FUJITSU SEMICONDUCTOR
CONTROLLER MANUAL

MN704-00001-1v0-E

F²MC-16FX
16-BIT MICROCONTROLLER
MB96680 series
HARDWARE MANUAL

FUJITSU SEMICONDUCTOR LIMITED

FUJITSU

F²MC-16FX
16-BIT MICROCONTROLLER
MB96680 series
HARDWARE MANUAL

For the information for microcontroller supports, see the following web site.
This web site includes the **"Customer Design Review Supplement"** which provides the latest cautions on system development and the minimal requirements to be checked to prevent problems before the system development.

http://edevice.fujitsu.com/micom/en-support/

FUJITSU SEMICONDUCTOR LIMITED

Preface

■ Objectives and intended reader

Thank you very much for your continued patronage of Fujitsu semiconductor products.

The MB96600 Super series has been developed as a general-purpose version of the F^{2}MC-16FX series, which is an original 16-bit single-chip microcontroller compatible with the Application Specific IC (ASIC).

This manual explains the functions and operation of the MB96600 Super series for designers who actually use the MB96600 Super series to design products.
Please read this manual first and refer to the respective "Data Sheet" of devices for device-specific details..

■ Trademark

F^{2}MC is the abbreviation of FUJITSU Flexible Microcontroller.
Other system and product names in this manual are trademarks of respective companies or organizations.
The symbols ™ and ® are sometimes omitted in this manual.

- The contents of this document are subject to change without notice.
  Customers are advised to consult with sales representatives before ordering.
- The information, such as descriptions of function and application circuit examples, in this document are presented solely for the purpose of reference to show examples of operations and uses of FUJITSU SEMICONDUCTOR device; FUJITSU SEMICONDUCTOR does not warrant proper operation of the device with respect to use based on such information. When you develop equipment incorporating the device based on such information, you must assume any responsibility arising out of such use of the information. FUJITSU SEMICONDUCTOR assumes no liability for any damages whatsoever arising out of the use of the information.
- Any information in this document, including descriptions of function and schematic diagrams, shall not be construed as license of the use or exercise of any intellectual property right, such as patent right or copyright, or any other right of FUJITSU SEMICONDUCTOR or any third party or does FUJITSU SEMICONDUCTOR warrant non-infringement of any third-party's intellectual property right or other right by using such information. FUJITSU SEMICONDUCTOR assumes no liability for any infringement of the intellectual property rights or other rights of third parties which would result from the use of information contained herein.
- The products described in this document are designed, developed and manufactured as contemplated for general use, including without limitation, ordinary industrial use, general office use, personal use, and household use, but are not designed, developed and manufactured as contemplated (1) for use accompanying fatal risks or dangers that, unless extremely high safety is secured, could have a serious effect to the public, and could lead directly to death, personal injury, severe physical damage or other loss (i.e., nuclear reaction control in nuclear facility, aircraft flight control, air traffic control, mass transport control, medical life support system, missile launch control in weapon system), or (2) for use requiring extremely high reliability (i.e., submersible repeater and artificial satellite).
  Please note that FUJITSU SEMICONDUCTOR will not be liable against you and/or any third party for any claims or damages arising in connection with above-mentioned uses of the products.
- Any semiconductor devices have an inherent chance of failure. You must protect against injury, damage or loss from such failures by incorporating safety design measures into your facility and equipment such as redundancy, fire protection, and prevention of over-current levels and other abnormal operating conditions.
- Exportation/release of any products described in this document may require necessary procedures in accordance with the regulations of the Foreign Exchange and Foreign Trade Control Law of Japan and/or US export control laws.
- The company names and brand names herein are the trademarks or registered trademarks of their respective owners.

Related Manuals

The manuals related to this series are listed below. See the manual appropriate to the applicable conditions.
The contents of these manuals are subject to change without notice. Contact us to check the latest versions available.

■ Hardware Manual

- **F^{2}MC-16FX FAMILY MB96F680 SERIES HARDWARE MANUAL (this manual)**

■ Data sheet

For details about device-specific, Pin Assignments, electrical characteristics, package dimensions, ordering information etc., see the following document.

- **16-bit PROPRIETARY MICROCONTROLLER F^{2}MC-16FX FAMILY DATA SHEET**

* The data sheets for each series are provided.
Please see the appropriate data sheet for the series that you are using.

■ CPU Programming manual

For details about the functions and operations of 16FX-CPU, see the following document.

- **F^{2}MC-16FX 16-BIT MICROCONTROLLER PROGRAMMING MANUAL**

How to Use This Manual

■ Finding a Function

The following methods can be used to search for the explanation of a desired function in this manual:

- Search from the table of the contents
  The table of the contents lists the manual contents in the order of description.

- Search from the register
  The address where each register is located is not described in the text. To verify the address of a register, see "I/O Map" in Appendixes.

■ Terminology

This manual uses the following terminology.

Term	Explanation
Word	Indicates access in units of 32 bits.
Half word	Indicates access in units of 16 bits.
Byte	Indicates access in units of 8 bits.

■ Notations

- The notations in bit configuration of the register explanation of this manual are written as follows.
  - bit : bit number
  - Attribute : Attributes for read and write of each bit
    - R : Read only
    - W : Write only
    - R/W : Readable/Writable
    - - : Undefined
  - Initial value : Initial value of the register after reset
    - 0 : Initial value is "0"
    - 1 : Initial value is "1"
    - X : Initial value is undefined

- The multiple bits are written as follows in this manual.
  Example : bit7:0 indicates the bits from bit7 to bit0

- The values such as for addresses are written as follows in this manual.
  - Hexadecimal number : "0x" is attached in the beginning of a value as a prefix (example : 0xFFFF)
  - Binary number : "0b" is attached in the beginning of a value as a prefix (example: 0b1111)
  - Decimal number : Written using numbers only (example : 1000)

CONTENTS

CHAPTER: OVERVIEW

The MB96600 Super series is a family member of the F^2MC-16FX micro controllers. It consists of many different series of microcontrollers, which are targeting different applications. Programming between all members of the Super series is common.

Management Code: 96F68OVERVIEW-E01.0

1. Features

The feature set of F^{2}MC-16FX family's MB96F680 series makes it especially well suited for automotive applications.

■ Features

- 16-bit core CPU, up to 32 MHz CPU clock, 31.2 ns instruction cycle time
- 0.18µm CMOS Process Technology
- Optimized instruction set for controller applications (bit, byte, word and long-word data types; 23 different addressing modes; barrel shift; variety of pointers)
- 8-byte instruction execution queue
- Signed multiply (16-bit × 16-bit) and divide (32-bit/16-bit) instructions available
- Internal voltage regulator supports reduced internal MCU voltage, offering low EMI and low power consumption figures
- Code Security Feature
- Up to 1 FULL-CAN interfaces; conforming to Version 2.0 Part A and Part B, ISO16845 certified
- Powerful interrupt functions (8 progr. priority levels; up to 7 external interrupts)
- Fast Interrupt processing
- Up to 2 channels DMA - Automatic transfer function independent of CPU, can be assigned freely to resources
- Three independent clock timers (23-bit RC clock timer, 23-bit Main clock timer, 17-bit Sub clock timer)
- Hardware Watchdog Timer
- Up to 2 channels full duplex USARTs (SCI/LIN)
- Up to 1 channels I^{2}C with 400 kbit/s
- Up to 14 channels analog inputs for A/D Converter (Resolution 10-bit or 8-bit)
- Up to 2 channels16-bit reload timer, and 1 channel for PPG
- Up to 4 channels ICU (Input capture unit) 16-bit
- Up to 2 channels 16-bit free running timer
- Up to 4 channels × 16-bit (Up to 8 channels × 8-bit) Programmable Pulse Generator
- Up to 2 channels Stepper Motor Controller with integrated high current output drivers
- LCD controller with up to 4 COM × 32 SEG, internal or external voltage generation
- Low Power Consumption - 13 operating modes (different Run, Sleep, Timer modes, Stop mode)
- 4 MHz to 8 MHz external clock (up to 16 MHz when using Fast Clock Input mode), on-chip PLL with programmable multiplication factor 1 ... 8
- 32-100 kHz Subsystem Clock
- 100kHz/2MHz internal RC clock
- Programmable Pull-up resistors for all ports
- Package: 80-pin plastic LQFP
- Controller Area Network (CAN) - License of Robert Bosch GmbH
- On-Chip debug system

2. Super Series Lineup

Table 2-1 provides an overview of the MB96600 Super series.

■ Super Series Lineup

The F^2MC-16FX MB96600 Super series covers the F^2MC-16FX series as shown in Table 2-1.

Table 2-1 MB96600 Super Series Lineup

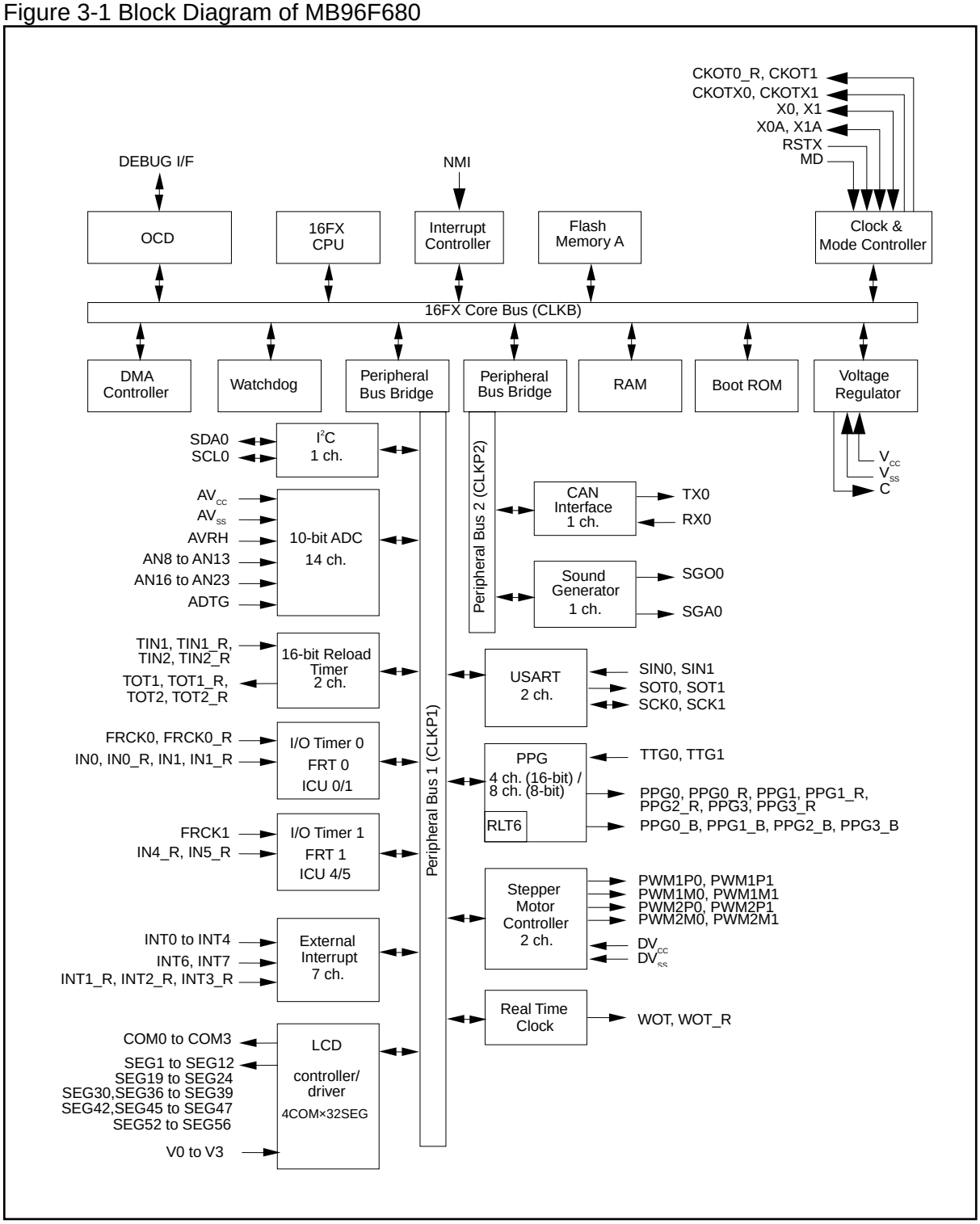
Series	Flash Products
MB96610	MB96F61x
MB96620	MB96F62x
MB96630	MB96F63x
MB96640	MB96F64x
MB96650	MB96F65x
MB96660	MB96F66x
MB96670	MB96F67x
MB96680	MB96F68x
MB96690	MB96F69x
MB966A0	MB96F6Ax
MB966B0	MB96F6Bx
MB966C0	MB96F6Cx

3. Block Diagram of MB96F680

Figure 3-1 shows a principle block diagram of MB96F680 series.

■ Block Diagram of MB96F680

Figure 3-1 Block Diagram of MB96F680



4. General Note on Using This Document

This chapter contains some general notes about this document.

■ Device Dependent Resources

The derivatives of a Microcontroller series may have different sets of resources (e.g. features, number of channels, interconnections). Details about an individual device can be found in the following documents:

- Number of resources and available channels: Datasheet of the series.
- Peculiarities of a resource in a derivative (e.g. interconnections between resources): "CHAPTER: OVERVIEW" in the Hardware manual of the series.

■ Described Resources

All chapters about resources of the Microcontroller like USART, Reload timers, General purpose ports describe the function of only one channel of this resource. The behavior of all other resources of the same type is the same. If there is an exception it is described in the "CHAPTER: OVERVIEW" or in the corresponding chapter of this Hardware manual.
In the Hardware manual chapters the placeholder "n" is used for a resource channel number (e.g. in the register name). This placeholder needs to be replaced with the resource channel number of the resource to be used.

Example: Registers of Reload Timer 0 and Reload Timer 1

Register name as in the Chapter "Reload timer"	Register name define in Header file for Reload Timer 0	Register name define in Header file for Reload Timer 1
TMCSRHn	TMCSRH0	TMCSRH1
TMCSRLn	TMRL0	TMRL1
TMRHn	TMRH0	TMRH1
TMRLn	TMRL0	TMRL1
TMRLRHn	TMRLRH0	TMRLRH1
TMRLRLn	TMRLRL0	TMRLRL1

5. 16-bit I/O-Timer Configuration

The number of available Free Running timers, Input Capture Units and Output Compare Units differs between different devices. Multiple Input Capture Units and Output Compare Units are connected to one Free Running timer. This chapter describes the relation between these modules.

■ Connection between Input Capture Units and Free Running Timers

Table 5-1 Connection between Input Capture Units and Free Running Timers

Free Running Timer	Input Capture Unit
FRT 0	ICU 0/1
FRT 1	ICU 4/5

6. Input Capture Unit Source Select for LIN-USART

The input source for the Input Capture units can be selected between an external pin (INn) and the LIN-USART Sync Field output. This chapter describes how to select the Input capture input and which LIN-USART can be connected to which ICU.

■ Overview of Connection between LIN-USART and ICU

Table 6-1 Connection between LIN-USARTs and ICUs

LIN-USART	ICU	Free running timer for ICU	ICU source select bit
USART0	ICU0	FRT0	ICE01:ICUS0
USART1	ICU1		ICE01:ICUS1

Note:

For availability of the required USART, Input capture unit and Free running timer channels please refer for the datasheet of your device.

■ Input Capture Unit Source Select

The source selection for ICU is done in the Input capture edge registers ICExy.

● Input Capture Edge Register ICE01

ICE01								
bit	15	14	13	12	11	10	9	8
	-	-	-	ICUS1	-	ICUS0	IEI1	IEI0
Attribute	-	-	-	R/W	-	R/W	R/W	R/W
Initial value	X	X	X	0	X	0	0	0

[bit15 to bit13] -: Reserved Bit

Read value is undefined, write always 0.

Read modify write operations to this register have no effect on these bits.

[bit12] ICUS1: Input Capture Source Select for ICU1

Bit	Description
0	The external pin IN1 is input source
1	USART1 is input source

[bit11] -: Reserved Bit

Read value is undefined, write always 0.

Read modify write operations to this register have no effect on this bit.

[bit10] ICUS0: Input Capture Source Select for ICU0

Bit	Description
0	The external pin IN0 is input source
1	USART0 is input source

[bit9] IEI1: Input Capture Valid Edge Indication Bit for ICU1

Bit	Description
0	Falling edge detected
1	Rising edge detected

[bit8] IEI0: Input Capture Valid Edge Indication Bit for ICU0

Bit	Description
0	Falling edge detected
1	Rising edge detected

● Input Capture Edge Register ICE45

ICE01

bit	15	14	13	12	11	10	9	8
	-	-	-	-	-	-	IEI5	IEI4
Attribute	-	-	-	-	-	-	R/W	R/W
Initial value	X	X	X	X	X	X	0	0

[bit15 to bit10] -: Reserved Bit

Read value is undefined, write always 0.

Read modify write operations to this register have no effect on these bits.

[bit9] IEI5: Input Capture Valid Edge Indication Bit for ICU5

Bit	Description
0	Falling edge detected
1	Rising edge detected

[bit8] IEI4: Input Capture Valid Edge Indication Bit for ICU4

Bit	Description
0	Falling edge detected
1	Rising edge detected

7. Peripheral Resource Pin Relocation

The input or output pin of some resources can be relocated. The location of these resource pins is defined by the peripheral resource pin relocation registers PRRR0 to PRRR13.

■ Overview of the Peripheral Resource Relocation Register

Table 7-1 Peripheral Resource Pin Relocation Register (PRRR0 to PRRR13)

Address	Name	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0	Description
0004D6	PRRR0	-	-	-	-	INT3_R	INT2_R	INT1_R	-	Peripheral resource relocation register 0
0004D7	PRRR1	-	-	-	-	-	-	-	-	Peripheral resource relocation register 1
0004D8	PRRR2	-	-	-	-	PPG3_R	PPG2_R	PPG1_R	PPG0_R	Peripheral resource relocation register 2
0004D9	PRRR3	-	-	TOT2_R	TIN2_R	TOT1_R	TIN1_R	-	-	Peripheral resource relocation register 3
0004DA	PRRR4	-	-	IN5_R	IN4_R	-	-	IN1_R	IN0_R	Peripheral resource relocation register 4
0004DB	PRRR5	-	-	-	-	-	-	-	-	Peripheral resource relocation register 5
0004DC	PRRR6	-	-	-	-	-	FRCK0_R	-	-	Peripheral resource relocation register 6
0004DD	PRRR7	-	-	-	-	-	-	-	-	Peripheral resource relocation register 7
0004DE	PRRR8	-	-	-	-	-	-	-	-	Peripheral resource relocation register 8
0004DF	PRRR9	-	-	CKOT0_R	-	-	-	-	-	Peripheral resource relocation register 9
000660	PRRR10	-	-	-	-	-	-	-	-	Peripheral resource relocation register 10
000661	PRRR11	-	-	-	-	-	-	-	-	Peripheral resource relocation register 11
000662	PRRR12	-	-	-	-	-	-	-	-	Peripheral resource relocation register 12
000663	PRRR13	-	-	-	WOT_R	-	-	-	-	Peripheral resource relocation register 13

-: Reserved Bit
Read value is undefined, write always 0.
Read modify write operations to this register have no effect on these bits.

● Availability of Resource Relocation Pins

If a resource relocation pin is not available on the device, the corresponding bit must be treated as undefined bit. Hence the read value of this bit is undefined, "0" must always be written to this bit and read modify write instructions on the register are not affected by this bit.
See the data sheet for the availability of resource relocation pins.

■ Peripheral Resource Relocation Register 0 (PRRR0)

PRRR0								
bit	7	6	5	4	3	2	1	0
	-	-	-	-	INT3_R	INT2_R	INT1_R	-
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

[bit7 to bit4] -: Reserved Bit

Read value is undefined, write always 0.

Read modify write operations to this register have no effect on these bits.

[bit3] INT3_R: External Interrupt 3 Input Relocation Bit

Bit	Description
0	Pin INT3 is used as input pin
1	Pin INT3_R is used as input pin

[bit2] INT2_R: External Interrupt 2 Input Relocation Bit

Bit	Description
0	Pin INT2 is used as input pin
1	Pin INT2_R is used as input pin

[bit1] INT1_R: External Interrupt 1 Input Relocation Bit

Bit	Description
0	Pin INT1 is used as input pin
1	Pin INT1_R is used as input pin

[bit0] INT0_R: External Interrupt 0 Input Relocation Bit

Read value is undefined, write always 0.

Read modify write operations to this register have no effect on this bit.

■ Peripheral Resource Relocation Register 2 (PRRR2)

PRRR2								
bit	7	6	5	4	3	2	1	0
	-	-	-	-	PPG3_R	PPG2_R	PPG1_R	PPG0_R
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

[bit7 to bit4] -: Reserved Bit

Read value is undefined, write always 0.

Read modify write operations to this register have no effect on these bits.

[bit3] PPG3_R: PPG 3 Output Relocation Bit

Bit	Description
0	Pin PPG3 is used as output pin
1	Pin PPG3_R is used as output pin

[bit2] PPG2_R: PPG 2 Output Relocation Bit

Bit	Description
0	Pin PPG2 is used as output pin
1	Pin PPG2_R is used as output pin

[bit1] PPG1_R: PPG 1 Output Relocation Bit

Bit	Description
0	Pin PPG1 is used as output pin
1	Pin PPG1_R is used as output pin

[bit0] PPG0_R: PPG 0 Output Relocation Bit

Bit	Description
0	Pin PPG0 is used as output pin
1	Pin PPG0_R is used as output pin

■ Peripheral Resource Relocation Register 3 (PRRR3)

PRRR3								
bit	15	14	13	12	11	10	9	8
	-	-	TOT2_R	TIN2_R	TOT1_R	TIN1_R	-	-
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

[bit15, bit14] -: Reserved Bit

Read value is undefined, write always 0.

Read modify write operations to this register have no effect on these bits.

[bit13] TOT2_R: Reload Timer 2 Output Relocation Bit

Bit	Description
0	Pin TOT2 is used as output pin
1	Pin TOT2_R is used as output pin

[bit12] TIN2_R: Reload Timer 2 Input Relocation Bit

Bit	Description
0	Pin TIN2 is used as input pin
1	Pin TIN2_R is used as input pin

[bit11] TOT1_R: Reload Timer 1 Output Relocation Bit

Bit	Description
0	Pin TOT1 is used as output pin
1	Pin TOT1_R is used as output pin

[bit10] TIN1_R: Reload Timer 1 Input Relocation Bit

Bit	Description
0	Pin TIN1 is used as input pin
1	Pin TIN1_R is used as input pin

[bit9, bit8] -: Reserved Bit

Read value is undefined, write always 0.

Read modify write operations to this register have no effect on these bits.

■ Peripheral Resource Relocation Register 4 (PRRR4)

PRRR4								
bit	7	6	5	4	3	2	1	0
	-	-	IN5_R	IN4_R	-	-	IN1_R	IN0_R
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

[bit7, bit6] -: Reserved Bit

Read value is undefined, write always 0.
Read modify write operations to this register have no effect on these bits.

[bit5] IN5_R: Input Capture 5 Input Relocation Bit

Bit	Description
0	Pin IN5 is used as input pin
1	Pin IN5_R is used as input pin

[bit4] IN4_R: Input Capture 4 Input Relocation Bit

Bit	Description
0	Pin IN4 is used as input pin
1	Pin IN4_R is used as input pin

[bit3, bit2] -: Reserved Bit

Read value is undefined, write always 0.
Read modify write operations to this register have no effect on these bits.

[bit1] IN1_R: Input Capture 1 Input Relocation Bit

Bit	Description
0	Pin IN1 is used as input pin
1	Pin IN1_R is used as input pin

[bit0] IN0_R: Input Capture 0 Input Relocation Bit

Bit	Description
0	Pin IN0 is used as input pin
1	Pin IN0_R is used as input pin

■ Peripheral Resource Relocation Register 6 (PRRR6)

PRRR6								
bit	7	6	5	4	3	2	1	0
	-	-	-	-	-	FRCK0_R	-	-
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

[bit7 to bit3] -: Reserved Bit

Read value is undefined, write always 0.

Read modify write operations to this register have no effect on these bits.

[bit2] FRCK0_R: Free Running Timer 0 Input Relocation Bit

Bit	Description
0	Pin FRCK0 is used as input pin
1	Pin FRCK0_R is used as input pin

[bit1, bit0] -: Reserved Bit

Read value is undefined, write always 0.

Read modify write operations to this register have no effect on these bits.

■ Peripheral Resource Relocation Register 9 (PRRR9)

PRRR9								
bit	15	14	13	12	11	10	9	8
	-	-	CKOT0_R	-	-	-	-	-
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	X	X	0	0	0	0	0	0

[bit15, bit14] -: Reserved Bit

Read value is undefined, write always 0.

Read modify write operations to this register have no effect on these bits.

[bit13] CKOT0_R: Clock Output 0 Relocation Bit

Bit	Description
0	Pin CKOT0 is used as output pin
1	Pin CKOT0_R is used as output pin

[bit12 to bit8] -: Reserved Bit

Read value is undefined, write always 0.

Read modify write operations to this register have no effect on these bits.

■ Peripheral Resource Relocation Register 13 (PRRR13)

PRRR13								
bit	15	14	13	12	11	10	9	8
	-	-	-	WOT_R	-	-	-	-
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	X	X	X	0	X	X	0	0

[bit15 to bit13] -: Reserved Bit

Read value is undefined, write always 0.

Read modify write operations to this register have no effect on these bits.

[bit12] WOT_R: Real Time Clock Output Relocation Bit

Bit	Description
0	Pin WOT is used as output pin
1	Pin WOT_R is used as output pin

[bit11 to bit8] -: Reserved Bit

Read value is undefined, write always 0.

Read modify write operations to this register have no effect on these bits.

CHAPTER: CPU

This chapter explains the CPU.

1. Overview
2. Hardware Structure
3. Memory Space
4. Special Registers
5. General-Purpose Registers
6. Prefix Codes

Management Code: 96F6CPU-E01.0

1. Overview

The F^{2}MC-16FX CPU core is a 16-bit CPU designed for applications that require high-speed real-time processing, such as home-use or vehicle-mounted electronic appliances. The F^{2}MC-16FX instruction set is designed for controller applications, and is capable of high-speed, highly efficient control processing.

■ Outline of the CPU

In addition to 16-bit data, the F^{2}MC-16FX CPU core can process 32-bit data by using an internal 32-bit accumulator. 32-bit data can be processed with some instructions. Up to 16 MBytes of memory space can be used, which can be accessed by either the linear pointer or bank method. The instruction set is compatible to F^{2}MC-16LX. The instruction set is compatible with high-level languages, has a rich set of addressing modes, multiplication and division instructions, and bit processing. The features of the F^{2}MC-16FX CPU are explained below.

● Fast Execution Speed

- Minimum instruction execution time: 31.2 ns (at 32 MHz; 4 MHz oscillation, 8 times clock multiplication)
- Basic instructions are executed in one cycle
- High speed processing using a 5 stage pipeline
- 8 byte instruction queue

● General Purpose Registers: 32 banks × 8 words × 16-bit

● Memory Space: 16 MBytes, Accessed in Linear or Bank Method

● Instruction Set Optimized for Controller Applications

- High code efficiency
- Rich data types: Bit, byte, word, long word
- Extended addressing modes: 23 types
- High-precision operation (32-bit length) based on 32-bit accumulator
- Signed and unsigned multiplication and division instructions

● Powerful Interrupt Functions

- Fast response speed (about 10 clock cycles CLKB)
- Eight priority levels (programmable)
- Non maskable interrupt (NMI)
- DMA transfer can serve interrupt requests (16 channels max.) without involving CPU

● Instruction Set Compatible with High-Level Language (C)/multitasking

- System stack pointer
- Instruction set symmetry
- Shift instructions

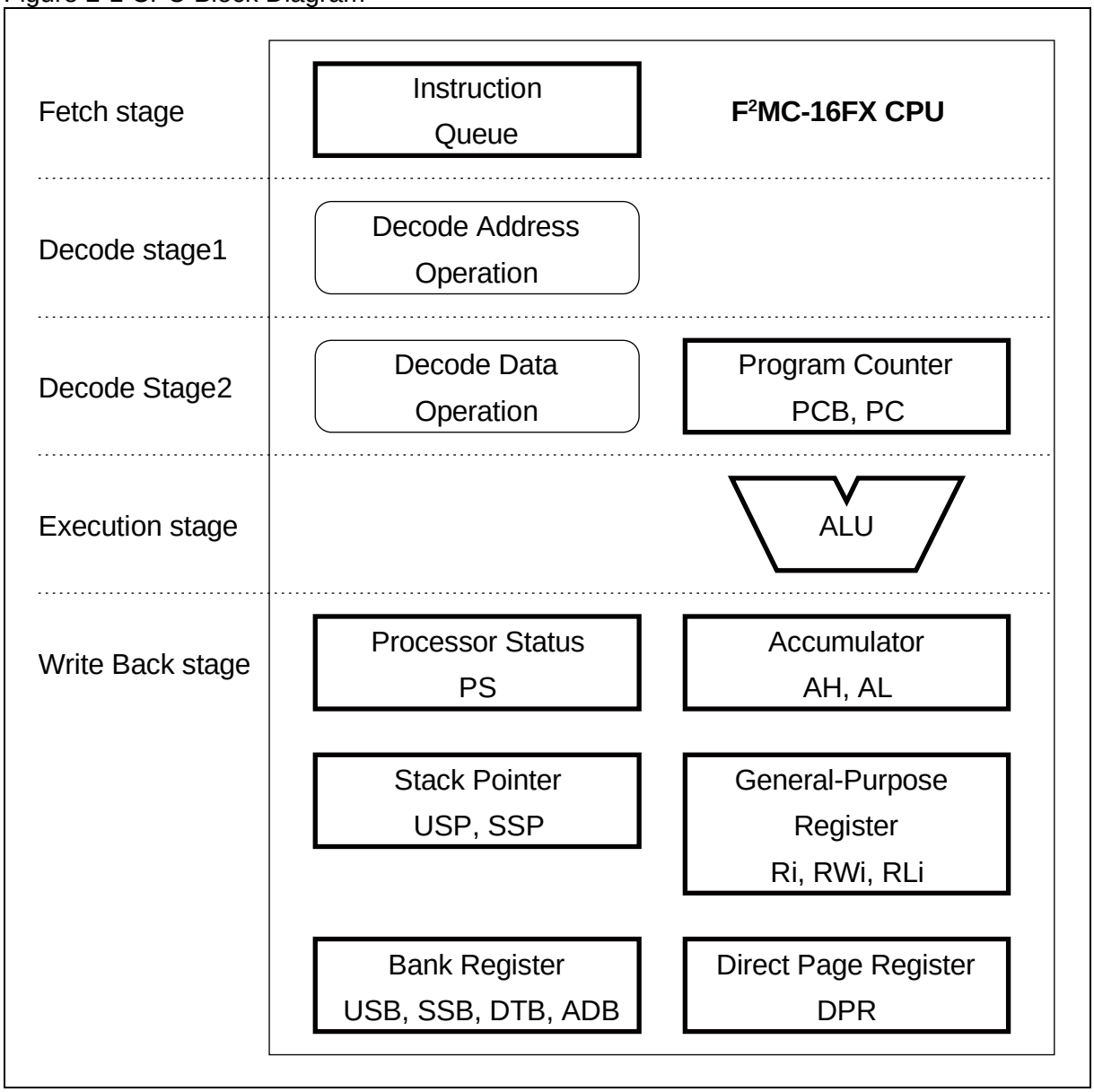
2. Hardware Structure

This section explains the hardware structure of the CPU and the 16FX core.

■ Hardware Structure of the CPU

● CPU Block Diagram

Figure 2-1 CPU Block Diagram

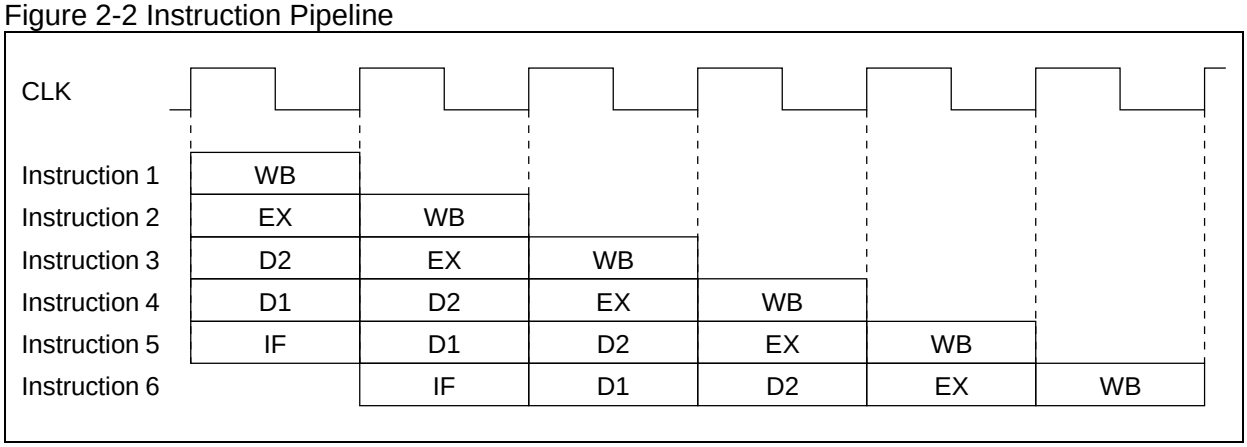


● CPU Pipeline Operation

To execute most instructions in one clock cycle, the CPU uses a five-stage instruction pipeline. The pipeline consists of the following stages:

- Instruction fetch (IF) : Fetches the instruction from instruction queue.
- Instruction decode 1 (D1) : Decodes the instruction and controls address operation.
- Instruction decode 2 (D2) : Decodes the instruction and selects operands and data operation.
- Execution (EX) : Executes the operation.
- Write back (WB) : Writes the operation result to a register or memory location.

Figure 2-2 Instruction Pipeline



Instructions are not executed out of order. Therefore, if instruction A enters the pipeline ahead of instruction B, instruction A always reaches write back stage before instruction B.
The standard instruction execution speed is one instruction per cycle. However, transfer instructions that involve memory wait, branch instructions and multi-cycle instructions require more than one cycle to execute. The instruction execution speed also drops if the delivery of instructions during code fetch is slow.

● Instruction Queue

The CPU has an instruction queue of 8 byte.
The instruction queue is filled by the fetch unit. Prefetch is used on consecutive addresses for code fetch. The prefetch mechanism removes drawbacks due to the latency of the pipelined implementation of the CPU and the system bus of the 16FX core.

● Program Counter

The program counter bank (PCB, upper 8-bit of the program address) and the program counter (PC, lower 16-bit of the program address) are controlled by the decode stage 1.
The 24-bit address of the concatenation of {PCB, PC} points to the instruction, which is executed next.

● ALU

The ALU is controlled by decode stage 2. The operation mode of the ALU is selected and the operands are loaded. The execution of the operation is performed in the next cycle.
The ALU is used for logical and arithmetical operations. Multiplication and division are included.

● CPU Registers and Memory Access

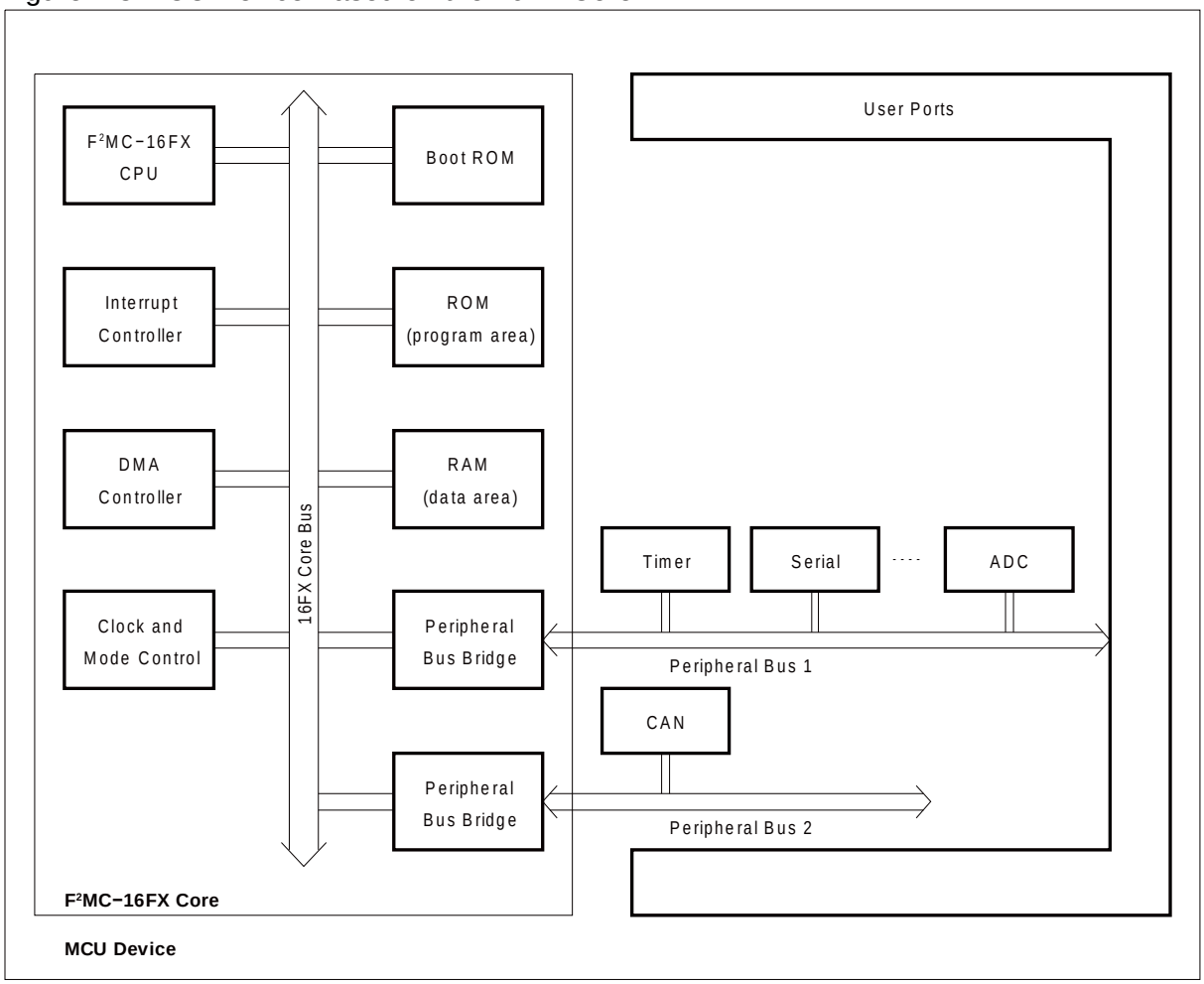
In the write back stage the result of the operation is written to CPU registers and/or to a memory location. All CPU registers accesses except the program counter are assigned to the last pipeline stage.

■ Hardware Structure of the 16FX Core

● Block Diagram

A sample configuration and the principle structure of a MCU device based on the 16FX Core is shown in Figure 2-3.

Figure 2-3 MCU Device Based on the 16FX Core



● Interrupt Controller

The interrupt controller evaluates the priority of incoming interrupt requests (IRQ) and selects the interrupt number with the highest priority. If accepted, the selected interrupt service is processed by the CPU. Each hardware IRQ has its own interrupt level register to control its priority.

● DMA Controller

The DMA controller can also serve IRQs, but without interrupting the actual program execution of the CPU. This can be used to automate data transfer between peripherals and memory.
Depending on the device, up to 16 DMA channels are usable. Each DMA channel can select an IRQ number to be served.

● Clock and Mode Control

This unit has control over the operation mode and monitors correct operation of the device. It supplies all units with their appropriate clock, depending on the operation mode.

● Boot ROM

After device initialization by reset, the program counter points to the boot ROM. The CPU starts the execution of the boot ROM program. After further device initialization the reset vector is fetched and the boot ROM code branches to user program execution starting at the reset vector.

● Peripheral Bus Bridge

The peripheral bus bridge acts as an interface between the system bus of the 16FX core and the peripheral bus connecting to all other MCU internal peripheral resources.
The peripheral bus bridge synchronizes between core clock and peripheral clock domains.

3. Memory Space

An F^{2}MC-16FX CPU has a 16-Mbyte memory space.

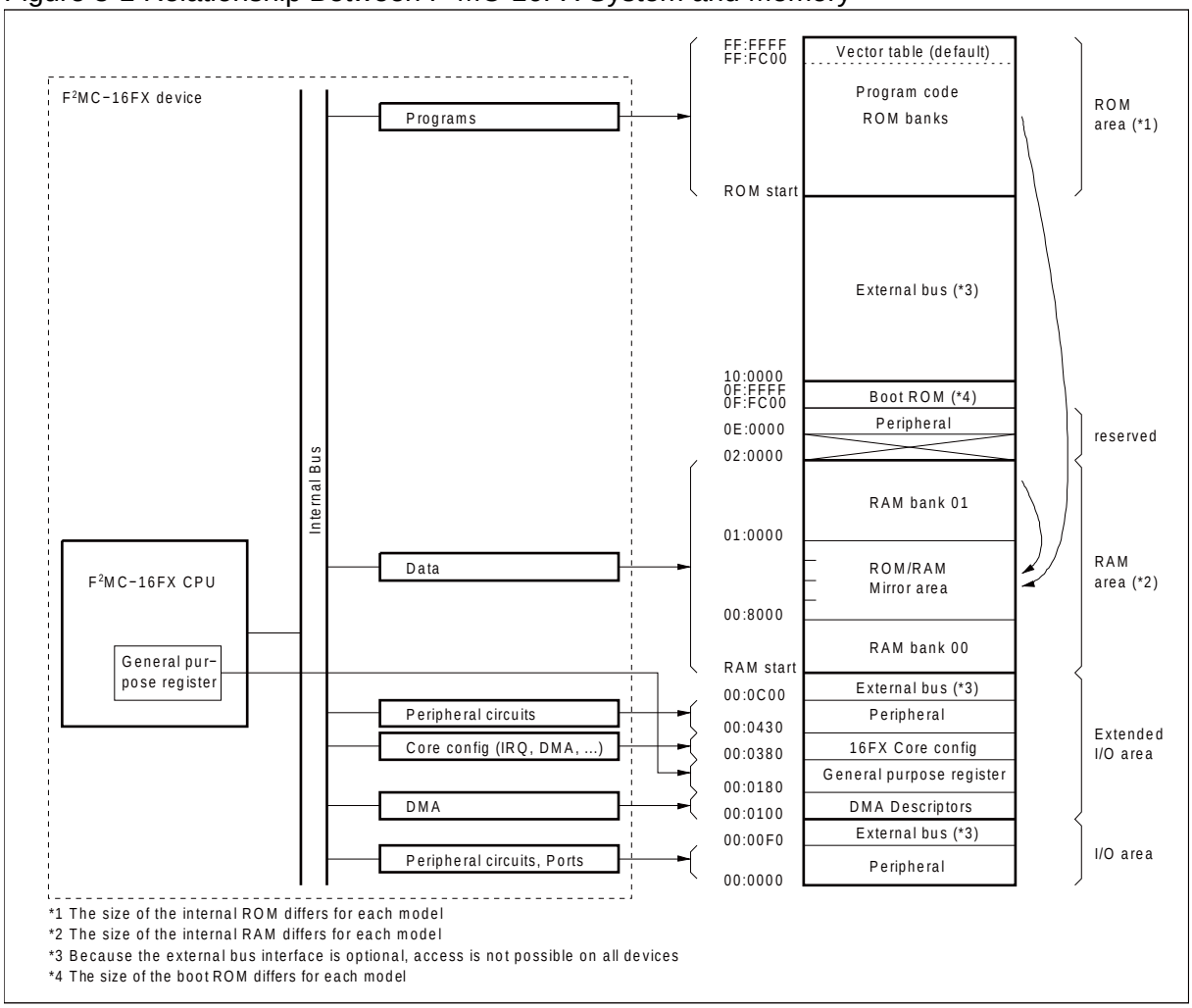
3.1. Memory Areas

All I/O addresses, programs and data are located in the 16-Mbyte memory space of the F²MC-16FX CPU. The CPU is able to access each resource through an address indicated by the 24-bit address bus.

■ Memory Areas in the F²MC-16FX System

Figure 3-1 shows a sample relationship between the F²MC-16FX system and memory map.

Figure 3-1 Relationship Between F²MC-16FX System and Memory



■ I/O Area

The address range of the I/O area is from $00:0000_H$ to $00:00FF_H$. This address range can be used for direct I/O addressing by specifying the 8-bit address as operand together with the instruction. If an instruction is using the direct I/O addressing method, the register is accessed regardless of the values specified by the direct page register (DPR) or the data bank register (DTB).

● Peripheral Function Control Register I/O Area (Address: $00:0000_H$ to $00:00EF_H$)

These registers control the built-in peripheral functions and ports.

■ Extended I/O Area

● DMA Descriptor Area (Address: 00:0100$_H$ to 00:017F$_H$)

This area retains the transfer modes, I/O addresses, transfer count, and buffer addresses.
This area can be used as ordinary RAM.

● General-Purpose Register Area (Address: 00:0180$_H$ to 00:037F$_H$)

General-purpose registers (GPRs) used for 8-bit, 16-bit and 32-bit arithmetic operations and transfers are mapped into this area. This area can be used as ordinary RAM.
Though GPRs are memory mapped, direct access by instructions using these registers is greatly improved with respect to 16LX.

● 16FX Core Control Register (Address: 00:0380$_H$ to 00:042F$_H$)

This area is for CPU core internal configuration registers.

● Peripheral Function Control Register Extended I/O Area (Address: 00:0430$_H$ to 00:0BFF$_H$ and 0E:0000$_H$ to 0E:FFFF$_H$)

Registers in this area control the built-in peripheral functions.

■ RAM Area

● Data Area in Bank 00 (Address: 00:1000$_H$ to 00:7FFF$_H$)

At this area static RAM is implemented with a size up to 28 kByte.
The size of internal RAM differs for each device. Thus the RAM start address depends on the implemented memory size.

● Mirror Area in Bank 00 (Address: 00:8000$_H$ to 00:FFFF$_H$)

This area is used to access the top-most section of 32 kbyte ROM by default. The ROM mirror function is variable in size. There are up to 4 segments of 8 kbyte each selectable as ROM mirror. The ROM bank to be accessed via bank 00 can be selected in the ROMM register.
If the ROM mirror function is switched off, RAM on bank 01 can be accessed via bank 00 in addition.
The mirror area in bank 00 can be used in order to support the small model of the C compiler. Since the least significant 16-bit are identical, the corresponding part for data access to the selected ROM or RAM bank can be referenced without using the far specification in the pointer declaration.

● Data Area in Bank 1 (Address: 01:0000$_H$ to 01:FFFF$_H$)

At this area static RAM is implemented with a size up to 64 kbyte.
The size of internal RAM differs for each device.

■ ROM Area

● Program Area (Address: Up to FF:FFFF$_H$)

ROM is built in as an internal program area.
The size and the ROM start address of internal ROM differs for each device.
By default, addresses FF:FC00$_H$ to FF:FFFF$_H$ are allocated by the vector table.

■ Vector Table Area

● Vector Table Area

This area is used as the vector table for the interrupt vectors and the reset vector. The start address of the corresponding processing routine is set as data in each vector table entry.
The area of the vector table can be changed by the user program in the table base register (TBR). It can be located either in the RAM, ROM or external area. This area is located at 1 kbyte starting at the address defined by the TBR. The address of the vector table area is

VEC_{START} = TBR × 100_H up to
VEC_{END} = TBR × 100_H + $3FF_H$.

The address of the vector table area is FF:$FC00_H$ to FF:$FFFF_H$ by default.
The vector table for vectored call instruction is located in the actual program counter bank specified by the PCB register. The CALLV instruction uses these 32 byte area, located at the top of the program counter bank from PCB:$FFE0_H$ to PCB:$FFFF_H$. The TBR has no influence on the location of the vectors used by the CALLV instruction.
Care must be taken, if the interrupt vector area (upper 8 entries of the interrupt vector table entries) is located at the top of the program counter bank and the CALLV instruction is used. Both vector tables then share the same location.

3.2. Linear Addressing Method

At linear addressing an entire 24-bit address is specified by an instruction. There are two types of linear addressing:

- 24-bit operand specification: Directly specifies a 24-bit address using operands.
- 32-bit register indirect specification: Indirectly specifies the 24 low-order bits of a 32-bit general-purpose register value as the address.

■ 24-bit Operand Specification

Figure 3-2 shows an example of 24-bit operand specification. Figure 3-3 shows an example of 32-bit register indirect specification.

Figure 3-2 Example of Linear Method (24-bit Register Operand Specification)

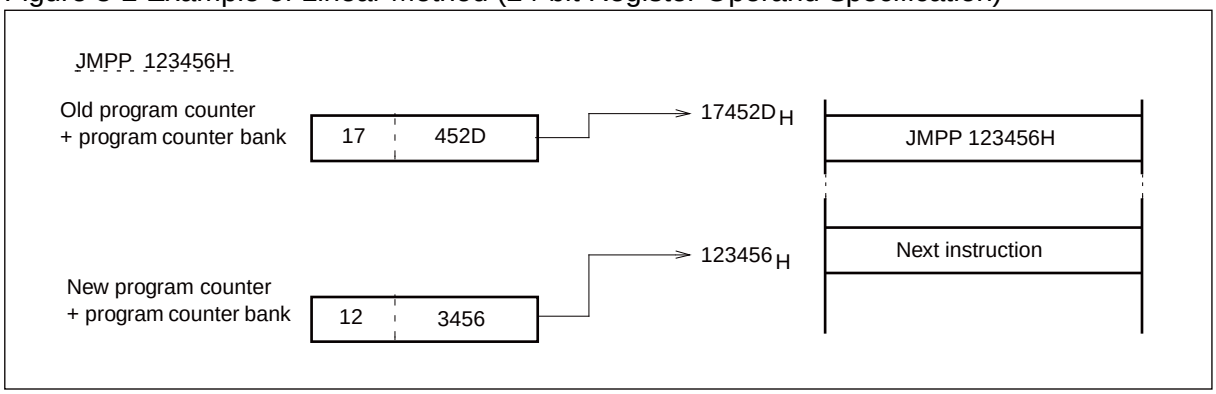
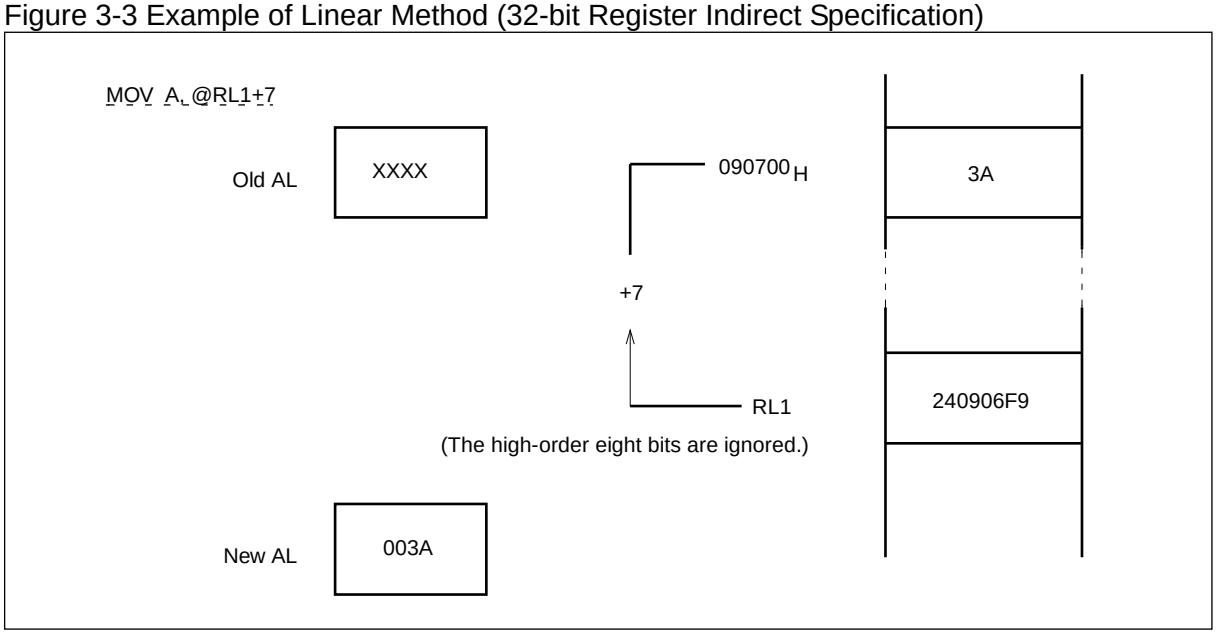


Figure 3-3 Example of Linear Method (32-bit Register Indirect Specification)



3.3. Bank Addressing Method

In the bank addressing method the eight high-order bits of an address are specified by an appropriate bank register, and the remaining 16 low-order bits are specified by an instruction.

■ Bank Addressing Method

In bank addressing, the 16 MByte space is divided into 256 banks (64 kBytes each). The five bank registers described below are used to specify the banks (upper 8 address bits) corresponding to each space:

● Program Counter Bank Register (PCB)

The 64 kByte bank specified by the PCB is called program counter (PC) space. The PC space contains instruction codes, the vector table of the CALLV instruction and immediate value data, for example.

● Data Bank Register (DTB)

The 64 Kbyte bank specified by the DTB is called data (DT) space. The DT space contains readable/writable data, and control/data registers for internal and external resources.

● User Stack Bank Register (USB) and System Stack Bank Register (SSB)

The 64 kByte bank specified by the USB or SSB is called stack (SP) space. The SP space is accessed when a stack access occurs during a push/pop instruction or interrupt register saving. The S flag in the condition code register determines the stack space USB or SSB to be accessed.

● Additional Data Bank Register (ADB)

The 64 kByte bank specified by the ADB is called additional data (AD) space. The AD space, for example, contains data that cannot fit into the DT space.
Table 3-1 lists the default spaces used in each addressing mode, which are pre-determined to improve instruction coding efficiency.

Table 3-1 Default Space

Default space	Addressing mode
Program space	PC indirect, program access or branch
Data space	Addressing mode using @RW0, @RW1, @RW4, @RW5, @A, addr16 or dir
Stack space	Addressing mode using PUSHW, POPW, @RW3 or @RW7
Additional data space	Addressing mode using @RW2 or @RW6

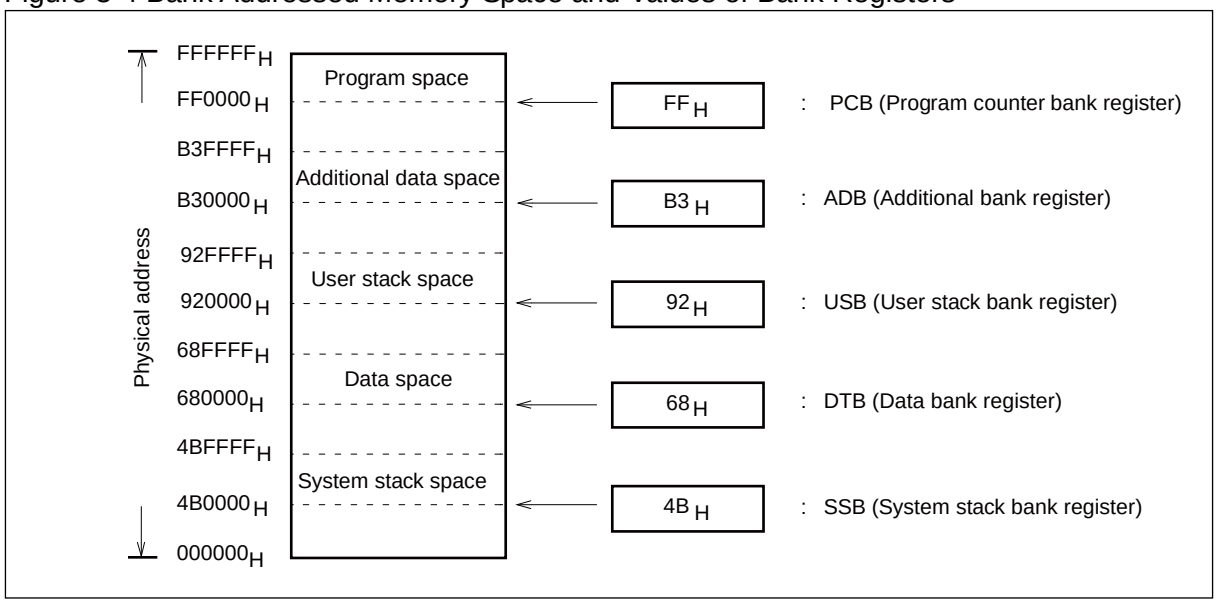
To use a non-default space for an addressing mode, specify a prefix code corresponding to a bank before the instruction. This enables access to the bank space corresponding to the specified prefix code.

Table 3-2 Bank Selection Prefixes

Bank selection prefix	Selected Space
PCB	Program space
DTB	Data space
ADB	Additional data space
SPB	Either the system or the user stack space is used, according to the stack flag status upon selection.

Figure 3-4 is an example of a memory space divided into register banks and shows the physical address of each space.

Figure 3-4 Bank Addressed Memory Space and Values of Bank Registers



■ Initialization of Bank Registers

After reset, the DTB, USB, SSB and ADB are initialized to 00_H. Because the reset address is fixed to the Boot ROM program start address 0F:FC00$_H$, the PCB is initialized to $0F_H$.

In external vector mode a value specified by the reset vector on address FF:FFDC$_H$ is loaded when leaving the Boot ROM code execution. In internal vector mode the PCB is loaded with a fixed value defined by the product.

At starting user program from specified reset vector address, the DT, SP, and AD spaces are allocated in bank 00_H (000000_H to $00FFFF_H$), and the PC space is allocated in the bank specified by the reset vector.

Table 3-3 Initialization of Bank Registers

Bank register	Initialization by reset	Initialization by Boot ROM	
		External vector mode	Internal vector mode
DTB	00_H	00_H	00_H
USB	00_H	00_H	00_H
SSB	00_H	00_H	00_H
ADB	00_H	00_H	00_H
PCB	$0F_H$	Byte read from FF:$FFDE_H$.	Value defined by the product.

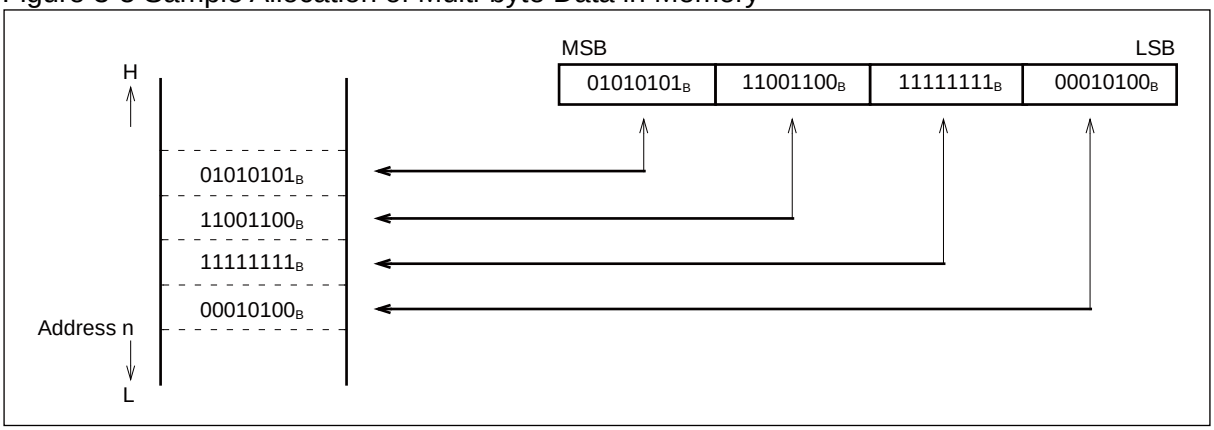
3.4. Multi-byte Data in Memory Space

Data is written to memory from the low-order address on. Therefore, for a 32-bit data item, the low-order 16-bit are transferred before the high-order 16-bit.
If a reset signal is input immediately after the low-order bits are written, the high-order bits might not be written.

■ Multi-byte Data Allocation in Memory Space

Figure 3-5 is a diagram of multi-byte data configuration in memory. The low-order eight bits of a data item are stored at address n, then next eight bits are stored at address n+1, etc.

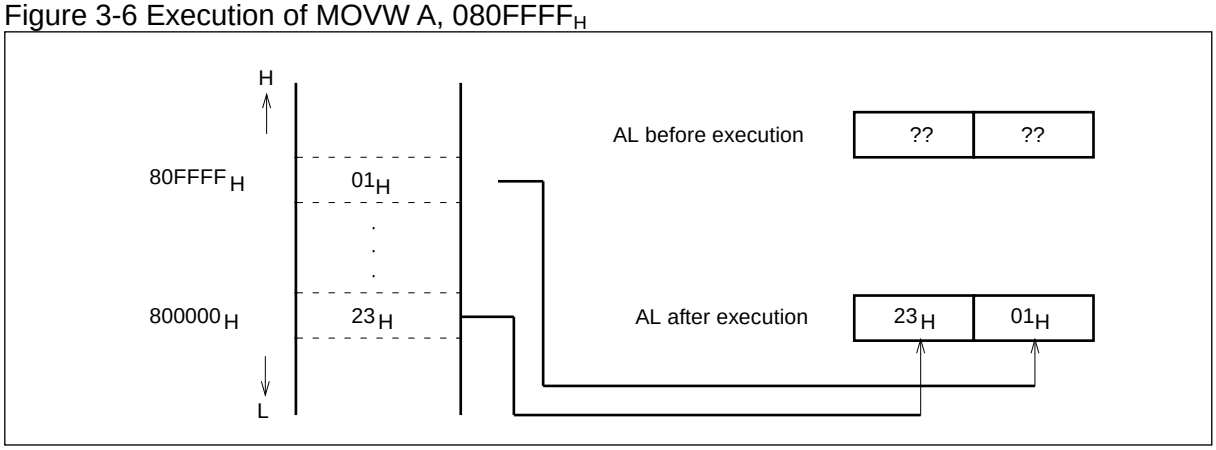
Figure 3-5 Sample Allocation of Multi-byte Data in Memory



■ Accessing Multi-byte Data

Fundamentally, accesses are made within a bank. For an instruction accessing a multi-byte data item, address $FFFF_H$ is followed by address 0000_H of the same bank. Figure 3-6 is an example of an instruction accessing multi-byte data.

Figure 3-6 Execution of MOVW A, $080FFFF_H$



4. Special Registers

The F^{2}MC-16FX CPU registers are classified into two types: special registers and general-purpose registers.
This section explains the special registers of the F^{2}MC-16FX CPU. The special registers are dedicated internal hardware of the CPU, and they have specific use defined by the CPU architecture. These registers can be accessed without using an address. The register operations are defined by specific instructions.

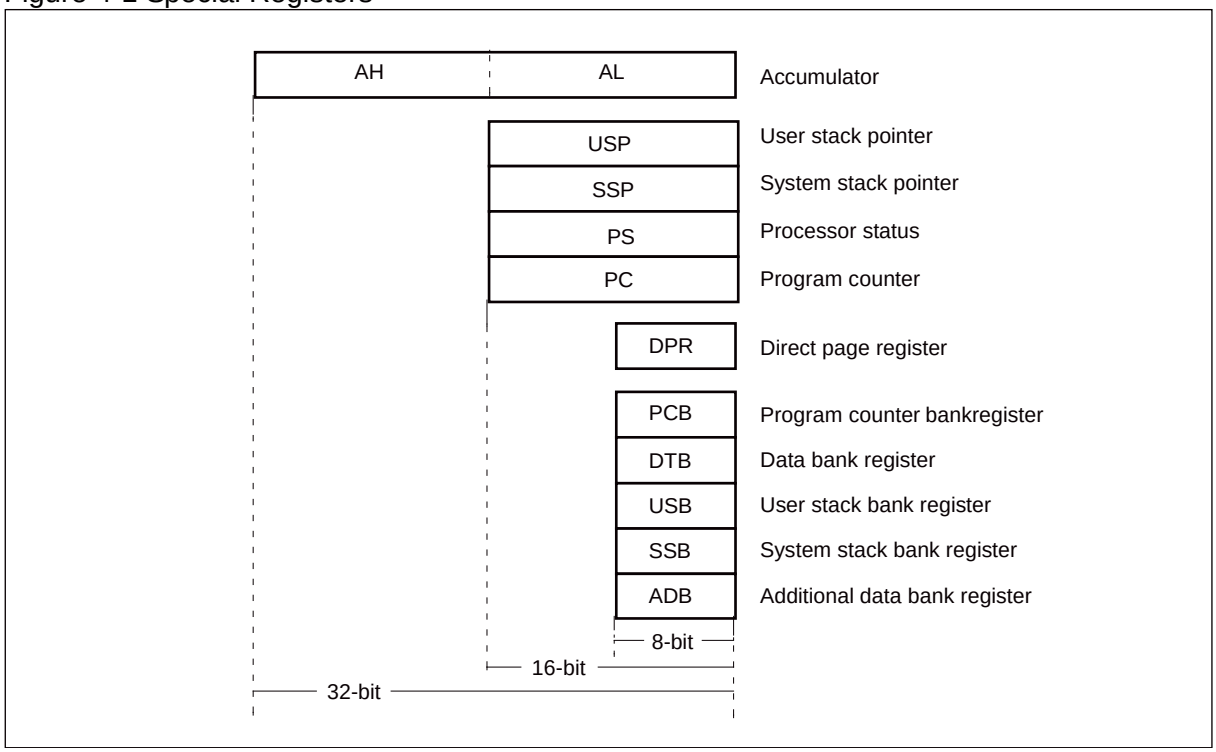
■ Special Registers

The F^{2}MC-16FX CPU has the following special registers:

- Accumulator (A=AH:AL): Two 16-bit accumulators (can be used as a single 32-bit accumulator)
- User stack pointer (USP): 16-bit user stack pointer
- System stack pointer (SSP): 16-bit system stack pointer
- Processor status (PS): 16-bit register indicating the system status
- Program counter (PC): 16-bit register holding the address of the next instruction to be executed
- Program counter bank register (PCB): 8-bit register indicating the program counter bank
- Data bank register (DTB): 8-bit register indicating the data bank
- User stack bank register (USB): 8-bit register indicating the user stack bank
- System stack bank register (SSB): 8-bit register indicating the system stack bank
- Additional data bank register (ADB): 8-bit register indicating the additional data bank
- Direct page register (DPR): 8-bit register indicating the page for direct access

Figure 4-1 is a diagram of the special registers.

Figure 4-1 Special Registers



4.1. Accumulator (A)

The accumulator (A) register consists of two 16-bit arithmetic operation registers (AH and AL), and is used as a temporary storage for operation results and transfer data.

■ Accumulator (A)

The A register consists of two 16-bit arithmetic operation registers (AH and AL). The A register is used as a temporary storage for operation results and transfer data. During 32-bit data processing, AH and AL are used together. Only AL is used for word processing in 16-bit data processing mode or for byte processing in 8-bit data processing mode (see Figure 4-2 and Figure 4-3). The data stored in the A register can be operated upon with the data in memory or registers (Ri, Rwi, or Rli). In the same manner as with the F^2MC-8L, when a word or shorter data item is transferred to AL, the previous data item in AL is automatically sent to AH (data preservation function). The data preservation function and operation between AL and AH help to improve processing efficiency.

When a byte or shorter data item is transferred to AL, the data is sign-extended or zero-extended and stored as a 16-bit data item in AL. The data in AL can be handled either as word or byte.

When a byte-processing arithmetic operation instruction is executed on AL, the high-order eight bits of AL before the operation are ignored. After the operation the high-order eight bits become zero.

The A register is not initialized by a reset. The A register holds an undefined value immediately after a reset.

Figure 4-2 Example of a 32-bit Data Transfer

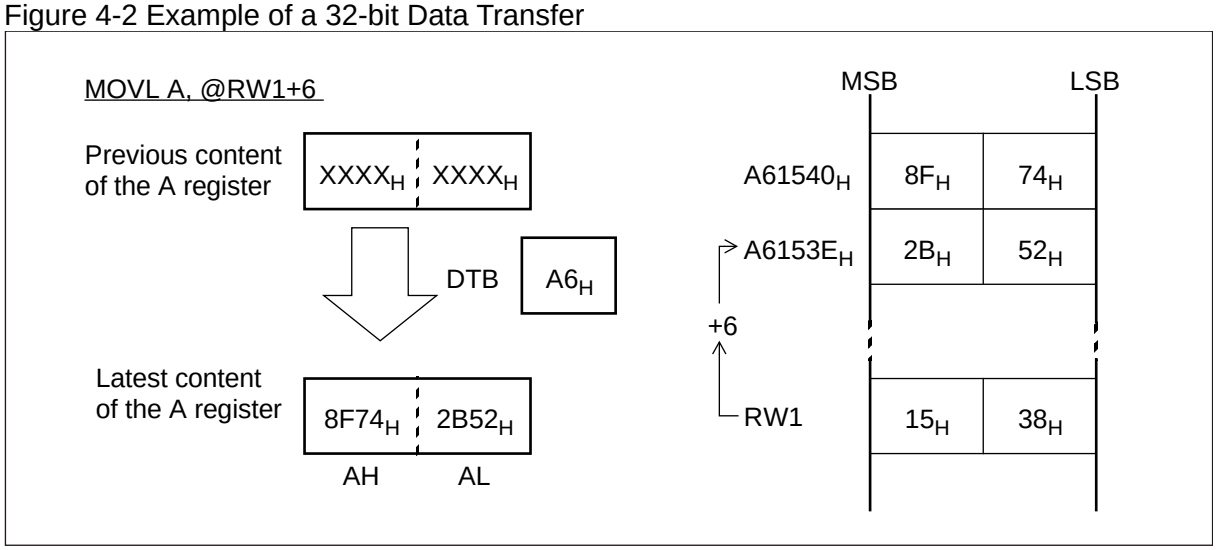
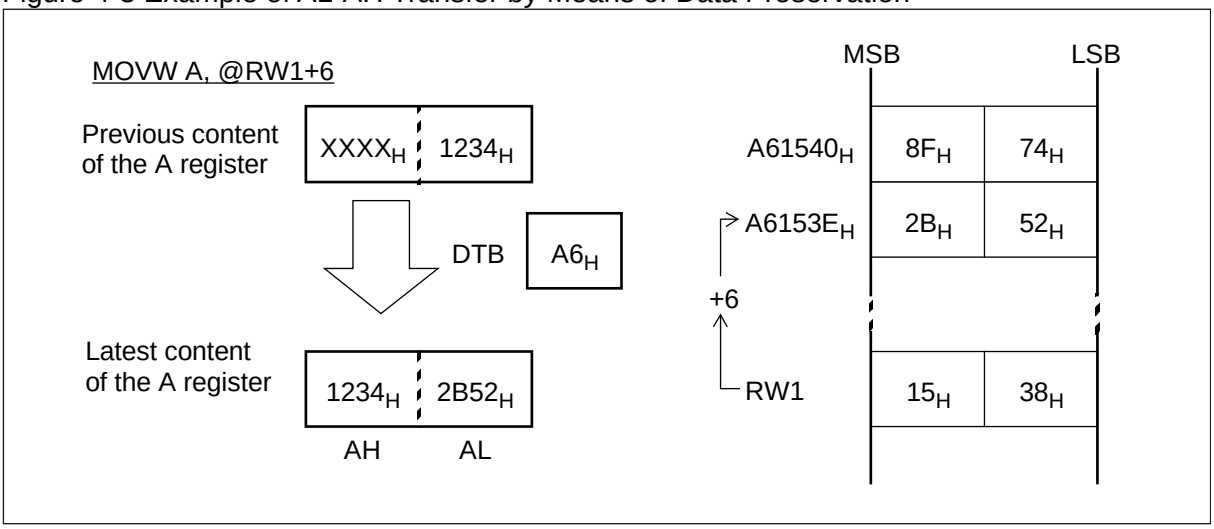


Figure 4-3 Example of AL-AH Transfer by Means of Data Preservation



4.2. User Stack Pointer (USP) and System Stack Pointer (SSP)

USP and SSP are 16-bit registers that indicate the memory addresses for saving and restoring data when a push/pop instruction or subroutine is executed.

■ User Stack Pointer (USP) and System Stack Pointer (SSP)

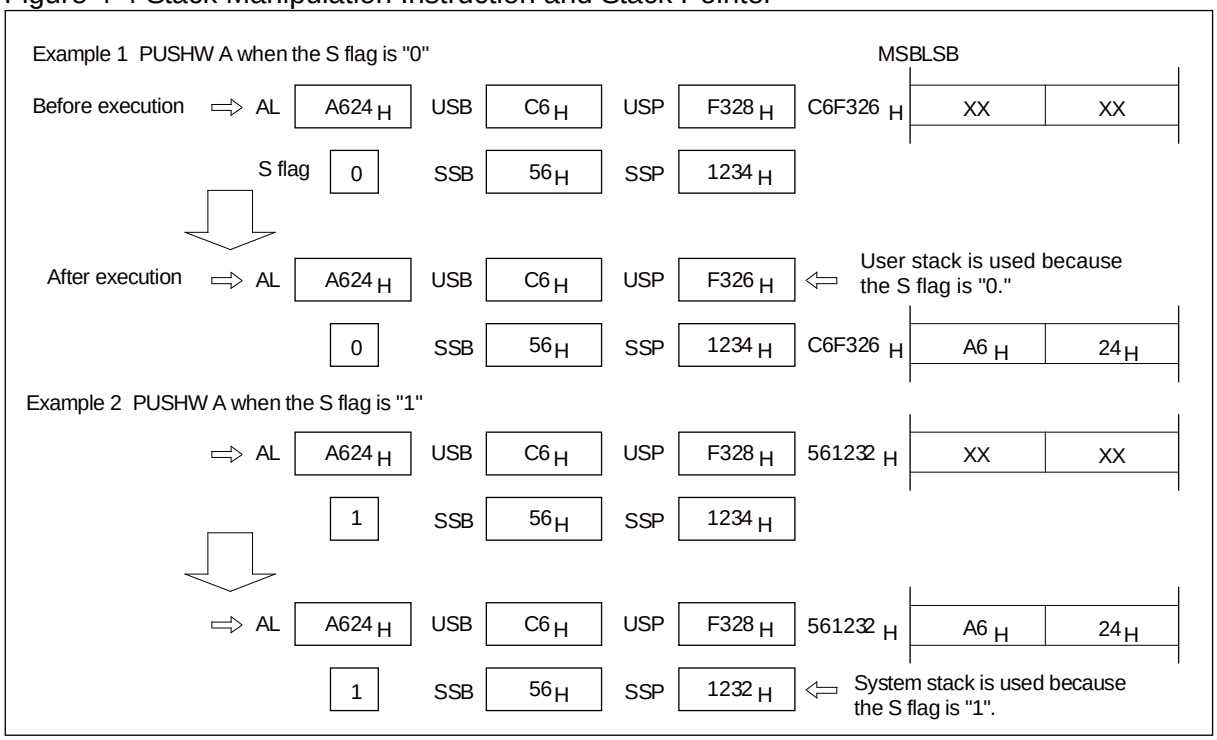
USP and SSP are 16-bit registers that indicate the memory addresses for saving and restoring data in the event of a push/pop instruction or subroutine execution. The USP and SSP registers are used by stack instructions.

The USP register is enabled when the S flag in the processor status register is "0", and the SSP register is enabled when the S flag is "1" (see Figure 4-4). Since the S flag is set when an interrupt is accepted, register values are always saved in the memory area indicated by SSP during interrupt processing. SSP is used for stack processing in an interrupt routine, while USP is used for stack processing outside an interrupt routine. If the stack space is not divided, use only the SSP.

During stack processing, the high-order eight bits of an address are indicated by SSB (for SSP) or USB (for USP).

USP and SSP are not initialized by a reset. Instead, the values are undefined.

Figure 4-4 Stack Manipulation Instruction and Stack Pointer



Note:

Specify an even-numbered address in the stack pointer whenever possible. An odd value will cause drawback in stack performance.

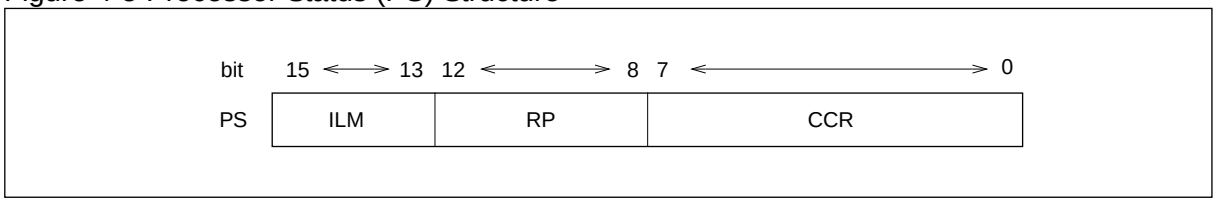
4.3. Processor Status (PS)

The PS register consists of the bits controlling the CPU Operation and indicating the CPU status.

■ Processor Status (PS)

As shown in Figure 4-5, the high-order byte of the PS register consists of a register bank pointer (RP) and an interrupt level mask register (ILM). The RP indicates the start address of a register bank. The low-order byte of the PS register is a condition code register (CCR), containing the flags to be set or reset depending on the results of instruction execution or interrupt occurrences.

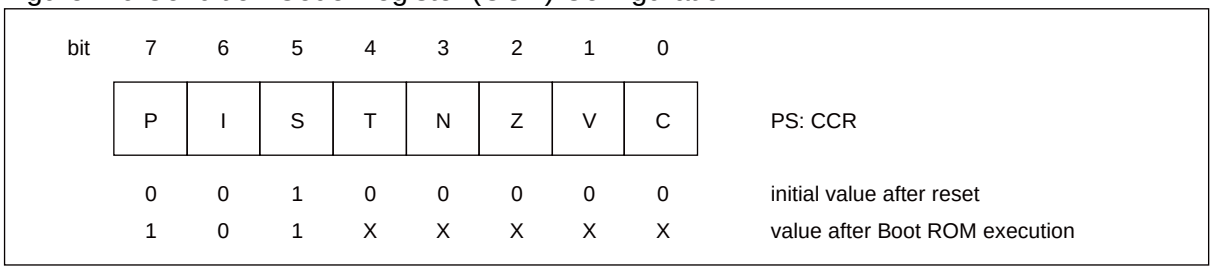
Figure 4-5 Processor Status (PS) Structure



■ Condition Code Register (CCR)

Figure 4-6 is the diagram of the condition code register configuration.

Figure 4-6 Condition Code Register (CCR) Configuration



● P: Priviledged Mode Flag:

P = 1 indicates user mode, P = 0 indicates privileged mode.
The P flag is cleared by a reset. However, the P flag will be set during execution of the Boot ROM code.
Only NMI and DSU interrupts will clear the P flag and disable all other hardware interrupts. If the P flag is cleared, ILM defines system interrupt levels of the privileged mode (P0 to P7). These interrupt levels have higher priority than any ILM setting in user mode (U0 to U7).
The P flag can be set by dedicated instructions (OR CCR #imm, POPW PS) or by restoring the processor status (RETI, JCTX @A). Restoring P=0 is not accepted, if P has been "1" before.

● I: Interrupt Enable Flag:

Interrupts other than software interrupts are enabled when the I flag is "1" and are masked when the I flag is "0". The I flag is cleared by a reset.

● S: Stack Flag:

When the S flag is "0", USP is enabled as the stack pointer.
When the S flag is "1", SSP is enabled as the stack pointer.
The S flag is set by an interrupt reception or a reset.

● T: Sticky Bit Flag:

A value of "1" is set in the T flag when there is at least one "1" in the data shifted out from the carry after execution of a logical right/arithmetic right shift instruction, otherwise, "0" is set in the T flag.
In addition, "0" is set in the T flag when the shift amount is zero.

● N: Negative Flag:

The N flag is set when the MSB of the operation result is "1", and is otherwise cleared.

● Z: Zero Flag:

The Z flag is set when the operation result is all zeroes, and is otherwise cleared.

● V: Overflow Flag:

The V flag is set when an overflow of a signed value occurs as a result of operation execution and is otherwise cleared.

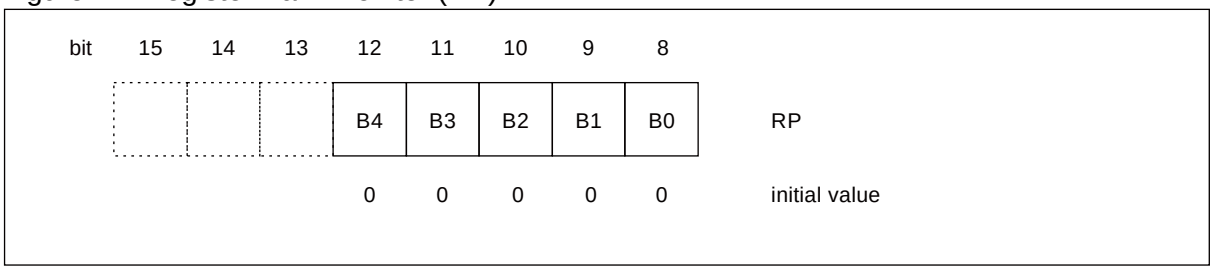
● C: Carry Flag:

The C flag is set when a carry-up or carry-down from the MSB or LSB occurs as a result of operation execution, and is otherwise cleared.

■ Register Bank Pointer (RP)

The RP register indicates the relationship between the general-purpose registers of the F^2MC-16FX and the internal RAM addresses. Specifically, the RP register indicates the first memory address of the currently used register bank in the following conversion expression: [00180_H + (RP) × 10_H]. The RP register consists of five bits, and can take a value between 00_H and $1F_H$. Register banks can be allocated at addresses from 000180_H to $00037F_H$ in memory.

Figure 4-7 Register Bank Pointer (RP)

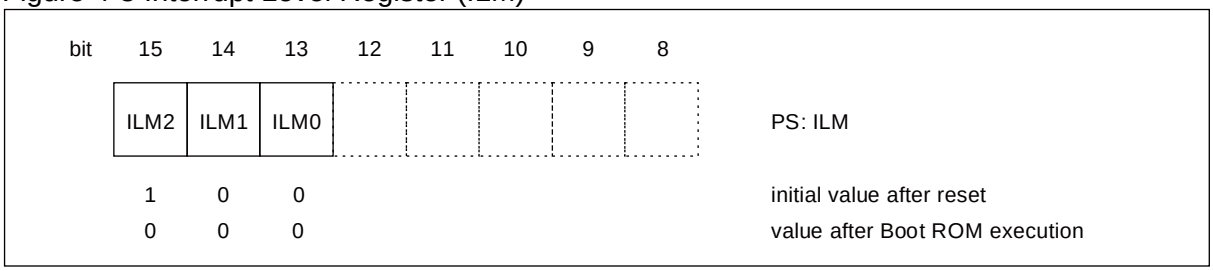


The RP register is initialized to all zeroes by a reset. An instruction may transfer an 8-bit immediate value to the RP register; however, only the low-order five bits of that data are used.

■ Interrupt Level Mask Register (ILM)

The ILM register consists of three bits, indicating the CPU interrupt masking level. An interrupt request is accepted only when the priority of the interrupt is higher than that indicated by the ILM register and the P flag. Highest priority interrupt is level P0 and lowest priority is level U7. Therefore, for an interrupt to be accepted, its level value must be smaller than the current ILM value (see Figure 4-8). In addition, the P flag has to be considered. When an interrupt is accepted, the level value of that interrupt is set in the P flag and ILM register. Thus, an interrupt of the same or lower priority cannot be accepted subsequently.

Figure 4-8 Interrupt Level Register (ILM)



ILM is initialized to 100_B by a reset. However, during execution of the Boot ROM program ILM is set to 000_B.

An instruction may transfer an 8-bit immediate value to the ILM register, but only the low-order three bits of that data are used (MOV ILM #imm8, POPW PS, RETI, JCTX @A).

If P=1 (in user level), any ILM change is possible. If P=0 (priviledged level), an ILM change is only accepted, if the new value defines a user level U0 to U7 (with P=1) or if the privileged level (P0 to P7) is increased. The lower levels of the privileged mode P0 to P7 can not be reached by execution of an instruction from a higher level. Writing 0 to the P flag and reducing the level with P=0 is only possible by NMI or a DSU interrupt.

Note:

The P flag can be understood as bit extension of ILM. Then it defines the most significant bit of the the interrupt level mask {P, ILM}.

After initialization with reset the CPU is in level P4. This disables all interrupts, including NMI, except for the DSU. After execution of the Boot ROM program the CPU is in level U0. Peripheral interrupts are disabled.
All privileged mode levels P0 to P7 are locked against entering or decreasing the level by an instruction. The levels P0 to P7 can only be increased. This protects the operation of NMI and DSU operation. Only DSU can interrupt the NMI or mask its acceptance during a debug session.
The user levels U0 to U7 are backward compatible to F^2MC-16LX interrupt levels 0 to 7. The P flag is not writeable in a user level.

Table 4-1 Levels Indicated by the P Flag and Interrupt Level Mask (ILM) Register

Level	P flag	ILM value	Acceptable interrupt level		
P0	0	0	none	Interrupts disabled	
P1	0	1	Level < P1	DSU	
P2	0	2	Level < P2	DSU	
P3	0	3	Level < P3	DSU	
P4	0	4	Level < P4	DSU	
P5	0	5	Level < P5	NMI, DSU	
P6	0	6	Level < P6	NMI, DSU	
P7	0	7	Level < P7	NMI, DSU	
U0	1	0	Level < U0	User Interrupts disabled	NMI, DSU
U1	1	1	Level < U1	User level 0	
U2	1	2	Level < U2	User level 0-1	
U3	1	3	Level < U3	User level 0-2	
U4	1	4	Level < U4	User level 0-3	
U5	1	5	Level < U5	User level 0-4	
U6	1	6	Level < U6	User level 0-5	
U7	1	7	Level < U7	User level 0-6	

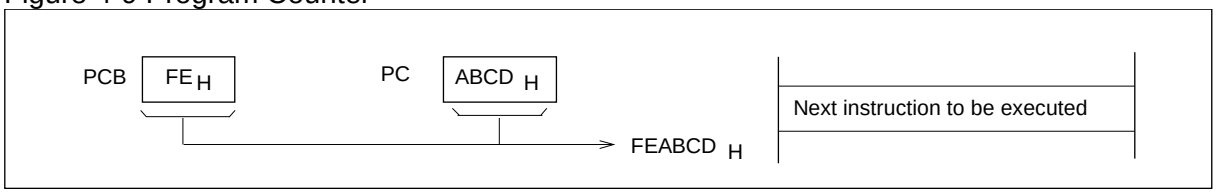
4.4. Program Counter (PC)

The PC register is a 16-bit counter that indicates the low-order 16-bit of the memory address of an instruction code to be executed by the CPU.

■ Program Counter (PC)

The PC register is a 16-bit counter that indicates the low-order 16-bit of the memory address of an instruction code to be executed by the CPU. The high-order eight bits of the address are indicated by the program counter bank register (PCB).
The PC register is updated by a branch instruction, subroutine call instruction, interrupt or reset. Within a linear program segment, the PC is incremented by the number of bytes of the last instruction.
The PC register can also be used as a base pointer for operand access.
Figure 4-9 shows the program counter.

Figure 4-9 Program Counter



The reset address is fixed to the Boot ROM program start address of 0F:$FC00_H$. At reset, the PC is initialized to $FC00_H$ and the PCB is initialized to $0F_H$.
In external vector mode a value specified by the reset vector on address FF:$FFDC_H$ is loaded when leaving the Boot ROM code execution. In internal vector mode the PCB and the PC are loaded with fixed values defined by the product.

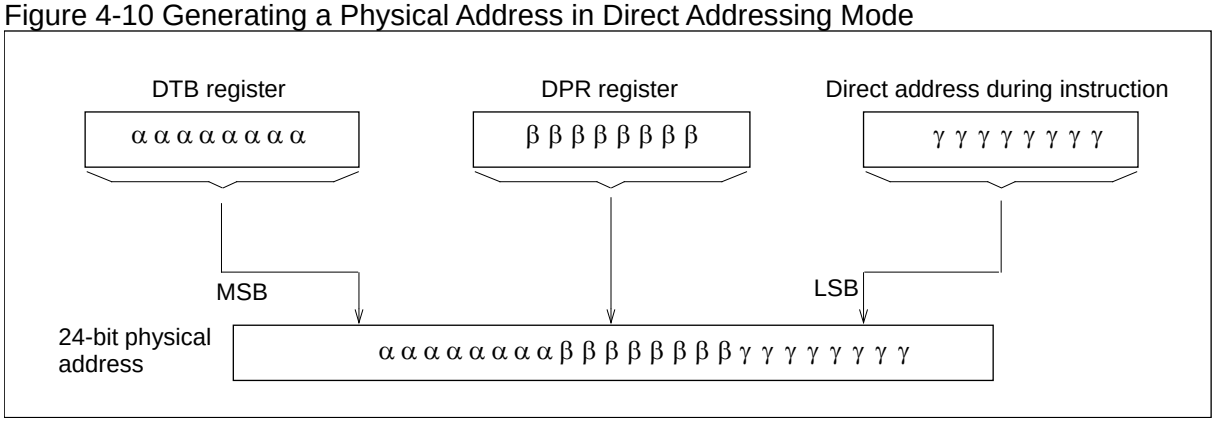
4.5. Direct Page Register (DPR)

The direct page register (DPR) specifies bits 8 to 15 (addr 8 to addr 15) of the operand address for direct addressing instructions.

■ Direct Page Register (DPR)

DPR specifies bits 8 to 15 of the instruction operands in direct addressing mode as shown in Figure 4-10.

Figure 4-10 Generating a Physical Address in Direct Addressing Mode



DPR is eight bits long, and is initialized to 01_H by a reset. DPR can be read or written to by an instruction.

4.6. Bank Register (PCB, DTB, ADB, USB, SSB)

Each bank register indicates a memory bank where a program space, data space, user stack space or additional data space is allocated

■ Bank Register

All bank registers are one byte long. Each bank register (PCB, DTB, USP, SSP, ADB) indicates the memory bank where the PC, DT, SP (user), SP (system), or AD space is allocated.
Bank registers other than PCB can be read and written to. PCB can be read but cannot be written to. The PCB register is updated upon the JMPP or CALLP instruction, branching to the entire 16-Mbyte space, upon the RETP or RETI instruction or upon an interrupt.
For details of the operation of bank registers, see "Section 3.3 Bank Addressing Method".

● Program Counter Bank Register (PCB)

Initial value: $0F_H$ after reset, and later a value from reset vector at user program start.

● Data Bank Register (DTB)

Initial value: 00_H.

● User Stack Bank Register (USB)

Initial value: 00_H.

● System Stack Bank Register (SSB)

Initial value: 00_H.

● Additional Data Bank Register (ADB)

Initial value: 00_H.

5. General-Purpose Registers

The F^{2}MC-16FX CPU registers are classified into two types: special registers and general-purpose registers.
This section explains the general-purpose registers (GPRs) of the F^{2}MC-16FX CPU.
GPRs can be accessed without addressing, similar to the special registers. The register operations are defined by specific instructions. In addition, the GPRs are accessible within the CPU address space. They can be used as ordinary memory.

5.1. Register Bank

A register bank consists of eight words. The register bank can be used as general-purpose registers for arithmetic operations.

■ Register Bank

A register bank consists of eight words. The register bank can be used as the following general-purpose registers for arithmetic operations:

- byte registers R0 to R7 (8-bit)
- word registers RW0 to RW7 (16-bit)
- long word registers RL0 to RL3 (32-bit).

In addition, registers of the register bank can be used as pointers and counters.
Table 5-1 lists the functions of the registers. Table 5-2 indicates the relationship between the registers.

Table 5-1 Register Functions

R0 to R7	Used as operands of instructions. Note: R0 is also used as a counter for shift or normalization instructions.
RW0 to RW7	Used as pointers. Used as operands of instructions. Note: RW0 is used as a counter for string instructions.
RL0 to RL3	Used as long pointers. Used as operands of instructions.

Table 5-2 Relationship Between Registers

Byte register	Word register	Long word registers
	RW0	RL0
	RW1	
	RW2	RL1
	RW3	
R0	RW4	RL2
R1		
R2	RW5	
R3		
R4	RW6	RL3
R5		
R6	RW7	
R7		

In the same manner as for an ordinary RAM area, the register bank values are not initialized by a reset. The status before a reset is maintained. When the power is turned on, the register bank will have an undefined value.

5.2. Addressing General-Purpose Registers

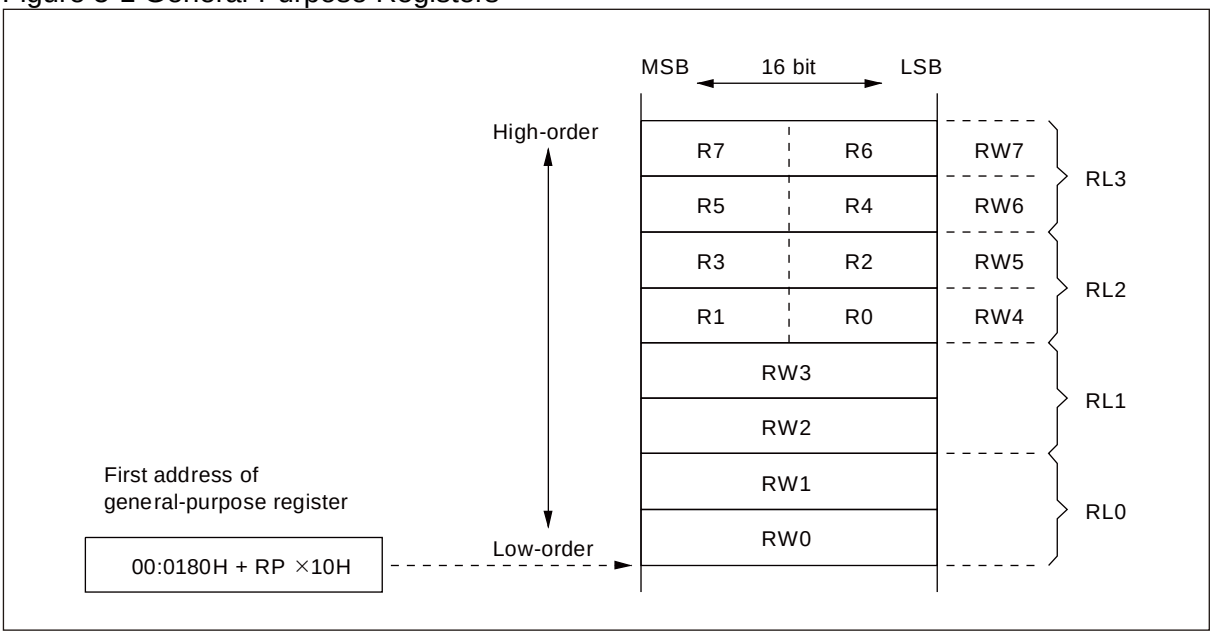
The general-purpose registers of the F^{2}MC-16FX use the register bank pointer (RP) to specify the currently used register bank. The register banks can be addressed between 00:0180$_H$ and 00:037F$_H$ in the memory space.

■ Addressing General-Purpose Registers

The F^{2}MC-16FX general-purpose registers are located from addresses 000180$_H$ to 00037F$_H$ of memory space. The register bank pointer (RP) indicates which of the above addresses are currently being used as a register bank. Each bank has the following three types of registers. These registers are mutually dependent as described in Figure 5-1.

- R0 to R7: 8-bit general-purpose registers
- RW0 to RW7: 16-bit general-purpose registers
- RL0 to RL3: 32-bit general-purpose registers

Figure 5-1 General-Purpose Registers



The relationship between the high-order and low-order bytes of a byte or word register is expressed as follows:

$$RW_{(i+4)} = R_{(i\times2+1)} \times 256 + R_{(i\times2)} \quad [i=0 \text{ to } 3]$$

The relationship between the high-order and low-order bytes of RLi and RW can be expressed as follows:

$$RL_{(i)} = RW_{(i\times2+1)} \times 65536 + RW_{(i\times2)} \quad [i=0 \text{ to } 3]$$

6. Prefix Codes

Placing a prefix code before an instruction partially changes the operation of that instruction. There are three types of prefix codes:

- Bank selection prefix
- Common register bank prefix
- Flag change inhibit prefix

This section explains each prefix.

6.1. Bank Selection Prefix

Placing a bank selection prefix before an instruction enables that instruction to access any specified memory space, regardless of the addressing method being used.

■ Bank Select Prefix

The memory space used for accessing data is determined for each addressing mode.
When a bank select prefix is placed before an instruction, the memory space used for accessing data by that instruction can be selected regardless of the addressing mode.
Table 6-1 lists the bank select prefixes and the corresponding memory spaces.

Table 6-1 Bank Select Prefix

Bank select prefix	Space selected
PCB	PC space
DTB	Data space
ADB	AD space
SPB	Either the SSP or USP space is used according to the stack flag value.

Note, that the effect of the prefixes varies for following instructions:

● Transfer Instructions (I/O Access)

MOV A, io	MOV io, A	MOVW A, io	MOVW io, A
MOV io, #imm8	MOV io, #imm16	MOVX A, io	

The I/O space is accessed, regardless of any bank selection prefix is specified.

● Bit Operation Instructions (I/O Access)

MOVB A, io:bp	MOVB io:bp, A	SETB io:bp	CLRB io:bp
BBC io:bp, rel	BBS io:bp, rel	WBTC io:bp	WBTS io:bp

The I/O space is accessed, regardless of any bank selection prefix is specified.

● Branch Instructions

RETI

The system stack bank (SSB) is used, regardless of any bank selection prefix is specified.

● String Operation Instructions

FILS	FILSW	SCEQ	SCWEQ
MOVS	MOVSW		

The bank register specified by the operand is used, regardless of any bank selection prefix is specified.

● Stack Operation Instructions

PUSHW	POPW	POPW PS
LINK	UNLINK	

The system stack bank (SSB) or user stack bank (USB) is used according to the S flag, regardless of any bank selection prefix is specified.

● Processor Status Word Change / Interrupt Control Instructions

AND CCR, #imm8	OR CCR, #imm8	MOV ILM, #imm8	POPW PS

The specified bank selection prefix affects the next instruction.

6.2. Common Register Bank Prefix (CMR)

The common register bank prefix (CMR) can be placed before an instruction that accesses a register bank. Then the target register of that instruction is changed to the common register bank. This is the register bank selected with RP=0, regardless of the current value of the register bank pointer (RP).

■ Common Register Bank Prefix (CMR)

To simplify data exchange between multiple tasks, it is necessary to establish a relative easy method of accessing the same pre-determined register bank, regardless of the RP value. To achieve this, the F^2MC-16FX has a so called common register bank that can be used by all tasks. The common bank is located between address $00:0180_H$ and $00:018F_H$. It is selected when the RP value is set to "0".
When CMR is placed before an instruction that accesses a register bank, that instruction accesses the common bank, regardless of the current RP value.

Note, that the effect of the prefix varies for the following instructions:

● String Operation Instructions

FILS	FILSW	SCEQ	SCWEQ
MOVS	MOVSW		

Placing a CMR prefix before the string instructions listed above is ignored. The counter register RW0 is always referenced by using the actual value of the register bank pointer RP, regardless of a CMR prefix is specified or not.

● Stack Operation Instructions

LINK	UNLINK

The system stack bank (SSB) or user stack bank (USB) is used according to the S flag, regardless of any bank selection prefix is specified.

● Processor Status Word Change / Interrupt Control Instructions

AND CCR, #imm8	OR CCR, #imm8	MOV ILM, #imm8	POPW PS

The specified prefix affects the next instruction.

6.3. Flag Change Inhibit Prefix (NCC)

Flag changes associated with the execution of an instruction can be inhibited by placing a flag change inhibit prefix code (NCC) before that instruction.

■ Flag Change Inhibit Prefix Code (NCC)

To disable flag changes, use the flag change inhibit prefix code (NCC). Placing NCC before an instruction disables flag changes associated with that instruction.

Note, that the effect of the prefix varies for the following instructions:

● String Operation Instructions

FILS	FILSW	SCEQ	SCWEQ

Placing a NCC prefix before the string instructions listed above is ignored. Flags in the CCR are changing according to the instruction specifications, regardless of a NCC prefix is specified or not.

● Processor Status Word Change / Interrupt Control Instructions

AND CCR, #imm8	OR CCR, #imm8	MOV ILM, #imm8	POPW PS

The specified prefix affects the next instruction.

● Branch Instructions

INT #vct8	INT9	INT addr16	INTP addr24
RETI			

CCR changes according to the instruction specifications regardless of a NCC prefix is specified or not.

● Context Switch Instruction

JCTX @A

CCR changes according to the instruction specifications regardless of the prefix.

6.4. Prefix Code Restrictions

This section lists the instructions, which reject interrupt requests during execution. If a prefix code is placed before such an instruction, the setting of the prefix code remains effective until the first instruction is executed after this interrupt rejecting instruction.
If conflicting prefix codes are specified consecutively, only the last specified prefix code is valid.

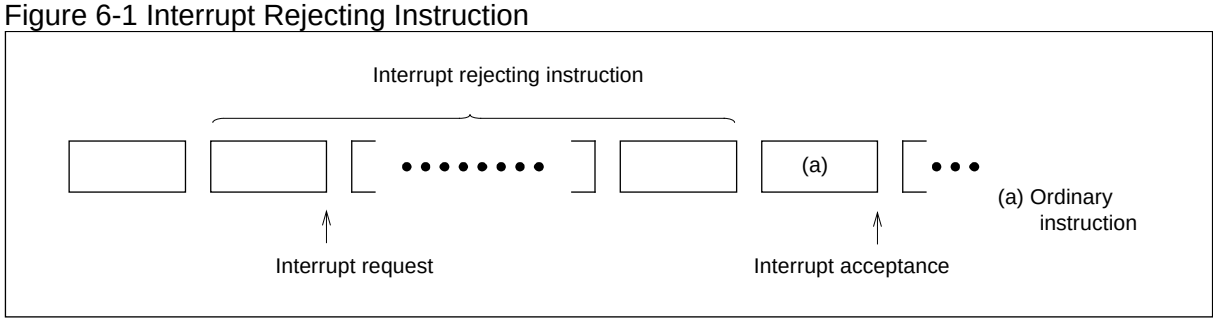
■ Interrupt Rejecting Instructions

Interrupt requests are not accepted during execution of following instructions and prefix codes:

MOV ILM, #imm8	AND CCR, #imm8	OR CCR, #imm8	POPW PS
PCB	ADB	DTB	SPB
NCC	CMR		

If a valid interrupt request occurs during execution of any of the above instructions, the interrupt is postponed until an instruction other than the above is executed. For details, see Figure 6-1.

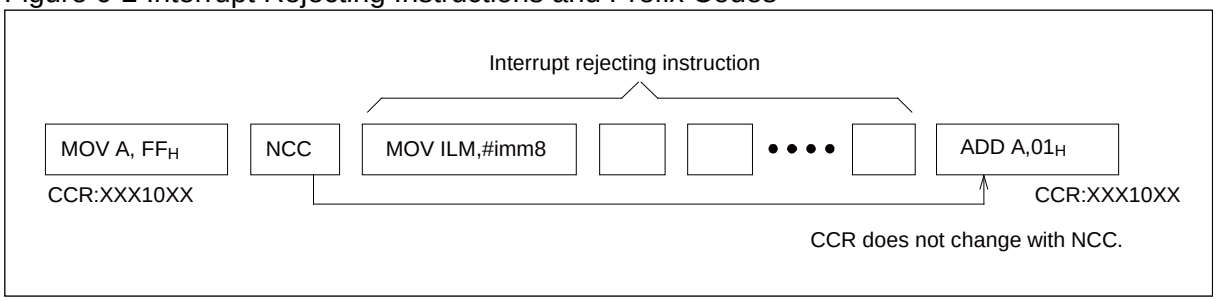
Figure 6-1 Interrupt Rejecting Instruction



■ Restrictions on Interrupt Rejecting Instructions and Prefix Codes

When a prefix code is placed before an interrupt rejecting instruction, the prefix code affects the first instruction after the code other than the interrupt rejecting instruction. For details, see Figure 6-2.

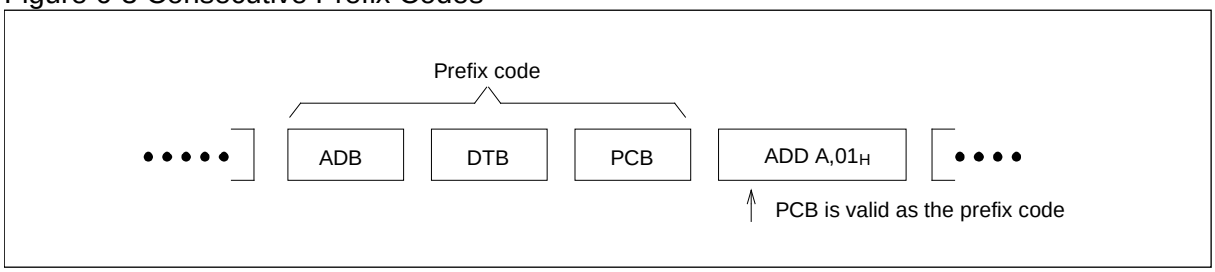
Figure 6-2 Interrupt Rejecting Instructions and Prefix Codes



■ Multiple Prefix Codes in Succession

When conflicting prefix codes are placed consecutively, the last one is valid.
In the figure below, conflicting prefix codes are PCB, ADB, DTB, and SPB. For details, see Figure 6-3.

Figure 6-3 Consecutive Prefix Codes



Prefixes CMR and NCC are not conflicting with bank selection prefixes PCB, ADB, DTB and SPB. CMR is not conflicting with NCC.

CHAPTER: INTERRUPTS

This chapter explains the interrupt functions and operations.

Management Code: 96F6INT-E01.0

1. Overview

The F^{2}MC-16FX has interrupt functions that terminate the currently executed program and transfer control to another specified program when a specific event occurs. There are four types of interrupt functions:

- Hardware interrupt: Interrupt processing due to an internal resource event
- Software interrupt: Interrupt processing due to a software event (instruction)
- Exception: Handling of an operation exception
- DMA: Data transfer without CPU interaction due to an internal resource event.

■ Hardware Interrupts

A hardware interrupt is activated by an interrupt request from an internal resource. A hardware interrupt request occurs when both the interrupt request flag and the interrupt enable flag in an internal resource are set.

● Specifying an Interrupt Level

An interrupt level can be specified for the hardware interrupt. To specify an interrupt level, use the level setting bits (IL0, IL1, and IL2) in the interrupt control register ICR.

For each hardware interrupt its own interrupt level (IL) can be specified. Access to a dedicated IL can be done by setting the index IX. Both IX and IL are accessible through the interrupt control register ICR.

● Masking a Hardware Interrupt Request

A hardware interrupt request can be masked by using the I flag and the ILM bits (ILM0, ILM1, and ILM2). The interrupt is executed only, when the I flag is set and the value of the interrupt level IL is smaller than the interrupt level mask ILM. In addition the P flag has to be set for hardware interrupt acceptance. P, I and ILM are parts of the processor status register PS of the CPU.

When an unmasked interrupt request occurs, the CPU saves 12 bytes of data that consists of registers PS, PC, PCB, DTB, ADB, DPR, and A in the memory area indicated by the system stack bank and pointer registers (SSB and SSP).

■ Software Interrupts

Interrupts requested by executing the INT instruction are software interrupts. An interrupt request by the INT instruction does not have an interrupt request or enable flag. An interrupt request is issued always by executing the INT instruction.

No interrupt level is assigned to the INT instruction. Therefore, ILM is not updated when the INT instruction is used. Instead, the I flag is cleared and the continuing interrupt requests are suspended.

■ Exceptions

Following software exceptions can be processed:

- Undefined instruction
- INT9
- INTE (only available on the EVA device)

Following hardware exceptions can be processed:

- NMI
- DSU break factors (only available on EVA device or if ODCU is available)

Exception processing is basically the same as interrupt processing. When an exception is detected during instruction execution, exception processing is performed. In general, exception processing occurs as a result of an unexpected operation. Therefore, use exception processing only for debugging programs or for activating recovery software in an emergency case.

■ Direct Memory Access (DMA)

● DMA Function

F^2MC-16FX offers a DMA function to automatically transfer data between peripheral resources and memory upon an interrupt.
The number of DMA channels is device dependent.
When a DMA data transfer of a specified count is completed, an interrupt processing program is automatically executed on the original IRQ channel. The handling of such an interrupt by DMA completion is same as for standard type of hardware interrupts.
For a detailed description of DMA, please see "CHAPTER 4. DMA".

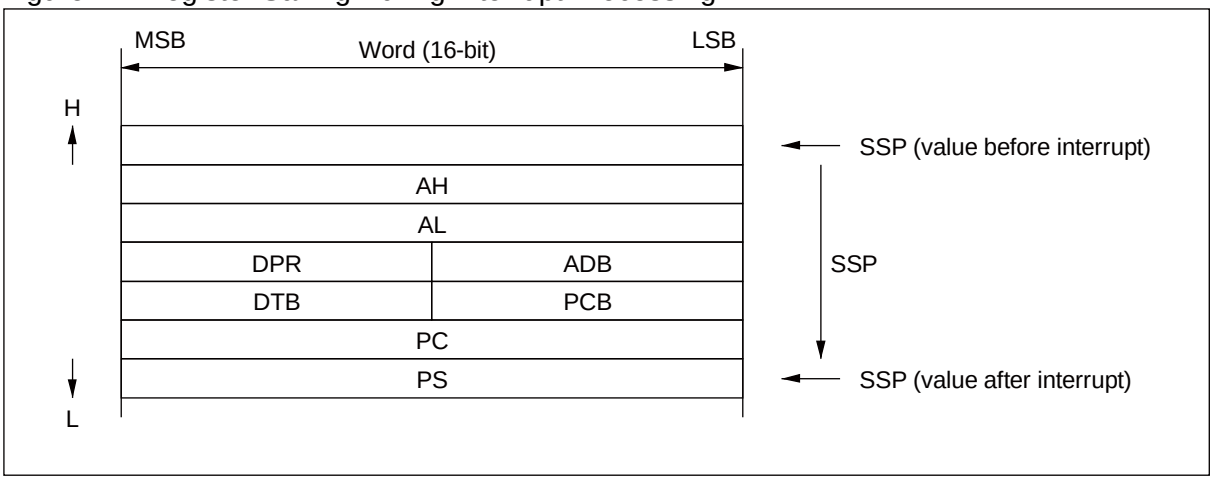
2. Interrupt Flow

Figure 2-2 shows the interrupt flow.

■ Interrupt Flow

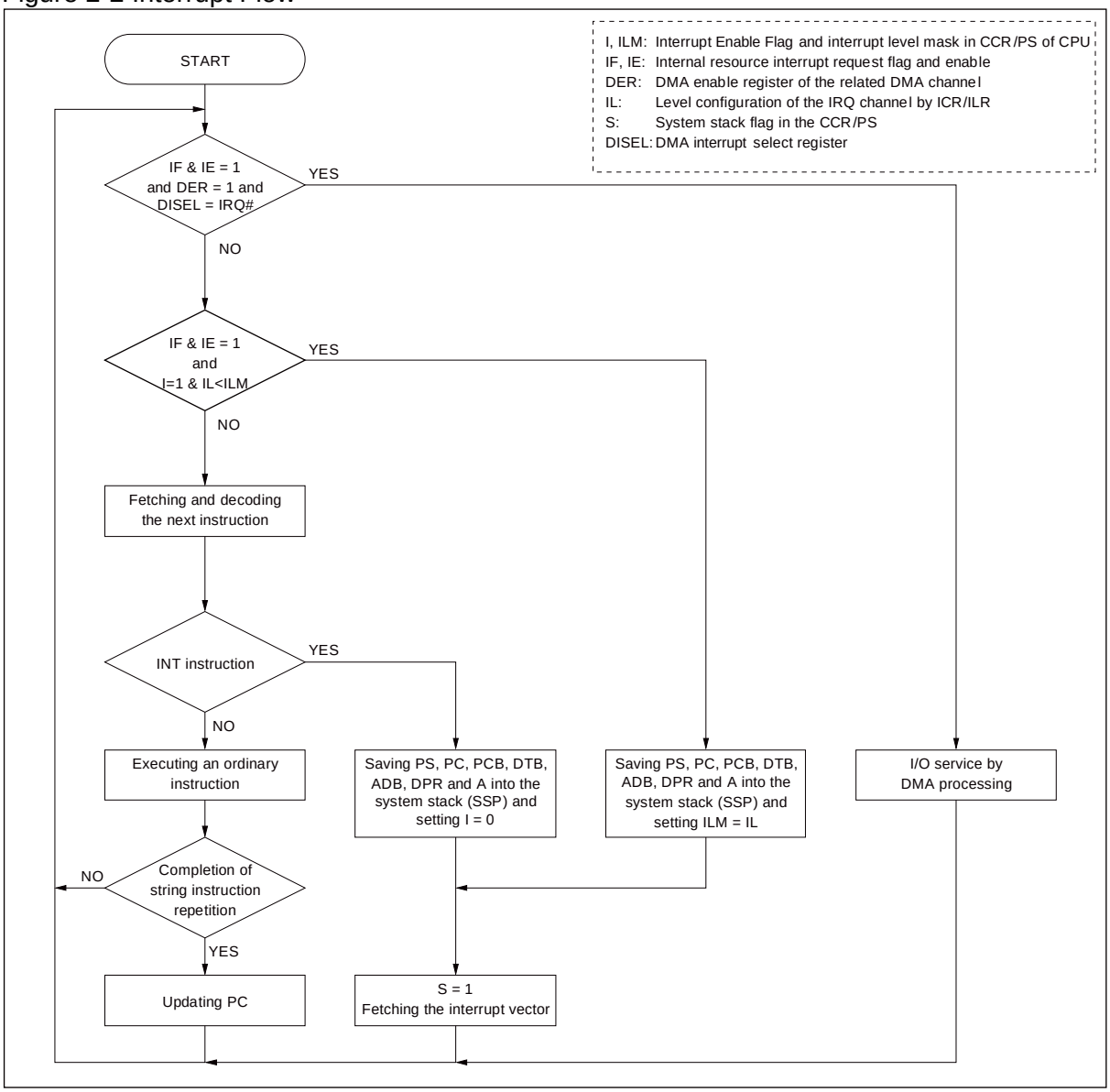
The interrupt processing flow is entered at occurrence of hardware interrupts, software interrupts or exceptions. For a detailed interrupt flow chart see Figure 2-2.
The CPU special registers are saved on the stack before the interrupt is processed (see Figure 2-1).

Figure 2-1 Register Saving During Interrupt Processing



At the end of the interrupt processing, the context of the CPU registers is restored while executing the RETI instruction. The CPU returns to normal program execution.

Figure 2-2 Interrupt Flow



3. Hardware Interrupts

In response to an interrupt request signal from an internal resource, the CPU pauses current program execution and transfers control to the interrupt processing program defined by the user.

■ Hardware Interrupts

A hardware interrupt occurs when the relevant conditions are satisfied as a result of two operations:

- comparison between the interrupt request level (IL) and the value in the interrupt level mask register (ILM) of PS in the CPU.
- See "CHAPTER 2 CPU" reference to the I flag value of PS.

The CPU performs the following processing when a hardware interrupt occurs:

- Saves the values in the PC, PS, AH, AL, PCB, DTB, ADB, and DPR registers of the CPU to the system stack.
- The S flag is set.
- Sets ILM in the PS register. The currently requested interrupt level (IL) is automatically set.
- Reads the corresponding interrupt vector and branches to the processing indicated by that value.

If the device is in standby mode, a hardware interrupt with IL<7 generates a wake-up event to the clock and mode control unit.

■ Structure of the Hardware Interrupt System

The interrupt status is indicated by internal resources, the ICR for the interrupt controller, and the PS value of the CPU. To use a hardware interrupt, make the following set-up:

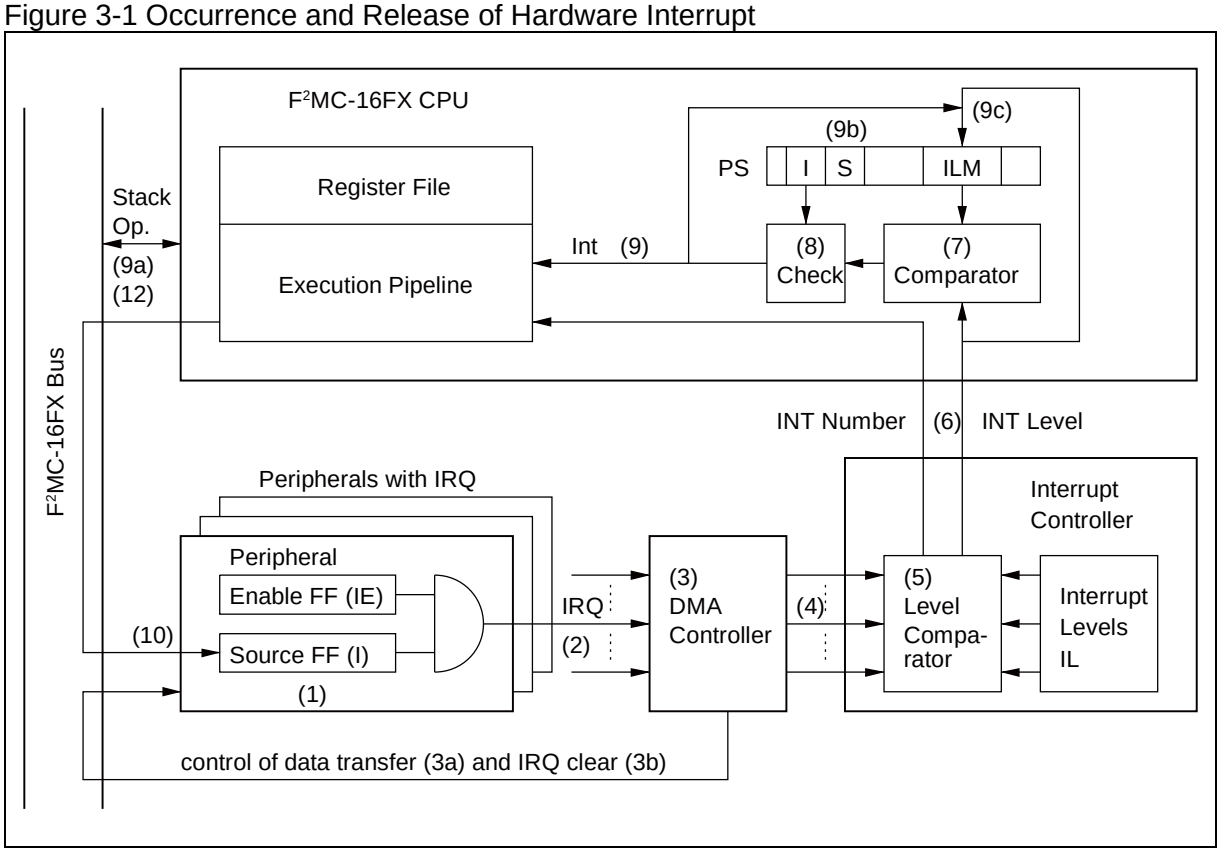
- Interrupt vector (in memory)
  - Consider the TBR value for a non-default location of the vector table.
  - The start address of the interrupt service routine has to be written to the appropriate interrupt vector (VecAddr = 4× (255-INT#) + 256×TBR).
- Peripheral resource
  - Use the Interrupt enable and request bits to control interrupt requests from peripheral resources.
- Interrupt controller
  - Assign interrupt levels (ICR:IL) for each interrupt, which can occur.
  - If interrupts occur simultaneously, a higher priority is defined by lower interrupt levels. IL=7 disables the interrupt.
  - If multiple requests are at the same level, the interrupt controller selects the request with the lowest interrupt number. In the case of same levels configured, the delayed interrupt has the lowest priority, independent from its interrupt number.
  - There is a fixed relationship between the interrupt requests and the ILs. A level can be defined by IL[n] for each hardware interrupt request IRQ[n] (for $n \geqq 12$).
- CPU
  - ILM and I in the PS register are used to compare the requested interrupt level (IL) with the current interrupt level mask (ILM) and to identify the interrupt enable status (I). For acceptance of hardware interrupts, the I flag has to be set and ILM has to be larger than IL.
  - During interrupt processing, the CPU saves 12bytes of data in CPU Special Registers to the memory area indicated by SSB and SSP. Thus the system stack pointer has to be initialized before using interrupts.
  - The CPU fetches three bytes of the interrupt vector and loads them onto PC and PCB. The interrupt handler routine has to start at this location. As a result, the interrupt processing program defined by the user is executed next. Normal operation is resumed at execution of the RETI instruction.

■ Hardware Interrupt Operation

Interrupt requests (IRQs) from peripheral resources are fed through the DMA controller before connecting to the interrupt controller. The DMA controller decides depending on it's channel configuration (DMA interrupt request select register DISEL and DMA enable DER:ENx bit), if the IRQ is handled by DMA transfer or passed to the interrupt controller. DMA transfers are accepted regardless of the status of the I flag and the interrupt level. The DMA controller has a fixed priority scheme, channel "0" has highest priority and channel 15 has lowest priority.

Figure 3-1 shows the processing flow from the occurrence of a hardware interrupt to the release of the interrupt request in an interrupt processing program.

Figure 3-1 Occurrence and Release of Hardware Interrupt



1. An interrupt cause occurs in a peripheral.
2. The interrupt enable bit in the peripheral is referenced. If interrupts are enabled, the peripheral issues an interrupt request (IRQ).
3. The DMA controller checks, if the IRQ should be handled by DMA. It evaluates for each channel, if the interrupt number of the asserted IRQ is selected and if DMA is enabled.
   - If the evaluation is true, the transfer is handled by DMA (3a). If the evaluation is false, interrupt processing is done by the interrupt controller, proceeding with step 4.
   - At the end of the DMA transfer, the interrupt bit is cleared in the peripheral (3b).
   - If the final transfer count is reached, the DMA completion interrupt is processed by the interrupt controller.
4. The interrupt controller receives the interrupt request.
5. The interrupt controller determines the priority levels of simultaneously requested interrupts.
6. The interrupt controller transfers the highest priority interrupt level and the corresponding interrupt number to the CPU.
7. The interrupt level requested by the interrupt controller is compared with the ILM value of the processor status register.
8. If the comparison shows that the requested level is lower than the current interrupt processing level (IL<ILM), the I flag value of the same processor status register is checked.

9. If the check in step 8 shows that the I flag indicates interrupt enable status, interrupt processing is performed as soon as the currently executing instruction is completed.
   · To save the CPU status, special CPU special registers are transferred to the system stack (9a).
   · The S bit is set to "1" (9b).
   · The requested level is written to the ILM bits (9c).
   · The interrupt vector is fetched.
   · Then control is transferred to the interrupt processing routine (branch to the address read as interrupt vector).
10. When the interrupt cause of step 1 is cleared by software in the user interrupt processing routine, the interrupt request is completed.
11. The RETI instruction is used to return from the interrupt processing routine as its last instruction.
12. The CPU status is restored from system stack and normal program execution is resumed.

■ Hardware Interrupt Processing Time

The time required for the CPU to execute the interrupt processing (stack operation, interrupt vector fetch, branch to the interrupt vector) is shown below. The value is valid if stack operation and interrupt vector fetch are executed without any wait cycles.

· Interrupt start: 10 cycles + c
· Interrupt return: 9 cycles + c (RETI instruction)

Table 3-1 Compensation Values (c) for Interrupt Processing Cycle Count

Address indicated by the stack pointer	Compensation value
Internal area, even-numbered address	0
Internal area, odd-numbered address	+2

In addition wait cycles for bus transfers have to be added, if any (e.g. access to vector table in slower ROM memory).

4. Software Interrupts

In response to execution of a special instruction, control is transferred from the program currently executed by the CPU to the interrupt processing program defined by the user. This is called the software interrupt function. A software interrupt occurs always when the software interrupt instruction is executed.

■ Software Interrupts

Software interrupt request issued by the INT instruction has no interrupt request or enable flag. A software interrupt request is always issued and accepted by executing the INT instruction.
The INT instruction does not have an interrupt level. Therefore, the INT instruction does not update ILM.
The INT instruction clears the I flag to suspend subsequent hardware interrupt requests.
The CPU performs the following processing when a software interrupt occurs:

- Saves the values in the PC, PS, AH, AL, PCB, DTB, ADB, and DPR registers of the CPU to the system stack.
- The S flag is set.
- Clears I flag in the PS register. Hardware interrupts are automatically disabled.
- Reads the corresponding interrupt vector and branches to the processing indicated by that value.

■ Structure of the Software Interrupt System

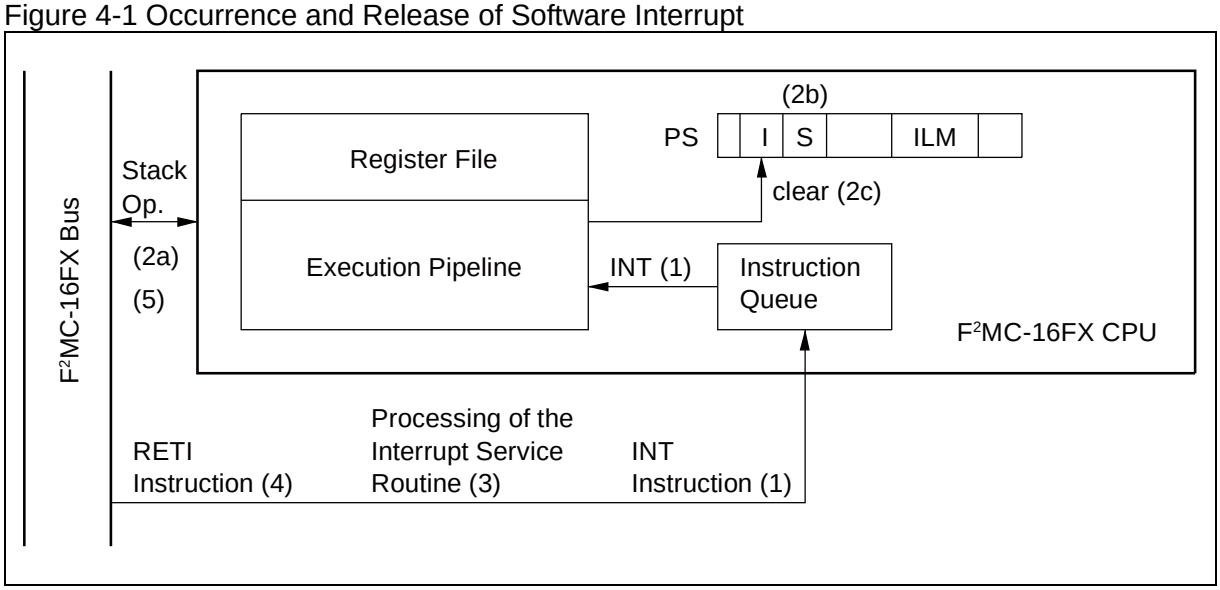
Software interrupts are fully handled within the CPU. To use a software interrupt, make the following set-up:

- Interrupt vector (in memory)
  - Consider the TBR value for a non-default location of the vector table.
  - The start address of the interrupt service routine has to be written to the appropriate interrupt vector (VecAddr = 4× (255-INT#) + 256×TBR).
- CPU
  - During interrupt processing, the CPU saves 12 bytes of data in CPU Special Registers to the memory area indicated by SSB and SSP. Thus the system stack pointer has to be initialized before using interrupts.
  - The CPU fetches three bytes of the interrupt vector and loads them onto PC and PCB. The interrupt handler routine has to start at this location. As a result, the interrupt processing program defined by the user is executed next. Normal operation is resumed at execution of the RETI instruction.

■ Software Interrupt Operation

When the CPU fetches and executes the software interrupt instruction, the software interrupt processing sequence is activated. The software interrupt processing sequence saves 12 bytes 12bytes of data in CPU Special Registers (PS, PC, PCB, DTB, ADB, DPR, and A) to the memory area indicated by SSB and SSP. The sequence then fetches three bytes of interrupt vector and loads them into PC and PCB, resets the I flag, and sets the S flag. Then, the sequence performs branch processing. As a result, the interrupt processing program defined by the user application program is executed next.
Figure 4-1 illustrates the flow from the occurrence of a software interrupt until the return from the interrupt processing program.

Figure 4-1 Occurrence and Release of Software Interrupt



1. The software interrupt instruction is executed.
2. Interrupt processing is performed by the CPU according to the software interrupt instruction.

· To save the CPU status, CPU special registers are transferred to the system stack (2a).
· The S bit is set to "1" (2b).
· The I flag is cleared to disable hardware interrupts (2c).
· The interrupt vector is fetched.
· Then control is transferred to the interrupt processing routine (branch to the address read as interrupt vector).

3. The Interrupt service routine is processed by the CPU.
4. The interrupt processing is completed with the RETI instruction in the user interrupt processing routine.
5. The CPU restores its context of special registers from system stack.
6. The CPU proceeds program execution with the next instruction after the INT instruction.

5. Multiple Interrupts

The F^{2}MC-16FX CPU supports multiple interrupts (simultaneous occurring interrupts and nested interrupt processing).

■ Multiple Hardware Interrupts

If an hardware interrupt of a higher priority (lower level value) occurs while another interrupt is being processed, control is transferred to the higher priority interrupt after the currently executing instruction is completed. After processing of the higher priority interrupt is completed, the original interrupt processing is resumed.

An interrupt of the same or lower priority may be generated while another interrupt is being processed. If this happens, the new interrupt request is suspended until the current interrupt processing is completed, unless the ILM value or I flag is changed by an instruction.

A DMA transfer cannot be interrupted and activated from multiple sources. While a DMA transfer is being processed, all other DMA requests are suspended. At simultaneous occurrence of requests for the DMA controller, the lowest channel number is processed first.

For a detailed description of DMA, please see "CHAPTER 4. DMA".

■ Multiple Software Interrupts

Software interrupts can not occur simultaneously. They are entered by executing the INT instruction and are always accepted. However, if an INT instruction is placed within an interrupt service routine, nested execution of software interrupts is possible.

■ Interrupt Acceptance Priority

Following table lists all interrupts with conditions for their acceptance.

Table 5-1 Control of Interrupt Acceptance Priority

Event	INT#	Type	Level	Acceptance condition	Action, if accepted	
Hardware event	-	Instruction Break (VEIB) system reserved	P2	Current instruction execution is finished, ILM>2 || P == 1	Save CPU status to system stack (In case of VEIB and VENMI interrupts the CPU status is stored in specific registers in DSU area. In case of INTE interrupts CPU status is stored in DSU registers) S = 1 Branch to interrupt vector	P = 0 ILM = 2
	-	Tool Break (VENMI) system reserved	U7 to U0 and P0 to P7	Current instruction execution is finished or string instruction are interruptable. Value of ILM and P depends of configuration in Debug system.		P = 0 ILM = 0 to 7 or P =1 ILM= 0 to 7 depending in configuration
	11	NMI	P4	Current instruction execution is finished or string instruction are interruptable. ILM>4 || P == 1		P = 0 ILM = 4
	from 13 on	Peripheral IRQ	IL U0...U7	Current instruction execution is finished or string instruction are interruptable. ILM > IL P == 1 I == 1 For multiple requests with same IL, smallest IRQ number is accepted.		
	12	Delayed INT	IL U0...U7	Current instruction execution is finished or string instruction are interruptable. ILM > IL P == 1 I == 1 No peripheral IRQs pending with same IL.		ILM = IL
	-	Software Instruction Break (INTE) system reserved	P2	always accepted		ILM = IL
Software event (Instruction)	9	INT9	-			P = 0 ILM = 2 I = 0
	10	Undefined instruction exception	-			I = 0
	all	INT instruction	-			I = 0
	-	RETI instruction	-			I = 0
					restore CPU status (including P, I, S, ILM)	

IL and ILM: Interrupt level and mask
I: Interrupt enable flag (Peripheral type interrupts)
S: System stack flag
P: Privileged mode flag (bit7 of CCR, PS)

Following table defines the naming of the interrupt levels, its corresponding P flag and ILM values. It also lists the interrupt cause, which can request the interrupt level.

Table 5-2 Interrupt Levels

Name	Category	P flag	ILM value	Priority	Used by
P0	Privileged mode	0	0	Highest	-
P1		0	1		-
P2		0	2		DSU
P3		0	3		-
P4		0	4		NMI
P5		0	5		-
P6		0	6		-
P7		0	7		-
U0	User mode	1	0		Peripherals
U1		1	1		
U2		1	2		
U3		1	3		
U4		1	4		
U5		1	5		
U6		1	6		
U7		1	7	Lowest	no request

6. Exceptions

The F^2MC-16FX performs exception processing at occurrence of various software and hardware events.

■ Software Exceptions (op-code)

Software exceptions are always accepted. Same as software interrupts, software exceptions disable any hardware interrupt acceptance. The software exceptions occur at code execution of following specific op-codes:

● Execution of an Undefined Instruction

All codes that are not defined in the instruction map are handled as undefined instructions. When an undefined instruction is executed, processing similar to the INT #10 software interrupt instruction is performed. Specifically, the PC value saved in the stack is the address at which the undefined instruction is stored. Processing can be restored by the RETI instruction, however it is of no use, because the same exception occurs again.

Operation:

(SSP)←(SSP)-2, ((SSP))←(AH)

(SSP)←(SSP)-2, ((SSP))←(AL)

(SSP)←(SSP)-2, ((SSP))←(DPR):(ADB)

(SSP)←(SSP)-2, ((SSP))←(DTB):(PCB)

(SSP)←(SSP)-2, ((SSP))←(PC)

(SSP)←(SSP)-2, ((SSP))←(PS)

(S)←1, (I)←0

(PCB)←Vector #10 address (upper byte)

(PC)←Vector #10 address (lower word)

● INT9

This instruction branches to the interrupt processing routine indicated by vector #9. Executing the RETI instruction in the interrupt routine restores the processing subsequent to the INT9 instruction.

Operation:

(SSP)←(SSP)-2, ((SSP))←(AH)

(SSP)←(SSP)-2, ((SSP))←(AL)

(SSP)←(SSP)-2, ((SSP))←(DPR):(ADB)

(SSP)←(SSP)-2, ((SSP))←(DTB):(PCB)

(SSP)←(SSP)-2, ((SSP))←(PC)+1

(SSP)←(SSP)-2, ((SSP))←(PS)

(S)←1, (I)←0

(PCB)←Vector #9 address (upper byte)

(PC)←Vector #9 address (lower word)

● INTE (System Reserved, Only Available with DSU, if DSU4 is used. Only Availabe at EVA device)

INTE is used to insert a software break point for the debug system, using the in circuit emulator (ICE). At insertion of a software instruction break, the first byte of the original instruction is replaced by INTE.
This instruction branches to the interrupt processing routine indicated by a fixed vector defined by the DSU. The PC value saved in the DSU register is the address at which INTE is stored. Executing the RETI instruction in the interrupt routine restores the processing at this location (INTE can be replaced by the original instruction at removal of the software break point).
The privileged mode flag (P flag) is cleared and the ILM register is set to 2 (enters level P2). This disables all hardware interrupts and exceptions. The P flag and ILM are restored at execution of the RETI instruction.

Operation:

(SSP)←(SSP)-2, DSU-register←(AH)

(SSP)←(SSP)-2, DSU-register←(AL)

(SSP)←(SSP)-2, DSU-register←(DPR):(ADB)

(SSP)←(SSP)-2, DSU-register←(DTB):(PCB)

(SSP)←(SSP)-2, DSU-register←(PC)

(SSP)←(SSP)-2, DSU-register←(PS)

(S)←1, (I)←0, (P)←0, (ILM)←2

(PCB)←Fixed vector from fetched

(PC)←Fixed vector from fetched

Without the DSU, INTE is handled same as the undefined instruction exception. Interrupt vector #10 is referenced. The P flag is not cleared and ILM is not updated. Even though the CPU context save in DSU4 mode is done at fixed DSU registers, SSP is decremented to keep consistency with the Softune Work Bench.

● INTE (System Reserved, Only Available with DSU, If OCDU is used)

INTE is used to insert a software break point for the debug system, using the debug interface. At insertion of a software instruction break, the first byte of the original instruction is replaced by INTE.
This instruction branches to the interrupt processing routine indicated by a fixed vector defined by the DSU. The PC value saved in the DSU register is the address at which INTE is stored. Executing the RETI instruction in the interrupt routine restores the processing at this location (INTE can be replaced by the original instruction at removal of the software break point).
The privileged mode flag (P flag) is cleared and the ILM register is set to 2 (enters level P2). This disables all hardware interrupts and exceptions. Two types of context switches are supported ("long context switch", "short context switch"). The P flag and ILM are restored at execution of the RETI instruction.

Operation "long context switch":

DSU-register ←(USB:SSB)

DSU-register ←(USP)

DSU-register ←(SSP)

DSU-register ←(AH)

DSU-register ←(AL)

DSU-register ←(DPR):(ADB)

DSU-register ←(DTB):(PCB)

DSU-register ←(PC)

DSU-register ←(PS)

(S)←1, (I)←0, (P)←0, (ILM)←2

(PCB)←Fixed vector fetched

(PC)←Fixed vector fetched

Operation "short context switch":
DSU-register ←(PCB)

DSU-register ←(PC)

DSU-register ←(PS)

(S)←1, (I)←0, (P)←0, (ILM)←2

(PCB)←Fixed vector fetched

(PC)←Fixed vector fetched

Without the DSU, INTE is handled same as the undefined instruction exception. Interrupt vector #10 is referenced. The P flag is not cleared and ILM is not updated.

■ Hardware Exceptions (Non Maskable Interrupts)

Hardware exceptions are external events, which are not maskable by any software instruction. Hardware exceptions with a higher level number, than the actual processed one, are suspended until execution of the RETI instruction restores the previous level. In addition, hardware exceptions disable any hardware interrupt acceptance.
At occurrence of multiple hardware exceptions at the same time, they will be accepted with following priority: VENMI > VEIB > NMI. If the current interrupt level mask and P flag setting allows it, hardware exceptions are accepted at the end of each instruction execution and during execution of string instructions. Since VENMI has configurable level in OCDU mode (0x0 to 0xF), the acceptance priority of VENMI as given above could change depending on its level configuration.

● NMI

NMI provides external hardware exception handling.
The privileged mode flag (P flag) is cleared and ILM is set to 4 (enters level P4). This disables all hardware interrupts from peripherals. The P flag and ILM are restored at execution of the RETI instruction.

Operation:

(SSP)←(SSP)-2, ((SSP))←(AH)

(SSP)←(SSP)-2, ((SSP))←(AL)

(SSP)←(SSP)-2, ((SSP))←(DPR):(ADB)

(SSP)←(SSP)-2, ((SSP))←(DTB):(PCB)

(SSP)←(SSP)-2, ((SSP))←(PC)

(SSP)←(SSP)-2, ((SSP))←(PS)

(S)←1, (P)←0, (ILM)←4

(PCB)←Vector #11 address (upper byte)

(PC)←Vector #11 address (lower word)

● Tool Break (VENMI, System Reserved, if DSU4 is Used. Only availabel at EVA device)

VENMI is provided for debugging with DSU. It implements various break factors.
The privileged mode flag (P flag) is cleared and ILM is set to 2 (enters level P2). This disables all hardware interrupts and exceptions. The P flag and ILM are restored at execution of the RETI instruction.

Operation:

(SSP)←(SSP)-2, DSU-register←(AH)

(SSP)←(SSP)-2, DSU-register←(AL)

(SSP)←(SSP)-2, DSU-register←(DPR):(ADB)

(SSP)←(SSP)-2, DSU-register←(DTB):(PCB)

(SSP)←(SSP)-2, DSU-register←(PC)

(SSP)←(SSP)-2, DSU-register←(PS)

(S)←1, (P)←0, (ILM)←2

(PCB)←Fixed vector fetched

(PC)←Fixed vector fetched

Even though the CPU context save in DSU4 mode is done at fixed DSU registers, SSP is decremented to keep consistency with the Softune Work Bench.

● Instruction Break (VEIB, System Reserved, if DSU4 is Used. Only availabel at EVA device)

VEIB is provided for debugging with DSU. It implements the instruction break after execution. Opposed to other hardware exceptions, VEIB is not accepted during execution of a string instruction.

Operation:

(SSP)←(SSP)-2, DSU-register←(AH)

(SSP)←(SSP)-2, DSU-register←(AL)

(SSP)←(SSP)-2, DSU-register←(DPR):(ADB)

(SSP)←(SSP)-2, DSU-register←(DTB):(PCB)

(SSP)←(SSP)-2, DSU-register←(PC)

(SSP)←(SSP)-2, DSU-register←(PS)

(S)←1, (P)←0, (ILM)←(2)

(PCB)←Fixed vector fetched

(PC)←Fixed vector fetched

Even though the CPU context save in DSU4 mode is done at fixed DSU registers, SSP is decremented to keep consistency with the Softune Work Bench.

● Tool Break (VENMI, System Reserved, if OCDU is Used)

VENMI is provided for debugging with DSU. It implements various break factors.
The privileged mode flag (P flag) and ILM are set according to configuration in OCDU. This disables all hardware interrupts and exceptions. Two types of context switches are supported ("long context switch", "short context switch"). It depends on OCDU configuration which type is used.
The P flag and ILM are restored at execution of the RETI instruction.

Operation "long context switch":

DSU-register ←(USB:SSB)

DSU-register ←(USP)

DSU-register ←(SSP)

DSU-register ←(AH)

DSU-register ←(AL)

DSU-register ←(DPR):(ADB)

DSU-register ←(DTB):(PCB)

DSU-register ←(PC)

DSU-register ←(PS)

(S)←1, (P)←programmable, (ILM)←programmable

(PCB)←Fixed vector fetch

(PC)←Fixed vector fetch

Operation "short context switch":

DSU-register ←(PCB)

DSU-register ←(PC)

DSU-register ←(PS)

(S)←1, (P)←programmable, (ILM)←programmable

(PCB)←Fixed vector fetch

(PC)←Fixed vector fetch

● Instruction Break (VEIB, System Reserved, if OCDU is Used)

VEIB is provided for debugging with DSU. It implements the instruction break before execution. The privileged mode flag (P flag) is cleared and ILM is set to 2 (enters level P2). This disables all hardware interrupts and exceptions. Two types of context switches are supported ("long context switch", "short context switch"). It depends on OCDU configuration which type is used.
The P flag and ILM are restored at execution of the RETI instruction.
Opposed to other hardware exceptions, VEIB is not accepted during execution of a string instruction.

Operation "long context switch":

DSU-register ←(USB:SSB)

DSU-register ←(USP)

DSU-register ←(SSP)

DSU-register ←(AH)

DSU-register ←(AL)

DSU-register ←(DPR):(ADB)

DSU-register ←(DTB):(PCB)

DSU-register ←(PC)

DSU-register ←(PS)

(S)←1, (P)←0, (ILM)←2

(PCB)←Fixed vector fetch

(PC)←Fixed vector fetch

Operation "short context switch":

DSU-register ←(PCB)

DSU-register ←(PC)

DSU-register ←(PS)

(S)←1, (P)←0, (ILM)←2

(PCB)←Fixed vector fetch

(PC)←Fixed vector fetch

7. Interrupt Vector

Hardware and software interrupts use the same vector table. The execution of interrupt service routines can be triggered by Interrupt request(IRQ) from peripheral or debug system(OCD/DSU4), or by executing the INT instruction, and specifying the number of the interrupt vector. Interrupt vectors are allocated between addresses as shown in Table 7-2. The location of the Interrupt vector table can be selected by the Interrupt Vector Table Base Register (TBR)..

■ Interrupt Vector

● Interrupt Vector Table Base Register (TBR)

TBRH

bit	15	14	13	12	11	10	9	8
	TB23	TB22	TB21	TB20	TB19	TB18	TB17	TB16
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	1	1	1	1	1	1	1	1

TBRL

bit	7	6	5	4	3	2	1	0
	TB15	TB14	TB13	TB12	TB11	TB10	-	-
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W0	R/W0
Initial value	1	1	1	1	1	1	0	0

The Table Base Register allows to relocate the interrupt vector table to any memory location in steps of 1 kbytes.
The value of the TBR defines the most significant 14-bit TB[23:10] of the 24-bit start address of the interrupt vector table. The least significant bits of TB[9:0] are fixed to "0".
The table base register TBR is initialized with $FFFC_H$ at reset, which results in an initial table base TB = $FFFC00_H$.
The interrupt vector table has a size of 1 kbyte (256 vector entries).

Table 7-1 Examples for TBR

TBR value	Start address of Interrupt vector table (table base)	End of Interrupt vector table	Comment
$FFFC_H$	$FF.FC00_H$	$FF.FFFF_H$	initial value (same as F^2MC-16LX)
$FB00_H$	$FB.0000_H$	$FB.03FF_H$	start of ROM bank FB
$00FC_H$	$00.FC00_H$	$00.FFFF_H$	end of bank "00"
0010_H	00.1000_H	$00.13FF_H$	inside of RAM-area
0000_H	00.0000_H	$00.03FF_H$	do not use, because of I/O-area

Note:

Interrupt Vector Table Base Register (TBR) and NMI control status register (NMI) can also be set via ROM configuration block e.g. to enable NMI before actual execution of user software starts. The initial value given above reflects the value after reset before it is changed via ROM configuration block. See "Section 7.4 Boot ROM program execution and Operation mode and ROM Configuration Block" for more details about ROM configuration block in "CHAPTER 7. RESETS AND STARTUP".

● Interrupt Vector Table

The interrupt vector table referenced during interrupt processing is assigned to addresses 256×TBR to 256×TBR+3FF$_H$ in memory. The reset defaults are from FFFC00$_H$ to FFFFFF$_H$ for the location of the vector table. If the vector table should not be located at top of ROM memory, another TBR value has to be configured.

Hardware interrupts, exceptions and software interrupts share the same vector table. Hence the interrupt service routine can either be called by a hardware interrupt or by the corresponding software interrupt.

The three bytes of each start address of the interrupt service routines have to be written to the appropriate interrupt vectors (VecAddr = 4× (255-INT#) + 256×TBR).

Table 7-2 Interrupt Vector Table

Interrupt / Vector number	Vector address	Index of level register in ICR	Hardware IRQ / Interrupt cause
INT 0 CALLV 0/1 (*1)	TB+3FC$_H$	-	-
INT 1 CALLV 2/3 (*1)	TB+3F8$_H$	-	-
INT 2 CALLV 4/5 (*1)	TB+3F4$_H$	-	-
INT 3 CALLV 6/7 (*1)	TB+3F0$_H$	-	-
INT 4 CALLV 8/9 (*1)	TB+3EC$_H$	-	-
INT 5 CALLV 10/11 (*1)	TB+3E8$_H$	-	-
INT 6 CALLV 12/13 (*1)	TB+3E4$_H$	-	-
INT 7 CALLV 14/15 (*1)	TB+3E0$_H$	-	-
INT 8 MODE Byte	TB+3DC$_H$	-	Reset
INT 9	TB+3D8$_H$	-	INT9 instruction
INT 10	TB+3D4$_H$	-	Undefined Instruction Exception
INT 11	TB+3D0$_H$	-	NMI
INT 12	TB+3CC$_H$	IL12	Delayed Interrupt

Interrupt / Vector number	Vector address	Index of level register in ICR	Hardware IRQ / Interrupt cause
INT 13	TB+3C8$_H$	IL13	RC Clock Timer
INT 14	TB+3C4$_H$	IL14	Main Clock Timer
INT 15	TB+3C0$_H$	IL15	Sub Clock Timer
INT 16	TB+3BC$_H$	IL16	<reserved>
INT 17	TB+3B8$_H$	IL17	Device specific peripheral.
INT 18	TB+3B4$_H$	IL18	
INT 19	TB+3B0$_H$	IL19	
～	～	～	
INT 254	TB+004$_H$	IL254	
INT 255	TB+000$_H$	-IL255	

(*1) When the program counter bank register (PCB) is same as TBRH, the CALLV instruction vector area overlaps the vector table of the INT#0 to INT#7 instruction. Ensure that the CALLV instruction does not use the same address as that of the INT#0 to INT#7 instruction, or do not use INT#0 to INT#7.

8. Interrupt Control Registers (ICR)

For each peripheral resource that has an interrupt function, there is an interrupt control register (ICR). The interrupt control register sets the interrupt level (IL) for the peripheral resource it is assigned to.

■ Interrupt Control Register (ICR)

ICR: IX

bit	15	14	13	12	11	10	9	8
	IX7	IX6	IX5	IX4	IX3	IX2	IX1	IX0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	1	1	0	0

ICR: IL

bit	7	6	5	4	3	2	1	0
	-	-	-	-	-	IL2	IL1	IL0
Attribute	-	-	-	-	-	R/W	R/W	R/W
Initial value	X	X	X	X	X	1	1	1

[bit15 to bit8] IX[7:0] : Index of the Interrupt Level (IL) to be Accessed

These bits are readable and writable, and specify the index of the interrupt level of the corresponding internal resource. It selects the number of the interrupt level to be accessed. IL[n] belongs to the peripheral interrupt request number IRQ[n], which both are related to the interrupt INT[n].

The system interrupts INT0 to INT11 have a fixed priority and thus have no interrupt levels. Writing to interrupt levels below the index of 12 has no effect, reading returns an undefined value. The same restriction applies for not available hardware interrupts above a device dependent maximum interrupt number.

Figure 8-1 illustrates the access to level registers by the IX pointer. The dashed line around IX and the selected IL shows the actual contents of the ICR.

Level configuration is written to or read from IL, where IX points to. To write the level configuration to a dedicated IL, specify the according index by writing IX before or simultaneously by word access. To read from a dedicated IL, IX must be written before reading IL.

Caution for the use of concurrent tasks:

In the case of concurrent tasks accessing the interrupt level information, be careful at the handling of the indexed access:

- Use word access to write information to ICR:IX and ICR:IL simultaneously.
- At read access, set the index ICR:IX and read the whole ICR register using word access. Check the ICR:IX value to match the intended index to be read for validation of the correct ICR:IL entry.

[bit7 to bit3] -: Undefined

Read access returns an undefined value.

Write always 0 to these bits.

Read modify write operations to this register have no effect on these bits.

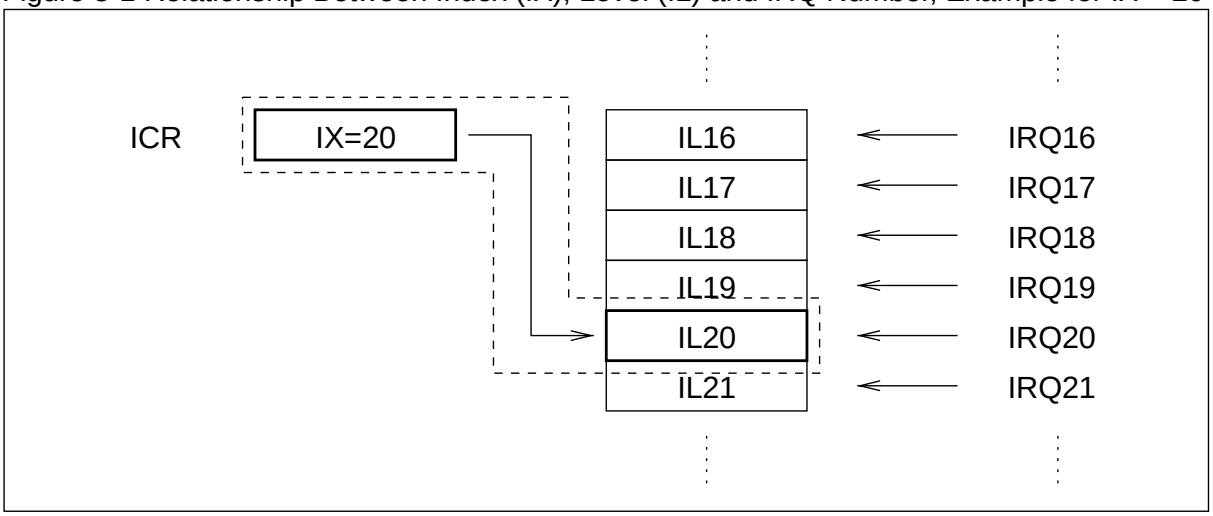
[bit2 to bit0] IL[2:0]: Interrupt Level Setting Bits

These bits are readable and writable, and specify the interrupt level of the corresponding internal resources. Upon a reset, these bits are initialized to level 7 (no interrupt). Table 8-1 describes the relationship between the interrupt level setting bits and interrupt levels.

Table 8-1 Interrupt Level Setting Bits and Interrupt Levels

IL2	IL1	IL0	Level
0	0	0	0 (Strongest)
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6 (Weakest)
1	1	1	7 (No interrupt)

Figure 8-1 Relationship Between Index (IX), Level (IL) and IRQ Number, Example for IX = 20



9. Non Maskable Interrupt (NMI)

The F^{2}MC-16FX CPU has a non maskable interrupt. By NMI Control Status Register (NMI), the, the feature of the external NMI pin can be enabled, it's level can be defined and a flag to quit the NMI request is provided.

■ NMI Control Status Register (NMI)

NMI								
bit	15	14	13	12	11	10	9	8
	-	-	-	-	-	LEV	EN	FLAG
Attribute	-	-	-	-	-	R/W	R/W1	R/W0
Initial value	X	X	X	X	X	1	0	X

−: no access
R/W: readable and writable
R/W1: readable, bit can be set only
R/W0: readable, bit can be cleared only

[bit15 to bit11] -: Undefined
Read access returns an undefined value.
Write always 0 to these bits.
Read modify write operations to this register have no effect on these bits.

[bit10] LEV: Signal Activity Level of the NMI Pin
The LEV bit defines the signal activity level of the NMI pin. A value of "1" defines logic high active input, a value of "0" define logic low active input of the NMI pin.
If the EN bit is not set, the LEV bit is readable and writable. If the EN bit is set, the state of LEV is locked. In that case LEV can only be read, writing to LEV with EN= "1" has no effect.
At reset the LEV bit is initialized to 1, thus the NMI pin is active high by default.

[bit9] EN: Non Maskable Interrupt Feature Enable Bit
The EN bit enables the feature to have a dedicated NMI pin. If EN is set to "0" the device has no NMI. The CPU will not react on signal level change at the NMI input. The pin can be used for an other function or general purpose. If EN is set to "1", the NMI pin is enabled. The CPU branches to the NMI exception processing, if an active signal level is detected at the NMI pin (defined by the LEV bit).
If EN is set to "1", the LEV and EN bits and NMI-relocation bit PRRR7:NMI_R are locked for writing. Neither the Signal Activity Level can be changed nor the NMI can be disabled after the NMI feature was enabled once. Only a device reset can change the EN bit back to "0".
At reset the EN bit is initialized to "0", thus the NMI feature is not enabled by default.
The LEV and EN bits must not be activated at same time (changed using the same access). EN must be enabled individually at last. Otherwise, an NMI can be caused due to relaxation time of the spike filter.

Note:

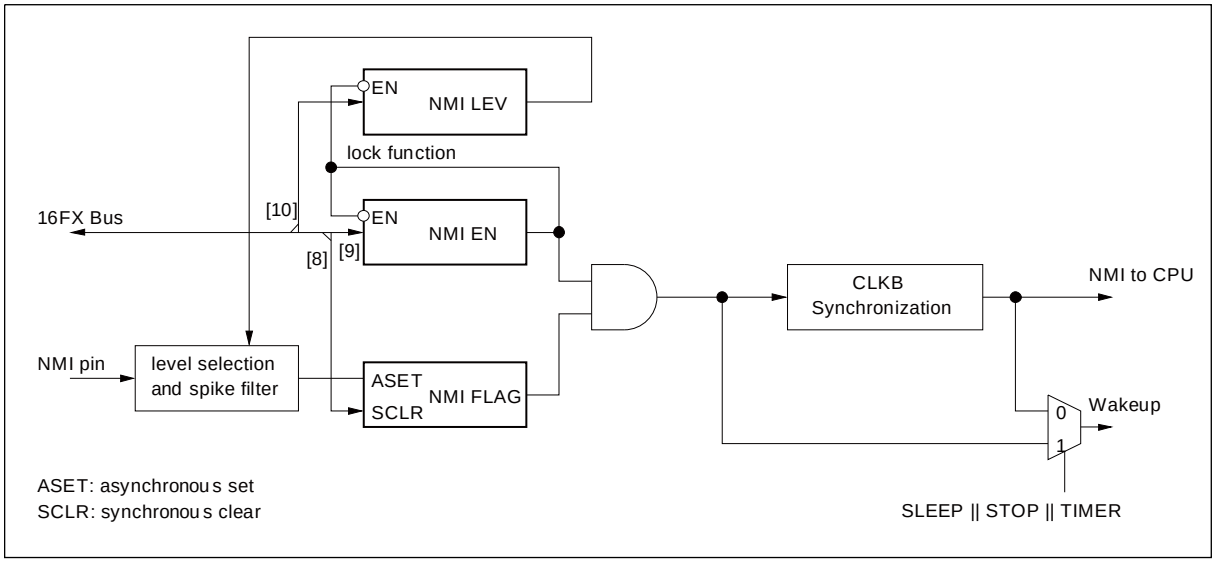
Available port (NMI or NMI_R) differ depending on each device. Please refer to "PIN ASSIGNMENTS" in Datasheet. If only port NMI_R is present on device, initial value of PRRR7:NMI_R would be "1".

[bit8] FLAG: Non Maskable Interrupt Flag

The NMI FLAG stores an asynchronous event of the NMI occurrence at the NMI pin.
A spike filter is used to filter out short pulses for spike suppression. The polarity of the pulses depends on the definition in the LEV bit.

The NMI flag is set by the hardware event (NMI occurrence) and can be cleared to "0" by software to quit the interrupt. An interrupt is only caused, if both the EN bit and FLAG are set.
The NMI FLAG can be read and cleared. Writing "1" to FLAG is ignored.
The NMI FLAG is undefined after reset. Before enabling the NMI, this flag should be cleared.
For bit manipulation, an RMW-read operation returns always "1" for this flag.

Figure 9-1 Operation of the NMI Control/Status Register



Note:

The NMI control status register (NMI) and Table base register (TBR) can also be configured via ROM configuration block e.g. to enable NMI before actual execution of user software starts. The initial value given above reflects the value after reset before it is changed via ROM configuration block. See "Section 7.4 Boot ROM program execution and Operation mode and ROM Configuration Block" for more details about ROM configuration block in " CHAPTER 7. RESETS AND STARTUP". for more details about ROM configuration block.

CHAPTER: DMA

This chapter explains the DMA functions and operations.

Management Code: 96F3DMA-E01.0

1. Overview

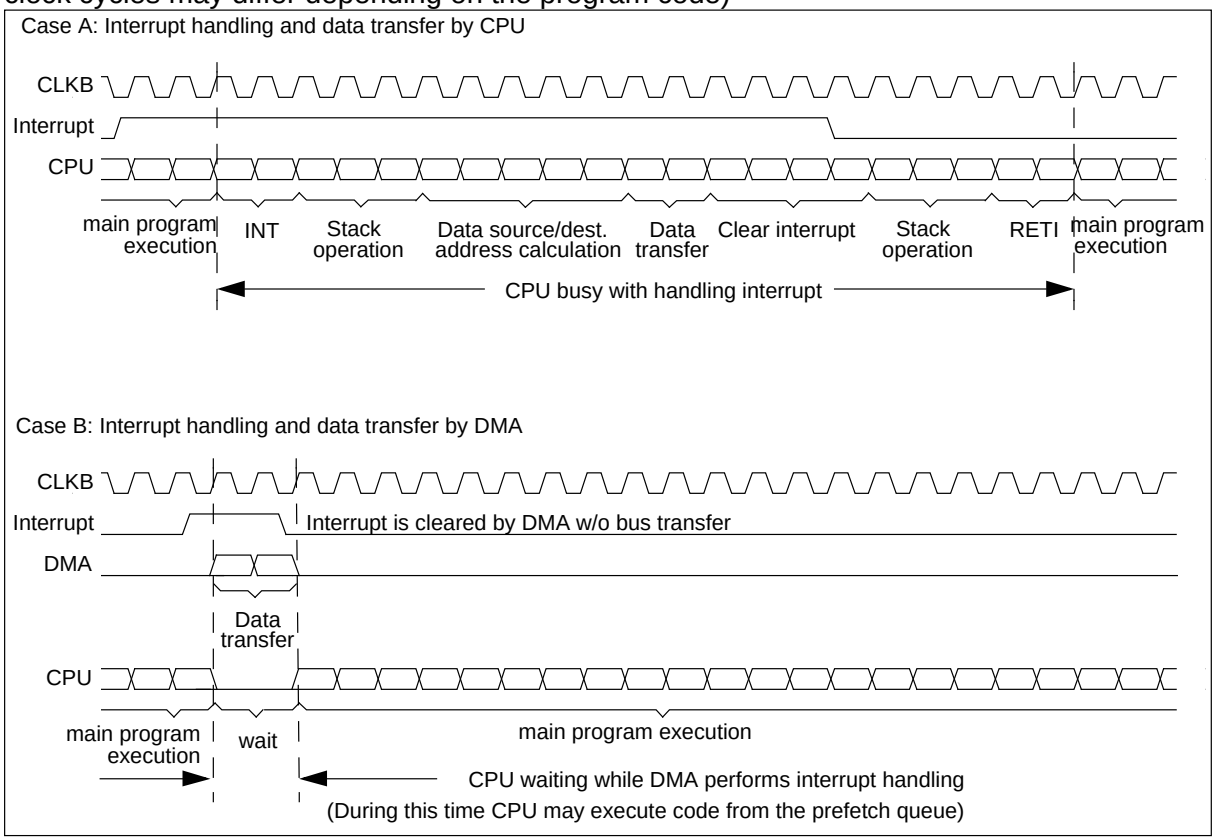
DMA enables automatic data transfer between memory and memory/peripheral resource registers or vice versa without interaction of the CPU. DMA replaces 16LX family's EI^{2}OS while offering the extended functionality at higher performance.
This section describes the DMA functions.

■ DMA Compared to Interrupt Handling:

DMA has the following advantages compared to handling the transfer via interrupt (see Figure 1-1):

- DMA reduces CPU processing load for interrupt handling
- No CPU register is used for the transfer, eliminating the need for register saving, which increases the transfer speed.
- The CPU bus is only used for the transfer operation itself. As consequence, the transfer speed is increased.

Figure 1-1 Comparison of Interrupt Handling by CPU and by DMA (principle only, number of clock cycles may differ depending on the program code)



■ DMA Functions

The DMA controller has the following functions:

- Automatic data transfer between peripheral resources (I/O) and memory
- CPU bus allocation during DMA transfer
  - Up to 16 DMA transfer channels with fixed priority scheme (the smaller the DMA channel number, the higher the priority of the request).
- Allow selection of whether or not to increment or decrement the transfer source and destination addresses (the buffer address may either be incremented, decremented or left unchanged; the I/O register address may either be incremented or left unchanged).
- DMA transfer can be started by a selectable peripheral resource interrupt (hardware IRQ).
- The DMA transfer is controlled by:
  - The DMA Enable Register (DER, bits 0...15)
  - The DMA interrupt select registers (DISEL 0 ... 15)
  - The DMA Stop Status Register (DSSR, bits 0...15)
  - The DMA Status Register (DSR, bits 0...15)
  - The DMA descriptors (8 byte each; DCT, IOA, DMACS, BAP)
  - The DMA IOA bank registers (IOABK)*
- DMA transfer can be stopped upon an error condition in the peripheral resource.
- At the end of a DMA transfer, processing automatically branches to the according interrupt service routine of the peripheral resource.

■ Structure

If enabled, the DMA controller serves interrupt requests of peripheral resources before they are passed to the interrupt controller. The structure of the interrupt system can be understood in that way, that the DMA controller is placed between the peripheral resources (IRQ sources) and the interrupt controller (see Figure 3.6-1 "Occurrence and release of hardware interrupt").

I/O service by DMA is handled by the following units:

- Peripheral resources
  - IF, IE: Interrupt flag and enable bits are used to control interrupt requests from resources (IRQ = IF && IE).
- DMA controller
  - DER and DISEL enable the DMA operation for a selected interrupt request. If DMA operation is not enabled, the request is passed to the interrupt controller.
  - Transfer information is stored in the DMA descriptors (data count, I/O address pointer, Buffer Address Pointer and control information) and the IOA bank registers.
  - The DMA controller processes the request (data transfer from/to peripheral) and clears the interrupt flag in the peripheral resource.
  - The DMA controller identifies its status (DSR) and indicates, if a stop condition has occured (DSSR). In case of DMA is stopped or completed, an interrupt is issued to the interrupt controller.
- Interrupt controller
  - The interrupt controller handles the interrupt due to completion or cancellation of DMA.
  - The ICR assigns interrupt levels, which determine the priority levels of simultaneously requested interrupts.
- CPU
  - I and ILM are used to compare the requested and current interrupt levels and to identify the interrupt enable status.
  - If the interrupt level is accepted, the CPU handles the interrupt request.

* IOA bank registers are not available for all devices. Please refer to the Datasheet.

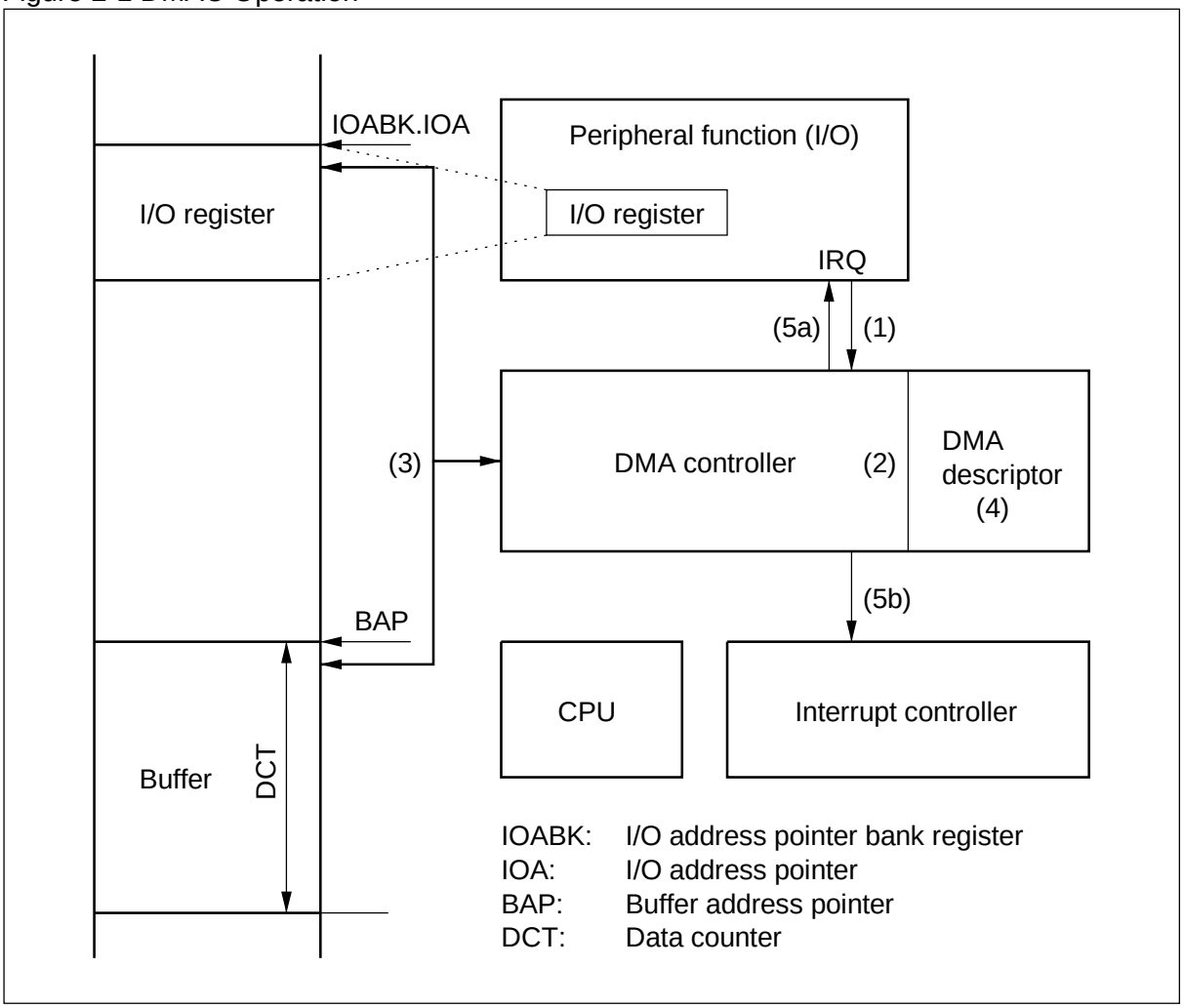
2. Operation

This section describes the DMA controller operation.

■ DMA Controller Operation

Figure 2-1 shows the DMA controller operation.

Figure 2-1 DMAC Operation



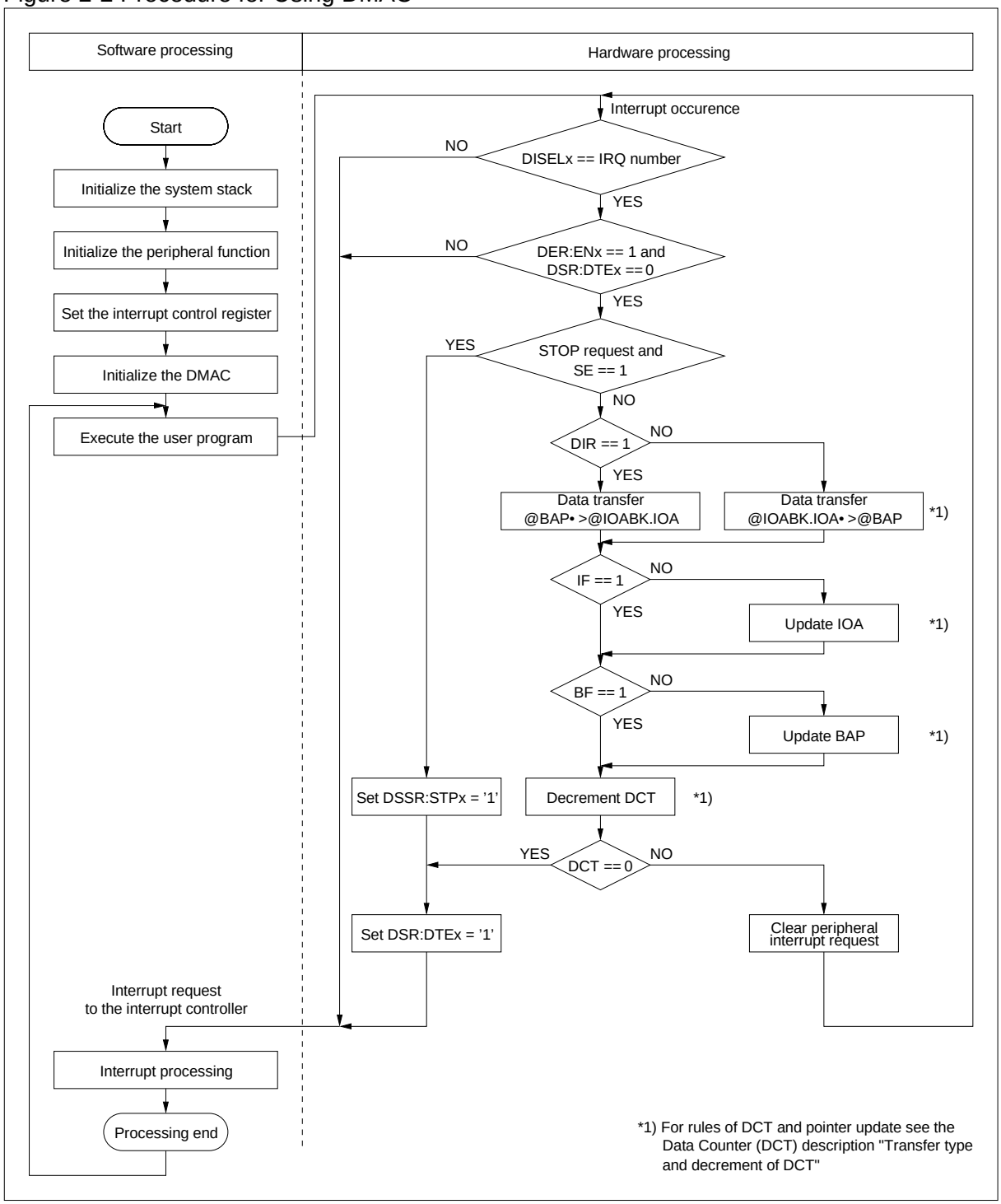
Data transfer using DMA is performed in the following order:

1. The peripheral resource (I/O) requests DMA transfer by asserting an interrupt.
2. When the corresponding bit of the DMA Enable Register (DER:EN) is "1" and the interrupt number matches the setting of the DISEL register, DMAC references from the descriptor the transfer source address, destination address, count and configuration of the channel.
3. DMA data transfer is started between I/O and memory.
4. DMA descriptors are updated.
5. After one item transferred (either Byte data or Word data)
   (a) Transfer has not been completed (DCT does not reach 0):
   DMAC clears the DMA transfer request (interrupt) of the peripheral resource.

   (b) At transfer end (DCT reaches 0):
   After completion of DMA transfer, the flag indicating completion of the transfer is set in the DMA Status Register (DSR:DTE bit), outputting an interrupt request to the interrupt controller.

■ Procedure for Using DMAC

Figure 2-2 Procedure for Using DMAC



■ Number of Cycles for the Data Transfer

The number of transfer cycles (bus cycles during DMA transfer) is the sum of all single transfers until DCT reaches 0. For reference of the number of cycles for a single DMA transfer, see Table 3-3, column "Transfer cycles".

3. Descriptor

For each DMA channel the DMA descriptor consists of 8 bytes. In addition each DMA channel offers 1 byte for extended configuration. DMA descriptor and extended configuration are used to set up a DMA transfer.

■ Configuration of DMA Descriptors and Extended Configuration

Each DMA channel has its own DMA descriptor of 8 bytes and the extended configuration register IOABK. The DMA descriptors are located from addresses 000100_H to $00017F_H$. The extended configuration registers are located from $000A00_H$ to $000A0F_H$. The following table shows the relation of DMA channels to the descriptor and extended configuration addresses.

The descriptor and extended configuration areas of not available channels are not accessible. Write "0" to the descriptor area of not available channels. Read access returns undefined values.

Before changing the configuration of a DMA channel it has to be ensured that the associated IRQ is deactivated. Alternatively the EN bit of the channel to be updated must be cleared.

Table 3-1 Relation of DMA Channel Numbers to DMA Descriptor and DMA Extended Configuration Addresses

DMA Channel	Descriptor Start Address	Descriptor End Address	IOABK
0	000100_H	000107_H	$000A00_H$
1	000108_H	$00010F_H$	$000A01_H$
2	000110_H	000117_H	$000A02_H$
3	000118_H	$00011F_H$	$000A03_H$
4	000120_H	000127_H	$000A04_H$
5	000128_H	$00012F_H$	$000A05_H$
6	000130_H	000137_H	$000A06_H$
7	000138_H	$00013F_H$	$000A07_H$
8	000140_H	000147_H	$000A08_H$
9	000148_H	$00014F_H$	$000A09_H$
10	000150_H	000157_H	$000A0A_H$
11	000158_H	$00015F_H$	$000A0B_H$
12	000160_H	000167_H	$000A0C_H$
13	000168_H	$00016F_H$	$000A0D_H$
14	000170_H	000177_H	$000A0E_H$
15	000178_H	$00017F_H$	$000A0F_H$

The structure of each channel is as shown in Figure 3-1.

Figure 3-1 DMA Descriptor and Extended Configuration

Address:	Extended Configuration	Attribute	Initial value:
0x000A00 + ch	Bank select of I/O register address pointer (IOABK)	(R/W)	0
	DMA Descriptor		
0x000107 + 8×ch	Upper 8 bits of data counter (DCTH)	(R/W)	X
0x000106 + 8×ch	Lower 8 bits of data counter (DCTL)	(R/W)	X
0x000105 + 8×ch	Upper 8 bits of I/O register address pointer (IOAH)	(R/W)	X
0x000104 + 8×ch	Lower 8 bits of I/O register address pointer (IOAL)	(R/W)	X
0x000103 + 8×ch	DMA control register (DMACS)	(R/W)	X
0x000102 + 8×ch	Upper 8 bits of buffer address pointer (BAPH)	(R/W)	X
0x000101 + 8×ch	Middle 8 bits of buffer address pointer (BAPM)	(R/W)	X
0x000100 + 8×ch	Lower 8 bits of buffer address pointer (BAPL)	(R/W)	X

R/W: readable and writable

The terms "ch" and "8×ch" define the address offsets of the descriptors and extended configuration, depending on the channel index "ch".

■ Each Register of DMA Descriptor

Each register contained in the DMA descriptor and extended configuration is explained in the following sections. The DMA descriptor registers must be initialized before setting DER:ENx to "1" because their initial values are undefined. The extended configuration registers are initialized at reset.

3.1. DMA Register List

Table 3-2 shows the register list of DMA.

Table 3-2 DMA Register List

Abbreviated Register Name	Register Name	Reference
IOABK	DMA I/O Address Pointer Bank Select	See 3.2
DCT	Data Count Register	See 3.3
IOA	I/O Register Address Pointer	See 3.4
DMACS	DMA Control Register	See 3.5
BAP	Buffer Address Pointer	See 3.6

3.2. DMA I/O Address Pointer Bank Select (IOABK)*

Each DMA channel has one I/O Address Pointer Bank Select Register (IOABK0 ... IOABK15). This register defines which bank is used to address the I/O space during DMA transfer.

■ DMA I/O Address Pointer Bank Select Register (IOABK)

IOABK0...15

bit	7	6	5	4	3	2	1	0
	BK7	BK6	BK5	BK4	BK3	BK2	BK1	BK0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

[bit7 to bit0] BK: Bank Select

For each DMA channel an IOABKx register exists, which defines the bank address for I/O register addressing during DMA transfer on this channel.

Note:

IOABK can be read and written.
The IOABK register is initialized to 0x00 at all resets.
At read operation, IOABK registers of not available channels return undefined value.
Write "0" to IOABK registers of not available channels.
During DMA transfer IOABK remains constant. A transfer across multiple banks is not possible.

* IOA bank registers are not available for all devices. Please refer to the Datasheet.

3.3. Data Count Register (DCT)

This section describes the Data Count Register (DCT).

■ Data Count Register (DCT)

The Data Count Register (DCT) is a 16-bit register to store the number of bytes to be transferred. After each data transfer, the Data Count Register is decremented by 1 at byte transfer or by 2 at word transfer.
When "0" is set to DCT, the maximum data transfer count (65,536 bytes) is set.
When the Data Count Register becomes "0", DMA transfer ends.

DCTH

bit	15	14	13	12	11	10	9	8
	C15	C14	C13	C12	C11	C10	C9	C8
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	X	X	X	X	X	X	X	X

DCTL

bit	7	6	5	4	3	2	1	0
	C7	C6	C5	C4	C3	C2	C1	C0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	X	X	X	X	X	X	X	X

■ Transfer Type and Decrement of DCT

The type of the transfer and the decrement value of DCT mainly depends on the DMACS:BW bit. However, also the configuration and status of IOA, BAP and DCT itself are considered.
Table 3-3 shows the transfers caused by different configuration or status of IOA, BAP, DCT and BW. The BAP increment/decrement operation depends on the DMACS:BPD bit.

Table 3-3 Transfer Type Resulting from Descriptor Configuration

Configuration/Status				Transfer type		Post transfer update of			Transfer cycles*
BW	DCT	IOA	BAP	I/O	Buffer	DCT	IOA when DMACS :IF = 1	BAP when DMACS :BF = 1	
0 (byte)	>0	-	-	byte	byte	-1	+1	+/-1	2
1 (word)	>1	even	even	word	word	-2	+2	+/-2	2
1 (word)	>1	even	odd	word	2 x byte	-2	+2	+/-2	3
1 (word)	>1	odd	even	2 x byte	word	-2	+2	+/-2	3
1 (word)	>1	odd	odd	2 x byte	2 x byte	-2	+2	+/-2	4
1 (word)	=1	-	-	byte	byte	-1	+1	+/-1	2

*: One transfer cycle is equal to one bus access

3.4. I/O Register Address Pointer (IOA)

This section describes the I/O Register Address Pointer (IOA).

■ I/O Register Address Pointer (IOA)

The I/O Register Address Pointer (IOA) is a 16-bit register and indicates the lower addresses (A15 to A0) of the I/O register. Its upper addresses (A23 to A16) can be set in IOABK register. The reset value of IOABK defaults to bank 0.

When "Update Performed" is specified with the DMACS:IF bit of the DMA Control Register, IOA is incremented by 1 at byte transfer; and incremented by 2 at word transfer. When "Update Not Performed" is specified with the DMACS:IF bit, IOA is fixed (IOA does not change).

IOAH

bit	15	14	13	12	11	10	9	8
	A15	A14	A13	A12	A11	A10	A9	A8
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	X	X	X	X	X	X	X	X

IOAL

bit	7	6	5	4	3	2	1	0
	A7	A6	A5	A4	A3	A2	A1	A0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	X	X	X	X	X	X	X	X

3.5. DMA Control Register (DMACS)

This section describes the DMA Control Register (DMACS).

■ DMA Control Register (DMACS)

The DMA Control Register (DMACS) is 8-bit long and is used:

- to specify, if the Buffer Address Pointer should be incremented or decremented (BPD)
- to specify whether to update or fix the Buffer Address Pointer and the I/O Register Address Pointer (BF, IF)
- to specify the transfer data format is byte or word (BW)
- to specify the transfer direction (DIR)
- to specify, if STOP request should be accepted (SE)

DMACS

bit	15	14	13	12	11	10	9	8
	-	-	BPD	IF	BW	BF	DIR	SE
Attribute	-	-	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	X	X	X	X	X	X	X	X

[bit15, bit14]-: Undefined

- Write always "0"
- read value is undefined
- Read modify write operations to this register have no effect on these bits.

[bit13] BPD: Buffer Pointer Decrement Bit

Bit	Description
0	In the case of BF=0, BAP is incremented after each transfer.
1	In the case of BF=0, BAP is decremented after each transfer. When both BPD and BW are set to "1" (word transfers in reverse order), even numbered values should be specified for IOA, BAP and DCT.

[bit12] IF: IOA Update or Fixed Selection

Bit	Description
0	After each data transfer, IOA is incremented.
1	IOA is not updated (fixed I/O address).

[bit11] BW: Transfer Data Length Specification

Bit	Description
0	Byte transfers are issued.
1	Word transfers are issued.

[bit10] BF: BAP Update or Fixed Selection

Bit	Description
0	After each data transfer, BAP is updated.
1	BAP is not updated (fixed buffer address).

[bit9] DIR: Data Transfer Direction

Bit	Description
0	Transfer from address specified by IOABK and IOA to address specified by BAP (@IOABK.IOA -> @BAP).
1	Transfer from address specified by BAP to address specified by IOABK and IOA (@BAP -> @IOABK.IOA).

[bit8] SE: DMA STOP Request Enable

Bit	Description
0	No reaction on DMA STOP request by peripheral.
1	DMA transfer can be stopped by a peripheral function (e.g. UART-RX).

3.6. Buffer Address Pointer (BAP)

This section describes the Buffer Address Pointer (BAP).

■ Buffer Address Pointer (BAP)

The Buffer Address Pointer (BAP) is a 24-bit register and is used to store addresses that will be used for DMA transfer. Any address can be specified.
When "Update Performed" is specified using the DMACS:BF bit, the BAP lower 16-bit (BAPM, BAPL) are incremented by "1" at byte transfer and incremented by "2" at word transfer. Its upper 8-bit do not change. When DMACS:BPD is set, BAP is decremented by "1" or "2" instead of incremented. When "Update Not Performed" is specified with the DMACS:BF bit, BAP is fixed (BAP does not change).

BAPH

bit	7	6	5	4	3	2	1	0
	B23	B22	B21	B20	B19	B18	B17	B16
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	X	X	X	X	X	X	X	X

BAPM

bit	15	14	13	12	11	10	9	8
	B15	B14	B13	B12	B11	B10	B9	B8
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	X	X	X	X	X	X	X	X

BAPL

bit	7	6	5	4	3	2	1	0
	B7	B6	B5	B4	B3	B2	B1	B0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	X	X	X	X	X	X	X	X

Note:

The area that can be specified using the I/O Address Bank Register (IOABK) and I/O Address Pointer (IOA) is "000000_H" to "$FFFFFF_H$". IOABK defaults to bank "00_H" after reset.
The area that can be specified using the Buffer Address Pointer (BAP) is "000000_H" to "$FFFFFF_H$".
The following addresses should not be specified for IOA and BAP: addresses of DMA controller internal registers (DISEL0...15, DSR, DSSR, DER); addresses of DMA descriptors "000100_H" to "$00017F_H$"; addresses of DMA IOABK registers "$000A00_H$" to "$000A0F_H$".

4. Registers

The DMA controller has three registers (DER, DSR, DSSR, each with one bit per DMA channel) and one register to select the interrupt (DISEL, one byte for each DMA channel).
The DMA descriptor is used to set-up the DMA transfer. It is explained in section "3 Descriptor".

Note for channels not available on the device:

At read operation, not available DSR, DSSR and DER bits return undefined value.
At read operation, not available DISEL registers return undefined value.
Write "0" to not available bits/registers.

■ List of DMA Registers

Abbreviated Register Name	Register Name	Reference
DISEL	DMA Interrupt Request Select Register	See 4.1
DSR	DMA Status Register	See 4.2
DSSR	DMA Stop Status Register	See 4.3
DER	DMA Enable Register	See 4.4

4.1. DMA Interrupt Request Select Register (DISEL)

Each DMA channel has one Interrupt Request Select Register (DISEL0 ... DISEL15). This register defines which IRQ number is used to trigger the DMA transfer on this channel.

■ DMA Interrupt Request Select Register (DISEL)

DISEL0...15

bit	7	6	5	4	3	2	1	0
	IS7	IS6	IS5	IS4	IS3	IS2	IS1	IS0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	1	1	0	0

[bit7 to bit0] IS: Interrupt Select

For each DMA channel a DISELx register exists, which defines the IRQ number to trigger the DMA transfer on this channel. The valid range of the selectable IRQ number is from 0x0C (12) to a device dependent maximum number. Only hardware interrupts can trigger DMA operation.

Note:

DISEL can be read and written.
The DISEL register is initialized to 0x0C (12) at reset.
At read operation, DISEL registers of not available channels return undefined value.
Write "0" to DISEL registers of not available channels.
Do not configure the same interrupt number in more than one enabled DMA channel.
When updating a DISEL register ensure that the associated IRQ is deactivated or the DMA channel is disabled.

4.2. DMA Status Register (DSR)

This section describes the DMA Status Register (DSR).

■ DMA StatuS Register (DSR)

DSRH

bit	15	14	13	12	11	10	9	8
	DTE15	DTE14	DTE13	DTE12	DTE11	DTE10	DTE9	DTE8
Attribute	R/W0	R/W0	R/W0	R/W0	R/W0	R/W0	R/W0	R/W0
Initial value	0	0	0	0	0	0	0	0

DSRL

bit	7	6	5	4	3	2	1	0
	DTE7	DTE6	DTE5	DTE4	DTE3	DTE2	DTE1	DTE0
Attribute	R/W0	R/W0	R/W0	R/W0	R/W0	R/W0	R/W0	R/W0
Initial value	0	0	0	0	0	0	0	0

[bit15 to bit0] DTEx: Data Transfer End Interrupt Request

For each DMA channel a DTE bit exists, which indicates the end of the data transfer. This could be the case if the Data Count Register (DCT) reaches 0 or if a stop request from a peripheral resource is asserted. With setting a DTE bit by the hardware of the DMA controller, an interrupt is requested.

Bit	Description
0	Indicates no interrupt request present for channel x.
1	Indicates that DMA transfer is completed or stopped and an interrupt is requested. If DTEx is set the DMA transfer is disabled on this channel.

Note:

Read access returns the status of all DTE bits of available channels in DSR.
At read operation, DTE bits of not available channels return undefined value.
A RMW-read instruction returns all DTE bits set to "1".
This bit is set by hardware only. Writing "1" to this bit is ignored.
Writing "0" clears this bit.
Write "0" to DTE bits of not available channels.

4.3. DMA Stop Status Register (DSSR)

This section describes the DMA Stop Status Register (DSSR)

■ DMA Stop Status Register (DSSR)

DSSRH

bit	15	14	13	12	11	10	9	8
	STP15	STP14	STP13	STP12	STP11	STP10	STP9	STP8
Attribute	R/W0	R/W0	R/W0	R/W0	R/W0	R/W0	R/W0	R/W0
Initial value	0	0	0	0	0	0	0	0

DSSRL

bit	7	6	5	4	3	2	1	0
	STP7	STP6	STP5	STP4	STP3	STP2	STP1	STP0
Attribute	R/W0	R/W0	R/W0	R/W0	R/W0	R/W0	R/W0	R/W0
Initial value	0	0	0	0	0	0	0	0

[bit15 to bit0] STPx: DMA Stop Status: Occurrence of Stop Request

Bit	Description
0	No STOP request occurred during DMA transfer on channel x, or STOP request was not enabled by the DMACS:SE bit.
1	Indicates that DMA transfer stopped due to STOP request issued by the peripheral resource during DMA transfer.

Note:

Read access returns the status of all STP bits of available channels in DSSR.
At read operation, STP bits of not available channels return undefined value.
A RMW-read instruction returns all STP bits set to "1".
This bit is set by hardware only. Writing "1" to the bit is ignored.
Writing "0" clears the bit.
Write "0" to STP bits of not available channels.
All DMA channels support the STOP request, however not all IRQ sources (peripherals) provide this feature.

4.4. DMA Enable Register (DER)

This section describes the DMA Enable Register (DER).

■ DMA Enable Register (DER)

DERH

bit	15	14	13	12	11	10	9	8
	EN15	EN14	EN13	EN12	EN11	EN10	EN9	EN8
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

DERL

bit	7	6	5	4	3	2	1	0
	EN7	EN6	EN5	EN4	EN3	EN2	EN1	EN0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

[bit15 to bit0] ENx: DMA Enable

Bit	Description
0	The interrupt request from the resource is not processed as a DMA start request, but it is passed to the interrupt controller.
1	The interrupt request from the resource is processed as a DMA start request. In consequence, the interrupt is handled by the DMA controller. At completion of the DMA transfer (DCT reaches 0) or stop request from peripheral, the interrupt request is output to the interrupt controller.

Note:

The register can be read and written.
At read/RMW-read operation, EN bits of not available channels return undefined value.
Write "0" to EN bits of not available channels.
Before entering any standby mode these bits must be cleared to zero. Otherwise interrupts associated to these DMA channels cannot be used for wakeup.

Do not change the configuration of a DMA channel, if its EN bit is set.

5. Examples of DMA Transfers

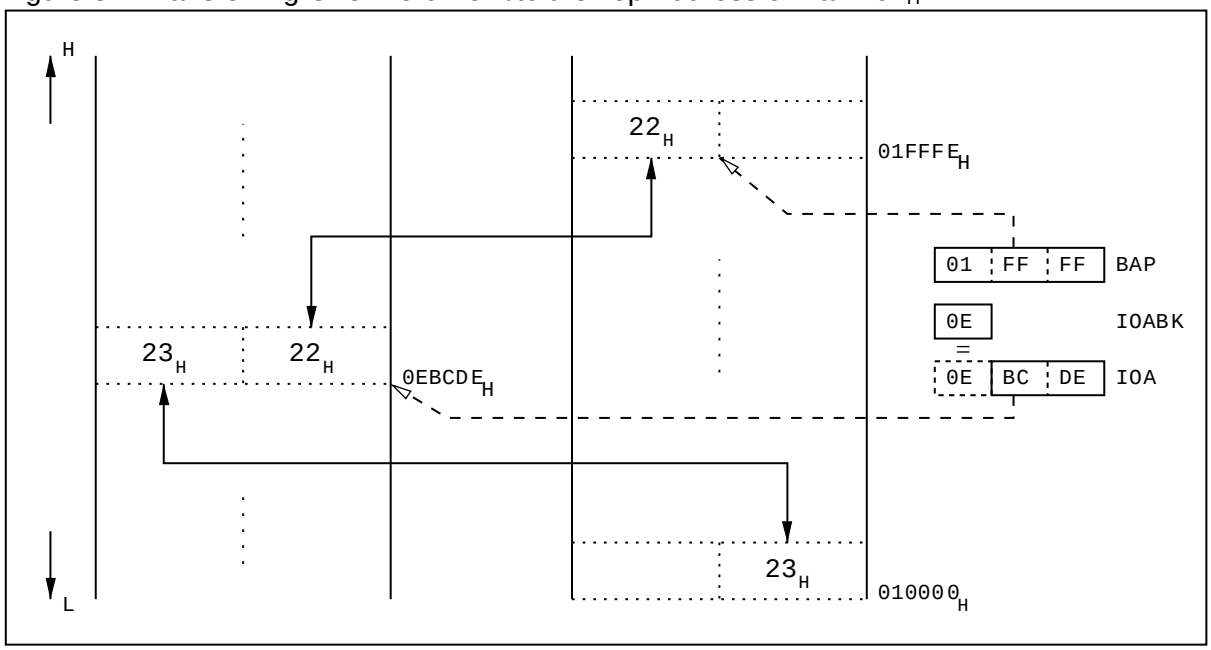
This section describes some of the lesser ordinary DMA transfer types or circumstances.

To simplify matters an even address somewhere in the memory bank selected by IOABK is used as IOA in the following examples. These examples still hold if the addresses of IOA and BAP are exchanged. The only exception is in the cases where BAP is to be decremented.

■ Wrap Around at Bank Boundary

A DMA transfer does not cross a bank boundary under any circumstances. If IOA or BAP are to be modified, their updated values fulfill this requirement. If a word has to be read from the top address of a bank, the lower byte is taken from the top address of the bank and the higher byte is taken from address 0 within the bank. Writing to the top address is handled accordingly (see Figure 5-1).

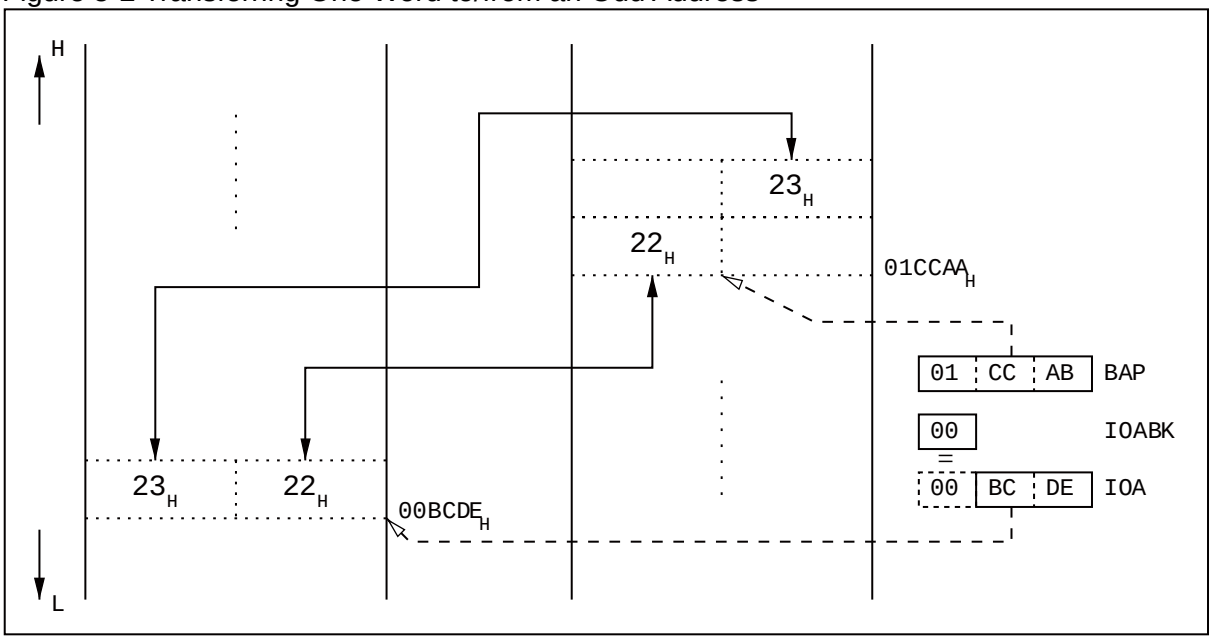
Figure 5-1 Transferring One Word from/to the Top Address of Bank 01_H



■ Word Transfer to an Odd Address

A word transfer incorporating an odd address is split in byte transfers as stated in Table 3-3. The result of such a transfer is shown in Figure 5-2.

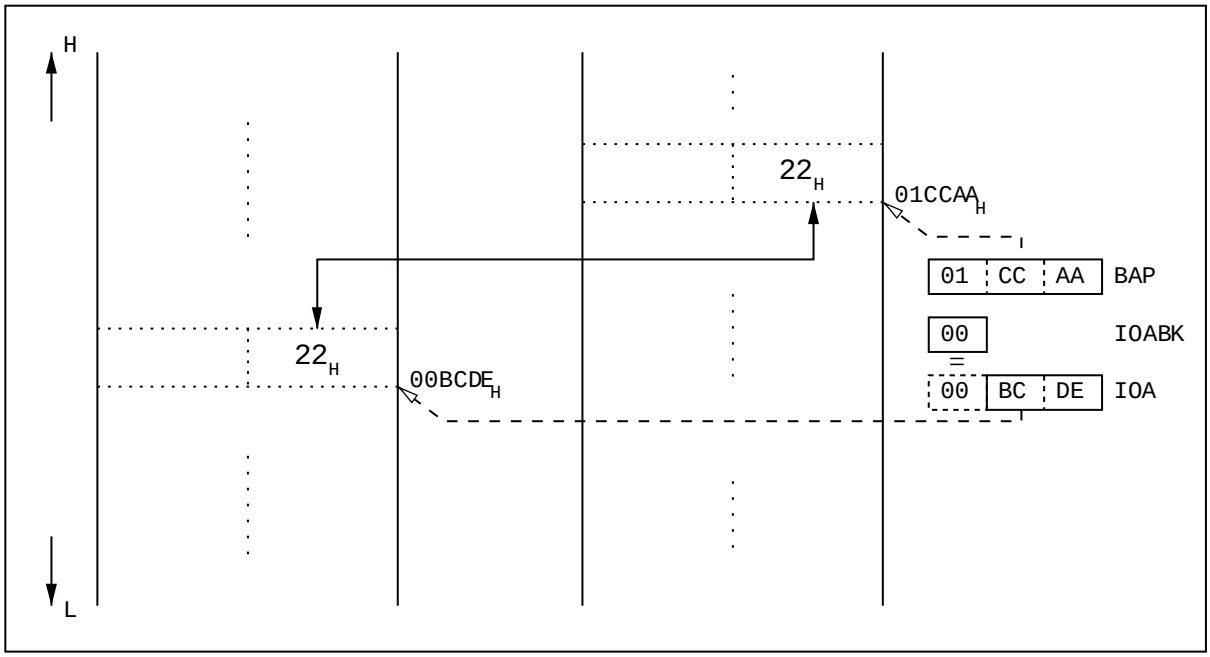
Figure 5-2 Transferring One Word to/from an Odd Address



■ Word Transfer with Odd Byte Count

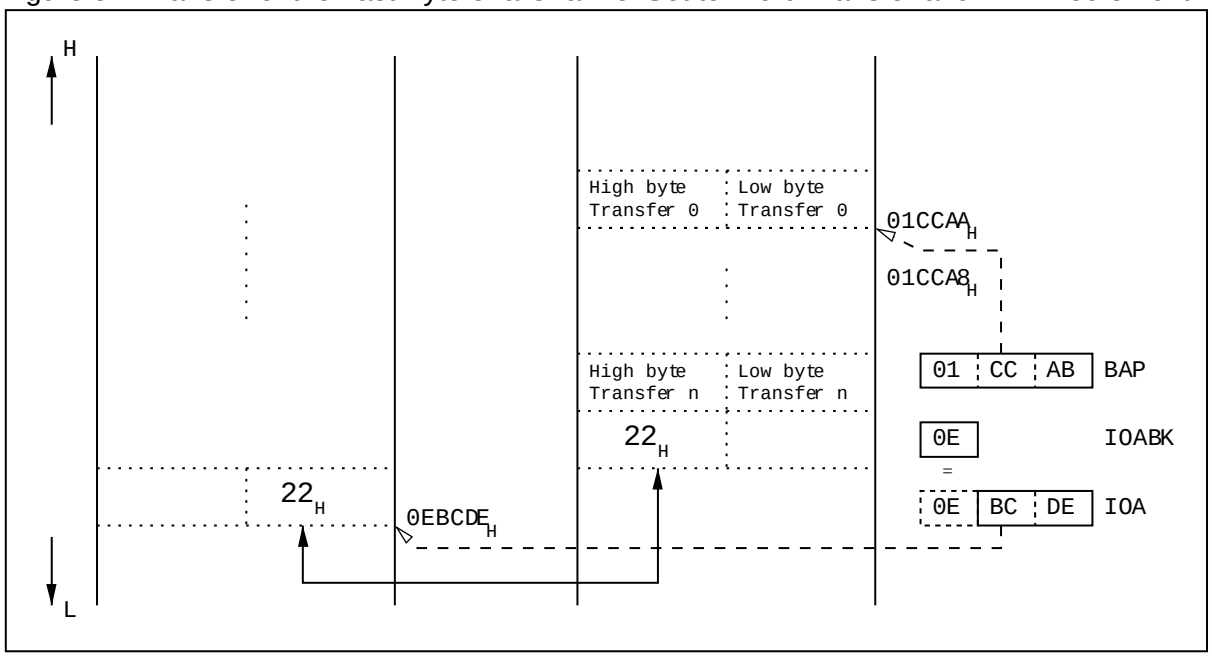
When DCT is set to an odd number of bytes and word transfers are used, the last transfer moves only one byte. In the case of BAP not being decremented (i.e. it is either fixed or to be incremented) the lower byte of the source is moved to the lower byte of the target (see Figure 5-3).

Figure 5-3 Transfer of the Last Byte of a Channel Set for Word Transfer



When BAP is to be decremented, the transferred byte is read from or written to the high byte of the word which is referred to by BAP (see Figure 5-4).

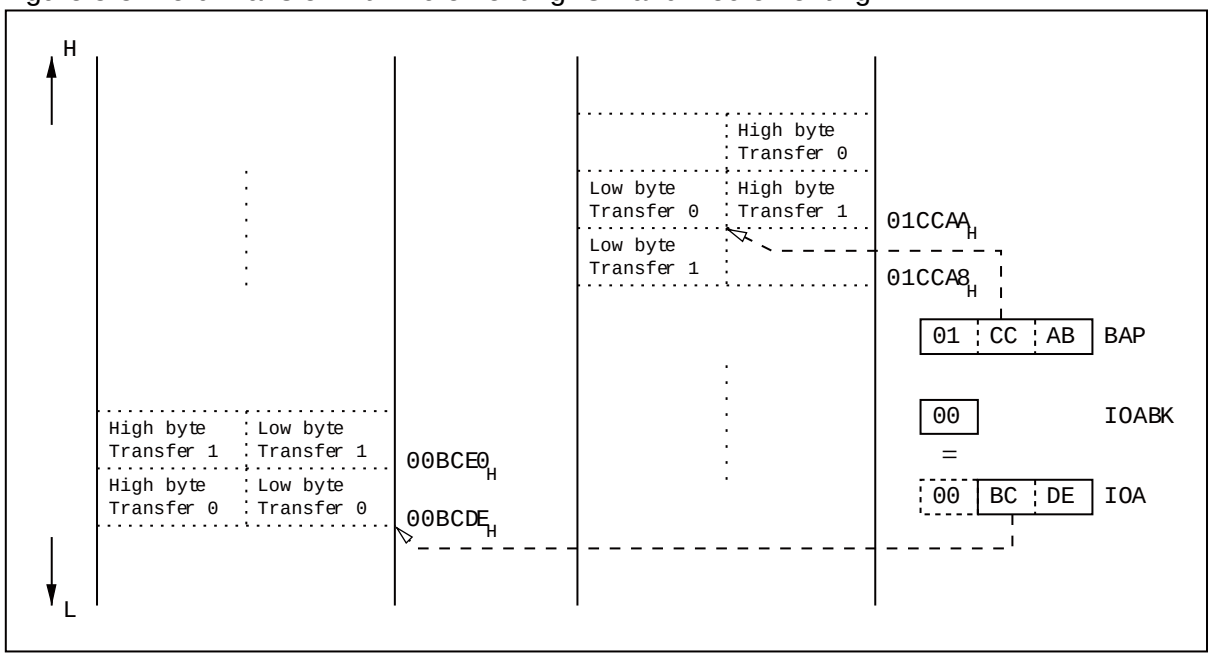
Figure 5-4 Transfer of the Last Byte of a Channel Set to Word Transfer and BAP Decrement



■ Word Transfers with Incrementing IOA and Decrementing BAP

When IOA is set to be incremented and BAP is set to be decremented, the order of the transmitted words is reversed, but not the order of the bytes inside a word. Figure 5-5 exemplifies the memory contents after the completion of two transfers.

Figure 5-5 Word Transfer with Incrementing IOA and Decrementing BAP



CHAPTER: DELAYED INTERRUPT

This chapter explains the functions and operations of the delayed interrupt.

Management Code: 96F3DELAYEDINT-E01.0

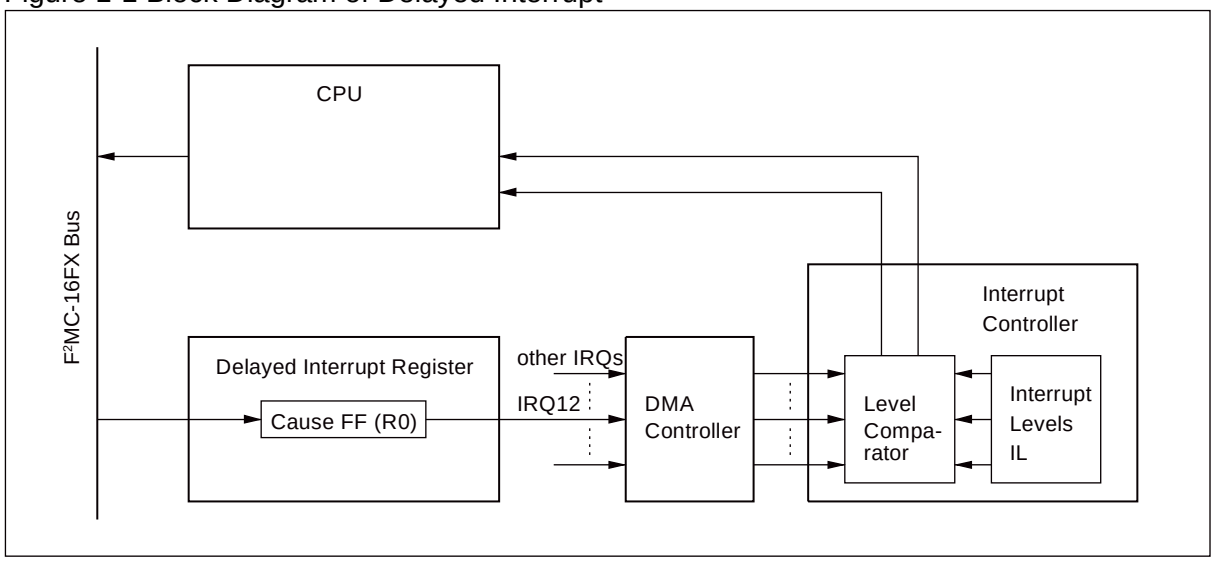
1. Overview

The Delayed Interrupt source module is used to generate interrupts for task switching. Using this module, interrupts to the F^{2}MC-16FX CPU can be requested and canceled by software.

■ Block Diagram of Delayed Interrupt

Figure 1-1 is a block diagram of the Delayed Interrupt source module.

Figure 1-1 Block Diagram of Delayed Interrupt



■ Notes on Operation

The Delayed Interrupt signal is activated by writing "1" to the corresponding bit of DIRR and inactivated by writing "0" to the same bit. Therefore, the interrupt bit in DIRR should be cleared to "0" within the interrupt service routine. Otherwise the same interrupt is serviced again right after the first interrupt service is completed.

Compared to other hardware interrupts, the delayed interrupt has lowest priority in the case of same levels are configured in the ICR register. For the delayed interrupt, priority is independent from the interrupt number. The delayed interrupt uses the interrupt vector of INT #12.

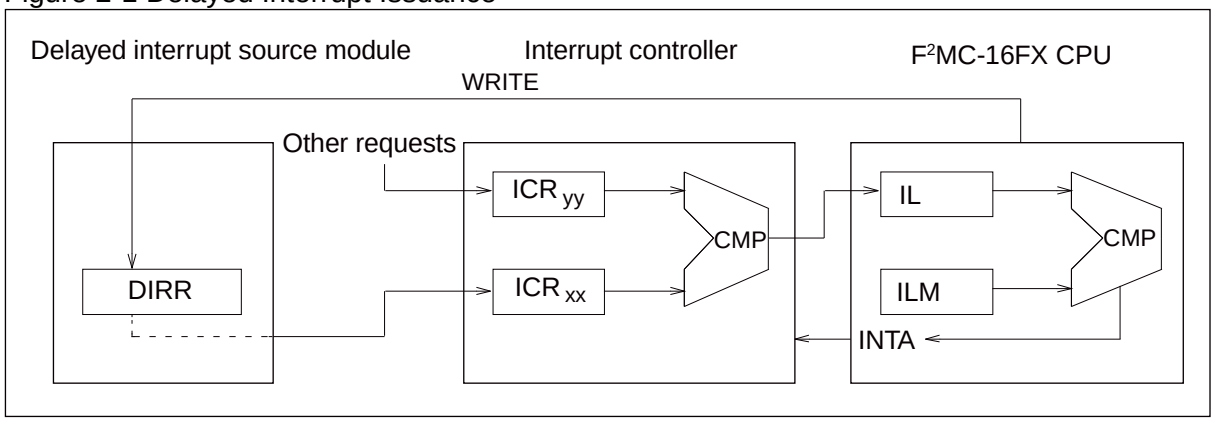
2. Operation

When the CPU writes "1" to the relevant bit of DIRR by software, the request latch in the Delayed Interrupt source module is set and an interrupt request is issued to the interrupt controller.

■ Delayed Interrupt Occurrence

When the CPU writes "1" to the relevant bit of DIRR by software, the request latch in the delayed interrupt source module is set and an interrupt request is issued to the interrupt controller. If this interrupt has the highest priority or if there is no other interrupt request, the interrupt controller issues an interrupt request to the F^{2}MC-16FX CPU. The F^{2}MC-16FX CPU compares the ILM bit of its internal CCR register and the interrupt request, and starts the hardware interrupt processing microprogram as soon as the current instruction is completed if the interrupt level of the request is higher than that of the ILM bit. Thus the interrupt processing routine for this interrupt is executed.

Figure 2-1 Delayed Interrupt Issuance



3. Register

DIRR controls request and cancellation of the Delayed Interrupt. Writing "1" to this register issues a delayed interrupt request, and writing "0" cancels the delayed interrupt request.

■ List of Delayed Interrupt Register

Abbreviated Register Name	Register Name	Reference
DIRR	Delayed Interrupt Cause Issuance/Cancellation register (Delayed Interrupt Request Register)	See 3.1

3.1. Delayed Interrupt Cause Issuance/Cancellation Register (DIRR: Delayed Interrupt Request Register)

■ Delayed Interrupt Cause Issuance/Cancellation Register (DIRR: Delayed Interrupt Request Register)

DIRR								
bit	7	6	5	4	3	2	1	0
	-	-	-	-	-	-	-	R0
Attribute	-	-	-	-	-	-	-	R/W
Initial value	X	X	X	X	X	X	X	0

[bit7 to bit1] -: Undefined Bits

- When these bits are read, the values are undefined.
- Write always "0" to these bits.

[bit0] R0: Delayed Interrupt Request Output Bit

This bit sets the generation/cancel of a delayed interrupt request.

Bit	Description
0	The delayed interrupt request is cleared.
1	The delayed interrupt request is output.

This bit is cleared after reset.

CHAPTER: CLOCKS

This chapter describes the clocks used by F^{2}MC-16FX family micro controllers.

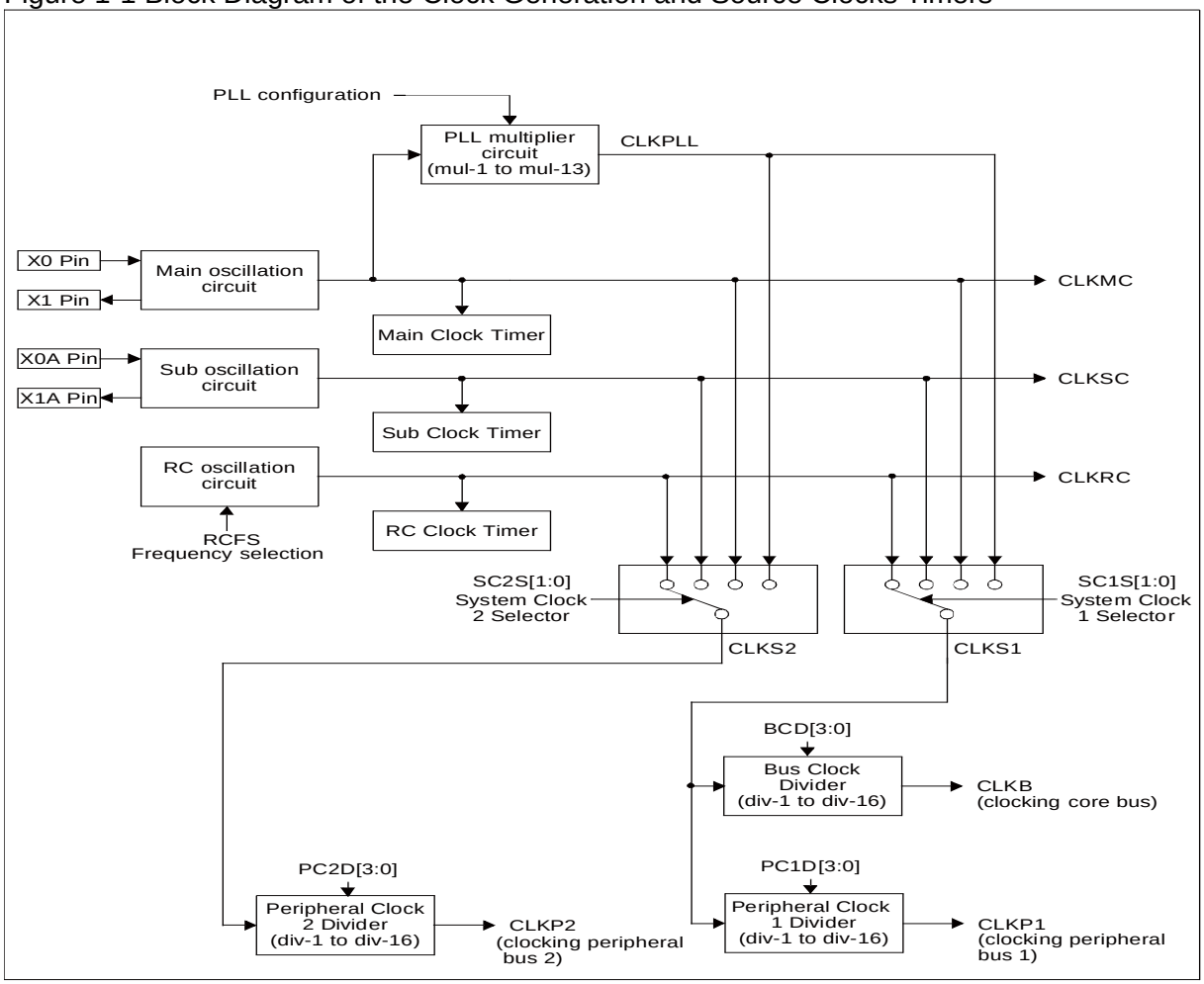
1. Overview
2. Clock Modes
3. Configuration of the PLL
4. Oscillation Stabilization Wait Time
5. Connection of an Oscillator or an External Clock to the Microcontroller
6. Registers

Management Code: 96F6CLK-E01.0

1. Overview

The F^{2}MC-16FX MCU offers up to 4 different clock sources (RC clock, Main clock, PLL clock and Sub clock). Flexible clock dividers allow an independent setting of the Bus clock (for the internal bus with the CPU and memories) and for the Peripheral clock frequencies.

Figure 1-1 Block Diagram of the Clock Generation and Source Clocks Timers



■ Source Clocks

The following clock signals are the clock sources of the F^{2}MC-16FX MCU.

● RC Clock (CLKRC)

The RC Clock (CLKRC) is the output clock of the internal RC oscillator. The RC oscillator can generate two different clock frequencies (2MHz and 100kHz nominal, see datasheet) which can be selected with the RCFS (RC Clock Frequency Select) bit of the CKFCR register.

● Main Clock (CLKMC)

The Main Clock (CLKMC) is the output clock of the main oscillation circuit. Either an external oscillator or an external clock can be connected to the main oscillation circuit.

● PLL Clock (CLKPLL)

The PLL Clock (CLKPLL) is obtained by multiplying the Main clock CLKMC with the internal PLL clock multiplier circuit (PLL oscillation circuit). Multiplication rates from 1 to 13 are available.
This clock is only available when the Main clock CLKMC is active.

● Sub Clock (CLKSC)

The Sub Clock (CLKSC) is the output clock of the sub oscillation circuit. Either an external oscillator or an external clock can be connected to the sub oscillation circuit.
If no Sub clock is required, then the pins can be used as general purpose I/O ports. The function is selected by the Sub Oscillator Configuration Marker SOCM in the ROM Configuration Block A (RCBA).

■ Internally Derived Clocks

The following clock signals are internally generated out of the source clocks.

● System Clock 1 (CLKS1)

The System clock 1 (CLKS1) is a master clock of the F^2MC-16FX MCU. It feeds the clock divider for the Bus clock (CLKB) and the Peripheral Clock 1 (CLKP1) and defines the clock mode of the CPU.
Depending on the SC1S[1:0] (System Clock 1 Select) bits of the CKSR register, one of the following 4 clocks can be selected as the System clock 1: CLKRC (RC clock), CLKMC (Main clock), CLKPLL or CLKSC (Sub clock).

● Bus Clock (CLKB)

The Bus Clock (CLKB) is the clock source for the internal bus, the CPU, internal memories and the DMA controller. It is stopped in all Standby modes (Sleep, Timer and Stop mode). It is generated by the programmable Bus clock divider out of the System clock 1 (CLKS1).

● Peripheral Clock 1 (CLKP1)

The Peripheral clock 1 (CLKP1) is the clock source for most peripheral resources. It is stopped in Timer and Stop mode. It is generated by the programmable Peripheral clock divider 1 out of the System clock 1 (CLKS1).

● System Clock 2 (CLKS2)

The System clock 2 (CLKS2) is a second master clock. It feeds the clock divider for the Peripheral Clock 2 (CLKP2) only.
Depending on the SC2S[1:0] (System Clock 2 Select) bits of the CKSR register, one of the following 4 clocks can be selected as the System clock 2: CLKRC (RC clock), CLKMC (Main clock), CLKPLL or CLKSC (Sub clock).
The setting of these bits does not influence the clock mode of the CPU which is defined by SC1S[1:0] only.

● Peripheral Clock 2 (CLKP2)

The Peripheral clock 2 (CLKP2) is the clock source for some dedicated peripheral resources. It is stopped in Timer and Stop mode. It is generated by the programmable Peripheral clock divider 2 out of the System clock 2 (CLKS2).

■ Source Clock Timers

The Main clock, RC clock and the Sub clock oscillators feed dedicated source clock timers (Main Clock Timer, RC Clock Timer and Sub Clock Timer) which are running independently of the selected System clock, Bus clock and Peripheral clock.

2. Clock Modes

Four clock modes are provided: RC clock mode, Main clock mode, PLL clock mode and Sub clock mode.

■ Definition of Clock Modes

The clock mode of the MCU is defined by the source for the System clock 1 (CLKS1) which is selected by the SC1S[1:0] bits. CLKS1 is the master clock for the Bus clock (CLKB) that supplies the internal bus with CPU, memories and the DMA controller, and for the Peripheral clock 1 (CLKP1) that supplies most peripheral modules.
The selection of the System clock 2 (CLKS2) can be made independent of the clock mode.

● RC clock Mode

In RC clock mode, the clock generated by the internal RC oscillator is used for the System clock 1 (CLKS1). Hence the Bus clock (CLKB) and the Peripheral clock 1 (CLKP1) are fed by the RC oscillator.
The RC clock frequency can be selected with the RC Clock Frequency Select bit (CKFCR: RCFS) between nominal 100kHz and 2MHz.
The Main clock, PLL clock and Sub clock can be disabled with the MCE, PCE and SCE bits if they are not used for the System clock 2 or the watchdog. Disabling these clocks also disables the corresponding source clock timers.

● Main Clock Mode

In Main clock mode, the output signal of the Main oscillator is used for the System clock 1 (CLKS1). Hence the Bus clock (CLKB) and the Peripheral clock 1 (CLKP1) are fed by CLKMC.
The RC clock, PLL clock and Sub clock can be disabled with the RCE, PCE and SCE bits if they are not used for the System clock 2, the watchdog or clock stop detect function. Disabling these clocks also disables the corresponding source clock timers.

● PLL Clock Mode

In PLL clock mode, the PLL clock CLKPLL is used for the System clock 1 (CLKS1). Hence the Bus clock (CLKB) and the Peripheral clock 1 (CLKP1) are fed by CLKPLL.
The frequency of this clock signal depends on the setting of the PLL Control Register and the Main clock.
The RC clock and Sub clock can be disabled with the RCE and SCE bits if they are not used for the System clock 2, the watchdog or clock stop detect function. Disabling these clocks also disables the corresponding source clock timers. The Main clock cannot be disabled in PLL clock mode.

● Sub Clock Mode

In Sub clock mode, the output signal of the Sub oscillator CLKSC is used for the System clock 1 (CLKS1). Hence the Bus clock (CLKB) and the Peripheral clock 1 (CLKP1) are fed by the CLKSC.
The RC clock, Main clock and PLL clock can be disabled with the RCE, MCE and PCE bits if they are not used for the System clock 2, the watchdog or clock stop detect function. Disabling these clocks also disables the corresponding source clock timers.

■ Clock Source Switching

Clock source switching means changing the clock source for CLKS1 or CLKS2.

● Changing the Clock Source

- The clock source selection can be done independently for CLKS1 and CLKS2
- A clock source switching is done by writing a new value to the System clock select bits (SC1S[1:0] for System clock 1 and SC2S[1:0] for System clock 2)
- A transition to a different clock source is possible only when the selected clock is ready (means the corresponding ready flag RCM, MCM, PCM or SCM in the Clock Monitor Register CKMR is set).
- If the selected clock is ready, then the clock source transition will be executed by starting the synchronization mechanism. The System Clock Monitor bits (SC1M for System clock 1 and SC2M for System clock 2) of the Clock Monitor register change to the new value after completion of this clock switching.
- Writing a second value to the System clock select bits before switching to the mode selected before has completed, cancels the first transition request.
- If the selected clock is not available or not stabilized, then switching to the new clock mode will be delayed until the selected clock is available and stable (clock ready flag CKMR: RCM, MCM, PCM or SCM set).
- After Stop mode release by wakeup interrupt, the System Clock Selectors directly select the clocks which are defined by the SC1S[1:0]/SC2S[1:0] bits, irrespective of the clock ready monitor bits. Hence do not select an unavailable clock before switching to Stop mode because this clock is used for the restart by a wakeup interrupt.

● Reset and Clock Source

The clock source for both System clocks (CLKS1 and CLKS2) is set to RC clock (CLKRC) by any reset. The RC clock stabilization time is applied after each reset.

● Access to Peripheral Resources Clocked with CLKP2

Note:

Do not access (read or write) any resource clocked with the peripheral clock 2 (CLKP2) directly after changing the setting of the SC1S or SC2S bits. Do always make sure that the requested clock transition has completed by reading the SC1M/SC2M monitor bits. Access to resources clocked with CLKP2 is allowed only if SC1M[1:0] equals to SC1S[1:0] and SC2M[1:0] equals to SC2S[1:0].

■ Activating and Disabling Source Clocks

- After each reset, the Main oscillator and RC oscillator are enabled while the PLL multiplier circuit is disabled.
- The Sub oscillator is disabled by a Power or External reset. Then the Boot ROM program reads the Sub Oscillator Configuration Marker SOCM and activates the selected mode of the Sub oscillator. A Software, Watchdog or Clock stop detection reset has no influence on this setting. Such a reset only initializes the SCE bit to "1".
- Source clocks selected for the System clocks CLKS1 or CLKS2 are always activated.
- Source clocks not used for the System clocks can be enabled and disabled with the corresponding enable bits in the CKSR register.
- Disable a source clock for current saving by setting the corresponding enable bit in the CKSR register to "0".
- Enable a source clock if it is needed for a certain function or for a fast System clock changing by setting the corresponding enable bit in the CKSR register to "1". After stabilization of the activated clock, the corresponding Clock monitor bit of the CKMR register is set and indicates the clock as "ready".
- The Main clock must be enabled if the PLL clock should be enabled (The setting of PCE has no effect when MCE is set to "0").
- Setting MCE to "0" does not disable the Main oscillator when System clock 1 or 2 is set to Main or PLL clock because the Main clock is the input signal for the PLL multiplier circuit.

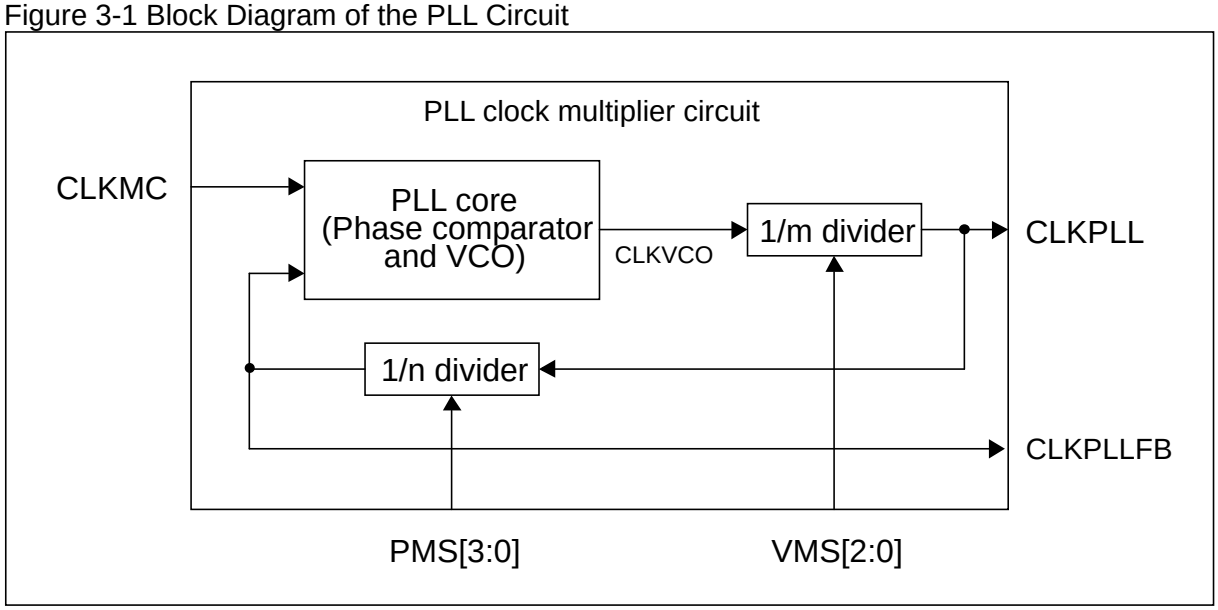
3. Configuration of the PLL

The PLL multiplier circuit is used to generate the PLL clock out of the Main clock. 13 different multiplier values (mul-1 to mul-13) are available.

■ Components of the PLL Clock Multiplier Circuit

The following block diagram describes the modules of the PLL.

Figure 3-1 Block Diagram of the PLL Circuit



● PLL Core (Phase comparator and VCO)

The PLL core consists of a phase comparator which controls a voltage controlled oscillator (VCO). The phase comparator compares the input clock CLKMC (Main clock) with the output clock of the "1/n divider". The VCO generates the output clock CLKVCO which frequency depends on the setting of the "1/m divider" and the "1/n divider": CLKVCO = CLKMC × m × n. The allowed frequency range of CLKVCO is specified in the Datasheet.

● "1/m Divider" (VCO clock divider)

The "1/m divider" is controlled by the VMS[2:0] VCO clock Multiplier Select bits. It defines the frequency relation between the PLL output clock CLKPLL and the VCO output clock CLKVCO. The minimum division value is "2" to guarantee a PLL output clock with a duty cycle of 50%. It is also used to keep the VCO in its operating range.

These bits must be set depending on the required CLKPLL frequency in a way that the resulting CLKVCO frequency is always within the permitted range as described in the datasheet. Otherwise the PLL will not lock and the resulting CLKPLL frequency is undefined.

Do not change the setting of these bits after activation of the PLL.

● "1/n Divider" (PLL clock divider)

The "1/n divider" is controlled by the PMS[3:0] PLL clock Multiplier Select bits. It defines the frequency multiplication value of the PLL clock multiplier circuit (relation between CLKMC and CLKPLL). Multiplication values from 1 to 13 are available. However the minimum and maximum permitted frequencies for CLKMC, CLKVCO and CLKPLL (->CLKS1/CLKS2) must be adhered. Do not change the setting of these bits after activation of the PLL.

● Setting the VCO Clock Multiplier Select Bits VMS[2:0]

The "1/m divider" is programmable to keep the VCO in its operating range as specified in the datasheet for a wide range of PLL clock frequencies. Write a value to the VMS[2:0] bits which results in a valid CLKVCO clock frequency. The table below shows recommended settings for a 4MHz Main clock. For other Main clock frequencies, it is necessary to adjust the VMS[2:0] setting based on the formula: "CLKVCO = CLKMC × m × n" .

Table 3-1 Recommended Settings for the PMS[3:0], VMS[2:0] Bits for CLKMC=4MHz

Main Oscillation frequency CLKMC	Requested PLL output frequency CLKPLL	Setting for PMS[3:0] bits (n division value)	Setting for VMS[2:0] bits (m division value)	VCO output frequency CLKVCO
4 MHz	4 MHz	"0000" (div 1)	"111" (div 16)	64 MHz
4 MHz	8 MHz	"0001" (div 2)	"011" (div 8)	64 MHz
4 MHz	12 MHz	"0010" (div 3)	"010" (div 6)	72 MHz
4 MHz	16 MHz	"0011" (div 4)	"001" (div 4)	64 MHz
4 MHz	20 MHz	"0100" (div 5)	"001" (div 4)	80 MHz
4 MHz	24 MHz	"0101" (div 6)	"001" (div 4)	96 MHz
4 MHz	28 MHz	"0110" (div 7)	"000" (div 2)	56 MHz
4 MHz	32 MHz	"0111" (div 8)	"000" (div 2)	64 MHz
4 MHz	36 MHz	"1000" (div 9)	"000" (div 2)	72 MHz
4 MHz	40 MHz	"1001" (div 10)	"000" (div 2)	80 MHz
4 MHz	44 MHz	"1010" (div 11)	"000" (div 2)	88 MHz
4 MHz	48 MHz	"1011" (div 12)	"000" (div 2)	96 MHz
4 MHz	52 MHz	"1100" (div 13)	"000" (div 2)	104 MHz

4. Oscillation Stabilization Wait Time

When the power is turned on, when stop mode is released or when a disabled clock is enabled, an oscillation stabilization wait time is required before the clock can be used.

■ Oscillation Stabilization Wait Interval

Ceramic and crystal oscillators which can be connected to the X0/X1 and X0A/X1A pins generally require several ms to stabilize at their natural frequency (oscillation frequency) when oscillation starts. The internal PLL multiplier circuit and the RC oscillator also need a certain stabilization time after activation. For this reason, CPU operation with the activated clock is not allowed immediately after oscillation starts but is allowed only after full oscillation stabilization.

After the oscillation stabilization wait interval has elapsed, the corresponding clock ready monitor bit will be set and the clock can be selected as System clock.

Because the oscillation stabilization time of the Main and Sub oscillator depends on the type of oscillator (crystal, ceramic, etc.), the proper oscillation stabilization wait interval for the oscillator used must be selected. An oscillation stabilization wait interval is selected by setting the Clock Stabilization Select Register (CKSSR). The stabilization time of the RC oscillator is fixed and the stabilization time of the PLL can be selected depending on the PLL and main clock frequency.

When the clock source is switched, the MCU runs with the previously selected clock until the newly selected clock is stabilized. When the corresponding clock ready flag is set, the MCU changes to the specified clock.

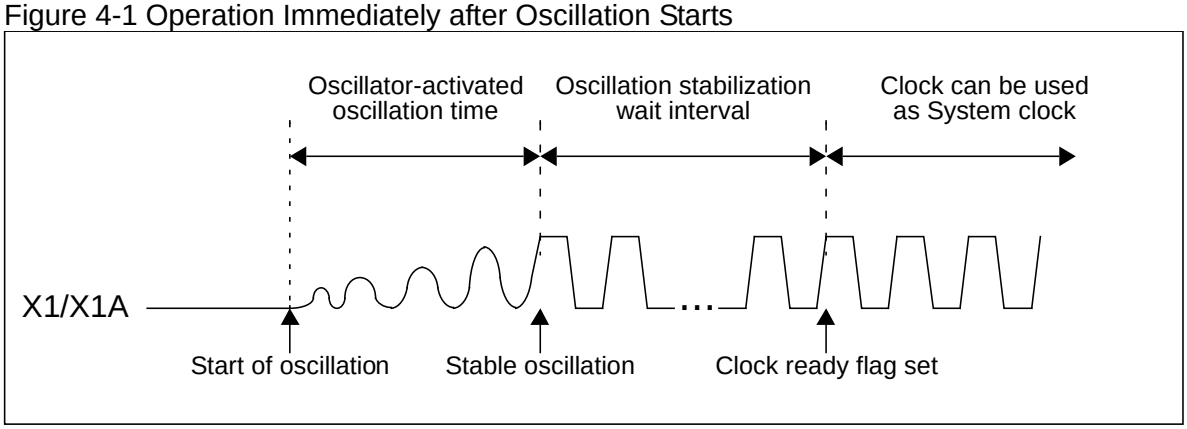
A Power reset or External reset clears all source clock timers and clock ready monitor bits and the corresponding oscillation stabilization wait interval is applied.

Software and Watchdog resets do not affect the source clock timers and Main and Sub clock monitor bits (no Main/Sub oscillation stabilization wait interval applied if clock was enabled before reset), but clear the RC and PLL clock monitor bits (PLL is disabled and RC oscillation stabilization time is applied).

Clock stop resets clear the source clock timer and clock monitor bit of the clock that caused the reset. This clock should not be used any more. The PLL is also disabled and the RC oscillation stabilization time applied.

Figure 4-1 shows the operation of the oscillators directly after activation.

Figure 4-1 Operation Immediately after Oscillation Starts



● RC Clock Stabilization Interval

The RC clock stabilization time is a fixed number of RC clock cycles (see datasheet). However after a Power reset or an External reset (RSTX falling edge), an additional wait time of 200 RC clock cycle is applied by the reset extension circuit (see "Section 8.3 Startup after Power and External reset" for more details).

● Main Clock Stabilization Interval

The Main clock stabilization time can be selected with the MCST[2:0] bits of the CKSSR register. 8 settings are possible as described in Section "6.3 Clock Stabilization Select Register (CKSSR)".

● Sub Clock Stabilization Interval

The Sub clock stabilization time can be selected with the SCST[1:0] bits of the CKSSR register. 4 settings are possible as described in Section "6.3 Clock Stabilization Select Register (CKSSR)".

● PLL Clock Stabilization Interval

The PLL clock stabilization time can be selected with the PCST bit of the CKSSR register. 2 settings are possible as described in Section "6.3 Clock Stabilization Select Register (CKSSR)".

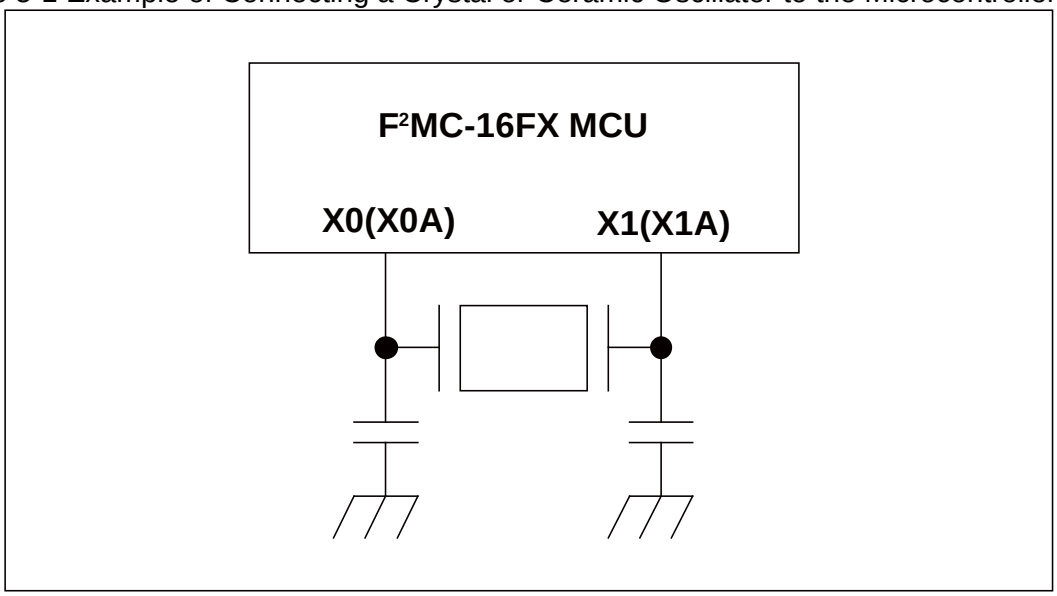
5. Connection of an Oscillator or an External Clock to the Microcontroller

The F²MC-16FX microcontroller contains two clock generation circuits for external oscillators, the Main oscillation circuit and the Sub oscillation circuit. Connecting an external oscillator to these circuits generates the Main clock and the Sub clock. Alternatively, externally generated clocks can be input to the micro controller.

■ Example of Connecting a Crystal or Ceramic Oscillator to the Microcontroller

Connect a crystal or ceramic oscillator as shown in the following diagram.

Figure 5-1 Example of Connecting a Crystal or Ceramic Oscillator to the Microcontroller



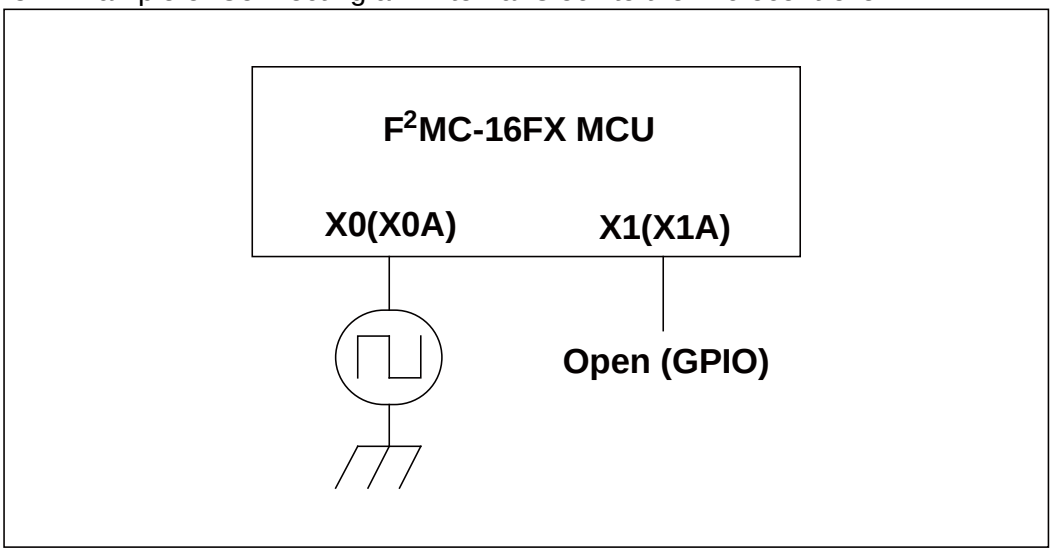
If this connection is used, then the CILCR:FCI bit* must be set to "0".

■ Example of Connecting an External Clock to the Microcontroller

When using an external clock, connect the clock signal to the X0 or X0A pin as shown in the example in Figure 5-2. Pin X1 must be open. Pin X1A can be used as general purpose I/O port.
The external clock input mode of the Main oscillator is activated by setting the CILCR:FCI bit to "1" before switching the device to the external Main clock.
The external clock input mode of the Sub oscillator is activated by setting the Sub Oscillator Configuration Marker SOCM in the ROM Configuration Block A (RCBA) to AA_H.
For the permitted frequency ranges in external clock input mode, refer to the datasheet.
Fujitsu Semiconductor Limited recommends to always use the external clock input mode when connecting an external clock to the oscillators.

*: For details about this register, see "and Section 6. Registers" of "CHAPTER 7 RESETS AND STARTUP".

Figure 5-2 Example of Connecting an External Clock to the Microcontroller



6. Registers

This section explains the configuration and functions of the registers used for the clock control.

■ List of Clock Control Registers

Abbreviated Register Name	Register Name	Reference
CKSR	Clock Selection Register	See 6.1
CKMR	Clock Monitor Register	See 6.2
CKSSR	Clock Stabilization Select Register	See 6.3
CKFCR	Clock Frequency Control Register	See 6.4
PLLCR	PLL Control Register	See 6.5

6.1. Clock Selection Register (CKSR)

The Clock Selection Register (CKSR) is used to control the System Clock selector 1 and 2 and the oscillation circuits.

■ Configuration of the Clock Selection Register (CKSR)

CKSR								
bit	15	14	13	12	11	10	9	8
	SCE	PCE	MCE	RCE	SC2S1	SC2S0	SC1S1	SC1S0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	1	0	1	1	0	0	0	0

[bit15] SCE: Sub Clock Enable Bit

Bit	Description
0	Sub oscillator stopped
1	Sub oscillator enabled

- This bit is used to enable/stop the Sub oscillation circuit.
- After a Power reset or an External reset, the Sub oscillator is always disabled and the Sub Oscillator Configuration Marker SOCM in the ROM Configuration Block A (RCBA) is read by the Boot ROM program. Depending on the configuration marker, the Sub oscillation circuit can be enabled in oscillation mode or in External clock input mode. At this time, the Sub oscillation stabilization time starts.
- The SCE bit has no function when the Sub oscillator is disabled by the configuration marker.
- Writing "1" to the SCE bit enables the Sub oscillator and writing "0" stops the oscillator.
- This bit is initialized to "1" (oscillator on) by each reset.
- The Sub oscillation stabilization time is applied after enabling the Sub oscillator. This time is defined by the SCST bits of the CKSSR register. After this time has elapsed, the Sub clock monitor bit (SCM) in the CKMR register is set.
- Selecting CLKSC as clock source for CLKS1 or CLKS2 activates the Sub oscillator independently of the setting of the SCE bit. The oscillator can only be stopped when both System clocks are running with a different clock source.

Note:

Do not disable the Sub oscillator if the clock source for the Watchdog is set to Sub clock. Setting SCE to "0" in this case causes a Watchdog reset.

[bit14] PCE: PLL Clock Enable Bit

Bit	Description
0	PLL stopped
1	PLL enabled

- This bit is used to enable/stop the PLL oscillation circuit.
- Writing "1" to this bit enables the PLL and writing "0" stops the PLL.
- MCE must also be set to "1" if the PLL should be enabled because the PLL is using the Main clock as input clock. For MCE = "0", the setting of PCE has no effect.
- This bit is initialized to "0" (PLL stopped) by each reset.
- The PLL stabilization time is applied after stabilization of the Main clock (MCM = "1") and enabling of the PLL. This time is defined by the PCST bit of the CKSSR register. After this time has elapsed, the PLL clock monitor bit (PCM) in the CKMR register is set.
- Selecting CLKPLL as clock source for CLKS1 or CLKS2 activates the PLL independently of the setting of the PCE bit. The PLL can only be stopped when both System clocks are running with different clock sources.

[bit13] MCE: Main Clock Enable Bit

Bit	Description
0	Main oscillator stopped
1	Main oscillator enabled

- This bit is used to enable/stop the Main oscillation circuit.
- Writing "1" to this bit enables the Main oscillator and writing "0" stops the oscillator.
- This bit is initialized to "1" (oscillator on) by each reset.
- The Main oscillation stabilization time is applied after enabling the Main oscillator. This time is defined by the MCST bits of the CKSSR register. After this time has elapsed, the Main clock monitor bit (MCM) in the CKMR register is set.
- Selecting CLKMC or CLKPLL as clock source for CLKS1 or CLKS2 activates the Main oscillator independently of the setting of the MCE bit. The oscillator can only be stopped when both System clocks are running with different clock sources.
- Setting this bit to "0" also disables the PLL clock (independently of the PCE setting) and sets PCM to "0" unless the PLL clock is used as System clock 1 or 2.

Note:

Do not disable the Main oscillator if the clock source for the Watchdog is set to Main clock. Setting MCE to "0" in this case causes a Watchdog reset.

[bit12] RCE: RC Clock Enable Bit

Bit	Description
0	RC oscillator stopped
1	RC oscillator enabled

- This bit is used to enable/stop the internal RC oscillator.
- Writing "1" to this bit enables the RC oscillator and writing "0" stops the oscillator.
- This bit is initialized to "1" (oscillator on) by each reset.
- The RC oscillation stabilization time is applied after enabling the RC oscillator. This time is specified in the datasheet (256 clock cycles for fast RC clock and 16 cycles for slow RC clock). After this time has elapsed, the RC clock monitor bit (RCM) in the CKMR register is set.
- Selecting CLKRC as clock source for CLKS1 or CLKS2 activates the RC oscillator independently of the setting of the RCE bit. The oscillator can only be stopped when both System clocks are running with a different clock source.

Notes:

- Do not disable the RC oscillator if the Clock stop detection reset is enabled. Setting RCE to "0" in this case causes a Clock stop detection reset.
- Do not disable the RC oscillator if the clock source for the Watchdog is set to RC clock. Setting RCE to "0" in this case causes a Watchdog reset.

[bit11, bit10] SC2S1 and SC2S0: System Clock 2 Select Bits

- These bits control the System Clock 2 Selector for the source of the Peripheral clock 2 according to the following table:

bit11	bit10	Description
0	0	RC clock (set CLKS2 to CLKRC)
0	1	Main clock (set CLKS2 to CLKMC)
1	0	PLL clock (set CLKS2 to CLKPLL)
1	1	Sub clock (set CLKS2 to CLKSC)

- Any reset initializes these bits to "00" (RC clock).
- Changing to a different clock source is delayed until the selected clock is ready (means the corresponding clock ready monitor bit in the Clock Monitor Register CKMR is set).
- If the selected clock is ready, then the clock source transition will be executed by starting the synchronization mechanism. The System Clock 2 Monitor bits (SC2M) of the Clock Monitor register indicate when this clock switching has completed.
- Writing a second value to this register before switching to the mode selected beforehand has completed cancels the first transition request.
- A transition to a clock which is not available (oscillator failed or is not existent) is not possible.

Note:

In Stop mode, the System Clock 2 Selector directly changes to the clock selected by the SC2S bits, independent of the corresponding clock ready monitor bit (in Stop mode, all clock monitor bits are cleared). Hence do not select an unavailable clock before switching to Stop mode.

[bit9, bit8] SC1S1 and SC1S0: System Clock 1 Select Bits

- These bits control the System Clock 1 Selector for the source of the Bus clock and the Peripheral clock 1 according to the following table:

bit9	bit8	Description
0	0	RC clock mode (set CLKS1 to CLKRC)
0	1	Main clock mode (set CLKS1 to CLKMC)
1	0	PLL clock mode (set CLKS1 to CLKPLL)
1	1	Sub clock mode (set CLKS1 to CLKSC)

- These bits control the CPU clock mode.
- Any reset initializes these bits to "00" (RC clock mode).
- A transition to a different clock mode is delayed until the selected clock is ready (means the corresponding clock ready monitor bit in the Clock Monitor Register CKMR is set).
- If the selected clock is ready, then the clock mode transition will be executed by starting the synchronization mechanism. The System Clock 1 Monitor bits (SC1M) of the Clock Monitor register indicate when this clock switching has completed.
- Writing a second value to this register before switching to the mode selected beforehand has completed cancels the first transition request.
- A transition to a clock which is not available (oscillator failed or is not existent) is not possible.

Note:

In Stop mode, the System Clock 1 Selector directly changes to the clock selected by the SC1S bits, independently of the corresponding clock ready monitor bit (in Stop mode, all clock monitor bits are cleared). Hence do not select an unavailable clock before switching to Stop mode.

6.2. Clock Monitor Register (CKMR)

The Clock Monitor Register (CKMR) is used to check the current status of the System clocks (Clock mode) and the status of the oscillation circuits.

■ Configuration of the Clock Monitor Register (CKMR)

CKMR								
bit	15	14	13	12	11	10	9	8
	SCM	PCM	MCM	RCM	SC2M1	SC2M0	SC1M1	SC1M0
Attribute	R	R	R	R	R	R	R	R
Initial value	X	X	X	X	X	X	X	X

[bit15] SCM: Sub Clock Monitor Bit

Bit	Description
0	Sub oscillator is not ready
1	Sub oscillator is ready

- This bit indicates if the sub oscillator is ready or not.
- SCM = "1" means that the sub oscillator is ready and can be used. If SCM = "1" although SCE was set to "0", then the Sub oscillator has not been disabled because the Sub clock is used for System Clock 1 or 2.
- SCM = "0" means that the sub oscillator is either disabled or the sub oscillation stabilization time is in effect.
- A Power or External reset and a Sub clock stop detection reset initializes this bit to "0".
- A Main clock stop detection, Software or Watchdog reset does not reset this bit.

Note:

If the Sub clock stabilization time setting CKSSR:SCST[1:0] is changed from a "short" time to a "longer" timer after setting of SCM, then SCM will not be cleared, even when the newly selected time is not yet expired. This includes a change of CKSSR:SCST[1:0] by reset assertion.

[bit14] PCM: PLL Clock Monitor Bit

Bit	Description
0	PLL is not ready
1	PLL is ready

- This bit indicates if the PLL is ready or not.
- PCM = "1" means that the PLL clock is ready and can be used. If PCM = "1" although PCE was set to "0", then the PLL has not been disabled because the PLL is used for System Clock 1 or 2.
- PCM = "0" means that the PLL is either disabled or the PLL stabilization time is in effect.
- Any reset initializes this bit to "0" (PLL disabled).

Note:

If the PLL clock stabilization time setting CKSSR:PCST is changed from "0" to "1" after setting of PCM, then PCM will not be cleared, even when the newly selected time is not yet expired.

[bit13] MCM: Main Clock Monitor Bit

Bit	Description
0	Main oscillator is not ready
1	Main oscillator is ready

- This bit indicates if the main oscillator is ready or not.
- MCM = "1" means that the main oscillator is ready and can be used. If MCM = "1" although MCE was set to "0", then the Main oscillator has not been disabled because the Main clock or PLL clock is used for System Clock 1 or 2.
- MCM = "0" means that the main oscillator is either disabled or the main oscillation stabilization time is in effect.
- A Power or External reset (RSTX falling edge) and a Main clock stop detection reset initializes this bit to "0". See "Section 8.3 Startup after Power and External reset" for more details regarding the effect of RSTX.
- A Sub clock stop detection, Software or Watchdog reset does not reset this bit.

Note:

If the Main clock stabilization time setting CKSSR:MCST[2:0] is changed from a "short" time to a "longer" timer after setting of MCM, then MCM will not be cleared, even when the newly selected time is not yet expired. This includes a change of CKSSR:MCST[2:0] by reset assertion.

[bit12] RCM: RC Clock Monitor Bit

Bit	Description
0	RC oscillator is not ready
1	RC oscillator is ready

- This bit indicates if the internal RC oscillator is ready or not.
- RCM = "1" means that the RC oscillator is ready and can be used. If RCM = "1" although RCE was set to "0", then the RC oscillator has not been disabled because the RC clock is used for System Clock 1 or 2.
- RCM = "0" means that the RC oscillator is either disabled or the RC oscillation stabilization time is in effect.
- Any reset initializes this bit to "0" and stops the operation of the MCU. After the RC oscillation stabilization wait time, this bit is set to "1" and the operation of the MCU resumes with the execution of the reset sequence.

[bit11, bit10] SC2M1 and SC2M0: System Clock 2 Monitor Bits

- These bits indicate which clock is currently used for the System Clock 2 according to the following table:

bit11	bit10	Description
0	0	CLKS2 is set to CLKRC (RC clock)
0	1	CLKS2 is set to CLKMC (Main clock)
1	0	CLKS2 is set to CLKPLL (PLL clock)
1	1	CLKS2 is set to CLKSC (Sub clock)

- The read value of the SC2M[1:0] bits can differ from the value written to the SC2S[1:0] System Clock 2 Select bits. This indicates that a requested clock mode transition has not been completed yet.
- An ongoing transition of the clock mode (synchronization mechanism in effect) is indicated as follows: Active clock mode shown by the SC2M[1:0] bits differs from the SC2S[1:0] setting although clock selected by SC2S[1:0] bits is ready (clock monitor bit "1"). The SC2M bits are updated after completion of the clock mode transition.
- Clock mode switching is not possible when the selected clock is not ready (clock monitor bit of the clock selected by the SC2S[1:0] bits is "0").

[bit9, bit8] SC1M1 and SC1M0: System Clock 1 Monitor Bits

- These bits indicate which clock is currently used for the System Clock 1 according to the following table:

bit9	bit8	Description
0	0	CLKS1 is set to CLKRC (RC clock)
0	1	CLKS1 is set to CLKMC (Main clock)
1	0	CLKS1 is set to CLKPLL (PLL clock)
1	1	CLKS1 is set to CLKSC (Sub clock)

- These bits show the currently active CPU clock mode.
- The read value of the SC1M[1:0] bits can differ from the value written to the SC1S[1:0] System Clock 1 Select bits. This indicates that a requested clock mode transition has not been completed yet.
- An ongoing transition of the clock mode (synchronization mechanism in effect) is indicated as follows: Active clock mode shown by the SC1M[1:0] bits differs from the SC1S[1:0] setting although clock selected by SC1S[1:0] bits is ready (clock monitor bit "1"). The SC1M[1:0] bits are updated after completion of the clock mode transition.
- Clock mode switching is not possible when the selected clock is not ready (clock monitor bit of the clock selected by the SC1S[1:0] bits is "0").

6.3. Clock Stabilization Select Register (CKSSR)

The Clock Stabilization Select Register (CKSSR) is used to select the stabilization times for the oscillation circuits, the PLL and for controlling the feedback resistors of the Main and Sub oscillation circuits.

■ Configuration of the Clock Stabilization Select Register (CKSSR)

CKSSR								
bit	7	6	5	4	3	2	1	0
	-	-	PCST	SCST1	SCST0	MCST2	MCST1	MCST0
Attribute	X	X	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	X	X	1	1	1	1	1	1

[bit7, bit6] -: Undefined

- Always write "0" to these bits.
- The read value of these bits is undefined.
- Read modify write operations to this register have no effect on these bits.

[bit5] PCST: PLL Clock Stabilization Time Select Bit

- This bits selects the stabilization time for the PLL clock according to the following table:

Bit	Description
0	2^{12} / CLKMC (approx. 1ms)
1	2^{14} / CLKMC (approx. 4ms)

- Any reset initializes this bit to "1".
- This bit defines after how many Main clock (CLKMC) cycles the PLL Clock Monitor (PCM) bit will be set, allowing to use the PLL clock as System clock.
- The PLL clock stabilization time starts counting after stabilization of the Main clock (MCM = "1") and after setting PCE to "1"
- Setting PCST to "1" is only permitted for a Main clock (CLKMC) frequency up to 8MHz.

Note:

If PCST is set to "0", then CKMR:PCM will be set after 2^{12} / CLKMC cycles. Changing PCST to "1" afterwards does not clear the CKMR:PCM bit.

[bit4, bit3] SCST1 to SCST0: Sub Clock Stabilization Time Select Bits

- These bits select the stabilization time for the Sub oscillation circuit according to the following table:

bit4	bit3	Description
0	0	2^{12} / CLKSC (125ms)
0	1	2^{14} / CLKSC (0.5s)
1	0	2^{15} / CLKSC (1s)
1	1	2^{16} / CLKSC (2s)

- Any reset initializes these bits to "11".
- These bits define after how many Sub clock (CLKSC) cycles the Sub Clock Monitor (SCM) bit will be set, allowing to use the Sub clock as System clock.

Note:

The oscillation stabilization wait interval must be set to a value appropriate for the oscillator used.
If a shorter stabilization time is selected by software and the oscillator is started, then CKMR:SCM will be set as soon as this shorter time is elapsed. Selecting a longer stabilization time afterwards does not clear the CKMR:SCM bit.

[bit2 to bit0] MCST2 to MCST0: Main Clock Stabilization Time Select Bits

- These bits select the stabilization time for the Main oscillation circuit according to the following table:

bit2	bit1	bit0	Description
0	0	0	2^{10} / CLKMC (approx. 256μs)
0	0	1	2^{12} / CLKMC (approx. 1ms)
0	1	0	2^{13} / CLKMC (approx. 2ms)
0	1	1	2^{14} / CLKMC (approx. 4ms)
1	0	0	2^{15} / CLKMC (approx. 8ms)
1	0	1	2^{16} / CLKMC (approx. 16ms)
1	1	0	2^{17} / CLKMC (approx. 32ms)
1	1	1	2^{18} / CLKMC (approx. 65ms)

- Any reset initializes these bits to "111".
- These bits define after how many Main clock (CLKMC) cycles the Main Clock Monitor (MCM) bit will be set, allowing to use the Main clock as System clock.
- Do not use a setting which results in stabilization time below 100μs in case a Main clock frequency above 10MHz is used. This is required to ensure a sufficient stabilization time for the voltage regulator at wakeup from Stop mode to Main Run mode.

Note:

The oscillation stabilization wait interval must be set to a value appropriate for the oscillator used.
If a shorter stabilization time is selected by software and the oscillator is started, then CKMR:MCM will be set as soon as this shorter time is elapsed. Selecting a longer stabilization time afterwards does not clear the CKMR:MCM bit.

6.4. Clock Frequency Control Register (CKFCR)

The Clock Frequency Control Register (CKFCR) is used to control the Peripheral clock dividers (1 and 2), the Bus clock divider and the RC oscillator frequency.

■ Configuration of the Clock Frequency Control Register (CKFCR)

The register can be accessed 16-bit wide (CKFCR) and 8-bit wide (low byte: CKFCRL, high byte: CKFCRH).

CKFCR								
bit	15	14	13	12	11	10	9	8
	PC2D3	PC2D2	PC2D1	PC2D0	PC1D3	PC1D2	PC1D1	PC1D0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

bit	7	6	5	4	3	2	1	0
	BCD3	BCD2	BCD1	BCD0	-	-	-	RCFS
Attribute	R/W	R/W	R/W	R/W	-	-	-	R/W
Initial value	0	0	0	0	X	X	X	1

[bit15 to bit12] PC2D3 to PC2D0: Peripheral Clock 2 Division Select Bits

· These bits control the clock divider for the Peripheral clock (CLKP2) according to the following table:

bit15	bit14	bit13	bit12	Description
0	0	0	0	CLKP2 is CLKS2 (divided by 1)
0	0	0	1	CLKP2 is CLKS2 divided by 2
0	0	1	0	CLKP2 is CLKS2 divided by 3
...	...	...	...	CLKP2 is CLKS2 divided by PC2D[3:0] + 1
1	1	1	0	CLKP2 is CLKS2 divided by 15
1	1	1	1	CLKP2 is CLKS2 divided by 16

· These bits are initialized to "0000" (CLKP2 = CLKS2) by each reset.

[bit11 to bit8] PC1D3 to PC1D0: Peripheral Clock 1 Division Select Bits

- These bits control the clock divider for the Peripheral clock (CLKP1) according to the following table:

bit11	bit10	bit9	bit8	Description
0	0	0	0	CLKP1 is CLKS1 (divided by 1)
0	0	0	1	CLKP1 is CLKS1 divided by 2
0	0	1	0	CLKP1 is CLKS1 divided by 3
...	...	...	...	CLKP1 is CLKS1 divided by PC1D[3:0] + 1
1	1	1	0	CLKP1 is CLKS1 divided by 15
1	1	1	1	CLKP1 is CLKS1 divided by 16

- These bits are initialized to "0000" (CLKP1 = CLKS1) by each reset.

[bit7 to bit4] BCD3 to BCD0: Bus Clock Division Select Bits

- These bits control the clock divider for the Bus clock (CLKB) according to the following table:

bit7	bit6	bit5	bit4	Description
0	0	0	0	CLKB is CLKS1 (divided by 1)
0	0	0	1	CLKB is CLKS1 divided by 2
0	0	1	0	CLKB is CLKS1 divided by 3
...	...	...	...	CLKB is CLKS1 divided by BCD[3:0] + 1
1	1	1	0	CLKB is CLKS1 divided by 15
1	1	1	1	CLKB is CLKS1 divided by 16

- These bits are initialized to "0000" (CLKB = CLKS1) by each reset.

[bit3 to bit1] -: Undefined

- Always write "0" to these bits.
- The read value of these bits is undefined.
- Read modify write operations to this register have no effect on these bits.

[bit0] RCFS: RC Clock Frequency Select Bit

Bit	Description
0	Nominal RC clock frequency set to 100kHz
1	Nominal RC clock frequency set to 2MHz

- This bit is used to set the clock frequency of the internal RC oscillator.
- Writing "1" to this bit sets the nominal RC clock frequency to 2MHz and writing "0" sets the nominal frequency to 100kHz.
- For minimum and maximum frequencies in both settings, please refer to the Datasheet.
- This bit is initialized to "1" (2MHz) by each reset.

Note:

The RC clock frequency can be changed at any time. However consider that the Clock stop detection circuit and the Watchdog reset can operate with the RC clock and that changing the RC clock frequency affects the behavior of these circuits. The Watchdog reset module has an operation mode where changing the RC clock frequency is not allowed and causes a Watchdog reset. Please read the corresponding description in the chapters about the Watchdog reset and the Clock stop detection reset for more details.

6.5. PLL Control Register (PLLCR)

The PLL Control Register (PLLCR) is used to control all functions of the PLL multiplier circuit.

■ Configuration of the PLL Control Register (PLLCR)

The register can be accessed 16-bit wide (PLLCR) and 8-bit wide (low byte: PLLCRL, high byte PLLCRH).

PLLCR								
bit	15	14	13	12	11	10	9	8
	-	-	-	-	-	-	-	-
Attribute	-	-	-	-	-	-	-	-
Initial value	X	X	X	X	X	X	X	X

bit	7	6	5	4	3	2	1	0
	VMS2	VMS1	VMS0	-	PMS3	PMS2	PMS1	PMS0
Attribute	R/W	R/W	R/W	-	R/W	R/W	R/W	R/W
Initial value	0	0	0	X	0	0	0	0

[bit15 to bit8] -: Undefined

- Always write "0" to these bits.
- The read value of these bits is undefined.
- Read modify write operations to this register have no effect on these bits.

[bit7 to bit5] VMS2 to VMS0: VCO Clock Multiplier Select Bits

- These bits control the VCO clock multiplier factor according to the following table:

bit7	bit6	bit5	Description
0	0	0	VCO Clock is CLKPLL multiplied by 2
0	0	1	VCO Clock is CLKPLL multiplied by 4
0	1	0	VCO Clock is CLKPLL multiplied by 6
0	1	1	VCO Clock is CLKPLL multiplied by 8
1	0	0	VCO Clock is CLKPLL multiplied by 10
1	0	1	VCO Clock is CLKPLL multiplied by 12
1	1	0	VCO Clock is CLKPLL multiplied by 14
1	1	1	VCO Clock is CLKPLL multiplied by 16

- These bits are initialized to "000" by each reset.
- The VCO clock multiplier setting defines the relation between the PLL output clock frequency (CLKPLL) and the output frequency of the PLL internal VCO.
- These bits must be set before the PLL is enabled by writing "1" to the CKSR: PCE bit or by selecting the PLL clock as source for CLKS1 or CLKS2.
- Do not change the setting of these bits after activation of the PLL.
- The VCO clock multiplier factor should be set depending on the PLL input clock CLKMC and the PLL multiplier setting. See "and Section 3. Configuration of the PLL" for more details.

[bit4] -: Undefined

- Always write "0" to this bit.
- The read value of this bit is undefined.
- Read modify write operations to this register have no effect on this bit.

[bit3 to bit0] PMS3 to PMS0: PLL Clock Multiplier Select Bits

- These bits control the PLL clock multiplier factor according to the following table:

bit3	bit2	bit1	bit0	Description
0	0	0	0	CLKPLL is CLKMC (multiplied by 1)
0	0	0	1	CLKPLL is CLKMC multiplied by 2
...	...	...	...	CLKPLL is CLKMC multiplied by PMS[3:0] + 1
1	1	0	0	CLKPLL is CLKMC multiplied by 13
1	1	0	1	not approved
1	1	1	0	
1	1	1	1	

- These bits are initialized to "0000" by each reset.
- The PLL multiplier setting defines the ratio between the Main clock frequency (CLKMC) and the PLL output frequency (CLKPLL).
- These bits must be set before the PLL is enabled by writing "1" to the CKSR: PCE bit or by selecting the PLL clock as source for CLKS1 or CLKS2.
- Do not change the setting of these bits after activation of the PLL.
- The permitted range of settings depends on the used Main clock frequency and the permitted frequency range for CLKVCO and CLKS1/CLKS2 (see Datasheet).

CHAPTER: RESETS AND STARTUP

This chapter describes the resets and the startup of the F^{2}MC-16FX family microcontrollers.

Management Code: 96F6RESET-E01.0

1. Overview

If a reset is generated, the CPU immediately stops the current execution process and waits for the reset to be cleared. The CPU then begins with the Boot ROM program execution. Depending on the Mode pin settings, the Boot ROM program branches into different operating modes or starts the user program.
The five causes of a reset are as follows:
- Power reset (Power-on or Low voltage)
- External reset request via the RSTX pin
- Clock stop detection
- Software reset request
- Watchdog timer overflow

■ Causes of a Reset

Table 1-1 lists the causes of a reset.

Table 1-1 Causes of a Reset

Type of reset	Cause	System clock CLKS1 and CLKS2 after reset release	Reading of mode pin setting	RAM contents
Power	When the power is turned on or when Vcc falls below the Low Voltage Detector level (LVL in CILCR register). Please refer to the Datasheet for the analog values.	RC clock (CLKRC)	Yes	Not guaranteed
External pin	L level input to RSTX pin			Maintained*
Clock stop detection	Failure of external oscillator		No	Not guaranteed
Software	A "1" is written to the SRSTG bit of the Reset Configuration Register (RCR).			Maintained
Watchdog timer	Watchdog timer overflow.			

*: Except for bytes at addresses $7FFC_H$ to $7FFF_H$.

The contents of the registers in the GPR bank 00 and 01 as well as the RAM 4 bytes of addresses $7FFC_H$ to $7FFF_H$ are undefined after each reset. The contents of the other GPR registers are either maintained or undefined (depending on reset cause as described above).
The MCU is running with the RC clock set to nominal 2MHz after each reset.

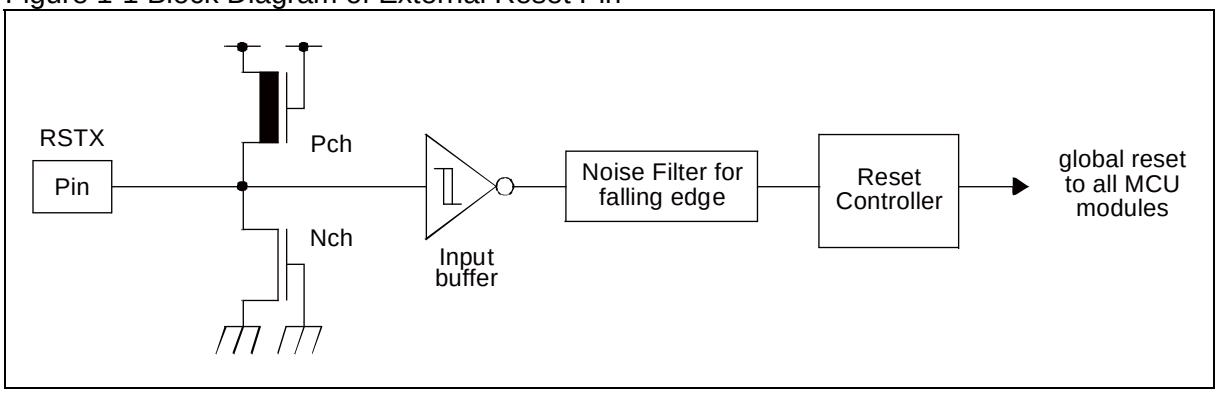
● Power Reset

A power reset is generated when the power is turned on with a power on rise time as specified in the datasheet (power-on reset) or when the low voltage detector detects that the power supply Vcc falls below a certain value as specified in the LVL bits of the CILCR register (low voltage reset). Refer to the Datasheet for the analog LVL levels. A power reset is internally prolonged by the reset extension circuit to a minimum length of 200 RC clock cycles
The contents of internal RAM and all registers with initial value "X" is undefined after a Power reset.

● External Pin Reset

An External pin reset is generated by a L level input to the external reset pin (RSTX pin). This signal is internally prolonged by the reset extension circuit to a minimum length of 200 RC clock cycles.
Because the external reset is asynchronous, it becomes active with little delay (some 10ns) after assertion, disregarding whether the device is clocked or not.
The contents of internal RAM and all registers which are not reset (initial value "X") is maintained except for the 4 bytes at the address $7FFC_H$ to $7FFF_H$.

Figure 1-1 Block Diagram of External Reset Pin



● Clock Stop Detection Reset

A Clock stop detection reset is generated upon a failure of the Main or Sub clock oscillator depending on the current operation mode and the setting of the Reset Configuration Register (RCR). See "and Section 5. Operation of the Clock Stop Detection Function and Reset" for more details.
The contents of internal RAM memory and all registers which are not reset (initial value "X") cannot be guaranteed after a Clock stop detection reset.

● Software Reset

A Software reset is an internal reset generated by writing "1" to the SRSTG bit of the Reset Configuration Register (RCR).
The contents of internal RAM and all registers which are not reset (initial value "X") is maintained.

● Watchdog Timer Reset

A Watchdog timer reset is generated when the watchdog timer is activated but not cleared within the defined period as specified in the "CHAPTER 10 WATCHDOG TIMER AND WATCHDOG RESET"
The contents of internal RAM and all registers which are not reset (initial value "X") is maintained.

Note:

The contents of the registers in the GPR bank 00 and 01 as well as the addresses $7FFC_H$-$7FFF_H$ is undefined after each reset. The contents of the other GPR registers are either maintained or undefined (depending on reset cause as Table 1-1).

Note:

Status of pins at reset: all pins are in Hi-Z state. See also Table 5-1.

---

2. Reset, System Clock and Stabilization Wait Times

The F^{2}MC-16FX family has five reset causes. The System clocks (CLKS1 and CLKS2) are set to RC clock after each reset. The stabilization wait time depends on the reset cause and the status of the RC oscillator when the reset occurs.

■ Reset Causes and Stabilization Wait Times

Table 2-1 summarizes which stabilization times are applied for the different reset causes.

Table 2-1 Reset Cause and Stabilization Wait Times

Reset cause	Stabilization wait time
Power reset	200 RC clock cycles Power reset extension time (approximately 100µs at 2MHz nominal RC clock frequency) plus the RC clock stabilization time (see datasheet). This wait time starts AFTER reaching a valid power supply (external and internal voltage).
External reset	200 RC clock cycles External reset extension time (approximately 100µs at 2MHz nominal RC clock frequency) plus the RC clock stabilization time (see datasheet). The External reset extension time starts already at the falling edge of the RSTX input signal while the RC clock stabilization time is applied after RSTX release or after the External reset extension time (after MAIN_RST release, Figure 3-2)
Clock stop detection reset	RC clock stabilization time is applied (see datasheet)
Software reset	
Watchdog timer reset	

Each reset activates the RC oscillator and Main oscillator, stops the PLL and resets the Clock Stabilization Select Register (CKSSR). The effect on the Sub oscillator depends on the reset type.
A Power or External reset clears all clock ready monitor bits and stops the Sub oscillator. The Sub oscillator can be activated by the Boot ROM program depending on the RCBA:SOCM marker.
Other types of resets only clear the PLL and RC clock ready monitor bit, but not the Main and Sub clock ready monitor bits. Active clocks stay active and disabled clocks are activated and become ready after the stabilization time defined in the CKSSR register (including the Sub oscillator if activated by the RCBA:SOCM marker before).
See "CHAPTER 6 CLOCKS", for more details about selected clocks and oscillation stabilization wait times.

■ Stabilization Wait Time in Case of an External Reset

The stabilization wait time in case of an External reset consists of two parts, the reset extension time and the RC clock stabilization time. The External reset extension time starts counting with the assertion of RSTX (falling edge) while the RC clock stabilization time starts counting after RSTX release. Hence the start of the program execution depends on the assertion time of RSTX as follows:

● External Reset asserted for more than 200 RC Clock Cycles

The reset extension time is already expired in this case. Hence the CPU starts executing the Boot ROM program after the RC clock stabilization time after clearing the External reset.

● External Reset asserted for less than 200 RC Clock Cycles

The reset extension time is not expired in this case. Hence the execution of the Boot ROM program by the CPU is delayed until the reset extension time plus the RC clock stabilization is expired.

■ Stabilization of the Main Oscillator

The initial value of the Main oscillation stabilization wait time selector is set to 2^{18} / Main clock cycles (MCST[2:0] = "111"). This value can be overwritten directly after start of the User program execution (when the MCU is running in RC clock mode). Select a value suitable for the used oscillator.
The Main oscillation stabilization wait time is measured with the Main clock counter which starts counting when the reset extension time of a Power or External reset is expired. Hence if RSTX is asserted for more than 200 RC clock cycles, then the Main oscillation stabilization wait time takes place already when RSTX is still asserted.
This allows controlling the Main oscillation stabilization time by RSTX. Release RSTX after stabilization of the Main oscillator and set MCST[2:0] to a value that already expired. However before disabling the Main oscillator (by writing to the MCE bit or by going to Stop mode), a suitable value must be written to the MCST[2:0] bits because this value will be used after reactivation of the Main oscillator.

■ Stabilization of the Sub Oscillator

The initial value of the Sub oscillation stabilization wait time selector is set to 2^{16} / Sub clock cycles (SCST[1:0] = "11_B"). This value can be overwritten directly after start of the User program execution (when the MCU is still running in RC clock mode). Select a value suitable for the used oscillator.
The Sub oscillation stabilization wait time is measured with the Sub clock counter which starts counting after activation of the oscillator by the Boot ROM program.

3. Startup after Power and External Reset

After a Power or External reset event, the MCU waits for the stabilization of the power supply and the RC oscillator. Then the MCU starts executing the Boot ROM program with the RC clock as clock source (RC Run mode).
A transition to another clock mode (Main clock, PLL clock or Sub clock) is possible after stabilization of the respective clock. The stabilization times for the external oscillators can be adjusted to the characteristics of the connected oscillators.

■ Initialization after Startup

After startup, the MCU must be reset to put all registers in the initial state. This is possible by applying an external reset or by using the Power reset functions.

● Initialization by External Reset

Applying a low level at the RSTX input during or after switching on the power supply initializes the complete MCU. Reset extension and stabilization wait times are applied as specified below.

● Initialization by Power Reset

The MCU can also be started without asserting an External reset by using the Power reset functions.
The on-chip power-on detector generates a reset pulse after switching on the power supply. This also activates the low voltage detector and reset function which keeps the MCU in the reset state until the power supply voltage level has reached the default threshold (Level 0).
Reset extension and stabilization wait times are applied as specified below.

■ Reset Extension

Power and External reset events are extended by the reset extension circuit to guarantee the stabilization of the voltage regulators and comparators before program execution starts.
The reset extension counter is an 8 bit counter that is initialized by a falling edge at the RSTX input pin, by the power-on detector and by a low voltage reset. It is clocked by the RC clock CLKRC.
This counter keeps the internal reset signal active for 200 RC clock cycles after the last falling edge of the RSTX input pin or after the last power-on or low voltage event was cleared. The reset output signal "MAIN_RST" is deactivated if the power supply was stable for at least 200 RC clock cycles (approximately 100µs at 2MHz nominal RC clock frequency).
If RSTX is asserted for more than 200 RC clock cycles, then this reset extension has no effect.

Figure 3-1 shows a block diagram of the reset extension circuit and Figure 3-2 shows a timing diagram of the External reset extension function.

Figure 3-1 Block Diagram of Reset Extension Circuit

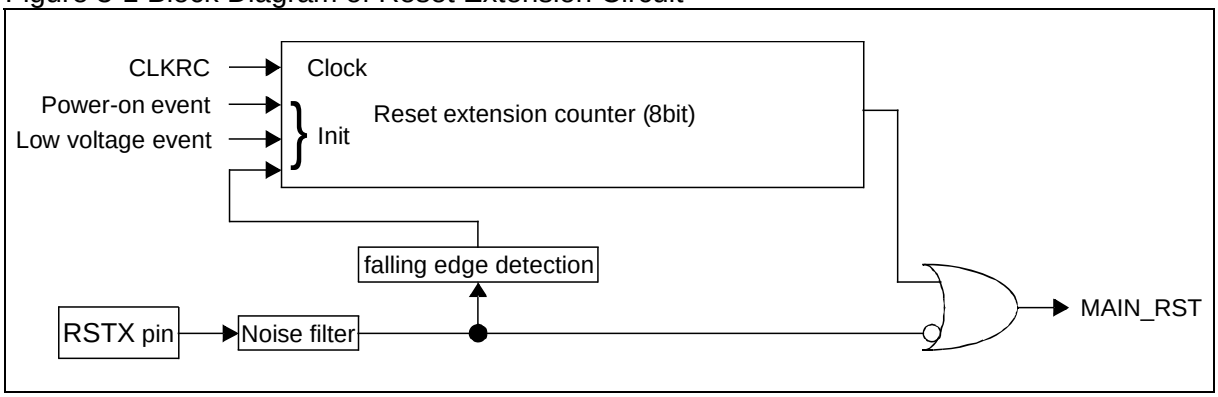
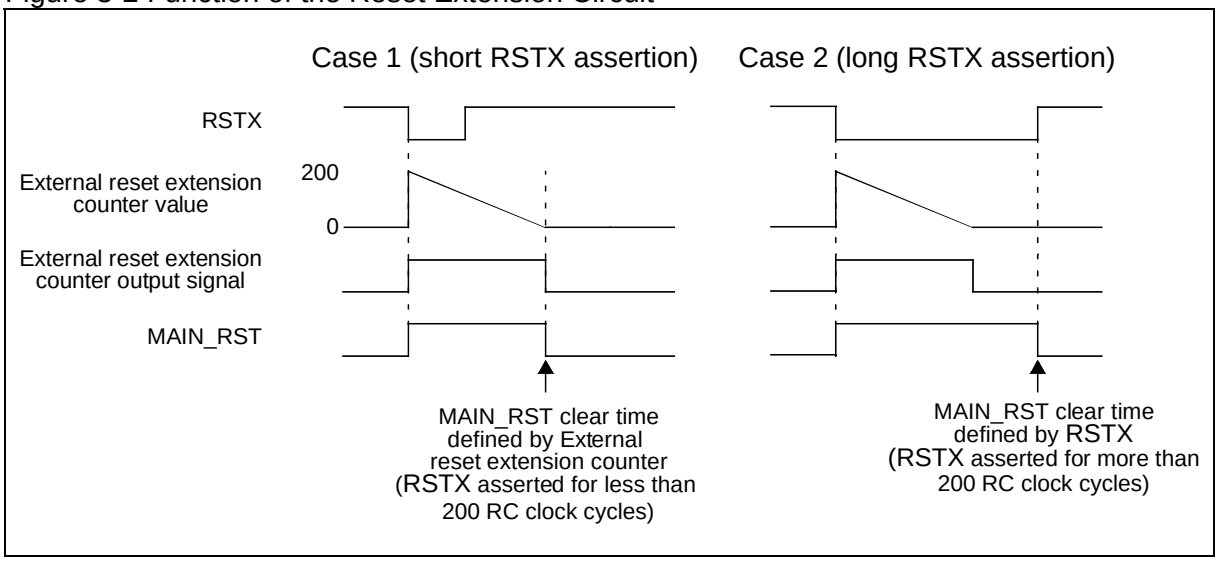


Figure 3-2 Function of the Reset Extension Circuit



■ Source Clock Timers and Clock Ready Monitor Bits

The Source clock timers (RC clock timer, Main clock timer and Sub clock timer) and the clock monitor bits (RCM, MCM, PCM and SCM) are all cleared by a Power or External reset event while the RC and Main oscillators are enabled. The Sub oscillator and PLL are disabled.

● RC Clock Timer, Sub Clock Timer, RCM Bit and SCM Bit

The RC clock timer and Sub clock timer are cleared and stopped by MAIN_RST. Hence the RC clock ready monitor bit RCM and the Sub clock ready monitor bit SCM cannot be set as long as MAIN_RST is active. The RC clock timer starts counting when MAIN_RST is released and the Sub clock timer starts when the Sub oscillator is enabled by the Boot ROM program.

● Main Clock Timer and MCM Bit

The Main clock timer is cleared and stopped during the External or Power reset extension time only. Hence the Main clock timer is activated 200 RC clock cycles after a RSTX falling edge or 200 RC clock cycles after clearing a power-on or low voltage event. Thus the Main oscillation stabilization wait time takes place even if RSTX is still asserted.

● PCM Bit

The PLL is always disabled and the PLL clock ready monitor bit PCM cleared by reset. The PLL must be enabled by software after User program start.

■ Start of Program Execution after Power or External Reset

The RC clock timer starts counting after deactivation of MAIN_RST. After the RC clock stabilization time, the RC clock ready monitor bit RCM will be set and the internal reset signal for the CPU will be released. After reset release, the CPU is always in RC clock mode with the RC clock frequency set to nominal 2MHz and starts executing the Boot ROM program.

The low Voltage Detector level (LVL bits of CILCR register) is initialized to the highest value that allows starting the MCU at the specified minimum Vcc power supply. If a higher detection value should be needed, the register value can be changed accordingly.

See also "and Section 2. Reset, System Clock and Stabilization Wait Times" for more details.

■ Transition to Main Clock Mode

A transition to Main clock mode is done by writing to the System clock select bits of the CKSR register. The transition however is delayed until the Main clock is stabilized which is indicated by the Main Clock ready Monitor bit MCM (set to "1").

The stabilization time of the Main oscillator is defined by the CKSSR: MCST[2:0] bits as described in "and Section 2. Reset, System Clock and Stabilization Wait Times".

Note:

Use of the Fast External Clock Input feature, requires setting the FCI bit of the CILCR register to 1 before switching to main clock mode. Please refer to the Datasheet for the input characteristics of the oscillator pin.

■ Transition to PLL Clock Mode

The PLL must be configured and enabled by software. The PLL stabilization wait time takes place after PLL activation and after stabilization of the Main clock (MCM="1").
A transition to PLL clock mode is possible when the PLL Clock ready Monitor bit PCM is set.
It is possible to directly switch from RC clock mode to PLL clock mode or via Main clock mode.

■ Transition to Sub Clock Mode

The Sub oscillator is enabled by the Boot ROM program controlled by a configuration marker after a Power or External reset. The Sub clock stabilization time starts after this activation.
The stabilization time of the Sub oscillator is determined by the CKSSR: SCST[1:0] bits which is set when the Sub Clock Monitor bit SCM is "1". A transition to Sub clock mode is delayed until SCM is set to "1".

4. Boot ROM Program Execution and Operation Mode and ROM Configuration Block

When the reset signal is released, the MCU starts with the execution of the internal Boot ROM program. The Boot ROM program reads the status of certain pins, which define the operation mode of the MCU. Depending on the pin setting, the MCU activates the Parallel Flash Programming mode, the Serial Communication mode or starts executing the User program.
After the activation of the MCU operation mode, further Boot ROM program execution is defined by the configuration stored in the so-called ROM Configuration Block in Flash/ROM, which also determines the Boot Vector (user program start address).
Finally, the Boot ROM program will start execution of the user program at the Boot Vector.

■ External Bus

Devices with external bus interface do not allow reading the Boot Vector from external memory. The MCU must always be started in Internal Vector mode.

■ Mode Pins

The operation mode of the MCU is defined by up to 3 pins:

- The Mode pin "MD". This pin must be set to "0" for selecting the Internal Vector mode or the Serial Communication mode. It must only be set to "1" for selecting the Parallel Flash programming mode.
- The pin " DEBUG I/F" of the On-chip debugging system is used to select between Internal Vector mode or the Serial Communication mode. This pin is also required for selecting the Parallel Flash programming mode.
- An additional pin " P17_0" is only required for selecting the Parallel Flash programming mode. This pin is shared with a general purpose I/O port. The datasheet specifies which port is shared with this function. For MD=0, this pin has no function.

Table 4-1 Mode Pin Settings

Mode pin setting			Operation mode name	Boot Vector Source	Remarks
MD	P17_0	DEBUG I/F			
0	x	0	Serial Communication mode	-	Used for serial Flash programming
0	x	1	Internal Vector mode	Internal memory or fixed value	Available for all devices
1	0	0	Reserved (Test mode)	-	
1	0	1	Reserved (Test mode)	-	
1	1	0	Reserved (Test mode)	-	
1	1	1	Parallel Flash programming mode	-	Available only for Flash devices

The setting of these 3 pins is internally sampled and can only be changed by a Power or External reset event (the Mode pin data is sampled when the signals MAIN_RST defined in Figure 3-1 is "Block diagram of reset extension circuit" is deactivated). Hence the operation mode cannot be changed by Clock stop detection, Software or Watchdog timer reset event.
The sampled mode pin signals are read by the Boot ROM program after reset release, which determines the operation mode.

■ Operation Modes

● Internal Vector Mode (MD="0", DEBUG I/F="1")

The Internal vector mode is a CPU run mode where the Boot Vector (user program start address) is read from an internal memory at FF:FFDC$_H$ or DF:0040$_H$.
After a Power or External reset (MAIN_RST) and before starting user program, the Boot ROM temporarily scans dedicated UART channels for an external communication request as in Serial Communication mode. Optionally, this scanning can be disabled, if the UART Scan Deactivation Marker is set in the ROM Configuration Block.
The Internal Vector mode is available for all MCU types.

● Serial Communication Mode (MD="0", DEBUG I/F="0")

This mode allows the reading and writing of any memory address by serial communicated read/write commands. Programming of the internal Flash is possible by executing tiny program in RAM.
Despite of dedicated read/write commands, the internal RAM area 0x7C00 to 0x7FFF might also be changed by other commands.
If the ROM/Flash security feature is activated, then reading, writing and programming is only possible after transmitting the correct security key.
See "CHAPTER 28 DUAL OPERATION FLASH MEMORY" and "CHAPTER 29 FLASH SECURITY" for more details.

● Parallel Flash Programming Mode (MD="1", P17_0="1", DEBUG I/F="1")

This mode allows the programming of the internal Flash on a parallel Flash programmer. This mode is available only for Flash devices.
See "CHAPTER 28 DUAL OPERATION FLASH MEMORY" for more details.

● Test Mode (MD="1" and either P17_0 or DEBUG I/F="0")

These modes are reserved for test functions.

■ ROM Configuration Block (RCB)

The ROM Configuration Block is a fixed area in Flash A/ROM and in Flash B (if present), which enables the user to configure the behavior of the Boot ROM. User program should not allocate it for program except for specified configuration data.
The ROM Configuration Block A (RCBA) is located at address DF:0000$_H$ - DF:01FF$_H$ (start address of Flash A or ROM). The ROM Configuration Block B (RCBB) is located at address DE:0000$_H$ - DE:002F$_H$ (start address of Flash B, if present). They are read and processed by Boot ROM. Its content can be set and changed by any method, which writes or erases Flash memory.
RCBA and RCBB configure the security setting of belonging Flash/ROM. In addition, the RCBA also contains the Boot sequence configuration block and the Debug ROM configuration block.

Table 4-2 Structure of ROM Configuration Block A (RCBA)

Offset	Sub-block	Marker	Comment
00_H-01_H	Security and Protection Configuration Block (SPCB)	MSBA	Secure Flash A/ROM by 9999_H, permit access by $FFFF_H$
02_H-11_H		MSUKA	Unlock key for secured Flash A/ROM. Set to "all FF_H" to permit access
12_H-$1B_H$		reserved	
$1C_H$-$1F_H$		FWPAMA	Activation of Flash write protection by $292D3A7B_H$
20_H-27_H		FWPSMA	Sector selection for Flash write protection
28_H-$2F_H$		reserved	
30_H-33_H	Boot Sequence Configuration Block (BSCB)	ABVAM	Selection of Alternative Boot Vector in internal vector mode by $292D3A7B_H$
34_H-37_H		USDM	Deactivation of UART scanning in internal vector mode by $292D3A7B_H$
38_H		UPREM	Enable Pull-up resistors for USARTs
39_H		reserved	
$3A_H$		SOCM	Sub Oscillator Configuration
$3B_H$		LVDRLM	LVD Reset Lock
$3C_H$-$3F_H$		reserved	
40_H-43_H	Boot Vector	ABVA	Alternative Boot Vector Address

Table 4-3 Structure of ROM Configuration Block B (RCBB)

Offset	Sub-block	Marker	Comment
00_H-01_H	Security and Protection configuration block (SPCB)	MSBB	Secure Flash B by 9999_H, permit access by $FFFF_H$
02_H-11_H		MSUKB	Unlock key for secured Flash B. Set to "all FF_H" to permit access
12_H-$1B_H$		reserved	
$1C_H$-$1F_H$		FWPAMB	Activation of Flash write protection by $292D3A7B_H$
20_H-27_H		FWPSMB	Sector selection for Flash write protection
28_H-$2F_H$		reserved	

■ Function Description of RCB Markers

● Security and Protection Configuration Block (SPCB)

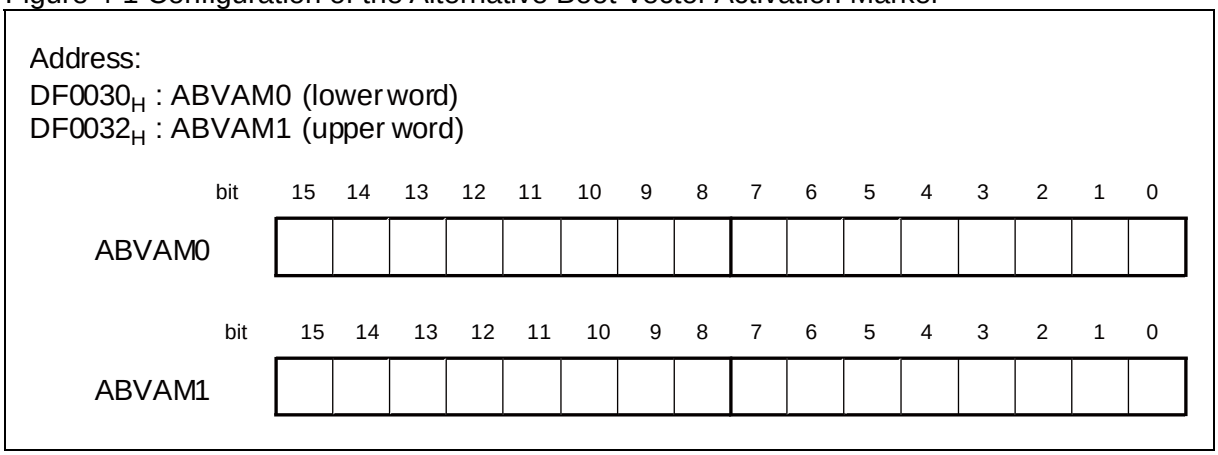
The read security feature prevents the unauthorized read-out of Flash/ROM contents and targets all cases other than read access by application in Internal Vector mode.
For details, see "Section 29.2 Usage".
The write protection feature prevents the unintended writing of Flash memory by application in Internal Vector mode.
For details, see "Section 6.8 Protecting sectors from being erased/written to" in "CHAPTER 28 DUAL OPERATION FLASH MEMORY".

● Boot Sequence Configuration Block (BSCB)

Selecting Alternative Boot Vector

In Internal Vector mode, the Boot Vector (user program start address) can be read from the alternative location DF:0040_H instead of FF:$FFDC_H$.

Figure 4-1 Configuration of the Alternative Boot Vector Activation Marker

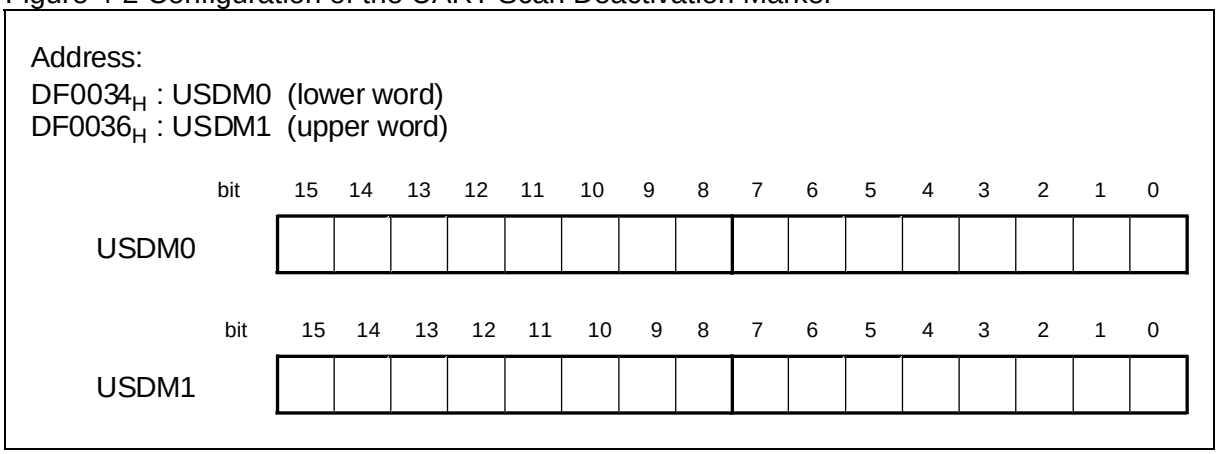


When the content of {ABVAM1, ABVAM0} = $292D3A7B_H$, then the Boot Vector is read from DF:0040_H. For any other value of {ABVAM1, ABVAM0}, the Boot Vector is read from the Boot Vector location at FF:$FFDC_H$.

Disabling temporary UART scan

After a Power or External reset and starting in Internal Vector mode, dedicated UART channels are scanned for a serial communication request as in Serial Communication mode without changing the mode pin setting. This scanning can be disabled to shorten the start-up time.

Figure 4-2 Configuration of the UART Scan Deactivation Marker

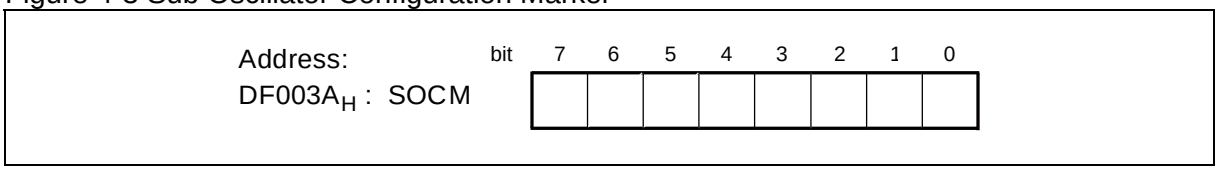


When the content of {USDM1, USDM0} = $292D3A7B_H$, then no temporary UART scanning is performed. For any other value of {USDM1, USDM0}, the Boot ROM will scan dedicated UART channels for limited time after main reset.

Activating the Sub oscillator

The Sub oscillator is shared with general purpose I/O ports. After a Power or External reset, the oscillator is disabled and the Sub Oscillator configuration Marker SOCM is read by the Boot ROM program.

Figure 4-3 Sub Oscillator Configuration Marker



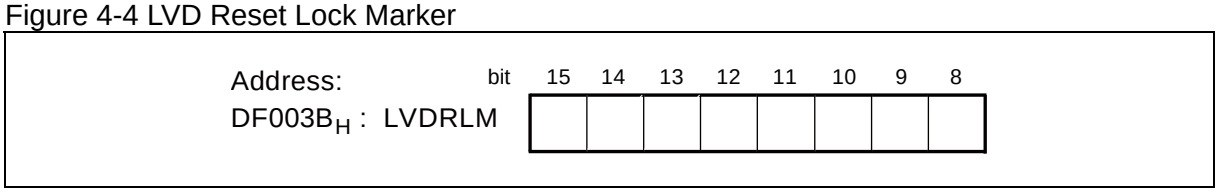
Depending on the content of SOCM, one of the following operation modes is selected:

- SOCM = 55_H: Sub Oscillator will be activated in "oscillation mode". A crystal or resonator must be connected to the X0A/X1A pins.
- SOCM = AA_H: Sub Oscillator will be activated in "external clock input mode". An external clock must be connected to the X0A pin. The X1A pin can be used independently as general purpose I/O port.
- All other values for SOCM: No Sub oscillator function will be activated. The X0A and X1A pins can both be used as general purpose I/O port.

Locking the LVD reset function

The Low Voltage Detector Reset function can be locked in the "active" state by setting the LVD Reset Lock Marker LVDRLM to 55_H. This marker is read after a Power or External reset.

Figure 4-4 LVD Reset Lock Marker

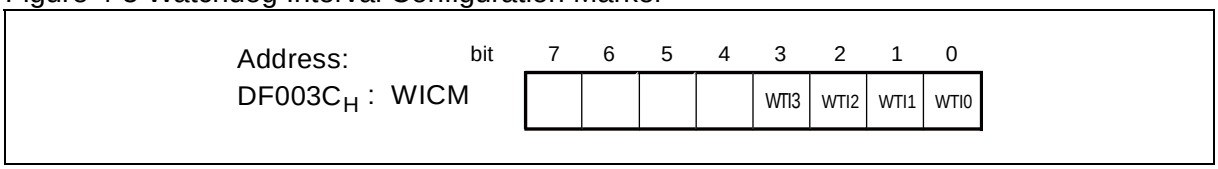


For LVDRLM=55_H, the low voltage detector will be locked in the active state. The setting of the RCR: LVDE bit has no function in this case. Also the low voltage reset function will be active, independent of the setting of the RCR:LVRE bit. Hence the LVD reset function cannot be disabled accidentally by software. Only the LVD threshold level and the LVD hysteresis can be adjusted.
For any setting different from LVDRLM=55_H, the LVD and reset function will be controlled by the RCR:LVDE and RCR:LVRE bits.

Setting the default interval time of the Watchdog Timer

The default interval time of the Watchdog Timer is defined by the WICM marker.

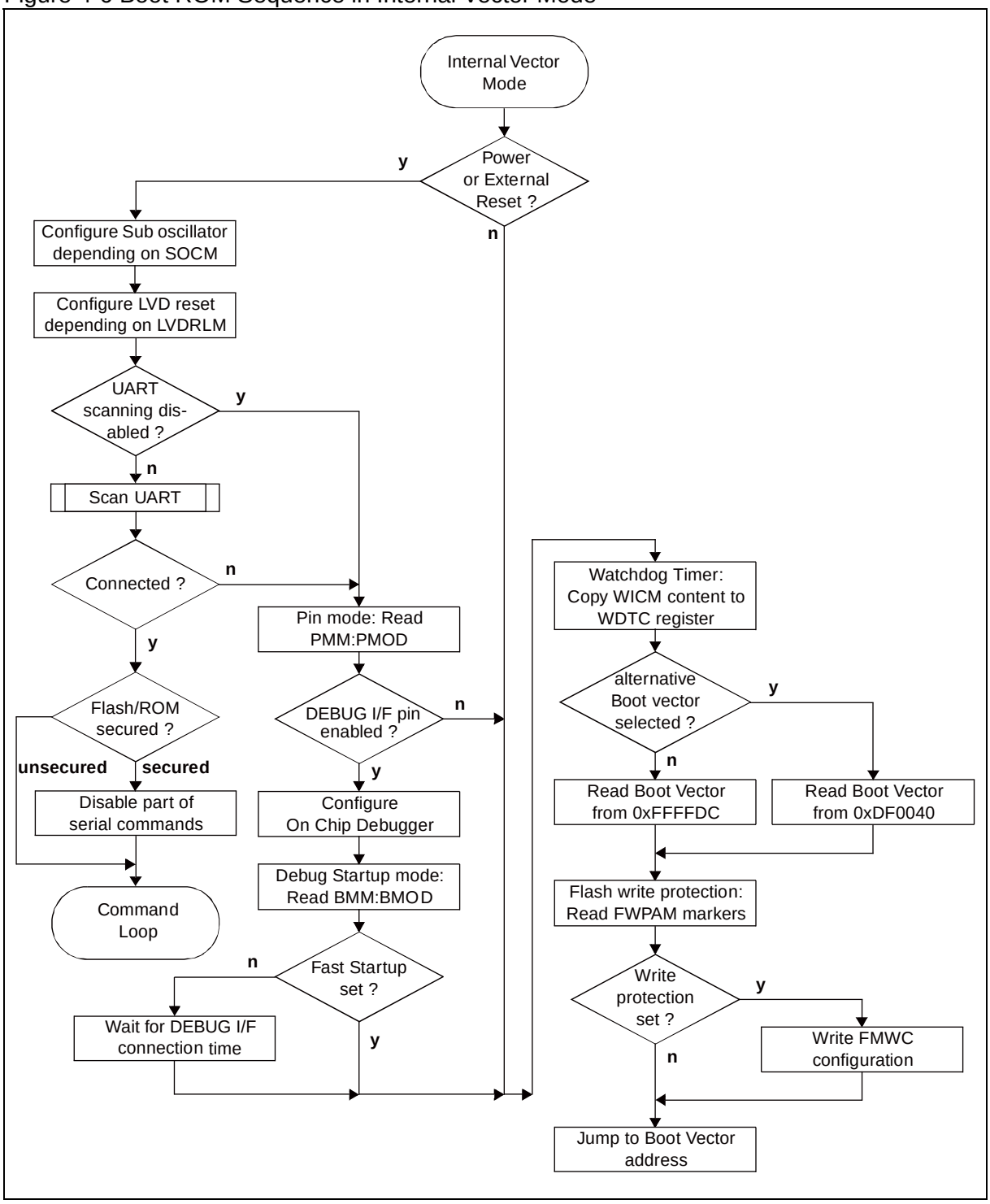
Figure 4-5 Watchdog Interval Configuration Marker



The content of this marker is copied to the WDTC:WTI[3:0] bits of the Watchdog Timer after each reset and defines the default Watchdog Timer interval length. See "CHAPTER 10 WATCHDOG TIMER AND WATCHDOG RESET" for more details.

■ Flowchart Internal Vector Mode

Figure 4-6 Boot ROM Sequence in Internal Vector Mode



5. Operation of the Clock Stop Detection Function and Reset

This section describes the operation of the clock stop detection circuit that detects a failure of the external Main or Sub oscillator.

■ Function of the Clock Stop Detection Circuit

The clock stop detection circuit observes the oscillation of the Main and Sub clock. If no rising edge of the observed clock was detected within a selected interval although the clock was enabled, then the corresponding clock missing flag is set. This function is always active when the RC oscillator is enabled. An additional Clock stop reset can be asserted if the failed clock was currently used for the System clocks CLKS1 or CLKS2 or the Watchdog timer to avoid a hang-up of the MCU. After such a reset the MCU always changes to RC clock mode where the CPU can check the reset cause and put the MCU into a safe state.

■ Configuration of the Clock Stop Detection Circuit

The clock stop detection circuit has 3 configuration bits, one for activating the clock stop reset function and two for selecting the detection interval.

● Clock Stop Detection Reset Enable Bit (RCR: CSDRE)

The initial value of the CSDRE bit after any reset is "0" (Clock stop detection reset disabled). Setting this bit to "1" activates the clock stop detection reset function and writing "0" disables the function. However writing to the CSDRE bit is possible only when the System clock 1 (CLKS1) is set to RC clock.

● Clock Stop Detection Interval Select Bits (RCR: MCSDI and SCSDI)

A missing clock is detected when no rising edge of the observed clock was detected within a certain interval time. This interval is a certain number of RC clock cycles and therefore depends on the selected RC clock frequency. Two bits allow a selection of this interval to adjust the observation time to the selected RC clock frequency and the observed external clock frequency.

Table 5-1 Clock Stop Detection Interval

Observed clock	Select bit	Setting	Clock Stop Detection Interval		
			RC clock cycles	Time for RC clock frequency of 1-4MHz (min - max) / (minimum external frequency)	Time for RC clock frequency of 50-200kHz (min - max) / (minimum external frequency)
Main clock	MCSDI	0 (initial value)	6 - 8	1.5µs - 8µs / (~700kHz)	30µs - 160µs / (~35kHz)
		1	3 - 4	0.75µs - 4µs / (~1.4MHz)	15µs - 80µs / (~70kHz)
Sub clock	SCSDI	0 (initial value)	384 - 512	96µs - 512µs / (~11kHz)	1.92ms - 10.2ms / (~0.55kHz)
		1	24 - 32	6µs - 32µs / (~170kHz)	12µs - 640µs / (~8.5kHz)

The minimum time specifies after which time between two rising edges of the observed clock a stop detection may occur. The maximum time specifies after which time a clock stop will be detected. Use a setting with a minimum time that is longer than the cycle time of the observed clock (the observed clock frequency must be higher than the minimum external frequency given in Table 5-1).
Changing the RC clock frequency during operation of the clock stop detect function is allowed but changes the detection interval.

■ Clock Missing Flags MCMF and SCMF

The Main Clock Missing Flag MCMF indicates a missing Main clock and the Sub Clock Missing Flag SCMF indicates a missing Sub clock.

● Status after Power-on

The clock missing flags MCMF and SCMF are set to "1" after a Power reset.

● Clearing the Clock Missing Flags

The MCMF and SCMF bits can be cleared by a read access to the RCCSRC register at address $00040B_H$.

● Setting the Clock Missing Flags

The Main clock stop detection circuit is enabled when the RC clock is running and the Main oscillator is active (active means the Main Clock Enable bit CKSR: MCE is "1" or the Main Clock Monitor bit CKMR: MCM is "1"). The Main Clock Missing Flag is set to "1" when no rising edge of the Main clock input signal was detected within the selected interval.
The Sub clock stop detection circuit is enabled when the RC clock is running and the Sub oscillator is active (active means the Sub Clock Enable bit CKSR: SCE is "1" or the Sub Clock Monitor bit CKMR: SCM is "1"). The Sub Clock Missing Flag is set to "1" when no rising edge of the Sub clock input signal was detected within the selected interval.

■ Clock Stop Detection Reset

The Clock stop detection reset function generates a reset if a clock failure is detected while this clock is used as source for the System clocks (CLKS1 or CLKS2) or the Watchdog timer.

● Activation of the Clock Stop Detection Reset

The Clock stop detection reset is enabled by setting the RCR: CSDRE bit to "1".

● Generating a Clock Stop Detection Reset

A Clock stop detection reset is generated in the following cases. The Clock Stop Detection Reset Enable bit CSDRE must always be set to "1" (reset enabled).

- When a missing Main clock is detected while the Main clock is stabilized (CKMR:MCM = "1") and the Main or PLL clock is used for the System clocks CLKS1 or CLKS2 or the Watchdog timer. The MCRST (Main clock stop detection Reset cause) bit is set and a reset is generated.
- When a missing Sub clock is detected while the Sub clock is stabilized (CKMR:SCM = "1") and used for the System clocks CLKS1 or CLKS2 or the Watchdog timer. The SCRST (Sub clock stop detection Reset cause) bit is set and a reset is generated.
- When a "0" is written to the CKSR: RCE bit (disable RC clock). Both clock stop detection reset cause bits are set and a reset is generated. Do not disable the RC clock as long as the Clock stop detection reset is enabled.

Note:

A clock stop detection reset is generated only when the failing clock is used for the System clock CLKS1 or CLKS2 or for the Watchdog timer. In case a failing clock is only used by a resource (for example Main clock timer or Sub clock timer) while the System clocks are set to another clock, no reset is generated. Use the Watchdog timer for recovery in case this can lead to hang-up of the system.

● Effect of a Clock Stop Detection Reset

A Clock stop detection reset asserts the global reset signal, sets the corresponding Clock stop detection reset cause bit and puts the MCU into reset state.
The source clock timer and clock ready monitor bit of the failed clock is also cleared by the Clock stop detection reset.
After reset release the MCU restarts in RC clock mode and the reset cause bits can be read. It is recommended to disable the clock that caused the reset by setting the MCE or SCE bit to "0".
The contents of internal RAM memory and all registers which are not reset (initial value "X") cannot be guaranteed after a Clock stop detection reset.

■ Clock Stop Detection Reset and Standby Modes

● Sleep and Timer mode

The clock stop detection reset function is active in Sleep and Timer mode. This means a Main Clock stop reset can be asserted in Run, Sleep and Timer mode if System clock CLKS1 or CLKS2 is set to Main or PLL clock mode (as indicated by the CKMR register). A Sub Clock stop reset can be asserted in Run, Sleep and Timer mode if System clock CLKS1 or CLKS2 is set to Sub clock mode.

● Stop Mode

All oscillators including the RC oscillator are disabled in Stop mode, hence no clock stop detection is possible.
If an interrupt is asserted in Stop mode, then the MCU changes to Run mode with the clocks specified in the CKSR register. The transition to Run mode is delayed until the clock selected with the CKSR: SC1S[1:0] bits is stabilized. The operation of the System Clock 2 is delayed until the clock selected with the CKSR: SC2S[1:0] bits is stabilized.

Note:

The Clock stop detection reset function is disabled during these stabilization times to avoid an accidental assertion of the reset during clock stabilization. This means there will be no Clock stop detection reset if the clock failed during Stop mode or within the stabilization time of the oscillator.
Always set BOTH system clocks to RC clock (CKSR: SC1S[1:0] and SC2S[1:0] = "00") when changing to Stop mode to avoid this situation. Changing CLKS1 or CLKS2 to Main, PLL or Sub clock after start in RC clock mode will not be executed if the corresponding clock did not restart and hence could not set its clock ready monitor bit. The software should detect this situation by a time-out condition using the RC clock timer.

■ Clock Stop Detection Reset and Watchdog Timer

A Clock stop detection reset is also asserted when the Main or Sub clock fails while used as source for the Watchdog timer. During clock stabilization (after Stop mode release) however this reset function is disabled.

6. Operation of the Low Voltage Reset and Interrupt Function

This section describes the operation of the low voltage reset and interrupt function.

■ Function of the Low Voltage Reset

The low voltage reset function is using the low voltage detector that compares the power supply voltage V_{CC} with an internally generated reference voltage. This reference voltage is programmable with the CILCR:LVL[3:0] bits.

If the reset function and the low voltage detector are enabled by the Low Voltage Reset Enable (LVRE) and the Low Voltage Detector Enable (LVDE) bits and V_{CC} falls below the selected detection level V_{DL}, then the reset extension counter is initialized and the PRST flag is set. See the datasheet for the specification of the selectable detection levels V_{DL}.

After recovery of the power supply voltage, the reset extension counter is released and the startup performed as described in "and Section 3. Startup after Power and External Reset".

■ Function of the Low Voltage Interrupt

The low voltage interrupt function is using the same low voltage detector as the low voltage reset function.

If the interrupt function and the low voltage detector are enabled by the Low Voltage Interrupt Enable (LVINTE) and the Low Voltage Detector Enable (LVDE) bits and V_{CC} falls below the selected detection level V_{DL}, then an interrupt is generated.

The low voltage reset function must be disabled, because a reset has a higher priority than an interrupt.

■ Configuration of the Low Voltage Detection and Reset Circuit

● Low Voltage Reset

The low voltage reset is controlled by the Low Voltage Reset Enable (LVRE) bit. Setting this bit to "0" disables the low voltage reset function and setting the bit to "1" enables the function. After Power-on or when asserting RSTX, this bit is always set to "1" (reset function enabled).

The LVRE bit must be set to "0" when the low voltage detector should be disabled. When the low voltage detector is switched on, the low voltage detector stabilization time must be applied before setting LVRE to "1". Otherwise a low voltage reset could be asserted wrongly because the low voltage detector outputs a wrong value. See the datasheet for the specification of the low voltage detector stabilization time.

If the low voltage reset circuit is always enabled (LVD Reset Lock Marker set to 55_H or devices with persistent LVD reset function), then the setting of the LVRE bit has no effect.

● Low Voltage Interrupt

The low voltage interrupt is controlled by the Low Voltage Interrupt Enable (LVINTE) bit. Setting this bit to "0" disables the low voltage interrupt function and setting the bit to "1" enables the function. After Power-on or when asserting RSTX, this bit is always set to "0" (interrupt function disabled).

The LVINTE bit must be set to "0" when the low voltage detector should be disabled. When the low voltage detector is switched on, the low voltage detector stabilization time must be applied before setting LVINTE to "1". Otherwise a low voltage interrupt could be asserted wrongly because the low voltage detector outputs a wrong value. See the datasheet for the specification of the low voltage detector stabilization time.

The Low Voltage Interrupt flag (LVINT) should be cleared by writing "0" before activating the interrupt function.

● Low Voltage Detector

The low voltage detector is controlled by the Low Voltage Detector Enable (LVDE) bit, the CILCR:LVL level select bits and the CILCR:LVHYS hysteresis select bit. Setting the LVDE bit to "0" disables the low voltage detector and setting the bit to "1" enables the detector. After Power-on or when asserting RSTX, this bit is always set to "1" (low voltage detector enabled). The CILCR:LVL level select bits are reset to "0000" (Level 0) and the CILCR:LVHYS hysteresis select bit to "0" by any reset.

When LVDE is set to "0", the LVRE and LVINTE bit must also be set to "0" at the same time or before. When the low voltage detector is switched on, the low voltage detector stabilization time must be applied before setting LVRE or LVINTE to "1".

If the low voltage detector is always enabled (LVD Reset Lock Marker set to 55_H or devices with persistent LVD reset function), then the setting of the LVDE bit has no effect. (low voltage detector is always active).

The detection level is always set to "Level 0" after reset. With this level, a reset can be generated when V_{CC} falls below the minimum value required for safe operation of the MCU. This level can be increased after startup by setting the CILCR:LVL bits. This allows the generation of a Low Voltage reset already at higher V_{CC} levels in case this is required by the system or if the LVD should be used to generate only interrupts.

● Effect on Current Consumption

The low voltage detector draws a current when it is activated (see datasheet for details). This current flows independently of the selected operation mode. If this is not acceptable when using standby modes, disable the low voltage detector and reset before changing to standby mode. Please be aware that the low voltage reset function will not be available in this case.

However if the LVD Reset Lock Marker set to 55_H or in devices with persistent LVD reset function, it is not possible to disable the low voltage detector and reset.

● Restrictions when using the Low Voltage Reset Function

After enabling the low voltage detector, the output value is undefined during the low voltage detector stabilization time. Hence the low voltage reset must not be used within this time.

The low voltage reset function has restrictions regarding the AC specification as described in the datasheet.

7. Registers

This section explains the configuration and functions of the registers used for the Reset Control.

■ Reset Control Registers

The Reset Controller has three registers, the Reset Configuration Register (RCR), the Reset Cause and Clock status Register (RCCSR/RCCSRC) and Clock Input and LVD Control Register (CILCR). The CILCR register controls the low voltage detector which is used for the LVD reset function.

■ List of Reset Control Registers

Abbreviated Register Name	Register Name	Reference
RCR	Reset Configuration Register	See 7.1
RCCSR/RCCSRC	Reset Cause and Clock Status Register	See 7.2
CILCR	Clock Input and LVD Control Register	See 7.3

7.1. Reset Configuration Register (RCR)

The Reset Configuration Register (RCR) is used to assert a Software reset, configure the low voltage reset and detector and to configure the Clock stop detection circuit.

■ Configuration of the Reset Configuration Register (RCR)

RCR								
bit	7	6	5	4	3	2	1	0
	-	-	SCSDI	MCSDI	CSDRE	LVDE	LVRE	SRSTG
Attribute	-	-	R/W	R/W	R/W	R/W	R/W	W
Initial value	X	X	0	0	0	1	1	0

[bit7, bit6] -: Undefined

- Always write "0" to these bits.
- The read value of these bits is undefined.
- Read modify write operations to this register have no effect on these bits..

[bit5] SCSDI: Sub Clock Stop Detection Interval Select Bit

Bit	Description
0	Sub clock stop detection interval time is 384-512 RC clock cycles
1	Sub clock stop detection interval time is 24-32 RC clock cycles

- This bit controls the measurement interval of the Sub clock stop detection circuit.
- This bit is initialized to "0" by any reset.
- The Sub clock stop detection circuit sets the Sub clock missing flag (RCCSR: SCMF) if no rising edge of the Sub clock input signal (CLKSC) was observed within the selected interval time.

[bit4] MCSDI: Main Clock Stop Detection Interval Select Bit

Bit	Description
0	Main clock stop detection interval time is 6-8 RC clock cycles
1	Main clock stop detection interval time is 3-4 RC clock cycles

- This bit controls the measurement interval of the Main clock stop detection circuit.
- This bit is initialized to "0" by any reset.
- The Main clock stop detection circuit sets the Main clock missing flag (RCCSR: MCMF) if no rising edge of the Main clock input signal (CLKMC) was observed within the selected interval time.

[bit3] CSDRE: Clock Stop Detection Reset Enable Bit

Bit	Description
0	Clock stop detection reset function disabled
1	Clock stop detection reset function enabled

- This bit enables the Clock stop detection reset function.
- This bit can only be written when the System clock CLKS1 is set to RC clock.
- This bit is initialized to "0" (Clock stop detection reset disabled) by any reset.
- After activation, a Clock stop detection reset is asserted in the following three cases:
  1. When a missing Main clock is detected while the Main or PLL clock is selected for the System clocks CLKS1 or CLKS2 or the Watchdog timer.
  2. When a missing Sub clock is detected while the Sub clock is selected for the System clocks CLKS1 or CLKS2 or the Watchdog timer.
  3. When "0" is written to the CKSR: RCE bit to disable the RC oscillator.

[bit2] LVDE: Low Voltage Detector Enable Bit

Bit	Description
0	Low Voltage Detector disabled
1	Low Voltage Detector enabled

- This bit controls the low voltage detector which is used for the Low voltage reset function.
- This bit is initialized to "1" (low voltage detector active) by an External and a Power (power-on) reset only.
- Do not set this bit to "0" as long as LVRE is set to "1".
- The low voltage detector is enabled by setting this bit to "1". However this circuit needs a stabilization time after activation as specified in the datasheet. Do not activate the Low voltage reset function before this time has elapsed.
- If the LVD reset function is locked in the "active" state (LVD Reset Lock Marker set to 55_H or devices with persistent LVD reset function), then the setting of this bit has no effect.

[bit1] LVRE: Low Voltage Reset Enable Bit

Bit	Description
0	Low Voltage reset function disabled
1	Low Voltage reset function enabled

- This bit controls the Low voltage reset function which is one reset cause of the Power reset.
- This bit is initialized to "1" (Low voltage reset active) by an External and a Power (power-on) reset only.
- Do not disable the low voltage detector (by writing "0" to the LVDE bit) as long as this bit is set to "1".
- Do not set this bit from "0" to "1" before the low voltage detector is enabled and stabilized.
- If the LVD reset function is locked in the "active" state (LVD Reset Lock Marker set to 55_H or devices with persistent LVD reset function), then the setting of this bit has no effect.
- This bit must be set to "0" (LVD reset function disabled) when the LVD interrupt function should be used.

[bit0] SRSTG: Software Reset Generation Bit

Bit	Description
0	No effect on operation
1	Internal software reset is generated

- The read value of this bit is always "0".

7.2. Reset Cause and Clock Status Register (RCCSR/RCCSRC)

The RCCSR/RCCSRC register shows the reset cause and the status of the Main and Sub clock

■ Configuration of the Reset Cause and Clock Status Register (RCCSR/RCCSRC)

The Reset Cause and Clock Status Register (RCCSR/RCCSRC) can be accessed at two addresses. A read access to address $00040B_H$ (RCCSRC) clears all bits of the register after reading while a read access to address $00040D_H$ (RCCSR) doesn't change the status of the register. Writing to the RCCSR/RCCSRC register is ignored.

RCCSRC

bit	15	14	13	12	11	10	9	8
	SCMF	MCMF	WRST	SRST	SCRST	MCRST	ERST	PRST
Attribute	R/C	R/C	R/C	R/C	R/C	R/C	R/C	R/C
Initial value	X	X	X	X	X	X	X	X

R/C: Read and Clear

RCCSR

bit	15	14	13	12	11	10	9	8
	SCMF	MCMF	WRST	SRST	SCRST	MCRST	ERST	PRST
Attribute	R	R	R	R	R	R	R	R
Initial value	X	X	X	X	X	X	X	X

[bit15] SCMF: Sub Clock Missing Flag

Bit	Description
0	No missing Sub clock was detected
1	Missing Sub clock was detected

- This bit indicates if a missing Sub clock was detected.
- After a Power reset, this bit is also set to "1".
- This bit is set to "1" if the Sub oscillator is enabled and no rising edge of the Sub clock input signal (CLKSC) was observed within the interval time defined by the RCR: SCSDI bit.
- This bit is cleared by a read access to RCCSRC at address $00040B_H$.

[bit14] MCMF: Main Clock Missing Flag

Bit	Description
0	No missing Main clock was detected
1	Missing Main clock was detected

- This bit indicates if a missing Main clock was detected.
- After a Power reset, this bit is also set to "1".
- This bit is set to "1" if the Main oscillator is enabled and no rising edge of the Main clock input signal (CLKMC) was observed within the interval time defined by the RCR: MCSDI bit.
- This bit is cleared by a read access to RCCSRC at address $00040B_H$.

[bit13] WRST: Watchdog Timer Reset Cause Bit

Bit	Description
0	No Watchdog timer reset was generated
1	Watchdog timer reset was generated

- This bit indicates if a Watchdog timer reset was generated.
- After a Power reset, this bit is also set to "1".
- This bit is cleared by a read access to RCCSRC at address $00040B_H$.
- This bit is set to "1" if a Watchdog timer reset was generated. Possible reset causes are:
  1. The Watchdog timer was activated but not cleared within the selected interval.
  2. An illegal value was written to the Watchdog Timer Clear Pattern register.
  3. "0" was written to the enable bit of the clock which is used as source clock of the Watchdog timer (CKSR: RCE, MCE or SCE) to disable this clock.
  4. RC clock frequency was changed although Watchdog Timer Clock Selection bits were set to "01" (changing RC clock frequency not allowed).
  5. Transition to Stop mode was requested although the WDTC:RSTP bit was set to "1".

[bit12] SRST: Software Reset Cause Bit

Bit	Description
0	No Software reset was generated
1	Software reset was generated

- This bit indicates if a Software reset was generated.
- After a Power reset, this bit is also set to "1".
- This bit is cleared by a read access to RCCSRC at address $00040B_H$.
- This bit is set to "1" if a Software reset was generated by writing "1" to the RCR: SRSTG bit.

[bit11] SCRST: Sub Clock Stop Detection Reset Cause Bit

Bit	Description
0	No Sub Clock stop detection reset was generated
1	Sub Clock stop detection reset was generated

- This bit indicates if a Sub Clock stop detection reset was generated.
- After a Power reset, this bit is also set to "1".
- This bit is cleared by a read access to RCCSRC at address $00040B_H$.
- This bit is set to "1" if a Sub Clock stop detection reset was generated. Possible reset causes are:
  1. A missing Sub clock was detected while the Sub clock was selected for the System clock CLKS1 or CLKS2 or the Watchdog timer.
  2. "0" was written to the enable bit of the RC clock (CKSR: RCE) although the Clock stop detection reset was enabled.

[bit10] MCRST: Main Clock Stop Detection Reset Cause Bit

Bit	Description
0	No Main Clock stop detection reset was generated
1	Main Clock stop detection reset was generated

- This bit indicates if a Main Clock stop detection reset was generated.
- After a Power reset, this bit is also set to "1".
- This bit is cleared by a read access to RCCSRC at address $00040B_H$.
- This bit is set to "1" if a Main Clock stop detection reset was generated. Possible reset causes are:
  1. A missing Main clock was detected while the Main or the PLL clock was selected for the System clock CLKS1 or CLKS2 or the Watchdog timer.
  2. "0" was written to the enable bit of the RC clock (CKSR: RCE) although the Clock stop detection reset was enabled.

[bit9] ERST: External Reset Cause Bit

Bit	Description
0	No External reset was generated
1	External reset was generated

- This bit indicates if an External reset was generated.
- This bit is set to "1" if an External reset was generated by asserting RSTX to "0".
- After a Power reset, this bit is also set to "1".
- This bit is cleared by a read access to RCCSRC at address $00040B_H$.

[bit8] PRST: Power Reset Cause Bit

Bit	Description
0	No Power reset was generated
1	Power reset was generated

- This bit indicates if a Power reset was generated.
- After Power-on, this bit is set to "1".
- This bit is cleared by a read access to RCCSRC at address $00040B_H$.
- This bit is set to "1" if a Power reset was generated. Possible reset causes are:
  1. Power-on event: The power was turned on according to the profile described in the datasheet.
  2. Low voltage event: The low voltage reset function was enabled and the power supply dropped below a value described in the datasheet.

■ Notes about Reset Cause Bits

● Multiple Reset Causes before reading Reset Cause Register

When multiple reset causes are generated before the reset cause register is read out, the corresponding reset cause bits of the RCCSR/RCCSRC register are all set to "1". If, for example, an External reset request via the RSTX pin and the Watchdog timer reset occur at the same time, the ERST and the WRST bits are both set to "1". The following table shows this correspondence.

Table 7-1 Correspondence between Reset Cause Bits and Reset Causes

Reset cause	PRST	ERST	MCRST	SCRST	SRST	WRST
Power reset (Power-on or low voltage)	1	1	1	1	1	1
External reset request via RSTX pin	*	1	*	*	*	*
Main clock stop detection reset	*	*	1	*	*	*
Sub clock stop detection reset	*	*	*	1	*	*
Software reset request	*	*	*	*	1	*
Watchdog timer overflow	*	*	*	*	*	1

*: Previous state maintained

● Power Reset

For PRST = "1", the software should be programmed so that it will ignore all other reset cause bits.

● Clearing the Reset Cause Bits

The reset cause bits are cleared only when reading the Reset Cause and Clock Status Register RCCSRC at address $00040B_H$. Any bit corresponding to a reset cause that has already been generated is not cleared when another reset is generated (a setting of "1" is retained).
After a Power reset, the register should be cleared by reading in order to initialize all bits.

Note:

If the power is turned on under conditions where power-on reset may not occur (power-on profile as specified in the datasheet not met), the value in RCCSR register is not guaranteed.

■ Notes about Clock Missing Flags

For details about the clock missing flags MCMF and SCMF see "and Section 5. Operation of the Clock Stop Detection Function and Reset".

7.3. Clock Input and LVD Control Register (CILCR)

The Clock Input and LVD Control Register (CILCR) is used to control additional functions of the oscillator circuit and the Low Voltage Detector.

■ Configuration of the Clock Input and LVD Control Register (CILCR)

The register can be accessed 16-bit wide (VRCR) and 8-bit wide (low byte: VRCR, high byte CILCR).

CILCR								
bit	15	14	13	12	11	10	9	8
	LVINTE	LVINT	LVHYS	FCI	LVL3	LVL2	LVL1	LVL0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

[bit15] LVINTE: Low Voltage Interrupt Enable

Bit	Description
0	Low voltage interrupt disabled
1	Low voltage interrupt enabled

- This bit is used to enable the low voltage interrupt function.
- A low voltage interrupt is generated when the LVINTE enable bit and the LVINT flag are both "1".
- This bit is reset to "0" by any reset.
- The RCR:LVDE bit (LVD reset enable) must be set to "0" when the low voltage interrupt is used, because the reset function has a higher priority.

[bit14] LVINT: Low Voltage Interrupt Flag

Bit	Description	
	Read	Write
0	No interrupt	Clear this bit
1	Interrupt requested	No effect

- This is the request flag for the low voltage interrupt.
- This bit is set to "1" when the Vcc power supply falls below the value specified by the LVL[3:0] bits.
- When this bit is set to "1", an interrupt request is issued if the interrupt enable bit LVINTE is set to "1".
- This bit is cleared by writing "0" and by any reset.
- This bit should always be cleared when activating the interrupt function (setting LVINTE to "1").
- "1" is always read by a read-modify-write instruction.

[bit13] LVHYS: Low Voltage Detector Hysteresis Select

Bit	Description
0	Small hysteresis selected
1	Big hysteresis selected

- This bit selects the hysteresis of the comparator of the low voltage detector.
- This bit is initialized to "0" by each reset. This selects the "small" hysteresis.
- Setting this bit to "1" selects the "big" hysteresis.
- The hysteresis voltage value for both settings is described in the datasheet.

[bit12] FCI: Fast External Clock Input Enable

- This bit is initialized to "0" by each reset.
- This bit selects the oscillation or Fast external Clock Input mode of the Main Oscillator.

Bit	Description
0	Oscillation mode
1	Fast external clock input mode

- This bit must be set to "0" when connecting an oscillator (crystal/resonator) to the Main Oscillator Pin X0 and X1. It is also possible to connect an external clock with low frequency to the oscillator pins in this mode (X0/X1 or only X0).
- For inputting an external clock with higher frequency at the X0 pin, this bit must be set to "1" before the device is switched to external (main) clock.
- The frequency ranges for both settings are described in the datasheet.

[bit11 to bit8] LVL3 to LVL0: Low Voltage Detector Level Select Bits

· These bits control the analog threshold level for the Low Voltage Detector (LVD):

bit11	bit10	bit9	bit8	Description
0	0	0	0	Level 0 *
0	0	0	1	Level 1 *
0	0	1	0	Level 2 *
0	0	1	1	Level 3 *
0	1	0	0	Level 4 *
0	1	0	1	Level 5 *
0	1	1	0	Level 6 *
0	1	1	1	Level 7 *
1	0	0	0	Level 8 *
1	0	0	1	Level 9 *
1	0	1	0	Level 10 *
1	0	1	1	Level 11 *
1	1	0	0	Level 12 *
1	1	0	1	Level 13 *
1	1	1	0	Level 14 *
1	1	1	1	Level 15 *

· These bits are initialized to "0000" by each reset.

· The Low Voltage Detector circuit will issue a reset or interrupt request when the corresponding function is enabled and the external supply voltage drops below the selected threshold value.

* Note: For the allowed settings and the analog threshold level, please refer to the datasheet.
For more details see "CHAPTER 7 RESETS AND STARTUP"

CHAPTER: STANDBY MODE AND VOLTAGE REGULATOR CONTROL CIRCUIT

This chapter explains the functions and operations of the standby mode control circuit and the control of the internal voltage regulator.

Management Code: 96F6MODE-E01.0

1. Overview

The F^{2}MC-16FX MCU has the following CPU operating modes:

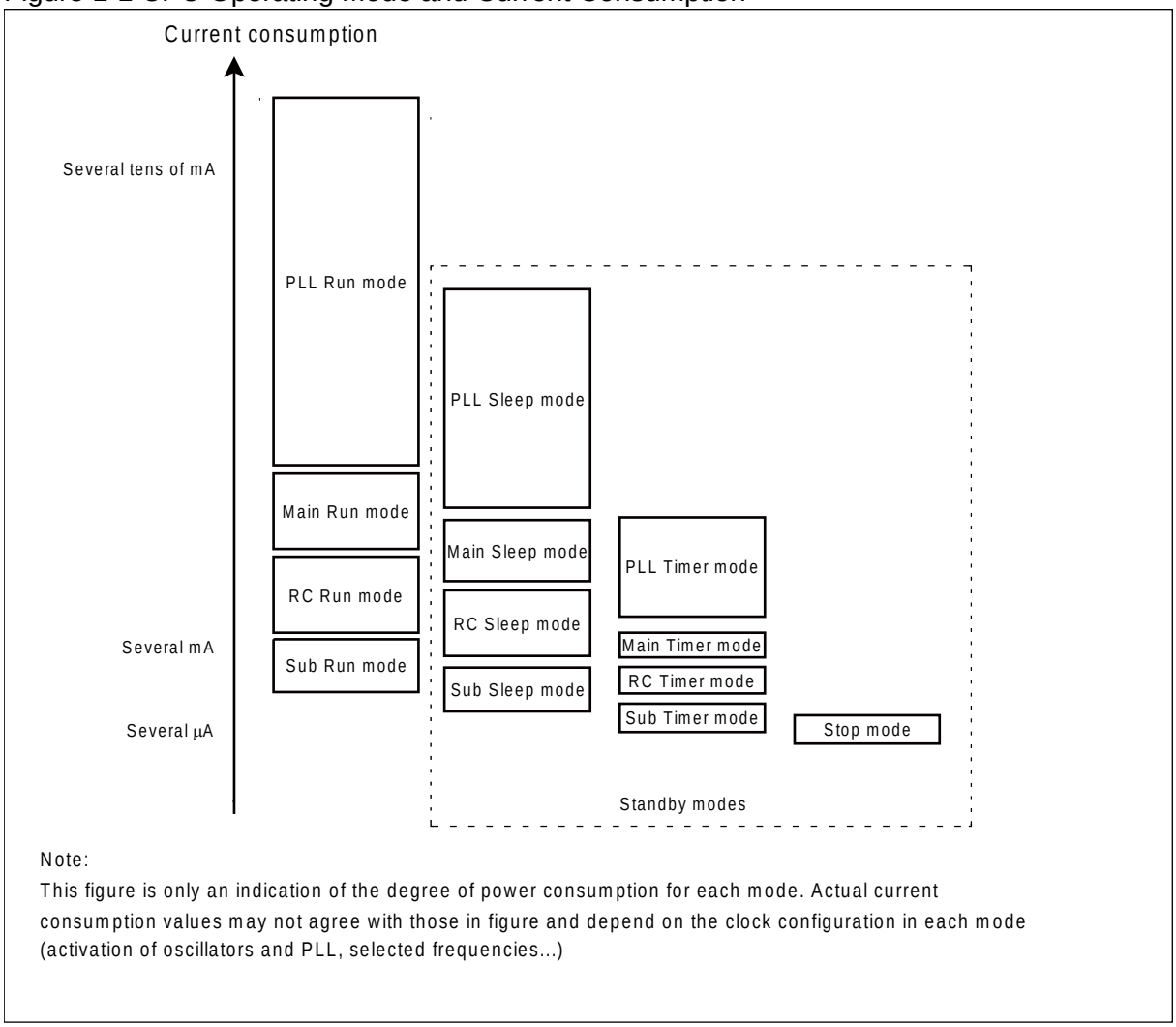
- Run mode
- Sleep mode
- Timer mode
- Stop mode

Sleep mode, Timer mode and Stop mode are called Standby modes. In Stop mode, all clocks are disabled. In the other CPU operating modes, the clock source and active oscillators/PLL are selectable.

■ CPU Operating Modes and Current Consumption

Figure 1-1 shows the relationship between the CPU operating modes and current consumption.
The power consumption of the operating modes can be influenced by the settings of the internal voltage regulator.

Figure 1-1 CPU Operating Mode and Current Consumption

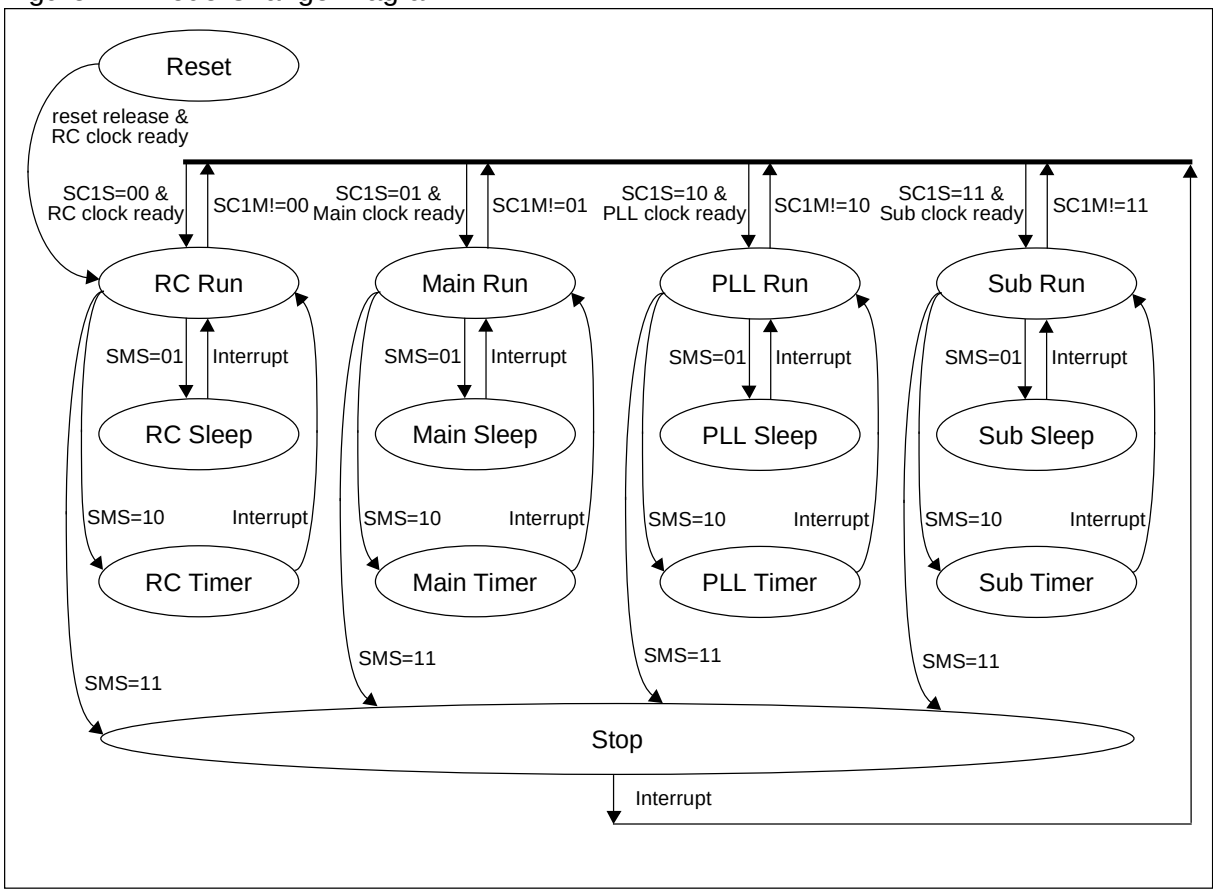


Note:
This figure is only an indication of the degree of power consumption for each mode. Actual current consumption values may not agree with those in figure and depend on the clock configuration in each mode (activation of oscillators and PLL, selected frequencies...)

■ Transitions between Different Operating Modes

The mode change diagram in
Figure 1-2 shows possible transitions between the different operating modes.

Figure 1-2 Mode Change Diagram



■ Run Mode

The Run mode is always defined by the clock selected as source for the System clock 1 (CLKS1). The System clock 2 (CLKS2) which is selected by the System clock 2 selector is independent of this Run mode. Hence modules connected to the Peripheral clock 2 (CLKP2) can operate with a different clock source because the Peripheral clock 2 is derived from the System clock 2.

● RC Run Mode

In this mode, the internal RC oscillation clock (CLKRC) is used for the System clock 1, operating the CPU and most peripheral functions. Two frequency settings are possible for the RC oscillator (nominal 2MHz or 100kHz). Independent clock dividers for the Bus clock operating the CPU (CLKB) and the Peripheral clock 1 operating most peripheral functions (CLKP1) are fed by the RC clock and allow flexible frequency settings.

The status of the main oscillator, the PLL multiplier circuit and the sub oscillator depend on the setting of the MCE, PCE and SCE bits when they are not used as System clock 2 (otherwise they are started automatically).

This mode is always active after reset release.

● Main Run Mode

In this mode, the Main oscillation clock (CLKMC) is used for the System clock 1, operating the CPU and most peripheral functions. Independent clock dividers for the Bus clock operating the CPU (CLKB) and the Peripheral clock 1 operating most peripheral functions (CLKP1) are fed by the Main clock and allow flexible frequency settings.
The status of the PLL multiplier circuit, the RC and sub oscillator depend on the setting of the PCE, RCE and SCE bits when they are not used as System clock 2 (otherwise they are started automatically).

● PLL Run Mode

In this mode, the PLL clock that is a multiple of the Main oscillation clock is used for the System clock 1, operating the CPU and most peripheral functions. Independent clock dividers for the Bus clock operating the CPU (CLKB) and the Peripheral clock 1 operating most peripheral functions (CLKP1) are fed by the PLL output clock and allow flexible frequency settings.
The status of the RC and sub oscillator depend on the setting of the RCE and SCE bits when they are not used as System clock 2 (otherwise they are started automatically).

● Sub Run Mode

In this mode, the Sub oscillation clock (CLKSC) is used for the System clock 1, operating the CPU and most peripheral functions. Independent clock dividers for the Bus clock operating the CPU (CLKB) and the Peripheral clock 1 operating most peripheral functions (CLKP1) are fed by the Sub clock and allow flexible frequency settings.
The status of the main oscillator, the PLL multiplier circuit and the RC oscillator depend on the setting of the MCE, PCE and RCE bits when they are not used as System clock 2 (otherwise they are started automatically).

Reference:

For more details, see "CHAPTER 6 CLOCKS".

■ Sleep Mode

In this mode, the standby control circuit stops the Bus clock (CLKB) which disables the CPU, internal memories and the DMA controller, thereby reducing power consumption.

● RC Sleep Mode

The RC Sleep mode is activated to stop the Bus clock (CLKB) in the RC clock mode. Resources connected to the Peripheral clock 1 operate on the RC clock.
Resources connected to the Peripheral clock 2 are operating with the clock selected by the System clock 2 selector.
The status of the main oscillator, the PLL multiplier circuit and the sub oscillator depend on the setting of the MCE, PCE and SCE bits when they are not used as System clock 2 (otherwise they are started automatically).

● Main Sleep Mode

The Main Sleep mode is activated to stop the Bus clock (CLKB) in the Main clock mode. Resources connected to the Peripheral clock 1 operate on the Main clock.
Resources connected to the Peripheral clock 2 are operating with the clock selected by the System clock 2 selector.
The status of the PLL multiplier circuit, the RC and Sub oscillator depend on the setting of the PCE, RCE and SCE bits when they are not used as System clock 2 (otherwise they are started automatically).

● PLL Sleep Mode

The PLL Sleep mode is activated to stop the Bus clock (CLKB) in the PLL clock mode. Resources connected to the Peripheral clock 1 operate on the PLL clock.
Resources connected to the Peripheral clock 2 are operating with the clock selected by the System clock 2 selector.
The status of the RC and Sub oscillator depend on the setting of the RCE and SCE bits when they are not used as System clock 2 (otherwise they are started automatically).

● Sub Sleep Mode

The Sub Sleep mode is activated to stop the Bus clock (CLKB) in the Sub clock mode. Resources connected to the Peripheral clock 1 operate on the Sub clock.
Resources connected to the Peripheral clock 2 are operating with the clock selected by the System clock 2 selector.
The status of the Main oscillator, the PLL multiplier circuit and the RC oscillator depend on the setting of the MCE, PCE and RCE bits when they are not used as System clock 2 (otherwise they are started automatically).

■ Timer Mode

In this mode, the standby control circuit stops supplying the System clocks (CLKS1 and CLKS2) which further reduces power consumption by stopping the Bus clock and also all Peripheral clocks. This stops all functions, excluding oscillators, the PLL and the corresponding source clock timers.

● RC Timer Mode

In RC Timer mode, the RC oscillator and RC clock timer is always active. The status of the PLL, the Main and Sub oscillator and corresponding source clock timers however depend on the setting of the PCE, MCE and SCE bits when they are not used as System clock 2 (otherwise they are started automatically).

● Main Timer Mode

In Main Timer mode, the Main oscillator and Main clock timer is always active. The status of the PLL, the RC and Sub oscillator and corresponding source clock timers however depend on the setting of the PCE, RCE and SCE bits when they are not used as System clock 2 (otherwise they are started automatically).

● PLL Timer Mode

In PLL Timer mode, the Main oscillator, the PLL and the Main clock timer is always active. The status of the RC and Sub oscillator and corresponding source clock timers however depend on the setting of the RCE and SCE bits when they are not used as System clock 2 (otherwise they are started automatically).

● Sub Timer Mode

In Sub Timer mode, the Sub oscillator and Sub clock timer is always active. The status of the PLL, the Main and RC oscillator and corresponding source clock timers however depend on the setting of the PCE, MCE and RCE bits when they are not used as System clock 2 (otherwise they are started automatically).

Note:

The different Timer modes may have the same behavior if more than one of the PCE, MCE, SCE and RCE bits is set to "1". However, the run mode that the device transits to, upon wakeup from timer mode by interrupt, depends on which of the four timer modes the device was in. See Table 6-1.

■ Stop Mode

In this mode, the standby control circuit causes all oscillations to stop. All functions are inactivated.

Note:

Because the Stop mode turns all oscillation clocks off, data can be retained by the lowest power consumption.

---

2. Usage Notes on Standby Mode

Note the following items when using the standby modes:
- Switching to a standby mode and Interrupt
- Release of the standby mode by an Interrupt
- Release of the Stop mode by an Interrupt
- Oscillation stabilization wait time after Stop mode release
- Clock mode switching

● Switching to a Standby Mode and Interrupt

When there is a pending interrupt request to the CPU, writing to the SMS bits in the SMCR register has no effect. In other words, the desired mode transition is not performed even after the interrupt service is completed. If the interrupt request is already cleared and if there are no other pending requests, the desired mode transition can be performed. Note that this behavior is not dependent on whether the setting in the PS register allows the CPU to accept the interrupt request.

● Release of the Standby Mode by an Interrupt

If an NMI or an interrupt request of interrupt priority level higher than seven is issued from a peripheral function during Sleep, Timer or Stop mode, the standby mode is released, which does not depend on whether the CPU accepts the interrupt.
After the release of the standby mode by an interrupt, normal processing is performed.
If the interrupt is accepted, the CPU first executes the remaining instructions in the pipeline as to the state before entering the standby mode. Then it continues with normal interrupt processing.
If the interrupt is not accepted, the CPU continues with the execution of the instructions in the pipeline and the following instructions.

● Release of the Stop Mode by an Interrupt

The Stop mode can be released by an external interrupt or by an NMI. As an external interrupt input cause, a H-level signal, L-level signal, rising edge, or falling edge can be selected.

● Oscillation Stabilization Wait Time after Stop Mode Release

Because all oscillators are halted in the Stop mode, an oscillation stabilization wait time is required. The setting of the SC1S and SC2S bits of the CKSR register define which clock is used after wakeup from Stop mode by interrupt. Oscillators and the PLL clock are enabled depending on the RCE, MCE, PCE, SCE, SC1S and SC2S bits and stabilization times are applied according to the setting of the clock stabilization select register (CKSSR). The CPU starts operating after stabilization of the System clock 1, selected by the SC1S bit. Modules using the System clock 2 start operating after stabilization of the System clock 2. The clocks selected by the SC1S and SC2S bits are always enabled, independent of the setting of the clock enable bits (RCE, MCE, PCE and SCE).

Note:

After Stop mode release, the CPU starts operating after stabilization of the System clock 1 selected by the SC1S bit. However another clock can be selected as System clock 2 with a longer stabilization time. If the CPU tries to access a peripheral resource clocked by the Peripheral clock 2, then the CPU will be put into hold state until CLKP2 is stabilized and the access can be completed.

After Stop mode release, the SC2M bits indicate the clock used before stop mode until the newly selected clock is ready (clock ready monitor bit of clock selected by SC2S bits is "1") because this register can only be updated with System clock 2.

Note:

Never set the SC1S or SC2S bits to a clock which does not exist (for example "Sub clock mode" when the sub oscillator function is not activated or no crystal/resonator is connected to sub oscillator pins) because program execution is delayed until the clock selected by SC1S is stabilized. It is recommended to always select the RC clock for both System clocks when changing to Stop mode.

If Stop mode is released by reset, the clock mode is always set to RC clock and the reset sequence is executed after stabilization of the RC oscillator. All registers and functions are reset to the initial state.

● Clock Mode Switching

When the clock mode is switched by writing to the SC1S/SC2S bits, do not switch to Sleep or Timer mode before the clock mode switching is completed. Confirm the completion of clock mode switching by referring to the SC1M/SC2M bits of the clock monitor register (CKMR).
A transition to Stop mode however is possible without waiting for the clock mode switching.
Only change the setting of the SC1S/SC2S bits when the SMS bits are "00" (no Standby mode transition request pending).

3. Using the Extended Standby Mode Control Register

The Extended Standby Mode Control Register has two functions:
- Control of the core bus start time after standby mode release
- Control of the Flash power-down function in standby mode

■ Core Bus Start Delay

After wakeup from Stop mode, the CPU (Bus clock CLKB) is started after stabilization of the source clock used for CLKS1. After wakeup from Sleep or Timer mode, CLKB is started immediately.
The first access on the core bus however is delayed by the programmable bus start delay time. This time can be selected by the ESMCR: BSD[3:0] bits.
This delay between activating the clocks and activating the core bus master eases the increase of the MCU current consumption at wakeup from standby mode.
Set a value to the ESMCR: BSD[3:0] bits depending on the CLKB frequency after wakeup from standby mode. The resulting delay time should be set to minimum 10µs.
If such a delay is not acceptable after wakeup from standby mode, then a shorter delay value can be selected. However this requires a bigger capacitor at the C-Pin to compensate the faster current increase.
See the datasheet for required C-Pin capacitors depending on the selected start delay and used clock frequency.
At wakeup from Sub Timer, Sub Sleep, RC (100kHz) Timer and RC (100kHz) Sleep mode, a fixed bus start delay time of 16 clock (CLKB) cycles is applied, independent of the ESMCR: BSD[3:0] register setting. This time is required for the voltage regulator stabilization after changing from the Low Power mode to the High Power mode.
When using the Low Power Mode A of the voltage regulator in Main Timer/Sleep or RC (2MHz) Timer/Sleep mode (by setting SMCR:LPMSS to "1"), then a Bus start delay time of minimum 100µs must be selected.

● Operation of Peripherals after Wakeup from Standby Mode

After wakeup from Stop mode, the peripherals start operating as soon as the source clocks selected for the system clocks are stabilized (peripherals connected to CLKP1 start operating when CLKS1 is stabilized and peripherals connected to CLKP2 start operating when CLKS2 is stabilized). The CPU start however is delayed by the bus start delay time. Hence there is a time where the peripherals operate already while the core bus is not yet active.
In Timer mode, the situation is similar. The peripherals resume operation immediately after the wakeup interrupt event while the CPU start is delayed.
In Sleep mode, the peripherals are always active. Hence the bus start delay is only visible as a wakeup delay for the CPU.

■ Flash Power-down Function

The Flash macro has a power-down/reset function which can be used in Standby mode for current saving. This function can be activated by setting the ESMCR:FPDS bit to "1".
For the effect on the current consumption, see the datasheet.

● Behavior for ESMCR:FPDS="1"

The Flash power-down/reset function can only be activated when no Flash program/erase operation is ongoing and the Flash Write command sequencer is "Idle". Hence it is recommended to check that DFSA:RDY (and DFSB:RDY if present) are both "1" and that DFSA:ST[1:0] (and DFSB:ST[1:0] if present) are both "00" before activating a Standby mode (Sleep, Timer or Stop mode) with ESMCR:FPDS="1".
For DFSA:RDY="0" or DFSB:RDY="0" or DFSA:ST[1:0] !="00" or DFSB:ST[1:0] !="00", setting FPDS to "1" has no effect (same behavior as for FPDS = "0").
When no Flash program/erase operation is ongoing and the Write Command sequencer is "Idle", then the Flash power-down/reset mode is activated at transition to any Standby mode. In this mode, the current consumption of the Flash is reduced and the State machine in the Flash is set to the "Read/Reset" state (even when the Flash was in the "Sector erase suspended" state before Standby mode entry).
After wakeup from Standby mode, the power-down function is left and a Flash stabilization time of max 150µs is applied. At wakeup from Stop mode, this Flash stabilization time is applied after the CLKS1 stabilization (after the clock selected by CKSR:SC1S is stabilized).
DFSA/B:RDY is "0" during this stabilization time and set to "1" after stabilization of the Flash. A read access to the Flash within this stabilization time (DFSA/B:RDY="0") stops the core bus until the Flash is stabilized.
A write access to the Flash macro is ignored while DFSA/B:RDY bit is set to "0" and the write busy interrupt flag is set.

Figure 3-1 Timing after wakeup from Stop mode with ESMCR:FPDS=1

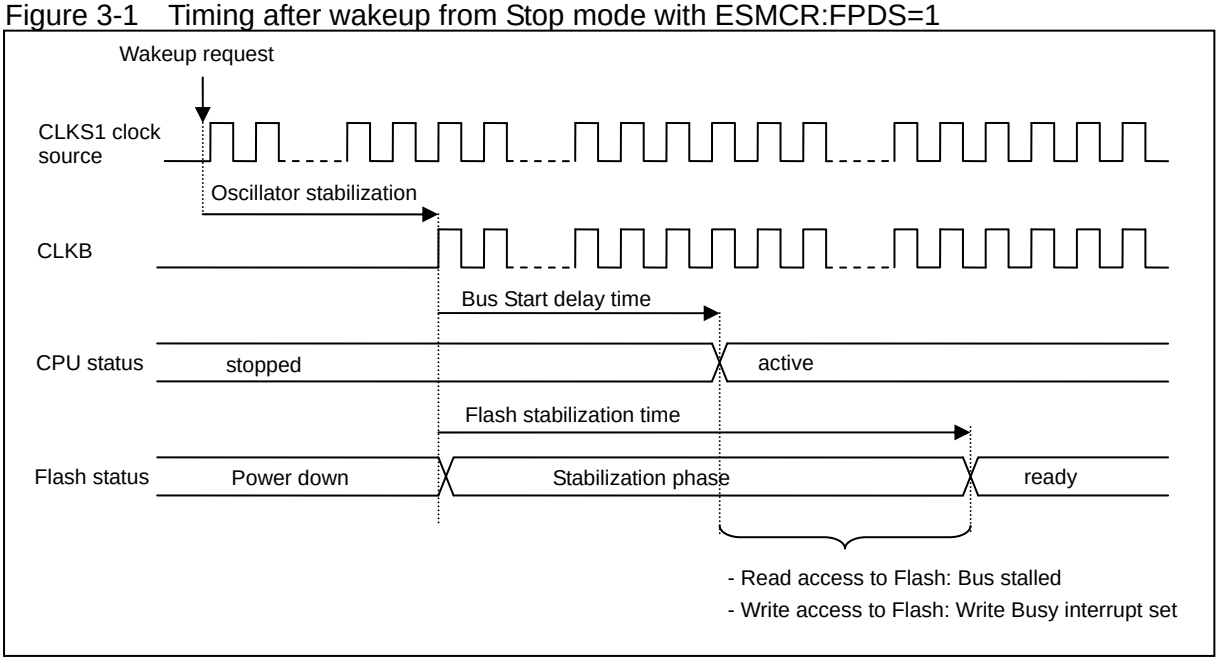
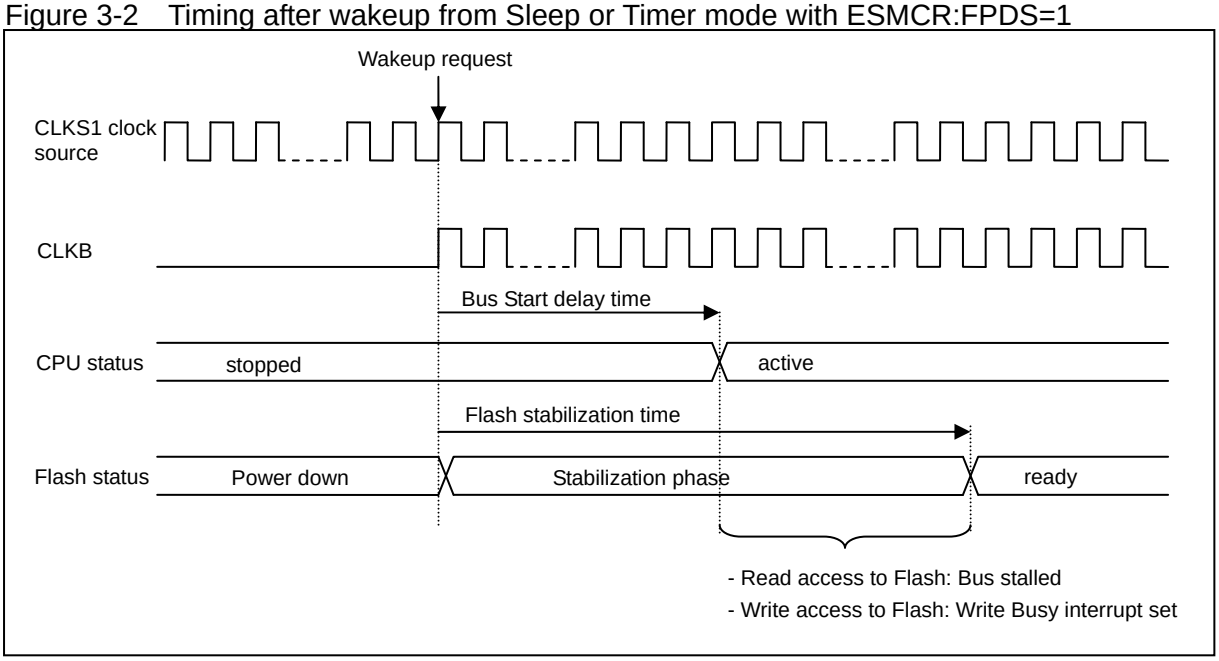


Figure 3-2 Timing after wakeup from Sleep or Timer mode with ESMCR:FPDS=1



● Behavior for ESMCR:FPDS="0"

Setting the ESMCR:FPDS bit to "0" does not activate the power- down/reset function of the Flash macro after entering a Standby mode. This leads to a higher current consumption in Standby mode.
The state of the Flash is not affected by the transition to Standby mode.
The Write command sequencer stops during Standby mode and resumes operation after wakeup.
A started Flash write/erase operation continues during Standby mode. However the interrupt functions of the Flash cannot be used to wakeup from Standby mode because the clock (CLKB) for the Flash interface is stopped in any Standby mode.
An ongoing Flash write/erase operation can increase the current consumption in the Standby mode significantly. Hence it is recommended to check that no Flash program/erase operation is ongoing and that the Write command sequencer is "Idle" before activating a Standby mode.

4. Voltage Regulator Operation

The F^{2}MC-16FX MCU are equipped with an on-chip voltage regulator which generates the power supply for the core logic out of the external power supply. This regulator has different operation modes with programmable output voltage. This feature can be used to optimize power consumption, especially in Standby modes.
This chapter describes the different operation modes of the regulator and how to set the output voltage.

■ High Power Mode

The "High Power mode" is the default operation mode of the voltage regulator. In this mode, the regulator is able to provide the maximum current consumed by the MCU in any mode. The output voltage for this mode is individually programmable. The default output voltage is 1.8V (adjusted to the process technology) and the current consumption of the regulator itself is several ten μA.
The voltage regulator is operating in this mode when the MCU is in Run mode or when the Main oscillator or the 2MHz RC oscillator is enabled (unless the SMCR:LPMSS bit is set to "1").

■ Low Power Mode

The "Low Power Mode" is the alternative operation mode of the voltage regulator. In the Low Power Mode, the regulator is able to provide a maximum transient current of ~1mA. The current consumption of the regulator itself is reduced to ~5μA.
Two different Low Power Modes are available. The output voltage for each mode is individually programmable. The default output voltage for both modes is 1.8V:

● Low Power Mode A

The voltage regulator is operating in this mode in Sub Sleep, Sub Timer, RC (100kHz) Sleep and RC (100kHz) Timer mode. The Main oscillator and the 2MHz RC oscillator must be disabled.
After setting the SMCR:LPMSS bit to "1", this mode can be selected in all Sleep and Timer modes with disabled PLL.

Note:

Selecting this mode using the SMCR:LPMSS bit is currently under evaluation and must not be used.

● Low Power Mode B

The voltage regulator is operating in this mode only in Stop mode.

4.1. Changing the Voltage Regulator operation mode

The voltage regulator operation modes are automatically controlled by hardware depending on the selected CPU operation mode. However in some modes and under certain conditions, it is permitted to manually change the regulator from High Power mode to Low Power mode for further current saving.

■ Overview

The current consumption of the MCU core depends on the operation mode of the MCU (Run, Sleep, Timer or Stop mode), the activated oscillators and PLL and the selected frequencies (CLKS1, CLKS2, CLKB, CLKP1 and CLKP2).

The automatic voltage regulator control logic sets the regulator to the High Power mode in CPU Run mode or whenever the Main or the 2MHz RC oscillator is activated. In this High Power mode, the regulator is able to drive any current consumed by the core. This guarantees a safe MCU operation independent of the selected CPU operation mode and the Main clock frequency.

Certain applications however require a very small MCU current consumption in dedicated Low Power CPU operation modes. This can be achieved by manually setting the voltage regulator to the Low Power mode even if the Main or 2MHz RC oscillator is active. This reduces the MCU current consumption by 20-30µA. Using this feature is permitted only for CPU operation modes with a small dynamic current consumption.

In the automatic voltage regulator control, the stabilization time of the regulator takes place during the stabilization of the oscillators or during the fixed Core Bus start delay time.

■ Setting Voltage regulator to Low Power mode by user program

The voltage regulator can be set to the Low Power mode by controlling the SMCR:LPMSS bit.

● Switching to the Low Power mode with the SMCR:LPMSS bit

This bit can be used to switch the voltage regulator to Low Power mode in certain Sleep and Timer modes. Setting the SMCR:LPMSS bit to "1" switches the voltage regulator to the Low Power mode with the next transition to a Sleep or Timer mode. This bit has no effect as long as the MCU is in Run mode. Setting the SMCR:LPMSS bit is possible before or together with setting the SMCR:SMS Standby mode request bits.

A wakeup interrupt switches the voltage regulator automatically back to the High Power mode. The transition to Run mode must be delayed with the Core Bus start delay function by 100µs to allow a sufficient stabilization of the voltage regulator.

At wakeup from Sub Sleep/Timer mode or RC (100kHz) Sleep/Timer mode, the Core Bus start delay time is automatically set to 16 CLKB clock cycles. However at wakeup from Main Sleep/Timer or RC (2MHz) Sleep/Timer mode with SMCR:LPMSS="1", a sufficiently long Core Bus start delay time must be selected manually.

■ Permitted configurations for setting the SMCR:LPMSS bit

The voltage regulator can be switched to the Low Power mode (setting SMCR:LMPSS to "1") in the following configurations:

Table 4-1 Permitted configurations for manually using the Low Power mode A

Operation mode	RC oscillator	Main oscillator	PLL	Sub oscillator
RC sleep*	Active (set to 100kHz or 2MHz)	Active with $f_{osc} \leq 4$MHz or disabled	Disabled	Active or disabled
RC timer				
Main sleep*	Active (set to 100kHz or 2MHz) or disabled	Active with $f_{osc} \leq 4$MHz		
Main timer				
Sub sleep*		Active with $f_{osc} \leq 4$MHz or disabled		Active
Sub timer				

*: System clock 2 must be set to Main clock, RC clock or Sub clock.

Note:

Check additional requirements regarding the C-Pin capacitor as described in the datasheet when using the SMCR:LPMSS bit.

4.2. Setting the Voltage Regulator Output Voltage

The output voltage of the regulator can be programmed to adjust the core voltage depending on the required performance or current consumption.

■ Voltage Setting in High Power Mode

The default output voltage of the regulator is 1.8V. This is the standard core voltage for the 0.18µm process. Changing the output voltage of the regulator in High Power mode is prohibited by Fujitsu Semiconductor Limited.

■ Voltage Setting in Low Power Modes

The default output voltage of the regulator in Low Power mode A and B is 1.8V.

● Changing the Core Voltage in Low Power Mode A

Changing the output voltage of the regulator in the Low Power mode A is prohibited by Fujitsu Semiconductor Limited.

● Changing the Core Voltage in Low Power Mode B

For applications which require a small current consumption in Stop mode, it is possible to reduce the core voltage to 1.2V by setting the VRCR:LPMB[1:0] bits to "00" before entering Stop mode. This reduces the leakage current of the MCU.
See datasheet for absolute values of the Stop mode current at 1.2V and 1.8V core voltage.

5. Standby Modes

The standby modes include the Sleep (RC Sleep, Main Sleep, PLL Sleep, Sub Sleep), Timer (RC Timer, Main Timer, PLL Timer, Sub Timer) and Stop modes.

■ Operation Status during Standby Mode

Table 5-1 shows the status of the clocks, CPU, Peripherals and external pins for the different standby modes.

Table 5-1 Operation Status during Standby Mode

Standby mode		Condition for switch	RC clock	Main clock	PLL clock	Sub clock	Bus clock	Peripheral clock 1	Peripheral clock 2	Pins	Release event
Sleep mode	RC Sleep mode	SC1M=00 SMS=01	Active	depends on MCE and SC2M bits	depends on PCE, MCE and SC2M bits	depends on SCE and SC2M bits	Stopped	RC clock	depends on SC2M bits	Active	Power, External, Watchdog or Clock stop reset, Interrupt
	Main Sleep mode	SC1M=01 SMS=01	depends on RCE and SC2M bits	Active	depends on PCE and SC2M bits	depends on SCE and SC2M bits		Main clock			
	PLL Sleep mode	SC1M=10 SMS=01	depends on RCE and SC2M bits	Active	Active	depends on SCE and SC2M bits		PLL clock			
	Sub Sleep mode	SC1M=11 SMS=01	depends on RCE and SC2M bits	depends on MCE and SC2M bits	depends on PCE, MCE and SC2M bits	Active		Sub clock			
Timer mode	RC Timer mode	SC1M=00 SMS=10	Active	depends on MCE and SC2M bits	depends on PCE, MCE and SC2M bits	depends on SCE and SC2M bits		Stopped(*)		depends on SPL bit (Hi-Z or last value retained)	
	Main Timer mode	SC1M=01 SMS=10	depends on RCE and SC2M bits	Active	depends on PCE and SC2M bits	depends on SCE and SC2M bits					
	PLL Timer mode	SC1M=10 SMS=10	depends on RCE and SC2M bits	Active	Active	depends on SCE and SC2M bits					
	Sub Timer mode	SC1M=11 SMS=10	depends on RCE and SC2M bits	depends on MCE and SC2M bits	depends on PCE, MCE and SC2M bits	Active					
Stop mode	Stop mode	SMS=11	Stopped					Stopped			Power or External reset, Interrupt

*: The RC clock timer, Main clock timer and Sub clock timer operate if the corresponding clock was not stopped
SC1M: System Clock 1 Monitor bits of clock monitor register (CKMR)
SC2M: System Clock 2 Monitor bits of clock monitor register (CKMR)
SMS: Standby Mode Select bits of standby mode control register (SMCR)
SPL: Pin state setting bit of standby mode control register (SMCR)
Hi-Z: High-impedance

5.1. Sleep mode (RC Sleep, Main Sleep, PLL Sleep, Sub Sleep Mode)

This mode causes the CPU operating clock (Bus clock CLKB) to stop while other components continue to operate. Transition to Sleep mode is done by writing to the standby mode control register (SMCR). Depending on the currently active clock mode (as indicated by the SC1M System Clock 1 Monitor bits), the MCU switches either to RC Sleep, Main Sleep, PLL Sleep or Sub Sleep mode.

■ Functions in Sleep Mode

● CPU, Internal Memory and Data Retention

In Sleep mode, the clock (Bus clock CLKB) supplying the CPU and the internal memory is stopped. However the contents of dedicated registers, such as accumulator, and the internal RAM are retained.

● Resources

The resource clocks (CLKP1 and CLKP2) are active in Sleep mode and the activated resources are operating in their last configuration.

● Source Clock Timers

The oscillators and corresponding source clock timers (RC clock timer, Main clock timer and Sub clock timer) are not affected by Sleep mode. They keep running according to their configuration before transition to Sleep mode.

● Resets and Interrupts

Resets and Interrupts are active in Sleep mode and can be used to release the Sleep mode.

● External Bus and Hold Function (only devices with External Bus)

The external bus is stopped in Sleep mode, however the external bus hold function is active (setting external bus pins to Hi-Z controlled by HRQ pin if hold function is enabled).

● Status of Pins

During Sleep mode, all pins (excluding those used for the external bus if present) retain their previous function.

■ Switching to Sleep Mode

Writing "01" to the SMS bits of the standby mode control register (SMCR) requests a switch to a Sleep mode. Depending on the current clock mode as indicated by the SC1M bits of the clock monitor register (CKMR), a transition to the corresponding Sleep mode is requested (RC Run -> RC Sleep, Main Run -> Main Sleep, PLL Run -> PLL Sleep, Sub Run -> Sub Sleep mode). The transition to Sleep mode takes place within a few clock cycles after the request.

Note:

To make sure instructions following the SMS write instruction are executed after wakeup from Sleep mode (and not before transition to Sleep mode), poll the SMS bits after setting to "01". Branch to the next instruction only when the SMS bits are cleared to "00". Reading "01" means that the transition to the Sleep mode has not been performed yet (transition request is still pending).

Note:

When a transition to Sleep Mode should be performed after changing the clock mode (by writing a new value to the System Clock 1 Select "SC1S" or System Clock 2 Select "SC2S" bits of the CKSR register), make sure the clock mode transition has been performed before writing the SMS bits. Clock mode transitions are delayed by the clock stabilization and synchronization mechanism. Always confirm the correct clock mode by reading the SC1M/SC2M (System Clock 1/2 Monitor) bits of the CKMR register before setting the SMS bits.
Only change the setting of the SC1S/SC2S bits when the SMS bits are "00" (no Standby mode transition request pending).

● Interrupt Request when Switching to Sleep Mode

When a hardware interrupt is pending, then writing to the SMS bits has no effect.

● Flash Program/Erase when Switching to Sleep Mode

No Flash program or erase command should be submitted or ongoing when setting the SMS bits. Otherwise the current consumption in Sleep mode will be increased and the setting of ESMCR:FPDS has no effect.
The interrupt functions of the Flash cannot be used for a wakeup from Sleep mode since the clock CLKB is disabled.

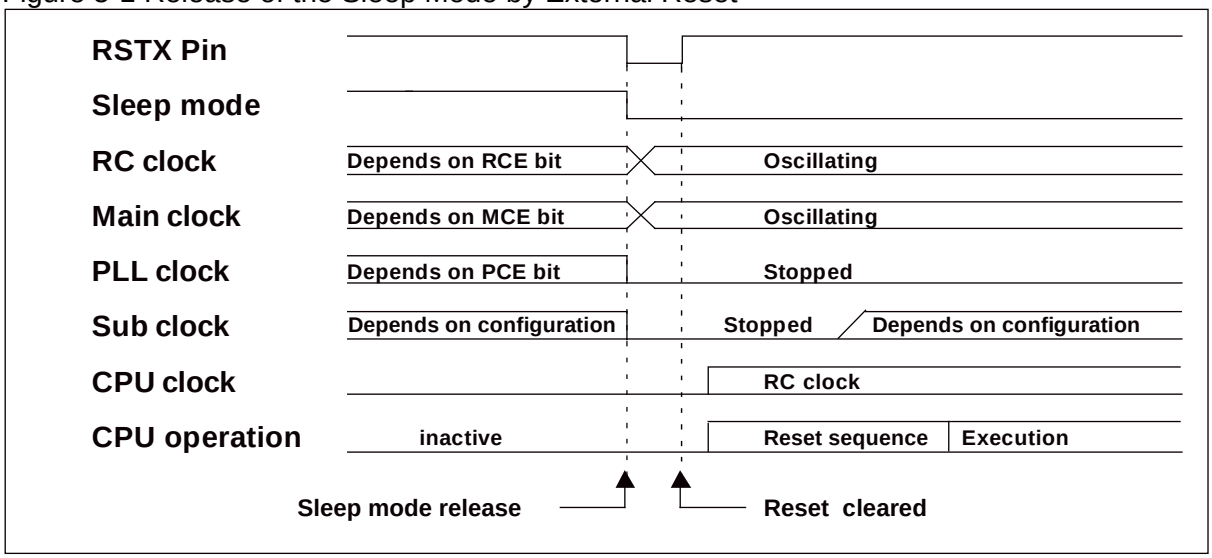
■ Release of Sleep Mode

The standby mode control circuit releases Sleep modes when a reset or an interrupt occurs. See also Table 6-1 for an overview.

● Return by a Reset

In case of a reset (Power, External, Clock stop detection or Watchdog reset), the MCU changes to the RC Run mode, independent of the last selected clock mode.

Figure 5-1 Release of the Sleep Mode by External Reset



● Return by an Interrupt

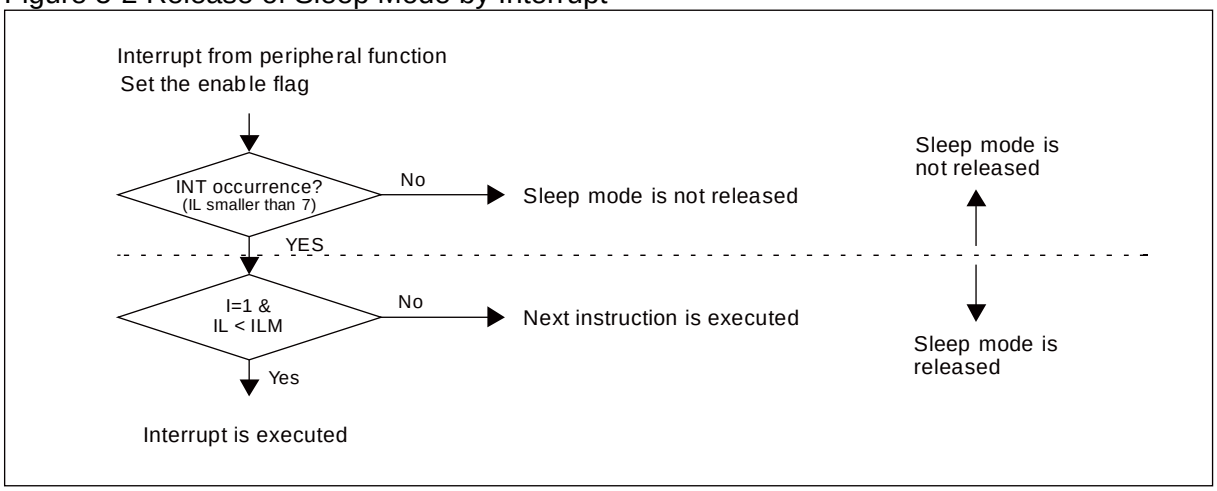
If an NMI interrupt or another interrupt request of higher than level seven is issued from a peripheral circuit during a Sleep mode, the MCU leaves the Sleep mode and returns to the corresponding Run mode (RC Sleep -> RC Run, Main Sleep -> Main Run, PLL Sleep -> PLL Run, Sub Sleep -> Sub Run). After the Sleep mode is released, the interrupt is handled as an ordinary interrupt.

If the interrupt is accepted according to the setting of the I flag of the condition code register (CCR), the interrupt level mask register (ILM) and interrupt control register (ICR), then the CPU finishes the last instructions which were stored in the pipeline and then executes the interrupt routine. After return from this interrupt, the CPU executes the instruction following the last instruction executed before entering the interrupt routine.

If the interrupt is not accepted, the CPU directly executes the instruction following the last instruction executed before entering the Sleep mode.

Figure 5-2 "Release of Sleep Mode by Interrupt" shows the release of a Sleep mode when an interrupt occurs (not NMI).

Figure 5-2 Release of Sleep Mode by Interrupt



5.2. Timer Mode (RC Timer, Main Timer, PLL Timer, Sub Timer mode)

This mode causes all functions, excluding oscillators, PLL and source clock timers, to stop. Transition to Timer mode is done by writing to the standby mode control register (SMCR). Depending on the currently active clock mode (as indicated by the SC1M System Clock 1 Monitor bits), the MCU switches either to RC Timer, Main Timer, PLL Timer or Sub Timer mode.

■ Functions in Timer Mode

● CPU, Internal Memory and Data Retention

In Timer mode, the Bus clock (CLKB) supplying the CPU and the internal memory is stopped. However the contents of dedicated registers, such as accumulator, and the internal RAM are retained.

● Resources

The resource clocks (CLKP1 and CLKP2) are stopped in Timer mode and all resources are frozen in their last state.

● Source Clock Timers

The oscillators and corresponding source clock timers (RC clock timer, Main clock timer and Sub clock timer) are not affected in Timer mode. They keep running according to their configuration before transition to Timer mode.

● Resets and Interrupts

Resets and Interrupts are active in Timer mode and can be used to release the Timer mode.

● External Bus and Hold Function (only devices with External Bus)

The external bus is stopped in Timer mode, however the external bus hold function is active (setting external bus pins to Hi-Z controlled by HRQ pin if hold function is enabled).

● Status of Pins

The SPL bit of the standby mode control register (SMCR) controls whether the external pins in the Timer mode retain the state they had immediately before switching to the Timer mode or go to the high-impedance state.

■ Switching to Timer Mode

Writing "10" to the SMS bits of the standby mode control register (SMCR) requests a switch to a Timer mode. Depending on the current clock mode as indicated by the SC1M bits of the clock monitor register (CKMR), a transition to the corresponding Timer mode is requested (RC Run -> RC Timer, Main Run -> Main Timer, PLL Run -> PLL Timer, Sub Run -> Sub Timer mode). The transition to Timer mode takes place within a few clock cycles after the request.

Note:

To make sure instructions following the SMS write instruction are executed after wakeup from Timer mode (and not before transition to Timer mode), poll the SMS bits after setting to "10". Branch to the next instruction only when the SMS bits are cleared to "00". Reading "10" means that the transition to the Timer mode has not been performed yet (transition request is still pending).

Note:

When a transition to Timer Mode should be performed after changing the clock mode (by writing a new value to the System Clock 1 Select "SC1S" or System Clock 2 Select "SC2S" bits of the CKSR register), make sure the clock mode transition has been performed before writing the SMS bits. Clock mode transitions are delayed by the clock stabilization and synchronization mechanism. Always confirm the correct clock mode by reading the SC1M/SC2M (System Clock 1/2 Monitor) bits of the CKMR register before setting the SMS bits.
Only change the setting of the SC1S/SC2S bits when the SMS bits are "00" (no Standby mode transition request pending).

● Interrupt Request when Switching to Timer Mode

When a hardware interrupt is pending, then writing to the SMS bits has no effect.

● Flash Program/Erase when Switching to Timer Mode

No Flash program or erase command should be submitted or ongoing when setting the SMS bits. Otherwise the current consumption in Timer mode will be increased and the setting of ESMCR:FPDS has no effect. The interrupt functions of the Flash cannot be used for a wakeup from Timer mode since the Bus clock (CLKB) is disabled.

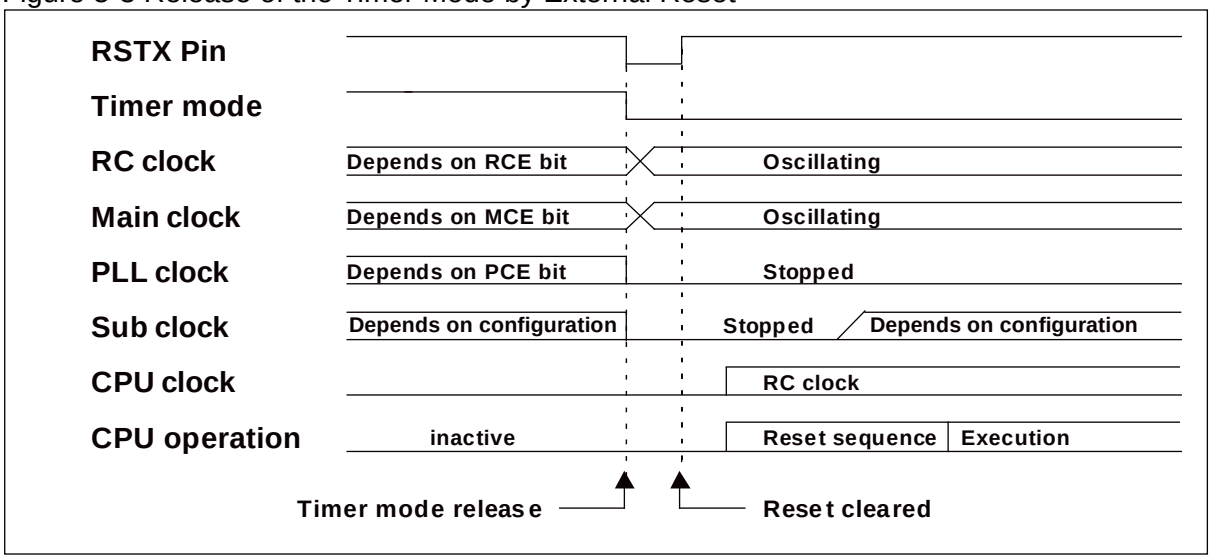
■ Release of Timer Mode

The standby control circuit releases Timer modes when a reset or an interrupt occurs. See also Table 6-1 for an overview.

● Return by a Reset

In case of a reset (Power, External, Clock stop detection or Watchdog reset), the MCU changes to the RC Run mode, independent of the last selected clock mode.

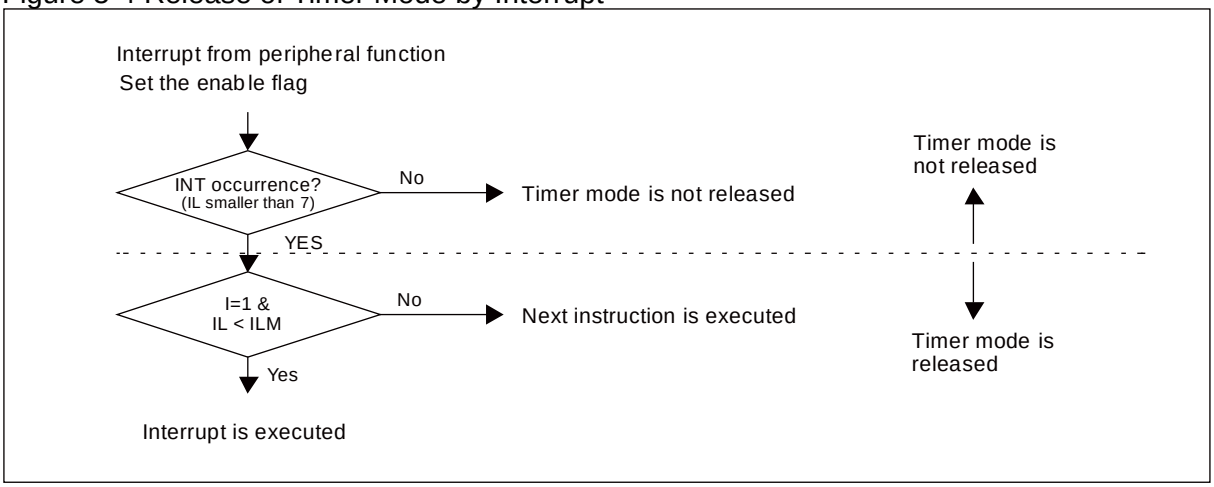
Figure 5-3 Release of the Timer Mode by External Reset



● Return by an Interrupt

If an NMI interrupt is asserted or if a source clock timer or External interrupt generates an interrupt request of higher than level seven during a Timer mode, the Timer mode is released and the MCU returns to the corresponding Run mode (RC Timer -> RC Run, Main Timer -> Main Run, PLL Timer -> PLL Run, Sub Timer -> Sub Run). After the Timer mode is released, the interrupt is handled as an ordinary interrupt.
If the interrupt is accepted according to the setting of the I flag of the condition code register (CCR), the interrupt level mask register (ILM) and interrupt control register (ICR), then the CPU finishes the last instructions which were stored in the pipeline and then executes the interrupt routine. After return from this interrupt, the CPU executes the instruction following the last instruction executed before entering the interrupt routine.
If the interrupt is not accepted, the CPU directly executes the instruction following the last instruction executed before entering the Timer mode.
Figure 5-4 "Release of Timer Mode by Interrupt" shows the release of a Timer mode when an interrupt occurs (not NMI).

Figure 5-4 Release of Timer Mode by Interrupt



5.3. Stop Mode

Because this mode causes all oscillators to stop and inactivates all functions, data can be retained by the lowest power consumption.

■ Functions in Stop Mode

● CPU, Internal Memory and Data Retention

In Stop mode, the Bus clock (CLKB) supplying the CPU and the internal memory is stopped. However the contents of dedicated registers, such as accumulator, and the internal RAM are retained.

● Resources

The resource clocks (CLKP1 and CLKP2) are stopped and all resources are frozen in their last state.

● Source Clock Timers and Clocks

The source clock timers (RC clock timer, Main clock timer and Sub clock timer) are all stopped and cleared together with all oscillators and the PLL. All oscillator ready flags and the PLL ready flag are cleared.

● Resets and Interrupts

Resets and Interrupts are active in Stop mode and can be used to release the Stop mode.

● External Bus and Hold Function (only devices with External Bus)

The external bus is stopped in Stop mode, however the external bus hold function is active (setting external bus pins to Hi-Z controlled by HRQ pin if hold function is enabled).

● Status of Pins

The SPL bit of the standby mode control register (SMCR) controls whether the external pins in the Stop mode retain the state they had immediately before switching to the Stop mode or go to the high-impedance state.

■ Switching to Stop Mode

Writing "11" to the SMS bits of the standby mode control register (SMCR) requests a switch to Stop mode, independent of the current clock mode. The transition to Stop mode takes place within a few clock cycles after the request.

Note:

To make sure instructions following the SMS write instruction are executed after wakeup from Stop mode (and not before transition to Stop mode), poll the SMS bits after setting to "11_B". Branch to the next instruction only when the SMS bits are cleared to "00_B". Reading "11_B" means that the transition to the Stop mode has not been performed yet (transition request is still pending).

Note:

Setting the SMS bits to "11_B" (Stop mode) is allowed together with changing the clock mode (by writing a new value to the SC1S or SC2S System Clock Select bits of the CKSR register). The new clock setting becomes effective immediately after leaving the Stop mode. In difference to the Sleep and Timer mode, it is not necessary to read the SC1M or SC2M (System Clock Monitor) bits of the CKMR register and wait for the clock mode transition before setting the SMS bits. However do not change the setting of the SC1S/SC2S bits after writing to the SMS bits (when a Standby mode transition request is pending).
Both System clocks should be set to RC clock when switching to Stop mode to ensure a reliable MCU startup even when the external clock fails. This is mandatory when the Clock stop detection reset function is used.
Never set the SC1S or SC2S bits to a clock which does not exist (for example "Sub clock mode" when the sub oscillator function is not activated or no crystal/resonator is connected to sub oscillator pins).

● Interrupt Request when Switching to Stop Mode

When a hardware interrupt is pending, then writing to the SMS bits has no effect.

● Flash Program/Erase when Switching to Stop Mode

No Flash program or erase command should be submitted or ongoing when setting the SMS bits. Otherwise the current consumption in Stop mode will be increased and the setting of ESMCR:FPDS has no effect. The interrupt functions of the Flash cannot be used for a wakeup from Stop mode since all clocks are disabled.

■ Release of Stop Mode

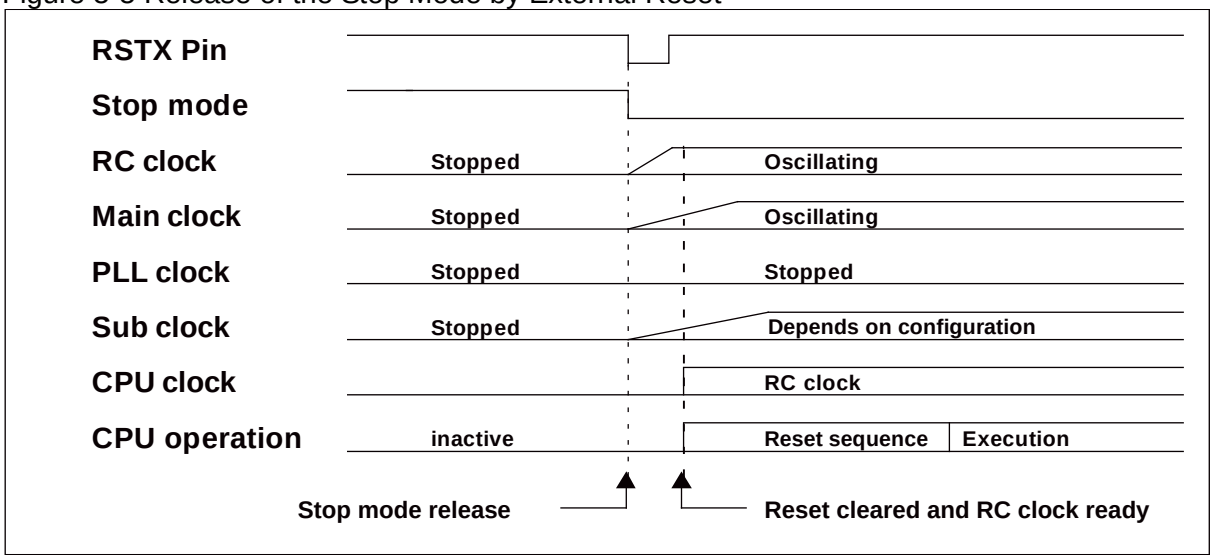
The standby control circuit releases the Stop mode when a reset or an interrupt occurs. See also Table 6-1 for an overview.

● Return by a Reset

In case of a Power reset or External reset, the MCU changes to the RC Run mode, independent of the last selected clock mode.

Other types of reset (Clock stop, Software or Watchdog) are not possible in Stop mode.

Figure 5-5 Release of the Stop Mode by External Reset



● Return by an Interrupt

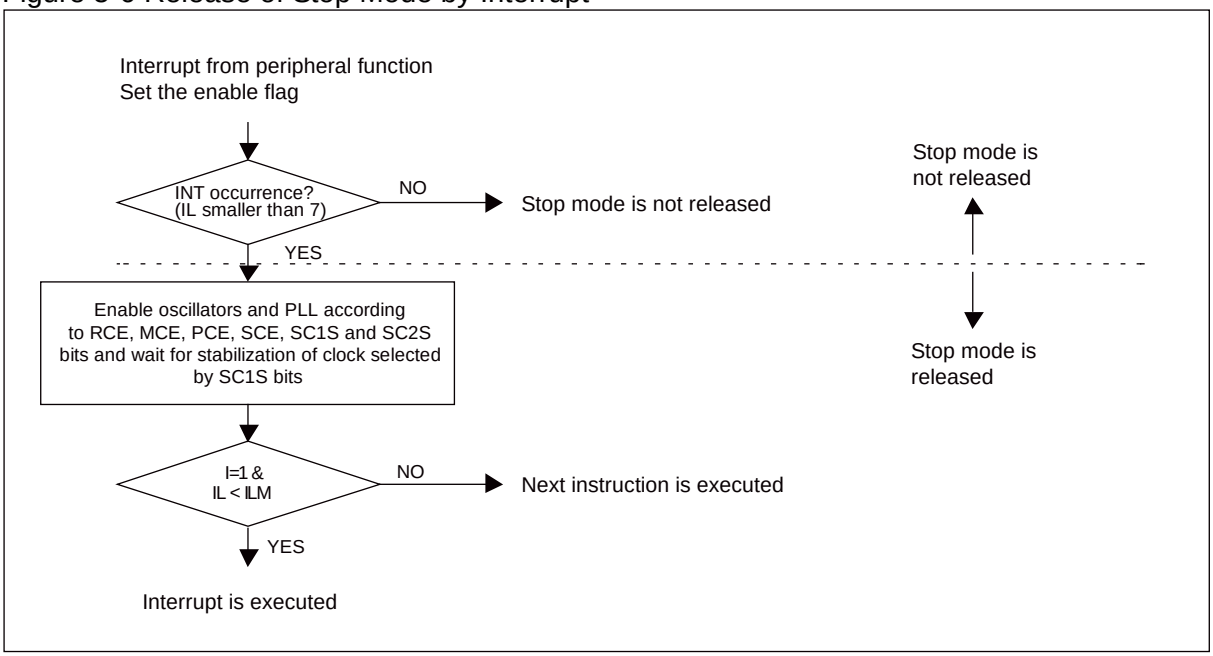
If an NMI interrupt is asserted or if an External interrupt generates an interrupt request of higher than level seven during Stop mode, the Stop mode is released and the MCU returns to the Run mode selected by the SC1S System Clock 1 Select bits (after applying the appropriate clock stabilization wait time). Oscillators are enabled according to the configuration of the CKSR register. After the Stop mode is released, the interrupt is handled as an ordinary interrupt.

If the interrupt is accepted according to the setting of the I flag of the condition code register (CCR), the interrupt level mask register (ILM) and interrupt control register (ICR), then the CPU finishes the last instructions which were stored in the pipeline and then executes the interrupt routine. After return from this interrupt, the CPU executes the instruction following the last instruction executed before entering the interrupt routine.

If the interrupt is not accepted, the CPU directly executes the instruction following the last instruction executed before entering the Stop mode.

Figure 5-6 shows the release of a Stop mode when an interrupt occurs (not NMI).

Figure 5-6 Release of Stop Mode by Interrupt



6. Mode Change Table and Operation Status

Table 6-1 shows the mode change table of the F^{2}MC-16FX MCU and Table 6-2 shows the operation status in each operating mode.

■ Mode Change Table

Table 6-1 Mode Change Table

Current mode (indicated by SC1M bits)	After Reset	After interrupt	By CPU command
RC Run	RC Run	RC Run	Main Run, PLL Run, Sub Run, RC Sleep, RC Timer, Stop
Main Run		Main Run	RC Run, PLL Run, Sub Run, Main Sleep, Main Timer, Stop
PLL Run		PLL Run	RC Run, Main Run, Sub Run PLL Sleep, PLL Timer, Stop
Sub Run		Sub Run	RC Run, Main Run, PLL Run Sub Sleep, Sub Timer, Stop
RC Sleep		RC Run	-
Main Sleep		Main Run	-
PLL Sleep		PLL Run	-
Sub Sleep		Sub Run	-
RC Timer		RC Run	-
Main Timer		Main Run	-
PLL Timer		PLL Run	-
Sub Timer		Sub Run	-
Stop		RC Run, Main Run PLL Run, Sub Run (depending on last value written to SC1S bits)	-

■ Operation Status in Each Operating Mode

Table 6-2 lists the operation status in each operating mode.

Table 6-2 Operation Status in Each Operating Mode

<table>
<tr><th rowspan="2">Operation mode</th><th rowspan="2">RC oscillator and RC clock timer</th><th rowspan="2">Main oscillator and Main clock timer</th><th rowspan="2">PLL clock</th><th rowspan="2">Sub oscillator and Sub clock timer</th><th rowspan="2">Clock source for Bus Clock CLKB</th><th colspan="2">Clock source for Peripheral Clock</th></tr>
<tr><th>CLKP1</th><th>CLKP2</th></tr>
<tr><td>RC run</td><td rowspan="3">Active</td><td rowspan="3">Depending on MCE and SC2M bits</td><td rowspan="3">Depending on PCE, MCE and SC2M bits</td><td rowspan="3">Depending on SCE and SC2M bits</td><td>RC clock</td><td rowspan="2">RC clock</td><td rowspan="2">Depending on SC2M bits</td></tr>
<tr><td>RC sleep</td><td rowspan="2">Stopped</td></tr>
<tr><td>RC timer</td><td colspan="2">Stopped</td></tr>
<tr><td>Main run</td><td rowspan="3">Depending on RCE and SC2M bits</td><td rowspan="3">Active</td><td rowspan="3">Depending on PCE and SC2M bits</td><td rowspan="3">Depending on SCE and SC2M bits</td><td>Main clock</td><td rowspan="2">Main clock</td><td rowspan="2">Depending on SC2M bits</td></tr>
<tr><td>Main sleep</td><td rowspan="2">Stopped</td></tr>
<tr><td>Main timer</td><td colspan="2">Stopped</td></tr>
<tr><td>PLL run</td><td rowspan="3">Depending on RCE and SC2M bits</td><td rowspan="3">Active</td><td rowspan="3">Active</td><td rowspan="3">Depending on SCE and SC2M bits</td><td>PLL clock</td><td rowspan="2">PLL clock</td><td rowspan="2">Depending on SC2M bits</td></tr>
<tr><td>PLL sleep</td><td rowspan="2">Stopped</td></tr>
<tr><td>PLL timer</td><td colspan="2">Stopped</td></tr>
<tr><td>Sub run</td><td rowspan="3">Depending on RCE and SC2M bits</td><td rowspan="3">Depending on MCE and SC2M bits</td><td rowspan="3">Depending on PCE, MCE and SC2M bits</td><td rowspan="3">Active</td><td>Sub clock</td><td rowspan="2">Sub clock</td><td rowspan="2">Depending on SC2M bits</td></tr>
<tr><td>Sub sleep</td><td rowspan="2">Stopped</td></tr>
<tr><td>Sub timer</td><td colspan="2">Stopped</td></tr>
<tr><td>Stop</td><td colspan="7">Stopped</td></tr>
<tr><td>External reset activated</td><td>RC oscillator on, RC clock Timer stopped</td><td>Main oscillator on, Main clock Timer active</td><td>Stopped</td><td>Sub oscillator off, Sub clock Timer stopped</td><td colspan="3">Stopped</td></tr>
</table>

7. Registers

This section explains the configuration and functions of the registers used for the standby mode and voltage regulator control.

■ List of Standby Mode and Voltage Regulator Control Registers

Abbreviated Register Name	Register Name	Reference
SMCR	Standby Mode Control Register	See 7.1
ESMCR	Extended Standby Mode Control Register	See 7.2
VRCR	Voltage Regulator Control Register	See 7.3

7.1. Standby Mode Control Register (SMCR)

This register switches to standby mode and controls the pin functions in Timer and Stop mode.

■ Standby Mode Control Register (SMCR)

SMCR								
bit	7	6	5	4	3	2	1	0
	-	-	-	LPMSS	-	SPL	SMS1	SMS0
Attribute	-	-	-	R/W	-	R/W	R/W	R/W
Initial value	X	X	X	0	X	0	0	0

[bit7 to bit5] -: Undefined

- Always write "0" to these bits.
- The read value of these bits is undefined.
- Read modify write operations to this register have no effect on these bits.

[bit4] LPMSS: Low Power Mode Set in Standby mode Bits

bit	Description
0	Voltage regulator controlled automatically
1	Regulator set to Low Power Mode in Standby mode

- This bit controls the voltage regulator in Standby modes (Sleep and Timer mode with active 2MHz RC clock or active Main clock). See "Section 4 Voltage Regulator Operation" for details.
- This bit is cleared by any reset.
- When this bit is set to "0"(initial value), then the voltage regulator is always set to the "High Power Mode" when ever the 2MHz RC oscillator or the Main oscillator is active.
- Setting this bit to "1", forces the voltage regulator to the "Low Power Mode A" after entering Sleep or Timer mode, even when the 2MHz RC oscillator or Main oscillator is active. This means the regulator output voltage is defined by the VRCR:LPMA[1:0] bits.
- Setting this bit is permitted only under certain conditions as described in section 4 "Voltage Regulator Operation".
- When this bit is set to "1", then a Bus start delay of minimum 100µs must be selected with the ESMCR:BSD[3:0] bits. Always write "0" to these bits.

[bit3] -: Undefined

- Always write "0" to these bits.
- The read value of these bits is undefined.
- Read modify write operations to this register have no effect on this bit.

[bit2] SPL: Pin State Setting Bit (for Timer and Stop mode

bit	Description
0	Previous pin state retained
1	High impedance

- The setting of this bit effects only the Timer mode and Stop mode.
- When this bit is set to "0", the level of the external pins is retained.
- When this bit is set to "1", the status of the external pins changes to high-impedance.
- This bit is initialized to "0" by a reset.

[bit1, bit0] SMS1 and SMS0: Standby Mode Select Bits

- These bits request switching to a Standby mode according to the following table:

bit1	bit0	Description
0	0	Run mode
0	1	Requests a transition to Sleep mode
1	0	Requests a transition to Timer mode
1	1	Requests a transition to Stop mode

- When "01" is written to these bits, a switch to Sleep mode is requested.
- When "10" is written to these bits, a switch to Timer mode is requested.
- When "11" is written to these bits, a switch to Stop mode is requested.
- After writing one of these three values, the bits are locked. It is not possible to select another Standby mode or Run mode.
- These bits are cleared to "00_B" by any pending hardware interrupt request (level smaller than 7), an NMI (if available) and by any reset.
- The read value of these bits shows the status of the Standby mode transition request. Reading a value other than "00_B" means that the transition to the selected Standby mode is still pending while reading "00_B" means that there is no Standby mode transition request pending.

Note:

All input pins are automatically disabled in Timer and Stop mode, except activated external interrupt input pins and the NMI input pin. The setting of the PIE port input enable bits have no effect in Timer and Stop mode.

7.2. Extended Standby Mode Control Register (ESMCR)

This register controls additional features related to standby modes.

■ Extended Standby Mode Control Register (ESMCR)

ESMCR								
bit	7	6	5	4	3	2	1	0
	-	-	-	FPDS	BSD3	BSD2	BSD1	BSD0
Attribute	-	-	-	R/W	R/W	R/W	R/W	R/W
Initial value	X	X	X	0	0	0	0	1

[bit7 to bit5] -: Undefined

- Always write "0" to these bits.
- The read value of these bits is undefined.
- Read modify write operations to this register have no effect on these bits.

[bit4] FPDS: Flash Power Down Select Bit

bit	Description
0	Flash macro not set to "power down" during Standby mode
1	Flash macro set to "power down" during Standby mode

- The setting of this bit controls the Flash macro in standby mode.
- When this bit is set to "1", the power-down/reset mode of the Flash macro is activated in standby mode (Sleep, Timer or Stop mode) when no Flash program/erase operation is ongoing. This reduces the current consumption of the MCU. At wakeup from standby mode, the program start is delayed until the Flash has left the power-down state and enters the "Read/Reset" state.
- When this bit is set to "0" or when a program/erase operation is ongoing at Standby mode entry, then the power-down/reset mode of the Flash macro will not be activated in standby mode. This results in a higher Standby mode current, but faster startup after standby mode release. The Flash macro state is not affected by the standby mode.
- An ongoing Flash write/erase operation is identified by DFSA/B:RDY=0 (Flash is not ready) or DFSA/B:ST[1:0] != "00" (Flash write command sequencer is not idle).
- This bit is initialized to "0" by any reset.

[bit3 to bit0] BSD3 to BSD0: Bus Start Delay Select Bits

- These bits control the bus start delay after wakeup from standby mode according to the following table:

bit3	bit2	bit1	bit0	Description
0	0	0	0	0 Bus clock (CLKB) cycles
0	0	0	1	16 Bus clock (CLKB) cycles
0	0	1	0	32 Bus clock (CLKB) cycles
0	0	1	1	64 Bus clock (CLKB) cycles
0	1	0	0	128 Bus clock (CLKB) cycles
0	1	0	1	256 Bus clock (CLKB) cycles
1	1	1	0	512 Bus clock (CLKB) cycles
0	1	1	1	1024 Bus clock (CLKB) cycles
1	X	X	X	2047 Bus clock (CLKB) cycles

- These bits define the delay between starting the Bus clock (CLKB) and starting the first transfer on the core bus after standby mode release.
- These bits are initialized to "0001_B" (16 Bus clock (CLKB) cycles) by any reset.

Note:

Do not change these bits, because this function is currently under evaluation by Fujitsu Semiconductor Limited.

7.3. Voltage Regulator Control Register (VRCR)

The voltage regulator control register defines the output voltage of the internal Voltage Regulator in different operation modes

■ Voltage Regulator Control Register (VRCR)

See "Section 4. Voltage Regulator Operation" for more details.

VRCR								
bit	7	6	5	4	3	2	1	0
	HPM1	HPM0	-	LPMA1	LPMA0	-	LPMB1	LPMB0
Attribute	R/W	R/W	-	R/W	R/W	-	R/W	R/W
Initial value	1	0	X	1	0	X	1	0

[bit7, bit6] HPM1, HPM0: High Power Mode Select Bits

- These bits select the output voltage of the regulator in High Power Mode according to the following table:

bit7	bit6	Description
0	0	1.6 V (Setting prohibited)
0	1	1.7 V (Setting prohibited)
1	0	1.8 V
1	1	1.9 V (Setting prohibited)

- Any reset initializes these bits to "10" (1.8V).
- The High Power Mode of the regulator is the standard operation mode which is used in all Run modes and when the Main oscillator, PLL or 2MHz RC oscillator is active.

Note:

Do not change these bits, because permitted settings are currently under evaluation by Fujitsu Semiconductor Limited.

[bit5] -: Undefined

- Always write "1" to this bit.
- The read value of this bit is undefined.
- Read modify write operations to this register have no effect on this bit.

[bit4, bit3] LPMA1, LPMA0: Low Power Mode A Select Bits

- These bits select the output voltage of the regulator in Low Power Mode A according to the following table:

bit4	bit3	Description
0	0	1.2 V (Setting prohibited)
0	1	1.6 V (Setting prohibited)
1	0	1.8 V
1	1	1.9 V (Setting prohibited)

- Any reset initializes these bits to "10" (1.8V).
- The Low Power Mode A of the regulator is used in Sub Sleep, Sub Timer, RC (100kHz) Sleep and RC (100kHz) Timer mode when the Main oscillator, PLL and 2MHz RC oscillator are all disabled. After setting SMCR:LPMSS to "1", the Low Power Mode A will be used independently of the activation of the Main oscillator and the 2MHz RC oscillator.

Note:

Do not change these bits, because permitted settings are currently under evaluation by Fujitsu Semiconductor Limited.

[bit2] -: Undefined

- Always write "1" to this bit.
- The read value of this bit is undefined.
- Read modify write operations to this register have no effect on this bit.

[bit1, bit0] LPMB1, LPMB0: Low Power Mode B Select Bits

- These bits select the output voltage of the regulator in Low Power Mode B according to the following table:

bit1	bit0	Description
0	0	1.2 V (Setting prohibited)
0	1	1.6 V (Setting prohibited)
1	0	1.8 V
1	1	1.9 V (Setting prohibited)

- Any reset initializes these bits to "10" (1.8V).
- The Low Power Mode B of the regulator is used only in Stop mode.

Note:

Do not change these bits, because permitted settings are currently under evaluation by Fujitsu Semiconductor Limited.

CHAPTER: SOURCE CLOCK TIMERS

This chapter explains the functions and operations of the three source clock timers (RC clock timer, Main clock timer and Sub clock timer).

Management Code: 96F6CLKTIMER-E01.0

1. Overview

The F^{2}MC-16FX MCU offers 3 independent source clock timers (RC clock timer, Main clock timer and Sub clock timer) which can issue interrupts at specified intervals and which are used to measure the oscillation stabilization time.

■ RC Clock Timer

The RC clock timer is clocked by the output signal of the internal RC oscillator (CLKRC) and is always running when the RC oscillator is enabled. It is also used to measure the RC clock stabilization wait time.

■ Main Clock Timer

The Main clock timer is clocked by the output signal of the Main oscillation circuit (CLKMC) and is always running when the Main oscillator is enabled. It is also used to measure the Main clock stabilization wait time.

■ Sub Clock Timer

The Sub clock timer is clocked by the output signal of the Sub oscillation circuit (CLKSC) and is always running when the Sub oscillator is enabled. It is also used to measure the Sub clock stabilization wait time.

2. RC Clock Timer

The RC clock timer consists of a 23-bit counter and a control register. The 23-bit counter divides the RC clock CLKRC. The RC clock timer issues interrupts at specified intervals based on carry signals of the RC clock counter.

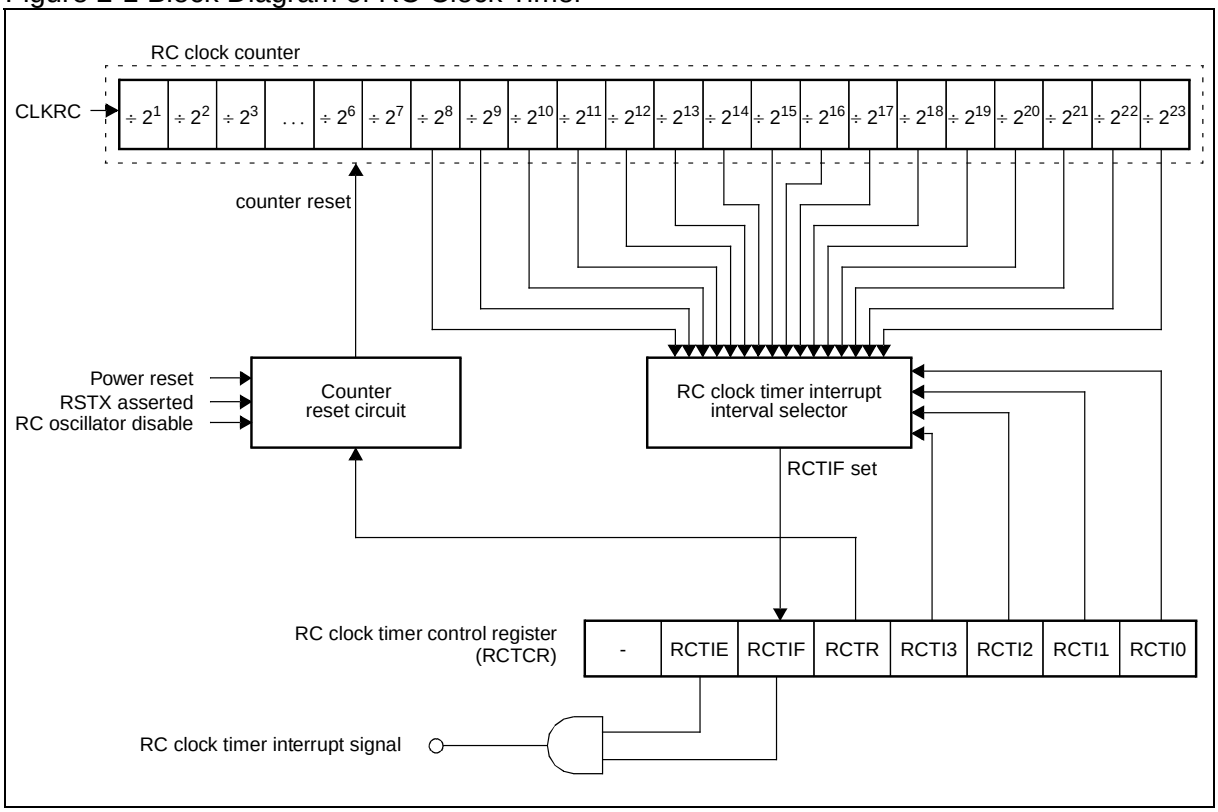
■ Outline of the RC Clock Timer

A Power or External reset initializes the RC clock timer to "0". The RC clock timer is also initialized when the RC clock is disabled (at transition to Stop mode or by setting RCE to "0"), or by writing "0" to the RCTR bit of the RCTCR register. The RC clock timer continues counting as long as the RC clock is supplied. The RC clock timer is used to measure the RC clock stabilization wait time and can be used to generate interval interrupts.

■ Block Diagram of RC Clock Timer

Figure 2-1 shows a block diagram of the RC clock timer.

Figure 2-1 Block Diagram of RC Clock Timer



2.1. Operations

The RC clock timer functions as an interval timer for generating interrupts at specified intervals.

■ RC Clock Counter

The RC clock counter consists of a 23-bit counter that is clocked with the RC clock CLKRC. When the RC clock is active, the RC clock counter always keeps counting. The RC clock frequency is set by the CKFCR: RCFS bit.

The RC clock counter is cleared when the RC clock is stopped (transition to Stop mode or RC clock disabled with CKSR: RCE bit and confirmed by the CKMR: RCM bit), by writing "0" to the RCTR bit of the RCTCR register, by a Power reset and an External reset.

Note:

The RC clock counter is cleared and stopped as long as the External reset is asserted. This also stops the RC clock stabilization time counter. Hence the RC clock stabilization time is delayed by RSTX and starts counting after deasserting the External reset.

■ Interval Interrupt Function

Interrupts are generated at specified intervals according to the carry signals of the RC clock counter. The RCTIF flag is set at the intervals specified with the RCTI[3:0] bits of the RCTCR register. The interval time starts when the RC clock timer is cleared and starts counting.

The RC clock stabilization time counter is connected to the RC clock counter. Hence when the RC clock is stopped (transition to Stop mode or RC clock disabled with CKSR: RCE bit and confirmed by the CKMR: RCM bit), then the RC clock counter is immediately cleared.

2.2. RC Clock Timer Control Register (RCTCR)

The RC Clock Timer Control Register (RCTCR) is used to control the RC clock timer interval interrupt function and to reset the RC clock timer.

■ Configuration of the RC Clock Timer Control Register (RCTCR)

RCTCR								
bit	7	6	5	4	3	2	1	0
	-	RCTIE	RCTIF	RCTR	RCTI3	RCTI2	RCTI1	RCTI0
Attribute	-	R/W	R/W	W	R/W	R/W	R/W	R/W
Initial value	X	0	0	1	0	0	0	0

[bit7] -: Undefined

- Always write "0" to this bit.
- The read value of this bit is undefined.
- Read modify write operations to this register have no effect on this bit.

[bit6] RCTIE: RC Clock Timer Interrupt Enable Bit

Bit	Description
0	Disable Interrupt
1	Enable Interrupt

- This bit is used to enable interval interrupts based on the RC clock timer.
- This bit is reset to "0" by any reset.

[bit5] RCTIF: RC Clock Timer Interrupt Flag

Bit	Description	
	Read	Write
0	No interrupt	Clear this bit
1	Interrupt requested	No effect

- This is an interrupt request flag for the RC clock timer.
- This bit is set to "1" for each interval specified with the RCTI[3:0] bits.
- When this bit is set to "1", an interrupt request is issued if the interrupt enable bit RCTIE is set to "1".
- This bit is cleared by writing "0" and by any reset.
- This bit should always be cleared when the RC clock timer is reset
- The RC clock timer interrupt interval bits RCTI[3:0] are reset to "0000" (2^8 / CLKRC cycles) by any reset. Hence, at 2^8 / CLKRC cycles after any reset, the RCTIF bit is set to "1".
- "1" is always read by a read-modify-write instruction.

[bit4] RCTR: RC Clock Timer Reset Bit

Bit	Description	
	Read	Write
0	Always reads 1	Reset all bits to 0
1		No effect

- This bit clears all bits of the RC clock counter.

Note:

When clearing the RC clock timer by writing "0" to this bit, also clear the RC clock timer interrupt flag by writing "0" to the RCTIF bit.

[bit3 to bit0] RCTI3 to RCTI0: RC Clock Timer Interrupt Interval Select bits

- These bits control the RC clock timer interrupt interval according to the following table:

bit3	bit2	bit1	bit0	Description
0	0	0	0	2^8 / CLKRC (approx. 128µs/2.5ms)
0	0	0	1	2^9 / CLKRC (approx. 256µs/5.1ms)
0	0	1	0	2^{10} / CLKRC (approx. 512µs/10.2ms)
0	0	1	1	2^{11} / CLKRC (approx. 1ms/20.5ms)
0	1	0	0	2^{12} / CLKRC (approx. 2ms/41ms)
0	1	0	1	2^{13} / CLKRC (approx. 4ms/82ms)
0	1	1	0	2^{14} / CLKRC (approx. 8ms/164ms)
0	1	1	1	2^{15} / CLKRC (approx. 16ms/328ms)
1	0	0	0	2^{16} / CLKRC (approx. 32ms/655ms)
1	0	0	1	2^{17} / CLKRC (approx. 65ms/1.3s)
1	0	1	0	2^{18} / CLKRC (approx. 131m/2.6s)
1	0	1	1	2^{19} / CLKRC (approx. 262ms/5.2s)
1	1	0	0	2^{20} / CLKRC (approx. 524ms/10.4s)
1	1	0	1	2^{21} / CLKRC (approx. 1.05s/21s)
1	1	1	0	2^{22} / CLKRC (approx. 2.1s/42s)
1	1	1	1	2^{23} / CLKRC (approx. 4.2s/84s)

- These bits are initialized to "0000" by any reset.
- When data is written to these bits, bit 5 (RCTIF) should be cleared at the same time.
- Please note that above calculated times are based on the nominal RC clock frequency. The actual frequency and interval times however vary. Please see the datasheet for more details regarding accuracy of the RC clock frequency.

3. Main Clock Timer

The Main clock timer consists of a 23-bit counter and a control register. The 23-bit counter divides the Main clock CLKMC. The Main clock timer issues interrupts at specified intervals based on carry signals of the Main clock counter.

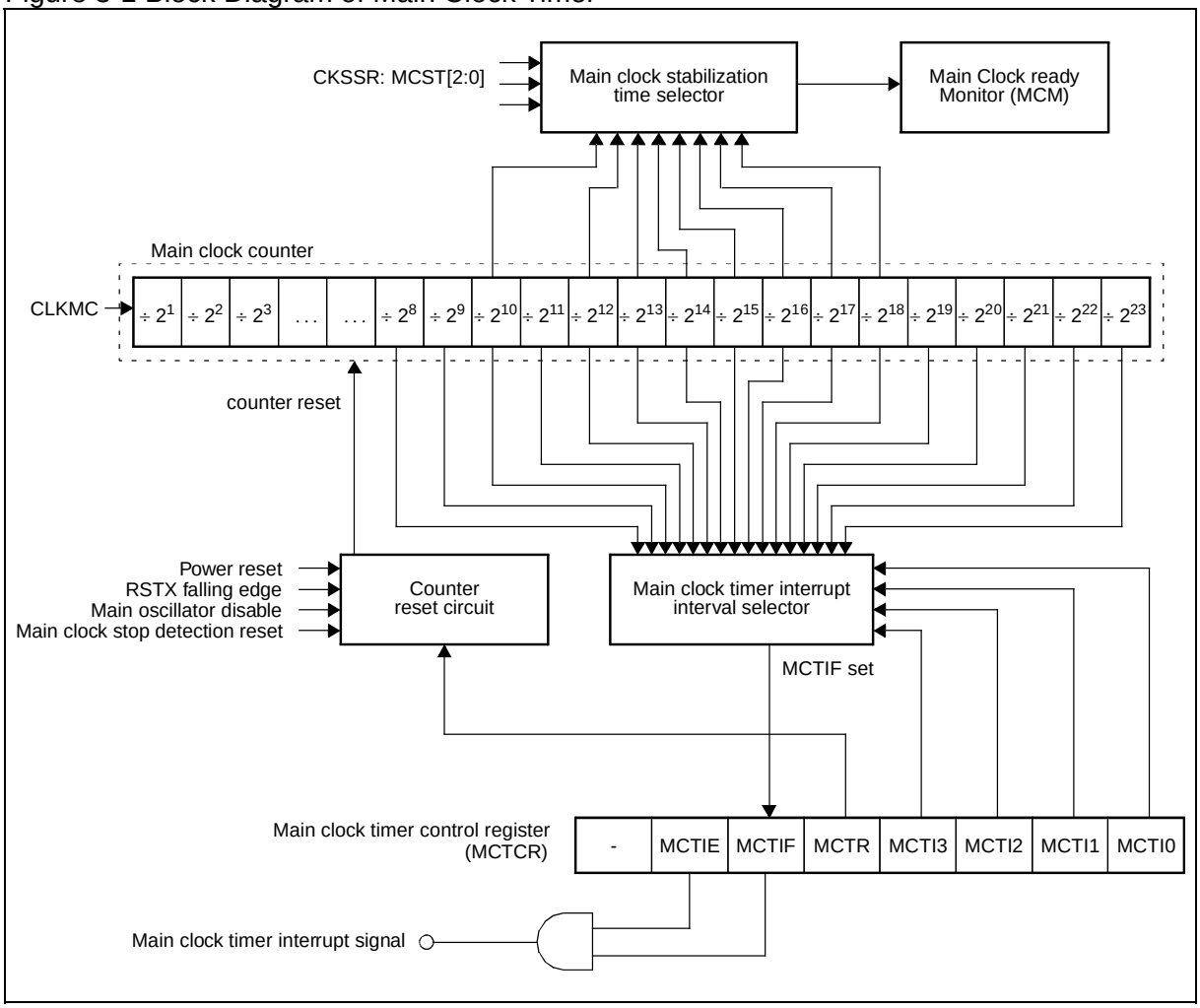
■ Outline of the Main Clock Timer

A Power reset or a falling edge at the RSTX input pin initializes the Main clock timer to "0". The Main clock timer is also initialized when the Main clock is disabled (at transition to Stop mode or by setting MCE to "0"), or by writing "0" to the MCTR bit of the MCTCR register. The Main clock timer continues counting as long as the Main clock is supplied. The Main clock timer is used to measure the Main clock stabilization wait time and can be used to generate interval interrupts.

■ Block Diagram of Main Clock Timer

Figure 3-1 shows a block diagram of the Main clock timer.

Figure 3-1 Block Diagram of Main Clock Timer



3.1. Operations

The Main clock timer functions as an interval timer for generating interrupts at specified intervals and as a timer for waiting for the Main oscillation to stabilize.

■ Main Clock Counter

The Main clock counter consists of a 23-bit counter that is clocked with the Main clock CLKMC. When the Main clock is active, the Main clock counter always keeps counting.
The Main clock counter is cleared when the Main clock is stopped (transition to Stop mode or Main clock disabled with CKSR: MCE bit and confirmed by the CKMR: MCM bit), by writing "0" to the MCTR bit of the MCTCR register, by a Power reset, by an External reset and by a Main clock stop detection reset.

Note:

An External reset only clears and stops the Main clock counter by detecting a falling edge at the RSTX input pin. After 200 RC clock cycles (when the reset extension time is expired), the Main clock counter reset is released and the counter starts counting even if the External reset is still asserted. Hence the Main clock stabilization time takes place already when RSTX is still active.

■ Interval Interrupt Function

Interrupts are generated at specified intervals according to the carry signals of the Main clock counter. The MCTIF flag is set at the intervals specified with the MCTI[3:0] bits of the MCTCR register. The interval time starts when the Main clock timer is cleared and starts counting.
When the Main clock is stopped (transition to Stop mode or Main clock disabled with CKSR: MCE bit and confirmed by the CKMR: MCM bit), the Main clock timer is used as a timer for waiting for the Main oscillation to stabilize upon recovery. Therefore, the Main clock counter is immediately cleared when the Main clock is stopped.

3.2. Main Clock Timer Control Register (MCTCR)

The Main Clock Timer Control Register (MCTCR) is used to control the Main clock timer interval interrupt function and to reset the Main clock timer.

■ Configuration of the Main Clock Timer Control Register (MCTCR)

MCTCR								
bit	15	14	13	12	11	10	9	8
	-	MCTIE	MCTIF	MCTR	MCTI3	MCTI2	MCTI1	MCTI0
Attribute	-	R/W	R/W	W	R/W	R/W	R/W	R/W
Initial value	X	0	0	1	0	0	0	0

[bit15] -: Undefined

- Always write "0" to this bit.
- The read value of this bit is undefined.
- Read modify write operations to this register have no effect on this bit.

[bit14] MCTIE: Main Clock Timer Interrupt Enable Bit

Bit	Description
0	Disable Interrupt
1	Enable Interrupt

- This bit is used to enable interval interrupts based on the Main clock timer.
- This bit is reset to "0" by any reset.

[bit13] MCTIF: Main Clock Timer Interrupt Flag

Bit	Description	
	Read	Write
0	No interrupt	Clear this bit
1	Interrupt requested	No effect

- This is an interrupt request flag for the Main clock timer.
- This bit is set to "1" for each interval specified with the MCTI[3:0] bits.
- When this bit is set to "1", an interrupt request is issued if the interrupt enable bit MCTIE is set to "1".
- This bit is cleared by writing "0" and by any reset.
- This bit should always be cleared when the Main clock timer is reset
- The Main clock timer interrupt interval bits MCTI[3:0] are reset to "0000" (2^8 / CLKMC cycles) by any reset. Hence, at 2^8 / CLKMC cycles after any reset, the MCTIF bit is set to "1".
- "1" is always read by a read-modify-write instruction.

[bit12] MCTR: Main Clock Timer Reset Bit

Bit	Description	
	Read	Write
0	Always reads 1	Reset all bits to 0
1		No effect

- This bit clears all bits of the Main clock counter.

Note:

When clearing the Main clock timer by writing "0" to this bit, also clear the Main clock timer interrupt flag by writing "0" to the MCTIF bit.

[bit11 to bit8] MCTI3 to MCTI0: Main Clock Timer Interrupt Interval Select Bits

- These bits control the Main clock timer interrupt interval according to the following table:

bit11	bit10	bit9	bit8	Description
0	0	0	0	2^8 / CLKMC (approx. 64µs)
0	0	0	1	2^9 / CLKMC (approx. 128µs)
0	0	1	0	2^{10} / CLKMC (approx. 256µs)
0	0	1	1	2^{11} / CLKMC (approx. 512µs)
0	1	0	0	2^{12} / CLKMC (approx. 1ms)
0	1	0	1	2^{13} / CLKMC (approx. 2ms)
0	1	1	0	2^{14} / CLKMC (approx. 4ms)
0	1	1	1	2^{15} / CLKMC (approx. 8ms)
1	0	0	0	2^{16} / CLKMC (approx. 16ms)
1	0	0	1	2^{17} / CLKMC (approx. 32ms)
1	0	1	0	2^{18} / CLKMC (approx. 65ms)
1	0	1	1	2^{19} / CLKMC (approx. 131ms)
1	1	0	0	2^{20} / CLKMC (approx. 262ms)
1	1	0	1	2^{21} / CLKMC (approx. 524ms)
1	1	1	0	2^{22} / CLKMC (approx. 1.049s)
1	1	1	1	2^{23} / CLKMC (approx. 2.097s)

- These bits are initialized to "0000" by any reset.
- When data is written to these bits, bit 13 (MCTIF) should be cleared at the same time.

4. Sub Clock Timer

The Sub clock timer consists of a 17-bit counter and a control register. The 17-bit counter divides the Sub clock CLKSC. The Sub clock timer issues interrupts at specified intervals based on carry signals of the Sub clock counter.

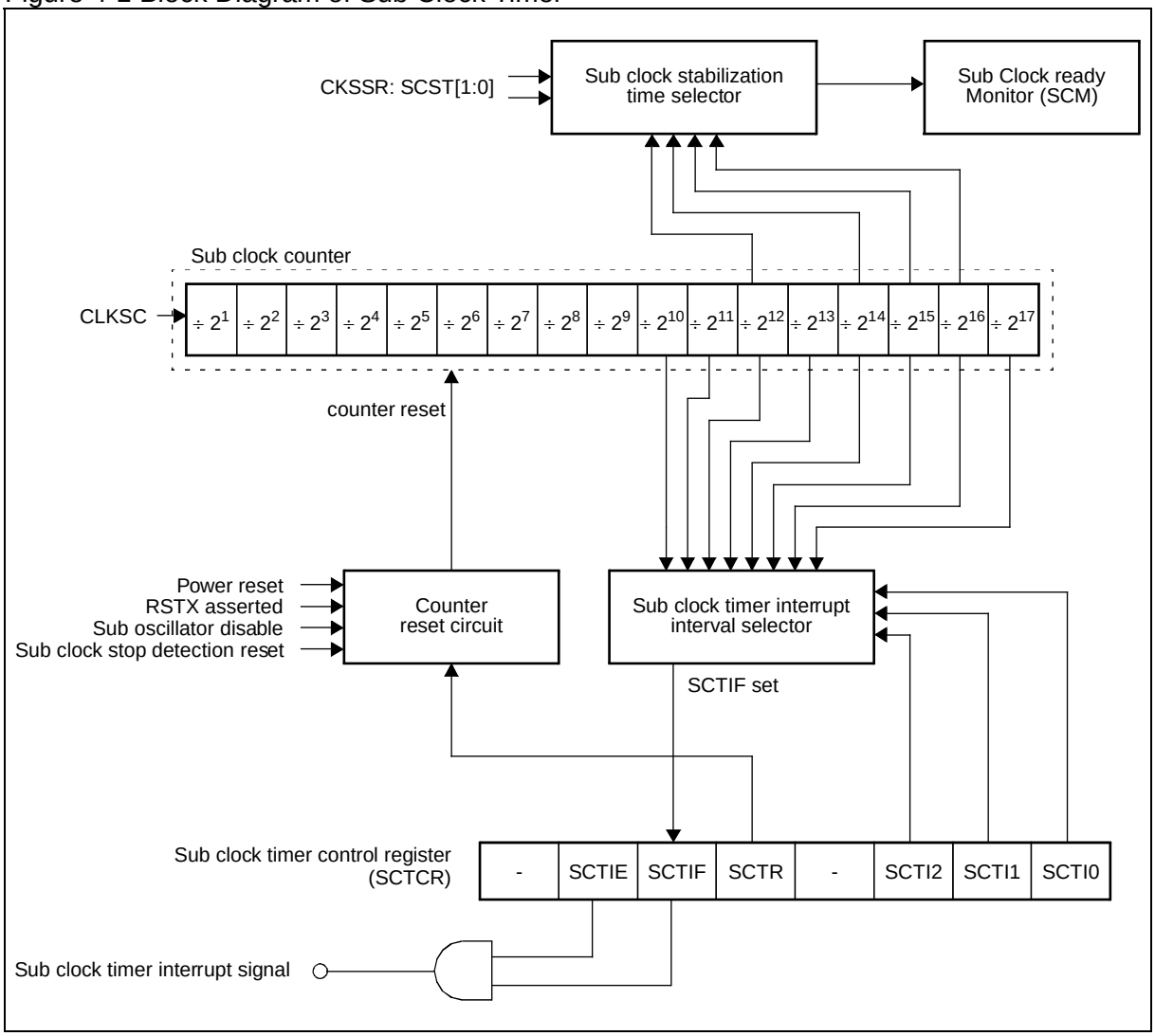
■ Outline of the Sub Clock Timer

A Power or External reset initializes the Sub clock timer to "0". The Sub clock timer is also initialized when the Sub clock is disabled (at transition to Stop mode or by setting SCE to "0"), or by writing "0" to the SCTR bit of the SCTCR register. The Sub clock timer continues counting as long as the Sub clock is supplied. The Sub clock timer is used to measure the Sub clock stabilization wait time and can be used to generate interval interrupts.

■ Block Diagram of Sub Clock Timer

Figure 4-1 shows a block diagram of the Sub clock timer.

Figure 4-1 Block Diagram of Sub Clock Timer



4.1. Operations

The Sub clock timer functions as an interval timer for generating interrupts at specified intervals and as a timer for waiting for the Sub oscillation to stabilize.

■ Sub Clock Counter

The Sub clock counter consists of a 17-bit counter that is clocked with the Sub clock CLKSC. When the Sub clock is active, the Sub clock counter always keeps counting.
The Sub clock counter is cleared when the Sub clock is stopped (transition to Stop mode or Sub clock disabled with CKSR: SCE bit and confirmed by the CKMR: SCM bit), by writing "0" to the SCTR bit of the SCTCR register, by a Power reset, an External reset and by a Sub clock stop detection reset.

Note:

The Sub clock is disabled by a Power or External reset. It is activated by the Boot ROM program after reset release depending on the Sub Oscillator Configuration Marker. Hence the Sub clock stabilization time starts after this activation.

■ Interval Interrupt Function

Interrupts are generated at specified intervals according to the carry signals of the Sub clock counter. The SCTIF flag is set at the intervals specified with the SCTI[2:0] bits of the SCTCR register. The interval time starts when the Sub clock timer is cleared and starts counting.
When the Sub clock is stopped (transition to Stop mode or Sub clock disabled with CKSR: SCE bit and confirmed by the CKMR: SCM bit), the Sub clock timer is used as a timer for waiting for the Sub oscillation to stabilize upon recovery. Therefore, the Sub clock counter is immediately cleared when the Sub clock is stopped.

4.2. Sub Clock Timer Control Register (SCTCR)

The Sub Clock Timer Control Register (SCTCR) is used to control the Sub clock timer interval interrupt function and to reset the Sub clock timer.

■ Configuration of the Sub Clock Timer Control Register (SCTCR)

SCTCR								
bit	7	6	5	4	3	2	1	0
	-	SCTIE	SCTIF	SCTR	-	SCTI2	SCTI1	SCTI0
Attribute	-	R/W	R/W	W	-	R/W	R/W	R/W
Initial value	X	0	0	1	X	0	0	0

[bit7] -: Undefined

- Write always "0".
- Read value is undefined.
- RMW instructions are not affected.

[bit6] SCTIE: Sub Clock Timer Interrupt Enable Bit

Bit	Description
0	Disable Interrupt
1	Enable Interrupt

- This bit is used to enable interval interrupts based on the Sub clock timer.
- This bit is reset to "0" by any reset.

[bit5] SCTIF: Sub Clock Timer Interrupt Flag

Bit	Description	
	Read	Write
0	No interrupt	Clear this bit
1	Interrupt requested	No effect

- This is an interrupt request flag for the Sub clock timer.
- This bit is set to "1" for each interval specified with the SCTI[2:0] bits.
- When this bit is set to "1", an interrupt request is issued if the interrupt enable bit SCTIE is set to "1".
- This bit is cleared by writing "0" and by any reset.
- This bit should always be cleared when the Sub clock timer is reset
- "1" is always read by a read-modify-write instruction.

[bit4] SCTR: Sub Clock Timer Reset Bit

Bit	Description	
	Read	Write
0	Always reads 1	Reset all bits to 0
1		No effect

- This bit clears all bits of the Sub clock counter.

Note:

When clearing the Sub clock timer by writing "0" to this bit, also clear the Sub clock timer interrupt flag by writing "0" to the SCTIF bit.

[bit3] -: Undefined

- Write always "0".
- Read value is undefined.
- RMW instructions are not affected.

[bit2 to bit0] SCTI2 to SCTI0: Sub Clock Timer Interrupt Interval Select Bits

- These bits control the Sub clock timer interrupt interval according to the following table:

bit2	bit1	bit0	Description
0	0	0	2^{10} / CLKSC (31.25 ms)
0	0	1	2^{11} / CLKSC (62.5 ms)
0	1	0	2^{12} / CLKSC (125 ms)
0	1	1	2^{13} / CLKSC (250 ms)
1	0	0	2^{14} / CLKSC (500 ms)
1	0	1	2^{15} / CLKSC (1 s)
1	1	0	2^{16} / CLKSC (2 s)
1	1	1	2^{17} / CLKSC (4 s)

- These bits are initialized to "000" by any reset.
- When data is written to these bits, bit 5 (SCTIF) should be cleared at the same time.

CHAPTER: WATCHDOG TIMER AND WATCHDOG RESET

This chapter explains the functions and operations of the Watchdog Timer and Reset.

1. Overview
2. Operation
3. Registers

Management Code: 96F6HWDG-E01.0

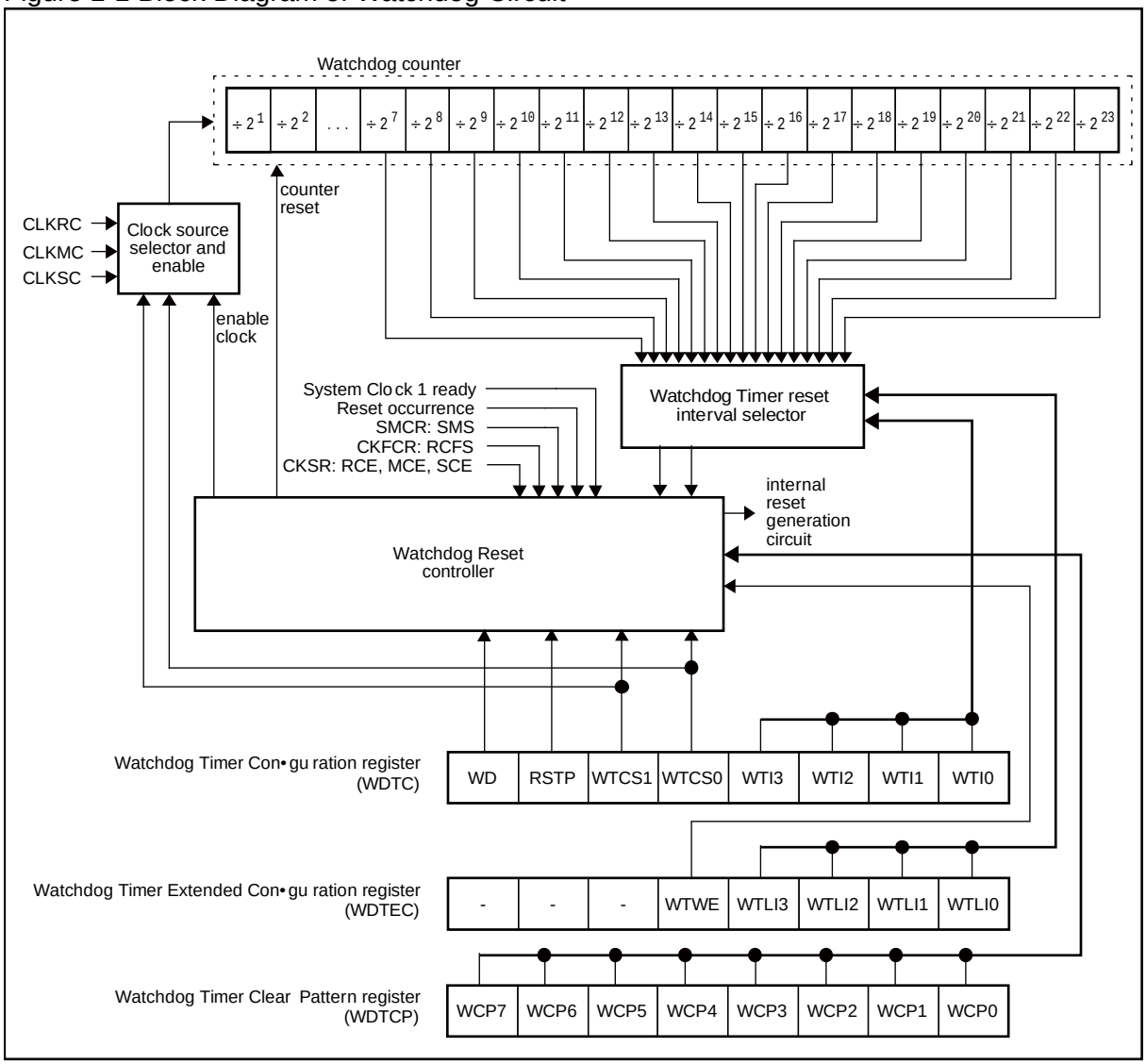
1. Overview

The watchdog circuit consists of a 23-bit watchdog counter, control registers and the Watchdog Reset controller. The 23-bit watchdog counter uses either the RC clock, the Main clock or the Sub clock as clock source.

■ Block Diagram of Watchdog Circuit

Figure 1-1 shows a block diagram of the watchdog circuit.

Figure 1-1 Block Diagram of Watchdog Circuit



2. Operation

The Watchdog Timer and Reset function can be used to detect a hang-up of the user program. If the Watchdog counter is not cleared within the specified time due to, for example, a program hang-up, the Watchdog timer resets the system.

■ Setting the default interval time of the Watchdog Timer

The Watchdog Timer is active at reset release (any reset type) with a Watchdog interval of 2^{12}/RC clock cycles (WDTC:WTI[3:0] = "0100"). The Watchdog Timer starts running when the bus clock CLKB is started.

Then the Boot-ROM program copies the content of the WICM marker in the ROM Configuration Block A to the WDTC:WTI[3:0] register bits without resetting the Watchdog Timer. This setting is then used as "default interval time".

Later the User program can change the Watchdog Timer configuration only once with the first access to the Watchdog Timer. This means the final Watchdog Timer configuration must be selected and activated within the Watchdog default interval time.

For safety the default Watchdog Timer interval time defined by the WICM marker should be selected as short as possible and the final configuration of the Watchdog Timer should be done as early as possible. However if the Main or Sub clock should be selected as clock source for the Watchdog Timer, then the default Watchdog Timer interval time must be at least longer than the oscillator stabilization time (the Watchdog Timer clock source cannot be set to Main or Sub clock before the oscillator is stabilized).

■ Locking and Unlocking of Watchdog Timer

The Watchdog Timer is activated by default using the initial values described in section "3. Registers". Only the Watchdog default interval time can be selected by the WICM marker in the ROM Configuration Block A.

If the first access to the Watchdog Timer registers by the User program follows the sequence given in Figure 2-1 the Watchdog Timer is unlocked. Any access to the Watchdog Timer different from the unlock sequence locks the initial settings. The two write accesses for the unlocking sequence are not affecting the actual Watchdog Timer registers.

The configuration of the unlocked Watchdog Timer can be changed while the Watchdog remains active with the default configuration. If the Watchdog Timer is unlocked it also can be disabled.

A read access to the configuration registers returns the default configuration values (which are used by the Watchdog) until the new configuration data is activated.

Note:

The WDTC and WDTEC register must be configured by Byte or Word write commands. Setting the different bits sequentially by using Set or Clear Bit(s) commands cannot work, because the invoked Read-Modify-Write operations always read the same default configuration values.

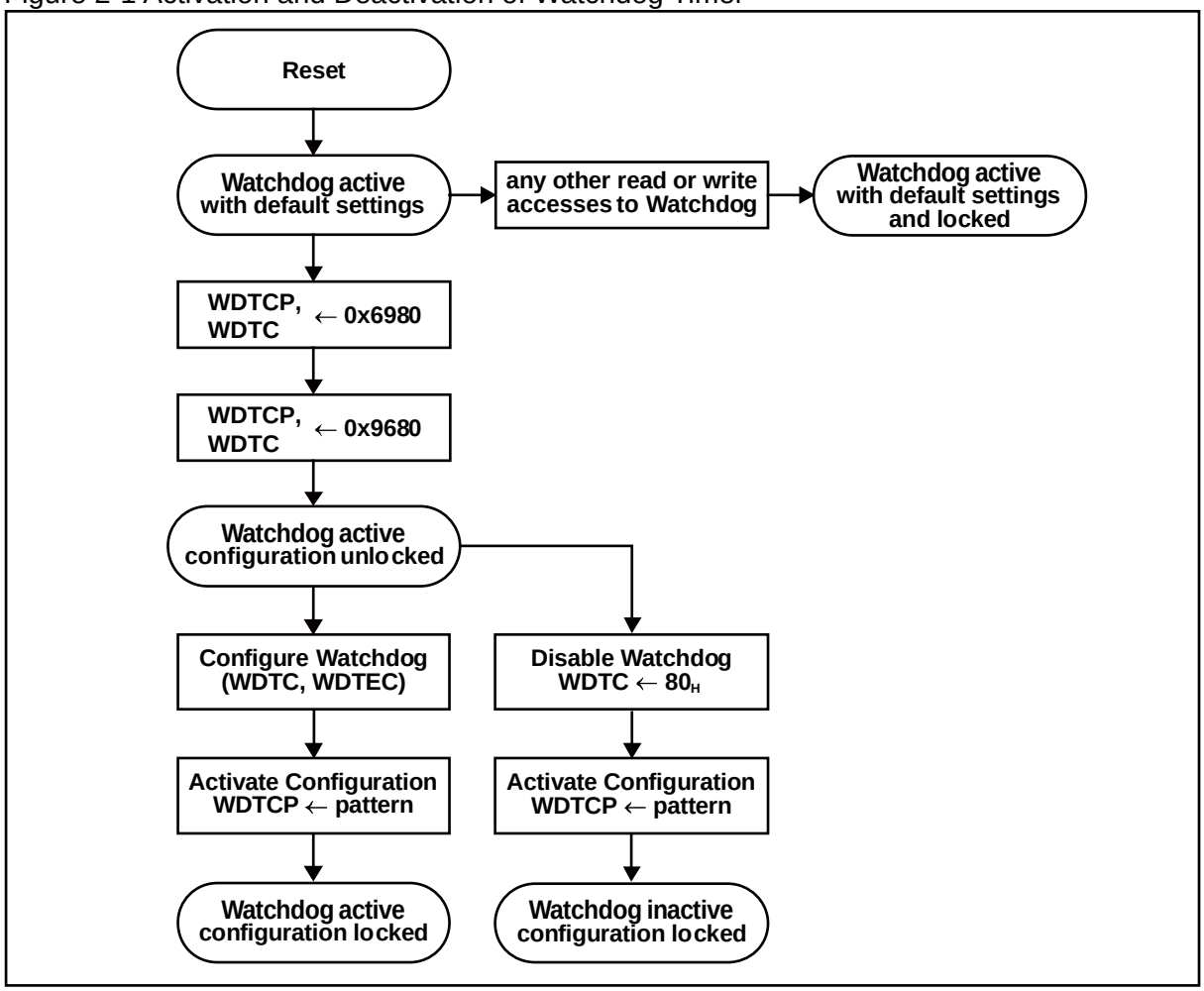
After the configuration of the unlocked Watchdog Timer has been changed the settings are activated by writing to the Watchdog Timer Clear Pattern register WDTCP. The Watchdog Timer operation then depends on the configuration written to the Watchdog Timer Configuration register WDTC and the Watchdog Timer Extended Configuration register WDTEC before. The write access to WDTC and WDTCP can be done by a single 16-bit access.

Once the Watchdog Timer is locked it is only possible to unlock it again after a reset (any type of reset).

■ Disabling of Watchdog Timer

The Watchdog Timer is activated by default. It can be disabled by the application, if the first access to the Watchdog Timer registers follows the sequence given in Figure 2-1. With this sequence the Watchdog Timer configuration is unlocked and the Watchdog Timer can be disabled by setting the WDTC:WD bit. To activate the settings and disable the Watchdog Timer then the Watchdog Timer Clear Pattern register WDTCP has to be written.

Figure 2-1 Activation and Deactivation of Watchdog Timer



■ Clearing the Watchdog Counter

The Watchdog counter can only be cleared by writing the appropriate data to the WDTCP register. This data depends on the Watchdog operation mode.
If the window Watchdog operation is disabled (WDTEC:WTWE = 0) and the counter is not cleared before the interval selected by the WDTC:WTI[3:0] bits expires, then a Watchdog reset is asserted.
If the window Watchdog operation is enabled (WDTEC:WTWE = 1) the Watchdog Timer must be cleared in the window between WDTEC:WTLI[3:0] and WDTC:WTI[3:0]. Clearing the timer before it reaches the lower limit defined by WDTEC:WTLI[3:0] asserts a Watchdog reset. Not clearing the timer before it reaches the upper limit defined by WDTC:WTI[3:0] causes a Watchdog reset as well.

Note:

If the selected lower limit is equal or greater than the selected upper limit and the window Watchdog operation is enabled (WDTEC:WTWE = 1) the Watchdog Timer can not be cleared within the selected window. Either the attempt to clear the Watchdog Timer or not clearing the Watchdog Timer will lead to a Watchdog reset.

■ Stopping the Watchdog Counter

The Watchdog counter can only be stopped (not cleared) when the MCU goes to Stop mode.

● Transition to Stop Mode

All oscillators are disabled in Stop mode, hence the Watchdog counter is also stopped (not cleared).

● Release of Stop Mode

Upon wakeup by interrupt, the watchdog counter starts running when the clock selected as source for the watchdog is stabilized and the CPU clock CLKB is reactivated (clock ready flags set of clocks selected by WDTC:WTCS[1:0] and CKSR:SC1S[1:0] select bit).

Note:

Do not use the Stop mode if the Watchdog must always be active. It is recommended to use the RC Timer mode instead as the mode with the smallest current consumption while disabling the Main and Sub oscillator and running the watchdog with the RC clock. Set WDTC:RSTP to "1" to force a Watchdog reset in case of an accidental transition to Stop mode.

■ Selecting the Clock Source for the Watchdog Counter

The Watchdog counter can operate with the RC clock (CLKRC), the Main clock (CLKMC) or the Sub clock (CLKSC) as clock source. Select a clock source depending on the requirements of the system by setting the WDTC:WTCS[1:0] bits. After locking of the Watchdog Timer it is not possible to change the clock source any more.
When the RC clock should be used as clock source and the system requires changing the RC clock frequency during operation, then set the WDTC:WTCS[1:0] bits to "00_B". This is the initial setting after reset release. Be aware that changing the RC clock frequency also changes the Watchdog Timer interval.
When the RC clock should be used as clock source but changing the RC clock frequency during operation is not intended, then set the WDTC:WTCS[1:0] bits to "01_B". In case of an accidental change of the RC clock frequency, a Watchdog reset will be asserted.
Setting the clock source to Main clock or Sub clock is only possible when this clock is stabilized (CKMR:MCM or CKMR:SCM is set). Changing the WDTC:WTCS bits to Main or Sub clock and activating this new setting by a write access to the WDTCP register before the corresponding clock monitor bit is set will cause a Watchdog reset.

Caution:

The Watchdog counter can only operate if the selected clock is stabilized (CKMR:RCM, MCM or SCM bit is "1"). It is not possible to switch the clock source from RC clock to Main clock or Sub clock as long as this clock is not stabilized (a Watchdog reset will be generated when attempting this).
However when the Stop mode is used, then the Watchdog is stopped during Stop mode assertion and during the clock stabilization time after Stop mode release. In case the Watchdog is running on the Main or Sub clock and this clock fails at Stop mode release, then neither a Watchdog nor a Clock stop reset will be asserted.
Hence it is strongly recommended to restart in RC Run mode after wakeup from Stop mode and detect the failing Main or Sub clock by software, for example by an RC Timer interrupt (see also "■ Clock stop detection reset and Standby modes" in "CHAPTER 7 RESETS AND STARTUP").

■ Standby Modes

In Sleep and Timer mode the Watchdog Timer operates according to its configuration and is neither cleared nor stopped. In Stop mode the Watchdog Timer is stopped but not cleared. After Stop mode release, the Watchdog timer resumes operation when both, the Watchdog timer clock source and the clock source for CLKS1 are stabilized.

■ Disabling the Source Clock

Setting the clock enable bit (CKSR:RCE, MCE, SCE) of the oscillator used as clock source for the Watchdog counter to "0" (disable oscillator) causes a Watchdog reset.

■ Setting the Watchdog Timer Interval

The Watchdog Timer interval can be selected with the WDTC:WTI[3:0] and WDTEC:WTLI[3:0] bits. After activation of the Watchdog Timer it is not possible to change the interval any more.
When using the RC oscillator as clock source, consider the frequency variation of this oscillator when calculating the interval length. The absolute times in Watchdog Timer Configuration register (WDTC) are based on the nominal oscillation frequency. Please see the datasheet for more details regarding the accuracy of the RC clock frequency.

■ Selecting the Operation Mode of the Watchdog Timer

The Watchdog Timer is activated by any reset in a default configuration. Any access to the Watchdog Timer registers different from unlocking as described in Figure 2-1 locks this configuration. In this default configuration the clear by writing "00_H" mode is selected.
The first write access to the WDTCP register after unlocking the Watchdog Timer as described in Figure 2-1 activates it and defines the operation mode. Depending on the data written to WDTCP at this first write access, one of the following two operation modes is selected:

● Clear by writing "00_H" and Assert Watchdog Reset when writing Other Values

This mode is activated by default after reset. It can also be selected by writing "00_H" to the WDTCP register when activating the Watchdog Timer as described in Figure 2-1.
In this mode, the Watchdog counter is cleared by writing "00_H" to the WDTCP register. Writing values other than 00_H causes a Watchdog reset. If the window function is enabled (WDTEC:WTWE = 1) any write access to the WDTCP register outside the clear window causes a Watchdog reset.

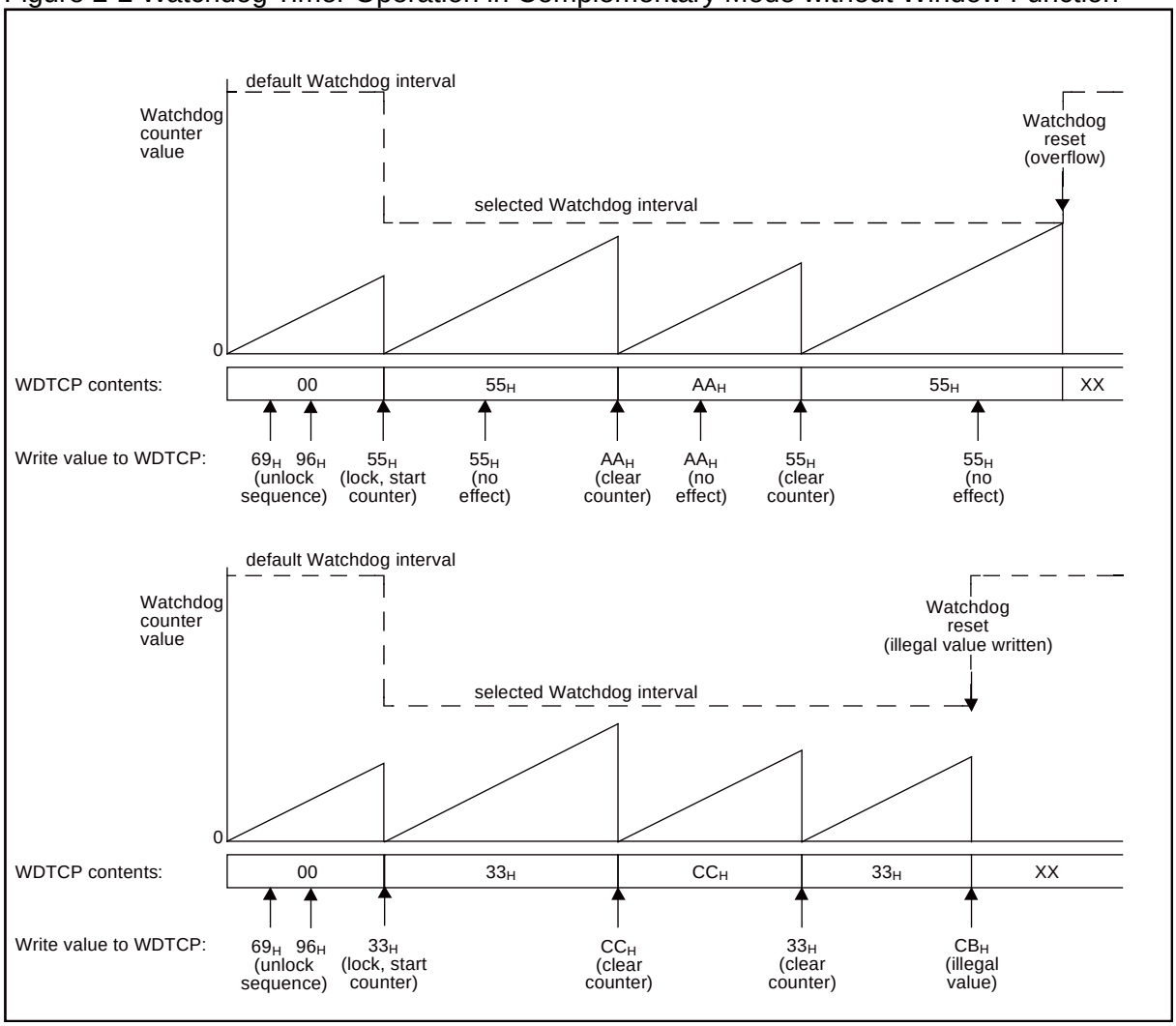
● Clear by writing Complementary Data and Assert Watchdog Reset when writing Invalid Data

This mode is selected by writing any value except "00_H" to the WDTCP register when activating the Watchdog Timer as described in Figure 2-1.
The data written at this write access is stored in the WDTCP register.
If the window function is disabled (WDTEC:WTWE = 0) the following applies:

- Writing the same data as currently stored in the WDTCP register has no effect.
- Writing the complementary data of the current WDTCP contents clears the Watchdog counter and replaces the current value in the WDTCP register. To clear the Watchdog Timer the next time, the write data must be complemented again every time (original data - complementary data - original data - complementary data...)
- Writing any data to the WDTCP register that differs from the current register contents or the complementary value causes a Watchdog reset.

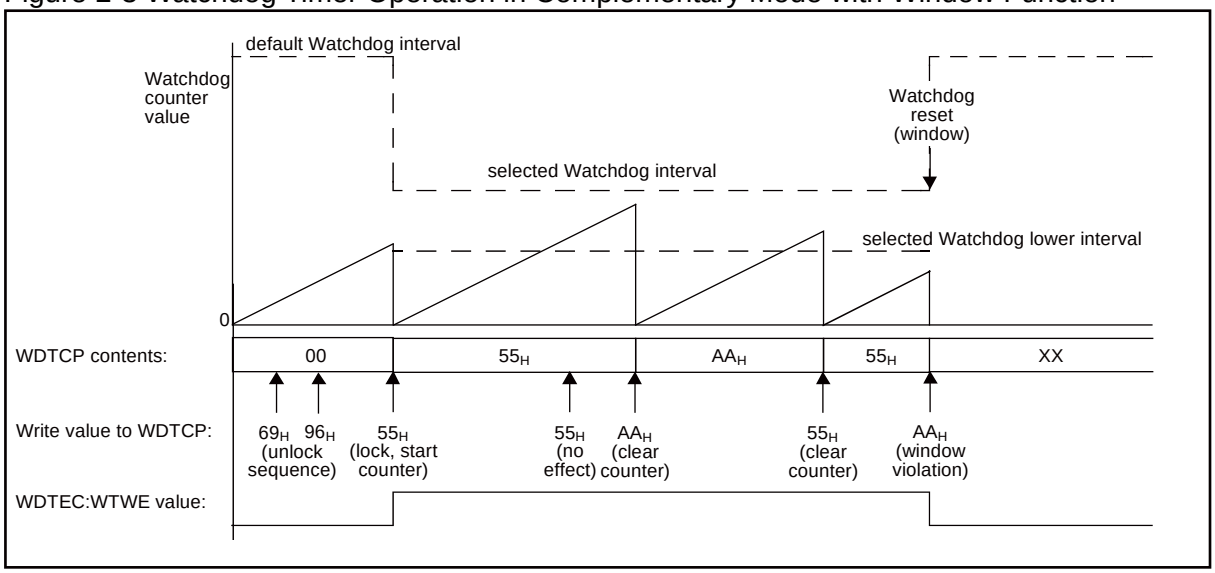
The following figure shows an example of the Watchdog Timer operation in this mode.

Figure 2-2 Watchdog Timer Operation in Complementary Mode without Window Function



If the window function is enabled (WDTEC:WTWE = 1) the same as above applies for write accesses within the clear window. All write accesses to the WDTCP register outside the selected clear window cause a Watchdog reset.
The following figure shows an example of the Watchdog Timer operation in this mode.

Figure 2-3 Watchdog Timer Operation in Complementary Mode with Window Function



■ Watchdog Timer Reset Causes

A Watchdog Timer reset can be asserted in the following cases:

- The Watchdog Timer was not cleared within the selected interval.
- Invalid data was written to the WDTCP register.
- "0" was written to the enable bit of the clock which is used as source clock for the Watchdog Timer (CKSR:RCE, MCE or SCE).
- The RC clock frequency was changed although WDTC:WTCS was set to "01".
- Transition to Stop mode was requested although WDTC:RSTP was set to "1".
- The Watchdog Timer clock source was set to Main clock and this setting was activated before CKMR:MCM was set.
- The Watchdog Timer clock source was set to Sub clock and this setting was activated before CKMR:SCM was set.
- The Watchdog Timer interval WDTC:WTI[3:0] is changed to a value that selects a shorter interval time than the current Watchdog counter value.

3. Registers

This section explains the configuration and functions of the registers used for the watchdog timer control.

■ List of Watchdog Timer Control Registers

Abbreviated Register Name	Register Name	Reference
WDTC	Watchdog Timer Configuration register	See 3.1
WDTEC	Watchdog Timer Extended Configuration register	See 3.2
WDTCP	Watchdog Timer Clear Pattern register	See 3.3

3.1. Watchdog Timer Configuration Register (WDTC)

The Watchdog Timer Configuration register (WDTC) is used to select the clock source and the watchdog interval and to activate some special functions of the Watchdog Timer.

■ Configuration of the Watchdog Timer Configuration Register (WDTC)

WDTC								
bit	7	6	5	4	3	2	1	0
	WD	RSTP	WTCS1	WTCS0	WTI3	WTI2	WTI1	WTI0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	*	*	*	*

*: Initial value depends on WICM Marker

[bit7] WD: Watchdog Disable

Bit	Description
0	Watchdog is enabled and active
1	Watchdog is disabled

- This bit is initialized to "0" by any reset.
- It is possible to unlock the Watchdog Timer configuration once after reset by writing the sequence described in section "2. Operation" to its WDTC and WDTCP registers. Any other access to the Watchdog Timer registers finally locks the Watchdog Timer configuration until reset.
- If the Watchdog Timer configuration is unlocked this bit can be changed.
- The setting of this bit is activated and locked when writing to WDTCP.
- Reconfiguring this bit and locking the Watchdog Reset function with one 16-bit access is possible if the Watchdog Timer configuration is unlocked.
- After a reset the Watchdog Timer configuration can be unlocked again and this bit can be changed.
- When reading this bit it returns the status of the Watchdog. The bit reads as "0" until this bit is set to "1" and this new setting is activated.

[bit6] RSTP: Watchdog Reset at Transition to Stop Mode

Bit	Description
0	No Watchdog Reset asserted at transition to Stop mode
1	Watchdog Reset asserted at transition to Stop mode

- This bit is initialized to "0" by any reset.
- It is possible to unlock the Watchdog Timer configuration once after reset by writing the sequence described in section "2. Operation" to its WDTC and WDTCP registers. Any other access to the Watchdog Timer registers finally locks the Watchdog Timer configuration until reset.
- If the Watchdog Timer configuration is unlocked this bit can be changed.
- The setting of this bit is activated and locked when writing to WDTCP.
- Reconfiguring this bit and locking the Watchdog Reset function with one 16-bit access is possible if the Watchdog Timer configuration is unlocked.
- After a reset the Watchdog Timer configuration can be unlocked again and this bit can be changed.
- When this bit is set to "0", then transition to Stop mode is allowed
- When this bit is set to "1", then writing "11" to SMCR:SMS (Stop mode) causes a Watchdog reset.
- Set this bit to "1" only when the application is not using the Stop mode.
- Reading this bit returns the value which is currently used by the Watchdog. If a new value is written after unlocking, then this new value cannot be read until the new setting is activated.

[bit5, bit4] WTCS1 to WTCS0: Watchdog Timer Clock Selection Bits

- These bits select the clock source for the Watchdog Timer according to the following table:

bit5	bit4	Description
0	0	CLKWT = CLKRC (internal RC oscillation clock, RC clock frequency can be changed during Watchdog operation)
0	1	CLKWT = CLKRC (internal RC oscillation clock, changing RC clock frequency during Watchdog operation causes Watchdog reset)
1	0	CLKWT = CLKMC (Main oscillation clock)
1	1	CLKWT = CLKSC (Sub oscillation clock)

- These bits are initialized to "00" by any reset.
- It is possible to unlock the Watchdog Timer configuration once after reset by writing the sequence described in section "2. Operation" to its WDTC and WDTCP registers. Any other access to the Watchdog Timer registers finally locks the Watchdog Timer configuration until reset.
- If the Watchdog Timer configuration is unlocked these bits can be changed.
- The setting of these bits is activated and locked when writing to WDTCP.
- Reconfiguring these bits and locking the Watchdog Reset function with one 16-bit access is possible if the Watchdog Timer configuration is unlocked.
- After a reset the Watchdog Timer configuration can be unlocked again and these bits can be changed.
- Setting WTCS[1:0] to "00" selects the internal RC oscillator and it is possible to change the RC clock frequency (by writing to CKFCR:RCFS bit) after activation of the Watchdog Reset function. Please note that changing the RC clock frequency also changes the Watchdog interval time.
- Setting WTCS[1:0] to "01" also selects the internal RC oscillator, but it is not allowed to change the RC clock frequency (by writing to CKFCR:RCFS bit) after activation of the Watchdog Reset function. Changing the RC clock frequency causes a Watchdog reset.
- Setting WTCS[1:0] to "10" selects the Main oscillator. The Main oscillator must be stabilized (CKMR:MCM must be "1") before this setting is activated by the write access to WDTCP. If this setting is activated while CKMR:MCM is "0", then a Watchdog reset will be asserted.
- Setting WTCS[1:0] to "11" selects the Sub oscillator. The Sub oscillator must be stabilized (CKMR:SCM must be "1") before this setting is activated by the write access to WDTCP. If this setting is activated while CKMR:SCM is "0", then a Watchdog reset will be asserted.
- Reading these bits returns the values which are currently used by the Watchdog. If a new value is written after unlocking, then this new value cannot be read until the new setting is activated.

[bit3 to bit0] WTI3 to WTI0: Watchdog Timer Interval Select

bit3	bit2	bit1	bit0	Description (CLKWT is clock selected by WTCS[1:0] bits)
0	0	0	0	2^8 / CLKWT
0	0	0	1	2^9 / CLKWT
0	0	1	0	2^{10} / CLKWT
0	0	1	1	2^{11} / CLKWT
0	1	0	0	2^{12} / CLKWT
0	1	0	1	2^{13} / CLKWT
0	1	1	0	2^{14} / CLKWT
0	1	1	1	2^{15} / CLKWT
1	0	0	0	2^{16} / CLKWT
1	0	0	1	2^{17} / CLKWT
1	0	1	0	2^{18} / CLKWT
1	0	1	1	2^{19} / CLKWT
1	1	0	0	2^{20} / CLKWT
1	1	0	1	2^{21} / CLKWT
1	1	1	0	2^{22} / CLKWT
1	1	1	1	2^{23} / CLKWT

- These bits select the Watchdog Timer Interval according to the following table:

WTI3	WTI2	WTI1	WTI0	Watchdog Timer Interval depending on selected clock source		
				RC clock selected (corresponding time for nominal RC clock frequency of 2MHz/100kHz)	Main clock selected (corresponding time for Main clock frequency of 4MHz)	Sub clock selected (corresponding time for Sub clock frequency of 32.768kHz)
0	0	0	0	2^8 / CLKRC (~128µs/2.5ms)	2^8 / CLKMC (~64µs)	2^8 / CLKSC (7.8ms)
0	0	0	1	2^9 / CLKRC (~256µs/5.1ms)	2^9 / CLKMC (~128µs)	2^9 / CLKSC (15.6ms)
0	0	1	0	2^{10} / CLKRC (~512µs/10.2ms)	2^{10} / CLKMC (~256µs)	2^{10} / CLKSC (31.25ms)
0	0	1	1	2^{11} / CLKRC (~1ms/20.5ms)	2^{11} / CLKMC (~512µs)	2^{11} / CLKSC (62.5ms)
0	1	0	0	2^{12} / CLKRC (~2ms/41ms)	2^{12} / CLKMC (~1ms)	2^{12} / CLKSC (125ms)
0	1	0	1	2^{13} / CLKRC (~4ms/82ms)	2^{13} / CLKMC (~2ms)	2^{13} / CLKSC (250ms)
0	1	1	0	2^{14} / CLKRC (~8ms/164ms)	2^{14} / CLKMC (~4ms)	2^{14} / CLKSC (500ms)
0	1	1	1	2^{15} / CLKRC (~16ms/328ms)	2^{15} / CLKMC (~8ms)	2^{15} / CLKSC (1s)
1	0	0	0	2^{16} / CLKRC (~32ms/655ms)	2^{16} / CLKMC (~16ms)	2^{16} / CLKSC (2s)
1	0	0	1	2^{17} / CLKRC (~65ms/1.3s)	2^{17} / CLKMC (~32ms)	2^{17} / CLKSC (4s)
1	0	1	0	2^{18} / CLKRC (~131ms/2.6s)	2^{18} / CLKMC (~65ms)	2^{18} / CLKSC (8s)
1	0	1	1	2^{19} / CLKRC (~262ms/5.2s)	2^{19} / CLKMC (~131ms)	2^{19} / CLKSC (16s)
1	1	0	0	2^{20} / CLKRC (~524ms/10.4s)	2^{20} / CLKMC (~262ms)	2^{20} / CLKSC (32s)
1	1	0	1	2^{21} / CLKRC (~1.05s/21s)	2^{21} / CLKMC (~524ms)	2^{21} / CLKSC (64s)
1	1	1	0	2^{22} / CLKRC (~2.1s/42s)	2^{22} / CLKMC (~1.049s)	2^{22} / CLKSC (128s)
1	1	1	1	2^{23} / CLKRC (~4.2s/84s)	2^{23} / CLKMC (~2.097s)	2^{23} / CLKSC (256s)

- These bits are initialized to "0100" by any reset. After reset release, the Boot ROM program copies the content of the WICM marker in the ROM Configuration Block A to these bits. This is done before the initial time of 2^{12}/RC clock cycles is elapsed. The new setting is then locked and can only be changed once by unlocking the register as described below.
- It is possible to unlock the Watchdog Timer configuration once after reset by writing the sequence described in section "2. Operation" to its WDTC and WDTCP registers. Any other access to the Watchdog Timer registers finally locks the Watchdog Timer configuration until reset.
- If the Watchdog Timer configuration is unlocked these bits can be changed.
- The setting of these bits is activated and locked when writing to WDTCP.
- Reconfiguring these bits and locking the Watchdog Reset function with one 16-bit access is possible if the Watchdog Timer configuration is unlocked.
- After a reset the Watchdog Timer configuration can be unlocked again and these bits can be changed.
- Reading these bits returns the values which are currently used by the Watchdog. If a new value is written after unlocking, then this new value cannot be read until the new setting is activated.

3.2. Watchdog Timer Extended Configuration Register (WDTEC)

The Watchdog Timer Extended Configuration register (WDTEC) is used to enable the window functionallity of the Watchdog Timer and to select the lower window limit of the watchdog interval.

■ Configuration of the Watchdog Timer Extended Configuration Register (WDTEC)

WDTEC								
bit	7	6	5	4	3	2	1	0
	-	-	-	WTWE	WTLI3	WTLI2	WTLI1	WTLI0
Attribute	-	-	-	R/W	R/W	R/W	R/W	R/W
Initial value	X	X	X	0	0	0	0	0

[bit7 to bit 5] -: Undefined

- Always write "0" to these bits.
- The read value of these bits is undefined.
- Read modify write operations to this register have no effect on these bits.

[bit4] WTWE: Watchdog Timer Window Enable

Bit	Description
0	Watchdog Timer window function disabled
1	Watchdog Timer window function enabled

- This bit is initialized to "0" by any reset.
- It is possible to unlock the Watchdog Timer configuration once after reset by writing the sequence described in section "2. Operation" to its WDTC and WDTCP registers. Any other access to the Watchdog Timer registers finally locks the Watchdog Timer configuration until reset.
- If the Watchdog Timer configuration is unlocked this bit can be changed.
- The setting of this bit is activated and locked when writing to WDTCP.
- After a reset the Watchdog Timer configuration can be unlocked again and this bit can be changed.
- When this bit is set to "0", the Watchdog Timer can be cleared anytime before reaching the value defined by WDTC:WTI[3:0].
- When this bit is set to "1", the Watchdog Timer can only be cleared after the value of WDTEC:WTLI[3:0] has been reached and before WDTC:WTI[3:0] is reached. If it is cleared outside the window defined by WDTEC:WTLI[3:0] and WDTC:WTI[3:0] a Watchdog reset is generated.
- Reading these bits returns the value which is currently used by the Watchdog. If a new value is written after unlocking, then this new value cannot be read until the new setting is activated.

[bit3 to bit0] WTLI3 to WTLI0: Watchdog Timer Lower Interval Select

bit3	bit2	bit1	bit0	Description (CLKWT is clock selected by WDTC:WTCS[1:0] bits)
0	0	0	0	2^7 / CLKWT
0	0	0	1	2^8 / CLKWT
0	0	1	0	2^9 / CLKWT
0	0	1	1	2^{10} / CLKWT
0	1	0	0	2^{11} / CLKWT
0	1	0	1	2^{12} / CLKWT
0	1	1	0	2^{13} / CLKWT
0	1	1	1	2^{14} / CLKWT
1	0	0	0	2^{15} / CLKWT
1	0	0	1	2^{16} / CLKWT
1	0	1	0	2^{17} / CLKWT
1	0	1	1	2^{18} / CLKWT
1	1	0	0	2^{19} / CLKWT
1	1	0	1	2^{20} / CLKWT
1	1	1	0	2^{21} / CLKWT
1	1	1	1	2^{22} / CLKWT

· These bits select the Watchdog Timer Lower Interval according to the following table:

WTLI 3	WTLI2	WTLI1	WTLI0	Watchdog Timer Lower Interval depending on selected clock source		
				RC clock selected (corresponding time for nominal RC clock frequency of 2MHz/100kHz)	Main clock selected (corresponding time for Main clock frequency of 4MHz)	Sub clock selected (corresponding time for Sub clock frequency of 32.768kHz)
0	0	0	0	2^7 / CLKRC (~64µs/1.25ms)	2^7 / CLKMC (~32µs)	2^7 / CLKSC (3.9ms)
0	0	0	1	2^8 / CLKRC (~128µs/2.5ms)	2^8 / CLKMC (~64µs)	2^8 / CLKSC (7.8ms)
0	0	1	0	2^9 / CLKRC (~256µs/5.1ms)	2^9 / CLKMC (~128µs)	2^9 / CLKSC (15.6ms)
0	0	1	1	2^{10} / CLKRC (~512µs/10.2ms)	2^{10} / CLKMC (~256µs)	2^{10} / CLKSC (31.25ms)
0	1	0	0	2^{11} / CLKRC (~1ms/20.5ms)	2^{11} / CLKMC (~512µs)	2^{11} / CLKSC (62.5ms)
0	1	0	1	2^{12} / CLKRC (~2ms/41ms)	2^{12} / CLKMC (~1ms)	2^{12} / CLKSC (125ms)
0	1	1	0	2^{13} / CLKRC (~4ms/82ms)	2^{13} / CLKMC (~2ms)	2^{13} / CLKSC (250ms)
0	1	1	1	2^{14} / CLKRC (~8ms/164ms)	2^{14} / CLKMC (~4ms)	2^{14} / CLKSC (500ms)
1	0	0	0	2^{15} / CLKRC (~16ms/328ms)	2^{15} / CLKMC (~8ms)	2^{15} / CLKSC (1s)
1	0	0	1	2^{16} / CLKRC (~32ms/655ms)	2^{16} / CLKMC (~16ms)	2^{16} / CLKSC (2s)
1	0	1	0	2^{17} / CLKRC (~65ms/1.3s)	2^{17} / CLKMC (~32ms)	2^{17} / CLKSC (4s)
1	0	1	1	2^{18} / CLKRC (~131ms/2.6s)	2^{18} / CLKMC (~65ms)	2^{18} / CLKSC (8s)
1	1	0	0	2^{19} / CLKRC (~262ms/5.2s)	2^{19} / CLKMC (~131ms)	2^{19} / CLKSC (16s)
1	1	0	1	2^{20} / CLKRC (~524ms/10.4s)	2^{20} / CLKMC (~262ms)	2^{20} / CLKSC (32s)
1	1	1	0	2^{21} / CLKRC (~1.05s/21s)	2^{21} / CLKMC (~524ms)	2^{21} / CLKSC (64s)
1	1	1	1	2^{22} / CLKRC (~2.1s/42s)	2^{22} / CLKMC (~1.049s)	2^{22} / CLKSC (128s)

· These bits are initialized to "0000" by any reset.
· It is possible to unlock the Watchdog Timer configuration once after reset by writing the sequence described in section "2. Operation" to its WDTC and WDTCP registers. Any other access to the Watchdog Timer registers finally locks the Watchdog Timer configuration.
· If the Watchdog Timer configuration is unlocked these bits can be changed.
· The setting of these bits is activated and locked when writing to WDTCP.
· After a reset the Watchdog Timer configuration can be unlocked again and these bits can be changed.
· Reading these bits returns the values which are currently used by the Watchdog. If a new value is written after unlocking, then this new value cannot be read until the new setting is activated.

3.3. Watchdog Timer Clear Pattern Register (WDTCP)

The Watchdog Timer Clear Pattern register (WDTCP) is used to activate and lock the Watchdog reset function and to clear the Watchdog counter.

■ Configuration of the Watchdog Timer Clear Pattern Register (WDTCP)

WDTCP								
bit	15	14	13	12	11	10	9	8
	WCP7	WCP6	WCP5	WCP4	WCP3	WCP2	WCP1	WCP0
Attribute	W	W	W	W	W	W	W	W
Initial value	0	0	0	0	0	0	0	0

[bit15 to bit8] WCP7 to WCP0: Watchdog Clear Pattern Bits

- The Watchdog reset function is activated after reset by hardware. The clear pattern used in this default configuration is 00_H.
- It is possible to unlock the Watchdog Timer configuration by writing the sequence described in section "2. Operation" to its WDTC and WDTCP registers. Any other access to the Watchdog Timer registers locks the Watchdog Timer configuration.
- If the Watchdog Timer configuration is unlocked a write access to this register activates the configuration in WDTC and WDTEC and locks the Watchdog Timer configuration. The value written to WDTCP determines the clear pattern and operation mode. The Watchdog counter is reset by this write access.
- Writing the WDTC and WDTCP register with one 16-bit access is possible if the Watchdog Timer configuration is unlocked. This configures WDTC, activates the settings, locks the Watchdog Timer configuration and resets the Watchdog counter.
- Once the Watchdog reset function is activated and locked, it can only be disabled by reset (any type of reset).
- After writing to this register, it is no longer possible to change the contents of the Watchdog Timer Configuration register (WDTC) and the Watchdog Timer Extended Configuration register (WDTEC).
- Depending on the value written to the WDTCP register when activating and locking the Watchdog Timer, the Watchdog Timer operates in one of the following Watchdog operation modes:
  1. "Clear by 00_H, assert reset for other values": Write 00_H to WDTCP:
     In this mode, the Watchdog counter is cleared by writing 00_H to the WDTCP register. Writing values other than 00_H causes a Watchdog reset. If the window function is enabled (WDTEC:WTWE = 1) any write access to the WDTCP register outside the clear window causes a Watchdog reset. See section "2. Operation" for more details.
  2. "Clear by complementary data" Write any value except 00_H to WDTCP:
     In this mode, the value written to the WDTCP register at the first time is memorized. Clearing the Watchdog counter is possible only by writing the complementary value what replaces the last register value. Writing the same value as before has no effect and writing other values causes a Watchdog reset. If the window function is enabled (WDTEC:WTWE = 1) any write access to the WDTCP register outside the clear window causes a Watchdog reset. See section "2. Operation" for more details.
- Reading this register always returns 00_H.

CHAPTER: I/O PORTS

This chapter explains the functions and operations of the I/O ports.

Management Code: 96F6IOPORT-E01.0

1. Overview

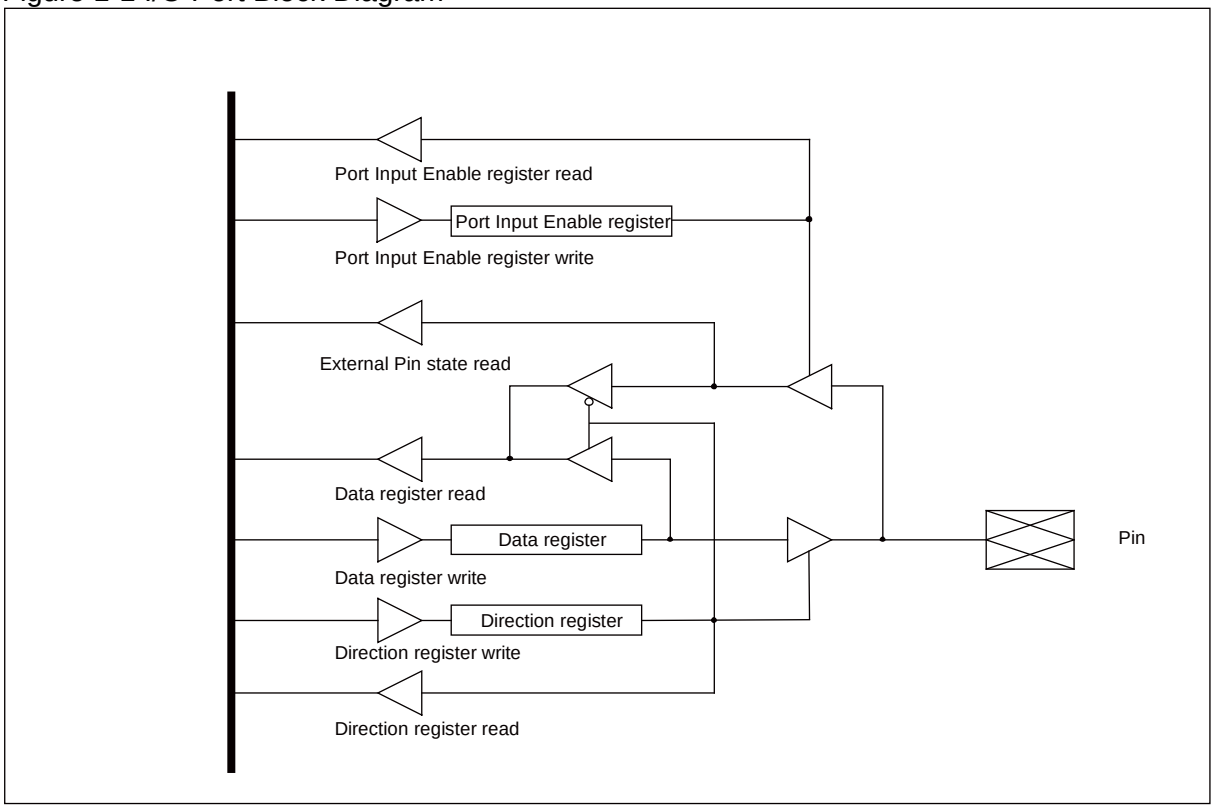
Each pin of the ports can be specified as input or output using the direction register, if the corresponding peripheral does not use the pin. Before using a pin as input, it must be enabled by setting the Port Input Enable register. When a pin is specified as input, the logic level at the pin is read. When a pin is specified as output, the data register value is read.

■ I/O Ports

When a pin is used as an output of other peripheral function, the peripheral output value is read regardless of the direction register value.

It is generally recommended that the read-modify-write instructions should not be used for setting the data register prior to setting the port as an output. This is because the read-modify-write instruction in this case results reading the logic level at the port rather than the register value.

Figure 1-1 I/O Port Block Diagram



2. Registers

Each general purpose port pin GPxx_y is controlled by seven types of registers.

■ I/O Port Registers

The general purpose port pin GPxx_y is controlled by bit y of the I/O port register xx. For example, the data direction of port pin GPxx_y is controlled by DDRxx_y.

■ List of I/O Port Registers

Abbreviated Register Name	Register Name	Reference
PDRnn	Port Data Register	See 2.1
EPSRnn	External Pin State Register	See 2.2
DDRnn	Data Direction Register	See 2.3
PIERnn	Port Input Enable Register	See 2.4
PHDR	Port High Drive Register	See 2.5
PUCR	Pull-Up Control Register	See 2.6
PDCRnn	Pull-Down Control Register	See 2.7

2.1. Port Data Register (PDRnn)

Note that R/W for I/O ports differ from R/W for memory in the following points:
• Input mode
Read: The level at the corresponding pin is read.
Write: Data is written to an output latch.
• Output mode
Read: The data register latch value is read.
Write: Data is written to an output latch and output to the corresponding pin.

■ Port Data Register (PDRnn)

PDRnn								
bit	7	6	5	4	3	2	1	0
	P7	P6	P5	P4	P3	P2	P1	P0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	X	X	X	X	X	X	X	X

[bit7 to bit0] Px (PDR00 to PDRnn)

When a Port Data Register is read, the read value depends on the corresponding bit in the Data Direction Register, on the current status of the resource that is connected to the same pin (if applicable) and the used instruction (RMW or not RMW). The following cases are possible:

Table 2-1 Reading Port Data Register

DDR value	Resource output	Pin value	Read value of "Read instructions"	Read value of "RMW instructions"
0 (input)	enabled	value of resource output	input pin state	value of PDR
	disabled	port input	input pin state	value of PDR
1 (output)	enabled	value of resource output	value of PDR	value of PDR
	disabled	value of PDR	value of PDR	value of PDR

Reading PDR value with RMW instructions ensures, that single bits of PDR are not accidentally overwritten. This may happen if resource function of pin is selected and pin value is read instead of PDR value.

Please refer to the following example for explanation of difference between reading PDR with RMW instruction on F^2MC-16LX and F^2MC-16FX.

Bit #	7654.3210
DDR	1111.1111 (= output)
PDR (= last value written to PDR)	0000.0000
OE (resource output enable)	0010.0000
Resource output	0010.0000
current pin value	0010.0000
read by RMW instruction from PDR on 16LX	0010.0000 -> e.g. MOV A, #0 OR PDR, A => writes back #b'0010.0000 => PDR is changed
read by RMW instruction from PDR on 16FX	0000.0000 -> e.g. MOV A, #0 OR PDR, A => writes back #b'0000.0000 => PDR is not changed

2.2. External Pin State Register (EPSRnn)

With external pin state register the current state of the external pin can be read.

■ External Pin State Register (EPSRnn)

EPSRnn								
bit	7	6	5	4	3	2	1	0
	PS7	PS6	PS5	PS4	PS3	PS2	PS1	PS0
Attribute	R	R	R	R	R	R	R	R
Initial value	X	X	X	X	X	X	X	X

[bit7 to bit0] PSx (EPSR00 to EPSRnn)

These bits mirror the status of the external port pin.
Writing to this register has no effect.
The external pin status register mirrors the status of the external port pin only if the corresponding bit in the Port Input Enable Register is set to "1".

2.3. Data Direction Register (DDRnn)

When a pin is used as a general purpose port, data direction register switches the corresponding pin to input mode or output mode.

■ Data Direction Register (DDRnn)

DDRnn								
bit	7	6	5	4	3	2	1	0
	D7	D6	D5	D4	D3	D2	D1	D0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

[bit7 to bit0] Dx (DDR00 to DDRnn)

These bits set the data direction of each port pin. When a pin is used as a port, the corresponding pin is controlled as described below:

Bit	Description
0	Input mode
1	Output mode

Note:

The Data Direction Register can be read independently from the status of the corresponding resource. However, the value of the DDR influences the result of a read access on the Port Data Register.
If pins are used as analog input or output, this value is ignored.

2.4. Port Input Enable Register (PIERnn)

When a pin is used as general purpose port, the digital input is enabled or disabled with port input enable register. If the input function of a pin is disabled in this register, the digital input function of the pin cannot be used. When the digital input is disabled, no transverse current is drawn by the input stage at any input pin potential.

■ Port Input Enable Register (PIERnn)

PIERnn								
bit	7	6	5	4	3	2	1	0
	IE7	IE6	IE5	IE4	IE3	IE2	IE1	IE0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

[bit7 to bit0] IEx (PIER00 to PIERnn)

These bits enable or disable the digital input of each port pin.

When a pin is used as an input port, the corresponding pin is controlled as described below:

Bit	Description
0	Disable digtal input
1	Enable digital input

Notes:

- If pins are used as analog input or output, then the digital input function is disabled when the analog function is enabled for pins.
- When the MCU enters Stop mode or Timer mode, the input of all ports is disabled irrespective of PIER setting, except for ports, for which an external interrupt is activated.

2.5. Port High Drive Register (PHDRnn)

The high drive option can be enabled or disabled with this register.
The Port High Drive Register is not available on all devices. Please refer to the datasheet of the corresponding device.

■ Port High Drive Register (PHDRnn)

PHDRnn								
bit	7	6	5	4	3	2	1	0
	HD7	HD6	HD5	HD4	HD3	HD2	HD1	HD0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

[bit7 to bit0] HDx (PHDR00 to PHDRnn)

These bits enable or disable the high drive option for each port pin.
Please refer to the data sheet for electrical characteristics.

Bit	Description
0	Disable high drive option
1	Enable high drive option

Note:

High drive option is not available for all ports. Please refer to the datasheet for details.

2.6. Pull-Up Control Register (PUCRnn)

The Pull-Up control resistor can be enabled or disabled with this register.

■ Pull-Up Control Register (PUCRnn)

PUCRnn								
bit	7	6	5	4	3	2	1	0
	PU7	PU6	PU5	PU4	PU3	PU2	PU1	PU0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

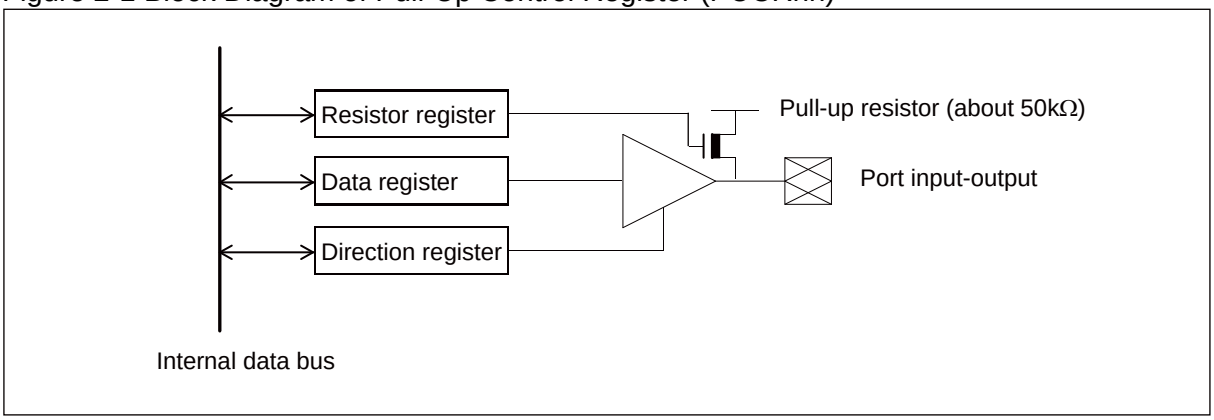
[bit7 to bit0] PUx (PUCR00 to PUCRnn)

These bits enable or disable the pull-up resistor for each port pin.

Bit	Description
0	No Pull-up resistor in input mode
1	Pull-up resistor in input mode

■ Block Diagram of Pull-Up Control Register (PUCRnn)

Figure 2-1 Block Diagram of Pull-Up Control Register (PUCRnn)



Note:

In output mode, the pull-up control register has no effect (no pull-up resistor). In stop and timer mode (SMCR:SPL=1), the state with no pull-up resistor is entered (high impedance).

2.7. Pull-Down Control Register (PDCRnn)

The Pull-Down resistor can be enabled or disabled with this register.
The Pull-Down Control Register is not available in all devices. Please refer to the datasheet of the corresponding device.

■ Pull-Down Control Register (PDCRnn)

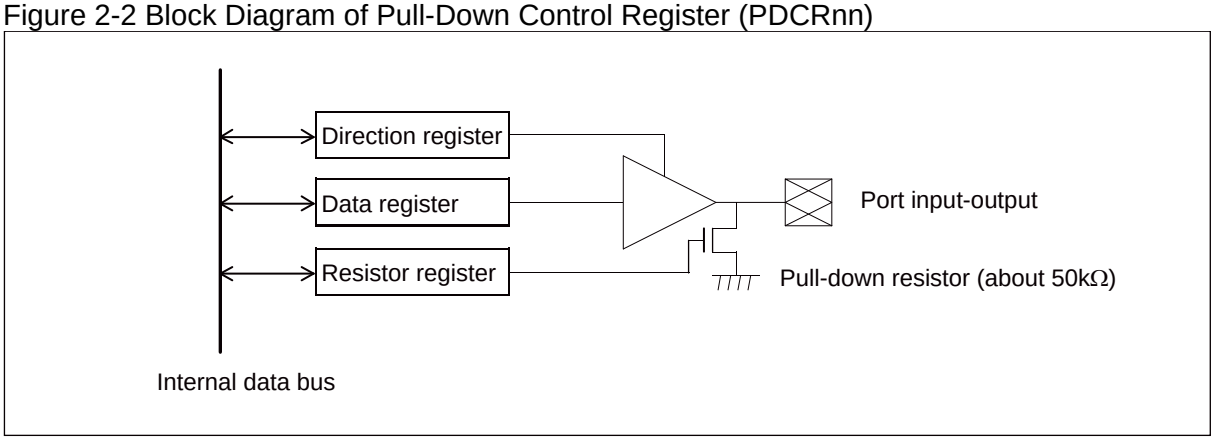
PDCRnn								
bit	7	6	5	4	3	2	1	0
	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

[bit7 to bit0] PDx (PDCR00 to PDCRnn)
These bits enable or disable the pull-down resistor for each port pin.

Bit	Description
0	No Pull-Down resistor in input mode
1	Pull-Down resistor in input mode

■ Block Diagram of Pull-Down Control Register (PDCRnn)

Figure 2-2 Block Diagram of Pull-Down Control Register (PDCRnn)



Note:

In output mode, this register has no effect (no pull-down resistor). In stop and timer mode (SMCR:SPL=1), the state with no pull-down resistor is entered (high impedance).

3. Register Usage

This chapter gives a summary about usage of I/O registers.

■ Register Settings for Different Port Usage

Figure 3-1 Register Settings for Different Port Usage

Register/ Port usage	DDR	PIER	EPSR	PUCR	PDCR (if avail.)	PHDR (if avail.)	PDR (read)	PDR (write)
initial state	0 (input)	0 (off)	undefined	0 (off)	0 (off)	0 (off)	undefined	value stored in PDR
General Purpose input	set to 0 (input)	set to 1 (on)	pin state	User setting	User setting	no effect	pin state	value is stored in PDR
General Purpose output	set to 1 (output)	set to 1 (on) for EPSR	pin state (if PIER = 1)	ignored (off)	ignored (off)	User setting	PDR	value is stored in PDR and shown on pin
Peripheral Resource input (digital)	set to 0 (input)	set to 1 (on)	pin state	User setting	User setting	no effect	pin state	value is stored only in PDR
Peripheral Resource output (digital)	ignored	set to 1 (on) for EPSR	pin state (if PIER = 1)	ignored (off) *	ignored (off) *	User setting	PDR	value is stored in PDR
Analog I/O	ignored	ignored (input is off)	undefined	ignored (off)	ignored (off)	ignored	undefined	value is stored in PDR

*: Exception: When the pin is used for I^2C bus SDA or SCL line, the user setting of PUCR remains active even when I^2C bus controller is enabled.

Note:

Peripheral Resource output is activated by the corresponding peripheral resource register.
To use the digital input function for a peripheral, the pin must be set to input in the corresponding PIER register,
Analog I/O is activated by register of analog peripheral resource e.g. by ADER for ADC.
Please refer to HWM chapter of analog and peripheral resource.
One resource output per pin is allowed.

CHAPTER: 16-BIT I/O TIMER

This chapter explains the functions and operations of the 16-bit I/O Timer.

Management Code: 96F3IOTIMER-E01.0

1. Overview

The 16-bit I/O Timer consists of a 16-bit Free-Running Timer and Output Compare Units and Input Capture Units. It is used to generate pulse sequences and to measure the time duration between external events.

■ 16-bit Free-Running Timer

- The 16-bit Free-Running Timer consists of the up counter, the control register, the 16-bit compare clear register, and the prescaler. The output value obtained from this counter is used as the basic timer of the Input Capture Unit.
- Eight counter clocks are available
- An interrupt can be generated upon a counter overflow or a match with compare register 0 and 4.
- The counter value can be initialized to "0000_H" upon a reset, software clear, or if a compare match is generated between this timer and the compare clear register values.

■ Output Compare Unit (2 channels per one module)

The Output Compare Unit consist of two 16-bit compare registers, two compare output latches, and two control registers.
When a 16-bit Free-Running Timer value matches the corresponding compare register value, the output level is toggled and an interrupt can be issued.

- The two compare registers can be used independently for each Output Compare Unit.
- Output pins can be controlled based on pairs of the two compare registers.

Output pins can be toggled by using the two compare registers.

- Initial values for output pins can be set.
- Interrupts can be generated upon a compare match.

■ Input Capture Unit (2 channels per one module)

The Input Capture Unit consists of two 16-bit capture registers and two control registers, each corresponding to two independent external input pins. The 16-bit Free-Running Timer values can be stored in the capture register and an interrupt is issued simultaneously upon detection of an edge of a signal input from an external input pin.

- The detection edge of an external input signal can be specified.

Rising, falling, or both edges can be chosen.

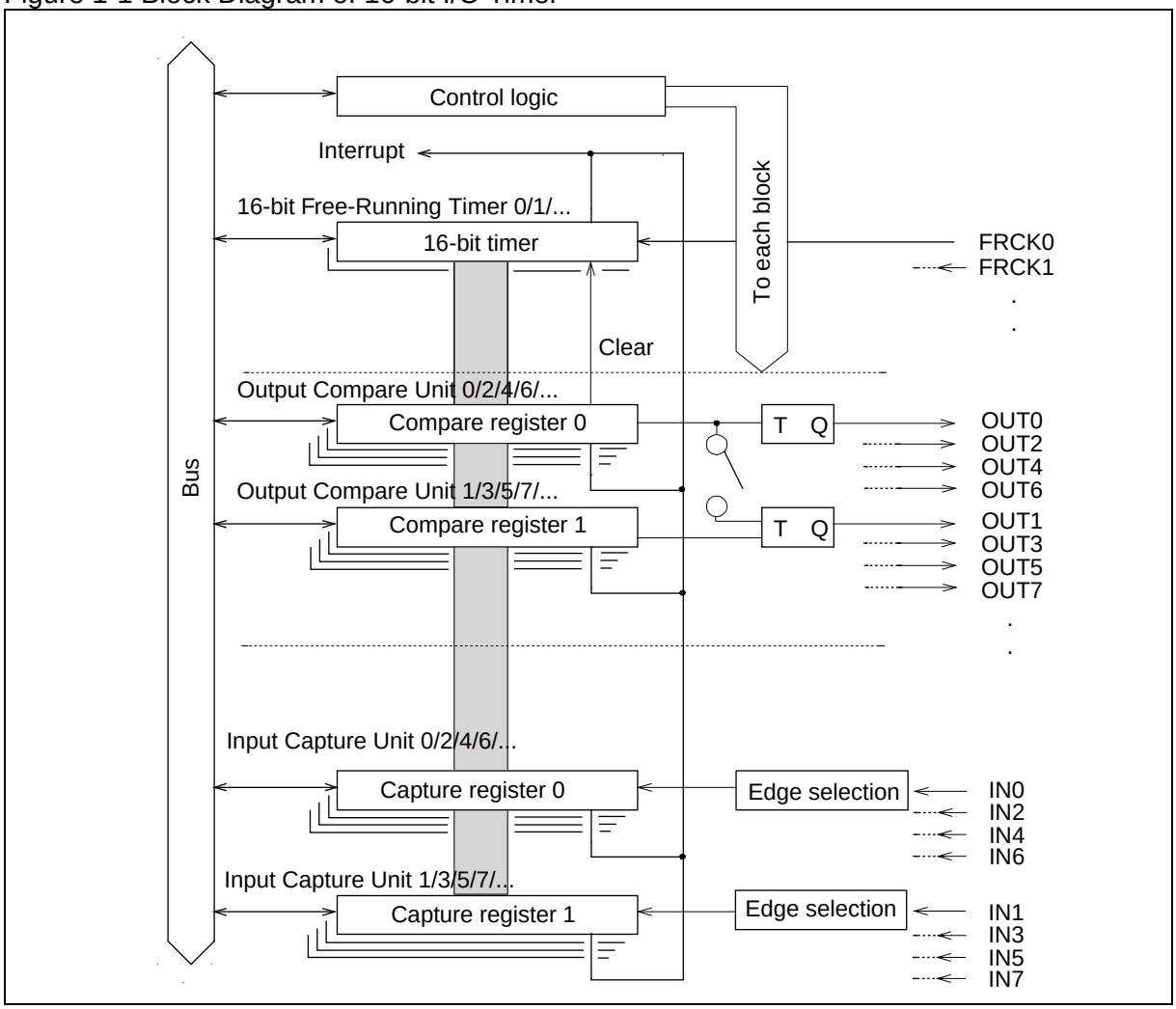
- The two input channels can operate independently.
- An interrupt can be issued upon a valid edge of an external input signal.

The DMA can be activated upon an input capture interrupt.

■ Block Diagram of the 16-bit I/O Timer

Figure 1-1 shows a block diagram of the 16-bit I/O Timer.

Figure 1-1 Block Diagram of 16-bit I/O Timer



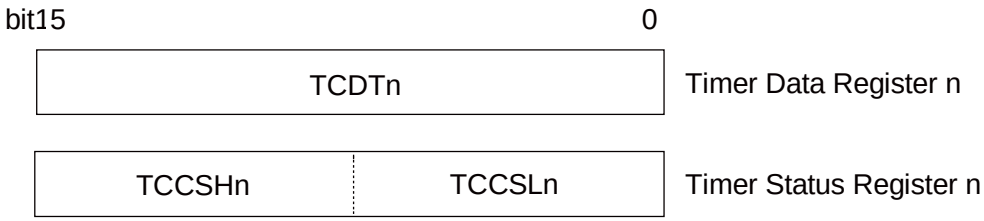
Note:

Please see the first chapter (Section "1.5 16-Bit I/O Timer Configuration" in "CHAPTER 1 OVERVIEW") and the tables therein to check the connections between input/output capture units and free running timers.

2. 16-bit I/O Timer Registers

The 16-bit I/O timer has the following registers:
- 16-bit Free-Running Timer registers
- 16-bit Output Compare registers
- 16-bit Input Capture registers

■ 16-bit Free-Running Timer Registers

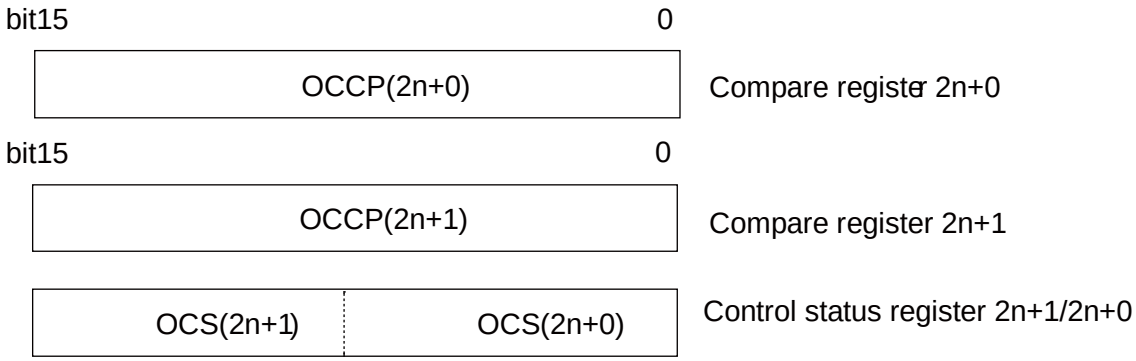


Remark:

The suffix "n" denotes the Free-Running Timer number (0, 1, 2, ...). The register name is composed by the register type name and the suffix. Hence, the following registers are available:

- for Free-Running Timer 0: TCDT0, TCCSH0, TCCSL0
- for Free-Running Timer 1: TCDT1, TCCSH1, TCCSL1
- etc.

■ 16-bit Output Compare Unit Registers

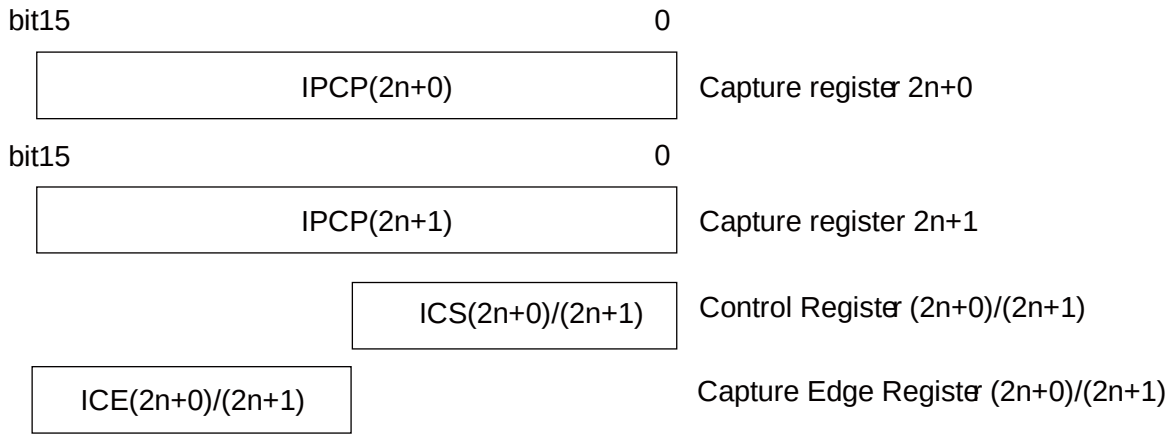


Remark:

The suffix "n" denotes the number of the Output Compare Unit (0, 1, 2, ...). The register name is composed by the register type name and the suffix. Hence, the following registers are available:

- for Output Compare Unit 0: OCCP0, OCCP1, OCS0, OCS1
- for Output Compare Unit 1: OCCP2, OCCP3, OCS2, OCS3
- etc.

■ 16-bit Input Capture Registers



Remark:

The suffix "n" denotes the number of the Input Capture Unit (0, 1, 2, ...). The register name is composed by the register type name and the suffix. Hence, the following registers are available:

- for Input Capture Unit 0: IPCP0, IPCP1, ICS01, ICE01
- for Input Capture Unit 1: IPCP2, IPCP3, ICS23, ICE23
- etc.

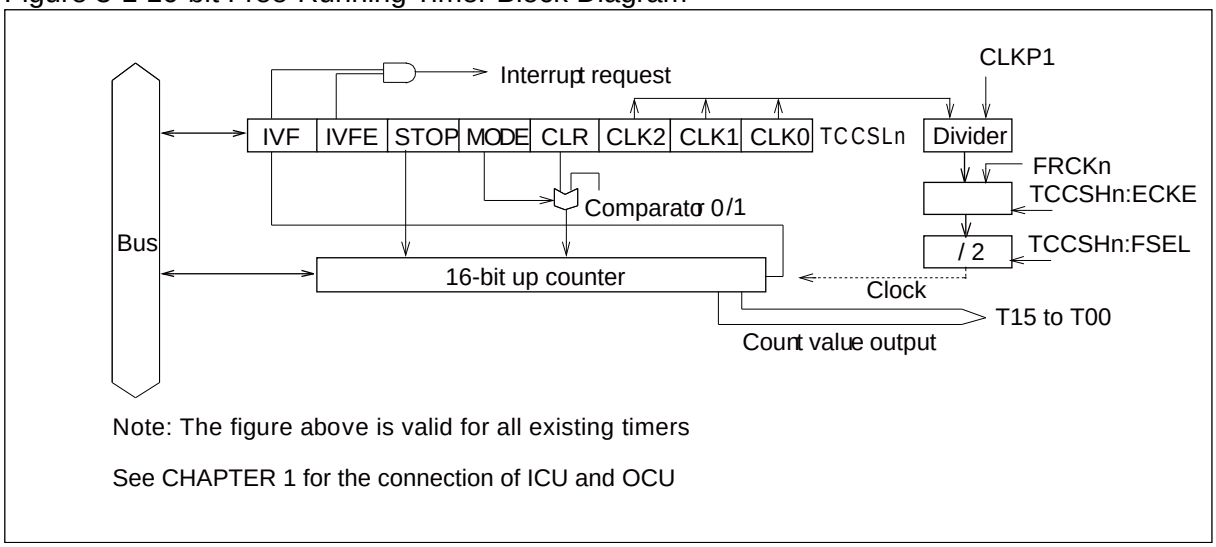
3. 16-bit Free-Running Timer

The 16-bit Free-Running Timer consists of a 16-bit up counter and a control status register. The count values of this timer are used as the base time for the Output Compare Units and Input Capture Units.

- Eight counter clock frequencies are available.
- An interrupt can be generated upon a counter value overflow.
- The counter value can be initialized upon a match with compare register 0 and compare register 4, depending on the mode.

■ 16-bit Free-Running Timer Block Diagram

Figure 3-1 16-bit Free-Running Timer Block Diagram



Note: The figure above is valid for all existing timers

See CHAPTER 1 for the connection of ICU and OCU

■ List of 16-bit Free-running Timer Registers

Abbreviated Register Name	Register Name	Reference
TCDTn	Data Register	See 3.2
TCCSLn	Control Status Register	See 3.3

3.1. Operation

The 16-bit Free-Running Timer starts counting from counter value "0000_H" after the reset is released. The counter value is used as the base time for the 16-bit Output Compare and 16-bit Input Capture operations.

■ 16-bit Free-Running Timer Operation

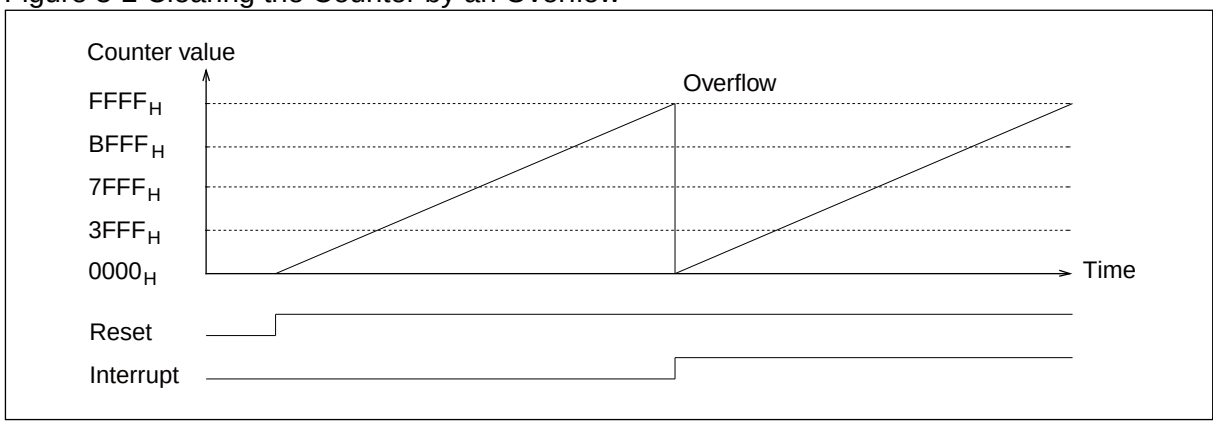
The counter value is cleared in the following conditions:

- When an overflow occurs.
- The compare clear register value matches the 16-bit Free-Running Timer value. (The mode must be set.)
- When a match with the Output Compare register 0 or Output Compare register 4 occurs. (This depends on the mode.)
- When "1" is written to the CLR bit of the TCCSL register during operation.
- When "0000_H" is written to the TCDT register during stop.
- Reset

An interrupt can be generated when an overflow occurs or when the counter is cleared by a match with the compare register 0 or 4. (Compare match interrupts can be used only in an appropriate mode.)

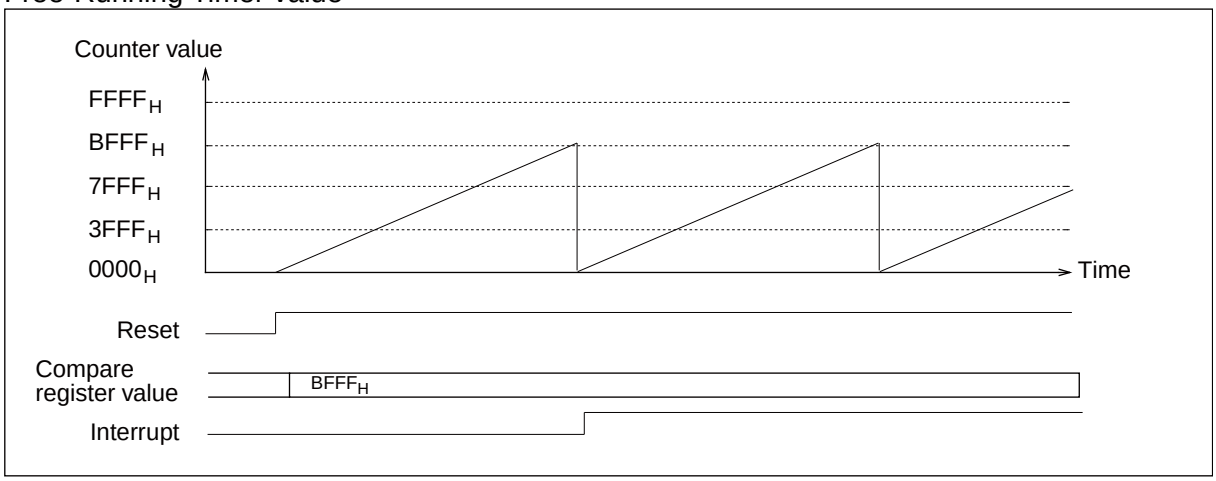
■ Clearing the Counter by an Overflow

Figure 3-2 Clearing the Counter by an Overflow



■ Clearing the Counter at Compare Clear Register Value Match

Figure 3-3 Clearing the Counter when the Compare Clear Register Value Matches the 16-bit Free-Running Timer value

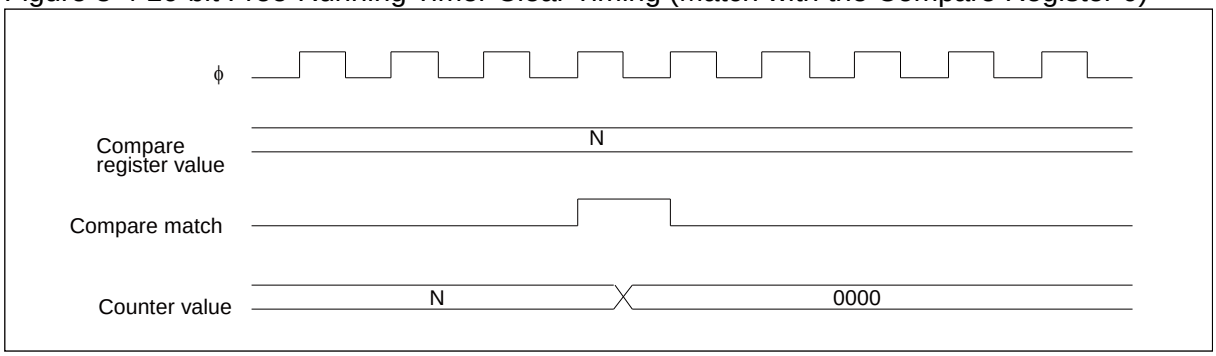


■ 16-bit Free-Running Timer Timing

● 16-bit Free-Running Timer Clear Timing (Match with the Compare Register 0)

The counter can be cleared upon a reset, software clear, or a match with the compare register 0. By a reset, the counter is immediately cleared. By a match with compare register 0 or Compare Clear Register value or by software clear (TCCS:CLR = 1), the counter is cleared in synchronization with the count timing.

Figure 3-4 16-bit Free-Running Timer Clear Timing (Match with the Compare Register 0)



3.2. Data Register (TCDTn)

The Data Register (TCDTn) can read the count value of the 16-bit Free-Running Timer. The counter value is cleared to "0000" upon a reset. The timer value can be set by writing a value to this register. However, ensure that the value is written while the operation is stopped (STOP=1).
The data register must be accessed by the word access instructions.

■ Data register of Free-Running Timer (TCDTn)

TCDTn								
bit	15	14	13	12	11	10	9	8
	T15	T14	T13	T12	T12	T10	T9	T8
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	X	X	X	X	X	X	X	X

bit	7	6	5	4	3	2	1	0
	T7	T6	T5	T4	T3	T2	T1	T0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	X	X	X	X	X	X	X	X

The 16-bit Free-Running Timer is initialized upon the following factors:

- Reset
- Clear bit (CLR) of control status register
- A match between compare register and the timer counter value.

3.3. Control Status Register (TCCSLn)

The control status register (TCCSLn) sets the operation mode of the 16-bit Free-Running Timer, starts and stops the 16-bit Free-Running Timer, and controls interrupts.

■ Control Status Register of Free-Running Timer Lower (TCCSLn)

TCCSLn								
bit	7	6	5	4	3	2	1	0
	IVF	IVFE	STOP	MODE	CLR	CLK2	CLK1	CLK0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

[bit7] IVF

This bit is the interrupt request flag bit and clear bit.

Bit	Description	
	Read	Write
0	No interrupt	A possible interrupt is cleared
1	Interrupt request	No effect

- "1" is always read during a read-modify-write cycle.
- This bit is set to "1" if the Free-Running Timer overflows or if the timer value matches with the compare register 0 or compare register 4 and the MODE bit is set to "1".

[bit6] IVFE

This bit enables the interrupt request.

Bit	Description
0	Interrupt disabled
1	Interrupt enabled

[bit5] STOP

The STOP bit is used to stop the timer.

Bit	Description
0	Counter enabled (operation)
1	Counter disabled (stop)

[bit4] MODE

Bit	Description
0	Initialization by reset or clear bit
1	Initialization by reset, clear bit, or compare register 0 or compare register 4

[bit3] CLR

Bit	Description	
	Read	Write
0	Read always "0"	No effect
1		Counter is initialized

· The CLR bit initializes the operating Free-Running Timer to the value "0000".

Note:

To initialize the counter value while the timer is stopped (STOP = 1), write "0000" to the data register (TCDT).

[bit2 to bit0] CLK2, CLK1, CLK0

CLK2	CLK1	CLK0	Count Clock Selection	Φ = 20 MHz	Φ = 16 MHz	Φ = 8 MHz	Φ = 4 MHz	Φ = 1 MHz
0	0	0	Φ	50 ns	62.5 ns	125 ns	0.25 µs	1 µs
0	0	1	Φ / 2	100 ns	125 ns	0.25 µs	0.5 µs	2 µs
0	1	0	Φ / 4	0.2 µs	0.25 µs	0.5 µs	1 µs	4 µs
0	1	1	Φ / 8	0.4 µs	0.5 µs	1 µs	2 µs	8 µs
1	0	0	Φ / 16	0.8 µs	1 µs	2 µs	4 µs	16 µs
1	0	1	Φ / 32	1.6 µs	2 µs	4 µs	8 µs	32 µs
1	1	0	Φ / 64	3.2 µs	4 µs	8 µs	16 µs	64 µs
1	1	1	Φ / 128	6.4 µs	8 µs	16 µs	32 µs	128 µs

Φ = Peripheral clock CLKP1

These bits are used to select the count clock for the 16-bit Free-Running Timer. The clock is updated immediately after a value is written to these bits. Therefore, ensure that the input capture and output compare operations are stopped before a value is written to these bits.

■ Control Status Register of Free-Running Timer Higher (TCCSHn)

TCCSHn								
bit	15	14	13	12	11	10	9	8
	ECKE	FSEL	-	-	-	-	-	-
Attribute	R/W	R/W	-	-	-	-	-	-
Initial value	0	1	X	X	X	X	X	X

[bit15] ECKE

Bit	Description
0	Select internal clock as clock source
1	Select external clock (FRCK) as clock source

This bit is used to select the internal or external clock source. Change this bit status while the output compare or input capture is stopped because the clock may be changed after writing data in the ECKE bit.

Note:

If the internal clock is selected, set bits CLK2 to CLK0 to the count clock value. This count clock is made the basic clock.
If the clock source is input from FRCK, set the corresponding port's data direction register (DDR) to "input" and set the input enable register (IER) to "enabled".

[bit14] FSEL

Bit	Description
0	Divide input clock by 2
1	Bypass input clock (no division)

This bit is used to control division of the counter clock source. Change this bit status while the output compare or input capture is stopped. The FSEL bit is used to specify the count clock division ratio.
If FSEL is set to "0", the count clock cycle specified by the count clock selection bit (TCCSL bit:CLK2 to CLK0) is divided by two.

[bit13 to bit8] -: Undefined

The read value of these bits is undefined.
Always write "0" to these bits.
Read modify write operations to this register have no effect on these bit.

4. Output Compare Unit

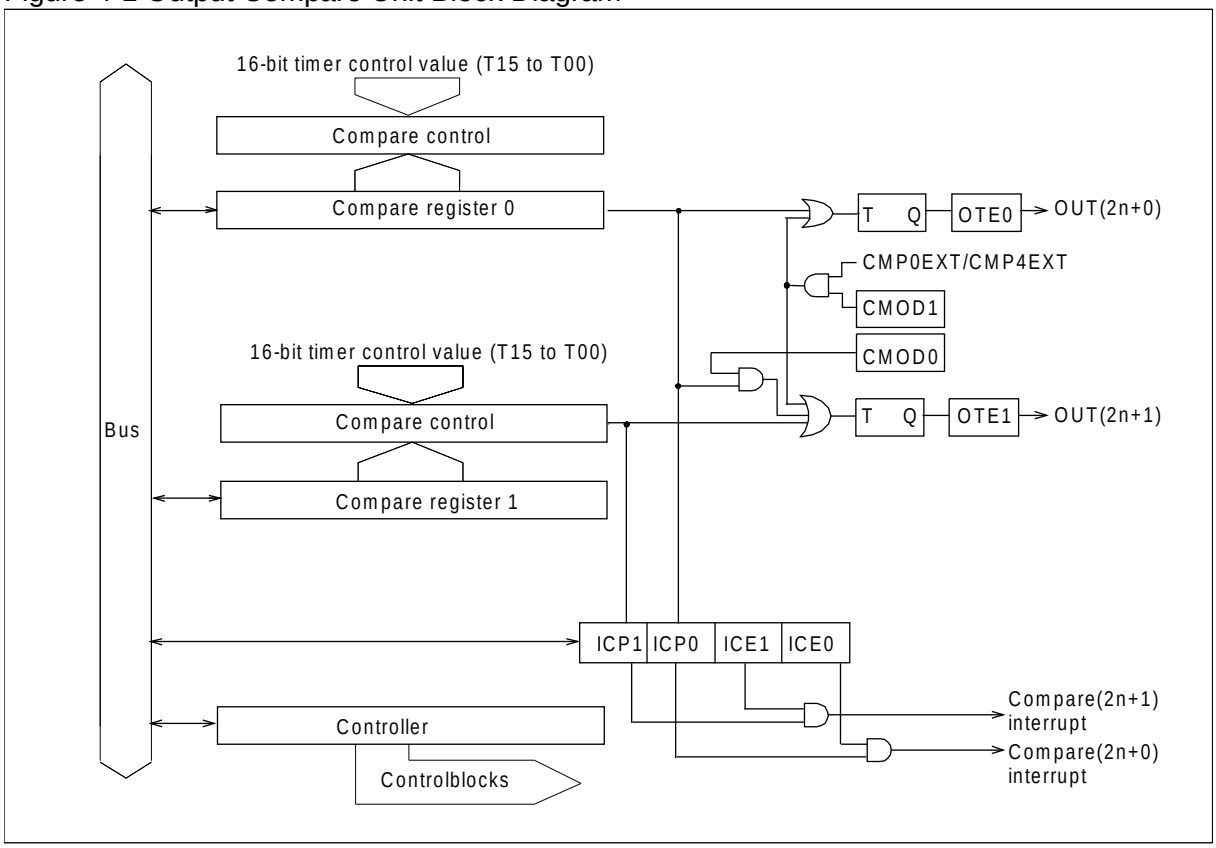
The Output Compare Unit (OCU) consists of two 16-bit compare registers, two compare output pins, and one control register. If the value written to the compare register of this module matches the 16-bit free-running timer value, the output level of the pin can be toggled and an interrupt can be issued.

■ Features of Output Compare Unit (OCU)

- For each Output Compare Unit, two compare registers exist which can be used independently. Depending on the mode setting, the two compare registers can be used to control pin outputs.
- The initial value for each pin output can be specified separately.
- An interrupt can be issued upon a match as a result of comparison.
- Each OCU can generate one PWM signal using two compare registers. (Except for OUT0 and OUT4 pins)
- Two PWM signals can be generated by one output compare module using three compare registers.
- The compare registers can be used to generate one cyclic waveform signal.

■ Output Compare Unit Block Diagram

Figure 4-1 Output Compare Unit Block Diagram



■ List of Output Compare Unit

Abbreviated Register Name	Register Name	Reference
OCCP(2n) / OCCP(2n+1)	Output Compare Register	See 4.2
OCS(2n) / OCS(2n+1)	Control Status Registers of Output Compare	See 4.3

4.1. Operation

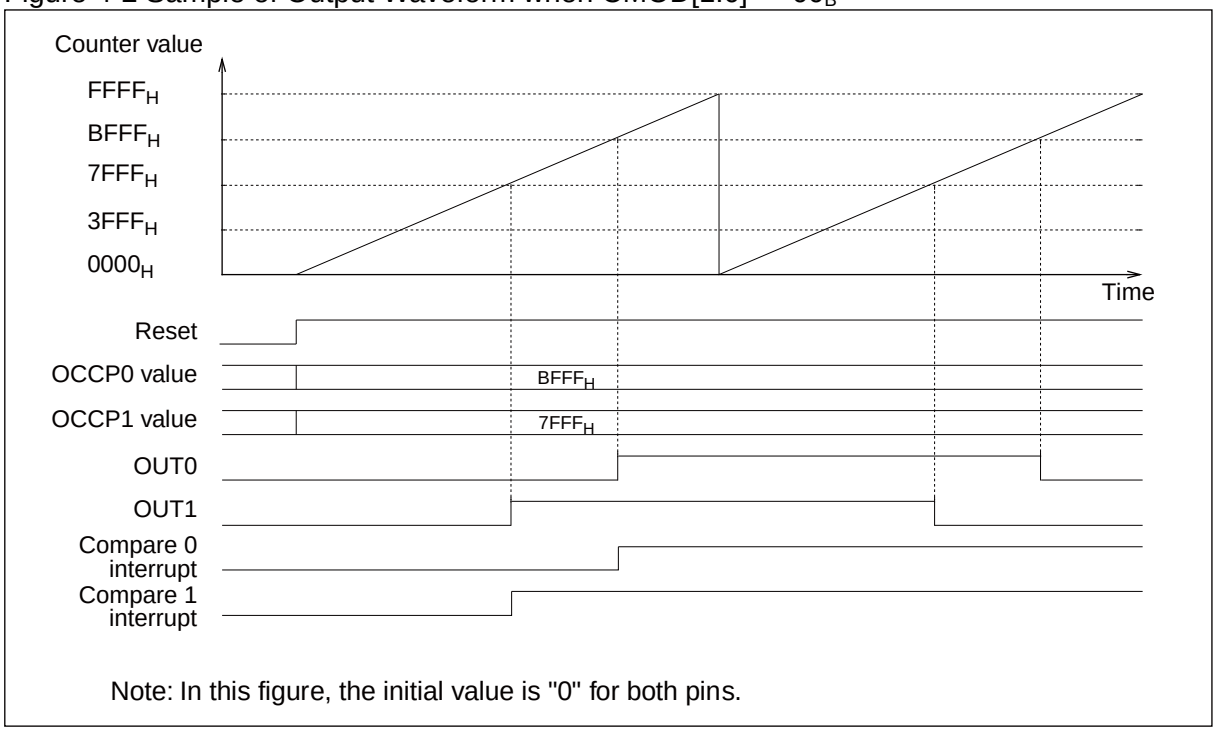
In the 16-bit Output Compare operation, an interrupt request flag can be set and the output level can be toggled when the specified compare register value matches the 16-bit free-run timer value. The CMOD0 and CMOD1 bits can be used to define the corresponding compare registers for each pin.

■ Sample Output Waveform when CMOD[1:0] = "00_B"

When CMOD[1:0] = "00_B", the output level of the pin corresponding to the compare register is reversed on every match with the register value. Each output value is controlled by one compare register.

OUT0: The level is only reversed by a match with compare register 0.
OUT1: The level is only reversed by a match with compare register 1.

Figure 4-2 Sample of Output Waveform when CMOD[1:0] = "00_B"



Note: In this figure, the initial value is "0" for both pins.

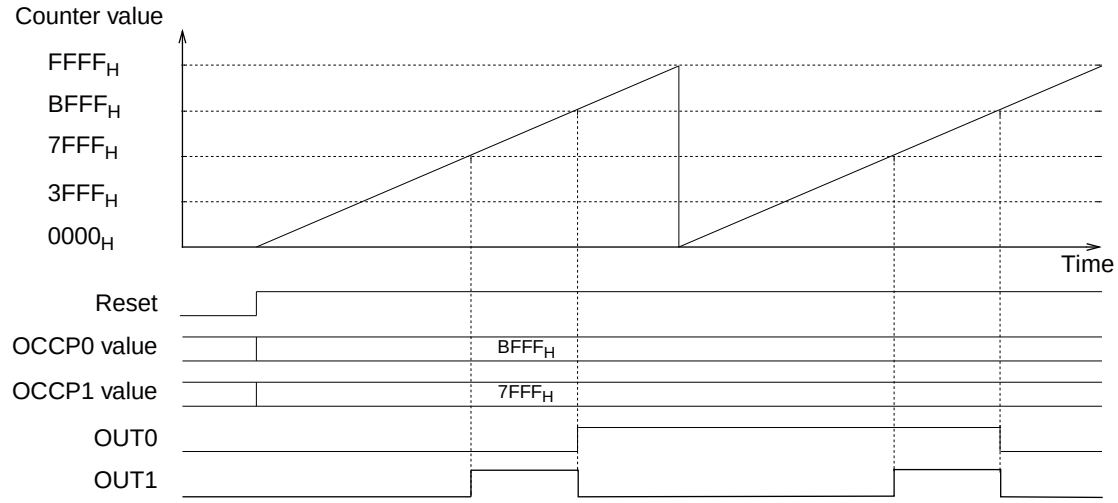
■ Sample Output Waveform with Two Compare Registers when CMOD[1:0] = "01_B"

When CMOD[1:0] = "01_B", the output level of the pin corresponding to compare register 0 (2) is reversed upon every match with the register value. This is identical to the behavior for CMOD[1:0] = "00_B". However, the output level of the second pin is reversed upon a match with either compare register 0 or compare register 1 (3). This allows to define a pulsed signal with one edge defined by the value in compare register 0 and the other edge defined by compare register 1 (3) or vice versa. If both compare registers have the same value, the operation is identical to the case for CMOD[1:0] = "00_B".
A pulse width modulated signal with differing frequency can be defined by using this mode together with the reset option by compare register match for the Free-Running Timer (MODE-bit in TCCSLx registers).

OUT0 (2): The level is only reversed by a match with compare register 0 (2).
OUT1 (3): The level is reversed by a match with compare register 0 (2) or with compare register 1 (3).

For OUT4, OUT5, OUT6 and OUT7, compare register 4 plays the same role as compare register 0 above.

Figure 4-3 Sample of a Output Waveform when CMOD[1:0] = "01_B" (no timer reset by match)



Note: In this figure, the initial value is "0" for both pins.

■ Sample Output Waveform when CMOD[1:0] = "10_B"

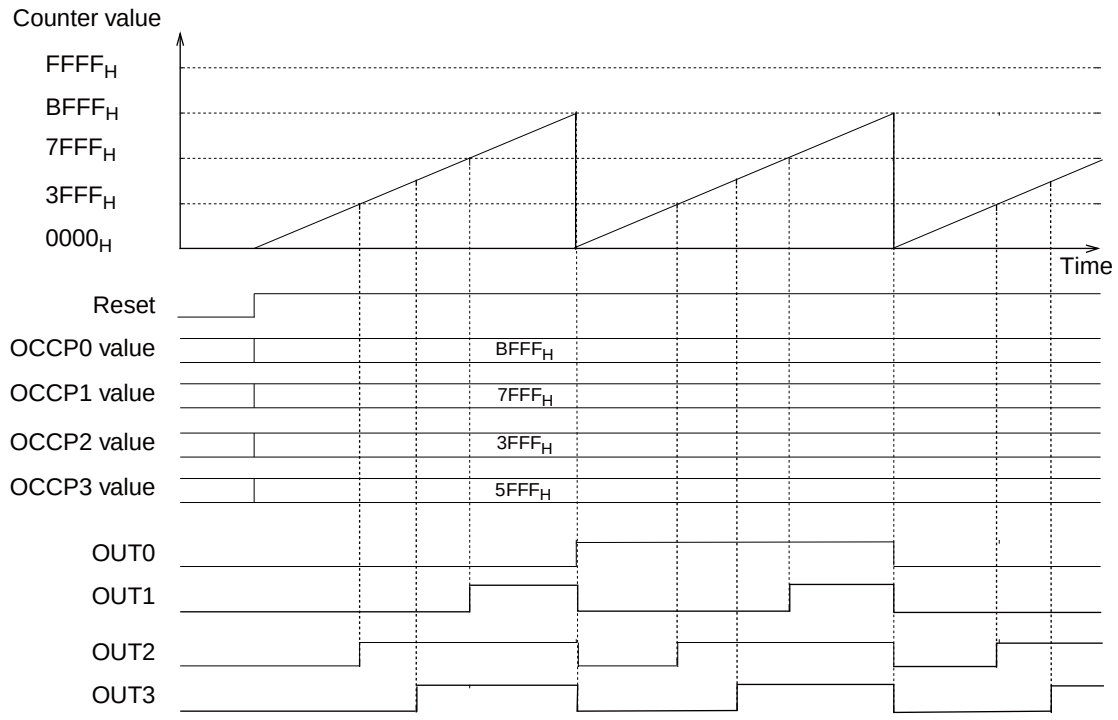
The operation mode defined by CMOD[1:0] = "10_B" is intended for the use of three pulse width modulated signals for each Free-Running Timer instead of two. If this mode is set to OCU module 1, a match of the timer value with compare register 0 reverses both OUT2 and OUT3. For the third pulsed signal, the CMOD[1:0] bits of OCU module 0 should be set to "01_B".

In register OCS1: CMOD[1:0] = "01_B"
OUT0: The level is only reversed by a match with compare register 0.
OUT1: The level is reversed by a match with compare register 0 or with compare register 1.

In register OCS3: CMOD[1:0] = "10_B"
OUT2: The level is reversed by a match with compare register 0 or with compare register 2.
OUT3: The level is reversed by a match with compare register 0 or with compare register 3.

For OUT4, OUT5, OUT6 and OUT7, compare register 4 plays the same role as compare register 0 above.

Figure 4-4 Output Waveform when OCS1:CMOD[1:0] = "01_B" and OCS3:CMOD[1:0] = "10_B"



Note: In this figure, the initial value is "0" for all pins. Timer reset is by match with compare register 0.

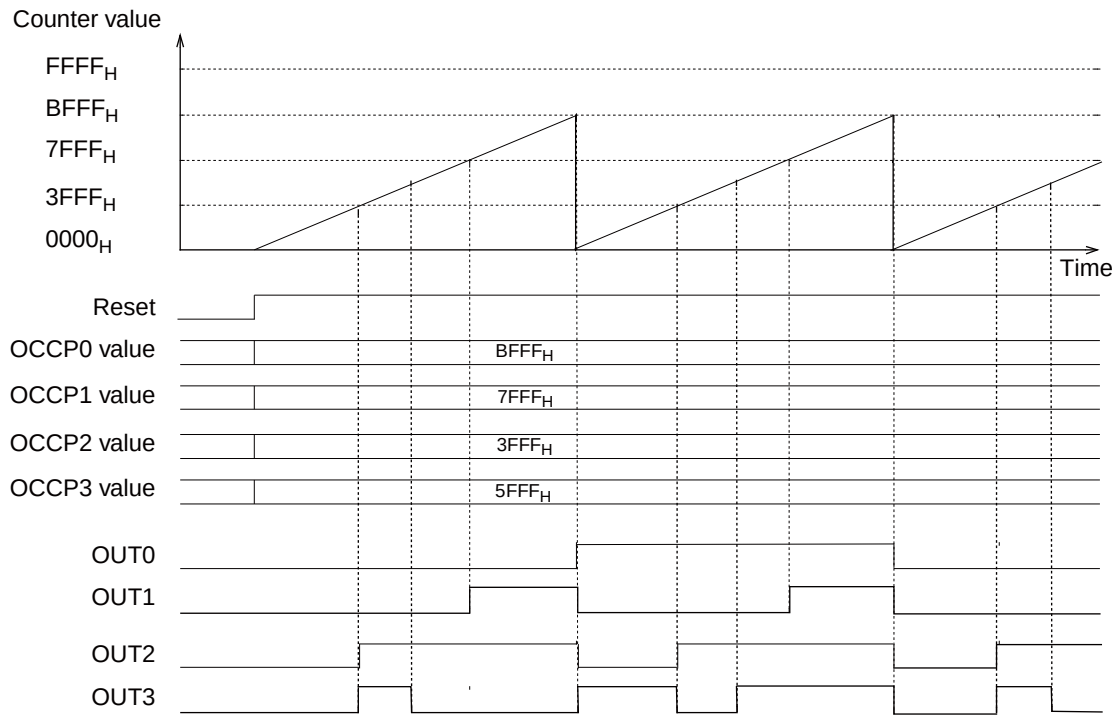
■ Sample Output Waveform when CMOD[1:0] = "11_B"

When CMOD[1:0] = "11_B", the output level of the OUT3 (OUT7) pin is reversed by the compare registers 0, 2 or 3 (4, 6 or 7). For the pin OUT1 (OUT5), this setting is identical to CMOD[1:0] = "01_B" (see also Table 4-1 Function of CMOD1 and CMOD0 Bits)

OUT0: The level is only reversed by a match with compare register 0.
OUT1: The level is reversed by a match with compare register 0 or with compare register 1.
OUT2: The level is reversed by a match with compare register 0 or with compare register 2.
OUT3: The level is reversed by a match with compare register 0, compare register 2 or with compare register 3.

For OUT4, OUT5, OUT6 and OUT7, compare register 4 plays the same role as compare register 0 above.

Figure 4-5 Output Waveform when OCS1:CMOD[1:0] = "11_B" and OCS3:CMOD[1:0] = "11_B"



Note: In this figure, the initial value is "0" for all pins. Timer reset is by match with compare register 0.

■ Output Compare Timing

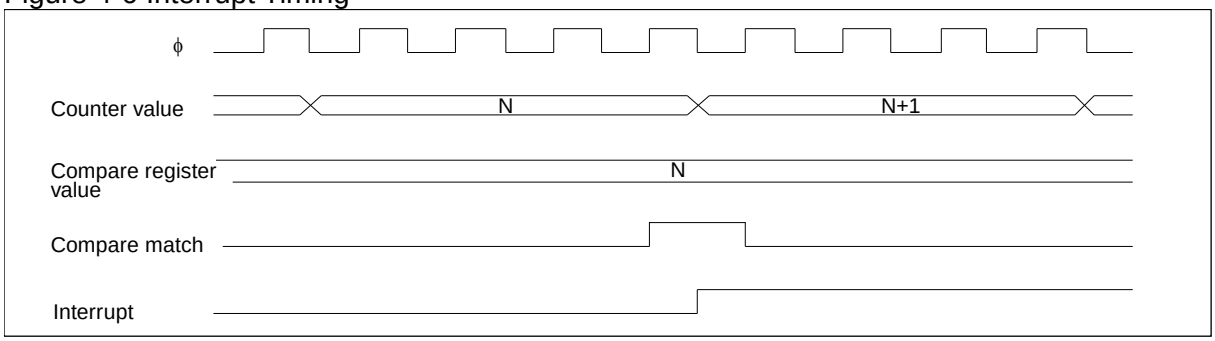
In output compare operation, a compare match signal is generated when the Free-Running Timer value matches the specified compare register value. The output value can be reversed and an interrupt can be issued. The output reverse timing upon a compare match is synchronized with the counter timing.

● Compare Operation Upon Update of Compare Register

When the compare register is updated, comparison with the counter value is not performed.

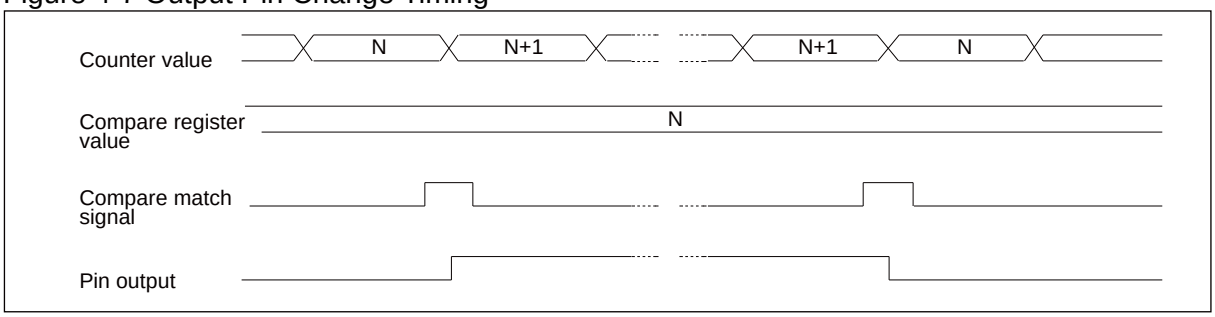
● Interrupt Timing

Figure 4-6 Interrupt Timing



● Output Pin Change Timing

Figure 4-7 Output Pin Change Timing



4.2. Output Compare Register (OCCP(2n) / OCCP(2n+1))

The 16-bit Output Compare registers are compared with the 16-bit Free-Running Timer. Since the initial register values are undefined, set appropriate value before enabling the operation. These registers must be accessed by the word access instructions. When the value of the register matches that of the 16-bit Free-Running Timer, a compare signal is generated and the output compare interrupt flag is set. If output is enabled, the output level corresponding to the compare register is reversed.
If a compare match coincides with write operation to this register, resulting compare operation is not predictable. Therefore make sure to evaluate the counter value of the free running timer or to initialize the timer before writing to this register.

■ Output Compare Register (OCCP(2n) / OCCP(2n+1))

OCCPn								
bit	15	14	13	12	11	10	9	8
	C15	C14	C13	C12	C12	C10	C9	C8
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	X	X	X	X	X	X	X	X

bit	7	6	5	4	3	2	1	0
	C7	C6	C5	C4	C3	C2	C1	C0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	X	X	X	X	X	X	X	X

Note:

This register can only be accessed as 16-bit word.

4.3. Control Status Registers of Output Compare (OCS(2n) / OCS(2n+1))

The Control Status Register (OCS(2n) / OCS(2n+1)) sets the operation mode of output compare, starts and stops output compare, controls interrupts, and sets the external output pins.

■ Output Compare Control Status Register (OCS(2n))

OCS(2n)								
bit	7	6	5	4	3	2	1	0
	ICP	ICP	ICE	ICE	-	-	CST	CST
Attribute	R/W	R/W	R/W	R/W	-	-	R/W	R/W
Initial value	0	0	0	0	X	X	0	0

Remark:

The suffix "n" denotes the Output Compare Unit number (0, 1, 2, ...). The register name and the bit names are composed by the register/bit type name and the suffix. For example:

- for Output Compare Unit 0: n = 0, hence OCS0 has bits CST0, CST1, ICE0, ICE1, ICP0, ICP1
- for Output Compare Unit 1: n = 1, hence OCS2 has bits CST2, CST3, ICE2, ICE3, ICP2, ICP3
- etc.

[bit7] ICP(2n+1)

Bit	Description	
	Read	Write
0	No compare match for channel (2n+1)	Clear this bit
1	Compare match for channel (2n+1)	No effect

- These bits are used as compare match status flags. When the compare register value matches the 16-bit free-run timer value, the bit is set to "1".
- When a compare match status bit ICP(2n+1) bit is set and the corresponding interrupt enable bit ICE(2n+1) is enabled, an output compare interrupt occurs.
- "1" is always read by a read-modify-write instruction.

[bit6] ICP(2n+0)

Bit	Description	
	Read	Write
0	No compare match for channel (2n+0)	Clear this bit
1	Compare match for channel (2n+0)	No effect

- These bits are used as compare match status flags. When the compare register value matches the 16-bit free-run timer value, the bit is set to "1".
- When a compare match status bit ICP(2n+0) bit is set and the corresponding interrupt enable bit ICE(2n+0) is enabled, an output compare interrupt occurs.
- "1" is always read by a read-modify-write instruction.

[bit5] ICE(2n+1)

Bit	Description
0	Output compare interrupt disabled for channel (2n+1)
1	Output compare interrupt enabled for channel (2n+1)

- These bits are used as output compare interrupt enable flags.
- When a compare match status bit ICP(2n+1) bit is set and the corresponding interrupt enable bit ICE(2n+1) is enabled, an output compare interrupt occurs.

[bit4] ICE(2n+0)

Bit	Description
0	Output compare interrupt disabled for channel (2n+0)
1	Output compare interrupt enabled for channel(2n+0)

- These bits are used as output compare interrupt enable flags.
- When a compare match status bit ICP(2n+0) bit is set and the corresponding interrupt enable bit ICE(2n+0) is enabled, an output compare interrupt occurs.

[bit3, bit2] -: Undefined

Always write "0" to these bits.
The read value is undefined.
Read modify write operations to this register have no effect on these bits.

[bit1] CST(2n+1)

Bit	Description
0	Compare operation disabled for channel (2n+1)
1	Compare operation enabled for channel (2n+1)

- These bits are used to enable the compare operation.

Note:

Ensure that a value is written to the compare register before the compare operation is enabled.
Since output compare is synchronized with the 16-bit Free-Running Timer clock, stopping the 16-bit Free-Running Timer stops compare operation.

[bit0] CST(2n+0)

Bit	Description
0	Compare operation disabled for channel (2n+0)
1	Compare operation enabled for channel (2n+0)

· These bits are used to enable the compare operation.

Note:

Ensure that a value is written to the compare register before the compare operation is enabled.
Since output compare is synchronized with the 16-bit Free-Running Timer clock, stopping the 16-bit Free-Running Timer stops compare operation.

n = 0, 1, 2, 3, ... is the Output Compare Unit number. The bit names are composed by the bit type name and the suffix, e. g. for n = 0: OCS0 has CST(2n+0) = CST0, CST(2n+1) = CST1.
for n = 1: OCS2 has CST(2n+0) = CST2, CST(2n+1) = CST3

■ Output Compare Control Status Register (OCS(2n+1))

OCS(2n+1)								
bit	15	14	13	12	11	10	9	8
	CMOD1	-	-	CMOD0	OTE	OTE	OTD	OTD
Attribute	R/W	-	-	R/W	R/W	R/W	R/W	R/W
Initial value	0	X	X	0	0	0	0	0

[bit15, bit12] CMOD0, CMOD1

bit15	bit12	Description
0	0	see description for details

These bits define the operation mode for the pin output signals. Depending on the defined mode, the level is toggled upon a match with different compare registers. See Table 4-1 and "4.1 Operation".

[bit14, bit13] -: Undefined

- Always write "0" to these bits.
- The read value is undefined.
- Read modify write operations to this register have no effect on these bits.

[bit11] OTE(2n+1)

These bits are used to enable the Output Compare Unit output pins. The initial value for these bits is "0".

Bit	Description
0	Output Compare Unit output is disabled The corresponding pin can be used as general-purpose port or for other peripheral function.
1	Output Compare Unit output is enabled

[bit10] OTE(2n+0)

These bits are used to enable the Output Compare Unit output pins. The initial value for these bits is "0".

Bit	Description
0	Output Compare Unit output is disabled The corresponding pin can be used as general-purpose port or for other peripheral function.
1	Output Compare Unit output is enabled

[bit9] OTD(2n+1)

Bit	Description
0	Sets "0" for compare pin output for unit (2n+1)
1	Sets"1" for compare pin output for unit (2n+1)

These bits are used to change the pin output level when the Output Compare Unit output pin is enabled. The initial value of the Output Compare Unit output pin is "0". These bits can only be changed if the compare operation for the corresponding channel is disabled (OCS(2n):CST(2n+0) or OCS(2n):CST(2n+1)). When read, these bits indicate the Output Compare Unit output pin value.

[bit8] OTD(2n+0)

Bit	Description
0	Sets "0" for compare pin output for unit (2n)
1	Sets"1" for compare pin output for unit (2n)

These bits are used to change the pin output level when the Output Compare Unit output pin is enabled. The initial value of the Output Compare Unit output pin is "0". These bits can only be changed if the compare operation for the corresponding channel is disabled (OCS(2n):CST(2n+0) or OCS(2n):CST(2n+1)). When read, these bits indicate the Output Compare Unit output pin value.

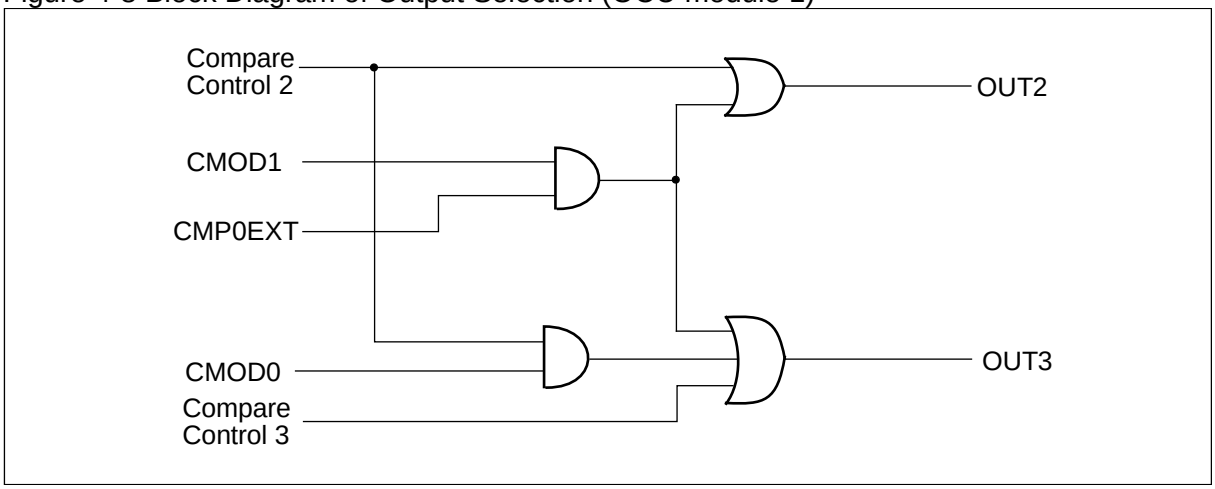
Table 4-1 Function of CMOD1 and CMOD0 Bits

Pin output value reversed upon match with compare register number.
How to read the table:

OCS(2n+1)		Register OCCPx	
CMOD1	CMOD0	output pin OUT(2n+0)	output pin OUT(2n+1)
0/1/x	0/1/x	OUT(2n+0) toggles at match with the listed OCCP numbers	OUT(2n+1) toggles at match with the listed OCCP numbers
0/1/x	0/1/x	OUT(2n+0) toggles at match with the listed OCCP numbers	OUT(2n+1) toggles at match with the listed OCCP numbers
OCS1		Toggle upon match of OCCPx	
CMOD1	CMOD0	OUT0	OUT1
x	0	x = 0	x = 1
x	1	x = 0	x = 0, x = 1
OCS3		Toggle upon match of OCCPx	
CMOD1	CMOD0	OUT2	OUT3
0	0	x = 2	x = 3
0	1	x = 2	x = 2, x = 3
1	0	x = 0, x = 2	x = 0, x = 3
1	1	x = 0, x = 2	x = 0, x = 2, x = 3

OCS5		Toggle upon match of OCCPx	
CMOD1	CMOD0	OUT4	OUT5
x	0	x = 4	x = 5
x	1	x = 4	x = 4, x = 5
OCS7		Toggle upon match of OCCPx	
CMOD1	CMOD0	OUT6	OUT7
0	0	x = 6	x = 7
0	1	x = 6	x = 6, x = 7
1	0	x = 4, x = 6	x = 4,x = 7
1	1	x = 4, x = 6	x = 4, x = 6, x = 7
OCS9		Toggle upon match of OCCPx	
CMOD1	CMOD0	OUT8	OUT9
x	0	x = 8	x = 9
x	1	x = 8	x = 8, x = 9
OCS11		Toggle upon match of OCCPx	
CMOD1	CMOD0	OUT10	OUT11
x	0	x = 10	x = 11
x	1	x = 10	x = 10, x = 11

Figure 4-8 Block Diagram of Output Selection (OCU module 1)



For OCU module 1, which requires a match with Output Compare Register 0 if CMOD[1:0] = "10_B" the comparison result from module 0 is carried inside by the CMP0EXT signal. Of course, this does not apply to module 0 itself. Here, no other register can be used but OCCP0 and OCCP1.
The equivalent situation applies to OCU module 3, where the result from module 2 is needed as CMP4EXT.

5. Input Capture Unit

The Input Capture Unit (ICU) detects a rising or falling edge or both edges of an external input signal and stores a 16-bit Free-Running Timer value at that time in a register. In addition, the Input Capture Unit can generate an interrupt upon detection of an edge. One Input Capture Unit consists of two input capture registers and one control register.

■ Features of the Input Capture Unit

- The valid edge of an external input can be selected from the following three types:

Table 5-1 Types of External Input Edges

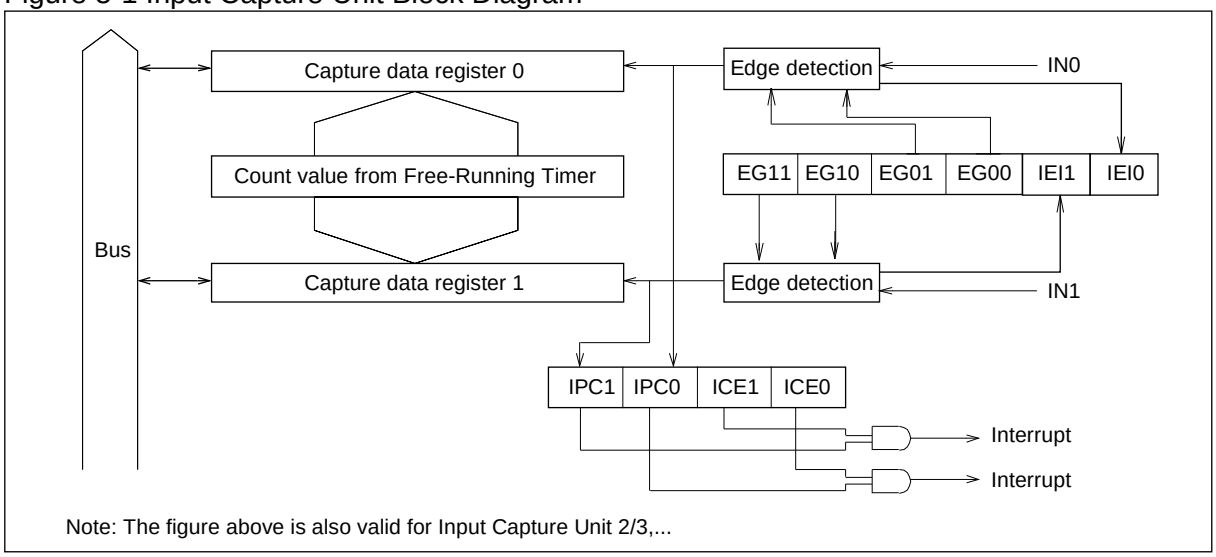
Rising edge	↑
Falling edge	↓
Both edges	↑↓

- An interrupt can be generated upon detection of a valid edge of an external input.

■ Input Capture Unit Block Diagram

Figure 5-1 shows a block diagram of the Input Capture Unit.

Figure 5-1 Input Capture Unit Block Diagram



Note: The figure above is also valid for Input Capture Unit 2/3,...

■ List of Input Capture Unit

Abbreviated Register Name	Register Name	Reference
IPCPn	Input Capture Data Register	See 5.2

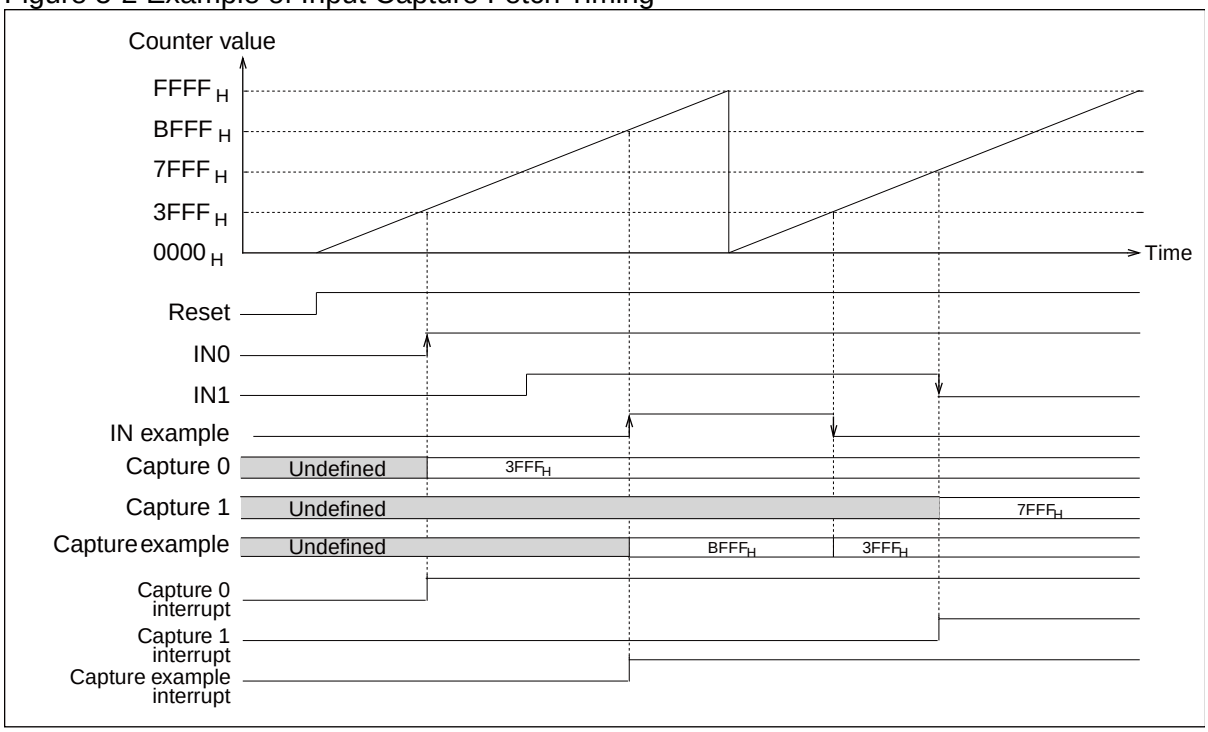
5.1. Operation

In 16-bit Input Capture operation, an interrupt can be generated upon detection of the specified edge, fetching the 16-bit Free-Running Timer value and writing it to the capture data register.

■ Example of Input Capture Fetch Timing

- Capture 0: Rising edge
- Capture 1: Falling edge
- Capture example: Both edges

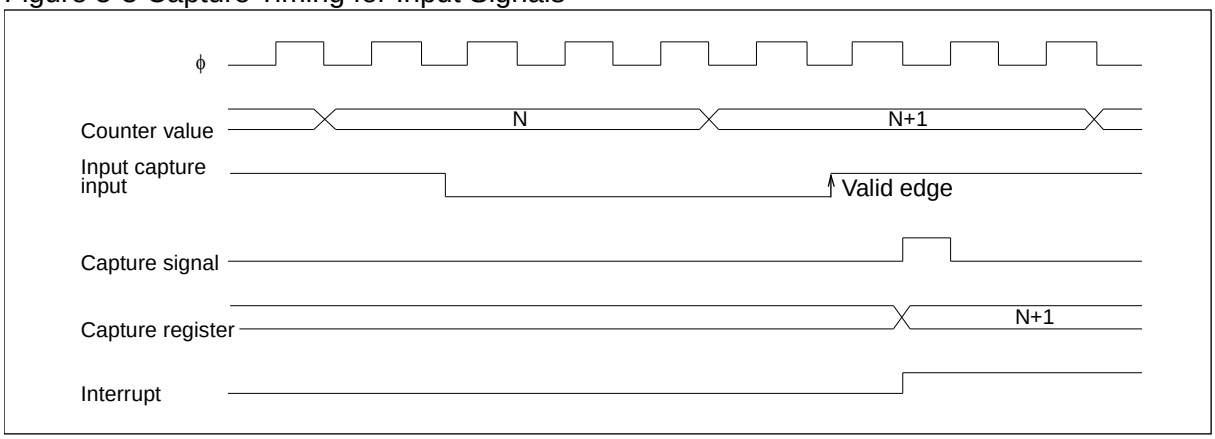
Figure 5-2 Example of Input Capture Fetch Timing



■ Input Capture Input Timing

● Capture Timing for Input Signals

Figure 5-3 Capture Timing for Input Signals



5.2. Input Capture Data Register (IPCPn)

The Input Capture Unit is configured by the Control Status Register (ICS(2n)(2n+1)) and the Edge Register (ICE(2n)(2n+1)).
The Input Capture Unit has a 16-bit data register (IPCPn). This register stores a value from the 16-bit Free-Running Timer when a valid edge of the corresponding external pin input waveform is detected. For data consistency, it should be accessed only in word mode.

■ Input Capture Data Register (IPCPn)

IPCPn								
bit	15	14	13	12	11	10	9	8
	CP15	CP14	CP13	CP12	CP11	CP10	CP09	CP08
Attribute	R	R	R	R	R	R	R	R
Initial value	X	X	X	X	X	X	X	X
bit	7	6	5	4	3	2	1	0
	CP07	CP06	CP05	CP04	CP03	CP02	CP01	CP00
Attribute	R	R	R	R	R	R	R	R
Initial value	X	X	X	X	X	X	X	X

Remark:

For consistency of data in the upper and lower byte, always read this register in word-mode.

■ Input Capture Control Status Register (ICS(2n)(2n+1))

ICS(2n)(2n+1)

bit	7	6	5	4	3	2	1	0
	ICP (2n+1)	ICP (2n)	ICE (2n+1)	ICE (2n)	EG (2n+1)1	(2n+1)0	EG(2n)1	EG(2n)0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Remark:

The suffix "n" denotes the Input Capture Unit number (0, 1, 2, ...). The register name and the bit names are composed by the register/bit type name and the suffix. For example:

- for Input Capture Unit 0: n = 0, hence ICS01 has bits EG01, EG00, EG11, EG10, ICE0, ICE1, ICP0, ICP1
- for Input Capture Unit 1: n = 1, hence ICS23 has bits EG21, EG20, EG21, EG20, ICE2, ICE3, ICP2, ICP3
- etc.

[bit7] ICP(2n+1): Interrupt Request Flag Bit (Input capture 2n+1)

Bit	Description	
	Read	Write
0	No edge detected	Clear this bit
1	Edge detected	No effect

- This bit is used as interrupt request flag for Input Capture Unit n, second channel.
- "1" is set to this bit upon detection of a valid edge of an external input pin.
- While the interrupt enable bit (ICE(2n+1)) is set, an interrupt can be generated upon detection of a valid edge.
- In read-modify-write operation, "1" is always read.

[bit6] ICP(2n): Interrupt Request Flag Bit (Input capture 2n)

Bit	Description	
	Read	Write
0	No edge detected	Clear this bit
1	Edge detected	No effect

- This bit is used as interrupt request flag for Input Capture Unit n, first channel.
- "1" is set to this bit upon detection of a valid edge of an external input pin.
- While the interrupt enable bit (ICE(2n)) is set, an interrupt can be generated upon detection of a valid edge.
- In read-modify-write operation, "1" is always read.

[bit5] ICE(2n+1): Interrupt Request Enable Bit (Input capture 2n+1)

Bit	Description
0	Disable Interrupt
1	Enable Interrupt

- This bit is used to enable input capture interrupt request for Input Capture Unit n, second channel.
- While "1" is written to this bit, an input capture interrupt is generated when the interrupt flag (ICP(2n+1)) is set.

[bit4] ICE(2n): Interrupt Request Enable Bit (Input capture 2n)

Bit	Description
0	Disable Interrupt
1	Enable Interrupt

- This bit is used to enable input capture interrupt request for Input Capture Unit n, first channel.
- While "1" is written to this bit, an input capture interrupt is generated when the interrupt flag (ICP(2n)) is set.

[bit3, bit2] EG(2n+1)1, EG(2n+1)0

bit3	bit2	Description
0	0	No edge detection (stop)
0	1	Rising edge detection
1	0	Falling edge detection
1	1	Both edges detection

- These bits are used to specify the valid edge polarity of an external input for Input Capture Unit n, second channel.
- These bits are also used to enable input capture operation.

[bit1, bit0] EG(2n)1, EG(2n)0

bit1	bit0	Description
0	0	No edge detection (stop)
0	1	Rising edge detection
1	0	Falling edge detection
1	1	Both edges detection

- These bits are used to specify the valid edge polarity of an external input for Input Capture Unit n, first channel.
- These bits are also used to enable input capture operation.

The suffix n = 0, 1, 2, 3, ... denotes the Input Capture Unit number. The bit names are composed by their type name and the suffix. Hence, for

n = 0: ICP1, ICP0, ICE1, ICE0, EG11, EG10, EG01, EG00
n = 1: ICP3, ICP2, ICE3, ICE2, EG31, EG30, EG21, EG20 etc.

■ Input Capture Unit Edge Register (ICE(2n)(2n+1))

The Input Capture Unit Edge Register (ICE(2n)(2n+1)) contain device dependent configuration bits. These registers are described in section "1.6 Input Capture Unit source select for LIN-USART" in "CHAPTER 1 OVERVIEW".

ICE(2n)(2n+1)

bit	15	14	13	12	11	10	9	8
	-	-	-	RESV	-	RESV	IEI(2n+1)	IEI(2n)
Attribute	-	-	-	-	-	-	R/W	R/W
Initial value	X	X	X	X	X	X	0	0

[bit15 to bit13, bit11] -: Undefined

- Write always "0".
- Read value is undefined.
- Read modify write operations to this register have no effect on these bits.

[bit12, bit10] RESV: Reserved Bits

These bits contain device specific configuration bits. For details, see "1.6 Input Capture Unit source select for LIN-USART of 1 OVERVIEW."

[bit9] IEI(2n+1): Valid Edge Indication Bit

- This bit is the edge indication bit for capture register IPCP(2n+1) to indicate that a rising or falling edge is detected.

Bit	Description
0	falling edge detected
1	rising edge detected

- This bit is read only.

Note:

The read value is meaningless, if EG(2n+1)1, EG(2n+1)0 = "00".
The read value is independent from edge selected by EG(2n+1)1 and EG(2n+1)0.

[bit8] IEI(2n): Valid Edge Indication Bit

- This bit is the edge indication bit for capture register IPCP0, IPCP2, IPCP4 and IPCP6 to indicate that a rising or falling edge is detected.

Bit	Description
0	falling edge detected
1	rising edge detected

- This bit is read only.

Note:

The read value is meaningless, if EG(2n)1, EG(2n)0 = "00".
The read value is independent from edge selected by EG(2n)1 and EG(2n)0.

The suffix n = 0, 1, 2, 3, ... denotes the Input Capture Unit number. The bit names are composed by their type name and the suffix. Hence, for

n = 0: IEI1, IEI0
n = 1: IEI3, IEI2 etc.

CHAPTER: 16-BIT RELOAD TIMER (WITH EVENT COUNT FUNCTION)

This chapter explains the functions and operations of the 16-bit Reload Timer (with the Event Count Function).

Management Code: FT04-E01.1

1. Overview

The 16-bit reload timer consists of a 16-bit down-counter, a 16-bit reload register, one input pin (TINn) and one output pin (TOTn), and control registers.

■ Outline of 16-bit Reload Timer (with Event Count Function)

The 16-bit reload timer has the following features

- External and internal clock/event source
- Trigger signal programmable as rising/falling edge or both
- Gated count function
- One-shot or reload counter mode
- Counter state can be made visible at external pin
- Prescaler with 6 different settings for the internal clock and 2 settings for the external clock
- Several Reload Timers can be cascaded to form a longer Reload Timer
- TIN latch function

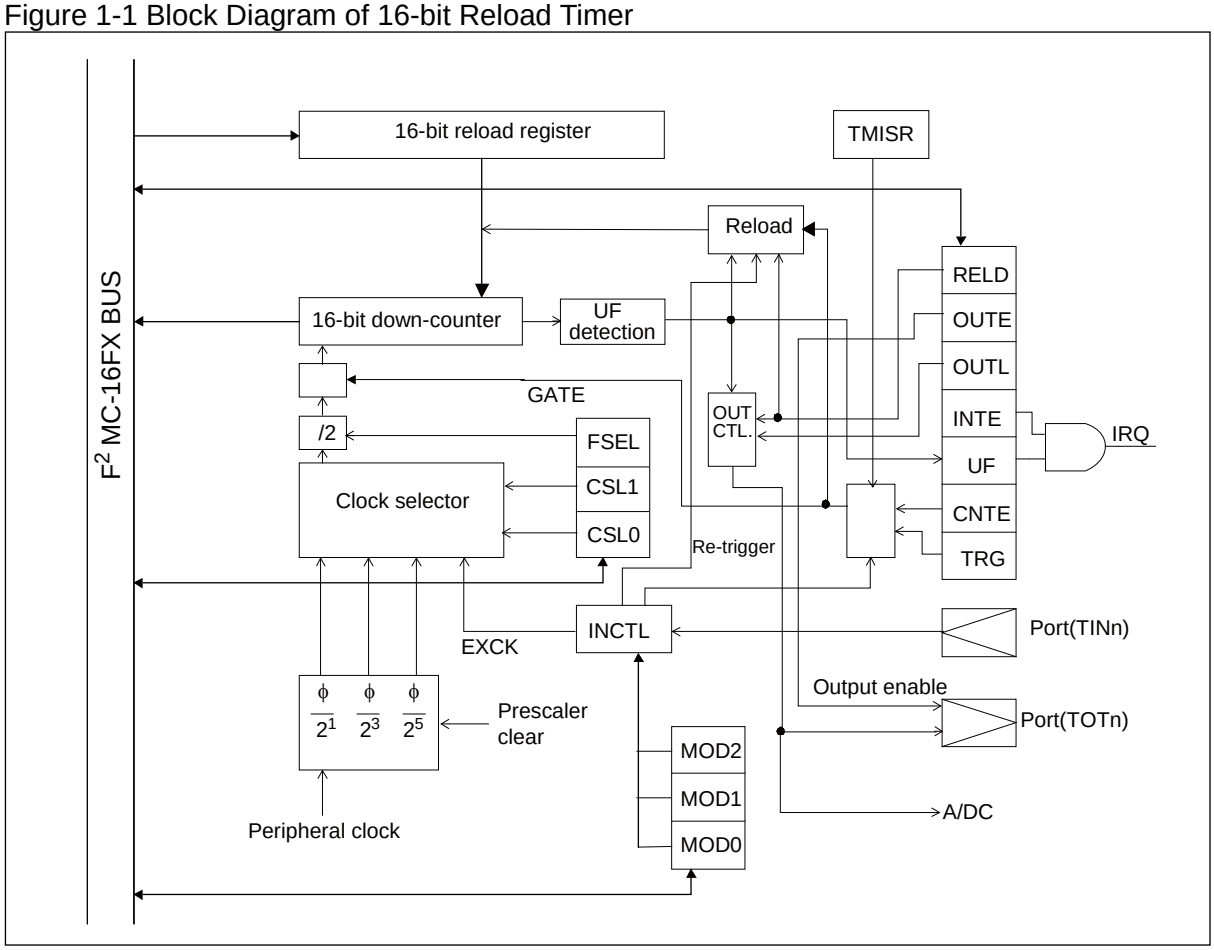
■ DMA and Interrupts

The timer can generate interrupts which can be used to start DMA transfers.

■ Block Diagram of 16-bit Reload Timer

Figure 1-1 shows a block diagram of the 16-bit reload timer.

Figure 1-1 Block Diagram of 16-bit Reload Timer



Note:

The suffix "n" denotes the Reload Timer number.
The RLT1 output can start conversion of the A/D converter.
RLT6 timer can be used as PPG clock source. It does not have TIN/TOT pins but, besides that restriction, it can be used as a normal reload timer if not required by the PPG operation.

2. Internal Clock and External Event Counter Operations

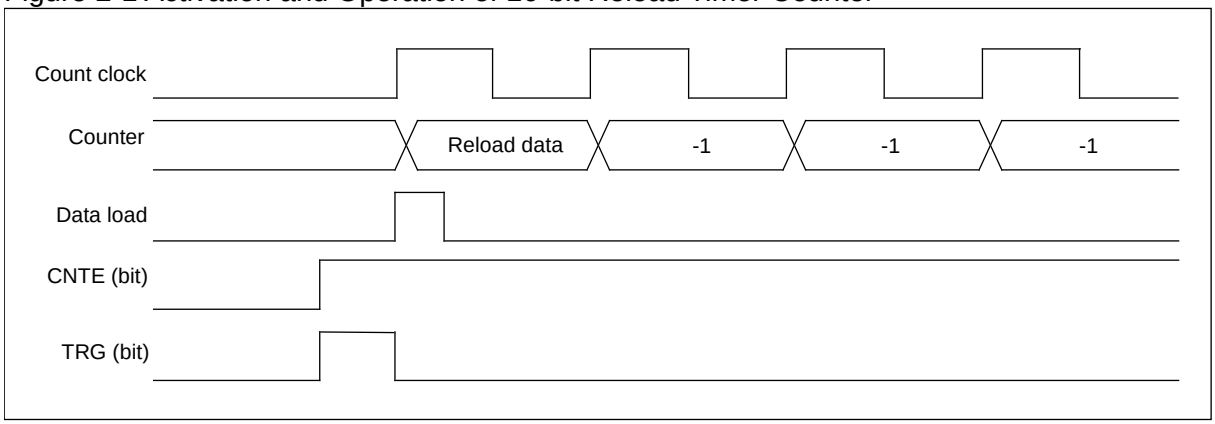
In internal clock mode, the peripheral clock CLKP1 with different divider settings can be selected as the clock source for operating the Reload Timer. The external input pin TINn can be selected as either a trigger input or gate input by a register setting.
In event counter mode, the TINn pin is used as an external event input pin. Each active edge on this pin (rising, falling or both) decrements the counter.
When FSEL = 0, then each second event is counted.

■ Internal Clock Operation of 16-bit Reload Timer

Writing "1" to both the CNTE and TRG bits in the control status register enables and starts counting at the same time. Using the TRG bit as a trigger input is always available when the timer is enabled (CNTE = "1"), regardless of the operation mode.

Figure 2-1 shows counter activation and counter operation.

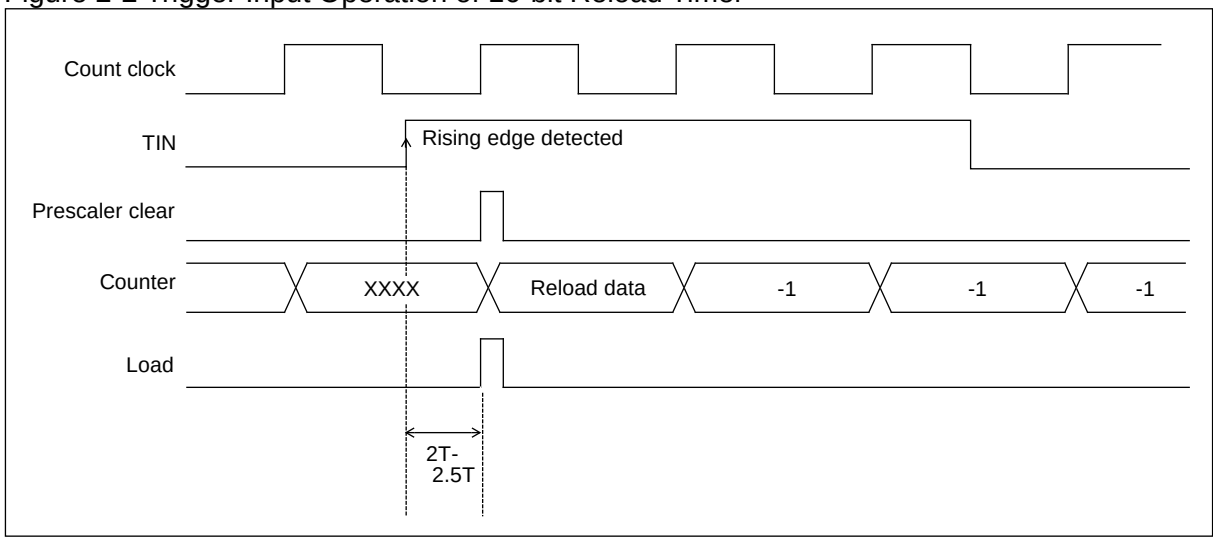
Figure 2-1 Activation and Operation of 16-bit Reload Timer Counter



■ Input Pin Functions of 16-bit Reload Timer (in Internal Clock Mode)

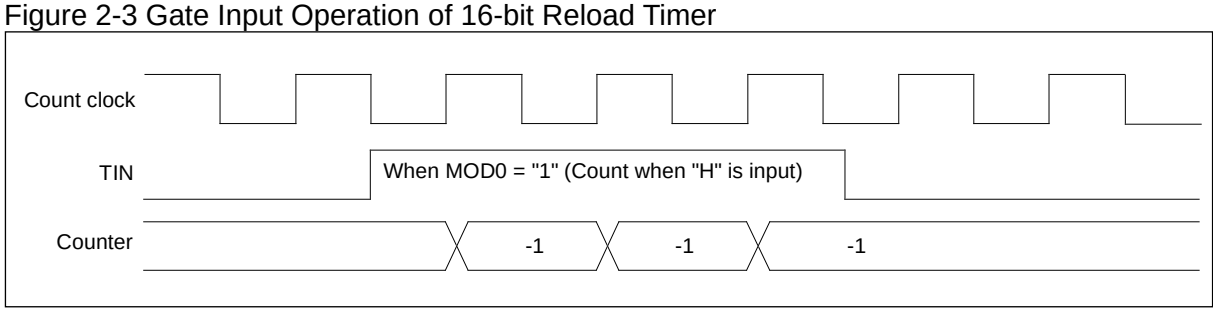
The TINn pin can be used as either a trigger input or a gate input when an internal clock is selected as the clock source. When used as a trigger input, an active edge causes the timer to load the reload register contents and resets the internal prescaler. Then, count operation starts. Minimum required pulse width of TIN is 2T + 200ns (T: cycle of peripheral clock CLKP1, 200ns is the minimum pulse length none filtered by the input noise filter).
Figure 2-2 shows the operation of trigger input.

Figure 2-2 Trigger Input Operation of 16-bit Reload Timer



When used as a gate input, the counter only counts while the active level specified by the MOD0 bit of the control register is input to the TINn pin. In this case, the count clock continues to operate unless stopped. The software trigger can be used in gate mode, regardless of the gate level. Input a pulse width of at least 2T + 200ns to the TINn pin. (T: cycle of peripheral clock CLKP1, 200ns is the minimum pulse length none filtered by the input noise filter).
Figure 2-3 shows the operation of gate input.

Figure 2-3 Gate Input Operation of 16-bit Reload Timer



■ External Event Counter

When external event count mode is selected, the TIN pin is used as an external event input. The counter counts on the active edge specified in the TMCSRn. Input a pulse width of at least 4T + 200ns (T: cycle of peripheral clock CLKP1) to the TINn pin.

3. Underflow Operation

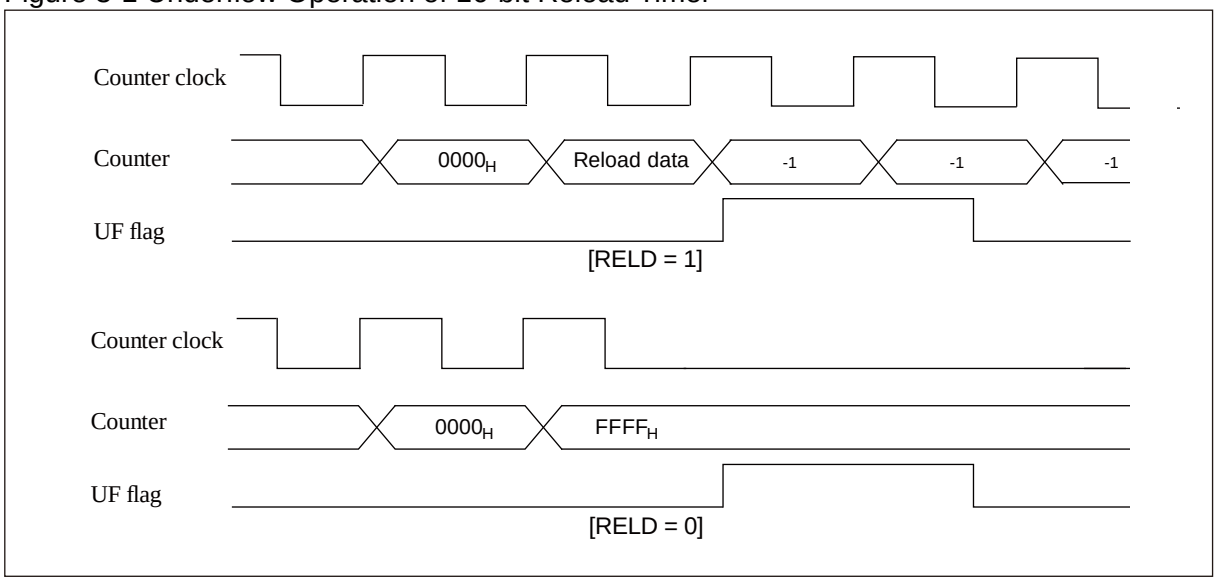
An underflow is defined for this timer as the time when the counter value changes from 0000_H to $FFFF_H$. Therefore, an underflow occurs after (reload register setting + 1) counts.

■ Underflow Operation of 16-bit Reload Timer

If the RELD bit in the control status register is "1" and an underflow occurs, the contents of the reload register is loaded into the counter and counting continues.
If the RELD bit in the control status register is "0", counting stops when counter reaches $FFFF_H$.
The UF bit in the control status register is set when the underflow occurs. If the INTE bit is "1" at this time, an interrupt request is generated.

Figure 3-1 shows the operation when an underflow occurs.

Figure 3-1 Underflow Operation of 16-bit Reload Timer



4. Output Pin Functions

In reload mode, the TOTn pin performs toggle output (inverts at each underflow). In one-shot mode, the TOTn pin is used as a pulse output that shows the configured level while the counting is in progress.

■ Output pin Functions of 16-bit Reload Timer

The OUTL bit of the control status register sets the output polarity.
When OUTL = "0", the initial value for toggle output is "L" and the one-shot pulse output is "H" while the count is in progress.
When OUTL = "1", the output waveforms are opposite.

Figure 4-1 and Figure 4-2 show the output pin functions.

Figure 4-1 Output Pin Function of 16-bit Reload Timer in Reload Mode

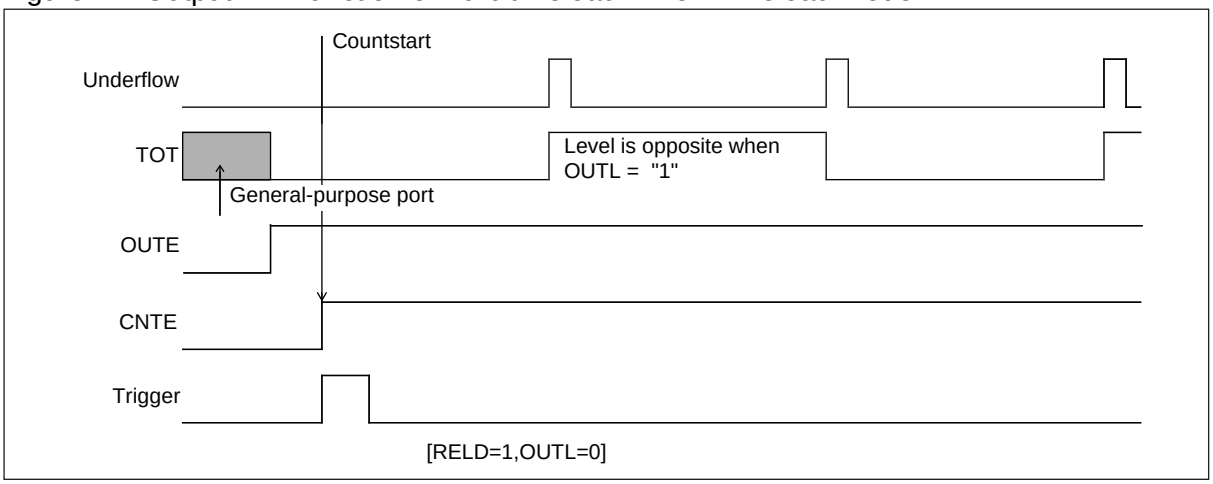
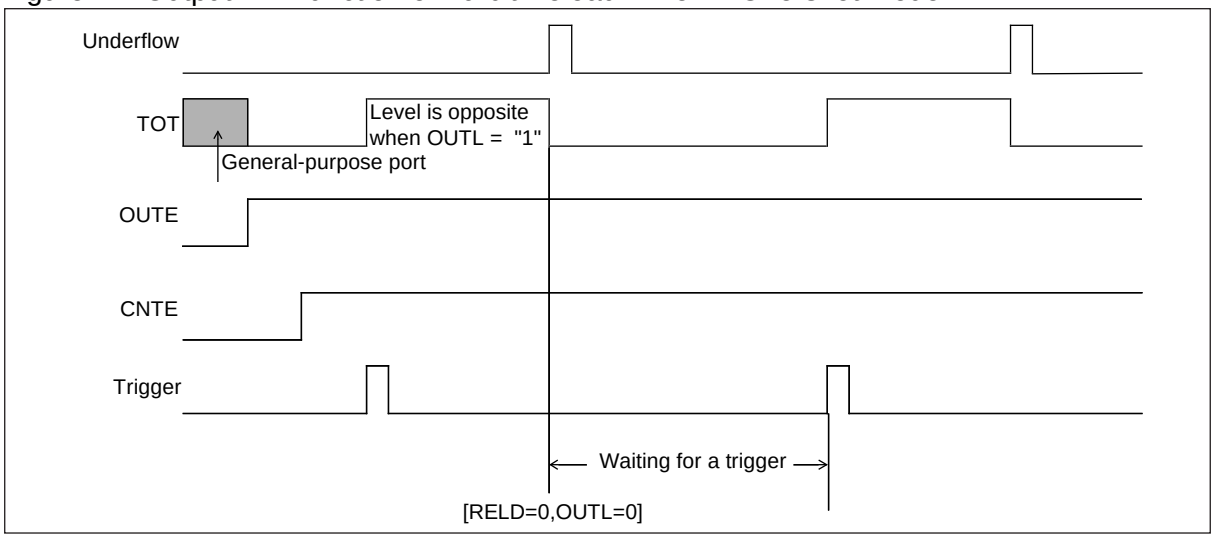


Figure 4-2 Output Pin Function of 16-bit Reload Timer in One-shot Mode



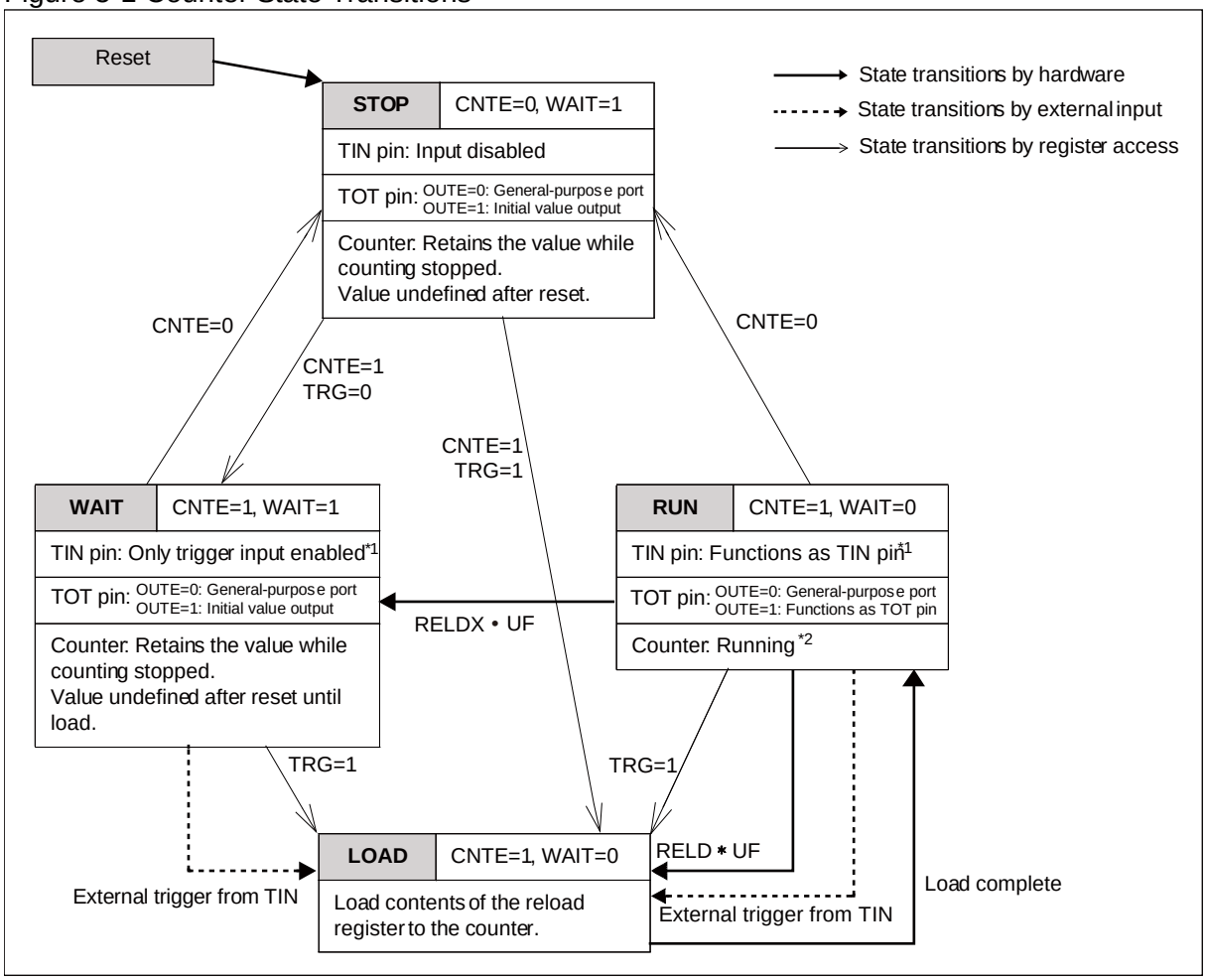
5. Counter Operation State

The counter state is determined by the CNTE bit in the control status register and the internal WAIT signal. Available states are: CNTE = "0" and WAIT = "1" (STOP state), CNTE = "1" and WAIT = "1" (WAIT state for trigger), and CNTE = "1" and WAIT = "0" (RUN state).

■ Counter Operation States

Figure 5-1 shows the transitions between each state.

Figure 5-1 Counter State Transitions

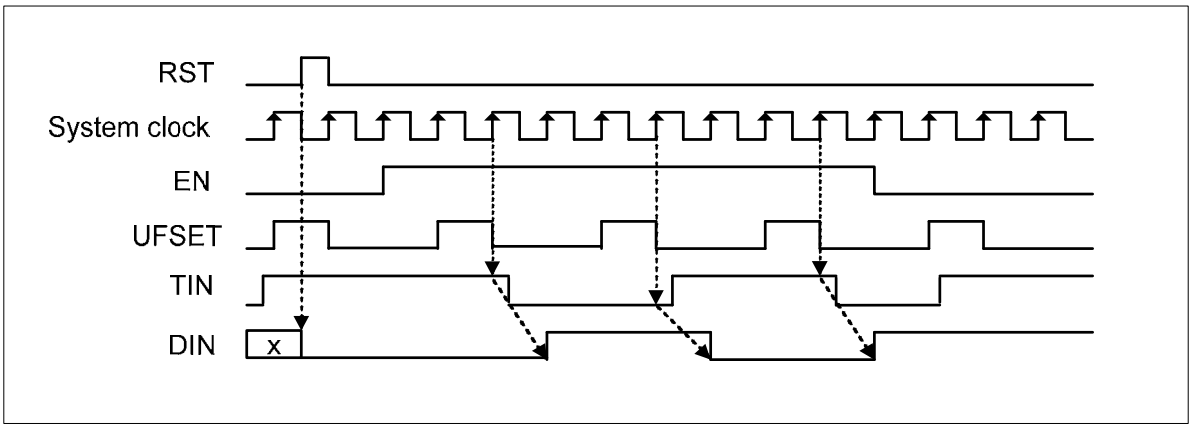


*1: Before using TIN pin, the corresponding bits of the DDR must be set to "0" and PIER register to "1".
*2: In "Gate Input mode": Counting can be influenced by TIN.

6. TIN Latch Function

The level of the terminal TIN can be latched when the underflow is occurred.

Figure 6-1 Operation of TIN Latch Function



7. 16-Bit Reload Timer (With Event Count Function)

This section explains the configuration and functions of the registers used for the 16-bit Reload Timer.

■ List of 16-bit Reload Timer Register

Abbreviated Register Name	Register Name	Reference
TMCSRn	Timer Control Status Register	See 7.1
TMRn	Register Layout of 16-bit Timer Register	See 7.2
TMRLRn	16-bit Reload Register	See 7.2

Note:

The 8-bit registers TMCSRHn, TMCSRLn can be accessed as 16-bit register TMCSRn.
The 8-bit registers TMRHn, TMRLn can be accessed as 16-bit register TMRn.
The 8-bit registers TMRLRHn, TMRLRLn can be accessed as 16-bit register TMRLRn.

7.1. Timer Control Status Register (TMCSRn)

The Timer Control Status Register controls the operation mode and interrupts for the 16-bit Reload Timer.

■ Register Layout of Timer Control Status Register (TMCSRn)

TMCSRHn								
bit	15	14	13	12	11	10	9	8
	DIN	EN	-	FSEL	CSL1	CSL0	MOD2	MOD1
Attribute	R	R/W	-	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	X	1	0	0	0	0

TMCSRLn								
bit	7	6	5	4	3	2	1	0
	MOD0	OUTE	OUTL	RELD	INTE	UF	CNTE	TRG
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

■ Register Contents of Timer Control Status Register (TMCSRn)

[bit15] DIN: TIN Data In

The level of the terminal TIN can be latched when the underflow is occurred.
Write operation has no effect.

[bit14] EN: TIN Latch Function Enable

This bit enables/disables the TIN latch function.

Bit	Description
0	TIN latch function is disabled
1	TIN latch function is enabled

[bit13] -: Undefined

- Write always '0'
- read value is undefined
- Read modify write operations to this register have no effect on this bit.

[bit12] FSEL: Count Clock Division Control

Specifies the count clock division ratio.
If the FSEL bit is set to "0", the count clock specified by the count clock selection bits (CSL1 and CSL0) is divided by two.
If the FSEL bit is set to "1", the count clock specified by the count clock selection bits (CSL1 and CSL0) is not divided (default).

[bit11, bit10] CSL1, CSL0: Clock Select 1, 0

Specifies the clock/event source and the clock division ratio.
Table 7-1 lists the selected clock sources for different FSEL and CSL0/1 settings.

Table 7-1 Clock Sources for CSL0/1 and FSEL Bit Settings

bit12	bit11	bit10	Description
1	0	0	CLKP1 / 2^1 (0.083 μs)
0	0	0	CLKP1 / 2^2 (0.167 μs)
1	0	1	CLKP1 / 2^3 (0.333 μs)
0	0	1	CLKP1 / 2^4 (0.667 μs)
1	1	0	CLKP1 / 2^5 (1.3 μs)
0	1	0	CLKP1 / 2^6 (2.6 μs)
1	1	1	External event count mode
0	1	1	External event count mode / 2

[bit9 to bit7] MOD2, MOD1, MOD0: Operation Mode and TINn Function

These bits set the operation mode and input pin (TINn) functions.

The MOD2 bit selects the input pin (TINn) function. When MOD2 = "0", the pin TINn is used as a trigger input. In this case, the reload register content is loaded to the counter when an active edge is input to the pin TINn and count operation proceeds. When MOD2 = "1", the timer operates in gated counter mode and the pin TINn is used as a gate input. In this mode, the counter only counts while an active level is input to the pin TINn.

Table 7-2 and Table 7-3 list the MOD2/1/0 bit settings.

Table 7-2 MOD2/1/0 Bit Settings for Internal Clock Mode (CSL0/1 = "00_B", "01_B", or "10_B")

bit9	bit8	bit7	Input Pin Function	Active Edge or Level
0	0	0	Trigger disabled	-
0	0	1	Trigger input	Rising edge
0	1	0		Falling edge
0	1	1		Both edges
1	x	0	Gate input	"L" level
1	x	1		"H" level

Bits marked as x in the table can be set to any value.

Table 7-3 MOD2/1/0 Bit Settings for Event Counter Mode (CSL0/1 = "11_B")

bit9	bit8	bit7	Input Pin Function	Active Edge or Level
x	0	0	-	-
	0	1	Event input	Rising edge
	1	0		Falling edge
	1	1		Both edges

Bits marked as x in the table can be set to any value.

[bit6] OUTE: Output Enable

If this bit is set to "1", the pin TOTn is used as Reload Timer output. If this bit is set to "0" the timer output TOTn is disabled.

[bit5] OUTL: Output Level

This bit sets the output level for the TOTn pin.

[bit4] RELD: Reload

This bit enables reload operations.

When RELD is "1", the timer operates in reload mode. In this mode, the timer loads the reload register contents into the counter and continues counting whenever an underflow occurs (when the counter value changes from 0000_H to $FFFF_H$).

When RELD is "0", the timer operates in one-shot mode. In this mode, the count operation stops when an underflow occurs due to the counter value changing from 0000_H to $FFFF_H$.

Table 7-4 OUTE, OUTL, and RELD Settings

bit6	bit5	bit4	Description
0	x	x	Timer output disabled
1	0	0	Output an "H" level pulse during counting.
1	1	0	Output an "L" level pulse during counting.
1	0	1	Toggle output. Starts with "L" level output. Changes level on timer reload.
1	1	1	Toggle output. Starts with "H" level output. Changes level on timer reload.

Bits marked as x in the table can be set to any value.

[bit3] INTE: Interrupt Enable

Timer interrupt request enable bit.

When INTE is "1", an interrupt request is generated when the UF bit changes to "1".

When INTE is "0", no interrupt request is generated, even when the UF bit changes to "1".

[bit2] UF: Underflow

Timer interrupt request flag. UF is set to "1" when an underflow occurs (when the counter value changes from 0000_H to $FFFF_H$). Cleared by writing "0" or by DMA Controller. Writing "1" to this bit has no effect. Read as "1" by read-modify-write instructions.

[bit1] CNTE: Count Enable

Timer count enable bit. Writing "1" to CNTE sets the timer to wait for a trigger. Writing "0" stops count operation.

[bit0] TRG: Trigger

Software trigger bit. Writing "1" to TRG applies a software trigger, causing the timer to load the reload register content to the counter and starts counting.

Writing "0" has no effect. Reading always returns "0". Applying a trigger using this register is only valid when CNTE = "1".

Writing "1" has no effect if CNTE = "0".

Set this bit in "Gate Input mode" to load the reload register content before counting starts.

7.2. Register Layout of 16-bit Timer Register (TMRn)/16-bit Reload Register (TMRLRn)

• TMRn contents
Reading this register returns the count value of the 16-bit Reload Timer. The initial value is undefined.
• TMRLRn contents
The 16-bit reload register holds the reload value. The initial value is undefined.

■ Register Layout of 16-bit Timer Register (TMRn)

TMRHn								
bit	15	14	13	12	11	10	9	8
Attribute	R	R	R	R	R	R	R	R
Initial value	X	X	X	X	X	X	X	X

TMRLn								
bit	7	6	5	4	3	2	1	0
Attribute	R	R	R	R	R	R	R	R
Initial value	X	X	X	X	X	X	X	X

■ Register Layout of 16-bit Reload Register (TMRLRn)

TMRLRHn								
bit	15	14	13	12	11	10	9	8
Attribute	W	W	W	W	W	W	W	W
Initial value	X	X	X	X	X	X	X	X

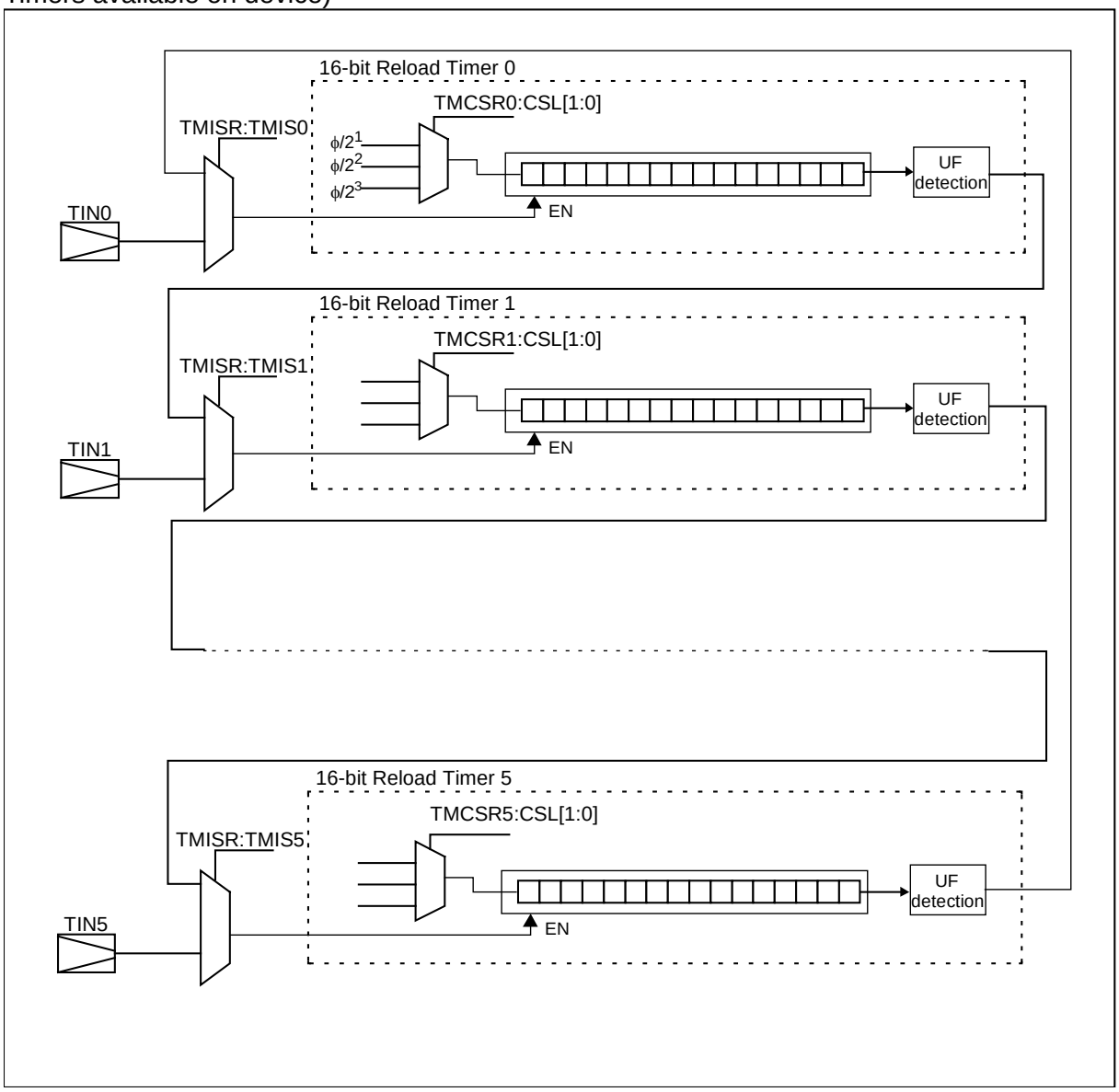
TMRLRLn								
bit	7	6	5	4	3	2	1	0
Attribute	W	W	W	W	W	W	W	W
Initial value	X	X	X	X	X	X	X	X

8. Cascading

The cascading of multiple adjacent 16-bit Reload Timers allows the user to create its own n × 16-bit-Reload Timer (Example: 3 adjacent Reload Timers available on device).

■ Cascading Overview

Figure 8-1 Block Diagram for Cascading of 16-bit Reload Timers (Example for 3 Reload Timers available on device)



■ List of Reload Timer Input Select Register

Abbreviated Register Name	Register Name	Reference
TMISR	Reload Timer Input Select Register	See 8.1

8.1. Reload Timer Input Select Register (TMISR)

■ Reload Timer Input Select Register (TMISR)

TMISR								
bit	7	6	5	4	3	2	1	0
	-	-	TMIS5	TMIS4	TMIS3	TMIS2	TMIS1	TMIS0
Attribute	-	-	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	X	X	0	0	0	0	0	0

[bit7, bit6] -: Undefined

- Write always '0'
- read value is undefined
- Read modify write operations to this register have no effect on these bits.

[bit5] TMIS5: Reload Timer 5 Input Select

Bit	Description
0	Use TIN5 as trigger input
1	Use Underflow event of Reload Timer 4 as trigger input

[bit4] TMIS4: Reload Timer 4 Input Select

Bit	Description
0	Use TIN4 as trigger input
1	Use Underflow event of Reload Timer 3 as trigger input

[bit3] TMIS3: Reload Timer 3 Input Select

Bit	Description
0	Use TIN3 as trigger input
1	Use Underflow event of Reload Timer 2 as trigger input

[bit2] TMIS2: Reload Timer 2 Input Select

Bit	Description
0	Use TIN2 as trigger input
1	Use Underflow event of Reload Timer 1 as trigger input

[bit1] TMIS1: Reload Timer 1 Input Select

Bit	Description
0	Use TIN1 as trigger input
1	Use Underflow event of Reload Timer 0 as trigger input

[bit0] TMIS0: Reload Timer 0 Input Select

Bit	Description
0	Use TIN0 as trigger input
1	Use Underflow event of Reload Timer 5 as trigger input

■ Description of Operation

The first Reload Timer in the chain can either have an internal clock or an external event as counter clock. It can be configured freely.
The next Reload Timer and all following in the chain must be set as follows:

- TMCSRn:MOD[2:0] = "101_B" : Gate count mode "H" level
- TMCSRn:CSL[1:0] = "00_B " and TMCSRn:FSEL = "1" : CLKP1 is divided by 2 (cf Table 7-1) but all RLT, except the first RLT, are clocked with CLKP1 divided by 2 to make sure that the UF signal, which is CLKP1 period long, is sampled.
- TMISR:TMISx = "1" : Use underflow signal of preceding Reload Timer to TMISx bit as count enable signal.

All Reload Timers must be triggered before counting starts to load the reload value and to change the Reload timers to RUN-State (see Figure 5-1).

CHAPTER: PROGRAMMABLE PULSE GENERATOR

This chapter explains the functions and operations of the Programmable Pulse Generator.

Management Code: 96F6PPG-E01.0

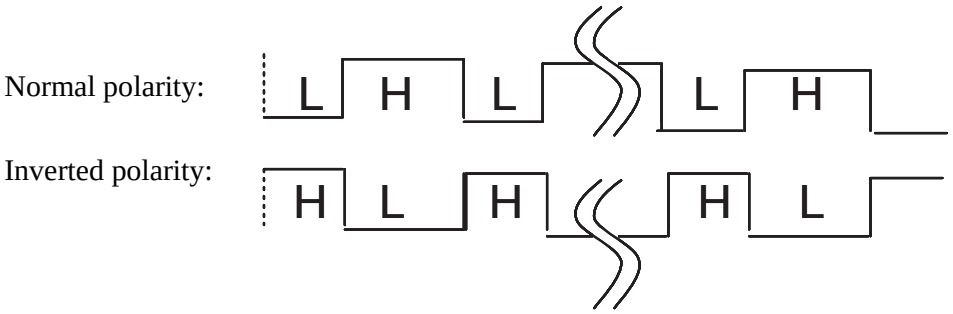
1. Overview

Programmable Pulse Generators (PPGs) are used to obtain one-shot (rectangular wave) output or pulse width modulation (PWM) output. With their software-programmable cycle and duty capability, possibility to postpone the start of PWM output signal generation by software and trigger an AD conversion, the PPGs comfortably fit into a broad range of applications. To increase flexibility, the PPGs can be configured as a 16-bit resolution PWM channel or two independent 8-bit resolution PWM outputs. Moreover, the PPGs can operate in a ramp mode, changing the output signal duty between defined start duty and end duty values.

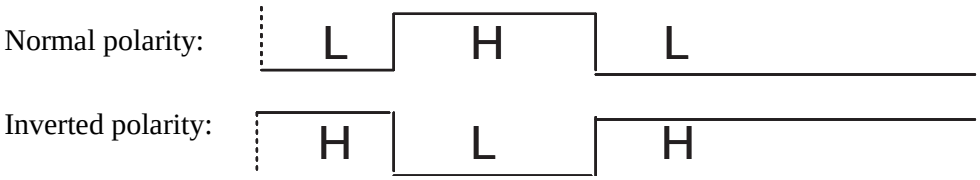
■ Features of the Programmable Pulse Generator

Output waveforms: The PPGs can generate the following kinds of waveforms

PWM waveform



One-shot waveform (Rectangular wave)



Clamped output

- Normal polarity: "L" Clamped output
- Inverted polarity: "H" Clamped output

Count clock: Choose from 8 choices:

- 1, 1/4, 1/16, 1/64 of the peripheral clock (CLKP1) or of selected Reload Timer underflow signal (see "CHAPTER 13 16-BIT RELOAD TIMER (WITH EVENT COUNT FUNCTION)")

Period: Setting range = Duty value ~ 65535 (defined with a 16-bit register) or Duty value ~ 255 (defined with a 8-bit register)

- Period = Count clock × (PCSR register value + 1)
- In full range operation mode period setting is defined as: Period = Count clock × PCSR register value
- (Example) Count clock = 32MHz(31.25ns), PCSR value = 63999 (16-bit operation mode)
- Period = 31.25ns × (63999+1) = 2ms

Duty: Setting range = 0 ~ Period value (defined with a 16-bit register or 8-bit register)

- Duty = Count clock × (PDUT register value + 1)
- In full range operation mode the duty setting is: Duty = Count clock × PDUT register value

Interrupt: Choose from six choices

- Software trigger or external trigger (TTGx pin)
- Counter borrow (cycle match)
- Duty match
- Counter borrow (cycle match) or duty match
- Defined timing point match within the PPG cycle
- End duty match during the ramp mode operation

Activation trigger:

- Software trigger
- Internal trigger
- External trigger (TTG pins)
- Common internal trigger, able to trigger all available PPG resources

Freely configurable on-chip available Reload Timer underflow signals as additional prescaler inputs

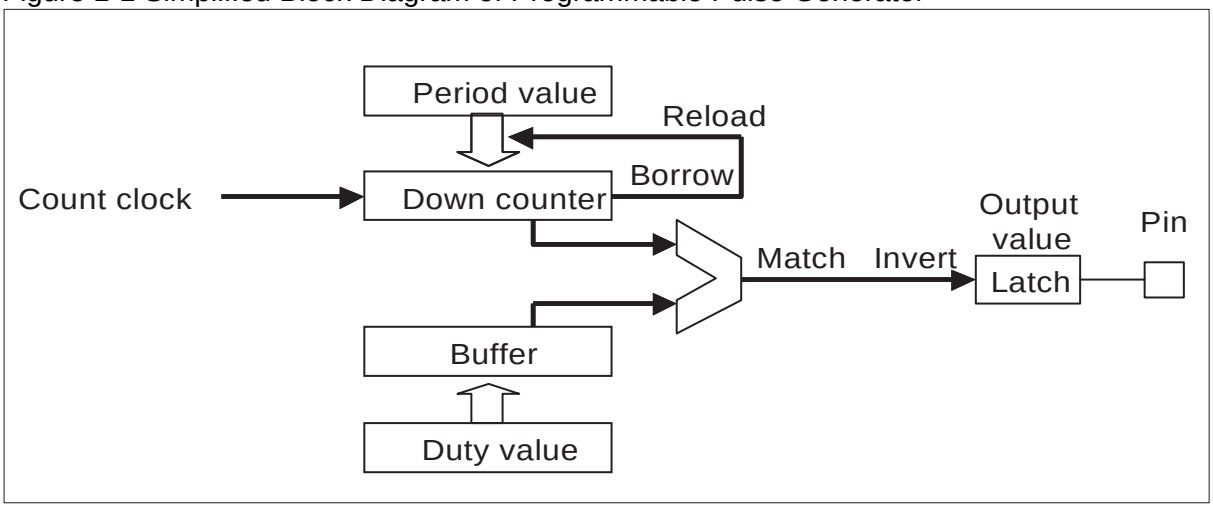
Special timing point within the PPG cycle can be configured in a way that when the PPG counter reaches the timing point value:

- ADC trigger can be generated
- an interrupt can be requested

Ramp mode operation allows to sweep the PWM signal duty between configured values of the start duty and the end duty, while controlling the ramp slope over the selected Reload Timer period

■ Simplified Block Diagram of Programmable Pulse Generator

Figure 1-1 Simplified Block Diagram of Programmable Pulse Generator

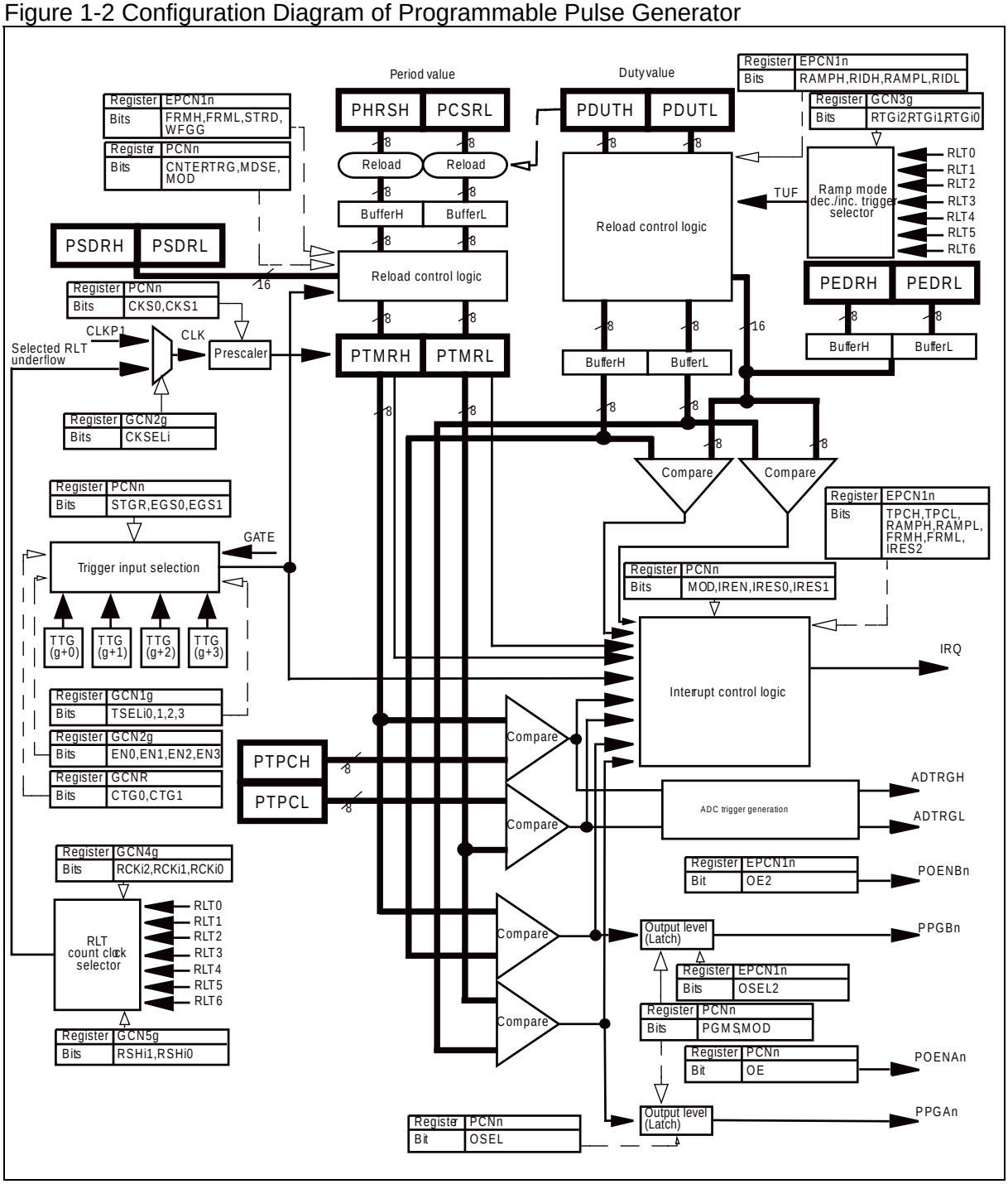


■ Configuration of Programmable Pulse Generator

Four Programmable Pulse Generators form a group with common GCN registers. In Figure 1-2 the following notation is used:

- n = Number of programmable pulse generator
- g = INT(n / 4) group number of programmable pulse generator
- i = (n MODULUS 4) index of programmable pulse generator within the group.

Figure 1-2 Configuration Diagram of Programmable Pulse Generator



Note:

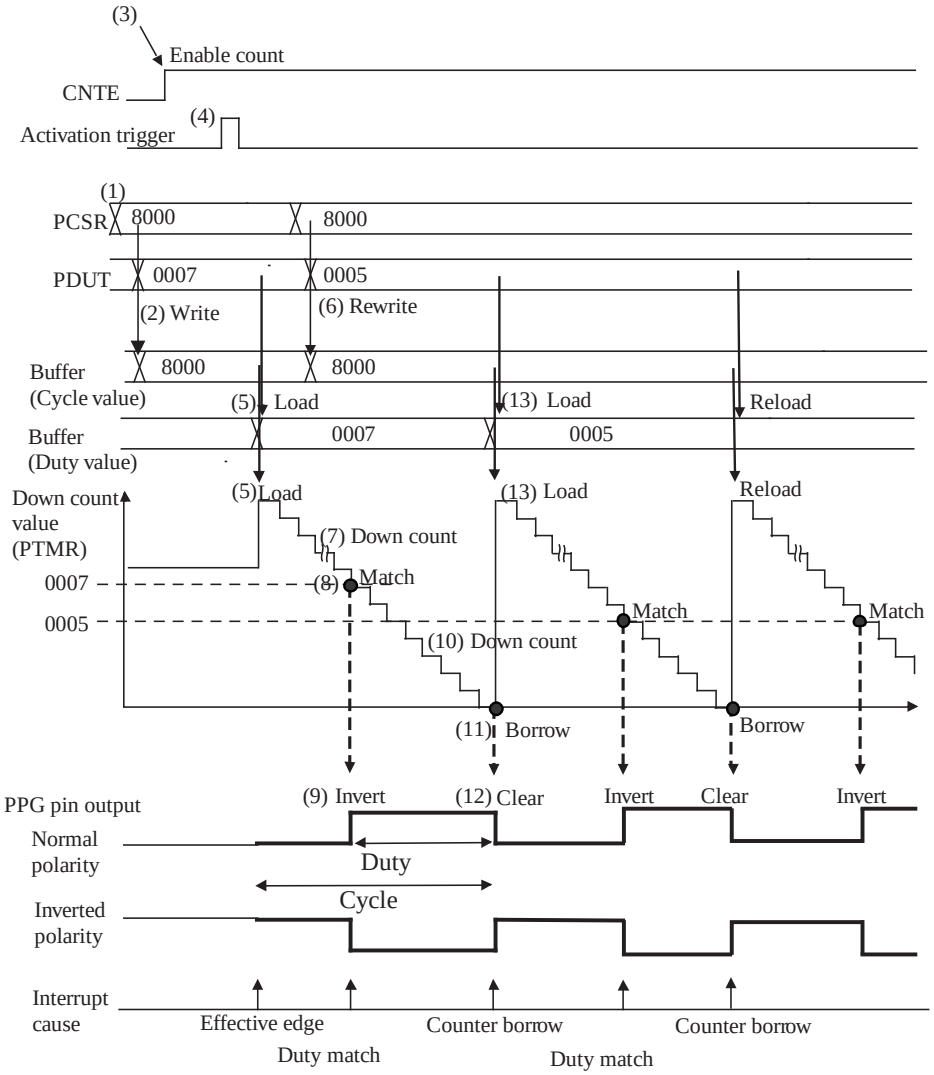
This diagram is also valid for the other available PPGs. For configuring RLT signals see "CHAPTER 13 16-BIT RELOAD TIMER (WITH EVENT COUNT FUNCTION)".
RLT can be used as a normal RLT if it is not required for the PPG operation.

2. Operation

The Programmable Pulse generators (PPGs) provide programmable pulse output independently or jointly. The individual modes of operation are described below

■ PWM Operation

In PWM operation, variable-duty pulses are generated from the PPG pin.



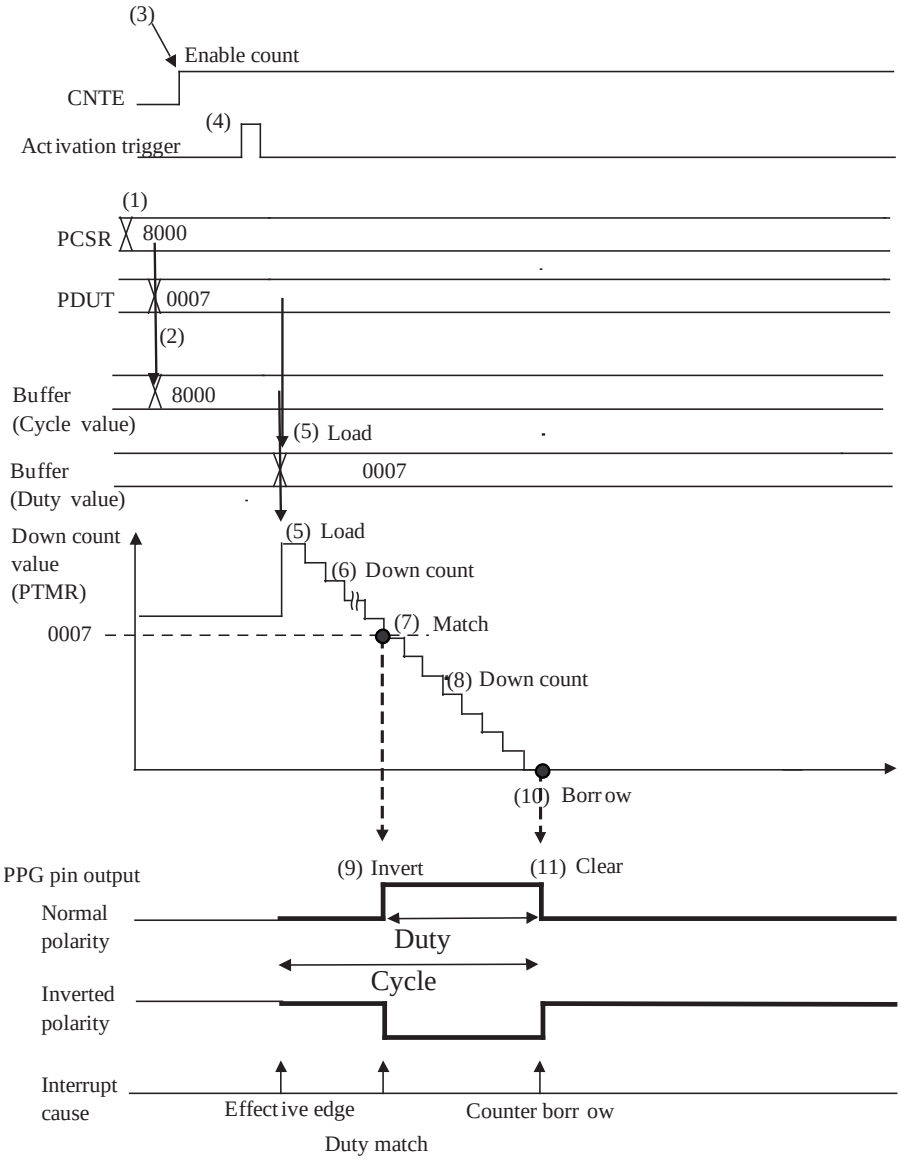
(1) Write a cycle value.
(2) Write a duty value and transfer the cycle value to buffers.
(3) Enable PPG operation.
(4) Generate an activation trigger.
(5) Load the cycle and duty values.
(6) Rewrite the duty value and transfer the cycle value to buffers.
(7) Counter down count.
(8) The down counter equals the duty value.
(9) Inverses the PPG pin output level.
(10) Counter down count.
(11) Counter borrow.
(12) Clear the PPG pin output level (return to normal).
(13) Reload the cycle value.
(14) Reload the duty value.
(15) Steps from (7) to (14) are iterated.

Equation:

- Period = {Period value (PCSR) + 1} × Count clock
- Duty = {Duty value (PDUT) + 1} × Count clock
- Width up to pulse output = {Period value (PCSR) – Duty value (PDUT)} × Count clock

■ One-Shot Operation

In one-shot operation, one-shot pulses are generated from the PPG pin. One-shot operation cannot be used in 8-bit mode or together with start delay feature.

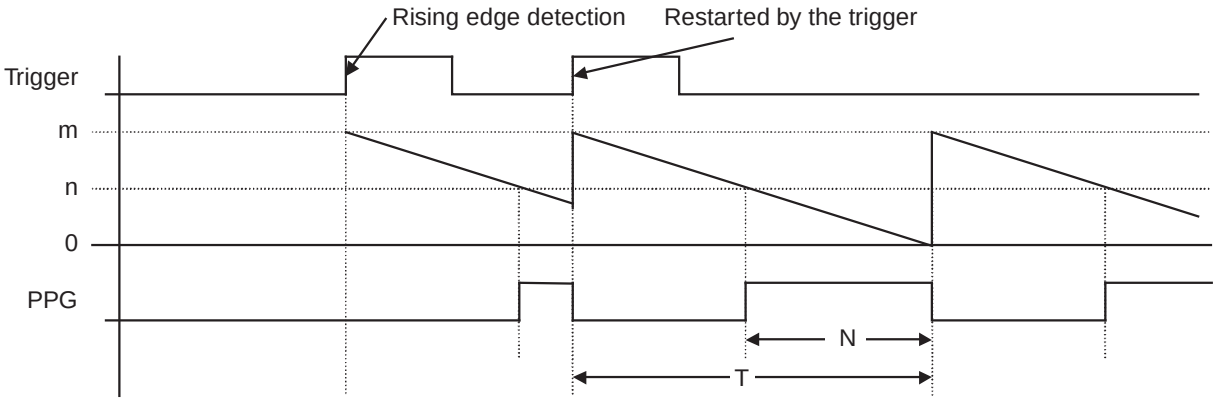


(1) Write a cycle value.
(2) Write a duty value and transfer the cycle value to buffers.
(3) Enable PPG operation.
(4) Generate an activation trigger.
(5) Load the cycle and duty values.
(6) Counter down count.
(7) The down counter equals the duty value.
(8) Inverses the PPG pin output level.
(9) Counter down count.
(10) Counter borrow.
(11) Clear the PPG pin output level (return to normal).
(12) The operating sequence is now completed

■ Restart Operation

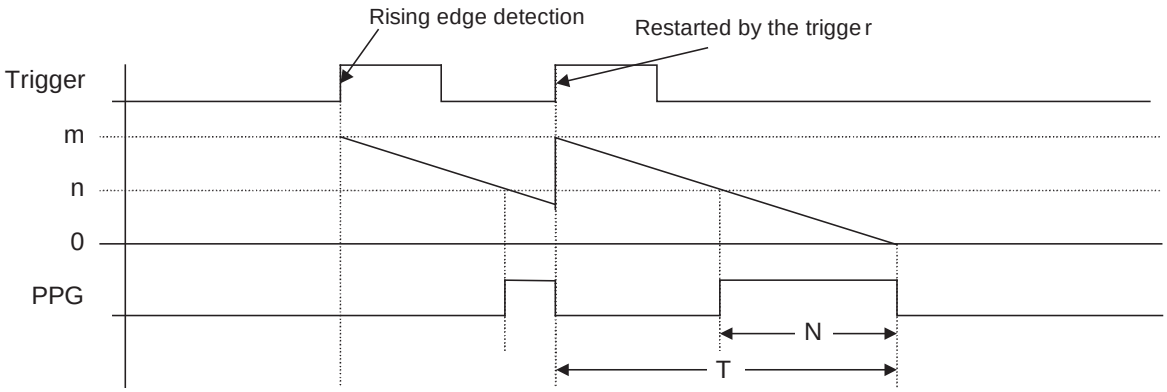
The restart operation is described below:

Restart available in PWM Operation:



N = duty, T = cycle

Restart available in One-shot Operation:

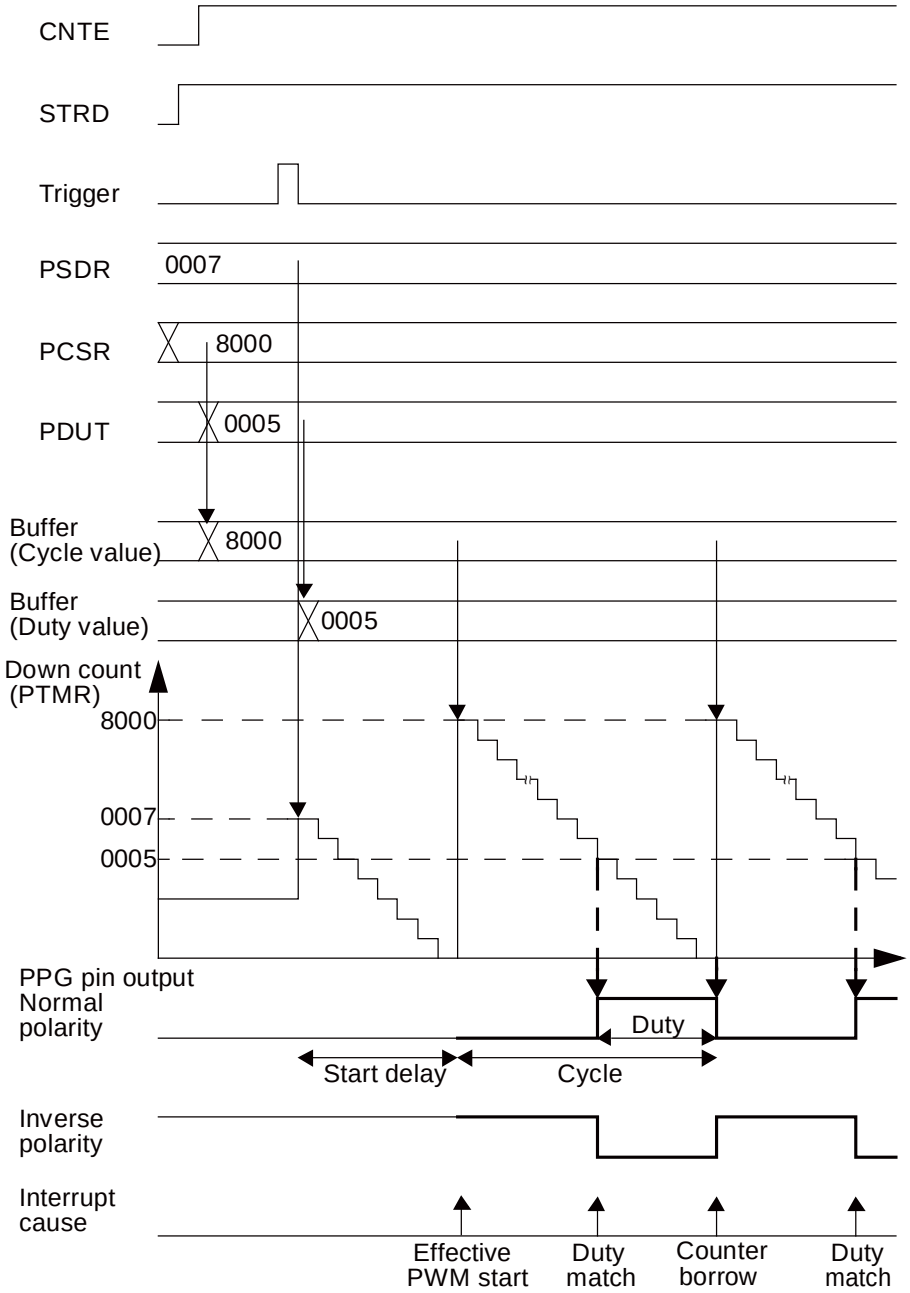


If a restart is not available, the second and subsequent triggers have no effect in both PWM and one-shot operations. (The second and subsequent triggers following a shutdown of the down counter are functional).

■ Start Delay Mode*

In start delay mode, generation of the PWM output is postponed by PSDR PPG count cycles.

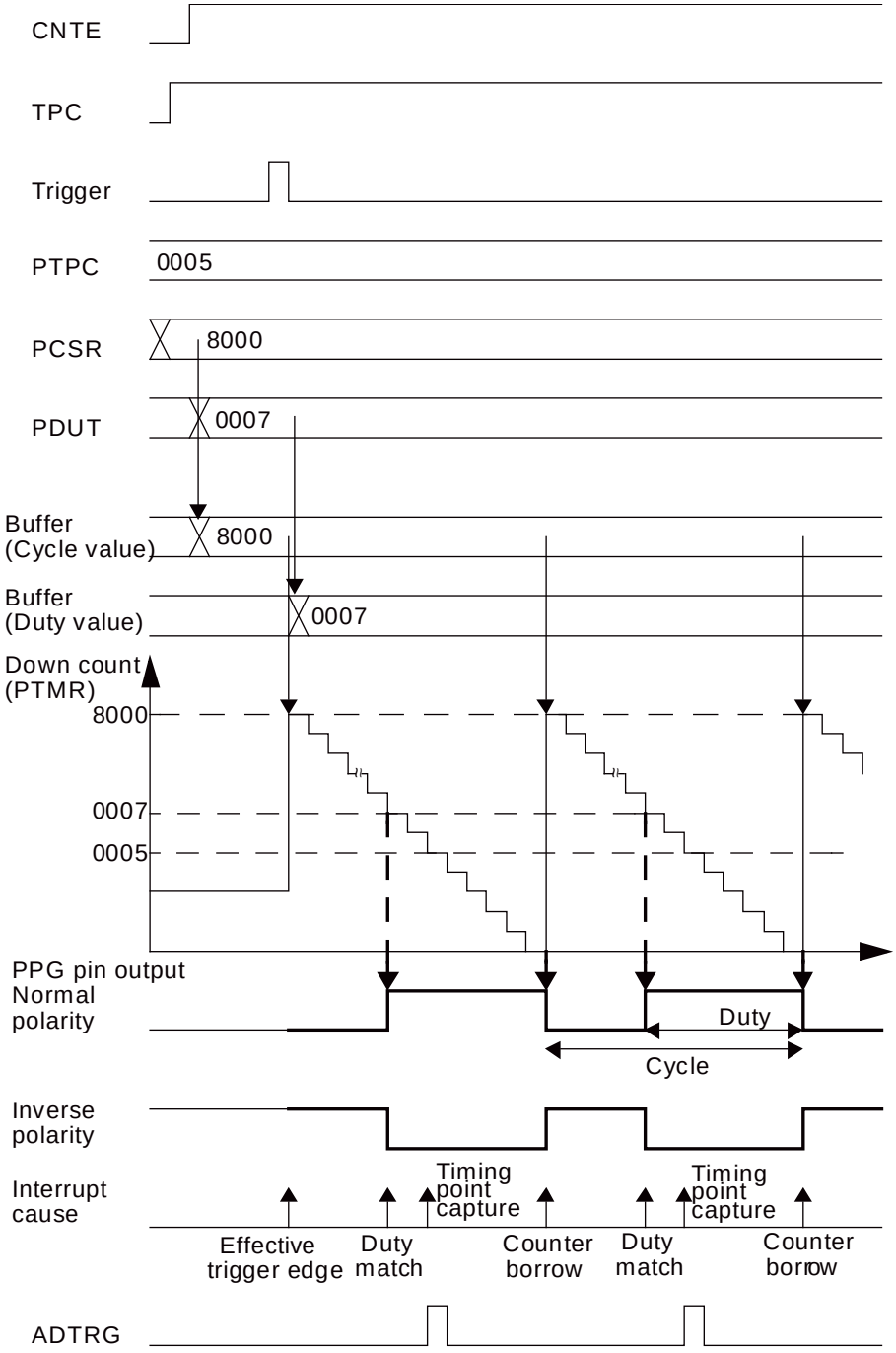
*: Because the start delay mode is optional, access is not possible on all devices. Please refer to the datasheet of the corresponding device.



■ Timing Point Capture*

In this operation mode, a timing point within PWM cycle can be defined to be used as possible interrupt request or ADC trigger signal.

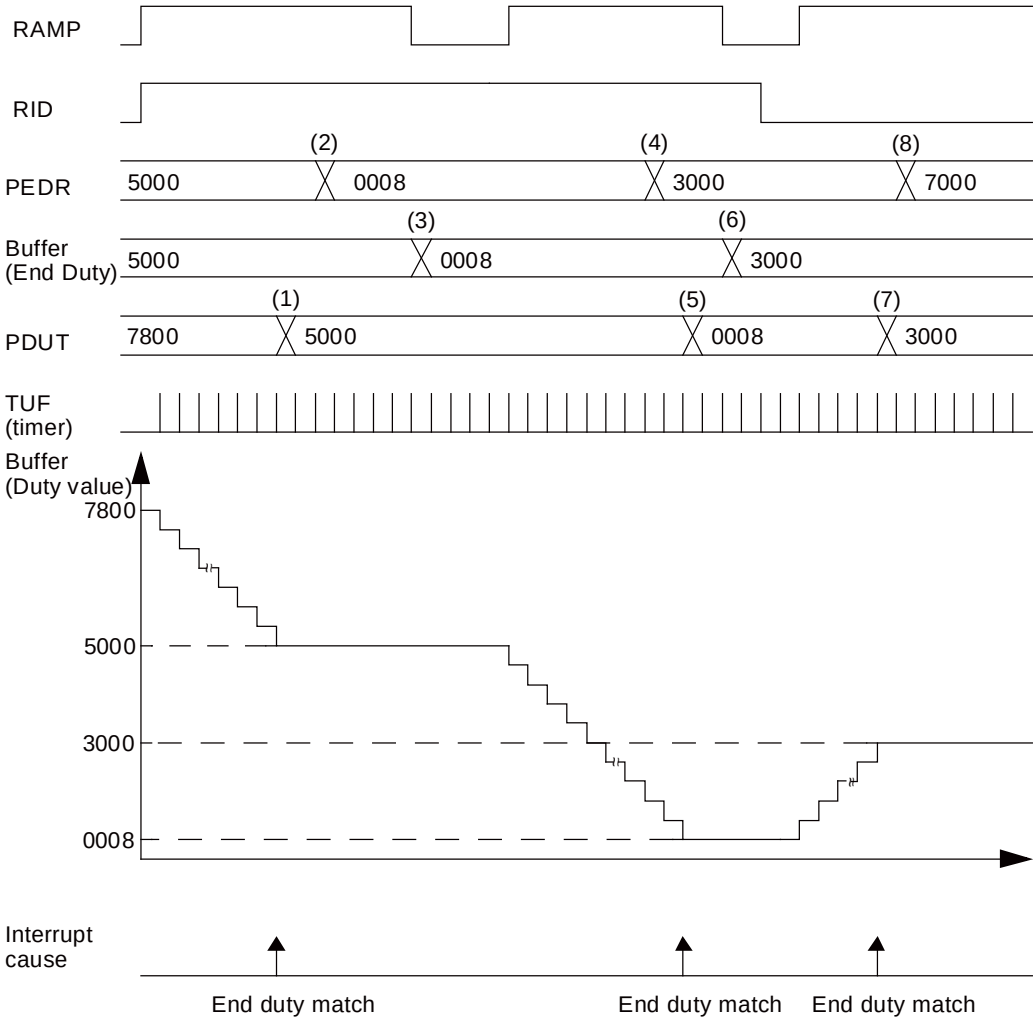
*: Because the timing point capture mode is optional, access is not possible on all devices. Please refer to the datasheet of the corresponding device.



■ Ramp Mode*

In the ramp mode PWM duty is incremented (PCN2:RIDH/RIDL="0") or decremented (PCN2:RIDH/RIDL="1") with every selected reload timer underflow pulse. PWM output waveform starts with the duty defined by PDUT register and the duty value is incremented/decremented until the end duty is reached.

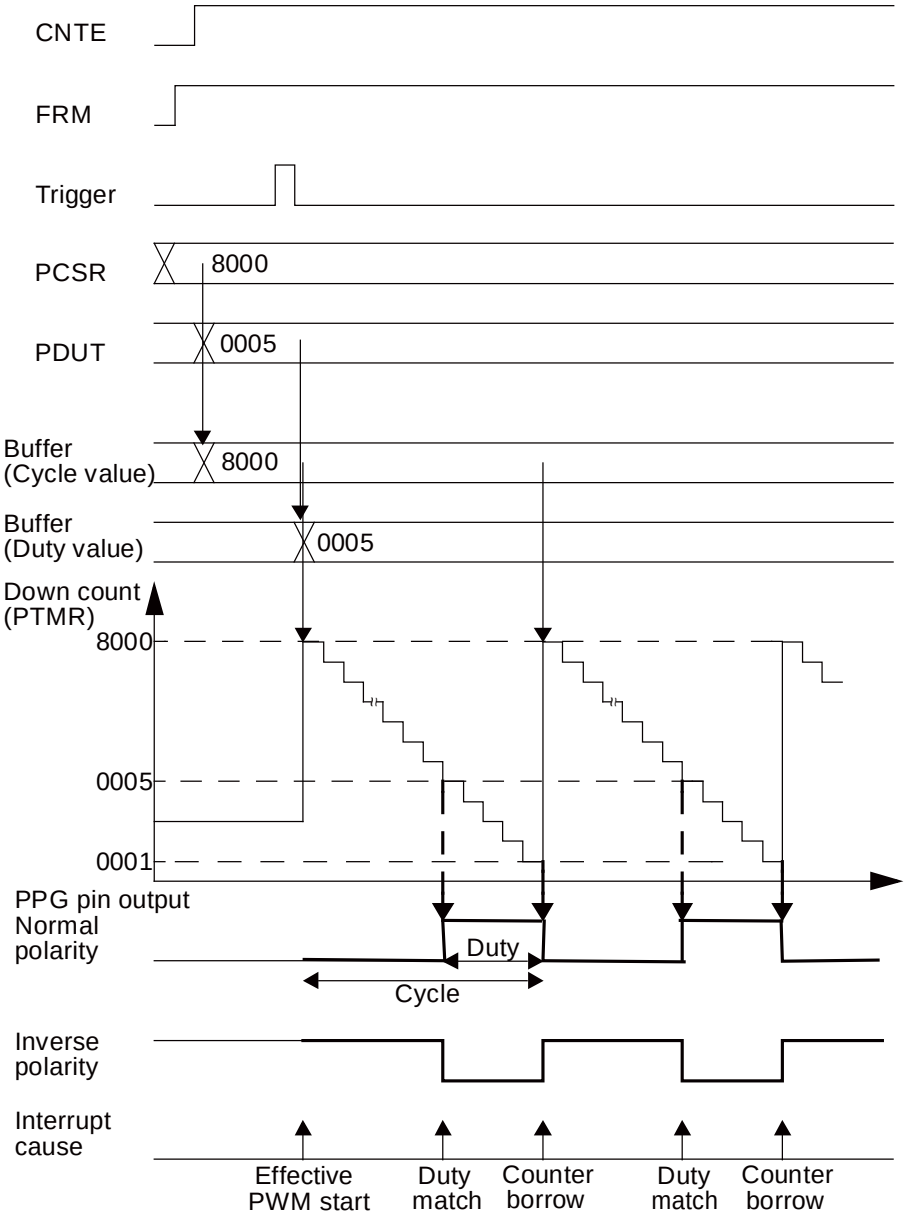
*: Because the ramp mode is optional, access is not possible on all devices. Please refer to the datasheet of the corresponding device.



(1) After the end duty is reached, PDUT register is updated to the end duty.
(2) New PEDR value is set by software.
(3) Since the ramp mode is disabled, PEDR value is transferred to the actual end duty.
(4) New PEDR value is set by software.
(5) End duty is reached, PDUT register is updated to the end duty.
(6) Ramp mode is disabled, hence PEDR value is transferred to the actual end duty.
(7) End duty is reached, PDUT register is updated to the end duty.
(8) New PEDR value is set by software, but it will become active (transferred to end duty buffer register) after the ramp mode is disabled.

■ Full Range Mode

PPG counter borrow happens at PTMR=1. As a consequence, for the setting PDUT=0 the PPG output pin stays "0" all the time and accordingly, for the setting PDUT=PCSR PPG output stays on high level during the whole operation time.



Equation:

- Period = Period value (PCSR) × Count clock
- Duty = Duty value (PDUT) × Count clock
- Width up to pulse output = {Period value (PCSR) – Duty value (PDUT)} × Count clock

3. Registers

This section explains the configuration and functions of the registers used for the Programmable Pulse Generator.

■ List of Programmable Pulse Generator Registers

Abbreviated Register Name	Register Name	Reference
PCNn	PPG Control Status register	See 3.1
EPCN1n	Extended PPG Control Status register 1	See 3.2
EPCN2n	Extended PPG Control Status register 2	See 3.3
GCN1g	General Control Register 1	See 3.4
GCN2g	General Control Register 2	See 3.5
GCN3g	General Control Register 3	See 3.6
GCN4g	General Control Register 4	See 3.7
GCN5g	General Control Register 5	See 3.8
PCSRn	PPG Cycle Setting Register	See 3.9
PDUTn	PPG Duty Setting Register	See 3.10
PTMRn	PPG Timer Register	See 3.11
PSDRn	PPG Start Delay Register	See 3.12
PTPCn	PPG Timing Point Capture register	See 3.13
PEDRn	PPG End Duty Register	See 3.14
GCNR	General Control Register	See 3.15

3.1. PPG Control Status Register (PCNn)

■ PPG Control Status Register (PCNn)

The PPG Control Status register (PCNn) controls the operations and status of the PPG.

PCNHn

bit	15	14	13	12	11	10	9	8
	CNTE	STGR	MDSE	RTRG	CKS1	CKS0	PGMS	MOD
Attribute	R/W	R0/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0
Rewrite during operation	O	O	X	X	X	X	X	X

PCNLn

bit	7	6	5	4	3	2	1	0
	EGS1	EGS0	IREN	IRQF	IRS1	IRS0	OE	OSEL
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0
Rewrite during operation	X	X	O	O	X	X	O	X

O : Rewritable
X : Not writable

[bit15] CNTE: Timer Enable Operation

Bit	Description
0	Stop
1	Operation

This bit enables the operation of the PPG.

[bit14] STGR: Software Trigger

Bit	Description
0	The operation is unaffected by writing (R0/W: The read value is always "0").
1	Software trigger activation

When the Software Trigger bit is set to "1", a software trigger is generated to activate the PPG, separately from the generation of an internal or external trigger (EN bit, reload timer output, TTG input). This trigger is independent from setting of the edge selection bits EGS1 and EGS0.

[bit13] MDSE: Mode Selection

Bit	Description
0	PWM operation
1	One-shot operation

When the Mode Selection bit is set to "0", a PWM operation is enabled to generate pulses in sequence.
When the Mode Selection bit is set to "1", pulse output takes place only once.

[bit12] RTRG: Restart Enable

Bit	Description
0	Disable restart.
1	Enable restart.

When the Restart Enable bit is set to "1", a trigger (software/internal/external) will restart the PPG operation (depending on the configuration of the triggers).

[bit11, bit10] CKS1, CKS0: Counter Flock Selection

bit11	bit10	Description
0	0	Clock selected by CKSEL
0	1	Clock selected by CKSEL divided by 4
1	0	Clock selected by CKSEL divided by 16
1	1	Clock selected by CKSEL divided by 64

[bit9] PGMS: PPG Output Mask Selection

Bit	Description
0	No output mask
1	Output mask (Output "L" level latched:OSEL="0")

When the PPG Output Mask Selection bit is set to "1", the PPG output can be clamped at "L" or "H" regardless of the mode, cycle, and duty settings.
The output level can be specified using the Output Polarity Specification bit (PCN:OSEL).

[bit8] MOD: PPG 16-bit/8-bit Operation Mode

Bit	Description
0	16-bit mode
1	8-bit mode

When the MOD bit is set to "0", a PWM output signal is defined with 16-bit resolution.
When the MOD bit is set to "1", 8-bit resolution is used for PWM output signal.

[bit7, bit6] EGS1, EGS0: Trigger Input Edge Selection

bit7	bit6	Description
0	0	no edge selected, triggering of PPG only possible by PCNHn:STRG
0	1	Rising edge
1	0	Falling edge
1	1	Both edges (rising edge, or, falling edge)

When EGS1=0 and EGS0=0, the PPG can only be triggered by PCNn:STRG. The triggers EN[3:0], Reload timer and external triggers are disabled for PPGn.
The other settings of EGS1 and EGS0 effect only the triggers EN[3:0], Reload timer and external triggers.
Writing "1" to PCNn:STRG triggers the corresponding PPG independent of the setting of PCNn:EGS1 and PCNn:EGS0.

[bit5] IREN: Interrupt Request Enable.

Bit	Description
0	Disable interrupt requests.
1	Enable interrupt requests.

[bit4] IRQF: Interrupt Request Flag

Bit	Description	
	Read	Write
0	No interrupt request	Clear the Interrupt Request flag.
1	Interrupt request	Writing "1" has no effect.

If the Interrupt Request flag (IRQF) equals "1" and writing "0" to the flag take place at the same time, the setting of the Interrupt Request flag (IRQF="1") by hardware has higher priority.
The read value is "1" regardless of the bit value when using a read modify write (RWM) instruction.

[bit3, bit2] IRS1, IRS0: Interrupt Cause Selection

EPCN1: IRS2	bit3	bit2	Description
0	0	0	Software trigger or external trigger input
0	0	1	Counter borrow (16-bit or both 8-bit parts if MOD="1")
0	1	0	The counter matches the duty value (16-bit or both 8-bit parts if MOD="1")
0	1	1	Counter borrow or the counter equals the duty value (16-bit or both 8-bit parts if MOD="1")
1	0	0	Timing point capture (16-bit or both 8-bit parts if MOD="1")
1	0	1	End duty match (16-bit or both 8-bit parts if MOD="1")

Together with IRS2 bit from EPCN1 register, the bits IRS1 and IRS0 select the operation in which to generate an interrupt request.

[bit1] OE: PPG Output Enable

Bit	Description
0	Output disabled
1	Output enabled

[bit0] OSEL: PPG Output Polarity Specification

Bit	Description
0	Normal polarity
1	Inverted polarity

When the PPG Output Mask Selection bit (PCN:PGMS) has been set to "1", if the Output Polarity Specification bit (OSEL) is set to "0", the output is clamped at "L"; if the Output Polarity Specification bit is set to "1", the output is clamped at "H".

3.2. Extended PPG Control Status Register 1 (EPCN1n)

■ Extended PPG Control Status Register 1 (EPCN1n)

The Extended PPG Control Status register 1 (EPCN1n) controls the operations and status of the PPG.

EPCN1Hn

bit	15	14	13	12	11	10	9	8
	-	-	WFGG	TRIG	IRS2	OE2	FRMH	FRML
Attribute	-	-	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	X	X	0	0	0	0	0	0
Rewrite during operation	-	-	X	O	X	O	X	X

EPCN1Ln

bit	7	6	5	4	3	2	1	0
	OSEL2	RIDH	RIDL	RAMPH	RAMPL	TPCH	TPCL	STRD
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0
Rewrite during operation	X	O	O	O	O	O	O	X

O : Rewritable
X : Not writable

[bit15, bit14] -: Undefined

Always write "0". The read value is undefined. Read modify write operations to this register have no effect on these bits.

[bit13] WFGG: Waveform Generator PPG Gating

Bit	Description
0	Waveform generator cannot gate the generation of the PWM output
1	Waveform generator can gate the generation of the PWM output

PWM output can be used and processed by Waveform Generator module. This is possible for both 16-bit and 8-bit PPG operation modes. However, in 8-bit PPG operation mode both PPG outputs (PPGA and PPGB) are gated and it is not possible to process only one PPG output by Waveform Generator and use the other PPG output for some other purpose.
To use this feature PPG enable/disable bit PCN:CNTE should stay at its inactive value "0". After the control bit WFGG is set to "1", Waveform Generator controls PPG operation through the GATE PPG input, that is used for both PPG triggering and PPG enabling/disabling. Accordingly, GATE input is added to possible PPG triggers, selectable via GCN1:TSEL control bits (GCN1:TSEL = "1100"). As a trigger, rising or falling edge of the GATE signals can be defined (via PCN:EGS control bits). Depending on the GATE signal edge used as trigger, GATE enabling/disabling level is accordingly defined (rising edge is trigger => high level enables PPG and the opposite). After the GATE signal changes to inactive level (PPG disabled), PPG output is always changed to the default level (the level before the PPG operation is enabled).

[bit12] TRIG: PPG Start/Trigger Event Flag

Bit	Description	
	Read	Write
0	No interrupt request	Clear the flag.
1	Interrupt request	Writing "1" has no effect.

TRIG flag is set to "1" when the PWM output generation is started. In start delay mode EPCN1:STRD = "1", it happens at the end of defined start delay. For all other operation modes (EPCN1:STRD = "0"), the flag becomes "1" when PPG trigger event is detected.
If the TRIG flag equals "1" and writing "0" to the flag take place at the same time, the setting of the flag (TRIG="1") by hardware has higher priority.
The read value is "1" regardless of the bit value when using a read modify write (RWM) instruction.

[bit11] IRS2 : Additional Interrupt Selection

IRS2	PCN: IRS1	PCN: IRS0	Description
0	0	0	Software trigger or external trigger input
0	0	1	Counter borrow (16-bit or both 8-bit parts if MOD="1")
0	1	0	The counter matches the duty value (16-bit or both 8-bit parts if MOD="1")
0	1	1	Counter borrow or the counter equals the duty value (16-bit or both 8-bit parts if MOD="1")
1	0	0	Timing point capture (16-bit or both 8-bit parts if MOD="1")
1	0	1	End duty match (16-bit or both 8-bit parts if MOD="1")

Together with PCN register bits IRS1 and IRS0, selects the operation in which to generate an interrupt request.

[bit10] OE2: PPGB Output Enable

Bit	Description
0	Output disabled
1	Output enabled

In 8-bit mode operation (PCN:MOD="1") the bit enables outputting of the PPGB signal (8-bit resolution PWM signal configured by higher 8-bit register parts).
In 16-bit operation mode (PCN:MOD="0") OE2 control bit can be set to "1" in order to generate 16-bit PPG signal also on PPGB output pin. If additionally EPCN1:OSEL2 bit is set to "1" 16-bit PPG signal with inverted polarity is sent on PPGB output pin.

[bit9] FRMH: Full Range Mode (High 8-bit part)

Bit	Description
0	PPGB output signal has period of PCSR+1 and duty of PDUT+1 count clock cycles
1	PPGB output signal has period of PCSR and duty of PDUT count clock cycles

When the bit is set to "1", the limit case PDUT=PCSR generates "H" on the PPGB output and for PDUT=0 "L" level is generated all time.

[bit8] FRML: Full Range Mode (16-bit or low 8-bit part)

Bit	Description
0	PPGA output signal has period of PCSR+1 and duty of PDUT+1 count clock cycles
1	PPGA output signal has period of PCSR and duty of PDUT count clock cycles

When the bit is set to "1", the limit case PDUT=PCSR generates "H" on the PPGA output and for PDUT=0 "L" level is generated all the time.

[bit7] OSEL2: PPGB Output Polarity Specification (High 8-bit part)

Bit	Description
0	PPGB output signal has normal polarity
1	PPGB output signal has inverted polarity

In 16-bit operation mode (PCN:MOD="0") if OE2 control bit is set to "1" the 16-bit PPG signal is also generated on PPGB output pin. If additionally OSEL2 bit is set to "1" 16-bit PPG signal with inverted polarity is sent on PPGB output pin.

[bit6] RIDH: Duty Increment/Decrement in Ramp Mode (High 8-bit part)

Bit	Description
0	Increment PWM duty of PPGB output in ramp mode until the end duty is reached
1	Decrement PWM duty of PPGB output in ramp mode until the end duty is reached

[bit5] RIDL: Duty Increment/Decrement in Ramp Mode (16-bit or low 8-bit part).

Bit	Description
0	Increment PWM duty of PPGA output in ramp mode until the end duty is reached
1	Decrement PWM duty of PPGA output in ramp mode until the end duty is reached

In 16-bit operation mode (PCN:MOD="0") RIDL bit controls duty increment/decrement function of 16-bit PWM signal and during 8-bit operation (PCN:MOD="1") it is dedicated to low 8-bit PWM output.

[bit4] RAMPH: Ramp Mode Selection (High 8-bit part).

Bit	Description
0	Disable ramp mode for PPGB output
1	Enable ramp mode for PPGB output

This bit enables ramp mode PWM operation for higher 8-bit part if PCN:MOD="1". In this mode PWM duty is incremented (PCN2 register bit RIDH is "0") or decremented (RIDH is "1") with every external trigger signal (selected one of seven possible reload timer underflow signals). PWM output waveform starts with the duty defined by PDUTH register and the duty value is incremented/decremented until the end duty value is reached, defined by the register PEDRH. Actual update of the PWM output duty takes place only at the end of PWM cycle.

[bit3] RAMPL: Ramp Mode Selection (16-bit or low 8-bit part).

Bit	Description
0	Disable ramp mode for PPGA output
1	Enable ramp mode for PPGA output

This bit enables ramp mode PWM operation for lower 8-bit part if PCN:MOD="1". During 16-bit operation, the bit controls ramp mode of 16-bit PWM. In this mode PWM duty is incremented (EPCN1 register bit RIDL is "0") or decremented (RIDL is "1") with every external trigger signal (selected one of seven possible reload timer underflow signals). PWM output waveform starts with the duty defined by PDUT/PDUTL register and the duty value is incremented/decremented until the end duty value is reached, defined by the register PEDR/PEDRL. Actual update of the PWM output duty takes place only at the end of PWM cycle.

[bit2] TPCH: Timing Point Capture Selection (High 8-bit part).

Bit	Description
0	Disable timing point capture mode for PPGB output
1	Enable timing point capture mode for PPGB output

If this bit is set to "1" and PCN:MOD="1", matching between PPG counter (higher 8-bit part) and PTPCH generates an ADC trigger signal.

[bit1] TPCL: Timing Point Capture Selection (16-bit or low 8-bit part).

Bit	Description
0	Disable timing point capture mode for PPGA output
1	Enable timing point capture mode for PPGA output

If this bit is set to "1" and PCN:MOD="1", matching between PPG counter (lower 8-bit part) and PTPCL generates an ADC trigger signal. Accordingly, during 16-bit operation ADC trigger signal is generated on the match between 16-bit PPG counter value and PTPC.

[bit0] STRD: Start Delay Mode

Bit	Description
0	Delayed start of PWM output generation is disabled
1	Delayed start of PWM output generation is enabled

When STRD bit is set to "1", the PWM output generation is delayed by PSDR + 1 cycles of the PPG counter clock. If the full range mode is activated (FRMH/FRML="1"), start is delayed for PSDR cycles of the PPG counter clock.

3.3. Extended PPG Control Status Register 2 (EPCN2n)

■ Extended PPG Control Status Register 2 (EPCN2n)

The Extended PPG Control/Status register 2 (EPCN2n) controls the operations and status of the PPG.

EPCN2Ln								
bit	7	6	5	4	3	2	1	0
	TCH	TCL	EDMH	EDML	DTH	DTL	PRDH	PRDL
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0
Rewrite during operation	O	O	O	O	O	O	O	O

O : Rewritable
X : Not writable

[bit7] TCH: Timing Point Capture Flag (High 8-bit part)

Bit	Description	
	Read	Write
0	No interrupt request	Clear the flag
1	Interrupt request	Writing "1" has no effect

The TCH flag is set to "1" when the PPG counter value PTMRH reaches the timing point defined by the PTPCH register. If the TCH flag equals "1" and writing "0" to the flag take place at the same time, the setting of the flag (TCH="1") by hardware has higher priority. In 16-bit operation mode (MOD="0") TCH flag has no meaning.
The read value is "1" regardless of the bit value when using a read modify write (RWM) instruction.

[bit6] TCL: Timing Point Capture Flag (16-bit or low 8-bit part)

Bit	Description	
	Read	Write
0	No interrupt request	Clear the flag
1	Interrupt request	Writing "1" has no effect

In 16-bit operation mode (PCN:MOD="0") TCL flag is set by matching of 16-bit registers PTMR and PTPC. If PCN:MOD="1", the flag is a result of comparing PTMRL and PTPCL. If the TCL flag equals "1" and writing "0" to the flag take place at the same time, the setting of the flag (TCL="1") by hardware has higher priority.
The read value is "1" regardless of the bit value when using a read modify write (RWM) instruction.

[bit5] EDMH: End Duty Match Flag in Ramp Mode (High 8-bit part)

Bit	Description	
	Read	Write
0	No interrupt request	Clear the flag
1	Interrupt request	Writing "1" has no effect

The EDMH flag is set to "1" when the current PWM duty has reached the duty value defined by the PEDRH register in ramp operation mode. If the EDMH flag equals "1" and writing "0" to the flag take place at the same time, the setting of the flag (EDMH="1") by hardware has higher priority. If the control bit EPCN1:RAMPH is "0" or in 16-bit operation mode (MOD="0") EDMH flag has no meaning.
The read value is "1" regardless of the bit value when using a read modify write (RWM) instruction.

[bit4] EDML: End Duty Match Flag in Ramp Mode (16-bit or low 8-bit part)

Bit	Description	
	Read	Write
0	No interrupt request	Clear the flag
1	Interrupt request	Writing "1" has no effect

In 16-bit operation mode (PCN:MOD="0") EDML flag is set by matching of the current duty value and PEDR. If PCN:MOD="1", the flag is a result of reaching the duty defined by PEDRL. If the EDML flag equals "1" and writing "0" to the flag take place at the same time, the setting of the flag (EDML="1") by hardware has higher priority. If the control bit EPCN1:RAMPL is "0" EDML flag has no meaning.
The read value is "1" regardless of the bit value when using a read modify write (RWM) instruction.

[bit3] DTH: Duty Match Flag (High 8-bit part)

Bit	Description	
	Read	Write
0	No interrupt request	Clear the flag
1	Interrupt request	Writing "1" has no effect

The DTH flag is set to "1" by matching of the registers PTMRH and PDUTH. If the DTH flag equals "1" and writing "0" to the flag take place at the same time, the setting of the flag (DTH="1") by hardware has higher priority. In 16-bit operation mode (MOD="0") DTH flag has no meaning.
The read value is "1" regardless of the bit value when using a read modify write (RWM) instruction.

[bit2] DTL: Duty Match Flag (16-bit or low 8-bit part)

Bit	Description	
	Read	Write
0	No interrupt request	Clear the flag
1	Interrupt request	Writing "1" has no effect

In 16-bit operation mode (PCN:MOD="0") DTL flag is set by matching of 16-bit registers PTMR and PDUT. If PCN:MOD="1", the flag is a result of comparing PTMRL and PDUTL. If the DTL flag equals "1" and writing "0" to the flag take place at the same time, the setting of the flag (DTL="1") by hardware has

higher priority.
The read value is "1" regardless of the bit value when using a read modify write (RWM) instruction.

[bit1] PRDH: Cycle Match Flag (High 8-bit part)

Bit	Description	
	Read	Write
0	No interrupt request	Clear the flag
1	Interrupt request	Writing "1" has no effect

The PRDH flag is set to "1" in case of PTMRH counter underflow. If the PRDH flag equals "1" and writing "0" to the flag take place at the same time, the setting of the flag (PRDH="1") by hardware has higher priority. In 16-bit operation mode (MOD="0") PRDH flag has no meaning.
The read value is "1" regardless of the bit value when using a read modify write (RWM) instruction.

[bit0] PRDL: Cycle Match Flag (16-bit or low 8-bit part)

Bit	Description	
	Read	Write
0	No interrupt request	Clear the flag
1	Interrupt request	Writing "1" has no effect

In 16-bit operation mode (PCN:MOD="0") PRDL flag is set by 16-bit PTMR counter underflow. If PCN:MOD="1", the flag is a result of PTMRL underflow. If the PRDL flag equals "1" and writing "0" to the flag take place at the same time, the setting of the flag (PRDL="1") by hardware has higher priority.
The read value is "1" regardless of the bit value when using a read modify write (RWM) instruction.

3.4. General Control Register 1 (GCN1g)

■ General Control Register 1 (GCN1g)

The General Control Register 1 (GCN1g) selects a trigger input to a PPG group of 4 PPGs.

GCN1Hg

bit	15	14	13	12	11	10	9	8
	TSEL33	TSEL32	TSEL31	TSEL30	TSEL23	TSEL22	TSEL21	TSEL20
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	1	1	0	0	1	0

GCN1Lg

bit	7	6	5	4	3	2	1	0
	TSEL13	TSEL12	TSEL11	TSEL10	TSEL03	TSEL02	TSEL01	TSEL00
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	1	0	0	0	0

In the following table, the index "i" denotes the PPG channel number within the group of PPGs (i = 0, 1, 2, or 3).

TSELi3	TSELi2	TSELi1	TSELi0	Activation trigger specification
0	0	0	0	EN0 bit (GCN2 register)
0	0	0	1	EN1 bit (GCN2 register)
0	0	1	0	EN2 bit (GCN2 register)
0	0	1	1	EN3 bit (GCN2 register)
0	1	0	0	16-bit reload timer 0 output
0	1	0	1	16-bit reload timer 1 output
0	1	1	0	CTG0 bit (GCNR register)
0	1	1	1	CTG1 bit (GCNR register)
1	0	0	0	External trigger (g + 0)
1	0	0	1	External trigger (g + 1)
1	0	1	0	External trigger (g + 2)
1	0	1	1	External trigger (g + 3)
1	1	0	0	Waveform Generator gatesignal
None of the above				Disabled

PPG0 to PPGn as selected are activated when the edge specified by the Trigger Input Edge Selection bits (PCNn:EGS[1:0]) are detected by the specified activation trigger.

3.5. General Control Register 2 (GCN2g)

■ General Control Register 2 (GCN2g)

The General Control Register 2 (GCN2g) generates internal trigger levels using software for a PPG block of 4 PPGs.

GCN2Hg								
bit	15	14	13	12	11	10	9	8
	-	-	-	-	CKSEL3	CKSEL2	CKSEL1	CKSEL0
Attribute	-	-	-	-	R/W	R/W	R/W	R/W
Initial value	X	X	X	X	0	0	0	0

GCN2Lg								
bit	7	6	5	4	3	2	1	0
	-	-	-	-	EN3	EN2	EN1	EN0
Attribute	-	-	-	-	R/W	R/W	R/W	R/W
Initial value	X	X	X	X	0	0	0	0

[bit15 to bit12] -: Undefined

Always write "0".
The read value is undefined.
Read modify write operations to this register have no effect on these bits.

[bit11 to bit8] CKSEL3 to CKSEL0: Prescalar Input Selection

Bit	Description
0	Use CLKP1 as prescaler input
1	Use RLT as prescaler input

[bit7 to bit4] -: Undefined

Always write "0".
The read value is undefined.
Read modify write operations to this register have no effect on these bits.

[bit3 to bit0] EN3 to EN0: EN Trigger Input

Bit	Description
0	Set the level to "L".
1	Set the level to "H".

Set the levels of internal triggers EN0, EN1, EN2 and EN3.
If any of the EN trigger inputs (EN0, EN1, EN2, EN3) is selected with the trigger specification bits (GCN1g:TSEL0[3:0], GCN1g:TSEL1[3:0], GCN1g:TSEL2[3:0], and GCN1g:TSEL3[3:0]) of PPGn, then the selected EN serves as a PPG trigger input bit.
If the state selected with the trigger input edge selection bit (EGS[1:0]) is generated by software using the trigger input bit (selected EN0, EN1, EN2, or EN3), the choice serves as an activation trigger to activate the PPG.

3.6. General Control Register 3 (GCN3g)

■ General Control Register 3 (GCN3g)

The General Control Register 3 (GCN3g) selects a trigger input for the ramp mode duty increment/decrement operation to a PPG group of 4 PPGs. One of seven reload timer underflow signals can be selected.

GCN3Hg

bit	15	14	13	12	11	10	9	8
	-	RTG32	RTG31	RTG30	-	RTG22	RTG21	RTG20
Attribute	-	R/W	R/W	R/W	-	R/W	R/W	R/W
Initial value	X	0	0	0	X	0	0	0

GCN3Lg

bit	7	6	5	4	3	2	1	0
	-	RTG12	RTG11	RTG10	-	RTG02	RTG01	RTG00
Attribute	-	R/W	R/W	R/W	-	R/W	R/W	R/W
Initial value	X	0	0	0	X	0	0	0

[bit15, bit11, bit7, bit3] -: Undefined

Always write "0".
The read value is undefined.
Read modify write operations to this register have no effect on these bits.
In the following table, the index "i" denotes the PPG channel number within the group of PPGs (i = 0, 1, 2, or 3).

[bit14 to bit12, bit10 to bit8, bit6 to bit4, bit2 to bit0] RTG32 to RTG30, RTG22 to RTG20, RTG12 to RTG10, RTG02 to RTG00: Ramp Mode Duty Increment/Decrement Trigger Selection

RTGi2	RTGi1	RTGi0	Trigger specification
0	0	0	Reload timer 0 underflow
0	0	1	Reload timer 1 underflow
0	1	0	Reload timer 2 underflow
0	1	1	Reload timer 3 underflow
1	0	0	Reload timer 4 underflow
1	0	1	Reload timer 5 underflow
1	1	0	Reload timer 6 underflow
None of the above			Reload timer 6 underflow

3.7. General Control Register 4 (GCN4g)

■ General Control Register 4 (GCN4g)

The General Control Register 4 (GCN4g) selects a reload timer input that can be used as PPG count clock to a PPG group of 4 PPGs. One of seven reload timer underflow signals can be selected. The chosen reload timer input becomes the PPG count clock (prescaler input), only if the bit GCN2g:CKSELi is set to "1".

GCN4Hg

bit	15	14	13	12	11	10	9	8
	-	RCK32	RCK31	RCK30	-	RCK22	RCK21	RCK20
Attribute	-	R/W	R/W	R/W	-	R/W	R/W	R/W
Initial value	X	1	1	0	X	1	1	0

GCN4Lg

bit	7	6	5	4	3	2	1	0
	-	RCK12	RCK11	RCK10	-	RCK02	RCK01	RCK00
Attribute	-	R/W	R/W	R/W	-	R/W	R/W	R/W
Initial value	X	1	1	0	X	1	1	0

[bit15, bit11, bit7, bit3] Undefined

Always write "0".
The read value is undefined.
Read modify write operations to this register have no effect on these bits.
In the following table, the index "i" denotes the PPG channel number within the group of PPGs (i = 0, 1, 2, or 3).

[bit14 to bit12, bit10 to bit8, bit6 to bit4, bit2 to bit0] RCK32 to RCK30, RCK22 to RCK20, RCK12 to RCK10, RCK02 to RCK00: Reload Timer Selection as Possible PPG Count Clock

RCKi2	RCKi1	RCKi0	Reload timer selected
0	0	0	Reload timer 0 underflow
0	0	1	Reload timer 1 underflow
0	1	0	Reload timer 2 underflow
0	1	1	Reload timer 3 underflow
1	0	0	Reload timer 4 underflow
1	0	1	Reload timer 5 underflow
1	1	0	Reload timer 6 underflow
None of the above			Reload timer 6 underflow

3.8. General Control Register 5 (GCN5g)

■ General Control Register 5 (GCN5g)

The General Control Register 5 (GCN5g) introduces a fine delay (phase shift) of a reload timer input that can be used as PPG count clock. The phase shift can be defined separately for every PPG module within a group of 4 PPGs. If this feature is used, the selected reload timer should be set to a reload cycle value of at least t 13 peripheral clock cycles (13 CLKP).

GCN5Lg

bit	7	6	5	4	3	2	1	0
	RSH31	RSH30	RSH21	RSH20	RSH11	RSH10	RSH01	RSH00
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

In the following table, the index "i" denotes the PPG channel number within the group of PPGs (i = 0, 1, 2 or 3).

[bit7 to bit6, bit5 to bit4, bit3 to bit2, bit1 to 0] RSH31 to 30, RSH21 to 20, RSH11 to 10, RSH01 to 00: Fine Delay of Reload Timer Signal that can be used as PPG Count Clock

RSHi1	RSHi0	Fine delay
0	0	No delay
0	1	4 CLKP cycles delay
1	0	8 CLKP cycles delay
1	1	12 CLKP cycles delay

3.9. PPG Cycle Setting Register (PCSRn)

■ PPG Cycle Setting Register (PCSRn)

The PPG Cycle Setting Register (PCSRn) controls the cycle of the PPG. In 8-bit operation mode PCSRH defines the cycle of the PPGB output and PCSRL sets the cycle of the PPGA signal.

PCSRHn								
bit	15	14	13	12	11	10	9	8
	D15	D14	D13	D12	D11	D10	D9	D8
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	X	X	X	X	X	X	X	X

PCSRLn								
bit	7	6	5	4	3	2	1	0
	D7	D6	D5	D4	D3	D2	D1	D0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	X	X	X	X	X	X	X	X

The PPG Period Setting registers come with buffers. Transfers from the buffers to the counter take place automatically upon counter borrow.
After the PPG Period Setting registers have been written, be sure to set PPG Duty Setting registers PDUT.
When the register is read, the buffered value, i. e. actual cycle setting is read.

3.10.PPG Duty Setting Register (PDUTn)

■ PPG Duty Setting Register (PDUTn)

The PPG Duty Setting Register (PDUTn) sets the duty of the PPG output waveform. In 8-bit operation mode PDUTH defines the duty of the PPGB output and PDUTL sets the duty of the PPGA signal.

PDUTHn

bit	15	14	13	12	11	10	9	8
	D15	D14	D13	D12	D11	D10	D9	D8
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	X	X	X	X	X	X	X	X

PDUTLn

bit	7	6	5	4	3	2	1	0
	D7	D6	D5	D4	D3	D2	D1	D0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	X	X	X	X	X	X	X	X

The PPG Duty Setting Registers are buffered. Transfers from the buffers to the counter take place automatically upon counter borrow. When the register is read, the buffered value, i. e. actual duty setting is read.
Set a value smaller than the setting of PPG Period Setting register PCSR in a PPG Duty Setting register. If the same value is set in a PPG Duty Setting register as is set in PPG Period Setting register PCSR, then

- "H" is always output to (OSEL="0") at normal polarity time.
- "L" is always output to (OSEL="1") at inverted polarity time.
  (The OSEL bit is an output polarity specification bit of the PPG control register PCN.)

3.11.PPG Timer Register (PTMRn)

■ PPG Timer Register (PTMRn)

The PPG Timer Register (PTMRn) reads the counts of PPGn. In 8-bit operation mode PTMRH defines the PPG counter dedicated to the PPGB output and PTMRL reflects the PPG counter dedicated to the PPGA signal.

PTMRHn								
bit	15	14	13	12	11	10	9	8
	D15	D14	D13	D12	D11	D10	D9	D8
Attribute	R	R	R	R	R	R	R	R
Initial value	1	1	1	1	1	1	1	1

PTMRLn								
bit	7	6	5	4	3	2	1	0
	D7	D6	D5	D4	D3	D2	D1	D0
Attribute	R	R	R	R	R	R	R	R
Initial value	1	1	1	1	1	1	1	1

The count of the 16-bit down counter can be read.

3.12.PPG Start Delay Register (PSDRn)

■ PPG Start Delay Register (PSDRn)

The PPG Start Delay Register (PSDRn) controls delay of PWM output generation by PSDR+1 cycles of PPG counter clock, when the PCN2:STRD bit is set to "1". In the full range mode PWM output delay is PSDR PPG count cycles. In 8-bit operation mode PSDRH defines the delay of the PPGB output and PSDRL sets the delay of the PPGA signal.

PSDRHn

bit	15	14	13	12	11	10	9	8
	D15	D14	D13	D12	D11	D10	D9	D8
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

PSDRLn

bit	7	6	5	4	3	2	1	0
	D7	D6	D5	D4	D3	D2	D1	D0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

PSDR value should be set during PPG module operation is disabled PCN:CNTE="0".

3.13.PPG Timing Point Capture Register (PTPCn)

■ PPG Timing Point Capture Register (PTPCn)

The PPG Timing Point Capture Register (PTPCn) sets the timing point within PWM cycle, that can be captured to generate an interrupt request or an ADC trigger if adequate control bits EPCN1:TPCH/TPCL are set to "1". In 8-bit operation mode PTPCH defines the timing point for the PPGB output and PTPCL sets the timing point for the PPGA signal.

PTPCHn

bit	15	14	13	12	11	10	9	8
	D15	D14	D13	D12	D11	D10	D9	D8
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

PTPCLn

bit	7	6	5	4	3	2	1	0
	D7	D6	D5	D4	D3	D2	D1	D0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

3.14.PPG End Duty Register (PEDRn)

■ PPG End Duty Register (PEDRn)

The PPG End Duty Register (PEDRn) sets the end point of the PWM duty ramp in ramp operation mode EPCN1:RAMPH/RAMPL="1". In 8-bit operation mode PEDRH defines the end duty of the PPGB output and PEDRL sets the end duty of the PPGA signal.

PEDRHn

bit	15	14	13	12	11	10	9	8
	D15	D14	D13	D12	D11	D10	D9	D8
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

PEDRLn

bit	7	6	5	4	3	2	1	0
	D7	D6	D5	D4	D3	D2	D1	D0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

In the ramp mode PWM duty is incremented (EPCN1:RIDH/RIDL="0") or decremented (EPCN1:RIDH/RIDL="1") with every selected reload timer underflow pulse. PWM output waveform starts with the duty defined by PDUT register and the duty value is incremented/decremented until the end duty is reached.

3.15.General Control Register (GCNR)

■ General Control Register (GCNR)

The General Control Register (GCNR) is common for all PPG modules. Two bits CTG0 and CTG1 can be used as common triggers for many or all on-chip available PPG modules.

GCNR								
bit	7	6	5	4	3	2	1	0
	-	-	-	-	-	-	CTG1	CTG0
Attribute	-	-	-	-	-	-	R/W	R/W
Initial value	X	X	X	X	X	X	0	0

[bit7 to bit2] -: Undefined

Always write "0".
The read value is undefined.
Read modify write operations to this register have no effect on these bits.

[bit1 to bit0] CTG1/0: Common Trigger Input

Bit	Description
0	Set the level to "L"
1	Set the level to "H"

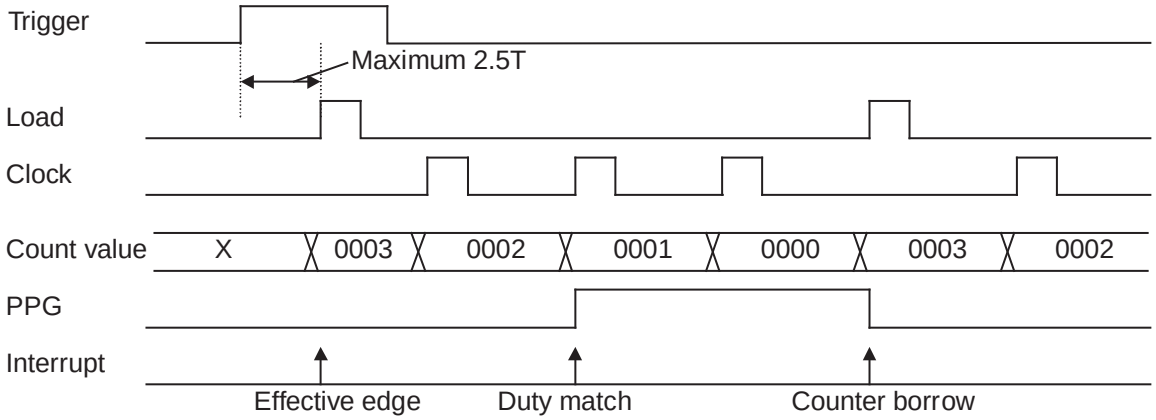
Set the levels of common triggers CTG0 and CTG1.
If any of the CTG trigger inputs (CTG0, CTG1) is selected with the trigger specification bits (GCN1g:TSEL0[3:0], GCN1g:TSEL1[3:0], GCN1g:TSEL2[3:0], and GCN1g:TSEL3[3:0]) of PPGn, then the selected CTG serves as a PPG trigger input bit.
If the state selected with the trigger input edge selection bit (EGS[1:0]) is generated by software using the common trigger bit (selected CTG0 or CTG1), the choice serves as an activation trigger to activate the PPG.

4. Cautions

This section describes the cautions to be considered while using the programmable pulse generator

■ Cautions

- If the Interrupt Request flag (PCN:IRQF) equals "1" and the Interrupt Request flag is set to "0" at the same timing, the setting of the Interrupt Request flag to "1" overrides the flag clear request.
- The first load has a maximum delay of 2.5T after the activation trigger. (T: Count clock)
  If the down counter is loaded and counts at the same time, the load operation overrides.



- Be sure to write duty value PDUT after cycle PCSR has been initialized and rewritten.
  (Always write in the order of (1) PCSR and (2) PDUT).
  Only the PDUT can be written for rewriting the duty.
- The duty value PDUT must be equal or smaller than the cycle value PCSR. If any larger value has been set, disable the operation of the PPG before replacing the duty value with a smaller value.
- To activate a PPG, it is necessary to set the Timer Operation Enable bits (PCN:CNTE) to "1" before or concurrently with the activation to enable the PPG operation.
- The values of mode (MDSE), restart enable (RTRG), count clock (CKS[1:0]), trigger input edge (EGS[1:0]), interrupt cause (IRS), internal trigger (TSEL), and output polarity specification (OSEL) may not be changed while the PPG is operating.
  If any of these values has been changed while the PPG was operating, disable the operation of the PPG before reloading the register.
- Whenever writing a value to GCN2, be sure to write "0" to any undefined part of the upper 4 bits.
  If "1" is written, disable the operation of the PPG before reloading the register.
- If Activation Trigger Specification bits (TSEL0[3:0]), (TSEL1[3:0]), (TSEL2[3:0]), (TSEL3[3:0]) have been set to any value outside the specified range (1101 - 1111), disable the operation of the PPG and then write the specified value to let the register return to normal.
- If the Timer Operation Enable bit (PCN:CNTE) is set to "0" to disable PPGn while it is operating, the PPG stops with its counter status being latched, but PPG output level goes after two machine cycles to the level defined with OSEL bit (default value of OSEL bit is '0').
- If the Timer Operation Enable bit(PCN:CNTE) is subsequently set to "1" to enable the PPG, PPG counter restarts from the point of interruption.
- One-shot pulse can be generated only in 16-bit operation mode and without start delay feature.

CHAPTER: EXTERNAL INTERRUPTS

This chapter explains the functions and operations of the External Interrupts.

Management Code: 96F3EXTINT-E01.0

1. Overview

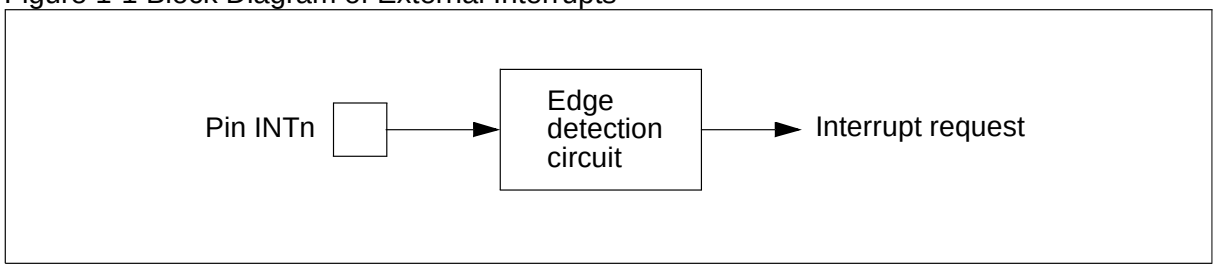
The External Interrupt detects a signal input to an external interrupt pin and generates an interrupt request.

■ External Interrupts

For an external interrupt request, four request levels are available: "H", "L", rising edge, and falling edge are available.

■ Block Diagram of External Interrupts

Figure 1-1 Block Diagram of External Interrupts



2. Registers

This section explains the configuration and functions of the registers used for the External Interrupt.

■ List of External Interrupt Registers

Abbreviated Register Name	Register Name	Reference
ENIRn	Interrupt Enable Register (External Interrupt Request Enable Register)	See 2.1
EIRRn	Interrupt flags (External Interrupt Request Register)	See 2.2
ELVRn	Request level setting register (External Interrupt Level Register)	See 2.3

2.1. Interrupt Enable Register (ENIRn: External Interrupt Request Enable Register)

■ Interrupt Enable Register (ENIRn: External Interrupt Request Enable Register)

ENIR0								
bit	7	6	5	4	3	2	1	0
	EN7	EN6	EN5	EN4	EN3	EN2	EN1	EN0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

ENIR1								
bit	7	6	5	4	3	2	1	0
	EN15	EN14	EN13	EN12	EN11	EN10	EN9	EN8
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

The ENIRn register contains the interrupt enable bits of the external interrupt module. When an ENx bit is enabled, an interrupt request is signaled if the specified interrupt event is detected at the corresponding pin. Writing "1" to these bits enables and writing "0" disables interrupt requests.

Note:

When an interrupt should be enabled, please also set the corresponding port’s Port Input Enable Register PIER (please refer to "Port Input Enable Register (PIERnn)"of "Section 11.2 I/O Port Registers" in "CHAPTER 11 I/O PORTS").

2.2. Interrupt Flags (EIRRn: External Interrupt Request Register)

■ Interrupt Flags (EIRRn: External Interrupt Request Register)

EIRR0								
bit	15	14	13	12	11	10	9	8
	ER7	ER6	ER5	ER4	ER3	ER2	ER1	ER0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

EIRR1								
bit	15	14	13	12	11	10	9	8
	ER15	ER14	ER13	ER12	ER11	ER10	ER9	ER8
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

The EIRRn register contains the interrupt flags.
If the specified interrupt event is detected at a pin, the corresponding ERx bit is set and if the ENx bit is "1", then an interrupt request is signaled to the interrupt controller. Writing "1" to these bits has no effect and writing "0" clears the flags. "1" is always read from these bits by the read-modify-write instructions.

Note:

When clearing the interrupt flag, make sure to clear the flag which caused the interrupt alone but not others.

2.3. Request Level Setting Register (ELVRn: External Interrupt Level Register)

■ Request Level Setting Register (ELVRn: External Interrupt Level Register)

ELVR0								
bit	7	6	5	4	3	2	1	0
	LB3	LA3	LB2	LA2	LB1	LA1	LB0	LA0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

bit	15	14	13	12	11	10	9	8
	LB7	LA7	LB6	LA6	LB5	LA5	LB4	LA4
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

ELVR1								
bit	7	6	5	4	3	2	1	0
	LB11	LA11	LB10	LA10	LB9	LA9	LB8	LA8
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

bit	15	14	13	12	11	10	9	8
	LB15	LA15	LB14	LA14	LB13	LA13	LB12	LA12
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

ELVRn defines the request event at the external pin. Each pin is assigned with two bits as described in Table 2-1. If a request is detected by the input level, the interrupt flag is set as long as the input is at the specified level even after the flag is reset by software.

Table 2-1 Interrupt Request Detection Factor for External Pins

LBx	LAx	Interrupt request detection factor
0	0	L level pin input
0	1	H level pin input
1	0	Rising edge pin input
1	1	Falling edge pin input

3. Notes on Using the External Interrupt Functions

The following points must be considered to use the External Interrupt function.
- Conditions on the behavior of external circuit for use of DMA
- Clearing interrupt flag
- External interrupt request level

■ Conditions on the Behavior of External Circuit for Use of DMA

- An external circuit which uses DMA must be able to inactivate the request signal when the requested DMA transfer is performed. For the correct usage of external interrupts to start DMA transfers please refer to application note mcu-an-300203-e.

■ Clearing Interrupt Flag

- When it is used as an external interrupt the interrupt flag must be cleared within the interrupt service routine. Otherwise the same service is performed again after the completion of the first interrupt service.
- When level detection is specified as the event input, the interrupt flag is set again even after it is cleared as long as the active level is kept at the input pin. In this case, the external cause of the request should be cleared or the interrupt enable bit should be cleared.

■ External Interrupt Request Level

- When edge detection is specified as the event input, the pulse of the input signal must have a minimum width to be recognized as an input edge and not be filtered by the noise filter. See datasheet of the specific device for the minimum pulse width length.
- When level detection is specified as the event input, the interrupt flag keeps active status once the specified level is input even after the input signal changes to the inactive level as shown in Figure 3-2. In order to clear the request, the interrupt flag must be cleared.

Figure 3-1 Clearing Interrupt Cause Register upon Level Set

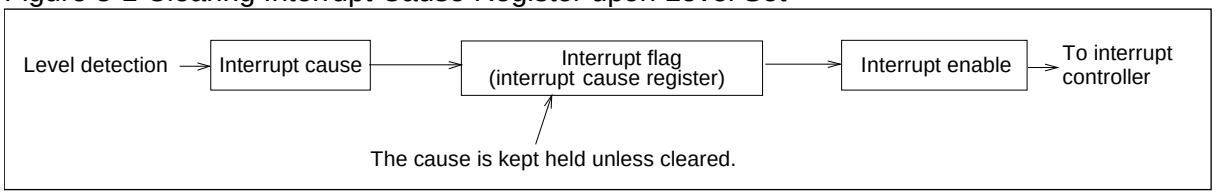
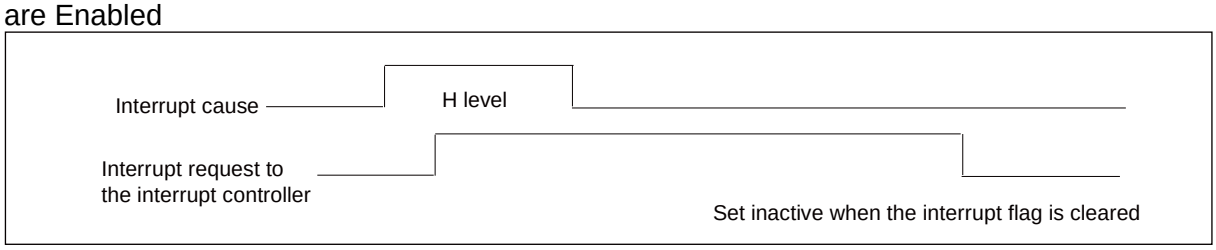


Figure 3-2 Interrupt Cause and Interrupt Request to the Interrupt Controller while Interrupts are Enabled



CHAPTER: A/D Converter

This chapter describes the functionality and the operation of the A/D converter.

Management Code: FS09-E02.1

1. Overview

The A/D converter converts analog input voltages into digital values.

■ Features of A/D Converter

- Conversion time: Refer to the datasheet.
- RC successive approximation conversion with sample and hold circuit.
- Resolution: 10-bit mode and 8-bit mode are available.
- Software selectable analog input channels
  Single-channel Conversion: One channel is selected and converted
  Scan Conversion: Multiple channels are converted sequentially

- Mode setting
  Single Mode: Converts each of the specified analog input channels once
  Continuous Mode: Converts the specified analog input channels repeatedly
  Pause Mode: Converts the specified analog input channels repeatedly, but requires reactivation for every channel (starting of conversion can be synchronized)

- Interrupt requests
  There are four types of interrupt requests:

  - INT interrupt request: With ADCS:INTE="1", an interrupt is requested when the A/D conversion result is transferred to ADCR register.
  - BLRF interrupt request: With ADDBCR:BLRIE="1", an interrupt is requested when buffer level limit configured by ADDBCR:BLL[2:0] is reached.
  - RCOINT interrupt request: With ADCC:RCOIE="1", ADCC:RCOE="1", an interrupt is requested when either of conditions below is met:
    1. ADRCOIRS:RCOIRS="0", and the A/D conversion result lies outside of the range defined by the upper/lower threshold
    2. ADRCOIRS:RCOIRS="1", and the A/D conversion result lies inside the range defined by the upper/lower threshold
  - CTPZF interrupt request: With ADPCIE:CTPIE="1", an interrupt is requested when ADC positive pulse counter decrements from "0x01" to "0x00".

  INT and BLRF interrupt requests can trigger the interrupt controller to activate DMA and transfer the A/D conversion result to memory.

- Selectable ADC activation method
  The A/D converter can be activated either by software, reload timer, or external trigger (falling edge), depending on the configuration of ADCS:STS1-0.

- Supported analog input channels
  Up to 32 analog input channels are supported.

- Data Buffer Function
  The data buffer can be configured to use up to eight data registers.

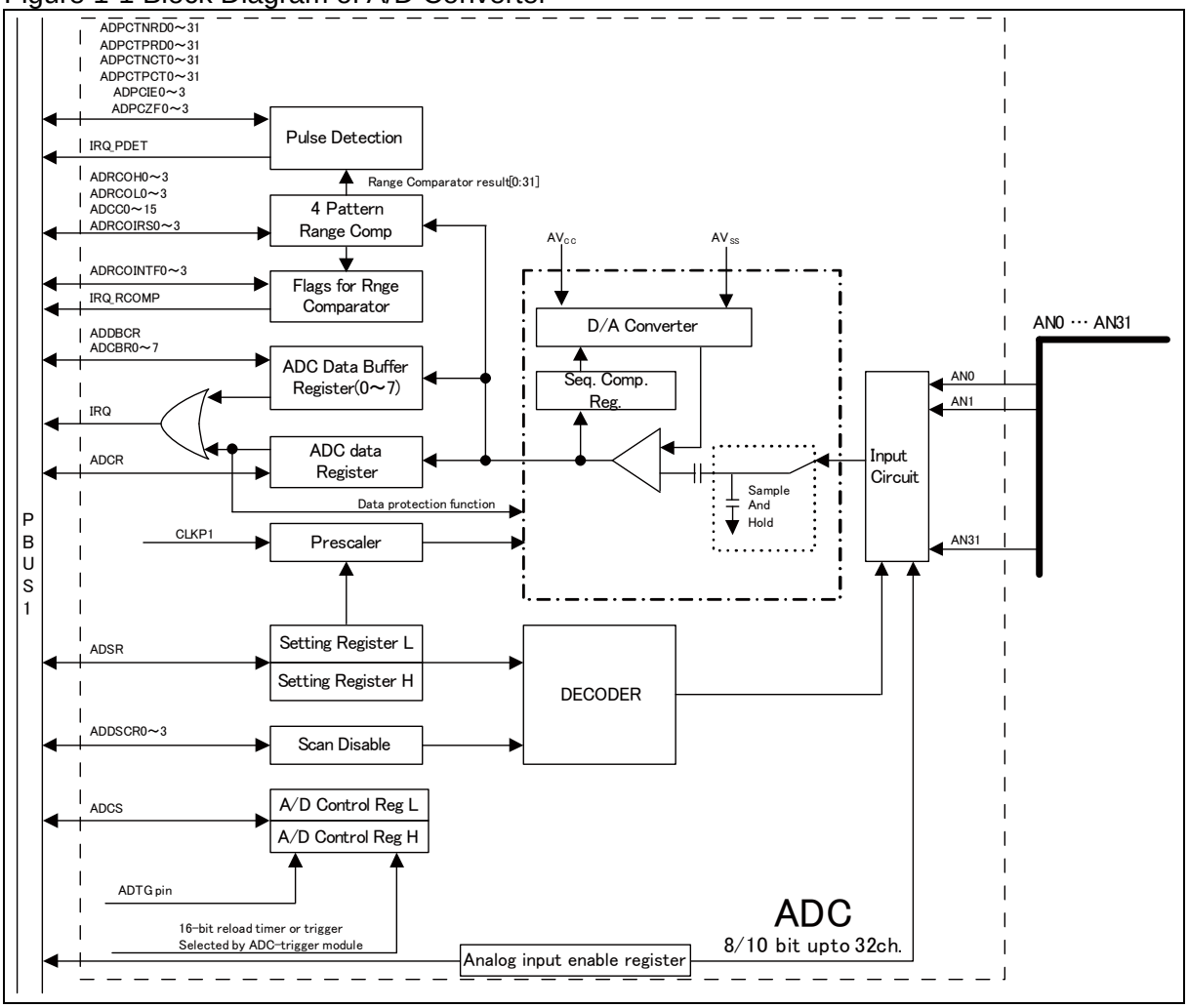
- Range Comparator Function
  The upper/lower threshold for each analog input channel is selected from four register values which are configurable.

- Scan Disable Function
  During Scan Conversion, this function enables the A/D converter to skip the unused analog input channels.

- ADC Pulse Detection Function
  This function is used to detect a pulse of "positive events" given the results from the range comparator.

■ Block Diagram of A/D Converter

Figure 1-1 Block Diagram of A/D Converter



Note:

Set pins to be used as analog input to "analog input enable" by setting "1" to corresponding bit of Analog Input Enable Register (ADER). Write always "0" to the ANxx, which are not used and does not exist in corresponding device.

2. Operation

A/D converter converts analog voltages into digital values using CR successive approximation with one of four operating modes: Single Mode 1, Single Mode 2, Continuous Mode, and Pause Mode.

■ Single Mode 1/2

In Single Mode 1/2, A/D converter performs A/D conversion on consecutive channels starting from the start channel (ADSR:ANS) and ending with the end channel (ADSR:ANE). If ADSR:ANS=ADSR:ANE, A/D conversion is performed only on the analog input channel specified by ADSR:ANS (ADSR:ANE).

- Example:
  ANS = 00000b, ANE = 00011b:
  Activation → AN0 → AN1 → AN2 → AN3 → Finish
  ANS = 00010b, ANE = 00010b:
  Activation → AN2 → Finish

■ Continuous Mode

In Continuous Mode, A/D converter repeatedly performs A/D conversion on the sequence of the analog input channels defined by the start channel (ADSR:ANS) and the end channel (ADSR:ANE) continues until it is stopped by writing "0" to ADCS:BUSY bit.

If A/D conversion is stopped by writing "0" to ADCS:BUSY while it is in progress, the A/D Data Register (ADCR) contains the result of the previous conversion. If ADSR:ANS=ADSR:ANE, A/D conversion is performed on only one analog input channel.

- Example 1:
  ANS = 00000b, ANE = 00011b:
  Activation → AN0 → AN1 → AN2 → AN3 → AN1 → AN2 → AN3
  →→→ Repeated Sequence

- Example 2:
  ANS = 00010b, ANE = 00010b:
  Activation → AN2 → AN2 → AN2 →→→ Repeated Sequence

■ Pause Mode

In Pause Mode, A/D conversion progresses in the same way as Continuous Mode except that it requires reactivation every time the conversion of an analog input channel is complete.

- Example 1:
  ANS = 00000b, ANE = 00011b:
  Activation → AN0 → Pause → Reactivation
  → AN1 → Pause → Reactivation
  → AN2 → Pause → Reactivation
  → AN3 → Pause → Reactivation
  → AN0 →→→ Repeated Sequence

- Example 2:
  ANS = 00010b, ANE = 00010b:
  Activation → AN2 → Pause → Reactivation
  → AN2 → Pause → Reactivation
  → AN2 → Pause → Reactivation
  →→→ Repeated Sequence

Here, the reactivation must be one of the methods enabled in the A/D control status register (ADCS:STS1-0).

3. Conversion Data Protection Function

A/D converter has the functionality to protect the A/D conversion data from being destroyed by the new conversion result by entering the pause state.

■ Conversion Data Protection Function Description

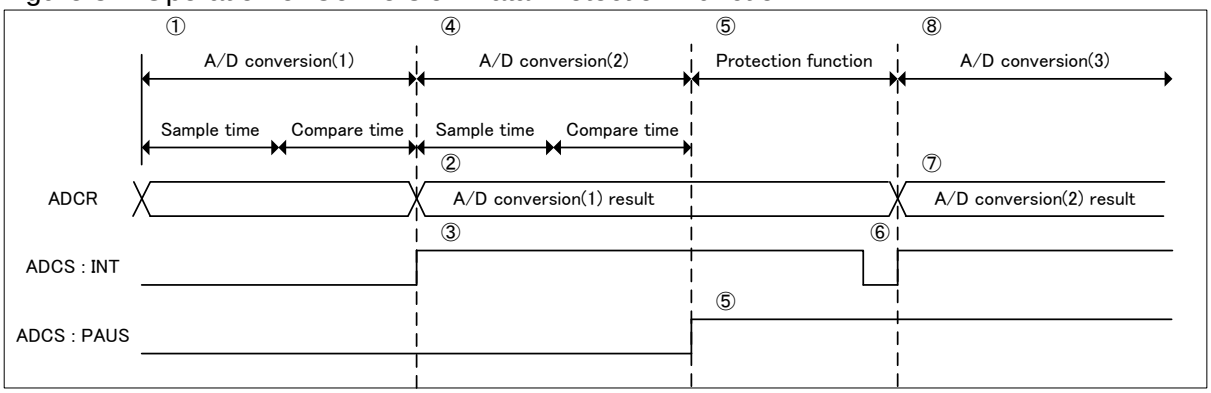
A/D Conversion Data Protection Function ensures that the A/D conversion result is retrieved by the user before it is updated by a new one. A/D conversion result is acquired using the successive approximation circuit and the A/D data register (ADCR). During the conversion on an analog input channel, each successive approximation step produces and retains 1-bit result. When the conversion finishes, the overall result is stored in A/D data register (ADCR). Below explains the behaviors of A/D converter when Conversion Data Protection Function is enabled, and when it is disabled.

- When the end of conversion interrupt is disabled (ADCS:INTE="0"), Conversion Data Protection Function is disabled. The A/D data register (ADCR) is updated with the new conversion result every time a conversion finishes. Therefore, the register always contains the newest conversion result.
- When the end of conversion interrupt is enabled (ADCS:INTE="1"), Conversion Data Protection Function is enabled. When the first A/D conversion finishes, the end of conversion interrupt flag is set to "1" (ADCS:INT="1"). If the next conversion finishes with this flag not cleared, the transfer of the A/D conversion result from the successive approximation circuit to the A/D data register (ADCR) is prevented, and the A/D conversion enters the pause state (ADCS:PAUS="1"). The result is transferred to the A/D data register (ADCR) when the end of conversion interrupt flag is cleared to "0" (write "0" to ADCS:INT) (refer to Figure 3-1 below).

Note:

Conversion Data Protection Function does not apply to the data buffer (ADCBR0-7). See "Section 4 Data Buffer Function" to see how the data buffer operates.

Figure 3-1 Operation of Conversion Data Protection Function



① A/D conversion is performed on an analog input channel (indicated as "A/D conversion (1)").
② The end of conversion flag (ADCS:INT) is set to "1", and the result of ① is stored in ADCR (indicated as "A/D conversion (1) result").
③ When ② occurs, if the end of conversion interrupt is enabled (ADCS:INTE="1"), the interrupt request is generated.
④ A/D conversion is performed on another analog input channel (indicated as "A/D conversion (2)").
⑤ Since ADCS:INT is not cleared, it is assumed that "A/D conversion (1) result" has not been retrieved when ④ is done. Conversion Data Protection Function protects "A/D conversion (1) result" by pausing ADC operation (ADCS:PAUS="1").
⑥ After "A/D conversion (1) result" is retrieved, ADCS:INT is cleared.
⑦ After ADCS:INT has been cleared, it is again set to “1”, and "A/D conversion (2)" is stored to ADCR (indicated as "A/D conversion (2) result").
⑧ Since ADCS:INT has been cleared in ⑥, ADC operation resumes by performing A/D conversion on another analog input channel (indicated as "A/D conversion (3)").

■ Conversion Data Protection Function: in case the A/D Conversion Result is Read by CPU

When an A/D conversion finishes with the end of conversion interrupt flag previously set to "1" (ADCS:INT="1") and the end of conversion interrupt enabled (INTE="1"), Conversion Data Protection Function enters the pause state (ADCS:PAUS="1") in order to prevent the content of the A/D data register (ADCR) from being destroyed. The conversion resumes when the end of conversion interrupt flag is cleared (writing "0" to ADCS:INT). Note that the A/D conversion pause bit (ADCS:PAUS) is not cleared to "0" automatically. Make sure to write "0" to ADCS:PAUS when the A/D conversion is resumed.

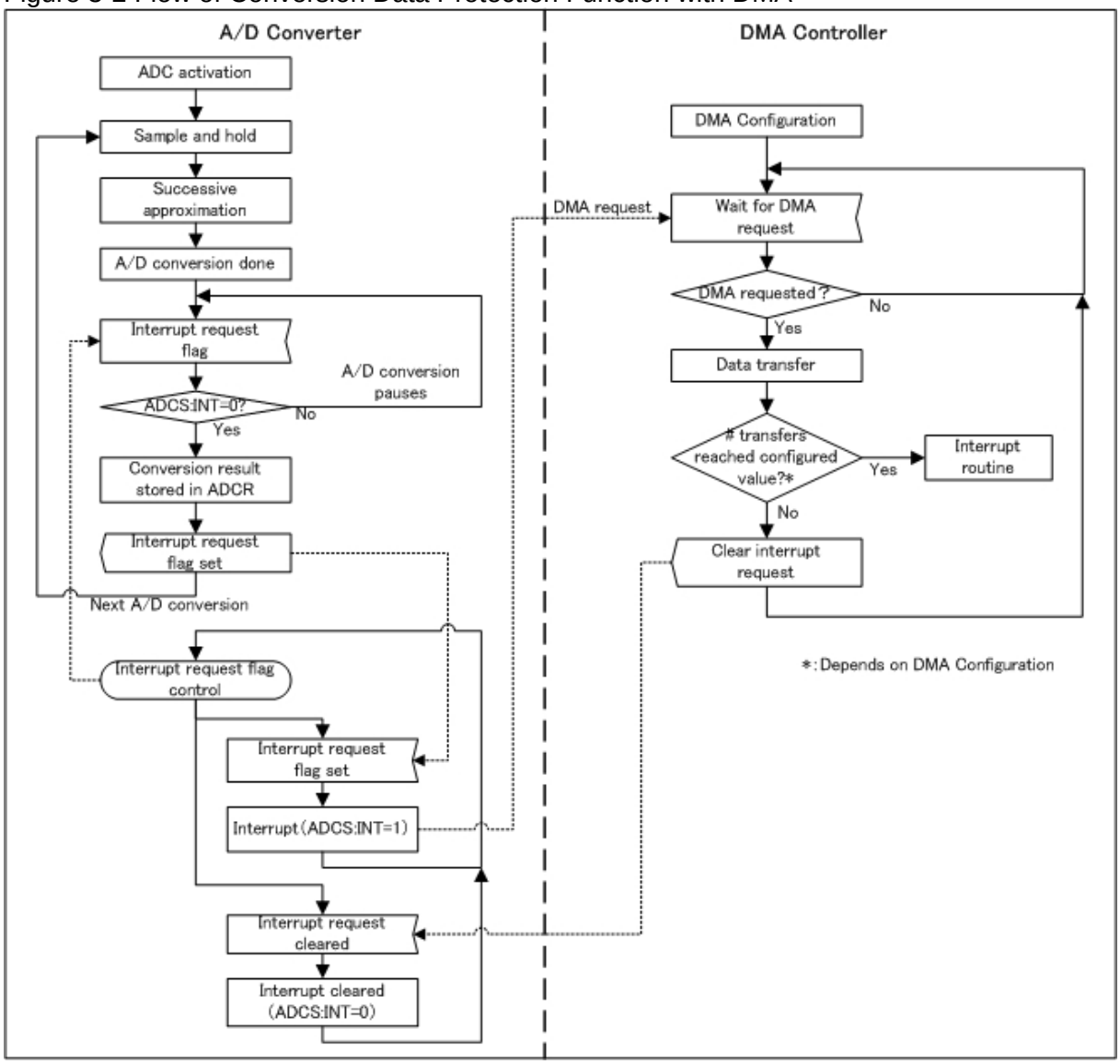
Notes:

- Disabling the end of conversion interrupt (ADCS:INTE="0") while the A/D conversion is the pause state could reactivate the conversion and update the content of the A/D data register (ADCR) before it is transferred to memory.
- For a scan conversion, read the conversion result before clearing the end of conversion interrupt flag (write "0" to ADCS:INT), since the content of the A/D data register is updated to a new value immediately after "0" is written to ADCS:INT.

■ Conversion Data Protection Function: in case the A/D Conversion Result is Transferred by DMA

With the end of conversion interrupt enabled (ADCS:INTE="1"), if an A/D conversion finishes while DMA is transferring the result of the previous conversion, Conversion Data Protection Function enters the pause state (ADCS:PAUS="1") in order to prevent the content of the A/D data register (ADCR) from being destroyed. When DMA notifies the A/D converter of the completion of the data transfer, the A/D conversion is resumed. Note that the A/D conversion pause bit (ADCS:PAUS) is not cleared to "0"automatically. Make sure to write "0" to ADCS:PAUS when the A/D conversion is resumed.

Figure 3-2 Flow of Conversion Data Protection Function with DMA



Notes:

- While DMA is in use for the data retrieval, do not clear the end of conversion interrupt flag manually by CPU write operation (writing "0" to ADCS:INT). Doing so overwrites the content of the A/D data register (ADCR) which is being transferred by DMA.
- While DMA is in use for the data retrieval, do not disable the end of conversion interrupt flag (ADCS:INT). Doing so during the pause state (while ADCS:PAUS="1") the content of the A/D data register (ADCR) is overwritten.
- While DMA is in use for the data retrieval, do not restart the A/D conversion. Restarting the A/D conversion during the pause state (while ADCS:PAUS="1") overwrites the content of the the A/D data register (ADCR).
- The end of conversion interrupt (controlled by ADCS:INTE) and the buffer limit reached interrupt (controlled by ADDBCR:BLRIE) are generated one common interrupt signal to notify CPU.
- DMA's notification of data transfer completion clears the buffer limit reached flag (ADDBCR:BLRF).

4. Data Buffer Function

Data Buffer Function buffers successive A/D conversion results. The number of the data buffer registers to be used can be set to one to eight by writing the value to ADDBCR:BLL2-0.

■ Data Buffer Function* Description

A/D converter stores the result of A/D conversion successively to the data buffer registers (ADCBR0-7). Buffering starts as soon as the current analog input channel number is greater than or equal to the value of the buffer start channel (ADDBCR:BSC). Note that the conversion results of the analog input channels are stored in the data buffer only if they are not skipped by Scan Disable Function. See "Section 6 Scan Disable Function" for further details.

The number of the data buffer registers to be used is configurable with ADDBCR:BLL2-0.

When the buffer pointer reaches the buffer level limit (ADDBCR:BLL2-0), the buffer limit reached flag is set (ADDBCR:BLRF="1"). The buffer limit reached interrupt is requested only if ADDBCR:BLRIEn="1". To clear the interrupt request, write "0" to ADDBCR:BLRFn, wait for the clear signal from DMA, or write to ADDBCRL.

Notes:

- When Data Buffer Function is in use, do not restart the A/D conversion. By doing so, the desired mapping between the data buffer registers (ADCBR0-7) and the analog input channels could be lost.
- Using Data Buffer Function with DMA requires DMAC burst transfer mode.
- This bit is cleared when a write access is made to ADDBCRL since a value is written to BLC2-0.

*: Because the Data Buffer Function start delay mode is optional, access is not possible on all devices. Please refer to the datasheet of the corresponding device.

■ Operation of Data Buffer Function with Wrap around Function Enabled (ADDBCR:WRAP="1")

Buffering starts with the data buffer register specified by the current buffer pointer (ADDBCR:BLC). When the buffer limit is reached (ADDBCR:BLRF="1"), the current buffer pointer (ADDBCR:BLC) is automatically cleared to "000b", and buffering continues from ADCBR0 by overwriting the old data buffer contents.

Note:

If the current buffer pointer is set to be larger than the buffer level limit (ADDBCR:BLC > ADDBCR:BLL), buffering starts with the data buffer register specified by ADDBCR:BLC, continues up to ADCBR7, goes back to ADCBR0, and ends with the data buffer register specified by ADDBCR:BLL (e.g. if BLC=6 and BLL=2, the order of buffering will be 6 → 7 → 0 → 1 → 2 → 0 → 1 → 2 → …).

■ Operation of Data Buffer Function with Wrap around Function Disabled (ADDBCR:WRAP="0")

Buffering starts with the data buffer register specified by the current buffer pointer (ADDBCR:BLC). When the buffer limit is reached (ADDBCR:BLRF="1"), buffering stops, and the current contents of the data buffer are not overwritten. In other words, the current buffer pointer (ADDBCR:BLC) is not cleared to "000_B" automatically (stops at the value equal to ADDBCR:BLL). In order to resume buffering, write to the lower byte of ADDBCR (ADDBCRL) (read the data buffer register values before resuming if the current buffer contents must be retrieved).

Notes:

- Buffering resumes as soon as a write access is made to ADDBCRL. Hence, if a write access is made to resume buffering while an A/D conversion is in progress, the next A/D conversion result is stored in the data buffer register specified by ADDBCR:BLC2-0 as soon as it becomes available.
- If the current buffer pointer is set to be larger than the buffer level limit (ADDBCR:BLC > ADDBCR:BLL), buffering starts with the data buffer register specified by ADDBCR:BLC, continues up to ADCBR7, goes back to ADCBR0, and ends with the data buffer register specified by ADDBCR:BLL (e.g. if BLC=6 and BLL=2, the order of buffering will be 6 → 7 → 0 → 1 → 2 → 0 → 1 → 2 → …).

■ Relationship between Conversion Data Protection Function and Wrap around Function

The purpose of Conversion Data Protection Function is to prevent the A/D conversion result from being updated with a new one before its retrieval. If an A/D conversion finishes with the end of conversion interrupt flag previously set (ADCS:INT="1"), the transfer of the A/D conversion result from the successive approximation circuit to the A/D data register (ADCR) is prevented by entering the pause state (ADCS:PAUS="1").

Data Buffer Function operates independently from Conversion Data Protection Function. In other words, when the buffer limit is reached with the buffer limit reached interrupt enabled (ADDBCR:BLRIE="1"), an interrupt request is generated, but Data Protection Function is not activated. The contents of the data buffer registers can be protected with the wrap around function enabled (ADDBCR:WRAP).

Note:

The end of conversion interrupt (controlled by ADCS:INTE) and the buffer limit reached interrupt (controlled by ADDBCR:BLRIE) are generated one common interrupt signal to notify CPU.

■ Relationship between Conversion Data Protection Function and Wrap around Function (ADCS:INTE="0", ADDBCR:BLRIE="0")

Neither the end of conversion interrupt nor the buffer limit reached interrupt occurs.

Also, Data Protection Function is disabled, and the content of the A/D data register (ADCR) is updated for every completion of A/D conversion.

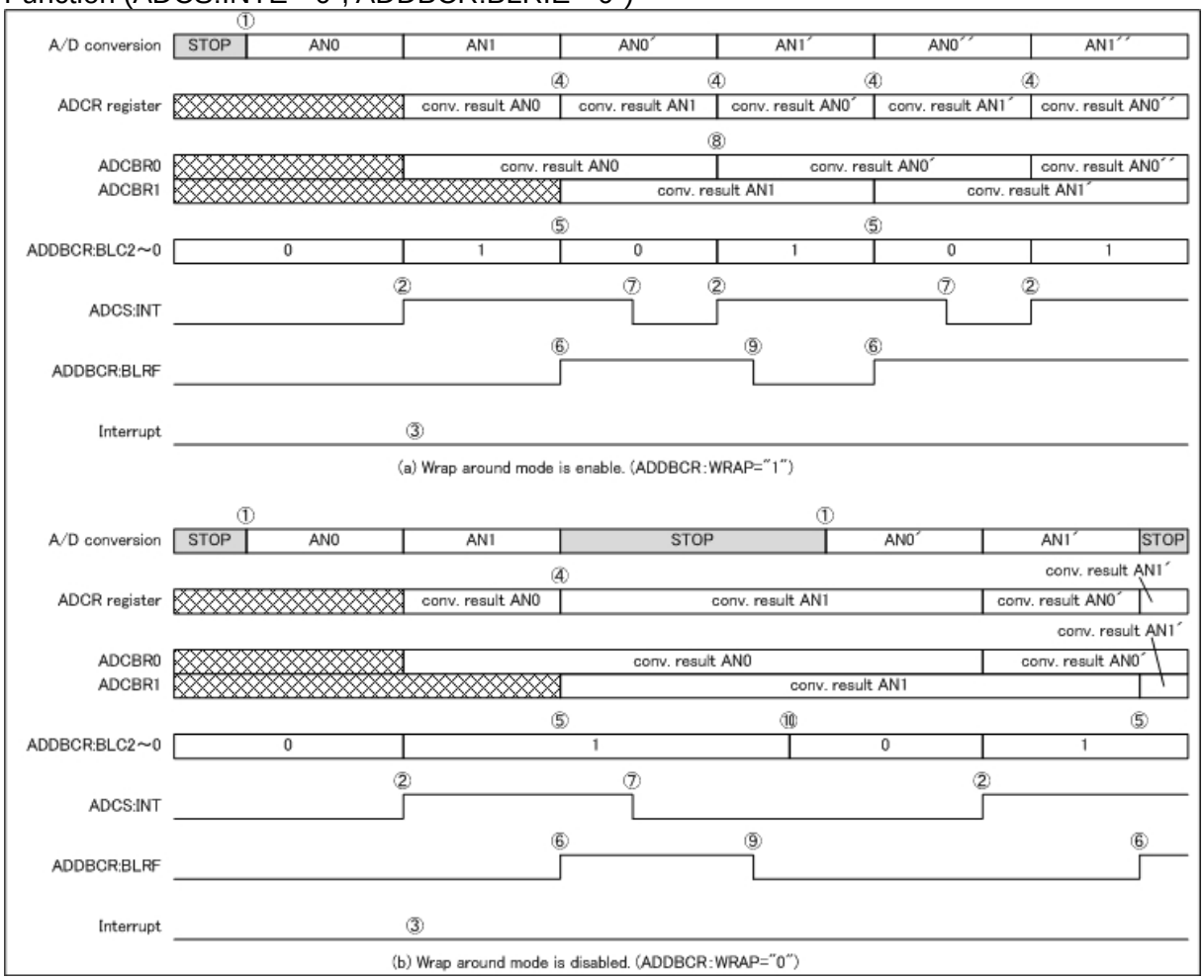
- In case ADDBCR:WRAP="1"
  When the buffer limit is reached (ADDBCR:BLRF="1"), the current buffer pointer (ADDBCR:BLC2-0) is automatically cleared to "000_B", and the old contents of the data buffer registers are overwritten with new A/D conversion results.

- In case ADDBCR:WRAP="0"
  When the buffer limit is reached (ADDBCR:BLRF="1"), the current buffer pointer (ADDBCR:BLC2-0) stops, and the contents of the data buffer registers are not overwritten. After reading the data buffer registers as necessary, the buffering can be resumed by making a write access to ADDBCRL. Note that while buffering is stopped, A/D conversion continues running.

Figure 4-1 below illustrates the relationship between Conversion Data Protection Function and the wrap around function with ADCS:INTE="0", ADDBCR:BLRIE="0", and with the following configuration:

- (a) ADCS:MD1-0 = "10_B" (continuous mode)
  (b) ADCS:MD1-0 = "01_B" (single mode 2)
- ADSR:ANS4-0: "00000_B" (AN0 is the start channel)
- ADSR:ANE4-0: "00001_B" (AN1 is the end channel)
- ADDBCR:BLL2-0 = "001_B" (up to ADCBR1 is used for buffering)
- ADDBCR:BSC4-0 = "00000_B" (ADCBR0 is the buffer start channel)

Figure 4-1 Relationship between Conversion Data Protection Function and Wrap around Function (ADCS:INTE="0", ADDBCR:BLRIE="0")



Operation of Data Buffer Function with ADCS:INTE="0" and ADDBCR:BLRIE="0"

① A/D conversion is initiated with a trigger.
② Conversion result of AN0 is stored in ADCR, and the end of conversion interrupt flag is set (ADCS:INT="1").
③ Interrupt does not occur since interrupt is disabled (ADCS:INTE="0" and ADDBCR:BLRIE="0").
④ Since Data Protection Function is disabled (ADCS:INTE="0"), the content of ADCR is updated when the new conversion finishes.
⑤ (a) If the wrap around function is enabled (ADDBCR:WRAP="1"), the current buffer pointer (ADDBCR:BLC) is automatically cleared to "000b".
(b) If the wrap around function is disabled (ADDBCR:WRAP="0"), the current buffer pointer is not automatically cleared, and it stops at the value of ADDBCR:BLL.
⑥ Buffer limit is reached and the buffer limit reached flag is set (ADDBCR:BLRF="1").
⑦ The end of conversion interrupt flag is cleared by writing "0" to it (ADCS:INT="0").
⑧ (a) If the wrap around function is enabled (ADDBCR:WRAP="1"), the data buffer register is overwritten by a new A/D conversion result.
⑨ The buffer limit reached interrupt flag is cleared by writing "0" to it (ADDBCR:BLRF="0").
⑩ (b) If the wrap around function is disabled (ADDBCR:WRAP="0"), a write access to ADDBCRL is made to resume buffering. Buffering is resumed with the data buffer registered specified by the written value of ADDBCR:BLC.

■ Relationship between Conversion Data Protection Function and Wrap around Function (ADCS:INTE="1", ADDBCR:BLRIE="0")

The end of conversion interrupt occurs for every completion of A/D conversion.

Conversion Data Protection Function is enabled. When an A/D conversion finishes with the end of conversion interrupt flag previously set to "1" (ADCS:INT="1"), A/D converter enters the pause state (ADCS:PAUS="1") in order to prevent the content of the A/D data register (ADCR) from being destroyed.

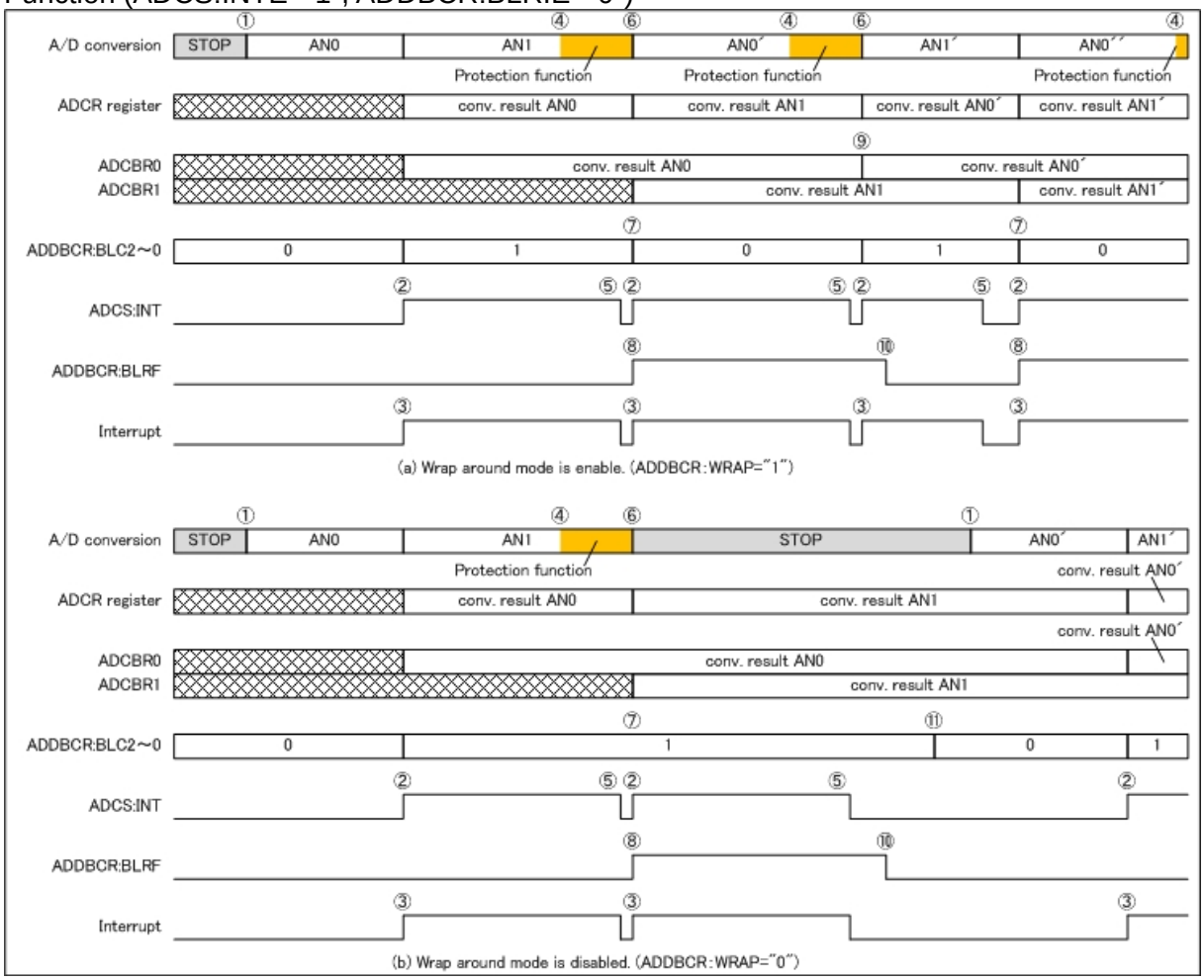
- In case ADDBCR:WRAP="1"
  When the buffer limit is reached (ADDBCR:BLRF="1"), the current buffer pointer (ADDBCR:BLC2-0) is automatically cleared to "000_B", and the old contents of the data buffer registers are overwritten with new A/D conversion results.

- In case ADDBCR:WRAP="0"
  When the buffer limit is reached (ADDBCR:BLRF="1"), the current buffer pointer (ADDBCR:BLC2-0) stops, and the contents of the data buffer registers are not overwritten. After reading the data buffer registers as necessary, the buffering can be resumed by making a write access to ADDBCRL. Note that while buffering is stopped, A/D conversion continues running.

Figure 4-2 below illustrates the relationship between Conversion Data Protection Function and the wrap around function with ADCS:INTE="1", ADDBCR:BLRIE="0", and with the following configuration:

- (a)ADCS:MD1-0 = "10_B" (continuous mode)
  (b)ADCS:MDI1-0= "01_B" (single mode 2)
- ADSR:ANS4-0 = "00000_B" (AN0 is the start channel)
- ADSR:ANE4-0 = "00001_B" (AN1 is the end channel)
- ADDBCR:BLL2-0 = "001_B" (up to ADCBR1 is used for buffering)
- ADDBCR:BSC4-0 = "00000_B" (ADCBR0 is the buffer start channel)

Figure 4-2 Relationship between Conversion Data Protection Function and Wrap around Function (ADCS:INTE="1", ADDBCR:BLRIE="0")



Operation of Data Buffer Function with ADCS:INTE="1" and ADDBCR:BLRIE="0"

① A/D conversion is initiated with a trigger.
② Conversion result of AN0 is stored in ADCR, and the end of conversion interrupt flag is set (ADCS:INT="1").
③ Then end of conversion interrupt occurs since it is enabled (ADCS:INTE="1").
④ Since Data Protection Function is enabled (ADCS:INTE="1"), the content of ADCR is protected by pausing A/D converter operation.
⑤ After reading the A/D data register (ADCR), the end of conversion interrupt flag (ADCS:INT) is cleared (by writing "0" to ADCS:INT or by DMA clear signal).
⑥ (a) With ADCS:INT cleared, the conversion result is transferred to ADCR, and the A/D conversion resumes.
(b) With ADCS:INT cleared, the conversion result is transferred to ADCR, and the A/D conversion stops since the buffer limit is reached.
⑦ Buffer limit is reached and the buffer limit reached flag is set (ADDBCR:BLRF="1").
⑧ The end of conversion interrupt flag is cleared by writing "0" to it (ADCS:INT="0").
⑨ (a) If the wrap around function is enabled (ADDBCR:WRAP="1"), the data buffer register is overwritten by a new A/D conversion result.
⑩ The buffer limit reached interrupt flag is cleared by writing "0" to it (ADDBCR:BLRF="0").
⑪ (b) If the wrap around function is disabled (ADDBCR:WRAP="0"), a write access to ADDBCRL is made to resume buffering. Buffering is resumed with the data buffer registered specified by the written value of ADDBCR:BLC.

■ Relationship between Conversion Data Protection Function and Wrap around Function (ADCS:INTE="0", ADDBCR:BLRIE="1")

The buffer limit reached flag occurs (ADDBCR:BLRF="1") when the buffer is filled up to the buffer register specified by ADDBCR:BLL2-0.

Data Protection Function is disabled, and the content of the A/D data register (ADCR) is updated for every completion of A/D conversion.

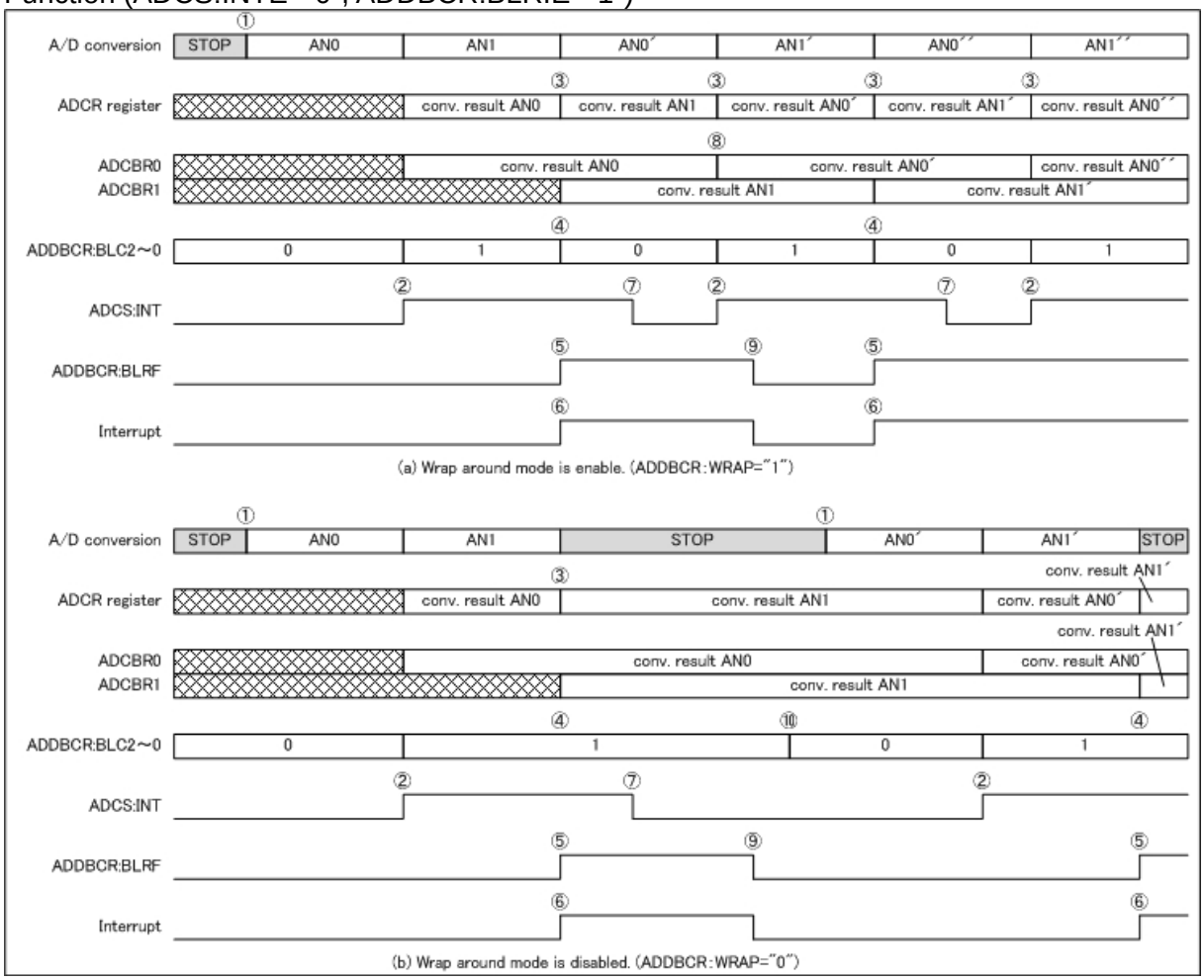
- In case ADDBCR:WRAP="1"
  When the buffer limit is reached (ADDBCR:BLRF="1"), the current buffer pointer (ADDBCR:BLC2-0) is automatically cleared to "000_B", and the old contents of the data buffer registers are overwritten with new A/D conversion results.

- In case ADDBCR:WRAP="0"
  When the buffer limit is reached (ADDBCR:BLRF="1"), the current buffer pointer (ADDBCR:BLC2-0) stops, and the contents of the data buffer registers are not overwritten. After reading the data buffer registers as necessary, the buffering can be resumed by making a write access to ADDBCRL. Note that while buffering is stopped, A/D conversion continues running.

Figure 4-3 below illustrates the relationship between Conversion Data Protection Function and the wrap around function with ADCS:INTE="0", ADDBCR:BLRIE="1", and with the following configuration:

- (a) ADCS:MD1-0 = "10_B" (continuous mode)
  (b) ADCS:MD1-0 = "01_B" (single mode 2)
- ADSR:ANS4-0: "00000_B" (AN0 is the start channel)
- ADSR:ANE4-0: "00001_B" (AN1 is the end channel)
- ADDBCR:BLL2-0 = "001_B" (up to ADCBR1 is used for buffering)
- ADDBCR:BSC4-0 = "00000_B" (ADCBR0 is the buffer start channel)

Figure 4-3 Relationship between Conversion Data Protection Function and Wrap around Function (ADCS:INTE="0", ADDBCR:BLRIE="1")



Operation of Data Buffer Function with ADCS:INTE="0" and ADDBCR:BLRIE="1"

① A/D conversion is initiated with a trigger.

② Conversion result of AN0 is stored in ADCR, and the end of conversion interrupt flag is set (ADCS:INT="1").

③ Since Data Protection Function is disabled (ADCS:INTE="0"), the content of ADCR is updated when the new conversion finishes.

④ (a) If the wrap around function is enabled (ADDBCR:WRAP="1"), the current buffer pointer (ADDBCR:BLC) is automatically cleared to "000_B".
(b) If the wrap around function is disabled (ADDBCR:WRAP="0"), the current buffer pointer is not automatically cleared, and it stops at the value of ADDBCR:BLL.

⑤ Buffer limit is reached and the buffer limit reached flag is set (ADDBCR:BLRF="1").

⑥ Since the buffer limit reached interrupt is enabled, an interrupt request is generated when the buffer limit is reached.

⑦ The end of conversion interrupt flag is cleared by writing "0" to it (ADCS:INT="0").

⑧ (a) If the wrap around function is enabled (ADDBCR:WRAP="1"), the data buffer register is overwritten by a new A/D conversion result.

⑨ The buffer limit reached interrupt flag is cleared by writing "0" to it (ADDBCR:BLRF="0").

⑩ (b) If the wrap around function is disabled (ADDBCR:WRAP="0"), a write access to ADDBCRL is made to resume buffering. Buffering is resumed with the data buffer registered specified by the written value of ADDBCR:BLC.

■ Relationship between Conversion Data Protection Function and Wrap around Function (ADCS:INTE="1", ADDBCR:BLRIE="1")

Both the end of conversion interrupt and the buffer limit reached interrupt are enabled (ADCS:INTE="1", ADDBCR:BLRIE="1"). The end of conversion interrupt and the buffer limit reached interrupt are generated one common interrupt signal to notify CPU.

Conversion Data Protection Function is enabled. When an A/D conversion finishes with the end of conversion interrupt flag previously set to "1" (ADCS:INT="1"), A/D converter enters the pause state (ADCS:PAUS="1") in order to prevent the content of the A/D data register (ADCR) from being destroyed.

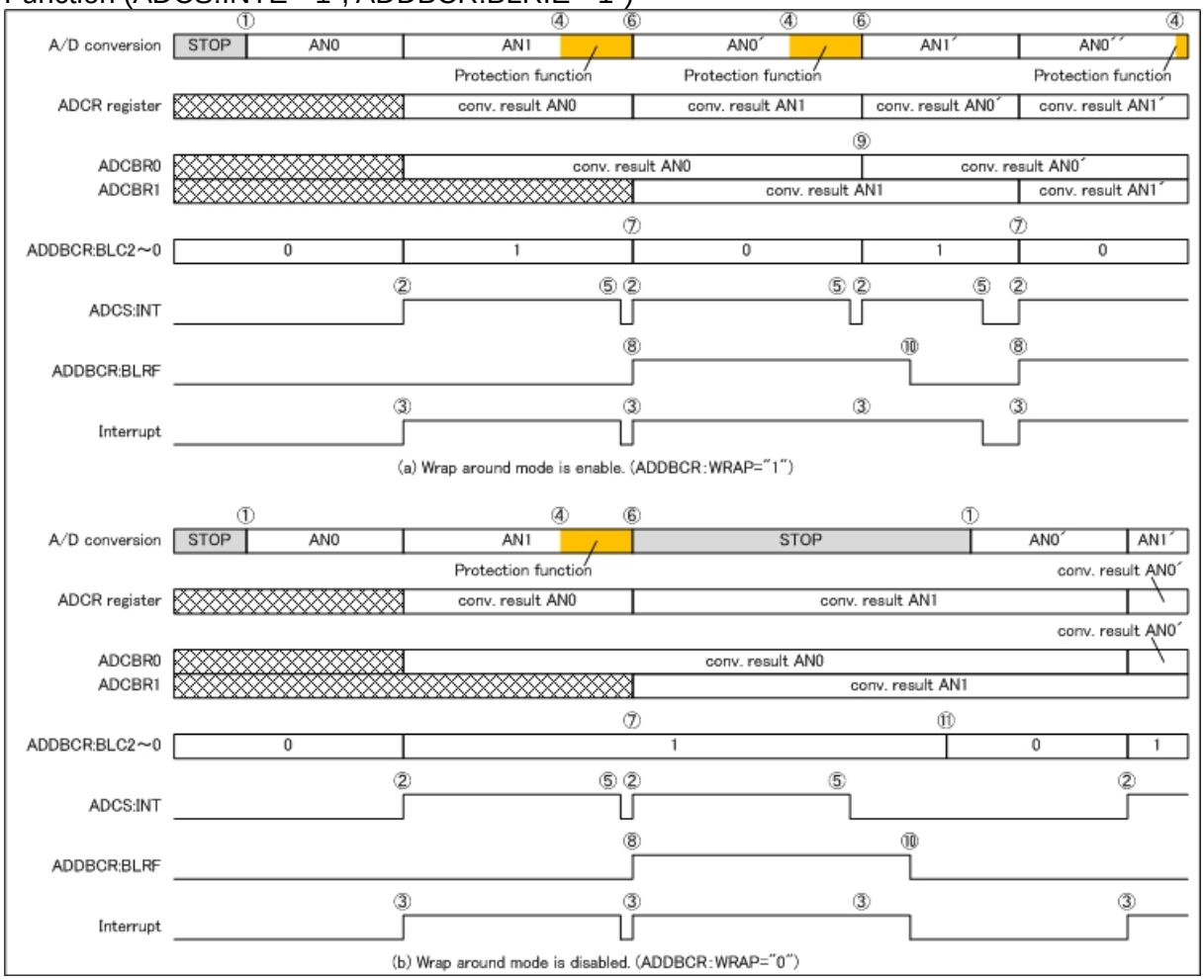
- In case ADDBCR:WRAP="1"
  When the buffer limit is reached (ADDBCR:BLRF="1"), the current buffer pointer (ADDBCR:BLC2-0) is automatically cleared to "000b", and the old contents of the data buffer registers are overwritten with new A/D conversion results.

- In case ADDBCR:WRAP="0"
  When the buffer limit is reached (ADDBCR:BLRF="1"), the current buffer pointer (ADDBCR:BLC2-0) stops, and the contents of the data buffer registers are not overwritten. After reading the data buffer registers as necessary, the buffering can be resumed by making a write access to ADDBCRL. Note that while buffering is stopped, A/D conversion continues running.

Figure 4-4 below illustrates the relationship between Conversion Data Protection Function and the wrap around function with ADCS:INTE="1", ADDBCR:BLRIE="1", and with the following configuration:

- (a) ADCS:MD1-0 = "10_B" (continuous mode)
  (b) ADCS:MD1-0 = "01_B" (single mode 1)
- ADSR:ANS4-0: "00000_B" (AN0 is the start channel)
- ADSR:ANE4-0: "00001_B" (AN1 is the end channel)
- ADDBCR:BLL2-0 = "001_B" (up to ADCBR1 is used for buffering)
- ADDBCR:BSC4-0 = "00000_B" (ADCBR0 is the buffer start channel)

Figure 4-4 Relationship between Conversion Data Protection Function and Wrap around Function (ADCS:INTE="1", ADDBCR:BLRIE="1")



Operation of Data Buffer Function with ADCS:INTE="1" and ADDBCR:BLRIE="1"

① A/D conversion is initiated with a trigger.

② Conversion result of AN0 is stored in ADCR, and the end of conversion interrupt flag is set (ADCS:INT="1").

③ Since ADCS:INTE="1" and ADDBCR:BLRIE="1", an interrupt is generated by ADCS:INT and ADCS:BLRF.

④ Since Data Protection Function is enabled (ADCS:INTE="1"), the content of ADCR is protected by pausing A/D converter operation.

⑤ After reading the A/D data register (ADCR), the end of conversion interrupt flag (ADCS:INT) is cleared (by writing "0"to ADCS:INT or by DMA clear signal).

⑥ (a) With ADCS:INT cleared, the conversion result is transferred to ADCR, and the A/D conversion resumes.
(b) With ADCS:INT cleared, the conversion result is transferred to ADCR, and the A/D conversion stops since the buffer limit is reached.

⑦ (a) If the wrap around function is enabled (ADDBCR:WRAP="1"), the current buffer pointer (ADDBCR:BLC) is automatically cleared to "000_B".
(b) If the wrap around function is disabled (ADDBCR:WRAP="0"), the current buffer pointer is not automatically cleared, and it stops at the value of ADDBCR:BLL.

⑧ Buffer limit is reached and the buffer limit reached flag is set (ADDBCR:BLRF="1").

⑨ (a) If the wrap around function is enabled (ADDBCR:WRAP="1"), the data buffer register is overwritten by a new A/D conversion result.

⑩ The buffer limit reached interrupt flag is cleared by writing "0" to it (ADDBCR:BLRF="0").

⑪ (b) If the wrap around function is disabled (ADDBCR:WRAP="0"), a write access to ADDBCRL is made to resume buffering. Buffering is resumed with the data buffer registered specified by the written value of ADDBCR:BLC.

5. Range Comparator Function

Range Comparator Function compares the result of A/D conversion with the upper/lower threshold value to determine whether the result is "in range" or "out of range". Range Comparator Function setting is organized as per analog input channel.

■ Range Comparator Function* Description

There are four registers for both the upper threshold value and the lower threshold value. Below shows four selectable combinations of the upper/lower registers:

- Upper threshold value = ADRCOH0 & Lower threshold value = ADRCOL0
- Upper threshold value = ADRCOH1 & Lower threshold value = ADRCOL1
- Upper threshold value = ADRCOH2 & Lower threshold value = ADRCOL2
- Upper threshold value = ADRCOH3 & Lower threshold value = ADRCOL3

Range Comparator Function enable setting (ADCC:RCOEn) and the range comparator interrupt enable setting (ADCC:RCOIEn) can be configured for each analog input channel.

The value of the inverted range selection bit (ADRCOIRS:RCOIRSn) determines whether Range Comparator Function checks for "in range" or "out of range".

In order to use the range comparator interrupt, enable Range Comparator Function (set ADCC:RCOEn to "1") and the range comparator interrupt (set ADCC:RCOIE= "1"). There are two conditions that set the range comparator interrupt flag to "1" (ADRCOINTF:RCOINTn="1"):

- A/D conversion result is either larger than the upper threshold value or lower than the lower threshold value when "out of range" check is selected (ADRCOIRS:RCOIRSn="0")
- A/D conversion result is smaller than the upper threshold value and larger than the lower threshold value when "in range" check is selected (ADRCOIRS:RCOIRSn="1")

For "out of range" check (ADRCOIRS:RCOIRSn="0"), the range comparator over threshold flag (ADRCOOF:RCOOFn) is updated when the range comparator interrupt flag is set from "0" to "1". If the A/D conversion result is larger than the upper threshold value, the flag is set to "1" (ADRCOOF:RCOOFn="1"). If the A/D conversion result is lower than the lower threshold value, the flag is set to "0" (ADRCOOF:RCOOFn="0").

For "in range" check (ADRCOIRS:RCOIRSn="1"), the range comparator over threshold flag is not updated (maintains the old value).

Figure 5-1 and Table 5-1 below illustrates the operation of Range Comparator Function with the following configuration.

- Used Analog Input Channel = AN0
- ADRCOH0 = arbitrary value
- ADRCOL0 = arbitrary value
- ADCC0 = "1100_B" (ADRCOH0 & ADRCOL0 selected as threshold values / Range Comparator Function enabled / range comparator interrupt enabled)

Note:

"n" indicates the analog input channel number (0-31).

*: Because the Range Comparator Function start delay mode is optional, access is not possible on all devices. Please refer to the datasheet of the corresponding device.

Figure 5-1 Operation of Range Comparator

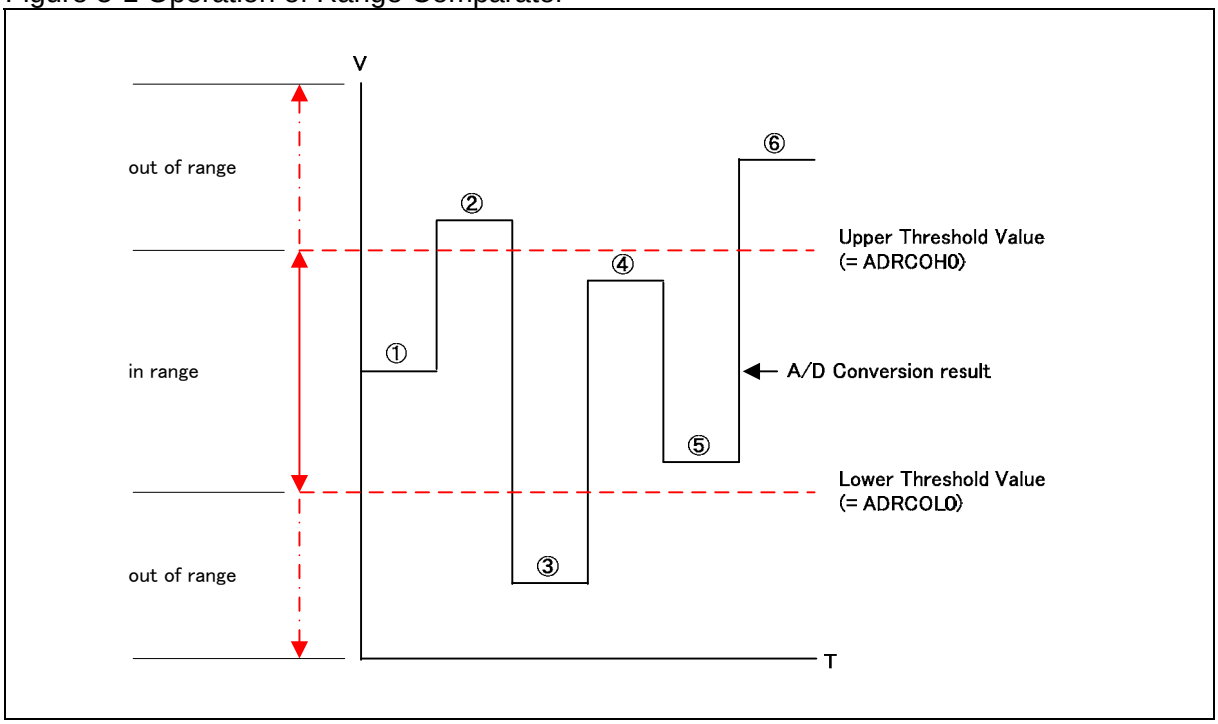


Table 5-1 Results of Range Comparisons Shown in Figure 5-1

	In Range Register Bits (ADRCOIRS0:RCOIRS0)											
	0 ("out of range" check)						1 ("in range" check)					
A/D Conversion State in Figure 5-1	①	②	③	④	⑤	⑥	①	②	③	④	⑤	⑥
Range Comparator Flag (ADRCOINTF:RCOINT0)	0	1 *A	1 *A	0	0	1 *A	1 *A	0	0	1	1	0
Range Comparator Over Threshold Flag (ADRCOOF0:RCOOF0)	*B	1	0	0 *B	0 *B	1	*B	*B	*B	*B	*B	*B

Notes:

- *A: After the range comparator interrupt flag is set (ADRCOINTF:RCOINT0="1"), it is cleared (ADRCOINTF:RCOINT0="0") by writing "0" to it.
- *B: Indicates "don't care". For "in range" check (ADRCOIRS:RCOIRSn="1"), the range comparator over threshold flag is not updated (maintains the old value).

6. Scan Disable Function

During Scan Conversion, Scan Disable Function can select analog input channels from the sequence of the channels defined by ADSR:ANS4-0 and ADSR:ANE4-0, in order to skip their A/D conversions.

■ Scan Disable Function* Description

Scan Disable Function enables the A/D converter to skip the conversions on specific analog input channels. This function can be enabled/disabled with the scan disable function enable bit (ADCS:CSEN).

If Scan Disable Function is disabled (ADCS:CSEN="0"), conversions for the analog input channels between the start channel (ADSR:ANS) and the end channel (ADSR:ANE) are performed consecutively.

If Scan Disable Function is enabled (ADCS:CSEN="1"), A/D conversions are not performed for the analog input channels that are configured as "skipped" with the disable channel for scan bit (ADDSCR:DCSn="1").

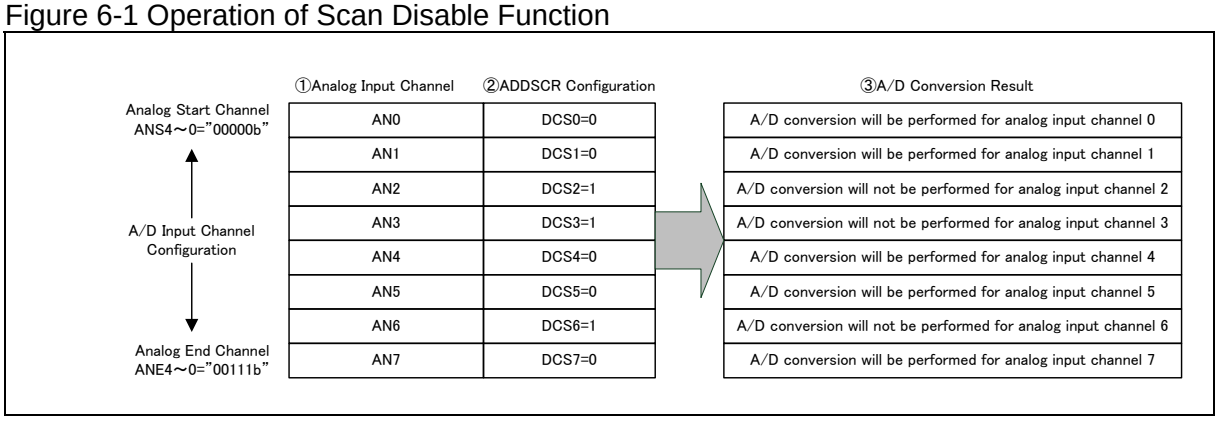
If Data Buffer Function is used, only conversion results of the unskipped analog input channels are stored in the data buffer (ADCBR0-7) sequentially.

Figure 6-1 illustrates the operation of Scan Disable Function with the following configuration:

- ADSR:ANS = "00000_B" (AN0 is the start channel)
- ADSR:ANE = "00111_B" (AN7 is the end channel)
- ADDCSR0 = "01001100_B"

*: Because the Scan Disable Function start delay mode is optional, access is not possible on all devices. Please refer to the datasheet of the corresponding device.

Figure 6-1 Operation of Scan Disable Function



Operation of Scan Disable Function

① A/D converter performs Scan Conversion from AN0 to AN7.
② The scan disable setting for each channel is indicated.
③ The conversions on selected channels are skipped.

Notes:

- "n" indicates the analog input channel number (0-31).
- Do not change the scan disable configuration (ADDSCR:DCS31-0) while A/D conversion is in progress (ADCS:BUSY="1"). If it has to be changed, make sure that A/D conversion is not in progress (check that ADCS:BUSY="0")
- Do not modify Scan Disable Function Enable bit while the A/D conversion is in progress (while ADCS.BUSY="1").
- Designate analog input channels that exist to the start channel (ADSR:ANS4-0) and the end channel (ADSR:ANE4-0).
- Do not skip all the analog input channels.

7. ADC Pulse Detection Function

Given the result of the range comparison, Pulse Detection Function counts the positive/negative events downward. A "pulse" of the positive events is detected with the ADC pulse detection zero flag when the positive counter decrements from "0x01" to "0x00". Negative counter is used as the noise filter. Pulse Detection Function is configured as per analog input channel.

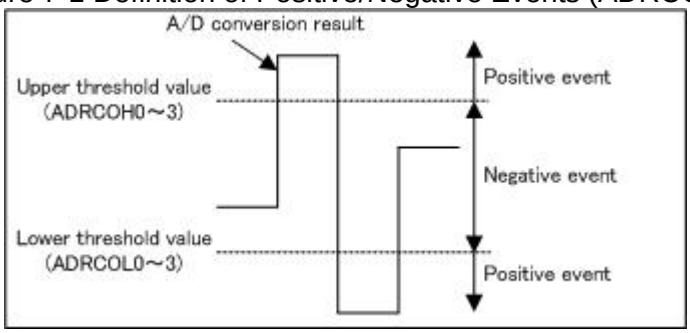
■ ADC Pulse Detection Function* Description

ADC Pulse Detection Function inputs the result of the range comparison for its operation. The result of the range comparison is either "positive" or "negative". The definitions of the "positive event" and the "negative event" are the following.

1. RCOIRSn="0" (check for "out of range")
   - The event is considered to be a positive event if the A/D conversion result is larger than the upper threshold value (ADRCOH0-3) or smaller than the lower threshold value (ADRCOL0-3).
   - The event is considered to be a negative event if the A/D conversion result is smaller than or equal to the upper threshold value (ADRCOH0-3) and larger than or equal to the lower threshold value (ADRCOL0-3).

   Figure 7-1 shows an example of how the positive/negative events are judged when "out of range" check is selected (ADRCOIRS:RCOIRSn="0").

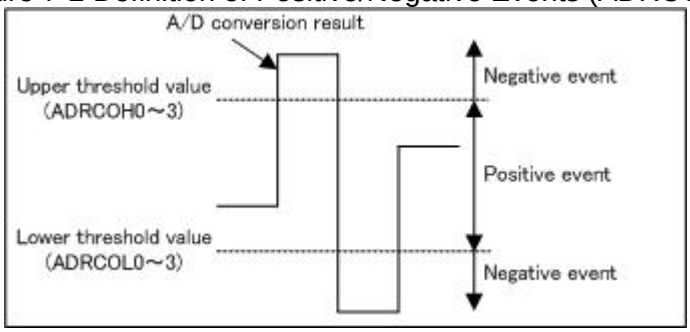
Figure 7-1 Definition of Positive/Negative Events (ADRCOIRS:RCOIRSn="0")



2. RCOIRSn="1" (check for "in range")
   - The event is considered to be a negative event if the A/D conversion result is larger than the upper threshold value (ADRCOH0-3) or smaller than the lower threshold value (ADRCOL0-3).
   - The event is considered to be a positive event if the A/D conversion result is smaller than or equal to the upper threshold value (ADRCOH0-3) and larger than or equal to the lower threshold value (ADRCOL0-3).

   Figure 7-2 shows an example of how the positive/negative events are judged when "out of range" check is being selected (ADRCOIRS:RCOIRSn="0").

Figure 7-2 Definition of Positive/Negative Events (ADRCOIRS:RCOIRSn="1")



*: Because ADC Pulse Detection Function start delay mode is optional, access is not possible on all devices. Please refer to the datasheet of the corresponding device.

Table 7-1 shows the summary of how positive/negative events are caused.

Table 7-1 Summary of Positive/Negative Event Generation

Value of RCOIRSn	Range Comparison Result	Type of Event
"0" (check for "out of range")	In range	Negative event
	Out of range	Positive event
"1" (check for "in range")	Out of range	Negative event
	In range	Positive event

When "0" is written to the ADC pulse counter zero flag (ADPCZF:CTPZFn), Pulse Detection Function starts its operation by initializing the corresponding ADC pulse counter register (ADPCTNCTn/ADPCTPCTn) with its reload counter value (ADPCTNRDn/ADPCTPRDn).

With the occurrence of a positive event, the ADC pulse positive counter (ADPCTPCT0-31) decrements. Similarly, the ADC pulse negative counter (ADPCTNCT0-31) decrements with the occurrence of a negative event.

When the ADC pulse positive counter decrements to from "0x01" to "0x00", the pulse counter zero flag (ADPCZF: CTPZFn) is set to "1". If the ADC pulse counter interrupt is enabled (ADPCIE0:CTPIEn="1"), an interrupt is generated. While the ADC pulse counter interrupt is set, Pulse Counter Function is stopped with the following counter status:

- ADC pulse positive counter (ADPCTPCTn) stops and retains the value "0x00".
- ADC pulse negative counter (ADPCTNCTn) stops and retains the value of the ADC pulse negative reload counter (ADPCTNRDn).

In order to clear the ADC pulse counter interrupt request, write "0" to the ADC pulse counter zero flag (ADPCZF:CTPZFn). Clearing the flag also reloads the ADC pulse counter (ADPCTPCTn/ADPCTNCTn) with the value of the corresponding ADC pulse reload counter (ADPCTPRDn /ADPCTNRDn) and reactivate Pulse Detection Function.

The ADC pulse positive counter (ADPCTPCT0-31) is initialized with the reload value (ADPCTPRDn) when

1. the ADC pulse negative counter (ADPCTNCT0-31) decrements from "0x01" to "0x00"
2. "0" is written to the ADC pulse counter zero flag (ADPCZF:CTPZFn)

On the other hand, the ADC pulse negative counter (ADPCTNCT0-31) is initialized with the reload value (ADPCTNRDn) when

3. a positive event occurs
4. "0" is written to the ADC pulse counter zero flag (ADPCZF:CTPZFn)

Each analog input channel has its own ADC pulse counter (ADPCTNCTn/ ADPCTPCTn), ADC pulse counter reload value (ADPCTNRDn/ ADPCTPRDn), ADC pulse counter interrupt enable setting (ADPCIE:CTPIEn), and ADC pulse counter zero flag (ADPCZF:CTPZFn).

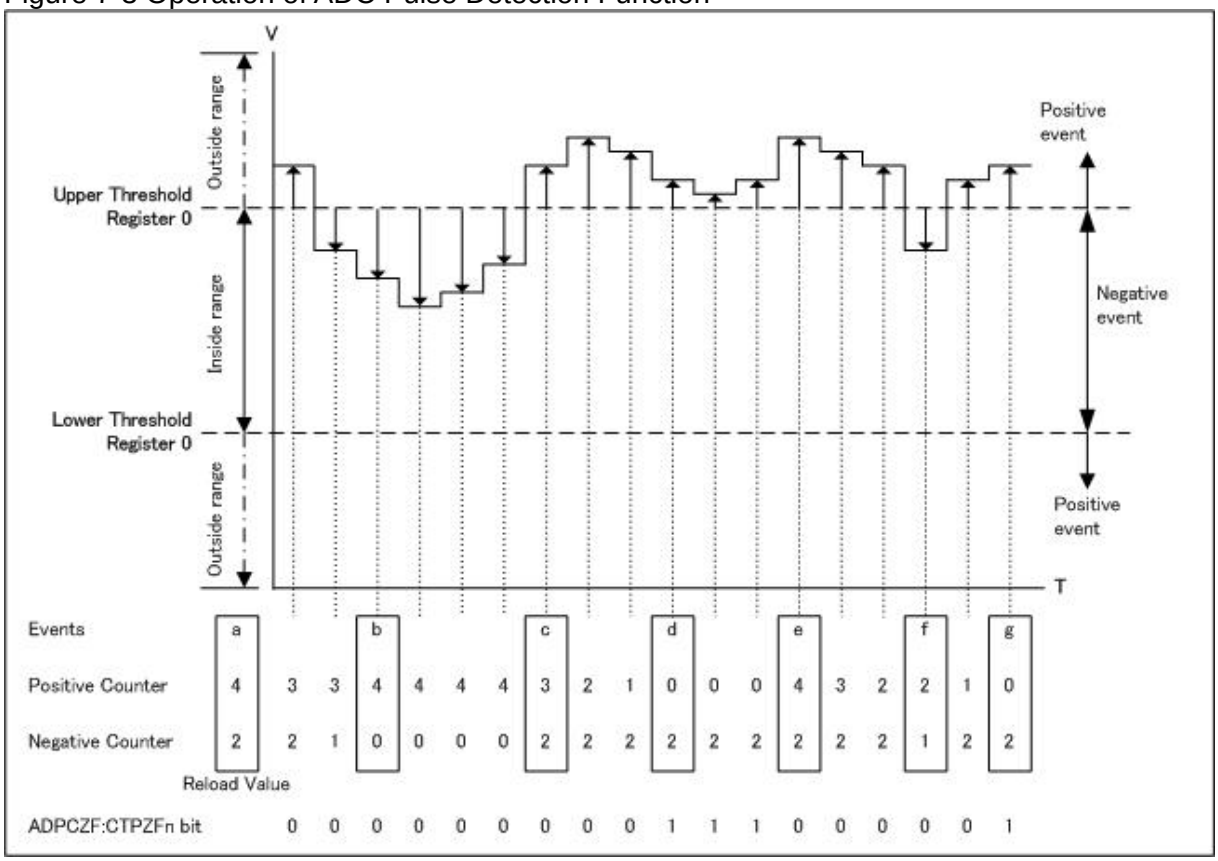
Figure 7-3 below illustrates the operation of ADC Pulse Detection Function with the following configuration:

- ADRCOIRS:RCOIRSn = "0_B" (check for "out of range")
- ADCC0-15:RCOn1/RCOn0 = "00_B" (check for "in range")
- ADPCTNRDn = "0x02"
- ADPCTPRDn = "0x04"

Note:

"n" indicates the analog input channel number (0-31).

Figure 7-3 Operation of ADC Pulse Detection Function



Operation of ADC Pulse Detection Function

a. "0" is written to ADC pulse counter zero flag (ADPCZF:CTPZFn) to initialize the ADC pulse counter (ADPCTNCTn/ADPCTPCTn) with the reload value (ADPCTNRDn/ ADPCTPRDn).
b. Since the negative counter has decremented to "0", the positive counter is initialized.
c. Since a positive event has occurred, the negative counter is initialized and the positive counter decrements.
d. Since the positive counter has decremented to "0", the ADC pulse counter zero flag (ADPCZF:CTPZFCn) is set to "1".
e. Since "0" is written to the ADC pulse counter zero flag (ADPCZF:CTPZFn), both the positive counter (ADPCTPCTn) and the negative counter (ADPCTNCTn) are initialized.
f. Since the negative counter has not decremented "0", the positive counter remains the same.
g. Since the positive counter has decrements to "0", the ADC pulse counter zero flag (ADPCZF:CTPZFn) is set to "1".

8. Registers

This section explains the configuration and functions of the registers used for the A/D Converter.

■ List of A/D Converter Registers

Abbreviated Register Name	Register Name	Reference
ADCS	A/D Control Status Register	See 8.1
ADCR	A/D Data Register	See 8.2
ADSR	A/D Setting Register	See 8.3
ADDBCR	A/D Data Buffer Control Register	See 8.4
ADCBR0-7	A/D Data Buffer Registers	See 8.5
ADRCOH0-3	Upper Threshold Registers for Range Comparator	See 8.6
ADRCOL0-3	Lower Threshold Registers for Range Comparator	See 8.7
ADCC0-15	Channel Control Registers for Range Comparator	See 8.8
ADRCOIRS0-3	Inverted Range Selection Registers for Range Comparator	See 8.9
ADRCOOF0-3	Range Comparator Over Threshold Flag Registers	See 8.10
ADRCOINTF0-3	Range Comparator Flag Registers	See 8.11
ADDSCR0-3	Disable Channel for Scan Register	See 8.12
ADPCTNRD0-31/ ADPCTPRD0-31	ADC Pulse Counter Reload Registers	See 8.13
ADPCTNCT0-31/ ADPCTPCT0-31	ADC Pulse Counter Registers	See 8.14
ADPCIE0-3	ADC Pulse Counter Interrupt Enable Registers	See 8.15
ADPCZF0-3	ADC Pulse Counter Zero Flag Registers	See 8.16
ADERx	Analog input enable register	See 8.17

8.1. A/D Control Status Register (ADCS)

The A/D control status register controls the A/D conversion and indicates the status of the A/D conversion.

■ A/D Control Status Register Higher (ADCSH)

ADCSH								
bit	15	14	13	12	11	10	9	8
	BUSY	INT	INTE	PAUS	STS1	STS0	STRT	RESV
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	W	-
Initial value	0	0	0	0	0	0	0	0

Notes:

- Reactivation while A/D conversion is in progress is only possible in Single Mode 1.
- In Single Mode 1, reactivation during A/D conversion is possible through software, external trigger, and reload timer.
- In Single Mode 2, Continuous Mode, and Pause Mode, reactivation during A/D conversion is not possible.

[bit15] BUSY: Busy Flag and Stop

Bit	Description	
	Read	Write
0	A/D conversion is not in progress	Terminates A/D conversion
1	A/D conversion is in progress	Has no effect on ADC operation

- Read operation to this bit shows the status of A/D conversion.
- This bit is set to "1" at the beginning of A/D conversion and cleared to "0" at the end of A/D conversion.

Notes:

- "1" is read with read-modify-write operation.
- In Single Mode, this bit is cleared at the end of A/D conversion.
- In Continuous Mode and Pause Mode, A/D conversion does not stop until "0" is written to this bit.
- Do not perform the stop operation and the ADC activation (software, reload timer, or external trigger) simultaneously. If this is done, the A/D conversion is terminated since the stop operation has a higher priority.

[bit14] INT: End of Conversion Interrupt Flag

Bit	Description	
	Read	Write
0	End of conversion interrupt condition is not met (ADCR register contains invalid data)	Interrupt request is cleared
1	End of conversion interrupt condition is met (A/D conversion result has been transferred to ADCR register)	No effect

- Regardless of end of conversion interrupt enable setting (ADCS.INTE), INT bit is set to "1" when A/D conversion result is stored in ADCR.
- The end of conversion interrupt is requested when this bit is set to "1" with end of conversion interrupt flag set (INTE="1"). If DMA transfer is configured, this interrupt request activates DMA. This bit is cleared to "0" either by writing "0" to it or by the clear signal from DMA.

Notes:

- When writing "0" to INT while INT="0", make sure that the A/D conversion is not in progress (check that BUSY="0").
- "1" is read with read-modify-write operation.
- If the set condition and the clear condition of INT bit are satisfied simultaneously, INT bit is cleared since the clear operation has a higher priority.

[bit13] INTE: End of Conversion Interrupt Enable

- This bit enables/disables the end of conversion interrupt requests.

Bit	Description
0	End of conversion interrupt request disabled (Conversion Data Protection Function disabled) Interrupt is not requested when the end of conversion interrupt flag is set.
1	End of conversion interrupt request enabled (Conversion Data Protection Function enabled) Interrupt is requested when the end of conversion interrupt flag is set.

- Set this bit to "1" when using DMA (DMA transfer is activated with interrupt request).
- To understand how A/D conversion data is protected, see "Section 3 Conversion Data Protection Function".

Notes:

- Do not set INTE to "0" when using Conversion Data Protection Function.
- If Data Buffer Function is used (ADDBCR:BLRIE="1") with INTE="1", A/D conversion is paused with every completion of A/D conversion. When A/D conversion is paused, the end of conversion interrupt flag must be cleared to "0" (write "0" to INT) in order to resume the conversion. See "Section 4 Data Buffer Function" for further details.

[bit12] PAUS: A/D Conversion Pause

Bit	Description	
	Read	Write
0	A/D conversion is not paused	Clears PAUS bit (no effect on operation)
1	A/D conversion is paused	Sets PAUS bit (no effect on operation)

- This bit is set to "1" when A/D conversion is paused.
- This bit is not cleared to "0" automatically. Therefore, write "0" to clear this bit to "0".
- To understand how A/D conversion data is protected, see "Section 3 Conversion Data Protection Function".
- The proper sequence for the PAUS bit clear operation after the A/D conversion is paused is described below.
  1. Clear the end of conversion interrupt flag (ADCS:INT) either by writing "0" to it.
  2. Clear the A/D conversion pause bit (ADCS:PAUS) by writing "0" to it.

Notes:

- The end of conversion interrupt flag (ADCS:INT) and the A/D conversion pause bit (ADCS:PAUS) cannot be cleared simultaneously.
- A/D conversion pause bit (ADCS.PAUS) cannot be cleared while DMA transfer is in progress. PAUS bit set to "1" indicates that A/D conversion has been paused during DMA transfer. Therefore, make sure to clear the pause bit (writing "0" to ADCS:PAUS) after the DMA transfer is complete.
- During Scan Conversion, unless transferred to memory (by DMA, etc), A/D conversion result stored in ADCR is destroyed every time a new A/D conversion completes. To avoid this, in case INTE="1" and the previous A/D conversion result has not been retrieved (INT="1"), storing the new A/D conversion result is prevented by pausing ADC operation (PAUS is set to "1"). A/D conversion resumes its operation when the retrieval of the previous A/D conversion result is complete or when "0" is written to INT bit.

[bit11, bit10] STS1, STS0: Start Source Select

bit11	bit10	Description		
		Software	External Trigger	Reload Timer
0	0	Enabled	Disabled	Disabled
0	1	Enabled	Enabled	Disabled
1	0	Enabled	Disabled	Enabled
1	1	Enabled	Enabled	Enabled

- These bits select the activation method(s) of A/D conversion.
- In Single Mode 1, if multiple activation methods are enabled, whichever method applied last triggers the next A/D conversion.
- In Single Mode 2, Continuous Mode, and Pause Mode, if multiple activation methods are enabled, whichever method applied first triggers the next A/D conversion.

Notes:

- External trigger activates A/D conversion with falling edge.
- Reload timer activates A/D conversion when it decrements to "0x00".
- Change in the activation method becomes immediately effective. Therefore, do not modify these bits while A/D conversion is in progress (while BUSY="1").

[bit9] STRT: A/D Converter Software Activation

Bit	Description	
	Read	Write
0	"0" is always read	Has no effect on operation
1		Starts A/D conversion

Writing "1" to this bit repeatedly can restart A/D conversion.

Notes:

- In Pause Mode, BUSY is set to "1" even when A/D conversion is not in progress. When performing software reactivation, make sure that INT bit has been cleared from "1". If A/D conversion has been activated by other factors (external trigger or reload timer) and is currently in progress when software activation is performed, reactivation does not occur.
- Do not perform stop operation and software activation (writing "0" to BUSY and writing "1" to STRT) simultaneously.

[bit8] RESV: Reserved Bit

Always write "0" to this bit.

"0" is always read from this bit.

■ A/D Control Status Register Lower (ADCSL)

ADCSL								
bit	7	6	5	4	3	2	1	0
	MD1	MD0	S10	CSEN	RESV			
Attribute	R/W	R/W	R/W	R/W	-	-	-	-
Initial value	0	0	0	0	0	0	0	0

Note:

When reactivating A/D converter in Single Mode 1 (MD1, MD0=00b) with software reactivation (STRT=1), make sure to write “0" to INT bit before the reactivation. Immediately after the reactivation, check the status of INT bit, and if INT="1", write “0" to INT bit once more.

[bit7, bit6] MD1, MD0: A/D Conversion Mode Set

bit7	bit6	Description
0	0	Single mode 1 (reactivation during A/D conversion is possible)
0	1	Single mode 2 (reactivation during A/D conversion is not possible)
1	0	Continuous mode (reactivation during A/D conversion is not possible)
1	1	Pause mode (reactivation during A/D conversion is not possible)

- These bits configure the operation mode of A/D conversion.
- Single Mode
  A/D conversion is performed only once on each of the specified analog input channels starting from the start channel (ANS4-ANS0) and ending with the end channel (ANE4-ANE0). The Single Mode conversion completes when the conversions of all the specified analog input channels are processed. BUSY bit shows "1" when Single Mode conversion is in progress and "0" after Single Mode conversion is finished.
- Continuous Mode
  A/D conversions are repeatedly performed on the sequence of the analog input channels defined by the start channel (ANS4-ANS0) and the end channel (ANE4-ANE0). BUSY bit shows "1" when Continuous Mode conversion is in progress. In order to terminate a Continuous Mode conversion, write "0" to BUSY bit.
- Pause Mode
  The operation is the same as Continuous Mode, except that the A/D conversion is paused every time the conversion on a channel finishes. Use the reactivation method(s) selected by STS1-STS0 to initiate the conversion on the next channel. BUSY bit shows "1" when Pause Mode conversion is in progress. In order to terminate a Pause Mode conversion, write "0" to BUSY bit.

Notes:

- The conversions in Continuous Mode and Pause Mode continue until "0" is written to BUSY bit.
- In Single Mode 2, Continuous Mode, and Pause Mode, reactivation (software, external trigger, and reload timer) while the A/D conversion is in progress is ignored.
- Do not modify these bits while the A/D conversion is in progress (while BUSY="1")
- Do not restart A/D conversion when using Data Buffer Function. As a result of restarting, the desired mapping between ADDBCR0-7 and analog input channels might be lost. See "Section 4 Data Buffer Function" for further details.

[bit5] S10: Resolution of A/D Conversion

Bit	Description
0	10-bit resolution (A/D conversion results are stored in D9-D0)
1	8-bit resolution (A/D conversion results are stored in D7-D0)

- This bit configures the resolution of A/D conversion.
- The table below describes how A/D conversion results are stored in A/D Data Register (ADCR) and A/D Data Buffer Register (ADCBR) depending on the resolution setting.

Bit	ADCR	ADCBR
0	A/D conversion results are stored in ADCR:D9-D0	A/D conversion results are stored in ADCBR:D9-D0
1	A/D conversion results are stored in ADCR:D7-D0	A/D conversion results are stored in ADCBR:D7-D0

Note:

Do not modify this bit while the A/D conversion is in progress (while BUSY="1")

[bit4] CSEN: Scan Disable Function Enable

Bit	Description
0	Scan Disable Function disabled
1	Scan Disable Function enabled

- This bit enables/disables Scan Disable Function.

See "Section 6 Scan Disable Function" for further details.

Notes:

- Write "0" to this bit if the device is not equipped with Scan Disable Function.
- Refer to the datasheet to see whether Scan Disable Function is present on the device.
- Do not modify this bit while the A/D conversion is in progress (while BUSY="1")

[bit3 to bit0] RESV: Reserved Bits

Always write "0" to this bit.

"0" is always read from this bit.

8.2. A/D Data Register (ADCR)

The A/D data register is used to store the digital value generated as a result of conversion. The register value is rewritten every time a conversion ends.

■ A/D Data Register (ADCR)

ADCRH								
bit	15	14	13	12	11	10	9	8
	RESV						D9	D8
Attribute	-	-	-	-	-	-	R	R
Initial value	0	0	0	0	0	0	0	0

ADCRL								
bit	7	6	5	4	3	2	1	0
	D7	D6	D5	D4	D3	D2	D1	D0
Attribute	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

[bit15 to bit10] RESV: Reserved Bits

The read value is "0".

[bit9 to bit0] D9 to D0: A/D Data

Bit	Description
	These bits store the results of the A/D converter. The register values are updated at the completion of each conversion. For 8-bit resolution the result is provided in D7 to D0. For 10-bit resolution the result is provided in D9 to D0.

- If the end of conversion interrupt is disabled (ADCS:INTE="0"), these bits store the result of the last conversion.
- These bits can be protected by Conversion Data Protection Function.
- If the end of conversion interrupt is enabled (ADCS:INTE="1"), these bits store the result of the conversion that took place when the end of conversion interrupt condition was met (ADCS:INT="1") last time.
- The result of read operation differs depending on the resolution setting.

Bit	Resolution	Result of Read Operation
0	10-bit	Bit9 to bit0 contain the A/D conversion result and can be read
1	8-bit	Bit7 to bit0 contain the A/D conversion result and can be read, but "0" is read from bit9, bit8

Notes:

- Do not write to these bits.
- When using 10-bit resolution mode (ADCS:S10="0"), use word access only to access A/D data register (ADCR).
- To understand how A/D conversion data is protected, see "Section 3 Conversion Data Protection Function".
- Clearing end of conversion interrupt request (writing "0" to ADCS:INT) must be performed after A/D data register (ADCR) is read.

8.3. A/D Setting Register (ADSR)

The A/D setting register (ADSR) configures the comparison time and the sampling time of A/D conversion and shows the current analog input channel being processed.

■ A/D Setting Register (ADSR)

ADSRH								
bit	15	14	13	12	11	10	9	8
	ST2	ST1	ST0	CT2	CT1	CT0	ANS4	ANS3
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

ADSRL								
bit	7	6	5	4	3	2	1	0
	ANS2	ANS1	ANS0	ANE4	ANE3	ANE2	ANE1	ANE0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Notes:

- Single Channel Conversion is performed if the end channel (ANE4 to ANE0) equals the start channel (ANS4 to ANS0).
- In Continuous Mode and Pause Mode, after the conversion finishes for the end channel (ANE4 to ANE0), the A/D conversion continues by returning to the start channel (ANS4 to ANS0).
- When Scan Disable Function is disabled (ADCS:CSEN=”0”), if ANS4 to ANS0 and ANE4 to ANE0 are configured so that ANS4 to ANS0>ANE4 to ANE0, the conversion starts with the start channel (ANS4 to ANS0), continues consecutively up to AN31, goes back to AN0, and finally finishes with the end channel (ANE4 to ANE0) (i.e. Start channel → ··· → AN31 → AN0 → ··· → End channel).
- When Scan Disable Function is enabled (ADCS:CSEN="1"), do not configure ANE4 to ANE0 and ANS4 to ANS0 to be greater than the number of the analog input channels equipped on the device.
- To see the number of channels equipped on the device, refer to the datasheet.
- ANS4 to ANS0 and ANE4 to ANE0 bits are valid for both using Data Register (ADCR) and Data Buffer (ADCBR0 to ADCBR7) as the destination of the conversion results.

[bit15 to bit13] ST2 to ST0: Analog Input Sampling Time Set

bit15	bit14	bit13	Description
0	0	0	Sampling time = 4 CLKP1 cycles
0	0	1	Sampling time = 6 CLKP1 cycles
0	1	0	Sampling time = 8 CLKP1 cycles
0	1	1	Sampling time = 12 CLKP1 cycles
1	0	0	Sampling time = 24 CLKP1 cycles
1	0	1	Sampling time = 36 CLKP1 cycles
1	1	0	Sampling time = 48 CLKP1 cycles
1	1	1	Sampling time = 128 CLKP1 cycles

- These bits configure the analog input sampling time.
- Refer to the datasheet to see which setting to chose.

Note:

Do not modify these bits while the A/D conversion is in progress (while ADCS:BUSY="1")

[bit12 to bit10] CT2 to CT0: A/D Comparison Time Set

bit12	bit11	bit10	Description
0	0	0	Comparison time = 22 CLKP1 cycles
0	0	1	Comparison time = 33 CLKP1 cycles
0	1	0	Comparison time = 44 CLKP1 cycles
0	1	1	Comparison time = 66 CLKP1 cycles
1	0	0	Comparison time = 88 CLKP1 cycles
1	0	1	Comparison time = 132 CLKP1 cycles
1	1	0	Comparison time = 176 CLKP1 cycles
1	1	1	Comparison time = 264 CLKP1 cycles

- These bits configure the comparison time of successive approximation.
- Refer to the datasheet to see which setting to chose.

Note:

Do not modify these bits while the A/D conversion is in progress (while ADCS:BUSY="1")

[bit9 to bit5] ANS4 to ANS0: Analog Start Channel Set

bit9	bit8	bit7	bit6	bit5	Description
0	0	0	0	0	AN0
0	0	0	0	1	AN1
0	0	0	1	0	AN2
. . .					
1	1	1	0	1	AN29
1	1	1	1	0	AN30
1	1	1	1	1	AN31

- These bits configure the analog input channel from which A/D conversion starts. Also, the analog input channel that is currently processed is returned by performing read operation on these bits.
- Read operation:
  - Indicates the analog input channel being converted if A/D conversion is in progress.
  - Indicates the last converted analog input channel if A/D conversion is not in progress.
  - If a new value is written to these bits before A/D conversion starts, the last converted analog input channel is returned with read operation.
- Write operation:
  - Configures the analog input channel from which A/D conversion starts.

Notes:

- The A/D conversion result stored in ADCR does not correspond to the analog input channel number read from ANS4-0, except for Single Channel Conversion.
- Always use word access when performing write operation on these bits. If byte access or read-modify-write operation is performed on these bits, A/D conversion might start beginning from an unexpected analog input channel.

[bit4 to bit0] ANE4 to ANE0: Analog End Channel Set

bit4	bit3	bit2	bit1	bit0	Description
0	0	0	0	0	AN0
0	0	0	0	1	AN1
0	0	0	1	0	AN2
. . .					
1	1	1	0	1	AN29
1	1	1	1	0	AN30
1	1	1	1	1	AN31

- These bits configure the analog input channel with which A/D conversion ends.

8.4. A/D Data Buffer Control Register (ADDBCR)

The data buffer control register configures the number of data buffer registers to be used, buffer start channel, interrupt enable/disable, and wrap around function enable/disable. With read operation, it shows the current buffer pointer and indicates whether or not the data buffer has reached its limit.

■ A/D Data Buffer Control Register Higher (ADDBCRH)

ADDBCRH								
bit	15	14	13	12	11	10	9	8
	WRAP	BLRF	BLRIE	BSC4	BSC3	BSC2	BSC1	BSC0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	1	1	1	1	1

[bit15] WRAP: Wrap around Function Enable

Bit	Description
0	Wrap around function is disabled.
1	Wrap around function is enabled.

- This bit enables/disables data buffer's wrap around function.
- With WRAP="0", when the buffer limit is reached (BLRF="1") no further update is made to the data buffer, and the current buffer pointer (BLC2-0) stops. To resume buffering, make a write access to ADDBCRL.
- With WRAP="1", when the buffer limit is reached (BLRF="1") the current buffer pointer (BLC2-0) is automatically cleared to "000_B", and buffering continues with ADCBR0.

Notes:

- Do not modify this bit while the A/D conversion is in progress (while ADCS:BUSY="1").
- Conversion Data Protection Function is not triggered by the data buffer. See "Section 4 Data Buffer Function" for further details.

[bit14] BLRF: Buffer Limit Reached Flag

Bit	Description	
	Read	Write
0	Buffer limit reached interrupt condition is not met. The number of the conversion results stored in the data buffer is less than the buffer level limit (BLL2-0).	Clears the buffer limit reached interrupt request.
1	Buffer limit reached interrupt condition is met. The number of the conversion results stored in the data buffer has reached the buffer level limit (BLL2-0).	No effect

- This bit indicates whether or not the buffer level limit (BLL2-0) has been reached.
- This bit is set to "1" when the data buffer register defined by the buffer limit level (BLL2-0) is written.
- Regardless of the buffer limit reached interrupt enable setting (ADDBCR:BLRIE), BLRF bit is always set to "1" when current buffer pointer (ADDBCR:BLC2-0) reaches the value of the buffer level limit (ADDBCR).
- An interrupt is caused when this bit is set to "1" with BLRIE="1". This bit can be cleared either by writing "0" to it, by DMA clear signal, or by writing to ADDBCRL.

Notes:

- If the set condition and the clear condition of BLRF bit are satisfied simultaneously, BLRF bit is cleared since the clear operation has a higher priority. Therefore, with WRAP="0", when clearing this bit by writing "0" to it, make sure that A/D conversion is not in progress (ADCS:BUSY="0") or that A/D conversion is in progress (ADCS:BUSY="1") with BLRF="1".
- If WRAP="0", clearing this bit to "0" when the buffer limit is reached does not resume buffering.
- This bit is cleared when a write access is made to ADDBCRL since a value is written to BLC2-0.
- "1" is read with read-modify-write operation.

[bit13] BLRIE: Buffer Limit Reached Interrupt Enable

Bit	Description
0	Buffer level reached interrupt disabled Interrupt is not requested when the buffer limit reached interrupt condition is met (BLRF=”1”).
1	Buffer level reached interrupt enabled Interrupt is requested when the buffer limit reached interrupt condition is met (BLRF=”1”).

- This bit enables/disables the buffer limit reached interrupt.

Note:

If Data Buffer Function is used with ADCS:INTE="1", A/D conversion is paused every time the conversion finishes, and the end of conversion flag (ADCS:INT) must be cleared in order to resume the conversion. See "Section 4 Data Buffer Function" for further details.

[bit12 to bit8] BSC4 to BSC0: Buffer Start Channel

bit12	bit11	bit10	bit9	bit8	Description
0	0	0	0	0	AN0 is buffer start channel
0	0	0	0	1	AN1 is buffer start channel
			. .		
1	1	1	1	1	AN31 is buffer start channel

- These bits define the analog input channel from which the data buffer starts storing the A/D conversion data.
- These bits specify the lowest analog input channel from which the buffer register is to be stored with the converted values. All analog input channels with the same or a higher number are written into the buffer registers. Note that the data buffer only stores the conversion results of analog input channels that are not skipped by Scan Disable Function.

Notes:

- When BSC4-0 is configured to be greater than the number of analog input channels equipped on the device, the A/D conversion results are not stored in the data buffer.
- To see the number of channels equipped on the device, refer to the datasheet.

■ A/D Data Buffer Control Register Lower (ADDBCRL)

ADDBCRL								
bit	7	6	5	4	3	2	1	0
	RESV		BLC2	BLC1	BLC0	BLL2	BLL1	BLL0
Attribute	-	-	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	1	1	1

Notes:

- With WRAP="0" (overwrite to data buffer disabled), when buffering is stopped with the buffer limit reached (BLRF="1"), writing to the lower byte of ADDBCR (ADDBCRL) restarts buffering. Buffering is restarted with the data buffer register corresponding to the value written to BLC2 to BLC0.
- With WRAP="1" (overwrite to data buffer enabled), when the buffer limit is reached (BLRF="1") the current buffer pointer (BLC2 to BLC0) is automatically cleared to “000b", and buffering continues with ADCBR0.
- This bit is cleared when a write access is made to ADDBCRL since a value is written to BLC2 to BLC0.
- If the current buffer pointer is set to be larger than the buffer level limit (ADDBCR:BLC > ADDBCR:BLL), buffering starts with the data buffer register specified by ADDBCR:BLC, continues up to ADCBR7, goes back to ADCBR0, and ends with the data buffer register specified by ADDBCR:BLL. See "Section 4. Data Buffer Function" for more details.

[bit7, bit6] RESV: Reserved Bits

Always write "0" to this bit.

"0" is always read from this bit.

[bit5 to bit3] BLC2 to BLC0: Current Buffer Pointer

bit5	bit4	bit3	Description
0	0	0	ADCBR0 is the current buffer
0	0	1	ADCBR1 is the current buffer
0	1	0	ADCBR2 is the current buffer
0	1	1	ADCBR3 is the current buffer
1	0	0	ADCBR4 is the current buffer
1	0	1	ADCBR5 is the current buffer
1	1	0	ADCBR6 is the current buffer
1	1	1	ADCBR7 is the current buffer

- These bits indicate the value of the current buffer pointer.
- The current buffer pointer is returned with read operation.
- Write operation on these bits modifies the current pointer.

Note:

These bits can be written anytime regardless of the state of the data buffer. If these bits are written while buffering (WRAP="0", or WRAP="1" with A/D conversion in progress (ADCS:BUSY="1") and the buffer limit not reached (BLRF=0)), the mapping between ADDBCR0-7 and analog input channels might be lost.

[bit2 to bit0] BLL2 to BLL0: Buffer Level Limit

bit2	bit1	bit0	Description
0	0	0	ADCBR0 is the last register to be used
0	0	1	ADCBR1 is the last register to be used
0	1	0	ADCBR2 is the last register to be used
0	1	1	ADCBR3 is the last register to be used
1	0	0	ADCBR4 is the last register to be used
1	0	1	ADCBR5 is the last register to be used
1	1	0	ADCBR6 is the last register to be used
1	1	1	ADCBR7 is the last register to be used

- These bits define the number of the buffer registers to be used.
- The data buffer stores the A/D conversion results starting from the buffer register 0 (ADDCBR0) up to the register specified in BLL2 to BLL 0.

8.5. A/D Data Buffer Registers (ADCBR0-7)

The A/D data buffer registers store multiple A/D conversion results. The number of data buffer registers to be used is configurable.

■ A/D Data Buffer Register (ADCBR0-7)

ADCBRH0-7

bit	15	14	13	12	11	10	9	8
	RESV						D9	D8
Attribute	-	-	-	-	-	-	R	R
Initial value	0	0	0	0	0	0	0	0

ADCBRL0-7

bit	7	6	5	4	3	2	1	0
	D7	D6	D5	D4	D3	D2	D1	D0
Attribute	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

[bit15 to bit10] RESV: Reserved Bits

The read value is "0".

[bit9 to bit0] D9 to D0: A/D Data Bits

Bit	Description
These bits store the result of the A/D converter. For 10-bit resolution the result is provided in D9 to D0. For 8-bit resolution the result is provided in D7 to D0, and "0" is read from D9, D8 with read operation.	

- A/D conversion result is stored in these bits if Data Buffer Function is enabled (ADDBCR configured accordingly).
- The result of read operation differs depending on the resolution setting.

Bit	Resolution	Description
0	10-bit	Bit9 to bit0 contain the A/D conversion result and can be read
1	8-bit	Bit7 to bit0 contain the A/D conversion result and can be read, but "0" is read from bit9, bit8

- A/D conversion results are stored starting from the buffer start channel (ADDBCR:BSC4-0).
- The number of the data buffer registers to be used can be configured with ADDBCR:BLL2-0.
- The results of Scan Conversion are stored in the data buffer registers in the consecutive order.

Notes:

- Do not write to these bits.
- When using 10-bit resolution mode, make word access only to access A/D data buffer registers (ADCBR0-7).
- Note that only the analog input channels not skipped by Scan Disable Function are stored in the data buffer registers (ADCBR0-7). See "Section 6 Scan Disable Function" for further details.

8.6. Upper Threshold Registers for Range Comparator (ADRCOH0-3)

The upper threshold registers for range comparator configure the upper threshold value to which the A/D conversion result is compared. Four different threshold values can be configured with these registers.

■ Upper Threshold Registers for Range Comparator Higher (ADRCOH0-3)

ADRCOHH0-3

bit	15	14	13	12	11	10	9	8
	RESV						C9	C8
Attribute	-	-	-	-	-	-	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

ADRCOHL0-3

bit	7	6	5	4	3	2	1	0
	C7	C6	C5	C4	C3	C2	C1	C0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

[bit15 to bit10] RESV: Reserved Bits

Always write "0" to this bit.

"0" is always read from this bit.

[bit9 to bit0] C9 to C0: Upper Threshold Value

Bit	Description
These bits set the upper threshold value. For 10-bit resolution, set the upper threshold value using C9 to C0. For 8-bit resolution, set the upper threshold value using C7 to C0. For 8-bit resolution, "0" is read from C9, C8 with read operation.	

- These bits configure the upper threshold value.
- Four different threshold values can be configured.
- Perform write operation in accordance with the resolution setting.

Bit	Resolution	Description
0	10-bit	Write threshold value to C9-0.
1	8-bit	Write threshold value to C7-0. Write "0" to C9-8.

Note:

Do not modify these bits when the A/D conversion is in progress (while ADCS:BUSY="1")

8.7. Lower Threshold Registers for Range Comparator (ADRCOL0-3)

The lower threshold registers for range comparator configure the lower threshold value to which the A/D conversion result is compared. Four different threshold values can be configured with these registers.

■ Lower Threshold Registers for Range Comparator Lower (ADRCOL0-3)

ADRCOLH0-3

bit	15	14	13	12	11	10	9	8
	RESV						C9	C8
Attribute	-	-	-	-	-	-	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

ADRCOLL0-3

bit	7	6	5	4	3	2	1	0
	C7	C6	C5	C4	C3	C2	C1	C0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

[bit15 to bit10] RESV: Reserved Bits

Always write "0" to this bit.

"0" is always read from this bit.

[bit9 to bit0] C9 to C0: Lower Threshold Value

Bit	Description
These bits set the lower threshold value. For 10-bit resolution, set the lower threshold value using C9 to C0. For 8-bit resolution, set the lower threshold value using C7 to C0. For 8-bit resolution, "0" is read from C9, C8 with read operation.	

- These bits configure the lower threshold value.
- Four different threshold values can be configured.
- Perform write operation in accordance with the resolution setting.

Bit	Resolution	Description
0	10-bit	Write threshold value to C9-0.
1	8-bit	Write threshold value to C7-0. Write "0" to C9-8.

Note:

Do not modify these bits when the A/D conversion is in progress (while ADCS:BUSY="1")

8.8. Channel Control Registers for Range Comparator (ADCC0-15)

The channel control registers for range comparator configure the upper/lower threshold values used for the comparison, range comparator enable/disable, and range comparator interrupt enable/disable, for each analog input channel. One byte serves two analog input channels together.

■ Channel Control Registers for Range Comparator (ADCC0-15)

ADCCj

bit	15	14	13	12	11	10	9	8
	RCOIE (2j+1)	RCOE (2j+1)	RCOS (2j+1)1	RCOS (2j+1)0	RCOIE (2j)	RCOE (2j)	RCOS (2j)1	RCOS (2j)0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

ADCCk

bit	7	6	5	4	3	2	1	0
	RCOIE (2k+1)	RCOE (2k+1)	RCOS (2k+1)1	RCOS (2k+1)0	RCOIE (2k)	RCOE (2k)	RCOS (2k)1	RCOS (2k)0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Notes:

- j = 1, 3, 5, 7, 9, 11, 13, 15
- k =0, 2, 4, 6, 8, 10, 12, 14
- "n" indicates the analog input channel number (0-31).

[bit15, bit11, bit7, bit3] RCOIEn: Range Comparator Interrupt Enable

Bit	Description
0	Range comparator interrupt is disabled Interrupt is not requested when the range comparator interrupt flag is set (ADRCOINTF:RCOINT="1").
1	Range comparator interrupt is enabled Interrupt is requested when the range comparator interrupt flag is set (ADRCOINTF:RCOINT="1").

- This bit enables/disables the range comparator interrupt.
- Each bit corresponds to a particular analog input channel.
  ADCC0: bit3 corresponds to AN0
  ADCC0: bit7 corresponds to AN1
  .
  .
  ADCC15: bit15 corresponds to AN31

Note:

Write "0" to the bits corresponding to the analog input channels that do not exist on the device.

[bit14, bit10, bit6, bit2] RCOEn: Range Comparator Channel Enable

Bit	Description
0	Range comparison is not performed
1	Range comparison is performed

- This bit determines whether or not the range comparison is performed for analog input channel n.
- Each bit corresponds to a particular analog input channel.
  ADCC0: bit2 corresponds to AN0
  ADCC0: bit6 corresponds to AN1
  .
  .
  ADCC15: bit14 corresponds to AN31

Note:

Write "0" to the bits corresponding to the analog input channels that do not exist on the device.

[bit13, bit12, bit9, bit8, bit5, bit4, bit1, bit0] RCOSn1-n0: Range Comparator Threshold Selection

bit13, bit9, bit5, bit1	bit12, bit8, bit4, bit0	Description
0	0	ADRCOH0/ADRCOL0 is selected as upper/lower threshold value
0	1	ADRCOH1/ADRCOL1 is selected as upper/lower threshold value
1	0	ADRCOH2/ADRCOL2 is selected as upper/lower threshold value
1	1	ADRCOH3/ADRCOL3 is selected as upper/lower threshold value

- These bits select the upper/lower threshold value used for the range comparison.
- Each bit corresponds to a particular analog input channel.
  ADCC0: bit1, bit0 correspond to AN0
  ADCC0: bit5, bit4 correspond to AN1
  .
  .
  ADCC15: bit13, bit12 correspond to AN31

Note:

Do not modify these bits while the A/D conversion is in progress (while ADCS:BUSY="1").

8.9. Inverted Range Selection Registers for Range Comparator (ADRCOIRS0-3)

The inverted range selection registers for range comparator controls whether the comparison is for "out of range" or "in range". These bits are organized as per analog input channel.

■ Inverted Range Selection Registers for Range Comparator (ADRCOIRS0-3)

ADRCOIRS x/8

Bit	x%16	(x-1)%16	(x-2)%16	(x-3)%16	(x-4)%16	(x-5)%16	(x-6)%16	(x-7)%16
	RCOIRS x	RCOIRS (x-1)	RCOIRS (x-2)	RCOIRS (x-3)	RCOIRS (x-4)	RCOIRS (x-5)	RCOIRS (x-6)	RCOIRS (x-7)
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

ADRCOIRS0

Bit	7	6	5	4	3	2	1	0
	RCOIRS 7	RCOIRS 6	RCOIRS 5	RCOIRS 4	RCOIRS 3	RCOIRS 2	RCOIRS 1	RCOIRS 0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Notes:

- x : max number of channels
- "n" indicates the analog input channel number (0-31).
- Do not modify these bits while the A/D conversion is in progress (while ADCS:BUSY="1").

[bit15 to bit0] RCOIRSn: In Range Register Bits

Bit	Description
0	Range comparator will check for "outside range" values (Checks whether the conversion result is outside of the range defined by the selected upper/lower threshold.)
1	Range comparator will check for "inside range" values (Checks whether the conversion result is inside of the range defined by the selected upper/lower threshold.)

- These bits determine what the range comparator checks for.
- Each bit corresponds to a particular analog input channel.
  ADRCOIRS0: bit0 corresponds to AN0
  ADRCOIRS0: bit1 corresponds to AN1
  .
  .
  ADRCOIRS3: bit15 corresponds to AN31

- The upper/lower threshold value used for the comparison is selected with ADCC:RCOSn1/RCOSn0.

Bit	Description

0	· The range comparator interrupt is generated if the conversion result is "out of range" with the range comparator interrupt enabled (ADCC:RCOIEn="1").
	· The conversion result is said to be "out of range" if one of the following conditions are satisfied:
	  · (A/D conversion result) > (upper threshold value)
	  · (A/D conversion result) < (lower threshold value)
	· When the conversion result is "out of range", the range comparator over threshold flag (ADRCOOF:RCOOFn) indicates whether it is higher than the upper threshold or lower than the lower threshold.

1	· The range comparator interrupt is generated if the conversion result is "in range" with the range comparator interrupt enabled (ADCC:RCOIEn="1").
	· The conversion result is said to be "in range" if both of the following conditions are satisfied:
	  · (A/D conversion result) ≦(upper threshold value)
	  · (A/D conversion result) ≧ (lower threshold value)

8.10. Range Comparator Over Threshold Flag Registers (ADRCOOF0-3)

The range comparator over threshold flag registers indicates the comparison result of the range comparator for each analog input channel.

■ Range Comparator Over Threshold Flag Registers (ADRCOOF0-3)

ADRCOOF x/8								
bit	x%16	(x-1)%16	(x-2)%16	(x-3)%16	(x-4)%16	(x-5)%16	(x-6)%16	(x-7)%16
	RCOOF x	RCOOF (x-1)	RCOOF (x-2)	RCOOF (x-3)	RCOOF (x-4)	RCOOF (x-5)	RCOOF (x-6)	RCOOF (x-7)
Attribute	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

ADRCOOF0								
bit	7	6	5	4	3	2	1	0
	RCOOF 7	RCOOF 6	RCOOF 5	RCOOF 4	RCOOF 3	RCOOF 2	RCOOF 1	RCOOF 0
Attribute	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

Notes:

- x :max number of channels
- "n" indicates the analog input channel number (0-31).

[bit15 to bit0] RCOOFn: Range Comparator Over Threshold Flag

Bit	Description
0	The result of the A/D conversion is below the lower threshold value (A/D conversion result of analog input channel n is smaller than the corresponding lower threshold value)
1	The result of the A/D conversion is above the upper threshold value (A/D conversion result of analog input channel n is larger than the corresponding upper threshold value)

- These bits indicate the comparison result of the range comparator.
- Each bit corresponds to a particular analog input channel.
  ADRCOOF0: bit0 corresponds to AN0
  ADRCOOF0: bit1 corresponds to AN1
  .
  .
  ADRCOOF3: bit15 corresponds to AN31
- The upper/lower threshold value used for the comparison is selected with ADCCn:RCOSn1/RCOSn0.

Notes:

- The range comparator over threshold flags have different meanings depending on the setting of ADRCOIRS:RCOIRSn.
- The table below describes the behavior of RCOOFn bit with the range comparator interrupt flag set from "0" to "1" (ADRCOINTF:RCOINTn="0" →"1"). RCOOFn bit maintains its previous value if the range comparator interrupt flag has already been set to "1" (ADRCOINTF:RCOINTn="1").

Bit	Description	
	RCOIRSn=0 (check for "out of range")	RCOIRSn=1 (check for "in range")
RCOOFn=0	A/D conversion result of analog input channel n is smaller than the corresponding lower threshold value	The value of RCOOFn has no meaning.
RCOOFn=1	A/D conversion result of analog input channel n is larger than the corresponding upper threshold value	

- "n" indicates the analog input channel number (0-31).

8.11. Range Comparator Flag Registers (ADRCOINTF0-3)

The flags in the range comparator over threshold flag registers indicate whether the range comparator interrupt condition is met for each analog input channel.

■ Range Comparator Flag Registers (ADRCOINTF0-3)

ADRCOINTF x/8								
bit	x%16	(x-1)%16	(x-2)%16	(x-3)%16	(x-4)%16	(x-5)%16	(x-6)%16	(x-7)%16
	RCOINTx	RCOINT(x-1)	RCOINT(x-2)	RCOINT(x-3)	RCOINT(x-4)	RCOINT(x-5)	RCOINT(x-6)	RCOINT(x-7)
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0
	|	|	|	|	|	|	|	|
	|	|	|	|	|	|	|	|
	|	|	|	|	|	|	|	|
	|	|	|	|	|	|	|	|
	|	|	|	|	|	|	|	|

ADRCOINTF0								
bit	7	6	5	4	3	2	1	0
	RCOINT7	RCOINT6	RCOINT5	RCOINT4	RCOINT3	RCOINT2	RCOINT1	RCOINT0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Notes:

- x : max number of channels
- "n" indicates the analog input channel number (0-31).

[bit15 to bit0] RCOINTn: Range Comparator Interrupt Flag

Bit	Description	
	Read	Write
0	Range comparator interrupt condition not met	Interrupt request is cleared
1	Range comparator interrupt condition met	Has no effect on operation

- These bits indicate whether the range comparator interrupt condition is met. The interrupt request can be cleared by writing "0" to it.
- Each bit corresponds to a particular analog input channel.
  ADRCOINTF0: bit0 corresponds to AN0
  ADRCOINTF0: bit1 corresponds to AN1
  .
  .
  ADRCOINTF3: bit15 corresponds to AN31
- Regardless of the range comparator interrupt enable setting (ADCC:RCOIEn), RCOINTn bit is set to "1" when Range Comparator Function detects the interrupt request condition.
- The upper/lower threshold value used for the comparison is selected with ADCCn:RCOSn1/RCOSn0.

Notes:

- "1" is read with read-modify-write operation.
- If the set condition and the clear condition of RCOINTn bit are satisfied simultaneously, RCOINTn bit is set to "1" since the set operation has a higher priority.

8.12. Disable Channel for Scan Register (ADDSCR0-3)

The disable channel for scan register determines which analog input channels are to be skipped during Scan Conversion.

■ Disable Channel for Scan Register (ADDSCR0-3)

ADDSCR x/8

Bit	x%16	(x-1)%16	(x-2)%16	(x-3)%16	(x-4)%16	(x-5)%16	(x-6)%16	(x-7)%16
	DCS x	DCS (x-1)	DCS (x-2)	DCS (x-3)	DCS (x-4)	DCS (x-5)	DCS (x-6)	DCS (x-7)
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

ADDSCR0

Bit	7	6	5	4	3	2	1	0
	DCS 7	DCS 6	DCS 5	DCS 4	DCS 3	DCS 2	DCS 1	DCS 0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Notes:

- x : max number of channels
- "n" indicates the analog input channel number (0-31).

[bit15 to bit0] DCSn: Disable Channel for Scan

Bit	Description
0	Corresponding channel is not skipped
1	Corresponding channel is skipped

- These bits control Scan Disable Function.
- Each bit corresponds to a particular analog input channel.
  ADDSCR0: bit0 corresponds to AN0
  ADDSCR0: bit1 corresponds to AN1
  .
  .
  ADDSCR3: bit15 corresponds to AN31
- The conversions of the unused analog input channels, within the range of the analog input channels defined by the start channel (ADSR:ANS4-0) and the end channel (ADSR:ANE4-0), can be skipped by setting the corresponding bits.

Notes:

- Do not skip all the analog input channels defined by the start channel (ADSR:ANS4-0) and the end channel (ADSR:ANE4-0).
- Do not modify these bits while the A/D conversion is in progress (while ADCS:BUSY="1").
- Write "1" to the bits corresponding to the analog input channels that do not exist on the device.

8.13. ADC Pulse Counter Reload Registers (ADPCTNRD0-31/ADPCTPRD0-31)

The ADC pulse counter reload registers hold the initial value of the positive counter and the negative counter of the ADC Pulse Detection Function for each analog input channel.

■ ADC Pulse Negative Counter Reload Registers (ADPCTNRD0-31)

ADPCTNRD0-31								
bit	15	14	13	12	11	10	9	8
	RESV			NRL4	NRL3	NRL2	NRL1	NRL0
Attribute	-	-	-	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

[bit15 to bit13] RESV: Reserved Bits

Always write "0" to this bit.

"0" is always read from this bit.

[bit12 to bit8] NRL4 to NRL0: ADC Pulse Negative Counter Reload Value

Bit	Description
	Holds the start value of the negative counter

- See "Section 7 ADC Pulse Detection Function" for further details.

Note:

When the value of ADC pulse negative counter reload value (ADPCTNRDn:NRL4-0) is "0x00", the negative counter is disabled.

■ ADC Pulse Positive Counter Reload Registers (ADPCTPRD0-31)

ADPCTPRD0-31								
bit	7	6	5	4	3	2	1	0
	PRL7	PRL6	PRL5	PRL4	PRL3	PRL2	PRL1	PRL0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

[bit7 to bit0] PRL7 to PRL0: ADC Pulse Positive Counter Reload Value

Bit	Description
Holds the start value of the positive counter	

- See "Section 7 ADC Pulse Detection Function" for further details.

Note:

When the value of ADC pulse positive counter reload value (ADPCTPRDn:PRL7-0) is "0x00", the Pulse Detection Function is disabled.

8.14. ADC Pulse Counter Registers (ADPCTNCT0-31/ADPCTPCT0-31)

The ADC pulse counter registers indicate the current values of the positive and negative counters. ADPCTNCT0-31 show the current value of the negative counter, while ADPCTPCT0-31 show the current value of the positive counter. Each analog input channel has its own positive/negative counter.

■ ADC Pulse Negative Counter (ADPCTNCT0-31)

ADPCTNCT0-31								
bit	15	14	13	12	11	10	9	8
	RESV			NCR4	NCR3	NCR2	NCR1	NCR0
Attribute	-	-	-	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

[bit15 to bit13] RESV: Reserved Bits
The read value is "0".

[bit12 to bit8] NCR4 to NCR0: ADC Pulse Negative Counter

Bit	Description
Shows the current value of the negative counter	

- The negative counter is a down counter (it decrements).
- The negative counter is initialized with the ADC pulse negative counter reload value (ADPCTNRDn) when
  1. a positive event occurs.
  2. "0" is written to the pulse counter zero flag (ADPCZF:CTPCZFn).

Notes:

- "n" indicates the analog input channel number (0-31).
- When the negative counter (ADPCTNCTn:NCR4-0) has the value "0x00", the negative counter for the corresponding analog input channel becomes inactive.
- If the pulse counter zero flag (ADPCFZ:CTPZFn) is set to "1" for the corresponding analog input channel, the negative counter (ADPCTNCTn) stops.

■ ADC Pulse Positive Counter (ADPCTPCT0-31)

ADPCTPCT0-31

bit	7	6	5	4	3	2	1	0
	PCR7	PCR6	PCR5	PCR4	PCR3	PCR2	PCR1	PCR0
Attribute	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

[bit7 to bit0] PCR7 to PCR0: ADC Pulse Positive Counter

Bit	Description
Shows the current value of the positive counter	

- The positive counter is a down counter (it decrements).
- The positive counter is initialized with the ADC pulse positive counter reload value (ADPCTPRDn) when
  3. the negative counter (ADPCTNRTn) decrements from "0x01" to "0x00"
  4. "0" is written to the pulse counter zero flag (ADPCFZ:CTPCZFn).

Notes:

- "n" indicates the analog input channel number (0-31).
- When the positive counter (ADPCTPCTn:PCR7-0) has the value "0x00", Pulse Detection Function for the corresponding analog input channel becomes inactive.
- If the pulse counter zero flag (ADPCFZ:CTPZFn) is set to "1" for the corresponding analog input channel, the positive counter (ADPCTPCTn) stops.
- In order to activate Pulse Detection Function and initialize the ADC pulse counter register (ADPCTNCTn) to the reload value (ADPCTNRDn), write "0" to the pulse counter zero flag (ADPCZF3-0:CTPCZFn).

8.15. ADC Pulse Counter Interrupt Enable Registers (ADPCIE0-3)

The ADC pulse counter interrupt enable registers enable/disable the ADC pulse counter interrupt.

■ ADC Pulse Counter Interrupt Enable Registers (ADPCIE0-3)

ADPCIE x/8

bit	x%16	(x-1)%16	(x-2)%16	(x-3)%16	(x-4)%16	(x-5)%16	(x-6)%16	(x-7)%16
	CTPIE x	CTPIE (x-1)	CTPIE (x-2)	CTPIE (x-3)	CTPIE (x-4)	CTPIE (x-5)	CTPIE (x-6)	CTPIE (x-7)
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

ADPCIE 0

bit	7	6	5	4	3	2	1	0
	CTPIE 7	CTPIE 6	CTPIE 5	CTPIE 4	CTPIE 3	CTPIE 2	CTPIE 1	CTPIE 0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Notes:

- x : max number of channels
- "n" indicates the analog input channel number (0-31).

[bit15 to bit0] CTPIEn: ADC Pulse Counter Interrupt Enable

Bit	Description
0	ADC pulse detection counter interrupt is disabled Interrupt is not requested when the ADC pulse counter interrupt flag is set (ADPCZF:CTPZFn="1").
1	ADC pulse detection counter interrupt is enabled Interrupt is requested when the ADC pulse counter interrupt flag is set (ADPCZF:CTPZFn="1").

- These bits enable/disable the ADC pulse counter interrupt.
- Each bit corresponds to a particular analog input channel.

  ADPCIE0: bit0 corresponds to AN0
  ADPCIE0: bit1 corresponds to AN1
  .
  .
  ADPCIE3: bit15 corresponds to AN31

Note:

Write "0" to the bits corresponding to the analog input channels that do not exist on the device.

8.16. ADC Pulse Counter Zero Flag Registers (ADPCZF0-3)

The ADC pulse counter zero flag is set to "1" when the positive counter (ADPCTPCTn) of the corresponding analog input channel decrements to zero.

■ ADC Pulse Counter Zero Flag Registers (ADPCZF0-3)

ADPCZF x/8								
bit	x%16	(x-1)%16	(x-2)%16	(x-3)%16	(x-4)%16	(x-5)%16	(x-6)%16	(x-7)%16
	CTPZF x	CTPZF (x-1)	CTPZF (x-2)	CTPZF (x-3)	CTPZF (x-4)	CTPZF (x-5)	CTPZF (x-6)	CTPZF (x-7)
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

ADPCZF0							
bit	7	6	5	4	3	2	1	0
	CTPZF 7	CTPZF 6	CTPZF 5	CTPZF 4	CTPZF 3	CTPZF 2	CTPZF 1	CTPZF 0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Notes:

- x : max number of channels
- "n" indicates the analog input channel number (0-31).

[bit15 to bit0] CTPZFn: ADC Pulse Counter Zero Flag

Bit	Description	
	Read	Write
0	Pulse detection interrupt condition is not met (The positive counter has not decremented to zero)	Clears pulse detection interrupt request
1	Pulse detection interrupt condition is met (The positive counter has decremented to zero)	Has no effect on operation

- The pulse counter zero flag (CTPZFn) is set to "1" when the corresponding positive counter (ADPCTNRDn) decrements from "0x01" to "0x00".
- Regardless of the ADC pulse counter interrupt enable setting (ADPCIE:CTPIEn), CTPZFn bit is set to "1" when Pulse Detection Function detects the interrupt request condition.
- Pulse detection interrupt is requested when the ADC pulse counter zero flag (CTPZFn) is set to "1" with ADPCIE:CTPIEn="1".
- These bits are cleared to "0" by writing "0" to it, and the pulse counter (ADPCTNRDn/ ADPCTPRDn) is reloaded with the value stored in the pulse counter reload register (ADPCTNCTn/ ADPCTPCTn).
- Each bit corresponds to a particular analog input channel.
  ADPCZF0: bit0 corresponds to AN0
  ADPCZF0: bit1 corresponds to AN1
  .
  .
  ADPCZF3: bit15 corresponds to AN31

Notes:

- "1" is read with read-modify-write operation.
- If the set condition and the clear condition of CTPZFn bit are satisfied simultaneously, CTPZFn bit is set to "1" since the set operation has a higher priority.

8.17. Analog Input Enable Register (ADERx)

This register enables the analog input functions for the AD converter.

■ Analog input enable register (ADERx)

ADER0

bit	7	6	5	4	3	2	1	0
	ADE 7	ADE 6	ADE 5	ADE 4	ADE 3	ADE 2	ADE 1	ADE0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

ADER(x/8 - 1)

bit	7	6	5	4	3	2	1	0
	ADE(x-1)	ADE(x-2)	ADE(x-3)	ADE(x-4)	ADE(x-5)	ADE(x-6)	ADE(x-7)	ADE(x-8)
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

x = Number of available AD channels

[bit7 to bit0] ADERx: Analog input enable Bits

If an external pin is used as an analog input for the A/D converter, the corresponding bit should be set to "1".

Mode selection	
0	Port input/output mode
1	Analog input mode

CHAPTER: A/D CONVERTER TRIGGER

This chapter explains the function and operation of the ADC Trigger.

Management Code: 96F6ADCTG-E01.0

1. Overview

The ADC Trigger triggers the ADC modules depending on the input from any of the available RLTs and PPGs.

■ Outline of ADC Trigger

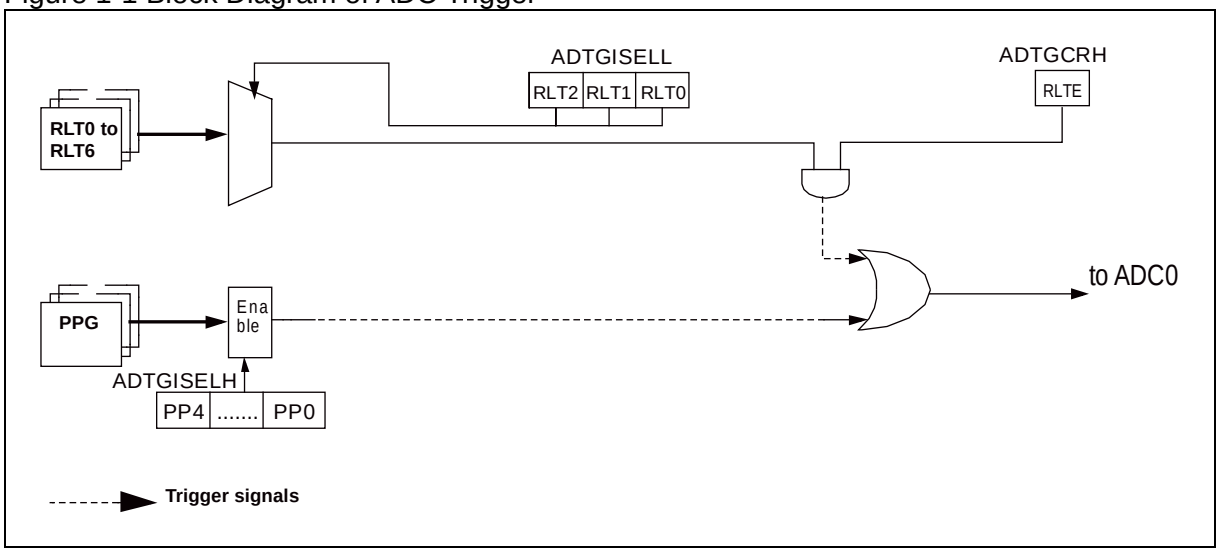
Any of the available RLT outputs and timing point events of the PPG can be chosen.
Following events can be selected as trigger:

- Timing point events from PPG
- Trigger due to the RLT timer output change

ADC module has a separate trigger module for its start trigger.

■ Block Diagram of ADC Trigger

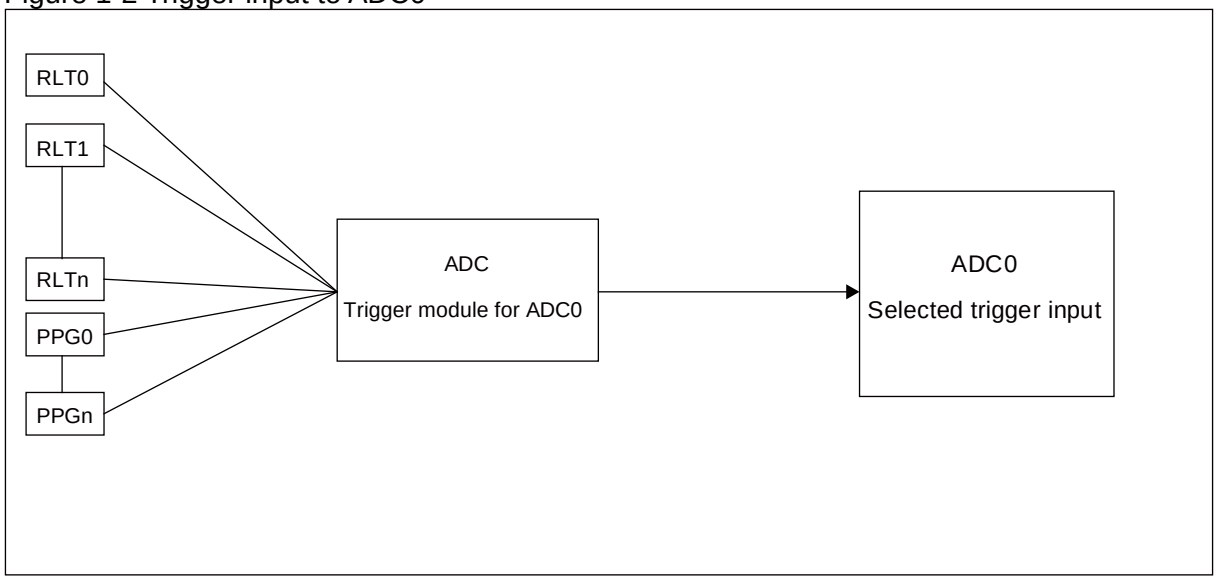
Figure 1-1 Block Diagram of ADC Trigger



■ Connection of the RLTs and the PPGs to ADC0

ADC0 module has a dedicated trigger for its triggering and any of the available RLTs and PPGs as an input to ADC trigger for ADC0 as shown in Figure 1-2.

Figure 1-2 Trigger input to ADC0



2. Operation

An ADC activation can be performed at Reload timer or PPG timing point event.

■ ADC Trigger Activation.

ADC Trigger module is used to trigger an ADC module.
Following event can be selected as trigger:

- Reload timer
- PPG timing point

3. Registers

This section explains the configuration and functions of the registers used for the ADC Trigger.

■ List of ADC Trigger Registers

Abbreviated Register Name	Register Name	Reference
ADTGCRn	ADC Trigger control register	See 3.1
ADTGISELn	ADC Trigger input select	See 3.2

3.1. ADC Trigger Control Register (ADTGCRn)

The ADC Trigger control register can be used to enable/disable the trigger output.

■ ADC Trigger Control Register Higher (ADTGCRHn)

ADTGCRHn								
bit	15	14	13	12	11	10	9	8
	-	-	-	-	-	-	-	RLTE
Attribute	-	-	-	-	-	-	-	R/W
Initial value	X	X	X	X	X	X	X	1

[bit15 to bit9] -: Undefined
- Write always "0".
- Read value is undefined.
- Read modify write operations to this register have no effect on these bits.

[bit8] RLTE

Bit	Description
0	Triggering by RLT is disabled
1	Triggering by RLT is enabled

RLTE bit enables RLT to trigger the ADC trigger.

■ ADC Trigger Control Register Lower (ADTGCRLn)

[bit7 to bit0] -: Undefined
Write always "0".
Read value is undefined.
Read modify write operations to this register have no effect on these bits.

3.2. ADC Trigger Input Select (ADTGISELn)

The ADC Trigger input select register is used to select the trigger source input from the Reload timers and PPG

■ ADC Trigger Input Select Higher (ADTGISELHn)

ADTGISELHn

bit	15	14	13	12	11	10	9	8
	-	-	-	PP4	PP3	PP2	PP1	PP0
Attribute	-	-	-	R/W	R/W	R/W	R/W	R/W
Initial value	X	X	X	0	0	0	0	0

[bit15 to bit13] -: Undefined
- Write always "0"
- Read value is undefined
- Read modify write operations to this register have no effect on these bits.

[bit12 to bit8] PP4 to PP0

bit12	bit11	bit10	bit9	bit8	Description
0	0	0	0	0	PPG triggers disabled
0	0	0	0	1	PPG group 0 triggers enabled
0	0	0	1	0	PPG group 1 triggers enabled
0	0	1	0	0	PPG group 2 triggers enabled
0	1	0	0	0	PPG group 3 triggers enabled
1	0	0	0	0	PPG group 4 triggers enabled

These bits select the PPG group that can generate ADC trigger. Initial value of the bits is "0", meaning that by default PPG modules are disabled as ADC triggers.
It is possible to select more than one PPG group as trigger source for the ADC. It must be take care that ADC conversion is finished before next trigger occurs. If ADC conversion is not yet finished when a trigger occurs, this trigger is ignored.

■ ADC Trigger Input Select lower (ADTGISELLn)

ADTGISELLn

bit	7	6	5	4	3	2	1	0
	-	-	-	-	-	RLT2	RLT1	RLT0
Attribute	-	-	-	-	-	R/W	R/W	R/W
Initial value	X	X	X	X	X	0	0	1

[bit7 to bit3] -: Undefined
- Write always "0"
- Read value is undefined
- Read modify write operations to this register have no effect on these bits.

[bit2 to bit0] RLT2 to RLT0
These bits select the input from any of the available RLTs.

bit2	bit1	bit0	Description
0	0	0	RLT 0 selected
0	0	1	RLT 1 selected
0	1	0	RLT 2 selected
0	1	1	RLT 3 selected
1	0	0	RLT 4 selected
1	0	1	RLT 5 selected
1	1	0	RLT 6 selected

CHAPTER: USART

This chapter explains the functions and operation of the LIN USART.

Management Code: 96F6USART-E01.0

1. Overview

The USART with LIN (Local Interconnect Network) - Function is a general-purpose serial data communication interface for performing synchronous or asynchronous communication with external devices. The USART provides bidirectional communication function (normal mode), master-slave communication function (multiprocessor mode in master/slave systems), and special features for LIN-bus systems (working both as master or as slave device).

■ USART Functions

The USART is a general-purpose serial data communication interface for transmitting serial data to and receiving data from another CPU and peripheral devices. It has the functions listed in Table 1-1.

Table 1-1 USART Functions

Item	Function
Data buffer	Full-duplex
Serial Input	5 times oversampling in asynchronous mode.
Transfer mode	· Clock synchronous (start-stop synchronization and start-stop-bit-option) · Clock asynchronous (using start-, stop-bits)
Baud rate	· A dedicated baud rate generator is provided, which consists of a 16-bit-reload counter · An external clock can be input and also be adjusted by the reload counter · Mechanism for Autobaud rate adjust is available in LIN mode.
Data length	· 7 bits (not in synchronous or LIN mode) · 8 bits
Signal mode	· Normal data mode. · Inverted data mode.
Start bit timing	Clock synchronization to the falling edge of the start bit in asynchronous mode (normal mode) resp. rising edge in inverted-mode
Reception error detection	· Framing error · Overrun error · Parity error (Not supported in Mode 1 and Mode 3) · Checksum error in LIN mode (Not available on all devices. Please see the datasheet of the corresponding device.) · Sync. field time out error in LIN mode · Parity error in Frame-ID in LIN mode
16 byte FIFO	· Independent 16 byte FIFO for transmission and reception · FIFO can be activated with transmit/receive trigger level for interrupt
Interrupt request	· Reception interrupt (reception data register full/FIFO level reached, reception error detect, LIN-Synch-break detect, LIN-auto header reception completed) · Transmission interrupt (transmission data register empty/FIFO level reached, transmission error detect, LIN auto header transmission completed, Last bit shifted out) · Interrupt request to ICU (LIN synch field detection) transmission and reception support for DMA function · Single interrupt at the end of header transmission and reception · FIFO trigger level interrupt · Last bit shifted out interrupt

Item	Function
Master-slave communication function (multiprocessor mode)	· One-to-n communication (one master to n slaves) (This function is supported both for master and slave system).
Synchronous mode	Function as Master- or Slave-USART
Transceiving pins	Direct access possible
LIN bus options	· Operation as master device · Operation as slave device · Generation of LIN-Sync-break · Detection of LIN-Sync-break (incl. detection during when LINSync field reception has already started) · ICUs can be used to measure the bit time of detected start/stop edges in the LIN-Sync-field See "Section 1.6 Input Capture Unit source select for LIN-USART" for the assignment of ICU to the Sync field signal of the LIN-USART. · Auto baud rate detection and adjustment feature for measuring the bit time of detected start/stop edges in the LIN-Sync-field instead of using ICU and updating the baud rate counter · Interrupt at the end of transmission/reception of complete header of LIN frame. · Checksum generation and checksum verification. · Detection of physical bus error · Internal loop back feature · Transmission /Reception FIFO of configurable depth 1 to 16 bytes
Synchronous serial clock	The synchronous serial clock can be output continuously on the SCK pin for synchronous communication with start & stop bits
Clock delay option	Special synchronous Clock Mode for delaying clock by 1/2 serial clock phase (useful for SPI)

■ USART Operation Modes

The USART operates in four different modes, which are determined by the MD0- and the MD1-bit of the Serial mode register (SMRn). Mode 0 and 2 are used for bidirectional serial communication, mode 1 for master/slave communication and mode 3 for LIN master/slave communication.

Table 1-2 USART Operation Modes

Operation mode		Data length		Synchronization of mode	Length of stop bit	data bit direction *1
		parity disabled	parity enabled			
0	normal mode	7 or 8		asynchronous	1 or 2	L/M
1	multiprocessor	7 or 8 + 1 *2	--	asynchronous	1 or 2	L/M
2	normal mode	8 or 9 (if SSM=1)		synchronous	0, 1 or 2	L/M
3	LIN mode	8	--	asynchronous	1	L

*1: means the data bit transfer format: LSB or MSB first.
*2: "+1" means the indicator bit of the address/data selection in the multiprocessor mode, instead of parity.

Note:

Mode 1 operation is supported both for master or slave operation of the USART in a master-slave connection system. In Mode 3 the USART function is locked to 8N1-Format, LSB first.

If the mode is changed, USART cuts off all possible transmission or reception and awaits then new action. The MD1 and MD0 bit of the Serial Mode Register (SMRn) determine the operation mode of the USART as shown in the following table:

Table 1-3 Mode Bit Setting

MD1	MD0	Mode	Description
0	0	0	Asynchronous (normal mode)
0	1	1	Asynchronous (multiprocessor mode)
1	0	2	Synchronous (normal mode)
1	1	3	Asynchronous (LIN mode)

2. Configuration

This section provides a short overview on the building blocks of the USART.

■ Block Diagram of USART

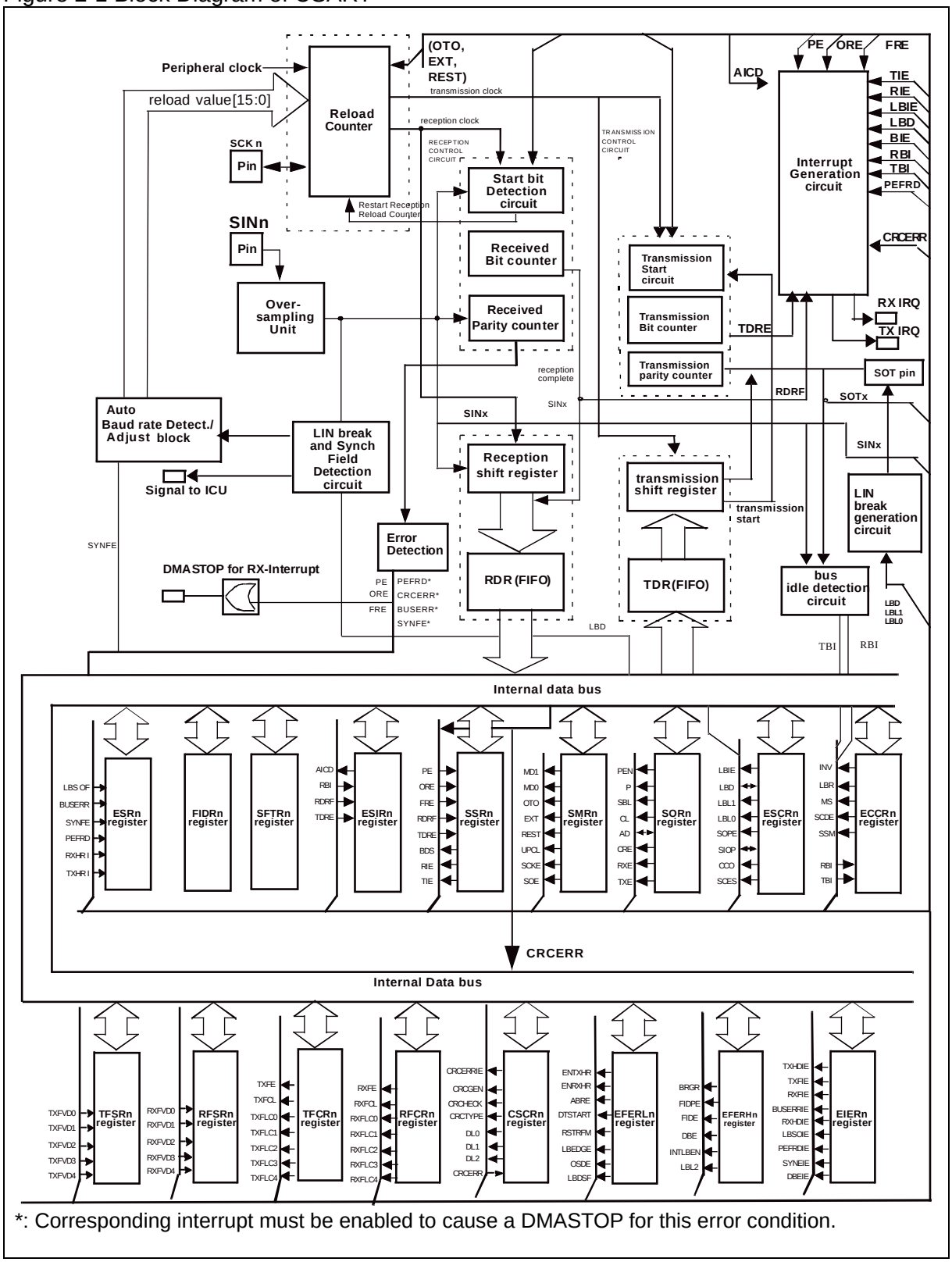
USART consists of the following blocks:

- Reload Counter
- Reception Control Circuit
- Reception Shift Register
- Reception Data Register (RDRn)
- Transmission Control Circuit
- Transmission Shift Register
- Transmission Data Register (TDRn)
- Error Detection Circuit
- Oversampling Unit
- Interrupt Generation Circuit
- LIN Synch Break/Synch Field Detection
- Bus Idle Detection Circuit
- Checksum Generation/Verification Circuit
- Automatic Baud Rate Detection and Adjustment Circuit
- FIFO Control Circuit
- Serial Mode Register (SMRn)
- Serial Control Register (SCRn)
- Serial Status Register (SSRn)
- Baud rate generation/reload register (BGRn)
- Extended Communication Control Register (ECCRn)
- Extended Status/Control Register (ESCRn)
- Extended Serial Interrupt Register (ESIRn)
- Extended Interrupt Enable Register (EIERn)
- Transmission FIFO Control Register (TFCRn)
- Reception FIFO Control Register (RFCRn)
- Transmission FIFO Status Register (TFSRn)
- Reception FIFO Status Register (RFSRn)
- Extended Feature Enable Register (EFERL/Hn)
- Extended Status Register (ESRn)
- Checksum Status and Control Register (CSCRn)
- Frame-ID Register (FIDRn)
- Sync Field Timeout Register (SFTRn)

Note:

The suffix "n" denotes the USART number.

Figure 2-1 Block Diagram of USART



*: Corresponding interrupt must be enabled to cause a DMASTOP for this error condition.

● Reload Counter

The reload counter works as the dedicated baud rate generator. It can select external input clock or internal clock for the transmitting and receiving clocks. The reload counter has a 16-bit register for the reload value. The current counter value of the transmission reload counter can be read via the BGRn.
When the auto baud rate detection/adjustment feature is enabled, the BGRn is loaded with value from the auto baud rate detection circuit.

● Reception Control Circuit

The reception control circuit consists of a received bit counter, start bit detection circuit, and received parity counter. The received bit counter counts reception data bits. When reception of one data frame for the specified data length is complete, the received bit counter sets the reception data register full flag. The start bit detection circuit detects start bits from the serial input signal and sends a signal to the reload counter to synchronize it to the falling edge of these start bits. The reception parity counter calculates the parity of the reception data.

● Reception Shift Register

The reception shift register fetches reception data input from the SINn pin, shifting the data bit by bit. When reception is complete, the reception shift register transfers receive data to the RDRn register.

● Reception Data Register (RDRn)

This register retains reception data. Serial input data is converted and stored in this register.

● Transmission Control Circuit

The transmission control circuit consists of a transmission bit counter, transmission start circuit, and transmission parity counter. The transmission bit counter counts transmission data bits. When the transmission of one data frame of the specified data length is complete, the transmission bit counter sets the Transmission data register empty flag. The transmission start circuit starts transmission when data is written to TDRn. The transmission parity counter generates a parity bit for data to be transmitted if parity is enabled.

● Transmission Shift Register

The transmission shift register loads data written to the TDRn and outputs the data to the SOTn pin, shifting the data bit by bit.

● Transmission Data Register (TDRn)

This register sets transmission data. Data written to this register is converted to serial data and is output to SOTn.

● Error Detection Circuit

The error detection circuit checks if there was any error during the last reception. If an error has occurred it sets the corresponding error flags.

● Oversampling Unit

The oversampling unit oversamples the incoming data at the SINn pin for five times. It is switched off in synchronous operation mode.

● Interrupt Generation Circuit

The interrupt generation circuit administers all cases of generating a reception or transmission interrupt. If a corresponding enable flag is set and an interrupt case occurs the interrupt will be generated immediately.

● LIN Synch Break and Synchronization Field Detection Circuit

The LIN break and LIN synchronization field detection circuit detects a LIN break, if a LIN master node is sending a message header. If a LIN break is detected a special flag bit is generated. The first and the fifth falling edge of the synchronization field is recognized by this circuit by generating an internal signal for the Input Capture Unit or Auto Baud rate detection/adjustment circuit to measure the actual serial clock time of the transmitting master node.

● LIN Synch Break Generation Circuit

The LIN break generation circuit generates a LIN break of a determined length.

● Bus Idle Detection Circuit

The bus idle detection circuit recognizes if neither reception nor transmission is going on. In this case, the circuit generates the special flag bits TBI and RBI.

● Checksum Generation/Verification Circuit

The checksum generation/verification circuit generates the checksum bytes during transmission of data bytes only in mode 3 (LIN mode). Similarly it verifies the calculated checksum against the received one during reception.

● Auto Baud Rate Detection and Adjustment Circuit

The auto baud rate detection and adjustment circuit programs the BGRn register by calculating the bit time of the Sync field for synchronization to the LIN master.

● Transmission / Reception FIFO

The Transmission / Reception FIFO are of 16 bytes each. It is used for storing the data during transmission and reception.

● LIN-USART Serial Mode Register (SMRn)

This register performs the following operations:

- Selecting the LIN-USART operation mode
- Selecting a clock input source
- Selecting if an external clock is connected "one-to-one" or connected to the reload counter
- Resetting dedicated reload timer
- Resetting the LIN-USART (preserving the settings of the registers)
- Specifying whether to enable serial data output to the corresponding pin
- Specifying whether to enable clock output to the corresponding pin

● Serial Control Register (SCRn)

This register performs the following operations:

- Specifying whether to provide parity bits
- Selecting parity bits
- Specifying a stop bit length
- Specifying a data length
- Selecting a frame data format in mode 1
- Clearing the error flags
- Specifying whether to enable transmission
- Specifying whether to enable reception

● Serial Status Register (SSRn)

This register performs the following functions:

- Indicating status of receive/transmit operations and errors
- Specifying LSB first or MSB first
- Receive interrupt enable/disable
- Transmit interrupt enable/disable

● Extended Status/Control Register (ESCRn)

This register performs the following functions:

- LIN synch break interrupt enable/disable
- Indicating LIN synch break detection
- Specifying LSB bits of LIN synch break length
- Directly accessing SINn and SOTn pins
- Specifying continuous clock output operation
- Specifying sampling clock edge

● Extended Communication Control Register (ECCRn)

This register performs the following functions:

- Indicating bus idle state
- Specifying synchronous clock
- Specifying LIN synch break generation

● Extended Serial Interrupt Register (ESIRn)

This register performs the following function:

- Change handling of interrupts to enable usage together with DMA

● Extended Interrupt Enable Register (EIERn)

This register performs the following function:

- Specifies interrupt enables for various interrupt sources

● Transmission FIFO Control Register (TFCRn)

This register performs the following function

- Enables transmission FIFO
- Specifies trigger levels for interrupt during transmission

● Reception FIFO Control Register (RFCRn)

This register performs the following function

- Enables reception FIFO
- Specifies trigger levels for interrupt during reception

● Transmission FIFO Status Register (TFSRn)

This register performs the following function

- Indicates the number of valid data bytes in transmission FIFO

● Reception FIFO Status Register (RFSRn)

This register performs the following function

- Indicates the number of valid data bytes in reception FIFO

● Extended Feature Enable Register (EFERn)

This 16-bit register performs the following function

- Enables interrupt at the end of transmission and reception of header in LIN mode
- Enables auto baud rate detection and adjustment feature in LIN mode
- Enables edge sensitive detection of LIN break when operating in LIN mode
- Disables resetting of reception state machine when errors are cleared by CRE
- Disables detection of low level of SINn after framing error as a valid start signal
- Enables detection of bus error
- Enables internal loop back test
- Enables frame-ID register, when operating in LIN mode
- Specifies MSB of LIN break length, when operating in LIN mode
- Enables the LIN break detect in sync field function

● Checksum Status and Control Register (CSCRn)

This register performs the following function in LIN mode.

- Enables checksum generation at transmission side and checksum verification at the reception side
- Selects between classic and enhanced checksum according to LIN Specification 2.1
- Indicates the status of checksum during reception
- Specifies the number of data bytes in a LIN frame
- Specifies the interrupt enable for checksum reception error interrupt

● Extended Status Register (ESRn)

This register performs the following function

- Indicates the status of bus error detection
- Indicates the status of reception of frame-ID in LIN mode
- Indicates the status of sync field detection in LIN mode

● Frame ID Data Register (FIDRn)

This register performs the following function

- Stores the value of frame-ID when USART is acting as a LIN mode

● Sync Field Timeout Register (SFTRn)

This register performs the following function in LIN mode

- Contains the 16 bit timeout value considered for sync field detection

3. Pins

This section describes the USART pins and provides a pin block diagram.

■ USART Pins

The USART pins are shared with general purpose ports. Table 3-1 lists the pin functions, I/O formats, and settings required to use the USART.

Table 3-1 USART Pins

Pin name	Pin function	I/O format	Pull-up	Standby control	Setting required to use pin
Pxx_i/SINn	Port I/O or serial data input	Please see "CHAPTER 11 I/O PORTS" for how to select available formats	Programmable pull-up resistor. (see "CHAPTER 11 I/O PORTS" how to activate the pull-up resistor)	Provided	Set corresponding general purpose port to "input": DDRxx:Di = 0 PIERxx:IEi = 1
Pyy_j/SOTn	Port I/O or serial data output				Set output enable mode: (SMRn: SOE = 1)
Pzz_k/SCKn	Port I/O or serial clock input/output				When a clock is input, then set corresponding general purpose port to "input": DDRzz:Dk = 0 PIERzz:IEk = 1
					When a clock is output, then set output enable mode: SMRn:SCKE = 1

4. Operation

USART operates in operation mode 0 for normal bidirectional serial communication, in mode 2 and 3 in bidirectional communication as master or slave, and in mode 1 as master or slave in multiprocessor communication.

■ Operation of USART

● Operation Modes

There are four USART operation modes: modes 0 to 3. As listed in Table 4-1, an operation mode can be selected according to the communication method.

Table 4-1 USART Operation Mode

Operation mode		Data length		Synchronization of mode	Length of stop bit	data bit direction *1
		parity disabled	parity enabled			
0	normal mode	7 or 8		asynchronous	1 or 2	L/M
1	multiprocessor	7 or 8 + 1*2	-	asynchronous	1 or 2	L/M
2	normal mode	8 or 9 (if SSM=1)		synchronous	0, 1 or 2	L/M
3	LIN mode	8	-	asynchronous	1	L

*1: means the data bit transfer format: LSB or MSB first
*2: "+1" means the indicator bit of the address/data selection in the multiprocessor mode, instead of parity.

Note:

Mode 1 operation is supported both for master or slave operation of USART in a master-slave connection system. In Mode 3 the USART function is locked to 8N1-Format, LSB first.
If the mode is changed, USART cuts off all possible transmission or reception and awaits then new action.

■ Inter-CPU Connection Method

External Clock One-to-one connection (normal mode) and master-slave connection (multiprocessor mode) can be selected. For either connection method, the data length, whether to enable parity, and the synchronization method must be common to all CPUs. Select an operation mode as follows:

- In the one-to-one connection method, operation mode 0 or 2 must be used in the two CPUs. Select operation mode 0 for asynchronous transfer mode and operation mode 2 for synchronous transfer mode. Note, that one CPU has to be configured as master and the other has to be configured as slave in synchronous mode 2.
- Select operation mode 1 for the master-slave connection method and use it either for the master or slave system.

■ Synchronization Methods

In asynchronous operation USART reception clock is automatically synchronized to the falling edge of a received start bit.
In synchronous mode the synchronization is performed either by the clock signal of the master device or by USART itself if operating as master.

■ Signal Mode

USART can treat data in normal format or inverted format.

■ Operation Enable Bit

USART controls both transmission and reception using the operation enable bit for transmission (SCRn: TXE) and reception (SCRn: RXE).

- If reception operation is disabled during reception (data is input to the reception shift register), finish frame reception and read the received data of the reception data register (RDRn). Then stop the reception operation.
- If the transmission operation is disabled during transmission (data is output from the transmission shift register), wait until there is no data in the transmission data register (TDRn) before stopping the transmission operation.

4.1. Operation in Asynchronous Mode (Op. Modes 0 and 1)

When USART is used in operation mode 0 (normal mode) or operation mode 1 (multiprocessor mode), the asynchronous transfer mode is selected.

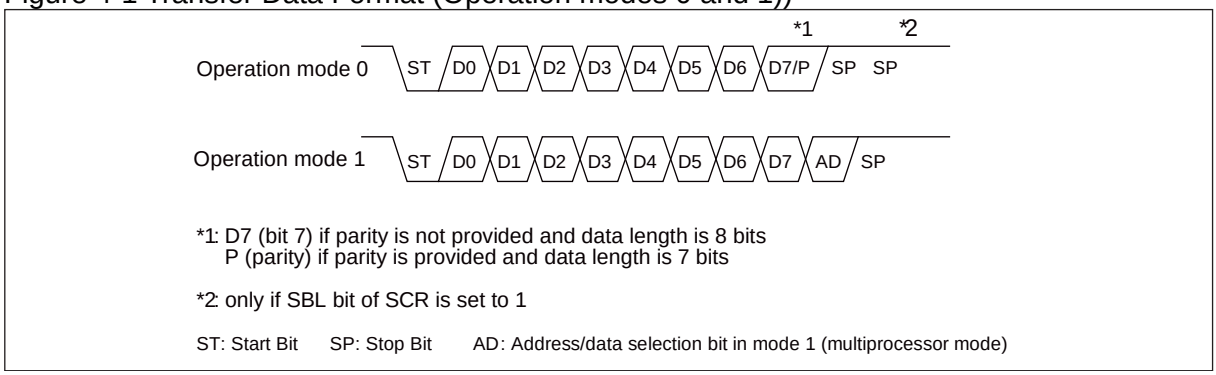
■ Operation in Asynchronous Mode

● Transfer Data Format

Generally each data transfer in the asynchronous mode operation begins with the start bit (low-level on bus) and ends with at least one stop bit (high-level). The direction of the bit stream (LSB first or MSB first) is determined by the BDS bit of the Serial Status Register (SSRn). The parity bit (if enabled) is always placed between the last data bit and the (first) stop bit.
In operation mode 0 the length of the data frame can be 7 or 8 bits, with or without parity, and 1 or 2 stop bits.
In operation mode 1 the length of the data frame can be 7 or 8 bits with a following address-/data-selection bit instead of a parity bit. 1 or 2 stop bits can be selected.
The calculation formula for the bit length of a transfer frame is:

Length = 1 + d + p + s
(d = number of data bits [7 or 8], p = parity [0 or 1], s = number of stop bits [1 or 2]

Figure 4-1 Transfer Data Format (Operation modes 0 and 1))



Note:

If BDS bit of the Serial Status Register (SSRn) is set to "1" (MSB first), the bit stream processes as: D7, D6, ..., D1, D0, (P).

During Reception both stop bits are detected, if selected. But the Reception data register full (RDRF) flag will go "1" at the first stop bit. The bus idle flag (ECCRn:RBI) goes "1" after the second stop bit if no further start bit is detected. (The second stop bit belongs to "bus activity", although it is just mark level.)

● Transmission Operation

If the Transmission Data Register Empty (TDRE) flag bit of the Serial Status Register (SSRn) is "1", transmission data is allowed to be written to the Transmission Data Register (TDRn). When data is written, the TDRE flag goes "0". If the transmission operation is enabled by the TXE-Bit ("1") of the Serial Control Register (SCRn), the data is written next to the transmission shift register and the transmission starts at the next clock cycle of the serial clock, beginning with the start bit. Thereby the TDRE flag goes "1", so that new data can be written to the TDRn.
If transmission interrupt is enabled (TIE = 1), the interrupt is generated by the TDRE flag. Note, that the initial value of the TDRE flag is "1", so that in this case if TIE is set to "1" an interrupt will occur immediately.
When the character length is set to 7 bits (CL = 0), the unused bit of the TDRn is always the MSB, independently from the bit direction setting in the BDS bit (LSB first or MSB first).

● Reception Operation

Reception operation is performed when it is enabled by the Reception Enable (RXE) flag bit of the SCRn. If a start bit is detected, a data frame is received according to the format specified by the SCRn. In case of errors, the corresponding error flags are set (PE, ORE, FRE). After the reception of the data frame the data is transferred from the serial shift register to the Reception Data Register (RDRn) and the Receive Data Register Full (RDRF) flag bits of SSRn and ESIRn registers are set.
If receive interrupt is enabled (RIE = 1) and ESIRn:AICD = 0, the interrupt is generated by SSRn:RDRF.
If receive interrupt is enabled (RIE = 1) and ESIRn:AICD = 1, the interrupt is generated by ESIRn:RDRF.
The data then has to be read by the CPU. By doing so, the SSRn:RDRF flag is cleared.
When ESIRn:AICD = 0, this also clears the interrupt.
When ESIRn:AICD = 1, the interrupt must be cleared by writing "1" to ESIRn:RDRF.
When the character length is set to 7 bits (CL = 0), the unused bit of the RDRn is always the MSB, independently from the bit direction setting in the BDS bit (LSB first or MSB first).

Note:

Only when the RDRF flag bit is set and no errors have occurred the Reception Data Register (RDRn) contains valid data.

● Used Clock

Use the internal clock or external clock. Select the baudrate generator (SMRn:EXT = 0 or 1, SMRn:OTO = 0) for desired baudrate.

● Stop Bit, Error Detection, and Parity:

Number of stop bit, 1 or 2 can be specified by the SBL bit of the SCRn register. When receiving and 2-bit is set to the stop bit, the second stop bit is checked in addition to the first stop bit. The RBI (bus idle) flag is set after the second stop bit. However the RDRF flag is set when the first stop bit is received. In mode 0, parity error, overrun error and framing error are checked. In mode 1, parity check is not supported and overrun error and framing error are checked. The PEN bit of the SCRn register enables/disables the parity bit and the P bit specifies even or odd parity in mode 0.

4.2. Operation in Synchronous Mode (Operation Mode 2)

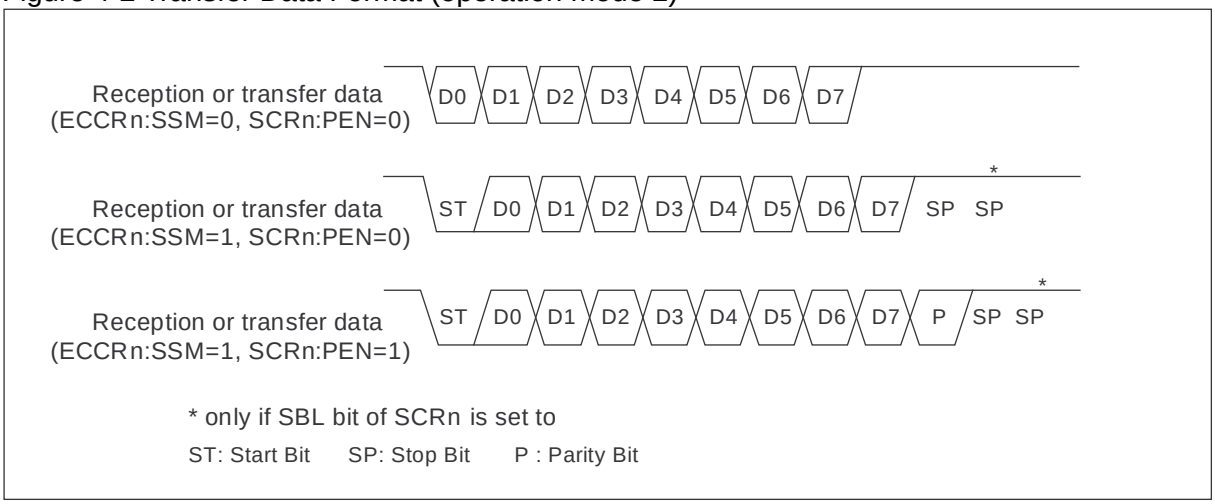
The clock synchronous transfer method is used for USART operation mode 2 (normal mode).

■ Operation in Synchronous Mode (Operation mode 2)

● Transfer Data Format

In the synchronous mode, 8-bit data is transferred without start or stop bits if the SSM bit of the Extended Communication Control Register (ECCRn) is 0. The figure below illustrates the data format during a transmission in the synchronous operation mode.

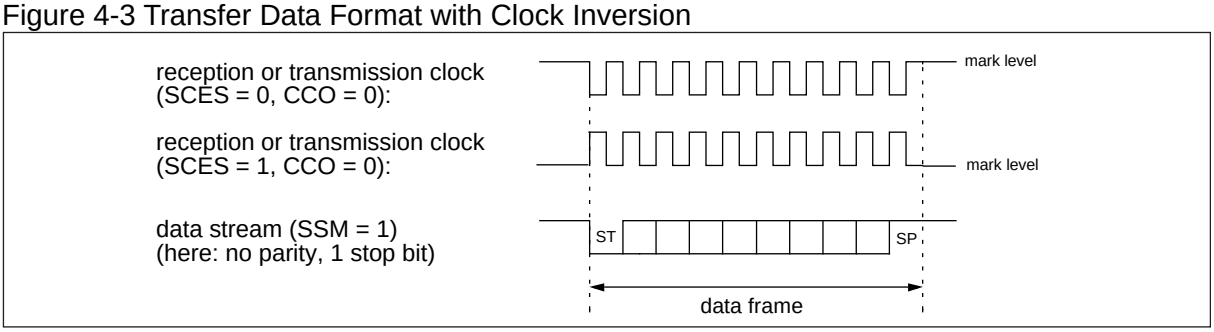
Figure 4-2 Transfer Data Format (operation mode 2)



● Clock Inversion and Start/Stop Bits in Mode 2

If the SCES bit of the Extended Status/Control Register (ESCRn) is set the serial clock is inverted. Therefore in slave mode USART samples the data bits at the falling edge of the received serial clock. Note, that in master mode if SCES is set the clock signal's mark level is "0". If the SSM bit of the Extended Communication Control Register (ECCRn) is set the data format gets additional start and stop bits like in asynchronous mode.

Figure 4-3 Transfer Data Format with Clock Inversion

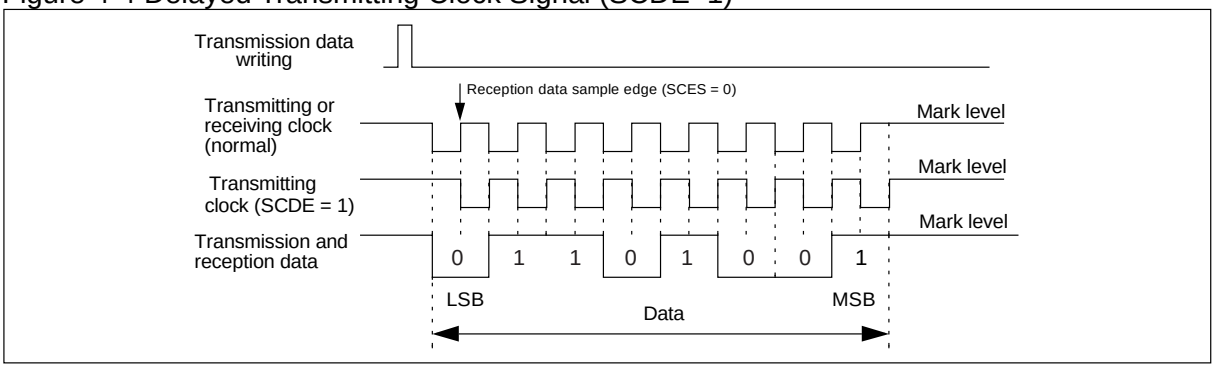


● Clock Supply:

In operation mode 2, the number of clock cycles for the clock signal must be the same as the number of bits for the data including start and stop bits. If the MS bit of the ECCRn register is "0" (master mode) and the SCKE bit of the SMRn register is "1" (clock output enabled), the consistent clock cycles are generated automatically. If the MS bit of the ECCRn register is "1" (slave mode), make sure that correct clock cycles are generated by the other communication device. While there is no communication, the clock signal must be kept at "1" as the mark level.

If the SCDE bit of the ECCRn register is "1", the clock output signal is delayed by the half of the serial clock cycle as shown in Figure 4-4. The operation is prepared for communication devices which use the falling edge of the serial clock signal for the data sampling.

Figure 4-4 Delayed Transmitting Clock Signal (SCDE=1)



If the SCES bit of the ESCRn register is "1", the serial clock signal is inverted. Receiving data is sampled at the falling edge of the serial clock.

If the MS bit of the ECCRn register is "0" (master mode) and the SCKE bit of the SMRn register is "1" (clock output enabled), the output clock signal is also inverted.

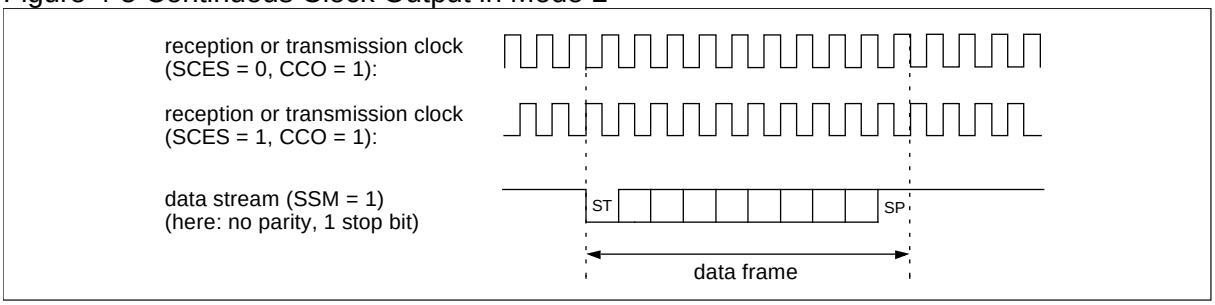
While there is no communication, the clock signal must be kept at "0" as the mark level.

If the CCO bit of the ESCRn register is "1", the serial clock is signaled even while there is no data communication. Therefore it is recommended to specify the start/stop bits as shown in Figure 4-5.

Table 4-2 Serial Data Input Sampling Depending on ECCRn:SCDE and ESCRn:SCES

S.No	ECCRn: SCDE	ESCRn: SCES	Serial data sampling edge	Serial data transmitting edge
1	0	0	Rising edge	Falling edge
2	0	1	Falling edge	Rising edge
3	1	0	Falling edge	Rising edge
4	1	1	Rising edge	Falling edge

Figure 4-5 Continuous Clock Output in Mode 2



● Error Detection:

If no start/stop bits is selected (ECCRn: SSM = 0) only overrun errors are detected.

● Communication:

For initialization of the synchronous mode, following settings have to be done:

Baud rate generator registers (BGRn):

Set the desired reload value for the dedicated baud rate reload counter.

Serial mode control register (SMRn):

MD1, MD0: "10_B" (Mode 2)
SCKE: "1" for the dedicated Baud Rate Reload Counter
"0" for external clock input
SOE: "1" for transmission and reception
"0" for reception only

Serial control register (SCRn):

RXE, TXE: set both of these flags to "0"
AD: no Address/Data selection - don’t care
CL: automatically fixed to 8-bit data - don’t care
CRE: "1" to clear receive error flags.

-- when SSM=0 (default):

PEN, P, SBL: don’t care

-- when SSM=1:

PEN: "1" if parity bit is added/detected, "0" if not
P: "0" for even parity, "1" odd parity
SBL: "1" for 2 stop bits, "0" for 1 stop bit.

Serial status register (SSRn):

BDS: "0" for LSB first, "1" for MSB first
RIE: "1" if interrupts are used; "0" receive interrupts are disabled.
TIE: "1" if interrupts are used; "0" transmission interrupts are disabled.

Extended communication control register (ECCRn):

SSM: "0" if no start/stop bits are desired (normal); "1" for adding start/stop bits (special)
MS: "0" for master mode (USART generates the serial clock); "1" for slave mode (USART receives serial clock from the master device)

Serial control register (SCRn):

RXE, TXE: set one or both of these control bits to "1" to begin communication.

4.3. Features of USART in LIN Mode

USART in LIN mode supports several additional features which reduces the interrupt load on the CPU.

■ Features of USART in LIN Mode

- Automatic Header Transmission
- Automatic Header Detection
- Automatic Baud Rate Detection/Adjustment
- LIN-Checksum Handling with Respect to Message Length
- Detection of Bus Error
- Internal Loop Back Feature
- Transmission/Reception FIFO of Configurable Depth of 16 Bytes Each
- Variable LIN Break Length Generation
- LIN Break Detect in Sync Field.

● Interrupt at End of Transmission of Complete Header as LIN Master

This feature is enabled by setting the bit (EFERn:ENTXHR to "1" and EIERn:TXHDIE to "1") and started by writing "1" to ESCRn:LBR. LIN will complete the transmission of header and interrupts the CPU at the end of transmission. Enabling transmission, asserting LIN break request and writing Frame-ID in Frame-ID data register (if enabled by EFERHn:FIDE = 1) or in TDRn or TX FIFO (if enabled) will result in following actions:

1. Generation of LIN break as per the length specified in Extended feature enable register (EFERHn:LBL2) and Extended status and control register (ESCRn:LBL1 and ESCRn:LBL0 bits).
2. Send Sync character 0x55.
3. Send Frame-ID programmed in Frame-ID register, TX-FIFO or TDRn.
4. A transmission interrupt (ESRn:TXHRI = 1) is generated at the end of the Frame-ID transmission (when stop bit is being sent).

● Interrupt at the End of Reception of Complete Header as LIN Slave

This feature is enabled by setting the bit (EFERLn:ENRXHR to "1" and EIERn:RXHDIE to "1") . LIN slave will complete the reception of the header and assert an reception interrupt at the end of the header reception. Enabling auto baud rate feature and the interrupt generation for reception will result in following things:

1. Detection of LIN break.
2. Reception of Sync field and getting adjusted to LIN network with the help of auto baud rate detection/ adjustment circuitry.
3. Reception of Frame-ID into FIDRn.
4. A reception interrupt (ESRn:RXHRI = 1) is generated at the end of reception of Frame-ID. Possible reception errors are indicated in the corresponding status flags.

Here the auto baud rate detection/adjustment block is used instead of ICU for calculating the one bit time of sync field received. By enabling (EFERLn:ENRXHR and EIERn:RXHDIE), following interrupts are not required when header reception is ongoing.

1. LIN break detect interrupt.
2. Two ICU interrupt for calculation of one bit time of sync field.

● Automatic Baud Rate Adjustment

This feature is enabled by setting EFERLn:ABRE = 1. In the auto baud rate detection/adjustment circuitry the time elapsed between first and fifth falling edge of Sync field is calculated without utilizing ICU. The calculated value is divided by 8 for determining bit time of sync field**.** The bit time of sync field is rounded-off by incrementing the value if the remainder is equal to or greater than 0.5 times of divider value 8**.** The 16-bit reload value of the baud rate counter is loaded into reload counter for generation of reception clock and transmission clock.

Note:

When this feature is enabled, ICU interrupts are not required for calculation of BGRn value but can be used in addition.

● LIN-Checksum Handling in Regard to Message Length

LIN-Checksum generation

This feature is enabled by setting the bit CSCRn:CRCGEN to "1". For LIN-checksum generation (Master sends data to Slave) the number of data bytes in the LIN frame is programmed as the data length in the checksum status/control register (CSCRn:D[2:0]). By enabling this feature, the checksum byte is generated and transmitted after the data bytes. Both classic and enhanced types of checksum types according to LIN specification 2.1 are supported. In case of classic checksum, only data bytes are considered for calculation of checksum value. In enhanced check sum both data byte and Frame-ID are considered for calculation of checksum value.

The selection of classic or enhanced checksum is done with the help of programmable bit CRCTYPE in check sum status/control register. The checksum contains the inverted eight bit sum with carry over all data bytes (classic checksum) or all data bytes and the protected identifier (enhanced checksum).

● LIN-Checksum Verification

This feature is enabled by setting the bit CSRCn:CRCCHECK to "1". For LIN-checksum verification the number of data bytes in the LIN frame is programmed as the data length in the checksum status/control register (CSCRn:DL[2:0]). By enabling this feature, the check sum byte received at the end of reception of data bytes (configured by CSCRn:DL[2:0]) is added with internally calculated checksum and checked whether it is equal to 0xFF. If the calculated sum is not equal to 0xFF, the CSCRn:CRCERR flag is set. This will result in a reception interrupt when EIERn:CRCERRIE is set to "1".

LIN-Checksum generation and verification are done according to LIN specification 2.1.

Notes:

- When master sends data to slave this verification can be used for self checking.
- Check sum verification is not performed if a parity error in the corresponding Frame-ID has bee received (ESRn:PEFRD="1"). To enable check sum verification, clear ESRn:PEFRD before receiving the frame data.

● Detection of Bus Error

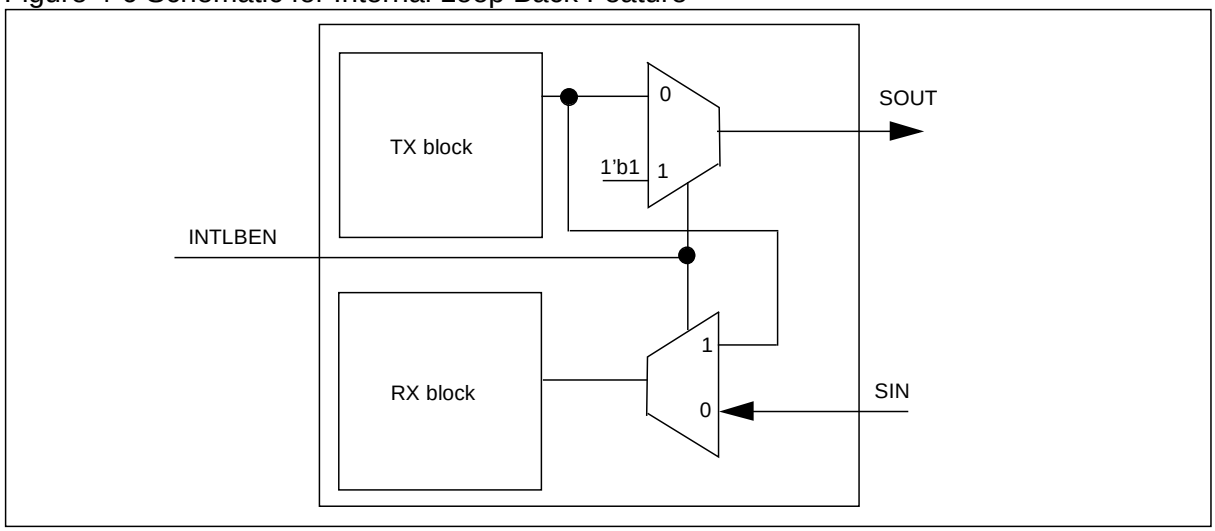
This feature is enabled by setting the bit EFERLn:DBE to "1". The detection of bus error is done by enabling both transmission and reception, so that LIN node can read back its own transmission (as the LIN is single wire network). The physical bus error such as shorted to ground or Vcc can be found by comparing the value of transmitted and received data. If there is a difference between the transmitted and received value then ESRn:BUSERR flag is set. This will result in a reception interrupt if EIERn:BUSERRIE is set to "1".

● Internal Loop Back Feature

This feature is enabled by setting the bit EFERHn:INTLBEN to "1".

By enabling this feature, the transmitter output is internally fed back to the receiver input so that the software can send and receive back the same data for the purpose of comparison. This reduces the tester time/debug efforts for testing LIN hardware.

Figure 4-6 Schematic for Internal Loop Back Feature



● FIFO Block for Transmission and Reception

This feature is enabled by setting the bits EFERn:TXFE, EFERn:RXFE to "1".Transmission and reception FIFO of configurable length 16 bytes each is available for storing the data bytes. The trigger level for TX/RX FIFO interrupts are set by programming the bits TFCRn:TXFLC[4:0], RFCRn:RXFLC[4:0] respectively to the required value in the range from 1 to 16. The number of valid data bytes in the Transmission and reception FIFO are indicated by the following register bits TFSRn:TXFVD[4:0] and RFSRn:RXFVD[4:0] respectively.

● Variable Length LIN Break Generation

LIN break of variable length from 13-20 bits can be generated. This is possible by setting the bits (EFERHn.LBL2, ESCRn:LBL[1:0]) to the required value.

4.4. Operation with LIN Function (Operation Mode 3)

USART can be used either as LIN-Master or LIN-Slave. For this LIN function a special mode is provided. Setting the USART to mode 3 configures the data format to 8N1-LSB-first format.

■ Operation in Synchronous LIN Mode (Operation mode 3)

● USART as LIN Master

In LIN master mode the master determines the baud rate of the whole sub bus, therefore slave devices have to synchronize to the master. The desired baud rate remains fixed in master operation after initialization.

If the automatic header transmission is disabled (EFERLn:ENTXHR = "0"):
Writing a "1" into the LBR bit of the Extended Communication Control Register (ECCRn) generates a 13 - 20 bit time low-level on the SOT pin, which is the LIN synchronization break and the start of a LIN message. Thereby the TDRE flag of the Serial Status Register (SSRn) goes "0" and is reset to "1" after the break, and generates a transmission interrupt for the CPU (if TIE of SSRn is "1"). The length of the Synchronization break to be sent can be determined by the LBL2 bit of EFERn and LBL1/0 bits of the ESCRn as follows:

Table 4-3 LIN break Length

LBL2	LBL1	LBL0	Length of Break
0	0	0	13 Bit times
0	0	1	14 Bit times
0	1	0	15 Bit times
0	1	1	16 Bit times
1	0	0	17 Bit times
1	0	1	18 Bit times
1	1	0	19 Bit times
1	1	1	20 Bit times

The Sync Field is sent as byte data of 0x55 after the LIN break. To prevent a transmission interrupt, the 0x55 can be written to the TDRn just after writing the "1" to the LBR bit, although the TDRE flag is "0". The internal transmission shifter waits until the LIN break has finished and shifts the TDRn value out afterwards. In this case no interrupt is generated after the LIN break and before the start bit of the sync field (0x55).

If automatic header transmission is enabled (EFERLn:ENTXHR = 1):
When ECCRn:LBR is set to "1", LIN break of length programmed is transmitted. After the transmission of LIN break is finished, Sync field value (0x55) is written internally by the USART to the transmission shift register. After the transmission of sync field USART transmits the Frame-ID in the Frame-ID data register (when EFERHn:FIDE = 1) or TX FIFO (when TFCRn:TXFE = 1) or TDRn. When there is a parity error in the received Frame-ID, the ESRn:PEFRD flag is set. This will result in an reception interrupt when EIERn:PEFRDIE is set to "1". A header transmission interrupt is asserted, after Frame-ID is transferred to the shift register (if TIE = 1 and LBSOIE = 0) or after Frame-ID is shifted out (TIE = 0 and LBSOIE = 1).

● USART as LIN Slave

In LIN slave mode USART has to synchronize to the master’s baud rate. If Reception is disabled (RXE = 0) but LIN break Interrupt is enabled (LBIE = 1) USART will generate a reception interrupt, if a synchronization break from the LIN master is detected, and indicates it with the LBD flag of the ESCRn. Writing "0" to this bit clears the receive interrupt request. The LIN slave may need to calculate the baud rate from the synch field. In this case, the time between the first falling edge to the fifth falling edge of the synch field is measured by the input capture module. For this purpose, the input capture module is connected to the LIN-USART with an internal signal. This internal signal changes from "0" to "1" at the first falling edge then "1" to "0" at the fifth falling edge. Therefore the input capture module should be set to detect both rising and falling edge. Also the input signal from the LIN-USART should be selected. The time measured by the input capture module represents 8 times of the baud rate clock cycle.

Therefore, baud rate setting value is summarized as follows:
without timer overflow: BGRn value = ((b - a) / 8) - 1
with timer overflow : BGRn value = (max + b - a) / 8

where max is the timer maximum value at the overflow occurs.
where a is the value of the ICU counter register after the first Interrupt
where b is the value of the ICU counter register after the second Interrupt

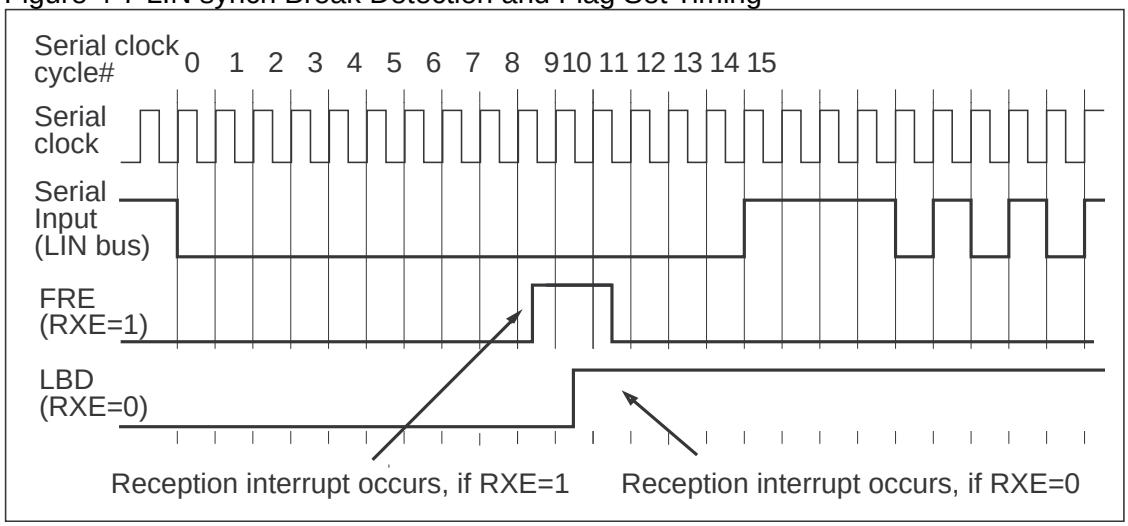
For the correspondence between other USARTs and ICUs, see "Section 12.3 16-bit Free-Running Timer"

When automatic header detection is enabled by setting EFERLn:ENRXHR to "1", LIN break, Sync field and Frame-ID are detected by USART automatically. If only EIERn:RXHDIE is enabled only one reception interrupt is set after receiving Frame-ID.
If EFERLn:ABRE = 0, baud rate is not automatically adjusted to the master baud rate. In this case baud rate detection can be done by ICU as described above.
If EFERLn:ABRE = 1, baud rate is adjusted automatically by the auto baud rate detection/adjustment circuitry after LIN break gets detected.
Detection of sync field is independent of EFERLn:ABRE.
If the sync field is not detected within the timeout value programmed in the sync field timeout register (SFTRn), ESRn:SYNFE flag is set. This will result in a reception interrupt when EIERn:SYNFEIE is set to "1". Else if the sync field gets detected within the timeout value, the calculated bit time of sync field is loaded to the reload counter for getting synchronized to LIN master.
Detection of sync field time out is independent of EFERLn:ABRE.

● LIN Synch Break Detection Interrupt and Flags

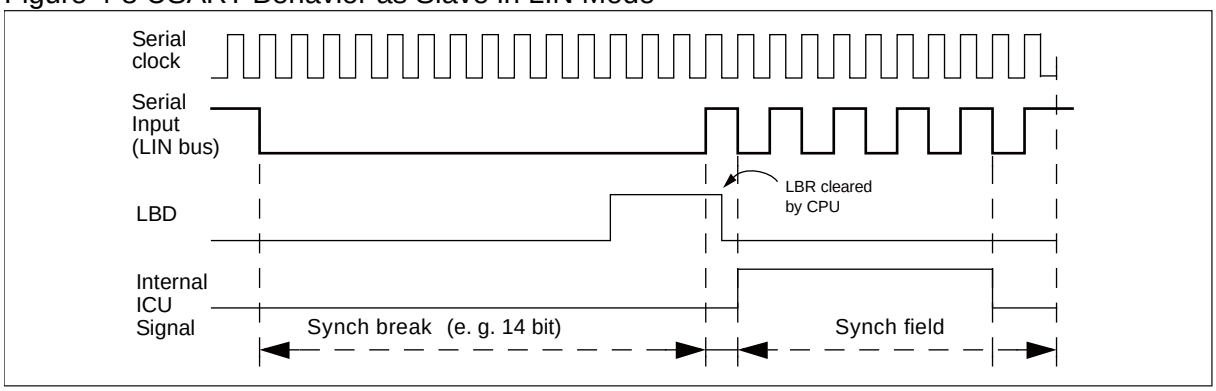
If a LIN Synch synchronization break is detected in the slave mode, the LIN Break Detected (LBD) flag of the ESCRn is set to "1". This causes an interrupt, if the LIN Break Interrupt Enable (LBIE) bit is set.

Figure 4-7 LIN synch Break Detection and Flag Set Timing



The figure above demonstrates the LIN synch break detection and flag set timing.
Note, that if reception is enabled (RXE = 1) and reception interrupt is enabled (RIE = 1) the Reception Data Framing Error flag (FRE) bit of the SSRn will cause a reception interrupt 2 bit times ("8N1") earlier than the LIN break interrupt, so it is recommended to turn off RXE to avoid that the FRE-Flag is set, if a LIN break is expected (if automatic header reception is not used). LIN sync break can be detected even if RXE is disabled.
LBD is only supported in operation mode 3.
The Figure 4-8 shows a typical start of a LIN message frame and the behavior of the USART.

Figure 4-8 USART Behavior as Slave in LIN Mode



● LIN bus timing

Figure 4-9LIN Bus Timing and USART Signals w/o Automatic Header Detection

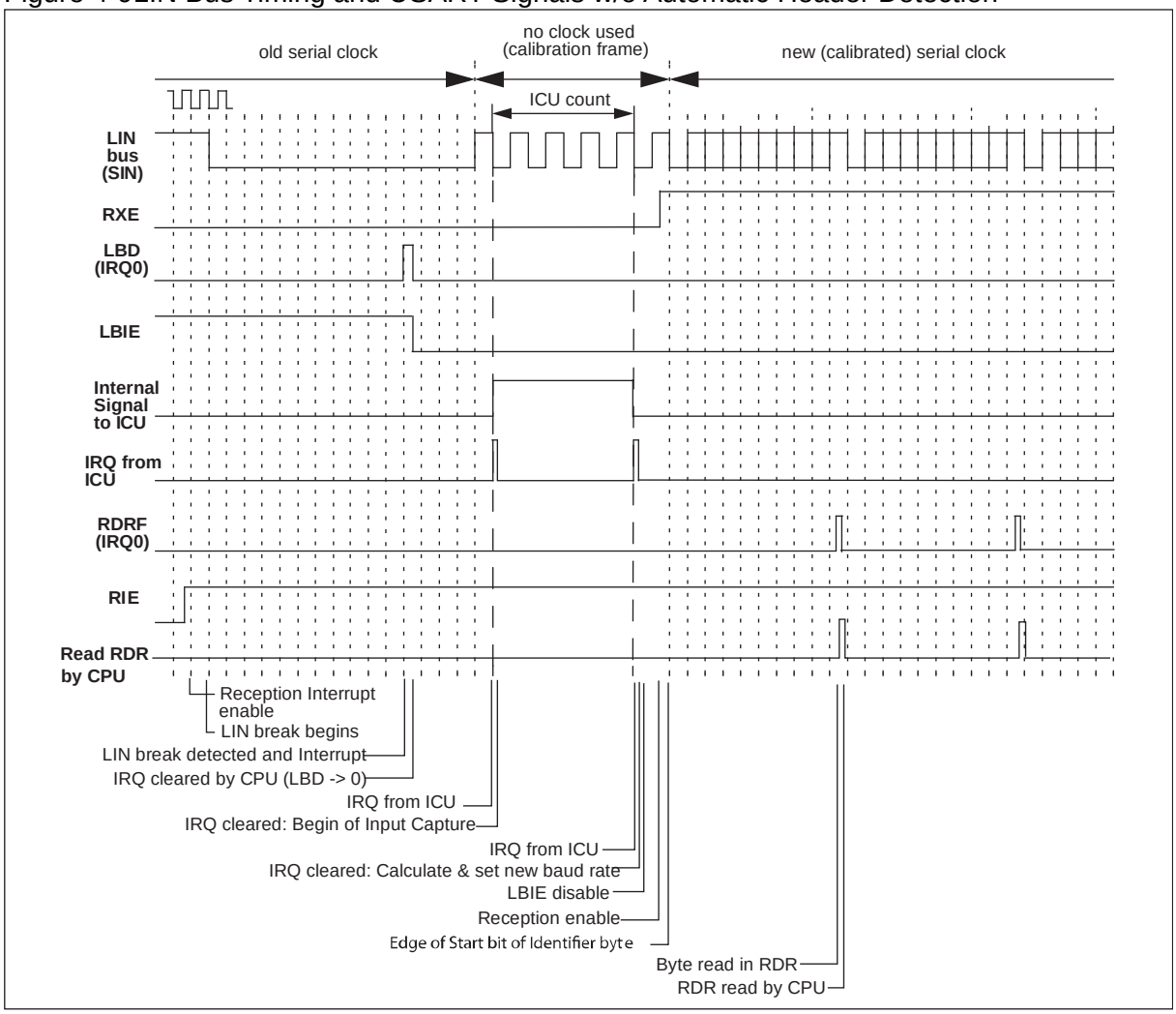
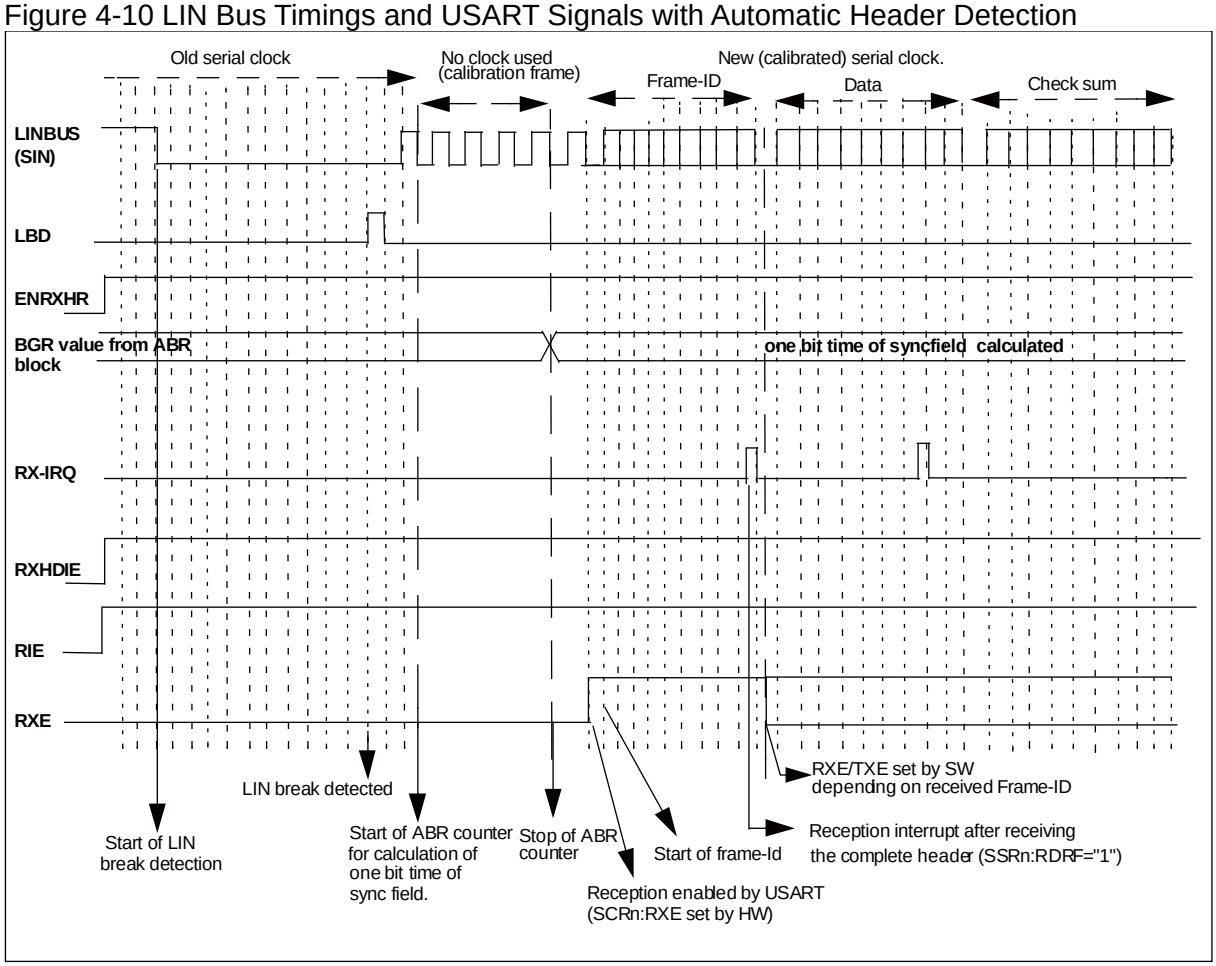


Figure 4-10 LIN Bus Timings and USART Signals with Automatic Header Detection



4.5. Direct Access to Serial Pins

USART allows the user to directly access to the transmission pin (SOTn) or the reception pin (SINn).

■ USART Direct Pin Access

The USART provides the ability for the software to access directly to serial input or output pin. The software can always monitor the incoming serial data by reading the SIOP bit of the ESCRn. If setting the Serial Output Pin direct access Enable (SOPE) bit of the ESCRn the software can force the SOTn pin to a desired value. Note that this access is only possible if the transmission shift register is empty (i.e. no transmission activity).

In LIN mode this function can be used for reading back the own transmission and is used for error handling if something is physically wrong with the single-wire LIN-bus.

Notes:

- Write the desired value to ESCRn:SIOP before enabling the output pin access to prevent undesired output level because ESCRn:SIOP holds the last written value.
- During a Read-Modify-Write operation the ESCRn:SIOP bit returns the actual value of the SOTn pin in the read cycle instead of the value of SINn during a normal read instruction.

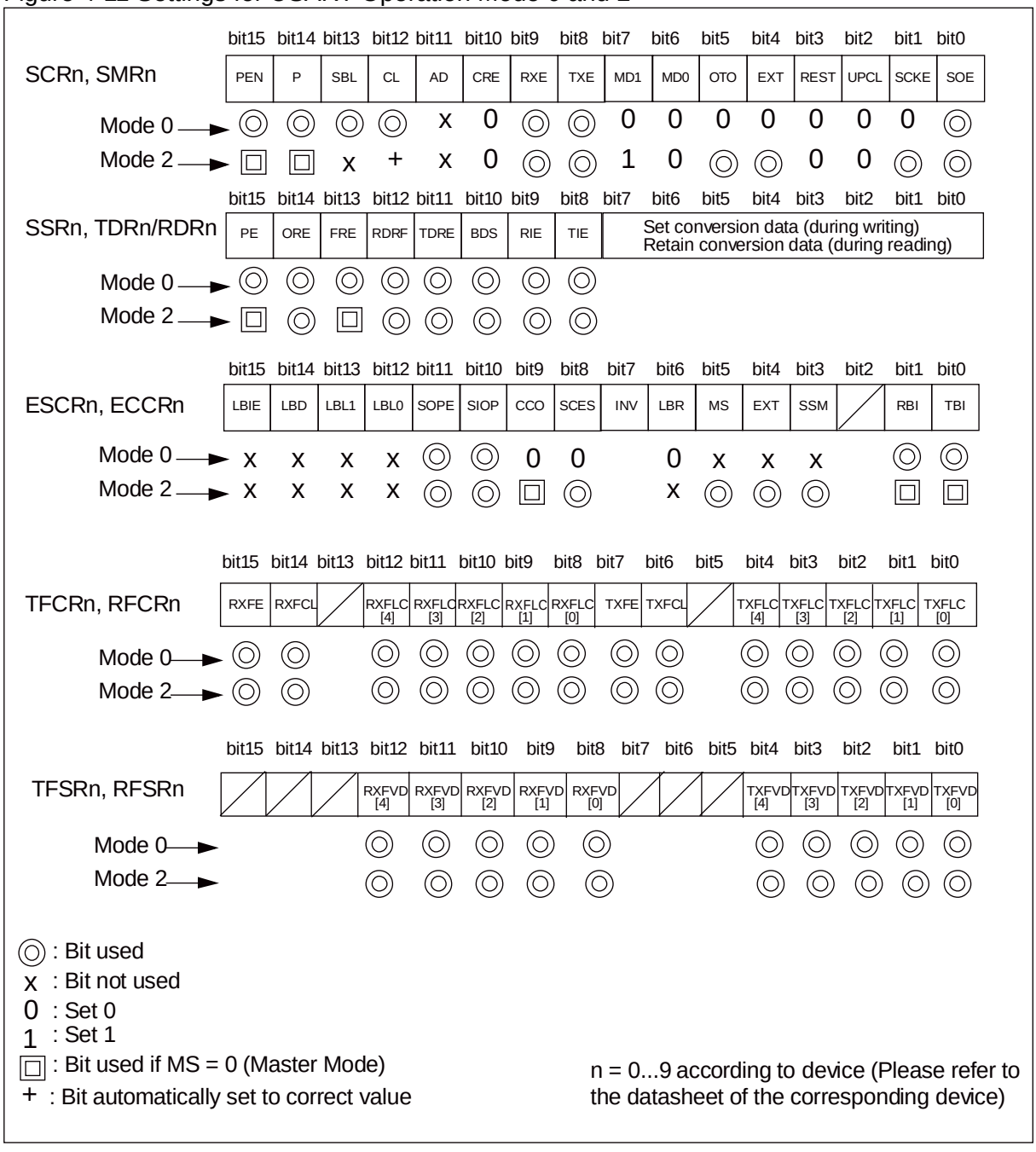
4.6. Bidirectional Communication Function (Normal Mode)

In operation mode 0 or 2, normal serial bidirectional communication is available. Select operation mode 0 for asynchronous communication and operation mode 2 for synchronous communication.

■ Bidirectional Communication Function

The settings shown in Figure 4-11 are required to operate USART in normal mode (operation mode 0 or 2).

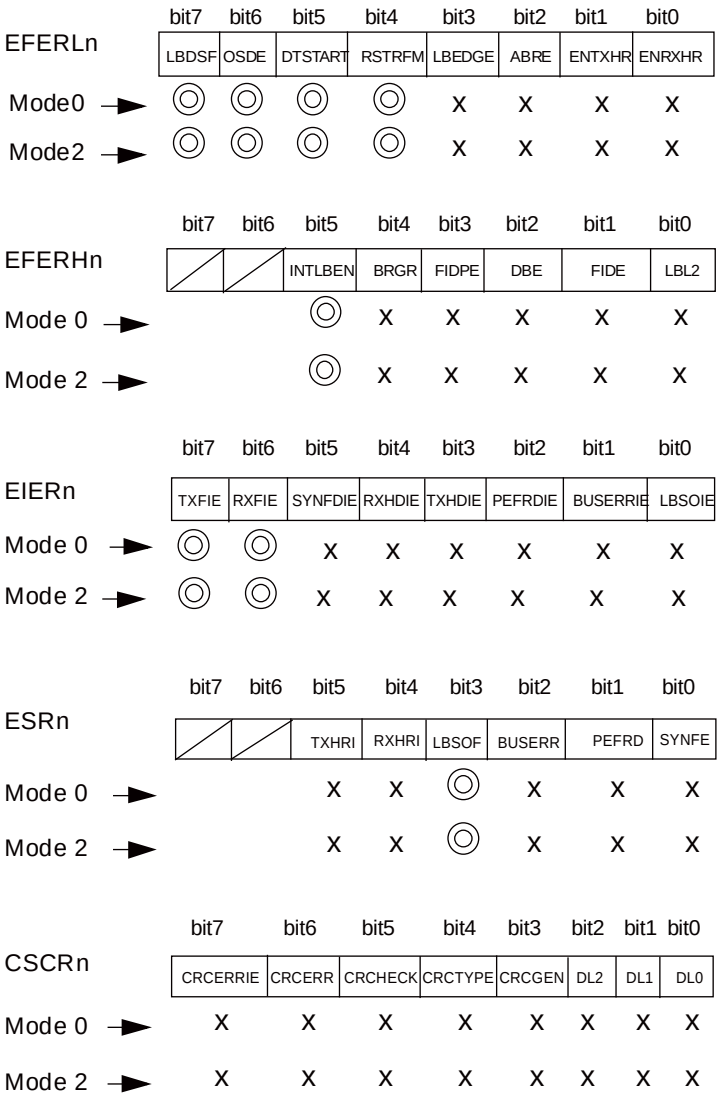
Figure 4-11 Settings for USART Operation Mode 0 and 2



◎ : Bit used
x : Bit not used
0 : Set 0
1 : Set 1
□ : Bit used if MS = 0 (Master Mode)
+ : Bit automatically set to correct value

n = 0...9 according to device (Please refer to the datasheet of the corresponding device)

Figure 4-12 Settings for USART Operation Mode 0 and 2 (contd)



◎ : Bit used
x : Bit not used
0 : Set 0
1 : Set 1
□ : Bit used if MS = 0 (Master Mode)
+ : Bit automatically set to correct value

n = 0...9 according to device (Please refer to the datasheet of the corresponding device)

● Inter-CPU Connection

As shown in Figure 4-13, interconnect two CPUs in USART mode 2.

Figure 4-13 Connection Example of USART Mode 2 Bidirectional Communication

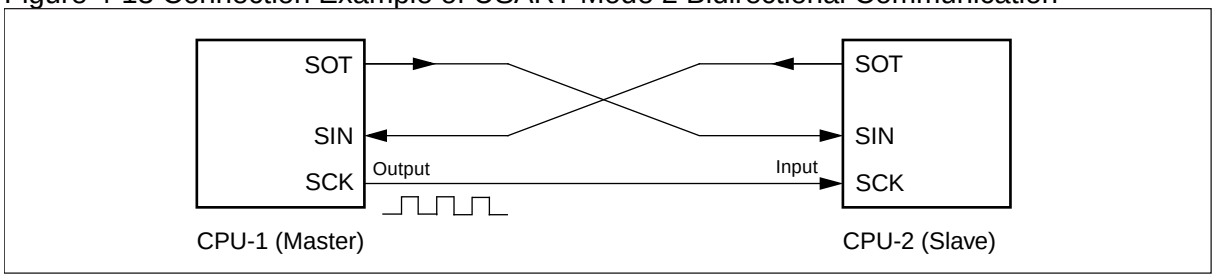
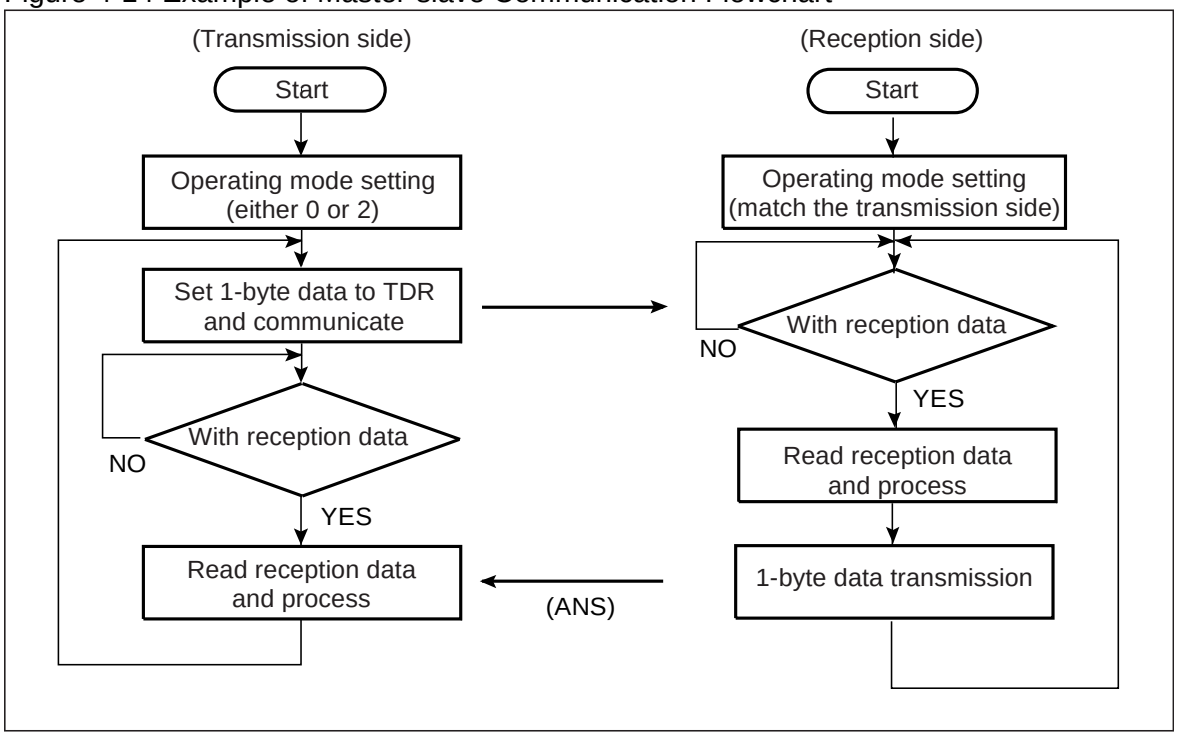


Figure 4-14 Example of Master-slave Communication Flowchart



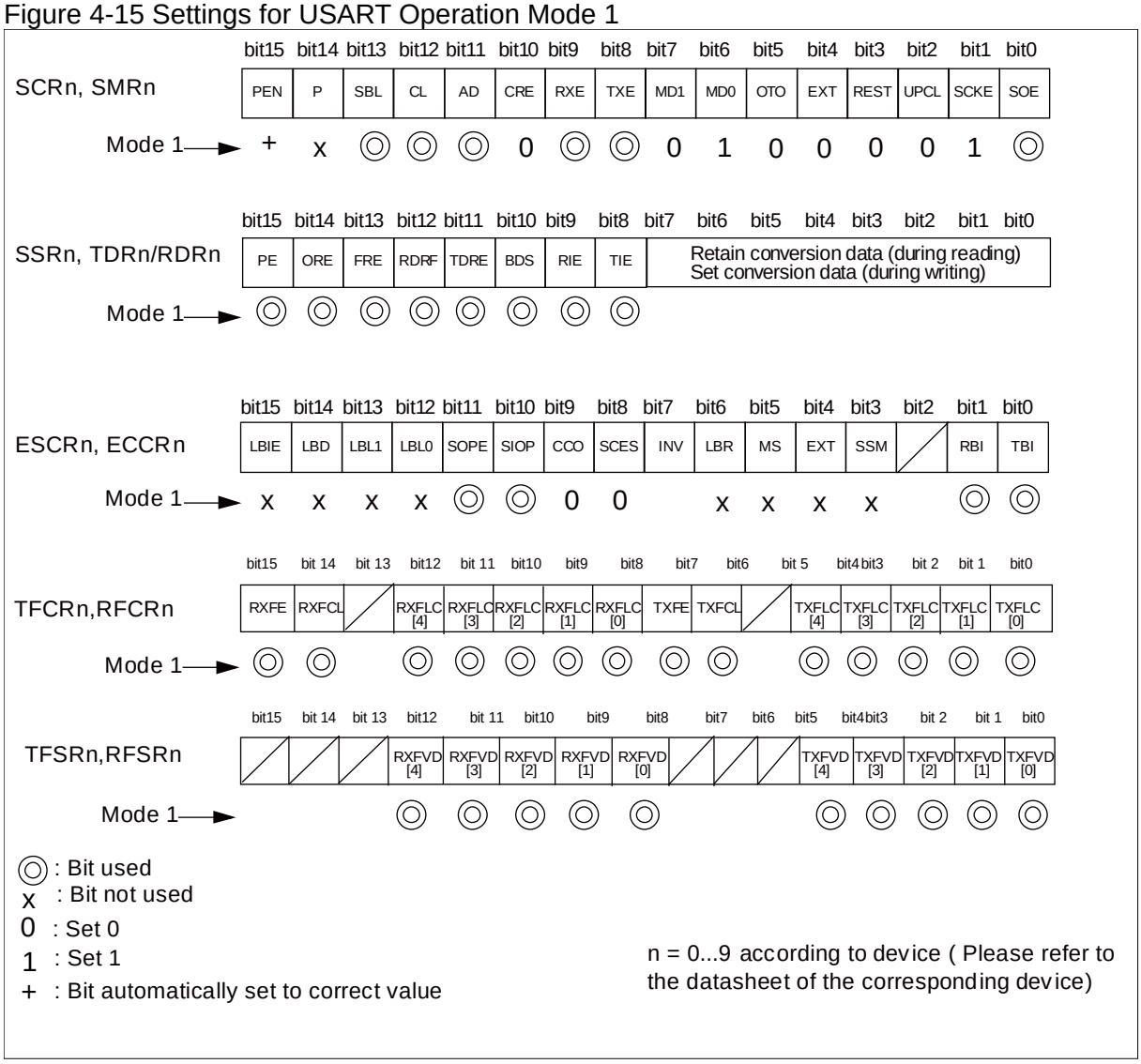
4.7. Master-Slave Communication Function (Multiprocessor Mode)

USART communication with multiple CPUs connected in master-slave mode is available for both master or slave systems.

■ Master-Slave Communication Function

The settings shown in Figure 4-15 are required to operate USART in multiprocessor mode (operation mode 1).

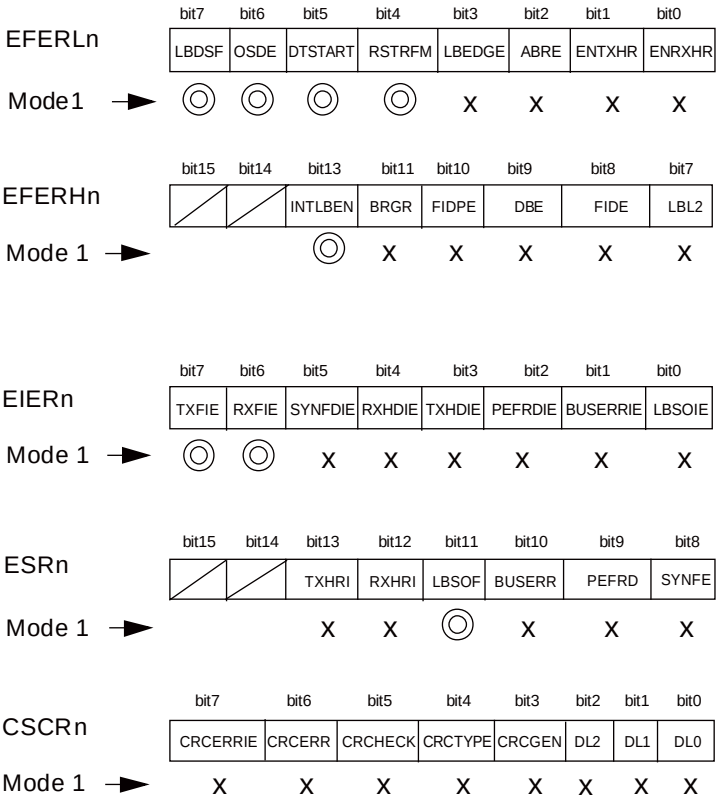
Figure 4-15 Settings for USART Operation Mode 1



◎ : Bit used
x : Bit not used
0 : Set 0
1 : Set 1
+ : Bit automatically set to correct value

n = 0...9 according to device (Please refer to the datasheet of the corresponding device)

Figure 4-16 Settings for USART Operation Mode 1 (Contd)



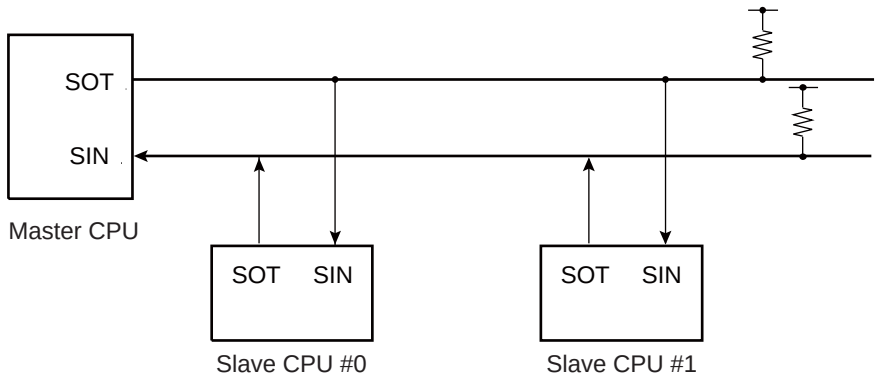
◎ : Bit used
x : Bit not used
0 : Set 0
1 : Set 1
□ : Bit used if MS = 0 (Master Mode)
+ : Bit automatically set to correct value

n = 0...9 according to device (Please refer to the datasheet of the corresponding device)

● Inter-CPU Connection

As shown in Figure 4-17, a communication system consists of one master CPU and multiple slave CPUs connected to two communication lines. USART can be used for the master or slave CPU.

Figure 4-17 Connection Example of USART Master-Slave Communication



● Function Selection

Select the operation mode and data transfer mode for master-slave communication as shown in Table 4-4.

Table 4-4 Selection of the Master-Slave Communication Function

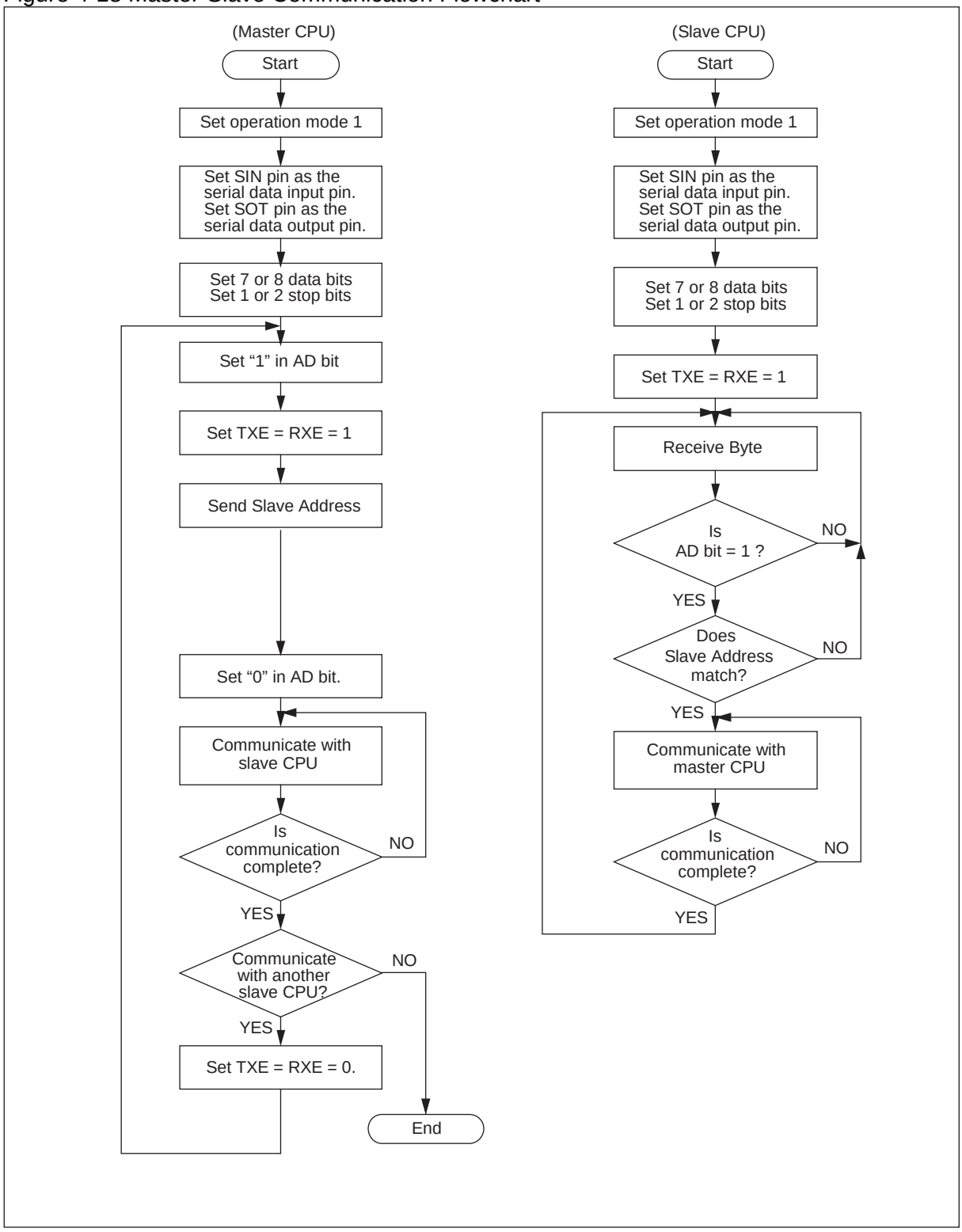
	Operation mode		Data	Parity	Synchronization method	Stop bit	Bit direction
	Master CPU	Slave CPU					
Address transmission and reception	Mode 1 (transmit/receive AD-bit)	Mode 1 (transmit/receive AD-bit)	AD="1" + 7- or 8-bit address	None	Asynchronous	1 or 2 bits	LSB or MSB first
Data transmission and reception			AD="0" + 7- or 8-bit data				

Communication procedure

When the master CPU transmits address data, communication starts. The AD bit in the address data is set to 1, and the communication destination slave CPU is selected. Each slave CPU checks the address data using a program. When the address data indicates the address assigned to a slave CPU, the slave CPU communicates with the master CPU.

Figure 4-18 shows a flowchart of master-slave communication (multiprocessor mode).

Figure 4-18 Master-Slave Communication Flowchart



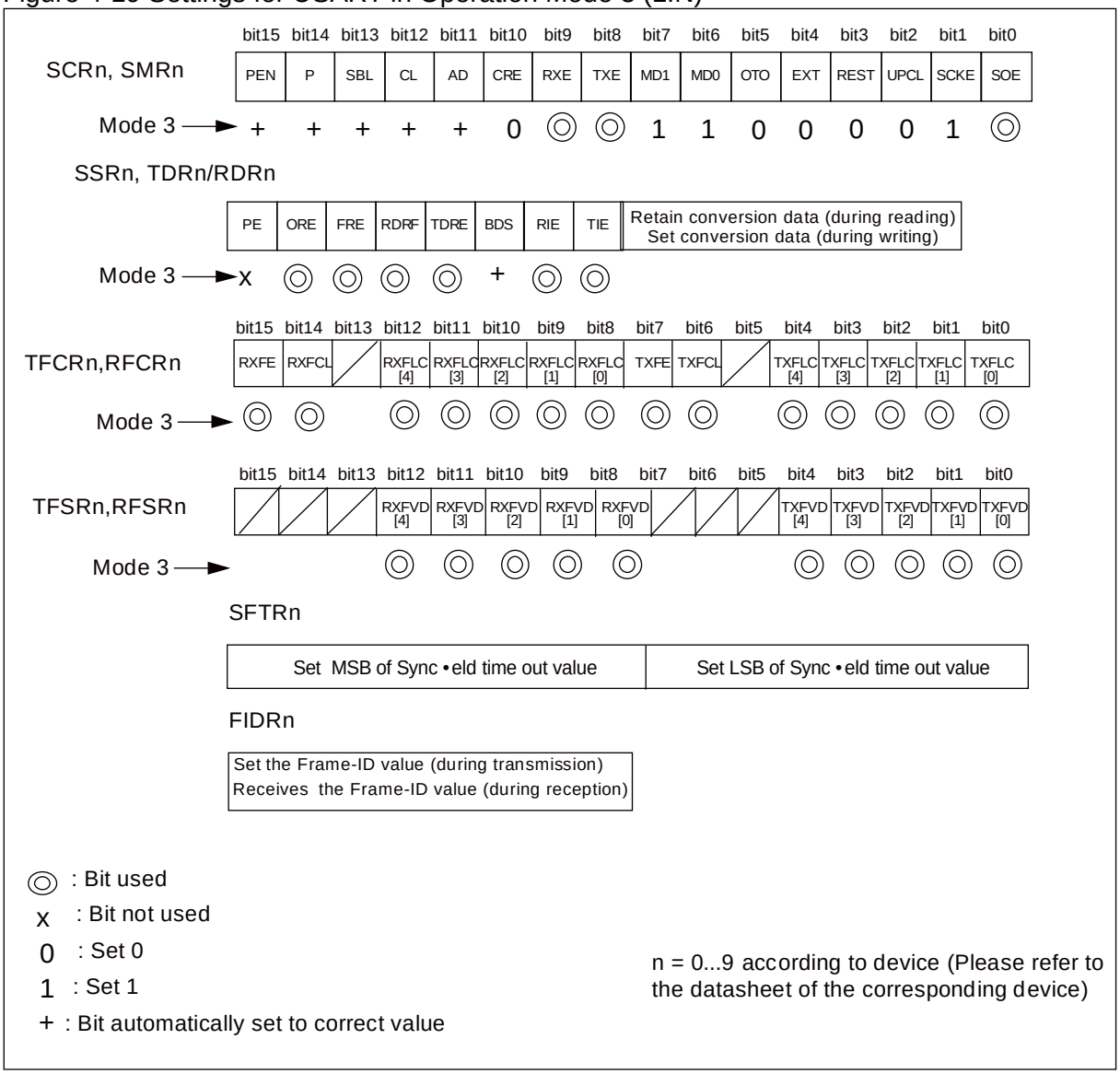
4.8. LIN Communication Function

USART communication with LIN devices is available for both LIN master or LIN slave systems.

■ LIN-Master-Slave Communication Function

The settings shown in the figure below are required to operate USART in LIN communication mode (operation mode 3).

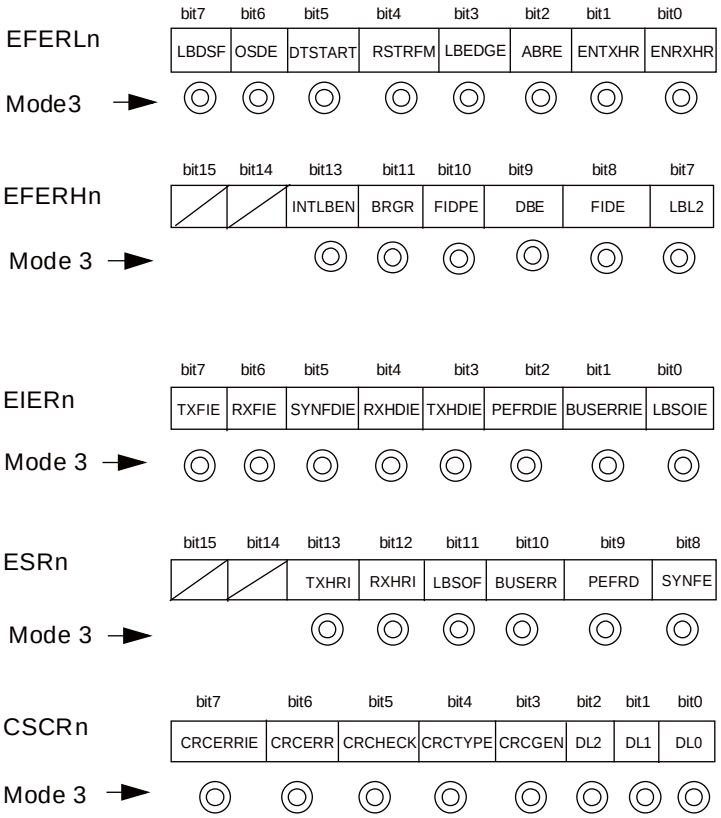
Figure 4-19 Settings for USART in Operation Mode 3 (LIN)



◎ : Bit used
x : Bit not used
0 : Set 0
1 : Set 1
+ : Bit automatically set to correct value

n = 0...9 according to device (Please refer to the datasheet of the corresponding device)

Figure 4-20 Settings for USART in Operation Mode 3 (Contd)



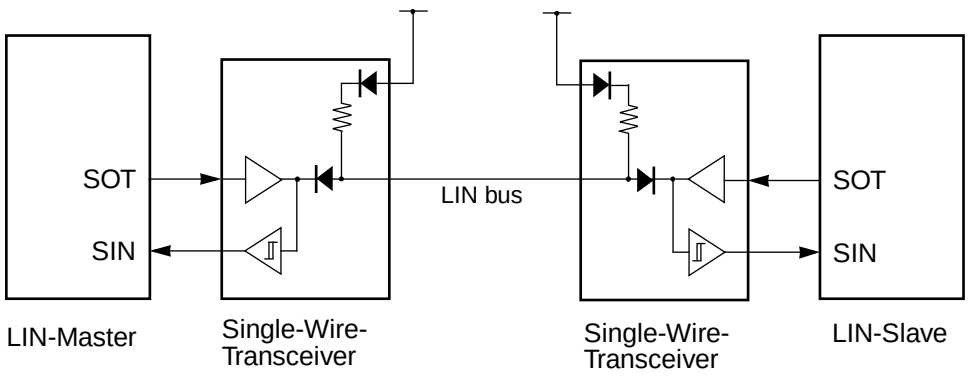
◎ : Bit used
x : Bit not used
0 : Set 0
1 : Set 1
□ : Bit used if MS = 0 (Master Mode)
+ : Bit automatically set to correct value

n = 0...9 according to device (Please refer to the datasheet of the corresponding device)

● LIN Device Connection

As shown in the figure below, a communication system of one LIN-Master device and a LIN-Slave device. USART can operate both as LIN-Master or LIN-Slave.

Figure 4-21 Connection Example of a Small LIN-Bus System

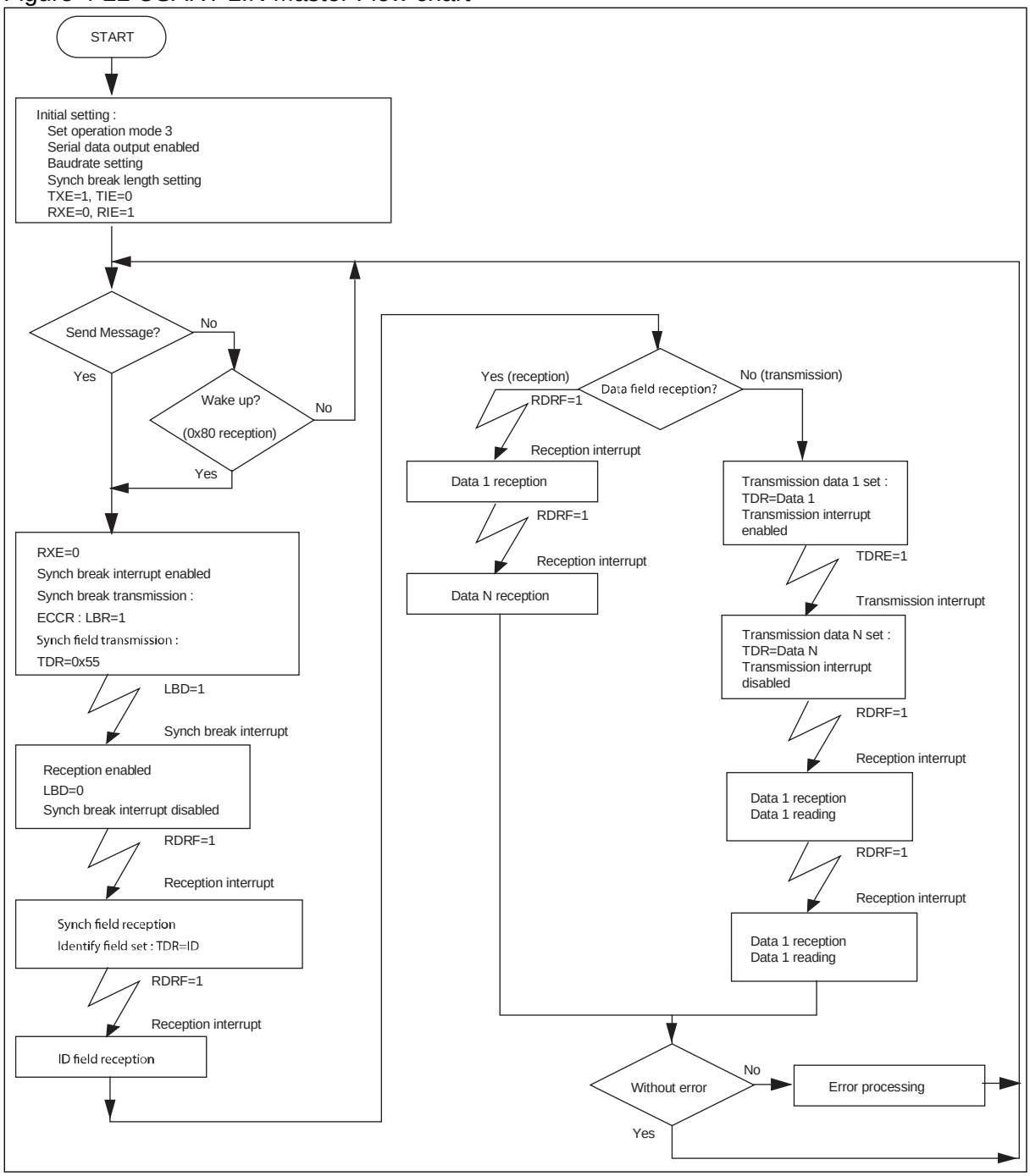


4.9. Sample Flowcharts for USART in LIN Communication (Operation Mode 3)

This section contains sample flowcharts for USART in LIN communication.

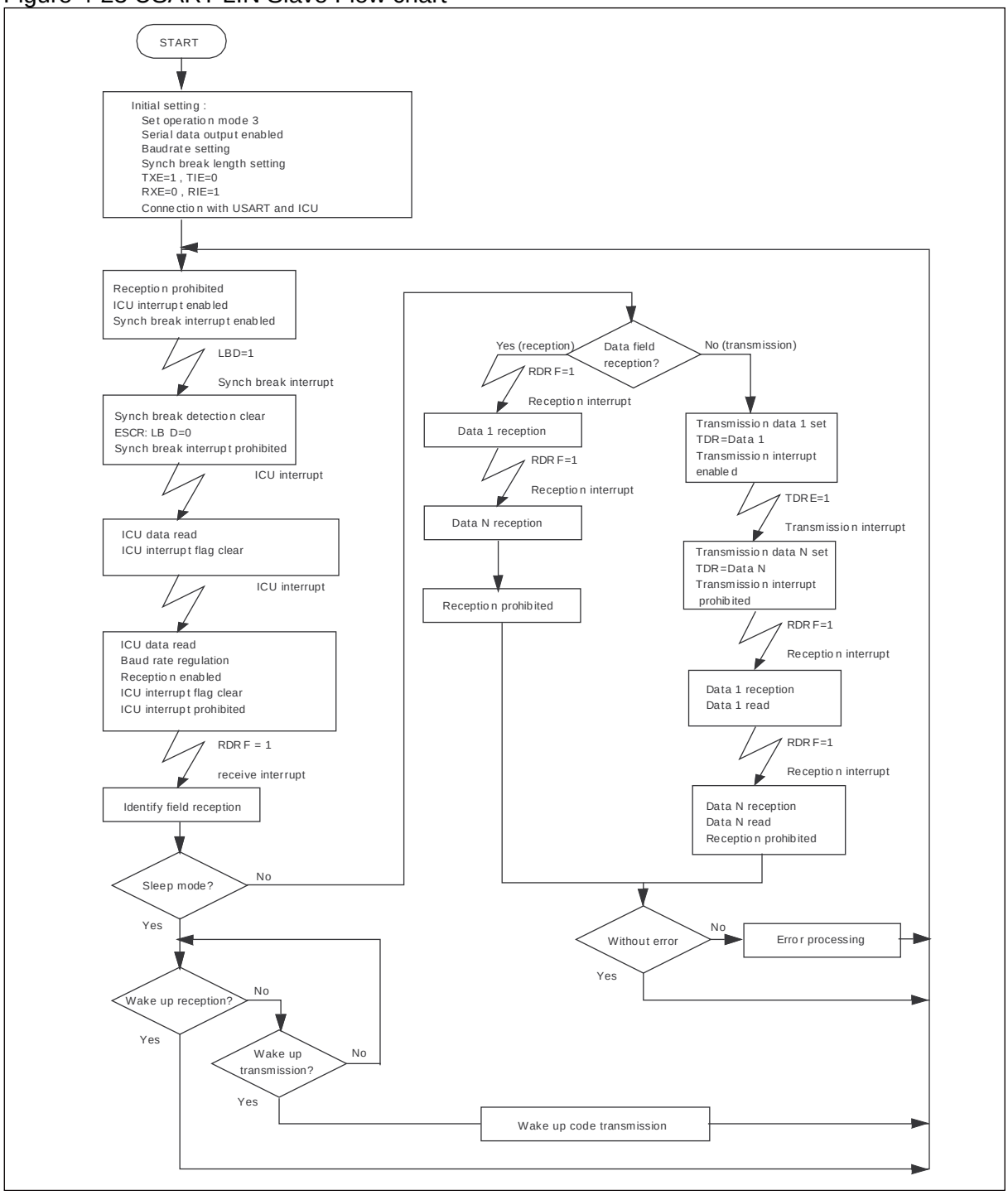
■ USART as Master Device

Figure 4-22 USART LIN Master Flow chart



■ USART as Slave Device

Figure 4-23 USART LIN Slave Flow chart



■ UART as Master Device with Additional Features

Figure 4-24 USART as Master Device Flow chart with All Additional Features used

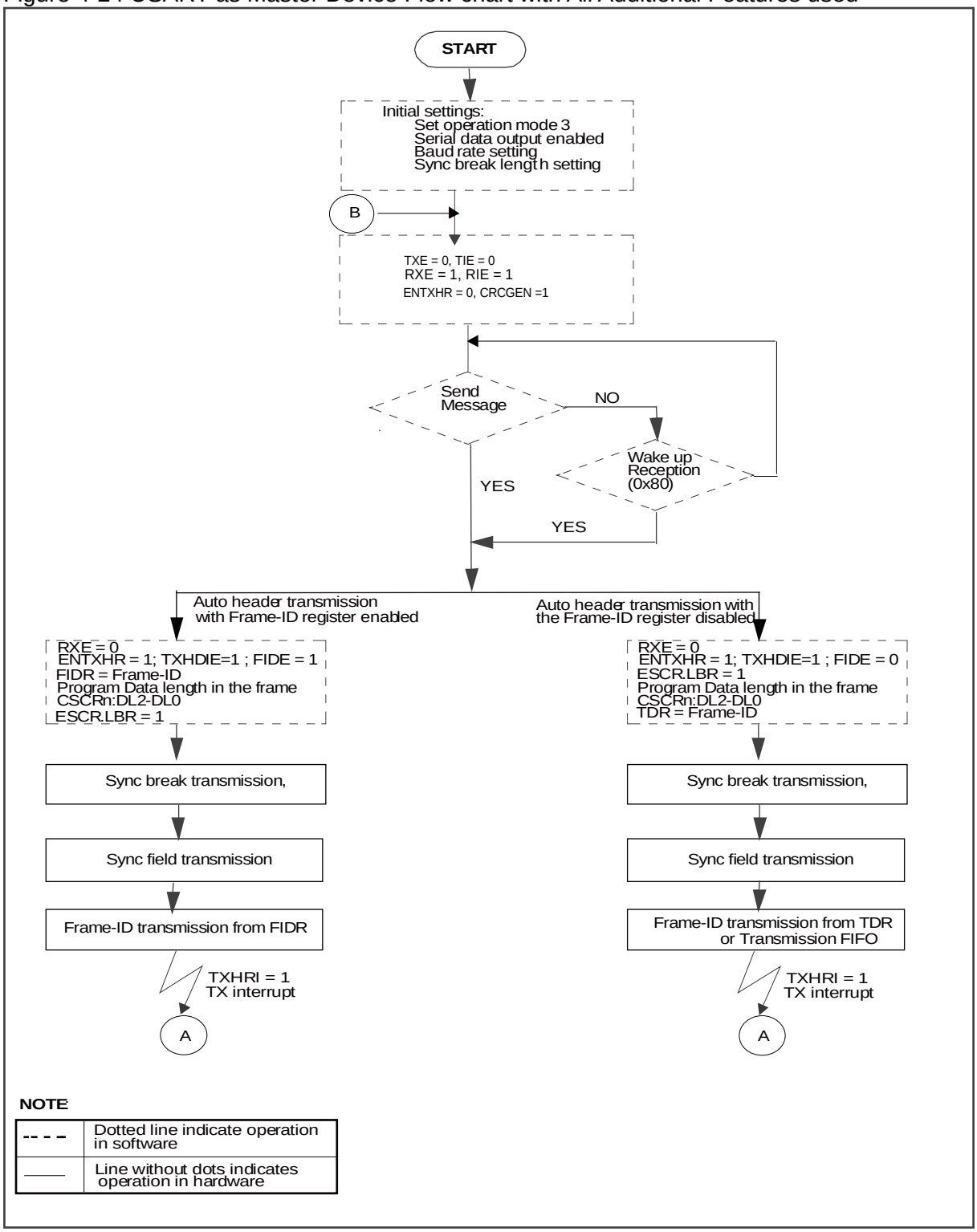
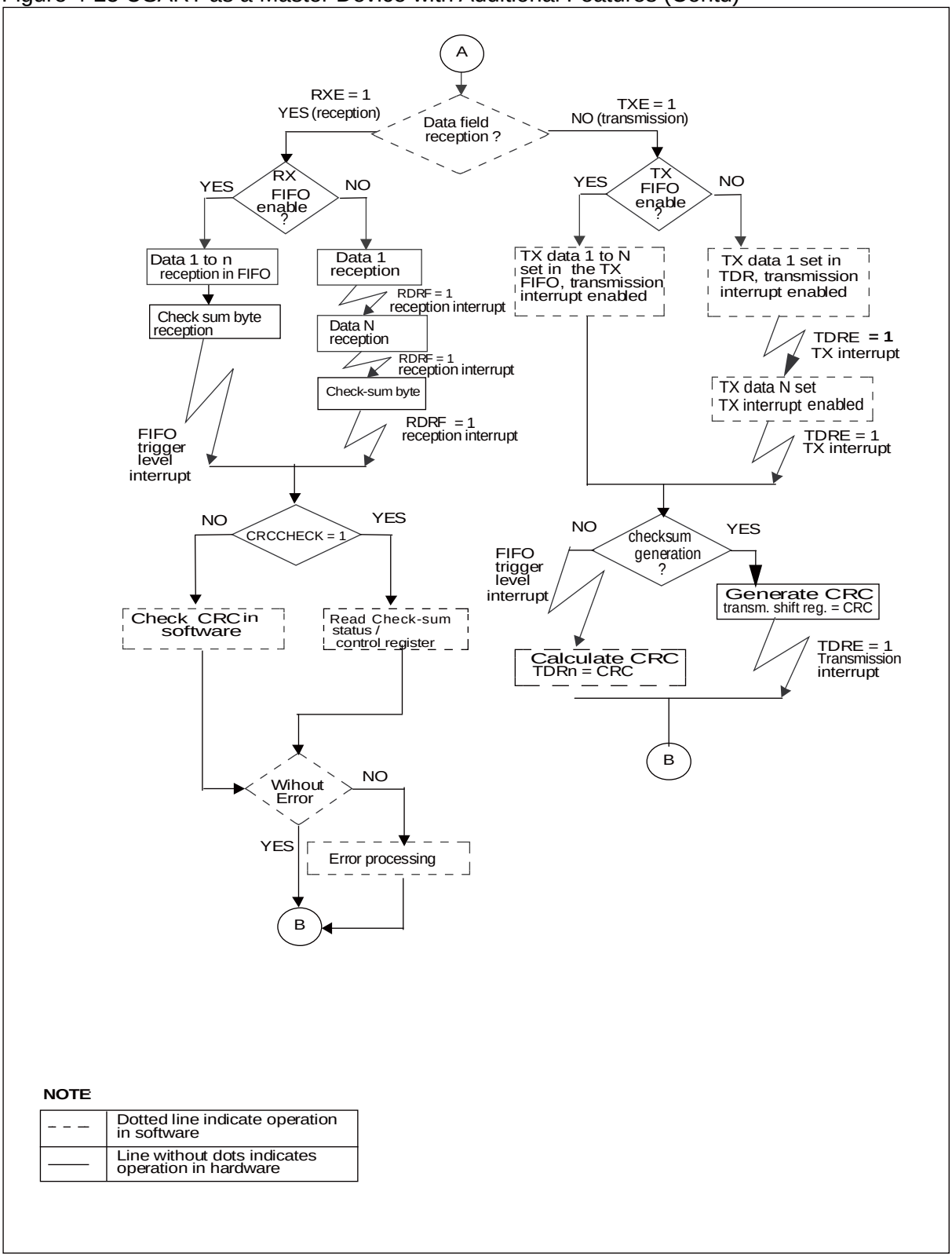


Figure 4-25 USART as a Master Device with Additional Features (Contd)



NOTE

- - -	Dotted line indicate operation in software
———	Line without dots indicates operation in hardware

Figure 4-26 USART as Slave Device with All Additional Features used

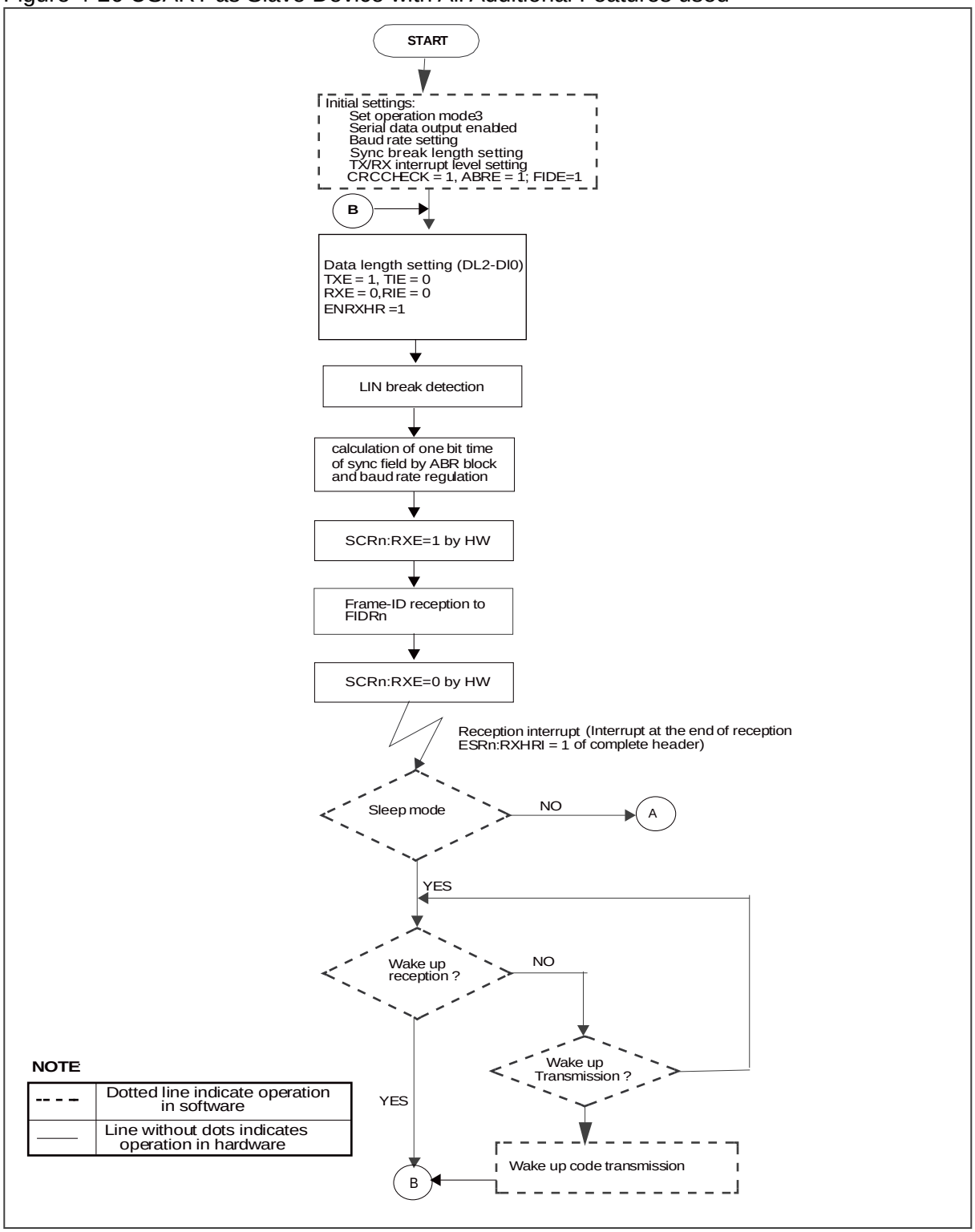
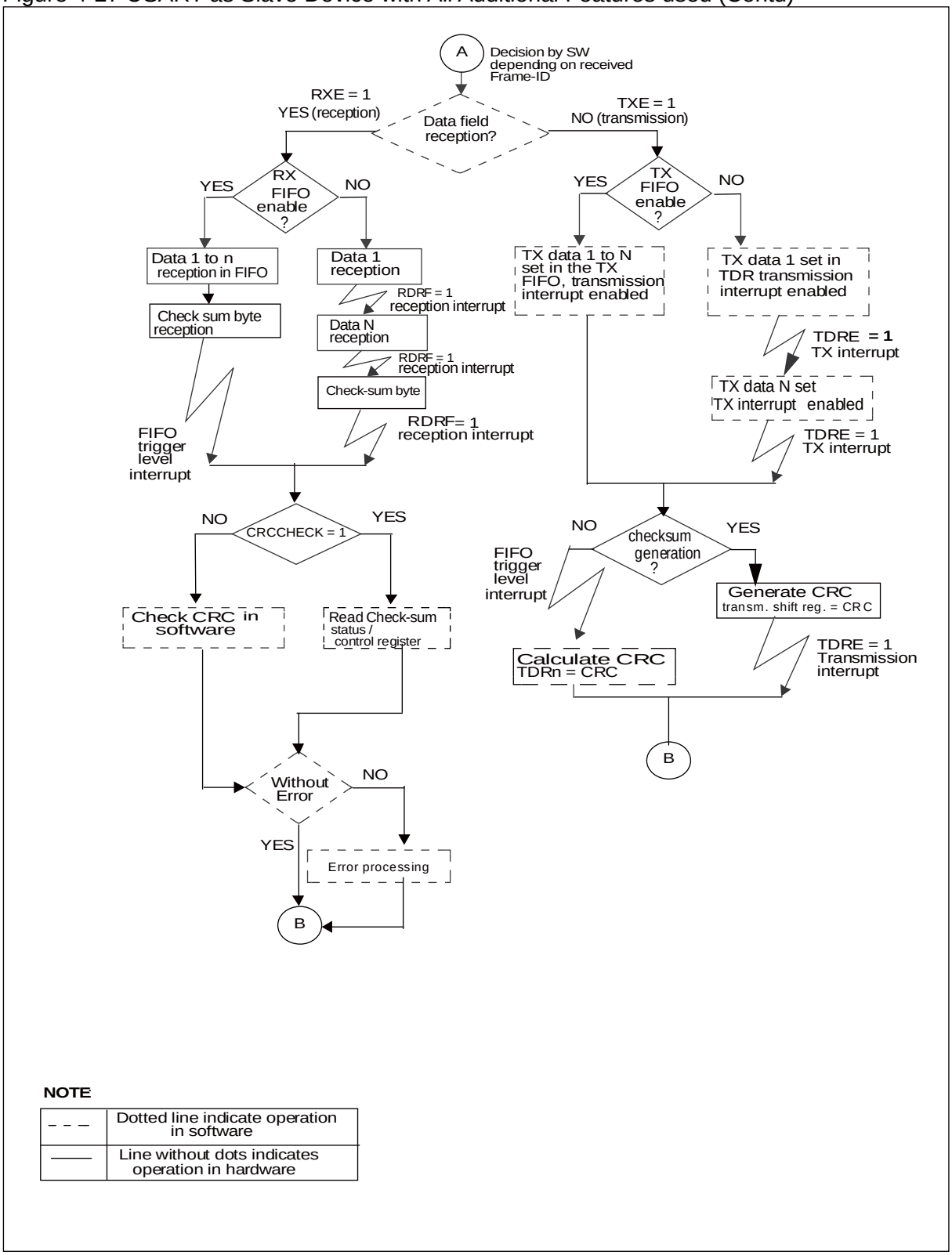


Figure 4-27 USART as Slave Device with All Additional Features used (Contd)



NOTE

- - -	Dotted line indicate operation in software
———	Line without dots indicates operation in hardware

5. Interrupts

USART uses both reception and transmission interrupts. An interrupt request can be generated for either of the following causes:

- Receive data is set in the Reception Data Register (RDRn), or a reception error occurs.
- Transmission data is transferred from the Transmission Data Register (TDRn) to the transmission shift register and started or when a transmission error occurs or last bit of a transmission was shifted out.
- A LIN break is detected.
- Parity error in the Frame-ID received.
- Checksum error while receiving data part of LIN frame.
- Sync field timeout error during detection of LIN Sync field.
- DMA is available for transmission and reception interrupts. Error condition causes a DMA-Stop event.

■ LIN-USART Interrupts

Table 5-1 Interrupt Control Bits and Interrupt Causes of LIN-USART

Reception/ transmission/ Test mode/ ICU	Interrupt request flag bit	Flag Register	Operation mode				Interrupt cause	Interrupt cause enable bit	How to clear the Interrupt Request
			0	1	2	3			
Reception	RDRF	SSRn	❍	❍	❍	❍	receive data is written to RDRn	SSRn:RIE	Receive data is read
	ORE	SSRn	❍	❍	❍	❍	Overrun error		"1" is written to clear rec. error bit (SCRn: CRE)
	FRE	SSRn	❍	❍	*1	❍	Framing error		
	PE	SSRn	❍	×	*1	×	Parity error in data bit in modes 0,2		
	LBD	ESCRn	×	×	×	❍	LIN synch break detected	ESCRn:LBIE	"0" is written to ESCRn: LBD
	CRCERR	CSCRn	×	×	×	❍	Error found in checksum verification	CSCRn:CRCERRIE	Cleared by writing "0" to the status flag CSCRn:CRCERR
	SYNFE	ESRn	×	×	×	❍	Sync field not detected because of time out error.	EIERn:SYNFEIE	Cleared by writing "0" to the status flag ESRn:SYNFE
	BUSERR	ESRn	×	×	×	❍	When buserror occurs	EIERn:BUSERRIE	Cleared by writing "0" to the status flag ESRn:BUSERR
	PEFRD	ESRn	×	×	×	❍	Parity error in Frame-ID received	EIERn:PEFRDIE	Cleared by writing "0" to the status flag ESRn:PEFRD
	RXHRI	ESRn	×	×	×	❍	Automatic reception of LIN frame header completed.	EIERn:RXHDIE	Cleared by writing "0" to the status flag ESRn:RXHRI

Reception/ transmission/ Test mode/ ICU	Interrupt request flag bit	Flag Register	Operation mode				Interrupt cause	Interrupt cause enable bit	How to clear the Interrupt Request
			0	1	2	3			
Transmission	TDRE	SSRn	❍	❍	❍	❍	TDRn/FIFO empty	SSRn:TIE	Write data to TDRn.
	LBSOF	EIERn	❍	❍	❍	❍	Transmitted Last bit of data in sync mode and stop bit in async mode	EIERn: LBSOIE	Cleared by writing next byte of data to TDRn/TX-FIFO or writing "0" to ESRn:LBSOF.
	TXHRI	ESRn	×	×	×	❍	Automatic transmission of LIN frame header completed.	EIERn:TXHDIE	Cleared by writing "0" to the status flag ESRn:TXHRI
Input Capture Unit	ICPy*2	ICSy*2	×	×	×	❍	1st falling edge of LIN synch field	ICSy:ICEy*2	disable ICPy temporarily
							5th falling edge of LIN synch field		disable ICPy

❍: Used
×: Unused
*1: Only available if ECCRn:SSM = 1
*2: For the Input Capture Unit number "y" see "Section 1.6 Input Capture Unit source select for LIN-USART"

● Receive Interrupt

If one of the following events occurs in reception mode, the corresponding flag bit of the Serial Status Register (SSRn), the Extended Serial Interrupt Register (ESIRn) and Extended Status register (ESRn) is set to "1":

- Data reception is complete, i. e. the received data was transferred from the serial input shift register to the Reception Data Register (RDRn) and data can be read: SSRn:RDRF, ESIRn:RDRF
- Overrun error, i. e. SSRn:RDRF = 1 and RDR was not read by the CPU and received next serial data: SSRn:ORE
- Framing error, i. e. a stop bit was expected, but a "0"-bit was received : SSRn:FRE
- Parity error, i. e. a wrong parity bit was detected : SSRn:PE
- Checksum error in the LIN frame received : ESRn:CRCERR
- Parity error in the LIN Frame-ID received : ESRn:PEFRD
- Reception of complete header: ESRn:RXHRI
- Timeout error during reception of LIN Sync field : ESRn:SYNFE
- Bus error during detection of physical bus error. ESRn:BUSERR in LIN mode.

If ESIR:AICD = 0 and at least one of the flag bits SSRn:RDRF, SSRn:ORE, SSRn:FRE, SSRn:PE is set to "1", and the reception interrupt is enabled (SSRn:RIE = 1), a reception interrupt request is generated.
If ESIR:AICD = 1 and at least one of the flag bits ESIRn:RDRF, SSRn:ORE, SSRn:FRE, SSRn: PE is set to "1" and the reception interrupt is enabled (SSRn:RIE = 1), a reception interrupt request is generated.
Similarly if (ESIRn:AICD = 0 or ESIRn:AICD = 1) with at least one of the flag bits and corresponding interrupt enable bits mentioned below are set to "1", a reception interrupt is generated

- ESRn:PEFRD and EIERn:PEFRDIE
- ESRn:SYNFE and EIERn:SYNFEIE
- ESRn:BUSERR and EIERn:BUSERRIE
- ESRn:CRCERR and CSCRn:CRCERRIE
- ESRn:RXHRI and EIERn:RXHDIE
- The interrupt corresponding to these flag bits are cleared by writing "0" to the status flag.

If the Reception Data Register (RDRn) is read, the SSRn:RDRF flag is automatically cleared to "0". Note SSRn:RDRF can only be cleared by reading the Reception Data REgister (RDRn) or by writing "1" to SCRn:UPCL.
The error flags (SSR:FRE, SSRn:PE and SSRn:ORE) are cleared to "0", if a "1" is written to the Clear Reception Error (CRE) flag bit of the Serial Control Register (SCRn). The RDRn contains only valid data if the RDRF flag is "1" and no error bits are set.
The error flag ESRn:CRCERR is cleared by writing "0" to it.
Note that the CRE flag is "write only" and by writing a "1" to it, it is internally held to "1" for one CLKP1 clock cycle.

● Transmission Interrupt

A transmission interrupt is generated if one of the following events occurs.
If transmission data is transferred from the Transmission Data Register (TDRn) to the transfer shift register and transfer is started, the Transmission Data Register Empty flag bit (TDRE) of the Serial Status Register (SSR) is set to "1". In this case an interrupt request is generated, if the Transmission Interrupt Enable (TIE) bit of the SSR was set to "1" before. Note, that the initial value of TDRE (after hardware or software reset) is "1". So an interrupt is generated immediately then, if the TIE flag is set to "1". Also note that the only way to reset the TDRE flag is writing data to the Transmission Data Register (TDRn).

- If last bit of data in sync mode or a stop bit in async mode is shifted out. In this case an interrupt request is generated, if the interrupt enable EIERn:LBSOIE is set to "1".

Note:

Do not use SSRn:TIE and EIERn:LBSOIE together at the same time.

Note:

When AICD = 0, the usart cannot be used with DMA. If DMA is used to handle data transfer of the USART, set AICD = 1.

● LIN Synchronization Break Interrupt

This paragraph is only relevant, if USART operates in mode 3 and when the automatic header transmission and reception is disabled (EFERLn:ENTXHR = 0 and EFERHn:ENRXHR = 0).
If the bus (serial input) goes "0" (dominant) for more than 11 bit times, the LIN Break Detected (LBD) flag bit of the Extended Status/Control Register (ESCRn) is set to "1". Note, that in this case after 9 bit times the reception error flags are set to "1", therefore the RXE flag has to set to "0", if only a LIN synch break detection is desired.
The interrupt and the ESCRn:LBD flag are cleared after writing a "0" to the LBD flag. This has to be performed before input capture interrupt for LIN sync field.

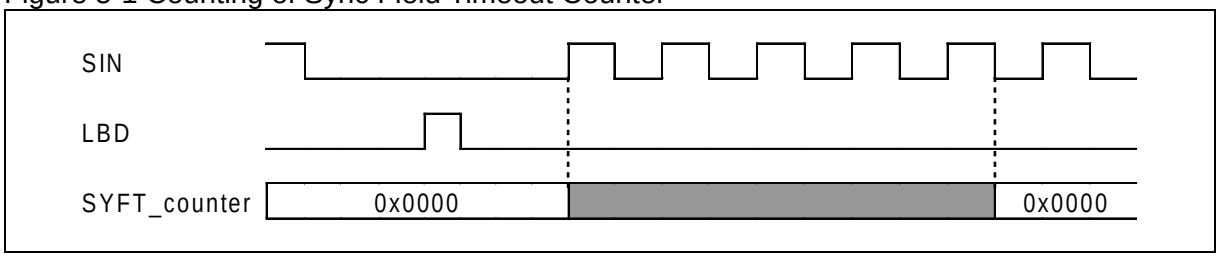
● LIN Synchronization Field Edge Detection Interrupts

This paragraph is only relevant, if USART operates in mode 3 as a LIN slave (when auto baud rate det./adj. feature is disabled (EFERn:ABRE = 0)). After a LIN break detection the next falling edge of the reception bus is indicated by USART. Simultaneously an internal signal connected to the ICU associated to the LIN-USART is set to "1" (please see "Section 1.6 Input Capture Unit source select for LIN-USART" for an overview which ICU is associated to which LIN-USART).
This signal is reset to "0" after the fifth falling edge of the LIN synchronization field. In both cases the associated ICU generates an interrupt, if "both edge detection" and the ICU interrupt are enabled. The difference of the ICU counter values is the serial clock multiplied by 8. Dividing it by 8 results in the new detected and calculated baud rate for the dedicated reload counter. This value - 1 has then to be written to the Baud Rate Generator Registers (BGRn). There is no need to restart the reload counter, because it is automatically reset if a falling edge of a start bit is detected.

● Sync Field Timeout Interrupt

This paragraph is relevant only for mode 3 operation when EFERLn:ABRE = 1. After the LIN break detection, the sync field timeout counter starts counting on the first rising edge of SIN (serial data input).

Figure 5-1 Counting of Sync Field Timeout Counter



The internal ABR counter stops counting on two conditions listed below:

1. When the fifth falling edge of the sync field gets detected.
2. When the sync field timeout counter equals the timeout value programmed in 16 bit sync field timeout register (SFTRn). In this case ESRn:SYNCE gets set. This results in an interrupt when EIERn:SYNFEIE is set. The internal counter uses the peripheral clock (CLKP1) for counting.

If ESRn:SYNFE is set, the Baud rate reload register is not updated with the value from Autobaud rate detection circuit. Therefore always clear ESRn:SYNFE before a new LIN-frame should be received.
Formula for calculating the value to be programmed in 16bit sync field time out register (SFTRn)

Sync field timeout register value (SFTRn) = Sync Field timeout time / CLKP1 time period

Sync field timeout time = one bit time for break delimiter + 8 bit for sync field + 1 start bit + 1 stop bit.
= 11 bit times

Considering the baud rate as 19200 and resource clock frequency as 16MHz.
BGR value = (16000000/19200) - 1 = 832
Bit time of Sync field is 832 clock cycles of resource clock.
So Sync field timeout time = 832 × (11) = 9152
Considering deviation of 14% from the actual = (14/100 × 9152) + 9152 = 10433
Considering the deviation of 14% from the baud rate value programmed in the BGR register.

Table 5-2 Calculating example of Sync field timeout time of each frequency

CLKP frequency [MHz]	Baud rate value= 19200	Baud rate value= 9600
16	10433	20866
20	13049	26098
24	15675	31350
32	20899	41798
48	31350	62700
56	36754	-

Sync field time out error can cause a DMA-stop. To enable Sync field time out as stop condition for DMA, enabled the sync field time out interrupt (set EIERn:SYNFEIE to "1").

● Parity Error in Frame-ID Interrupt

This paragraph is relevant only for mode 3 operation, when EIERn:PEFRDIE = 1. When there is a parity error in a Frame-ID received, the ESRn:PEFRD flag gets set. A reception interrupt will be generated when the status flag PEFRD is "1" and the interrupt enable PEFRDIE is set. This interrupt can be cleared by writing "0" to the status flag (PEFRD). Parity error in Frame-ID can cause a DMA-stop. To enable Parity error in Frame-ID as stop condition for DMA, enabled the Parity error in Frame-ID interrupt (set EIERn:PEFRDIE to "1").

● Checksum Error Interrupt

This paragraph is relevant only for mode 3 operation, when CSCRn:CRCERRIE = 1. After the checksum byte is received, the internal sum calculated is added with received checksum and checked whether it is equal to 0xFF. If the sum calculated is not equal to 0xFF, then the CSCRn:CRCERR flag is set and reception interrupt is asserted (if the interrupt enable CSCRn:CRCERRIE is set to "1"). This interrupt is cleared by writing "0" to the flag CRCERR. CRC error can cause a DMA-stop. To enable CRC error as stop condition for DMA, enabled the CRC error interrupt (set CSCRn:CRCERRIE to "1").

● Bus error interrupt

This feature is enabled by setting the bit EFERLn.DBE to "1". The detection of bus error is done by enabling both transmission and reception, so that LIN node can read back its own transmission (as the LIN is single wire network). The physical bus error such as shorted to ground or Vcc can be found by comparing the value of transmitted and received data. If there is a difference between the transmitted and received value then ESRn:BUSERR flag is set. This will result in a reception interrupt if IERn:BUSERRIE is set to "1".

Bus error can cause a DMA-stop. To enable Bus error as stop condition for DMA, enabled the Bus error interrupt (set EIERn:BUSERRIE to "1").

■ USART DMA Functions

The USART can be operated with DMA, for reception or transmission. Please see "CHAPTER 4 DMA" for details about DMA.

Any one of following error condition causes a DMA-Stop event.

PE, ORE, FRE, BUSERR, PEFRD, CRCERR, SYNFE.

Note:

When USART is served by DMA, set ESIRn:AICD = "1".

In addition, in case of an error while reception, the ongoing DMA transfer would be stopped, if the SE bit of DMACS register of the corresponding DMA channel is set.

5.1. Reception Interrupt Generation and Flag Set Timing

The following are the reception interrupt causes: completion of reception (SSRn:RDRF), auto header detection (ESRn:RXHRI) and occurrence of a reception error (SSRn:PE, ORE, or FRE or ESRn:PEFRD or CRCERR).

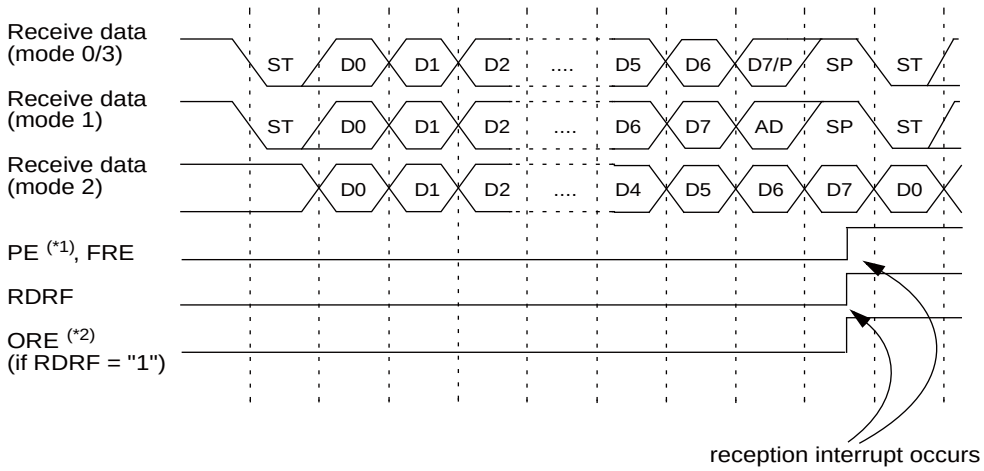
■ Reception Interrupt Generation and Flag Set Timing

Generally a reception interrupt is generated, if the received data is complete (when ESIRn:AICD = 0: SSRn:RDRF = 1, when ESIRn:AICD = 1, ESIRn:RDRF = 1) and the reception interrupt enable (RIE) bit of the Serial Status Register (SSRn) was set to "1". This interrupt is generated if the first stop bit is detected in mode 0, 1, 2 (if SSM = 1), 3, or the last data bit was read in mode 2 (if SSM = 0).

Note:

If a reception error has occurred, the Reception Data Register (RDRn) contains invalid data in each mode.

Figure 5-2 Reception Operation and Flag Set Timing (When reception FIFO is disabled)



*1: The PE flag will always remain "0" in mode 1 or 3.

*2: ORE only occurs, if the reception data is not read by the CPU (RDRF = 1) and another data frame is read.

ST: Start Bit SP: Stop Bit AD: Mode 1 (multi processor) address/data selection bit

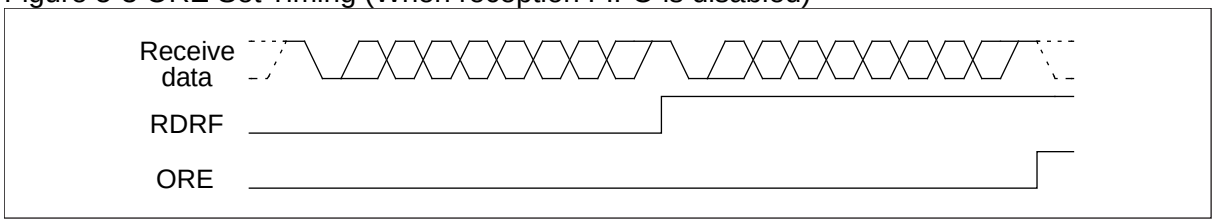
Note:

When reception FIFO is enabled (RFCRn:RXFE set to "1"), SSRn:RDRF flag will be asserted when the FIFO reaches the trigger level programmed in the reception FIFO control register (RFCRn:RXFLC).

Note:

The example in Figure 5-2 does not show all possible reception options for mode 0 and 3. Here it is: "7p1" and "8N1" (p = "E" [even] or "O" [odd]).

Figure 5-3 ORE Set Timing (When reception FIFO is disabled)



Note:

When reception FIFO is enabled (RFCRn:RXFE set to "1"), SSRn:ORE flag will be set when reception FIFO overflows (i.e.) when the reception FIFO is full and another data byte is received.

Figure 5-4 CRCERR Timing

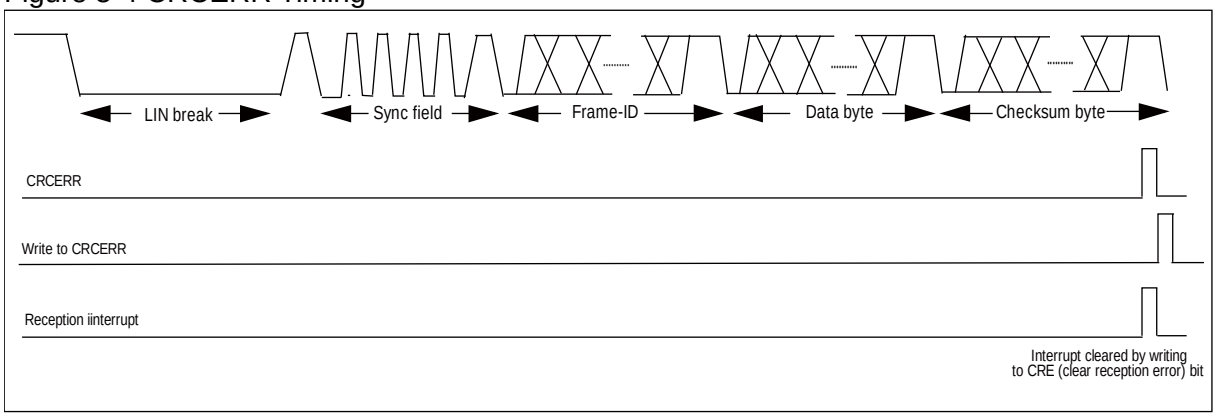


Figure 5-5 PEFRD Timing

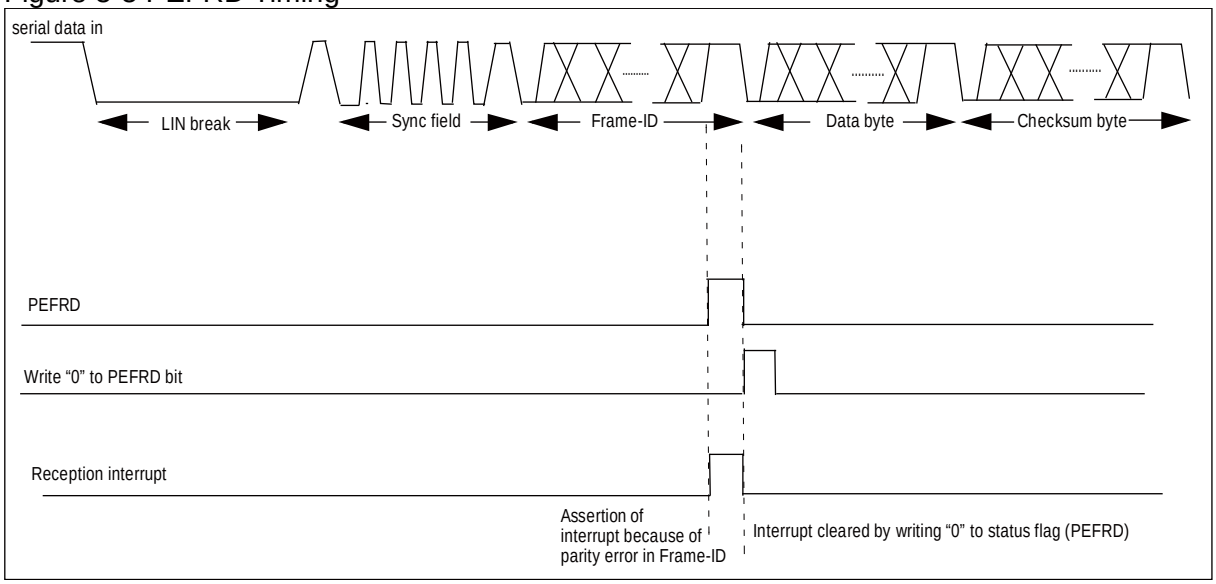
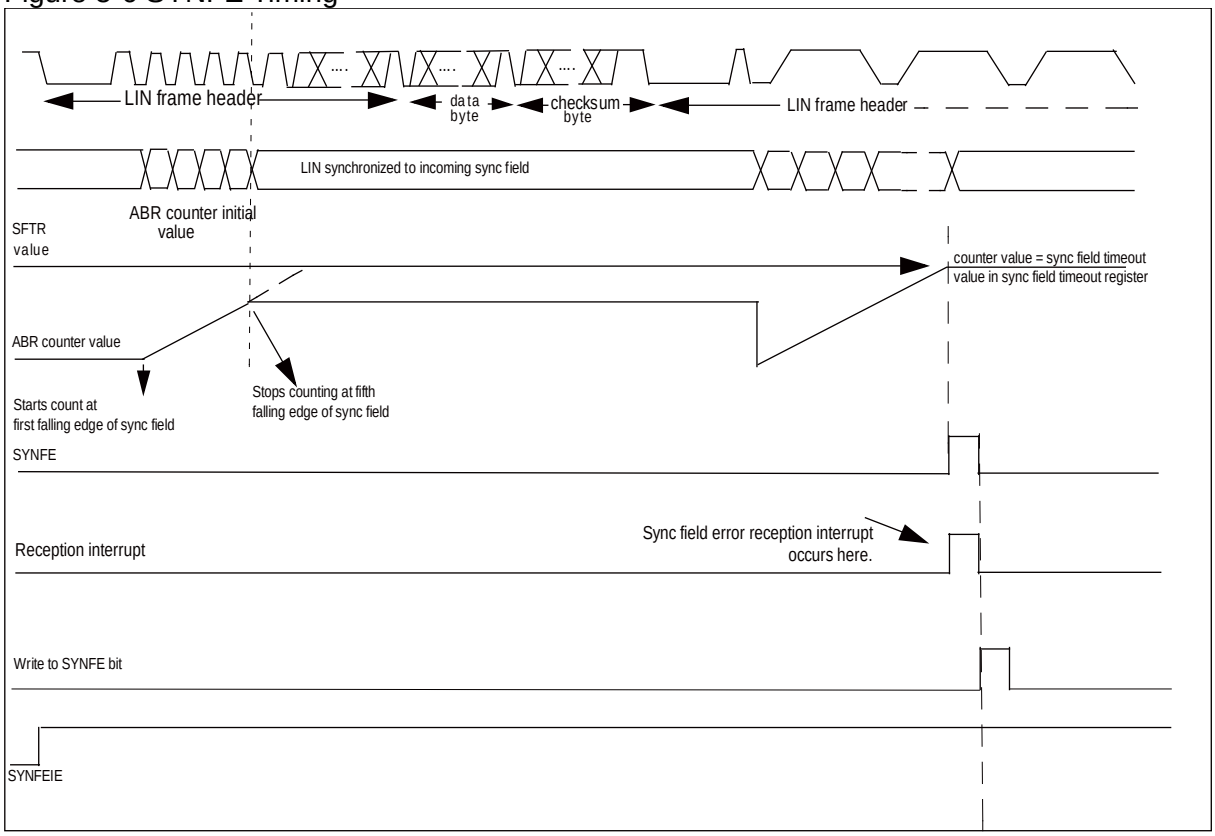


Figure 5-6 SYNFE Timing



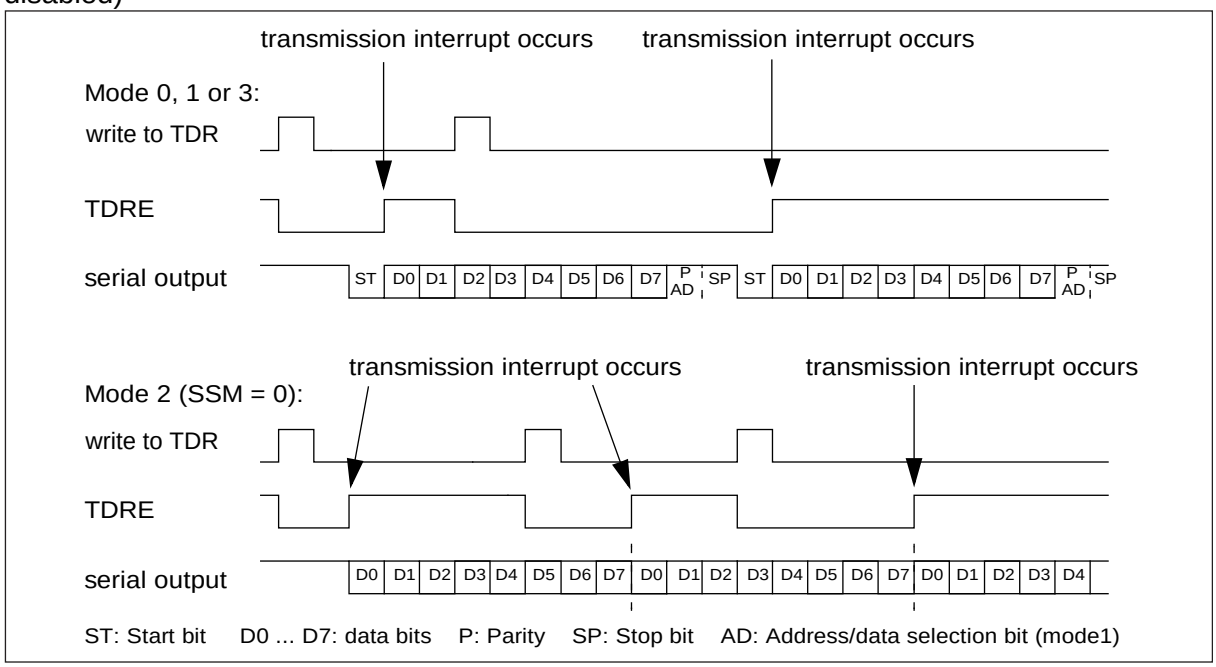
5.2. Transmission Interrupt Generation and Flag Set Timing

A transmission interrupt is generated when the transmission data is transferred from transmission data register (TDRn) to transmission shift register or at when last bit was shifted out.

■ Transmission Interrupt Generation and Flag Set Timing using TIE

A transmission interrupt is generated, when the next data to be sent is ready to be written to the Transmission Data Register (TDRn), i. e. the TDRn is empty, and the transmission interrupt is enabled by setting the Transmission Interrupt Enable (TIE) bit of the Serial Status Register (SSRn) to "1".
The Transmission Data Register Empty (TDRE) flag bit of the SSRn indicates an empty TDRn. Writing data to TDRn clears the TDRE bit.
The following figure demonstrates the transmission operation and flag set timing for the four modes of USART.

Figure 5-7 Transmission Operation and Flag Set Timing (When transmission FIFO is disabled)



Note:

The example in Figure 5-7 does not show all possible transmission options for mode 0. Here it is: "8p1" (p = "E" [even] or "O" [odd]). Parity is not provided in mode 3 or 2, if SSM = 0.

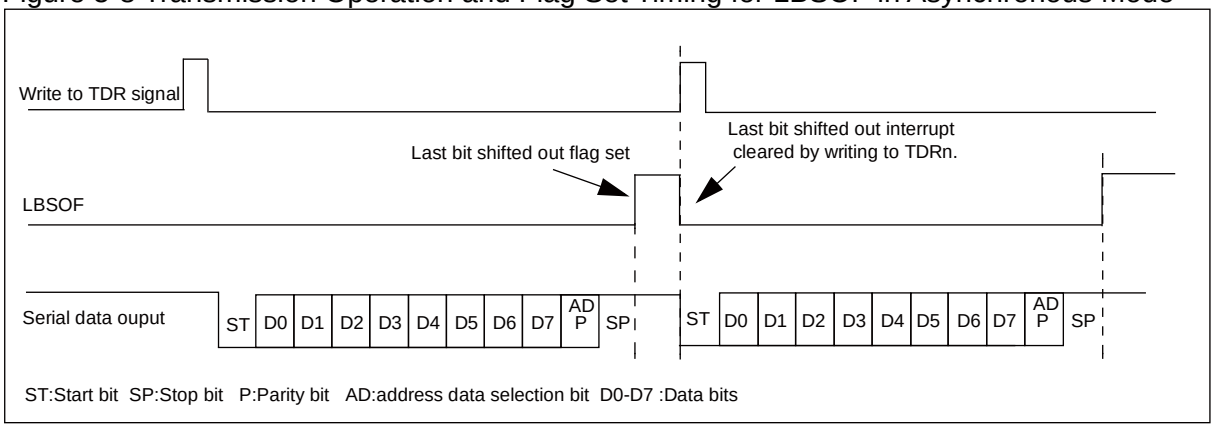
Note:

When transmission FIFO is enabled, the SSRn:TDRE flag will be asserted when the FIFO reaches the trigger level programmed in the transmission FIFO control register (TFCRn:TXFLC).

■ Transmission Interrupt Request Generation Timing using LBSOIE

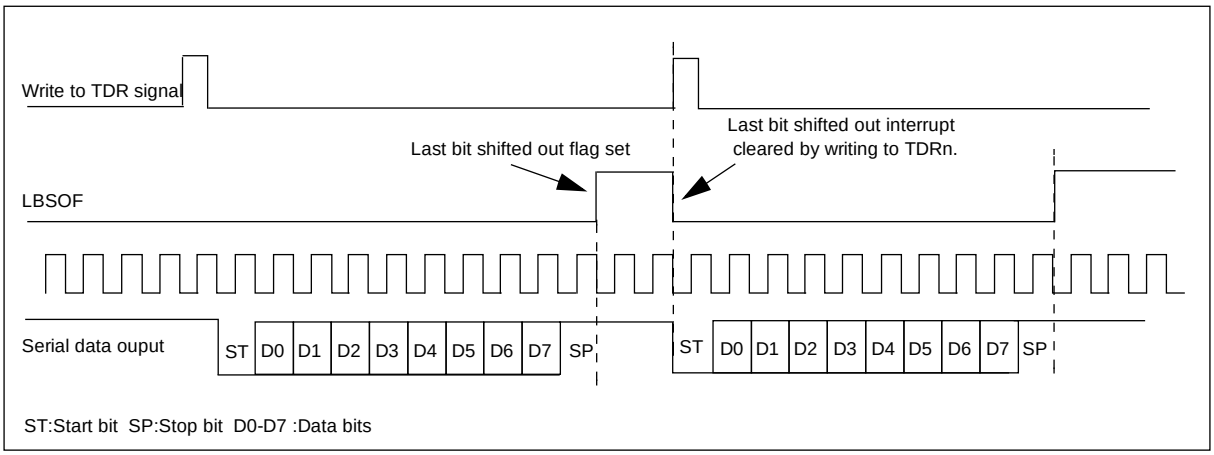
If the ESRn:LBSOF flag is set to "1" when the Last bit shifted out interrupt is enabled (EIERn:LBSOIE=1), transmission interrupt request is generated. Below figure shows the timing for asynchronous mode with one stop bit. ESRn:LBSOF is set at the end of the bit time of the last transferred bit.

Figure 5-8 Transmission Operation and Flag Set Timing for LBSOF in Asynchronous Mode



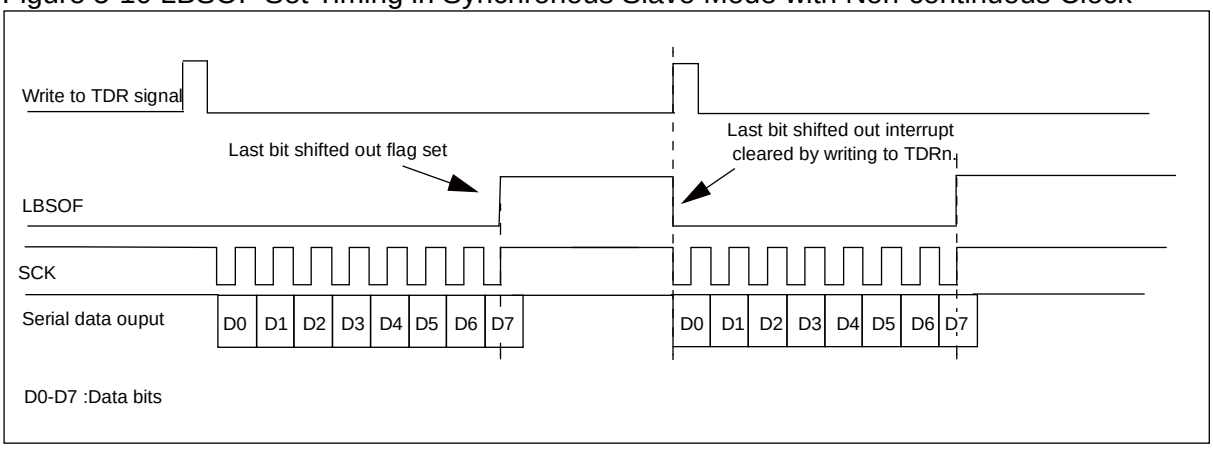
Below figure shows the flag set timing in synchronous master mode and synchronous slave mode with continues clock (ESCRn:CCO="1"). ESRn:LBSOF is set at the end of the bit time of the last transferred bit.

Figure 5-9 LBSOF Set Timing in Synchronous Master Mode and Synchronous Slave Mode with Continuous Clock



Below figure shows the flag set timing in synchronous slave mode without continues clock (ESCRn:CCO="0"). ESRn:LBSOF is set after half of the bit time of the last transferred bit.

Figure 5-10 LBSOF Set Timing in Synchronous Slave Mode with Non-continuous Clock



6. Baud Rates

One of the following can be selected for the USART serial clock source:
- Dedicated baud rate generator (Reload Counter)
- External clock as it is (clock input to the SCKn pin)
- External clock connected to the baud rate generator (Reload Counter)

■ USART Baud Rate Selection

The baud rate selection circuit is designed as shown below. One of the following three types of baud rates can be selected:

● Baud Rates Determined using the Dedicated Baud Rate Generator (Reload counter)

USART has two independent internal reload counters for transmission and reception serial clock. The baud rate can be selected via the 16-bit reload value determined by the Baud Rate Generator Registers (BGRn). The reload counter divides the peripheral clock CLKP1 by the value set in the Baud Rate Generator Registers.

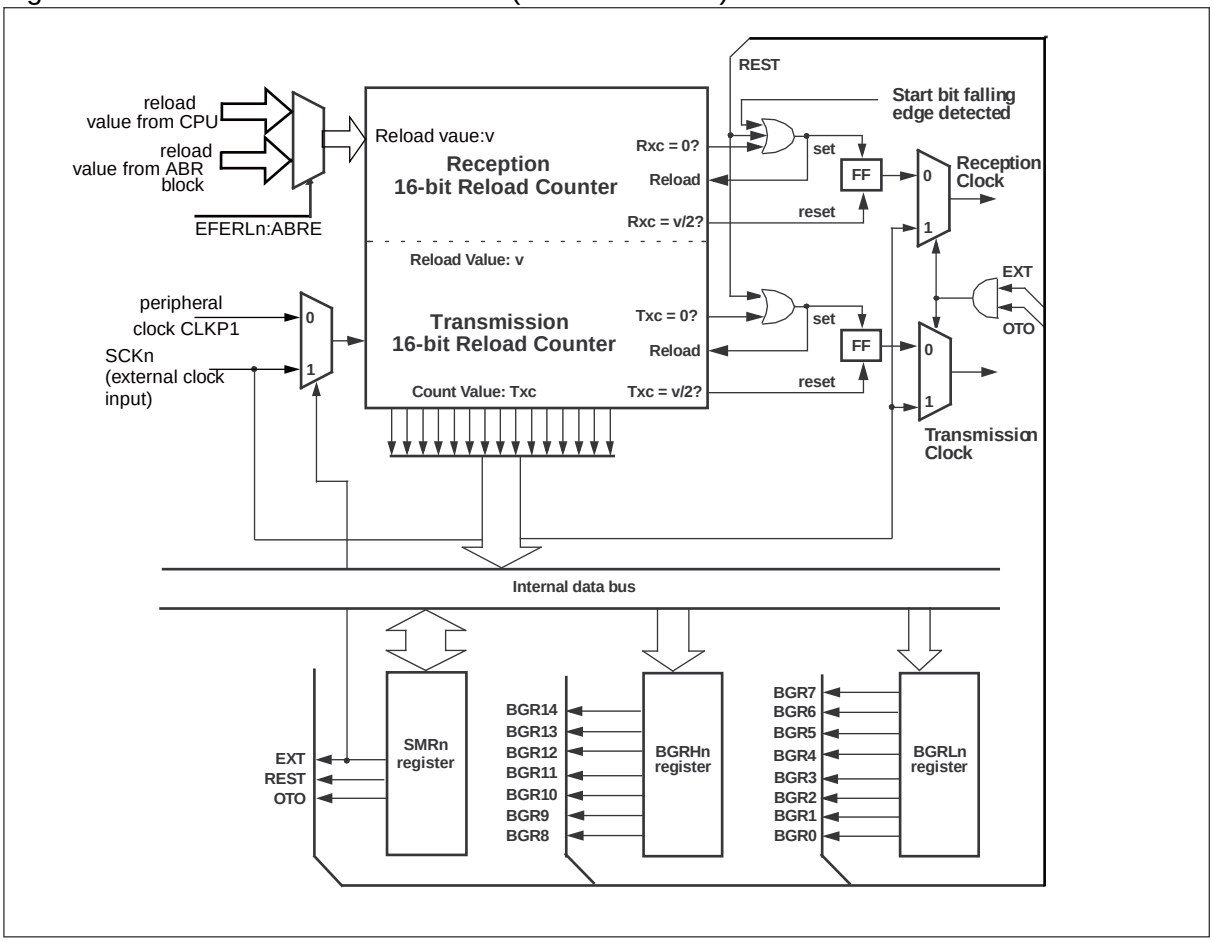
● Baud Rates Determined using External Clock (One-to-one mode)

The clock input from USART clock pulse input pins (SCK) is used as it is (synchronous). Any baud rate less than the peripheral clock (CLKP1) divided by 4 and is divisible can be set externally.

● Baud Rates Determined using the Dedicated Baud Rate Generator with External Clock

An external clock source can also be connected internally to the reload counter. In this mode it is used instead of the internal peripheral clock CLKP1. This was designed to use quartz oscillators with special frequencies and having the possibility to divide them.

Figure 6-1 Baud Rate Selection Circuit (Reload counter)



6.1. Setting the Baud Rate

This section describes how the baud rates are set and the resulting serial clock frequency is calculated.

■ Calculating the Baud Rate

The both 16-bit reload counters are programmed by the baud rate generator registers (BGRn). The following calculation formula should be used to set the desired baud rate:

Reload Value:

$v = \mathbf{[\Phi / b] - 1}$,

where **Φ** is the peripheral clock CLKP1, b the baud rate and [] gaussian brackets (mathematical rounding function).

● Example of Calculation

If the peripheral clock CLKP1 is 16 MHz and the desired baud rate is 19200 bps baud then the reload value v is:

$v = \mathbf{[16 \times 10^6 / 19200] - 1 = 832}$

The exact baud rate can then be recalculated: $b_{exact} = \Phi / (v + 1)$, here it is: $16 \times 10^6 / 833 = 19207.6831$

Note:

Setting the reload value to 0 stops the reload counter. For this reason the minimum division ratio is 2. For asynchronous communication, the reload value must be greater than equal to 4 because 5 times over-sampling is performed internally.

■ Suggested Division Ratios for Different Peripheral Clock CLKP1 Frequencies and Baud Rates

The following settings are suggested for different peripheral clock CLKP1 frequencies and baud rates:

Table 6-1 Suggested Baud Rates and Reload Values at Different Peripheral Clock CLKP1 Frequencies

Baud rate	16 MHz		24 MHz		32 MHz		48 MHz		56 MHz	
	value	dev.	value	dev.	value	dev.	value	dev.	value	dev.
4M	-	-	5	0	7	0	11	0	13	0
2M	7	0	11	0	15	0	23	0	27	0
1M	15	0	23	0	31	0	47	0	55	0
500000	31	0	47	0	63	0	95	0	111	0
460800	-	-	51	-0.16	68	-0.6	103	-0.16	120	-0.43
250000	63	0	95	0	127	0	191	0	223	0
230400	-	-	103	-0.16	137	-0.6	207	-0.16	242	-0.02
153600	103	-0.16	155	-0.16	207	-0.16	311	-0.16	363	-0.16
125000	127	0	191	0	255	0	383	0	447	0
115200	138	0.08	207	-0.16	276	-0.2	415	-0.16	485	-0.02
76800	207	-0.16	311	-0.16	-	-	624	0	728	-0.02
57600	277	0.08	416	0.08	554	-0.1	832	-0.04	971	-0.02
38400	416	0.08	624	0	832	-0.04	1249	0	1457	-0.02
28800	554	-0.01	832	-0.03	1110	-0.01	1665	-0.04	1943	-0.02
19200	832	-0.03	1249	0	1665	-0.04	2499	0	2915	-0.02
10417	1535	<0.01	2303	<0.01	3070	-0.02	4606	-0.01	5374	-0.015
9600	1666	0.02	2499	0	3332	-0.01	4999	0	5832	-0.005
7200	2221	<0.01	3332	<0.01	4443	-0.01	6665	-0.01	7776	-0.01
4800	3332	<0.01	4999	0	6665	-0.01	9999	0	11665	-0.005
2400	6666	<0.01	9999	0	13332	-0.002	19999	0	23332	-0.001
1200	13332	<0.01	19999	0	26665	-0.002	39999	0	46665	-0.001
600	26666	<0.01		-	53332	-0.0006		-		-
300	-	-	-	-	-	-	-	-	-	-

Notes:

- Deviations are given in%.
- Maximum Synchronous Baud Rate: Peripheral clock CLKP1 frequency divided by 5.

■ Using External Clock

If the EXT bit of the SMRn register is set, an external clock is selected, which has to be connected to the SCKn pin. The external clock is used in the same way as the peripheral clock CLKP1 to the baud rate reload counter.

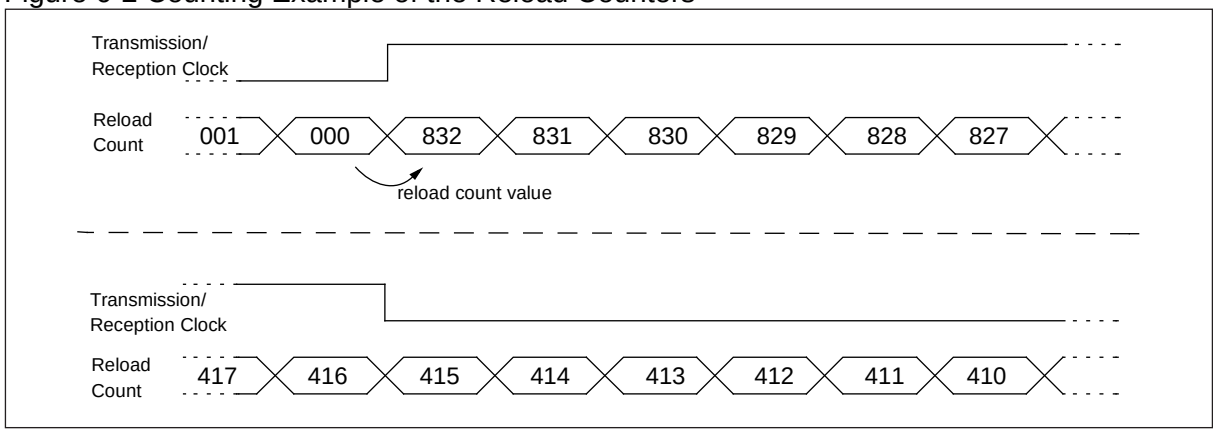
If One-to-one External Clock Input Mode (SMRn:OTO) is selected the SCKn signal is directly connected to the USART serial clock inputs. This is needed for the USART synchronous mode 2 operating as slave device.

Note, that in any case the resulting clock signal is synchronized to the peripheral clock CLKP1 in the USART module. This means that indivisible clock rates will result in phase unstable signals.

■ Counting Example

Assume the reload value is 832. The Figure 6-2 demonstrates the behavior of both Reload Counters:

Figure 6-2 Counting Example of the Reload Counters



Note:

The falling edge of the Serial Clock Signal always occurs after | (v + 1) / 2 |.
where v is the reload value.

6.2. Restarting the Reload Counter

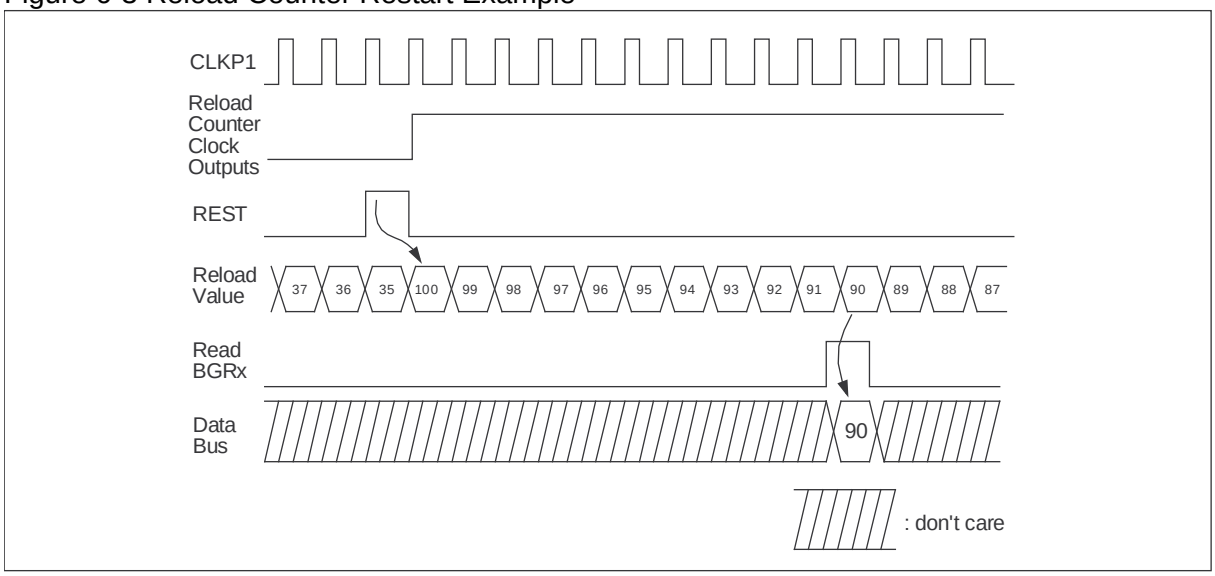
The Reload Counters can be restarted of the following reasons:
Transmission and reception reload counter:
- Global MCU reset
- USART programmable clear (SMRn:UPCL bit)
- User programmable restart (SMRn:REST bit)

Reception reload counter:
- Start bit falling edge detection in asynchronous mode

■ Programmable Restart

If the REST bit of the Serial Mode Register (SMRn) is set by the user, both Reload Counters are restarted at the next clock cycle. This feature is intended to use the Transmission Reload Counter as a small timer.
The following figure illustrates a possible usage of this feature (assume that the reload value is 100.)

Figure 6-3 Reload Counter Restart Example



In this example the number of peripheral clock CLKP1 cycles (cyc) after REST is then:

cyc = v - c + 1 = 100 - 90 + 1 = 11,

where v is the reload value and c is the read counter value.

Note:

If USART is reset by setting SMRn:UPCL, the Reload Counters will restart too.

· Automatic restart
In asynchronous USART mode if a falling edge of a start bit is detected the Reception Reload Counter is restarted. This is intended to synchronize the serial input shifter to the incoming serial data stream.

● Clearing Reload Counters

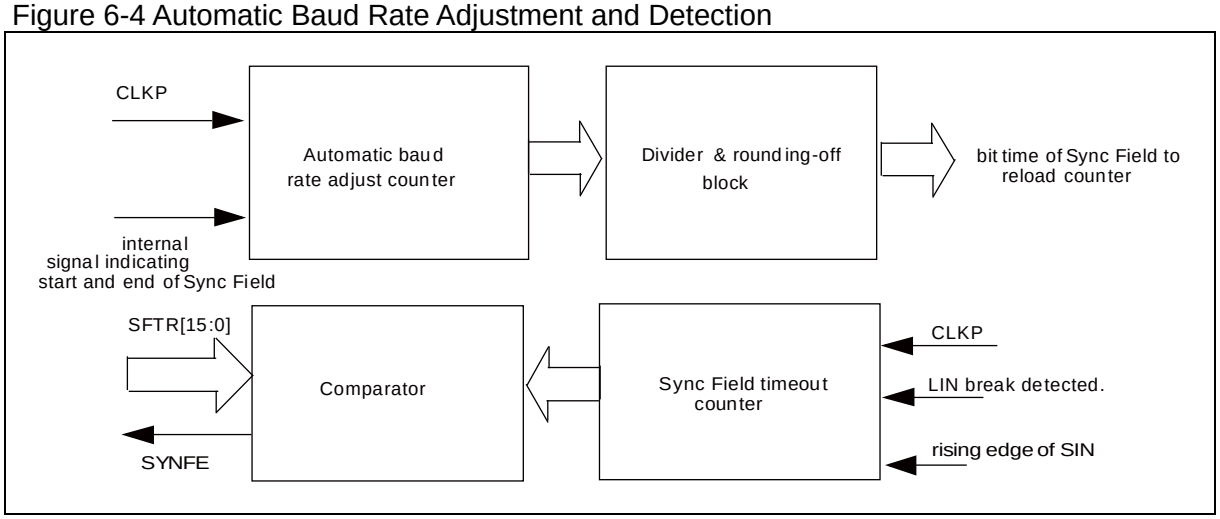
The baud rate reload/counter register (BGRn) and the baud rate reload counters are cleared to "0" by the MCU global reset and the counters stops. The reload counters are cleared to "0" by writing "1" to the UPCL bit in the SMRn register. However the value stored in the reload register is kept unchanged and the counters start from reload value immediately. Writing "0" to the REST bit does not clear the counters and they restart from reload value immediately.

6.3. Automatic Baud Rate Detection and Adjustment

Automatic baud rate detection and adjustment calculates bit time with the help of sync field and the calculated value is loaded into the reload counters for generation of transmission and reception clock

■ Automatic Baud Rate Detection and Adjustment

Figure 6-4 Automatic Baud Rate Adjustment and Detection



Automatic baud rate adjustment and detection circuitry consist of a 16 bit counter which calculates the time elapsed between first and fifth falling edge of sync field. The calculated value is divided by 8 to get the bit time of sync field. The bit time of sync field is rounded-off by incrementing the value if the remainder is greater than or equal to 0.5 times of divider (8). The 16-bit reload value of the baud rate counter is loaded into reload counter for generation of reception clock and transmission clock.
The automatic baud rate detection and adjustment circuitry also consist of 16 bit sync field timeout counter. This counter starts counting at rising edge of SIN after the LIN-break gets detected. When the 16 bit value of sync field timeout counter reaches the value programmed in the SFTRn (Sync field timeout register) before the fifth falling edge of sync field, ESRn:SYNFE is set and reception interrupt is asserted when the EIERn:SYNFEIE is set to "1".

7. Registers

This section explains the configuration and functions of the registers used for the USART.

■ List of USART Registers

Abbreviated Register Name	Register Name	Reference
SCRn	Serial Control Register	See 7.1
SMRn	Serial Mode Register	See 7.2
SSRn	Serial Status Register	See 7.3
RDRn/TDRn	Reception and Transmission Data Register	See 7.4
ESCRn	Extended Status/Control Register	See 7.5
ECCRn	Extended Communication Control Register	See 7.6
BGRn	Baud Rate Generation / Reload Counter Register	See 7.7
ESIRn	Extended Serial Interrupt Register	See 7.8
TFCRn	Transmission FIFO Control Register	See 7.9
RFCRn	Reception FIFO Control Register	See 7.10
TFSRn	Transmission FIFO Status Register	See 7.11
RFSRn	Reception FIFO Status Register	See 7.12
SFTRn	Sync Field Time-out Register	See 7.13
CSCRn	Checksum Status and Control Register	See 7.14
FIDRn	Frame-ID Data Register	See 7.15
EFERLn	Extended Feature Enable Register	See 7.16
EFERHn	Extended Feature enable register	See 7.17
EIERn	Extended Interrupt Enable Register	See 7.18
ESRn	Extended Status Register	See 7.19

7.1. Serial Control Register (SCRn)

This register specifies parity bits, selects the stop bit and data lengths, selects a frame data format in mode 1, clears the reception error flags, and specifies whether to enable transmission and reception.

■ Serial Control Register (SCRn)

SCRn								
bit	15	14	13	12	11	10	9	8
	PEN	P	SBL	CL	AD	CRE	RXE	TXE
Attribute	R/W	R/W	R/W	R/W	R/W	W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

n = 0...9 according to device (Please refer to the datasheet of the corresponding device)

[bit15] PEN: Parity Enable Bit

Bit	Description
0	Parity disabled
1	Parity enabled

This bit selects whether to add a parity bit during transmission or detect it during reception.
Parity is provided in mode 0 and in mode 2 if SSM of the ECCRn is selected. This bit is fixed to 0 (no parity) in mode 1 (multiprocessor) and mode 3 (LIN mode).

[bit14] P: Parity Selection Bit

Bit	Description
0	Even Parity enabled
1	Odd Parity enabled

When parity is provided and enabled this bit selects even (0) or odd (1) parity.

[bit13] SBL: Stop Bit Length Selection Bit

Bit	Description
0	1 stop bit
1	2 stop bits

This bit selects the length of the stop bit of an asynchronous data frame or a synchronous frame if SSM of the ECCRn is selected. This bit is fixed to 0 (1 stop bit) in mode 3 (LIN).

[bit12] CL: Data Length Selection Bit

Bit	Description
0	7bits
1	8bits

This bit specifies the length of transmission or reception data. This bit is fixed to 1 (8 bits) in mode 2 and 3.

[bit11] AD: Address/Data Selection Bit

Bit	Description
0	Data bit
1	Address bit

This bit specifies the data format in multiprocessor mode 1. Writing to this bit is provided for a master CPU, reading from it for slave CPU. A 1 indicates an address frame, a 0 indicates a usual data frame.

Note:

Please read the hints about using this bit in "8. Notes on Using".

[bit10] CRE: Clear Reception Error Flags Bit

Bit	Description	
	Read	Write
0	Read always returns zero	No effect
1		Clear all reception errors (PE, FRE, ORE)

This bit clears the FRE, ORE and PE flag of the Serial Status Register (SSRn). It can also reset any ongoing reception depending on EFER:RSTRFM.(Default is reset)

[bit9] RXE: Reception Enable Bit

Bit	Description
0	Disable Reception
1	Enable Reception

This bit enables/disables LIN-USART reception.
The LIN synch break detection in mode 3 remains unaffected.

Notes:

- If reception is disabled (RXE = 0) during receiving, it is stopped immediately. In this case, data is not guaranteed.
- If EFERHn.INTLBEN =1, SCRn: RXE is set automatically (EFERHn.INTLBEN =1 then read value of SCRn: RXE =1).
- If EFERHn.ENRXHR(Automatic header reception) = 1 and frame-id reception(When the LIN-USART is receiving the frame id in LIN mode) the RXE is set automatically for the entire duration of frame id reception (In mode 3 with EFERHn.ENRXHR =1 the read value of SCRn: RXE =1 during the frame id reception phase).

[bit8] TXE: Transmission Enable Bit

Bit	Description
0	Disable Transmission
1	Enable Transmission

This bit enables/disables LIN-USART transmission.

Note:

If transmission is disabled (TXE = 0) during transmitting, it is stopped immediately. In this case, data is not guaranteed.

7.2. Serial Mode Register (SMRn)

This register selects an operation mode and baud rate clock and specifies whether to enable output of serial data and clocks to the corresponding pin.

■ Serial Mode Register (SMRn)

SMRn								
bit	7	6	5	4	3	2	1	0
	MD1	MD0	OTO	EXT	REST	UPCL	SCKE	SOE
Attribute	R/W	R/W	R/W	R/W	W	W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

n = 0...9 according to device (Please refer to the datasheet of the corresponding device)

[bit7, bit6] MD1 and MD0: Operation Mode Selection Bits

bit7	bit6	Description
0	0	Mode 0: Asynchronous normal
0	1	Mode 1: Asynchronous Multiprocessor
1	0	Mode 2: Synchronous
1	1	Mode 3: Asynchronous LIN

These two bits set the USART operation mode.

Note:

Changing the mode bit MD1, MD0 causes any ongoing reception or transmission to be aborted.

[bit5] OTO: One-to-one External Clock Selection Bit

Bit	Description
0	Use ext. Clock with Baud Rate Generator (Reload C.)
1	Use external Clock as is

This bit sets an external clock directly to the LIN-USART's serial clock. This function is used for operating mode 2 (synchronous) slave mode operation.

[bit4] EXT: External Clock Selection Bit

Bit	Description
0	Use internal Baud Rate Generator (Reload Counter)
1	Use external Serial Clock Source

This bit executes internal or external clock source for the reload counter

[bit3] REST: Restart of Transmission Reload Counter Bit

Bit	Description	
	Read	Write
0	Always 0	No effect
1		Restart Counter

If a 1 is written to this bit the reload counter is restarted. Writing 0 to it has no effect. Reading from this bit always returns "0".

[bit2] UPCL: USART Programmable Clear Bit (Software reset)

Bit	Description	
	Read	Write
0	Always 0	No effect
1		Reset LIN-USART

Writing a 1 to this bit resets LIN-USART immediately. The register settings are preserved. Possible reception or transmission will cut off.
This bit also resets TX/RX FIFO block, checksum generation/verification logic, header detection state machine and sync field detection timeout counter.
All flags (TDRE, RDRF, LBD, PE, ORE, FRE) are cleared and the Reception Data Register (RDRn) contains 00_H.
Writing 0 to this bit has no effect.
Reading from it always returns 0.
LIN-USART reset should be performed after disabling the reception and transmission and its interrupt enable bits.

[bit1] SCKE: Serial Clock Output Enable

Bit	Description
0	Serial clock output pin of LIN-USART disabled or LIN-USART clock input pin
1	Serial clock output pin of LIN-USART enabled

This bit controls the serial clock I/O ports.
When this bit is 0, SCKn pin operates as general purpose I/O port or serial clock input pin. When this bit is 1 and no other peripheral resource uses this pin as output, the pin operates as serial clock output pin and outputs clock in operating mode 2 (synchronous).

Note:

When using SCKn pin as serial clock input (SCKE = 0) pin, set the corresponding bit of DDR as input port and set the corresponding bit of PIER = 1. Also, select external clock (EXT = 1) using the external clock selection bit.

Reference:

When the SCKn pin is assigned to serial clock output (SCKE = 1) and no other peripheral resource uses this pin as output, it functions as the serial clock output pin (SCKn) regardless of the status of general input-output ports.

[bit0] SOE: Serial Data Output Enable Bit

Bit	Description
0	LIN-UART serial data output pin disabled
1	LIN-UART serial data output pin enabled

This bit enables or disables the output of serial data.
When this bit is 0, SOTn pin operates as general purpose I/O pin. When this bit is 1and no other peripheral resource uses this pin as output, SOTn pin operates as serial data output pin.

Reference:

When the output of serial data is enabled (SOE = 1) and no other peripheral resource uses this pin as output, SOTn pin functions as serial data output pin (SOTn) regardless of the status of general input-output ports.

7.3. Serial Status Register (SSRn)

This register shows the transmission and reception status. It also enables and disables the transmission and reception interrupts.

■ Serial Status Register (SSRn)

SSRn								
bit	15	14	13	12	11	10	9	8
	PE	ORE	FRE	RDRF	TDRE	BDS	RIE	TIE
Attribute	R	R	R	R	R	R/W	R/W	R/W
Initial value	0	0	0	0	1	0	0	0

n = 0...9 according to device (Please refer to the datasheet of the corresponding device)

[bit15] PE: Parity Error Flag Bit

Bit	Description
0	No parity error occurred
1	A parity error occurred during reception

- This bit is set to 1 when a parity error occurs during reception.
- It is cleared when 1 is written to the CRE bit of the serial control register (SCRn).
- A reception interrupt request is output when this bit and the RIE bit are 1.
- Data in the reception data register (RDRn) is invalid when this flag is set.
- If RX FIFO is enabled (RFCRn:RXFE), PE remains 1 till it is cleared by software.

[bit14] ORE: Overrun Error Flag Bit

Bit	Description
0	No overrun error occurred
1	An overrun error occurred during reception

- This bit is set to 1 when an overrun error occurs during reception.
- It is cleared when 1 is written to the CRE bit of the serial control register (SCRn).
- A reception interrupt request is output when this bit and the RIE bit are 1.
- Data in the reception data register (RDRn) is invalid when this flag is set.
- If RX FIFO is enabled (RFCRn:RXFE = 1), ORE is set when the reception FIFO is full and another data byte received.

[bit13] FRE: Framing Error Flag Bit

Bit	Description
0	No framing error occurred
1	A framing error occurred during reception

- This bit is set to 1 when a framing error occurs during reception.
- It is cleared when 1 is written to the CRE bit of the serial control register (SCRn).
- A reception interrupt request is output when this bit and the RIE bit are 1.
- Data in the reception data register (RDRn) is invalid when this flag is set.
- If RX FIFO is enabled (RFCRn:RXFE), FRE remains 1 till it is cleared by software.

[bit12] RDRF: Receive Data Full Flag Bit

Bit	Description
0	Reception data register is empty
1	Reception data register is full

- This flag indicates the status of the reception data register (RDRn).
- This bit is set to 1 when reception data is loaded into RDRn.
- It is cleared when the reception data register (RDRn) is read.
- A reception interrupt request is output when this bit and the RIE bit are 1.
- When RX FIFO is enabled (EFERn:RXFE = 1), this bit is set to 1 when the number of data in the RX FIFO reaches the trigger level programmed in reception FIFO control register (RFCRn:RXFLC[4:0]).

Note:

ESIRn:RDRF has the same behavior as this bit, but it is not cleared when the reception data register (RDRn) is read.

[bit11] TDRE: Transmission Data Empty Flag Bit

Bit	Description
0	Transmission data register is full
1	Transmission data register is empty

- This flag indicates the status of the transmission data register (TDRn).
- This bit is cleared to 0 when transmission data is written to TDRn and is set to 1 when data is loaded into the transmission shift register and transmission starts.
- A transmission interrupt request is generated if both this bit and the TIE bit are 1.
- If the LBR bit in the ECCRn register is set to "1" while the TDRE bit is "1", then this bit once changes to "0". After the completion of LIN synch break generator, the TDRE bit changes back to "1".
- When TX FIFO is enabled (EFER:TXFE = 1), the TDRE flag is asserted when the number of data in the TX FIFO reaches the programmed trigger level in transmission FIFO control register (TFCRn:TXFLC[4:0]).

Notes:

- This bit is set to 1 (TDRn empty) as its initial value.
- ESIRn:TDRE has the same behavior as this bit, but it is not cleared when transmission data is written to TDRn.

[bit10] BDS: Transfer Direction Selection Bit

Bit	Description
0	Send / receive LSB first
1	Send / receive MSB first

- This bit selects whether to transfer serial data from the least significant bit (LSB first, BDS = 0) or the most significant bit (MSB first, BDS = 1).

Notes:

- The high-order and low-order sides of serial data are interchanged with each other during reading from or writing to the serial data register. If this bit is set to another value after the data is written to the RDRn register, the data becomes invalid.
- This bit is fixed to 0 in mode 3 (LIN).

[bit9] RIE: Receive Interrupt Request Enable Bit

Bit	Description
0	Disables Reception Interrupt
1	Enables Reception Interrupt

This bit enables/disables the reception interrupt. If any of the RDRF, PE, ORE, FRE bits is set and this bit is "1", then a reception interrupt is signaled to the interrupt controller.

[bit8] TIE: Transmission Interrupt Request Enable Bit

Bit	Description
0	Disables Transmission Interrupt
1	Enables Transmission Interrupt

- This bit enables or disables the transmission interrupt.
- A transmission interrupt request is output when this bit and the TDRE bit are "1".
- This interrupt can be used as replacement of last bit shifted out interrupt (EIERn:LBSOIE). Only one of these interrupts should be enabled at the same time.

7.4. Reception and Transmission Data Register (RDRn/TDRn)

The reception data register (RDRn) holds the received data. The transmission data register (TDRn) holds the transmission data. Both RDRn and TDRn registers are located at the same address.

■ Reception and Transmission Data Registers (RDRn/TDRn)

RDRn								
bit	7	6	5	4	3	2	1	0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

TDRn								
bit	7	6	5	4	3	2	1	0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

n = 0...9 according to device (Please refer to the datasheet of the corresponding device)

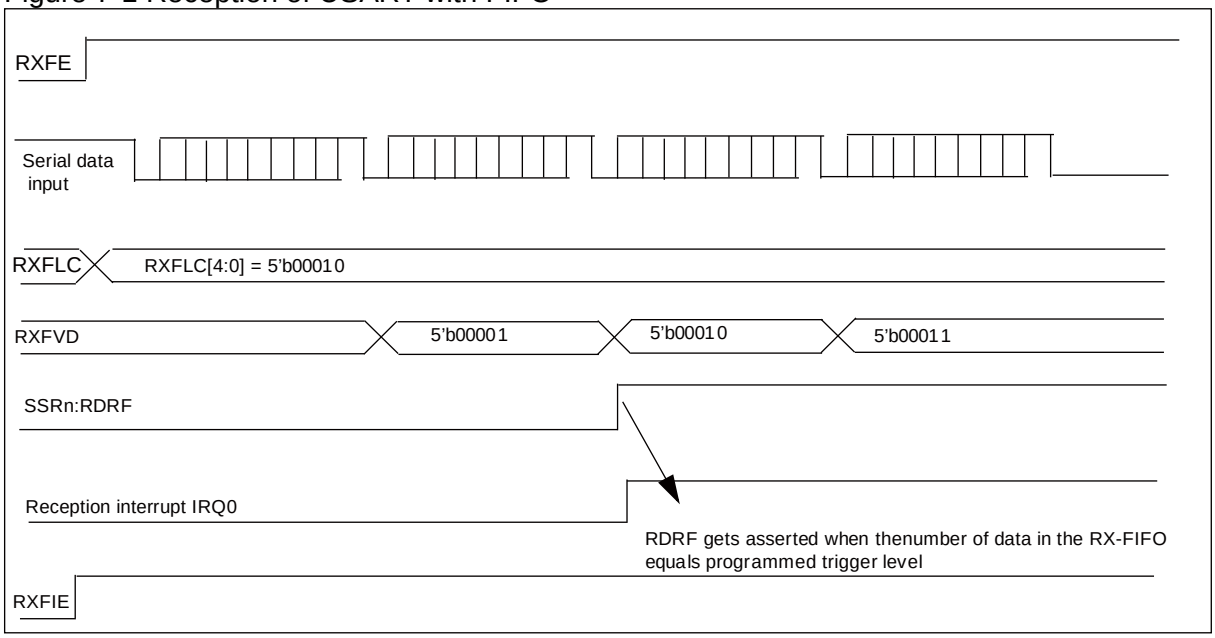
[bit7 to bit0]

Bit	Description
Read	Read from Reception Data Register
Write	Write to Transmission Data Register

● Reception:

RDRn is the register that contains reception data. The serial data signal transmitted to the SINn pin is converted in the shift register and stored there. When the data length is 7 bits, the uppermost bit (D7) contains 0. When reception is complete the data is stored in this register and the reception data full flag bit (SSRn:RDRF) is set to 1. If a receive interrupt request is enabled at this point, a receive interrupt occurs.
Read RDRn when the RDRF bit of the status register (SSRn) is 1. The RDRF bit is cleared automatically to 0 when RDRn is read. Also the receive interrupt is cleared if it is enabled and no error has occurred.
Data in RDRn is invalid when a reception error occurs (SSRn: PE, ORE, or FRE = 1).
If the RX FIFO is enabled (RFCRn:RXFE = 1), RDRn contains the next value of the RX FIFO to be read. SSRn:RDRF is set when the number of data in the FIFO is greater or equal to the programmed trigger level in the RFCRn:RXFLC[4:0].
RDRn and RX-FIFO-content is reset to 00000000_H at reset and SMRn:UPCL = 1.

Figure 7-1 Reception of USART with FIFO



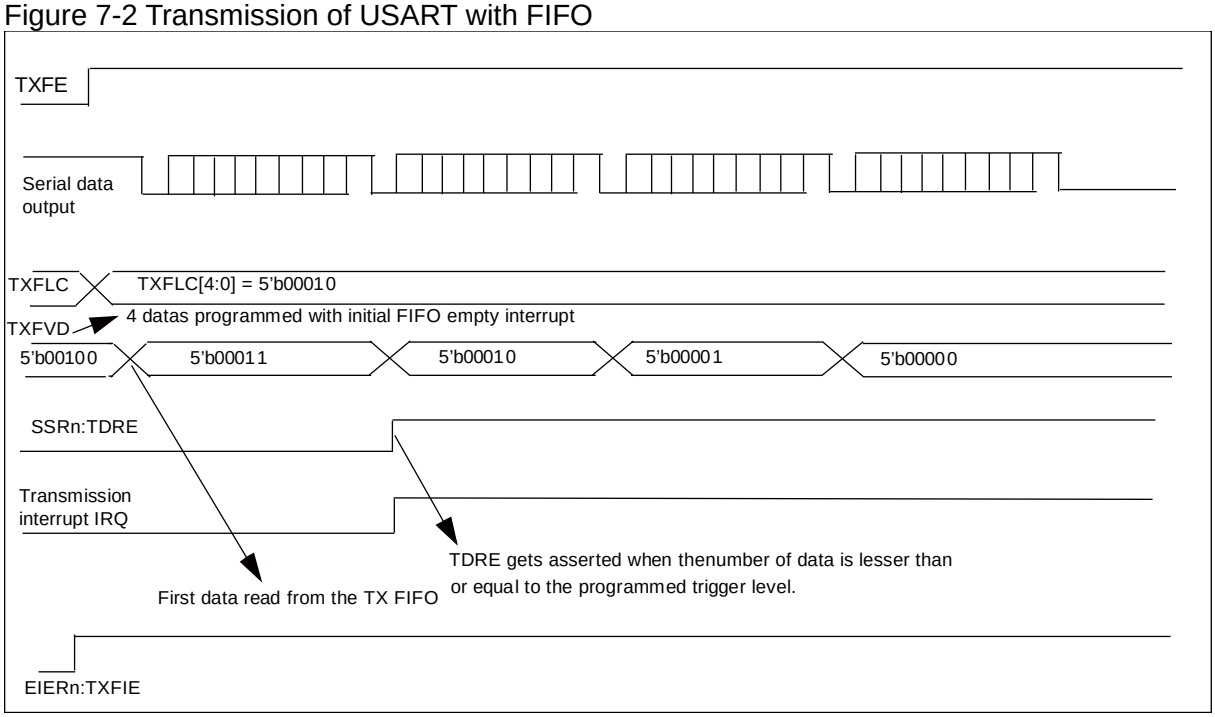
- **Transmission:**

  When transmission data is written to this register, the transmission data empty flag bit (SSRn:TDRE) is cleared to 0. When transfer to the transmission shift register is complete and starts, the bit is set to 1. When the TDRE bit is 1, the next part of transmission data can be written. If output transmission interrupt requests have been enabled, a transmission interrupt is generated. Write the next part of transmission data when a transmission interrupt is generated or the TDRE bit is 1.

  If the TX FIFO is enabled (TFCRn:TXFE = 1), SSRn:TDRE is set when the number of data in the TX FIFO is less or equal to programmed trigger level in the TFCRn:TXFLC[4:0].

  TDRn and TX-FIFO-content is reset to 11111111_H at reset.

Figure 7-2 Transmission of USART with FIFO



Note:

TDRn is a write-only register and RDRn is a read-only register. These registers are located in the same address, so the read value is different from the write value. Therefore, instructions that perform a read-modify-write (RMW) operation, such as the INC/DEC instruction, cannot be used.

7.5. Extended Status/Control Register (ESCRn)

This register provides several LIN functions, direct access to the SINn and SOTn pin and setting for USART synchronous clock mode.

■ Extended Status/Control Register (ESCRn)

ESCRn								
bit	15	14	13	12	11	10	9	8
	LBIE	LBD	LBL1	LBL0	SOPE	SIOP	CCO	SCES
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	1	0	0

n = 0...9 according to device (Please refer to the datasheet of the corresponding device)

[bit15] LBIE: LIN Synch Break Detection Interrupt Enable Bit

Bit	Description
0	LIN synch break interrupt disable
1	LIN synch break interrupt enable

- This bit enables/disables LIN synch break interrupt
- LIN synch break interrupt is connected to the receive interrupt. When in mode 3 the LBD bit is set and this bit is "1", a receive interrupt is signaled to the interrupt controller. This bit is fixed to "0" in operation mode 0, mode 1 and 2..

[bit14] LBD: LIN Synch Break Detected Flag

Bit	Description	
	Read	Write
0	No LIN synch break detected	Clear LIN synch break detected flag
1	LIN synch break detected	No effect

- This bit goes 1 if a LIN synch break was detected in operating mode 3. Writing a 0 to it clears this bit and the corresponding interrupt, if it is enabled.
- It is recommended to write "0" to the RXE bit in the SCRn register before using this bit.
- Read-modify-write instructions always return 1. Note that this does not indicate a LIN synch break.

[bit13, bit12] LBL1/0: LIN Synch Break Length Selection

- These two bits determine how many serial bit times the LIN synch break is generated by USART. Receiving a LIN synch break is always fixed to 11 bit times.
- During transmission the LIN break length can be varied from 13 to 20 bits times using these two bits and EFER:LBL2 bit. The table for various LIN break length configuration is given below.

LBL2	LBL1	LBL0	Description
0	0	0	13 bit times
0	0	1	14 bit times
0	1	0	15 bit times
0	1	1	16 bit times
1	0	0	17 bit times
1	0	1	18 bit times
1	1	0	19 bit times
1	1	1	20 bit times

[bit11] SOPE: Serial Output Pin Direct Access Enable*

Bit	Description
0	Serial Output pin direct access disable
1	Serial Output pin direct access enable

Setting this bit to 1 enables the direct write to the SOTn pin, if SMRn:SOE = 1. *

[bit10] SIOP: Serial Input/Output Pin Direct Access *

Bit	Description	
	Read	Write (if SOPE = "1")
0	Reading the actual value of SIN	SOT is forced to "0"
1		SOT is forced to "1"

- Normal read instructions always return the actual value of the SINn pin. Writing to it sets the bit value to the SOTn pin, if SOPE = 1.
- During a Read-Modify-Write instruction the bit returns the SOTn value in the read cycle. *

*: See Table 7-1 Description of the interaction of SOPE and SIOP

Table 7-1 Description of the Interaction of SOPE and SIOP

SOPE	SIOP	Reading from SIOP	Writing to SIOP
0	R/W	Returns current value of SINn	Has no effect on SOTn, but holds the written value
1	R/W	Returns current value of SINn	Write "0" or "1" to SOTn
1	RMW	Reads current value of SOTn and writes it back	

[bit9] CCO: Continuous Clock Output Enable Bit

Bit	Description
0	Continuous Clock Output disabled
1	Continuous Clock Output enabled

This bit enables a continuous serial clock at the SCKn pin if USART operates in master mode 2 (synchronous) and the SCKn pin is configured as a clock output.

Note:

When CCO bit is "1", use ECCRn:SSM bit as setting to "1".

[bit8] SCES: Serial Clock Edge Selection Bit

Bit	Description
0	Sampling on rising clock edge (normal)
1	Sampling on falling clock edge (inverted clock) When ECCRn:SCDE is set to "0"

- This bit inverts the serial clock signal in operation mode 2 (synchronous communication).
- Receiving data is sampled at the falling edge of the internal clock,when ECCRn:SCDE =0 . If ECCRn:MS bit is "0" (master mode) and SMRn:SCKE is "1" (clock output enabled), the output clock signal is also inverted.
- This bit is used in only in mode 2, Value is ignored in other modes. Read value is always "0" in other modes

Note:

When SCES bit is "1", it is prohibited to occur software reset. And please change this bit only when transferring and receiving is disabled.

7.6. Extended Communication Control Register (ECCRn)

The extended communication control register provides bus idle recognition interrupt settings, synchronous clock settings, and the LIN break generation.

■ Extended Communication Control Register (ECCRn)

ECCRn								
bit	7	6	5	4	3	2	1	0
	INV	LBR	MS	SCDE	SSM	RESV	RBI	TBI
Attribute	R/W	W	R/W	R/W	R/W	R/W	R	R
Initial value	0	0	0	0	0	0	X	X

n = 0...9 according to device (Please refer to the datasheet of the corresponding device)

[bit7] INV: Invert Serial Data

Bit	Description
0	Serial Data is not inverted
1	Serial Data is inverted

- This bit inverts the serial data at SINn and SOTn pin. SCKn is not affected (see ESCRn: SCES).
- RMW instructions do not affect this bit.

[bit6] LBR: Generating LIN Synch Break Bit

Bit	Description	
	Read	Write
0	Always read 0	No effect
1		Generate LIN synch break

Writing a "1" to this bit generates a LIN synch break of the length selected by ESCRn:LBL[1:0] and EFERn:LBL2 in mod3 and mod0.Read value is always "0".

[bit5] MS: Master/Slave Mode Selection Bit

Bit	Description
0	Master mode (generating serial clock)
1	Slave mode (receiving external serial clock)

- This bit selects master or slave mode of USART in synchronous mode 2.
- If master mode is selected, USART generates the synchronous clock by itself.
- If slave mode is selected, USART receives external serial clock.
- This bit is fixed to "0" in operation mode 0, 1 and 3.

Note:

If slave mode is selected, the clock source must be external and set to "One-to-One" (SMRn:SCKE = 0, EXT = 1, OTO = 1).

[bit4] SCDE: Serial Clock Delay Enable Bit

Bit	Description
0	Disable clock delay
1	Enable clock delay

If this bit is set the serial output clock is delayed as shown in Figure 4-4 if USART operates in master mode 2, value is ignored in other modes. Read value is always "0" in other modes.

[bit3] SSM: Start/Stop Bit Mode Enable

Bit	Description
0	No start/stop bits in synchronous mode 2
1	Enable start/stop bits in synchronous mode 2

This bit adds start and stop bits to the synchronous data format in operation mode 2, value is ignored in other modes. Read value is always "0" in other modes.

[bit2] RESV: Reserved Bits

- Write always "0".
- Read value is undefined.
- Read modify write operations to this register have no effect on this bit

[bit1] RBI: Reception Bus Idle Flag Bit*

Bit	Description
0	Reception is ongoing
1	No reception activity

*: Not used in mode 2 when MS = "1"

- This bit is "1" if there is no reception activity on the SINn pin.
- It is "0" when reception activity is ongoing on the SINn pin.
- Do not use this bit in mode 2 when ECCRn:MS="1".

[bit0] TBI: Transmission Bus Idle Flag Bit*

Bit	Description
0	Transmission is ongoing
1	No transmission activity

*: Not used in mode 2 when MS = "1"

- This bit is "1" if there is no transmission activity on the SOTn pin.
- Do not use this bit in mode 2 when ECCRn:MS = "1".

7.7. Baud Rate Generation / Reload Counter Register (BGRn)

The baud rate/reload counter register sets the division ratio for the serial clock. The current counter value of the transmission reload counter or the reload value can be read.

■ Baud Rate Generation / Reload Register (BGRn)

BGRn

bit	15	14	13	12	11	10	9	8
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

bit	7	6	5	4	3	2	1	0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

n = 0...9 according to device (Please refer to the datasheet of the corresponding device)

[bit15 to bit8]

Bit	Description
Read	Read bit 15 to 8 of transmission reload counter
Write	Write bit 15 to 8 of reload value to counter

[bit7 to bit0]

Bit	Description
Read	Read bit 7 to 0 of transmission reload counter
Write	Write bit 7 to 0 of reload value to counter

■ Baud Rate Generation / Reload Counter Register

The baud rate/reload counter registers determine the division ratio for the serial clock.
If the Automatic baud rate detection is enabled, the value written to BGR is overwritten when the baud rate has been detected by the module.
Both registers can be read or written via byte or word access.
Read behavior depends on EFERn:BRGR.

7.8. Extended Serial Interrupt Register (ESIRn)

The extended serial interrupt register contains control bits to change the interrupt handling of the USART for improved interrupt handling and to enable DMA to handle USART data transfers.

■ Extended Serial Interrupt Register (ESIRn)

ESIRn								
bit	7	6	5	4	3	2	1	0
	-	-	-	-	TDRE	RDRF	RBI	AICD
Attribute	-	-	-	-	R/W	R/W	R/W	R/W
Initial value	X	X	X	X	1	0	X	0

n = 0...9 according to device (Please refer to the datasheet of the corresponding device)

[bit7 to bit4] -:Undefined

Read value is undefined, always write 0.
Read modify write operations to this register have no effect on this bit

[bit3] TDRE: Transmission Data Register Empty (Sticky behavior)

Bit	Description
0	Transmission data register is full
1	Transmission data register is empty

- This flag has the same function as SSRn:TDRE but it is not cleared when data is written to the transmission data register TDRn.
- This flag indicates the status of the transmission data register (TDRn).
- This bit is set to 1 when data is loaded into the transmission shift register and transmission starts.
- This bit must be cleared by writing "0".
- When AICD = 0 and SSRn:TIE = 1, then SSRn:TDRE is used to generate a transmission interrupt.
- When AICD = 1 and SSRn:TIE = 1, then ESIRn:TDRE is used to generate a transmission interrupt.

Note:

This bit is set to 1 (TDR empty) as its initial value.

[bit2] RDRF: Reception Data Register Full (Sticky behavior)

Bit	Description
0	Reception data register is empty
1	Reception data register is full

- This flag has the same function as SSRn:RDRF but it is not cleared when data is read from the reception data register RDRn.
- This flag indicates the status of the reception data register (RDRn).
- This bit is set to 1 when reception data is loaded into RDRn.

- This bit must be cleared by writing "0".
- When AICD = 0 and SSRn:RIE = 1, then SSRn:RDRF is used to generate a reception interrupt.
- When AICD = 1 and SSRn:RIE = 1, then ESIRn:RDRF is used to generate a reception interrupt.

[bit1] RBI: Reception Bus Idle (Sticky behavior)

Bit	Description
0	Reception is ongoing or is finished
1	No reception activity since last clear

- This flag has the same function as ECCRn:RBI but it is not cleared when the reception bus is no longer idle.
- When there is no reception activity ongoing, this bit is set to "1".
- It must be cleared by writing "0".
- When AICD = 0 and ECCRn:BIE = 1, then ECCRn:RBI is used to generate a reception interrupt.
- When AICD = 1 and ECCRn:BIE = 1, then ESIRn:RBI is used to generate a reception interrupt.

Note:

When an interrupt shall be generated by a bus idle condition, set AICD = 1. When AICD = 0, ECCRn:RBI would be used for interrupt generation. As soon as the bus is no longer idle, ECCRn:RBI is "0" and the interrupt may be removed even before the interrupt handling routine could acknowledge it.

[bit0] AICD: Auto Interrupt Clear Disable

Bit	Description
0	Auto Interrupt Clear is enabled (DMA can not be used)
1	Auto Interrupt Clear is disabled (DMA can be used)

- When this bit is "0", then SSRn:TDRE, SSRn: RDRF, ESRn:PEFRD, CSCRn:CRCERR, ESRn:BUSERR, ESRn:SYNFE and ECCRn:RBI are used to generate interrupts. In this mode, the USART can not be used together with DMA.
- When this bit is "1", then ESIRn:TDRE, ESIRn:RDRF, ESRn:PEFRD CSCRn:CRCERR, ESRn:BUSERR, ESRn:SYNFE and ESIRn:RBI are used to generate interrupts. In this mode, the USART can be used together with DMA.

Note:

When AICD = 0, the USART can not be used with DMA. If DMA is used to handle data transfers of the USART, set AICD = 1.

7.9. Transmission FIFO Control Register (TFCRn)

This register specifies the interrupt level for transmission FIFO, enables the transmission FIFO and resets the FIFO pointer.

■ Transmission FIFO Control Register (TFCRn)

TFCRn								
bit	15	14	13	12	11	10	9	8
	TXFE	TXFCL	-	TXFLC4	TXFLC3	TXFLC2	TXFLC1	TXFLC0
Attribute	R/W	W	-	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	X	0	0	0	0	0

n = 0...9 according to device (Please refer to the datasheet of the corresponding device)

[bit15] TXFE: Transmission FIFO Enable Bit

Bit	Description
0	Transmission FIFO is disabled
1	Transmission FIFO is enabled

[bit14] TXFCL: Transmission FIFO Clear Bit

Bit	Description	
	Read	Write
0	Always read 0	No effect
1		FIFO pointer reset to 5’b00000

Notes:

- Transmission FIFO pointers are also cleared when mode change occurs.
- Transmission FIFO pointers are also cleared when SMRn:UPCL register bit is set to "1".

[bit13] -:Undefined

Read value is undefined, always write 0

Read modify write operations to this register have no effect on this bit.

[bit12 to bit8] TXFLC[4:0]: Transmission FIFO Interrupt Level Configuration

- These bits specify the transmission FIFO interrupt level.
- When the number of data in the transmission FIFO is less or equal to the trigger level programmed by these bits, a transmission interrupt is asserted.

bit12	bit11	bit10	bit9	bit8	Description
0	0	0	0	0	TX interrupt if TX FIFO is empty
0	0	0	0	1	TX interrupt if number of data ≤ 1 data byte
0	0	0	1	0	TX interrupt if number of data ≤ 2 data bytes
0	0	0	1	1	TX interrupt if number of data ≤ 3 data bytes
0	0	1	0	0	TX interrupt if number of data ≤ 4 data bytes
0	0	1	0	1	TX interrupt if number of data ≤ 5 data bytes
0	0	1	1	0	TX interrupt if number of data ≤ 6 data bytes
0	0	1	1	1	TX interrupt if number of data ≤ 7 data bytes
0	1	0	0	0	TX interrupt if number of data ≤ 8 data bytes
0	1	0	0	1	TX interrupt if number of data ≤ 9 data bytes
0	1	0	1	0	TX interrupt if number of data ≤ 10 data bytes
0	1	0	1	1	TX interrupt if number of data ≤ 11 data bytes
0	1	1	0	0	TX interrupt if number of data ≤ 12 data bytes
0	1	1	0	1	TX interrupt if number of data ≤ 13 data bytes
0	1	1	1	0	TX interrupt if number of data ≤ 14 data bytes
0	1	1	1	1	TX interrupt if number of data ≤ 15 data bytes
1	0	0	0	0	TX interrupt if number of data ≤ 16 data bytes

- Reading these bits returns the written value.
- Note: The bit combinations from 17 to 32 are not allowed as the FIFO size is 16 bytes.

7.10.Reception FIFO Control Register (RFCRn)

This register specifies interrupt level of reception FIFO, enables the reception FIFO and resets the FIFO pointer.

■ Reception FIFO Control Register (RFCRn)

RFCRn								
bit	7	6	5	4	3	2	1	0
	RXFE	RXFCL	-	RXFLC4	RXFLC3	RXFLC2	RXFLC1	RXFLC0
Attribute	R/W	W	-	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	X	0	0	0	0	0

n = 0...9 according to device (Please refer to the datasheet of the corresponding device)

[bit7] RXFE: Reception FIFO Enable

Bit	Description
0	Reception FIFO is disabled
1	Reception FIFO is enabled

[bit6] RXFCL: Reception FIFO Clear Bit

Bit	Description	
	Read	Write
0	Always read 0	No effect
1		FIFO pointer reset to 5’b00000

Notes:

- Reception FIFO pointers are also cleared when mode change occurs.
- Reception FIFO pointers are also cleared when SMRn:UPCL register bit is set to "1".

[bit5] -:Undefined

Read value is undefined, always write 0

Read modify write operations to this register have no effect on this bit.

[bit4 to bit0] RXFLC[4:0]: Reception FIFO Level Configuration

- These bits specify the reception FIFO interrupt level.
- When the number of data in the reception FIFO is greater than or equal to the trigger level value programmed by these bits, a reception interrupt is asserted.

bit4	bit3	bit2	bit1	bit0	Description
0	0	0	0	0	RX interrupt if RX FIFO is empty
0	0	0	0	1	RX interrupt if number of data ≥ 1 data byte
0	0	0	1	0	RX interrupt if number of data ≥ 2 data bytes
0	0	0	1	1	RX interrupt if number of data ≥ 3 data bytes
0	0	1	0	0	RX interrupt if number of data ≥ 4 data bytes
0	0	1	0	1	RX interrupt if number of data ≥ 5 data bytes
0	0	1	1	0	RX interrupt if number of data ≥ 6 data bytes
0	0	1	1	1	RX interrupt if number of data ≥ 7 data bytes
0	1	0	0	0	RX interrupt if number of data ≥ 8 data bytes
0	1	0	0	1	RX interrupt if number of data ≥ 9 data bytes
0	1	0	1	0	RX interrupt if number of data ≥ 10 data bytes
0	1	0	1	1	RX interrupt if number of data ≥ 11 data bytes
0	1	1	0	0	RX interrupt if number of data ≥ 12 data bytes
0	1	1	0	1	RX interrupt if number of data ≥ 13 data bytes
0	1	1	1	0	RX interrupt if number of data ≥ 14 data bytes
0	1	1	1	1	RX interrupt if number of data ≥ 15 data bytes
1	0	0	0	0	RX interrupt if number of data ≥ 16 data bytes

- Note: The bit combinations from 17 to 32 are not allowed as the FIFO size is 16 bytes.

7.11. Transmission FIFO Status Register (TFSRn)

This register indicates the number of valid data bytes available in the transmit FIFO.

■ Transmit FIFO Status Register (TFSRn)

TFSRn								
bit	15	14	13	12	11	10	9	8
	-	-	-	TXFVD4	TXFVD3	TXFVD2	TXFVD1	TXFVD0
Attribute				R	R	R	R	R
Initial value				0	0	0	0	0

n = 0...9 according to device (Please refer to the datasheet of the corresponding device)

[bit15 to bit13] -:Undefined
Read value is 0, always write 0
Read modify write operations to this register have no effect on this bit.

[bit12 to bit8] TXFVD[4:0]: Transmission FIFO Valid Data

- These bits indicate the number of valid data in the transmission FIF0.

bit12	bit11	bit10	bit9	bit8	Description
0	0	0	0	0	the number of valid data is empty
0	0	0	0	1	the number of valid data = 1
0	0	0	1	0	the number of valid data = 2
0	0	0	1	1	the number of valid data = 3
0	0	1	0	0	the number of valid data = 4
0	0	1	0	1	the number of valid data = 5
0	0	1	1	0	the number of valid data = 6
0	0	1	1	1	the number of valid data = 7
0	1	0	0	0	the number of valid data = 8
0	1	0	0	1	the number of valid data = 9
0	1	0	1	0	the number of valid data = 10
0	1	0	1	1	the number of valid data = 11
0	1	1	0	0	the number of valid data = 12
0	1	1	0	1	the number of valid data = 13
0	1	1	1	0	the number of valid data = 14
0	1	1	1	1	the number of valid data = 15
1	0	0	0	0	the number of valid data = 16

7.12.Reception FIFO Status Register (RFSRn)

The Reception FIFO status register indicates the number of valid data available in the reception FIFO.

■ Reception FIFO Status Register (RFSRn)

RFSRn								
bit	7	6	5	4	3	2	1	0
	-	-	-	RXFVD4	RXFVD3	RXFVD2	RXFVD1	RXFVD0
Attribute				R	R	R	R	R
Initial value				0	0	0	0	0

n = 0...9 according to device (Please refer to the datasheet of the corresponding device)

[bit7 to bit5] -:Undefined
Read value is 0, always write 0
Read modify write operations to this register have no effect on this bit.

[bit4 to bit0] RXFVD[4:0]: Receive FIFO Valid Data

- These bits indicate the number of valid data in the reception FIF0.

bit4	bit3	bit2	bit1	bit0	Description
0	0	0	0	0	the number of valid data is empty
0	0	0	0	1	the number of valid data = 1
0	0	0	1	0	the number of valid data = 2
0	0	0	1	1	the number of valid data = 3
0	0	1	0	0	the number of valid data = 4
0	0	1	0	1	the number of valid data = 5
0	0	1	1	0	the number of valid data = 6
0	0	1	1	1	the number of valid data = 7
0	1	0	0	0	the number of valid data = 8
0	1	0	0	1	the number of valid data = 9
0	1	0	1	0	the number of valid data = 10
0	1	0	1	1	the number of valid data = 11
0	1	1	0	0	the number of valid data = 12
0	1	1	0	1	the number of valid data = 13
0	1	1	1	0	the number of valid data = 14
0	1	1	1	1	the number of valid data = 15
1	0	0	0	0	the number of valid data = 16

7.13.Sync Field Time-out Register (SFTRn)

This register specifies the timeout value for sync field detection

■ Sync Field Timeout Register (SFTRn)

SFTRn

bit	15	14	13	12	11	10	9	8
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

bit	7	6	5	4	3	2	1	0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

n = 0...9 according to device (Please refer to the datasheet of the corresponding device)

[bit15 to bit8]

Bit	Description
Read	Read bit 15 to 8 of time-out value to SFTRn
Write	Write bit 15 to 8 of time-out value to SFTRn

[bit7 to bit0]

Bit	Description
Read	Read bit 7 to 0 of time-out value to SFTRn
Write	Write bit 7 to 0 of time-out value to SFTRn

● Syncfield Timeout Register

SFTRn contains 16 bits of timeout value for sync field detection. At the rising edge of SINn after LIN break, the Sync field timeout counter starts incrementing. When the Sync field timeout counter value is less than the value programmed in the SFTRn and the fifth falling edge of Sync field is detected, ESRn:SYNFE (timeout error flag) will not be asserted. If the Sync field timeout counter value reaches the Sync field time out value before detecting the fifth falling edge of the Sync field, the ESRn:SYNFE (timeout error flag) is set. This will result in a reception interrupt if the EIERn:SYNFEIE is set to "1".
When this register is set to 0x0000, sync field time out detection is disabled (default).

7.14.Checksum Status and Control Register (CSCRn)

Checksum status and control register consists of control bits for enabling checksum generation and verification during transmission and reception respectively and LIN frame data length. It also contains the status bit for indicating the status of check sum error.

■ Checksum Status and Control Register (CSCRn)

CSCRn								
bit	7	6	5	4	3	2	1	0
	CRCERRIE	CRCERR	CRCTYPE	CRCCHECK	CRCGEN	DL2	DL1	DL0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

n = 0...9 according to device (Please refer to the datasheet of the corresponding device)

[bit7] CRCERRIE: Checksum Error Interrupt Enable

Bit	Description
0	Checksum error interrupt disabled
1	Checksum error interrupt enabled

- This bit functions as the checksum error interrupt enable.
- When this bit is set to "0" and checksum error occurs (CSCRn:CRCERR = 1), reception interrupt will not be generated.
- When this bit is set to "1" and checksum error occurs (CSCRn:CRCERR = 1), reception interrupt will be generated.

[bit6] CRCERR: Checksum Error Flag

Bit	Description	
	Read	Write
0	No checksum error detected	Clear checksum error flag
1	Checksum error detected	No effect

- The checksum error is indicated by this flag.
- When the CRCERR flag is asserted an reception interrupt is generated when CSCRn:CRCERRIE is "1".
- The interrupt due to CRCERR flag is cleared by writing "0" to this register bit (CSCRn:CRCERR) or by writing "1" to SCRn:CRE.
- The check sum is not checked if a parity error in the Frame-ID has been received before (ESRn:PEFRD="1"). To enable check sum verification, clear ESRn:PEFRD before receiving the frame data.

[bit5] CRCTYPE: Type of Checksum

Bit	Description
0	Classic checksum
1	Enhanced checksum

- This bit sets the checksum type.
- When this bit is set to "0", classic check sum calculation is performed only over the data bytes.
- When this bit is set to "1", enhanced checksum calculation is performed over the data bytes as well as the Frame-ID. EFERn:FIDE (Frame-ID enable) determines if Frame-ID register or RDRn/TDRn resp. RX/TX-FIFO is used.
- Checksum calculation is performed according to LIN specification 2.1.

[bit4] CRCCHECK: Checksum Verification

Bit	Description
0	Disable checksum verification during reception
1	Enable checksum verification during reception

[bit3] CRCGEN: Checksum Generation

Bit	Description
0	Disable checksum generation during transmission
1	Enable checksum generation during transmission

[bit2 to bit0] DL[2:0]: Data Length in LIN Frame

bit2	bit1	bit0	Description
0	0	0	1 byte
0	0	1	2 bytes
0	1	0	3 bytes
0	1	1	4 bytes
1	0	0	5 bytes
1	0	1	6 bytes
1	1	0	7 bytes
1	1	1	8 bytes

- The data length for the current frame needs to be programmed in this register for performing checksum calculation. Data length range from 1 to 8 can be programmed.

● Checksum Status and Control Register.

The checksum calculation is done only in LIN mode. The checksum contains the inverted eight bit sum with carry over all data bytes or all data bytes and the protected identifier (Frame-ID). The checksum calculation over the data bytes only is called classic checksum and checksum calculation over both data bytes and the protected identifier (Frame-ID) is called enhanced checksum. Checksum generation and Checksum verification are done according to the LIN specification 2.1.

● Checksum Enabling

CRCGEN	CRCHECK	Description
0	0	No checksum generation/Checksum verification
0	1	Checksum check on received data
1	0	Checksum generation for transmitting checksum byte
1	1	Checksum generated is sent and checksum check on data received back

Note:

CRCGEN/CRCCHECK: Can only be set if Mode 3({MD1,MD0} = "11") is selected.

7.15.Frame-ID Data Register (FIDRn)

This register stores the value of the Frame-ID in LIN mode.

■ Frame-ID Data Register (FIDRn)

FIDRn								
bit	7	6	5	4	3	2	1	0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

n = 0...9 according to device (Please refer to the datasheet of the corresponding device)

[bit7 to bit0] Frame-ID Data Register Bits

● Frame-ID Data Register:

FIDRn register contains the Frame-ID used for header transmission or reception depending on enabled automatic header transmission or reception.
If LIN-USART is used as LIN-Master and Frame-ID register is enabled by EFERn:FIDE Frame-ID is sent from this register. The value written to the Frame-ID register must be an 8-bit value including the parity bits of the Frame-ID.
If LIN-USART is used as LIN-Slave and Frame-ID register is enabled by EFERn:FIDE, Frame-ID is stored in this register including the parity bits.
When Frame-ID register is enabled along with FIFO, the Sync field data has to be set in the FIFO. In this case the Sync field value will be transmitted from the transmission FIFO and the Frame-ID data from Frame-ID register. From the start to end of Frame-ID transmission the read to TX FIFO will be halted.

7.16.Extended Feature Enable Register (EFERLn)

This register enables interrupt at the end of transmission/reception of header, auto baud rate detection/adjustment feature and edge sensitive detection of LIN break. It also disables resetting of reception state machine when CRE is set and disables detection of low level of SINn after FRE as start bit.

■ Extended Feature Enable Register (EFERLn)

EFERLn								
bit	7	6	5	4	3	2	1	0
	LBDSF	OSDE	DTSTART	RSTRFM	LBEDGE	ABRE	ENRXHR	ENTXHR
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

n = 0...9 according to device (Please refer to the datasheet of the corresponding device)

[bit7] LBDSF: LIN Break Detect Special Function Bit

Bit	Description
0	LIN break detect in sync field function disabled
1	LIN break detect in sync field function enabled

[bit6] OSDE: Oversampling Disable Bit

Bit	Description
0	5 times oversampling of serial data input is enabled.
1	5 times oversampling of serial data input is disabled

[bit5] DTSTART: Detect Start Bit

Bit	Description
0	Low level of SIN after FRE is detected as start bit
1	Next falling edge of SIN after FRE is detected as start bit

- This bit specifies that a low level (without falling edge) on SINn after a framing error (SSRn:FRE) will be detected as start bit.

[bit4] RSTRFM: Reset Reception State Machine

Bit	Description
0	Resetting of reception state machine enabled
1	Resetting of reception state machine disabled

- This bit specifies that the reception state machine will be reset or not when the SCRn:CRE is set to "1".

[bit3] LBEDGE: Edge Sensitive Detection of LIN Break

Bit	Description
0	LIN break detection is level sensitive
1	LIN break detection is falling edge sensitive

- This bit specifies the falling edge sensitive detection of LIN break signal.
- When this bit is set to "0", USART in LIN mode detects low level on SIN as the start of LIN break.
- When this bit is set to "1", USART in LIN mode detects falling edge of SIN as the start of LIN break.

[bit2] ABRE: Auto Baud Rate Detection/Adjustment Enable

Bit	Description
0	Auto baud rate detection/adjustment is disabled
1	Auto baud rate detection/adjustment is enabled

[bit1] ENRXHR: Automatic Header Detection Enable

Bit	Description
0	Automatic header reception is disabled
1	Automatic header reception is enabled

- This bit serves as the feature enable for single interrupt at the end of reception of LIN frame header.
- When this bit is set to "0", all steps of header reception must be handled by software (LIN break detection, LIN sync field detection, etc.).
- When this bit is set to "1", header reception is handled by USART till Frame-ID is received. Reception interrupt is generated after complete header is received and EIERn:RXHDIE is set to "1".

[bit0] ENTXHR: Automatic Header Transmission Enable

Bit	Description
0	Automatic header transmission is disabled
1	Automatic header transmission is enabled

- This bit serves as the feature enable for interrupt at end of transmission of LIN frame header.
- When this bit is set to "0", all steps of header transmission must be handled by software (LIN break generation, sending LIN sync field, etc.).
- When this bit is set to "1", header transmission is handled by USART till Frame-ID is transmitted. This handling is started by setting ECCRn:LBR to "1". Transmission interrupt is generated when Frame-ID is transmitted and EIERn:TXHDIE is set to "1".

7.17.Extended Feature Enable Register (EFERHn)

This register enables internal loop back testing, Frame-ID register and detection of bus error. It also specifies MSB bit of LIN break length.

■ Extended Feature Enable Register (EFERHn)

EFERHn								
bit	15	14	13	12	11	10	9	8
	-	-	INTLBEN	BRGR	FIDPE	DBE	FIDE	LBL2
Attribute	-	-	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	X	X	0	0	0	0	0	0

n = 0...9 according to device (Please refer to the datasheet of the corresponding device)

[bit15, bit14] -: Undefined

Write always "0".
Read value is undefined.
Read modify write operations to this register have no effect on these bits.

[bit13] INTLBEN: Internal Loop Back Enable

Bit	Description
0	Internal loop back is disabled
1	Internal loop back is enabled

- When internal loop back testing is enabled, the TX output is internally fed back to the RX input so that software can compare the transmitted and received value.

[bit12] BRGR: Baud Rate Read Enable Bit

Bit	Description
0	Reads reload counter value
1	Reads the reload value wrriten by CPU or ABR block

- Baud rate generation register read
- When this bit is set to "0", read to baud rate generation register returns the current value of reload counter.
- When this bit is set to "1", read to baud rate generation register returns the reload value written by the CPU or auto baud rate generation/adjustment circuitry

[bit11] FIDPE: Frame-ID Parity Calculation Enable Bit

Bit	Description
0	Parity check of frame-ID is disabled in LIN mode.
1	Parity check of frame-ID is enabled in LIN mode

- This bit enabled the check of the parity bits in the received Frame-ID. This setting is only valid if LIN-USART is configured as LIN-Slave. Parity is checked according to calculation specified in LIN-Specification 2.1.

[bit10] DBE: Detection of Bus Error Enable

Bit	Description
0	Detection of bus error disabled
1	Detection of bus error enabled

Notes:

- Enable/Disable of Bus Error Detect feature should be done only when bus is idle.
- Bus Error will be continuously triggered, if bus error detection is enabled (EFERHn:DBE = "1") in internal loop back mode (EFERHn:INTLBEN = "1"), because in internal loop back mode SOUT is always "1".

[bit9] FIDE: Frame-ID Register Enable

Bit	Description
0	Frame-ID is transmitted/received from TDRn/RDRn or FIFO
1	Frame-ID is transmitted/received from Frame-ID reg.

- This bit acts as the Frame-ID register enable when operating in LIN mode.
- When this bit is specified as "0", the Frame-ID of the LIN frame header is transmitted from or received to the TDRn/RDRn respectively or TX-/RX-FIFO.
- When this bit is set to "1", the Frame-ID of LIN frame header is transmitted from or received to the Frame-ID register.

[bit8] LBL2: MSB Bit of LIN Break Length

Bit	Description
0	LBL[1:0] set LIN break length from 13 to 16
1	LBL[1:0] set LIN break length from 17 to 20

- This bit serves as the MSB bit of the LIN break length configuration when operating in LIN mode.
- Table for LIN break length setting is given below.

LBL2	ESCRn:LBL1	ESCRn: LBL0	LIN break length
0	0	0	13 bit times
0	0	1	14 bit times
0	1	0	15 bit times
0	1	1	16 bit times
1	0	0	17 bit times
1	0	1	18 bit times
1	1	0	19 bit times
1	1	1	20 bit times

7.18.Extended Interrupt Enable Register (EIERn)

This register specifies interrupt enables for various interrupt sources.

■ Configuration Bits of Extended Interrupt Enable Register (EIERn)

EIERn								
bit	7	6	5	4	3	2	1	0
	TXFIE	RXFIE	SYNFEIE	RXHDIE	TXHDIE	PEFRDIE	BUSERRIE	LBSOIE
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

n = 0...9 according to device (Please refer to the datasheet of the corresponding device)

[bit7] TXFIE: Transmission FIFO Interrupt Enable

Bit	Description
0	Transmission FIFO trigger level interrupt is disabled
1	Transmission FIFO trigger level interrupt is enabled

[bit6] RXFIE: Reception FIFO Interrupt Enable

Bit	Description
0	Reception FIFO trigger level interrupt is disabled
1	Reception FIFO trigger level interrupt is enabled

[bit5] SYNFEIE: Sync Field Error Interrupt Enable

Bit	Description
0	Sync field time out error interrupt disabled
1	Sync field time out error interrupt enabled

- This bit can only be used in LIN mode.

[bit4] RXHDIE: Reception Header Interrupt Enable

Bit	Description
0	Reception header interrupt disabled
1	Reception header interrupt enabled

- When this bit is set to "0", interrupt generation at end of header reception is disabled.
- When this bit is set to "1", interrupt generation at end of header reception is enabled. EFERn:ENRXHR must be set to "1" to use this interrupt.
- This bit can only be used in LIN mode.

[bit3] TXHDIE: Transmission Header Interrupt Enable

Bit	Description
0	Transmission header interrupt disabled
1	Transmission header interrupt enabled

- When this bit is set to "0", interrupt generation at end of header transmission is disabled.
- When this bit is set to "1", interrupt generation at end of header transmission is enabled. EFERn:ENTXHR must be set to "1" to use this interrupt.
- This bit can only be used in LIN mode.

[bit2] PEFRDIE: Parity Error in Frame-ID Interrupt Enable

Bit	Description
0	Parity error in Frame-ID interrupt disabled
1	Parity error in Frame-ID interrupt enabled

- When this bit is set to "0", reception interrupt corresponding to parity error in Frame-ID is disabled.
- When this bit is set to "1", reception interrupt corresponding to parity error in Frame-ID is enabled.
- This bit can only be used in LIN mode.

[bit1] BUSERRIE: Bus Error Interrupt Enable

Bit	Description
0	Bus error interrupt disabled
1	Bus error interrupt enabled

- When this bit is set to "0", reception interrupt corresponding to bus error is disabled.
- When this bit is set to "1", reception interrupt corresponding to bus error is enabled.
- This bit can only be used in LIN mode or if SOTn and SINn are connected K-line adapter.

[bit0] LBSOIE: Last Bit Shifted Out Interrupt Enable

Bit	Description
0	Last bit shifted out interrupt disabled
1	Last bit shifted out interrupt enabled

- When this bit is set to "0", transmission interrupt corresponding to last bit shift out is disabled.
- When this bit is set to "1", transmission interrupt corresponding to last bit shift out in byte field is enabled. This interrupt can be used as replacement of transmission data register empty interrupt (SSRn:TIE). Only one of these interrupts should be enabled at the same time.

7.19.Extended Status Register (ESRn)

This register indicates the status of detection of sync field, Frame-ID reception and detection of bus error.

■ Extended Status Register (ESRn)

ESRn								
bit	15	14	13	12	11	10	9	8
	-	-	TXHRI	RXHRI	LBSOF	BUSERR	PEFRD	SYNFE
Attribute	-	-	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	X	X	0	0	0	0	0	0

n = 0...9 according to device (Please refer to the datasheet of the corresponding device)

[bit15, bit14] -: Undefined

Write always "0".
Read value is undefined.
Read modify write operations to this register have no effect on these bits.

[bit13] TXHRI: Transmission of Header Interrupt Flag

Bit	Description
0	Automatic transmission of header not completed
1	Automatic transmission of header completed

- TXHRI is set at the same time as the stop bit of the Frame-ID is transmitted.
- Read-modify-write instructions always return 1.

[bit12] RXHRI: Reception of Header Interrupt Flag

Bit	Description
0	Automatic reception of header not completed
1	Automatic reception of header completed

- Read-modify-write instructions always return 1.

[bit11] LBSOF: Last Bit Shift Out Flag

Bit	Description	
	Read	Write
0	Last bit not yet shifted out	Clear LBSOF
1	Last bit shifted out	No effect

- This bit indicates the last bit of transmission data byte is shifted out.
  In case of synchronous mode (mode 2) and if USART is master (ECCRn:MS="0") or continuous clock mode is enabled in slave mode (ECCRn:MS="1" and ESCRn:CCO="1"), this bit is set when the last data bit of data byte or last stop bit was shifted out.
  In case of synchronous mode (mode 2) and continuous clock is disabled in slave mode (ECCRn:MS="1" and ESCRn:CCO="0"), this flag is set in the middle of the last data bit or stop bit.
  In case of asynchronous mode, this bit is set when the stop bit was shifted out.
- This bit is cleared by writing "0" to this bit or by writing a value to the TDRn/TX-FIFO.
- Read-modify-write instructions always return 1.

[bit10] BUSERR: Bus Error Flag

Bit	Description
0	No physical bus error detected.
1	Physical bus error detected

- A reception interrupt will be asserted, when this bit is "1" as a result of bus error. This interrupt can be cleared by writing "0" to this bit.
- This bit can only be used in LIN mode or if SOTn and SINn are connected K-line adapter.
- Read-modify-write instructions always return 1.

[bit9] PEFRD: Parity Error in Frame-ID

Bit	Description
0	Frame-ID is received without error
1	Frame-ID is received with parity error

- When this bit is "0", it indicates the reception of Frame-ID without parity error.
- When this bit is "1", it indicates parity error during reception of Frame-ID.
- A reception interrupt will be asserted when this flag becomes one as a result of parity error in Frame-ID. This interrupt is cleared by writing "0" to this bit.
- Read-modify-write instructions always return 1.

[bit8] SYNFE: Sync Field Error

Bit	Description
0	Sync field got detected correctly
1	Time out error during sync field detection

- When this bit is "0", it indicates detection of Sync field without timeout error.
- When this bit is "1", it indicates timeout error during Sync field detection.
- A reception interrupt will be asserted, when this bit is "1" because of Sync field time out error. This interrupt can be cleared by writing "0" to this bit.
- Read-modify-write instructions always return 1.

Note:

When Autobaud rate detection is used and ESRn:SYNFE is set, Baud rate reload register is not updated.

8. Notes on Using

Notes on using USART are given below.

■ Notes on using USART

● Enabling Operation

In USART, the control register (SCRn) has TXE (transmission) and RXE (reception) operation enable bits. Both, data transmission and reception operations, must be enabled before the communication starts because they have been disabled as the default value (initial value). The operation can also be canceled by disabling these bits.
Automatic LIN header transmission and reception in mode 3 are independent of TXE and RXE.

In synchronous slave mode (mode2) the transmission enable (SCRn.TXE) and reception enable (SCRn.RXE) should be set before the clock starts at the SCK port.

● Auto Header Detection in LIN Mode.

During reception of Frame-ID in auto header detection feature, the read to SCR register will return RXE bit as "1", though the SCR[1] is written with the value of "0".

● Communication Mode Setting

Set the communication mode while the system is not operating. If the mode is changed during transmission or reception, the transmission or reception is stopped and possible data will be lost.

● Transmission Interrupt Enabling Timing

The default (initial value) of the transmission data empty flag bit (SSRn:TDRE) is "1" (no transmission data and transmission data write enable state). A transmission interrupt request is generated as soon as the transmission interrupt request is enabled (SSRn:TIE=1). Be sure to set the TIE flag to "1" after setting the transmission data to avoid an immediate interrupt.

● Using LIN Operation Mode 3

The LIN features are available in mode 3, but using mode 3 sets the USART data format automatically to LIN format (8N1, LSB first). Note, that the length of the synch break for transmission is variable but for reception it is fixed 11-bit times.

● Changing Operation Settings

It is strongly recommended to reset the USART after changing operation settings. Particularly if (for example) start-/stop-bits added to or removed from the data format.
It is recommended to disable the communication (RXE = "0", TXE = "0"), if the USART setting or mode is changed or the USART is reset.

● Using Synchronous Slave Mode without Continuous Clock (ESCRn:CCO = 0)

In synchronous slave mode without continuous clock, the write to transmission data register (TDRn) must be done before providing the clock for transmission operation.
The approximate time before which the data must be written into the TDR , should be greater than half serial clock time period plus one CLKP time period (Assuming that it takes one CLKP time period for data to be written to the TDR register).

● Using Transmission/Reception FIFO

FIFO has to be cleared using TFCRn:TXFCL/RFCRn:RXFCL before enabling or disabling the respective FIFO.

● Using Auto Header Transmission Feature without enabling Frame-ID Register (EFERHn:FIDE = 0) in LIN Mode

For auto header transmission with the Frame-ID register disabled (EFERHn:FIDE = 0), the below specified order of programming is recommended.

1. EFERLn:ENTXHR= 1
2. ECCRn:LBR = 1
3. TDRn= Frame-ID value

If frame-ID is written into TDR, before setting ECCRn:LBR =1, data from TDR is transmitted before the LIN break and sync field because it is handled as normal data. This is similar to the flow utilized when auto header transmission is disabled.

● Using Last Bit Shift Out Interrupt

In all the modes, synchronization of status flag ESRn:LBSOF in bus bridge will add up to the interrupt latency, and also the ISR (interrupt service routine) call will add up to the interrupt latency.

Note:

If settings in the Serial Mode Register (SMRn) are desired, it is not useful to set the UPCL bit at the same time to reset the USART. The correct operation settings are not guaranteed in this case. Thus it is recommended to set the bits of the SMRn and then to set them again plus the UPCL bit later.

● LIN Slave Settings

Set the baud rate before receiving the first LIN synch break for the slave operation. Otherwise, duration of the synch break can not be correctly checked against the minimum requirement of the LIN specification (13 master bit time and 11 slave bit time).

● Bus Idle Function

The Bus Idle Function cannot be used in synchronous slave mode 2.

● AD Bit (Serial control register (SCRn): address/data type select bit)

Special care has to be taken when using the SCRn:AD bit (Address-Data-Bit for multiprocessor mode 1) of the Serial Control Register. This bit is both a control and a flag bit, because writing to it sets the AD bit for transmission, whereas reading from it returns the last received AD bit. Internally, the received and the transmitted value are stored in different registers, but in Read-Modify-Write instructions, the received value is read, modified and then written back for transmission. This can lead to a wrong value in the AD bit, when one of the other bits in the same register is accessed by an instruction of this kind.

● Clearing Reception Errors

Clearing reception errors resets the reception state machine when EFERn:RSTRFM = 0. Therefore check any reception errors before the next start-bit or start condition is met, to not disturb any ongoing reception. If EFERn:RSTRFM="1" reception state machine is not reset by SCRn:CRE.

● LIN Synch Field Wait State

In mode 3 (LIN operation), the LBD bit in the ESCRn register is set to "1" if the input signal is kept at "0" for more than or equal to 10-bit times. Then the USART waits for the following synch field to be received. If the USART is set into this state for other reasons than the synch break, it should be initialized by the software reset (SMRn:UPCL=1).

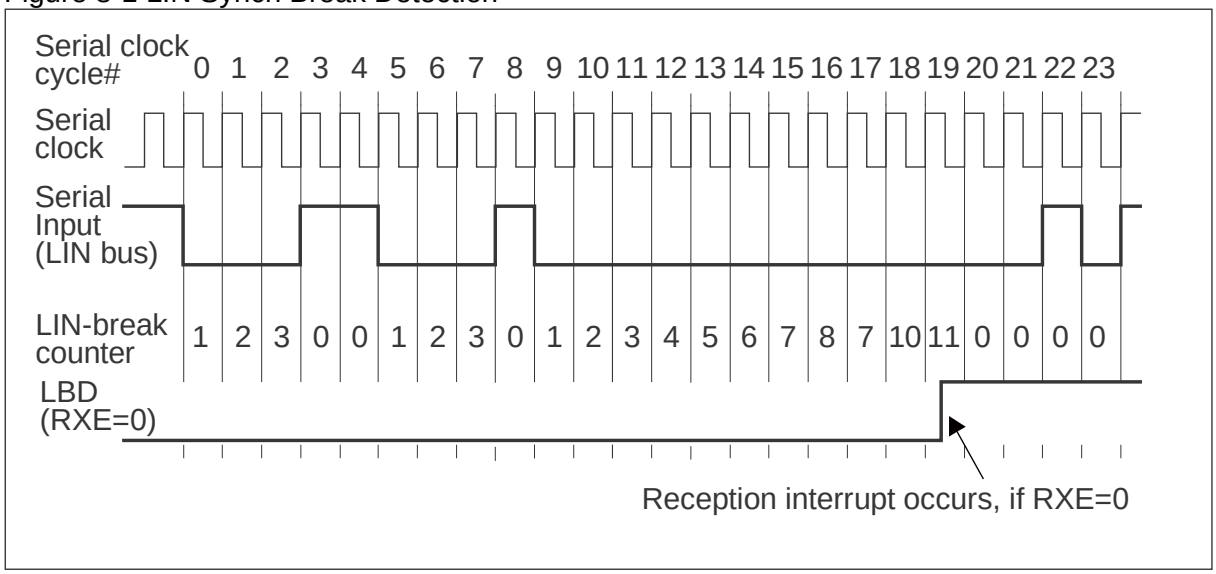
In mode 3, LIN-Break detection is always working in the background and is level sensitive or edge sensitive depending on EFERn:LBEDGE. Be careful in case of a bus error (bus always dominant). The LIN-Break detection flag (LBD) will go "1" or stay "1" after each 10.5 bit times when LBEDGE = 0 independent from enabled or disabled LIN break detection interrupt. If you use LIN-Break detection interrupt, be sure to check and clear always this flag in your reception interrupt handler.

If EFERn:LBEDGE is "0" (level sensitive detection of LIN-break start) LIN-break detection is restarted with every high level on SIN till a valid LIN-sync field has been detected.

If EFERn:LBEDGE is "1" (edge sensitive detection of LIN-break start) LIN-break detection is restarted with every falling edge on SIN till a valid LIN-sync field has been detected.

Below figure shows the behavior of the LIN-break detection counter. This figure doesn't distinguish between EFERn:LBEDGE settings because the behavior in this scenario is same.

Figure 8-1 LIN Synch Break Detection

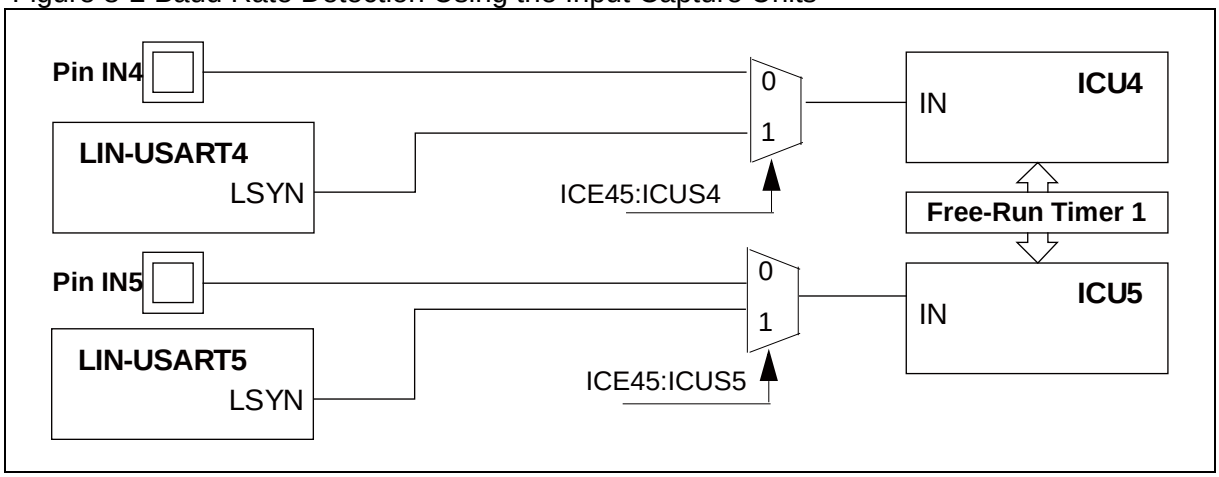


After a LIN-break has been detected by the LIN-USART, the LIN-break detection counter must be reset by SMRn:UPCL before it can detect a new LIN-break.

● Baud Rate Detection using the Input Capture Units (ICU)

The USARTs provide the signal LSYN that can be connected to the ICU so that LSYN's pulse length can be measured to derive the baud rate. The connection of the LSYN signals to the ICUs is controlled by the Input capture edge register ICE(2n)(2n+1). For assignment of ICUs and USARTs please check "CHAPTER 1.6 Input Capture Unit source select for LIN-USART". This is independent from EFERn:ABRE setting. Automatic baud rate detection/adjustment function can be used alternatively.

Figure 8-2 Baud Rate Detection Using the Input Capture Units



If the ICUSx bit in the ICE (2n)(2n+1) register is cleared and the peripheral resource input is enabled, the ICU is connected to its corresponding input pin IN. If the ICUSx bit is set to "1" the corresponding LIN-USART is connected to the ICU.
The user has to take into account that: Different ICUs share one free running timer (prescaler).

● Effects of Reception Errors and CRE Bit.

If EFERLn:RSTRFM = 0
CRE resets reception state machine and next falling edge at SINn starts reception of new byte. Therefore either set CRE bit immediately (within half bit time) after receiving errors to prevent data stream de-synchronization or wait an application dependent time after receiving errors and set CRE, when SINn is idle.
If EFERLn:RSTRFM = 1
When CRE is set and the register bit EFERLn:RSTRFM is set as "1", reception state machine is not reset.

Figure 8-3 Timing of the CRE Bit (EFERn:RSTRFM = 0)

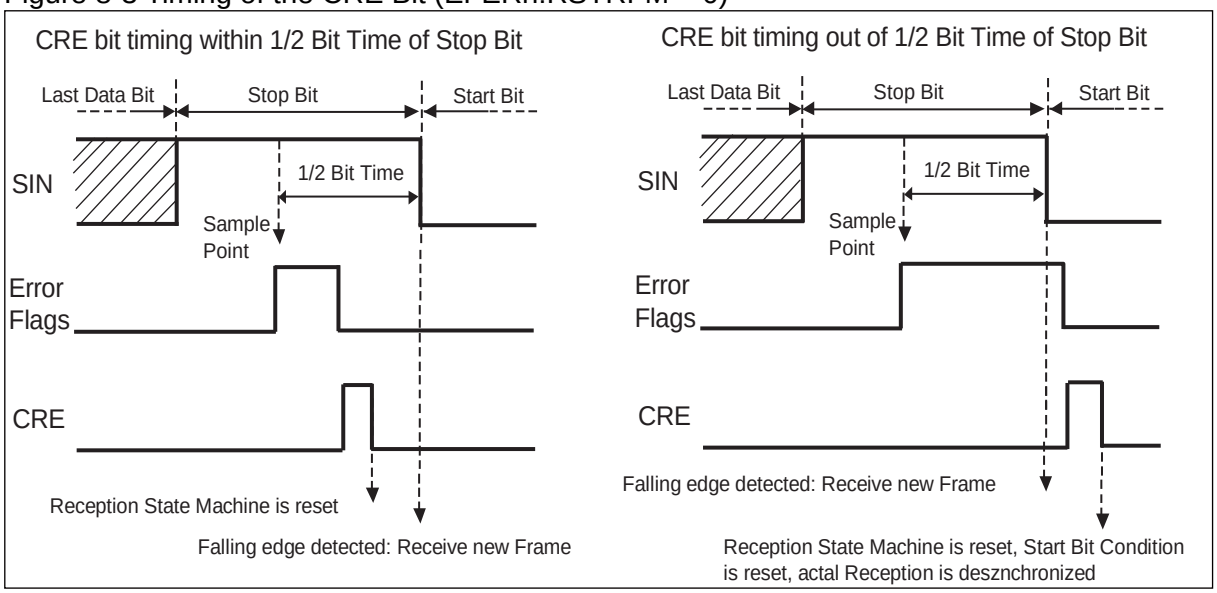
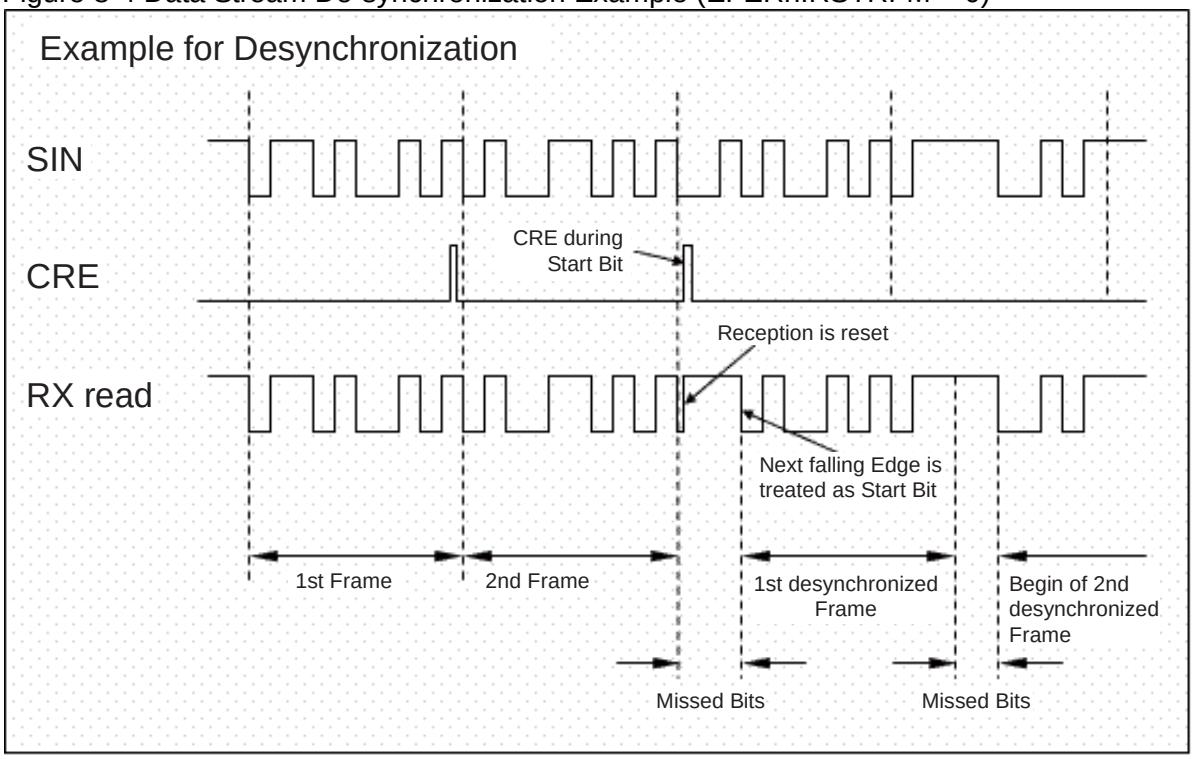


Figure 8-4 Data Stream De-synchronization Example (EFERn:RSTRFM = 0)



● Start Bit Detection

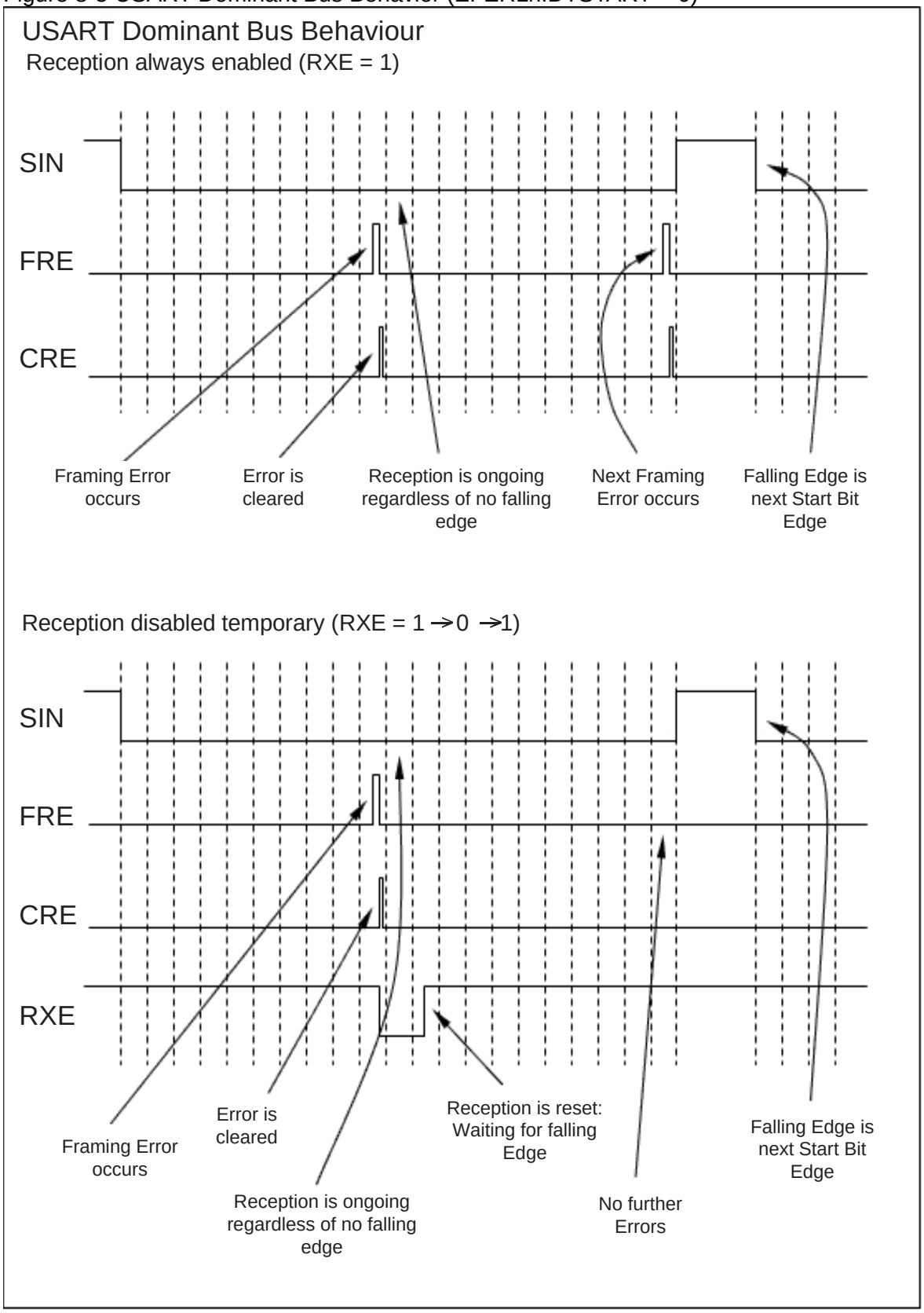
If EFERLn.DTSTART = 0

In case a framing error occurred (stop bit: SINn = "0") and the next start bit (SINn = "0") follows immediately, this start bit is recognized regardless of no falling edge before. This is used to keep the USART synchronized to the data stream and to determine bus always dominant errors (See Figure 8-5 lower figure) by producing next framing errors, if a recessive stop bit is expected. If this behavior is not wanted, please disable the reception temporarily (RXE = 1 -> 0 -> 1) after framing error. In this case, reception goes on at next falling edge on SINn. (See Figure 8-5 lower figure).

If EFERLn.DTSTART = "1":

Next falling edge of SIN input after FRE is considered as valid start bit.

Figure 8-5 USART Dominant Bus Behavior (EFERLn:DTSTART = 0)



● To avoid unwanted glitch at SCK.

"When performing software reset (writing "1" to SMR:UPCL) in master mode 2 (synchronous), with the mark level set to "0" (ESCRn:SCES = "1") special care has to be taken to avoid pulses on SCK. Please stick to the following precautions:

Once:

Set output data for the port function of the SCK pin to 0 by writing "0" the related PDR register bit and enable port output function for the SCK pin by writing "1" to the related DDR register bit.

At every software reset:

Disable SCK output by writing "0" to SMRn:SCKE before performing software reset by writing "1" to SMRn:UPCL. Then enable SCK output again by writing "1" to SMRn:SCKE

CHAPTER: 400 kHz I^{2}C INTERFACE

This section describes the functions and operation of the fast I^{2}C interface.

1. Overview
2. Operation
3. Programming Flow Charts
4. Registers

Management Code: 96F3I2C-E01.0

1. Overview

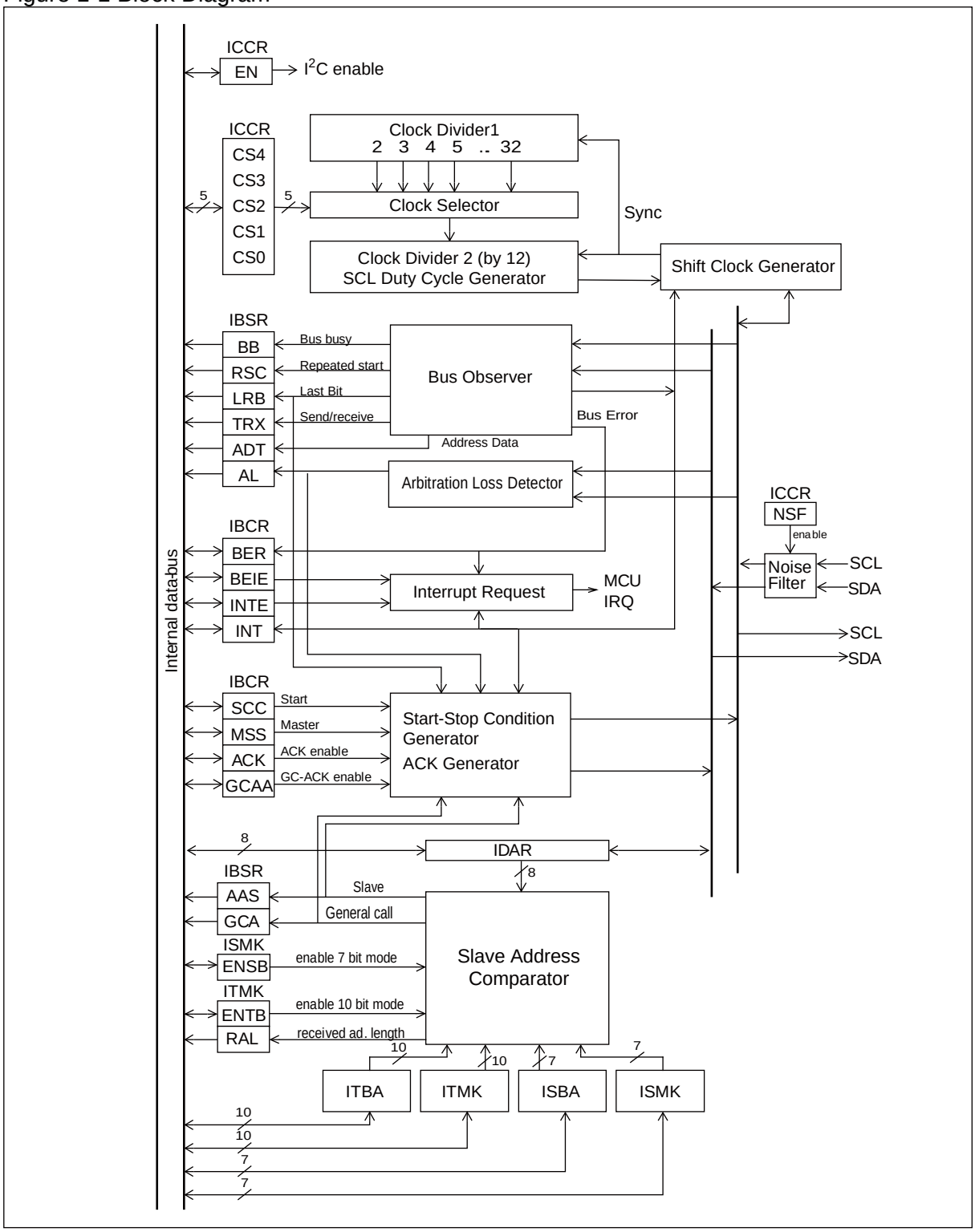
The I^2C interface is a serial I/O port supporting the Inter IC bus, operating as a master/slave device on the I^2C bus.

■ Features

- Master/slave transmitting and receiving functions
- Arbitration function
- Clock synchronization function
- General call addressing support
- Transfer direction detection function
- Repeated start condition generation and detection function
- Bus error detection function
- 7-bit addressing as master and slave
- 10-bit addressing as master and slave
- Possibility to give the interface a seven and a ten bit slave address
- Acknowledging upon slave address reception can be disabled (Master-only operation)
- Address masking to give interface several slave addresses (in 7 and 10-bit mode)
- Up to 400 KBit transfer rate
- Possibility to use built-in noise filters for SDA and SCL
- Can receive data at 400 KBit if peripheral clock CLKP1 is higher than 6 MHz regardless of prescaler setting
- Can generate MCU interrupts on transmission and bus error events
- Supports being slowed down by a slave on bit and byte level

The I^2C interface does not support SCL clock stretching on bit level since it can receive the full 400 kBit data rate if the peripheral clock CLKP1 is higher than 6 MHz regardless of the prescaler setting. However, clock stretching on byte level is performed since SCL is pulled low during an interrupt (INT="1" in IBCR register).

Figure 1-1 Block Diagram



2. Operation

The I^2C bus executes communication using two bi-directional bus lines, the serial data line (SDA) and serial clock line (SCL). The I^2C interface has two open-drain I/O pins (SDA/SCL) corresponding to these lines, enabling wired logic applications.

■ Start Conditions

When the bus is free (BB="0" in IBSR, MSS="0" in IBCR), writing "1" to the MSS bit places the I^2C interface in master mode and generates a start condition.

If a "1" is written to it while the bus is idle (MSS="0" and BB="0"), a start condition is generated and the contents of the IDAR register (which should be address data) is sent.

Repeated start conditions can be generated by writing "1" to the SCC bit when in bus master mode and interrupt status (MSS="1" and INT="1" in IBCR).

If a "1" is written to the MSS bit while the bus is in use (BB="1" and TRX="0" in IBSR; MSS="0"and INT="0"in IBCR), the interface waits until the bus is free and then starts sending.

If the interface is addressed as slave with write access (data reception) in the meantime, it will start sending after the transfer ended and the bus is free again. If the interface is sending data as slave in the meantime, it will not start sending data if the bus is free again. It is important to check whether the interface was addressed as slave (MSS="0" in IBCR and AAS="1" in IBSR), sent the data byte successfully (MSS="1" in IBCR) or failed to send the data byte (AL="1" in IBSR) at the next interrupt.

Writing "1" to the MSS bit or SCC bit in any other situation has no significance.

■ Stop Conditions

Writing "0" to the MSS bit in master mode (MSS="1" and INT="1" in IBCR) generates a stop condition and places the device in slave mode. Writing "0" to the MSS bit in any other situation has no significance.

After clearing the MSS bit, the interface tries to generate a stop condition which might fail if a certain external condition causes signal transition caused from "1" to "0" at the SCL line before the generation of this stop condition. In this case, the AL bit is set to "1" and interrupt is signaled at the end of the next byte.

■ Slave Address Detection

In slave mode, after a start condition is generated the BB is set to "1" and data sent from the master device is received into the IDAR register.

After the reception of eight bits, the contents of the IDAR register is compared to the ISBA register using the bit mask stored in ISMK if the ENSB bit in the ISMK register is "1". If a match results, the AAS bit is set to "1" and an acknowledge signal is sent to the master. Then bit 0 of the received data (bit 0 of the IDAR register) is inverted and stored in the TRX bit.

If the ENTB bit in the ITMK register is "1" and a ten bit address header (11110, TA1, TA0, write access) is detected, the interface sends an acknowledge signal to the master and stores the inverted last data bit in the TRX register. No interrupt is generated. Then, the next transferred byte is compared (using the bit mask stored in ITMK) to the lower byte of the ITBA register. If a match is found, an acknowledge signal is sent to the master, the AAS bit is set and an interrupt is generated.

If the interface was addressed as slave and detects a repeated start condition, the AAS bit is set after reception of the ten bit address header (11110, TA1, TA0, read access) and an interrupt is generated.

Since there are separate registers for the ten and seven bit address and their bit masks, it is possible to make the interface acknowledge on both addresses by setting the ENSB (in ISMK) and ENTB (in ITMK) bits.

The received slave address length (seven or ten bit) may be determined by reading the RAL bit in the ITMK register (this bit is valid if the AAS bit is set only).

It is also possible to give the interface no slave address by setting both bits to "0" if it is only used as a master.

All slave address bits may be masked with their corresponding mask register (ITMK or ISMK).

■ Slave Address Masking

Only the bits set to "1" in the mask registers (ITMK / ISMK) are used for address comparison, all other bits are ignored. The received slave address can be read from the ITBA (if ten bit address received, RAL="1") or ISBA (if seven bit address received, RAL="0") register if the AAS bit in the IBSR register is "1".
If the bit masks are cleared, the interface can be used as a bus monitor since it will always be addressed as slave. Note that this is not a real bus monitor because it acknowledges upon any slave address reception, even if there is no other slave listening.

■ Addressing Slaves

In master mode, after a start condition is generated the BB and TRX bits are set to "1" and the contents of the IDAR register is sent in MSB first order. After address data is sent and an acknowledge signal was received from the slave device, bit0 of the sent data (bit0 of the IDAR register after sending) is inverted and stored in the TRX bit. Acknowledgement by the slave may be checked using the LRB bit in the IBSR register. This procedure also applies to a repeated start condition.
In order to address a ten bit slave for write access, two bytes have to be sent. The first one is the ten bit address header which consists of the bit sequence "1 1 1 1 0 A9 A8 0", it is followed by the second byte containing the lower eight bits of the ten bit slave address (A7 to A0).
A ten bit slave is accessed for reading by sending the above byte sequence and generating a repeated start condition (SCC bit in IBCR) followed by a ten bit address header with read access (1 1 1 1 0 A9 A8 1).
Summary of the address data bytes:
7-bit slave, write access: Start condition - A6 A5 A4 A3 A2 A1 A0 0.
7-bit slave, read access: Start condition - A6 A5 A4 A3 A2 A1 A0 1.
10-bit slave, write access: Start condition - 1 1 1 1 0 A9 A8 0 - A7 A6 A5 A4 A3 A2 A1 A0.
10-bit slave, read access: Start condition - 1 1 1 1 0 A9 A8 1 - A7 A6 A5 A4 A3 A2 A1 A0 - repeated start - 1 1 1 1 0 A9 A8 1.

■ Arbitration

During sending in master mode, if another master device is sending data at the same time, arbitration is performed. If a device is sending the data value "1" and the data on the SDA line has an "L" level value, the device is considered to have lost arbitration, and the AL bit is set to "1." Also, the AL bit is set to "1" if a start condition is detected at the first bit of a data byte but the interface did not want to generate one or the generation of a start or stop condition failed by some reason.
Arbitration loss detection clears both the MSS and TRX bit and immediately places the device in slave mode so it is able to acknowledge if its own slave address is being sent.

■ Acknowledgement

Acknowledge bits are sent from the receiver to the transmitter. The ACK bit in the IBCR register can be used to select whether to send an acknowledgment when data bytes are received.
When data is send in slave mode (read access from another master), if no acknowledgement is received from the master, the TRX bit is set to "0" and the device goes to receiving mode. This enables the master to generate a stop condition as soon as the slave has released the SCL line.
In master mode, acknowledgement by the slave can be checked by reading the LRB bit in the IBSR register.

3. Programming Flow Charts

Each programming flow charts for the 400 kHz I^2C interface is shown below.

■ Programming Flow Charts

Figure 3-1 Example of Slave Addressing and Sending Data

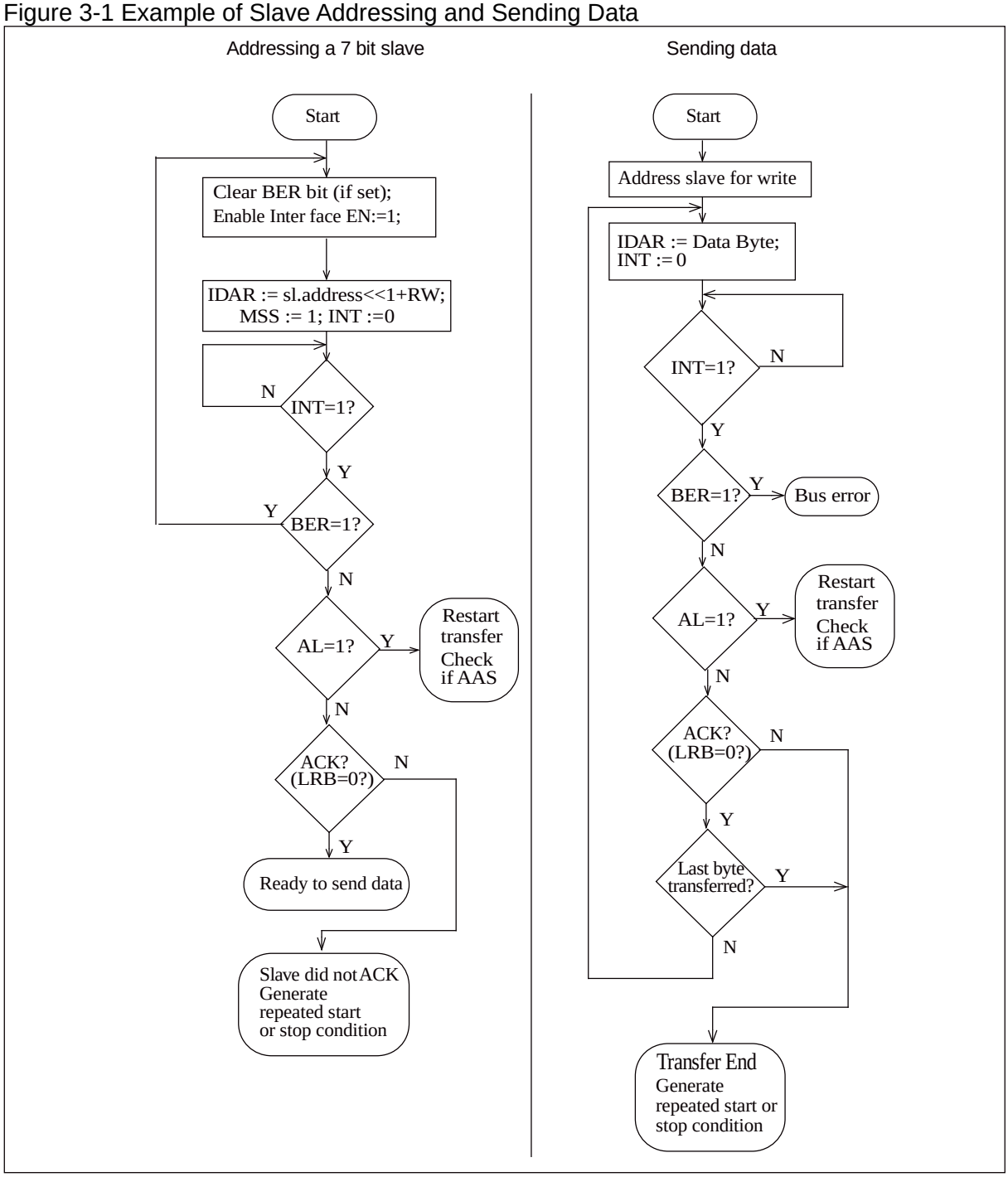
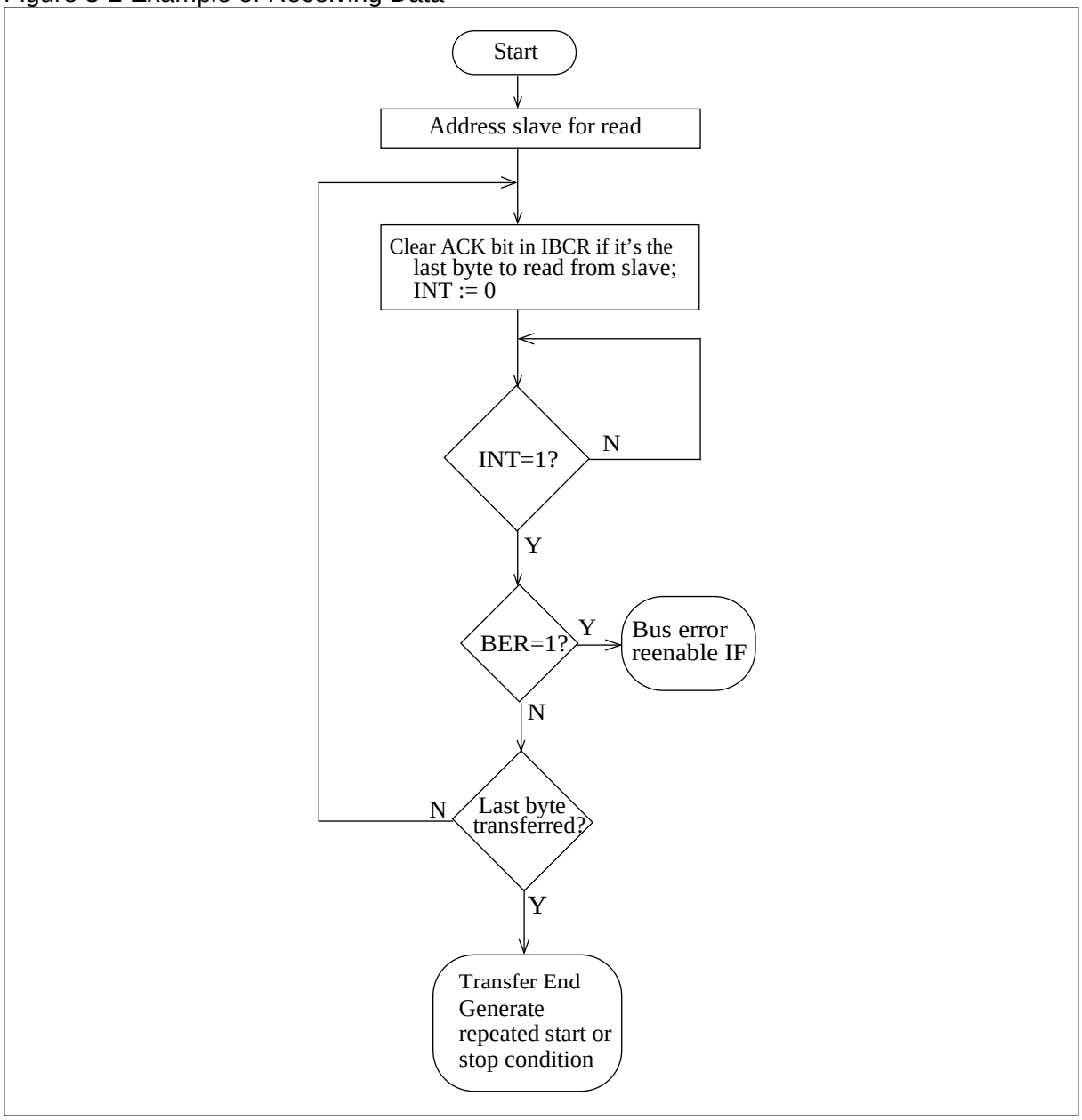


Figure 3-2 Example of Receiving Data



4. Registers

This section describes the function of the I^2C interface registers in detail.

■ List of I^2C Interface Registers

Abbreviated Register Name	Register Name	Reference
IBSRn	Bus Status Register	See 4.1
IBCRn	Bus Control Register	See 4.2
ITBAn	Ten Bit Slave Address Register	See 4.3
ITMKn	Ten Bit Slave Address Mask Register	See 4.4
ISBAn	Seven Bit Slave Address Register	See 4.5
ISMKn	Seven Bit Slave Address Mask Register	See 4.6
IDARn	Data Register	See 4.7
ICCRn	Clock Control Register	See 4.8

4.1. Bus Status Register (IBSRn)

The bus status register (IBSRn) has the following functions:
- Bus busy detection
- Repeated start condition detection
- Arbitration loss detection
- Acknowledge detection
- Data transfer direction indication
- Addressing detection as slave
- General call address detection
- Address data detection

■ Bus Status Register (IBSRn)

This register is read-only, all bits are controlled by the hardware. All bits are cleared if the interface is not enabled (EN = "0" in ICCRn).

IBSRn

bit	7	6	5	4	3	2	1	0
	BB	RSC	AL	LRB	TRX	AAS	GCA	ADT
Attribute	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

[bit7] BB: Bus Busy Bit

This bit indicates the status of the I^2C bus.

Bit	Description
0	Stop condition detected (bus idle)
1	Start condition detected (bus in use)

This bit is set to "1" if a start condition is detected. It is reset upon a stop condition.

[bit6] RSC: Repeated Start Condition Bit

This bit indicates detection of a repeated start condition.

Bit	Description
0	Repeated start condition not detected
1	Bus in use, repeated start condition detected

This bit is cleared at the end of an address data transfer (ADT="0") or detection of a stop condition.

[bit5] AL: Arbitration Loss Bit

This bit indicates an arbitration loss.

Bit	Description
0	No arbitration loss detected
1	Arbitration loss detected during master sending

This bit is cleared by writing "0" to the INT bit or by writing "1" to the MSS bit in the IBCR register.
An arbitration loss occurs if:

- the data sent does not match the data read on the SDA line at the rising SCL edge
- a repeated start condition is generated by another master in the first bit of a data byte
- The interface could not generate a start or stop condition because signal transition caused from "1" to "0" by a certain external condition observed at the SCL line.

[bit4] LRB: Last Received Bit

This bit is used to indicate the acknowledge from the receiving device.

Bit	Description
0	Receiver did acknowledge
1	Receiver did not acknowledge

It is changed by the hardware upon reception of bit9 (acknowledge bit) and is also cleared by a start or stop condition.

[bit3] TRX: Transferring Data Bit

This bit indicates data transmission operation.

Bit	Description
0	Not transmitting data
1	Transmitting data

It is set to "1" if:

- a start condition was generated in master mode
- the first byte has been transferred, and read access is executed in slave mode, or the data is being sent in master mode

It is set to "0" if:

- the bus is idle (BB="0")
- an arbitration loss occurred
- "1" is written to the SCC bit during master interrupt (MSS="1" and INT="1")
- the MSS bit being cleared during master interrupt (MSS="1" and INT="1")
- the interface is in slave mode and the last transferred byte was not acknowledged
- the interface is in slave mode and it is receiving data
- the interface is in master mode and is reading data from a slave

[bit2] AAS: Addressed as Slave Bit

This bit indicates detection of a slave addressing.

Bit	Description
0	not addressed as slave
1	Addressed as slave

This bit is cleared by a (repeated-) start or stop condition. It is set if the interface detects its seven and/or ten bit slave address.

[bit1] GCA: General Call Address Bit

This bit indicates detection of a general call address (0x00).

Bit	Description
0	General call address not received as slave
1	General call address received as slave

This bit is cleared by a (repeated-) start or stop condition.

[bit0] ADT: Address Data Transfer Bit

This bit indicates the detection of an address data transfer.

Bit	Description
0	Incoming data is not address data (or bus not in use)
1	Incoming data is address data

This bit is set to "1" by a start condition. It is cleared after the second byte if a ten bit slave address header with write access is detected, else it is cleared after the first byte.
"After the first or second byte" means the following:

- "0" is written to the MSS bit during a master interrupt (MSS="1" and INT="1" in IBCR)
- "1" is written to the SCC bit during a master interrupt (MSS="1" and INT="1" in IBCR)
- the INT bit is being cleared
- the beginning of every byte transfer if the interface is not involved in the current transfer as master or slave

4.2. Bus Control Register (IBCRn)

The Bus Control Register (IBCRn) has the following functions:
- Interrupt enabling flags
- Interrupt generation flag
- Bus error detection flag
- Repeated start condition generation
- Master / slave mode selection
- General call acknowledge generation enabling
- Data byte acknowledge generation enabling

■ Bus Control Register (IBCRn)

Write access to this register should only occur while the INT="1" or if a transfer is to be started. The user should not write to this register during an ongoing transfer since changes to the ACK or GCAA bits could result in bus errors. All bits in this register except the BER and the BEIE bit are cleared if the interface is not enabled (EN="0" in ICCR).

IBCRn

bit	15	14	13	12	11	10	9	8
	BER	BEIE	SCC	MSS	ACK	GCAA	INTE	INT
Attribute	R/W	R/W	W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

[bit15] BER: Bus Error Bit

This bit is the bus error interrupt flag. It is set by the hardware and cleared by the user.
It always reads "1" in a Read-Modify-Write access.

Bit	Description	
	Read	Write
0	No bus error detected	Clear bus error interrupt flag
1	One of the error conditions described below detected.	No effect

If this bit is set, the EN bit of the ICCR register is cleared. The I^2C interface is stopped, and data transfer is halted. All the bits of the IBSR and IBCR registers except BER and BEIE are cleared. The BER bit must be cleared before the interface is enabled again.
This bit is set to "1" if:

- an activation or stop condition is detected at an illegal location during the transfer of the address data or bit2 to bit9 (acknowledgement bits).
- the 10-bit address header with 10-bit read access is received before the 10-bit write access is performed.

When the interface is enabled during the transfer, the detection of the first two conditions shown above is enabled after the first stop condition is received, in order to prevent an incorrect bus error report from being issued.

[bit14] BEIE: Bus Error Interrupt Enable Bit

This bit enables the bus error interrupt. It only can be changed by the user.

Bit	Description
0	Bus error interrupt disabled
1	Bus error interrupt enabled

Setting this bit to "1" enables MCU interrupt generation when the BER bit is set to "1".

[bit13] SCC: Start Condition Continue Bit

This bit is used to generate a repeated start condition. It is write only - it always reads "0".

Bit	Description
0	No effect:
1	Generate repeated start condition during master transfer

A repeated start condition is generated if a "1" is written to this bit while an interrupt in master mode (MSS="1" and INT="1") and the INT bit is cleared automatically.

[bit12] MSS: Master Slave Select Bit

This is the master/slave mode selection bit. It can only be set by the user, but it can be cleared by the user and the hardware.

Bit	Description
0	Go to slave mode
1	Go to master mode, generate start condition and send address data byte in IDAR register. It is cleared if an arbitration loss event occurs during master sending.

If a "0" is written to it during a master interrupt (MSS="1" and INT="1"), the INT bit is cleared automatically, a stop condition will be generated and the data transfer ends. Note that the MSS bit is reset immediately the generation of the stop condition can be checked by polling the BB bit in the IBSR register.
If a "1" is written to it while the bus is idle (MSS="0" and BB="0"), a start condition is generated and the contents of the IDAR register (which should be address data) is sent.
If a "1" is written to the MSS bit while the bus is in use (BB="1" and TRX="0" in IBSR; MSS="0" in IBCR), the interface waits until the bus is free and then starts sending.
If the interface is addressed as slave with write access (data reception) in the meantime, it will start sending after the transfer ended and the bus is free again. If the interface is sending data as slave in the meantime (AAS="1" and TRX="1" in IBSR), it will not start sending data if the bus is free again. It is important to check whether the interface was addressed as slave (AAS="1" in IBSR), sent the data byte successfully (MSS="1" in IBCR) or failed to send the data byte (AL="1" in IBSR) at the next interrupt!

[bit11] ACK: Acknowledge Bit

This is the acknowledge generation on data byte reception enable bit. It can only be changed by the user.

Bit	Description
0	No Acknowledge on data byte reception
1	Acknowledge on data byte reception

This bit is not valid when receiving address bytes in slave mode - if the interface detects its 7 or 10-bit slave address, it will acknowledge if the corresponding enable bit (ENTB in ITMK or ENSB in ISMK) is set. Write access to this bit should occur during an interrupt (INT="1") or if the bus is idle (BB="0" in the IBSR register). In addition the interface must be enabled (EN="1" in ICCR) and there has to be no bus error (BER="0" in IBCR).

[bit10] GCAA: General Call Address Acknowledge Bit

This bit enables acknowledge generation when a general call address is received. It only can be changed by the user.

Bit	Description
0	No acknowledge on general call address byte reception
1	Acknowledge on general call address byte reception

Write access to this bit should occur during an interrupt (INT="1") or if the bus is idle (BB="0" in IBSR register), write access to this bit is only possible if the interface is enabled (EN="1" in ICCR) and if there is no bus error (BER="0" in IBCR).

[bit9] INTE: Interrupt Enable Bit

This bit enables the MCU interrupt generation. It only can be changed by the user.

Bit	Description
0	Interrupt disabled
1	Interrupt enabled

Setting this bit to "1" enables MCU interrupt generation when the INT bit is set to "1" (by the hardware).

[bit8] INT: Interrupt Flag Bit

This bit is the transfer end interrupt request flag. It is changed by the hardware and can be cleared by the user. It always reads "1" in a Read-Modify-Write access.

Bit	Description	
	Read	Write
0	Transfer not ended or not involved in current transfer or bus is idle	Clear transfer end interrupt request flag
1	Set at the end of a 1-byte data transfer or reception including the acknowledge bit * under the following conditions:	No effect

* under the following conditions:
Device is bus master.
Device is addressed as slave.
General call address received.
Arbitration loss occurred.
Set at the end of an address data reception (after first byte if seven bit address received, after second byte if ten bit address received) including the acknowledge bit if the device is addressed as slave.

While this bit is "1" the SCL line will hold an "L" level signal. Writing "0" to this bit clears the setting, releases the SCL line, and executes transfer of the next byte or a repeated start or stop condition is generated. Additionally, this bit is cleared if a "1" is written to the SCC bit or the MSS bit is being cleared.

■ SCC, MSS and INT Bit Competition

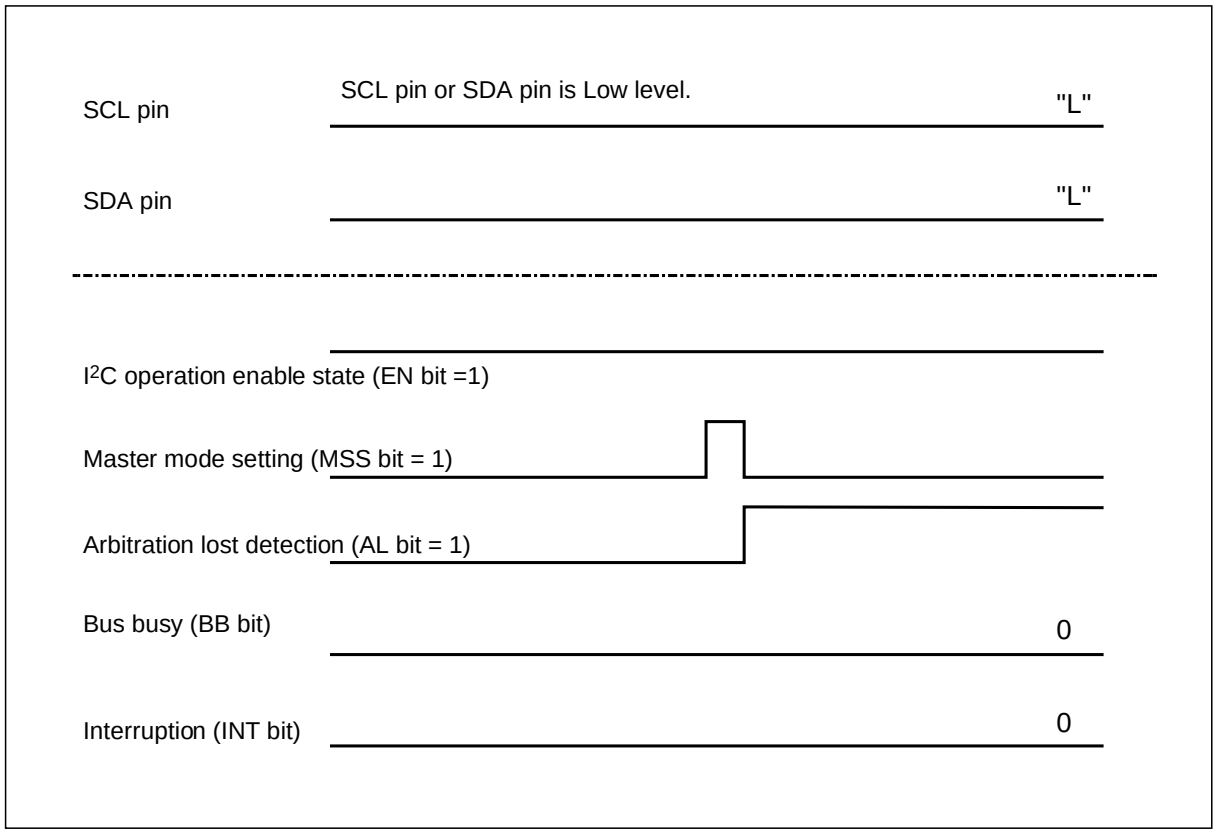
Simultaneously writing to the SCC, MSS and INT bits causes a competition to transfer the next byte, to generate a repeated start condition or to generate a stop condition. In these cases the order of priority is as follows:

- Next byte transfer and stop condition generation. When "0" is written to the INT bit and "0" is written to the MSS bit, the MSS bit takes priority and a stop condition is generated.
- Next-byte transfer and re-activation condition generation: When "0" is written to the INT bit and "1" is written to the SCC bit, writing of the SCC bit has precedence. A re-activation condition is generated, and the content of the IDAR register is transmitted.
- Repeated start condition generation and stop condition generation. When a "1" is written to the SCC bit and "0" to the MSS bit, the MSS bit clearing takes priority. A stop condition is generated and the interface enters slave mode. If specific conditions are met, the AL (arbitration lost) bit does not set the INT (interrupt) bit. Those conditions are presented in Figure 4-1 Diagram of Timing at which an Interrupt upon Detection of "AL bit = 1" does not Occur and Figure 4-2 Diagram of Timing at which an Interrupt upon Detection of "AL bit = 1" does not Occur.

● Case 1: When SCL and SDA Signals are Kept at "L";

The AL bit is set immediately after the MSS bit set to "1" while the BB bit is indicating "0" (no start condition is detected). However the AL bit will not set the INT bit under this circumstances.

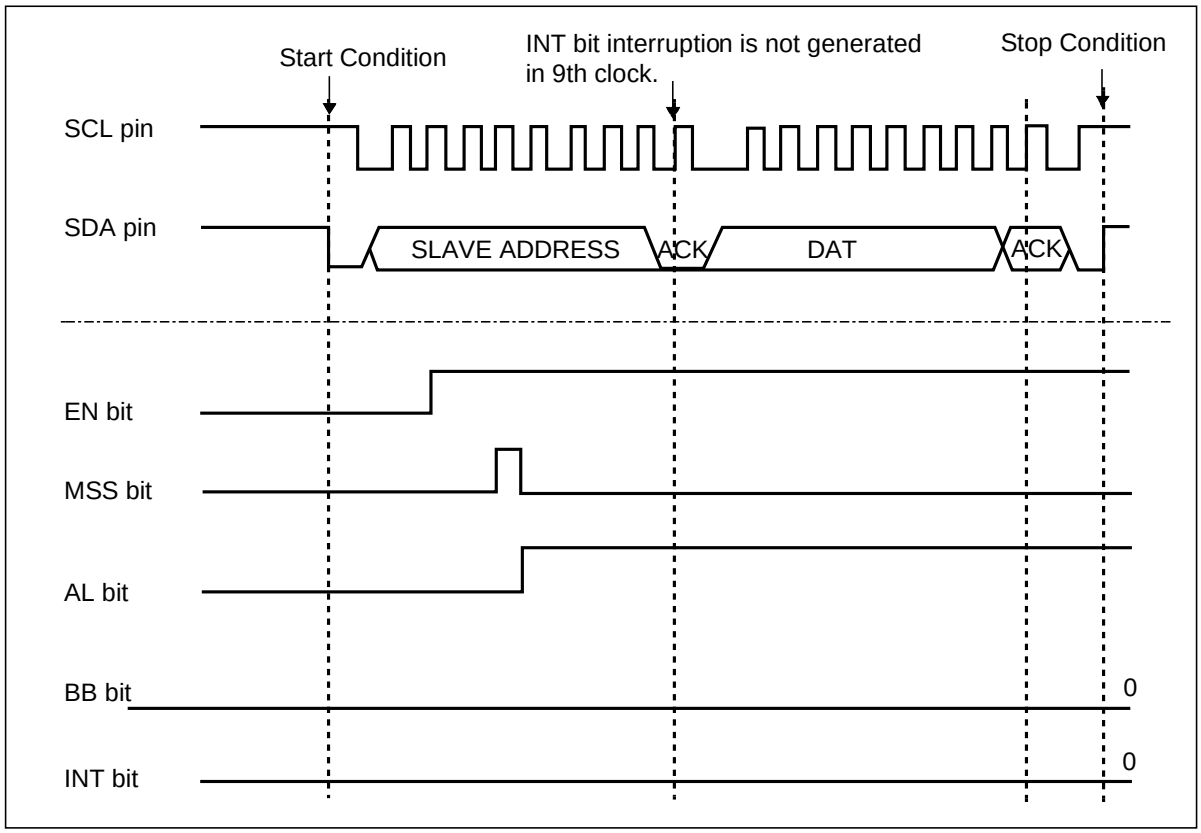
Figure 4-1 Diagram of Timing at which an Interrupt upon Detection of "AL bit = 1" does not Occur



● Case 2: When I²C Interface is Enabled while There is Ongoing Communication with Another Bus Master;

The interface participates in the I²C bus while the bus is occupied with ongoing communication if the EN bit is set from "0" to "1". In this case the BB bit stays "0" (no start condition is detected) and setting the MSS bit to "1" results in the AL bit indicating "1". However the AL bit will not set the INT bit under this circumstance.

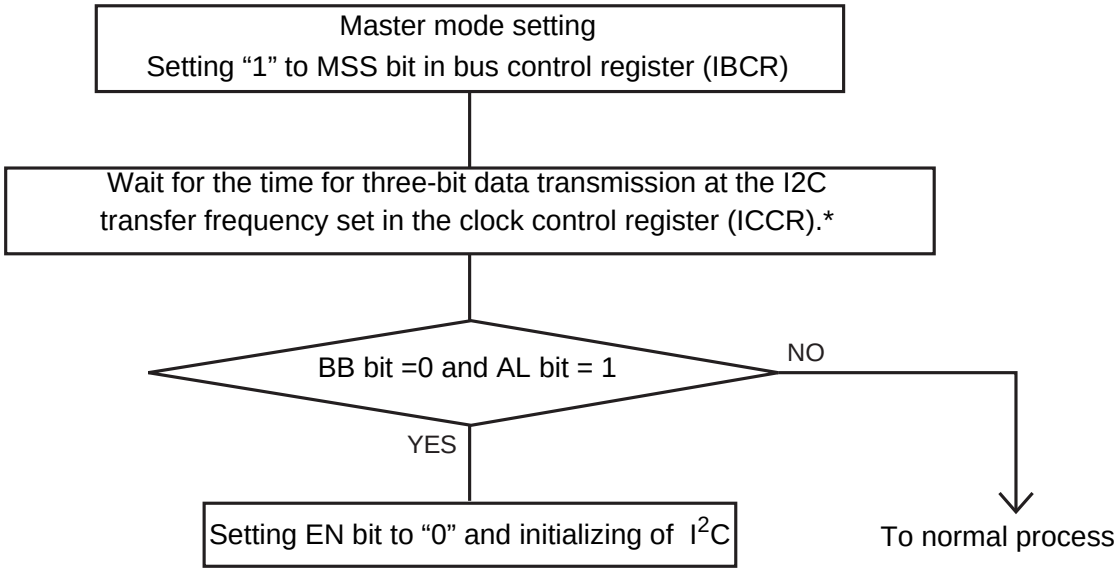
Figure 4-2 Diagram of Timing at which an Interrupt upon Detection of "AL bit = 1" does not Occur



If a symptom as described above can occur, follow the procedure below for software processing.

1. Execute the instruction that generates a start condition (set the MSS bit to 1).
2. Use, for example, the timer function to wait for the time for three - bit data transmission at the I²C transfer frequency set in the ICCR register.*
   Example: Time for three-bit data transmission at an I²C transfer frequency of 100 kHz = $(1 / (100 \times 10^3)) \times 3 = 30$
3. Check the AL and BB bits in the IBSR register and, if the AL and BB bits are 1 and 0, respectively, set the EN bit in the ICCR register to 0 to initialize I²C. When the AL and BB bits are not so, perform normal processing.

A sample flow is given below.

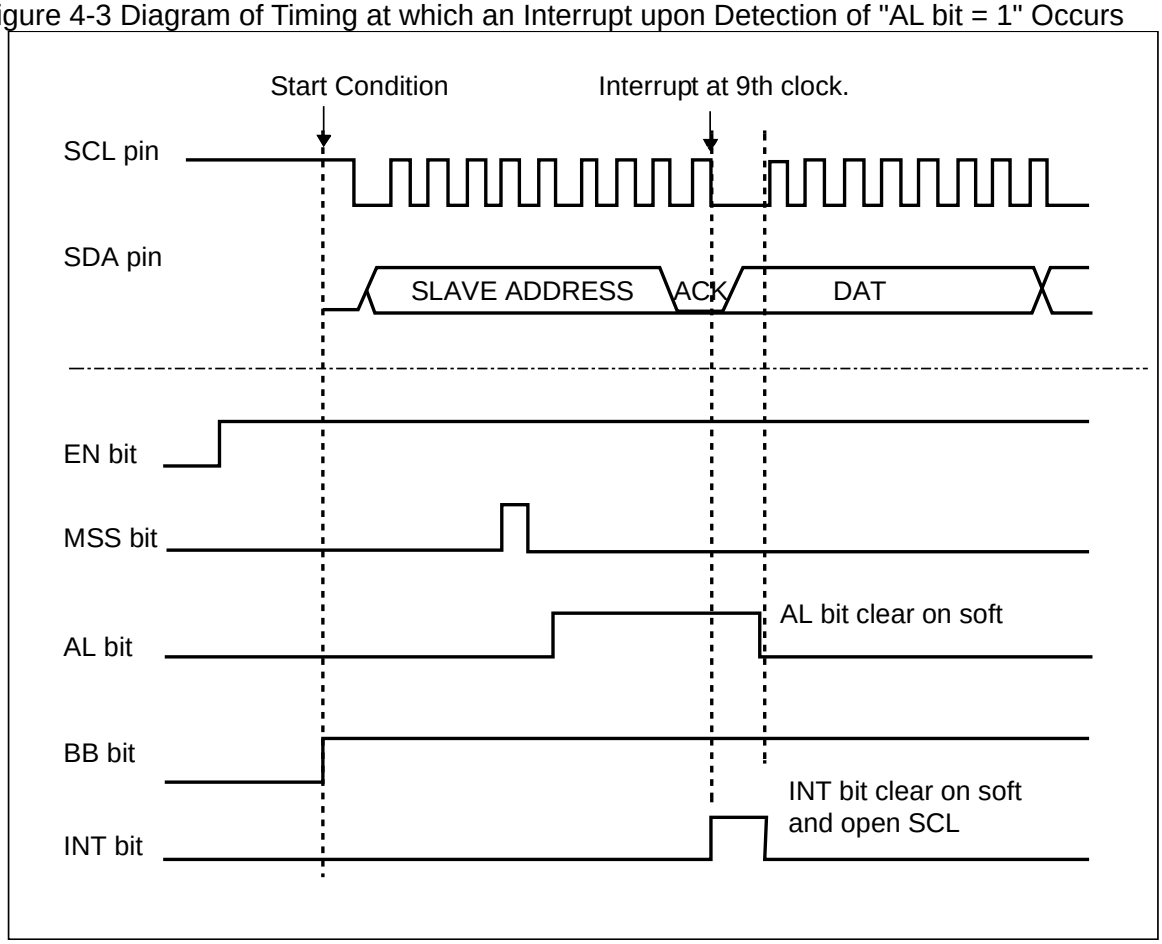


*: Arbitration lost is detected within 3-bit time after setting the MSS bit to "1".

- **Example of Occurrence of an Interrupt (INT bit = 1) upon Detection of "AL Bit = 1"**

When an instruction which generates a start condition is executed (setting the MSS bit to 1) with "bus busy" detected (BB bit = 1) and arbitration is lost, the INT bit interrupt occurs upon detection of "AL bit = 1".

Figure 4-3 Diagram of Timing at which an Interrupt upon Detection of "AL bit = 1" Occurs



4.3. Ten Bit Slave Address Register (ITBAn)

The Ten Bit Slave Address Register (ITBAn) designates the ten bit slave address.

■ Ten Bit Slave Address Register (ITBAn)

Write access to this register is only possible if the interface is disabled (EN="0" in ICCR).

ITBAHn

bit	15	14	13	12	11	10	9	8
	-	-	-	-	-	-	TA9	TA8
Attribute	-	-	-	-	-	-	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

ITBALn

bit	7	6	5	4	3	2	1	0
	TA7	TA6	TA5	TA4	TA3	TA2	TA1	TA0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

[bit15 to bit10] -: Undefined

These bits always return "0".

[bit9 to bit0] TA9 to TA0: Ten Bit Slave Address

When address data is received in slave mode, it is compared to the ITBA register if the ten bit address is enabled (ENTB="1" in the ITMK register). An acknowledge is sent to the master after reception of a ten bit address header with write access[*1]. Then, the second incoming byte is compared to the TBAL register. If a match is detected, an acknowledge signal is sent to the master device and the AAS bit is set.
Additionally, the interface acknowledges upon the reception of a ten bit header with read access[*2] after a repeated start condition.
All bits of the slave address may be masked using the ITMK register. The received ten bit slave address is written back to the ITBA register, it is only valid while the AAS bit in the IBSR register is "1".

*1: A ten bit header (write access) consists of the following bit sequence: 11110, TA9, TA8, 0.

*2: A ten bit header (read access) consists of the following bit sequence: 11110, TA9, TA8, 1.

4.4. Ten Bit Slave Address Mask Register (ITMKn)

The Ten Bit Slave Address Mask Register (ITMKn) contains the ten bit slave address mask and the ten bit slave address enable bit.

■ Ten Bit Address Mask Register (ITMKn)

ITMKHn								
bit	15	14	13	12	11	10	9	8
	ENTB	RAL	-	-	-	-	TM9	TM8
Attribute	R/W	R	-	-	-	-	R/W	R/W
Initial value	0	0	1	1	1	1	1	1

ITMKLn								
bit	7	6	5	4	3	2	1	0
	TM7	TM6	TM5	TM4	TM3	TM2	TM1	TM0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	1	1	1	1	1	1	1	1

[bit15] ENTB: Enable Ten Bit Slave Address Bit

This bit enables the ten bit slave address (and the acknowledging upon its reception). Write access to this bit is only possible if the interface is disabled (EN="0" in ICCR).

Bit	Description
0	Ten bit address disabled
1	Ten bit address enabled

[bit14] RAL: Received Slave Address Length Bit

This bit indicates whether the interface was addressed as a seven or ten bit slave. It is read-only.

Bit	Description
0	Addressed as seven bit slave
1	Addressed as ten bit slave

This bit can be used to determine whether the interface was addressed as a seven or ten bit slave if both slave addresses are enabled (ENTB="1" and ENSB="1"). Its contents is only valid if the AAS bit in the IBSR register is "1". This bit is also reset if the interface is disabled (EN="0" in ICCR).

[bit13 to bit10] -: Undefined

These bits always return "1" during reading.

[bit9 to bit0] TM9 to TM0: Ten Bit Slave Address Mask Bits

This register is used to mask the ten bit slave address of the interface. Write access to these bits is only possible if the interface is disabled (EN="0" in ICCR).

Bit	Description
0	Bit is not used in slave address comparison
1	Bit is used in slave address comparison

This can be used to make the interface acknowledge on multiple ten bit slave addresses. Only the bits set to "1" in this register are used in the ten bit slave address comparison. The received slave address is written back to the ITBA register and thus may be determined by reading the ITBA register if the AAS bit in the IBSR register is "1".

Note:

If the address mask is changed after the interface has been enabled, the slave address should also be set again since it could have been overwritten by a previously received slave address.

4.5. Seven Bit Slave Address Register (ISBAn)

The Seven Bit Slave Address Register (ISBAn) contains the seven bit slave address.

■ Seven Bit Slave Address Register (ISBAn)

Write access to this register is only possible if the interface is disabled (EN="0" in ICCR).

ISBAn								
bit	7	6	5	4	3	2	1	0
	-	SA6	SA5	SA4	SA3	SA2	SA1	SA0
Attribute	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

[bit7] -: Undefined

This bit always returns "0" during reading.

[bit6 to bit0] SA6 to SA0: Seven Bit Slave Address Bits

When address data is received in slave mode, it is compared to the ISBA register if the seven bit address is enabled (ENSB="1" in the ISMK register). If a match is detected, an acknowledge signal is sent to the master device and the AAS bit is set.

All bits of the slave address may be masked using the ISMK register. The received seven bit slave address is written back to the ISBA register, it is only valid while the AAS bit in the IBSR register is "1".

The interface does not compare the contents of this register to the incoming data if a ten bit header or a general call is received.

4.6. Seven Bit Slave Address Mask Register (ISMKn)

The Seven Bit Slave Address Mask Register (ISMKn) contains the seven bit slave address mask and the seven bit slave address enable bit.

■ Seven Bit Slave Address Mask Register (ISMKn)

This register contains the seven bit slave address mask and the seven bit mode enable bit. Write access to this register is only possible if the interface is disabled (EN="0" in ICCR).

ISMKn								
bit	15	14	13	12	11	10	9	8
	ENSB	SM6	SM5	SM4	SM3	SM2	SM1	SM0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	1	1	1	1	1	1	1

[bit15] ENSB: Enable Seven Bit Slave Address Bit

This bit enables the seven bit slave address (and the acknowledging upon its reception).

Bit	Description
0	Seven bit slave address disabled
1	Seven bit slave address enabled

[bit14 to bit8] SM6 to SM0: Seven Bit Slave Address Mask Bits

This register is used to mask the seven bit slave address of the interface.

Bit	Description
0	Bit is not used in slave address comparison.
1	Bit is used in slave address comparison.

This can be used to make the interface acknowledge on multiple seven bit slave addresses. Only the bits set to "1" in this register are used in the seven bit slave address comparison. The received slave address is written back to the ISBA register and may thus may be determined by reading the ISBA register if the AAS bit in the IBSR register is "1".

Note:

If the address mask is changed after the interface has been enabled, the slave address should also be set again since it could have been overwritten by a previously received slave address.

4.7. Data Register (IDARn)

The Data Register for the 400 kHz I^2C Interface (IDARn) is used to send and receive data by the CPU.

■ Data Register (IDARn)

IDARn								
bit	7	6	5	4	3	2	1	0
	D7	D6	D5	D4	D3	D2	D1	D0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

[bit7 to bit0] D7 to D0: Data Bits

The data register is used in serial data transfer, and transfers data MSB-first. This register is double buffered on the write side, so that when the bus is in use (BB="1"), write data can be loaded to the register for serial transfer. The data byte is loaded into the internal transfer register if the INT bit in the IBCR register is being cleared or the bus is idle (BB="0" in IBSR). In a read access, the internal register is read directly, therefore received data values in this register are only valid if INT="1" in the IBCR register.

4.8. Clock Control Register (ICCRn)

The Clock Control Register (ICCRn) has the following functions:
- Enable test mode
- Enable I/O pad noise filters
- Enable I^2C interface operation
- Setting the serial clock frequency

■ Clock Control Register (ICCRn)

ICCRn								
bit	15	14	13	12	11	10	9	8
	-	NSF	EN	CS4	CS3	CS2	CS1	CS0
Attribute	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	1	1	1	1	1

[bit15] -: Undefined
This bit always returns "0" during reading.

[bit14] NSF: I/O Pad Noise Filter Enable Bit
This bit enables the noise filters built into the SDA and SCL I/O pads.
The noise filter will suppress single spikes with a pulse width of 0 ns (minimum) and between 1 and 1.5 cycles of the peripheral clock CLKP1. The maximum depends on the phase relationship between I^2C signals (SDA, SCL) and peripheral clock CLKP1.
It should be set to "1" if the interface is transmitting or receiving at data rates above 100 KBit.

[bit13] EN: Enable Bit
This bit enables the I^2C interface operation. It can only be set by the user but may be cleared by the user and the hardware.

Bit	Description
0	Interface disabled
1	Interface enabled

When this bit is set to "0" all bits in the IBSR register and IBCR register (except the BER and BEIE bits) are cleared and the module is disabled and the I^2C lines are left open. It is cleared by the hardware if a bus error occurs (BER="1" in IBCR).

Note:

The interface immediately stops transmitting or receiving if it is being disabled. This might leave the I^2C bus in an undesired state.

[bit12 to bit8] CS4 to CS0: Clock Prescaler Bits

These bits select the serial bit rate. They can only be changed if the interface is disabled (EN="0") or the EN bit is being cleared in the same write access.

n	CS4	CS3	CS2	CS1	CS0	
1	0	0	0	0	1	Bitrate: ϕ / 28(+1)
2	0	0	0	1	0	Bitrate: ϕ / 40(+1)
3	0	0	0	1	1	Bitrate: ϕ / 52(+1)
4	0	0	1	0	0	Bitrate: ϕ / 64(+1)
...						
31	1	1	1	1	1	Bitrate: ϕ / 400(+1)

(+1) means: Add 1 to divisor, if noise filter is enabled

■ Clock Prescaler Settings

The calculation formula for CS0 to CS4 is determined as follows:

$$\text{Bitrate} = \frac{\phi}{n*12 + 16} \quad n>0 \quad \phi\text{: Peripheral clock CLKP1, Noise filter disabled}$$

$$\text{Bitrate} = \frac{\phi}{n*12 + 17} \quad n>0 \quad \phi\text{: Peripheral clock CLKP1, Noise filter enabled}$$

Table 4-1 Prescaler Settings

n	CS4	CS3	CS2	CS1	CS0
1	0	0	0	0	1
2	0	0	0	1	0
3	0	0	0	1	1
...					
31	1	1	1	1	1

Note: Do not use n=0 prescaler setting, it violates SDA/SCL timings.

■ Common Peripheral Clock CLKP1 Frequencies

Table 4-2 shows the most common peripheral clock CLKP1 frequencies with their prescaler settings and the resulting sending bit rate:

Table 4-2 Common Peripheral Clock CLKP1 Frequencies

CLKP1 frequency [MHz]	100 Kbit (Noise filter disabled)		400 Kbit (Noise filter enabled)	
	n	Bit rate [Kbit]	n	Bit rate [Kbit]
24	19	98	4	369
20	16	96	3	377
16	12	100	2	390
40/3 = 13.3	10	98	2	325
12	9	96	2	292
64/6 = 10.6	8	94	1	367
10	7	100	1	344
8	6	90	1	275

CHAPTER: CAN CONTROLLER

This chapter explains the functions and operations of the CAN controller.

Management Code: 96F3CAN-E01.0

1. Overview

The CAN performs communication according to the CAN protocol version 2.0 part A and B. The bit rate can be programmed to values up to 1MBit/s. For the connection to the physical layer additional transceiver hardware is required.

■ Features

The CAN implements the following features:

- Supports CAN protocol version 2.0 part A and B
- Bit rates up to 1 MBit/s
- 32 (up to 128) Message Objects
- Each Message Object has its own identifier mask
- Programmable FIFO mode (concatenation of Message Objects)
- Maskable interrupt
- Disabled Automatic Retransmission mode for Time Triggered CAN applications
- Programmable loop-back mode for self-test operation

Note:

In this chapter, the suffix "n" in the register names denotes the CAN controller number.

■ Abbreviations

This chapter uses the following terms and abbreviations:

Term	Meaning
CAN	Controller Area Network
BSP	Bit Stream Processor
BTL	Bit Timing Logic
CRC	Cyclic Redundancy Check
DLC	Data Length Code
EML	Error Management Logic
FSM	Finite State Machine
TTCAN	Time Triggered CAN

2. Functional Description

This chapter provides an overview of the CAN module's operating modes and how to use them.

■ Software Initialisation

The software initialization is started by setting the bit INIT in the CAN Control Register CTRLRLn, either by software or by a hardware reset, or by going Bus Off.
While INIT is set, all message transfer from and to the CAN bus is stopped, the status of the CAN bus output CAN_TX is recessive (HIGH). The counters of the EML are unchanged. Setting INIT does not change any configuration register.
To initialize the CAN Controller, the CPU has to set up the Bit Timing Register BTRn and each Message Object. If a Message Object is not needed, it is sufficient to set it's MSGVAL bit to not valid. Otherwise, the whole Message Object has to be initialized.
Access to the Bit Timing Register BTRn and to the BRP Extension Register BRPERn for the configuration of the bit timing is enabled when both bits INIT and CCE in the CAN Control Register CTRLRLn are set.
Resetting INIT (by CPU only) finishes the software initialisation. Afterwards the Bit Stream Processor BSP synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits (= Bus Idle) before it can take part in bus activities and starts the message transfer.
The initialization of the Message Objects is independent of INIT and can be done on the fly, but the Message Objects should all be configured to particular identifiers or set to not valid before the BSP starts the message transfer.
To change the configuration of a Message Object during normal operation, the CPU has to start by setting MSGVAL bit to not valid. When the configuration is completed, MSGVAL bit is set to valid again.

■ CAN Message Transfer

Once the CAN is initialized and INIT is reset to zero, the CAN's CAN Core synchronizes itself to the CAN bus and starts the message transfer.
Received messages are stored into their appropriate Message Objects if they pass the Message Handler's acceptance filtering. The whole message including all arbitration bits, DLC and eight data bytes is stored into the Message Object. If the Identifier Mask is used, the arbitration bits, which are masked to "don't care", may be overwritten in the Message Object.
The CPU may read or write each message any time via the Interface Registers, the Message Handler guarantees data consistency in case of concurrent accesses.
Messages to be transmitted are updated by the CPU. If a permanent Message Object (arbitration and control bits set up during configuration) exists for the message, only the data bytes are updated and then TXRQST bit with NEWDAT bit are set to start the transmission. If several transmit messages are assigned to the same Message Object (when the number of Message Objects is not sufficient), the whole Message Object has to be configured before the transmission of this message is requested.
The transmission of any number of Message Objects may be requested at the same time, they are transmitted subsequently according to their internal priority. Messages may be updated or set to not valid any time, even when their requested transmission is still pending. The old data will be discarded when a message is updated before its pending transmission has started. Depending on the configuration of the Message Object, the transmission of a message may be requested autonomously by the reception of a remote frame with a matching identifier.

■ Disabled Automatic Retransmission

According to the CAN Specification (see ISO11898, 6.3.3 Recovery Management), the CAN provides means for automatic retransmission of frames that have lost arbitration or that have been disturbed by errors during transmission. The frame transmission service will not be confirmed to the user before the transmission is successfully completed. By default, this means that automatic retransmission is enabled. It can be disabled to enable the CAN to work within a Time Triggered CAN (TTCAN, see ISO11898-1) environment. The Disabled Automatic Retransmission mode is enabled by programming bit DAR in the CAN Control Register CTRLRn to "1". In this operation mode the programmer has to consider the different behaviour of bits TXRQST and NEWDAT in the Control Registers of the Message Buffers:
When a transmission starts bit TXRQST of the respective Message Buffer is reset, while bit NEWDAT remains set.
When the transmission completed successfully bit NEWDAT is reset.
When a transmission failed (lost arbitration or error) bit NEWDAT remains set. To restart the transmission the CPU has to set TXRQST back to one.
When the host requests the transmission of several messages at the same time, only two of these messages will be transmitted. For all other requested transmit messages, the TXRQST bits will be reset, but no transmission will be started, NEWDAT and INTPND will be left unchanged. For the two messages that are transmitted, the TXRQST and NEWDAT bits will be reset and, if enabled by TXIE, INTPND will be set.

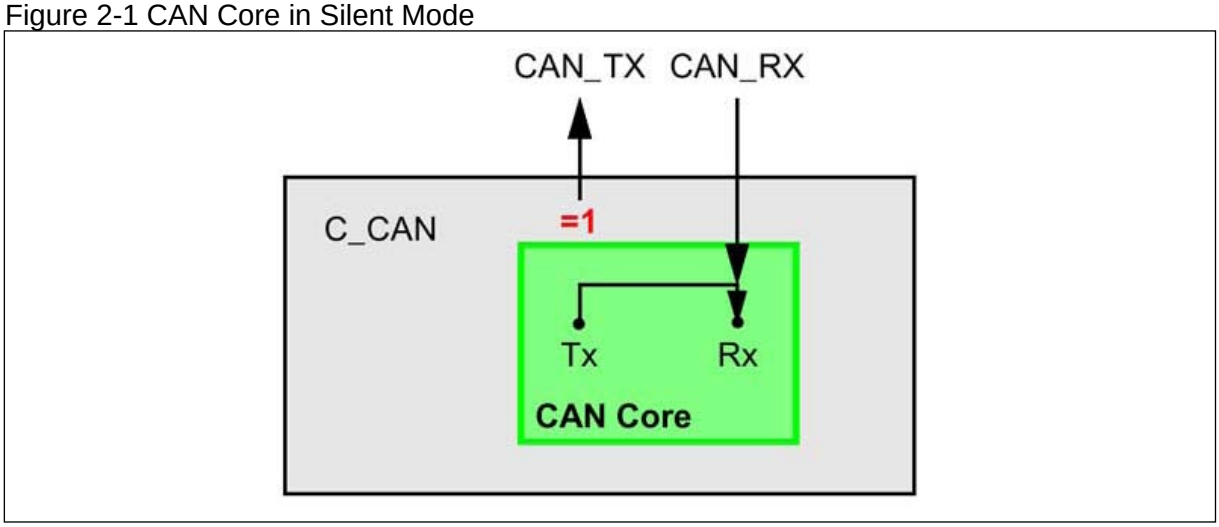
■ Test Mode

The Test Mode is entered by setting bit TEST in the CAN Control Register to "1". In Test Mode the bits TX1, TX0, LBACK, SILENT and BASIC in the Test Register TESTRn are writable. Bit RX monitors the state of pin CAN_RX and therefore is only readable. All Test Register functions are disabled when bit TEST is reset to zero.

■ Silent Mode

The CAN Core can be set in Silent Mode by programming the Test Register TESTRn bit SILENT to "1". In Silent Mode, the CAN is able to receive valid data frames and valid remote frames, but it sends only recessive bits on the CAN bus and it cannot start a transmission. If the CAN Core is required to send a dominant bit (ACK bit, overload flag, active error flag), the bit is rerouted internally so that the CAN Core monitors this dominant bit, although the CAN bus may remain in recessive state. The Silent Mode can be used to analyse the traffic on a CAN bus without affecting it by the transmission of dominant bits (Acknowledge Bits, Error Frames). Figure 2-1 shows the connection of signals CAN_TX and CAN_RX to the CAN Core in Silent Mode.
In ISO 11898-1, the Silent Mode is called the Bus Monitoring Mode.

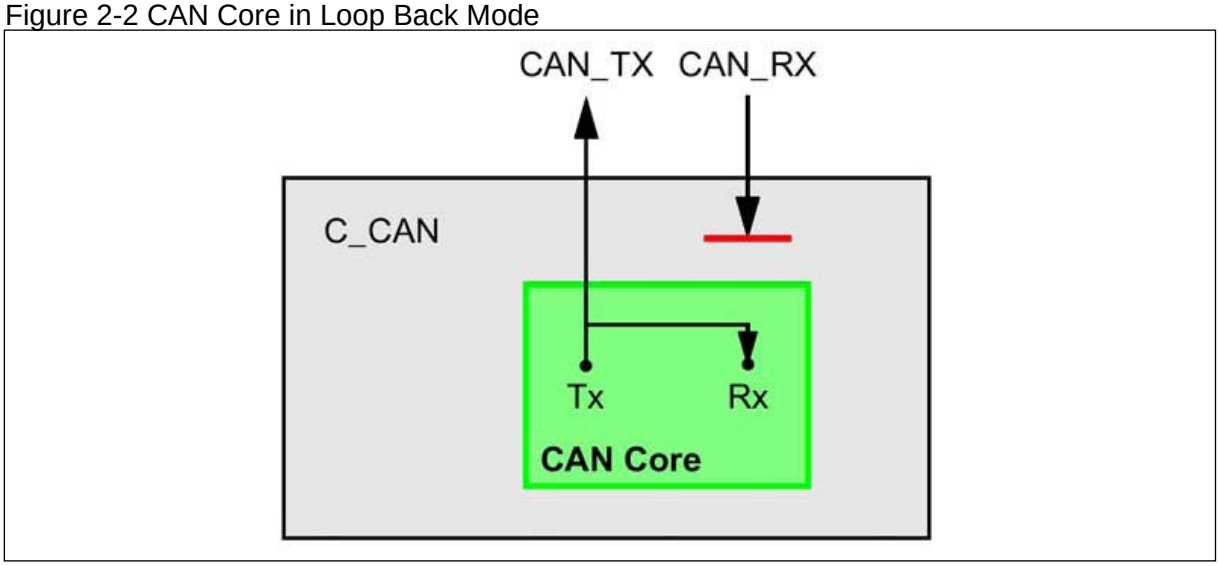
Figure 2-1 CAN Core in Silent Mode



■ Loop Back Mode

The CAN Core can be set in Loop Back Mode by programming the Test Register TESTRn bit LBACK to one. In Loop Back Mode, the CAN Core treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) into a Receive Buffer. Figure 2-2 shows the connection of signals CAN_TX and CAN_RX to the CAN Core in Loop Back Mode.

Figure 2-2 CAN Core in Loop Back Mode

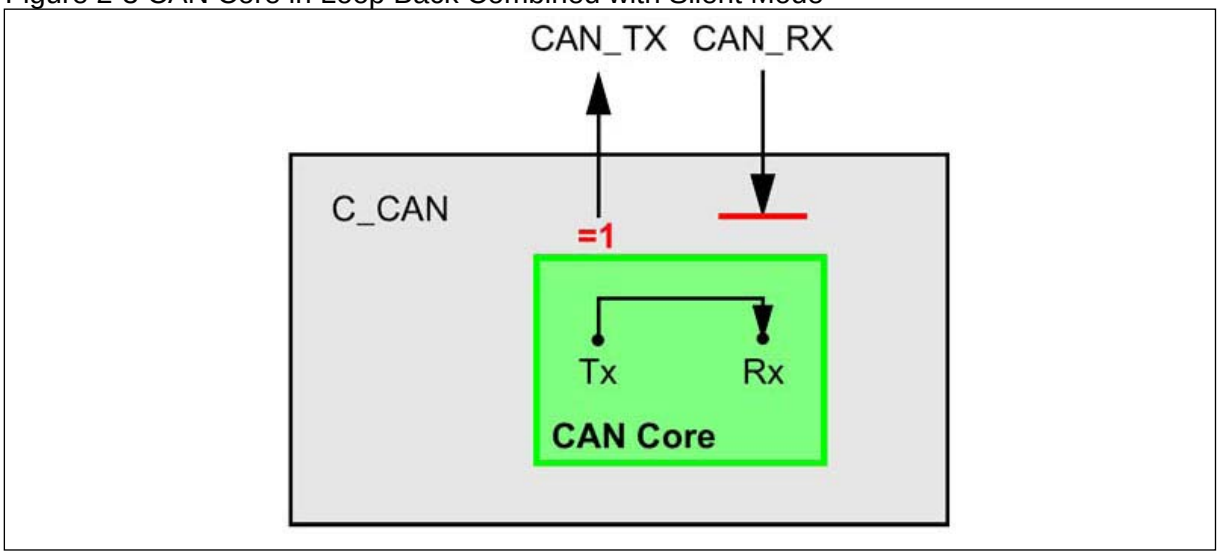


This mode is provided for self-test functions. To be independent from external stimulation, the CAN Core ignores acknowledge errors (recessive bit sampled in the acknowledge slot of a data/remote frame) in Loop Back Mode. In this mode the CAN Core performs an internal feedback from its Tx output to its Rx input. The actual value of the CAN_RX input pin is disregarded by the CAN Core. The transmitted messages can be monitored at the CAN_TX pin.

■ Loop Back Combined with Silent Mode

It is also possible to combine Loop Back Mode and Silent Mode by programming bits LBACK and SILENT to "1" at the same time. This mode can be used for a "Hot Selftest", meaning the CAN can be tested without affecting a running CAN system connected to the pins CAN_TX and CAN_RX. In this mode the CAN_RX pin is disconnected from the CAN Core and the CAN_TX pin is held recessive. Figure 2-3 shows the connection of signals CAN_TX and CAN_RX to the CAN Core in case of the combination of Loop Back Mode with Silent Mode.

Figure 2-3 CAN Core in Loop Back Combined with Silent Mode



■ Basic Mode

The CAN Core can be set in Basic Mode by programming the Test Register TESTRn bit BASIC to "1". In this mode the CAN module runs without the Message RAM.

The IF1 Registers are used as Transmit Buffer. The transmission of the contents of the IF1 Registers is requested by writing the BUSY bit of the IF1 Command Request Register IF1CREQn to "1". The IF1 Registers are locked while the BUSY bit is set. The BUSY bit indicates that the transmission is pending. As soon the CAN bus is idle, the IF1 Registers are loaded into the shift register of the CAN Core and the transmission is started. When the transmission has completed, the BUSY bit is reset and the locked IF1 Registers are released.

A pending transmission can be aborted at any time by resetting the BUSY bit in the IF1 Command Request Register IF1CREQn while the IF1 Registers are locked. If the CPU has reset the BUSY bit, a possible retransmission in case of lost arbitration or in case of an error is disabled.

The IF2 Registers are used as Receive Buffer. After the reception of a message the contents of the shift register is stored into the IF2 Registers, without any acceptance filtering.

Additionally, the actual contents of the shift register can be monitored during the message transfer. Each time a read Message Object is initiated by writing the BUSY bit of the IF2 Command Request Register IF1CREQn to "1", the contents of the shift register are stored into the IF2 Registers.

In Basic Mode the evaluation of all Message Object related control and status bits and of the control bits of the IFx Command Mask Registers IF1CMSKn is turned off. The message number of the Command Request Registers is not evaluated. The NEWDAT and MSGLST bits of the IF2 Message Control Register IF2MCTRn retain their function, DLC3-0 will show the received DLC, the other control bits will be read as "0".

■ Software Control of Pin CAN_TX

Four output functions are available for the CAN transmit pin CAN_TX. Additionally to its default function – the serial data output – it can drive the CAN Sample Point signal to monitor CAN Core's bit timing and it can drive constant dominant or recessive values. The last two functions, combined with the readable CAN receive pin CAN_RX, can be used to check the CAN bus' physical layer.

The output mode of pin CAN_TX is selected by programming the Test Register TESTRn bits TX1 and TX0 as described in "6 CAN Protocol Related Registers".

The three test functions for pin CAN_TX interfere with all CAN protocol functions. CAN_TX must be left in its default function when CAN message transfer or any of the test modes Loop Back Mode, Silent Mode, or Basic Mode are selected.

3. CAN Application

This section describes how to use the CAN module in the application.

■ Management of Message Objects

The configuration of the Message Objects in the Message RAM will (with the exception of the bits MSGVAL, NEWDAT, INTPND, and TXRQST) not be affected by resetting the chip. All the Message Objects must be initialized by the CPU or they must be not valid (MSGVAL = "0") and the bit timing must be configured before the CPU clears the INIT bit in the CAN Control Register.

The configuration of a Message Object is done by programming Mask, Arbitration, Control and Data field of one of the two interface register sets to the desired values. By writing to the corresponding IFx Command Request Register, the IFx Message Buffer Registers are loaded into the addressed Message Object in the Message RAM.

When the INIT bit in the CAN Control Register is cleared, the CAN Protocol Controller state machine of the CAN_Core and the Message Handler State Machine control the CAN's internal data flow. Received messages that pass the acceptance filtering are stored into the Message RAM, messages with pending transmission request are loaded into the CAN_Core's Shift Register and are transmitted via the CAN bus. The CPU reads received messages and updates messages to be transmitted via the IFx Interface Registers. Depending on the configuration, the CPU is interrupted on certain CAN message and CAN error events.

■ Message Handler State Machine

The Message Handler controls the data transfer between the Rx/Tx Shift Register of the CAN Core, the Message RAM and the IFx Registers.

The Message Handler FSM controls the following functions:

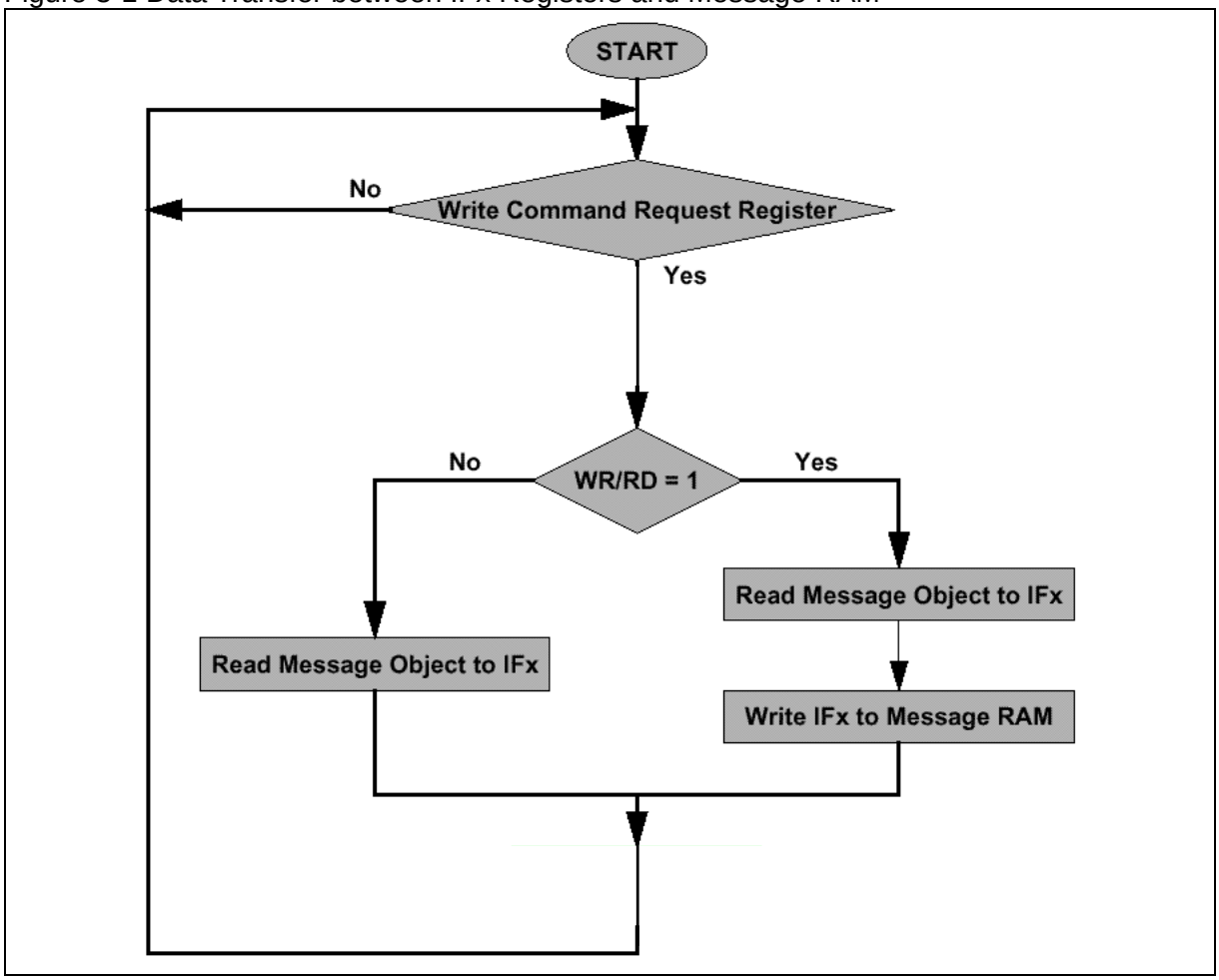
- Data Transfer from IFx Registers to the Message RAM
- Data Transfer from Message RAM to the IFx Registers
- Data Transfer from Shift Register to the Message RAM
- Data Transfer from Message RAM to Shift Register
- Data Transfer from Shift Register to the Acceptance Filtering unit
- Scanning of Message RAM for a matching Message Object
- Handling of TXRQST flags.
- Handling of interrupts.

■ Data Transfer from/to Message RAM

When the CPU initiates a data transfer between the IFx Registers and Message RAM, the Message Handler sets an internal busy signal which delays a consecutive access. After the transfer has completed, the busy signal is set back and the consecutive access is executed.

The respective Command Mask Register specifies whether a complete Message Object or only parts of it will be transferred. Due to the structure of the Message RAM it is not possible to write single bits/bytes of one Message Object, it is always necessary to write a complete Message Object into the Message RAM. Therefore the data transfer from the IFx Registers to the Message RAM requires a read-modify-write cycle. First the parts of the Message Object that are not to be changed are read from the Message RAM and then the complete contents of the Message Buffer Registers are transferred to the Message Object.

Figure 3-1 Data Transfer between IFx Registers and Message RAM



After the partial write of a Message Object, the Message Buffer Registers that are not selected in the Command Mask Register will be set to the actual contents of the selected Message Object.
After the partial read of a Message Object, the Message Buffer Registers that are not selected in the Command Mask Register will be left unchanged.

■ Transmission of Messages

If the shift register of the CAN Core cell is ready for loading and if there is no data transfer between the IFx Registers and Message RAM, the MSGVAL bits in the Message Valid Register and the TXRQST bits in the Transmission Request Register are evaluated. The valid Message Object with the highest priority pending transmission request is loaded into the shift register by the Message Handler and the transmission is started. The Message Object's NEWDAT bit is reset.
After a successful transmission and if no new data was written to the Message Object (NEWDAT = "0") since the start of the transmission, the TXRQST bit will be reset. If TXIE is set, INTPND will be set after a successful transmission. If the CAN has lost the arbitration or if an error occurred during the transmission, the message will be retransmitted as soon as the CAN bus is free again. If meanwhile the transmission of a message with higher priority has been requested, the messages will be transmitted in the order of their priority.

■ Acceptance Filtering of Received Messages

When the arbitration and control field (Identifier + IDE + RTR + DLC) of an incoming message is completely shifted into the Rx/Tx Shift Register of the CAN Core, the Message Handler FSM starts the scanning of the Message RAM for a matching valid Message Object.
To scan the Message RAM for a matching Message Object, the Acceptance Filtering unit is loaded with the arbitration bits from the CAN Core shift register. Then the arbitration and mask fields (including MSGVAL, UMASK, NEWDAT, and EOB) of Message Object 1 are loaded into the Acceptance Filtering unit and compared with the arbitration field from the shift register. This is repeated with each following Message Object until a matching Message Object is found or until the end of the Message RAM is reached.
If a match occurs, the scanning is stopped and the Message Handler FSM proceeds depending on the type of frame (Data Frame or Remote Frame) received.

■ Reception of Data Frame

The Message Handler FSM stores the message from the CAN Core shift register into the respective Message Object in the Message RAM. Not only the data bytes, but all arbitration bits and the Data Length Code are stored into the corresponding Message Object. This is implemented to keep the data bytes connected with the identifier even if arbitration mask registers are used.
The NEWDAT bit is set to indicate that new data (not yet seen by the CPU) has been received. The CPU should reset NEWDAT when it reads the Message Object. If at the time of the reception the NEWDAT bit was already set, MSGLST is set to indicate that the previous data (supposedly not seen by the CPU) is lost.
If the RXIE bit is set, the INTPND bit is set, causing the Interrupt Register to point to this Message Object.
The TXRQST bit of this Message Object is reset to prevent the transmission of a Remote Frame, while the requested Data Frame has just been received.

■ Reception of Remote Frame

When a Remote Frame is received, three different configurations of the matching Message Object have to be considered:

1) DIR = "1" (direction = transmit), RMTEN = "1", UMASK = "1" or "0"

   At the reception of a matching Remote Frame, the TXRQST bit of this Message Object is set. The rest of the Message Object remains unchanged.

2) DIR = "1" (direction = transmit), RMTEN = "0", UMASK = "0"

   At the reception of a matching Remote Frame, the TXRQST bit of this Message Object remains unchanged; the Remote Frame is ignored.

3) DIR = "1" (direction = transmit), RMTEN = "0", UMASK = "1"

   At the reception of a matching Remote Frame, the TXRQST bit of this Message Object is reset. The arbitration and control field (Identifier + IDE + RTR + DLC) from the shift register is stored into the Message Object in the Message RAM and the NEWDAT bit of this Message Object is set. The data field of the Message Object remains unchanged; the Remote Frame is treated similar to a received Data Frame.

■ Receive / Transmit Priority

The receive/transmit priority for the Message Objects is attached to the message number. Message Object 1 has the highest priority, while Message Object 32 (the highest implemented message object number) has the lowest priority. If more than one transmission request is pending, they are serviced due to the priority of the corresponding Message Object.

■ Configuration of a Transmit Object

Figure 3-2 shows how a Transmit Object should be initialised.

Figure 3-2 Initialisation of a Transmit Object

MSGVAL	ARB	DATA	MASK	EOB	DIR	NEWDAT	MSGLST	RXIE	TXIE	INTPND	RMTEN	TXRQST
1	appl.	appl.	appl.	1	1	0	0	0	appl.	0	appl.	0

The Arbitration Registers (ID28-0 and XTD bit) are given by the application. They define the identifier and type of the outgoing message. If an 11-bit Identifier ("Standard Frame") is used, it is programmed to ID28 - ID18, ID17 - ID0 can then be disregarded.
If the TXIE bit is set, the INTPND bit will be set after a successful transmission of the Message Object.
If the RMTEN bit is set, a matching received Remote Frame will cause the TXRQST bit to be set; the Remote Frame will autonomously be answered by a Data Frame.
The Data Registers (DLC3-0, Data0-7) are given by the application, TXRQST and RMTEN may not be set before the data is valid.
The Mask Registers (MSK28-0, UMASK, MXTD, and MDIR bits) may be used (UMASK = "1") to allow groups of Remote Frames with similar identifiers to set the TXRQST bit. For details see "■ Transmission of Messages". Handle with care. The DIR bit should not be masked.

■ Updating a Transmit Object

The CPU may update the data bytes of a Transmit Object any time via the IFx Interface registers, neither MSGVAL nor TXRQST have to be reset before the update.

Note:

When the lowest priority message buffer is used for transmission, setting TXRQST = "0" may cause a delay of transmission, when TXRQST is set to "1" again. Please refer to the note at the description of TXRQST in "9.2 Transmission Request Registers (TREQR1n, TREQR2n)".

Even if only a part of the data bytes are to be updated, all four bytes of the corresponding IFx Data A Register or IFx Data B Register have to be valid before the content of that register is transferred to the Message Object. Either the CPU has to write all four bytes into the IFx Data Register or the Message Object is transferred to the IFx Data Register before the CPU writes the new data bytes.
When only the (eight) data bytes are updated, first 0x0087 is written to the Command Mask Register and then the number of the Message Object is written to the Command Request Register, concurrently updating the data bytes and setting TXRQST.
To prevent the reset of TXRQST at the end of a transmission that may already be in progress while the data is updated, NEWDAT has to be set together with TXRQST. For details see "■ Transmission of Messages".
When NEWDAT is set together with TXRQST, NEWDAT will be reset as soon as the new transmission has started.

■ Configuration of a Receive Object

Figure 3-3 shows how a Receive Object should be initialised.

Figure 3-3 Initialisation of a Receive Object

MSGVAL	ARB	DATA	MASK	EOB	DIR	NEWDAT	MSGLST	RXIE	TXIE	INTPND	RMTEN	TXRQST
1	appl.	appl.	appl.	1	0	0	0	appl.	0	0	0	0

The Arbitration Registers (ID28-0 and XTD bit) are given by the application. They define the identifier and type of accepted received messages. If an 11-bit Identifier ("Standard Frame") is used, it is programmed to ID28 - ID18, ID17 - ID0 can then be disregarded. When a Data Frame with an 11-bit Identifier is received, ID17 - ID0 will be set to "0".
If the RXIE bit is set, the INTPND bit will be set when a received Data Frame is accepted and stored in the Message Object.
The Data Length Code (DLC3-0) is given by the application. When the Message Handler stores a Data Frame in the Message Object, it will store the received Data Length Code and eight data bytes. If the Data Length Code is less than 8, the remaining bytes of the Message Object will be overwritten by non specified values.
The Mask Registers (MSK28-0, UMASK, MXTD, and MDIR bits) may be used (UMASK = "1") to allow groups of Data Frames with similar identifiers to be accepted. For details see "■ Reception of Data Frame".
The DIR bit should not be masked in typical applications.

■ Handling of Received Messages

The CPU may read a received message any time via the IFx Interface registers, the data consistency is guaranteed by the Message Handler state machine.
Typically the CPU will write first 0x007F to the Command Mask Register and then the number of the Message Object to the Command Request Register. That combination will transfer the whole received message from the Message RAM into the Message Buffer Register. Additionally, the bits NEWDAT and INTPND are cleared in the Message RAM (not in the Message Buffer).
If the Message Object uses masks for acceptance filtering, the arbitration bits show which of the matching messages has been received.
The actual value of NEWDAT shows whether a new message has been received since last time this Message Object was read. The actual value of MSGLST shows whether more than one message has been received since last time this Message Object was read. MSGLST will not be automatically reset.
By means of a Remote Frame, the CPU may request another CAN node to provide new data for a receive object. Setting the TXRQST bit of a receive object will cause the transmission of a Remote Frame with the receive object’s identifier. This Remote Frame triggers the other CAN node to start the transmission of the matching Data Frame. If the matching Data Frame is received before the Remote Frame could be transmitted, the TXRQST bit is automatically reset.

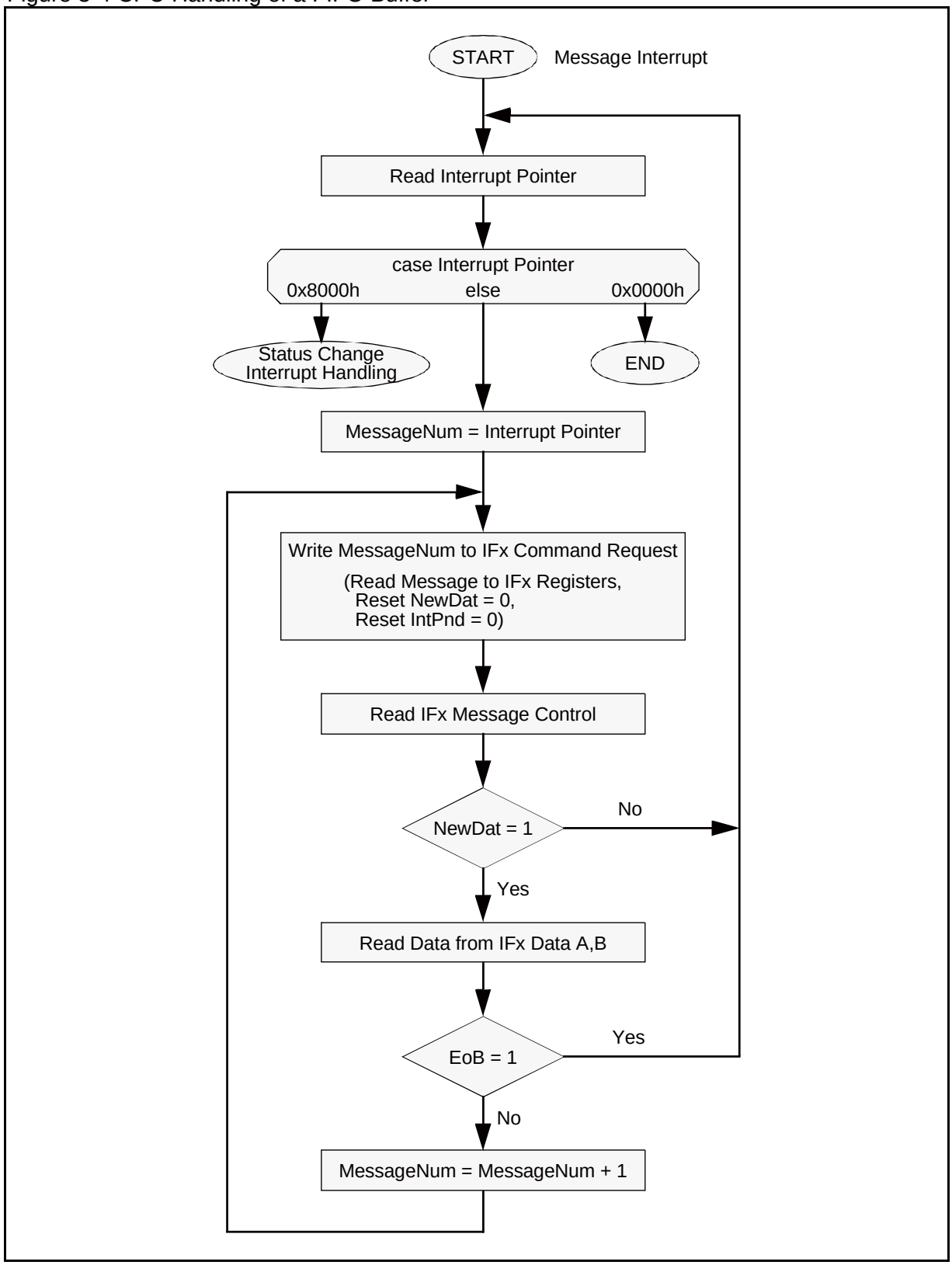
■ Configuration of a FIFO Buffer

With the exception of the EOB bit, the configuration of Receive Objects belonging to a FIFO Buffer is the same as the configuration of a (single) Receive Object, see "■ Configuration of a Receive Object".
To concatenate two or more Message Objects into a FIFO Buffer, the identifiers and masks (if used) of these Message Objects have to be programmed to matching values. Due to the implicit priority of the Message Objects, the Message Object with the lowest number will be the first Message Object of the FIFO Buffer. The EOB bit of all Message Objects of a FIFO Buffer except the last have to be programmed to zero. The EOB bit of the last Message Object of a FIFO Buffer is set to one, configuring it as the End of the Block.

■ Reading from a FIFO Buffer

When the CPU transfers the contents of Message Object to the IFx Message Buffer registers by writing its number to the IFx Command Request Register, the corresponding Command Mask Register should be programmed the way that bits NEWDAT and INTPND are reset to zero (TXRQST = "1" and CIP = "1"). The values of these bits in the Message Control Register always reflect the status before resetting the bits.
To assure the correct function of a FIFO Buffer, the CPU should read out the Message Objects starting at the FIFO Object with the lowest message number. Figure 3-4 CPU Handling of a FIFO Buffer shows how a set of Message Objects which are concatenated to a FIFO Buffer can be handled by the CPU.

Figure 3-4 CPU Handling of a FIFO Buffer



■ Handling of Interrupts

If several interrupts are pending, the CAN Interrupt Register will point to the pending interrupt with the highest priority, disregarding their chronological order. An interrupt remains pending until the CPU has cleared it.

The Status Interrupt has the highest priority. Among the message interrupts, the Message Object's interrupt priority decreases with increasing message number.

A message interrupt is cleared by clearing the Message Object's INTPND bit. The Status Interrupt is cleared by reading the Status Register.

The interrupt identifier INTID in the Interrupt Register INTRn indicates the cause of the interrupt. When no interrupt is pending, the register will hold the value zero. If the value of the Interrupt Register is different from zero, then there is an interrupt pending and, if IE is set, the interrupt line to the CPU is active. The interrupt line remains active until the Interrupt Register is back to value zero (the cause of the interrupt is reset) or until IE is reset.

The value 0x8000 indicates that an interrupt is pending because the CAN Core has updated (not necessarily changed) the Status Register (Error Interrupt or Status Interrupt). This interrupt has the highest priority. The CPU can update (reset) the status bits RXOK, TXOK and LEC, but a write access of the CPU to the Status Register can never generate or reset an interrupt.

All other values indicate that the source of the interrupt is one of the Message Objects, INTID points to the pending message interrupt with the highest interrupt priority.

The CPU controls whether a change of the Status Register may cause an interrupt (bits EIE and SIE in the CAN Control Register) and whether the interrupt line becomes active when the Interrupt Register is different from zero (bit IE in the CAN Control Register). The Interrupt Register will be updated even when IE is reset.

The CPU has two possibilities to follow the source of a message interrupt. First it can follow the INTID in the Interrupt Register and second it can poll the Interrupt Pending Register (see "Section 9 Message Handler Registers").

An interrupt service routine reading the message that is the source of the interrupt may read the message and reset the Message Object's INTPND at the same time (bit CIP in the Command Mask Register IFxCMSKn). When INTPND is cleared, the Interrupt Register will point to the next Message Object with a pending interrupt.

■ Bit Time and Bit Rate

CAN supports bit rates in the range of lower than 1kBit/s up to 1000kBit/s. Each member of the CAN network has its own clock generator, usually a quartz oscillator. The timing parameter of the bit time (i.e. the reciprocal of the bit rate) can be configured individually for each CAN node, creating a common bit rate even though the CAN nodes' oscillator periods (fosc) may be different.

The frequencies of these oscillators are not absolutely stable, small variations are caused by changes in temperature or voltage and by deteriorating components. As long as the variations remain inside a specific oscillator tolerance range (df), the CAN nodes are able to compensate for the different bit rates by resynchronising to the bit stream.

According to the CAN specification, the bit time is divided into four segments (see Figure 3-5). The Synchronisation Segment, the Propagation Time Segment, the Phase Buffer Segment 1, and the Phase Buffer Segment 2. Each segment consists of a specific, programmable number of time quanta (see Table 1). The length of the time quantum (tq) , which is the basic time unit of the bit time, is defined by the CAN controller's system clock fsys and the Baud Rate Prescaler (BRP) : tq = BRP / fsys. The CAN's system clock fsys is the frequency of its CAN_CLK input.

The Synchronisation Segment Sync_Seg is that part of the bit time where edges of the CAN bus level are expected to occur; the distance between an edge that occurs outside of Sync_Seg and the Sync_Seg is called the phase error of that edge. The Propagation Time Segment Prop_Seg is intended to compensate for the physical delay times within the CAN network. The Phase Buffer Segments Phase_Seg1 and Phase_Seg2 surround the Sample Point. The (Re-)Synchronisation Jump Width (SJW) defines how far a resynchronisation may move the Sample Point inside the limits defined by the Phase Buffer Segments to compensate for edge phase errors.

Figure 3-5 Bit Timing

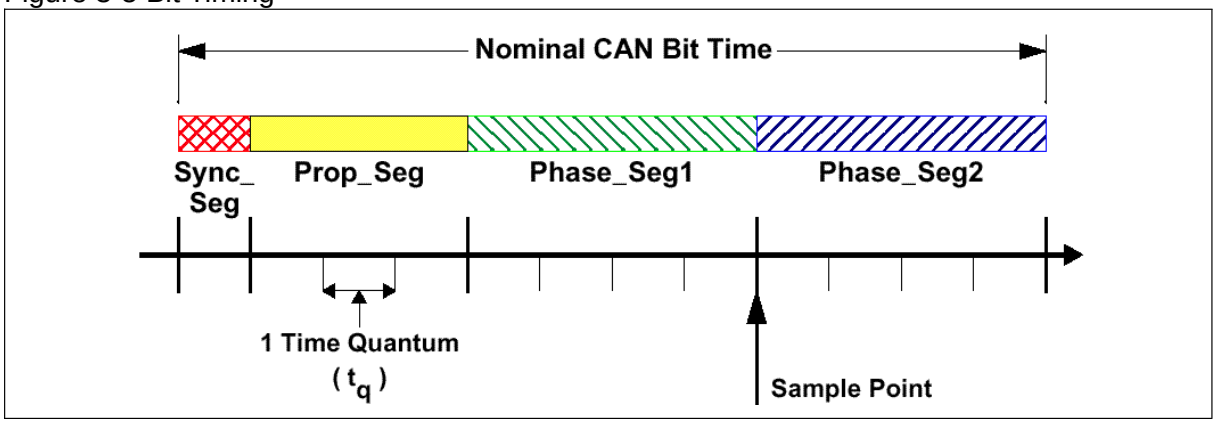


Table 3-1 Parameters of the CAN Bit Time

Parameter	Range	Remark
BRP	[1..32]	defines the legth of the time quantum tq
Sync_Seg	1 tq	fixed length, synchronisation of us into system clock
Prop_Seg	[1..8] tq	compensates for the physical delay times
Phase_Seg1	[1..8] tq	may be lengthened temporarily by synchronisation
Phase_Seg2	[1..8] tq	may be shortened temporarily by synchronisation
SJW	[1..4] tq	may not be longer than either Phase Buffer Segment

Note

This table describes the minimum programmable ranges required by the CAN protocol.
As described earlier in this chapter for BTRn register fields, programming is as follow:

Prop_Seg + Phase_Seg1 = TSEG1 value +1
Phase_Seg2 = TSEG2 value + 1

4. Register Description

This section lists the CAN registers and describes the function of each register in detail.

■ Programmer's Model

The CAN module allocates an address space of 256 bytes (64 words). The CAN registers can be accessed from the CPU in byte and word.

The two sets of interface registers (IF1 and IF2) control the CPU access to the Message RAM. They buffer the data to be transferred to and from the RAM, avoiding conflicts between CPU accesses and message reception/transmission.

If several CAN modules are present on a device then they are located linearly in the address space with a constant offset of 256 bytes. Please refer to the data sheet for the base address of each CAN module on the device.

Table 4-1 CAN Register Summary

Offset	Byte Register Name	Word Register Name	Description	Initial value
CANn base address + 0x0	CTRLRLn	CTRLRn	CAN n - Control register	000X0001
CANn base address + 0x1	CTRLRHn		CAN n - Control register (reserved)	XXXXXXXX
CANn base address + 0x2	STATRLn	STATRn	CAN n - Status register	00000000
CANn base address + 0x3	STATRHn		CAN n - Status register (reserved)	XXXXXXXX
CANn base address + 0x4	ERRCNTLn	ERRCNTn	CAN n - Error Counter (Transmit)	00000000
CANn base address + 0x5	ERRCNTHn		CAN n - Error Counter (Receive)	00000000
CANn base address + 0x6	BTRLn	BTRn	CAN n - Bit Timing Register	00000001
CANn base address + 0x7	BTRHn		CAN n - Bit Timing Register	X0100011
CANn base address + 0x8	INTRLn	INTRn	CAN n - Interrupt Register	00000000
CANn base address + 0x9	INTRHn		CAN n - Interrupt Register	00000000
CANn base address + 0xA	TESTRLn	TESTRn	CAN n - Test Register	000000XX
CANn base address + 0xB	TESTRHn		CAN n - Test Register (reserved)	XXXXXXXX
CANn base address + 0xC	BRPERLn	BRPERn	CAN n - BRP Extension register	XXXX0000
CANn base address + 0xD	BRPERHn		CAN n - BRP Extension register (reserved)	XXXXXXXX
IF1 registers				
CANn base address + 0x10	IF1CREQLn	IF1CREQn	CAN n - IF1 Command request register	00000001
CANn base address + 0x11	IF1CREQHn		CAN n - IF1 Command request register	0XXXXXXX
CANn base address + 0x12	IF1CMSKLn	IF1CMSKn	CAN n - IF1 Command Mask register	00000000

Offset	Byte Register Name	Word Register Name	Description	Initial value
CANn base address + 0x13	IF1CMSKHn		CAN n - IF1 Command Mask register (reserved)	XXXXXXXX
CANn base address + 0x14	IF1MSK1Ln	IF1MSK1n	CAN n - IF1 Mask Register	11111111
CANn base address + 0x15	IF1MSK1Hn		CAN n - IF1 Mask Register	11111111
CANn base address + 0x16	IF1MSK2Ln	IF1MSK2n	CAN n - IF1 Mask Register	11111111
CANn base address + 0x17	IF1MSK2Hn		CAN n - IF1 Mask Register	11X11111
CANn base address + 0x18	IF1ARB1Ln	IF1ARB1n	CAN n - IF1 Arbitration register	00000000
CANn base address + 0x19	IF1ARB1Hn		CAN n - IF1 Arbitration register	00000000
CANn base address + 0x1A	IF1ARB2Ln	IF1ARB2n	CAN n - IF1 Arbitration register	00000000
CANn base address + 0x1B	IF1ARB2Hn		CAN n - IF1 Arbitration register	00000000
CANn base address + 0x1C	IF1MCTRLn	IF1MCTRn	CAN n - IF1 Message Control Register	0XXX0000
CANn base address + 0x1D	IF1MCTRHn		CAN n - IF1 Message Control Register	00000000
CANn base address + 0x1E	IF1DTA1Ln	IF1DTA1n	CAN n - IF1 Data A1	00000000
CANn base address + 0x1F	IF1DTA1Hn		CAN n - IF1 Data A1	00000000
CANn base address + 0x20	IF1DTA2Ln	IF1DTA2n	CAN n - IF1 Data A2	00000000
CANn base address + 0x21	IF1DTA2Hn		CAN n - IF1 Data A2	00000000
CANn base address + 0x22	IF1DTB1Ln	IF1DTB1n	CAN n - IF1 Data B1	00000000
CANn base address + 0x23	IF1DTB1Hn		CAN n - IF1 Data B1	00000000
CANn base address + 0x24	IF1DTB2Ln	IF1DTB2n	CAN n - IF1 Data B2	00000000
CANn base address + 0x25	IF1DTB2Hn		CAN n - IF1 Data B2	00000000
IF2 registers				
CANn base address + 0x40	IF2CREQLn	IF2CREQn	CAN n - IF2 Command request register	00000001
CANn base address + 0x41	IF2CREQHn		CAN n - IF2 Command request register	0XXXXXXX
CANn base address + 0x42	IF2CMSKLn	IF2CMSKn	CAN n - IF2 Command Mask register	00000000
CANn base address + 0x43	IF2CMSKHn		CAN n - IF2 Command Mask register (reserved)	XXXXXXXX

Offset	Byte Register Name	Word Register Name	Description	Initial value
CANn base address + 0x44	IF2MSK1Ln	IF2MSK1n	CAN n - IF2 Mask Register	11111111
CANn base address + 0x45	IF2MSK1Hn		CAN n - IF2 Mask Register	11111111
CANn base address + 0x46	IF2MSK2Ln	IF2MSK2n	CAN n - IF2 Mask Register	11111111
CANn base address + 0x47	IF2MSK2Hn		CAN n - IF2 Mask Register	11X11111
CANn base address + 0x48	IF2ARB1Ln	IF2ARB1n	CAN n - IF2 Arbitration register	00000000
CANn base address + 0x49	IF2ARB1Hn		CAN n - IF2 Arbitration register	00000000
CANn base address + 0x4A	IF2ARB2Ln	IF2ARB2n	CAN n - IF2 Arbitration register	00000000
CANn base address + 0x4B	IF2ARB2Hn		CAN n - IF2 Arbitration register	00000000
CANn base address + 0x4C	IF2MCTRLn	IF2MCTRn	CAN n - IF2 Message Control Register	0XXX0000
CANn base address + 0x4D	IF2MCTRHn		CAN n - IF2 Message Control Register	00000000
CANn base address + 0x4E	IF2DTA1Ln	IF2DTA1n	CAN n - IF2 Data A1	00000000
CANn base address + 0x4F	IF2DTA1Hn		CAN n - IF2 Data A1	00000000
CANn base address + 0x50	IF2DTA2Ln	IF2DTA2n	CAN n - IF2 Data A2	00000000
CANn base address + 0x51	IF2DTA2Hn		CAN n - IF2 Data A2	00000000
CANn base address + 0x52	IF2DTB1Ln	IF2DTB1n	CAN n - IF2 Data B1	00000000
CANn base address + 0x53	IF2DTB1Hn		CAN n - IF2 Data B1	00000000
CANn base address + 0x54	IF2DTB2Ln	IF2DTB2n	CAN n - IF2 Data B2	00000000
CANn base address + 0x55	IF2DTB2Hn		CAN n - IF2 Data B2	00000000
CANn base address + 0x80	TREQR1Ln	TREQR1n	CAN n - Transmission Request Register	00000000
CANn base address + 0x81	TREQR1Hn		CAN n - Transmission Request Register	00000000
CANn base address + 0x82	TREQR2Ln	TREQR2n	CAN n - Transmission Request Register	00000000
CANn base address + 0x83	TREQR2Hn		CAN n - Transmission Request Register	00000000
CANn base address + 0x90	NEWDT1Ln	NEWDT1n	CAN n - New Data Register	00000000
CANn base address + 0x91	NEWDT1Hn		CAN n - New Data Register	00000000

Offset	Byte Register Name	Word Register Name	Description	Initial value
CANn base address + 0x92	NEWDT2Ln	NEWDT2n	CAN n - New Data Register	00000000
CANn base address + 0x93	NEWDT2Hn		CAN n - New Data Register	00000000
CANn base address + 0xA0	INTPND1Ln	INTPND1n	CAN n - Interrupt Pending Register	00000000
CANn base address + 0xA1	INTPND1Hn		CAN n - Interrupt Pending Register	00000000
CANn base address + 0xA2	INTPND2Ln	INTPND2n	CAN n - Interrupt Pending Register	00000000
CANn base address + 0xA3	INTPND2Hn		CAN n - Interrupt Pending Register	00000000
CANn base address + 0xB0	MSGVAL1Ln	MSGVAL1n	CAN n - Message Valid Register	00000000
CANn base address + 0xB1	MSGVAL1Hn		CAN n - Message Valid Register	00000000
CANn base address + 0xB2	MSGVAL2Ln	MSGVAL2n	CAN n - Message Valid Register	00000000
CANn base address + 0xB3	MSGVAL2Hn		CAN n - Message Valid Register	00000000
CANn base address + 0xCE	COERn		CAN n - Output enable register	XXXXXXX0

■ Hardware Reset Description

After hardware reset, the registers of the CAN hold the values described as initial value in Table 4-1. Additionally, the busoff state is reset and the output CAN_TX is set to recessive (HIGH). The value 0x0001 (INIT = "1") in the CAN Control Register CTRLRLn enables the software initialisation. The CAN does not influence the CAN bus until the CPU resets INIT to "0".

The data stored in the Message RAM is not affected by a hardware reset. After power-on, the contents of the Message RAM is undefined.

5. CAN Device Related Registers

These registers are related to the device which hosts the CANbus controller.

■ List of CAN Device Related Registers

Abbreviated Register Name	Register Name	Reference
COERn	CAN Output Enable Register	See 5.1

5.1. CAN Output Enable Register (COERn)

The CAN Output Enable Register (COERn) controls whether the device pins is used as CANbus controller's TX pin or not.

■ CAN Output Enable Register (COERn)

COERn								
bit	7	6	5	4	3	2	1	0
	RESV	RESV	RESV	RESV	RESV	RESV	RESV	OE
Attribute	R/W0	R/W0	R/W0	R/W0	R/W0	R/W0	R/W0	R/W
Initial value	X	X	X	X	X	X	X	0

[bit7 to bit1] RESV: Reserved Bits

- Write always "0".
- Read value is undefined.
- Read modify write operations to this register have no effect on this bit.

[bit0] OE: Output Enable

Bit	Description
0	CAN TX is disabled. Pin can be used as General Purpose Port or for other peripheral function
1	CAN TX is enabled.

6. CAN Protocol Related Registers

These registers are related to the CAN protocol controller in the CAN Core. They control the operating modes and the configuration of the CAN bit timing and provide status information.

■ List of CAN Protocol Related Registers

Abbreviated Register Name	Register Name	Reference
CTRLRn	CAN Control Register	See 6.1
STATRn	Status Register	See 6.2
ERRCNTn	Error Counter	See 6.3
BTRn	Bit Timing Register	See 6.4
TESTRn	Test Register	See 6.5
BRPERn	BRP Extension Register	See 6.6

6.1. CAN Control Register (CTRLRn)

The CAN Control Register (CTRLRn) controls the basic operation modes of the CAN controller.

■ CAN Control Register (CTRLRn)

CTRLRHn

bit	15	14	13	12	11	10	9	8
	RESV	RESV	RESV	RESV	RESV	RESV	RESV	RESV
Attribute	R	R	R	R	R	R	R	R
Initial value	X	X	X	X	X	X	X	X

CTRLRLn

bit	7	6	5	4	3	2	1	0
	TEST	CCE	DAR	RESV	EIE	SIE	IE	INIT
Attribute	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W
Initial value	0	0	0	X	0	0	0	1

[bit15 to bit8] RESV: Reserved Bits

- Write always "0".
- Read value is undefined.
- Read modify write operations to this register have no effect on this bit.

[bit7] TEST: Test Mode Enable

Bit	Description
0	Normal Operation.
1	Test Mode.

[bit6] CCE: Configuration Change Enable

Bit	Description
0	The CPU has no write access to the Bit Timing Register.
1	The CPU has write access to the Bit Timing Register (while Init = "1")

[bit5] DAR: Disable Automatic Retransmission

Bit	Description
0	Automatic Retransmission of disturbed messages enabled.
1	Automatic Retransmission disabled.

Note:

When the C_CAN is configured to work in DAR-mode (Disable Automatic Retransmission) and the host requests the transmission of several messages at the same time, only two of these messages will be transmitted. For all other requested transmit messages, the TXRQST bits will be reset, but no transmission will be started, NEWDAT and INTPND will be left unchanged. For the two messages that are transmitted, the TXRQST and NEWDAT bits will be reset and, if enabled by TXIE, INTPND will be set.

[bit4] RESV: Reserved Bit

- Write always "0".
- Read value is undefined.
- Read modify write operations to this register have no effect on this bit.

[bit3] EIE: Error Interrupt Enable

Bit	Description
0	Disabled - No Error Status Interrupt will be generated.
1	Enabled - A change in the bits BOff or EWarn in the Status Register will generate an interrupt.

[bit2] SIE: Status Change Interrupt Enable

Bit	Description
0	Disabled - No Status Change Interrupt will be generated.
1	Enabled - An interrupt will be generated when a message transfer is successfully completed or a CAN bus error is detected.

[bit1] IE: Module Interrupt Enable

Bit	Description
0	Disabled - Module Interrupt is always inactive.
1	Enabled - Interrupts will be generated. The request remains active until all pending interrupts are processed.

[bit0] INIT: Initialization

Bit	Description
0	Normal Operation
1	Initialization is started.

Note:

The busoff recovery sequence (see CAN Specification Rev. 2.0) cannot be shortened by setting or resetting INIT. If the device goes busoff, it will set INIT of its own accord, stopping all bus activities. Once INIT has been cleared by the CPU, the device will then wait for 129 occurrences of Bus Idle (129×11 consecutive recessive bits) before resuming normal operation. At the end of the busoff recovery sequence, the Error Management Counters will be reset.

Note:

During the waiting time after the resetting of INIT, each time a sequence of 11 recessive bits has been monitored, a Bit0Error code is written to the Status Register, enabling the CPU to readily check up whether the CAN bus is stuck at dominant level or continuously disturbed and to monitor the proceeding of the busoff recovery sequence.

6.2. Status Register (STATRn)

The Status Register (STATRn) shows the status of the CAN controller.

■ Status Register (STATRn)

STATRHn

bit	15	14	13	12	11	10	9	8
	RESV	RESV	RESV	RESV	RESV	RESV	RESV	RESV
Attribute	R	R	R	R	R	R	R	R
Initial value	X	X	X	X	X	X	X	X

STATRLn

bit	7	6	5	4	3	2	1	0
	BOFF	EWARN	EPASS	RXOK	TXOK	LEC		
Attribute	R	R	R	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

[bit15 to bit8] RESV: Reserved Bits

- Write always "0".
- Read value is undefined.
- Read modify write operations to this register have no effect on this bit.

[bit7] BOFF: Busoff Status

Bit	Description
0	The CAN module is not busoff.
1	The CAN module is in busoff state.

[bit6] EWARN: Warning Status

Bit	Description
0	Both error counters are below the error warning limit of 96.
1	At least one of the error counters in the EML has reached the error warning limit of 96.

[bit5] EPASS: Error Passive

Bit	Description
0	The CAN Core is error active.
1	The CAN Core is in the error passive state as defined in the CAN Specification.

[bit4] RXOK: Received a Message Successfully

Bit	Description
0	Since this bit was last reset by the CPU, no message has been successfully received. This bit is never reset by the CAN Core.
1	Since this bit was last reset (to zero) by the CPU, a message has been successfully received (independent of the result of acceptance.

[bit3] TXOK: Transmitted a Message Successfully

Bit	Description
0	Since this bit was last reset by the CPU, no message has been successfully transmitted. This bit is never reset by the CAN Core.
1	Since this bit was last reset by the CPU, a message has been successfully (error free and acknowledged by at least one other node) transmitted.

[bit2 to bit0] LEC: Last Error Code (Type of the last error to occur on the CAN bus)

Bit	Description	
0	No Error	
1	Stuff Error	More than 5 equal bits in a sequence have occurred in a part of a received message where this is not allowed.
2	Form Error	A fixed format part of a received frame has the wrong format.
3	AckError	The message this CAN Core transmitted was not acknowledged by another node.
4	Bit1Error	During the transmission of a message (with the exception of the arbitration field), the device wanted to send a recessive level (bit of logical value "1"), but the monitored bus value was dominant.
5	Bit0Error	During the transmission of a message (or acknowledge bit or active error flag, or overload flag), the device wanted to send a dominant level (data or identifier bit logical value "0"), but the monitored Bus value was recessive. During busoff recovery this status is set each time a sequence of 11 recessive bits has been monitored. This enables the CPU to monitor the proceeding of the busoff recovery sequence (indicating the bus is not stuck at dominant level or continuously disturbed).
6	CRCError	The CRC check sum was incorrect in the message received, the CRC received for an incoming message does not match with the calculated CRC for the received data.
7	unused	When the LEC shows the value "7", no CAN bus event was detected since the CPU wrote this value to the LEC.

The LEC field holds a code which indicates the type of the last error to occur on the CAN bus. This field will be cleared to "0" when a message has been transferred (reception or transmission) without error. The unused code "7" may be written by the CPU to check for updates.

■ Status Interrupts

A Status Interrupt is generated by bits BOFF and EWARN (Error Interrupt) or by RXOK, TXOK, and LEC (Status Change Interrupt) assuming that the corresponding enable bits in the CAN Control Register are set. A change of bit EPASS or a write to RXOK, TXOK,or LEC will never generate a Status Interrupt.
Reading the Status Register will clear the Status Interrupt value (8000h) in the Interrupt Register, if it is pending.

6.3. Error Counter (ERRCNTn)

The Error Counter Register (ERRCNTn) contains the Receive and Transmit Error Counters REC and TEC.

■ Error Counter (ERRCNTn)

ERRCNTHn

bit	15	14	13	12	11	10	9	8
	RP	REC6-0						
Attribute	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

ERRCNTLn

bit	7	6	5	4	3	2	1	0
	TEC7-0							
Attribute	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

[bit15] RP: Receive Error Passive

Bit	Description
0	The Receive Error Counter is below the error passive level.
1	The Receive Error Counter has reached the error passive level as defined in the CAN Specification.

[bit14 to bit8] RE: Receive Error Counter C6-0
Actual state of the Receive Error Counter. Values between 0 and 127.

[bit7 to bit0] TEC7-0: Transmit Error Counter
Actual state of the Transmit Error Counter. Values between 0 and 255.

6.4. Bit Timing Register (BTRn)

The Bit Timing Register (BTRn) enables controlling of the CAN bus controller bit timing.

■ Bit Timing Register (BTRn)

BTRHn								
bit	15	14	13	12	11	10	9	8
	RESV	TSEG2			TSEG1			
Attribute	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	X	0	1	0	0	0	1	1

BTRLn								
bit	7	6	5	4	3	2	1	0
	SJW		BRP					
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	1

[bit15] RESV: Reserved Bit

- Write always "0".
- Read value is undefined.
- Read modify write operations to this register have no effect on this bit.

[bit14 to bit12] TSEG2: The Time Segment after the Sample Point

Bit	Description
0x0-0x7	Valid values for TSEG2 are [0 … 7]. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.

[bit11 to bit8] TSEG1: The Time Segment before the Sample Point

Bit	Description
0x01-0x0F	Valid values for TSEG1 are [1 … 15]. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used

[bit7, bit6] SJW: (Re)Synchronisation Jump Width

Bit	Description
0x0-0x3	Valid programmed values are 0-3. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.

[bit5 to bit0] BRP: Baud Rate Prescaler

Bit	Description
0x00-0x3F	The value by which the peripheral clock CLKP2 frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid values for the Baud Rate Prescaler are[0 … 63]. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.

Note:

With a peripheral clock CLKP2 frequency of 8 MHz, the reset value of 0x2301 configures the CAN for a bit rate of 500 kBit/s. The registers are only writable if bits CCE and INIT in the CAN Control Register are set.

6.5. Test Register (TESTRn)

The Test Register (TESTRn) offers functionality to test the CAN bus system.

■ Test Register (TESTRn)

TESTRHn

bit	15	14	13	12	11	10	9	8
	RESV	RESV	RESV	RESV	RESV	RESV	RESV	RESV
Attribute	R	R	R	R	R	R	R	R
Initial value	X	X	X	X	X	X	X	X

TESTRLn

bit	7	6	5	4	3	2	1	0
	RX	TX1	TX0	LBACK	SILENT	BASIC	RESV	RESV
Attribute	R	R/W	R/W	R/W	R/W	R/W	R	R
Initial value	0	0	0	0	0	0	X	X

[bit15 to bit8] RESV: Reserved Bits

- Write always "0".
- Read value is undefined.
- Read modify write operations to this register have no effect on this bit.

[bit7] RX: Monitors the Actual Value of the CAN_RX Pin

- Write always "0".
- Read modify write operations to this register have no effect on this bit.

Bit	Description
0	The CAN bus is dominant (CAN_RX = "0").
1	The CAN bus is recessive (CAN_RX = "1").

[bit6, bit5] TX1-0: Control of CAN_TX pin

Bit	Description
00	Reset value, CAN_TX is controlled by the CAN Core.
01	Sample Point can be monitored at CAN_TX pin.
10	CAN_TX pin drives a dominant ("0") value.
11	CAN_TX pin drives a recessive ("1") value.

[bit4] LBACK: Loop Back Mode

Bit	Description
0	Loop Back Mode is disabled.
1	Loop Back Mode is enabled.

[bit3] SILENT: Silent Mode

Bit	Description
0	Normal operation.
1	The module is in Silent Mode.

[bit2] BASIC: Basic Mode

Bit	Description
0	Basic Mode disabled.
1	IF1 Registers used as Tx Buffer, IF2 Registers used as Rx Buffer.

[bit1, bit0] RESV: Reserved Bits

- Write always "0".
- Read value is undefined.
- Read modify write operations to this register have no effect on this bit.

Note:

Write access to the Test Register is enabled by setting bit TEST in the CAN Control Register CTRLRn. The different test functions may be combined, but TX1-0 /= "00" disturbs message transfer.

6.6. BRP Extension Register (BRPERn)

The BRP Extension Register (BRPERn) contains an extension for the baud rate generator prescaler.

■ BRP Extension Register (BRPERn)

BRPERHn								
bit	15	14	13	12	11	10	9	8
	RESV	RESV	RESV	RESV	RESV	RESV	RESV	RESV
Attribute	R	R	R	R	R	R	R	R
Initial value	X	X	X	X	X	X	X	X

BRPERLn								
bit	7	6	5	4	3	2	1	0
	RESV	RESV	RESV	RESV	BRPE			
Attribute	R	R	R	R	R/W	R/W	R/W	R/W
Initial value	X	X	X	X	0	0	0	0

[bit15 to bit4] RESV: Reserved Bits

- Write always "0".
- Read value is undefined.
- Read modify write operations to this register have no effect on this bit.

[bit3 to bit0] BRPE: Baud Rate Prescaler Extension

Bit	Description
0x00-0x0F	By programming BRPE the Baud Rate Prescaler can be extended to values up to 1023. The actual interpretation by the hardware is that one more than the value programmed by BRPE (MSBs) and BRP (LSBs) is used.

7. Message Interface Register Sets

To avoid access conflicts between the CPU and the CAN bus controller in accessing the Message RAM, there are two sets of Interface Registers which are used to control the CPU access to the Message RAM.

■ Overview of Message Interface Register Sets

There are two sets of Interface Registers which are used to control the CPU access to the Message RAM. The Interface Registers avoid conflicts between CPU access to the Message RAM and CAN message reception and transmission by buffering the data to be transferred. A complete Message Object (see "Section 8 Message Object in the Message Memory") or parts of the Message Object may be transferred between the Message RAM and the IFx Message Buffer registers in one single transfer.

The function of the two interface register sets is identical (except for test mode BASIC). They can be used the way that one set of registers is used for data transfer to the Message RAM while the other set of registers is used for the data transfer from the Message RAM, allowing both processes to be interrupted by each other. Table 7-1 gives an overview of the two Interface Register sets.

Each set of Interface Registers consists of Message Buffer Registers controlled by their own Command Registers. The Command Mask Register specifies the direction of the data transfer and which parts of a Message Object will be transferred. The Command Request Register is used to select a Message Object in the Message RAM as target or source for the transfer and to start the action specified in the Command Mask Register.

Table 7-1 IF1 and IF2 Message Interface Register Sets

Address	IF1 register set	Address	IF2 register set
CAN Base + 0x10	IF1 Command Request (IF1CREQn)	CAN Base + 0x40	IF2 Command Request (IF2CREQn)
CAN Base + 0x12	IF1 Command Mask (IF1CMSKn)	CAN Base + 0x42	IF2 Command Mask (IF2CMSKn)
CAN Base + 0x14	IF1 Mask 1 (IF1MSK1n)	CAN Base + 0x44	IF2 Mask 1 (IF2MSK1n)
CAN Base + 0x16	IF1 Mask 2 (IF1MSK2n)	CAN Base + 0x46	IF2 Mask 2 (IF2MSK2n)
CAN Base + 0x18	IF1 Arbitration 1 (IF1ARB1n)	CAN Base + 0x48	IF2 Arbitration 1 (IF2ARB1n)
CAN Base + 0x1A	IF1 Arbitration 2 (IF1ARB2n)	CAN Base + 0x4A	IF2 Arbitration 2 (IF2ARB2n)
CAN Base + 0x1C	IF1 Message Control (IF1MCTRn)	CAN Base + 0x4C	IF2 Message Control (IF2MCTRn)
CAN Base + 0x1E	IF1 Data A1 (IF1DTA1n)	CAN Base + 0x4E	IF2 Data A1 (IF2DTA1n)
CAN Base + 0x20	IF1 Data A2 (IF1DTA2n)	CAN Base + 0x50	IF2 Data A2 (IF2DTA2n)
CAN Base + 0x22	IF1 Data B1 (IF1DTB1n)	CAN Base + 0x52	IF2 Data B1 (IF2DTB1n)
CAN Base + 0x24	IF1 Data B2 (IF1DTB2n)	CAN Base + 0x54	IF2 Data B2 (IF2DTB2n)

■ List of Message Interface Registers

Abbreviated Register Name	Register Name	Reference
IFxCREQn	IFx Command Request Register	See 7.1
IFxCMSKn	IFx Command Mask Register	See 7.2
IFxMSK1n, IFxMSK2n	IFx Mask Register	See 7.3
IFxARB1n, IFxARB2n	IFx Arbitration Register	See 7.4
IFxMCTRn	IFx Message Control Register	See 7.5
IFxDTA1n, IFxDTA2n, IFxDTB1n, IFxDTB2n	IFx Data A and Data B Register	See 7.6

7.1. IFx Command Request Registers (IFxCREQn)

A message transfer is started as soon as the CPU has written the message number to the Command Request Register (IFxCREQn). With this write operation the CPU is notified that a transfer is in progress. If a CPU access to the CAN happens while the transfer is in progress then this access is delayed until the transfer has finished. After 3 to 6 CAN_CLK periods, the transfer between the Interface Register and the Message RAM has completed and the upcoming CPU access is executed.

■ IFx Command Request Registers (IFxCREQn)

IFxCREQHn

bit	15	14	13	12	11	10	9	8
	BUSY	RESV	RESV	RESV	RESV	RESV	RESV	RESV
Attribute	R/W	R	R	R	R	R	R	R
Initial value	0	X	X	X	X	X	X	X

IFxCREQLn

bit	7	6	5	4	3	2	1	0
	MSGN							
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	1

[bit15] BUSY: Busy Flag

Bit	Description
0	Reset to zero when read/write action has finished
1	Set to one when writing to the IFx Command Request Register

[bit14 to bit8] RESV: Reserved Bits

- Write always "0".
- Read value is undefined.
- Read modify write operations to this register have no effect on this bit.

[bit5 to bit0] MSGN: Message Number (for 32 message buffer CANs, please refer to the data sheet for the number of message buffers of the device.)

Bit	Description
0x00	Not a valid Message Number, interpreted as 0x20.
0x01-0x20	Valid Message Number, the Message Object in the Message RAM is selected for data transfer.
0x21-0xFF	Not a valid Message Number, interpreted as 0x01-0x1F.

[bit5 to bit0] MSGN: Message Number (for 64 message buffer CANs, , please refer to data sheet for the number of message buffers of the device.)

Bit	Description
0x00	Not a valid Message Number, interpreted as 0x20.
0x01-0x40	Valid Message Number, the Message Object in the Message RAM is selected for data transfer.
0x41-0xFF	Not a valid Message Number, interpreted as 0x01-0x3F.

[bit7 to bit0] MSGN: Message Number (for 128 message buffer CANs, please refer to the data sheet for the number of message buffers of the device.)

Bit	Description
0x00	Not a valid Message Number, interpreted as 0x80.
0x01-0x80	Valid Message Number, the Message Object in the Message RAM is selected for data transfer.
0x81-0xFF	Not a valid Message Number, interpreted as 0x01-0x7F.

Note:

The Busy Flag can only be used in BASIC mode (see "■ Basic Mode"). When using the Message RAM (BASIC=0) the hardware interface controls the access for read and write and this flag is always read as "0".

Note:

When a Message Number that is not valid is written into the Command Request Register, the Message Number will be transformed into a valid value and that Message Object will be transferred.

7.2. IFx Command Mask Register (IFxCMSKn)

The control bits of the IFx Command Mask Register (IFxCMSKn) specify the transfer direction and select which of the IFx Message Buffer Registers are source or target of the data transfer.

■ IFx Command Mask Register (IFxCMSKn)

IFxCMSKHn

bit	15	14	13	12	11	10	9	8
	RESV	RESV	RESV	RESV	RESV	RESV	RESV	RESV
Attribute	R	R	R	R	R	R	R	R
Initial value	X	X	X	X	X	X	X	X

IFxCMSKLn

bit	7	6	5	4	3	2	1	0
	WRRD	MASK	ARB	CONTROL	CIP	TXREQ	DATAA	DATAB
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

[bit15 to bit8] RESV: Reserved Bits

- Write always "0".
- Read value is undefined.
- Read modify write operations to this register have no effect on this bit.

[bit7] WRRD: Write / Read

Bit	Description
0	Read: Transfer data from the Message Object addressed by the Command Request Register into the selected Message Buffer Registers.
1	Write: Transfer data from the selected Message Buffer Registers to the Message Object addressed by the Command Request Register.

The other bits of IFx Command Mask Register have different functions depending on the transfer direction:

■ Direction = Write

[bit6] MASK: Access Mask Bits

Bit	Description
0	Mask bits unchanged.
1	Transfer Identifier Mask + MDir + MXtd to Message Object.

[bit5] ARB: Access Arbitration Bits

Bit	Description
0	Arbitration bits unchanged.
1	Transfer Identifier + Dir + Xtd + MsgVal to Message Object.

[bit4] CONTROL: Access Control Bits

Bit	Description
0	Control Bits unchanged.
1	Transfer Control Bits to Message Object.

[bit3] CIP: Clear Interrupt Pending Bit

When writing to a Message Object, this bit is ignored.

[bit2] TXREQ: Access Transmission Request Bit

Bit	Description
0	TXRQST bit unchanged
1	set TXRQST bit

[bit1] DATAA: Access Data Bytes 0-3

Bit	Description
0	Data Bytes 0-3 unchanged.
1	Transfer Data Bytes 0-3 to Message Object.

[bit0] DATAB: Access Data Bytes 4-7

Bit	Description
0	Data Bytes 4-7 unchanged.
1	Transfer Data Bytes 4-7 to Message Object.

■ Direction = Read

[bit6] MASK: Access Mask Bits

Bit	Description
0	Mask bits unchanged.
1	Transfer Identifier Mask + MDir + MXtd to IFx Message Buffer Register.

[bit5] ARB: Access Arbitration Bits

Bit	Description
0	Arbitration bits unchanged.
1	Transfer Identifier + Dir + Xtd + MsgVal to IFx Message Buffer Register.

[bit4] CONTROL: Access Control Bits

Bit	Description
0	Control Bits unchanged.
1	Transfer Control Bits to IFx Message Buffer Register.

[bit3] CIP: Clear Interrupt Pending Bit

Bit	Description
0	INTPND bit remains unchanged.
1	Clear INTPND bit in the Message Object.

[bit2] TXREQ: Access New Data Bit

Bit	Description
0	NEWDAT bit remains unchanged.
1	Clear NEWDAT bit in the Message Object.

[bit1] DATAA: Access Data Bytes 0-3

Bit	Description
0	Data Bytes 0-3 unchanged.
1	Transfer Data Bytes 0-3 to IFx Message Buffer Register.

[bit0] DATAB: Access Data Bytes 4-7

Bit	Description
0	Data Bytes 4-7 unchanged.
1	Transfer Data Bytes 4-7 to IFx Message Buffer Register.

Note:

A read access to a Message Object can be combined with the reset of the control bits INTPND and NEWDAT. The values of these bits transferred to the IFx Message Control Register always reflect the status before resetting these bits.

7.3. IFx Mask Registers (IFxMSK1n, IFxMSK2n)

The bits of the Message Buffer registers mirror the Message Objects in the Message RAM. The bits of the IFx Mask Registers (IFxMSK1n, IFxMSK2n) specify which parts of the message arbitration registers (IFxARB1n, IFxARB2n) are evaluated.

■ IFx Mask Registers (IFxMSK1n, IFxMSK2n)

IFxMSK1Hn

bit	15	14	13	12	11	10	9	8
	MSK15-8							
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	1	1	1	1	1	1	1	1

IFxMSK1Ln

bit	7	6	5	4	3	2	1	0
	MSK7-0							
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	1	1	1	1	1	1	1	1

IFxMSK2Hn

bit	15	14	13	12	11	10	9	8
	MXTD	MDIR	RESV	MSK28-24				
Attribute	R/W	R/W	-	R/W	R/W	R/W	R/W	R/W
Initial value	1	1	X	1	1	1	1	1

[bit13] RESV: Reserved Bit

- Write always "1".
- Read value is undefined.
- Read modify write operations to this register have no effect on this bit.

IFxMSK2Ln

bit	7	6	5	4	3	2	1	0
	MSK23-16							
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	1	1	1	1	1	1	1	1

7.4. IFx Arbitration Registers (IFxARB1n, IFxARB2n)

The bits of the Message Buffer registers mirror the Message Objects in the Message RAM. In case of transmission, the bits of the message arbitration registers (IFxARB1n, IFxARB2n) specify the ID of the message. In case of reception, the bits of the message arbitration registers (IFxARB1n, IFxARB2n) specify the acceptance filter.

■ IFx Arbitration Registers (IFxARB1n, IFxARB2n)

IFxARB1Hn

bit	15	14	13	12	11	10	9	8
	ID15-8							
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

IFxARB1Ln

bit	7	6	5	4	3	2	1	0
	ID7-0							
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

IFxARB2Hn

bit	15	14	13	12	11	10	9	8
	MSGVAL	XTD	DIR	ID28-24				
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

IFxARB2Ln

bit	7	6	5	4	3	2	1	0
	ID23-16							
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

7.5. IFx Message Control Register (IFxMCTRn)

The bits of the Message Buffer registers mirror the Message Objects in the Message RAM. The bits of the Message Control Register (IFxMCTRn) contain status information and control bits.

■ IFx Message Control Register (IFxMCTRn)

IFxMCTRHn

bit	15	14	13	12	11	10	9	8
	NEWDAT	MSGLST	INTPND	UMASK	TXIE	RXIE	RMTEN	TXRQST
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

IFxMCTRLn

bit	7	6	5	4	3	2	1	0
	EOB	-	-	-	DLC3-0			
Attribute	R/W	-	-	-	R/W	R/W	R/W	R/W
Initial value	0	X	X	X	0	0	0	0

[bit6 to bit4] -: Undefined Bit

- Write always "0".
- Read value is undefined.
- Read modify write operations to this register have no effect on this bit.

7.6. IFx Data A and Data B Registers (IFxDTA1n, IFxDTA2n, IFxDTB1n, IFxDTB2n)

The data bytes of CAN messages are stored in the IFx Message Buffer Registers.

■ IFx Data A and Data B Registers (IFxDTAn, IFxDTBn)

In a CAN Data Frame, Data(0) is the first, Data(7) is the last byte to be transmitted or received. In CAN's serial bit stream, the MSB of each byte will be transmitted first.

The data bytes of CAN messages are stored in the IFx Message Buffer Registers in the following order:

Table 7-2 IFx Data A and Data B Registers List

Register	Address	Address + 0 content	Address + 1 content
IFxDTA1n	x = 1: CANn base + 0x1E x = 2: CANn base + 0x4E	Data(0)	Data(1)
IFxDTA2n	x = 1: CANn base + 0x20 x = 2: CANn base + 0x50	Data(2)	Data(3)
IFxDTB1n	x = 1: CANn base + 0x22 x = 2: CANn base + 0x52	Data(4)	Data(5)
IFxDTB2n	x = 1: CANn base + 0x24 x = 2: CANn base + 0x54	Data(6)	Data(7)

8. Message Object in the Message Memory

There are 32 Message Objects (up to 128 depending on the implementation) in the Message RAM. To avoid conflicts between CPU access to the Message RAM and CAN message reception and transmission, the CPU cannot directly access the Message Objects, these accesses are handled via the IFx Interface Registers.

■ Structure of a Message Object in the Message Memory

Figure 8-1 gives an overview of the two structure of a Message Object.

Figure 8-1 Structure of a Message Object in the Message Memory

Message Object												
UMASK	MSK28-0	MXTD	MDIR	EOB	NEWDAT		MSGLST	RXIE	TXIE	INTPND	RMTEN	TXRQST
MSGVAL	ID28-0	XTD	DIR	DLC3-0	Data0	Data1	Data2	Data3	Data4	Data5	Data6	Data7

MSGVAL: Message Valid

Bit	Description
0	The Message Object is ignored by the Message Handler.
1	The Message Object is configured and should be considered by the Message Handler.

Note:

The CPU must reset the MSGVAL bit of all unused Messages Objects during the initialization before it resets bit INIT in the CAN Control Register. This bit must also be reset before the identifier ID28-0, the control bits XTD, DIR, or the Data Length Code DLC3-0 are modified, or if the Message Object is no longer required.

UMASK: Use Acceptance Mask

Bit	Description
0	Mask ignored.
1	Use Mask (MSK28-0, MXTD, and MDIR) for acceptance filtering.

Note:

If the UMASK bit is set to one, the Message Object’s mask bits have to be programmed during initialization of the Message Object before MSGVAL is set to "1".

ID28-0: Message Identifier

Bit	Description
ID28 - ID0	29-bit Identifier ("Extended Frame").
ID28 - ID18	11-bit Identifier ("Standard Frame").

MSK28-0: Identifier Mask

Bit	Description
0	The corresponding bit in the identifier of the message object cannot inhibit the match in the acceptance filtering.
1	The corresponding identifier bit is used for acceptance filtering.

XTD: Extended Identifier

Bit	Description
0	The 11-bit ("standard") Identifier will be used for this Message Object.
1	The 29-bit ("extended") Identifier will be used for this Message Object.

MXTD: Mask Extended Identifier

Bit	Description
0	The extended identifier bit (XTD) has no effect on the acceptance filtering
1	The extended identifier bit (XTD) is used for acceptance filtering.

Note:

When 11-bit ("standard") Identifiers are used for a Message Object, the identifiers of received Data Frames are written into bits ID28 to ID18. For acceptance filtering, only these bits together with mask bits MSK28 to MSK18 are considered.

DIR: Message Direction

Bit	Description
0	Direction = receive: On TXRQST, a Remote Frame with the identifier of this Message Object is transmitted. On reception of a Data Frame with matching identifier, that message is stored in this Message Object.
1	Direction = transmit: On TXRQST, the respective Message Object is transmitted as a Data Frame. On reception of a Remote Frame with matching identifier, the TXRQST bit of this Message Object is set (if RMTEN = one).

MDIR: Mask Message Direction

Bit	Description
0	The message direction bit (DIR) has no effect on the acceptance filtering.
1	The message direction bit (DIR) is used for acceptance filtering.

The Arbitration Registers ID28-0, XTD, and DIR are used to define the identifier and type of outgoing messages and are used (together with the mask registers MSK28-0, MXTD and MDIR) for acceptance filtering of incoming messages. A received message is stored into the valid Message Object with matching identifier and Direction = receive (Data Frame) or Direction = transmit(Remote Frame). Extended frames can be stored only in Message Objects with XTD = one, standard frames in Message Objects with XTD = zero. If a received message (Data Frame or Remote Frame) matches with more than one valid Message Object, it is stored into that with the lowest message number. For details see "■ Acceptance Filtering of Received Messages".

EOB: End of Buffer

Bit	Description
0	Message Object belongs to a FIFO Buffer and is not the last Message Object of that FIFO Buffer.
1	Single Message Object or last Message Object of a FIFO Buffer.

Note:

This bit is used to concatenate two ore more Message Objects (up to 32) to build a FIFO Buffer. For single Message Objects (not belonging to a FIFO Buffer) this bit must always be set to 1. For details on the concatenation of Message Objects see "■ Configuration of a FIFO Buffer".

NEWDAT: New Data

Bit	Description
0	No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU.
1	The Message Handler or the CPU has written new data into the data portion of this Message Object.

MSGLST: Message Lost (only valid for Message Objects with direction = receive)

Bit	Description
0	No message lost since last time this bit was reset by the CPU.
1	The Message Handler stored a new message into this object when NewDat was still set, the CPU has lost a message.

RXIE: Receive Interrupt Enable

Bit	Description
0	INTPND will be left unchanged after a successful reception of a frame.
1	INTPND will be set after a successful reception of a frame.

TXIE: Transmit Interrupt Enable

Bit	Description
0	INTPND will be left unchanged after the successful transmission of a frame.
1	INTPND will be set after a successful transmission of a frame.

INTPND: Interrupt Pending

Bit	Description
0	This message object is not the source of an interrupt.
1	This message object is the source of an interrupt. The Interrupt Identifier in the Interrupt Register will point to this message object if there is no other interrupt source with higher priority.

RMTEN: Remote Enable

Bit	Description
0	At the reception of a Remote Frame, TXRQST is left unchanged.
1	At the reception of a Remote Frame, TXRQST is set.

TXRQST: Transmit Request

Bit	Description
0	This Message Object is not waiting for transmission.
1	The transmission of this Message Object is requested and is not yet done.

Note:

When the lowest priority message buffer is used for transmission, setting TXRQST = 0 may cause a delay of transmission, when TXRQST is set to "1" again.
Depending on the exact time when TXRQST was set to 0, the message may not be transmitted immediately after setting TXRQST = 1, but after any of the following events:

1. there is activity ongoing on the CANbus
2. a transmission request is issued on another message object
3. the CANbus is initialized by the INIT bit

In general, there is no need to cancel an ongoing transmission by setting TXRQST = 0. If the content of a message object needs to be changed while TXRQST = 1, it is sufficient to update the message object via the CPU interface registers (Identifier, DLC, Data, with TXRQST and NEWDAT, optionally TXIE). The updated content will be transmitted at the next opportunity.

DLC3-0: Data Length Code

Bit	Description
0-8	Data Frame has 0-8 data bytes.
9-15	Data Frame has 8 data bytes.

Note:

The Data Length Code of a Message Object must be defined the same as in all the corresponding objects with the same identifier at other nodes. When the Message Handler stores a data frame, it will write the DLC to the value given by the received message.

Data0	1st data byte of a CAN Data Frame
Data1	2nd data byte of a CAN Data Frame
Data2	3rd data byte of a CAN Data Frame
Data3	4th data byte of a CAN Data Frame
Data4	5th data byte of a CAN Data Frame
Data5	6th data byte of a CAN Data Frame
Data6	7th data byte of a CAN Data Frame
Data7	8th data byte of a CAN Data Frame

Note:

Byte Data0 is the first data byte shifted into the shift register of the CAN Core during a reception, byte Data7 is the last. When the Message Handler stores a Data Frame, it will write all the eight data bytes into a Message Object. If the Data Length Code is less than 8, the remaining bytes of the Message Object will be overwritten by non specified values.

9. Message Handler Registers

All Message Handler registers are read-only. Their contents (TXRQST, NEWDAT, INTPND, and MSGVAL bits of each Message Object and the Interrupt Identifier) is status information provided by the Message Handler FSM.

■ List of Message Interface Registers

Abbreviated Register Name	Register Name	Reference
INTRn	Interrupt Register	See 9.1
TREQR1n, TREQR2n	Transmission Request Register	See 9.2
NEWDT1n, NEWDT2n	New Data Register	See 9.3
INTPND1n, INTPND2n	Interrupt Pending Register	See 9.4
MSGVAL1n, MSGVAL2n	Message Valid Register	See 9.5

9.1. Interrupt Register (INTRn)

The Interrupt Register (INTRn) points to the pending interrupt of highest priority.

■ Interrupt Register (INTRn)

INTRH								
bit	15	14	13	12	11	10	9	8
	INTID15-8							
Attribute	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

INTRL								
bit	7	6	5	4	3	2	1	0
	INTID7-0							
Attribute	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

■ Function of the Interrupt Register (INTRn)

(For 32 message buffer CANs)

[bit15 to bit0] INTID: Interrupt Identifier (the number here indicates the source of the interrupt)

Bit	Description
0x0000	No interrupt is pending.
0x0001-0x0020	Number of Message Object which caused the interrupt.
0x0021-0x7FFF	unused.
0x8000	Status Interrupt.
0x8001-0xFFFF	unused.

(For 128 message buffer CANs)

[bit15 to bit0] INTID: Interrupt Identifier (the number here indicates the source of the interrupt)

Bit	Description
0x0000	No interrupt is pending.
0x0001-0x0080	Number of Message Object which caused the interrupt.
0x0081-0x7FFF	unused.
0x8000	Status Interrupt.
0x8001-0xFFFF	unused.

If several interrupts are pending, the CAN Interrupt Register will point to the pending interrupt with the highest priority, disregarding their chronological order. An interrupt remains pending until the CPU has cleared it. If IntId is different from 0x0000 and IE is set, the interrupt line to the CPU is active. The interrupt line remains active until IntId is back to value 0x0000 (the cause of the interrupt is reset) or until IE is reset.
The Status Interrupt has the highest priority. Among the message interrupts, the Message Object's interrupt priority decreases with increasing message number.
A message interrupt is cleared by clearing the Message Object's INTPND bit. The Status Interrupt is cleared by reading the Status Register.

9.2. Transmission Request Registers (TREQR1n, TREQR2n)

The Transmission Request Registers (TREQR1n, TREQR2n) control the transmission of message objects.

■ Transmission Request Registers (TREQR1n, TREQR2n)

TREQR1Hn								
bit	15	14	13	12	11	10	9	8
	TXRQST16-9							
Attribute	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

TREQR1Ln								
bit	7	6	5	4	3	2	1	0
	TXRQST8-1							
Attribute	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

TREQR2Hn								
bit	15	14	13	12	11	10	9	8
	TXRQST32-25							
Attribute	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

TREQR2Ln								
bit	7	6	5	4	3	2	1	0
	TXRQST24-17							
Attribute	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

■ Function of the Transmission Request Registers (TREQR1n, TREQR2n)

[bit32 to bit1] TXRQST: Transmission Request Bits (of all Message Objects)

Bit	Description
0	This Message Object is not waiting for transmission.
1	The transmission of this Message Object is requested and is not yet done.

These registers hold the TXRQST bits of the 32 Message Objects. By reading out the TXRQST bits, the CPU can check for which Message Object a Transmission Request is pending. The TXRQST bit of a specific Message Object can be set/reset by the CPU via the IFx Message Interface Registers or by the Message Handler after reception of a Remote Frame or after a successful transmission.

Note:

When the lowest priority message buffer is used for transmission, setting TXRQST = "0" may cause a delay of transmission, when TXRQST is set to "1" again.
Depending on the exact time when TXRQST was set to 0, the message may not be transmitted immediately after setting TXRQST = "1", but after any of the following events:

1. there is activity ongoing on the CANbus
2. a transmission request is issued on another message object
3. the CANbus is initialized by the INIT bit

In general, there is no need to cancel an ongoing transmission by setting TXRQST = 0. If the content of a message object needs to be changed while TXRQST = 1, it is sufficient to update the message object via the CPU interface registers (Identifier, DLC, Data, with TXRQST and NEWDAT, optionally TXIE). The updated content will be transmitted at the next opportunity.

If more than 32 message buffers are implemented, the following table gives an overview about the additional flags:

Table 9-1 Additional Flags when more than 32 Message Buffers Exist

Register	Bit	addr+0	addr+1	addr+2	addr+3
TREQR 4 & 3	TXRQST 64-33 (address 0x84)	TXRQST64-57	TXRQST56-49	TXRQST48-41	TXRQST40-33
TREQR 6 & 5	TXRQST 96-65 (address 0x88)	TXRQST96-89	TXRQST88-81	TXRQST80-73	TXRQST72-65
TREQR 8 & 7	TXRQST 128-97 (address 0x8C)	TXRQST128-121	TXRQST120-113	TXRQST112-105	TXRQST104-97

9.3. New Data Registers (NEWDT1n, NEWDT2n)

The New Data Registers (NEWDT1n, NEWDT2n) indicate per message buffer that new data has been received.

■ New Data Registers (NEWDT1n, NEWDT2n)

NEWDT1Hn								
bit	15	14	13	12	11	10	9	8
	NEWDAT16-9							
Attribute	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

NEWDT1Ln								
bit	7	6	5	4	3	2	1	0
	NEWDAT8-1							
Attribute	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

NEWDT2Hn								
bit	15	14	13	12	11	10	9	8
	NEWDAT32-25							
Attribute	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

NEWDT2Ln								
bit	7	6	5	4	3	2	1	0
	NEWDAT24-17							
Attribute	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

■ Function of the New Data Registers (NEWDT1n, NEWDT2n)

[bit32 to bit1] NEWDAT: New Data Bits (of all Message Objects)

Bit	Description
0	No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU.
1	The Message Handler or the CPU has written new data into the data portion of this Message Object.

These registers hold the NEWDAT bits of the 32 Message Objects. By reading out the NEWDAT bits, the CPU can check for which Message Object the data portion was updated. The NEWDAT bit of a specific Message Object can be set/reset by the CPU via the IFx Message Interface Registers or by the Message Handler after reception of a Data Frame or after a successful transmission.

If more than 32 message buffers are implemented, the following table gives an overview about the additional flags:

Table 9-2 Additional Flags when more than 32 Message Buffers Exist

Register	Bit	addr+0	addr+1	addr+2	addr+3
NEWDT 4 & 3	NEWDAT 64-33 (address 0x94)	NEWDAT64-57	NEWDAT56-49	NEWDAT48-41	NEWDAT40-33
NEWDT 6 & 5	NEWDAT 96-65 (address 0x98)	NEWDAT96-89	NEWDAT88-81	NEWDAT80-73	NEWDAT72-65
NEWDT 8 & 7	NEWDAT 128-97 (address 0x9C)	NEWDAT128-121	NEWDAT120-113	NEWDAT112-105	NEWDAT104-97

9.4. Interrupt Pending Registers (INTPND1n, INTPND2n)

The Interrupt Pending Registers (INTPND1n, INTPND2n) indicate whether a message object caused an interrupt or not.

■ Interrupt Pending Registers (INTPND1n, INTPND2n)

INTPND1Hn								
bit	15	14	13	12	11	10	9	8
	INTPND16-9							
Attribute	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

INTPND1Ln								
bit	7	6	5	4	3	2	1	0
	INTPND8-1							
Attribute	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

INTPND2Hn								
bit	15	14	13	12	11	10	9	8
	INTPND32-25							
Attribute	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

INTPND2Ln								
bit	7	6	5	4	3	2	1	0
	INTPND24-17							
Attribute	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

■ Function of Interrupt Pending Registers (INTPND1n, INTPND2n)

[bit32 to bit1] INTPND: Interrupt Pending Bits (of all Message Objects)

Bit	Description
0	This message object is not the source of an interrupt.
1	This message object is the source of an interrupt.

These registers hold the INTPND bits of the 32 Message Objects. By reading out the INTPND bits, the CPU can check for which Message Object an interrupt is pending. The INTPND bit of a specific Message Object can be set/reset by the CPU via the IFx Message Interface Registers or by the Message Handler after reception or after a successful transmission of a frame. This will also affect the value of INTID in the Interrupt Register INTRn.

If more than 32 message buffers are implemented, the following table gives an overview about the additional flags:

Table 9-3 Additional Flags when more than 32 Message Buffers Exist

Register	Bit	addr+0	addr+1	addr+2	addr+3
INTPND 4 & 3	INTPND 64-33 (address 0xA4)	INTPND64-57	INTPND56-49	INTPND48-41	INTPND40-33
INTPND 6 & 5	INTPND 96-65 (address 0xA8)	INTPND96-89	INTPND88-81	INTPND80-73	INTPND72-65
INTPND 8 & 7	INTPND 128-97 (address 0xAC)	INTPND128-121	INTPND120-113	INTPND112-105	INTPND104-97

9.5. Message Valid Registers (MSGVAL1n, MSGVAL2n)

The Message Valid Registers (MSGVAL1n, MSGVAL2n) show the validity status for each message object.

■ Message Valid Registers (MSGVAL1n, MSGVAL2n)

MSGVAL1Hn								
bit	15	14	13	12	11	10	9	8
	MSGVAL16-9							
Attribute	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

MSGVAL1Ln								
bit	7	6	5	4	3	2	1	0
	MSGVAL8-1							
Attribute	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

MSGVAL2Hn								
bit	15	14	13	12	11	10	9	8
	MSGVAL32-25							
Attribute	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

MSGVAL2Ln								
bit	7	6	5	4	3	2	1	0
	MSGVAL24-17							
Attribute	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

■ Function of the Message Valid Registers (MSGVAL1n, MSGVAL2n)

[bit32] MSGVAL: Message Valid Bits (of all Message Objects)

Bit	Description
1	This Message Object is configured and should be considered by the Message Handler.

These registers hold the MSGVAL bits of the 32 Message Objects. By reading out the MSGVAL bits, the CPU can check which Message Object is valid. The MSGVAL bit of a specific Message Object can be set/reset by the CPU via the IFx Message Interface Registers.

If more than 32 message buffers are implemented, the following table gives an overview about the additional flags:

Table 9-4 Additional Flags when more than 32 Message Buffers Exist

Register	Bit	addr+0	addr+1	addr+2	addr+3
MSGVAL 4 & 3	MSGVAL 64-33 (address 0xB4)	MSGVAL64-57	MSGVAL56-49	MSGVAL48-41	MSGVAL40-33
MSGVAL 6 & 5	MSGVAL 96-65 (address 0xB8)	MSGVAL96-89	MSGVAL88-81	MSGVAL80-73	MSGVAL72-65
MSGVAL 8 & 7	MSGVAL 128-97 (address 0xBC)	MSGVAL128-121	MSGVAL120-113	MSGVAL112-105	MSGVAL104-97

CHAPTER: CLOCK OUTPUT FUNCTION

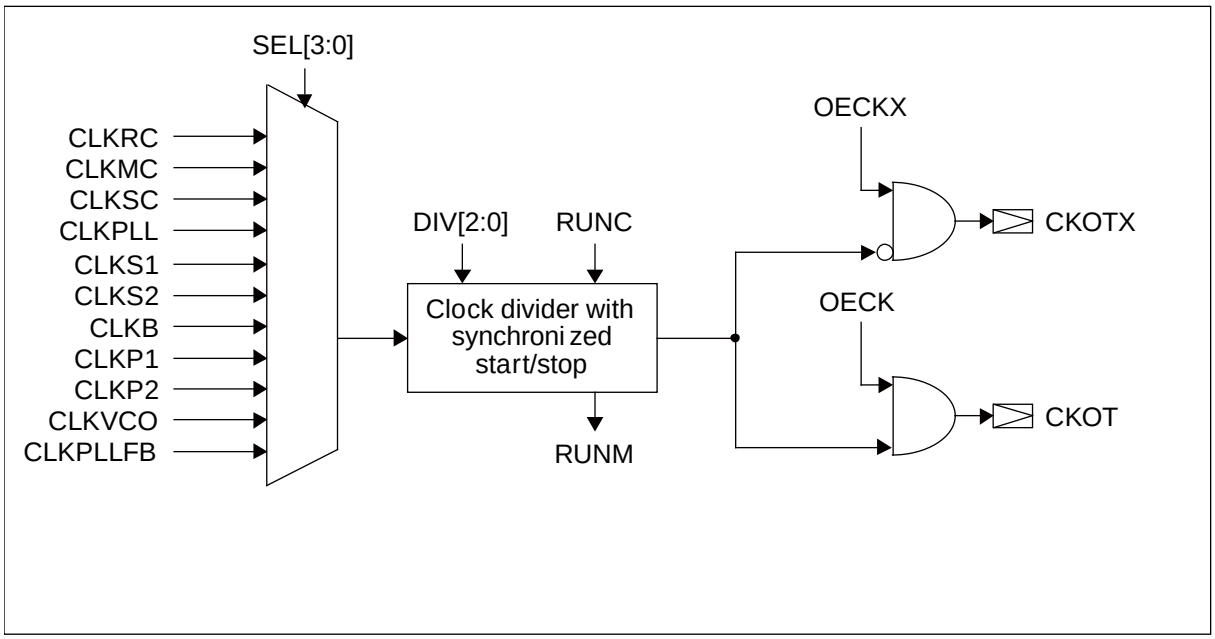
This chapter describes the functions and operations of the clock output function.

Management Code: 96F6CLKOUT-E01.0

1. Overview

The clock output function can be used to output up to two internal clocks to the four clock output pins CKOT0/CKOTX0 and CKOT1/CKOTX1 (one non-inverted and one inverted output pin for each clock). The clocks can be output directly or divided with the internal clock divider.

■ Block Diagram of the Clock Output Function (one channel)



For the definition of the clocks that can be selected, see "CHAPTER 6 CLOCKS".

2. Operation

The clock output function can be used to output two independent clocks at the CKOT0/CKOTX0 and CKOT1/CKOTX1 output pins. The clocks can be activated and deactivated synchronously.

■ Activating the Clock Output Function

Activate the clock output function for the CKOT0/CKOTX0 and CKOT1/CKOTX1 pins as follows:

- Configure the corresponding COCR0, COCR1 register by selecting the requested clock and division value.
- Enable the CKOT and/or CKOTX output pin by setting the corresponding output enable bit to "1". The start bit RUNC0, RUNC1 should be set to "1" at the same time or after setting the output enable.

The CKOT0, CKOT1 pins output "0" and the CKOTX0, CKOTX1 pins output "1" directly after activating the output enable.
After setting the RUNC0, RUNC1 start bit to "1", the synchronization circuit is activated and starts outputting the selected clock at the corresponding CKOT0, CKOT1 / CKOTX0, CKOTX1 pins. Reading the RUNM1, RUNM0 bit now as "0" indicates that the output clock is not yet started (synchronization mechanism is still in effect) and reading "1" means that the output clock is started.

■ Disabling the Clock Output Function

Disable the clock output function as follows:

- Stop the clock output by setting the RUNC0, RUNC1 bit to "0". This stopping is done by using the synchronization circuit. Confirm that the clock is actually stopped by reading the RUNM0, RUNM1 bit (set to "0").
- When the clock is stopped, it is allowed to change the configuration of the COCR register or to disable the resource output by setting the output enable bit to "0".

The output value at the CKOT0, CKOT1 / CKOTX0, CKOTX1 pins after stopping the clock is again "0" for the CKOT0, CKOT1 pins and "1" for the CKOTX0, CKOTX1 pins.
The synchronization circuit for starting/stopping the output clock needs an active source clock. If the source clock selected by the SEL[3:0] bits is disabled, then the status of the output clock cannot be changed.
Setting the RUNC0, RUNC1 bit to another value is possible, however the RUNM monitor bit will not change its value.

■ Changing the Clock Output Configuration Register Contents

The COCR0, COCR1 register should only be written if the corresponding clock output is stopped (RUNC0=0, RUNC1=0). Otherwise a spike can be output.

3. Configuration and Activation Registers

The Clock Output Configuration Register (COCR) selects the output clock and controls the clock divider. The COCR0 register controls the CKOT0/CKOTX0 output clock and the COCR1 register controls the CKOT1/CKOTX1 output clock.
The Clock Output Activation Register (COAR) enables the CKOT0, CKOTX0, CKOT1 and CKOTX1 output pins and synchronously starts and stops the selected clocks.

■ List of Clock Output Function Registers

Abbreviated Register Name	Register Name	Reference
COCR	Clock Output Configuration Register	See 3.1
COAR	Clock Output Activation Register	See 3.2

3.1. Clock Output Configuration Register (COCR)

■ Clock Output Configuration Register (COCR)

COCR								
bit	15/7	14/6	13/5	12/4	11/3	10/2	9/1	8/0
	RESV	DIV2	DIV1	DIV0	SEL3	SEL2	SEL1	SEL0
Attribute	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	X	0	0	0	0	0	0	0

[bit15, bit7] RESV: Reserved Bit

- Write always "0".
- Read value is undefined.
- Read modify write operations to this register have no effect on this bit.

[bit14 to bit12, bit6 to bit4] DIV2 to DIV0: Clock Division Bits

- These bits configure the clock output divider according to the following table:

bit14, bit6	bit13, bit5	bit12, bit4	
DIV2	DIV1	DIV0	Description
0	0	0	Output clock is not divided
0	0	1	Output clock is divided by 2
0	1	0	Output clock is divided by 4
0	1	1	Output clock is divided by 8
1	0	0	Output clock is divided by 16
1	0	1	Output clock is divided by 32
1	1	0	Output clock is divided by 64
1	1	1	Output clock is divided by 128

- Any reset initializes these bits to "000" (Output clock not divided).
- If the setting of these bits is changed while the clock output function is active (corresponding RUNC and OE bits set to "1"), a spike can be output.
- A read access returns the setting value of this control bit.

[bit11 to bit8, bit3 to bit0] SEL3 to SEL0: Clock Select Bits

- These bits select which clock is output to the CKOT0/CKOT1 pins according to the following table:

bit11, bit3	bit10, bit2	bit9, bit1	bit8, bit0	
SEL3	SEL2	SEL1	SEL0	Description
0	0	0	0	No clock
0	0	0	1	CLKRC (RC clock)
0	0	1	0	CLKMC (Main clock)
0	0	1	1	CLKSC (Sub clock)
0	1	0	0	CLKPLL (PLL output clock)
0	1	0	1	Reserved
0	1	1	0	CLKS1 (System clock 1)
0	1	1	1	CLKS2 (System clock 2)
1	0	0	0	CLKB (Bus clock)
1	0	0	1	CLKP1 (Peripheral clock 1)
1	0	1	0	CLKP2 (Peripheral clock 2)
1	0	1	1	CLKVCO (VCO clock of PLL)
1	1	0	0	CLKPLLFB (PLL feedback clock)
1	1	0	1	Reserved
1	1	1	0	Reserved
1	1	1	1	Reserved

- Any reset initializes these bits to "0000" (no clock is selected).
- If the setting of these bits is changed while the clock output function is active (corresponding RUNC and OE bits set to "1"), a spike can be output.
- A read access returns the setting value of this control bit.

3.2. Clock Output Activation Register (COAR)

■ Clock Output Activation Register (COAR)

COAR								
bit	15	14	13	12	11	10	9	8
	RUNM1	RUNC1	CKOXE1	CKOE1	RUNM0	RUNC0	CKOXE0	CKOE0
Attribute	R	R/W	R/W	R/W	R	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

[bit15] RUNM1: Clock Run Monitor Bit Channel 1

Bit	Description
0	CKOT1/CKOTX1 clock is stopped
1	CKOT1/CKOTX1 clock is active

- Reading this bit returns the status of the start/stop synchronization circuit and indicates if starting/stopping the clock has finished.
- Reading "1" means that the CKOT1/CKOTX1 clock output is started.
- Reading "0" means that the CKOT1/CKOTX1 clock output is stopped (the CKOT1 pin outputs a low level and the CKOTX1 pin outputs a high level).
- Writing to this bit has no effect.

[bit14] RUNC1: Clock Run Control Bit Channel 1

Bit	Description
0	Stops the CKOT1/CKOTX1 clock
1	Starts the CKOT1/CKOTX1 clock

- This bit controls the synchronous start/stop function of the CKOT1/CKOTX1 clock outputs.
- Setting this bit to "1" starts the clock outputs and writing "0" stops the clock outputs.
- This bit is initialized to "0" by any reset (clock outputs stopped).
- The RUNC1 and the CKOE1 and/or CKOXE1 bits must be set to "1" in order to output the selected clock at the CKOT1 and/or CKOTX1 pins.
- The RUNC1 bit can be written at any time. However the starting and stopping takes some clock cycles and is performed only when the selected clock is active.
- A read access returns the setting value of this control bit. Use the RUNM1 monitor bit to check if the synchronized starting/stopping has finished.

[bit13] CKOXE1: Clock Output Enable Bit for CKOTX1 Pin

Bit	Description
0	CKOTX1 clock output is disabled
1	CKOTX1 clock output is enabled

- This bit controls the resource output enable of the inverted clock output function.
- Writing "1" to this bit enables the CKOTX1 output pin. Do not enable any other resource output shared with the CKOTX1 pin when setting this bit to "1".
- Setting this bit to "0" disables the clock output (initial value).
- A read access returns the bit setting.
- If the setting of this bit is changed while the RUNM1 bit is "1", a spike can be output.

[bit12] CKOE1: Clock Output Enable Bit for CKOT1 Pin

Bit	Description
0	CKOT1 clock output is disabled
1	CKOT1 clock output is enabled

- This bit controls the resource output enable of the direct clock output function.
- Writing "1" to this bit enables the CKOT1 output pin. Do not enable any other resource output shared with the CKOT1 pin when setting this bit to "1".
- Setting this bit to "0" disables the clock output (initial value).
- A read access returns the bit setting.
- If the setting of this bit is changed while the RUNM1 bit is "1", a spike can be output.

[bit11] RUNM0: Clock Run Monitor Bit Channel 0

Bit	Description
0	CKOT0/CKOTX0 clock is stopped
1	CKOT0/CKOTX0 clock is active

- Reading this bit returns the status of the start/stop synchronization circuit and indicates if starting/stopping the clock output has finished.
- Reading "1" means that the CKOT0/CKOTX0 clock output is started.
- Reading "0" means that the CKOT0/CKOTX0 clock output is stopped (the CKOT0 pin outputs a low level and the CKOTX0 pin outputs a high level).
- Writing to this bit has no effect.

[bit10] RUNC0: Clock Run Control Bit Channel 0

Bit	Description
0	Stops the CKOT0/CKOTX0 clock
1	Starts the CKOT0/CKOTX0 clock

- This bit controls the synchronous start/stop function of the CKOT0/CKOTX0 clock outputs.
- Setting this bit to "1" starts the clock outputs and writing "0" stops the clock outputs.
- This bit is initialized to "0" by any reset (clock outputs stopped).
- The RUNC0 and the CKOE0 and/or CKOXE0 bits must be set to "1" in order to output the selected clock at the CKOT0 and/or CKOTX0 pins.
- The RUNC0 bit can be written at any time. However the starting and stopping takes some clock cycles and is performed only when the selected clock is active.
- A read access returns the setting value of this control bit. Use the RUNM0 monitor bit to check if the synchronized starting/stopping has finished.

[bit9] CKOXE0: Clock Output Enable Bit for CKOTX0 Pin

Bit	Description
0	CKOTX0 clock output is disabled
1	CKOTX0 clock output is enabled

- This bit controls the resource output enable of the inverted clock output function.
- Writing "1" to this bit enables the CKOTX0 output pin. Do not enable any other resource output shared with the CKOTX0 pin when setting this bit to "1".
- Setting this bit to "0" disables the clock output (initial value).
- A read access returns the bit setting.
- If the setting of this bit is changed while the RUNM0 bit is "1", a spike can be output.

[bit8] CKOE0: Clock Output Enable Bit for CKOT0 Pin

Bit	Description
0	CKOT0 clock output is disabled
1	CKOT0 clock output is enabled

- This bit controls the resource output enable of the direct clock output function.
- Writing "1" to this bit enables the CKOT0 output pin. Do not enable any other resource output shared with the CKOT0 pin when setting this bit to "1".
- Setting this bit to "0" disables the clock output (initial value).
- A read access returns the bit setting.
- If the setting of this bit is changed while the RUNM0 bit is "1", a spike can be output.

CHAPTER: REAL TIME CLOCK

This chapter explains the functions and operations of the real time clock.

Management Code: 96F3RTC-E01.0

1. Overview

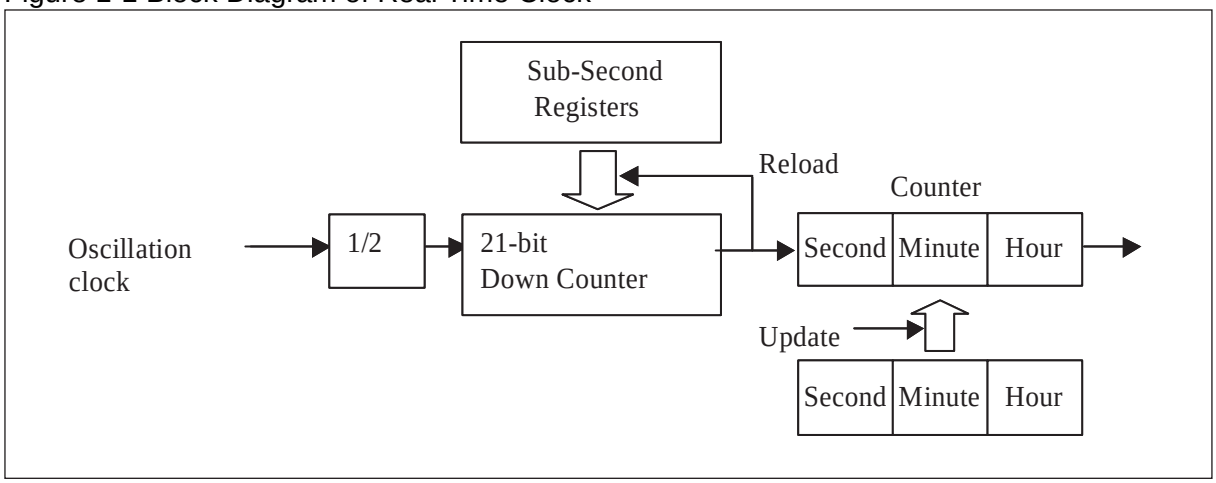
The Real Time Clock consists of the Timer Control register, Sub-second register, Second/Minute/Hour registers, 1/2 clock divider, 21-bit prescaler and Second/Minute/Hour counters.
Real Time Clock (RTC) continues to count elapsed time even in the timer mode to provide the current real time (HH/MM/SS) based on main oscillation (4MHz), sub oscillation (32kHz) or RC oscillation (~100kHz/~2MHz). This allows precise time counting without a return from an interrupt during stand by periods.

■ Features of Real Time Clock

- Information: Time count (HH/MM/SS). (This clock continues to operate even in Timer Mode.)
- Operational on main oscillation (4MHz), sub oscillation (32kHz) or RC oscillation (~100kHz or ~2MHz)
- Time unit: Selected clock divided by 2
- Operation clock:
  - For register access: CLKP1
  - For time count: Main-clock (CLKMC), Sub-clock (CLKSC) or RC clock (CLKRC)
- Time: Initial setting and adjustment are possible.
- Interrupt: Interrupts can be generated at any of the five intervals: 1 half-second, 1 second, 1 minute, 1 hour and 1 day.
- Others: By changing the value of the sub-second register, interrupts can be generated at any interval (from short to long).

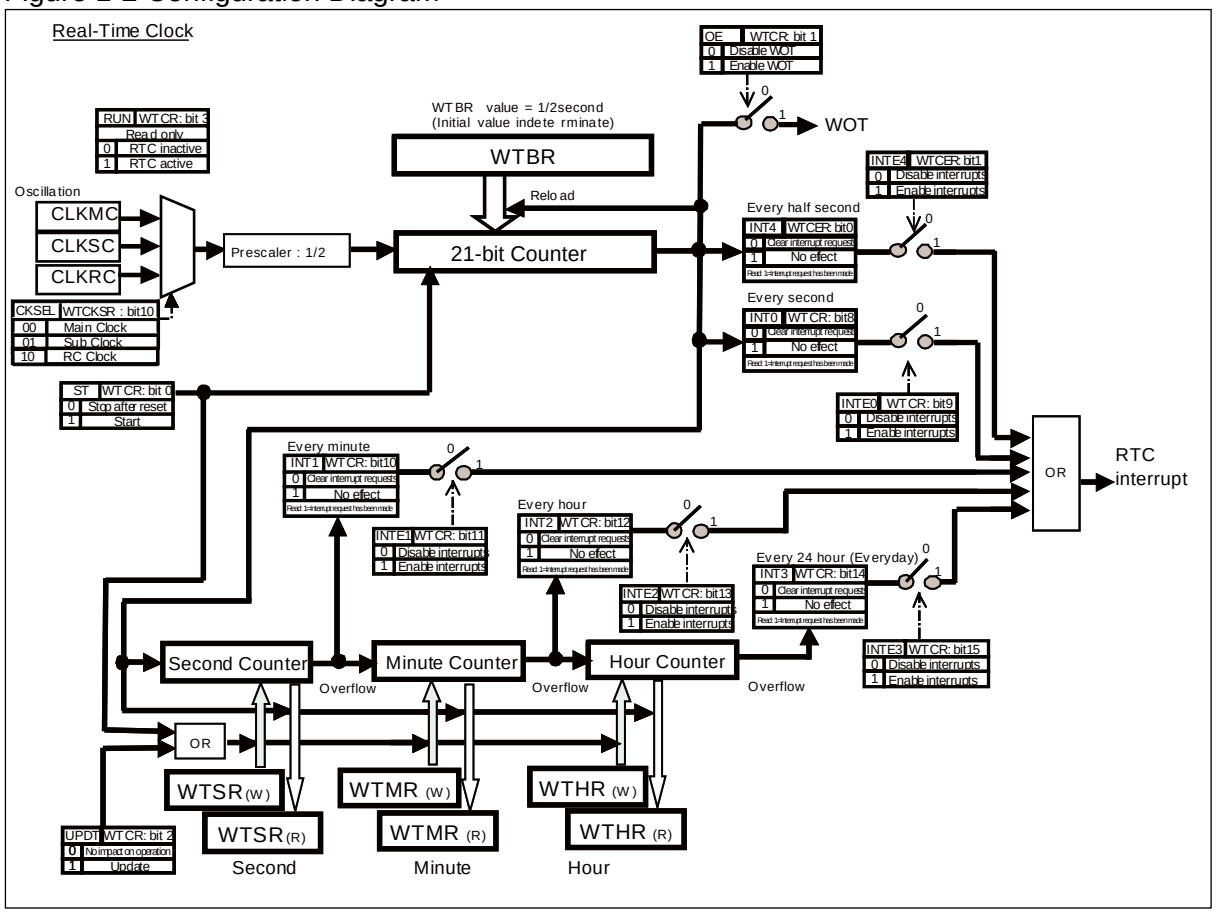
■ Block Diagram of Real Time Clock

Figure 1-1 Block Diagram of Real Time Clock



■ Configuration Diagram

Figure 1-2 Configuration Diagram



2. Operation

This section describes the real time clock operation.

■ Real Time Clock Operation

Figure 2-1 Real Time Clock Operation

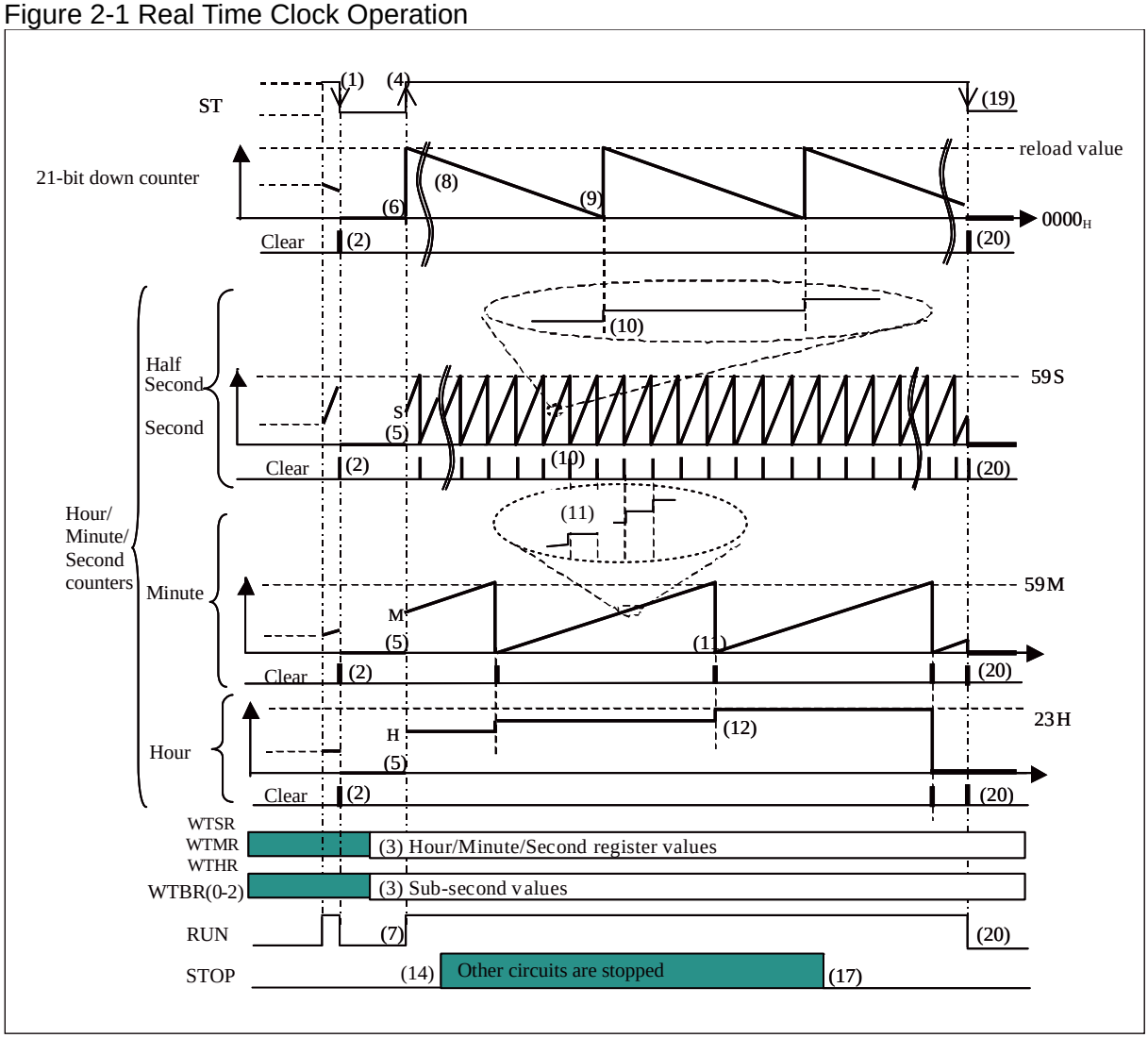
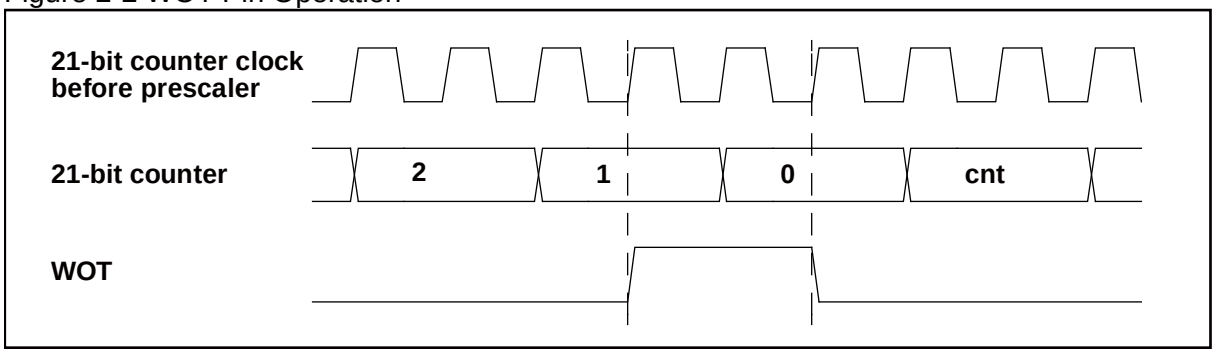


Figure 2-2 WOT Pin Operation



1. The start bit (ST) is set to "1" and then "0". (Register initialization operation)
2. This (ST="0") resets to 0 and stops the 21-bit down counter and the hour/minute/second timers.
3. The software writes hour, minute, and second values to the hour/minute/second registers, WTHR/WTMR/WTSR.
4. The software writes the appropriate values to the sub-second registers, WTBRL0/WTBRH0/WTBR1.
5. The interrupt request bits (INT0, INT1, INT2, INT3 and INT4) are initialized, and the interrupt request enable bits (INTE0, INTE1, INTE2, INTE3 and INTE4) are set to "interrupts enabled".
6. The start bit (ST) is set to "1".
7. This (ST="1") causes the values of the hour/minute/second registers, WTHR/WTMR/WTSR, to be loaded to the hour/minute/second timers.
8. The values of the sub-second registers, WTBRL0/WTBRH0/WTBR1, are loaded to the 21-bit down counter.
9. The run flag (RUN) is set to "1".
10. The 21-bit down counter begins counting at the main clock (CLKMC) divided by 2 (4/2 MHz), subclock (CLKSC) divided by 2 (32.768/2 kHz) or RC clock (CLKRC) divided by 2 (100/2 kHz or 2/2MHz).
11. When the 21-bit down counter reaches "000000_H", the value of the sub-second registers is loaded to the 21-bit down counter, generating a half-second interrupt request. Each second half-second interrupt a second interrupt will be generated.
12. When the second counter counts up to "59", the counter is cleared next time when the counter counts up, at which the minute counter counts up, generating a 1-minute interrupt request.
13. When the minute counter counts up to "59", the counter is cleared next time when the counter counts up, at which the hour counter counts up, generating a 1-hour interrupt request.
14. When the hour counter counts up to "23", the counter is cleared next time when the counter counts up, at which a 1-day interrupt request is generated.
15. The software changes the state of the MCU to Timer Mode by setting the bits SMCR:SMS[1:0] to "10" (see "Section 9.2 Standby Mode Control Register (SMCR)" for details). The real-time clock continues to operate in the Timer Mode.
16. The device recovers from Timer Mode (by interrupt request).
17. This (ST="0") resets and stops the 21-bit down counter and the hour/minute/second counters.

3. Registers

This section explains the configuration and functions of the registers used for the Timer Control.

Note:

WTCR, WTCER, WTCKSR are initialized by all reset causes except unused and reserved bits.
WTBR is not reset.
WTSR, WTMR, WTHR are initialized only by the Power-on reset, Low-voltage reset and external resets and not by any other reset, except for unused bits.

■ List of Timer Control Registers

Abbreviated Register Name	Register Name	Reference
WTCR	Timer Control Register	See 3.1
WTCER	Timer Control Extended Register	See 3.2
WTCKSR	Clock Select Register	See 3.3
WTBR	Sub-Second Register	See 3.4
WTSR, WTMR, WTHR	Second/Minute/Hour Registers	See 3.5

3.1. Timer Control Register (WTCR)

The bit configuration of timer control register is shown.

■ Timer Control Register (WTCR)

WTCR								
bit	15	14	13	12	11	10	9	8
	INTE3	INT3	INTE2	INT2	INTE1	INT1	INTE0	INT0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

bit	7	6	5	4	3	2	1	0
	RESV	RESV	RESV	-	RUN	UPDT	OE	ST
Attribute	R/W	R/W	R/W	-	R	R/W	R/W	R/W
Initial value	X	X	X	X	0	0	0	0

Note:

WTCR is initialized by all reset causes except for reserved or unused bits.

[bit15] INTE3: Enable Interrupt Requests at 1-Day Intervals

When the hour counter overflows, this flag is set to "1".

Bit	Description
0	No interrupt requests
1	Generate interrupt requests at 1-day (24 hour) intervals.

[bit14] INT3: 1-Day Interrupt Request Flag

Bit	Description	
	Read	Write
0	No interrupt requests	Clear the flag.
1	Generate interrupt requests at 1-day (24 hour) intervals.	Writing does not affect the operation.

[bit13] INTE2: Enable Interrupt Requests at 1-Hour Intervals

When the minute counter overflows, this flag is set to "1".

Bit	Description
0	No interrupt requests
1	Generate interrupt requests at 1-hour intervals.

[bit12] INT2: 1-Hour Interrupt Request Flag

Bit	Description	
	Read	Write
0	No interrupt requests	Clear the flag.
1	Generate interrupt requests at 1-hour intervals.	Writing does not affect the operation.

[bit11] INTE1: Enable Interrupt Requests at 1-Minute Intervals

When the minute counter overflows, this flag is set to "1".

Bit	Description
0	No interrupt requests
1	Generate interrupt requests at 1-minute intervals.

[bit10] INT1: 1-Minute Interrupt Request Flag

Bit	Description	
	Read	Write
0	No interrupt requests	Clear the flag.
1	Generate interrupt requests at 1-minute intervals.	Writing does not affect the operation.

[bit9] INTE0: Enable Interrupt Requests at 1-Second Intervals

When the 21-bit down counter is set to "0", this flag is set to "1".

Bit	Description
0	No interrupt requests
1	Generate interrupt requests at 1-second intervals.

[bit8] INT0: 1-Second Interrupt Request Flag

Bit	Description	
	Read	Write
0	No interrupt requests	Clear the flag.
1	Generate interrupt requests at 1-second intervals.	Writing does not affect the operation.

[bit7 to bit5] RESV: Reserved Bits

- Write always "0".
- Read value is undefined.
- Read modify write operations to this register have no effect on this bit.

[bit4] -: Undefined

- Write always "0".
- Read value is undefined.
- Read modify write operations to this register have no effect on this bit.

[bit3] RUN: Operation Status

Bit	Description
0	The Real-time clock module is inactive.
1	The Real-time clock module is active.

[bit2] UPDT: Update

Before writing "1" to the update bit (UPDT), the hour/minute/second registers must be set to the values with which to update the hour/minute/second counters. The hour/minute/second registers are updated on reloading to the 21-bit down counter.

Bit	Description
0	The update has been completed. (Writing "0" does not affect the operation.)
1	Update the hour/minute/second counters with the values of the hour/minute/second registers, respectively.

Note:

Do not write "1" to UPDT bit continuously.Please write "1" to the update bit (UPDT) after next count over occurs in 1/2 clock divider, when hour/minute/second registers are set.
Do not write "1" to RTC start bit(ST) and update bit(UPDT) at the same time when RTC is stopped (ST="0").
Please write "1" to UPDT only when RTC is activated(ST="1")
Do not write "1" to UPDT while RTC is stopped(ST="0"). It is prohibited.

[bit1] OE: Output Enable

Bit	Description
0	The WOT external pin can be used as a general purpose I/O or for another peripheral block.
1	The WOT external pin serves as the output for the 21-bit down counter.

[bit0] ST: Start

Bit	Description
0	The Real-time clock module stops to operate, and the 21-bit down counter and the hour/minute/second counters are cleared.
1	The settings of the hour/minute/second registers are loaded to the hour/minute/second counters, and the Real-time clock module starts to operate.

Application Notes:

The Sub-second register of the RTC module stores the reload value for the 21-bit counter. This value is reloaded after the reload counter reaches "0". When modifying all three bytes, make sure the reload operation will not be performed in between the write instructions. Otherwise the 21-bit prescaler loads the incorrect value of the combination of new data and old data bytes. It is generally recommended that the Sub-Second register is updated while the ST bit is "0".

However, if this update is done immediately after an RTC second interrupt, there should be enough time to securely modify the registers until the next reload operation (next second interrupt) even if ST is not set to "0" and the module is in operation.

When updating the registers by using the ST bit the following must be taken into account:

The new value is written into the registers with the rising edge of the RUN bit. This RUN bit is clocked by the RTC clock (main clock, sub-oscillator clock or RC clock depending on device and mode). To make sure that the update is done properly, write the new values into the registers, set ST to "0", wait for the RUN bit to go low and then start the circuit again by setting ST to "1". RUN will go low at the second rising edge of the RTC clock after ST has been set to "0". It will rise again at the half second rising edge of RTC clock after ST has been set to "1". If this operation is to be done several times directly after each other, wait for RUN to go to high before setting ST to low again.

3.2. Timer Control Extended Register (WTCER)

The bit configuration of timer control extended register is shown.

■ Timer Control Extended Register (WTCER)

WTCER								
bit	7	6	5	4	3	2	1	0
	-	-	-	-	-	-	INTE4	INT4
Attribute	-	-	-	-	-	-	R/W	R/W
Initial value	X	X	X	X	X	X	0	0

Note:

WTCER is initialized by all reset causes except for unused bits.

[bit7 to bit2] -: Undefined

- Write always "0".
- Read value is undefined.
- Read modify write operations to this register have no effect on this bit.

[bit1] INTE4: Enable Interrupt Requests at Half-second (500ms) Intervals

When the 21-bit counter overflows, this flag is set to "1".

Bit	Description
0	No interrupt requests
1	Generate interrupt requests at half-second (500 ms) intervals.

[bit0] INT4: Half-second (500ms) Interrupt Request Flag

Bit	Description	
	Read	Write
0	No interrupt requests	Clear the flag.
1	Generate interrupt requests at half-second (500 ms) intervals.	Writing does not affect the operation.

3.3. Clock Select Register (WTCKSR)

The bit configuration of clock select register is shown.

■ Clock Select Register (WTCKSR)

The Clock Select Register is used to select the clock source for Real Time Clock.

WTCKSR								
bit	15	14	13	12	11	10	9	8
	-	-	-	-	-	-	CKSEL1	CKSEL0
Attribute	-	-	-	-	-	-	R/W	R/W
Initial value	-	-	-	-	-	-	0	0

[bit15 to bit10] -: Undefined

- Write always "0".
- Read value is undefined.
- Read modify write operations to this register have no effect on this bit.

[bit9, bit8] CKSEL[1:0]: Real Time Clock Clock Select Bit

Bit	Description
00	Main clock (CLKMC) used
01	Sub clock (CLKSC) used
10	RC clock (CLKRC) used
11	prohibited

Note:

WTCKSR is initialized by all reset causes except for unused bits.

Application Notes:

When the setting of WTCKSR:CKSEL[1:0] shall be changed, the clock on which the Real time clock is currently running must stay enabled for at least 3 further cycles after change. Otherwise the Real time clock does not change to the new setting. The former clock can be switched off afterwards.

Example: Real time clock shall be changed from Main clock to RC-clock:

Main clock oscillator is enabled after power on. Therefore is must be enabled when WTCKSR:CKSEL[1:0] is changed to "RC clock" by writing "10" to it. Otherwise real time clock cannot switch to RC clock. The main oscillator must stay enabled for at least three main clock cycles after the write access to WTCKSR:CKSEL[1:0].

3.4. Sub-Second Register (WTBR)

The sub-second register stores a reload value for the 21-bit counter that divides the oscillation clock. The reload value is usually set so that the 21-bit counter will output exactly within a half second cycle. This register is not initialized by reset, but 21-bit counter is initialized by reset.

■ Sub-Second Register (WTBR)

WTBRH0

bit	15	14	13	12	11	10	9	8
	D15	D14	D13	D12	D11	D10	D9	D8
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	X	X	X	X	X	X	X	X

WTBRL0

bit	7	6	5	4	3	2	1	0
	D7	D6	D5	D4	D3	D2	D1	D0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	X	X	X	X	X	X	X	X

WTBR1

bit	7	6	5	4	3	2	1	0
	-	-	-	D20	D19	D18	D17	D16
Attribute	-	-	-	R/W	R/W	R/W	R/W	R/W
Initial value	-	-	-	X	X	X	X	X

Application Notes:

The Sub-Second Registers, WTBR, holds values to be reloaded to the 21-bit down counter. When the 21-bit down counter value becomes "0", the settings of WTBR are reloaded to the 21-bit down counter.

Remark: The reload value to be set in the sub-second registers corresponds to the time for half a second. One second is reached after counting twice the reload value set in WTBR.

Table 3-1 Example Configuration of WTBR Registers for Different Clock Configurations

	WTBR1	WTBRH0	WTBRL0
Main oscillator, 4MHz	$0F_H$	42_H	$3F_H$
RC oscillator, 2MHz	07_H	$A1_H$	$1F_H$
RC oscillator, 100kHz	00_H	61_H	$A7_H$
Sub oscillator, 32.768kHz	00_H	$1F_H$	FF_H

3.5. Second/Minute/Hour Registers (WTSR, WTMR, WTHR)

The Second/Minute/Hour registers store the time information. It is a binary representation of the second, minute and hour.
Reading these registers simply returns the counter values. These registers are write accessible however, the written data is loaded in the counters after the UPDT bit is set to "1". These registers and counter are initialized by reset.

■ Second/Minute/Hour Registers

WTSR								
bit	15	14	13	12	11	10	9	8
	-	-	S5	S4	S3	S2	S1	S0
Attribute	-	-	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	X	X	0	0	0	0	0	0

WTHR								
bit	15	14	13	12	11	10	9	8
	-	-	-	H4	H3	H2	H1	H0
Attribute	-	-	-	R/W	R/W	R/W	R/W	R/W
Initial value	X	X	X	0	0	0	0	0

WTMR								
bit	7	6	5	4	3	2	1	0
	-	-	M5	M4	M3	M2	M1	M0
Attribute	-	-	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	X	X	0	0	0	0	0	0

Note:

WTSR, WTMR, WTHR are initialized only by the Power-on reset, Low-voltage reset and external resets and not by any other reset except for unused bits.

Application Notes:

The values written to the hour/minute/second registers are the initial values to be loaded to the hour/minute/second counters.
By setting "1" to the update bit (WTCR:UPDT) or the start bit (WTCR:ST), the hour/minute/second register values are written to the hour/minute/second counters.
The hour/minute/second counter values are saved to the hour/minute/second registers every time when the second counter overflows, that is, at intervals of one minute. When the hour/minute/second counters are read, the saved count values, not written ones, are read.
The hour/minute/second registers consist of two separate sets of registers: one for reading and the other for writing. The write registers get updated when CPU writes into the hour/minute/second registers. The read registers get updated values from the internal logic and these are the values available when CPU tries to read from the hour/minute/second registers anytime. Hence, if it is intented to stop and continue the realtime clock some time after it has been set, make sure to update the write registers with the most recent values of the read registers.
Since there are three byte-registers, make sure the obtained values from the registers are consistent.

For example, obtained value of "1 hour, 59 minute, 59 second" could be "1 hour, 59 minute, 59 second" or "1 hour, 0 minute, 0 second" or "2 hour, 0 minute, 0 second".
Also when the CPU operation clock CLKB is a fraction of the Realtime clock’s operating clock, the read values from these registers may be inconsistent, because they change their value during the read operation. Therefore it is recommended to use the 1-second interrupt to trigger the read instructions, since after this interrupt these registers are not updated for 1 second.
Do not write impossible data (e.g. 60 seconds) into the WTSR, WTMR or WTHR register.

4. Cautions

This section describes the cautions to be considered while using the real time clock.

■ Cautions

- Setting the interrupt request flags (WTCR:INT0, WTCR:INT1, WTCR:INT2, WTCR:INT4 and WTCER:INT4) to "1" due to overflow, and writing "0" to that bit have occurred at the same time, the flag is set to "1" (Flag setting takes precedence).
- Writing "1" to the update bit (WTCR:UPDT) and update completion have occurred at the same time, the update bit (UPDT) is set to "0".
- When the second counter holds the value of 59, even if "1" is written to the update bit (WTCR:UPDT), the hour/minute/second counters are not updated, leaving the update bit to remain "0".
  In order to update the hour/minute/second counters, it is recommended that "0" should be written to the start bit (WTCR:ST), the hour/minute/second counters be cleared to "0", and then "1" be written to the start bit (ST).
- If you stop the peripheral clock (CLKP1) after updating the hour/minute/second counters using the update bit (WTCR:UPDT), read the hour/minute/second registers to confirm that they have been updated before stopping the peripheral clock CLKP1.
- When you start to use the Real Time Clock module, change the start bit (ST) from "1" to "0", and clear the hour/minute/second counters and the 21-bit down counter to "0".
- The Sub-second register of the RTC module stores the reload value for the 21-bit prescaler. This value is reloaded after the reload counter reaches "0". When modifying all three bytes, make sure the reload operation will not be performed in between the write instructions. Otherwise the 21-bit prescaler loads the incorrect value of the combination of new data and old data bytes. It is generally recommended that the Sub-Second register is updated while the ST bit is "0".
- However, if this update is done immediately after an RTC second interrupt there should be enough time to securely modify the registers until the next reload operation (next second interrupt) even if ST is not set to "0" and the module is in operation.
- When updating the registers by using the ST bit the following must be taken into account: The new value is written into the registers with the rising edge of the RUN bit. This RUN bit is clocked by the RTC clock (32 kHz, 100kHz, 2MHz or 4 MHz depending on device and mode). To make sure that the update is done properly, write the new values into the registers, set ST to "0", wait for the RUN bit to go low and then start the circuit again by setting ST to "1". RUN will go low at the second rising edge of the RTC clock after ST has been set to "0". It will rise again at the half second rising edge of RTC clock after ST has been set to "1". If this operation is to be done several times directly after each other, wait for RUN to go to high before setting ST to low again.
- If a reload has occurred during updating the sub-second registers, WTBRL0, WTBRH0, WTBR1, an unexpected value may be reloaded to the 21-bit down counter. Therefore, it is recommended that the sub-second register, WTBR, should be updated with the start bit (WTCR:ST) set to "0".
- If all the sub-second registers, WTBRL0, WTBRH0, WTBR1, are set to "0", the 21-bit down counter does not operate, resulting in the real-time clock module to be inoperational.
- If a carry has occurred during reading from the hour/minute/second registers, WTHR/WTMR/WTSR, inappropriate values may be read. To avoid this, it is recommended that interrupts (INT0 to INT4) should be used to read time (HH/MM/SS).
- In order for the real time clock module to function properly, the frequency of the subclock (CLKSC) or the RC-clock (CLKRC) must be much lower than that of the peripheral clock (CLKP1). If not, correct values cannot be read from WTHR/WTMR/WTSR.
- Note that only byte-access is allowed to these register. So, when these registers are read at the very timing of changing over the hour or minute boundary as shown below, there is a possibility of misjudging the time. So, read several times to get a logically consistent value.
  Example: Read begins at the second register: 02:59:59=> 03:59:59 => 03:00:00
  Read begins at the hour register: 02:59:59 => 02:00:00 => 03:00:00
  In this case, the current time should be interpreted as 3 o’clock.

CHAPTER: CLOCK CALIBRATION UNIT

This chapter explains the functions and operation of the Clock Calibration Unit.

1. Overview

The Clock Calibration Module provides possibilities to calibrate the sub oscillator clock or the RC oscillator clock with respect to the main oscillator clock. This chapter gives an overview of the Clock Calibration Unit, describes the registers and provides some application notes.

■ Description

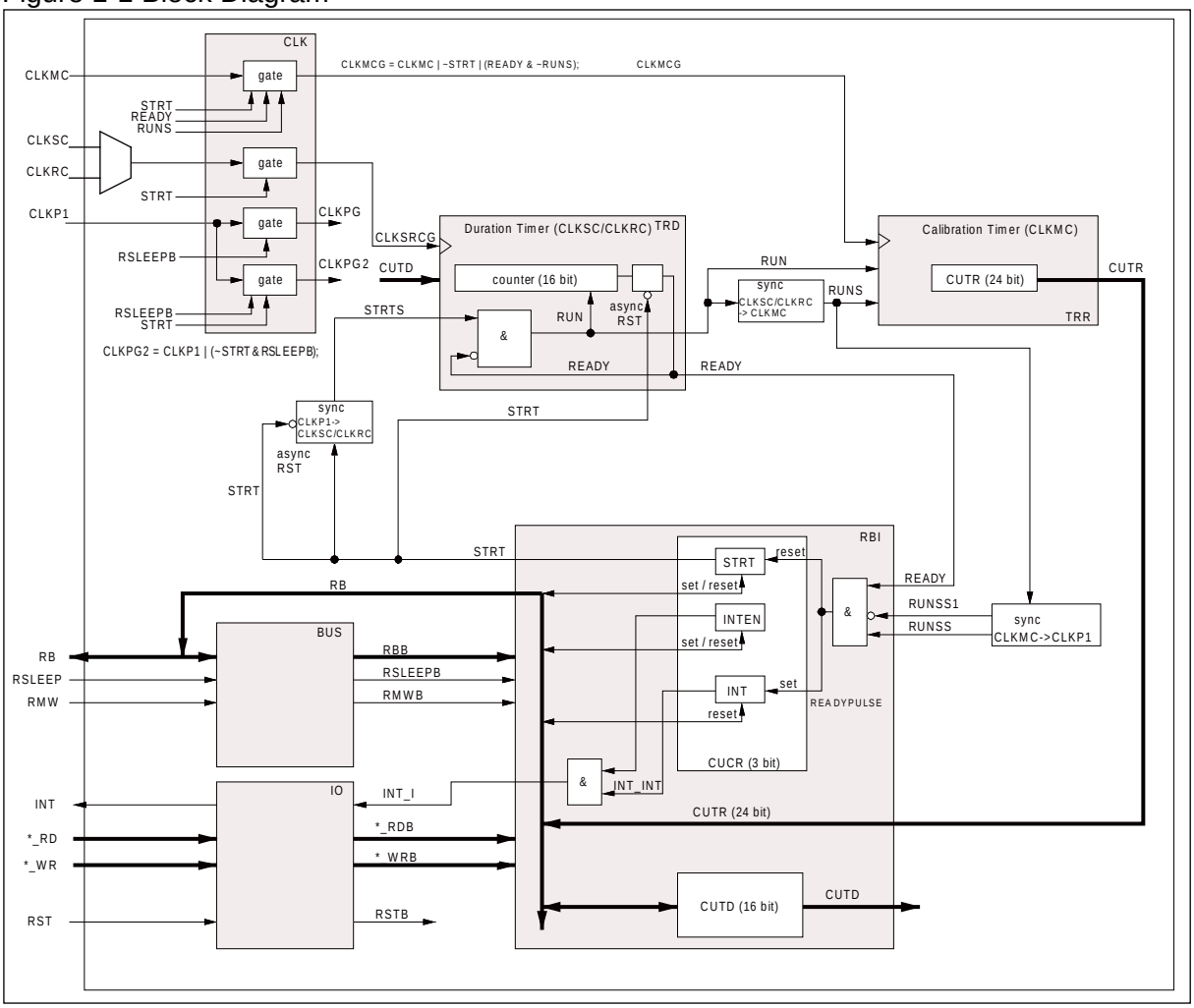
This hardware allows the software to measure time generated by the sub oscillator clock CLKSC or the RC oscillator clock CLKSC with the main oscillator clock CLKMC.

By utilizing this hardware in conjunction with software processing, the accuracy of the sub oscillator clock or RC oscillator clock can come closer to that of the main oscillator clock. The measurement result from the Clock Calibration Unit can be processed by the software and the setting required for the Real Time Clock Module can be obtained.

This module consists of two timers. The Duration Timer operates with the sub oscillator clock CLKSC or RC-clock CLKRC. The Calibration Timer operates with the main oscillator clock CLKMC. Typical values are 32kHz for CLKSC, 4MHz for CLKMC and 100kHz or 2MHz for CLKRC. The Duration Timer triggers the Calibration Timer and the resulting Calibration Timer value is stored into a register. The value stored in this register can be used for the subsequent software processing to calculate the desired Real Time Clock module’s setting.

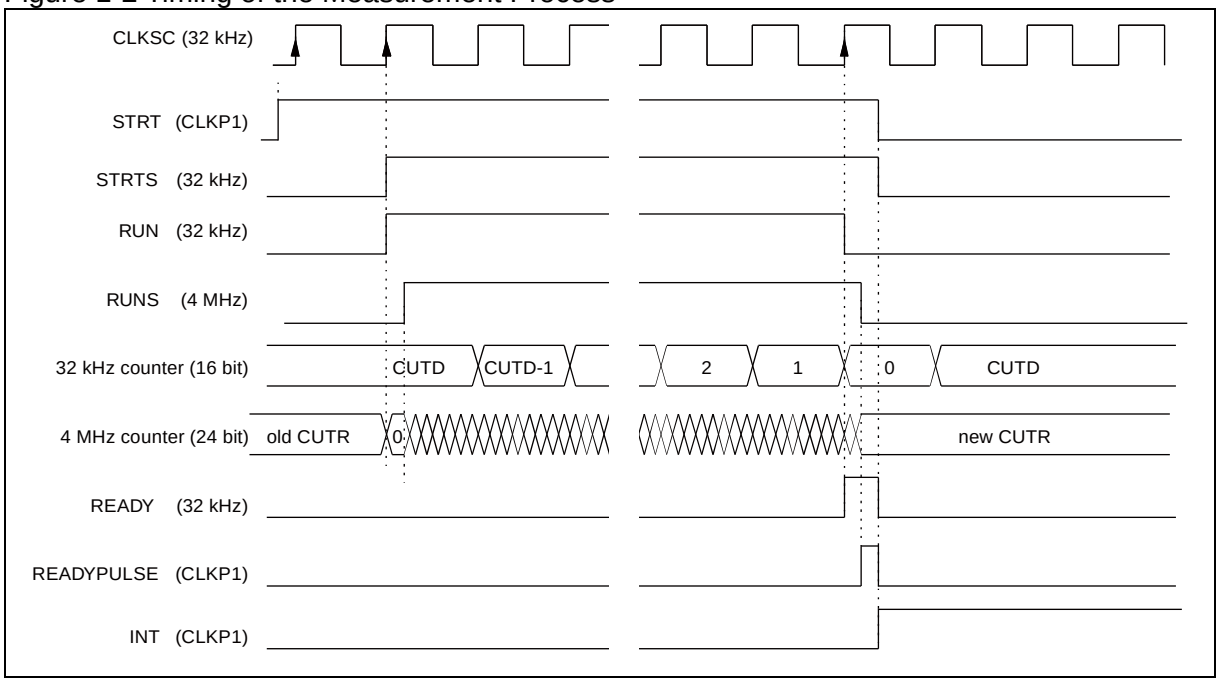
■ Block Diagram

Figure 1-1 Block Diagram



■ Timing

Figure 1-2 Timing of the Measurement Process



■ Clocks

The module operates with 3 different clocks: The main oscillator clock CLKMC, the sub-clock CLKSC (or the RC clock CLKRC) and the peripheral clock CLKP1. For an overview about available clocks, please refer to Figure 1-1. Synchronization circuits adapt the different domains.
All 3 clocks are gated. CLKSC (or CLKRC) and CLKMC clock are switched off if STRT is "0". CLKPG is gated by RSLEEP and CLKPG2 by RSLEEP and STRT for the 2-bit, which are set/reset by hardware.
The clock frequencies have to fulfill the following requirements:

- $T_{CLKSC/CLKRC} > 2 \times T_{CLKMC} + 3 \times T_{CLKP1}$
- $T_{CLKMC} < 1/2 \times T_{CLKSC/CLKRC} - 3/2 \times T_{CLKP1}$
- $T_{CLKP1} < 1/3 \times T_{CLKSC/CLKRC} - 2/3 \times T_{CLKP1}$

Table 1-1 Example of Valid Clock Ratios which Fulfill the Requirements

	CLKSC		CLKRC		CLKMC		CLKP1	
Maximum operation speed	100 kHz	10 us	2 MHz	500 ns	10 MHz	100 ns	50 MHz	20 ns
Normal operation	32 kHz	31.25 us	100 kHz	10 us	4 MHz	250 ns	>2 MHz	500 ns

2. Registers Description

This section explains the configuration and functions of the registers used for the calibration unit.

■ List of Calibration Unit Registers

Abbreviated Register Name	Register Name	Reference
CUCR	Clock Calibration Unit Control Register	See 2.1
CUTD	Clock Calibration Unit Duration Timer Data Register	See 2.2
CUTR	Clock Calibration Unit Calibration Timer Data Register (24-bit)	See 2.3

2.1. Clock Calibration Unit Control Register (CUCR)

■ Clock Calibration Unit Control Register (CUCR)

CUCR								
bit	7	6	5	4	3	2	1	0
	RESV	-	-	STRT	-	CKSEL	INT	INTEN
Attribute	R/W0	-	-	R/W	-	R/W	R/W	R/W
Initial value	X	X	X	0	X	0	0	0

The Clock Calibration Unit Control Register (CUCR) has the following functions:

start / stop calibration measurement
enable / disable interrupt
indicates the end of the calibration measurement

[bit7] RESV: Reserved Bit
Always write "0". The read value is undefined. Read-modify-write is not affected.

[bit6 to bit5] -: Undefined
Always write "0". The read value is undefined. Read-modify-write is not affected.

[bit4] STRT: Calibration Start

0	Calibration stopped, module switched off (default)
1	Start calibration

When the STRT bit is set to "1" by the software, the calibration starts. The Duration Timer starts counting down from the value stored in the Duration Timer Data Register CUTD and the Calibration starts counting up from zero.
When the Duration Timer reaches zero, this bit is reset to "0" by the hardware.
If "0" is written into this bit by the software during the calibration process, the calibration is immediately stopped. If writing "0" by the software and reset to "0" by the hardware happens at the same time, the hardware operation supersede the software operation. This means the calibration is properly completed and the INT bit is set to "1". Writing "1" to this bit during the calibration has no effect.

[bit3] -: Undefined
Always write "0". The read value is undefined. Read-modify-write is not affected.

[bit2] CKSEL: Input Clock Select

0	Use CLKSC (default)
1	Use CLKRC

The RC-clock frequency depends on the setting of CKSCR:RCFS bit (RC-Oscillator frequency select).
Devices with sub clock: If the Clock setting shall be changed, the clock which is currently selected must be active for at lease 3 cycles if it was enabled at any time before. Otherwise the selected clock does not change to the new clock.

[bit1] INT: Interrupt

0	Calibration ongoing / module inactive (default)
1	Calibration completed

This bit indicates the end of the calibration. When the Duration Timer reaches zero after the start of calibration, the Calibration Timer Data Register stores the last Calibration Timer value and the INT bit is set to "1".
The read-modify-write operation to this bit results in reading "1". Writing "0" to this bit clears this flag (INT=0). Writing "1" to this bit has no effect.
The interrupt flag INT is not reset by hardware. Therefore it must be reset by software before starting a new calibration. Otherwise the end of the calibration process is only signalized by the STRT bit (the INT flag stays "1" also during calibration).

[bit0] INTEN: Interrupt Enable

0	Interrupt disabled (default)
1	Interrupt enabled

This is the interrupt enable bit corresponding to the INT bit. When this bit is set to "1" and the INT bit is set by the hardware, the calibration module signals an interrupt to the CPU. The INT-bit itself is not affected by the INTEN bit and is set by hardware even if interrupts are disabled (INTEN=0).

2.2. Clock Calibration Unit Duration Timer Data Register (CUTD)

■ Clock Calibration Unit Duration Timer Data Register (CUTD)

CUTDH								
bit	15	14	13	12	11	10	9	8
	TDD15	TDD14	TDD13	TDD12	TDD11	TDD10	TDD9	TDD8
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	1	0	0	0	0	0	0	0

CUTDL								
bit	7	6	5	4	3	2	1	0
	TDD7	TDD6	TDD5	TDD4	TDD3	TDD2	TDD1	TDD0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

The Clock Calibration Unit Duration Timer Data Register (CUTD) holds the value which determines the duration of calibration, i. e. it stores Duration Timer reload value.When the calibration is started, the stored value is loaded into the Low Speed Timer and the timer starts counting down until it reaches zero. The Clock Calibration Unit Duration Timer operates with CLKSC or with CLKRC clock, depending on the setting of CUCR:CKSEL.
Default value is 0x8000 which corresponds to a measurement duration of 1 second, if CLKSC is used with a 32.768 kHz crystal.
This register should be written only when the calibration is inactive (STRT=0).
If CUTD is initialized with 0x0000 an underflow will occur and the measurement will take (0xFFFF + 1) × $T_{CLKSC/CLKRC}$
In order to achieve a measurement duration of 1 second at a 32kHz oscillation, the CUTD register has to be loaded with 0x8000 = 32768 dec. This number results from the exact oscillation frequency of the crystal, which is F_{osc}=32768 Hz. The ideal values of the measurement results (if a 4.00 MHz crystal is used) are shown in the following table.

32kHz: Ideal Measurement Results Depending on Measurement Duration

Duration of calibration	CUTD value	CUTR value
2 sec	0x0000	0x7A1200
1.75 sec	0xE000	0x6ACFC0
1.5 sec	0xC000	0x5B8D80
1.25 sec	0xA000	0x4C4B40
1 sec	0x8000	0x3D0900
0.75 sec	0x6000	0x2DC6C0
0.5 sec	0x4000	0x1E8480
0.25 sec	0x2000	0x0F4240

In order to achieve a measurement duration of half a second at a 100kHz oscillation, the CUTD register has to be loaded with 0xC350 = 50000 dec. This number results from the exact oscillation frequency of F_{osc}=100000 Hz. The ideal values of the measurement results (if a 4.00 MHz crystal is used) are shown in the following table.

100kHz: Ideal Measurement Results Depending on Measurement Duration

Duration of calibration	CUTD value	CUTR value
0.5 sec	0xC350	0x1E8480
0.25 sec	0x61A8	0x0F4240
0.125 sec	0x30D4	0x07A120
0.1 sec	0x2710	0x061A80

The duration of the whole process from writing a 1 into the STRT bit until STRT is reset by hardware is longer than the actual calibration measurement time, due to synchronization between the different clock domains. Process Duration < (CUTD + 3) × $T_{CLKSC/CLKRC}$.
The calibration measurement time is CUTD × T_{CLKRC}.

2.3. Clock Calibration Unit Calibration Timer Data Register (24-bit) (CUTR)

■ Clock Calibration Unit Calibration Timer Data Register (24-bit) (CUTR)

The Clock Calibration Unit Calibration Timer Data Register stores the result of the calibration. When the calibration is started, the Calibration Timer starts counting up from zero. When the Duration Timer reaches zero, the Calibration Timer stops counting and the register holds the calibration result until the next calibration is triggered by software.

Precaution:

Reading this register during calibration results in random values. The end of calibration is indicated by the INT bit and the STRT bit in the CUCR register.

After INT has changed from 0 to 1 / STRT has changed from 1 to 0, the value of CUTR is valid.

CUTR2L

bit	7	6	5	4	3	2	1	0
	TDR7	TDR6	TDR5	TDR4	TDR3	TDR2	TDR1	TDR0
Attribute	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

CUTR2H

bit	15	14	13	12	11	10	9	8
	TDR15	TDR14	TDR13	TDR12	TDR11	TDR10	TDR9	TDR8
Attribute	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

CUTR1L

bit	7	6	5	4	3	2	1	0
	TDR23	TDR22	TDR21	TDR20	TDR19	TDR18	TDR17	TDR16
Attribute	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

CUTR1H

bit	15	14	13	12	11	10	9	8
	-	-	-	-	-	-	-	-
Attribute	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	0	0	0

Writing into this register by software has no effect.
The Calibration Timer operates with the main oscillator clock CLKMC.

3. Application Notes

This section lists application notes concerning accuracy of the calibration, power dissipation and measurement duration.

■ 32 kHz

The setting of the Duration Timer Data Register can be calculated in the following way.
If the duration of 1 second is desired for the calibration, 8000Hex = 32768Dec should be set in the Duration Timer Data Register and it represents 32,768 pulses of the 32.768kHz oscillation clock.
This setting should result in the stored value of approximately 3D0900Hex in the Calibration Timer Data Register. This value represents 4,000,000 pulses of the 4MHz oscillator.

Ideal Measurement Results (CUTR) with 32.768kHz and 4.0MHz Oscillators

Duration of calibration	CUTD value	CUTR value
2 sec	0x0000	0x7A1200
1.75 sec	0xE000	0x6ACFC0
1.5 sec	0xC000	0x5B8D80
1.25 sec	0xA000	0x4C4B40
1 sec	0x8000	0x3D0900
0.75 sec	0x6000	0x2DC6C0
0.5 sec	0x4000	0x1E8480
0.25 sec	0x2000	0x0F4240

■ 100 kHz

The setting of the Duration Timer Data Register can be calculated in the following way.
If the duration of 0.5 second is desired for the calibration, C350Hex = 50000Dec should be set in the Duration Timer Data Register and it represents 50,000 pulses of the 100kHz oscillation clock.
This setting should result in the stored value of approximately 1E8480Hex in the Calibration Timer Data Register. This value represents 2,000,000 pulses of the 4MHz oscillator.

Ideal Measurement Results (CUTR) with 100kHz and 4.0MHz Oscillators

Duration of calibration	CUTD value	CUTR value
0.5 sec	0xC350	0x1E8480
0.25 sec	0x61A8	0x0F4240
0.125 sec	0x30D4	0x07A120
0.1 sec	0x2710	0x061A80

The purpose to use of the Clock Calibration Unit is to reduce power dissipation of the MCU while maintaining an accurate clock, for example for a daytime display.

■ Accuracy

The accuracy of the calibration is dependent on the clock frequency used by the Calibration Timer and the duration of the calibration. The maximum error of the Calibration Timer is +/-1 digit. If the clock frequency is 4MHz and the duration of the calibration is 1 second, the achieved accuracy is calculated in the following way:

0.25μs (Clock cycle time) / 1 second (duration)=0.25 ppm.

In general:

Accuracy = (Clock cycle time of Calibration Timer) / (Duration of calibration)

■ Power Dissipation

Suppose the current consumption I_{RUN} in run mode is 20 times the consumption I_{RTC} in RTC mode ($I_{RUN} = 20 \times I_{RTC}$).
If the MCU is woken up from RTC mode by software to trigger the calibration measurement every minute and the duration of the calibration is set to 1 second, the increase in the power dissipation can be $20 \times I_{RTC}/60 = 1/3 \times I_{RTC}$.
Therefore the software has to make sure that the increase should not affect the hardware limitations coming from the system requirements. For example, the software has to be designed to trigger the calibration least frequently.
It is generally recommended that the theoretical increase in power dissipation is not more than 5% particularly in the RTC mode of the MCU.

■ Measurement Limits

The limit to the duration of the calibration is roughly 2 seconds if the Duration Timer is operating with a 32kHz clock (0.5 seconds if operating with 100kHz). On the other hand, the Calibration Timer can measure time up to 4 seconds if it is working with a 4MHz clock.

CHAPTER: LCD CONTROLLER/DRIVER

This chapter describes the functions and operations of the LCD Controller/Driver.

Management Code: 96F6LCD-E01.0

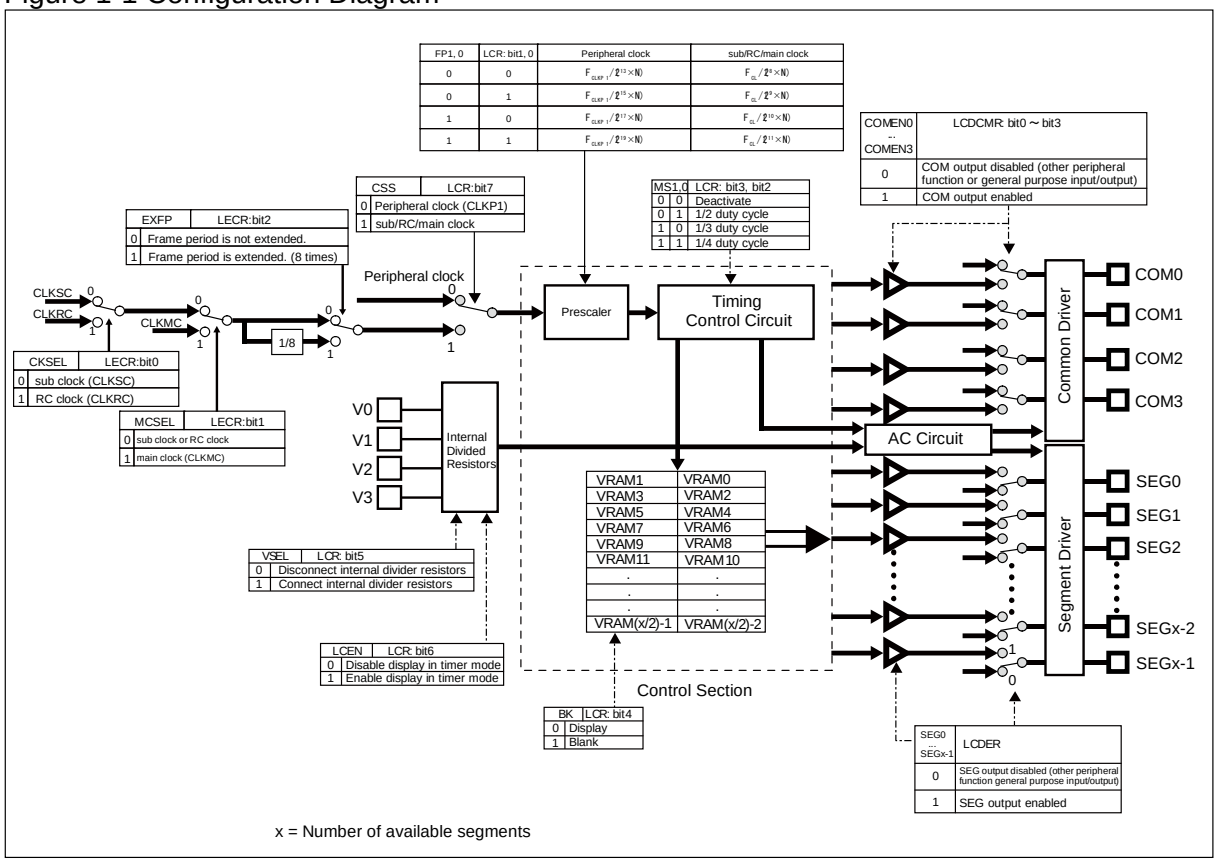
1. Configuration of LCD Controller/Driver

The LCD Controller/Driver consists of the blocks listed below. Functionally, it consists of the controller section, which generates a segment signal and common signal based on display RAM data, and the driver section, which drives the LCD.

- LCD control register (LCR)
- Display RAM
- Prescaler
- Timing controller
- AC circuit
- Common driver
- Segment driver
- Divide resistor

■ Configuration Diagram

Figure 1-1 Configuration Diagram



Note:

Please refer to "Table 3-2 Frame Period" for details of the frame cycle setting (EXFP, FP[1:0]bit).

1.1. LCD Controller/Driver's divide resistors

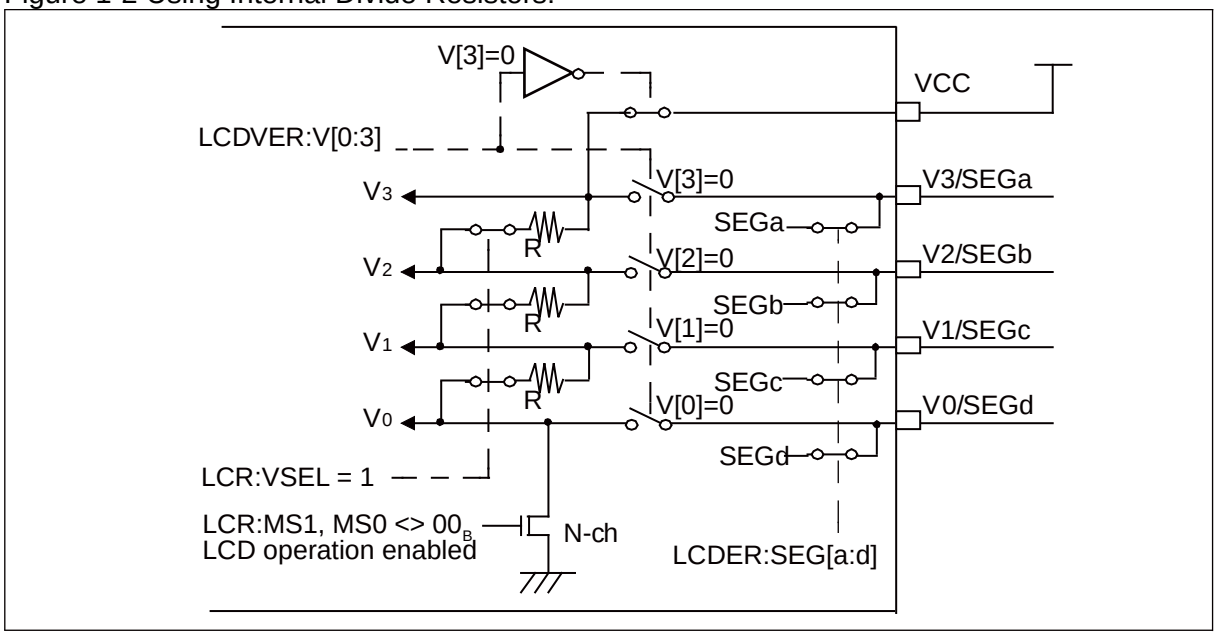
The LCD drive voltage is generated either by external voltage divide resistors or internal voltage divide resistors.
The brightness can be controlled by connecting a variable resister between the V_{CC} and V3 pins.
When internal divide resistors are used, the corresponding pins can be used as segment drivers.

■ Selection of Internal or External Divide Resistors

Use the LCD drive power supply control bit (LCR:VSEL)

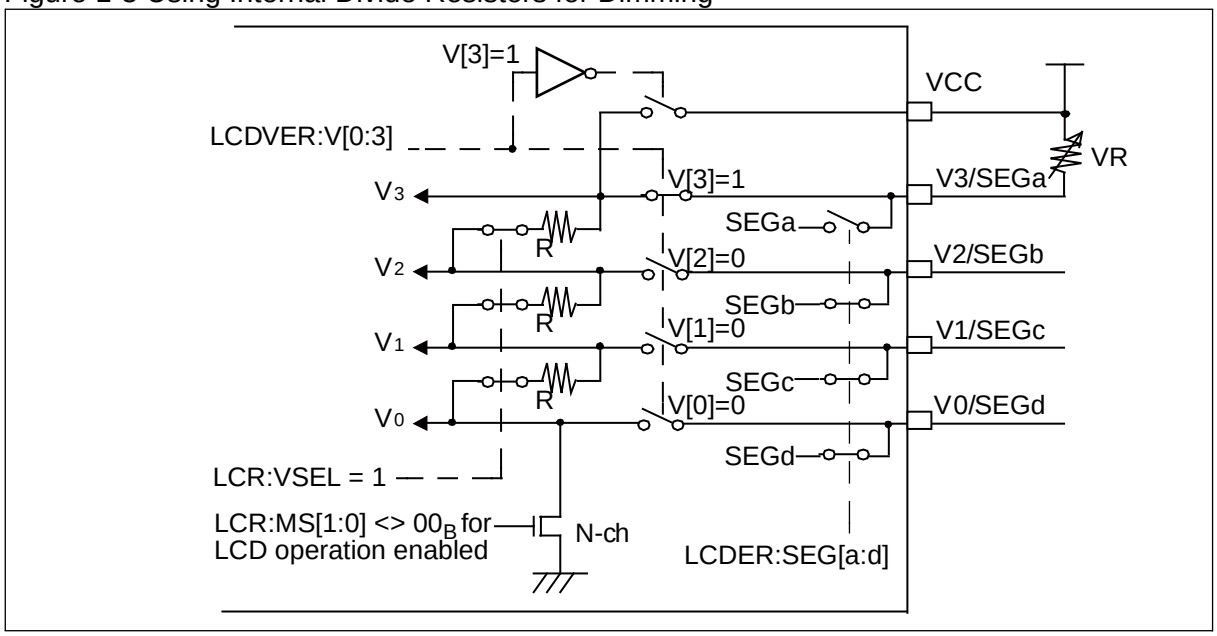
Controlled operation	LCD drive power supply control bit (VSEL).
To use external divide resistors (Internal divide resistors disconnected)	Set to "0".
To use internal divide resistors (Internal divide resistors connected)	Set to "1".

Figure 1-2 Using Internal Divide Resistors:



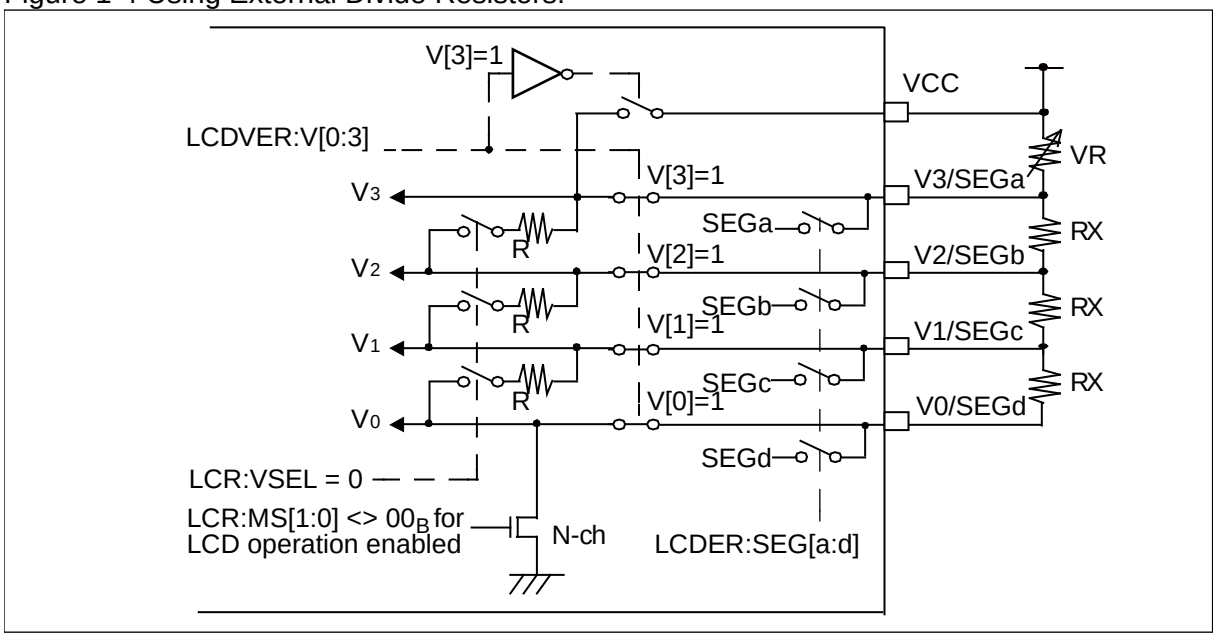
The LCD driving voltage is generated by internal divide resistors. When pins V0..V3 are disabled by setting the corresponding bit in LCDVER:V[3:0] to zero, additional segment pins become available. Since LCDVER:V3 = 0, the internal resistor chain is connected to VCC.

Figure 1-3 Using Internal Divide Resistors for Dimming



Dimming with an external resistor when using the internal voltage divide resistors is possible by setting LCDVER:V3 = 1. The internal resistor chain is connected to pin V3. An external variable resistor connected to V3 is used for dimming. The setting of LCDER:SEGa has no effect.

Figure 1-4 Using External Divide Resistors:



The LCD driving voltage is generated by connecting external divide resistors to the LCD drive power supply pins (V0 to V3).
Each V pin must be enabled by setting LCDVER:V3 to V0 = 1111_B. For each enabled V pin, the corresponding segment enable bit LCDVER:SEG[x] has no effect.
To avoid the effect of internal divide resistors, the resistors must be disconnected by setting LCR:VSEL = 0.

■ Usage of External Divide Resistors to Shut off the Current when LCD is deactivated

The V0 pin is internally connected to Vss (GND) via a transistor. For this reason, the current generated on deactivating LCD controller can be shut off by connecting external divide resistors to the V0 pin on the Vss side. To shut off the current, use the display mode select bit (LCR:MS1, MS0 = "00_B").

2. Operation

This section describes step by step how to use the LCD Controller/Driver.

■ LCD Controller/Driver Operation

- Set values to the display data memory (VRAM) in advance.
- Make necessary settings to each register.
- When the frame period generation clock oscillates, LCD drive waveform is output through common/segment output pins (COM0 to COM3, SEG0 to SEGx-1).
- VRAM contents are automatically read in synchronization with common signals to be output through segment output pins.
  - If the bit is set to "1", the selected waveform is output through the segment output pins. If the bit is set to "0", non-selected waveform is output through the segment output pins.
- If the display mode is set to 1/2 duty cycle, non-selected waveform is output through the COM2 and COM3 pins. For the 1/3 duty cycle, the COM3 pin is used to output non-selected waveform.
- The output waveform is a 2-frame AC waveform in accordance with the duty cycle setting, and drives LCD.
- When MS1, MS0 = "00" is used to deactivate LCD, a "L" level is output through both common and segment pins.
- If LCD operation is enabled in the Timer mode (LCEN="1"), the LCD Controller/Driver continues to display even when MCU enters Timer mode.
  - Note that frame period generation clock signals must be supplied at this time
- LCD display can be blanked by selecting "blank" (BK="1") in blanking selection.
  - Note that non-selected waveform continues to be output.
- When LCD deactivation is selected (MS1, MS0="00_B") with the display mode, LCD stops operating.

■ Output Waveform during LCD Controller/Driver Operation (1/2 duty cycle)

Only COM0 and COM1 outputs are used for LCD display. Neither COM2 nor COM3 output is used.

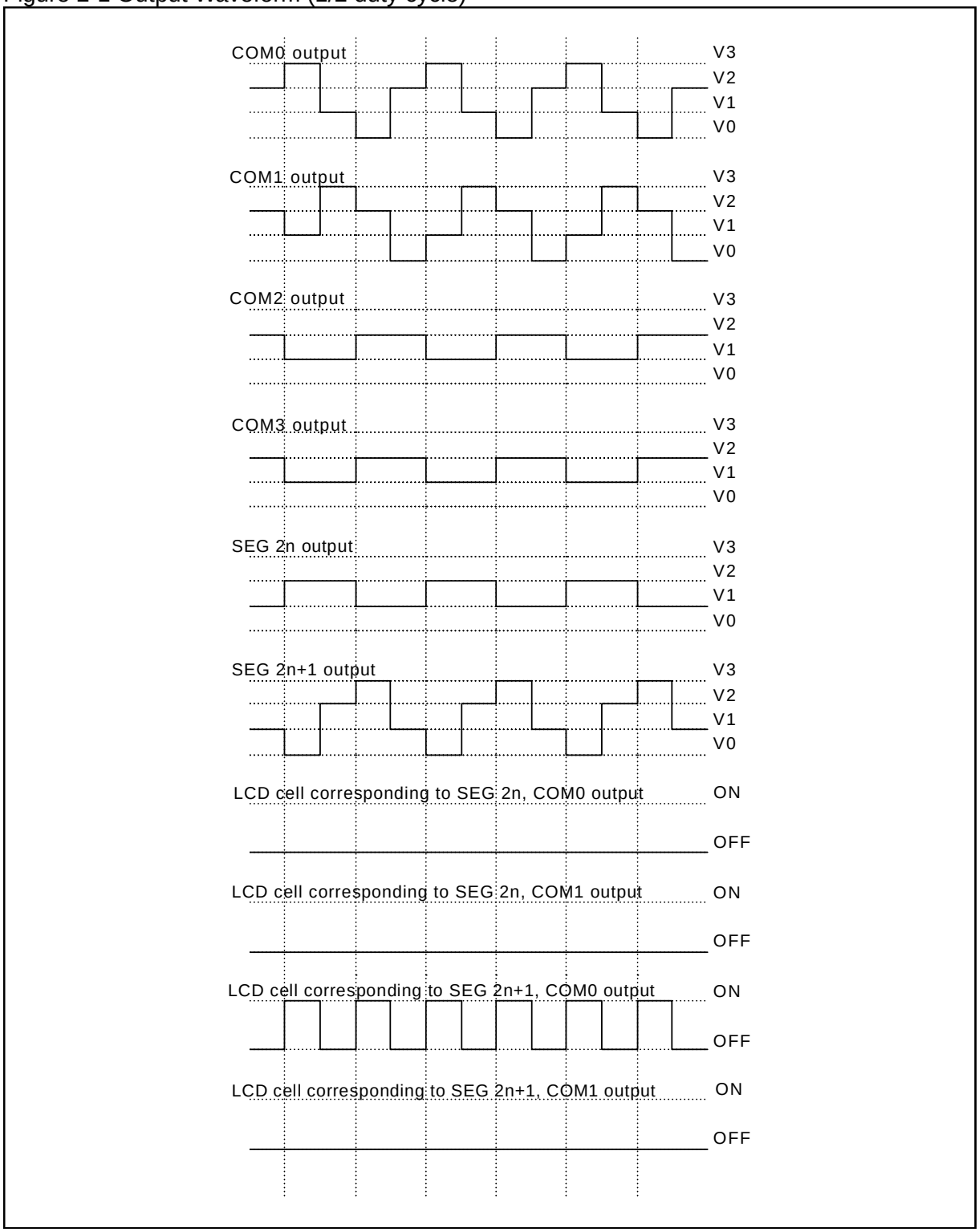
Example of 1/3 Bias Output Waveform

LCD cells with the maximum voltage difference between common and segment outputs are lit.

Table 2-1 Example of Data Memory Contents for Display

Segment	Contents of data memory for display			
	COM3 output	COM2 output	COM1 output	COM0 output
SEG 2n output	-	-	0	0
SEG2n+1 output	-	-	0	1

Figure 2-1 Output Waveform (1/2 duty cycle)



■ Output Waveform during LCD Controller/Driver Operation (1/3 duty cycle)

In the 1/3 duty cycle output mode, COM0, COM1 and COM2 outputs are used for LCD display. COM3 output is not used.

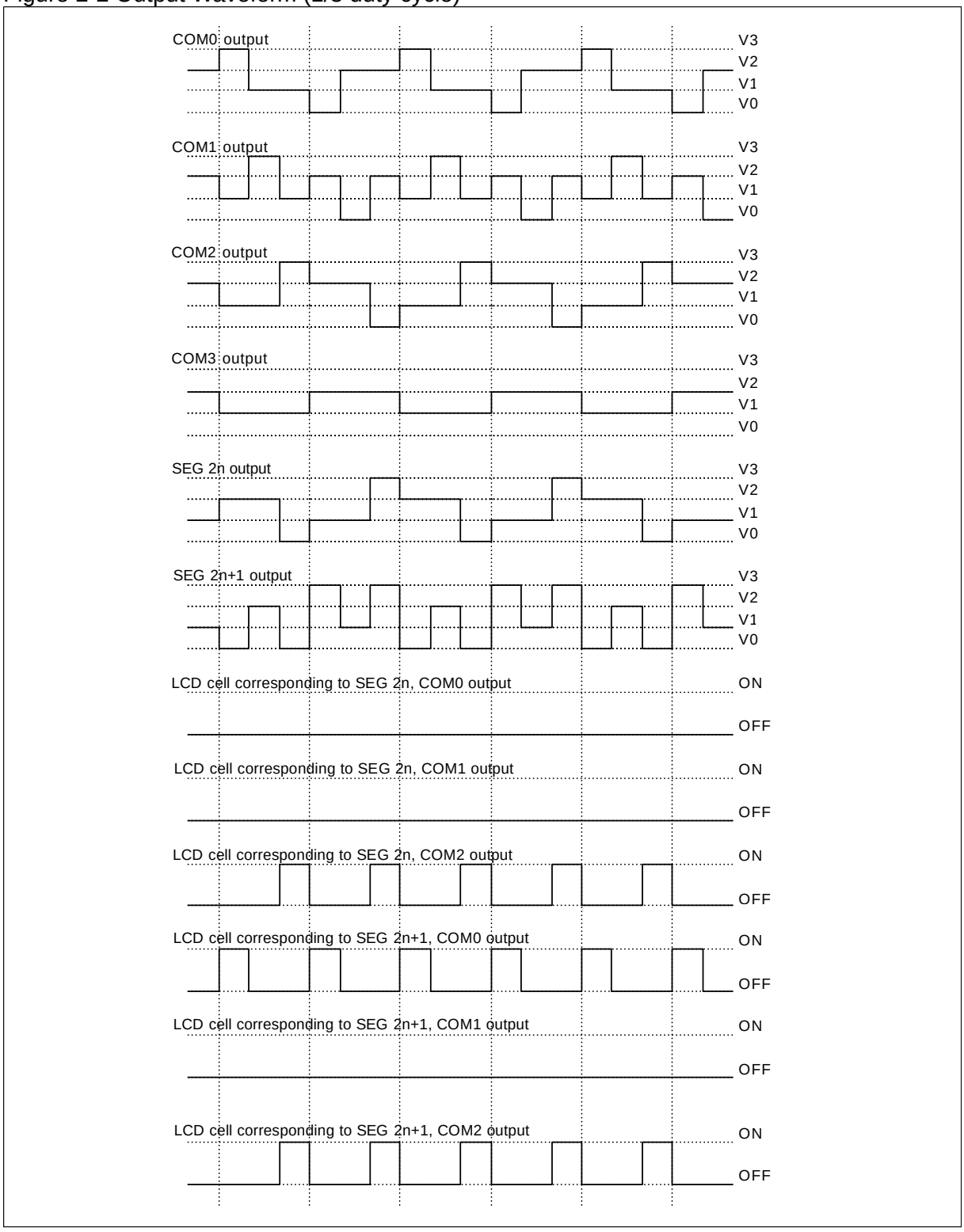
Example of 1/3 Bias Output Waveform

LCD cells with the maximum voltage difference between common and segment outputs are lit.

Table 2-2 Example of Data Memory Contents for Display

Segment	Contents of data memory for display			
	COM3 output	COM2 output	COM1 output	COM0 output
SEG 2n output	-	1	0	0
SEG2n+1 output	-	1	0	1

Figure 2-2 Output Waveform (1/3 duty cycle)



■ Output Waveform during LCD Controller/Driver Operation (1/4 duty cycle)

In the 1/4 duty cycle output mode, COM0, COM1, COM2, and COM3 outputs are all used for LCD display.

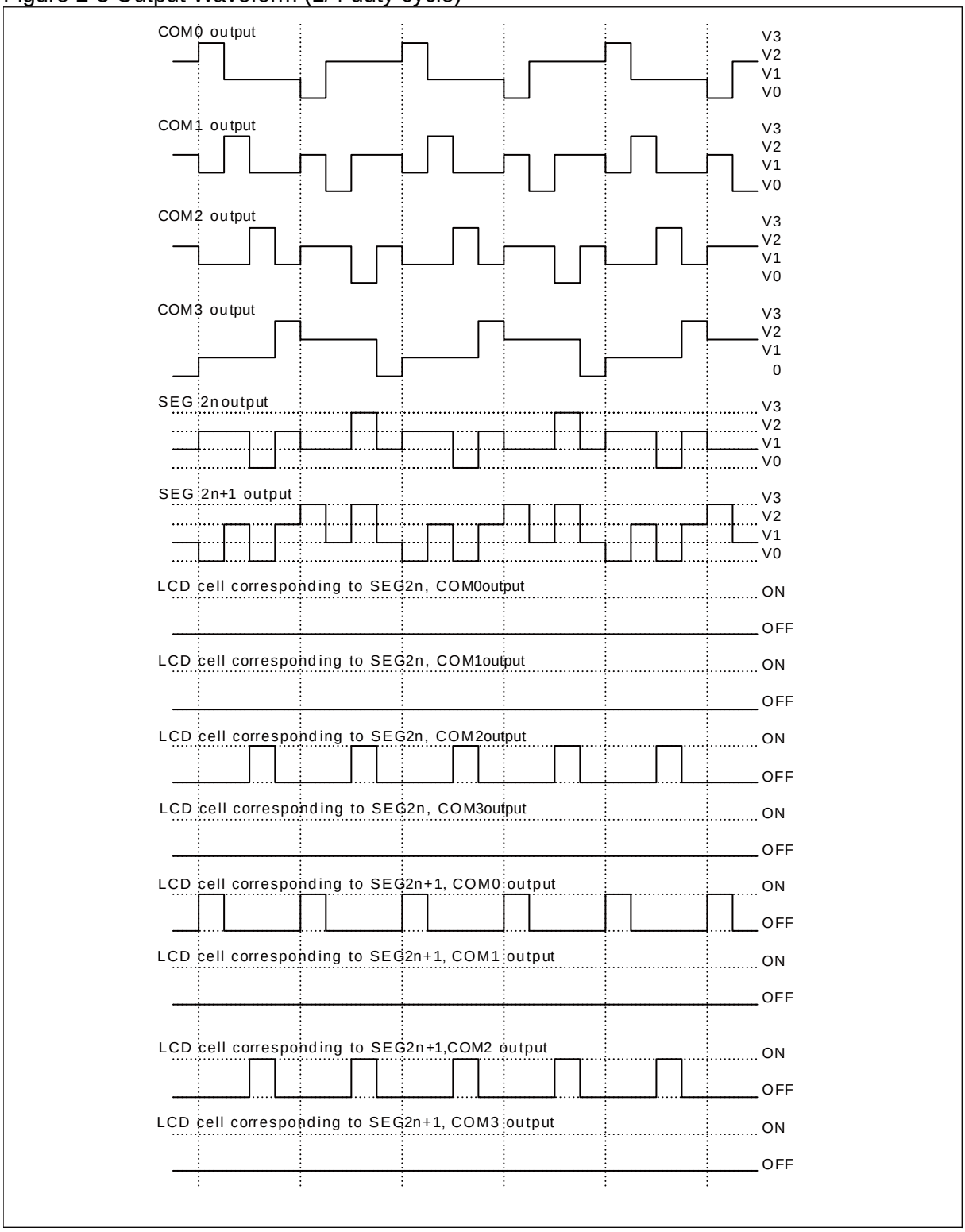
Example of 1/3 Bias Output Waveform

LCD cells with the maximum voltage difference between common and segment outputs are lit.

Table 2-3 Example of Data Memory Contents for Display

Segment	Contents of data memory for display			
	COM3 output	COM2 output	COM1 output	COM0 output
SEG 2n output	0	1	0	0
SEG2n+1 output	0	1	0	1

Figure 2-3 Output Waveform (1/4 duty cycle)



3. LCD Controller Registers

This section describes the registers related to the configuration of LCD Controller/Driver.

■ List of LCD Controller Registers

Abbreviated Register Name	Register Name	Reference
LCR	LCD Control register	See 3.1
LCDCMR	LCD Common Pin switching Register	See 3.2
LECR	LCD Extended Control Register	See 3.3

3.1. LCD Control Register (LCR)

■ LCD Control Register (LCR)

LCR								
bit	7	6	5	4	3	2	1	0
	CSS	LCEN	VSEL	BK	MS1	MS0	FP1	FP0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	1	0	0	0	0

[bit7] CSS: Select the Frame Period Generation Clock

Bit	Description
0	Peripheral clock (CLKP1)
1	clock selected by LECR:CKSEL and LECR:MCSEL

Note:

When the peripheral clock CLKP1 is selected and the MCU is in STOP mode or in Timer mode, the LCD stops operation, because CLKP1 is stopped.

[bit6] LCEN: Enable Operation in Timer Mode

Bit	Description
0	Disable LCD display in the Timer mode
1	Enable LCD display in the Timer mode

Note:

To enable LCD display in Timer mode, the frame period generation clock select bit (CSS) must be also set to "1" and the selected clock (sub clock CLKSC or RC clock CLKSC or main clock CLKMC) must be enabled.

[bit5] VSEL: Control LCD Drive Power Supply

Bit	Description
0	Disconnect internal divider resistors
1	Connect internal divider resistors

Notes:

To connect external divider resistors, the LCD drive power supply control bit (VSEL) must be set to "0"
When external divide resistors are selected, the corresponding segments can not be enabled. The setting of the corresponding LCDER register bits is ignored.
When internal divide resistors are selected the external voltage pins can not be connected. The setting of LCDVER:V3 to V0 is ignored

[bit4] BK: Select Blanking

Bit	Description
0	Enable LCD display
1	Disable (blank) LCD display

Note:

Only the first frame cycle might become shorter than the set value according to timing in which BK is set "0".

[bit3, bit2] MS[1:0] Select a Display Mode

Table 3-1 Function Settings

bit3	bit2	Description
0	0	Deactivate LCD
0	1	1/2 duty cycle output mode (Time division number: N=2, COM0, COM1)
1	0	1/3 duty cycle output mode (Time division number: N=3, COM0-COM2)
1	1	1/4 duty cycle output mode (Time division number: N=4, COM0-COM3)

Note:

If the display mode select bit (MS1, MS0) is set to "00_B", the LCD Controller is disabled, i. e. a "L" level is output to the common/segment pins.]

[bit1, bit0] FP[1:0] Frame Period

Table 3-2 Frame Period

LECR: EXPF	bit1	bit0	When peripheral clock CLKP1 is selected:	When sub clock CLKSC or RC-Clock CLKRC or main clock CLKMC is selected
0	0	0	$F_{CLKP1}/(2^{13} \times N)$	$F_{CL}/(2^{8} \times N)$
0	0	1	$F_{CLKP1}/(2^{15} \times N)$	$F_{CL}/(2^{9} \times N)$
0	1	0	$F_{CLKP1}/(2^{17} \times N)$	$F_{CL}/(2^{10} \times N)$
0	1	1	$F_{CLKP1}/(2^{19} \times N)$	$F_{CL}/(2^{11} \times N)$
1	0	0	$F_{CLKP1}/(2^{13} \times N)$	$F_{CL}/(2^{11} \times N)$
1	0	1	$F_{CLKP1}/(2^{15} \times N)$	$F_{CL}/(2^{12} \times N)$
1	1	0	$F_{CLKP1}/(2^{17} \times N)$	$F_{CL}/(2^{13} \times N)$
1	1	1	$F_{CLKP1}/(2^{19} \times N)$	$F_{CL}/(2^{14} \times N)$

- EXFP: Extended FP bits in LECR
- F_{CLKP1}: Peripheral clock (CLKP1) frequency
- F_{CL}: Subclock (CLKSC) or RC-Clock (CLKRC) or main clock CLKMC frequency
- N: Time division number (Selected with the display mode select bits, MS1 and MS0.)
- Select an appropriate value in accordance with the frame frequency of your LCD panel.

3.2. LCD Common Pin switching Register (LCDCMR)

■ LCD Common Pin switching Register (LCDCMR)

LCDCMR								
bit	7	6	5	4	3	2	1	0
	DTCH	-	-	-	COMEN3	COMEN2	COMEN1	COMEN0
Attribute	R/W	-	-	-	R/W	R/W	R/W	R/W
Initial value	0	X	X	X	0	0	0	0

[bit7] DTCH: Bias Control

Bit	Description
0	1/3 bias
1	Not Approved

Set LCDCMR:DTCH to "0" to use the LCD Controller.

[bit6 to bit4] -: Undefined

Write always "0" to these bits.
Read value is undefined.
Read modify write operations to this register have no effect on this bit.

[bit3 to bit0] COMEN[3:0]: Common Driver Enable

Bit	Description
0	corresponding common driver (COMx) is disabled
1	corresponding common driver (COMx) is enabled

If COMENx is set to "1", the digital function (input/output) of the corresponding pin is disabled.

3.3. LCD Extended Control Register (LECR)

■ LCD Extended Control Register (LECR)

LECR								
bit	7	6	5	4	3	2	1	0
	-	-	-	-	-	EXFP	MCSEL	CKSEL
Attribute	-	-	-	-	-	R/W	R/W	R/W
Initial value	X	X	X	X	X	0	0	0

[bit7 to bit3] -: Undefined

Write always "0" to these bits.
Read value is undefined.
Read modify write operations to this register have no effect on this bit.

[bit2] EXFP: Extended Frame Period Bit

Bit	Description
0	Frame period is not extended.
1	Frame period is extended (8 times). (by selecting Sub clock, RC-clock or main clock)

See Table 3-2 Frame Period.

[bit1] MCSEL: Main Clock Selection Bit

Bit	Description
0	use sub clock or RC clock
1	use main clock

See Table 3-2 Frame Period.

[bit0] CKSEL: Clock Selection Bit

Bit	Description
0	use Sub clock CLKSC
1	use RC-Clock CLKRC

If Sub clock or RC-clock or main clock shall be used as LCD clock, the LCR:CSS bit must be set.

Devices with sub clock: If the Clock setting shall be changed, the clock which is currently selected must be active for at lease 3 cycles if it was enabled at any time before. Otherwise the selected clock does not change to the new clock. However, there is no foregoing limitation in the device without a sub-clock when changing from the sub-clock setting.

4. LCD Enable Registers (LCDER, LCDVER)

This section describes the LCD enable registers.

■ List of LCD Enable Registers

Abbreviated Register Name	Register Name	Reference
LCDER	LCD Segment Enable Registers	See 4.1
LCDVER	Voltage Line Enable Register	See 4.1

4.1. LCD Segment Enable Registers (LCDER) and Voltage Line Enable Register (LCDVER)

■ LCD Segment Enable Registers (LCDER) and Voltage Line Enable Register (LCDVER)

LCDER0

bit	7	6	5	4	3	2	1	0
	SEG7	SEG6	SEG5	SEG4	SEG3	SEG2	SEG1	SEG0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

LCDER1

bit	15	14	13	12	11	10	9	8
	SEG15	SEG14	SEG13	SEG12	SEG11	SEG10	SEG9	SEG8
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

LCDER2

bit	7	6	5	4	3	2	1	0
	SEG23	SEG22	SEG21	SEG20	SEG19	SEG18	SEG17	SEG16
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

LCDER3

bit	15	14	13	12	11	10	9	8
	SEG31	SEG30	SEG29	SEG28	SEG27	SEG26	SEG25	SEG24
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

LCDER4

bit	7	6	5	4	3	2	1	0
	SEG39	SEG38	SEG37	SEG36	SEG35	SEG34	SEG33	SEG32
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

LCDER5

bit	15	14	13	12	11	10	9	8
	SEG47	SEG46	SEG45	SEG44	SEG43	SEG42	SEG41	SEG40
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

LCDER6

bit	7	6	5	4	3	2	1	0
	SEG55	SEG54	SEG53	SEG52	SEG51	SEG50	SEG49	SEG48
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

LCDER7

bit	15	14	13	12	11	10	9	8
	SEG63	SEG62	SEG61	SEG60	SEG59	SEG58	SEG57	SEG56
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

LCDER8

bit	7	6	5	4	3	2	1	0
	SEG71	SEG70	SEG69	SEG68	SEG67	SEG66	SEG65	SEG64
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

LCDER9

bit	15	14	13	12	11	10	9	8
	-	-	-	-	-	-	-	-
Attribute	-	-	-	-	-	-	-	-
Initial value	X	X	X	X	X	X	X	X

LCDVER

bit	7	6	5	4	3	2	1	0
	-	-	-	-	V3	V2	V1	V0
Attribute	-	-	-	-	R/W	R/W	R/W	R/W
Initial value	X	X	X	X	0	0	0	0

0 - LCD output (SEGxx) or input (Vx) is disabled, other resource function can be used or pin is General purpose I/O
1 - LCD output (SEGxx) or input (Vx) is enabled, other pin functions cannot be used (digital function is disabled)
"-" - Reserved bit: write always "0", read value is undefined, RMW access is not affected

Note:

When an internal divide resistor x is activated by LCDVER:Vx = 1, the setting in the LCD enable register LCDERy:SEGz of the segment sharing the pin of Vx is ignored and the segment is disabled. This is implemented to avoid a shortcut between a segment output and an internal voltage divider.
Write always "0" to the SEG which not used.

5. LCD Controller/Driver Display RAM

The display RAM is a 36 x 8 bit display memory area used to generate a segment output signal.

■ Memory Area (VRAM) for Setting Display Data

- VRAM0 (SEG1, SEG0): (Access: Byte)

  ·
  ·
  ·

- VRAM(x/2 - 1) (SEGx-1, SEGx-2): (Access: Byte)

Note:

x: Maximum number of segments

■ Bit Configuration of Display RAM

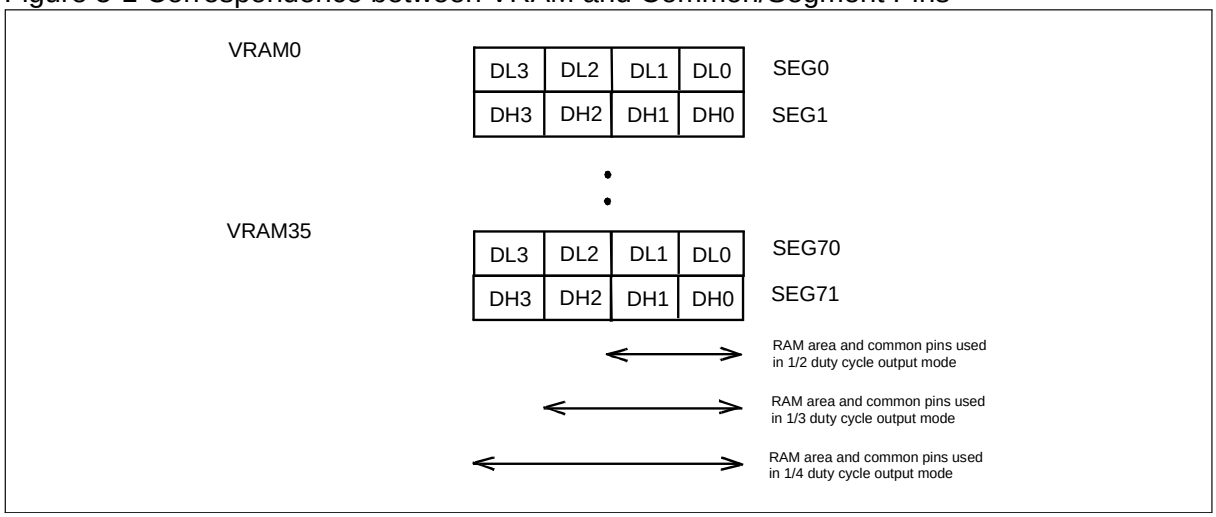
VRAM								
bit	7	6	5	4	3	2	1	0
	DH3	DH2	DH1	DH0	DL3	DL2	DL1	DL0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	X	X	X	X	X	X	X	X

Note:

Regardless of the operation of LCD Controller/Driver, RAM can be read and written any time.

■ Correspondence between VRAM and Common/Segment Pins

Figure 5-1 Correspondence between VRAM and Common/Segment Pins



6. Cautions

Cautions covering the usage of LCD controller/driver.

- Switching the frame period generation clocks:
  - Frame period generation clocks (LCR:CSS , LECR:MCSEL,CKSEL) can be switched even during LCD display.
  - However, switching may cause some screen flicker. To avoid such flicker, be sure to set the blanking select bit (LCR:BK) to "1" (blank display) before switching.
- Depending on your LCD, different external divider resistors are used. Use appropriate resistor values.
- When the display mode is set to 1/2 duty cycle, non-selected waveform is output through the COM2 and COM3 pins. For 1/3 duty cycle, the COM3 pin is used to output non-selected waveform.
- Inappropriate selection or setting of frame period generation clock (CSS), LCD drive power supply control (VSEL), duty cycle (MS1, MS0) and frame period (FP1, FP0) results in inappropriate LCD display
- By default, LCD display is disabled in MCU Timer modes. To enable LCD display in MCU Timer modes, use LCR:LCEN and make sure the required clock is operating.
- LCD display is disabled in MCU STOP mode.
- Take care that the LCD display is not enabled if the required clock is not operating. Otherwise the LCD panel might be damaged by DC voltage. Refer to the following tables.

Table 6-1 LCD Operation Except in MCU Timer Modes or MCU STOP Mode

CSS	CKSEL	MCSEL	Sub clock CLKSC running	Main clock CLKMC running	RC clock CLKRC running	Operation for MS1, MS0 ≠ "00_B"
0	X	X	X	X	X	LCD controller is running with Peripheral Clock CLKP1.
1	0	0	No	X	X	**Do not use this setting! DC voltage is applied to the LCD panel.**
1	0	0	Yes	X	X	LCD controller is running with Sub Clock CLKSC.
1	1	0	X	No	X	**Do not use this setting! DC voltage is applied to the LCD panel.**
1	1	0	X	Yes	X	LCD controller is running with RC Clock CLKRC.
1	X	1	X	X	No	**Do not use this setting! DC voltage is applied to the LCD panel.**
1	X	1	X	X	Yes	LCD controller is running with Main Clock CLKMC.

Table 6-2 LCD Operation in MCU Timer Modes

CSS	LCEN	CKSEL	MCSEL	Sub clock CLKSC running	RC clock CLKRC running	Main clock CLKMC running	Operation for MS1, MS0 ≠ "00_B"
0	0	X	X	X	X	X	LCD display is deactivated.
0	1	X	X	X	X	X	**Do not use this setting! DC voltage is applied to the LCD panel.**
1	0	X	X	X	X	X	LCD display is deactivated.
1	1	0	X	No	X	X	**Do not use this setting! DC voltage is applied to the LCD panel.**
1	1	0	X	Yes	X	X	LCD controller is running with Sub Clock CLKSC.
1	1	1	X	X	No	X	**Do not use this setting! DC voltage is applied to the LCD panel.**
1	1	1	X	X	Yes	X	LCD controller is running with RC Clock CLKRC.
1	1	X	1	X	X	No	**Do not use this setting! DC voltage is applied to the LCD panel.**
1	1	X	1	X	X	Yes	LCD controller is running with Main Clock CLKMC.

For MS1, MS0 = "00_B" and in MCU STOP mode, the LCD display is deactivated.

CHAPTER: STEPPING MOTOR CONTROLLER

This chapter explains the functions and operations of the Stepping Motor Controller.

Management Code: FIP005S-E01.4

1. Overview

This section outlines the Stepping Motor Controller.

The Stepping Motor Controller consists of two PWM pulse generators and four motor drivers.

The four motor drivers have a high-output driving capability, and two motor coils can be connected directly to four stepping motor controller pins. The motor rotation is designed to be controlled by a combination of the PWM pulse generators and selector logic circuits. The synchronization mechanism enables synchronous operation of two PWM pulse generators to operate surely.

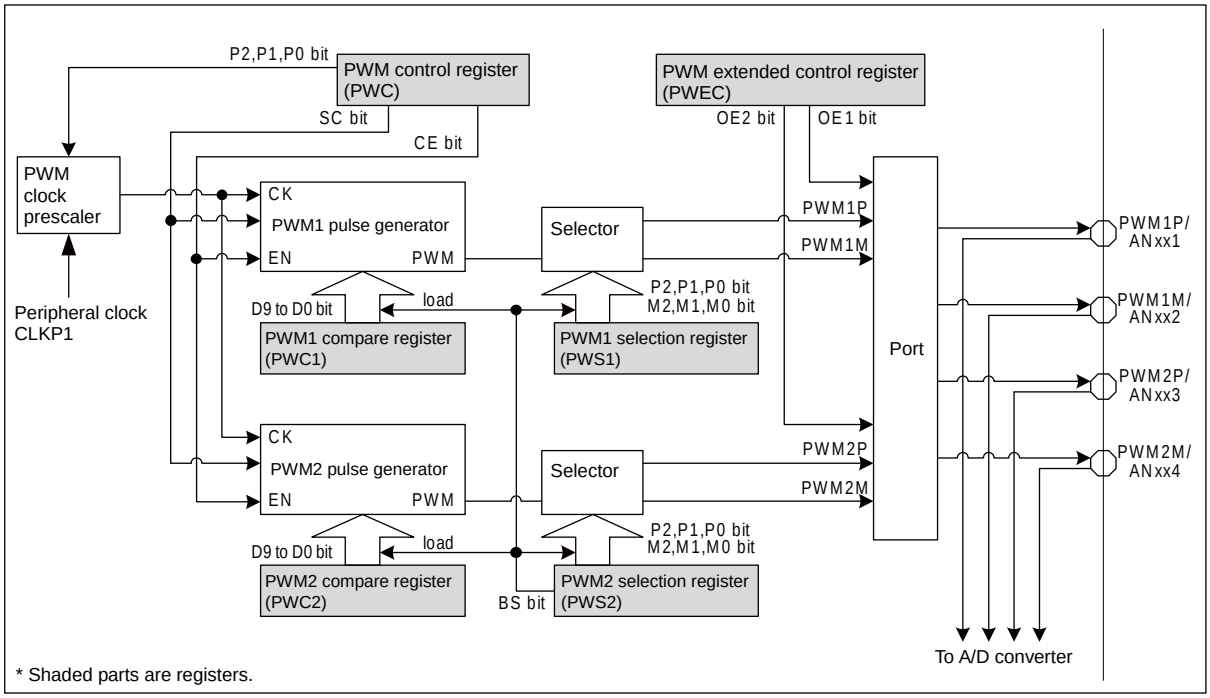
- PWM pulse generator
  - PWM operating clock (the division ratio from a peripheral clock) can be selected by a program.
  - 8-bit operation mode or 10-bit operation mode can be selected by a program.
    - For the PWM pulse width in 8-bit operation mode, the duty can be specified within the range from 0% to 99.6%.
    - For the PWM pulse width in 10-bit operation mode, the duty can be specified within the range from 0% to 99.9%.

- Selecting the motor drive signal (selector logic)
  - Signal levels "H" and "L", PWM pulse and "Hi-Z" to be output to the Stepping Motor Controller pins are selected by a program.

2. Configuration

Figure 2-1 shows a block diagram of the Stepping Motor Controller.

Figure 2-1 Block Diagram of Stepping Motor Controller



- **PWM Clock Prescaler**
  It divides the peripheral clock CLKP1 to generate the clock to be used for the PWM pulse generators.

- **PWM1 Pulse Generator / PWM2 Pulse Generator**
  These generate the PWM pulse. 8-bit operation mode or 10-bit operation mode can be selected for generating the PWM pulse.

- **Selector**
  It selects the level of the signal to be output to the Stepping Motor Controller pins.

- **Port**
  It controls the I/O port for the Stepping Motor Controller pins.

- **PWM Control Register (PWC)**
  It starts/stops the Stepping Motor Controller, sets the PWM prescaler, and selects PWM pulse generation operation mode.

- **PWM Extended Control Register (PWEC)**
  It controls the output of the Stepping Motor Controller in the I/O port control.

- **PWM1 Compare Register (PWC1) / PWM2 Compare Register (PWC2)**
  These set the PWM pulse width.

- **PWM1 Selection Register (PWS1) / PWM2 Selection Register (PWS2)**
  These select signal levels "H" and "L", PWM pulse and "Hi-Z" to be output to the Stepping Motor Controller pins. They set the rewrite instruction for the synchronization system for the pulse generators PWM1 and PWM2.

3. Explanation of Operations

This section explains the operations of the Stepping Motor Controller.

■ Operation of PWM Pulse Generator

The PWM pulse generator can select the operating clock to be used by the PWM pulse generator. For the PWM operating clock, the division ratio of the peripheral clock can be selected by the PWM operating clock selection bits (PWC:P2 to P0).

The PWM pulse generator can select 8-bit operation or 10-bit operation for the counter that controls the PWM cycle. When the 8/10 bits switching bit (PWC:SC) is set to "0", 8-bit operation (PWM cycle: 256 counts) is selected. When it is set to "1", 10-bit operation (PWM cycle: 1024 counts) is selected.

Table 3-1 Relationship between PWM Operating Clock and PWM Cycle

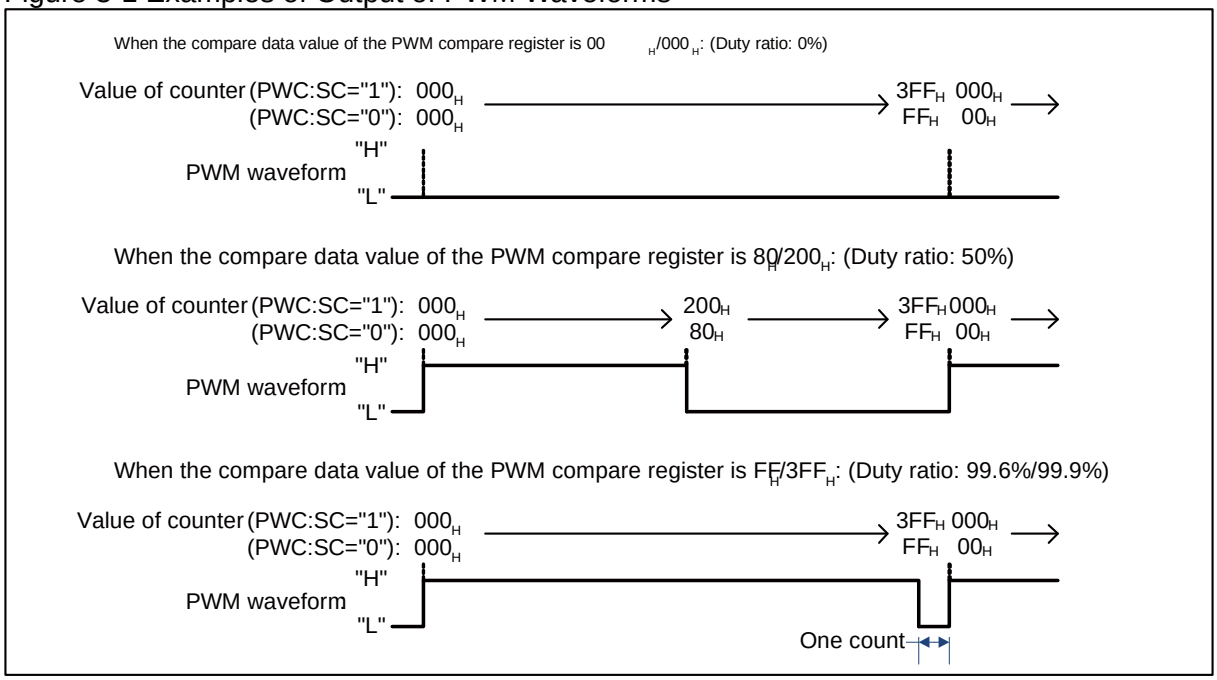
PWC P2 to P0	PWM Operating Clock	PWM Cycle (CLKP1 = 16MHz)		PWM Cycle (CLKP1 = 32MHz)	
		PWC:SC="0"	PWC:SC="1"	PWC:SC="0"	PWC:SC="1"
"000_B"	CLKP1	16.0 µs	64.0 µs	8.0 µs	32.0 µs
"001_B"	1/4 × CLKP1	64.0 µs	256.0 µs	32.0 µs	128.0 µs
"010_B"	1/5 × CLKP1	80.0 µs	320.0 µs	40.0 µs	160.0 µs
"011_B"	1/6 × CLKP1	96.0 µs	384.0 µs	48.0 µs	192.0 µs
"100_B"	1/8 × CLKP1	128.0 µs	512.0 µs	64.0 µs	256.0 µs
"101_B"	1/10 × CLKP1	160.0 µs	640.0 µs	80.0 µs	320.0 µs
"110_B"	1/12 × CLKP1	192.0 µs	768.0 µs	96.0 µs	384.0 µs
"111_B"	1/16 × CLKP1	256.0 µs	1024.0 µs	128.0 µs	512.0 µs

CLKP1: Peripheral clock 1

When the count enable bit (PWC:CE) is set to "1", the PWM pulse generator starts its counter, and it starts counting up from "00_H"("000_H") at the rising edge of the selected PWM operating clock. The PWM output pulse remains at the "H" level until the counter value matches the value of the Compare Register (PWC1:D9 to D0, PWC2:D9 to D0). After matching with the Compare Data (PWC1:D9 to D0, PWC2:D9 to D0), it remains at the "L" level until the counter value overflows: "FF_H"→"00_H"("$3FF_H$"→"000_H").

Figure 3-1 shows examples of the output of the PWM waveforms generated by the PWM pulse generator.

Figure 3-1 Examples of Output of PWM Waveforms



■ Selection of Motor Drive Signals

The motor drive signals that are output to each pin related to the Stepping Motor Controller can be selected from four different types of signals according to the settings of the PWM1 Selection Register (PWS1) and PWM2 Selection Register (PWS2).

Table 3-2 Selection of Motor Drive Signals and Setting of PWM1 Selection Register

PWS1:P2 to P0	PWM1P Output	PWS1:M2 to M0	PWM1M Output
"000_B"	L	"000_B"	L
"001_B"	H	"001_B"	H
"$01X_B$"	PWM pulse	"$01X_B$"	PWM pulse
"$1XX_B$"	HI-Z	"$1XX_B$"	HI-Z

X: don't care

Table 3-3 Selection of Motor Drive Signals and Setting of PWM2 Selection Register

PWS2:P2 to P0	PWM2P Output	PWS2:M2 to M0	PWM2M Output
"000_B"	L	"000_B"	L
"001_B"	H	"001_B"	H
"$01X_B$"	PWM pulse	"$01X_B$"	PWM pulse
"$1XX_B$"	HI-Z	"$1XX_B$"	HI-Z

X: don't care

■ Synchronization System of Stepping Motor Controller

The PWM pulse generator and motor drive signal selection have a synchronization system to synchronize PWM1 and PWM2.

The synchronization system is controlled by the Rewrite bit (PWS2:BS). Until the Rewrite bit (PWS2:BS) is set to "1", changes made to the two PWM Compare registers (PWC1,PWC2) and two PWM Selection registers (PWS1,PWS2) are not reflected in the output signal.

When "1" is set to the Rewrite bit (PWS2:BS), the PWM pulse generator and motor drive signal selection load the values of the registers at the end of the current PWM cycle. The Rewrite bit (PWS2:BS) is cleared to "0" automatically at the beginning of the next PWM cycle.

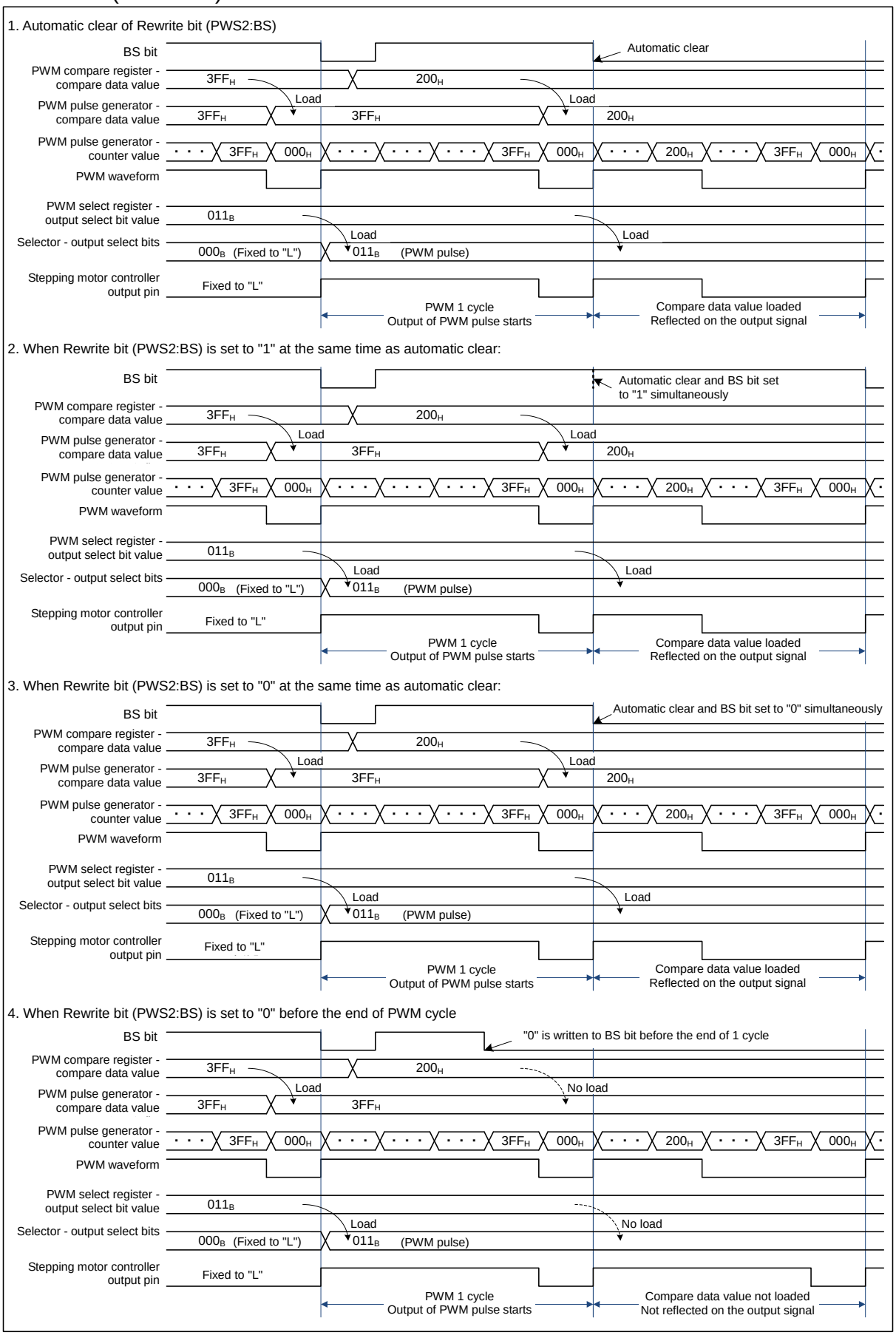
Also, if there is a conflict between the automatic clear of the Rewrite bit (PWS2:BS) to "0" and access to the Rewrite bit (PWS2:BS) by a program, the following operation applies.

- When the Rewrite bit (PWS2:BS) is set to "1" and it is set to "1" by a program at the beginning of the PWM cycle, at the same time as an automatic clear to "0", the Rewrite bit (PWS2:BS) is set to "1" (i.e. no change is made) and automatic clearing is cancelled.
- When the Rewrite bit (PWS2:BS) is set to "1" and it is set to "0" by a program at the beginning of the PWM cycle, at the same time as an automatic clear to "0", the Rewrite bit (PWS2:BS) is cleared to "0", the PWM pulse generator and the selector do not load the values of the registers at the end of the current PWM cycle.

Figure 3-2 shows the load operation of the PWM Compare Registers and PWM Selection Registers by the Rewrite bit (PWS2:BS).

1. Automatic clear of the Rewrite bit (PWS2:BS):
   Load operation is executed and reflected on the output signal.
2. When the Rewrite bit (PWS2:BS) is set to "1" at the same time as the automatic clear:
   Load operation is executed and reflected on the output signal.
3. When the Rewrite bit (PWS2:BS) is set to "0" at the same time as the automatic clear:
   Load operation is executed and reflected on the output signal.
4. When the Rewrite bit (PWS2:BS) is set to "0" before the end of the PWM cycle:
   Load operation is not executed or reflected on the output signal.

Figure 3-2 Load Operation of PWM Compare Registers and PWM Selection Registers by Rewrite Bit (PWS2:BS)

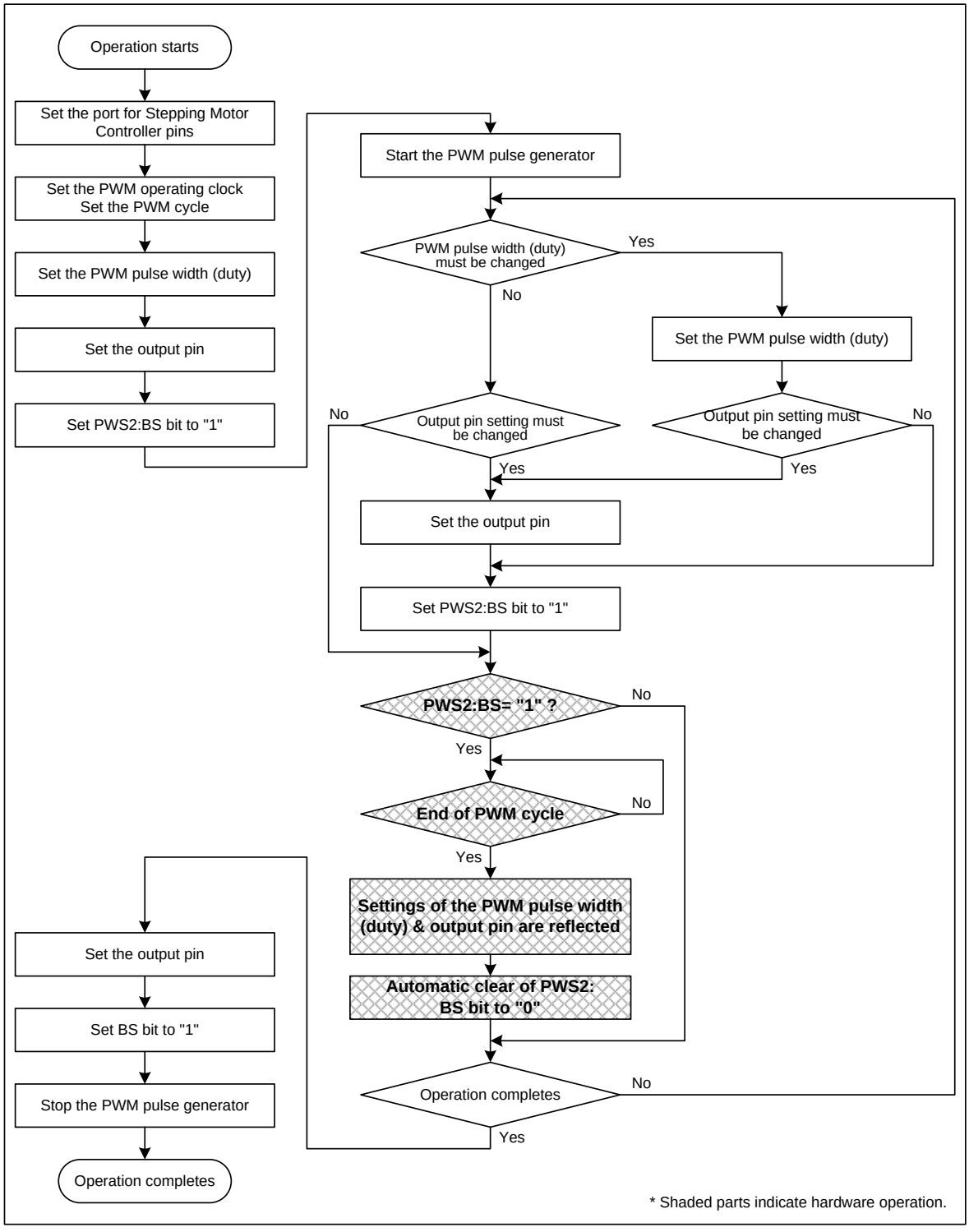


4. Example of Setting Procedure

This section shows an example of the setting procedure for the Stepping Motor Controller.

■ Example of Setting Procedure for Stepping Motor Controller

Figure 4-1 Example of Setting Procedure for Stepping Motor Controller



● Setting the Port for the Stepping Motor Controller Pin

Set the Stepping Motor Controller pin to the PWM output pin.

As for the setting of the PWM output pin, set the output enable bits (PWEC:OE2, PWEC:OE1) to "1".

● Setting the PWM Operating Clock

The PWM operating clock is set by the PWM operating clock select bits (PWC:P2 to P0). For the setting of the PWM operating clock select bits (PWC:P2 to P0), see "Section 5.1 PWM Control Register (PWC)".

● Setting the PWM Cycle

The PWM cycle is determined by the PWM operating clock and the operation mode (8-bit/10-bit) of the PWM pulse generator. The operation mode of the PWM pulse generator is switched by the 8/10 bits switching bit (PWC:SC). For the setting of the 8/10 bits switching bit (PWC:SC), see "Section 5.1 PWM Control Register (PWC)".

The PWM cycle length can be calculated by the following formulas.

- In 8-bit operation (PWC:SC="0"): (1 / PWM operating clock frequency) × 256
- In 10-bit operation (PWC:SC="1"): (1 / PWM operating clock frequency) × 1024

● Setting the PWM Pulse Width (Duty)

The PWM pulse width (duty value) is set by the Compare Data (PWC1:D9 to D0, PWC2:D9 to D0). For the setting of the Compare Data (PWC1:D9 to D0, PWC2:D9 to D0), see "Section 5.3 PWM1 and PWM2 Compare Registers (PWC1, PWC2)".

The Compare Data (PWC1:D9 to D0, PWC2:D9 to D0) and the duty value have a relationship based on the following formulas.

- In 8-bit operation (PWC:SC="0"):
  Compare Data (PWC1:D7 to D0, PWC2:D7 to D0) = Duty × (256 / 100)

- In 10-bit operation (PWC:SC="1"):
  Compare Data (PWC1:D9 to D0, PWC2:D9 to D0) = Duty × (1024 / 100)

Below are the ranges in which Compare Data (PWC1:D9 to D0, PWC2:D9 to D0) can be set.

- In 8-bit operation (PWC:SC="0"): 0% to 99.6% ("00_H" to "FF_H")
- In 10-bit operation (PWC:SC="1"): 0% to 99.9% ("000_H" to "$3FF_H$")

● Setting the Output Pin

As for the setting of the output pin, select "L", "H", PWM pulse or "Hi-Z" for the motor drive signals to be output to each pin of the Stepping Motor Controller.

The output pin is set by the output select bits (PWS1:P2 to P0, PWS1:M2 to M0, PWS2:P2 to P0, PWS2:M2 to M0). For the setting method for the output select bits, see "Section 5.4 PWM1 and PWM2 Selection Registers (PWS1, PWS2)".

● PWS2:BS="1" Setting: PWM Pulse Width (Duty) Setting and Output Pin Setting Reflected

To reflect the setting of the PWM pulse width (duty) and the setting of the output pin, set the Rewrite bit (PWS2:BS) to "1". Setting the Rewrite bit (PWS2:BS) to "1" updates the setting of the PWM pulse width (duty) and the setting of the output pin at the end of the current PWM cycle. At the same time, the Rewrite bit (PWS2:BS) is automatically cleared to "0". For the timing of reflecting the setting of the PWM pulse width (duty) and the setting of the output pin, see Figure 3-2.

Note:

If the PWM pulse generator is stopped (PWC:CE="0"), the setting of the PWM pulse width (duty) and the setting of the output pin are not reflected.

● Starting/Stopping the PWM Pulse Generator

The PWM pulse generator is started/stopped by the count enable bit (PWC:CE). For the setting method of the count enable bit (PWC:CE), see "Section 5.1 PWM Control Register (PWC)".

Notes:

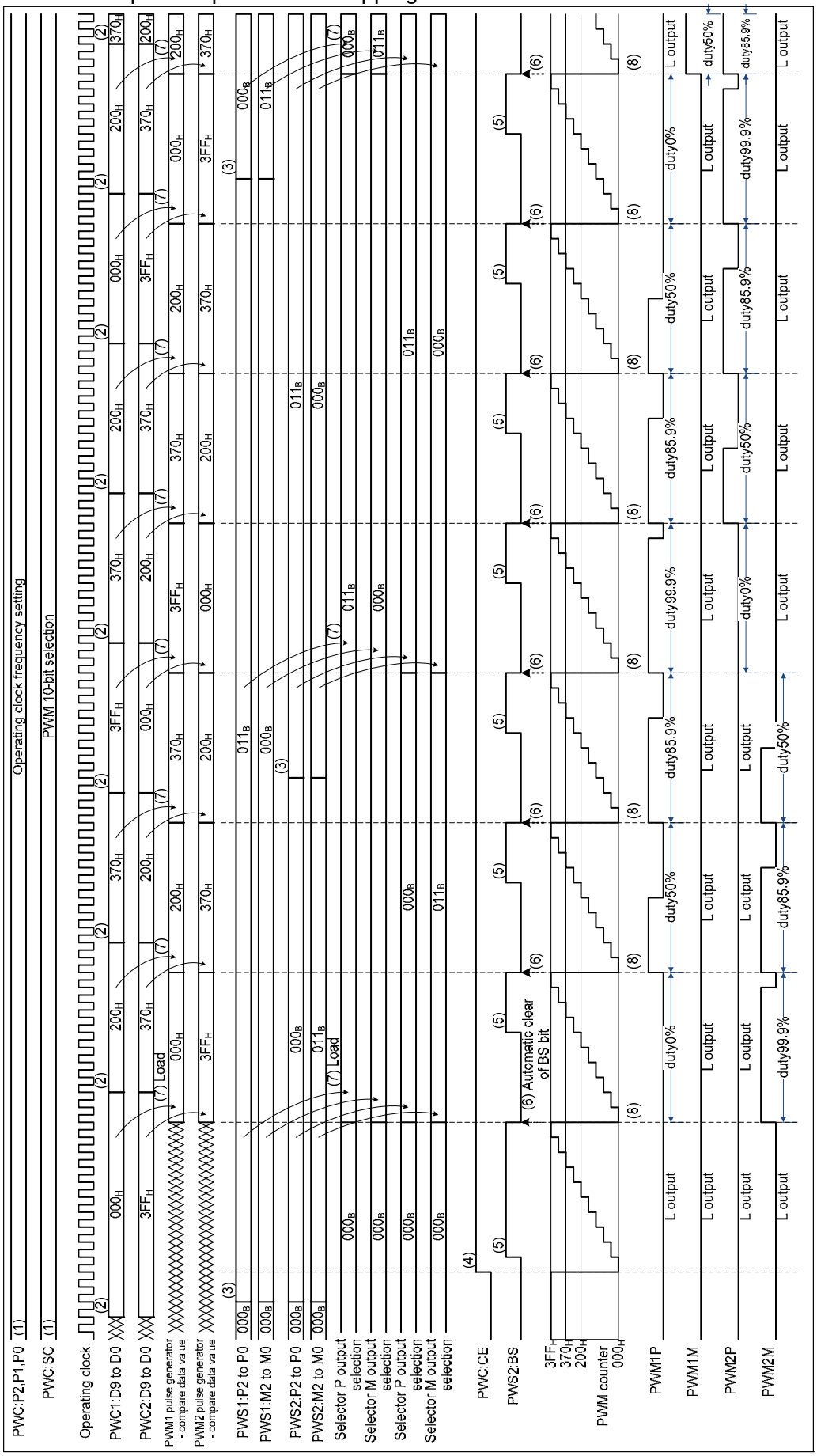
- Set the count enable bit (PWC:CE) to "1", after the setting of the PWM operating clock selection bits (PWC:P2 to P0) and the 8/10 bits switching bit (PWC:SC) is completed.
- If the PWM pulse generator is stopped (PWC:CE="0") when the PWM pulse is selected by the output select bits (PWS1:P2 to P0, PWS1:M2 to M0, PWS2:P2 to P0, PWS2:M2 to M0), the pin for which the PWM pulse is selected is fixed to the "L" level.

■ Example of Operation of Stepping Motor Controller

Figure 4-2 shows an example of the operation of the Stepping Motor Controller.

1. Set the PWM operating clock and the PWM cycle.
2. Set the PWM pulse width (duty).
3. Set the output pin.
4. Start the PWM pulse generator.
5. Set the Rewrite bit (PWS2:BS) to "1" and instruct the setting to be updated.
6. The Rewrite bit (PWS2:BS) is automatically cleared to "0".
7. Update to the new setting values.
   - Load the Compare Data values of PWM1 and PWM2 Compare Registers (PWC1, PWC2) to the PWM1 and PWM2 pulse generators.
   - Load the values of the output select bits (P2 to P0, M2 to M0) of the PWM1 and PWM2 Selection Registers (PWS1, PWS2) to the PWM1 and PWM2 selectors.
8. Output the PWM pulse.

Figure 4-2 Example of Operation of Stepping Motor Controller



5. Registers

This section explains the configuration and functions of the registers used for the Stepping Motor Controller.

■ List of Stepping Motor Controller Registers

Register Abbreviation	Register Name	Reference
PWC	PWM Control Register	See 5.1
PWEC	PWM Extended Control Register	See 5.2
PWC1	PWM1 Compare Register	See 5.3
PWC2	PWM2 Compare Register	See 5.3
PWS1	PWM1 Selection Register	See 5.4
PWS2	PWM2 Selection Register	See 5.4

5.1. PWM Control Register (PWC)

The PWM Control Register starts/stops the Stepping Motor Controller, sets the PWM prescaler, and selects the operation mode for generation of the PWM pulse.

■ PWM Control Register (PWC)

● Register Configuration

bit	7	6	5	4	3	2	1	0
	RESV	P2	P1	P0	CE	SC	RESV	RESV
Attribute	-	R/W	R/W	R/W	R/W	R/W	-	-
Initial value	0	0	0	0	0	0	0	0

● Register Functions

[bit7] RESV: Reserved Bit

Always write "0" to this bit.

"0" is always read from this bit.

[bit6 to bit4] P2 to P0: PWM Operating Clock Select Bits

P2 to P0 bits specify the clock input signal to the PWM pulse generators.

bit6	bit5	bit4	Description
0	0	0	Peripheral clock (CLKP1)
0	0	1	1/4 × Peripheral clock (CLKP1)
0	1	0	1/5 × Peripheral clock (CLKP1)
0	1	1	1/6 × Peripheral clock (CLKP1)
1	0	0	1/8 × Peripheral clock (CLKP1)
1	0	1	1/10 × Peripheral clock (CLKP1)
1	1	0	1/12 × Peripheral clock (CLKP1)
1	1	1	1/16 × Peripheral clock (CLKP1)

Change the P2 to P0 bits, when the count enable bit is stopped (CE="0").

When changing the setting of P2 to P0 bits, up to 18 peripheral clock cycles (CLKP1) are required to reflect settings.

[bit3] CE: Count Enable Bit

The CE bit enables operation of the PWM pulse generator.

Bit	Description
0	Initializes and stops the operation of the PWM pulse generator.
1	Starts the operation of the PWM pulse generator.

When "1" is set to the CE bit, the PWM pulse generator starts its operating. The PWM2 pulse generator starts, one peripheral clock cycle (CLKP1) after the PWM1 pulse generator starts, in order to reduce the switching noise generated by the output driver.

Notes:

- Set the CE bit to "1", after the setting of the PWM operating clock select bits (P2 to P0) and the 8/10 bits switching bit (SC) is completed.
- If the PWM pulse generator is stopped (CE="0") when the PWM pulse is selected by the output select bits (PWS1/PWS2:P2 to P0, PWS1/PWS2: M2 to M0), the pin selected by the PWM pulse is fixed to the "L" level.

[bit2] SC: 8/10-bits Switching Bit

The SC bit specifies the operation mode of the PWM pulse generator.

Bit	Description
0	8-bit operation mode
1	10-bit operation mode

[bit1,bit0] RESV: Reserved Bits

Always write "0" to this bit.

"0" is always read from this bit.

5.2. PWM Extended Control Register (PWEC)

The PWM Extended Control Register enables the PWM output.

■ PWM Extended Control Register (PWEC)

● Register Configuration

bit	15	14	13	12	11	10	9	8
	RESV	RESV	RESV	RESV	RESV	RESV	OE2	OE1
Attribute	-	-	-	-	-	-	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

● Register Functions

[bit15 to bit10] RESV: Reserved Bits

Always write "0" to this bit.

"0" is always read from this bit.

[bit9] OE2: Output Enable Bit

The OE2 bit enables/disables the PWM output.

Bit	Description
0	Disables the output of PWM2P and PWM2M.
1	Enables the output of PWM2P and PWM2M.

[bit8] OE1: Output Enable Bit

The OE1 bit enables/disables the PWM output.

Bit	Description
0	Disables the output of PWM1P and PWM1M.
1	Enables the output of PWM1P and PWM1M.

Note:

To use the PWM output pin as a general-purpose I/O port, disable the output enable bit (OE2="0", OE1="0").

5.3. PWM1 and PWM2 Compare Registers (PWC1, PWC2)

The value of the two 8 (10) bits compare register of PWM1 and PWM2 determine the width of the PWM pulse. The value "0x00" ("0x000"), which is set in the compare register, indicates that the PWM duty is 0%, while the value "0xFF" ("0x3FF"), which is set in the compare register, indicates that the PWM duty is 99.6% (99.9%).

■ PWM1 and PWM2 Compare Registers (PWC1,PWC2)

● Register Configuration: PWM1 Compare Register (PWC1)

bit	15	14	13	12	11	10	9	8
	RESV	RESV	RESV	RESV	RESV	RESV	D9	D8
Attribute	-	-	-	-	-	-	R/W	R/W
Initial value	0	0	0	0	0	0	X	X

bit	7	6	5	4	3	2	1	0
	D7	D6	D5	D4	D3	D2	D1	D0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	X	X	X	X	X	X	X	X

● Register Configuration: PWM2 Compare Register (PWC2)

bit	15	14	13	12	11	10	9	8
	RESV	RESV	RESV	RESV	RESV	RESV	D9	D8
Attribute	-	-	-	-	-	-	R/W	R/W
Initial value	0	0	0	0	0	0	X	X

bit	7	6	5	4	3	2	1	0
	D7	D6	D5	D4	D3	D2	D1	D0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	X	X	X	X	X	X	X	X

● Register Functions

[bit15 to bit10] RESV: Reserved Bits

Always write "0" to this bit.

"0" is always read from this bit.

[bit9 to bit0] D9 to D0: Compare Data

The D9 to D0 bits set the PWM pulse width.

Bit	Description
	Set the PWM pulse width.

- When the 8/10 bits switching bit is set to 8-bit operation mode (PWC:SC="0"), a compare data value is set to the D7 to D0 bits.
- When the 8/10 bits switching bit is set to10-bit operation mode (PWC:SC="1"), a compare data value is set to the D9 to D0 bits.

Notes:

- PWM1and PWM2 Compare Registers (PWC1, PWC2) can be accessed at any timing. The changed value, however, is reflected on the pulse width at the end of the current PWM cycle, after the rewrite bit (PWS2:BS) is set to "1".
- The values of the D9 and D8 bits are undefined, when the 8/10 bits operation mode (PWC:SC) is set to "0" and PWM is operating in 8-bit mode.
- PWM1 and PWM2 Compare Registers (PWC1, PWC2) must always be accessed by word access.

5.4. PWM1 and PWM2 Selection Registers (PWS1, PWS2)

The PWM1 and PWM2 Selection Registers select one of "L", "H", PWM pulse or "Hi-Z" for the output of the external pin of the Stepping Motor Controller.

■ PWM1 Selection Register (PWS1)

● Register Configuration

bit	7	6	5	4	3	2	1	0
	RESV	RESV	P2	P1	P0	M2	M1	M0
Attribute	-	-	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

● Register Functions

[bit7,bit6] RESV: Reserved Bits
Always write "0" to this bit.

"0" is always read from this bit.

[bit5 to bit3] P2 to P0: Output Select Bits
The P2 to P0 bits select the output signal for PWM1P.

bit5	bit4	bit3	Description
0	0	0	"L"
0	0	1	"H"
0	1	don't care	PWM pulse
1	don't care	don't care	"Hi-Z"

[bit2 to bit0] M2 to M0: Output Select Bits
The M2 to M0 bits select the output signal for PWM1M.

bit2	bit1	bit0	Description
0	0	0	"L"
0	0	1	"H"
0	1	don't care	PWM pulse
1	don't care	don't care	"Hi-Z"

■ PWM2 Selection Register (PWS2)

● Register Configuration

bit	15	14	13	12	11	10	9	8
	RESV	BS	P2	P1	P0	M2	M1	M0
Attribute	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

● Register Functions

[bit15] RESV: Reserved Bit

Always write "0" to this bit.

"0" is always read from this bit.

[bit14] BS: Rewrite

The BS bit synchronizes the setting for the PWM output.

Bit	Description
0	Disables the rewriting of the setting for the PWM output.
1	Enables the rewriting of the setting for the PWM output.

Until the BS bit is set to "1", changes made to the two PWM Compare registers (PWC1,PWC2) and two PWM Selection registers (PWS1,PWS2) are not reflected in the output signal.

When "1" is set to the BS bit, the PWM pulse generator and the selector load the values of the registers at the end of the current PWM cycle. The BS bit is cleared to "0" automatically at the beginning of the next PWM cycle.

Notes:

- When " BS="1" and it is set to "1" by a program at the beginning of the PWM cycle, at the same time as an automatic clear to "0", the BS bit is set to 1 (i.e. no change is made) and automatic clearing is cancelled.
- When " BS="1" and it is set to "0" by a program at the beginning of the PWM cycle, at the same time as an automatic clear to "0", the BS bit is cleared to "0", the PWM pulse generator and the selector do not load the values of the registers at the end of the current PWM cycle.
- When BS = 1, and a read-modify-write (RMW) instruction is executed other than the BS bit, "1" is read from the BS bit and "1" is written again back to the BS bit. When the BS bit is automatically cleared to "0" by the starting PWM cycle in between read and write, the BS bit is set again back to "1". Therefore, values of the registers are loaded in PWM pulse generator and selector even when the BS bit was not set to "1" by the end of the next PWM cycle.

[bit13 to bit11] P2 to P0: Output Select Bits

The P2 to P0 bits select the output signal for PWM2P.

bit13	bit12	bit11	Description
0	0	0	"L"
0	0	1	"H"
0	1	don't care	PWM pulse
1	don't care	don't care	"Hi-Z"

[bit10 to bit8] M2 to M0: Output Select Bits

The M2 to M0 bits select the output signal for PWM2M.

bit10	bit9	bit8	Description
0	0	0	"L"
0	0	1	"H"
0	1	don't care	PWM pulse
1	don't care	don't care	"Hi-Z"

6. Usage Notes

The notes when using the Stepping Motor Controller are described below.

■ Notes when Changing the PWM Setting

- Set the count enable bit (PWC:CE) to "1", after the PWM operating clock selection bits (PWC:P2 to P0) and 8/10 bits switching bit (PWC:SC) are set.

- PWM1 Compare register (PWC1), PWM2 Compare register (PWC2), PWM1 Selection register (PWS1), and PWM2 Selection register (PWS2) can always be accessed. To change the setting of the PWM's "H" width or PWM output, write the setting values to PWM1 Compare register, PWM2 Compare register, PWM1 Selection register, and PWM2 Selection register, then set the Rewrite bit (PWS2:BS) to "1" (or do this simultaneously).
  - If the Rewrite bit (PWS2:BS) is set to "1", the new setting value will become effective at the end of the current PWM cycle, and at the same time, the PWS2:BS bit is automatically cleared to "0".
  - Also, if writing "1" to the Rewrite bit (PWS2:BS) and an automatic clear of the Rewrite bit (PWS2:BS) to "0" at the beginning of the PWM cycle both occur at the same time, "1" is written to the Rewrite bit (PWS2:BS) and the clear of the Rewrite bit (PWS2:BS) to "0" is cancelled.
  - When the Rewrite bit (PWS2:BS) is set to "1", and a read-modify-write (RMW) instruction is executed other than the Rewrite bit (PWS2:BS), "1" is read from the Rewrite bit (PWS2:BS) and "1" is written again back to the Rewrite bit (PWS2:BS). When the Rewrite bit (PWS2:BS) is automatically cleared to "0" by the starting PWM cycle in between read and write, the Rewrite bit (PWS2:BS) is set again back to "1". Therefore, values of the registers are loaded in PWM pulse generator and selector even when the Rewrite bit (PWS2:BS) was not set to "1" by the end of the next PWM cycle.

  Unless the PWM pulse generator is started (PWC:CE="1") in order to perform the above operation, the new setting value will not become effective; therefore, the selection of the motor drive signal cannot be changed either.

- If the PWM pulse generator is stopped (PWC:CE="0") when the PWM pulse is selected by the output select bits (PWS1:P2 to P0, PWS1:M2 to M0, PWS2:P2 to P0, PWS2:M2 to M0), the pin for which the PWM pulse is selected is fixed to the "L" level.

- When the SC bit of the PWM Control Register is set to "0" and PWM is operating in 8-bit mode, the values of the D9 and D8 bits are undefined.

- The PWM1 and PWM2 Compare Registers must always be accessed by word access.

CHAPTER: SOUND GENERATOR

This chapter explains the functions and operations of the sound generator.

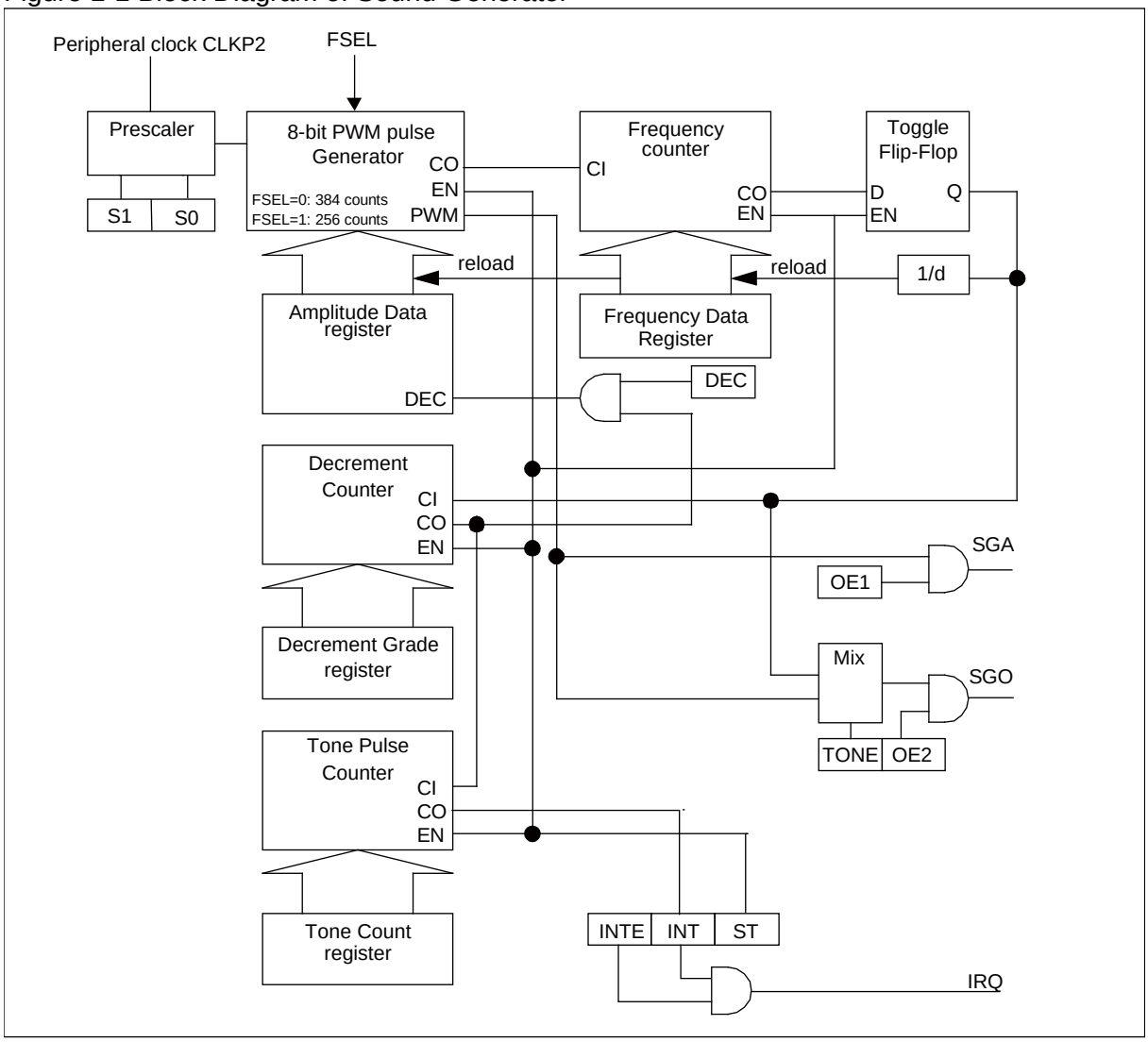
Management Code: 96F3SG-E01.0

1. Overview

The Sound Generator consists of the Sound Control Register, Frequency Data Register, Amplitude Data Register, Decrement Grade Register, Tone Count Register, PWM Pulse Generator, Frequency Counter, Decrement Counter and Tone Pulse Counter.

■ Block Diagram of Sound Generator

Figure 1-1 Block Diagram of Sound Generator



2. Registers

This section explains the configuration and functions of the registers used for the Sound Generator.

Note:

The Sound Generator Control Register SGCRn can be accessed as a 16-bit register or as two 8-bit registers SGCRHn and SGCRLn.

■ List of Sound Generator Registers

Abbreviated Register Name	Register Name	Reference
SGCRn	Sound Generator Control Register	See 2.1
SGARn	Amplitude Data Register	See 2.2
SGFRn	Frequency Data Register	See 2.3
SGTRn	Tone Count Register	See 2.4
SGDRn	Decrement Grade Register	See 2.5

2.1. Sound Generator Control Register (SGCRn)

The Sound Control Register (SGCRn) controls the operation status of the Sound Generator by controlling interrupts and setting the external output pins.

■ Sound Generator Control Register Higher (SGCRHn)

SGCRHn								
bit	15	14	13	12	11	10	9	8
	RESV	-	-	-	-	FSEL	BUSY	DEC
Attribute	R/W	-	-	-	-	R/W	R	R/W
Initial value	X	X	X	X	X	1	0	0

[bit15] RESV: Reserved Bit

Always write "0" to this bit. Read value is undefined. Read-Modify-Write is not affected.

[bit14 to bit11] -: Undefined

Write always '0'

read value is undefined

Read modify write operations to this register have no effect on this bit.

[bit10] FSEL: PWM Counter Switching Bit

Bit	Description
0	384 count
1	256 count

This bit switches the count value of the PWM counter between 256 and 384. Stop operation when switching the count value. After FSEL is switched, set the amplitide data, frequency data, decrement grade and tone register again.

[bit9] BUSY: Busy Bit

Bit	Description
0	Operation completed
1	Generator busy

This bit indicates whether the sound generator is in operation. This bit is set to "1" upon the ST bit is set to "1". It is reset to "0" when the ST bit is reset to "0" and the operation is completed at the end of one tone cycle. Any write instructions performed on this bit has no effect.

[bit8] DEC: Auto-decrement Enable Bit

Bit	Description
0	Auto-decrement disabled
1	Auto-decrement enabled

The DEC bit is prepared for an automatic degradation of the sound in conjunction with the Decrement Grade Register.
If this bit is set to "1", the stored value in the Amplitude Data Register is decremented by 1(one), every time when the Decrement counter counts the number of tone pulses from the toggle flip-flop specified by the Decrement Grade register.

■ Sound Generator Control Register Lower (SGCRLn)

SGCRLn								
bit	7	6	5	4	3	2	1	0
	S1	S0	TONE	OE2	OE1	INTE	INT	ST
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

[bit7, bit6] S1, S0: Operation Clock Select Bits

bit7	bit6	Description
0	0	Peripheral clock CLKP2
0	1	1/2 Peripheral clock CLKP2
1	0	1/4 Peripheral clock CLKP2
1	1	1/8 Peripheral clock CLKP2

These bits specify the clock input signal for the sound generator.

[bit5] TONE: Tone Output Bit

Bit	Description
0	Tone and PWM mixed
1	SGO output

When this bit is set to "1", the SGO signal becomes a simple square-waveform (tone pulses) from the toggle flip-flop. Otherwise it is the mixed (AND logic) signal of the tone and PWM pulses.

[bit4] OE2: Sound Output Enable Bit

Bit	Description
0	General purpose pin
1	SGO Output enabled

When this bit is set to "1", the external pin is assigned as the SGO output. Otherwise the pin can be used as a general purpose I/O. To enable the SGO output, the corresponding bit of the port direction register should also be set to "1".

[bit3] OE1: Amplitude Output Enable Bit

Bit	Description
0	General purpose pin
1	SGA Output enbaled

When this bit is set to "1", the external pin is assigned as the SGA output. Otherwise the pin can be used as a general purpose I/O. To enable the SGA output, the corresponding bit of the Port Direction register should also be set to "1".
The SGA signal is the PWM pulses from the PWM pulse generator representing the amplitude of the sound.

[bit2] INTE: Interrupt Enable Bit

Bit	Description
0	Interrupt disabled
1	Interrupt enabled

This bit enables the interrupt signal of the Sound Generator. When this bit is "1" and the INT bit is set to "1", the Sound Generator signals an interrupt.

[bit1] INT: Interrupt Bit

Bit	Description	
	Read	Write
0	no interrupt	clear interrupt
1	interrupt request	no effect

This bit is set to "1" when the Tone Pulse counter counts the number of the tone pulses specified by the Tone Count register and Decrement Grade register.
This bit is reset to "0" by writing "0". Writing "1" has no effect and Read-Modify-Write instructions always result in reading "1".

[bit0] ST: Start Bit

Bit	Description
0	Stop operation
1	Start operation

This bit is for starting the operation of the Sound Generator. While this bit is "1", the Sound Generator perform its operation.
When this bit is reset to "0", the Sound Generator stops its operation at the end of the current tone cycle. The BUSY bit indicates whether the Sound Generator is fully stopped.
When this bit is changed from "0" to "1", the value of Frequency Data register, Amplitude Data register, Decrement Grade register, and Tone Count register is loaded into each counter.

2.2. Amplitude Data Register (SGARn)

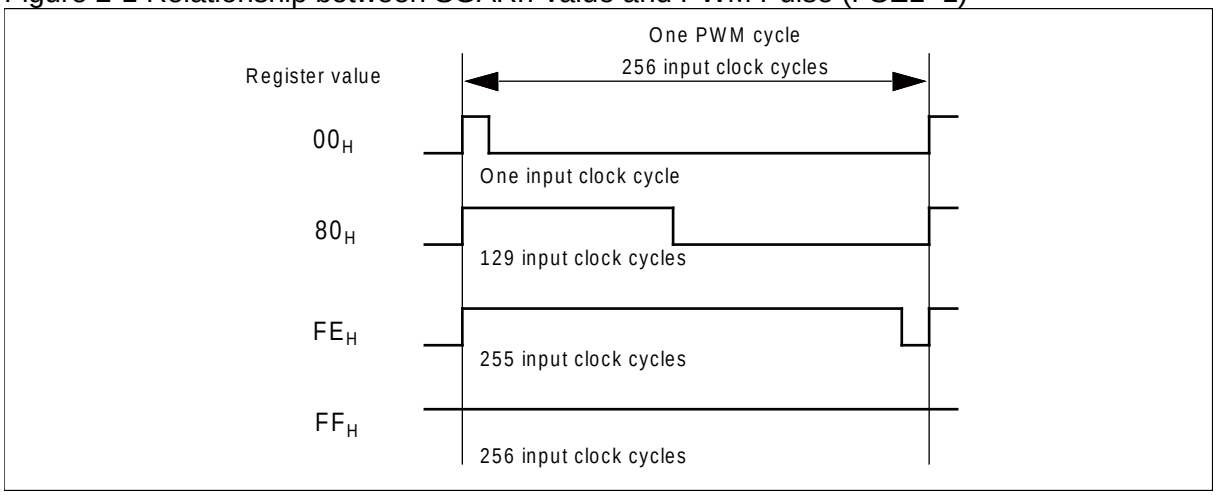
The Amplitude Data Register (SGARn) stores the reload value for the PWM pulse generator. The register value represents the amplitude of the sound. The register value is reloaded into the PWM pulse generator at falling edge of tone signal.

■ Amplitude Data Register (SGARn)

SGARn								
bit	7	6	5	4	3	2	1	0
	D7	D6	D5	D4	D3	D2	D1	D0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	X	X	X	X	X	X	X	X

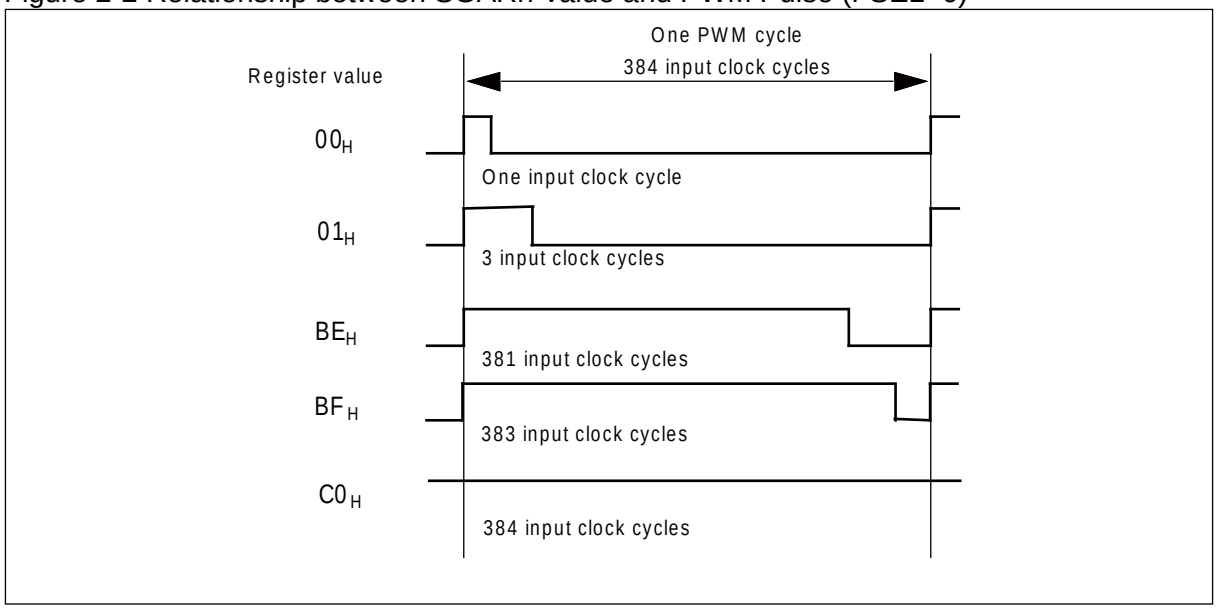
When the DEC bit is "1" and the Decrement counter reaches its reload value, this register value is decremented by 1(one). When the register value reaches "00", further decrements are not performed. However the Sound Generator continues its operation until the ST bit is cleared.
Figure 2-1 and Figure 2-2 show the relationship between the register value and the PWM pulse.

Figure 2-1 Relationship between SGARn Value and PWM Pulse (FSEL=1)



When the register value is set to "FF_H", the PWM signal is always "1".

Figure 2-2 Relationship between SGARn Value and PWM Pulse (FSEL=0)



Note:

When SGCRHn:FSEL is "0", the PWM signal is always set to "1" for $C0_H$ to FF_H.
When SGCRHn:DEC is "1", the PWM signal will fluctuate due to signal decrementation if the SGARn value is set to value greater than $C0_H$.

2.3. Frequency Data Register (SGFRn)

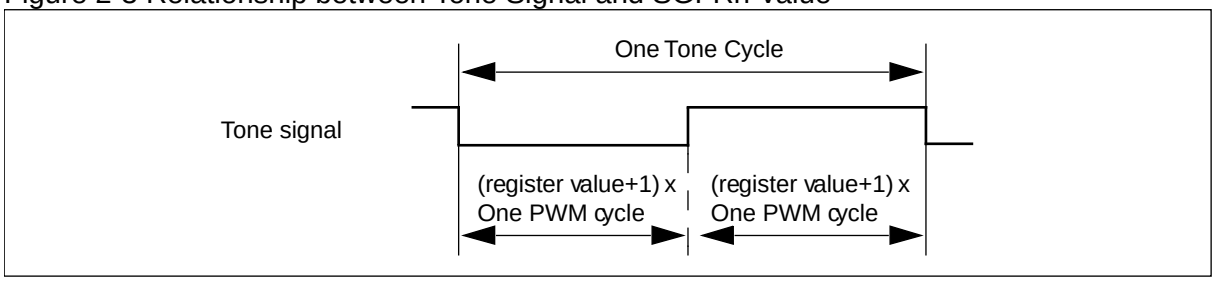
The Frequency Data Register (SGFRn) stores the reload value for the Frequency counter. The stored value represents the frequency of the sound (or the tone signal from the toggle flip-flop). The register value is reloaded into the counter at Frequency counter underflow and PWM pulse generator underflow.
The following figure shows the relationship between the tone signal and the register value.

■ Frequency Data Register (SGFRn)

SGFRn								
bit	7	6	5	4	3	2	1	0
	D7	D6	D5	D4	D3	D2	D1	D0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	X	X	X	X	X	X	X	X

Figure 2-3 shows the relationship between a tone signal and a register value.

Figure 2-3 Relationship between Tone Signal and SGFRn Value



It should be noted that modifications of the register value during operation may alter the duty cycle of 50% depending on the timing of the modification.

2.4. Tone Count Register (SGTRn)

The Tone Count Register (SGTRn) stores the reload value for the Tone Pulse counter. The Tone Pulse counter accumulates the number of tone pulses (or number of decrement operations) and when it reaches the reload value it sets the INT bit, reducing the frequency of interrupts by the number of programmed tones. The register value is reloaded into the counter at Tone Pulse counter underflow, Decrement counter underflow, and falling edge of tone signal.

■ Tone Count Register (SGTRn)

SGTRn								
bit	7	6	5	4	3	2	1	0
	D7	D6	D5	D4	D3	D2	D1	D0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	X	X	X	X	X	X	X	X

The count input of the Tone Pulse counter is connected to the carry-out signal from the Decrement counter. When the Tone count register is set to "00", the Tone Pulse counter sets the INT bit every carry-out from the Decrement counter. Thus the number of accumulated tone pulses is:

((Decrement Grade register) + 1) × ((Tone Count register) + 1)

i.e. when both registers are set to "00_H", the INT bit is set every tone cycle.

2.5. Decrement Grade Register (SGDRn)

The Decrement Grade Register (SGDRn) stores the reload value for the Decrement counter. It automatically decrements the reload value of the Amplitude Data register to reduce the frequency of interrupts. The register value is reloaded into the counter at Decrement counter underflow and falling edge of tone signal.

■ Decrement Grade Register (SGDRn)

SGDRn								
bit	7	6	5	4	3	2	1	0
	D7	D6	D5	D4	D3	D2	D1	D0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	X	X	X	X	X	X	X	X

When the DEC bit is "1" and the Decrement counter counts the number of tone pulses up to the reload value, the stored value in the Amplitude Data register is decremented by 1(one) at the end of the tone cycle.
This operation realizes automatic degradation of the sound with fewer number of CPU interventions.
It should be noted that the number of the tone pulses specified by this register equals to "register value + 1".
When the Decrement Grade register is set to "00_H ", the decrement operation is performed every tone cycle.

CHAPTER: ROM/RAM MIRRORING MODULE

This chapter explains the ROM/RAM mirroring module.

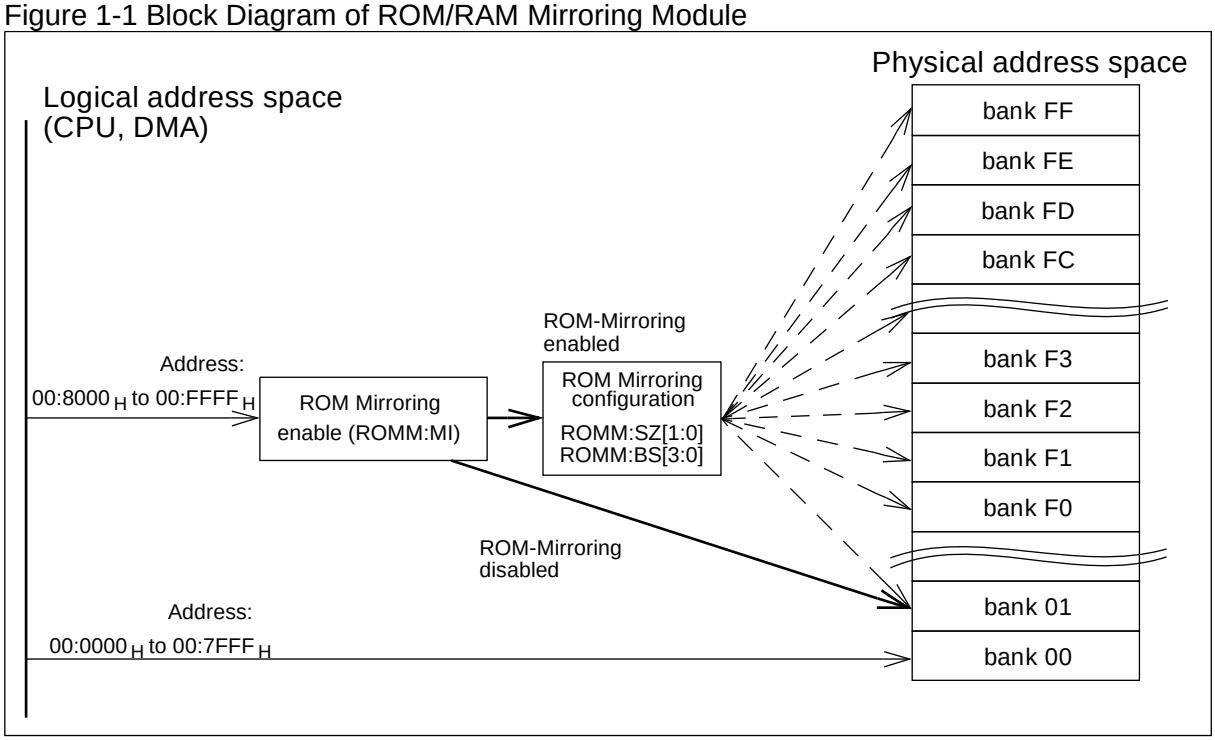
Management Code: 96F3MIRROR-E01.0

1. Overview

The ROM/RAM-Mirroring module maps the logical address (e.g. CPU address) of bank 00 to a physical address in a selected ROM bank. The mirrored ROM size is configurable. Unused ROM-Mirror area is used for mirroring RAM from bank 01.

■ Block Diagram of ROM/RAM Mirroring Module

Figure 1-1 Block Diagram of ROM/RAM Mirroring Module



Only ROM-addresses of the upper half of ROM banks $F0_H$ to FF_H ($Fx:8000_H$ to $Fx:FFFF_H$) can be mirrored to bank 00_H. The lower half of a ROM bank is never mirrored. If the ROM-Mirroring function is disabled, bank 01_H is mirrored to bank 00_H.

2. Registers

The ROM Mirroring register (ROMM) configures all functions of the ROM mirroring module:
- the mirrored ROM bank
- the size of the mirrored ROM
- the ROM mirror enable

■ List of ROM Mirroring Register

Abbreviated Register Name	Register Name	Reference
ROMM	ROM mirroring register	See 2.1

2.1. ROM Mirroring Register (ROMM)

■ ROM Mirroring Register (ROMM)

ROMM								
bit	7	6	5	4	3	2	1	0
	BS3	BS2	BS1	BS0	-	SZ1	SZ0	MI
Attribute	R/W	R/W	R/W	R/W	-	R/W	R/W	R/W
Initial value	1	1	1	1	X	1	1	1

[bit7 to bit4] BS[3:0]: Bank Selection Bits

Bit	Description
0000	part of Bank $F0_H$ is mirrored
....	part of Bank Fx is mirrored
1111	part of Bank FF_H is mirrored

These bits select the mirrored ROM-bank. BS[3:0] correspond to bit[3:0] of the memory bank address.

Bit	mirrored Memory bank
1111	FF
1110	FE
1101	FD
...	...
0001	F1
0000	F0

The initial value is "1111" (Bank FF_H selected).

[bit3] -: Undefined

Write always "0" to this bit.
The read value is undefined.
RMW instructions are not affected.

[bit2, bit1] SZ[1:0]: Mirror Size Select Bits

bit2	bit1	Description
0	0	8kB ROM mirrored
0	1	16kB ROM mirrored
1	0	24kB ROM mirrored
1	1	32kB ROM mirrored

These bits select the size of the mirrored ROM-Bank. See Table 2-1 for detailed description.
The initial value is "11".

[bit0] MI: Mirror Enable Bit

Bit	Description
0	ROM-Mirroring function is disabled
1	ROM-Mirroring function is enabled

This bit enables the ROM mirroring function.

- Writing "0" disables the ROM-Mirroring function. The addresses $01:8000_H$ to $01:FFFF_H$ are mirrored to bank 00.
- Writing "1" enables the ROM-Mirroring function. The image of the ROM data in the bank selected by ROMM:BS[3:0] can also be accessed in the 00 bank.
- The initial value is "1" (ROM-Mirror enabled).

If the selected ROM-Mirror size is less than 32kB, address area from bank 01_H is mirrored to bank 00_H instead. If bank 01_H contains RAM above address $01:8000_H$ this RAM can be accesses via bank 00_H.

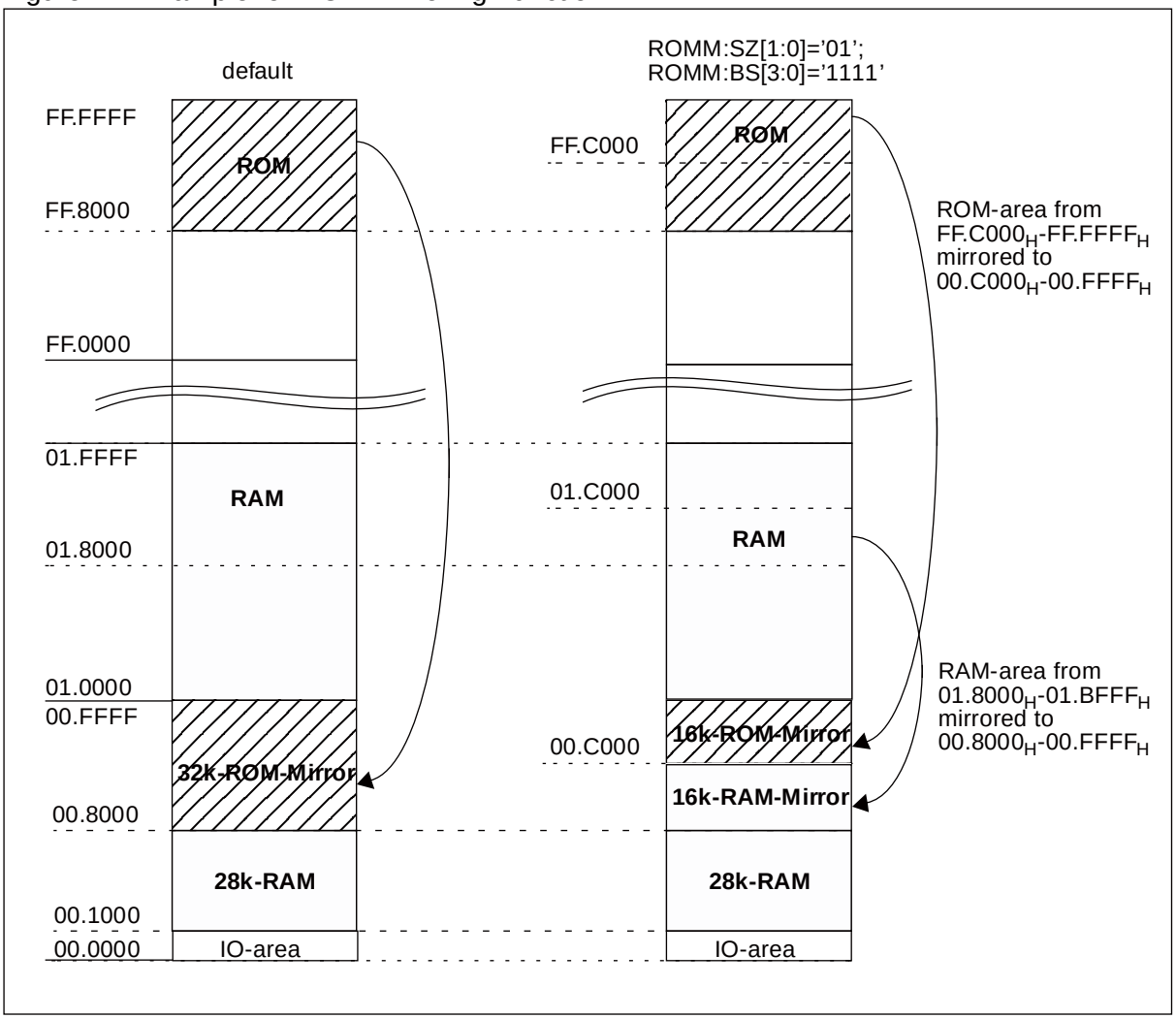
Table 2-1 Description of Mirror Size Select Bits SZ[1:0]

MI	SZ1	SZ0		mirrored ROM/RAM-Size	selected ROM/RAM-area for mirroring (physical address)	ROM/RAM-mirrored to area (logical address)
0	-	-	ROM	0kB	no ROM mirroring	no ROM mirroring
			RAM	32kB	$01:8000_H$-$01:FFFF_H$	$00:8000_H$-$00:FFFF_H$
1	0	0	ROM	8kB	$Fx:E000_H$ - $Fx:FFFF_H$	$00:E000_H$ - $00:FFFF_H$
			RAM	24kB	$01:8000_H$-$01:DFFF_H$	$00:8000_H$-$00:DFFF_H$
1	0	1	ROM	16kB	$Fx:C000_H$ - $Fx:FFFF_H$	$00:C000_H$ - $00:FFFF_H$
			RAM	16kB	$01:8000_H$-$01:BFFF_H$	$00:8000_H$-$00:BFFF_H$
1	1	0	ROM	24kB	$Fx:A000_H$ - $Fx:FFFF_H$	$00:A000_H$ - $00:FFFF_H$
			RAM	8kB	$01:8000_H$-$01:9FFF_H$	$00:8000_H$-$00:9FFF_H$
1	1	1	ROM	32kB	$Fx:8000_H$ - $Fx:FFFF_H$	$00:8000_H$ - $00:FFFF_H$
			RAM	0kB	no RAM-Mirroring	no RAM-Mirroring

Note:

- Do not write the ROM mirroring register (ROMM) when code is executed from addresses inside the mirrored memory area.
- At least, one NOP instruction or one instruction which do not access to mirrored memory area must follow the instruction that set to ROM mirroring register (ROMM).

Figure 2-1 Example for ROM-Mirroring Function



CHAPTER: DUAL OPERATION FLASH MEMORY

This chapter explains the functions and operation of the flash memory.

Management Code: 96F6FLASH-E01.0

1. Overview

Dual Operation Flash consists of the upper bank and lower bank, allowing concurrent execution of an erase/program and a read in two banks, which is not allowed in conventional flash products. The flash memory is used to store the user program and data. See "User ROM Memory map for Flash devices" in the datasheet for a detailed flash configuration of each device.
Instructions from the CPU can be used to write data to and erase data from the flash memory. Internal CPU control therefore enables rewriting of the flash memory while the MCU is mounted on a PCB. As a result, improvements in programs and data can be performed efficiently.

■ Flash Memory Types

The datasheet describes a detailed sector configuration for each device.

● Flash

This is the standard flash macro which is optimized for high read performance. Up to two modules of this type (Flash A and Flash B) are available on F^2MC-16FX MCUs.
The Flash A is mapped in the CPU memory map to the banks DF to FF. Small sectors are located in bank DF. Depending on the size of the Flash A, large sectors are located starting at bank F8 or higher up to bank FF.
On some devices, a second flash memory, called Flash B is available which can be operated independently. The small sectors of this Flash are located in bank DE. Devices with a bigger Flash B also have large sectors between bank F0 and F7.
Small sectors can be used for on-chip E^2PROM emulation, boot loader etc..

■ Flash Memory Features

● Features of Flash

- Use of automatic program algorithm.
- Erase pause/restart functions provided.
- Detection of completion of writing/erasing using data polling or toggle bit functions.
- Detection of completion of writing/erasing using CPU interrupts.
- Sector erase function (any combination of sectors).
- Detection of completion of writing/erasing using DFSA/DFSB register.
- Program/erase possible by CPU (user program and serial flash writer) and in Flash mode (parallel flash writer).
- Byte and Word programming is possible in CPU and Flash mode.
- Flash reading cycle time: Minimum of 1 machine cycles.
- CPU reading is done 16 bit wide.
- Programmable erase/write protection in CPU mode (sector-wise).
- Code security to prevent read-out in a way not intended by the application.
- 2 CPU write access modes: direct access and command sequencer access.
- Support for data programming by DMA.
- An erase/program and a read can be executed concurrently in two banks configuration

● Dual Operation Flash Memory

Programming and erasing flash memories are enabled by an instruction from the CPU via the flash memory I/F circuit. This allows efficient reprogramming and programming data in the mounted state under CPU control.
The minimum sectors are as compact as 8 Kbytes that can be handled easily as program/data areas.
Data can be reprogrammed not only by program execution in RAM but also by program execution in flash memory because of dual operation. In addition, an erase/write and reading of the different banks (the upper and lower banks) can be executed concurrently.

Upper bank	Lower bank
Read	
Read	Write/Sector erase
Write/Sector erase	Read
Chip erase	

Note:

The Flash memory manufacturer code and device code are not available.

■ Writing to/Erasing Flash Memory

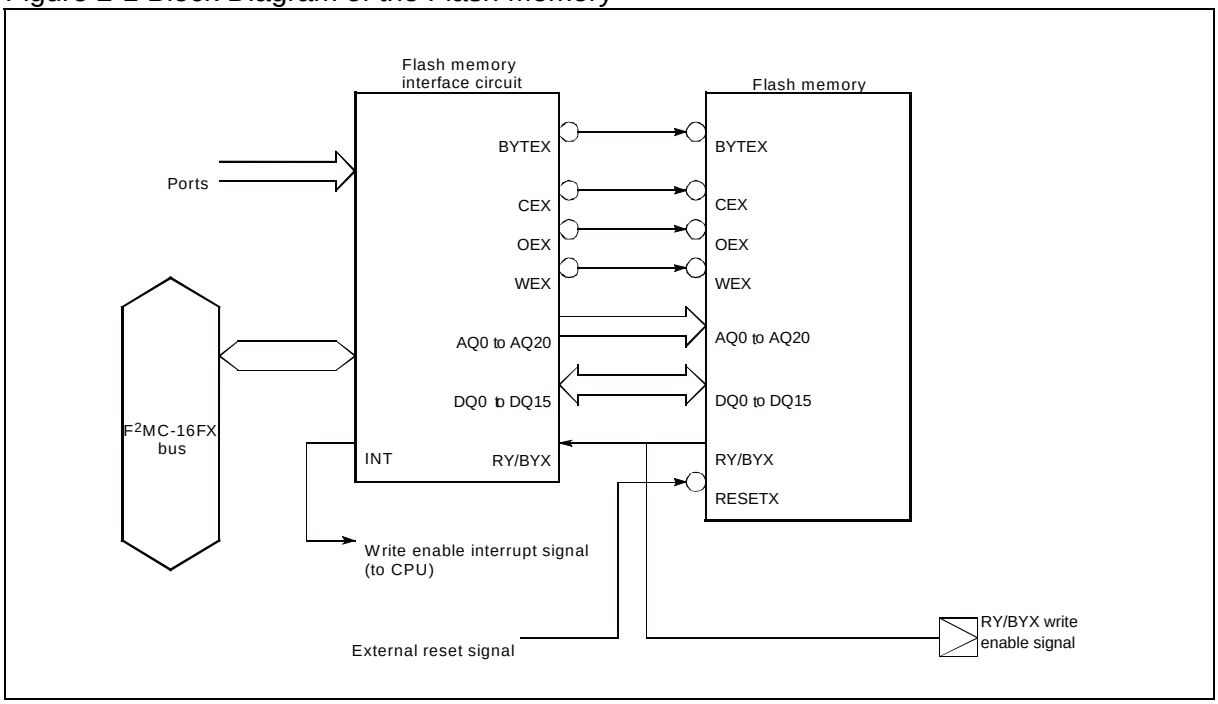
A sector erase operation can be suspended. In this suspend state it is possible to read data from other sectors. The dual operation flash memory enables program execution from the flash memory and programming the data to the flash memory with using interrupt.
It also eliminates the need for a conventional process to download a program to RAM for executing to program the data into the flash memory, resulting in reduced download time and no need to consider power shutdown for RAM data maintenance. When one bank is in state of write/sector erase (busy state), the other bank can fetch program. The user program must not fetch an instruction to the busy state bank.

2. Block Diagram and Sector Configuration

This chapter shows a basic block diagram of the embedded flash memory and the global sector configuration.

■ Block Diagram of the Flash Memory

Figure 2-1 Block Diagram of the Flash Memory



■ Sector Configuration of the Flash Memory

Figure 2-2 shows the sector configuration of the Flash memory. The addresses in the figure indicate the high-order and low-order addresses of each sector.

Depending on the size of the Flash A memory, up to 4 small sectors of 8KByte size each are available in SA1 to SA4, starting at SA1 upwards. And Flash A has one 512bytes sector for Flash Security and MCU configuration. It called SAS and located below SA1. It has 8KB address space but it has only 512bytes. The data is mirrored 16 times in their address space. (For example: Accessing to $DF0200_H$ can read the data of $DF0000_H$)

Depending on the size of the Flash A memory, a number of large sectors of 64KByte size are available in SA39 to SA32, starting at SA39 downwards.

When the Flash B memory is available on the product, depending on the size of the Flash B memory, up to 4 small sectors of 8KByte size each are available in SB1 to SB4, starting at SB1 upwards. Flash B also has one 512bytes sector. This is the same as Flash A.

Depending on the size of the Flash B memory, an additional number of large sectors of 64KByte size are available in SB31 to SB24 starting at SB31 downwards.

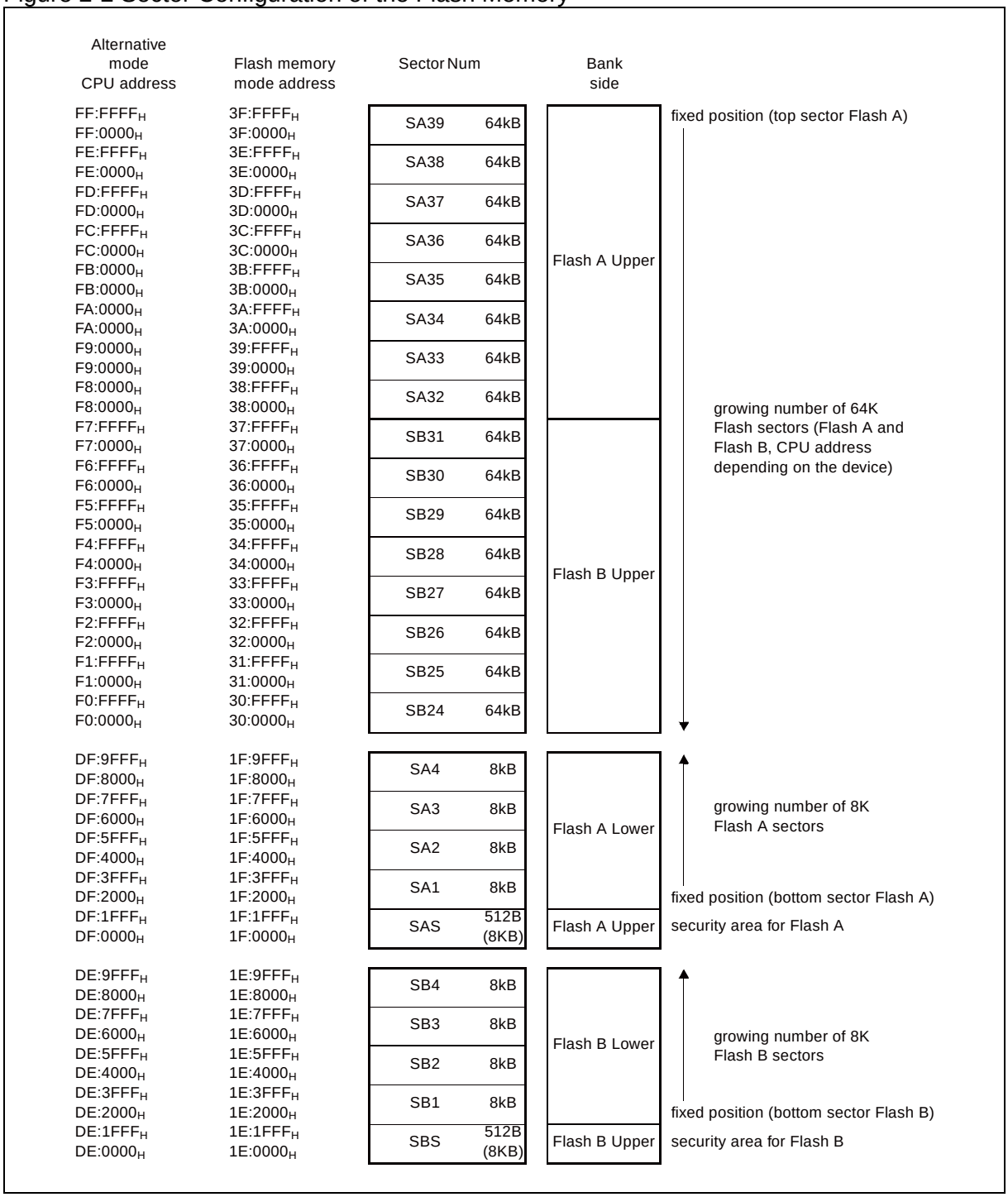
See the datasheet for the availability of all sectors in each device.

■ Bank Configuration

The flash memory consists of two banks.

Upper one ranging from SA32 to SA39 and SAS, from SB24 to SB31 and SBS depending on the size of Flash A and Flash B. Lower one from SA1 to SA4 (Flash A), one from SB1 to SB4 (Flash B). See the datasheet for the availability of all sectors in each device.

Figure 2-2 Sector Configuration of the Flash Memory



3. Modes

The flash memory can be accessed in two different ways: Flash memory mode and CPU mode. Flash memory mode enables data to be directly written to or erased from the external pins. The CPU mode enables data to be written to or erased from the CPU via the internal bus. Use the mode pins MD, P17_0, DEBUGI/F to select the mode as described in Table 7.4-1.

■ Flash Memory Mode

The CPU stops when the mode pins are set to MD=1, MD=1, P17_0=1, DEBUG I/F=1 while the reset signal is asserted. The flash memory interface circuit is connected directly to ports, enabling direct control from the external pins. This mode makes the MCU seem like a standard flash memory to the external pins, and write/erase can be performed using a flash memory programmer.
In flash memory mode, all operations supported by the flash memory automatic algorithm can be used.
For more details about the Flash memory mode, refer to the parallel Flash programming specification.

Note:

The sector protect function of MBM29LV200TC via V_{ID} (12 V) pin is not supported. Instead, there is a sector write protection function that prevents accidental erase/write in CPU mode. Please see "Section 7.5 Dual Operation Flash Write Access Control Register 0-1 (DFWC0A, DFWC0B, DFWC1A, DFWC1B)" for details.

■ CPU Mode

The Flash memory is located in the DE to FF banks in the CPU memory space. The flash memory can be read-accessed like ordinary mask ROM and also program-accessed from the CPU via the flash memory interface circuit.
Since writing/erasing the flash memory is performed by instructions from the CPU via the flash memory interface circuit, this mode allows rewriting even when the MCU is soldered on the target board.

4. Starting the Flash Memory Automatic Algorithm

Write and erase to flash memory is performed by launching the flash memory's own Automatic Algorithms. The following commands are available: Read/Reset, Write, Chip Erase and Sector Erase. The erase suspend and restart function is available during Sector erase.

■ Command Sequence Table

Automatic Algorithms for flash memory write/erase are launched by writing one to six bytes or words to the Flash memory in succession according to Table 4-1.

The data of the commands must be written to the low-order byte (except the program data of the write command). Hence the commands must be written to an even address.

Commands can be written in byte or word mode. The data written to the high-order byte (in case of a word access) is ignored. The data width used for the 4th bus write cycle of the write command (program address and program data) determines the write mode of the Flash (byte or word).

This interface includes a Write command sequencer which can automatically send the write sequence to the Flash. This Write command sequencer supports both byte and word programming.

Disable the Write command sequencer (DFCA, DFCB:CSWE="0") for submitting any of the commands below.

Table 4-1 Command Sequence Table

Command sequence	Bus write access	1st bus write cycle		2nd bus write cycle		3rd bus write cycle		4th bus write cycle		5th bus write cycle		6th bus write cycle	
		Address	Data	Address	Data	Address	Data	Address	Data	Address	Data	Address	Data
Read/Reset	1	$mmmXXX_H$ (even)	$F0_H$	-	-	-	-	-	-	-	-	-	-
Write program	4	$mmmAAA_H$	AA_H	$mmm554_H$	55_H	$mmmAAA_H$	$A0_H$	PA	PD	-	-	-	-
Chip Erase	6	$mmmAAA_H$	AA_H	$mmm554_H$	55_H	$mmmAAA_H$	80_H	$mmmAAA_H$	AA_H	$mmm554_H$	55_H	$mmmAAA_H$	10_H
Sector Erase	6	$mmmAAA_H$	AA_H	$mmm554_H$	55_H	$mmmAAA_H$	80_H	$mmmAAA_H$	AA_H	$mmm554_H$	55_H	$sssXXX_H$ (even)	30_H
Sector Erase Suspend	1	$sssXXX_H$ (even)	$B0_H$	-	-	-	-	-	-	-	-	-	-
Sector Erase Restart	1	$sssXXX_H$ (even)	30_H	-	-	-	-	-	-	-	-	-	-

Notes:

- The addresses in the table are the values in the CPU memory map. All addresses and data are represented using hexadecimal notation. The letter X indicates an arbitrary value.
- CPU/Flash address bit 1 is ignored for command writing (except for the program address PA). Hence a command can be written for example to address AAA or AA8.
- The addresses mmm in the table must point to the flash memory non-write protected sector (Flash A/B) on which the command is meant to operate. Otherwise the write command can not be sent to the Flash.
- The addresses sss in the table must point to the flash memory sector on which the sector command is meant to operate. See "and Section 2. Block Diagram and Sector Configuration".
- PA: Write address. Only even addresses can be specified when writing in word mode. For byte mode, also odd addresses are permitted.
- PD: Write data.
- Writing an illegal address or data, or writing them in the incorrect order, will reset the Flash memory to the read mode.
- It is possible to read data from the Flash between the write cycles of the commands. The command execution starts after the last write cycle.
- Writing a command to the Flash is possible only when the corresponding write enable signal (DFCA:WE, DFCB:WE) is set to 1.
- The commands must not be written to sectors which are write protected. Any write access to a write protected sector is regarded as the illegal access.
  The commands are canceled by the Flash IF.
- Read/Reset command can be used to abort the command sequence. 1st to 2nd write commands and 1st to 5th chip/sector erase commands are canceled by Read/Reset command after sending those commands. After sending the last command, Read/Reset command can not be used to cancel the write or erase operation.

5. Confirming the Automatic Algorithm Execution State

The Flash memory performs the write/erase sequence via automatic algorithms. It thus has hardware for informing the outside world when it has finished internal operations.

■ Hardware Sequence Flags

A read access to the target flash sector returns the status of the hardware sequence flags instead of the flash memory data as long as the automatic algorithm is ongoing. They can be used to check the current status of the write/erase operation. The Hardware sequence flags can read from an even address or an odd address. The hardware sequence flags consist of the bits DQ[15,7], DQ[14,6], DQ[13,5], DQ[11,3] and DQ[10,2]. The functions of these bits are:

- Data polling flag (DQ[15,7])
- Toggle bit flag (DQ[14,6])
- Timing limit exceeded flag (DQ[13,5])
- Sector erase timer flag (DQ[11,3])
- Toggle bit-2 flag (DQ[10,2])

Table 5-1 Bit Assignments of Hardware Sequence Flags

Bit No.	[15,7]	[14,6]	[13,5]	[12,4]	[11,3]	[10,2]	[9,1]	[8,0]
Hardware sequence flag	DQ[15,7]	DQ[14,6]	DQ[13,5]	-	DQ[11,3]	DQ[10,2]	-	-

Notes:

- The values of the other bits ([12,9,8,4,1,0]) are not defined. Software must be written in a way to tolerate any value of these bits.
- Reading the hardware sequence flags of the Flash can be done independently of the setting of the corresponding write enable signal (DFCSA:WE, DFCSB:WE).

Note:

For reading the hardware sequence flags of the Flash, the following rules must be adhered:

For checking whether automatic writing or erasing is being executed, the hardware sequence flags or the status register (DFSA, DFSB) can be read. The hardware sequence flags however show more detailed information.
After writing/erasing has terminated successfully, the Flash returns to the read/reset state. This must be confirmed by one of these flags (RDY or hardware sequence flags) before performing the next operation such as data read to the same bank or data write.
In addition, the hardware sequence flags can be used to confirm whether the second or subsequent sector erase code write is valid.

Table 5-2 shows the function of all flags in each Flash status.

Table 5-2 Hardware Sequence Flag and Status Register Functions

Status		Accessed address for hardware sequence flag	Hardware sequence flag				
			DQ[15,7]	DQ[14,6]	DQ[13,5]	DQ[11,3]	DQ[10,2]
Normal operation	Read/Reset	Any address	DATA:[15,7]	DATA:[14,6]	DATA:[13,5]	DATA:[11,3]	DATA:[10,2]
	Write (embedded program algorithm running)	Address being programmed (Bank include the writing address)	DATA:[15,7]	Toggle	0	0	0
		Other bank address	DATA:[15,7]	DATA:[14,6]	DATA:[13,5]	DATA:[11,3]	DATA:[10,2]
	Chip erase (embedded erase algorithm running)	Any address	0	Toggle	0	1	Toggle
	Sector erase wait (Flash waits for inputting further sector erase requests)	Sectors which are already specified for erase	0	Toggle	0	0	Toggle
		Other sectors (Ongoing bank)	0	Toggle	0	0	Toggle
		Other sectors (Not ongoing bank)	¯DATA:[15,7]	DATA:[14,6]	DATA:[13,5]	DATA:[11,3]	DATA:[10,2]
	Sector erase (embedded erase algorithm running)	Sectors which are specified for erase (all sectors in case of Chip erase)	0	Toggle	0	1	Toggle
		Other sectors (Ongoing bank)	0	Toggle	0	1	Toggle
		Other sectors (Not ongoing bank)	DATA:[15,7]	DATA:[14,6]	DATA:[13,5]	DATA:[11,3]	DATA:[10,2]
	Sector erase suspended	Sectors which are specified for erase	No-Toggle	0	0	1	Toggle
		Other sectors	DATA:[15,7]	DATA:[14,6]	DATA:[13,5]	DATA:[11,3]	DATA:[10,2]
	Writing of another sector while Flash is in erase suspended state	Address being programmed (erase suspended bank)	¯DATA:[15,7]	Toggle	0	1	0
		Address being programmed (no erase suspended bank)	¯DATA:[15,7]	Toggle	0	1	0
	Chip/sector erase finished (Read/Reset state)	Sector which was erased	DATA:[15,7] = 1	DATA:[14,6] = 1	DATA:[13,5] = 1	DATA:[11,3] = 1	DATA:[10,2] = 1

Status		Accessed address for hardware sequence flag	Hardware sequence flag				
			DQ[15,7]	DQ[14,6]	DQ[13,5]	DQ[11,3]	DQ[10,2]
Timing limit exceeded	Write	Address being programmed	$\overline{\text{DATA:[15,7]}}$	Toggle	1	0	0
	Write during suspend	Address being programmed	$\overline{\text{DATA:[15,7]}}$	Toggle	1	1	0
	Chip erase	Chip erase which caused the Timeout during erase	0	Toggle	1	1	Toggle
	Sector erase	Sector erase which caused the Timeout during erase	0	Toggle	1	1	Toggle
		Other sectors (sector erase is ongoing bank)	0	Toggle	1	1	Toggle
		Other sectors (sector erase is not ongoing bank)	DATA:[15,7]	DATA:[14,6]	DATA:[13,5]	DATA:[11,3]	DATA:[10,2]

The function of each hardware sequence flag is described in the following sections.

5.1. Data Polling Flag (DQ[15,7])

The data polling flag (DQ[15,7]) indicates if the automatic algorithm (write or erase) is being executed or has terminated.

■ Data Polling Flag (DQ[15,7])

The function of the DQ[15,7] flag in each Flash status is described in Table 5-2.

● Write

A read access during execution of the automatic write algorithm causes the flash memory to output the inverted value of bit[15,7] of the program data (DATA:[15,7]) regardless which Flash address is being read.
A read access after termination of the automatic write algorithm is handled as a regular Flash read access and returns bit[15,7] (DATA:[15,7]) of the currently addressed Flash cell. Hence for correctly identifying the termination of a write command, polling should always be performed from the memory location which is being programmed.

● Word and Byte Writing

A word write command writes data to the high-order byte (DATA[15:8]) and low-order byte (DATA[7:0]). DQ[15,7] shows the inverted value of DATA[15,7].
A byte write command to an even address writes data to the low-order byte at DATA[7:0] only. DQ7 shows the inverted value of DATA[7].
A byte write command to an odd address writes data to the high-order byte at DATA[15:8] only. In this case DQ15, shows the inverted value of DATA[15].

● Chip/Sector Erase

When the Flash address which belongs to the same bank to the automatic algorithm is being read. The read access during execution of the automatic erase algorithm causes the flash memory to output 0, during the sector erase wait time, also 0 is output.
A read access after termination of the automatic erase algorithm or a read access to the other bank is handled as a regular Flash read access and returns the data of the currently addressed Flash cell. Accessing an erased cell returns 1.

● Sector Erase Suspend

A read access in sector erase suspend state causes the flash memory to output the no-toggling value, if the address belongs to the sector being erased. This value is not used to determine the Flash status.
The flash memory outputs bit[15,7] (DATA:[15,7]) of the addressed memory cell if the address does not belong to the sector being erased.

Referencing this flag together with the toggle bit flag (DQ[10,2]) enables a decision to be made on whether the flash memory is in the erase suspended state and which sector is being erased.

5.2. Toggle Bit Flag (DQ[14,6])

The toggle bit flag (DQ[14,6]) together with the data polling flag (DQ[15,7]) indicates if the automatic algorithm (write or erase) is being executed or has terminated.

■ Toggle Bit Flag (DQ[14,6])

The function of the DQ[14,6] flag in each Flash status is described in Table 5-2.

● Write and Sector Erase

Successive read accesses during execution of the automatic write or erase algorithm causes the flash memory to toggle the DQ[14,6] flag for every read cycle, when the Flash address which belongs to the same bank to the automatic algorithm is being read.
A read access after termination of the automatic algorithm is handled as a regular Flash read access and returns bit[14,6] (DATA:[14,6]) of the currently addressed Flash cell. Accessing an erased cell returns 1.

● Chip Erase

Successive read accesses during execution of the automatic write or chip erase algorithm causes the flash memory to toggle the DQ[14,6] flag for every read cycle, regardless which Flash address is being read.
A read access after termination of the automatic algorithm is handled as a regular Flash read access and returns bit[14,6] (DATA:[14,6]) of the currently addressed Flash cell. Accessing an erased cell returns 1.

● Sector Erase Suspend

A read access in sector erase suspend state causes the flash memory to output DQ[14,6] = 0 if the address belongs to the sector being erased.
The flash memory outputs bit[14,6] (DATA:[14,6]) of the addressed memory cell if the address does not belong to the sector being erased.

5.3. Timing Limit Exceeded Flag (DQ[13,5])

The timing limit exceeded flag (DQ[13,5]) indicates that the execution of the automatic algorithm has exceeded the timing (internal pulse count) prescribed in the flash memory.

■ Timing Limit Exceeded Flag (DQ[13,5])

The function of the DQ[13,5] flag in each Flash status is described in Table 5-2.

● Write and Sector Erase

A read access during execution of the automatic write or erase algorithm causes the flash memory to output the status of the DQ[13,5] flag, when the Flash address which belongs to the same bank to the automatic algorithm is being read.
A read access after successful termination of the automatic write or erase algorithm is handled as a regular Flash read access and returns bit[13,5] (DATA:[13,5]) of the currently addressed Flash cell.
The DQ[13,5] flag is read as 0 as long as the program or erase time is within the prescribed time (maximum time required for write/erase). After this time has been exceeded, 1 is returned as read value.
An unsuccessful write or erase operation can be determined if DQ[13,5] is 1 while DQ[14,6] and DQ[15,7] shows that the automatic algorithm is still being executed.
For example, writing 1 to a flash memory address where 0 has been written will cause the fail state to occur. In this case, the flash memory will lock and execution of the automatic algorithm will not terminate..Note that this state does not indicate that the flash memory is faulty, but has been used incorrectly. When this state occurs, execute the Read/Reset command.

● Chip Erase

A read access during execution of the automatic chip erase algorithm causes the flash memory to output the status of the DQ[13,5] flag, regardless which Flash address is being read.
A read access after successful termination of the automatic chip erase algorithm is handled as a regular Flash read access and returns bit[13,5] (DATA:[13,5]) of the currently addressed Flash cell.
The DQ[13,5] flag is read as 0 as long as the erase time is within the prescribed time (maximum time required for erase). After this time has been exceeded, 1 is returned as read value.
An unsuccessful write or erase operation can be determined if DQ[13,5] is 1 while DQ[14,6] and DQ[15,7] shows that the automatic algorithm is still being executed.

● Sector Erase Suspend

A read access in sector erase suspend state causes the flash memory to output DQ[13,5] = 0 if the address belongs to the sector being erased.
The flash memory outputs bit[13,5] (DATA:[13,5]) of the addressed memory cell if the address does not belong to the sector being erased.

5.4. Sector Erase Timer Flag (DQ[11,3])

The sector erase timer flag (DQ[11,3]) indicates if the execution of the sector erase command has been started of if the sector erase wait period is being applied which allows to submit further sector erase commands.

■ Sector Erase Timer Flag (DQ[11,3])

After submitting a sector erase command sequence, the sector erase wait period is applied. Within this period it is possible to submit further sector erase commands. The DQ[11,3] flag can be used to identify this wait period.

The function of the DQ[11,3] flag in each Flash status is described in Table 5-2.

● Sector Erase

A read access during execution of the automatic erase algorithm causes the flash memory to output the status of the DQ[11,3] flag. A read access during the sector erase wait period returns 0. After exceeding this wait period, the actual sector erase starts and a read access to the DQ[11,3] flag returns 1. Further sector erase commands will be ignored when DQ[11,3] is 1.

A read access after successful termination of the automatic erase algorithm is handled as a regular Flash read access and returns bit[11,3] (DATA:[11,3]) of the currently addressed Flash cell. Accessing an erased cell returns 1.

See "Section 6.5 Erasing Optional Data (Sector erase)" for more details about submitting multiple sector erase commands.

The DQ[11,3] flag should be checked before and after submitting further sector erase commands. If DQ[11,3] is read as 1 after submitting the erase command, then this additional sector erase request may not be accepted.

● Sector Erase Suspend

A read access in sector erase suspend state causes the flash memory to output DQ[11,3] = 1 if the address belongs to the sector being erased.

The flash memory outputs bit[11,3] (DATA:[11,3]) of the addressed memory cell if the address does not belong to the sector being erased.

5.5. Toggle Bit-2 Flag (DQ[10,2])

The toggle bit-2 flag (DQ[10,2]) indicates if a sector is in the erase-suspended state. It can also be used to check which sectors are specified for erase.

■ Toggle Bit-2 Flag (DQ[10,2])

The DQ[10,2] toggle bit can be used together with the DQ[14,6] toggle bit to determine whether the Flash is in the sector erase suspend state or if the erase algorithm is currently being executed.
The function of the DQ[10,2] flag in each Flash status is described in Table 5-2.

● Sector/Chip Erase

Successive read accesses during execution of the automatic sector erase algorithm causes the flash memory to toggle the DQ[10,2] flag for every read cycle if the read address points to a sector which is specified for erase. A read access to another sector in ongoing bank also returns toggle. For a chip erase, DQ[10,2] toggles for all sectors.
A read access after termination of the automatic algorithm is handled as a regular Flash read access and returns bit[10,2] (DATA:[10,2]) of the currently addressed Flash cell. Accessing an erased cell returns 1.

● Sector Erase Suspend

Successive read accesses during sector erase suspend also causes the flash memory to toggle the DQ[10,2] flag for every read cycle if the read address points to a sector which is specified for erase. A read access to another sector returns bit[10,2] (DATA:[10,2]) of the currently addressed Flash cell.
Both DQ[10,2] and DQ[14,6] are used for detecting an erase-suspended sector (DQ[10,2] toggles, but DQ[14,6] does not).
Sectors which are not specified for erase can be programmed in erase suspend state. Reading from such a sector during erase-suspend-program mode returns DATA:[10,2] = 0.

● Timing Limit Exceeded

In case of an unsuccessful chip or sector erase operation (Timing limit exceeded, DQ[13,5]=1), DQ[10,2] can be used to identify which bank caused the timeout state. DQ[10,2] will only toggle when accessing the bank which caused the timeout.

6. Detailed Explanation of Writing to and Erasing Flash Memory

This section describes each operation procedure of the flash memory: Read/Reset, Write, Chip Erase, Sector Erase, Sector Erase Suspend and Sector Erase Restart.

■ Detailed Explanation of Flash Memory Write/Erase

By issuing a command sequence (see Table 4-1 in "4. Starting the Flash Memory Automatic Algorithm") the flash memory executes the automatic algorithm to perform Read/Reset, Write, Chip Erase, Sector Erase, Sector Erase Suspend or Sector Erase Restart operations.
In between write accesses of the command sequence, other bus read/write cycles can be performed as long as no write access to the flash memory other than defined by the command sequence is performed. Even reading the Flash between the write accesses is possible.
Termination of the automatic algorithm can be determined by polling the hardware sequence flags, polling the RDY flag (DFSA:RDY/DFSB:RDY) or by interrupt. At normal termination, the flash memory returns to the read/reset state.
Each operation of the flash memory is described in the following chapters:

- Setting the read/reset state
- Writing data by submitting the write command sequence
- Writing data by using the Write Command Sequencer
- Erasing all data (chip erase)
- Erasing optional data (sector erase)
- Suspending sector erase
- Restarting sector erase

6.1. Setting The Read/Reset State

This section describes the procedure for issuing the Read/Reset command to set the flash memory to the read/reset state.

■ Setting the Flash Memory to the Read/Reset State

The read/reset state is the initial state of the flash memory. When the power is turned on and when a command terminates normally, the flash memory is set to the read/reset state. In the read/reset state, any command can be input.

In the Timeout state (DQ[13,5]=1), the flash memory can be set to the read/reset state by sending the Read/Reset command in the command sequence table (see Table 4-1 in "4. Starting the Flash Memory Automatic Algorithm") continuously to the target sector in the flash memory.

The Read/Reset command is not required to read data by a regular read. The Read/Reset command is mainly used to initialize the automatic algorithm when a command does not terminate normally.

Read/Reset command can also be used to abort the command sequence. 1st to 2nd write commands and 1st to 5th chip/sector erase commands are canceled by Read/Reset command after sending those commands.

The Read/Reset command has no effect in normal operation of the write or erase automatic algorithm. It cannot be used to interrupt an ongoing write or erase command execution. Also resetting the Flash from the sector erase suspended state is not possible.

6.2. Writing Data by Submitting the Write Command Sequence

This section describes how to write data to the flash memory by sending the write command sequence.

■ Starting the Write Automatic Algorithm

The data write automatic algorithm of the flash memory can be started by sending the Write command in the command sequence table (see Table 4-1 in "4. Starting the Flash Memory Automatic Algorithm") continuously to the target sector in the flash memory. When data write to the target address is completed in the fourth command cycle, the automatic algorithm for writing is started.

● Specifying Addresses

Writing to the Flash is possible in byte or word mode. When writing in word mode, an even address must be specified in the write data cycle.
Writing can be done in any order of addresses or even if the sector boundary is exceeded. However, the Write command writes only data of one byte or word for each execution.

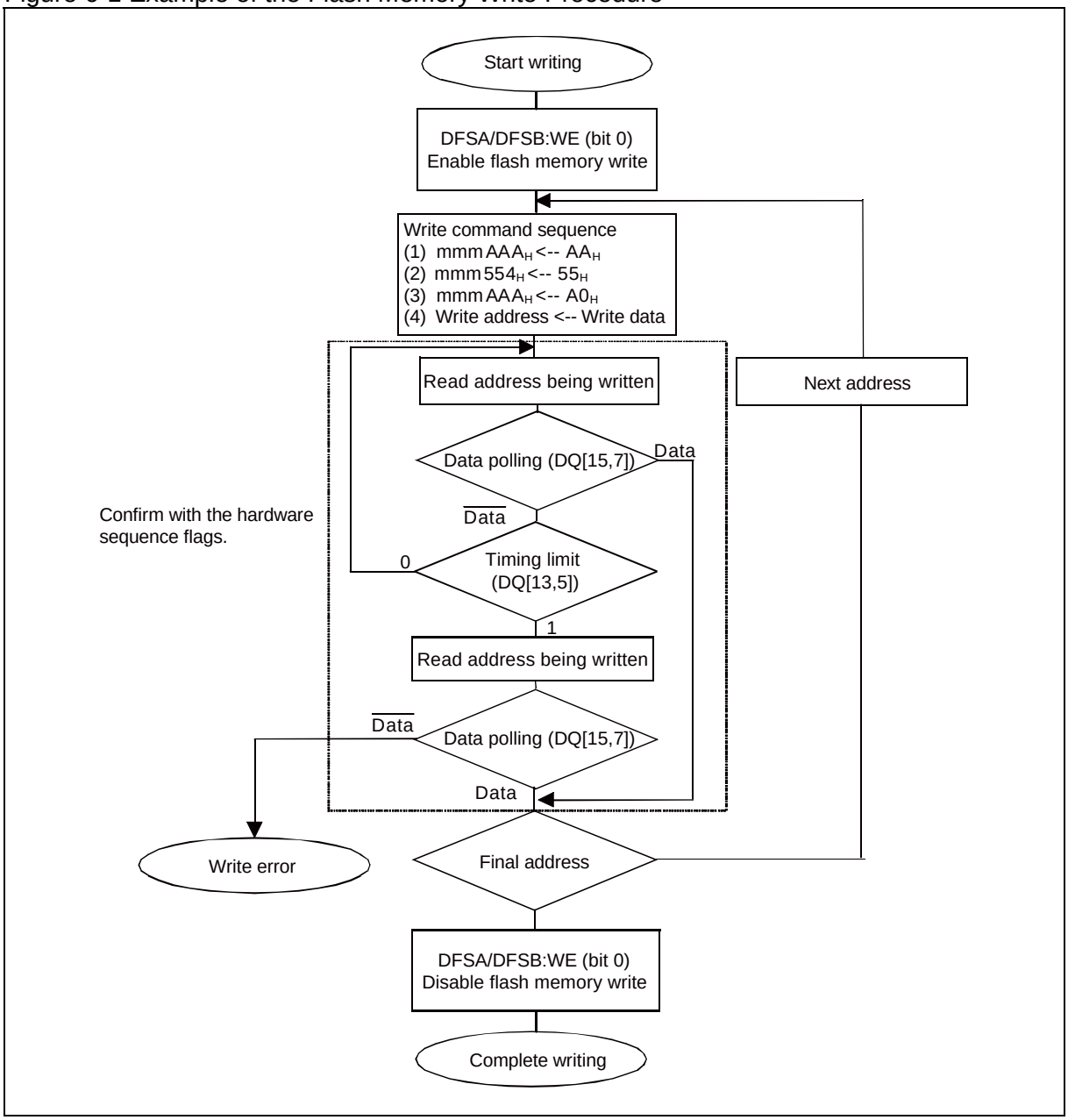
● Notes on Writing Data

Writing cannot return a data bit in the Flash from 0 to 1. When trying to program a bit to "1" which is already set to "0", the data polling algorithm (DQ[15,7]) or toggle operation (DQ[14,6]) does not terminate and the timing limit exceeded flag (DQ[13,5]) be set after the prescribed maximum time for writing. A bit programmed to "0" can only be set to "1" by an erase operation.
All commands are ignored during execution of the automatic write algorithm. If a CPU reset (Power reset, external reset, software reset, watchdog reset or clock stop reset) is asserted during writing, the data of the written addresses will be unpredictable. And entering to standby mode with low power consumption mode also causes it. (ESMCR:FPDS)

■ Example for Writing to the Flash Memory

Figure 6-1 shows an example of the procedure for writing to the flash memory. The hardware sequence flags (see "and Section 5. Confirming the Automatic Algorithm Execution State") can be used to determine the state of the automatic algorithm in the flash memory. Here, the data polling flag (DQ[15,7]) can be used to confirm that writing has terminated. The status register (DFSA/DFSB:PGMS) is also used to confirm the writing termination.
Data polling must be performed to the memory location which is being programmed. Otherwise the transition of the DQ[15,7] flag at the termination of the write command cannot be detected.
When the write automatic algorithm starts or stops, the Flash changes synchronously between the automatic algorithm execution and the read/reset or sector erase suspend state.

Figure 6-1 Example of the Flash Memory Write Procedure



Note:

The address "mmm" must point to the unprotected flash memory sector on which the command is meant to operate.

6.3. Writing the Data by using the Write Command Sequencer

This section describes how to write the data by using the Write command sequencer instead of manually submitting the write command sequence.

■ Using the Write Command Sequencer

The Flash can be written by two methods. Either in direct mode by manually submitting the write sequence as described in the previous section (DFCA: WE=1,CSWE=0/ DFCB: WE=1,CSWE=0) or by using the Write command sequencer (DFCA:WE=1,CSWE=1 / DFCB:WE=1,CSWE=1).
After activating the Write command sequencer, each CPU write access to the Flash is handled as a write request to the flash memory. The Flash interface prefixes each write data with the Write command sequence. It is not possible to submit any other command sequence to the Flash as long as DFCA/DFCB:WE, CSWE is set to "1".

● Activating and Deactivating the Write Command Sequencer

After any reset, the Write command sequencer is always disabled. It can be activated by setting DFCA/DFCB:CSWE to "1". This should be done only when the Flash is in the Read/Reset or Sector erase suspend state and no command sequence has been written to the flash (not even the first write cycle of a new command sequence).
The ongoing Write command sequencer can not stop, if the command sequencer is writing the data to the Flash, DFC register which has CSWE,WE bits can not be written.

● Specifying Addresses

Writing to the Flash in Write command sequencer mode is possible in byte or word mode. When writing in word mode, an even address must be specified.

● Writing and Reading the Flash with Activated Write Command Sequencer

Writing the Flash is possible only when the Write command sequencer is in the Idle state (DFSA/DFSB:ST[1:0]="00"). When CS is in the submitting write command state (DFSA/DFSB:ST[1:0]="01"), CPU read access is waited till end of submitting write command. User also can check the hardware sequence flags to read an ongoing bank in the checking flash state (DFSA/DFSB:ST[1:0]="10").
A flash write access when the Write command sequencer is not idle is also ignored and will set the error flag DFIS:WBINT. Always check the Write command sequencer status bits DFSA/DFSB:ST[1:0] before writing to the Flash with activated Write command sequencer. If CSWE is set but WE is not set, CS ignores the write command with no error.

■ Interrupt functions of the Write command sequencer and DMA access

The Write command sequencer can only handle one write command at a time. A second write request is ignored when a write operation is ongoing. The progress of a write operation can be checked by reading the Write command sequencer state or by using the interrupt functions. Memory blocks can be transferred by using the DMA function. Write errors are indicated by 2 error interrupt flags.

● Idle Interrupt IDLINT (Mainly for programming by DMA)

The DFISA/DFISB:IDLINT flag is set when the Write command sequencer is ready to receive new write data. This is the case after enabling the Write command sequencer (setting DFCA/DFCB:CSWE,WE to "1") and when the command sequencer finishes a write operation. The interrupt and DMA function of this flag is enabled by the corresponding DFICA/DFICB:IDLINTE bit.
This flag can be used to initiate a DMA transfer to the Flash. The DMA and interrupt module must be configured before activating the Write command sequencer. Setting DFCA/DFCB:CSWE or DFICA/DFICB:IDLINTE to "1" will then assert the first interrupt which starts the DMA operation.
The IDLINT flag can be cleared by CPU writing and by the DMA clear function.

● Finish Interrupt FININT (Mainly for programming by CPU)

The DFISA/DFISB:FININT flag is set after a successful termination of a write command. It can be used to inform the CPU about the result of the started write command and that the Write command sequencer is ready to program the next data. The interrupt function of this flag is enabled by the corresponding DFICA/DFISB:FININTE bit.
The FININT flag can be cleared by CPU writing and by the DMA clear function.

● Error Interrupts (Write error and Ignored Write command)

An unsuccessful termination of a write operation when using the Write command sequencer is denoted:

- Write error flag DFICA/DFISB:ERINT: This error flag is set when the written data is not correct.
- Write busy Interrupt flag DFISA/DFISB:WBINT. This error flag is set when CPU writes command while the flash memory is busy.

The interrupt function of these 2 flags is enabled by the corresponding DFICx:ERINTE, WBINTE bit. A DMA Stop request is also asserted in case of an error interrupt.

6.4. Erasing All Data (Chip Erase)

This section describes the procedure for issuing the Chip Erase command to erase all data in a flash memory module.

■ Erasing All Data in the Flash Memory Module (Chip erase)

All data can be erased from a flash memory module by sending the Chip Erase command in the command sequence table (see Table 4-1 in "4. Starting the Flash Memory Automatic Algorithm") continuously to any sector in the target flash memory module.

The Chip Erase command is executed in six bus operations. When writing of the sixth cycle is completed, the chip erase operation is started. For chip erase, the user does not need to write 0 to the flash memory before erasing. During execution of the automatic erase algorithm, the flash memory automatically writes 0 for verification before all of the cells are erased (preprogramming).

6.5. Erasing Optional Data (Sector erase)

This section describes the procedure for issuing the Sector Erase command to erase optional data (sector erase) in the flash memory. Individual sectors can be erased. Multiple sectors can also be specified at one time.

■ Starting the Sector Erase Automatic Algorithm

Optional sectors in the flash memory can be erased by sending the Sector Erase command in the command sequence table (see Table 4-1 in "4. Starting the Flash Memory Automatic Algorithm") continuously to the target sector in the flash memory.

● Specifying Sectors

A Sector erase is initiated by submitting the sector erase command (six write operations) to the target sector. After submitting the last command (30_H) to an even address in the target sector, the sector erase wait time is applied for approximately 35 to 70µs. To erase multiple sectors, write the erase code 30_H (sixth cycle of the command sequence) to the next target sector within this wait time. The first 5 cycles of the sector erase command do not have to be written in this case.

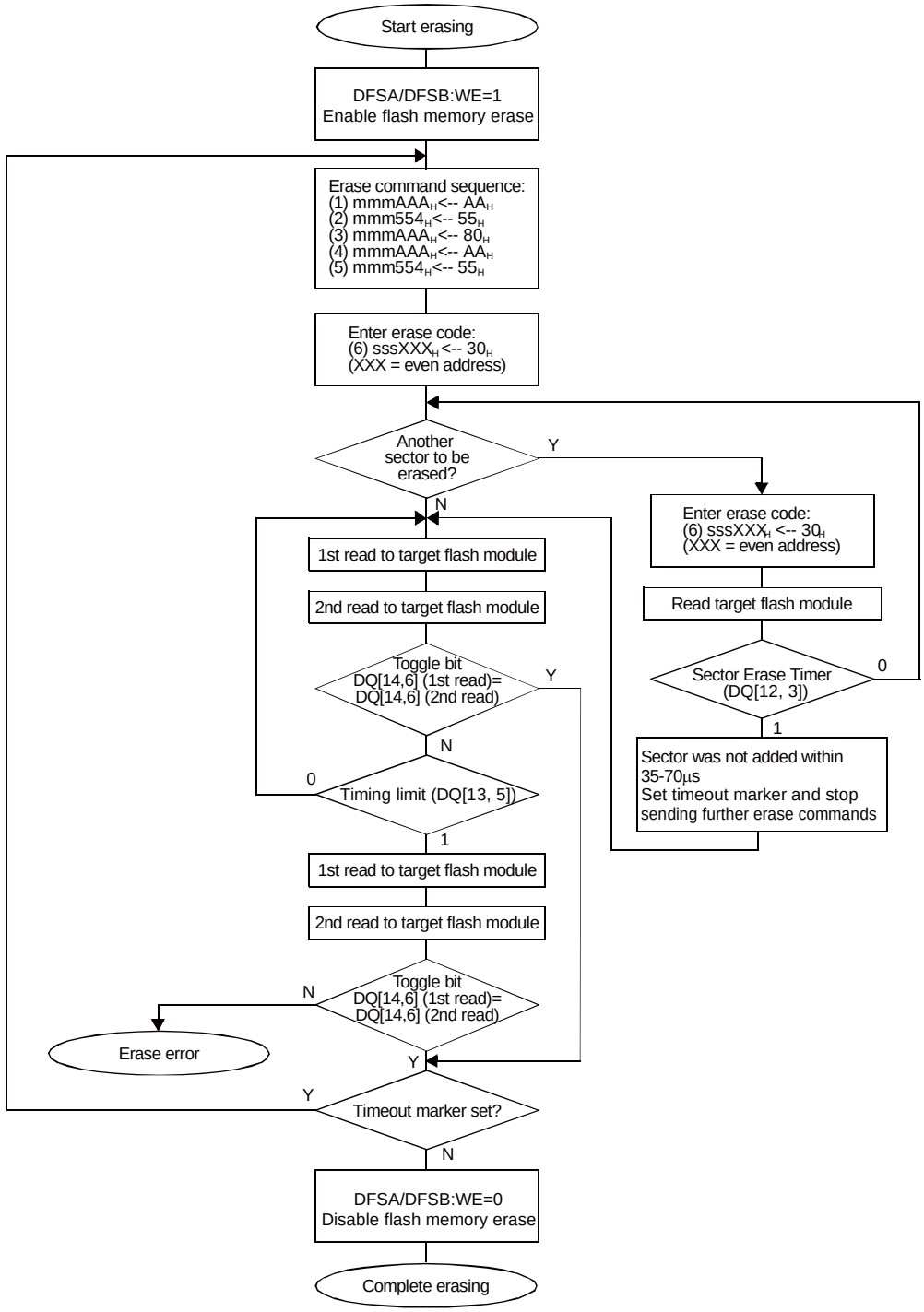
● Notes on Specifying Multiple Sectors

Erase is started when the sector erase wait period of 35 to 70µs terminates after the first sector erase code (30_H) has been written. Each writing of a sector erase code does not restart the sector erase wait period. The sector erase timer flag DQ3 must be checked after submitting each erase code to make sure it has been accepted.

■ Example for Erasing Sectors in the Flash Memory

The hardware sequence flags (see "and Section 5. Confirming the Automatic Algorithm Execution State") can be used to determine the state of the automatic algorithm in the flash memory. Figure 6-2 shows an example where the toggle bit flag (DQ[14,6]) is used to confirm that erasing has terminated.
For example to safely identify the Erase error state, the DQ[14,6] toggle bits must be checked again after reading DQ[13,5]=1.

Figure 6-2 Example of the Flash Memory Sector Erase Procedure



Note:

The address "mmm" must point to the flash memory module on which the command is meant to operate.
The address "sss" must point to the flash memory sector on which the command is meant to operate.

6.6. Suspending Sector Erase

This section describes the procedure for issuing the Sector Erase Suspend command to suspend erasing of flash memory sectors. In this state, data can be read from or written to sectors that are not specified to be erased.

■ Suspending Erasing of Flash Memory Sectors

Erasing of flash memory sectors can be suspended by sending the Sector Erase Suspend command ($B0_H$) to an even address in the target flash memory sector. Submitting this command suspends the sector erase operation being executed. In this suspend state, it is possible to read data from or to write data to sectors that are not specified to be erased. Further erase operations are forbidden.

The Sector Erase Suspend command is valid only while the sector erase operation is in progress or while the sector erase wait time is being applied. The command will be ignored during chip erase or write operations. If a Sector Erase Suspend command is issued during sector erase suspend, the command will also be ignored.

Entering the Sector Erase Suspend command during the sector erase wait period will immediately terminate sector erase wait and suspend the requested erase operation. While the sector erase is in progress, it takes a maximum period of 35μs from submitting the Sector Erase Suspend command until the flash changes to the suspend state.

The status register (DFSA/DFSB:ESPS) indicates that the suspend command was submitted to the Flash memory. Check the DFSA/DFSB:RDY or hardware status flag to know whether the Flash is in suspend state or not.

● Notes on using the Sector Erase Suspend Function of the Flash:

After restarting a sector erase, a minimum wait time of 2ms must be applied before submitting a sector erase suspend command.

6.7. Restarting Sector Erase

This section describes the procedure for issuing the Sector Erase Restart command to restart suspended erasing of flash memory sectors.

■ Restarting Erasing of Flash Memory Sectors

Suspended erasing of flash memory sectors can be restarted by sending the Sector Erase Restart command (30_H) to an even address in the target flash memory module. If a Sector Erase Restart command is issued during sector erase, the command will be ignored.

The Sector Erase Restart command is used to restart erasing of sectors from the sector erase suspend state set by using the Sector Erase Suspend command.

6.8. Protecting Sectors from being Erased/Written to

This section describes the procedure to protect sectors of the Flash from being unintentionally erased or written to.

■ Function of the Sector Write Protection

Individual sectors of the Flash can be protected from being erased (by sector erase command) or from being written (by write command). This can be configured by programming the Flash Memory Write Control registers. For a description of these registers, please see "Section 7.5 Dual Operation Flash Write Access Control Register 0-1 (DFWC0A, DFWC0B, DFWC1A, DFWC1B)". This feature can be used to prevent the application from an unintended erase or rewrite of specified Flash sectors. It is still possible to write/erase the flash memory in Parallel Flash programming mode or in Serial Communication mode.
An attempt to send a command sequence to a write protected sector is cancelled. This includes all write accesses of any command sequence. Hence for submitting a command sequence to the Flash, only sectors which are not write protected must be addressed.
If one sector of a flash module is write protected, then it is not possible to perform a chip erase operation. All write control bits of not existing sectors should also be set to "1" to allow chip erase.
The sector write protection has no effect in Parallel Flash programming mode.

■ Activating the Write Protection by User Program

The user program can configure the sector write protection by direct programming of the Dual operation Flash Write Access Control register (DFWC0A,DFWC0B, DFWC1A,DFWC1B).

■ Activating the Write Protection without User Program Interaction

Individual sectors of the flash memory can be protected from being erased or written to by filling in data into dedicated locations of the Flash memory (ROM configuration blocks RCBA/RCBB). No user program interaction is needed to activate the write protection in this case.
The write protection for sectors of Flash A is defined by the Flash Write Protection Activation Marker A (FWPAMA1/0) and the Flash Write Protection Sector Markers A (FWPSMA0, FWPSMA4) stored in RCBA.
The write protection for sectors of Flash B is defined by the Flash Write Protection Activation Marker B (FWPAMB1/0) and the Flash Write Protection Sector Markers B (FWPSMB0, FWPSMB3) stored in RCBB.
After any reset and before user program execution the following is executed (in Internal Vector modes):

Flash A:
When the content of {FWPAMA1, FWPAMA0} = $292D3A7B_H$, then the content of FWPSMA0/FWPSMA4 is transferred to DFWC0A/DFWC1A for all existing sectors of Flash A.
For any other value of {FWPAMA1, FWPAMA0}, the content of FWPSMA0/FWPSMA4 is ignored and no transfer to the DFWC0A/DFWC1A registers is performed.

Flash B (only devices with Flash B):
When the content of {FWPAMB1, FWPAMB0} = $292D3A7B_H$, then the content of FWPSMB0/FWPSMB3 is transferred to DFWC0B/DFWC1B for all existing sectors of Flash B.
For any other value of {FWPAMB1, FWPAMB0}, the content of FWPSMB0/FWPSMB3 is ignored and no transfer to the DFWC0B/DFWC1B registers is performed.

As long as the sector containing the flash write protection markers is not protected against write/erase. it is possible to modify the content of the markers by writing to this marker or by erasing the Flash memory sector containing the marker. The new content is activated by asserting any reset.
After write protection for a set of sectors is activated by Flash Write Protection Markers, the user program can still add individual sectors to the set of write protected sectors by setting the corresponding bits in the Dual operation Flash Write Access Control register 0-1 (DFWC0A, DFWC0B, DFWC1A, DFWC1B) to "0".
Correspondences to the DFWC* registers are below

- FWPSMA0 -> DFWC0A
- FWPSMA4 -> DFWC1A
- FWPSMB0 -> DFWC0B
- FWPSMB3 -> DFWC1B

In Serial communication mode, the configuration data is not automatically transferred to the DFWC registers.
If the temporary UART scan is enabled in Internal Vector mode, then the configuration data is transferred to the DFWC registers after leaving the serial communication.

Figure 6-3 Configuration of the Flash Write Protection Activation Marker (FWPAMA0/1, FWPAMB0/1)

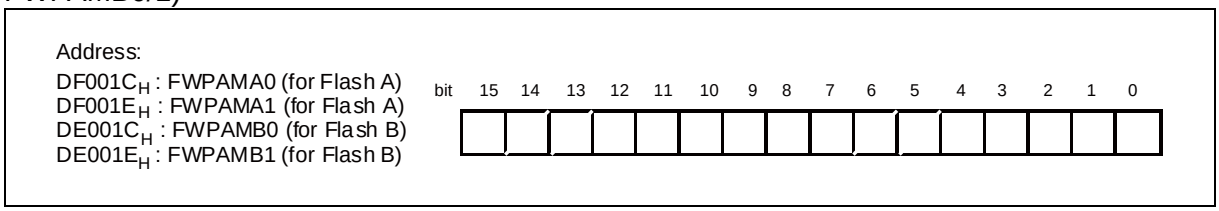
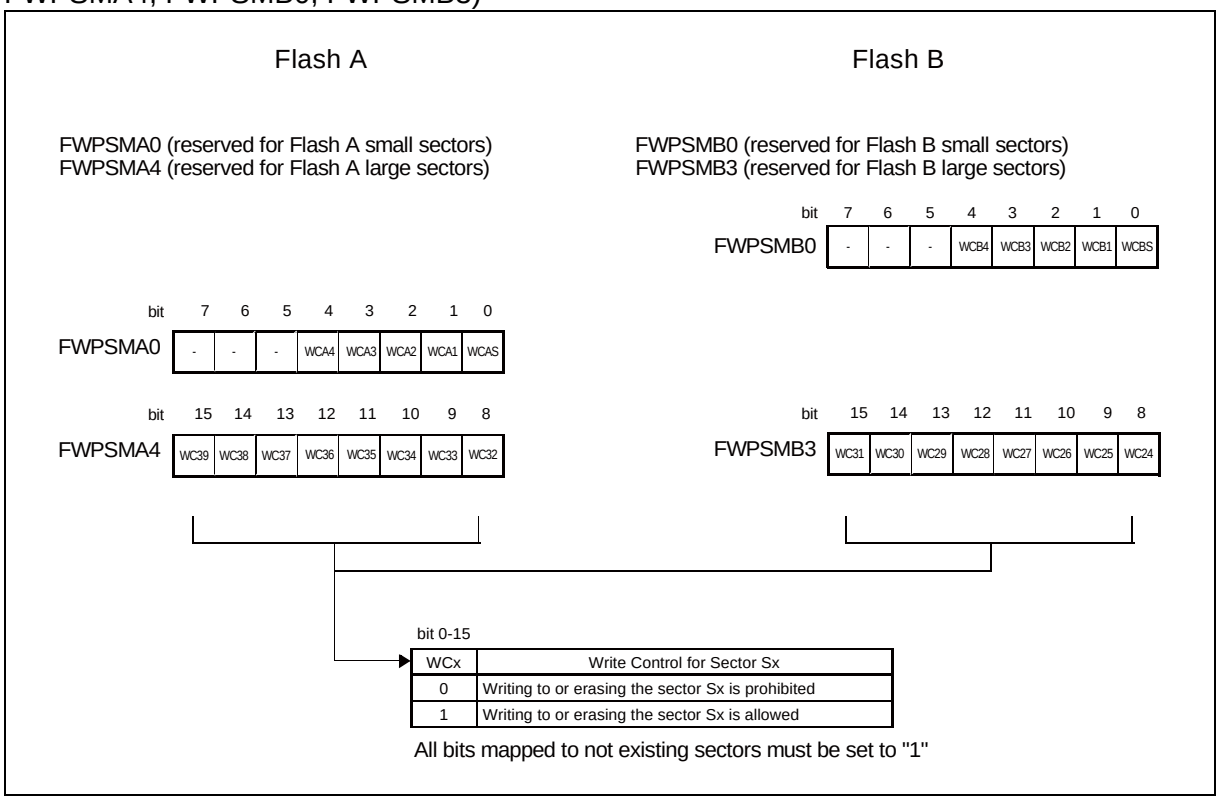


Figure 6-4 Configuration of the Flash Write Protection Sector Markers (FWPSMA0, FWPSMA4, FWPSMB0, FWPSMB3)



7. Registers

This chapter describes the control/status registers of the Dual operation Flash which are needed to control the flash memory erase/write processes.

■ Dual Operation Flash Registers

The Flash A memory is controlled by the following registers:

- Dual operation Flash Configuration Register (DFCA)
- Dual operation Flash Interrupt Control Register (DFICA)
- Dual operation Flash Status Register (DFSA)
- Dual operation Flash Interrupt Status Register (DFISA)

The Flash B memory is controlled by the following registers:

- Dual operation Flash Configuration Register (DFCB)
- Dual operation Flash Interrupt Control Register (DFICB)
- Dual operation Flash Status Register (DFSB)
- Dual operation Flash Interrupt Status Register (DFISB)

The permissions to erase/write a sector of the Flash memory are stored in the following registers:
Dual operation Flash Write Access Control register 0-1 (DFWC0A, DFWC0B, DFWC1A, DFWC1B)

■ List of Dual Operation Flash Registers

Abbreviated Register Name	Register Name	Reference
DFCA,DFCB	Dual operation Flash Configuration Register	See 7.1
DFICA,DFICB	Dual operation Flash Interrupt Control Register	See 7.2
DFSA,DFSB	Dual operation Flash Status Register	See 7.3
DFISA,DFISB	Dual operation Flash Interrupt Status Register	See 7.4
DFWC0A, DFWC0B, DFWC1A, DFWC1B	Dual operation Flash Write Access Control Register 0-1	See 7.5

7.1. Dual operation Flash Configuration Register (DFCA, DFCB)

Dual operation Flash Configuration Register (DFCA for Flash A memory, DFCB for Flash B memory) is used to configure the access mode and the access wait of the flash memory.
The DFCA (DFCB) register can not be written when CS is not IDLE state.
When the DFCA (DFCB) register is written, Bus cannot be accessed for 4 cycles.

■ Dual Operation Flash Configuration Register (DFCA, DFCB)

DFCA,DFCB

bit	7	6	5	4	3	2	1	0
	-	-	-	-	-	TMG	CSWE	WE
Attribute	-	-	-	-	-	R/W	R/W	R/W
Initial value	X	X	X	X	X	0	0	0

[bit7 to bit3] -: Undefined

- These bits are not used and does not affect to configuration of Flash Memory Interface.
- The read value of these bits are undefined.
- Always "0" write to these bits.
- Read modify write operations to this register have no effect on these bits.

[bit2] TMG: Flash Timing Register

Bit	Description
0	1 cycle (0 wait), Read and Write access to the Flash takes 1 cycle
1	2 cycles (1 wait) , Read and Write access to the Flash takes 2 cycles

- These bits control wait cycle for Flash memory read/write operations.
- When the Command Sequencer is in IDLE state or disabled, this bit can be written.
- The initial value of TMG bit is "0".
- This bit is readable and writable.

Please see Figure 7-1.

[bit1] CSWE: Command Sequencer Write Enable

Bit	Description
0	Disabled Command Sequencer for write operation
1	Enable Command Sequencer for write operation

- This bit enables command sequencer for write operation.
- When Command Sequencer is in IDLE state or disabled, this bit can be written.
- When this bit is set to "1", command sequencer will generate an automatic write sequence and proper interrupt/flag for each Flash write access. WE bit must be also set to 1 to write the data to Flash by CS.
- When this bit is "0", command sequencer is disabled and the write sequence is provided to the Flash manually.
- For erase/suspend operation, this bit shall be set to "0" and complete sequence is provided manually.
- The initial value of this bit is "0". (Command Sequencer Write is disabled)

[bit0] WE: Write Enable

Bit	Description
0	Writing to the Flash is disabled
1	Writing to the Flash is enabled

- This bit enables/disables writing to the Flash.
- When CS is in IDLE state or disabled, this bit can be written.
- CEX and WEX are forced to "1" during write access when this bit is "0". Hence no write/erase commands can be sent to the Flash.
- Setting this bit to "1" enables WEX, CEX. Write Command can be sent to Flash.
- The initial value of this bit is "0".

7.2. Dual Operation Flash Interrupt Control Register (DFICA, DFICB)

Dual operation Flash Interrupt Control Register (DFICA for Flash A memory, DFICB for Flash B memory) controls Interrupt functions of Flash-IF, this register affects both Direct mode functions and Command Sequencer functions.

■ Dual Operation Flash Interrupt Control Register (DFICA, DFICB)

DFICA,DFICB

bit	15	14	13	12	11	10	9	8
	-	ERINTE	FININTE	IDLINTE	WBINTE	WPSINTE	HANGINTE	RDYINTE
Attribute	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	X	0	0	0	0	0	0	0

[bit15] -: Undefined

- This bit is not used and does not affect to configuration of Flash Memory Interface.
- The read value of this bit is undefined.
- Always "0" write to this bit.
- Read modify write operations to this register have no effect on this bit.

[bit14] ERINTE: Error Interrupt Enable

Bit	Description
0	Write error Interrupt is disabled
1	Write error Interrupt is enabled

- This bit enables ERINT interrupts for Flash Memory at Command Sequencer mode.
- An interrupt is generated when this bit is set to "1" and ERINT flag gets set by error in write operation.
- When this bit is set to "0", write error interrupt can not be generated.
- The initial value of this bit is "0".
- This bit is readable and writable.

[bit13] FININTE: Finish Interrupt Enable

Bit	Description
0	Finish Interrupt is disabled
1	Finish Interrupt is enabled

- This bit enables finish interrupt (FININT) for Flash Memory at Command Sequencer mode.
- An interrupt is generated when this bit is set to "1" and FININT flag gets set by completion of successful write operation.
- When this bit is set to "0", finish interrupt can not be generated.
- The initial value of this bit is "0".
- This bit is readable and writable.

[bit12] IDLINTE: Idle Interrupt Enable

Bit	Description
0	Idle Interrupt is disabled
1	Idle Interrupt is enabled

- This bit enables idle interrupt (IDLINT) for Flash Memory at Command Sequencer mode.
- An interrupt is generated when this bit is set to "1" and IDLINT flag gets set by command sequencer idle state.
- When this bit is set to "0", Idle interrupt can not be generated.
- The initial value of this bit is "0".
- This bit is readable and writable.

[bit11] WBINTE: Write Busy Interrupt Enable

Bit	Description
0	Write busy Interrupt function is disabled
1	Write busy Interrupt function is enabled

- This bit permits the CPU to generate an interrupt when the write command is sent to the Flash memory during Flash can not accept the command.
- For detail, please refer the section of DFISA/DFISB:WBINT.
- WBINT interrupt is generated when this bit is set to "1".
- When this bit is set to "0", WBINT interrupt can not be generated.
- The initial value of the WBINTE bit is "0".
- This bit is readable and writable.

[bit10] WPSINTE: Write to Protected Sector Interrupt Enable

Bit	Description
0	Write to protected Sector Interrupt function is disabled
1	Write to protected Sector Interrupt function is enabled

- This bit permits the CPU to generate an interrupt when the write command is sent to the Flash memory to protected sector.
- About sector protection, please refer the section of sector write protection.
- WPSINT interrupt is generated when this bit is set to "1".
- When this bit is set to "0", WPSINT interrupt can not be generated.
- The initial value of the WPSINTE bit is "0".
- This bit is readable and writable.

[bit9] HANGINTE: HANG Interrupt Enable

Bit	Description
0	Hang Interrupt function is disabled
1	Hang Interrupt function is enabled

- This bit permits the CPU to generate an interrupt when Flash asserts HANG signal during programming/erasing.
- HANG interrupt is generated when this bit is set to "1" and HANGINT flag gets set by rising edge of HANG signal.
- When this bit is set to "0", HANG interrupt can not be generated.
- The initial value of the HANGINTE bit is "0".
- This bit is readable and writable.

[bit8] RDYINTE: RDY Interrupt Enable

Bit	Description
0	Ready Interrupt function is disabled
1	Ready Interrupt function is enabled

- This bit permits the CPU to generate an interrupt when a write/erase command terminates.
- RDY interrupt is generated when this bit is set to "1" and RDYINT flag gets set by rising edge of RDY signal.
- When this bit is set to "0", RDY interrupt can not be generated.
- The initial value of the RDYINTE bit is "0".
- This bit is readable and writable.

7.3. Dual Operation Flash Status Register (DFSA, DFSB)

Dual operation Flash Status Register (DFSA for Flash A memory, DFSB for Flash B memory) shows the status of write operation. This register shows the status of both Direct mode and Command Sequencer mode.

■ Dual Operation Flash Status Register (DFSA, DFSB)

DFSA,DFSB

bit	7	6	5	4	3	2	1	0
	ST1	ST0	PGMS	SERS	ESPS	CERS	HANG	RDY
Attribute	R	R	R	R	R	R	R	R
Initial value	1	1	0	0	0	0	0	1

[bit7, bit6] ST1, ST0: Status Bits for CS

- Status bits indicate 4 states of write command sequencer.
- Meaning of status bits are below.

bit7	bit6	Description
0	0	CS is in idle state
0	1	Sending write command
1	0	Checking flash status (wait end of programming)
1	1	CS is disabled (CSWE = 0 and CS is in idle state)

[bit5] PGMS: Program Status

Bit	Description
0	Flash Memory is not under programming state
1	Flash Memory is under programming state

- This bit shows the status of the Flash PGMS output.
- Writing to this bit has no effect.
- A read value of "1" indicates that Flash Memory have accepted Program command. Any command can not be accepted in this state.
- A read value of "0" indicates that Flash Memory is not under Programming.

[bit4] SERS: Sector Erase Status

Bit	Description
0	Flash Memory is not under sector erase state
1	Flash Memory is under sector erase or suspend state

- This bit shows the status of the Flash SERS output.
- Writing to this bit has no effect.
- A read value of "1" indicates that Flash Memory have accepted the Sector Erase command. Only the suspend command is accepted in this state. This bit is also set to "1" during Erase Suspend Status is set to "1". Whether the Flash is erasing flash or in erase suspend can be indicated by this bit.
- A read value of "0" indicates that Flash Memory is not in Sector Erase operation.

[bit3] ESPS: Erase Suspend Status

Bit	Description
0	Flash Memory is not under erase suspend state
1	Flash Memory had accepted erase suspend command

- This bit shows the status of the Flash ESPS output.
- Writing to this bit has no effect.
- A read value of "1" indicates that Flash Memory have accepted the Erase Suspend command. The resume command or write program command is accepted in this state.
- ESPS doesn't indicate actual Erase suspend status. Check the RDY signal is set to "1" to know whether the Flash macro is in Erase suspend state or not. If RDY shows "1" and ESPS shows "1", Flash is in erase suspend state.
- A read value of "0" indicates that Flash Memory is not in Erase suspend state.

[bit2] CERS: Chip Erase Status

Bit	Description
0	Flash Memory is not under chip erase state
1	Flash Memory is under chip erase state

- This bit shows the status of the Flash CERS output.
- Writing to this bit has no effect.
- A read value of "1" indicates that Flash Memory have accepted Chip Erase command and doing Chip Erase. Any command is not accepted in this state.
- A read value of "0" indicates that Flash Memory is not under Chip Erasing.

[bit1] HANG: Flash HANG Status

Bit	Description
0	Flash Memory is operating normally
1	Flash Memory is in HANG state

- This bit shows the status of the Flash HANG output.
- Writing to this bit has no effect.
- A read value of "1" indicates that the Flash Memory is in HANG state. Only the read/reset command is accepted in this state.
- A read value of "0" indicates that the Flash Memory is operating normally. Any command can be written to the Flash.
- This register is set to "1" when Flash could not execute the command correctly.
  - The written bit was tried to change the bit from "0" to "1".
  - User repeats a lot of Sector Erase suspend/resume operations in a short period.
  - The Flash could not execute accidentally the sequence which was received.
- This register is set to "0" when a Reset command is sent to the Flash and the Flash accepts it. Then the Flash is ready again and the Flash accepts next read/write command. Hardware reset can be also reset the Flash to read/reset state,

[bit0] RDY: Flash RDY Status

Bit	Description
0	Write/erase ongoing
1	Flash is ready (no write/erase ongoing)

- This bit shows the status of the Flash RDY output.
- Writing to this bit has no effect.
- A read value of "0" indicates that a program/erase command is currently executed. Only the reset from HANG state and sector erase add/suspend commands are accepted in this state.
- A read value of "1" indicates that no program/erase command is currently executed. Any command can be written to the Flash.
- If DFSA/DFSB.HANG bit is asserted, RDY will also be set to "0" until the read/reset command is sent to the Flash.
- RDY and HANG bits shall always be cared for correct operation.
- This bit does not change immediately after the last write/erase command was sent. Please make sure whether the Flash is writing/erasing or not by reading this bit at least two cycles after the last write/erase command.

See Table 5-2 to more details about hardware sequence flags and status register functions.

7.4. Dual Operation Flash Interrupt Status Register (DFISA, DFISB)

Dual operation Flash Interrupt Status Register (DFISA for Flash A memory, DFISB for Flash B memory) shows interrupt status of the Flash memory.

■ Dual Operation Flash Interrupt Status Register (DFISA, DFISB)

DFISA,DFISB

bit	15	14	13	12	11	10	9	8
	-	ERINT	FININT	IDLINT	WBINT	WPSINT	HANGINT	RDYINT
Attribute	-	R/W0	R/W0	R/W0	R/W0	R/W0	R/W0	R/W0
Initial value	X	0	0	0	0	0	0	0

[bit15] -: Undefined

- This bit is not used and does not affect to configuration of Flash Memory Interface.
- The read value of this bit is undefined.
- Always "0" write to this bit.
- Read modify write operations to this register have no effect on this bit .

[bit14] ERINT: Command Sequencer Write Error Interrupt Flag

Bit	Description
0	CS is disabled or is writing or has succeeded to write
1	Writing the data to the Flash by CS has failed

- This bit is the write error interrupt flag of the Command Sequencer function.
- This bit is initialized to "0" by reset.
- It is set to "1" by command sequencer which could not write the data correctly to the Flash.
- The WERINT bit can be cleared by writing to "0". Writing to "1" has no effect.
- The read cycle of a read-modify-write access always returns "1".
- It can't be cleared by IIOC.
- Interrupt function can be enabled by ERINTE.
- Conditions for setting this bit are below.
  - The data was tried to write to write protected sector by CS.
  - Some incomplete commands were sent to the Flash and then the data was written by CS.
  - The data could not be written to the Flash accidentally.
  - Written data was tried to change the data bit of the Flash from "0" to "1".
- User can know the cause of ERROR by checking other interrupt flags.
  - WPSINT - CS couldn't write the data due to it was tried to write to protected sector.
  - HANGINT - CS couldn't write the data due to an incorrect data was tried to write or the flash can not write the data accidently.

[bit13] FININT: Finish Interrupt

Bit	Description
0	CS is disabled or is writing or has failed to write
1	Writing the data to Flash with CS has successfully ended

- This bit is the write finish interrupt flag of the Command Sequencer function.
- This bit is initialized to "0" by reset.
- It is set to "1" by command sequencer finishing a write command successfully.
- The FININT bit can be cleared by writing to "0" or IIOC. Writing to "1" has no effect.
- The read cycle of a read-modify-write access always returns "1".
- Interrupt function can be enabled by FININTE.

[bit12] IDLINT: Idle Interrupt

Bit	Description
0	CS is disabled or is writing the data to Flash
1	CS is ready for taking data to write to Flash

- This bit is the Idle interrupt flag of the Command Sequencer interrupt function.
- This bit is initialized to "0" by reset.
- It is set to "1" by command sequencer enabled or command sequencer returns to idle state.
- The IDLINT bit can be cleared by writing to "0" or IIOC. Writing to "1" has no effect.
- The read cycle of a read-modify-write access always returns "1".
- Interrupt function can be enabled by IDLINTE.

[bit11] WBINT: Write Busy Interrupt Flag

Bit	Description
0	Write command in busy is not detected.
1	Write command in busy is detected.

- This bit is the interrupt flag of the Write busy interrupt function.
- This bit is initialized to "0" by reset.
- It is set to "1" by a rising edge of the Write busy detection signal.
- Conditions for setting this bit are below.
  - Write access during CS write.
  - Any write command during Programming / Chip Erasing was sent.
  - A "not" erase suspend/resume/sector-add command during Sector Erasing was sent.
  - Write access right after the MCU resumes from standby modes (Stop, Timer, Sleep).
- Conditions for "not" setting this bit during write/erase operation are below.
  - Any write command during HANG was sent.
  - Any write command during DFSA/DFSB:ESPS is "1" was sent.
- The WBINT bit can be cleared by writing "0". Writing "1" has no effect.
- The read cycle of a read-modify-write access always return "1".

[bit10] WPSINT: Write to Protected Sector Interrupt Flag

Bit	Description
0	Write command to protected Sector isn't detected
1	Write command to protected Sector is detected.

- This bit is the interrupt flag of the Write to protected sector interrupt function.
- This bit is initialized to "0" by reset.
- It is set to "1" by a rising edge of the Write to Protected Sector detection signal.
- WPSINT is asserted when a write command is sent to protected sector.
- The WPSINT bit can be cleared by writing "0". Writing "1" has no effect.
- The read cycle of a read-modify-write access always return "1".

[bit9] HANGINT: HANG Interrupt Flag

Bit	Description
0	Flash is operating normally
1	Flash detects write/erase error during operation

- This bit is the interrupt flag of the Flash Hang state interrupt function.
- This bit is initialized to "0" by reset.
- It is set to "1" by a rising edge of the HANG signal.
- The HANGINT bit can be cleared by writing "0". Writing "1" has no effect.
- The read cycle of a read-modify-write access always return "1".
- If HANGINT is set, the Read/Reset command shall be sent to the Flash to operate Flash normally. Or Hardware reset can be also reset the Flash.
- Conditions for setting this bit are below.
  - The written bit was tried to change "0" to "1".
  - Sector Erase suspend/resume commands were sent many times in a short time.
  - The Flash could not execute accidentally the sequence which was received.

[bit8] RDYINT: RDY Interrupt Flag

Bit	Description
0	No write/erase command termination is detected
1	Write/erase command has terminated

- This bit is the interrupt flag of the Flash RDY interrupt function.
- This bit is initialized to "0" by reset.
- It is set to "1" by a rising edge of the RDY signal.
- The RDYINT bit can be cleared by writing "0". Writing "1" has no effect.
- The read cycle of a read-modify-write access always return "1".

7.5. Dual Operation Flash Write Access Control Register 0-1 (DFWC0A, DFWC0B, DFWC1A, DFWC1B)

Dual operation Flash Write Access Control register 0-1(DFWC0A, DFC1A for Flash A memory, DFWC0B, DFWC1B for Flash B memory) controls write access of Flash Memory.

■ Dual Operation Flash Write Access Control Register 0-1 (DFWC0A, DFWC0B, DFWC1A, DFWC1B)

DFWC0B

bit	7	6	5	4	3	2	1	0
	-	-	-	SB4	SB3	SB2	SB1	SBS
Attribute	-	-	-	R/W	R/W	R/W	R/W	R/W
Initial value	X	X	X	0	0	0	0	0

DFWC0A

bit	15	14	13	12	11	10	9	8
	-	-	-	SA4	SA3	SA2	SA1	SAS
Attribute	-	-	-	R/W	R/W	R/W	R/W	R/W
Initial value	X	X	X	0	0	0	0	0

DFWC1B

bit	7	6	5	4	3	2	1	0
	SB31	SB30	SB29	SB28	SB27	SB26	SB25	SB24
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

DFWC1A

bit	15	14	13	12	11	10	9	8
	SA39	SA38	SA37	SA36	SA35	SA34	SA33	SA32
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

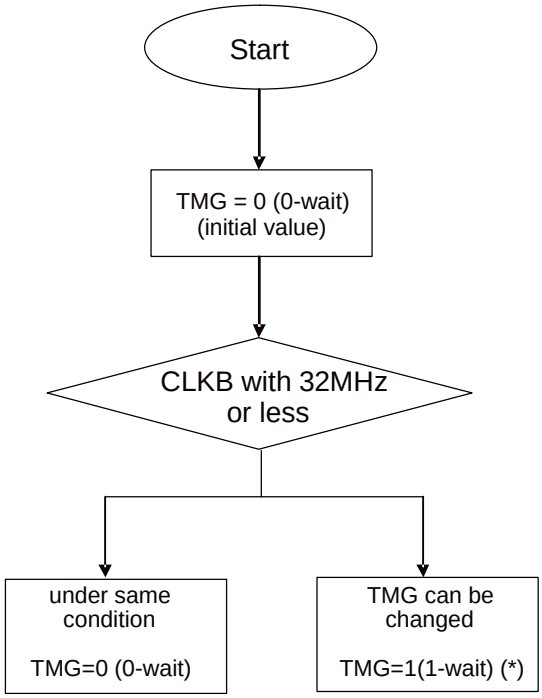
Note:

About the initial value for user program, please see “6.8 Protecting Sectors from being Erased/Written to” for details.

[bit15 to bit0] SA39 to SA32, SB31 to SB24, SA1 to SA4, SB1 to SB4, SAS, SBS: Write Access Control Bits

- These bits control the corresponding sector can be written and erased or not.
- After reset, these bits are reset to "0" (no programming/erasing is possible).
- Writing a bit to "1" enables programming/erasing of the corresponding sector. However setting bits to "1" is possible only when first write access to the register after reset.
- Writing a bit back to "0" disables programming/erasing again. A bit can be written to "0" at any time.
- SAS, SBS are for security sector of Flash.
- SB* is prepared for Flash B. Only when the Flash B exists, these bits are available.

Figure 7-1 Relationship between TMG bit Values and Flash Memory Access Cycles



Note:

* When CLKB≦32MHz, use with TMG = 0(0-wait) is recommended.

- 32 MHz 0 wait <- for best performance or when write to Flash

When using it with CLKB≦32MHz and 1wait:

Time from the command execution to the Flash to the status output might become long.
8×CLKB (Min.) to 64×CLKB (Max.)

This time is different depending on the access situation to the Flash.
It explains concretely.
If the access to the Flash is executed after transmitting the command to the Flash,
it become 8×CLKB(Min.).
However, if the access isn't executed, it become 64×CLKB(Max.).

8. Notes on using

This section contains notes on using flash memory.

■ Notes on using Flash Memory

● Input of a CPU Reset (RST)

A CPU reset (Power reset, external reset, software reset, watchdog reset or clock stop reset) asserts the flash hardware reset. Such a reset assertion during flash writing causes the data being written to be undefined. Resetting the device once execution of sector erase has begun will corrupt the data in the sector. In that case, restart the erase on this sector and allow it to complete.

● Program Access to Flash Memory

When the automatic algorithm is operating in one bank, read data access to this bank is disabled.
Hence, make sure that the CPU is not executing code from a bank of the flash memory that is erased/written. If one bank is to be erase/written, make sure that the code is executed only from the other bank or Flash B (if available) or from RAM.
For the same reason make sure that the table base register (TBR) is not pointing to an interrupt vector table located in a bank of the flash memory to be erased or written to. Program TBR to point to an interrupt vector table shall be in the other bank or other flash memory (if available) or in RAM, or disable interrupts completely before stating the automatic algorithm.

● DMA

DMA can be used to serve write interrupts.

9. Programming example

This section presents a Flash memory programming example.

■ Programming Example1

```
;===================================================================
; THIS SAMPLE CODE IS PROVIDED AS IS AND IS SUBJECT TO ALTERATIONS.
; FUJITSU SEMICONDUCTOR ACCEPTS NO RESPONSIBILITY OR LIABILITY
; FOR ANY ERRORS OR ELIGIBILITY FOR ANY PURPOSES.
;
; (C) FUJITSU SEMICONDUCTOR 2010
;===================================================================
;
; This program erases Flash A sector SA1(DF2000h) and programs 0xAA55
; to SA1. Afterwards the results are output on port P00 as follows:
; Write and Erase is executed from the program on RAM which
; is transferred from FLASH .
;
    ; Bit# 0 1 2 3 4 5 6 7
    ;      ---------------
    ;      | | |   | | |
    ;      | | |   | | |
    ;      | | |   | | +---- EERR: Erase time out error
    ;      | | |   | +------ EOK1: Successfully erased (with "time out warning")
    ;      | | |   +-------- EOK0: Successfully erased
    ;      | | +------------ PERR: Program time out error
    ;      | +-------------- POK1: Successfully programmed (with "time out warning")
    ;      +---------------- POK0: Successfully programmed

    .PROGRAM FLASH_W_E
    .TITLE FLASH_WRITE_ERASE

        ; Defines
            #set POK0 0x0001
            #set POK1 0x0002
            #set PERR 0x0004
            #set EOK0 0x0010
            #set EOK1 0x0020
            #set EERR 0x0040
            #set DQ3  0x0008                    ; sector erase start
            #set DQ5  0x0020                    ; time out bit
            #set DQ7  0x0080                    ; data polling bit

    .SECTION RESVECT, CONST, LOCATE=0xFFFFDC
    .DATA.E START
    .SECTION BOOT_SELECT, CONST, LOCATE=0xDF0030
    .DATA.L 0xFFFFFFFF


        ; I/O
            PDR00:    .EQU 0x0000               ; Port data register 00
            DDR00:    .EQU 0x0430               ; Port direction register 00

            DFWC0A:   .EQU 0x03F9               ; Sector SAS, SA1 - SA4
            DFWC1A:   .EQU 0x03FD               ; Sector SA39 - SA32

            DFCA:     .EQU 0x03E8               ; Flash A Configration Register
            DFSA:     .EQU 0x03EA               ; Flash A Status Register
            DFICA:    .EQU 0x03E9               ; Flash A Interrupt Configuration Register
            DFISA:    .EQU 0x03EB               ; Flash A Interrupt Status  Register

            CKSR:     .EQU 0x0401               ; Clock select
            CKSSR:    .EQU 0x0402               ; Clock stabilization control
            CKMR:     .EQU 0x0403               ; Clock stabilization status
            CKFCR:    .EQU 0x0404               ; Clock frequency control
```

```
            PLLCR:    .EQU 0x0406               ; PLL settings
        ; RAM
            RAMSTART: .EQU 0x7000               ; Start address of RAM code
        ; FLASH
            SA1:      .EQU 0x00DF2000           ; SA1 data sector
            SA1_554:  .EQU 0x00DF2554           ; Flash sequence address 1
            SA1_AAA:  .EQU 0x00DF2AAA           ; Flash sequence address 2

    ;=====================================================================
    ;
    ; Main program in Flash memory
            .SECTION CODE_FLASH, CODE, LOCATE=0xFE0000

_start:     NOP

START:      AND CCR, #0x80                      ; disable interrupts
            MOV RP, #0                          ; register bank #0
            MOV CKSSR, #0xF9                    ; set clock stabilization time 2^12/CLKMC
            MOVW PLLCR, #0x0007                 ; CPU: 32 MHz, CLKP2: 8 MHz for 4 MHz ext.
            MOVW CKFCR, #0x3001                 ; CLKB/1, CLKP1/1, CLKP2/4

            MOV DFCA, #0x00                      ; 0 wait states, CS Disable, Write Disable

            MOV CKSR, #0xFA                     ; PLL Clock Select
            MOVL A, #0x00006800                 ; top of system stack pointer
            MOVW SP, A
            SWAPW
            MOV SSB, A

            ; copy RAM code to RAM
            MOVW A, #RAMSTART                   ; start of RAM code
            MOVW A, #(0xFFFF & FLASH)           ; start of ROM code
            MOVW RW0, #SIZEOF(CODE_FLASH_RAM)   ; number of code bytes
            MOVS ADB, PCB

            MOV A, #0xFF                        ; init port
            MOV DDR00, A
            MOVN A, #0                          ; 0x00 to port 00
            MOV I:PDR00, A

            MOV A, #0x1F                        ; unlock SAS,SA1 - SA4
            MOV DFWC0A, A

PLLWAIT:    BBC CKMR:6, PLLWAIT                 ; wait for clock stabilization
            CALLP (RAMSTART)                    ; erase sector
            MOV PDR00, A                        ; status flag to port 00
            MOVW RW6, A                         ; save status flag
            CALLP (RAMSTART + 4)                ; write Data
            MOVW A, RW6                         ; restore status flag
            ORW A
            MOV PDR00, A                        ; show result

            LOOP: BRA LOOP                      ; endless loop


    ;====== FLASH ERASE/WRITE RAM CODE ================================
    ;
    ; RAM code functions
    .SECTION CODE_FLASH_RAM, CODE, LOCATE=0xFE9000
    ; jump table
    FLASH: JMPP (ERASE - 0xFE9000 + RAMSTART)
    JMPP (WRITE - 0xFE9000 + RAMSTART)
    ;====== SECTOR ERASE ==============================================
    ;
    ; erases sector SA1
    ; input: none
    ; output (in A): EOK0 = successfully erased
    ; EOK1 = successfully erased with "pre-time-out"
```

```
    ; EERR = time out error

ERASE:      SETB DFCA:  0   ; Set Write Enable

            ; Sector erase sequence
            MOVL A,#SA1_AAA
            MOVL RL0, A
            MOV A, #0xAA ; *0xDF2AAA = 0xAA
            MOVW @RL0, A
            MOVL A,#SA1_554
            MOVL RL0, A
            MOV A, #0x55 ; *0xDF2554 = 0x55
            MOVW @RL0, A
            MOVL A,#SA1_AAA
            MOVL RL0, A
            MOV A, #0x80 ; *0xDF2AAA = 0x80
            MOVW @RL0, A
            MOVL A,#SA1_AAA
            MOVL RL0, A
            MOV A, #0xAA ; *0xDF2AAA = 0xAA
            MOVW @RL0, A
            MOVL A,#SA1_554
            MOVL RL0, A
            MOV A, #0x55 ; *0xDF2554 = 0x55
            MOVW @RL0, A
            MOVL A,#SA1
            MOVL RL0, A
            MOV A, #0x30 ; *0xDF2000 = 0x30
            MOVW @RL0, A

            ; Wait for sector erase start
E_DQ3:      MOVW A, @RL0+0                  ; read sector state
            MOVN A, #DQ3                    ; Till DQ3=1, wait for starting sector erase
            ANDW A
            BEQ  E_DQ3                      ;

            ; Data polling algorithm
            MOVN A, #0                      ; status flag in RW2
            MOVW RW2, A                     ; initialize RW2

E_LOOP:     MOVW A, @RL0+0                  ; read sector state
            ANDW A, #DQ7                    ; data polling
            CWBNE A, #DQ7, E_1

            MOV A, #EOK0                    ; successful erased
            MOVW RW2, A

E_1:        MOVW A, @RL0+0                  ; read sector state
            ANDW A, #DQ5                    ; time out?
            CWBNE A, #DQ5, E_2

            MOVW A, @RL0+0                  ; read sector state
            ANDW A, #DQ7                    ; data polling
            CWBNE A, #DQ7, E_ERR            ; DQ5=1,DQ7=0 then the HANG & erase failed

            MOV A, #EOK1                    ; successful erased
            MOVW RW2, A                     ; DQ5=1,DQ7=1 then the HANG but erased.
            BRA E_2

E_ERR:      MOV A, #EERR                    ; time out error
            MOVW RW2, A

E_2:        MOVW A, RW2                     ; loop till getting the result
            BZ E_LOOP

            CLRB DFCA:0                     ; reset write enable

            MOVW A, RW2                     ; status flag in A
```

```
RAMEND:     RETP

    ;====== FLASH WRITE ==========================================
    ;
    ; writes 0xAA55 to start of sector SA1 (0xDF2000)
    ; input: none
    ; output (in A): POK0 = successfully written
    ; POK1 = successfully written with "pre-time-out"
    ; PERR = time out error

WRITE:      SETB DFCA:0     ;

            ; Flash write sequence
            MOVL A,#SA1_AAA
            MOVL RL0, A
            MOV A, #0xAA ; *0xDF2AAA = 0xAA
            MOVW @RL0, A
            MOVL A,#SA1_554
            MOVL RL0, A
            MOV A, #0x55 ; *0xDF2554 = 0x55
            MOVW @RL0, A
            MOVL A,#SA1_AAA
            MOVL RL0, A
            MOV A, #0xA0 ; *0xDF2AAA = 0xA0
            MOVW @RL0, A
            MOVL A, #SA1
            MOVL RL0, A
            MOVW A, #0xAA55                     ; *0xDF2000 = dummy data
            MOVW @RL0, A

            ; Data polling algorithm
            MOVN A, #0                          ; status flag in RW2
            MOVW RW2, A                         ; initialize RW2

W_LOOP:     MOVW A, @RL0+0                      ; read Flash state
            MOVW RW4, #0xAA55                   ; trying to write data
            ANDW A, #DQ7                        ; data polling
            MOVW RW5, A
            MOVW A, RW4
            ANDW A, #DQ7                        ; check the writing status
            CMPW A, RW5
            BNZ W_1                             ; false then write success

            MOVW A, #POK0                       ; successful written
            MOVW RW2, A

W_1:        MOVW A, @RL0+0                      ; read Flash state
            ANDW A, #DQ5                        ; time out?
            CWBNE A, #DQ5, W_2

            MOVW A, @RL0+0                      ; read Flash state
            MOVW RW4, #0xAA55                   ; trying to write  data
            ANDW A, #DQ7                        ; data polling
            MOVW RW5, A
            MOVW A, RW4
            ANDW A, #DQ7                        ; check the writing status
            CMPW A, RW5
            BNZ W_ERR                           ; false then write success

            MOVW A, #POK1                       ; successful written
            MOVW RW2, A
            BRA W_2

W_ERR:     MOVW A, #PERR                       ; time out error
            MOVW RW2, A

W_2:        MOVW A, RW2                          ; loop till getting the result.
            BZ W_LOOP
```

```
        CLRB DFCA:0                     ; write disable

        MOVW A, RW2                     ; status flag in A
        RETP

; ===== SA1 DUMMY DATA ==================================
;
; SA1 filled with 4 0x00 bytes to confirm sector erase
        .SECTION SA1_DUMMY_DATA, CODE, LOCATE=0xDF2000
        .DATA.L 0x00000000
        .END START                      ; for debugging
```

■ Programming Example2

```
;=================================================================
; THIS SAMPLE CODE IS PROVIDED AS IS AND IS SUBJECT TO ALTERATIONS.
; FUJITSU SEMICONDUCTOR ACCEPTS NO RESPONSIBILITY OR LIABILITY
; FOR ANY ERRORS OR ELIGIBILITY FOR ANY PURPOSES.
;
; (C) FUJITSU SEMICONDUCTOR 2010
;=================================================================;
; This program erases Flash A sector SA1(DF2000h) and programs 0xAA55
; to SA1. Afterwards the results are output on port P00 as follows:
; Write and Erase is executed from the program on Flash directly.
; This way only can be done on DualOperation Flash or Independent two Flashes.
;

    ; Bit# 0 1 2 3 4 5 6 7
    ;      ---------------
    ;      | | |   | | |
    ;      | | |   | | |
    ;      | | |   | | +---- EERR: Erase time out error
    ;      | | |   | +------ EOK1: Successfully erased (with "time out warning")
    ;      | | |   +-------- EOK0: Successfully erased
    ;      | | +------------ PERR: Program time out error
    ;      | +-------------- POK1: Successfully programmed (with "time out warning")
    ;      +---------------- POK0: Successfully programmed

    .PROGRAM FLASH_W_E
    .TITLE FLASH_WRITE_ERASE

        ; Defines
          #set POK0 0x0001
          #set POK1 0x0002
          #set PERR 0x0004
          #set EOK0 0x0010
          #set EOK1 0x0020
          #set EERR 0x0040
          #set DQ3  0x0008                  ; sector erase start
          #set DQ5  0x0020                  ; time out bit
          #set DQ7  0x0080                  ; data polling bit

    .SECTION RESVECT, CONST, LOCATE=0xFFFFDC
    .DATA.E START
    .SECTION BOOT_SELECT, CONST, LOCATE=0xDF0030
    .DATA.L 0xFFFFFFFF


        ; I/O
          PDR00:    .EQU 0x0000           ; Port data register 00
          DDR00:    .EQU 0x0430           ; Port direction register 00

          DFWC0A:    .EQU 0x03F9          ; Sector SA5, SA1 - SA4
          DFWC1A:    .EQU 0x03FD          ; Sector SA39-32

           DFCA:     .EQU 0x03E8           ; Flash A Configuration Register
```

```
            DFSA:     .EQU 0x03EA              ; Flash A Status Register
            DFICA:    .EQU 0x03E9              ; Flash A Interrupt Configuration Register
            DFISA:    .EQU 0x03EB              ; Flash A Interrupt Status  Register

            CKSR:     .EQU 0x0401              ; Clock select
            CKSSR:    .EQU 0x0402              ; Clock stabilization control
            CKMR:     .EQU 0x0403              ; Clock stabilization status
            CKFCR:    .EQU 0x0404              ; Clock frequency control
            PLLCR:    .EQU 0x0406              ; PLL settings
        ; RAM
            RAMSTART: .EQU 0x7000              ; Start address of RAM code
        ; FLASH
            SA1:      .EQU 0x00DF2000          ; SA1 data sector
            SA1_554:  .EQU 0x00DF2554          ; Flash sequence address 1
            SA1_AAA:  .EQU 0x00DF2AAA          ; Flash sequence address 2

    ;====================================================================
    ;
    ; Main program in Flash memory
            .SECTION CODE_FLASH, CODE, LOCATE=0xFE0000

_start:     NOP

START:      AND CCR, #0x80                     ; disable interrupts
            MOV RP, #0                         ; register bank #0
            MOV CKSSR, #0xF9                   ; set clock stabilization time 2^12/CLKMC
            MOVW PLLCR, #0x0007                ; CPU: 32 MHz, CLKP2: 8 MHz for 4 MHz ext.
            MOVW CKFCR, #0x3001                ; CLKB/1, CLKP1/1, CLKP2/4

            MOV DFCA, #0x00                    ; 0 wait states, CS Disable, Write Disable

            MOV CKSR, #0xFA                            ; PLL Clock Select
            MOVL A, #0x00006800                ; top of system stack pointer
            MOVW SP, A
            SWAPW
            MOV SSB, A

            MOV A, #0xFF                       ; init port
            MOV DDR00, A
            MOVN A, #0                         ; 0x00 to port 00
            MOV I:PDR00, A

            MOV A, #0x1F                       ; unlock SAS,SA1 - SA4
            MOV DFWC1, A

PLLWAIT:    BBC CKMR:6, PLLWAIT                ; wait for clock stabilization
            CALLP (ERASE)                       ; erase sector
            MOV PDR00, A                       ; status flag to port 00
            MOVW RW6, A                        ; save status flag
            CALLP (WRITE)                              ; write Data
            MOVW A, RW6                        ; restore status flag
            ORW A
            MOV PDR00, A                       ; show result

            LOOP: BRA LOOP                     ; endless loop

    ;====== SECTOR ERASE ========================================
    ;
    ; erases sector SA1
    ; input: none
    ; output (in A): EOK0 = successfully erased
    ; EOK1 = successfully erased with "pre-time-out"
    ; EERR = time out error

ERASE:      SETB DFCA:  0   ; Set Write Enable

            ; Sector erase sequence
            MOVL A,#SA1_AAA
```

```
            MOVL RL0, A
            MOV A, #0xAA ; *0xDF2AAA = 0xAA
            MOVW @RL0, A
            MOVL A,#SA1_554
            MOVL RL0, A
            MOV A, #0x55 ; *0xDF2554 = 0x55
            MOVW @RL0, A
            MOVL A,#SA1_AAA
            MOVL RL0, A
            MOV A, #0x80 ; *0xDF2AAA = 0x80
            MOVW @RL0, A
            MOVL A,#SA1_AAA
            MOVL RL0, A
            MOV A, #0xAA ; *0xDF2AAA = 0xAA
            MOVW @RL0, A
            MOVL A,#SA1_554
            MOVL RL0, A
            MOV A, #0x55 ; *0xDF2554 = 0x55
            MOVW @RL0, A
            MOVL A,#SA1
            MOVL RL0, A
            MOV A, #0x30 ; *0xDF2000 = 0x30
            MOVW @RL0, A

            ; Wait for sector erase start
E_DQ3:      MOVW A, @RL0+0                  ; read sector state
            MOVN A, #DQ3                            ; Till DQ3=1, wait for starting sector erase
            ANDW A
            BEQ  E_DQ3                              ;

            ; Data polling algorithm
            MOVN A, #0                      ; status flag in RW2
            MOVW RW2, A                             ; initialize RW2

E_LOOP:     MOVW A, @RL0+0                  ; read sector state
            ANDW A, #DQ7                    ; data polling
            CWBNE A, #DQ7, E_1

            MOV A, #EOK0                    ; successful erased
            MOVW RW2, A

E_1:        MOVW A, @RL0+0                  ; read sector state
            ANDW A, #DQ5                    ; time out?
            CWBNE A, #DQ5, E_2

            MOVW A, @RL0+0                  ; read sector state
            ANDW A, #DQ7                    ; data polling
            CWBNE A, #DQ7, E_ERR                    ; DQ5=1,DQ7=0 then the HANG & erase failed

            MOV A, #EOK1                    ; successful erased
            MOVW RW2, A                             ; DQ5=1,DQ7=1 then the HANG but erased.
            BRA E_2

E_ERR:      MOV A, #EERR                    ; time out error
            MOVW RW2, A

E_2:        MOVW A, RW2                             ; loop till getting the result
            BZ E_LOOP

            CLRB DFCA:0                     ; reset write enable

            MOVW A, RW2                     ; status flag in A
RAMEND:     RETP

    ;====== FLASH WRITE ========================================
    ;
    ; writes 0xAA55 to start of sector SA1 (0xDF2000)
    ; input: none
```

```
        ; output (in A): POK0 = successfully written
        ; POK1 = successfully written with "pre-time-out"
        ; PERR = time out error

WRITE:      SETB DFCA:0     ;

            ; Flash write sequence
            MOVL A,#SA1_AAA
            MOVL RL0, A
            MOV A, #0xAA ; *0xDF2AAA = 0xAA
            MOVW @RL0, A
            MOVL A,#SA1_554
            MOVL RL0, A
            MOV A, #0x55 ; *0xDF2554 = 0x55
            MOVW @RL0, A
            MOVL A,#SA1_AAA
            MOVL RL0, A
            MOV A, #0xA0 ; *0xDF2AAA = 0xA0
            MOVW @RL0, A
            MOVL A, #SA1
            MOVL RL0, A
            MOVW A, #0xAA55                                     ; *0xDF2000 = dummy data
            MOVW @RL0, A

            ; Data polling algorithm
            MOVN A, #0                          ; status flag in RW2
            MOVW RW2, A                                 ; initialize RW2

W_LOOP:     MOVW A, @RL0+0                              ; read Flash state
            MOVW RW4, #0xAA55                       ; trying to write data
            ANDW A, #DQ7                                ; data polling
            MOVW RW5, A
            MOVW A, RW4
            ANDW A, #DQ7                                ; check the writing status
            CMPW A, RW5
            BNZ W_1                                             ; false then write success

            MOVW A, #POK0                                               ; successful written
            MOVW RW2, A

W_1:        MOVW A, @RL0+0                      ; read Flash state
            ANDW A, #DQ5                                ; time out?
            CWBNE A, #DQ5, W_2

            MOVW A, @RL0+0                      ; read Flash state
            MOVW RW4, #0xAA55               ; trying to write  data
            ANDW A, #DQ7                        ; data polling
            MOVW RW5, A
            MOVW A, RW4
            ANDW A, #DQ7                                ; check the writing status
            CMPW A, RW5
            BNZ W_ERR                                   ; false then write success

            MOVW A, #POK1                       ; successful written
            MOVW RW2, A
            BRA W_2

W_ERR:      MOVW A, #PERR                       ; time out error
            MOVW RW2, A

W_2:        MOVW A, RW2                                 ; loop till getting the result.
            BZ W_LOOP

            CLRB DFCA:0                         ; write disable

            MOVW A, RW2                         ; status flag in A
            RETP
```

```
        ; ===== SA1 DUMMY DATA ==================================
        ;
        ; SA1 filled with 4 0x00 bytes to confirm sector erase
              .SECTION SA1_DUMMY_DATA, CODE, LOCATE=0xDF2000
              .DATA.L 0x00000000
              .END START                          ; for debugging
```

■ Programming Example3

```
;=====================================================================
; THIS SAMPLE CODE IS PROVIDED AS IS AND IS SUBJECT TO ALTERATIONS.
; FUJITSU SEMICONDUCTOR ACCEPTS NO RESPONSIBILITY OR LIABILITY
; FOR ANY ERRORS OR ELIGIBILITY FOR ANY PURPOSES.
;
; (C) FUJITSU SEMICONDUCTOR 2010
;=====================================================================
; This program erases Flash A sector SA1 and programs 0xAA55
; to SA1. Afterwards the results are output on port P00 as follows:
; Write the data to Flash by Command Sequencer.
; Write and Erase is executed from the program on Flash directly.
; This way only can be done on DualOperation Flash or Independent two Flashes.

;
    ; Bit# 0 1 2 3 4 5 6 7
    ;      ---------------
    ;      | | |   | | |
    ;      | | |   | | |
    ;      | | |   | | +---- EERR : Erase time out error
    ;      | | |   | +------ EOK1 : Successfully erased (with "time out warning")
    ;      | | |   +-------- EOK0 : Successfully erased
    ;      | | +------------ PERR1: Program time out error
    ;      | +-------------- PERR0: Program to write protected sector error
    ;      +---------------- POK0 : Successfully programmed

    .PROGRAM FLASH_W_E
    .TITLE FLASH_WRITE_ERASE

        ; Defines
            #set POK0  0x0001
            #set PERR0 0x0002
            #set PERR1 0x0004
            #set EOK0  0x0010
            #set EOK1  0x0020
            #set EERR  0x0040
            #set DQ3   0x0008                   ; sector erase start
            #set DQ5   0x0020                   ; time out bit
            #set DQ7   0x0080                   ; data polling bit

    .SECTION RESVECT, CONST, LOCATE=0xFFFFDC
    .DATA.E START
    .SECTION BOOT_SELECT, CONST, LOCATE=0xDF0030
    .DATA.L 0xFFFFFFFF

        ; I/O
            PDR00:    .EQU 0x0000               ; Port data register 00
            DDR00:    .EQU 0x0430               ; Port direction register 00

            DFWC0A:   .EQU 0x03F9               ; Sector SAS, SA1 - SA4
            DFWC1A:   .EQU 0x03FD               ; Sector SA39 - SA32

            DFCA:     .EQU 0x03E8               ; Flash A Configuration Register
            DFSA:     .EQU 0x03EA               ; Flash A Status Register
            DFICA:    .EQU 0x03E9               ; Flash A Interrupt Configuration Register
            DFISA:    .EQU 0x03EB               ; Flash A Interrupt Status  Register

            CKSR:     .EQU 0x0401               ; Clock select
            CKSSR:    .EQU 0x0402               ; Clock stabilization control
            CKMR:     .EQU 0x0403               ; Clock stabilization status
```

```
            CKFCR:    .EQU 0x0404               ; Clock frequency control
            PLLCR:    .EQU 0x0406               ; PLL settings

        ; FLASH
            SA1:      .EQU 0x00DF2000           ; SA1 data sector
            SA1_554:  .EQU 0x00DF2554           ; Flash sequence address 1
            SA1_AAA:  .EQU 0x00DF2AAA           ; Flash sequence address 2

    ;=====================================================================
    ;
    ; Main program in Flash memory
            .SECTION CODE_FLASH, CODE, LOCATE=0xFE0000

_start:     NOP

START:      AND CCR, #0x80                      ; disable interrupts
            MOV RP, #0                          ; register bank #0
            MOV CKSSR, #0xF9                    ; set clock stabilization time 2^12/CLKMC
            MOVW PLLCR, #0x0006                 ; CPU: 28 MHz, CLKP2: 7.5 MHz for 4 MHz ext.
            MOVW CKFCR, #0x3001                 ; CLKB/1, CLKP1/1, CLKP2/4

            MOV DFCA, #0x00                      ; 0 wait states, CS Disable, Write Disable

            MOV CKSR, #0xFA         ; PLL Clock Select
            MOVL A, #0x00006800                 ; top of system stack pointer
            MOVW SP, A
            SWAPW
            MOV SSB, A

            MOV A, #0xFF                        ; init port
            MOV DDR00, A
            MOVN A, #0                          ; 0x00 to port 00
            MOV I:PDR00, A

            MOV A, #0x1F                        ; unlock SAS,SA1 - SA4
            MOV DFWC1, A

PLLWAIT:    BBC CKMR:6, PLLWAIT                 ; wait for clock stabilization

            EI;
//Enable interrupt

            ; WRITE BY CS
            MOV DFCA,#0x03      ; CSWE,WE=1 for CS write
            MOV DFICA,#0x60     ; ERINT,FININT enable

            ; CS Flash write
            MOVL A, #SA1
            MOVL RL0, A
            MOVW A, #0xAA55 ; *0xDF2000 = dummy data
            MOVW @RL0, A
LOOP:       BRA LOOP                            ; endless loop

//IRQ JUMP TABLE
            .SECTION IRQTABLE, CODE, LOCATE=0xFFFC00 //Please change the actual value
            .DATA.L 0x00FF8000

//IRQ HANDLER
            .SECTION FLASHINTHDL, CODE, LOCATE=0xFF8000
            DI;
//Disable Interrupt

// This handler only handle the CS FIN/ERINT.
// Please add the other cases for normal use.
            MOV R3, DFISA;                      ;get IRQ Flag
            MOV DFISA, #0x00                    ;clear DFISA
            MOV A, R3                           ;
            AND A, #0x40                        ;
```

```
            BNZ ERROR                       ;ERINT check
            MOV R3, A
            AND A, #0x20                    ;FININT check
            BZ  EXIT                                                ;Other Interrupts call this
handler.
            MOVW RW6,#POK0                  ;Store the result
            JMP EXIT

ERROR:      MOV R3,A
            AND A,#0x40                     ;fail by Write protect
            BZ ER1                          ;
            MOVW RW6,#PERR0

ER1:        MOV R3,A
            AND A,#0x20                     ;fail by HANG
            BZ EXIT                         ;
                        MOVW RW2,RW6;
                        ORW RW2,#PERR1
            MOVW RW6,RW2                                        ;Mix the error result

            MOV DFCA,#0x01                  ;CSWE disable
            MOVL A, #SA1
            MOVL RL0, A
            MOV A, #0xF0                    ;Reset Command
            MOVW @RL0, A

EXIT:       NOP                             ;Exit
```

CHAPTER: FLASH SECURITY

This chapter explains the functions and operation of the Flash security.

Management Code: 96F6FLASHSEC-E01.0

1. Overview

The Flash security is a feature to prevent that the content of the Flash memory is read-out in a way not intended in the application.

■ Flash Security Features

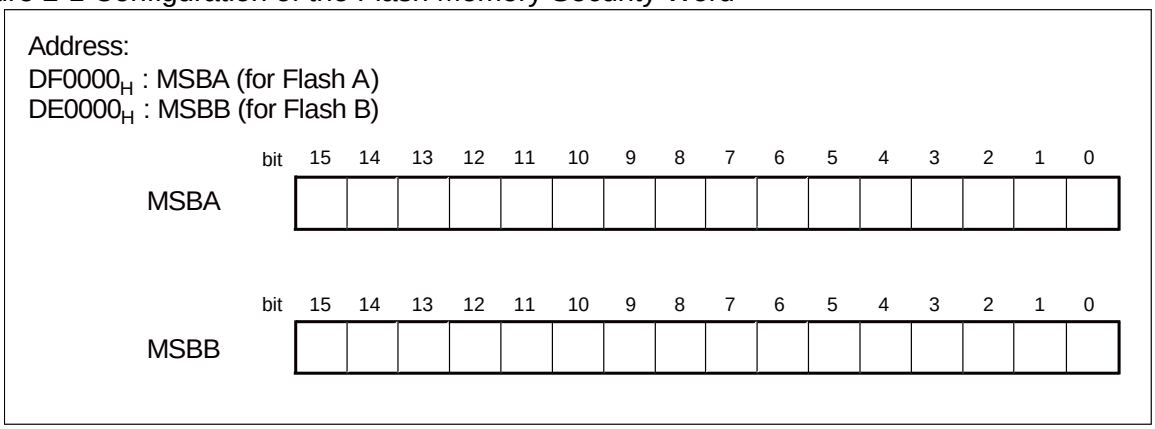
- Read protection feature to prevent reading out the content of the Flash memory when not intended in the application. Only internal user program that has been started in internal vector mode has access to the content of the Flash memory. The Flash memory cannot be read by:
  - external parallel flash programmer
  - serial communication mode
- In all modes the internal Flash can still be erased.
- This function can be canceled by entering 128-bit security key in serial communication mode, even if flash security has been configured. Security key can also be disabled permanently.

2. Usage

This section describes how the Flash security is enabled and disabled and how the content of the Flash memory is protected against unintended read-out.

■ Enabling the Flash Security

Figure 2-1 Configuration of the Flash Memory Security Word



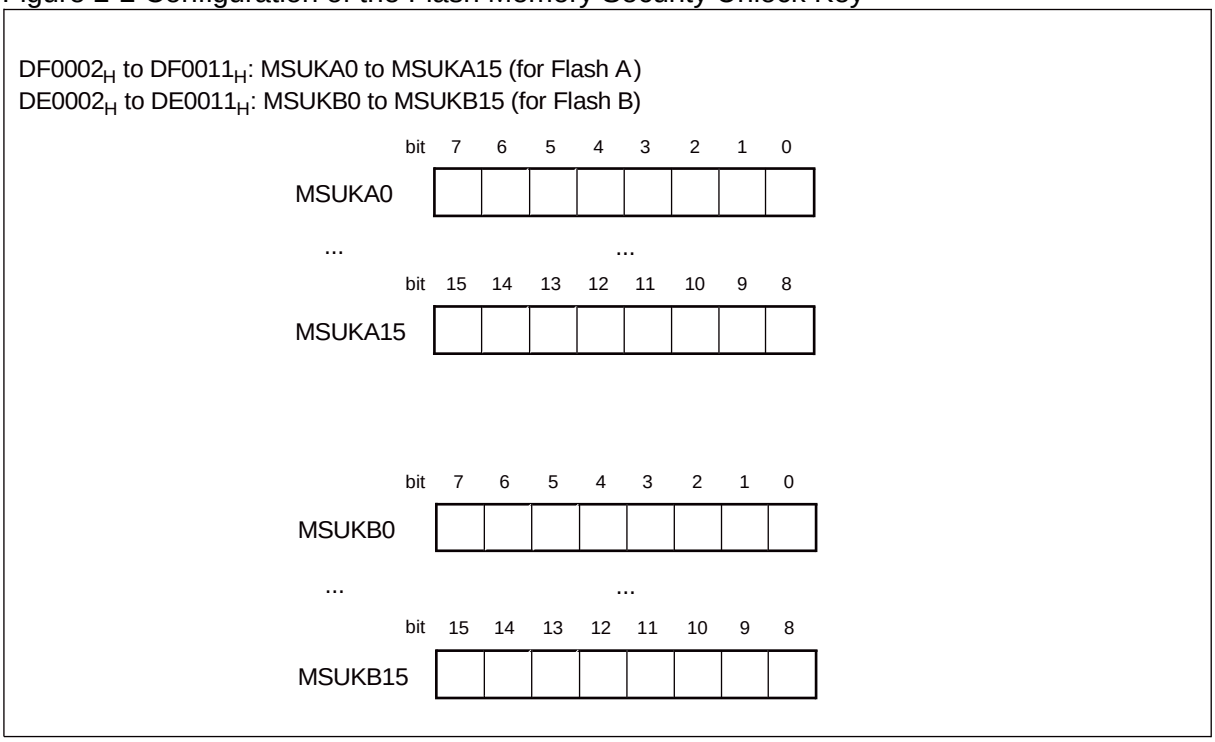
Flash memory security can be enabled for Flash A and Flash B by programming the flash memory security bytes MSBA and MSBB.
When the content of MSBA is 9999_H, then the Flash memory is protected. If the content of MSBA is set to $FFFF_H$ and the content of the unlocking key MSUKA0-MSUKA15 is set to all FF_H, then the Flash memory is not protected. No other value than 9999_H or $FFFF_H$ should be written to the MSBA register.
When the content of MSBB is 9999_H, then the according flash memory is protected. If the content of MSBB is set to $FFFF_H$ and the content of the unlocking key MSUKA0-MSUKA15 is set to all FF_H, then the according flash memory is not protected. No other value than 9999_H or $FFFF_H$ should not be written to the MSBB register.

Notes:

- After erasing the flash, the value of all bits in flash memory is "1". Hence, when the flash memory location of MSBA or MSBB is not written, the content of it is FF_H. In this case, the content of the flash memory is not protected.
- On devices with two flash macros, the same setting should be used for both Flash macros. Setting one flash macro to “protected” also disables access to the second Flash macro..

■ Unlocking the Flash Security

Figure 2-2 Configuration of the Flash Memory Security Unlock Key



A Flash memory which is protected (Memory security word is 9999_H) can still be read-out in serial communication mode, after an unlock key has been received by the USART which is used for serial communication mode. For details how to send the unlock key via the USART, please see application note MCU-AN-300213-E.
The unlock key is compared against the key stored in the unlock key bytes MSUKA0 to MSUKA15 (for Flash A memory). If the received key matches the key stored in the Flash memory, the memory can be read-out by subsequent serial programming commands.
When the unlock key value stored in MSUKA0 to MSUKA15/MUSKB0 to MSUKB15 is all zero, the protection of the corresponding Flash memory can not be unlocked.

Notes:

- When the unlock key values are all zero, the protection of the Flash memory can not be unlocked. Therefore, analysis of Flash contents cannot be performed. Feature to be used only if necessary.
- For unprotected device, the key stored in MSUKA0 to MSUKA15/MUSKB0 to MSUKB15 must be set to all FF_H.

■ Disabling Flash Security

On flash memory devices, the Flash security can be disabled as follows:

- When starting in internal vector mode, the application program can disable the flash security by
  - Clearing the value of the memory security word and unlocking key by erasing the sector containing this configuration. (SAS,SBS)
  - Clearing the value of the memory security word and unlocking key by erasing the complete flash memory with the "Chip erase" command.
- When starting in serial communication mode, the flash security can be disabled by
  - Clearing the value of the memory security word and unlocking key by erasing the sector containing this configuration after the flash security has been unlocked.
  - Clearing the value of the memory security word and unlocking key by erasing the complete flash memory with the "Chip erase" command.
- When starting in parallel flash programming mode the flash security can be disabled only by
  - Clearing the value of the memory security word and unlocking key by erasing the complete flash memory with the "Chip erase" command.

CHAPTER: EXAMPLES OF SERIAL PROGRAMMING CONNECTION

This chapter describes how to connect the MCU for serial programming of the Flash memory.

1. Basic Configuration of Serial Programming Connection
2. Example of connecting a PC for programming the Flash Microcontroller
3. Example of connecting a programming tool for programming the Flash Microcontroller

Management Code: 96F6SERIALPRG-E01.0

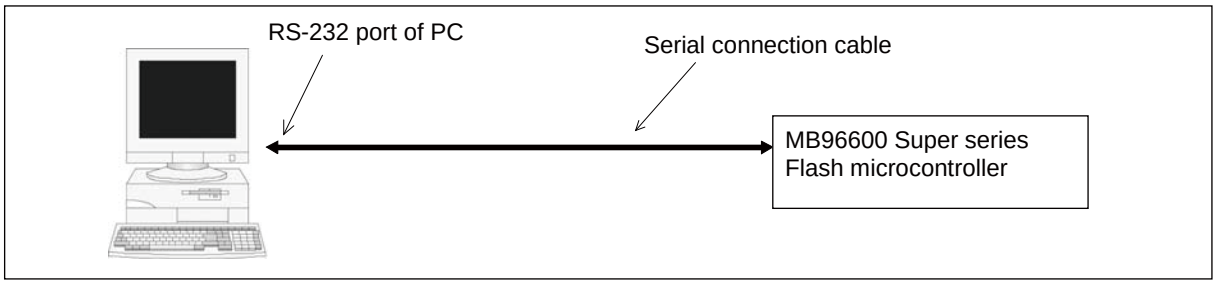
1. Basic Configuration of Serial Programming Connection

The MB96600 Super series Flash Microcontrollers support Flash ROM serial onboard programming. This section describes how to connect the Microcontroller to the programming equipment.

■ Basic Configuration of Serial Programming

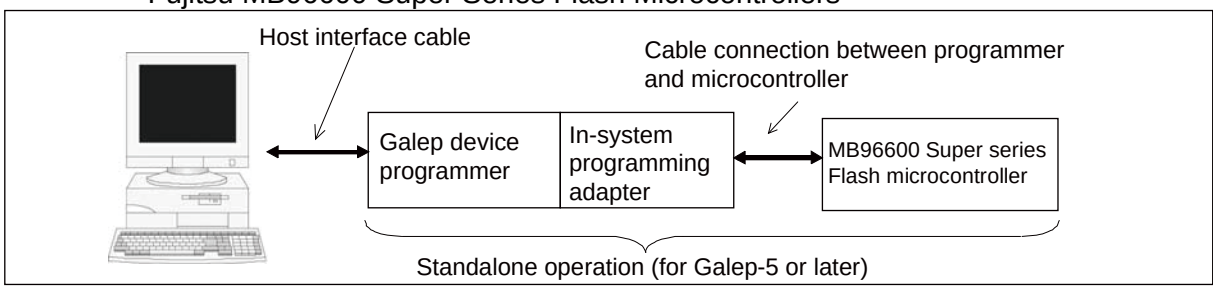
In a minimal system, the MB96600 Super series Flash microcontroller is connected via the USART in asynchronous mode to the RS-232 port of the PC.

Figure 1-1 Connection between PC Serial Port to Fujitsu MB96600 Super Series Flash Microcontrollers



For high speed serial on board programming, programming tools of different vendors can be used. For example, the Galep device programmer from CONITEC DATASYSTEMS supports Fujitsu MB96600 Super series Flash memory microcontrollers.

Figure 1-2 Connection between CONITEC DATASYSTEMS Galep Device Programmer to Fujitsu MB96600 Super Series Flash Microcontrollers



Note:

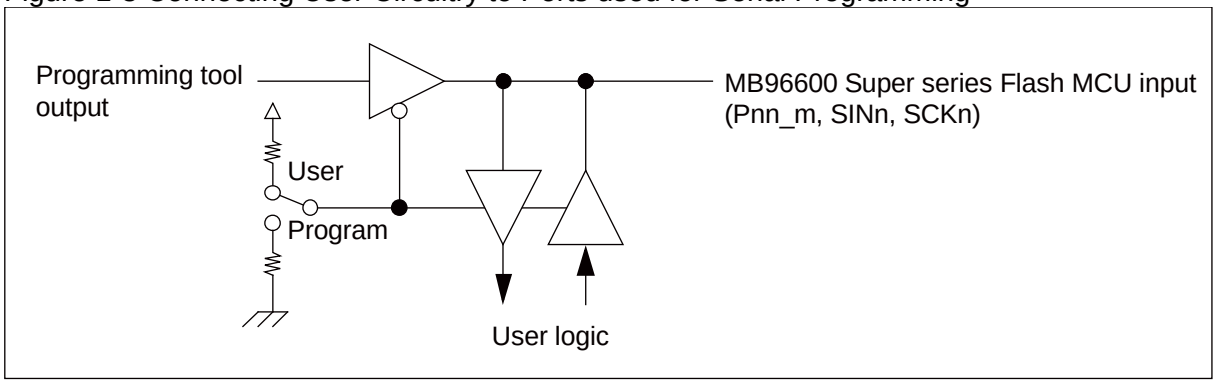
Please refer to CONITEC DATASYSTEMS for details about the Galep device programmer.

Table 1-1 Pins used for Fujitsu Serial Onboard Programming

Pin	Function	Additional information
MD	Mode pin	Controls the operation mode of the MCU. For details, please see "Section 7.4 Boot ROM program execution and Operation mode and ROM Configuration Block".
DEBUG I/F	On Chip Debugger input/output pin	Used for On-Chip debug system and to select Flash serial programming mode and Flash parallel programming mode. For details, please see "Section 7.4 Boot ROM program execution and Operation mode and ROM Configuration Block".
X0, X1	Oscillation pins	Some programming tools require quartz precise timing of the MCU. For this reason, the on-chip quartz oscillator circuit must be used.
Pnn_m	Communication handshaking	Some programming tools require a handshake signal to speed up communicating with the MCU. Please refer to the datasheet about the port recommended for handshaking.
RSTX	Reset pin	Some programming tools require to control the reset line of the MCU.
SINn	Serial data input pin	The USART is used for communication between the programming tool and the MCU. The MCU detects automatically, which USART from a predefined set of USARTs is connected to the programming tool. Please refer to the datasheet for the USART channel numbers that can be used for programing the MCU.
SOTn	Serial data output pin	
SCKn	Serial clock signal input pin	
C	C pin	This external capacitor pin is used to stabilize the power supply. Please refer to the datasheet for the recommended capacitance values and types.
V_{CC}	Power voltage supply pin	If the power is supplied from the user system, the device programmer power supply does not need to be connected. In case the device programmer supply is connected, please connect so that the power supply of the user side is not short-circuited.
V_{SS}	GND pin	Common to the ground of the flash microcomputer programmer.

When the signals driven by the device programmer are also used for the user system, the control circuit shown in Figure 1-3 is required. The circuit enables or disables the programming connection depending on the state of a jumper.

Figure 1-3 Connecting User Circuitry to Ports used for Serial Programming



■ Serial Data Baud Rate

The MCU determines automatically the serial data baud rate as adjusted in the programming tool or PC running programming software.

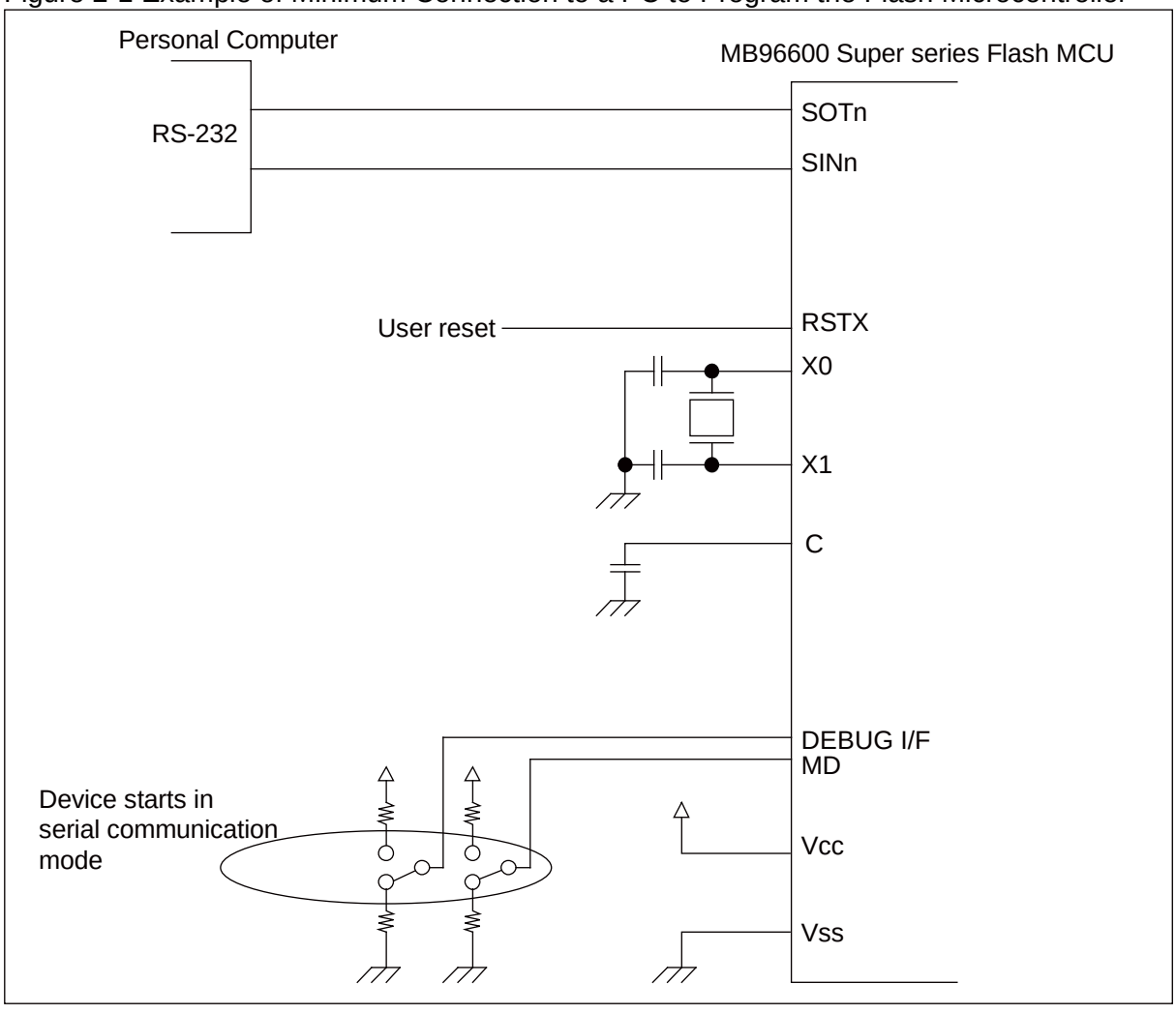
2. Example of connecting a PC for programming the Flash Microcontroller

Figure 2-1 shows an example of the minimum connection to a PC to program the flash microcontroller.

■ Example of Minimum Connection to a PC to Program the Flash Microcontroller

For a minimum connection between a PC and the Flash MCU, only a USART in asynchronous mode needs to be connected. Please refer to the datasheet for an overview of USART channels available for flash serial programming.

Figure 2-1 Example of Minimum Connection to a PC to Program the Flash Microcontroller



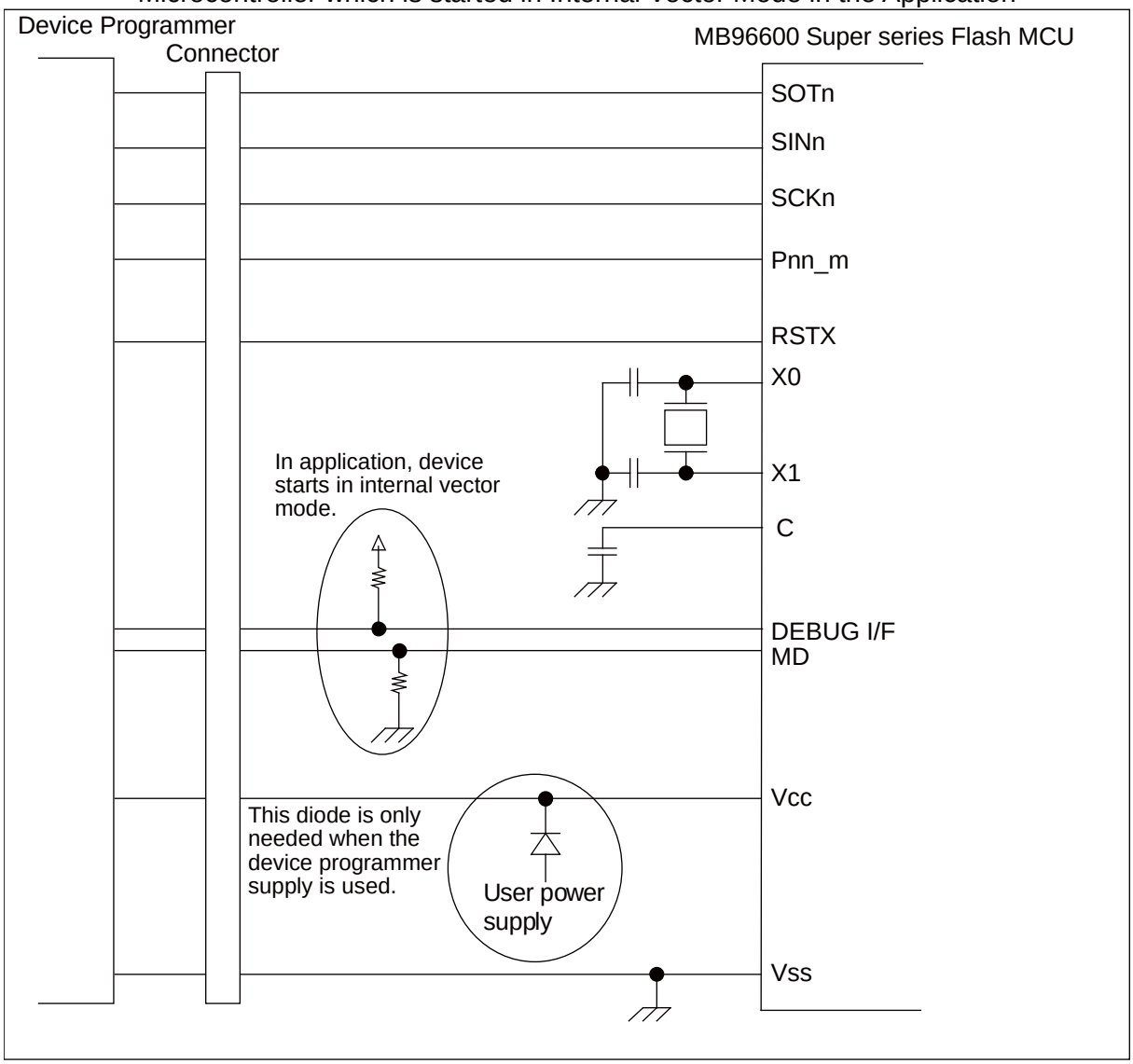
3. Example of connecting a programming tool for programming the Flash Microcontroller

Figure 3-1 is an example of connecting a programming tool to program the flash microcontroller.

■ Example of Connecting a Programming Tool to Program the Flash Microcontroller

For high speed programming a device programmer must be connected as shown in the example below. Please refer to the datasheet for an overview of USART channels available for flash serial programming. Please refer to the datasheet to determine the port used for handshaking between the device programmer and the MCU.

Figure 3-1 Example of a Connection to a Device Programmer to Program the Flash Microcontroller which is started in Internal Vector Mode in the Application



CHAPTER: ON CHIP DEBUGGER (OCD)

This OCD is an embedded debug support unit to offer on-chip debug features in F^2MC-16FX. It has basic debug features (control CPU execution, access to CPU registers and memory), simple debug support features (event detection, execution time measurement, trace, and the like) and debug security.

1. Features
2. Configuration
3. Guideline to Design DEBUG I/F on Target Board

* The contents of this chapter will be extended in the next Hardware Manual release.
For more information about OCD, please consult with FUJITSU sales representatives.

Management Code: 96F6OCD-E01.0

1. Features

This section explains features of on chip debugger (OCD).

- One-wire debug tool I/F
  - Hot-plugging (DSU activate)
  - DSU standby entry (DSU inactivate by command or by timeout)
  - Fast startup mode (Mode decision by setting stored on Flash)

- Debug security function

- Debug mode control function

- Execution control function
  - Status display functions (chip status, CPU status, and the like)
  - Debug command execution control function
  - Small-scale debug main memory (4bytes)
  - CPU register save register (PCB:PC,PS)
  - PC monitor function
  - Issue resets

- Break function
  - Step execution break
  - Event trigger break
  - Forced break (Tool break, selectable interrupt mask level)
  - Specific instruction break
  - Software break
  - Control on interrupt acceptance immediately after the execution start address

- Debug DMA function (DDMA function)
  - Support of transfer modes (address mode, verify mode, DEBUG I/F burst transfer)

- Event function
  - Code event: 4- 8
  - Conditional code event: 2
  - Area code event: 3-areas (Max)
  - Data event: 2-8
  - Area data event: 3-areas (Max)
  - Interrupt event: 2
  - DMA event: 1
  - Event sequencer: 2 levels + reset

- Execution time measurement timer function
  - Go-Break measurement
  - Inter-trigger measurement (single measurement/cumulative measurement)

- Trace function
  - Branch trace (with HW interrupt indication)
  - Number of trace frames: 42

- Semi-hosting feature
  - 1-byte message buffer

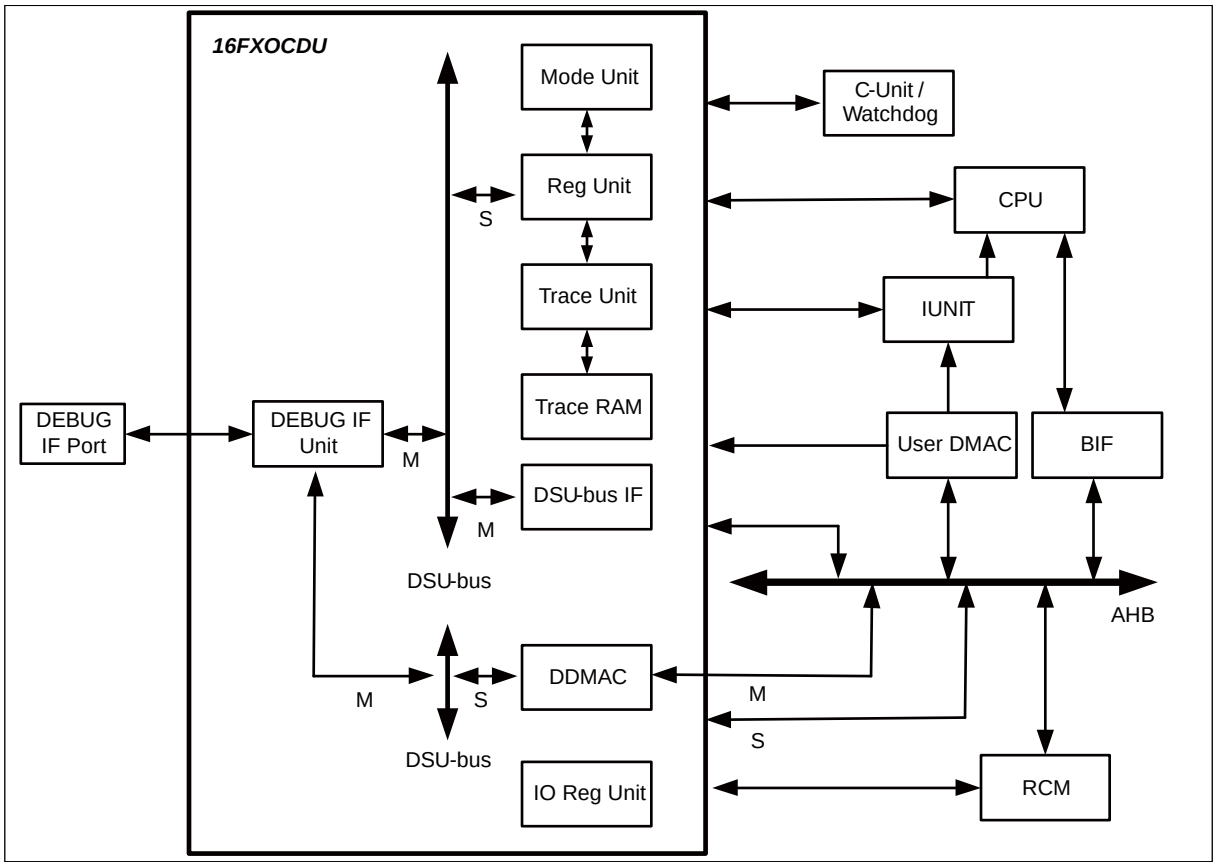
- DSU recovery feature
  - DSU restore with backup register (for chip reset without power-on)
  - DSU recovery from flash (for chip reset with power-on)

2. Configuration

The section explains configuration of OCD.

■ Block Diagram of OCD

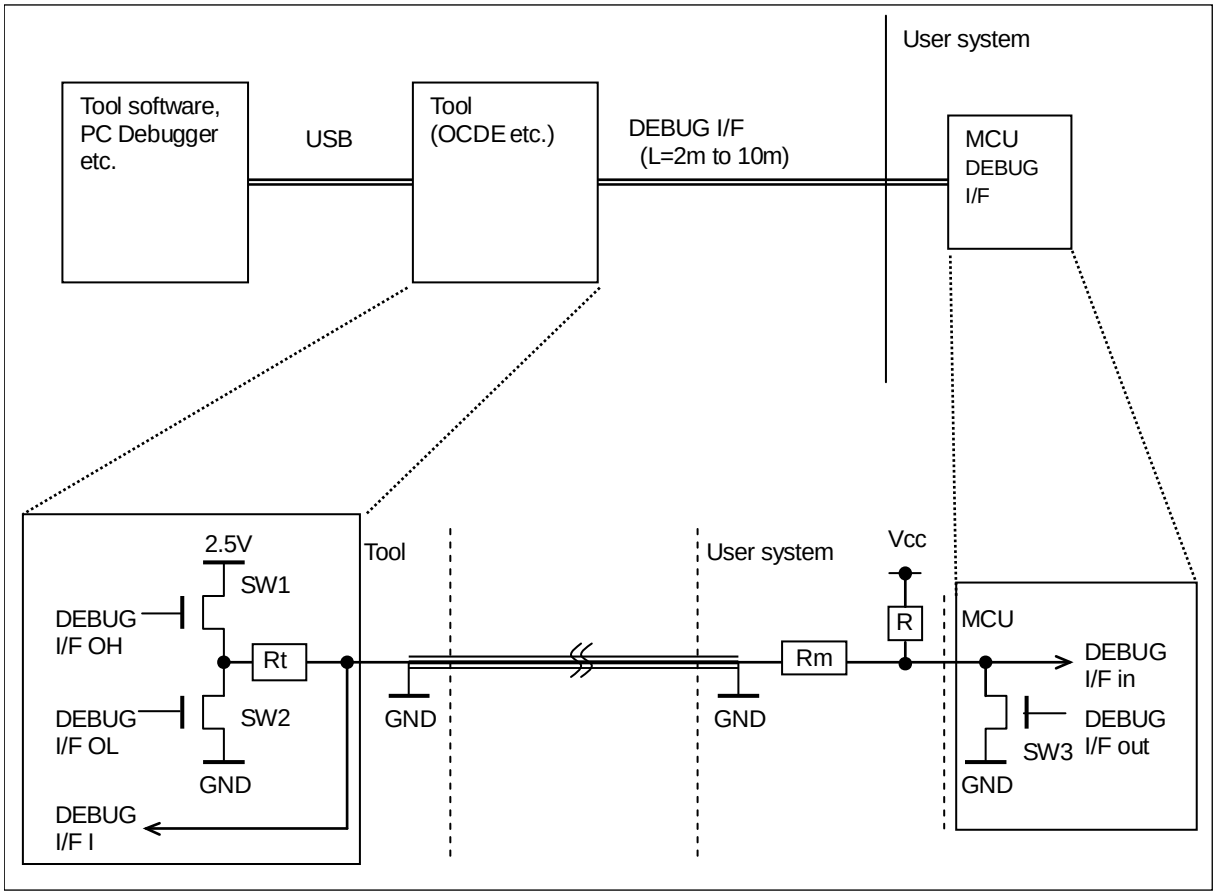
Figure2-1 Block Diagram of OCD



DEBUG IF Unit	This unit has DEBUG interface and is bus master of DSU-bus.
DSU-bus IF Unit	This unit has AHB slave interface and is bus master of DSU-bus.
Reg Unit	This unit has main debug functions - includes all OCD and DBG registers except for DDMA registers. - controls CPU modes and detects CPU status - detects break factors and controls break requests - controls events and detects events - controls execution timer
Mode Unit	This unit controls start up modes and connection/disconnection of OCD.
Trace Unit	This unit controls trace features and accesses to trace RAM.
Trace RAM	This RAM buffers trace data.
DDMAC	This unit is a debug DMA controller. It has a master interface of AHB.
DSU-bus	Internal bus to connect each block inside OCD. It includes arbitration between DEBUG I/F and AHB bus masters.
IO Reg Unit	This unit has user IO registers.

■ OCD Connection Diagram

Figure2-2 OCD Connection Diagram

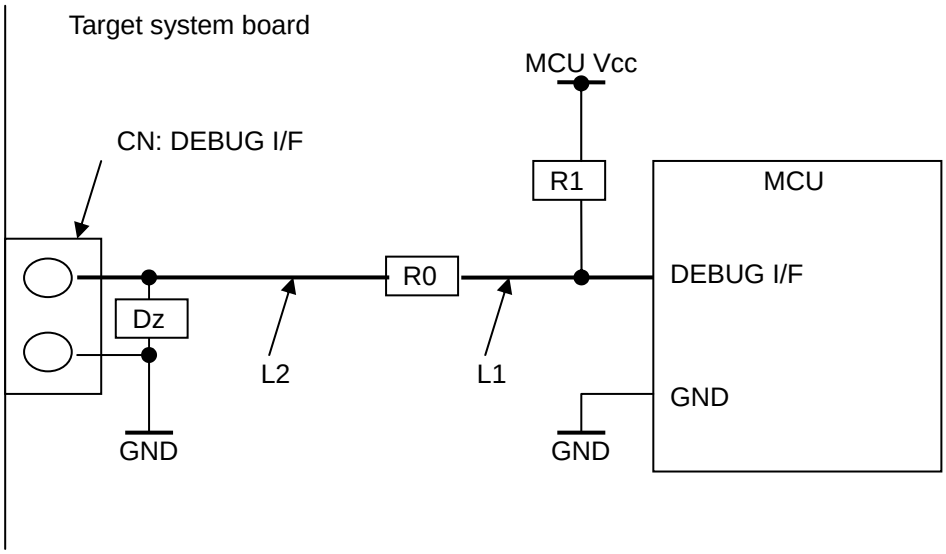


- Rt=Zo-(ON-resistance of SW1, SW2)
- Rm=Zo-(ON-resistance of SW3), defined in MCU
- For fixed level in no driving DEBUG I/F case, pull-down resistor (100K Ω) is added in tool side and pull-up or pull-down resistor (10K Ω) is added in MCU side. Error tolerance for these pull-up or pull-down resistors is -5% to +5%.

3. Guideline to Design DEBUG I/F on Target Board

The section explains guideline to design DEBUG I/F on target board.

■ Standard of Designing DEBUG I/F on Target Board

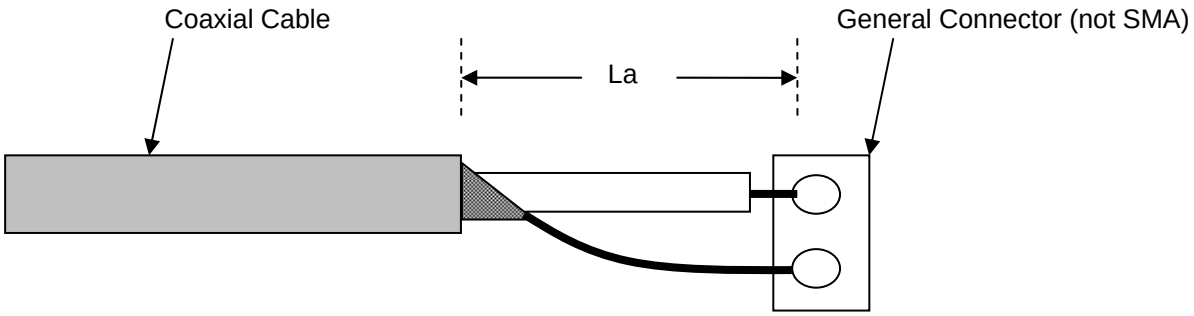


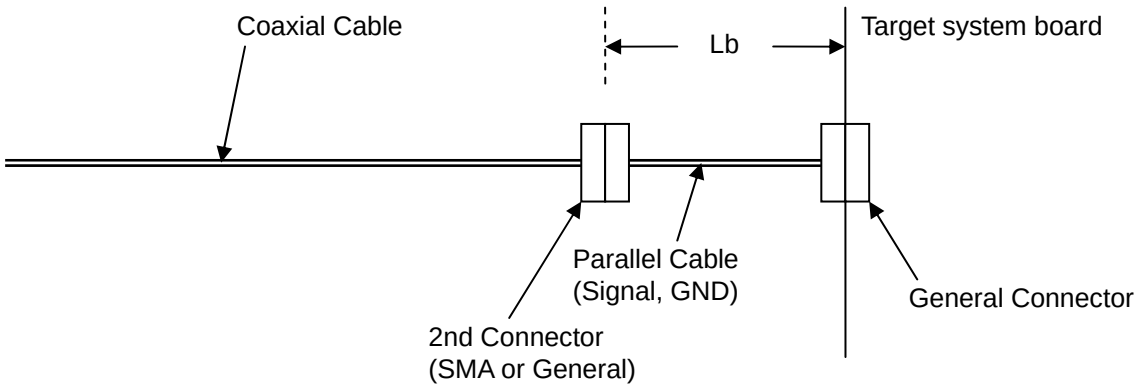
Item	Object	Contents	Priority
01	Selection of CN (1)	SMA connector HRM-300-126B (Hirose Electric) or equivalent component	Recommended
02	Selection of CN (2)	Connector for signal line proper to 50MHz logic signal interface. Current capacity is larger than 25mA. When multi-pin connector is used, DEBUG I/F pin and GND pin must be adjoining, and DEBUG I/F pin is located as far as possible from other high speed or large current signal pin.	Mandatory
03	Selection of R0	Resistor indicated in MCU standard* (Output impedance becomes 50Ω when MCU outputs low). Pw (allowable power dissipation) is more than 0.1W. * R0 = 43 Ω @ F^2MC-16FX MB96600 series	Mandatory
04	Selection of R1	10KΩ (-5% to + 5%). Pw=0.1W (minimum). To use DEBUG I/F, Pull-up is necessary.	Mandatory
05	Selection and layout of Dz	ESD counter measuring Zener Diode for high speed signal line. Capacity is within 5pF. Zener voltage is more than maximum voltage of DEBUG I/F + 0.1V. Locate near to CN.	Recommended
06	Wiring of R0 and Dz	As short as possible.	Mandatory
07	Wiring of GND	Design impedance between MCU and CN, ss small as possible. For multi-layer board with GND layer, connect with GND layer directly.	Mandatory
08	Wire length of L1	Within 5cm. As short as possible.	Mandatory
09	Wire length of L2	Within 15cm, when applying item 01, Within 15cm including the length from R0 to impedance changing point of coaxial cable, when not applying item 01.	Mandatory

Item	Object	Contents	Priority
10	Characteristic impedance of L2	50Ω.	Recommended
11	VIA hole of L1 and L2	2 or less in wire (excluding both ends of the line).	Mandatory
12	Countermeasure for noise of L1 and L2 (1)	Surround both side of the wire with GND pattern.	Recommended
13	Countermeasure for noise of L1 and L2 (2)	Avoid parallel wiring with other signal. For parallel wiring part, prepare gaps to reduce cross-talk between parallel wiring enough.	Mandatory

■ How to Handle General-purpose Coaxial Cable

La and Lb is included to length limitation of L2, in following diagram.
Connecting coaxial cable directly to target board with general connector has more advantage in electric characteristic point of view, than using second connector case (including parallel cable).





APPENDIX

The appendixes provide I/O maps.

A. I/O Map of MB96680

A. I/O Map of MB96680

The I/O Map lists the addresses to be assigned to the registers in the peripheral blocks.

■ I/O maps of MB96680

Table A-1 : I/O map of MB96680

Address	Register	Abbreviation 8-bit access	Abbreviation 16-bit access	Attribute	Initial value
000000_H	I/O Port P00-Port Data Register	PDR00	-	R/W	XXXXXXXX
000001_H	I/O Port P01-Port Data Register	PDR01	-	R/W	XXXXXXXX
000002_H	I/O Port P02-Port Data Register	PDR02	-	R/W	XXXXXXXX
000003_H	I/O Port P03-Port Data Register	PDR03	-	R/W	XXXXXXXX
000004_H	I/O Port P04-Port Data Register	PDR04	-	R/W	XXXXXXXX
000005_H	I/O Port P05-Port Data Register	PDR05	-	R/W	XXXXXXXX
000006_H	I/O Port P06-Port Data Register	PDR06	-	R/W	XXXXXXXX
000007_H	Reserved	-	-	-	-
000008_H	I/O Port P08-Port Data Register	PDR08	-	R/W	XXXXXXXX
000009_H, $00000A_H$	Reserved	-	-	-	-
$00000B_H$	I/O Port P11-Port Data Register	PDR11	-	R/W	XXXXXXXX
$00000C_H$	I/O Port P12-Port Data Register	PDR12	-	R/W	XXXXXXXX
$00000D_H$	I/O Port P13-Port Data Register	PDR13	-	R/W	XXXXXXXX
$00000E_H$-000010_H	Reserved	-	-	-	-
000011_H	I/O Port P17-Port Data Register	PDR17	-	R/W	XXXXXXXX
000012_H-000017_H	Reserved	-	-	-	-
000018_H	ADC0-Control Status Register Lower	ADCSL	ADCS	R/W	00000000
000019_H	ADC0-Control Status Register Higher	ADCSH	-	R/W	00000000
$00001A_H$	ADC0-Data Register Lower	ADCRL	ADCR	R	00000000
$00001B_H$	ADC0-Data Register Higher	ADCRH	-	R	00000000
$00001C_H$	ADC0-Setting Register	-	ADSR	R/W	00000000
$00001D_H$	ADC0-Setting Register	-	-	R/W	00000000
$00001E_H$-$00001F_H$	Reserved	-	-	-	-
000020_H	FRT0-Data Register	-	TCDT0	R/W	XXXXXXXX
000021_H	FRT0-Data Register	-	-	R/W	XXXXXXXX
000022_H	FRT0-Control Status Register Lower	TCCSL0	TCCS0	R/W	00000000
000023_H	FRT0-Control Status Register Higher	TCCSH0	-	R/W	01XXXXXX
000024_H	FRT1-Data Register	-	TCDT1	R/W	XXXXXXXX
000025_H	FRT1-Data Register	-	-	R/W	XXXXXXXX
000026_H	FRT1-Control Status Register Lower	TCCSL1	TCCS1	R/W	00000000
000027_H	FRT1-Control Status Register Higher	TCCSH1	-	R/W	01XXXXXX
000028_H-$00003F_H$	Reserved	-	-	-	-
000040_H	ICU0/ICU1-Control Status Register	ICS01	-	R/W	00000000
000041_H	ICU0/ICU1-Edge Register	ICE01	-	R/W	XXX0X000
000042_H	ICU0-Capture Data Register Lower	IPCPL0	IPCP0	R	XXXXXXXX
000043_H	ICU0-Capture Data Register Higher	IPCPH0	-	R	XXXXXXXX
000044_H	ICU1-Capture Data Register Lower	IPCPL1	IPCP1	R	XXXXXXXX
000045_H	ICU1-Capture Data Register Higher	IPCPH1	-	R	XXXXXXXX
000046_H-$00004B_H$	Reserved	-	-	-	-
$00004C_H$	ICU4/ICU5-Control Status Register	ICS45	-	R/W	00000000

Address	Register	Abbreviation 8-bit access	Abbreviation 16-bit access	Attribute	Initial value
$00004D_H$	ICU4/ICU5-Edge Register	ICE45	-	R/W	XXX0X000
$00004E_H$	ICU4-Capture Data Register Lower	IPCPL4	IPCP4	R	XXXXXXXX
$00004F_H$	ICU4-Capture Data Register Higher	IPCPH4	-	R	XXXXXXXX
000050_H	ICU5-Capture Data Register Lower	IPCPL5	IPCP5	R	XXXXXXXX
000051_H	ICU5-Capture Data Register Higher	IPCPH5	-	R	XXXXXXXX
000052_H - 000057_H	Reserved	-	-	-	-
000058_H	EXTINT0-External Interrupt Enable Register	ENIR0	-	R/W	00000000
000059_H	EXTINT0-External Interrupt Request Register	EIRR0	-	R/W	00000000
$00005A_H$	EXTINT0-External Interrupt Level Register Lower	ELVRL0	ELVR0	R/W	00000000
$00005B_H$	EXTINT0-External Interrupt Level Register Higher	ELVRH0	-	R/W	00000000
$00005C_H$ - 000063_H	Reserved	-	-	-	-
000064_H	RLT1-Timer Control Status Register Lower	TMCSRL1	TMCSR1	R/W	00000000
000065_H	RLT1-Timer Control Status Register Higher	TMCSRH1	-	R/W	00X10000
000066_H	RLT1-16-bit Reload Register	-	TMRLR1	W	XXXXXXXX
000066_H	RLT1-16-bit Timer Register	-	TMR1	R	XXXXXXXX
000067_H	RLT1-16-bit Reload Register	-	-	W	XXXXXXXX
000067_H	RLT1-16-bit Timer Register	-	-	R	XXXXXXXX
000068_H	RLT2-Timer Control Status Register Lower	TMCSRL2	TMCSR2	R/W	00000000
000069_H	RLT2-Timer Control Status Register Higher	TMCSRH2	-	R/W	00X10000
$00006A_H$	RLT2-16-bit Reload Register	-	TMRLR2	W	XXXXXXXX
$00006A_H$	RLT2-16-bit Timer Register	-	TMR2	R	XXXXXXXX
$00006B_H$	RLT2-16-bit Reload Register	-	-	W	XXXXXXXX
$00006B_H$	RLT2-16-bit Timer Register	-	-	R	XXXXXXXX
$00006C_H$ - $00006F_H$	Reserved	-	-	-	-
000070_H	RLT6-Timer Control Status Register Low (dedicated RLT for PPG)	TMCSRL6	TMCSR6	R/W	00000000
000071_H	RLT6-Timer Control Status Register High (dedicated RLT for PPG)	TMCSRH6	-	R/W	00X10000
000072_H	RLT6-16-bit Reload Register (dedicated RLT for PPG)	-	TMRLR6	W	XXXXXXXX
000072_H	RLT6-16-bit Timer Register (dedicated RLT for PPG)	-	TMR6	R	XXXXXXXX
000073_H	RLT6-16-bit Reload Register (dedicated RLT for PPG)	-	-	W	XXXXXXXX
000073_H	RLT6-16-bit Timer Register (dedicated RLT for PPG)	-	-	R	XXXXXXXX
000074_H	PPG3-PPG0-General Control Register 1 Lower	GCN1L0	GCN10	R/W	00010000
000075_H	PPG3-PPG0-General Control Register 1 Higher	GCN1H0	-	R/W	00110010
000076_H	PPG3-PPG0-General Control Register 2 Lower	GCN2L0	GCN20	R/W	XXXX0000
000077_H	PPG3-PPG0-General Control Register 2 Higher	GCN2H0	-	R/W	XXXX0000

Address	Register	Abbreviation 8-bit access	Abbreviation 16-bit access	Attribute	Initial value
000078_H	PPG0-Timer Register	-	PTMR0	R	11111111
000079_H	PPG0-Timer Register	-	-	R	11111111
$00007A_H$	PPG0-Cycle Setting Register	PCSRL0	PCSR0	R/W	XXXXXXXX
$00007B_H$	PPG0-Cycle Setting Register	PCSRH0	-	R/W	XXXXXXXX
$00007C_H$	PPG0-Duty Setting Register	PDUTL0	PDUT0	R/W	XXXXXXXX
$00007D_H$	PPG0-Duty Setting Register	PDUTH0	-	R/W	XXXXXXXX
$00007E_H$	PPG0-Control Status Register Lower	PCNL0	PCN0	R/W	00000000
$00007F_H$	PPG0-Control Status Register Higher	PCNH0	-	R/W	00000000
000080_H	PPG1-Timer Register	-	PTMR1	R	11111111
000081_H	PPG1-Timer Register	-	-	R	11111111
000082_H	PPG1-Cycle Setting Register	PCSRL1	PCSR1	R/W	XXXXXXXX
000083_H	PPG1-Cycle Setting Register	PCSRH1	-	R/W	XXXXXXXX
000084_H	PPG1-Duty Setting Register	PDUTL1	PDUT1	R/W	XXXXXXXX
000085_H	PPG1-Duty Setting Register	PDUTH1	-	R/W	XXXXXXXX
000086_H	PPG1-Control Status Register Lower	PCNL1	PCN1	R/W	00000000
000087_H	PPG1-Control Status Register Higher	PCNH1	-	R/W	00000000
000088_H	PPG2-Timer Register	-	PTMR2	R	11111111
000089_H	PPG2-Timer Register	-	-	R	11111111
$00008A_H$	PPG2-Cycle Setting Register	PCSRL2	PCSR2	R/W	XXXXXXXX
$00008B_H$	PPG2-Cycle Setting Register	PCSRH2	-	R/W	XXXXXXXX
$00008C_H$	PPG2-Duty Setting Register	PDUTL2	PDUT2	R/W	XXXXXXXX
$00008D_H$	PPG2-Duty Setting Register	PDUTH2	-	R/W	XXXXXXXX
$00008E_H$	PPG2-Control Status Register Lower	PCNL2	PCN2	R/W	00000000
$00008F_H$	PPG2-Control Status Register Higher	PCNH2	-	R/W	00000000
000090_H	PPG3-Timer Register	-	PTMR3	R	11111111
000091_H	PPG3-Timer Register	-	-	R	11111111
000092_H	PPG3-Cycle Setting Register	PCSRL3	PCSR3	R/W	XXXXXXXX
000093_H	PPG3-Cycle Setting Register	PCSRH3	-	R/W	XXXXXXXX
000094_H	PPG3-Duty Setting Register	PDUTL3	PDUT3	R/W	XXXXXXXX
000095_H	PPG3-Duty Setting Register	PDUTH3	-	R/W	XXXXXXXX
000096_H	PPG3-Control Status Register Lower	PCNL3	PCN3	R/W	00000000
000097_H	PPG3-Control Status Register Higher	PCNH3	-	R/W	00000000
000098_H - $0000AB_H$	Reserved	-	-	-	-
$0000AC_H$	I^2C0-Bus Status Register	IBSR0	-	R	00000000
$0000AD_H$	I^2C0-Bus Control Register	IBCR0	-	R/W	00000000
$0000AE_H$	I^2C0-Ten Bit Slave Address Register Lower	ITBAL0	ITBA0	R/W	00000000
$0000AF_H$	I^2C0-Ten Bit Slave Address Register Higher	ITBAH0	-	R/W	00000000
$0000B0_H$	I^2C0-Ten Bit Slave Address Mask Register Lower	ITMKL0	ITMK0	R/W	11111111
$0000B1_H$	I^2C0-Ten Bit Slave Address Mask Register Higher	ITMKH0	-	R/W	00111111
$0000B2_H$	I^2C0-Seven Bit Slave Address Register	ISBA0	-	R/W	00000000
$0000B3_H$	I^2C0-Seven Bit Address Mask Register	ISMK0	-	R/W	01111111
$0000B4_H$	I^2C0-Data Register	IDAR0	-	R/W	00000000
$0000B5_H$	I^2C0-Clock Control Register	ICCR0	-	R/W	00011111
$0000B6_H$ - $0000BF_H$	Reserved	-	-	-	-
$0000C0_H$	USART0-Serial Mode Register	SMR0	-	R/W	00000000
$0000C1_H$	USART0-Serial Control Register	SCR0	-	R/W	00000000
$0000C2_H$	USART0-Transmission Register	TDR0	-	W	00000000
$0000C2_H$	USART0-Reception Register	RDR0	-	R	00000000
$0000C3_H$	USART0-Serial Status Register	SSR0	-	R/W	00001000

Address	Register	Abbreviation 8-bit access	Abbreviation 16-bit access	Attribute	Initial value
$0000C4_H$	USART0-Extended Communication Control Register	ECCR0	-	R/W	000000XX
$0000C5_H$	USART0-Extended.Status/Control Register	ESCR0	-	R/W	00000100
$0000C6_H$	USART0-Baud Rate Generation/Reload Register Lower	BGRL0	BGR0	R/W	00000000
$0000C7_H$	USART0-Baud Rate Generation/Reload Register Higher	BGRH0	-	R/W	00000000
$0000C8_H$	USART0-Extended Serial Interrupt Register	ESIR0	-	R/W	XXXX10X0
$0000C9_H$	USART0-Frame-ID Data Register	FIDR0	-	R/W	00000000
$0000CA_H$	USART1-Serial Mode Register	SMR1	-	R/W	00000000
$0000CB_H$	USART1-Serial Control Register	SCR1	-	R/W	00000000
$0000CC_H$	USART1-Transmission Register	TDR1	-	W	00000000
$0000CC_H$	USART1-Reception Register	RDR1	-	R	00000000
$0000CD_H$	USART1-Serial Status Register	SSR1	-	R/W	00001000
$0000CE_H$	USART1- Extended Communication Control Register	ECCR1	-	R/W	000000XX
$0000CF_H$	USART1-Extended Status/Control Register	ESCR1	-	R/W	00000100
$0000D0_H$	USART1-Baud Rate Generation/Reload Register Lower	BGRL1	BGR1	R/W	00000000
$0000D1_H$	USART1-Baud Rate Generation/Reload Register Higher	BGRH1	-	R/W	00000000
$0000D2_H$	USART1-Extended Serial Interrupt Register	ESIR1	-	R/W	XXXX10X0
$0000D3_H$ - $0000EB_H$	Reserved	-	-	-	-
$0000EC_H$	PPG - General Control register	GCNR0		R/W	XXXXXX00
$0000ED_H$ - $0000FF_H$	Reserved	-	-	-	-
000100_H	DMA0-Buffer Address Pointer Lower	BAPL0	-	R/W	XXXXXXXX
000101_H	DMA0-Buffer Address Pointer Middle	BAPM0	-	R/W	XXXXXXXX
000102_H	DMA0-Buffer Address Pointer Higher	BAPH0	-	R/W	XXXXXXXX
000103_H	DMA0-DMA Control Register	DMACS0	-	R/W	XXXXXXXX
000104_H	DMA0-I/O Register Address Pointer Lower	IOAL0	IOA0	R/W	XXXXXXXX
000105_H	DMA0-I/O Register Address Pointer Higher	IOAH0	-	R/W	XXXXXXXX
000106_H	DMA0-Data Count Register Lower	DCTL0	DCT0	R/W	XXXXXXXX
000107_H	DMA0-Data Count Register Higher	DCTH0	-	R/W	XXXXXXXX
000108_H	DMA1-Buffer Address Pointer Lower	BAPL1	-	R/W	XXXXXXXX
000109_H	DMA1-Buffer Address Pointer Middle	BAPM1	-	R/W	XXXXXXXX
$00010A_H$	DMA1-Buffer Address Pointer Higher	BAPH1	-	R/W	XXXXXXXX
$00010B_H$	DMA1-DMA Control Register	DMACS1	-	R/W	XXXXXXXX
$00010C_H$	DMA1-I/O Register Address Pointer Lower	IOAL1	IOA1	R/W	XXXXXXXX
$00010D_H$	DMA1-I/O Register Address Pointer Higher	IOAH1	-	R/W	XXXXXXXX
$00010E_H$	DMA1-Data Count Register Lower	DCTL1	DCT1	R/W	XXXXXXXX
$00010F_H$	DMA1-Data Count Register Higher	DCTH1	-	R/W	XXXXXXXX
000110_H - $00017F_H$	Reserved	-	-	-	-
000180_H - $00037F_H$	CPU-General Purpose registers (RAM access)	GPR_RAM	-	R/W	-

Address	Register	Abbreviation 8-bit access	Abbreviation 16-bit access	Attribute	Initial value
000380_H	DMA0-Interrupt Request Select Register 0	DISEL0	-	R/W	00001100
000381_H	DMA1-Interrupt Request Select Register 1	DISEL1	-	R/W	00001100
000382_H - $00038F_H$	Reserved	-	-	-	-
000390_H	DMA-Status Register Lower	DSRL	DSR	R/W	00000000
000391_H	DMA-Status Register Higher	DSRH	-	R/W	00000000
000392_H	DMA-Stop Status Register Lower	DSSRL	DSSR	R/W	00000000
000393_H	DMA-Stop Status Register Higher	DSSRH	-	R/W	00000000
000394_H	DMA-Enable Register Lower	DERL	DER	R/W	00000000
000395_H	DMA-Enable Register Higher	DERH	-	R/W	00000000
000396_H - $00039F_H$	Reserved	-	-	-	-
$0003A0_H$	Interrupt level register	ILR	ICR	R/W	XXXXX111
$0003A1_H$	Interrupt index register	IDX	-	R/W	00001100
$0003A2_H$	Interrupt Vector Table Base Register Lower	TBRL	TBR	R/W	11111100
$0003A3_H$	Interrupt Vector Table Base Register Higher	TBRH	-	R/W	11111111
$0003A4_H$	Delayed Interrupt Request Register	DIRR	-	R/W	XXXXXXX0
$0003A5_H$	NMI Control Status Register	NMI	-	R/W	XXXXX10X
$0003A6_H$ - $0003AD_H$	Reserved	-	-	-	-
$0003AE_H$	ROM Mirroring Control Register	ROMM	-	R/W	1111X111
$0003AF_H$ - $0003E7_H$	Reserved	-	-	-	-
0003E8	Dual operation Flash Configuration Register 0A	DFCA	-	R/W	XXXXX000
0003E9	Dual operation Interrupt Control Register 1A	DFICA	-	R/W	X0000000
0003EA	Dual operation Flash Status Register A	DFSA	-	R	11000001
0003EB	Dual operation Flash Interrupt Status Register A	DFISA	-	R/W	X0000000
$0003EC_H$ - $0003F8_H$	Reserved	-	-	-	-
$0003F9_H$	Dual operation Flash Write Access Control Register 0A	DFWC0A	-	R/W	XXX00000
$0003FA_H$ - $0003FC_H$	Reserved	-	-	-	-
$0003FD_H$	Dual operation Flash Write Access Control Register 1A	DFWC1A	-	R/W	00000000
$0003FE_H$ - $0003FF_H$	Reserved	-	-	-	-
000400_H	Standby Mode Control Register	SMCR	-	R/W	XXX0X000
000401_H	Clock Selection Register	CKSR	-	R/W	10110000
000402_H	Clock Stabilization Select Register	CKSSR	-	R/W	XX111111
000403_H	Clock Monitor Register	CKMR	-	R	XXXXXXXX
000404_H	Clock Frequency Control Register Lower	CKFCRL	CKFCR	R/W	0000XXX1
000405_H	Clock Frequency Control Register Higher	CKFCRH	-	R/W	00000000
000406_H	PLL Control Register Lower	PLLCRL	PLLCR	R/W	000X0000
000407_H	PLL Control Register Higher	PLLCRH	-	R/W	XXXXXXXX
000408_H	RC Clock Timer Control Register	RCTCR	-	R/W	X0010000
000409_H	Main Clock Timer Control Register	MCTCR	-	R/W	X0010000
$00040A_H$	Sub Clock Timer Control Register	SCTCR	-	R/W	X001X000

Address	Register	Abbreviation 8-bit access	Abbreviation 16-bit access	Attribute	Initial value
$00040B_H$	Reset Cause and Clock Status Register with clear function	RCCSRC	-	R	XXXXXXXX
$00040C_H$	Reset Configuration Register	RCR	-	R/W	XX000110
$00040D_H$	Reset Cause and Clock Status Register	RCCSR	-	R	XXXXXXXX
$00040E_H$	Watchdog Timer Configuration Register	WDTC	-	R/W	0000**** *: depends on WICM Marker
$00040F_H$	Watchdog Timer Clear Pattern Register	WDTCP	-	W	00000000
000410_H - 000414_H	Reserved	-	-	-	-
000415_H	Clock Output Activation Register	COAR	-	R/W	00000000
000416_H	Clock Output Configuration Register 0	COCR0	-	R/W	X0000000
000417_H	Clock Output Configuration Register 1	COCR1	-	R/W	X0000000
000418_H - $00041F_H$	Reserved	-	-	-	-
000420_H	Watchdog Timer Extended Configuration Register	WDTEC	-	R/W	XXX00000
000421_H - $00042B_H$	Reserved	-	-	-	-
$00042C_H$	Voltage Regulator Control Register	VRCR	-	R/W	10X10X10
$00042D_H$	Clock Input and LVD Control Register	CILCR	-	R/W	00000000
$00042E_H$	Extended Standby Mode Control Register	ESMCR	-	R/W	XXX00001
$00042F_H$	Reserved	-	-	-	-
000430_H	I/O Port P00-Data Direction Register	DDR00	-	R/W	00000000
000431_H	I/O Port P01-Data Direction Register	DDR01	-	R/W	00000000
000432_H	I/O Port P02-Data Direction Register	DDR02	-	R/W	00000000
000433_H	I/O Port P03-Data Direction Register	DDR03	-	R/W	00000000
000434_H	I/O Port P04-Data Direction Register	DDR04	-	R/W	00000000
000435_H	I/O Port P05-Data Direction Register	DDR05	-	R/W	00000000
000436_H	I/O Port P06-Data Direction Register	DDR06	-	R/W	00000000
000437_H	Reserved	-	-	-	-
000438_H	I/O Port P08-Data Direction Register	DDR08	-	R/W	00000000
000439_H , $00043A_H$	Reserved	-	-	-	-
$00043B_H$	I/O Port P11-Data Direction Register	DDR11	-	R/W	00000000
$00043C_H$	I/O Port P12-Data Direction Register	DDR12	-	R/W	00000000
$00043D_H$	I/O Port P13-Data Direction Register	DDR13	-	R/W	00000000
$00043E_H$ - 000440_H	Reserved	-	-	-	-
000441_H	I/O Port P17-Data Direction Register	DDR17	-	R/W	00000000
000442_H, 000443_H	Reserved	-	-	-	-
000444_H	I/O Port P00-Port Input Enable Register	PIER00	-	R/W	00000000
000445_H	I/O Port P01-Port Input Enable Register	PIER01	-	R/W	00000000
000446_H	I/O Port P02-Port Input Enable Register	PIER02	-	R/W	00000000
000447_H	I/O Port P03-Port Input Enable Register	PIER03	-	R/W	00000000
000448_H	I/O Port P04-Port Input Enable Register	PIER04	-	R/W	00000000
000449_H	I/O Port P05-Port Input Enable Register	PIER05	-	R/W	00000000
$00044A_H$	I/O Port P06-Port Input Enable Register	PIER06	-	R/W	00000000
$00044B_H$	Reserved	-	-	-	-
$00044C_H$	I/O Port P08-Port Input Enable Register	PIER08	-	R/W	00000000
$00044D_H$, $00044E_H$	Reserved	-	-	-	-
$00044F_H$	I/O Port P11-Port Input Enable Register	PIER11	-	R/W	00000000
000450_H	I/O Port P12-Port Input Enable Register	PIER12	-	R/W	00000000

Address	Register	Abbreviation 8-bit access	Abbreviation 16-bit access	Attribute	Initial value
000451_H	I/O Port P13-Port Input Enable Register	PIER13	-	R/W	00000000
000452_H - 000454_H	Reserved	-	-	-	-
000455_H	I/O Port P17-Port Input Enable Register	PIER17	-	R/W	00000000
000456_H - $00049B_H$	Reserved	-	-	-	-
$00049C_H$	I/O Port P08-Port High Drive Register	PHDR08	-	R/W	00000000
$00049D_H$ - $0004A7_H$	Reserved	-	-	-	-
$0004A8_H$	I/O Port P00-Pull-Up Control Register	PUCR00	-	R/W	00000000
$0004A9_H$	I/O Port P01-Pull-Up Control Register	PUCR01	-	R/W	00000000
$0004AA_H$	I/O Port P02-Pull-Up Control Register	PUCR02	-	R/W	00000000
$0004AB_H$	I/O Port P03-Pull-Up Control Register	PUCR03	-	R/W	00000000
$0004AC_H$	I/O Port P04-Pull-Up Control Register	PUCR04	-	R/W	00000000
$0004AD_H$	I/O Port P05-Pull-Up Control Register	PUCR05	-	R/W	00000000
$0004AE_H$	I/O Port P06-Pull-Up Control Register	PUCR06	-	R/W	00000000
$0004AF_H$	Reserved	-	-	-	-
$0004B0_H$	I/O Port P08-Pull-Up Control Register	PUCR08	-	R/W	00000000
$0004B1_H$, $0004B2_H$	Reserved	-	-	-	-
$0004B3_H$	I/O Port P11-Pull-Up Control Register	PUCR11	-	R/W	00000000
$0004B4_H$	I/O Port P12-Pull-Up Control Register	PUCR12	-	R/W	00000000
$0004B5_H$	I/O Port P13-Pull-Up Control Register	PUCR13	-	R/W	00000000
$0004B6_H$ - $0004B8_H$	Reserved	-	-	-	-
$0004B9_H$	I/O Port P17-Pull-Up Control Register	PUCR17	-	R/W	00000000
$0004BA_H$, $0004BB_H$	Reserved	-	-	-	-
$0004BC_H$	I/O Port P00-External Pin State Register	EPSR00	-	R	XXXXXXXX
$0004BD_H$	I/O Port P01-External Pin State Register	EPSR01	-	R	XXXXXXXX
$0004BE_H$	I/O Port P02-External Pin State Register	EPSR02	-	R	XXXXXXXX
$0004BF_H$	I/O Port P03-External Pin State Register	EPSR03	-	R	XXXXXXXX
$0004C0_H$	I/O Port P04-External Pin State Register	EPSR04	-	R	XXXXXXXX
$0004C1_H$	I/O Port P05-External Pin State Register	EPSR05	-	R	XXXXXXXX
$0004C2_H$	I/O Port P06-External Pin State Register	EPSR06	-	R	XXXXXXXX
$0004C3_H$	Reserved	-	-	-	-
$0004C4_H$	I/O Port P08-External Pin State Register	EPSR08	-	R	XXXXXXXX
$0004C5_H$, $0004C6_H$	Reserved	-	-	-	-
$0004C7_H$	I/O Port P11-External Pin State Register	EPSR11	-	R	XXXXXXXX
$0004C8_H$	I/O Port P12-External Pin State Register	EPSR12	-	R	XXXXXXXX
$0004C9_H$	I/O Port P13-External Pin State Register	EPSR13	-	R	XXXXXXXX
$0004CA_H$ - $0004CC_H$	Reserved	-	-	-	-
$0004CD_H$	I/O Port P17-External Pin State Register	EPSR17	-	R	XXXXXXXX
$0004CE_H$, $0004D0_H$	Reserved	-	-	-	-
$0004D1_H$	ADC0 Analog Input Enable Register 1	ADER1	-	R/W	00000000
$0004D2_H$	ADC0 Analog Input Enable Register 2	ADER2	-	R/W	00000000
$0004D3_H$, $0004D5_H$	Reserved	-	-	-	-
$0004D6_H$	Peripheral Resource Relocation Register 0	PRRR0	-	R/W	00000000
$0004D7_H$	Reserved	-	-	-	-

Address	Register	Abbreviation 8-bit access	Abbreviation 16-bit access	Attribute	Initial value
0004D8$_H$	Peripheral Resource Relocation Register 2	PRRR2	-	R/W	00000000
0004D9$_H$	Peripheral Resource Relocation Register 3	PRRR3	-	R/W	00000000
0004DA$_H$	Peripheral Resource Relocation Register 4	PRRR4	-	R/W	00000000
0004DB$_H$	Reserved	-	-	-	-
0004DC$_H$	Peripheral Resource Relocation Register 6	PRRR6	-	R/W	00000000
0004DD$_H$, 0004DE$_H$	Reserved	-	-	-	-
0004DF$_H$	Peripheral Resource Relocation Register 9	PRRR9	-	R/W	XX000000
0004E0$_H$	RTC-Sub Second Register Lower	WTBRL0	WTBR0	R/W	XXXXXXXX
0004E1$_H$	RTC-Sub Second Register Middle	WTBRH0	-	R/W	XXXXXXXX
0004E2$_H$	RTC-Sub-Second Register Higher	WTBR1	-	R/W	XXXXXXXX
0004E3$_H$	RTC-Second Register	WTSR	-	R/W	XX000000
0004E4$_H$	RTC-Minute Register	WTMR	-	R/W	XX000000
0004E5$_H$	RTC-Hour Register	WTHR	-	R/W	XXX00000
0004E6$_H$	RTC-Timer Control Extended Register	WTCER	-	R/W	XXXXXX00
0004E7$_H$	RTC-Clock Select Register	WTCKSR	-	R/W	XXXXXX00
0004E8$_H$	RTC-Timer Control Register Low	WTCRL	WTCR	R/W	XXXX0000
0004E9$_H$	RTC-Timer Control Register High	WTCRH	-	R/W	00000000
0004EA$_H$	CAL-Control Register	CUCR	-	R/W	XXX0X000
0004EB$_H$	Reserved	-	-	-	-
0004EC$_H$	CAL-Duration Timer Data Register Lower	CUTDL	CUTD	R/W	00000000
0004ED$_H$	CAL-Duration Timer Data Register Higher	CUTDH	-	R/W	10000000
0004EE$_H$	CAL-Calibration Timer Register 2 Lower	CUTR2L	CUTR2	R	00000000
0004EF$_H$	CAL-Calibration Timer Register 2 Higher	CUTR2H	-	R	00000000
0004F0$_H$	CAL-Calibration Timer Register 1 Lower	CUTR1L	CUTR1	R	00000000
0004F1$_H$	CAL-Calibration Timer Register 1 Higher	CUTR1H	-	R	00000000
0004F2$_H$ - 0004F9$_H$	Reserved	-	-	-	-
0004FA$_H$	RLT-Input Select Register (for Cascading)	TMISR	-	R/W	XX000000
0004FB$_H$ - 0005DF$_H$	Reserved	-	-	-	-
0005E0$_H$	SMC0-PWM Control Register	PWC0	-	R/W	00000000
0005E1$_H$	SMC0-PWM Extended Control Register	PWEC0	-	R/W	00000000
0005E2$_H$	SMC0-PWM 1 Compare Register	-	PWC10	R/W	XXXXXXXX
0005E3$_H$	SMC0-PWM 1 Compare Register	-	-	R/W	000000XX
0005E4$_H$	SMC0-PWM 2 Compare Register	-	PWC20	R/W	XXXXXXXX
0005E5$_H$	SMC0-PWM 2 Compare Register	-	-	R/W	000000XX
0005E6$_H$	SMC0-PWM1 Select Register	PWS10	-	R/W	00000000
0005E7$_H$	SMC0-PWM2 Select Register	PWS20	-	R/W	00000000
0005E8$_H$, 0005E9$_H$	Reserved	-	-	-	-
0005EA$_H$	SMC1-PWM Control Register	PWC1	-	R/W	00000000
0005EB$_H$	SMC1-PWM Extended Control Register	PWEC1	-	R/W	00000000
0005EC$_H$	SMC1-PWM 1 Compare Register	-	PWC11	R/W	XXXXXXXX
0005ED$_H$	SMC1-PWM 1 Compare Register	-	-	R/W	000000XX
0005EE$_H$	SMC1-PWM2 Compare Register	-	PWC21	R/W	XXXXXXXX
0005EF$_H$	SMC1-PWM2 Compare Register	-	-	R/W	000000XX

Address	Register	Abbreviation 8-bit access	Abbreviation 16-bit access	Attribute	Initial value
$0005F0_H$	SMC1-PWM1 Select Register	PWS11	-	R/W	00000000
$0005F1_H$	SMC1-PWM2 Select Register	PWS21	-	R/W	00000000
$0005F2_H$ - $00061B_H$	Reserved	-	-	-	-
$00061C_H$	LCD-Segment Enable Register 0 (SEG 7-0)	LCDER0	-	R/W	00000000
$00061D_H$	LCD-Segment Enable Register 1 (SEG 15-8)	LCDER1	-	R/W	00000000
$00061E_H$	LCD-Segment Enable Register 2 (SEG 23-16)	LCDER2	-	R/W	00000000
$00061F_H$	LCD-Segment Enable Register 3 (SEG 31-24)	LCDER3	-	R/W	00000000
000620_H	LCD-Segment Enable Register 4 (SEG 39-32)	LCDER4	-	R/W	00000000
000621_H	LCD-Segment Enable Register 5 (SEG 47-40)	LCDER5	-	R/W	00000000
000622_H	LCD-Segment Enable Register 6 (SEG 55-48)	LCDER6	-	R/W	00000000
000623_H	LCD-Segment Enable Register 7 (SEG 63-56)	LCDER7	-	R/W	00000000
000624_H, 000625_H	Reserved	-	-	-	-
000626_H	LCD-Segment Enable Register V (Vx)	LCDVER	-	R/W	XXXX0000
000627_H	LCD-Extended Control Register	LECR	-	R/W	XXXXX000
000628_H	LCD-Common pin switching register	LCDCMR	-	R/W	0XXX0000
000629_H	LCD-Control Register	LCR	-	R/W	00010000
$00062A_H$	LCD-Display RAM for Segment 1-0	VRAM0	-	R/W	XXXXXXXX
$00062B_H$	LCD-Display RAM for Segment 3-2	VRAM1	-	R/W	XXXXXXXX
$00062C_H$	LCD-Display RAM for Segment 5-4	VRAM2	-	R/W	XXXXXXXX
$00062D_H$	LCD-Display RAM for Segment 7-6	VRAM3	-	R/W	XXXXXXXX
$00062E_H$	LCD-Display RAM for Segment 9-8	VRAM4	-	R/W	XXXXXXXX
$00062F_H$	LCD-Display RAM for Segment 11-10	VRAM5	-	R/W	XXXXXXXX
000630_H	LCD-Display RAM for Segment 13-12	VRAM6	-	R/W	XXXXXXXX
000631_H, 000632_H	Reserved	-	-	-	-
000633_H	LCD-Display RAM for Segment 19-18	VRAM9	-	R/W	XXXXXXXX
000634_H	LCD-Display RAM for Segment 21-20	VRAM10	-	R/W	XXXXXXXX
000635_H	LCD-Display RAM for Segment 23-22	VRAM11	-	R/W	XXXXXXXX
000636_H	LCD-Display RAM for Segment 25-24	VRAM12	-	R/W	XXXXXXXX
000637_H, 000638_H	Reserved	-	-	-	-
000639_H	LCD-Display RAM for Segment 31-30	VRAM15	-	R/W	XXXXXXXX
$00063A_H$, $00063B_H$	Reserved	-	-	-	-
$00063C_H$	LCD-Display RAM for Segment 37-36	VRAM18	-	R/W	XXXXXXXX
$00063D_H$	LCD-Display RAM for Segment 39-38	VRAM19	-	R/W	XXXXXXXX
$00063E_H$	Reserved	-	-	-	-
$00063F_H$	LCD-Display RAM for Segment 43-42	VRAM21	-	R/W	XXXXXXXX
000640_H	LCD-Display RAM for Segment 45-44	VRAM22	-	R/W	XXXXXXXX
000641_H	LCD-Display RAM for Segment 47-46	VRAM23	-	R/W	XXXXXXXX
000642_H, 000643_H	Reserved	-	-	-	-
000644_H	LCD-Display RAM for Segment 53-52	VRAM26	-	R/W	XXXXXXXX
000645_H	LCD-Display RAM for Segment 55-54	VRAM27	-	R/W	XXXXXXXX
000646_H	LCD-Display RAM for Segment 57-56	VRAM28	-	R/W	XXXXXXXX

Address	Register	Abbreviation 8-bit access	Abbreviation 16-bit access	Attribute	Initial value
000647_H - 000662_H	Reserved	-	-	-	-
000663_H	Peripheral Resource Relocation Register 13	PRRR13	-	R/W	XXX0XX00
000664_H - $0006FF_H$	Reserved	-	-	-	-
000700_H	CAN0-Control Register Lower	CTRLRL0	CTRLR0	R/W	000X0001
000701_H	CAN0-Control Register Higher (Reserved)	CTRLRH0	-	R	XXXXXXXX
000702_H	CAN0-Status Register Lower	STATRL0	STATR0	R/W	00000000
000703_H	CAN0-Status Register Higher (Reserved)	STATRH0	-	R	XXXXXXXX
000704_H	CAN0-Error Counter Lower (Transmit)	ERRCNTL0	ERRCNT0	R	00000000
000705_H	CAN0-Error Counter Higher (Receive)	ERRCNTH0	-	R	00000000
000706_H	CAN0-Bit Timing Register Lower	BTRL0	BTR0	R/W	00000001
000707_H	CAN0-Bit Timing Register Higher	BTRH0	-	R/W	X0100011
000708_H	CAN0-Interrupt Register Lower	INTRL0	INTR0	R	00000000
000709_H	CAN0-Interrupt Register Higher	INTRH0	-	R	00000000
$00070A_H$	CAN0-Test Register Lower	TESTRL0	TESTR0	R/W	000000XX
$00070B_H$	CAN0-Test Register Higher (reserved)	TESTRH0	-	R	XXXXXXXX
$00070C_H$	CAN0-BRP Extension Register Lower	BRPERL0	BRPER0	R/W	XXXX0000
$00070D_H$	CAN0-BRP Extension Register Higher (reserved)	BRPERH0	-	R	XXXXXXXX
$00070E_H$, $00070F_H$	Reserved	-	-	-	-
000710_H	CAN0-IF1 Command Request Register Lower	IF1CREQL0	IF1CREQ0	R/W	00000001
000711_H	CAN0-IF1 Command Request Register Higher	IF1CREQH0	-	R/W	0XXXXXXX
000712_H	CAN0-IF1 Command Mask Register Lower	IF1CMSKL0	IF1CMSK0	R/W	00000000
000713_H	CAN0-IF1 Command Mask Register Higher (reserved)	IF1CMSKH0	-	R	XXXXXXXX
000714_H	CAN0-IF1 Mask 1 Register Lower	IF1MSK1L0	IF1MSK10	R/W	11111111
000715_H	CAN0-IF1 Mask 1 Register Higher	IF1MSK1H0	-	R/W	11111111
000716_H	CAN0-IF1 Mask 2 Register Lower	IF1MSK2L0	IF1MSK20	R/W	11111111
000717_H	CAN0-IF1 Mask 2 Register Higher	IF1MSK2H0	-	R/W	11X11111
000718_H	CAN0-IF1 Arbitration 1 Register Lower	IF1ARB1L0	IF1ARB10	R/W	00000000
000719_H	CAN0-IF1 Arbitration 1 Register Higher	IF1ARB1H0	-	R/W	00000000
$00071A_H$	CAN0-IF1 Arbitration 2 Register Lower	IF1ARB2L0	IF1ARB20	R/W	00000000
$00071B_H$	CAN0-IF1 Arbitration 2 Register Higher	IF1ARB2H0	-	R/W	00000000
$00071C_H$	CAN0-IF1 Message Control Register Lower	IF1MCTRL0	IF1MCTR0	R/W	0XXX0000
$00071D_H$	CAN0-IF1 Message Control Register Higher	IF1MCTRH0	-	R/W	00000000
$00071E_H$	CAN0-IF1 Data A1 Register Lower	IF1DTA1L0	IF1DTA10	R/W	00000000
$00071F_H$	CAN0-IF1 Data A1 Register Higher	IF1DTA1H0	-	R/W	00000000
000720_H	CAN0-IF1 Data A2 Register Lower	IF1DTA2L0	IF1DTA20	R/W	00000000
000721_H	CAN0-IF1 Data A2 Register Higher	IF1DTA2H0	-	R/W	00000000
000722_H	CAN0-IF1 Data B1 Register Lower	IF1DTB1L0	IF1DTB10	R/W	00000000
000723_H	CAN0-IF1 Data B1 Register Higher	IF1DTB1H0	-	R/W	00000000
000724_H	CAN0-IF1 Data B2 Register Lower	IF1DTB2L0	IF1DTB20	R/W	00000000
000725_H	CAN0-IF1 Data B2 Register Higher	IF1DTB2H0	-	R/W	00000000
000726_H - $00073F_H$	Reserved	-	-	-	-

Address	Register	Abbreviation 8-bit access	Abbreviation 16-bit access	Attribute	Initial value
000740_H	CAN0-IF2 Command Request Register Lower	IF2CREQL0	IF2CREQ0	R/W	00000001
000741_H	CAN0-IF2 Command Request Register Higher	IF2CREQH0	-	R/W	0XXXXXXX
000742_H	CAN0-IF2 Command Mask Register Lower	IF2CMSKL0	IF2CMSK0	R/W	00000000
000743_H	CAN0-IF2 Command Mask Register Higher (reserved)	IF2CMSKH0	-	R	XXXXXXXX
000744_H	CAN0-IF2 Mask 1 Register Lower	IF2MSK1L0	IF2MSK10	R/W	11111111
000745_H	CAN0-IF2 Mask 1 Register Higher	IF2MSK1H0	-	R/W	11111111
000746_H	CAN0-IF2 Mask 2 Register Lower	IF2MSK2L0	IF2MSK20	R/W	11111111
000747_H	CAN0-IF2 Mask 2 Register Higher	IF2MSK2H0	-	R/W	11X11111
000748_H	CAN0-IF2 Arbitration 1 Register Lower	IF2ARB1L0	IF2ARB10	R/W	00000000
000749_H	CAN0-IF2 Arbitration 1 Register Higher	IF2ARB1H0	-	R/W	00000000
$00074A_H$	CAN0-IF2 Arbitration 2 Register Lower	IF2ARB2L0	IF2ARB20	R/W	00000000
$00074B_H$	CAN0-IF2 Arbitration 2 Register Higher	IF2ARB2H0	-	R/W	00000000
$00074C_H$	CAN0-IF2 Message Control Register Lower	IF2MCTRL0	IF2MCTR0	R/W	0XXX0000
$00074D_H$	CAN0-IF2 Message Control Register Higher	IF2MCTRH0	-	R/W	00000000
$00074E_H$	CAN0-IF2 Data A1 Register Lower	IF2DTA1L0	IF2DTA10	R/W	00000000
$00074F_H$	CAN0-IF2 Data A1 Register Higher	IF2DTA1H0	-	R/W	00000000
000750_H	CAN0-IF2 Data A2 Register Lower	IF2DTA2L0	IF2DTA20	R/W	00000000
000751_H	CAN0-IF2 Data A2 Register Higher	IF2DTA2H0	-	R/W	00000000
000752_H	CAN0-IF2 Data B1 Register Lower	IF2DTB1L0	IF2DTB10	R/W	00000000
000753_H	CAN0-IF2 Data B1 Register Higher	IF2DTB1H0	-	R/W	00000000
000754_H	CAN0-IF2 Data B2 Register Lower	IF2DTB2L0	IF2DTB20	R/W	00000000
000755_H	CAN0-IF2 Data B2 Register Higher	IF2DTB2H0	-	R/W	00000000
000756_H - $00077F_H$	Reserved	-	-	-	-
000780_H	CAN0-Transmission Request 1 Register Low	TREQR1L0	TREQR10	R	00000000
000781_H	CAN0-Transmission Request 1 Register High	TREQR1H0	-	R	00000000
000782_H	CAN0-Transmission Request 2 Register Low	TREQR2L0	TREQR20	R	00000000
000783_H	CAN0-Transmission Request 2 Register High	TREQR2H0	-	R	00000000
000784_H - $00078F_H$	Reserved	-	-	-	-
000790_H	CAN0-New Data 1 Register Low	NEWDT1L0	NEWDT10	R	00000000
000791_H	CAN0-New Data 1 Register High	NEWDT1H0	-	R	00000000
000792_H	CAN0-New Data 2 Register Low	NEWDT2L0	NEWDT20	R	00000000
000793_H	CAN0-New Data 2 Register High	NEWDT2H0	-	R	00000000
000794_H - $00079F_H$	Reserved	-	-	-	-
$0007A0_H$	CAN0-Interrupt Pending 1 Register Low	INTPND1L0	INTPND10	R	00000000
$0007A1_H$	CAN0-Interrupt Pending 1 Register High	INTPND1H0	-	R	00000000
$0007A2_H$	CAN0-Interrupt Pending 2 Register Low	INTPND2L0	INTPND20	R	00000000
$0007A3_H$	CAN0-Interrupt Pending 2 Register High	INTPND2H0	-	R	00000000
$0007A4_H$ - $0007AF_H$	Reserved	-	-	-	-
$0007B0_H$	CAN0-Message Valid 1 Register Low	MSGVAL1L0	MSGVAL10	R	00000000
$0007B1_H$	CAN0-Message Valid 1 Register High	MSGVAL1H0	-	R	00000000
$0007B2_H$	CAN0-Message Valid 2 Register Low	MSGVAL2L0	MSGVAL20	R	00000000

Address	Register	Abbreviation 8-bit access	Abbreviation 16-bit access	Attribute	Initial value
$0007B3_H$	CAN0-Message Valid 2 Register High	MSGVAL2H0	-	R	00000000
$0007B4_H$ - $0007CD_H$	Reserved	-	-	-	-
$0007CE_H$	CAN0-Output Enable Register	COER0	-	R/W	XXXXXXX0
$0007CF_H$	Reserved	-	-	-	-
$0007D0_H$	SG0-Control Register Lower	SGCRL0	SGCR0	R/W	00000000
$0007D1_H$	SG0-Control Register Higher	SGCRH0	-	R/W	XXXXX100
$0007D2_H$	SG0-Frequency Register	SGFR0	-	R/W	XXXXXXXX
$0007D3_H$	SG0-Amplitude Register	SGAR0	-	R/W	XXXXXXXX
$0007D4_H$	SG0-Decrement Register	SGDR0	-	R/W	XXXXXXXX
$0007D5_H$	SG0-Tone Register	SGTR0	-	R/W	XXXXXXXX
$0007D6_H$ - $0009FF_H$	Reserved	-	-	-	-
$000A00_H$	DMA-IO Address Pointer Bank Select Register 0	IOABK0	-	R/W	00000000
$000A01_H$	DMA-IO Address Pointer Bank Select Register 1	IOABK1	-	R/W	00000000
$000A02_H$ - $000A7F_H$	Reserved	-	-	-	-
$000A80_H$	PPG0-Extended PPG Control Status Register 1 Lower	EPCN1L0	EPCN10	R/W	00000000
$000A81_H$	PPG0-Extended PPG Control Status Register 1 Higher	EPCN1H0	-	R/W	XX000000
$000A82_H$	PPG1-Extended PPG Control Status Register 1 Lower	EPCN1L1	EPCN11	R/W	00000000
$000A83_H$	PPG1-Extended PPG Control Status Register 1 Higher	EPCN1H1	-	R/W	XX000000
$000A84_H$	PPG2-Extended PPG Control Status Register 1 Lower	EPCN1L2	EPCN12	R/W	00000000
$000A85_H$	PPG2-Extended PPG Control Status Register 1 Higher	EPCN1H2	-	R/W	XX000000
$000A86_H$	PPG3-Extended PPG Control Status Register 1 Lower	EPCN1L3	EPCN13	R/W	00000000
$000A87_H$	PPG3-Extended PPG Control Status Register 1 Higher	EPCN1H3	-	R/W	XX000000
$000A88_H$ - $000B54_H$	Reserved	-	-	-	-
$000B55_H$	ADC0-Range Comparator Over Threshold Flag Register	ADRCOOF10	-	R	00000000
$000B56_H$	ADC0-Range Comparator Over Threshold Flag Register	ADRCOOF20	-	R	00000000
$000B57_H$ - $000B5C_H$	Reserved	-	-	-	-
$000B5D_H$	ADC0-Range Comparator Flag Register	ADRCOINTF10	-	R/W	00000000
$000B5E_H$	ADC0-Range Comparator Flag Register	ADRCOINTF20	-	R/W	00000000
$000B5F_H$ - $000B64_H$	Reserved	-	-	-	-
$000B65_H$	ADC0-Pulse Counter Interrupt Enable Register 1	ADPCIE10	-	R/W	00000000
$000B66_H$	ADC0-Pulse Counter Interrupt Enable Register 2	ADPCIE20	-	R/W	00000000
$000B67_H$ - $000B68_H$	Reserved	-	-	-	-
$000B69_H$	ADC0-Pulse Counter Zero Flag Register 1	ADPCZF10	-	R/W	00000000

Address	Register	Abbreviation 8-bit access	Abbreviation 16-bit access	Attribute	Initial value
$000B6A_H$	ADC0-Pulse Counter Zero Flag Register 2	ADPCZF20	-	R/W	00000000
$000B6B_H$ - $000BA9_H$	Reserved	-	-	-	-
$000BAA_H$	USART0-Checksum Status and Control Register	CSCR0	-	R/W	00000000
$000BAB_H$	USART0-Extended Status Register	ESR0	-	R/W	XX000000
$000BAC_H$	Reserved	-	-	-	-
$000BAD_H$	USART1-Extended Status Register	ESR1	-	R/W	XXXX00XX
$000BAE_H$ - $000BEF_H$	Reserved	-	-	-	-
$000BF0_H$ - $000BF9_H$	OCD Area	-	-	-	-
$000BFA_H$ - $000BFF_H$	Reserved				
$0E9000_H$ - $0E92FF_H$	Reserved	-	-	-	-
$0E9300_H$	PPG3-PPG0 - General Control Register 3 Lower	GCN3L0	GCN30	R/W	X000X000
$0E9301_H$	PPG3-PPG0 - General Control Register 3 Higher	GCN3H0	-	R/W	X000X000
$0E9302_H$	PPG3-PPG0 - General Control Register 4 Lower	GCN4L0	GCN40	R/W	X110X110
$0E9303_H$	PPG3-PPG0 - General Control Register 4 Higher	GCN4H0	-	R/W	X110X110
$0E9304_H$	PPG3-PPG0 - General Control Register 5 Lower	GCN5L0	GCN50	R/W	00000000
$0E9305_H$ - $0E930F_H$	Reserved	-	-	-	-
$0E9310_H$	PPG0 - Extended PPG Control Status Register 2 Lower	EPCN2L0	EPCN20	R/W	00XX0000
$0E9311_H$ - $0E9313_H$	Reserved	-	-	-	-
$0E9314_H$	PPG0 - Timing Point Capture Register Lower	PTPCL0	PTPC0	R/W	00000000
$0E9315_H$	PPG0 - Timing Point Capture Register Higher	PTPCH0	-	R/W	00000000
$0E9316_H$ - $0E931F_H$	Reserved	-	-	-	-
$0E9320_H$	PPG1 - Extended PPG Control Status Register 2 Lower	EPCN2L1	EPCN21	R/W	00XX0000
$0E9321_H$ - $0E9323_H$	Reserved	-	-	-	-
$0E9324_H$	PPG1 - Timing Point Capture Register Lower	PTPCL1	PTPC1	R/W	00000000
$0E9325_H$	PPG1 - Timing Point Capture Register Higher	PTPCH1	-	R/W	00000000
$0E9326_H$ - $0E932F_H$	Reserved	-	-	-	-
$0E9330_H$	PPG2 - Extended PPG Control Status register 2 Lower	EPCN2L2	EPCN22	R/W	00XX0000
$0E9331_H$ - $0E9333_H$	Reserved	-	-	-	-
$0E9334_H$	PPG2 - Timing Point Capture Register Lower	PTPCL2	PTPC2	R/W	00000000

Address	Register	Abbreviation 8-bit access	Abbreviation 16-bit access	Attribute	Initial value
$0E9335_H$	PPG2 - Timing Point Capture Register Higher	PTPCH2	-	R/W	00000000
$0E9336_H$ - $0E933F_H$	Reserved	-	-	-	-
$0E9340_H$	PPG3 - Extended PPG Control Status Register 2 Lower	EPCN2L3	EPCN23	R/W	00XX0000
$0E9341_H$ - $0E9343_H$	Reserved	-	-	-	-
$0E9344_H$	PPG3 - Timing Point Capture Register Lower	PTPCL3	PTPC3	R/W	00000000
$0E9345_H$	PPG3 - Timing Point Capture Register Higher	PTPCH3	-	R/W	00000000
$0E9346_H$ - $0E949F_H$	Reserved	-	-	-	-
$0E94A0_H$	ADTG-ADC0 Trigger Control Register Lower	ADTGCRL0	ADTGCR0	R/W	XXXXXXXX
$0E94A1_H$	ADTG-ADC0 Trigger Control Register Higher	ADTGCRH0	-	R/W	XXXXXXX1
$0E94A2_H$	ADTG-ADC0 Trigger Input Select Lower	ADTGISELL0	ADTGISEL0	R/W	XXXXX001
$0E94A3_H$	ADTG-ADC0 Trigger Input Select Higher	ADTGISELH0	-	R/W	XXX00000
$0E94A4_H$ - $0E951F_H$	Reserved	-	-	-	-
$0E9520_H$	ADC0-Lower Threshold Register for Range Comparator 0 Lower	ADRCOLL00	ADRCOL00	R/W	00000000
$0E9521_H$	ADC0-Lower Threshold Register for Range Comparator 0 Higher	ADRCOLH00	-	R/W	00000000
$0E9522_H$	ADC0-Upper Threshold Register for Range Comparator 0 Lower	ADRCOHL00	ADRCOH00	R/W	00000000
$0E9523_H$	ADC0-Upper Threshold Register for Range Comparator 0 Higher	ADRCOHH00	-	R/W	00000000
$0E9524_H$	ADC0-Lower Threshold Register for Range Comparator 1 Lower	ADRCOLL10	ADRCOL10	R/W	00000000
$0E9525_H$	ADC0-Lower Threshold Register for Range Comparator 1 Higher	ADRCOLH10	-	R/W	00000000
$0E9526_H$	ADC0-Upper Threshold Register for Range Comparator 1 Lower	ADRCOHL10	ADRCOH10	R/W	00000000
$0E9527_H$	ADC0-Upper Threshold Register for Range Comparator 1 Higher	ADRCOHH10	-	R/W	00000000
$0E9528_H$	ADC0-Lower Threshold Register for Range Comparator 2 Lower	ADRCOLL20	ADRCOL20	R/W	00000000
$0E9529_H$	ADC0-Lower Threshold Register for Range Comparator 2 Higher	ADRCOLH20	-	R/W	00000000
$0E952A_H$	ADC0-Upper Threshold Register for Range Comparator 2 Lower	ADRCOHL20	ADRCOH20	R/W	00000000
$0E952B_H$	ADC0-Upper Threshold Register for Range Comparator 2 Higher	ADRCOHH20	-	R/W	00000000
$0E952C_H$	ADC0-Lower Threshold Register for Range Comparator 3 Lower	ADRCOLL30	ADRCOL30	R/W	00000000
$0E952D_H$	ADC0-Lower Threshold Register for Range Comparator 3 Higher	ADRCOLH30	-	R/W	00000000
$0E952E_H$	ADC0-Upper Threshold Register for Range Comparator 3 Lower	ADRCOHL30	ADRCOH30	R/W	00000000
$0E952F_H$	ADC0-Upper Threshold Register for Range Comparator 3 Higher	ADRCOHH30	-	R/W	00000000

Address	Register	Abbreviation 8-bit access	Abbreviation 16-bit access	Attribute	Initial value
$0E9530_H$	Reserved	-	-	-	-
$0E9531_H$	ADC0-Disable Channel for scan Register 1	ADDSCR10	-	R/W	00000000
$0E9532_H$	ADC0-Disable Channel for scan Register 2	ADDSCR20	-	R/W	00000000
$0E9533_H$ - $0E9540_H$	Reserved	-	-	-	-
$0E9541_H$	ADC0-Inverted Range Selection Register 1 for Range Comparator	ADRCOIRS10	-	R/W	00000000
$0E9542_H$	ADC0-Inverted Range Selection Register 2 for Range Comparator	ADRCOIRS20	-	R/W	00000000
$0E9543_H$ - $0E9553_H$	Reserved	-	-	-	-
$0E9554_H$	ADC0-Channel Control Register 4 for Range Comparator	ADCC040	-	R/W	00000000
$0E9555_H$	ADC0-Channel Control Register 5 for Range Comparator	ADCC050	-	R/W	00000000
$0E9556_H$	ADC0-Channel Control Register 6 for Range comparator	ADCC060	-	R/W	00000000
$0E9557_H$	Reserved	-	-	-	-
$0E9558_H$	ADC0-Channel Control Register 8 for Range Comparator	ADCC080	-	R/W	00000000
$0E9559_H$	ADC0-Channel Control Register 9 for Range Comparator	ADCC090	-	R/W	00000000
$0E955A_H$	ADC0-Channel Control Register 10 for Range Comparator	ADCC100	-	R/W	00000000
$0E955B_H$	ADC0-Channel Control Register 11 for Range Comparator	ADCC110	-	R/W	00000000
$0E955C_H$ - $0E959F_H$	Reserved	-	-	-	-
$0E95A0_H$	ADC0-Pulse Positive Counter Reload Register 8	ADPCTPRD80	-	R/W	00000000
$0E95A1_H$	ADC0-Pulse Negative Counter Reload Register 8	ADPCTNRD80	-	R/W	00000000
$0E95A2_H$	ADC0-Pulse Positive Counter Reload Register 9	ADPCTPRD90	-	R/W	00000000
$0E95A3_H$	ADC0-Pulse Negative Counter Reload Register 9	ADPCTNRD90	-	R/W	00000000
$0E95A4_H$	ADC0-Pulse Positive Counter Reload Register 10	ADPCTPRD100	-	R/W	00000000
$0E95A5_H$	ADC0-Pulse Negative Counter Reload Register 10	ADPCTNRD100	-	R/W	00000000
$0E95A6_H$	ADC0-Pulse Positive Counter Reload Register 11	ADPCTPRD110	-	R/W	00000000
$0E95A7_H$	ADC0-Pulse Negative Counter Reload Register 11	ADPCTNRD110	-	R/W	00000000
$0E95A8_H$	ADC0-Pulse Positive Counter Reload Register 12	ADPCTPRD120	-	R/W	00000000
$0E95A9_H$	ADC0-Pulse Negative Counter Reload Register 12	ADPCTNRD120	-	R/W	00000000
$0E95AA_H$	ADC0-Pulse Positive Counter Reload Register 13	ADPCTPRD130	-	R/W	00000000
$0E95AB_H$	ADC0-Pulse Negative Counter Reload Register 13	ADPCTNRD130	-	R/W	00000000

Address	Register	Abbreviation 8-bit access	Abbreviation 16-bit access	Attribute	Initial value
$0E95AC_H$ - $0E95AF_H$	Reserved	-	-	-	-
$0E95B0_H$	ADC0-Pulse Positive Counter Reload Register 16	ADPCTPRD160	-	R/W	00000000
$0E95B1_H$	ADC0-Pulse Negative Counter Reload Register 16	ADPCTNRD160	-	R/W	00000000
$0E95B2_H$	ADC0-Pulse Positive Counter Reload Register 17	ADPCTPRD170	-	R/W	00000000
$0E95B3_H$	ADC0-Pulse Negative Counter Reload Register 17	ADPCTNRD170	-	R/W	00000000
$0E95B4_H$	ADC0-Pulse Positive Counter Reload Register 18	ADPCTPRD180	-	R/W	00000000
$0E95B5_H$	ADC0-Pulse Negative Counter Reload Register 18	ADPCTNRD180	-	R/W	00000000
$0E95B6_H$	ADC0-Pulse Positive Counter Reload Register 19	ADPCTPRD190	-	R/W	00000000
$0E95B7_H$	ADC0-Pulse Negative Counter Reload Register 19	ADPCTNRD190	-	R/W	00000000
$0E95B8_H$	ADC0-Pulse Positive Counter Reload Register 20	ADPCTPRD200	-	R/W	00000000
$0E95B9_H$	ADC0-Pulse Negative Counter Reload Register 20	ADPCTNRD200	-	R/W	00000000
$0E95BA_H$	ADC0-Pulse Positive Counter Reload Register 21	ADPCTPRD210	-	R/W	00000000
$0E95BB_H$	ADC0-Pulse Negative Counter Reload Register 21	ADPCTNRD210	-	R/W	00000000
$0E95BC_H$	ADC0-Pulse Positive Counter Reload Register 22	ADPCTPRD220	-	R/W	00000000
$0E95BD_H$	ADC0-Pulse Negative Counter Reload Register 22	ADPCTNRD220	-	R/W	00000000
$0E95BE_H$	ADC0-Pulse Positive Counter Reload Register 23	ADPCTPRD230	-	R/W	00000000
$0E95BF_H$	ADC0-Pulse Negative Counter Reload Register 23	ADPCTNRD230	-	R/W	00000000
$0E95C0_H$ - $0E95DF_H$	Reserved	-	-	-	-
$0E95E0_H$	ADC0-Pulse Positive Counter Register 8	ADPCTPCT80	-	R	00000000
$0E95E1_H$	ADC0-Pulse Negative Counter Register 8	ADPCTNCT80	-	R	00000000
$0E95E2_H$	ADC0-Pulse Positive Counter Register 9	ADPCTPCT90	-	R	00000000
$0E95E3_H$	ADC0-Pulse Negative Counter Register 9	ADPCTNCT90	-	R	00000000
$0E95E4_H$	ADC0-Pulse Positive Counter Register 10	ADPCTPCT100	-	R	00000000
$0E95E5_H$	ADC0-Pulse Negative Counter Register 10	ADPCTNCT100	-	R	00000000
$0E95E6_H$	ADC0-Pulse Positive Counter Register 11	ADPCTPCT110	-	R	00000000
$0E95E7_H$	ADC0-Pulse Negative Counter Register 11	ADPCTNCT110	-	R	00000000
$0E95E8_H$	ADC0-Pulse Positive Counter Register 12	ADPCTPCT120	-	R	00000000
$0E95E9_H$	ADC0-Pulse Negative Counter Register 12	ADPCTNCT120	-	R	00000000
$0E95EA_H$	ADC0-Pulse Positive Counter Register 13	ADPCTPCT130	-	R	00000000
$0E95EB_H$	ADC0-Pulse Negative Counter Register 13	ADPCTNCT130	-	R	00000000
$0E95EC_H$ - $0E95EF_H$	Reserved	-	-	-	-
$0E95F0_H$	ADC0-Pulse Positive Counter Register 16	ADPCTPCT160	-	R	00000000

Address	Register	Abbreviation 8-bit access	Abbreviation 16-bit access	Attribute	Initial value
0E95F1$_H$	ADC0-Pulse Negative Counter Register 16	ADPCTNCT160	-	R	00000000
0E95F2$_H$	ADC0-Pulse Positive Counter Register 17	ADPCTPCT170	-	R	00000000
0E95F3$_H$	ADC0-Pulse Negative Counter Register 17	ADPCTNCT170	-	R	00000000
0E95F4$_H$	ADC0-Pulse Positive Counter Register 18	ADPCTPCT180	-	R	00000000
0E95F5$_H$	ADC0-Pulse Negative Counter Register 18	ADPCTNCT180	-	R	00000000
0E95F6$_H$	ADC0-Pulse Positive Counter Register 19	ADPCTPCT190	-	R	00000000
0E95F7$_H$	ADC0-Pulse Negative Counter Register 19	ADPCTNCT190	-	R	00000000
0E95F8$_H$	ADC0-Pulse Positive Counter Register 20	ADPCTPCT200	-	R	00000000
0E95F9$_H$	ADC0-Pulse Negative Counter Register 20	ADPCTNCT200	-	R	00000000
0E95FA$_H$	ADC0-Pulse Positive Counter Register 21	ADPCTPCT210	-	R	00000000
0E95FB$_H$	ADC0-Pulse Negative Counter Register 21	ADPCTNCT210	-	R	00000000
0E95FC$_H$	ADC0-Pulse Positive Counter Register 22	ADPCTPCT220	-	R	00000000
0E95FD$_H$	ADC0-Pulse Negative Counter Register 22	ADPCTNCT220	-	R	00000000
0E95FE$_H$	ADC0-Pulse Positive Counter Register 23	ADPCTPCT230	-	R	00000000
0E95FF$_H$	ADC0-Pulse Negative Counter Register 23	ADPCTNCT230	-	R	00000000
0E9600$_H$-0E97BF$_H$	Reserved	-	-	-	-
0E97C0$_H$	USART0-Extended Feature Enable Register Lower	EFERL0	EFER0	R/W	00000000
0E97C1$_H$	USART0-Extended Feature Enable Register Higher	EFERH0	-	R/W	XX000000
0E97C2$_H$-0E97C5$_H$	Reserved	-	-	-	-
0E97C6$_H$	USART0-Synch Field Time-Out Register Lower	SFTRL0	SFTR0	R/W	00000000
0E97C7$_H$	USART0-Synchfield_Time-Out Register Higher	SFTRH0	-	R/W	00000000
0E97C8$_H$	USART0-Extended Interrupt Enable Register	EIER0	-	R/W	XX000000
0E97D0$_H$	USART1-Extended Feature Enable Register Lower	EFERL1	EFER1	R/W	00000XXX
0E97D1$_H$	USART1-Extended Feature Enable Register Higher	EFERH1	-	R/W	XX00X0X0
0E97D2$_H$-0E97D7$_H$	Reserved	-	-	-	-
0E97D8$_H$	USART1-Extended Interrupt Enable Register	EIER1	-	R/W	XXXXXX00
0E97D9$_H$-0E9897$_H$	Reserved	-	-	-	-
0E9898$_H$	I/O Port P08 - Pull-Down Control Register	PDCR08		R/W	00000000
0E9899$_H$-0E9FFF$_H$	Reserved	-	-	-	-

MN704-00001-1v0-E

FUJITSU SEMICONDUCTOR · CONTROLLER MANUAL

F^2MC-16FX

16-BIT MICROCONTROLLER

MB96680 series

HARDWARE MANUAL

May 2011 the first edition

Published **FUJITSU SEMICONDUCTOR LIMITED**

Edited Sales Promotion Department.