selects the ADC analog input. Upon enabling, the functionality switches to the digital input. 0_B Pad is enabled, digital input selected. 1_B Pad is disabled, ADC S&H 0 analog input 7 selected. (default)
PDIS3	3	rw	**Pad Disable for Port 2 Pin 3** This bit disables or enables the digital pad function. The disabled (default) state of the pad selects the ADC analog input. Upon enabling, the functionality switches to the digital input. 0_B Pad is enabled, digital input selected. 1_B Pad is disabled, ADC S&H 1 analog input 5 selected. (default)

Field	Bits	Type	Description
PDIS4	4	rw	**Pad Disable for Port 2 Pin 4** This bit disables or enables the digital pad function. The disabled (default) state of the pad selects the ADC analog input. Upon enabling, the functionality switches to the digital input. 0_B Pad is enabled, digital input selected. 1_B Pad is disabled, ADC S&H 1 analog input 6 selected. (default)
PDIS5	5	rw	**Pad Disable for Port 2 Pin 5** This bit disables or enables the digital pad function. The disabled (default) state of the pad selects the ADC analog input. Upon enabling, the functionality switches to the digital input. 0_B Pad is enabled, digital input selected. 1_B Pad is disabled, ADC S&H 1 analog input 7 selected. (default)
PDIS6	6	rw	**Pad Disable for Port 2 Pin 6** This bit disables or enables the digital pad function. The disabled (default) state of the pad selects the ADC analog input. Upon enabling, the functionality switches to the digital input. 0_B Pad is enabled, digital input selected. 1_B Pad is disabled, ADC S&H 0 analog input 0 selected. (default)
PDIS7	7	rw	**Pad Disable for Port 2 Pin 7** This bit disables or enables the digital pad function. The disabled (default) state of the pad selects the ADC analog input. Upon enabling, the functionality switches to the digital input. 0_B Pad is enabled, digital input selected. 1_B Pad is disabled, ADC S&H 1 analog input 1 selected. (default)
PDIS8	8	rw	**Pad Disable for Port 2 Pin 8** This bit disables or enables the digital pad function. The disabled (default) state of the pad selects the ADC analog input. Upon enabling, the functionality switches to the digital input. 0_B Pad is enabled, digital input selected. 1_B Pad is disabled, ADC S&H 0 analog input 1 and ADC S&H 1 analog input 0 selected. (default)

Field	Bits	Type	Description
PDIS9	9	rw	**Pad Disable for Port 2 Pin 9** This bit disables or enables the digital pad function. The disabled (default) state of the pad selects the ADC analog input. Upon enabling, the functionality switches to the digital input. 0_B Pad is enabled, digital input selected. 1_B Pad is disabled, ADC S&H 0 analog input 2 and ADC S&H 1 analog input 4 selected. (default)
PDIS10	10	rw	**Pad Disable for Port 2 Pin 10** This bit disables or enables the digital pad function. The disabled (default) state of the pad selects the ADC analog input. Upon enabling, the functionality switches to the digital input/output. 0_B Pad is enabled, digital input/output selected. 1_B Pad is disabled, ADC S&H 0 analog input 3 and ADC S&H 1 analog input 2 selected. (default)
PDIS11	11	rw	**Pad Disable for Port 2 Pin 11** This bit disables or enables the digital pad function. The disabled (default) state of the pad selects the ADC analog input. Upon enabling, the functionality switches to the digital input/output. 0_B Pad is enabled, digital input/output selected. 1_B Pad is disabled, ADC S&H 0 analog input 4 and ADC S&H 1 analog input 3 selected. (default)
0	[31:12]	r	**Reserved** Read as 0; should be written with 0.

17.8.4 Port Output Register

The port output register determines the value of a GPIO pin when it is selected by Pn_IOCRx as output. Writing a 0 to a Pn_OUT.Px (x = 0-15) bit position delivers a low level at the corresponding output pin. A high level is output when the corresponding bit is written with a 1. Note that the bits of Pn_OUT.Px can be individually set/reset by writing appropriate values into the port output modification register Pn_OMR, avoiding read-modify-write operations on the Pn_OUT, which might affect other pins of the port.

The Pn_OUT is also used to store/drive a defined value for the input in Deep-Sleep mode. For details on this, see the **Port Pin Power Save Register**. That is also the only use of the Pn_OUT register in the analog and digital input port P2.2 - P2.9.

P0_OUT
Port 0 Output Register **(0000_H)** **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	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
Px (x = 0-15)	x	rwh	**Port 0 Output Bit x** This bit determines the level at the output pin P0.x if the output is selected as GPIO output. 0_B The output level of P0.x is 0. 1_B The output level of P0.x is 1. P0.x can also be set/reset by control bits of the P0_OMR register.
0	[31:16]	r	**Reserved** Read as 0; should be written with 0.

P1_OUT
Port 1 Output Register **(0000_H)** **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										P5	P4	P3	P2	P1	P0
r										rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
Px **(x = 0-5)**	x	rwh	**Port 1 Output Bit x** This bit determines the level at the output pin P1.x if the output is selected as GPIO output. 0_B The output level of P1.x is 0. 1_B The output level of P1.x is 1. P1.x can also be set/reset by control bits of the P1_OMR register.
0	[31:6]	r	**Reserved** Read as 0; should be written with 0.

P2_OUT
Port 2 Output Register **(0000_H)** **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				P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
r				rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
Px **(x = 0-11)**	x	rwh	**Port 2 Output Bit x** This bit determines the level at the output pin P2.x if the output is selected as GPIO output. 0_B The output level of P2.x is 0. 1_B The output level of P2.x is 1. P2.x can also be set/reset by control bits of the P2_OMR register.
0	[31:12]	r	**Reserved** Read as 0; should be written with 0.

17.8.5 Port Output Modification Register

The port output modification register contains control bits that make it possible to individually set, reset, or toggle the logic state of a single port line by manipulating the output register.

P0_OMR
Port 0 Output Modification Register

(0004_H) **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PR15	**PR14**	**PR13**	**PR12**	**PR11**	**PR10**	**PR9**	**PR8**	**PR7**	**PR6**	**PR5**	**PR4**	**PR3**	**PR2**	**PR1**	**PR0**
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PS15	**PS14**	**PS13**	**PS12**	**PS11**	**PS10**	**PS9**	**PS8**	**PS7**	**PS6**	**PS5**	**PS4**	**PS3**	**PS2**	**PS1**	**PS0**
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
PSx (x = 0-15)	x	w	**Port 0 Set Bit x** Setting this bit will set or toggle the corresponding bit in the port output register P0_OUT. The function of this bit is shown in **Table 17-7**.
PRx (x = 0-15)	x + 16	w	**Port 0 Reset Bit x** Setting this bit will reset or toggle the corresponding bit in the port output register P0_OUT. The function of this bit is shown in **Table 17-7**.

P1_OMR
Port 1 Output Modification Register

(0004$_H$) Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0										PR5	PR4	PR3	PR2	PR1	PR0
r										w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0										PS5	PS4	PS3	PS2	PS1	PS0
r										w	w	w	w	w	w

Field	Bits	Type	Description
PSx (x = 0-5)	x	w	**Port 1 Set Bit x** Setting this bit will set or toggle the corresponding bit in the port output register P1_OUT. The function of this bit is shown in **Table 17-7**.
PRx (x = 0-5)	x + 16	w	**Port 1 Reset Bit x** Setting this bit will reset or toggle the corresponding bit in the port output register P1_OUT. The function of this bit is shown in **Table 17-7**.
0	[15:6], [31:22]	r	**Reserved** Read as 0; should be written with 0.

P2_OMR
Port 2 Output Modification Register

(0004$_H$) Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0				PR11	PR10	PR9	PR8	PR7	PR6	PR5	PR4	PR3	PR2	PR1	PR0
r				w	w	w	w	w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				PS11	PS10	PS9	PS8	PS7	PS6	PS5	PS4	PS3	PS2	PS1	PS0
r				w	w	w	w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
PSx (x = 0-11)	x	w	**Port 2 Set Bit x** Setting this bit will set or toggle the corresponding bit in the port output register P2_OUT. The function of this bit is shown in **Table 17-7**.
PRx (x = 0-11)	x + 16	w	**Port 2 Reset Bit x** Setting this bit will reset or toggle the corresponding bit in the port output register P2_OUT. The function of this bit is shown in **Table 17-7**.
0	[15:12], [31:28]	r	**Reserved** Read as 0; should be written with 0.

Note: Register Pn_OMR is virtual and does not contain any flip-flop. A read action delivers the value of 0. A 8 or 16-bits write behaves like a 32-bit write padded with zeros.

Table 17-7 Function of the Bits PRx and PSx

PRx	PSx	Function
0	0	Bit Pn_OUT.Px is not changed.
0	1	Bit Pn_OUT.Px is set.
1	0	Bit Pn_OUT.Px is reset.
1	1	Bit Pn_OUT.Px is toggled.

17.8.6 Port Input Register

The logic level of a GPIO pin can be read via the read-only port input register Pn_IN. Reading the Pn_IN register always returns the current logical value at the GPIO pin, synchronized to avoid meta-stabilities, independently whether the pin is selected as input or output.

P0_IN
Port 0 Input Register **(0024_H)** **Reset Value: 0000 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
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
Px **(x = 0-15)**	x	rh	**Port 0 Input Bit x** This bit indicates the level at the input pin P0.x. 0_B The input level of P0.x is 0. 1_B The input level of P0.x is 1.
0	[31:16]	r	**Reserved** Read as 0.

P1_IN
Port 1 Input Register **(0024_H)** **Reset Value: 0000 00XX$_H$**

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										P5	P4	P3	P2	P1	P0
r										rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
Px **(x = 0-5)**	x	rh	**Port 1 Input Bit x** This bit indicates the level at the input pin P1.x. 0_B The input level of P1.x is 0. 1_B The input level of P1.x is 1.
0	[31:6]	r	**Reserved** Read as 0.

P2_IN
Port 2 Input Register **(0024_H)** **Reset Value: 0000 0XXX$_H$**

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				**P11**	**P10**	**P9**	**P8**	**P7**	**P6**	**P5**	**P4**	**P3**	**P2**	**P1**	**P0**
r				rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
Px **(x = 0-11)**	x	rh	**Port 2 Input Bit x** This bit indicates the level at the input pin P2.x. 0_B The input level of P2.x is 0. 1_B The input level of P2.x is 1.
0	[31:12]	r	**Reserved** Read as 0.

17.8.7 Port Pin Power Save Register

When the XMC1100 enters Deep-Sleep mode, pins with enabled Pin Power Save option are set to a defined state and the input Schmitt-Trigger as well as the output driver stage are switched off.

Note: Do not enable the Pin Power Save function for pins configured for Hardware Control (Pn_HWSEL.HWx != 00_B). Doing so may result in an undefined behavior of the pin when the device enters the Deep Sleep state.

P0_PPS
Port 0 Pin Power Save Register

(0070_H) **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
PPS 15	**PPS 14**	**PPS 13**	**PPS 12**	**PPS 11**	**PPS 10**	**PPS 9**	**PPS 8**	**PPS 7**	**PPS 6**	**PPS 5**	**PPS 4**	**PPS 3**	**PPS 2**	**PPS 1**	**PPS 0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
PPSx (x = 0-15)	x	rw	**Port 0 Pin Power Save Bit x** 0_B Pin Power Save of P0.x is disabled. 1_B Pin Power Save of P0.x is enabled.
0	[31:16]	r	**Reserved** Read as 0.

P1_PPS
Port 1 Pin Power Save Register

(0070$_H$) 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										PPS 5	PPS 4	PPS 3	PPS 2	PPS 1	PPS 0
r										rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
PPSx (x = 0-5)	x	rw	**Port 1 Pin Power Save Bit x** 0_B Pin Power Save of P1.x is disabled. 1_B Pin Power Save of P1.x is enabled.
0	[31:6]	r	**Reserved** Read as 0.

P2_PPS
Port 2 Pin Power Save Register

(0070$_H$) 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				PPS 11	PPS 10	PPS 9	PPS 8	PPS 7	PPS 6	PPS 5	PPS 4	PPS 3	PPS 2	PPS 1	PPS 0
r				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
PPSx (x = 0-11)	x	rw	**Port 2 Pin Power Save Bit x** 0_B Pin Power Save of P2.x is disabled. 1_B Pin Power Save of P2.x is enabled.
0	[31:12]	r	**Reserved** Read as 0.

Deep-Sleep Pin Power Save behavior

The actual behavior in Deep-Sleep mode with enabled Pin Power Save is controlled by the Pn_IOCRy.PCx bit field (**Page 17-14**) of the respective pin. **Table 17-8** shows the coding.

Table 17-8 PCx Coding in Deep-Sleep mode

PCx[4:0]	I/O	Normal Operation or PPSx=0_B	Deep-Sleep mode and PPSx=1_B
$0X000_B$	Direct Input	See **Table 17-5**	Input value=Pn_OUTx
$0X001_B$			Input value=0_B; pull-down deactivated
$0X010_B$			Input value=1_B; pull-up deactivated
$0X011_B$			Input value=Pn_OUTx, storing the last sampled input value
$0X100_B$	Inverted Input	See **Table 17-5**	Input value=$\overline{\text{Pn\_OUTx}}$
$0X101_B$			Input value=1_B; pull-down deactivated
$0X110_B$			Input value=0_B; pull-up deactivated
$0X111_B$			Input value=$\overline{\text{Pn\_OUTx}}$, storing the last sampled input value
$1XXXX_B$	Output	See **Table 17-5**	Output driver off, Input Schmitt-Trigger off, no pull device active, Input value=Pn_OUTx

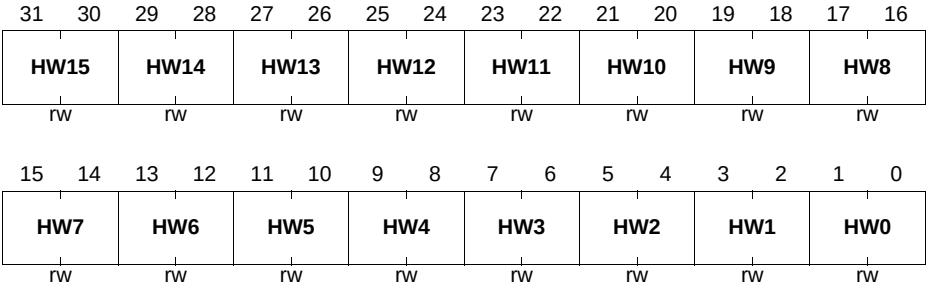
17.8.8 Port Pin Hardware Select Register

Some peripherals require direct hardware control of their I/Os. As multiple such peripheral I/Os are mapped on some pins, the register Pn_HWSEL is used to select which peripheral has the control over the pin.

Note: Pn_HWSEL.HWx just pre-assigns the hardware-control of the pin to a certain peripheral, but the peripheral itself decides when to take control over it. As long as the peripheral does not take control of a given pin via HWx_EN, the configuration of this pin is still defined by the configuration registers and it is available as GPIO or for other alternate functions. This might be because the selected peripheral has provisions to just activate a subset of its pins, or because the peripheral is not active at all.

This mechanism can also be used to prohibit the hardware control of certain pins to a peripheral, in case the application does not need the respective functionality and the peripheral has no provisions to disable the hardware control selectively.

P0_HWSEL
Port 0 Pin Hardware Select Register (0074_H) **Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
HWx (x = 0-15)	[2*x+1: 2*x]	rw	**Port 0 Pin Hardware Select Bit x** 00_B Software control only. 01_B HW0 control path can override the software configuration. 10_B HW1 control path can override the software configuration. 11_B Reserved.

P1_HWSEL
Port 1 Pin Hardware Select Register (0074$_H$) **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				HW5		HW4		HW3		HW2		HW1		HW0	
r				rw		rw		rw		rw		rw		rw	

Field	Bits	Type	Description
HWx (x = 0-5)	[2*x+1: 2*x]	rw	**Port 1 Pin Hardware Select Bit x** 00_B Software control only. 01_B HW0 control path can override the software configuration. 10_B HW1 control path can override the software configuration. 11_B Reserved.
0	[31:12]	r	**Reserved** Read as 0; should be written with 0.

P2_HWSEL
Port 2 Pin Hardware Select Register (0074$_H$) **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								HW11		HW10		HW9		HW8	
r								rw		rw		rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HW7		HW6		HW5		HW4		HW3		HW2		HW1		HW0	
rw		rw		rw		rw		rw		rw		rw		rw	

Field	Bits	Type	Description
HWx (x = 0-11)	[2*x+1: 2*x]	rw	**Port 2 Pin Hardware Select Bit x** 00_B Software control only. 01_B HW0 control path can override the software configuration. 10_B HW1 control path can override the software configuration. 11_B Reserved.
0	[31:24]	r	**Reserved** Read as 0; should be written with 0.

17.9 Package Pin Summary

The following general building block is used to describe each pin:

Table 17-9 Package Pin Mapping Description

Function	Package A	Package B	...	Pad Type
Px.y	N	N		Pad Class

The table is sorted by the “Function” column, starting with the regular Port pins (Px.y), followed by the supply pins.

The following columns, titled with the supported package variants, lists the package pin number to which the respective function is mapped in that package.

The “Pad Type” indicates the employed pad type:

- STD_INOUT/AN (standard bi-directional pads with analog input)
- High Current (high current bi-directional pads)
- STD_IN/AN (standard input pads with analog input)
- Power (power supply)

Details about the pad properties are defined in the Datasheet.

Table 17-10 Package Pin Mapping

Function	TSSOP 38	TSSOP 16	Pad Type	Notes
P0.0	17	7	STD_INOUT/AN	
P0.1	18	-	STD_INOUT/AN	
P0.2	19	-	STD_INOUT/AN	
P0.3	20	-	STD_INOUT/AN	
P0.4	21	-	STD_INOUT/AN	
P0.5	22	8	STD_INOUT/AN	
P0.6	23	9	STD_INOUT/AN	
P0.7	24	10	STD_INOUT/AN	
P0.8	27	11	STD_INOUT/AN	
P0.9	28	12	STD_INOUT/AN	
P0.10	29	-	STD_INOUT/AN	
P0.11	30	-	STD_INOUT/AN	
P0.12	31	-	STD_INOUT/AN	
P0.13	32	-	STD_INOUT/AN	

Table 17-10 Package Pin Mapping (cont'd)

Function	TSSOP 38	TSSOP 16	Pad Type	Notes
P0.14	33	13	STD_INOUT/AN	
P0.15	34	14	STD_INOUT/AN	
P1.0	16	-	High Current	
P1.1	15	-	High Current	
P1.2	14	-	High Current	
P1.3	13	-	High Current	
P1.4	12	-	High Current	
P1.5	11	-	High Current	
P2.0	35	15	STD_INOUT/AN	
P2.1	36	-	STD_INOUT/AN	
P2.2	37	-	STD_IN/AN	
P2.3	38	-	STD_IN/AN	
P2.4	1	-	STD_IN/AN	
P2.5	2	-	STD_IN/AN	
P2.6	3	16	STD_IN/AN	
P2.7	4	1	STD_IN/AN	
P2.8	5	1	STD_IN/AN	
P2.9	6	2	STD_IN/AN	
P2.10	7	3	STD_INOUT/AN	
P2.11	8	4	STD_INOUT/AN	
VSS	9	5	Power	Supply GND, ADC reference GND
VDD	10	6	Power	Supply VDD, ADC reference voltage
VSSP	25	-	Power	I/O port ground
VDDP	26	-	Power	I/O port supply

17.10 Port I/O Functions

17.10.1 Port Pin for Boot Modes

Port functions can be overruled by the boot mode selected. The type of boot mode is selected via BMI. **Table 17-11** shows the port pins used for the various boot modes.

Table 17-11 Port Pin for Boot Modes

Pin	Boot	Boot Description
P0.13	CS(O)	SSC BSL mode
P0.14	SWDIO_0	Debug mode (SWD)
	SPD_0	Debug mode (SPD)
	RX/TX	ASC BSL half-duplex mode
	RX	ASC BSL full-duplex mode
	SCLK(O)	SSC BSL mode
P0.15	SWDCLK_0	Debug mode (SWD)
	TX	ASC BSL full-duplex mode
	DATA(I/O)	SSC BSL mode
P1.2	SWDCLK_1	Debug mode (SWD)
	TX	ASC BSL full-duplex mode
P1.3	SWDIO_1	Debug mode (SWD)
	SPD_1	Debug mode (SPD)
	RX/TX	ASC BSL half-duplex mode
	RX	ASC BSL full-duplex mode

17.10.2 Port I/O Function Description

The following general building block is used to describe each PORT pin:

Table 17-12 Port I/O Function Description

Function	Outputs			Inputs		
	ALT1	ALTn	HWO0	HWI0	Input	Input
P0.0		MODA.OUT	MODB.OUT	MODB.INA	MODC.INA	
Pn.y	MODA.OUT				MODA.INA	MODC.INB

Pn.y is the port pin name, defining the control and data bits/registers associated with it. As GPIO, the port is under software control. Its input value is read via Pn_IN.y, Pn_OUT defines the output value.

Up to seven alternate output functions (ALT1/2/3/4/5/6/7) can be mapped to a single port pin, selected by Pn_IOCR.PC. The output value is directly driven by the respective module, with the pin characteristics controlled by the port registers (within the limits of the connected pad).

The port pin input can be connected to multiple peripherals. Most peripherals have an input multiplexer to select between different possible input sources.

The input path is also active while the pin is configured as output. This allows to feedback an output to on-chip resources without wasting an additional external pin.

By Pn_HWSEL (**Chapter 17.8.8**) it is possible to select between different hardware “masters” (HWO0/HWI0, HWO1/HWI1). The selected peripheral can take control of the pin(s). Hardware control overrules settings in the respective port pin registers.

Table 17-13 Port I/O Functions

Function	Outputs									Inputs								
	ALT1	ALT2	ALT3	ALT4	ALT5	ALT6	ALT7	HWO0	HWO1	HWI0	HWI1	Input	Input	Input	Input	Input	Input	Input
P0.0	ERU0. PDOUT0		ERU0. GOUT0	CCU40.OUT0		USIC0_CH0. SELO0	USIC0_CH1. SELO0					CCU40.IN0C				USIC0_CH0. DX2A	USIC0_CH1. DX2A	
P0.1	ERU0. PDOUT1		ERU0. GOUT1	CCU40.OUT1			SCU. VDROP					CCU40.IN1C						
P0.2	ERU0. PDOUT2		ERU0. GOUT2	CCU40.OUT2		VADC0. EMUX02						CCU40.IN2C						
P0.3	ERU0. PDOUT3		ERU0. GOUT3	CCU40.OUT3		VADC0. EMUX01						CCU40.IN3C						
P0.4				CCU40.OUT1		VADC0. EMUX00	WWDT. SERVICE_OUT											
P0.5				CCU40.OUT0														
P0.6				CCU40.OUT0		USIC0_CH1. MCLKOUT	USIC0_CH1. DOUT0					CCU40.IN0B				USIC0_CH1. DX0C		
P0.7				CCU40.OUT1		USIC0_CH0. SCLKOUT	USIC0_CH1. DOUT0					CCU40.IN1B				USIC0_CH0. DX1C	USIC0_CH1. DX0D	USIC0_CH1. DX1C
P0.8				CCU40.OUT2		USIC0_CH0. SCLKOUT	USIC0_CH1. SCLKOUT					CCU40.IN2B				USIC0_CH0. DX1B	USIC0_CH1. DX1B	
P0.9				CCU40.OUT3		USIC0_CH0. SELO0	USIC0_CH1. SELO0					CCU40.IN3B				USIC0_CH0. DX2B	USIC0_CH1. DX2B	
P0.10						USIC0_CH0. SELO1	USIC0_CH1. SELO1									USIC0_CH0. DX2C	USIC0_CH1. DX2C	
P0.11				USIC0_CH0. MCLKOUT		USIC0_CH0. SELO2	USIC0_CH1. SELO2									USIC0_CH0. DX2D	USIC0_CH1. DX2D	
P0.12						USIC0_CH0. SELO3						CCU40.IN0A	CCU40.IN1A	CCU40.IN2A	CCU40.IN3A	USIC0_CH0. DX2E		
P0.13	WWDT. SERVICE_OUT					USIC0_CH0. SELO4										USIC0_CH0. DX2F		
P0.14						USIC0_CH0. DOUT0	USIC0_CH0. SCLKOUT									USIC0_CH0. DX0A	USIC0_CH0. DX1A	
P0.15						USIC0_CH0. DOUT0	USIC0_CH1. MCLKOUT									USIC0_CH0. DX0B		
P1.0		CCU40.OUT0					USIC0_CH0. DOUT0		USIC0_CH0. DOUT0		USIC0_CH0. HWIN0					USIC0_CH0. DX0C		
P1.1	VADC0. EMUX00	CCU40.OUT1				USIC0_CH0. DOUT0	USIC0_CH1. SELO0		USIC0_CH0. DOUT1		USIC0_CH0. HWIN1					USIC0_CH0. DX0D	USIC0_CH0. DX1D	USIC0_CH1. DX2E
P1.2	VADC0. EMUX01	CCU40.OUT2					USIC0_CH1. DOUT0		USIC0_CH0. DOUT2		USIC0_CH0. HWIN2					USIC0_CH1. DX0B		
P1.3	VADC0. EMUX02	CCU40.OUT3				USIC0_CH1. SCLKOUT	USIC0_CH1. DOUT0		USIC0_CH0. DOUT3		USIC0_CH0. HWIN3					USIC0_CH1. DX0A	USIC0_CH1. DX1A	
P1.4	VADC0. EMUX10	USIC0_CH1. SCLKOUT				USIC0_CH0. SELO0	USIC0_CH1. SELO1									USIC0_CH0. DX5E	USIC0_CH1. DX5E	
P1.5	VADC0. EMUX11	USIC0_CH0. DOUT0				USIC0_CH0. SELO1	USIC0_CH1. SELO2									USIC0_CH1. DX5F		

Table 17-13 Port I/O Functions (cont'd)

Function	Outputs									Inputs									
	ALT1	ALT2	ALT3	ALT4	ALT5	ALT6	ALT7	HWO0	HWO1	HWI0	HWI1	Input	Input	Input	Input	Input	Input	Input	Input
P2.0	ERU0. PDOUT3	CCU40.OUT0	ERU0. GOUT3			USIC0_CH0. DOUT0	USIC0_CH0. SCLKOUT						VADC0. G0CH5		ERU0.0B0	USIC0_CH0. DX0E	USIC0_CH0. DX1E	USIC0_CH1. DX2F	
P2.1	ERU0. PDOUT2	CCU40.OUT1	ERU0. GOUT2			USIC0_CH0. DOUT0	USIC0_CH1. SCLKOUT						VADC0. G0CH6		ERU0.1B0	USIC0_CH0. DX0F	USIC0_CH1. DX3A	USIC0_CH1. DX4A	
P2.2													VADC0. G0CH7		ERU0.0B1	USIC0_CH0. DX3A	USIC0_CH0. DX4A	USIC0_CH1. DX5A	
P2.3													VADC0. G1CH5		ERU0.1B1	USIC0_CH0. DX5B	USIC0_CH1. DX3C	USIC0_CH1. DX4C	
P2.4													VADC0. G1CH6		ERU0.0A1	USIC0_CH0. DX3B	USIC0_CH0. DX4B	USIC0_CH1. DX5B	
P2.5													VADC0. G1CH7		ERU0.1A1	USIC0_CH0. DX5D	USIC0_CH1. DX3E	USIC0_CH1. DX4E	
P2.6													VADC0. G0CH0		ERU0.2A1	USIC0_CH0. DX3E	USIC0_CH0. DX4E	USIC0_CH1. DX5D	
P2.7													VADC0. G1CH1		ERU0.3A1	USIC0_CH0. DX5C	USIC0_CH1. DX3D	USIC0_CH1. DX4D	
P2.8													VADC0. G0CH1	VADC0. G1CH0	ERU0.3B1	USIC0_CH0. DX3D	USIC0_CH0. DX4D	USIC0_CH1. DX5C	
P2.9													VADC0. G0CH2	VADC0. G1CH4	ERU0.3B0	USIC0_CH0. DX5A	USIC0_CH1. DX3B	USIC0_CH1. DX4B	
P2.10	ERU0. PDOUT1	CCU40.OUT2	ERU0. GOUT1				USIC0_CH1. DOUT0						VADC0. G0CH3	VADC0. G1CH2	ERU0.2B0	USIC0_CH0. DX3C	USIC0_CH0. DX4C	USIC0_CH1. DX0F	
P2.11	ERU0. PDOUT0	CCU40.OUT3	ERU0. GOUT0			USIC0_CH1. SCLKOUT	USIC0_CH1. DOUT0						VADC0. G0CH4	VADC0. G1CH3	ERU0.2B1	USIC0_CH1. DX0E	USIC0_CH1. DX1E		

Boot and Startup, Bootstrap Loaders, and User Routines

18 Boot and Startup

The startup sequence of the XMC1100 is a process taking place before user application software takes control of the system and is comprising of two major phases (see **Figure 18-1**), split in several distinctive steps:

Hardware Controlled Startup Phase

The hardware controlled startup phase gets performed automatically after power up of the microcontroller. This part is generic and it ensures basic configuration common to most applications. The hardware setup needs to ensure fulfillment of requirements specified in Data Sheet in order to enable reliable start up of the microcontroller before control is handed over to the user software. The sequence where boot code gets executed is considered a part of the hardware controlled phase of the startup sequence.

For details of the setup requirements, please refer to Data Sheet.

Software Controlled Startup Phase

The software controlled startup phase is the part where the application specific configuration gets applied with user software. It involves several steps that are critical for proper operation of the microcontroller in the application context and may also involve some optional configuration actions in order to improve system performance and stability in the application context.

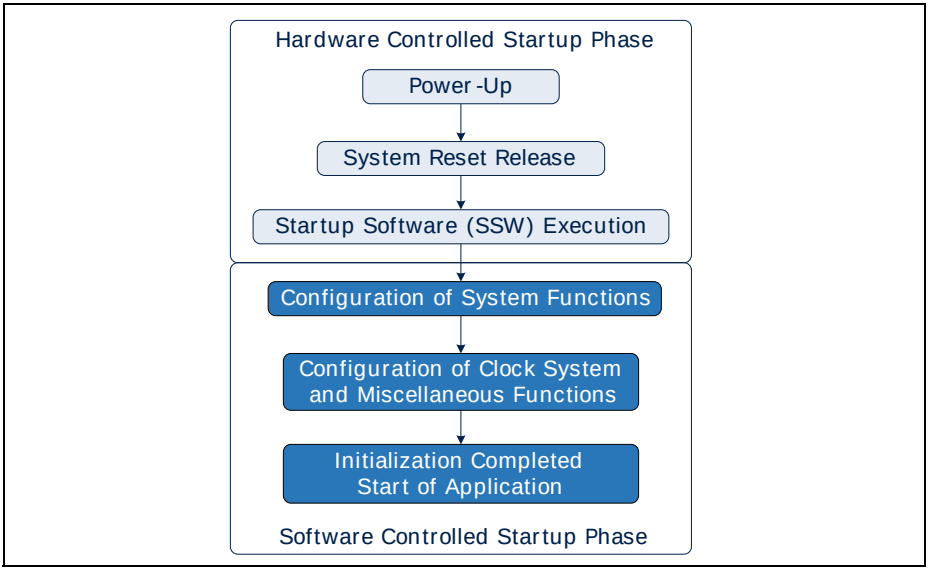


Figure 18-1 Startup sequence

18.1 Startup Sequence and System Dependencies

A more detailed description of the startup sequence is available in the following sub-chapters.

18.1.1 Power-Up

Power up of the microcontroller gets performed by applying V_{DDP} supply (for details of the supply requirements, please refer to Data Sheet). Once V_{DDP} reaches a threshold, the internal core voltage, V_{DDC}, is generated automatically by the EVR.

18.1.2 System Reset Release

The internal power-on reset generation is based on the supply and core voltage validation. When V_{DDP} and V_{DDC} reaches a stable threshold level, the power-on reset is released. Next, after the on-chip oscillators generate a stable clock output, the system reset is released automatically and the SSW code starts to run.

The system reset can be triggered from various sources like software controlled CPU reset register or a watchdog time-out-triggered reset. For more details on reset control details, please refer to the Reset Control Unit (RCU) section in the SCU chapter.

The cause of the last reset gets automatically stored in the SCU_RSTSTAT register and can be checked by user software to determine the state of the system and for debug purpose. The reset status in the SCU_RSTSTAT register shall be reset with SCU_RSTCLR register after each startup in order to ensure consistent source indication after the next reset.

After reset release, MCLK and PCLK are running at 8MHz and most of the peripherals' clocks are disabled except for CPU, memories and PORT. It is recommended to disable the clock of the unused modules in order to reduce power consumption.

18.1.3 Startup Software (SSW) Execution

After the reset is deasserted, CPU starts to execute the SSW code from the ROM memory. To indicate the start of the SSW execution, the pullup device in P0.8 is enabled. Pullup device is disabled during reset.

Another important task that is done by SSW is to read the Boot Mode Index (BMI) stored in Flash and decide the startup mode (such as User Productive mode and Bootstrap Loader) selected by the user. For more details about the various startup modes and handling of BMI, please refer to **Section 18.2**.

To handle the initial hardfault error during the SSW execution, SSW installs "jump to itself" instruction at SRAM location $2000'000C_H$ as temporary HardFault handler - until the user code installs its own. It is installed upon master reset only.

During SSW execution, the SRAM area between $2000'00C0_H$ and $2000'0200_H$ is reserved for usage by XMC1100 SSW, therefore the user software should not store in this area data which must be preserved throughout (non-power) resets.

The MCLK and PCLK frequency that is used to execute the SSW and to start the user code can be configured by user. In addition, some of the peripheral clock can also be enabled (default disabled) by the SSW. Detailed description can be found in the **Section 18.1.3.1**.

18.1.3.1 Clock system handling by SSW

XMC1100 SSW is able to change device clock settings to allow flexibility of device performance e.g. speed vs. power consumption optimization during start-up.

The clock handling by SSW is user configurable by installing values in Flash. For this purpose two word locations - CLK_VAL1 and CLK_VAL2 - are assigned in Flash (refer to **Table 18-1**), the values from these locations are processed by SSW as follows:

- if CLK_VAL1[31]=0 - CLK_VAL1[19:0] bits are installed into SCU_CLKCR[19:0] register - this configures clock dividers and selection
- if CLK_VAL2[31]=0 - CLK_VAL2[10:0] bits are installed into SCU_CGATCLR0[10:0] - this enables the clock of the selected peripherals

In case some CLK_VALx location is not programmed by the user - like in device delivery state - its bit[31]=1 and no installation is done into the respective register(s).

Limited device frequency change

To avoid big jumps/drops in power consumption, the user-configurable value to be installed during start-up from CLK_VAL1 into CLKCR.IDIV is limited to the range 1..16, resulting in possible range of MCLK after re-configuration 2..32 MHz from initial 8MHz - i.e. MCLK frequency can be increased or decreased no more than 4 times (rounded, ignoring potential FDIV influence).

In case CLK_VAL1 (refer to **Table 18-1**) contains IDIV value out of the 1..16 range:

- if IDIV=0 - SSW installs IDIV=1 instead
- if IDIV>16 - SSW install IDIV=16 instead

18.1.4 Configuration of Special System Functions as part of User code initialization

Special system functions are may be required to perform actions that improve system stability and robustness. The following special system functions or modules require initialization as part of the User code initialization before the actual user application starts:

- Start Address and Initial Stack Pointer Value
- Setup the Interrupt handler

- Supply Voltage Brown-out Detection
- Watchdog Timer (WDT)

Start Address and Initial Stack Pointer Value

The start address and the inital stack pointer values in **Table 18-1** need to be defined by the user. It can be done together with the downloading of the user code into the flash memory.

Setup the Interrpt Handler

The vector table is remapped to the SRAM based on the remapped vector table in CPU chapter. The interrupt service routine can be placed in these SRAM address or user can also have a branch instruction to the routine that is placed in other memory location. For details, please refer to CPU chapter.

V_{DDP} Brown Out Detection

V_{DDP} brown out detection mechanism allow active monitoring of the supply voltage and a corrective reaction in case the voltage level is below a programmed threshold. An interrupt request will be flagged if a programmed condition is detected. For details, please refer to the Power Control Unit (PCU) section in the SCU chapter.

Watchdog Timer

The Watchdog Timer requires a clock source selection and activation. It is highly recommended to use reliable clock source, preferably independent from the system clock source in order to ensure corrective action in case of a system failure which will bring the microcontroller into a safe operation state. In XMC1100, the default WDT clock is the standby clock. For more details, please refer to WDT chapter. For details of the WDT module configuration please refer to the “Initialization and Control Sequence” section of the WDT chapter.

18.1.5 Configuration of Clock System and Miscellaneous Functions

The following functions are available and may require configuration in the user code:

- Clock System Configuration
- System Reset Configuration
- Debug Suspend Configuration

Clock System Configuration

MCLK and PCLK frequency can be configured to be different from the user settings during SSW execution as described in **Section 18.1.3.1**. The SCU_CRCLK.IDIV/FDIV bits are used to program the target clock frequency. The ratio between MCLK and PCLK is also selectable via bit SCU_CRCLK.PCLKSEL. In addition, some of the peripheral

clocks can be enable/disable via the SCU_CGATCLR0/SCU_CGATSET0 register respectively. It is also recommended to enable the oscillator watchdog for loss of clock detection.

System Reset Configuration

Several events such as Flash ECC error or SRAM parity error can be used to trigger a system reset using the SCU_RSTCON register. For details, please refer to the reset control unit (RCU) section in the SCU chapter.

Debug Suspend Configuration

The XMC1100 device supports a suspend capability for peripherals, if the program execution of the CPU is stopped by the debugger, e.g. with a breakpoint,or with the C_HALT. This allows the debugging of critical states of the whole microcontroller. The suspend function has to be enabled or disabled locally at the peripheral based on the application and the debugging approach. Refer to Debug System chapter for details.

18.2 Start-up Modes

Startup Software (short name SSW) is the first software executed by CPU after any device leaves reset state.

SSW is stored in the ROM memory, but nevertheless its flow is influenced by other “flexible” factors as:

- type of the event triggering SSW execution
- start-up configuration as selected by the user in so called Boot Mode Index (short notation BMI)

18.2.1 Start-up modes in XMC1100

Start-up mode selection in XMC1100 is done by the SSW after reset. It based on **Boot Mode Index (BMI)** stored in flash configuration sector 0 (CS0) meaning no device pin is used for configuration purpose. To program a new BMI, user need to call the user function “Request BMI installation” as described in the BSL and User Routines chapter. The selection and handling of BMI by SSW is described in **Section 18.2.3**. The default start-up mode in device delivery state is the ASC BSL mode.

A summary of the supported start-up modes follows.

18.2.1.1 User productive mode

User code is started from Flash memory, no debugging is possible in this mode.

The address of the first user instruction - to where SSW jumps at its end - is taken from Flash location $1000'1004_H$.

SSW does not check either the target address is inside user Flash. Therefore HardFault will be generated if it's outside and the execution will jump to exception handler in SRAM (upon master reset an endless loop is installed by SSW).

18.2.1.2 User mode with debug enabled

User code is started from Flash memory, the first address is determined as in **User productive mode** (see **Section 18.2.1.1**).

Debug interface is configured according to **Boot Mode Index (BMI)** value and enabled.

18.2.1.3 User mode with debug enabled and Halt After Reset (HAR)

User code is started from Flash memory as in the two modes above.

Debug interface is configured according to BMI value and enabled, the CPU is stalled upon exit from SSW and before the first user instruction (for the handling refer to **Section 18.2.3.2**).

18.2.1.4 Standard Bootstrap Loader modes

In this mode (short notation Standard BSL):

- user code is downloaded into SRAM
- at the SSW end the execution jumps to SRAM
- no debugging is possible

The Standard BSL mode supported in XMC1100 uses USIC0 module for communication with the host:

- ASC - using channel 0 or 1 (auto-detection by firmware) of USIC0 module and ASC (UART) protocol for download
  This is the default start-up mode - selected by BMI value in device delivery state.
- SSC - using USIC0 channel 0 and SSC (SPI) standard protocol for download
  In this mode SPI-compatible serial EEPROM is supposed to be connected as slave device to XMC1100.

The functionality of standard BSL routines is described in the BSL chapter.

18.2.1.5 Bootstrap Loader modes with time-out

These modes are using the Standard BSL routines for downloading, but the functionality is different:

- SSW first checks either an external device is connected/active, meaning SSW waits to receive Start and Header Bytes from the host - either via USIC channel 0 or 1 - for time-out which duration is taken from BMI.BLSTO (refer to **Section 18.2.2**):

  - if yes
    * user code is downloaded into SRAM
    * at the SSW end the execution jumps to SRAM
  - if not - at the SSW end the execution jumps to Flash as in **User productive mode**
- no debugging is possible in this mode, independently either code execution starts from Flash or SRAM

Two BSL modes with time-out are supported in XMC1100, both using USIC0 module for communication with the host. Besides the difference in interface selection, also the check performed as part from the above sequence is different:

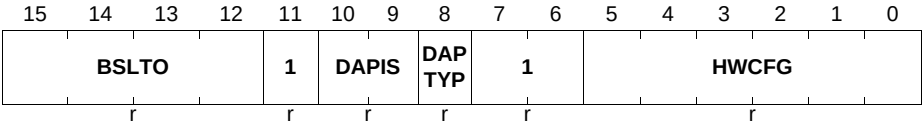
- ASC mode
  - SSW configures USIC0 channels 0 and 1 in ASC (UART) mode
  - SSW waits to receive Start and Header Bytes from the host - either via channel 0 or 1 - for time-out which duration is taken from BMI.BLSTO (refer to **Section 18.2.2**)
  - if Start+Header Bytes are received - Standard ASC BSL is further executed for downloading and starting user code from SRAM; otherwise execution from Flash will be taken
- SSC mode
  - SSW configures USIC0 channel 0 as Master in SSC (SPI) mode
  - SSW initiates communication with SPI-compatible serial EEPROM supposed to be connected as slave device
  - if data received upon Read Command sent by SSW corresponds to the expected (serial EEPROM) protocol - Standard SSC BSL is further executed for downloading and starting user code from SRAM; otherwise execution from Flash will be taken

Note: If BSLTO=0 is configured in BMI - SSW at all does not perform any BSL check but directly takes User Productive Mode.

18.2.2 Boot Mode Index (BMI)

The Boot Mode Index is 2 Byte value stored in Flash (refer to **Table 18-1**) and holding information about start-up mode and debug configuration of the device. Additionally to BMI at 1000'0E00$_H$ its inverse value is stored at 1000'0E10$_H$ for check purposes.

BMI
Boot Mode Index **CS0 offset 0E00$_H$** **Delivery state: FFC0$_H$**



Field	Bits	Typ	Description
HWCFG	[5:0]	r	**Start-up mode selection:** 000000_B ASC Bootstrap Loader mode (ASC_BSL) 000001_B User productive Mode (UPM) 000011_B User mode with debug enabled (UMD) 000111_B User mode with debug enabled and HAR (UMHAR) 001000_B SSC Bootstrap Loader mode (SSC_BSL) 010000_B ASC BSL mode with time-out (ASC_BSLTO) 011000_B SSC BSL mode with time-out (SSC_BSLTO) 101010_B Secure Bootstrap Loader mode (SBSL) 1) else Not defined 2)
DAPTYP	8	r	**DAP Type Selection** Coding as of SCU_DAPCON.DAPTYP
DAPDIS	[10:9]	r	**SWD/SPD Input/Output Selection** Coding as of SCU_DAPCON.DAPDIS
BSLTO	[15:12]	r	**ASC BSL Time-out value** The time-out duration is BSLTO*2664000 MCLK cycles, the supported time-out range is 0.3-5s (333...4995ms)
1	[7:6], 11	r	**Reserved, must be programmed to 1**

1) Only in device versions supporting it

2) ATTENTION: Installing one of these values will cause device delivery state to be restored upon the next reset - refer to **Section 18.2.3**

18.2.3 Start-up mode selection

The start-up modes in XMC1100 are selected and handled according to Boot Mode Index (BMI) value read from Flash - refer to **Section 18.2.1**.

The BMI evaluation can lead to:

- taking User mode - SSW will jump at its end into User Flash and start execution from the location which address is stored at $1000'1004_H$ if:
  - a kind of User Mode is selected in BMI
  - ASC BSL with time-out is selected but no valid Start+Header Bytes are received within the selected time frame
- taking Bootstrap Loader mode - executing ASC BSL then jumping into SRAM at $2000'0200_H$
- taking Secure Bootstrap Loader (SBSL) mode - if SBSL is selected in BMI and supported for the device
- executing (endless) loop - waiting for valid Start+Header Bytes to be received via ASC upon ASC BSL mode selected without time-out
- erasing the complete user Flash and installing ASC BSL (or SBSL if supported) mode as new BMI value - upon invalid BMI value or if Secure BSL is selected but not supported for the device. Afterwards master reset is triggered and device is started in a mode according to the new BMI.

18.2.3.1 BMI handling by SSW

This SSW part is intended to be executed after the user function (in ROM) “Request BMI installation” (described in the BSL and User Routines chapter) has been called and has triggered a system reset. Therefore the conditions under which this handling is performed are:

- the last reset was a system reset requested by CPU
- the two half words of SSW0 register are inverse to each other

If all the above conditions are true, the SSW considers SSW0[15:0] content as new BMI value to be installed and executes the following:

- check either the current BMI (in CS0) is User_Productive AND the new value is NOT User_Productive:
  - if yes - erase the complete user Flash and install ASC_BSL (or SBSL if supported) mode as new BMI value
  - if not - install the new BMI value from SSW0[15:0]
- instal all zero in SSW0 to indicate no BMI-programming upon the next reset
- request master reset to be triggered

Upon the last action, a master reset is triggered and a next SSW execution will start with the new BMI value installed in CS0.

18.2.3.2 Debug system handling

If debug system must be enabled - start-up mode with debug support is selected in Boot Mode Index (refer to **Section 18.2.2**) - SSW performs the following:

- configure the debug interface from BMI value
- enable the debug interface
- if start-up mode with Halt After Reset request (UMHAR) is selected in BMI upon master reset - enter an endless NOP (no operation) loop. From this point on, an external tool (debugger) can take over the control of the device, in particular:
  – if/when a debugger connects to device, it can halt the CPU and check BMI and/or the core program counter (PC) register
  – if HAR is selected, this can be identified by BMI value (refer to **Section 18.2.2**) and the PC value will point inside the ROM. The debugger can then manipulate PC and start the user code as desired - the default start address (taken without HAR) is according to the content at location $1000'1004_H$ in user Flash (refer to **Table 18-1**)

18.3 Data in Flash for SSW and User SW

Table 18-1 shows the data in Flash which is relevant for SSW execution and can be used by user software in XMC1100.

Table 18-1 Flash data for SSW and user SW in XMC1100

Address	Length	Function	Target location
		Start-up mode selection:	
$1000'0E00_H$	2 B	Boot Mode Index (BMI)	---
$1000'0E10_H$	2 B	Inverse BMI	---
		Chip Identification:	
$1000'0F00_H$	4 B	Chip ID	SCU_IDCHIP
$1000'0F04_H$	28 B	Chip Variant Identification Number	---
$1000'0FF0_H$	16 B	Unique Chip ID	---
		Temperature-sensor related data:	
$1000'0F20_H$	2 B	ANA_TSE_k1	---
$1000'0F22_H$	2 B	ANA_TSE_k3	---
$1000'0F24_H$	4 B	ANA_TSE_k2	---
$1000'0F28_H$	2 B	ANATSEMON_min	---
$1000'0F2A_H$	2 B	ANATSEMON_max	---
		DCO calibration data:	

Table 18-1 Flash data for SSW and user SW in XMC1100

Address	Length	Function	Target location
$1000'0F40_H$	2 B	DCO_ADJL_RT Frequency Low Adjustment value measured at room temperature (5 LSbits)	---
$1000'0F42_H$	2 B	DCO_ADJL_HT Frequency Low Adjustment value measured at high temperature (5 LSbits)	---
		Application software related data:	
$1000'1000_H$	4 B	Initial Stack Pointer value after SSW	SP_main
$1000'1004_H$	4 B	Start address after SSW in User modes	PC
		Clock system related data:	
$1000'1010_H$	4 B	Clock configuration value CLK_VAL1	If bit[31]=0: bits[19:0] into SCU_CLKCR[19:0]
$1000'1014_H$	4 B	Clock gating configuration value CLK_VAL2	If bit[31]=0: bits[10:0] into SCU_CGATCLR0[10:0]

19 Bootstrap Loaders (BSL) and User Routines

This chapter describes the Bootstrap Loaders (BSL) and the user-accessible routines in the ROM memory.

The ASC (UART) BSL (**Section 19.1**) and the SSC BSL (**Section 19.2**) is entered with the BMI settings as described in the Boot and Startup Chapter. The main purpose of BSL Mode is to allow easy and quick programming/erasing of the Flash.

Section 19.3 describes 5 user-accessible routines that can be called by application software. These routines are located in the ROM memory.

19.1 ASC (UART) Bootstrap Loader

ASC (UART) Standard Bootstrap Loader (ASC BSL) in XMC1100 uses USIC0 module to download code/data into SRAM starting at address 2000'0200H.

After the last code/data Byte is received and stored, the Startup Software starts user code from address 20000'0200H.

19.1.1 Pin usage

ASC BSL in XMC1100 is selected by a single BMI value but it allows to download user code/data via several variants of pins, modes and USIC0 channels:

- Channel 0
  - full duplex mode with RxD at P0.14 and TxD at P0.15
  - half duplex mode with RxD/TxD at P0.14
- Channel 1
  - full duplex mode with RxD at P1.3 and TxD at P1.2
  - half duplex mode with RxD/TxD at P1.3

Note: The above set of pins also accommodate all the assignments for SWD/SPD (debug) interfaces.

19.1.2 ASC BSL execution flow

The complete ASC Bootstrap Loader procedure consists of two parts as described below.

19.1.2.1 ASC BSL entry check sequence

Downloading is only initiated after a Start (zero) and a Header (described below) Bytes are received through one of the channels whereas both the channels are active ("listening") until the selection is done by the routine itself as follows:

- Channel selection - based on the fact at which RxD pin (see the above assignments) first Start+Header Bytes are received
- Full/half duplex selection - based on the Header Byte received

*Note: For definitions of header/acknowledge byte values exchanged with the host - refer to **Section 19.1.3**.*

The ASC bootloader functionality includes:

1. enable and configure the both USIC0 communication channels 0 and 1 as follows:
   a) ASC mode, 8 data bits, 1 stop bit, no parity
2. configure ASC clock-generation circuitry for measurement
3. perform continuously the following sequence for both USIC0 channels one after another
   a) check either Start zero Byte has been received
      if yes - then:
   b) reconfigure ASC clock-generating circuitry for normal operation according to the baudrate measured from the Start Byte, then:
   c) check either Header Byte has been received
      if yes - then:
   d) check the Header Byte received
      * if equal to BSL_ASC_F or BSL_ENC_F - full duplex mode is selected - continue with step 4.
      * if equal to BSL_ASC_H or BSL_ENC_H - half duplex mode is selected - continue with step 4.
      * otherwise - go to sequence for the other channel
4. select the current (Cy) channel for further handling; restore default configuration for the channel C(y-1) which is not selected
5. for the selected channel and mode (full/half duplex) :
   a) configure TxD pin and ASC transmitter
   b) adapt (in case) RxD pin settings
   c) enable transmission, single shot mode
6. check which Header byte was received:
   a) BSL_ASC_F or BSL_ASC_H - standard BSL entry sequence was executed, the communication will use further the baudrate detected - continue with **ASC BSL download sequence** (see **Section 19.1.2.2**)
   b) BSL_ENC_F or BSL_ENC_H - the communication during **ASC BSL download sequence** will potentially use different (e.g. higher) baudrate than the initially detected - continue with the **Enhanced ASC BSL entry sequence** (see below))

The above check sequence is executed

- in ASC BSL mode without time-out - until valid Start+Header Bytes are received
- otherwise - only for some time according to BMI.BSLTO

Note: If mode with time-out is selected but BMI.BSLTO=0 - User mode with start from Flash is directly taken.

Enhanced ASC BSL entry sequence

This is an extension of the standard **ASC BSL entry check sequence** and will be used when higher baudrates are required for the communication channel.

It includes the following steps:

1. Send BSL_ENC_ID to acknowledge the establishment of the initial baud rate and entry into enhanced ASC BSL
2. Send current PDIV divider from USIC0_CHy_BRG register - the 10-bit value is sent in 2 bytes (most significant byte first)
3. Receive from host the 10-bit value to be written into the STEP bit field in USIC0_CHy_FDR register
4. Configure USIC baud rate generator accordingly
5. For the selected channel and mode (full/half duplex):
   a) Configure TxD pin and ASC transmitter
   b) Adapt (in case) RxD pin settings
   c) Enable transmission, single shot mode
6. Send BSL_BR_OK to acknowledge the establishment of the new baud rate
7. Receive from host BSL_BR_OK to indicate that the communication channel has been successfully established
   - If no or wrong acknowledge value is received, a device software reset will be triggered

Figure 19-1 shows the outline of the sequence to configure the baud rate.

Requirements for Host

For the host to support enhanced ASC BSL, the host additionally has to:

- Receive from XMC1100 device, the current 10-bit PDIV value in USIC0_CHy_BRG register
- Calculate the MCLK that the device is running on based on the received PDIV through the formula:

(19.1)

$$MCLK = \text{Initial Baud Rate} \times (PDIV + 1) \times 8$$

- Select a new baud rate that is supported by the enhanced ASC BSL given the MCLK (see **Table 19-1**)
- Calculate the scaling factor, i.e. ratio of the new baud rate to the initial baud rate. Some examples are given in **Table 19-2**. The scaling factor is rounded to the nearest integer in order to meet a frequency deviation of +/- 3%.

Table 19-1 Supported Baud Rates

MCLK (MHz)	Standard baud rates supported with ASC BSL (kHz)		Max. baud rate supported with Enhanced ASC BSL (MHz)
	Min.	Max.	
8	1.2	28.8	0.999
16	2.4	57.6	1.998
32	4.8	115.2	3.996

Table 19-2 Scaling Factor Examples

Initial standard baud rates (kHz) ---A	Targeted higher baud rates (kHz) ---B	Scaling factor rounded to the nearest integer (B/A)
9.6	1500	156
19.2	1500	78
57.6	1500	26
115.2	1500	13

- Calculate the 10-bit value to be written into the STEP bit field of USIC0_CHy_FDR register through the formula:

(19.2)

$$\text{STEP (in fractional divider mode)} = \frac{1024 \times \text{Scaling Factor}}{\text{PDIV} + 1}$$

- Send the 10-bit STEP value to the device
- Wait until the BSL_BR_OK is received from the device
- Echo the BSL_BR_OK back to the device

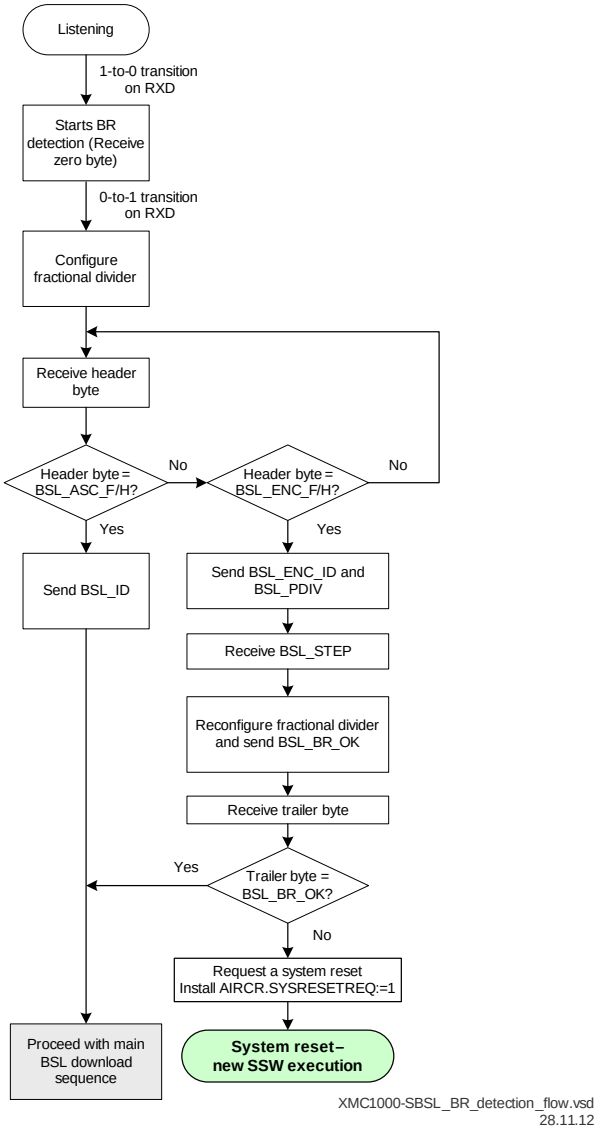


Figure 19-1 Baud Rate configuration sequence during XMC1100 ASC BSL entry

19.1.2.2 ASC BSL download sequence

After the baud rate has been detected/configured and channel/mode (full/half duplex) selected, ASC BSL awaits 4 bytes describing the length of the application from the host (refer to **Figure 19-2**). The least significant byte is received first.

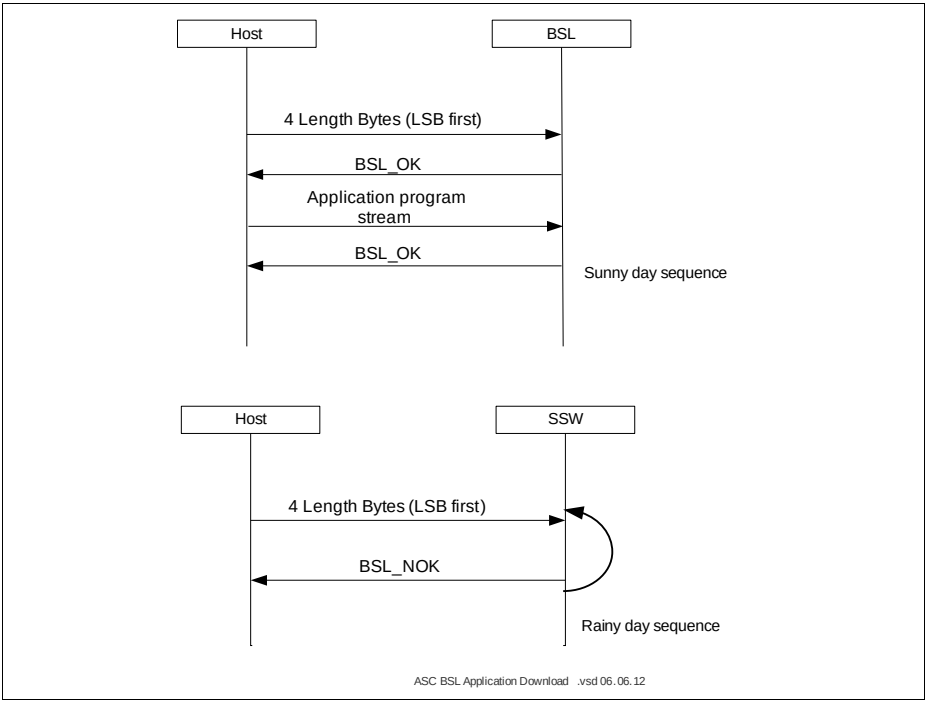


Figure 19-2 XMC1100 Standard ASC BSL: Application download protocol

If application length is found alright by SSW, a BSL_OK byte is sent to the host following which the latter sends the byte stream belonging to the application. After the byte stream has been received, SSW terminates the protocol by sending a final OK byte and then cedes control to the downloaded application.

If application length is found to be in error (application length greater than device SRAM size), a BSL_NOK byte is transmitted back to the host and the SSW resumes awaiting the length bytes.

19.1.3 ASC BSL protocol data definitions

The interaction between XMC1100 ASC Bootstrap Loader routine and the host implements handshake protocol based on **Figure 19-3** and request/acknowledge/data Bytes defined in **Table 19-3**.

Table 19-3 Handshake protocol data definitions in XMC1100 ASC BSL

Name	Length, Byte	Value	Description
Requests/data/acknowledge sent by the Host:			
BSL_ASC_F	1	6CH	Header requesting full duplex ASC mode with the current baud rate
BSL_ASC_H	1	12H	Header requesting half duplex ASC mode with the current baud rate
BSL_ENC_F	1	93_H	Header requesting full duplex ASC mode with a request to switch the baud rate
BSL_ENC_H	1	ED_H	Header requesting half duplex ASC mode with a request to switch the baud rate
BSL_STEP	2	$0XXX_H$	10-bit value (LSB aligned) to be programmed into selected USIC channel's FDR.STEP bit field. Most significant 6 bits should contain all 0.
BSL_BR_OK	1	$F0_H$	Final baud rate is established in enhanced ASC BSL
Acknowledges sent by XMC1100 firmware:			
BSL_ID	1	5DH	Start and header bytes are received, baud rate is established
BSL_ENC_ID	1	$A2_H$	Start and header byte(s) are received in enhanced ASC BSL, initial baud rate is established
BSL_PDIV	2	$0XXX_H$	10-bit value (LSB aligned) containing the selected USIC channel's BRG.PDIV bit field value. Most significant 6 bits should contain all 0.
BSL_BR_OK	1	$F0_H$	Final baud rate is established in enhanced ASC BSL.
BSL_OK	1	01H	Data received is OK
BSL_NOK	1	02H	Failure encountered during data reception

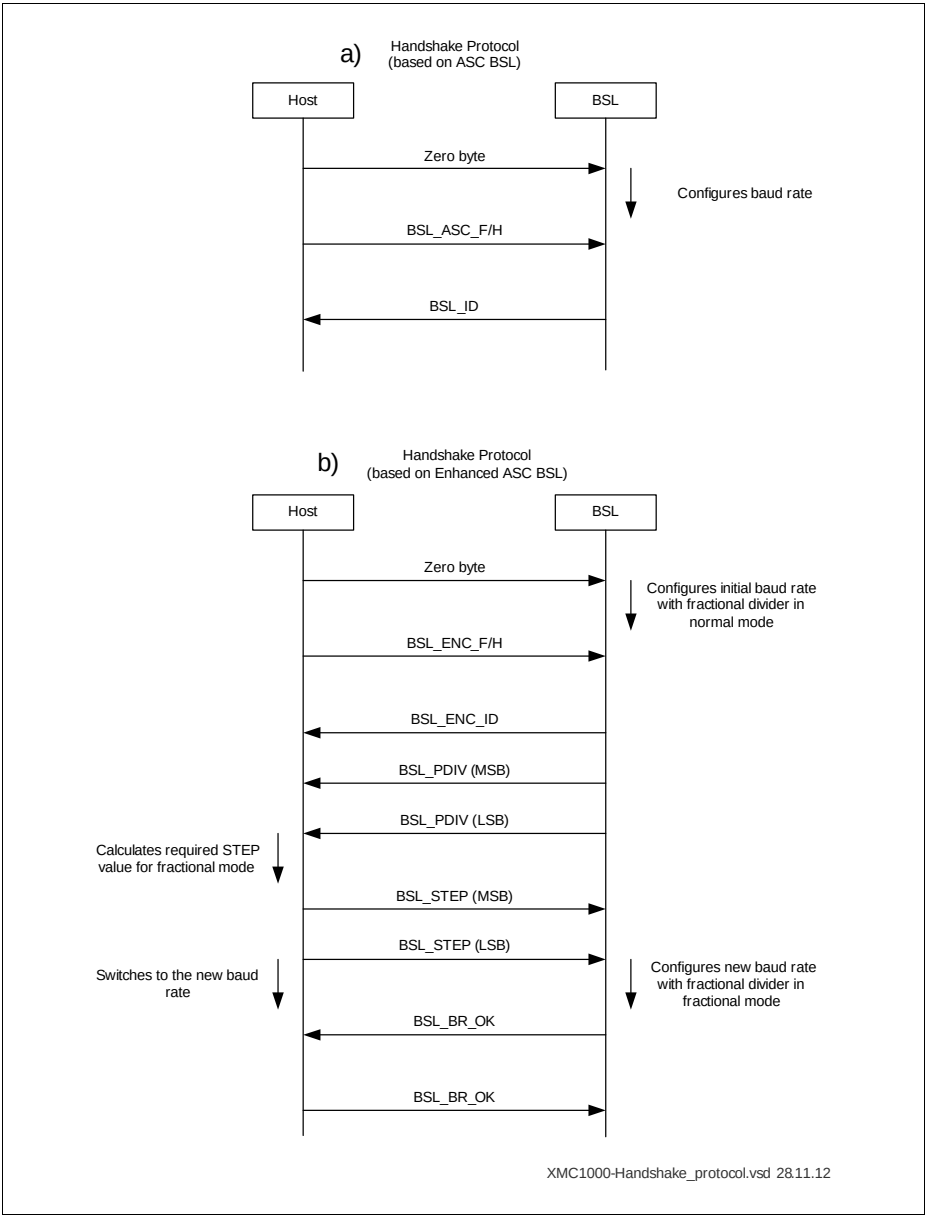


Figure 19-3 Handshake protocol for XMC1100 ASC BSL entry

19.2 SSC Bootstrap loader

This procedure downloads code from a SPI-compatible serial EEPROM into SRAM starting at address 2000'0200H using Channel 0 of USIC0 Module. The length of the code can be arbitrary up to the user-available SRAM size.

XMC1100 is the master SPI-device, following pins are used by the bootloader:

- MRST (master data input) and MTSR (master data output) - on the same P0.15, half-duplex mode used
- SCLK (clock signal) - P0.14, open drain
- SLS (chip-select CS to the EEPROM) - P0.13, open drain

Note: Due to the specific functionality of the bootloader - performing one Sequential Read operation only - there is no need of a dedicated SLS-control.

Note: Due to XMC1100 pin configured as open drain, external pull-up resistors need to be added on the bus

A SPI-compatible serial EEPROM of type 25xxx must be connected to the pins as above defined. The clock signal is configured at MCLK/10 frequency upon device startup.

The SSC bootloader in XMC1100 is able to communicate with a very broad range on serial-EEPROM types: from such using 8-bit addressing (size up to 2K bit, quite outdated already) up to devices using 24-bit addressing (1M bit and above). The connected EEPROM type is determined by examining the received header bytes, as indicated in **Table 19-4**.

Note: Besides the bootloader supports all the device-types, the code/data length which can be downloaded is limited by the size of available SRAM.

Table 19-4 SSC BL: Determining the EEPROM Type and data-flow

SSC Frame		EEPROM with 8-bit addressing connected		EEPROM with 16-bit addressing connected		EEPROM with 24-bit addressing connected	
N	data	P11-send	P11-receive	P11-send	P11-receive	P11-send	P11-receive
1	03_H	Read command	XX_H default level	Read command	XX_H default level	Read command	XX_H default level
2	00_H	Address	XX_H	Address_H	XX_H	Address_H	XX_H
3	00_H	dummy	Identification	Address_L	XX_H	Address_M	XX_H

Table 19-4 SSC BL: Determining the EEPROM Type and data-flow

SSC Frame		EEPROM with 8-bit addressing connected		EEPROM with 16-bit addressing connected		EEPROM with 24-bit addressing connected	
4	00_H	dummy	Size	dummy	Identification	Address_L	XX_H
5	00_H	dummy	Data Byte 1	dummy	Size, High B	dummy	Identification
6	00_H	dummy	Data Byte 2	dummy	Size, Low B	dummy	Size, High B
7	00_H	dummy	Data Byte 3	dummy	Data Byte 1	dummy	Size, Mid B
8	00_H	dummy	Data Byte 4	dummy	Data Byte 2	dummy	Size, Low B
9 …	…	dummy	Data Byte 5 ...n	dummy	Data Byte 3 ...n	dummy	Data Byte 1 ...n

The EEPROM connected for downloading must contain:

- identification Byte D5H at the first address (0000H)
- the length of the code (Code_Length) in Bytes as follows:
  - in case of an EEPROM with 8-bit addressing - in one Byte at address 0001H
  - in case of an EEPROM with 16-bit addressing - in two Bytes at addresses 0001H/0002H ordered high/low
  - in case of an EEPROM with 24-bit addressing - in three Bytes at addresses 0001H/0002H/0003H ordered high/middle/low
- the code of defined length follows sequentially

The SSC bootloader functionality includes:

1. enable the communication channel by setting USIC0_C0_KSCFG.MODEN:=1
2. assure clock gating for USIC0 module is de-asserted
3. configure the USIC0_C0 channel as follows:
   a) SSC mode, 8 data bits, MSB first
   b) SCLK frequency = MCLK/10
   c) no SLS activated
   d) pin-configuration as defined above
   e) enable transmission
4. de-select EEPROM by setting CS=1
5. enable EEPROM (CS=0) and send Read command (03h)
6. send two zero Bytes to request data from address 0000h
7. Upon any next Byte transfer as long as the EEPROM is still selected (CS=0) - data from then next following address will be read out.
8. the last Byte received is checked:

a) if equal to the Identification Byte (D5H) - the EEPROM uses 8-bit addressing:
-SSW reads one more Byte and saves it as Code_Length - 1B only effective
-continue with p.10

9. read one more Byte:
   a) if equal to the Identification Byte (D5H) - the EEPROM uses 16-bit addressing:
   -SSW reads two Bytes in order High/Low to determine Code_length (2 Bytes)
   -continue with p.10
10. read one more Byte:
   a) if equal to the Identification Byte (D5H) - the EEPROM uses 24-bit addressing:
   -SSW reads three more Bytes in order High/Middle/Low to determine Code_length (3 Bytes)
   -continue with p.10
   b) if Not - directly exit the sequence
11. check the value of Code_length:
   a) if 0 or greater than allowed in XMC1100- directly exit the sequence
12. read [Code_Length] Bytes and store them sequentially from the beginning of SRAM (address 2000'0200H) on
13. disable EEPROM (CS=1)
14. disable the communication channel by resetting USIC0_C0_KSCFG.MODEN:=0
15. set SSW flag BSL_act=1 and exit the sequence

After the last Byte is received and stored, the Startup Software starts the code downloaded from address 20000'0200H.

19.3 Firmware routines available for the user

Several user routines (refer to **Table 19-5**) are available inside ROM so they can be called by application software.

Table 19-5 User routines' in XMC1100 ROM

XMC1100 ROM		Description
Address	**Content**	
0000'0100H	_NvmErase	Pointer to **Erase Flash Page** routine
0000'0104H	_NvmProgVerify	Pointer to **Erase, Program & Verify Flash Page** routine
0000'0108H	_BmiInstallationReq	Pointer to **Request BMI installation** routine
0000'010CH	_CalcTemperature	Pointer to **Calculate chip temperature** routine
0000'0120H	_CalcTSEVAR	Pointer to **Calculate target level for temperature comparison** routine

NVM-related functions (see **Section 19.3.1** and **Section 19.3.2**) return status indication as shown in **Table 19-6**.

Table 19-6 Status indicators returned by NVM routines in XMC1100 ROM

Status Indicator		Description
Symbolic name	**Value**	
NVM_PASS	0001’0000H	Function succeeded
NVM_E_FAIL	8001’0001H	Generic error
NVM_E_SRC_ AREA_EXCEEDED	8001’0003H	Source data is not in RAM
NVM_E_SRC_ ALIGNMENT	8001’0004H	Source data is not 4 Byte aligned
NVM_E_DST_ AREA_EXCEEDED	8001’0005H	Destination data is not (completely) located in NVM
NVM_E_DST_ ALIGNMENT	8001’0006H	Destination data is not properly aligned
NVM_E_NVM_FAIL	8001’0009H	NVM module can not be physically accessed
NVM_E_VERIFY	8001’0010H	Verification of the written page not successful

The description below provides an overview of the features.

19.3.1 Erase Flash Page

XMC1100 Flash can be erased with granularity of one page = 16 blocks of 16 Bytes = 256 Bytes using this routine.

- Input parameter
  - logical address of the Flash Page to be erased, must be page aligned and in NVM address range (pageAddr)
- Return status (refer to **Table 19-6**)
  - OK (NVM_PASS)
  - invalid address (NVM_E_DST_AREA_EXCEEDED, NVM_E_DST_ALIGNMENT)
  - operation failed (NVM_E_FAIL, NVM_E_NVM_FAIL, NVM_E_VERIFY)
- Prototype
  NVM_STATUS XMC1000_NvmErasePage (unsigned long pageAddr)

19.3.2 Erase, Program & Verify Flash Page

This function performs erase (skipped if not necessary), program and verify of selected Flash page.

- Input parameters
  – logical address of the target Flash Page (dstAddr)
  – address in SRAM where the data starts (srcAddr)
- Return status (refer to **Table 19-6**)
  – OK (NVM_PASS)
  – invalid addresses (NVM_E_SRC_AREA_EXCEEDED, NVM_E_SRC_ALIGNMENT, NVM_E_DST_AREA_EXCEEDED, NVM_E_DST_ALIGNMENT)
  – operation failed (NVM_E_FAIL, NVM_E_NVM_FAIL, NVM_E_VERIFY)
- Prototype
  NVM_STATUS XMC1000_NvmProgVerify (unsigned long srcAddr, unsigned long dstAddr)

19.3.3 Request BMI installation

This procedure initiates installation of a new BMI value. In particular, it can be used as well as to restore the state upon delivery for a device already in User Productive mode.

- Input parameter
  – BMI value to be installed (requestedBmiValue)
- Return status - only upon error, if OK the procedure triggers a reset respectively does not return to calling routine
  – wrong input BMI value (0001H)
- Prototype
  unsigned long XMC1000_BmiInstallationReq (unsigned short requestedBmiValue)

19.3.4 Calculate chip temperature

This procedure calculates the current chip temperature as measured by the XMC1100 built-in sensor, based on data from Flash including trimming values and pre-calculated constants (refer to **Table 19-7**) and data from the actual measurement (read from Temperature Sensor Counter2 Monitor Register ANATSEMON).

- Input parameter
  – none
- Return status
  – chip temperature in degree Kelvin
- Prototype
  unsigned long XMC1000_CalcTemperature (void)

19.3.5 Calculate target level for temperature comparison

This procedure - a kind of reverse of **Calculate chip temperature** - calculates the value which must be installed in ANATSEVAR register to get indication in ANATSESTATUS.TSE_CP_VAR_STAT bit when the chip temperature is above/below some target/threshold.

- Input parameter
  – threshold temperature in degree Kelvin - allowed range 223...423 (temperature)
- Return status
  – equivalent sensor threshold value for the temperature provided as input parameter
- Prototype
  unsigned long XMC1000_CalcTSEVAR (unsigned long temperature)

19.4 Data in Flash used by the User Routines

Table 19-7 shows the data in Flash which is used by the user routines in XMC1100.

Table 19-7 Basic Flash data for SSW and user SW in XMC1100

Address	Length	Function	Target location
Start-up mode selection:			
$1000'0E00_H$	2 B	Boot Mode Index (BMI)	---
$1000'0E10_H$	2 B	Inverse BMI	---
Temperature-sensor related data:			
$1000'0F20_H$	2 B	ANA_TSE_k1	---
$1000'0F22_H$	2 B	ANA_TSE_k3	---
$1000'0F24_H$	4 B	ANA_TSE_k2	---
$1000'0F28_H$	2 B	ANATSEMON_min	---
$1000'0F2A_H$	2 B	ANATSEMON_max	---

Debug System

20 Debug System (DBG)

The debug system is an extension to the regular processor architecture. The XMC1100 Series Microcontrollers provide a complete hardware debug solution, with hardware breakpoint and watchpoint options. This allows high system visibility of the processor, memory and available peripherals. The debug functions are implemented with a standard ARM Cortex M0 configuration. Debug functions are integrated into the ARM Cortex-M0 processor architecture. The debug system supports serial wire debug and single pin debug interfacing.

References

[9] Cortex-M0 Technical Reference Manual

[10] Cortex-M0 Integration and Implementation Manual

[11] Cortex-M0 User Guide

[12] ARMv6-M Architecture Reference Manual

[13] ARM Debug Interface V5

[14] CoreSight Components Technical Reference Manual

[15] CoreSight DAP-Lite

20.1 Overview

The basic debug functionality of the Cortex-M0 debug system is limited to invasive debug and includes processor halt, single step, processor core register access, Reset and HardFault Vector Catch. Besides the hardware debug components unlimited software breakpoints are available. The Debugger can connect to the debug system via the Serial Wire Debug (SWD) or Single Pin Debug (SPD) interface port and uses CoreSight infrastructure and the regular access flow. This permits debugger to identify the processor and its debug capabilities. The debug system allows a full system memory access and access to zero-wait state system slaves via the internal debug access port (DAP) connected to the system bus matrix. This access is non-intrusive and a debugger can access the devices, including memory when the processor is halted or running. Core register full access is available, if the processor is halted. The System Control Space (SCS) provides debug through registers available. The Data Watchpoint Unit (DWT) provides two watchpoint register sets, each implement data address and PC based watchpoint functionality including comparator address masking. The processor BrakPoint Unit (BPU) provides four PC based breakpoint register. The system is accessible by the tool through DAP. DAP permits access to debug resources when the processor is running, halted, or held in reset. A processor enters Debug state if it is

configured to halt on a debug event, and a debug event occurs. The supported debug infrastructure can be identified by checking the ROM table.

20.1.1 Features

The accurate Debug and Trace System provides the following functionality:

- Serial Wire Debug Port (SWD) provides Serial Wire Debug, which allows to debug via 2 pins.
- Single Pin Debug (SPD) provides a debug capability via 1 pin.
- Processor halt, single-step, processor core register access
- Reset and HaldFault Vector Catch.
- Software breakpoints
- Full system memory access
- 4 Hardware breakpoints supported
- 2 watchpoints supported

*Note: Please refer to **ARMv6-M Architecture Reference Manual** for more detailed informations on the debug functionality.*

20.1.2 Block Diagram

The Debug system block diagram is shown in **Figure 20-1**.

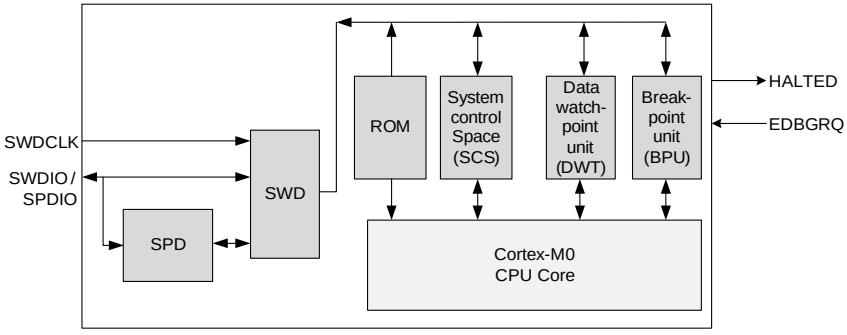


Figure 20-1 Debug and Trace System block diagram

20.2 Debug System Operation

The Debug System provides general debug options. Debug options are based on break points and CPU halt. Debug resources available are Data Watchpoint and Trace, Brakpoint unit, ROM table and the SCS system control block and debug control block. Debug capabilities can be accessed by a debug tool via Serial Wire Debug interface

(SWD) or Single Pin Debug (SPD). The selection of the access protocol (SWD or SPD) is done by BMI mode setting.

20.2.1 System Control Space (SCS)

The SCS is the area where the debugger can have direct access to the memory mapped register debug register. The debug resources together with the debug register in the SCS are accessible through the DAP interface. Accessible resources are for example system control and ID registers including system control block. Additionally the system timer and the interrupt controller (NVIC) can be accessed via the SCS.

20.2.2 Data Watchpoint and Trace (DWT)

The DWT unit provides an external Program Counter (PC) sampling capability based on PC sample register and comparators, that support watchpoints for address matching and PC watchpoints for instruction address matching. The PC sampling feature and watchpoint support operate independently of each other and a watchpoint event is asynchronous to the instruction that caused it. The XMC1100 supports two watchpoints, each supporting compare, mask and function registers. Watchpoint events result in a processor halt and enters the processor system into debug state. Data address matching results in a creation of a watchpoint event. Instruction address matching in a creation of a PC watchpoint event. The DWT register are accessible through the DAP interface.

A DWT program counter sample register permits a debugger to periodically sample the PC without halting the processor and allows coarse grained profiling. The PC sampling feature and the watchpoint support operate independently of each other. The register is defined so that a debugger can access it without changing the behavior of any code currently executing on the device.

The recommended mechanism for generating a breakpoint on a single instruction address is to use the BPU. The DWT based mechanism must be used to generate a PC matching event on a range of addresses. The debug return address value for a watchpoint event must be that of an instruction to be executed after the instruction responsible for generating the watchpoint.

20.2.3 Break Point Unit (BPU)

The Breakpoint (BKPT) instruction provides software breakpoints. It can cause a running system to halt depending on the debug configuration. A BKPT is a synchronous debug event, caused by execution of a BKPT instruction or by a match in the BPU. A BKPT causes the processor to enter debug state. The Debug tool can use this situation to investigate the system state when the instruction at a particular address is reached. The BPU provides support for breakpoint functionality on instruction fetches, based on instruction address comparator. The M0 provides four breakpoint comparator registers. Each breakpoint comparator register includes its own enable bit. The comparators match

instruction fetches from the Code memory region and operate only on instruction read accesses. The comparators do not match data read or data write access. Address matching is available on the upper half word, lower half word or both half words. Potential reasons for a debug event are a debug halt, a BKPT, a DWT trap and Vector CATCH.

20.2.4 ROM Table

To identify the Cortex-M0 processor with the respective Debug System components the debugger locates and identifies the ROM table. ROM table Identification values are predefined and contain pointers to the SCS, BPU and DWT. Each of the debug modules requires it's own ROM table and indicate whether the block is present. The ROM table can be accessed through the DAP interface.

20.2.5 Debug tool interface access - SWD

The tool access based on SWD must use a connection sequence, to ensure that hot-plugging the serial connection does not result in unintentional transfer. The connection sequence ensures that the SWD is synchronized correctly to the header. The sequence consists of a sequence of 50 clock cycles data = 1s. This connection sequences is also used as a line reset sequence. The protocol requires that any run of 50 consecutive 1s on the data input is detected as line reset, regardless of the state of the protocol. After the line reset the debugger reads IDCODE register, which gives confirmation that correct packet frame alignment is has been achieved.

20.2.5.1 SWD based transfers

The SWD interface requires to continue the clock for a number of cycles after the data phase of any data transfer. This ensures the transfer is completely clocked through the SWD. The transfer completion can be achieved by a immediate start of a new SWD operation, continuos clocking of SWD interface until the host starts a new SWD operation or after the data parity bit the clock is continued for at least 8 more clock rising edges, before stopping the clock.

Each sequence of operation on the serial wire consists of two or three phases and is defined from debugger point of view. The package request and acknowledge response phases are always there. The data transfer phase is only present when either data read or data write request is followed by a valid acknowledge response or the Overrun Detect flag is set.

The SWD protocol applies a simple parity check to all packet request and data transfer phases. On a packet request the parity check is made over the four bit header. On a data transfer the parity check is made over the complete 32 data bits. The parity is added directly after the packet request and after the data transfers. The parity bits are not included in the parity calculation.

Serial Wire Debug protocol operation

The SWD protocol supports the following operations:

- **Write operation** including OK response (3-Phases: 8-bit write packet request, 3-bit OK ACK, 33-bit data write including1-bit parity at the end)
- **Read operation** including OK response (3-Phases: 8-bit read packet request, 3-bit OK ACK, 33-bit data read including a 1-bit parity at the end).
- **WAIT response** to read or write operation request (2 Phases: 8-bit read or write packet request, 3-bit WAIT ACK response)
- **FAULT response** Read or Write operation (2-Phases: 8-bit read or write packet request, 3bit FAULT ACK response)
- **Protocol error** sequence occurs when target failed to return a ACK response (1 Phase: 8-bit Packet request - line not driven by target for 32 bit cycles)

Note: A single-cycle turnaround period between the operation phases is required, for the transfer direction change only. If there is no transfer direction change, no turnaround period is introduced. If the protocol ends with a data transfer towards the debugger a turnaround period is introduced. After the single-cycle turnaround time the protocol must be proceeded.

20.2.5.2 SWD based errors

On the SWD interface protocol errors can occur. These errors might be detected by the parity checks on the data. A message header error can be detected by a parity error. A debugger not receiving a response or a port does not respond to the message, the debugger must back off. During the back off mechanism the debugger does read the IDCODE register and ensures the Debug Port is responsive and then retries the original access.

Errors can be returned by the DAP itself or might come from a debug resource. When an error occurs the error bit remains sticky until cleared. A debugger must check the error status after performing a series of transaction to be aware of an error.

Error conditions within the SWD interface are recorded using sticky flags. Potential errors are read and write errors, overrun detection, protocol errors. When the sticky bit is set to 1 it remains set until it is explicitly cleared to 0. When an error is flagged the current transaction is completed and any additional access port access transaction is discarded until the sticky flag is cleared to 0. A debugger must check periodically the Control/Status register after performing a series of transactions to detect the error.

A read and write error might occur within the DAP or come from the resource being accessed. Additionally a read or write error might be generated if the debugger makes an access port transaction request while the debugger power domain is powered down.

An overrun error might be detected on a sequence of transactions. Therefore the debugger must check after each sequence. The Overrun detection mode can be left by clearing the STICKYORUN and ORUNDETECT.

On the Serial Wire Interface protocol errors can occur, for example because of wire-level errors. These errors might be detected by the parity checks of data. If the SWD interface detects a parity error in a message header of the debug message the error is reflected. If the interface does not receive a response to a message, the debugger must back off. It must then requires a read of the IDCODE register, to ensure the debug port is responsive, before retrying the original access, again. If the SWD detects a parity error in the data phase of a write transaction, it sets the sticky write data error flag (WDATAERR) in the control and status register. Subsequent accesses from the debugger, other than IDCODE, CTRL/STAT or ABORT, result in a FAULT response.

20.2.6 Debug tool interface access - Single Pin Debug (SPD)

The SPD protocol based tool interface access allows to debug the system using a single pin only. The bit frequency is 2 MHz and allows an effective SWD telegram of 1.4 Mbits/s. The protocol is very robust against clock deviations between tool and device.

The SPD protocol encodes the SWD protocol bits with the distance between the SPD signal edges. A SWD value of ‘0’ is encoded with a short distance of 0.5 µs, a value of ‘1’ with a distance of 1 µs. This encoding is used in both transfer directions. Initially the SPD signal level always start with a positive SPD signal pulse with 1 µs width, encoded SPD start bit, which is not part of the SWD protocol. The transmission of the rest of the telegram will be independent of the edge direction. As the protocol starts with a logic one start bit detection it is recommended to finish the protocol with a logic 0 signaling level. If the SPD signal level is high after the last bit of the telegram, that tool to add at ‘0’ bit with a negative edge after 0.5 µs. This can be achieved by the tool transferring an odd number of bits.

SWD has a clean request response protocol, where the tool is always the requestor and the device executes and sends a response. So the SWD module will change the direction as defined by the SWD protocol operations (**Section 20.2.5.1**).

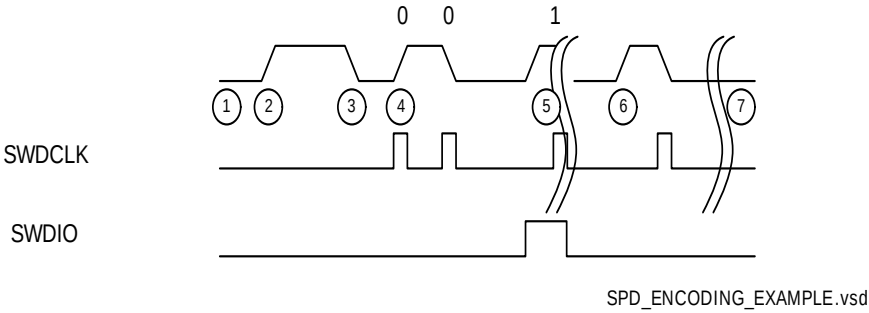


Figure 20-2 SPD Encoding Example

As a basic example a simple transfer is described in Figure before. Initially (1) the SPD module is in IDLE state. During IDLE state the SPD module does not generate a clock to the SWD module. At point (2) the SPD detects a rising edge and moves to a receive state. The first rising edge of the protocol is a start bit for the SPD module only and is not transferred to the SWD module. The following bits are the SWD header information and transferred to the SWD module. At point (5) a logic 1 shown and at (6) a logic zero. At the time the protocol direction is changed (which means for the implementation to go through IDLE state) the clock to SWD is switched off as shown in state (7). The SPD module requires to end on a Low signal value therefore it is required to have an odd number of bits moved into the interface.

Protocol transfer requirements between SWD and SPD module

The data transfer to and from SWD protocol is single bit based on the SWD clock. The SWD clock is generated by the SPD module and directly derived from the transferred bit. In RECEIVE direction the SPD module transfers the data to the SWD module bit wise, with one clock cycle per bit received.

The SPD module switches off SWD clock when going through module IDLE.

The Debugger adds a SPD start bit in front of the SWD telegram when it writes data to the device (form SPD point of view, RECEIVE). The SPD RECEIVE start bit is not transferred to the SWD module. The RECEIVE start bit is in front of the original SWD protocol on every new RECEIVE start, which is in front of the SWD header and again in front of debugger write data. Additionally the SPD module requires in RECEIVE direction a protocol to end with a logic zero signal level, which is achieved by an odd number of bits transferred by the tool. Based on the SWD protocol this is already achieved in the header phase by adding the described SPD start bit. During acknowledge phase as this phase does have 3 bits only. During the protocol data phase the Debugger has to add one more zero bit. (SPD Start bit, 32-bit data, parity bit, SPD odd bit as logic Zero).

As the SWD protocol requires in RECEIVE direction at least 8 clocks at the end of the data transferred to finish the protocol operation the Debugger has to add 9 logic zeros to the protocol (8 for this requirement and 1 to have an odd number).

SPD protocol based tool Interaction

All SPD communication is initiated by the tool. The tool will send a SWD header encoded to a SPD telegram (header) and then switches the SPD signal direction to input and wait for the acknowledge from the device.

SPD Receive:

Based on a low level the tool starts the protocol with SPD start bit. The start bit is indicated with a rising edge and a signal duration of 1µs. The following bits are based on the SWD protocol sequence.

The start sequence: SPD start bit + SWD HEADER + 0 Bit + 0 Bit (the two additional 0 bits at the end are required to achieve a low level at the signaling line.).

After receiving the header, the SWD will output his acknowledge response. Thus, SPD will change from RECEIVE to IDLE to SEND and ACKto IDLE again.

Leaving IDLE to RECEIVE state to transfer write data the start bit from tool is also required.

The SPD start bit must be a logic one with the defined time duration of a logic one. Is the signal duration longer it turns into a timeout situation. Is the signal duration shorter the start bit is not recognized.

The debug tool not sending data continuously in SPD RECEIVE direction, generates a Timeout in SPD module, which brings it to IDLE mode again. Starting from IDLE state the tool is required to provide SPD start bit for a new communication. A timeout is recognized after 1.4 µs.

A permanent write (SPD RECEIVE) allows to proceed the protocol without SPD start bit. In this case the SPD interface remains in RECEIVE state and only the SWD receive protocol has to be performed. In this case the SWD header has to start with a logic 1 as defined by the SWD protocol.

SPD SEND:

The tool is required to end a RECEIVE at low level. Based on the low level the SEND start is indicated by a rising edge. SEND has to end with a signal low level, too.

IDLE:

From RECEIVE to IDLE - driven by SPD or by timeout.

From SEND to IDLE - driven by SWD module caused by direction change .

Time to switch from RECEIVE to SEND is between 375 and 500 ns and always goes through IDLE phase. The tool must change from write (SPD RECEIVE) to read (SPD SEND) direction within 375 ns. SEND always starts with a rising edge.

20.2.7 Debug accesses and Flash protection

The XMC1100 Flash implements read protection for sector 0 only. All other sectors can be read. Additionally a user configuration erase and write protection is available, which allows to protect Flash content from unintended writes or erase. Special care is taken, that the debugger can't bypass this protection.

Because of this, per default and after a system reset the debug interface is disabled. The Boot Mode Index (BMI) determines if the debug mode can be entered and the debug interface enabled at the end of the SSW.

Before the BMI is configured to user productive mode a change of BMI to arbitrary value is supported, afterwards BMI can only be restored to its default value, which does not enable debug access.

Besides the BMI configuration the Debug system supports a protection mechanism which prevents a debug access to the system address area during the time startup software (SSW) is running. Fimware controls the debug access, based on register setting.

Software based Breakpoints are supported, but prevented during SSW.

20.2.8 Halt after reset

There are two possibilities to perform a halt after reset. The first possibility is Infineon specific an allows to perform a CPU halt after "power on"/"master" reset (HAR) at the very first instruction of the Application code. The HAR activation is based on BMI settings and SSW executing endless loop, allowing a debugger to take control over the device. A defined configuration sequence driven by the connected debug tool is required in this case. The second possibility is the regular ARM flow halt after reset, which is based on breakpoints and a system reset. This halt after reset is also named "Warm Reset". In both cases the ARM system can not be accessed for security reasons from the debug port during the time SSW is running, as the physical pins are not available during that time. Based on the BMI setting the debug capability can be enabled at the end of the SSW by firmware. The default BMI configuration does not enable debug access.

20.2.8.1 HAR

The BMI can be configured to allow a HAR (UMHAR) at the end of the SSW, which always requires a power on reset to be executed. A new BMI configuration to BMI.UMHAR (User Mode with debug enabled and HAR) requires to perform a master reset to boot in UMHAR mode.

At a HAR (Cold Reset situation), the system comes from a PORST (or Master Reset) . To achieve a HAR, the tool has to register (CDBGPWRUPREQ) and enable (DHSCR.C_DEBUGEN) the debug system. Additionally it has to halt (DHCSR.C_HALT) the CPU and manipulate the PC (Program Counter) to point to the start address of the user code. The debugger registration and debug system enable is possible at the end of

the firmware, where firmware is waiting for a tool registration. During SSW execution the debugger as no access capability and cannot set the enabling bits CDBGPWRUPREQ, C_DEBUGEN or halt the CPU. The address of the user code start address can be read from address 0x10001004 in user Flash. It is recommended to check the Boot Mode Index BMI.HWCFG bit for UMHAR before manipulating the PC.

The following Figure (**Figure** HAR - Halt After Reset) shows the software flow based on the modules participating to the HAR.

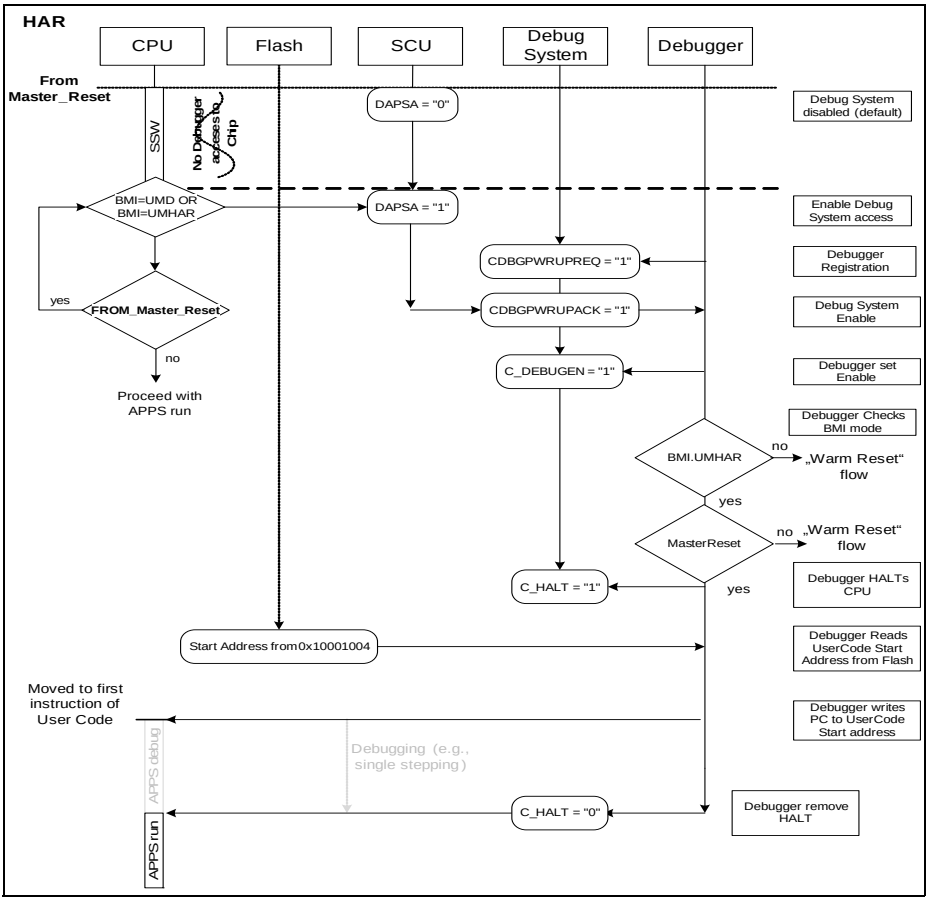


Figure 20-3 HAR - Halt After Reset Flow

20.2.8.2 Warm Reset

A halt after system reset (Warm Reset) can be achieved by programming a break point at the first instruction of the application code. Additionally the CDBGPWRUPREQ (tool registration) and the C_DEBUGEN has to be set by the debugger. After a system reset, the HAR situation is not considered, as the reset is not coming from PORST (Master_Reset).

Note: The CDBGPWRUPREQ and C_DEBUGEN does not have to be set after a system reset, if they have already been set before and the debugger remains registered. The bits are not affected by the system reset.

A tool hot plug situation allows to debug the system by enabling the CDBGPWRUPREQ and the C_DEBUGEN register. It is recommended to check the Boot Mode Index BMI.HWCFG. The Warm Reset flow can be done in both BMI modes, UMD (User mode with debug enabled) and UMHAR. In both cases the “Warm Reset” flow is based on breakpoint setting and system reset afterwards.

In general after a tool hot plug break points can be set or the CPU can directly be HALTED and stops at the actual program stage. To stop at the very first instruction of the user code the tool has to fire a system reset after setting the breakpoint there.

The following Figure (**Figure** Hot Plug and Warm Reset) illustrates the debug tool HOT PLUG situation or the Halt after Warm Reset (system reset) and how to proceed to come to a Halt situation.

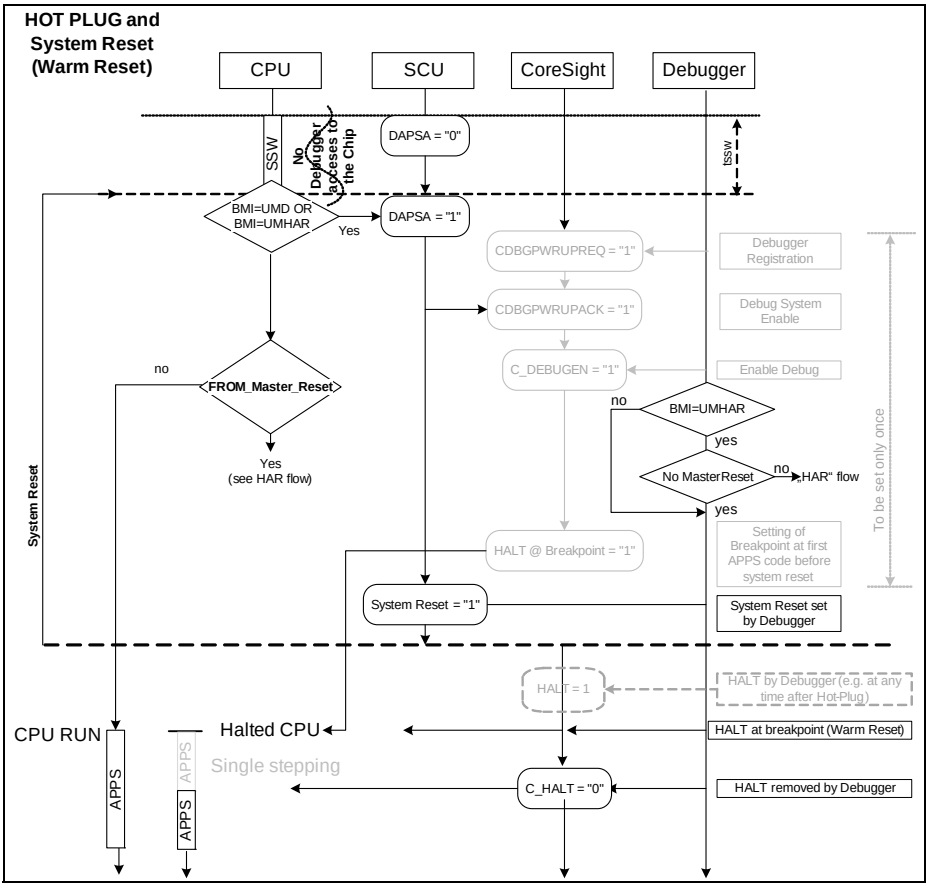


Figure 20-4 HOT PLUG or Warm Reset Flow

20.2.9 Halting Debug and Peripheral Suspend

The XMC1100 device supports a suspend capability for peripherals, if the program execution of the CPU is stopped by the debugger, e.g. with a breakpoint,or with the C_HALT. This allows to debug critical states of the whole microcontroller. Whether the suspend function is supported or not has to be configured locally at the peripheral.

In some cases it is important to keep certain peripherals running, e.g. a PWM or a CAN node, to avoid system errors or even critical damage to the application. Because of this, the peripheral allows to configure how it behaves, when the CPU enters halt debug

mode. Per default a peripheral is not sensitive on a suspend request. Sensitivity can be configured based on the system use case.

It can be decided at the peripheral to support a Hard Suspend or a Soft Suspend. At a Hard Suspend situation the clock at the peripheral is switched off immediately, without waiting on acknowledge from the module. At a soft suspend the peripheral can decide when to suspend. Usually at the end of the actual active transfer.

A Watchdog timer is only running when the suspend bus is not active. This is particularly useful as it can't be serviced by a halted CPU. A configuration option is available, which allows to enable the Watchdog timer also during suspend. This allows to debug Watchdog behavior, if a debugger is connected.

The user has to ensure, that always only those peripherals are sensitive to suspend, which are intended to be. To address this, each peripheral supporting suspend does have an enable register which allows to enable the suspend feature. The following table (**Table 20-1**) shows the peripherals, supporting or not supporting peripheral suspend or detailed information on the peripheral suspend behavior during soft suspend can be found at the respective peripheral chapter.

Table 20-1 Peripheral Suspend support

Peripheral	supported	default mode	Hard Suspend	Soft Suspend
RTC	yes	not active	yes	no
USIC	yes	not active	no	yes
CCU4	yes	not active	yes	yes
WDT	yes	active	yes	no
VADC	yes	not active	yes	yes
PRNG	no	- - -	- - -	- - -

Note: Enable debug suspend function in the user initialization code after every reset. In addition the user has to consider debug suspend register at peripherals can only be configured after the clock of the module is enabled via CGATCLR0 register.

Important tool provider note:

The peripheral suspend logic is connected to the system reset. A system reset activation results in a peripheral suspend configuration loss. Therefore the suspend configuration at the peripherals "outlasting" a system reset, requires the tool to reconfigure the suspend configuration after system reset. This is achieved by shadowing the user peripheral suspend configuration in the tool and set a HW breakpoint at the first instruction of use code, if the suspend function is activated. After the CPU halts at that breakpoint the tool has to reconfigure the suspend configuration at the peripheral, as it has been configured before the system reset.

Note: Suspend activation results in loosing one HW breakpoint, as it is used by the tool to handle the suspend reconfiguration after system reset.

A debug tool should offer the peripheral suspend reconfiguration after system reset with an option to proceed Usercode without an user interaction requirement after tool reconfiguration or remain stopped at first line of user code and wait for user action.

(The user configures the desired suspend behavior only once and the tool stores the configuration to have it present for a automatic reconfiguration after a system reset.)

20.2.10 Debug System based processor wake-up

The debugger can wake-up a processor in sleep mode (sleep and deep sleep mode). The wake-up is based on a new pending interrupt event from the debug system, which is a HALT. The interrupt event does also wake-up the processor, if the interrupt is disabled or has insufficient priority to cause exception entry. The interrupt wake-up mode is derived by the NVIC (Programmable Multiple Priority Interrupt System) available in the processor system.The debugger is able to register, enable the Debug System and HALT the CPU also in sleep and deep sleep mode.

Note: The wake-up from deep-sleep mode via debugger halt is only supported using SWD interface. A wake-up from deep-sleep based on SPD interface is not supported.

20.2.11 Debug Access Server (DAS)

The DAS API provides an abstraction of the physical device interface for tool access. The key paradigm of DAS is to read or write data in one or several address spaces of the target device.

DAS Features

- Standard interface for all types of tools
- Efficient and robust methods for data transfer
- Several independent tools can share the same physical interface
- Infineon's DAS miniWiggler support SWD and SPD.

Note: DAS is not XMC1100 specific. It can be used for all Infineon 8-, 16, and 32-bit microcontrollers with DAP, SWD, SPD or JTAG interface. For more information refer to www.infineon.com/DAS.

20.2.12 Debug Signals

XMC1100 MC product family provides debug capability using ARM M0 Debug port SWD or the Infineon proprietary SPD. The SWD Port has 2 interface signals (Clock + Bidirectional data). The SPD port has one interface signal (SPDIO - bidirectional data) with an overlay of SWD interface SWDIO pin.

Table 20-2 SWD toplevel IO signal

Signal	Direction	Function
SWDCLK	I	Serial Wire Clock. This pin is the clock for debug module when running in Serial Wire debug mode.
SWDIO	I / O	Serial Wire debug data IO. Used by an external debug tool to communicate with and control the Cortex-M0 debug system.

The HALTED and EDBGRQ signals can be used to halt the CPU based on an external event. This allows to halt an external hardware synchronously with the CPU. The halt can be recognized by the external hardware based on the HALTED output. This can be used for example to synchronize with an logic analyzer and request a breakpoint or in a multi CPU scenario.

20.2.12.1 Internal pull-up and pull-down on SWD/SPD pins

It is a requirement to ensure none floating SWD/SPD input pins, as they are directly connected to flip-flops controlling the debug function. To avoid any uncontrolled I/O voltage levels internal pull-up and pull-downs on SWD/SPD input pins are provided.

- SWDIO/SPD: Internal pull-up
- SWCLK: Internal pull-down

20.2.13 Reset

The debug system register bits are reset by Power-on reset. Other reset in the system do not have an effect on the debug register, if the tool is registered.

20.3 Debug System Power Save Operation

The Debug System is in the “always-on” power domain, which allows to connect the debugger to the device. The Debug System does not support any special power mode or power save operation. The power supply of the Debug System is directly connected to the main chip part including the system components.

20.4 Service Request Generation

The debugger can set DHCSR.C_MASKINTS to 1 to prevent PendSV, SysTick and external configured interrupts from occurring.

20.5 Debug behavior

The Debug System is based on events. Whereas events are an entry to debug state, if halting debug is enabled. Debug events available are:

- Internal halt request
- Breakpoint
- Watchpoint
- Vector catch
- External debug request based on

The following events are synchronous debug events:

- Breakpoint debug events, caused by execution of a BKPT instruction
- Breakpoint debug events, caused by execution of a match in the BPU
- Vector catch debug events
- Step debug events (DHCSR.C_STEP)

The following events are asynchronous debug events:

- Watchpoint debug events, including PC match watchpoints
- Halt request debug event (DHCSR.C_HALT)
- External signal based debug request event (EDBGRQ signal)

Asynchronous debug events do have a lower prioritization then synchronous generated events. An instruction can generate a number of synchronous debug events. It also can generate a number of asynchronous exceptions.

The Debug Fault Status register (DFSR) contains a status bit for each debug event. The bits are write-one-to-clear. These bits are set to 1 when a debug event causes the processor to halt or generate an exception. The bits are also updated if the event is ignored.

Software must write 0xA05F to the DHCSR.DBGKEQ register in order to be able to have debug support available form CPU side.

20.6 Power, Reset and Clock

The requirements for power, reset and clock are derived from ARM.

20.6.1 Power management

There is no power management implemented for the debug system. . Nevertheless the tool does has to follow the regular tool flow by setting the CDBGPWRUPREQ in order to register the tool.

20.6.2 Debug System reset

The SWD register are in the power on reset (PORST) domain.

The Debug logic is reset by system reset, if no tool is connected to chip and therefore not registered, which is indicated by **CDBGPWRUPREQ** not set.

Processor reset

A processor or warm reset (SYSRESETn) initializes the majority of the processor, excluding debug logic, BKPT unit, DWT unit and SCS.

PowerOn reset

PORESETn reset initializes the SWD access port, the AHB-AP logic and all other debug system related functions.

Normal operation

During normal operation the resets PORESETn and SYSRESETn are deasserted.

Processor reset affects Debug System

A System reset also affects the Debug System, if a tool is not registered. The tool registration is reflected by an activation of the register bit CDBGPWRUPACK, which activates by a debugger enabling the CDBGPWRUPREQ. .

20.6.3 Debug System Clocks

The Debug system clock is called DCLK and is derived directly from SCLK (free running clock). DCLK must always be driven while a debugger is connected.

20.7 Initialization and System Dependencies

20.7.1 ID Codes

The following sub chapters describes the available IDs.

20.7.1.1 CPUID

CPUID is $410CC200_H$.

20.7.1.2 ROM Table

The DAP controller comes with the default ARM JTAG IDs defined by the IDCODE. This value is: $0BB11477_H$ for SW-DP.

The JEP106 ID is defined by the JEDEC Standard for Infineon as $C1_H$

To identify Infineon as manufacturer and identify XMC1100 by it's device ID, a ROM table is added. A 4 k memory block in system memory is defined, which contains the IDs.

The CortexM0 ROM table is located at 0xE00F_F000 and is defined with the following values:

- M0 ROM table, at 0xE00F_F000 - PID JEP106: 0x3B/0x4; part# = 471
- Data Watchpoint/Trace, at 0xE000_1000: PID JEP106: 0x3B/0x4; part# = 0x008
- Breakpoint Unit, at 0xE000_2000: PID JEP106 = 0x3B/0x4; part# = 0x00B

The format of the ROM table is illustrated in **Table 20-3**.

Table 20-3 PID Values of XMC1100 ROM Table

Name	Offset	Description	Reference
PID0	$FE0_H$	Peripheral ID0	Part Number [7:0]
PID1	$FE4_H$	Peripheral ID1	bits [7:4] JEP106 ID code [3:0] bits [3:0] Part Number [11:8]
PID2	$FE8_H$	Peripheral ID2	bits [7:4] Revision bit [3] == 1: JEDEC assigned ID fields bits [2:0] JEP106 ID code [6:4]
PID3	FEC_H	Peripheral ID3	bits [7:4] RevAnd, minor revision field bits [3:0] if non-zero indicate a customer-modified block
PID4	$FD0_H$	Peripheral ID4	bits [7:4] 4KB count bits [3:0] JEP106 continuation code

The ROM table values representing Infineon Part Number, JEP106 and Revision number can be checked in the SCU chapter.

20.8 Debug System Registers

Registers Overview

The absolute register address is calculated by adding:

The DWT, BPU, ROM table, DCB and debug register in the SCS are accessible memory mapped by the debugger and also from the CPU.

Table 20-4 Registers Address Space

Module	Base Address	End Address	Note
DWT	$E000\ 1000_H$	$E000\ 1FFF_H$	Data Watchpoint
BP	$E000\ 2000_H$	$E000\ 2FFF_H$	Breakpoint Unit
SCS	$E000\ E000_H$	$E000\ EEFF_H$	SCS (here DBG CTRL)
ROM	$E00F\ F000_H$	$E00F\ FFFF_H$	ROM table area

Table 20-5 Register Overview

Short Name	Description	Offset Addr.	Access Mode		Description See
			Read	Write	
SCS_DFSR	Debug Fault Status Register	$D30_H$			Page 20-20
SCS_DHCSR	Debug Halting Control and Status Register	$DF0_H$			Page 20-22
SCS_DCRSR	Debug Core Register Selector Register	$DF4_H$			Page 20-28
SCS_DCRDR	Debug Core Register Data Register	$DF8_H$			Page 20-29
SCS_DEMCR	Debug Exception and Monitor Control Register	DFC_H			Page 20-30
DWT_CTRL	DWT Control Register	000_H			Page 20-31
DWT_PCSR	DWT program Counter Sample Register	$01C_H$			Page 20-32
DWT_COMP0	DWT Comparator register	020_H			Page 20-32
DWT_COMP1	DWT Comparator register	030_H			Page 20-32
DWT_MASK0	DWT MASK register	024_H			Page 20-33
DWT_MASK1	DWT MASK register	034_H			Page 20-33

Table 20-5 Register Overview (cont'd)

Short Name	Description	Offset Addr.	Access Mode		Description See
			Read	Write	
DWT_FUNCTION0	DWT Comparator Function register	028_H			Page 20-34
DWT_FUNCTION1	DWT Comparator Function register	038_H			Page 20-34
BP_CTRL	Breakpoint Control register	000_H			Page 20-35
BP_COMP0	Breakpoint Comparator register	008_H			Page 20-36
BP_COMP1	Breakpoint Comparator register	008C			Page 20-36
BP_COMP2	Breakpoint Comparator register	010_H			Page 20-36
BP_COMP3	Breakpoint Comparator register	014_H			Page 20-36

20.8.1 DFSR - Debug Fault Status Register

SCS_DFSR

The DFSR registers provides the top level reason why a debug event has occurred. Writing 1 to a register bit clears the bit to 0. A read of the HALTED bit by an instruction executed by stepping returns an UNKNOWN value.

SCS_DFSR

Debug Fault Status Register ($D30_H$) 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											EXTERNAL	VCATCH	DWTTRAP	BKPT	HALTED
r											rw	rw	rw	rw	rw

Field	Bits	Type	Description
HALTED	0	rw	**HALTED** Indicates a debug event generated by a C_HALT or C_STEP request, triggered by a write to the DHCSR. 0_B no active halt request debug event. 1_B halt request debug event active.
BKPT	1	rw	**BKPT** Indicates a debug event generated by BKPT instruction execution or a breakpoint match in the BPU. 0_B no breakpoint debug event. 1_B at least one breakpoint debug event.
DWTTRAP	2	rw	**DWTTRAP** Indicates a debug event generated by the DWT. 0_B no debug events generated by the DWT. 1_B at least one debug event generated by the DWT.
VCATCH	3	rw	**VCATCH** Indicates whether a vector catch debug event was generated. 0_B no vector catch debug event generated. 1_B vector catch debug event generated. *Note: The corresponding*
EXTERNAL	4	rw	**EXTERNAL** Indicates an asynchronous debug event generated because of EDBGRQ being asserted. 0_B no EDBGRQ debug event. 1_B EDBGRQ debug event.
0	[31:5]	r	**Reserved**

20.8.2 DHCSR - Debug Halting Control and Status Register

SCS_DHCSR

Controls halting debug. When C_DEBUGEN is set to 1, C_STEP and C_MASKINTS must not be modified when the processor is running (S_HALT is 0 when the processor is running). When C_DEBUGEN is set to 0, the processor ignores the values of all other bits in this register.

A separate register is represented below for read and write as the function changes at some bits access based.

SCS_DHCSR

Debug Halting Control and Status Register [Read Mode]

($DF0_H$) **Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0						S_RESET_ST	S_RETIRE_ST	0				S_LOCKUP	S_SLEEP	S_HALT	S_REGRDY
r						r	r	r				r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0						r						C_MASKINTS	C_STEP	C_HALT	C_DEBUGEN
r												r	r	r	r

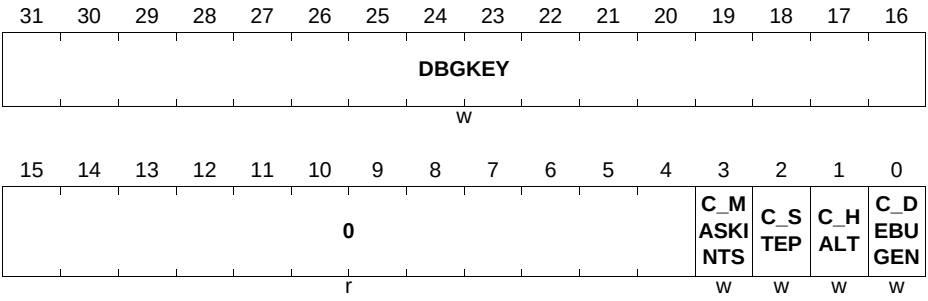
Field	Bits	Type	Description
C_DEBUGEN	0	r	**Halting debug enable bit** 0_B Halting debug disabled. 1_B Halting debug enabled. *Note: If a debugger writes to DHCSR to change the value of this bit from 0 to 1, it must also write 0 to the C_MASKINTS bit, otherwise behavior is UNPREDICTABLE. This bit can only be written from the DAP Access to the DHCSR from Software running on the processor it cannot be set. This bit is 0 after Power-on reset.*
C_HALT	1	r	**Processor halt bit.** The effects of writes to this bit are: 0_B Request a halted processor to run. 1_B Request a running processor to halt. *Note: This bit is unknown after power-on reset*

Field	Bits	Type	Description
C_STEP	2	r	**Processor step bit.** The effects of writes to this bit are: 0_B Single-stepping disabled. 1_B Single-stepping enabled. *Note: This bit is unknown after power-on reset*
C_MASKINTS	3	r	**Mask PEDSV, SysTick and external configurable interrupts.** The effects of writes to this bit are: 0_B Do not mask 1_B Mask PendSV, SysTick and external configurable interrupts. The effect of any attempt to change the value of this bit is UNPREDICTABLE unless both: • before the write to DHCSR, the value of the C_HALT bit is 1 • the write to the DHCSR that changes the C_MASKINTS bit also writes 1 to the C_HALT bit. This means that a single write to DHCSR cannot set the C_HALT to 0 and change the value of the C_MASKINTS bit. When DHCSR.C_DEBUGEN is set to 0, the value of this bit is UNKNOWN. This bit is UNKNOWN after Power-on reset.
S_REGRDY	16	r	**S_REGRDY status - a handshake flag for transfers through DCRDR** How to work with: • Writing to DCRSR clears the bit to 0. • Completion of the DCRDR transfer then sets the bit to 1 Check DCRDR for more information. 0_B There has been a wrote to the DCRDR, but the transfer is not complete. 1_B The transfer to or from the DCRDR is complete. *Note: This bit is only valid when the processor is in Debug State, otherwise the bit is UNKONWN.*

Field	Bits	Type	Description
S_HALT	17	r	**S_HALT indicates whether the processor is in Debug state.** 0_B Not in Debug State. 1_B In Debug State.
S_SLEEP	18	r	**S_SLEEP indicates whether the processor is sleeping.** 0_B Not sleeping. 1_B Sleeping. *Note: The debugger must set the DHCSR.C_HALT bit to 1 to gain control, or wait for an interrupt or other wake-up event to wake-up the system.*
S_LOCKKUP	19	r	**S_LOCKUP indicates whether the processor is locked up because of an unrecoverable exception.** 0_B Not locked up. 1_B Locked up. *Note: This bit is only read as 1 when accessed by a remote debugger using the DAP. The value of 1 indicates that the processor is running, but locked up due to an unrecoverable exception case.*

Field	Bits	Type	Description
S_RETIRE_ST	24	r	**S_RETIRE_ST - When not in Debug state, indicates whether the processor has completed the execution of an instruction since the last read of DHCSR:** 0_B No instruction has completed since the last DHCSR read. 1_B At least one instructions has completed since last DHCSR read. This is a sticky bit, that clears to 0 on a read of DHCSR. This bit is UNKNOWN: • after a Local reset, but is set to 1 as soon as the processor completes execution of an instruction • when S_LOCKUP is set to 1 • when S_HALT is set to 1 *Note: When the processor is not in Debug state, a debugger can check this bit to determine if the processor is stalled on a load store or fetch access.*
S_RESET_ST	25	r	**S_RESET_ST indicates whether the processor has been reset since the last read of DHCSR.** 0_B No reset since last DHCSR read. 1_B At least one rest since last DHCSR read. *Note: This is a sticky bit, that clears to 0 on a read of DHCSR.*
0	[15:4], [23:20], [31:26]	r	**Reserved**

SCS_DHCSR
Debug Halting Control and Status Register [Write Mode]
($DF0_H$) **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
C_DEBUGEN	0	w	**Halting debug enable bit** 0_B Halting debug disabled. 1_B Halting debug enabled. *Note: If a debugger writes to DHCSR to change the value of this bit from 0 to 1, it must also write 0 to the C_MASKINTS bit, otherwise behavior is UNPREDICTABLE. This bit can only be written from the DAP Access to the DHCSR from Software running on the processor it cannot be set. This bit is 0 after Power-on reset.*
C_HALT	1	w	**Processor halt bit.** The effects of writes to this bit are: 0_B Request a halted processor to run. 1_B Request a running processor to halt. *Note: This bit is unknown after power-on reset*
C_STEP	2	w	**Processor step bit.** The effects of writes to this bit are: 0_B Single-stepping disabled. 1_B Single-stepping enabled. *Note: This bit is unknown after power-on reset*

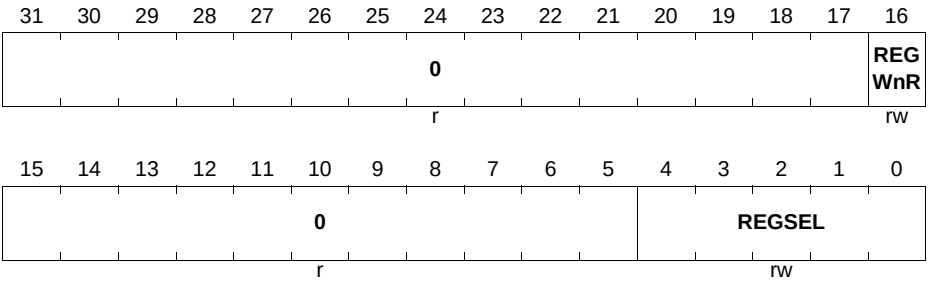
Field	Bits	Type	Description
C_MASKINTS	3	w	**Mask PEDSV, SysTick and external configurable interrupts.** The effects of writes to this bit are: 0_B Do not mask 1_B Mask PendSV, SysTick and external configurable interrupts. The effect of any attempt to change the value of this bit is UNPREDICTABLE unless both: • before the write to DHCSR, the value of the C_HALT bit is 1 • the write to the DHCSR that changes the C_MASKINTS bit also writes 1 to the C_HALT bit. This means that a single write to DHCSR cannot set the C_HALT to 0 and change the value of the C_MASKINTS bit. When DHCSR.C_DEBUGEN is set to 0, the value of this bit is UNKNOWN. This bit is UNKNOWN after Power-on reset.
DBGKEY	[31:16]	w	**DEBUG Key Bits [31:16]!!!** Software must write **0xA05F** to this field to enable write access to bits [15:0], otherwise the processor ignores the write access
0	[15:4]	r	**Reserved**

20.8.3 DCRSR - Debug Core Register Selector Register

SCS_DCRSR

The DCRSR together with DCRDR (Debug Core Register Data Register) provides debug access to the ARM core register and special-purpose registers. A write to DCRSR specifies the register to transfer, whether the transfer is a read or a write, and starts the transfer. This register is only accessible in Debug state.

SCS_DCRSR
Debug Core Register Selector Register ($DF4_H$) **Reset Value: 0000 0000$_H$**



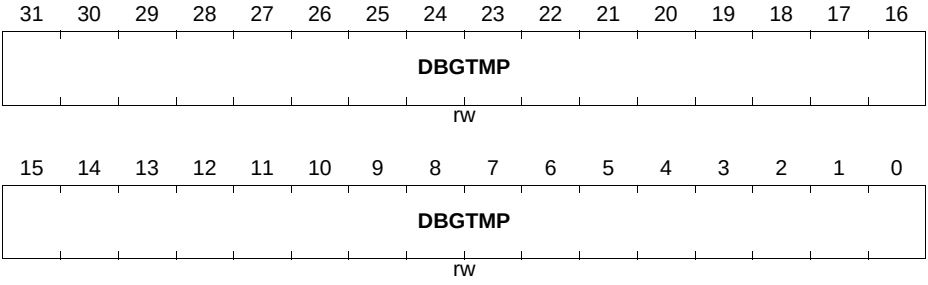
Field	Bits	Type	Description
REGSEL	[4:0]	rw	**REGSEL - Specifies the ARM core register or special-purpose register to transfer** 00000_B- 01100_B ARM core register R0-R12. For example, 0b00000 specifies R0 and 0b00101 specifies R5 01101_B The current SP. See also values 0b10001 and 0b10010. 01110_B LR. 01111_B Debug Return Address. 10000_B xPSR. 10001_B Main stack pointer, MSP. 10010_B Process stack pointer, PSP. 10100_B Bits[31:24] Control; Bits[23:8] Reserved; Bits[7:0] PRIMASK. In each field, the valid bits are packed with leading zeros. For example DCRDR [31L26] is 0b00000.
REGWnR	16	rw	**REGWnR specifies the type of access for the transfer** 0_B read. 1_B write.
0	[31:17], [15:5]	r	**Reserved**

20.8.4 DCRDR - Debug Core Register Data Register

SCS_DCRDR

The DCRDR works with the DCRSR (Debug Core Register Selector Register). The DCRDR provides debug access to the ARM core registers and special-purpose registers. The DCRDR is the data register for these accesses.

SCS_DCRDR
Debug Core Register Data Register (DF8$_H$) **Reset Value: xxxx xxxx$_H$**



Field	Bits	Type	Description
DBGTMP	[31:0]	rw	**DBGTMP - Data temporary cache, for reading and writing register.** This register is UNKONWN: • on reset • when DHCSR.S_HALT = 0 • when DHCSR.S_REGRDY = 0 during execution of a DCRSR based transaction that updates the register.

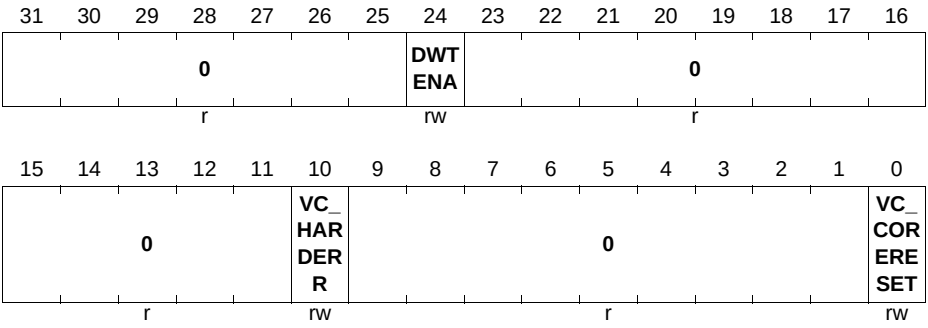
20.8.5 DEMCR - Debug Exception and Monitor Control Register

SCS_DEMCR

The DEMCR purpose is to manage vector catch behavior and enable the DWT.

A Power-on reset sets all register bits to 0. A local reset sets DWTENA to 0 but does not affect VC-HARDERR or VC_CORERESET.

SCS_DEMCR
Debug Exception and Monitor Control Register (DFC_H) Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
VC_CORERESET	0	rw	**VC_CORERESET - Enable Reset Vector Catch. This causes a Local reset to handle a running system** 0_B Reset Vector Catch disabled. 1_B Reset Vector Catch enabled. *Note: If DHCSR.C_DEBUGEN is set to 0, the processor ignores the value of this bit.*
VC_HARDERR	10	rw	**VC_CORERESET - Enable Reset Vector Catch. This causes a Local reset to halt a running system.** 0_B halting debug trap disabled. 1_B hating debug trap enabled. *Note: If DHCSR.C_DEBUGEN is set to 0, the processor ignores the value of this bit.*
DWTENA	24	rw	**DWTENA - Global enable for all features configured by the DWT unit.** 0_B DWT disabled. 1_B DWT enabled. *Note: When DWTENA is set to 0 DWT registers return UNKNOWN values on reads. In addition the processor ignores writes to the DWT while DWTENA is 0.*

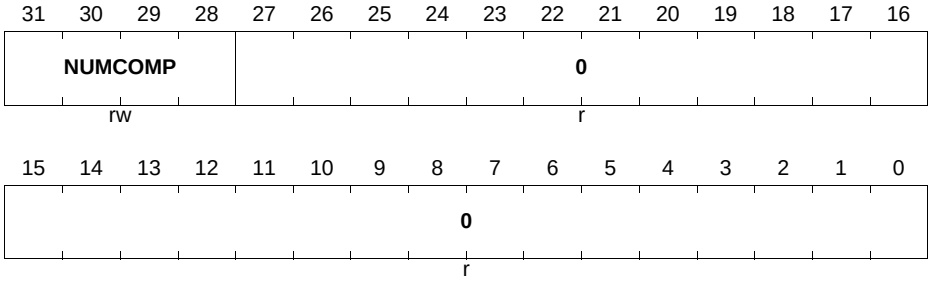
Field	Bits	Type	Description
0	[31:25], [23:16], [15:11], [9:1]	r	**Reserved**

20.8.6 DWT_CTRL - Data Watchpoint Control Register

DWT_CTRL

The DWT_CTRL register defines the number of comparators implemented.

DWT_CTRL
Debug Halting Control and Status Register (000_H) **Reset Value: $0000\ 0000_H$**



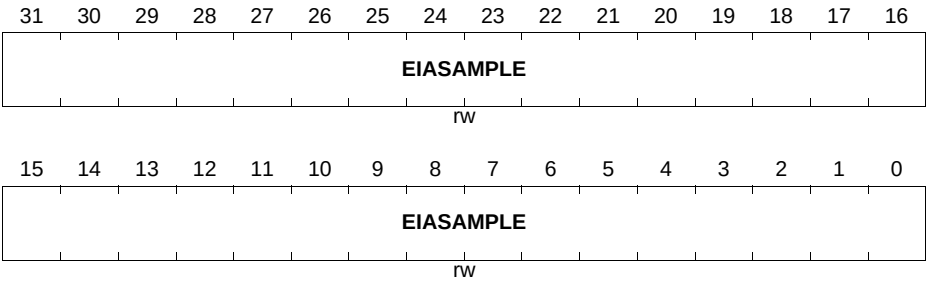
Field	Bits	Type	Description
NUMCOMP	[31:28]	rw	**Number of comparators available** 0000_BNo comparator support.
0	[27:0]	r	**Reserved**

20.8.7 DWT_PCSR - Program Counter Sample Register

DWT_PCSR

The DWT_PCSR register samples the current value of the program counter. The register is UNKNOW on reset.

DWT_PCSR
Program Counter Sample Register (01C$_H$) **Reset Value: xxxx xxxx$_H$**



Field	Bits	Type	Description
EIASAMPLE	[31:0]	rw	**EIASAMPLE** Executed instruction address sample register

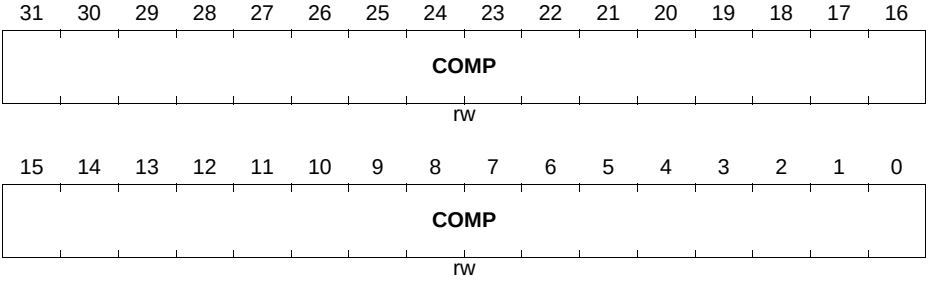
20.8.8 DWT_COMPx - DWT Comparator register

DWT_COMPx

The DWT_COMPx register provides a reference value for use by comparator x. The value is UNKNOW on reset. DWT_CTRL.NUMCOMP defines the number of implemented DWT_COMPx register, from 0 to (NUMCOMP-1).

The DWT_COMP [1 .. 0] register are implemented.

DWT_COMP0
DWT Comparator register 0 (020$_H$) **Reset Value: xxxx xxxx$_H$**
DWT_COMP1
DWT Comparator register 1 (030$_H$) **Reset Value: xxxx xxxx$_H$**



Field	Bits	Type	Description
COMP	[31:0]	rw	**COMP** Reference value for comparison

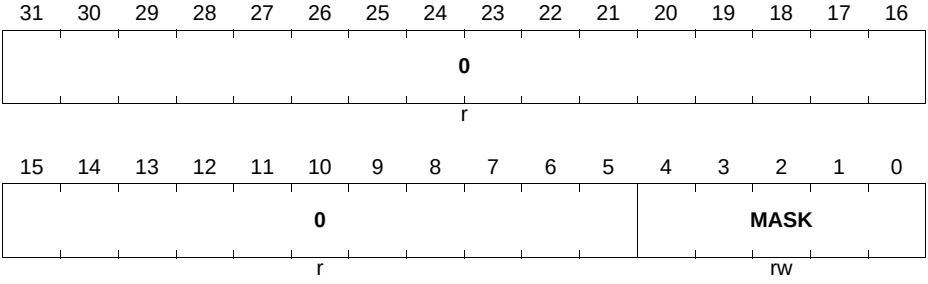
20.8.9 DWT_MASKx - DWT Comparator Mask Register

DWT_MASKx

The DWT_MASKx register provides the size of the ignore mask applied to the access address range matching by comparator x. The value is UNKONWN on reset. DWT_CTRL.NUMCOMP defines the number of implemented DWT_COMPx register, from 0 to (NUMCOMP-1).

The DWT_MASK [1 .. 0] register are implemented.

DWT_MASK0
Debug Halting Control and Status Register (24_H) **Reset Value: 0000 0000$_H$**
DWT_MASK1
Debug Halting Control and Status Register (34_H) **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
MASK	[4:0]	rw	**MASK** The size of the ignore mask applied to address range matching. Writing all ones to this field and reading it back can be used to determine the maximum mask size supported (not all mask bit must be implemented).
0	[31:5]	r	**Reserved**

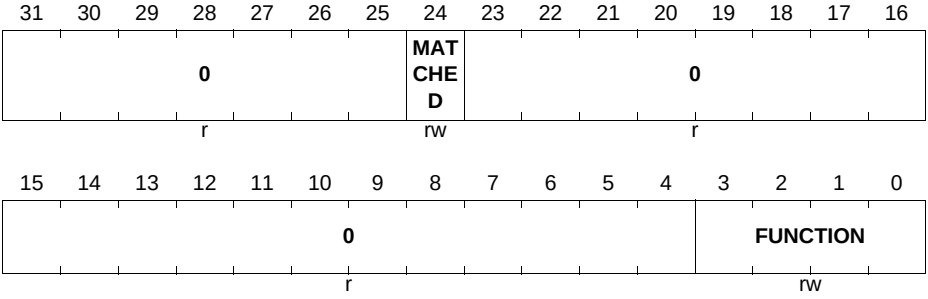
20.8.10 DWT_FUNCTIONx - Comparator Function Register

DWT_FUNCTIONx

The DWT_FUNCTIONx register controls the operation of the comparator DWT_COMPx register. DWT_CTRL.NUMCOMP defines the number of implemented DWT_FUNCTIONx registers, from 0 to (NUMCOMP-1).

The DWT_FUNCTION [1 .. 0] register are implemented.

DWT_FUNCTION0
Comparator Function Register **(28_H)** **Reset Value: 0000 0000$_H$**
DWT_FUNCTION1
Comparator Function Register **(38_H)** **Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
FUNCTION	[3:0]	rw	**FUNCTION** Select action on comparator match- 0000_BDisabled. 0001_BReserved. 0010_BReserved. 0011_BReserved. 0100_BPC watchpoint event - input laddr. 0101_BWatchpoint event - input Daddr (read only) 0110_BWatchpoint event - input Daddr (write only) 0111_BWatchpoint event - input Daddr (read/write) $1xxx_B$ reserved *Note: This field is set to 0 on a Power-on reset.*

Field	Bits	Type	Description
MATCHED	24	rw	**MATCHED** Comparator match. It indicates that the operation defined by FUNCTION has occurred since the bit was last read. 0_B the associated comparator has matched. 1_B the associated comparator has not matched. *Note: Reading the register clears this bit to 0.*
0	[31:25], [23:16], [15:4]	r	**Reserved**

20.8.11 BP_CTRL - Breakpoint Control Register

BP_CTRL

The Breakpoint Control Register provides BPU implementation information and the global enable for the BPU.

BP_CTRL
Breakpoint Control Register **(000_H)** **Reset Value: 0000 0040_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								NUM_CODE				0		KEY	ENABLE
r								r				r		rw	rw

Field	Bits	Type	Description
ENABLE	0	rw	**ENABLE the BPU** 0_B BPU is disabled. 1_B BPU is enabled. *Note: This bit is set to 0 on Power-on reset.*

Field	Bits	Type	Description
KEY	1	rw	**KEY RAZ on reads, SBO for writes.** If written as zero, the write to the register is ignored.
NUM_CODE	[7:4]	r	**NUM_CODE, the number of breakpoint comparators.**
0	[31:8], [3:2]	r	**Reserved**

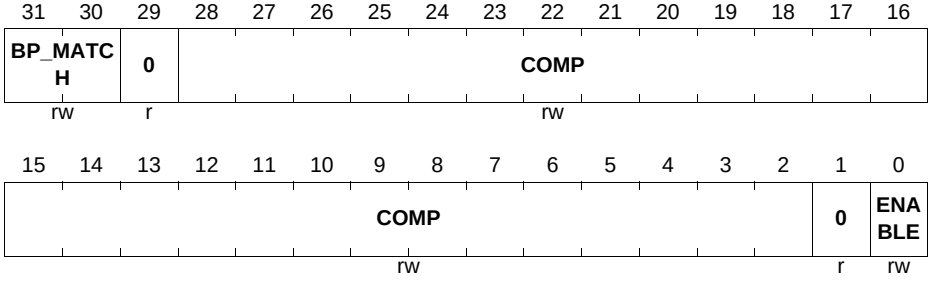
20.8.12 Breakpoint Comparator Registers

BP_COMPx

The BP_COMPx register holds a breakpoint address for comparison with instruction addresses in the Code memory region. A comparator can only be enabled when BP_CTRL.ENABLE is set to 1. BP_CTRL.NUM_CODE defines the number of implemented BP_COMPx registers, from 0 to (NUM_CODE-1).

The BP_COMP [3 .. 0] register are implemented.

BP_COMP0
Brakpoint Comparator X Register **(008_H)** **Reset Value: 0000 0000_H**
BP_COMP1
Brakpoint Comparator X Register **($00C_H$)** **Reset Value: 0000 0000_H**
BP_COMP2
Brakpoint Comparator X Register **(010_H)** **Reset Value: 0000 0000_H**
BP_COMP3
Brakpoint Comparator X Register **(014_H)** **Reset Value: 0000 0000_H**



Field	Bits	Type	Description
ENABLE	0	rw	**ENABLE the comparator** 0_B Comparator is disabled. 1_B Comparator is enabled. *Note: This bit is set to 0 on a Power-on reset. BP_CTRL:ENABLE must also be set to 1 to enable a comparator.*
COMP	[28:2]	rw	**Stores bits [28:0] of the comparison address.** The comparison address is compared with the address from the Code memory region. Bits [31:29] and [1:0] of the comparison address are zero. 0_B Comparator is disabled. 1_B Comparator is enabled. *Note: The field is UNKNOWN on Power-on reset.*
BP_MATCH	[31:30]	rw	**BP_MATCH defines the behavior when the COMP address is matched.** 00_B no breakpoint matching. 01_B breakpoint on lower halfword, upper is unaffected. 10_B breakpoint on upper halfword, lower is unaffected. 11_B breakpoint on both lower and upper halfwords. *Note: The field is UNKNOWN on reset.*
0	29, 1	r	**Reserved**

Lists of Figures and Tables

List of Figures

List of Figures

List of Figures

List of Figures

List of Figures

List of Tables

List of Tables

List of Tables

w w w . i n f i n e o n . c o m

Published by Infineon Technologies AG