User Manual

32

Hardware

V850E/Dx3 - DG3

32-bit Single-Chip Microcontroller

µPD70F3416
µPD70F3417

All information contained in these materials, including products and product specifications, represents information on the product at the time of publication and is subject to change by Renesas Electronics Corp. without notice. Please review the latest information published by Renesas Electronics Corp. through various means, including the Renesas Electronics Corp. website (http://www.renesas.com).

Renesas Electronics
www.renesas.com

R01UH0027ED0420, Rev. 4.20
Feb 15, 2013

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: “Standard”, “High Quality”, and “Specific”. The recommended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as “Specific” without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as “Specific” or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics.

The quality grade of each Renesas Electronics product is “Standard” unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

“Standard”: Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.

“High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.

“Specific”: Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.

8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU products from Renesas. For detailed usage notes on the products covered by this manual, refer to the relevant sections of the manual. If the descriptions under General Precautions in the Handling of MPU/MCU Products and in the body of the manual differ from each other, the description in the body of the manual takes precedence.

1. Handling of unused Pins

   Handle unused pins in accord with the directions given under Handling of Unused Pins in the manual.

   – The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.

2. Processing at power-on

   The state of the product is undefined at the moment when power is supplied.

   – The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.

     In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.

     In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.

3. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited.

   – The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.

4. Clock signals

   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.

   – When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.

5. Differences between products

   Before changing from one product to another, i.e. to one with a different part number, confirm that the change will not lead to problems.

   – The characteristics of MPU/MCU in the same group but having different part numbers may differ because of the differences in internal memory capacity and layout pattern. When changing to products of different part numbers, implement a system-evaluation test for each of the products.

How to use this manual

Purpose and target readers

This manual is designed to provide the user with an understanding of the hardware functions of the microcontroller. It is intended for users designing application systems incorporating the microcontroller. A basic knowledge of electric circuits, logical circuits, and microcontrollers is necessary in order to use this manual.

Special notations

Following special notations are used throughout this document:

Note Additional remark or tip

Caution Item deserving extra attention

Electrical specifications

This manual does not present any electrical specifications.
Refer to the Data Sheet for detailed definitions of all electrical properties.
For information about the Data Sheet document, refer to the section *“Related Documents”* in the chapter *“Introduction”*.

Additional documents

Following types of documents are available for the V850E/Dx3 - DG3 microcontrollers. Make sure to refer to the latest versions of these documents. The newest versions of the documents listed may be obtained from the Renesas Electronics Web site.

Document Type	Description	Document
Data sheet	Hardware overview and electrical characteristics	Refer to the section *“Related Documents”* in the chapter *“Introduction”*
User’s manual: Hardware	Hardware specifications (pin assignments, memory maps, functional modules specifications and operation description) Note: Refer to the application notes for details on using functional modules.	
User’s manual: 32-bit Microprocessor Core Architecture	Description of CPU, its instruction set and processor protection functions	
Application note	Information on using peripheral functions and application examples, sample programs and information on writing programs in assembly language and C	Available from Renesas Electronics Web site
Renesas technical update	Product specifications, updates on documents, etc.	

Content of this manual

In the following brief hints are given where to find certain information about the V850E/Dx3 - DG3 microcontrollers.

Product overview

Refer to the chapter “Introduction” for an overview of the features of all target microcontrollers and their block diagrams.
Order codes for all devices and a list of related documents is given here as well.

CPU core functions

The functions of the CPU core (e.g. instruction set, processor protection functions, etc.) are not subject to this manual. Refer to the separate CPU core manual, shown in the section “Related Documents” in the chapter “Introduction”.

CPU Subsystem functions

The functions of the CPU Subsystem (including address map, operation modes, etc.) are described in the chapter “CPU System Function”.
The section “Write protected Registers” in this chapter describes how to deal with registers, that feature special write protection facilities.
If the microcontroller has separate bus systems beside the CPU Subsystem to connect certain functional modules, refer to the chapter “Bus Architecture”.

Port functions

The chapter “Port Functions” describes all input/output port related functions, such as port sharing, I/O buffer control, port filters.
The features and electrical properties of the I/O buffers are not subject to this manual, but are described in the Data Sheet.

Interrupt functions

Refer to the chapter “Interrupt Controller”.
Note that the function of each interrupt source is not described here, but in the related chapter of the module, that generates the interrupt.

DMA/DTS functions

Refer to the chapter “DMA/DTS Controller” or “DMA Controller”, if the target microcontroller does not feature DTS functions.
Note that the function of each DMA/DTS trigger source is not described here, but in the related chapter of the module, that generates the trigger signal.

Flash memory

For microcontrollers with on-chip flash memory refer to the chapter “Flash Memory” for information about the flash memories structure and features, programming facilities, etc.

Stand-by functions

How to set the microcontroller in stand-by modes and wake it up again is described in the sub-chapter “Power Save Modes” in the chapter “Clock Generator”.

Code protection and security

Facilities to protect program code in on-chip flash memory (if available) from illegal read-out via external flash programming equipment or debuggers is described in the chapter “Code Protection and Security”.

Clock supply The chapter “Clock Controller” describes the generation and operation of all clocks, provide to the entire microcontroller.

Resets The sources that can generate reset signals to all or dedicated internal modules and how to control them is described in the chapter “Reset Controller”.

Functional modules The description of most functional modules, like timers, serial interfaces, etc. is provided in separate chapters.

Debugging The main features on the On-Chip Debug Unit of the microcontroller is described in the chapter “On-chip Debug Unit (OCD)”.
Note that the description of the external debugger tool is not subject to this manual.

Power supply The chapter “Power Supply Scheme” provides information which modules of the microcontrollers are supplied by which external power supply pins.
Note that the specification of the external power supply is not subject to this manual. Refer to the Data Sheet for detailed definitions of the power supply.

Boundary scan If the target microcontroller supports boundary scan testing, refer to the chapter “Boundary Scan” for information about available Boundary Scan features.

Notation of numbers and symbols

Symbols Symbols and notation are used as follows:

• Weight in data notation:	Left is high order column, right is low order column
• Active low notation:	$\overline{\text{xxx}}$ (pin or signal name is over-scored) or /xxx (slash before signal name)
• Memory map address:	High order at high stage and low order at low stage

Numeric notation

• Binary:	xxxx or xxx_B
• Decimal:	xxxx
• Hexadecimal:	xxxx_H or 0x xxxx

Numeric prefixes representing powers of 2 (address space, memory capacity):

• K (kilo):	$2^{10} = 1024$
• M (mega):	$2^{20} = 1024^2 = 1{,}048{,}576$
• G (giga):	$2^{30} = 1024^3 = 1{,}073{,}741{,}824$

Register contents X, x = don’t care

Diagrams

Block diagrams do not necessarily show the exact wiring in hardware but the functional structure.

Timing diagrams are for functional explanation purposes only, without any relevance to the real hardware implementation.

List of abbreviations and acronyms

Abbreviation	Full form
ACIA	Asynchronous Communication Interface Adapter
bps	bits per second
CRC	Cyclic Redundancy Check
DMA	Direct Memory Access
DMAC	Direct Memory Access Controller
GSM	Global System for Mobile Communications
Hi-Z	High Impedance
IEBus	Inter Equipment Bus
I/O	Input / Output
IrDA	Infrared Data Association
LSB	Least Significant Bit
MSB	Most Significant Bit
NC	Non-Connect
PLL	Phase Locked Loop
PWM	Pulse Width Modulation
SFR	Special Function Registers
SIM	Subscriber Identity Module
UART	Universal Asynchronous Receiver / Transmitter
VCO	Voltage Controlled Oscillator

Trademarks All trademarks are the property of their respective owners.

Register and bit descriptions

Register access description

Each register description includes information how to access the register in following format:

Access	This register can be accessed in <access units>.
Address	<MODULE_base> + X_H

(1) Register access width

The <access units> are declared in multiple of bytes as

- 32-bit units
- 16-bit units
- 8-bit units

The <access unit> defines the bytes, which hold defined bits.
Since all registers can be accessed with 32-bit, 16-bit and 8-bit width, despite of the <access unit>, a related number of upper significant bits become undefined, as shown in the following table:

• V[n:m]:	bits have defined values (refer to register the description)
• x:	writing has not no effect, reading returns an undefined value
• n.a.:	not subject to the read/write access

Register <access unit>	Access width	Read/write value		
		bit[31:16]	bit[15:8]	bit[7:0]
8 bit	32-bit	x	x	V[7:0]
	16-bit	n.a.	x	V[7:0]
	8-bit	n.a.	n.a.	V[7:0]
16 bit	32-bit	x	V[15:8]	V[7:0]
	16-bit	n.a.	V[15:8]	V[7:0]
	8-bit	n.a.	n.a.	V[7:0]
32 bit	32-bit	V[31:16]	V[15:8]	V[7:0]
	16-bit	n.a.	V[15:8]	V[7:0]
	8-bit	n.a.	n.a.	V[7:0]

(2) Register addresses

The addresses of most registers are given as offsets (X_H) to a module's base address <MODULE_base> ("MODULE" stands for the module's shortcut). The base address is defined in the first - product specific - section of the chapter under the key word *"Register addresses"*.

The addresses of the registers are always aligned to word - i.e. 32-bit - boundaries (at addresses xxxx xxx0_H, xxxx xxx4_H, xxxx xxx8_H, xxxx xxxC_H), independent of the access width. Thus an 8-bit access targets always bit[7:0], a 16-bit access always bit[15:0].

(3) Exceptions: CPU Subsystem registers

The registers of modules of the CPU Subsystem (e.g. Interrupt Controller, DMA Controller) provide also registers with 1-bit access units with H/W protected read-modify-write accesses:

The register of these modules allow also access on byte and half-word (16-bit) aligned addresses (addresses xxxx xxx0_H to xxxx xxxF_H).

Bit access description

Named and unnamed bits

A register may hold named and unnamed bits.

Named bits are declared with a bit name. In general these bits are referenced with the register name and bit name in the following format:

RegisterName.BitName

Unnamed bits don't have a name and are shown in the bit image of a register with

- 0, if their value is fixed to 0
- 1, if their value is fixed to 1
- x, if their value is undefined

The access options to register bits are defined by following access attributes:

- read/write: R/W
- read-only: R
- write-only: W

The effect of any register bit read/write access is determined by the register bit attribute and is summarizes in the following table:

Bit attribute	Write access	Read access
R/W	defines the bit value	returns the bit value
R	has no effect	returns the bit value
W	defines the bit value	returns undefined or fixed bit value[a]

[a] The return value is defined in the register description.

Initial values Unnamed bits are shown in the register bit image with their initial (default) values.

In most cases these bits are declared as “R”, thus its initial value is read back and writing has no effect.

If the default value of an unnamed bit

- must not be changed or
- must be changed

in order to guarantee proper operation of the module, the bit is declared as “R/W” and special instructions are given how to handle that bit.

Following an example of an 8-bit register:

Initial Value 80_H

7	6	5	4	3	2	1	0
1	0[a]	0	0	0[b]	BitName2	BitName1	BitName0
R	R/W	R	R	R/W	W	R	R/W

[a] The default value “0” of bit 6 must be changed to “1” before the module is used.
[b] The default value “0” of bit 3 must not be changed.

Caution

1. The default value “0” of bit 6 must be changed to “1” before the module is used.
2. The default value “0” of bit 3 must not be changed.

Table 1-1 <RegisterName> register contents

Bit position	Bit name	Function
6	Bit 7	**Caution:** The default value “0” of bit 6 must be changed to “1” before the module is used.
2	BitName2	Explanation of BitName2 function, when writing “0” or “1” Information what is read back (“0” or “1” or “undefined”)
1	BitName1	Explanation of BitName1 read value (writing to BitName1 has no effect)
0	BitName0	Explanation of BitName0 function, when writing “0” or “1”

Further information

For further information see http://www.renesas.com.

Table of Contents

Chapter 18 CAN Controller (CAN) 546

Chapter 1 Introduction

V850E/Dx3 series The V850E/Dx3 is a product series in Renesas Electronics V850 family of single-chip microcontrollers designed for automotive applications. Beside the V850E/Dx3 - DG3 the poduct series comprises the V850E/DJ3 and V850E/DL3 devices. For further information about V850E/DJ3 and V850E/DL3 refer to the user's manual
"V850E/Dx3 - DJ3/DL3"
Document number R01UH0129ED

1.1 General

The V850E/Dx3 - DG3 single-chip microcontroller devices make the performance gains attainable with 32-bit RISC-based controllers available for embedded control applications. The integrated V850 CPU offers easy pipeline handling and programming, resulting in compact code size comparable to 16-bit CISC CPUs.

The V850E/Dx3 - DG3 provide an excellent combination of general purpose peripheral functions, like serial communication interfaces (UARTs, Clocked Serial Interfaces), timers, and measurement inputs (A/D Converter), with dedicated CAN network support.

The devices offer specific power-saving modes to manage the power consumption effectively under varying conditions.

Thus equipped, the V850E/Dx3 - DG3 product line is ideally suited for automotive applications, like dashboard or body. It is also an excellent choice for other applications where a combination of sophisticated peripheral functions and CAN network support is required.

(1) V850E CPU

The V850E CPU core is a RISC processor. Through the use of basic instructions that can be executed in one clock period combined with an optimized pipeline architecture, it achieves marked improvements in instruction execution speed.

In addition, to make it ideal for use in digital control applications, a 32-bit hardware multiplier enables this CPU to support multiply instructions, saturated multiply instructions, bit operation instructions, etc.

Through two-byte basic instructions and instructions compatible with high level languages, the object code efficiency in a C compiler is increased, and program size can be reduced.

Further, because the on-chip interrupt controller provides high-speed interrupt response and processing, this device is well suited for high level real-time control applications.

(2) On-chip flash memory

The V850E/Dx3 - DG3 microcontrollers have on-chip flash memory. It is possible to program the controllers directly in the target environment where they are mounted.

With this feature, system development time can be reduced and system maintainability after shipping can be markedly improved.

(3) A full range of software development tools

A development system is available that includes an optimized C compiler, debugger, in-circuit emulator, simulator, system performance analyzer, and other elements.

1.2 Features Summary

The following table provides a quick summary of the most outstanding features.

Table 1-1 V850E/Dx3 - DG3 features summary (1/3)

CPU	
Core	V850E1
Number of instructions	81
Minimum instruction execution time	41.667 ns (@ ϕ = 24 MHz)
General registers	32 registers (32 bits each)
Instruction set	
V850E (compatible with V850 plus additional powerful instructions for reducing code and increasing execution speed)	
Signed multiplication (16 bits × 16 bits → 32 bits or 32 bits × 32 bits → 64 bits): 1 to 2 clocks	
Saturated operation instructions (with overflow/underflow detection) 32-bit shift instructions: 1 clock	
Bit manipulation instructions	
Load/store instructions with long/short format	
Signed load instructions	
Internal flash memory	
Size	• 256 KB (µPD70F3417) • 128 KB (µPD70F3416)
Flash protection	external programmer security function
Secure self programming	
Internal data RAM	
Size	• 12 KB (µPD70F3417) • 6 KB (µPD70F3416)

Table 1-1 V850E/Dx3 - DG3 features summary (2/3)

Clock Generator	
Internal spread-spectrum PLL	24 MHz ± 5 %
Internal PLL (peripheral clock supply)	8-fold PLL
CPU frequency range	up to 24 MHz
Peripheral frequency range	up to 16 MHz
Main crystal frequency range (main oscillator)	4 MHz
Sub oscillator	32 KHz (typ.)
Ring oscillator	240 KHz (typ.)
Clock supervision	2 channels: • main oscillator monitor • sub oscillator monitor
Auxiliary frequency output	
Built-in power saving modes	
HALT / IDLE / WATCH / Sub-WATCH / STOP	
I/O ports	
Input/output ports	72
Input ports	8
A/D Converter	
Number of channels	8
Resolution	10-bit
Conversion modes	• Continuous select mode • Continuous scan mode • Timer trigger mode • Software trigger mode
Analog input channels shared with digital input port functionality	
Serial interfaces	
Synchroneous: CSI (CSIB)	2 channels
Asynchroneous: UART (UARTA)	2 channels with LIN support
I^{2}C (IIC)	1 channel
CAN (CAN)	1 channel with 32 message buffer
Timers	
16-bit multi purpose timer/event counter (TMP)	1 channel
16-bit multi purpose timer/counter (TMG)	2 channel
16-bit multi purpose timer/counter (TMZ)	6 channels
Watch Timer (WT)	1 channel
Watch Calibration Timer (WCT)	1 channel
Watchdog Timer (WDT)	1 channel
LCD Controller/Driver	
Segment signal output	max. 40
Common signal output	max. 4
Modes	1/4 duty, 1/3 bias

Table 1-1 V850E/Dx3 - DG3 features summary (3/3)

Stepper Motor Controller/Driver	
Number of channels	4
Resolution	8-bit and 8-bit + 1
Sound Generator	
Number of channels	1
Volume	9-bit volume level accuracy
Sound frequency	100 Hz to 6 KHz with min. resolution of ± 20 Hz
Sound duration	256 steps
Interrupts and exceptions	
Non-maskable interrupts	2 sources
Maskable interrupts	51
Software exceptions	32 sources
Exception trap	2 sources
ROM Correction	
Number of channels	8 channels by “Data Replacement”
Power supply supervision	
Power-On-Clear	Generates reset at power-up and in case of power loss
Single supply operating voltage	
Range	3.5 V to 5.5 V (refer to Data Sheet)
Temperature range	
Range	T_a = −40 to +85°C (@ ϕ = 24 MHz)
Package	
Package	100-pin LQFP
Package size	14 mm × 14 mm
Pin pitch	0.5 mm
CMOS technology	

Note The CAN controller of this device fulfils the requirements according ISO 11898. Additionally, the CAN controller was tested according to the test procedures required by ISO 16845. The CAN controller has successfully passed all test patterns. Beyond these test patterns, other tests like robustness tests and processor interface tests as recommended by C&S/FH Wolfenbuettel have been performed with success.

1.3 Product Series Overview

Table 1-2 shows the common and different features of the microcontrollers.

An overview of the feature differences gives *Table 1-3*.

Table 1-2 V850E/Dx3 - DG3 product series overview

Part number		µPD70F3417	µPD70F3416
Internal memory	Flash	256 KB	128 KB
	RAM	12 KB	6 KB
Operating clock	Main oscillator with SSCG[a]	24 MHz typ., 25.2 MHz max.[b]	
	Ring oscillator	240 KHz typ.	
	Sub oscillator	32 KHz typ.	
I/O ports	Input/Output	72	
	Input	8	
A/D converter		8 channels	
Timers	TMZ	6 channels	
	TMP	1 channels	
	TMG	2 channels	
	WDT	1 channel	
	Watch	provided	
	Watch calibration	provided	
Serial interfaces	CAN	1 channel	
	UARTA	2 channels	
	CSIB	2 channels	
	I^2C	1 channel	
Interrupts	External	4 channels	
	Internal	51 channels	
	NMI	2 channels	
Other functions	ROM Correction	6 channels	
	Power-On-Clear	provided	
	Clock supervision	2 channels	
	Sound Generator	1 channel	
	Stepper Motor Controller/Driver	4 channels	
	LCD-Controller/Driver	40 x 4	
	Auxiliary frequency output	provided	
Operating voltage		3.5 V to 5.5 V[b]	
Package		100-pin LQFP, 0.5 mm pin pitch	

a) SSCG: spread spectrum Clock Generator

b) Refer to the Data Sheetcurrent data sheet (U19509EE)

1.4 Description

Figure 1-1 provides a functional block diagram of the V850E/DG3.

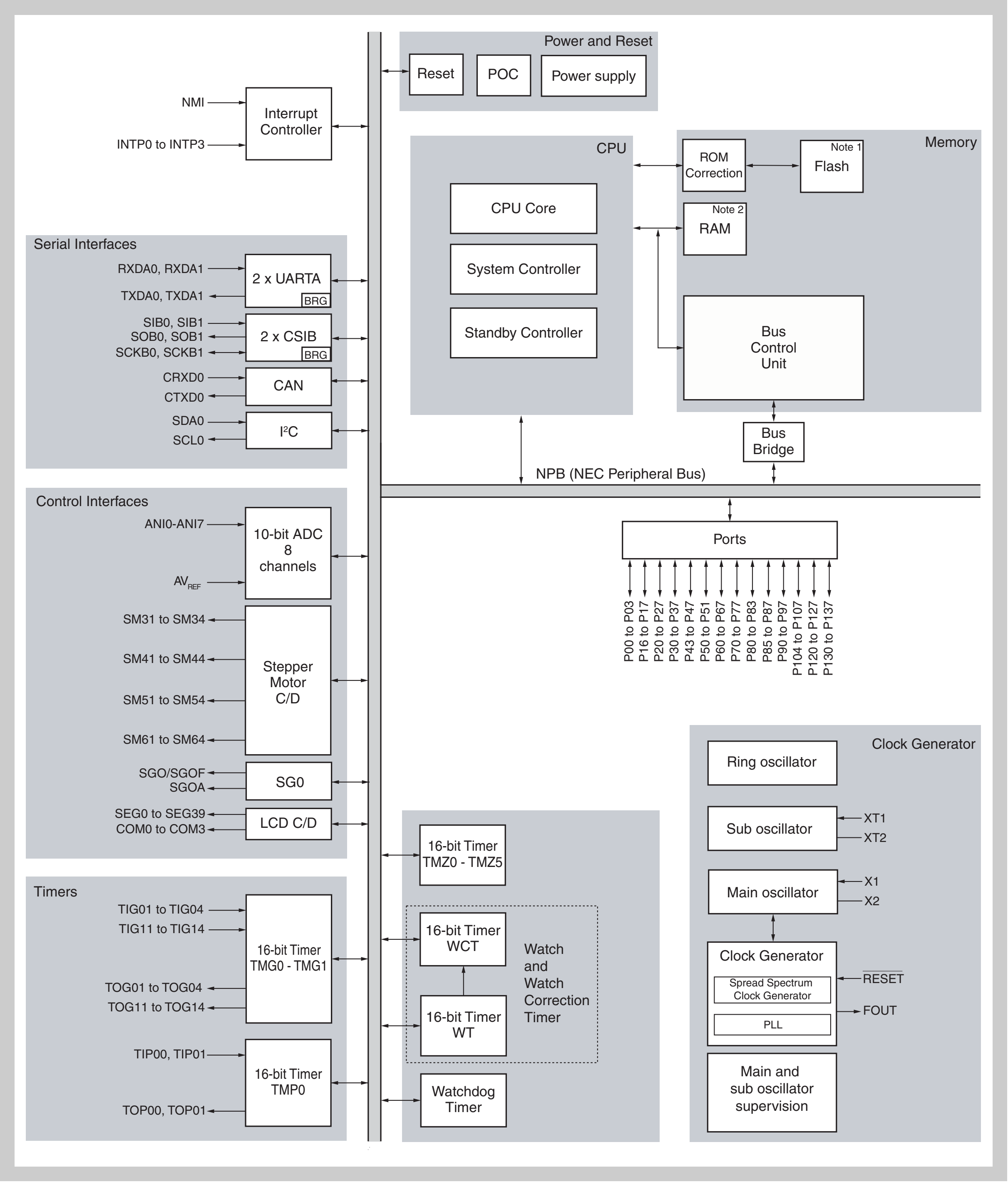


Figure 1-1 V850E/DG3 block diagram

Table 1-3 summarizes the different features of the of the V850E/DG3 µPD70F3417 and µPD70F3416 microcontrollers, marked as “Notes” in *Figure 1-1*.

Table 1-3 **Feature set differences**

Note	Feature	µPD70F3417	µPD70F3416
1	Flash	256 KB	128 KB
2	RAM	12 KB	6 KB

Structure of the diagram

In the diagram, the building blocks are grouped according to their function.

At the top of the diagram, you find the functions for controlling power supply and reset.

The upper right-hand section shows the building blocks of the CPU system and the memory interface components. The I/O ports are summarized below that section.

The left-hand section of the block diagram identifies the interfaces to peripherals and also the built-in timers. All these components are connected to and can be controlled via the internal bus.

The Clock Generator, depicted in the lower right-hand section, plays a central role. It generates and monitors not only the clocks for the CPU and the peripheral interfaces, but also governs the power save modes that can be entered when the device is not in use.

Structure of the manual

This manual explains how to use the V850E/Dx3 - DG3 microcontroller devices. It provides comprehensive information about the building blocks, their features, and how to set registers in order to enable or disable specific functions.

The manual provides individual chapters for the building blocks. These chapters are organized according to the grouping in the diagram.

- Core functions

  “Pin Functions“ on page 29
  “CPU System Functions“ on page 73
  “Clock Generator“ on page 100
  “Interrupt Controller (INTC)“ on page 180

- Memory access

  “Flash Memory“ on page 164
  “Bus Control Unit (BCU)“ on page 217
  “ROM Correction Function (ROMC)“ on page 226
  “Code Protection and Security“ on page 243

- Timers

  “16-bit Timer/Event Counter P (TMP)“ on page 246
  “16-bit Interval Timer Z (TMZ)“ on page 328
  “16-bit Multi-Purpose Timer G (TMG)“ on page 338
  “Watch Timer (WT)“ on page 376
  “Watchdog Timer (WDT)“ on page 395

- Serial interfaces

  “Asynchronous Serial Interface (UARTA)“ on page 404
  “Clocked Serial Interface (CSIB)“ on page 437

“I^{2}C Bus (IIC)“ on page 472
“CAN Controller (CAN)“ on page 546

- Control interfaces

  “A/D Converter (ADC)“ on page 685
  “Stepper Motor Controller/Driver (Stepper-C/D)“ on page 710
  “LCD Controller/Driver (LCD-C/D)“ on page 723
  “Sound Generator (SG)“ on page 736

- Power and reset

  “Power Supply Scheme“ on page 754
  “Reset“ on page 757

1.5 Related Documents

Table 1-4 Related documents

Document number	Title
U14559EJ3V1UM00	V850E1 32-Bit Microprocessor Core Architecture
R01DS0109EDxxxx	V850E/DG3 Data Sheet

1.6 Ordering Information

Table 1-5 V850E/DG3 ordering information

Order code	Pin/package	Memory size	Remarks
UPD70F3416GC(A)-UEU-QS-AX	100 pin LQFP	128 KB flash	–
UPD70F3417GC(A)-UEU-QS-AX	100 pin LQFP	256 KB flash	–

Chapter 2 Pin Functions

This chapter lists the ports of the microcontroller. It presents the configuration of the ports for alternative functions. Noise elimination on input signals is explained and a recommendation for the connection of unused pins is given at the end of the chapter.

2.1 Overview

The microcontroller offers various pins for input/output functions, so-called ports. The ports are organized in port groups.

To allocate other than general purpose input/output functions to the pins, several control registers are provided.

For a description of the terms pin, port or port group, see *“Terms“ on page 32*.

Features summary

- Number of ports and port groups:
  - Port groups: 13
  - I/O ports: 72
  - Input ports: 8
- 5V I/O:
  Can be used as 3V I/O with degraded electrical parameters. Please refer to the Data Sheet.
- 24 high-drive ports for direct stepper motor drive.
- Configuration possible for individual pins.
- The following features can be selected for most of the pins:
  - One out of two input thresholds
  - One out of two input characteristics (Schmitt and non-Schmitt)
  - Output current limit
  - Open drain emulation
- The following registers are offered for most of the ports:
  - Direct register for reading the pin values
  - Port register with selectable read source (for improved bit set / bit clear capabilities)

2.1.1 Description

This microcontroller has the port groups shown below.

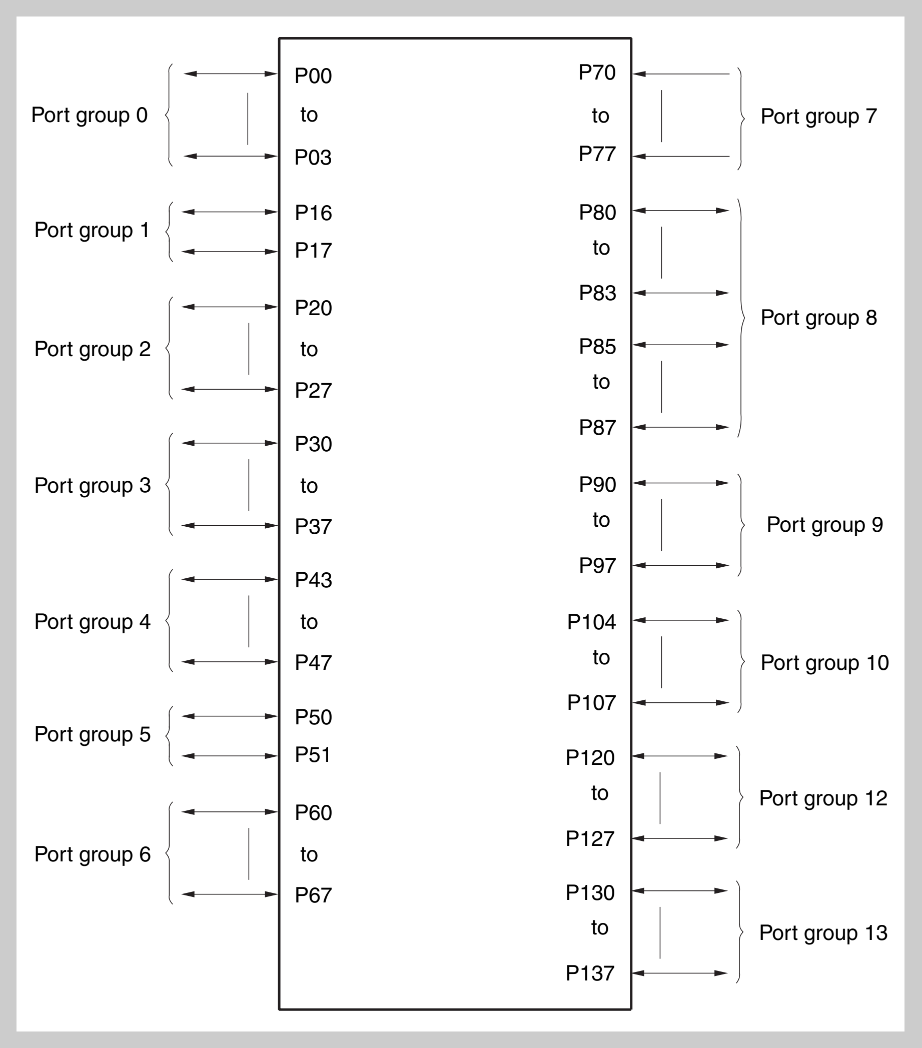


Figure 2-1 Port groups

Port group overview

Table 2-1 gives an overview of the port groups. For each port group it shows the supported functions in port mode and in alternative mode. Any port group can operate in 8-bit or 1-bit units.

Table 2-1 Functions of each port group

Port group name	Function	
	Port mode	**Alternative mode**
0	4-bit input/output	• External interrupt 0 to 3 • Non maskable interrupt
1	2-bit input/output	• I^{2}C0 data/clock line
2	8-bit input/output	• Timer TMG0 to TMG1 channels • LCD controller segment signal output
3	8-bit input/output	• Timer TMP0 channels • UARTA0 transmit/receive data, • UARTA1 transmit/receive data • LCD controller segment signal output
4	5-bit input/output	• Clocked Serial Interface CSIB1 data/clock line • LCD controller segment signal output • CAN0 transmit/receive data
5	2-bit input/output	• Sound Generator outputs • Frequency output
6	8-bit input/output	• Timer TMP0 channels • LCD controller segment signal output • I^{2}C0 data/clock line
7	8-bit input	• A/D Converter input
8	7-bit input/output	• LCD controller segment signal output • Frequency output • Inverted frequency output • UARTA0 transmit/receive data
9	8-bit input/output	• Clocked Serial Interface CSIB1 data/clock line • LCD controller segment signal output • LCD controller common signal output
10	4-bit input/output	• LCD controller segment signal output • Clocked Serial Interface CSIB0 data/clock line
12	8-bit input/output	• Stepper Motor Controller/Driver outputs
13	8-bit input/output	• Stepper Motor Controller/Driver outputs • Timer TMG0 to TMG1 channels

Pin configuration

To define the function and the electrical characteristics of a pin, several control registers are provided.

- For a general description of the registers, see *“Port Group Configuration Registers“ on page 33*.
- For every port, detailed information on the configuration registers is given in *“Port Group Configuration“ on page 46*.

There are three types of control circuits, defined as port types. For a description of the port types, see *“Port Types Diagrams“ on page 44*.

2.1.2 Terms

In this section, the following terms are used:

- **Pin**

  Denotes the physical pin. Every pin is uniquely denoted by its pin number.

  A pin can be used in several modes. Depending on the selected mode, a pin name is allocated to the pin.

- **Port group**

  Denotes a group of pins. The pins of a port group have a common set of port mode control registers.

- **Port mode / Port**

  A pin in port mode works as a general purpose input/output pin. It is then called “port”.

  The corresponding name is Pnm. For example, P07 denotes port 7 of port group 0. It is referenced as “port P07”.

- **Alternative mode**

  In alternative mode, a pin can work in various non-general purpose input/output functions, for example, as the input/output pin of on-chip peripherals.

  The corresponding pin name depends on the selected function. For example, pin INTP0 denotes the pin for one of the external interrupt inputs.

  Note that for example P00 and INTP0 denote the same physical pin. The different names indicate the function in which the pin is being operated.

- **Port type**

  A control circuit evaluates the settings of the configuration registers. There are different types of control circuits, called “port types”.

2.1.3 Noise elimination

The input signals at some pins are passing a filter to remove noise and glitches. The microcontroller supports both analog and digital filters. The analog filters are always applied to the input signals, whereas the digital filters can be enabled/disabled by control registers.

See *“Noise Elimination“ on page 66* for a detailed description.

2.2 Port Group Configuration Registers

This section starts with an overview of all configuration registers and then presents all registers in detail. The configuration registers are classified in the following groups:

- *“Pin function configuration“ on page 34*
- *“Pin data input/output“ on page 37*
- *“Configuration of electrical characteristics“ on page 39*
- *“Alternative input selection“ on page 42*

2.2.1 Overview

For the configuration of the individual pins of the port groups, the following registers are used:

Table 2-2 Registers for port group configuration

Register name	Shortcut	Function
Port mode register	PMn	Pin function configuration
Port mode control register	PMCn	
Port function control register	PFCn	
Port LCD control register	PLCDCn	
Port register	Pn	Pin data input/output
Port read control register	PRCn	
Port pin read register	PPRn	
Port drive strength control register	PDSCn	Configuration of electrical characteristics
Port input characteristic control register	PICCn	
Port input level control register	PILCn	
Port open drain control register	PODCn	
Peripheral function select register	PFSR0 to PFSR3	Alternative input selection

2.2.2 Pin function configuration

The registers for pin function configuration define the general function of a pin:

- input mode or output mode
- port mode or alternative mode
- selection of one of the alternative output functions ALT1-OUT/ALT2-OUT
- pin usage for LCD Controller/Driver output LCD_OUT

An overview of the register settings is given in the table below.

Table 2-3 Pin function configuration (overview)

Function	Registers				I/O
	PLCDC	PMC	PFC	PM	
Port mode (output)	0	0	X	0	O
Port mode (input)			X	1	I
Alternative output 1 mode		1	0	0	O
Alternative output 2 mode			1	0	O
Alternative input mode			X	1	I
LCD signal output (segment or common signal)	1	X	X	X	O

(1) PMn - Port mode register

The PMn register specifies whether the individual pins of the port group n are in input mode or in output mode.

For port groups with up to eight ports, this is an 8-bit register. For port groups with up to 16 ports, this is a 16-bit register.

Access This register can be read/written in 8-bit and 1-bit units.
16-bit registers can also be read/written in 16-bit units.

Address see *“Port Group Configuration“ on page 46*

Initial Value FF_H or $FFFF_H$. This register is initialized by any reset.

7	6	5	4	3	2	1	0
PMn7	PMn6	PMn5	PMn4	PMn3	PMn2	PMn1	PMn0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PMn15	PMn14	PMn13	PMn12	PMn11	PMn10	PMn9	PMn8	PMn7	PMn6	PMn5	PMn4	PMn3	PMn2	PMn1	PMn0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 2-4 PMn register contents

Bit position	Bit name	Function
7 to 0 or 15 to 0	PMn[7:0] or PMn[15:0]	Specifies input/output mode of the corresponding pin 0: Output mode (output enabled) 1: Input mode (output disabled)

(2) PMCn - Port mode control register

The PMCn register specifies whether the individual pins of port group n are in port mode or in alternative mode.

For port groups with up to eight ports, this is an 8-bit register. For port groups with up to 16 ports, this is a 16-bit register.

Access This register can be read/written in 8-bit and 1-bit units.
16-bit registers can also be read/written in 16-bit units.

Address see *“Port Group Configuration“ on page 46*

Initial Value 00_H or 0000_H. This register is initialized by any reset.

7	6	5	4	3	2	1	0
PMCn7	PMCn6	PMCn5	PMCn4	PMCn3	PMCn2	PMCn1	PMCn0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PMCn15	PMCn14	PMCn13	PMCn12	PMCn11	PMCn10	PMCn9	PMCn8	PMCn7	PMCn6	PMCn5	PMCn4	PMCn3	PMCn2	PMCn1	PMCn0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 2-5 PMCn register contents

Bit position	Bit name	Function
7 to 0 or 15 to 0	PMCn[7:0] or PMC[15:0]	Specifies the operation mode of the corresponding pin 0: Port mode 1: Alternative mode

(3) PFCn - Port function control register

If a pin is in alternative mode and serves as an output pin (PMn.PMnm = 0) some pins offer two output functions ALT1-OUT and ALT2-OUT.

The 8-bit PFCn register specifies which output function of a pin is to be used.

Access This register can be read/written in 8-bit and 1-bit units.

Address see *“Port Group Configuration“ on page 46*

Initial Value PFC0: 20_H

other PFCn: 00_H

This register is initialized by any reset.

7	6	5	4	3	2	1	0
PFCn7	PFCn6	PFCn5	PFCn4	PFCn3	PFCn2	PFCn1	PFCn0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 2-6 PFCn register contents

Bit position	Bit name	Function
7 to 0	PFCn[7:0]	Specifies the output function of the pin 0: Alternative output mode 1 (ALT1-OUT) 1: Alternative output mode 2 (ALT2-OUT) See *“Port Group Configuration“ on page 46* for a list of the possible output modes.

(4) PLCDCn - Port LCD control register

Some port groups comprise pins for signal output of the LCD Controller Driver. For those port groups, the 8-bit PLCDCn register specifies whether an individual pin of port group n serves as an output pin of the LCD Controller/ Driver or not.

Access This register can be read/written in 8-bit and 1-bit units.

Address see *“Port Group Configuration“ on page 46*

Initial Value 00_H. This register is initialized by any reset.

7	6	5	4	3	2	1	0
PLCDCn7	PLCDCn6	PLCDCn5	PLCDCn4	PLCDCn3	PLCDCn2	PLCDCn1	PLCDCn0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 2-7 PLCDCn register contents

Bit position	Bit name	Function
7 to 0	PLCDCn[7:0]	Enables LCD function of the pin: 0: Pin is not allocated to the LCD Controller/Driver. Pin function is specified in PMn, PMCn and PFCn 1: Pin serves as an output pin of the LCD Controller/ Driver. Data is output directly from buffers of the LCD Controller/Driver. Bit Pn.Pnm is neglected.

Note If PLCDCn.PLCDCnm = 1, the settings of the bits m in registers PMn, PMCn, and PFCn are neglected.

2.2.3 Pin data input/output

If a pin is in port mode, the registers for pin data input/output specify the input and output data.

(1) Pn - Port register

In port mode (PMCn.PMCnm=0), data is input from or output to an external device by writing or reading the Pn register.

For port groups with up to eight ports, this is an 8-bit register. For port groups with up to 16 ports, this is a 16-bit register.

Access This register can be read/written in 8-bit and 1-bit units.
16-bit registers can also be read/written in 16-bit units.

Address see *“Port Group Configuration“ on page 46*

Initial Value 00_H or 0000_H. This register is cleared by any reset.

Note After reset, the ports are in input mode (PMn.PMnm = 1). The read input value is determined by the port pins.

7	6	5	4	3	2	1	0
Pn7	Pn6	Pn5	Pn4	Pn3	Pn2	Pn1	Pn0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pn15	Pn14	Pn13	Pn12	Pn11	Pn10	Pn9	Pn8	Pn7	Pn6	Pn5	Pn4	Pn3	Pn2	Pn1	Pn0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 2-8 Pn register contents

Bit position	Bit name	Function
7 to 0 or 15 to 0	Pn[7:0] or Pn[15:0]	Data, see *Table 2-9* for details.

Note The value written to register Pn is retained until a new value is written to register Pn.

Data is written to or read from the Pn register as follows:

Table 2-9 Writing/reading register Pn

Function	PRC	PM	I/O
Write to Pn and output contents of Pn to pins	X	0	O
Write to Pn without affecting the pin status	X	1	I
Read from Pn and thus read the pin status	0	1	I
Read from Pn and disregard the pin status	X	0	O
	1	1	I

(2) PRCn - Port read control register

In input mode (PMn.PMnm = 1), the 8-bit PRCn register specifies whether the pin status or the contents of register Pn are read (see also *Table 2-9*). Each PRCn register contains only one control bit which defines the read source of all ports of the entire port group n.

Access This register can be read/written in 8-bit and 1-bit units.

Address see *“Port Group Configuration“ on page 46*

Initial Value 00_H. This register is cleared by any reset.

7	6	5	4	3	2	1	0
X	X	X	X	X	X	X	PRCn0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 2-10 PRCn register contents

Bit position	Bit name	Function
0	PRCn0	Specifies which data are to be read in port group n: 0: Pin status is read 1: Contents of Pn are read

Note If PMn.PMnm = 0, the contents of Pn are read in any case—independent of PRCn.PRCnm.

(3) PPRn - Port pin read register

The PPRn register reflects the actual pin value, independent of the control registers set-up.

For port groups with up to eight ports, this is an 8-bit register. For port groups with up to 16 ports, this is a 16-bit register.

Access This register is read-only, in 8-bit and 1-bit units.
16-bit registers can also be read in 16-bit units.

Address see *“Port Group Configuration“ on page 46*

Initial Value 00_H or 0000_H. This register is cleared by any reset.

7	6	5	4	3	2	1	0
PPRn7	PPRn6	PPRn5	PPRn4	PPRn3	PPRn2	PPRn1	PPRn0
R	R	R	R	R	R	R	R

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PPRn15	PPRn14	PPRn13	PPRn12	PPRn11	PPRn10	PPRn9	PPRn8	PPRn7	PPRn6	PPRn5	PPRn4	PPRn3	PPRn2	PPRn1	PPRn0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Table 2-11 PPRn register contents

Bit position	Bit name	Function
7 to 0 or 15 to 0	PPRn[7:0] or PPRn[15:0]	Actual pin value

2.2.4 Configuration of electrical characteristics

The registers for the configuration of electrical characteristics are briefly described in the following. For details refer to the Data Sheet.

(1) PDSCn - Port drive strength control register

The 8-bit PDSCn register selects the output current limiting function for high- or low-drive strength.

Access This register can be read/written, in 8-bit and 1-bit units.

Address see *“Port Group Configuration“ on page 46*

Initial Value 00_H. This register is cleared by any reset.

7	6	5	4	3	2	1	0
PDSCn7	PDSCn6	PDSCn5	PDSCn4	PDSCn3	PDSCn2	PDSCn1	PDSCn0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 2-12 PDSCn register contents

Bit position	Bit name	Function
7 to 0	PDSCn[7:0]	Specifies output current limiting function: 0: Limit 1. 1: Limit 2.

For the detailed specification of "Limit 1" and "Limit 2" refer to the Data Sheet.

(2) PICCn - Port input characteristic control register

The 8-bit PICCn register selects between Schmitt Trigger or non-Schmitt Trigger input characteristics.

Access This register can be read/written in 8-bit and 1-bit units.

Address see *“Port Group Configuration“ on page 46*

Initial Value FF_H. This register is cleared by any reset.

7	6	5	4	3	2	1	0
PICCn7	PICCn6	PICCn5	PICCn4	PICCn3	PICCn2	PICCn1	PICCn0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 2-13 PICCn register contents

Bit position	Bit name	Function
7 to 0	PICCn[7:0]	Specifies Trigger input characteristics: 0: non-Schmitt Trigger 1: Schmitt Trigger

(3) PILCn - Port input level control register

The 8-bit PILCn register selects between different input characteristics for Schmitt Trigger (PICCn.PICCnm = 1) and non-Schmitt Trigger (PICCn.PICCnm = 0).

Access This register can be read/written in 8-bit and 1-bit units.

Address see *“Port Group Configuration“ on page 46*

Initial Value 00_H

This register is initialized by any reset.

7	6	5	4	3	2	1	0
PILCn7	PILCn6	PILCn5	PILCn4	PILCn3	PILCn2	PILCn1	PILCn0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 2-14 PILCn register contents

Bit position	Bit name	Function
7 to 0	PILCn[7:0]	Selects the input level: for Schmitt Trigger (PICCn.PICCnm = 1): 0: Schmitt 1 1: Schmitt 2 for non-Schmitt Trigger (PICCn.PICCnm = 0): 0: CMOS1 1: CMOS2

(4) PODCn - Port open drain control register

The PODCn register selects the output buffer function as push-pull or open-drain emulation.

Access This register can be read/written in 8-bit and 1-bit units.

Address see *“Port Group Configuration“ on page 46*

Initial Value 00_H. This register is cleared by any reset.

7	6	5	4	3	2	1	0
PODCn7	PODCn6	PODCn5	PODCn4	PODCn3	PODCn2	PODCn1	PODCn0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 2-15 PODCn register contents

Bit position	Bit name	Function
7 to 0	PODCn[7:0]	Specifies the output buffer function: 0: push-pull 1: open drain emulation output mode

If open drain emulation is enabled the output function of the concerned pin is automatically enabled as well, independently of the PMn.PMnm setting.

Caution Depending on the capacitive load applied to an output pin Pnm (PMnm = 0) in open-drain emulation (PODCnm = 1) a change from low to high level may take a remarkable rise time.

Hence a read of the port pin status

- via the PPRn register or
- Pn register with PRCn = 0 (pin status read)

immediately after setting Pnm to high level may still return low level at Pnm.

Particular attention is needed when a read-modify-write instruction (SET1, CLR1, NOT1) is executed after setting Pnm = 1 (with PRCn0 = 0) to manipulate another port pin of the same port group n during the rise time of the Pnm output.
In this case the read of Pnm may show 0 (though it should be 1) and the 0 is written back to Pnm at the end of the read-modify-write instruction.
Consequently Pnm may never reach high level at the output pin.

2.2.5 Alternative input selection

Alternative input functions of CSIB1, UARTA0, I^{2}C0, and TMG0 are provided on two pins each. Thus you can select on which pin the alternative function should appear. For this purpose, four peripheral function select registers PFSRk (k = 0, 2, 3) are provided.

Note The selection of the alternative *input* function is done by a different circuit than the selection of the alternative *output* function. Therefore, the registers for selecting the alternative input functions (PFSR) are not reflected in the block diagrams of the port types in chapter *“Port Types Diagrams“ on page 44*.

(1) PFSR0 - Peripheral function select register

The 8-bit PFSR0 register selects the alternative input paths for the peripheral functions CSIB1 and I^{2}C0.

Access This register can be read/written in 8-bit units.

Address FFFF F720$_H$

Initial Value 01$_H$. This register is initialized by any reset.

7	6	5	4	3	2	1	0
0	0	0	PFSR04	0	0	PFSR01	0
R[a]	R[a]	R[a]	R/W	R[a]	R/W	R/W	R[a]

[a] zzThis bit may be written, but write is ignored.

Table 2-16 PFSR0 register contents

Bit position	Bit name	Function
4	PFSR04	Specifies the alternative input path for I^{2}C0: 0: SCL0 is input from P17 (SCL0_0) SDA0 is input from P16 (SDA0_0) 1: SCL0 is input from P64 (SCL0_1) SDA0 is input from P65 (SDA0_1)
1	PFSR01	Specifies the alternative input path for CSIB1: 0: SCKB1 is input from P45 (SCKB1_0) SIB1 is input from P43 (SIB1_0) 1: SCKB1 is input from P92 (SCKB1_1) SIB1 is input from P90 (SIB1_1)

(2) PFSR2 - Peripheral function select register

The 8-bit PFSR2 register selects the alternative input paths for the peripheral functions TMG0.

Access This register can be read/written in 8-bit units.

Address FFFF $F724_H$

Initial Value 01_H. This register is initialized by any reset.

7	6	5	4	3	2	1	0
0	0	0	0	PFSR23	PFSR22	PFSR21	PFSR20
R[a]	R[a]	R[a]	R[a]	R/W	R/W	R/W	R/W

[a] This bit may be written, but write is ignored.

Table 2-17 PFSR2 register contents

Bit position	Bit name	Function
3	PFSR23	Specifies the alternative input path for timer channel 4 of TMG0: 0: TIG04 is input from P23 (TIG04_0) 1: TIG04 is input from P133 (TIG04_1)
2	PFSR22	Specifies the alternative input path for timer channel 3 of TMG0: 0: TIG03 is input from P22 (TIG03_0) 1: TIG03 is input from P132 (TIG03_1)
1	PFSR21	Specifies the alternative input path for timer channel 2 of TMG0: 0: TIG02 is input from P21 (TIG02_0) 1: TIG02 is input from P131 (TIG02_1)
0	PFSR20	Specifies the alternative input path for timer channel 1 of TMG0: 0: TIG01 is input from P20 (TIG01_0) 1: TIG01 is input from P130 (TIG01_1)

(3) PFSR3 - Peripheral function select register

The 8-bit PFSR3 register selects the alternative input paths for the peripheral functions TMG2, UARTA0 and UARTA1.

Access This register can be read/written in 8-bit units.

Address FFFF $F726_H$

Initial Value 01_H. This register is initialized by any reset.

7	6	5	4	3	2	1	0
0	0		PFSR34				
R[a]	R[a]	R/W	R/W	R/W	R[a]	R[a]	R[a]

[a] These bits may be written, but write is ignored.

Table 2-18 PFSR3 register contents

Bit position	Bit name	Function
4	PFSR34	Specifies the alternative input path for UARTA0: 0: RXDA0 is input from P31 (RXDA0_0) 1: RXDA0 is input from P87 (RXDA0_1)

2.3 Port Types Diagrams

The control circuits that evaluate the settings of the configuration registers are of different types. This chapter presents the block diagrams of all port types.

(1) Port type M

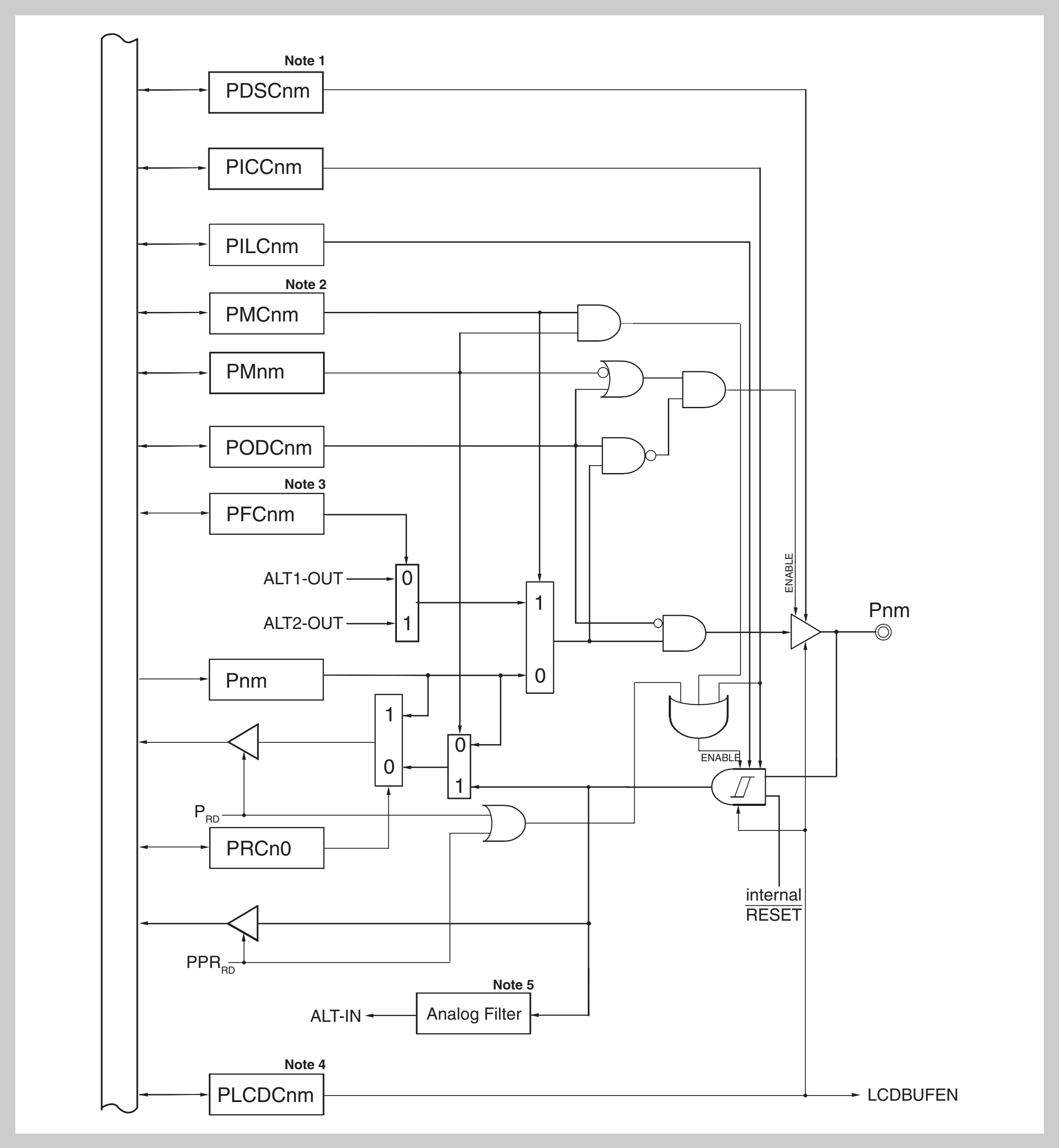


Figure 2-2 Block diagram: port type M

Note
1. The PDSC register is not provided for port groups 12 and 13.
2. The PMC register is not provided for port group 0.
3. The PFC register is not provided for port groups 0, 1, 3, 4, 6, 10 and 12.
4. The PLCDC register is not provided for port groups 0, 1, 4, 5, 12 and 13.
5. The analog filter is provided only for the external interrupt port group 0.

(2) Port type B

This port type holds for pins that only work in input mode. Pins of port type B are used for the corresponding alternative input function A/D converter input. At the same time, the pin status can also be read via the port register Pn, so that the pin also works in port function.

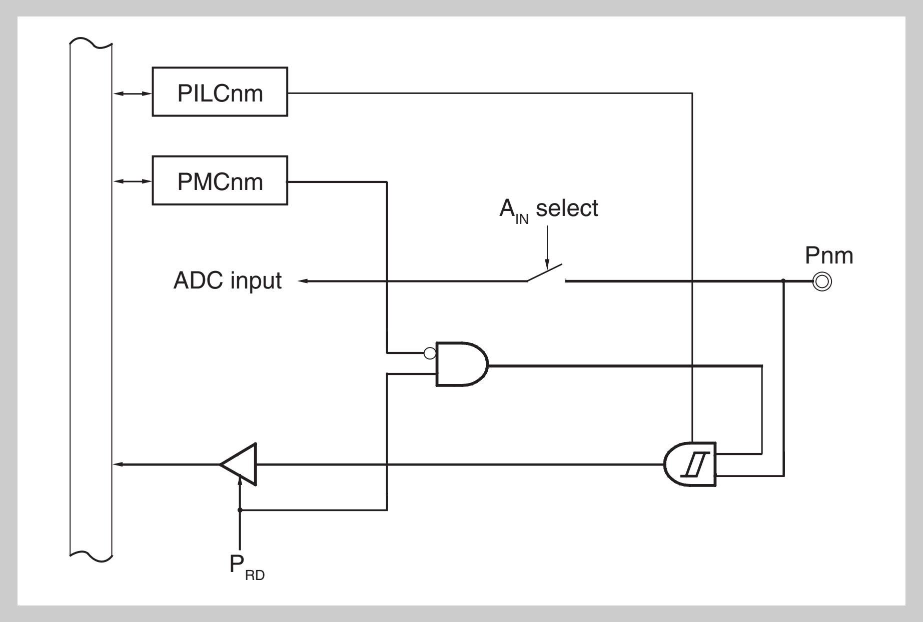


Figure 2-3 Block diagram: port type B

A/D conversion of the level at Pnm is independent of any register settings. For reading the pin status via the Pn register PMCnm has to be set to 0.

Since the accuracy of an A/D conversion may degrade when Pn is read during the sampling time of the A/D converter, it is recommended to disable the port pin read by PCMnm = 1 during A/D conversion.

2.4 Port Group Configuration

This section provides an overview of the port groups (*Table 2-19*) and of the pin functions (*Table 2-20 on page 48*). In *Table 2-53 on page 70* it is listed how the pin functions change if the microcontroller is reset or if it is in one of the standby modes.

In the subsections, for every port group the settings of the configuration registers is listed. Further, the addresses and initial values of the configuration registers are given.

2.4.1 Port group configuration lists

Table 2-19 provides an overview of the functions available at each port pin.

Table 2-19 Port group list (1/3)

Port group name	Port name	Alternative outputs ALT1_OUT/ALT2_OUT/ LCD_OUT	Alternative inputs	Port type
0	P00	–	INTP0/NMI	M
	P01	–	INTP1	M
	P02	–	INTP2	M
	P03	–	INTP3	M
1	P16	SDA0	SDA0	M
	P17	SCL0	SCL0	M
2	P20	TOG01/SEG0	TIG01	M
	P21	TOG02/SEG1	TIG02	M
	P22	TOG03/SEG2	TIG03	M
	P23	TOG04/SEG3	TIG04	M
	P24	TOG11/SEG4	TIG11	M
	P25	TOG12/SEG5	TIG12	M
	P26	TOG13/SEG6	TIG13	M
	P27	TOG14/SEG7	TIG14	M
3	P30	TXDA0	–	M
	P31	–	RXDA0	M
	P32	TXDA1/SEG31	–	M
	P33	SEG29	RXDA1	M
	P34	TOP01/SEG8	–	M
	P35	SEG9	–	M
	P36	SEG10	–	M
	P37	SEG11	–	M
4	P43	SEG22	SIB1	M
	P44	SOB1/SEG21	–	M
	P45	SCKB1/SEG20	SCKB1	M
	P46	–	CRXD0	M
	P47	CTXD0	–	M

Table 2-19 Port group list (2/3)

Port group name	Port name	Alternative outputs ALT1_OUT/ALT2_OUT/ LCD_OUT	Alternative inputs	Port type
5	P50	FOUT/SGOA	–	M
	P51	SGO	–	M
6	P60	TOP00/SEG12	TIP00	M
	P61	TOP01/SEG13	TIP01	M
	P62	SEG14	–	M
	P63	SEG15	–	M
	P64	SCL0/SEG16	SCL0	M
	P65	SDA0/SEG17	SDA0	M
	P66	SEG18	–	M
	P67	SEG19	–	M
7	P70	–	ANI0	B
	P71	–	ANI1	B
	P72	–	ANI2	B
	P73	–	ANI3	B
	P74	–	ANI4	B
	P75	–	ANI5	B
	P76	–	ANI6	B
	P77	–	ANI7	B
8	P80	SEG26	–	M
	P81	SEG25	–	M
	P82	SEG24	–	M
	P83	$\overline{\text{FOUT}}$/SEG23	–	M
	P85	FOUT/SEG27	–	M
	P86	TXDA0/SEG30	–	M
	P87	SEG28	RXDA0	M
9	P90	–	SIB1/SEG36	M
	P91	SOB1/SEG37	–	M
	P92	SCKB1/SEG38	SCKB1	M
	P93	SEG39	–	M
	P94	COM0	–	M
	P95	COM1	–	M
	P96	COM2	–	M
	P97	COM3	–	M
10	P104	SEG35	–	M
	P105	SEG34	SIB0	M
	P106	SOB0/SEG33	–	M
	P107	SCKB0/SEG32	SCKB0	M

Table 2-19 Port group list (3/3)

Port group name	Port name	Alternative outputs ALT1_OUT/ALT2_OUT/ LCD_OUT	Alternative inputs	Port type
12	P120	SM51	–	M
	P121	SM52	–	M
	P122	SM53	–	M
	P123	SM54	–	M
	P124	SM61	–	M
	P125	SM62	–	M
	P126	SM63	–	M
	P127	SM64	–	M
Note: Port group 12 is equipped with high drive buffers for stepper motor control.				
13	P130	SM31/TOG01	TIG01	M
	P131	SM32/TOG02	TIG02	M
	P132	SM33/TOG03	TIG03	M
	P133	SM34/TOG04	TIG04	M
	P134	SM41/TOG11	TIG11	M
	P135	SM42/TOG12	TIG12	M
	P136	SM43/TOG13	TIG13	M
	P137	SM44/TOG14	TIG14	M
Note: Port group 13 is equipped with high drive buffers for stepper motor control.				

2.4.2 Alphabetic pin function list

Table 2-20 provides a list of all pin function names in alphabetic order.

Table 2-20 Alphabetic pin functions list (1/5)

Pin name	I/O	Pin function	Port	Pin number	
				µPD703416 µPD703417	µPD70F3416 µPD70F3417
ANI0	I	A/D Converter input 0 to 7	P70	100	
ANI1			P71	99	
ANI2			P72	98	
ANI3			P73	97	
ANI4			P74	96	
ANI5			P75	95	
ANI6			P76	94	
ANI7			P77	93	
AVDD	–	ADC supply voltage	no ports	2	
AVREF	–	ADC reference voltage input	no ports	1	
AVSS	–	ADC ground	no ports	3	

Table 2-20 Alphabetic pin functions list (2/5)

Pin name	I/O	Pin function	Port	Pin number	
				µPD703416 µPD703417	µPD70F3416 µPD70F3417
BVDD50	–	I/O buffer supply voltage	no ports	41	
BVDD51				74	
BVSS50	–	I/O buffer supply ground	no ports	42	
BVSS51				75	
COM0	O	LCD common lines 0 to 3	P94	80	
COM1			P95	81	
COM2			P96	82	
COM3			P97	83	
CRXD0	I	CAN0 receive data	P46	30	
CTXD0	O	CAN0 transmit data	P47	31	
FLMD0	I	Primary operating mode select pin	no ports	87	
FLMD1	I	Secondary operating mode select pin	P50	–	28
FOUT	O	Frequency output	P50	28	
			P85	65	
$\overline{\text{FOUT}}$	O	Inverted frequency output	P83	61	
INTP0	I	External interrupts 0 to 3	P00	27	
INTP1			P01	26	
INTP2			P02	25	
INTP3			P03	24	
NMI	I	Non-maskable interrupt	P00	27	
REGC0	–	External capacitor connection	no ports	85	
$\overline{\text{RESET}}$	I	Reset input	no ports	90	
RXDA0	I	UARTA0 receive data	P31	32	
			P87	66	
RXDA1	I	UARTA1 receive data	P33	67	
SCKB0	I/O	Clocked Serial Interface CSIB0 clock line	P107	70	
SCKB1	I/O	Clocked Serial Interface CSIB1 clock line	P45	58	
			P92	78	
SCL0	I/O	I^{2}C0 clock line	P17	35	
			P64	54	
SDA0	I/O	I^{2}C0 data line	P16	34	
			P65	55	

Table 2-20 Alphabetic pin functions list (3/5)

Pin name	I/O	Pin function	Port	Pin number	
				µPD703416 µPD703417	µPD70F3416 µPD70F3417
SEG0	O	LCD segment lines 0 to 32	P20	36	
SEG1			P21	37	
SEG2			P22	38	
SEG3			P23	39	
SEG4			P24	40	
SEG5			P25	43	
SEG6			P26	44	
SEG7			P27	45	
SEG8			P34	46	
SEG9			P35	47	
SEG10			P36	48	
SEG11			P37	49	
SEG12			P60	50	
SEG13			P61	51	
SEG14			P62	52	
SEG15			P63	53	
SEG16			P64	54	
SEG17			P65	55	
SEG18			P66	56	
SEG19			P67	57	
SEG20			P45	58	
SEG21			P44	59	
SEG22			P43	60	
SEG23			P83	61	
SEG24			P82	62	
SEG25			P81	63	
SEG26			P80	64	
SEG27			P85	65	
SEG28			P87	66	
SEG29			P33	67	
SEG30			P86	68	
SEG31			P32	69	
SEG32			P107	70	
SEG33	O	LCD segment lines 33 to 39	P106	71	
SEG34			P105	72	
SEG35			P104	73	
SEG36			P90	76	
SEG37			P91	77	
SEG38			P92	78	
SEG39			P93	79	

Table 2-20 Alphabetic pin functions list (4/5)

Pin name	I/O	Pin function	Port	Pin number	
				µPD703416 µPD703417	µPD70F3416 µPD70F3417
SGO	O	Sound Generator output	P51	29	
SGOA	O	Sound Generator amplitude PWM output	P50	28	
SIB0	I	Clocked Serial Interface CSIB0 data input	P105	72	
SIB1	I	Clocked Serial Interface CSIB1 data input	P43	60	
			P90	76	
SM31	O	Stepper motor 3 output sin +	P130	14	
SM32	O	Stepper motor 3 output sin –	P131	15	
SM33	O	Stepper motor 3 output cos +	P132	16	
SM34	O	Stepper motor 3 output cos –	P133	17	
SM41	O	Stepper motor 4 output sin +	P134	20	
SM42	O	Stepper motor 4 output sin –	P135	21	
SM43	O	Stepper motor 4 output cos +	P136	22	
SM44	O	Stepper motor 4 output cos –	P137	23	
SM51	O	Stepper motor 5 output sin +	P120	4	
SM52	O	Stepper motor 5 output sin –	P121	5	
SM53	O	Stepper motor 5 output cos +	P122	6	
SM54	O	Stepper motor 5 output cos –	P123	7	
SM61	O	Stepper motor 6 output sin +	P124	10	
SM62	O	Stepper motor 6 output sin –	P125	11	
SM63	O	Stepper motor 6 output cos +	P126	12	
SM64	O	Stepper motor 6 output cos –	P127	13	
SMVDD50	–	Stepper Motor Controller/Driver supply voltage	no ports	8	
SMVDD51				18	
SMVSS50	–	Stepper Motor Controller/Driver ground	no ports	9	
SMVSS51				19	
SOB0	O	Clocked Serial Interface CSIB0 data output	P106	71	
SOB1	O	Clocked Serial Interface CSIB1 data output	P44	59	
			P91	77	
TIG01	I	Timer TMG0 channel 1 input	P20	36	
			P130	14	
TIG02	I	Timer TMG0 channel 2 input	P21	37	
			P131	15	
TIG03	I	Timer TMG0 channel 3 input	P22	38	
			P132	16	
TIG04	I	Timer TMG0 channel 4 input	P23	39	
			P133	17	
TIG11	I	Timer TMG1 channels 1 input	P24	40	
			P134	20	
TIG12	I	Timer TMG1 channels 2 input	P25	43	
			P135	21	

Table 2-20 Alphabetic pin functions list (5/5)

Pin name	I/O	Pin function	Port	Pin number	
				µPD703416 µPD703417	µPD70F3416 µPD70F3417
TIG13	I	Timer TMG1 channels 3 input	P26	44	
			P136	22	
TIG14	I	Timer TMG1 channels 4 input	P27	45	
			P137	23	
TIP00	I	Timer TMP0 channel 0 capture input	P60	50	
TIP01	I	Timer TMP0 channel 1 capture input	P61	51	
TOG01	O	Timer TMG0 channel 1 output	P20	36	
			P130	14	
TOG02	O	Timer TMG0 channel 2 output	P21	37	
			P131	15	
TOG03	O	Timer TMG0 channel 3 output	P22	38	
			P132	16	
TOG04	O	Timer TMG0 channel 4 output	P23	39	
			P133	17	
TOG11	O	Timer TMG1 channel 1 output	P24	40	
			P134	20	
TOG12	O	Timer TMG1 channel 2 output	P25	43	
			P135	21	
TOG13	O	Timer TMG1 channel 3 output	P26	44	
			P136	22	
TOG14	O	Timer TMG1 channel 4 output	P27	45	
			P137	23	
TOP00	O	Timer TMP0 channel 0 output	P60	50	
TOP01	O	Timer TMP0 channel 1 output	P34	46	
			P61	51	
TXDA0	O	UARTA0 transmit data	P30	33	
			P86	68	
TXDA1	O	UARTA1 transmit data	P32	69	
VDD50	–	Core supply voltage	no ports	84	
VSS50	–	Core supply ground	no ports	86	
X1	–	Main oscillator terminals	no ports	88	
X2				89	
XT1	–	Sub oscillator terminals	no ports	91	
XT2				92	

Note Alternative *input* functions of CSIB1, UART0, I^{2}C0, and TMG0 are provided on two pins each. Thus you can select on which pin the alternative function should appear.
Refer to *“Alternative input selection“ on page 42*.

2.4.3 Port group 0

- Port group 0 is an 4-bit port group. In alternative mode, it comprises pins for the following functions:
- External interrupt (INTP0 to INTP3)
- Non-maskable interrupt (NMI)

Port group 0 includes the following pins:

Table 2-21 Port group 0: pin functions and port types

Pin functions in different modes			Pin function after reset	Port type
Port mode (PMCnm = 0)	Alternative mode (PMCnm = 1)			
	output mode (PMnm = 0)	Input mode (PMnm = 1)		
P00 (I/O)	–	INTP0/NMI	P00 (I)	M
P01 (I/O)	–	INTP1	P01 (I)	M
P02 (I/O)	–	INTP2	P02 (I)	M
P03 (I/O)	–	INTP3	P03 (I)	M

Note For configuring P00 as NMI and/or INTP0 refer also to *“Edge and Level Detection Configuration“ on page 204*.

Table 2-22 Port group 0: configuration registers

Register	Address	Initial value	Used bits							
PM0	FFFF F420$_H$	FF$_H$	X	X	X	X	PM03	PM02	PM01	PM00
PMC0	FFFF F440$_H$	00$_H$	X	X	X	X	PMC03	PMC02	PMC01	PMC00
P0	FFFF F400$_H$	00$_H$	X	X	X	X	P03	P02	P01	P00
PRC0	FFFF F3E0$_H$	00$_H$	X	X	X	X	X	X	X	PRC00[a]
PPR0	FFFF F3C0$_H$	00$_H$	X	X	X	X	PPR03	PPR02	PPR01	PPR00
PDSC0	FFFF F300$_H$	00$_H$	X	X	X	X	PDSC03	PDSC02	PDSC01	PDSC00
PICC0	FFFF F380$_H$	FF$_H$	X	X	X	X	PICC03	PICC02	PICC01	PICC00
PILC0	FFFF F3A0$_H$	00$_H$	X	X	X	X	PILC03	PILC02	PILC01	PILC00
PODC0	FFFF F360$_H$	00$_H$	X	X	X	X	PODC03	PODC02	PODC01	PODC00

[a] The setting of PRC00 is valid for the entire port group.

Access All 8-bit registers can be accessed in 8-bit or 1-bit units.

2.4.4 Port group 1

Port group 1 is a 2-bit port group. In alternative mode, it comprises pins for the following functions:

- I^2C0 data/clock line (SDA0/SCL0)

Port group 1 includes the following pins:

Table 2-23 Port group 1: pin functions and port types

Pin functions in different modes			Pin function after reset	Port type
Port mode (PMCnm = 0)	Alternative mode (PMCnm = 1)			
	output mode (PMnm = 0)	Input mode (PMnm = 1)		
P16 (I/O)	SDA0[a]	SDA0	P16 (I)	M
P17 (I/O)	SCL0[a]	SCL0	P17 (I)	M

[a] In I^2C function mode open drain emulation has to be enabled (PODC1.PODC16 = 1 and PODC1.PODC17 = 1). Thus output function is enabled automatically, although PMnm = 1.

Note Alternative *input* functions SDA0 and SCL0 are provided on two pins each. Thus you can select on which pin the alternative function should appear. If alternative functions SDA0/SCL0 are used at P16/17 make sure to set also PFSR0.PFSR04 = 0.
Refer to *“Alternative input selection“ on page 42*.

Table 2-24 Port group 1: configuration registers

Register	Address	Initial value	Used bits							
PM1	FFFF $F422_H$	FF_H	PM17	PM16	X	X	X	X	X	X
PMC1	FFFF $F442_H$	00_H	PMC17	PMC16	X	X	X	X	X	X
P1	FFFF $F402_H$	00_H	P17	P16	X	X	X	X	X	X
PRC1	FFFF $F3E2_H$	00_H	X	X	X	X	X	X	X	PRC10[a] PRC1_0[b]
PPR1	FFFF $F3C2_H$	00_H	PPR17	PPR16	X	X	X	X	X	X
PDSC1	FFFF $F302_H$	00_H	PDSC17	PDSC16	X	X	X	X	X	X
PICC1	FFFF $F382_H$	FF_H	PICC17	PICC16	X	X	X	X	X	X
PILC1	FFFF $F3A2_H$	00_H	PILC17	PILC16	X	X	X	X	X	X
PODC1	FFFF $F362_H$	00_H	PODC17	PODC16	X	X	X	X	X	X

[a] The setting of PRC10/PRC1_0 is valid for the entire port group.
[b] Both bit names may be used.

Access All 8-bit registers can be accessed in 8-bit or 1-bit units.

2.4.5 Port group 2

Port group 2 is an 8-bit port group. In alternative mode, it comprises pins for the following functions:

- Timer TMG0 to TMG1 channels
  (TIG01 to TIG04, TOG01 to TOG04, TIG11 to TIG14, TOG11 to TOG14)
- LCD controller segment signal output (SEG0 to SEG7)

Port group 2 includes the following pins:

Table 2-25 Port group 2: pin functions and port types

Pin functions in different modes					Pin function after reset	Port type
Port mode (PMCnm = 0)	Alternative mode (PMCnm = 1)			LCD mode (PLCDCnm = 1)		
	Output mode (PMnm = 0)		Input mode (PMnm = 1)			
	PFCnm = 0 ALT1-OUT	PFCnm = 1 ALT2-OUT				
P20 (I/O)	–	TOG01	TIG01	SEG0	P20 (I)	M
P21 (I/O)	–	TOG02	TIG02	SEG1	P21 (I)	M
P22 (I/O)	TOG03		TIG03	SEG2	P22 (I)	M
P23 (I/O)	TOG04		TIG04	SEG3	P23 (I)	M
P24 (I/O)	TOG11		TIG11	SEG4	P24 (I)	M
P25 (I/O)	TOG12		TIG12	SEG5	P25 (I)	M
P26 (I/O)	TOG13		TIG13	SEG6	P26 (I)	M
P27 (I/O)	TOG14		TIG14	SEG7	P27 (I)	M

Note

1. For pins that support only one alternative output mode, the PFCnm bit is not available.
2. Alternative *input* functions of TMG0 are provided on two pins each. Thus you can select on which pin the alternative function should appear. Refer to *"Alternative input selection" on page 42*.

Table 2-26 Port group 2: Configuration registers

Register	Address	Initial value	Used bits							
PM2	FFFF F424$_H$	FF$_H$	PM27	PM26	PM25	PM24	PM23	PM22	PM21	PM20
PMC2	FFFF F444$_H$	00$_H$	PMC27	PMC26	PMC25	PMC24	PMC23	PMC22	PMC21	PMC20
PFC2	FFFF F464$_H$	00$_H$	X	X	X	X	X	X	PFC21	PFC20
PLCDC2	FFFF F344$_H$	00$_H$	PLCDC27	PLCDC26	PLCDC25	PLCDC24	PLCDC23	PLCDC22	PLCDC21	PLCDC20
P2	FFFF F404$_H$	00$_H$	P27	P26	P25	P24	P23	P22	P21	P20
PRC2	FFFF F3E4$_H$	00$_H$	X	X	X	X	X	X	X	PRC20[a]
PPR2	FFFF F3C4$_H$	00$_H$	PPR27	PPR26	PPR25	PPR24	PPR23	PPR22	PPR21	PPR20
PDSC2	FFFF F304$_H$	00$_H$	PDSC27	PDSC26	PDSC25	PDSC24	PDSC23	PDSC22	PDSC21	PDSC20
PICC2	FFFF F384$_H$	FF$_H$	PICC27	PICC26	PICC25	PICC24	PICC23	PICC22	PICC21	PICC20
PILC2	FFFF F3A4$_H$	00$_H$	PILC27	PILC26	PILC25	PILC24	PILC23	PILC22	PILC21	PILC20
PODC2	FFFF F364$_H$	00$_H$	PODC27	PODC26	PODC25	PODC24	PODC23	PODC22	PODC21	PODC20

[a] The setting of PRC20 is valid for the entire port group.

Access All 8-bit registers can be accessed in 8-bit or 1-bit units.

2.4.6 Port group 3

Port group 3 is an 8-bit port group. In alternative mode, it comprises pins for the following functions:

- UARTA0 transmit/receive data (TXDA0, RXDA0)
- UARTA1 transmit/receive data (TXDA1, RXDA1)
- LCD controller segment signal output (SEG8 to SEG11, SEG29, SEG31)
- Timer TMP0 channels (TOP01)

Port group 3 includes the following pins:

Table 2-27 Port group 3: pin functions and port types

Pin functions in different modes				Pin function after reset	Port type
Port mode (PMCnm = 0)	Alternative mode (PMCnm = 1)		LCD mode (PLCDCnm = 1)		
	Output mode (PMnm = 0)	Input mode (PMnm = 1)			
P30 (I/O)	TXDA0	–	–	P30 (I)	M
P31 (I/O)	–	RXDA0	–	P31 (I)	M
P32 (I/O)	TXDA1	–	SEG31	P32 (I)	M
P33 (I/O)	–	RXDA1	SEG29	P33 (I)	M
P34 (I/O)	TOP01	–	SEG8	P34 (I)	M
P35 (I/O)	–	–	SEG9	P35 (I)	M
P36 (I/O)	–	–	SEG10	P36 (I)	M
P37 (I/O)	–	–	SEG11	P37 (I)	M

Note Alternative *input* function RXDA0 of UARTA0 is provided on two pins. Thus you can select on which pin the alternative function should appear. Refer to *“Alternative input selection“ on page 42*.

Table 2-28 Port group 3: configuration registers

Register	Address	Initial value	Used bits							
PM3	FFFF F426$_{H}$	FF$_{H}$	PM37	PM36	PM35	PM34	PM33	PM32	PM31	PM30
PMC3	FFFF F446$_{H}$	00$_{H}$	X	X	X	PMC34	X	PMC32	X	PMC30
PLCDC3	FFFF F346$_{H}$	00$_{H}$	PLCDC37	PLCDC36	PLCDC35	PLCDC34	PLCDC33	PLCDC32	X	X
P3	FFFF F406$_{H}$	00$_{H}$	P37	P36	P35	P34	P33	P32	P31	P30
PRC3	FFFF F3E6$_{H}$	00$_{H}$	X	X	X	X	X	X	X	PRC30[a]
PPR3	FFFF F3C6$_{H}$	00$_{H}$	PPR37	PPR36	PPR35	PPR34	PPR33	PPR32	PPR31	PPR30
PDSC3	FFFF F306$_{H}$	00$_{H}$	PDSC37	PDSC36	PDSC35	PDSC34	PDSC33	PDSC32	PDSC31	PDSC30
PICC3	FFFF F386$_{H}$	FF$_{H}$	PICC37	PICC36	PICC35	PICC34	PICC33	PICC32	PICC31	PICC30
PILC3	FFFF F3A6$_{H}$	00$_{H}$	PILC37	PILC36	PILC35	PILC34	PILC33	PILC32	PILC31	PILC30
PODC3	FFFF F366$_{H}$	00$_{H}$	PODC37	PODC36	PODC35	PODC34	PODC33	PODC32	PODC31	PODC30

[a] The setting of PRC30 is valid for the entire port group.

Access All 8-bit registers can be accessed in 8-bit or 1-bit units.

2.4.7 Port group 4

Port group 4 is an 5-bit port group. In alternative mode, it comprises pins for the following functions:

- Clocked Serial Interface CSIB1 data/clock line (SIB1, SOB1, SCKB1)
- LCD controller segment signal output (SEG20 to SEG22)
- CAN0 transmit/receive data (CTXD0, CRXD0)

Port group 4 includes the following pins:

Table 2-29 Port group 4: pin functions and port types

Pin functions in different modes				Pin function after reset	Port type
Port mode (PMCnm = 0)	Alternative mode (PMCnm = 1)		LCD mode (PLCDCnm = 1)		
	Output mode (PMnm = 0)	Input mode (PMnm = 1)			
P43 (I/O)	–	SIB1	SEG22	P43 (I)	M
P44 (I/O)	SOB1	–	SEG21	P44 (I)	M
P45 (I/O)	SCKB1	SCKB1	SEG20	P45 (I)	M
P46 (I/O)	–	CRXD0	–	P46 (I)	M
P47 (I/O)	CTXD0	–	–	P47 (I)	M

Note Alternative *input* functions SIB1 and SCKB1of CSIB1 are provided on two pins each. Thus you can select on which pin the alternative function should appear. Refer to *“Alternative input selection“ on page 42*.

Table 2-30 Port group 4: configuration registers

Register	Address	Initial value	Used bits							
PM4	FFFF F428$_H$	FF$_H$	PM47	PM46	PM45	PM44	PM43	X	X	X
PMC4	FFFF F448$_H$	00$_H$	PMC47	PMC46	PMC45	PMC44	PMC43	X	X	X
PLCDC4	FFFF F348$_H$	00$_H$	X	X	PLCDC45	PLCDC44	PLCDC43	X	X	X
P4	FFFF F408$_H$	00$_H$	P47	P46	P45	P44	P43	X	X	X
PRC4	FFFF F3E8$_H$	00$_H$	X	X	X	X	X	X	X	PRC40
PPR4	FFFF F3C8$_H$	00$_H$	PPR47	PPR46	PPR45	PPR44	PPR43	X	X	X
PDSC4	FFFF F308$_H$	00$_H$	PDSC47	PDSC46	PDSC45	PDSC44	PDSC43	X	X	X
PICC4	FFFF F388$_H$	FF$_H$	PICC47	PICC46	PICC45	PICC44	PICC43	X	X	X
PILC4	FFFF F3A8$_H$	00$_H$	PILC47	PILC46	PILC45	PILC44	PILC43	X	X	X
PODC4	FFFF F368$_H$	00$_H$	PODC47	PODC46	PODC45	PODC44	PODC43	X	X	X

Note It is recommended to configure the ports used for CAN data transmit CTXDn to its highest drive strength to Limit2 by PDSCn.PDSCnm = 1 for CAN baud rates above 200 Kbit/sec.

Access All 8-bit registers can be accessed in 8-bit or 1-bit units.

2.4.8 Port group 5

Port group 5 is an 2-bit port group. In alternative mode, it comprises pins for the following functions:

- Sound Generator outputs (SGO, SGOA)
- Frequency output (FOUT)

Port group 5 includes the following pins:

Table 2-31 Port group 5: pin functions and port types

Pin functions in different modes				Pin function after reset	Port type
Port mode (PMCnm = 0)	**Alternative mode (PMCnm = 1)**				
	Output mode (PMnm = 0)		**Input mode (PMnm = 1)**		
	PFCnm = 0 ALT1-OUT	**PFCnm = 1 ALT2-OUT**			
P50 (I/O)	FOUT	SGOA	–	P50 (I)	M
P51 (I/O)	SGO		–	P51 (I)	M

Note 1. For pins that support only one alternative output mode, the PFCnm bit is not available.

Table 2-32 Port group 5: configuration registers

Register	Address	Initial value	Used bits							
PM5	FFFF F42A$_H$	FF$_H$	X	X	X	X	X	X	PM51	PM50
PMC5	FFFF F44A$_H$	00$_H$	X	X	X	X	X	X	PMC51	PMC50
PFC5	FFFF F46A$_H$	00$_H$	X	X	X	X	X	X	X	PFC50
P5	FFFF F40A$_H$	00$_H$	X	X	X	X	X	X	P51	P50
PRC5	FFFF F3EA$_H$	00$_H$	X	X	X	X	X	X	X	PRC50[a]
PPR5	FFFF F3CA$_H$	00$_H$	X	X	X	X	X	X	PPR51	PPR50
PDSC5	FFFF F30A$_H$	00$_H$	X	X	X	X	X	X	PDSC51	PDSC50
PICC5	FFFF F38A$_H$	FF$_H$	X	X	X	X	X	X	PICC51	PICC50
PILC5	FFFF F3AA$_H$	00$_H$	X	X	X	X	X	X	PILC51	PILC50
PODC5	FFFF F36A$_H$	00$_H$	X	X	X	X	X	X	PODC51	PODC50

[a] The setting of PRC50 is valid for the entire port group.

Access All 8-bit registers can be accessed in 8-bit or 1-bit units.

2.4.9 Port group 6

Port group 6 is an 8-bit port group. In alternative mode, it comprises pins for the following functions:

- Timer TMP0 channels (TIP00, TOP00, TOP01)
- LCD controller segment signal output (SEG12 to SEG19)
- I^{2}C0 data/clock line (SDA0, SCL0)

Port group 6 includes the following pins:

Table 2-33 Port group 6: pin functions and port types

Pin functions in different modes				Pin function after reset	Port type
Port mode (PMCnm = 0)	Alternative mode (PMCnm = 1)		LCD mode (PLCDCnm = 1)		
	Output mode (PMnm = 0)	Input mode (PMnm = 1)			
P60 (I/O)	TOP00	TIP00	SEG12	P60 (I)	M
P61 (I/O)	TOP01	TIP01	SEG13	P61 (I)	M
P62 (I/O)	–	–	SEG14	P62 (I)	M
P63 (I/O)	–	–	SEG15	P63 (I)	M
P64 (I/O)	SCL0[a]	SCL0	SEG16	P64 (I)	M
P65 (I/O)	SDA0[a]	SDA0	SEG17	P65 (I)	M
P66 (I/O)	–	–	SEG18	P66 (I)	M
P67 (I/O)	–	–	SEG19	P67 (I)	M

[a] In I^{2}C function mode open drain emulation has to be enabled (PODC6.PODC64 = 1 and PODC6.PODC65 = 1). Thus output function is enabled automatically, although PMnm = 1.

Note Alternative *input* functions SDA0 and SCL0 of I^{2}C0 are provided on two pins each. Thus you can select on which pin the alternative function should appear. If alternative functions SDA0/SCL0 are used at P64/65 make sure to set also PFSR0.PFSR04 = 1.
Refer to *"Alternative input selection" on page 42*.

Table 2-34 Port group 6: configuration registers

Register	Address	Initial value	Used bits							
PM6	FFFF F42C$_H$	FF$_H$	PM67	PM66	PM65	PM64	PM63	PM62	PM61	PM60
PMC6	FFFF F44C$_H$	00$_H$	X	X	PMC65	PMC64	X	X	PMC61	PMC60
PLCDC6	FFFF F34C$_H$	00$_H$	PLCDC67	PLCDC66	PLCDC65	PLCDC64	PLCDC63	PLCDC62	PLCDC61	PLCDC60
P6	FFFF F40C$_H$	00$_H$	P67	P66	P65	P64	P63	P62	P61	P60
PRC6	FFFF F3EC$_H$	00$_H$	X	X	X	X	X	X	X	PRC60[a]
PPR6	FFFF F3CC$_H$	00$_H$	PPR67	PPR66	PPR65	PPR64	PPR63	PPR62	PPR61	PPR60
PDSC6	FFFF F30C$_H$	00$_H$	PDSC67	PDSC66	PDSC65	PDSC64	PDSC63	PDSC62	PDSC61	PDSC60
PICC6	FFFF F38C$_H$	FF$_H$	PICC67	PICC66	PICC65	PICC64	PICC63	PICC62	PICC61	PICC60
PILC6	FFFF F3AC$_H$	00$_H$	PILC67	PILC66	PILC65	PILC64	PILC63	PILC62	PILC61	PILC60
PODC6	FFFF F36C$_H$	00$_H$	PODC67	PODC66	PODC65	PODC64	PODC63	PODC62	PODC61	PODC60

[a] The setting of PRC60 is valid for the entire port group.

Access All 8-bit registers can be accessed in 8-bit or 1-bit units.

2.4.10 Port group 7

Port group 7 is a 8-bit port group. It includes pins for the A/D Converter input.

The pins of this port group only work in input mode (port type B). They are used for their alternative input function A/D converter input. At the same time, the pin status can also be read via the port register Pn, so that the pin also works in port mode.

Port group 7 includes the following pins:

Table 2-35 Port group 7: pin functions and port types

Pin functions in different modes		Pin function after reset	Port type
Port mode (PMCnm = 0)	Alternative input mode (PMCnm = 1)		
P70 (I)	ANI0	P70 (I)	B
P71 (I)	ANI1	P71 (I)	B
P72 (I)	ANI2	P72 (I)	B
P73 (I)	ANI3	P73 (I)	B
P74 (I)	ANI4	P74 (I)	B
P75 (I)	ANI5	P75 (I)	B
P76 (I)	ANI6	P76 (I)	B
P77 (I)	ANI7	P77 (I)	B

Note All pins of port group 7 always function in alternative input mode, i.e. A/D conversion of the level at P7m is independent of any register settings.

For reading the pin status via the P7 register PMC7m has to be set to 0.

Since the accuracy of an A/D conversion may degrade when P7 is read during the sampling time of the A/D converter, it is recommended to disable the port pin read by PMC7m = 1 during A/D conversion.

Table 2-36 Port group 7: configuration registers

Register	Address	Initial value	Used bits							
PMC7	FFFF F44E$_H$	00$_H$	PMC77	PMC76	PMC75	PMC74	PMC73	PMC72	PMC71	PMC70
P7	FFFF F40E$_H$	00$_H$	P77	P76	P75	P74	P73	P72	P71	P70
PILC7	FFFF F3AE$_H$	00$_H$	PILC77	PILC76	PILC75	PILC74	PILC73	PILC72	PILC71	PILC70

Access All 8-bit registers can be accessed in 8-bit or 1-bit units.

2.4.11 Port group 8

Port group 8 is an 8-bit port group. In alternative mode, it comprises pins for the following functions:

- LCD controller segment signal output (SEG23 to SEG28, SEG30)
- Frequency output (FOUT)
- Inverted frequency output ($\overline{\text{FOUT}}$)
- UARTA0 transmit/receive data (TXDA0, RXDA0)

Port group 8 includes the following pins:

Table 2-37 Port group 8: pin functions and port types

Pin functions in different modes					Pin function after reset	Port type
Port mode (PMCnm = 0)	Alternative mode (PMCnm = 1)			LCD mode (PLCDCnm = 1)		
	Output mode (PMnm = 0)		Input mode (PMnm = 1)			
	PFCnm = 0 ALT1-OUT	PFCnm = 1 ALT2-OUT				
P80 (I/O)	–		–	SEG26	P80 (I)	M
P81 (I/O)	–		–	SEG25	P81 (I)	M
P82 (I/O)	–		–	SEG24	P82 (I)	M
P83 (I/O)	–	$\overline{\text{FOUT}}$	–	SEG23	P83 (I)	M
P85 (I/O)	FOUT		–	SEG27	P85 (I)	M
P86 (I/O)	TXDA0		–	SEG30	P86 (I)	M
P87 (I/O)	–		RXDA0	SEG28	P87 (I)	M

Note

1. For pins that support only one alternative output mode, the PFCnm bit is not available.
2. Alternative *input* functions of UART0 are provided on two pins each. Thus you can select on which pin the alternative function should appear. Refer to *“Alternative input selection“ on page 42*.

Table 2-38 Port group 8: configuration registers

Register	Address	Initial value	Used bits							
PM8	FFFF F430$_H$	FF$_H$	PM87	PM86	PM85	X	PM83	PM82	PM81	PM80
PMC8	FFFF F450$_H$	00$_H$	PMC87	PMC86	PMC85	X	PMC83	X	X	X
PFC8	FFFF F470$_H$	00$_H$	X	X	X	X	PFC83	X	X	X
PLCDC8	FFFF F350$_H$	00$_H$	PLCDC87	PLCDC86	PLCDC85	X	PLCDC83	PLCDC82	PLCDC81	PLCDC80
P8	FFFF F410$_H$	00$_H$	P87	P86	P85	X	P83	P82	P81	P80
PRC8	FFFF F3F0$_H$	00$_H$	X	X	X	X	X	X	X	PRC80[a]
PPR8	FFFF F3D0$_H$	00$_H$	PPR87	PPR86	PPR85	X	PPR83	PPR82	PPR81	PPR80
PDSC8	FFFF F310$_H$	00$_H$	PDSC87	PDSC86	PDSC85	X	PDSC83	PDSC82	PDSC81	PDSC80
PICC8	FFFF F390$_H$	FF$_H$	PICC87	PICC86	PICC85	X	PICC83	PICC82	PICC81	PICC80
PILC8	FFFF F3B0$_H$	00$_H$	PILC87	PILC86	PILC85	X	PILC83	PILC82	PILC81	PILC80
PODC8	FFFF F370$_H$	00$_H$	PODC87	PODC86	PODC85	X	PODC83	PODC82	PODC81	PODC80

[a] The setting of PRC80 is valid for the entire port group.

Access All 8-bit registers can be accessed in 8-bit or 1-bit units.

2.4.12 Port group 9

Port group 9 is an 8-bit port group. In alternative mode, it comprises pins for the following functions:

- Clocked Serial Interface CSIB1 data/clock line (SCKB1, SOB1, SIB1)
- LCD controller segment signal output (SEG36 to SEG39)
- LCD controller common signal output (COM0 to COM4)

Port group 9 includes the following pins:

Table 2-39 Port group 9: pin functions and port types

Pin functions in different modes					Pin function after reset	Port type
Port mode (PMCnm = 0)	Alternative mode (PMCnm = 1)			LCD mode (PLCDCnm = 1)		
	Output mode (PMnm = 0)		Input mode (PMnm = 1)			
	PFCnm = 0 ALT1-OUT	PFCnm = 1 ALT2-OUT				
P90 (I/O)	–		SIB1	SEG36	P90 (I)	M
P91 (I/O)	–	SOB1	–	SEG37	P91 (I)	M
P92 (I/O)	–	SCKB1	SCKB1	SEG38	P92 (I)	M
P93 (I/O)	–		–	SEG39	P93 (I)	M
P94 (I/O)	–		–	COM0	P94 (I)	M
P95 (I/O)	–		–	COM1	P95 (I)	M
P96 (I/O)	–		–	COM2	P96 (I)	M
P97 (I/O)	–		–	COM3	P97 (I)	M

Note 1. For pins that support only one alternative output mode, the PFCnm bit is not available.

2. Alternative *input* functions of CSIB1 are provided on two pins each. Thus you can select on which pin the alternative function should appear. Refer to *"Alternative input selection" on page 42*.

Table 2-40 Port group 9: configuration registers

Register	Address	Initial value	Used bits							
PM9	FFFF F432$_H$	FF$_H$	PM97	PM96	PM95	PM94	PM93	PM92	PM91	PM90
PMC9	FFFF F452$_H$	00$_H$	X	X	X	X	X	PMC92	PMC91	PMC90
PFC9	FFFF F472$_H$	00$_H$	X	X	X	X	X	PFC92	PFC91	X
PLCDC9	FFFF F352$_H$	00$_H$	PLCDC97	PLCDC96	PLCDC95	PLCDC94	PLCDC93	PLCDC92	PLCDC91	PLCDC90
P9	FFFF F412$_H$	00$_H$	P97	P96	P95	P94	P93	P92	P91	P90
PRC9	FFFF F320$_H$	00$_H$	X	X	X	X	X	X	X	PRC90[a]
PPR9	FFFF F3D2$_H$	00$_H$	PPR97	PPR96	PPR95	PPR94	PPR93	PPR92	PPR91	PPR90
PDSC9	FFFF F312$_H$	00$_H$	PDSC97	PDSC96	PDSC95	PDSC94	PDSC93	PDSC92	PDSC91	PDSC90
PICC9	FFFF F392$_H$	FF$_H$	PICC97	PICC96	PICC95	PICC94	PICC93	PICC92	PICC91	PICC90
PILC9	FFFF F3B2$_H$	00$_H$	PILC97	PILC96	PILC95	PILC94	PILC93	PILC92	PILC91	PILC90
PODC9	FFFF F372$_H$	00$_H$	PODC97	PODC96	PODC95	PODC94	PODC93	PODC92	PODC91	PODC90

[a] The setting of PRC90 is valid for the entire port group.

Access All 8-bit registers can be accessed in 8-bit or 1-bit units.

2.4.13 Port group 10

Port group 10 is an 4-bit port group. In alternative mode, it comprises pins for the following functions:

- LCD controller segment signal output (SEG32 to SEG35)
- Clocked Serial Interface CSIB0 data/clock line (SIB0, SOB0, SCKB0)

Port group 10 includes the following pins:

Table 2-41 Port group 10: pin functions and port types

Pin functions in different modes				Pin function after reset	Port type
Port mode (PMCnm = 0)	Alternative mode (PMCnm = 1)		LCD mode (PLCDCnm = 1)		
	Output mode (PMnm = 0)	Input mode (PMnm = 1)			
P104(I/O)	–	–	SEG35	P104(I)	M
P105 (I/O)	–	SIB0	SEG34	P105 (I)	M
P106 (I/O)	SOB0	–	SEG33	P106 (I)	M
P107 (I/O)	SCKB0	SCKB0	SEG32	P107 (I)	M

Table 2-42 Port group 10: configuration registers

Register	Address	Initial value	Used bits							
PM10	FFFF F434$_H$	FF$_H$	PM107	PM106	PM105	PM104	X	X	X	X
PMC10	FFFF F454$_H$	00$_H$	PMC107	PMC106	PMC105	X	X	X	X	X
PLCDC10	FFFF F354$_H$	00$_H$	PLCDC107	PLCDC106	PLCDC105	PLCDC104	X	X	X	X
P10	FFFF F414$_H$	00$_H$	P107	P106	P105	P104	X	X	X	X
PRC10	FFFF F3F4$_H$	00$_H$	X	X	X	X	X	X	X	PRC100[a]
PPR10	FFFF F3D4$_H$	00$_H$	PPR107	PPR106	PPR105	PPR104	X	X	X	X
PDSC10	FFFF F314$_H$	00$_H$	PDSC107	PDSC106	PDSC105	PDSC104	X	X	X	X
PICC10	FFFF F394$_H$	FF$_H$	PICC107	PICC106	PICC105	PICC104	X	X	X	X
PILC10	FFFF F3B4$_H$	00$_H$	PILC107	PILC106	PILC105	PILC104	X	X	X	X
PODC10	FFFF F374$_H$	00$_H$	PODC107	PODC106	PODC105	PODC104	X	X	X	X

[a] The setting of PRC100 is valid for the entire port group.

Access All 8-bit registers can be accessed in 8-bit or 1-bit units.

2.4.14 Port group 12

Port group 12 is an 8-bit port group. In alternative mode, it comprises pins for the following functions:

- Stepper Motor Controller/Driver outputs (SM51 to SM54, SM61 to SM64)

Port group 12 includes the following pins:

Table 2-43 Port group 12: pin functions and port types

Pin functions in different modes			Pin function after reset	Port type
Port mode (PMCnm = 0)	Alternative mode (PMCnm = 1)			
	Output mode (PMnm = 0)	Input mode (PMnm = 1)		
P120 (I/O)	SM51	–	P120 (I)	M
P121 (I/O)	SM52	–	P121 (I)	M
P122 (I/O)	SM53	–	P122 (I)	M
P123 (I/O)	SM54	–	P123 (I)	M
P124 (I/O)	SM61	–	P124 (I)	M
P125 (I/O)	SM62	–	P125 (I)	M
P126 (I/O)	SM63	–	P126 (I)	M
P127 (I/O)	SM64	–	P127 (I)	M

Note Port group 12 is equipped with high driver buffers for stepper motor control.

Table 2-44 Port group 12: configuration registers

Register	Address	Initial value	Used bits							
PM12[a]	FFFF F438$_H$	FF$_H$	PM127	PM126	PM125	PM124	PM123	PM122	PM121	PM120
PMC12	FFFF F458$_H$	00$_H$	PMC127	PMC126	PMC125	PMC124	PMC123	PMC122	PMC121	PMC120
P12	FFFF F418$_H$	00$_H$	P127	P126	P125	P124	P123	P122	P121	P120
PRC12	FFFF F3F8$_H$	00$_H$	X	X	X	X	X	X	X	PRC120[b]
PPR12	FFFF F3D8$_H$	00$_H$	PPR127	PPR126	PPR125	PPR124	PPR123	PPR122	PPR121	PPR120
PICC12	FFFF F398$_H$	FF$_H$	PICC127	PICC126	PICC125	PICC124	PICC123	PICC122	PICC121	PICC120
PILC12	FFFF F3B8$_H$	00$_H$	PILC127	PILC126	PILC125	PILC124	PILC123	PILC122	PILC121	PILC120
PODC12	FFFF F378$_H$	00$_H$	PODC127	PODC126	PODC125	PODC124	PODC123	PODC122	PODC121	PODC120

[a] PM12 has to be changed from its default value FF$_H$ to 00$_H$ in order to enable the Stepper Motor Controller/ Driver outputs

[b] The setting of PRC120 is valid for the entire port group.

Access All 8-bit registers can be accessed in 8-bit or 1-bit units.

2.4.15 Port group 13

Port group 13 is an 8-bit port group. In alternative mode, it comprises pins for the following functions:

- Stepper Motor Controller/Driver outputs (SM31 to SM34, SM41 to SM44)
- Timer TMG0 to TMG1 channels (TIG01 to TIG04, TOG01 to TOG04, TIG11 to TIG14, TOG11 to TOG14)

Port group 13 includes the following pins:

Table 2-45 Port group 13: pin functions and port types

Pin functions in different modes				Pin function after reset	Port type
Port mode (PMCnm = 0)	Alternative mode (PMCnm = 1)				
	output mode (PMnm = 0)		Input mode (PMnm = 1)		
	PFCnm = 0 ALT1-OUT	PFCnm = 1 ALT2-OUT			
P130 (I/O)	SM31	TOG01	TIG01	P130 (I)	M
P131 (I/O)	SM32	TOG02	TIG02	P131 (I)	M
P132 (I/O)	SM33	TOG03	TIG03	P132 (I)	M
P133 (I/O)	SM34	TOG04	TIG04	P133 (I)	M
P134 (I/O)	SM41	TOG11	TIG11	P134 (I)	M
P135 (I/O)	SM42	TOG12	TIG12	P135 (I)	M
P136 (I/O)	SM43	TOG13	TIG13	P136 (I)	M
P137 (I/O)	SM44	TOG14	TIG14	P137 (I)	M

Note

1. Alternative *input* functions of TMG0 are provided on two pins each. Thus you can select on which pin the alternative function should appear. Refer to *“Alternative input selection“ on page 42*.
2. Port group 13 is equipped with high driver buffers for stepper motor control.

Table 2-46 Port group 13: configuration registers

Register	Address	Initial value	Used bits							
PM13	FFFF F43A$_H$	FF$_H$	PM137	PM136	PM135	PM134	PM133	PM132	PM131	PM130
PMC13	FFFF F45A$_H$	00$_H$	PMC137	PMC136	PMC135	PMC134	PMC133	PMC132	PMC131	PMC130
PFC13	FFFF F47A$_H$	00$_H$	PFC137	PFC136	PFC135	PFC134	PFC133	PFC132	PFC131	PFC130
P13	FFFF F41A$_H$	00$_H$	P137	P136	P135	P134	P133	P132	P131	P130
PRC13	FFFF F3FA$_H$	00$_H$	X	X	X	X	X	X	X	PRC130[a]
PPR13	FFFF F3DA$_H$	00$_H$	PPR137	PPR136	PPR135	PPR134	PPR133	PPR132	PPR131	PPR130
PICC13	FFFF F39A$_H$	FF$_H$	PICC137	PICC136	PICC135	PICC134	PICC133	PICC132	PICC131	PICC130
PILC13	FFFF F3BA$_H$	00$_H$	PILC137	PILC136	PILC135	PILC134	PILC133	PILC132	PILC131	PILC130
PODC13	FFFF F37A$_H$	00$_H$	PODC137	PODC136	PODC135	PODC134	PODC133	PODC132	PODC131	PODC130

[a] The setting of PRC130 is valid for the entire port group.

Access All 8-bit registers can be accessed in 8-bit or 1-bit units.

2.5 Noise Elimination

The input signals at some pins are passed through a filter to remove noise and glitches. The microcontroller supports both analog and digital filters. The analog filters are always applied to the input signals, whereas the digital filters can be enabled/disabled by control registers.

2.5.1 Analog filtered inputs

The external interrupts INTPn, NMI and the external $\overline{\text{RESET}}$ input are passed through an analog filter to remove noise and glitches. The analog filter suppresses input pulses that are shorter than a specified puls width (refer to the Data Sheet). This assures the hold time for the external interrupt signals.

The analog filter operates in all modes (normal mode and standby modes). It is only effective if the corresponding pin works in alternative input mode and not as a general purpose I/O port.

2.5.2 Digitally filtered inputs

The inputs of the peripherals listed below are passed through a digital filter to remove noise and glitches.

The digital filter operates in all modes, which have the PLL enabled. Thus, it does not operate in Watch, Sub-watch and Idle mode. The digital filter is only effective if the corresponding pin works in alternative input mode and not as a general purpose I/O port.

The digital input filter is available for the following external signals:

Table 2-47 Digitally filtered external signals

Module	Signal	Comment
CSIB0	SIB0, SCKB0	For high clock rates of the Clocked Serial Interface, the digital filter should be disabled. Otherwise, desired input pulses may be removed by the digital filter.
CSIB1	SIB1, SCKB1	
TMP0	TIP00, TIP01	
TMG0	TIG01 to TIG04	
TMG1	TIG11 to TIG14	

Note The Timers G provide additional digital noise filters at their capture inputs TIGn1 to TIGn4. Refer also to the Data Sheet for the minimum capture inputs pulse widths.

Filter operation The input terminal signal is sampled with the sampling frequency f_S. Spikes shorter than 2 sampling cycles are suppressed and no internal signal is generated. Pulses longer than 3 sampling cycles are recognized as valid pulses and an internal signal is generated. For pulses between 2 and 3 sampling cycles, the behaviour is not defined. The filter operation is illustrated in *Figure 2-4*.

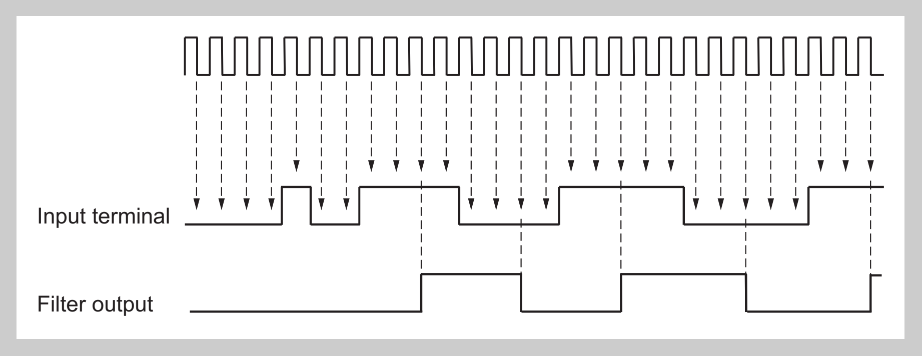


Figure 2-4 Digital noise removal example

The minimum input terminal pulse width to be validated is defined by the sampling frequency f_S. The sampling frequency f_S is PCLK0.

Table 2-48 Digital noise removal features

Sampling frequency **f_S = PCLK0**	**Minimum pulse width to generate an internal signal**
16 MHz (PLL enabled)	0.125 – 0.1875 µsec
4 MHz (PLL disabled)	0.5 – 0.75 µsec

The digital filter function can be individually enabled for each of the aforementioned external input signals. The filter is enabled/disabled by the 16-bit registers DFEN0 and DFEN1.

(1) DFEN0 - Digital filter enable register

The 16-bit DFEN0 register enables/disables the digital filter for TMP0 to TMP3 and TMG0 input channels and for CSIB0 to CSIB2 input channels.

Access This register can be read/written in 16-bit, 8-bit and 1-bit units.

Address FFFF F710$_H$

Initial Value 0000$_H$. This register is cleared by any reset.

15	14	13	12	11	10	9	8
DFENC15	DFENC14						
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

7	6	5	4	3	2	1	0
DFENC7	DFENC6		DFENC4	DFENC3		DFENC1	DFENC0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 2-49 DFEN0 register contents

Bit position	Bit name	Function
15 to 0	DFENC[15:0]	Enables/disables the digital noise elimination filter for the corresponding input signal: 0: Digital filter is disabled. 1: Digital filter is enabled. For an assignment of bit positions to input signals see table *Table 2-50*.

Table 2-50 Assignment of input signals to bit positions for register DFEN0

Bit position	Bit name	Input signal	Description
0	DFENC0	SIB0	CSIB0 data input[a]
1	DFENC1	SIB1	CSIB1 data input[a]
3	DFENC3	SCKIB0	CSIB0 clock input[a]
4	DFENC4	SCKIB1	CSIB1 clock input[a]
6	DFENC6	TIP00	Timer TMP0 channel 0 capture input
7	DFENC7	TIP01	Timer TMP0 channel 1 capture input
13	DFENC13	TIP31	Timer TMP3 channel 1 capture input
14	DFENC14	TIG01	Timer TMG0 channel 1 capture input
15	DFENC15	TIG02	Timer TMG0 channel 2 capture input

[a] Note that for high clock rates of the Clocked Serial Interface, the digital filter should be disabled. Otherwise, desired input pulses may be removed by the digital filter.

(2) DFEN1 - Digital filter enable register

The 16-bit DFEN1 register enables/disables the digital filter for TMG0, TMG1 and TMP0 input channels.

Access This register can be read/written in 16-bit, 8-bit and 1-bit units.

Address FFFF F712$_\mathrm{H}$

Initial Value 0000$_\mathrm{H}$. This register is cleared by any reset.

15	14	13	12	11	10	9	8
X	X	X	X	DFENC27			
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

7	6	5	4	3	2	1	0
	DFENC22	DFENC21	DFENC20	DFENC19	DFENC18	DFENC17	DFENC16
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 2-51 DFEN1 register contents

Bit position	Bit name	Function
11 to 0	DFENC[27:16]	Enables/disables the digital noise elimination filter for the corresponding input signal: 0: Digital filter is disabled. 1: Digital filter is enabled. For an assignment of bit positions to input signals see table *Table 2-52*.

Table 2-52 Assignment of input signals to bit positions for register DFEN1

Bit position	Bit name	Input signal	Description
0	DFENC16	TIG03	Timer TMG0 channel 3 capture input
1	DFENC17	TIG04	Timer TMG0 channel 4 capture input
2	DFENC18	TIG11	Timer TMG1 channel 1 capture input
3	DFENC19	TIG12	Timer TMG1 channel 2 capture input
4	DFENC20	TIG13	Timer TMG1 channel 3 capture input
5	DFENC21	TIG14	Timer TMG1 channel 4 capture input
6	DFENC22	TIP00	Timer TMP0 channel 0 capture input
11	DFENC27	TIP10	Timer TMP1 channel 0 capture input

2.6 Pin Functions in Reset and Power Save Modes

The following table summarizes the status of the pins during reset and power save modes and after release of these operating states in normal operation mode, i.e. FLMD0 = 0.

In contrast to all other power save modes the HALT mode suspends only the CPU operation and has no effect on any pin status.

Table 2-53 Pin functions during and after reset / power save modes

Operating status		Pin status
Power-On-Clear and any reset	during	Hi-Z (3-state)
	after	input port mode
HALT mode	during	same as before HALT mode
	after	
IDLE, WATCH, Sub-WATCH, STOP mode	during	same as before power save mode: • Output signals are valid and output levels are remained. • Input signals with wake-up capability[a] are valid. • Input signals without wake-up capability are ignored.
	after	same as before power save mode

[a] Inputs with wake-up capability: external interrupts (INTPn, NMI) and CANn receive data (CRXDn)

2.7 Recommended Connection of unused Pins

If a pin is not used, it is recommended to connect it as follows:

- output pins: leave open
- input pins: connect to V_{DD5} or V_{SS5}

Sub oscillator connection If no sub oscillator crystal is connected , connect XT1 to V_{SS} and leave XT2 open.

Note If the overall maximum output current of a concerned pin group exceeds its maximum value the output buffer can be damaged. We recommend the placement of a series resistor to prevent damage in case of accidentally enabled outputs. Refer to the absolute maximum rating parameter in the Data Sheet.

2.8 Package Pins Assignment

The following sections show the location of pins in top view. Every pin is labelled with its pin number and all possible pin names.

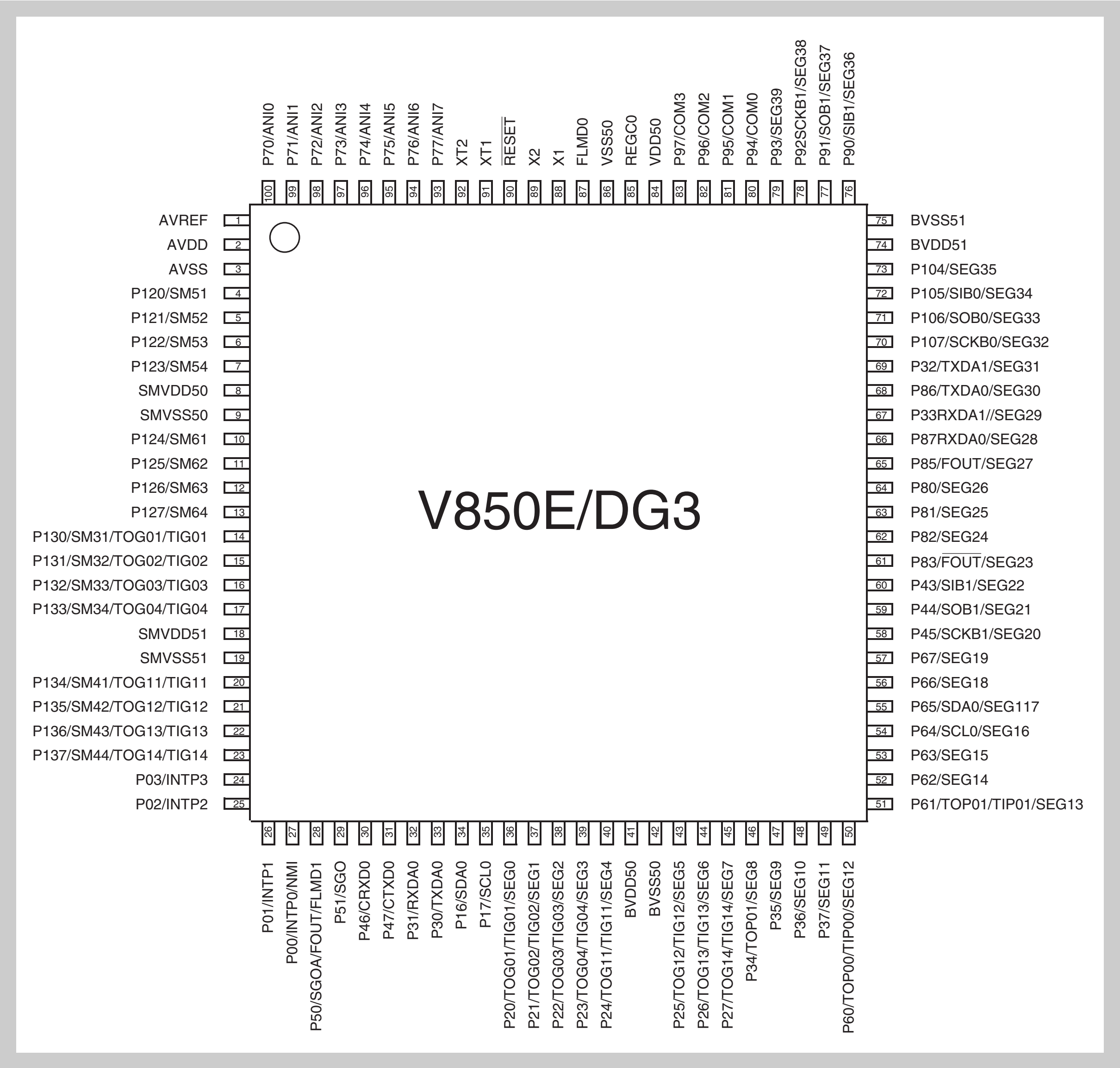


Figure 2-5 Pin overview

Chapter 3 CPU System Functions

This chapter describes the registers of the CPU, the operation modes, the address space and the memory areas.

3.1 Overview

The CPU is founded on Harvard architecture and it supports a RISC instruction set. Basic instructions can be executed in one clock period. Optimized five-stage pipelining is supported. This improves instruction execution speed.

In order to make the microcontroller ideal for use in digital control applications, a 32-bit hardware multiplier enables this CPU to support multiply instructions, saturated multiply instructions, bit operation instructions, etc.

Features summary The CPU has the following special features:

- Memory space:
  - 64 MB linear program space
  - 4 GB linear data space
- 32 general purpose registers
- Internal 32-bit architecture
- Five-stage pipeline
- Efficient multiplication and division instructions
- Saturation logic (saturated operation instructions)
- Barrel shifter (32-bit shift in one clock cycle)
- Instruction formats: long and short
- Four types of bit manipulation instructions: set, clear, not, test

3.1.1 Description

The figure below shows a block diagram of the microcontroller, focusing on the CPU and modules that interact with the CPU directly. *Table 3-1* lists the bus types.

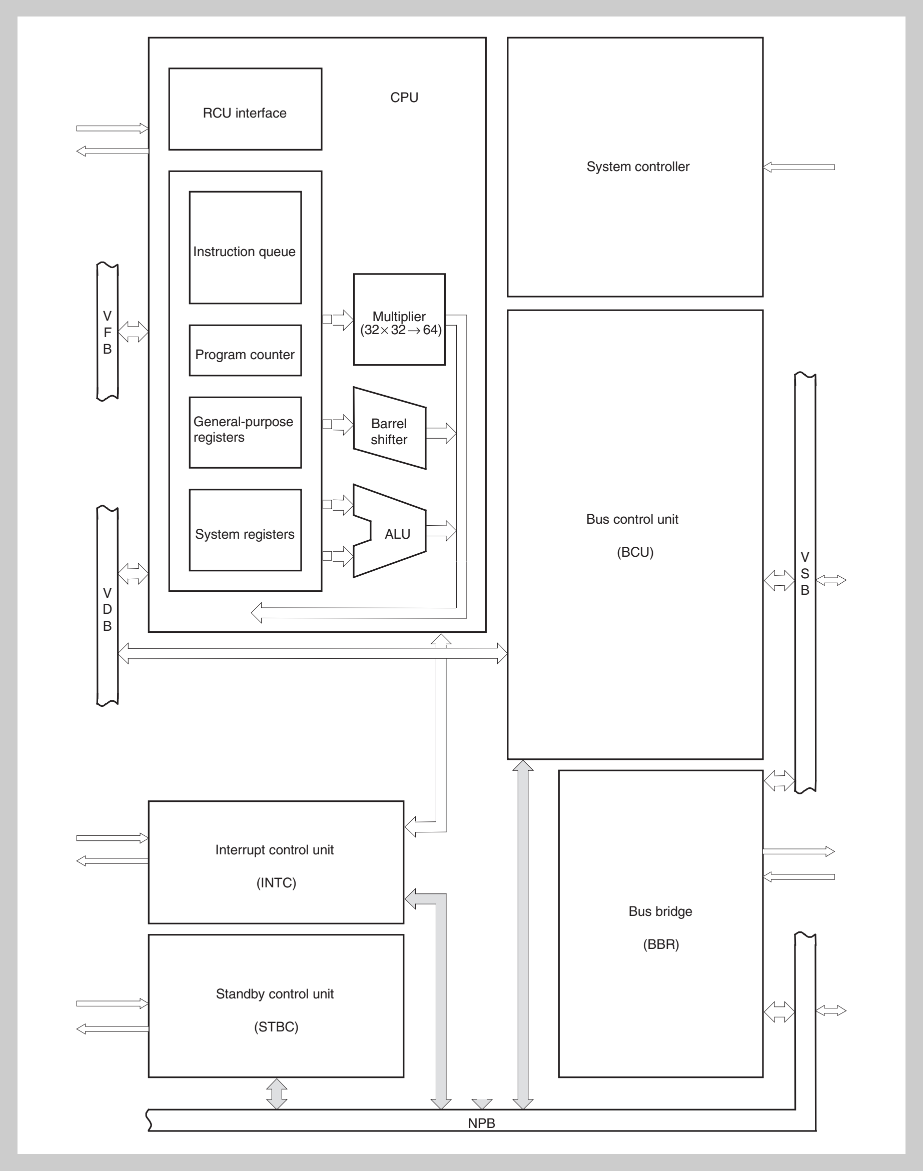


Figure 3-1 CPU system

The shaded busses are used for accessing the configuration registers of the concerned modules.

Table 3-1 Bus types

Bus type	Function
NPB – Peripheral Bus	Bus interface to the peripherals (internal bus).
VSB – V850 System Bus	Bus interface to the Memory Controller for access to external memory, additional internal memory and to the NPB bus bridge BBR.
VFB – V850 Fetch Bus	Interface to the internal flash or ROM.
VDB – V850 Data Bus	Interface to the internal RAM.

3.2 CPU Register Set

There are two categories of registers:

- General purpose registers
- System registers

All registers are 32-bit registers. An overview is given in the figure below. For details, refer to V850E1 User's Manual Architecture.

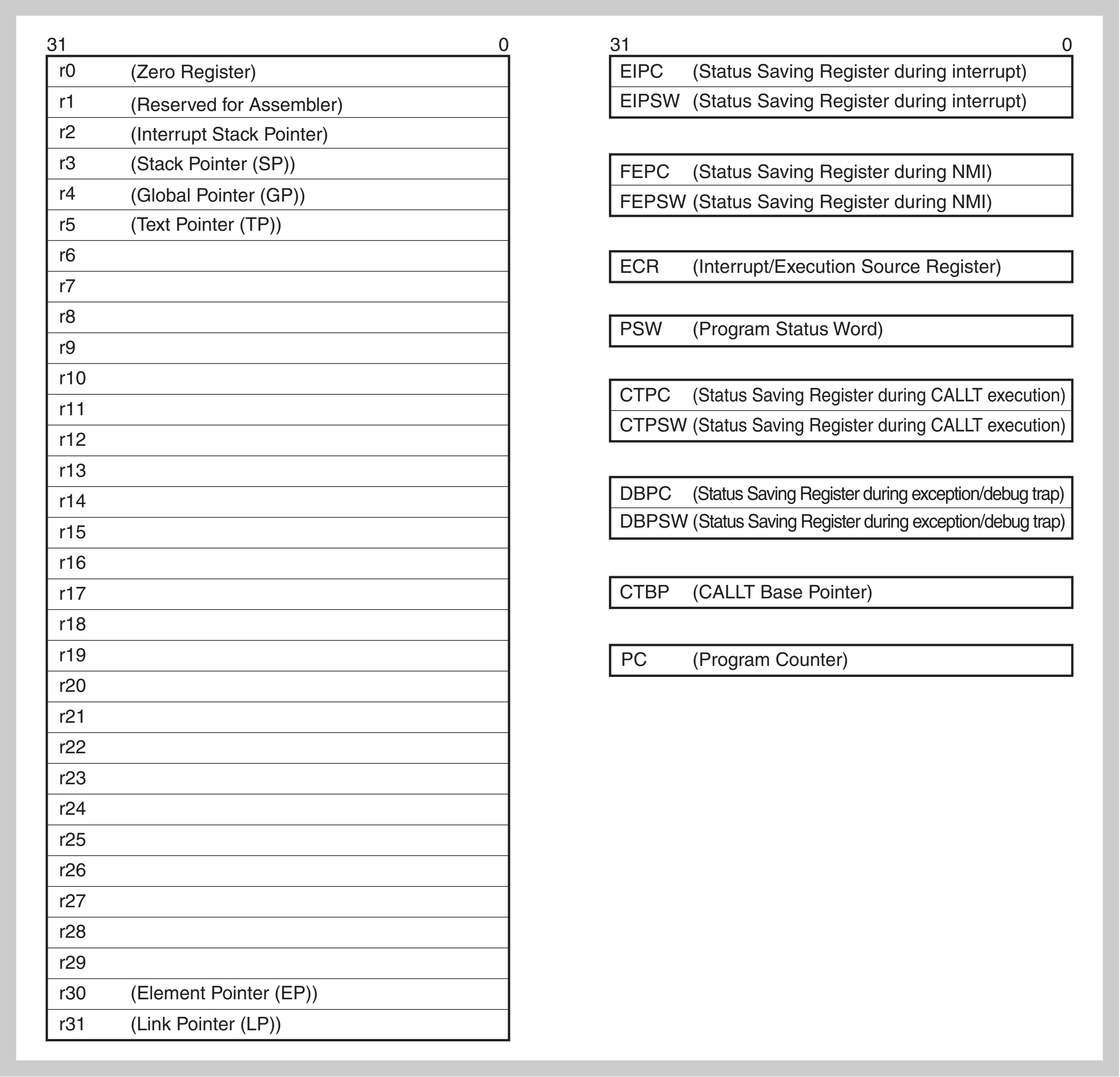

31
0
r0 (Zero Register)
r1 (Reserved for Assembler)
r2 (Interrupt Stack Pointer)
r3 (Stack Pointer (SP))
r4 (Global Pointer (GP))
r5 (Text Pointer (TP))
r6
r7
r8
r9
r10
r11
r12
r13
r14
r15
r16
r17
r18
r19
r20
r21
r22
r23
r24
r25
r26
r27
r28
r29
r30 (Element Pointer (EP))
r31 (Link Pointer (LP))
31
0
EIPC (Status Saving Register during interrupt)
EIPSW (Status Saving Register during interrupt)
FEPC (Status Saving Register during NMI)
FEPSW (Status Saving Register during NMI)
ECR (Interrupt/Execution Source Register)
PSW (Program Status Word)
CTPC (Status Saving Register during CALLT execution)
CTPSW (Status Saving Register during CALLT execution)
DBPC (Status Saving Register during exception/debug trap)
DBPSW (Status Saving Register during exception/debug trap)
CTBP (CALLT Base Pointer)
PC (Program Counter)

Figure 3-2 CPU register set

Some registers are write protected. That means, writing to those registers is protected by a special sequence of instructions. Refer to *“Write Protected Registers“ on page 96* for more details.

3.2.1 General purpose registers (r0 to r31)

Each of the 32 general purpose registers can be used as a data variable or address variable.

However, the registers r0, r1, r3 to r5, r30, and r31 may implicitly be used by the assembler/compiler (see table *Table 3-2*). For details refer to the documentation of your assembler/compiler.

Table 3-2 General purpose registers

Register name	Usage	Operation
r0	Zero register	Always holds 0. It is used for operations using 0 and offset 0 addressing.[a]
r1	Assembler-reserved register	Used for 32-bit direct addressing.[b]
r2	User address/data variable register	
r3	Stack pointer (SP)	Used to generate stack frame when function is called.[b]
r4	Global pointer (GP)	Used to access global variable in data area.[b]
r5	Text pointer (TP)	Used to indicate the start of the text area (where program code is located).[b]
r6 to r29	User address/data variable registers	
r30	Element pointer (EP)	Base pointer when memory is accessed by means of instructions SLD (short format load) and SST (short format store).[a]
r31	Link pointer (LP)	Used when calling a function.[b]

a) Registers r0 and r30 are used by dedicated instructions.

b) Registers r1, r3, r4, r5, and r31 may be used by the assembler/compiler.

Caution Before using registers r1, r3 to r5, r30, and r31, their contents must be saved so that they are not lost. The contents must be restored to the registers after the registers have been used.

3.2.2 System register set

System registers control the status of the CPU and hold interrupt information. Additionally, the program counter holds the instruction address during program execution.

To read/write the system registers, use instructions LDSR (load to system register) or STSR (store contents of system register), respectively, with a specific system register number (regID) indicated below.
The program counter states an exception. It cannot be accessed via LDSR or STSR instructions. No regID is allocated to the program counter.

Example STSR 0, r2

Stores the contents of system register 0 (EIPC) in general purpose register r2.

System register numbers The table below gives an overview of all system registers and their system register number (regID). It shows whether a load/store instruction is allowed (×) for the register or not (–).

Table 3-3 System register numbers

regID	System register name	Shortcut	Operand specification	
			LDSR	STSR
0	Status saving register during interrupt (stores contents of PC)	EIPC	×	×
1	Status saving register during interrupt (stores contents of PSW)	EIPSW	×	×
2	Status saving register during non-maskable interrupts (stores contents of PC)	FEPC	×	×
3	Status saving register during non-maskable interrupts (stores contents of PSW)	FEPSW	×	×
4	Interrupt source register	ECR	–	×
5	Program status word	PSW	×	×
6 to 15	Reserved (operations that access these register numbers cannot be guaranteed).		–	–
16	Status saving register during CALLT execution (stores contents of PC)	CTPC	×	×
17	Status saving register during CALLT execution (stores contents of PSW)	CTPSW	×	×
18	Status saving register during exception/debug trap (stores contents of PC)	DBPC	×[a]	×
19	Status saving register during exception/debug trap (stores contents of PSW)	DBPSW	×[a]	×
20	CALLT base pointer	CTBP	×	×
21 to 31	Reserved (operations that access these register numbers cannot be guaranteed).		–	–

[a] Reading from this register is only enabled between a DBTRAP exception (exception handler address 0000 0060_H) and the exception handler terminating DBRET instruction. DBTRAP exceptions are generated upon ILGOP and ROM Correction detections (refer to *“Interrupt Controller (INTC)“ on page 180* and *“ROM Correction Function (ROMC)“ on page 226*).

(1) PC - Program counter

The program counter holds the instruction address during program execution. The lower 26 bits are valid, and bits 31 to 26 are fixed to 0. If a carry occurs from bit 25 to 26, it is ignored. Branching to an odd address cannot be performed. Bit 0 is fixed to 0.

Access This register can not be accessed by any instruction.

Initial Value 0000 0000_H. The program counter is cleared by any reset.

31 26	25 1	0
fixed to 0	instruction address during execution	0

(2) EIPC, FEPC, DBPC, CTPC - PC saving registers

The PC saving registers save the contents of the program counter for different occasions, see *Table 3-4*.

When one of the occasions listed in *Table 3-4* occurs, except for some instructions, the address of the instruction following the one being executed is saved to the saving registers.
For more details refer to *Table 3-9 on page 81* and to the *“Interrupt Controller (INTC)“ on page 180*.

All PC saving registers are built up as the PC, with the initial value 0xxx xxxx_H (x = undefined).

Table 3-4 PC saving registers

Register	Shortcut	Saves contents of PC in case of
Status saving register during interrupt	EIPC	• software exception • maskable interrupt
Status saving register during non-maskable interrupts	FEPC	• non-maskable interrupt
Status saving register during exception/debug trap	DBPC[a]	• exception trap • debug trap • debug break • during a single-step operation
Status saving register during CALLT execution	CTPC	• execution of CALLT instruction

[a] Reading from this register is only enabled between a DBTRAP exception (exception handler address 0000 0060_H) and the exception handler terminating DBRET instruction. DBTRAP exceptions are generated upon ILGOP and ROM Correction detections (refer to *“Interrupt Controller (INTC)“ on page 180* and *“ROM Correction Function (ROMC)“ on page 226*).

Note When multiple interrupt servicing is enabled, the contents of EIPC or FEPC must be saved by program—because only one PC saving register for maskable interrupts and non-maskable interrupts is provided, respectively.

Caution When setting the value of any of the PC saving registers, use even values (bit 0 = 0). If bit 0 is set to 1, the setting of this bit is ignored.
This is because bit 0 of the program counter is fixed to 0.

(3) PSW - Program status word

The 32-bit program status word is a collection of flags that indicates the status of the program (result of instruction execution) and the status of the CPU.

If the bits in the register are modified by the LDSR instruction, the PSW will take on the new value immediately after the LDSR instruction has been executed.

Initial Value 0000 0020$_H$. The program status is initialized by any reset.

31	8	7	6	5	4	3	2	1	0
fixed to 0		NP	EP	ID	SAT	CY	OV	S	Z
R	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 3-5 PSW register contents

Bit position	Flag	Function
7	NP	Indicates that non-maskable interrupt (NMI) servicing is in progress. This flag is set when NMI request is acknowledged, and multiple interrupt servicing is disabled. 0: NMI servicing is not in progress. 1: NMI servicing is in progress.
6	EP	Indicates that exception processing is in progress. This flag is set when an exception occurs. Even when this bit is set, interrupt requests can be acknowledged. 0: Exception processing is not in progress. 1: Exception processing is in progress.
5	ID	Indicates whether a maskable interrupt request can be acknowledged. 0: Interrupts enabled. 1: Interrupts disabled. **Note:** Setting this flag will disable interrupt requests even while the LDSR instruction is being executed.
4	SAT[a]	For saturated operation processing instructions only: Indicates that the operation result is saturated due to overflow. 0: Not saturated. 1: Saturated. **Note:** **1.** This is a cumulative flag: The bit is not automatically cleared if subsequent instructions lead to not saturated results. To clear this bit, use the LDSR instruction to set PSW.SAT = 0. **2.** In a general arithmetic operation this bit is neglected. It is neither set nor cleared.
3	CY	Carry/borrow flag. Indicates whether a carry or borrow occurred as a result of the operation. 0: Carry or borrow did not occur 1: Carry or borrow occurred.
2	OV[a]	Overflow flag. Indicates whether an overflow occurred as a result of the operation. 0: Overflow did not occur. 1: Overflow occurred.
1	S[a]	Sign flag. Indicates whether the result of the operation is negative. 0: Result is positive or zero. 1: Result is negative.
0	Z	Zero flag. Indicates whether the result of the operation is zero. 0: Result is not zero. 1: Result is zero.

[a] In the case of saturate instructions, the SAT, S, and OV flags will be set according to the result of the operation as shown in the table below. Note that the SAT flag is set only when the OV flag has been set during a saturated operation.

Saturated operation instructions The following table shows the setting of flags PWS.SAT, PWS.OV, and PWS.S, depending on the status of the operation result.

Table 3-6 Saturation-processed operation result

Status of operation result	Flag status			Saturation-processed operation result
	SAT	OV	S	
Maximum positive value exceeded	1	1	0	7FFF $FFFF_H$
Maximum negative value exceeded	1	1	1	8000 0000_H
Positive (maximum not exceeded)	x[a]	0	0	Operation result itself
Negative (maximum not exceeded)			1	

[a] Retains the value before operation.

(4) EIPSW, FEPSW, DBPSW, CTPSWPSW saving registers

The PSW saving registers save the contents of the program status word for different occasions, see *Table 3-4*.

When one of the occasions listed in *Table 3-4* occurs, the current value of the PSW is saved to the saving registers.

All PSW saving registers are built up as the PSW, with the initial value 0000 $0xxx_H$ (x = undefined).

Table 3-7 PSW saving registers

Register	Shortcut	Saves contents of PSW in case of
Status saving register during interrupt	EIPSW	• software exception • maskable interrupt
Status saving register during non-maskable interrupts	FEPSW	• non-maskable interrupt
Status saving register during exception/debug trap	DBPSW[a]	• exception trap • debug trap • debug break • during a single-step operation
Status saving register during CALLT execution	CTPSW	• execution of CALLT instruction

[a] Reading from this register is only enabled between a DBTRAP exception (exception handler address 0000 0060_H) and the exception handler terminating DBRET instruction. DBTRAP exceptions are generated upon ILGOP and ROM Correction detections (refer to *“Interrupt Controller (INTC)“ on page 180* and *“ROM Correction Function (ROMC)“ on page 226*).

Note When multiple interrupt servicing is enabled, the contents of EIPSW or FEPSW must be saved by program—because only one PSW saving register for maskable interrupts and non-maskable interrupts is provided, respectively.

Caution Bits 31 to 26 of EIPC and bits 31 to 12 and 10 to 8 of EIPSW are reserved for future function expansion (fixed to 0).When setting the value of EIPC, FEPC, or CTPC, use even values (bit 0 = 0).
If bit 0 is set to 1, the setting of this bit is ignored. This is because bit 0 of the program counter is fixed to 0.

(5) ECR - Interrupt/exception source register

The 32-bit ECR register displays the exception codes if an exception or an interrupt has occurred. With the exception code, the interrupt/exception source can be identified.

For a list of interrupts/exceptions and corresponding exception codes, see *Table 3-9 on page 81*.

Initial Value $0000\ 0000_H$. This register is cleared by any reset.

31 to 26	25 to 0
FECC	EICC

Table 3-8 ECR register contents

Bit position	Bit name	Function
31 to 16	FECC	Exception code of non-maskable interrupt (NMI)
15 to 0	EICC	Exception code of exception or maskable interrupts

The following table lists the exception codes.

Table 3-9 Interrupt/execution codes (1/2)

Interrupt/Exception Source			Classification	Exception Code	Handler Address	Value restored to EIPC/FEPC
Name		Trigger				
Non-maskable interrupts (NMI)		NMI0 input	Interrupt	0010_H	$0000\ 0010_H$	next PC (see Note)
		NMI1 input	Interrupt	0020_H	$0000\ 0020_H$	next PC (see Note)
		NMI2 input	Interrupt	0030_H	$0000\ 0030_H$	next PC (see Note)
Maskable interrupt		refer to *“Interrupt Controller (INTC)“on page 180*	Interrupt	refer to *“Interrupt Controller (INTC)“ on page 180*	• higher 16 bits: 0000_H • lower 16 bits: exception code	next PC (see Note)
Software exception	TRAP0n (n = 0 to F_H)	TRAP instruction	Exception	$004n_H$	$0000\ 0040_H$	next PC
	TRAP1n (n = 0 to F_H)	TRAP instruction	Exception	$005n_H$	$0000\ 0050_H$	next PC

Table 3-9 Interrupt/execution codes (2/2)

Interrupt/Exception Source		Classification	Exception Code	Handler Address	Value restored to EIPC/FEPC
Name	Trigger				
Exception trap (ILGOP)	Illegal instruction code	Exception	0060_H	$0000\ 0060_H$	next PC
Debug trap	DBTRAP instruction	Exception	0060_H	$0000\ 0060_H$	next PC

If an interrupt (maskable or non-maskable) is acknowledged during instruction execution, generally, the address of the instruction *following* the one being executed is saved to the saving registers, except when an interrupt is acknowledged during execution of one of the following instructions:

- load instructions (SLD.B, SLD.BU, SLD.H, SLD.HU, SLD.W)
- divide instructions (DIV, DIVH, DIVU, DIVHU)
- PREPARE, DISPOSE instruction (only if an interrupt is generated before the stack pointer is updated)

In this case, the address of the *interrupted* instruction is restored to the EIPC or FEPC, respectively. Execution is stopped, and after the completion of interrupt servicing the execution is resumed.

(6) CTBP - CALLT base pointer

The 32-bit CALLT base pointer is used with the CALLT instruction. The register content is used as a base address to generate both a 32-bit table entry address and a 32-bit target address.

Initial Value Undefined

31	30	29	28	27	26	25 ... 1	0
0	0	0	0	0	0	base address	0
R[a]	R[a]	R[a]	R[a]	R[a]	R[a]	R/W	R

[a] These bits may be written, but write is ignored.

3.3 Operation Modes

This section describes the operation modes of the CPU and how the modes are specified.

The following operation modes are available for the flash memory devices:

- Normal operation mode
- Flash programming mode

After reset release, the microcontroller starts to fetch instructions from an internal boot ROM which contains the internal firmware. The firmware checks the FLMD0 pin, and optionally also the FLMD1 pin, to set the operation mode after reset release according to *Table 3-10*.

Table 3-10 Selection of operation modes for flash memory devices

FLMD0	FLMD1 (P50)[a]	Operation Mode
0	X	Normal operation mode (fetch from flash)
1	0	Flash programming mode
	1	Setting prohibited

[a] The FLMD1 pin function is shared with P50.

3.3.1 Normal operation mode

In normal operation mode, the internal flash memory is not re-programmed.

After reset release, the firmware acquires the user's reset vector from the extra area of the flash memory. The reset vector contains the start address of the user’s program code. The firmware branches to that address. Program execution is started.

3.3.2 Flash programming mode (flash memory devices only)

In flash programming mode, the internal flash memory is erased and re-programmed.

After reset release, the firmware initiates loading of the user's program code from the external flash programmer and programs the flash memory.

After detaching the external flash programmer, the microcontroller can be started up with the new user's program in normal operation mode.

For more information see section *“Flash Memory“ on page 164*.

3.4 Address Space

In the following sections, the address space of the CPU is explained. Size and addresses of CPU address space and physical address space are explained. The address range of data space and program space together with their wrap-around properties are presented.

3.4.1 CPU address space and physical address space

The CPU supports the following address space:

- 4 GB CPU address space
  With the 32-bit general purpose registers, addresses for a 4 GB memory can be generated. This is the maximum address space supported by the CPU.
- 64 MB physical address space
  The CPU provides 64 MB physical address space. That means that a maximum of 64 MB internal or external memory can be accessed.

Any 32-bit address is translated to its corresponding physical address by ignoring bits 31 to 26 of the address. Thus, 64 addresses point to the same physical memory address. In other words, data at the physical address 0000 0000_H can additionally be accessed by addresses 0400 0000_H, 0800 0000_H, …, F800 0000_H, or FC00 0000_H.

The 64 MB physical address space is seen as 64 images in the 4 GB CPU address space:

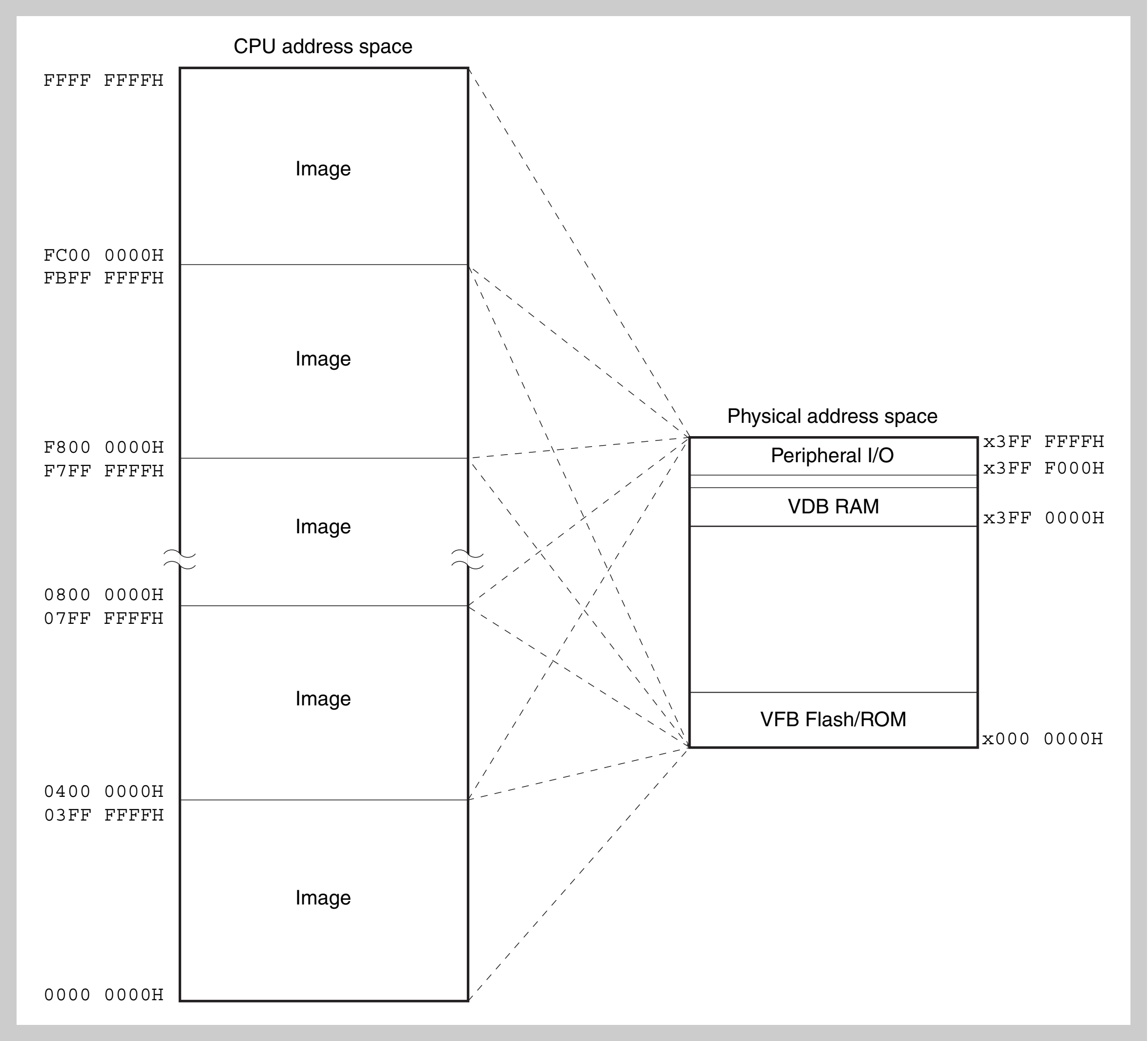


Figure 3-3 Images in the CPU address space

3.4.2 Program and data space

The CPU allows the following assignment of data and instructions to the CPU address space:

- 4 GB as data space
  The entire CPU address space can be used for operand addresses.
- 64 MB as program space
  Only the lower 64 MB of the CPU address space can be used for instruction addresses. When an instruction address for a branch instruction is calculated and moved to the program counter (PC), then bits 31 to 26 are set to zero.

Figure 3-4 shows the assignment of the CPU address space to data and program space.

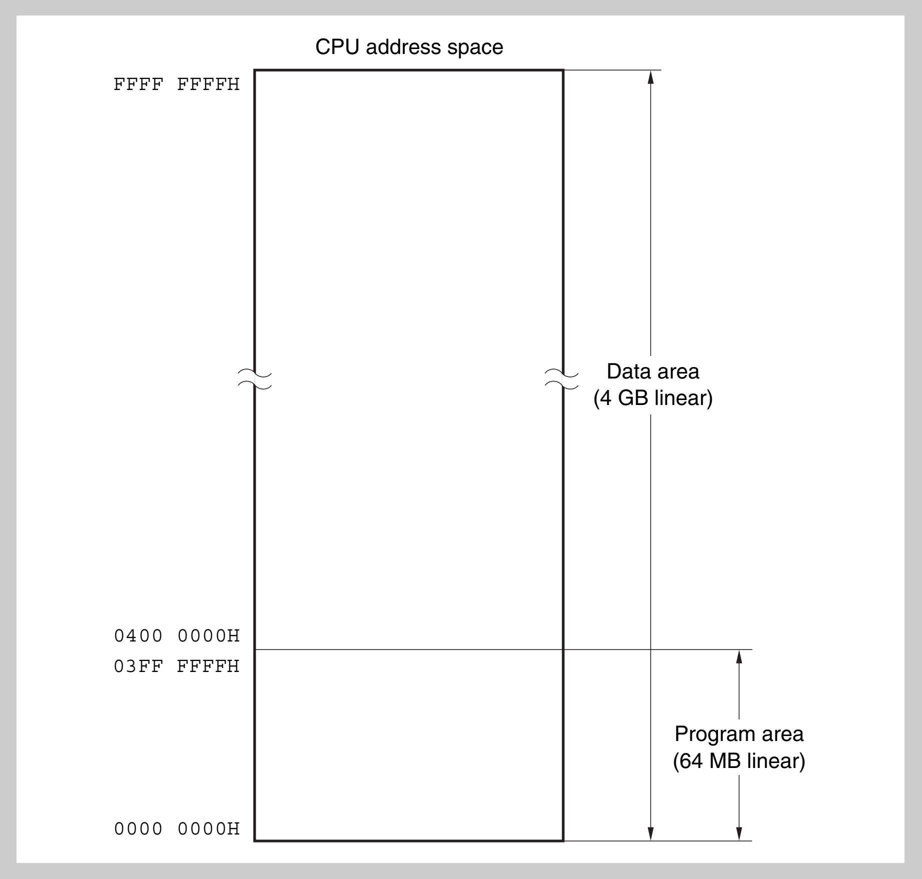


Figure 3-4 CPU address space

(1) Wrap-around of data space

If an operand address calculation exceeds 32 bits, only the lower 32 bits of the result are considered. Therefore, the addresses 0000 0000_H and FFFF $FFFF_H$ are contiguous addresses. This results in a wrap-around of the data space:

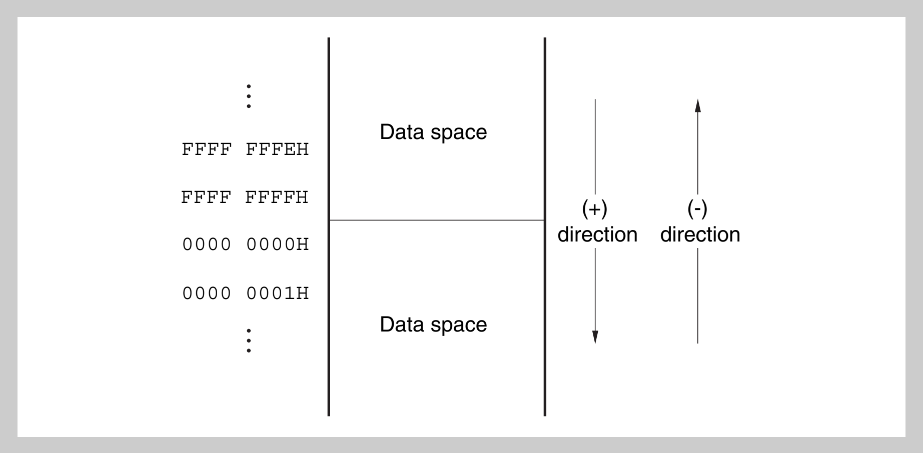


Figure 3-5 Wrap-around of data space

(2) Wrap-around of program space

If an instruction address calculation exceeds 26 bits, only the lower 26 bits of the result are considered. Therefore, the addresses 0000 0000_H and 03FF $FFFF_H$ are contiguous addresses. This results in a wrap-around of the program space:

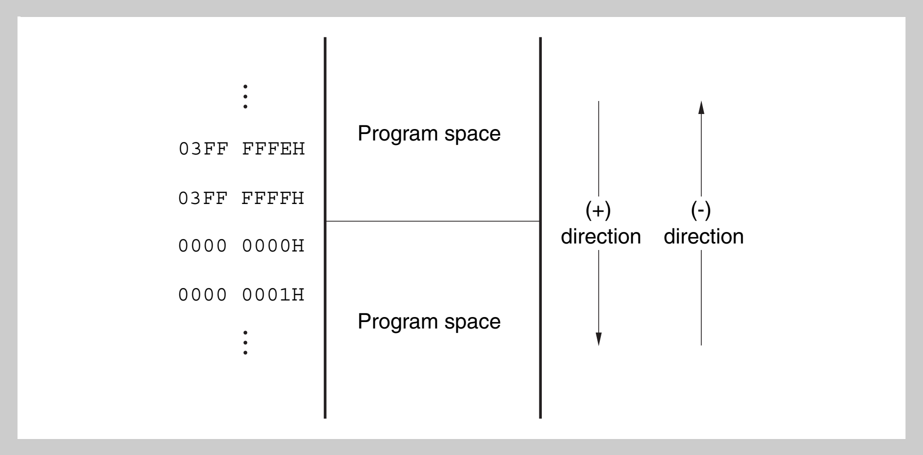


Figure 3-6 Wrap-around of program space

Caution No instruction can be fetched from the 4 KB area of 03FF $F000_H$ to 03FF $FFFF_H$ because this area is defined as peripheral I/O area. Therefore, do not execute any branch to this area.

3.5 Memory

In the following sections, the memory of the CPU is introduced. Specific memory areas are described and a recommendation for the usage of the address space is given.

3.5.1 Memory areas

The internal memory of the CPU provides several areas:

- Internal VFB flash area
- Internal VDB RAM area
- External memory area
- Internal fixed peripheral I/O area
- Programmable peripheral I/O area

The areas are briefly described below.

(1) Internal VFB flash areas

The 1 MB area between addresses 0000 0000$_{H}$ and 001F FFFF$_{H}$ is provided as the internal flash area and is accessible via the VFB (V850 Fetch Bus). It cannot be used for access to external memory.

(2) Internal VDB RAM area

After reset

The internal VDB RAM consists of several separated RAM blocks. If a reset occurs while writing to one RAM block, only the contents of that RAM block may be corrupted. The contents of the other RAM blocks remain unaffected.

Table 3-11 summarizes the VDB (V850 Data Bus) RAM blocks compilation and their address assignment.

Table 3-11 Internal VDB RAM areas

Block		
Number	**Size**	**Address**
0	8 KB	03FF 0000$_{H}$ – 03FF 1FFF$_{H}$
1	8 KB	03FF 2000$_{H}$ – 03FF 3FFF$_{H}$
2	8 KB	03FF 4000$_{H}$ – 03FF 5FFF$_{H}$
3	8 KB	03FF 6000$_{H}$ – 03FF 7FFF$_{H}$
4	8 KB	03FF 8000$_{H}$ – 03FF 9FFF$_{H}$
5	8 KB	03FF 4A000$_{H}$ – 03FF BFFF$_{H}$
6	8 KB	03FF C000$_{H}$ – 03FF DFFF$_{H}$
7	4 KB	03FF E000$_{H}$ – 03FF EFFF$_{H}$

Note that the internal firmware, which is processed after reset, uses some RAM (refer to *“General reset performance“ on page 758*).

(3) External memory area (μPD70F3419 only)

All address areas that do not address any internal memory or peripheral I/O registers can be used as external memory area.

Access to the external memory area uses the chip select (CS) signals assigned to each memory area.

For access to external memory, see *“Bus and Memory Control (BCU, MEMC)“ on page 255*.

The internal memory of the CPU provides several areas:

- Internal VFB flash area for flash memory devices
- Internal VFB OTF area for OTF devices
- Internal VDB RAM area
- Internal VSB flash area
- Internal VSB RAM area
- External memory area
- Internal fixed peripheral I/O area
- Programmable peripheral I/O area

The areas are briefly described below.

(4) Internal VFB flash/OTF areas

Table 3-12 summarizes the size and addresses of the flash memories, which are accessible via the VFB (V850 Fetch Bus).

Table 3-12 VFB flash and OTF memory

Device	Flash	OTF	Address range
μPD70F3420 (OTF)	–	128 KB	0000 0000_H to 0001 $FFFF_H$
μPD70F3421 (OTF)	–	256 KB	0000 0000_H to 0003 $FFFF_H$
μPD70F3421	256 KB	–	0000 0000_H to 0003 $FFFF_H$
μPD70F3422 (OTF)	–	384 KB	0000 0000_H to 0005 $FFFF_H$
μPD70F3422	384 KB	–	0000 0000_H to 0005 $FFFF_H$
μPD70F3423	512 KB	–	0000 0000_H to 0007 $FFFF_H$
μPD70F3424	512 KB	–	0000 0000_H to 0007 $FFFF_H$
μPD70F3425	1 MB	–	0000 0000_H to 000F $FFFF_H$
μPD70F3426	1 MB	–	0000 0000_H to 000F $FFFF_H$
μPD70F3427	1 MB	–	0000 0000_H to 000F $FFFF_H$

(5) Internal VDB RAM area

After reset The internal VDB RAM consists of several separated RAM blocks. If a reset occurs while writing to one RAM block, only the contents of that RAM block may be corrupted. The contents of the other RAM blocks remain unaffected.

Table 3-17 summarizes the VDB (V850 Data Bus) RAM blocks compilation and their address assignment.

Table 3-13 **Internal VDB RAM areas**

Device	RAM size	Block		
		Number	Size	Address
µPD70F3420 (OTF)	6 KB	0	4 KB	03FF 0000_H – 03FF $0FFF_H$
		1	2 KB	03FF 1000_H – 03FF $17FF_H$
µPD70F3421 µPD70F3421 (OTF)	12 KB	0	4 KB	03FF 0000_H – 03FF $0FFF_H$
		1	8 KB	03FF 1000_H – 03FF $2FFF_H$
µPD70F3422 µPD70F3422 (OTF) µPD70F3423	20 KB	0	8 KB	03FF 0000_H – 03FF $1FFF_H$
		1	8 KB	03FF 2000_H – 03FF $3FFF_H$
		2	4 KB	03FF 4000_H – 03FF $4FFF_H$
µPD70F3424	24 KB	0	8 KB	03FF 0000_H – 03FF $1FFF_H$
		1	8 KB	03FF 2000_H – 03FF $3FFF_H$
		2	8 KB	03FF 4000_H – 03FF $5FFF_H$
µPD70F3425[a]	32 KB	0	16 KB	03FF 0000_H – 03FF $3FFF_H$
		1	16 KB	03FF 4000_H – 03FF $7FFF_H$
µPD70F3426 µPD70F3427	60 KB	0	16 KB	03FF 0000_H – 03FF $3FFF_H$
		1	16 KB	03FF 4000_H – 03FF $7FFF_H$
		2	16 KB	03FF 8000_H – 03FF $BFFF_H$
		3	12 KB	03FF $C000_H$ – 03FF $EFFF_H$

[a] The µPD70F3425's 32 KB RAM area 03FF 0000_H to 03FF $7FFF_H$ is mirrored to the subsequent area 03FF 8000_H to 03FF $FFFF_H$. Since the upper 4 KB 03FF $F000_H$ to 03FF $FFFF_H$ is used to access the fixed peripheral I/O area, the RAM mirror must not be used to access the RAM.

Note that the internal firmware, which is processed after reset, uses some RAM (refer to *"General reset performance" on page 758*).

(6) Internal VSB flash area (µPD70F3426 only)

The µPD70F3426 provides additional flash memory, accessible via the VSB (V850 System Bus).

Table 3-14 **Internal VSB flash memory**

Device	Flash size	Address range
µPD70F3426	1 MB	0010 0000_H to 001F $FFFF_H$

(7) Internal VSB RAM area (µPD70F3426 only)

The µPD70F3426 provides additional RAM, accessible via the VSB (V850 System Bus).

Table 3-15 **Internal VSB RAM**

Device	RAM size	Block		
		Number	Size	Address
µPD70F3426	16 KB	0	12 KB	0060 0000_H – 0060 $2FFF_H$
		1	4 KB	0060 3000_H – 0060 $3FFF_H$

(8) External memory area (µPD70F3427 only)

All address areas that do not address any internal memory or peripheral I/O registers can be used as external memory area.

Access to the external memory area uses the chip select (CS) signals assigned to each memory area.

For access to external memory, see *"Bus and Memory Control (BCU, MEMC)" on page 258*.

The internal memory of the CPU provides several areas:

- Internal VFB flash area
- Internal VDB RAM area
- Internal VSB flash area
- Internal VSB RAM area
- External memory area
- Internal fixed peripheral I/O area
- Programmable peripheral I/O area

The areas are briefly described below.

(9) Internal VFB flash areas

Table 3-16 summarizes the size and addresses of the flash memories, which are accessible via the VFB (V850 Fetch Bus).

Table 3-16 VFB flash memory

Device	Flash	Address range
µPD70F3421	256 KB	0000 0000_H to 0003 $FFFF_H$
µPD70F3422	384 KB	0000 0000_H to 0005 $FFFF_H$
µPD70F3423	512 KB	0000 0000_H to 0007 $FFFF_H$
µPD70F3424	512 KB	0000 0000_H to 0007 $FFFF_H$
µPD70F3425	1 MB	0000 0000_H to 000F $FFFF_H$
µPD70F3426A	1 MB	0000 0000_H to 000F $FFFF_H$
µPD70F3427	1 MB	0000 0000_H to 000F $FFFF_H$

(10) Internal VDB RAM area

After reset

The internal VDB RAM consists of several separated RAM blocks. If a reset occurs while writing to one RAM block, only the contents of that RAM block may be corrupted. The contents of the other RAM blocks remain unaffected.

Table 3-17 summarizes the VDB (V850 Data Bus) RAM blocks compilation and their address assignment.

Table 3-17 **Internal VDB RAM areas**

Device	RAM size	Block		
		Number	Size	Address
µPD70F3421	12 KB	0	4 KB	03FF 0000_H – 03FF $0FFF_H$
		1	8 KB	03FF 1000_H – 03FF $2FFF_H$
µPD70F3422	20 KB	0	8 KB	03FF 0000_H – 03FF $1FFF_H$
		1	8 KB	03FF 2000_H – 03FF $3FFF_H$
		2	4 KB	03FF 4000_H – 03FF $4FFF_H$
µPD70F3423	20 KB	0	8 KB	03FF 0000_H – 03FF $1FFF_H$
		1	8 KB	03FF 2000_H – 03FF $3FFF_H$
		2	4 KB	03FF 4000_H – 03FF $4FFF_H$
µPD70F3424	24 KB	0	8 KB	03FF 0000_H – 03FF $1FFF_H$
		1	8 KB	03FF 2000_H – 03FF $3FFF_H$
		2	8 KB	03FF 4000_H – 03FF $5FFF_H$
µPD70F3425[a]	32 KB	0	16 KB	03FF 0000_H – 03FF $3FFF_H$
		1	16 KB	03FF 4000_H – 03FF $7FFF_H$
µPD70F3426A µPD70F3427	60 KB	0	16 KB	03FF 0000_H – 03FF $3FFF_H$
		1	16 KB	03FF 4000_H – 03FF $7FFF_H$
		2	16 KB	03FF 8000_H – 03FF $BFFF_H$
		3	12 KB	03FF $C000_H$ – 03FF $EFFF_H$

a) The µPD70F3425's 32 KB RAM area 03FF 0000_H to 03FF $7FFF_H$ is mirrored to the subsequent area 03FF 8000_H to 03FF $FFFF_H$. Since the upper 4 KB 03FF $F000_H$ to 03FF $FFFF_H$ is used to access the fixed peripheral I/O area, the RAM mirror must not be used to access the RAM.

Note that the internal firmware, which is processed after reset, uses some RAM (refer to *"General reset performance" on page 758*).

(11) Internal VSB flash area (µPD70F3426A only)

The µPD70F3426A provides additional flash memory, accessible via the VSB (V850 System Bus).

Table 3-18 **Internal VSB flash memory**

Device	Flash size	Address range
µPD70F3426A	1 MB	0010 0000_H to 001F $FFFF_H$

(12) Internal VSB RAM area (µPD70F3426A only)

The µPD70F3426A provides additional RAM, accessible via the VSB (V850 System Bus).

Table 3-19 **Internal VSB RAM**

Device	RAM size	Block		
		Number	Size	Address
µPD70F3426A	24 KB	0	12 KB	0060 0000_H – 0060 $2FFF_H$
		1	12 KB	0060 3000_H – 0060 $5FFF_H$

(13) External memory area (µPD70F3427 only)

All address areas that do not address any internal memory or peripheral I/O registers can be used as external memory area.

Access to the external memory area uses the chip select (CS) signals assigned to each memory area.

For access to external memory, see *“Bus and Memory Control (BCU, MEMC)“ on page 258*.

The internal memory of the CPU provides several areas:

- Internal VFB flash area
- Internal VDB RAM area
- Internal fixed peripheral I/O area
- Programmable peripheral I/O area

The areas are briefly described below.

(14) Internal VFB flash areas

The internal VFB ROM and flash areas are listed in the below table.

Table 3-20 VFB flash memory

Device	Flash	Address range
µPD70F3416	128 KB	0000 0000_H to 0001 $FFFF_H$
µPD70F3417	256 KB	0000 0000_H to 0003 $FFFF_H$

(15) Internal VDB RAM area

The internal VDB RAM areas are listed in the below table.

Table 3-21 Internal VDB RAM areas

Device	RAM size	Address
µPD70F3416	6 KB	03FF 0000_H – 03FF $17FF_H$
µPD70F3417	12 KB	03FF 0000_H – 03FF $2FFF_H$

Note that the internal firmware, which is processed after reset, uses some RAM (refer to *“General reset performance“ on page 758*).

3.5.2 Fixed peripheral I/O area

The 4 KB area between addresses 03FF F000$_H$ and 03FF FFFF$_H$ is provided as the fixed peripheral I/O area. Accesses to these addresses are passed over to the NPB bus (internal bus).

The following registers are memory-mapped to the peripheral I/O area:

- All registers of peripheral functions
- Registers of timers
- Configuration registers of interrupt and bus controllers
- Configuration registers of the clock controller

For a list of all peripheral I/O registers, see *“Special Function Registers“ on page 775*.

Note

1. Because the physical address space covers 64 MB, the address bits A[31:26] are not considered. Thus, this address space can also be addressed via the area FFFF 0000$_H$ to FFFF FFFF$_H$. This has the advantage that the area can be indirectly addressed by an offset and the zero base r0.
   Therefore, in this manual, all addresses of peripheral I/O registers in the 4 KB peripheral I/O area are given in the range FFFF F000$_H$ to FFFF FFFF$_H$ instead of 03FF F000$_H$ to 03FF FFFF$_H$.
2. The *fixed* peripheral I/O area is mirrored to the upper 4 KB of the *programmable* peripheral I/O area PPA - regardless of the base address of the PPA. If data is written to one area, it appears also in the other area.
3. Program fetches cannot be executed from any peripheral I/O area.
4. Word registers, that means 32-bit registers, are accessed in two half word accesses. The lower two address bits are ignored.
5. For registers in which byte access is possible, if half word access is executed:
   - During read operation: The higher 8 bits become undefined.
   - During write operation: The lower 8 bits of data are written to the register.

Caution Addresses that are not defined as registers are reserved for future expansion. If these addresses are accessed, the operation is undefined and not guaranteed.

(1) Programmable peripheral I/O area

A 16 KB area is provided as a programmable peripheral I/O area (PPA). The PPA can be freely located. The base address of the programmable peripheral I/O area is specified by the initialization of the peripheral area selection control register (BPC).

See *“Bus and Memory Control (BCU, MEMC)“ on page 258* for details.

3.5.3 Recommended use of data address space

When accessing operand data in the data space, one register has to be used for address generation. This register is called pointer register. With relative addressing, an instruction can access operand data at all addresses that lie in the range of ±32 KB relative to the address in the pointer register.

By this offset addressing method load/store instructions can be accommodated in a single 32-bit instruction word, resulting in faster program execution and smaller code size.

To enhance the efficiency of using the pointer in consideration of the memory map, the following is recommended:

For efficient use of the relative addressing feature, the data segments should be located in the address range FFFF $F800_H$ to 0000 0000_H and 0000 0000_H to 0000 $7FFF_H$. The peripheral I/O registers and the internal RAM is aligned to the upper bound, thus the registers and a part of the RAM can be addressed via relative addressing, with base address 0 (r0).

It is recommended to locate flash memory data segments in the area up to 0000 $7FFF_H$, so access to these constant data can utilize also relative addressing.

Use the r0 register as pointer register for operand addressing. Since the r0 register is fixed to zero by hardware, it can be used as a pointer register and, at the same time, for any other purposes, where a zero register is required. Thus, no other general purpose register has to be reserved as pointer register.

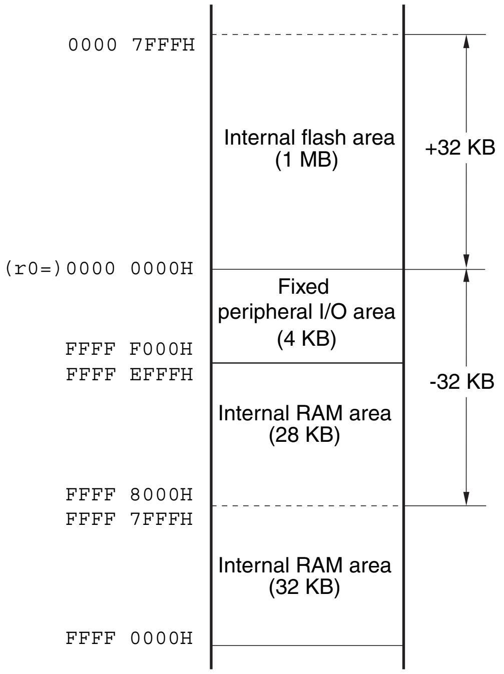


Figure 3-7 Example application of wrap-around

3.6 Write Protected Registers

Write protected registers are protected from inadvertent write access due to erroneous program execution, etc. Write access to a write protected register is only given immediately after writing to a corresponding write enable register. For a write access to the write protected registers you have to use the following instructions:

1. Store instruction (ST/SST instruction)
2. Bit operation instruction (SET1/CLR1/NOT1 instruction)

When *reading* write protected registers, no special sequence is required.

The following table gives an overview of the write protected registers and their corresponding write enable registers.

For some registers, incorrect store operations can be checked by a flag of the corresponding status register. This is also marked in the table below.

Table 3-22 Overview of write protected registers

Write protected register	Shortcut	Corresponding write enable register	Shortcut	Status register	For details see
Clock control register	CKC	Peripheral command register	PHCMD	PHS	*“Clock Generator“ on page 100*
Watchdog timer clock control register	WCC				
Processor clock control register	PCC				
Watch Timer clock control register	TCC				
SPCLK control register	SCC				
FOUTCLK control register	FCC				
I^2C clock control register	ICC				
Main oscillator clock monitor mode register	CLMM	Main oscillator clock monitor command protection register	PRCMDCMM	–	*“Clock Generator“ on page 100*
Sub oscillator clock monitor mode register	CLMS	Sub oscillator clock monitor command protection register	PRCMDCMS		
Power save control register	PSC	Command register	PRCMD	–	*“Clock Generator“ on page 100*
Self-programming enable control register	SELFEN	Sequence protect register	SELFENP	–	*“Flash Memory“ on page 164*
Watchdog timer frequency select register	WDCS	Watchdog timer security register	WCMD	WPHS	*“Watchdog Timer (WDT)“ on page 395*
Watchdog timer mode register	WDTM				
N-Wire security disable control register	RSUDISC	RSUDISC write protection register	RSUDISCP	–	*“On-Chip Debug Unit“ on page 930*

Example Start the Watchdog Timer

The following example shows how to write to the write protected register WDTM. The example starts the Watchdog Timer.

```
do {
     _WPRERR = 0;

     DI();
     WCMD = 0x5A;
     WDTM = 0x80;
     EI();

} while (_WPRERR != 0)
```

Note

1. Make sure that the compiler generates two consecutive assembler “store” instructions to WCMD and WDTM from the associated C statements.
2. Special care must be taken when writing to registers PCS and PRCMD. Please refer to *“Clock Generator“ on page 100* for details.

Since any action between writing to a write enable register and writing to a protected register destroys this sequence, the effects of interrupts transfers have to be considered:

Interrupts In order to prevent any maskable interrupt to be acknowledged between the two write instructions in question, shield this sequence by DI - EI (disable interrupt - enable interrupt).
However, any non-maskable interrupt can still be acknowledged.

The above examples checks WPHS.WPRERR for that purposes and repeats the sequence until the write to WDTM was successful.

3.7 Instructions and Data Access Times

The below *Table 3-23* and *Table 3-24* list the instruction execution and data access cycles, required when accessing instructions or data in VFB flash and VDB RAM.

The access time depends on the

- memory type (flash, RAM) and access bus (VFB, VDB)
- number of latency cycles for the memory type
- type of data (instructions/data)
- type of access (consecutive/random addresses)
- device, i.e. maximum clock frequency

In general the CPU is able to execute most instructions in one clock cycle (single-cycle instructions), provided no additional clock cycles are required to access the memory.
Note that for some instructions the CPU requires more clock cycles to execute anyway (multi-cycle instructions), regardless of the memory access time.

The memory access time in a real application is deterministic, but can hardly be predicted, as this heavily depends on the status of the microcontroller and its components, the program flow and concurrent processes, like interrupts, accesses to peripheral registers via the NPB, etc.
Thus the figures in the below tables assume

- all busses (VFB, VDB, NPB) are not occupied, i.e. collision with other bus traffic is excluded
- 32-bit instruction/data accesses to word-aligned - that means 32-bit aligned - addresses
- data is not accessed via the same bus as the instruction is fetched from

Consequently “1 clock cycle” means: the instruction/data access takes one CPU clock cycle and the CPU is supplied with an instruction/data in each clock: the memory access time is invisible and has no effect.

Instruction execution

The given numbers of cycles in *Table 3-23* describe the time required to execute a single-cycle instruction, fetched from the respective memory:

- Consecutive access
  describes the number of cycles required to fetch instructions from the memory on consecutive addresses.
- Random access
  describes the number of cycles required to access the memory in case instructions are accessed on random, i.e. non-consecutive, addresses. In case of instruction flow branches a CPU’s pipeline break occurs and an additional cycle is required to refill the pipeline. The table figures include this cycle.

In case instructions and data are accessed via the same bus, all accesses - instruction fetch and data access - are regarded as random accesses.

Table 3-23 Single-cycle instructions execution times in CPU clock cycles

Memory	Access type	μPD70F3416 μPD70F3417
VFB flash	Consecutive	1
	Random	1[a]
VDB RAM	Consecutive	1
	Random	1[a]

[a] These values include the additional clock cycle, caused by the CPU's pipeline break

Data access The given numbers of cycles in *Table 3-24* describe the time additionally required when an instruction accesses data in the respective memory.

Note that data accesses are always random accesses.

Table 3-24 Additional time for data accesses in CPU clock cycles

Data access memory	Instruction code fetch bus	μPD70F3416 μPD70F3417
VFB flash	VFB	1
VDB RAM	VFB	0

Chapter 4 Clock Generator

The clock generator provides the clock signals needed by the CPU and the on-chip peripherals.

4.1 Overview

The clock generator can generate the required clock signals from the following sources:

- Main oscillator - a built-in oscillator with external crystal and a nominal frequency of 4 MHz
- Sub oscillator - a built-in oscillator with external crystal and a nominal frequency of 32 kHz
- Internal oscillator - an internal oscillator without external components and a nominal frequency of 240 kHz

Features summary

Special features of the clock generator are:

- Choice of oscillators to reduce power consumption in stand-by mode
- Frequency multiplication by two PLL synthesizers:
  - Fixed frequency PLL for accurate timings
  - Spread spectrum PLL (SSCG) for reduced electromagnetic interference
- Individual clock source selection for CPU and groups of peripherals
- Five specific power save modes:
  - HALT mode
  - IDLE mode
  - WATCH mode
  - Sub-WATCH mode
  - STOP mode
- Vital system registers are write-protected by a special write sequence
- Direct main oscillator clock feed-through for watch clock correction support
- Separate clock monitors for main and sub oscillator to detect oscillator malfunction

4.1.1 Description

The clock generator is built up as illustrated in the following figure.

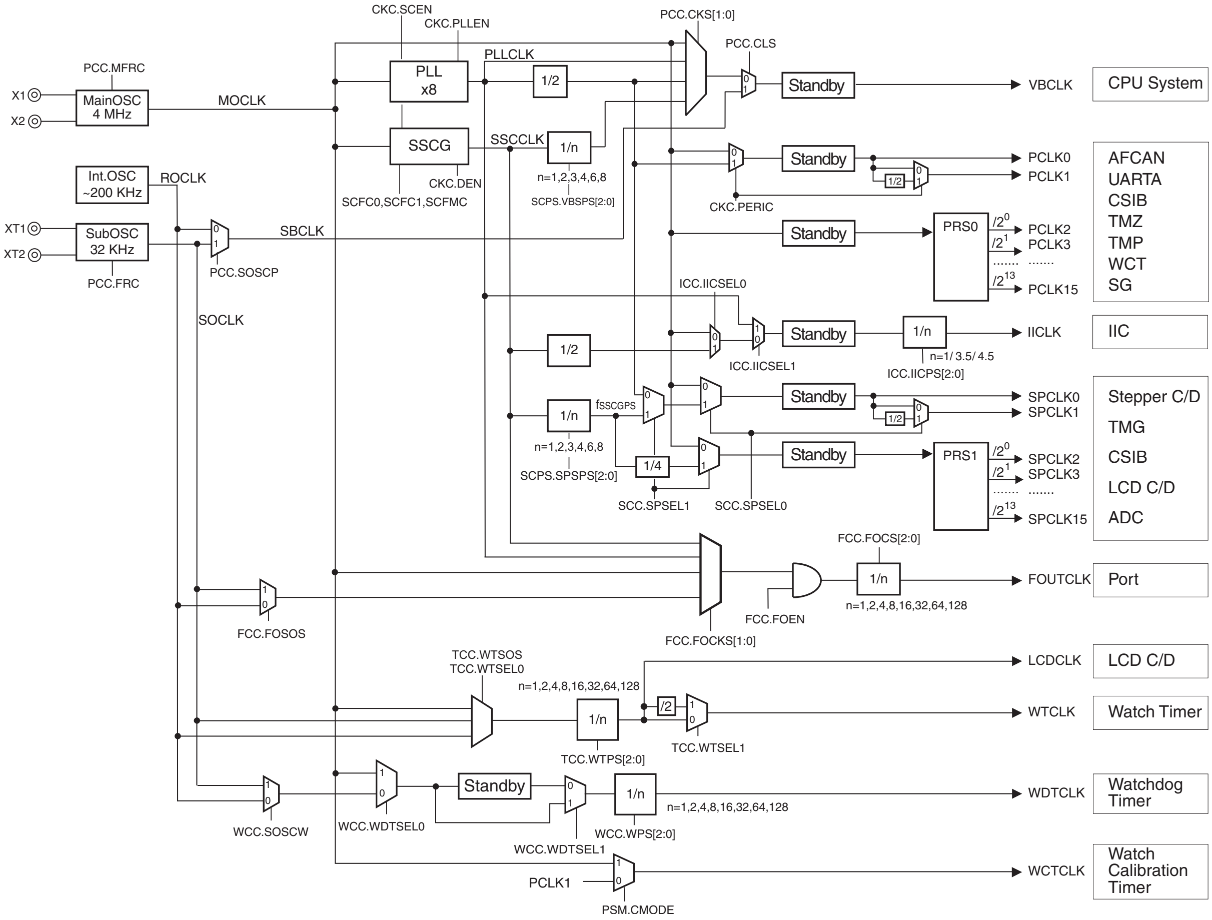


Figure 4-1 Block diagram of the Clock Generator

The left-hand side of the figure shows how the three oscillators can be connected to the CPU, the two PLLs, and to certain peripheral modules. Software-controlled selectors allow you to specify the signal paths.

PLL The integrated PLL synthesizer multiplies the frequency of the main oscillator by eight. This yields a frequency of 32 MHz. The CPU can use the PLL output directly. The output frequency of the PLL divided by two can supply the peripherals of the microcontroller and also the CPU.

SSCG The spread spectrum clock generator (SSCG) can generate a frequency-modulated clock (modulation frequency and width can be chosen) that helps to eliminate electromagnetic interference (EMI). The SSCG includes a programmable frequency multiplier/divider that can multiply the frequency of the main oscillator by up to 16.

The SSCG can supply the CPU system and some of the peripherals.

(1) CPU clocks

The CPU can be clocked directly by any of the oscillators, or by the output of one of the PLLs.

The following table gives an overview of the available CPU clocks.

Table 4-1 Clock sources and frequencies for the CPU

Clock source	Frequency	Description
Ring osc	~240 KHz	Default clock source after reset release. Selectable as clock source for Sub-WATCH mode release.
Sub osc	32 KHz	Selectable as clock source for Sub-WATCH mode release.
Main osc	4 MHz	Always selected after power save mode release except on Sub-WATCH mode release or default clock setting.[a] On Sub-WATCH mode release or default clock setting, main or sub oscillator can be selected.
PLL	16 MHz	$f_{main} \times 8/2$[b] can be selected for CPU clock supply.
SSCG	8 MHz[c]	$f_{main} \times 12/6$[d] can be selected for CPU clock supply.
	12 MHz[c]	$f_{main} \times 12/4$[d] can be selected for CPU clock supply.
	16 MHz[c]	$f_{main} \times 12/3$[d] can be selected for CPU clock supply.
	24 MHz[c]	$f_{main} \times 12/2$[d] can be selected for CPU clock supply.

[a] See also *“CPU operation after power save mode release“ on page 157*
[b] Multiplication is performed by the PLL, the division by the PLL post scaler.
[c] Center output frequency of the SSCG, can be modulated up to +/- 5%.
[d] Multiplication is performed by the SSCG, the division by the SSCG post scaler.

(2) Peripheral clocks

The right-hand side of *Figure 4-1 on page 101* shows how the clocks for the peripheral modules are generated and distributed.

PCLK clocks Peripherals that require precise timings are connected to *PCLKn* signals.

Such peripherals are the CAN controllers, the UARTs, the Timers Z and P, and the clocked serial interfaces CSIB. The Watch Calibration Timer WCT can be connected to PCLK1 or directly to the main oscillator.

The clocks PCLK0…1 can be derived from the main oscillator or the PLL output. The PCLK2…15 clocks are always derived from the main oscillator.

SPCLK clocks Peripherals that tolerate or demand a spread spectrum clock (like PWM output timers) are connected to *SPCLKn* signals.

Such peripherals are the Stepper Motor Controller/Driver, the Timers G and Y, the sound generator, the clocked serial interfaces CSIB (CSIB can also be connected to a PCLK), the LCD Bus I/F and Controller/Driver, and the A/D converter.

The clocks SPCLK0…1 can be derived from the main oscillator, the SSCG, or the PLL. The SPCLK2…15 clocks can be derived from the main oscillator or the SSCG.

IICLK clock The clock IICLK for the I^2C interface has it’s own programmable frequency divider. The clock source can be chosen from the PLL, SSCG or main oscillator.

(3) Special clocks

The figure shows also some special clock signals. These are dedicated clocks for the LCD controller/driver, Watch Timer, Watchdog Timer, and Watch Calibration Timer. These clocks are directly derived from the oscillators and bypass the PLLs.

LCDCLK The LCD Controller/Driver can be clocked by SPCLK7, SPCLK9, or LCDCLK.

WTCLK This is the clock for the Watch Timer. It forms the time base for updating the internal bookkeeping of daytime and calendar.

Note that LCDCLK and WTCLK have a common source and a fixed frequency ratio (1/1 or 1/2).

WCTCLK This is the clock for the Watch Calibration Timer WCT. The WCT is used in conjunction with the Watch Timer for calibrating the time base during power save modes by utilizing the main oscillator as the stable clock source. WCTCLK can also be derived from PCLK1.

FOUTCLK FOUTCLK is a clock signal that can be used for external devices. It is connected to the pin FOUT and can provide almost any of the internal clock frequencies (not phase-synchronized). FOUTCLK must be enabled before it can be used.

WDTCLK This is the clock for the Watchdog Timer that is used for recovering from a system deadlock. WDTCLK is available (and hence the Watchdog Timer running) as long as the chosen clock source is active. Optionally WDTCLK can be stopped during a power save mode.

(4) Stand-by control

In the block diagram, you find also boxes labelled “Standby”. These boxes symbolize the switches that are used to disable circuits when the microcontroller enters one of the various power save modes.

The following clocks are subject to automatic stand-by control:

CPU system clock, PCLK, SPCLK, IICLK, optionally WDTCLK.

The following clocks can be operating during power save modes (stand-by) as long as their clock oscillator source is available:

FOUTCLK, LCDCLK, WTCLK, WCTCLK, optionally WDTCLK.

4.1.2 Clock monitors

The microcontroller contains clock monitors to monitor the operation of the 4 MHz main oscillator and the 32 KHz sub oscillator. In case of malfunction, these monitors can generate a system reset.

The monitors require that the built-in internal oscillator is active. For details see *“Operation of the Clock Monitors“ on page 162*.

4.1.3 Power save modes overview

The microcontroller provides the following stand-by modes: HALT, IDLE, WATCH, Sub-WATCH, and STOP. Application systems which are designed in a way that they switch between these modes according to operation purposes, reduce power consumption efficiently.

The following explanations provide a general overview and refer to the default settings. Some settings can be changed, for example the activity of the watch and watchdog clocks and hence the connected timers.

For details, please refer to *“Power save modes description“ on page 141* and the register descriptions.

HALT mode In this mode, the *clock supply to the CPU* is suspended while other on-chip peripherals continue to operate. Combining this mode with the normal operating mode results in intermittent operation and reduces the overall system power consumption.

This mode is entered by executing the HALT instruction.

All other power save modes are entered by setting the registers PSM and PSC.

IDLE mode In this mode, the *clock distribution* is stopped and hence the whole system. The oscillators, Clock Generator (PLL, SSCG, frequency multipliers, dividers), Watch Timer, and Watchdog Timer remain operating.

This mode allows quick return to the normal operating mode in response to a release signal, because it is not necessary to wait for oscillators or PLLs to settle.

This mode provides low power consumption. Power is only consumed by the oscillators (main oscillator, sub oscillator), Clock Generator (PLL and SSCG), and Watch Timer / Watchdog Timer.

WATCH mode In this mode, the *Clock Generator* (PLL and SSCG) stops operation. Therefore, the entire system except Watch Timer / Watchdog Timer stops.

This mode provides low power consumption. Power is only consumed by the oscillators (main oscillator, sub oscillator), and the Watch Timer / Watchdog Timer circuits.

Sub-WATCH mode In this mode, not only the Clock Generator is stopped but also the *main oscillator*. Watch Timer / Watchdog Timer are switched to the sub or internal oscillator. Therefore, the entire system except Watch Timer / Watchdog Timer stops.

This mode provides very low power consumption. Power is only consumed by the sub oscillator and Watch Timer / Watchdog Timer circuits.

STOP mode In this mode, the entire system stops.

This mode provides ultra-low power consumption. Power is only consumed by leakage current and the sub oscillator (if a crystal is connected).

4.1.4 Start conditions

After any reset release, the internal oscillator is always selected as the clock source. The oscillation stabilization time for the internal oscillator is ensured by hardware. The CPU clock VBCLK is derived from the internal oscillator.

Several clocks are operating based on the internal oscillator clock after reset. As soon as the main oscillator, which is started by the internal firmware, is stable the source of these clocks is automatically changed to the main oscillator. Therefore depending on the firmware operation and the main oscillator stabilization time these clocks may already be operating with the main oscillator, when the user's program is started.

Internal firmware starts the main oscillator. PLL and SSCG remain stopped.

When the firmware passes control to the application software, software has to ensure that the main oscillator has stabilized and to start the PLL and SSCG.

Note Clock supply for most peripherals is not available unless the main oscillator operates.

CPU access to peripherals that have no clock supply may cause system deadlock.

Table 4-2 Clock Generator status after reset release

Item	Status	Remarks
Main oscillator	stopped	started by internal firmware
Sub oscillator	operates	
Internal oscillator	operates	
SSCG	stopped	
PLL	stopped	
VBCLK (CPU system)	operates	based on internal oscillator clock
IICLK	operates	based on internal/main oscillator clock[a]
PCLK0, PCLK1	operates	based on internal/main oscillator clock[a]
PCLK2…PCLK15	operates	based on internal/main oscillator clock[a]
SPCLK0, SPCLK1	operates	based on internal/main oscillator clock[a]
SPCLK2…SPCLK15	operates	based on internal/main oscillator clock[a]
FOUTCLK	operates	based on internal/main oscillator clock[a]
LCDCLK / WTCLK	operates[b]	based on internal/main oscillator clock[a]
WDTCLK	operates	based on internal/main oscillator clock[a]
WCTCLK	operates	based on internal/main oscillator clock[a]

a) Starts with internal oscillator, automatically changed to main oscillator, when main oscillator stable.

b) If the reset was caused by Power-On Clear (POC) or external $\overline{\text{RESET}}$, the clock source for LCDCLK and WTCLK is set to internal oscillator. If the reset was caused by a different source, the clock selection for LCDCLK / WTCLK remains unchanged.

4.1.5 Start-up guideline

After reset release, the internal firmware starts the main oscillator, but hands over control to the user's software without ensuring that the main oscillator has stabilized.

After that, the user's software will typically:

1. Ensure that the main oscillator has stabilized (check CGSTAT.OSCSTAT).
2. Switch the source of LCDCLK/WTCLK and WDTCLK to main oscillator (if desired).
3. Start the PLL (set CKC.PLLEN) and wait until the PLL has stabilized (refer to the Data Sheet).
4. If the SSCG is going to be used:
   Write SSCG registers to set up the SSCG. This is only possible when the SSCG is switched off.
   Start the SSCG (set CKC.SCEN) and wait until the SSCG has stabilized (refer to the Data Sheet).
5. Write the PCC register to specify the SSCG as the clock source for the CPU.
6. Set up the clock sources for the peripherals according to application requirements.
7. The default value of following registers must be changed after reset:
   – ADA0M1.bit7 = 1 (refer to *"ADC Registers" on page 689*)

4.2 Clock Generator Registers

The Clock Generator is controlled and operated by means of the following registers (the list is sorted according to memory allocation):

Table 4-3 Clock Generator register overview

Register name	Shortcut	Address	Write-protected by register
PSC write protection register	PRCMD	FFFF $F1FC_H$	
Power save control register	PSC	FFFF $F1FE_H$	PRCMD
Stand-by control register	STBCTL	FFFF $FCA2_H$	STBCTLP
Stand-by control protection register	STBCTLP	FFFF $FCAA_H$	
Sub oscillator clock monitor control register	CLMCS	FFFF $F71A_H$	
Command protection register	PHCMD	FFFF $F800_H$	
Peripheral status register	PHS	FFFF $F802_H$	
Power save mode register	PSM	FFFF $F820_H$	
Clock Generator control register	CKC	FFFF $F822_H$	PHCMD
Clock Generator status register	CGSTAT	FFFF $F824_H$	
Watchdog timer clock control register	WCC	FFFF $F826_H$	PHCMD
Processor clock control register	PCC	FFFF $F828_H$	PHCMD
SSCG Frequency modulation control register	SCFMC	FFFF $F82A_H$	
SSCG Frequency control 0 register	SCFC0	FFFF $F82C_H$	
SSCG Frequency control 1 register	SCFC1	FFFF $F82E_H$	
SSCG post scaler control register	SCPS	FFFF $F830_H$	
SPCLK control register	SCC	FFFF $F832_H$	PHCMD
FOUTCLK control register	FCC	FFFF $F834_H$	PHCMD
Watch Timer clock control register	TCC	FFFF $F836_H$	PHCMD
IIC clock control register	ICC	FFFF $F838_H$	PHCMD
Set default clock register	SDC	FFFF $F83C_H$	PHCMD
Main oscillator clock monitor mode register	CLMM	FFFF $F870_H$	PRCMDCMM
Sub oscillator clock monitor mode register	CLMS	FFFF $F878_H$	PRCMDCMS
CLMM write protection register	PRCMDCMM	FFFF $FCB0_H$	
CLMS write protection register	PRCMDCMS	FFFF $FCB2_H$	

Note Some registers are write-protected to avoid inadvertent changes. Data can be written to these registers only in a special sequence of instructions, so that the register contents is not easily rewritten in case of a program hang-up.

Writing to a protected register is only possible immediately after writing to the associated write protection register.

The subsequent register descriptions are grouped as follows:

- **General Clock Generator Registers:**
  - *"CKC - Clock Generator control register" on page 109*
  - *"CGSTAT - Clock Generator status register" on page 110*
  - *"PHCMD - Command protection register" on page 111*
  - *"PHS - Peripheral status register" on page 112*
  - *"PCC - Processor clock control register" on page 113*
  - *"SDC - Set default clock register" on page 115*
  -
- **SSCG Control Registers:**
  - *"SCFC0 - SSCG frequency control register 0" on page 117*
  - *"SCFC1 - SSCG frequency control register 1" on page 118*
  - *"SCFMC - SSCG frequency modulation control register" on page 119*
  - *"SCPS - SSCG post scaler control register" on page 121*
- **Control Registers for Peripheral Clocks:**
  - *"WCC - Watchdog Timer clock control register" on page 122*
  - *"TCC - Watch Timer clock control register" on page 124*
  - *"SCC - SPCLK control register" on page 126*
  - *"FCC - FOUTCLK control register" on page 127*
  - *"ICC - IIC clock control register" on page 129*
- **Control Registers for Power Save Modes:**
  - *"PSM - Power save mode register" on page 130*
  - *"PSC - Power save control register" on page 133*
  - *"PRCMD - PSC write protection register" on page 134*
  - *"STBCTL- Stand-by control register" on page 135*
  - *"STBCTLP - Stand-by control protection register" on page 135*
- **Clock Monitor Registers:**
  - *"CLMM - Main oscillator clock monitor mode register" on page 136*
  - *"PRCMDCMM - CLMM write protection register" on page 137*
  - *"CLMS - Sub oscillator clock monitor register" on page 138*
  - *"PRCMDCMS - CLMS write protection register" on page 139*
  - *"CLMCS - Sub oscillator clock monitor control register" on page 140*

4.2.1 General clock generator registers

The general Clock Generator registers control and reflect the operation of the Clock Generator.

(1) CKC - Clock Generator control register

The 8-bit CKC register controls the clock management.

Access This register can be read/written in 8-bit units.

Writing to this register is protected by a special sequence of instructions. Please refer to *“PHCMD - Command protection register“ on page 111* for details.

Address FFFF F822$_H$.

Initial Value 00$_H$. The register is initialized by any reset.

7	6	5	4	3	2	1	0
PLLEN	SCEN	DEN	0	PERIC	0	0	0
R/W	R/W	R/W	R[a]	R/W	R[a]	R[a]	R[a]

[a] These bits may be written, but write is ignored.

Table 4-4 CKC register contents

Bit position	Bit name	Function
7	PLLEN[a]	PLL enable: 0: PLL disabled. 1: PLL on. It is not possible to clear this bit by writing to the register. The bit is automatically cleared in WATCH, Sub-WATCH, or STOP mode, or if bit SDC.SDCR is set to 1.
6	SCEN[a]	SSCG enable: 0: SSCG disabled. 1: SSCG on. It is not possible to clear this bit by writing to the register. The bit is automatically cleared in WATCH, Sub-WATCH, or STOP mode, or if bit SDC.SDCR is set to 1.
5	DEN	SSCG dithering mode enable: 0: SSCG uses fixed multiplication factor determined by SCFC0, SCFC1 1: SSCG is in dithering mode. The base frequency, determined by the registers SCFC0, SCFC1, is modulated according to the setup of register SCFMC. DEN must not be toggled while SCEN is 1.
3	PERIC	Clock source selection for PCLK0/1: 0: Main oscillator is clock source for peripheral clocks PCLK0, PCLK1. 1: PLL (x4) is clock source for peripheral clocks PCLK0, PCLK1. This bit is automatically cleared in WATCH, Sub-WATCH, or STOP mode, or if bit SDC.SDCR is set to 1.

[a] Before enabling PLLEN or SCEN, make sure that the main oscillator is running and has settled (see also CGSTAT register). The CPU must operate on the sub, internal or main oscillator clock when setting PLLEN or SCEN to 1. Before selecting the SSCG / PLL outputs as clock sources for peripherals, ensure by software that the SSCG / PLL stabilization time has elapsed.The stabilization times are defined in the Data Sheet.

(2) CGSTAT - Clock Generator status register

The 8-bit CGSTAT register is read-only. It indicates the status of the main oscillator and the status of the clock generator after wake-up from power save mode.

Access This register can be read in 8-bit units.

Address FFFF F824$_H$.

Initial Value 0000 1101$_B$. The register is initialized by any reset.

7	6	5	4	3	2	1	0
CMPLPSM	0	0	0	1	1	OSCSTAT	1
R	R	R	R	R	R	R	R

Table 4-5 CGSTAT register contents

Bit position	Bit name	Function
7	CMPLPSM	Completed power save mode entry: 0: Power save mode configuration not completed. 1: Power save mode configuration completed. This bit is cleared when the clock generator has accepted a power save mode request. However if CGSTAT.CMLPSM was already 0 before a power save mode request it can not be used as an indicator that the clock generator has accepted this power save mode request. This bit is set when the clock generator has completely set up it's power save mode configuration, i.e. all registers are set up, PLL and SSCG are switched off. However if CGSTAT.CMLPSM was already 1 before a power save mode request it can not be used as the only indicator that the clock generator has completed power save mode configuration. If the clock generator has not accepted a power save mode request this bit remains unchanged. Refer also to “*“CPU operation after power save mode release“ on page 157*”.
1	OSCSTAT	Main oscillator status indicator (determined by counter): 0: Main oscillator has not settled. 1: Main oscillator has stabilized. The OSCSTAT flag is cleared whenever the main oscillator is switched to stand-by mode due to entering the Sub-WATCH or STOP mode. After the main oscillator is restarted, the oscillation stabilization counter will count up from 0 to 40.960 (approx. 10ms @ 4 MHz) to assure stable oscillator operation. When the oscillation stabilization counter reaches 40.960, the counter is stopped, and the OSCSTAT flag is set.

(3) PHCMD - Command protection register

The 8-bit PHCMD register is write-only. It is used to protect other registers from unintended writing.

Access This register must be written in 8-bit units.

Address FFFF $F800_H$.

Initial Value The contents of this register is undefined.

7	6	5	4	3	2	1	0
x	x	x	x	x	x	x	x
W	W	W	W	W	W	W	W

PHCMD protects the registers that may have a significant influence on the application system from inadvertent write access, so that the system does not stop in case of a program hang-up.

Any data written to this register is ignored. Only the write action is monitored.

After writing to the PHCMD register, you are permitted to write once to one of the protected registers. This must be done immediately after writing to the PHCMD register. After the second write action, or if the second write action does not follow immediately, all protected registers are write-locked again.

Caution In case a high level programming language is used, make sure that the compiler translates the two write instructions to PHCMD and the protected register into two consecutive assembler “store” instructions.

With this method, the protected registers can only be rewritten in a specific sequence. Illegal write access to a protected register is inhibited.

The following registers are protected by PHCMD:

– CKC:	Clock control register
– FCC:	FOUTCLK control register
– ICC:	I^2C clock control register
– PCC:	Processor clock control register
– SCC:	SPCLK control register
– TCC:	Watch Timer clock control register
– WCC:	Watchdog timer clock control register
– SDC:	Set default clock register

An invalid write attempt to one of the above registers sets the error flag PHS.PRERR. PHS.PRERR is also set, if a write access to PHCMD is not immediately followed by an access to one of the protected registers.

(4) PHS - Peripheral status register

The 8-bit PHS register indicates the status of a write attempt to a register protected by PHCMD (see also *“PHCMD - Command protection register“ on page 111*).

Access This register can be read/written in 8-bit units.

Address FFFF F802$_H$.

Initial Value 00$_H$. The register is cleared by any reset.

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	PRERR
R[a]	R[a]	R[a]	R[a]	R[a]	R[a]	R[a]	R/W

[a] These bits may be written, but write is ignored.

Table 4-6 PHS register contents

Bit position	Bit name	Function
0	PRERR	Write error status: 0: Write access was successful. 1: Write access failed. You can clear this register by writing 0 to it. Setting this register to 1 by software is not possible.

Note PHS.PRERR is set, if a write access to register PHCMD is not directly followed by a write access to one of the write-protected registers.

(5) PCC - Processor clock control register

The 8-bit PCC register controls the CPU clock. This register can be changed only once after reset or power save mode release.

Access This register can be read/written in 8-bit units.

Writing to this register is protected by a special sequence of instructions. Please refer to *“PHCMD - Command protection register“ on page 111* for details.

Address FFFF $F828_H$.

Initial Value 10_H. The register is initialized by any reset.

7	6	5	4	3	2	1	0
FRC	0	MFRC	CLS	0	SOSCP	CKS1	CKS0
R/W	R[a]	R/W	R[a]	R[a]	R/W	R/W	R/W

[a] These bits may be written, but write is ignored.

Table 4-7 PCC register contents (1/2)

Bit position	Bit name	Function
7	FRC	Sub oscillator circuit: Control of internal return resistance 0: Resistor connected. 1: Resistor disconnected. Set FRC only to 1, if the sub oscillator is not used.
5	MFRC	Main oscillator circuit: Control of internal return resistance 0: Resistor connected. 1: Resistor disconnected. Do not change the initial setting. To ensure correct operation of the main oscillator, the internal feed-back resistor must remain connected.
4	CLS	Processor clock source monitor flag: 0: Main oscillator operation—source can be the output of main oscillator, PLL, or SSCG (selection through CKS[1:0]). The main oscillator is enabled by the internal firmware. 1: Sub clock operation: 32 kHz sub or 240kHz internal oscillator (selection through bit SOSCP). This is the default after reset. It is not possible to set this bit to 1 by writing to the register. On Sub-WATCH release, the CLS bit is set to the state of PSM.OSCDIS. This is the only way to set CLS to 1, which means, the main oscillator remains stopped and the CPU is supplied with the sub clock chosen by SOSCP. CLS is automatically cleared when the processor clock source is changed by writing to PCC.CKS[1:0]. If CLS is 1, the bits CKS[1:0] have no meaning.
2	SOSCP	Sub clock selection: 0: internal oscillator is used for sub clock operation. 1: sub oscillator is used for sub clock operation. This setting takes effect when bit CLS is 1. **Caution:** Do not specify the sub oscillator, if the sub oscillator is not enabled or not connected.

Table 4-7 PCC register contents (2/2)

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>1 to 0</td><td>CKS[1:0]</td><td>Processor clock connection:
<table>
<tr><th>CKS1</th><th>CKS0</th><th>Selected clock connection</th></tr>
<tr><td>0</td><td>0</td><td>Main oscillator</td></tr>
<tr><td>0</td><td>1</td><td>SSCG</td></tr>
<tr><td>1</td><td>0</td><td>PLL (main oscillator frequency x4)</td></tr>
<tr><td>1</td><td>1</td><td>PLL (main oscillator frequency x8)</td></tr>
</table>
As long as PCC.CLS = 1 these bits are ignored. For changing the processor clock source these bits must be written. By this CLS is set to 0 automatically.</td></tr>
</table>

Note

1. Switching to an unstable or not available clock is not protected by hardware. You must monitor the CGSTAT register or count the required stabilization time by software before switching to make sure not to select an unstable clock source.
   Ensure also that the stabilization times of the PLL and SSCG (refer to the Data Sheet) have elapsed before using any PLL or SSCG output clock.
2. Switching to sub clock after Sub-WATCH and WATCH mode release or writing 1 to SDC.SDCR is monitored in the CLS flag. The CLS flag can not be changed to 1 by software.
3. FRC, MFRC and SOSCP are not changed when power save modes are entered or released.

Write protection

Write protection of this register is achieved by following both conditions:

- The register can be written only once after any reset.
- The register is protected by a special sequence via the PHCMD register.
  A fail of a write by the special sequence is reflected by PHS.PRERR = 1.

If a write is correctly performed by the special sequence after the register has already once been written successfully PHS.PRERR remains 0, though the write has been ignored.

PHS.PRERR shows violations of the special sequence only. It does not reflect attempts to write the register more than once after reset or power save mode wake-up.

(6) SDC - Set default clock register

The 8-bit SDC register can be used to reset the Clock Generator to default state. This is the state that is set after power save mode release.

Depending on the flags PSM.OSCDIS and PCC.SOSCP, the main, sub, or internal oscillator becomes the CPU clock source. Both PLL and SSCG are disabled, and all CPU and peripheral clock selections as well as the SSCG setup can be rewritten.

Access This register can be read/written in 8-bit units.

Writing to this register is protected by a special sequence of instructions. Please refer to *“PHCMD - Command protection register“ on page 111* for details.

Address FFFF $F83C_H$.

Initial Value 00_H. The register is cleared by any reset.

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	SDCR
R[a]	R[a]	R[a]	R[a]	R[a]	R[a]	R[a]	R/W

[a] These bits may be written, but write is ignored.

Table 4-8 SDC register contents

Bit position	Bit name	Function
0	SDCR	Set default Clock Generator configuration: 0: Normal operation. 1: Establish default clock settings. The bit SDC.SDCR can be set by writing 1. Clearing SDC.SDCR by writing 0 is not possible, but is done automatically after default clock settings are completed.

Setting SDC.SDCR has the following effects:

- SDC.SDCR remains set until the default clock setting procedure has finished. After that, it is automatically cleared.
- Depending on the bits PSM.OSCDIS and PCC.SOSCP, either main, sub, or internal oscillator is chosen as the clock source of the CPU.
- CKC.PERIC is cleared—the main oscillator is the clock source for PCLK0/1.
- SCC.SPSEL[1:0] is cleared—the main oscillator is the clock source for all SPCLK clocks.
- ICC.IICSEL[1:0] is cleared—the main oscillator is the clock source for IICLK.
- CKC.PLLEN and CKC.SCEN are cleared—but PLL and SSCG are not stopped.

Note

1. For further information concerning default clock setting refer to *“Power save mode activation“ on page 154*.
2. As long as SDC.SDCR is set, do not access any Clock Generator register except SDC.

4.2.2 SSCG control registers

This section describes the registers used for controlling the spread spectrum Clock Generator SSCG.

For modulating the SSCG output clock it's dithering mode must be enabled by CKC.DEN = 1.

Reconfiguration of SSCG registers

The SSCG control registers SCFC0, SCFC1 and SCFMC can only be rewritten with new settings if the SSCG is switched off, i.e. if

- the SSCG is disabled by CKC.SCEN = 0
- the SSCG is safely switched off after a power save mode wake-up. Refer to *"CPU operation after power save mode release" on page 157* for a procedure to ensure that the SSCG is switched off after wake-up.

During operation of the SSCG the registers may only be rewritten with the values, they already have.

(1) SCFC0 - SSCG frequency control register 0

The 8-bit SCFC0 register controls the frequency modulation of the SSCG. It determines the SSCG output frequency and is used in conjunction with register SCFC1.

The center SSCG output frequency is f_{SSCGc} = (4 MHz × N/M) / 2. This register defines the divisor “m” and thus M = m + 1.

Access This register can be read/written in 8-bit or 1-bit units.

Address FFFF $F82C_H$.

Initial Value 52_H. The register is initialized by any reset.

7	6	5	4	3	2	1	0
0[a]	SCFC06	SCFC05	SCFC04	SCFC03	SCFC02	SCFC01	SCFC00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

[a] The default value “0” of this bit must not be altered.

Table 4-9 SCFC0 register contents

Bit position	Bit name	Function
6 to 5	SCFC0[6:5]	Must be set to 01_B
4 to 3	SCFC0[4:3]	Must be set according to *Table 4-10*
2 to 0	SCFC0[2:0]	Determines the divisor m

Note

1. This register can only be rewritten with a new value if the SSCG is switched off. Refer to the explanation at the beginning of this section.
2. The initial value of this register must be changed after reset.

Frequency calculation If dithering mode is disabled (CKC.DEN = 0) the SSCG outputs its center frequency f_{SSCGc}:

$$f_{SSCGc} = (4\text{ MHz } \times N/M) / 2$$

where:

- M = m + 1 = SCFC0.SCFC0[2:0] + 1
- N = n + 1 = SCFC1.SCFC1[6:0] + 1

The values to be written into SCFC0 and SCFC1 are restricted. Possible combinations are:

Table 4-10 Supported settings of N (n) and M (m)

$f_{SSCGmax}$	M (m)	N (n)	SCFC0	SCFC1
48 MHz[a]	4 (3)	96 (95)	$2B_H$	DF_H

[a] The 48 MHz SSCG output frequency has to be divided by the SSCG post scaler. Thus set SCPS = 21_H for 24 MHz, SCPS = 22_H for 16 MHz, SCPS = 23_H for 12 MHz or SCPS = 25_H for 8 MHz operation.

(2) SCFC1 - SSCG frequency control register 1

The 8-bit SCFC1 register controls the frequency multiplication of the SSCG. It determines the SSCG output frequency and is used in conjunction with register SCFC0.

The center SSCG output frequency is $f_{SSCGc} = (4\ \text{MHz} \times N/M) / 2$. This register defines the factor “n” and thus N = n + 1.

Access This register can be read/written in 8-bit or 1-bit units.

Address FFFF $F82E_H$.

Initial Value EB_H. The register is initialized by any reset.

7	6	5	4	3	2	1	0
1	SCFC16	SCFC15	SCFC14	SCFC13	SCFC12	SCFC11	SCFC10
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 4-11 SCFC1 register contents

Bit position	Bit name	Function
6 to 0	SCFC1[6:0]	Determines the factor n

Note

1. Bits 7 is set to 1 and must not be changed.
2. This register can only be rewritten with a new value if the SSCG is switched off. Refer to the explanation at the beginning of this section.

(3) SCFMC - SSCG frequency modulation control register

The 8-bit SCFMC register controls the frequency modulation of the SSCG in dithering mode (when CKC.DEN = 1).

Access This register can be read/written in 8-bit or 1-bit units.

Address FFFF F82A_H.

Initial Value 00_H. The register is initialized by any reset.

7	6	5	4	3	2	1	0
0	0	0	SCFMC4	SCFMC3	SCFMC2	SCFMC1	SCFMC0
R	R	R	R/W	R/W	R/W	R/W	R/W

Table 4-12 SCFMC register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>4 to 2</td><td>SCFMC[4:2]</td><td>Frequency modulation range control:
<table>
<tr><th>SCFMC4</th><th>SCFMC3</th><th>SCFMC2</th><th>FM range</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>± 0.5 % (typical value)</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>± 1.0 % (typical value)</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>± 2.0 % (typical value)</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>± 3.0 % (typical value)</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>± 4.0 % (typical value)</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>± 5.0 % (typical value)</td></tr>
<tr><td colspan="3">other settings</td><td>prohibited</td></tr>
</table></td></tr>
<tr><td>1 to 0</td><td>SCFMC[1:0]</td><td>Frequency modulation frequency control:
<table>
<tr><th>SCFMC1</th><th>SCFMC0</th><th>Modulation frequency</th></tr>
<tr><td>0</td><td>0</td><td>40 kHz (typical value)</td></tr>
<tr><td>0</td><td>1</td><td>50 kHz (typical value)</td></tr>
<tr><td>1</td><td>0</td><td>60 kHz (typical value)</td></tr>
<tr><td>1</td><td>1</td><td>prohibited</td></tr>
</table></td></tr>
</table>

Note

1. This register can only be rewritten with a new value if the SSCG is switched off. Refer to the explanation at the beginning of this section.
2. The given modulation ranges and frequencies are typical values. Refer also to the Data Sheet.

In dithering mode, the SSCG output frequency f_{SSCG} varies according to the FM range, specified by SCFMC[4:2], around it's center value:

$$f_{SSCG} = f_{SSCGc} \pm \text{(FM range)}$$

The time of one full cycle is given by the period of the modulation frequency specified in SCFMC[1:0].

Example If:

- SCFC0 = $2B_H$, SCFC1 = DF_H: center frequency f_{SSCGc} = 48 MHz
- [SCFMC[4:2]] = 101_B: FM range = 5 %
- [SCFMC[1:0]] = 01_B: modulation frequency = 50 KHz

Then:

- The SSCG frequency is swept between about 45.6 MHz and 50.4 MHz.
- One sweep cycle takes typically 20 µs.

(4) SCPS - SSCG post scaler control register

The 8-bit SCPS register controls the two independent SSCG post scalers (frequency dividers) for the CPU system clock VBCLK and for the modulated peripheral clocks SPCLK.

Access This register can be read/written in 8-bit or 1-bit units.

Address FFFF F830$_H$.

Initial Value 21$_H$. The register is initialized by any reset.

7	6	5	4	3	2	1	0
0	SPSPS2	SPSPS1	SPSPS0	0	VBSPS2	VBSPS1	VBSPS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 4-13 SCPS register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>6 to 4</td><td>SPSPS[2:0]</td><td>SSCG clock divider selection for generating SPCLK0:
<table>
<tr><th>SPSPS2</th><th>SPSPS1</th><th>SPSPS0</th><th>Clock divider setting</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>SPCLK0 = SSCG out frequency / 1</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>SPCLK0 = SSCG out frequency / 2</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>SPCLK0 = SSCG out frequency / 3</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>SPCLK0 = SSCG out frequency / 4</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>not supported</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>SPCLK0 = SSCG out frequency / 6</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>not supported</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>SPCLK0 = SSCG out frequency / 8</td></tr>
</table></td></tr>
<tr><td>2 to 0</td><td>VBSPS[2:0]</td><td>SSCG clock divider selection for generating VBCLK:
<table>
<tr><th>VBSPS2</th><th>VBSPS1</th><th>VBSPS0</th><th>Clock divider setting</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>prohibited</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>VBCLK = SSCG out frequency / 2</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>VBCLK = SSCG out frequency / 3</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>VBCLK = SSCG out frequency / 4</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>not supported</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>VBCLK = SSCG out frequency / 6</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>not supported</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>VBCLK = SSCG out frequency / 8</td></tr>
</table></td></tr>
</table>

Note This register can only be written when the SSCG enable bit CKC.SCEN is cleared (SSCG switched off).

4.2.3 Control registers for peripheral clocks

This section describes the registers used for specifying the sources and operation modes for the clocks provided for the on-chip peripherals.

These clocks are the clocks for the Watchdog and Watch Timers, the SPCLKn clocks, FOUTCLK, and IICLK.

Note Be aware that the WCC register controls not only the generation of the Watchdog Timer clock. It defines also the run/stop mode of the sub and internal oscillators when certain power save modes are entered.

(1) WCC - Watchdog Timer clock control register

The 8-bit WCC register controls the Watchdog Timer clock. This register can be changed only once after any reset.

Writing to this register is protected by a special sequence of instructions. Please refer to *“PHCMD - Command protection register“ on page 111* for details.

Access This register can be read/written in 8-bit units.

Address FFFF F826$_H$.

Initial Value 00$_H$. The register is initialized by any reset.

7	6	5	4	3	2	1	0
SOSTP	WPS2	WPS1	WPS0	ROSTP	SOSCW	WDTSEL1	WDTSEL0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 4-14 WCC register contents (1/2)

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>7</td><td>SOSTP</td><td>Sub oscillator STOP mode control
1: Sub oscillator will stop when STOP mode is entered.
0: Sub oscillator will not stop when STOP mode is entered.</td></tr>
<tr><td>6 to 4</td><td>WPS[2:0]</td><td>WDT clock divider selection:
<table>
<tr><th>WPS2</th><th>WPS1</th><th>WPS0</th><th>Clock divider setting</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1 / 2</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1 / 4</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>1 / 8</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>1 / 16</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>1 / 32</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>1 / 64</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1 / 128</td></tr>
</table></td></tr>
<tr><td>3</td><td>ROSTP</td><td>Internal oscillator stop control:
1: Internal oscillator stops if WATCH, Sub-WATCH or STOP mode is entered
0: Internal oscillator always operates</td></tr>
</table>

Table 4-14 WCC register contents (2/2)

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>2, 0</td><td>SOSCW,
WDTSEL0</td><td>Watchdog Timer clock source selection:
<table>
<tr><th>SOSCW</th><th>WDTSEL0</th><th>WDT clock source</th></tr>
<tr><td>0</td><td>0</td><td>Internal oscillator</td></tr>
<tr><td>1</td><td>0</td><td>Sub oscillator</td></tr>
<tr><td>0</td><td>1</td><td>Main oscillator</td></tr>
<tr><td>1</td><td>1</td><td>Setting prohibited</td></tr>
</table>
By default, the sub oscillator is disabled in STOP mode (see bit SOSTP). If SOSTP is 1, choose main or internal oscillator before entering STOP mode.
Caution: Do not specify the sub oscillator, if the sub oscillator is not enabled or not connected.</td></tr>
<tr><td>1</td><td>WDTSEL1</td><td>Watchdog Timer clock stand-by control
0: WDTCLK stops in IDLE, WATCH, Sub-WATCH and STOP modes.
1: WDTCLK operates as long as the selected clock source operates.</td></tr>
</table>

Note

1. For security reasons, the WCC register should always be programmed after reset, even if the default settings are used.
2. Watch and Watchdog Timer clocks are not gated during the sub oscillator stabilization period after STOP-mode release. This may generate spikes on the clock supply of the watch and Watchdog Timers.

Write protection

Write protection of this register is achieved in two ways:

- The register can be written only once after Power-On-Clear reset or external $\overline{\text{RESET}}$.
- The register is protected by a special sequence via the PHCMD register.
  A fail of a write by the special sequence is reflected by PHS.PRERR = 1.

If a write is correctly performed by the special sequence after the register has already once been written successfully PHS.PRERR remains 0, though the write has been ignored.

PHS.PRERR shows violations of the special sequence only. It does not reflect attempts to write the register more than once after reset.

(2) TCC - Watch Timer clock control register

The 8-bit TCC register determines the Watch Timer and LCD controller clock source and the setting of the associated clock dividers. This register can be changed only once after Power-On-Clear reset or external $\overline{\text{RESET}}$.

Access This register can be read/written in 8-bit units.

Writing to this register is protected by a special sequence of instructions. Please refer to *“PHCMD - Command protection register“ on page 111* for details.

Address FFFF $F836_H$.

Initial Value 00_H. The register is initialized at power-on and by external $\overline{\text{RESET}}$.

7	6	5	4	3	2	1	0
0	WTPS2	WTPS1	WTPS0	0	WTSOS	WTSEL1	WTSEL0
R[a]	R/W	R/W	R/W	R[a]	R/W	R/W	R/W

[a] These bits may be written, but write is ignored.

Table 4-15 TCC register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>6 to 4</td><td>WTPS[2:0]</td><td>LCDCLK clock divider selection::
<table>
<tr><th>WTPS2</th><th>WTPS1</th><th>WTPS0</th><th>Clock divider setting</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1 / 2</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1 / 4</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>1 / 8</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>1 / 16</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>1 / 32</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>1 / 64</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1 / 128</td></tr>
</table></td></tr>
<tr><td>1</td><td>WTSEL1</td><td>WTCLK (Watch Timer clock) divider setting:
0: WTCLK = LCDCLK.
1: WTCLK = LCDCLK / 2.</td></tr>
<tr><td>2, 0</td><td>WTSOS,
WTSEL0</td><td>Clock source for Watch Timer and LCD controller:
<table>
<tr><th>WTSOS</th><th>WTSEL0</th><th>Clock source</th></tr>
<tr><td>0</td><td>0</td><td>Internal oscillator</td></tr>
<tr><td>1</td><td>0</td><td>Sub oscillator</td></tr>
<tr><td>0</td><td>1</td><td>Main oscillator</td></tr>
<tr><td>1</td><td>1</td><td>Setting prohibited</td></tr>
</table>
By default, the sub oscillator is disabled in STOP mode (see bit WCC.SOSTP). If WCC.SOSTP is 1, choose main or internal oscillator before entering STOP mode.
<b>Caution:</b> Do not specify the sub oscillator, if the sub oscillator is not enabled or not connected.</td></tr>
</table>

Note Only POC and external $\overline{\text{RESET}}$ can clear the TCC register. Only one write access to TCC is allowed after reset release. Once the TCC has been written, it ignores new write accesses until the next POC or external $\overline{\text{RESET}}$ is issued.

Write protection Write protection of this register is achieved in two ways:

- The register can be written only once after Power-On-Clear reset or external $\overline{\text{RESET}}$.
- The register is protected by a special sequence via the PHCMD register.
  A fail of a write by the special sequence is reflected by PHS.PRERR = 1.

If a write is correctly performed by the special sequence after the register has already once been written successfully PHS.PRERR remains 0, though the write has been ignored.

PHS.PRERR shows violations of the special sequence only. It does not reflect attempts to write the register more than once after reset.

(3) SCC - SPCLK control register

The 8-bit SCC register selects the SPCLK sources.

Access This register can be read/written in 8-bit or 1-bit units.

Writing to this register is protected by a special sequence of instructions. Please refer to *“PHCMD - Command protection register“ on page 111* for details.

Address FFFF F832$_H$.

Initial Value 00$_H$. The register is initialized by entering WATCH, Sub-WATCH, or STOP mode, or if control bit SDC.SDCR is set.

7	6	5	4	3	2	1	0
0	0	0	0	0	0	SPSEL1	SPSEL0
R	R	R	R	R	R	R/W	R/W

Table 4-16 SCC register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>1 to 0</td><td>SPSEL[1:0]</td><td>Source selection for generating the SPCLK clocks:
<table>
<tr><th rowspan="2">SPSEL1</th><th rowspan="2">SPSEL0</th><th colspan="3">Clock sources</th></tr>
<tr><th>SPCLK0</th><th>SPCLK1</th><th>SPCLK2</th></tr>
<tr><td>0</td><td>0</td><td>Main osc</td><td>Main osc</td><td>Main osc</td></tr>
<tr><td>0</td><td>1</td><td>PLL / 2</td><td>PLL / 4</td><td>Main osc</td></tr>
<tr><td>1</td><td>0</td><td colspan="3">not supported</td></tr>
<tr><td>1</td><td>1</td><td>SSCG_{PS}</td><td>SSCG_{PS} / 2</td><td>SSCG_{PS} / 4</td></tr>
</table>
</td></tr>
</table>

Note 1. “Main osc” is the clock MOCLK provided by the main oscillator.

2. “PLL” is the clock PLLCLK provided by the PLL.

“SSCG_{PS}“ is the clock provided by the SSCG post scaler for SPCLK (see also *“SCPS - SSCG post scaler control register“ on page 121*).

(4) FCC - FOUTCLK control register

The 8-bit FCC register configures the output clock FOUTCLK that can be used for external devices.

Access This register can be read/written in 8-bit or 1-bit units.

Writing to this register is protected by a special sequence of instructions. Please refer to *“PHCMD - Command protection register“ on page 111* for details.

Address FFFF F834$_H$.

Initial Value 00$_H$. The register is initialized by any reset.

7	6	5	4	3	2	1	0
FOEN	FOCS2	FOCS1	FOCS0	0	FOSOS	FOCKS1	FOCKS0
R/W	R/W	R/W	R/W	R	R/W	R/W	R/W

Table 4-17 FCC register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>7</td><td>FOEN</td><td>Output clock FOUTCLK enable:
0: FOUTCLK is disabled.
1: FOUTCLK is enabled.</td></tr>
<tr><td>6 to 4</td><td>FOCS[2:0]</td><td>Output clock divider setting for FOUTCLK:
<table>
<tr><th>FOCS2</th><th>FOCS1</th><th>FOCS0</th><th>Clock divider setting</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>FOUTCLK = selected clock source / 1</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>FOUTCLK = selected clock source / 2</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>FOUTCLK = selected clock source / 4</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>FOUTCLK = selected clock source / 8</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>FOUTCLK = selected clock source / 16</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>FOUTCLK = selected clock source / 32</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>FOUTCLK = selected clock source / 64</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>FOUTCLK = selected clock source / 128</td></tr>
</table></td></tr>
<tr><td>2 to 0</td><td>FOSOS,
FOCKS[1:0]</td><td>Clock source selection for FOUTCLK:
<table>
<tr><th>FOSOS</th><th>FOCKS1</th><th>FOCKS0</th><th>Clock source</th></tr>
<tr><td>x</td><td>0</td><td>0</td><td>Main oscillator</td></tr>
<tr><td>x</td><td>0</td><td>1</td><td>SSCG</td></tr>
<tr><td>x</td><td>1</td><td>0</td><td>PLL</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>Internal oscillator</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>Sub oscillator</td></tr>
</table>
<b>Caution:</b> Do not specify the sub oscillator, if the sub oscillator is not enabled or not connected.</td></tr>
</table>

Note

1. FOUTCLK is not influenced by stand-by modes of the microcontroller. It runs as long as it is enabled and the selected clock source operates. Application software must stop FOUTCLK by clearing the FOEN bit to minimize power consumption in stand-by modes.
2. There is an upper frequency limit for the output buffer of the FOUTCLK function. Do not select a frequency higher than the maximum output buffer frequency. Please refer to the Data Sheet for the frequency limit.

(5) ICC - IIC clock control register

The 8-bit ICC register determines the I^{2}C clock source and the clock divider setting for IICLK.

Access This register can be read/written in 8-bit units.

Writing to this register is protected by a special sequence of instructions. Please refer to *“PHCMD - Command protection register“ on page 111* for details.

Address FFFF F838$_H$.

Initial Value 00$_H$. The register is cleared by any reset.

7	6	5	4	3	2	1	0
0	IICPS2	IICPS1	IICPS0	0	0	IICSEL1	IICSEL0
R[a]	R/W	R/W	R/W	R[a]	R[a]	R/W	R/W

[a] These bits may be written, but write is ignored.

Table 4-18 ICC register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>6 to 4</td><td>IICPS[2:0]</td><td>Divider setting for IICLK:
<table>
<tr><th>IICPS2</th><th>IICPS1</th><th>IICPS0</th><th>Clock divider setting</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>1 / 3.5</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1 / 4.5</td></tr>
<tr><td colspan="3">other settings</td><td>not supported</td></tr>
</table></td></tr>
<tr><td>1 to 0</td><td>IICSEL[1:0]</td><td>Clock source for IICLK:
<table>
<tr><th>IICSEL1</th><th>IICSEL0</th><th>Clock source</th></tr>
<tr><td>0</td><td>0</td><td>Main oscillator</td></tr>
<tr><td>0</td><td>1</td><td>SSCG / 2</td></tr>
<tr><td>1</td><td>x</td><td>PLL</td></tr>
</table></td></tr>
</table>

Note

1. On release of WATCH, Sub-WATCH and STOP mode or when the SDC.SDCR bit is set, IICSEL[1:0] is cleared—the main oscillator is selected as the IIC clock source.
Pay attention if PSM.OSCDIS = 1 before entering any of the above power save modes, because the main oscillator will be disabled. Therefore the I^{2}C interface will have no clock supply after power save mode release.

2. The connected I^{2}C interfaces must be disabled before switching IICPS[2:0]. To switch the IICPS bits, first disable the I^{2}C interface by clearing the enable bit in the IIC control register, then switch IICPS[2:0] and finally re-enable the IIC interface.

4.2.4 Control registers for power save modes

The registers described in this section control the begin and end of the power save modes IDLE, WATCH, Sub-WATCH, and STOP.

Please refer to *“Power save mode activation“ on page 154* for instructions and an example on how to enter a power save mode.

(1) PSM - Power save mode register

The 8-bit PSM register specifies the power save mode and controls the clock generation after reset and Sub-WATCH mode release. In addition, it specifies the source of the Watch Calibration Timer clock WCTCLK.

Access This register can be read/written in 8-bit or 1-bit units.

Address FFFF $F820_H$.

Initial Value 08_H. The register is initialized by any reset.
Since the main oscillator is started by the internal firmware are reset, PSM enters the user’s program with the setting 00_H.

7	6	5	4	3	2	1	0
0	CMODE	0	0	OSCDIS	0	PSM1	PSM0
R	R/W	R	R	R/W	R	R/W	R/W

Table 4-19 PSM register contents (1/3)

Bit position	Bit name	Function
6	CMODE	Watch Calibration Timer clock selection: 0: PCLK1. 1: Main oscillator.

Table 4-19 PSM register contents (2/3)

Bit position	Bit name	Function
3	OSCDIS	Main oscillator disable/enable control during and after power save mode: 0: Main oscillator enabled. 1: Main oscillator disabled. **Caution:** If OSCDIS is set to 1, the main oscillator clock supply for the Watch Timer and the LCD Controller/Driver are stopped immediately. Thus these function stop their operation immediately as well, when the main oscillator is used as the clock source. OSCDIS determines also the behaviour of the main oscillator during and after power save mode. The effect of this bit differs, depending on the power save mode. • Sub-WATCH mode During Sub-WATCH mode the main oscillator is always stopped. OSCDIS determines whether the main oscillator shall be started and chosen as CPU clock source or should remain stopped after Sub-WATCH mode release. 0: Main oscillator enable. The main oscillator is started after Sub-WATCH mode release and the CPU is supplied with the main oscillator clock, after the oscillation stabilization time has elapsed. 1: Main oscillator disable. The main oscillator remains stopped after Sub-WATCH release. The CPU is supplied with the selected sub clock—either sub oscillator or internal oscillator (see bit PCC.SOSCP). Since the reset value of OSCDIS is 1 and PCC.SOSCP is 0 the CPU starts always with the internal oscillator clock after reset release. In both cases, the application software must start the main oscillator by clearing the OSCDIS bit. After the oscillator stabilization time has elapsed (see bit CGSTAT.OSCSTAT), the main oscillator can be used as system clock source by setting the PCC register accordingly. • WATCH mode This bit determines whether the main oscillator shall be stopped or remain in operation during WATCH mode. In either case after WATCH mode release the CPU is operating on the main oscillator. 0: Main oscillator enable. The main oscillator is operating during WATCH mode. After WATCH mode release the CPU is supplied with the main oscillator clock. 1: Main oscillator disable. The main oscillator is stopped during WATCH mode. After WATCH mode release the main oscillator is started and the CPU is supplied with the main oscillator clock, after the oscillation stabilization time has elapsed. **Note:** In case the main oscillator is chosen as the CPU clock after power save mode release (i.e. after Sub-WATCH mode release with OSCDIS = 0 or after WATCH mode release) the start-up phase of the CPU differs depending on the history of the main oscillator status indicator CGSTAT.OSCSTAT. – main oscillator never used before If the main oscillator has never been stable before entering and releasing power save mode (CGSTAT.OSCSTAT has never been set to 1), the CPU starts operation on the internal oscillator. After the main oscillator has become stable, it is used as the CPU clock. – main oscillator already used before If the main oscillator has already been stable before entering and releasing power save mode (CGSTAT.OSCSTAT has already been set to 1), the CPU starts operation on main oscillator, after the main oscillator has become stable.

Table 4-19 PSM register contents (3/3)

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>1 to 0</td><td>PSM[1:0]</td><td>Power save mode selection:
<table>
<tr><th>PSM1</th><th>PSM0</th><th>Power save mode</th></tr>
<tr><td>0</td><td>0</td><td>IDLE</td></tr>
<tr><td>0</td><td>1</td><td>STOP</td></tr>
<tr><td>1</td><td>0</td><td>WATCH</td></tr>
<tr><td>1</td><td>1</td><td>Sub-WATCH mode (main oscillator shut down)</td></tr>
</table>
It is not possible to switch to IDLE or WATCH mode when the CPU is operated by a sub clock. If IDLE or WATCH mode is selected during sub clock operation, the Sub-WATCH mode will be entered.</td></tr>
</table>

(2) PSC - Power save control register

The 8-bit PSC register is used to enter or leave the power save mode specified in register PSM.

Access This register can be read/written in 8-bit or 1-bit units.

Writing to this register is protected by a special sequence of instructions. Please refer to *“PRCMD - PSC write protection register“ on page 134* for details.

Address FFFF F1FE$_H$.

Initial Value 00$_H$. The register is cleared by any reset.

7	6	5	4	3	2	1	0
0	NMIWDTM	NMI0M	INTM	0	0	STP	0
R	R/W	R/W	R/W	R	R	R/W	R

Table 4-20 PSC register contents

Bit position	Bit name	Function
6	NMIWDTM	Mask for non-maskable interrupt request from WDT: 0: Permit NMIWDT request during power save mode. 1: Prohibit NMIWDT request during power save mode.
5	NMI0M	Mask for non-maskable interrupt request 0: 0: Permit external NMI0 request during power save mode 1: Prohibit external NMI0 request during power save mode.
4	INTM	Mask for maskable interrupt request: 0: Permit maskable interrupt requests during power save mode.[a] 1: Prohibit maskable interrupt requests during power save mode.
1	STP	Enter/release power save mode: 0: Power save mode is released. 1: Power save mode is entered.

[a] Only dedicated maskable interrupts have wake-up capability, refer to *“Power save modes description“ on page 141*.

Note

1. If bits 7, 3, 2, and 0 are not set to 0, proper operation of the controller can not be guaranteed.
2. PSC.STP is automatically cleared when the controller is awakened from power save mode.
3. Entering a power save mode requires some attention, refer to *“Power save mode activation“ on page 154*.

(3) PRCMD - PSC write protection register

The 8-bit PRCMD register protects the register PSC from inadvertent write access, so that the system does not stop in case of a program hang-up.

After data has been written to the PRCMD register, the first write access to register PSC is valid. All subsequent write accesses are ignored. Thus, the value of PSC can only be rewritten in a specified sequence, and illegal write access is inhibited.

Access This register can only be written in 8-bit units.

Address FFFF F1FC$_H$

Initial Value The contents of this register is undefined.

7	6	5	4	3	2	1	0
x	x	x	x	x	x	x	x
W	W	W	W	W	W	W	W

Caution Before writing to PRCMD, make sure that all DMA channels are disabled. Otherwise, a direct memory access could occur between the write access to PRCMD and the write access to PSC. If that happens, the power save mode may not be entered.

Caution In case a high level programming language is used, make sure that the compiler translates the two write instructions to PRCMD and PSC into two consecutive assembler “store” instructions.

(4) STBCTL- Stand-by control register

The 8-bit STBCTL register is used to control the stand-by function of the voltage regulators.

Access This register can be read/written in 8-bit or 1-bit units.

Writing to this register is protected by a special sequence of instructions. Please refer to *“STBCTLP - Stand-by control protection register“ on page 135* for details.

Address FFFF FCA2$_H$.

Initial Value 00$_H$. The register is cleared by any reset.

7	6	5	4	3	2	1	0
0	0	0	0	0	0	STYCD	STBYMD
R	R	R	R	R	R	R/W	R/W

Table 4-21 STBCTL register contents

Bit position	Bit name	Function
1	STBYCD	Enable stand-by function of VDD50 and VDD51 voltage regulators: 0: Stand-by function disabled 1: Stand-by function enabled
0	STBYMD	Enable stand-by function of VDD52 voltage regulator: 0: Stand-by function disabled 1: Stand-by function enabled

In order to reduce the power consumption during power save modes the stand-by function of the voltage regulators should be enabled during the initialization.

If a dedicated microcontroller does not include any of the voltage regulators dedicated to the controls bit STBCTL.STBYCD andSTBCTL.STBYMD, the status of the control bit has no function. Thus the initialization for enabling the stand-by functions by STBCTL = 03$_H$ can be retained. For further details concerning voltage regulators refer to *“Power Supply Scheme“ on page 947*.

(5) STBCTLP - Stand-by control protection register

The 8-bit STBCTLP register protects the register STBCTL from inadvertent write access.

After data has been written to the STBCTLP register, the first write access to register STBCTL is valid. All subsequent write accesses are ignored. Thus, the value of STBCTL can only be rewritten in a specified sequence, and illegal write access is inhibited.

Access This register can only be written in 8-bit units.

Address FFFF FCAA$_H$

Initial Value The contents of this register is undefined.

7	6	5	4	3	2	1	0
X	X	X	X	X	X	X	X
W	W	W	W	W	W	W	W

4.2.5 Clock monitor registers

The following registers are used to control the monitor circuits of the main oscillator clock and the sub oscillator clock.

Please refer to *“Operation of the Clock Monitors“ on page 162* for supplementary information.

(1) CLMM - Main oscillator clock monitor mode register

The 8-bit CLMM register is used to enable the monitor for the main oscillator clock.

Access This register can be read/written in 8-bit or 1-bit units.

Writing to this register is protected by a special sequence of instructions. Please refer to *“PRCMDCMM - CLMM write protection register“ on page 137* for details.

Address FFFF F870$_H$.

Initial Value 00$_H$. This register is cleared by any reset.

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	CLMEM
R	R	R	R	R	R	R	R/W

Table 4-22 CLMM register contents

Bit position	Bit name	Function
0	CLMEM	Clock monitor enable: 0: Clock monitor for main oscillator disabled. 1: Clock monitor for main oscillator enabled. This bit can only be cleared by reset.

Note CLMM.CLMEM can be set at any time. However, the clock monitor is only activated after the main oscillator has stabilized, indicated by CGSTAT.OSCSTAT = 1.

(2) PRCMDCMM - CLMM write protection register

The 8-bit PRCMDCMM register protects the register CLMM from inadvertent write access, so that the system does not stop in case of a program hang-up.

After data has been written to the PRCMDCMM register, the first write access to register CLMM is valid. All subsequent write accesses are ignored. Thus, the value of CLMM can only be rewritten in a specified sequence, and illegal write access is inhibited.

Access This register can only be written in 8-bit units.

Address FFFF FCB0$_H$

Initial Value The contents of this register is undefined.

7	6	5	4	3	2	1	0
X	X	X	X	X	X	X	X
W	W	W	W	W	W	W	W

After writing to the PRCMDCMM register, you are permitted to write once to CLMM. The write access to CLMM must happen with the immediately following instruction.

Caution In case a high level programming language is used, make sure that the compiler translates the two write instructions to PRCMDCMM and CLMM into two consecutive assembler “store” instructions.

(3) CLMS - Sub oscillator clock monitor register

The 8-bit CLMS register is used to enable the monitor for the sub oscillator clock.

Access This register can be read/written in 8-bit or 1-bit units.

Writing to this register is protected by a special sequence of instructions. Please refer to *“PRCMDCMS - CLMS write protection register“ on page 139* for details.

Address FFFF F878$_H$.

Initial Value 00$_H$. The register is initialized by any reset.

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	CLMES
R	R	R	R	R	R	R	R/W

Table 4-23 CLMS register contents

Bit position	Bit name	Function
0	CLMES	Clock monitor enable: 0: Clock monitor for sub oscillator disabled. 1: Clock monitor for sub oscillator enabled. This bit can only be cleared by reset.

Note Setting CLMS.CLMES to 1 does not start the sub oscillator clock monitor. To start the clock monitor CLMCS.CMRT has to be set to 1 afterwards.

CLMCS.CMRT must not be set before the sub oscillator has stabilized.

(4) PRCMDCMS - CLMS write protection register

The 8-bit PRCMDCMS register protects the register CLMS from inadvertent write access, so that the system does not stop in case of a program hang-up.

After data has been written to the PRCMDCMS register, the first write access to register CLMS is valid. All subsequent write accesses are ignored. Thus, the value of CLMS can only be rewritten in a specified sequence, and illegal write access is inhibited.

Access This register can only be written in 8-bit units.

Address FFFF FCB2$_H$

Initial Value The contents of this register is undefined.

7	6	5	4	3	2	1	0
X	X	X	X	X	X	X	X
W	W	W	W	W	W	W	W

After writing to the PRCMDCMS register, you are permitted to write once to CLMS. The write access to CLMS must happen with the immediately following instruction.

Caution In case a high level programming language is used, make sure that the compiler translates the two write instructions to PRCMDCMS and CLMS into two consecutive assembler “store” instructions.

(5) CLMCS - Sub oscillator clock monitor control register

The 8-bit CLMCS register is used to start the monitor of the sub oscillator clock.

Access This register can be read/written in 8-bit or 1-bit units.

Address FFFF $F71A_H$.

Initial Value 00_H. The register is initialized by any reset.

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	CMRT
R	R	R	R	R	R	R	R/W

Table 4-24 CLMCS register contents

Bit position	Bit name	Function
0	CMRT	Sub oscillator clock monitor start: 0: Clock monitor for sub oscillator off. 1: Clock monitor for sub oscillator on.

Setting CLMCS.CMRT to 1 generates a trigger to activate the sub oscillator clock monitor.

Note

1. The sub oscillator clock monitor can only be started, if it has been enabled by setting CLMS.CLMES to 1.
2. Make sure that the sub oscillator stabilization time has elapsed before starting the clock monitor.

Caution Starting the sub oscillator clock monitor requires a special procedure. Refer to *“Operation of the Clock Monitors“ on page 162*.

4.3 Power Save Modes

This chapter describes the various power save modes and how they are operated. For details see:

- *“Power save modes description“ on page 141*
- *“Power save mode activation“ on page 154*
- *“CPU operation after power save mode release“ on page 157*

4.3.1 Power save modes description

This section explains the various power save modes in detail.

During power save mode

During all power save modes, the pins behave as follows:

- All output pins retain their function. That means all outputs are active, provided the required clock source is available.
- All input pins remain as input pins.
- All input pins with stand-by wake-up capability remain active, the function of all others is disabled.

During all power save modes, the main and sub oscillator clock monitors remain active, provided that the monitored oscillator is operating. If the oscillator is switched off during stand-by, the associated clock monitor enters stand-by as well.

Wake-up signals

The following signals can awake the controller from power save modes IDLE, WATCH, Sub-WATCH, STOP:

- Reset signals
  - external $\overline{\text{RESET}}$
  - Power-On-Clear reset RESPOC
  - Watchdog Timer reset RESWDT
The Watchdog Timer must be configured to generate the reset WDTRES in case of overflow (WDTM.WDTMODE = 1) and it’s input clock WDTCLK must be active during stand-by.
  - Clock monitors resets RESCMM, RESCMS
The main oscillator respectively sub oscillator must be active during stand-by.
- Non maskable interrupts
  - NMI0
The appropriate port must be configured correctly.
  - NMIWDT
The Watchdog Timer must be configured to generate the in case of overflow (WDTM.WDTMODE = 0) and it’s input clock WDTCLK must be active during stand-by.
- Maskable interrupts
  - external interrupts INTPn
The appropriate port must be configured correctly.
  - CAN wake up interrupts INTCnWUP
The appropriate port and the CAN (CnCTRL.PSMODE[1:0] = 01_B) must be configured correctly.

- Watch Timer interrupts INTWTnUV
  The Watch Timer clock WTCLK must be active and the Watch Timer must be enabled.
- Watch Calibration Timer interrupt INTTM01
  The Watch Calibration Timer clock WCTCLK must be active and the Watch Calibration Timer must be enabled.
- Voltage Comparators interrupts INTVCn
  The Voltage Comparators must be enabled.
- CSIB receive interrupts INTCBnR
  The CSIB must be operated in slave reception mode and the appropriate ports must be configured correctly.

Note that not all these signals are available in all power save modes.

The following signals can awake the controller from the power save mode HALT, provided the appropriate ports and modules are correctly configured and the required clocks are active:

- all reset signals
- the non-maskable interrupts NMI0, NMIWDT
- all maskable interrupts

To grant wake-up capability to maskable interrupts these interrupts have to be unmasked by setting the dedicated mask flags xxMK to 0 (refer to *“Interrupt Controller (INTC)“ on page 180*).

A general disable of maskable interrupts acknowledgement (“DI”, i.e. PSW.ID = 1) does not affect their wake-up capability.

After power save mode

After power save mode release, the clock source for CPU operation should be checked. If the user application issues a wake-up request immediately after power save mode request, the power save mode may not be entered and the clock sources remain as programmed before the stand-by request.

After power save mode release, the same procedure as for system reset is required to set up the clock supply for the application.

Note

In the following tables the clock status "operates" does not necessarily mean that the functions that use this clock source are operating as well.

(1) HALT mode

The HALT mode can be entered from normal run mode. In HALT mode, all clock settings remain unchanged. Only the CPU clock is suspended and hence program execution.

Table 4-25 Clock Generator status in HALT mode

Item	Status	Remarks
Main oscillator	unchanged	
Sub oscillator	operates	
Internal oscillator	operates	
SSCG	unchanged	
PLL	unchanged	
VBCLK (CPU system)	suspended	Clock setup is unchanged
IICLK	unchanged	
PCLK0, PCLK1	unchanged	
PCLK2…PCLK15	unchanged	
SPCLK0, SPCLK1	unchanged	
SPCLK2…SPCLK15	unchanged	
FOUTCLK	unchanged	
WTCLK / LCDCLK	unchanged	
WDTCLK	unchanged	
WCTCLK	unchanged	

The HALT mode can be released by any unmasked maskable interrupt, NMI or system reset.

On HALT mode release, all clock settings remain unchanged. The CPU clock resumes operation.

(2) IDLE mode

The IDLE mode can be entered from any run mode. The main oscillator must be operating. IDLE mode can not be entered if the CPU is clocked by the sub or internal oscillator.

In IDLE mode, the clock distribution is stopped (refer to the “Standby” switches in *Figure 4-1, “Block diagram of the Clock Generator,” on page 101*).

The states of all clock sources, that means, sub and internal oscillator as well as SSCG and PLL, remain unchanged. If a clock source was operating before entering IDLE mode, it continues operating.

Table 4-26 Clock Generator status in IDLE mode

Item	Status	Remarks
Main oscillator	unchanged	
Sub oscillator	operates	
Internal oscillator	operates	
SSCG	unchanged	
PLL	unchanged	
VBCLK (CPU system)	stopped	
IICLK	stopped	
PCLK0, PCLK1	stopped	
PCLK2…PCLK15	stopped	
SPCLK0, SPCLK1	stopped	
SPCLK2…SPCLK15	stopped	
FOUTCLK	unchanged	
WTCLK / LCDCLK	unchanged	
WDTCLK	unchanged/stopped	Stopped if WCC.WDTSEL1 = 0
WCTCLK	unchanged/stopped	Depends on clock selector PSM.CMODE

The IDLE mode can be released by

- the unmasked maskable interrupts INTPn, INTCnWUP INTWTnUV, INTTM01, INTVCn, INTCBnR
- NMI0, NMIWDT
- $\overline{\text{RESET}}$, RESPOC, RESWDT, RESCMM, RESCMS

On IDLE mode release, the CPU clock and peripheral clocks are supplied by the main oscillator.

(3) WATCH mode

In WATCH mode, the clock supply for the CPU system and the majority of peripherals is stopped.

The main oscillator continues operation. PLL and SSCG are stopped. By default, internal oscillator and sub oscillator operation is not affected. For exceptions see *“Internal and sub oscillator operation“ on page 160*.

Depending on register settings, the Watchdog Timer clock supply can continue or stop.

Table 4-27 Clock Generator status in WATCH mode

Item	Status	Remarks
Main oscillator	unchanged/stopped	Stopped if PSM.OSCDIS = 1
Sub oscillator	operates	
Internal oscillator	operates/stopped	Stopped if WCC.ROSTP = 1
SSCG	stopped	
PLL	stopped	
VBCLK (CPU system)	stopped	
IICLK	stopped	
PCLK0, PCLK1	stopped	
PCLK2…PCLK15	stopped	
SPCLK0, SPCLK1	stopped	
SPCLK2…SPCLK15	stopped	
FOUTCLK	unchanged/stopped	Stopped, if the selected clock source stops
WTCLK / LCDCLK	unchanged/stopped	Stopped, if the selected clock source stops
WDTCLK	unchanged/stopped	Stopped, if the selected clock source stops or if WCC.WDTSEL1 = 0
WCTCLK	unchanged/stopped	Depends on clock selector PSM.CMODE

The WATCH mode can be released by

- the unmasked maskable interrupts INTPn, INTCnWUP INTWTnUV, INTTM01, INTVCn, INTCBnR
- NMI0, NMIWDT
- $\overline{\text{RESET}}$, RESPOC, RESWDT, RESCMM, RESCMS

On WATCH mode release, the CPU starts operation using the following clocks:

- if PSM.OSCDIS = 1: sub clock source selected before WATCH mode was entered, that means, either internal oscillator or sub oscillator (defined by PCC.SOSCP)
- if PSM.OSCDIS = 0: main oscillator

If the internal oscillator was stopped before entering the WATCH mode, the oscillation stabilization time for the internal oscillator is ensured by hardware after WATCH mode release.

PLL and SSCG remain stopped after WATCH release.

Peripheral clock supply is switched to main oscillator supply, if PSM.OSCDIS = 0, otherwise the internal oscillator is used for peripheral clocks.

(4) Sub-WATCH mode

In Sub-WATCH mode, the clock supply for the CPU and the majority of peripherals is stopped. Main oscillator, PLL, and SSCG are stopped. By default, internal oscillator and sub oscillator operation is not influenced. For exceptions see *“Internal and sub oscillator operation“ on page 160*.

Depending on register settings, the Watchdog Timer clock supply can continue operation or stop.

Table 4-28 Clock Generator status in Sub-WATCH mode

Item	Status	Remarks
Main oscillator	stopped	
Sub oscillator	operates	
Internal oscillator	operates/stopped	Stopped if WCC.ROSTP = 1
SSCG	stopped	
PLL	stopped	
VBCLK (CPU system)	stopped	
IICLK	stopped	
PCLK0, PCLK1	stopped	
PCLK2…PCLK15	stopped	
SPCLK0, SPCLK1	stopped	
SPCLK2…SPCLK15	stopped	
FOUTCLK	unchanged	Stopped, if the selected clock source stops
WTCLK / LCDCLK	unchanged	Stopped, if the selected clock source stops
WDTCLK	unchanged/stopped	Stopped, if the selected clock source stops or if WCC.WDTSEL1 = 0
WCTCLK	stopped	

The Sub-WATCH mode can be released by

- the unmasked maskable interrupts INTPn, INTCnWUP INTWTnUV, INTVCn, INTCBnR
- NMI0, NMIWDT
- $\overline{\text{RESET}}$, RESPOC, RESWDT, RESCMM, RESCMS

On Sub-WATCH mode release, the CPU starts operation using the following clocks:

- if PSM.OSCDIS = 1: sub clock source selected before Sub-WATCH mode was entered, that means, either internal oscillator or sub oscillator (defined by PCC.SOSCP)
- if PSM.OSCDIS = 0: main oscillator

If the internal oscillator was stopped before entering the Sub-WATCH mode, the oscillation stabilization time for the internal oscillator is ensured by hardware after Sub-WATCH release.

PLL and SSCG remain stopped after Sub-WATCH release.

Peripheral clock supply is switched to main oscillator supply, if PSM.OSCDIS = 0, otherwise the internal oscillator is used for peripheral clocks.

(5) STOP mode

In STOP mode, all clock sources are stopped, except sub and internal oscillator. These can be configured in register WCC to stop as well. No clock is available, and no internal self-timed processes operates.

Table 4-29 Clock Generator status in STOP mode

Item	Status	Remark
Main oscillator	stopped	
Sub oscillator	operates/stopped	Stopped if WCC.SOSTP = 1
Internal oscillator	operates/stopped	Stopped if WCC.ROSTP = 1
SSCG	stopped	
PLL	stopped	
VBCLK (CPU system)	stopped	
IICLK	stopped	
PCLK0, PCLK1	stopped	
PCLK2…PCLK15	stopped	
SPCLK0, SPCLK1	stopped	
SPCLK2…SPCLK15	stopped	
FOUTCLK	stopped	
WTCLK / LCDCLK	unchanged/stopped	Stopped, if the selected clock source stops
WDTCLK	unchanged/stopped	Stopped, if the selected clock source stops or if WCC.WDTSEL1 = 0
WCTCLK	stopped	

The STOP mode can be released by

- the unmasked maskable interrupts INTPn, INTCnWUP INTVCn, INTCBnR, INTWT0UV, INTWT1UV
- NMI0, NMIWDT
- $\overline{\text{RESET}}$, RESPOC, RESWDT, RESCMM, RESCMS

On STOP mode release, the CPU clock and peripheral clocks are supplied by the main oscillator.

(6) Clock status summary

Table 4-30 on page 149 summarizes the status of all clocks delivered by the Clock Generator in the different states.

“Normal” describes all status except reset and power save modes.

The HALT mode is not listed in the table. It does not change any of the table items, but stops only the CPU core operation.

Below the table you find the explanation of the terms used in the table.

Table 4-30 Status of oscillators and Clock Generator output clocks (1/2)

Macro	Clock signal	Condition	Reset	Reset release	Normal	IDLE	IDLE release	STOP	STOP release	WATCH	WATCH release	Sub-WATCH	Sub-WATCH
Oscillators													
Main-osc	–	OSCDIS=0	n.a.		on	on		stop	on	on	on	stop	on
		OSCDIS=1	stop		stop	n.a.				stop	n.a.		stop
Sub-osc	–	SOSTP=1	n.a.		on	on		stop	on	on		on	
		SOSTP=0	on					on					
Internal-osc	–	ROSTP=1	n.a.		on	on		stop	on	stop	on	stop	on
		ROSTP=0	on					on		on		on	
SSCG/PLL													
SSCG	–	–	stby		scen	scen		stby		stby		stby	
PLL	–	–			pllen	pllen							
Clock Generator output clocks													
CPU system clock	VBCLK	CLS/CKS = 000_B	n.a.		MOCLK	off	MOCLK	off	MOCLK	off	MOCLK	off	MOCLK
		CLS/CKS = 001_B			SSCCLK		n.a.		n.a.		n.a.		n.a.
		CLS/CKS = $01x_B$			PLLCLK		n.a.		n.a.		n.a.		n.a.
		CLS/CKS = $1xx_B$	off	ROCLK	SBCLK		n.a.		n.a.		n.a.		SBCLK
Peripheral clocks	PCLK0 PCLK1	PERIC=0	off	MOCLK[a]	MOCLK	off	MOCLK	off	MOCLK	off	MOCLK	off	MOCLK
		PERIC=1	n.a.		PLLCLK		PLLCLK		n.a.		n.a.		n.a.
	PCLK2 - PCLK15	–	off	MOCLK[a]	MOCLK	off	MOCLK	off	MOCLK	off	MOCLK	off	MOCLK
	SPCLK0 SPCLK1	SPSEL[1:0]=00_B	off	MOCLK[a]	MOCLK	off	MOCLK	off	MOCLK	off	MOCLK	off	MOCLK
		SPSEL[1:0]=01_B	n.a.		PLLCLK		PLLCLK		n.a.		n.a.		n.a.
		SPSEL[1:0]=11_B			SSCCLK		SSCCLK		n.a.		n.a.		n.a.
	SPCLK2 - SPCLK15	SPSEL1=0	off	MOCLK[a]	MOCLK	off	MOCLK	off	MOCLK	off	MOCLK	off	MOCLK
		SPSEL1=1	n.a.		PLLCLK		PLLCLK		n.a.		n.a.		n.a.

Table 4-30 Status of oscillators and Clock Generator output clocks (2/2)

Macro	Clock signal	Condition		Reset	Reset release	Normal	IDLE	IDLE release	STOP	STOP release	WATCH	WATCH release	Sub-WATCH	Sub-WATCH
Watch Calibration Timer	WCTCLK	CMODE=0		off	MOCLK[a]	PCLK1	off	PCLK1	off	PCLK1	off	PCLK1	off	PCLK1
		CMODE=1		n.a.		MOCLK	MOCLK	MOCLK		MOCLK	MOCLK	MOCLK		MOCLK
Watchdog Timer	WDTCLK	SOSC=0 WDTSEL0=0	WDTSEL1=0	off	ROSCK	ROSCK	off	ROSCK	off	ROSCK	off	ROSCK	off	ROSCK
			WDTSEL1=1	n.a.			ROSCK		ROSCK (off[b])		ROSCK (off[b])		ROSCK (off[b])	
		SOSC=1 WDTSEL0=0	WDTSEL1=0	n.a.		SOCLK	off	SOCLK	off	SOCLK	off	SOCLK	off	SOCLK
			WDTSEL1=1				SOCLK		SOCLK (off[b])		SOCLK		SOCLK	
		SOSC=x WDTSEL0=1	WDTSEL1=0	n.a.		MOCLK	off	MOCLK	off	MOCLK	off	MOCLK	off	MOCLK
			WDTSEL1=1				MOCLK				MOCLK			
I^{2}C	IICLK	IICSEL[1:0]=00_B		off	MOCLK[a]	MOCLK	off	MOCLK	off	MOCLK	off	MOCLK	off	MOCLK
		IICSEL[1:0]=01_B		n.a.		SSCSCK		SSCSCK		n.a.		n.a.		n.a.
		IICSEL[1:0]=$1x_B$				PLLSCK		PLLSCK		n.a.		n.a.		n.a.
Watch Timer LCD-C/D	WTCLK LCDCLK	WTSOS/WTSEL0=00_B		ROSCK		ROSCK	ROSCK		ROSCK (off[b])	ROSCK	ROSCK (off[b])	ROSCK	ROSCK (off[b])	ROSCK
		WTSOS/WTSEL0=10_B		n.a.		SOCLK	SOCLK		SOCLK (off[b])	SOCLK	SOCLK	SOCLK	SOCLK	SOCLK
		WTSOS/WTSEL0=$x1_B$				MOCLK	MOCLK		off	MOCLK	MOCLK (off[c])	MOCLK	MOCLK (off[c])	MOCLK
Port	FOUTCLK	FOSOS/FOCKS1/0=$x00_B$		off		MOCLK	MOCLK		MOCLKLCD-C/D		MOCLK		MOCLK	
		FOSOS/FOCKS1/0=$x01_B$		n.a.		SSCCLK	SSCCLK		SSCCLK		SSCCLK		SSCCLK	
		FOSOS/FOCKS1/0=$x10_B$				PLLCLK	PLLCLK		PLLCLK		PLLCLK		PLLCLK	
		FOSOS/FOCKS1/0=011_B				ROCLK	ROCLK		ROCLK		ROCLK		ROCLK	
		FOSOS/FOCKS1/0=111_B				SOCLK	SOCLK		SOCLK		SOCLK		SOCLK	

a) After reset release these clocks are supplied with the internal ROCLK. When the main oscillator is stable, these clocks are automatically changed to MOCLK.

b) ROCLK (SOCLK) remains clock source, but internal oscillator (sub oscillator) may be stopped in the respective power save mode by WCC.ROSTP = 1 (WCC.SOSTP = 1).

c) MOSCLK remains clock source, but main oscillator may be stopped in the respective power save mode by PSM.OSCDIS = 1.

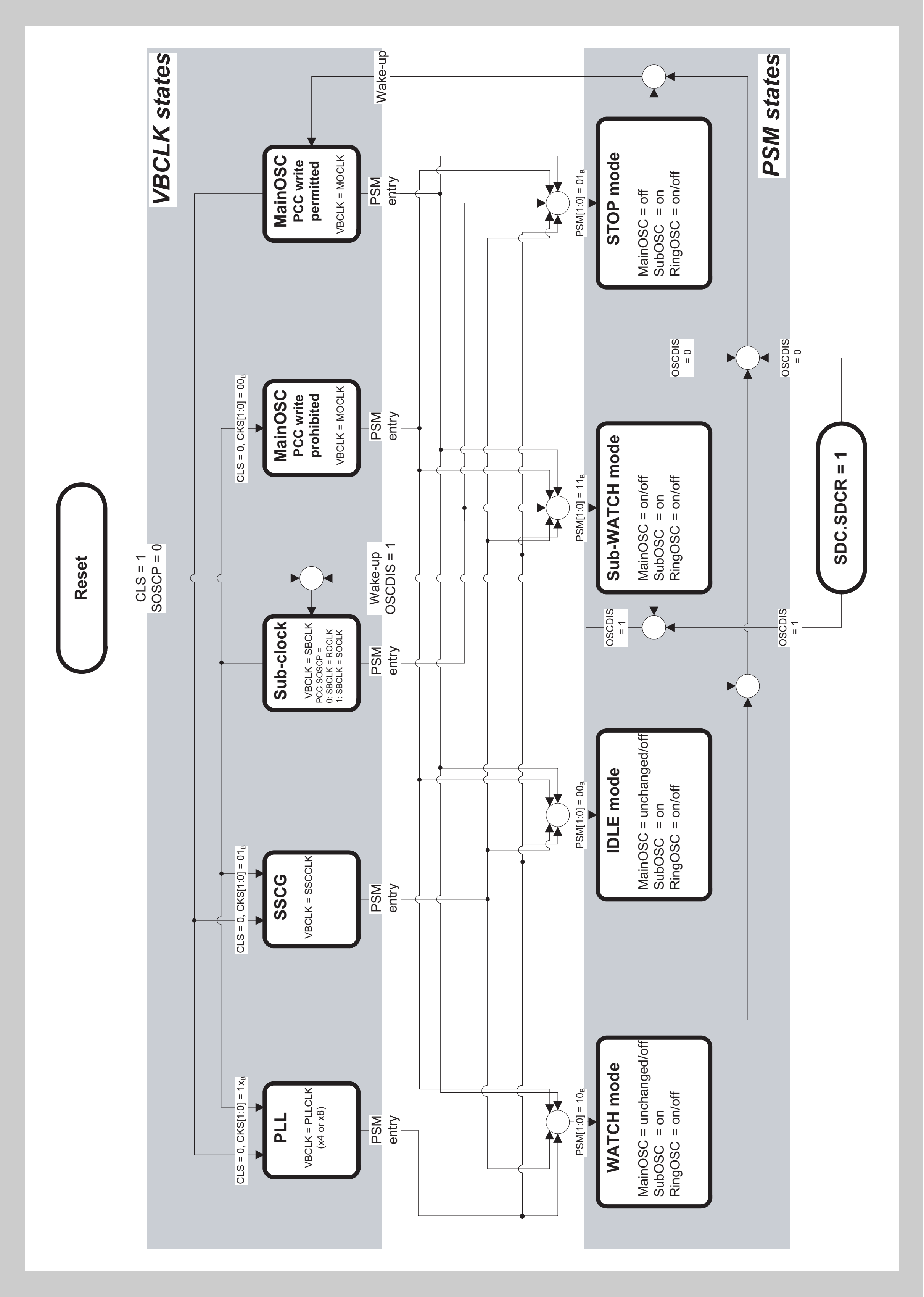
In the table following terms are used:

stop:	Oscillator stopped
on:	Oscillator operating
stby:	PLL/SSCG in standby, no clock output
pllen/scen:	PLL/SSCG generates clock output
off:	Clock inactive
n.a.	not applicable (control bits are determined by hardware)

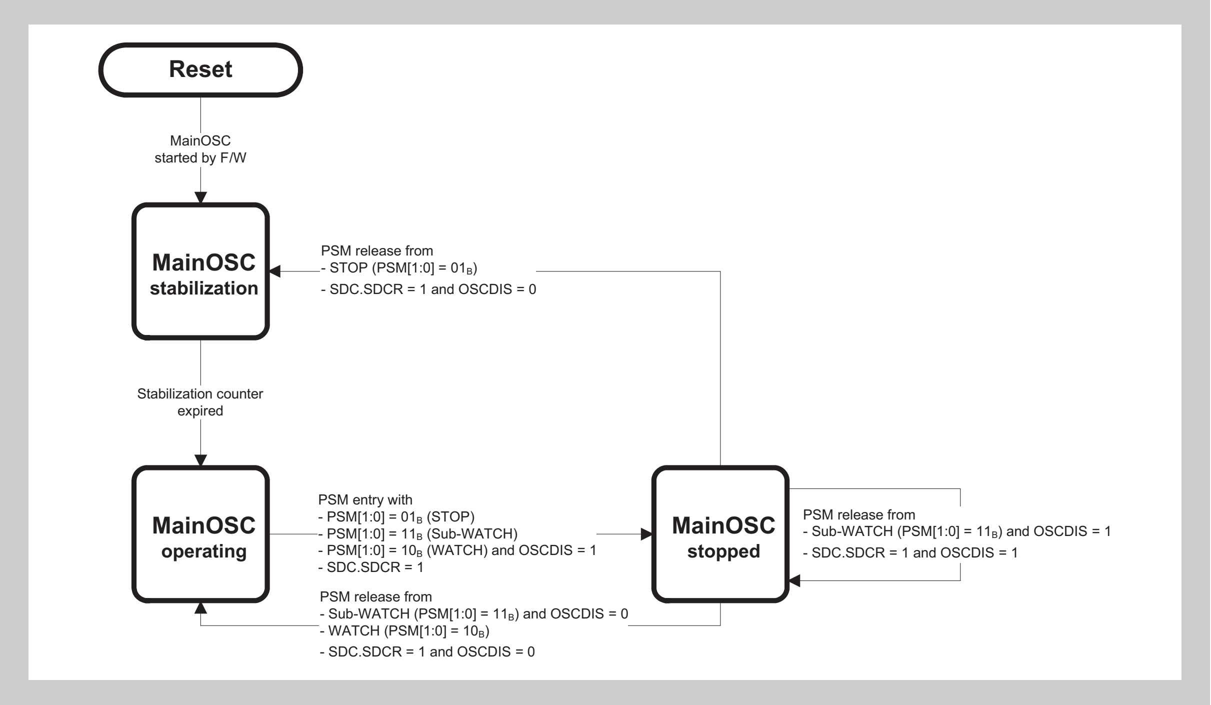
MOCLK:	Main oscillator clock
ROCLK:	Internal oscillator clock
SOCLK:	Sub oscillator clock
SBCLK:	Sub clock – PCC.SOSCP = 0: ROCLK – PCC.SOSCP = 1: SOCLK
PLLCLK:	PLL output clock
SSCGCLK:	SSCG output clock

4.3.2 Clock Generator state transitions

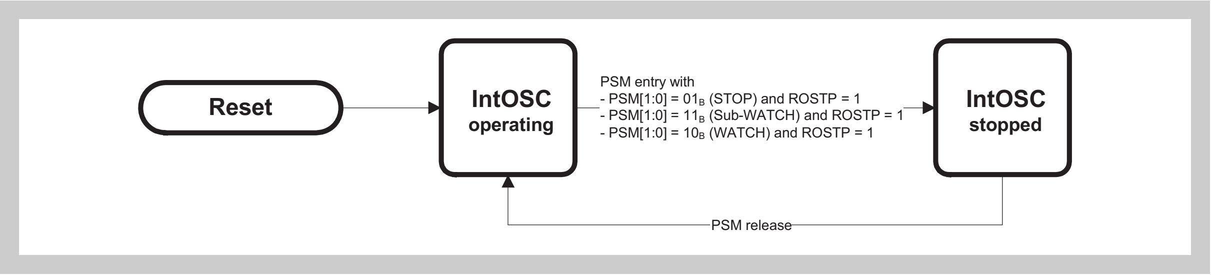
(1) VBCLK state transitions



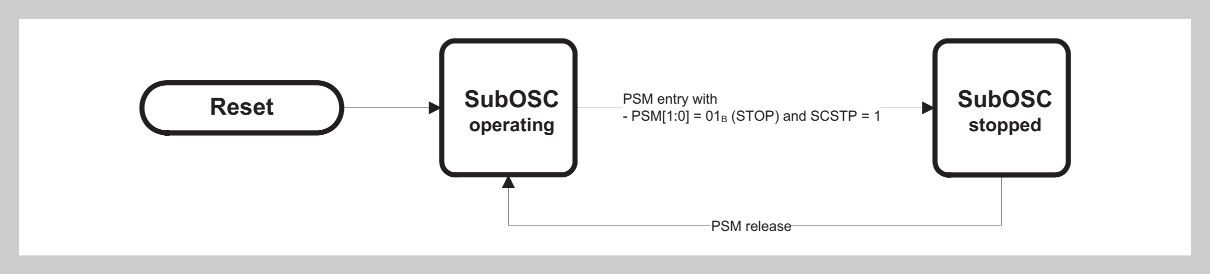
(2) Main oscillator state transitions


Reset
MainOSC started by F/W
MainOSC stabilization
PSM release from
- STOP (PSM[1:0] = 01_B)
- SDC.SDCR = 1 and OSCDIS = 0
Stabilization counter expired
MainOSC operating
PSM entry with
- PSM[1:0] = 01_B (STOP)
- PSM[1:0] = 11_B (Sub-WATCH)
- PSM[1:0] = 10_B (WATCH) and OSCDIS = 1
- SDC.SDCR = 1
MainOSC stopped
PSM release from
- Sub-WATCH (PSM[1:0] = 11_B) and OSCDIS = 1
- SDC.SDCR = 1 and OSCDIS = 1
PSM release from
- Sub-WATCH (PSM[1:0] = 11_B) and OSCDIS = 0
- WATCH (PSM[1:0] = 10_B)
- SDC.SDCR = 1 and OSCDIS = 0

(3) Internal oscillator states


Reset
IntOSC operating
PSM entry with
- PSM[1:0] = 01_B (STOP) and ROSTP = 1
- PSM[1:0] = 11_B (Sub-WATCH) and ROSTP = 1
- PSM[1:0] = 10_B (WATCH) and ROSTP = 1
IntOSC stopped
PSM release

(4) Sub oscillator states



4.3.3 Power save mode activation

In the following procedures for securely entering a power save mode are described.

Stepper-C/D shut down

In order to minimize power consumption during power save modes the Stepper Motor Controller/Driver needs to be shut down in a special sequence. Refer to *"MCNTCn0, MCNTCn1 - Timer mode control registers" on page 714*.

(1) HALT mode

For entering the HALT mode proceed as follows:

1. Mask all interrupts which shall not have wake-up capability by xxIC.xxMK = 1 and discard all possibly pending interrupts by xxIC.xxIF = 0.
2. Unmask all interrupts which shall have wake-up capability by xxIC.xxMK = 0.
3. Execute the "halt" instruction.
4. Insert at least five "nop" instruction after the "halt" instruction.

(2) WATCH, Sub-WATCH, STOP and IDLE mode

For entering these power save mode proceed as follows:

1. In case maskable interrupts shall be used for wake-up unmask these interrupts by IMRm.xxMK = 0 (refer to *"IMR0 to IMR5 - Interrupt mask registers" on page 200*).
2. Mask all other interrupts, i.e.
   - none wake-up capable interrupts
   - wake-up capable interrupts which shall not be used for wake-up

   by IMRm.xxMK = 1. This prevents the power save mode entry procedure from being interrupted by these interrupts.
3. It is recommended to disable interrupt acknowledgement by the "di" instruction.
4. Specify the desired power save mode in PSM.PSM[1:0].
5. Enable writing to the write-protected register PSC by writing to PRCMD.
6. Write to PSC for specifying permitted wake-up events and activate the power save mode by setting PSC.STP to 1.

Example The following example shows how to initialize and enter a WATCH, Sub-WATCH, STOP or IDLE power save mode.

First the desired power save mode is specified (WATCH mode in this example, that means PSM.PSM[1:0] = 10_B).

The PSC register is a write-protected register, and the PRCMD register is the corresponding write-enable register. PRCMD has to be written immediately before writing to PSC.

In this example, maskable interrupts are permitted to leave the power save mode.

```
             // xxIC.xxMK = 0          // mask all none wake-up interrupts
             // xxIC.xxMK = 1          // unmask all wake-up interrupts
      di
      mov    0x02,r10
      st.b   r10,PSM[r0]               // PSM.PSM[1:0] = 10B: WATCH mode
      mov    0x62,r10
      st.b   r10,PRCMD[r0]             // enable write to PSC
             r10,PSC[r0]               // wake up by maskable interrupts
      st.b                             // and enter power save mode
      nop
      nop
      nop
      nop
      nop
                                       // after wake-up
              // xxIC.xxIF = 0         // discard all unwanted pending interrupts
      ei
```

Be aware of the following notes when entering power save mode using the above sequence:

Note

1. It is recommended to disable maskable interrupt acknowledgement in general by the “di” instruction (step 3.) to prevent any pending interrupt from being served during the power save mode set-up procedure. This makes it also possible to completely control the process after wake-up, since no pending interrupt will be unintentional acknowledged. Before enabling interrupt acknowledgement by the “ei” instruction (step 16.) after wake-up, all unwanted interrupts can be discarded by setting xxIC.xxIF = 0 (step 15.).
   Since the wake-up capability of the unmasked wake-up interrupts is not affected by “di”, such interrupts shall be masked (step 1.) by IMRm.xxMK = 1.
2. The store instruction to PRCMD will not allow to acknowledge any interrupt until processing of the subsequent instruction is complete. That means, an interrupt will not be acknowledged before the store to PSC. This presupposes that both store instructions are performed consecutively, as shown in the above example.
   If another instruction is placed between steps 7 and 8, an interrupt request may be acknowledged in between, and the power save mode may not be entered.
   However if the “di” instruction was executed before (step 3.) none interrupt will be acknowledged anyway.

3. At least 5 “nop” instructions must follow the power down mode setting, that means after the write to PSC. The microcontroller requires this time to enter power down mode.

4. The data written to the PRCMD register must be the same data that shall be written to the write-protected register afterwards.
   The above example ensures this method, since the contents of r10 is first written to PRCMD and then immediately to PSC.

5. Make sure that all DMA channels are disabled. Otherwise a DMA could happen between steps 7 and 8, and the power down mode may not be entered at all.
   Further on do not perform write operations to PRCMD and write-protected registers by DMA transfers.

6. No special sequence is required for reading the PSC register.

Caution If a wake-up event occurs within the 5 “nop” instructions after a power save mode request (PSC.STP = 1) the microcontroller is immediately returning from power save mode, but may have not at all or only partly entered the power save mode. Following three situations can occur:

1. power save mode request not accepted
   wake-up configuration not established, PLL/SSCG are operating
2. power save mode request accepted, but not completed
   wake-up configuration established, but PLL/SSCG operating
3. power save mode request accepted and completed
   wake-up configuration established, PLL/SSCG stopped

4.3.4 CPU operation after power save mode release

Clock Generator re-configuration

The clock for the CPU system can be switched only once after reset, power save mode release, or the default clock setup request (SDC.SDCR = 1).

The clocks for the Watchdog Timer, Watch Timer, and LCD Controller/Driver can be switched only once after system reset.

Access to peripherals that have no clock supply in Sub-WATCH mode may cause system deadlock. This can happen if the main oscillator remains disabled.

Wake-up configuration

Wake-up configuration established means that all registers and clock paths are set to their wake-up state.

The software should check after wake-up whether the expected wake-up configuration has been completely established. This can be achieved by observing

- following clock generator registers, which are modified by power save mode entry and wake-up
  - after WATCH, Sub-WATCH, STOP wake-up following bits are cleared:
    CKC.PLLEN, CKC.SCEN, CKC.PERIC, SCC.SEL, ICC.SEL
  - after IDLE or STOP wake-up following bits are cleared:
    PCC.CLS, PCC.CKS
  - after Sub-WATCH or WATCH wake-up
    PCC.CLS/PCC.CKS = 000_B, if PSM.OSCDIS = 0
    PCC.CLS/PCC.CKS = 100_B, if PSM.OSCDIS = 1
- the “completed power save mode” bit CGSTAT.CMPLPSM
  - CGSTAT.CMPLPSM = 0 if a power save mode request has been accepted but not completed, wake-up configuration established, but PLL/SSCG operating (provided that CGSTAT.CMPLPSM = 1 before power save mode request)
  - CGSTAT.CMPLPSM = 1 if a power save mode has been completely entered, wake-up configuration established, PLL/SSCG stopped (provided that a power save mode request has been accepted before, i.e. CGSTAT.CMPLPSM = 1 $\rightarrow$ 0)

Note that CGSTAT.CMPLPSM is set to 0 if a power save mode request is accepted. If it was 0 before it does not change it’s state.

Table 4-31 summarizes the different configurations.

Table 4-31 **Power save mode wake-up configurations**

CGSTAT.CMPLPSM		Registers and clock paths[a]	Configuration after wake-up
before PSM-RQ[b]	after wake-up		
0	0	not changed	PSM-RQ not accepted
		changed	PSM-RQ accepted configuration done, but PLL/ SSCG operating
	1	not changed	not possible
		changed	PSM-RQ accepted configuration done, PLL/SSCG off
1	0	not changed	not possible
		changed	PSM-RQ accepted configuration done, but PLL/ SSCG operating
	1	not changed	PSM-RQ not accepted
		changed	PSM-RQ accepted configuration done, PLL/SSCG off

a) A change of a register's contents can only be taken as an indicator if it is before power save mode request different to the wake-up configuration.

b) PSM-RQ: power save mode request (PSC.STP = 1)

If the power save mode request was accepted the entire clock generator can be reconfigured after wake-up. Afterwards set CKC.PLLEN = 1 and CKC.SCEN = 1 and wait the stabilization times before using the PLL and SSCG as clock sources.

Set default clock

If the power save mode wake-up configuration is entered by setting SDC.SDCR = 1 all registers and clock paths settings are performed, but the PLL and SSCG are still operating, that means CGSTAT.PSM remains unchanged.

The entire clock generator can be reconfigured, i.e. all registers can be written.

If the SSCG configuration shall not be changed, set CKC.PLLEN = 1 and CKC.SCEN = 1. The SSCG respectively PLL can be used immediately as clock sources without waiting the stabilization times.

If the SSCG configuration shall be changed, rewrite the SSCG configuration registers, set CKC.PLLEN = 1 and CKC.SCEN = 1. In this case make sure the stabilization times has elapsed before using the PLL or SSCG as clock sources.

After IDLE and STOP

On return from IDLE or STOP mode, the bits PCC.CLS, PCC.CKS1, and PCC.CKS2 are cleared. After IDLE mode, the main oscillator is still running; on return from STOP mode, it is automatically started.

As a result, the main oscillator is chosen and enabled as the source for the CPU system clock VBCLK.

After WATCH

In WATCH mode the main oscillator operation depends on PSM.OSCDIS:

- If PSM.OSCDIS was 0 before entering WATCH mode the main oscillator remains active. After WATCH mode release the main oscillator is chosen as the CPU system clock.
- If PSM.OSCDIS was 1 before entering WATCH mode the main oscillator is stopped during WATCH mode. After WATCH mode release the main oscillator is automatically started, the oscillator stabilization time is waited and the main oscillator is chosen as the CPU system clock.

After Sub-WATCH

In Sub-WATCH mode the main oscillator is stopped. On return from Sub-WATCH, PCC.CLS is set to the status of PSM.OSCDIS.

- If PSM.OSCDIS was 0 before entering Sub-WATCH, the main oscillator is started and chosen as the source for the CPU system clock (PCC.CLS = 0, PCC.CKS[1:0] = 00_B).
- If PSM.OSCDIS was 1 before entering Sub-WATCH, the main oscillator remains stopped, and the CPU is clocked by a sub clock (PCC.CLS = 1, PCC.CKS[1:0] = xx_B).

“Sub clock” means the clocks supplied by either the 32 KHz sub oscillator or the 200 kHz internal oscillator. The selection must be made in the PCC register *before* entering the Sub-WATCH or WATCH mode:

- PCC.SOSCP = 0: Internal oscillator
- PCC.SOSCP = 1: Sub oscillator

Software can switch from sub clock CPU operation to normal run mode (by enabling the main oscillator by PSM.OSCDIS = 0) or re-enter Sub-WATCH respectively WATCH mode.

After HALT

On return from HALT mode the CPU resumes operation with the same clock settings as before HALT was entered.

4.4 Clock Generator Operation

4.4.1 Internal and sub oscillator operation

By default, sub and internal oscillator operate during all power save modes.

However, it can be specified in the WCC register that the sub oscillator stops in STOP mode (WCC.SOSTP).

It can also be specified that the internal oscillator stops in WATCH, Sub-WATCH, and STOP mode (WCC.ROSTP).

These bits can be written once after system reset, independent of the reset source.

4.4.2 Watch Timer and Watch Calibration Timer clocks

The Watch Timer input clock WTCLK can be derived directly from the main, sub, or internal oscillator. Therefore, the WT can be operating in all power save modes.

Because PCLK1 is stopped during power save modes, the Watch Calibration Timer input clock WCTCLK can be directly connected to the main oscillator output.

Note WCTCLK is not available in Sub-WATCH and STOP mode where the main oscillator is stopped. These modes must be released before the WCT can operate.

4.4.3 Clock output FOUTCLK

The Clock Generator output signal FOUTCLK supplies a clock for external components. It can be derived from any internal clock source, that means internal oscillator, sub oscillator, main oscillator, PLL, or SSCG. A dedicated frequency divider is available to scale the output clock down.

FOUTCLK must be enabled by register setting (FCC.FOEN = 1). It is not influenced by the power save modes. But FOUTCLK stops, if the selected clock source stops.

After reset release, FOUTCLK is disabled (register FCC is cleared), and the pin FOUT put in input mode.

Note

1. If you change the configuration of FOUTCLK or enable/disable the selected clock source while FOUTCLK is active, glitches or irregular clock periods may appear at the output pin.
2. The clock signal FOUTCLK cannot be used to synchronize external circuitry to other output signals of the microcontroller—it has no specified phase relation to other output signals.
3. There is an upper frequency limit for the output buffer of the FOUTCLK function. Do not select a frequency higher than the maximum output buffer frequency. Please refer to the Data Sheet for the frequency limit.

4.4.4 Default clock generator setup

The Clock Generator can be reset to the clock settings that are used by default after power save mode release. This is done by setting bit SDC.SDCR.

For this kind of reset, it is not necessary to enter a power save mode, and no wake-up signal is required.

If reset to defaults is requested, CKC.PLLEN and CKC.SCEN are cleared. However the PLL and SSCG remain active. The CPU clock source is switched to sub, internal, or main oscillator (depending on the bits PSM.OSCDIS and PCC.SOSCP). Peripheral clock sources are switched to main oscillator. For details see *“SDC - Set default clock register“ on page 115*.

This feature reduces the total power consumption of peripherals and CPU.

It provides also a way to stop the PLL and SSCG. These PLLs must be stopped if the clock sources for the CPU or peripherals shall be changed.

Note While the clock sources are switched, the peripheral clocks are suspended. Therefore, the timing of peripheral modules may be inaccurate until the reset has finished.

4.4.5 Operation of the Clock Monitors

The microcontroller provides two separate clock monitors to watch the activity of the main oscillator and the sub oscillator.

(1) Description

The functional block diagram is shown below.

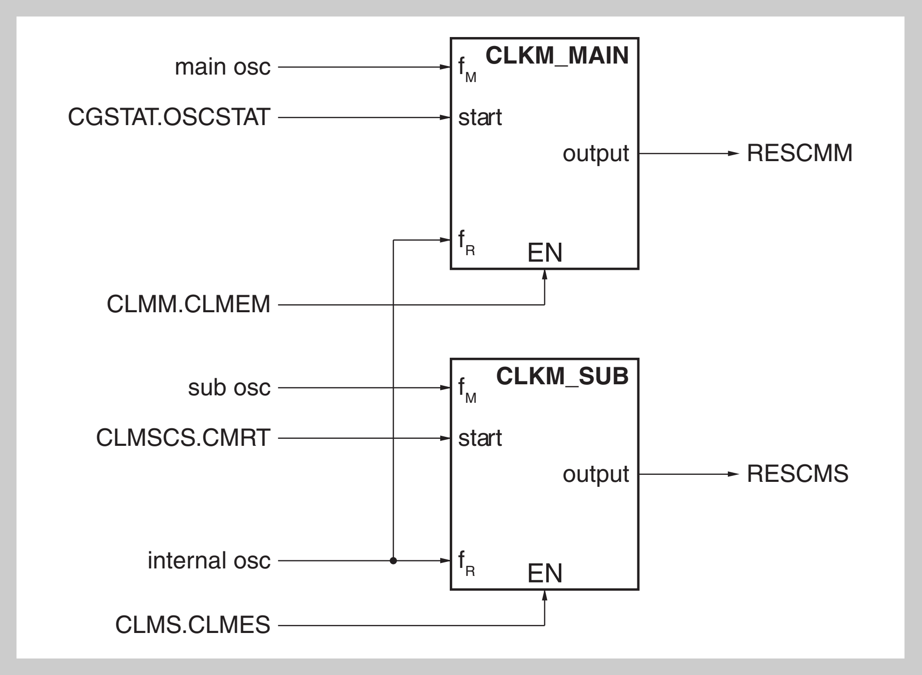


Figure 4-2 Clock Monitors block diagram

The clock monitors use the internal oscillator (f_R) for monitoring the main and sub oscillators (f_M).

If the main oscillator clock monitor detects a malfunction of the main oscillator (no pulse), it generates the reset request RESCMM. If the sub oscillator clock monitor detects a malfunction of the sub oscillator, it generates the reset request RESCMS.

(2) Start and stop

Before the clock monitors can be started, they have to be enabled by setting CLMM.CLMEM and CLMS.CLMES to 1.

Main oscillator monitor start

After enabling CLMM.CLMEM = 1 the main oscillator monitor is automatically started as soon as the main oscillator is stable, indicated by CGSTAT.OSCSTAT = 1.

Sub oscillator monitor start

After enabling CLMM.CLMES = 1 the sub oscillator monitor must be started by software by setting CLMCS.CMRT to 1.

After starting the sub oscillator clock monitor by CLMCS.CMRT = 1 clear CLMCS.CMRT by software.

Since CLMCS.CMRT = 1 is synchronized with the internal oscillator any change of this bit has to be maintained for at least 65 internal oscillator periods T_{ROSC} = 1/f_{ROSC} to become effective. Therefore a wait period has to be assured before this bit is changed again.

Proceed as follows to start the sub oscillator clock monitor:

1. After reset enable sub oscillator clock monitor:
   PRCMDCM = FF_Hpermit write to CLMS
   CLMS.CLMES = 1enable sub oscillator clock monitor
2. Make sure the sub oscillator is stable.
3. Start sub oscillator clock monitor after reset and after power save mode wake-up:
   CLMCS.CMRT = 1
4. Wait for 65 internal oscillator periods T_{ROSC} before resetting CMRT:
   wait (65 x max(T_{ROSC}))
5. Clear CLMCS.CMRT:
   CLMCS.CMRT = 0
6. Before CMRT should be set to 1 again, wait for 65 internal oscillator periods T_{ROSC}:
   wait (65 x max(T_{ROSC}))

Note that the minimum internal oscillator frequency min(f_{ROSC}) (max(T_{ROSC})) has to be taken into account for the wait time in steps (3) and (5).

Caution The sub oscillator clock monitor is sometimes already started by setting CLMS.CLMES = 1, i.e. without CLMCS.CMRT = 1.
In these cases it would not be required to start the sub oscillator by setting CLMCS.CMRT = 1 additionally.

Since it is unpredictable whether the clock monitor has already started after CLMS.CLMES = 1 the procedure described above should be followed in any case.

(3) Operation during and after power save modes

Main oscillator stopped If the main oscillator is stopped, its clock monitor changes to stand-by. When the main oscillator is restarted after power save mode release, the main oscillator clock monitor restarts automatically.

Sub oscillator stopped If the sub oscillator is stopped, its clock monitor stops.

When the sub oscillator is restarted after power save mode release, the sub oscillator clock monitor does not start automatically.

Software must ensure that the sub oscillator stabilization time has elapsed and then start the monitor by setting CLMCS.CMRT to 1.

Internal oscillator stopped If the internal oscillator is stopped, both clock monitors’ operation is suspended. Their operation is automatically resumed as soon as the internal oscillator is restarted.

Chapter 5 Flash Memory

The µPD70F3416 and µPD70F3417 microcontrollers are equipped with internal flash memory. The flash memory is attached to the V850 Fetch Bus VFB interface of the V850E CPU core. It is used for program code and storage of constant data.

When fetching an instruction, 4 bytes of the VFB flash memory can be accessed in 1 clock, and 4 bytes of the VSB flash memory can be accessed in 2 clocks.

The flash memory can be written mounted on the target board (on-board write), by connecting a dedicated flash programmer to the target system.

Flash memory is commonly used in the following development environments and applications:

- For altering software after solder-mounting of the microcontroller on the target system.
- For differentiating software in small-scale production of various models.
- For data adjustment when starting mass production.

5.1 Overview

Features summary

- Internal VFB flash memory:
  - μPD70F3417: 256 KB
  - μPD70F3416: 128 KB
- Operation speed up to 24 MHz by 2-way interleaved access:
  - 4-byte/1 CPU clock cycle access for consecutive instruction fetches
  - 4-byte/3 CPU clock cycles access for random instruction and data fetches
- All-blocks batch erase or single block erase
- Erase/write with single power supply
- Communication with dedicated flash programmer via various serial interfaces
- On-board and off-board programming
- Flash memory programming by self-programming

5.1.1 Flash memory address assignment

The 256 KB flash memory of μPD70F3417 is made up of 64 blocks. *Figure 5-1* shows the address assignment of the flash memory blocks.

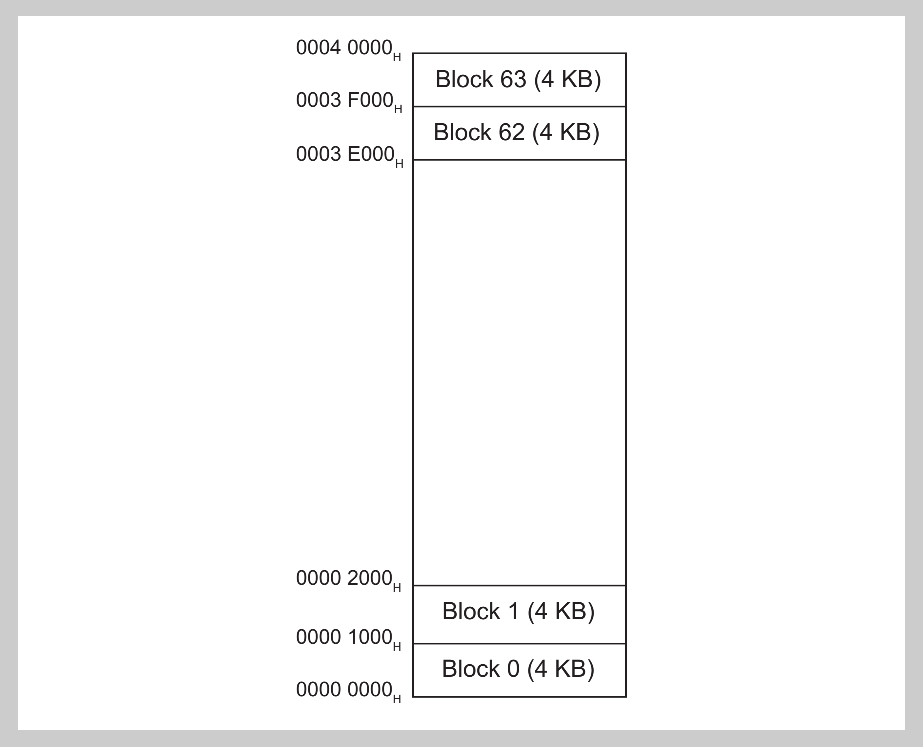


Figure 5-1 Address assignment of μPD70F3417 flash memory blocks

The 128 KB flash memory of µPD70F3416 is made up of 32 blocks. *Figure 5-2* shows the address assignment of the flash memory blocks.

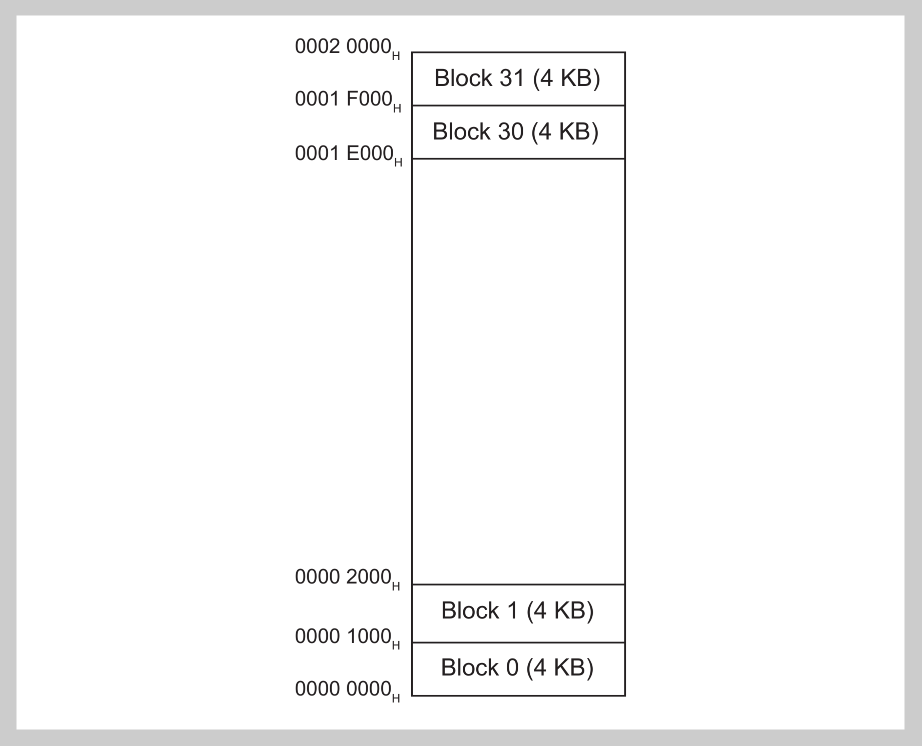


Figure 5-2 Address assignment of µPD70F3416 flash memory blocks

5.1.2 Flash memory erasure and rewrite

The following functions can be carried out by use of the flash memory self-programming library.

(1) Flash memory erasure

According to it's block structure the flash memory can be erased in two different modes.

- All-blocks batch erasure
  - µPD70F3417: 0000 0000_H to 0003 $FFFF_H$
  - µPD70F3416: 0000 0000_H to 0001 $FFFF_H$
- Block erasure
  Each 4 KB flash memory block can be erased separately.

(2) Flash memory rewrite

Once a complete block has been erased it can be rewritten in units of 8 byte. Each unit can be rewritten only once after erasure of the complete block.

5.1.3 Flash memory programming

The internal flash memory can be programmed in three different ways:

- Programming via self-programming
- Programming with external flash programmer

While the self-programming mode can be initiated from the normal operation mode the external flash programmer mode is entered immediately after release of a system reset. Refer to *“Operation Modes“ on page 83* for details on how to enter normal operation or external flash programming mode.

5.1.4 Boot block swapping

The microcontrollers with flash memory support secure boot block swapping. This will swap two 32 KB blocks at the bottom end (starting from address 0000 0000_H) of the Flash Memory. The block size is fixed and can not be changed.

For comprehensive information concerning secure boot block swapping refer to the application note “Self-Programming” (document no. U16929EE), which explains also the functions of the self-programming library. The latest version of this document can be loaded via the URL

http://www.renesas.eu/updates

5.2 Flash Self-Programming

The internal flash memory can be programmed via the secure self-programming facility. This feature enables the user's application to re-program the flash memory. The self-programming functions are part of the internal firmware, which resides in an extra internal ROM. The user's application can call the self-programming functions via the self-programming library, provided by Renesas Electronics.

Caution During self-programming make sure to disable all ROM correction facilities, as enabled ROM corrections may conflict with the internal firmware.

Start of self-programming The self-programming functions can be started out of the normal user mode of the microcontroller.

Self-programming must be in particular enabled in order to avoid unintended re-programming of the flash. Two ways to enable self-programming are provided:

- by setting the external FLMD0 pin to high level
  This requires some external components or wiring, e.g. connecting an output port to FLMD0.
- by setting an internal register bit
  This way does not need any special external components or wiring.

The following registers are used to enable self-programming internally by software.

5.2.1 Flash self-programming registers

For safety reasons flash self-programming needs to be explicitly enabled by use of two registers:

Table 5-1 Flash self-programming enable register overview

Register name	Shortcut	Address
Self-programming enable control register	SELFEN	FFFF FCA0$_H$
Self-programming enable protection register	SELFENP	FFFF FCA8$_H$

(1) SELFEN - Self-programming enable control register

The 8-bit SELFEN register enables the self-programming functions by software. It is an internal substitute to enabling self-programming by rising the FLMD0 pin to high level.

Access This registers can be read/written in 8-bit or 1-bit units.

Writing to this register is protected by a special sequence of instructions. Please refer to *“SELFENP - Self-programming enable protection register“ on page 169* for details.

Address FFFF FCA0$_H$

Initial Value 00$_H$. This register is cleared by any reset.

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	FLEN
R	R	R	R	R	R	R	R/W

Bit position	Bit name	Function
0	FLEN	Enable self-programming 0: Flash write/erase function is controlled by the FLMD0 pin 1: Flash write/erase function is enabled

(2) SELFENP - Self-programming enable protection register

The 8-bit SELFENP register protects the register SELFEN from inadvertent write access, so that the system does not stop in case of a program hang-up.

After data has been written to the SELFENP register, the first write access to register SELFEN is valid. All subsequent write accesses are ignored. Thus, the value of SELFEN can only be rewritten in a specified sequence, and illegal write access is inhibited.

Access This registers can be written in 8-bit units.

Address FFFF FCA8$_H$

Initial Value The contents of this register is undefined.

7	6	5	4	3	2	1	0
X	X	X	X	X	X	X	X
W	W	W	W	W	W	W	W

Caution In case a high level programming language is used, make sure that the compiler translates the two write instructions to SELFENP and SELFEN into two consecutive assembler “store” instructions.

Peripherals and pin functions All peripheral functions of the microcontroller continue operation during the self-programming process. Further the functions of all pins do not change.

5.2.2 Interrupt handling during flash self-programming

This microcontroller provides functions to maintain interrupt servicing during the self-programming procedure.

It is recommended to refer to the application note “Self-Programming” (document no. U16929EE) for comprehensive information concerning flash self-programming, which explains also the functions of the self-programming library. The latest version of this document can be loaded via the URL

http://www.renesas.eu/updates

Since neither the interrupt vector table nor the interrupt handler routines, which are normally located in the flash memory, are accessible during self-programming, interrupt acknowledges have to be re-routed to non-flash memory, i.e. to the internal RAM .

Therefore a prerequisite is necessary to enable interrupt servicing during self-programming:

- The concerned interrupt handler routine needs to be copied to the internal RAM.

Note 1. Note that this special interrupt handling adds some interrupt latency time.

2. Special interrupt handling is done only during the flash programming environment is activated. If self-programming is deactivated, the normal interrupt vector table in the flash memory is used.

All interrupt vectors are relocated to one entry point in the internal RAM:

- New entry point of *all* maskable interrupts is the 1st address of the internal RAM. A handler routine must check the interrupt source. The interrupt request source can be identified via the interrupt/exception source register ECR.EICC (refer to *“System register set“ on page 77*)
- New entry point of *all* non maskable interrupts is the word address following the maskable interrupt entry, i.e. the second address of the internal RAM. The interrupt request source can be identified via the interrupt/exception source register ECR.FECC (refer to*“System register set“ on page 77*).

In general a jump to a special handler routine will be placed at the 1st and 2nd internal RAM address, which identifies the interrupt sources and branches to the correct interrupt service routine.

The function serving the interrupt needs to be compiled as an interrupt function (i.e. terminate with a RETI instruction, save/restore all used registers, etc.).

5.3 Flash Programming with Flash Programmer

A dedicated flash programmer can be used for on-board or off-board writing of the flash memory.

(1) On-board programming

The contents of the flash memory can be rewritten with the microcontroller mounted on the target system. Mount a connector that connects the flash programmer on the target system.

A CSI or a UART interface can optionally be used for the communication between the external flash programmer and the V850 microcontroller.

All signals, including clock and power supply, can be provided by the external flash programmer. However, an on-board clock to the X1 input may be used instead of the clock, provided by the flash programmer.

(2) Off-board programming

The flash memory of the microcontroller can be written before the device is mounted on the target system, by using a dedicated program adapter (FA series).

All signals, including clock and power supply, are provided by the external flash programmer.

Note The FA series is a product of Naito Densei Machida Mfg. Co., Ltd.

5.3.1 Programming environment

The necessary environment to write a program to the flash memory of the microcontroller is shown below.

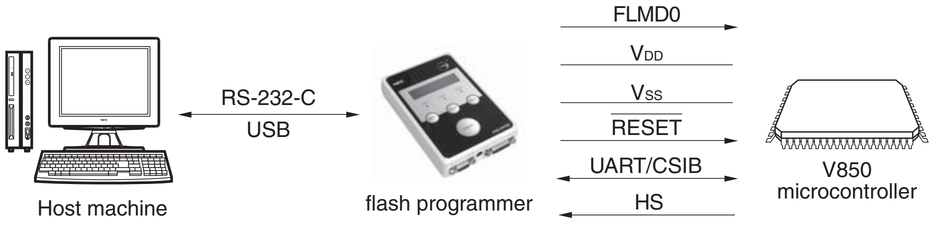


Figure 5-3 Environment to write program to flash memory

A host machine is required for controlling the flash programmer.

Following microcontroller serial interfaces can be used as the interface between the flash programmer and the microcontroller:

- asynchronous serial interface UART
- clocked serial interface CSIB

If the CSIB interface is used with handshake, the flash programmer's HS signal is connected to a certain V850 port. The port used as the handshake port is given in *Table 5-2*.

Flash memory programming off-board requires a dedicated program adapter.

UARTA0 or CSIB0 is used as the interface between the flash programmer and the microcontroller. Flash memory programming off-board requires a dedicated program adapter (FA series).

5.3.2 Communication mode

The communication between the flash programmer and the microcontroller utilizes the Asynchronous Serial Interface UARTA0 or the synchronous serial interface CSIB0.

For programming via the synchronous serial interface CSIB0 without handshake and with handshake modes are supported. In the latter mode the port pin P84 is used for the programmer's handshake signal HS.

(1) UARTA0

Transfer rate: 4.800 to 153.600 bps

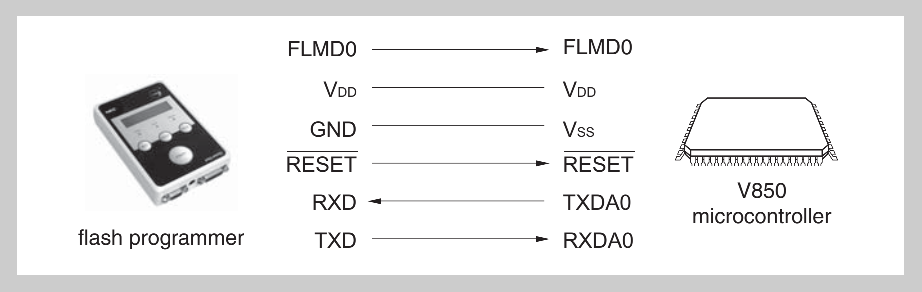


Figure 5-4 Communication with flash programmer via UARTA0

(2) CSIB0 without handshake

Serial clock: up to 2.5 MHz (MSB first)

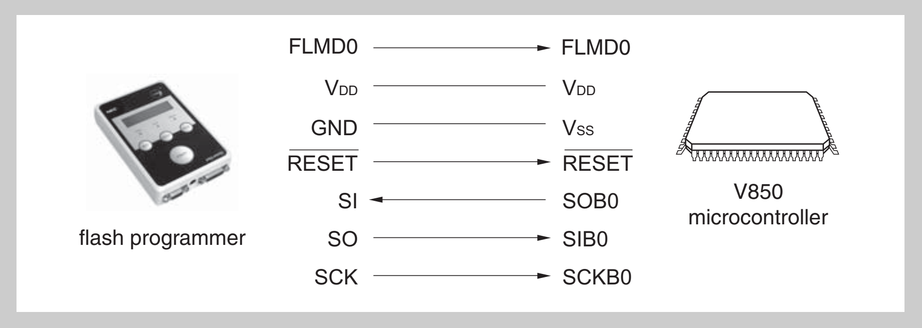


Figure 5-5 Communication with flash programmer via CSIB0 without handshake

(3) CSIB0 with handshake (CSIB0 + HS)

Serial clock: Up to 2.5 MHz (MSB first)

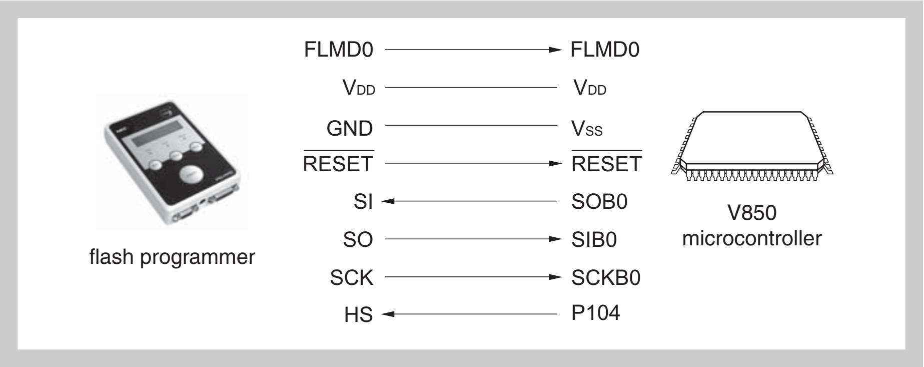


Figure 5-6 Communication with flash programmer via CSIB0 with handshake

The flash programmer outputs a transfer clock and the microcontroller operates as a slave.

If the PG-FP5 is used as the flash programmer, it generates the following signals for the microcontroller. For details, refer to the PG-FP5 User's Manual (U18865E).

Table 5-2 Signals generated by flash programmer PG-FP5

PG-FP5			Controller	Connection		
Signal name	I/O	Pin function	Pin name	UARTA0	CSIB0	CSIB0 + HS
FLMD0	Output	Write enable/disable, mode setting	FLMD0	○	○	○
FLMD1	Output	Mode setting	FLMD1	×	×	×
V_{DD}	I/O	V_{DD} voltage generation/voltage monitor	V_{DD}	○	○	○
GND	–	Ground	V_{SS}	○	○	○
CLK	Output	Clock output to the controller	X1	×	×	×
$\overline{\text{RESET}}$	Output	Reset signal	$\overline{\text{RESET}}$	○	○	○
SI/RxD	Input	Receive signal	SOB0/ TXDA0	○	○	○
SO/TxD	Output	Transmit signal	SIB0/RXDA0	○	○	○
SCK	Output	Transfer clock	SCKB0	×	○	○
HS	Input	Handshake signal for CSIB0 + HS communication	P104	×	×	○

Note ○: must be connected
×: does not need to be connected

5.3.3 Pin connection

A connector must be mounted on the target system to connect the flash programmer for on-board writing. In addition, a function to switch between the normal operation mode and flash memory programming mode must be provided on the board.

When the flash memory programming mode is set, all the pins not used for flash memory programming are in the same status as immediately after reset.

In the normal operation mode, 0 V is input to the FLMD0 pin. The pull-down resistor at the FLMD0 pin ensures normal operation mode if no flash programmer is connected. In the flash memory programming mode, the V_{DD} write voltage is supplied to the FLMD0 pin. Additionally the FLMD1 pin, shared with port P50, has to hold 0 V level.

An example of connection of the FLMD0 and FLMD1 pins is shown below. Alternatively the FLMD1 pin may also be connected directly to the FLMD1 signal of the flash programmer.

Table 5-3 Operation mode settings

Pins		Operation mode
FLMD0	FLMD1 (P50)	
0		Normal operation mode (fetch from flash)
1	0	Flash programming mode
	1	Setting prohibited

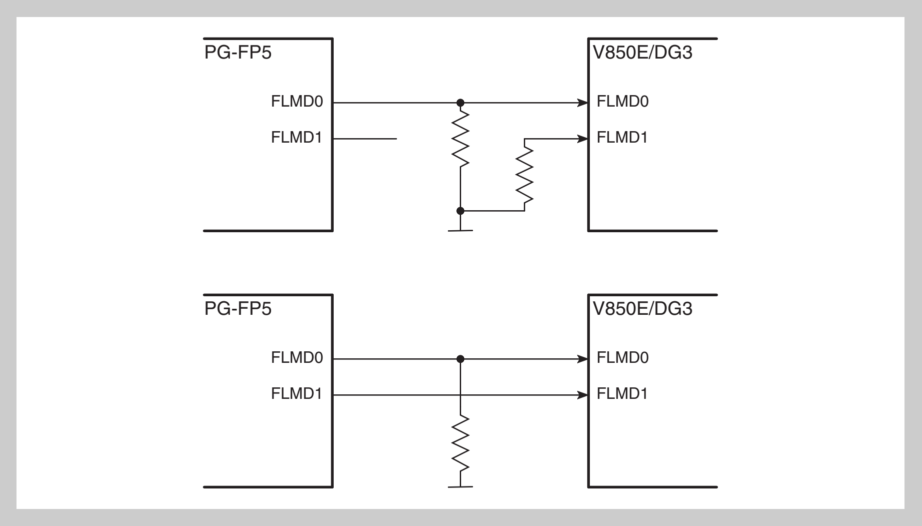


Figure 5-7 Example of connection to flash programmer PG-FP5 in CSI and UART mode

(1) Serial interface pins

The pins used by each serial interface are shown in the table below.

Table 5-4 Pins used by each serial interface

Serial interface	Pins
UARTA0	TXDA0, RXDA0 at pins P30/P31
CSIB0	SOB0, SIB0, SCKB0 at pins P105 - P107
CSIB0 + HS	SOB0, SIB0, SCKB0, P104

In flash programming mode the output drive strength control of the pins TXDA0, SOB0 and P104 is disabled. By this means the port pins provide maximum driver capability in order to maximize the transmission data rate to the flash programmer.

Caution

1. Since the output drive strength control of the pins TXDA0, SOB0 and P104 is disabled during programming these pins are not short-circuit proof any more. Short circuits at these pins may permanently damage the device.
2. If other devices are connected to the serial interface pins in use for flash memory programming in on-board programming mode take care that the concerned signals do not conflict with the signals of the flash programmer and the microcontroller. Output pins of the other devices must be isolated or set in high impedance state. Ensure that the other devices do not malfunction because of flash programmer signals.
3. Pay attention in particular if the flash programmer's $\overline{\text{RESET}}$ signal is connected also to an on-board reset generation circuit. The reset output of the reset generator may ruin the flash programming process and may need to be isolated.
4. All the port pins, including the pin connected to the flash programmer, go into an output high-impedance state in the flash memory programming mode. If there is a problem such as that an external device connected to a port prohibits the output high-impedance state, connect the port to V_{DD} or V_{SS} via a resistor.
5. Connect all oscillator pins in the same way as in the normal operation mode.
6. Supply the same power to all power supply pins, including reference voltages, power regulator pins, etc., as in the normal operation mode.

5.3.4 Programming method

In the following the flash programming flow is described, if the CSI or the UART is used as the communication interface.

(1) Flash memory control

The procedure to manipulate the flash memory is illustrated below.

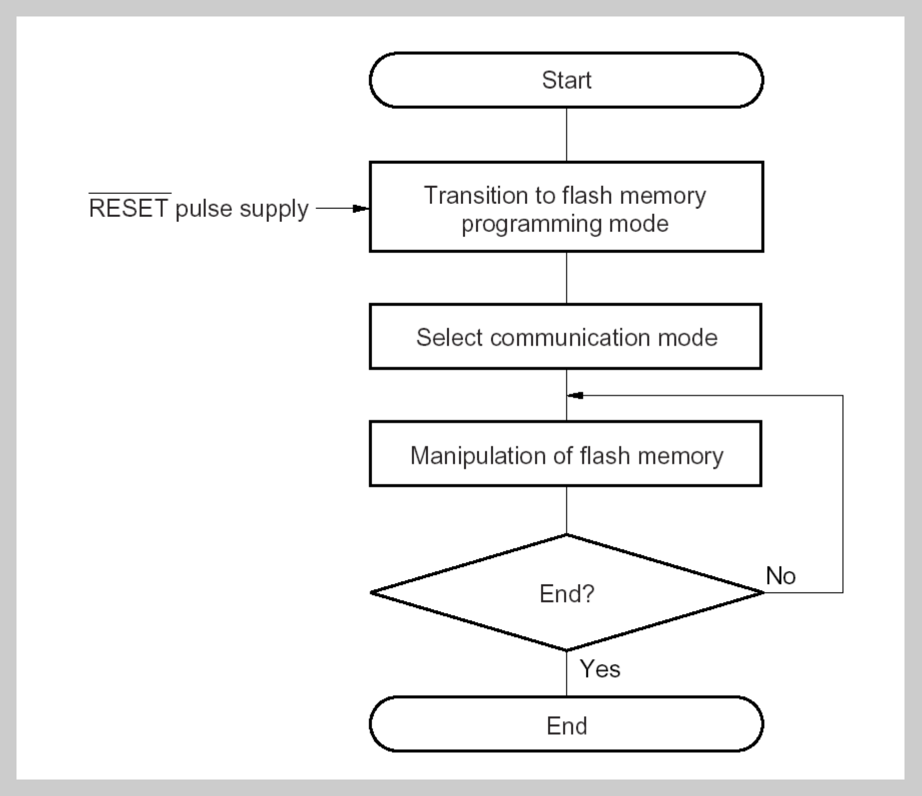

Start
RESET pulse supply
Transition to flash memory programming mode
Select communication mode
Manipulation of flash memory
End?
No
Yes
End

Figure 5-8 Flash memory manipulation procedure

(2) Flash memory programming mode

To rewrite the contents of the flash memory by using the flash programmer, set the microcontroller in the flash memory programming mode.

To set this mode, set the FLMD0 and FLMD1 pins as shown in *Table 5-6* and release RESET.

The communication interface is chosen by applying a specified number of pulses to the MODE pin after reset release. Note that this is handled by the flash programmer.

Figure 5-9 gives an example how the UARTA0 is established for the communication between the flash programmer and the microcontroller.

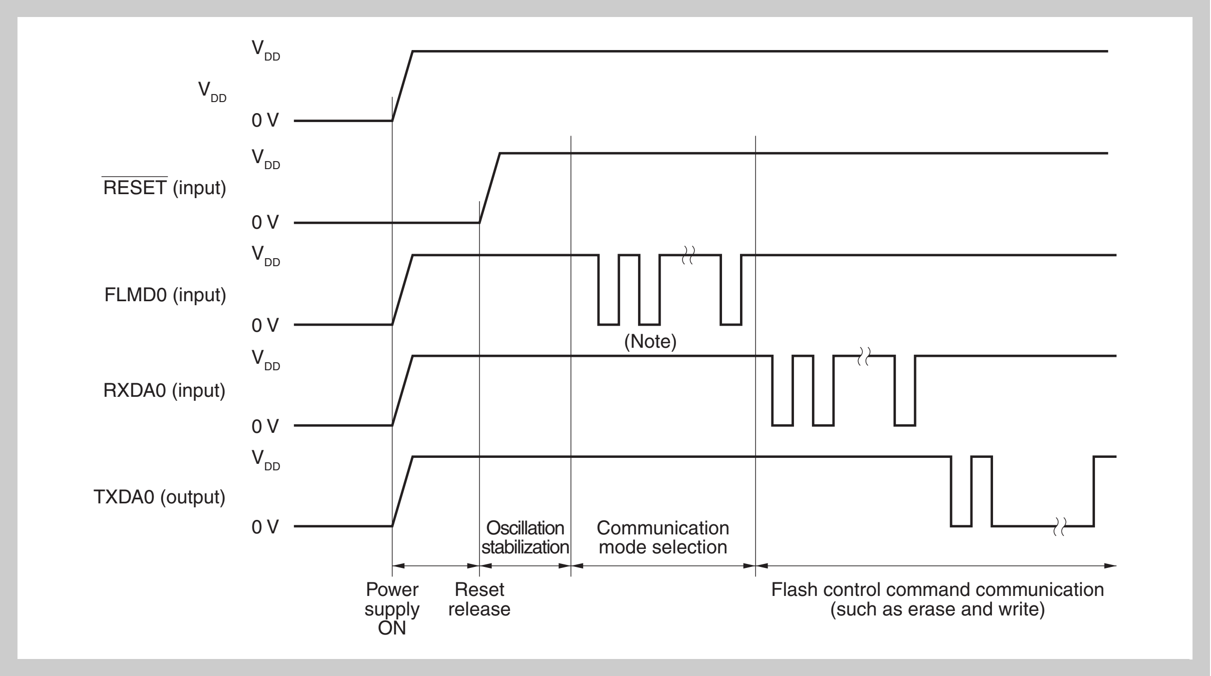


Figure 5-9 Flash memory programming mode start-up

Note The number of clocks to be inserted differs depending on the chosen communication mode. For details, refer to *Table 5-5*.

(3) Selecting communication mode

The communication mode is selected by applying a specified number of pulses to the MODE pin after the flash memory programming mode is set. These MODE pulses are generated by the flash programmer.

The relationship between the number of pulses and the communication mode is shown in the table below.

Table 5-5 Communication modes

MODE pulses	Communication mode	Remark
0	UARTA0	Communication rate: 9,600 bps (after reset), LSB first
8	CSIB0	microcontroller operates as slave, MSB first
11	CSIB0 + HS	microcontroller operates as slave, MSB first
Others	-	Setting prohibited

Note When UARTA0 is selected, the receive clock is calculated based on the reset command that is sent from the flash programmer after reception of the MODE pulses.

(4) Communication commands

The microcontroller communicates with the flash programmer via commands. The commands sent to the microcontroller are called commands, and the response signals sent by the microcontroller to the flash programmer are called response commands.

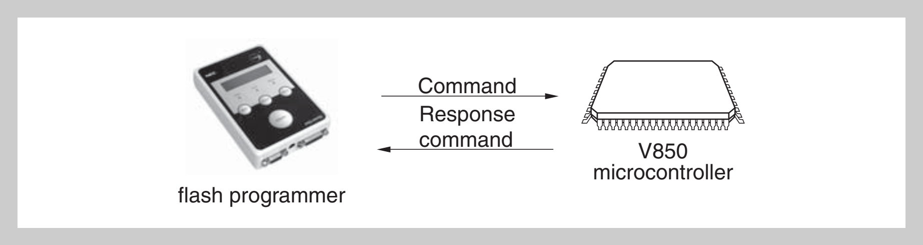


Figure 5-10 Communication commands

The following table lists the flash memory control commands of the microcontroller. All these commands are issued by the flash programmer, and the microcontroller performs the corresponding processing.

Table 5-6 Flash memory control commands

Classification	Command name	Support			Function
		CSIB	CSIB + HS	UARTA	
Blank check	Block blank check command	√	√	√	Checks erasure status of entire memory.
Erase	Chip erase command	√	√	√	Erase all memory contents including area that holds security flags, reset vector and other flash options
	Block erase command	√	√	√	Erases memory contents of specified block.
Write	Write command	√	√	√	Writes data by specifying write address and number of bytes to be written, and executes verify check.
Verify	Verify command	√	√	√	Compares input data with all memory contents.
Read	Read command	√	√	√	Read flash memory contents
System setting and control	Reset command	√	√	√	Escapes from each status.
	Oscillation frequency setting command	√	√	√	Sets oscillation frequency.
	Baud rate setting command	–	–	√	Sets baud rate when UART is used.
	Silicon signature command	√	√	√	Reads silicon signature information.
	Version acquisition command	√	√	√	Reads version information of device.
	Status command	√	√	–	Acquires operation status.
	Security setting command	√	√	√	Sets security of chip erasure, block erasure, and writing.

The microcontroller returns a response command to the command issued by the flash programmer. The response commands sent by the microcontroller are listed below.

Table 5-7 Response commands

Response command name	Function
ACK	Acknowledges command/data.
NAK	Acknowledges illegal command/data.

Chapter 6 Interrupt Controller (INTC)

This controller is provided with a dedicated Interrupt Controller (INTC) for interrupt servicing and can process a large amount of maskable and two non-maskable interrupt requests.

An interrupt is an event that occurs independently of program execution, and an exception is an event whose occurrence is dependent on program execution. Generally, an exception takes precedence over an interrupt.

This controller can process interrupt requests from the on-chip peripheral hardware and external sources. Moreover, exception processing can be started by the TRAP instruction (software exception) or by generation of an exception event (i.e. fetching of an illegal opcode) (exception trap).

Eight levels of software-programmable priorities can be specified for each interrupt request. Starting of interrupt servicing takes no fewer than 5 system clocks after the generation of an interrupt request.

6.1 Features

- Interrupts
  - Non-maskable interrupts: 2 sources
  - Maskable interrupts:

- internal peripherals:	45 sources
- external:	4 sources
- software:	2 sources

  - 8 levels of programmable priorities (maskable interrupts)
  - Multiple interrupt control according to priority
  - Masks can be specified for each maskable interrupt request
  - Noise elimination, edge detection and valid edge specification, level detection for external interrupt request signals
  - Wake-up capable
    (analogue noise elimination for external interrupt request signals)
  - NMI and INTP0 share the same pin
- Exceptions
  - Software exceptions: 2 channels with each 16 sources
  - Exception traps: 2 sources (illegal opcode exception and debug trap)

Table 6-1 Interrupt/exception source list (1/3)

Type	Classification	Interrupt/Exception Source			Default Priority	Exception Code	Handler Address	Restored PC
		Name	Generating Source	Generating Unit				
Reset	Interrupt	RESET	$\overline{\text{RESET}}$ input	Pin	–	0000_H	00000000_H	undef.
Non-maskable	Interrupt	NMI0	NMI Input	PORT	–	0010_H	00000010_H	nextPC
		NMIWDT	Watchdog Timer	WDT	–	0020_H	00000020_H	nextPC
		NMI2	Unused	–	–	0030_H	00000030_H	nextPC
Software exception	Exception	TRAP0n (n = 0 to F_H)	TRAP instruction	–	–	004nH (n = 0 to F_H)	00000040_H	nextPC
	Exception	TRAP1n (n = 0 to F_H)	TRAP instruction	–	–	005nH (n = 0 to F_H)	00000050_H	nextPC
Exception trap	Exception	ILGOP/ DBTRAP	Illegal opcode/ DBTRAP instruction	–	–	0060_H	00000060_H	nextPC
Maskable	Interrupt	Reserved	Reserved	—	0...1	$0xx0_H$	$00000xx0_H$	nextPC
	Interrupt	INTWT0UV	WT0 underflow	WT0	2	$00A0_H$	$000000A0_H$	next PC
	Interrupt	INTWT1UV	WT1 underflow	WT1	3	$00B0_H$	$000000B0_H$	next PC
	Interrupt	Reserved	Reserved	–	4	$00C0_H$	$000000C0_H$	next PC
	Interrupt	INTTM01	Watch calibration timer capture compare	WCT	5	$00D0_H$	$000000D0_H$	next PC
	Interrupt	INTP0	External interrupt 0	PORT	6	$00E0_H$	$000000E0_H$	next PC
	Interrupt	INTP1	External interrupt 1	PORT	7	$00F0_H$	$000000F0_H$	next PC
	Interrupt	INTP2	External interrupt 2	PORT	8	0100_H	00000100_H	next PC
	Interrupt	INTP3	External interrupt 3	PORT	9	0110_H	00000110_H	next PC
	Interrupt	Reserved	Reserved	—	10...12	$0xx0_H$	$00000xx0_H$	nextPC
	Interrupt	INTTZ0UV	TMZ0 underflow	TMZ0	13	0150_H	00000150_H	next PC
	Interrupt	INTTZ1UV	TMZ1 underflow	TMZ1	14	0160_H	00000160_H	next PC
	Interrupt	INTTZ2UV	TMZ2 underflow	TMZ2	15	0170_H	00000170_H	next PC
	Interrupt	INTTZ3UV	TMZ4 underflow	TMZ3	16	0180_H	00000180_H	next PC
	Interrupt	INTTZ4UV	TMZ4 underflow	TMZ4	17	0190_H	00000190_H	nextPC
	Interrupt	INTTZ5UV	TMZ5 underflow	TMZ5	18	$01A0_H$	00000190_H	nextPC
	Interrupt	INTTP0OV	TMP0 overflow	TMP0	19	$01B0_H$	$000001B0_H$	next PC
	Interrupt	INTTP0CC0	TMP0 capture compare channel 0	TMP0	20	$01C0_H$	$000001C0_H$	next PC
	Interrupt	INTTP0CC1	TMP0 capture compare channel 1	TMP0	21	$01D0_H$	$000001D0_H$	next PC
	Interrupt	Reserved	Reserved	—	22...30	$0xx0_H$	$00000xx0_H$	nextPC
	Interrupt	INTTG0OV0	TMG0 overflow interrupt 0	TMG0	31	0270_H	00000270_H	next PC
	Interrupt	INTTG0OV1	TMG0 overflow interrupt 1	TMG0	32	0280_H	00000280_H	next PC
	Interrupt	INTTG0CC0	TMG0 capture compare channel 0	TMG0	33	0290_H	00000290_H	next PC
	Interrupt	INTTG0CC1	TMG0 capture compare channel 1	TMG0	34	$02A0_H$	$000002A0_H$	next PC
	Interrupt	INTTG0CC2	TMG0 capture compare channel 2	TMG0	35	$02B0_H$	$000002B0_H$	next PC

Table 6-1 Interrupt/exception source list (2/3)

Type	Classification	Interrupt/Exception Source			Default Priority	Exception Code	Handler Address	Restored PC
		Name	Generating Source	Generating Unit				
Maskable	Interrupt	INTTG0CC3	TMG0 capture compare channel 3	TMG0	36	02C0$_H$	000002C0$_H$	next PC
	Interrupt	INTTG0CC4	TMG0 capture compare channel 4	TMG0	37	02D0$_H$	000002D0$_H$	next PC
	Interrupt	INTTG0CC5	TMG0 capture compare channel 5	TMG0	38	02E0$_H$	000002E0$_H$	next PC
	Interrupt	INTTG1OV0	TMG1 overflow interrupt 0	TMG1	39	02F0$_H$	000002F0$_H$	next PC
	Interrupt	INTTG1OV1	TMG1 overflow interrupt 1	TMG1	40	0300$_H$	00000300$_H$	next PC
	Interrupt	INTTG1CC0	TMG1 capture compare channel 0	TMG1	41	0310$_H$	00000310$_H$	next PC
	Interrupt	INTTG1CC1	TMG1 capture compare channel 1	TMG1	42	0320$_H$	00000320$_H$	next PC
	Interrupt	INTTG1CC2	TMG1 capture compare channel 2	TMG1	43	0330$_H$	00000330$_H$	next PC
	Interrupt	INTTG1CC3	TMG1 capture compare channel 3	TMG1	44	0340$_H$	00000340$_H$	next PC
	Interrupt	INTTG1CC4	TMG1 capture compare channel 4	TMG1	45	0350$_H$	00000350$_H$	next PC
	Interrupt	INTTG1CC5	TMG1 capture compare channel 5	TMG1	46	0360$_H$	00000360$_H$	next PC
	Interrupt	Reserved	Reserved	—	47...48	0xx0$_H$	00000xx0$_H$	next PC
	Interrupt	INTAD	ADC end of conversion	ADC	49	0390$_H$	00000390$_H$	next PC
	Interrupt	INTC0ERR	CAN0 error interrupt	CAN0	50	03A0$_H$	000003A0$_H$	next PC
	Interrupt	INTC0WUP	CAN0 wake up interrupt	CAN0	51	03B0$_H$	000003B0$_H$	next PC
	Interrupt	INTC0REC	CAN0 receive interrupt	CAN0	52	03C0$_H$	000003C0$_H$	next PC
	Interrupt	INTC0TRX	CAN0 transmit interrupt	CAN0	53	03D0$_H$	000003D0$_H$	next PC
	Interrupt	INTCB0RE	CSIB0 receive error interrupt	CSIB0	54	03E0$_H$	000003E0$_H$	next PC
	Interrupt	INTCB0R	CSIB0 receive complete interrupt	CSIB0	55	03F0$_H$	000003F0$_H$	next PC
	Interrupt	INTCB0T	CSIB0 transmit interrupt	CSIB0	56	0400$_H$	00000400$_H$	next PC
	Interrupt	INTUA0RE	UARTA0 receive error interrupt	UARTA0	57	0410$_H$	00000410$_H$	next PC
	Interrupt	INTUA0R	UARTA0 receive complete interrupt	UARTA0	58	0420$_H$	00000420$_H$	next PC
	Interrupt	INTUA0T	UARTA0 transmit interrupt	UARTA0	59	0430$_H$	00000430$_H$	next PC
	Interrupt	INTUA1RE	UARTA1 receive error interrupt	UARTA1	60	0440$_H$	00000440$_H$	next PC
	Interrupt	INTUA1R	UARTA1 receive complete interrupt	UARTA1	61	0450$_H$	00000450$_H$	next PC
	Interrupt	INTUA1T	UARTA1 transmit interrupt	UARTA1	62	0460$_H$	00000460$_H$	next PC
	Interrupt	INTIIC0	IIC0 interrupt	IIC0	63	0470$_H$	00000470$_H$	next PC

Table 6-1 Interrupt/exception source list (3/3)

Type	Classification	Interrupt/Exception Source			Default Priority	Exception Code	Handler Address	Restored PC
		Name	Generating Source	Generating Unit				
Maskable	Interrupt	Reserved	Reserved	—	64...69	$0xx0_H$	$00000xx0_H$	next PC
	Interrupt	INT70	not generated by hardware[a]	—	70	$04E0_H$	$000004E0_H$	next PC
	Interrupt	INT71	not generated by hardware[a]	—	71	$04F0_H$	$000004F0_H$	next PC
	Interrupt	Reserved	Reserved	—	72...88	$0xx0_H$	$00000xx0_H$	next PC
	Interrupt	INTCB1RE	CSIB1 receive error interrupt	CSIB1	89	0610_H	00000610_H	next PC
	Interrupt	INTCB1R	CSIB1 receive complete interrupt	CSIB1	90	0620_H	00000620_H	next PC
	Interrupt	INTCB1T	CSIB1 transmit interrupt	CSIB1	91	0630_H	00000630_H	next PC

[a] These interrupts can be used as software triggered interrupts.

Default priority: The priority order when two or more maskable interrupt requests are generated at the same time.
The highest priority is 0.

1. Restored PC: The value of the PC saved to EIPC or FEPC when interrupt/exception processing is started. However, the value of the PC saved when an interrupt is acknowledged during division (DIV, DIVH, DIVU, DIVHU) instruction execution is the value of the PC of the current instruction (DIV, DIVH, DIVU, DIVHU).
2. nextPC: The PC value that starts the processing following interrupt/exception processing.
3. The execution address of the illegal instruction when an illegal opcode exception occurs is calculated by (Restored PC – 4).

6.2 Non-Maskable Interrupts

A non-maskable interrupt request is acknowledged unconditionally, even when interrupts are in the interrupt disabled (DI) status.

Non-maskable interrupts of this microcontroller are available for the following two requests:

- NMI0: NMI pin input
- NMIWDT: Non-maskable Watchdog Timer interrupt request

When the valid edge specified by the ESEL0, ESEL00 and ESEL01 bits of the Interrupt mode register 0(INTM0) is detected on the NMI pin, the interrupt occurs.

The Watchdog Timer interrupt request is only effective as non-maskable interrupt if the WDTMODE bit of the Watchdog Timer mode register (WDTM) is set 0.

If multiple non-maskable interrupts are generated at the same time, the highest priority servicing is executed according to the following priority order (the lower priority interrupt is ignored):

NMIWDT > NMI0

Note that if a NMI from port pin or NMIWDT request is generated while NMI from port pin is being serviced, the service is executed as follows.

(1) If a NMI0 is generated while NMI0 is being serviced

The new NMI0 request is held pending regardless of the value of the PSW.NP bit. The pending NMIVC request is acknowledged after servicing of the current NMI0 request has finished (after execution of the RETI instruction).

(2) If a NMIWDT request is generated while NMI0 is being serviced

If the PSW.NP bit remains set (1) while NMI0 is being serviced, the new NMIWDT request is held pending. The pending NMIWDT request is acknowledge after servicing of the current NMI0 request has finished (after execution of the RETI instruction).

If the PSW.NP bit is cleared (0) while NMI0 is being serviced, the newly generated NMIWDT request is executed (NMI0 servicing is halted).

Caution

1. Although the values of the PC and PSW are saved to an NMI status save register (FEPC, FEPSW) when a non-maskable interrupt request is generated, only the NMI0 can be restored by the RETI instruction at this time. Because NMIWDT cannot be restored by the RETI instruction, the system must be reset after servicing this interrupt.
2. If PSW.NP is cleared to 0 by the LDSR instruction during non-maskable interrupt servicing, a NMI0 interrupt afterwards cannot be acknowledged correctly.

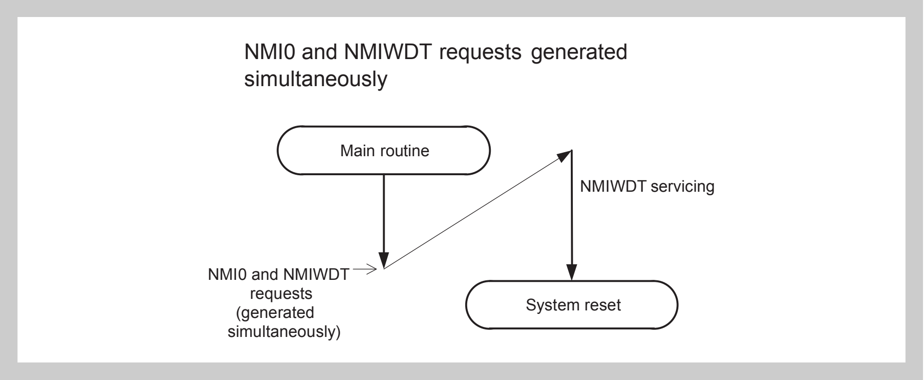

NMI0 and NMIWDT requests generated simultaneously
Main routine
NMIWDT servicing
NMI0 and NMIWDT requests (generated simultaneously)
System reset

Figure 6-1 Example of non-maskable interrupt request acknowledgement operation: multiple NMI requests generated at the same time

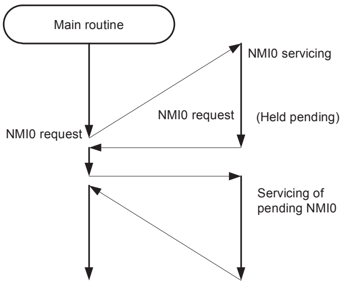
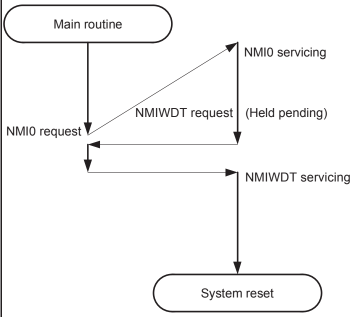
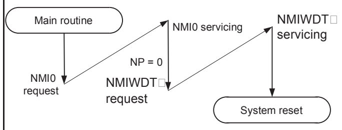
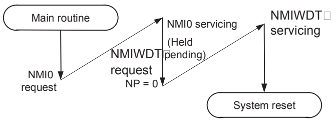
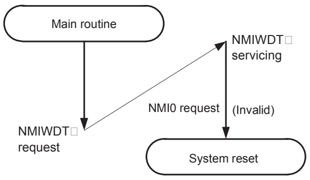
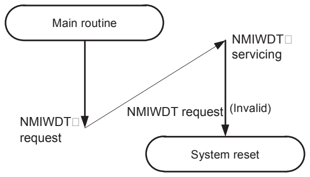
NMI being serviced	NMI request generated during NMI servicing	
	NMI0	NMIWDT
NMI0	NMI0 request generated during NMI0 servicing  Main routine NMI0 servicing NMI0 request (Held pending) NMI0 request Servicing of pending NMI0	NMIWDT request generated during NMI0 servicing (NP = 1 retained before NMI1 request)  Main routine NMI0 servicing NMIWDT request (Held pending) NMI0 request NMIWDT servicing System reset NMIWDT request generated during NMI0 servicing (NP=0 set before NMIWDT request)  Main routine NMI0 servicing NMIWDT servicing NP = 0 NMI0 request NMIWDT request System reset NMIWDT request generated during NMI0 servicing (NP=0 set after NMIWDT request)  Main routine NMI0 servicing NMIWDT servicing (Held pending) NMIWDT request NP = 0 NMI0 request System reset
NMIWDT	NMI0 request generated during NMIWDT servicing  Main routine NMIWDT servicing NMI0 request (Invalid) NMIWDT request System reset	NMI1 request generated during NMIWDT servicing  Main routine NMIWDT servicing NMIWDT request (Invalid) NMIWDT request System reset

Figure 6-2 Example of non-maskable interrupt request acknowledgement operation: NMI request generated during NMI servicing

6.2.1 Operation

If a non-maskable interrupt is generated, the CPU performs the following processing, and transfers control to the handler routine:

(1) Saves the restored PC to FEPC.

(2) Saves the current PSW to FEPSW.

(3) Writes exception code 0010_H to the higher halfword (FECC) of ECR.

(4) Sets the NP and ID bits of the PSW and clears the EP bit.

(5) Sets the handler address corresponding to the non-maskable interrupt to the PC, and transfers control.

The processing configuration of a non-maskable interrupt is shown in *Figure 6-3*.

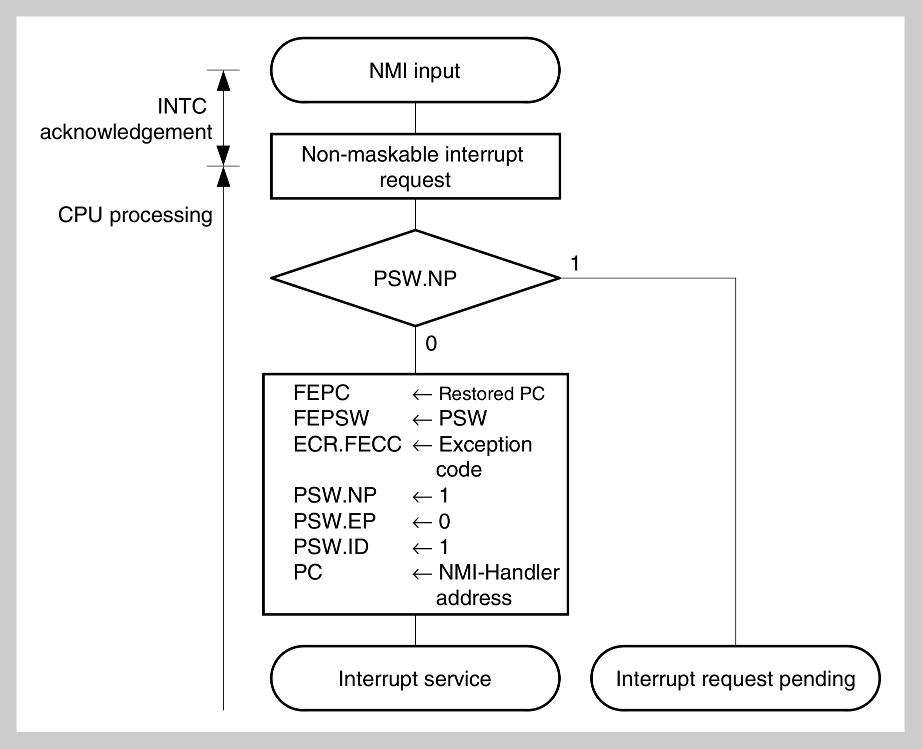


Figure 6-3 Processing configuration of non-maskable interrupt

6.2.2 Restore

(1) NMI0

Execution is restored from the non-maskable interrupt (NMI0) processing by the RETI instruction.

When the RETI instruction is executed, the CPU performs the following processing, and transfers control to the address of the restored PC.

<1> Restores the values of the PC and the PSW from FEPC and FEPSW, respectively, because the EP bit of the PSW is 0 and the NP bit of the PSW is 1.

<2> Transfers control back to the address of the restored PC and PSW.

Figure 6-4 illustrates how the RETI instruction is processed.

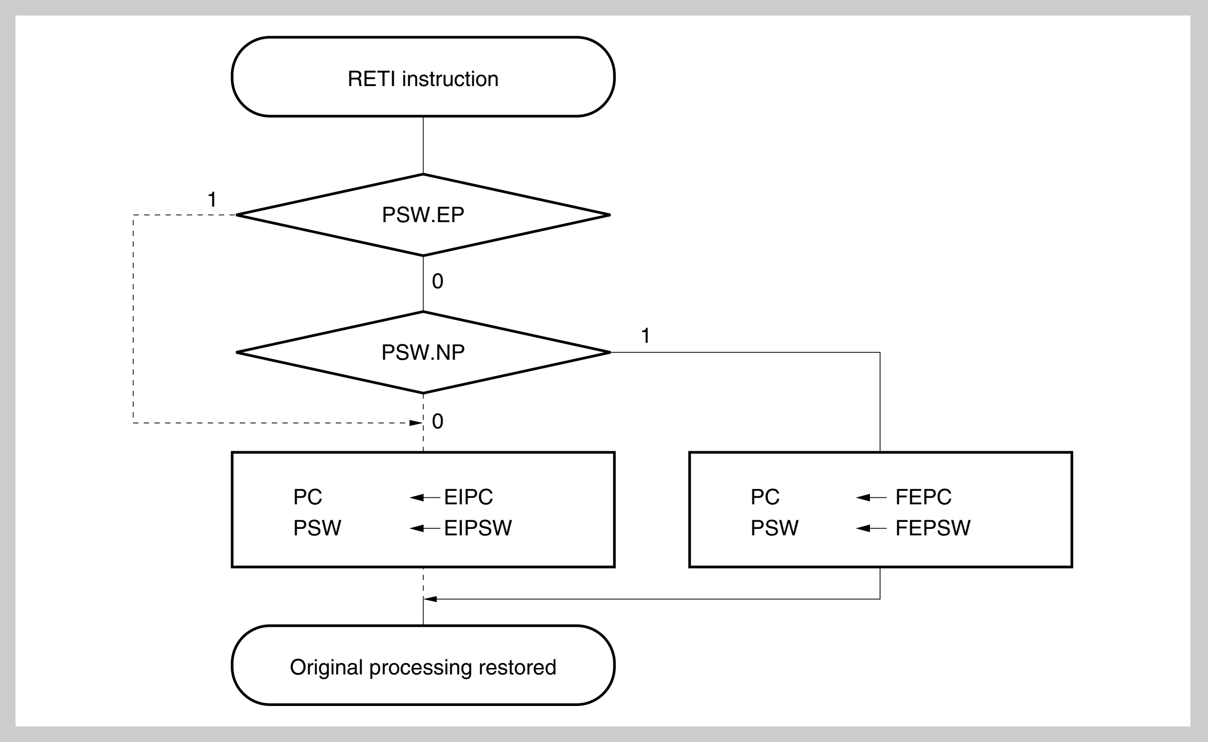


Figure 6-4 RETI instruction processing

Caution When the PSW.EP bit and PSW.NP bit are changed by the LDSR instruction during non-maskable interrupt processing, in order to restore the PC and PSW correctly during recovery by the RETI instruction, it is necessary to set PSW.EP back to 0 and PSW.NP back to 1 using the LDSR instruction immediately before the RETI instruction.

Note The solid line indicates the CPU processing flow.

(2) NMIWDT

Restoring by RETI instruction is not possible. Perform a system reset after interrupt servicing.

6.2.3 Non-maskable interrupt status flag (NP)

The NP flag is a status flag that indicates that non-maskable interrupt (NMI) processing is under execution.

This flag is set when an NMI interrupt has been acknowledged, and masks all interrupt requests and exceptions to prohibit multiple interrupts from being acknowledged.

	31																							8	7	6	5	4	3	2	1	0	Initial value
PSW	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	NP	EP	ID	SAT	CY	OV	S	Z	00000020H

Bit position	Bit name	Function
7	NP	Indicates whether NMI interrupt processing is in progress. 0: No NMI interrupt processing 1: NMI interrupt currently being processed

6.2.4 NMI0 control

The NMI0 can be configured to generate an NMI upon a rising, falling or both edges at the NMI pin. To enable respectively disable the NMI0 and to configure the edge refer to *“Edge and Level Detection Configuration“ on page 204*.

6.3 Maskable Interrupts

Maskable interrupt requests can be masked by interrupt control registers.

If two or more maskable interrupt requests are generated at the same time, they are acknowledged according to the default priority. In addition to the default priority, eight levels of priorities can be specified by using the interrupt control registers (programmable priority control).

When an interrupt request has been acknowledged, the acknowledgement of other maskable interrupt requests is disabled and the interrupt disabled (DI) status is set.

When the EI instruction is executed in an interrupt processing routine, the interrupt enabled (EI) status is set, which enables servicing of interrupts having a higher priority than the interrupt request in progress (specified by the interrupt control register). Note that only interrupts with a higher priority will have this capability; interrupts with the same priority level cannot be nested.

However, if multiple interrupts are executed, the following processing is necessary.

(1) Save EIPC and EIPSW in memory or a general-purpose register before executing the EI instruction.

(2) Execute the DI instruction before executing the RETI instruction, then reset EIPC and EIPSW with the values saved in (1).

6.3.1 Operation

If a maskable interrupt occurs by INT input, the CPU performs the following processing, and transfers control to a handler routine:

(1) Saves the restored PC to EIPC.

(2) Saves the current PSW to EIPSW.

(3) Writes an exception code to the lower halfword of ECR (EICC).

(4) Sets the ID bit of the PSW and clears the EP bit.

(5) Sets the handler address corresponding to each interrupt to the PC, and transfers control.

The processing configuration of a maskable interrupt is shown in *Figure 6-5*.

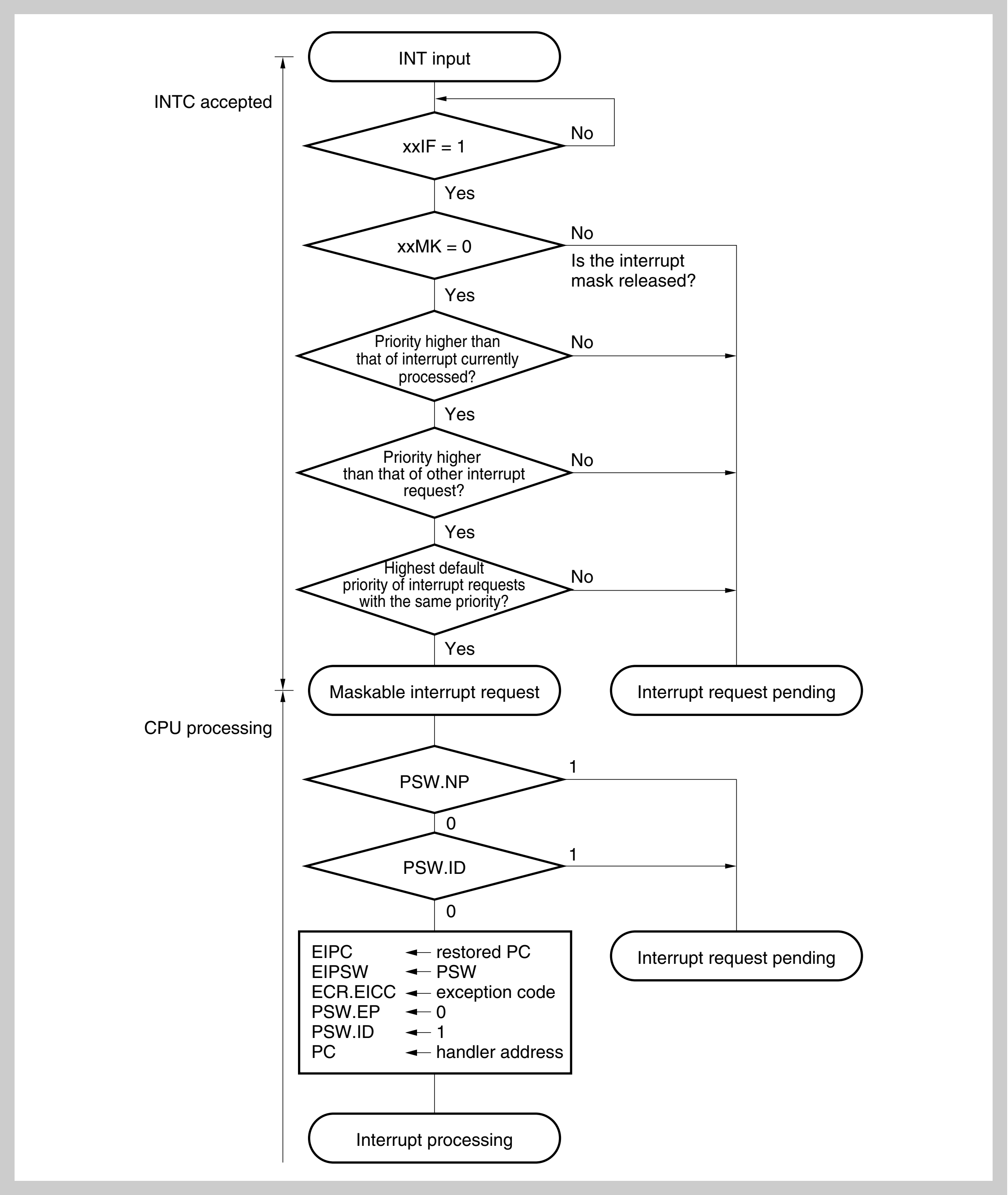

INTC accepted
INT input
xxIF = 1
No
Yes
xxMK = 0
No
Is the interrupt mask released?
Yes
Priority higher than that of interrupt currently processed?
No
Yes
Priority higher than that of other interrupt request?
No
Yes
Highest default priority of interrupt requests with the same priority?
No
Yes
Maskable interrupt request
Interrupt request pending
CPU processing
PSW.NP
1
0
PSW.ID
1
0
EIPC ← restored PC
EIPSW ← PSW
ECR.EICC ← exception code
PSW.EP ← 0
PSW.ID ← 1
PC ← handler address
Interrupt request pending
Interrupt processing

Figure 6-5 Maskable interrupt processing

Note For the ISPR register, see *“ISPR - In-service priority register“ on page 202*.

An INT input masked by the Interrupt Controllers and an INT input that occurs while another interrupt is being processed (when PSW.NP = 1 or PSW.ID = 1) are held pending internally by the Interrupt Controller. In such case, if the interrupts are unmasked, or when PSW.NP = 0 and PSW.ID = 0 as set by the RETI and LDSR instructions, input of the pending INT starts the new maskable interrupt processing.

6.3.2 Restore

Recovery from maskable interrupt processing is carried out by the RETI instruction.

When the RETI instruction is executed, the CPU performs the following steps, and transfers control to the address of the restored PC.

(1) Restores the values of the PC and the PSW from EIPC and EIPSW because the EP bit of the PSW is 0 and the NP bit of the PSW is 0.

(2) Transfers control to the address of the restored PC and PSW.

Figure 6-6 illustrates the processing of the RETI instruction.

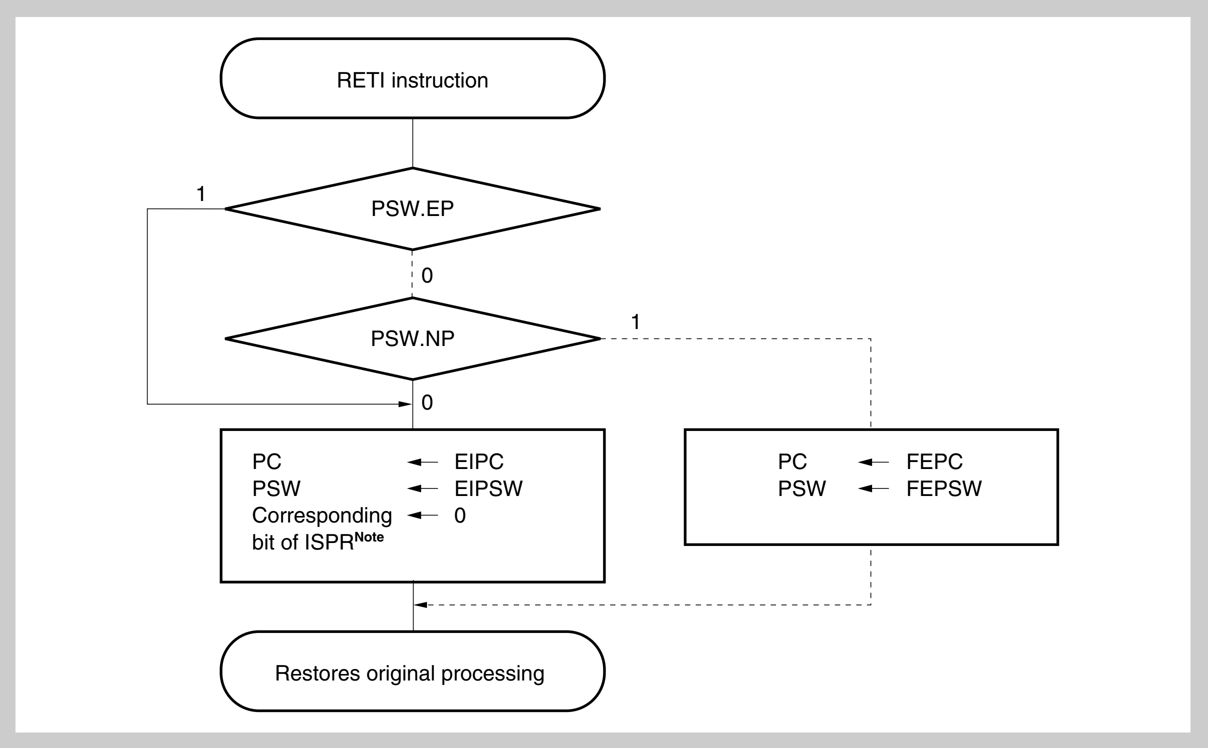


Figure 6-6 RETI instruction processing

Note 1. For the ISPR register, see *“ISPR - In-service priority register“ on page 202*.

2. The solid lines show the CPU processing flow.

Caution When the PSW.EP bit and the PSW.NP bit are changed by the LDSR instruction during maskable interrupt processing, in order to restore the PC and PSW correctly during recovery by the RETI instruction, it is necessary to set PSW.EP back to 0 and PSW.NP back to 0 using the LDSR instruction immediately before the RETI instruction.

6.3.3 Priorities of maskable interrupts

This microcontroller provides multiple interrupt servicing in which an interrupt is acknowledged while another interrupt is being serviced. Multiple interrupts can be controlled by priority levels.

There are two types of priority level control: control based on the default priority levels, and control based on the programmable priority levels that are specified by the interrupt priority level specification bit (xxPRn) of the interrupt control register (xxICn). When two or more interrupts having the same priority level specified by the xxPRn bit are generated at the same time, interrupts are serviced in order depending on the priority level allocated to each interrupt request type (default priority level) beforehand. For more information, refer to the interrupt/exception source list table. The programmable priority control customizes interrupt requests into eight levels by setting the priority level specification flag.

Note that when an interrupt request is acknowledged, the ID flag of PSW is automatically set to 1. Therefore, when multiple interrupts are to be used, clear the ID flag to 0 beforehand (for example, by placing the EI instruction in the interrupt service program) to set the interrupt enable mode.

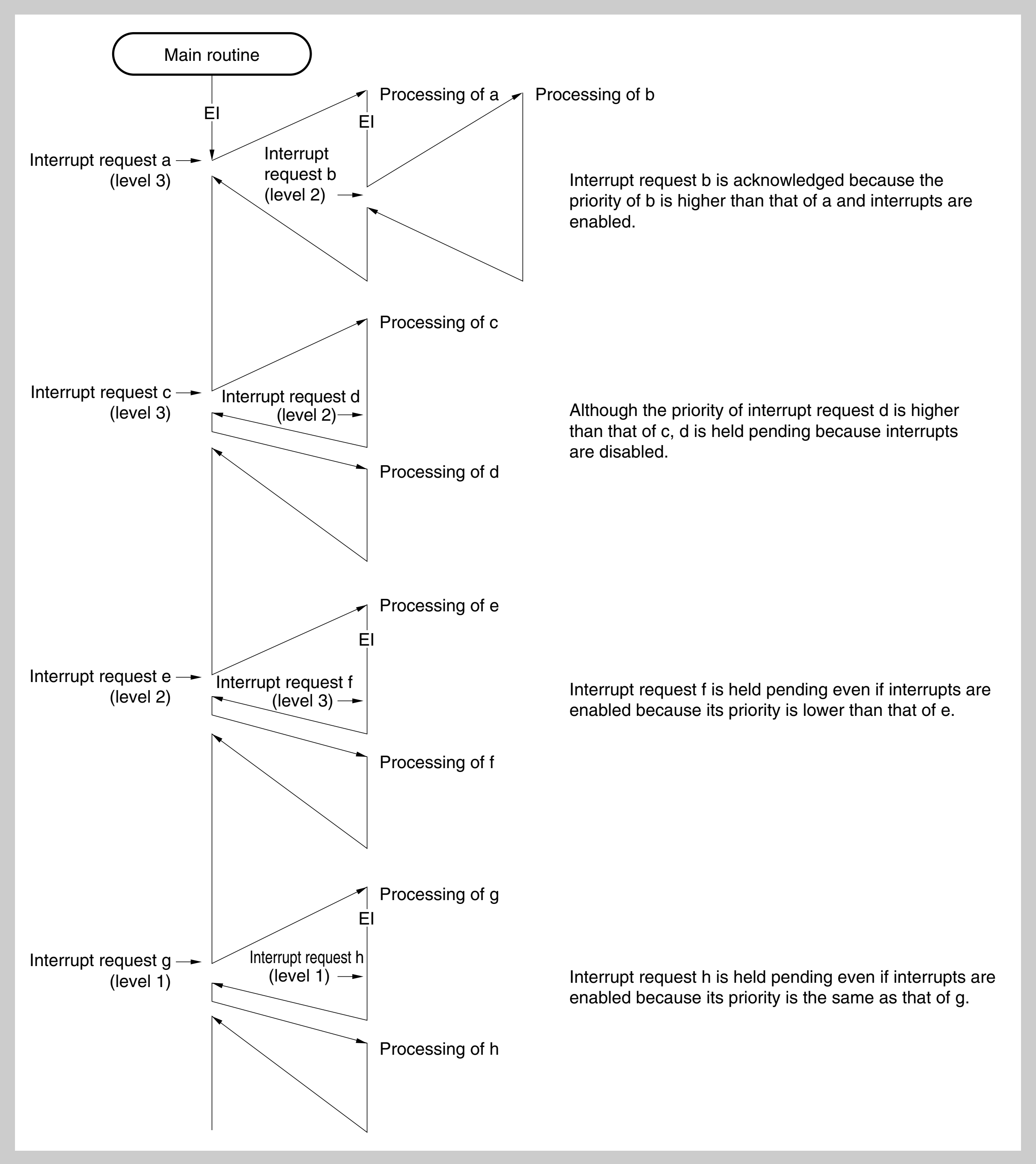
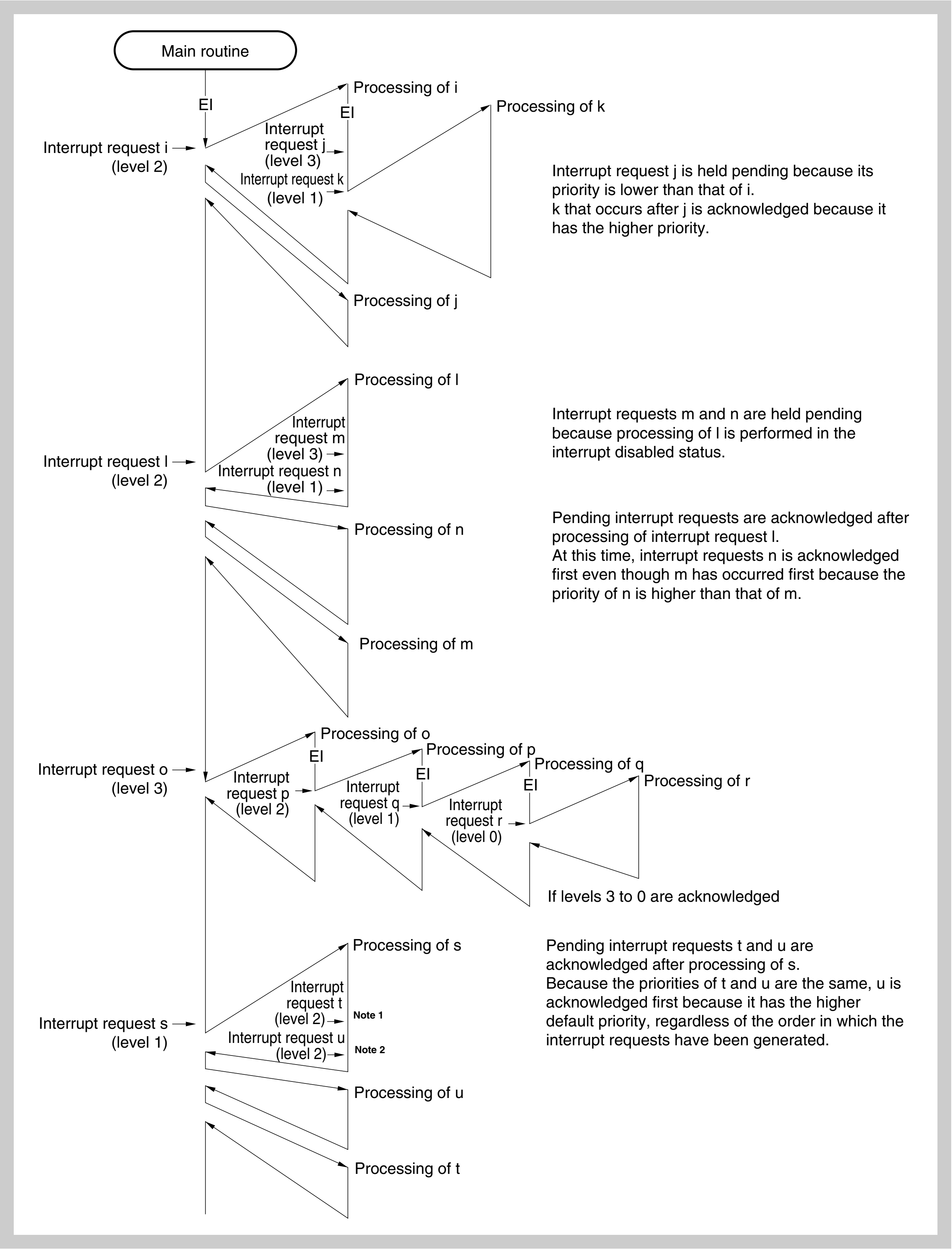

Main routine
EI
Interrupt request a
(level 3)
Processing of a
EI
Interrupt
request b
(level 2)
Processing of b
Interrupt request b is acknowledged because the priority of b is higher than that of a and interrupts are enabled.
Processing of c
Interrupt request c
(level 3)
Interrupt request d
(level 2)
Although the priority of interrupt request d is higher than that of c, d is held pending because interrupts are disabled.
Processing of d
Processing of e
EI
Interrupt request e
(level 2)
Interrupt request f
(level 3)
Interrupt request f is held pending even if interrupts are enabled because its priority is lower than that of e.
Processing of f
Processing of g
EI
Interrupt request g
(level 1)
Interrupt request h
(level 1)
Interrupt request h is held pending even if interrupts are enabled because its priority is the same as that of g.
Processing of h

Figure 6-7 Example of processing in which another interrupt request is issued while an interrupt is being processed (1/2)

Caution The values of the EIPC and EIPSW registers must be saved before executing multiple interrupts. When returning from multiple interrupt servicing, restore the values of EIPC and EIPSW after executing the DI instruction.

Note

1. <a> to <u> in the figure are the temporary names of interrupt requests shown for the sake of explanation.
2. The default priority in the figure indicates the relative priority between two interrupt requests.


Main routine
EI
Interrupt request i (level 2)
Processing of i
EI
Interrupt request j (level 3)
Interrupt request k (level 1)
Processing of k
Processing of j
Interrupt request l (level 2)
Processing of l
Interrupt request m (level 3)
Interrupt request n (level 1)
Processing of n
Processing of m
Interrupt request o (level 3)
Processing of o
EI
Interrupt request p (level 2)
Processing of p
EI
Interrupt request q (level 1)
Processing of q
EI
Interrupt request r (level 0)
Processing of r
Interrupt request s (level 1)
Processing of s
Interrupt request t (level 2)
Note 1
Interrupt request u (level 2)
Note 2
Processing of u
Processing of t

Interrupt request j is held pending because its priority is lower than that of i.
k that occurs after j is acknowledged because it has the higher priority.

Interrupt requests m and n are held pending because processing of l is performed in the interrupt disabled status.

Pending interrupt requests are acknowledged after processing of interrupt request l.
At this time, interrupt requests n is acknowledged first even though m has occurred first because the priority of n is higher than that of m.

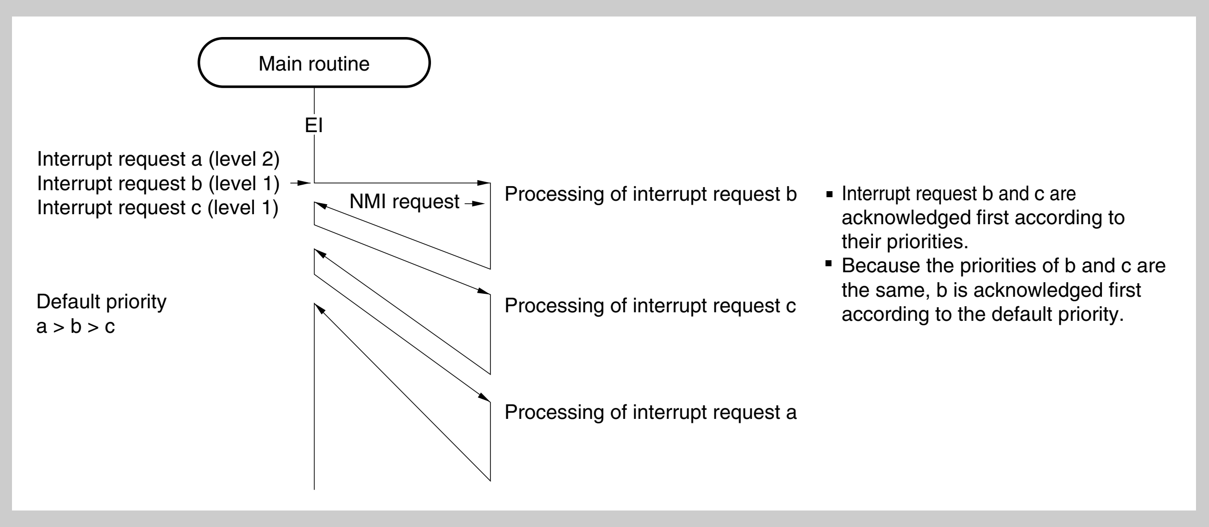
If levels 3 to 0 are acknowledged

Pending interrupt requests t and u are acknowledged after processing of s.
Because the priorities of t and u are the same, u is acknowledged first because it has the higher default priority, regardless of the order in which the interrupt requests have been generated.

Figure 6-8 Example of processing in which another interrupt request is issued while an interrupt is being processed (2/2)

Caution The values of the EIPC and EIPSW registers must be saved before executing multiple interrupts. When returning from multiple interrupt servicing, restore the values of EIPC and EIPSW after executing the DI instruction.

Note
1. Lower default priority
2. Higher default priority


Main routine
EI
Interrupt request a (level 2)
Interrupt request b (level 1)
Interrupt request c (level 1)
NMI request
Processing of interrupt request b
Processing of interrupt request c
Processing of interrupt request a
Default priority
a > b > c

- Interrupt request b and c are acknowledged first according to their priorities.
- Because the priorities of b and c are the same, b is acknowledged first according to the default priority.

Figure 6-9 Example of processing interrupt requests simultaneously generated

Caution The values of the EIPC and EIPSW registers must be saved before executing multiple interrupts. When returning from multiple interrupt servicing, restore the values of EIPC and EIPSW after executing the DI instruction.

Remark <a> to <c> in the figure are the temporary names of interrupt requests shown for the sake of explanation.

6.3.4 xxIC - Maskable interrupts control register

An interrupt control register is assigned to each interrupt request (maskable interrupt) and sets the control conditions for each maskable interrupt request.

Access This register can be read/written in 8-bit or 1-bit units.

Address FFFF F110$_H$ to FFFF F18E$_H$ (refer to *Table 6-3 on page 198*)

Initial Value 47$_H$

7	6	5	4	3	2	1	0
xxIF	xxMK	0	0	0	xxPR2	xxPR1	xxPR0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 6-2 xxIC register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>7</td><td>xxIF</td><td>This is an interrupt request flag.
0: Interrupt request not issued
1: Interrupt request issued
The flag xxIFn is reset automatically by the hardware if an interrupt request is acknowledged.</td></tr>
<tr><td>6</td><td>xxMK</td><td>This is an interrupt mask flag.
0: Enables interrupt processing
1: Disables interrupt processing (pending)</td></tr>
<tr><td>2 to 0</td><td>xxPR2 to xxPR0</td><td>8 levels of priority order are specified for each interrupt.
<table>
<tr><th>xxPR2</th><th>xxPR1</th><th>xxPR0</th><th>Interrupt priority specification bit</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>Specifies level 0 (highest)</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Specifies level 1</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>Specifies level 2</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>Specifies level 3</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>Specifies level 4</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Specifies level 5</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>Specifies level 6</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>Specifies level 7 (lowest)</td></tr>
</table></td></tr>
</table>

Note xx: identification name of each peripheral unit (WT0UV-WT1UV, TM01, P0-P3, TZ0UV-TZ5UV, TP0OV, TP0CC0, TP0CC1, TG0OV0-TG1OV0, TG0OV1-TG1OV1, TG0CC0-TG1CC0, TG0CC1-TG1CC1, TG0CC2-TG1CC2, TG0CC3-TG1CC3, TG0CC4-TG1CC4, TG0CC5-TG1CC5, AD, C0ERR, C0WUP, C0REC, C0TRX, CB0RE-CB1RE, CB0R-CB1R, CB0T-CB1T, UA0RE-UA1RE, UA0R-UA1R, UA0T-UA1T, IIC0, INT70, INT71)

.

The address and bit of each interrupt control register are shown in the following table.

Table 6-3 Addresses and bits of interrupt control registers (1/2)

Address	Register	Bit							
		7	6	5	4	3	2	1	0
$FFFFF114_H$	WT0UVIC	WT0UVIF	WT0UVMK	0	0	0	WT0UVPR2	WT0UVPR1	WT0UVPR0
$FFFFF116_H$	WT1UVIC	WT1UVIF	WT1UVMK	0	0	0	WT1UVPR2	WT1UVPR1	WT1UVPR0
$FFFFF11A_H$	TM01IC	TM01IF	TM01MK	0	0	0	TM01PR2	TM01PR1	TM01PR0
$FFFFF11C_H$	P0IC	P0IF	P0MK	0	0	0	P0PR2	P0PR1	P0PR0
$FFFFF11E_H$	P1IC	P1IF	P1MK	0	0	0	P1PR2	P1PR1	P1PR0
$FFFFF120_H$	P2IC	P2IF	P2MK	0	0	0	P2PR2	P2PR1	P2PR0
$FFFFF122_H$	P3IC	P3IF	P3MK	0	0	0	P3PR2	P3PR1	P3PR0
$FFFFF12A_H$	TZ0UVIC	TZ0UVIF	TZ0UVMK	0	0	0	TZ0UVPR2	TZ0UVPR1	TZ0UVPR0
$FFFFF12C_H$	TZ1UVIC	TZ1UVIF	TZ1UVMK	0	0	0	TZ1UVPR2	TZ1UVPR1	TZ1UVPR0
$FFFFF12E_H$	TZ2UVIC	TZ2UVIF	TZ2UVMK	0	0	0	TZ2UVPR2	TZ2UVPR1	TZ2UVPR0
$FFFFF130_H$	TZ3UVIC	TZ3UVIF	TZ3UVMK	0	0	0	TZ3UVPR2	TZ3UVPR1	TZ3UVPR0
$FFFFF132_H$	TZ4UVIC	TZ4UVIF	TZ4UVMK	0	0	0	TZ4UVPR2	TZ4UVPR1	TZ4UVPR0
$FFFFF134_H$	TZ5UVIC	TZ5UVIF	TZ5UVMK	0	0	0	TZ5UVPR2	TZ5UVPR1	TZ5UVPR0
$FFFFF136_H$	TP0OVIC	TP0OVIF	TP0OVMK	0	0	0	TP0OVPR2	TP0OVPR1	TP0OVPR0
$FFFFF138_H$	TP0CC0IC	TP0CC0IF	TP0CC0MK	0	0	0	TP0CC0PR2	TP0CC0PR1	TP0CC0PR0
$FFFFF13A_H$	TP0CC1IC	TP0CC1IF	TP0CC1MK	0	0	0	TP0CC1PR2	TP0CC1PR1	TP0CC1PR0
$FFFFF14E_H$	TG0OV0IC	TG0OV0IF	TG0OV0MK	0	0	0	TG0OV0PR2	TG0OV0PR1	TG0OV0PR0
$FFFFF150_H$	TG0OV1IC	TG0OV1IF	TG0OV1MK	0	0	0	TG0OV1PR2	TG0OV1PR1	TG0OV1PR0
$FFFFF152_H$	TG0CC0IC	TG0CC0IF	TG0CC0MK	0	0	0	TG0CC0PR2	TG0CC0PR1	TG0CC0PR0
$FFFFF154_H$	TG0CC1IC	TG0CC1IF	TG0CC1MK	0	0	0	TG0CC1PR2	TG0CC1PR1	TG0CC1PR0
$FFFFF156_H$	TG0CC2IC	TG0CC2IF	TG0CC2MK	0	0	0	TG0CC2PR2	TG0CC2PR1	TG0CC2PR0
$FFFFF158_H$	TG0CC3IC	TG0CC3IF	TG0CC3MK	0	0	0	TG0CC3PR2	TG0CC3PR1	TG0CC3PR0
$FFFFF15A_H$	TG0CC4IC	TG0CC4IF	TG0CC4MK	0	0	0	TG0CC4PR2	TG0CC4PR1	TG0CC4PR0
$FFFFF15C_H$	TG0CC5IC	TG0CC5IF	TG0CC5MK	0	0	0	TG0CC5PR2	TG0CC5PR1	TG0CC5PR0
$FFFFF15E_H$	TG1OV0IC	TG1OV0IF	TG1OV0MK	0	0	0	TG1OV0PR2	TG1OV0PR1	TG1OV0PR0
$FFFFF160_H$	TG1OV1IC	TG1OV1IF	TG1OV1MK	0	0	0	TG1OV1PR2	TG1OV1PR1	TG1OV1PR0
$FFFFF162_H$	TG1CC0C	TG1CC0IF	TG1CC0MK	0	0	0	TG1CC0PR2	TG1CC0PR1	TG1CC0PR0
$FFFFF164_H$	TG1CC1IC	TG1CC1IF	TG1CC1MK	0	0	0	TG1CC1PR2	TG1CC1PR1	TG1CC1PR0
$FFFFF166_H$	TG1CC2IC	TG1CC2IF	TG1CC2MK	0	0	0	TG1CC2PR2	TG1CC2PR1	TG1CC2PR0
$FFFFF168_H$	TG1CC3IC	TG1CC3IF	TG1CC3MK	0	0	0	TG1CC3PR2	TG1CC3PR1	TG1CC3PR0
$FFFFF16A_H$	TG1CC4IC	TG1CC4IF	TG1CC4MK	0	0	0	TG1CC4PR2	TG1CC4PR1	TG1CC4PR0
$FFFFF16C_H$	TG1CC5IC	TG1CC5IF	TG1CC5MK	0	0	0	TG1CC5PR2	TG1CC5PR1	TG1CC5PR0
$FFFFF172_H$	ADIC	ADIF	ADMK	0	0	0	ADPR2	ADPR1	ADPR0
$FFFFF174_H$	C0ERRIC	C0ERRIF	C0ERRMK	0	0	0	C0ERRPR2	C0ERRPR1	C0ERRPR0
$FFFFF176_H$	C0WUPIC	C0WUPIF	C0WUPMK	0	0	0	C0WUPPR2	C0WUPPR1	C0WUPPR0
$FFFFF178_H$	C0RECIC	C0RECIF	C0RECMK	0	0	0	C0RECPR2	C0RECPR1	C0RECPR0
$FFFFF17A_H$	C0TRXIC	C0TRXIF	C0TRXMK	0	0	0	C0TRXPR2	C0TRXPR1	C0TRXPR0
$FFFFF17C_H$	CB0REIC	CB0REIF	CB0REMK	0	0	0	CB0REPR2	CB0REPR1	CB0REPR0
$FFFFF17E_H$	CB0RIC	CB0RIF	CB0RMK	0	0	0	CB0RPR2	CB0RPR1	CB0RPR0
$FFFFF180_H$	CB0TIC	CB0TIF	CB0TMK	0	0	0	CB0TPR2	CB0TPR1	CB0TPR0

Table 6-3 Addresses and bits of interrupt control registers (2/2)

Address	Register	Bit							
		7	6	5	4	3	2	1	0
FFFFF182$_H$	UA0REIC	UA0REIF	UA0REMK	0	0	0	UA0REPR2	UA0REPR1	UA0REPR0
FFFFF184$_H$	UA0RIC	UA0RIF	UA0RMK	0	0	0	UA0RPR2	UA0RPR1	UA0RPR0
FFFFF186$_H$	UA0TIC	UA0TIF	UA0TMK	0	0	0	UA0TPR2	UA0TPR1	UA0TPR0
FFFFF188$_H$	UA1REIC	UA1REIF	UA1REMK	0	0	0	UA1REPR2	UA1REPR1	UA1REPR0
FFFFF18A$_H$	UA1RIC	UA1RIF	UA1RMK	0	0	0	UA1RPR2	UA1RPR1	UA1RPR0
FFFFF18C$_H$	UA1TIC	UA1TIF	UA1TMK	0	0	0	UA1TPR2	UA1TPR1	UA1TPR0
FFFFF18E$_H$	IIC0IC	IIC0IF	IIC0MK	0	0	0	IIC0PR2	IIC0PR1	IIC0PR0
FFFFF19C$_H$	INT70IC	INT70IF	INT70MK	0	0	0	INT70PR2	INT70PR1	INT70PR0
FFFFF19E$_H$	INT71IC	INT71IF	INT71MK	0	0	0	INT71PR2	INT71PR1	INT71PR0
FFFFF1C2$_H$	CB1REIC	CB1REIF	CB1REMK	0	0	0	CB1REPR2	CB1REPR1	CB1REPR0
FFFFF1C4$_H$	CB1RIC	CB1RIF	CB1RMK	0	0	0	CB1RPR2	CB1RPR1	CB1RPR0
FFFFF1C6$_H$	CB1TIC	CB1TIF	CB1TMK	0	0	0	CB1TPR2	CB1TPR1	CB1TPR0

6.3.5 IMR0 to IMR5 - Interrupt mask registers

These registers set the interrupt mask state for the maskable interrupts.

The xxMK bit of the IMRm (m = 0 to 5) registers is equivalent to the xxMK bit of the xxIC register.

Caution IMask bits without function, indicated with “1”, must not be altered. Make sure to set them “1” when writing to the register.

Access These registers can be read/written in 16-bit and 8-bit units.

The address of the lower 8-bit register IMRmL is equal to that of the 16-bit IMRm register, and the higher 8-bit register IMRmH can be accessed on the following address (address (IMRm) + 1).

Address

IMR0, IMR0L:	FFFF F100$_H$	IMR0H:	FFFF F101$_H$
IMR1, IMR1L:	FFFF F102$_H$	IMR1H:	FFFF F103$_H$
IMR2, IMR2L:	FFFF F104$_H$	IMR2H:	FFFF F105$_H$
IMR3, IMR3L:	FFFF F106$_H$	IMR3H:	FFFF F107$_H$
IMR4, IMR4L:	FFFF F108$_H$	IMR4H:	FFFF F109$_H$
IMR5, IMR5L:	FFFF F10A$_H$	IMR5H:	FFFF F10B$_H$

Initial Value FFFF$_H$

IMR0

15	14	13	12	11	10	9	8
TZ2UVMK	TZ1UVMK	TZ0UVMK	1	1	1	P3MK	P2MK
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

7	6	5	4	3	2	1	0
P1MK	P0MK	TM01MK	1	WT1UVMK	WT0UVMK	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

IMR1

7	6	5	4	3	2	1	0
TG0OV0MK	1	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

7	6	5	4	3	2	1	0
1	1	TP0CC1MK	TP0CC0MK	TP0OVMK	TZ5UVMK	TZ4UVMK	TZ3UVMK
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

IMR2

7	6	5	4	3	2	1	0
1	TG1CC5MK	TG1CC4MK	TG1CC3MK	TG1CC2MK	TG1CC1MK	TG1CC0MK	TG1OV1MK
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

7	6	5	4	3	2	1	0
TG1OV0MK	TG0CC5MK	TG0CC4MK	TG0CC3MK	TG0CC2MK	TG0CC1MK	TG0CC0MK	TG0OV1MK
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

IMR3

7	6	5	4	3	2	1	0
IIC0MK	UA1TMK	UA1RMK	UA1REMK	UA0TMK	UA0RMK	UA0REMK	CB0TMK
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

7	6	5	4	3	2	1	0
CB0RMK	CB0REMK	C0TRXMK	C0RECMK	C0WUPMK	C0ERRMK	ADMK	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

IMR4

7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

7	6	5	4	3	2	1	0
INT71MK	INT70MK	1	1	1	1	SG0MK	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

IMR5

7	6	5	4	3	2	1	0
1	1	1	1	CB1TMK	CB1RMK	CB1REMK	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit position	Bit name	Function
15 to 0	xxMK	Interrupt mask flag. 0: Interrupt servicing enabled 1: Interrupt servicing disabled (pending)

Note xx: identification name of each peripheral unit (WT0UV-WT1UV, TM01, P0-P3, TZ0UV-TZ3UV, TP0OV, TP0CC0, TP0CC1, TG0OV0-TG1OV0, TG0OV1-TG1OV1, TG0CC0-TG1CC0, TG0CC1-TG1CC1, TG0CC2-TG1CC2, TG0CC3-TG1CC3, TG0CC4-TG1CC4, TG0CC5-TG1CC5, AD, C0ERR, C0WUP, C0REC, C0TRX, CB0RE-CB1RE, CB0R-CB1R, CB0T-CB1T, UA0RE-UA1RE, UA0R-UA1R, UA0T-UA1T, IIC0, INT70, INT71)

6.3.6 ISPR - In-service priority register

This register holds the priority level of the maskable interrupt currently acknowledged. When an interrupt request is acknowledged, the bit of this register corresponding to the priority level of that interrupt request is set to 1 and remains set while the interrupt is serviced.

When the RETI instruction is executed, the bit corresponding to the interrupt request having the highest priority is automatically reset to 0 by hardware. However, it is not reset to 0 when execution is returned from non-maskable interrupt servicing or exception processing.

This register is read-only in 8-bit or 1-bit units.

This register can be read/written in 8-bit or 1-bit units.

Address FFFF F19A$_H$

Initial Value 00$_H$

7	6	5	4	3	2	1	0
ISPR7	ISPR6	ISPR5	ISPR4	ISPR3	ISPR2	ISPR1	ISPR0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit position	Bit name	Function
7 to 0	ISPR7 to ISPR0	Indicates priority of interrupt currently acknowledged 0: Interrupt request with priority n not acknowledged 1: Interrupt request with priority n acknowledged

Note n = 0 to 7 (priority level)

6.3.7 Maskable interrupt status flag (ID)

The ID flag is bit 5 of the PSW and this controls the maskable interrupt's operating state, and stores control information regarding enabling or disabling of interrupt requests.

Initial Value 0000 0020$_H$. The program status is initialized by any reset.

31	8	7	6	5	4	3	2	1	0
fixed to 0		NP	EP	ID	SAT	CY	OV	S	Z
R	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit position	Bit name	Function
5	ID	Indicates whether maskable interrupt processing is enabled or disabled. 0: Maskable interrupt request acknowledgement enabled 1: Maskable interrupt request acknowledgement disabled (pending) This bit is set to 1 by the DI instruction and reset to 0 by the EI instruction. Its value is also modified by the RETI instruction or LDSR instruction when referencing to PSW. Non-maskable interrupt requests and exceptions are acknowledged regardless of this flag. when a maskable interrupt is acknowledged, the ID flag is automatically set to 1 by hardware. The interrupt request generated during the acknowledgement disabled period (ID = 1) is acknowledged when the PIFn bit of PICn register is set to 1, and the ID flag is reset to 0.

6.3.8 External maskable interrupts

This microcontroller provides maskable external interrupts INTPn with the following features:

- Analog input filter (refer to *“Analog filtered inputs“ on page 97*)
- Interrupt detection selectable for each interrupt input:
  - Rising edge
  - Falling edge
  - Both edges: rising and falling edge
  - High level
  - Low level
- Wakeup capability from stand-by mode of INTPn upon
  - Rising edge
  - Falling edge
  - Both edges: rising and falling edge

For configuration of the external interrupt events refer to *“Edge and Level Detection Configuration“ on page 204*.

6.3.9 Software interrupts

This microcontroller provides maskable software interrupts to for processing of an interrupt service routine by the application software.

For initiating a software interrupt the interrupt request flag xxIC.xxIF of the concerned software interrupt “xx” must be set to 1. The following processing is identical to that of all other maskable interrupts.

6.4 Edge and Level Detection Configuration

The microcontroller provides the maskable external interrupts INTPn and one non-maskable interrupt (NMI).

INTPn can be configured to generate interrupts upon edges or levels, the NMI can be set up to react on edges.

(1) INTM0 to INTM1 - External interrupt configuration register

External interrupt function is configured by the registers INTM0…INTM1.

Access These registers can be read/written in 8-bit and 1-bit units.

Address	INTM0:	FFFF $F700_H$
	INTM1:	FFFF $F702_H$
	INTM2:	FFFF $F704_H$
	INTM3:	FFFF $F706_H$

Initial Value 00_H

	7	6	5	4	3	2	1	0
INTM0	0	ELSEL1	ESEL11	ESEL10	NMIEN	ELSEL0	ESEL01	ESEL00
	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W

	7	6	5	4	3	2	1	0
INTM1	0	ELSEL3	ESEL31	ESEL30	0	ELSEL2	ESEL21	ESEL20
	R	R/W	R/W	R/W	R	R/W	R/W	R/W

	7	6	5	4	3	2	1	0
INTM2	0	ELSEL5	ESEL51	ESEL50	0	ELSEL4	ESEL41	ESEL40
	R	R/W	R/W	R/W	R	R/W	R/W	R/W

	7	6	5	4	3	2	1	0
INTM3	0	ELSEL7	ESEL71	ESEL70	0	ELSEL6	ESEL61	ESEL60
	R	R/W	R/W	R/W	R	R/W	R/W	R/W

The register bits ELSELn, ESELn1 and ESELn0 configure the INTPn interrupt function:

ELSELn	ESELn1	ESELn0	Function
0	0	0	falling edge
0	0	1	rising edge
0	1	0	prohibited to use
0	1	1	falling and rising edge
1	0	0	low level detection
1	0	1	high level detection
1	1	0	low level detection
1	1	1	high level detection

The NMI and INTP0 share the same pin. The register bits NMIEN, ESEL0, ESEL01 and ESEL00 configure the NMI and INTP0 interrupt function:

NMIEN	ESEL0	ESEL01	ESEL00	Function	
				NMI	INTP0
0	0	0	0	masked	falling edge
	0	0	1		rising edge
	0	1	0		prohibited
	0	1	1		both edges
	1	0	0		low level
	1	0	1		high level
	1	1	0		low level
	1	1	1		high level
1	0	0	0	falling edge	falling edge
	0	0	1	rising edge	rising edge
	0	1	0	prohibited	prohibited
	0	1	1	both edges	both edges
	1	0	0	falling edge	low level
	1	0	1	rising edge	high level
	1	1	0	prohibited	low level
	1	1	1	both edges	high level

Caution The NMI configuration bits INTM0.NMIEN and INTM0.ESEL0[1:0] can only be changed if INTM0.NMIEN = 0.
Due to INTM0.NMIEN = 0 after reset the NMI function is disabled and must be enabled by the application software. Once enabled, the NMI function cannot be disabled by software.
Specify INTM0.ESEL0[1:0] before or at the same time with setting INTM0.NMIEN = 1.

Note that INTM0.ESEL0 can be written independently of INTM0.NMIEN.

6.5 Software Exception

A software exception is generated when the CPU executes the TRAP instruction, and can be always acknowledged.

6.5.1 Operation

If a software exception occurs, the CPU performs the following processing, and transfers control to the handler routine:

(1) Saves the restored PC to EIPC.

(2) Saves the current PSW to EIPSW.

(3) Writes an exception code to the lower 16 bits (EICC) of ECR (interrupt source).

(4) Sets the EP and ID bits of the PSW.

(5) Sets the handler address (00000040_H or 00000050_H) corresponding to the software exception to the PC, and transfers control.

Figure 6-10 illustrates the processing of a software exception.

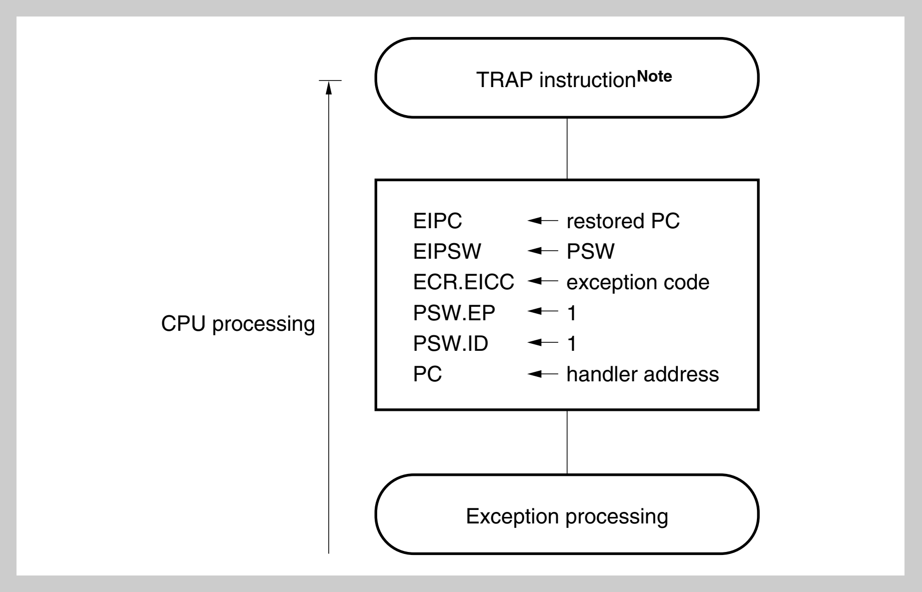


Figure 6-10 Software exception processing

Note TRAP Instruction Format: TRAP vector (the vector is a value from 0 to $1F_H$.)

The handler address is determined by the TRAP instruction’s operand (vector). If the vector is 0 to $0F_H$, it becomes 00000040_H, and if the vector is 10_H to $1F_H$, it becomes 00000050_H.

6.5.2 Restore

Recovery from software exception processing is carried out by the RETI instruction.

By executing the RETI instruction, the CPU carries out the following processing and shifts control to the restored PC's address.

(1) Loads the restored PC and PSW from EIPC and EIPSW because the EP bit of the PSW is 1.

(2) Transfers control to the address of the restored PC and PSW.

Figure 6-11 illustrates the processing of the RETI instruction.

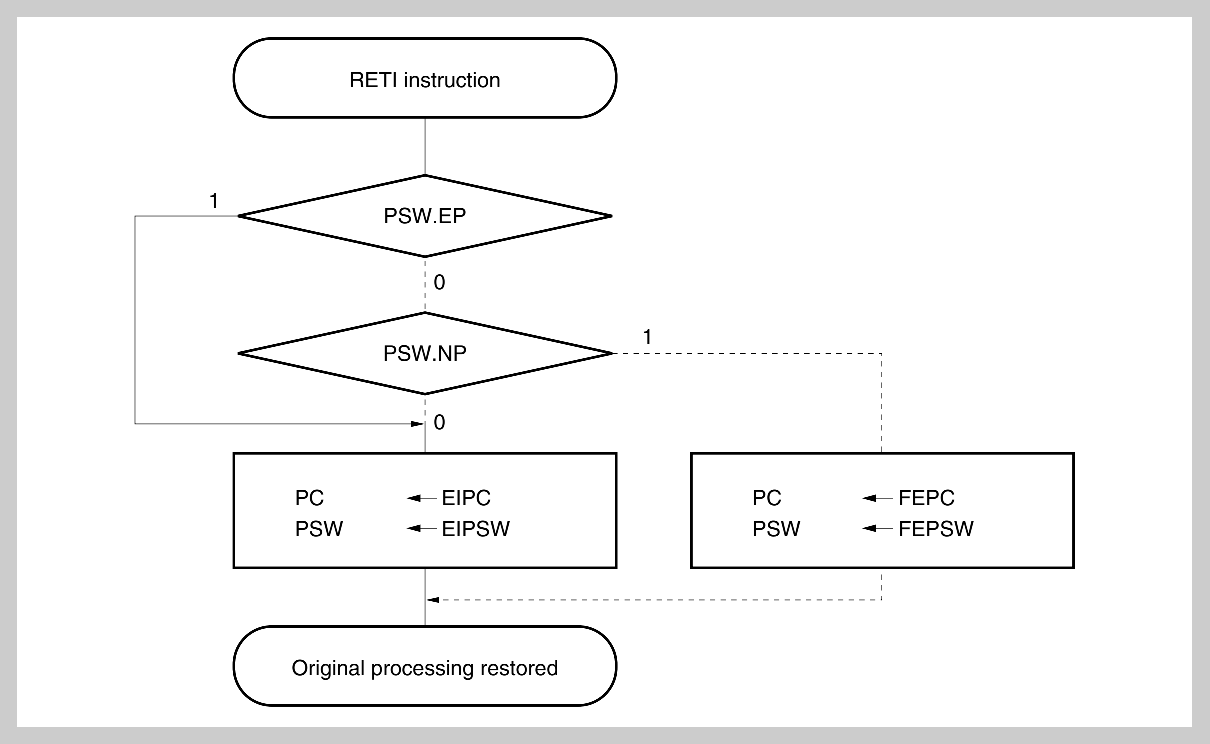


Figure 6-11 RETI instruction processing

Caution When the PSW.EP bit and the PSW.NP bit are changed by the LDSR instruction during the software exception processing, in order to restore the PC and PSW correctly during recovery by the RETI instruction, it is necessary to set PSW.EP back to 1 using the LDSR instruction immediately before the RETI instruction.

Note The solid lines show the CPU processing flow.

6.5.3 Exception status flag (EP)

The EP flag is bit 6 of PSW, and is a status flag used to indicate that exception processing is in progress. It is set when an exception occurs.

Initial Value 0000 0020$_H$. The program status is initialized by any reset.

31	8	7	6	5	4	3	2	1	0
fixed to 0		NP	EP	ID	SAT	CY	OV	S	Z
R	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit position	Bit name	Function
6	EP	Shows that exception processing is in progress. 0: Exception processing not in progress. 1: Exception processing in progress.

6.6 Exception Trap

An exception trap is an interrupt that is requested when an illegal execution of an instruction takes place. For this microcontroller, an illegal opcode exception (ILGOP: Illegal Opcode Trap) is considered as an exception trap.

6.6.1 Illegal opcode definition

The illegal instruction has an opcode (bits 10 to 5) of 111111_B, a sub-opcode (bits 23 to 26) of 0111_B to 1111_B, and a sub-opcode (bit 16) of 0_B. An exception trap is generated when an instruction applicable to this illegal instruction is executed.

31 27	26 23	22	16	15 11	10 5	4 0
× × × × ×	0 1 1 1 to 1 1 1 1	× × × × × ×	0	× × × × ×	1 1 1 1 1 1	× × × × ×

Note ×: Arbitrary

(1) Operation

If an exception trap occurs, the CPU performs the following processing, and transfers control to the handler routine:

(1) Saves the restored PC to DBPC.

(2) Saves the current PSW to DBPSW.

(3) Sets the NP, EP, and ID bits of the PSW.

(4) Sets the handler address (00000060_H) corresponding to the exception trap to the PC, and transfers control.

Figure 6-12 illustrates the processing of the exception trap.

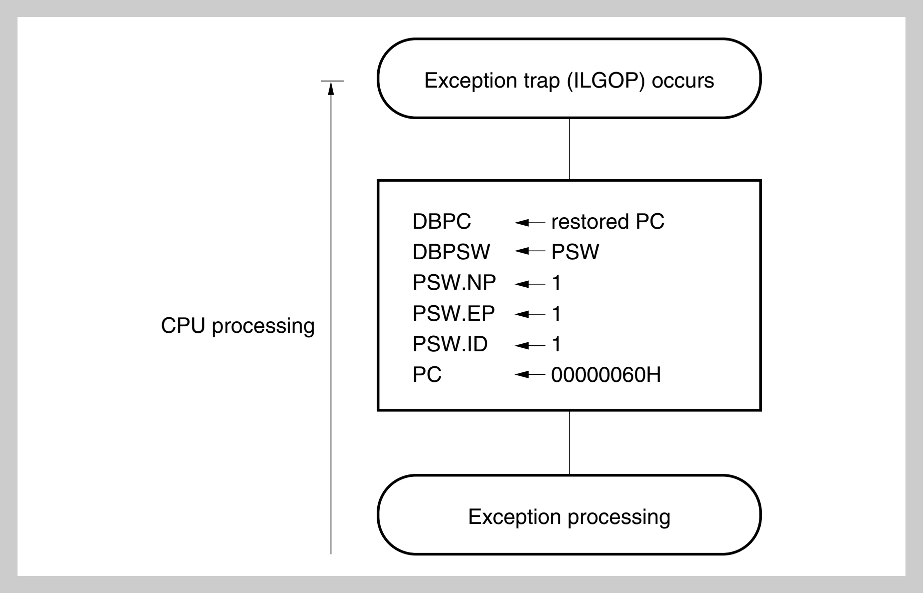


Figure 6-12 Exception trap processing

(2) Restore

Recovery from an exception trap is carried out by the DBRET instruction. By executing the DBRET instruction, the CPU carries out the following processing and controls the address of the restored PC.

(1) Loads the restored PC and PSW from DBPC and DBPSW.

(2) Transfers control to the address indicated by the restored PC and PSW.

Figure 6-13 illustrates the restore processing from an exception trap.

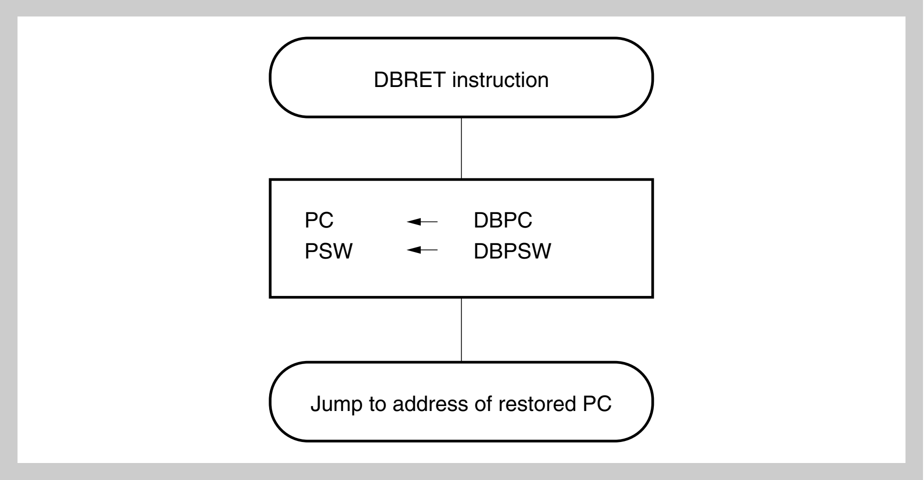


Figure 6-13 Restore processing from exception trap

6.6.2 Debug trap

The debug trap is an exception that can be acknowledged every time and is generated by execution of the DBTRAP instruction.

When the debug trap is generated, the CPU performs the following processing.

(1) Operation

When the debug trap is generated, the CPU performs the following processing, transfers control to the debug monitor routine, and shifts to debug mode.

(1) Saves the restored PC to DBPC.

(2) Saves the current PSW to DBPSW.

(3) Sets the NP, EP and ID bits of the PSW.

(4) Sets the handler address (00000060_H) corresponding to the debug trap to the PC and transfers control.

Figure 6-14 illustrates the processing of the debug trap.

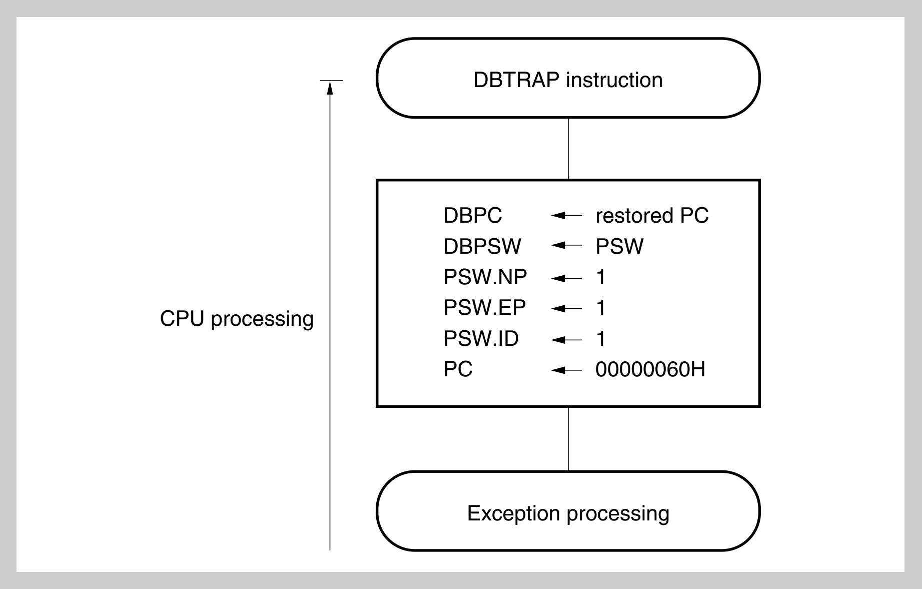


Figure 6-14 Debug trap processing

(2) Restore

Recovery from a debug trap is carried out by the DBRET instruction. By executing the DBRET instruction, the CPU carries out the following processing and controls the address of the restored PC.

(1) Loads the restored PC and PSW from DBPC and DBPSW.

(2) Transfers control to the address indicated by the restored PC and PSW.

Figure 6-15 illustrates the restore processing from a debug trap.

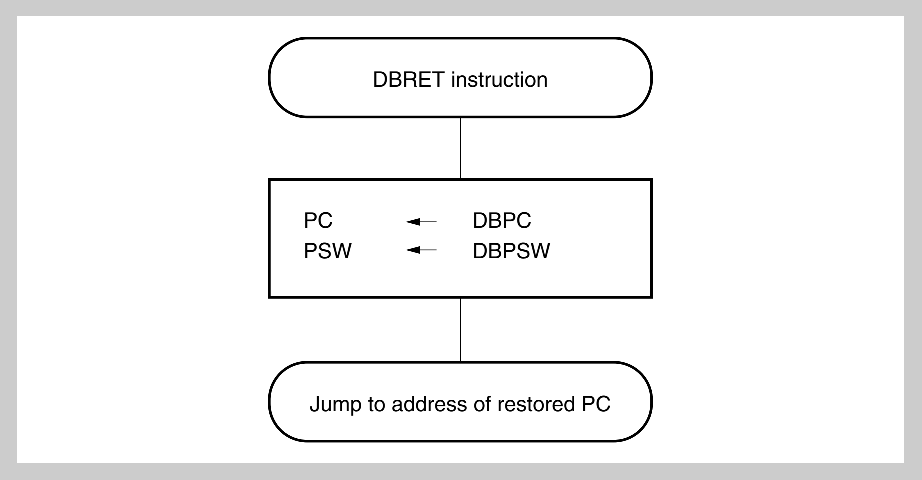


Figure 6-15 Restore processing from debug trap

6.7 Multiple Interrupt Processing Control

Multiple interrupt processing control is a process by which an interrupt request that is currently being processed can be interrupted during processing if there is an interrupt request with a higher priority level, and the higher priority interrupt request is received and processed first.

If there is an interrupt request with a lower priority level than the interrupt request currently being processed, that interrupt request is held pending.

Maskable interrupt multiple processing control is executed when an interrupt has an enable status (ID = 0). Thus, if multiple interrupts are executed, it is necessary to have an interrupt enable status (ID = 0) even for an interrupt processing routine.

If a maskable interrupt enable or a software exception is generated in a maskable interrupt or software exception service program, it is necessary to save EIPC and EIPSW.

This is accomplished by the following procedure.

(1) Acknowledgment of maskable interrupts in service program

Service program of maskable interrupt or exception

...	
...	
• EIPC saved to memory or register	
• EIPSW saved to memory or register	
• EI instruction (interrupt acknowledgment enabled)	
...	
	...
	Higher priority maskable interrupt acknowledgment
	...
...	
• DI instruction (interrupt acknowledgment disabled)	
• Saved value restored to EIPSW	
• Saved value restored to EIPC	
• RETI instruction	

(2) Generation of exception in service program

Service program of maskable interrupt or exception

...	
...	
• EIPC saved to memory or register	
• EIPSW saved to memory or register	
...	
• TRAP instruction	
	...
	TRAP/exception acknowledgment
	...
...	
• Saved value restored to EIPSW	
• Saved value restored to EIPC	
• RETI instruction	

The priority order for multiple interrupt processing control has 8 levels, from 0 to 7 for each maskable interrupt request (0 is the highest priority), but it can be set as desired via software. Setting of the priority order level is done using the PPRn0 to PPRn2 bits of the interrupt control request register (PICn), which is provided for each maskable interrupt request. After system reset, an interrupt request is masked by the PMKn bit and the priority order is set to level 7 by the PPRn0 to PPRn2 bits.

The priority order of maskable interrupts is as follows.

(High) Level 0 > Level 1 > Level 2 > Level 3 > Level 4 >
Level 5 > Level 6 > Level 7 (Low)

Interrupt processing that has been suspended as a result of multiple processing control is resumed after the processing of the higher priority interrupt has been completed and the RETI instruction has been executed.

A pending interrupt request is acknowledged after the current interrupt processing has been completed and the RETI instruction has been executed.

Caution In a non-maskable interrupt processing routine (time until the RETI instruction is executed), maskable interrupts are suspended and not acknowledged.

6.8 Interrupt Response Time

The following table describes the interrupt response time (from interrupt generation to start of interrupt processing).

Except in the following cases, the interrupt response time is a minimum of 5 clocks.

- During software or hardware STOP mode
- When an external bus is accessed
- When there are two or more successive interrupt request non-sampling instructions (see *"Periods in Which Interrupts Are Not Acknowledged" on page 216*).
- When the interrupt control register is accessed

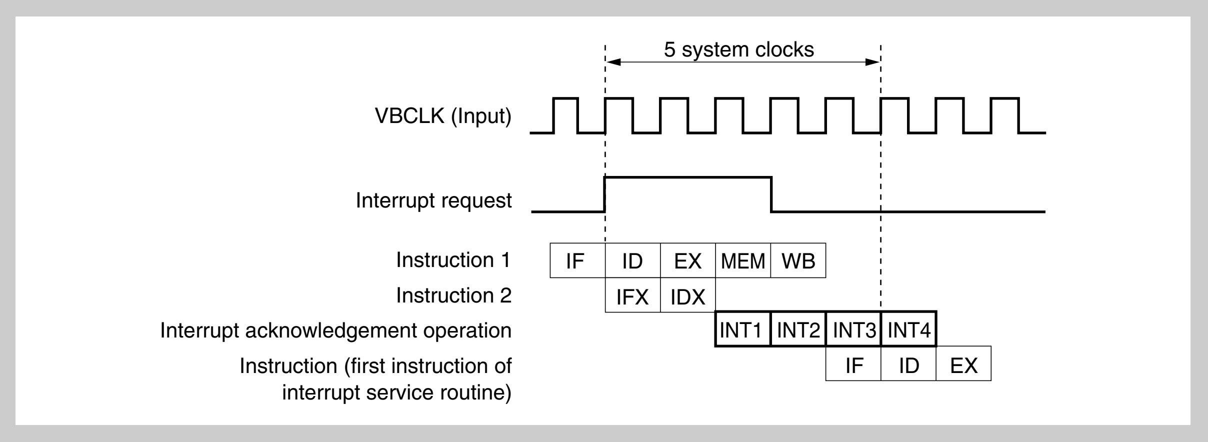


Figure 6-16 Pipeline operation at interrupt request acknowledgment (outline)

Note INT1 to INT4: Interrupt acknowledgement processing
IFx: Invalid instruction fetch
IDx: Invalid instruction decode

Note If the same interrupt occures during the interrupt acknowledge time of 5 cycles, this new interrupt will discarded. The next interrupt of the same source will only be registered after these 5 cycles.

Table 6-4 Interrupt response time

Interrupt response time (internal system clocks)			Condition
	Internal interrupt	External interrupt	
Minimum	5	5 + analog delay time	The following cases are exceptions: • In IDLE/software STOP mode • External bit access • Two or more interrupt request non-sample instructions are executed • Access to interrupt control register
Maximum	11	11 + analog delay time	

6.9 Periods in Which Interrupts Are Not Acknowledged

An interrupt is acknowledged while an instruction is being executed. However, no interrupt will be acknowledged between an interrupt non-sample instruction and the next instruction.

The interrupt request non-sampling instructions are as follows:

- EI instruction
- DI instruction
- LDSR reg2, 0x5 instruction (for PSW)
- The store instruction for the maskable interrupt control registers (xxIC), in-service priority register (ISPR), and command register (PRCMD).

Chapter 7 Bus Control Unit (BCU)

The Bus Control Unit provides access to the on-chip peripheral input/output modules and controls this access.

7.1 Overview

The figure below shows a block diagram of the modules that are necessary for accessing the on-chip peripherals.

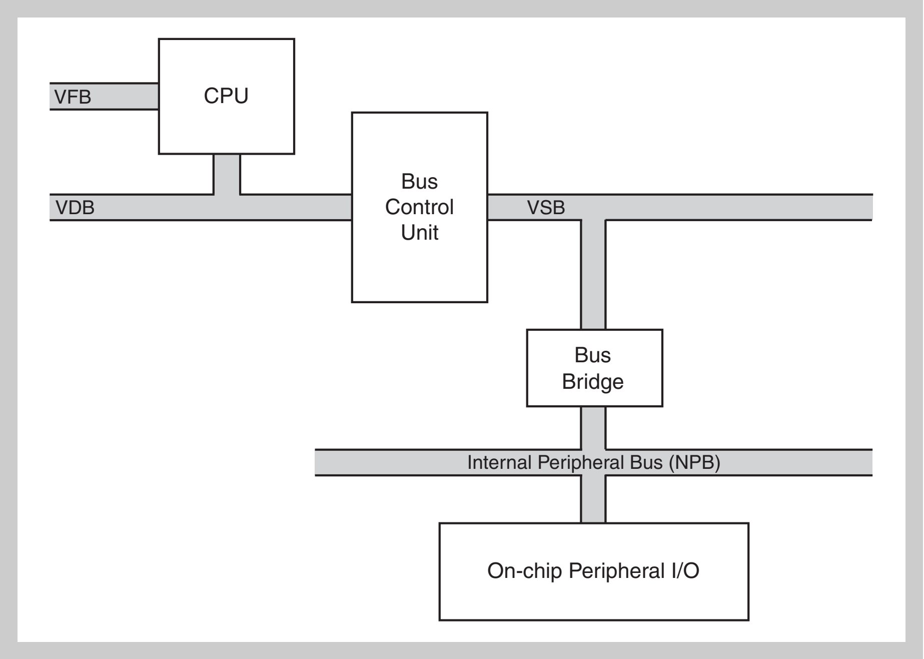


Figure 7-1 Bus Control unit block diagram

Busses The busses are abbreviated as follows:

- NPB: Peripheral bus
- VSB: V850 system bus
- VDB: V850 data bus
- VFB: V850 fetch bus

BCU The Bus Control Unit (BCU) controls the CPU access to the VSB, and therefore via the bus bridge also to the on-chip peripherals.

The BCU holds configuration registers to

- determine the 16 KB of the programmable peripheral I/O area
- set up the proper timing on the NPB

7.2 Peripheral I/O area

Two areas are reserved for the registers of the on-chip peripheral functions. These areas are called “peripheral I/O areas”:

Table 7-1 Peripheral I/O areas

Name	Address range	Size
Fixed peripheral I/O area	03FF F000$_H$ to 03FF FFFF$_H$	4 KB
Programmable peripheral I/O area (PPA)	Can be allocated at arbitrary addresses. Base address is defined in the BPC register.	16 KB

7.2.1 Fixed peripheral I/O area

The fixed peripheral I/O area holds the registers of the on-chip peripheral I/O functions.

Note Because the address space covers 64 MB, the address bits A[31:26] are not considered. Therefore, in this manual, all addresses of peripheral I/O registers in the 4 KB peripheral I/O area are given in the range FFFF F000$_H$ to FFFF FFFF$_H$ instead of 03FF F000$_H$ to 03FF FFFF$_H$.

7.2.2 Programmable peripheral I/O area (PPA)

The usage and the address range of the PPA is configurable. The PPA extends the fixed peripheral I/O area and assigns an additional 12 KB address space for accessing on-chip peripherals.

The figure below illustrates the programmable peripheral I/O area (PPA).

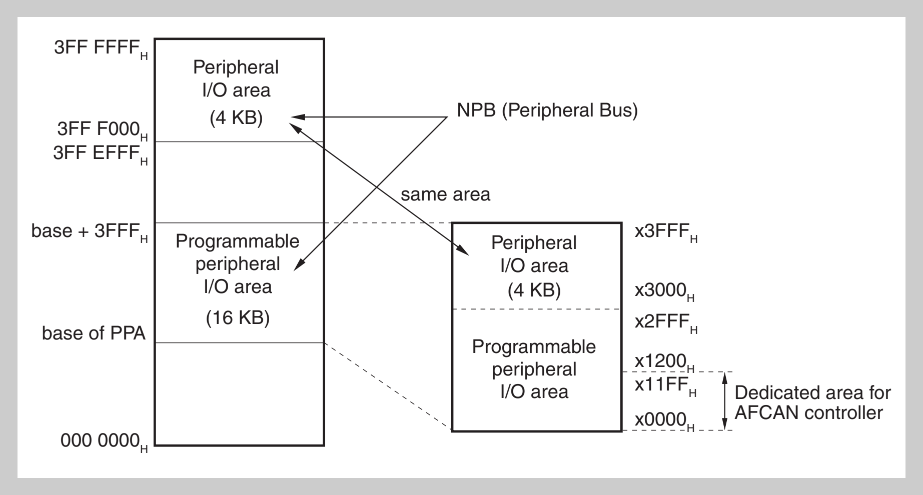

3FF FFFF$_H$
Peripheral I/O area (4 KB)
NPB (Peripheral Bus)
3FF F000$_H$
3FF EFFF$_H$
same area
base + 3FFF$_H$
Programmable peripheral I/O area (16 KB)
base of PPA
000 0000$_H$
Peripheral I/O area (4 KB)
x3FFF$_H$
x3000$_H$
x2FFF$_H$
Programmable peripheral I/O area
x1200$_H$
x11FF$_H$
x0000$_H$
Dedicated area for AFCAN controller

Figure 7-2 Programmable peripheral I/O area

The CAN modules registers and message buffers are allocated to the PPA. Refer to *“CAN module register and message buffer addresses“ on page 572* for information how the calculate the register and message buffer addresses of the CAN modules.

Caution If the programmable peripheral I/O area overlaps one of the following areas, the programmable peripheral I/O area becomes ineffective:

- Peripheral I/O area
- ROM/flash memory area
- RAM area

Note

1. The *fixed* peripheral I/O area is mirrored to the upper 4 KB of the *programmable* peripheral I/O area—regardless of the base address of the PPA. If data is written in one area, data having the same contents is also written in the other area.
2. All address definitions in this manual that refer to the programmable peripheral area assume that the base address of the PPA is 03FE C000$_H$, that means BPC = 8FFB$_H$.

7.2.3 NPB access timing

All accesses to the peripheral I/O areas are passed over to the NPB bus via the VSB - NPB bus bridge BBR. Read and write access times to registers via the NPB depend on the register (refer to *“Registers Access Times“ on page 767*), the system clock VBCLK and the setting of the VSWC register.

The CPU operation during an access to a register via the NPB depends also on the kind of peripheral I/O area:

- Fixed peripheral I/O area

  During a read or write access the CPU operation stops until the access via the NPB is completed.

- Programmable peripheral I/O area

  During a read access the CPU operation stops until the read access via the NPB is completed.

  During a write access the CPU operation continues operation, provided any preceded NPB access is already finished. If a preceded NPB access is still ongoing the CPU stops until this access is finished and the NPB is cleared.

Caution Pay attention at write accesses to NPB peripheral I/O registers via the programmable peripheral I/O area.

Since the CPU may continue operation, even though the data has not yet been transferred to its destination register, inconsistencies may occur between the program flow and the status of the registers.

In particular register set-ups which change an operational status of a certain module require special notice, like, for instance, masking/unmasking of interrupts via maskable interrupt control registers xxIC, enabling/disabling timers, etc.

7.3 Boundary operation conditions

The microcontroller device has the following boundary operation conditions:

(1) Program space

Instruction fetches from the internal peripheral I/O area are inhibited and yield NOP operations.

If a branch instruction exists at the upper limit of the internal RAM area, a prefetch operation (invalid fetch) that straddles over the internal peripheral I/O area does not occur.

(2) Data space

The microcontroller device is provided with an address misalign function.

By this function, data of any format (word: 32 bit, halfword: 16 bit, byte: 8 bit) can be placed to any address in memory, even though the address is not aligned to the data format (that means address 4n for words, address 2n for halfwords).

- Unaligned halfword data access
  When the LSB of the address is A0 =1, two byte accesses are performed.
- Unaligned word data access
  When the LSB of the address is A0 =1, two byte and one halfword accesses are performed. In total it takes 3 bus cycles.
  - When the LSBs of the address are A[1:0] =10_B, two halfword accesses are performed.

Note Accessing data on misaligned addresses takes more than one bus cycle to complete data read/write. Consequently, the bus efficiency will drop.

7.4 BCU Registers

Access to on-chip peripherals is controlled and operated by registers of the Bus Control Unit (BCU):

Table 7-2 Bus Control Unit register overview

Register name	Shortcut	Address
Peripheral area selection control register	BPC	FFFF $F064_H$
Internal peripheral function wait control register	VSWC	FFFF $F06E_H$

(1) BPC - Peripheral area selection control register

The 16-bit BPC register defines whether the programmable peripheral I/O area (PPA) is used or not and determines the starting address of the PPA.

Access This register can be read/written in 16-bit units.

Address FFFF F064$_H$

Initial Value 0000$_H$

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PA15	0	PA13	PA12	PA11	PA10	PA9	PA8	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0

Table 7-3 BPC register contents

Bit Position	Bit Name	Function
15	PA15	Select usage of programmable peripheral I/O area (PPA). 0: PPA disabled 1: PPA enabled
13 to 0	PA[13:0]	Bits PA[13:0] specify bits 27 to 14 of the starting address of the PPA. The other bits of the address are fixed to 0.

Caution Bit 14 must always be 0.
The base address PBA of the programmable peripheral area sets the start address of the 16 KB PPA in a range of 256 MB. The 256 MB page is mirrored 16 times to the entire 32-bit address range.

The base address PBA is calculated by

$$PBA = BPC.PA[13:0] \times 2^{14}$$

Table 7-4 shows how the base address PBA of the programmable peripheral area is assembled

Table 7-4 Address range of programmable peripheral area (16 KB)

31	…	28	27 … 14	13	…	1	0	bit
0	…	0	BPC.PA[13:0]	1	…	1	1	
	…		…		…			
0	…	0	BPC.PA[13:0]	0	…	0	1	
0	…	0	BPC.PA[13:0]	0	…	0	0	PBA

Note The recommended setting for the BPC register is 8FFB$_H$. With this configuration the programmable peripheral area is mapped to the address range 03FE C000$_H$ to 03FE FFFF$_H$. With this setting the CAN message buffer registers are accessible via the addresses given in *“CAN Controller (CAN)“ on page 546*.
The fixed peripheral area is mirrored to the address range 03FE E000$_H$ to 03FE FFFF$_H$.

(2) VSWC - Peripheral function wait control register

The 8-bit VSWC register defines the wait states inserted when accessing peripheral special function registers via the internal bus. Both address setup and data wait states are based on the system clock.

Access This register can be read/written in 8-bit or 1-bit units.

Address FFFF F06E$_H$

Initial Value 77$_H$

7	6	5	4	3	2	1	0
0	SUWL2	SUWL1	SUWL0	0	VSWL2	VSWL1	VSWL0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 7-5 VSWC register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>6 to 4</td><td>SUWL[2:0]</td><td>Address setup wait for internal bus:
<table>
<tr><th>SUWL2</th><th>SUWL1</th><th>SUWL0</th><th>Number of address setup wait states</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1 CPU system clock (VBCLK)</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>2 CPU system clock (VBCLK)</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>3 CPU system clock (VBCLK)</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>4 CPU system clock (VBCLK)</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>5 CPU system clock (VBCLK)</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>6 CPU system clock (VBCLK)</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>7 CPU system clock (VBCLK)</td></tr>
</table></td></tr>
<tr><td>2 to 0</td><td>VSWL[2:0]</td><td>Data wait for internal bus:
<table>
<tr><th>VSWL2</th><th>VSWL1</th><th>VSWL0</th><th>Number of data wait states</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1 CPU system clock (VBCLK)</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>2 CPU system clock (VBCLK)</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>3 CPU system clock (VBCLK)</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>4 CPU system clock (VBCLK)</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>5 CPU system clock (VBCLK)</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>6 CPU system clock (VBCLK)</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>7 CPU system clock (VBCLK)</td></tr>
</table></td></tr>
</table>

Depending on the system clock f_{VBCLK} the setups according to the following table should be applied for VSWC.

Table 7-6 Recommended timing for internal bus

System clock[a] f_{VBCLK}	≤ 16 MHz	≤ 25 MHz
SUWL	0	0
VSWL	0	1
VSWC	00_H	01_H

[a] When deriving the system clock from the modulated clock of the SSCG, the maximum clock determines the correct register setting.

Chapter 8 ROM Correction Function (ROMC)

This microcontroller features following ROM correction facilities:

- “Data Replacement” ROM correction:
  - 1 x 6 channels for VFB flash memory and ROM

  The individual channels of each “Data Replacement” ROM correction are identified by “n” (n = 0 to 5)

Caution During self-programming make sure to disable all ROM correction facilities, as enabled ROM corrections may conflict with the internal firmware.

8.1 Overview

The ROM Correction Function is used to replace part of the internal flash memory with user defined data.

By using this function, program bugs found in the internal flash memory can be corrected.

The “Data Replacement” ROM correction unit allows to replace any data read from the internal flash memory by a user defined substitute. Thus instructions as well as constant data read from flash memory can be replaced.

8.2 “Data Replacement” ROM Correction Unit

8.2.1 Features

- 6 correction channels for VFB flash/ROM (n = 0 to 5)
- Programmable correction address for each channel
- Programmable correction value for each channel (the value can be an instructions as well as data)
- Correction of aligned and unaligned instructions/data
- Correction of halfwords and words
- Enable/disable of each channel individually by software

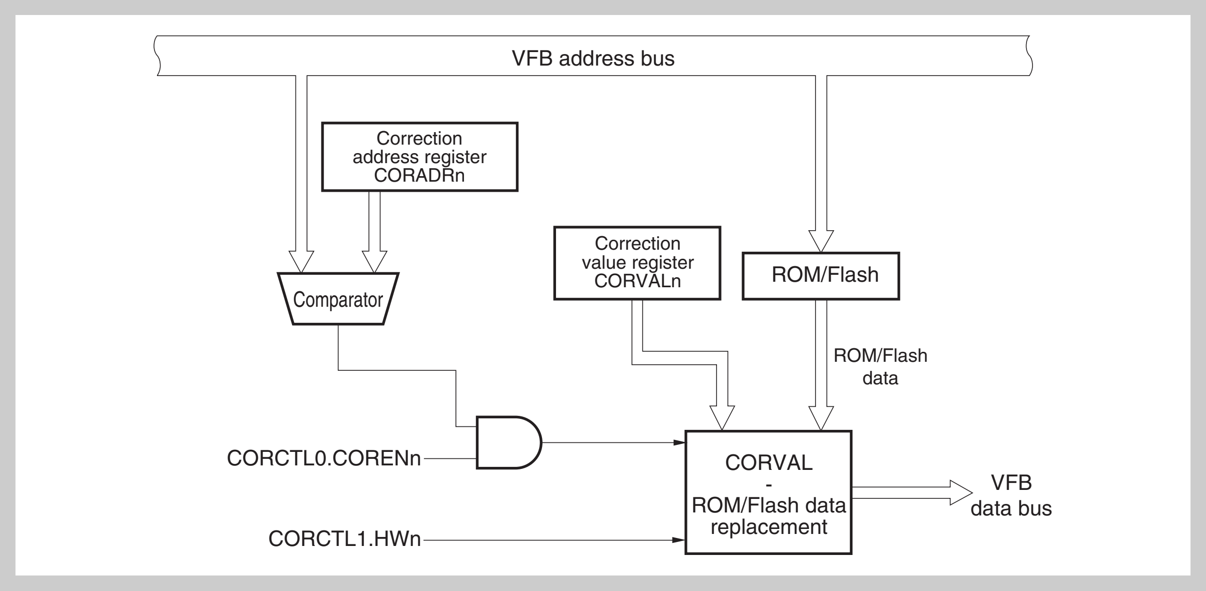

VFB address bus
Correction address register CORADRn
Comparator
Correction value register CORVALn
ROM/Flash
ROM/Flash data
CORCTL0.CORENn
CORCTL1.HWn
CORVAL - ROM/Flash data replacement
VFB data bus

Figure 8-1 “Data Replacement” ROM correction block diagram

8.2.2 “Data Replacement” ROM correction operation

The “Data Replacement” ROM correction unit compares the address on the V850 fetch bus (VFB) with the contents of the programmable correction address registers CORADRn. If an address matches, a programmable value (instructions or data) is put on the V850 fetch bus instead of the ROM contents. If no address matches, the ROM contents is passed on the fetch bus as normal.

The V850E architecture supports 16-bit as well as 32-bit instructions/data with support of aligned and unaligned instruction/data placement. *Figure 8-2* shows the different alignments of code/data inside the ROM.

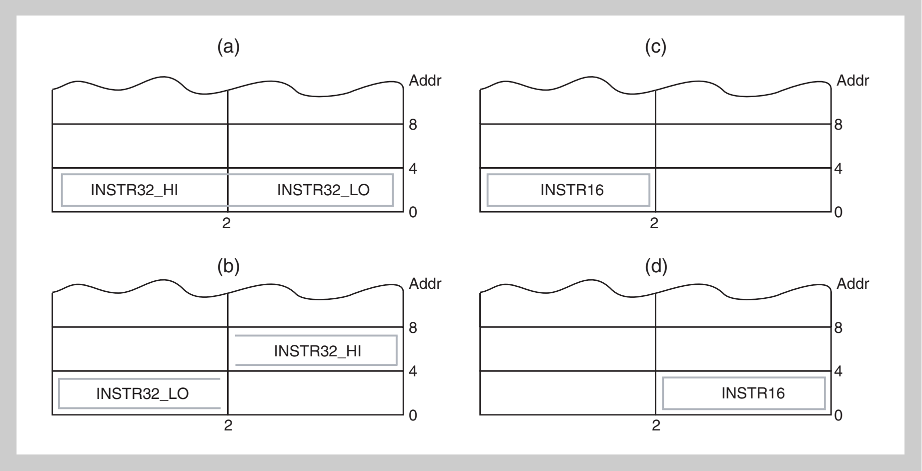


Figure 8-2 Alignment of instructions and data in the internal ROM/flash

(a) 32-bit word aligned data replacement

The 32-bit wide code/data is aligned to a word boundary. Upper and lower halfword are replaced directly by the 32-bit correction value.

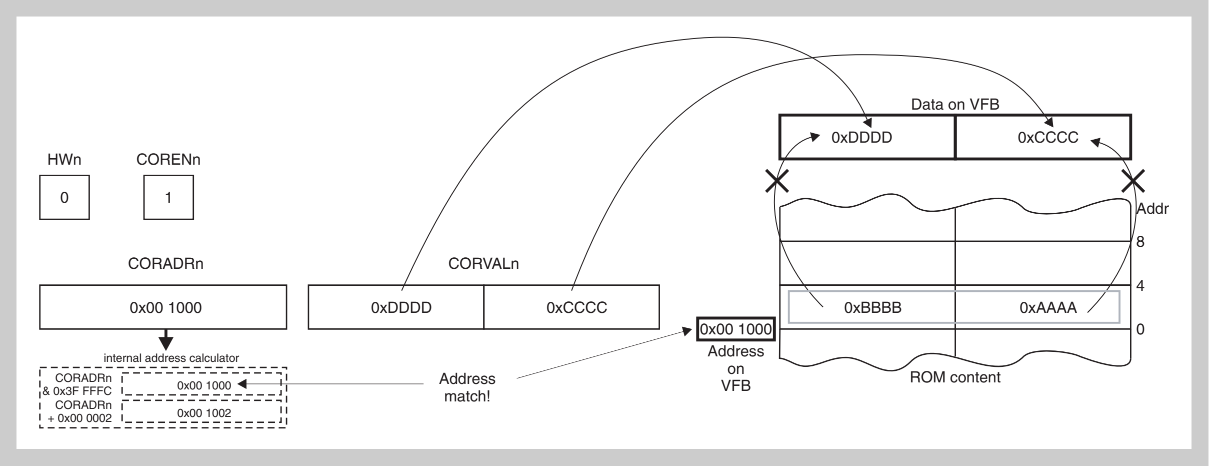


Figure 8-3 32-bit word aligned data replacement

(b) 32-bit word unaligned data replacement

The 32-bit wide code/data is not aligned to a word boundary. For the first VFB access the upper half word is replaced by the lower 16-bit of the correction value (refer to *Figure 8-4* (a)). For the second VFB access the lower halfword is replaced by the upper 16-bit of the correction value (refer to *Figure 8-4* (b)).

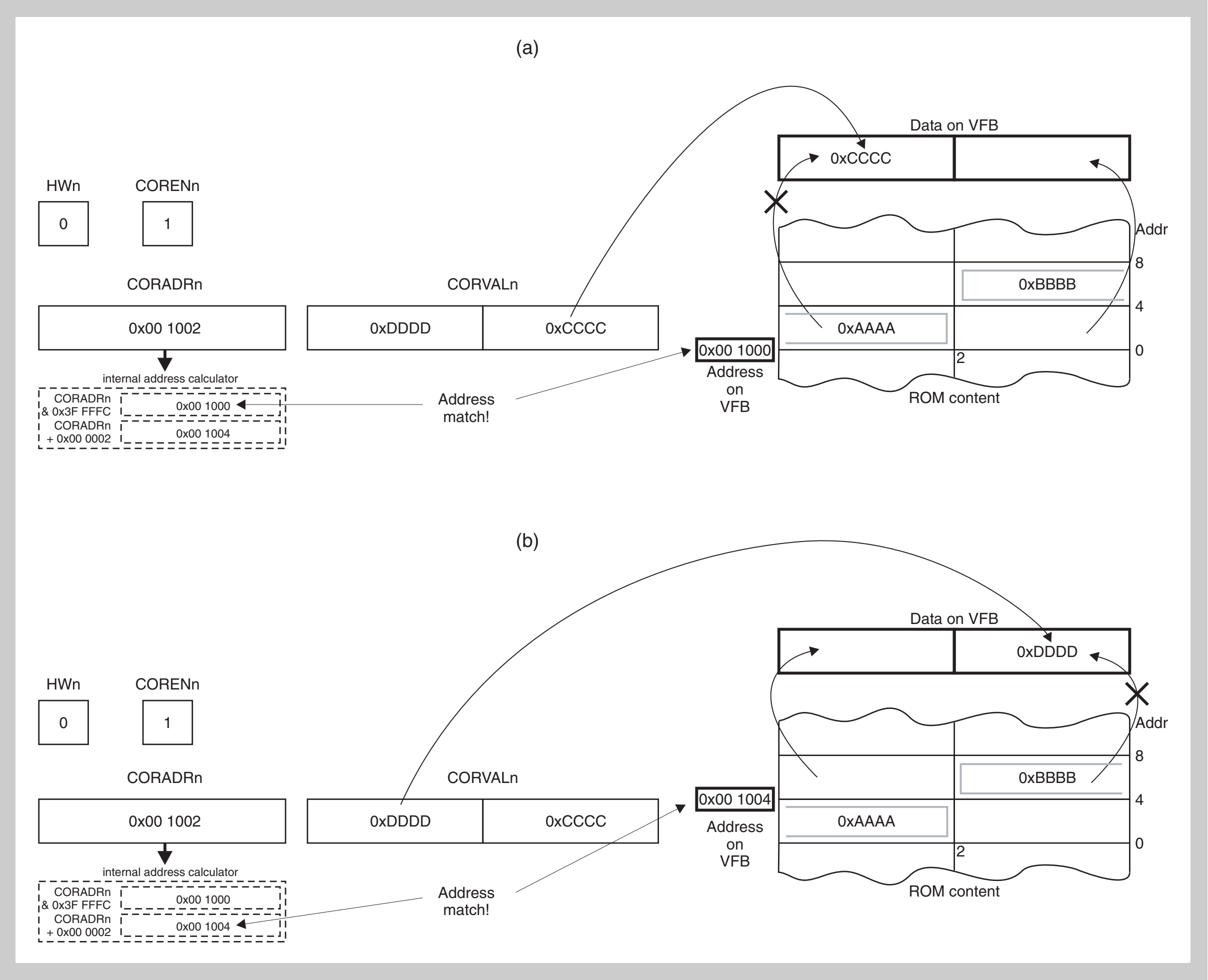


Figure 8-4 32-bit word unaligned data replacement

(c) 16-bit halfword aligned data replacement

The 16-bit wide code/data can be replaced directly by the 16-bit correction value. The upper halfword is not replaced but the original ROM contents is put on the fetch bus.

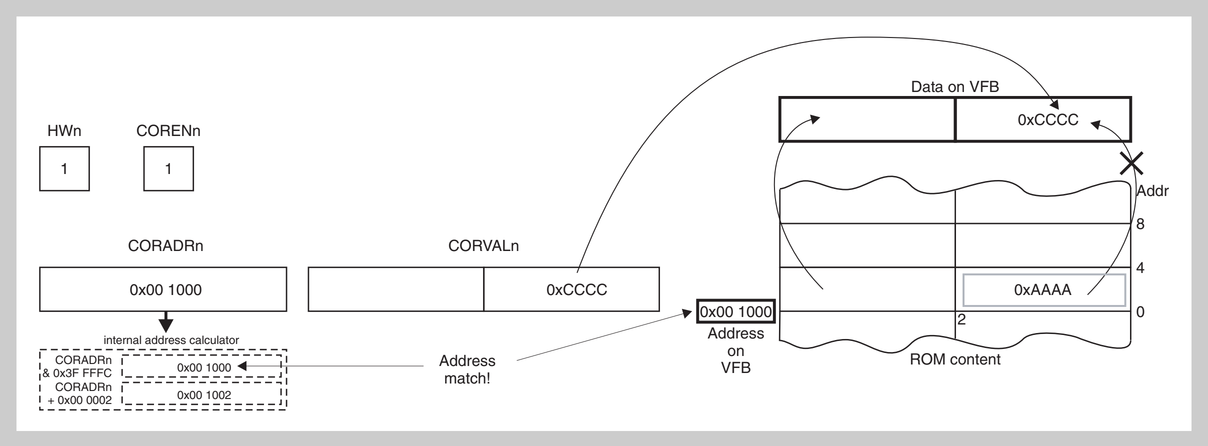


Figure 8-5 16-bit halfword aligned data replacement

(d) 16-bit halfword unaligned data replacement

The 16-bit wide code/data can be replaced directly by the 16-bit correction value. The lower halfword is not replaced but the original ROM contents is put on the fetch bus.

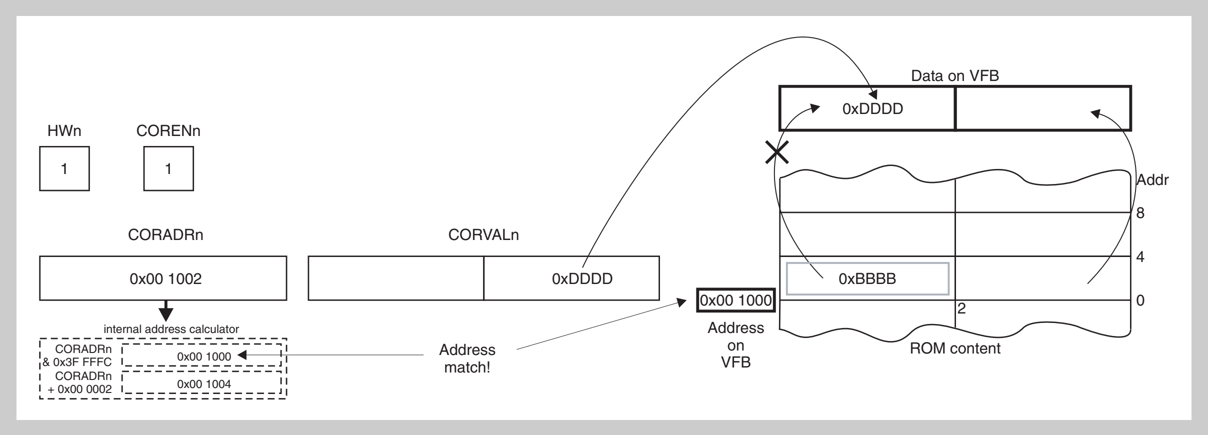


Figure 8-6 16-bit halfword unaligned data replacement

8.2.3 Setting of ROM correction addresses

The CPU supports access to (32-bit) word and (16-bit) half word aligned and unaligned instructions and data.

Aligned words have an address with the lowest two address bits equal 00_B, i.e. ADDRESS modulo 4 = 00_B.

Any access to the ROM is always performed on word aligned addresses. As a consequence access to an unaligned word yields two accesses.

The word in *Figure 8-7* is accessed in two cycles via address 0x00 and 0x04.

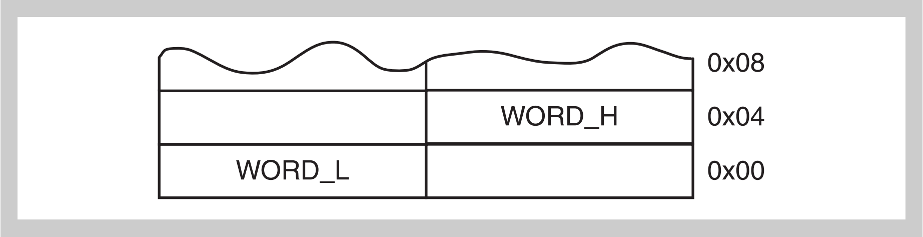


Figure 8-7 Unaligned word addressing

Consequently a ROM correction of an unaligned word is also split into two steps (refer to *"32-bit word unaligned data replacement" on page 229*).

Caution Any (32-bit) aligned word must not contain correction targets of more than one ROM correction channel.

In case of an unaligned word correction (CORCTL1.HWn=0), i.e. CORADRn mod 4 = 10_B, any part (word or half word) of the following two aligned words must not be specified as any other correction address:

- CORADRn div 4
- (CORADRn div 4) + 4

Following consequence applies:

The correction address of an unaligned word must have a distance of at least 6 byte to all other correction addresses, i.e. CORADRn ≥ CORADRm + 6.

One exception consists in the following case. If an unaligned halfword correction address CORADRm (CORCTL1.HWn = 1) precedes in terms of the addresses an unaligned word correction CORADRn (CORCTL1.HWn = 0), a distance of 4 byte is sufficient: CORADRn ≥ CORADRm + 4.

If the setting of ROM correction addresses conflicts with the above, an unaligned word correction shall be split into two halfword corrections. Thus also halfwords of different correction words can be combined in order to correct them in a single aligned access cycle.

Table 8-1 illustrates different combinations and advises how to avoid above conflicts.

Unaligned word and halfword correction

HWORD1	WORD0_H	$x100_B$
WORD0_L	HWORD0	$x000_B$

combine for 2 aligned corrections

CORADRn = $x000_B$	CORVALn = WORD0_L << 16 + HWORD0
CORADRm = $x100_B$	CORVALm = HWORD1 << 16 + WORD0_H

Halfwords correction

HWORD3	HWORD2	$xx00_B$

combine for 1 aligned access

CORADRn = $x000_B$	CORVALn = HWORD3 << 16 + HWORD2

Unaligned words correction

no correction target	WORD1_H	$x100_B$
WORD1_L	WORD2_H	$x100_B$
WORD2_L	no correction target	$x000_B$

combine for 3 aligned corrections

CORADRn = 0000_B	CORVALn = WORD1_H
CORADRm = 0100_B	CORVALm = WORD1_L << 16 + WORD2_H
CORADRl = 1000_B	CORVALl = WORD2_L << 16

Isolated unaligned word correction

no correction target	WORD0_H	$x100_B$
WORD0_L	no correction target	$x000_B$

single unaligned correction

CORADRn = $x010_B$	CORVALn = WORD3_L << 16 + WORD3_H

Table 8-1 ROM correction address settings

8.2.4 “Data Replacement” ROM correction registers

(1) CORCTL0 - VFB flash/ROM “Data Replacement” ROM correction control register 0

This register enables or disables the “Data Replacement” VFB flash/ROM ROM correction of each channel.

Access This register can be read/written in 8- and 1-bit units.

Address FFFF $F900_H$

Initial Value 00_H

7	6	5	4	3	2	1	0
0	0	CORCEN5	CORCEN4	CORCEN3	CORCEN2	CORCEN1	CORCEN0
R	R	R/W	R/W	R/W	R/W	R/W	R/W

Table 8-2 CORCTL0 register contents

Bit Position	Bit Name	Function
5 to 0	CORCENn	ROM correction channel 0: ROM correction for channel n disabled 1: ROM correction for channel n enabled

ROM correction of channel n should only be enabled after the correction address (CORADRn), the correction value (CORVALn) and the word/halfword selection (CORCTL1) have been set.

(2) CORCTL1 - VFB flash/ROM “Data Replacement” ROM correction control register 1

This register determines whether the word (32-bit) or halfword (16-bit) value of CORVALn replaces the VFB flash/ROM contents.

Access This register can be read/written in 8- and 1-bit units.

Address FFFF $F901_H$

Initial Value 00_H

7	6	5	4	3	2	1	0
0	0	HW5	HW4	HW3	HW2	HW1	HW0
R	R	R/W	R/W	R/W	R/W	R/W	R/W

Table 8-3 CORCTL1 register contents

Bit Position	Bit Name	Function
5 to 0	HWn	Word - halfword 0: Word value of CORVALn replaces the flash/ROM contents 1: Halfword value of CORVALn replaces the flash/ROM contents

Note CORCTL1.HWn shall only be changed when the corresponding channel is disabled (CORCTL0.CORCENn = 0).

(3) CORADRnL - VFB flash/ROM “Data Replacement” ROM correction low address register

These registers hold the lower 16 bit of the address where the VFB flash/ROM ROM correction should be performed.

Access These registers can be read/written in 16- and 8-bit units.

Address

CORADR0L, CORADR0LL:	FFFF F910$_H$	CORADR0LH:	FFFF F911$_H$
CORADR1L, CORADR1LL:	FFFF F914$_H$	CORADR1LH:	FFFF F915$_H$
CORADR2L, CORADR2LL:	FFFF F918$_H$	CORADR2LH:	FFFF F919$_H$
CORADR3L, CORADR3LL:	FFFF F91C$_H$	CORADR3LH:	FFFF F91D$_H$
CORADR4L, CORADR4LL:	FFFF F920$_H$	CORADR4LH:	FFFF F921$_H$
CORADR5L, CORADR5LL:	FFFF F924$_H$	CORADR5LH:	FFFF F925$_H$

Initial Value 0000$_H$

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CORADRn[15:0]															0
R/W															

Table 8-4 CORADRnL register contents

Bit Position	Bit Name	Function
15 to 0	CORADRn [15:0]	Lower 16 bit of the ROM correction address of channel n. Bit 0 is fixed to 0, writing to this bit is ignored.

Note CORADRnL shall only be changed when the corresponding channel is disabled (CORCTL0.CORCENn = 0).

(4) CORADRnH - VFB flash/ROM “Data Replacement” ROM correction high address register

These registers hold the upper 6 bit of the address where the VFB flash/ROM ROM correction should be performed.

Access These registers can be read/written in 16- and 8-bit units.

Address

CORADR0H, CORADR0HL:	FFFF $F912_H$	CORADR0HH:	FFFF $F913_H$
CORADR1H, CORADR1HL:	FFFF $F916_H$	CORADR1HH:	FFFF $F917_H$
CORADR2H, CORADR2HL:	FFFF $F91A_H$	CORADR2HH:	FFFF $F91B_H$
CORADR3H, CORADR3HL:	FFFF $F91E_H$	CORADR3HH:	FFFF $F91F_H$
CORADR4H, CORADR4HL:	FFFF $F922_H$	CORADR4HH:	FFFF $F923_H$
CORADR5H, CORADR5HL:	FFFF $F926_H$	CORADR5HH:	FFFF $F927_H$

Initial Value 0000_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	CORADRn[21:16]					

R/W

Table 8-5 CORADRnH register contents

Bit Position	Bit Name	Function
5 to 0	CORADRn [21:16]	Lower 16 bit of the ROM correction address of channel n. Bits 15 to 6are fixed to 0, writing to these bits is ignored.

Caution The ROM correction address CORADRn[21:0] must not exceed the upper address of the internal VSB ROM respectively VSB flash memory.
If the internal VSB ROM/flash memory size is less than 4 MB the appropriate number of upper address bits of CORADRn[21:0] must be set to 0.

Example: If the internal VSB ROM/flash memory size is 1 MB, CORADRn[21:20], i.e. bit 5 and 4 of the CORADRnH, must be set to 00_B. The allowed address range is 0000 0000_H to 000F $FFFF_H$.

Note CORADRnH shall only be changed when the corresponding channel is disabled (CORCTL0.CORCENn = 0).

(5) CORVALnL - VFB flash/ROM “Data Replacement” ROM correction value register

These registers hold the lower 16 bit of the value that shall replace the original value from the VFB flash/ROM.

Access These registers can be read/written in 16-bit units.

Address CORVAL0L: FFFF $F930_H$
CORVAL1L: FFFF $F934_H$
CORVAL2L: FFFF $F938_H$
CORVAL3L: FFFF $F93C_H$
CORVAL4L: FFFF $F940_H$
CORVAL5L: FFFF $F944_H$

Initial Value 0000_H

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">CORVALn[15:0]</td></tr>
<tr><td colspan="16">R/W</td></tr>
</table>

Table 8-6 CORVALnL register contents

Bit Position	Bit Name	Function
15 to 0	CORVALn [15:0]	Lower 16 bit of the correction value to replace the ROM contents.

Note CORVALnL shall only be changed when the corresponding channel is disabled (CORCTL0.CORCENn = 0).

(6) CORVALnH - VFB flash/ROM “Data Replacement” ROM correction value register

These registers hold the upper 16 bit of the value that shall replace the original value from the VFB flash/ROM.

Access These registers can be read/written in 16-bit units.

Address

CORVAL0H:	FFFF F932$_H$
CORVAL1H:	FFFF F936$_H$
CORVAL2H:	FFFF F93A$_H$
CORVAL3H:	FFFF F93E$_H$
CORVAL4H:	FFFF F942$_H$
CORVAL5H:	FFFF F946$_H$

Initial Value 0000$_H$

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">CORVALn[31:16]</td></tr>
<tr><td colspan="16">R/W</td></tr>
</table>

Table 8-7 CORVALnH register contents

Bit Position	Bit Name	Function
15 to 0	CORVALn [31:16]	Upper 16 bit of the correction value to replace the ROM contents.

Note CORVALnH shall only be changed when the corresponding channel is disabled (CORCTL0.CORCENn = 0).

8.3 “DBTRAP” ROM Correction Unit

- 1x 8 channels for VFB flash memory
- The individual channels of “DBTRAP” ROM correction unit are identified by “m” (m = 0 to 7)
- Programmable correction address for each channel
- “DBTRAP” exception processing upon correction address match
- Enable/Disable of each channel individually by software

Caution The “DBTRAP” ROM correction unit is also used by the N-Wire on-chip debug unit. Thus ROM correction will not be performed on these correction channels when the microcontroller is operating in N-Wire debug mode.

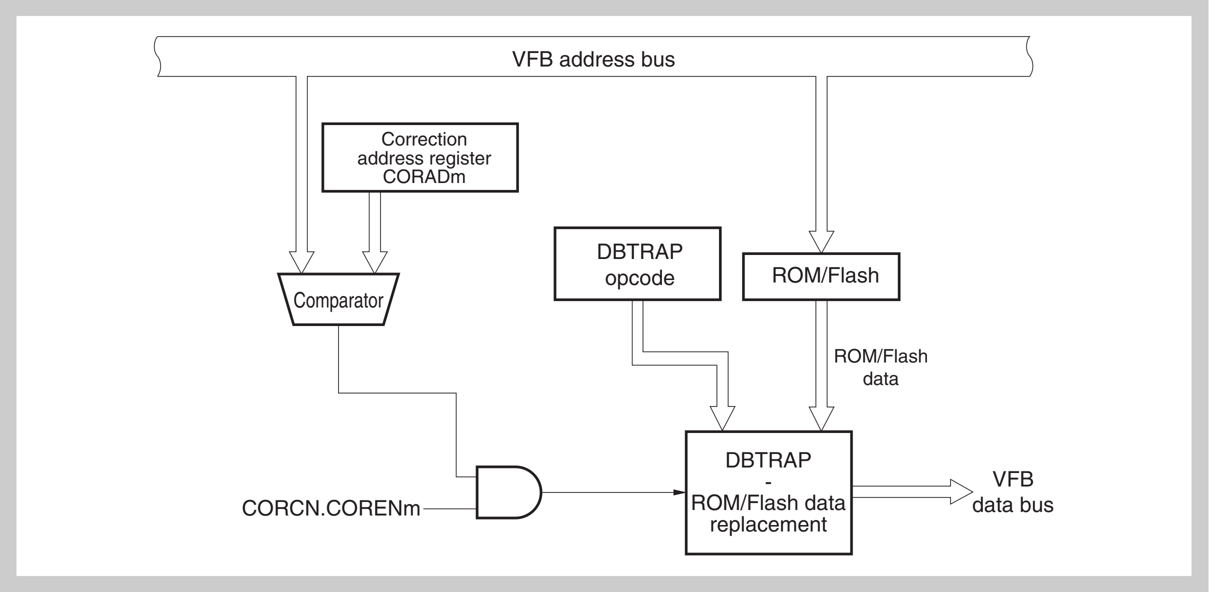


Figure 8-8 “DBTRAP” ROM correction block diagram

8.3.1 “DBTRAP” ROM correction operation

The “DBTRAP” ROM correction unit compares the address on the V850 fetch bus (VFB) with the contents of the programmable correction address registers CORADm. If an address matches, the DBTRAP instruction opcode is put on the V850 fetch bus instead of the ROM contents. If no address matches, the ROM contents is passed on the fetch bus as normal.

The DBTRAP exception branches to the DBTRAP/ILGOP exception handler address 0000 0060_H, which comprises the user’s ROM correction instructions.

Since the ROM correction routines for all correction channels are invoked at the DBTRAP exception handler address 0000 0060_H, the exception handler has to evaluate first the right correction routine to be executed. This is done by reading the DBPC register, which holds the address next to the correction address of CORADm, which has caused the DBTRAP exception. If non of CORADm matches DBPC - 2, DBTRAP was generated by an illegal opcode detection event ILGOP. For further details concerning DBTRAP/ILGOP handling refer to *“Exception Trap“ on page 209*.

Figure 8-9 outlines a typical program flow for using the “DBTRAP” ROM correction.

1. If the address CORADm to be corrected and the fetch address of the internal ROM memory match, the instruction code fetched from ROM is replaced by the DBTRAP instruction.
2. When the DBTRAP instruction is executed, execution branches to address 0000 0060_H.
3. The DBTRAP evaluation routine identifies the cause of the DBTRAP exception and launches either the appropriate ROM correction routine or the ILGOP handler.
4. In case several consecutive ROM instruction are replaced by ROM correction code the return address in DBPC must be corrected. It may also be required to correct some flags in the DBPSW register.
5. Return processing is started by the DBRET instruction.

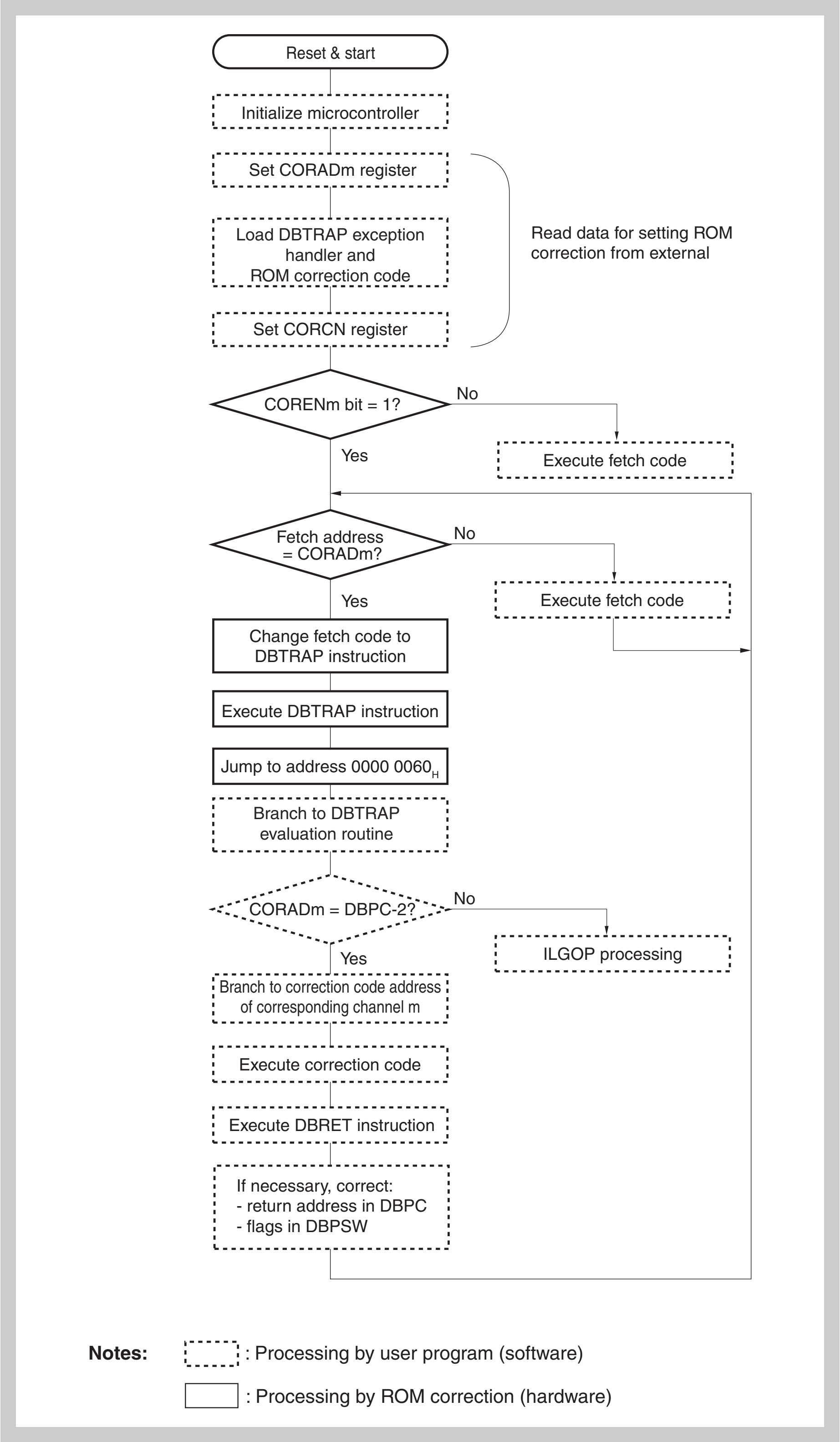

Reset & start
Initialize microcontroller
Set CORADm register
Load DBTRAP exception handler and ROM correction code
Set CORCN register
Read data for setting ROM correction from external
CORENm bit = 1?
No
Execute fetch code
Yes
Fetch address = CORADm?
No
Execute fetch code
Yes
Change fetch code to DBTRAP instruction
Execute DBTRAP instruction
Jump to address 0000 0060H
Branch to DBTRAP evaluation routine
CORADm = DBPC-2?
No
ILGOP processing
Yes
Branch to correction code address of corresponding channel m
Execute correction code
Execute DBRET instruction
If necessary, correct:
- return address in DBPC
- flags in DBPSW
Notes:
: Processing by user program (software)
: Processing by ROM correction (hardware)

Figure 8-9 ROM correction operation and program flow

8.3.2 “DBTRAP” ROM correction registers

(1) CORCN - VFB flash/ROM “DBTRAP” ROM correction control register

This register enables or disables the VFB flash/ROM correction of each channel.

Access This register can be read/written in 8- and 1-bit units.

Address FFFF F880$_H$

Initial Value 0000$_H$

7	6	5	4	3	2	1	0
COREN7	COREN6	COREN5	COREN4	COREN3	COREN2	COREN1	COREN0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 8-8 CORCN register contents

Bit Position	Bit Name	Function
7 to 0	CORENm	ROM correction channel 0: ROM correction for channel m disabled 1: ROM correction for channel m enabled

Note ROM correction of channel n should only be enabled after the correction address CORADm has been set.

(2) CORADm - VFB flash/ROM “DBTRAP” ROM Correction address register

These registers hold the address where the VFB flash/ROM correction should be performed.

Access These registers can be read/written in 32-bit (CORADm) and 16-bit units (CORADmL for bits 15 to 0, CORADmH for bits 31 to 16).

Address

CORAD0, CORAD0L:	FFFF F840$_H$	CORAD0H:	FFFF F842$_H$
CORAD1, CORAD1L:	FFFF F844$_H$	CORAD1H:	FFFF F846$_H$
CORAD2, CORAD2L:	FFFF F848$_H$	CORAD2H:	FFFF F84A$_H$
CORAD3, CORAD3L:	FFFF F84C$_H$	CORAD3H:	FFFF F84E$_H$
CORAD4, CORAD4L:	FFFF F850$_H$	CORAD4H:	FFFF F852$_H$
CORAD5, CORAD5L:	FFFF F854$_H$	CORAD5H:	FFFF F856$_H$
CORAD6, CORAD6L:	FFFF F858$_H$	CORAD6H:	FFFF F85A$_H$
CORAD7, CORAD7L:	FFFF F85C$_H$	CORAD7H:	FFFF F85E$_H$

Initial Value 0000 0000$_H$

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CORADm[15:0]															0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	0	0	0	0	CORADm[19:16]			
R/W															

Table 8-9 CORADm register contents

Bit Position	Bit Name	Function
19 to 0	CORADm [19:0]	Lower 16 bit of the ROM correction address of channel m. Bit 0 and bits 31 to 20 are fixed to 0, writing to these bits is ignored.

Caution The ROM correction address CORADm[19:0] must not exceed the upper address of the internal ROM respectively flash memory.
If the internal ROM/flash memory size is less than 1 MB the appropriate number of upper address bits of CORADm[19:0] must be set to 0.

Note CORADm shall only be changed when the corresponding channel is disabled (CORCN.CORENm = 0).

Chapter 9 Code Protection and Security

9.1 Overview

The microcontroller supports various methods for protecting the program code in the flash memory from undesired access, such as illegal read-out or illegal reprogramming.

Some interfaces offer in general access to the internal flash memory: external flash programmer interface, self-programming facilities and test interfaces.

In the following the security relevant items are listed. The features to protect the internal flash memory data from being read by unauthorized persons are described.

For more information on the flash memory, see *“Flash Memory“ on page 164*.

The following sections give an overview about supported code protection methods.

9.2 Flash Writer and Self-Programming Protection

In general, illegal read-out and re-programming of the flash memory contents is possible via the flash writer interface and the self-programming feature. For protection of the flash memory, the following flags provide various protection levels.

The flags can be set by flash programmers. For a description of flash memory programming see *“Flash Memory“ on page 164*.

(1) Program protection flag (Program protection function)

Set this flag to disable the programming function via flash writer interface. This flag does not affect the self-programming interface.

The flag is valid for the whole flash memory.

(2) Chip erase protection flag (Chip erase protection function)

Set this flag to disable the chip erase function via flash writer interface. This flag does not affect the self-programming interface.

(3) Block erase protection flag (Block erase protection function)

Set this flag to disable the feature to erase single blocks via flash writer interface. This flag does not affect the self-programming interface.

This flag does not affect the chip erase function.

The flag is valid for the whole flash memory.

(4) Read-out protection flag (Read-out protection function)

Set this flag to disable the feature that allows reading back the flash memory via flash writer interface. This flag does not affect the self-programming interface.

This flag is valid for the whole flash memory.

(5) Boot block cluster protection flag

Set this flag to disable erasure and rewrite of the boot block cluster.
The boot block cluster can not be manipulated in any way (no erase/write).

This applies in serial and self-programming mode.

Once this flag is set, it is impossible to reset this flag. Thus the boot block cluster content can not be changed any more.

9.3 Additional Firmware Functions

The internal firmware provides several additional features related to protection and security. These are listed above.

9.3.1 ID-field

A dedicated 64-byte ID-field is provided to hold user defined information, like for instance S/W versions.

The ID-field is stored in the user space of the flash memory, starting at address 0000 0800_H.

The firmware allows to read the ID-field via an external flash programmer data even if the read-out protection flag (refer to *9.2 on page 244*) is set.

9.3.2 Checksum calculation

A dedicated firmware function calculates a checksum over the flash memory contents.

The algorithm to calculate the checksum is “Standard CRC32”.

The checksum calculation starts from address 0000 0000_H to the address stored at 0000 0840_H to 0000 0843_H.

The bytes stored at 0000 0840_H to 0000 0843_H are subject to the checksum.

The 64-byte ID-field is not subject to this checksum.

The firmware allows to read the checksum via an external flash programmer data even if the read-out protection flag (refer to *9.2 on page 244*) is set.

9.3.3 Variable reset vector

The reset vector, determining the start of the user’s program is stored in an “extra area” of the flash memory. This vector is configurable via an external flash programmer and by self-programming.

Chapter 10 16-bit Timer/Event Counter P (TMP)

Timer P (TMP) is a 16-bit timer/event counter.

The V850E/Dx3 - DG3 microcontrollers have following instances of the 16-bit timer/event counter TMP:

TMP	All devices
Instances	1
Names	TMP0

Throughout this chapter, the individual instances of Timer P are identified by “n”, for example TMPn, or TPnCTL0 for the TMPn control register 0.

10.1 Overview

Features summary

An outline of TMPn is shown below.

- Clock selection: 8 ways
- Capture/trigger input pins: 2
- External event count input pins: 1
- External trigger input pins: 1
- Timer/counters: 1
- Capture/compare registers: 2
- Capture/compare match interrupt request signals: 2
- Timer output pins: 2

10.2 Functions

TMPn has the following functions.

- Interval timer
- External event counter
- External trigger pulse output
- One-shot pulse output
- PWM output
- Free-running timer
- Pulse width measurement

10.3 Configuration

TMPn includes the following hardware.

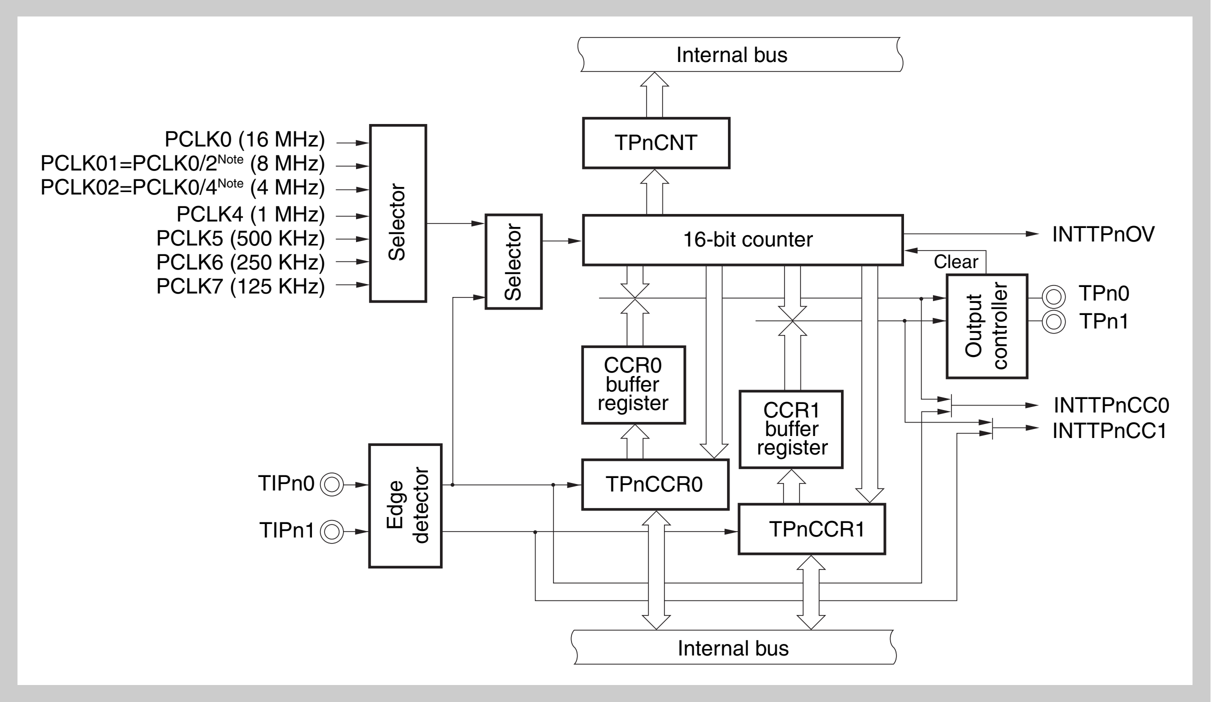


Figure 10-1 Block diagram of TMPn

The second (PCLK01) and the third (PCLK02) clock selector input is not supplied from the Clock Generator, but derived from the first selector input PCLK0 inside the timer P.
In case the PLL is disabled the PCLKx clocks are supplied from the main oscillator, i.e.:

- PCLK0 = 4 MHz
- PCLK01 = PCLK0/2 = 2 MHz
- PCLK02 = PCLK0/4 = 1 MHz

For information about PCLKx, please refer to *“Clock Generator“ on page 100.*

(1) 16-bit counter

This 16-bit counter can count internal clocks or external events.

The count value of this counter can be read by using the TPnCNT register.

When the TPnCTL0.TPnCE bit = 0, the value of the 16-bit counter is FFFFH. If the TPnCNT register is read at this time, 0000H is read.

Reset input clears the TPnCE bit to 0. Therefore, the 16-bit counter is set to FFFFH.

(2) CCR0 buffer register

This is a 16-bit compare register that compares the count value of the 16-bit counter.

When the TPnCCR0 register is used as a compare register, the value written to the TPnCCR0 register is transferred to the CCR0 buffer register. When the count value of the 16-bit counter matches the value of the CCR0 buffer register, a compare match interrupt request signal (INTTPnCC0) is generated.

The CCR0 buffer register cannot be read or written directly.

The CCR0 buffer register is cleared to 0000H after reset, as the TPnCCR0 register is cleared to 0000H.

(3) CCR1 buffer register

This is a 16-bit compare register that compares the count value of the 16-bit counter.

When the TPnCCR1 register is used as a compare register, the value written to the TPnCCR1 register is transferred to the CCR1 buffer register. When the count value of the 16-bit counter matches the value of the CCR1 buffer register, a compare match interrupt request signal (INTTPnCC1) is generated.

The CCR1 buffer register cannot be read or written directly.

The CCR1 buffer register is cleared to 0000H after reset, as the TPnCCR1 register is cleared to 0000H.

(4) Edge detector

This circuit detects the valid edges input to the TIPn0 and TIPn1 pins. No edge, rising edge, falling edge, or both the rising and falling edges can be selected as the valid edge by using the TPnIOC1 and TPnIOC2 registers.

(5) Output controller

This circuit controls the output of the TOPn0 and TOPn1 pins. The output controller is controlled by the TPnIOC0 register.

(6) Selector

This selector selects the count clock for the 16-bit counter. Eight types of internal clocks or an external event can be selected as the count clock.

10.4 TMP Registers

The TMPn are controlled and operated by means of the following registers:

Table 10-1 **TMPn registers overview**

Register name	Shortcut	Address
TMPn control registers 0	TPnCTL0	<base>
TMPn control registers 1	TPnCTL1	<base> + 1_H
TMPn I/O control register 0	TPnIOC0	<base> + 2_H
TMPn I/O control register 1	TPnIOC1	<base> + 3_H
TMPn I/O control register 2	TPnIOC2	<base> + 4_H
TMPn option registers 0	TPnOPT0	<base> + 5_H
TMPn capture/compare registers 0	TPnCCR0	<base> + 6_H
TMPn capture/compare registers 1	TPnCCR1	<base> + 8_H
TMPn counter read buffer register	TPnCNT	<base> + A_H

Table 10-2 **TMPn register base address**

Timer	Base address
TMP0	FFFF $F660_H$

(1) TPnCTL0 - TMPn control register 0

The TPnCTL0 register is an 8-bit register that controls the operation of TMPn.

Access This register can be read/written in 8-bit or 1-bit units.

Address <base>

Initial Value 00_H. This register is initialized by any reset.
The same value can always be written to the TPnCTL0 register by software.

7	6	5	4	3	2	1	0
TPnCE	0	0	0	0	TPnCKS2	TPnCKS1	TPnCKS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 10-3 TPnCTL0 register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>7</td><td>TPnCE</td><td>TMPn operation disable/enable:
0: TMPn operation disabled (TMPn reset asynchronously: reset of TPn0PT0.TPnOVF bit, 16-bit counter, timer output (TOPn0, TOPn1 pins)
1: TMPn operation enabled (TMPn operation starts)</td></tr>
<tr><td>2 to 0</td><td>TPnCKS[2:0]</td><td>Internal count clock selection:
<table>
<tr><th>TPnCKS2</th><th>TPnCKS1</th><th>TPnCKS0</th><th>Internal count clock</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>PCLK0</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>PCLK01 = PCLK0/2</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>PCLK02 = PCLK0/4</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>Prohibited</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>PCLK4</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>PCLK5</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>PCLK6</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>PCLK7</td></tr>
</table>
</td></tr>
</table>

Caution

1. Set the TPnCKS2 to TPnCKS0 bits when the TPnCE bit = 0.
2. When the value of the TPnCE bit is changed from 0 to 1, the TPnCKS2 to TPnCKS0 bits can be set simultaneously.
3. Be sure to clear bits 3 to 6 to 0.

Note For information about PCLKx, please refer to *“Clock Generator“ on page 100*.

(2) TPnCTL1 - TMPn control register 1

The TPnCTL1 register is an 8-bit register that controls the operation of TMPn.

Access This register can be read/written in 8-bit or 1-bit units.

Address <base> + 1_H

Initial Value 00_H. This register is initialized by any reset.

7	6	5	4	3	2	1	0
0	TPnEST	TPnEEE	0	0	TPnMD2	TPnMD1	TPnMD0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 10-4 TPnCTL1 register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>6</td><td>TPnEST</td><td>Software trigger control.
0: –
1: Generate a valid signal for external trigger input.
• In one-shot pulse output mode:
A one-shot pulse is output with writing 1 to the TPnEST bit as the trigger.
• In external trigger pulse output mode:
A PWM waveform is output with writing 1 to the TPnEST bit as the trigger.</td></tr>
<tr><td>5</td><td>TPnEEE</td><td>Count clock selection:
0: Disable operation with external event count input.
(Perform counting with the count clock selected by the TPnCTL0.TPnCK0 to TPnCK2 bits.)
1: Enable operation with external event count input.
(Perform counting at the valid edge of the external event count input signal.)
The TPnEEE bit selects whether counting is performed with the internal count clock or the valid edge of the external event count input.</td></tr>
<tr><td>2 to 0</td><td>TPnMD[2:0]</td><td>Timer mode selection:
<table>
<tr><th>TPnMD2</th><th>TPnMD1</th><th>TPnMD0</th><th>Timer mode</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>Interval timer</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>External event count</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>External trigger pulse output</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>One-shot pulse output</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>PWM output</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Free-runnning timer</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>Pulse width measurement</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>Setting prohibited</td></tr>
</table></td></tr>
</table>

Caution

1. The TPnEST bit is valid only in the external trigger pulse output mode or one-shot pulse output mode. In any other mode, writing 1 to this bit is ignored.
2. External event count input is selected in the external event count mode regardless of the value of the TPnEEE bit.

3. Set the TPnEEE and TPnMD2 to TPnMD0 bits when the TPnCTL0.TPnCE bit = 0. (The same value can be written when the TPnCE bit = 1.) The operation is not guaranteed when rewriting is performed with the TPnCE bit = 1. If rewriting was mistakenly performed, clear the TPnCE bit to 0 and then set the bits again.
4. Be sure to clear bits 3, 4, and 7 to 0.

(3) TPnIOC0 - TMPn I/O control register 0

The TPnIOC0 register is an 8-bit register that controls the timer output (TOPn0, TOPn1 pins).

Access This register can be read/written in 8-bit or 1-bit units.

Address <base> + 2_H

Initial Value 00_H. This register is initialized by any reset.

7	6	5	4	3	2	1	0
0	0	0	0	TPnOL1	TPnOE1	TPnOL0	TPnOE0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 10-5 TPnIOC0 register contents

Bit position	Bit name	Function
3	TPnOL1	TOPn1 pin output level setting: 0: TOPn1 pin output inversion disabled 1: TOPn1 pin output inversion enabled
2	TPnOE1	TOPn1 pin output setting: 0: Timer output disable – when TPnOL1 = 0: low level is output from TOPn1 pin – when TPnOL1 = 1: high level is output from TOPn1 pin 1: Timer output enable (A square wave is output from TOPn1 pin.)
1	TPnOL0	TOPn0 pin output level setting: 0: TOPn0 pin output inversion disabled 1: TOPn0 pin output inversion enabled
0	TPnOE0	TOPn0 pin output setting: 0: Timer output disable – when TPnOL0 = 0: low level is output from TOPn0 pin – when TPnOL0 = 1: high level is output from TOPn0 pin 1: Timer output enable (A square wave is output from TOPn0 pin.)

Caution

1. Rewrite the TPnOL1, TPnOE1, TPnOL0, and TPnOE0 bits when the TPnCTL0.TPnCE bit = 0. (The same value can be written when the TPnCE bit = 1.) If rewriting was mistakenly performed, clear the TPnCE bit to 0 and then set the bits again.
2. Even if the TPnOLm bit is manipulated when the TPnCE and TPnOEm bits are 0, the TOPnm pin output level varies (m = 0, 1).

(4) TPnIOC1 - TMPn I/O control register 1

The TPnIOC1 register is an 8-bit register that controls the valid edge of the capture trigger input signals (TIPn0, TIPn1 pins).

Access This register can be read/written in 8-bit or 1-bit units.

Address <base> + 3_H

Initial Value 00_H. This register is initialized by any reset.

7	6	5	4	3	2	1	0
0	0	0	0	TPnIS3	TPnIS2	TPnIS1	TPnIS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 10-6 TPnIOC1 register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>3 to 2</td><td>TPnIS[3:2]</td><td>Capture trigger input signal (TIPn1 pin) valied edge setting:
<table>
<tr><th>TPnIS3</th><th>TPnIS2</th><th>Capture trigger valid edge of TIPn1</th></tr>
<tr><td>0</td><td>0</td><td>No edge detection (capture operation invalid)</td></tr>
<tr><td>0</td><td>1</td><td>Detection of rising edge</td></tr>
<tr><td>1</td><td>0</td><td>Detection of falling edge</td></tr>
<tr><td>1</td><td>1</td><td>Detection of both edges</td></tr>
</table></td></tr>
<tr><td>1 to 0</td><td>TPnIS[1:0]</td><td>Capture trigger input signal (TIPn0 pin) valied edge setting:
<table>
<tr><th>TPnIS1</th><th>TPnIS0</th><th>Capture trigger valid edge of TIPn0</th></tr>
<tr><td>0</td><td>0</td><td>No edge detection (capture operation invalid)</td></tr>
<tr><td>0</td><td>1</td><td>Detection of rising edge</td></tr>
<tr><td>1</td><td>0</td><td>Detection of falling edge</td></tr>
<tr><td>1</td><td>1</td><td>Detection of both edges</td></tr>
</table></td></tr>
</table>

Caution

1. Rewrite the TPnIS3 to TPnIS0 bits when the TPnCTL0.TPnCE bit = 0. (The same value can be written when the TPnCE bit = 1.) If rewriting was mistakenly performed, clear the TPnCE bit to 0 and then set the bits again.
2. The TPnIS3 to TPnIS0 bits are valid only in the free-running timer mode and the pulse width measurement mode. In all other modes, a capture operation is not possible.

(5) TPnIOC2 - TMPn I/O control register 2

The TPnIOC2 register is an 8-bit register that controls the valid edge of the external event count input signal (TIPn0 pin) and external trigger input signal (TIPn0 pin).

Access This register can be read/written in 8-bit or 1-bit units.

Address <base> + 4_H

Initial Value 00_H. This register is initialized by any reset.

7	6	5	4	3	2	1	0
0	0	0	0	TPnEES1	TPnEES0	TPnETS1	TPnETS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 10-7 TPnIOC2 register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>3 to 2</td><td>TPnEES[1:0]</td><td>External event count input signal (TIPn0 pin) valid edge setting:
<table>
<tr><th>TPnEES1</th><th>TPnEES0</th><th>External event count valid edge of TIPn0</th></tr>
<tr><td>0</td><td>0</td><td>No edge detection (external event invalid)</td></tr>
<tr><td>0</td><td>1</td><td>Detection of rising edge</td></tr>
<tr><td>1</td><td>0</td><td>Detection of falling edge</td></tr>
<tr><td>1</td><td>1</td><td>Detection of both edges</td></tr>
</table></td></tr>
<tr><td>1 to 0</td><td>TPnETS[1:0]</td><td>Capture trigger input signal (TIPn0 pin) valid edge setting:
<table>
<tr><th>TPnETS1</th><th>TPnETS0</th><th>External trigger input valid edge of TIPn0</th></tr>
<tr><td>0</td><td>0</td><td>No edge detection (external trigger invalid)</td></tr>
<tr><td>0</td><td>1</td><td>Detection of rising edge</td></tr>
<tr><td>1</td><td>0</td><td>Detection of falling edge</td></tr>
<tr><td>1</td><td>1</td><td>Detection of both edges</td></tr>
</table></td></tr>
</table>

Caution

1. Rewrite the TPnEES1, TPnEES0, TPnETS1, and TPnETS0 bits when the TPnCTL0.TPnCE bit = 0. (The same value can be written when the TPnCE bit = 1.) If rewriting was mistakenly performed, clear the TPnCE bit to 0 and then set the bits again.
2. The TPnEES1 and TPnEES0 bits are valid only when the TPnCTL1.TPnEEE bit = 1 or when the external event count mode (TPnCTL1.TPnMD2 to TPnCTL1.TPnMD0 bits = 001) has been set.

(6) TPnOPT0 - TMPn option register 0

The TPnOPT0 register is an 8-bit register used to set the capture/compare operation and detect an overflow.

Access This register can be read/written in 8-bit or 1-bit units.

Address <base> + 5_H

Initial Value 00_H. This register is initialized by any reset.

7	6	5	4	3	2	1	0
0	0	TPnCCS1	TPnCCS10	0	0	0	TPnOVF
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 10-8 TPnOPT0 register contents

Bit position	Bit name	Function
5	TPnCCS1	TPnCCR1 register capture/compare selection: 0: compare register selected 1: capture register selected The TPnCCS1 bit setting is valid only in the free-running timer mode.
4	TPnCCS0	TPnCCR0 register capture/compare selection: 0: compare register selected 1: capture register selected The TPnCCS0 bit setting is valid only in the free-running timer mode.
0	TPnOVF	TMPn overflow detection flag: Set (1): Overflow occurred Reset (0): TPnOVF bit 0 written or TPnCTL0.TPnCE bit = 0 • The TPnOVF bit is reset when the 16-bit counter count value overflows from FFFFH to 0000H in the free-running timer mode or the pulse width measurement mode. • An interrupt request signal (INTTPnOV) is generated at the same time that the TPnOVF bit is set to 1. The INTTPnOV signal is not generated in modes other than the free-running timer mode and the pulse width measurement mode. • The TPnOVF bit is not cleared even when the TPnOVF bit or the TPnOPT0 register are read when the TPnOVF bit = 1. • The TPnOVF bit can be both read and written, but the TPnOVF bit cannot be set to 1 by software. Writing 1 has no influence on the operation of TMPn.

Caution

1. Rewrite the TPnCCS1 and TPnCCS0 bits when the TPnCE bit = 0. (The same value can be written when the TPnCE bit = 1.) If rewriting was mistakenly performed, clear the TPnCE bit to 0 and then set the bits again.
2. Be sure to clear bits 1 to 3, 6, and 7 to 0.

(7) TPnCCR0 - TMPn capture/compare register 0

The TPnCCR0 register can be used as a capture register or a compare register depending on the mode.

This register can be used as a capture register or a compare register only in the free-running timer mode, depending on the setting of the TPnOPT0.TPnCCS0 bit. In the pulse width measurement mode, the TPnCCR0 register can be used only as a capture register. In any other mode, this register can be used only as a compare register.

The TPnCCR0 register can be read or written during operation.

Access This register can be read/written in 16-bit units.

Address <base> + 6_H

Initial Value 0000_H. This register is initialized by any reset.

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">CCR0 value</td></tr>
<tr><td colspan="16">R/W</td></tr>
</table>

(a) Function as compare register

The TPnCCR0 register can be rewritten even when the TPnCTL0.TPnCE bit = 1.

The set value of the TPnCCR0 register is transferred to the CCR0 buffer register. When the value of the 16-bit counter matches the value of the CCR0 buffer register, a compare match interrupt request signal (INTTPnCC0) is generated. If TOPn0 pin output is enabled at this time, the output of the TOPn0 pin is inverted.

When the TPnCCR0 register is used as a cycle register in the interval timer mode, external event count mode, external trigger pulse output mode, one-shot pulse output mode, or PWM output mode, the value of the 16-bit counter is cleared (0000H) if its count value matches the value of the CCR0 buffer register.

(b) Function as capture register

When the TPnCCR0 register is used as a capture register in the free-running timer mode, the count value of the 16-bit counter is stored in the TPnCCR0 register if the valid edge of the capture trigger input pin (TIPn0 pin) is detected. In the pulse-width measurement mode, the count value of the 16-bit counter is stored in the TPnCCR0 register and the 16-bit counter is cleared (0000H) if the valid edge of the capture trigger input pin (TIPn0) is detected.

Even if the capture operation and reading the TPnCCR0 register conflict, the correct value of the TPnCCR0 register can be read.

The following table shows the functions of the capture/compare register in each mode, and how to write data to the compare register.

Table 10-9 Function of capture/compare register in each mode and how to write compare register

Operation mode	Capture/compare register	How to write compare register
Interval timer	Compare register	Anytime write
External event counter	Compare register	Anytime write
External trigger pulse output	Compare register	Batch write
One-shot pulse output	Compare register	Anytime write
PWM output	Compare register	Batch write
Free-running timer	Capture/compare register	Anytime write
Pulse width measurement	Capture register	-

(8) TPnCCR1 - TMPn capture/compare register 1

The TPnCCR1 register can be used as a capture register or a compare register depending on the mode.

This register can be used as a capture register or a compare register only in the free-running timer mode, depending on the setting of the TPnOPT0.TPnCCS1 bit. In the pulse width measurement mode, the TPnCCR1 register can be used only as a capture register. In any other mode, this register can be used only as a compare register.

The TPnCCR1 register can be read or written during operation.

Access This register can be read/written in 16-bit units.

Address <base> + 8_H

Initial Value 0000_H. This register is initialized by any reset.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1 value															
R/W															

(a) Function as compare register

The TPnCCR1 register can be rewritten even when the TPnCTL0.TPnCE bit = 1.

The set value of the TPnCCR1 register is transferred to the CCR1 buffer register. When the value of the 16-bit counter matches the value of the CCR1 buffer register, a compare match interrupt request signal (INTTPnCC1) is generated. If TOPn1 pin output is enabled at this time, the output of the TOPn1 pin is inverted.

(b) Function as capture register

When the TPnCCR1 register is used as a capture register in the free-running timer mode, the count value of the 16-bit counter is stored in the TPnCCR1 register if the valid edge of the capture trigger input pin (TIPn1 pin) is detected. In the pulse-width measurement mode, the count value of the 16-bit counter is stored in the TPnCCR1 register and the 16-bit counter is cleared (0000H) if the valid edge of the capture trigger input pin (TIPn1) is detected.

Even if the capture operation and reading the TPnCCR1 register conflict, the correct value of the TPnCCR1 register can be read.

The following table shows the functions of the capture/compare register in each mode, and how to write data to the compare register.

Table 10-10 Function of capture/compare register in each mode and how to write compare register

Operationmode	Capture/compare register	How to write compare register
Interval timer	Compare register	Anytime write
External event counter	Compare register	Anytime write
External trigger pulse output	Compare register	Batch write
One-shot pulse output	Compare register	Anytime write
PWM output	Compare register	Batch write
Free-running timer	Capture/compare register	Anytime write
Pulse width measurement	Capture register	-

(9) TPnCNT - TMPn counter read buffer register

The TPnCNT register is a read buffer register that can read the count value of the 16-bit counter.

If this register is read when the TPnCTL0.TPnCE bit = 1, the count value of the 16-bit timer can be read.

The value of the TPnCNT register is cleared to 0000_H when the TPnCE bit = 0. If the TPnCNT register is read at this time, the value of the 16-bit counter ($FFFF_H$) is not read, but 0000_H is read.

Access This register can be read only in 16-bit units.

Address <base> + A_H

Initial Value 0000_H. This register is initialized by any reset, as the TPnCE bit is cleared to 0.

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">TPnCNT value</td></tr>
<tr><td colspan="16">R</td></tr>
</table>

10.5 Operation

TMPn can perform the following operations.

Operation	TPnCTL1.TPnEST Bit (Software Trigger Bit)	TIPn0 Pin (Ext. Trigger Input)	Capture/ Compare Register Setting	Compare Register Write
Interval timer mode	Invalid	Invalid	Compare only	Anytime write
External event count mode[Note 1]	Invalid	Invalid	Compare only	Anytime write
External trigger pulse output mode[Note 2]	Valid	Valid	Compare only	Batch write
One-shot pulse output mode[Note 2]	Valid	Valid	Compare only	Anytime write
PWM output mode	Invalid	Invalid	Compare only	Batch write
Free-running timer mode	Invalid	Invalid	Switching enabled	Anytime write
Pulse width measurement mode[Note 2]	Invalid	Invalid	Capture only	Not applicable

Note 1. To use the external event count mode, specify that the valid edge of the TIPn0 pin capture trigger input is not detected (by clearing the TPnIOC1.TPnIS1 and TPnIOC1.TPnIS0 bits to “00”).

2. When using the external trigger pulse output mode, one-shot pulse output mode, and pulse width measurement mode, select the internal clock as the count clock (by clearing the TPnCTL1.TPnEEE bit to 0).

10.5.1 Interval timer mode (TPnMD2 to TPnMD0 = 000)

In the interval timer mode, an interrupt request signal (INTTPnCC0) is generated at the specified interval if the TPnCTL0.TPnCE bit is set to 1. A square wave whose half cycle is equal to the interval can be output from the TOPn0 pin.

Usually, the TPnCCR1 register is not used in the interval timer mode.

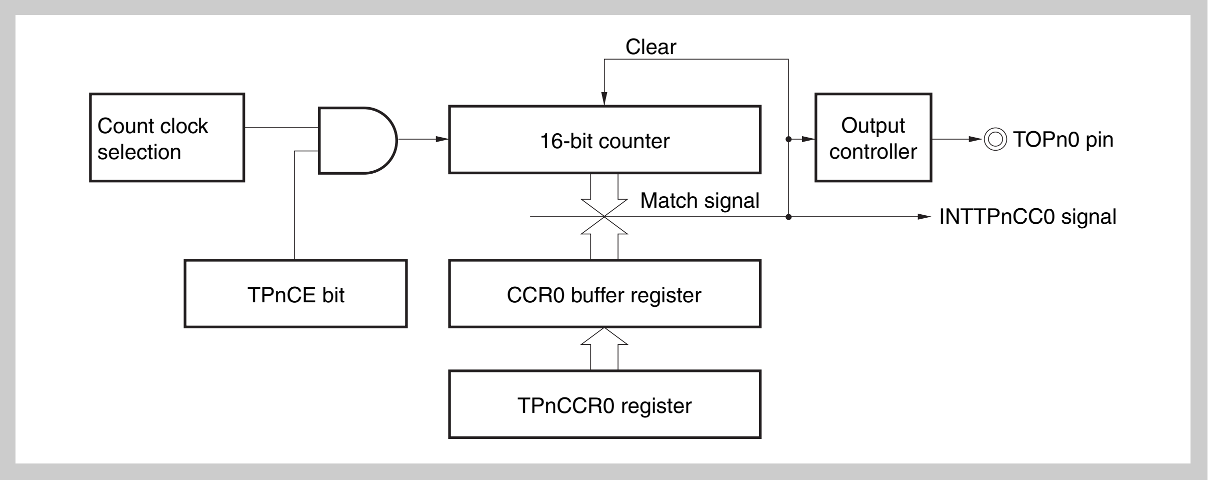


Figure 10-2 Configuration of interval timer

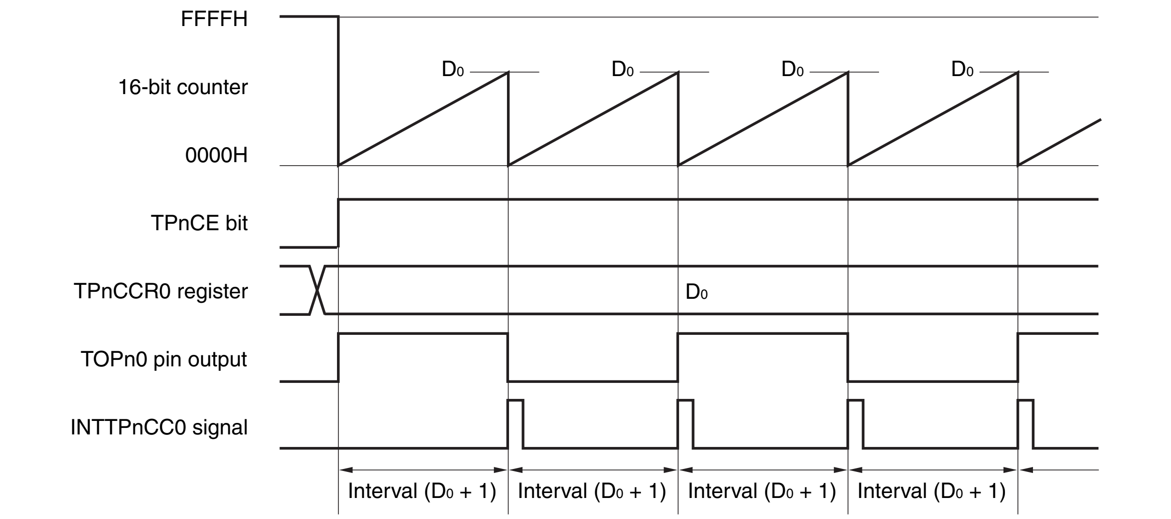


Figure 10-3 Basic timing of operation in interval timer mode

When the TPnCE bit is set to 1, the value of the 16-bit counter is cleared from FFFFH to 0000H in synchronization with the count clock, and the counter starts counting. At this time, the output of the TOPn0 pin is inverted. Additionally, the set value of the TPnCCR0 register is transferred to the CCR0 buffer register.

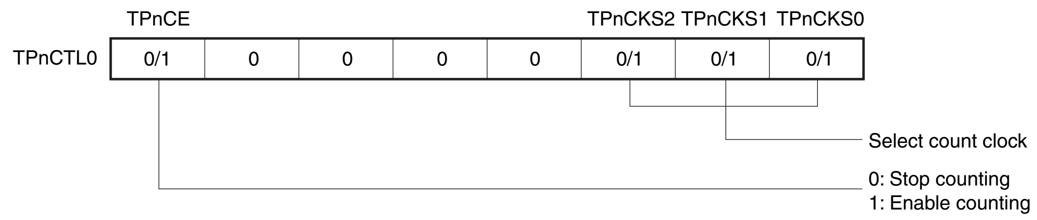
When the count value of the 16-bit counter matches the value of the CCR0 buffer register, the 16-bit counter is cleared to 0000H, the output of the TOPn0 pin is inverted, and a compare match interrupt request signal (INTTPnCC0) is generated.

The interval can be calculated by the following expression.

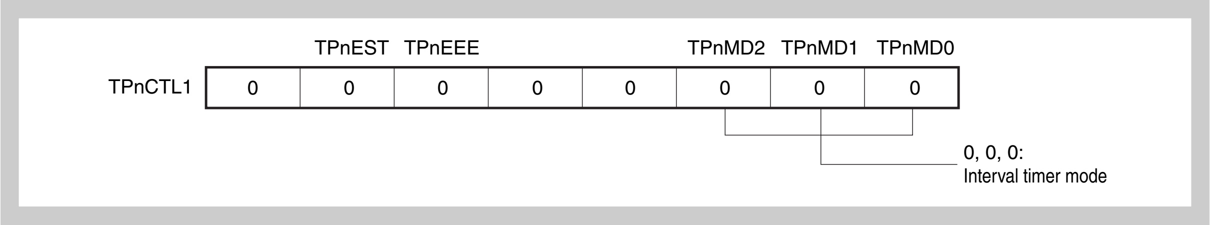
Interval = (Set value of TPnCCR0 register + 1) × Count clock cycle

(1) Register setting for interval timer mode operation

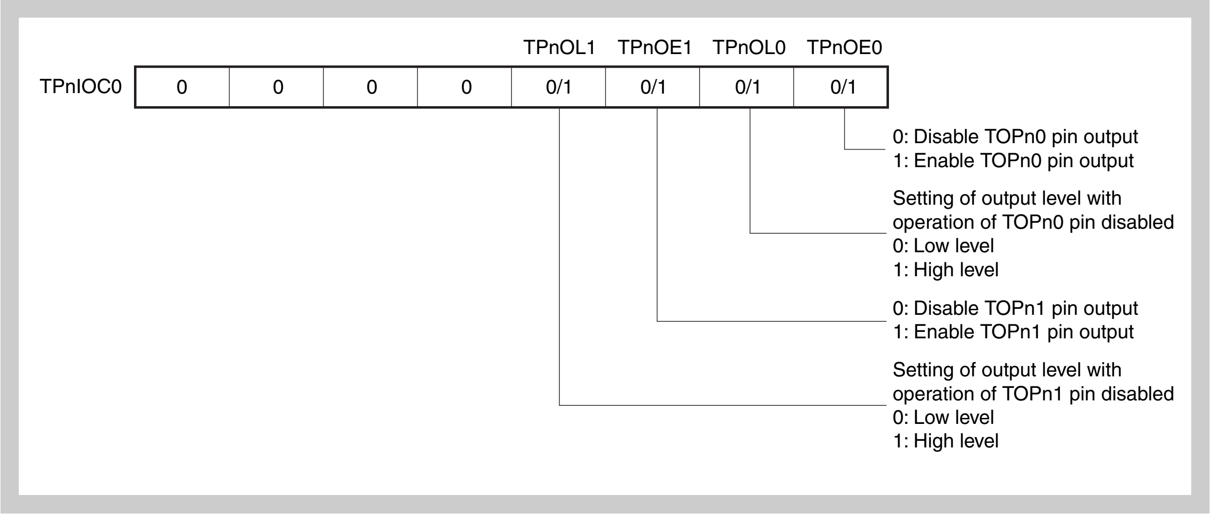
(a) TMPn control register 0 (TPnCTL0)



(b) TMPn control register 1 (TPnCTL1)



(c) TMPn I/O control register 0 (TPnIOC0)


TPnOL1
TPnOE1
TPnOL0
TPnOE0
TPnIOC0
0
0
0
0
0/1
0/1
0/1
0/1
0: Disable TOPn0 pin output
1: Enable TOPn0 pin output
Setting of output level with operation of TOPn0 pin disabled
0: Low level
1: High level
0: Disable TOPn1 pin output
1: Enable TOPn1 pin output
Setting of output level with operation of TOPn1 pin disabled
0: Low level
1: High level

(d) TMPn counter read buffer register (TPnCNT)

By reading the TPnCNT register, the count value of the 16-bit counter can be read.

(e) TMPn capture/compare register 0 (TPnCCR0)

If the TPnCCR0 register is set to D_0, the interval is as follows.

Interval = (D_0 + 1) × Count clock cycle

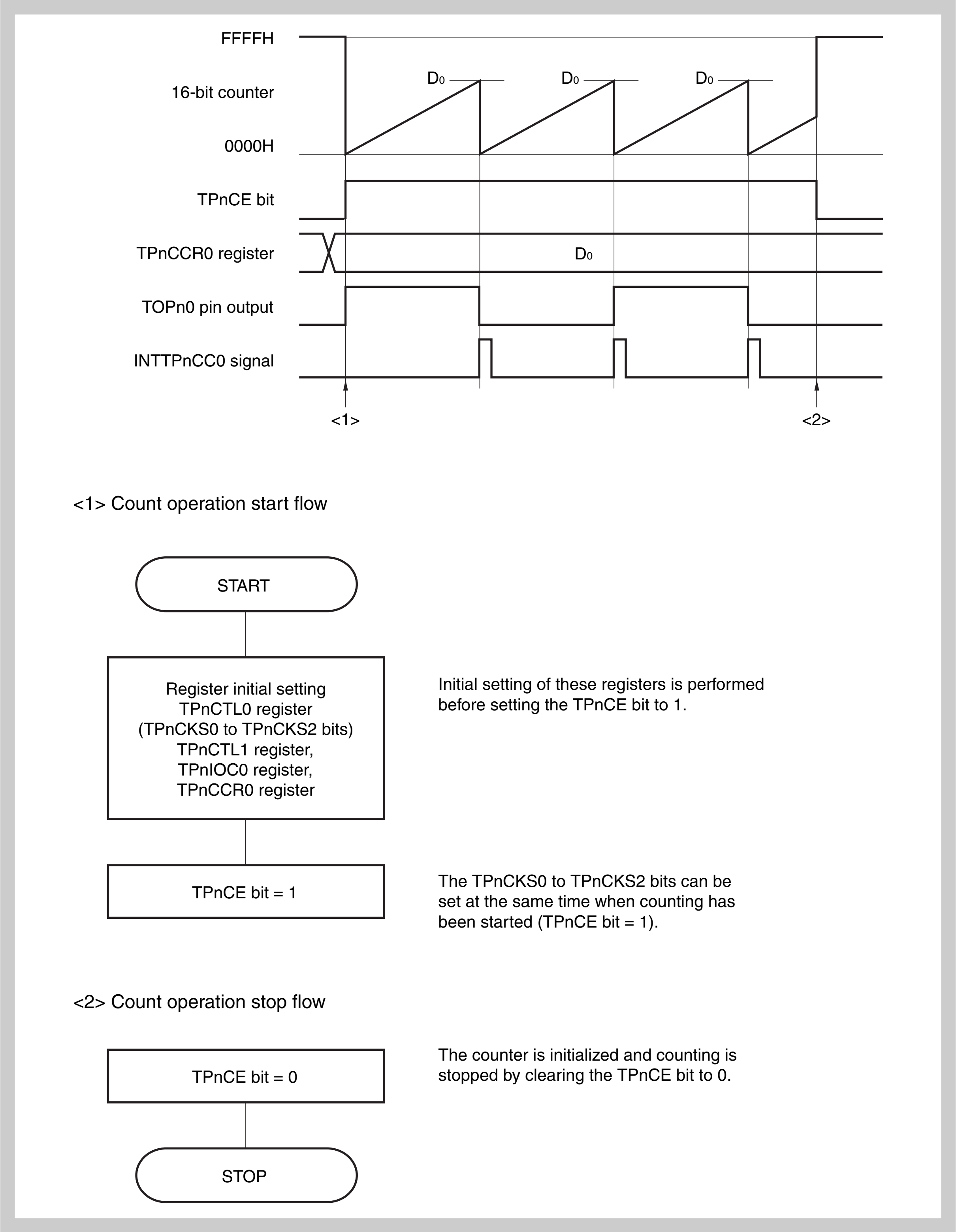
(f) TMPn capture/compare register 1 (TPnCCR1)

Usually, the TPnCCR1 register is not used in the interval timer mode. However, the set value of the TPnCCR1 register is transferred to the CCR1 buffer register. A compare match interrupt request signal (INTTPnCC1) is generated when the count value of the 16-bit counter matches the value of the CCR1 buffer register.

Therefore, mask the interrupt request by using the corresponding interrupt mask flag (TPnCCMK1).

Note TMPn I/O control register 1 (TPnIOC1), TMPn I/O control register 2 (TPnIOC2), and TMPn option register 0 (TPnOPT0) are not used in the interval timer mode.

(2) Interval timer mode operation flow



Initial setting of these registers is performed before setting the TPnCE bit to 1.

The TPnCKS0 to TPnCKS2 bits can be set at the same time when counting has been started (TPnCE bit = 1).

The counter is initialized and counting is stopped by clearing the TPnCE bit to 0.

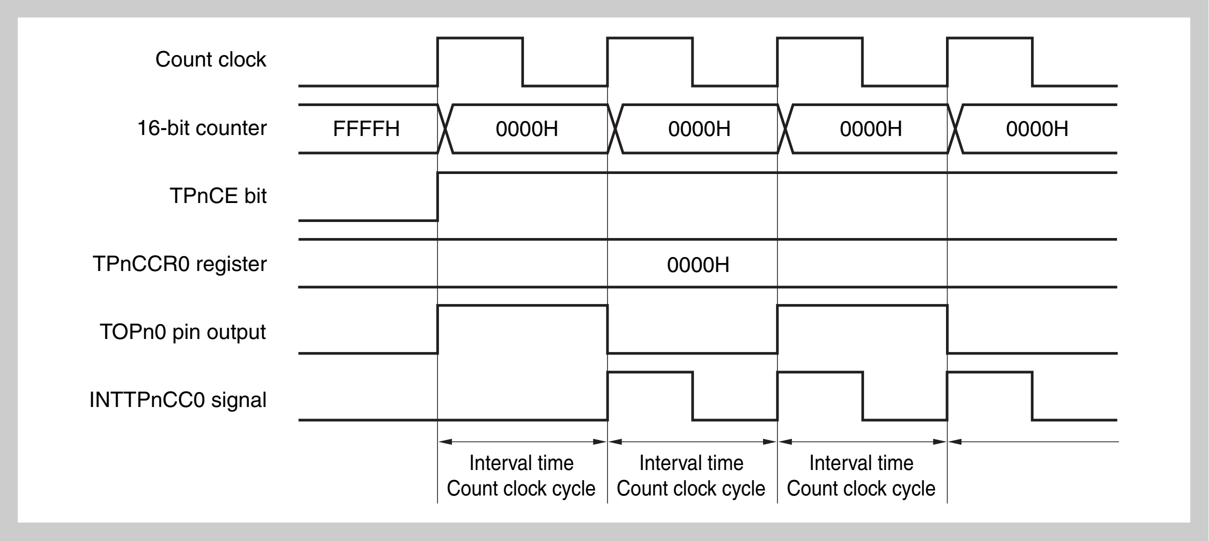
Figure 10-4 Software processing flow in interval timer mode

(3) Interval timer mode operation timing

(a) Operation if TPnCCR0 register is set to 0000H

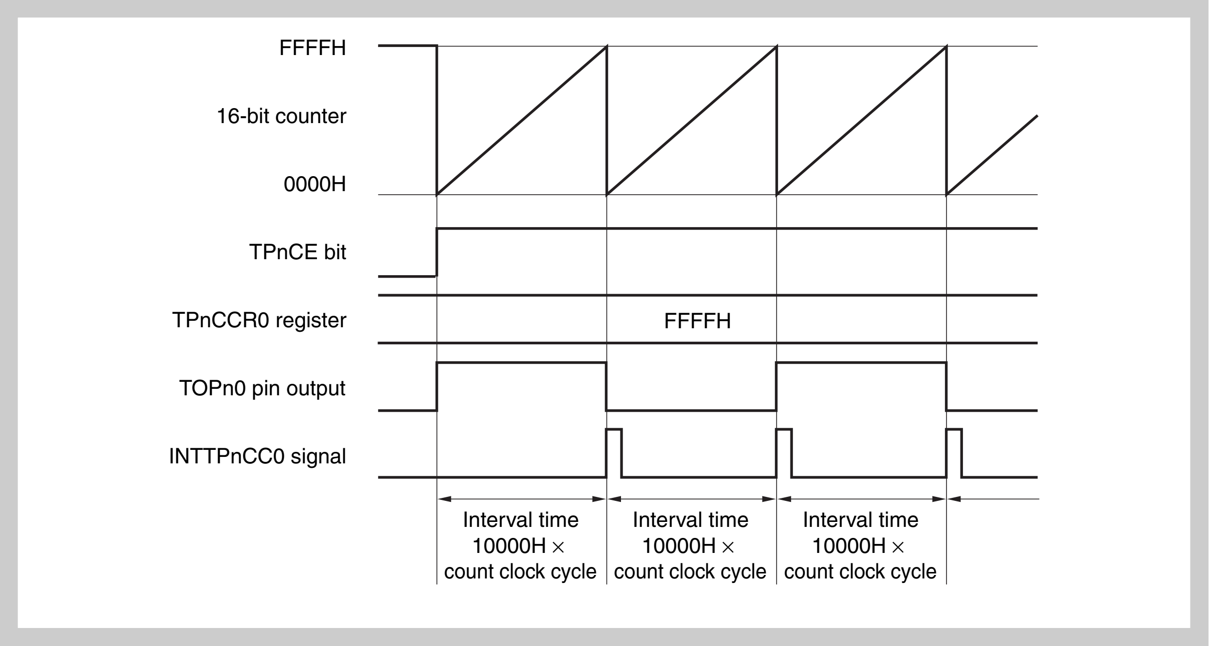
If the TPnCCR0 register is set to 0000H, the INTTPnCC0 signal is generated at each count clock, and the output of the TOPn0 pin is inverted.

The value of the 16-bit counter is always 0000H.



(b) Operation if TPnCCR0 register is set to FFFFH

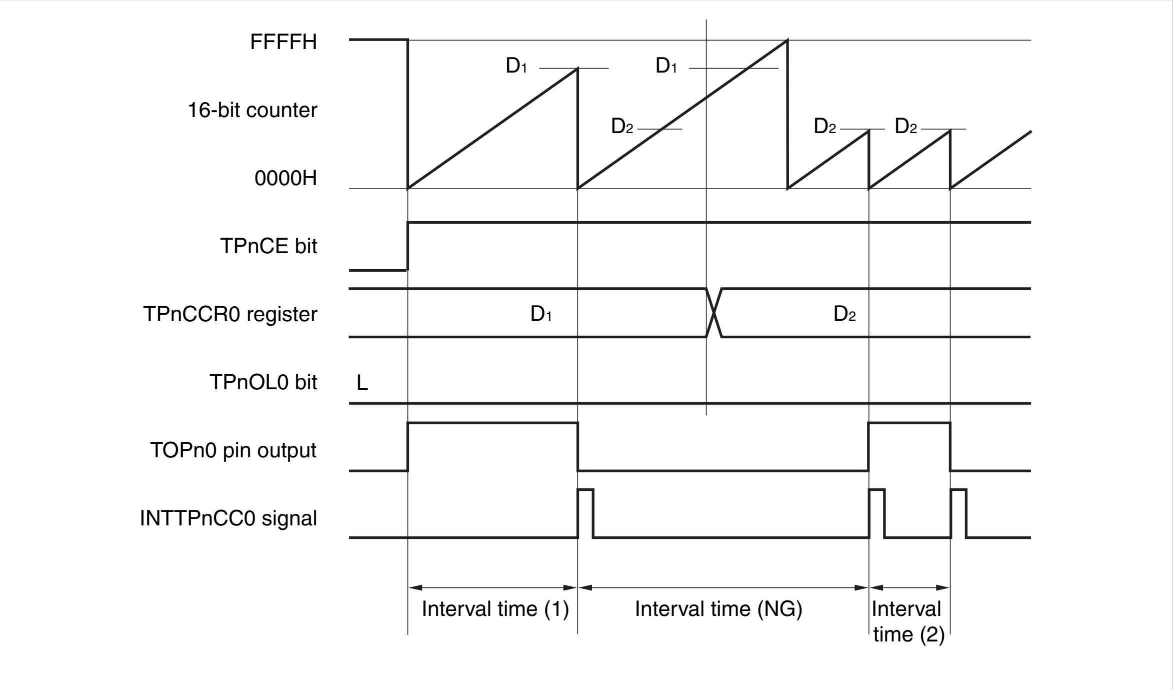
If the TPnCCR0 register is set to FFFFH, the 16-bit counter counts up to FFFFH. The counter is cleared to 0000H in synchronization with the next count-up timing. The INTTPnCC0 signal is generated and the output of the TOPn0 pin is inverted. At this time, an overflow interrupt request signal (INTTPnOV) is not generated, nor is the overflow flag (TPnOPT0.TPnOVF bit) set to 1.



(c) Notes on rewriting TPnCCR0 register

To change the value of the TPnCCR0 register to a smaller value, stop counting once and then change the set value.

If the value of the TPnCCR0 register is rewritten to a smaller value during counting, the 16-bit counter may overflow.



Note
1. Interval time (1): (D_1 + 1) × Count clock cycle
2. Interval time (NG): (10000H + D_2 + 1) × Count clock cycle
3. Interval time (2): (D_2 + 1) × Count clock cycle

If the value of the TPnCCR0 register is changed from D_1 to D_2 while the count value is greater than D_2 but less than D_1, the count value is transferred to the CCR0 buffer register as soon as the TPnCCR0 register has been rewritten. Consequently, the value of the 16-bit counter that is compared is D_2.

Because the count value has already exceeded D_2, however, the 16-bit counter counts up to FFFFH, overflows, and then counts up again from 0000H. When the count value matches D_2, the INTTPnCC0 signal is generated and the output of the TOPn0 pin is inverted.

Therefore, the INTTPnCC0 signal may not be generated at the interval time "(D_1 + 1) × Count clock cycle" or "(D_2 + 1) × Count clock cycle" originally expected, but may be generated at an interval of "(10000H + D_2 + 1) × Count clock period".

(d) Operation of TPnCCR1 register

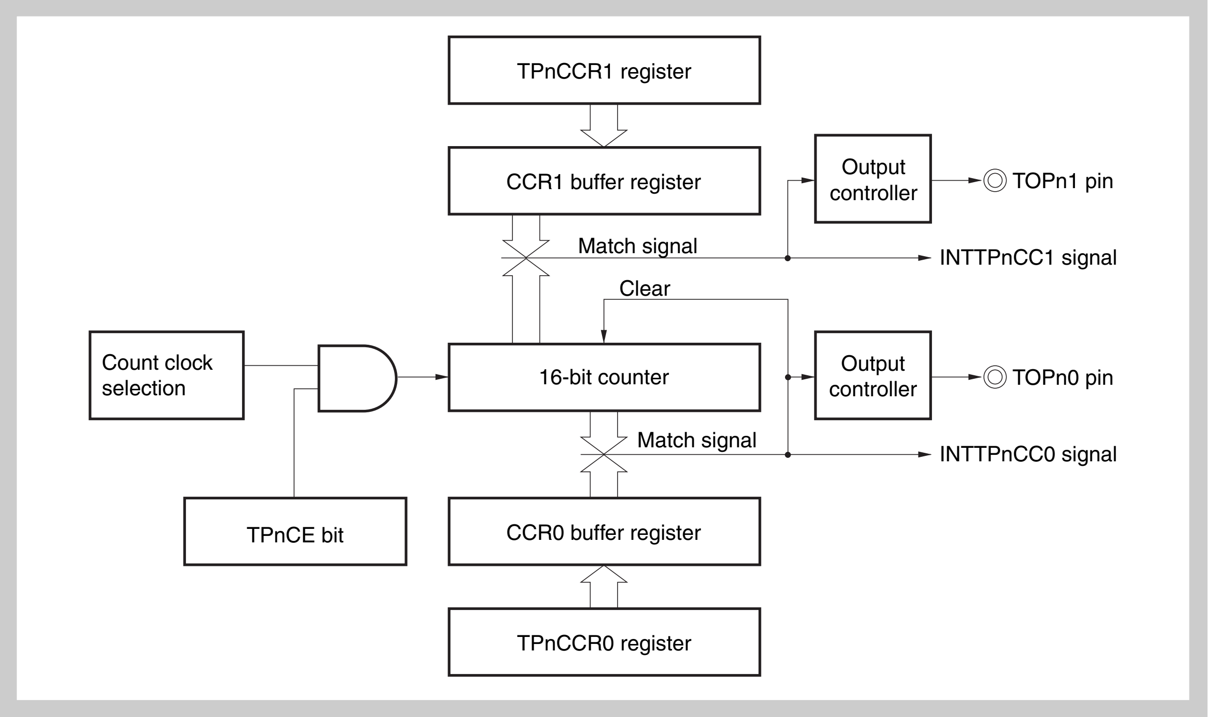


Figure 10-5 Configuration of TPnCCR1 register

If the set value of the TPnCCR1 register is less than the set value of the TPnCCR0 register, the INTTPnCC1 signal is generated once per cycle. At the same time, the output of the TOPn1 pin is inverted.

The TOPn1 pin outputs a square wave with the same cycle as that output by the TOPn0 pin.

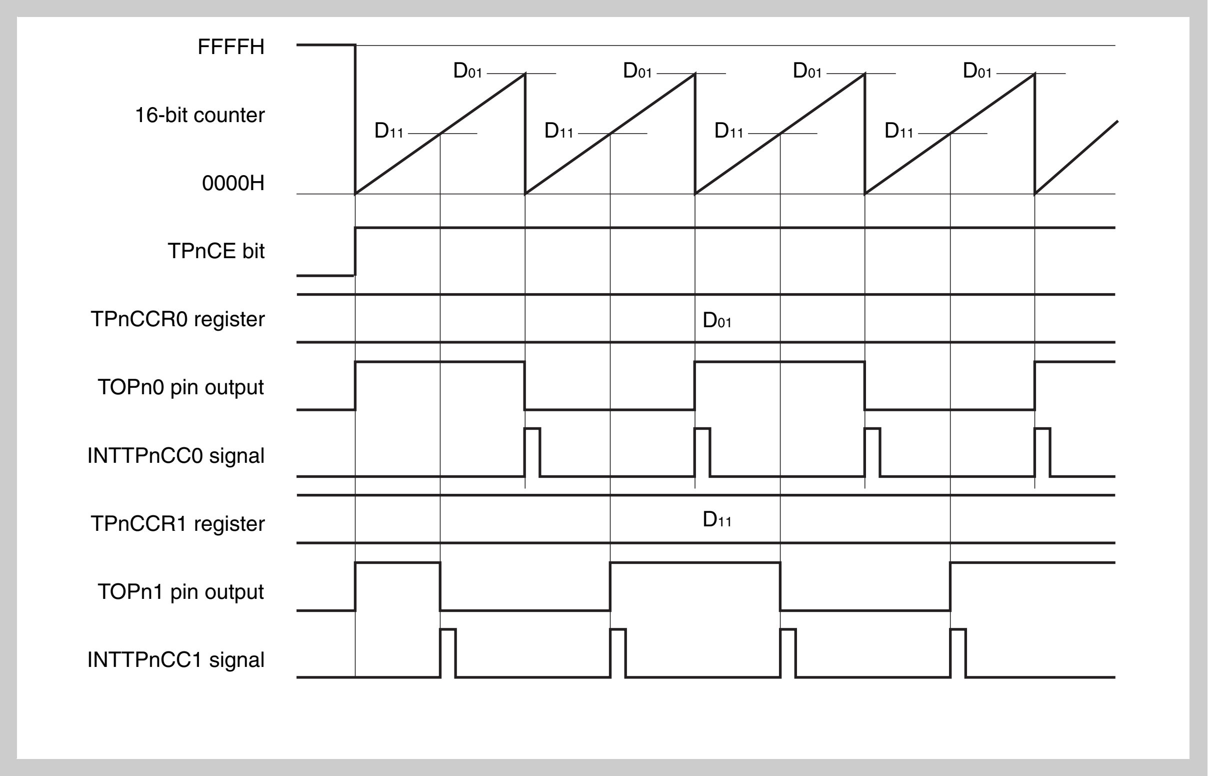


Figure 10-6 Timing chart when $D_{01} \geq D_{11}$

If the set value of the TPnCCR1 register is greater than the set value of the TPnCCR0 register, the count value of the 16-bit counter does not match the value of the TPnCCR1 register. Consequently, the INTTPnCC1 signal is not generated, nor is the output of the TOPn1 pin changed.

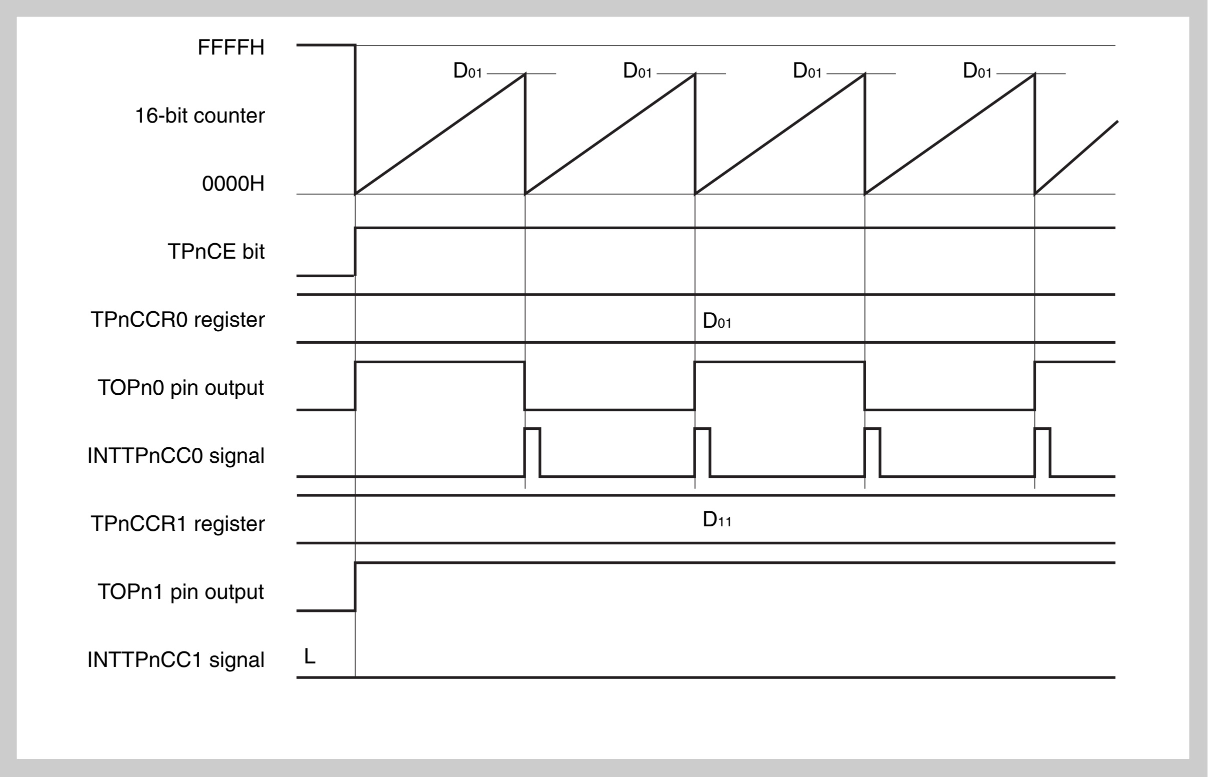


Figure 10-7 Timing chart when $D_{01} < D_{11}$

10.5.2 External event count mode (TPnMD2 to TPnMD0 = 001)

In the external event count mode, the valid edge of the external event count input is counted when the TPnCTL0.TPnCE bit is set to 1, and an interrupt request signal (INTTPnCC0) is generated each time the specified number of edges have been counted. The TOPn0 pin cannot be used.

Usually, the TPnCCR1 register is not used in the external event count mode.

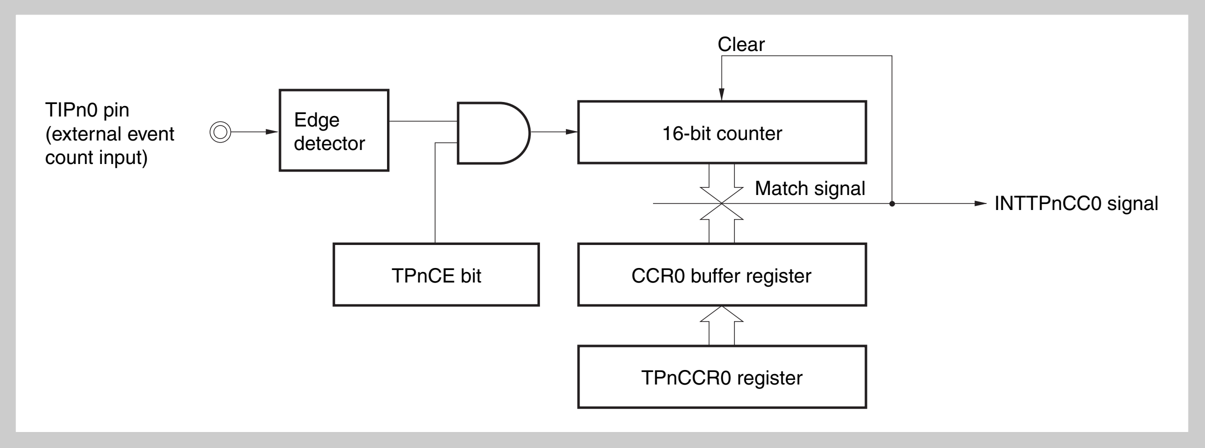


Figure 10-8 Configuration in external event count mode

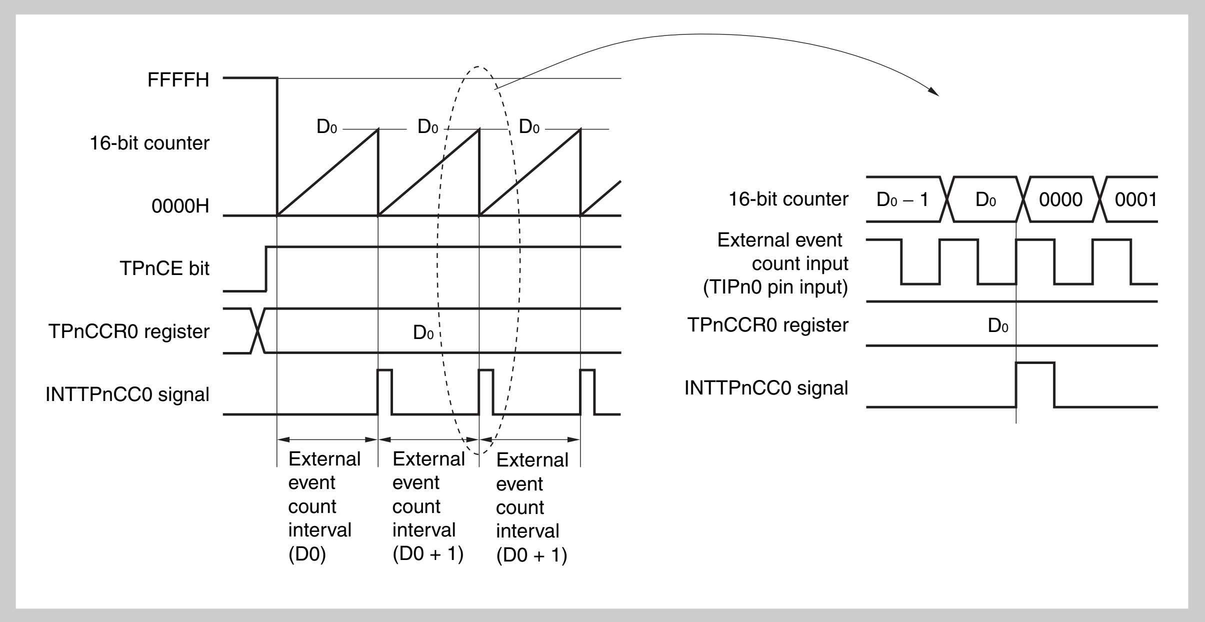


Figure 10-9 Basic timing in external event count mode

Caution This figure shows the basic timing when the rising edge is specified as the valid edge of the external event count input.

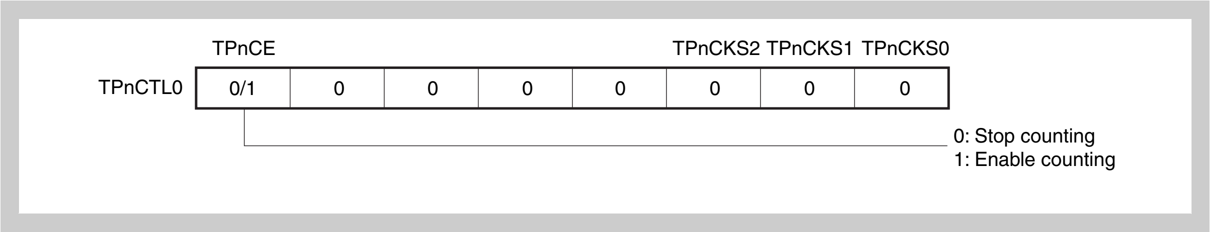
When the TPnCE bit is set to 1, the value of the 16-bit counter is cleared from FFFFH to 0000H. The counter counts each time the valid edge of external event count input is detected. Additionally, the set value of the TPnCCR0 register is transferred to the CCR0 buffer register.

When the count value of the 16-bit counter matches the value of the CCR0 buffer register, the 16-bit counter is cleared to 0000H, and a compare match interrupt request signal (INTTPnCC0) is generated.

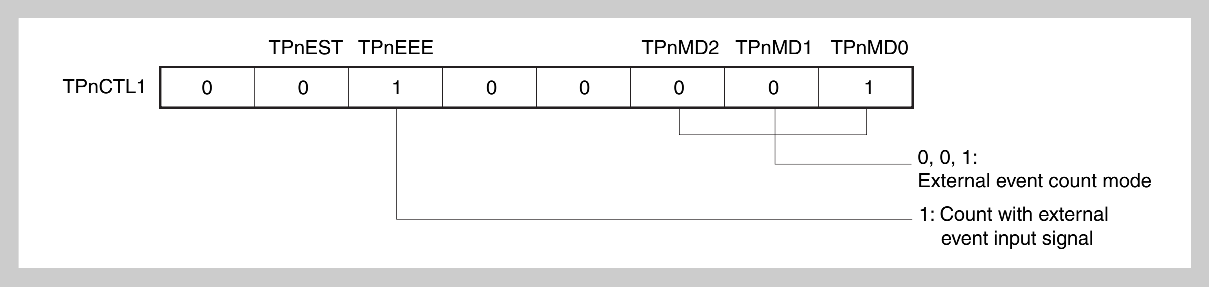
The INTTPnCC0 signal is generated each time the valid edge of the external event count input has been detected (set value of TPnCCR0 register + 1) times.

(1) Register setting for operation in external event count mode

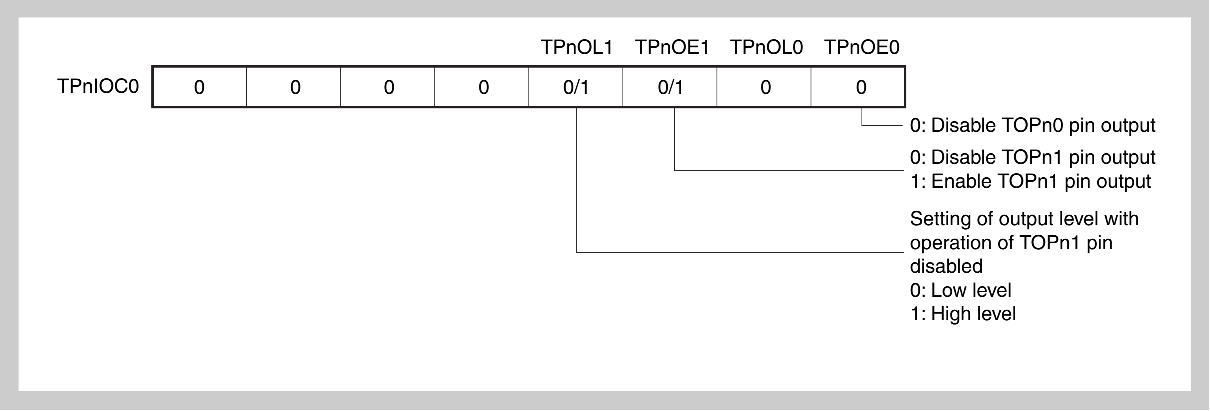
(a) TMPn control register 0 (TPnCTL0)



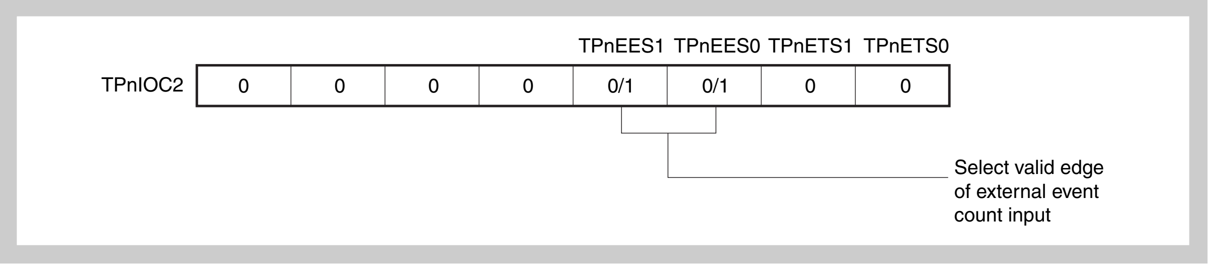
(b) TMPn control register 1 (TPnCTL1)



(c) TMPn I/O control register 0 (TPnIOC0)


TPnOL1
TPnOE1
TPnOL0
TPnOE0
TPnIOC0
0
0
0
0
0/1
0/1
0
0
0: Disable TOPn0 pin output
0: Disable TOPn1 pin output
1: Enable TOPn1 pin output
Setting of output level with
operation of TOPn1 pin
disabled
0: Low level
1: High level

(d) TMPn I/O control register 2 (TPnIOC2)



					TPnEES1	TPnEES0	TPnETS1	TPnETS0
TPnIOC2	0	0	0	0	0/1	0/1	0	0

Select valid edge of external event count input

(e) TMPn counter read buffer register (TPnCNT)

The count value of the 16-bit counter can be read by reading the TPnCNT register.

(f) TMPn capture/compare register 0 (TPnCCR0)

If D_0 is set to the TPnCCR0 register, the counter is cleared and a compare match interrupt request signal (INTTPnCC0) is generated when the number of external event counts reaches (D_0 + 1).

(g) TMPn capture/compare register 1 (TPnCCR1)

Usually, the TPnCCR1 register is not used in the external event count mode. However, the set value of the TPnCCR1 register is transferred to the CCR1 buffer register. When the count value of the 16-bit counter matches the value of the CCR1 buffer register, a compare match interrupt request signal (INTTPnCC1) is generated.

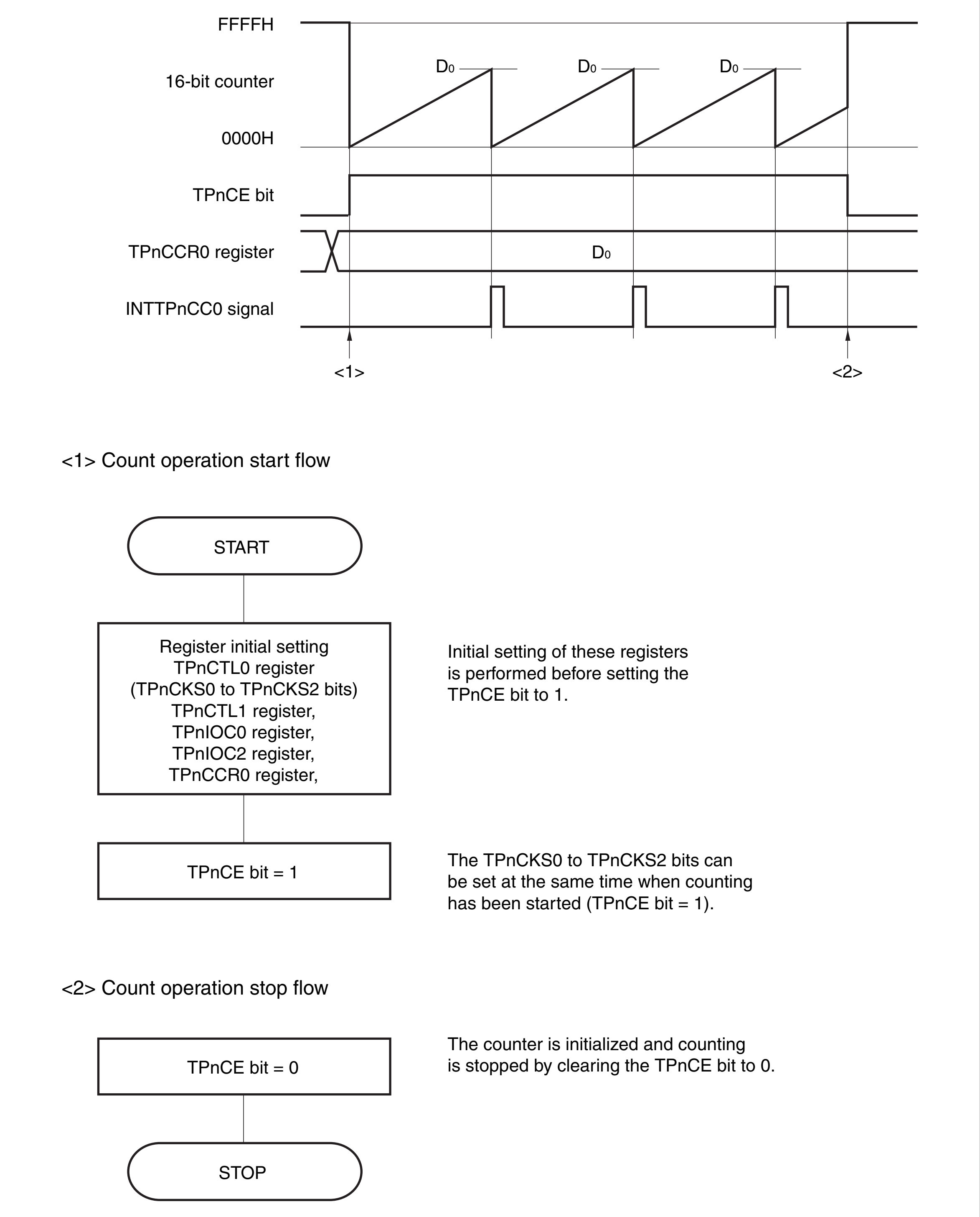
Therefore, mask the interrupt signal by using the interrupt mask flag (TPnCCMK1).

Note TMPn I/O control register 1 (TPnIOC1) and TMPn option register 0 (TPnOPT0) are not used in the external event count mode.

Caution When the compare register TPnCCR0 (TPnCCR1) is set to 0000_H and the external event counter mode is started the first interrupt INTTPnCC0 (INTTPnCC1) occurs upon the first timer overflow (TPnCNT: $FFFF_H \rightarrow 0000_H$), but not with the first external count event.

Afterwards the following interrupts INTTPnCC0 (INTTPnCC1) are generated as specified, i.e. with each external count event.

(2) External event count mode operation flow



<1> Count operation start flow

Initial setting of these registers is performed before setting the TPnCE bit to 1.

The TPnCKS0 to TPnCKS2 bits can be set at the same time when counting has been started (TPnCE bit = 1).

<2> Count operation stop flow

The counter is initialized and counting is stopped by clearing the TPnCE bit to 0.

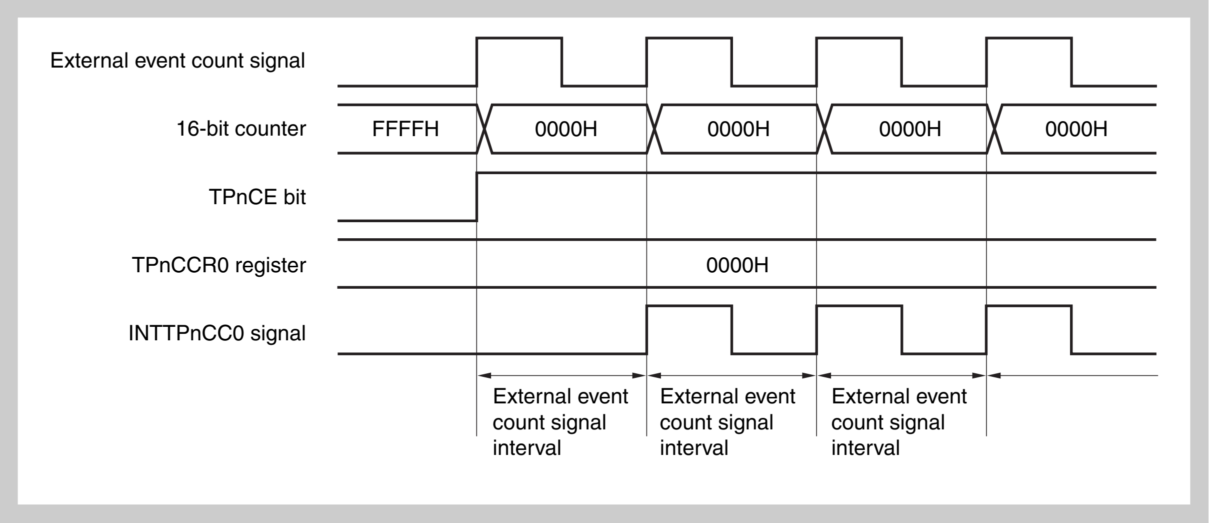
Figure 10-10 Flow of software processing in external event count mode

(3) Operation timing in external event count mode

(a) Operation if TPnCCR0 register is set to 0000H

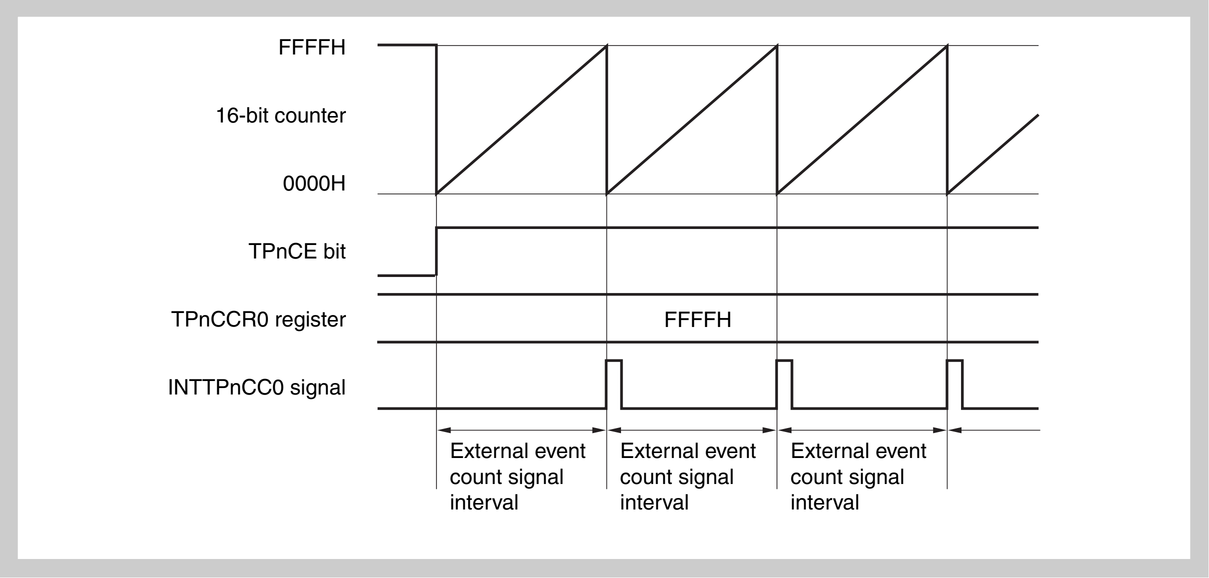
If the TPnCCR0 register is set to 0000H, the INTTPnCC0 signal is generated each time the valid signal of the external event count signal has been detected.

The 16-bit counter is always 0000H.


External event count signal
16-bit counter
FFFFH
0000H
0000H
0000H
0000H
TPnCE bit
TPnCCR0 register
0000H
INTTPnCC0 signal
External event count signal interval
External event count signal interval
External event count signal interval

(b) Operation if TPnCCR0 register is set to FFFFH

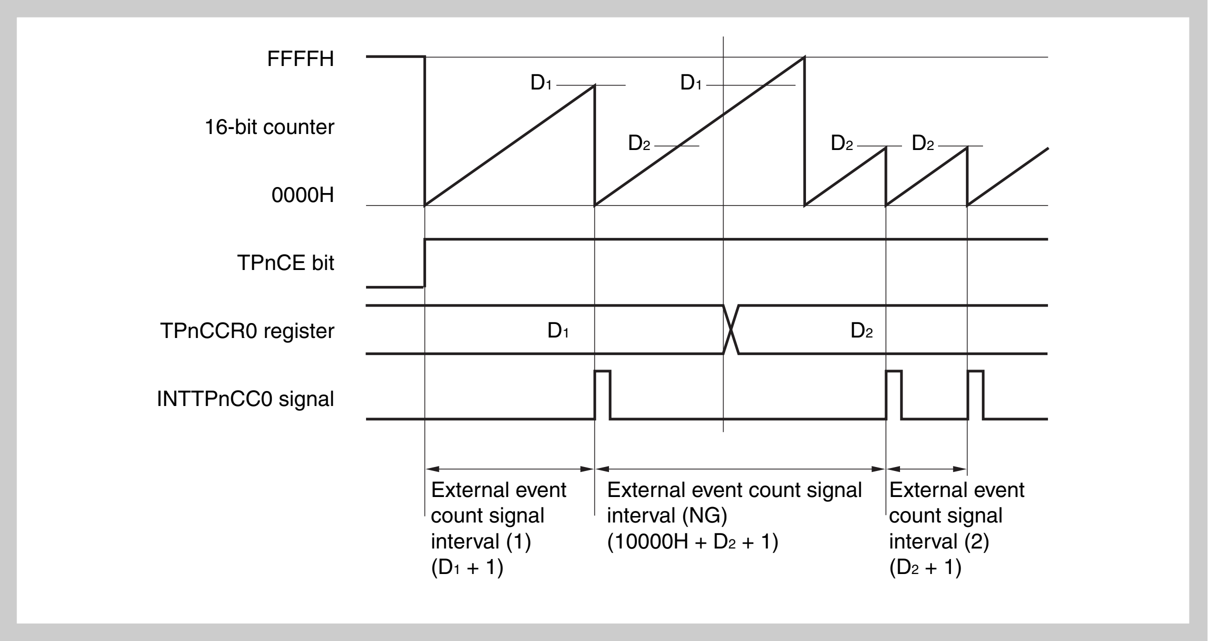
If the TPnCCR0 register is set to FFFFH, the 16-bit counter counts to FFFFH each time the valid edge of the external event count signal has been detected. The 16-bit counter is cleared to 0000H in synchronization with the next count-up timing, and the INTTPnCC0 signal is generated. At this time, the TPnOPT0.TPnOVF bit is not set.


FFFFH
16-bit counter
0000H
TPnCE bit
TPnCCR0 register
FFFFH
INTTPnCC0 signal
External event count signal interval
External event count signal interval
External event count signal interval

(c) Notes on rewriting the TPnCCR0 register

To change the value of the TPnCCR0 register to a smaller value, stop counting once and then change the set value.

If the value of the TPnCCR0 register is rewritten to a smaller value during counting, the 16-bit counter may overflow.


FFFFH
16-bit counter
0000H
TPnCE bit
TPnCCR0 register
INTTPnCC0 signal
D1
D1
D2
D2
D2
D1
D2
External event count signal interval (1) (D1 + 1)
External event count signal interval (NG) (10000H + D2 + 1)
External event count signal interval (2) (D2 + 1)

If the value of the TPnCCR0 register is changed from D_1 to D_2 while the count value is greater than D_2 but less than D_1, the count value is transferred to the CCR0 buffer register as soon as the TPnCCR0 register has been rewritten. Consequently, the value that is compared with the 16-bit counter is D_2.

Because the count value has already exceeded D_2, however, the 16-bit counter counts up to FFFFH, overflows, and then counts up again from 0000H. When the count value matches D_2, the INTTPnCC0 signal is generated.

Therefore, the INTTPnCC0 signal may not be generated at the valid edge count of “(D_1 + 1) times” or “(D_2 + 1) times” originally expected, but may be generated at the valid edge count of “(10000H + D_2 + 1) times”.

(d) Operation of TPnCCR1 register

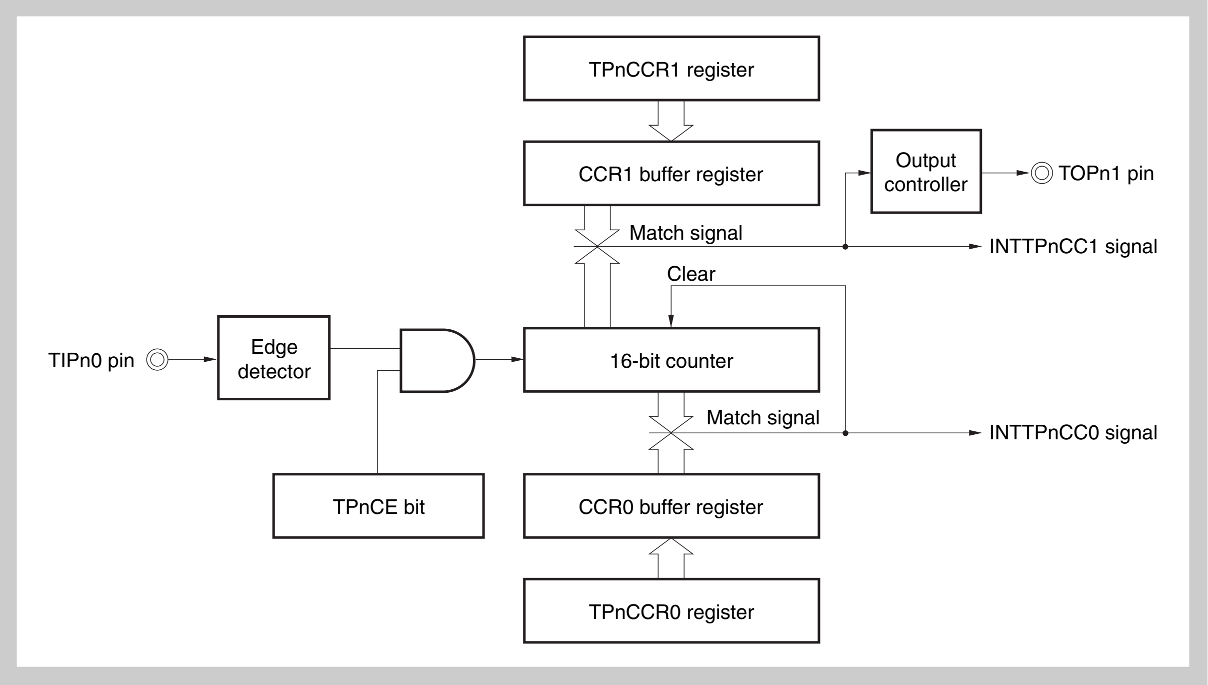


Figure 10-11 Configuration of TPnCCR1 register

If the set value of the TPnCCR1 register is smaller than the set value of the TPnCCR0 register, the INTTPnCC1 signal is generated once per cycle. At the same time, the output signal of the TOPn1 pin is inverted.

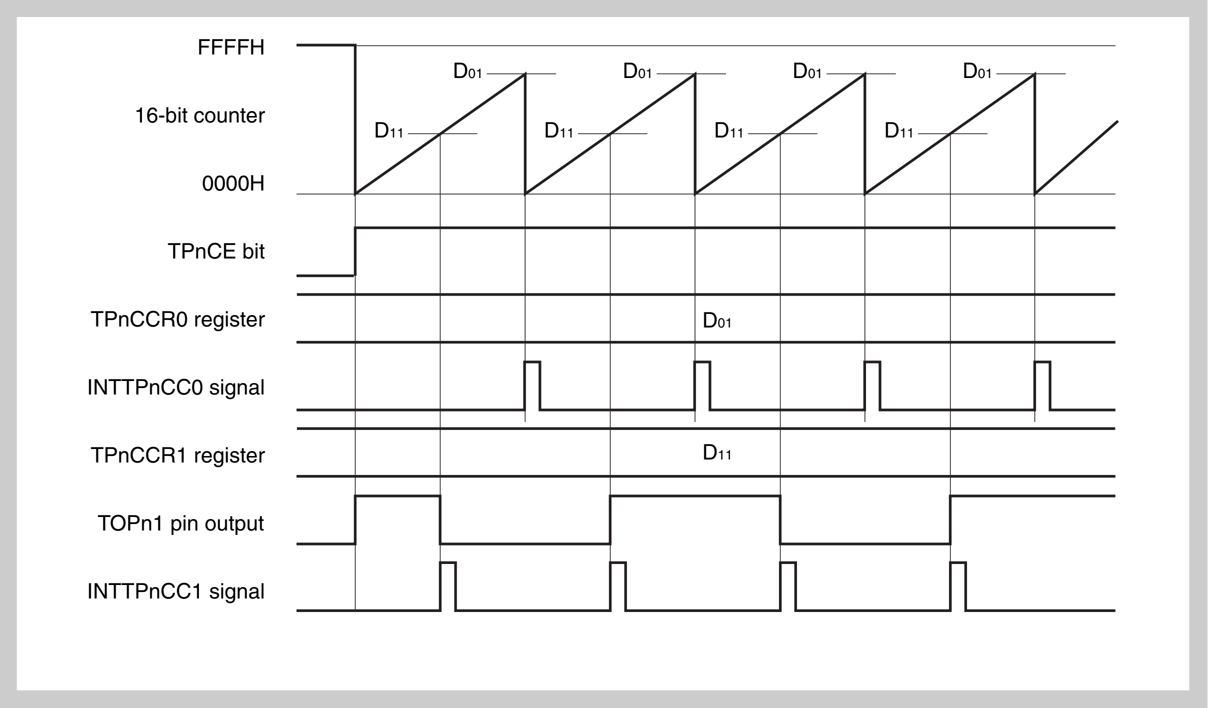


Figure 10-12 Timing chart when $D_{01} \geq D_{11}$

If the set value of the TPnCCR1 register is greater than the set value of the TPnCCR0 register, the INTTPnCC1 signal is not generated because the count value of the 16-bit counter and the value of the TPnCCR1 register do not match. Nor is the output signal of the TOPn1 pin changed.

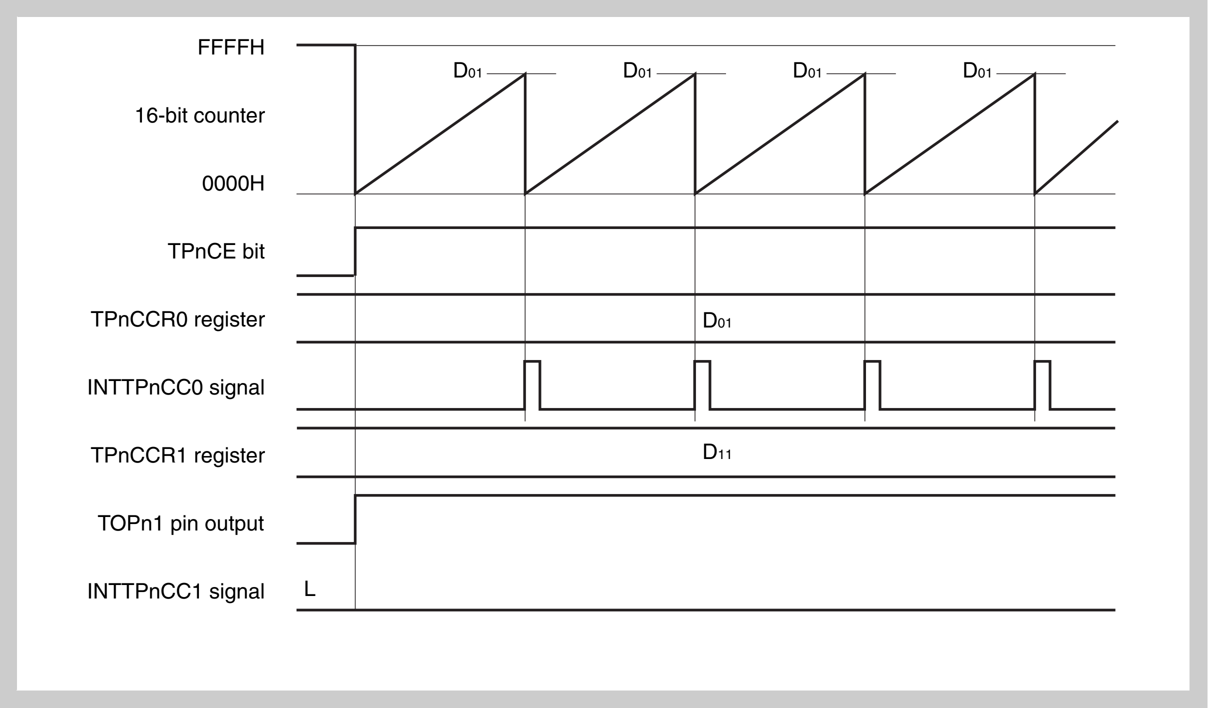


Figure 10-13 Timing chart when $D_{01} < D_{11}$

10.5.3 External trigger pulse output mode (TPnMD2 to TPnMD0 = 010)

In the external trigger pulse output mode, 16-bit timer/event counter P waits for a trigger when the TPnCTL0.TPnCE bit is set to 1. When the valid edge of an external trigger input signal is detected, 16-bit timer/event counter P starts counting, and outputs a PWM waveform from the TOPn1 pin.

Pulses can also be output by generating a software trigger instead of using the external trigger. When using a software trigger, a square wave that has one cycle of the PWM waveform as half its cycle can also be output from the TOPn0 pin.

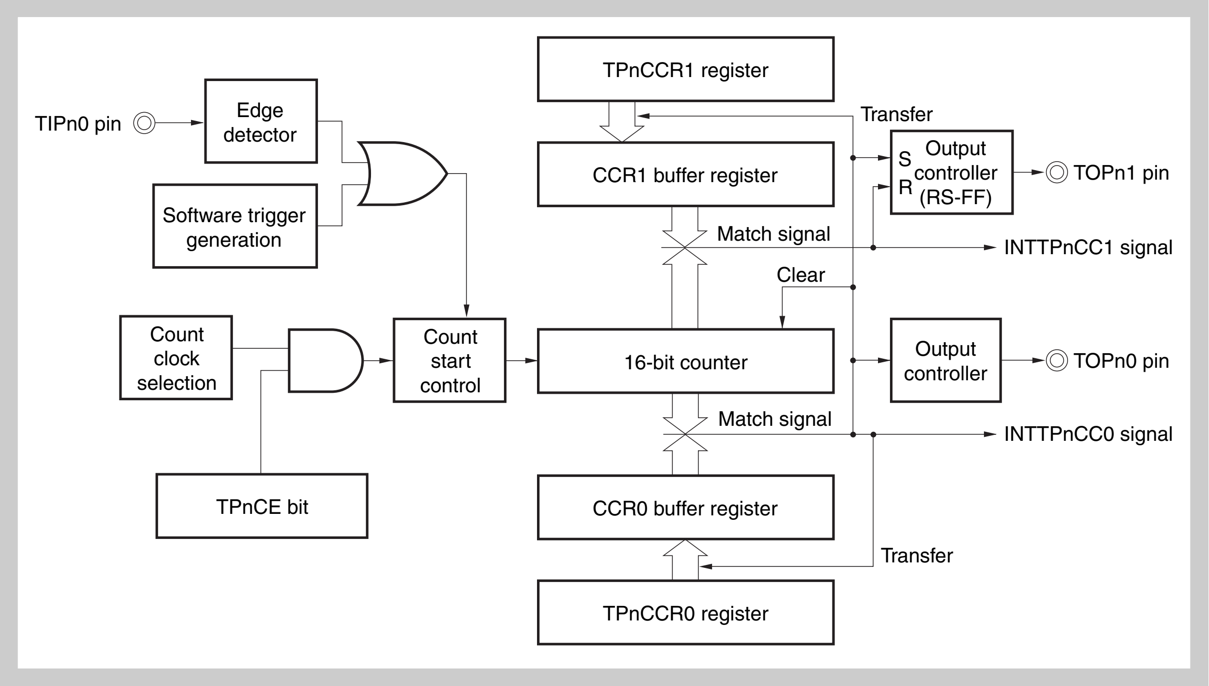


Figure 10-14 Configuration in external trigger pulse output mode

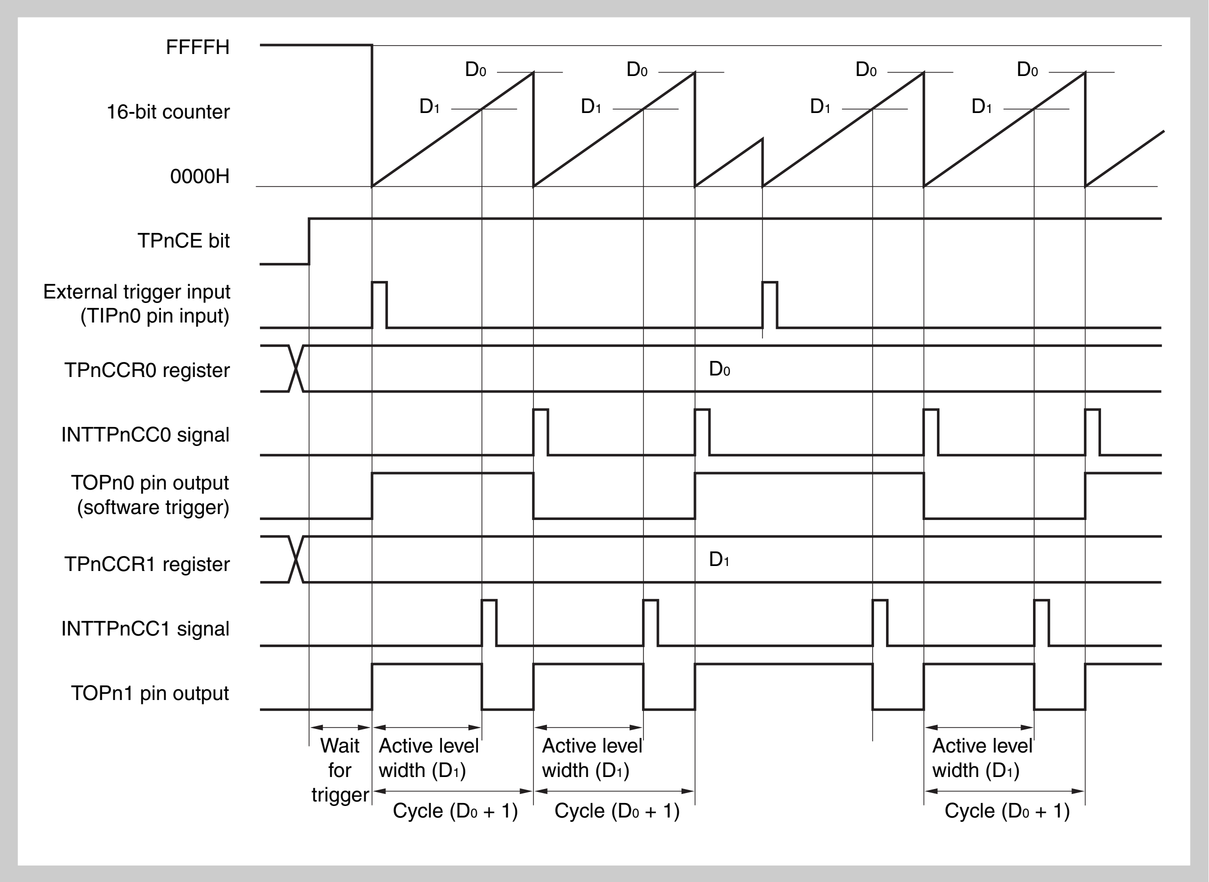


Figure 10-15 Basic timing in external trigger pulse output mode

16-bit timer/event counter P waits for a trigger when the TPnCE bit is set to 1. When the trigger is generated, the 16-bit counter is cleared from FFFFH to 0000H, starts counting at the same time, and outputs a PWM waveform from the TOPn1 pin.

If the trigger is generated again while the counter is operating, the counter is cleared to 0000H and restarted.

The active level width, cycle, and duty factor of the PWM waveform can be calculated as follows.

Active level width = (Set value of TPnCCR1 register) × Count clock cycle

Cycle = (Set value of TPnCCR0 register + 1) × Count clock cycle

Duty factor = (Set value of TPnCCR1 register)/(Set value of TPnCCR0 register + 1)

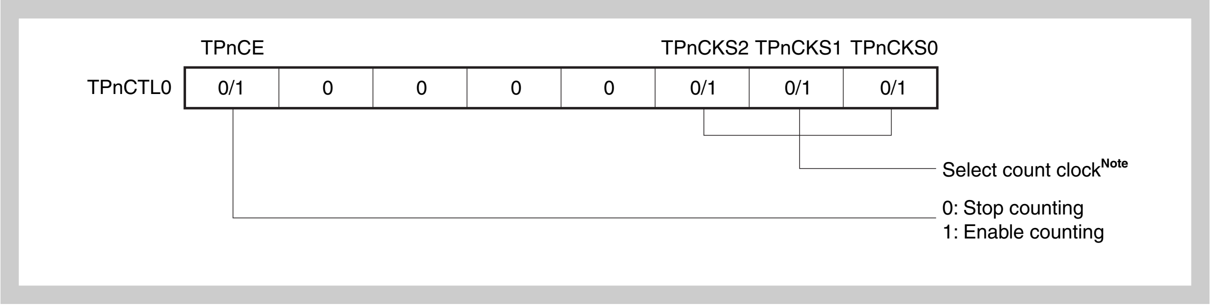
The compare match request signal INTTPnCC0 is generated when the 16-bit counter counts next time after its count value matches the value of the CCR0 buffer register, and the 16-bit counter is cleared to 0000H. The compare match interrupt request signal INTTPnCC1 is generated when the count value of the 16-bit counter matches the value of the CCR1 buffer register.

The value set to the TPnCCRm register is transferred to the CCRm buffer register when the count value of the 16-bit counter matches the value of the CCRm buffer register and the 16-bit counter is cleared to 0000H.

The valid edge of an external trigger input signal, or setting the software trigger (TPnCTL1.TPnEST bit) to 1 is used as the trigger.

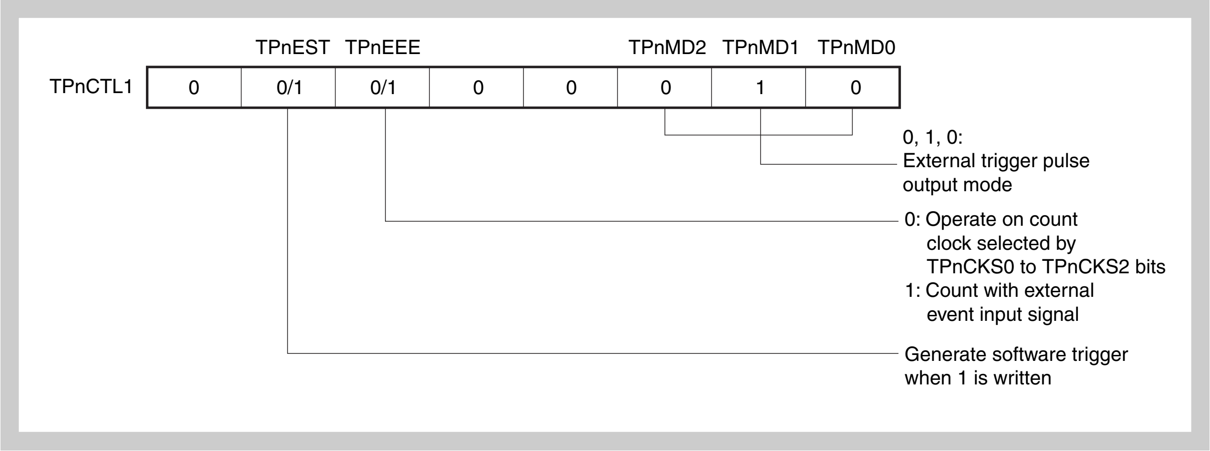
(1) Setting of registers in external trigger pulse output mode

(a) TMPn control register 0 (TPnCTL0)

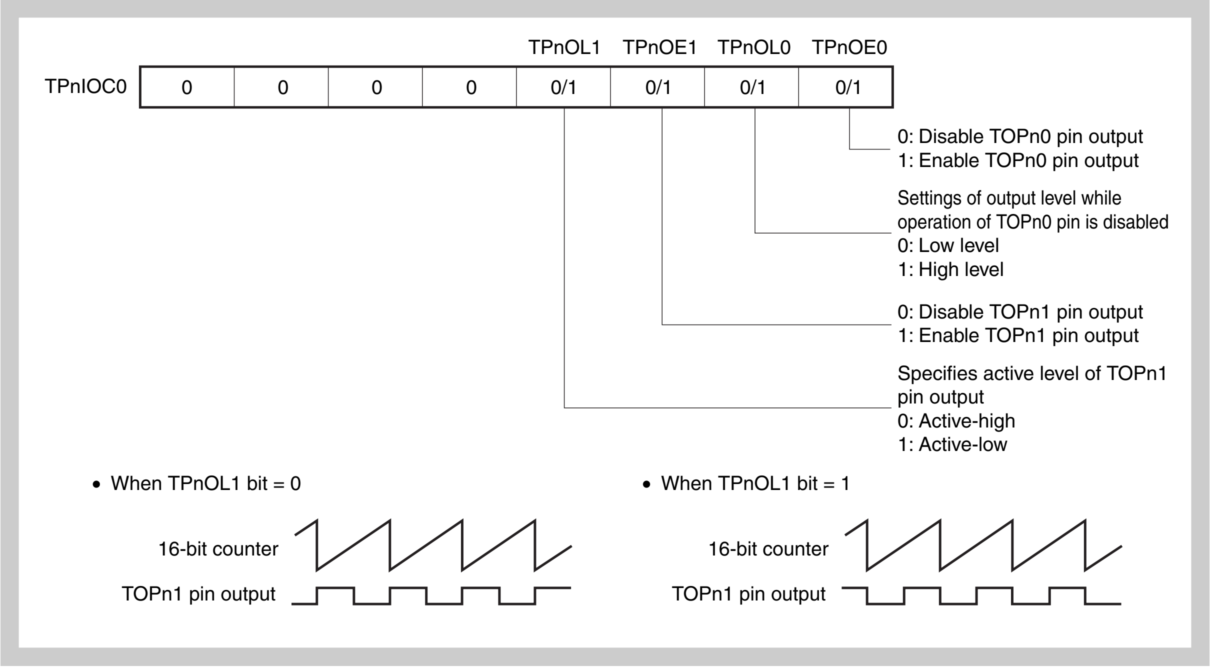


Note The setting is invalid when the TPnCTL1.TPnEEE bit = 1.

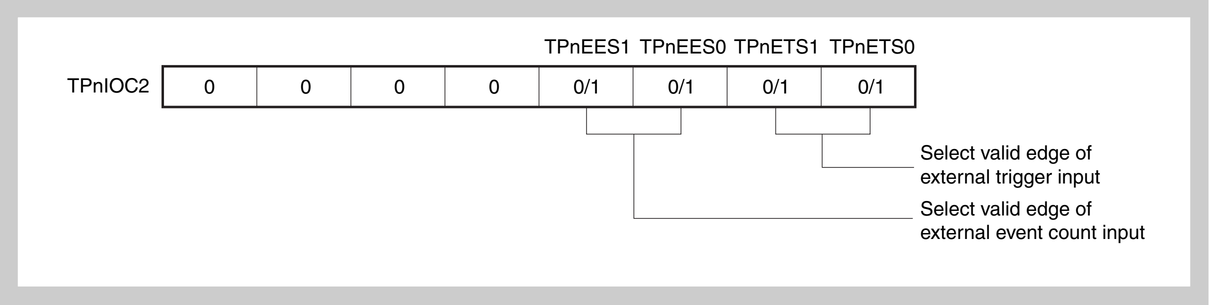
(b) TMPn control register 1 (TPnCTL1)



(c) TMPn I/O control register 0 (TPnIOC0)


TPnOL1 TPnOE1 TPnOL0 TPnOE0
TPnIOC0
0 0 0 0 0/1 0/1 0/1 0/1
0: Disable TOPn0 pin output
1: Enable TOPn0 pin output
Settings of output level while
operation of TOPn0 pin is disabled
0: Low level
1: High level
0: Disable TOPn1 pin output
1: Enable TOPn1 pin output
Specifies active level of TOPn1
pin output
0: Active-high
1: Active-low
• When TPnOL1 bit = 0
16-bit counter
TOPn1 pin output
• When TPnOL1 bit = 1
16-bit counter
TOPn1 pin output

(d) TMPn I/O control register 2 (TPnIOC2)



					TPnEES1	TPnEES0	TPnETS1	TPnETS0
TPnIOC2	0	0	0	0	0/1	0/1	0/1	0/1

Select valid edge of external trigger input
Select valid edge of external event count input

(e) TMPn counter read buffer register (TPnCNT)

The value of the 16-bit counter can be read by reading the TPnCNT register.

(f) TMPn capture/compare registers 0 and 1 (TPnCCR0 and TPnCCR1)

If D_0 is set to the TPnCCR0 register and D_1 to the TPnCCR1 register, the cycle and active level of the PWM waveform are as follows.

Cycle = (D_0 + 1) × Count clock cycle

Active level width = D_1 × Count clock cycle

Note TMPn I/O control register 1 (TPnIOC1) and TMPn option register 0 (TPnOPT0) are not used in the external trigger pulse output mode.

(2) Operation flow in external trigger pulse output mode

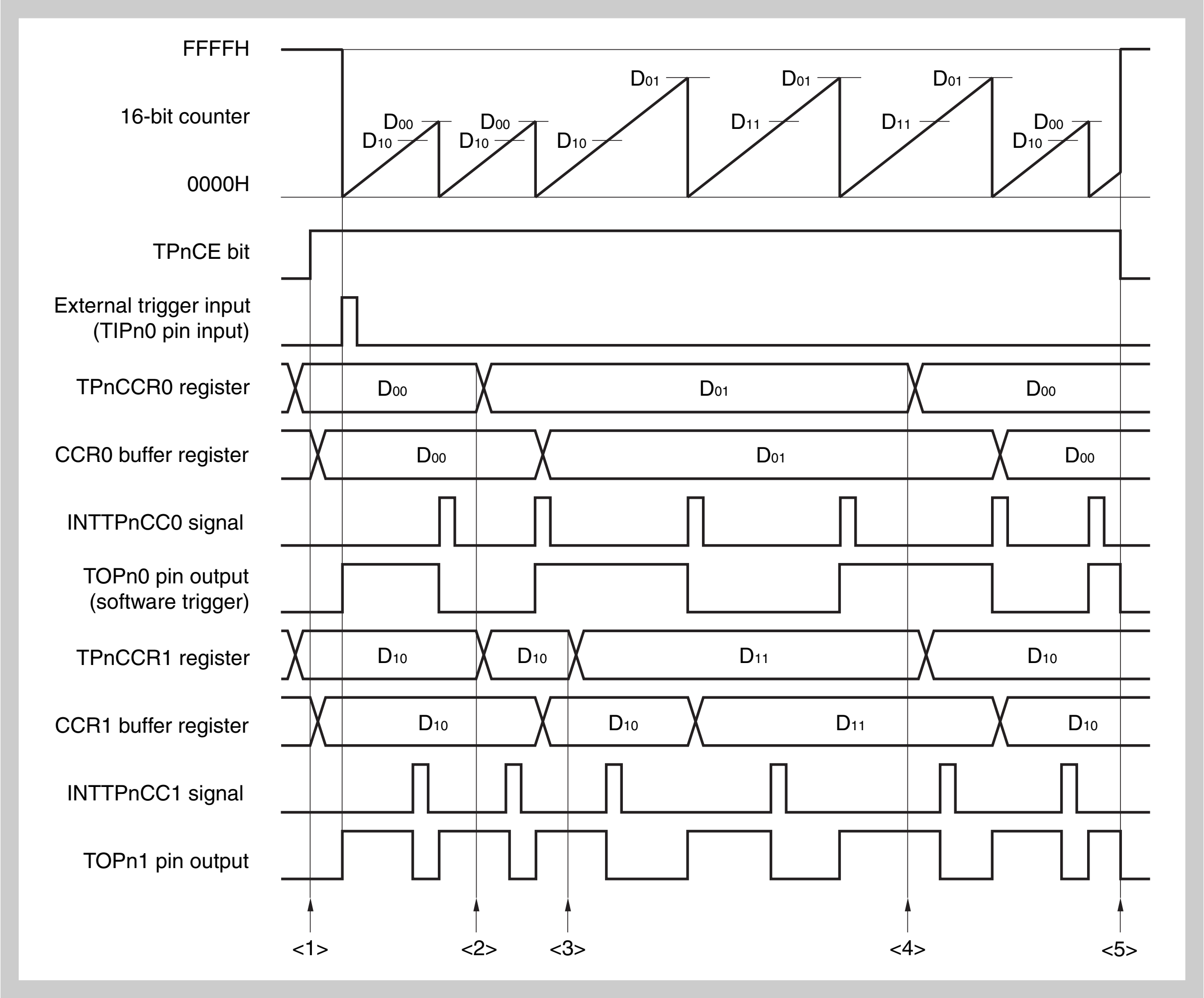


Figure 10-16 Software processing flow in external trigger pulse output mode (1/2)

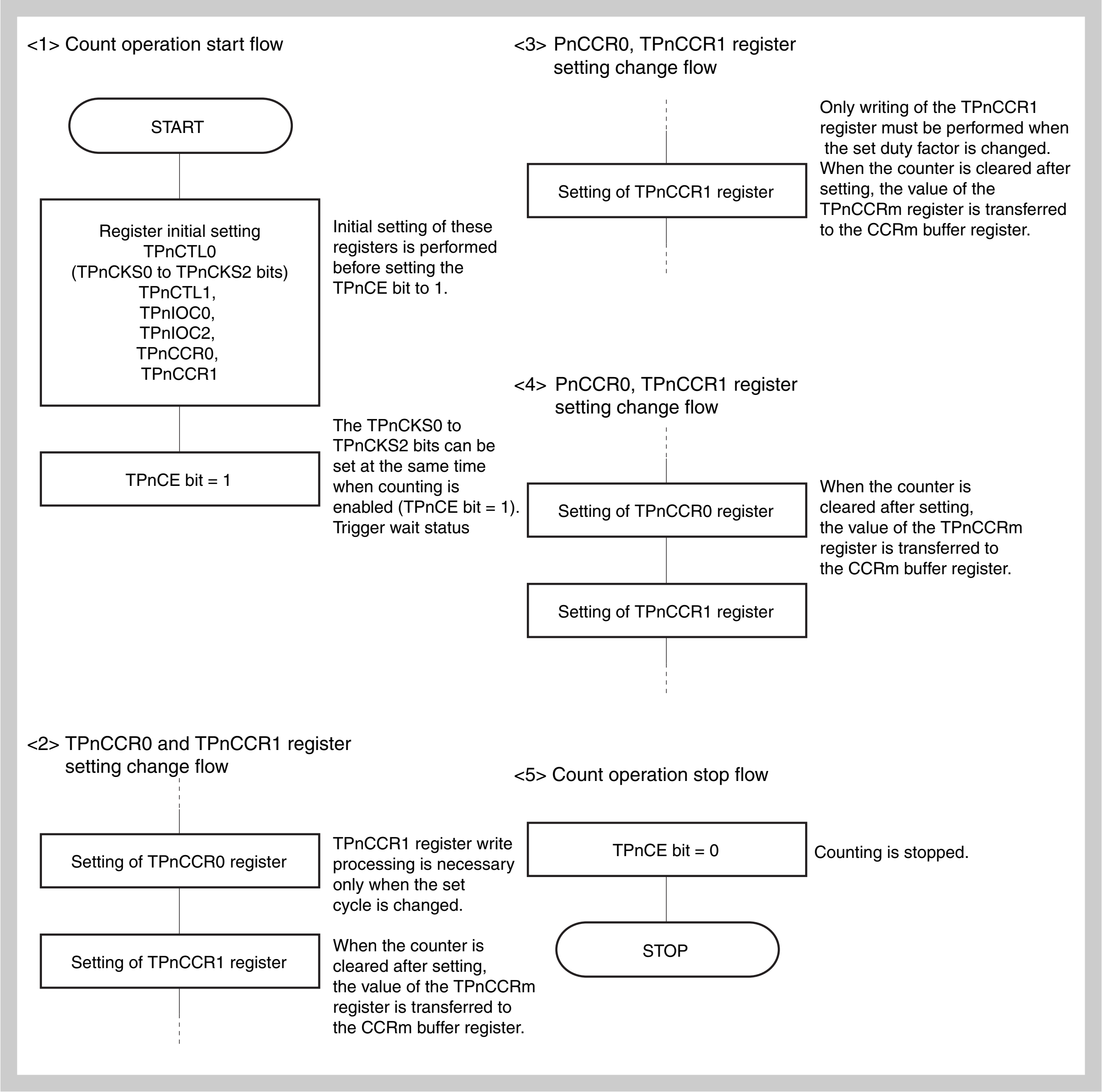

<1> Count operation start flow
START
Register initial setting
TPnCTL0
(TPnCKS0 to TPnCKS2 bits)
TPnCTL1,
TPnIOC0,
TPnIOC2,
TPnCCR0,
TPnCCR1
Initial setting of these registers is performed before setting the TPnCE bit to 1.
TPnCE bit = 1
The TPnCKS0 to TPnCKS2 bits can be set at the same time when counting is enabled (TPnCE bit = 1).
Trigger wait status
<2> TPnCCR0 and TPnCCR1 register setting change flow
Setting of TPnCCR0 register
TPnCCR1 register write processing is necessary only when the set cycle is changed.
Setting of TPnCCR1 register
When the counter is cleared after setting, the value of the TPnCCRm register is transferred to the CCRm buffer register.
<3> PnCCR0, TPnCCR1 register setting change flow
Setting of TPnCCR1 register
Only writing of the TPnCCR1 register must be performed when the set duty factor is changed.
When the counter is cleared after setting, the value of the TPnCCRm register is transferred to the CCRm buffer register.
<4> PnCCR0, TPnCCR1 register setting change flow
Setting of TPnCCR0 register
When the counter is cleared after setting, the value of the TPnCCRm register is transferred to the CCRm buffer register.
Setting of TPnCCR1 register
<5> Count operation stop flow
TPnCE bit = 0
Counting is stopped.
STOP

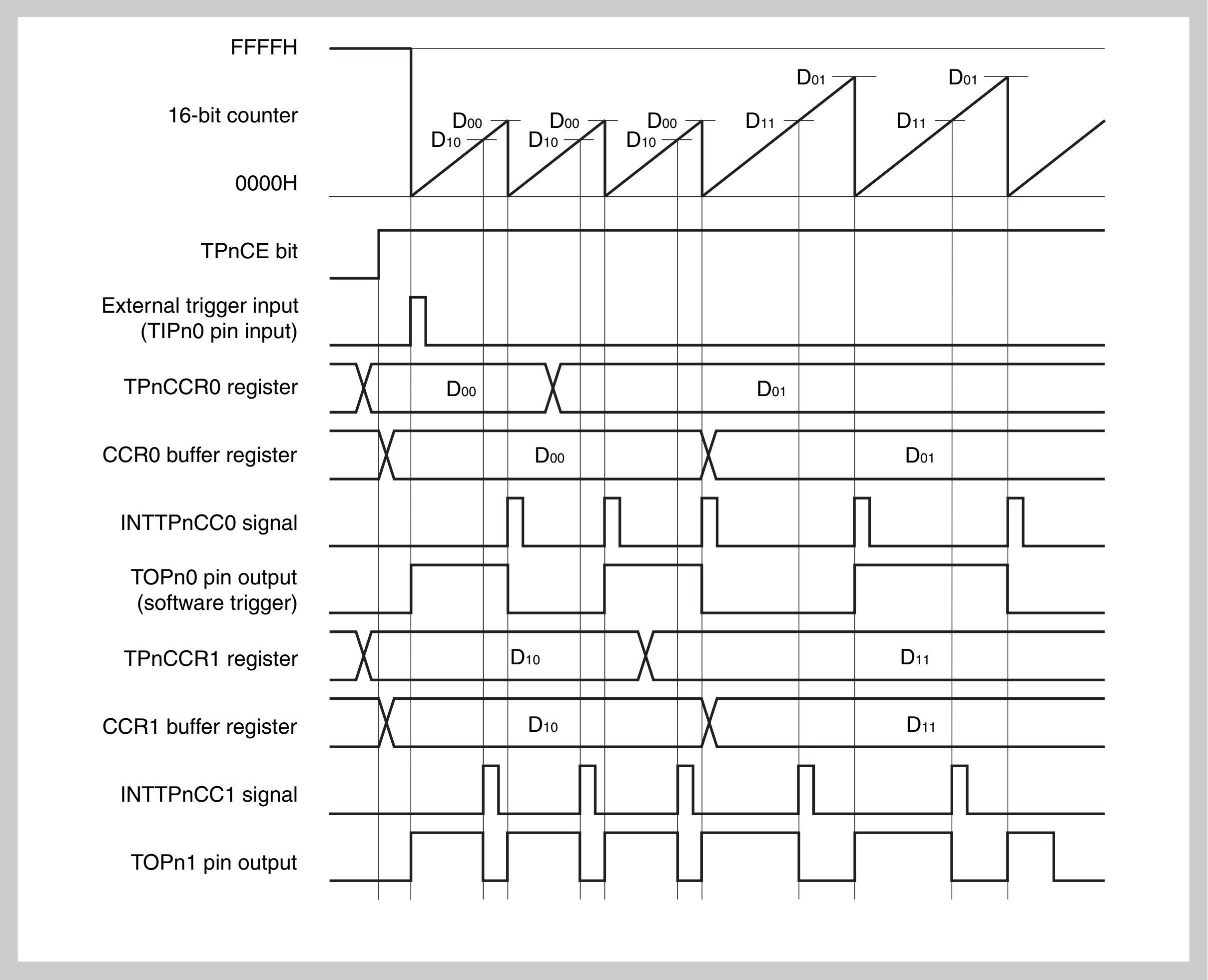
Figure 10-17 Software processing flow in external trigger pulse output mode (2/2)

(3) External trigger pulse output mode operation timing

(a) Note on changing pulse width during operation

To change the PWM waveform while the counter is operating, write the TPnCCR1 register last.

Rewrite the TPnCCRm register after writing the TPnCCR1 register after the INTTPnCC0 signal is detected.



In order to transfer data from the TPnCCRm register to the CCRm buffer register, the TPnCCR1 register must be written.

To change both the cycle and active level width of the PWM waveform at this time, first set the cycle to the TPnCCR0 register and then set the active level width to the TPnCCR1 register.

To change only the cycle of the PWM waveform, first set the cycle to the TPnCCR0 register, and then write the same value to the TPnCCR1 register.

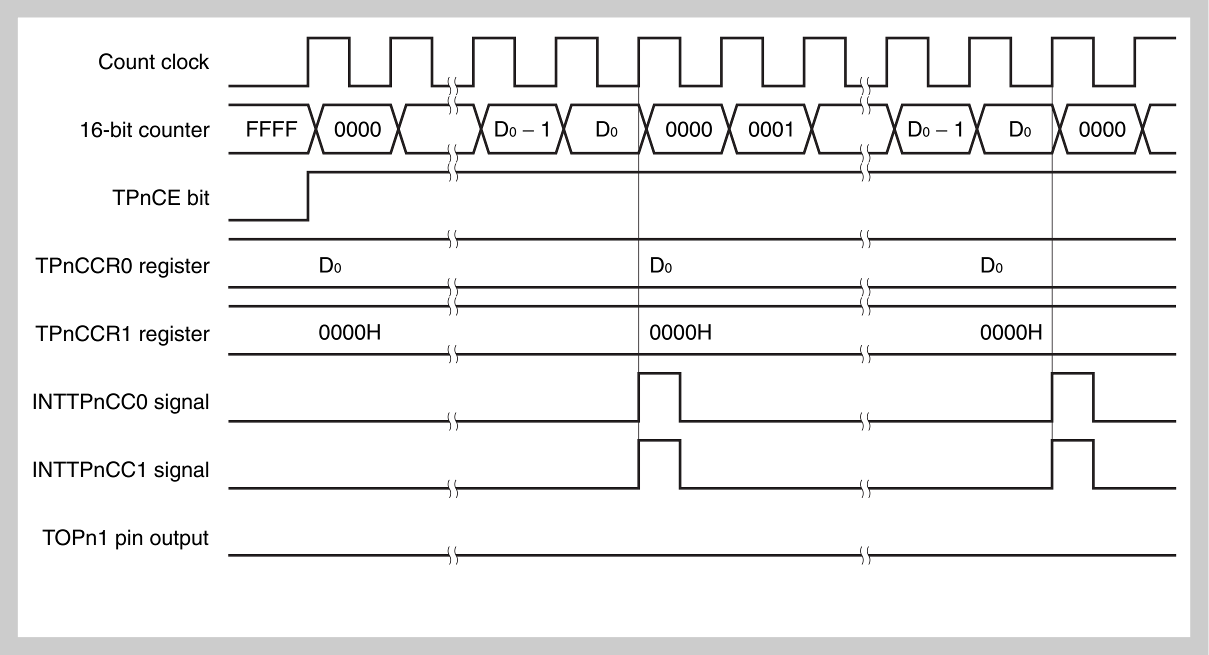
To change only the active level width (duty factor) of the PWM waveform, only the TPnCCR1 register has to be set.

After data is written to the TPnCCR1 register, the value written to the TPnCCRm register is transferred to the CCRm buffer register in synchronization with clearing of the 16-bit counter, and is used as the value compared with the 16-bit counter.

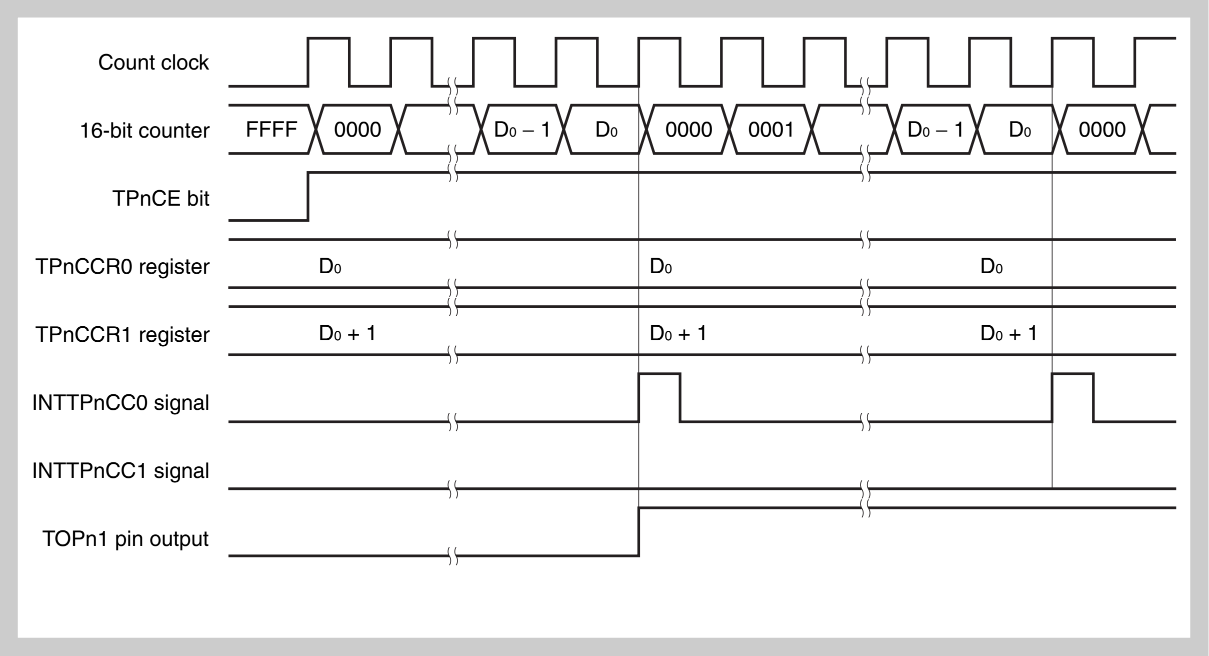
To write the TPnCCR0 or TPnCCR1 register again after writing the TPnCCR1 register once, do so after the INTTPnCC0 signal is generated. Otherwise, the value of the CCRm buffer register may become undefined because the timing of transferring data from the TPnCCRm register to the CCRm buffer register conflicts with writing the TPnCCRm register.

(b) 0%/100% output of PWM waveform

To output a 0% waveform, set the TPnCCR1 register to 0000H. If the set value of the TPnCCR0 register is FFFFH, the INTTPnCC1 signal is generated periodically.

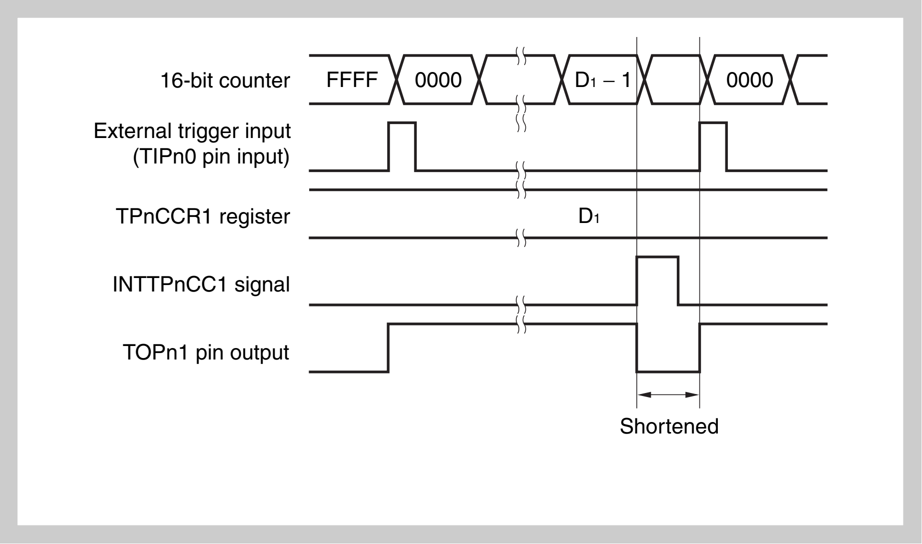


To output a 100% waveform, set a value of (set value of TPnCCR0 register + 1) to the TPnCCR1 register. If the set value of the TPnCCR0 register is FFFFH, 100% output cannot be produced.

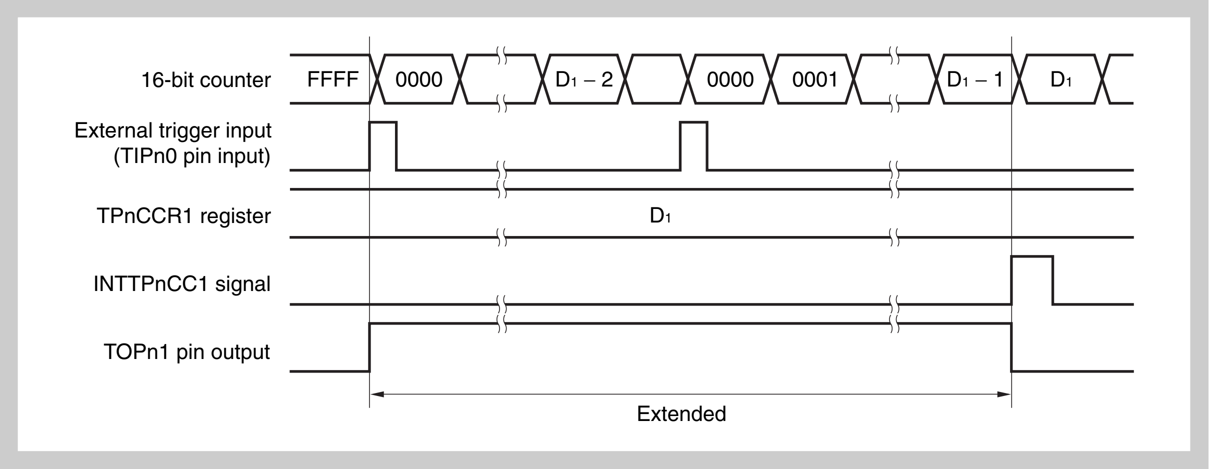


(c) Conflict between trigger detection and match with TPnCCR1 register

If the trigger is detected immediately after the INTTPnCC1 signal is generated, the 16-bit counter is immediately cleared to 0000H, the output signal of the TOPn1 pin is asserted, and the counter continues counting. Consequently, the inactive period of the PWM waveform is shortened.

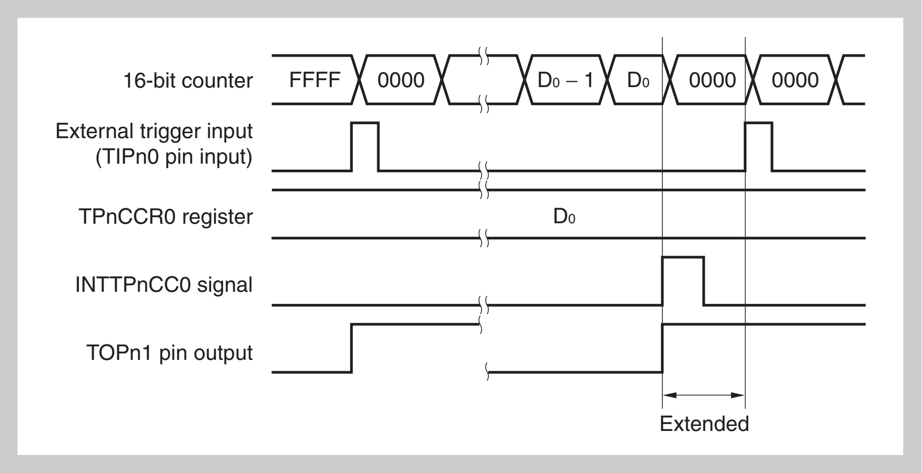


If the trigger is detected immediately before the INTTPnCC1 signal is generated, the INTTPnCC1 signal is not generated, and the 16-bit counter is cleared to 0000H and continues counting. The output signal of the TOPn1 pin remains active. Consequently, the active period of the PWM waveform is extended.

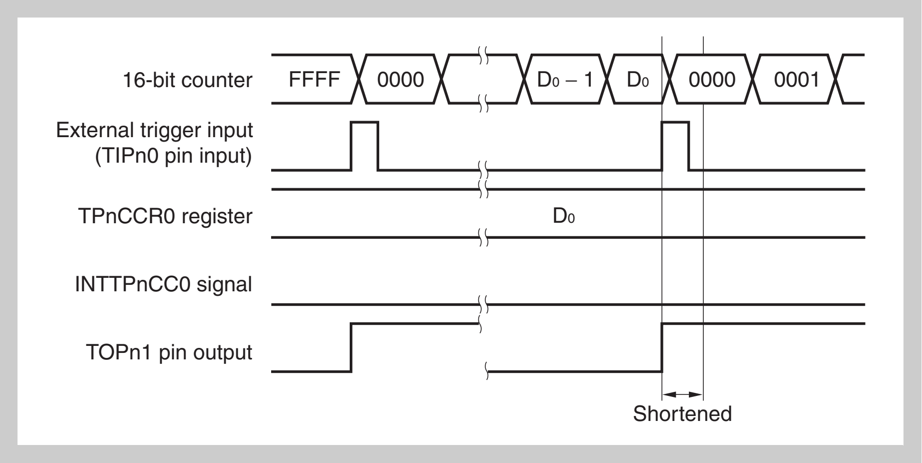


(d) Conflict between trigger detection and match with TPnCCR0 register

If the trigger is detected immediately after the INTTPnCC0 signal is generated, the 16-bit counter is cleared to 0000H and continues counting up. Therefore, the active period of the TOPn1 pin is extended by time from generation of the INTTPnCC0 signal to trigger detection.

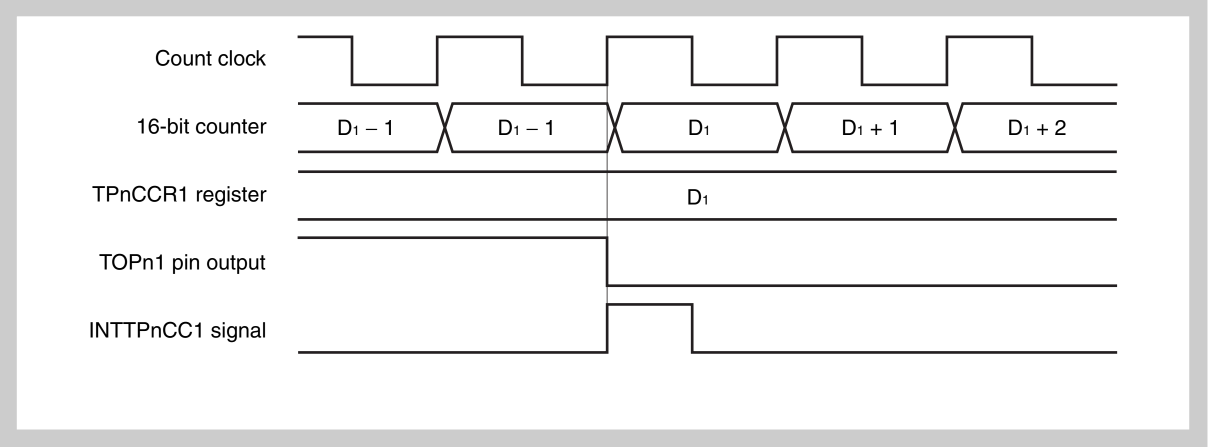


If the trigger is detected immediately before the INTTPnCC0 signal is generated, the INTTPnCC0 signal is not generated. The 16-bit counter is cleared to 0000H, the TOPn1 pin is asserted, and the counter continues counting. Consequently, the inactive period of the PWM waveform is shortened.



(e) Generation timing of compare match interrupt request signal (INTTPnCC1)

The timing of generation of the INTTPnCC1 signal in the external trigger pulse output mode differs from the timing of other INTTPnCC1 signals; the INTTPnCC1 signal is generated when the count value of the 16-bit counter matches the value of the TPnCCR1 register.



Usually, the INTTPnCC1 signal is generated in synchronization with the next count up, after the count value of the 16-bit counter matches the value of the TPnCCR1 register.

In the external trigger pulse output mode, however, it is generated one clock earlier. This is because the timing is changed to match the timing of changing the output signal of the TOPn1 pin.

10.5.4 One-shot pulse output mode (TPnMD2 to TPnMD0 = 011)

In the one-shot pulse output mode, 16-bit timer/event counter P waits for a trigger when the TPnCTL0.TPnCE bit is set to 1. When the valid edge of an external trigger input is detected, 16-bit timer/event counter P starts counting, and outputs a one-shot pulse from the TOPn1 pin.

Instead of the external trigger, a software trigger can also be generated to output the pulse. When the software trigger is used, the TOPn0 pin outputs the active level while the 16-bit counter is counting, and the inactive level when the counter is stopped (waiting for a trigger).

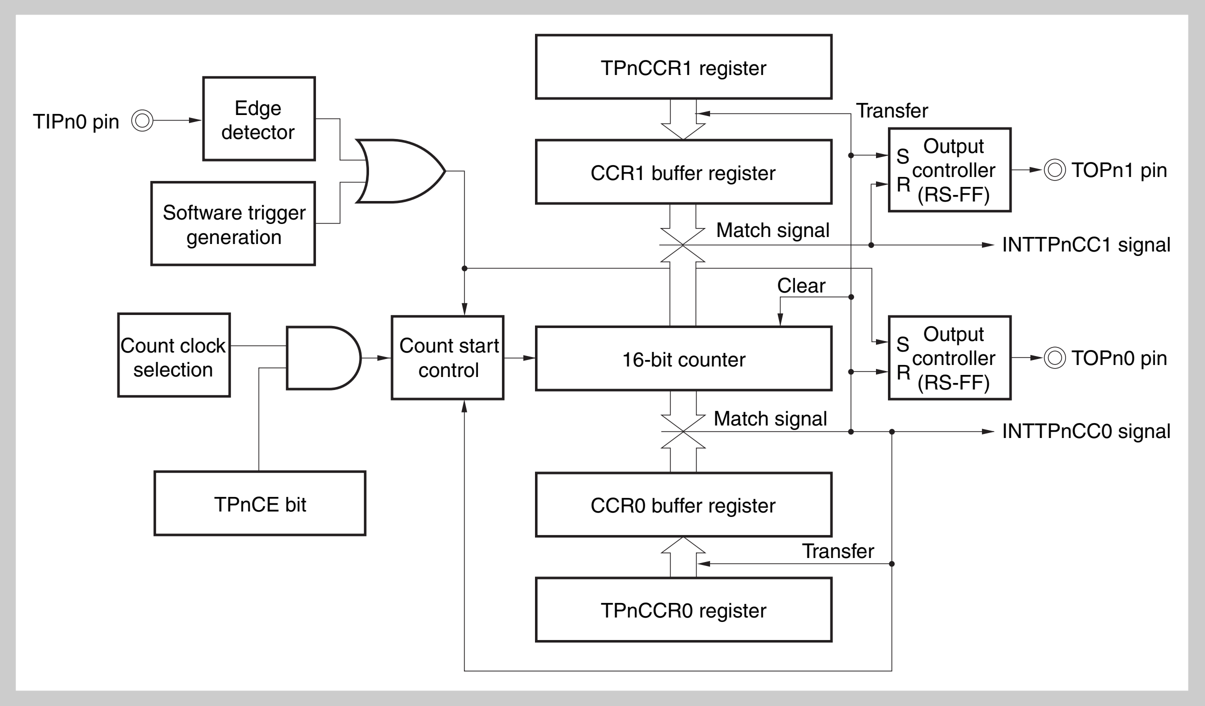


Figure 10-18 Configuration in one-shot pulse output mode

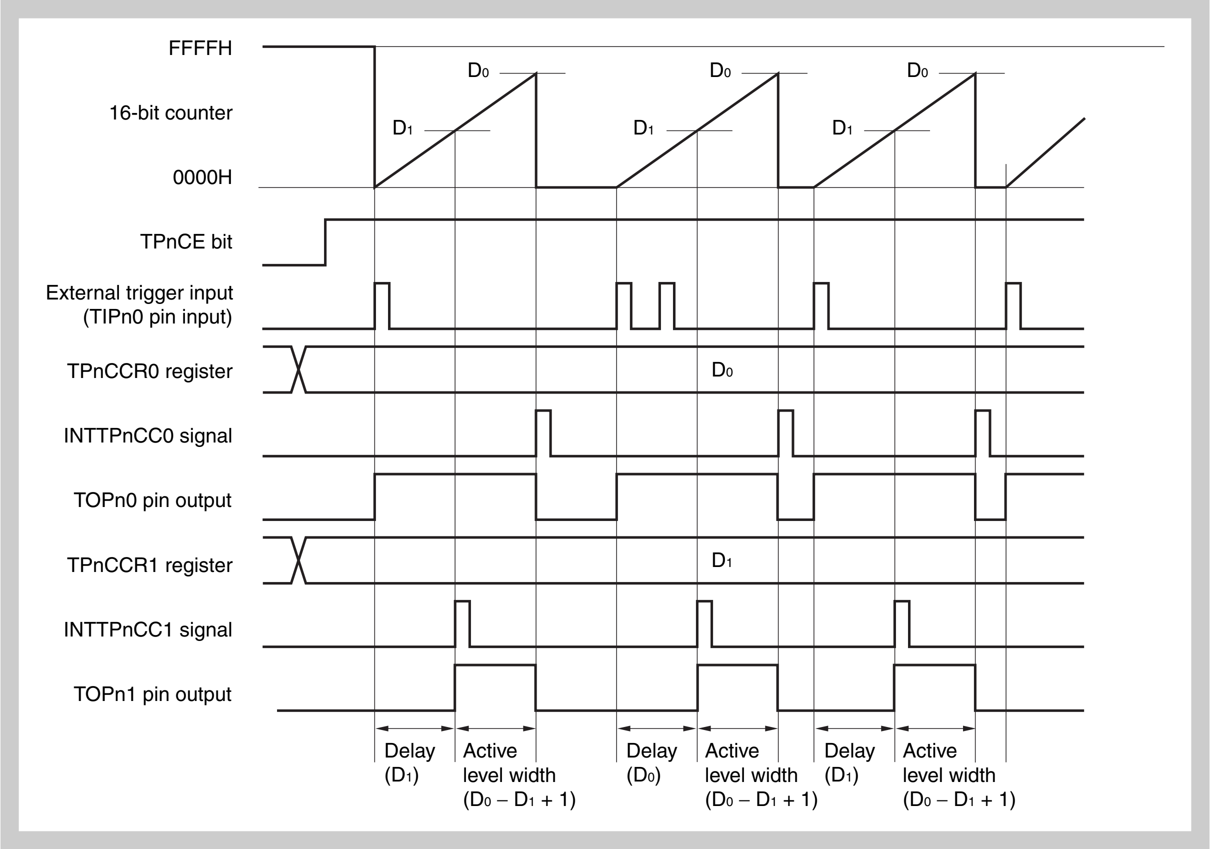


Figure 10-19 Basic timing in one-shot pulse output mode

When the TPnCE bit is set to 1, 16-bit timer/event counter P waits for a trigger. When the trigger is generated, the 16-bit counter is cleared from FFFFH to 0000H, starts counting, and outputs a one-shot pulse from the TOPn1 pin. After the one-shot pulse is output, the 16-bit counter is set to FFFFH, stops counting, and waits for a trigger. If a trigger is generated again while the one-shot pulse is being output, it is ignored.

The output delay period and active level width of the one-shot pulse can be calculated as follows.

Output delay period = (Set value of TPnCCR1 register) × Count clock cycle

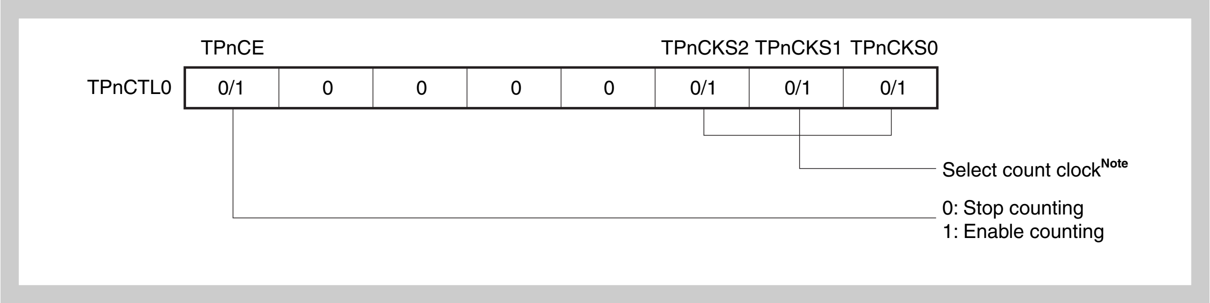
Active level width =
(Set value of TPnCCR0 register – Set value of TPnCCR1 register + 1) × Count clock cycle

The compare match interrupt request signal INTTPnCC0 is generated when the 16-bit counter counts after its count value matches the value of the CCR0 buffer register. The compare match interrupt request signal INTTPnCC1 is generated when the count value of the 16-bit counter matches the value of the CCR1 buffer register.

The valid edge of an external trigger input or setting the software trigger (TPnCTL1.TPnEST bit) to 1 is used as the trigger.

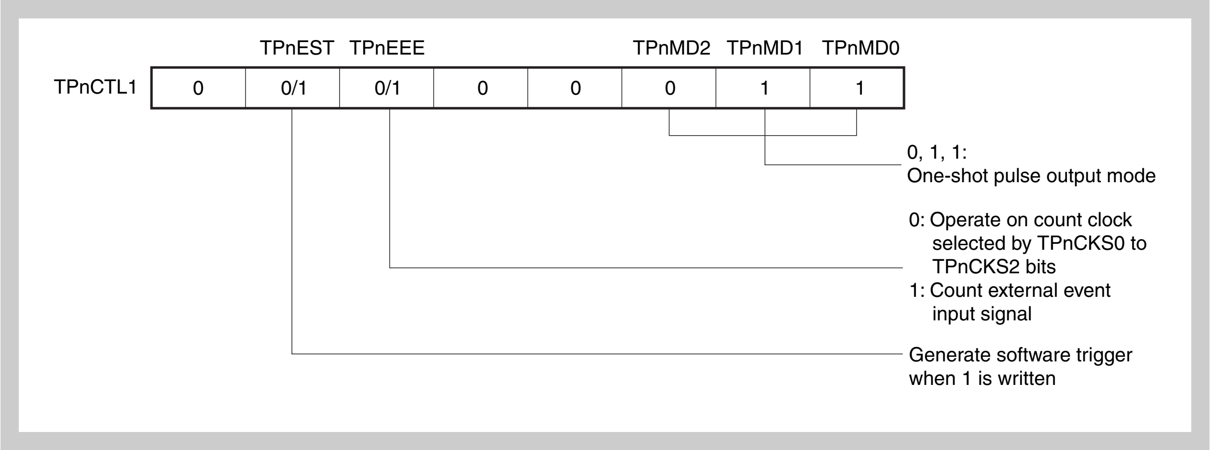
(1) Setting of registers in one-shot pulse output mode

(a) TMPn control register 0 (TPnCTL0)



Note The setting is invalid when the TPnCTL1.TPnEEE bit = 1.

(b) TMPn control register 1 (TPnCTL1)


TPnEST
TPnEEE
TPnMD2
TPnMD1
TPnMD0
TPnCTL1
0
0/1
0/1
0
0
0
1
1
0, 1, 1:
One-shot pulse output mode
0: Operate on count clock selected by TPnCKS0 to TPnCKS2 bits
1: Count external event input signal
Generate software trigger when 1 is written

(c) TMPn I/O control register 0 (TPnIOC0)

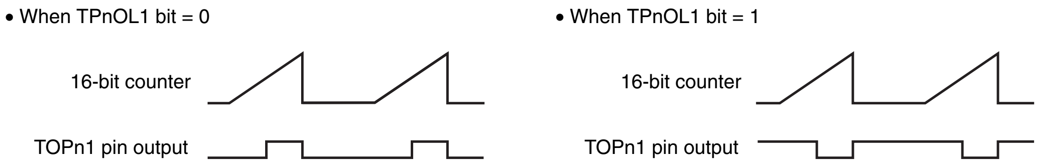
					TPnOL1	TPnOE1	TPnOL0	TPnOE0
TPnIOC0	0	0	0	0	0/1	0/1	0/1	0/1

TPnOE0:
0: Disable TOPn0 pin output
1: Enable TOPn0 pin output

TPnOL0:
Setting of output level while operation of TOPn0 pin is disabled
0: Low level
1: High level

TPnOE1:
0: Disable TOPn1 pin output
1: Enable TOPn1 pin output

TPnOL1:
Specifies active level of TOPn1 pin output
0: Active-high
1: Active-low



(d) TMPn I/O control register 2 (TPnIOC2)

					TPnEES1	TPnEES0	TPnETS1	TPnETS0
TPnIOC2	0	0	0	0	0/1	0/1	0/1	0/1

TPnETS1, TPnETS0: Select valid edge of external trigger input

TPnEES1, TPnEES0: Select valid edge of external event count input

(e) TMPn counter read buffer register (TPnCNT)

The value of the 16-bit counter can be read by reading the TPnCNT register.

(f) TMPn capture/compare registers 0 and 1 (TPnCCR0 and TPnCCR1)

If D_0 is set to the TPnCCR0 register and D_1 to the TPnCCR1 register, the active level width and output delay period of the one-shot pulse are as follows.

Active level width = $(D_1 - D_0 + 1) \times$ Count clock cycle

Output delay period = $D_1 \times$ Count clock cycle

Note TMPn I/O control register 1 (TPnIOC1) and TMPn option register 0 (TPnOPT0) are not used in the one-shot pulse output mode.

(2) Operation flow in one-shot pulse output mode

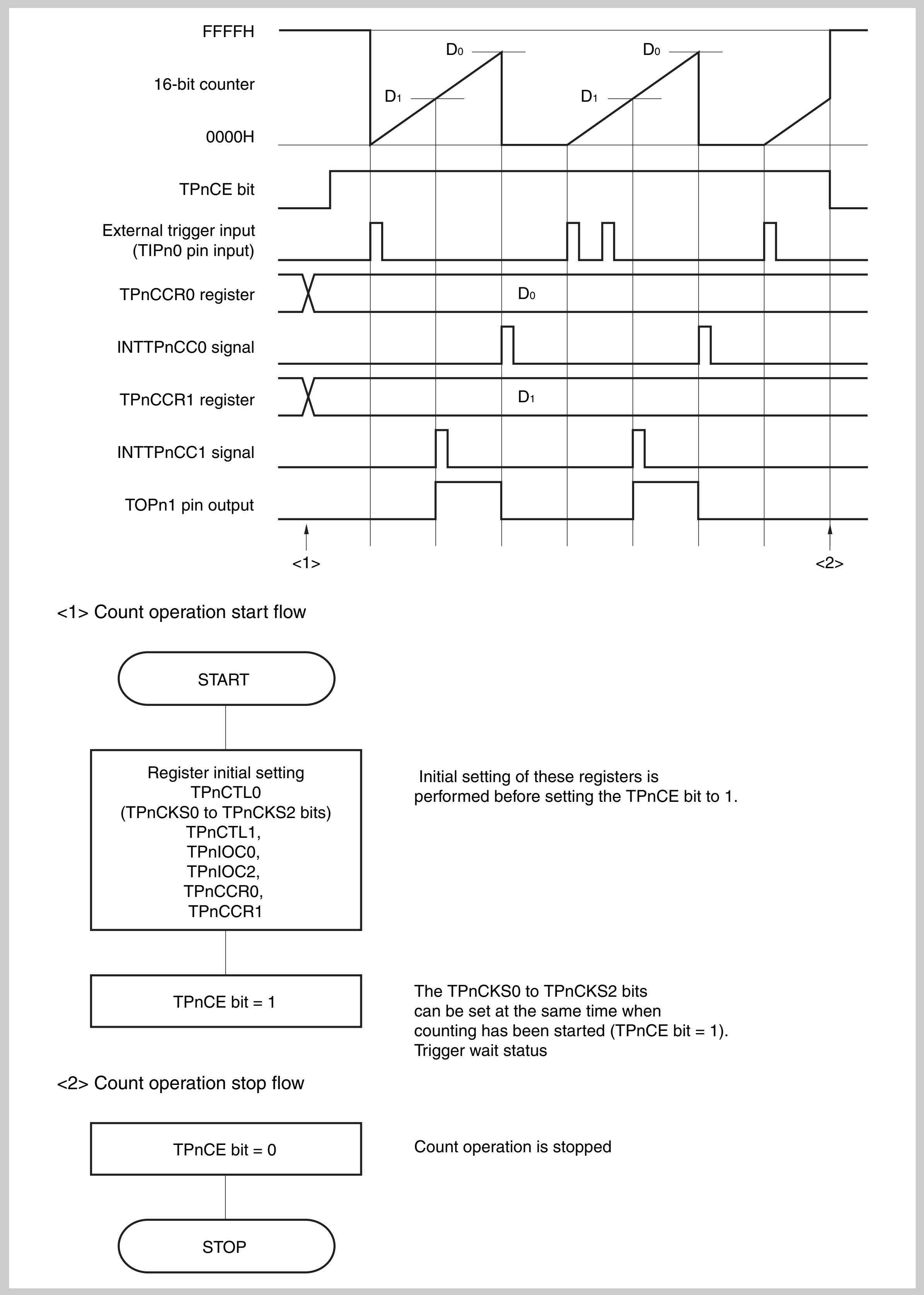


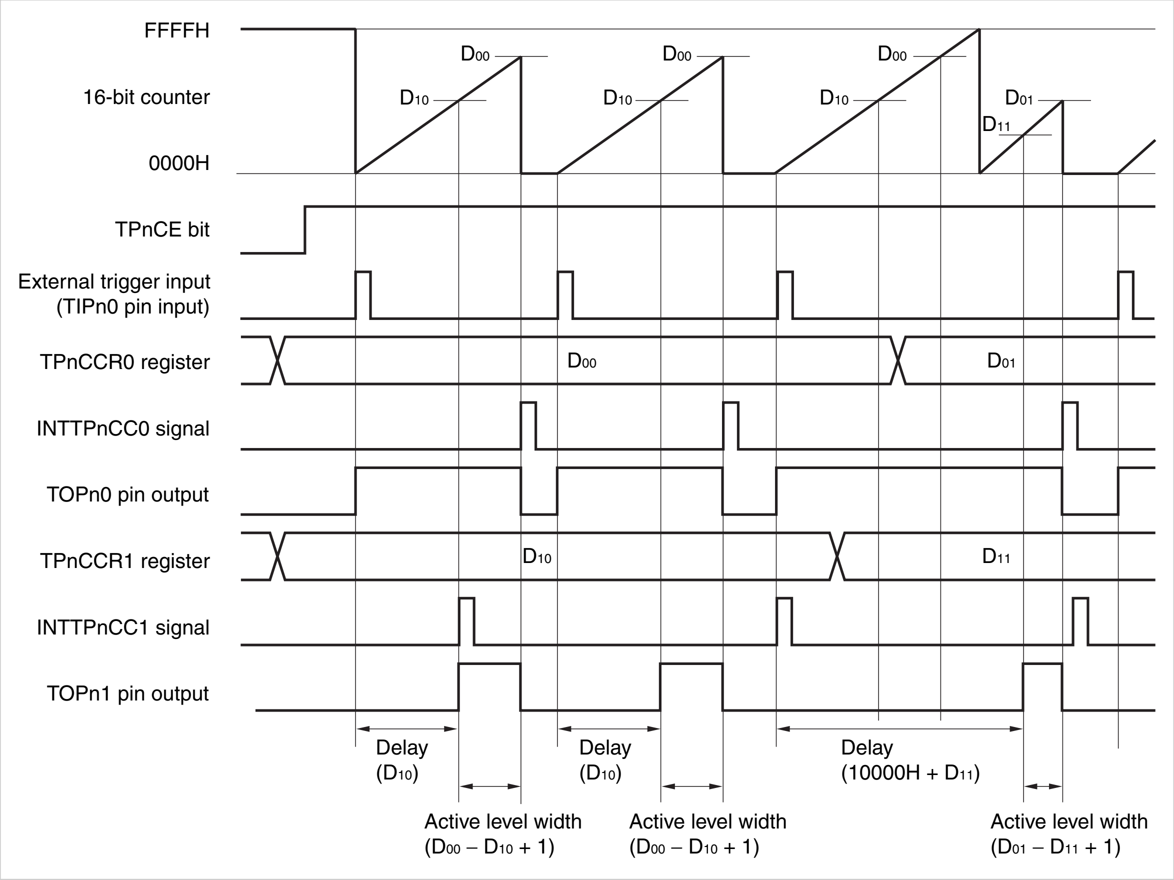
Figure 10-20 Software processing flow in one-shot pulse output mode

(3) Operation timing in one-shot pulse output mode

(a) Note on rewriting TPnCCRm register

To change the set value of the TPnCCRm register to a smaller value, stop counting once, and then change the set value.

If the value of the TPnCCRm register is rewritten to a smaller value during counting, the 16-bit counter may overflow.

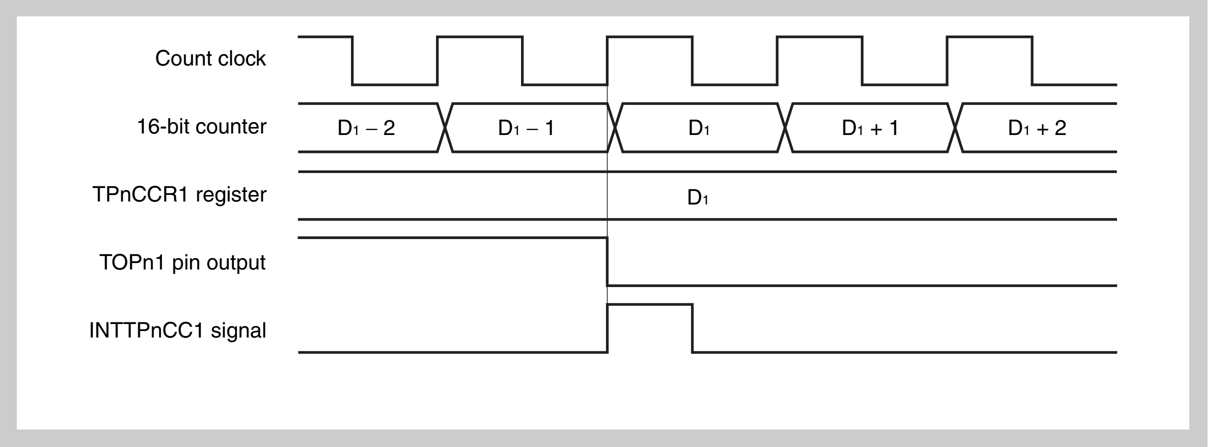


When the TPnCCR0 register is rewritten from D_{00} to D_{01} and the TPnCCR1 register from D_{10} to D_{11} where $D_{00} > D_{01}$ and $D_{10} > D_{11}$, if the TPnCCR1 register is rewritten when the count value of the 16-bit counter is greater than D_{11} and less than D_{10} and if the TPnCCR0 register is rewritten when the count value is greater than D_{01} and less than D_{00}, each set value is reflected as soon as the register has been rewritten and compared with the count value. The counter counts up to FFFFH and then counts up again from 0000H. When the count value matches D_{11}, the counter generates the INTTPnCC1 signal and asserts the TOPn1 pin. When the count value matches D_{01}, the counter generates the INTTPnCC0 signal, deasserts the TOPn1 pin, and stops counting.

Therefore, the counter may output a pulse with a delay period or active period different from that of the one-shot pulse that is originally expected.

(b) Generation timing of compare match interrupt request signal (INTTPnCC1)

The generation timing of the INTTPnCC1 signal in the one-shot pulse output mode is different from other INTTPnCC1 signals; the INTTPnCC1 signal is generated when the count value of the 16-bit counter matches the value of the TPnCCR1 register.



Usually, the INTTPnCC1 signal is generated when the 16-bit counter counts up next time after its count value matches the value of the TPnCCR1 register.

In the one-shot pulse output mode, however, it is generated one clock earlier. This is because the timing is changed to match the change timing of the TOPn1 pin.

10.5.5 PWM output mode (TPnMD2 to TPnMD0 = 100)

In the PWM output mode, a PWM waveform is output from the TOPn1 pin when the TPnCTL0.TPnCE bit is set to 1.

In addition, a pulse with one cycle of the PWM waveform as half its cycle is output from the TOPn0 pin.

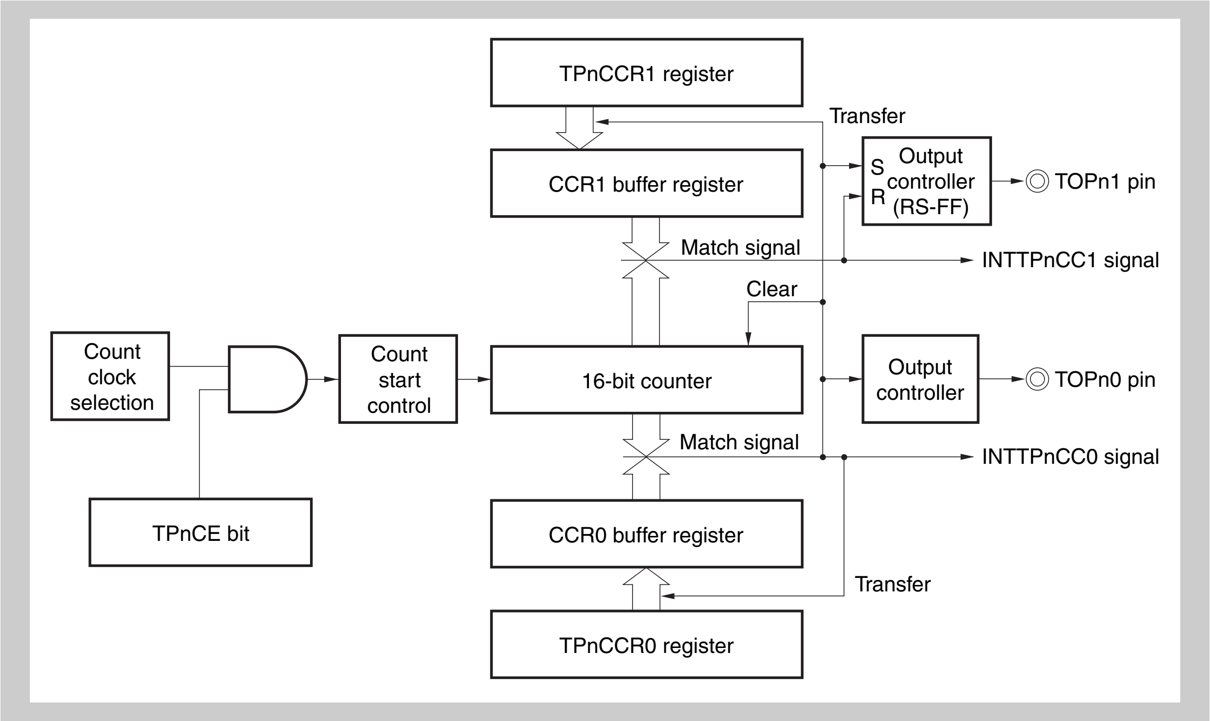


Figure 10-21 Configuration in PWM output mode

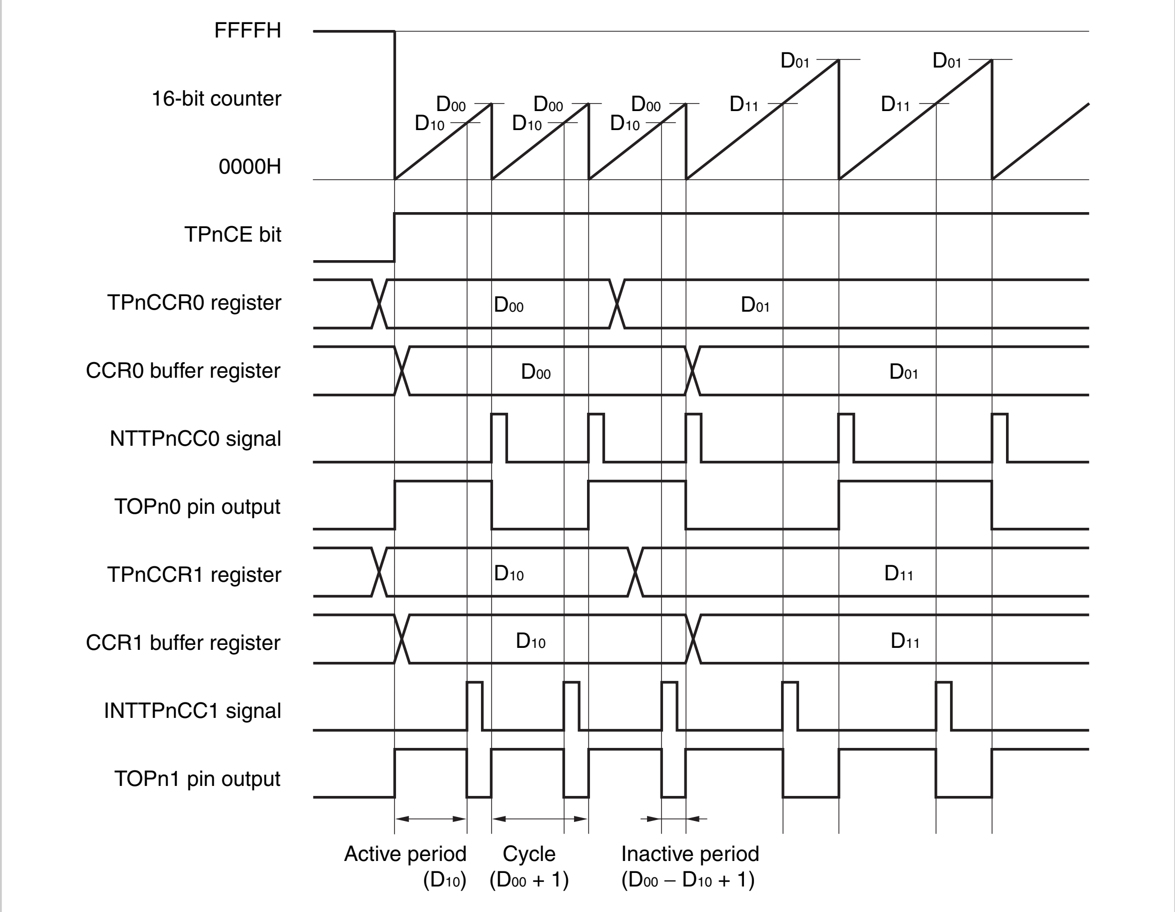


Figure 10-22 Basic timing in PWM output mode

When the TPnCE bit is set to 1, the 16-bit counter is cleared from FFFFH to 0000H, starts counting, and outputs a PWM waveform from the TOPn1 pin.

The active level width, cycle, and duty factor of the PWM waveform can be calculated as follows.

Active level width = (Set value of TPnCCR1 register) × Count clock cycle

Cycle = (Set value of TPnCCR0 register + 1) × Count clock cycle

Duty factor = (Set value of TPnCCR1 register)/(Set value of TPnCCR0 register + 1)

The PWM waveform can be changed by rewriting the TPnCCRm register while the counter is operating. The newly written value is reflected when the count value of the 16-bit counter matches the value of the CCR0 buffer register and the 16-bit counter is cleared to 0000H.

The compare match interrupt request signal INTTPnCC0 is generated when the 16-bit counter counts next time after its count value matches the value of the CCR0 buffer register, and the 16-bit counter is cleared to 0000H. The compare match interrupt request signal INTTPnCC1 is generated when the count value of the 16-bit counter matches the value of the CCR1 buffer register.

The value set to the TPnCCRm register is transferred to the CCRm buffer register when the count value of the 16-bit counter matches the value of the CCRm buffer register and the 16-bit counter is cleared to 0000H.

(1) Setting of registers in PWM output mode

(a) TMPn control register 0 (TPnCTL0)

	TPnCE					TPnCKS2	TPnCKS1	TPnCKS0
TPnCTL0	0/1	0	0	0	0	0/1	0/1	0/1

TPnCKS2 to TPnCKS0: Select count clock[Note]

TPnCE:
0: Stop counting
1: Enable counting

Note The setting is invalid when the TPnCTL1.TPnEEE bit = 1.

(b) TMPn control register 1 (TPnCTL1)

		TPnEST	TPnEEE			TPnMD2	TPnMD1	TPnMD0
TPnCTL1	0	0	0	0	0	1	0	0

TPnMD2 to TPnMD0:
1, 0, 0:
PWM output mode

(c) TMPn I/O control register 0 (TPnIOC0)

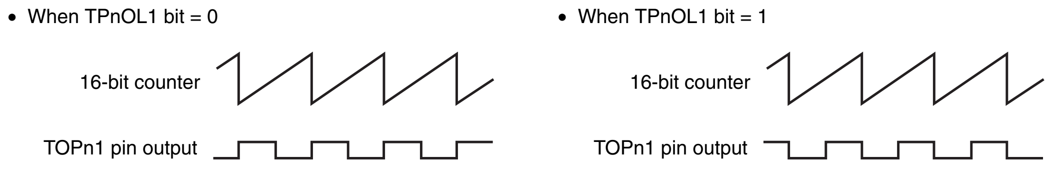
					TPnOL1	TPnOE1	TPnOL0	TPnOE0
TPnIOC0	0	0	0	0	0/1	0/1	0/1	0/1

TPnOE0:
0: Disable TOPn0 pin output
1: Enable TOPn0 pin output

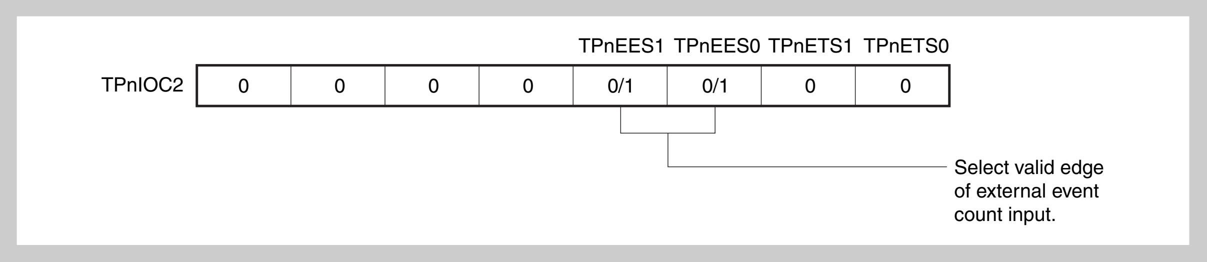
TPnOL0:
Setting of output level while operation of TOPn0 pin is disabled
0: Low level
1: High level

TPnOE1:
0: Disable TOPn1 pin output
1: Enable TOPn1 pin output

TPnOL1:
Specifies active level of TOPn1 pin output
0: Active-high
1: Active-low



(d) TMPn I/O control register 2 (TPnIOC2)



					TPnEES1	TPnEES0	TPnETS1	TPnETS0
TPnIOC2	0	0	0	0	0/1	0/1	0	0

Select valid edge of external event count input.

(e) TMPn counter read buffer register (TPnCNT)

The value of the 16-bit counter can be read by reading the TPnCNT register.

(f) TMPn capture/compare registers 0 and 1 (TPnCCR0 and TPnCCR1)

If D_0 is set to the TPnCCR0 register and D_1 to the TPnCCR1 register, the cycle and active level of the PWM waveform are as follows.

Cycle = (D_0 + 1) × Count clock cycle

Active level width = D_1 × Count clock cycle

Note TMPn I/O control register 1 (TPnIOC1) and TMPn option register 0 (TPnOPT0) are not used in the PWM output mode.

(2) Operation flow in PWM output mode

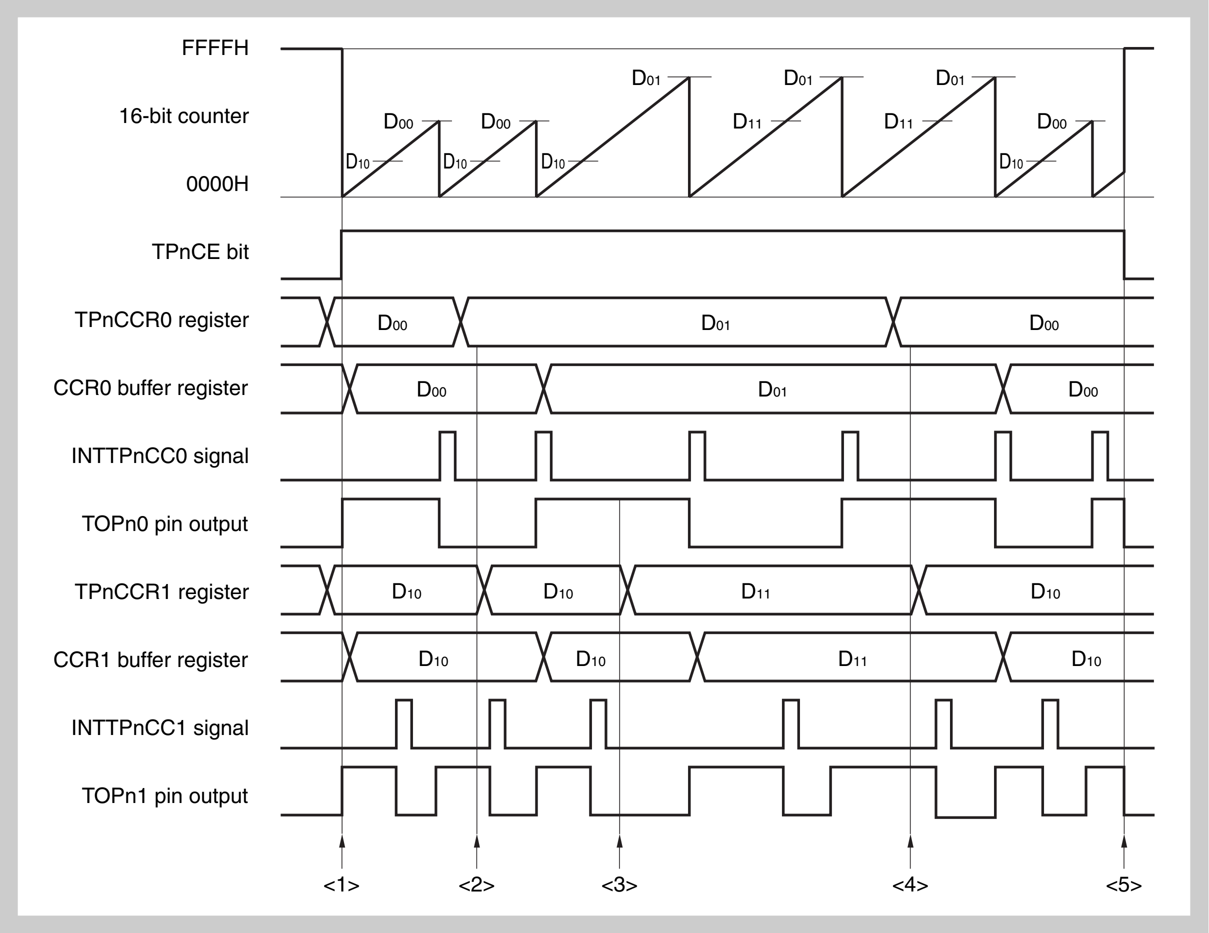
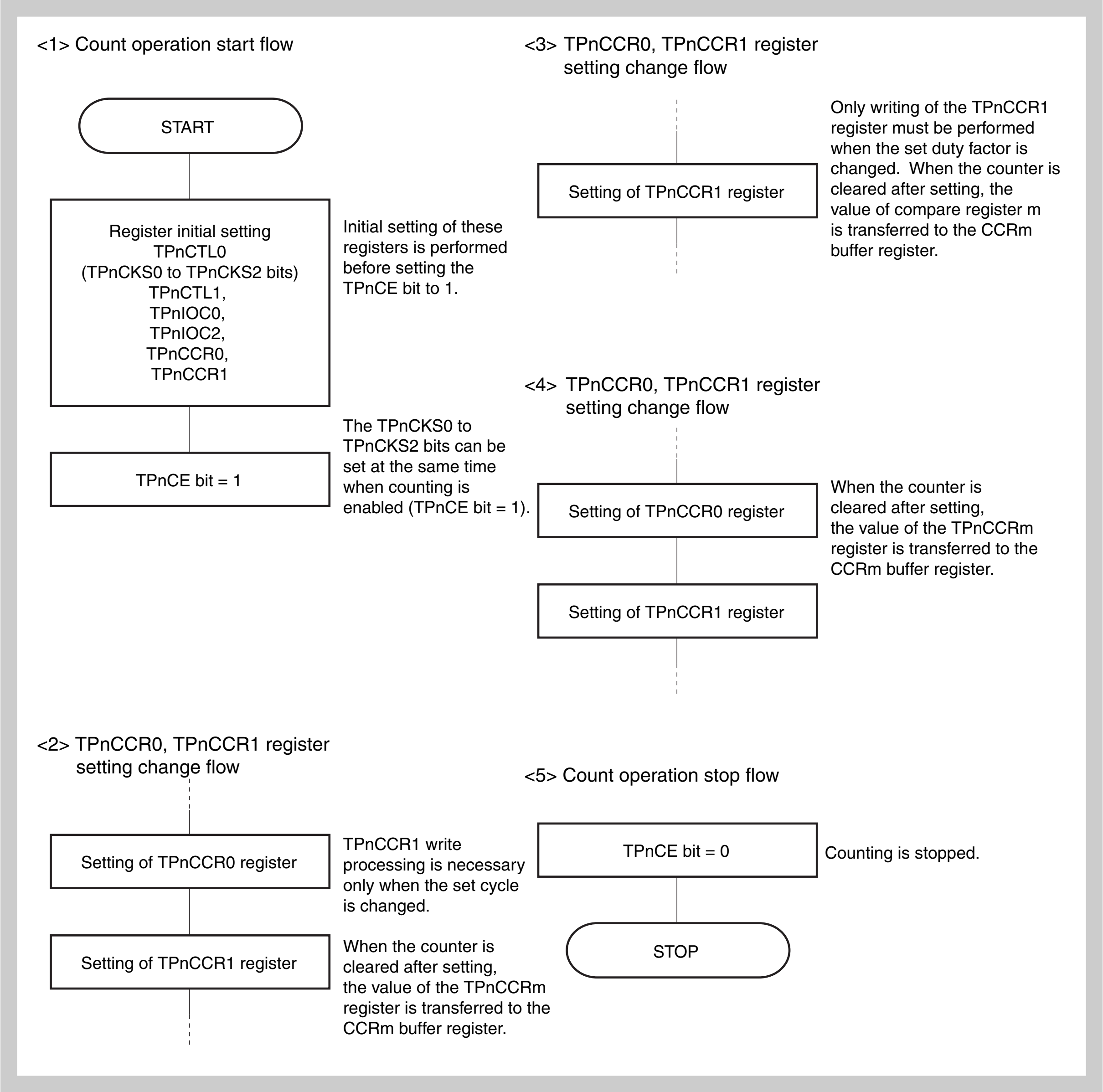


Figure 10-23 Software processing flow in PWM output mode (1/2)


<1> Count operation start flow
START
Register initial setting
TPnCTL0
(TPnCKS0 to TPnCKS2 bits)
TPnCTL1,
TPnIOC0,
TPnIOC2,
TPnCCR0,
TPnCCR1
TPnCE bit = 1
<2> TPnCCR0, TPnCCR1 register setting change flow
Setting of TPnCCR0 register
Setting of TPnCCR1 register
<3> TPnCCR0, TPnCCR1 register setting change flow
Setting of TPnCCR1 register
<4> TPnCCR0, TPnCCR1 register setting change flow
Setting of TPnCCR0 register
Setting of TPnCCR1 register
<5> Count operation stop flow
TPnCE bit = 0
STOP

Initial setting of these registers is performed before setting the TPnCE bit to 1.

The TPnCKS0 to TPnCKS2 bits can be set at the same time when counting is enabled (TPnCE bit = 1).

TPnCCR1 write processing is necessary only when the set cycle is changed.

When the counter is cleared after setting, the value of the TPnCCRm register is transferred to the CCRm buffer register.

Only writing of the TPnCCR1 register must be performed when the set duty factor is changed. When the counter is cleared after setting, the value of compare register m is transferred to the CCRm buffer register.

When the counter is cleared after setting, the value of the TPnCCRm register is transferred to the CCRm buffer register.

Counting is stopped.

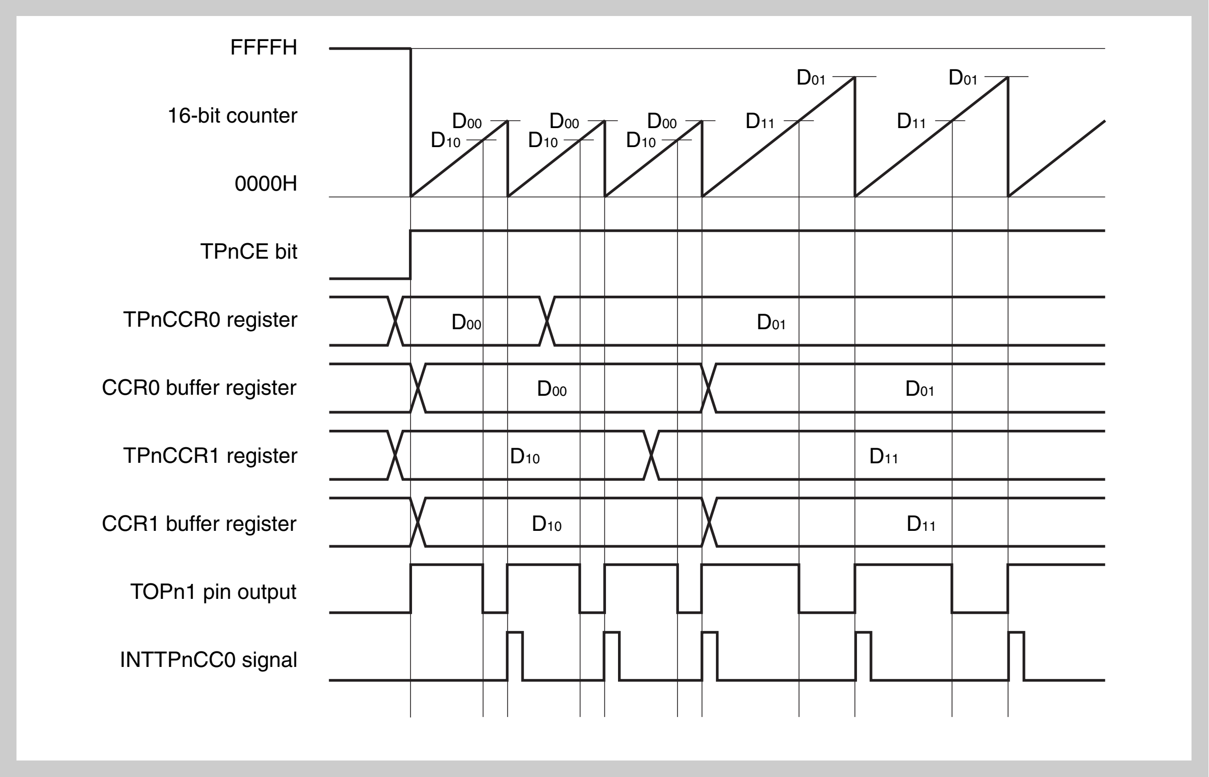
Figure 10-24 Software processing flow in PWM output mode (1/2)

(3) PWM output mode operation timing

(a) Changing pulse width during operation

To change the PWM waveform while the counter is operating, write the TPnCCR1 register last.

Rewrite the TPnCCRm register after writing the TPnCCR1 register after the INTTPnCC1 signal is detected.



To transfer data from the TPnCCRm register to the CCRm buffer register, the TPnCCR1 register must be written.

To change both the cycle and active level of the PWM waveform at this time, first set the cycle to the TPnCCR0 register and then set the active level to the TPnCCR1 register.

To change only the cycle of the PWM waveform, first set the cycle to the TPnCCR0 register, and then write the same value to the TPnCCR1 register.

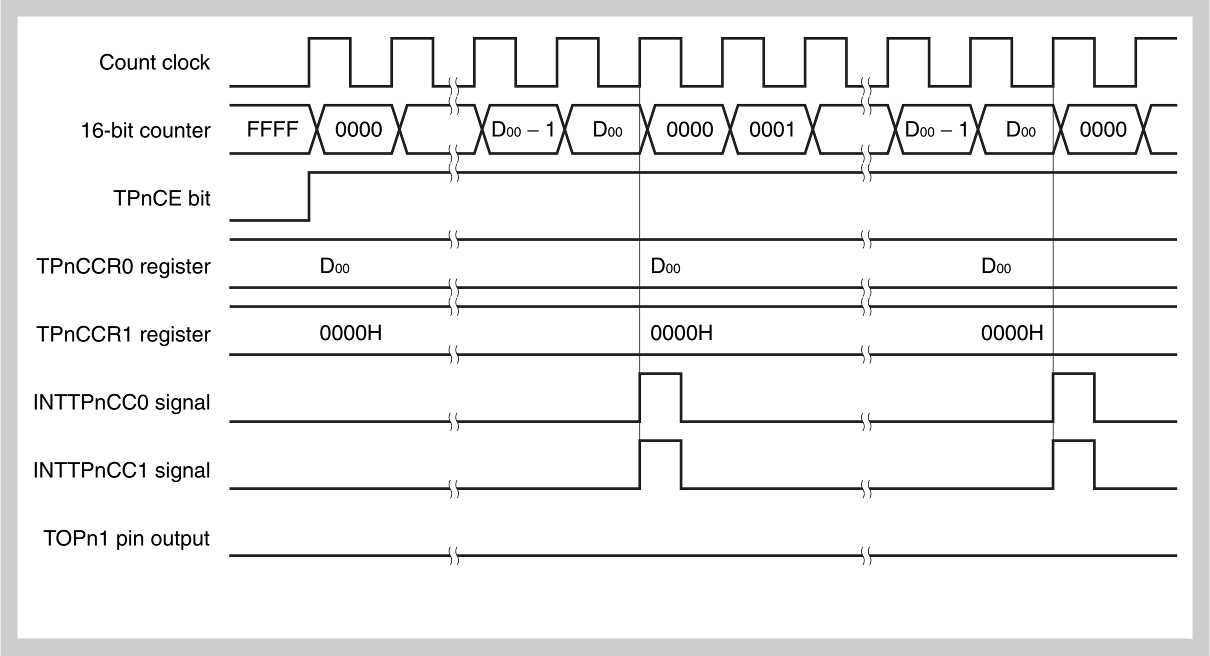
To change only the active level width (duty factor) of the PWM waveform, only the TPnCCR1 register has to be set.

After data is written to the TPnCCR1 register, the value written to the TPnCCRm register is transferred to the CCRm buffer register in synchronization with clearing of the 16-bit counter, and is used as the value compared with the 16-bit counter.

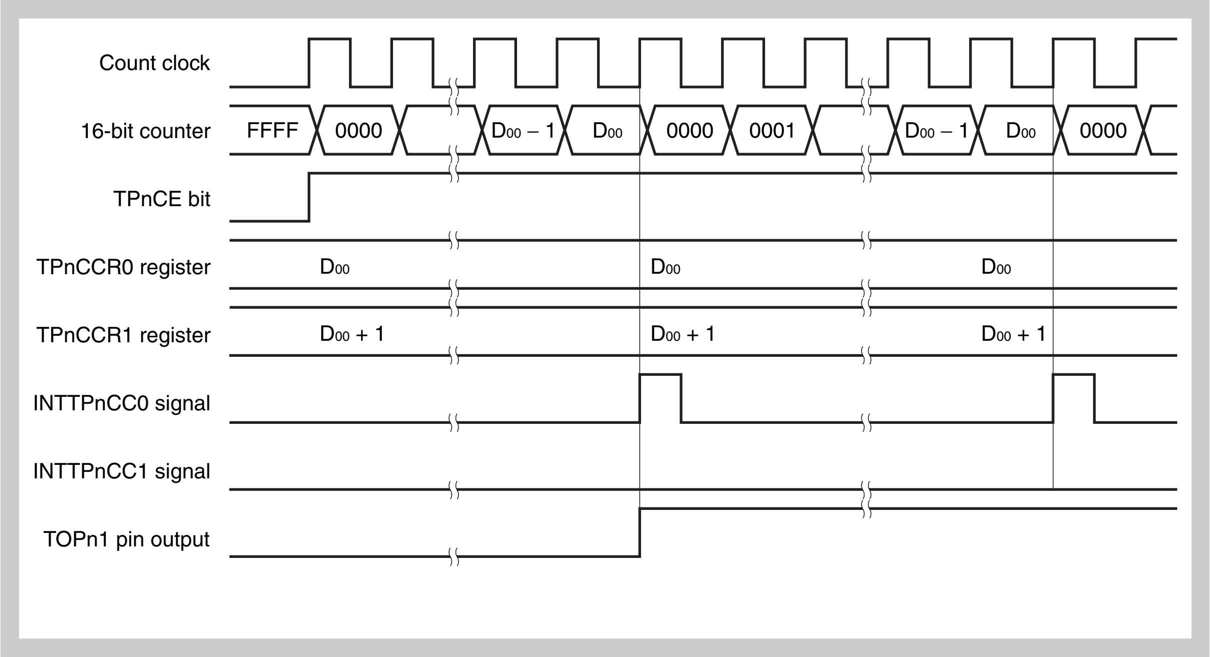
To write the TPnCCR0 or TPnCCR1 register again after writing the TPnCCR1 register once, do so after the INTTPnCC0 signal is generated. Otherwise, the value of the CCRm buffer register may become undefined because the timing of transferring data from the TPnCCRm register to the CCRm buffer register conflicts with writing the TPnCCRm register.

(b) 0%/100% output of PWM waveform

To output a 0% waveform, set the TPnCCR1 register to 0000H. If the set value of the TPnCCR0 register is FFFFH, the INTTPnCC1 signal is generated periodically.

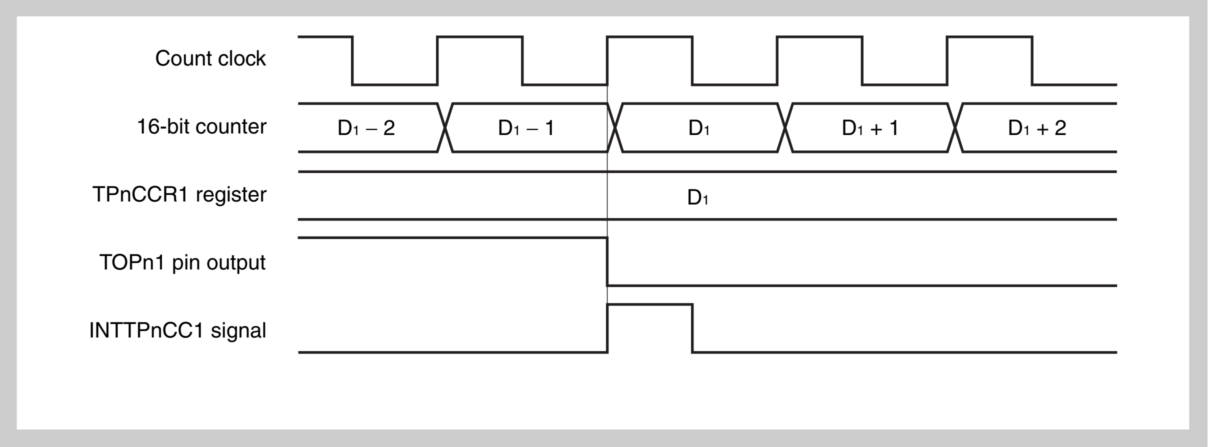


To output a 100% waveform, set a value of (set value of TPnCCR0 register + 1) to the TPnCCR1 register. If the set value of the TPnCCR0 register is FFFFH, 100% output cannot be produced.



(c) Generation timing of compare match interrupt request signal (INTTPnCC1)

The timing of generation of the INTTPnCC1 signal in the PWM output mode differs from the timing of other INTTPnCC1 signals; the INTTPnCC1 signal is generated when the count value of the 16-bit counter matches the value of the TPnCCR1 register.



Usually, the INTTPnCC1 signal is generated in synchronization with the next counting up after the count value of the 16-bit counter matches the value of the TPnCCR1 register.

In the PWM output mode, however, it is generated one clock earlier. This is because the timing is changed to match the change timing of the output signal of the TOPn1 pin.

10.5.6 Free-running timer mode (TPnMD2 to TPnMD0 = 101)

In the free-running timer mode, 16-bit timer/event counter P starts counting when the TPnCTL0.TPnCE bit is set to 1. At this time, the TPnCCRm register can be used as a compare register or a capture register, depending on the setting of the TPnOPT0.TPnCCS0 and TPnOPT0.TPnCCS1 bits.

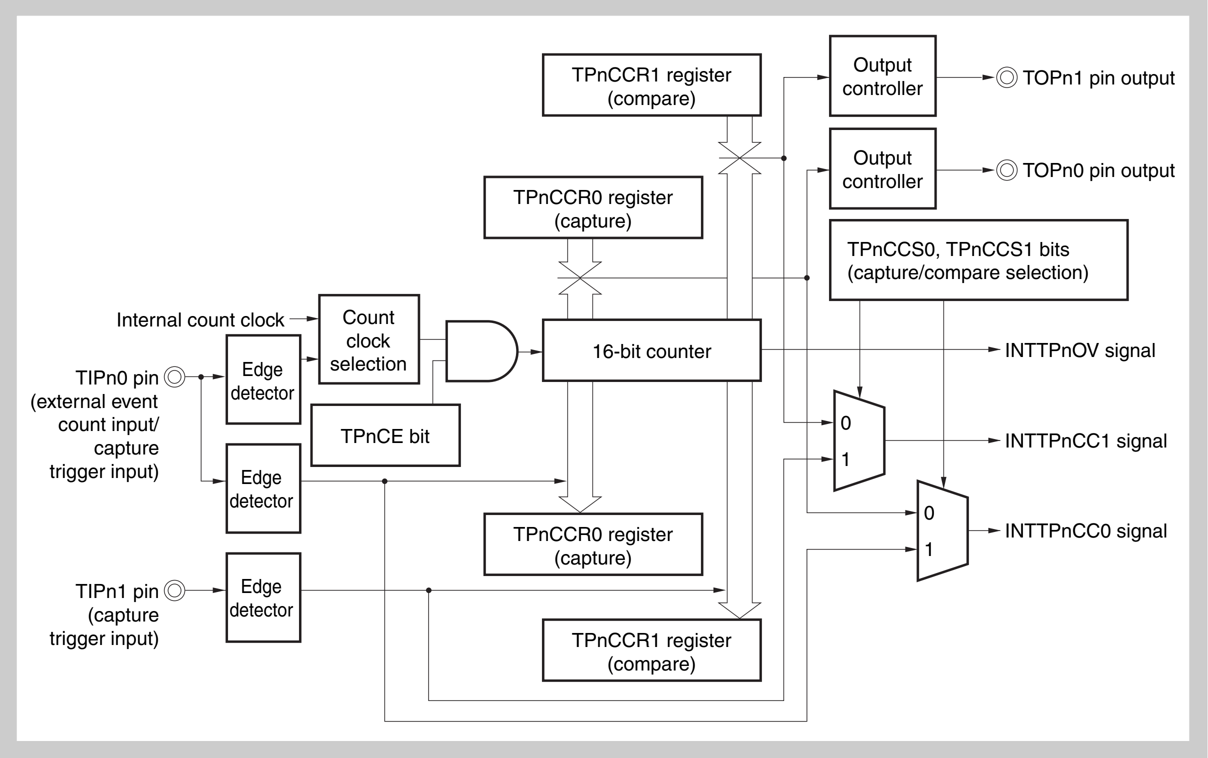

TPnCCR1 register (compare)
Output controller
TOPn1 pin output
Output controller
TOPn0 pin output
TPnCCR0 register (capture)
TPnCCS0, TPnCCS1 bits (capture/compare selection)
Internal count clock
Count clock selection
16-bit counter
INTTPnOV signal
TIPn0 pin (external event count input/ capture trigger input)
Edge detector
TPnCE bit
0
1
INTTPnCC1 signal
Edge detector
0
1
INTTPnCC0 signal
TPnCCR0 register (capture)
TIPn1 pin (capture trigger input)
Edge detector
TPnCCR1 register (compare)

Figure 10-25 Configuration in free-running timer mode

When the TPnCE bit is set to 1, 16-bit timer/event counter P starts counting, and the output signals of the TOPn0 and TOPn1 pins are inverted. When the count value of the 16-bit counter later matches the set value of the TPnCCRm register, a compare match interrupt request signal (INTTPnCCm) is generated, and the output signal of the TOPnm pin is inverted.

The 16-bit counter continues counting in synchronization with the count clock. When it counts up to FFFFH, it generates an overflow interrupt request signal (INTTPnOV) at the next clock, is cleared to 0000H, and continues counting. At this time, the overflow flag (TPnOPT0.TPnOVF bit) is also set to 1. Clear the overflow flag to 0 by executing the CLR instruction by software.

The TPnCCRm register can be rewritten while the counter is operating. If it is rewritten, the new value is reflected at that time, and compared with the count value.

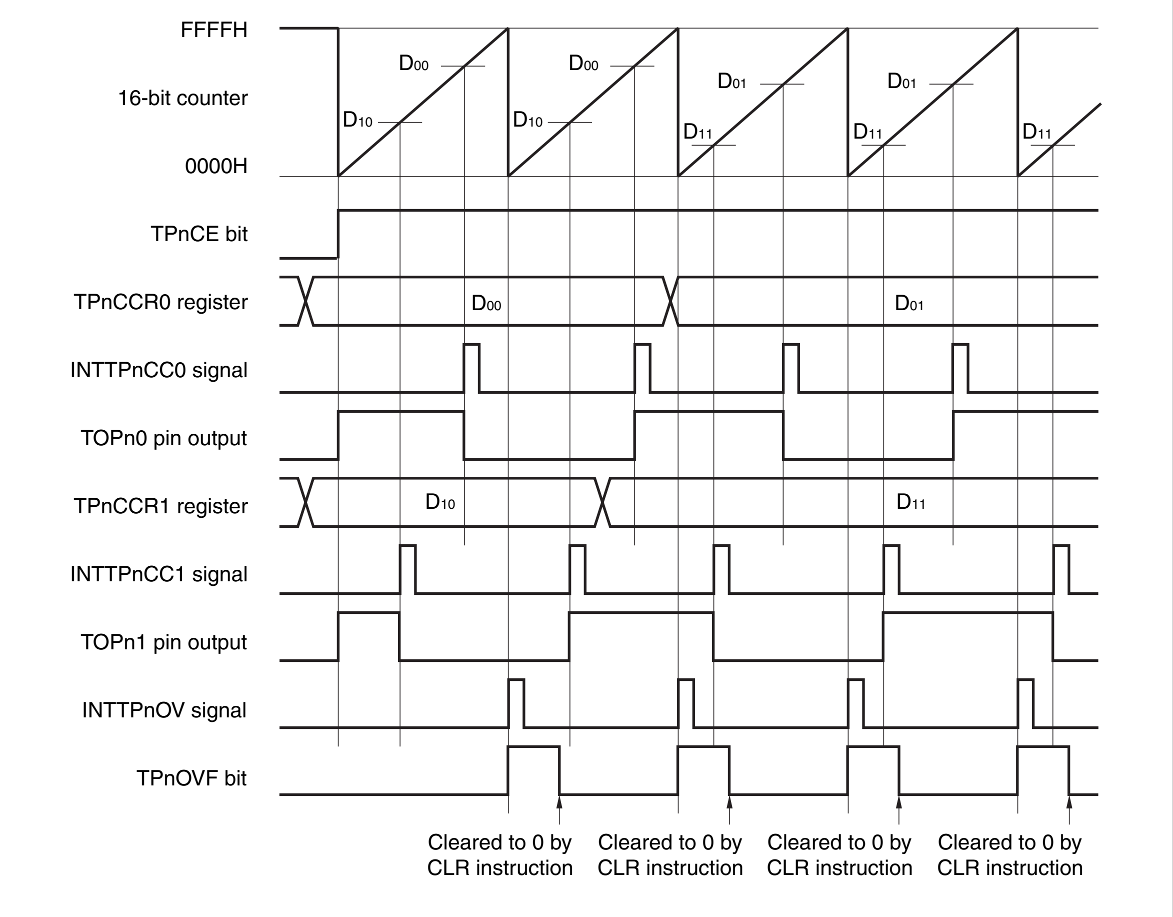


Figure 10-26 Basic timing in free-running timer mode (compare function)

When the TPnCE bit is set to 1, the 16-bit counter starts counting. When the valid edge input to the TIPnm pin is detected, the count value of the 16-bit counter is stored in the TPnCCRm register, and a capture interrupt request signal (INTTPnCCm) is generated.

The 16-bit counter continues counting in synchronization with the count clock. When it counts up to FFFFH, it generates an overflow interrupt request signal (INTTPnOV) at the next clock, is cleared to 0000H, and continues counting. At this time, the overflow flag (TPnOPT0.TPnOVF bit) is also set to 1. Clear the overflow flag to 0 by executing the CLR instruction by software.

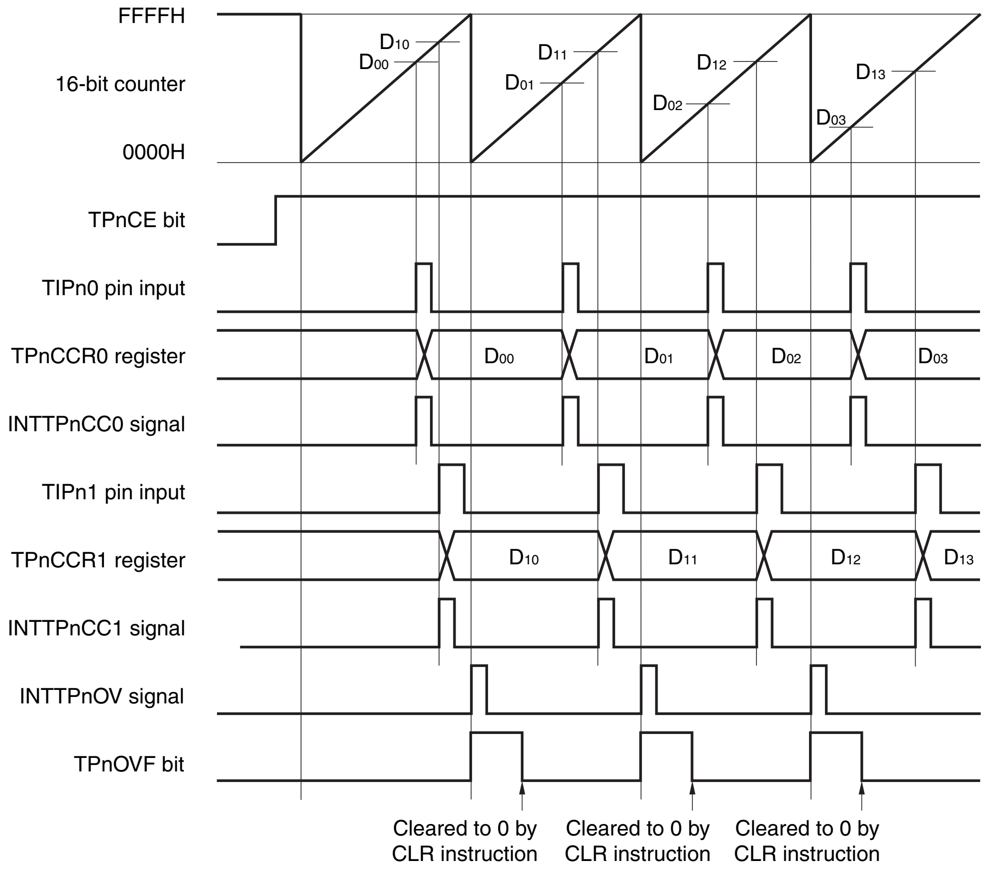
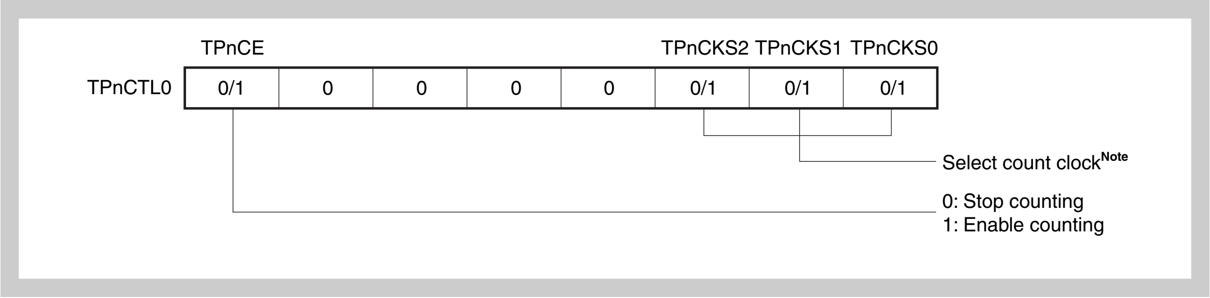


Figure 10-27 Basic timing in free-running timer mode (capture function)

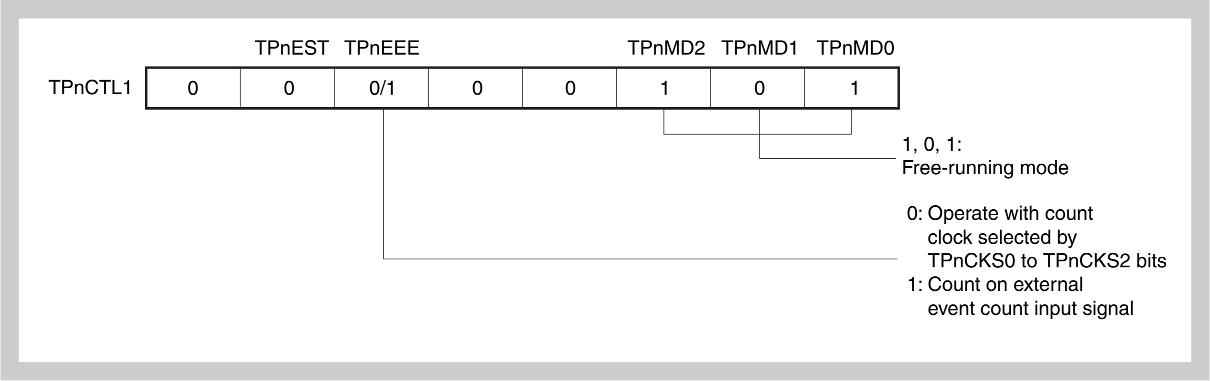
(1) Register setting in free-running timer mode

(a) TMPn control register 0 (TPnCTL0)

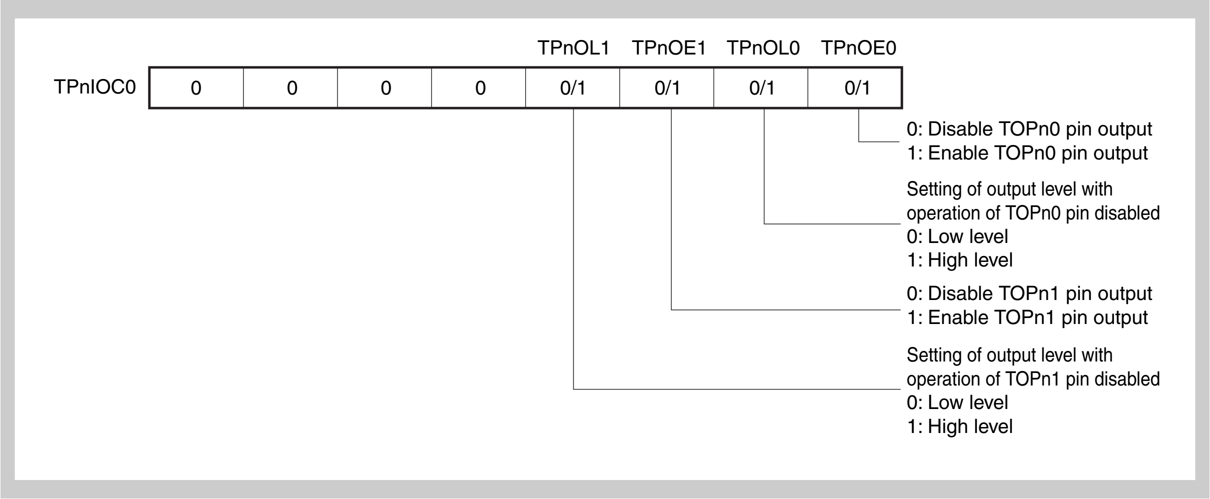


Note The setting is invalid when the TPnCTL1.TPnEEE bit = 1

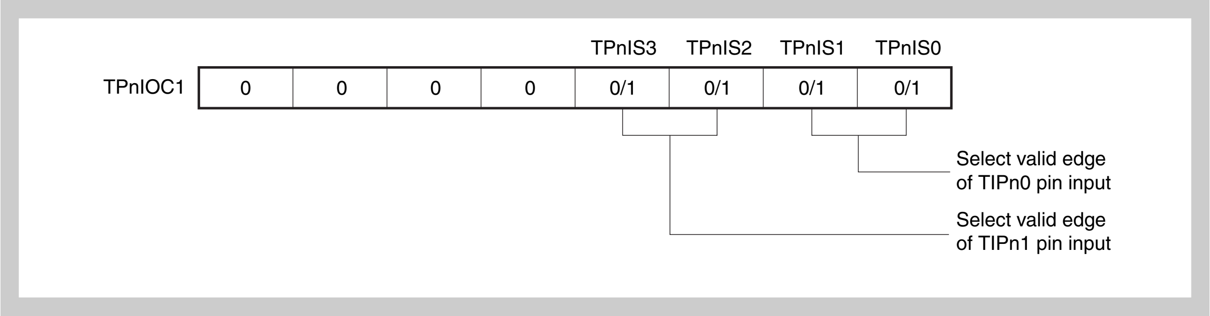
(b) TMPn control register 1 (TPnCTL1)


TPnEST TPnEEE
TPnMD2 TPnMD1 TPnMD0
TPnCTL1
0 0 0/1 0 0 1 0 1
1, 0, 1:
Free-running mode
0: Operate with count clock selected by TPnCKS0 to TPnCKS2 bits
1: Count on external event count input signal

(c) TMPn I/O control register 0 (TPnIOC0)


TPnOL1 TPnOE1 TPnOL0 TPnOE0
TPnIOC0
0 0 0 0 0/1 0/1 0/1 0/1
0: Disable TOPn0 pin output
1: Enable TOPn0 pin output
Setting of output level with operation of TOPn0 pin disabled
0: Low level
1: High level
0: Disable TOPn1 pin output
1: Enable TOPn1 pin output
Setting of output level with operation of TOPn1 pin disabled
0: Low level
1: High level

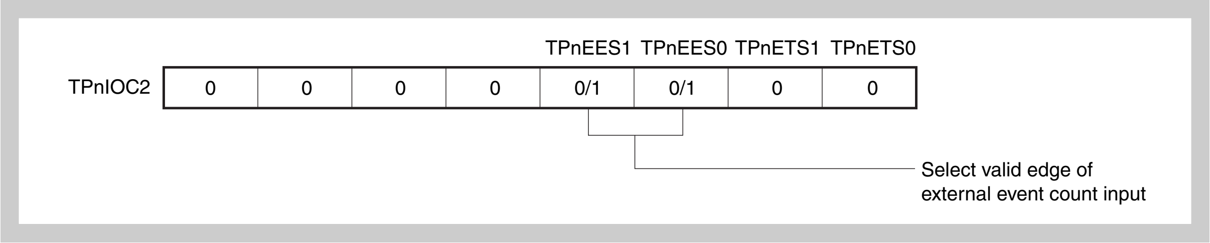
(d) TMPn I/O control register 1 (TPnIOC1)



TPnIOC1					TPnIS3	TPnIS2	TPnIS1	TPnIS0
	0	0	0	0	0/1	0/1	0/1	0/1

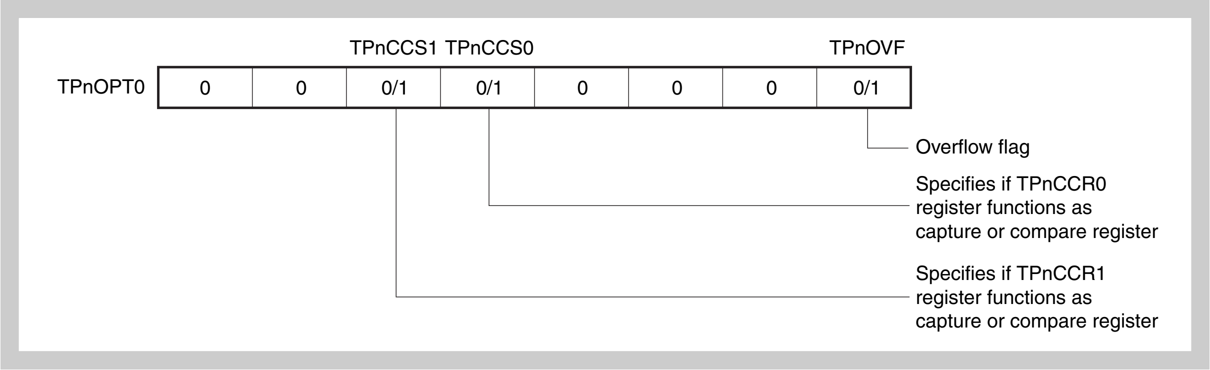
Select valid edge of TIPn0 pin input
Select valid edge of TIPn1 pin input

(e) TMPn I/O control register 2 (TPnIOC2)



TPnIOC2					TPnEES1	TPnEES0	TPnETS1	TPnETS0
	0	0	0	0	0/1	0/1	0	0

(f) TMPn option register 0 (TPnOPT0)



TPnOPT0			TPnCCS1	TPnCCS0				TPnOVF
	0	0	0/1	0/1	0	0	0	0/1

Overflow flag
Specifies if TPnCCR0 register functions as capture or compare register
Specifies if TPnCCR1 register functions as capture or compare register

(g) TMPn counter read buffer register (TPnCNT)

The value of the 16-bit counter can be read by reading the TPnCNT register.

(h) TMPn capture/compare registers 0 and 1 (TPnCCR0 and TPnCCR1)

These registers function as capture registers or compare registers depending on the setting of the TPnOPT0.TPnCCSm bit.

When the registers function as capture registers, they store the count value of the 16-bit counter when the valid edge input to the TIPnm pin is detected.

When the registers function as compare registers and when D_m is set to the TPnCCRm register, the INTTPnCCm signal is generated when the counter reaches (D_m + 1), and the output signal of the TOPnm pin is inverted.

(2) Operation flow in free-running timer mode

(a) When using capture/compare register as compare register

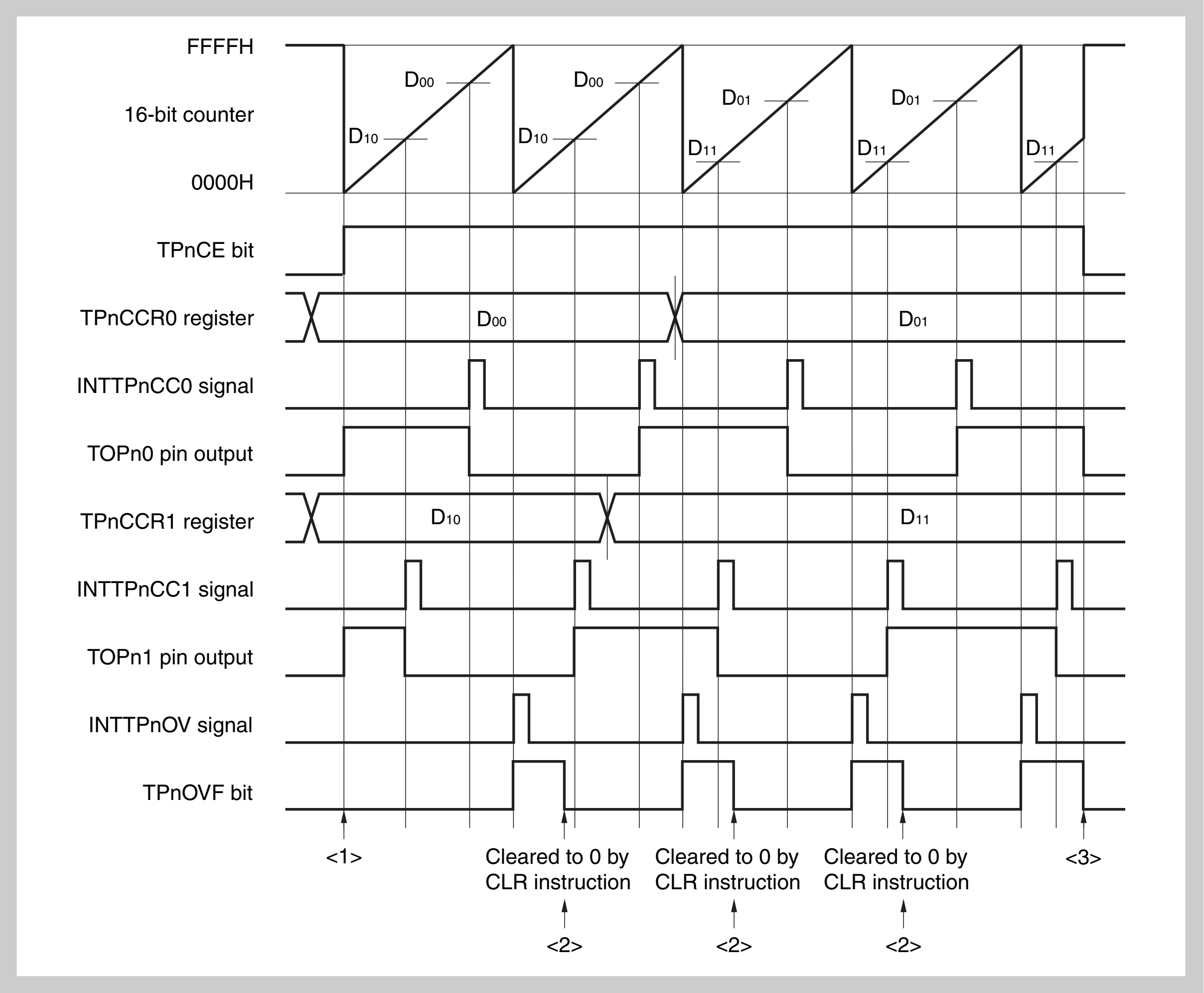


Figure 10-28 Software processing flow in free-running timer mode (compare function) (1/2)

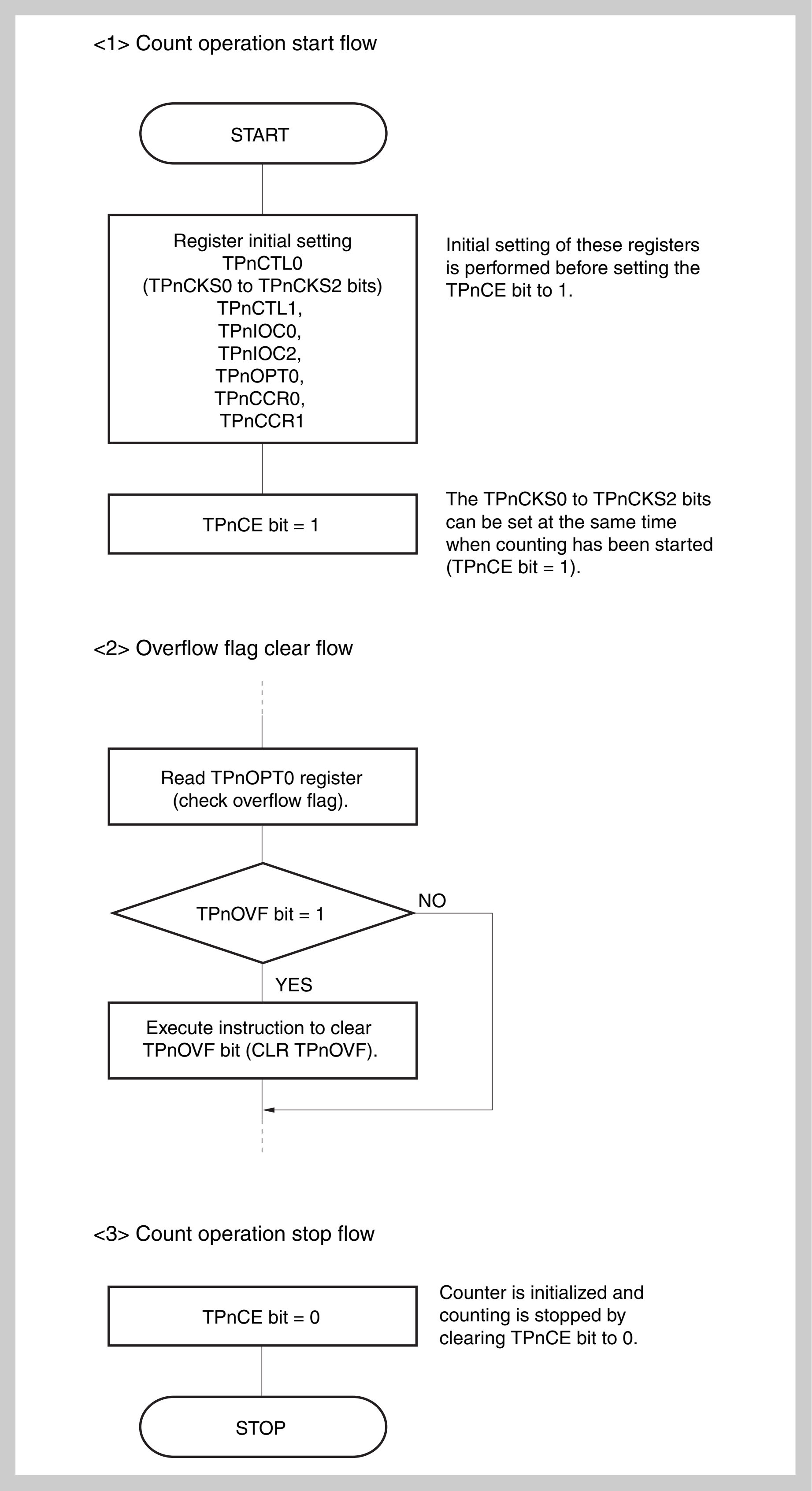

<1> Count operation start flow
START
Register initial setting
TPnCTL0
(TPnCKS0 to TPnCKS2 bits)
TPnCTL1,
TPnIOC0,
TPnIOC2,
TPnOPT0,
TPnCCR0,
TPnCCR1
Initial setting of these registers is performed before setting the TPnCE bit to 1.
TPnCE bit = 1
The TPnCKS0 to TPnCKS2 bits can be set at the same time when counting has been started (TPnCE bit = 1).
<2> Overflow flag clear flow
Read TPnOPT0 register (check overflow flag).
TPnOVF bit = 1
NO
YES
Execute instruction to clear TPnOVF bit (CLR TPnOVF).
<3> Count operation stop flow
TPnCE bit = 0
Counter is initialized and counting is stopped by clearing TPnCE bit to 0.
STOP

Figure 10-29 Software processing flow in free-running timer mode (compare function) (2/2)

(b) When using capture/compare register as capture register

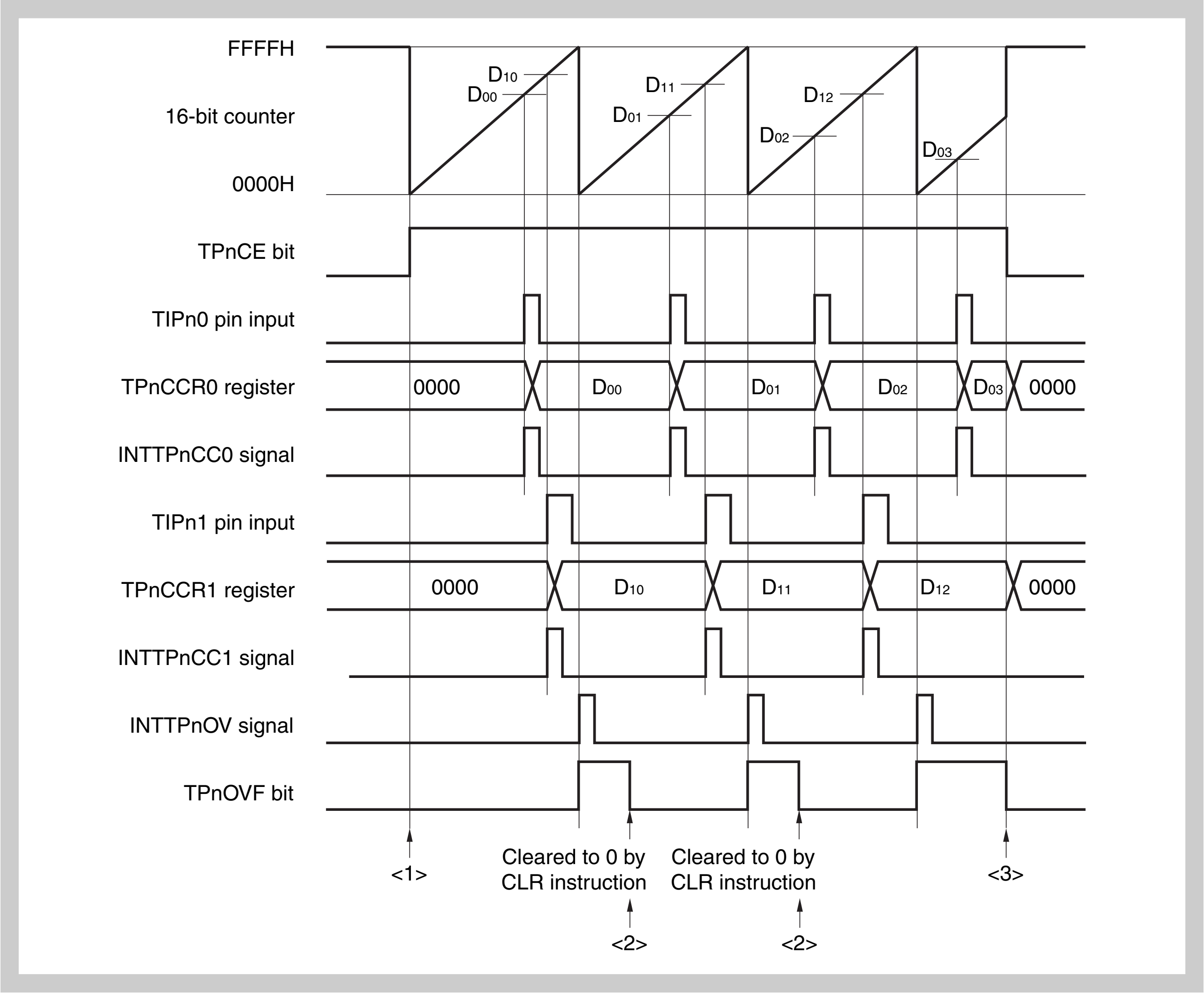


Figure 10-30 Software processing flow in free-running timer mode (capture function) (1/2)

]

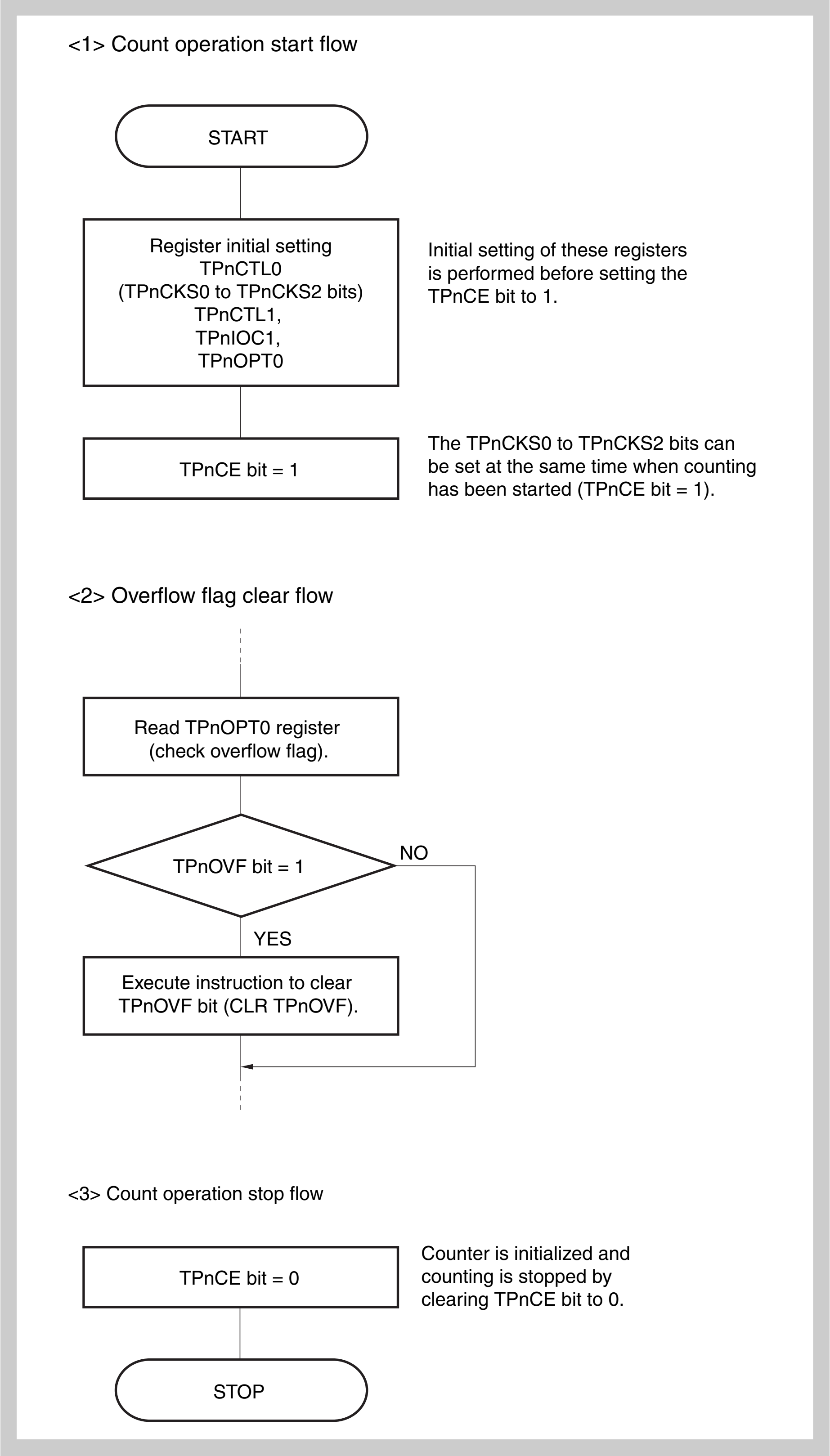
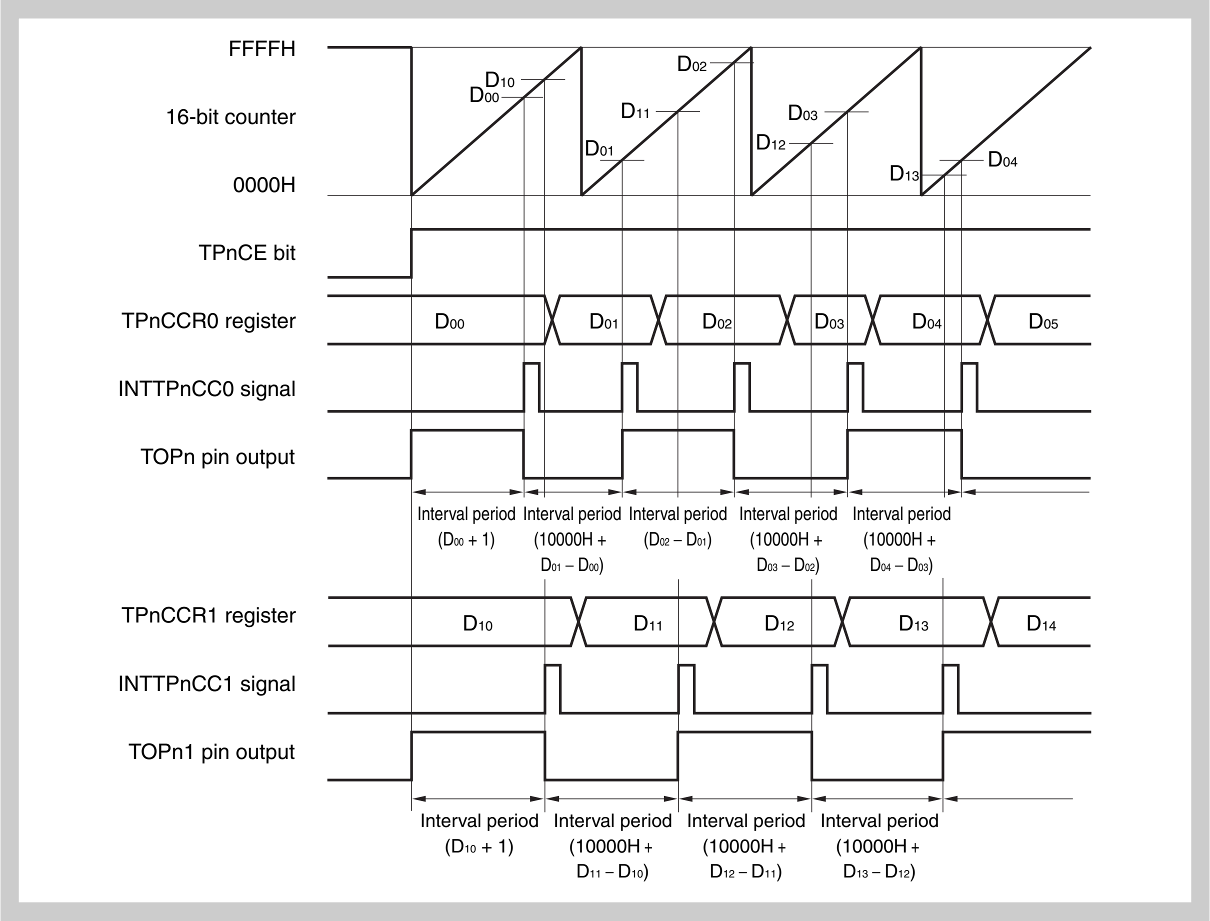

<1> Count operation start flow
START
Register initial setting
TPnCTL0
(TPnCKS0 to TPnCKS2 bits)
TPnCTL1,
TPnIOC1,
TPnOPT0
Initial setting of these registers is performed before setting the TPnCE bit to 1.
TPnCE bit = 1
The TPnCKS0 to TPnCKS2 bits can be set at the same time when counting has been started (TPnCE bit = 1).
<2> Overflow flag clear flow
Read TPnOPT0 register (check overflow flag).
TPnOVF bit = 1
NO
YES
Execute instruction to clear TPnOVF bit (CLR TPnOVF).
<3> Count operation stop flow
TPnCE bit = 0
Counter is initialized and counting is stopped by clearing TPnCE bit to 0.
STOP

Figure 10-31 Software processing flow in free-running timer mode (capture function) (2/2)

(3) Operation timing in free-running timer mode

(a) Interval operation with compare register

When 16-bit timer/event counter P is used as an interval timer with the TPnCCRm register used as a compare register, software processing is necessary for setting a comparison value to generate the next interrupt request signal each time the INTTPnCCm signal has been detected.


FFFFH
16-bit counter
0000H
D10
D00
D01
D11
D02
D12
D03
D13
D04
TPnCE bit
TPnCCR0 register
D00
D01
D02
D03
D04
D05
INTTPnCC0 signal
TOPn pin output
Interval period (D00 + 1)
Interval period (10000H + D01 – D00)
Interval period (D02 – D01)
Interval period (10000H + D03 – D02)
Interval period (10000H + D04 – D03)
TPnCCR1 register
D10
D11
D12
D13
D14
INTTPnCC1 signal
TOPn1 pin output
Interval period (D10 + 1)
Interval period (10000H + D11 – D10)
Interval period (10000H + D12 – D11)
Interval period (10000H + D13 – D12)

When performing an interval operation in the free-running timer mode, two intervals can be set with one channel.

To perform the interval operation, the value of the corresponding TPnCCRm register must be re-set in the interrupt servicing that is executed when the INTTPnCCm signal is detected.

The set value for re-setting the TPnCCRm register can be calculated by the following expression, where “D_m” is the interval period.

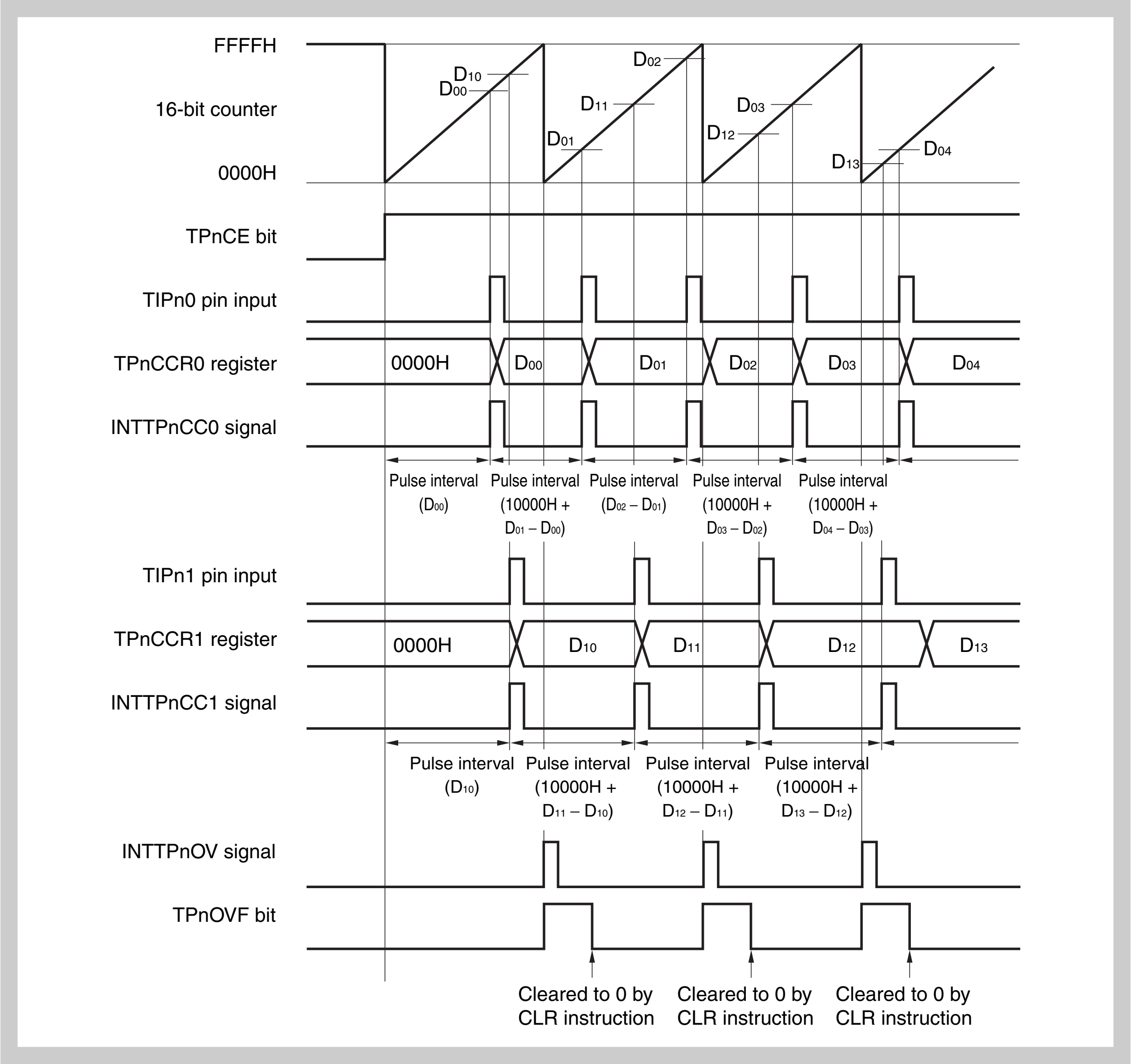
Compare register default value: $D_m – 1$

Value set to compare register second and subsequent time:
Previous set value + D_m

(If the calculation result is greater than FFFFH, subtract 10000H from the result and set this value to the register.)

(b) Pulse width measurement with capture register

When pulse width measurement is performed with the TPnCCRm register used as a capture register, software processing is necessary for reading the capture register each time the INTTPnCCm signal has been detected and for calculating an interval.


FFFFH
16-bit counter
0000H
D10
D00
D02
D11
D01
D03
D12
D13
D04
TPnCE bit
TIPn0 pin input
TPnCCR0 register
0000H
D00
D01
D02
D03
D04
INTTPnCC0 signal
Pulse interval (D00)
Pulse interval (10000H + D01 – D00)
Pulse interval (D02 – D01)
Pulse interval (10000H + D03 – D02)
Pulse interval (10000H + D04 – D03)
TIPn1 pin input
TPnCCR1 register
0000H
D10
D11
D12
D13
INTTPnCC1 signal
Pulse interval (D10)
Pulse interval (10000H + D11 – D10)
Pulse interval (10000H + D12 – D11)
Pulse interval (10000H + D13 – D12)
INTTPnOV signal
TPnOVF bit
Cleared to 0 by CLR instruction
Cleared to 0 by CLR instruction
Cleared to 0 by CLR instruction

When executing pulse width measurement in the free-running timer mode, two pulse widths can be measured with one channel.

To measure a pulse width, the pulse width can be calculated by reading the value of the TPnCCRm register in synchronization with the INTTPnCCm signal, and calculating the difference between the read value and the previously read value.

(c) Processing of overflow when two capture registers are used

Care must be exercised in processing the overflow flag when two capture registers are used. First, an example of incorrect processing is shown below.

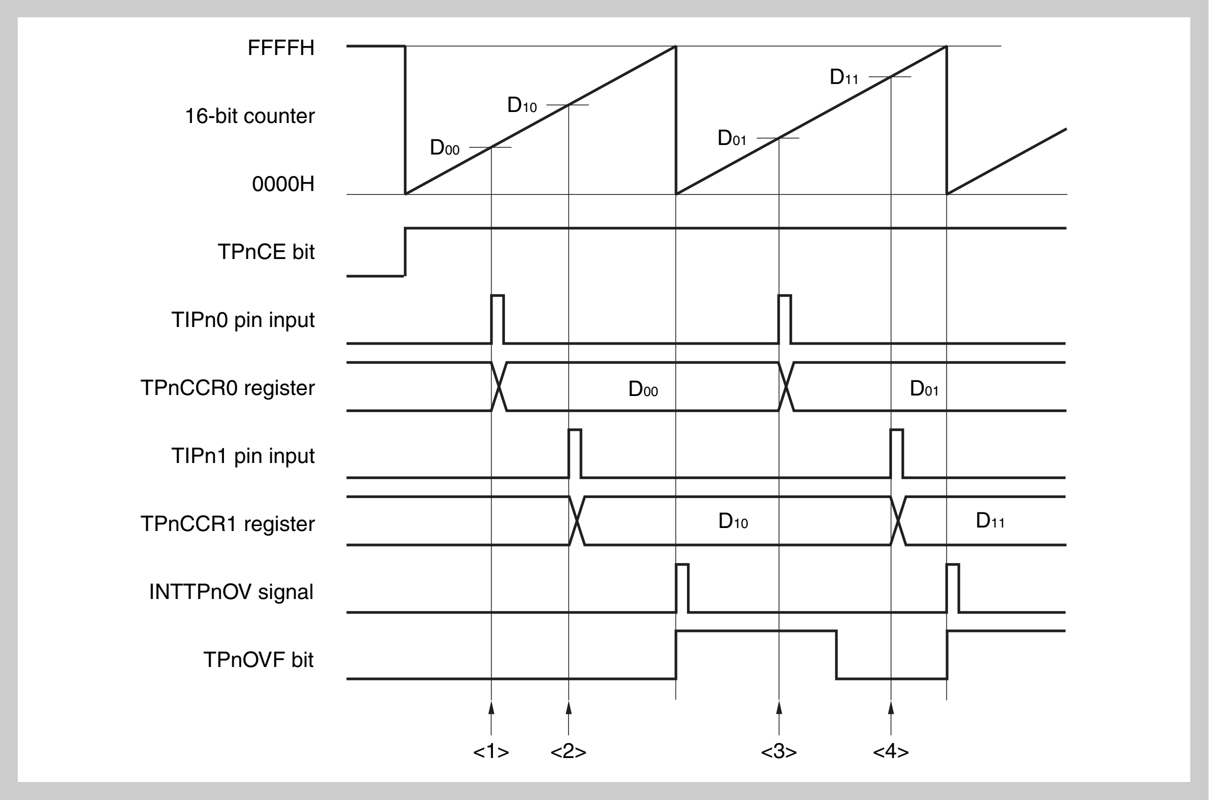


Figure 10-32 Example of incorrect processing when two capture registers are used

The following problem may occur when two pulse widths are measured in the free-running timer mode.

<1> Read the TPnCCR0 register (setting of the default value of the TIPn0 pin input).

<2> Read the TPnCCR1 register (setting of the default value of the TIPn1 pin input).

<3> Read the TPnCCR0 register.

Read the overflow flag. If the overflow flag is 1, clear it to 0.

Because the overflow flag is 1, the pulse width can be calculated by (10000H + D_{01} - D_{00}).

<4> Read the TPnCCR1 register.

Read the overflow flag. Because the flag is cleared in <3>, 0 is read.

Because the overflow flag is 0, the pulse width can be calculated by (D_{11} - D_{10}) (incorrect).

When two capture registers are used, and if the overflow flag is cleared to 0 by one capture register, the other capture register may not obtain the correct pulse width.

Use software when using two capture registers. An example of how to use software is shown below.

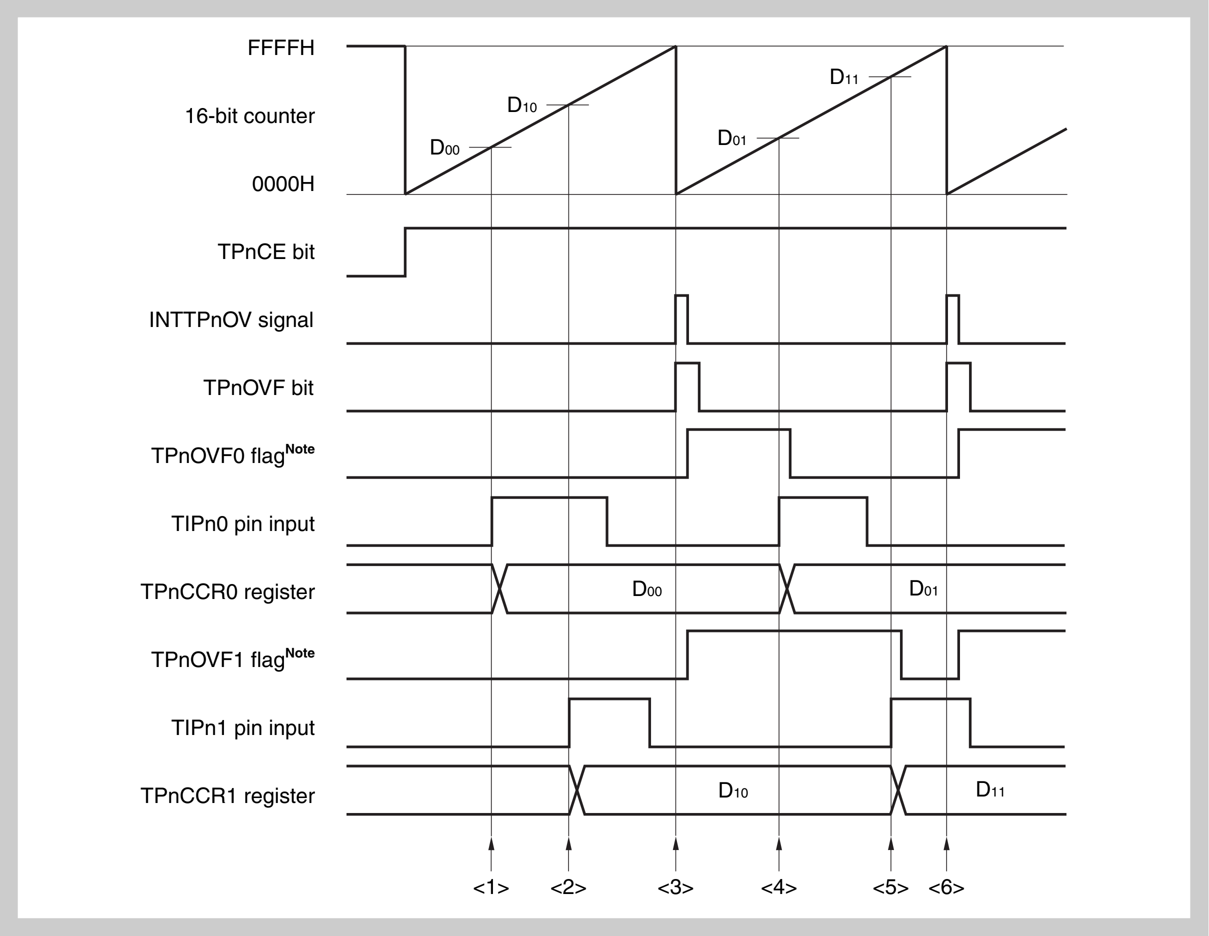


Figure 10-33 Example when two capture registers are used (using overflow interrupt)

Note The TPnOVF0 and TPnOVF1 flags are set on the internal RAM by software.

<1> Read the TPnCCR0 register (setting of the default value of the TIPn0 pin input).

<2> Read the TPnCCR1 register (setting of the default value of the TIPn1 pin input).

<3> An overflow occurs. Set the TPnOVF0 and TPnOVF1 flags to 1 in the overflow interrupt servicing, and clear the overflow flag to 0.

<4> Read the TPnCCR0 register.

Read the TPnOVF0 flag. If the TPnOVF0 flag is 1, clear it to 0.

Because the TPnOVF0 flag is 1, the pulse width can be calculated by (10000H + D_{01} - D_{00}).

<5> Read the TPnCCR1 register.

Read the TPnOVF1 flag. If the TPnOVF1 flag is 1, clear it to 0 (the TPnOVF0 flag is cleared in <4>, and the TPnOVF1 flag remains 1).

Because the TPnOVF1 flag is 1, the pulse width can be calculated by (10000H + D_{11} - D_{10}) (correct).

<6> Same as <3>

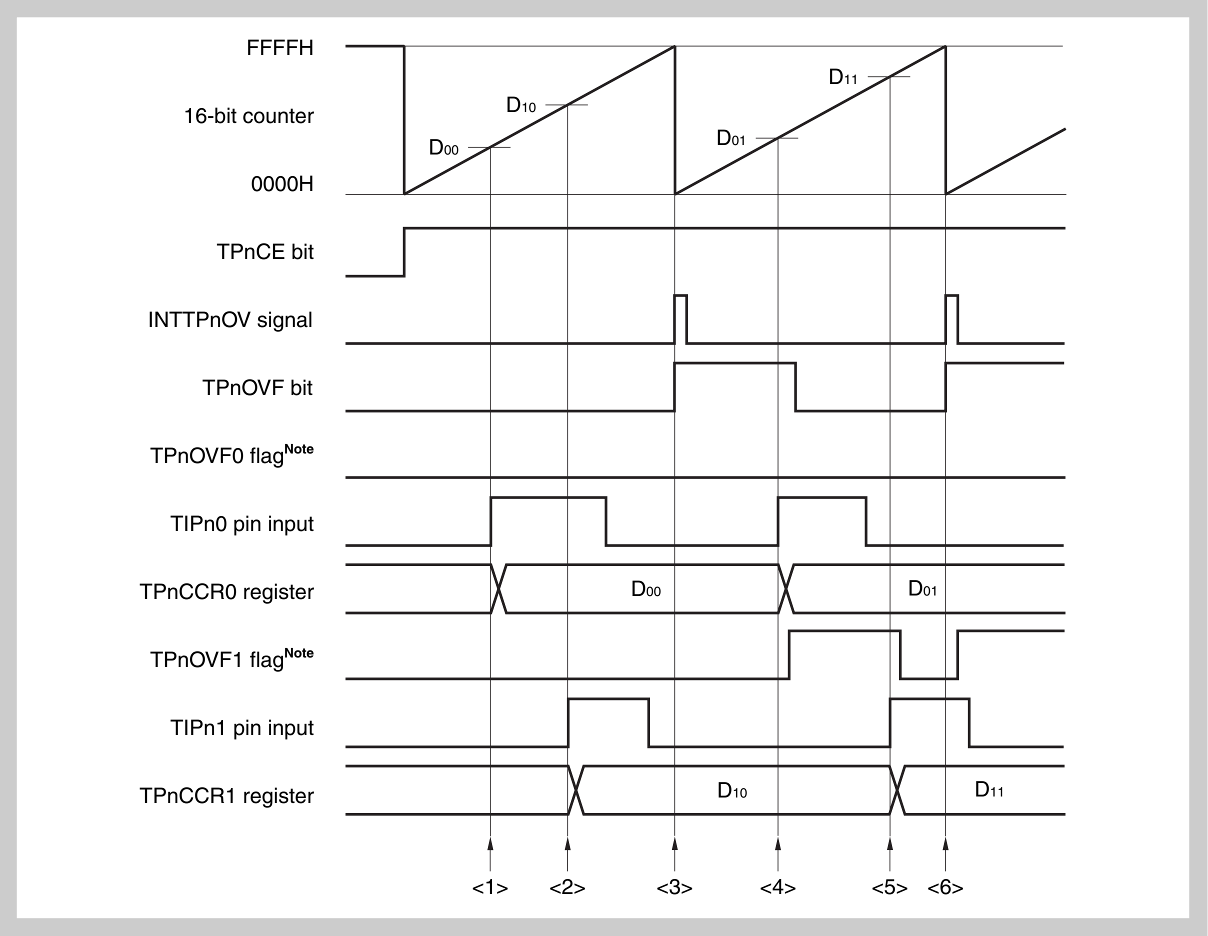


Figure 10-34 Example when two capture registers are used (without using overflow interrupt)

Note The TPnOVF0 and TPnOVF1 flags are set on the internal RAM by software.

<1> Read the TPnCCR0 register (setting of the default value of the TIPn0 pin input).

<2> Read the TPnCCR1 register (setting of the default value of the TIPn1 pin input).

<3> An overflow occurs. Nothing is done by software.

<4> Read the TPnCCR0 register.

Read the overflow flag. If the overflow flag is 1, set only the TPnOVF1 flag to 1, and clear the overflow flag to 0.

Because the overflow flag is 1, the pulse width can be calculated by (10000H + D_{01} - D_{00}).

<5> Read the TPnCCR1 register.

Read the overflow flag. Because the overflow flag is cleared in <4>, 0 is read.

Read the TPnOVF1 flag. If the TPnOVF1 flag is 1, clear it to 0.

Because the TPnOVF1 flag is 1, the pulse width can be calculated by (10000H + D_{11} - D_{10}) (correct).

<6> Same as <3>

(d) Processing of overflow if capture trigger interval is long

If the pulse width is greater than one cycle of the 16-bit counter, care must be exercised because an overflow may occur more than once from the first capture trigger to the next. First, an example of incorrect processing is shown below.

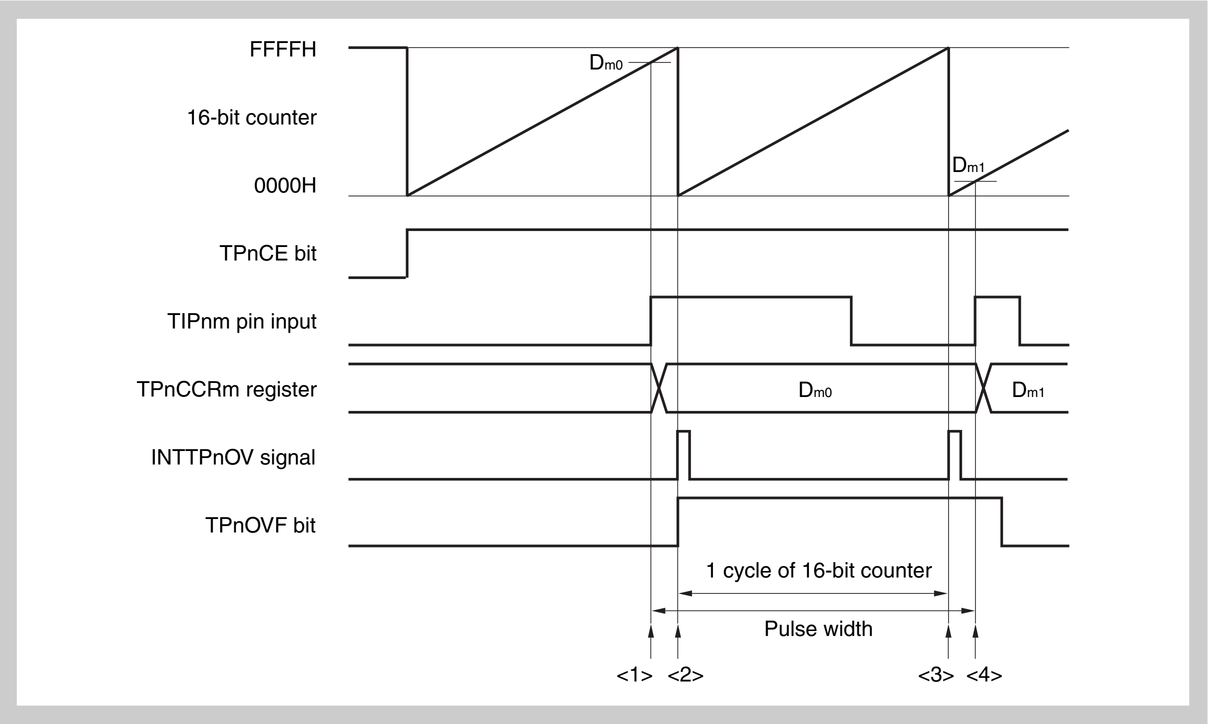


Figure 10-35 Example of incorrect processing when capture trigger interval is long

The following problem may occur when long pulse width is measured in the free-running timer mode.

<1> Read the TPnCCRm register (setting of the default value of the TIPnm pin input).

<2> An overflow occurs. Nothing is done by software.

<3> An overflow occurs a second time. Nothing is done by software.

<4> Read the TPnCCRm register.

Read the overflow flag. If the overflow flag is 1, clear it to 0.

Because the overflow flag is 1, the pulse width can be calculated by (10000H + D_{m1} - D_{m0}) (incorrect).

Actually, the pulse width must be (20000H + D_{m1} - D_{m0}) because an overflow occurs twice.

If an overflow occurs twice or more when the capture trigger interval is long, the correct pulse width may not be obtained.

If the capture trigger interval is long, slow the count clock to lengthen one cycle of the 16-bit counter, or use software. An example of how to use software is shown next.

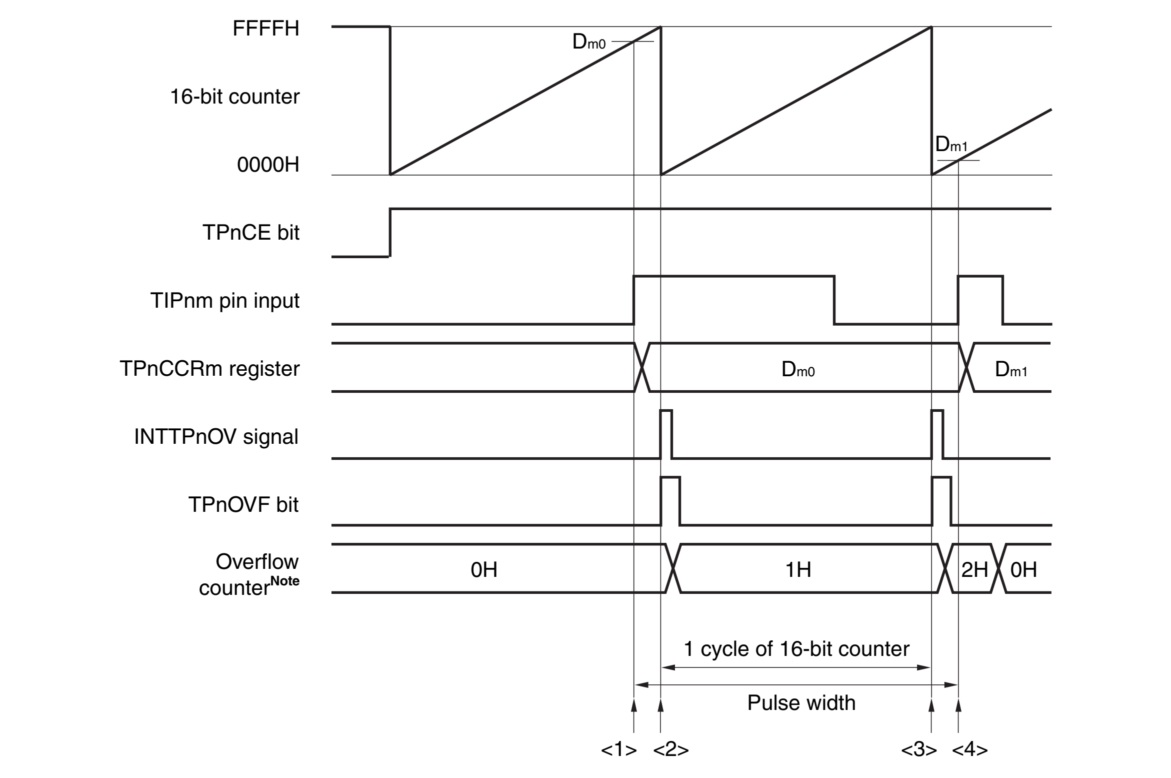


Figure 10-36 Example when capture trigger interval is long

Note The overflow counter is set arbitrarily by software on the internal RAM.

<1> Read the TPnCCRm register (setting of the default value of the TIPnm pin input).

<2> An overflow occurs. Increment the overflow counter and clear the overflow flag to 0 in the overflow interrupt servicing.

<3> An overflow occurs a second time. Increment (+1) the overflow counter and clear the overflow flag to 0 in the overflow interrupt servicing.

<4> Read the TPnCCRm register.

Read the overflow counter.

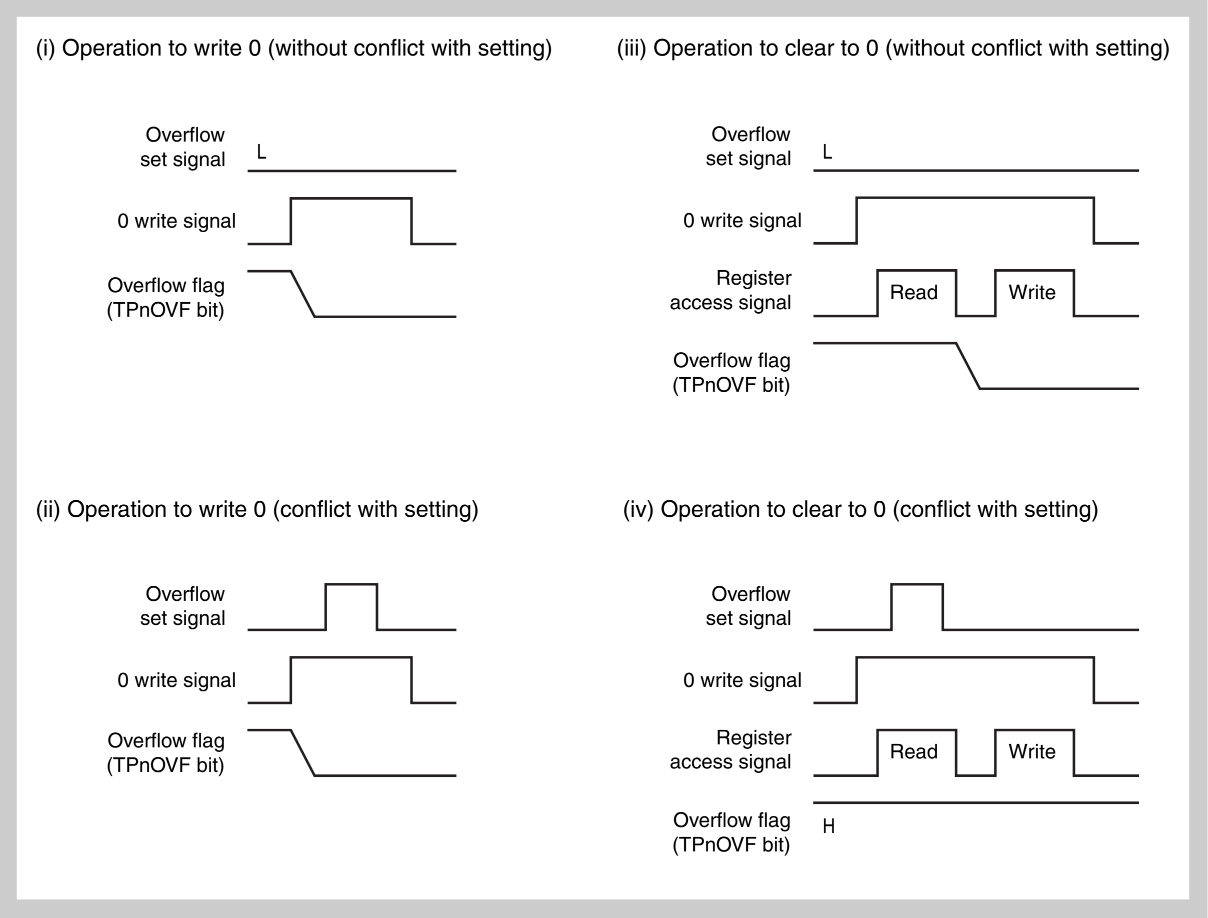
When the overflow counter is “N”, the pulse width can be calculated by (N × 10000H + D_{m1} – D_{m0}).

In this example, the pulse width is (20000H + D_{m1} – D_{m0}) because an overflow occurs twice.

Clear the overflow counter (0H).

(e) Clearing overflow flag

The overflow flag can be cleared to 0 by clearing the TPnOVF bit to 0 with the CLR instruction and by writing 8-bit data (bit 0 is 0) to the TPnOPT0 register. To accurately detect an overflow, read the TPnOVF bit when it is 1, and then clear the overflow flag by using a bit manipulation instruction.


(i) Operation to write 0 (without conflict with setting)
Overflow set signal
L
0 write signal
Overflow flag (TPnOVF bit)
(iii) Operation to clear to 0 (without conflict with setting)
Overflow set signal
L
0 write signal
Register access signal
Read
Write
Overflow flag (TPnOVF bit)
(ii) Operation to write 0 (conflict with setting)
Overflow set signal
0 write signal
Overflow flag (TPnOVF bit)
(iv) Operation to clear to 0 (conflict with setting)
Overflow set signal
0 write signal
Register access signal
Read
Write
Overflow flag (TPnOVF bit)
H

To clear the overflow flag to 0, read the overflow flag to check if it is set to 1, and clear it with the CLR instruction. If 0 is written to the overflow flag without checking if the flag is 1, the set information of overflow may be erased by writing 0 ((ii) in the above chart). Therefore, software may judge that no overflow has occurred even when an overflow actually has occurred.

If execution of the CLR instruction conflicts with occurrence of an overflow when the overflow flag is cleared to 0 with the CLR instruction, the overflow flag remains set even after execution of the clear instruction.

10.5.7 Pulse width measurement mode (TPnMD2 to TPnMD0 = 110)

In the pulse width measurement mode, 16-bit timer/event counter P starts counting when the TPnCTL0.TPnCE bit is set to 1. Each time the valid edge input to the TIPnm pin has been detected, the count value of the 16-bit counter is stored in the TPnCCRm register, and the 16-bit counter is cleared to 0000H.

The interval of the valid edge can be measured by reading the TPnCCRm register after a capture interrupt request signal (INTTPnCCm) occurs.

Select either the TIPn0 or TIPn1 pin as the capture trigger input pin. Specify “No edge detected” by using the TPnIOC1 register for the unused pins.

When an external clock is used as the count clock, measure the pulse width of the TIPn1 pin because the external clock is fixed to the TIPn0 pin. At this time, clear the TPnIOC1.TPnIS1 and TPnIOC1.TPnIS0 bits to 00 (capture trigger input (TIPn0 pin): No edge detected).

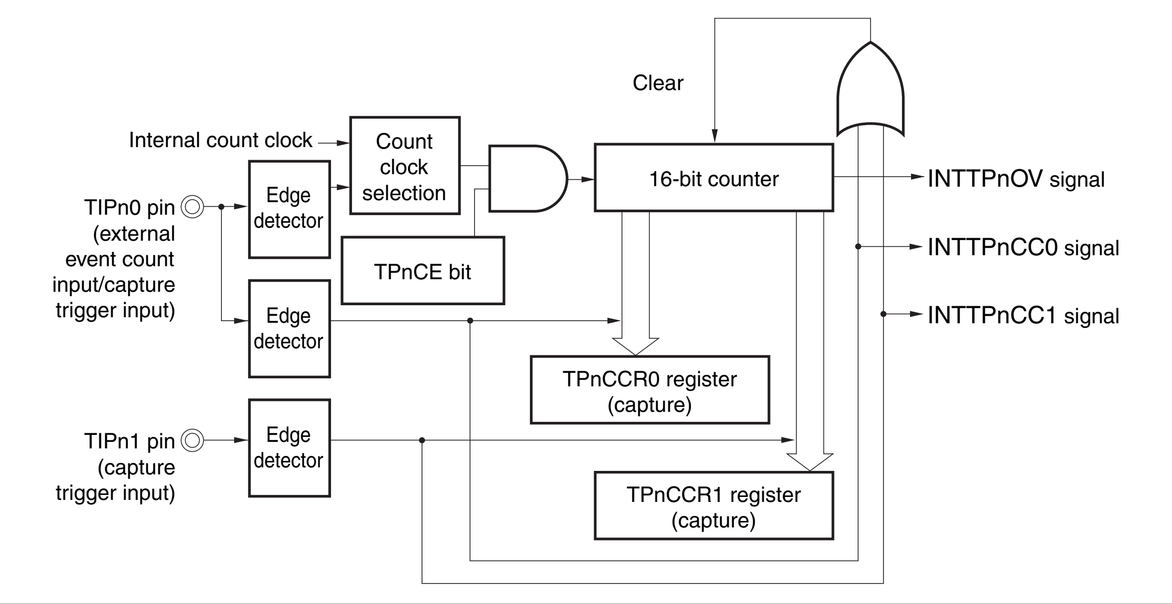


Figure 10-37 Configuration in pulse width measurement mode

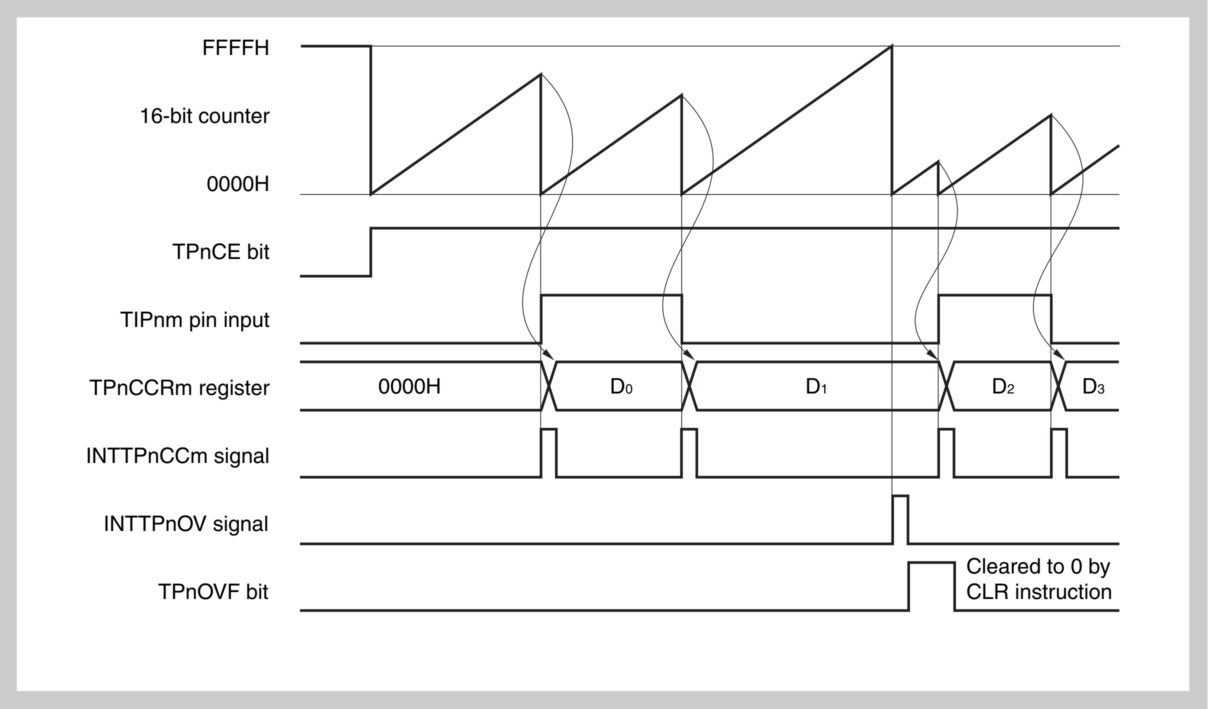


Figure 10-38 Basic timing in pulse width measurement mode

When the TPnCE bit is set to 1, the 16-bit counter starts counting. When the valid edge input to the TIPnm pin is later detected, the count value of the 16-bit counter is stored in the TPnCCRm register, the 16-bit counter is cleared to 0000H, and a capture interrupt request signal (INTTPnCCm) is generated.

The pulse width is calculated as follows.

First pulse width = $(D_0 + 1) \times$ Count clock cycle

Second and subsequent pulse width = $(D_N - D_{N-1}) \times$ Count clock cycle

If the valid edge is not input to the TIPnm pin even when the 16-bit counter counted up to FFFFH, an overflow interrupt request signal (INTTPnOV) is generated at the next count clock, and the counter is cleared to 0000H and continues counting. At this time, the overflow flag (TPnOPT0.TPnOVF bit) is also set to 1. Clear the overflow flag to 0 by executing the CLR instruction via software.

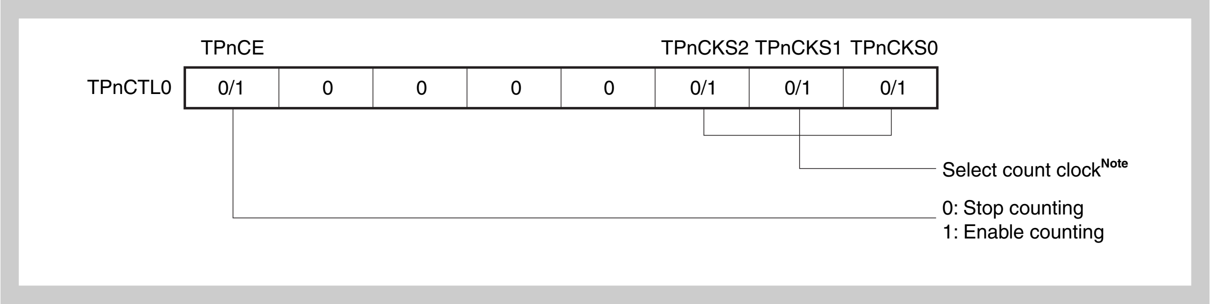
If the overflow flag is set to 1, the pulse width can be calculated as follows.

First pulse width = $(D_0 + 10001H) \times$ Count clock cycle

Second pulse width and on = $(10000H + D_N - D_{N-1}) \times$ Count clock cycle

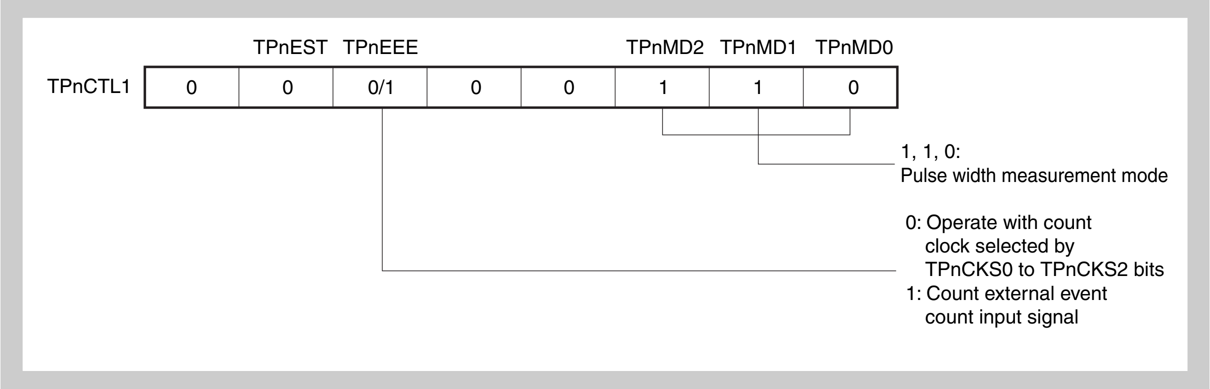
(1) Register setting in pulse width measurement mode

(a) TMPn control register 0 (TPnCTL0)

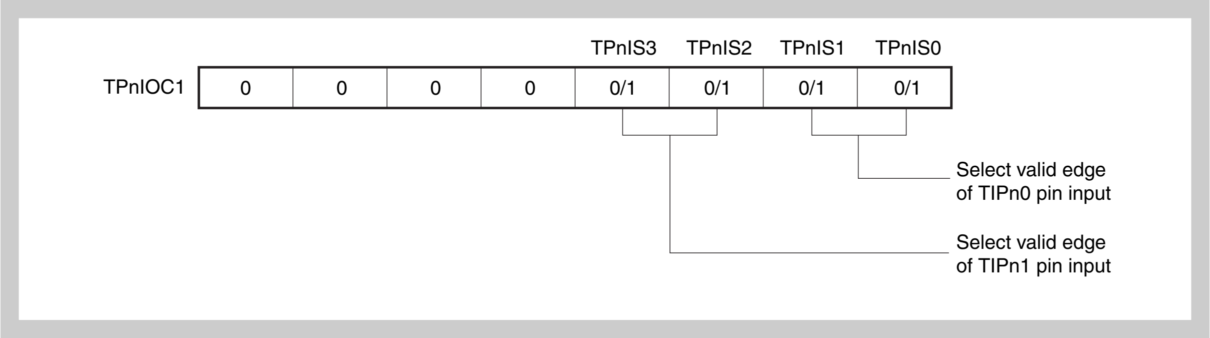


Note Setting is invalid when the TPnEEE bit = 1.

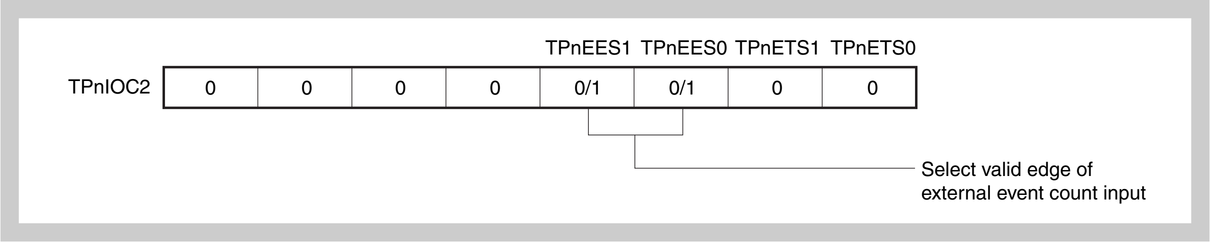
(b) TMPn control register 1 (TPnCTL1)


TPnEST TPnEEE
TPnMD2 TPnMD1 TPnMD0
TPnCTL1
0 0 0/1 0 0 1 1 0
1, 1, 0:
Pulse width measurement mode
0: Operate with count clock selected by TPnCKS0 to TPnCKS2 bits
1: Count external event count input signal

(c) TMPn I/O control register 1 (TPnIOC1)


TPnIS3 TPnIS2 TPnIS1 TPnIS0
TPnIOC1
0 0 0 0 0/1 0/1 0/1 0/1
Select valid edge of TIPn0 pin input
Select valid edge of TIPn1 pin input

(d) TMPn I/O control register 2 (TPnIOC2)


TPnEES1 TPnEES0 TPnETS1 TPnETS0
TPnIOC2
0 0 0 0 0/1 0/1 0 0
Select valid edge of external event count input

(e) TMPn option register 0 (TPnOPT0)

			TPnCCS1	TPnCCS0				TPnOVF
TPnOPT0	0	0	0	0	0	0	0	0/1

TPnOVF: Overflow flag

(f) TMPn counter read buffer register (TPnCNT)

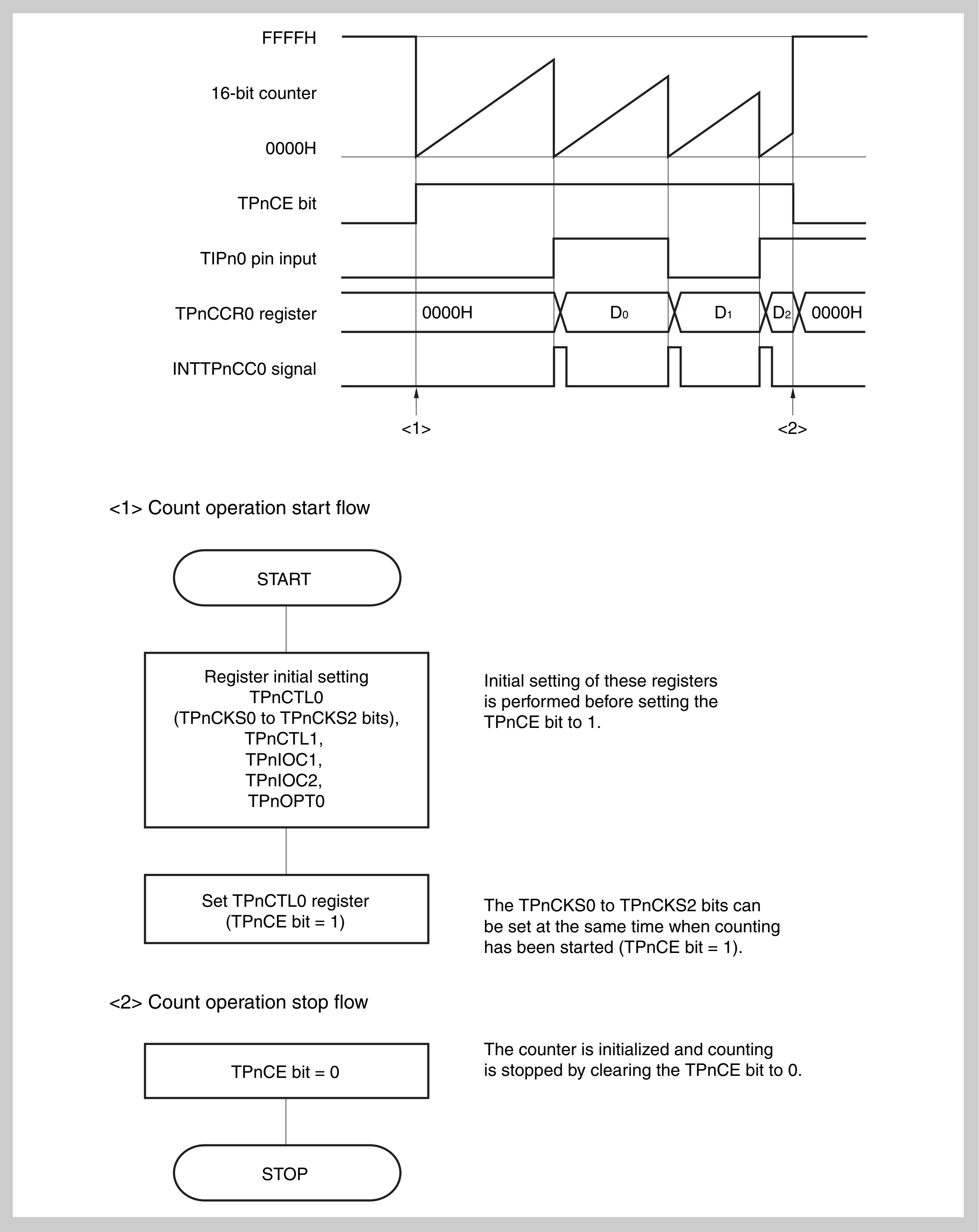
The value of the 16-bit counter can be read by reading the TPnCNT register.

(g) TMPn capture/compare registers 0 and 1 (TPnCCR0 and TPnCCR1)

These registers store the count value of the 16-bit counter when the valid edge input to the TIPnm pin is detected.

Note TMPn I/O control register 0 (TPnIOC0) is not used in the pulse width measurement mode.

(2) Operation flow in pulse width measurement mode



Initial setting of these registers is performed before setting the TPnCE bit to 1.

The TPnCKS0 to TPnCKS2 bits can be set at the same time when counting has been started (TPnCE bit = 1).

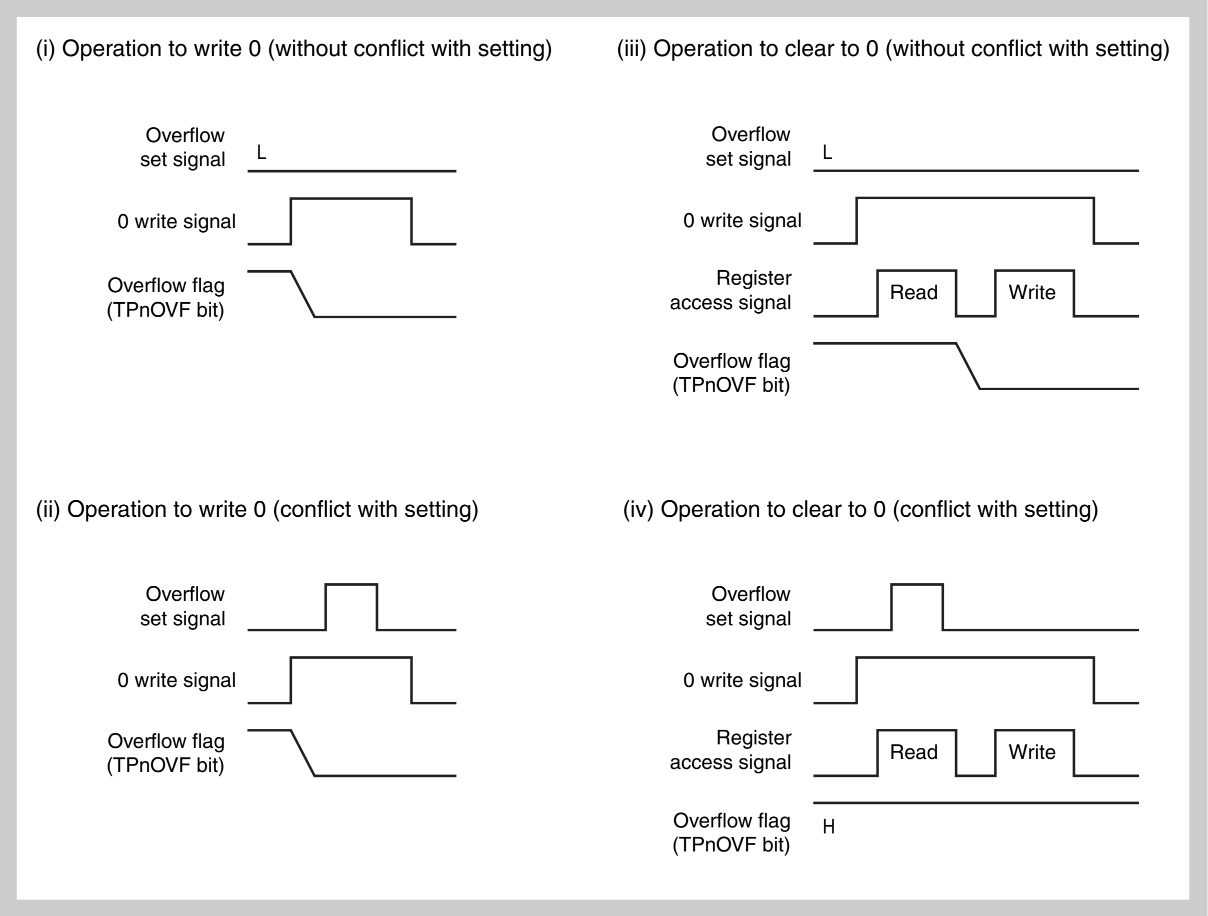
The counter is initialized and counting is stopped by clearing the TPnCE bit to 0.

Figure 10-39 Software processing flow in pulse width measurement mode

(3) Operation timing in pulse width measurement mode

(a) Clearing overflow flag

The overflow flag can be cleared to 0 by clearing the TPnOVF bit to 0 with the CLR instruction and by writing 8-bit data (bit 0 is 0) to the TPnOPT0 register. To accurately detect an overflow, read the TPnOVF bit when it is 1, and then clear the overflow flag by using a bit manipulation instruction.


(i) Operation to write 0 (without conflict with setting)
Overflow set signal
L
0 write signal
Overflow flag (TPnOVF bit)
(iii) Operation to clear to 0 (without conflict with setting)
Overflow set signal
L
0 write signal
Register access signal
Read
Write
Overflow flag (TPnOVF bit)
(ii) Operation to write 0 (conflict with setting)
Overflow set signal
0 write signal
Overflow flag (TPnOVF bit)
(iv) Operation to clear to 0 (conflict with setting)
Overflow set signal
0 write signal
Register access signal
Read
Write
Overflow flag (TPnOVF bit)
H

To clear the overflow flag to 0, read the overflow flag to check if it is set to 1, and clear it with the CLR instruction. If 0 is written to the overflow flag without checking if the flag is 1, the set information of overflow may be erased by writing 0 ((ii) in the above chart). Therefore, software may judge that no overflow has occurred even when an overflow actually has occurred.

If execution of the CLR instruction conflicts with occurrence of an overflow when the overflow flag is cleared to 0 with the CLR instruction, the overflow flag remains set even after execution of the clear instruction.

10.5.8 Timer output operations

The following table shows the operations and output levels of the TOPn0 and TOPn1 pins.

Table 10-11 Timer output control in each mode

Operation Mode	TOPn1 Pin	TOPn0 Pin
Interval timer mode	Square wave output	
External event count mode	Square wave output	–
External trigger pulse output mode	External trigger pulse output	Square wave output
One-shot pulse output mode	One-shot pulse output	
PWM output mode	PWM output	
Free-running timer mode	Square wave output (only when compare function is used)	
Pulse width measurement mode	–	

Table 10-12 Truth table of TOPn0 and TOPn1 pins under control of timer output control bits

TPnIOC0.TPnOLm Bit	TPnIOC0.TPnOEm Bit	TPnCTL0.TPnCE Bit	Level of TOPnm Pin
0	0	×	Low-level output
	1	0	Low-level output
		1	Low level immediately before counting, high level after counting is started
1	0	×	High-level output
	1	0	High-level output
		1	High level immediately before counting, low level after counting is started

10.6 Operating Precautions

10.6.1 Capture operation in pulse width measurement and free-running mode

When the capture operation is used in pulse width measurement or free-running mode the first captured counter value of the capture registers TPnCCR0/ TPnCCR, i.e. after the timer is enabled (TPnCTL0.TPnCE = 1), may be $FFFF_H$ instead of 0000_H if the chosen count clock of the TMP is not the maximum, i.e. if TPnCTL0.TPnCKS[2:0] $\neq$ 0.

10.6.2 Count jitter for PCLK4 to PCLK7 count clocks

When specifying PCLK4 to PCLK7 as the count clock, a jitter of maximum ± 1 period of PCLK0 may be applied to the counter’s count clock input.

Chapter 11 16-bit Interval Timer Z (TMZ)

Timer Z (TMZ) is a general purpose 16-bit timer/counter.

The V850E/Dx3 - DG3 microcontrollers have following instances of the general purpose Timer Z:

TMZ	All devices
Instances	6
Names	TMZ0 to TMZ5

Throughout this chapter, the individual instances of Timer Z are identified by “n”, for example TMZn, or TZnCTL for the TMZn control register.

11.1 Overview

Each Timer Z has one down-counter. When the counter reaches zero, the timer generates the maskable interrupt INTTZnUV.

Features summary The TMZ can be used as:

- Interval timer
- Free running timer

Special features of the TMZ are:

- One of six peripheral clocks can be selected
- One reload register
- Two readable counter registers
- When the device is in debug mode, the timer can be stopped at breakpoint
- TMZ5 can be used for triggering the A/D Converter.

11.1.1 Description

The TMZ has no external connections. It is built up as illustrated in the following figure.

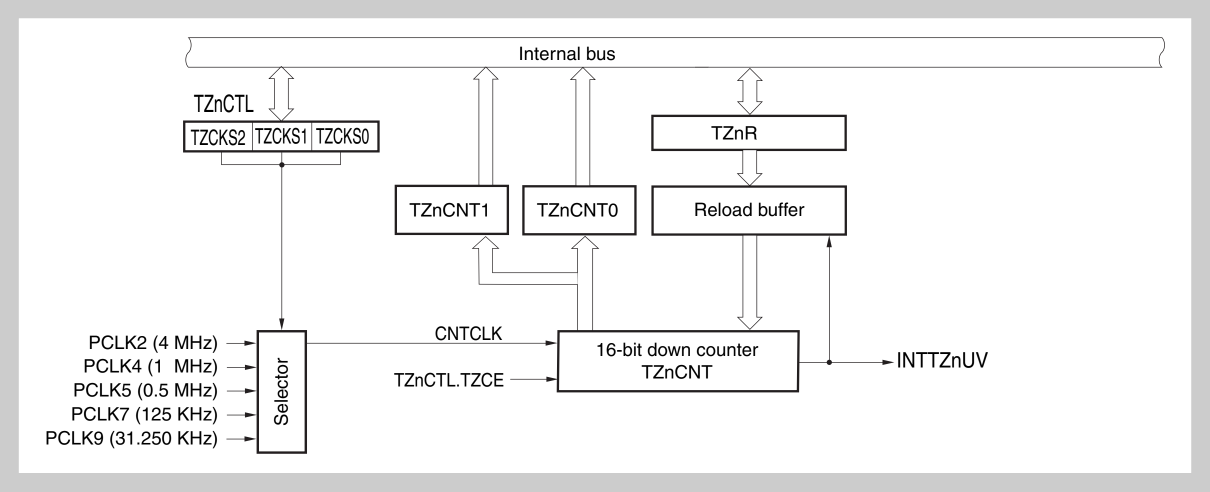


Figure 11-1 Block diagram of Timer Z (TMZn)

The control register TZnCTL allows you to choose the count clock CNTCLK and to enable the timer. The latter is done by setting TZnCTL.TZCE to 1.

As soon as the timer is enabled, it is possible to write a start value to the reload register TZnR.

11.1.2 Principle of operation

When it is enabled, the counter starts as soon as a non-zero value is written to the reload register TZnR and copied to the reload buffer.

When the counter reaches zero, it generates an INTTZnUV interrupt, reloads its start value from the reload buffer, and continues counting.

Two read-only registers (TZnCNT0 and TZnCNT1) provide the updated counter value. For details about these registers please refer to *“TZnCNT0 - TMZn synchronized counter register“ on page 332* and *“TZnCNT1 - TMZn non-synchronized counter register“ on page 333*.

11.2 TMZ Registers

Each Timer Z is controlled and operated by means of the following four registers:

Table 11-1 Timer Z registers overview

Register name	Shortcut	Address
Timer Z synchronized read register	TZnCNT0	<base>
Timer Z non-synchronized read register	TZnCNT1	<base> + 2_H
Timer Z reload register	TZnR	<base> + 4_H
Timer Z control register	TZnCTL	<base> + 6_H

Table 11-2 Base addresses of Timer Z

Timer	Base address
TMZ0	FFFF $F600_H$
TMZ1	FFFF $F608_H$
TMZ2	FFFF $F610_H$
TMZ3	FFFF $F618_H$
TMZ4	FFFF $F620_H$
TMZ5	FFFF $F628_H$

(1) TZnCTL - TMZn timer control register

The 8-bit TZnCTL register controls the operation of the Timer Z.

Access This register can be read/written in 8-bit or 1-bit units.

Address <base> + 6_H

Initial Value 00_H. This register is cleared by any reset.

7	6	5	4	3	2	1	0
TZCE	0	0	0	0	TZCKS2	TZCKS1	TZCKS0
R/W	R	R	R	R	R/W	R/W	R/W

Table 11-3 TZnCTL register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>7</td><td>TZCE</td><td>Timer Z counter enable:
0: Disable count operation (the timer stops immediately with the count value 0000_H and does not operate).
1: Enable count operation (the timer starts when a non-zero start value is written to the register TZnR after TZnCTL.TZCE=1).</td></tr>
<tr><td>2 to 0</td><td>TZCKS[2:0]</td><td>Selects the counter clock CNTCLK:
<table>
<tr><th>TZCKS2</th><th>TZCKS1</th><th>TZCKS0</th><th>Counter clock selection</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>PCLK2 (4 MHz)</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>PCLK4 (1 MHz)</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>PCLK5 (0.5 MHz)</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>PCLK7 (0.125 MHz)</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>PCLK9 (31.250 KHz)</td></tr>
<tr><td colspan="3">Others than above</td><td>Setting prohibited</td></tr>
</table>
</td></tr>
</table>

Note Change bits TZnCTL.TZCKS[2:0] only when TZnCTL.TZCE = 0.

When TZnCTL.TZCE = 0, it is possible to select the clock and enable the counter with one write operation.

(2) TZnCNT0 - TMZn synchronized counter register

The TZnCNT0 register is the synchronized register that can be used to read the present value of the 16-bit counter.

“Synchronized” means that the read access via the internal bus is synchronized with the maximum counter clock (PCLK2). The synchronization process may cause a delay, but the resulting value is reliable.

Access This register is read-only, in 16-bit units.

Address <base> of TMZn

Initial Value 0000_H. This register is cleared by any reset and when TZnCTL.TZCE = 0.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Updated counter value (synchronized)															
R															

Caution Reading TZnCNT0 immediately after start of the counter by setting TZnR > 0 may return 0000_H instead of the correct counter value. Refer to *(4) “TZnR - Reload register“* for details.

(3) TZnCNT1 - TMZn non-synchronized counter register

The TZnCNT1 register is the non-synchronized register that can be used to read the present value of the corresponding 16-bit counter.

“Non-synchronized” means that the read access via the internal bus is not synchronized with the counter clock. It returns the instantaneous value immediately, with the risk that this value is just being updated by the counter and therefore in doubt.

Access This register is read-only, in 16-bit units.

Address <base> + 2_H

Initial Value 0000_H. This register is cleared by any reset and when TZnCTL.TZCE = 0.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Instantaneous counter value (non-synchronized)															
R															

Note The value read from this register can be incorrect, because the read access is not synchronized with the counter clock.

Therefore, this register shall be read multiple times within one period of the counter clock cycle.

If the difference between the first and the second value is not greater than one, you can consider the second value to be correct. If the difference between the two values is greater than one, you have to read the register a third time and compare the third value with the second. Again, the difference must not be greater than one.

If the read accesses do not happen within one period of the counter clock cycle, the difference between the last two values will usually be greater than one. In this case, you can only repeat the procedure or estimate the updated counter value.

Caution Reading TZnCNT1 immediately after start of the counter by setting TZnR > 0 may return 0000_H instead of the correct counter value. Refer to *(4) “TZnR - Reload register“* for details.

(4) TZnR - Reload register

The TZnR register is a dedicated register for setting the reload value of the corresponding counter.

Access This register can be read/written in 16-bit units.

Address <base> + 4_H

Initial Value 0000_H. This register is cleared by any reset and when TZnCTL.TZCE = 0.

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">Down-counter load value</td></tr>
<tr><td colspan="16">R/W</td></tr>
</table>

Note

1. TZnR can only be written when TZnCTL.TZCE = 1.
2. The load value must be non-zero (0001_H … $FFFF_H$).
3. To operate the timer in free running mode, set TZnR to $FFFF_H$.
4. The first interval after starting the counter can require additional clock cycles. For details refer to *"Timer start and stop" on page 336*.
5. Transfer of the TZnR content after writing to TZnR requires additional clock cycles, until the value is set to the counter register TZnCNT. Thus during that transfer time T_{Trans} reading of TZnCNT0 respectively TZnCNT1 returns 0000_H instead of the correct value.
   The transfer time T_{Trans} depends on the chosen count clock and are given in *Table 11-4*.

Table 11-4 Transfer times T_{Trans} of TZnR to TZnCNT

TZnCTL.TZCKS	T_{CNTCLK} period	TznR to TZnCNT transfer time T_{Trans}	
		minimum	maximum
000_B	T_{PCLK2}	T_{PCLK2}	T_{PCLK2}
010_B	4 x T_{PCLK2}	5 x T_{PCLK2}	9 x T_{PCLK2}
011_B	8 x T_{PCLK2}	9 x T_{PCLK2}	17 x T_{PCLK2}
100_B	32 x T_{PCLK2}	33 x T_{PCLK2}	65 x T_{PCLK2}
101_B	128 x T_{PCLK2}	128 x T_{PCLK2}	257 x T_{PCLK2}

11.3 Timing

The contents of the reload register TZnR can be changed at any time, provided the timer is enabled. The contents is then copied to the reload buffer. However, the counter reloads its start value from the buffer when the counter reaches 0.

Caution When specifying PCLK4, PLCK5, PCLK7 or PCLK9 as the count clock, a jitter of maximum ± 1 period of PCLK2 may be applied to the TZnCNT counter's count clock input.

11.3.1 Steady operation

Steady operation is illustrated in the following figure.

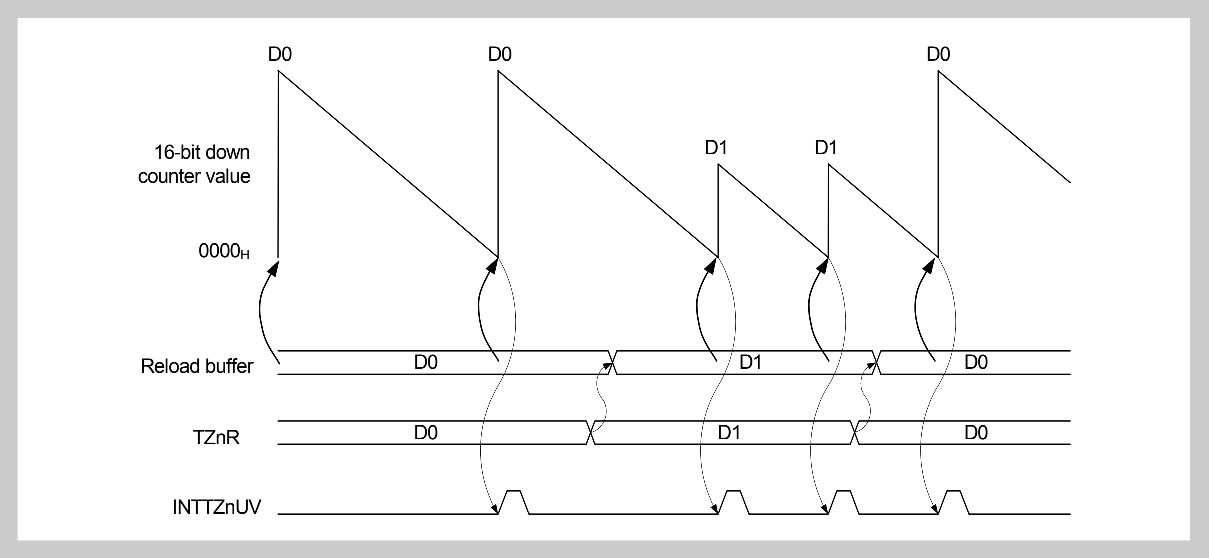


Figure 11-2 Reload timing and interrupt generation

D0 and D1 are two different reload values.

Note that there is a delay between writing to TZnR and making the data available in the reload buffer, depending on the previous reload value and the chosen count clock.

11.3.2 Timer start and stop

(1) Timer Z start

The Timer TZn is enabled by setting TZnCTL.TZCE to 1.

The subsequent write access to register TZnR with non-zero data starts the timer. After that, it is prepared to load the value written to register TZnR into the reload buffer and the counter.

The interval time, i.e. the time between the INTTZnUV interrupts, depends on the chosen count clock T_{CNTCLK} (selected by TZnCTL.TZCKS) and calculates to

$$T_{interval} = ([TZnR] + 1) \times T_{CNTCLK}$$

However the time of the first interval after starting the counter by setting TZnR > 0 may be longer than the steady intervals afterwards.

The length of the first interval also depends on whether the counter has already been enabled a certain time before it’s started.

In the following the interval times for both cases are given as a multiple of T_{PCLK2}, the period of the PCLK2 input clock.

T_{ACmin}, T_{ACmax} Since the access time to the TZnR register adds also to the uncertainty of the first interval duration, the below tables contain two values, which are calculated as follows:

- T_{ACmin} = (SUWL + VSWL + 3) x $1/f_{VBCLK}$
- T_{ACmax} = [2 x (SUWL + VSWL) + 4.5] x $1/f_{VBCLK}$

The values SUWL and VSWL depend on the chosen CPU system clock VBCLK and are set up in the VSWC register (refer to *“Bus and Memory Control (BCU, MEMC)“ on page 255*).
Note that the above access times assume that the NPB bus is not occupied and the write access to TZnR is immediately passed to the Timer Z.

Timer enabled *Table 11-5* shows the interval times under following conditions:

- timer is enabled by TZnCTL.TZCE = 1
- timer is started by setting TZnR > 0 after at least 2 PCLK2 clock periods after timer enable

Table 11-5 TMZ interval times (timer enabled since minimum 2 PCLK2 clocks)

TZnCTL .TZCKS	T_{CNTCLK} period	1st interval		Following intervals
		minimum	maximum	
000_B	T_{PCLK2}	T_{ACmin} + ([TZnR]+2) x T_{PCLK2}	T_{ACmax} + ([TZnR]+3) x T_{PCLK2}	([TZnR]+1) x T_{PCLK2}
010_B	4 x T_{PCLK2}	T_{ACmin} + (4[TZnR]+6) x T_{PCLK2}	T_{ACmax} + (4[TZnR]+11) x T_{PCLK2}	4 x ([TZnR]+1) x T_{PCLK2}
011_B	8 x T_{PCLK2}	T_{ACmin} + (8[TZnR]+10) x T_{PCLK2}	T_{ACmax} + (8[TZnR]+19) x T_{PCLK2}	8 x ([TZnR]+1) x T_{PCLK2}
100_B	32 x T_{PCLK2}	T_{ACmin} + (32[TZnR]+34) x T_{PCLK2}	T_{ACmax} + (32[TZnR]+67) x T_{PCLK2}	32 x ([TZnR]+1) x T_{PCLK2}
101_B	128 x T_{PCLK2}	T_{ACmin} + (128[TZnR]+130) x T_{PCLK2}	T_{ACmax} + (128[TZnR]+259) x T_{PCLK2}	128 x ([TZnR]+1) x T_{PCLK2}

Timer disabled *Table 11-6* shows the interval times under following conditions:

- timer disabled: TZnCTL.TZCE = 0
- timer is enabled by TZnCTL.TZCE = 1
- timer is started by setting TZnR > 0 immediately after enable, i.e. within 2 PCLK2 clock periods after timer enable

Table 11-6 TMZ interval times (timer started within 2 PCLK2 clocks after enable)

TZnCTL .TZCKS	T_{CNTCLK} period	1st interval		Following intervals
		minimum	maximum	
000_B	T_{PCLK2}	T_{ACmin} + ([TZnR]+4.5) x T_{PCLK2}	T_{ACmax} + ([TZnR]+6.5) x T_{PCLK2}	([TZnR]+1) x T_{PCLK2}
010_B	4 x T_{PCLK2}	T_{ACmin} + (4[TZnR]+7.5) x T_{PCLK2}	T_{ACmax} + (4[TZnR]+13.5) x T_{PCLK2}	4 x ([TZnR]+1) x T_{PCLK2}
011_B	8 x T_{PCLK2}	T_{ACmin} + (8[TZnR]+11.5) x T_{PCLK2}	T_{ACmax} + (8[TZnR]+21.5) x T_{PCLK2}	8 x ([TZnR]+1) x T_{PCLK2}
100_B	32 x T_{PCLK2}	T_{ACmin} + (32[TZnR]+35.5) x T_{PCLK2}	T_{ACmax} + (32[TZnR]+69.5) x T_{PCLK2}	32 x ([TZnR]+1) x T_{PCLK2}
101_B	128 x T_{PCLK2}	T_{ACmin} + (128[TZnR]+131.5) x T_{PCLK2}	T_{ACmax} + (128[TZnR]+261.5) x T_{PCLK2}	128 x ([TZnR]+1) x T_{PCLK2}

(2) Timer Z stop

The timer stops when TZnCTL.TZCE is cleared. This write access is not synchronized. The timer is immediately stopped, and its registers are reset.

Chapter 12 16-bit Multi-Purpose Timer G (TMG)

The V850E/Dx3 - DG3 microcontrollers have following instances of the 16-bit multi-purpose Timer G:

TMG	All devices
Instances	2
Names	TMG0 to TMG1

Throughout this chapter, the individual instances of Timer G are identified by “n”, for example TMGn, or TMGMn for the TMGn mode register.

Note Throughout this chapter, the following indexes are used:

n:	for each of the Timer G instances
m = 1 to 4:	for the free assignable Input/Output-channels
x = 0, 1:	for bit-index, i.e. one of the 2 counters of each Timer Gn
y = 0 to 5:	for all of the 6 capture/ compare-channels

12.1 Features of Timer G

Features summary The timers Gn operate as:

- Pulse interval and frequency measurement counter
- Interval timer
- Programmable pulse output
- PWM output timer

One capture input of Timer G0 is connected to the time stamp output of the CAN0 module and can therefore be used for CAN time stamp functions.

12.2 Function Overview of Each Timer Gn

- 16-bit timer/counter (TMGn0, TMGn1): 2 channels
- Bit length
  - Timer Gn registers (TMGn0, TMGn1): 16 bits
- Capture/compare register (GCCny): 6
  - 16-bit
  - 2 registers are assigned fix to the corresponding one of the 2 counters
  - 4 free assignable registers to one of the 2 counters
- Count clock division selectable by prescaler (frequency of peripheral clock: f_{SPCLK0} = 16 MHz)
  - In 8 steps from f_{SPCLK0}/2 to f_{SPCLK0}/256
- Interrupt request sources
  - Edge detection circuit with noise elimination.
  - Compare-match interrupt requests: 6 types
    Perform comparison of capture/compare register with one of the 2 counters (TMGn0, TMGn1) and generate the INTCCGny (y = 0 to 5) interrupt upon compare match.
  - Timer counter overflow interrupt requests: 2 types
    In free run mode the INTTMGn0 (INTTMGn1) interrupt is generated when the count value of TMGn0 (TMGn1) toggles from FFFFH to 0000H.
  - In match and clear mode the INTTMGn0 (INTTMGn1) interrupt is generated when the count value of TMGn0 (TMGn1) matches the GCC0 (GCC1) value.
- PWM output function
  - Control of the outputs of TOGn1 through TOGn4 pin in the compare mode. PWM output can be performed using the compare match timing of the GCCn1 to GCCn4 register and the corresponding timebase (TMGn0, TMGn1).
- Output delay operation
  - A clock-synchronized output delay can be added to the output signal of pins TOGn1 to TOGn4.
  - This is effective as an EMI counter measure.
- Edge detection and noise elimination filter
  - External signals shorter than 1 count clock (f_{COUNTn}, not f_{SPCLK0}) are eliminated as noise.

Note The TIGn1 to TIGn4 and TOGn1 to TOGn4 are each alternative function pins.

The following figure shows the block diagram of Timer Gn.

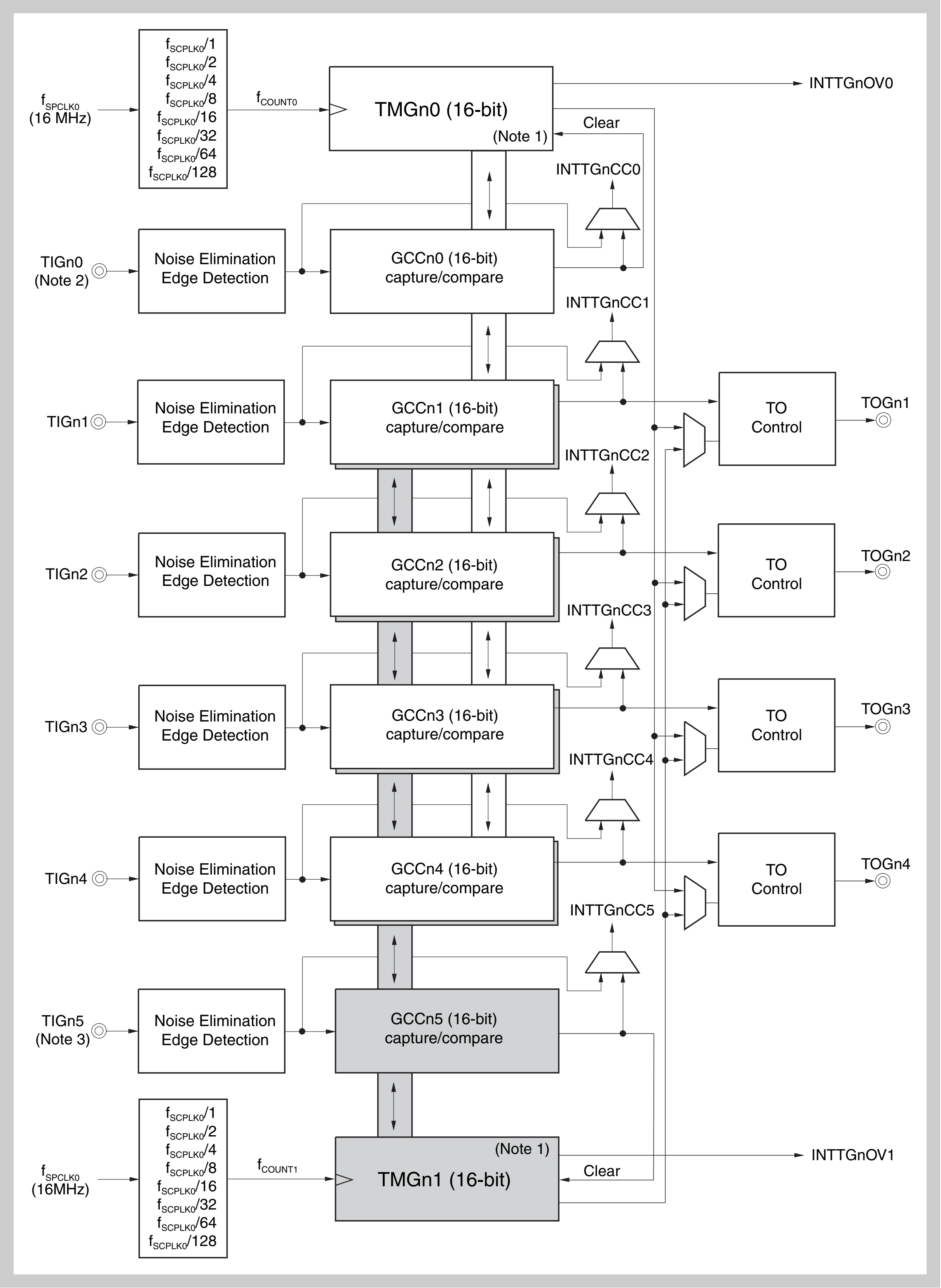


Figure 12-1 Block Diagram of Timer Gn

Note

1. TMGn0/TMGn1 are cleared by GCCn0/GCCn5 register compare match.
2. TIGn0 is not connected
3. TIGn5 differs:
   – n = 0: CAN0 time stamp TSOUTCAN0 -> TIG05
   – n = 1: TIG15 is not connected

12.3 Basic Configuration

The basic configuration is shown below.

Table 12-1 Timer Gn configuration list

Count clock	Register	R/W	Generated interrupt signal	Capture trigger	Timer output PWM
f_{SPCLK0} f_{SPCLK0} /2, f_{SPCLK0} /4, f_{SPCLK0} /8, f_{SPCLK0} /16, f_{SPCLK0} /32, f_{SPCLK0} /64, f_{SPCLK0} /128	TMGn0	R	INTTMGn0	-	-
	TMGn1	R	INTTMGn1	-	-
	GCCn0	R/W	INTCCGn0	TIGn0	-
	GCCn1	R/W	INTCCGn1	TIGn1	TOGn1
	GCCn2	R/W	INTCCGn2	TIGn2	TOGn2
	GCCn3	R/W	INTCCGn3	TIGn3	TOGn3
	GCCn4	R/W	INTCCGn4	TIGn4	TOGn4
	GCCn5	R/W	INTCCGn5	TIGn5	-

Note f_{SPCLK0}: Internal peripheral clock

12.4 TMG Registers

The Timers Gn are controlled and operated by means of the following registers:

Table 12-2 TMGn registers overview

Register name	Shortcut	Address
Timer Gn mode register	TMGMn	<base>
Timer Gn channel mode register	TMGCMn	<base> + 2_H
Timer Gn output control register	OCTLGn	<base> + 4_H
Timer Gn time base status register	TMGSTn	<base> + 6_H
Timer Gn count register 0	TMG00	<base> + 8_H
Timer Gn count register 1	TMG01	<base> + A_H
Timer Gn capture/compare register 0	GCC00	<base> + C_H
Timer Gn capture/compare register 1	GCC01	<base> + E_H
Timer Gn capture/compare register 2	GCC02	<base> + 10_H
Timer Gn capture/compare register 3	GCC03	<base> + 12_H
Timer Gn capture/compare register 4	GCC04	<base> + 14_H
Timer Gn capture/compare register 5	GCC05	<base> + 16_H

Table 12-3 TMGn register base address

Timer	Base address
TMG0	FFFF $F6A0_H$
TMG1	FFFF $F6C0_H$

(1) TMGMn - Timer Gn mode register

Access This register can be read/written in 16-bit, 8-bit or 1-bit units.
The low byte TMGMn.bit[7:0] is accessible separately under the name TMGMnL, the high byte TMGMn.bit[15:8] under the name TMGMnH.

Address TMGMn, TMGMnL:<base>
TMGMnH:<base> + 1_H

Initial Value 0000_H. This register is cleared by any reset.

15	14	13	12	11	10	9	8
POWERn	OLDEn	CSEn12	CSEn11	CSEn10	CSE002	CSEn01	CSEn00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

7	6	5	4	3	2	1	0
CCSGn5	CCSGn0	0	0	CLRGn1	TMGn1E	CLRGn0	TMGn0E
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 12-4 TMGMn register contents (1/2)

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>15</td><td>POWERn</td><td>Timer Gn Operation control.
0: operation Stop
the capture registers and TMGSTn register are cleared
the TOGnm pins are inactive all the time
1: operation enable
Note: At least 7 peripheral clocks (f_{SPCLK0}) are needed to start the timer function</td></tr>
<tr><td>14</td><td>OLDEn</td><td>Set Output Delay Operation.
0: Don’t perform output delay operation
1: Set output delay to n count-clocks
Caution: When the POWERn bit is set, the rewriting of this bit is prohibited! Simultaneously writing with the POWERn bit is allowed.
Note: The delay operation is used for EMI counter measures.</td></tr>
<tr><td>13 to 8</td><td>CSEnx[2:0]</td><td>Selects internal count clock of TMG
<table>
<tr><th>CSEnx2</th><th>CSEnx1</th><th>CSEnx0</th><th>Count clock</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>f_{SPCLK0}</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>$f_{SPCLK0}/2$</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>$f_{SPCLK0}/4$</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>$f_{SPCLK0}/8$</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>$f_{SPCLK0}/16$</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>$f_{SPCLK0}/32$</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>$f_{SPCLK0}/64$</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>$f_{SPCLK0}/128$</td></tr>
</table>
Caution: When the POWERn bit is set, the rewriting of this bits are prohibited! Simultaneously writing with the POWERn bit is allowed.</td></tr>
</table>

Table 12-4 TMGMn register contents (2/2)

Bit position	Bit name	Function
7, 6	CCSGn5 CCSGn0	Specifies the mode of the TMGn0 (TMGn1)(CCSGn5 for TMGn1, CCSGn0 for TMGn0): 0: Free-run mode for TMGn1 (TMGn0), GCCn5 (GCCn0) in capture mode (an detected edge at pin TIGn5 (TIGn0) stores the value of TMGn1 (TMGn0) in GCCn5 (GCCn0) and an interrupt INTCCGn5 (INTCCGn0) is output) 1: Match and Clear mode of the TMGn1 (TMGn0), GCCn5 (GCCn0) in compare mode (when the data of GCCn5 (GCCn0) match the count value of the TMGn1 (TMGn0), the counter is cleared and the interrupt INTCCGn5 (INTCCGn0) occurs) **Caution:** When the POWERn bit is set, the rewriting of this bits are prohibited! Simultaneously writing with the POWERn bit is allowed.
3, 1	CLRGnx	Specifies software clear for TMGnx 0: Continue TMGnx operation 1: Clears (0) the count value of TMGnx, the corresponding TOGnx is deactivated. **Note:** TMGnx starts 1 peripheral-clock after this bit is set this bit is not readable (always read 0)
2, 0	TMGnxE	Specifies TMGnx count operation enable/disable 0: Stop count operation the counter holds the immediate preceding value the corresponding TOGnx is deactivated 1: Enable count operation **Note:** **1.** the counter needs at least 1 peripheral-clock (f_{SPCLK0}) to stop **2.** the counter needs at least 4 peripheral-clocks (f_{SPCLK0}) to start

(2) TMGCMn - Timer Gn channel mode register

This register specifies the assigned counter (TMGn0 or TMGn1) for the GCCnm register.
Furthermore it specifies the edge detection for the TIGny input pins.

Access This register can be read/written in 16-bit, 8-bit or 1-bit units.
The low byte TMGCMn.bit[7:0] is accessible separately under the name TMGCMnL, the high byte TMGCMn.bit[15:8] under the name TMGCMnH.

Address TMGCMn, TMGCMnL:<base> + 2_H
TMGCMnH:<base> + 3_H

Initial Value 0000_H. This register is cleared by any reset.

15	14	13	12	11	10	9	8
TBGn4	TBGn3	TBGn2	TBGn1	IEGn51	IEGn50	IEGn41	IEGn40
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

7	6	5	4	3	2	1	0
IEGn31	IEGn30	IEGn21	IEGn20	IEGn11	IEGn10	IEGn01	IEGn00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 12-5 TMGCMn register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>15 to 12</td><td>TBGnm</td><td>Assigns Capture/Compare registers GCCn1 to GCCn4 to one of the 2 counters TMGn0 or TMGn1:
0: Set TMGn0 as the corresponding counter to GCCnm register and TIGnm/ TOGnm pin
1: Set TMGn1 as the corresponding counter to GCCnm register and TIGnm/ TOGnm pin</td></tr>
<tr><td>11 to 0</td><td>IEGny1,
IEGny0</td><td>Specifies the valid edge of external capture signal input pin (TIGnm) for the capture register performing capture-match with the assigned counter TMGn0 or TMGn1:
<table>
<tr><th>IEGny1</th><th>IEGny0</th><th>Valid edge</th></tr>
<tr><td>0</td><td>0</td><td>Falling edge</td></tr>
<tr><td>0</td><td>1</td><td>Rising edge</td></tr>
<tr><td>1</td><td>0</td><td>No edge detection performed</td></tr>
<tr><td>1</td><td>1</td><td>Both rising and falling edges</td></tr>
</table>
</td></tr>
</table>

(3) OCTLGn - Timer Gn output control register

This register controls the timer output from the TOGnm pin and the capture or compare modus for the GCCnm register.

Access This register can be read/written in 16-bit, 8-bit or 1-bit units. The low byte OCTLGn.bit[7:0] is accessible separately under the name OCTLGnL, the high byte OCTLGn.bit[15:8] under the name OCTLGnH.

Address OCTLGn, OCTLGnL:<base> + 4_H
OCTLGnH:<base> + 5_H

Initial Value 4444_H. This register is cleared by any reset.

15	14	13	12	11	10	9	8
SWFGn4	ALVGn4	CCSGn4	0	SWFGn3	ALVGn3	CCSGn3	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

7	6	5	4	3	2	1	0
SWFGn2	ALVGn2	CCSGn2	0	SWFGn1	ALVGn1	CCSGn1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Caution

1. When the POWERn bit is set, the rewriting of CCSGnm is prohibited
2. When the POWERn bit and TMGn0E bit (TMGn1E bit) are set at the same time, the rewriting of the ALVGnm bits is prohibited.

Table 12-6 OCTLGn register contents

Bit position	Bit name	Function
15, 11, 7, 3	SWFGnm	Fixes the TOGnm pin output level according to the setting of ALVGnm bit. 0: disable TOGnm to inactive level 1: enable TOGnm
14, 10, 6, 2	ALVGnm	Specifies the active level of the TGOnm pin output. 0: Active level is 0 1: Active level is 1 **Caution:** Don’t write this bit, before ENFGn0 or ENFGn1 of TMGSTn is 0, so first clear TMGn0E or TMGn1E bit of the TMGMn register and check ENFGn0 or ENFGn1 bit before writing.
13, 9, 5, 1	CCSGnm	Specifies Capture/Compare mode selection: 0: Capture mode: if external edge is detected the INTCCGnm interrupt occurs, the corresponding counter value is written to GCCnm 1: Compare mode: if GCCnm matches with corresponding timebase the INTCCGnm interrupt occurs, if SWFGm is set the PWM output mode is set **Caution:** Don’t write this bit, before POWERn bit of TMGMnH is 0.

(4) TMGSTn - Time base status register

The TMGSTn register indicates the status of TMGn0 and TMGn1. For the CCFGny bit see *“Operation in Free-Run Mode“ on page 353*.

Access This register can be read in 8-bit or 1-bit units.

Address <base> + 6_H

Initial Value 00_H. This register is cleared by any reset.

7	6	5	4	3	2	1	0
ENFGn1	ENFGn0	CCFGn5	CCFGn4	CCFGn3	CCFGn2	CCFGn1	CCFGn0
R	R	R	R	R	R	R	R

Table 12-7 TMGSTn register contents

Bit position	Bit name	Function
5 to 0	CCFGny	Indicates TMGn0 or TMGn1 overflow status. 0: No overflow 1: Overflow **Caution:** The CCFGny bit is set if a TMGnx overflow has occurred between two capture input signals. This flag is only updated if the corresponding GCCny register was read, so first read the GCCny register and then read this flag if necessary
7 to 6	ENFGnx	Indicates TMGnx operation. 0: indicates operation stopped 1: indicates operation

(5) TMGn0, TMGn1 - Timer Gn 16-bit counter registers

The features of the counters TMGn0 and TMGn1 are listed below:

- Free-running counter that enables counter clearing by compare match of registers GCCn0/GCCn5
- Counter clear can be set by software.
- Counter stop can be set by software.

Access These registers can be read in 16-bit units.

Address TMGn0:<base> + 8_H
TMGn1:<base> + A_H

Initial Value 0000_H. This register is cleared by any reset.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TMGn0/TMGn1 value															
R															

(6) GCCn0, GCCn5 - Timer Gn capture/compare registers of the 2 counters

The GCCn0, GCCn5 registers are 16-bit capture/compare registers of Timer Gn. These registers are fixed assigned to the counter registers:

- GCCn0 is fixed assigned to timebase TMGn0
- GCCn5 is fixed assigned to timebase TMGn1

Capture mode In the *capture register mode*, GCCn0 (GCCn5) captures the TMGn0 (TMGn1) count value if an edge is detected at pin TIGn0 (TIGn5).

Compare mode In the *compare register mode*, GCCn0 (GCCn5) detects match with TMGn0 (TMGn1) and clears the assigned Timebase. So this “match and clear mode” is used to reduce the number of valid bits of the counter TMGn0 (TMGn1).

Caution If in Compare Mode write to this registers *before* POWERn and ENFGnx bit are "1" at the same time.

Access In capture mode, these registers can be read in 16-bit units.
In compare mode, these registers can be read/written in 16-bit units.

Address GCCn0:<base> + C_H
GCCn5:<base> + 16_H

Initial Value 0000_H. These registers are cleared by any reset.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GGCn0/GGCn5 value															
R/W															

(7) GCCn1 to GCCn4 - Timer G capture/compare registers with external PWW-output function

The GCCn1 to GCCn4 registers are 16-bit capture/compare registers of Timer Gn. They can be assigned to one of the two counters either TMGn0 or TMGn1.

Capture mode In the capture register mode, these registers capture the value of TMGn0 when the TBGnm bit (m = 1 to 4) of the TMGCMnH register = 0. When the TBGnm bit = 1, these registers hold the value of TMGn1.

Compare mode In compare mode, these registers represent the actual compare value and the TOGnm-Output (m = 1 to 4) can generate a PWW if they are activated.

Access In capture mode, these registers can be read in 16-bit units.
In compare mode, these registers can be read/written in 16-bit units.

Address GCCn1:<base> + E_H
GCCn2:<base> + 10_H
GCCn3:<base> + 12_H
GCCn4:<base> + 14_H

Initial Value 0000_H. These registers are cleared by any reset.

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">GGCn1 to GGCn4 value</td></tr>
<tr><td colspan="16">R/W</td></tr>
</table>

12.5 Output Delay Operation

When the OLDEn bit is set, different delays of count clock period are added to the TOGnm pins:

Output pin	Delay $1/f_{COUNT}$
TOGn1	0
TOGn2	1
TOGn3	2
TOGn4	3

The figure below shows the timing for the case where the count clock is set to $f_{SPCLK0}/2$. However, 0FFFH is set in GCCn0.

Similar delays are added also when a transition is made from the active to inactive level. So, a relative pulse width is guaranteed.

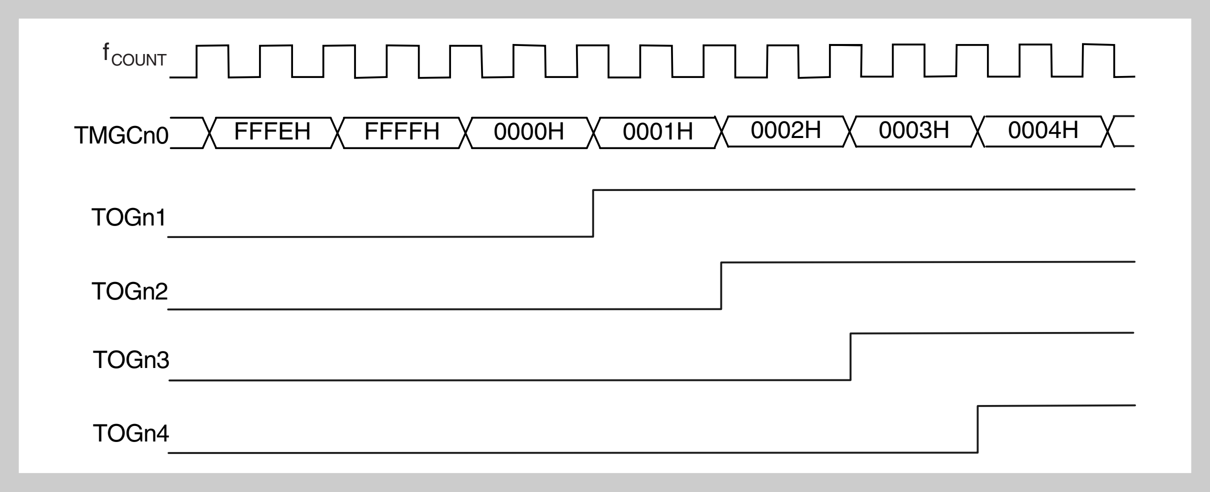


Figure 12-2 Timing of Output delay operation

In this case the count clock is set to $f_{SPCLK0}/2$.

12.6 Explanation of Basic Operation

(1) Overview of the mode settings

The Timer Gn includes 2 channels of 16-bit counters (TMGn0/TMGn1), which can operate as independently timebases. TMGn0 (TMGn1) can be set by CCSGn0 bit (CCSGn5 bit) in the following modes:

- free-run mode
- match and clear mode

When a timer output (TOGnm) or INTCCGnm interrupt is used, one of the two counters can be selected by setting the TBGnm bit (m = 1 to 4) of the TMGCMHn register.

The tables below indicate the interrupt output and timer output states dependent on the register setting values.

Table 12-8 Interrupt output and timer output states dependent on the register setting values

Register setting value				State of each output pin			
CCSGn0	TBGnm	SWFGnm	CCSGnm	**INTTMGn0**	**INTCCGn0**	**INTCCGnm**	**TOGnm**
0 Free-run mode	0	0	0	Overflow interrupt	TI0 edge detection	TIm edge detection	Tied to inactive level
			1			GCCnm match	
		1	0			TIm edge detection	
			1			CMPGm match	PWM (free run)
1 Match and clear mode		0	0	Overflow interrupt[Note 1]	GCCn0 match[Note 2]	TIm edge detection	Tied to inactive level
			1			GCCnm match	
		1	0			TIm edge detection	
			1			CMPGm match	PWM (match and clear)

Note
1. An interrupt is generated only when the value of the GCCn0 register is FFFFH.
2. An interrupt is generated only when the value of the GCCn0 register is not FFFFH.
3. The setting of the CCSGnm bit in combination with the SWFGnm bit sets the mode for the timing of the actualization of new compare values.
   - In compare mode the new compare value will be immediately active.
   - In PWM mode the new compare value will be active first after the next overflow or match & clear of the assigned counter (TMGn0, TMGn1).

Table 12-9 Interrupt output and timer output states dependent on the register setting values

Register setting value				State of each output pin			
CCSGn5	TBGnm	SWFGnm	CCSGnm	**INTTMGn1**	**INTCCGn5**	**INTCCGnm**	**TOGnm**
0 Free-run mode	1	0	0	Overflow interrupt	TI5 edge detection	TIm edge detection	Tied to inactive level
			1			GCCnm match	
		1	0			TIm edge detection	
			1			CMPGm match	PWM (free run)
1 Match and clear mode		0	0	Overflow interrupt[Note 1]	GCCn5 match[Note 2]	TIm edge detection	Tied to inactive level
			1			GCCnm match	
		1	0			TIm edge detection	
			1			CMPGm match	PWM (match and clear)

Note

1. An interrupt is generated only when the value of the GCCn5 register is FFFFH.
2. An interrupt is generated only when the value of the GCCn5 register is not FFFFH.
3. The setting of the CCSGnm bit in combination with the SWFGnm bit sets the mode for the timing of the actualization of new compare values.
   - In compare mode the new compare value will be immediately active.
   - In PWM mode the new compare value will be active first after the next overflow or match & clear of the assigned counter (TMGn0, TMGn1).

12.7 Operation in Free-Run Mode

This operation mode is the standard mode for Timer Gn operations. In this mode the 2 counter TMGn0 and TMGn1 are counting up from 0000H to FFFFH, generates an overflow and start again. In the match and clear mode, which is described in Chapter *12.8 on page 363* the fixed assigned register GCCn0 (GCCn5) is used to reduce the bit-size of the counter TMGn0 (TMGn1).

(1) Capture operation (free run)

Basic settings:

Bit	Value	Remark
CCSGn0	0	free run mode
CCSGn5	0	
SWFGnm	0	disable TOGnm
TBGnm	X	assign counter for GCCnm 0: TMGn0 1: TMGn1

(a) Example: Pulse width or period measurement of the TIGny input signal (free run)

Capture setting method:

(1) When using one of the TOGn1 to TOGn4 pins, select the corresponding counter with the TBGnm bit. When TIGn0 is used, the corresponding counter is TMGn0. When TIGn5 is used, the corresponding counter is TMGn1.

(2) Select a count clock cycle with the CSE12 to CSE10 bits (TMGn1) or CSE02 to CSE02 bits (TMGn0).

(3) Select a valid TIGny edge with the IEGny1 and IEGny0 bits. A rising edge, falling edge, or both edges can be selected.

(4) Start timer operation by setting POWERn bit and TMGn0E bit for TMGn0 or TMGn1E bit for TMGn1.

Capture operation:

(1) When a specified edge is detected, the value of the counter is stored in GCCny and an edge detection interrupt (INTCCGny) is output.

(2) When the counter overflows, an overflow interrupt (INTTMGn0 or INTTMGn1) is generated.

(3) If an overflow has occurred between capture operations, the CCFGny flag is set when GCCny is read. Correct capture data by checking the value of CCFGny.

Using CCFGny:

When using GCCny as a capture register, use the procedure below.

<1> After INTCCGny (edge detection interrupt) generation, read the corresponding GCCny register.

<2> Check if the corresponding CCFGny bit of the TMGSTn register is set.

<3> If the CCFGny bit is set, the counter was cleared from the previous captured value.

CCFGny is set when GCCny is read. So, after GCCny is read, the value of CCFGny should be read. Using the procedure above, the value of CCFGny corresponding to GCCny can be read normally.

Caution If two or more overflows occur between captures, a software-based measure needs to be taken to count overflow interrupts (INTTMGn0, INTTMGn1).

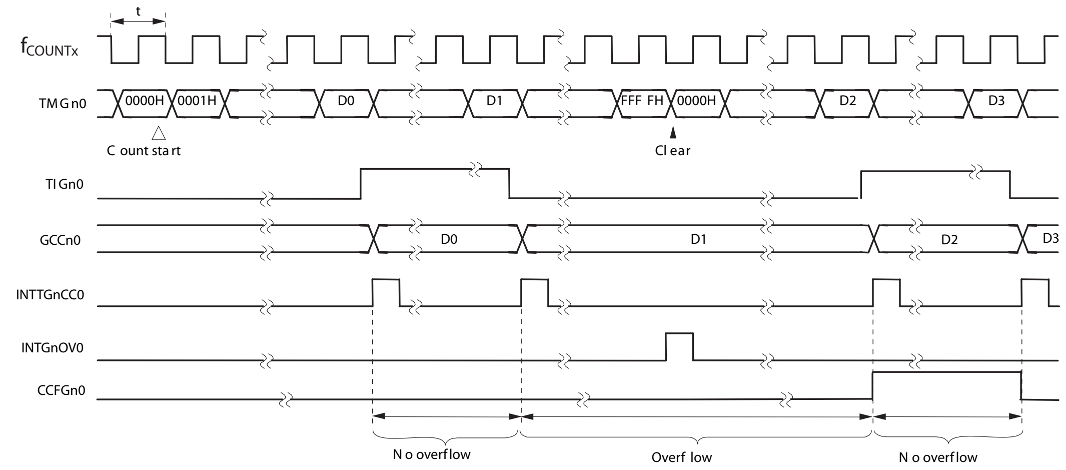


Figure 12-3 Timing when both edges of TIGn0 are valid (free run)

Note The figure above shows an image. In actual circuitry, 3 to 4 periods of the count-up signal are required from the input of a waveform to TIGn0 until a capture interrupt is output.

(b) Timing of capture trigger edge detection

The Tin inputs are fitted with an edge-detection and noise-elimination circuit.

Because of this circuit, 3 periods to less than 4 periods of the count clock are required from edge input until an interrupt signal is output and capture operation is performed. The timing chart is shown below.

Basic settings (x = 0, 1 and y = 0 to 5):

Bit	Value	Remark
CSEx2	0	Count clock = $f_{SPCLK0}/4$
CSEx1	1	
CSEx0	0	
IEGny1	1	detection of both edges
IEGny0	1	

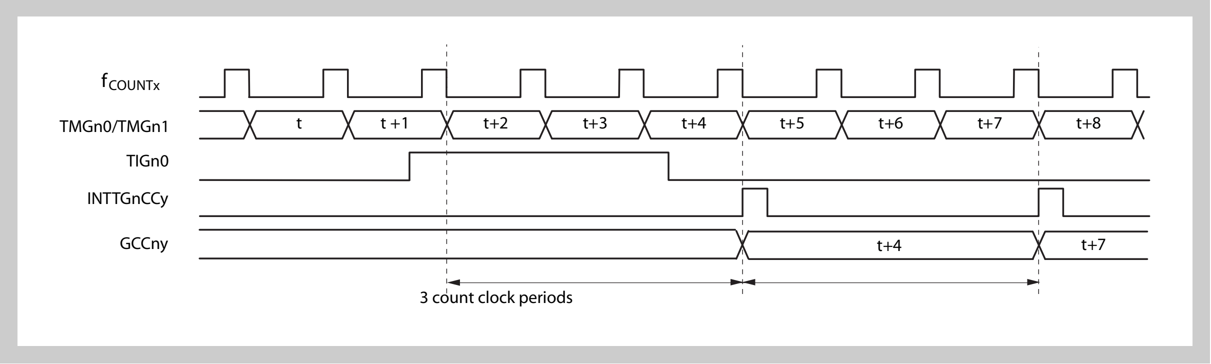


Figure 12-4 Timing of capture trigger edge detection (free run)

(c) Timing of starting capture trigger edge detection

A capture trigger input signal (TIGny) is synchronized in the noise eliminator for internal use.

Edge detection starts when 1 count clock period (f_{COUNT}) has been input after timer count operation starts. (This is because masking is performed to prevent the initial TIGny level from being recognized as an edge by mistake.). The timing chart for starting edge detection is shown below.

Basic settings (x = 0, 1 and y = 0 to 5):

Bit	Value	Remark
CSEx2	0	Count clock = $f_{SPCLK0}/4$
CSEx1	1	
CSEx0	0	
IEGny1	1	detection of both edges
IEGny0	1	

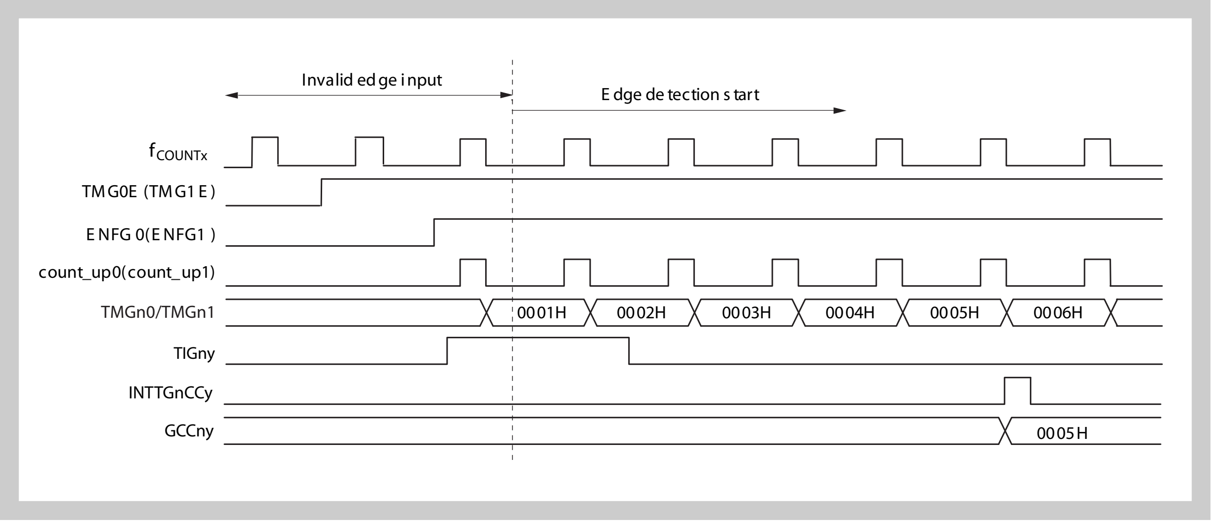


Figure 12-5 Timing of starting capture trigger edge detection

(2) Compare operation (free run)

Basic settings (m = 1 to 4):

Bit	Value	Remark
CCSGn0	0	free run mode
CCSGn5	0	
SWFGnm	0	disable TOGnm
CCSGnm	1	Compare mode for GCCnm
TBGnm	X	assign counter for GCCnm 0: TMGn0 1: TMGn1

(a) Example: Interval timer (free run)

Setting method interval timer:

(1) An usable compare register is one of GCCn1 to GCCn4, and the corresponding counter (TMGn0 or TMGn1) must be selected with the TBGnm bit.

(2) Select a count clock cycle with the CSE12 to CSE10 bits (TMGn1 register) or CSE02 to CSE00 bits (TMGn0 register).

(3) Write data to GCCnm.

(4) Start timer operation by setting POWERn and TMGn0E (or TMGn1E).

Compare Operation:

(1) When the value of the counter matches the value of GCCnm (m = 0 to 4), a match interrupt (INTCCGnm) is output.

(2) When the counter overflows, an overflow interrupt (INTTMGn0/ INTTMGn1) is generated.

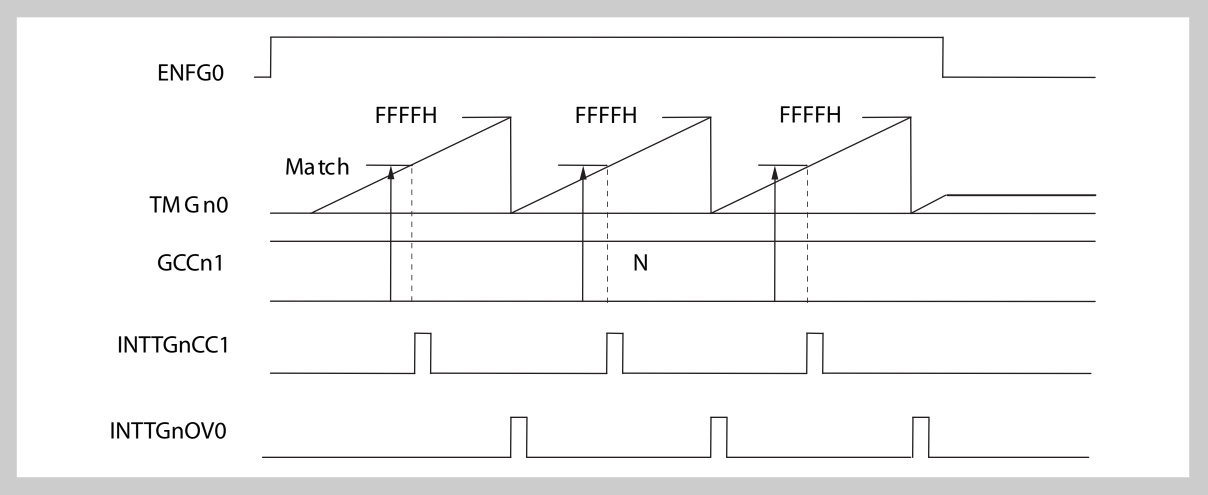


Figure 12-6 Timing of compare mode (free run)

Data N is set in GCCn1, and the counter TMGn0 is selected.

(b) When the value 0000H is set in GCCnm

INTCCGnm is activated when the value of the counter becomes 0001H.

INTTMGn0/INTTMGn1 is activated when the value of the counter changes from FFFFH to 0000H.

Note, however, that even if no data is set in GCCnm, INTCCGnm is activated immediately after the counter starts.

(c) When the value FFFFH is set in GCCnm

INTCCGnm and INTTMGn0/INTTMGn1 are activated when the value of the counter changes from FFFFH to 0000H.

(d) When GCCnm is rewritten during operation

When GCCn1 is rewritten from 5555H to AAAAH. TMGn0 is selected as the counter.

The following operation is performed:

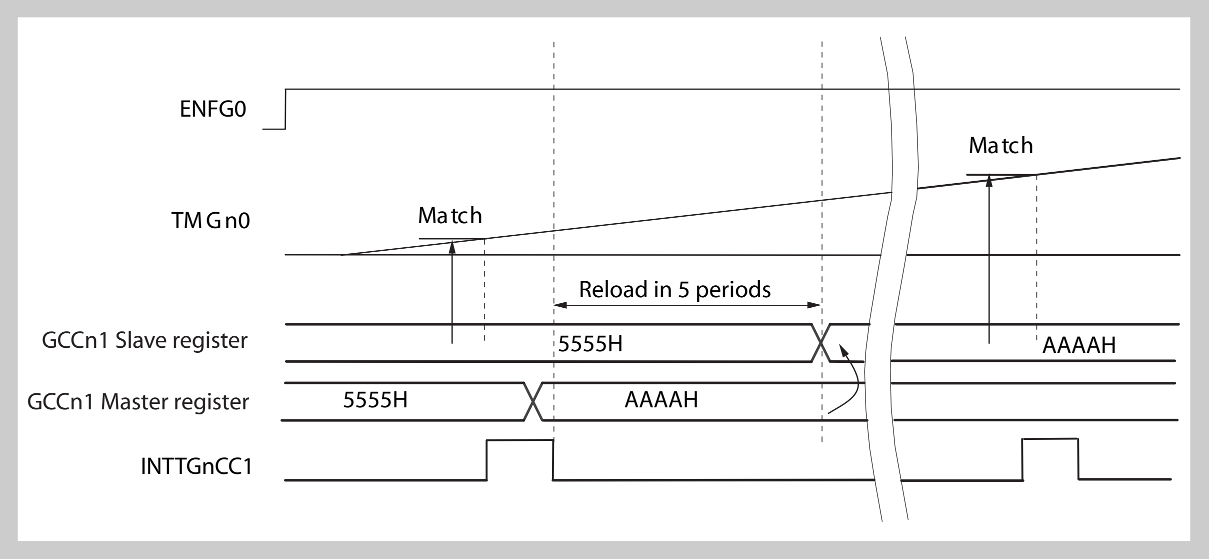


Figure 12-7 Timing when GCCn1 is rewritten during operation (free run)

Caution To perform successive write access during operation, for rewriting the GCCny register (n = 1 to 4), you have to wait for minimum 7 peripheral clocks periods (f_{SPCLK0}).

(3) PWM output (free run)

Basic settings (m = 1 to 4):

Bit	Value	Remark
CCSGn0	0	free run mode
CCSGn5	0	
SWFGnm	1[Note]	enable TOGnm
CCSGnm	1[Note]	Compare mode for GCCnm
TBGnm	X	assign counter for GCCnm 0: TMGn0 1: TMGn1

Note The PWM mode is activated by setting the SWFGnm and the CCSGnm bit to "1".

PWM setting method:

(1) An usable compare register is one of GCCn1 to GCCn4, and the corresponding counter must be selected with the TBGnm bit.

(2) Select a count clock cycle with the CSE12 to CSE10 bits (TMGn1 register) or CSE02 to CSE00 bits (TMGn0 register).

(3) Specify the active level of a timer output (TOGnm pin) with the ALVGnm bit.

(4) When using multiple timer outputs, the user can prevent TOGnm from becoming active simultaneously by setting the OLDEn bit of TMGMHn register to provide step-by-step delays for TOGnm. (This capability is useful for reducing noise and current.)

(5) Write data to GCCnm.

(6) Start timer operation by setting POWERn bit and TMGn0E bit (or TMGn1E bit).

PWM operation:

(1) When the value of the counter matches the value of GCCnm, a match interrupt (INTCCGnm) is output.

(2) When the counter overflows, an overflow interrupt (INTTMGn0 or INTTMGn1) is generated.

(3) TOGnm does not make a transition until the first overflow occurs. (Even if the counter is cleared by software, TOGnm does not make a transition until the next overflow occurs. After the first overflow occurs, TOGnm is activated.

(4) When the value of the counter matches the value of GCCnm, TOGnm is deactivated, and a match interrupt (INTCCGnm) is output. The counter is not cleared, but continues count-up operation.

(5) The counter overflows, and INTTMGn0 or INTTMGn1 is output to activate TOGnm. The counter resumes count-up operation starting with 0000H.

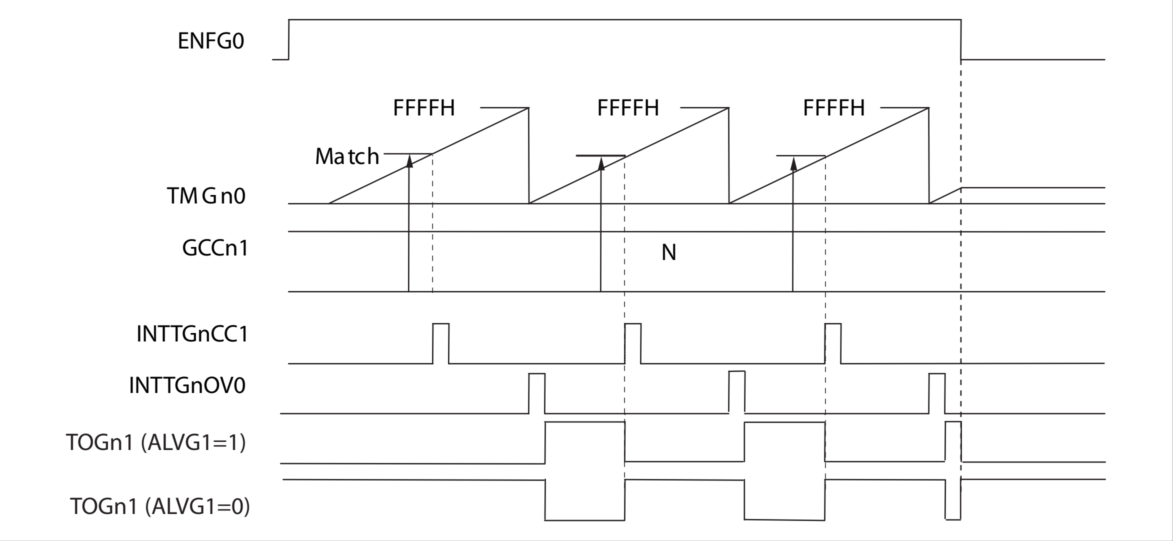


Figure 12-8 Timing of PWM operation (free run)

Data N is set in GCCn1, counter TMGn0 is selected.

(a) When 0000H is set in GCCnm (m = 1 to 4)

When 0000H is set in GCCnm, TOGnm is tied to the inactive level.

The figure below shows the state of TOGn1 when 0000H is set in GCCn1, and TMGn0 is selected.

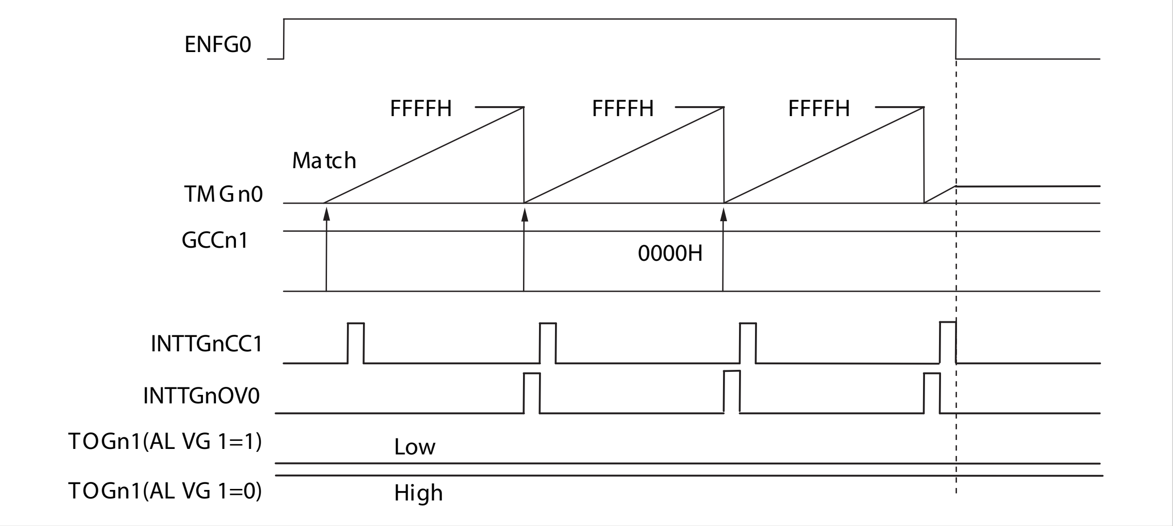


Figure 12-9 Timing when 0000H is set in GCCnm (free run)

GCCn1 and TMGn0 are selected.

(b) When FFFFH is set in GCCnm (m = 1 to 4)

When FFFFH is set in GCCnm, TOGnm outputs the inactive level for one clock period immediately after each counter overflow (except the first overflow).

The figure shows the state of TOGn1 when FFFFH is set in GCCn1, and TMGn0 is selected.

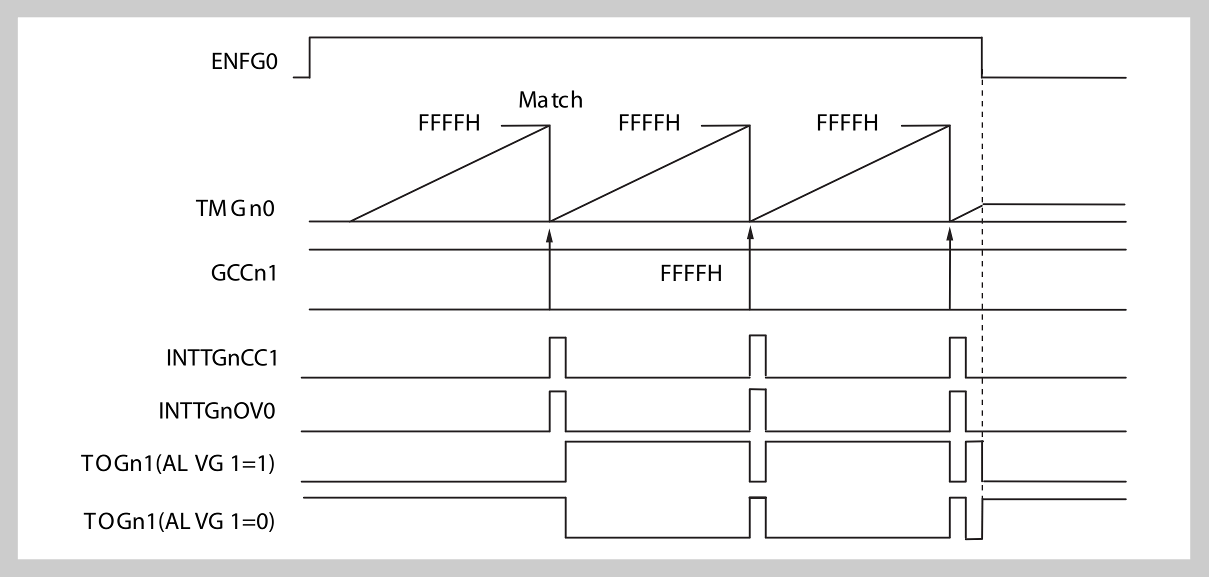


Figure 12-10 Timing when FFFFH is set in GCCnm (free run)

GCCn1 and TMGn0 are selected.

(c) When GCCnm is rewritten during operation (m = 1 to 4)

When GCCn1 is rewritten from 5555H to AAAAH, the operation shown below is performed.

The figure below shows a case where TMGn0 is selected for GCCn1.

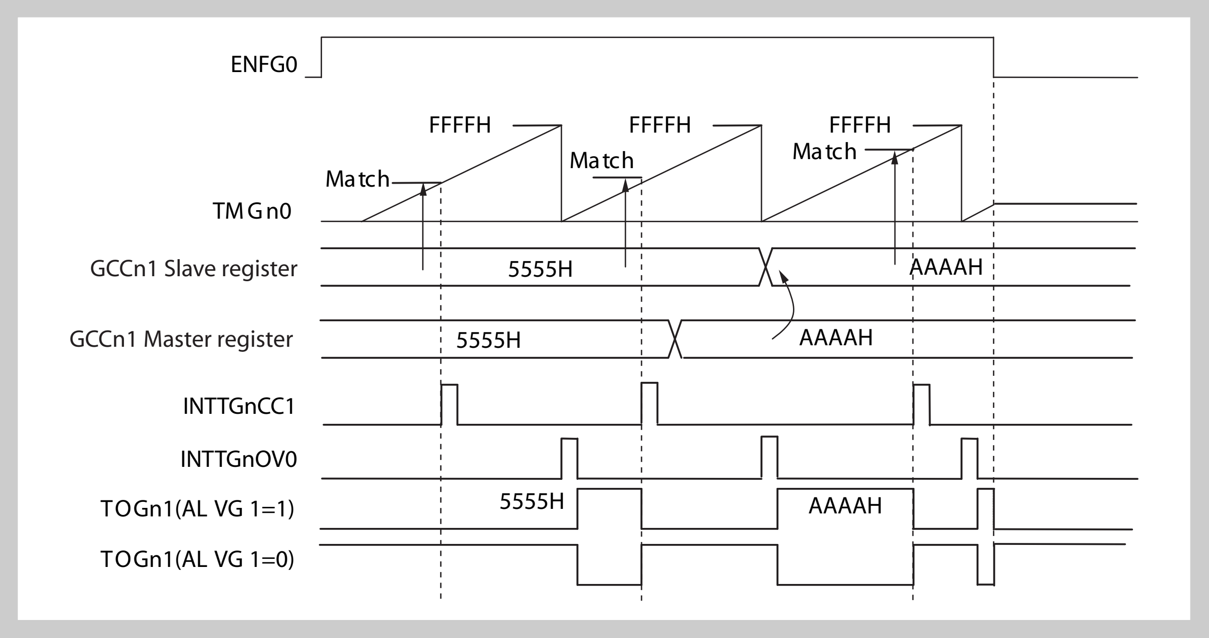


Figure 12-11 Timing when GCCnm is rewritten during operation (free run)

GCCn1 and TMGn0 are selected.

If GCCn1 is rewritten to AAAAH after the second INTCCGn1 is generated as shown in the figure above, AAAAH is reloaded to the GCCn1 register when the next overflow occurs.

The next match interrupt (INTCCGn1) is generated when the value of the counter is AAAAH. The pulse width also matches accordingly.

12.8 Match and Clear Mode

The match and clear mode is mainly used reduce the number of valid bits of the counters (TMGn0, TMGn1).

Therefore the fixed assigned register GCCn0 (GCCn1) is used to compare its value with the counter TMGn0 (TMGn1). If the values match, than an interrupt is generated and the counter is cleared. Than the counter starts up counting again.

(1) Capture operation (match and clear)

Basic settings (m = 1 to 4):

Bit	Value	Remark
CCSGn0	1	match and clear mode
CCSGn5	1	
SWFGnm	0	disable TOGnm
CCSGnm	0	Capture mode for GCCnm
TBGnm	X	assign counter for GCCnm 0: TMGn0 1: TMGn1

(a) Example: Pulse width measurement or period measurement of the TIGnm input signal

Setting method:

(1) When using one of TOGn1 to TOGn4 pin, select the corresponding counter with the TBGnm bit. When CCSGn0 = 1, TI0 cannot be used. When CCSGn5 = 1, TIGn5 cannot be used.

(2) Select a count clock cycle with the CSE12 to CSE10 (TMGn1) bits or CSE02 to CSE00 (TMGn0) bits.

(3) Select a valid TIGnm edge with the IEGnm1 and IEGnm0 bit. A rising edge, falling edge, or both edges can be selected.

(4) Set an upper limit on the value of the counter in GCCn0 or GCCn5.

(5) Start timer operation by setting POWERn bit and TMGn0E bit (or TMGn1E bit).

Operation:

(1) When a specified edge is detected, the value of the counter is stored in GCCnm, and an edge detection interrupt (INTCCGnm) is output.

(2) When the value of GCCn0 or GCCn5 matches the value of the counter, INTCCGn0 (INTCCGn5) is output, and the counter is cleared. This operation is referred to as "match and clear".

(3) If a match and clear event has occurred between capture operations, the CCFGny flag is set when GCCny is read. Correct capture data by checking the value of CCFGny.

(b) Example: Capture where both edges of TIGnm are valid (match and clear)

For the timing chart TMGn0 is selected as the counter corresponding to TOGn1, and 0FFFH is set in GCCn0.

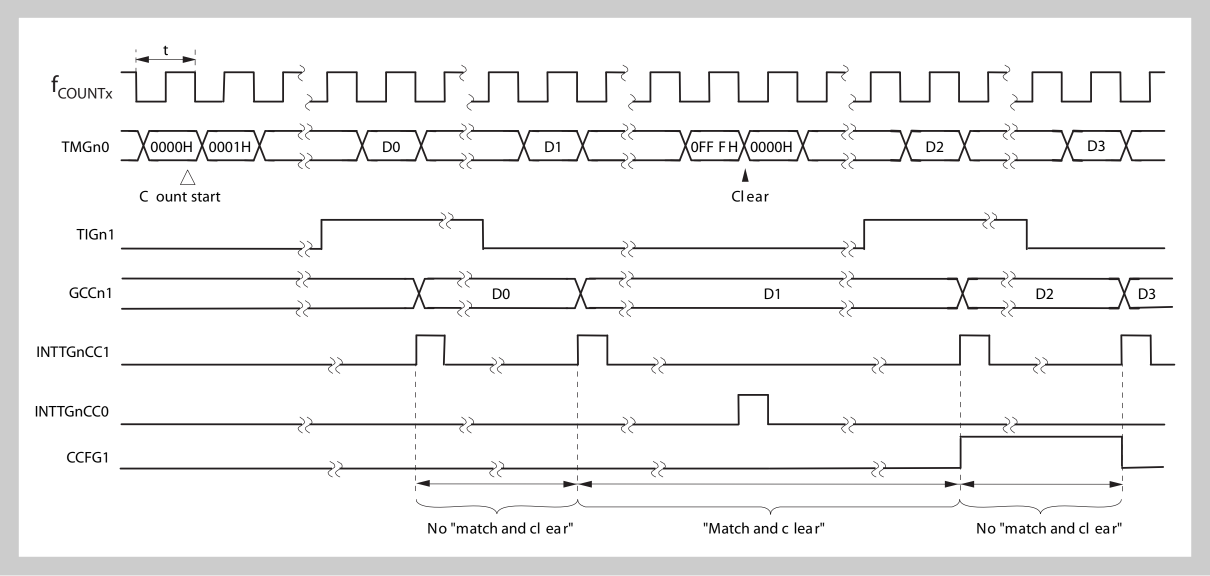


Figure 12-12 Timing when both edges of TIGnm are valid (match and clear)

Note The figure above shows an image. In actual circuitry, 3 to 4 periods of the count-up signal (f_{COUNT}) are required from the input of a waveform to TOGn1 until a capture interrupt is output. (See *Figure 12-4 on page 355*.)

Caution If two or more match and clear events occur between captures, a software-based measure needs to be taken to count INTCCGn0 or INTCCGn5.

(c) When 0000H is set in GCCn0 or GCCn5 (match and clear)

When 0000H is set in GCCn0 (GCCn5), the value of the counter is fixed at 0000H, and does not operate. Moreover, INTCCGn0 (INTCCGn5) continues to be active.

(d) When FFFFH is set in GCCn0 or GCCn5 (match and clear)

When FFFFH is set in GCCn0 (GCCn5), operation equivalent to the free-run mode is performed. When an overflow occurs, INTTMGn0 (INTTMGn1) is generated, but INTCCGn0 (INTCCGn5) is not generated.

(2) Compare operation (match and clear)

Basic settings (m = 1 to 4):

Bit	Value	Remark
CCSGn0	1	match and clear mode
CCSGn5	1	
SWFGnm	0	disable TOGnm
CCSGnm	1	Compare mode for GCCnm
TBGnm	X	assign counter for GCCnm 0: TMGn0 1: TMGn1

(a) Example: Interval timer (match and clear)

Setting Method

(1) An usable compare register is one of GCCn1 to GCCn4, and the corresponding counter must be selected with the TBGnm bit.

(2) Select a count clock cycle with the CSE12 to CSE10 bits (TMGn1) or CSE02 to CSE00 bits (TMGn0).

(3) Set an upper limit on the value of the counter in GCCn0 or GCCn5.

(4) Write data to GCCnm.

(5) Start timer operation by setting the POWERn bit and TMGxE bit (x = 0, 1).

Operation:

(1) When the value of the counter matches the value of GCCnm, a match interrupt (INTCCGnm) is output.

(2) When the value of GCCn0 or GCCn5 matches the value of the counter, INTCCGn0 (or INTCCGn5) is output, and the counter is cleared. This operation is referred to as "match and clear".

(3) The counter resumes count-up operation starting with 0000H.

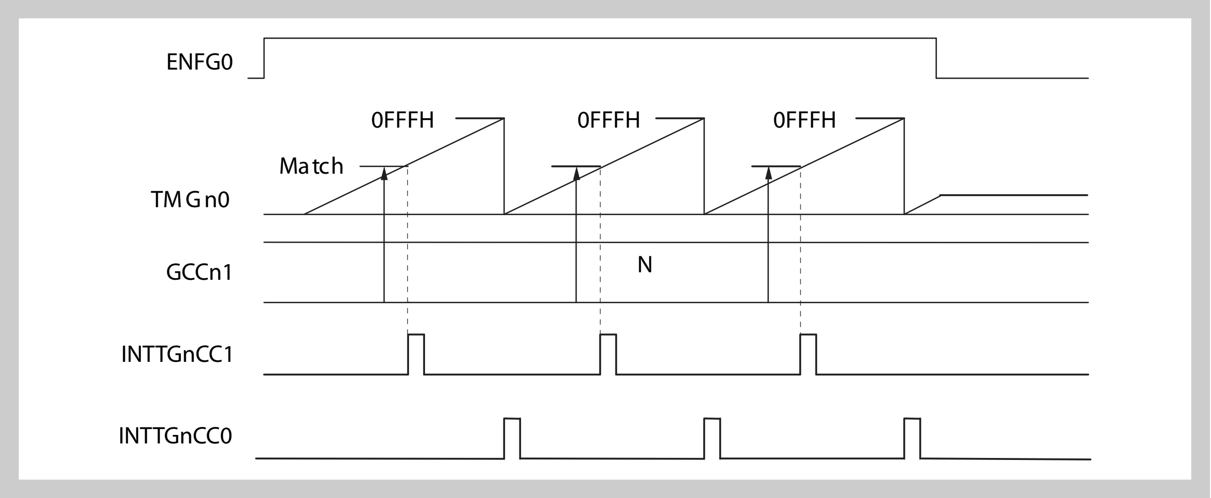


Figure 12-13 Timing of compare operation (match and clear)

In this example, the data N is set in GCCn1, and TMGn0 is selected.

0FFFH is set in GCCn0. Here, N < 0FFFH.

(b) When 0000H is set in GCCn0 or GCCn5 (match and clear)

When 0000H is set in GCCn0 or GCCn5, the value of the counter is fixed at 0000H, and does not operate. Moreover, INTCCGn0 (or INTCCGn5) continues to be active.

(c) When FFFFH is set in GCCn0 or GCCn5 (match and clear)

When FFFFH is set in GCCn0 or GCCn5, operation equivalent to the free-run mode is performed. When an overflow occurs, INTTMGn0 (or INTTMGn1) is generated, but INTCCGn0 (or INTCCGn5) is not generated.

(d) When 0000H is set in GCCnm (m = 1 to 4) (match and clear)

INTCCGnm is activated when the value of the counter becomes 0001H.

Note, however, that even if no data is set in GCCnm, INTCCGnm is activated immediately after the counter starts.

(e) When a value exceeding the value of GCCn0 or GCCn5 is set in GCCnm (m = 1 to 4) (match and clear)

INTCCGnm is not generated.

(f) When GCCnm (m = 1 to 4) is rewritten during operation (match and clear)

When the value of GCCn1 is changed from 0555H to 0AAAH, the operation described below is performed.

TMGn0 is selected as the counter, and 0FFFH is set in GCCn0.

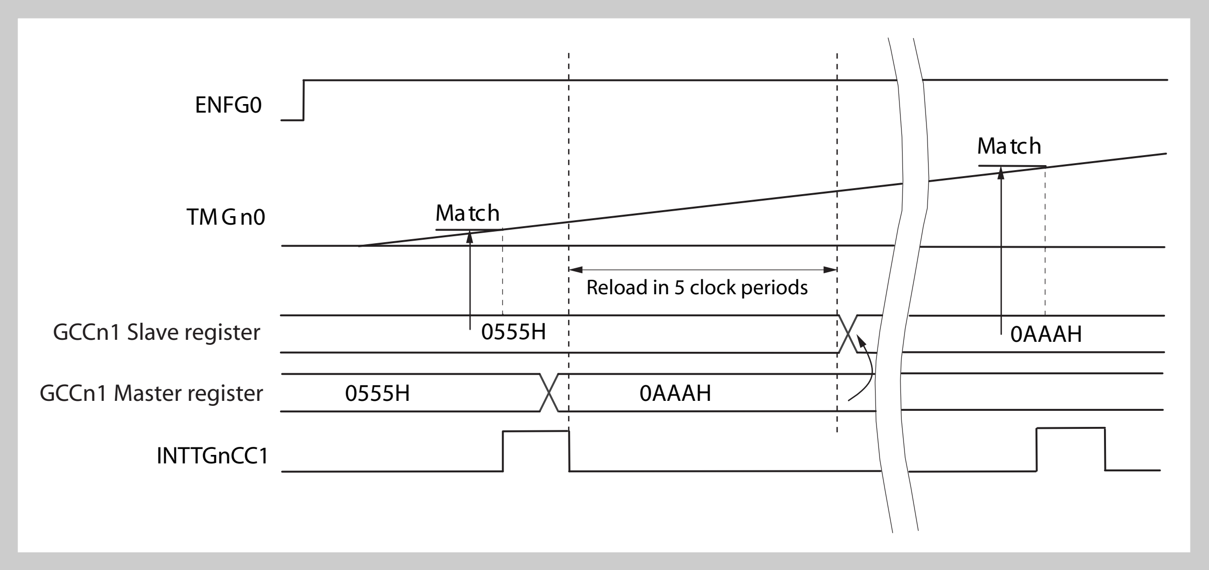


Figure 12-14 Timing when GCCnm is rewritten during operation (match and clear)

Caution To perform successive write access during operation, for rewriting the GCCny register, you have to wait for minimum 7 peripheral clocks periods (f_{SPCLK0}).

(3) PMW output (match and clear)

Basic settings (m = 1 to 4):

Bit	Value	Remark
CCSGn0	1	match and clear mode
CCSGn5	1	
SWFGnm	1 Note	enable TOGnm
CCSGnm	1 Note	Compare mode for GCCnm
TBGnm	X	assign counter for GCCnm 0: TMGn0 1: TMGn1

Note The PWM mode is activated by setting the SWFGnm and the CCSGnm bit to "1".

Setting Method:

(1) An usable compare register is one of GCCn1 to GCCn4, and the corresponding counters TMGn0 or TMGn1 must be selected with the TBGnm bit (m = 1 to 4).

(2) Select a count clock cycle with the CSE12 to CSE10 (TMGn1) bits or CSE02 to CSE00 (TMGn0) bits.

(3) Specify the active level of a timer output (TOGnm) with the ALVGnm bit.

(4) When using multiple timer outputs, the user can prevent TOGnm from making transitions simultaneously by setting the OLDEn bit of TMGMHn register. (This capability is useful for reducing noise and current.)

(5) Set an upper limit on the value of the counter in GCCn0 or GCCn5. (Timer Dn 0000H is forbidden)

(6) Write data to GCCnm.

(7) Start count operation by setting POWERn bit and TMGn0E bit (or TMGn1E bit).

Operation of PWM (match and clear):

(1) When the value of the counter matches the value of GCCnm, a match interrupt (INTCCGnm) is output.

Caution Do not set 0000H in GCCn0 or GCCn5 in match and clear modus.

(2) When the value of GCCn0 (GCCn5) matches the value of the counter, INTCCGn0 (INTCCGn5) is output, and the counter is cleared. This operation is referred to as "match and clear".

(3) TOGnm does not make a transition until the first match and clear event.

(4) TOGnm makes a transition to the active level after the first match and clear event.

(5) When the value of the counter matches the value of GCCnm, TOGnm makes a transition to the inactive level, and a match interrupt (INTCCGnm) is output.

(6) When the next match and clear event occurs, INTCCGn0 (INTCCGn5) is output, and the counter is cleared. The counter resumes count-up operation starting with 0000H.

Example Data N is set, and the counter TMGn0 is selected.
0FFFH is set in GCCn0 and N < 0FFFH.

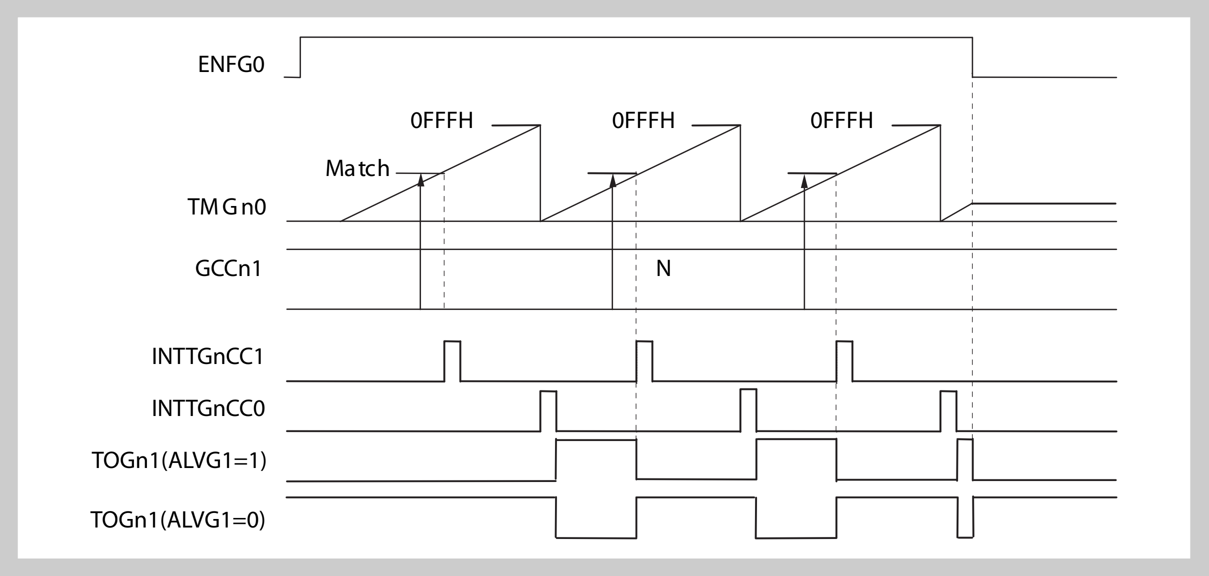


Figure 12-15 Timing of PWM operation (match and clear)

When 0000H is set in GCCn0 (GCCn5), the value of the counter is fixed at 0000H, and the counter does not operate. The waveform of INTCCGn0 (INTCCGn5) varies, depending on whether the count clock is the reference clock or the sampling clock.

(a) When FFFFH is set in GCCn0 or GCCn5 (match and clear)

When FFFFH is set in GCCn0 (GCCn5), operation equivalent to the free-run mode is performed. When an overflow occurs, INTTMGn0 (INTTMGn1) is generated, but INTCCGn0 (INTCCGn5) is not generated.

(b) When 0000H is set in GCCnm (match and clear)

When 0000H is set in GCCnm, TOGnm is tied to the inactive level.

The figure below shows the state of TOGn1 when 0000H is set in GCCn1, and TMGn0 is selected.

Note, however, that 0FFFH is set in GCCn0.

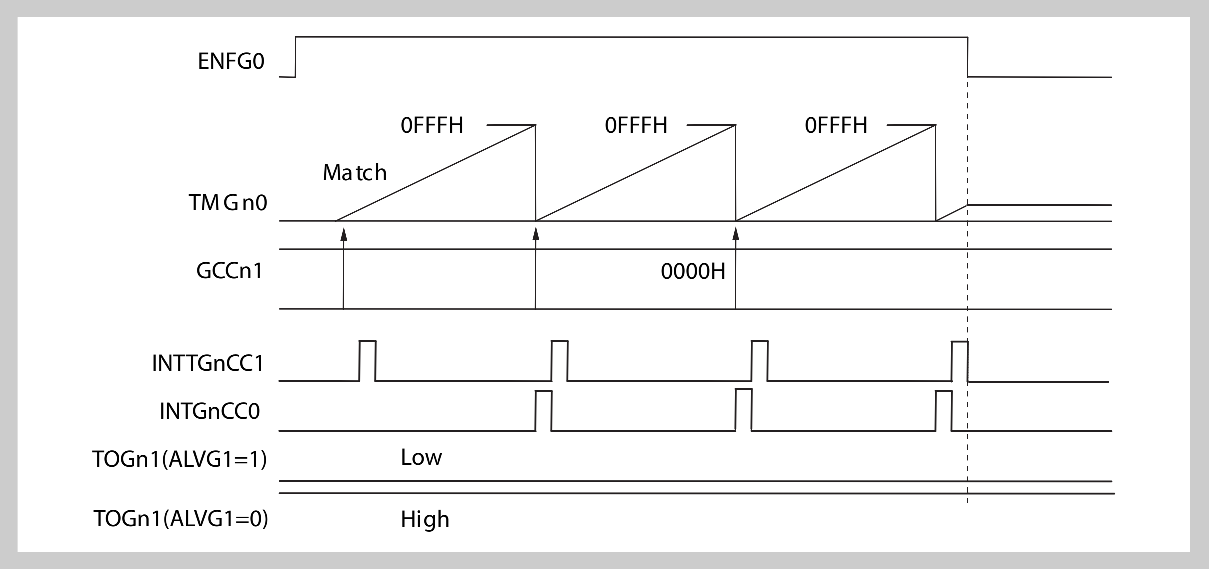


Figure 12-16 Timing when 0000H is set in GCCnm (match and clear)

(c) When the same value as set in GCCn0 or GCCn5 is set in GCCnm (match and clear)

When the same value as set in GCCn0 (GCCn5) is set in GCCnm, TOGnm outputs the inactive level for only one clock period immediately after each match and clear event (excluding the first match and clear event).

The figure below shows the state of TOGn1 when 0FFFH is set in GCCn0 and GCCn1, and TMGn0 is selected.

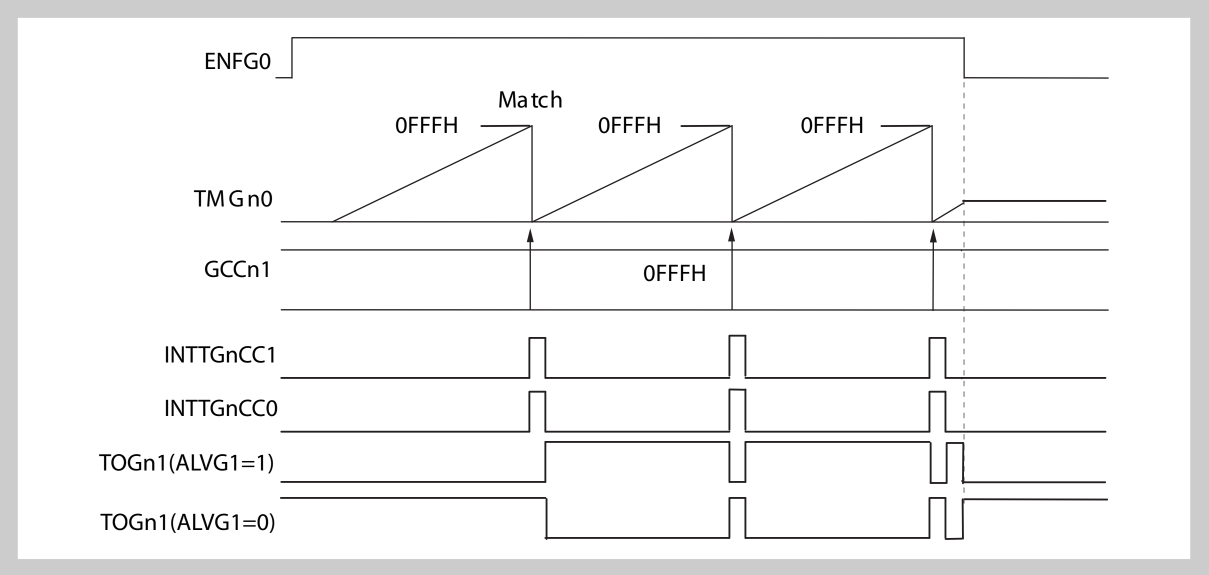


Figure 12-17 Timing when the same value as set in GCCn0/GCCn5 is set in GCCnm (match and clear)

(d) When a value exceeding the value set in GCCn0 or GCCn5 is set in GCCnm (match and clear)

When a value exceeding the value set in GCCn0 (GCCn5) is set in GCCnm, TOGnm starts and continues outputting the active level immediately after the first match and clear event (until count operation stops.)

The figure shows the state of TOGn1 when 0FFFH is set in GCCn0, 1FFFH is set in GCCn1, and TMGn0 is selected.

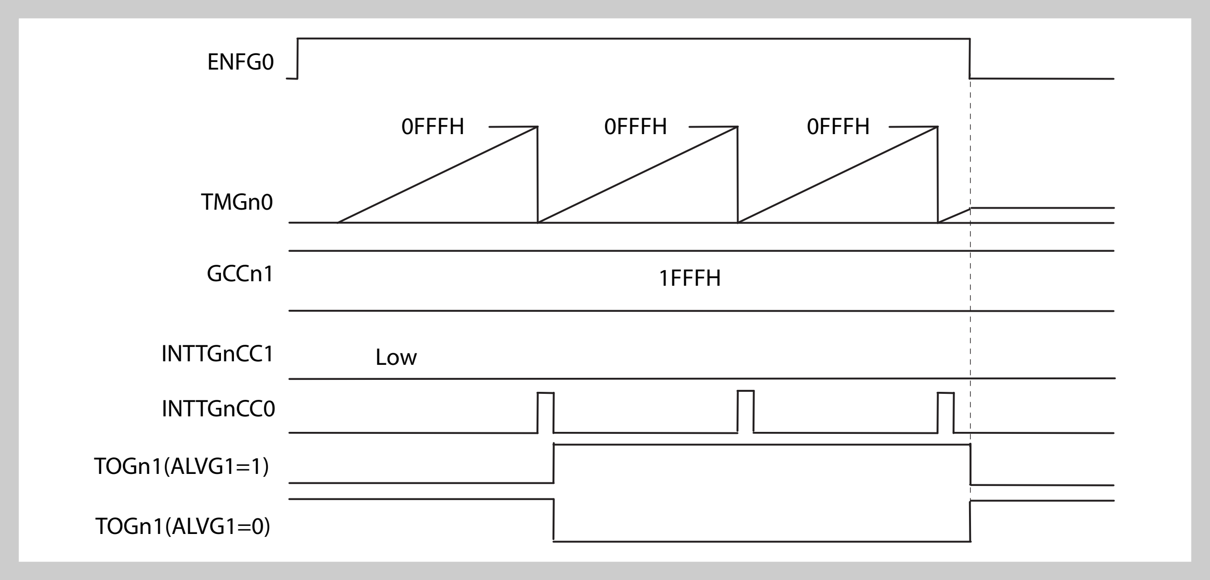


Figure 12-18 Timing when the value of GCCnm exceeding GCCn0 or GCCn5 (match and clear)

(e) When GCCnm is rewritten during operation (match and clear)

When GCCn1 is rewritten from 0555H to 0AAAH, the operation shown below is performed.

The figure below shows a case where 0FFFH is set in GCCn0, and TMGn0 is selected for GCCn1.

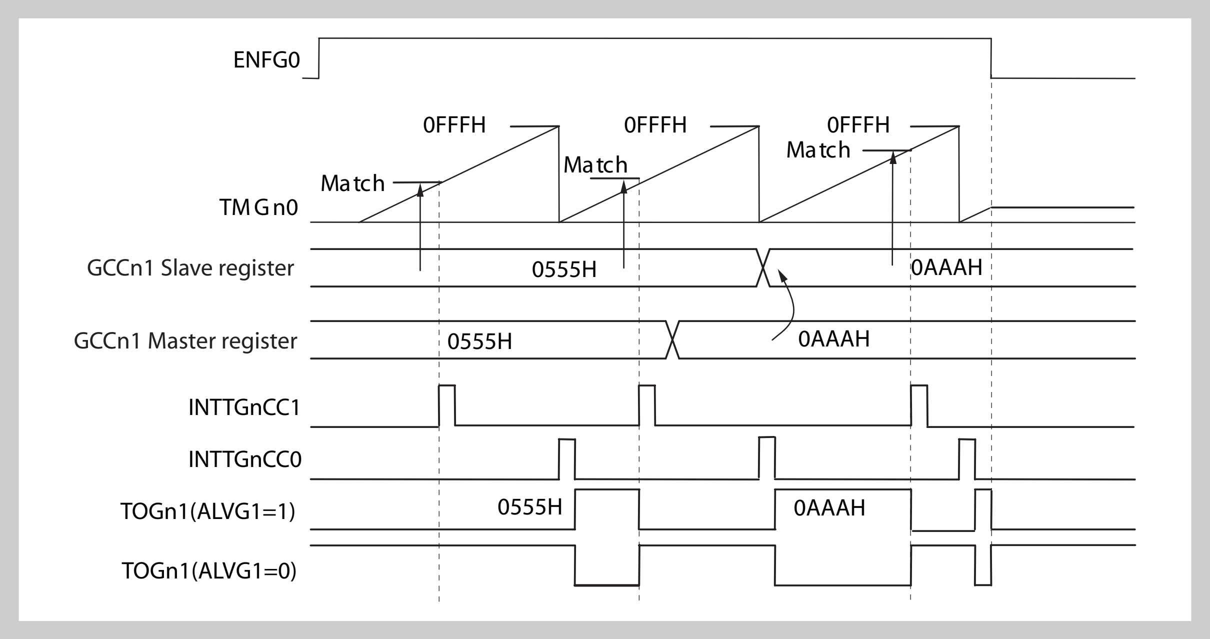


Figure 12-19 Timing when GCCnm is rewritten during operation (match and clear)

If GCCn1 is rewritten to 0AAAH after the second INTCCGn1 is generated as shown in the figure above, 0AAAH is reloaded to the GCCn1 register when the next overflow occurs.

The next match interrupt (INTCCGn1) is generated when the value of the counter is 0AAAH. The pulse width also matches accordingly.

12.9 Edge Noise Elimination

The edge detection circuit has a noise elimination function. This function regards:

- a pulse **not wider than 1 count clock** period as a **noise**, and does not detect it as an edge.
- a pulse **not shorter than 2 count clock** periods is detected normally as an **edge**.
- a pulse **wider than 1 count clock period but shorter than 2 count clock** periods may be **detected as an edge or may be eliminated as noise**, depending on the timing.

(This is because the count-up signal of the counter is used for sampling timing.) The upper figure below shows the timing chart for performing edge detection. The lower figure below shows the timing chart for not performing edge detection.

Basic settings (x = 0, 1 and y = 0 to 5):

Bit	Value	Remark
CSEx2	0	Count clock = $f_{SPCLK0}/4$
CSEx1	1	
CSEx0	0	
IEGny1	1	detection of both edges
IEGny0	1	

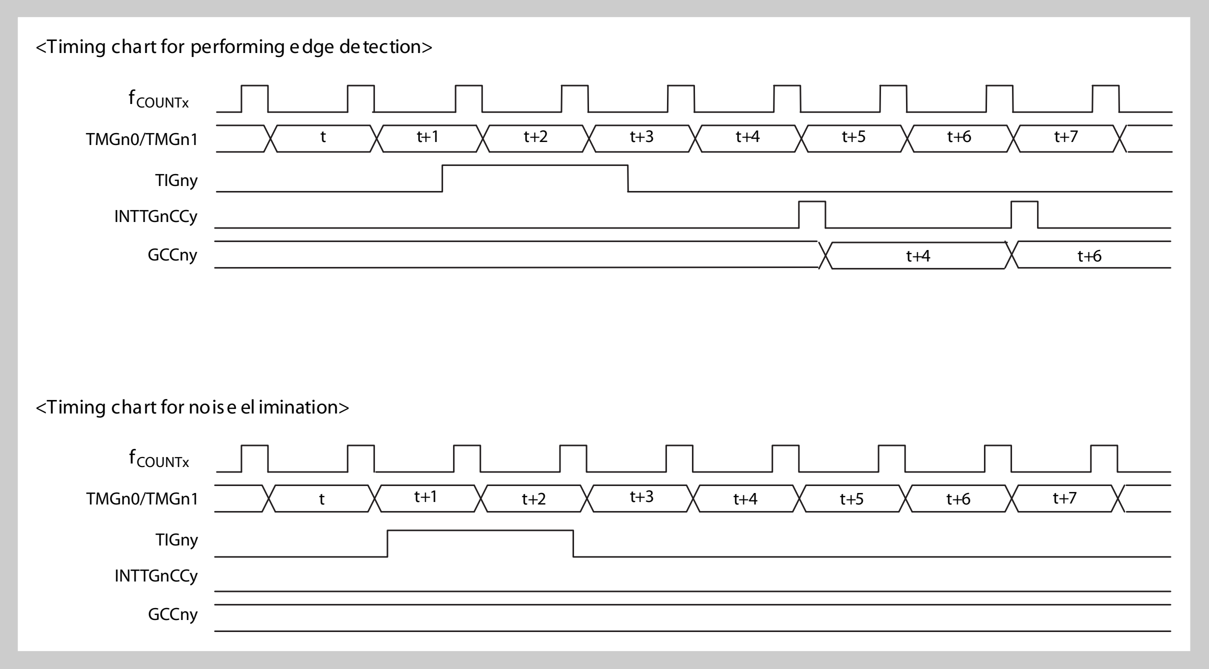


Figure 12-20 Timing of edge detection noise elimination

12.10 Precautions Timer Gn

(1) When POWERn bit of TMGMHn register is set

The rewriting of the CSEn2 to CSEn0 bits of TMGMHn register is prohibited.

These bits set the prescaler for the Timer Gn counter.

The rewriting of the CCSGny bits (y = 0 to 5) is prohibited.

This bits (OCTLGnL and OCTLGnH registers) set the capture mode or the compare mode to the GCCy register. For the GCCn0 register and the GCCn5 register these bits (TMGMLn register) set the “free run” or “match and clear” mode of the TMGn0 and TMGn1 counter.

The rewriting of the TMGCMnL and the TMGCMHn register is prohibited.

These registers configure the counter (TMGn0 or TMGn1) for the GCCnm register (m = 1 to 4) and define the edge detection for the TIGnm input pins (falling, rising, both).

Even when POWERn bit is set, TOGnm output is switched by switching the ALVGnm bit of OCTLGnL and OCTLGnH registers.

These bits configure the active level of the TOGnm pins (m = 1 to 4).

(2) When POWERn bit and TMGxE bit are set (x = 0, 1)

The rewriting of ALVGnm is prohibited (m = 1 to 4).

These bits configure the active level of the TOGnm pins (m = 1 to 4).

When in compare-mode the rewriting of the GCCn0 or GCCn5 register is prohibited.

In compare mode these registers set the value for the “match and clear” mode of the TMGn0 and TMGn1 counter.

(3) Functionality

When the POWERn bit is set to “0”, regardless of the SWFGnm bit (OCTLGnL and OCTLGnH registers), the TOGnm pins are tied to the inactive level.

The SWFGnm bit enables or disables the output of the TOGnm pins. This bit can be rewritten during timer operation.

The CLRGx bit (x = 0, 1) is a flag. If this bit is read, a "0" is read at all times.

This bit clears the corresponding counter (TMGn0 or TMGn1)

When GCCnm register (m = 1 to 4) are used in capture operation:

If two or more overflows of TMGn0 or TMGn1 occur between captures, a software-based measure needs to be taken to count overflow interrupts (INTTMGn0 or INTTMGn1).

If only one overflow is necessary, the CCFGny bits (y = 0 to 5) can be used for overflow detection.

Only the overflow of the TMGn0 or TMGn1counter clears the CCFGny bit (TMGSTn register). The software-based clearing via CLRGn0 or CLRGn1 bit (TMGMLn register) doesn't affect these bits.

The CCFGny bit is set if a TMGn0 (TMGn1) overflow occurs. This flag is only updated if the corresponding GCCny register was read, so first read the GCCny register and then read this flag if necessary.

(4) Timing

The delay of each timer output TOGnm (m = 1 to 4) varies according to the setting of the count clock with the CSEx2 to CSEx0 bits (x = 0, 1).

In capture operation 3 to 4 periods of the count-clock (f_{COUNT}) signal are required from the TIGny pin (y = 0 to 5) until a capture interrupt is output.

When TMGxE (x = 0, 1) is set earlier or simultaneously with POWERn bit, than the Timer Gn needs 7 peripheral clock periods (f_{SPCLK0}) to start counting.

When TMGxE (x = 0, 1) is set later than POWERn bit, than the Timer Gn needs 4 peripheral clock periods (f_{SPCLK0}) to start counting.

When a capture register (GCCny) is read, the capturing is disable during read operation. This is intended to prevent undefined data during reading. So, if a contention occurs between an external trigger signal and the read operation, capture operation may be cancelled, and old data may be read.

GCCnm register (m = 1 to 4) in Compare mode:

After setting the POWERn bit you have to wait for 10 peripheral clock periods (f_{SPCLK0}) to perform write access to the GCCnm register (m = 1 to 4).

To perform successive write access during operation, for rewriting the GCCnm register, you have to wait for minimum 7 peripheral clock periods (f_{SPCLK0}).

Chapter 13 Watch Timer (WT)

The Watch Timer (WT) generates interrupts at regular time intervals. These interrupts are generally used as ticks for updating the internal daytime and calendar.

The Watch Timer includes two identical counters. Throughout this chapter, the counters are identified as WTn, where n = 0 to 1.

13.1 Overview

The Watch Timer consists of two 16-bit down-counters, WT0 and WT1, and includes the Watch Calibration Timer WCT.

WT0 The load value that must be set for WT0 depends on the chosen clock frequency and the desired time interval between two interrupts.

For example, WT0 can be set up to generate an interrupt every second (INTWT0UV).

During normal operation, the clock of WT0 (WTCLK) is directly derived from the precision main oscillator. It bypasses the PLL and SSCG.

However, the WTCLK can also be derived from the sub or internal oscillator. This is useful if the main oscillator is switched off in order to save power.

WT1 WT1 is clocked by the interrupts generated by WT0. It can, for example, generate an interrupt every hour (or whatever wake-up time is required).

This interrupt (INTWT1UV) can be used to escape from Sub-WATCH mode and hence to revive the main oscillator if necessary.

WCT The sub or internal oscillators used in Sub_WATCH mode are not as stable as the main oscillator. The time between two WT0 interrupts may be slightly shorter or longer than desired.

Therefore a third timer - the Watch Calibration Timer (WCT) - can be used occasionally to measure the time between two interrupts INTWT0UV.

WCT requires the main oscillator clock for this measurement. Its clock, WCTCLK, always stops if the main oscillator stops, that means if STOP mode or Sub-WATCH mode are entered.

Based on the measurement result, a new load value for WT0 can be calculated. This is the solution to regain precise intervals between WT0 interrupts. After the adjustment of WT0, the system can return to Sub-WATCH mode where the main oscillator is stopped.

Features summary

Special features of the Watch Timer are:

- Periodic interrupts (clock ticks) generated by two down-counters
- Two reload registers, one for each counter
- Choice of oscillators to reduce power consumption in stand-by mode
- Can operate in all power save modes
- Clock correction in stand-by mode by means of the Watch Calibration Timer
- In debug mode, the counters WT0 and WT1 can be stopped at breakpoint

Special features of the Watch Calibration Timer are:

- 16-bit capture / compare register CR001
- Capture / trigger input for INTWT0UV with edge specification for INTWT0UV interval measurement
- Capture / match interrupt request signal INTTM01

13.1.1 Description

The following figure shows the structure of the Watch Timer and its connection to the Watch Calibration Timer.

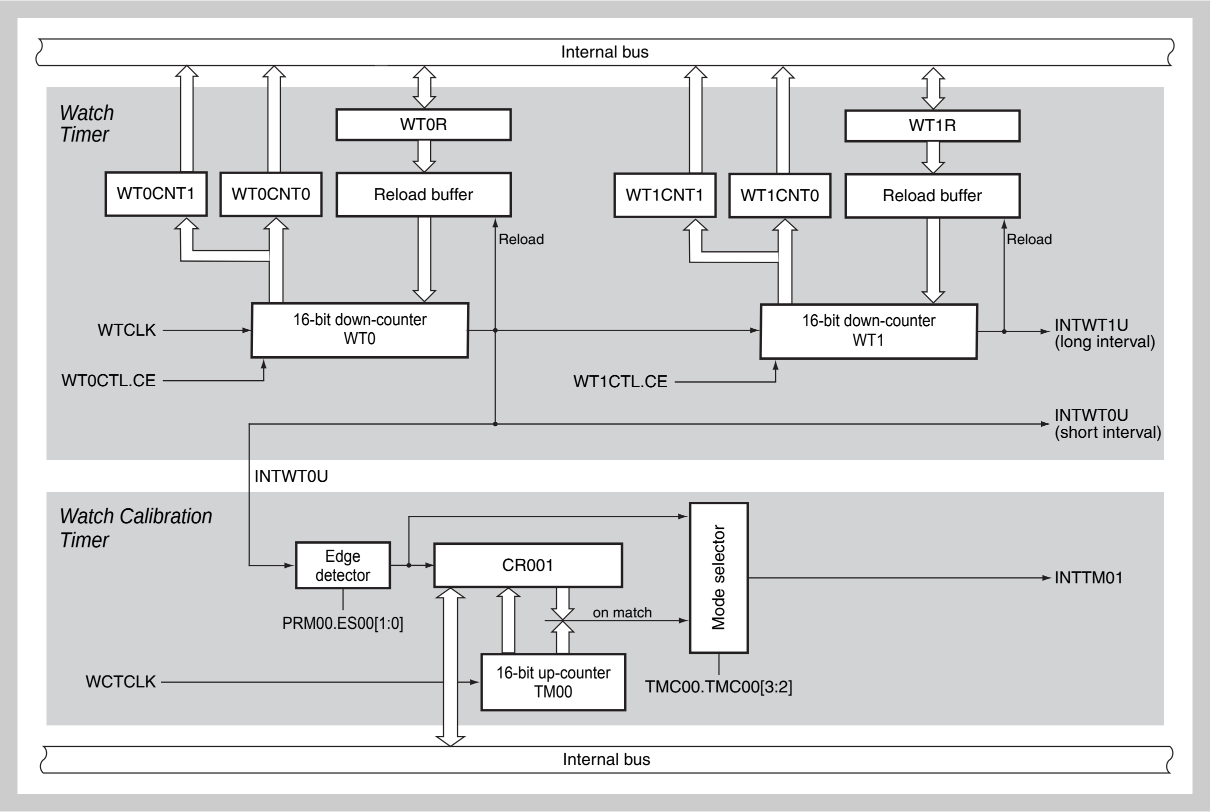


Figure 13-1 Watch Timer configuration

As shown in the figure, WT0 is clocked by WTCLK, a clock generated by the Clock Generator. When WT0 counts down to zero, it generates the INTWT0UV interrupt.

WT1 is clocked by the interrupts INTWT0UV. When WT1 reaches zero, it generates the interrupt INTWT1UV.

Two control registers WTnCTL are used to enable the counters. This is done by setting WTnCTL.WTCE to 1.

As soon as the counters are enabled, it is possible to write a start value to the reload registers WT0R and WT1R.

WCT is a capture/compare timer. In this application, it measures the time between two INTWT0UV interrupts. It is clocked by WCTCLK, another clock generated by the Clock Generator.

13.1.2 Principle of operation

In order to generate an interrupt every one or two seconds, WTCLK is usually set to a frequency around 30 KHz. Then, a load value around 2^{15} will yield a running time of about 1 s.

(1) Operation control of WT0

The source and frequency of WTCLK are specified in the Clock Generator register TCC.

The Clock Generator contains a programmable frequency divider that makes it possible to scale down the selected clock source.

Note WTCLK uses the same clock source and clock divider as the LCD Controller/ Driver clock LCDCLK. The frequency f_{WTCLK} can be the same as f_{LCDCLK} or f_{LCDCLK} / 2. For details refer to *"Clock Generator" on page 100*.

Typical settings and the resulting maximum time interval between two interrupts are listed in the table below.

Table 13-1 Typical Settings of WTCLK

Clock source	Clock divider setting	WTCLK Frequency	Max. period of INTWT0UV[a]
4 MHz main osc.	1 / 128	31.25 KHz	2.097 s
32 kHz sub osc.	1	32.768 KHz (typ.)	2.0 s
240 kHz internal osc.	1 / 8	30 KHz (typ.)	2.184533 s

[a] The maximum period corresponds to a counter load value of $2^{16} – 1$.

Note that you can double the maximum period by setting TCC.WTSEL1 to 1.

The clock input can be disabled (WT0CTL.WTCE = 0). This stops the Watch Timer. After reset, the timer is also stopped.

When WT0 is enabled and a non-zero reload value is specified, the counter decreases with every rising edge of WTCLK. When the counter reaches zero, the interrupt INTWT0UV is active high for one clock cycle. Upon undeflow, i.e. with the next clock, the timer reloads its start value and resumes down-counting. The load value can be freely chosen

(2) Operation of WT1

Once WT1 is enabled and a non-zero reload value is specified, its counter decreases with every interrupt INTWT0UV.

When WT1 reaches zero, it generates the interrupt INTWT1UV. Upon undeflow, i.e. with the next clock, the timer reloads its load value and restarts down-counting. The load value can be freely chosen.

Starting WT1 requires some attention. For further details refer to *"Watch Timer start-up" on page 386*.

(3) Operation of WCT

The third counter WCT is used for clock correction. This counter is connected to PCLK1 (8 MHz) or directly to the 4 MHz main oscillator. It is used to measure the time between two INTWT0UV requests.

For this measurement, WCT is configured as a capture timer.

Once it is enabled, the WCT counter is increased with every rising edge of its clock. When the value FFFF_H is reached, the counter sets a flag and restarts with 0000_H.

The interrupt INTWT0UV from counter WT0 triggers the capture operation. At every INTWT0UV, the count value is captured, and the interrupt INTTM01 is generated. From the counter difference between two consecutive capture events, the accuracy of the WTCLK can be measured, and WT0 or WT1, respectively, can be corrected.

The WCT can be programmed to restart counting after the capture operation.

Note The WCT detects the valid edge of INTWT0UV by sampling its input signal (the INTWT0UV interrupt line) with WCTCLK. The capture operation is only performed if the same level after a valid edge is detected two times in series.

As a consequence, the time interval measurement will only work correctly if $f_{WTCLK} < f_{WCTCLK} / 2$.

13.2 Watch Timer Registers

The Watch Timer counters WT0 and WT1 are controlled and operated by means of the following registers:

Table 13-2 WTn registers overview

Register name	Shortcut	Address
Watch timer synchronized read register	WTnCNT0	<base>
Watch timer non-synchronized read register	WTnCNT1	<base> + 2_H
Watch timer reload register	WTnR	<base> + 4_H
Watch timer control register	WTnCTL	<base> + 6_H

Table 13-3 WTn register base addresses

Timer	Base address
WT0	FFFF F560$_H$
WT1	FFFF F570$_H$

(1) WTnCTL - WTn timer control register

The 8-bit WTnCTL register controls the operation of the timer WTn.

Access This register can be read/written in 8-bit or 1-bit units.

Address <base> + 6_H

Initial Value 00_H. This register is cleared by any reset.

7	6	5	4	3	2	1	0
WTCE	0	0	0	0	0	0	0
R/W	R	R	R	R	R	R	R

Table 13-4 WTnCTL register contents

Bit position	Bit name	Function
7	WTCE	Watch timer counter enable: 0: Disable count operation (the timer stops immediately with the count value 0000_H and does not operate). 1: Enable count operation.

Note

1. If WTnCTL.WTCE is 1, the counter starts after the counter's load value has been written to the reload register WTnR. As long as WTnR is zero, no counting is performed, and no interrupts INTWTnUV are generated.
2. The first interval from counter start to the first underflow takes at least four clock cycles more than the following intervals. For details refer to *“Watch Timer start-up“ on page 386*.

(2) WTnCNT0 - WTn synchronized counter register

The WTnCNT0 register is the synchronized register that can be used to read the present value of the 16-bit counter.

“Synchronized” means that the read access via the internal bus is synchronized with the counter clock. The synchronization process causes a delay, but the resulting value is reliable.

Access This register is read-only, in 16-bit units.

Address <base> of WTn

Initial Value 0000_H. This register is cleared by any reset and if WTnCTL.WTCE = 0.

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">Updated counter value (synchronized)</td></tr>
<tr><td colspan="16">R</td></tr>
</table>

Note Due to the low frequencies of the counter clocks, the synchronization can take about up to two WTCLK. For a quick response, it is recommended to read the non-synchronized counter register WTnCNT1.

(3) WTnCNT1 - WTn non-synchronized counter read register

The WTnCNT1 register is the non-synchronized register that can be used to read the present value of the corresponding 16-bit counter.

“Non-synchronized” means that the read access via the internal bus is not synchronized with the counter clock. It returns the instantaneous value immediately, with the risk that this value is just being updated by the counter and therefore in doubt.

Access This register is read-only, in 16-bit units.

Address <base> + 2_H

Initial Value 0000_H. This register is cleared by any reset and if WTnCTL.WTCE = 0.

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">Instantaneous counter value (non-synchronized)</td></tr>
<tr><td colspan="16">R</td></tr>
</table>

Note The value read from this register can be incorrect, because the read access is not synchronized with the counter clock.

Therefore, this register shall be read multiple times within one period of the counter clock cycle.

If the difference between the first and the second value is not greater than one, you can consider the second value to be correct. If the difference between the two values is greater than one, you have to read the register a third time and compare the third value with the second. Again, the difference must not be greater than one.

If the read accesses do not happen within one period of the counter clock cycle, the difference between the last two values will be greater than one. In this case, you can only repeat the procedure or estimate the updated counter value.

Reading the counter value via WTnCNT1 instead of WTnCNT0 is only reasonable if the CPU clock is remarkably higher than WTCLK and the overhead of multiple reading WTnCNT1 is justifiable.

(4) WTnR - WTn reload register

The WTnR register is a dedicated register for setting the reload value of the corresponding counter.

Access This register can be read/written in 16-bit units.

Address <base> + 4_H

Initial Value 0000_H. This register is cleared by any reset and if WTnCTL.WTCE = 0.

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="16">Down-counter load value</td></tr>
<tr><td colspan="16">R/W</td></tr>
</table>

Note

1. WTnR can only be written if WTnCTL.WTCE = 1 (counter enabled).
2. The load value must be non-zero (0001_H …$FFFF_H$).
3. The contents of this register is automatically copied to the reload buffer. The counters load their values from the respective buffers at underflow. To ensure correct operation, this register shall not be written twice within three cycles of the counter clock. A second write attempt within that time span is ignored.
   This time span of three cycles of the counter clock is stalled and not cleared if WTnCTL.WTCE is cleared to 0. After restarting the counter by setting WTnCTL.WTCE back to 1, the time span will continue to elapse, but the counter will not be started automatically.
4. The value read from WTnR is the target start value. It is not necessarily identical with the current start value that is stored in the reload buffer. The buffer may not yet be updated.

13.3 Watch Timer Operation

This section describes the operation of the Watch Timer counters in detail.

13.3.1 Timing of steady operation

The contents of the reload registers WTnR can be changed at any time, provided the corresponding counter is enabled. The contents is then copied to the reload buffer. The counters WTn reload their start value from the buffer upon underflow.

This is illustrated in the following figure.

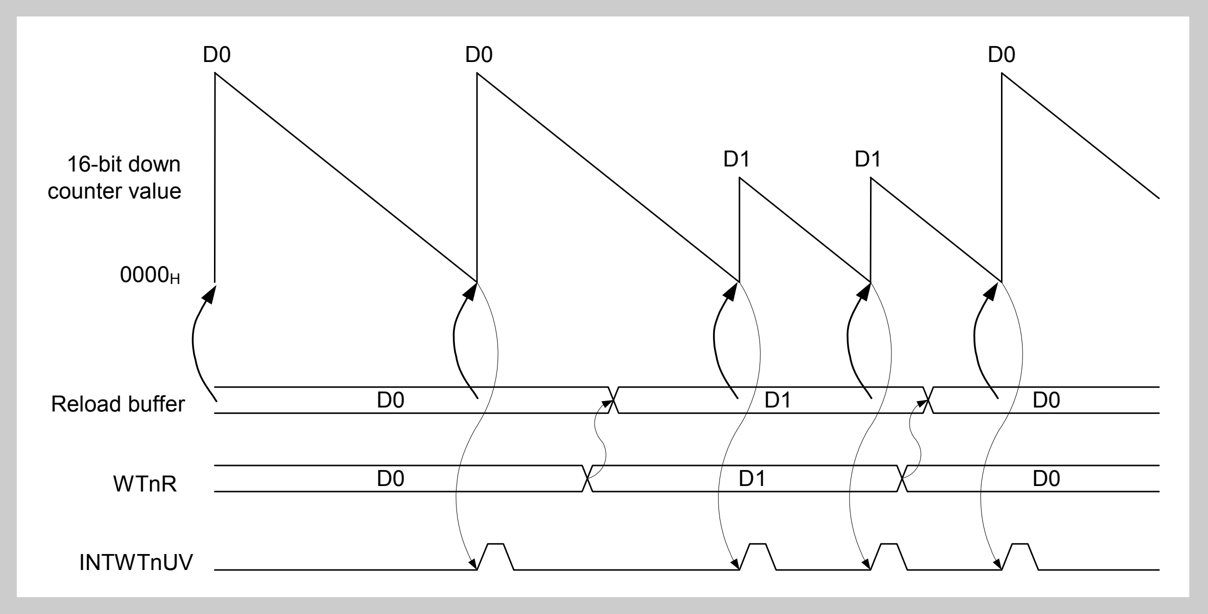


Figure 13-2 Reload timing and interrupt generation

D0 and D1 are two different reload values.

Note also that there is a delay between writing to WTnR and making the data available in the reload buffer, depending on the previous reload value and the chosen count clock.

13.3.2 Watch Timer start-up

The first interval after starting WT0 and WT1 until their first underflow takes at least four additional input clock cycles. At this point in time, the values of the counter registers WTnCNT are not correct.

After the first automatic reload of the WTnR value, the counter registers WTnCNT hold the correct number of clock cycles since the last underflow.

In the following, the start-up procedure of WT1 is described, because of its higher relevance from an application point of view. However, all statements refer also to WT0.

Start-up timing Starting WT1 correctly requires some attention in order to avoid wrong calculation of the watch time.

If WT1 is used as an extended Watch Timer counter, two steps in the following order are required:

- The counter has to be enabled by setting WT1CTL.WTCE = 1.
- The counter's reload register WT1R has to be set to a non-zero value.

Both actions consume a different amount of input clock cycles to become effective, as shown in the following diagram.

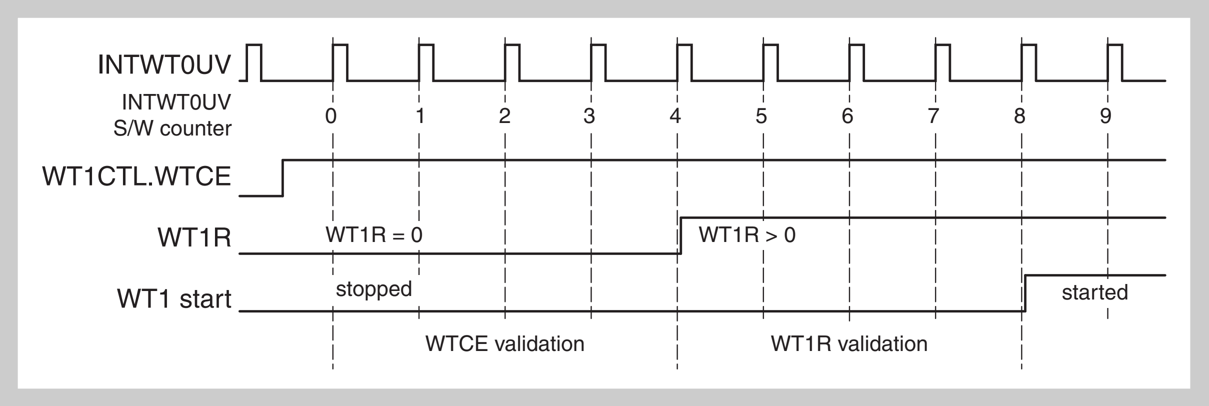


Figure 13-3 WT1 start-up timing

To start the counter in a deterministic way, the above actions have to be synchronized to the WT1 input clock, which is INTWT0UV. For that purpose it is recommended to maintain a software counter that is increased inside the INTWT0UV interrupt service routine. By this means, it is ensured that the actions are performed at the correct point in time.

Setting WT1CTL.WTCE to 1 enables WT1. The write access can happen at any time. Due to internal clock synchronization, it takes at least five complete input clock cycles, that means four INTWT0UV intervals (WTCE validation time 0 –> 1 –> 2 –> 3 –> 4) to become effective. After that, WT1 is prepared to acknowledge the reload value.

S/W counter state “4” indicates that the reload value can be written now (WT1R > 0). This time, at least three complete input clock cycles (WTR1 validation time 5 –> 6 –> 7 –>8) are required to take over the reload value from WT1R to the reload buffer and to start counting. At S/W counter state “8” the counter WT1CNT is preloaded with the WT1R contents.

As a consequence, register WT1CNT does not show the correct number of INTWT0UV events after WT1R > 0, but a value of four less:

- 1 INTWT0UV cycle 4 –> 5 taken for the cycle WT1R is written
- 3 INTWT0UV cycles 5 –> 6 –> 7 –> 8 for WT1R validation time

The above calculation assumes that WT1R is written within one INTWT0UV cycle, which is highly probable, considering INTWT0UV to be the "one second tick".

However, it may happen that the write to WT1R is delayed because of other circumstances (nested interrupts, etc.) and may happen after S/W counter state 3.

Thus, WT1 would start later, since the 3 clock WTR1 validation time is maintained.

In order to recognize that situation, read the WT1CNT1 register and compare its contents with the value written to WT1R. If both are equal, WTR1 has been written before S/W counter state 3, add four when reading WT1CNT. If they are not equal check again at next INTWT0UV and add the proper number of correction cycles.

13.4 Watch Calibration Timer Registers

The Watch Calibration Timer is controlled by means of the following registers:

Table 13-5 WCT registers overview

Register name	Shortcut	Address
WCT capture / compare register	CR001	<base> + 4_H
WCT mode control register	TMC00	<base> + 6_H
WCT prescaler mode register	PRM00	<base> + 7_H
WCT capture / compare control register	CRC00	<base> + 8_H

Table 13-6 WCT register base address

Timer	Base address
WCT	FFFF $F5E0_H$

(1) TMC00 - WCT mode control register

The 8-bit TMC00 register controls the operation of the WCT.

Access This register can be read/written in 8-bit or 1-bit units.

Address <base> + 6_H

Initial Value 00_H. This register is cleared by any reset.

7	6	5	4	3	2	1	0
0	0	0	0	TMC003	TMC002	0	OVF00
R	R	R	R	R/W	R/W	R	R/W

Table 13-7 TMC00 register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>3 to 2</td><td>TMC00[3:2]</td><td>Operation mode selection:
<table>
<tr><th>TMC003</th><th>TMC002</th><th>Operating mode</th></tr>
<tr><td>0</td><td>0</td><td>Stop mode</td></tr>
<tr><td>0</td><td>1</td><td>Free-running mode. Generates interrupt on match between TM00 and CR001.</td></tr>
<tr><td>1</td><td>x</td><td>Setting prohibited.</td></tr>
</table></td></tr>
<tr><td>0</td><td>OVF00</td><td>Counter overflow indicator:
0: No overflow
1: Overflow occurred</td></tr>
</table>

Note

1. If an attempt is made to change the setting of TMC00[3:2] while the timer is running, these bits are cleared and the timer is stopped. If the timer is stopped, you can change the operation mode.
2. The OVF00 bit is set if the counter reaches $FFFF_H$ and once more if the counter continues with 0000_H. Clearing OVF00 within that time has no effect.

(2) PRM00 - WCT prescaler mode register

The 8-bit PRM00 register is used to select the “valid edge” of INTWT0UV for interval measurements.

Access This register can be read/written in 8-bit or 1-bit units.

Address <base> + 7_H

Initial Value 00_H. This register is cleared by any reset.

7	6	5	4	3	2	1	0
0	0	ES001	ES000	0	0	0	0
R	R	R/W	R/W	R	R	R	R

Table 13-8 PRM00 register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>5 to 4</td><td>ES00[1:0]</td><td>Edge selection:
<table>
<tr><th>ES001</th><th>ES000</th><th>Valid edge</th></tr>
<tr><td>0</td><td>0</td><td>Falling edge</td></tr>
<tr><td>0</td><td>1</td><td>Rising edge</td></tr>
<tr><td>1</td><td>0</td><td>Setting prohibited</td></tr>
<tr><td>1</td><td>1</td><td>Both rising and falling edges</td></tr>
</table>
</td></tr>
</table>

All other bits are initialized as zero and must not be changed.

Note

1. If both edges of INTWT0UV are specified as valid, INTWT0UV interval measurement is not possible.
2. Stop the timer before changing ES00[1:0].

(3) CRC00 - WCT capture / compare control register

The 8-bit CRC00 register controls the operation of the capture/compare register CR001.

Access This register can be read/written in 8-bit or 1-bit units.

Address <base> + 8_H

Initial Value 00_H. This register is cleared by any reset.

7	6	5	4	3	2	1	0
0	0	0	0	0	CRC002	0	0
R	R	R	R	R	R/W	R/W	R/W

Table 13-9 CRC00 register contents

Bit position	Bit name	Function
2	CRC002	Selects the operation mode of CR001: 0: Operates as a compare register. 1: Operates as capture register.

Note

1. Stop the timer before changing the contents of this register.
2. If both the rising edge and falling edge are specified as valid for the INTWT0UV signal, the interval measurement does not work.
3. Be sure to set bits 7 to 3 and 1 to 0 to 0.

(4) CR001 - WCT capture / compare register 1

The 16-bit CR001 register can be used as a capture register or as a compare register. Whether it is used as a capture register or compare register is specified in bit CRC00.CRC002.

- Compare mode:
  In compare mode, the value written to CR001 is continually compared with the count value of TM00. If the two values match, the interrupt request INTTM01 is generated.
- Capture mode:
  In capture mode, the count value of TM00 is copied to CR001 upon a valid edge of INTWT0UV. Then, the interrupt INTTM01 is generated.

  The valid edge of INTWT0UV can be selected as a capture trigger. The valid edge is specified in PRM00.ES00[1:0].
  In capture mode, a read access to register CR001 is not synchronized with the counter operation. Read access and register update can occur simultaneously. If that happens, CR001 holds the correct value, but the value read is undefined.

Access In compare mode, this register can be read/written in 16-bit units. In capture mode, it cannot be written.

Address <base> + 4_H

Initial Value 0000_H. This register is cleared by any reset.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Compare or captured value (captured from TM00)															
R(W)															

Note Stop the timer before changing the contents of this register.

13.5 Watch Calibration Timer Operation

The Watch Calibration Timer WCT is used to measure the time between two successive occurrences of the Watch Timer WT0 underflow interrupt INTWT0UV.

The WCT is supplied with the stable clock WCTCLK:

- WCTCLK = 4 MHz main oscillator, if PSM.CMODE = 1
- WCTCLK = 8 MHz PCLK1, if PSM.CMODE = 0

For further details refer to *“Clock Generator“ on page 100*.

The measured INTWT0UV interval time gives an indication about the accuracy of the sub or internal oscillator. A correction value can be calculated to calibrate WT0 and WT1 by changing their reload values.

The interval measurement can be performed in the WCT free-running operating mode.

If a timer overflow can occur during the interval measurement, take care for regarding also the overflow flag TMC00.OVF00 for calculating the interval correctly.

The timer operating as a free-running counter performs following actions upon detection of a the valid edge of INTWT0UV:

- it copies the present counter value of register TM00 to CR001,
- it generates the interrupt request INTTM01.

The valid edge (rising edge, falling edge) is specified in register PRM00. If both edges are specified, CR001 cannot perform a capture operation.

Setup example	TMC00 = 0000 0100_B:	Free running mode
	CRC00 = 0000 0100_B:	CR001 as capture register with INTWT0UV as capture signal
	PRM00.ES00[1:0] = 01_B:	Rising edge

The following figure is not to scale but illustrates the operation.

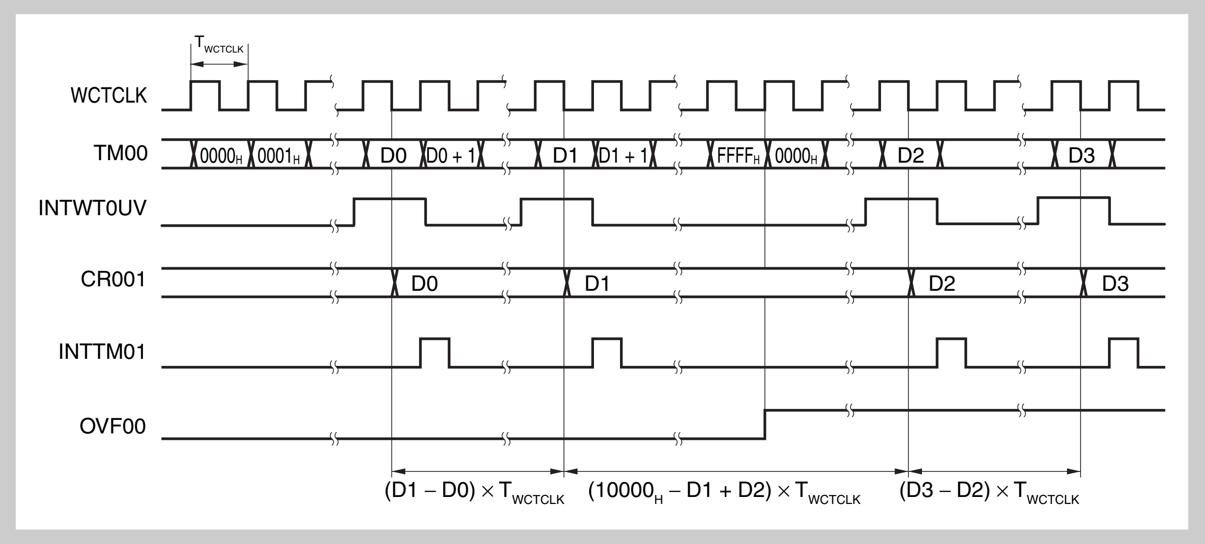


Figure 13-4 Timing in free-running mode

As shown in the figure, the interrupt INTTM01 can be used as a trigger for reading the register CR001.

The interval duration must be calculated from the difference between the present and the previous value of CR001.

Note If TM00 overflows between two occurrences of INTWT0UV, that means between two capture triggers, the overflow flag TMC00.OVF00 is set. Therefore, check also TMC00.OVF00 when reading the second capture value in order to calculate the interval correctly, because an overflow may happen during the measurement.

Consider the chosen periods for INTWT0UV and of WCTCLK.

Chapter 14 Watchdog Timer (WDT)

The Watchdog Timer is used to escape from a system deadlock or program runaway. If it is not restarted within a certain time, the Watchdog Timer flows over and interrupts or even resets the microcontroller.

14.1 Overview

The Watchdog Timer contains an up-counter that is driven by the Watchdog Timer clock WDTCLK. This clock can be derived from the main oscillator, the internal oscillator, or the sub oscillator. It's frequency can be identical with the frequency of the source clock or a fraction thereof.

Features summary

The Watchdog Timer

- can generate the non-maskable interrupt NMIWDT
- can generate a hardware reset by means of the internal signal RESWDT
- has a programmable running time (set in terms of 2^n multiples of WDTCLK periods)
- is specially protected against inadvertent setup changes

14.1.1 Description

The following figure shows a simplified block diagram.

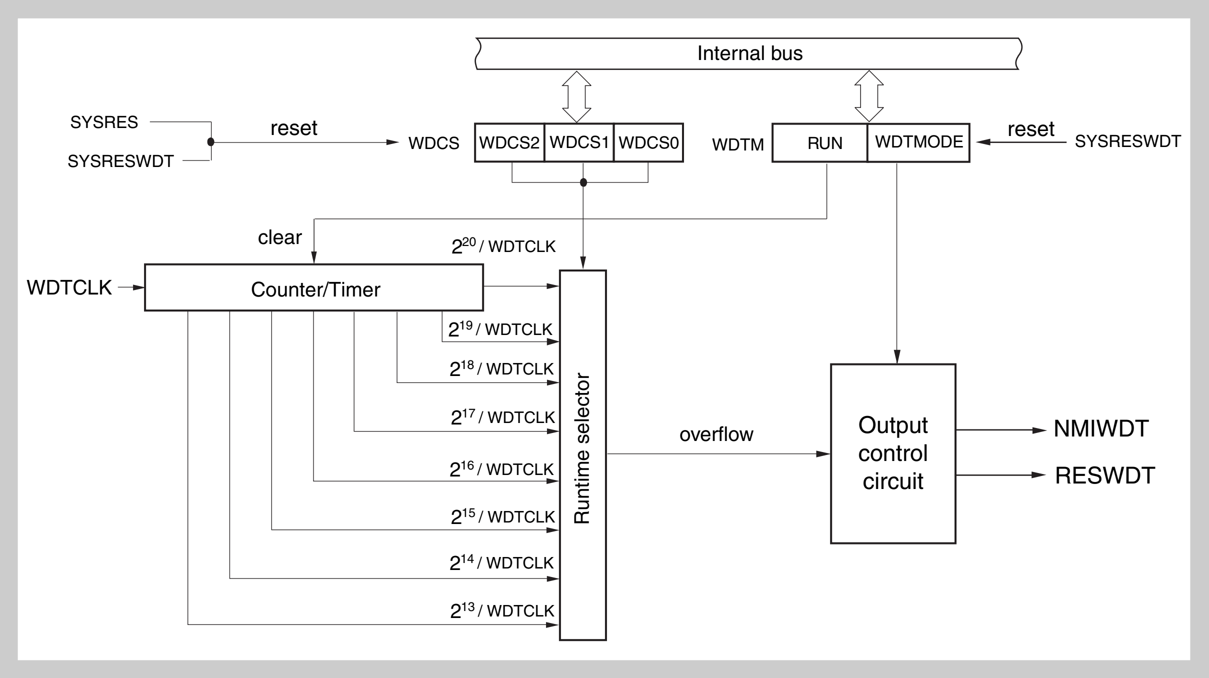


Figure 14-1 Block diagram of the Watchdog Timer

As shown in the figure, the WDCS register controls the running time and the WDTM register the operating mode.

The running time can be chosen between 2^{13} and 2^{20} times the period of the Watchdog Timer clock WDTCLK.

The figure shows also, that the run and mode settings of the WDTM register are only cleared by SYSRESWDT.

14.1.2 Principle of operation

Before the Watchdog Timer is started, its running time and mode have to be configured.

The Watchdog Timer has two operating modes:

- Mode 0 (generate non-maskable interrupt NMIWDT)
- Mode 1 (generate reset request RESWDT)

The mode is defined by the bit WDTM.WDTMODE. The mode can only be changed after SYSRESWDT, that means, after external $\overline{\text{RESET}}$ or Power-On Clear.

(1) Watchdog Timer mode 0 (generate non-maskable interrupt NMIWDT)

If WDTM.WDTMODE is 0, the Watchdog Timer is in interrupt-request mode. This is the default after initialization.

Setting bit WDTM.RUN to 1 starts the counter. Without intervention, the timer will now run until the specified time has elapsed and then generate the non-maskable interrupt NMIWDT. After that, the counter is reset to zero and starts counting again.

(2) Watchdog Timer mode 1 (generate reset request RESWDT)

If WDTM.WDTMODE is 1, the Watchdog Timer is in reset-request mode.

Setting bit WDTM.RUN to 1 starts the counter. Without intervention, the timer will now run until the specified time has elapsed and then generate the internal RESWDT signal. After that, the counter operation is stopped until the system reset SYSRES or SYSRESWDT occurs.

(3) Watchdog Timer running

Once it is running, the Watchdog Timer cannot be stopped by software. It can only be stopped by the reset signal SYSRESWDT. This signal is generated by the Reset module at power-on and external $\overline{\text{RESET}}$.

The way to prevent the timer from flowing over is writing to the register WDTM before the specified time has elapsed. The write access resets the counter to zero.

14.1.3 Watchdog Timer clock

The Watchdog Timer clock WDTCLK is generated by the Clock Generator. It can be derived from the main, internal or sub oscillator.

The generation of WDTCLK is controlled by the WCC register of the Clock Generator.

In this register, it is possible to choose the main, sub, or internal oscillator as the clock source (WCC.SOSCW, WCC.WDTSEL0).

You can also choose a suitable frequency divider between 1 and 128 (WCC.WPS[2:0]).

WDTCLK is subject to a stand-by mode control. WDTCLK can optionally be stopped in IDLE, WATCH, Sub-WATCH and STOP mode (WCC.WDTSEL1).

Please refer to *“Clock Generator“ on page 100* for further details.

Note Once the timer has been started, do not switch off the selected clock source of WDTCLK.

When the microcontroller is in HALT mode, the Watchdog Timer remains active.

The activity in the other power save modes can be specified by the WCC.WDTSEL1 control bit. By default (WCC.WDTSEL1 = 0), WDTCLK stops during IDLE, WATCH, Sub-WATCH and STOP mode.
With WCC.WDTSEL1 = 1 WDTCLK operates as long as the selected clock source operates.

When the WDTCLK resumes operation, the Watchdog Timer is not reset but continues counting. To prevent a quick and unexpected overflow, it is recommended to write to WDTM and thus clear the Watchdog Timer counter before entering one of these power save modes.

14.1.4 Reset behavior

The reset of the Watchdog Timer is controlled by the two reset inputs SYSRES and SYSRESWDT. The respective signals are generated by the Reset module.

Every reset sets the WDCS register to the longest possible running time.

SYSRESWDT The watchdog reset SYSRESWDT is used to initialize the Watchdog Timer. This signal is generated at power-on and after external $\overline{\text{RESET}}$.

After SYSRESWDT, all registers are set to their reset values, and the timer is stopped. You have to write the required settings to the WDCS register and may start the counter. Once the counter has been started, it cannot be reprogrammed or stopped unless the next reset (SYSRES or SYSRESWDT) occurs.

SYSRES SYSRES is generated by all reset sources.

SYSRES does not reset the register WDTM. That means, the timer status (running or stopped) and mode (generate interrupt or reset request) remain unchanged.

If the Watchdog Timer was running before SYSRES was released, the counter is automatically cleared and restarts with the new timing.

Note

1. Every reset clears also the WCC register. That means, the WDTCLK has the frequency of the 240 kHz internal oscillator. In combination with the largest time factor (2^{20}), this yields a running time of 4.37 s.
2. After any reset, the write protection for WDCS is disabled. WDCS can be written once to specify a shorter time interval. After that, the WDCS register is write-protected.

14.2 Watchdog Timer Registers

The Watchdog Timer is controlled by means of the following registers:

Table 14-1 Watchdog Timer registers overview

Register name	Shortcut	Address
Watchdog Timer clock selection register	WDCS	<base>
Watchdog Timer command protection register	WCMD	<base> + 2_H
Watchdog Timer mode register	WDTM	<base> + 4_H
Watchdog Timer command status register	WPHS	<base> + 6_H

The registers WDCS and WDTM are protected against accidental changes. A special write procedure, employing the WCMD register, ensures that these registers are not easily rewritten in case of a program hang-up.

Their contents can only be changed after a reset.

In addition, the registers are write-protected when the timer is running. Their protection status is indicated in the WDTM register.

Table 14-2 WDT register base address

Timer	Base address
WDT	FFFF $F590_H$

Note Only byte access is supported for the registers WDCS, WCMD and WDTM. The registers are allocated at even addresses. Thus, they cannot be written by a consecutive byte write sequence or a consecutive half word or word write sequence.

(1) WDCS - WDT clock selection register

The 8-bit WDCS register is used to specify the running time of the Watchdog Timer.

Access This register can be read/written in 8-bit or 1-bit units.

Writing to this register is protected by a special sequence of instructions. Please refer to *“WCMD - WDT command protection register“ on page 402* for details.

Address <base>

Initial Value 07_H. This register is initialized by SYSRESWDT and SYSRES.

7	6	5	4	3	2	1	0
					WDCS2	WDCS1	WDCS0
R	R	R	R	R	R/W	R/W	R/W

Table 14-3 WDCS register contents

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>2 to 0</td><td>WDCS[2:0]</td><td>Specifies the running time of the Watchdog Timer
<table>
<tr><th>WDCS2</th><th>WDCS1</th><th>WDCS0</th><th>Running time calculation</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>$2^{13} / f_{WDTCLK}$</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>$2^{14} / f_{WDTCLK}$</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>$2^{15} / f_{WDTCLK}$</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>$2^{16} / f_{WDTCLK}$</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>$2^{17} / f_{WDTCLK}$</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>$2^{18} / f_{WDTCLK}$</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>$2^{19} / f_{WDTCLK}$</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>$2^{20} / f_{WDTCLK}$</td></tr>
</table>
</td></tr>
</table>

Note The WDCS register must be considered in conjunction with the WCC register of the Clock Generator. The source and frequency of WDTCLK are defined in the WCC register.

The running time depends on the frequency of the chosen clock. The following table shows two examples for 4 MHz and 32 KHz.

Table 14-4 Running time examples

WDCS2	WDCS1	WDCS0	Calculation	Time until overflow	
				f_{WDTCLK} = 4 MHz (main oscillator)	f_{WDTCLK} = 32 KHz (sub oscillator)
0	0	0	$2^{13} / f_{WDTCLK}$	2 ms	256 ms
0	0	1	$2^{14} / f_{WDTCLK}$	4.1 ms	512 ms
0	1	0	$2^{15} / f_{WDTCLK}$	8.2 ms	1.02 s
0	1	1	$2^{16} / f_{WDTCLK}$	16.4 ms	2.05 s
1	0	0	$2^{17} / f_{WDTCLK}$	32.8 ms	4.10 s
1	0	1	$2^{18} / f_{WDTCLK}$	65.5 ms	8.20 s
1	1	0	$2^{19} / f_{WDTCLK}$	131 ms	16.38 s
1	1	1	$2^{20} / f_{WDTCLK}$	262 ms	32.77 s

These are just two examples for WDTCLK. The actual clock signal depends on the clock divider settings and the external oscillator resonators.

Note Every reset sets the WDCS register to 07_H, which means the longest time interval.

After SYSRESWDT, the timer is always stopped and initialized. You can write a smaller value to the register.

After SYSRES, the WDTM register is not cleared. If the Watchdog Timer was running before SYSRES occurred, it remains active. To specify a shorter interval:

1. Write one byte to the WCMD register (the value is ignored)
2. Immediately after that, write one byte with the desired value of WDCS[2:0] to the WDCS register

The write operation resets the watchdog counter to zero, and it continues with the new timing.

Note When the timer is active, WDCS can only be written once after reset. Then, the register is locked until the next reset occurs (WDTM.LOCK_CS = 1).

(2) WDTM - WDT mode register

This register sets the operating mode of the Watchdog Timer and enables or disables counting.

When the Watchdog Timer is running and shall not overflow, it is necessary to write to WDTM before the specified running time has elapsed.

Access This register can be read/written in 8-bit units. Once the Watchdog Timer is started (WDTM.RUN = 1), the contents of this register cannot be changed.

Writing to this register is protected by a special sequence of instructions. Please refer to *"WCMD - WDT command protection register" on page 402* for details.

Address <base> + 4_H

Initial Value 00_H. This register is cleared by SYSRESWDT. This stops the timer and unlocks the registers. The register remains unchanged after SYSRES.

7	6	5	4	3	2	1	0
RUN	LOCK_TM	LOCK_CS	0	WDTMODE	0	0	0
R/W	R	R	R	R/W	R	R	R

Table 14-5 WDTM register contents

Bit Position	Bit Name	Function
7	RUN	Watchdog Timer running: 0: Timer stopped 1: Timer running (with the time interval specified in register WDCS)
6	LOCK_TM	WDTM register protection status: 0: Register unlocked 1: Register locked (write-protected)
5	LOCK_CS	WDCS register protection status: 0: Register unlocked 1: Register locked (write-protected)
3	WDTMODE	Watchdog Timer operation mode: 0: Mode 0: Generates the non-maskable interrupt NMIWDT upon overflow 1: Mode 1: Generates RESWDT upon overflow

Note After SYSRESWDT, the timer is always stopped and initialized. You can change the register contents by writing.

When the timer is running, you can also write to this register, but the write operation does not change the register contents (WDTM.LOCK_TM = 1). When the timer is running, the write access resets the counter.

To write to the WDTM register:

1. Write one byte to the WCMD register (the value is ignored).
2. Immediately after that, write one byte to the WDTM register (the value is ignored).

With this procedure, restarting the counter is always possible, regardless of the register's write protection status.

(3) WCMD - WDT command protection register

The 8-bit WCMD register is write-only. It is used to protect the WDTM and WDCS registers from unintended writing.

Access This register can be written in 8-bit units.

Address <base> + 2_H

Initial Value Undefined

7	6	5	4	3	2	1	0
x	x	x	x	x	x	x	x
W	W	W	W	W	W	W	W

Any data written to this register is ignored. Only the write action is monitored.

After writing to the WCMD register, you are permitted to write once to one of the protected registers. This must be done immediately after writing to the WCMD register. If the second write action does not follow immediately, the protected registers are write-locked again. See also *“Write Protected Registers“ on page 96*.

With this method, the protected registers can only be rewritten in a specific sequence. Illegal write access to a protected register is inhibited.

The following registers are protected:

- WDCS: Watchdog clock selection register
- WDTM: Watchdog mode control register

An invalid write attempt to one of the above registers sets the error flag WPHS.WPRERR. WPHS.WPRERR is also set, if a write access to WCMD is not followed by an access to one of the protected registers.

Data read from the WCMD register is undefined.

Caution In case a high level programming language is used, make sure that the compiler translates the two write instructions to WCMD and the protected register into two consecutive assembler “store” instructions.

(4) WPHS - WDT command status register

The WPHS register monitors the success of a write instruction to the WDTM and WDCS registers.

If the write operation to WDTM or WDCS failed because WCMD was not written immediately before writing to WDTM or WDCS, the WPRERR flag is set.

Access This register can be read/written in 8-bit or 1-bit units. After a write access, the register is cleared.

Address <base> + 6_H

Initial Value 00_H. This register is cleared by SYSRESWDT and any write access.

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	WPRERR
R	R	R	R	R	R	R	R

Table 14-6 WPHS register contents

Bit Position	Bit Name	Function
0	WPRERR	Error flag: 0: No WDTM or WDCS register writing error has occurred 1: A WDTM or WDCS register writing error has occurred

Chapter 15 Asynchronous Serial Interface (UARTA)

The V850E/Dx3 - DG3 microcontrollers have following instances of the universal Asynchronous Serial Interface UARTA:

UARTA	All devices
Instances	2
Names	UARTA0 to UARTA1

Throughout this chapter, the individual instances of UARTA are identified by “n”, for example, UARTAn, or UAnCTL0 for the UARTAn control register 0.

15.1 Features

- Transfer rate: 300 bps to 1000 kbps (using dedicated baud rate generator)
- Full-duplex communication:
  - Internal UARTA receive data register n (UAnRX)
  - Internal UARTA transmit data register n (UAnTX)
- 2-pin configuration:
  - TXDAn: Transmit data output pin
  - RXDAn: Receive data input pin
- Reception error output function
  - Parity error
  - Framing error
  - Overrun error
- Interrupt sources: 3
  - Reception complete interrupt (INTUAnR):

    This interrupt occurs upon transfer of receive data from the shift register to receive buffer register n after serial transfer completion, in the reception enabled status.
  - Transmission enable interrupt (INTUAnT):

    This interrupt occurs upon transfer of transmit data from the transmit buffer register to the shift register in the transmission enabled status.
  - Receive error interrupt (INTUAnRE):

    This interrupt occurs upon transfer of erroneous receive data.
- Character length: 7, 8 bits
- Parity function: Odd, even, 0, none
- Transmission stop bit: 1, 2 bits
- On-chip dedicated baud rate generator
- MSB-/LSB-first transfer selectable
- Transmit/receive data inverted input/output possible

- 13 to 20 bits selectable for the SBF (Sync Break Field) in the LIN (Local Interconnect Network) communication format
  - Recognition of 11 bits or more possible for SBF reception in LIN communication format
  - SBF reception flag provided

15.2 Configuration

The block diagram of the UARTAn is shown below.

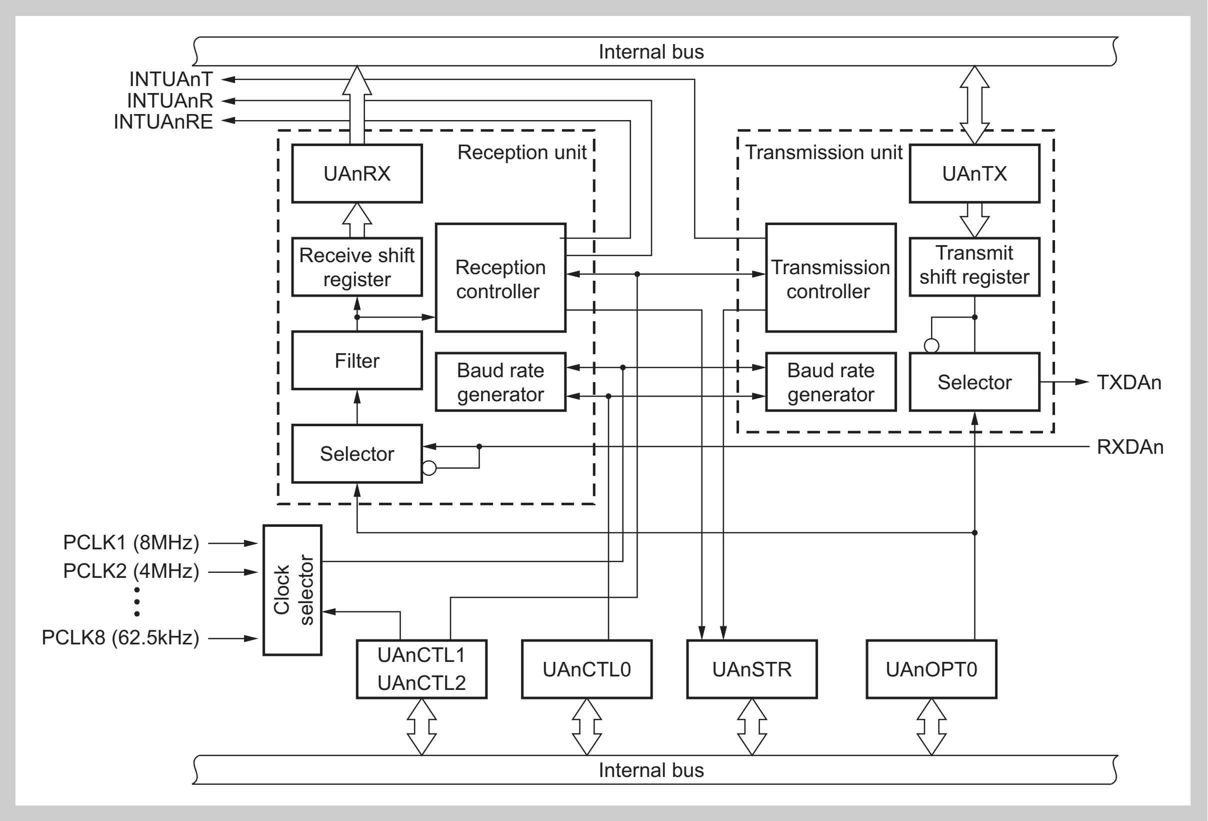


Figure 15-1 Block diagram of Asynchronous Serial Interface UARTAn

Note For the configuration of the baud rate generator, see *Figure 15-11 on page 428*.

UARTAn consists of the following hardware units.

(1) UARTAn control register 0 (UAnCTL0)

The UAnCTL0 register is an 8-bit register used to specify the UARTAn operation.

(2) UARTAn control register 1 (UAnCTL1)

The UAnCTL1 register is an 8-bit register used to select the input clock for the UARTAn.

(3) UARTAn control register 2 (UAnCTL2)

The UAnCTL2 register is an 8-bit register used to control the baud rate for the UARTAn.

(4) UARTAn option control register 0 (UAnOPT0)

The UAnOPT0 register is an 8-bit register used to control serial transfer for the UARTAn.

(5) UARTAn status register (UAnSTR)

The UAnSTRn register consists of flags indicating the error contents when a reception error occurs. Each one of the reception error flags is set (to 1) upon occurrence of a reception error and is reset (to 0) by reading the UAnSTR register.

(6) UARTAn receive shift register

This is a shift register used to convert the serial data input to the RXDAn pin into parallel data. Upon reception of 1 byte of data and detection of the stop bit, the receive data is transferred to the UAnRX register.

This register cannot be manipulated directly.

(7) UARTAn receive data register (UAnRX)

The UAnRX register is an 8-bit register that holds receive data. When 7 characters are received, 0 is stored in the highest bit (when data is received LSB first).

In the reception enabled status, receive data is transferred from the UARTAn receive shift register to the UAnRX register in synchronization with the completion of shift-in processing of 1 frame.

Transfer to the UAnRX register also causes the reception complete interrupt request signal (INTUAnR) to be output.

(8) UARTAn transmit shift register

The transmit shift register is a shift register used to convert the parallel data transferred from the UAnTX register into serial data.

When 1 byte of data is transferred from the UAnTX register, the shift register data is output from the TXDAn pin.

This register cannot be manipulated directly.

(9) UARTAn transmit data register (UAnTX)

The UAnTX register is an 8-bit transmit data buffer. Transmission starts when transmit data is written to the UAnTX register. When data can be written to the UAnTX register (when data of one frame is transferred from the UAnTX register to the UARTAn transmit shift register), the transmission enable interrupt request signal (INTUAnT) is generated.

15.3 UARTA Registers

The asynchronous serial interfaces UARTAn are controlled and operated by means of the following registers:

Table 15-1 UARTAn registers overview

Register name	Shortcut	Address
UARTAn control register 0	UAnCTL0	<base>
UARTAn control register 1	UAnCTL1	<base> + 1_H
UARTAn control register 2	UAnCTL2	<base> + 2_H
UARTAn option control register 0	UAnOPT0	<base> + 3_H
UARTAn status register	UAnSTR	<base> + 4_H
UARTAn receive data register	UAnRX	<base> + 6_H
UARTAn transmit data register	UAnTX	<base> + 7_H

Table 15-2 UARTAn register base address

Timer	Base address
UARTA0	FFFF $FA00_H$
UARTA1	FFFF $FA10_H$

(1) UAnCTL0 - UARTAn control register 0

The UAnCTL0 register is an 8-bit register that controls the UARTAn serial transfer operation.

Access This register can be read/written in 8-bit or 1-bit units.

Address <base>

Initial Value 10_H. This register is cleared by any reset.

7	6	5	4	3	2	1	0
UAnPWR	UAnTXE	UAnRXE	UAnDIR	UAnPS1	UAnPS0	UAnCL	UAnSL
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 15-3 UAnCTL0 register contents (1/2)

Bit position	Bit name	Function
7	UAnPWR	UARTAn operation disable/enable: 0: Disable UARTAn operation and reset the UARTAn asynchronously 1: Enable UARTAn operation The UARTAn operation is controlled by the UAnPWR bit. The TXDAn pin output is fixed to high level by clearing the UAnPWR bit to 0 (fixed to low level if UAnOPT0.UAnTDL bit = 1).
6	UAnTXE	Transmit operation disable/enable: 0: Disable transmit operation 1: Enable transmit operation • To start transmission, set the UAnPWR bit to 1 and then set the UAnTXE bit to 1. • To stop transmission, clear the UAnTXE bit to 0 and then the UAnPWR bit to 0. • To initialize the transmission unit, clear UAnTXE bit to 0, wait for two cycles of the base clock, and then set UAnTXE bit to 1 again. Otherwise, initialization may not be executed.
5	UAnRXE	Receive operation disable/enable: 0: Disable receive operation 1: Enable receive operation • To start reception, set the UAnPWR bit to 1 and then set the UAnRXE bit to 1. • To stop reception, clear the UAnRXE bit to 0 and then the UAnPWR bit to 0. • To initialize the reception unit, clear UAnRXE bit to 0, wait for two cycles of the base clock, and then set UAnRXE bit to 1 again. Otherwise, initialization may not be executed.
4	UAnDIR	Transfer direction mode specification (MSB/LSB): 0: MSB first transfer 1: LSB first transfer

Table 15-3 UAnCTL0 register contents (2/2)

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>3, 2</td><td>UAnPS[1:0]</td><td>Parity selection
<table>
<tr><th rowspan="2">UAnPS1</th><th rowspan="2">UAnPS0</th><th colspan="2">Parity selection during</th></tr>
<tr><th>transmission</th><th>reception</th></tr>
<tr><td>0</td><td>0</td><td>No parity output</td><td>Reception with no parity</td></tr>
<tr><td>0</td><td>1</td><td>0 parity output</td><td>Reception with 0 parity</td></tr>
<tr><td>1</td><td>0</td><td>Odd parity output</td><td>Odd parity check</td></tr>
<tr><td>1</td><td>1</td><td>Even parity output</td><td>Even parity check</td></tr>
</table>
• This register is rewritten only when the UAnPWR bit = 0 or the UAnTXE bit = the UAnRXE bit = 0.
• If “Reception with 0 parity” is selected during reception, a parity check is not performed.
Therefore, since the UAnSTR.UAnPE bit is not set, no error interrupt is output.
• When transmission and reception are performed in the LIN format, clear the UAnPS1 and UAnPS0 bits to 00.</td></tr>
<tr><td>1</td><td>UAnCL</td><td>Specification of data character length of 1 frame of transmit/receive data
0: 7 bits
1: 8 bits
This register can be rewritten only when the UAnPWR bit = 0 or the UAnTXE bit =
the UAnRXE bit = 0.</td></tr>
<tr><td>0</td><td>UAnSL</td><td>Specification of length of stop bit for transmit data
0: 1 bit
1: 2 bits
This register can be rewritten only when the UAnPWR bit = 0 or the UAnTXE bit = the UAnRXE bit = 0.</td></tr>
</table>

Note For details of parity, see *“Parity types and operations“ on page 426*.

(2) UAnCTL1 - UARTAn control register 1

The UAnCTL1 register is an 8-bit register used to select the input clock for the UARTAn.

For details, see *“UAnCTL1 - UARTAn control register 1“ on page 429*.

(3) UAnCTL2 - UARTAn control register 2

The UAnCTL2 register is an 8-bit register used to control the baud rate for the UARTAn.

For details, see *“UAnCTL2 - UARTAn control register 2“ on page 430*.

(4) UAnOPT0 - UARTAn option control register 0

The UAnOPT0 register is an 8-bit register that controls the serial transfer operation of the UARTAn register.

Access This register can be read/written in 8-bit or 1-bit units.

Address <base> + 3_H

Initial Value 14_H. This register is cleared by any reset.

7	6	5	4	3	2	1	0
UAnSRF	UAnSRT	UAnSTT	UAnSBL2	UAnSBL1	UAnSBL0	UAnTDL	UAnRDL
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 15-4 UAnOPT0 register contents (1/2)

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>7</td><td>UAnSRF</td><td>SBF reception flag:
0: When the UAnCTL0.UAnPWR bit = UAnCTL0.UAnRXE bit = 0 are set. Also upon normal end of SBF reception.
1: During SBF reception
• SBF (Sync Brake Field) reception is judged during LIN communication.
• The UAnSRF bit is held at 1 when an SBF reception error occurs, and then SBF reception is started again.</td></tr>
<tr><td>6</td><td>UAnSRT</td><td>SBF reception trigger:
0: –
1: SBF reception trigger
• This is the SBF reception trigger bit during LIN communication, and when read, “0” is always read. For SBF reception, set the UAnSRT bit (to 1) to enable SBF reception.
• Set the UAnSRT bit after setting the UAnPWR bit = UAnRXE bit = 1.</td></tr>
<tr><td>5</td><td>UAnSTT</td><td>SBF transmission trigger:
0: –
1: SBF transmission trigger [a]
• This is the SBF transmission trigger bit during LIN communication, and when read, “0” is always read.
• Set the UAnSTT bit after setting the UAnPWR bit = UAnTXE bit = 1.</td></tr>
<tr><td>4 to 2</td><td>UAnSBL[2:0]</td><td>SBF transmission length selection:
<table>
<tr><th>UAnSBL2</th><th>UAnSBL1</th><th>UAnSBL0</th><th>SBF transmission length</th></tr>
<tr><td>1</td><td>0</td><td>1</td><td>13-bit output (default value)</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>14-bit output</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>15-bit output</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>16-bit output</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>17-bit output</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>18-bit output</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>19-bit output</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>20-bit output</td></tr>
</table>
This register can be set when the UAnPWR bit = 0 or when the UAnTXE bit = 0.</td></tr>
</table>

Table 15-4 **UAnOPT0 register contents (2/2)**

Bit position	Bit name	Function
1	UAnTDL	Transmit data level bit 0: Normal output of transfer data 1: Inverted output of transfer data • The output level of the TXDAn pin can be inverted using the UAnTDL bit. • This register can be set when the UAnPWR bit = 0 or when the UAnTXE bit = 0.
0	UAnRDL	Receive data level bit 0: Normal input of transfer data 1: Inverted input of transfer data • The output level of the RXDAn pin can be inverted using the UAnRDL bit. • This register can be set when the UAnPWR bit = 0 or the UAnRXE bit = 0.

a) Before starting the SBF transmission by UAnOPT0.UAnSTT = 1 make sure that no data transfer is ongoing, that means check that UAnSTR.UAnTSF = 0.

(5) UAnSTR - UARTAn status register

The UAnSTR register is an 8-bit register that displays the UARTAn transfer status and reception error contents.

Access This register can be read or written in 8-bit or 1-bit units.

Address <base> + 4_H

Initial Value 00_H. This register is cleared by any reset.

The initialization conditions are shown below.

Register/Bit	Initialization conditions
UAnSTR register	• Reset • UAnCTL0.UAnPWR = 0
UAnTSF bit	• UAnCTL0.UAnTXE = 0
UAnPE, UAnFE, UAnOVE bits	• 0 write • UAnCTL0.UAnRXE = 0

7	6	5	4	3	2	1	0
UAnTSF	0	0	0	0	UAnPE	UAnFE	UAnOVE
R	R/W	R/W	R/W	R/W	R/W[a]	R/W[a]	R/W[a]

a) These bits can only be cleared by writing, They cannot be set by writing 1 (even if 1 is written, the value is retained).

Table 15-5 UAnSTR register contents

Bit position	Bit name	Function
7	UAnTSF	Transfer status flag: 0: – When the UAnPWR bit = 0 or the UAnTXE bit = 0 has been set. – When, following transfer completion, there was no next data transfer from UAnTX register 1: Write to UAnTXB bit The UAnTSF bit is always 1 when performing continuous transmission. When initializing the transmission unit, check that the UAnTSF bit = 0 before performing initialization. The transmit data is not guaranteed when initialization is performed while the UAnTSF bit = 1.
2	UAnPE	Parity error flag: 0: – When the UAnPWR bit = 0 or the UAnRXE bit = 0 has been set. – When 0 has been written 1: When parity of data and parity bit do not match during reception. • The operation of the UAnPE bit is controlled by the settings of the UAnCTL0.UAnPS1 and UAnCTL0.UAnPS0 bits. • The UAnPE bit can be read and written, but it can only be cleared by writing 0 to it, and it cannot be set by writing 1 to it. When 1 is written to this bit, the value is retained.
1	UAnFE	Framing error flag 0: – When the UAnPWR bit = 0 or the UAnRXE bit = 0 has been set – When 0 has been written 1: When no stop bit is detected during reception • Only the first bit of the receive data stop bits is checked, regardless of the value of the UAnCTL0.UAnSL bit. • The UAnFE bit can be both read and written, but it can only be cleared by writing 0 to it, and it cannot be set by writing 1 to it. When 1 is written to this bit, the value is retained.
0	UAnOVE	Overrun error flag 0: – When the UAnPWR bit = 0 or the UAnRXE bit = 0 has been set. – When 0 has been written 1: When receive data has been set to the UAnRXB register and the next receive operation is completed before that receive data has been read • When an overrun error occurs, the data is discarded without the next receive data being written to the receive buffer. • The UAnOVE bit can be both read and written, but it can only be cleared by writing 0 to it. When 1 is written to this bit, the value is retained.

(6) UAnRX - UARTAn receive data register

The UAnRX register is an 8-bit buffer register that stores parallel data converted by the receive shift register.

The data stored in the receive shift register is transferred to the UAnRX register upon completion of reception of 1 byte of data.

During LSB-first reception when the data length has been specified as 7 bits, the receive data is transferred to bits 6 to 0 of the UAnRX register and the MSB always becomes 0. During MSB-first reception, the receive data is transferred to bits 7 to 1 of the UAnRX register and the LSB always becomes 0.

When an overrun error (UAnOVE) occurs, the receive data at this time is not transferred to the UAnRX register and is discarded.

Access This register can be read only in 8-bit units.

Address <base> + 6_H

Initial Value FF_H. This register is cleared by any reset.
In addition to reset input, the UAnRX register can be set to FF_H by clearing the UAnCTL0.UAnPWR bit to 0.

<table>
<tr><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="8">Receive data</td></tr>
<tr><td colspan="8">R</td></tr>
</table>

(7) UAnTX - UARTAn transmit data register

The UAnTX register is an 8-bit register used to set transmit data.

Access This register can be read or written in 8-bit units.

Address <base> + 7_H

Initial Value FF_H. This register is cleared by any reset.

<table>
<tr><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="8">Transmit data</td></tr>
<tr><td colspan="8">R/W</td></tr>
</table>

15.4 Interrupt Request Signals

The following three interrupt request signals are generated from UARTAn:

- Reception complete interrupt request signal (INTUAnR)
- Receive error interrupt request signal (INTUAnRE)
- Transmission enable interrupt request signal (INTUAnT)

(1) Reception complete interrupt request signal (INTUAnR)

A reception complete interrupt request signal is output when data is shifted into the receive shift register and transferred to the UAnRX register in the reception enabled status.

In case of erroneous reception, the reception error interrupt INTUanRE is generated instead of INTUAnR.

No reception complete interrupt request signal is generated in the reception disabled status.

(2) Receive error interrupt request signal (INTUAnRE)

A receive error interrupt request is generated if an error condition occurred during reception, as reflected by UAnSTR.UAnPE (parity error flag), UAnSTR.UAnFE (framing error flag), UAnSTR.UAnOVE (overrun error flag).

Note that INTUAnR and INTUAnRE do exclude each other: upon correct reception of data only INTUAnR is generated. In case of a reception error INTUAnRE is generated only.

(3) Transmission enable interrupt request signal (INTUAnT)

If transmit data is transferred from the UAnTX register to the UARTAn transmit shift register with transmission enabled, the transmission enable interrupt request signal is generated.

15.5 Operation

15.5.1 Data format

Full-duplex serial data reception and transmission is performed.

As shown in the figures below, one data frame of transmit/receive data consists of a start bit, character bits, parity bit, and stop bit(s).

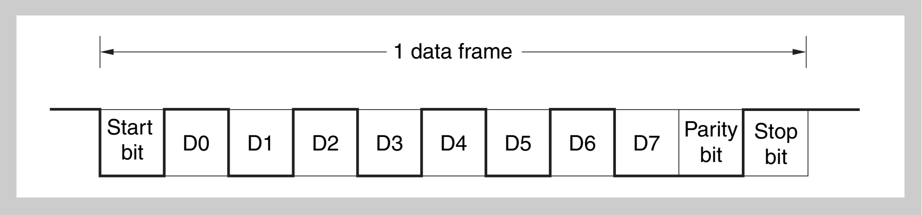
Specification of the character bit length within 1 data frame, parity selection, specification of the stop bit length, and specification of MSB/LSB-first transfer are performed using the UAnCTL0 register.

Moreover, control of UART output/inverted output for the TXDAn bit is performed using the UAnOPT0.UAnTDL bit.

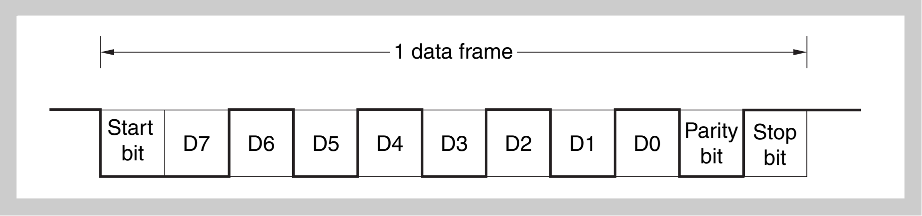
- Start bit..........................1 bit
- Character bits................7 bits/8 bits
- Parity bit ....Even parity/odd parity/0 parity/no parity
- Stop bit..........................1 bit/2 bits

(1) UARTA transmit/receive data format

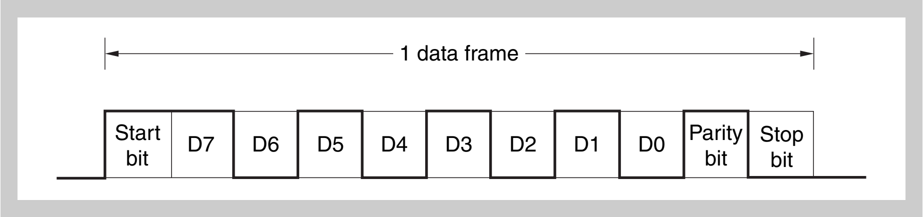
(a) 8-bit data length, LSB first, even parity, 1 stop bit, transfer data: 55H



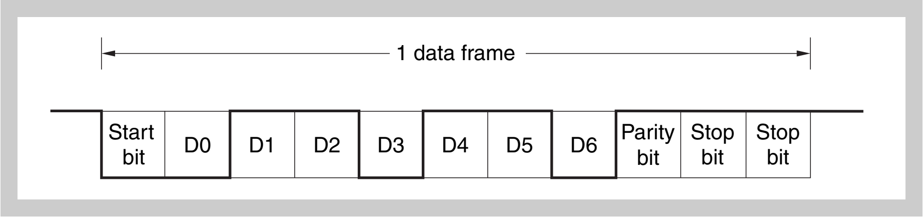
(b) 8-bit data length, MSB first, even parity, 1 stop bit, transfer data: 55H



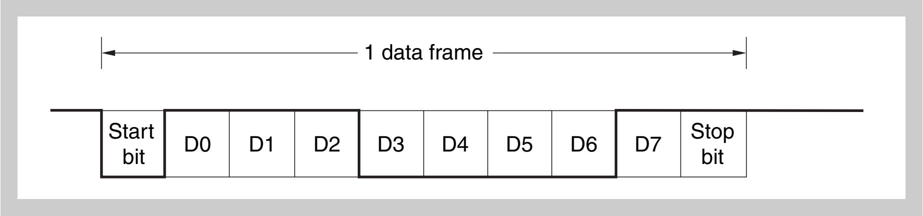
(c) **8-bit data length, MSB first, even parity, 1 stop bit, transfer data: 55H, TXDAn inversion**



(d) **7-bit data length, LSB first, odd parity, 2 stop bits, transfer data: 36H**



(e) **8-bit data length, LSB first, no parity, 1 stop bit, transfer data: 87H**



15.5.2 SBF transmission/reception format

The UARTA has an SBF (Sync Break Field) transmission/reception control function to enable use of the LIN function.

About LIN LIN stands for Local Interconnect Network and is a low-speed (1 to 20 kbps) serial co

mmunication protocol intended to aid the cost reduction of an automotive network.

LIN communication is single-master communication, and up to 15 slaves can be connected to one master.

The LIN slaves are used to control the switches, actuators, and sensors, and these are connected to the LIN master via the LIN network.

Normally, the LIN master is connected to a network such as CAN (Controller Area Network).

In addition, the LIN bus uses a single-wire method and is connected to the nodes via a transceiver that complies with ISO9141.

In the LIN protocol, the master transmits a frame with baud rate information and the slave receives it and corrects the baud rate error. Therefore, communication is possible when the baud rate error in the slave is ±15% or less.

Figure 15-2 and *Figure 15-3* outline the transmission and reception manipulations of LIN.

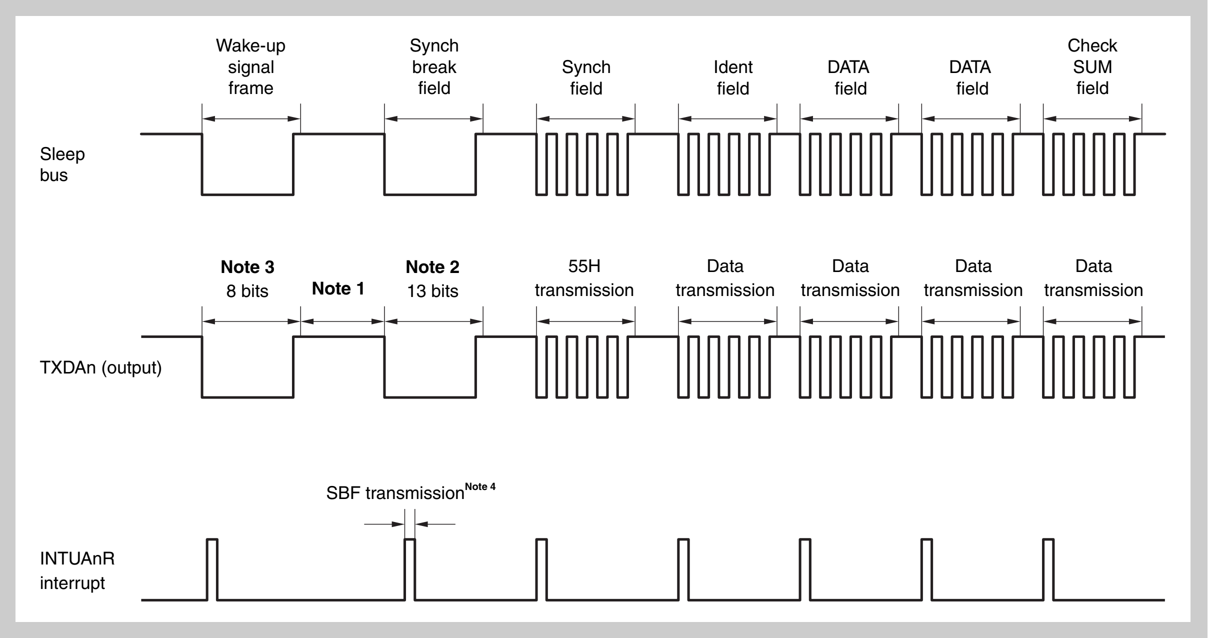


Figure 15-2 LIN transmission manipulation outline

Note

1. The interval between each field is controlled by software.
2. SBF output is performed by hardware. The output width is the bit length set by the UAnOPT0.UAnSBL2 to UAnOPT0.UAnSBL0 bits. If even finer output width adjustments are required, such adjustments can be performed using the UAnCTLn.UAnBRS7 to UAnCTLn.UAnBRS0 bits.
3. 80H transfer in the 8-bit mode is substituted for the wakeup signal frame.

4. A transmission enable interrupt request signal (INTUAnT) is output at the start of each transmission. The INTUAnT signal is also output at the start of each SBF transmission.

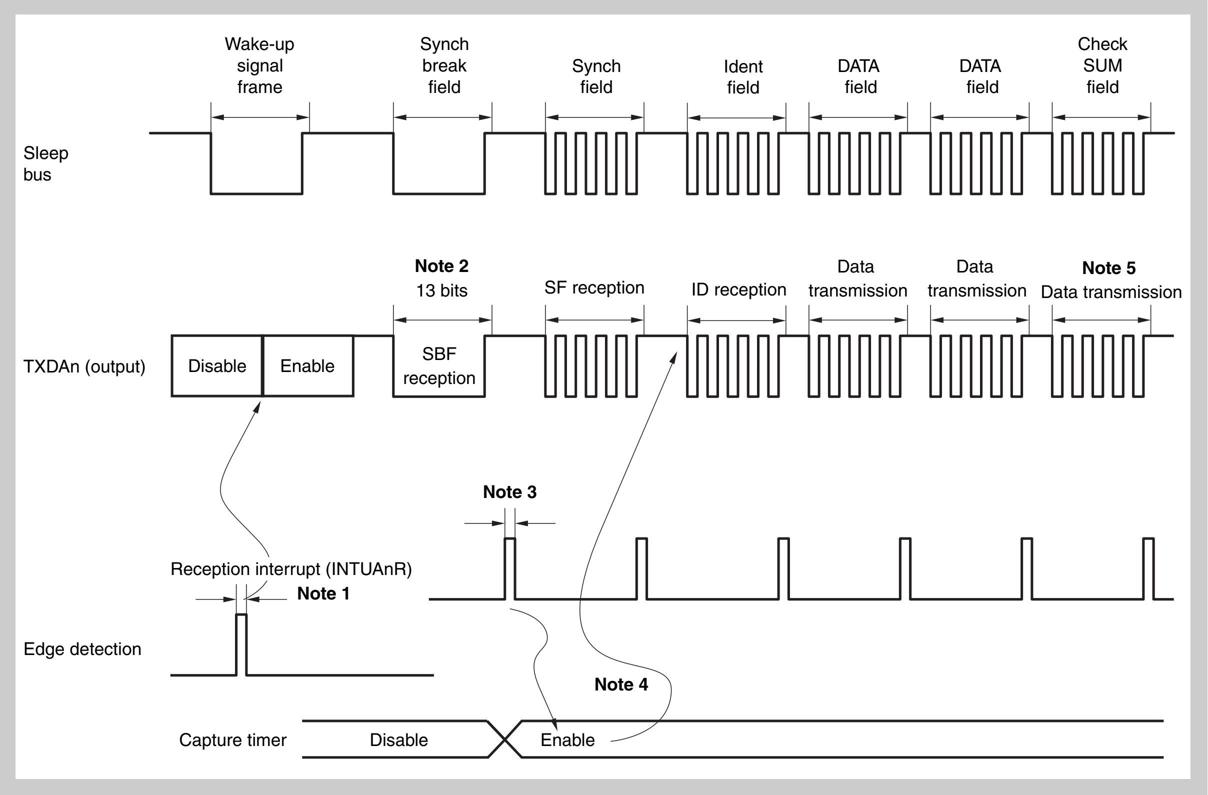


Figure 15-3 LIN reception manipulation outline

Note
1. The wakeup signal is sent by the pin edge detector, UARTAn is enabled, and the SBF reception mode is set.
2. The receive operation is performed until detection of the stop bit. Upon detection of SBF reception of 11 or more bits, normal SBF reception end is judged, and an interrupt signal is output. Upon detection of SBF reception of less than 11 bits, an SBF reception error is judged, no interrupt signal is output, and the mode returns to the SBF reception mode.
3. If SBF reception ends normally, an interrupt request signal is output. The timer is enabled by an SBF reception complete interrupt. Moreover, error detection for the UAnSTR.UAnOVE, UAnSTR.UAnPE, and UAnSTR.UAnFE bits is suppressed and UART communication error detection processing and UARTAn receive shift register and data transfer of the UAnRX register are not performed. The UARTAn receive shift register holds the initial value, FFH.
4. The RXDAn pin is connected to TI (capture input) of the timer, the transfer rate is calculated, and the baud rate error is calculated. The value of the UAnCTL2 register obtained by correcting the baud rate error after dropping UARTA enable is set again, causing the status to become the reception status.
5. Check-sum field distinctions are made by software. UARTAn is initialized following CSF reception, and the processing for setting the SBF reception mode again is performed by software.

15.5.3 SBF transmission

When the UAnCTL0.UAnPWR bit = UAnCTL0.UAnTXE bit = 1, the transmission enabled status is entered, and SBF transmission is started by setting (to 1) the SBF transmission trigger (UAnOPT0.UAnSTT bit).

Thereafter, a low level the width of bits 13 to 20 specified by the UAnOPT0.UAnSBL2 to UAnOPT0.UAnSBL0 bits is output. A transmission enable interrupt request signal (INTUAnT) is generated upon SBF transmission start. Following the end of SBF transmission, the UAnSTT bit is automatically cleared. Thereafter, the UART transmission mode is restored.

Transmission is suspended until the data to be transmitted next is written to the UAnTX register, or until the SBF transmission trigger (UAnSTT bit) is set.

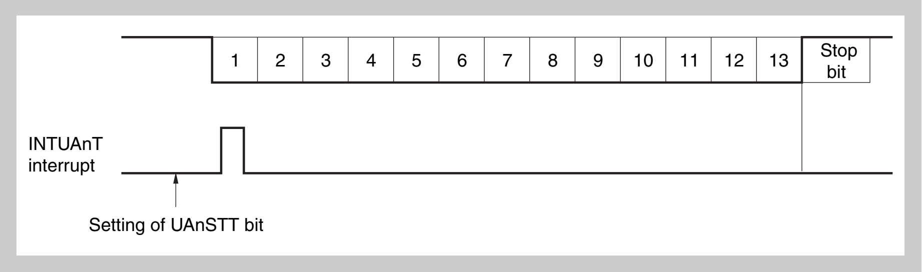


Figure 15-4 SBF transmission

15.5.4 SBF reception

The reception enabled status is achieved by setting the UAnCTL0.UAnPWR bit to 1 and then setting the UAnCTL0.UAnRX bit to 1.

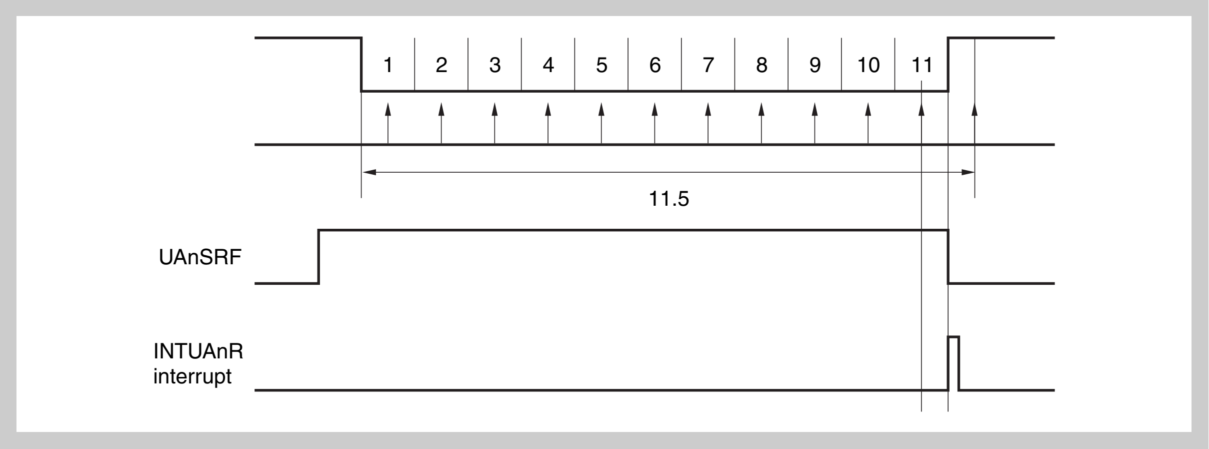
The SBF reception wait status is set by setting the SBF reception trigger (UAnOPT0.UAnSTR bit) to 1.

In the SBF reception wait status, similarly to the UART reception wait status, the RXDAn pin is monitored and start bit detection is performed.

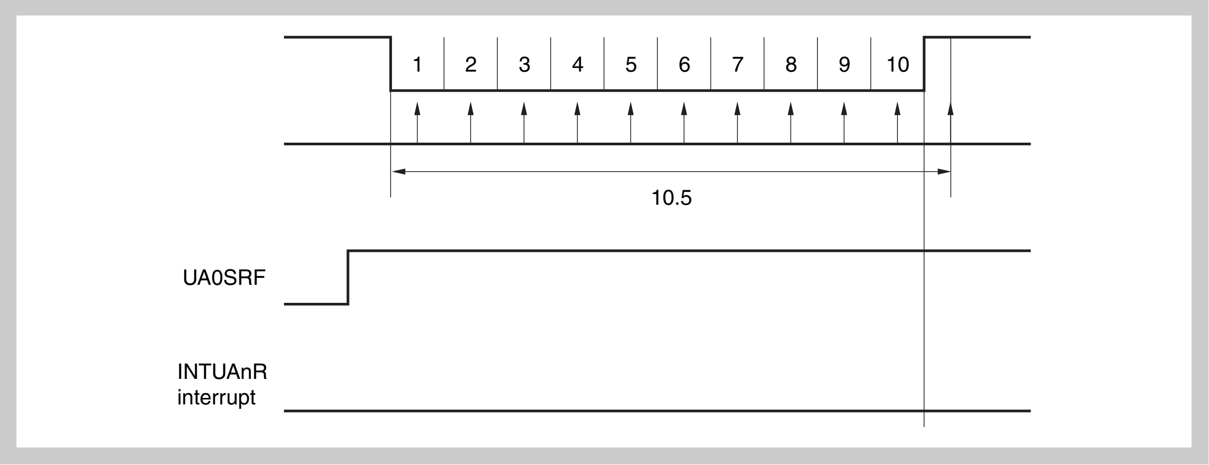
Following detection of the start bit, reception is started and the internal counter counts up according to the set baud rate.

When a stop bit is received, if the SBF width is 11 or more bits, normal processing is judged and a reception complete interrupt request signal (INTUAnR) is output. The UAnOPT0.UAnSRF bit is automatically cleared and SBF reception ends. Error detection for the UAnSTR.UAnOVE, UAnSTR.UAnPE, and UAnSTR.UAnFE bits is suppressed and UART communication error detection processing is not performed. Moreover, data transfer of the UARTAn reception shift register and UAnRX register is not performed and FFH, the initial value, is held. If the SBF width is 10 or fewer bits, reception is terminated as error processing without outputting an interrupt, and the SBF reception mode is returned to. The UAnSRF bit is not cleared at this time.

(a) Normal SBF reception (detection of stop bit in more than 10.5 bits)



(b) SBF reception error (detection of stop bit in 10.5 or fewer bits)



15.5.5 UART transmission

A high level is output to the TXDAn pin by setting the UAnCTL0.UAnPWR bit to 1.

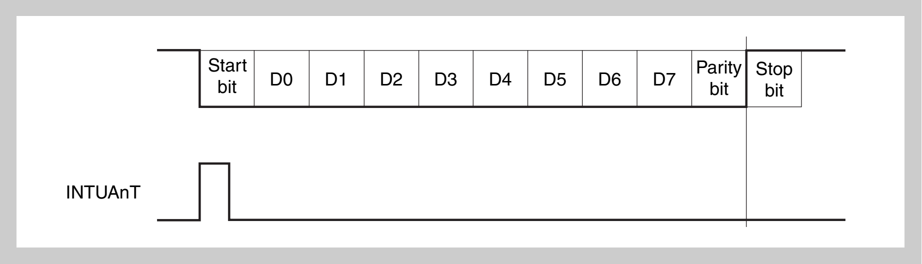
Next, the transmission enabled status is set by setting the UAnCTL0.UAnTXE bit to 1, and transmission is started by writing transmit data to the UAnTX register. The start bit, parity bit, and stop bit are automatically added.

Since the CTS (transmit enable signal) input pin is not provided in UARTAn, use a port to check that reception is enabled at the transmit destination.

The data in the UAnTX register is transferred to the UARTAn transmit shift register upon the start of the transmit operation.

A transmission enable interrupt request signal (INTUAnT) is generated upon completion of transmission of the data of the UAnTX register to the UARTAn transmit shift register, and thereafter the contents of the UARTAn transmit shift register are output to the TXDAn pin.

Write of the next transmit data to the UAnTX register is enabled by generating the INTUAnT signal.



Note LSB first

15.5.6 Continuous transmission procedure

UARTAn can write the next transmit data to the UAnTX register when the UARTAn transmit shift register starts the shift operation. The transmit timing of the UARTAn transmit shift register can be judged from the transmission enable interrupt request signal (INTUAnT).

An efficient communication rate is realized by writing the data to be transmitted next to the UAnTX register during transfer.

Caution During continuous transmission execution, perform initialization after checking that the UAnSTR.UAnTSF bit is 0. The transmit data cannot be guaranteed when initialization is performed while the UAnTSF bit is 1.

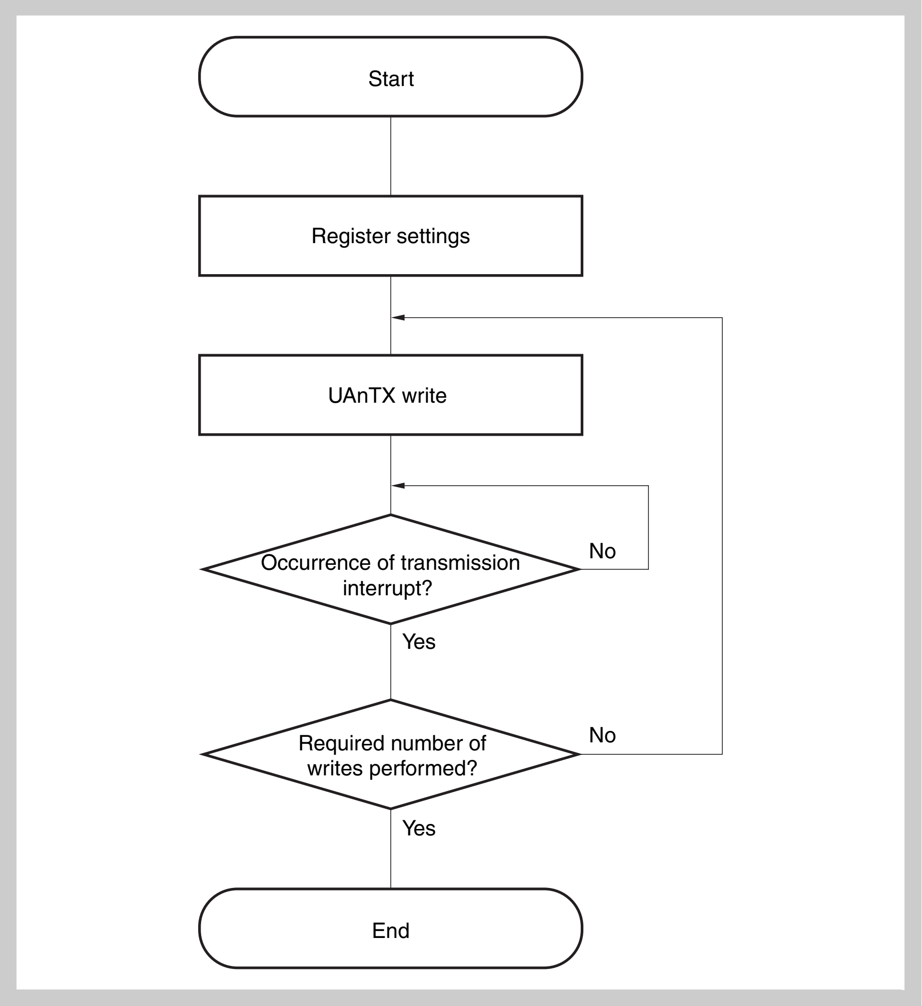

Start
Register settings
UAnTX write
Occurrence of transmission interrupt?
No
Yes
Required number of writes performed?
No
Yes
End

Figure 15-5 Continuous transmission processing flow

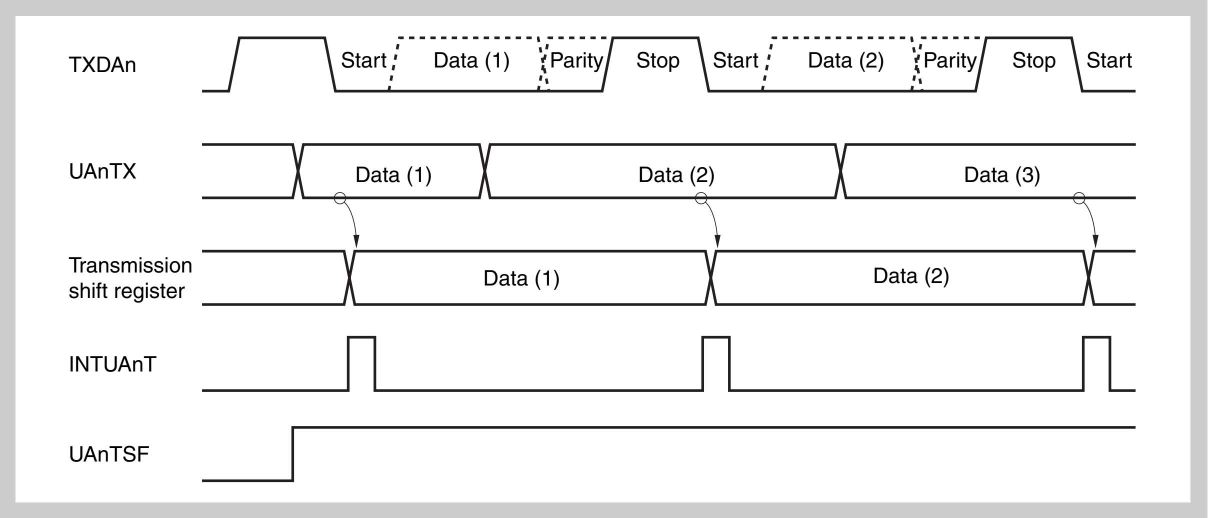


Figure 15-6 Continuous transmission operation timing —transmission start

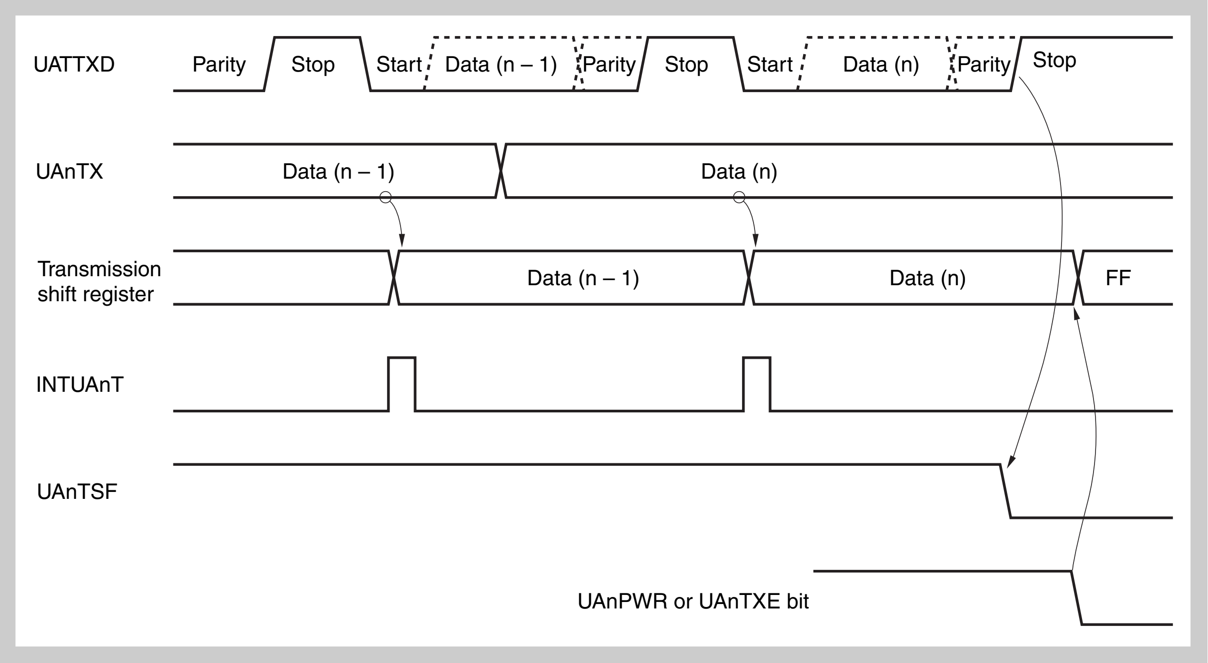


Figure 15-7 Continuous transmission operation timing—transmission end

15.5.7 UART reception

The reception wait status is set by setting the UAnCTL0.UAnPWR bit to 1 and then setting the UAnCTL0.UAnRXE bit to 1. In the reception wait status, the RXDAn pin is monitored and start bit detection is performed.

Start bit detection is performed using a two-step detection routine.

First the rising edge of the RXDAn pin is detected and sampling is started at the falling edge. The start bit is recognized if the RXDAn pin is low level at the start bit sampling point. After a start bit has been recognized, the receive operation starts, and serial data is saved to the UARTAn receive shift register according to the set baud rate.

When the reception complete interrupt request signal (INTUAnR) is output upon reception of the stop bit, the data of the UARTAn receive shift register is written to the UAnRX register. However, if an overrun error (UAnSTR.UAnOVE bit) occurs, the receive data at this time is not written to the UAnRX register and is discarded.

Even if a parity error (UAnSTR.UAnPE bit) or a framing error (UAnSTR.UAnFE bit) occurs during reception, reception continues until the reception position of the first stop bit, and INTUAnR is output following reception completion.

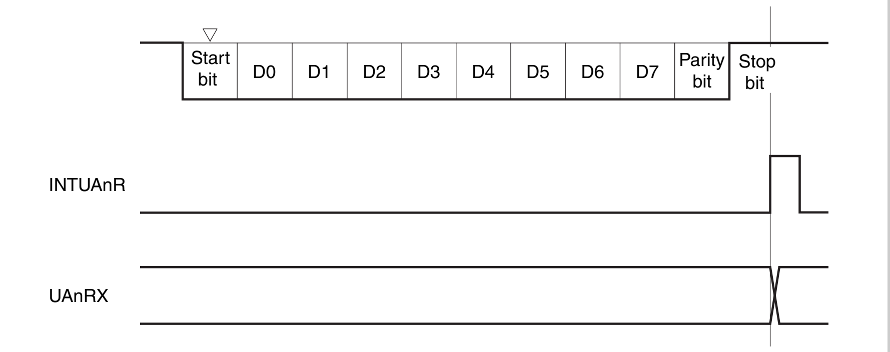


Figure 15-8 UART reception

Caution

1. Be sure to read the UAnRX register even when a reception error occurs. If the UAnRX register is not read, an overrun error occurs during reception of the next data, and reception errors continue occurring indefinitely.
2. The operation during reception is performed assuming that there is only one stop bit. A second stop bit is ignored.
3. When reception is completed, read the UAnRX register after the reception complete interrupt request signal (INTUAnR) has been generated, and clear the UAnPWR or UAnRXE bit to 0. If the UAnPWR or UAnRXE bit is cleared to 0 before the INTUAnR signal is generated, the read value of the UAnRX register cannot be guaranteed.

4. If receive completion processing (INTUAnR signal generation) of UARTAn and the UAnPWR bit = 0 or UAnRXE bit = 0 conflict, the INTUAnR signal may be generated in spite of these being no data stored in the UAnRX register.
   To complete reception without waiting INTUAnR signal generation, be sure to clear (0) the interrupt request flag (UAnRIF) of the UAnRIC register, after setting (1) the interrupt mask flag (UAnRMK) of the interrupt control register (UAnRIC) and then set (1) the UAnPWR bit = 0 or UAnRXE bit = 0.

15.5.8 Reception errors

Errors during a receive operation are of three types: parity errors, framing errors, and overrun errors. Data reception result error flags are set in the UAnSTR register and a reception error interrupt request signal (INTUAnRE) is output when an error occurs.

It is possible to ascertain which error occurred during reception by reading the contents of the UAnSTR register.

Clear the reception error flag by writing 0 to it after reading it.

Table 15-6 Reception error causes

Error flag	Reception error	Cause
UAnPE	Parity error	Received parity bit does not match the setting
UAnFE	Framing error	Stop bit not detected
UAnOVE	Overrun error	Reception of next data completed before data was read from receive buffer

Note Note that even in case of a parity or framing error, data is transferred from the receive shift register to the receive data register UAnRX. Consequently the data from UAnRX must be read. Otherwise an overrun error UAnSTR.UAnOVE will occur at reception of the next data.

In case of an overrun error, the receive shift register data is not transferred to UAnRX, thus the previous data is not overwritten.

15.5.9 Parity types and operations

Caution When using the LIN function, fix the UAnPS1 and UAnPS0 bits of the UAnCTL0 register to 00.

The parity bit is used to detect bit errors in the communication data. Normally the same parity is used on the transmission side and the reception side.

In the case of even parity and odd parity, it is possible to detect odd-count bit errors. In the case of 0 parity and no parity, errors cannot be detected.

(1) Even parity

- During transmission
  The number of bits whose value is “1” among the transmit data, including the parity bit, is controlled so as to be an even number. The parity bit values are as follows.
  - Odd number of bits whose value is “1” among transmit data:1
  - Even number of bits whose value is “1” among transmit data:0
- During reception
  The number of bits whose value is “1” among the reception data, including the parity bit, is counted, and if it is an odd number, a parity error is output.

(2) Odd parity

- During transmission
  Opposite to even parity, the number of bits whose value is “1” among the transmit data, including the parity bit, is controlled so that it is an odd number. The parity bit values are as follows.
  - Odd number of bits whose value is “1” among transmit data: 0
  - Even number of bits whose value is “1” among transmit data: 1
- During reception
  The number of bits whose value is “1” among the receive data, including the parity bit, is counted, and if it is an even number, a parity error is output.

(3) 0 parity

During transmission, the parity bit is always made 0, regardless of the transmit data.

During reception, parity bit check is not performed. Therefore, no parity error occurs, regardless of whether the parity bit is 0 or 1.

(4) No parity

No parity bit is added to the transmit data.

Reception is performed assuming that there is no parity bit. No parity error occurs since there is no parity bit.

15.5.10 Receive data noise filter

This filter samples the RXDAn pin using the base clock of the prescaler output.

When the same sampling value is read twice, the match detector output changes and the RXDAn signal is sampled as the input data. Therefore, data not exceeding 2 clock width is judged to be noise and is not delivered to the internal circuit (see *Figure 15-10*). See *“Base clock“ on page 428* regarding the base clock.

Moreover, since the circuit is as shown in *Figure 15-9*, the processing that goes on within the receive operation is delayed by 3 clocks in relation to the external signal status.

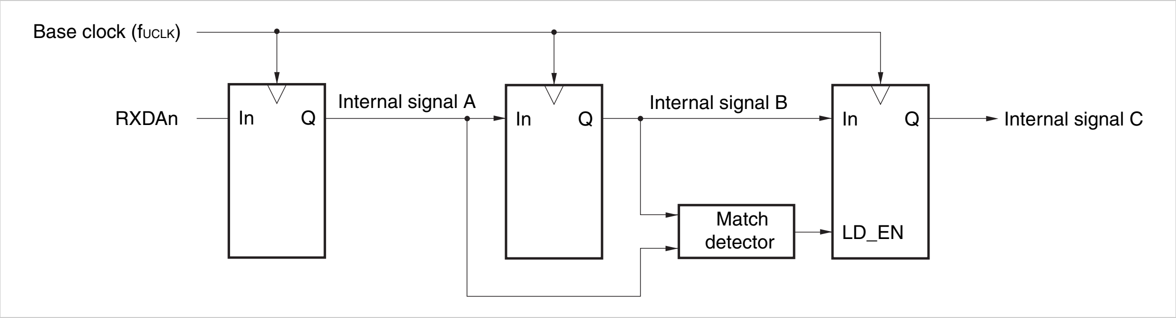


Figure 15-9 Noise filter circuit

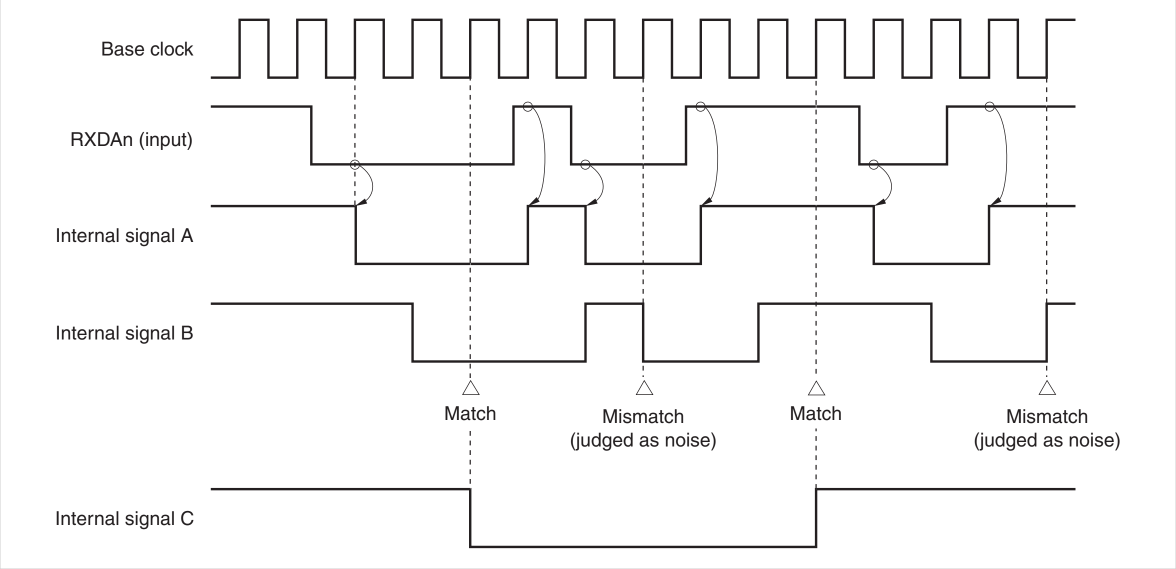


Figure 15-10 Timing of RXDAn signal judged as noise

15.6 Baud Rate Generator

The dedicated baud rate generator consists of a source clock selector block and an 8-bit programmable counter, and generates a serial clock during transmission and reception with UARTAn. Regarding the serial clock, a dedicated baud rate generator output can be selected for each channel.

There is an 8-bit counter for transmission and another one for reception.

15.6.1 Baud Rate Generator configuration

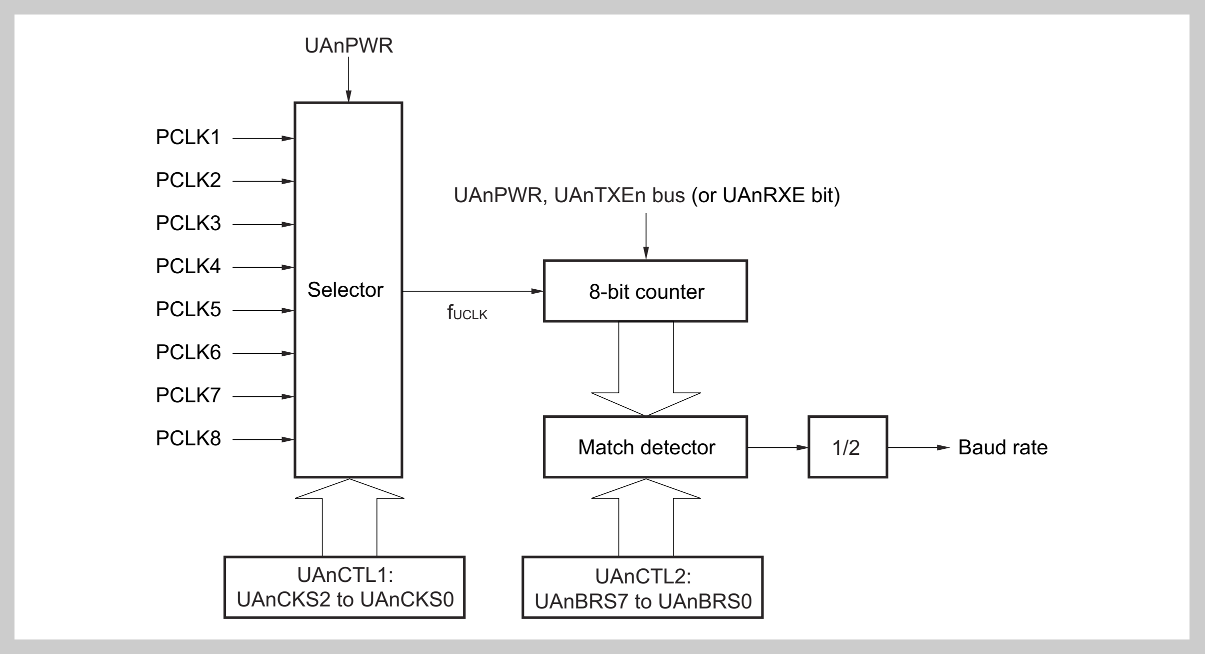


Figure 15-11 Configuration of baud rate generator

(a) Base clock

When the UAnCTL0.UAnPWR bit is 1, the clock selected by the UAnCTL1.UAnCKS[2:0] bits is supplied to the 8-bit counter. This clock is called the base clock. When the UAnPWR bit = 0, f_{UCLK} is fixed to the low level.

(b) Serial clock generation

A serial clock can be generated by setting the UAnCTL1 register and the UAnCTL2 register.

The base clock is selected by UAnCTL1.UAnCKS2 to UAnCTL1.UAnCKS0 bits.

The frequency division value for the 8-bit counter can be set using the UAnCTL2.UAnBRS[7:0] bits.

15.6.2 Baud Rate Generator registers

(1) UAnCTL1 - UARTAn control register 1

The UAnCTL1 register is an 8-bit register that selects the UARTAn base clock.

Access This register can be read or written in 8-bit units.

Address <base> + 1_H

Initial Value 00_H. This register is cleared by any reset.

7	6	5	4	3	2	1	0
0	0	0	0	0	UAnCKS2	UAnCKS1	UAnCKS0
R	R	R	R	R	R/W	R/W	R/W

Caution Clear the UAnCTL0.UAnPWR bit to 0 before rewriting the UAnCTL1 register.

Table 15-7 UAnCTL1 register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>2 to 0</td><td>UAnCKS[2:0]</td><td>Base clock f_{UCLK} selection:
<table>
<tr><th>UAnCKS2</th><th>UAnCKS1</th><th>UAnCKS0</th><th>Base clock f_{UCLK}</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>PCLK1</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>PCLK2</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>PCLK3</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>PCLK4</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>PCLK5</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>PCLK6</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>PCLK7</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>PCLK8</td></tr>
</table>
</td></tr>
</table>

(2) UAnCTL2 - UARTAn control register 2

The UAnCTL2 register is an 8-bit register that selects the baud rate (serial transfer speed) clock of UARTAn.

Access This register can be read or written in 8-bit units.

Address <base> + 1_H

Initial Value FF_H. This register is cleared by any reset.

7	6	5	4	3	2	1	0
UAnBRS7	UAnBRS6	UAnBRS5	UAnBRS4	UAnBRS3	UAnBRS2	UAnBRS1	UAnBRS0
R	R	R	R	R	R/W	R/W	R/W

Caution Clear the UAnCTL0.UAnPWR bit to 0 or clear the UAnTXE and UAnRXE bits to 00 before rewriting the UAnCTL2 register.

Table 15-8 UAnCTL2 register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>2 to 0</td><td>UAnBRS[7:0]</td><td>Serial clock setting
<table>
<tr><th>UAn BR S7</th><th>UAn BR S6</th><th>UAn BR S5</th><th>UAn BR S4</th><th>UAn BR S3</th><th>UAn BR S2</th><th>UAn BR S1</th><th>UAn BR S0</th><th>k</th><th>Serial clock</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>x</td><td>x</td><td>x</td><td>Setting prohibited</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>4</td><td>$f_{UCLK}/4$</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td><td>5</td><td>$f_{UCLK}/5$</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>6</td><td>$f_{UCLK}/6$</td></tr>
<tr><td>:</td><td>:</td><td>:</td><td>:</td><td>:</td><td>:</td><td>:</td><td>:</td><td>:</td><td>:</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>252</td><td>$f_{UCLK}/252$</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>253</td><td>$f_{UCLK}/253$</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>254</td><td>$f_{UCLK}/254$</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>255</td><td>$f_{UCLK}/255$</td></tr>
</table>
</td></tr>
</table>

Note f_{UCLK}: clock frequency selected by UAnCTL1.UAnCKS[2:0]

15.6.3 Baud rate calculation

The baud rate is obtained by the following equation.

$$\text{Baud rate} = \frac{f_{UCLK}}{2 \times k} \text{ [bps]}$$

f_{UCLK} = Frequency of base clock selected by the UAnCTL1.UAnCKS[2:0]

k = Value set using the UAnCTL2.UAnBRS[7:0] bits
(k = 4, 5, 6, …, 255)

15.6.4 Baud rate error

The baud rate error is obtained by the following equation.

$$\text{Error (\%)} = \left(\frac{\text{Actual baud rate (baud rate with error)}}{\text{Target baud rate (correct baud rate)}} - 1\right) \times 100 \text{ [\%]}$$

Caution

1. The baud rate error during transmission must be within the error tolerance on the receiving side.
2. The baud rate error during reception must satisfy the range indicated in chapter *“Allowable baud rate range during reception“ on page 432*.

Example Base clock frequency = 8MHz

Setting value of

- UAnDTL1.UAnCKS[2:0] = 001B (PCLK2 = 4MHz)

- UAnCTL2.UAnBRS[7:0] = 0000 1101B (k = 13)

Target baud rate = 153,600 bps

Baud rate = 4MHz/ (2 × 13) = 153,846 [bps]

Error = (153,846/153,600 – 1) × 100

= 0.160 [%]

15.6.5 Baud rate setting example

Table 15-9 Baud rate generator setting data (1/2)

Target baud rate	UAnCTL1		UAnCTL2		Effective baud rate [bps]	Baud rate error (%)
[bps]	Selector	Divider	Divider k			
300	07H	128	68H	104	300.48	0.16
600	07H	128	34H	52	600.96	0.16
1,200	07H	128	1AH	26	1,201.92	0.16
2,400	07H	128	0DH	13	2,403.85	0.16
4,800	06H	64	0DH	13	4,807.69	0.16
9,600	05H	32	0DH	13	9,615.38	0.16
19,200	04H	16	0DH	13	19,230.77	0.16
31,250	05H	32	04H	4	31,250.00	0.00

Table 15-9 Baud rate generator setting data (2/2)

38,400	03H	8	0DH	13	38,461.54	0.16
76,800	02H	4	0DH	13	76,923.08	0.16
153,600	01H	2	0DH	13	153,846.15	0.16
312,500	00H	1	0DH	13	307,692.31	–1.54

Note *Table 15-9* assumes normal operation mode, i.e. PCLK1 = 8 MHz.

15.6.6 Allowable baud rate range during reception

The baud rate error range at the destination that is allowable during reception is shown below.

Caution The baud rate error during reception must be set within the allowable error range using the following equation.

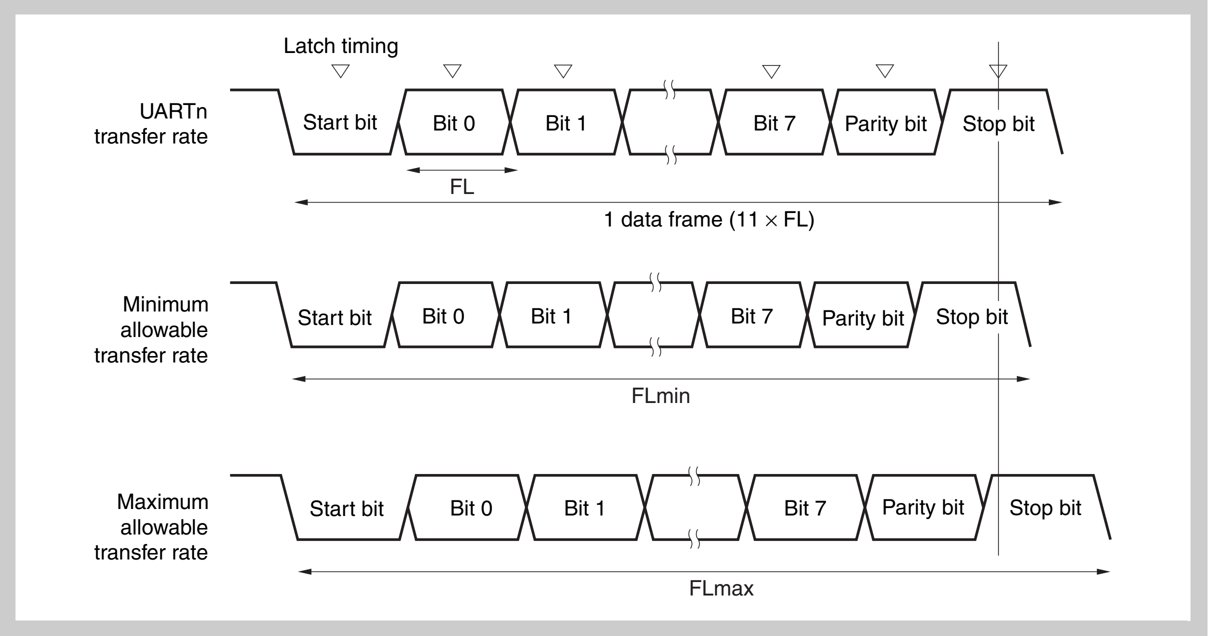


Figure 15-12 Allowable baud rate range during reception

As shown in *Figure 15-12*, the receive data latch timing is determined by the counter set using the UAnCTL2 register following start bit detection. The transmit data can be normally received if up to the last data (stop bit) can be received in time for this latch timing.

When this is applied to 11-bit reception, the following is the theoretical result.

FL = (Brate)$^{-1}$

Brate: UARTAn baud rate

k: Setting value of UAnCTL2.UAnBRS[7:0]

FL: 1-bit data length

Latch timing margin: 2 clocks

Minimum allowable transfer rate:

$$FL_{min} = 11 \times FL - \frac{k-2}{2k} \times FL = \frac{21k+2}{2k} \times FL$$

Therefore, the maximum baud rate that can be received by the destination is as follows.

$$BRmax = (FLmin/11)^{-1} = \frac{22k}{21k+2} \times Brate$$

Similarly, obtaining the following maximum allowable transfer rate yields the following.

$$\frac{10}{11} \times FLmax = 11 \times FL - \frac{k+2}{2k} \times FL = \frac{21k-2}{2k} \times FL$$

$$FLmax = \frac{21k-2}{20k} \times FL \times 11$$

Therefore, the minimum baud rate that can be received by the destination is as follows.

$$BRmin = (FLmax/11)^{-1} = \frac{20k}{21k-2} \times Brate$$

Obtaining the allowable baud rate error for UARTAn and the destination from the above-described equations for obtaining the minimum and maximum baud rate values yields the following.

Table 15-10 Maximum/minimum allowable baud rate error

Division ratio (k)	Maximum allowable baud rate error	Minimum allowable baud rate error
4	+2.32%	-2.43%
8	+3.52%	-3.61%
20	+4.26%	-4.30%
50	+4.56%	-4.58%
100	+4.66%	-4.67%
255	+4.72%	-4.72%

Note 1. The reception accuracy depends on the bit count in 1 frame, the input clock frequency, and the division ratio (k). The higher the input clock frequency and the larger the division ratio (k), the higher the accuracy.

2. k: Setting value of UAnCTL2.UAnBRS[7:0]

15.6.7 Baud rate during continuous transmission

During continuous transmission, the transfer rate from the stop bit to the next start bit is usually 2 base clocks longer. However, timing initialization is performed via start bit detection by the receiving side, so this has no influence on the transfer result.

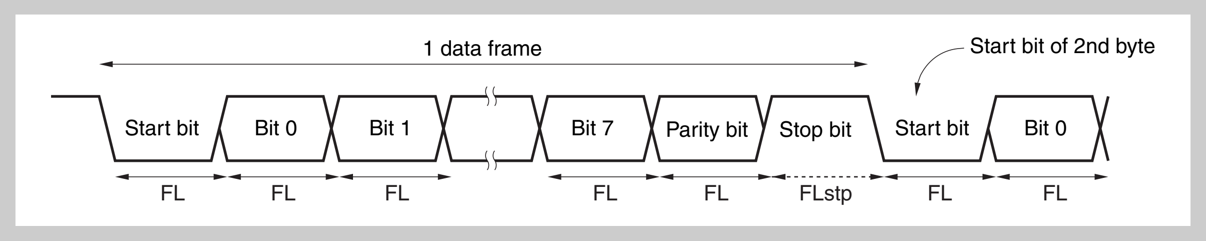


Figure 15-13 Transfer rate during continuous transfer

Assuming 1 bit data length: FL; stop bit length: FLstp; and base clock frequency: f_{UCLK}, we obtain the following equation.

$$FLstp = FL + 2/f_{UCLK}$$

Therefore, the transfer rate during continuous transmission is as follows.

$$\text{Transfer rate} = 11 \times FL + (2/f_{UCLK})$$

15.7 Cautions

15.7.1 UARTAn behaviour during and after power save mode

When the clock supply to UARTAn is stopped (for example, in IDLE or STOP mode), the operation stops with each register retaining the value it had immediately before the clock supply was stopped. The TXDAn pin output also holds and outputs the value it had immediately before the clock supply was stopped. However, the operation is not guaranteed after the clock supply is resumed. Therefore, after the clock supply is resumed, the circuits should be initialized by setting the UAnCTL0.UAnPWR, UAnCTL0.UAnRXEn, and UAnCTL0.UAnTXEn bits to 000.

15.7.2 UARTAn behaviour during debugger break

The UARTAn continues to operate in debugger break-mode, provided all clocks are continuing.

Reception When the UARTAn is in reception mode and an external device is sending data during debugger break-mode the UARTAn may produce an overflow error as it continues to receive data during break-mode.

Thus the overflow error flag UAnSTR.UAnOVE may be set and the reception error interrupt INTUAnRE may be generated. Further all following received data are discarded.

Note that the reception error interrupt INTUAnRE will not be served during break-mode, but after resuming run-mode, provided the interrupt controller is configured accordingly.

Transmission If the debugger’s break-mode is entered while the UARTAn is transmitting data, the transmission is completed, and finally a transmission enable interrupt request INTUAnT is generated.

Note that the transmission enable interrupt INTUAnT will not be served during break-mode, but after resuming run-mode, provided the interrupt controller is configured accordingly.

15.7.3 UARTAn operation stop

If both of the following actions in UARTAn happen at the same time the INTUAnR signal may be generated inadvertently and no data is stored in the UAnRX register:

- INTUAnR is generated due to completion of a serial receive operation,
- UAnPWR bit or UAnRXE bit is cleared (set to 0).

Workaround To avoid the generation of the INTUAnR signal when UAnPWR bit or UAnRXE bit is cleared (set to 0) do the following:

1. Set (set to 1) the interrupt mask flag (UAnRMK) of the interrupt control

register (UAnRIC),

2. Clear (set to 0) the UAnPWR bit or UAnRXE bit,
3. Clear (set to 0) the interrupt request flag (UAnRIF) of the UAnRIC register.

Chapter 16 Clocked Serial Interface (CSIB)

The V850E/Dx3 - DG3 microcontrollers have following instances of the clocked serial interface CSIB:

CSIB	All devices
Instances	2
Names	CSIB0 to CSIB1

Throughout this chapter, the individual instances of clocked serial interface are identified by “n”, for example CSIBn, or CBnCTL0 for the control register 0 of CSIBn.

16.1 Features

- Transfer rate: 8 Mbps to 2 kbps (using dedicated baud rate generator)
  The maximum transfer rate is the maximum transfer rate of the digital circuitry. It does neither regard any output buffer driver strength limitation nor the external capacitive load. Both might reduce the practically achievable maximum baud rate.
- Master mode and slave mode selectable
- 8-bit to 16-bit transfer, 3-wire serial interface
- 3 interrupt request signals (INTCBnT, INTCBnR, INTCBnRE)
- Serial clock and data phase switchable
- Transfer data length selectable in 1-bit units between 8 and 16 bits
- Transfer data MSB-first/LSB-first switchable
- 3-wire transfer

3-wire transfer	SOBn:	Serial data output
	SIBn:	Serial data input
	SCKBn:	Serial clock input/output

  Transmission mode, reception mode, and transmission/reception mode specifiable
- Dedicated baud rate generator for each interface instance
- Modulated and stable clock sources available

16.2 Configuration

The following shows the block diagram of CSIBn.

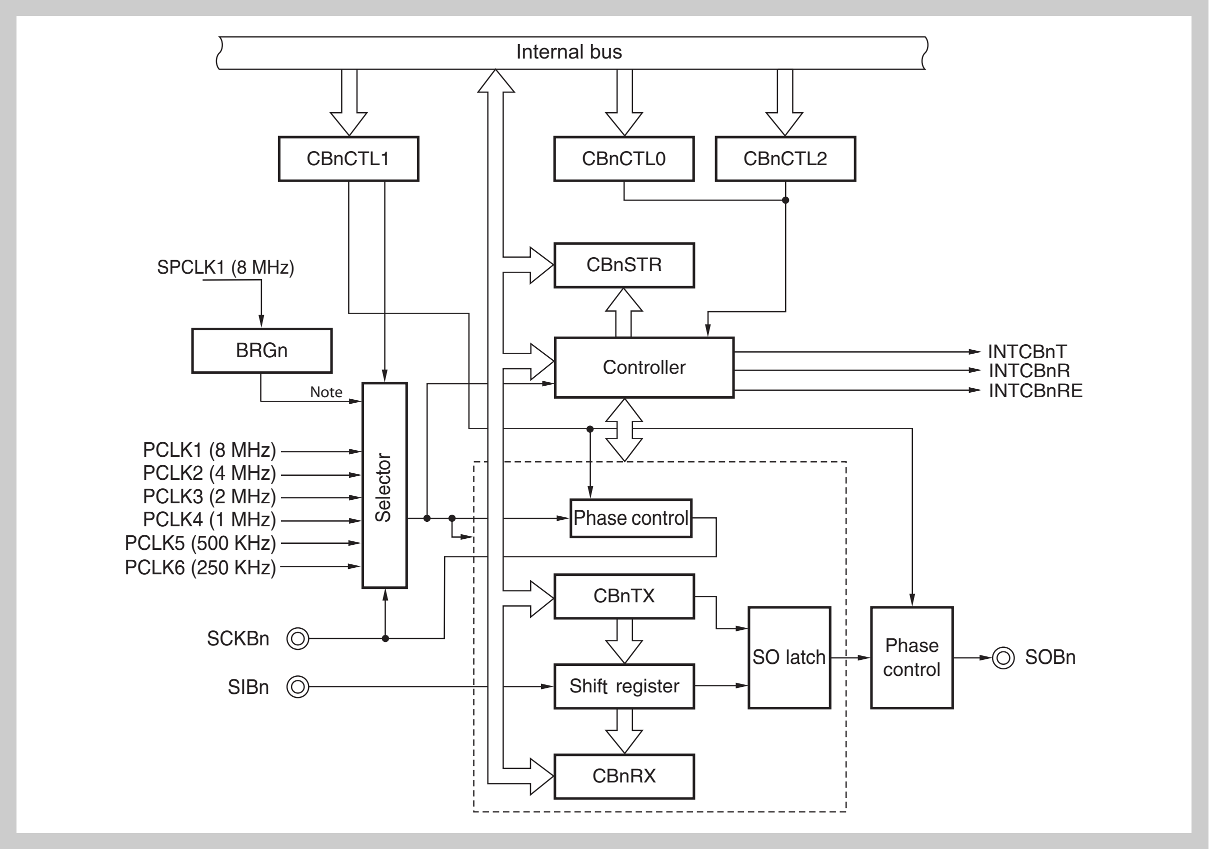


Figure 16-1 Block diagram of CSIBn

Note The clock is generated by the dedicated baud rate generator BRGn.

16.3 CSIB Control Registers

The clocked serial interfaces CSIBn are controlled and operated by means of the following registers:

Table 16-1 CSIBn registers overview

Register name	Shortcut	Address
CSIBn control register 0	CBnCTL0	<base>
CSIBn control register 1	CBnCTL1	<base> + 1_H
CSIBn control register 2	CBnCTL2	<base> + 2_H
CSIBn status register	CBnSTR	<base> + 3_H
CSIBn receive data register	CBnRX	<base> + 4_H
CSIBn transmit data register	CBnTX	<base> + 6_H

Table 16-2 CSIBn register base address

Timer	Base address
CSIB0	FFFF $FD00_H$
CSIB1	FFFF $FD10_H$

(1) CBnCTL0 - CSIBn control register 0

CBnCTL0 is a register that controls the CSIBn serial transfer operation.

Access This register can be read/written in 8-bit or 1-bit units.

Address <base>

Initial Value 01_H. This register is cleared by any reset.

7	6	5	4	3	2	1	0
CBnPWR	CBnTXE[a]	CBnRXE[a]	CBnDIR[a]	0	0	CBnTMS[a]	CBnSCE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

[a] These bits can only be rewritten when the CBnPWR bit = 0. However, CBnPWR bit = 1 can also be set at the same time as rewriting these bits.

Table 16-3 CBnCTL0 register contents (1/2)

Bit position	Bit name	Function
7	CBnPWR	CSIBn operation disable/enable: 0: Disable CSIBn operation and reset the CSIBn registers 1: Enable CSIBn operation The CBnPWR bit controls the CSIBn operation and resets the internal circuit.
6	CBnTXE	Transmit operation disable/enable: 0: Disable transmit operation 1: Enable transmit operation The SOBn output is low level when the CBnTXE bit is 0.
5	CBnRXE	Receive operation disable/enable: 0: Disable receive operation 1: Enable receive operation When the CBnRXE bit is cleared to 0, no reception complete interrupt is output even when the prescribed data is transferred in order to disable the receive operation, and the receive data (CBnRX register) is not updated.

Table 16-3 CBnCTL0 register contents (2/2)

Bit position	Bit name	Function
4	CBnDIR	Transfer direction mode specification (MSB/LSB): 0: MSB first transfer 1: LSB first transfer
1	CBnTMS	Transfer mode specification (MSB/LSB): 0: Single transfer mode 1: Continuous transfer mode
0	CBnSCE	Specification of start transfer disable/enable: 0: Communication start trigger invalid 1: Communication start trigger valid • In master mode This bit enables or disables the communication start trigger. (a)In single transmission or transmission/reception mode, or continuous transmission or continuous transmission/reception mode A communication operation can be started only when the CBnSCE bit is 1. Set the CBnSCE bit to 1. (b)In single reception mode Clear the CBnSCE bit to 0 before reading the receive data (CBnRX register). If the CBnSCE bit is read while it is 1, the next communication operation is started. (c)In continuous reception mode Clear the CBnSCE bit to 0 one communication clock before reception of the last data is completed The CBnSCE bit is not cleared to 0 one communication clock before the completion of the last data reception, the next communication operation is automatically started. • In slave mode This bit enables or disables the communication start trigger. Set the CBnSCE bit to 1.

(2) CBnCTL1 - CSIBn control register 1

CBnCTL1 is an 8-bit register that controls the CSIBn serial transfer operation.

Access This register can be read/written in 8-bit or 1-bit units.

Address <base> + 1_H

Initial Value 00_H. This register is cleared by any reset.

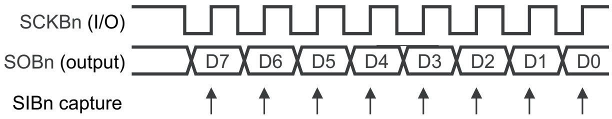
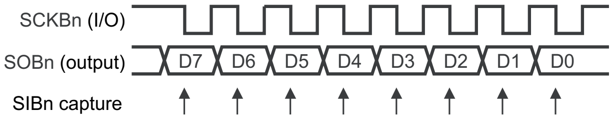
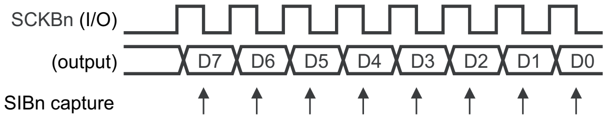
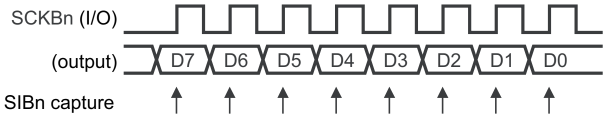
7	6	5	4	3	2	1	0
0	0	0	CBnCKP	CBnDAP	CBnCKS2	CBnCKS1	CBnCKS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Caution The CBnCTL1 register can be rewritten only when the CBnCTL0.CBnPWR bit = 0.

Table 16-4 CBnCTL1 register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>4
3</td><td>CBnCKP
CBnDAP</td><td>Specification of data transmission/reception timing in relation to SCKBn.
Refer to <i>Table 16-5</i>.</td></tr>
<tr><td>2 to 0</td><td>CBnCKS[2:0]</td><td>Communication clock setting
<table>
<tr><th>CBnCKS2</th><th>CBnCKS1</th><th>CBnCKS0</th><th>Communication clock</th><th>Mode</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>f_{BRGn}</td><td>Master</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>f_{PCLK1}</td><td>Master</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>f_{PCLK2}</td><td>Master</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>f_{PCLK3}</td><td>Master</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>f_{PCLK4}</td><td>Master</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>f_{PCLK5}</td><td>Master</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>f_{PCLK6}</td><td>Master</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>External clock SCKBn</td><td>Slave</td></tr>
</table></td></tr>
</table>

Table 16-5 Specification of data transmission/reception timing in relation to SCKBn

Communication type	CBnCKP	CBnDAP	SIBn/SOBN timing in relation to SCKBn
Communication type 1	0	0	 SCKBn (I/O) SOBn (output) SIBn capture D7 D6 D5 D4 D3 D2 D1 D0
Communication type 2	0	1	 SCKBn (I/O) SOBn (output) SIBn capture D7 D6 D5 D4 D3 D2 D1 D0
Communication type 3	1	0	 SCKBn (I/O) (output) SIBn capture D7 D6 D5 D4 D3 D2 D1 D0
Communication type 4	1	1	 SCKBn (I/O) (output) SIBn capture D7 D6 D5 D4 D3 D2 D1 D0

(3) CBnCTL2 - CSIBn control register 2

CBnCTL2 is an 8-bit register that controls the number of CSIBn serial transfer bits.

Access This register can be read/written in 8-bit units.

Address <base> + 2_H

Initial Value 00_H. This register is cleared by any reset.

7	6	5	4	3	2	1	0
0	0	0	0	CBnCL3	CBnCL2	CBnCL1	CBnCL0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Caution The CBnCTL2 register can be rewritten only when the CBnCTL0.CBnPWR bit = 0 or when both the CBnTXE and CBnRXE bits = 0.

Table 16-6 CBnCTL2 register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>3 to 0</td><td>CBnCL[3:0]</td><td>Number of serial transfer bits
<table>
<tr><th>CBnCL3</th><th>CBnCL2</th><th>CBnCL1</th><th>CBnCL0</th><th>Number of serial transfer bits</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>8 bits</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>9 bits</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>10 bits</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>11 bits</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td><td>12 bits</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1</td><td>13 bits</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td><td>14 bits</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>1</td><td>15 bits</td></tr>
<tr><td>1</td><td>x</td><td>x</td><td>x</td><td>16 bits</td></tr>
</table></td></tr>
</table>

Note If the number of transfer bits is other than 8 or 16, prepare and use data stuffed from the LSB of the CBnTX and CBnRX registers.

(a) Transfer data length change function

The CSIBn transfer data length can be set in 1-bit units between 8 and 16 bits using the CBnCTL2.CBnCL3 to CBnCTL2.CBnCL0 bits.

When the transfer bit length is set to a value other than 16 bits, set the data to the CBnTX or CBnRX register starting from the LSB, regardless of whether the transfer start bit is the MSB or LSB. Any data can be set for the higher bits that are not used, but the receive data becomes 0 following serial transfer.

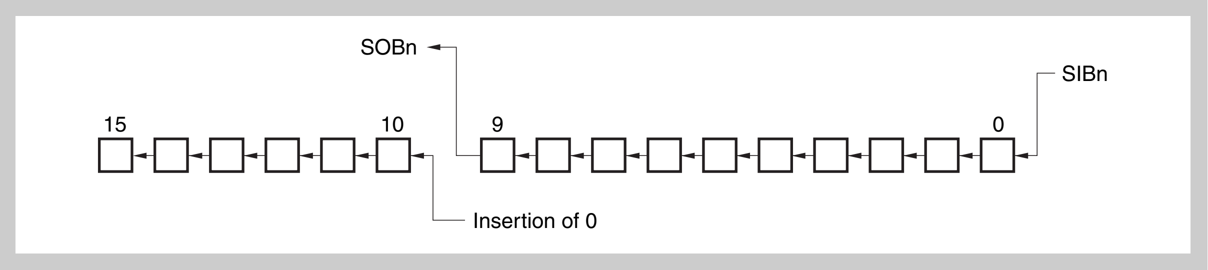


Figure 16-2 (i) Transfer bit length = 10 bits, MSB first

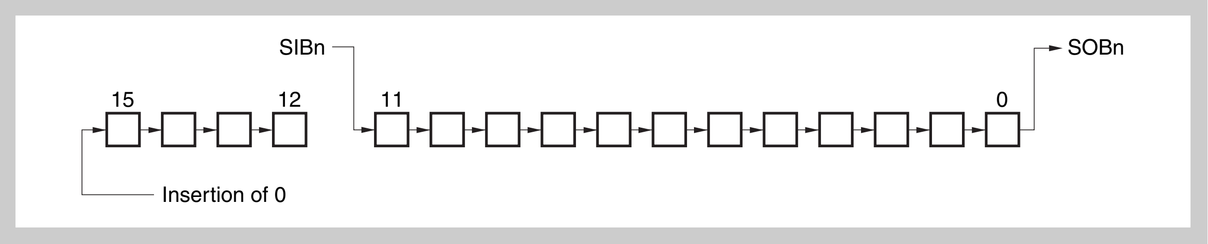


Figure 16-3 (ii) Transfer bit length = 12 bits, LSB first

(4) CBnSTR - CSIBn status register

CBnSTR is an 8-bit register that displays the CSIBn status.

Access This register can be read/written in 8-bit or 1-bit units.
Bit CBnTSF is read-only.

Address <base> + 3_H

Initial Value 00_H. This register is cleared by any reset.
In addition to reset input, the CBnSTR register can be initialized by clearing the CBnCTL0.CBnPWR bit to 0.

7	6	5	4	3	2	1	0
CBnTSF	0	0	0	0	0	0	CBnOVE
R	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 16-7 CBnSTR register contents

Bit position	Bit name	Function
7	CBnTSF	Communication status flag 0: Communication stopped 1: Communicating During transmission, this register is set when data is prepared in the CBnTX register, and during reception, it is set when a dummy read of the CBnRX register is performed. When transfer ends, this flag is cleared to 0 at the last edge of the clock.
0	CBnOVE	Overrun error flag 0: No overrrun 1: Overrun • An overrun error occurs when the next reception starts without performing a CPU read of the value of the receive buffer, upon completion of the receive operation. The CBnOVE flag displays the overrun error occurrence status in this case. • The CBnOVE flag is cleared by writing 0 to it. It cannot be set even by writing 1 to it.

Note In case of an overrun error, the reception error interrupt INTCBnRE behaves different, depending on the transfer mode:

- Continuous transfer mode
  The reception error interrupt INTCBnRE is generated instead of the reception completion interrupt INTCBnR.
- Single transfer mode
  No interrupt is generated.

In either case the overflow flag CBnSTR.CBnOVE is set to 1 and the previous data in CBnRX will be overwritten with the new data.

(5) CBnRX - CSIBn receive data register

The CBnRX register is a 16-bit buffer register that holds receive data.

Access This register can be read-only in 16-bit units.
If the transfer data length is 8 bits, the lower 8 bits of this register are read-only in 8-bit units as the CBnRXL register.

Address <base> + 4_H

Initial Value 0000_H. This register is cleared by any reset.
In addition to reset input, the CBnRX register can be initialized by clearing (to 0) the CBnPWR bit of the CBnCTL0 register.

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

The receive operation is started by reading the CBnRX register in the reception enabled status.

(6) CBnTX - CSIB transmit data register

The CBnTX register is a 16-bit buffer register used to write the CSIBn transfer data.

Access This register can be read/written in 16-bit units.
If the transfer data length is 8 bits, the lower 8 bits of this register are read/write in 8-bit units as the CBnTXL register.

Address <base> + 6_H

Initial Value 0000_H. This register is cleared by any reset.
In addition to reset input, the CBnTX register can be initialized by clearing (to 0) the CBnPWR bit of the CBnCTL0 register.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Transmit data															
R/W															

The transmit operation is started by writing data to the CBnTX register in the transmission enabled status.

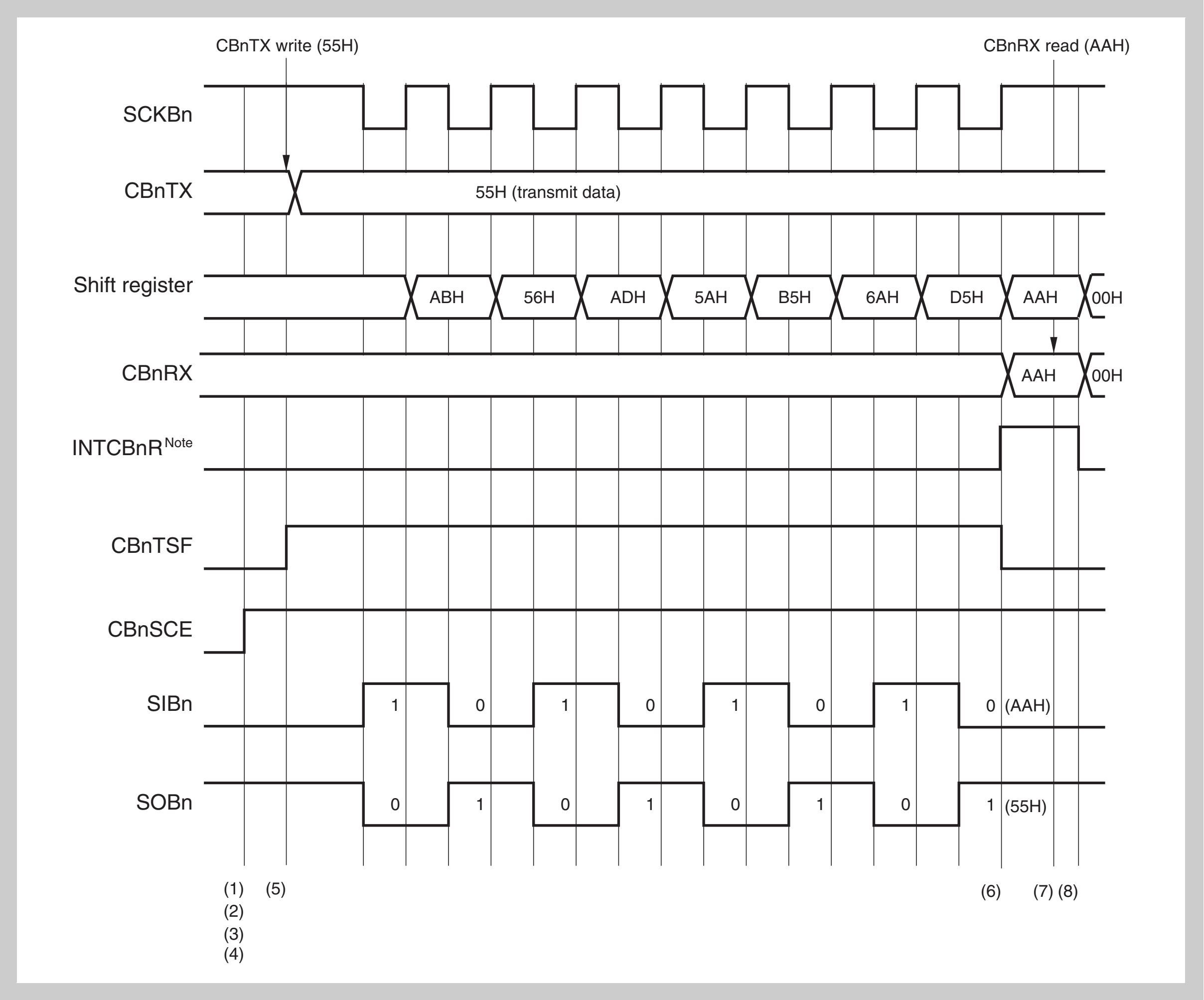
Note The communication start conditions are shown below:

- Transmission mode (CBnTXE bit = 1, CBnRXE bit = 0):
  Write to CBnTX register
- Transmission/reception mode (CBnTXE bit = 1, CBnRXE bit = 1):
  Write to CBnTX register
- Reception mode (CBnTXE bit = 0, CBnRXE bit = 1):
  Read from CBnRX register

16.4 Operation

16.4.1 Single transfer mode (master mode, transmission/reception mode)

MSB first (CBnCTL0.CBnDIR bit = 0), communication type 1 (see 16.4 (2) CSIBn control register 1 (CBnCTL1), transfer data length = 8 bits (CBnCTL2.CBnCL3 to CBnCTL2.CBnCL0 bits = 0, 0, 0, 0)



(1) Clear the CBnCTL0.CBnPWR bit to 0.

(2) Set the CBnCTL1 and CBnCTL2 registers to specify the transfer mode.

(3) Set the CBnTXE, CBnRXE, and CBnSCE bits of the CBnCTL0 register to 1 at the same time as specifying the transfer mode using the CBnDIR bit, to set the transmission/reception enabled status.

(4) Set the CBnPWR bit to 1 to enable the CSIBn operation.

(5) Write transfer data to the CBnTX register (transmission start).

(6) The reception complete interrupt request signal (INTCBnR) is output.

(7) Read the CBnRX register before clearing the CBnPWR bit to 0.

(8) Check that the CBnSTR.CBnTSF bit = 0 and set the CBnPWR bit to 0 to stop operation of CSIBn (end of transmission/reception).

To continue transfer, repeat steps (5) to (7) before (8).

In transmission mode or transmission/reception mode, communication is not started by reading the CBnRX register.

Note

1. In single transmission or single transmission/reception mode, the INTCBnT signal is not generated. When communication is complete, the INTCBnR signal is generated.
2. The processing of steps (3) and (4) can be set simultaneously.

Caution In case the CSIB interface is operating in

- single transmit/reception mode (CBnCTL0.CBnTMS = 0)
- communication type 2 respectively type 4 (CBnCTL1.CBnDAP = 1)

pay attention to following effect:

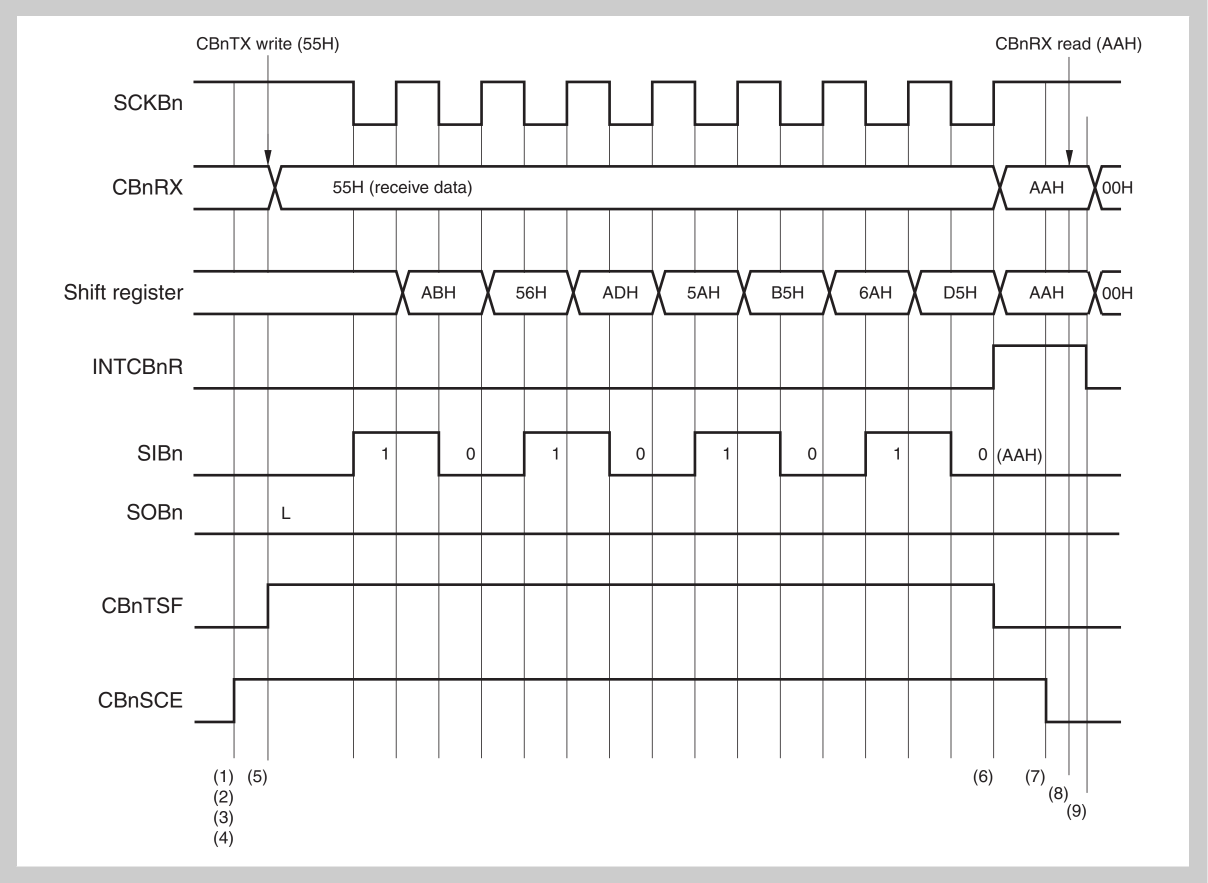
In case the next transmit should be initiated immediately after the occurrence of the reception completion interrupt INTCBnR any write to the CBnTX register is ignored as long as the communication status flag is still reflecting an ongoing communication (CBnTSF = 1). Thus the new transmission will not be started.

For transmitting data continuously use one of the following options:

- Use continuous transfer mode (CBnCTL0.CBnTMS = 1).
- If single transfer mode (CBnCTL0.CBnTMS = 0) should be used, CBnSTR.CBnTSF = 0 needs to be verified before writing data to the CBnTX register.

16.4.2 Single transfer mode (master mode, reception mode)

MSB first (CBnCTL0.CBnDIR bit = 0), communication type 1 (see 16.4 (2) CSIBn control register 1 (CBnCTL1), transfer data length = 8 bits (CBnCTL2.CBnCL3 to CBnCTL2.CBnCL0 bits = 0, 0, 0, 0)



(1) Clear the CBnCTL0.CBnPWR bit to 0.

(2) Set the CBnCTL1 and CBnCTL2 registers to specify the transfer mode.

(3) Set the CBnCTL0.CBnRXE and CBnCTL0.CBnSCE bits to 1, CBnCTL0.TXE to 0, at the same time as specifying the transfer mode using the CBnDIR bit, to set the reception enabled status.

(4) Set the CBnPWR bit to 1 to enable the CSIBn operation.

(5) Perform a dummy read of the CBnRX register (reception start trigger).

(6) The reception complete interrupt request signal (INTCBnR) is output.

(7) Set the CBnSCE bit to 0 to set the final receive data status.

(8) Read the CBnRX register.

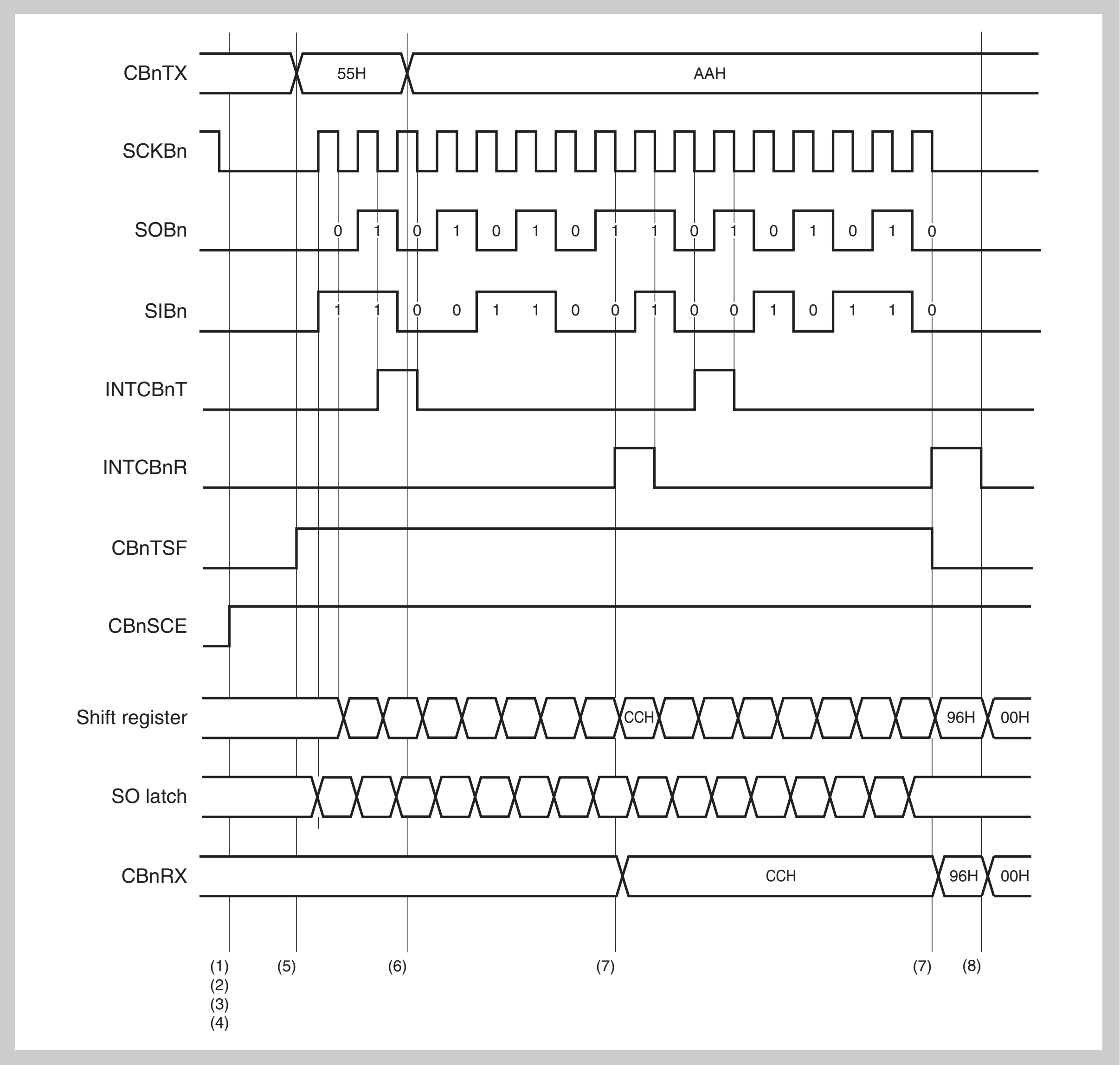
(9) Check that the CBnSTR.CBnTSF bit = 0 and set the CBnPWR bit to 0 to stop the CSIBn operation (end of reception).

To continue transfer, repeat steps (5) and (6) before (7). (At this time, (5) is not a dummy read, but a receive data read combined with the reception trigger.)

Note The processing of steps (3) and (4) can be set simultaneously.

16.4.3 Continuous mode (master mode, transmission/reception mode)

MSB first (CBnCTL0.CBnDIR bit = 0), communication type 3 (see 16.4 (2) CSIBn control register 1 (CBnCTL1)), transfer data length = 8 bits (CBnCTL2.CBnCL3 to CBnCTL2.CBnCL0 bits = 0, 0, 0, 0)



(1) Clear the CBnCTL0.CBnPWR bit to 0.

(2) Set the CBnCTL1 and CBnCTL2 registers to specify the transfer mode.

(3) Set the CBnTXE, CBnRXE, and CBnSCE bits of the CBnCTL0 register to 1 at the same time as specifying the transfer mode using the CBnDIR bit, to set the transmission/reception enabled status.

(4) Set the CBnPWR bit to 1 to enable the CSIBn operation.

(5) Write transfer data to the CBnTX register (transmission start).

(6) The transmission enable interrupt request signal (INTCBnT) is received and transfer data is written to the CBnTX register.

(7) The reception complete interrupt request signal (INTCBnR) is output.

Read the CBnRX register before the next receive data arrives or before the CBnPWR bit is cleared to 0.

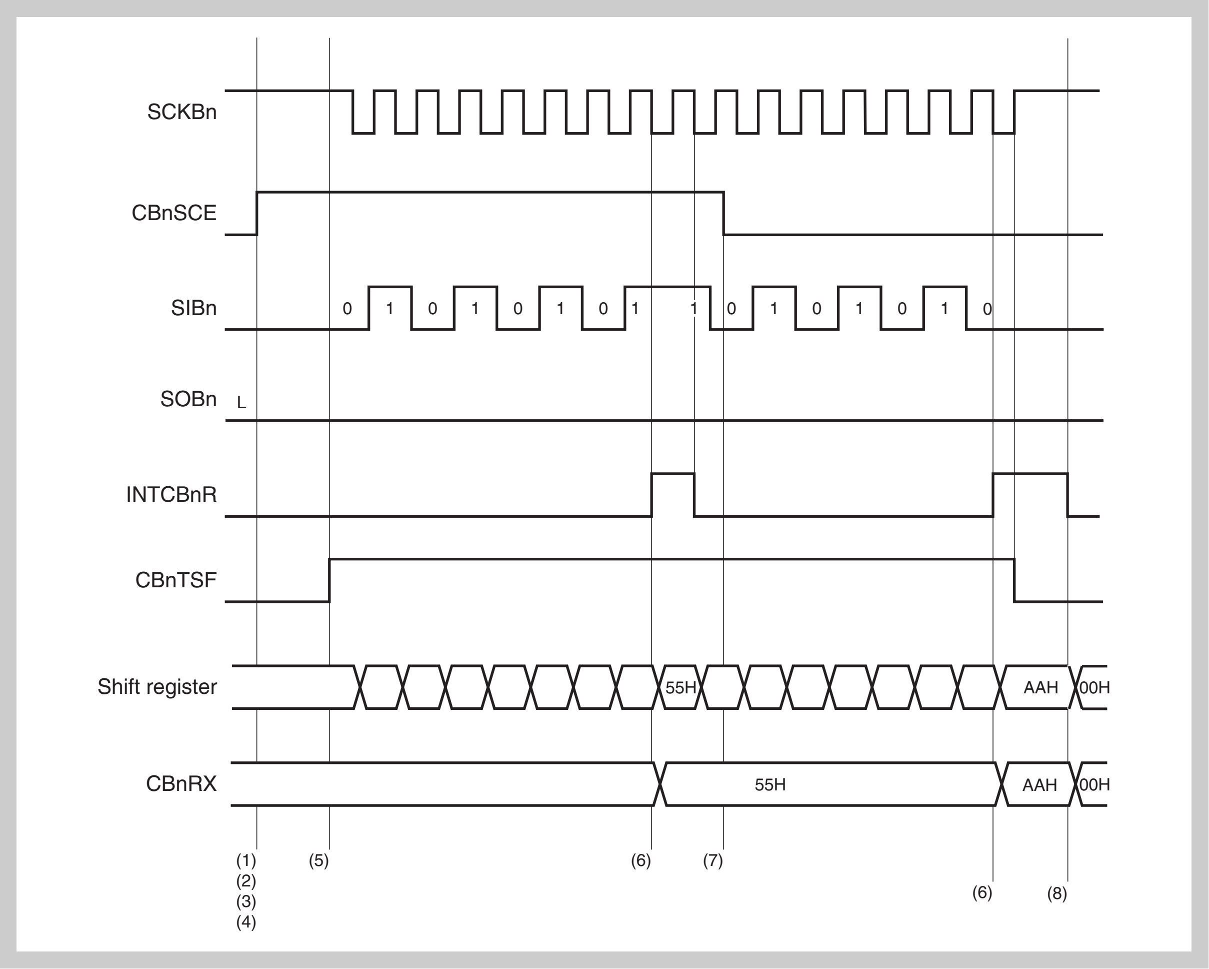
(8) Check that the CBnSTR.CBnTSF bit = 0 and set the CBnPWR bit to 0 to stop the operation of CSIBn (end of transmission/reception).

To continue transfer, repeat steps (5) to (7) before (8).

In transmission mode or transmission/reception mode, the communication is not started by reading the CBnRX register.

16.4.4 Continuous mode (master mode, reception mode)

MSB first (CBnCTL0.CBnDIR bit = 0), communication type 2 (see 16.4 (2) CSIBn control register 1 (CBnCTL1)), transfer data length = 8 bits (CBnCTL2.CBnCL3 to CBnCTL2.CBnCL0 bits = 0, 0, 0, 0)



(1) Clear the CBnCTL0.CBnPWR bit to 0.

(2) Set the CBnCTL1 and CBnCTL2 registers to specify the transfer mode.

(3) Set the CBnCTL0.CBnRXE bit to 1 at the same time as specifying the transfer mode using the CBnDIR bit, to set the reception enabled status.

(4) Set the CBnPWR bit to 1 to enable the CSIBn operation.

(5) Perform a dummy read of the CBnRX register (reception start trigger).

(6) The reception complete interrupt request signal (INTCBnR) is output.

Read the CBnRX register before the next receive data arrives or before the CBnPWR bit is cleared to 0.

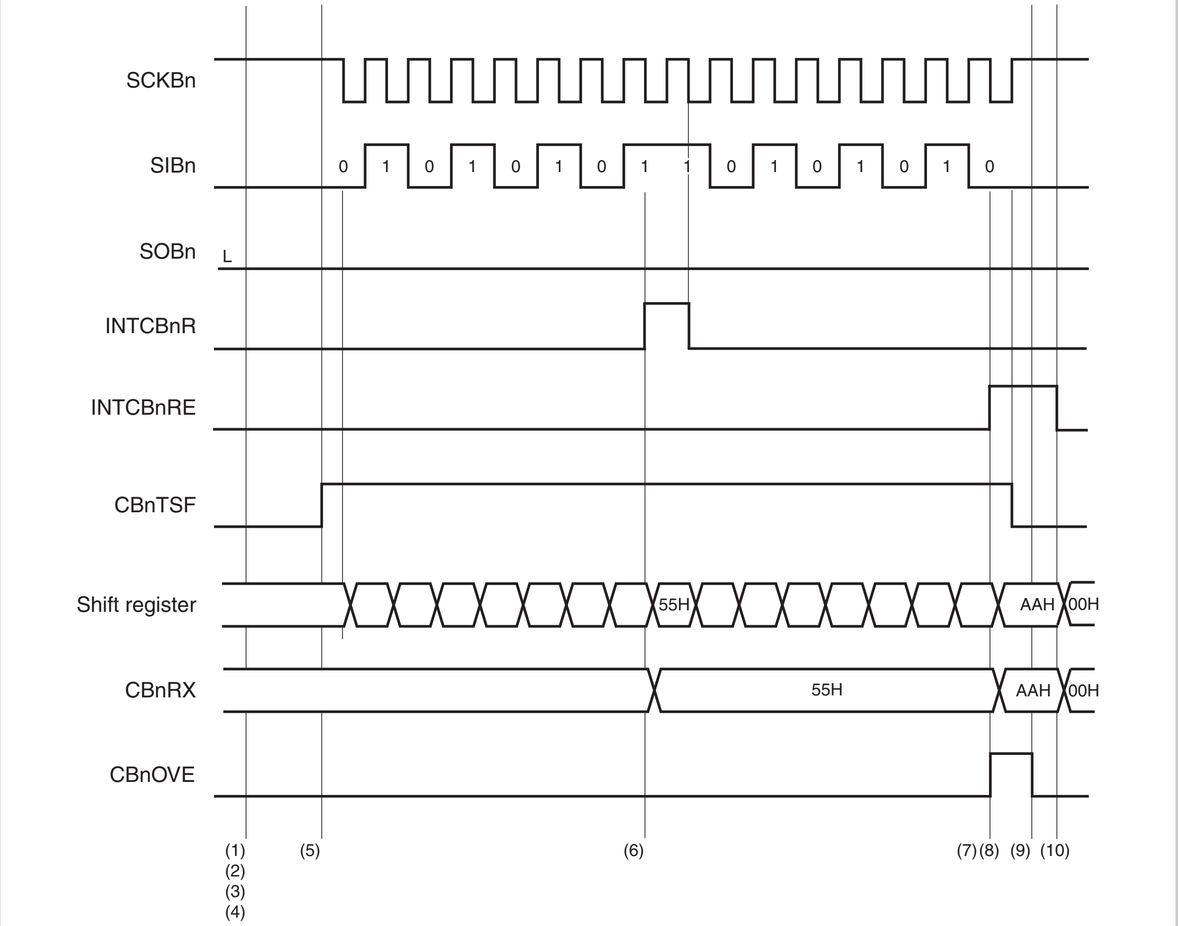
(7) Set the CBnCTL0.CBnSCE bit = 0 while the last data being received to set the final receive data status.

(8) Check that the CBnSTR.CBnTSF bit = 0 and set the CBnPWR bit to 0 to stop the operation of CSIBn (end of reception).

To continue transfer, repeat steps (5) and (6) before (7).

16.4.5 Continuous reception mode (error)

MSB first (CBnCTL0.CBnDIR bit = 0), communication type 2 (see 16.4 (2) CSIBn control register 1 (CBnCTL1)), transfer data length = 8 bits (CBnCTL2.CBnCL3 to CBnCTL2.CBnCL0 bits = 0, 0, 0, 0)



(1) Clear the CBnCTL0.CBnPWR bit to 0.

(2) Set the CBnCTL1 and CBnCTL2 registers to specify the transfer mode.

(3) Set the CBnCTL0.CBnRXE bit to 1 at the same time as specifying the transfer mode using the CBnDIR bit, to set the reception enabled status.

(4) Set the CBnPWR bit = 1 to enable CSIBn operation.

(5) Perform a dummy read of the CBnRX register (reception start trigger).

(6) The reception complete interrupt request signal (INTCBnR) is output.

(7) If the data could not be read before the end of the next transfer, the CBnSTR.CBnOVE flag is set to 1 upon the end of reception and the reception error interrupt INTCBnRE is output.

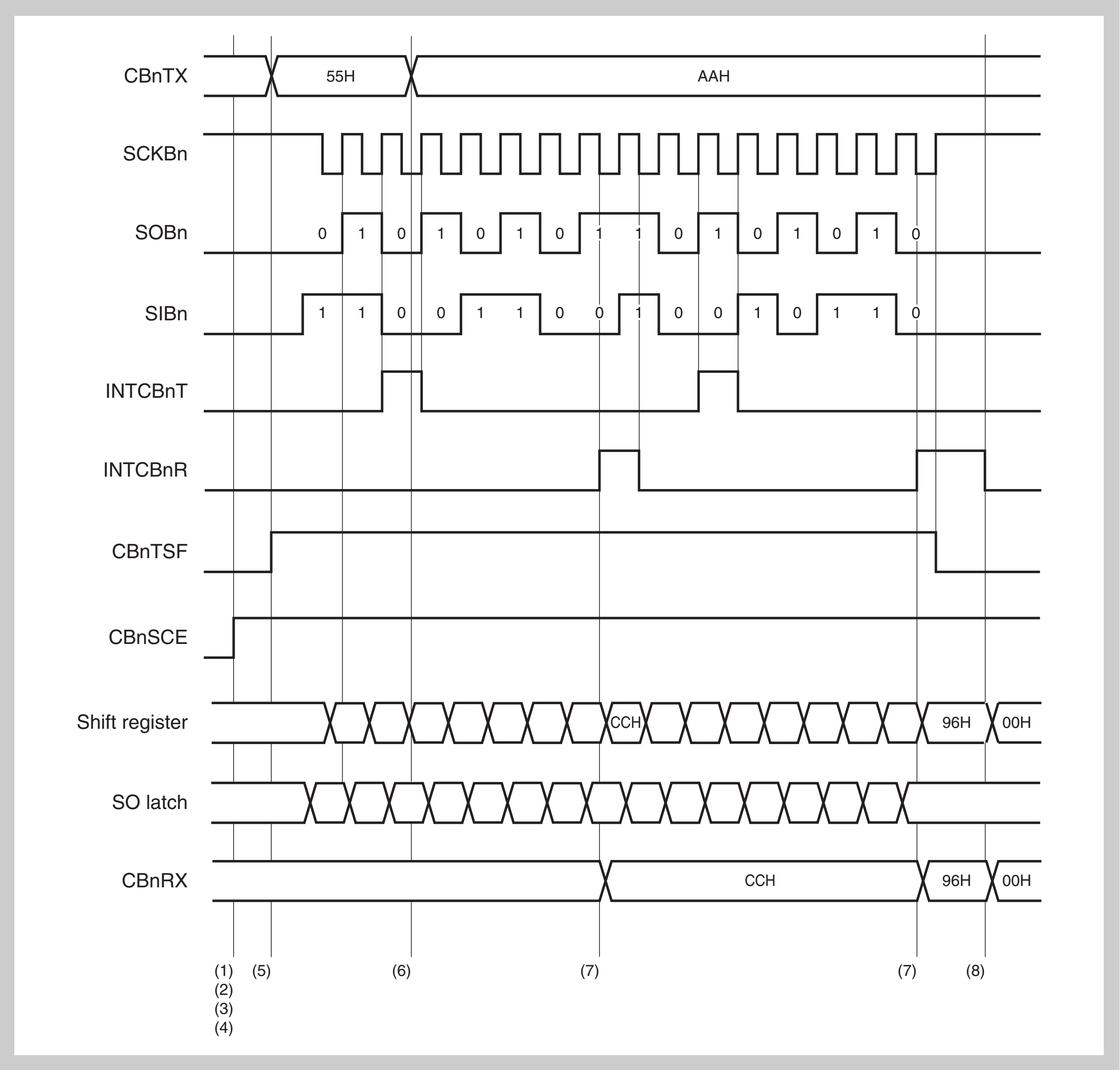
(8) Overrun error processing is performed after checking that the CBnOVE bit = 1 in the INTCBnRE interrupt servicing.

(9) Clear CBnOVE bit to 0.

(10)Check that the CBnSTR.CBnTSF bit = 0 and set the CBnPWR bit to 0 to stop the operation CSIBn (end of reception).

16.4.6 Continuous mode (slave mode, transmission/reception mode)

MSB first (CBnCTL0.CBnDIR bit = 0), communication type 2 (see 16.4 (2) CSIBn control register 1 (CBnCTL1)), transfer data length = 8 bits (CBnCTL2.CSnCL3 to CBnCTL2.CBnCL0 bits = 0, 0, 0, 0)



(1) Clear the CBnCTL0.CBnPWR bit to 0.

(2) Set the CBnCTL1 and CBnCTL2 registers to specify the transfer mode.

(3) Set the CBnTXE, CBnRXE and CBnSCE bits of the CBnCTL0 register to 1 at the same time as specifying the transfer mode using the CBnDIR bit, to set the transmission/reception enabled status.

(4) Set the CBnPWR bit to 1 to enable supply of the CSIBn operation.

(5) Write the transfer data to the CBnTX register.

(6) The transmission enable interrupt request signal (INTCBnT) is received and the transfer data is written to the CBnTX register.

(7) The reception complete interrupt request signal (INTCBnR) is output.

Read the CBnRX register.

(8) Check that the CBnSTR.CBnTSF bit = 0 and set the CBnPWR bit to 0 to stop the operation of CSIBn (end of transmission/reception).

To continue transfer, repeat steps (5) to (7) before (8).

Note In order to start the entire data transfer the CBnTX register has to be written initially, as done in step (5) above. If this step is omitted also no data will be received.

Discontinued transmission In case the CSIB is operating in continuous slave transmission mode (CBnCTL0.CBnTMS = 1, CBnCTL1.CBnCKS[2:0] = 111_B) and new data is not written to the CBnTX register the SOBn pin outputs the level of the last bit.

Table 16-4 outlines this behaviour.

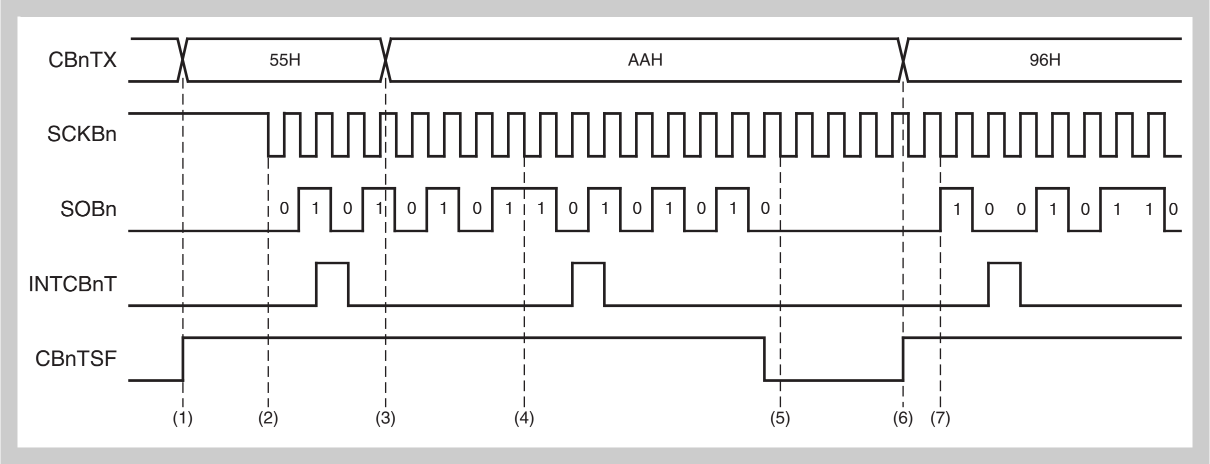


Figure 16-4 Discontinued slave transmission

The example shows the situation that two data bytes (55_H, AA_H) are transmitted correctly, but the third (96_H) fails.

(1) Data 55_H is written (by the CPU) to CBnTX.

(2) The master issues the clock SCKBn and transmission of 55_H starts.

(3) INTCBnT is generated and the next data AA_H is written to CBnTX promptly, i.e. before the first data has been transmitted completely.

(4) Transmission of the second data AA_H continues correctly and INTCBnT is generated. But this time the next data is not written to CBnTX in time.

(5) Since there is no new data available in CBnTX, but the master continuous to apply SCKBn clocks, SOBn remains at the level of the transmitted last bit.

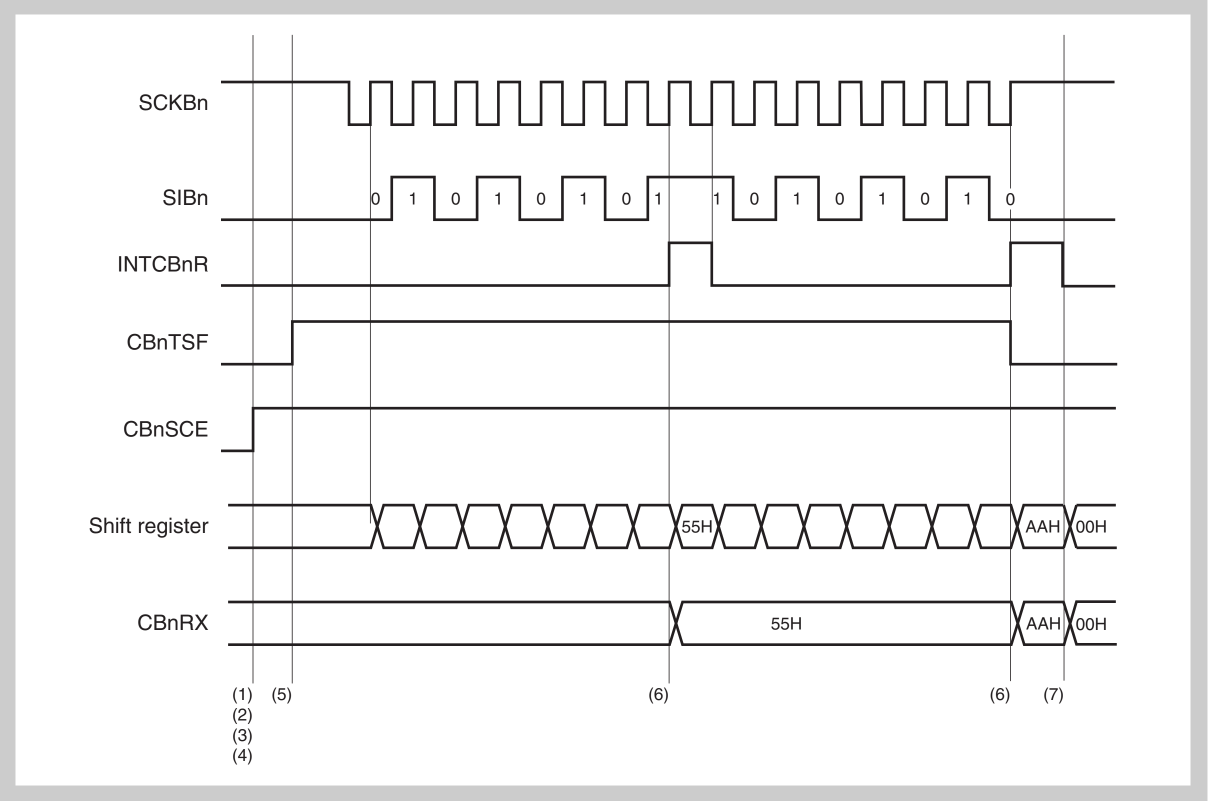
(6) New data (96_H) is written to CBnTX.

(7) With the next SCKBn cycle transmission of the new data (96_H) starts.

As a consequence the master receives a corrupted data byte from (5) onwards, which is made up of a random number of the repeated last bit of the former data and some first bits of the new data.

16.4.7 Continuous mode (slave mode, reception mode)

MSB first (CBnCTL0.CBnDIR bit = 0), communication type 1 (see 16.4 (2) CSIBn control register 1 (CBnCTL1)), transfer data length = 8 bits (CBnCTL2.CBnCL3 to CBnCTL2.CBnCL0 bits = 0, 0, 0, 0)



(1) Clear the CBnCTL0.CBnPWR bit to 0.

(2) Set the CBnCTL1 and CBnCTL2 registers to specify the transfer mode.

(3) Set the CBnCTL0.CBnRXE and CBnCTL0.CBnSCE bits to 1 at the same time as specifying the transfer mode using the CBnDIR bit, to set the reception enabled status.

(4) Set the CBnPWR bit = 1 to enable CSIBn operation.

(5) Perform a dummy read of the CBnRX register (reception start trigger).

(6) The reception complete interrupt request signal (INTCBnR) is output.
Read the CBnRX register.

(7) Check that the CBnSTR.CBnTSF bit = 0 and set the CBnPWR bit to 0 to stop the operation of CSIBn (end of reception).

To continue transfer, repeat steps (5) and (6) before (7).

16.4.8 Clock timing

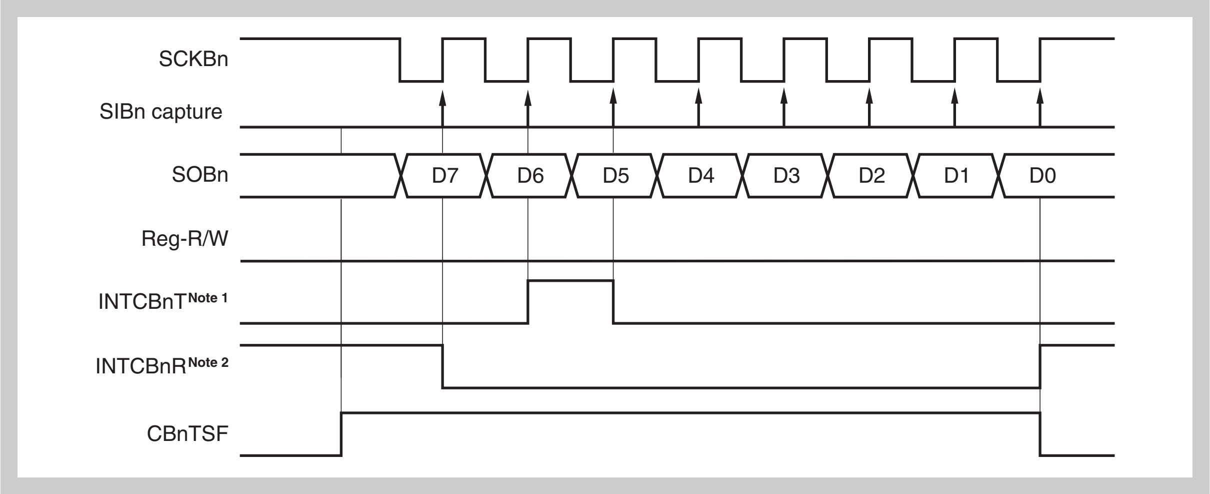


Figure 16-5 (i) Communication type 1 (CBnCKP = 0, CBnDAP = 0)

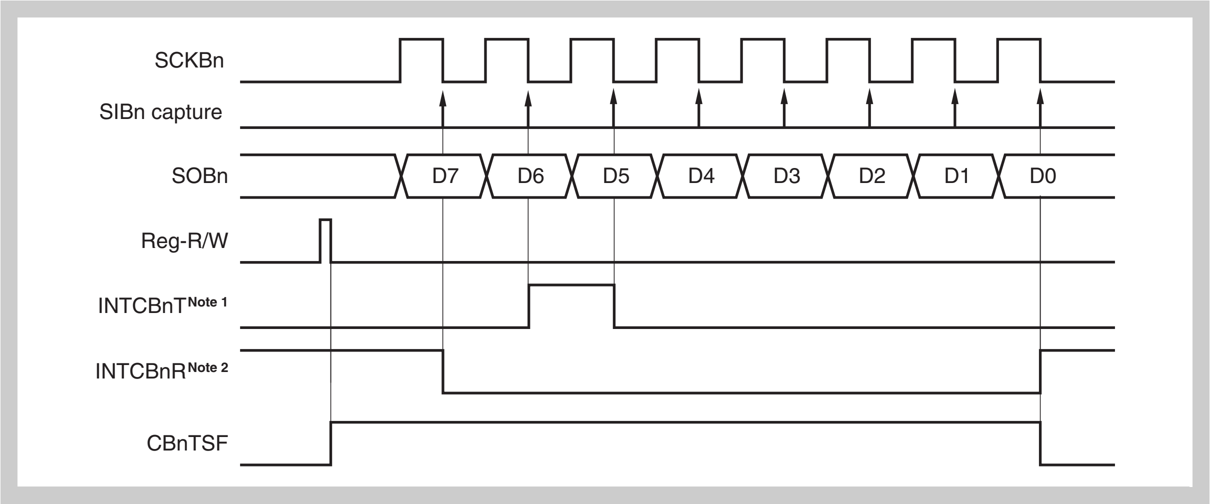


Figure 16-6 (ii) Communication type 3 (CBnCKP = 1, CBnDAP = 0)

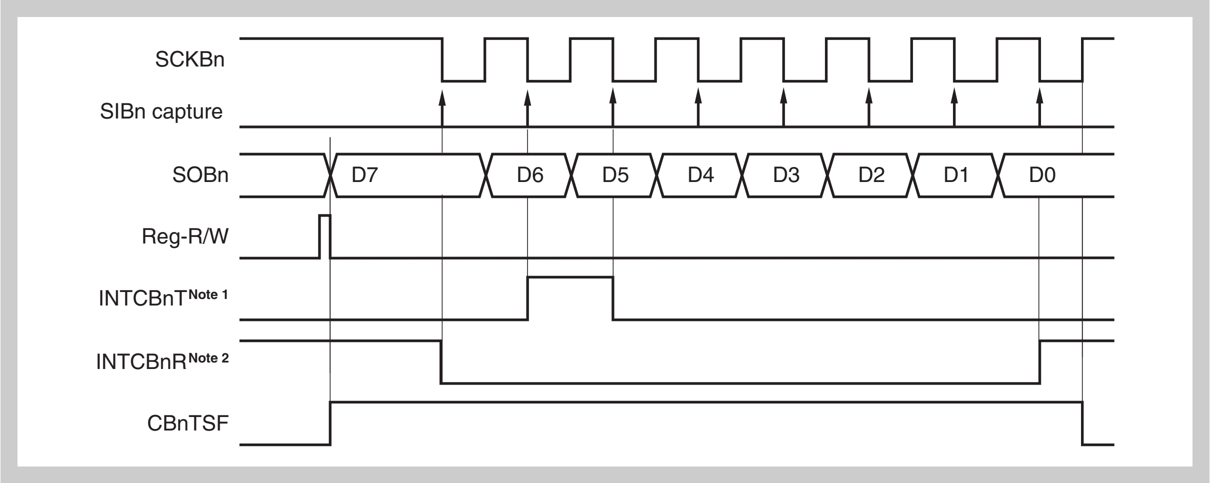


Figure 16-7 (iii) Communication type 2 (CBnCKP = 0, CBnDAP = 1)

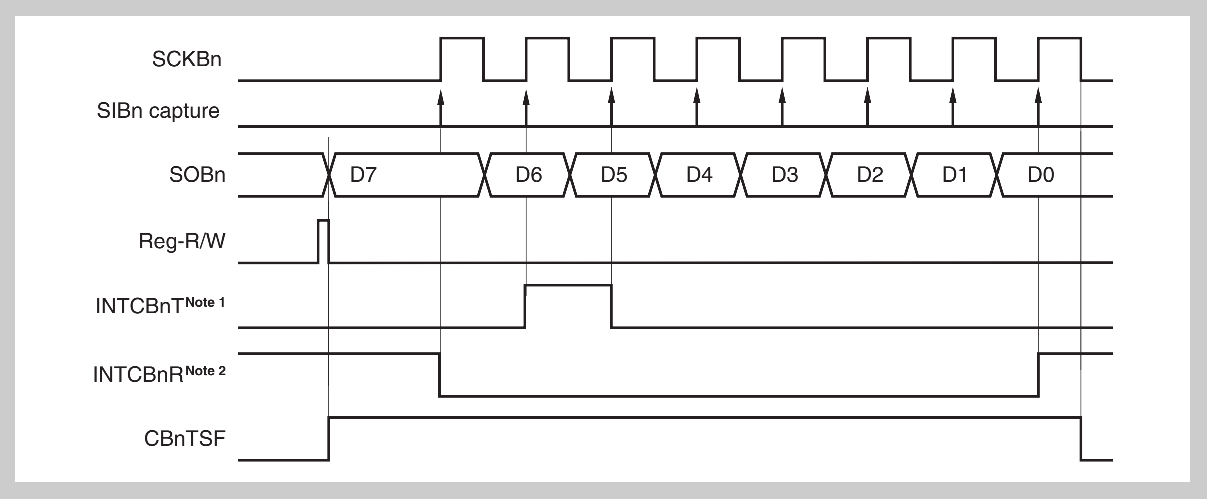


Figure 16-8 (iv) Communication type 4 (CBnCKP = 1, CBnDAP = 1)

Note

1. The INTCBnT interrupt is set when the data written to the transmit buffer is transferred to the data shift register in the continuous transmission or continuous transmission/reception modes. In the single transmission or single transmission/reception modes, the INTCBnT interrupt request signal is not generated, but the INTCBnR interrupt request signal is generated upon completion of communication.
2. The INTCBnR interrupt occurs if reception is correctly completed and receive data is ready in the CBnRX register while reception is enabled, and if an overrun error occurs. In the single mode, the INTCBnR interrupt request signal is generated even in the transmission mode, upon completion of communication.

16.5 Output Pins

(1) SCKBn pin

When CSIBn operation is disabled (CBnCTL0.CBnPWR bit = 0), the SCKBn pin output status is as follows.

CBnCKP	CBnCKS2	CBnCKS1	CBnCKS0	SCKBn pin output
0	Don’t care	Don’t care	Don’t care	Fixed to high level
1	1	1	1	High impedance
	Other than above			Fixed to low level

Note The output level of the SCKBn pin changes if any of the CBnCTL1.CBnCKP and CBnCKS2 to CBnCKS0 bits is rewritten.

(2) SOBn pin

When CSIBn operation is disabled (CBnPWR bit = 0), the SOBn pin output status is as follows.

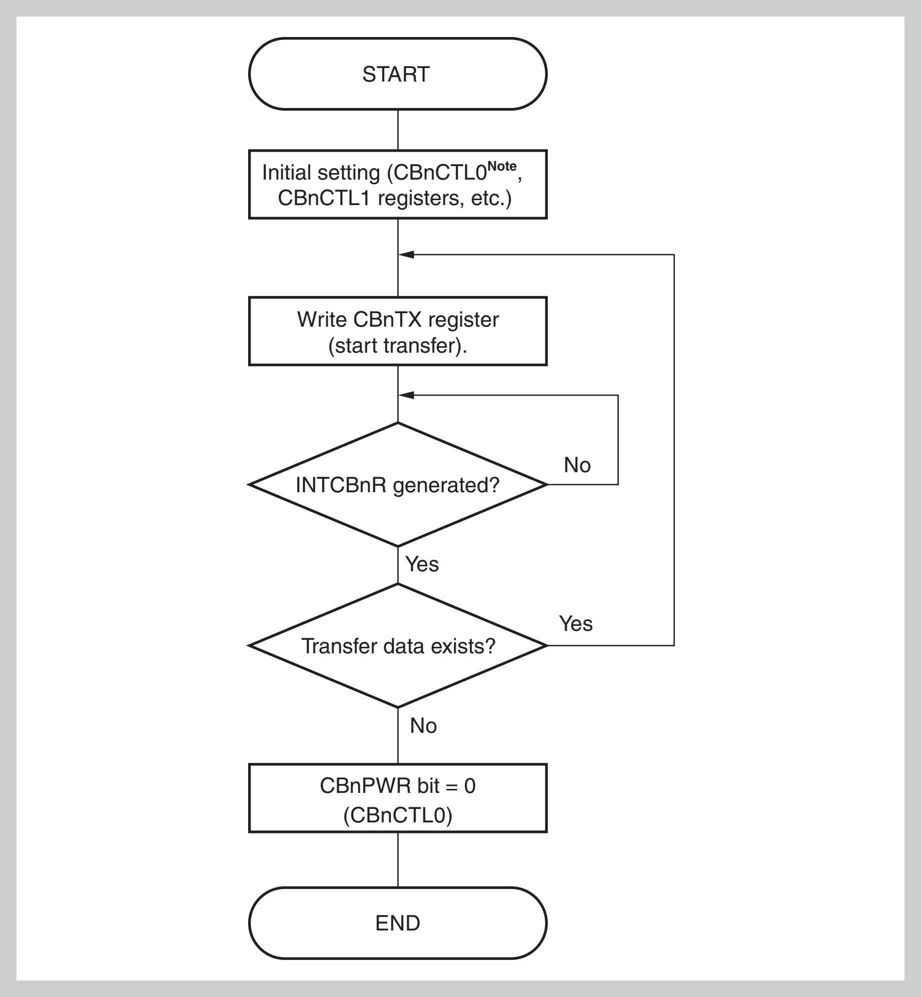
CBnTXE	CBnDAP	CBnDIR	SOBn pin output
0	×	×	Fixed to low level
1	0	×	SOBn latch value (low level)
	1	0	CBnTXn value (MSB)
		1	CBnTXn value (LSB)

Note 1. The SOBn pin output changes when any one of the CBnCTL0.CBnTXE, CBnCTL0.CBnDIR bits, and CBnCTL1.CBnDAP bit is rewritten.

2. ×: don’t care

16.6 Operation Flow

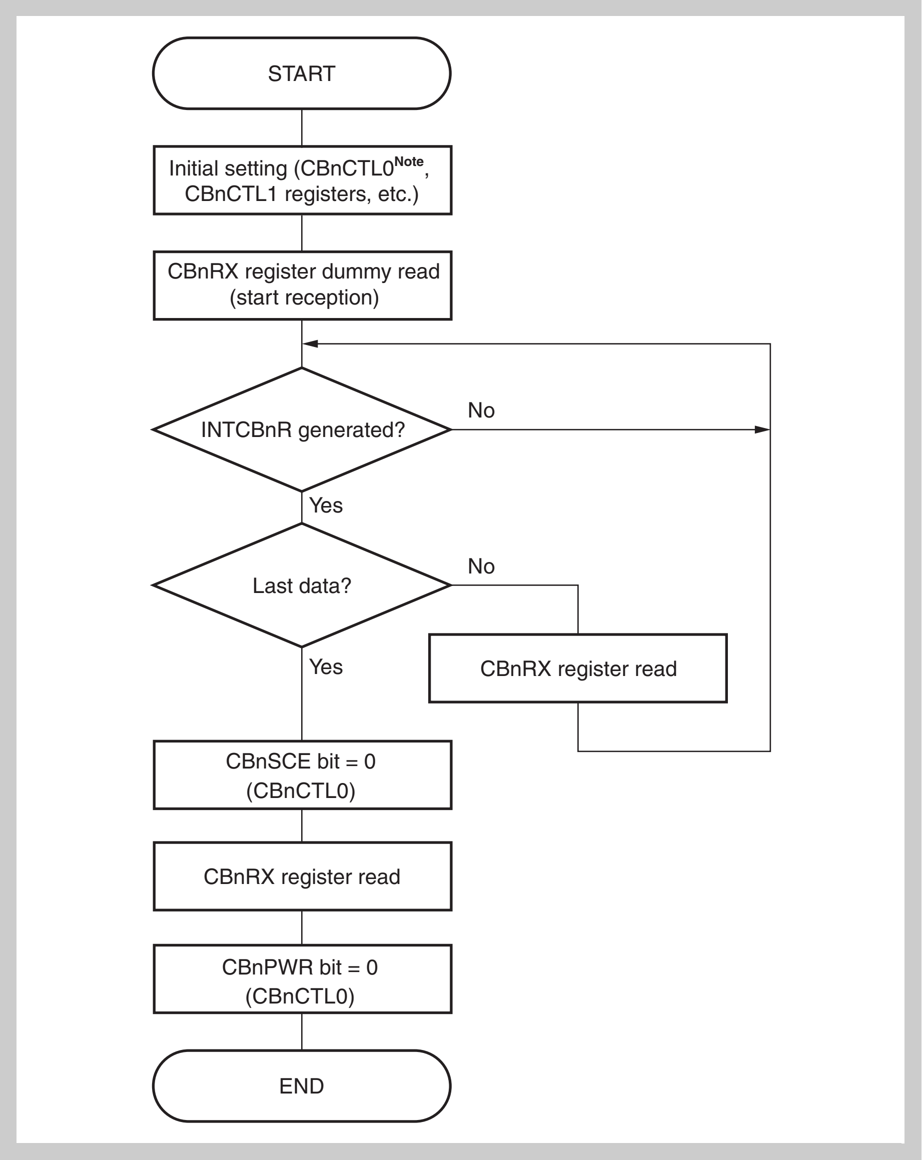
(1) Single transmission


START
Initial setting (CBnCTL0Note, CBnCTL1 registers, etc.)
Write CBnTX register (start transfer).
INTCBnR generated?
No
Yes
Transfer data exists?
Yes
No
CBnPWR bit = 0 (CBnCTL0)
END

Note Set the CBnSCE bit to 1 in the initial setting.

Caution In the slave mode, data cannot be correctly transmitted if the next transfer clock is input earlier than the CBnTX register is written.

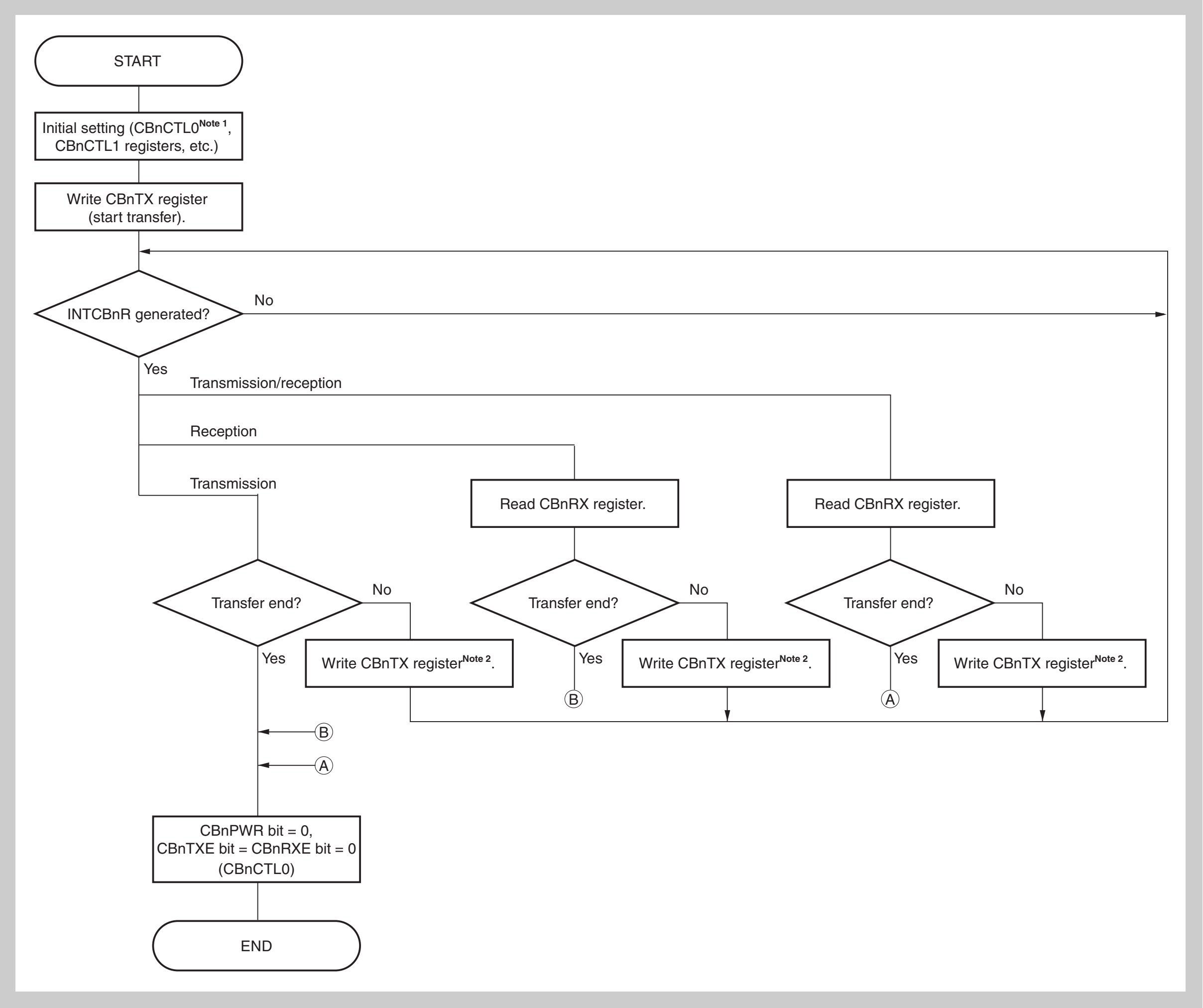
(2) Single reception



Note Set the CBnSCE bit to 1 in the initial setting.

Caution In the single mode, data cannot be correctly received if the next transfer clock is input earlier than the CBnRX register is read.

(3) Single transmission/reception

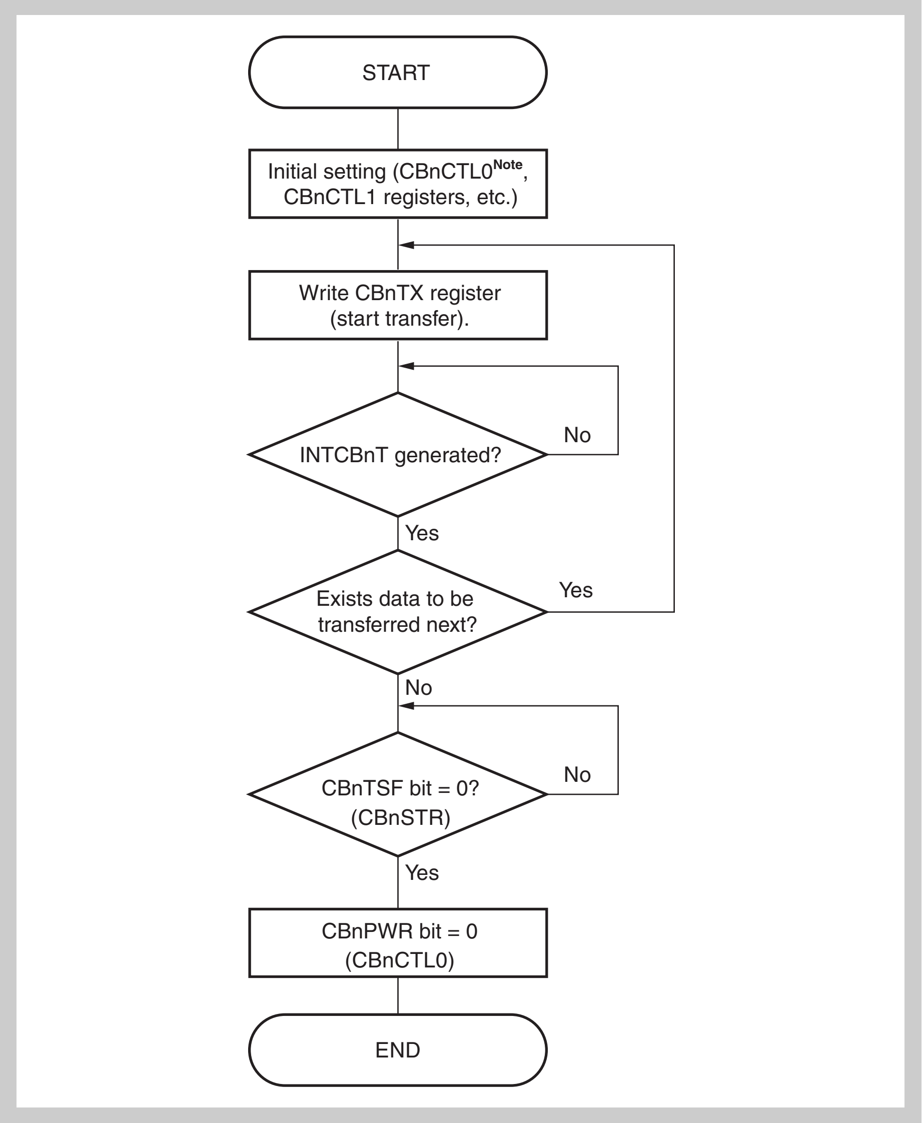

START
Initial setting (CBnCTL0Note 1, CBnCTL1 registers, etc.)
Write CBnTX register (start transfer).
INTCBnR generated?
No
Yes
Transmission/reception
Reception
Transmission
Read CBnRX register.
Read CBnRX register.
Transfer end?
No
Yes
Write CBnTX registerNote 2.
Transfer end?
No
Yes
Write CBnTX registerNote 2.
Transfer end?
No
Yes
Write CBnTX registerNote 2.
B
A
B
A
CBnPWR bit = 0,
CBnTXE bit = CBnRXE bit = 0
(CBnCTL0)
END

Note 1. Set the CBnSCE bit to 1 in the initial setting.

2. If the next transfer is reception only, dummy data is written to the CBnTX register.

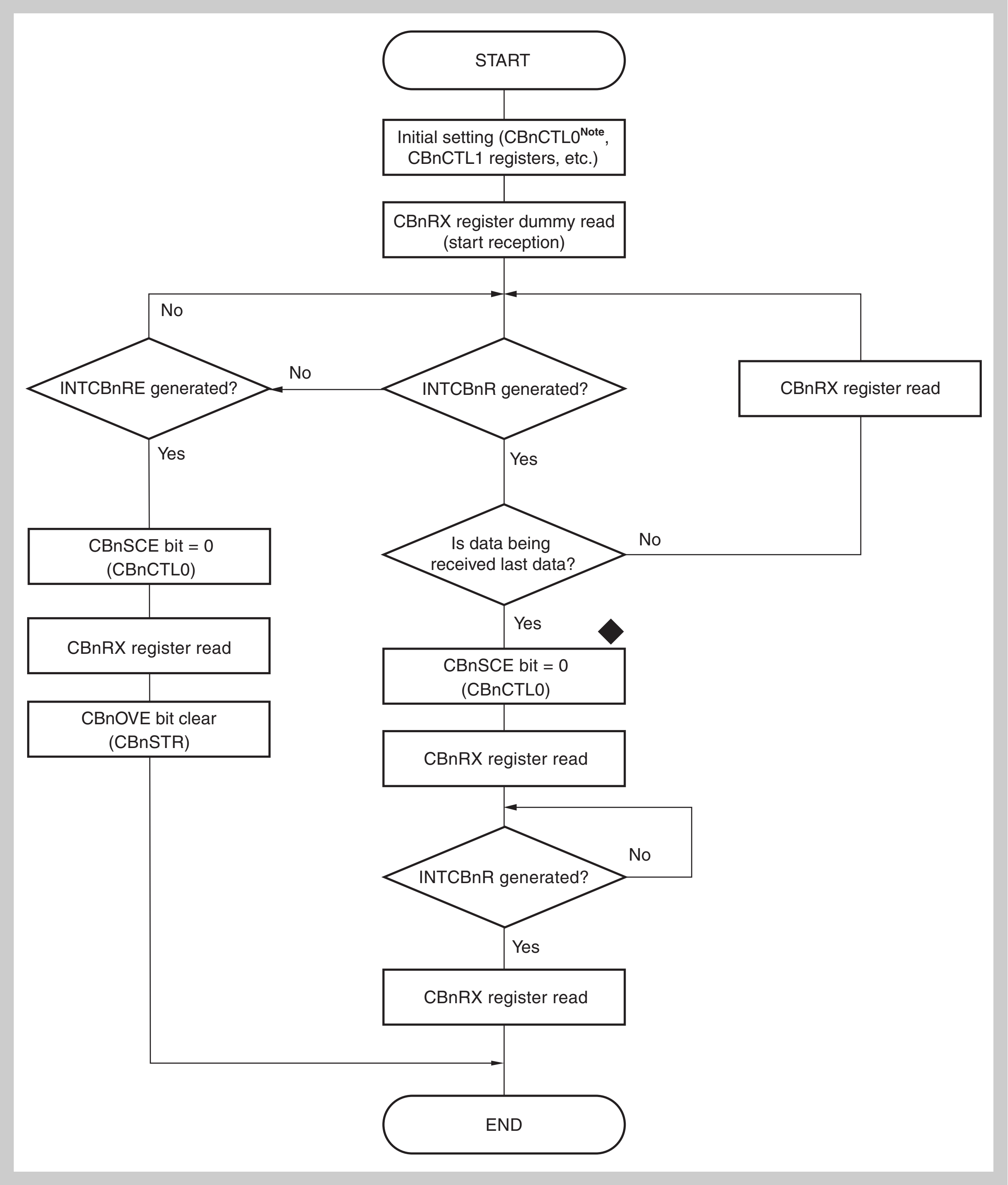
Caution Even in the single mode, the CBnSTR.CBnOVE flag is set to 1. If only transmission is used in the transmission/reception mode, therefore, programming without checking the CBnOVE flag is recommended.

(4) Continuous transmission


START
Initial setting (CBnCTL0Note, CBnCTL1 registers, etc.)
Write CBnTX register (start transfer).
INTCBnT generated?
No
Yes
Exists data to be transferred next?
Yes
No
CBnTSF bit = 0? (CBnSTR)
No
Yes
CBnPWR bit = 0 (CBnCTL0)
END

Note Set the CBnSCE bit to 1 in the initial setting.

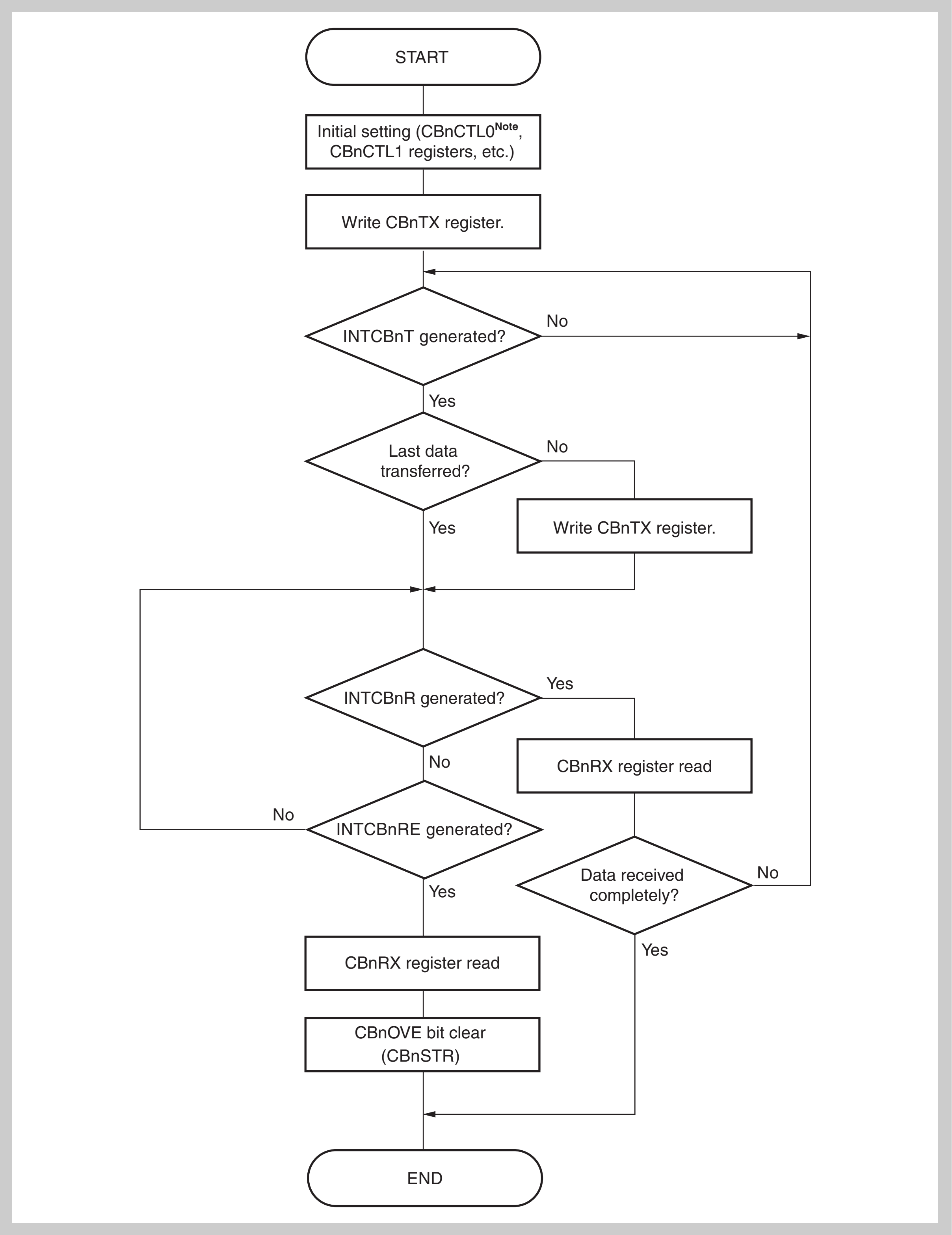
(5) Continuous reception



Note Set the CBnSCE bit to 1 in the initial setting.

Caution In the master mode, the clock is output without limit when dummy data is read from the CBnRX register. To stop the clock, execute the flow marked ◆ in the above flowchart.
In the slave mode, malfunction due to noise during communication can be prevented by executing the flow marked ◆ in the above flowchart.
Before resuming communication, set the CBnCTL0.CBnSCE bit to 1, and read dummy data from the CBnRX register.

(6) Continuous transmission/reception

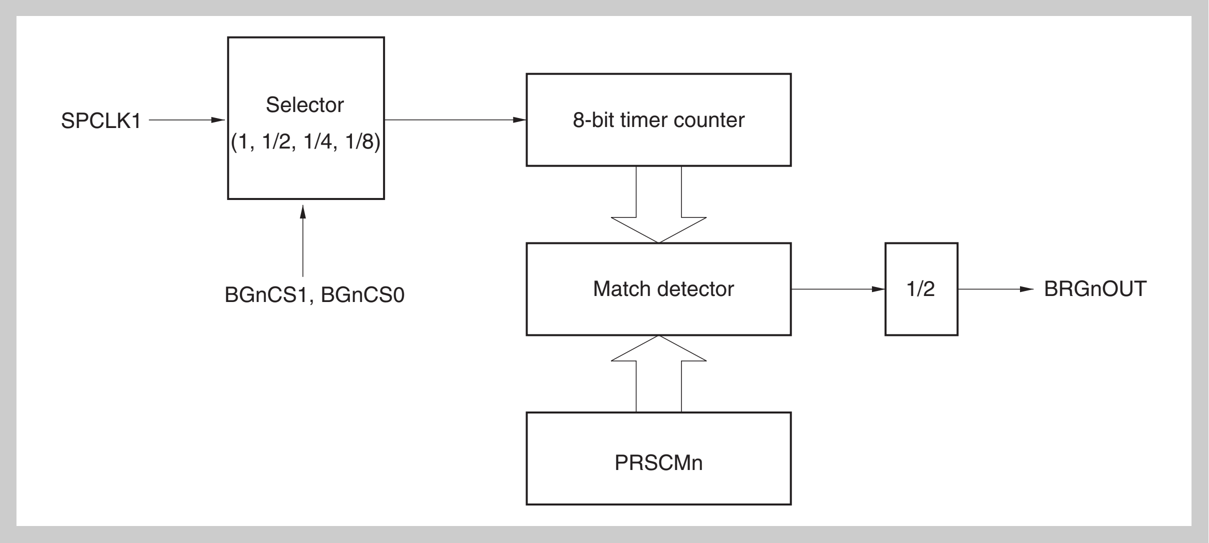


Note Set the CBnSCE bit to 1 in the initial setting.

16.7 Baud Rate Generator

16.7.1 Overview

Each CSIBn interface is equipped with a dedicated baud rate generator.



16.7.2 Baud Rate Generator registers

The Baud Rate Generators BRGn are controlled and operated by means of the following registers:

Table 16-8 BRGn registers overview

Register name	Shortcut	Address
BRGn prescaler mode register	PRSMn	<BRG_base>
BRGn prescaler compare register	PRSCMn	<BRG_base> + 1_H

Table 16-9 BRGn register base address

Timer	Base address <BRG_base>
BRG0	FFFF $FDC0_H$
BRG1	FFFF $FDE0_H$

(1) PRSMn - Prescaler mode registers

The PRSMn registers control generation of the baud rate signal for CSIB.

Access This register can be read/written in 8-bit or 1-bit units.

Address <BRG_base>

Initial Value 00_H. This register is cleared by any reset.

7	6	5	4	3	2	1	0
0	0	0	BGCEn	0	0	BGCSn1	BGCSn0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 16-10 PRSMn register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>4</td><td>BGCEn</td><td>Baud rate output
0: disabled
1: enabled</td></tr>
<tr><td>1 to 0</td><td>BGCSn[1:0]</td><td>Input clock selection
<table>
<tr><th>BGCSn1</th><th>BGCSn0</th><th>Input clock selection (f_{BGCSn})</th><th>Setting value k</th></tr>
<tr><td>0</td><td>0</td><td>f_{SPCLK1}</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>$f_{SPCLK1}/2$</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>$f_{SPCLK1}/4$</td><td>2</td></tr>
<tr><td>1</td><td>1</td><td>$f_{SPCLK1}/8$</td><td>3</td></tr>
</table></td></tr>
</table>

Caution

1. Do not rewrite the PRSMn register during operation.
2. Set the BGCSn[1:0] bits before setting the BGCEn bit to 1.

(2) PRSCMn - Prescaler compare registers

The PRSCMn registers are 8-bit compare registers.

Access This register can be read/written in 8-bit units.

Address <BRG_base> + 1_H

Initial Value 00_H. This register is cleared by any reset.

7	6	5	4	3	2	1	0
PRSCMn7	PRSCMn6	PRSCMn5	PRSCMn4	PRSCMn3	PRSCMn2	PRSCMn1	PRSCMn0
R/W							

Caution

1. Do not rewrite the PRSCMn register during operation.
2. Set the PRSCMn register before setting the PRSMn.BGCEn bit to 1.

16.7.3 Baud rate calculation

The transmission/reception clock is generated by dividing the main clock. The baud rate generated from the main clock is obtained by the following equation.

$$f_{BRGn} = \frac{f_{SPCLK1}}{2^k \times N \times 2}$$

Note		
	f_{BRGn}:	BRGn count clock
	f_{SPCLK1}:	Main clock oscillation frequency
	k:	PRSMn.BGCSn[1:0] register setting value ($0 \le k \le 3$)
	N:	PRSCMn.PRSCMn[7:0] register value if PRSCMn = 00H: N = 256.

16.8 Cautions

16.8.1 CSIBn behaviour during debugger break

The CSIBn continues to operate in debugger break-mode, provided all clocks are continuing.
The CSIBn continuous to operate during debugger break-mode

- in continuous reception/transmission mode
- in slave reception/transmission mode

Reception

When the CSIBn is in reception mode and an external device is sending data during debugger break-mode the CSIBn may produce an overflow error as it continues to receive data during break-mode.

Thus the overflow error flag UCBnSTR.CBnOVE may be set and the reception error interrupt INTCBnRE may be generated. Further all following received data are discarded.

Note that the reception error interrupt INTCBnRE will not be served during break-mode, but after resuming run-mode, provided the interrupt controller is configured accordingly.

Transmission

If the debugger's break-mode is entered while the CSIBn is transmitting data, the transmission is completed, and finally a transmission enable interrupt request INTCBnT is generated.

Note that the transmission enable interrupt INTCNnT will not be served during break-mode, but after resuming run-mode, provided the interrupt controller is configured accordingly.

16.8.2 CSIB operation stop

(1) Details - Master mode operation

When any channel of CSIB is operated with a peripheral clock source different to the clock source of the CPU, the CSIB may stop operating.
Depending on the CSIB operating configuration the CSIB behaves as described below.

- Transmit mode or transmit/receive mode:
  Any write to the related CBnTX0 register will no longer start a transmission sequence. Furthermore the related transmission interrupt request will not be generated.
- Receive mode:
  Any read from the related CBnRX0 register will no longer start a receive sequence. Furthermore the related receive interrupt request will not be generated.

The described CSIBn stuck condition can be escaped by initiating a system reset or by a sequential clear and set of the CBnCTL0.CBnPWR bit.

(2) Details - Slave mode operation

When any channel of CSIB is operated in slave mode and an external clock signal is input via the SCKBn pin while no transmission or reception sequence is in progress the CSIB may stop operating.
Depending on the CSIB operating configuration the CSIB behaves as described below.

- Transmit mode or transmit/receive mode
  Any further write to the CBnTX0 register followed by an external input clock signal input will no longer start a transmission sequence. Furthermore the related transmission interrupt request will not be generated.
- Receive mode:
  Any read from the related CBnRX0 register followed by an external input clock signal input will no longer start a receive sequence. Furthermore the related receive interrupt request will not be generated.

The described CSIBn stuck condition can be escaped by initiating a system reset or by a sequential clear and set of the CBnCTL0.CBnPWR bit.

(3) Workaround - Master mode operation

In order to avoid the CSIBn stuck condition in master mode use only the following CPU clock to CSIBn input clock combinations:

CPU clock source	SCC. SPSEL[1:0]	CKC. PERIC	BRGn clock source (SPCLK1)	CSIB clock input
4 MHz main osc	00	0	4 MHz main osc	PCLK6 .. 1, BRGn
	00	1	4 MHz main osc	PCLK6 .. 2, BRGn
	X1	0	PLL/SSCG	PCLK6 .. 1
	X1	1	PLL/SSCG	PCLK6 .. 2
PLL	01	0	PLL	BRGn
	01	1		PCLK1, BRGn
SSCG	11	X	SSCG	BRGn

(4) Workaround - Slave mode operation

In order to avoid the CSIBn stuck condition in slave mode take the following precautions.

- Transmit mode or transmit/receive mode:
  Make sure the external CSIBn clock is not input in parallel when writing to the CBnTX0 register after a transmission sequence is finished
- Receive mode:
  Make sure the external CSIBn clock is not input in parallel when reading from the CBnRX0 register after a reception sequence is finished.

Chapter 17 I^{2}C Bus (IIC)

The V850E/Dx3 - DG3 microcontrollers have following instances of the I^{2}C Bus interface IIC:

IIC	All devices
Instances	1
Names	IIC0

Throughout this chapter, the individual instances of I^{2}C Bus interface are identified by “n”, for example IICn, or IICCn for the IICn control register.

17.1 Features

The I^{2}C provides a synchronous serial interface with the following features:

- Supports Master and Slave mode
- 8-bit data transfer
- Transfer speed
  - up to 100 kbit/s (Standard Mode)
  - up to 381kbit/s (Fast Mode)
- I^{2}C root clock sources from main oscillator, PLL and SSCG
- Two wire interface
  - SCLn: serial clock
  - SDAn: serial data
- Noise filter on SCLn and SDAn input
  - spikes with a width of less than one period of IICLK are suppressed

17.2 I^2C Pin Configuration

The I^2C function requires to define the pins SCL0 and SDA0 as input and open drain output pins simultaneously. In the following the pin configuration registers are listed to be set up properly for I^2C:

- PFSR0.PFSR04 = 1/0: select input for I^2C0
- PLCDC6.PLCDC64/65 = 0: no LCD output (if applicable)
- PMCn.PMCnm = 1: alternative mode
- Input type:
  - PICCn.PICCnm = 0: non-Schmitt Trigger input for standard mode
  - PICCn.PICCnm = 1: Schmitt Trigger input for fast-speed mode
- PILCn.PILCnm = 0: CMOS1 level
- PDSCn.PDSCnm = 1: drive strength control Limit2
- PODCn.PODCnm = 1: open drain output
- PMn.PMnm = 1: input mode

It is recommended to set the output mode as the last step.

Table 17-2 shows how to set up the registers for activating I^2C0 from different pin groups.

Table 17-1 I^2C interface pins set up

I^2Cn	PFSR0 register	Pins and pin group	Register settings
I^2C0	PFSR0.PFSR04 = 0	SDA0/SCL0 via P16/P17	PMC1.PMC1[7:6] = 11_B PICC1.PICC1[7:6] = $00_B/11_B$[a] PILC1.PILC1[7:6] = 00_B PDSC1.PDSC1[7:6] = 11_B PODC1.PODC1[7:6] = 11_B PM1.PM1[7:6] = 11_B
	PFSR0.PFSR04 = 1	SCL0/SDA0 via P64/P65	PLCDC6.PLCDC6[5:4] = 00_B PMC6.PMC6[5:4] = 11_B PICC6.PICC6[5:4] = $00_B/11_B$[a] PILC6.PILC6[5:4] = 00_B PDSC6.PDSC6[5:4] = 11_B PODC6.PODC6[5:4] = 11_B PM6.PM6[5:4] = 11_B

a) PICCnm = 00_B for standard mode, PICCnm = 11_B for fast-speed mode

17.3 I^2C Pin Configuration

The I^2C function requires to define the pins SCLn and SDAn as input and open drain output pins simultaneously. In the following the pin configuration registers are listed to be set up properly for I^2C:

- PFSR0.PFSR04/5 = 1/0: select input for I^2Cn (where applicable)
- PLCDCn.PLCDCnm = 0: no LCD output (where applicable)
- PFCn.PFCnm = 1/0: select ALT1-/ALT2-OUT (where applicable)
- PMCn.PMCnm = 1: alternative mode
- Input type:
  - PICCn.PICCnm = 0: non-Schmitt Trigger input for standard mode
  - PICCn.PICCnm = 1: Schmitt Trigger input for fast-speed mode
- PILCn.PILCnm = 0: CMOS1 level
- PDSCn.PDSCnm = 1: drive strength control Limit2
- PODCn.PODCnm = 1: open drain output
- PMn.PMnm = 1: input mode

It is recommended to set the output mode in the last step.

Table 17-2 shows how to set up the registers for activating I^2C0 and I^2C1 from different pin groups.

Table 17-2 I^2C interface pins set up

I^2Cn	PFSR0 register	Pins and pin group	Register settings
I^2C0	PFSR0.PFSR04 = 0	SDA0/SCL0 via P16/P17	PMC1.PMC1[7:6] = 11_B PICC1.PICC1[7:6] = $00_B/11_B$[a] PILC1.PILC1[7:6] = 00_B PDSC1.PDSC1[7:6] = 11_B PODC1.PODC1[7:6] = 11_B PM1.PM1[7:6] = 11_B
	PFSR0.PFSR04 = 1	SCL0/SDA0 via P64/P65	PLCDC6.PLCDC6[5:4] = 00_B PFC6.PFC6[5:4] = 00_B PMC6.PMC65 = 1_B PICC6.PICC6[5:4] = $00_B/11_B$[a] PILC6.PILC6[5:4] = 00_B PDSC6.PDSC6[5:4] = 11_B PODC6.PODC6[5:4] = 11_B PM6.PM6[5:4] = 11_B
I^2C1	PFSR0.PFSR05 = 0	SDA1/SCL1 via P20/P21	PLCDC2.PLCDC2[1:0] = 00_B PMC2.PMC2[1:0] = 11_B PICC2.PICC2[1:0] = $00_B/11_B$[a] PILC2.PILC2[1:0] = 00_B PDSC2.PDSC2[1:0] = 11_B PODC2.PODC2[1:0] = 11_B PM2.PM2[1:0] = 11_B
	PFSR0.PFSR05 = 1	SDA1/SCL1 via P30/P31	PFC3.PFC30 = 1_B PMC3.PMC3[1:0] = 11_B PICC3.PICC3[1:0] = $00_B/11_B$[a] PILC3.PILC3[1:0] = 00_B PDSC3.PDSC3[1:0] = 11_B PODC3.PODC3[1:0] = 11_B PM3.PM3[1:0] = 11_B

a) PICCnm = 00_B for standard mode, PICCnm = 11_B for fast-speed mode

17.4 I^{2}C Pin Configuration

The I^{2}C function requires to define the pins SCLn and SDAn as input and open drain output pins simultaneously. In the following the pin configuration registers are listed to be set up properly for I^{2}C:

- PFSR0.PFSR04/5 = 1/0: select input for I^{2}Cn (where applicable)
- PLCDCn.PLCDCnm = 0: no LCD output (where applicable)
- PFCn.PFCnm = 1/0: select ALT1-/ALT2-OUT (where applicable)
- PMCn.PMCnm = 1: alternative mode
- Input type:
  - PICCn.PICCnm = 0: non-Schmitt Trigger input for standard mode
  - PICCn.PICCnm = 1: Schmitt Trigger input for fast-speed mode
- PILCn.PILCnm = 0: CMOS1 level
- PDSCn.PDSCnm = 1: drive strength control Limit2
- PODCn.PODCnm = 1: open drain output
- PMn.PMnm = 1: input mode

It is recommended to set the output mode as the last step.

Table 17-2 shows how to set up the registers for activating I^{2}C0 and I^{2}C1 from different pin groups.

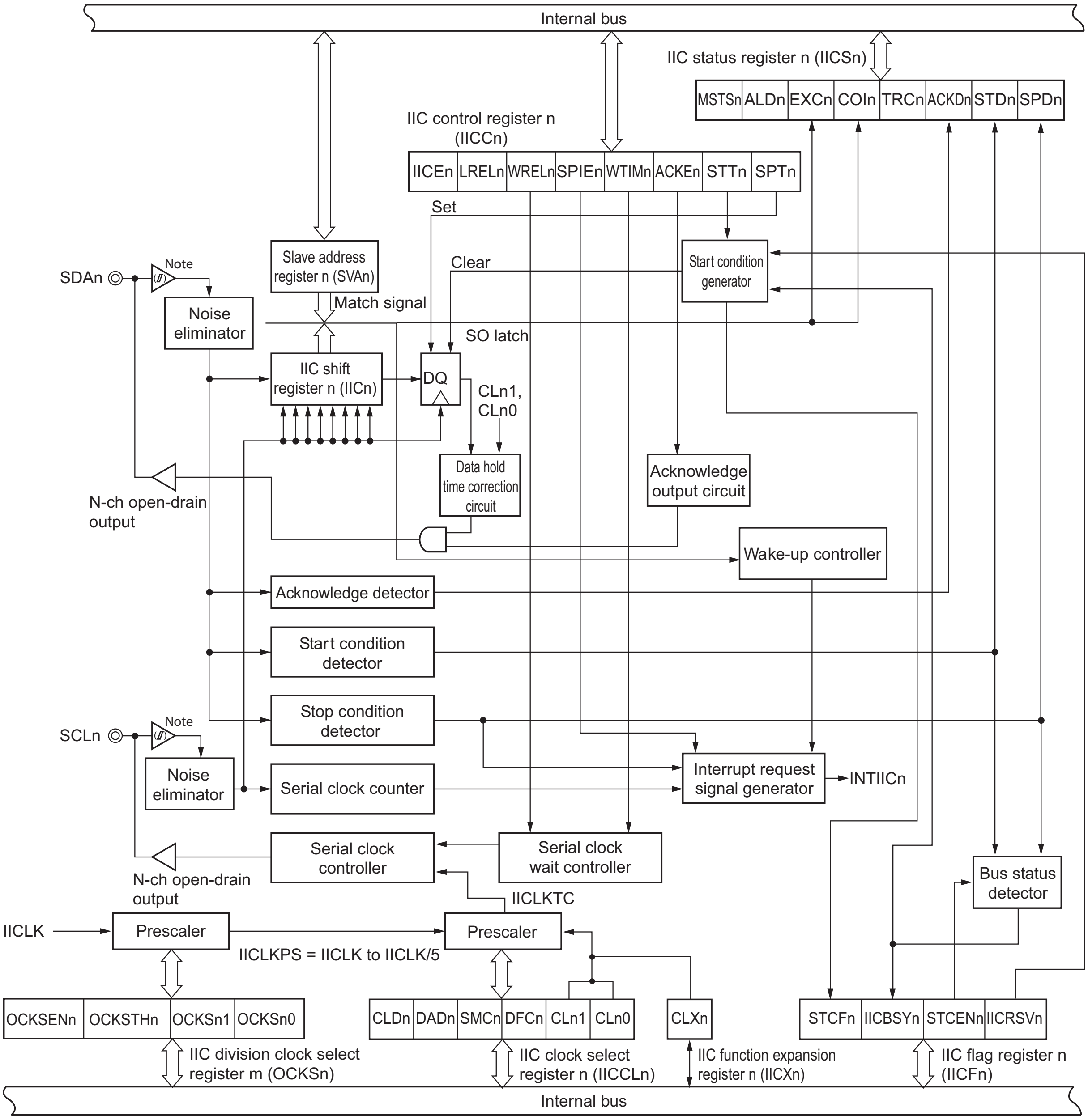
Table 17-3 I^2C interface pins set up

I^2Cn	PFSR0 register	Pins and pin group	Register settings
I^2C0	PFSR0.PFSR04 = 0	SDA0/SCL0 via P16/P17	PMC1.PMC1[7:6] = 11_B PICC1.PICC1[7:6] = $00_B/11_B$[a] PILC1.PILC1[7:6] = 00_B PDSC1.PDSC1[7:6] = 11_B PODC1.PODC1[7:6] = 11_B PM1.PM1[7:6] = 11_B
	PFSR0.PFSR04 = 1	SCL0/SDA0 via P64/P65	PLCDC6.PLCDC6[5:4] = 00_B PFC6.PFC6[5:4] = 00_B PMC6.PMC6[5:4] = 11_B PICC6.PICC6[5:4] = $00_B/11_B$[a] PILC6.PILC6[5:4] = 00_B PDSC6.PDSC6[5:4] = 11_B PODC6.PODC6[5:4] = 11_B PM6.PM6[5:4] = 11_B
I^2C1	PFSR0.PFSR05 = 0	SDA1/SCL1 via P20/P21	PLCDC2.PLCDC2[1:0] = 00_B PFC2.PFC2[1:0] = 00_B PMC2.PMC2[1:0] = 11_B PICC2.PICC2[1:0] = $00_B/11_B$[a] PILC2.PILC2[1:0] = 00_B PDSC2.PDSC2[1:0] = 11_B PODC2.PODC2[1:0] = 11_B PM2.PM2[1:0] = 11_B
	PFSR0.PFSR05 = 1	SDA1/SCL1 via P30/P31	PFC3.PFC30 = 1_B PMC3.PMC3[1:0] = 11_B PICC3.PICC3[1:0] = $00_B/11_B$[a] PILC3.PILC3[1:0] = 00_B PDSC3.PDSC3[1:0] = 11_B PODC3.PODC3[1:0] = 11_B PM3.PM3[1:0] = 11_B

[a] PICCnm = 00_B for standard mode, PICCnm = 11_B for fast-speed mode

17.5 Configuration

The block diagram of the I^{2}C0n is shown below.



Note: Schmitt Trigger input buffer for fast-speed mode, non Schmitt Trigger for standard mode

Figure 17-1 Block diagram of I^{2}C0n

A serial bus configuration example is shown below.

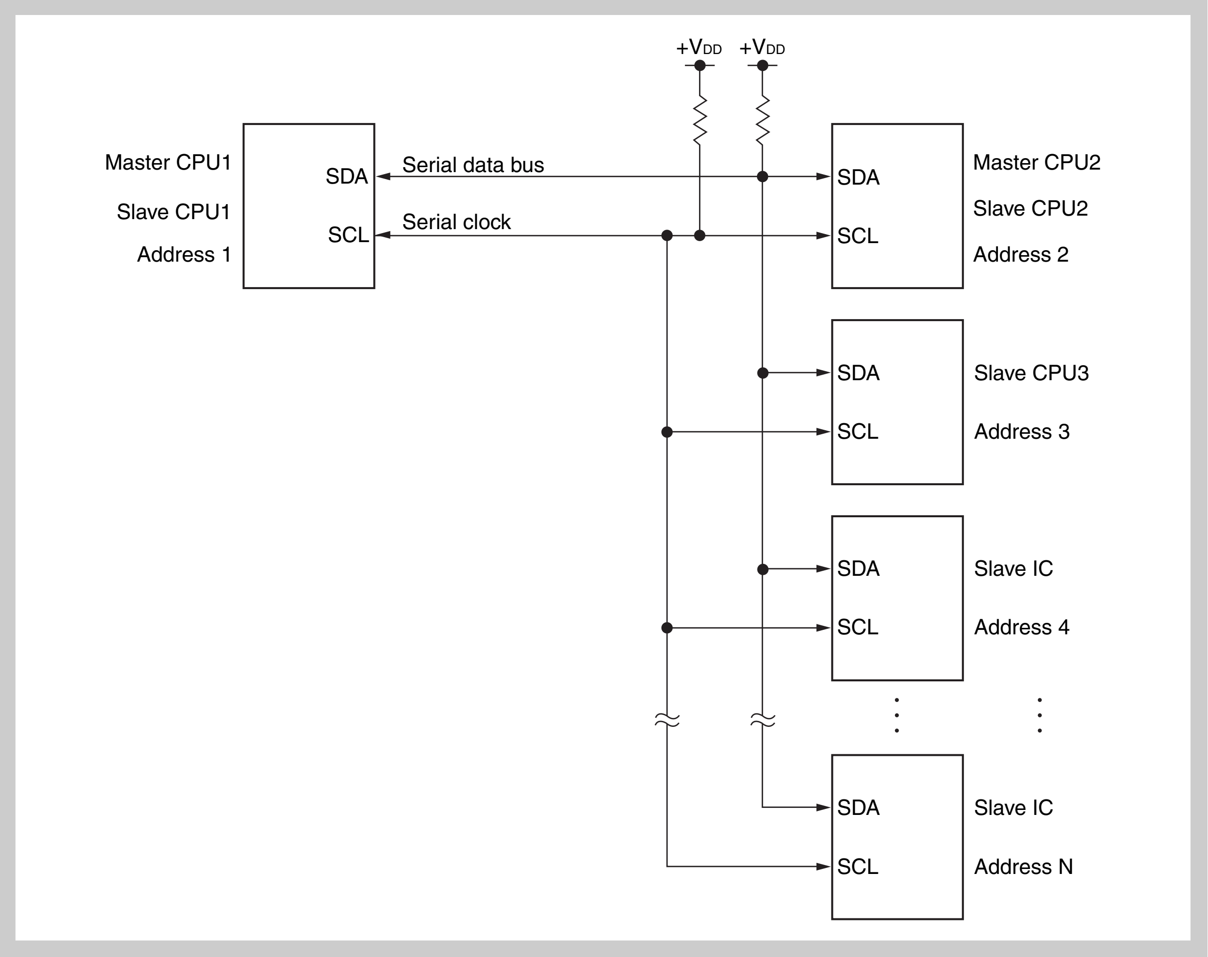


Figure 17-2 Serial bus configuration example using I^{2}C bus

I^{2}C0n includes the following hardware.

(1) IIC shift register n (IICn)

The IICn register converts 8-bit serial data into 8-bit parallel data and vice versa, and can be used for both transmission and reception.

Write and read operations to the IICn register are used to control the actual transmit and receive operations.

(2) Slave address register n (SVAn)

The SVAn register sets local addresses when in slave mode.

(3) SO latch

The SO latch is used to retain the output level of the SDAn pin.

(4) Wakeup controller

This circuit generates an interrupt request when the address received by this register matches the address value set to the SVAn register or when an extension code is received.

(5) Prescaler

This selects the sampling clock to be used.

(6) Serial clock counter

This counter counts the serial clocks that are output and the serial clocks that are input during transmit/receive operations and is used to verify that 8-bit data was transmitted or received.

(7) Interrupt request signal generator

This circuit controls the generation of interrupt request signals (INTIICn).

An I^2C interrupt is generated following either of two triggers:

- Falling edge of eighth or ninth clock of the serial clock (set by IICCn.WTIMn bit)
- Interrupt occurrence due to stop condition detection (set by IICCn.SPIEn bit)

(8) Serial clock controller

In master mode, this circuit generates the clock output via the SCLn pin from the sampling clock.

(9) Serial clock wait controller

This circuit controls the wait timing.

(10) $\overline{\text{ACK}}$ output circuit, stop condition detector, start condition detector, and $\overline{\text{ACK}}$ detector

These circuits are used to output and detect various control signals.

(11) Data hold time correction circuit

This circuit generates the hold time for data corresponding to the falling edge of the SCLn pin.

(12) Start condition generator

A start condition is issued when the IICCn.STTn bit is set.

However, in the communication reservation disabled status (IICFn.IICRSVn = 1), this request is ignored and the IICFn.STCFn bit is set if the bus is not released (IICFn.IICBSYn = 1).

(13) Bus status detector

Whether the bus is released or not is ascertained by detecting a start condition and stop condition.

However, the bus status cannot be detected immediately after operation, so set the bus status detector to the initial status by using the IICFn.STCENn bit.

17.6 IIC Registers

The I^2C serial interfaces IICn are controlled and operated by means of the following registers:

Table 17-4 IICn registers overview

Register name	Shortcut	Address
IICn shift register	IICn	<base>
IICn control register	IICCn	<base> + 2_H
IICn slave address register	SVAn	<base> + 3_H
IICn clock select register	IICCLn	<base> + 4_H
IICn function expansion register	IICXn	<base> + 5_H
IICn status register	IICSn	<base> + 6_H
IICn flag register	IICF0n	<base> + A_H
IICn division clock select registers	OCKSn	<base> + 20_H

Table 17-5 IICn register base address

IICn	Base address <base>
IIC0	FFFF $FD80_H$

Note IICn control register
The IICCn registers enable/stop I2C operations, set the wait timing and other I2C operations.
These registers can be read or written in 8-bit or 1-bit units. However, set the SPIEn, WTIMn, and ACKEn bits when the IICn.IICEn bit is 0 or during the wait period. When setting the IICn.IICEn bit from “0” to “1”, these bits can also be set at the same time.

(1) IICCn - IICn control registers

The IICCn registers enable/stop I^2Cn operations, set the wait timing, and set other I^2Cn operations.

Access This register can be read/written in 8-bit or 1-bit units.

Address <base> + 2_H

Initial Value 00_H. This register is cleared by any reset.

7	6	5	4	3	2	1	0
IICEn	LRELn	WRELn	SPIEn	WTIMn	ACKEn	STTn	SPTn
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

IICEn	Specification of I^2Cn operation enable/disable
0	Operation stopped. IICSn register reset [Note]. Internal operation stopped.
1	Operation enabled.
Condition for clearing (IICEn = 0)	**Condition for setting (IICEn = 1)**
• Cleared by instruction • After reset	• Set by instruction

Note The IICS register, IICFn.STCFn and IICFn.IICBSYn bits, and IICCLn.CLDn and IICCLn.DADn bits are reset.

LRELn	Exit from communications
0	Normal operation
1	This exits from the current communication operation and sets stand-by mode. This setting is automatically cleared after being executed. Its uses include cases in which a locally irrelevant extension code has been received. The SCLn and SDAn lines are set to high impedance. The STTn and SPTn bits and the MSTSn, EXCn, COIn, TRCn, ACKDn, and STDn bits of the IICSn register are cleared.
The stand-by mode following exit from communications remains in effect until the following communication entry conditions are met. • After a stop condition is detected, restart is in master mode. • An address match occurs or an extension code is received after the start condition.	
Condition for clearing (LRELn = 0)	**Condition for setting (LRELn = 1)**
• Automatically cleared after execution • After reset	• Set by instruction

WRELn	Wait cancellation control
0	Wait not cancelled
1	Wait cancelled. This setting is automatically cleared after wait is cancelled.
Condition for clearing (WRELn = 0)	**Condition for setting (WRELn = 1)**
• Automatically cleared after execution • After reset	• Set by instruction

SPIEn	Enable/disable generation of interrupt request when stop condition is detected
0	Disabled
1	Enabled
Condition for clearing (SPIEn = 0)	**Condition for setting (SPIEn = 1)**
• Cleared by instruction • After reset	• Set by instruction

WTIMn	Control of wait and interrupt request generation
0	Interrupt request is generated at the eighth clock's falling edge. Master mode: After output of eight clocks, clock output is set to low level and wait is set. Slave mode: After input of eight clocks, the clock is set to low level and wait is set for the master device. In order to generate the ninth clock on SCLn the wait status must be cancelled by writing to IICn or setting IICCn.WRELn = 1. Consequently the ninth clock will be delayed until the wait status is cancelled.
1	Interrupt request is generated at the ninth clock's falling edge. Master mode: After output of nine clocks, clock output is set to low level and wait is set. Slave mode: After input of nine clocks, the clock is set to low level and wait is set for the master device.
During address transfer, an interrupt occurs at the falling edge of the ninth clock regardless of this bit setting. This bit setting becomes valid when the address transfer is completed. In master mode, a wait is inserted at the falling edge of the ninth clock during address transfer. For a slave device that has received a local address, a wait is inserted at the falling edge of the ninth clock after an $\overline{\text{ACK}}$ signal is issued. When the slave device has received an extension code, however, a wait is inserted at the falling edge of the eighth clock.	
Condition for clearing (WTIMn = 0)	**Condition for setting (WTIMn = 1)**
• Cleared by instruction • After reset	• Set by instruction

ACKEn	Acknowledgement control
0	Acknowledgment disabled.
1	Acknowledgment enabled. During the ninth clock period, the SDAn line is set to low level. However, $\overline{\text{ACK}}$ is invalid in other than extension mode during address transfers.
Condition for clearing (ACKEn = 0)	**Condition for setting (ACKEn = 1)**
• Cleared by instruction • After reset	• Set by instruction

<table>
<tr><th>STTn</th><th colspan="2">Start condition trigger</th></tr>
<tr><td>0</td><td colspan="2">Start condition is not generated.</td></tr>
<tr><td>1</td><td colspan="2">When bus is released (in STOP mode):
A start condition is generated (for starting as master). The SDAn line is changed from high level to low level and then the start condition is generated. Next, after the rated amount of time has elapsed, the SCLn line is changed to low level.
During communication with a third party:
If the communication reservation function is enabled (IICFn.IICRSVn = 0)
• This trigger functions as a start condition reserve flag. When set, it releases the bus and then automatically generates a start condition.
If the communication reservation function is disabled (IICRSVn = 1)
• The IICFn.STCFn bit is set. This trigger does not generate a start condition.
In the wait state (when master device):
A restart condition is generated after the wait is released.</td></tr>
<tr><td colspan="3">Cautions concerning set timing
For master reception: Cannot be set during transfer. Can be set only when the ACKEn bit has been set to 0 and the slave has been notified of final reception.
For master transmission: A start condition cannot be generated normally during the $\overline{\text{ACK}}$ period. Set during the wait period.
For slave: Even when the communication reservation function is disabled (IICRSVn bit = 1), the communication reservation status is entered.</td></tr>
<tr><th colspan="2">Condition for clearing (STTn = 0) [Note]</th><th>Condition for setting (STTn = 1)</th></tr>
<tr><td colspan="2">• Cleared by loss in arbitration
• Cleared after start condition is generated by master device
• When the LRELn = 1 (communication save)
• When the IICEn= 0 (operation stop)
• After reset</td><td>• Set by instruction</td></tr>
</table>

Note The STTn bit is 0 if it is read immediately after data setting.

<table>
<tr><th>SPTn</th><th>Stop condition trigger</th></tr>
<tr><td>0</td><td>Stop condition is not generated.</td></tr>
<tr><td>1</td><td>Stop condition is generated (termination of master device’s transfer).
After the SDAn line goes to low level, either set the SCLn line to high level or wait until it goes to high level. Next, after the rated amount of time has elapsed, the SDAn line is changed from low level to high level and a stop condition is generated.</td></tr>
<tr><td colspan="2">Cautions concerning set timing
For master reception: Cannot be set during transfer.
Can be set only when the ACKEn bit has been set to 0 and during the wait period after the slave has been notified of final reception.
For master transmission: A stop condition cannot be generated normally during the $\overline{\text{ACK}}$ period. Set during the wait period.
• SPTn cannot be set at the same time as the STTn bit.
• The SPTn bit can be set only when in master mode Note 1.
• When the WTIMn bit has been set to 0 and the SPTn bit is set during the wait period that follows output of eight clocks, note that a stop condition will be generated during the high-level period of the ninth clock. When the ninth clock must be output to apply the $\overline{\text{ACK}}$ on the bus by the receiving device, proceed as follows:
- Change IICCn.WTIMn from 0 to 1 in order to receive an additional interrupt after the ninth clock.
- Cancel the wait state by IICCn.WRELn = 1 or by writing to the IICn register.
- Upon the interrupt after the ninth clock require to set the stop condition by IICCn.STPn = 1. By this the wait status will be cancelled and the stop condition will be generated on the bus.</td></tr>
<tr><th>Condition for clearing (SPTn = 0) Note 2</th><th>Condition for setting (SPTn = 1)</th></tr>
<tr><td>• Cleared by loss in arbitration
• Automatically cleared after stop condition is detected
• When the LRELn = 1 (communication save)
• When the IICEn = 0 (operation stop)
• After reset</td><td>• Set by instruction</td></tr>
</table>

Note

1. Set the SPTn bit only in master mode. However, when communication reservation is enabled (IICFn.IICRSVn = 0), the SPTn bit must be set and a stop condition generated before the first stop condition is detected following the switch to the operation enabled status. For details, see *“Cautions“ on page 532*.
2. Clearing the IICEn bit to 0 invalidates the signals of this flag.
3. The SPTn bit is 0 if it is read immediately after data setting.

Caution When the TRCn = 1, the WRELn bit is set during the ninth clock and wait is canceled, after which the TRCn bit is cleared and the SDAn line is set to high impedance.

(2) IICSn - IICn status registers

The IICSn registers indicate the status of the I^2Cn bus.

Access This register can only be read in 8-bit or 1-bit units.

Address <base> + 6_H

Initial Value 00_H. This register is cleared by any reset.

7	6	5	4	3	2	1	0
MSTSn	ALDn	EXCn	COIn	TRCn	ACKDn	STDn	SPDn
R	R	R	R	R	R	R	R

MSTSn	Master device status
0	Slave device status or communication stand-by status
1	Master device communication status
Condition for clearing (MSTSn = 0)	**Condition for setting (MSTSn = 1)**
• When a stop condition is detected • When the ALDn = 1 (arbitration loss) • Cleared by LRELn = 1 (communication save) • When the IICEn bit changes from 1 to 0 (operation stop) • After reset	• When a start condition is generated

ALDn	Arbitration loss detection
0	This status means either that there was no arbitration or that the arbitration result was a “win”.
1	This status indicates the arbitration result was a “loss”. The MSTSn bit is cleared.
Condition for clearing (ALDn = 0)	**Condition for setting (ALDn = 1)**
• Automatically cleared after the IICSn register is read[Note] • When the IICEn bit changes from 1 to 0 (operation stop) • After reset	• When the arbitration result is a “loss”.

Note Any bit manipulation instruction targetting this register also clears this bit.

EXCn	Detection of extension code reception
0	Extension code was not received.
1	Extension code was received.
Condition for clearing (EXCn = 0)	**Condition for setting (EXCn = 1)**
• When a start condition is detected • When a stop condition is detected • Cleared by LRELn = 1 (communication save) • When the IICEn bit changes from 1 to 0 (operation stop) • After reset	• When the higher four bits of the received address data are either “0000” or “1111” (set at the rising edge of the eighth clock).

COIn	Matching address detection
0	Addresses do not match.
1	Addresses match.
Condition for clearing (COIn = 0)	**Condition for setting (COIn = 1)**
• When a start condition is detected • When a stop condition is detected • Cleared by LRELn bit = 1 (communication save) • When the IICEn bit changes from 1 to (operation stop) • After reset	• When the received address matches the local address (SVAn register) (set at the rising edge of the eighth clock).

TRCn	Transmit/receive status detection
0	Receive status (other than transmit status). The SDAn line is set to high impedance.
1	Transmit status. The value in the SO latch is enabled for output to the SDAn line (valid starting at the falling edge of the first byte's ninth clock).
Condition for clearing (TRCn = 0)	**Condition for setting (TRCn = 1)**
• When a stop condition is detected • Cleared by LRELn = 1 (communication save) • When the IICEn bit changes from 1 to 0 (operation stop) • Cleared by WRELn = 1[Note] • When the ALDn bit changes from 0 to 1 (arbitration loss) • After reset Master • When "1" is output to the first byte's LSB (transfer direction specification bit) Slave • When a start condition is detected When not used for communication	Master • When a start condition is generated Slave • When "1" is input by the first byte's LSB (transfer direction specification bit)

ACKDn	ACK detection
0	ACK was not detected.
1	ACK was detected.
Condition for clearing (ACKDn = 0)	**Condition for setting (ACKD = 1)**
• When a stop condition is detected • At the rising edge of the next byte's first clock • Cleared by LRELn = 1 (communication save) • When the IICEn bit changes from 1 to 0 (operation stop) • After reset	• After the SDAn bit is set to low level at the rising edge of the SCLn pin's ninth clock

Note The TRCn bit is cleared and SDAn line becomes high impedance when the WRELn bit is set and the wait state is canceled at the ninth clock by TRCn = 1.

STDn	Start condition detection	
0	Start condition was not detected.	
1	Start condition was detected. This indicates that the address transfer period is in effect	
Condition for clearing (STDn = 0)		**Condition for setting (STDn = 1)**
• When a stop condition is detected • At the rising edge of the next byte's first clock following address transfer • Cleared by LRELn = 1 (communication save) • When the IICEn bit changes from 1 to 0 (operation stop) • After reset		• When a start condition is detected

SPDn	Stop condition detection	
0	Stop condition was not detected.	
1	Stop condition was detected. The master device's communication is terminated and the bus is released.	
Condition for clearing (SPDn = 0)		**Condition for setting (SPDn = 1)**
• At the rising edge of the address transfer byte's first clock following setting of this bit and detection of a start condition • When the IICEn bit changes from 1 to 0 (operation stop) • After reset		• When a stop condition is detected

(3) IICFn - IICn flag registers

The registers set the I^2Cn operation mode and indicate the I^2C bus status.

Access This register can be read/written in 8-bit or 1-bit units.
STCFn and IICBSYn bits are read-only.

Address <base> + A_H

Initial Value 00_H. This register is cleared by any reset.

7	6	5	4	3	2	1	0
STCFn	IICBSYn	0	0	0	0	STCENn	IICRSVn
R	R	R/W	R/W	R/W	R/W	R/W	R/W

IICRSVn enables/disables the communication reservation function.

The initial value of the IICBSYn bit is set by using the STCENn bit (see *"Cautions" on page 532*).

The IICRSVn and STCENn bits can be written only when operation of I^2Cn is disabled (IICCn.IICEn = 0). After operation is enabled, IICFn can be read.

STCFn	STTn clear	
0	Start condition issued	
1	Start condition cannot be issued, STTn bit cleared	
Condition for clearing (STCFn = 0)		**Condition for setting (STCFn = 1)**
• Cleared by IICCn.STTn = 1 • After reset		• When start condition is not issued and STTn flag is cleared during communication reservation is disabled (IICRSVn = 1).

IICBSYn	I^2Cn bus status	
0	Bus released status	
1	Bus communication status	
Condition for clearing (IICBSYn = 0)		**Condition for setting (IICBSYn = 1)**
• When stop condition is detected • After reset		• When start condition is detected • By setting the IICCn.IICEn bit when the STCENn = 0

STCENn	Initial start enable trigger	
0	Start conditions cannot be generated until a stop condition is detected following operation enable (IICEn bit = 1).	
1	Start conditions can be generated even if a stop condition is not detected following operation enable (IICEn = 1).	
Condition for clearing (STCENn = 0)		**Condition for setting (STCENn = 1)**
• When start condition is detected • After reset		• Setting by instruction

IICRSVn	Communication reservation function disable bit
0	Communication reservation enabled
1	Communication reservation disabled

Condition for clearing (IICRSVn = 0)	Condition for setting (IICRSVn = 1)
• Clearing by instruction • After reset	• Setting by instruction

Note Bits 6 and 7 are read-only bits.

Caution

1. Write the STCENn bit only when operation is stopped (IICEn = 0).
2. When the STCENn = 1, the bus released status (IICBSYn = 0) is recognized regardless of the actual bus status immediately after the I^2Cn bus operation is enabled. Therefore, to issue the first start condition (STTn = 1), it is necessary to confirm that the bus has been released, so as to not disturb other communications.
3. Write the IICRSVn bit only when operation is stopped (IICEn = 0).

(4) IICCLn - IICn clock select registers

The IICCLn registers set the transfer clock for the I^{2}Cn bus.

The SMCn, CLn1, and CLn0 bits are set by the combination of the IICXn.CLXn bit and the OCKSTHn, OCKSn[1:0] bits of the OCKSn register (see *“Transfer rate setting“ on page 493*).

Access This register can be read/written in 8-bit or 1-bit units.
CLDn and DADn bits are read-only.

Address <base> + 4_H

Initial Value 00_H. This register is cleared by any reset.

7	6	5	4	3	2	1	0
0	0	CLDn	DADn	SMCn	DFCn	CLn1	CLn0
R/W	R/W	R	R	R/W	R/W	R/W	R/W

CLDn	Detection of SCLn pin level (valid only when IICCn.IICEn = 1)
0	The SCLn pin was detected at low level.
1	The SCLn pin was detected at high level.
Condition for clearing (CLDn = 0)	**Condition for setting (CLDn = 1)**
• When the SCLn pin is at low level • When the IICEn = 0 (operation stop) • After reset	• When the SCLn pin is at high level

DADn	Detection of SDAn pin level (valid only when IICEn = 1)
0	The SDAn pin was detected at low level.
1	The SDAn pin was detected at high level.
Condition for clearing (DADn = 0)	**Condition for setting (DAD0n = 1)**
• When the SDAn pin is at low level • When the IICEn = 0 (operation stop) • After reset	• When the SDAn pin is at high level

SMCn	Operation mode switching
0	Operation in standard mode.
1	Operation in fast-speed mode.

DFCn	Digital filter operation control
0	Digital filter off.
1	Digital filter on.
The digital filter can be used only in fast-speed mode. In fast-speed mode, the transfer clock does not vary regardless of the DFCn bit setting (on/off). The digital filter is used to eliminate noise in fast-speed mode.	

(5) IICXn - IICn function expansion registers

The IICXn registers provide additional transfer data rate configuration in fast-speed mode. Setting of the IICXn.CLXn is performed in combination with the IICCLn.SMCn, IICCLn.CLn[1:0], OCKSn.OCKSTHn and OCKSn.OCKSn[1:0] (refer to *“Transfer rate setting“ on page 493*)

Access This register can be read/written in 8-bit or 1-bit units.

Address <base> + 5_H

Initial Value 00_H. This register is cleared by any reset.

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	CLXn
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

(6) OCKSn - IICn division clock select registers

The OCKSn registers control the I^{2}Cn division clock.

Access This register can be read/written in 8-bit or 1-bit units.

Address <base> + 20_H

Initial Value 00_H. This register is cleared by any reset.

7	6	5	4	3	2	1	0
0	0	0	OCKSENn	OCKSTHn	0	OCKSn1	OCKSn0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

OCKSENn	Operation setting of I^{2}C clock
0	Disable I^{2}C division clock operation
1	Enable I^{2}C division clock operation

OCKSTHn	OCKSn1	OCKSn0	Output clock IICLKPS
0	0	0	IICLK/2
0	0	1	IICLK/3
0	1	0	IICLK/4
0	1	1	IICLK/5
1	0	0	IICLK
Other than above			Setting prohibited

(7) Transfer rate setting

The nominal transfer rate of the I^{2}C interface is determined by the following means:

- the root clock source for the I^{2}C clock IICLK can be chosen as
  - main oscillator (4 MHz): ICC.IICSEL1 = 0
  - 32 MHz clock from the PLL: ICC.IICSEL1 = 1
- a prescaler in the Clock Generator divides the chosen clock source by
  - 1.0: ICC.IICPS[2:0]=000_B
  - 3.5: ICC.IICPS[2:0]=101_B
  - 4.5: ICC.IICPS[2:0]=111_B

The output clock IICLK supplies the IIC interface.

- The IICLK can be divided by 1 to 5, configured by OCKSn.OCKSTHn and OCKSn.OCKSn[1:0] (refer to *“OCKSn - IICn division clock select registers“ on page 492*). The output clock of this divider is named IICLKPS.
- IICLK respectively IICLKPS is passed through another configurable divider that finally outputs the clock for the serial transfer IICLKTC. This divider is configured by IICCLn.CL[1:0] and IICXn.CLX0 according to the following table:

Note The clock chosen as the input clock, that means IICLK or IICLKPS, must lie in the range of 1 MHz to 10 MHz.

IICXn.CLXn	IICCLn.SMCn	IICCLn.CLn1	IICCLn.CLn0	Input clock	Transfer clock	Mode
0	0	0	0	$f_{IICLKPS}$	$f_{IICLKPS}/44$	standard
		0	1	$f_{IICLKPS}$	$f_{IICLKPS}/86$	standard
		1	0	f_{IICLK}	$f_{IICLK}/86$	standard
		1	1	$f_{IICLKPS}$	$f_{IICLKPS}/66$	standard
	1	0	0	$f_{IICLKPS}$	$f_{IICLKPS}/24$	fast-speed
		0	1	$f_{IICLKPS}$	$f_{IICLKPS}/24$	fast-speed
		1	0	f_{IICLK}	$f_{IICLK}/24$	fast-speed
		1	1	$f_{IICLKPS}$	$f_{IICLKPS}/18$	fast-speed
1	0	x	x	n.a.	n.a.	n.a.
	1	0	0	$f_{IICLKPS}$	$f_{IICLKPS}/12$	fast-speed
		0	1	$f_{IICLKPS}$	$f_{IICLKPS}/12$	fast-speed
		1	0	f_{IICLK}	$f_{IICLK}/12$	fast-speed

Following table lists set-ups for some useful I^{2}C transfer clocks.

Clock Generator		Prescaler		I^{2}C module set-up				Transfer clock [KHz]
IICPS [2:0]	divisor	OCKSn	divisor	IICCLn. SMCn	IICXn. CLXn	IICCLn. CLn[1:0]	divisor	
101_B	3.5	1 0000_B = 10_H	2	1	1	00_B	12	380,95
101_B	3.5	1 0010_B = 12_H	4	1	0	00_B	24	95,24
111_B	4.5	1 0010_B = 12_H	4	1	0	11_B	18	98,77

Note The calculations in the above table assumes that IICLK is 32 MHz (IIC.IICSEL1 = 1)

Clock Stretching Heavy capacitive load and the dimension of the external pull-up resistor on the I^{2}C bus pins may yield extended rise times of the rising edge of SCLn and SDAn. Since the controller senses the level of the I^{2}C bus signals it recognizes such situation and takes countermeasures by stretching the clock SCLn in order to ensure proper high level time t_{SCLH} of SCLn.

After the microcontroller releases the (open-drain) SCLn pin it waits until the SCLn level exceeds the valid high level threshold V_{thH}. Then it does not pull SCLn to low level before the nominal high level time t_{SCLH_nom} has elapsed.

This mechanism is the same used, when a slow I^{2}C slave device is pulling down SCLn to low level to initiate a wait state.

Figure 17-3 shows an example.

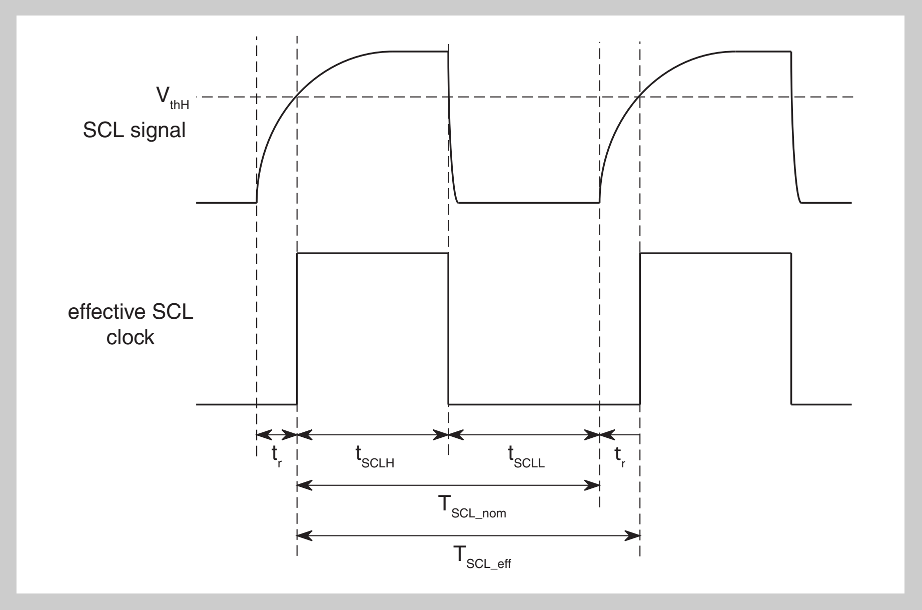


Figure 17-3 **Clock Stretching of SCLn**

The effective clock frequency appearing at the SCLn pin calculates to

$f_{SCL\_eff} = 1 / (T_{SCL\_nom} + t_r)$

With a nominal frequency of f_{SCL_nom} = 395 KHz (T_{SCL_nom} = 2.532 µs and a rise time of t_r = 135 ns the effective frequency is f_{eff} = 375 KHz.

(8) IICn - IICn shift registers

The IICn registers are used for serial transmission/reception (shift operations) synchronized with the serial clock.

A wait state is released by writing the IICn register during the wait period, and data transfer is started.

Access This register can be read/written in 8-bit units.
Data should not be written to the IICn register during a data transfer.

Address <base>

Initial Value 00_H. This register is cleared by any reset.

7	6	5	4	3	2	1	0
Input/output data							
R/W							

(9) SVAn - IICn slave address registers

The SVAn registers hold the I^2C bus’s slave addresses.

Access This register can be read/written in 8-bit units.
Bit 0 should be fixed to 0.

Address <base> + 3_H

Initial Value 00_H. This register is cleared by any reset.

7	6	5	4	3	2	1	0
Slave address							
R/W							

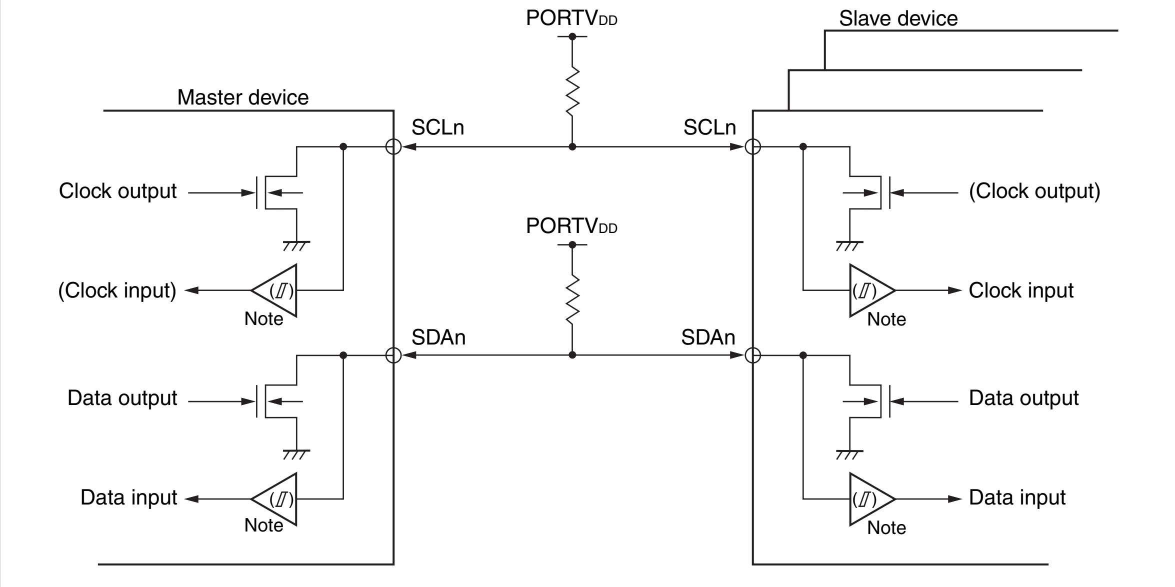
17.7 I^{2}C Bus Pin Functions

The serial clock pin (SCLn) and serial data bus pin (SDAn) are configured as follows.

- SCLn
  This pin is used for serial clock input and output.

  This pin is an N-ch open-drain output for both master and slave devices. Input is Schmitt Trigger input for fast-speed mode respectively non Schmitt Trigger for standard mode.
- SDAn
  This pin is used for serial data input and output.
  This pin is an N-ch open-drain output for both master and slave devices. Input is Schmitt Trigger input for fast-speed mode respectively non Schmitt Trigger for standard mode.

Since outputs from the serial clock line and the serial data bus line are N-ch open-drain outputs, an external pull-up resistor is required.



Note: Schmitt Trigger input buffer for fast-speed mode, non Schmitt Trigger for standard mode

Figure 17-4 Pin configuration diagram

17.8 I^{2}C Bus Definitions and Control Methods

The following section describes the I^{2}C bus's serial data communication format and the signals used by the I^{2}C bus. The transfer timing for the "start condition”, “data”, and “stop condition” output via the I^{2}C bus’s serial data bus is shown below.

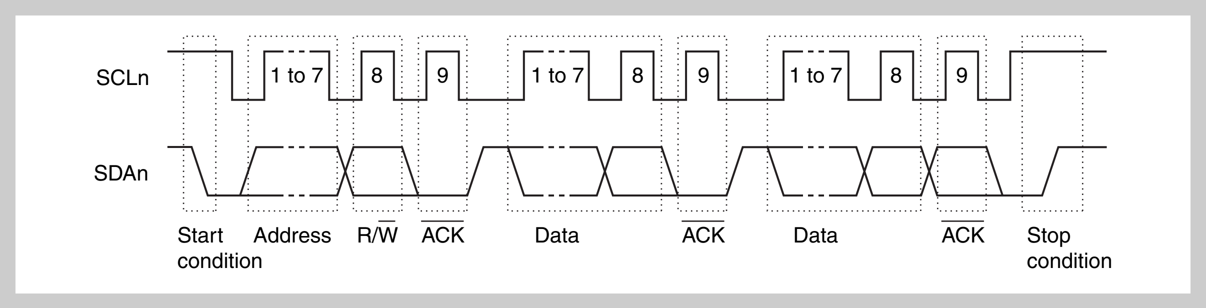


Figure 17-5 I^{2}C bus serial data transfer timing with stop termination

Instead of a stop condition the master may also send a repeated start condition, when it wishes to keep hold of the bus and to start a new data transfer.

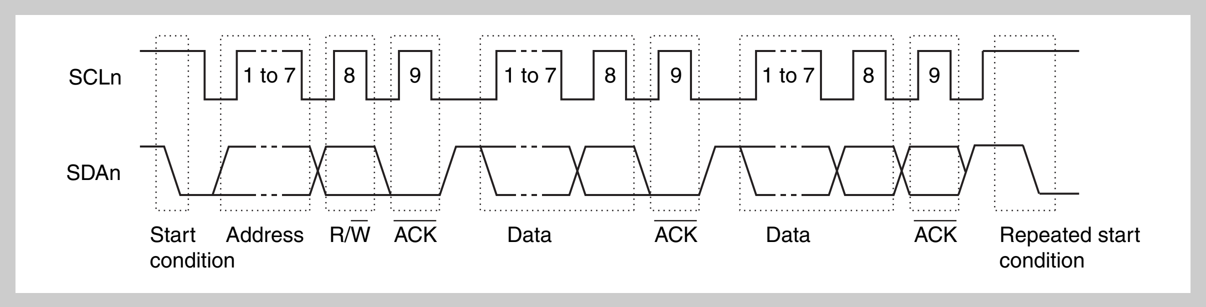


Figure 17-6 I^{2}C bus serial data transfer timing with restart

The master device outputs the start condition, slave address, and stop condition.

The acknowledge signal (ACK) can be output by either the master or slave device (normally, it is output by the device that receives 8-bit data).

The serial clock (SCLn) is continuously output by the master device. However, in the slave device, the SCLn pin’s low-level period can be extended and a wait can be inserted.

17.8.1 Start condition

A start condition is met when the SCLn pin is high level and the SDAn pin changes from high level to low level. The start condition for the SCLn and SDAn pins is a signal that the master device outputs to the slave device when starting a serial transfer. The slave device can detect the start condition.

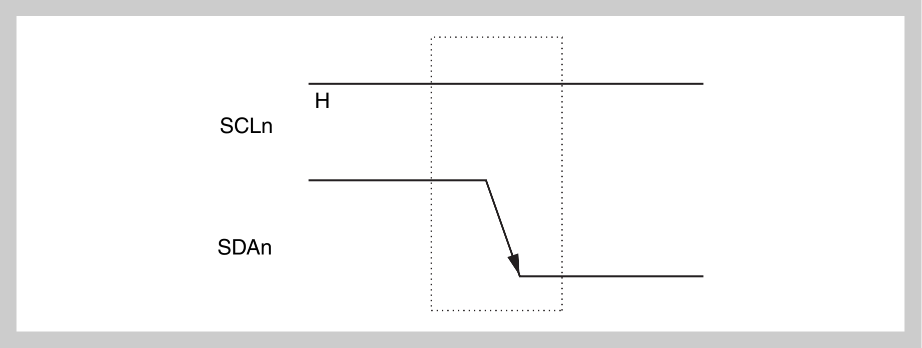


Figure 17-7 Start condition

A start condition is output when the IICCn.STTn bit is set (1) after a stop condition has been detected (IICSn.SPDn bit = 1). When a start condition is detected, the IICSn.STDn bit is set (1). By setting IICCN.STTn=1 the master device will also cancel its own wait status.

17.8.2 Addresses

The 7 bits of data that follow the start condition are defined as an address.

An address is a 7-bit data segment that is output in order to select one of the slave devices that are connected to the master device via the bus lines. Therefore, each slave device connected via the bus lines must have a unique address.

The slave devices include hardware that detects the start condition and checks whether or not the 7-bit address data matches the data values stored in the SVAn register. If the address data matches the values of the SVAn register, the slave device is selected and communicates with the master device until the master device transmits a start condition or stop condition.

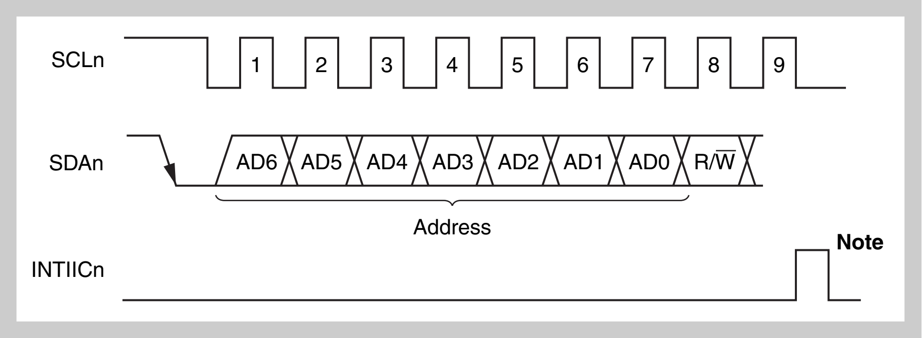


Figure 17-8 Address

Note The interrupt request signal (INTIICn) is generated if a local address or extension code is received during slave device operation.

The slave address and the eighth bit, which specifies the transfer direction as described in *"Transfer direction specification" on page 499*, are written together to IIC shift register n (IICn) and then output. Received addresses are written to the IICn register.

The slave address is assigned to the higher 7 bits of the IICn register.

17.8.3 Transfer direction specification

In addition to the 7-bit address data, the master device sends 1 bit that specifies the transfer direction. When this transfer direction specification bit has a value of 0, it indicates that the master device is transmitting data to a slave device. When the transfer direction specification bit has a value of 1, it indicates that the master device is receiving data from a slave device.

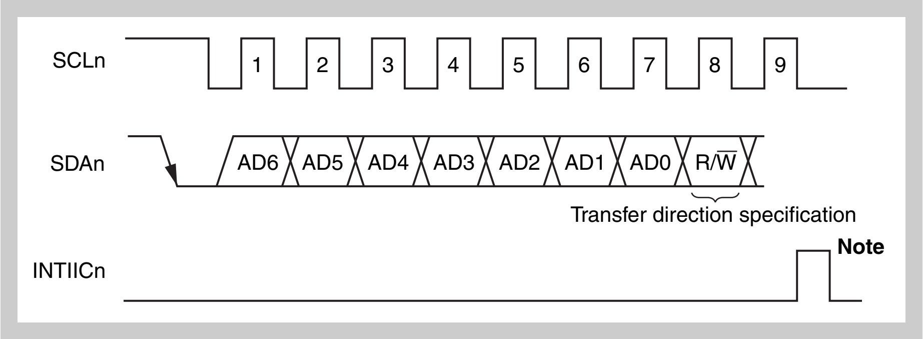


Figure 17-9 Transfer direction specification

Note The INTIICn signal is generated if a local address or extension code is received during slave device operation.

17.8.4 Acknowledge signal ($\overline{\text{ACK}}$)

The acknowledge signal ($\overline{\text{ACK}}$) is used by the transmitting and receiving devices to confirm serial data reception.

The receiving device returns one $\overline{\text{ACK}}$ signal for each 8 bits of data it receives. The transmitting device normally receives an $\overline{\text{ACK}}$ signal after transmitting 8 bits of data. However, when the master device is the receiving device, it does not output an $\overline{\text{ACK}}$ signal after receiving the final data to be transmitted. The transmitting device detects whether or not an $\overline{\text{ACK}}$ signal is returned after it transmits 8 bits of data. When an $\overline{\text{ACK}}$ signal is returned, the reception is judged as normal and processing continues. If the slave device does not return an $\overline{\text{ACK}}$ signal, the master device outputs either a stop condition or a restart condition and then stops the current transmission. Failure to return an $\overline{\text{ACK}}$ signal may be caused by the following two factors.

(a) Reception was not performed normally.

(b) The final data was received.

When the receiving device sets the SDAn line to low level during the ninth clock, the $\overline{\text{ACK}}$ signal becomes active (normal receive response).

When the IICCn.ACKEn bit is set to 1, automatic $\overline{\text{ACK}}$ signal generation is enabled.

Transmission of the eighth bit following the 7 address data bits causes the IICSn.TRCn bit to be set. When this TRCn bit's value is 0, it indicates receive mode. Therefore, the ACKEn bit should be set to 1.

When the slave device is receiving (when TRCn bit = 0), if the slave device does not need to receive any more data after receiving several bytes, clearing the ACKEn bit to 0 will prevent the master device from starting transmission of the subsequent data.

Similarly, when the master device is receiving (when TRCn bit = 0) and the subsequent data is not needed and when either a restart condition or a stop condition should therefore be output, clearing the ACKEn bit to 0 will prevent the $\overline{\text{ACK}}$ signal from being returned. This prevents the MSB from being output via the SDAn line (i.e., stops transmission) during transmission from the slave device.

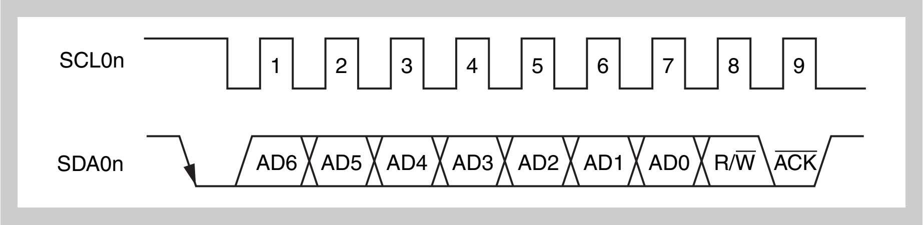


Figure 17-10 $\overline{\text{ACK}}$ signal

When the local address is received, an $\overline{\text{ACK}}$ signal is automatically output in synchronization with the falling edge of the SCLn pin's eighth clock regardless of the value of the ACKEn bit. No $\overline{\text{ACK}}$ signal is output if the received address is not a local address.

The $\overline{\text{ACK}}$ signal output method during data reception is based on the wait timing setting, as described below.

When 8-clock wait is selected (IICCn.WTIMn bit = 0):

The $\overline{\text{ACK}}$ signal is output at the falling edge of the SCLn pin's eighth clock if the ACKEn bit is set to 1 before wait cancellation.

When 9-clock wait is selected (IICCn.WTIMn bit = 1):

The $\overline{\text{ACK}}$ signal is automatically output at the falling edge of the SCLn pin's eighth clock if the ACKEn bit has already been set to 1.

17.8.5 Stop condition

When the SCLn pin is high level, changing the SDAn pin from low level to high level generates a stop condition.

A stop condition is a signal that the master device outputs to the slave device when serial transfer has been completed. When used as the slave device, the start condition can be detected.

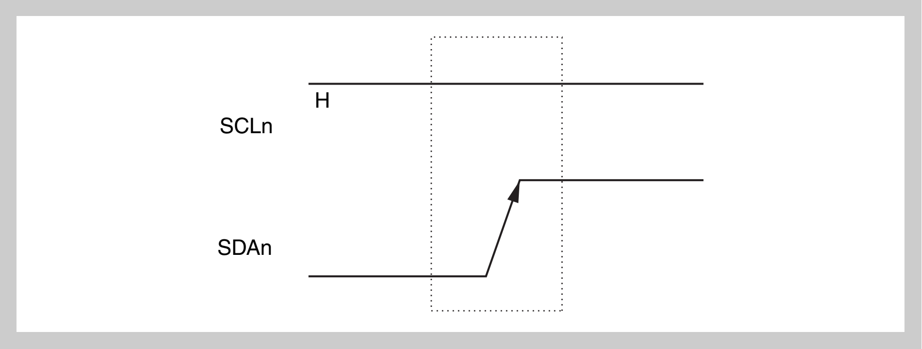


Figure 17-11 Stop condition

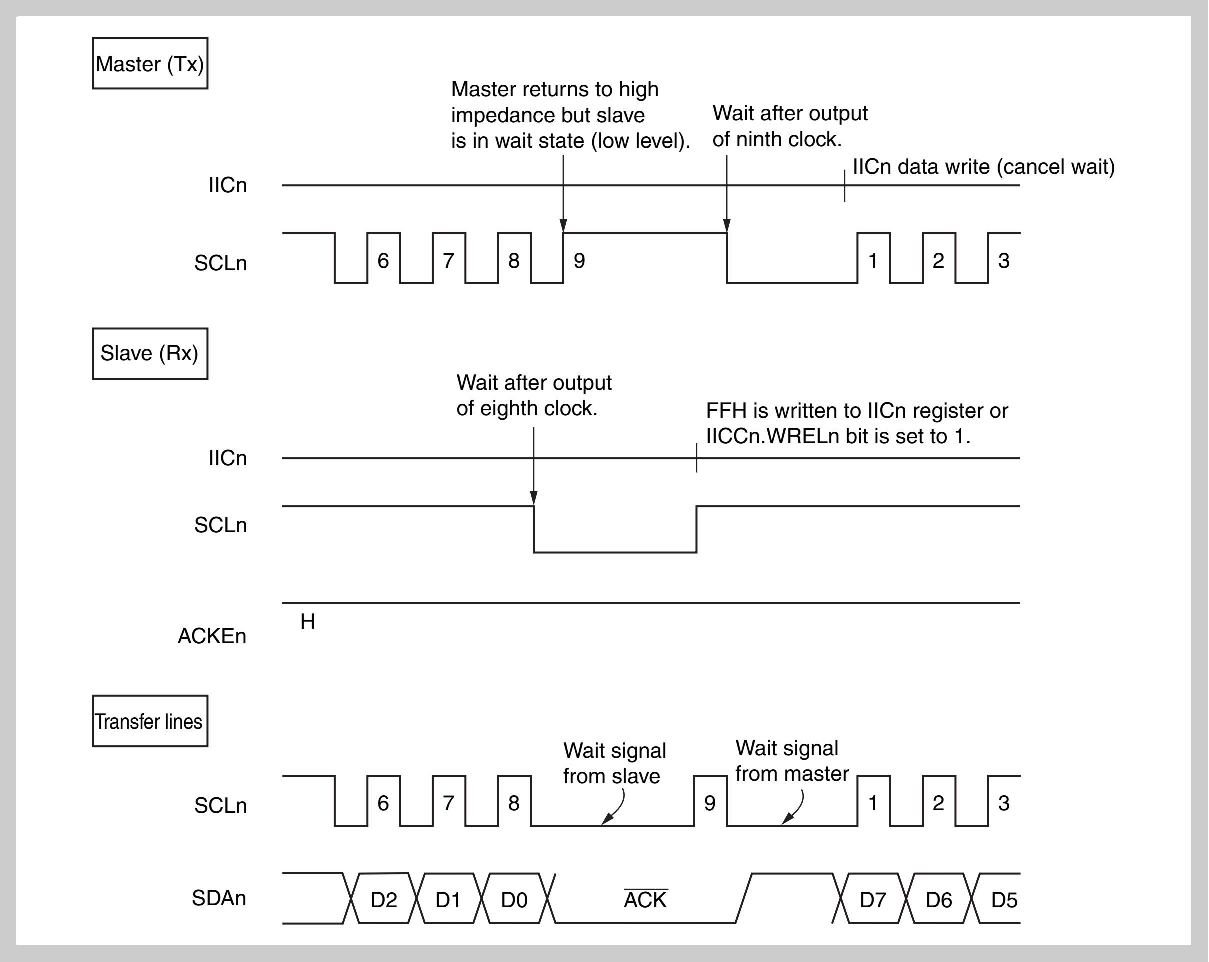
A stop condition is generated when the IICCn.SPTn bit is set to 1. When the stop condition is detected, the IICSn.SPDn bit is set to 1 and the INTIICn signal is generated when the IICCn.SPIEn bit is set to 1. By setting IICCN.STPn=1 the master device will also cancel its own wait status.

17.8.6 Wait signal ($\overline{\text{WAIT}}$)

The wait signal ($\overline{\text{WAIT}}$) is used to notify the communication partner that a device (master or slave) is preparing to transmit or receive data (i.e., is in a wait state).

Setting the SCLn pin to low level notifies the communication partner of the wait status. When the wait status has been cancelled for both the master and slave devices, the next data transfer can begin.

(1) When master device has a nine-clock wait and slave device has an eight-clock wait (master: transmission, slave: reception, and IICCn.ACKEn bit = 1)



Master (Tx)
Master returns to high impedance but slave is in wait state (low level).
Wait after output of ninth clock.
IICn data write (cancel wait)
IICn
SCLn
6
7
8
9
1
2
3
Slave (Rx)
Wait after output of eighth clock.
FFH is written to IICn register or IICCn.WRELn bit is set to 1.
IICn
SCLn
ACKEn
H
Transfer lines
Wait signal from slave
Wait signal from master
SCLn
6
7
8
9
1
2
3
SDAn
D2
D1
D0
ACK
D7
D6
D5

Figure 17-12 Wait signal (1/2)

(2) When master and slave devices both have a nine-clock wait (master: transmission, slave: reception, and ACKEn bit = 1)

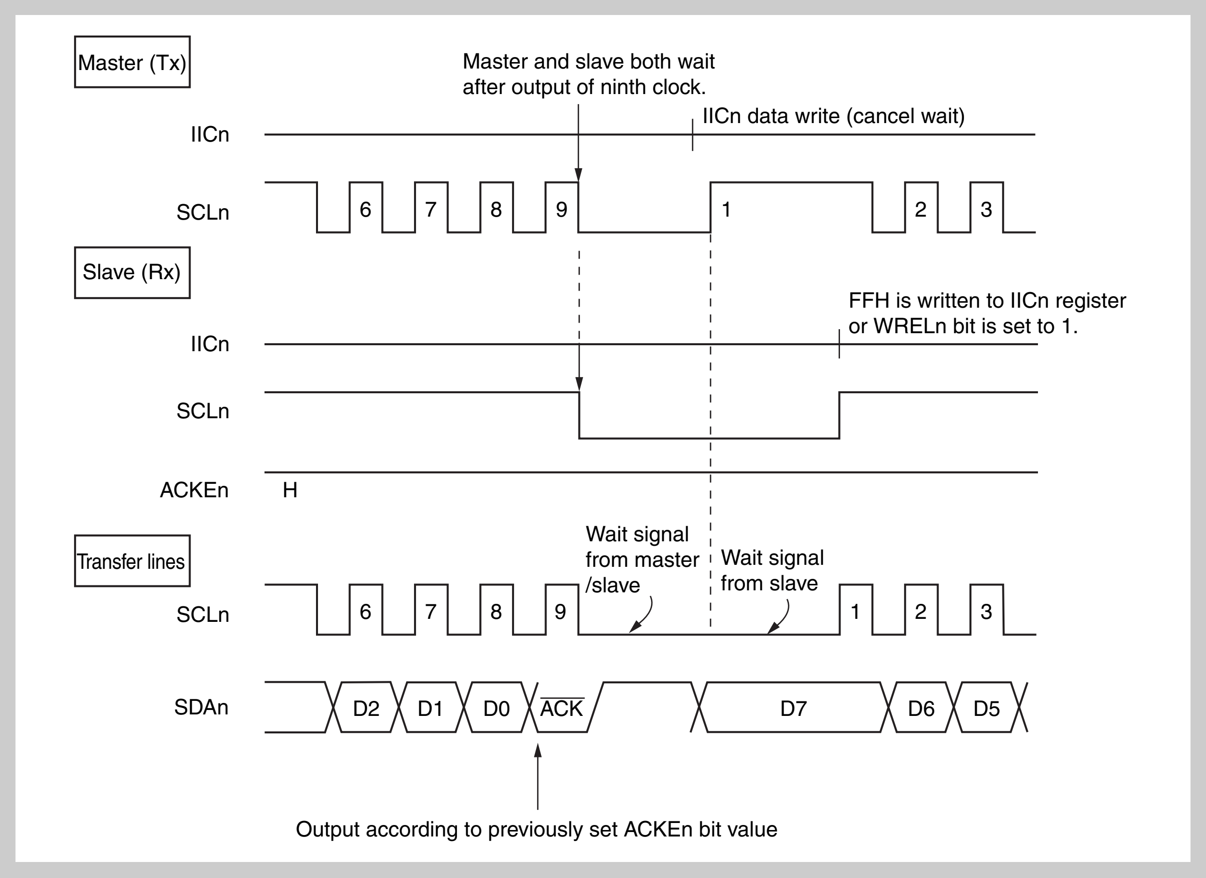

Master (Tx)
Master and slave both wait after output of ninth clock.
IICn
IICn data write (cancel wait)
SCLn
6
7
8
9
1
2
3
Slave (Rx)
FFH is written to IICn register or WRELn bit is set to 1.
IICn
SCLn
ACKEn
H
Transfer lines
Wait signal from master /slave
Wait signal from slave
SCLn
6
7
8
9
1
2
3
SDAn
D2
D1
D0
$\overline{\text{ACK}}$
D7
D6
D5
Output according to previously set ACKEn bit value

Figure 17-13 Wait signal (2/2)

A wait may be automatically generated depending on the setting of the IICCn.WTIMn bit.

Normally, when the IICCn.WRELn bit is set to 1 or when FFH is written to the IICn register on the receiving side, the wait status is cancelled and the transmitting side writes data to the IICn register to cancel the wait status.

The master device can also cancel its own wait status via either of the following methods.

- By setting the IICCn.STTn bit to 1
- By setting the IICCn.SPTn bit to 1

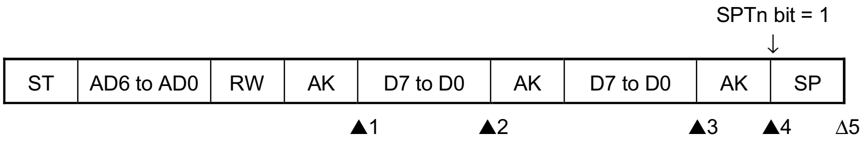
17.9 I^{2}C Interrupt Request Signals (INTIICn)

The following shows the value of the IICSn register at the INTIICn interrupt request signal generation timing and at the INTIICn signal timing.

17.9.1 Master device operation

(1) Start ~ Address ~ Data ~ Data ~ Stop (normal transmission/reception)

<1> When WTIMn bit = 0



▲1: IICSn register = 10XXX110B

▲2: IICSn register = 10XXX000B

▲3: IICSn register = 10XXX000B (WTIMn bit = 1)

▲4: IICSn register = 10XXXX00B

Δ 5: IICSn register = 00000001B

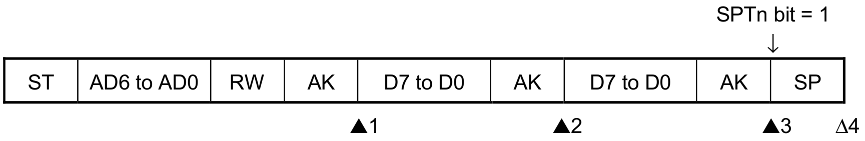
Remarks 1. ▲: Always generated

Δ: Generated only when SPIEn bit = 1

X: don't care

2. n = 0 to 2

<2> When WTIMn bit = 1



▲1: IICSn register = 10XXX110B

▲2: IICSn register = 10XXX100B

▲3: IICSn register = 10XXXX00B

Δ 4: IICSn register = 00000001B

Remarks 1. ▲: Always generated

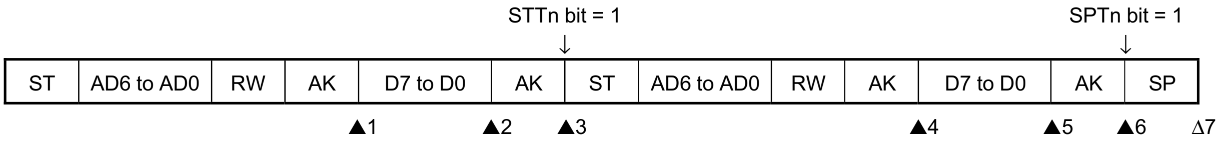
Δ: Generated only when SPIEn bit = 1

X: don't care

2. n = 0 to 2

(2) Start ~ Address ~ Data ~ Start ~ Address ~ Data ~ Stop (restart)

<1> When WTIMn bit = 0



▲1: IICSn register = 10XXX110B

▲2: IICSn register = 10XXX000B (WTIMn bit = 1)

▲3: IICSn register = 10XXXX00B (WTIMn bit = 0)

▲4: IICSn register = 10XXX110B (WTIMn bit = 0)

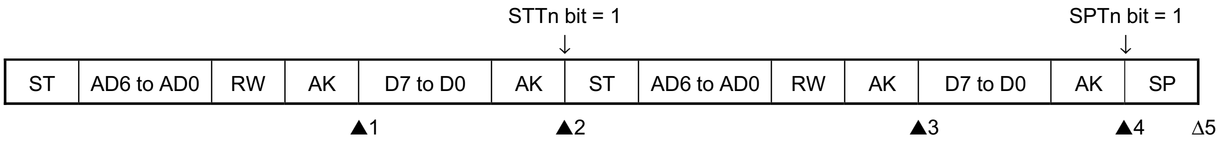
▲5: IICSn register = 10XXX000B (WTIMn bit = 1)

▲6: IICSn register = 10XXXX00B

Δ 7: IICSn register = 00000001B

Remarks 1. ▲: Always generated
Δ: Generated only when SPIEn bit = 1
X: don’t care
2. n = 0 to 2

<2> When WTIMn bit = 1



▲1: IICSn register = 10XXX110B

▲2: IICSn register = 10XXXX00B

▲3: IICSn register = 10XXX110B

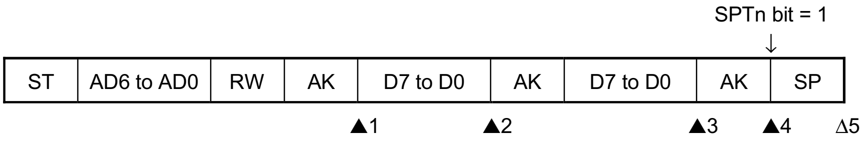
▲4: IICSn register = 10XXXX00B

Δ 5: IICSn register = 00000001B

Remarks 1. ▲: Always generated
Δ: Generated only when SPIEn bit = 1
X: don’t care
2. n = 0 to 2

(3) Start ~ Code ~ Data ~ Data ~ Stop (extension code transmission)

<1> When WTIMn bit = 0



▲1: IICSn register = 1010X110B

▲2: IICSn register = 1010X000B

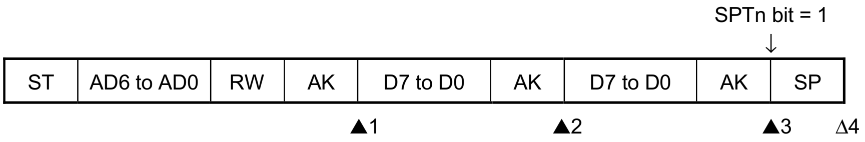
▲3: IICSn register = 1010X000B (WTIMn bit = 1)

▲4: IICSn register = 1010XX00B

Δ 5: IICSn register = 00000001B

Remarks 1. ▲: Always generated
Δ: Generated only when SPIEn bit = 1
X: don't care
2. n = 0 to 2

<2> When WTIMn bit = 1



▲1: IICSn register = 1010X110B

▲2: IICSn register = 1010X100B

▲3: IICSn register = 1010XX00B

Δ 4: IICSn register = 00000001B

Remarks 1. ▲: Always generated
Δ: Generated only when SPIEn bit = 1
X: don't care
2. n = 0 to 2

17.9.2 Slave device operation

(1) Start ~ Address ~ Data ~ Data ~ Stop

<1> When WTIMn bit = 0

ST	AD6 to AD0	RW	AK	D7 to D0	AK	D7 to D0	AK	SP
			▲1		▲2		▲3	Δ4

▲1: IICSn register = 0001X110B

▲2: IICSn register = 0001X000B

▲3: IICSn register = 0001X000B

Δ 4: IICSn register = 00000001B

Remarks 1. ▲: Always generated

Δ: Generated only when SPIEn bit = 1

X: don’t care

2. n = 0 to 2

<2> When WTIMn bit = 1

ST	AD6 to AD0	RW	AK	D7 to D0	AK	D7 to D0	AK	SP
			▲1			▲2	▲3	Δ4

▲1: IICSn register = 0001X110B

▲2: IICSn register = 0001X100B

▲3: IICSn register = 0001XX00B

Δ 4: IICSn register = 00000001B

Remarks 1. ▲: Always generated

Δ: Generated only when SPIEn bit = 1

X: don’t care

2. n = 0 to 2

(2) Start ~ Address ~ Data ~ Start ~ Address ~ Data ~ Stop

<1> When WTIMn bit = 0 (after restart, address match)

ST	AD6 to AD0	RW	AK	D7 to D0	AK	ST	AD6 to AD0	RW	AK	D7 to D0	AK	SP
			▲1	▲2					▲3	▲4		Δ5

▲1: IICSn register = 0001X110B

▲2: IICSn register = 0001X000B

▲3: IICSn register = 0001X110B

▲4: IICSn register = 0001X000B

Δ 5: IICSn register = 00000001B

Remarks 1. ▲: Always generated

Δ: Generated only when SPIEn bit = 1

X: don't care

2. n = 0 to 2

<2> When WTIMn bit = 1 (after restart, address match)

ST	AD6 to AD0	RW	AK	D7 to D0	AK	ST	AD6 to AD0	RW	AK	D7 to D0	AK	SP
			▲1		▲2				▲3		▲4	Δ5

▲1: IICSn register = 0001X110B

▲2: IICSn register = 0001XX00B

▲3: IICSn register = 0001X110B

▲4: IICSn register = 0001XX00B

Δ 5: IICSn register = 00000001B

Remarks 1. ▲: Always generated

Δ: Generated only when SPIEn bit = 1

X: don't care

2. n = 0 to 2

(3) Start ~ Address ~ Data ~ Start ~ Code ~ Data ~ Stop

<1> When WTIMn bit = 0 (after restart, extension code reception)

ST	AD6 to AD0	RW	AK	D7 to D0	AK	ST	AD6 to AD0	RW	AK	D7 to D0	AK	SP

▲1 ▲2 ▲3 ▲4 ∆5

▲1: IICSn register = 0001X110B

▲2: IICSn register = 0001X000B

▲3: IICSn register = 0010X010B

▲4: IICSn register = 0010X000B

∆ 5: IICSn register = 00000001B

Remarks 1. ▲: Always generated

∆: Generated only when SPIEn bit = 1

X: don’t care

2. n = 0 to 2

<2> When WTIMn bit = 1 (after restart, extension code reception)

ST	AD6 to AD0	RW	AK	D7 to D0	AK	ST	AD6 to AD0	RW	AK	D7 to D0	AK	SP

▲1 ▲2 ▲3 ▲4 ▲5 ∆6

▲1: IICSn register = 0001X110B

▲2: IICSn register = 0001XX00B

▲3: IICSn register = 0010X010B

▲4: IICSn register = 0010X110B

▲5: IICSn register = 0010XX00B

∆ 6: IICSn register = 00000001B

Remarks 1. ▲: Always generated

∆: Generated only when SPIEn bit = 1

X: don’t care

2. n = 0 to 2

(4) Start ~ Address ~ Data ~ Start ~ Address ~ Data ~ Stop

<1> When WTIMn bit = 0 (after restart, address mismatch (= not extension code))

ST	AD6 to AD0	RW	AK	D7 to D0	AK	ST	AD6 to AD0	RW	AK	D7 to D0	AK	SP
			▲1	▲2					▲3			△4

▲1: IICSn register = 0001X110B

▲2: IICSn register = 0001X000B

▲3: IICSn register = 00000X10B

△ 4: IICSn register = 00000001B

Remarks 1. ▲: Always generated

△: Generated only when SPIEn bit = 1

X: don’t care

2. n = 0 to 2

<2> When WTIMn bit = 1 (after restart, address mismatch (= not extension code))

ST	AD6 to AD0	RW	AK	D7 to D0	AK	ST	AD6 to AD0	RW	AK	D7 to D0	AK	SP
			▲1		▲2				▲3			△4

▲1: IICSn register = 0001X110B

▲2: IICSn register = 0001XX00B

▲3: IICSn register = 00000X10B

△ 4: IICSn register = 00000001B

Remarks 1. ▲: Always generated

△: Generated only when SPIEn bit = 1

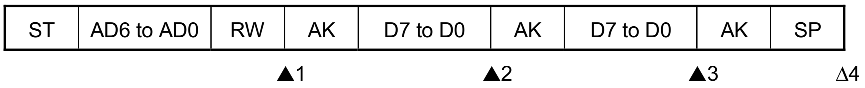
X: don’t care

2. n = 0 to 2

17.9.3 Slave device operation (when receiving extension code)

(1) Start ~ Code ~ Data ~ Data ~ Stop

<1> When WTIMn bit = 0



▲1: IICSn register = 0010X010B

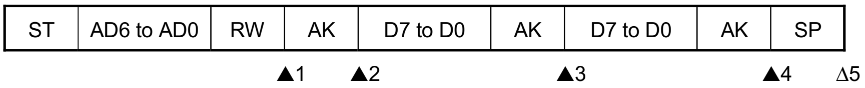
▲2: IICSn register = 0010X000B

▲3: IICSn register = 0010X000B

Δ 4: IICSn register = 00000001B

Remarks 1. ▲: Always generated
Δ: Generated only when SPIEn bit = 1
X: don't care
2. n = 0 to 2

<2> When WTIMn bit = 1



▲1: IICSn register = 0010X010B

▲2: IICSn register = 0010X110B

▲3: IICSn register = 0010X100B

▲4: IICSn register = 0010XX00B

Δ 5: IICSn register = 00000001B

Remarks 1. ▲: Always generated
Δ: Generated only when SPIEn bit = 1
X: don't care
2. n = 0 to 2

(2) Start ~ Code ~ Data ~ Start ~ Address ~ Data ~ Stop

<1> When WTIMn bit = 0 (after restart, address match)

ST	AD6 to AD0	RW	AK	D7 to D0	AK	ST	AD6 to AD0	RW	AK	D7 to D0	AK	SP	
			▲1		▲2					▲3	▲4		Δ5

▲1: IICSn register = 0010X010B

▲2: IICSn register = 0010X000B

▲3: IICSn register = 0001X110B

▲4: IICSn register = 0001X000B

Δ 5: IICSn register = 00000001B

Remarks 1. ▲: Always generated

Δ: Generated only when SPIEn bit = 1

X: don’t care

2. n = 0 to 2

<2> When WTIMn bit = 1 (after restart, address match)

ST	AD6 to AD0	RW	AK	D7 to D0	AK	ST	AD6 to AD0	RW	AK	D7 to D0	AK	SP	
			▲1	▲2		▲3				▲4		▲5	Δ6

▲1: IICSn register = 0010X010B

▲2: IICSn register = 0010X110B

▲3: IICSn register = 0010XX00B

▲4: IICSn register = 0001X110B

▲5: IICSn register = 0001XX00B

Δ 6: IICSn register = 00000001B

Remarks 1. ▲: Always generated

Δ: Generated only when SPIEn bit = 1

X: don’t care

2. n = 0 to 2

(3) Start ~ Code ~ Data ~ Start ~ Code ~ Data ~ Stop

<1> When WTIMn bit = 0 (after restart, extension code reception)

ST	AD6 to AD0	RW	AK	D7 to D0	AK	ST	AD6 to AD0	RW	AK	D7 to D0	AK	SP
		▲1		▲2				▲3		▲4		Δ5

▲1: IICSn register = 0010X010B

▲2: IICSn register = 0010X000B

▲3: IICSn register = 0010X010B

▲4: IICSn register = 0010X000B

Δ 5: IICSn register = 00000001B

Remarks 1. ▲: Always generated

Δ: Generated only when SPIEn bit = 1

X: don't care

2. n = 0 to 2

<2> When WTIMn bit = 1 (after restart, extension code reception)

ST	AD6 to AD0	RW	AK	D7 to D0	AK	ST	AD6 to AD0	RW	AK	D7 to D0	AK	SP
		▲1	▲2		▲3			▲4	▲5		▲6	Δ7

▲1: IICSn register = 0010X010B

▲2: IICSn register = 0010X110B

▲3: IICSn register = 0010XX00B

▲4: IICSn register = 0010X010B

▲5: IICSn register = 0010X110B

▲6: IICSn register = 0010XX00B

Δ 7: IICSn register = 00000001B

Remarks 1. ▲: Always generated

Δ: Generated only when SPIEn bit = 1

X: don't care

2. n = 0 to 2

(4) Start ~ Code ~ Data ~ Start ~ Address ~ Data ~ Stop

<1> When WTIMn bit = 0 (after restart, address mismatch (= not extension code))

ST	AD6 to AD0	RW	AK	D7 to D0	AK	ST	AD6 to AD0	RW	AK	D7 to D0	AK	SP
			▲1		▲2				▲3			Δ4

▲1: IICSn register = 0010X010B

▲2: IICSn register = 0010X000B

▲3: IICSn register = 00000X10B

Δ 4: IICSn register = 00000001B

Remarks 1. ▲: Always generated

Δ: Generated only when SPIEn bit = 1

X: don't care

2. n = 0 to 2

<2> When WTIMn bit = 1 (after restart, address mismatch (= not extension code))

ST	AD6 to AD0	RW	AK	D7 to D0	AK	ST	AD6 to AD0	RW	AK	D7 to D0	AK	SP
			▲1	▲2		▲3			▲4			Δ5

▲1: IICSn register = 0010X010B

▲2: IICSn register = 0010X110B

▲3: IICSn register = 0010XX00B

▲4: IICSn register = 00000X10B

Δ 5: IICSn register = 00000001B

Remarks 1. ▲: Always generated

Δ: Generated only when SPIEn bit = 1

X: don't care

2. n = 0 to 2

17.9.4 Operation without communication

(1) Start ~ Code ~ Data ~ Data ~ Stop

ST	AD6 to AD0	RW	AK	D7 to D0	AK	D7 to D0	AK	SP
								Δ1

Δ 1: IICSn register = 00000001B

Remarks 1. Δ: Generated only when SPIEn bit = 1

2. n = 0 to 2

17.9.5 Arbitration loss operation (operation as slave after arbitration loss)

(1) When arbitration loss occurs during transmission of slave address data

<1> When WTIMn bit = 0

ST	AD6 to AD0	RW	AK	D7 to D0	AK	D7 to D0	AK	SP
			▲1	▲2		▲3		Δ4

▲1: IICSn register = 0101X110B (Example: When ALDn bit is read during interrupt servicing)

▲2: IICSn register = 0001X000B

▲3: IICSn register = 0001X000B

Δ 4: IICSn register = 00000001B

Remarks 1. ▲: Always generated

Δ: Generated only when SPIEn bit = 1

X: don’t care

2. n = 0 to 2

<2> When WTIMn bit = 1

ST	AD6 to AD0	RW	AK	D7 to D0	AK	D7 to D0	AK	SP
			▲1		▲2		▲3	Δ4

▲1: IICSn register = 0101X110B (Example: When ALDn bit is read during interrupt servicing)

▲2: IICSn register = 0001X100B

▲3: IICSn register = 0001XX00B

Δ 4: IICSn register = 00000001B

Remarks 1. ▲: Always generated

Δ: Generated only when SPIEn bit = 1

X: don’t care

2. n = 0 to 2

(2) When arbitration loss occurs during transmission of extension code

<1> When WTIMn bit = 0

ST	AD6 to AD0	RW	AK	D7 to D0	AK	D7 to D0	AK	SP	
			▲1		▲2		▲3		Δ4

▲1: IICSn register = 0110X010B (Example: When ALDn bit is read during interrupt servicing)

▲2: IICSn register = 0010X000B

▲3: IICSn register = 0010X000B

Δ 4: IICSn register = 00000001B

Remarks 1. ▲: Always generated
Δ: Generated only when SPIEn bit = 1
X: don’t care
2. n = 0 to 2

<2> When WTIMn bit = 1

ST	AD6 to AD0	RW	AK	D7 to D0	AK	D7 to D0	AK	SP	
			▲1	▲2		▲3		▲4	Δ5

▲1: IICSn register = 0110X010B (Example: When ALDn bit is read during interrupt servicing)

▲2: IICSn register = 0010X110B

▲3: IICSn register = 0010X100B

▲4: IICSn register = 0010XX00B

Δ 5: IICSn register = 00000001B

Remarks 1. ▲: Always generated
Δ: Generated only when SPIEn bit = 1
X: don’t care
2. n = 0 to 2

17.9.6 Operation when arbitration loss occurs

(1) When arbitration loss occurs during transmission of slave address data

ST	AD6 to AD0	RW	AK	D7 to D0	AK	D7 to D0	AK	SP

▲1 ∆2

▲1: IICSn register = 01000110B (Example: When ALDn bit is read during interrupt servicing)

∆ 2: IICSn register = 00000001B

Remarks 1. ▲: Always generated

∆: Generated only when SPIEn bit = 1

2. n = 0 to 2

(2) When arbitration loss occurs during transmission of extension code

ST	AD6 to AD0	RW	AK	D7 to D0	AK	D7 to D0	AK	SP

▲1 ∆2

▲1: IICSn register = 0110X010B (Example: When ALDn bit is read during interrupt servicing)

IICCn.LRELn bit is set to 1 by software

∆ 2: IICSn register = 00000001B

Remarks 1. ▲: Always generated

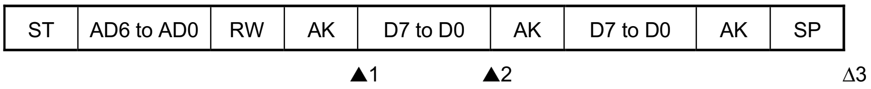
∆: Generated only when SPIEn bit = 1

X: don’t care

2. n = 0 to 2

(3) When arbitration loss occurs during data transfer

<1> When WTIMn bit = 0



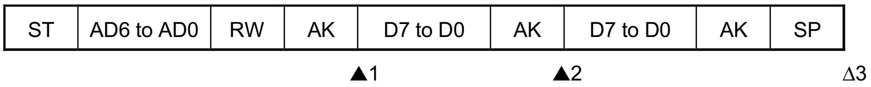
▲1: IICSn register = 10001110B

▲2: IICSn register = 01000000B (Example: When ALDn bit is read during interrupt servicing)

Δ 3: IICSn register = 00000001B

Remarks 1. ▲: Always generated
Δ: Generated only when SPIEn bit = 1
2. n = 0 to 2

<2> When WTIMn bit = 1



▲1: IICSn register = 10001110B

▲2: IICSn register = 01000100B (Example: When ALDn bit is read during interrupt servicing)

Δ 3: IICSn register = 00000001B

Remarks 1. ▲: Always generated
Δ: Generated only when SPIEn bit = 1
2. n = 0 to 2

(4) When arbitration loss occurs due to restart condition during data transfer

<1> Not extension code (Example: Address mismatch)

ST	AD6 to AD0	RW	AK	D7 to Dn	ST	AD6 to AD0	RW	AK	D7 to D0	AK	SP
			▲1					▲2			Δ3

▲1: IICSn register = 1000X110B

▲2: IICSn register = 01000110B (Example: When ALDn bit is read during interrupt servicing)

Δ 3: IICSn register = 00000001B

Remarks 1. ▲: Always generated

Δ: Generated only when SPIEn bit = 1

X: don’t care

2. Dn = D6 to D0

n = 0 to 2

<2> Extension code

ST	AD6 to AD0	RW	AK	D7 to Dn	ST	AD6 to AD0	RW	AK	D7 to D0	AK	SP
			▲1					▲2			Δ3

▲1: IICSn register = 1000X110B

▲2: IICSn register = 0110X010B (Example: When ALDn bit is read during interrupt servicing)

IICCn.LRELn bit is set to 1 by software

Δ 3: IICSn register = 00000001B

Remarks 1. ▲: Always generated

Δ: Generated only when SPIEn bit = 1

X: don’t care

2. Dn = D6 to D0

n = 0 to 2

(5) When arbitration loss occurs due to stop condition during data transfer

ST	AD6 to AD0	RW	AK	D7 to Dn	SP
			▲1		Δ2

▲1: IICSn register = 1000X110B

Δ 2: IICSn register = 01000001B

Remarks 1. ▲: Always generated

Δ: Generated only when SPIEn bit = 1

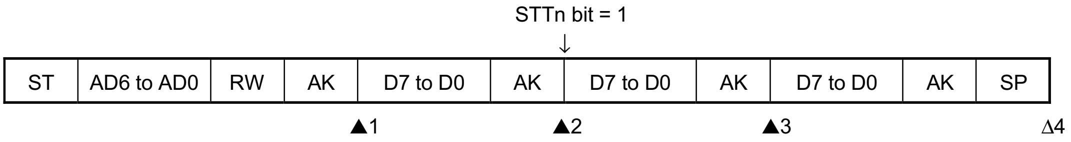
X: don’t care

2. Dn = D6 to D0

n = 0 to 2

(6) When arbitration loss occurs due to low level of SDAn pin when attempting to generate a restart condition

When WTIMn bit = 1



ST	AD6 to AD0	RW	AK	D7 to D0	AK	D7 to D0	AK	D7 to D0	AK	SP	
			▲1		▲2		▲3				Δ4

▲1: IICSn register = 1000X110B

▲2: IICSn register = 1000XX00B

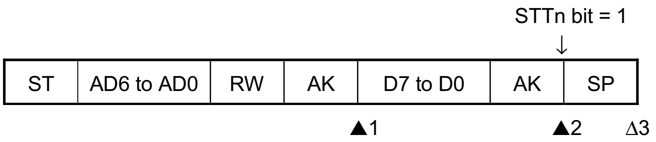
▲3: IICSn register = 01000100B (Example: When ALDn bit is read during interrupt servicing)

Δ 4: IICSn register = 00000001B

Remarks 1. ▲: Always generated
Δ: Generated only when SPIEn bit = 1
X: don't care
2. n = 0 to 2

(7) When arbitration loss occurs due to a stop condition when attempting to generate a restart condition

When WTIMn bit = 1



ST	AD6 to AD0	RW	AK	D7 to D0	AK	SP	
			▲1		▲2		Δ3

▲1: IICSn register = 1000X110B

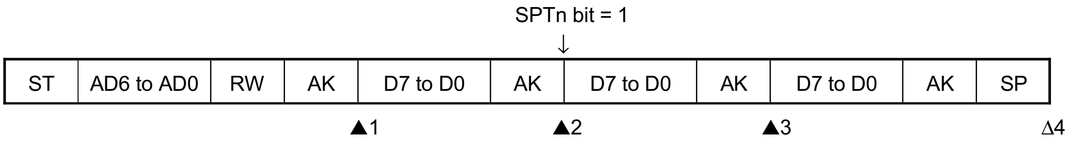
▲2: IICSn register = 1000XX00B

Δ 3: IICSn register = 01000001B

Remarks 1. ▲: Always generated
Δ: Generated only when SPIEn bit = 1
X: don't care
2. n = 0 to 2

(8) When arbitration loss occurs due to low level of SDAn pin when attempting to generate a stop condition

When WTIMn bit = 1



▲1: IICSn register = 1000X110B

▲2: IICSn register = 1000XX00B

▲3: IICSn register = 01000000B (Example: When ALDn bit is read during interrupt servicing)

Δ 4: IICSn register = 00000001B

Remarks 1. ▲: Always generated
Δ: Generated only when SPIEn bit = 1
X: don't care

2. n = 0 to 2

17.10 Interrupt Request Signal (INTIICn)

The setting of the IICCn.WTIMn bit determines the timing by which the INTIICn register is generated and the corresponding wait control, as shown below.

Table 17-6 INTIICn generation timing and wait control

WTIMn Bit	During Slave Device Operation			During Master Device Operation		
	Address	Data Reception	Data Transmission	Address	Data Reception	Data Transmission
0	**9**Notes 1, 2	**8**Note 2	**8**Note 2	9	8	8
1	**9**Notes 1, 2	**9**Note 2	**9**Note 2	9	9	9

Note 1. The slave device's INTIICn signal and wait period occur at the falling edge of the ninth clock only when there is a match with the address set to the SVAn register.
At this point, the $\overline{\text{ACK}}$ signal is output regardless of the value set to the IICCn.ACKEn bit. For a slave device that has received an extension code, the INTIICn signal occurs at the falling edge of the eighth clock.
When the address does not match after restart, the INTIICn signal is generated at the falling edge of the ninth clock, but no wait occurs.

2. If the received address does not match the contents of the SVAn register and an extension code is not received, neither the INTIICn signal nor a wait occurs.

3. The numbers in the table indicate the number of the serial clock's clock signals. Interrupt requests and wait control are both synchronized with the falling edge of these clock signals.

(1) During address transmission/reception

- Slave device operation:
  Interrupt and wait timing are determined regardless of the WTIMn bit.
- Master device operation:
  Interrupt and wait timing occur at the falling edge of the ninth clock regardless of the WTIMn bit.

(2) During data reception

- Master/slave device operation: Interrupt and wait timing is determined according to the WTIMn bit.

(3) During data transmission

- Master/slave device operation: Interrupt and wait timing is determined according to the WTIMn bit.

(4) Wait cancellation method

The four wait cancellation methods are as follows.

- By setting the IICCn.WRELn bit to 1
- By writing to the IICn register
- By start condition setting (IICCn.STTn bit = 1)[Note]
- By stop condition setting (IICCn.SPTn bit = 1)[Note]

Note Master only

When an 8-clock wait has been selected (WTIMn bit = 0), the output level of the $\overline{\text{ACK}}$ signal must be determined prior to wait cancellation.

(5) Stop condition detection

The INTIICn signal is generated when a stop condition is detected.

17.11 Address Match Detection Method

In I^{2}C bus mode, the master device can select a particular slave device by transmitting the corresponding slave address.

Address match detection is performed automatically by hardware. The INTIICn signal occurs when a local address has been set to the SVAn register and when the address set to the SVAn register matches the slave address sent by the master device, or when an extension code has been received.

17.12 Error Detection

In I^{2}C bus mode, the status of the serial data bus pin (SDAn) during data transmission is captured by the IICn register of the transmitting device, so the data of the IICn register prior to transmission can be compared with the transmitted IICn data to enable detection of transmission errors. A transmission error is judged as having occurred when the compared data values do not match.

17.13 Extension Code

- When the higher 4 bits of the receive address are either 0000 or 1111, the extension code flag (IICSn.EXCn bit) is set for extension code reception and an interrupt request signal (INTIICn) is issued at the falling edge of the eighth clock.

  The local address stored in the SVAn register is not affected.

- If 11110xx0 is set to the SVAn register by a 10-bit address transfer and 11110xx0 is transferred from the master device, the results are as follows. Note that the INTIICn signal occurs at the falling edge of the eighth clock

  – Higher four bits of data match: EXCn bit = 1

  – Seven bits of data match: IICSn.COIn bit = 1

- Since the processing after the interrupt request signal occurs differs according to the data that follows the extension code, such processing is performed by software.

  For example, when operation as a slave is not desired after the extension code is received, set the IICCn.LRELn bit to 1 and the CPU will enter the next communication wait state.

Table 17-7 Extension code bit definitions

Slave Address	R/W Bit	Description
0000 000	0	General call address
0000 000	1	Start byte
0000 001	X	CBUS address
0000 010	X	Address that is reserved for different bus format
1111 0xx	X	10-bit slave address specification

17.14 Arbitration

When several master devices simultaneously output a start condition (when the IICCn.STTn bit is set to 1 before the IICSn.STDn bit is set to 1), communication between the master devices is performed while the number of clocks is adjusted until the data differs. This kind of operation is called arbitration.

When one of the master devices loses in arbitration, an arbitration loss flag (IICSn.ALDn bit) is set to 1 via the timing by which the arbitration loss occurred, and the SCLn and SDAn lines are both set to high impedance, which releases the bus.

Arbitration loss is detected based on the timing of the next interrupt request signal (the eighth or ninth clock, when a stop condition is detected, etc.) and the setting of the ALDn bit to 1, which is made by software.

For details of interrupt request timing, see *“I^{2}C Interrupt Request Signals (INTIICn)“ on page 504*.

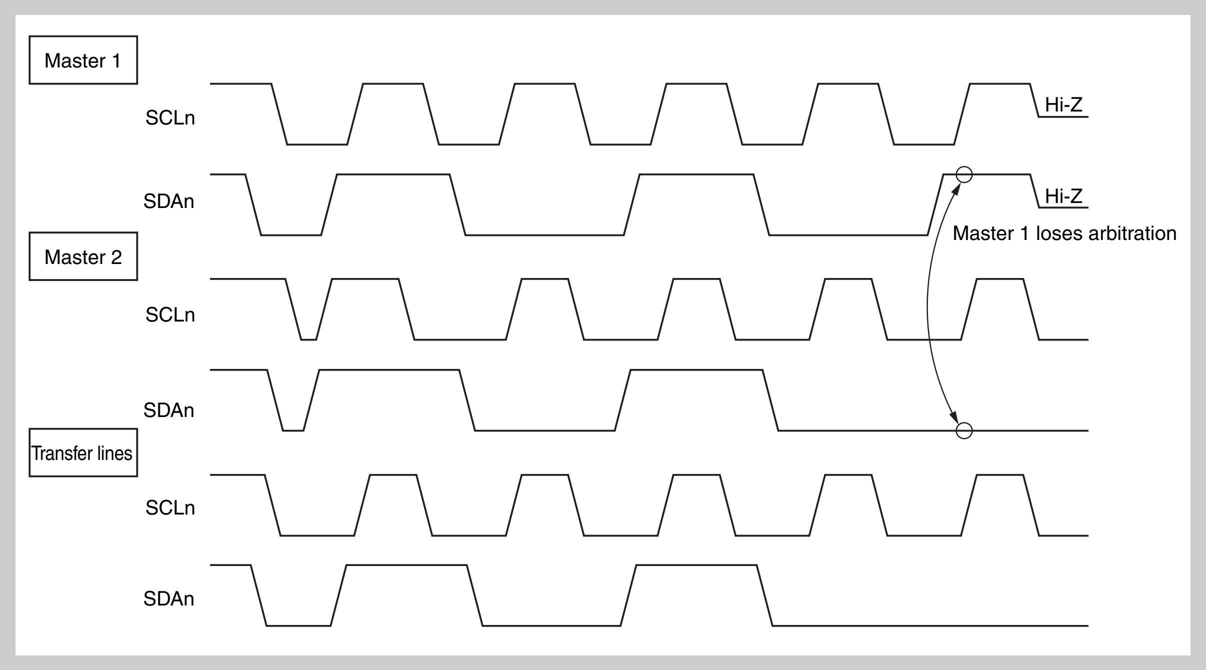


Figure 17-14 Arbitration timing example

Table 17-8 Status during arbitration and interrupt request signal generation timing

Status During Arbitration	Interrupt Request Generation Timing
Transmitting address transmission	At falling edge of eighth or ninth clock following byte transfer[Note 1]
Read/write data after address transmission	
Transmitting extension code	
Read/write data after extension code transmission	
Transmitting data	
$\overline{\text{ACK}}$ signal transfer period after data reception	
When restart condition is detected during data transfer	
When stop condition is detected during data transfer	When stop condition is output (when IICCn.SPIEn bit = 1)[Note 2]
When SDAn pin is low level while attempting to output restart condition	At falling edge of eighth or ninth clock following byte transfer[Note 1]
When stop condition is detected while attempting to output restart condition	When stop condition is output (when IICCn.SPIEn bit = 1)[Note 2]
When DSA0n pin is low level while attempting to output stop condition	At falling edge of eighth or ninth clock following byte transfer[Note 1]
When SCLn pin is low level while attempting to output restart condition	

Note 1. When the IICCn.WTIMn bit = 1, an interrupt request signal occurs at the falling edge of the ninth clock. When the WTIMn bit = 0 and the extension code's slave address is received, an interrupt request signal occurs at the falling edge of the eighth clock.

2. When there is a possibility that arbitration will occur, set the SPIEn bit to 1 for master device operation.

17.15 Wakeup Function

The I^2C bus slave function is a function that generates an interrupt request signal (INTIICn) when a local address and extension code have been received.

This function makes processing more efficient by preventing unnecessary interrupt request signals from occurring when addresses do not match.

When a start condition is detected, wakeup stand-by mode is set. This wakeup stand-by mode is in effect while addresses are transmitted due to the possibility that an arbitration loss may change the master device (which has output a start condition) to a slave device.

However, when a stop condition is detected, the IICCn.SPIEn bit is set regardless of the wakeup function, and this determines whether interrupt request signals are enabled or disabled.

17.16 Communication Reservation

17.16.1 Communication reservation function is enabled (IICFn.IICRSVn bit = 0)

To start master device communications when not currently using the bus, a communication reservation can be made to enable transmission of a start condition when the bus is released.
There are two modes in which the bus is not used:

- when arbitration results in neither master nor slave operation
- when an extension code is received and slave operation is disabled (acknowledge is not returned and the bus was released when the IICCn.LRELn bit was set to 1).

If the IICCn.STTn bit is set to 1 while the bus is not used, a start condition is automatically generated and a wait status is set after the bus is released (after a stop condition is detected).

When the bus release is detected (when a stop condition is detected), writing to the IICn register causes master address transfer to start. At this point, the IICCn.SPIEn bit should be set to 1.

When STTn has been set to 1, the operation mode (as start condition or as communication reservation) is determined according to the bus status.

- If the bus has been released:
  start condition is generated
- If the bus has not been released (standby mode):
  Communication reservation

To detect which operation mode has been determined for the STTn bit, set the STTn bit to 1, wait for the wait period, then check the IICSn.MSTSn bit.

The wait periods, which should be set via software, are listed in *Table 17-9*. These wait periods can be set by the SMCn, CLn1, and CLn0 bits of the IICCLn register and the IICXn.CLXn bit.

Table 17-9 Wait periods with communication reservation function enabled

Prescaler		I^{2}C module input clock	I^{2}C module set-up				Transfer clock IICLKTC	Mode	Waiting time in IICLK cycles
OCKS	IICLKPS		IICNn. CLXn	IICCLn .SMCn	IICCLn .CLn1	IICCLn .CLn0			
18_H	IICLK	IICLKPS	0	0	0	0	IICLK/44	standard	26
10_H	IICLK/2		0	0	0	0	1/2 * IICLK/44		52
11_H	IICLK/3		0	0	0	0	1/3 * IICLK/44		78
12_H	IICLK/4		0	0	0	0	14 * IICLK/44		104
13_H	IICLK/5		0	0	0	0	1/5 * IICLK/44		130
18_H	IICLK	IICLKPS	0	0	0	1	IICLK/86	standard	47
10_H	IICLK/2		0	0	0	1	1/2 * IICLK/86		94
11_H	IICLK/3		0	0	0	1	1/3 * IICLK/86		141
12_H	IICLK/4		0	0	0	1	14 * IICLK/86		188
13_H	IICLK/5		0	0	0	1	1/5 * IICLK/86		235
X	X	IICLK	0	0	1	0	IICLK/86		47
18_H	IICLK	IICLKPS	0	0	1	1	IICLK/66	standard	38
10_H	IICLK/2		0	0	1	1	1/2 * IICLK/66		76
11_H	IICLK/3		0	0	1	1	1/3 * IICLK/66		114
12_H	IICLK/4		0	0	1	1	14 * IICLK/66		152
13_H	IICLK/5		0	0	1	1	1/5 * IICLK/66		190
18_H	IICLK	IICLKPS	0	1	0	X	IICLK/24	fast-speed	16
10_H	IICLK/2		0	1	0	X	1/2 * IICLK/24		32
11_H	IICLK/3		0	1	0	X	1/3 * IICLK/24		48
12_H	IICLK/4		0	1	0	X	14 * IICLK/24		64
13_H	IICLK/5		0	1	0	X	1/5 * IICLK/24		80
X	X	IICLK	0	1	1	0	IICLK/24		16
18_H	IICLK	IICLKPS	0	1	1	1	IICLK/18	fast-speed	13
10_H	IICLK/2		0	1	1	1	1/2 * IICLK/18		26
11_H	IICLK/3		0	1	1	1	1/3 * IICLK/18		39
12_H	IICLK/4		0	1	1	1	14 * IICLK/18		52
13_H	IICLK/5		0	1	1	1	1/5 * IICLK/18		65
18_H	IICLK	IICLKPS	1	1	0	X	IICLK/12	fast-speed	10
10_H	IICLK/2		1	1	0	X	1/2 * IICLK/12		20
11_H	IICLK/3		1	1	0	X	1/3 * IICLK/12		30
12_H	IICLK/4		1	1	0	X	14 * IICLK/12		40
13_H	IICLK/5		1	1	0	X	1/5 * IICLK/12		50
X	X	IICLK	1	1	1	0	IICLK/12		10

The communication reservation timing is shown below.

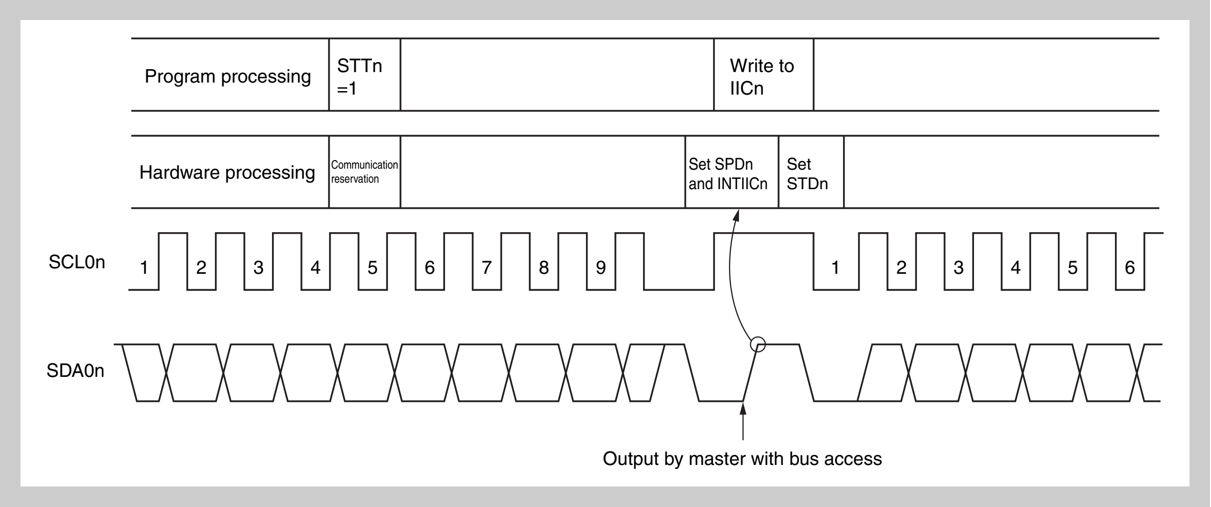


Figure 17-15 Communication reservation timing

Communication reservations are accepted via the following timing. After the IICSn.STDn bit is set to 1, a communication reservation can be made by setting the IICCn.STTn bit to 1 before a stop condition is detected.

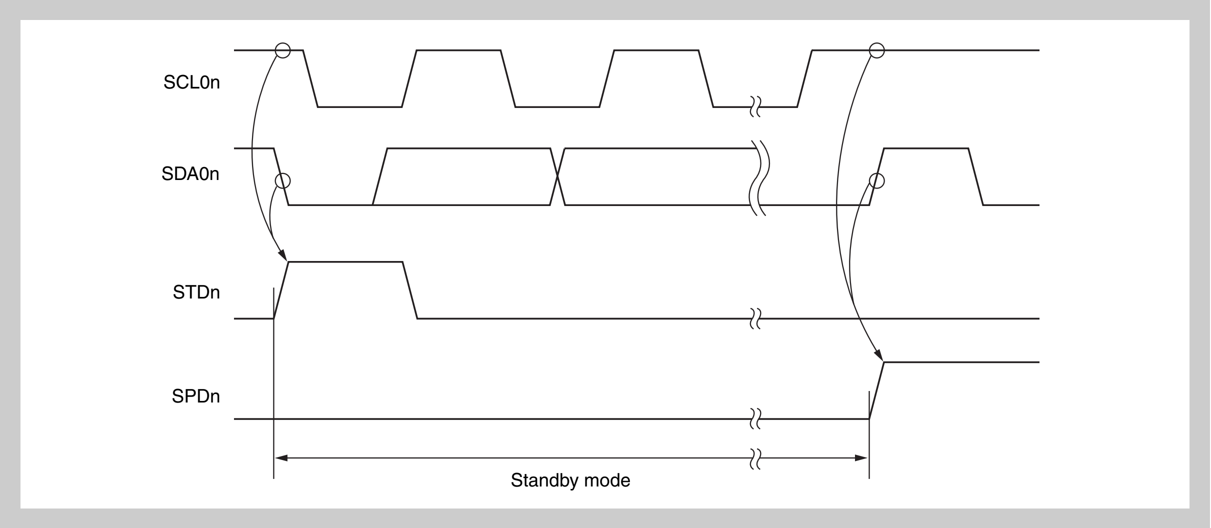


Figure 17-16 Timing for accepting communication reservations

The communication reservation flowchart is illustrated below.

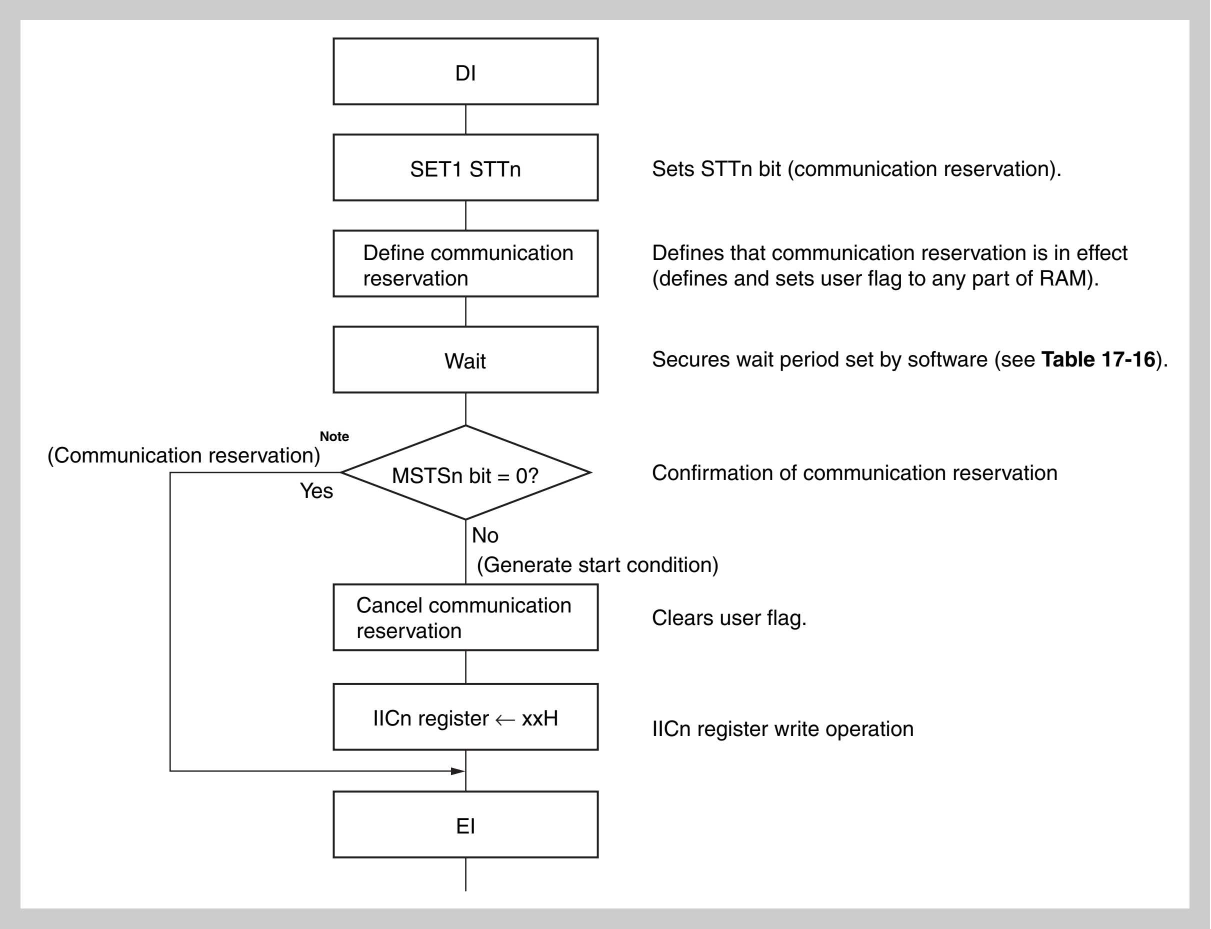

DI
SET1 STTn
Sets STTn bit (communication reservation).
Define communication reservation
Defines that communication reservation is in effect (defines and sets user flag to any part of RAM).
Wait
Secures wait period set by software (see **Table 17-16**).
(Communication reservation)
Note
MSTSn bit = 0?
Yes
Confirmation of communication reservation
No
(Generate start condition)
Cancel communication reservation
Clears user flag.
IICn register ← xxH
IICn register write operation
EI

Figure 17-17 Communication reservation flowchart

Note The communication reservation operation executes a write to the IICn register when a stop condition interrupt request occurs.

17.16.2 Communication reservation function is disabled (IICFn.IICRSVn bit = 1)

When the IICCn.STTn bit is set when the bus is not used in a communication during bus communication, this request is rejected and a start condition is not generated.
There are two modes in which the bus is not used:

- when arbitration results in neither master nor slave operation
- when an extension code is received and slave operation is disabled (acknowledge is not returned and the bus was released when the IICCn.LRELn bit was set to 1)

To confirm whether the start condition was generated or request was rejected, check the IICFn.STCFn flag. The time shown in *Table 17-10* is required until the STCFn flag is set after setting the STTn bit to 1. Therefore, secure the time by software.

Table 17-10 Wait periods with communication reservation function disabled

Prescaler		I^2C module input clock	I^2C module set-up				Waiting time in IICLK cycles
OCKS	IICLKPS		IICNn. CLXn	IICCLn .SMCn	IICCLn .CLn1	IICCLn .CLn0	
18_H	IICLK	IICLKPS	X	X	0	X	5
10_H	IICLK/2		X	X	0	X	10
11_H	IICLK/3		X	X	0	X	15
12_H	IICLK/4		X	X	0	X	20
13_H	IICLK/5		X	X	0	X	25
X	X	IICLK	X	X	1	0	5

17.17 Cautions

(1) When IICFn.STCENn bit = 0

Immediately after the I^{2}C0n operation is enabled, the bus communication status (IICFn.IICBSYn bit = 1) is recognized regardless of the actual bus status. To execute master communication in the status where a stop condition has not been detected, generate a stop condition and then release the bus before starting the master communication.

Use the following sequence for generating a stop condition.

<1> Set the IICCLn register.
<2> Set the IICCn.IICEn bit.
<3> Set the IICCn.SPTn bit.

(2) When IICFn.STCENn bit = 1

Immediately after I^{2}C0n operation is enabled, the bus released status (IICBSYn bit = 0) is recognized regardless of the actual bus status. To issue the first start condition (IICCn.STTn bit = 1), it is necessary to confirm that the bus has been released, so as to not disturb other communications.

(3) When the IICCn.IICEn bit is set to 1 while communications with other devices are in progress, the start condition may be detected depending on the status of the communication line. Be sure to set the IICCn.IICEn bit to 1 when the SCL0n and SDA0n lines are high level.

(4) Determine the operation clock frequency by the IICCLn, IICXn, and OCKSm registers before enabling the operation (IICCn.IICEn bit = 1). To change the operation clock frequency, clear the IICCn.IICEn bit to 0 once.

(5) After the IICCn.STTn and IICCn.SPTn bits have been set to 1, they must not be reset without being cleared to 0 first.

(6) If transmission has been reserved, set the IICCN.SPIEn bit to 1 so that an interrupt request is generated by the detection of a stop condition. After an interrupt request has been generated, the wait status will be released by writing communication data to I2Cn, then transferring will begin. If an interrupt is not generated by the detection of a stop condition, transmission will halt in the wait status because an interrupt request was not generated.

However, it is not necessary to set the SPIEn bit to 1 for the software to detect the IICSn.MSTSn bit.

17.18 Communication Operations

17.18.1 Master operation with communication reservation

The following shows the flowchart for master communication when the communication reservation function is enabled (IICFn.IICRSVn bit = 0) and the master operation is started after a stop condition is detected (IICFn.STCENn bit = 0).

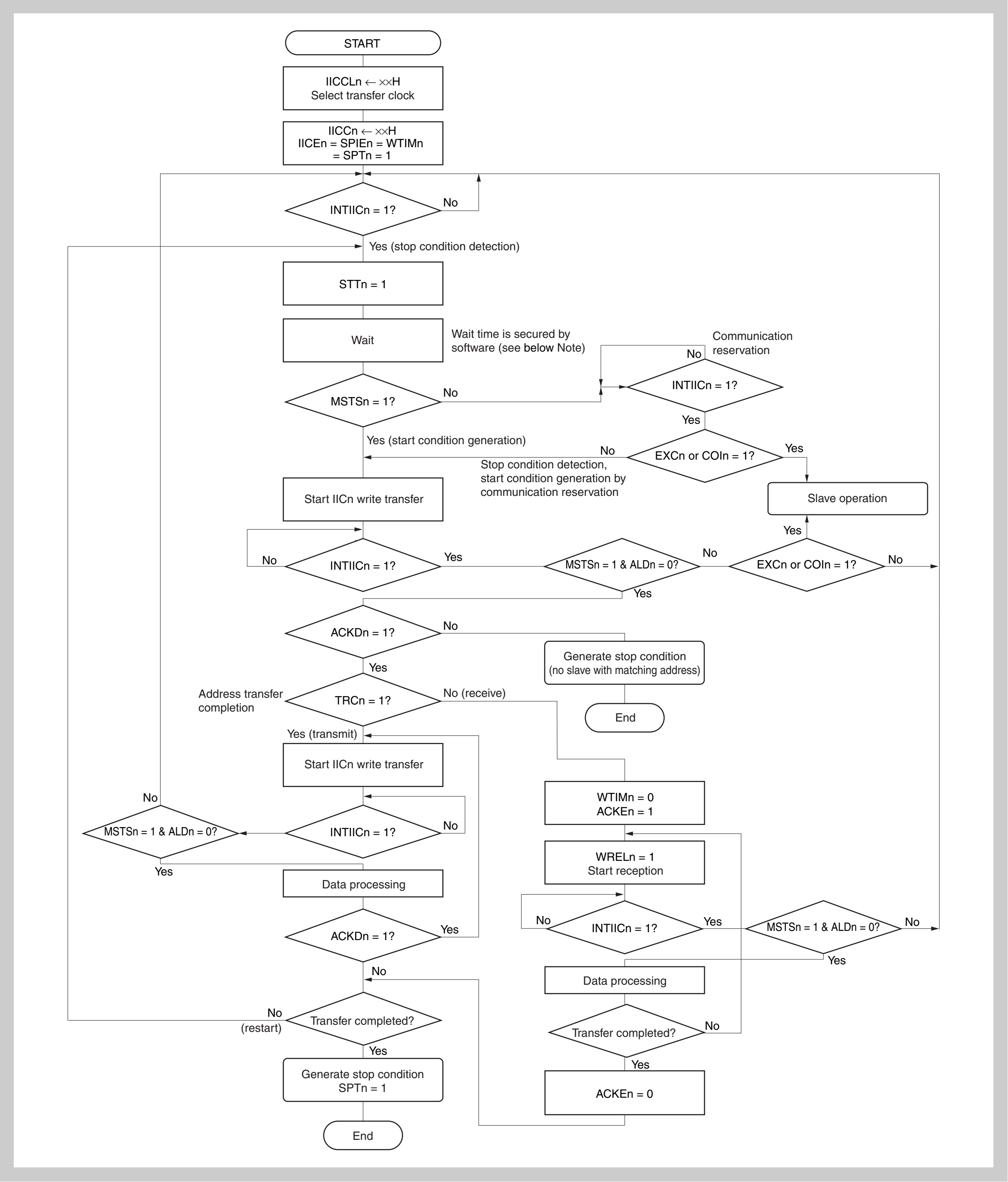


Figure 17-18 Master operation flowchart with communication reservation

Note Refer to *Table 17-9 on page 528*.

17.18.2 Master operation without communication reservation

The following shows the flowchart for master communication when the communication reservation function is disabled (IICRSVn bit = 1) and the master operation is started without detecting a stop condition (STCENn bit = 1).

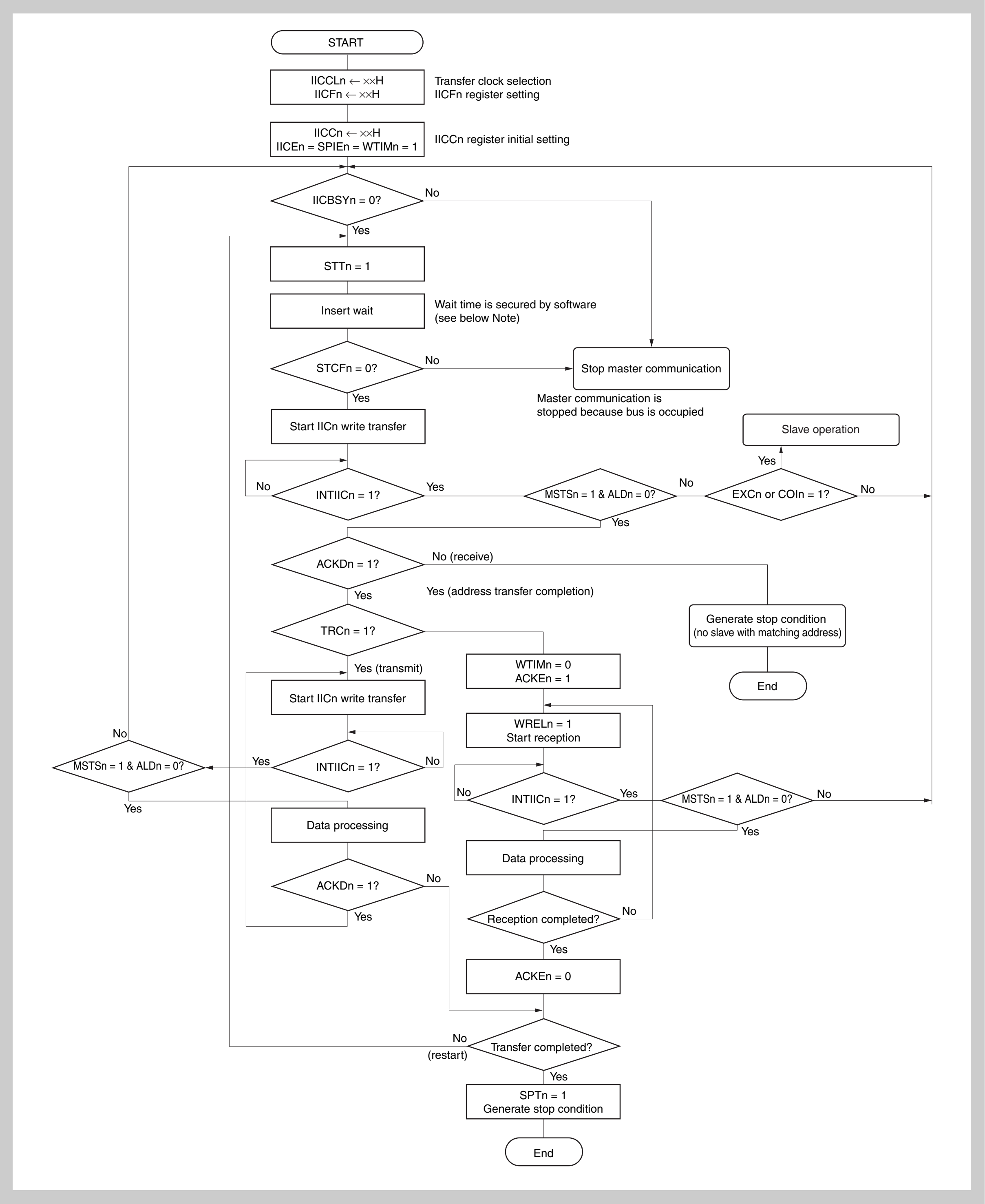


Figure 17-19 Master operation flowchart without communication reservation

Note Refer to *Table 17-10 on page 531*.

17.18.3 Slave operation

The following shows the processing procedure of the slave operation.

Basically, the operation of the slave device is event-driven. Therefore, processing by an INTIICn interrupt (processing requiring a significant change of the operation status, such as stop condition detection during communication) is necessary.

The following description assumes that data communication does not support extension codes. Also, it is assumed that the INTIICn interrupt servicing performs only status change processing and that the actual data communication is performed during the main processing.

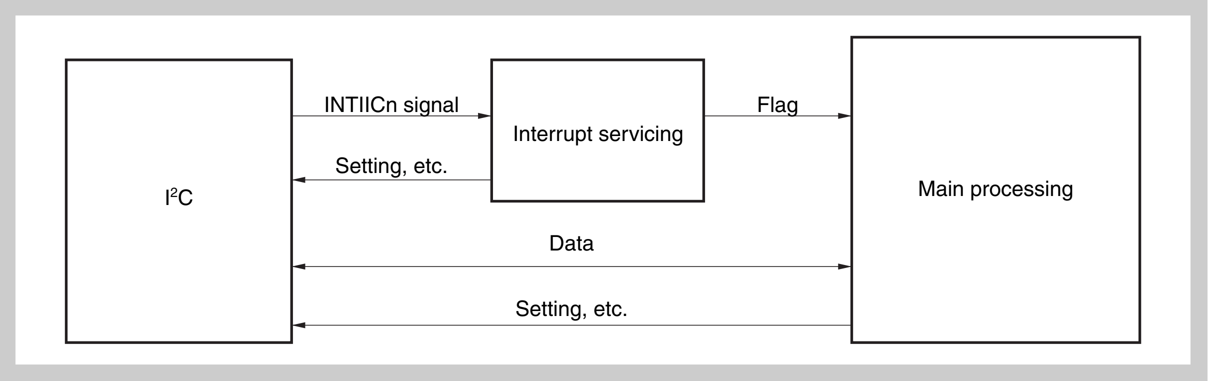


Figure 17-20 Software outline during slave operation

Therefore, the following three flags are prepared so that the data transfer processing can be performed by transmitting these flags to the main processing instead of INTIICn signal.

(1) Communication mode flag

This flag indicates the following communication statuses.

- Clear mode:
  Data communication not in progress
- Communication mode
  Data communication in progress (valid address detection stop condition detection, $\overline{\text{ACK}}$ signal from master not detected, address mismatch)

(2) Ready flag

This flag indicates that data communication is enabled. This is the same status as an INTIICn interrupt during normal data transfer. This flag is set in the interrupt processing block and cleared in the main processing block. The ready flag for the first data for transmission is not set in the interrupt processing block, so the first data is transmitted without clear processing (the address match is regarded as a request for the next data).

(3) Communication direction flag

This flag indicates the direction of communication and is the same as the value of IICSn.TRCn bit.

The following shows the operation of the main processing block during slave operation.

Start I^2C0n and wait for the communication enabled status. When communication is enabled, perform transfer using the communication mode flag and ready flag (the processing of the stop condition and start condition is performed by interrupts, conditions are confirmed by flags).

For transmission, repeat the transmission operation until the master device stops returning $\overline{\text{ACK}}$ signal. When the master device stops returning $\overline{\text{ACK}}$ signal, transfer is complete.

For reception, receive the required number of data and do not return $\overline{\text{ACK}}$ signal for the next data immediately after transfer is complete. After that, the master device generates the stop condition or restart condition. This causes exit from communications.

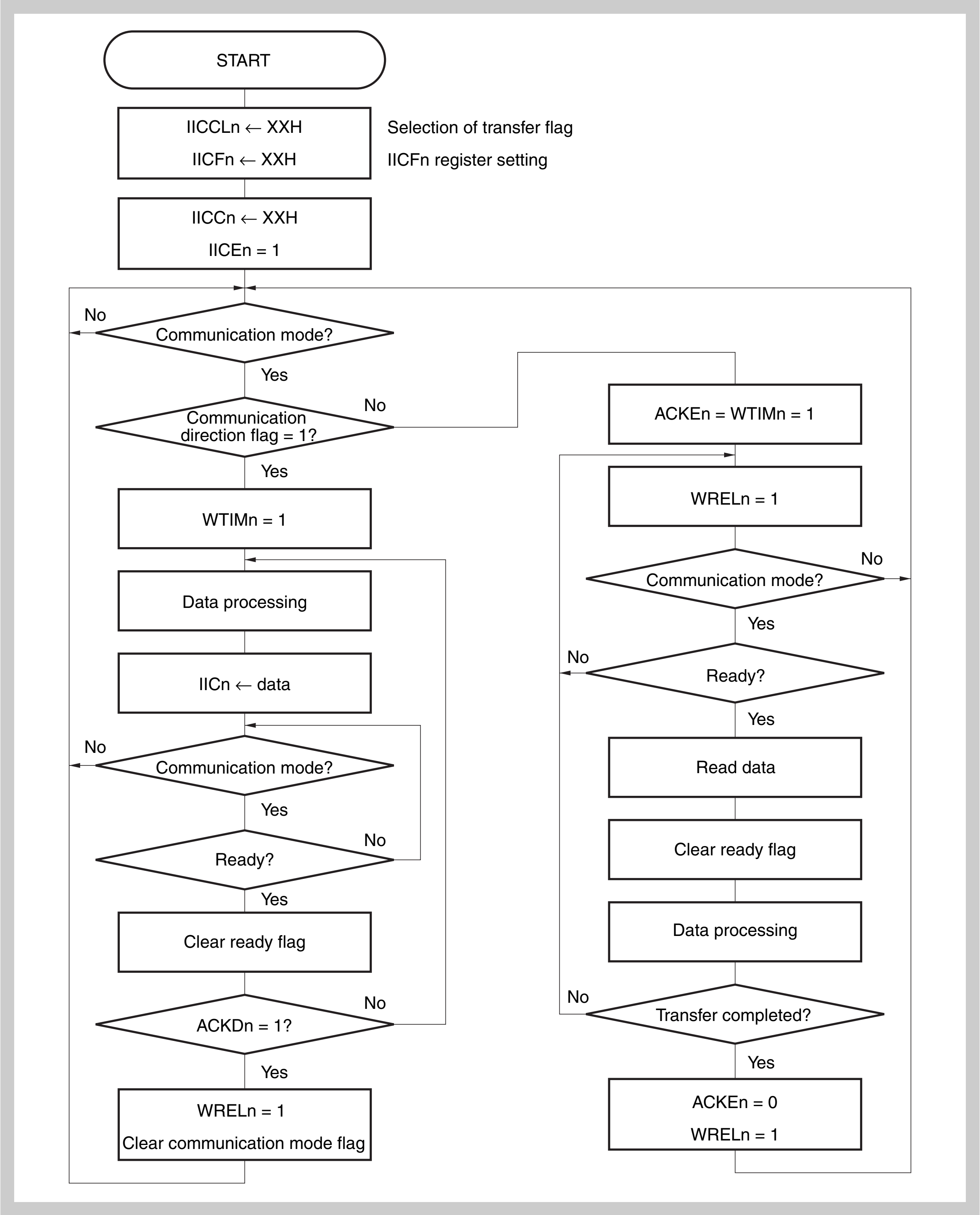


Figure 17-21 Slave operation flowchart (1)

The following shows an example of the processing of the slave device by an INTIICn interrupt (it is assumed that no extension codes are used here).

During an INTIICn interrupt, the status is confirmed and the following steps are executed.

<1> When a stop condition is detected, communication is terminated.

<2> When a start condition is detected, the address is confirmed. If the address does not match, communication is terminated. If the address matches, the communication mode is set and wait is released, and operation returns from the interrupt (the ready flag is cleared).

<3> For data transmission/reception, when the ready flag is set, operation returns from the interrupt while the IIC0n bus remains in the wait status.

Note <1> to <3> in the above correspond to <1> to <3> in *Figure 17-22*.

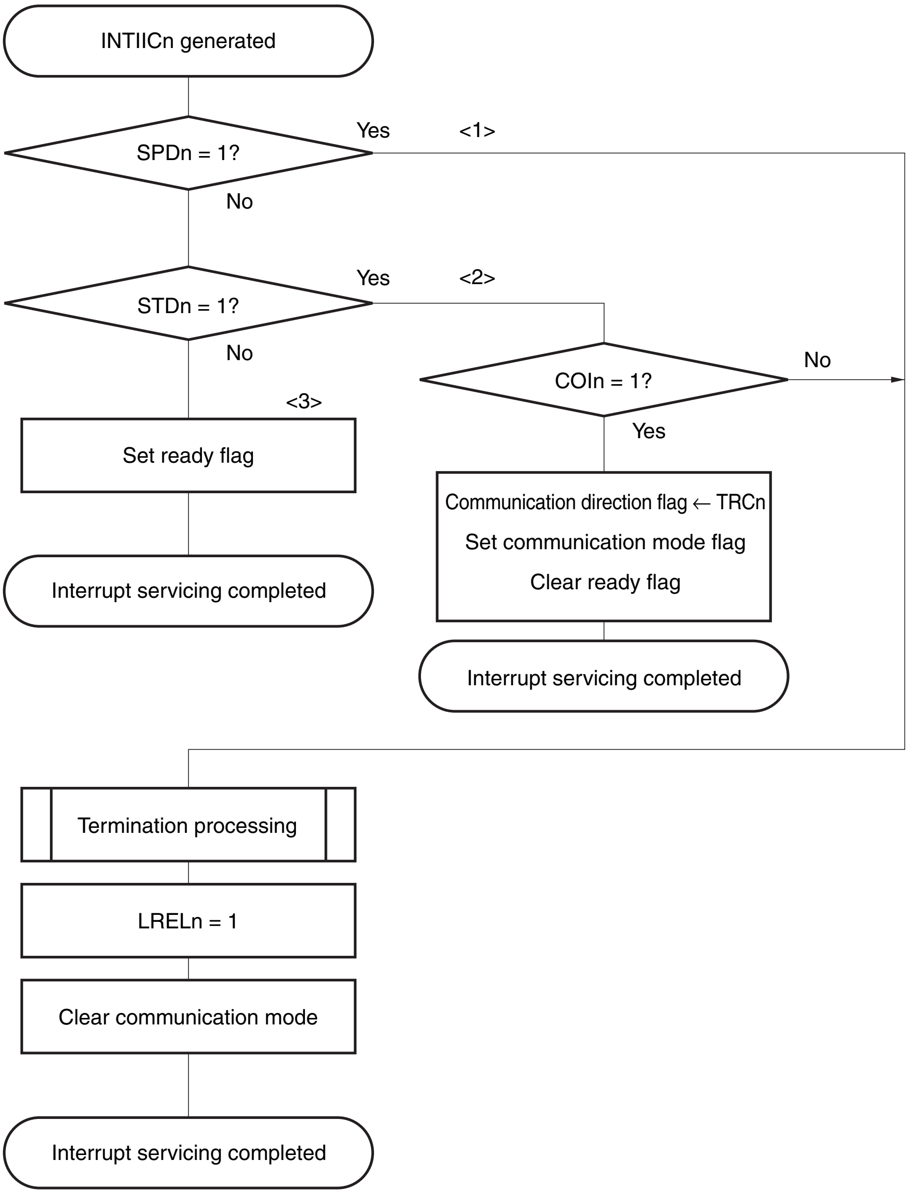


Figure 17-22 Slave operation flowchart (2)

17.19 Timing of Data Communication

When using I^{2}C bus mode, the master device outputs an address via the serial bus to select one of several slave devices as its communication partner.

After outputting the slave address, the master device transmits the IICSn.TRCn bit, which specifies the data transfer direction, and then starts serial communication with the slave device.

The shift operation of the IICn register is synchronized with the falling edge of the serial clock pin (SCLn). The transmit data is transferred to the SO latch and is output (MSB first) via the SDAn pin.

Data input via the SDAn pin is captured by the IICn register at the rising edge of the SCLn pin.

The data communication timing is shown below.

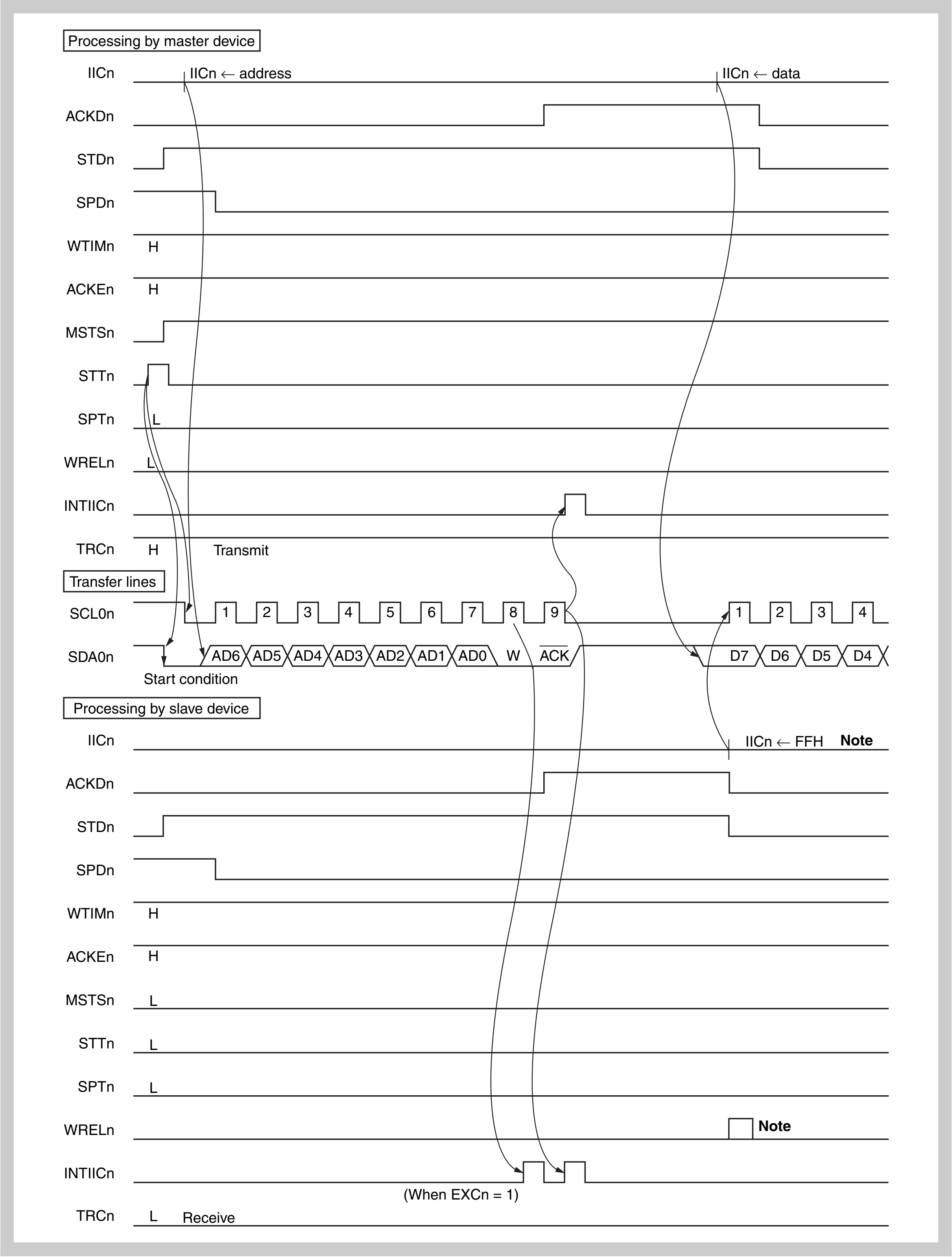


Figure 17-23 Example of master to slave communication (when 9-clock wait is selected for both master and slave) (1/3) start condition ~ address

Note To cancel slave wait, write FFH to IICn or set WRELn.

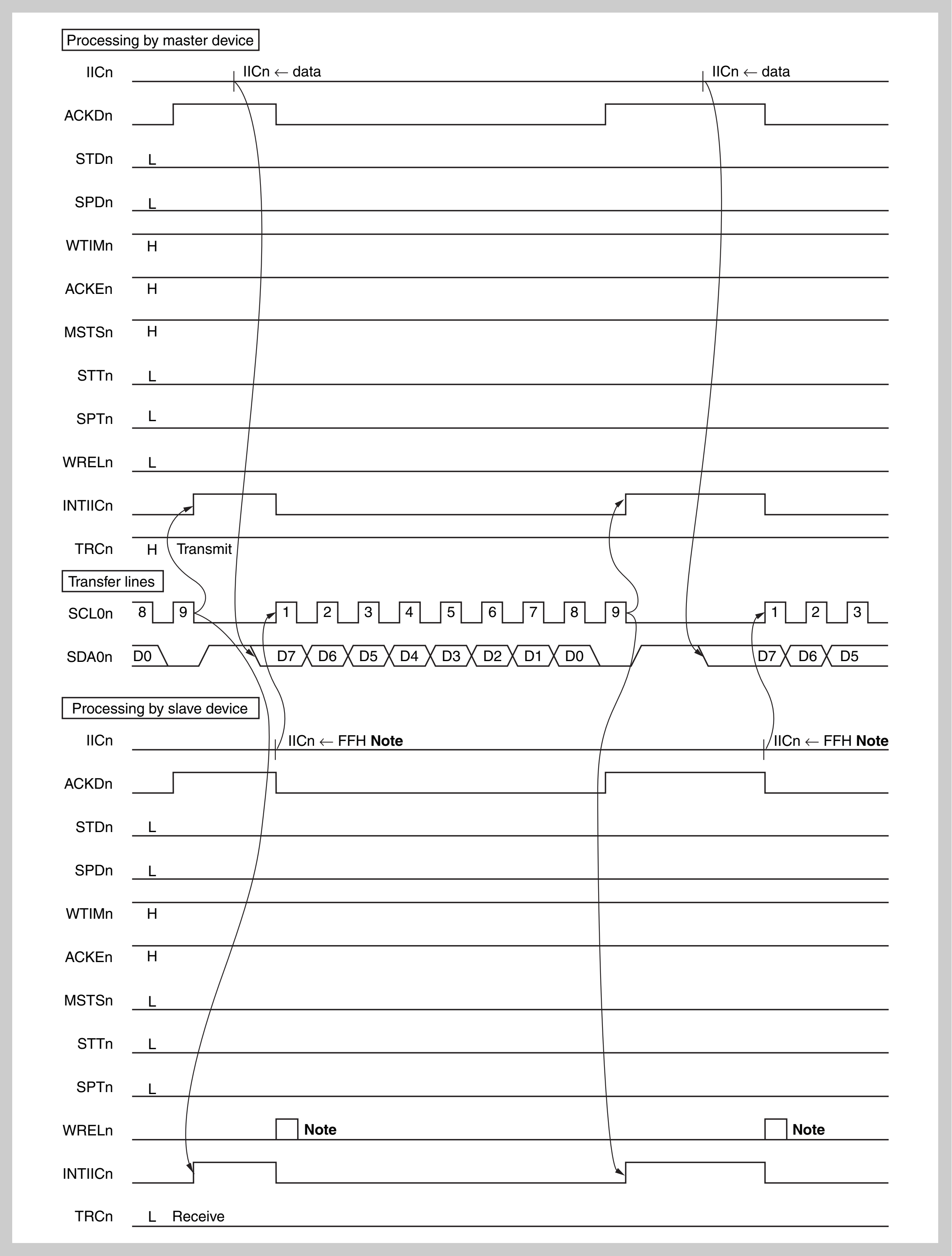


Figure 17-24 **Example of master to slave communication (when 9-clock wait is selected for both master and slave) (2/3) (b) data**

Note To cancel slave wait, write FFH to IICn or set WRELn.

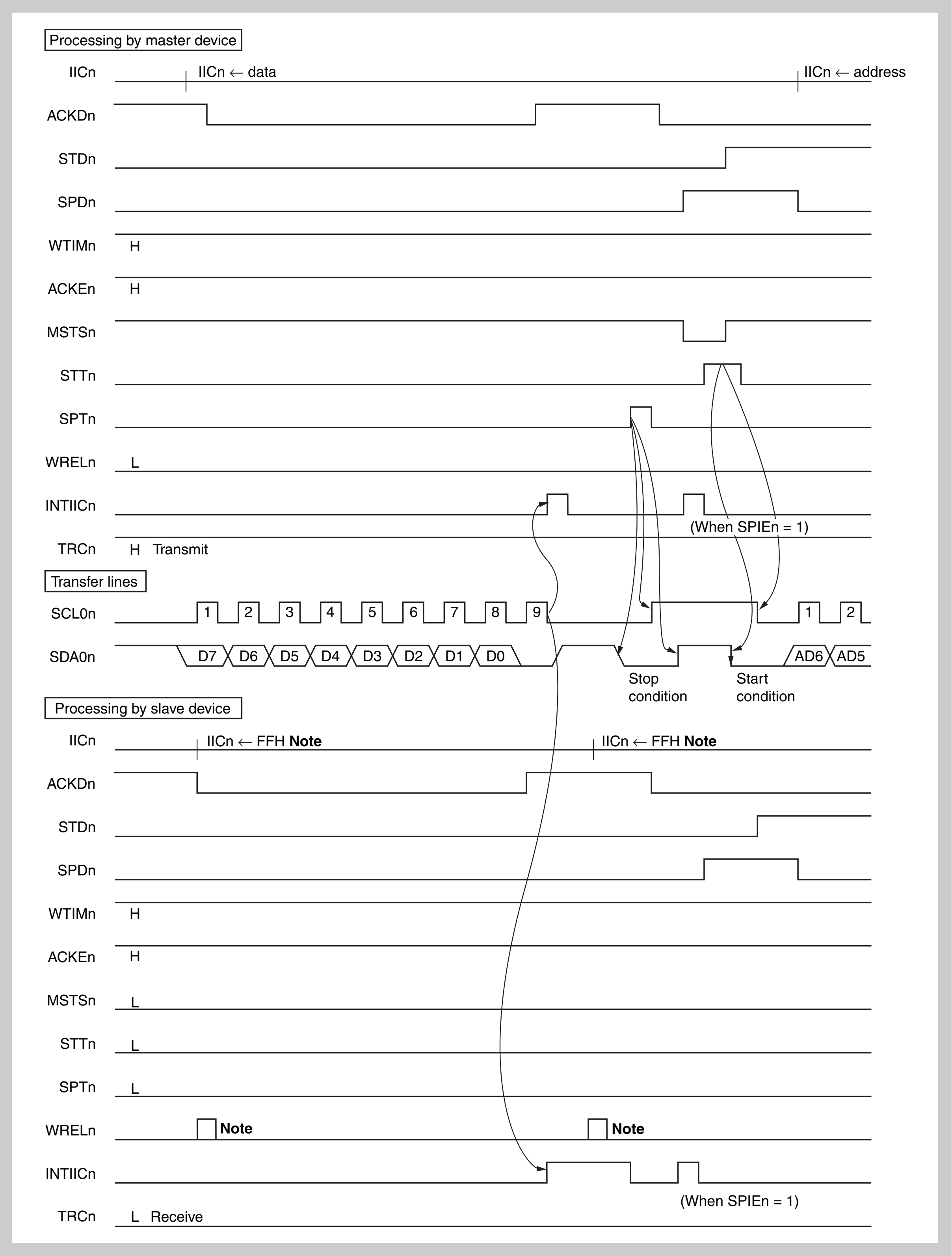


Figure 17-25 Example of master to slave communication (when 9-clock wait is selected for both master and slave) (3/3) (c) stop condition

Note To cancel slave wait, write FFH to IICn or set WRELn.

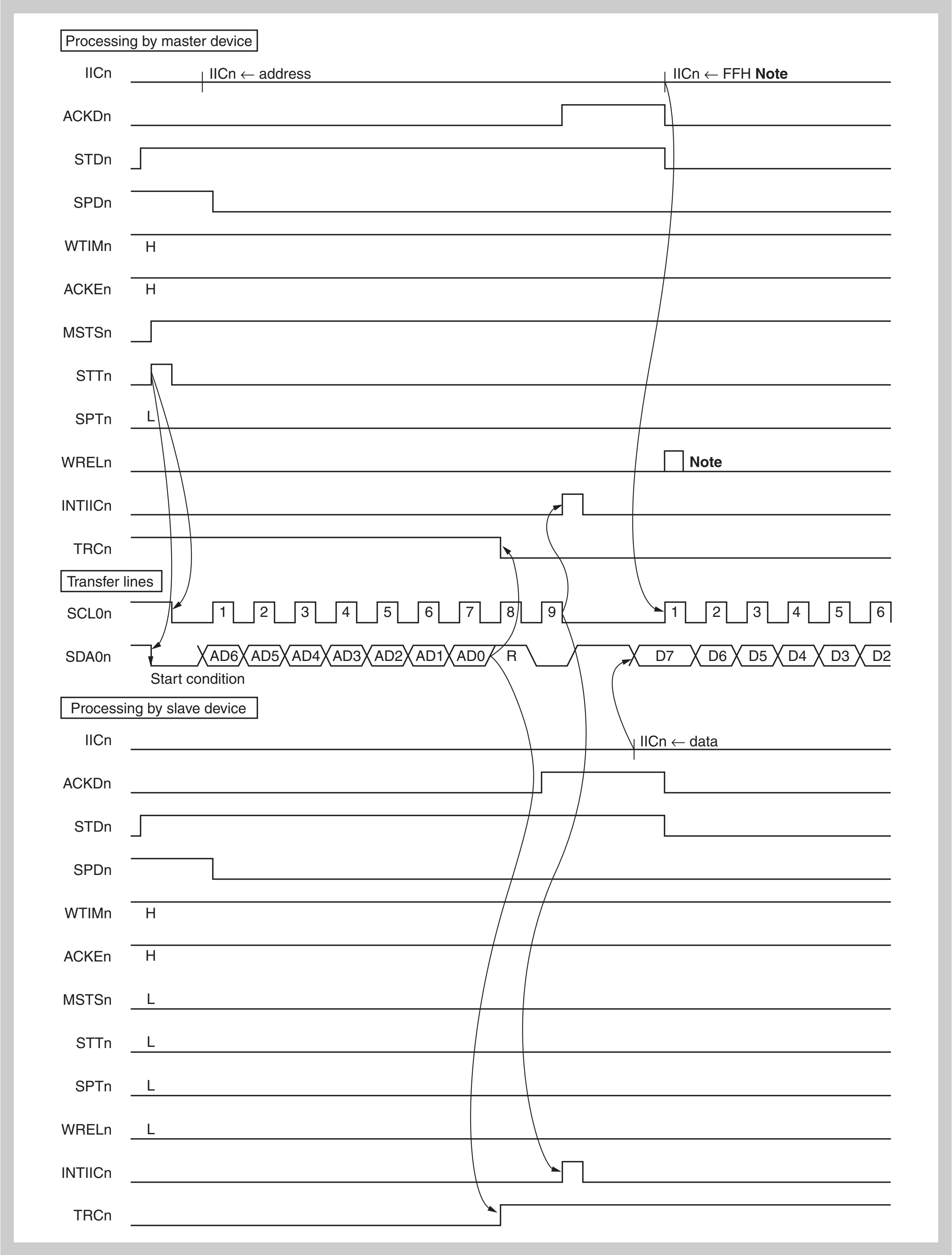


Figure 17-26 Example of slave to master communication (when 9-clock wait is selected for both master and slave) (1/3)
(a) start condition ~ address

Note To cancel master wait, write FFH to IICn or set WRELn.

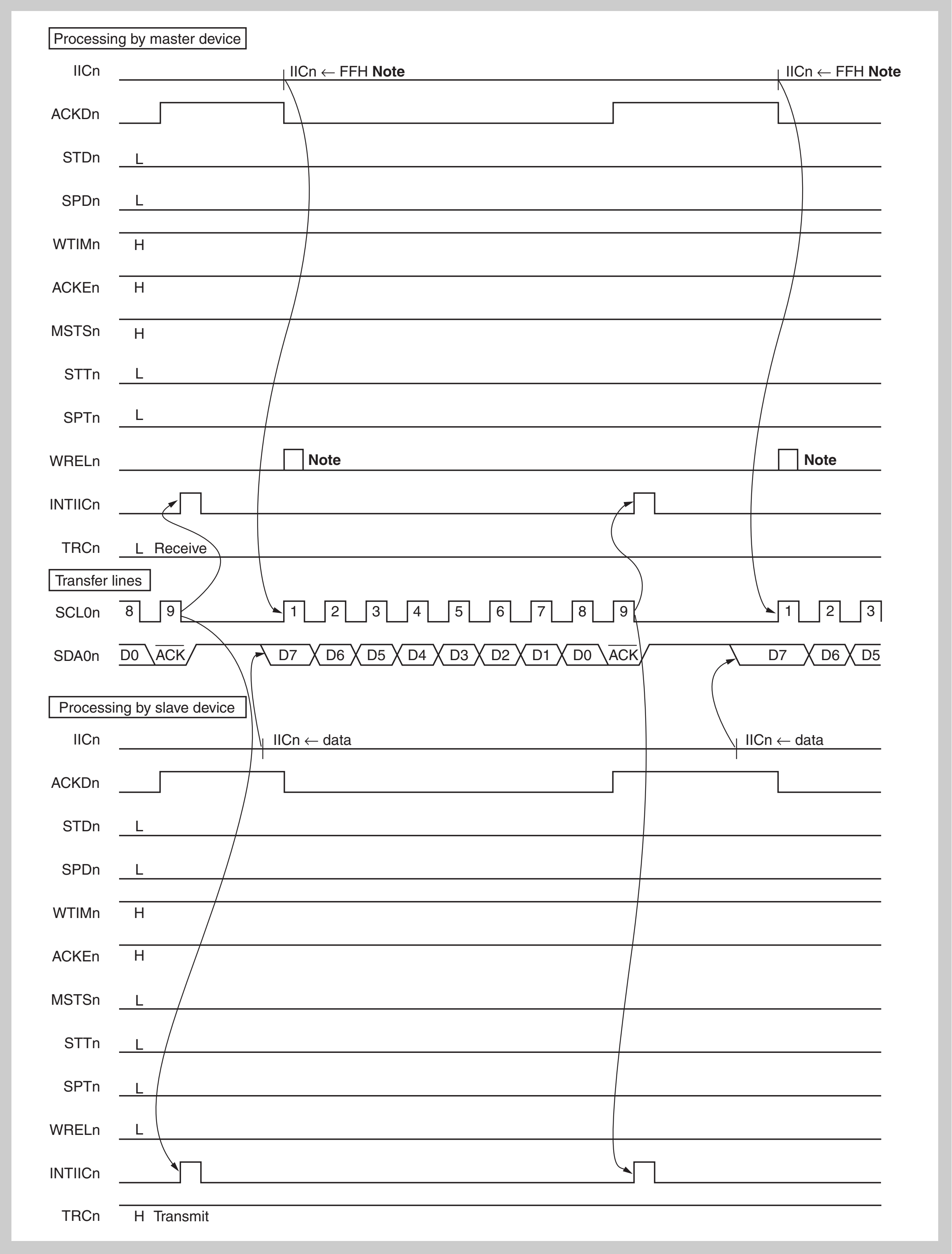


Figure 17-27 Example of slave to master communication (when 9-clock wait is selected for both master and slave) (2/3) (b) data

Note To cancel master wait, write FFH to IICn or set WRELn.

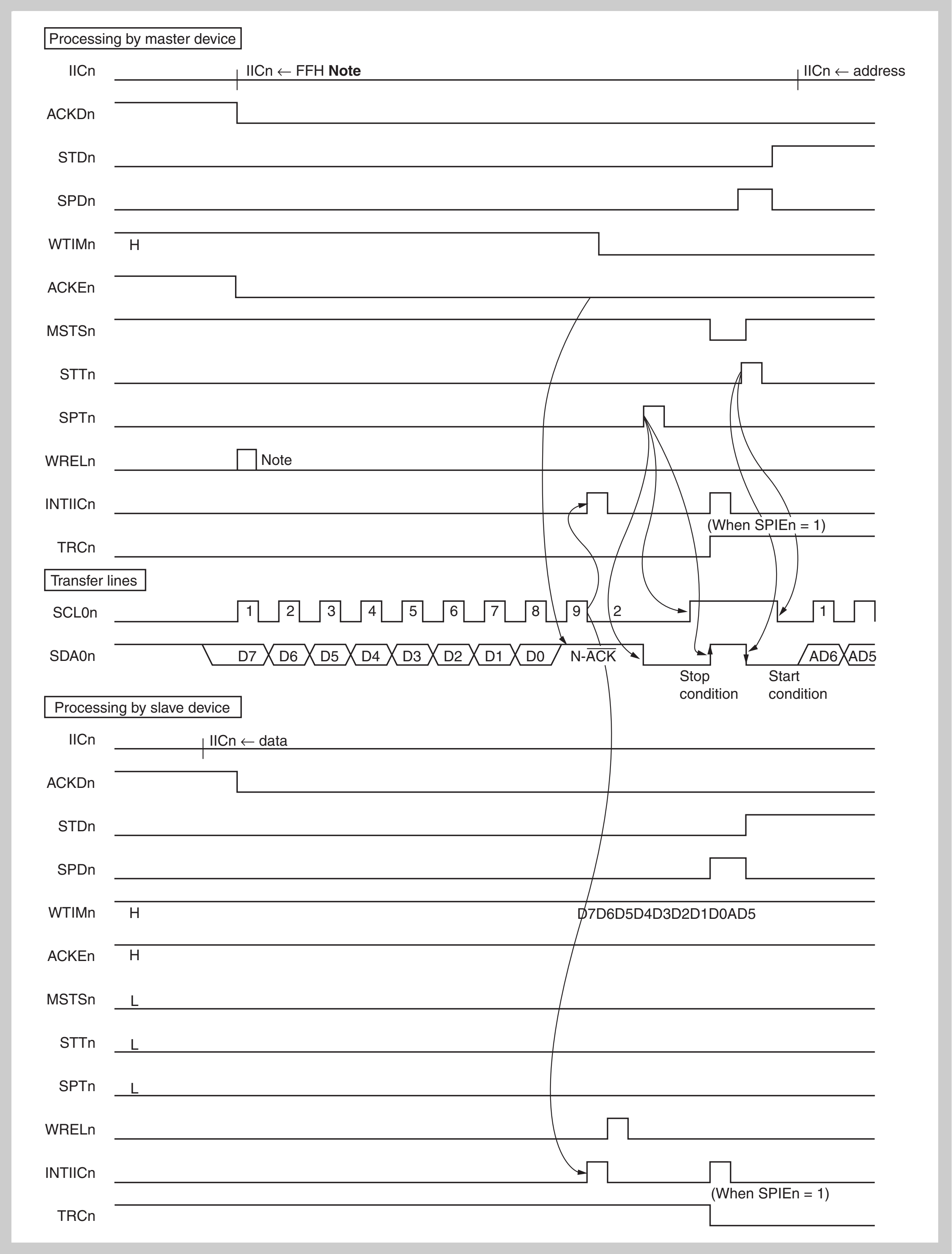


Figure 17-28 **Example of slave to master communication (when 9-clock wait is selected for both master and slave) (3/3) (c) stop condition**

Note To cancel master wait, write FFH to IICn or set WRELn.

Chapter 18 CAN Controller (CAN)

These microcontrollers feature an on-chip n-channel CAN (Controller Area Network) controller that complies with the CAN protocol as standardized in ISO 11898.

The V850E/Dx3 - DG3 microcontrollers have following number of channels of the CAN controller:

CAN	All devices
Instances	1
Names	CAN0

Note

1. Throughout this chapter, the individual CAN channels are identified by “n”, for example CANn, or CnGMCTRL for the CANn global control register.
2. Throughout this chapter, the CAN message buffer registers are identified by “m” (m = 0 to 31), for example C0MDATA4m for CAN0 message data byte 4 of message buffer register m.
3. It is recommended to configure the ports used for CAN data transmit CTXDn to its highest drive strength to Limit2 by PDSCn.PDSCnm = 1 for CAN baud rates above 200 Kbit/sec.

18.1 Features

- Compliant with ISO 11898 and tested according to ISO/DIS 16845 (CAN conformance test)
- Standard frame and extended frame transmission/reception enabled
- Transfer rate: 1 Mbps max. (if CAN clock input ≥ 8 MHz, for 32 channels)
- 32 message buffers per channel
- Receive/transmit history list function
- Automatic block transmission function
- Multi-buffer receive block function
- Mask setting of four patterns is possible for each channel
- Wake-Up capability on CAN receive data pins CRXDn
- Data bit time, communication baud rate and sample point can be controlled by CAN module bit-rate prescaler register (CnBRP) and bit rate register (CnBTR)
  - As an example the following sample-point configurations can be configured:
  - 66.7%, 70.0%, 75.0%, 80.0%, 81.3%, 85.0%, 87.5%
  - Baudrates in the range of 10 kbps up to 1000 kbps can be configured
- Enhanced features:
  - Each message buffer can be configured to operate as a transmit or a receive message buffer
  - Transmission priority is controlled by the identifier or by mailbox number (selectable)
  - A transmission request can be aborted by clearing the dedicated Transmit-Request flag of the concerned message buffer.
  - Automatic block transmission operation mode (ABT)
  - Time stamp function for CAN0 in collaboration with timer Timer G0 capture channel

18.1.1 Overview of functions

Table 18-1 presents an overview of the CAN Controller functions.

Table 18-1 Overview of functions

Function	Details
Protocol	CAN protocol ISO 11898 (standard and extended frame transmission/reception)
Baud rate	Maximum 1 Mbps (CAN clock input ≥ 8 MHz)
Data storage	Storing messages in the CAN RAM
Number of messages	• 32 message buffers per channel • Each message buffer can be set to be either a transmit message buffer or a receive message buffer.
Message reception	• Unique ID can be set to each message buffer. • Mask setting of four patterns is possible for each channel. • A receive completion interrupt is generated each time a message is received and stored in a message buffer. • Two or more receive message buffers can be used as a FIFO receive buffer (multi-buffer receive block function). • Receive history list function
Message transmission	**Unique ID can be set to each message buffer.** • Transmit completion interrupt for each message buffer • Message buffer number 0 to 7 specified as the transmit message buffer can be set for automatic block transfer. Message transmission interval is programmable (automatic block transmission function (hereafter referred to as “ABT”)). • Transmission history list function
Remote frame processing	Remote frame processing by transmit message buffer
Time stamp function	• The time stamp function can be set for a message reception when a 16-bit timer is used in combination. • Time stamp capture trigger can be selected (SOF or EOF in a CAN message frame can be detected.). • The time stamp function can be set for a transmit message.
Diagnostic function	• Readable error counters • “Valid protocol operation flag” for verification of bus connections • Receive-only mode • Single-shot mode • CAN protocol error type decoding • Self-test mode
Release from bus-off state	• Forced release from bus-off (by ignoring timing constraint) possible by software. • No automatic release from bus-off (software must re-enable).
Power save mode	• CAN Sleep mode (can be woken up by CAN bus) • CAN Stop mode (cannot be woken up by CAN bus)

18.1.2 Configuration

The CAN Controller is composed of the following four blocks.

- NPB interface
  This functional block provides an NPB (Peripheral I/O Bus) interface and means of transmitting and receiving signals between the CAN module and the host CPU.
- MCM (Message Control Module)
  This functional block controls access to the CAN protocol layer and to the CAN RAM within the CAN module.
- CAN protocol layer
  This functional block is involved in the operation of the CAN protocol and its related settings.
- CAN RAM
  This is the CAN memory functional block, which is used to store message IDs, message data, etc.

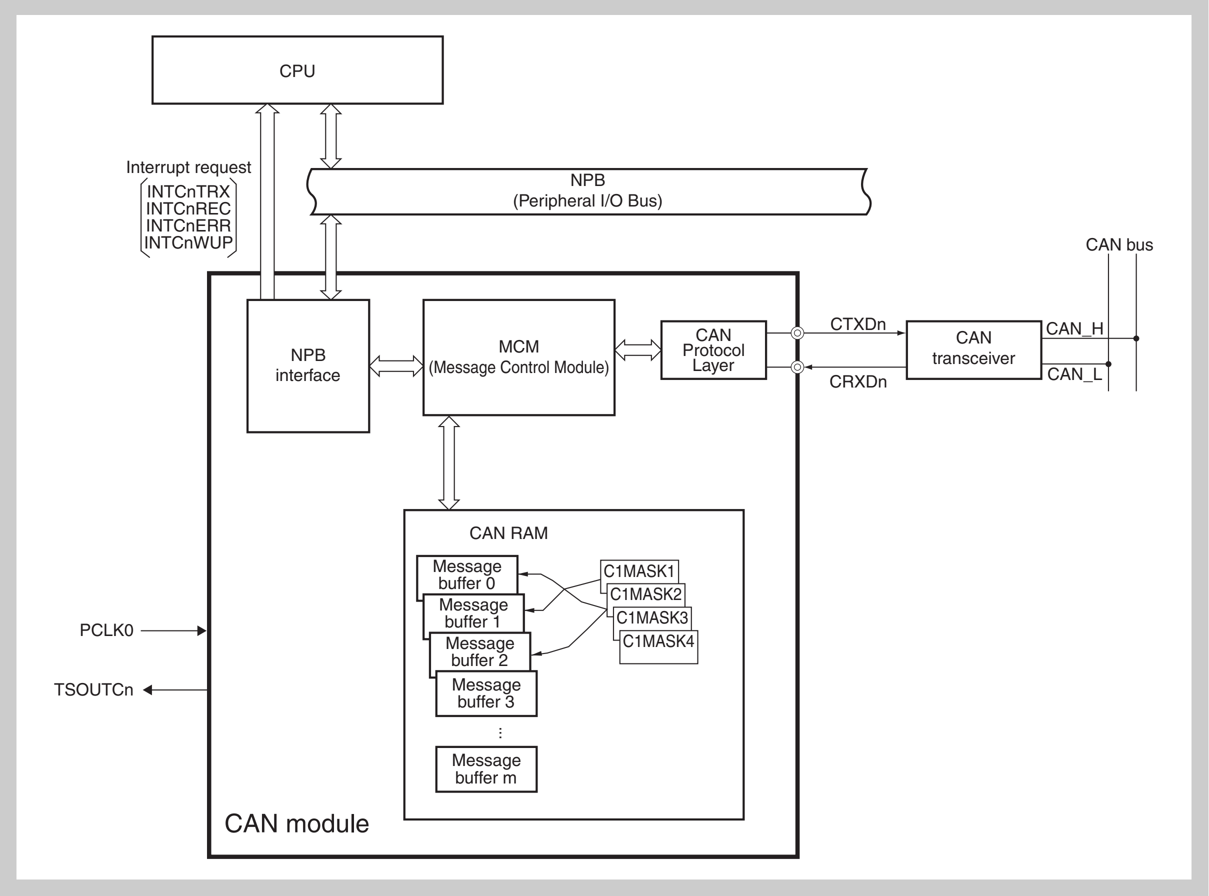


Figure 18-1 Block diagram of CAN module

18.2 CAN Protocol

CAN (Controller Area Network) is a high-speed multiplex communication protocol for real-time communication in automotive applications (class C). CAN is prescribed by ISO 11898. For details, refer to the ISO 11898 specifications.

The CAN specification is generally divided into two layers: a physical layer and a data link layer. In turn, the data link layer includes logical link and medium access control. The composition of these layers is illustrated below.

Higher	Data link layer[Note]	• Logical link control (LLC)	• Acceptance filtering • Overload report • Recovery management
↕		• Medium access control (MAC)	• Data capsuled/not capsuled • Frame coding (stuffing/no stuffing) • Medium access management • Error detection • Error report • Acknowledgement • Seriated/not seriated
Lower	Physical layer		Prescription of signal level and bit description

Figure 18-2 Composition of layers

Note CAN Controller specification

18.2.1 Frame format

(1) Standard format frame

- The standard format frame uses 11-bit identifiers, which means that it can handle up to 2,048 messages.

(2) Extended format frame

- The extended format frame uses 29-bit (11 bits + 18 bits) identifiers, which increases the number of messages that can be handled to 2,048 × 2^{18} messages.
- An extended format frame is set when “recessive level” (CMOS level of “1”) is set for both the SRR and IDE bits in the arbitration field.

18.2.2 Frame types

The following four types of frames are used in the CAN protocol.

Table 18-2 Frame types

Frame Type	Description
Data frame	Frame used to transmit data
Remote frame	Frame used to request a data frame
Error frame	Frame used to report error detection
Overload frame	Frame used to delay the next data frame or remote frame

(1) Bus value

The bus values are divided into dominant and recessive.

- Dominant level is indicated by logical 0.
- Recessive level is indicated by logical 1.
- When a dominant level and a recessive level are transmitted simultaneously, the bus value becomes dominant level.

18.2.3 Data frame and remote frame

(1) Data frame

A data frame is composed of seven fields.

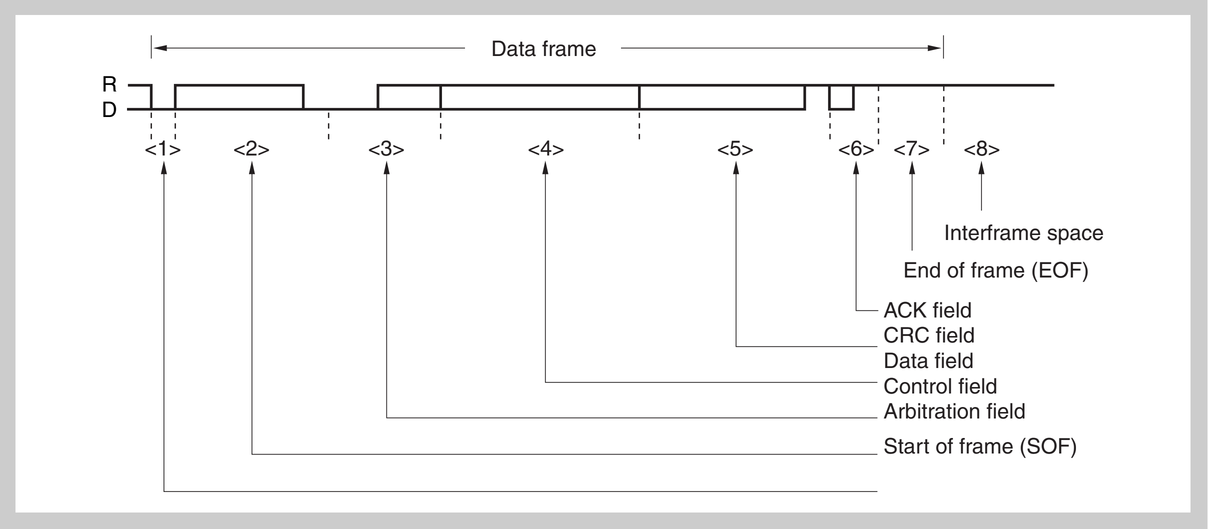


Figure 18-3 Data frame

Note D: Dominant = 0
R: Recessive = 1

(2) Remote frame

A remote frame is composed of six fields.

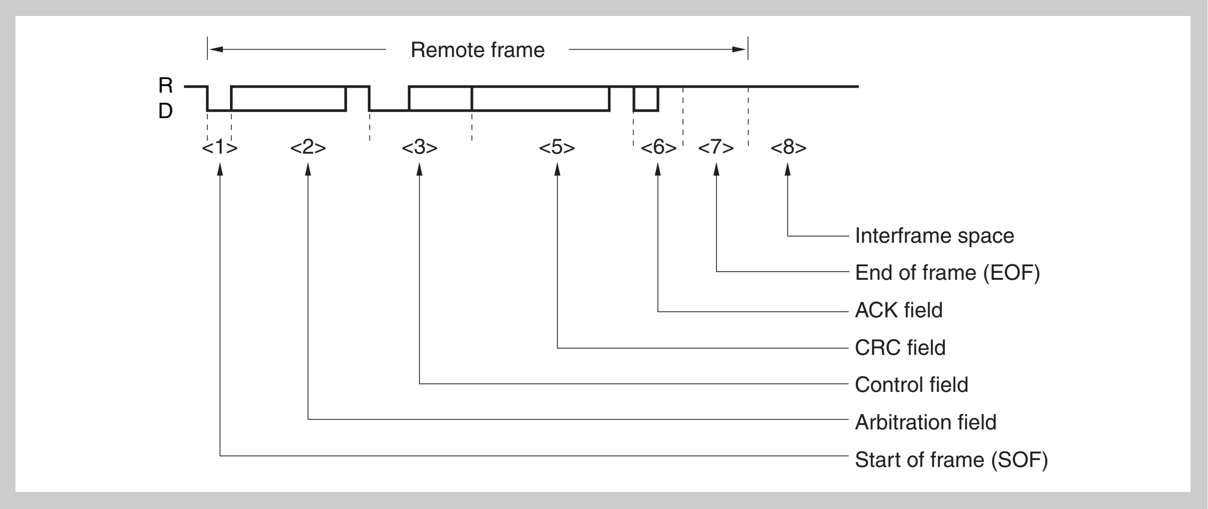


Figure 18-4 Remote frame

Note 1. The data field is not transferred even if the control field’s data length code is not “0000_B”.

2. D: Dominant = 0
R: Recessive = 1

(3) Description of fields

(a) Start of frame (SOF)

The start of frame field is located at the start of a data frame or remote frame.

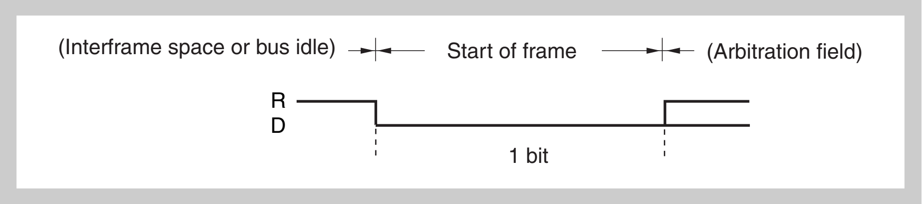


Figure 18-5 Start of frame (SOF)

Note D: Dominant = 0
R: Recessive = 1

- If dominant level is detected in the bus idle state, a hard-synchronization is performed (the current TQ is assigned to be the SYNC segment).
- If dominant level is sampled at the sample point following such a hard-synchronization, the bit is assigned to be a SOF. If recessive level is detected, the protocol layer returns to the bus idle state and regards the preceding dominant pulse as a disturbance only. No error frame is generated in such case.

(b) Arbitration field

The arbitration field is used to set the priority, data frame/remote frame, and frame format.

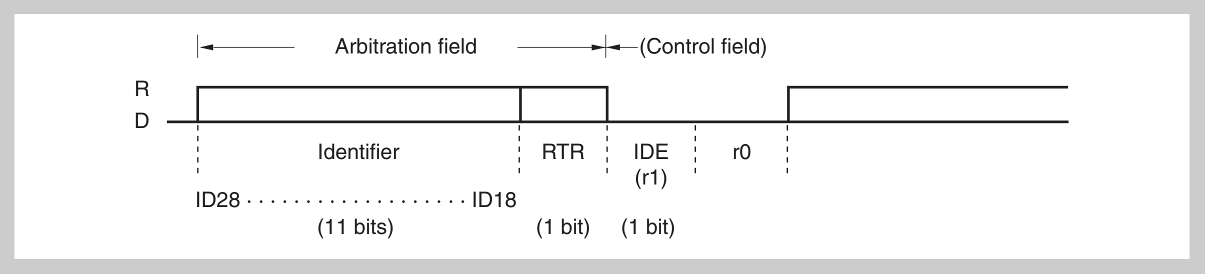


Figure 18-6 Arbitration field (in standard format mode)

Caution 1. ID28 to ID18 are identifiers.

2. An identifier is transmitted MSB first.

Note D: Dominant = 0
R: Recessive = 1

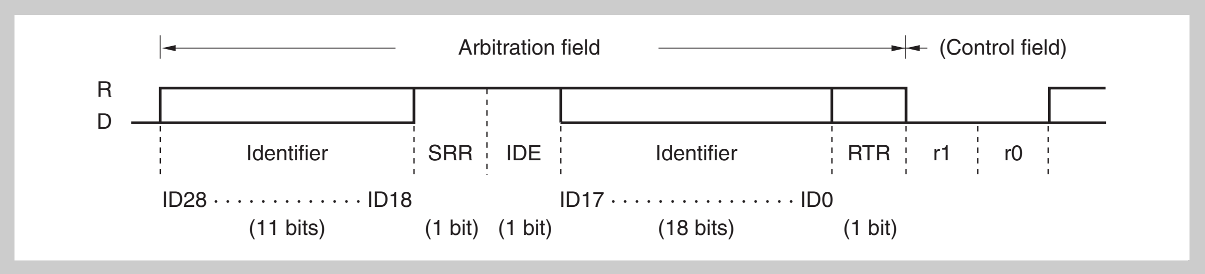


Figure 18-7 Arbitration field (in extended format mode)

Caution 1. ID28 to ID18 are identifiers.

2. An identifier is transmitted MSB first.

Note D: Dominant = 0
R: Recessive = 1

Table 18-3 RTR frame settings

Frame type	RTR bit
Data frame	0 (D)
Remote frame	1 (R)

Table 18-4 Frame format setting (IDE bit) and number of identifier (ID) bits

Frame format	SRR bit	IDE bit	Number of bits
Standard format mode	None	0 (D)	11 bits
Extended format mode	1 (R)	1 (R)	29 bits

(c) Control field

The control field sets “DLC” as the number of data bytes in the data field (DLC = 0 to 8).

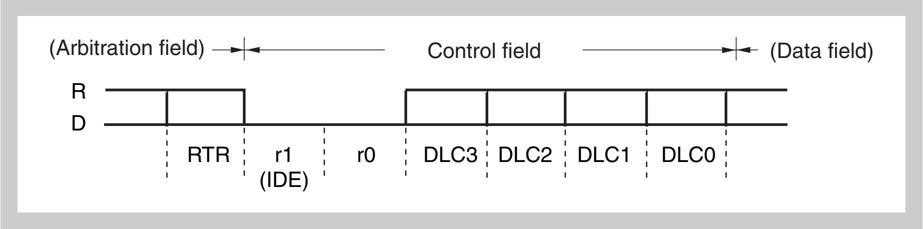


Figure 18-8 Control field

Note D: Dominant = 0
R: Recessive = 1

In a standard format frame, the control field’s IDE bit is the same as the r1 bit.

Table 18-5 Data length setting

Data length code				Data byte count
DLC3	DLC2	DLC1	DLC0	
0	0	0	0	0 bytes
0	0	0	1	1 byte
0	0	1	0	2 bytes
0	0	1	1	3 bytes
0	1	0	0	4 bytes
0	1	0	1	5 bytes
0	1	1	0	6 bytes
0	1	1	1	7 bytes
1	0	0	0	8 bytes
Other than above				8 bytes regardless of the value of DLC3 to DLC0

Caution In the remote frame, there is no data field even if the data length code is not 0000_B.

(d) Data field

The data field contains the amount of data (byte units) set by the control field. Up to 8 units of data can be set.

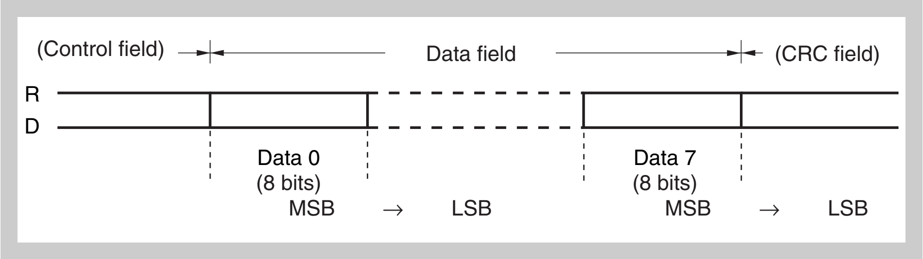


Figure 18-9 Data field

Note D: Dominant = 0
R: Recessive = 1

(e) CRC field

The CRC field is a 16-bit field that is used to check for errors in transmit data.

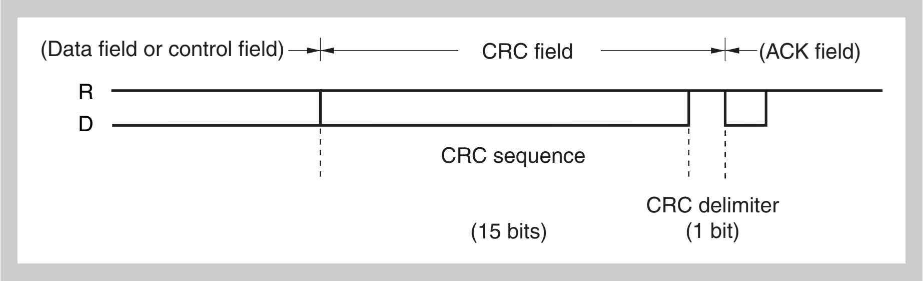


Figure 18-10 CRC field

Note D: Dominant = 0
R: Recessive = 1

- The polynomial P(X) used to generate the 15-bit CRC sequence is expressed as follows.

  $P(X) = X^{15} + X^{14} + X^{10} + X^{8} + X^{7} + X^{4} + X^{3} + 1$

- Transmitting node: Transmits the CRC sequence calculated from the data (before bit stuffing) in the start of frame, arbitration field, control field, and data field.

- Receiving node: Compares the CRC sequence calculated using data bits that exclude the stuffing bits in the receive data with the CRC sequence in the CRC field. If the two CRC sequences do not match, the node issues an error frame.

(f) ACK field

The ACK field is used to acknowledge normal reception.

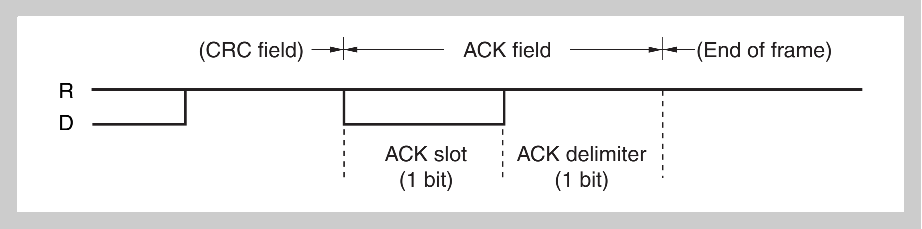


Figure 18-11 ACK field

Note D: Dominant = 0
R: Recessive = 1

- If no CRC error is detected, the receiving node sets the ACK slot to the dominant level.
- The transmitting node outputs two recessive-level bits.

(g) End of frame (EOF)

The end of frame field indicates the end of data frame/remote frame.

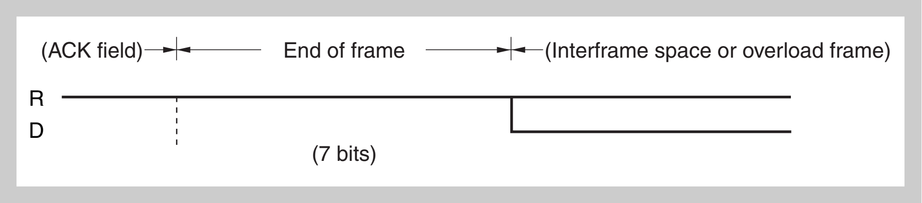


Figure 18-12 End of frame (EOF)

Note D: Dominant = 0
R: Recessive = 1

(h) Interframe space

The interframe space is inserted after a data frame, remote frame, error frame, or overload frame to separate one frame from the next.

- The bus state differs depending on the error status.
  - **Error active node**

    The interframe space consists of a 3-bit intermission field and a bus idle field.

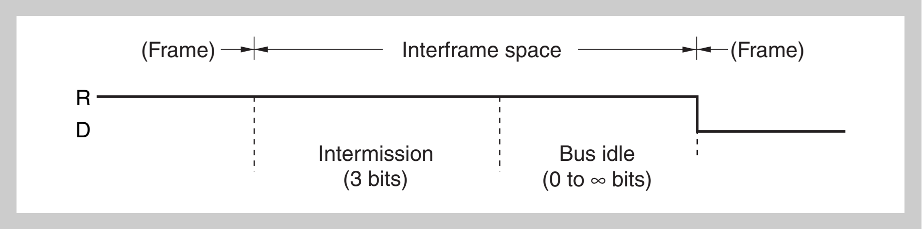


Figure 18-13 Interframe space (error active node)

Note 1. Bus idle: State in which the bus is not used by any node.

2. D: Dominant = 0
R: Recessive = 1

– **Error passive node**

The interframe space consists of an intermission field, a suspend transmission field, and a bus idle field.

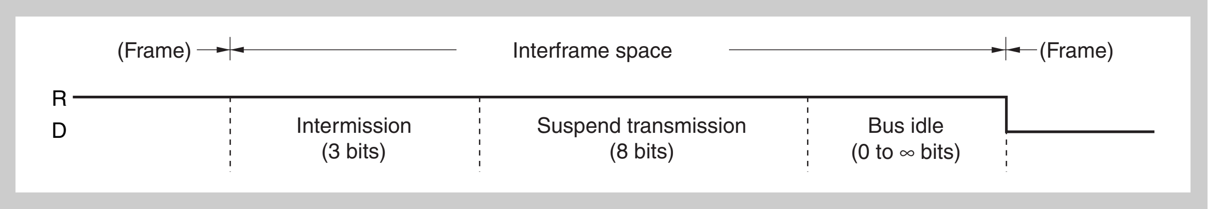


Figure 18-14 Interframe space (error passive node)

Note 1. Bus idle: State in which the bus is not used by any node.
Suspend transmission: Sequence of 8 recessive-level bits transmitted from the node in the error passive status.

2. D: Dominant = 0
R: Recessive = 1

Usually, the intermission field is 3 bits. If the transmitting node detects a dominant level at the third bit of the intermission field, however, it executes transmission.

- Operation in error status

Table 18-6 Operation in error status

Error status	Operation
Error active	A node in this status can transmit immediately after a 3-bit intermission.
Error passive	A node in this status can transmit 8 bits after the intermission.

18.2.4 Error frame

An error frame is output by a node that has detected an error.

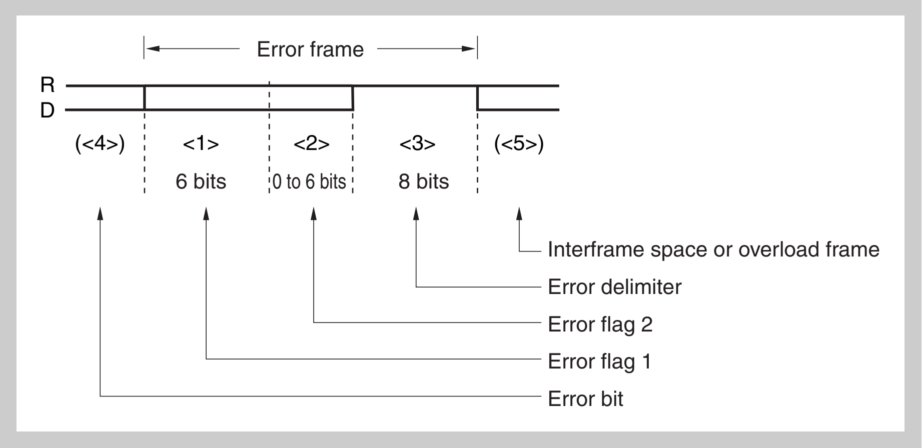


Figure 18-15 Error frame

Note D: Dominant = 0
R: Recessive = 1

Table 18-7 Definition of error frame fields

No.	Name	Bit count	Definition
<1>	Error flag 1	6	Error active node: Outputs 6 dominant-level bits consecutively. Error passive node: Outputs 6 recessive-level bits consecutively. If another node outputs a dominant level while one node is outputting a passive error flag, the passive error flag is not cleared until the same level is detected 6 bits in a row.
<2>	Error flag 2	0 to 6	Nodes receiving error flag 1 detect bit stuff errors and issues this error flag.
<3>	Error delimiter	8	Outputs 8 recessive-level bits consecutively. If a dominant level is detected at the 8th bit, an overload frame is transmitted from the next bit.
<4>	Error bit	–	The bit at which the error was detected. The error flag is output from the bit next to the error bit. In the case of a CRC error, this bit is output following the ACK delimiter.
<5>	Interframe space/ overload frame	–	An interframe space or overload frame starts from here.

18.2.5 Overload frame

An overload frame is transmitted under the following conditions.

- When the receiving node has not completed the reception operation
- If a dominant level is detected at the first two bits during intermission
- If a dominant level is detected at the last bit (7th bit) of the end of frame or at the last bit (8th bit) of the error delimiter/overload delimiter

Note The CAN is internally fast enough to process all received frames not generating overload frames.

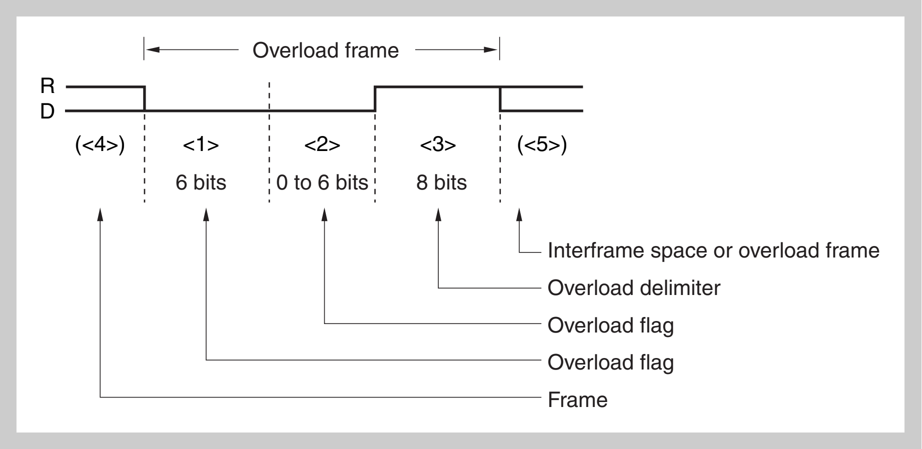


Figure 18-16 Overload frame

Note D: Dominant = 0
R: Recessive = 1

Table 18-8 Definition of overload frame fields

No	Name	Bit count	Definition
<1>	Overload flag	6	Outputs 6 dominant-level bits consecutively.
<2>	Overload flag from other node	0 to 6	The node that received an overload flag in the interframe space outputs an overload flag.
<3>	Overload delimiter	8	Outputs 8 recessive-level bits consecutively. If a dominant level is detected at the 8th bit, an overload frame is transmitted from the next bit.
<4>	Frame	–	Output following an end of frame, error delimiter, or overload delimiter.
<5>	Interframe space/overload frame	–	An interframe space or overload frame starts from here.

18.3 Functions

18.3.1 Determining bus priority

(1) When a node starts transmission:

- During bus idle, the node that output data first transmits the data.

(2) When more than one node starts transmission:

- The node that consecutively outputs the dominant level for the longest from the first bit of the arbitration field has the bus priority (if a dominant level and a recessive level are simultaneously transmitted, the dominant level is taken as the bus value).
- The transmitting node compares its output arbitration field and the data level on the bus.

Table 18-9 Determining bus priority

Level match	Continuous transmission
Level mismatch	Stops transmission at the bit where mismatch is detected and starts reception at the following bit

(3) Priority of data frame and remote frame

- When a data frame and a remote frame are on the bus, the data frame has priority because its RTR bit, the last bit in the arbitration field, carries a dominant level.

Note If the extended-format data frame and the standard-format remote frame conflict on the bus (if ID28 to ID18 of both of them are the same), the standard-format remote frame takes priority.

18.3.2 Bit stuffing

Bit stuffing is used to establish synchronization by appending 1 bit of inverted-level data if the same level continues for 5 bits, in order to prevent a burst error.

Table 18-10 Bit stuffing

Transmission	During the transmission of a data frame or remote frame, when the same level continues for 5 bits in the data between the start of frame and the ACK field, 1 inverted-level bit of data is inserted before the following bit.
Reception	During the reception of a data frame or remote frame, when the same level continues for 5 bits in the data between the start of frame and the ACK field, reception is continued after deleting the next bit.

18.3.3 Multi masters

As the bus priority (a node acquiring transmit functions) is determined by the identifier, any node can be the bus master.

18.3.4 Multi cast

Although there is one transmitting node, two or more nodes can receive the same data at the same time because the same identifier can be set to two or more nodes.

18.3.5 CAN sleep mode/CAN stop mode function

The CAN sleep mode/CAN stop mode function puts the CAN Controller in waiting mode to achieve low power consumption.

The controller is woken up from the CAN sleep mode by bus operation but it is not woken up from the CAN stop mode by bus operation (the CAN stop mode is controlled by CPU access).

18.3.6 Error control function

(1) Error types

Table 18-11 Error types

Type	Description of error		Detection state	
	Detection method	**Detection condition**	**Transmission/ reception**	**Field/frame**
Bit error	Comparison of the output level and level on the bus (except stuff bit)	Mismatch of levels	Transmitting/ receiving node	Bit that is outputting data on the bus at the start of frame to end of frame, error frame and overload frame.
Stuff error	Check of the receive data at the stuff bit	6 consecutive bits of the same output level	Receiving node	Start of frame to CRC sequence
CRC error	Comparison of the CRC sequence generated from the receive data and the received CRC sequence	Mismatch of CRC	Receiving node	CRC field
Form error	Field/frame check of the fixed format	Detection of fixed format violation	Receiving node	CRC delimiter ACK field End of frame Error frame Overload frame
ACK error	Check of the ACK slot by the transmitting node	Detection of recessive level in ACK slot	Transmitting node	ACK slot

(2) Output timing of error frame

Table 18-12 Output timing of error frame

Type	Output timing
Bit error, stuff error, form error, ACK error	Error frame output is started at the timing of the bit following the detected error.
CEC error	Error frame output is started at the timing of the bit following the ACK delimiter.

(3) Processing in case of error

The transmission node re-transmits the data frame or remote frame after the error frame. (However, it does not re-transmit the frame in the single-shot mode.)

(4) Error state

(a) Types of error states

The following three types of error states are defined by the CAN specification:

- Error active
- Error passive
- Bus-off

These types of error states are classified by the values of the TEC7 to TEC0 bits (transmission error counter bits) and the REC6 to REC0 bits (reception error counter bits) as shown in *Table 18-13*.

The present error state is indicated by the CAN module information register (CnINFO).

When each error counter value becomes equal to or greater than the error warning level (96), the TECS0 or RECS0 bit of the CnINFO register is set to 1. In this case, the bus state must be tested because it is considered that the bus has a serious fault. An error counter value of 128 or more indicates an error passive state and the TECS1 or RECS1 bit of the CnINFO register is set to 1.

- If the value of the transmission error counter is greater than or equal to 256 (actually, the transmission error counter does not indicate a value greater than or equal to 256), the bus-off state is reached and the BOFF bit of the CnINFO register is set to 1.
- If only one node is active on the bus at startup (i.e., a particular case such as when the bus is connected only to the local station), ACK is not returned even if data is transmitted. Consequently, re-transmission of the error frame and data is repeated. In the error passive state, however, the transmission error counter is not incremented and the bus-off state is not reached.

Table 18-13 Types of error states

Type	Operation	Value of error counter	Indication of CnINFO register	Operation specific to error state
Error active	Transmission	0 to 95	TECS1, TECS0 = 00	Outputs an active error flag (6 consecutive dominant-level bits) on detection of the error.
	Reception	0 to 95	RECS1, RECS0 = 00	
	Transmission	96 to 127	TECS1, TECS0 = 01	
	Reception	96 to 127	RECS1, RECS0 = 01	
Error passive	Transmission	128 to 255	TECS1, TECS0 = 11	Outputs a passive error flag (6 consecutive recessive-level bits) on detection of the error. Transmits 8 recessive-level bits, in between transmissions, following an intermission (suspend transmission).
	Reception	128 or more	RECS1, RECS0 = 11	
Bus-off	Transmission	256 or more (not indicated)Note	BOFF = 1, TECS1, TECS0 = 11	Communication is not possible. Messages are not stored when receiving frames, however, the following operations of <1>, <2>, and <3> are done. <1> TSOUT toggles. <2> REC is incremented/decremented. <3> VALID bit is set. If the CAN module is entered to the initialization mode and then transition request to any operation mode is made, and when 11 consecutive recessive-level bits are detected 128 times, the error counter is reset to 0 and the error active state can be restored.

Note The value of the transmission error counter (TEC) is invalid when the BOFF bit is set to 1. If an error that increments the value of the transmission error counter by +8 while the counter value is in a range of 248 to 255, the counter is not incremented and the bus-off state is assumed.

(b) Error counter

The error counter counts up when an error has occurred, and counts down upon successful transmission and reception. The error counter is updated immediately after error detection.

Table 18-14 Error counter

State	Transmission error counter (TEC7 to TEC0 bits)	Reception error counter (REC6 to REC0 bits)
Receiving node detects an error (except bit error in the active error flag or overload flag).	No change	+1 (when REPS = 0)
Receiving node detects dominant level following error flag of error frame.	No change	+8 (when REPS = 0)
Transmitting node transmits an error flag. [As exceptions, the error counter does not change in the following cases.] <1> ACK error is detected in error passive state and dominant level is not detected while the passive error flag is being output. <2> A stuff error is detected in an arbitration field that transmitted a recessive level as a stuff bit, but a dominant level is detected.	+8	No change
Bit error detection while active error flag or overload flag is being output (error-active transmitting node)	+8	No change
Bit error detection while active error flag or overload flag is being output (error-active receiving node)	No change	+8 (REPS bit = 0)
When the node detects 14 consecutive dominant-level bits from the beginning of the active error flag or overload flag, and then subsequently detects 8 consecutive dominant-level bits. When the node detects 8 consecutive dominant levels after a passive error flag	+8 (transmitting)	+8 (during reception, when REPS = 0)
When the transmitting node has completed transmission without error (±0 if error counter = 0)	–1	No change
When the receiving node has completed reception without error	No change	• –1 (1 ≤ REC6 to REC0 ≤ 127, when REPS = 0) • ±0 (REC6 to REC0 = 0, when REPS = 0) • Value of 119 to 127 is set (when REPS = 1)

(c) Occurrence of bit error in intermission

An overload frame is generated.

Caution If an error occurs, it is controlled according to the contents of the transmission error counter and reception error counter before the error occurred. The value of the error counter is incremented after the error flag has been output.

(5) Recovery from bus-off state

When the CAN module is in the bus-off state, the CAN module permanently sets its output signals (CTXDn) to recessive level.

The CAN module recovers from the bus-off state in the following bus-off recovery sequence.

1. **A request to enter the CAN initialization mode**
2. **A request to enter a CAN operation mode**

   (a)Recovery operation through normal recovery sequence

   (b)Forced recovery operation that skips recovery sequence

(a) Recovery from bus-off state through normal recovery sequence

The CAN module first issues a request to enter the initialization mode (refer too timing <1> in *Figure 18-17 on page 566*). This request will be immediately acknowledged, and the OPMODE bits of the CnCTRL. register are cleared to 000_B. Processing such as analyzing the fault that has caused the bus-off state, re-defining the CAN module and message buffer using application software, or stopping the operation of the CAN module can be performed by clearing the GOM bit to 0.

Next, the module requests to change the mode from the initialization mode to an operation mode (refer to timing <2> in *Figure 18-17 on page 566*). This starts an operation to recover the CAN module from the bus-off state. The conditions under which the module can recover from the bus-off state are defined by the CAN protocol ISO 11898, and it is necessary to detect 11 consecutive recessive-level bits 128 times. At this time, the request to change the mode to an operation mode is held pending until the recovery conditions are satisfied. When the recovery conditions are satisfied (refer to timing <3> in *Figure 18-17 on page 566*), the CAN module can enter the operation mode it has requested. Until the CAN module enters this operation mode, it stays in the initialization mode. Completion to be requested operation mode can be confirmed by reading the OPMODE bits of the CnCTRL register.

During the bus-off period and bus-off recovery sequence, the BOFF bit of the CnINFO register stays set (to 1). In the bus-off recovery sequence, the reception error counter (REC[6:0]) counts the number of times 11 consecutive recessive-level bits have been detected on the bus. Therefore, the recovery state can be checked by reading REC[6:0].

Caution In the bus-off recovery sequence, REC[6:0] counts up (+1) each time 11 consecutive recessive-level bits have been detected. Even during the bus-off period, the CAN module can enter the CAN sleep mode or CAN stop mode. To start the bus-off recovery sequence, it is necessary to transit to the initialization mode once. However, when the CAN module is in either CAN sleep mode or CAN stop mode, transition request to the initialization mode is not accepted, thus you have to release the CAN sleep mode first. In this case, as soon as the CAN sleep mode is released, the bus-off recovery sequence starts and no transition to initialization mode is necessary. If the can module detects a dominant edge on the CAN bus while in sleep mode even during bus-off, the sleep mode will be left and the bus-off recovery sequence will start.

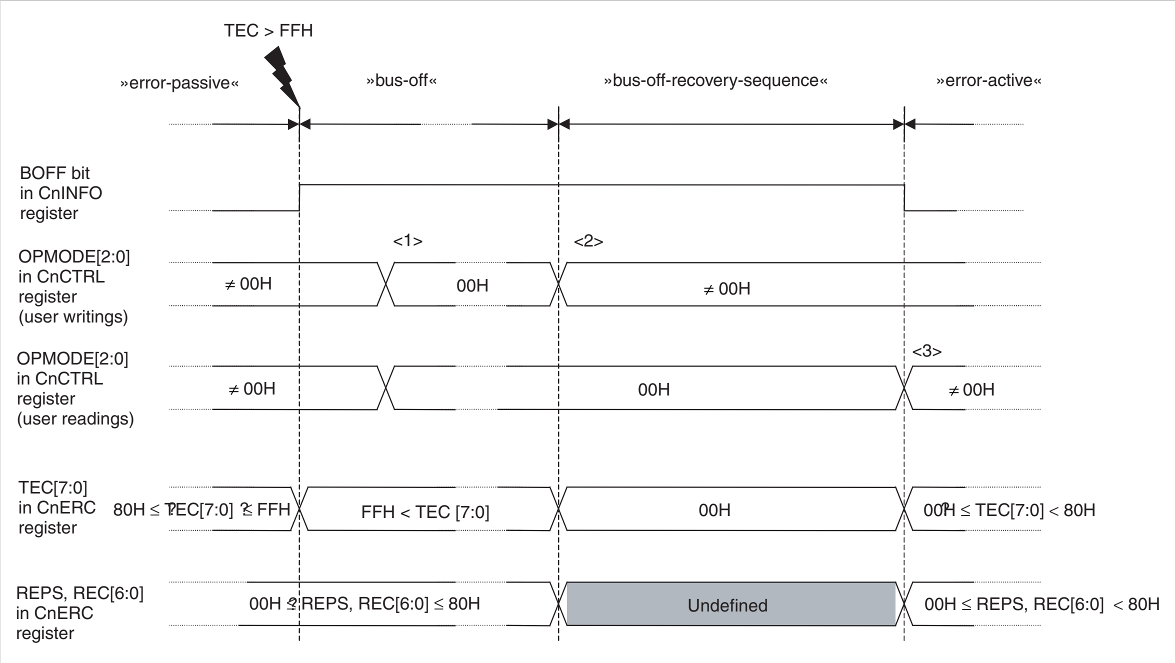

TEC > FFH
»error-passive«
»bus-off«
»bus-off-recovery-sequence«
»error-active«
BOFF bit in CnINFO register
<1>
<2>
OPMODE[2:0] in CnCTRL register (user writings)
≠ 00H
00H
≠ 00H
<3>
OPMODE[2:0] in CnCTRL register (user readings)
≠ 00H
00H
≠ 00H
TEC[7:0] in CnERC register
80H ≤ TEC[7:0] ≤ FFH
FFH < TEC [7:0]
00H
00H ≤ TEC[7:0] < 80H
REPS, REC[6:0] in CnERC register
00H ≤ REPS, REC[6:0] ≤ 80H
Undefined
00H ≤ REPS, REC[6:0] < 80H

Figure 18-17 Recovery from bus-off state through normal recovery sequence

(b) Forced recovery operation that skips bus-off recovery sequence

The CAN module can be forcibly released from the bus-off state, regardless of the bus state, by skipping the bus-off recovery sequence. Here is the procedure.

First, the CAN module requests to enter the initialization mode. For the operation and points to be noted at this time, *“Recovery from bus-off state through normal recovery sequence“ on page 565*.

Next, the module requests to enter an operation mode. At the same time, the CCERC bit of the CnCTRL register must be set to 1.

As a result, the bus-off recovery sequence defined by the CAN protocol ISO 11898 is skipped, and the module immediately enters the operation mode. In this case, the module is connected to the CAN bus after it has monitored 11 consecutive recessive-level bits. For details, refer to the processing in *Figure 18-55 on page 679*.

Caution This function is not defined by the CAN protocol ISO 11898. When using this function, thoroughly evaluate its effect on the network system.

(6) Initializing CAN module error counter register (CnERC) in initialization mode

If it is necessary to initialize the CAN module error counter register (CnERC) and CAN module information register (CnINFO) for debugging or evaluating a program, they can be initialized to the default value by setting the CCERC bit of the CnCTRL register in the initialization mode. When initialization has been completed, the CCERC bit is automatically cleared to 0.

Caution

1. This function is enabled only in the initialization mode. Even if the CCERC bit is set to 1 in a CAN operation mode, the CnERC and CnINFO registers are not initialized.
2. The CCERC bit can be set at the same time as the request to enter a CAN operation mode.

18.3.7 Baud rate control function

(1) Prescaler

The CAN controller has a prescaler that divides the clock (f_{CAN}) supplied to CAN. This prescaler generates a CAN protocol layer basic system clock (f_{TQ}) derived from the CAN module system clock (f_{CANMOD}), and divided by 1 to 256 (*"CnBRP - CANn module bit rate prescaler register" on page 599*).

(2) Data bit time (8 to 25 time quanta)

One data bit time is defined as shown in *Figure 18-18 on page 568*.

The CAN Controller sets time segment 1, time segment 2, and reSynchronization Jump Width (SJW) of data bit time, as shown in *Figure 18-18*. Time segment 1 is equivalent to the total of the propagation (prop) segment and phase segment 1 that are defined by the CAN protocol specification. Time segment 2 is equivalent to phase segment 2.

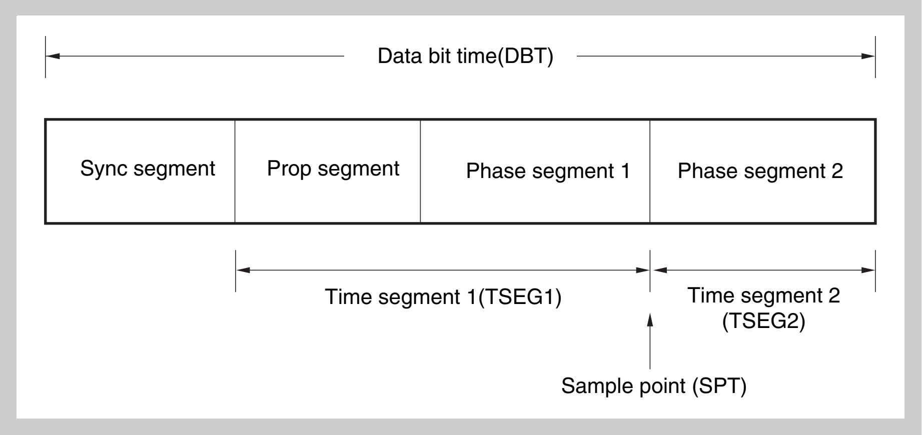


Figure 18-18 Segment setting

Table 18-15 Segment setting

Segment name	Settable range	Notes on setting to conform to CAN specification
Time segment 1 (TSEG1)	2TQ to 15TQ	-
Time segment 2 (TSEG2)	1TQ to 8TQ	IPT of the CAN controller is 0TQ. To conform to the CAN protocol specification, therefore, a length less or equal to phase segment 1 must be set here. This means that the length of time segment 1 minus 1TQ is the settable upper limit of time segment 2.
Resynchronization Jump Width (SJW)	1TQ to 4TQ	The length of time segment 1 minus 1TQ or 4 TQ, whichever is smaller.

Note
1. IPT: Information Processing Time
2. TQ: Time Quanta

Reference: The CAN protocol specification defines the segments constituting the data bit time as shown in *Figure 18-19*.

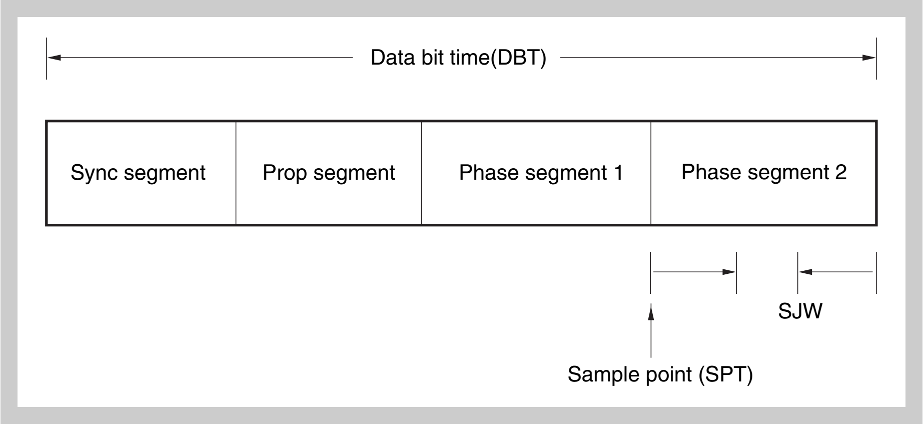


Figure 18-19 Configuration of data bit time defined by CAN specification

Table 18-16 Configuration of data bit time defined by CAN specification

Segment name	Settable range	Notes on setting to conform to CAN specification
Sync segment (Synchronization segment)	1	This segment starts at the edge where the level changes from recessive to dominant when hardware synchronization is established.
Prop segment	Programmable to 1 to 8 or more	This segment absorbs the delay of the output buffer, CAN bus, and input buffer. The length of this segment is set so that ACK is returned before the start of phase segment 1. Time of prop segment ≥ (Delay of output buffer) + 2 × (Delay of CAN bus) + (Delay of input buffer) This segment compensates for an error of data bit time. The longer this segment, the wider the permissible range but the slower the communication speed.
Phase segment 1	Programmable to 1 to 8	
Phase segment 2	Phase segment 1 or IPT, whichever greater	
SJW	Programmable from 1TQ to length of segment 1 or 4TQ, whichever is smaller	This width sets the upper limit of expansion or contraction of the phase segment during resynchronization.

Note IPT: Information Processing Time

(3) Synchronizing data bit

- The receiving node establishes synchronization by a level change on the bus because it does not have a sync signal.
- The transmitting node transmits data in synchronization with the bit timing of the transmitting node.

(a) Hardware synchronization

This synchronization is established when the receiving node detects the start of frame in the interframe space.

- When a falling edge is detected on the bus, that TQ means the sync segment and the next segment is the prop segment. In this case, synchronization is established regardless of SJW.

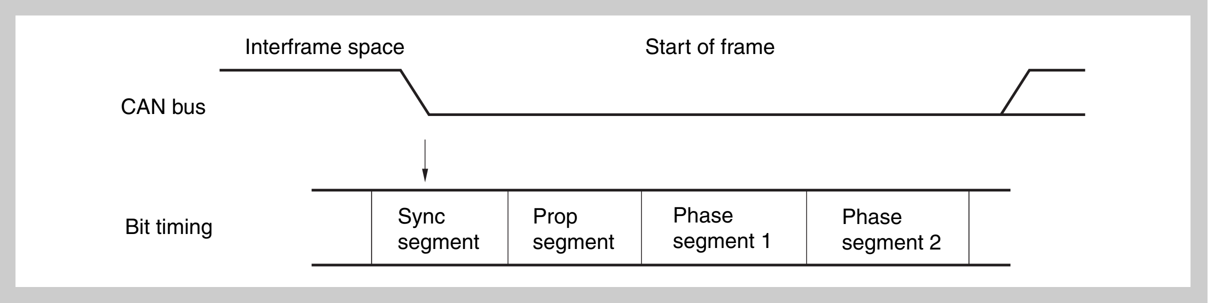


Figure 18-20 Adjusting synchronization of data bit

(b) Resynchronization

Synchronization is established again if a level change is detected on the bus during reception (only if a recessive level was sampled previously).

- The phase error of the edge is given by the relative position of the detected edge and sync segment.

  <Sign of phase error>

0:	If the edge is within the sync segment
Positive:	If the edge is before the sample point (phase error)
Negative:	If the edge is after the sample point (phase error)

  If phase error is positive: Phase segment 1 is lengthened by specified SJW.

  If phase error is negative: Phase segment 2 is shortened by specified SJW.

- The sample point of the data of the receiving node moves relatively due to the “discrepancy” in the baud rate between the transmitting node and receiving node.

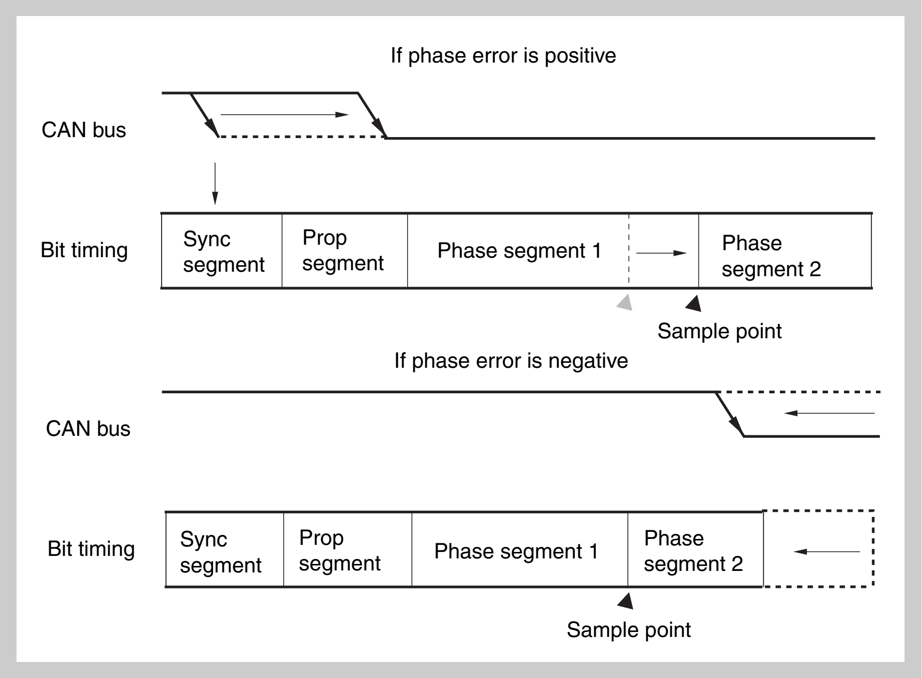


Figure 18-21 Resynchronization

18.4 Connection with Target System

The CAN module has to be connected to the CAN bus using an external transceiver.

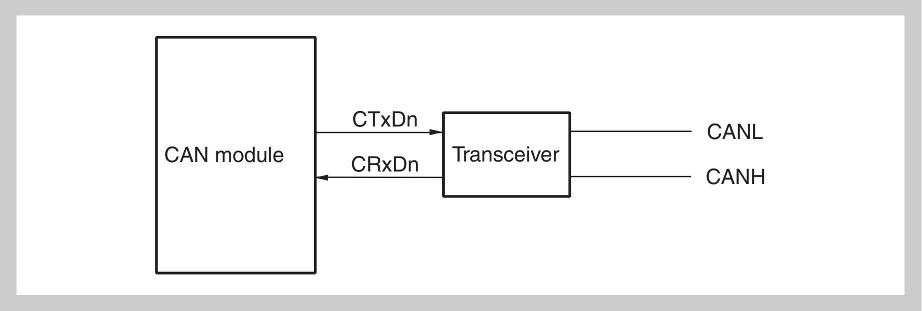


Figure 18-22 Connection to CAN bus

18.5 Internal Registers of CAN Controller

18.5.1 CAN module register and message buffer addresses

In this chapter all register and message buffer addresses are defined as address offsets to different base addresses.

Since all registers are accessed via the programmable peripheral area the bottom address is defined by the BPC register (refer to *“Programmable peripheral I/O area“ on page 94* or to *“Programmable peripheral I/O area (PPA)“ on page 265*).

The addresses given in the following tables are offsets to the programmable peripheral area base address PBA.

The recommended setting of PBA is 8FFB$_H$. This setting would define the programmable peripheral area base address

PBA = 03FE C000$_H$

Table 18-17 lists all base addresses used throughout this chapter.

Table 18-17 CAN module base addresses

Base address name	Base address of	Address	Address for BPC =8FFB$_H$
C0RBaseAddr	CAN0 registers	PBA + 000$_H$	03FE C000$_H$
C0MBaseAddr	CAN0 message buffers	PBA + 100$_H$	03FE C100$_H$

In the following <CnRBaseAddr> respectively <CnMBaseAddr> are used for the base address names for CAN channel n.

18.5.2 CAN Controller configuration

Table 18-18 List of CAN Controller registers

Item	Register Name
CANn global registers	CANn global control register (CnGMCTRL)
	CANn global clock selection register (CnGMCS)
	CANn global automatic block transmission control register (CnGMABT)
	CANn global automatic block transmission delay setting register (CnGMABTD)
CANn module registers	CANn module mask 1 register (CnMASK1L, CnMASK1H)
	CANn module mask 2 register (CnMASK2L, CnMASK2H)
	CANn module mask3 register (CnMASK3L, CnMASK3H)
	CANn module mask 4 registers (CnMASK4L, CnMASK4H)
	CANn module control register (CnCTRL)
	CANn module last error information register (CnLEC)
	CANn module information register (CnINFO)
	CANn module error counter register (CnERC)
	CANn module interrupt enable register (CnIE)
	CANn module interrupt status register (CnINTS)
	CANn module bit rate prescaler register (CnBRP)
	CANn module bit rate register (CnBTR)
	CANn module last in-pointer register (CnLIPT)
	CANn module receive history list register (CnRGPT)
	CANn module last out-pointer register (CnLOPT)
	CANn module transmit history list register (CnTGPT)
	CANn module time stamp register (CnTS)
CANn message buffer registers	CANn message data byte 01 register m (CnMDATA01m)
	CANn message data byte 0 register m (CnMDATA0m)
	CANn message data byte 1 register m (CnMDATA1m)
	CANn message data byte 23 register m (CnMDATA23m)
	CANn message data byte 2 register m (CnMDATA2m)
	CANn message data byte 3 register m (CnMDATA3m)
	CANn message data byte 45 register m (CnMDATA45m)
	CANn message data byte 4 register m (CnMDATA4m)
	CANn message data byte 5 register m (CnMDATA5m)
	CANn message data byte 67 register m (CnMDATA67m)
	CANn message data byte 6 register m (CnMDATA6m)
	CANn message data byte 7 register m (CnMDATA7m)
	CANn message data length register m (CnMDLCm)
	CANn message configuration register m (CnMCONFm)
	CANn message ID register m (CnMIDLm, CnMIDHm)
	CANn message control register m (CnMCTRLm)

18.5.3 CAN registers overview

(1) CANn global and module registers

The following table lists the address offsets to the CANn register base address CnRBaseAddr.

Table 18-19 CANn global and module registers

Address offset	Register name	Symbol	R/W	Access			After reset
				1-bit	8-bit	16-bit	
000_H	CANn global control register	CnGMCTRL	R/W	–	–	√	0000_H
002_H	CANn global clock selection register	CnGMCS			√		$0F_H$
006_H	CANn global automatic block transmission register	CnGMABT		–	–	√	0000_H
008_H	CANn global automatic block transmission delay register	CnGMABTD		–	√	–	00_H
040_H	CANn module mask 1 register	CnMASK1L		–	–	√	Undefined
042_H		CnMASK1H		–	–	√	Undefined
044_H	CANn module mask 2 register	CnMASK2L		–	–	√	Undefined
046_H		CnMASK2H		–	–	√	Undefined
048_H	CANn module mask 3 register	CnMASK3L		–	–	√	Undefined
$04A_H$		CnMASK3H		–	–	√	Undefined
$04C_H$	CANn module mask 4 register	CnMASK4L		–	–	√	Undefined
$04E_H$		CnMASK4H		–	–	√	Undefined
050_H	CANn module control register	CnCTRL		–	–	√	0000_H
052_H	CANn module last error code register	CnLEC		–	√	–	00_H
053_H	CANn module information register	CnINFO	R	–	√	–	00_H
054_H	CANn module error counter register	CnERC		–	–	√	0000v
056_H	CANn module interrupt enable register	CnIE	R/W	–	–	√	0000_H
058_H	CANn module interrupt status register	CnINTS		–	–	√	0000_H
$05A_H$	CANn module bit-rate prescaler register	CnBRP		–	√	–	FF_H
$05C_H$	CANn module bit-rate register	CnBTR		–	–	√	$370F_H$
$05E_H$	CANn module last in-pointer register	CnLIPT	R	–	√	–	Undefined
060_H	CANn module receive history list register	CnRGPT	R/W	–	–	√	$xx02_H$
062_H	CANn module last out-pointer register	CnLOPT	R	–	√	–	Undefined
064_H	CANn module transmit history list register	CnTGPT	R/W	–	–	√	$xx02_H$
066_H	CANn module time stamp register	CnTS		–	–	√	0000_H

(2) CANn message buffer registers

The addresses in the following table denote the address offsets to the CANn message buffer base address:

CnMBaseAddr.

Example CAN0, message buffer register m = 14 = E_H, byte 6 C0MDATA614 has the address E_H x 20_H + 6_H + C0MBaseAddr

Note The message buffer register number m in the register symbols has 2 digits, for example,
COMDATA01m = COMDATA0100 for m = 0.

Table 18-20 CANn message buffer registers

Address offset	Register name	Symbol	R/W	Access			After reset
				1-bit	8-bit	16-bit	
mx20_H + 0_H	CANn message data byte 01 register m	CnMDATA01m	R/W	–	–	√	Undefined
mx20_H + 0_H	CANn message data byte 0 register m	CnMDATA0m		–	√	–	Undefined
mx20_H + 1_H	CANn message data byte 1 register m	CnMDATA1m		–	√	–	Undefined
mx20_H + 2_H	CANn message data byte 23 register m	CnMDATA23m		–	–	√	Undefined
mx20_H + 2_H	CANn message data byte 2 register m	CnMDATA2m		–	√	–	Undefined
mx20_H + 3_H	CANn message data byte 3 register m	CnMDATA3m		–	√	–	Undefined
mx20_H + 4_H	CANn message data byte 45 register m	CnMDATA45m		–	–	√	Undefined
mx20_H + 4_H	CANn message data byte 4 register m	CnMDATA4m		–	√	–	Undefined
mx20_H + 5_H	CANn message data byte 5 register m	CnMDATA5m		–	√	–	Undefined
mx20_H + 6_H	CANn message data byte 67 register m	CnMDATA67m		–	–	√	Undefined
mx20_H + 6_H	CANn message data byte 6 register m	CnMDATA6m		–	√	–	Undefined
mx20_H + 7_H	CANn message data byte 7 register m	CnMDATA7m		–	√	–	Undefined
mx20_H + 8_H	CANn message data length register m	CnMDLCm		–	√	–	0000 $xxxx_B$
mx20_H + 9_H	CANn message configuration register m	CnMCONFm		–	√	–	Undefined
mx20_H + A_H	CANn message identifier register m	CnMIDLm		–	–	√	Undefined
mx20_H + C_H		CnMIDHm		–	–	√	Undefined
mx20_H + E_H	CANn message control register m	CnMCTRLm		–	–	√	0x00 0000 0000 0000_B

18.5.4 Register bit configuration

Table 18-21 CAN global register bit configuration

Address offset[a]	Symbol	Bit 7/15	Bit 6/14	Bit 5/13	Bit 4/12	Bit 3/11	Bit 2/10	Bit 1/9	Bit 0/8
00_H	CnGMCTRL (W)	0	0	0	0	0	0	0	Clear GOM
01_H		0	0	0	0	0	0	Set EFSD	Set GOM
00_H	CnGMCTRL (R)	0	0	0	0	0	0	EFSD	GOM
01_H		MBON	0	0	0	0	0	0	0
02_H	CnGMCS	0	0	0	0	CCP3	CCP2	CCP1	CCP0
06_H	CnGMABT (W)	0	0	0	0	0	0	0	Clear ABTTRG
07_H		0	0	0	0	0	0	Set ABTCLR	Set ABTTRG
06_H	CnGMABT (R)	0	0	0	0	0	0	ABTCLR	ABTTRG
07_H		0	0	0	0	0	0	0	0
08_H	CnGMABTD	0	0	0	0	ABTD3	ABTD2	ABTD1	ABTD0

a) Base address: <CnRBaseAddr>

Table 18-22 CAN module register bit configuration (1/2)

Address offset[a]	Symbol	Bit 7/15	Bit 6/14	Bit 5/13	Bit 4/12	Bit 3/11	Bit 2/10	Bit 1/9	Bit 0/8
40_H	CnMASK1L	CMID7 to CMID0							
41_H		CMID15 to CMID8							
42_H	CnMASK1H	CMID23 to CMID16							
43_H		0	0	0	CMID28 to CMID24				
44_H	CnMASK2L	CMID7 to CMID0							
45_H		CMID15 to CMID8							
46_H	CnMASK2H	CMID23 to CMID16							
47_H		0	0	0	CMID28 to CMID24				
48_H	CnMASK3L	CMID7 to CMID0							
49_H		CMID15 to CMID8							
$4A_H$	CnMASK3H	CMID23 to CMID16							
$4B_H$		0	0	0	CMID28 to CMID24				
$4C_H$	CnMASK4L	CMID7 to CMID0							
$4D_H$		CMID15 to CMID8							
$4E_H$	CnMASK4H	CMID23 to CMID16							
$4F_H$		0	0	0	CMID28 to CMID24				
50_H	CnCTRL (W)	0	Clear AL	Clear VALID	Clear PSMODE1	Clear PSMODE0	Clear OPMODE2	Clear OPMODE1	Clear OPMODE0
51_H		Set CCERC	Set AL	0	Set PSMODE1	Set PSMODE0	Set OPMODE2	Set OPMODE1	Set OPMODE0
50_H	CnCTRL (R)	CCERC	AL	VALID	PS MODE1	PS MODE0	OP MODE2	OP MODE1	OP MODE0
51_H		0	0	0	0	0	0	RSTAT	TSTAT

Table 18-22 CAN module register bit configuration (2/2)

Address offset[a]	Symbol	Bit 7/15	Bit 6/14	Bit 5/13	Bit 4/12	Bit 3/11	Bit 2/10	Bit 1/9	Bit 0/8
52_H	CnLEC (W)	0	0	0	0	0	0	0	0
52_H	CnLEC (R)	0	0	0	0	0	LEC2	LEC1	LEC0
53_H	CnINFO	0	0	0	BOFF	TECS1	TECS0	RECS1	RECS0
54_H	CnERC	TEC7 to TEC0							
55_H		REC7 to REC0							
56_H	CnIE (W)	0	0	Clear CIE5	Clear CIE4	Clear CIE3	Clear CIE2	Clear CIE1	Clear CIE0
57_H		0	0	Set CIE5	Set CIE4	Set CIE3	Set CIE2	Set CIE1	Set CIE0
56_H	CnIE (R)	0	0	CIE5	CIE4	CIE3	CIE2	CIE1	CIE0
57_H		0	0	0	0	0	0	0	0
58_H	CnINTS (W)	0	0	Clear CINTS5	Clear CINTS4	Clear CINTS3	Clear CINTS2	Clear CINTS1	Clear CINTS0
59_H		0	0	0	0	0	0	0	0
58_H	CnINTS (R)	0	0	CINTS5	CINTS4	CINTS3	CINTS2	CINTS1	CINTS0
59_H		0	0	0	0	0	0	0	0
$5A_H$	CnBRP	TQPRS7 to TQPRS0							
$5C_H$	CnBTR	0	0	0	0	TSEG13 to TSEG10			
$5D_H$		0	0	SJW1, SJW0		0	TSEG22 to TSEG20		
$5E_H$	CnLIPT	LIPT7 to LIPT0							
60_H	CnRGPT (W)	0	0	0	0	0	0	0	Clear ROVF
61_H		0	0	0	0	0	0	0	0
60_H	CnRGPT (R)	0	0	0	0	0	0	RHPM	ROVF
61_H		RGPT7 to RGPT0							
$F62_H$	CnLOPT	LOPT7 to LOPT0							
64_H	CnTGPT (W)	0	0	0	0	0	0	0	Clear TOVF
65_H		0	0	0	0	0	0	0	0
64_H	CnTGPT (R)	0	0	0	0	0	0	THPM	TOVF
65_H		TGPT7 to TGPT0							
66_H	CnTS (W)	0	0	0	0	0	Clear TSLOCK	Clear TSSEL	Clear TSEN
67_H		0	0	0	0	0	Set TSLOCK	Set TSSEL	Set TSEN
66_H	CnTS (R)	0	0	0	0	0	TSLOCK	TSSEL	TSEN
67_H		0	0	0	0	0	0	0	0
68_H to FF_H	-	Access prohibited (reserved for future use)							

[a] Base address: <CnRBaseAddr>

Table 18-23 Message buffer register bit configuration

Address offset[a]	Symbol	Bit 7/15	Bit 6/14	Bit 5/13	Bit 4/12	Bit 3/11	Bit 2/10	Bit 1/9	Bit 0/8
0_H	CnMDATA01m	Message data (byte 0)							
1_H		Message data (byte 1)							
0_H	CnMDATA0m	Message data (byte 0)							
1_H	CnMDATA1m	Message data (byte 1)							
2_H	CnMDATA23m	Message data (byte 2)							
3_H		Message data (byte 3)							
2_H	CnMDATA2m	Message data (byte 2)							
3_H	CnMDATA3m	Message data (byte 3)							
4H	CnMDATA45m	Message data (byte 4)							
5_H		Message data (byte 5)							
4_H	CnMDATA4m	Message data (byte 4)							
5_H	CnMDATA5m	Message data (byte 5)							
6_H	CnMDATA67m	Message data (byte 6)							
7_H		Message data (byte 7)							
6_H	CnMDATA6m	Message data (byte 6)							
7_H	CnMDATA7m	Message data (byte 7)							
8_H	CnMDLCm	0				MDLC3	MDLC2	MDLC1	MDLC0
9_H	CnMCONFm	OWS	RTR	MT2	MT1	MT0	0	0	MA0
A_H	CnMIDLm	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
B_H		ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8
C_H	CnMIDHm	ID23	ID22	ID21	ID20	ID19	ID18	ID17	ID16
D_H		IDE	0	0	ID28	ID27	ID26	ID25	ID24
E_H	CnMCTRLm (W)	0	0	0	Clear MOW	Clear IE	Clear DN	Clear TRQ	Clear RDY
F_H		0	0	0	0	Set IE	0	Set TRQ	Set RDY
E_H	CnMCTRLm (R)	0	0	0	MOW	IE	DN	TRQ	RDY
F_H		0	0	MUC	0	0	0	0	0

a) Base address: <CnMBaseAddr>

Note For calculation of the complete message buffer register addresses refer to *“CAN registers overview“ on page 574*.

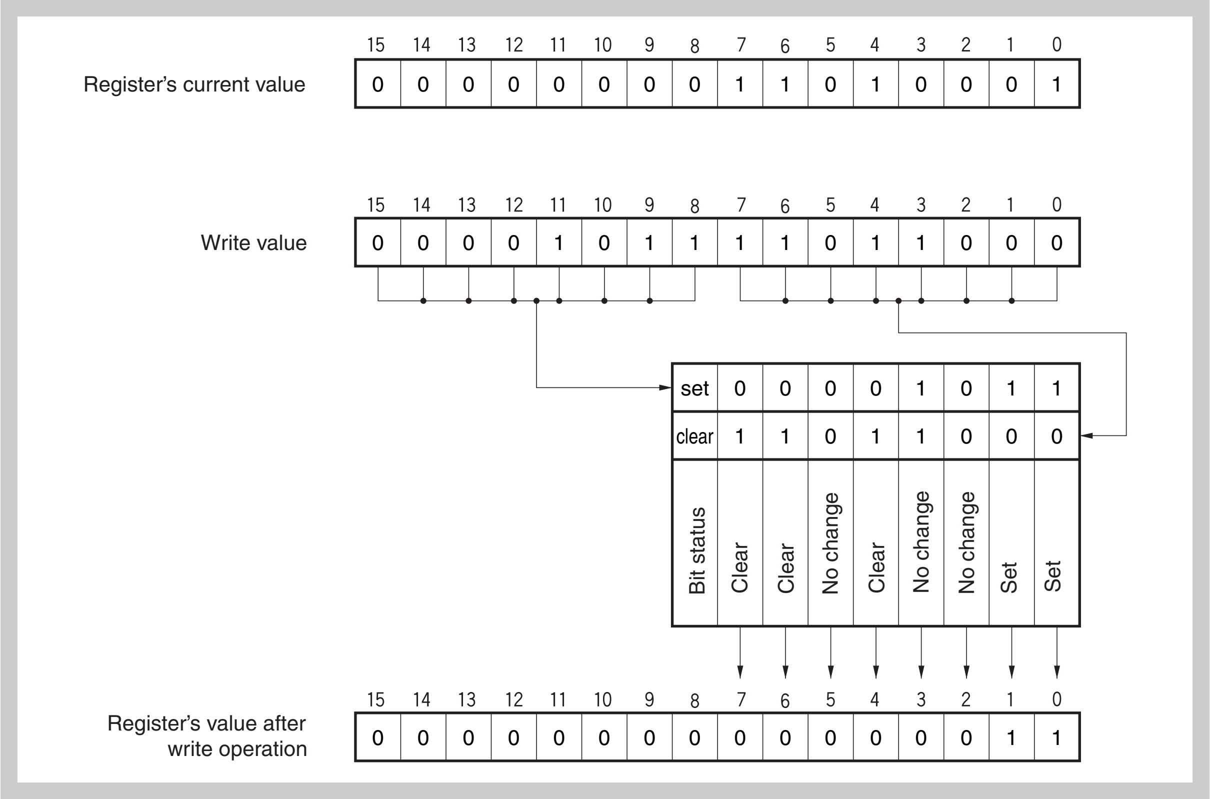
18.6 Bit Set/Clear Function

The CAN control registers include registers whose bits can be set or cleared via the CPU and via the CAN interface. An operation error occurs if the following registers are written directly. Do not write any values directly via bit manipulation, read/modify/write, or direct writing of target values.

- CANn global control register (CnGMCTRL)
- CANn global automatic block transmission control register (CnGMABT)
- CANn module control register (CnCTRL)
- CANn module interrupt enable register (CnIE)
- CANn module interrupt status register (CnINTS)
- CANn module receive history list register (CnRGPT)
- CANn module transmit history list register (CnTGPT)
- CANn module time stamp register (CnTS)
- CANn message control register (CnMCTRLm)

All the 16 bits in the above registers can be read via the usual method. Use the procedure described in *Figure 18-23* below to set or clear the lower 8 bits in these registers.

Setting or clearing of lower 8 bits in the above registers is performed in combination with the higher 8 bits (refer to the bit status after set/clear operation is specified in *Figure 18-26*). *Figure 18-23* shows how the values of set bits or clear bits relate to set/clear/no change operations in the corresponding register.



	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Register’s current value	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	1
Write value	0	0	0	0	1	0	1	1	1	1	0	1	1	0	0	0

set	0	0	0	0	1	0	1	1
clear	1	1	0	1	1	0	0	0
Bit status	Clear	Clear	No change	Clear	No change	No change	Set	Set

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Register’s value after write operation	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Figure 18-23 Example of bit setting/clearing operations

(1) Bit status after bit setting/clearing operations

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

Set 0 ... 7	Clear 0 ... 7	Status of bit n after bit set/clear operation
0	0	No change
0	1	0
1	0	1
1	1	No change

18.7 Control Registers

(1) CnGMCTRL - CANn global control register

The CnGMCTRL register is used to control the operation of the CAN module.

Access This register can be read/written in 16-bit units.

Address <CnRBaseAddr> + 000_H

Initial Value 0000_H. The register is initialized by any reset.

(a) CnGMCTRL read

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

MBON	Bit enabling access to message buffer register, transmit/receive history registers
0	Write access and read access to the message buffer register and the transmit/receive history list registers is disabled.
1	Write access and read access to the message buffer register and the transmit/receive history list registers is enabled.

Caution

1. While the MBON bit is cleared (to 0), software access to the message buffers (CnMDATA0m, CnMDATA1m, CnMDATA01m, CnMDATA2m, CnMDATA3m, CnMDATA23m, CnMDATA4m, CnMDATA5m, CnMDATA45m, CnMDATA6m, CnMDATA7m, CnMDATA67m, CnMDLCm, CnMCONFm, CnMIDLm, CnMIDHm, and CnMCTRLm), or registers related to transmit history or receive history (CnLOPT, CnTGPT, CnLIPT, and CnRGPT) is disabled.
2. This bit is read-only. Even if 1 is written to the MBON bit while it is 0, the value of the MBON bit does not change, and access to the message buffer registers, or registers related to transmit history or receive history remains disabled.

Note The MBON bit is cleared (to 0) when the CAN module enters CAN sleep mode/ CAN stop mode, or when the GOM bit is cleared (to 0).
The MBON bit is set (to 1) when the CAN sleep mode/CAN stop mode is released, or when the GOM bit is set (to 1).

EFSD	Bit enabling forced shut down
0	Forced shut down disabled.
1	Forced shut down enabled by subsequent clearing of GOM bit to 0.

Caution

1. To request forced shut down, the GOM bit must be cleared to 0 in a subsequent, immediately following access after the EFSD bit has been set to 1. If access to another register (including reading the CnGMCTRL register) is executed (even during NMI processing or DMAC operation) without clearing the GOM bit immediately after the EFSD bit has been set to 1, the EFSD bit is forcibly cleared to 0, and the forced shut down request is invalid.
2. EFSD only works, if no continuous DMA transfer is performed.

GOM	Global operation mode bit
0	CAN module is disabled from operating.
1	CAN module is enabled to operate.

Caution The GOM can be cleared only in the initialization mode or immediately after EFSD bit is set (to 1).

(b) CnGMCTRL write

15	14	13	12	11	10	9	8
0	0	0	0	0	0	Set EFSD	Set GOM
7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	Clear GOM

Set EFSD	EFSD bit setting
0	No change in EFSD bit.
1	EFSD bit set to 1.

Set GOM	Clear GOM	GOM bit setting
0	1	GOM bit cleared to 0.
1	0	GOM bit set to 1.
Other than above		No change in GOM bit.

Caution Set the GOM bit and EFSD bit always separately.

(2) CnGMCS - CANn global clock selection register

The CnGMCS register is used to select the CAN module system clock.

Access This register can be read/written in 8-bit units.

Address <CnRBaseAddr> + 002_H

Initial Value $0F_H$. The register is initialized by any reset.

7	6	5	4	3	2	1	0
0	0	0	0	CCP3	CCP2	CCP1	CCP0

CCP3	CCP2	CCP1	CCP1	CAN module system clock (f_{CANMOD})
0	0	0	0	$f_{CAN}/1$
0	0	0	1	$f_{CAN}/2$
0	0	1	0	$f_{CAN}/3$
0	0	1	1	$f_{CAN}/4$
0	1	0	0	$f_{CAN}/5$
0	1	0	1	$f_{CAN}/6$
0	1	1	0	$f_{CAN}/7$
0	1	1	1	$f_{CAN}/8$
1	0	0	0	$f_{CAN}/9$
1	0	0	1	$f_{CAN}/10$
1	0	1	0	$f_{CAN}/11$
1	0	1	1	$f_{CAN}/12$
1	1	0	0	$f_{CAN}/13$
1	1	0	1	$f_{CAN}/14$
1	1	1	0	$f_{CAN}/15$
1	1	1	1	$f_{CAN}/16$ (default value)

Note f_{CAN} = clock supplied to CAN

(3) CnGMABT - CANn global automatic block transmission control register

The CnGMABT register is used to control the automatic block transmission (ABT) operation.

Access This register can be read/written in 16-bit units.

Address <CnRBaseAddr> + 006_H

Initial Value 0000_H. The register is initialized by any reset.

(a) CnGMABT read

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

ABTCLR	Automatic block transmission engine clear status bit
0	Clearing the automatic transmission engine is completed.
1	The automatic transmission engine is being cleared.

Note

1. Set the ABTCLR bit to 1 while the ABTTRG bit is cleared to 0.
   The operation is not guaranteed if the ABTCLR bit is set to 1 while the ABTTRG bit is set to 1.
2. When the automatic block transmission engine is cleared by setting the ABTCLR bit to 1, the ABTCLR bit is automatically cleared to 0 as soon as the requested clearing processing is complete.

ABTTRG	Automatic block transmission status bit
0	Automatic block transmission is stopped.
1	Automatic block transmission is under execution.

Caution

1. Do not set the ABTTRG bit (1) in the initialization mode. If the ABTTRG bit is set in the initialization mode, the operation is not guaranteed after the CAN module has entered the normal operation mode with ABT.
2. Do not set the ABTTRG bit (1) while the CnCTRL.TSTAT bit is set (1). Confirm TSTAT = 0 directly in advance before setting ABTTRG bit.

(b) CnGMABT write

15	14	13	12	11	10	9	8
0	0	0	0	0	0	Set ABTCLR	Set ABTTRG

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	Clear ABTTRG

Caution Before changing the normal operation mode with ABT to the initialization mode, be sure to set the CnGMABT register to the default value (0000_H) and confirm the CnGMABT register is surely initialized to the default value (0000_H).

Set ABTCLR	Automatic block transmission engine clear request bit
0	The automatic block transmission engine is in idle status or under operation.
1	Request to clear the automatic block transmission engine. After the automatic block transmission engine has been cleared, automatic block transmission is started from message buffer 0 by setting the ABTTRG bit to 1.

Set ABTTRG	Clear ABTTRG	Automatic block transmission start bit
0	1	Request to stop automatic block transmission.
1	0	Request to start automatic block transmission.
Other than above		No change in ABTTRG bit.

(4) CnGMABTD - CANn global automatic block transmission delay register

The CnGMABTD register is used to set the interval at which the data of the message buffer assigned to ABT is to be transmitted in the normal operation mode with ABT.

Access This register can be read/written in 8-bit units.

Address <CnRBaseAddr> + 008_H

Initial Value 00_H. The register is initialized by any reset.

7	6	5	4	3	2	1	0
0	0	0	0	ABTD3	ABTD2	ABTD1	ABTD0

ABTD3	ABTD2	ABTD1	ABTD0	Data frame interval during automatic block transmission in DBT[a]
0	0	0	0	0 DBT (default value)
0	0	0	1	2^5 DBT
0	0	1	0	2^6 DBT
0	0	1	1	2^7 DBT
0	1	0	0	2^8 DBT
0	1	0	1	2^9 DBT
0	1	1	0	2^{10} DBT
0	1	1	1	2^{11} DBT
1	0	0	0	2^{12} DBT
Other than above				Setting prohibited

a) Unit: Data bit time (DBT)

Caution

1. Do not change the contents of the CnGMABTD register while the ABTTRG bit is set to 1.
2. The timing at which the ABT message is actually transmitted onto the CAN bus differs depending on the status of transmission from the other station or how a request to transmit a message other than an ABT message (message buffers 8 to 31) is made.

(5) CnMASKaL, CnMASKaH - CANn module mask control register (a = 1 to 4)

The CnMASKaL and CnMASKaH registers are used to extend the number of receivable messages into the same message buffer by masking part of the identifier (ID) comparison of a message and invalidating the ID of the masked part.

(a) CANn module mask 1 register (CnMASK1L, CnMASK1H)

Access These registers can be read/written in 16-bit units.

Address CnMASK1L: <CnRBaseAddr> + 040_H
CnMASK1H: <CnRBaseAddr> + 042_H

Initial Value Undefined.

CnMASK1L

15	14	13	12	11	10	9	8
CMID15	CMID14	CMID13	CMID12	CMID11	CMID10	CMID9	CMID8
7	6	5	4	3	2	1	0
CMID7	CMID6	CMID5	CMID4	CMID3	CMID2	CMID1	CMID0

CnMASK1H

15	14	13	12	11	10	9	8
0	0	0	CMID28	CMID27	CMID26	CMID25	CMID24
7	6	5	4	3	2	1	0
CMID23	CMID22	CMID21	CMID20	CMID19	CMID18	CMID17	CMID16

(b) CANn module mask 2 register (CnMASK2L, CnMASK2H)

Access These registers can be read/written in 16-bit units.

Address CnMASK2L: <CnRBaseAddr> + 044_H
CnMASK2H: <CnRBaseAddr> + 046_H

Initial Value Undefined.

CnMASK2L

15	14	13	12	11	10	9	8
CMID15	CMID14	CMID13	CMID12	CMID11	CMID10	CMID9	CMID8
7	6	5	4	3	2	1	0
CMID7	CMID6	CMID5	CMID4	CMID3	CMID2	CMID1	CMID0

CnMASK2H

15	14	13	12	11	10	9	8
0	0	0	CMID28	CMID27	CMID26	CMID25	CMID24
7	6	5	4	3	2	1	0
CMID23	CMID22	CMID21	CMID20	CMID19	CMID18	CMID17	CMID16

(c) CANn module mask 3 register (CnMASK3L, CnMASK3H)

Access These registers can be read/written in 16-bit units.

Address CnMASK3L: <CnRBaseAddr> + 048_H
CnMASK3H: <CnRBaseAddr> + $04A_H$

Initial Value Undefined.

CnMASK3L

15	14	13	12	11	10	9	8
CMID15	CMID14	CMID13	CMID12	CMID11	CMID10	CMID9	CMID8
7	6	5	4	3	2	1	0
CMID7	CMID6	CMID5	CMID4	CMID3	CMID2	CMID1	CMID0

CnMASK3H

15	14	13	12	11	10	9	8
0	0	0	CMID28	CMID27	CMID26	CMID25	CMID24
7	6	5	4	3	2	1	0
CMID23	CMID22	CMID21	CMID20	CMID19	CMID18	CMID17	CMID16

(d) CANn module mask 4 register (CnMASK4L, CnMASK4H)

Access These registers can be read/written in 16-bit units.

Address CnMASK4L: <CnRBaseAddr> + $04C_H$
CnMASK4H: <CnRBaseAddr> + $04E_H$

Initial Value Undefined.

CnMASK4L

15	14	13	12	11	10	9	8
CMID15	CMID14	CMID13	CMID12	CMID11	CMID10	CMID9	CMID8
7	6	5	4	3	2	1	0
CMID7	CMID6	CMID5	CMID4	CMID3	CMID2	CMID1	CMID0

CnMASK4H

15	14	13	12	11	10	9	8
0	0	0	CMID28	CMID27	CMID26	CMID25	CMID24
7	6	5	4	3	2	1	0
CMID23	CMID22	CMID21	CMID20	CMID19	CMID18	CMID17	CMID16

CMID28 to CMID0	Mask pattern setting of ID bit
0	The ID bits of the message buffer set by the CMID28 to CMID0 bits are compared with the ID bits of the received message frame.
1	The ID bits of the message buffer set by the CMID28 to CMID0 bits are not compared with the ID bits of the received message frame (they are masked).

Note Masking is always defined by an ID length of 29 bits. If a mask is assigned to a message with a standard ID, the CMID17 to CMID0 bits are ignored. Therefore, only the CMID28 to CMID18 bits of the received ID are masked. The same mask can be used for both the standard and extended IDs.

(6) CnCTRL - CANn module control register

The CnCTRL register is used to control the operation mode of the CAN module.

Access This register can be read/written in 16-bit units.

Address <CnRBaseAddr> + 050_H

Initial Value 0000_H. The register is initialized by any reset.

(a) CnCTRL read

15	14	13	12	11	10	9	8
0	0	0	0	0	0	RSTAT	TSTAT
7	6	5	4	3	2	1	0
CCERC	AL	VALID	PSMODE1	PSMODE0	OPMODE2	OPMODE1	OPMODE0

RSTAT	Reception status bit
0	Reception is stopped.
1	Reception is in progress.

Note 1. The RSTAT bit is set to 1 under the following conditions (timing)

- The SOF bit of a receive frame is detected
- On occurrence of arbitration loss during a transmit frame

2. The RSTAT bit is cleared to 0 under the following conditions (timing)

- When a recessive level is detected at the second bit of the interframe space
- On transition to the initialization mode at the first bit of the interframe space

TSTAT	Transmission status bit
0	Transmission is stopped.
1	Transmission is in progress.

Note 1. The TSTAT bit is set to 1 under the following conditions (timing)

- The SOF bit of a transmit frame is detected

2. The TSTAT bit is cleared to 0 under the following conditions (timing)

- During transition to bus-off state
- On occurrence of arbitration loss in transmit frame
- On detection of recessive level at the second bit of the interframe space
- On transition to the initialization mode at the first bit of the interframe space

CCERC	Error counter clear bit
0	The CnERC and CnINFO registers are not cleared in the initialization mode.
1	The CnERC and CnINFO registers are cleared in the initialization mode.

Note

1. The CCERC bit is used to clear the CnERC and CnINFO registers for re-initialization or forced recovery from the bus-off state. This bit can be set to 1 only in the initialization mode.
2. When the CnERC and CnINFO registers have been cleared, the CCERC bit is also cleared to 0 automatically.
3. The CCERC bit can be set to 1 at the same time as a request to change the initialization mode to an operation mode is made.
4. The CCERC bit is read-only in the CAN sleep mode or CAN stop mode.
5. The receive data may be corrupted in case of setting the CCERC bit to (1) immediately after entering the INIT mode from self-test mode.

AL	Bit to set operation in case of arbitration loss
0	Re-transmission is not executed in case of an arbitration loss in the single-shot mode.
1	Re-transmission is executed in case of an arbitration loss in the single-shot mode.

Note The AL bit is valid only in the single-shot mode.

VALID	Valid receive message frame detection bit
0	A valid message frame has not been received since the VALID bit was last cleared to 0.
1	A valid message frame has been received since the VALID bit was last cleared to 0.

Note

1. Detection of a valid receive message frame is not dependent upon storage in the receive message buffer (data frame) or transmit message buffer (remote frame).
2. Clear the VALID bit (0) before changing the initialization mode to an operation mode.
3. If only two CAN nodes are connected to the CAN bus with one transmitting a message frame in the normal mode and the other in the receive-only mode, the VALID bit is not set to 1 before the transmitting node enters the error passive state, because in receive-only mode no acknowledge is generated.
4. To clear the VALID bit, set the Clear VALID bit to 1 first and confirm that the VALID bit is cleared. If it is not cleared, perform clearing processing again.

PSMODE1	PSMODE0	Power save mode
0	0	No power save mode is selected.
0	1	CAN sleep mode
1	0	Setting prohibited
1	1	CAN stop mode

Caution

1. Transition to and from the CAN stop mode must be made via CAN sleep mode. A request for direct transition to and from the CAN stop mode is ignored.
2. The MBON flag of CnGMCTRL must be checked after releasing a power save mode, prior to access the message buffers again.
3. CAN sleep mode requests are kept pending, until cancelled by software or entered on appropriate bus condition (bus idle). Software can check the actual status by reading PSMODE.

OPMODE2	OPMODE1	OPMODE0	Operation mode
0	0	0	No operation mode is selected (CAN module is in the initialization mode).
0	0	1	Normal operation mode
0	1	0	Normal operation mode with automatic block transmission function (normal operation mode with ABT)
0	1	1	Receive-only mode
1	0	0	Single-shot mode
1	0	1	Self-test mode
Other than above			Setting prohibited

Caution Transit to initialization mode or power saving modes may take some time. Be sure to verify the success of mode change by reading the values, before proceeding.

Note The OPMODE0 to OPMODE2 bits are read-only in the CAN sleep mode or CAN stop mode.

(b) CnCTRL write

15	14	13	12	11	10	9	8
Set CCERC	Set AL	0	Set PSMODE1	Set PSMODE0	Set OPMODE2	Set OPMODE1	Set OPMODE0
7	6	5	4	3	2	1	0
0	Clear AL	Clear VALID	Clear PSMODE1	Clear PSMODE0	Clear OPMODE2	Clear OPMODE1	Clear OPMODE0

Set CCERC	Setting of CCERC bit
1	CCERC bit is set to 1.
Other than above	CCERC bit is not changed.

Set AL	Clear AL	Setting of AL bit
0	1	AL bit is cleared to 0.
1	0	AL bit is set to 1.
Other than above		AL bit is not changed.

Clear VALID	Setting of VALID bit
0	VALID bit is not changed.
1	VALID bit is cleared to 0.

Set PSMODE0	Clear PSMODE0	Setting of PSMODE0 bit
0	1	PSMODE0 bit is cleared to 0.
1	0	PSMODE0 bit is set to 1.
Other than above		PSMODE0 bit is not changed.

Set PSMODE1	Clear PSMODE1	Setting of PSMODE1 bit
0	1	PSMODE1 bit is cleared to 0.
1	0	PSMODE1 bit is set to 1.
Other than above		PSMODE1 bit is not changed.

Set OPMODE0	Clear OPMODE0	Setting of OPMODE0 bit
0	1	OPMODE0 bit is cleared to 0.
1	0	OPMODE0 bit is set to 1.
Other than above		OPMODE0 bit is not changed.

Set OPMODE1	Clear OPMODE1	Setting of OPMODE1 bit
0	1	OPMODE1 bit is cleared to 0.
1	0	OPMODE1 bit is set to 1.
Other than above		OPMODE1 bit is not changed.

Set OPMODE2	Clear OPMODE2	Setting of OPMODE2 bit
0	1	OPMODE2 bit is cleared to 0.
1	0	OPMODE2 bit is set to 1.
Other than above		OPMODE2 bit is not changed.

(7) CnLEC - CANn module last error information register

The CnLEC register provides the error information of the CAN protocol.

Access This register can be read/written in 8-bit units.

Address <CnRBaseAddr> + 052_H

Initial Value 00_H. The register is initialized by any reset.

7	6	5	4	3	2	1	0
0	0	0	0	0	LEC2	LEC1	LEC0

Note

1. The contents of the CnLEC register are not cleared when the CAN module changes from an operation mode to the initialization mode.
2. If an attempt is made to write a value other than 00_H to the CnLEC register by software, the access is ignored.

LEC2	LEC1	LEC0	Last CAN protocol error information
0	0	0	No error
0	0	1	Stuff error
0	1	0	Form error
0	1	1	ACK error
1	0	0	Bit error. (The CAN module tried to transmit a recessive-level bit as part of a transmit message (except the arbitration field), but the value on the CAN bus is a dominant-level bit.)
1	0	1	Bit error. (The CAN module tried to transmit a dominant-level bit as part of a transmit message, ACK bit, error frame, or overload frame, but the value on the CAN bus is a recessive-level bit.)
1	1	0	CRC error
1	1	1	Undefined

(8) CnINFO - CANn module information register

The CnINFO register indicates the status of the CAN module.

Access This register is read-only in 8-bit units.

Address <CnRBaseAddr> + 053_H

Initial Value 00_H. The register is initialized by any reset.

7	6	5	4	3	2	1	0
0	0	0	BOFF	TECS1	TECS0	RECS1	RECS0

BOFF	Bus-off state bit
0	Not bus-off state (transmit error counter $\leq$ 255). (The value of the transmit counter is less than 256.)
1	Bus-off state (transmit error counter > 255). (The value of the transmit counter is 256 or more.)

TECS1	TECS0	Transmission error counter status bit
0	0	The value of the transmission error counter is less than that of the warning level (< 96).
0	1	The value of the transmission error counter is in the range of the warning level (96 to 127).
1	0	Undefined
1	1	The value of the transmission error counter is in the range of the error passive or bus-off status ($\geq$ 128).

RECS1	RECS0	Reception error counter status bit
0	0	The value of the reception error counter is less than that of the warning level (< 96).
0	1	The value of the reception error counter is in the range of the warning level (96 to 127).
1	0	Undefined
1	1	The value of the reception error counter is in the error passive range ($\geq$ 128).

(9) CnERC - CANn module error counter register

The CnERC register indicates the count value of the transmission/reception error counter.

Access This register is read-only in 16-bit units.

Address <CnRBaseAddr> + 054_H

Initial Value 0000_H. The register is initialized by any reset.

15	14	13	12	11	10	9	8
REPS	REC6	REC5	REC4	REC3	REC2	REC1	REC0
7	6	5	4	3	2	1	0
TEC7	TEC6	TEC5	TEC4	TEC3	TEC2	TEC1	TEC0

REPS	Reception error passive status bit
0	The reception error counter is not in the error passive range (< 128)
1	The reception error counter is in the error passive range (≥ 128)

REC6 to REC0	Reception error counter bit
0 to 127	Number of reception errors. These bits reflect the status of the reception error counter. The number of errors is defined by the CAN protocol.

Note REC6 to REC0 of the reception error counter are invalid in the reception error passive state (CnINFO.RECS[1:0] = 11_B).

TEC7 to TEC0	Transmission error counter bit
0 to 255	Number of transmission errors. These bits reflect the status of the transmission error counter. The number of errors is defined by the CAN protocol.

Note The TEC7 to TEC0 bits of the transmission error counter are invalid in the bus-off state (CnINFO.BOFF = 1).

(10) CnIE - CANn module interrupt enable register

The CnIE register is used to enable or disable the interrupts of the CAN module.

Access This register can be read/written in 16-bit units.

Address <CnRBaseAddr> + 056_H

Initial Value 0000_H. The register is initialized by any reset.

(a) CnIE read

15	14	13	12	11	10	9	8
0	0	0	0	0	0	0	0
7	6	5	4	3	2	1	0
0	0	CIE5	CIE4	CIE3	CIE2	CIE1	CIE0

CIE5 to CIE0	CAN module interrupt enable bit
0	Output of the interrupt corresponding to interrupt status register CINTSx is disabled.
1	Output of the interrupt corresponding to interrupt status register CINTSx is enabled.

(b) CnIE write

15	14	13	12	11	10	9	8
0	0	Set CIE5	Set CIE4	Set CIE3	Set CIE2	Set CIE1	Set CIE0
7	6	5	4	3	2	1	0
0	0	Clear CIE5	Clear CIE4	Clear CIE3	Clear CIE2	Clear CIE1	Clear CIE0

Set CIE5	Clear CIE5	Setting of CIE5 bit
0	1	CIE5 bit is cleared to 0.
1	0	CIE5 bit is set to 1.
Other than above		CIE5 bit is not changed.

Set CIE4	Clear CIE4	Setting of CIE4 bit
0	1	CIE4 bit is cleared to 0.
1	0	CIE4 bit is set to 1.
Other than above		CIE4 bit is not changed.

Set CIE3	Clear CIE3	Setting of CIE3 bit
0	1	CIE3 bit is cleared to 0.
1	0	CIE3 bit is set to 1.
Other than above		CIE3 bit is not changed.

Set CIE2	Clear CIE2	Setting of CIE2 bit
0	1	CIE2 bit is cleared to 0.
1	0	CIE2 bit is set to 1.
Other than above		CIE2 bit is not changed.

Set CIE1	Clear CIE1	Setting of CIE1 bit
0	1	CIE1 bit is cleared to 0.
1	0	CIE1 bit is set to 1.
Other than above		CIE1 bit is not changed.

Set CIE0	Clear CIE0	Setting of CIE0 bit
0	1	CIE0 bit is cleared to 0.
1	0	CIE0 bit is set to 1.
Other than above		CIE0 bit is not changed.

(11) CnINTS - CANn module interrupt status register

The CnINTS register indicates the interrupt status of the CAN module.

Access This register can be read/written in 16-bit units.

Address <CnRBaseAddr> + 058_H

Initial Value 0000_H. The register is initialized by any reset.

(a) CnINTS read

15	14	13	12	11	10	9	8
0	0	0	0	0	0	0	0
7	6	5	4	3	2	1	0
0	0	CINTS5	CINTS4	CINTS3	CINTS2	CINTS1	CINTS0

CINTS5 to CINTS0	CAN interrupt status bit
0	No related interrupt source event is pending.
1	A related interrupt source event is pending.

Interrupt status bit	Related interrupt source event
CINTS5	Wakeup interrupt from CAN sleep mode[a]
CINTS4	Arbitration loss interrupt
CINTS3	CAN protocol error interrupt
CINTS2	CAN error status interrupt
CINTS1	Interrupt on completion of reception of valid message frame to message buffer m
CINTS0	Interrupt on normal completion of transmission of message frame from message buffer m

a) The CINTS5 bit is set only when the CAN module is woken up from the CAN sleep mode by a CAN bus operation. The CINTS5 bit is not set when the CAN sleep mode has been released by software.

(b) CnINTS write

15	14	13	12	11	10	9	8
0	0	0	0	0	0	0	0
7	6	5	4	3	2	1	0
0	0	Clear CINTS5	Clear CINTS4	Clear CINTS3	Clear CINTS2	Clear CINTS1	Clear CINTS0

Clear CINTS5 to CINTS0	Setting of CINTS5 to CINTS0 bits
0	CINTS5 to CINTS0 bits are not changed.
1	CINTS5 to CINTS0 bits are cleared to 0.

Caution Please clear the status bit of this register with software when the confirmation of each status is necessary in the interrupt processing, because these bits are not cleared automatically.

(12) CnBRP - CANn module bit rate prescaler register

The CnBRP register is used to select the CAN protocol layer basic system clock (f_{TQ}). The communication baud rate is set to the CnBTR register.

Access This register can be read/written in 8-bit units.

Address <CnRBaseAddr> + $05A_H$

Initial Value FF_H. The register is initialized by any reset.

7	6	5	4	3	2	1	0
TQPRS7	TQPRS6	TQPRS5	TQPRS4	TQPRS3	TQPRS2	TQPRS1	TQPRS0

TQPRS7 to TQPRS0	CAN protocol layer base system clock (f_{TQ})
0	$f_{CANMOD}/1$
1	$f_{CANMOD}/2$
n	$f_{CANMOD}/(n+1)$
:	:
255	$f_{CANMOD}/256$ (default value)

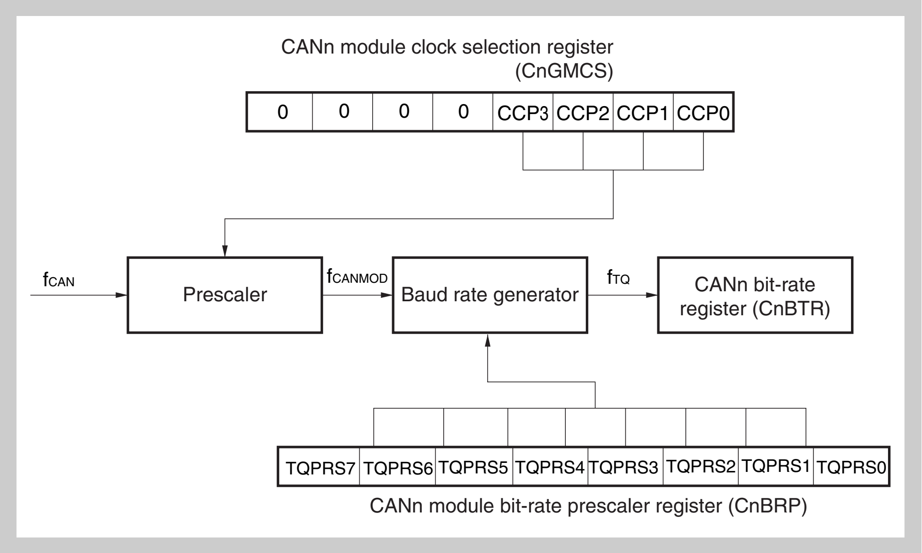


Figure 18-24 CAN module clock

Note

f_{CAN}:	clock supplied to CAN
f_{CANMOD}:	CAN module system clock
f_{TQ}:	CAN protocol layer basic system clock

Caution The CnBRP register can be write-accessed only in the initialization mode.

(13) CnBTR - CANn module bit rate register

The CnBTR register is used to control the data bit time of the communication baud rate.

Access This register can be read/written in 16-bit units.

Address <CnRBaseAddr> + $05C_H$

Initial Value $370F_H$. The register is initialized by any reset.

15	14	13	12	11	10	9	8
0	0	SJW1	SJW0	0	TSEG22	TSEG21	TSEG20
7	6	5	4	3	2	1	0
0	0	0	0	TSEG13	TSEG12	TSEG11	TSEG10

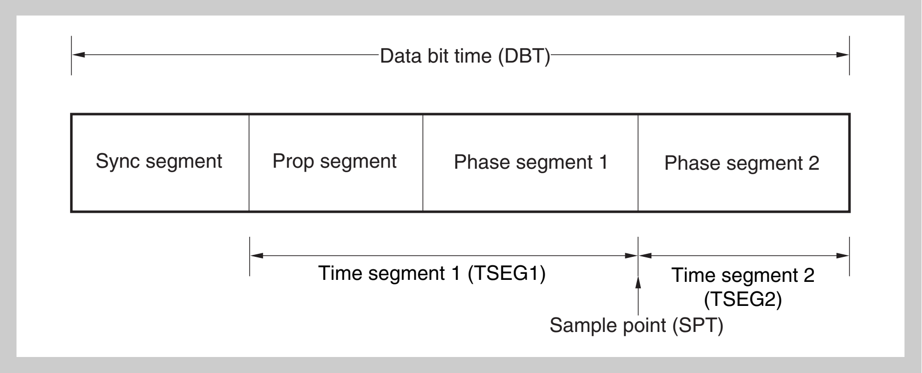


Figure 18-25 Data bit time

SJW1	SJW0	Length of synchronization jump width
0	0	$1T_Q$
0	1	$2T_Q$
1	0	$3T_Q$
1	1	$4T_Q$ (default value)

TSEG22	TSEG21	TSEG20	Length of time segment 2
0	0	0	$1T_Q$
0	0	1	$2T_Q$
0	1	0	$3T_Q$
0	1	1	$4T_Q$
1	0	0	$5T_Q$
1	0	1	$6T_Q$
1	1	0	$7T_Q$
1	1	1	$8T_Q$ (default value)

TSEG13	TSEG12	TSEG11	TSEG10	Length of time segment 1
0	0	0	0	Setting prohibited
0	0	0	1	$2T_Q$[a]
0	0	1	0	$3T_Q$[a]
0	0	1	1	$4T_Q$
0	1	0	0	$5T_Q$
0	1	0	1	$6T_Q$
0	1	1	0	$7T_Q$
0	1	1	1	$8T_Q$
1	0	0	0	$9T_Q$
1	0	0	1	$10T_Q$
1	0	1	0	$11T_Q$
1	0	1	1	$12T_Q$
1	1	0	0	$13T_Q$
1	1	0	1	$14T_Q$
1	1	1	0	$15T_Q$
1	1	1	1	$16T_Q$ (default value)

[a] This setting must not be made when the CnBRP register = 00_H

Note $T_Q = 1/f_{TQ}$ (f_{TQ}: CAN protocol layer basic system clock)

(14) CnLIPT - CANn module last in-pointer register

The CnLIPT register indicates the number of the message buffer in which a data frame or a remote frame was last stored.

Access This register is read-only in 8-bit units.

Address <CnRBaseAddr> + $05E_H$

Initial Value Undefined.

7	6	5	4	3	2	1	0
LIPT7	LIPT6	LIPT5	LIPT4	LIPT3	LIPT2	LIPT1	LIPT0

LIPT7 to LIPT0	Last in-pointer register (CnLIPT)
0 to 31	When the CnLIPT register is read, the contents of the element indexed by the last in-pointer (LIPT) of the receive history list are read. These contents indicate the number of the message buffer in which a data frame or a remote frame was last stored.

Note The read value of the CnLIPT register is undefined if a data frame or a remote frame has never been stored in the message buffer. If the RHPM bit of the CnRGPT register is set to 1 after the CAN module has changed from the initialization mode to an operation mode, therefore, the read value of the CnLIPT register is undefined.

(15) CnRGPT - CANn module receive history list register

The CnRGPT register is used to read the receive history list.

Access This register can be read/written in 16-bit units.

Address <CnRBaseAddr> + 060_H

Initial Value $xx02_H$. The register is initialized by any reset.

(a) CnRGPT read

15	14	13	12	11	10	9	8
RGPT7	RGPT6	RGPT5	RGPT4	RGPT3	RGPT2	RGPT1	RGPT0
7	6	5	4	3	2	1	0
0	0	0	0	0	0	RHPM	ROVF

RGPT7 to RGPT0	Receive history list read pointer
0 to 31	When the CnRGPT register is read, the contents of the element indexed by the receive history list get pointer (RGPT) of the receive history list are read. These contents indicate the number of the message buffer in which a data frame or a remote frame has been stored.

RHPM[a]	Receive history list pointer match
0	The receive history list has at least one message buffer number that has not been read.
1	The receive history list has no message buffer numbers that have not been read.

a) The read value of the RGPT0 to RGPT7 bits is invalid when the RHPM bit = 1.

ROVF[a]	Receive history list overflow bit
0	All the message buffer numbers that have not been read are preserved. All the numbers of the message buffers in which a new data frame or remote frame has been received and stored are recorded to the receive history list (the receive history list has a vacant element).
1	At least 23 entries have been stored since the host processor has serviced the RHL last time (i.e. read CnRGPT). The first 22 entries are sequentially stored while the last entry can have been overwritten whenever newly received message is stored because all buffer numbers are stored at position LIPT-1 when ROVF bit is set. Thus the sequence of receptions can not be recovered completely now.

a) If ROVF is set, RHPM is no longer cleared on message storage, but RHPM is still set, if all entries of CnRGPT are read by software.

(b) CnRGPT write

15	14	13	12	11	10	9	8
0	0	0	0	0	0	0	0

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	Clear ROVF

Clear ROVF	Setting of ROVF bit
0	ROVF bit is not changed.
1	ROVF bit is cleared to 0.

(16) CnLOPT - CANn module last out-pointer register

The CnLOPT register indicates the number of the message buffer to which a data frame or a remote frame was transmitted last.

Access This register is read-only in 8-bit units.

Address <CnRBaseAddr> + 062_H

Initial Value Undefined

7	6	5	4	3	2	1	0
LOPT7	LOPT6	LOPT5	LOPT4	LOPT3	LOPT2	LOPT1	LOPT0

LOPT7 to LOPT0	Last out-pointer of transmit history list (LOPT)
0 to 31	When the CnLOPT register is read, the contents of the element indexed by the last out-pointer (LOPT) of the receive history list are read. These contents indicate the number of the message buffer to which a data frame or a remote frame was transmitted last.

Note The value read from the CnLOPT register is undefined if a data frame or remote frame has never been transmitted from a message buffer. If the CnTGPT.THPM bit is set to 1 after the CAN module has changed from the initialization mode to an operation mode, therefore, the read value of the CnLOPT register is undefined.

(17) CnTGPT - CANn module transmit history list register

The CnTGPT register is used to read the transmit history list.

Access This register can be read/written in 16-bit units.

Address <CnRBaseAddr> + 064_H

Initial Value $xx02_H$. The register is initialized by any reset.

(a) CnTGPT read

15	14	13	12	11	10	9	8
TGPT7	TGPT6	TGPT5	TGPT4	TGPT3	TGPT2	TGPT1	TGPT0
7	6	5	4	3	2	1	0
0	0	0	0	0	0	THPM	TOVF

TGPT7 to TGPT0	Transmit history list read pointer
0 to 31	When the CnTGPT register is read, the contents of the element indexed by the read pointer (TGPT) of the transmit history list are read. These contents indicate the number of the message buffer to which a data frame or a remote frame was transmitted last.

THPM[a]	Transmit history pointer match
0	The transmit history list has at least one message buffer number that has not been read.
1	The transmit history list has no message buffer numbers that have not been read.

a) The read value of the TGPT0 to TGPT7 bits is invalid when the THPM bit = 1.

TOVF[a]	Transmit history list overflow bit
0	All the message buffer numbers that have not been read are preserved. All the numbers of the message buffers to which a new data frame or remote frame has been transmitted are recorded to the transmit history list (the transmit history list has a vacant element).
1	At least 7 entries have been stored since the host processor has serviced the THL last time (i.e. read CnTGPT). The first 6 entries are sequentially stored while the last entry can have been overwritten whenever a message is newly transmitted because all buffer numbers are stored at position LOPT-1 when TOVF bit is set. Thus the sequence of transmissions can not be recovered completely now.

a) If TOVF is set, THPM is no longer cleared on message transmission, but THPM is still set, if all entries of CnTGPT are read by software.

Note Transmission from message buffers 0 to 7 is not recorded to the transmit history list in the normal operation mode with ABT.

(b) CnTGPT write

15	14	13	12	11	10	9	8
0	0	0	0	0	0	0	0

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	Clear TOVF

Clear TOVF	Setting of TOVF bit
0	TOVF bit is not changed.
1	TOVF bit is cleared to 0.

(18) CnTS - CANn module time stamp register

The CnTS register is used to control the time stamp function.

Access This register can be read/written in 16-bit units.

Address <CnRBaseAddr> + 066_H

Initial Value 0000_H. The register is initialized by any reset.

(a) CnTS read

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

Note The lock function of the time stamp function must not be used when the CAN module is in the normal operation mode with ABT.

TSLOCK	Time stamp lock function enable bit
0	Time stamp lock function stopped. The TSOUT signal is toggled each time the selected time stamp capture event occurs.
1	Time stamp lock function enabled. The TSOUT signal is toggled each time the selected time stamp capture event occurs. However, the TSOUT output signal is locked when a data frame has been correctly received to message buffer 0[a].

[a] The TSEN bit is automatically cleared to 0.

TSSEL	Time stamp capture event selection bit
0	The time capture event is SOF.
1	The time stamp capture event is the last bit of EOF.

TSEN	TSOUT operation setting bit
0	TSOUT toggle operation is disabled.
1	TSOUT toggle operation is enabled.

(b) CnTS write

15	14	13	12	11	10	9	8
0	0	0	0	0	Set TSLOCK	Set TSSEL	Set TSEN
7	6	5	4	3	2	1	0
0	0	0	0	0	Clear TSLOCK	Clear TSSEL	Clear TSEN

Set TSLOCK	Clear TSLOCK	Setting of TSLOCK bit
0	1	TSLOCK bit is cleared to 0.
1	0	TSLOCK bit is set to 1.
Other than above		TSLOCK bit is not changed.

Set TSSEL	Clear TSSEL	Setting of TSSEL bit
0	1	TSSEL bit is cleared to 0.
1	0	TSSEL bit is set to 1.
Other than above		TSSEL bit is not changed.

Set TSEN	Clear TSEN	Setting of TSEN bit
0	1	TSEN bit is cleared to 0.
1	0	TSEN bit is set to 1.
Other than above		TSEN bit is not changed.

(19) CnMDATAxm, CnMDATAzm - CANn message data byte register (x = 0 to 7, z = 01, 23, 45, 67)

The CnMDATAxm, CnMDATAzm registers are used to store the data of a transmit/receive message.

Access The CnMDATAzm registers can be read/written in 16-bit units.
The CnMDATAxm registers can be read/written in 8-bit units.

Address Refer to *“CAN registers overview“ on page 574.*

Initial Value Undefined.

CnMDATA01m

15	14	13	12	11	10	9	8
MDATA0115	MDATA0114	MDATA0113	MDATA0112	MDATA0111	MDATA0110	MDATA019	MDATA018
7	6	5	4	3	2	1	0
MDATA017	MDATA016	MDATA015	MDATA014	MDATA013	MDATA012	MDATA011	MDATA010

CnMDATA0m

7	6	5	4	3	2	1	0
MDATA07	MDATA06	MDATA05	MDATA04	MDATA03	MDATA02	MDATA01	MDATA00

CnMDATA1m

7	6	5	4	3	2	1	0
MDATA17	MDATA16	MDATA15	MDATA14	MDATA13	MDATA12	MDATA11	MDATA1

CnMDATA23m

15	14	13	12	11	10	9	8
MDATA2315	MDATA2314	MDATA2313	MDATA2312	MDATA2311	MDATA2310	MDATA239	MDATA238
7	6	5	4	3	2	1	0
MDATA237	MDATA236	MDATA235	MDATA234	MDATA233	MDATA232	MDATA231	MDATA230

CnMDATA2m

7	6	5	4	3	2	1	0
MDATA27	MDATA26	MDATA25	MDATA24	MDATA23	MDATA22	MDATA21	MDATA20

CnMDATA3m

7	6	5	4	3	2	1	0
MDATA37	MDATA36	MDATA35	MDATA34	MDATA33	MDATA32	MDATA31	MDATA30

CnMDATA45m

15	14	13	12	11	10	9	8
MDATA4515	MDATA4514	MDATA4513	MDATA4512	MDATA4511	MDATA4510	MDATA459	MDATA458
7	6	5	4	3	2	1	0
MDATA457	MDATA456	MDATA455	MDATA454	MDATA453	MDATA452	MDATA451	MDATA450

CnMDATA4m

7	6	5	4	3	2	1	0
MDATA47	MDATA46	MDATA45	MDATA44	MDATA43	MDATA42	MDATA41	MDATA40

CnMDATA5m

7	6	5	4	3	2	1	0
MDATA57	MDATA56	MDATA55	MDATA54	MDATA53	MDATA52	MDATA51	MDATA50

CnMDATA67m

15	14	13	12	11	10	9	8
MDATA6715	MDATA6714	MDATA6713	MDATA6712	MDATA6711	MDATA6710	MDATA679	MDATA678
7	6	5	4	3	2	1	0
MDATA677	MDATA676	MDATA675	MDATA674	MDATA673	MDATA672	MDATA671	MDATA670

CnMDATA6m

7	6	5	4	3	2	1	0
MDATA67	MDATA66	MDATA65	MDATA64	MDATA63	MDATA62	MDATA61	MDATA60

CnMDATA7m

7	6	5	4	3	2	1	0
MDATA77	MDATA76	MDATA75	MDATA74	MDATA73	MDATA72	MDATA71	MDATA70

(20) CnMDLCm - CANn message data length register m

The CnMDLCm register is used to set the number of bytes of the data field of a message buffer.

Access This register can be read/written in 8-bit units.

Address Refer to *“CAN registers overview“ on page 574.*

Initial Value 0000xxxx$_B$. The register is initialized by any reset.

7	6	5	4	3	2	1	0
0	0	0	0	MDLC3	MDLC2	MDLC1	MDLC0

MDLC3	MDLC2	MDLC1	MDLC0	Data length of transmit/receive message
0	0	0	0	0 bytes
0	0	0	1	1 byte
0	0	1	0	2 bytes
0	0	1	1	3 bytes
0	1	0	0	4 bytes
0	1	0	1	5 bytes
0	1	1	0	6 bytes
0	1	1	1	7 bytes
1	0	0	0	8 bytes
1	0	0	1	Setting prohibited (If these bits are set during transmission, 8-byte data is transmitted regardless of the set DLC value when a data frame is transmitted. However, the DLC actually transmitted to the CAN bus is the DLC value set to this register.)[Note]
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

Note The data and DLC value actually transmitted to CAN bus are as follows.

Type of transmit frame	Length of transmit data	DLC transmitted
Data frame	Number of bytes specified by DLC (However, 8 bytes if DLC ≥ 8)	MDLC3 to MDLC0 bits
Remote frame	0 bytes	

Caution

1. Be sure to set bits 7 to 4 to 0000$_B$.
2. Receive data is stored in as many CnMDATAxm register as the number of bytes (however, the upper limit is 8) corresponding to DLC of the received frame. The CnMDATAxm register in which no data is stored is undefined.

(21) CnMCONFm - CANn message configuration register m

The CnMCONFm register is used to specify the type of the message buffer and to set a mask.

Access This register can be read/written in 8-bit units.

Address Refer to *“CAN registers overview“ on page 574.*

Initial Value Undefined.

7	6	5	4	3	2	1	0
OWS	RTR	MT2	MT1	MT0	0	0	MA0

OWS	Overwrite control bit
0	The message buffer that has already received a data frame[a] is not overwritten by a newly received data frame. The newly received data frame is discarded.
1	The message buffer that has already received a data frame[a] is overwritten by a newly received data frame.

[a] The “message buffer that has already received a data frame” is a receive message buffer whose the CnMCTRLm.DN bit has been set to 1.

Note A remote frame is received and stored, regardless of the setting of OWS and DN. A remote frame that satisfies the other conditions (ID matches, RTR = 0, TRQ = 0) is always received and stored in the corresponding message buffer (interrupt generated, DN flag set, MDLC[3:0] updated, and recorded to the receive history list).

RTR	Remote frame request bit[a]
0	Transmit a data frame.
1	Transmit a remote frame.

[a] The RTR bit specifies the type of message frame that is transmitted from a message buffer defined as a transmit message buffer. Even if a valid remote frame has been received, the RTR bit of the transmit message buffer that has received the frame remains cleared to 0. Even if a remote frame whose ID matches has been received from the CAN bus with the RTR bit of the transmit message buffer set to 1 to transmit a remote frame, that remote frame is not received or stored (interrupt generated, DN flag set, the MDLC0 to MDLC3 bits updated, and recorded to the receive history list).

MT2	MT1	MT0	Message buffer type setting bit
0	0	0	Transmit message buffer
0	0	1	Receive message buffer (no mask setting)
0	1	0	Receive message buffer (mask 1 set)
0	1	1	Receive message buffer (mask 2 set)
1	0	0	Receive message buffer (mask 3 set)
1	0	1	Receive message buffer (mask 4 set)
Other than above			Setting prohibited

MA0	Message buffer assignment bit
0	Message buffer not used.
1	Message buffer used.

Caution Be sure to write 0 to bits 2 and 1.

(22) CnMIDLm, CnMIDHm - CANn message ID register m

The CnMIDLm and CnMIDHm registers are used to set an identifier (ID).

Access These registers can be read/written in 16-bit units.

Address Refer to *“CAN registers overview“ on page 574.*

Initial Value Undefined.

CnMIDLm

15	14	13	12	11	10	9	8
ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8
7	6	5	4	3	2	1	0
ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0

CnMIDHm

15	14	13	12	11	10	9	8
IDE	0	0	ID28	ID27	ID26	ID25	ID24
7	6	5	4	3	2	1	0
ID23	ID22	ID21	ID20	ID19	ID18	ID17	ID16

IDE	Format mode specification bit
0	Standard format mode (ID28 to ID18: 11 bits)[a]
1	Extended format mode (ID28 to ID0: 29 bits)

[a] The ID17 to ID0 bits are not used.

ID28 to ID0	Message ID
ID28 to ID18	Standard ID value of 11 bits (when IDE = 0)
ID28 to ID0	Extended ID value of 29 bits (when IDE = 1)

Caution

1. Be sure to write 0 to bits 14 and 13 of the CnMIDHm register.
2. Be sure to align the ID value according to the given bit positions into this registers. Note that for standard ID, the ID value must be shifted to fit into ID28 to ID11 bit positions.

(23) CnMCTRLm - CANn message control register m

The CnMCTRLm register is used to control the operation of the message buffer.

Access This register can be read/written in 16-bit units.

Address Refer to *“CAN registers overview“ on page 574.*

Initial Value 00x0 0000 0000 0000_B. The register is initialized by any reset.

(a) CnMCTRLm read

15	14	13	12	11	10	9	8
0	0	MUC	0	0	0	0	0
7	6	5	4	3	2	1	0
0	0	0	MOW	IE	DN	TRQ	RDY

MUC[a]	Bit indicating that message buffer data is being updated
0	The CAN module is not updating the message buffer (reception and storage).
1	The CAN module is updating the message buffer (reception and storage).

[a] The MUC bit is undefined until the first reception and storage is performed.

MOW[a]	Message buffer overwrite status bit
0	The message buffer is not overwritten by a newly received data frame.
1	The message buffer is overwritten by a newly received data frame.

[a] The MOW bit is not set to 1 even if a remote frame is received and stored in the transmit message buffer with the DN bit = 1.

IE	Message buffer interrupt request enable bit
0	Receive message buffer: Valid message reception completion interrupt disabled. Transmit message buffer: Normal message transmission completion interrupt disabled.
1	Receive message buffer: Valid message reception completion interrupt enabled. Transmit message buffer: Normal message transmission completion interrupt enabled.

DN	Message buffer data update bit
0	A data frame or remote frame is not stored in the message buffer.
1	A data frame or remote frame is stored in the message buffer.

TRQ	Message buffer transmission request bit
0	No message frame transmitting request that is pending or being transmitted is in the message buffer.
1	The message buffer is holding transmission of a message frame pending or is transmitting a message frame.

RDY	Message buffer ready bit
0	The message buffer can be written by software. The CAN module cannot write to the message buffer.
1	Writing the message buffer by software is ignored (except a write access to the RDY, TRQ, DN, and MOW bits). The CAN module can write to the message buffer.

(b) CnMCTRLm write

15	14	13	12	11	10	9	8
0	0	0	0	Set IE	0	Set TRQ	Set RDY
7	6	5	4	3	2	1	0
0	0	0	Clear MOW	Clear IE	Clear DN	Clear TRQ	Clear RDY

Clear MOW	Setting of MOW bit
0	MOW bit is not changed.
1	MOW bit is cleared to 0.

Set IE	Clear IE	Setting of IE bit
0	1	IE bit is cleared to 0.
1	0	IE bit is set to 1.
Other than above		IE bit is not changed.

Clear DN	Setting of DN bit
1	DN bit is cleared to 0.
0	DN bit is not changed.

Set TRQ	Clear TRQ	Setting of TRQ bit
0	1	TRQ bit is cleared to 0.
1	0	TRQ bit is set to 1.
Other than above		TRQ bit is not changed.

Set RDY	Clear RDY	Setting of RDY bit
0	1	RDY bit is cleared to 0.
1	0	RDY bit is set to 1.
Other than above		RDY bit is not changed.

Set IE bit and RDY bit always separately.

Caution

1. Do not set the DN bit to 1 by software. Be sure to write 0 to bit 10.
2. Do not set the TRQ bit and the RDY bit (1) at the same time. Set the RDY bit (1) before setting the TRQ bit.
3. Do not clear the RDY bit (0) during message transmission. Follow the transmission abort process about clearing the RDY bit (0) for redefinition of the message buffer.
4. Clear again when RDY bit is not cleared even if this bit is cleared.
5. Be sure that RDY is cleared before writing to the other message buffer registers, by checking the status of the RDY bit.

18.8 CAN Controller Initialization

18.8.1 Initialization of CAN module

Before CAN module operation is enabled, the CAN module system clock needs to be determined by setting the CCP[3:0] bits of the CnGMCS register by software. Do not change the setting of the CAN module system clock after CAN module operation is enabled.

The CAN module is enabled by setting the GOM bit of the CnGMCTRL register.

For the procedure of initializing the CAN module, refer to *“Operation of CAN Controller“ on page 656*.

18.8.2 Initialization of message buffer

After the CAN module is enabled, the message buffers contain undefined values. A minimum initialization for all the message buffers, even for those not used in the application, is necessary before switching the CAN module from the initialization mode to one of the operation modes.

- Clear the RDY, TRQ, and DN bits of all CnMCTRLm registers to 0.
- Clear the MA0 bit of all CnMCONFm registers to 0.

18.8.3 Redefinition of message buffer

Redefining a message buffer means changing the ID and control information of the message buffer while a message is being received or transmitted, without affecting other transmission/reception operations.

(1) To redefine message buffer in initialization mode

Place the CAN module in the initialization mode once and then change the ID and control information of the message buffer in the initialization mode. After changing the ID and control information, set the CAN module to an operation mode.

(2) To redefine message buffer during reception

Perform redefinition as shown in *Figure 18-38*.

(3) To redefine message buffer during transmission

To rewrite the contents of a transmit message buffer to which a transmission request has been set, perform transmission abort processing (see *“Transmission abort process except for in normal operation mode with automatic block transmission (ABT)“ on page 635* and *“Transmission abort process except for ABT transmission in normal operation mode with automatic block transmission (ABT)“ on page 635*). Confirm that transmission has been aborted or completed, and then redefine the message buffer. After redefining the transmit message buffer, set a transmission request using the procedure described below. When setting a transmission request to a message buffer that has been redefined without aborting the transmission in progress, however, the 1-bit wait time is not necessary.

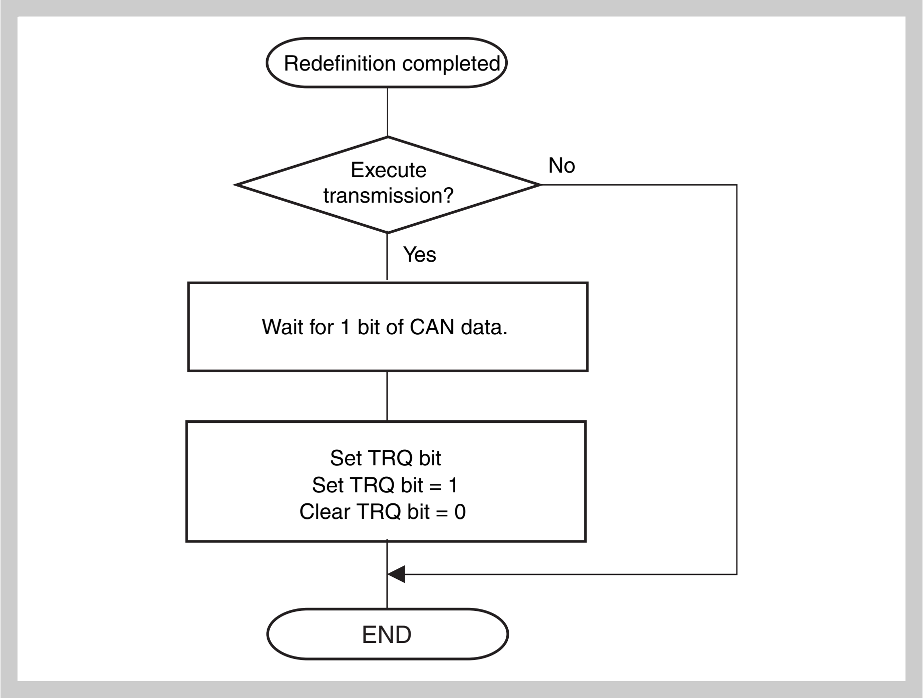


Figure 18-26 Setting transmission request (TRQ) to transmit message buffer after redefinition

Caution

1. When a message is received, reception filtering is performed in accordance with the ID and mask set to each receive message buffer. If the procedure in *Figure 18-38 on page 659* is not observed, the contents of the message buffer after it has been redefined may contradict the result of reception (result of reception filtering). If this happens, check that the ID and IDE received first and stored in the message buffer following redefinition are those stored after the message buffer has been redefined. If no ID and IDE are stored after redefinition, redefine the message buffer again.
2. When a message is transmitted, the transmission priority is checked in accordance with the ID, IDE, and RTR bits set to each transmit message buffer to which a transmission request was set. The transmit message buffer having the highest priority is selected for transmission. If the procedure in *Figure 18-26 on page 618* is not observed, a message with an ID not having the highest priority may be transmitted after redefinition.

18.8.4 Transition from initialization mode to operation mode

The CAN module can be switched to the following operation modes.

- Normal operation mode
- Normal operation mode with ABT
- Receive-only mode
- Single-shot mode
- Self-test mode

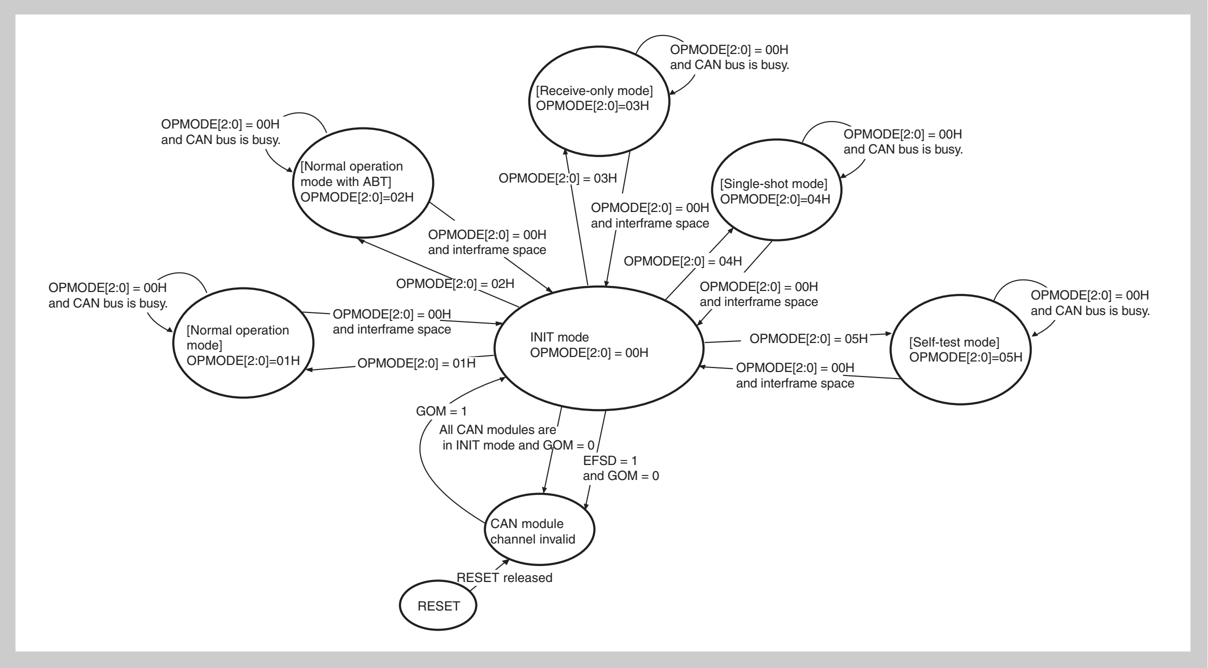


Figure 18-27 Transition to operation modes

The transition from the initialization mode to an operation mode is controlled by the bit string OPMODE[2:0] in the CnCTRL register.

Changing from one operation mode into another requires shifting to the initialization mode in between. Do not change one operation mode to another directly; otherwise the operation will not be guaranteed.

Requests for transition from an operation mode to the initialization mode are held pending when the CAN bus is not in the interframe space (i.e., frame reception or transmission is in progress), and the CAN module enters the initialization mode at the first bit in the interframe space (the values of the OPMODE[2:0] bits are changed to $000_{B)}$. After issuing a request to change the mode to the initialization mode, read the OPMODE[2:0] bits until their value becomes 000_B to confirm that the module has entered the initialization mode (see *Figure 18-36 on page 657*).

18.8.5 Resetting error counter CnERC of CAN module

If it is necessary to reset the CAN module error counter CnERC and CAN module information register CnINFO when re-initialization or forced recovery from the bus-off status is made, set the CCERC bit of the CnCTRL register to 1 in the initialization mode. When this bit is set to 1, the CnERC and CnINFO registers are cleared to their default values.

18.9 Message Reception

18.9.1 Message reception

In all the operation modes, the complete message buffer area is analyzed to find a suitable buffer to store a newly received message. All message buffers satisfying the following conditions are included in that evaluation (RX-search process).

- Used as a message buffer
(MA0 bit of CnMCONFm register set to 1.)
- Set as a receive message buffer
(MT[2:0] bits of CnMCONFm register are set to 001_B, 010_B, 011_B, 100_B, or 101_B.)
- Ready for reception
(RDY bit of CnMCTRLm register is set to 1.)

When two or more message buffers of the CAN module receive a message, the message is stored according to the priority explained below. The message is always stored in the message buffer with the highest priority, not in a message buffer with a low priority. For example, when an unmasked receive message buffer and a receive message buffer linked to mask 1 have the same ID, the received message is not stored in the message buffer linked to mask 1, even if that message buffer has not received a message and a message has already been received in the unmasked receive message buffer. In other words, when a condition has been set in two or more message buffers with different priorities, the message buffer with the highest priority always stores the message; the message is not stored in message buffers with a lower priority. This also applies when the message buffer with the highest priority is unable to store a message (i.e., when DN = 1 indicating that a message has already been received, but rewriting is disabled because OWS = 0). In this case, the message is not actually stored in the candidate message buffer with the highest priority, but neither is it stored in a message buffer with a lower priority.

Table 18-24 MBRB priorities

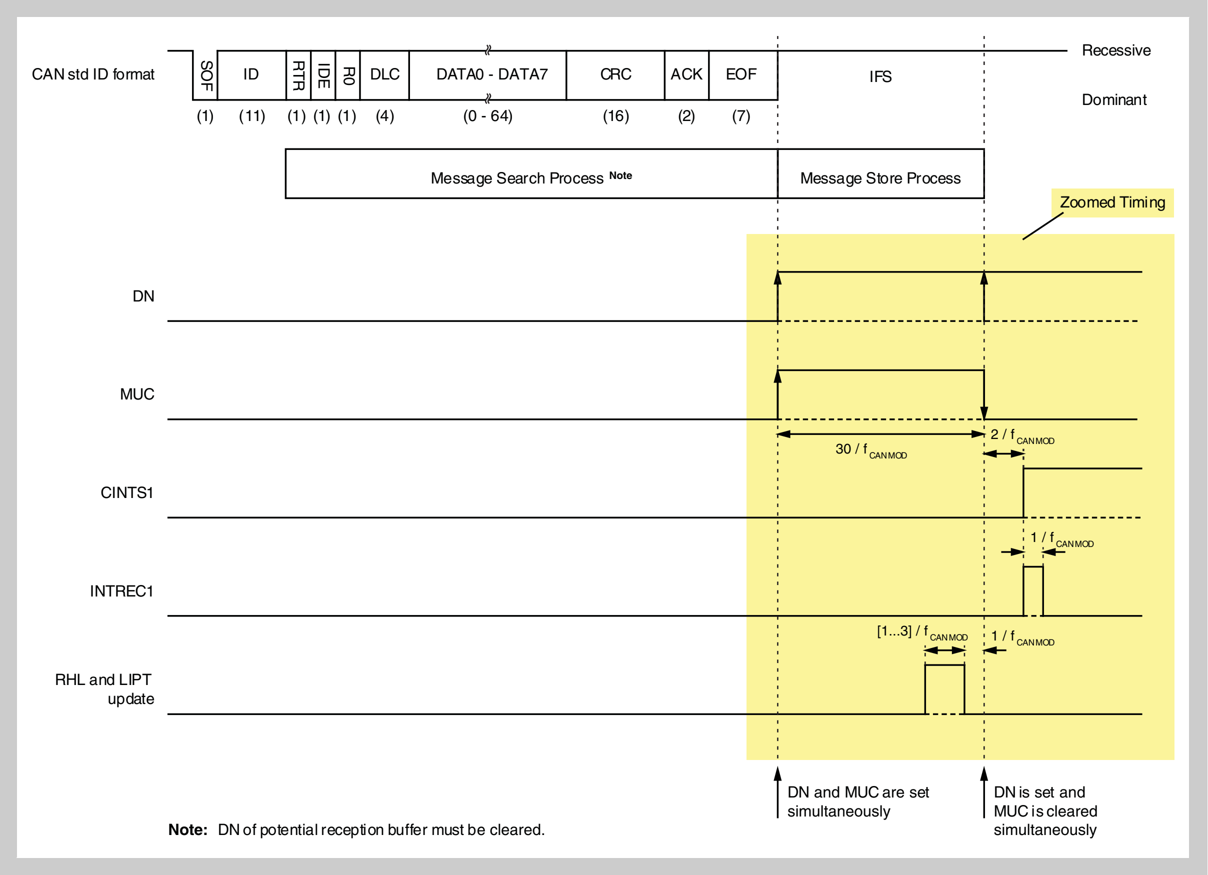
Priority	Storing condition if same ID is set	
1 (high)	Unmasked message buffer	DN bit = 0
		DN bit = 1 and OWS bit = 1
2	Message buffer linked to mask 1	DN bit = 0
		DN bit = 1 and OWS bit = 1
3	Message buffer linked to mask 2	DN bit = 0
		DN bit = 1 and OWS bit = 1
4	Message buffer linked to mask 3	DN bit = 0
		DN bit = 1 and OWS bit = 1
5 (low)	Message buffer linked to mask 4	DN bit = 0
		DN bit = 1 and OWS bit = 1

18.9.2 Receive data read

To keep data consistency when reading CAN message buffers, perform the data reading according to *Figure 18-49 on page 672* to *Figure 18-52 on page 676*.

During message reception, the CAN module sets DN of the CnMCTRLm register two times: at the beginning of the storage process of data to the message buffer, and again at the end of this storage process. During this storage process, the MUC bit of the CnMCTRLm register of the message buffer is set. (Refer to *Figure 18-28 on page 622*.)

The receive history list is also updated just before the storgage process. In addition, during storage process (MUC = 1), the RDY bit of the CnMCTRL register of the message buffer is locked to avoid the coincidental data WR by CPU. Note the storage process may be disturbed (delayed) when the CPU accesses the message buffer.



Note: DN of potential reception buffer must be cleared.

Figure 18-28 DN and MUC bit setting period (for standard ID format)

Note If a message shall be stored in a message buffer, the DN bit of this buffer must be cleared before the Message Search Process is started, i.e., right after the ID of the frame is on the bus. In worst case, this happens 15 CAN bits after EOF of the previous frame. Consider to use more than one Message Buffer for reception of a frame, if CAN frames are appearing back-to-back on the bus and none shall be lost.

18.9.3 Receive history list function

The receive history list (RHL) function records in the receive history list the number of the receive message buffer in which each data frame or remote frame was received and stored. The RHL consists of storage elements equivalent to up to 23 messages, the last in-message pointer (LIPT) with the corresponding CnLIPT register and the receive history list get pointer (RGPT) with the corresponding CnRGPT register.

The RHL is undefined immediately after the transition of the CAN module from the initialization mode to one of the operation modes.

The CnLIPT register holds the contents of the RHL element indicated by the value of the LIPT pointer minus 1. By reading the CnLIPT register, therefore, the number of the message buffer that received and stored a data frame or remote frame first can be checked. The LIPT pointer is utilized as a write pointer that indicates to what part of the RHL a message buffer number is recorded. Any time a data frame or remote frame is received and stored, the corresponding message buffer number is recorded to the RHL element indicated by the LIPT pointer. Each time recording to the RHL has been completed, the LIPT pointer is automatically incremented. In this way, the number of the message buffer that has received and stored a frame will be recorded chronologically.

The RGPT pointer is utilized as a read pointer that reads a recorded message buffer number from the RHL. This pointer indicates the first RHL element that the CPU has not read yet. By reading the CnRGPT register by software, the number of a message buffer that has received and stored a data frame or remote frame can be read. Each time a message buffer number is read from the CnRGPT register, the RGPT pointer is automatically incremented.

If the value of the RGPT pointer matches the value of the LIPT pointer, the RHPM bit (receive history list pointer match) of the CnRGPT register is set to 1. This indicates that no message buffer number that has not been read remains in the RHL. If a new message buffer number is recorded, the LIPT pointer is incremented and because its value no longer matches the value of the RGPT pointer, the RHPM bit is cleared. In other words, the numbers of the unread message buffers exist in the RHL.

If the LIPT pointer is incremented and matches the value of the RGPT pointer minus 1, the ROVF bit (receive history list overflow) of the CnRGPT register is set to 1. This indicates that the RHL is full of numbers of message buffers that have not been read. When further message reception and storing occur, the last recorded message buffer number is overwritten by the number of the message buffer that received and stored the newly received message. In this case, after the ROVF bit has been set (1), the recorded message buffer numbers in the RHL do not completely reflect the chronological order. However messages itself are not lost and can be located by CPU search in message buffer memory with the help of the DN-bit.

Caution If the history list is in the overflow condition (ROVF is set), reading the history list contents is still possible, until the history list is empty (indicated by RHPM flag set). Nevertheless, the history list remains in the overflow condition, until ROVF is cleared by software. If ROVF is not cleared, the RHPM flag will also not be updated (cleared) upon a message storage of newly received frame. This may lead to the situation, that RHPM indicates an empty history list, although a reception has taken place, while the history list is in the overflow state (ROVF and RHPM are set).

As long as the RHL contains 23 or less entries the sequence of occurrence is maintained. If more receptions occur without reading the RHL by the host processor, complete sequence of receptions can not be recovered.

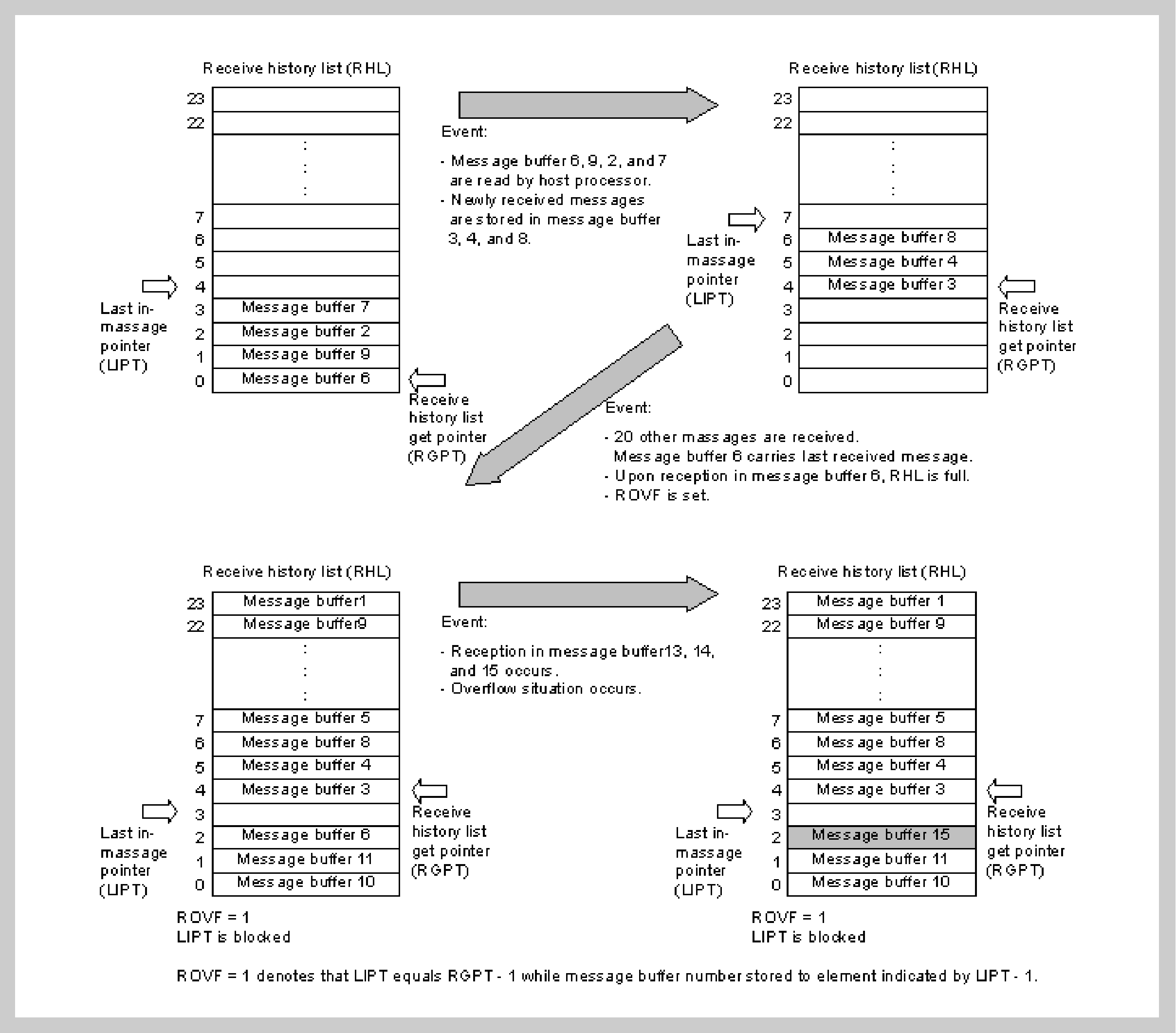

Receive history list (RHL)
23
22
:
7
6
5
4
3 Message buffer 7
2 Message buffer 2
1 Message buffer 9
0 Message buffer 6
Last in-massage pointer (LIPT)
Receive history list get pointer (RGPT)
Event:
- Message buffer 6, 9, 2, and 7 are read by host processor.
- Newly received messages are stored in message buffer 3, 4, and 8.
Receive history list (RHL)
23
22
:
7
6 Message buffer 8
5 Message buffer 4
4 Message buffer 3
3
2
1
0
Last in-massage pointer (LIPT)
Receive history list get pointer (RGPT)
Event:
- 20 other massages are received. Message buffer 6 carries last received message.
- Upon reception in message buffer 6, RHL is full.
- ROVF is set.
Receive history list (RHL)
23 Message buffer1
22 Message buffer9
:
7 Message buffer 5
6 Message buffer 8
5 Message buffer 4
4 Message buffer 3
3
2 Message buffer 6
1 Message buffer 11
0 Message buffer 10
Last in-massage pointer (LIPT)
Receive history list get pointer (RGPT)
ROVF = 1
LIPT is blocked
Event:
- Reception in message buffer13, 14, and 15 occurs.
- Overflow situation occurs.
Receive history list (RHL)
23 Message buffer 1
22 Message buffer 9
:
7 Message buffer 5
6 Message buffer 8
5 Message buffer 4
4 Message buffer 3
3
2 Message buffer 15
1 Message buffer 11
0 Message buffer 10
Last in-massage pointer (LIPT)
Receive history list get pointer (RGPT)
ROVF = 1
LIPT is blocked
ROVF = 1 denotes that LIPT equals RGPT - 1 while message buffer number stored to element indicated by LIPT - 1.

Figure 18-29 Receive history list

18.9.4 Mask function

For any message buffer, which is used for reception, the assignment to one of four global reception masks (or no mask) can be selected.

By using the mask function, the message ID comparison can be reduced by masked bits, herewith allowing the reception of several different IDs into one buffer.

While the mask function is in effect, an identifier bit that is defined to be 1 by a mask in the received message is not compared with the corresponding identifier bit in the message buffer.

However, this comparison is performed for any bit whose value is defined as 0 by the mask.

For example, let us assume that all messages that have a standard-format ID, in which bits ID27 to ID25 are 0 and bits ID24 and ID22 are 1, are to be stored in message buffer 14. The procedure for this example is shown below.

1. Identifier to be stored in message buffer

ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21	ID20	ID19	ID18
x	0	0	0	1	x	1	x	x	x	x

**2. Identifier to be configured in message buffer 14 (example)
(Using CnMIDL14 and CnMIDH14 registers)**

ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21	ID20	ID19	ID18
x	0	0	0	1	x	1	x	x	x	x
ID17	ID16	ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7
x	x	x	x	x	x	x	x	x	x	x
ID6	ID5	ID4	ID3	ID2	ID1	ID0				
x	x	x	x	x	x	x				

Note

1. ID with the ID27 to ID25 bits cleared to 0 and the ID24 and ID22 bits set to 1 is registered (initialized) to message buffer 14.
2. Message buffer 14 is set as a standard format identifier that is linked to mask 1 (MT[2:0] of CnMCONF14 register are set to 010_B).

**Mask setting for CAN module 1 (mask 1) (example)
(Using CAN1 address mask 1 registers L and H (C1MASKL1 and C1MASKH1))**

CMID28	CMID27	CMID26	CMID25	CMID24	CMID23	CMID22	CMID21	CMID20	CMID19	CMID18
1	0	0	0	0	1	0	1	1	1	1
CMID17	CMID16	CMID15	CMID14	CMID13	CMID12	CMID11	CMID10	CMID9	CMID8	CMID7
1	1	1	1	1	1	1	1	1	1	1
CMID6	CMID5	CMID4	CMID3	CMID2	CMID1	CMID0				
1	1	1	1	1	1	1				

1: Not compared (masked)
0: Compared

The CMID27 to CMID24 and CMID22 bits are cleared to 0, and the CMID28, CMID23, and CMID21 to CMID0 bits are set to 1.

18.9.5 Multi buffer receive block function

The multi buffer receive block (MBRB) function is used to store a block of data in two or more message buffers sequentially with no CPU interaction, by setting the same ID to two or more message buffers with the same message buffer type. These message buffers can be allocated anywhere in the message buffer memory, they do not even have to follow each other adjacently.

Suppose, for example, the same message buffer type is set to 10 message buffers, message buffers 10 to 19, and the same ID is set to each message buffer. If the first message whose ID matches an ID of the message buffers is received, it is stored in message buffer 10. At this point, the DN bit of message buffer 10 is set, prohibiting overwriting the message buffer when subsequent messages are received.

When the next message with a matching ID is received, it is received and stored in message buffer 11. Each time a message with a matching ID is received, it is sequentially (in the ascending order) stored in message buffers 12, 13, and so on. Even when a data block consisting of multiple messages is received, the messages can be stored and received without overwriting the previously received matching-ID data.

Whether a data block has been received and stored can be checked by setting the IE bit of the CnMCTRLm register of each message buffer. For example, if a data block consists of k messages, k message buffers are initialized for reception of the data block. The IE bit in message buffers 0 to (k-2) is cleared to 0 (interrupts disabled), and the IE bit in message buffer k-1 is set to 1 (interrupts enabled). In this case, a reception completion interrupt occurs when a message has been received and stored in message buffer k-1, indicating that MBRB has become full. Alternatively, by clearing the IE bit of message buffers 0 to (k-3) and setting the IE bit of message buffer k-2, a warning that MBRB is about to overflow can be issued.

The basic conditions of storing receive data in each message buffer for the MBRB are the same as the conditions of storing data in a single message buffer.

Caution

1. MBRB can be configured for each of the same message buffer types. Therefore, even if a message buffer of another MBRB whose ID matches but whose message buffer type is different has a vacancy, the received message is not stored in that message buffer, but instead discarded.
2. MBRB does not have a ring buffer structure. Therefore, after a message is stored in the message buffer having the highest number in the MBRB configuration, a newly received message will not be stored in the message buffer having the lowest message buffer number.
3. MBRB operates based on the reception and storage conditions; there are no settings dedicated to MBRB, such as function enable bits. By setting the same message buffer type and ID to two or more message buffers, MBRB is automatically configured.
4. With MBRB, “matching ID” means “matching ID after mask”. Even if the ID set to each message buffer is not the same, if the ID that is masked by the mask register matches, it is considered a matching ID and the buffer that has this ID is treated as the storage destination of a message.
5. The priority between MBRBs is mentioned in the table *Table 18-24*.

18.9.6 Remote frame reception

In all the operation modes, when a remote frame is received, the message buffer that is to store the remote frame is searched from all the message buffers satisfying the following conditions.

- Used as a message buffer
  (MA0 bit of CnMCONFm register set to 1.)
- Set as a transmit message buffer
  (MT[2:0] bits in CnMCONFm register set to 000_B)
- Ready for reception
  (RDY bit of CnMCTRLm register set to 1.)
- Set to transmit message
  (RTR bit of CnMCONFm register is cleared to 0.)
- Transmission request is not set.
  (TRQ bit of CnMCTRLm register is cleared to 0.)

Upon acceptance of a remote frame, the following actions are executed if the ID of the received remote frame matches the ID of a message buffer that satisfies the above conditions.

- The DLC[3:0] bit string in the CnMDLCm register store the received DLC value.
- The CnMDATA0m to CnMDATA7m registers in the data area are not updated (data before reception is saved).
- The DN bit of the CnMCTRLm register is set to 1.
- The CINTS1 bit of the CnINTS register is set to 1 (if the IE bit in the CnMCTRLm register of the message buffer that receives and stores the frame is set to 1).
- The receive completion interrupt (INTCnREC) is output (if the IE bit of the message buffer that receives and stores the frame is set to 1 and if the CIE1 bit of the CnIE register is set to 1).
- The message buffer number is recorded in the receive history list.

Caution When a message buffer is searched for receiving and storing a remote frame, overwrite control by the OWS bit of the CnMCONFm register of the message buffer and the DN bit of the CnMCTRLm register are not checked. The setting of OWS is ignored, and DN is set in any case.
If more than one transmit message buffer has the same ID and the ID of the received remote frame matches that ID, the remote frame is stored in the transmit message buffer with the lowest message buffer number.

18.10 Message Transmission

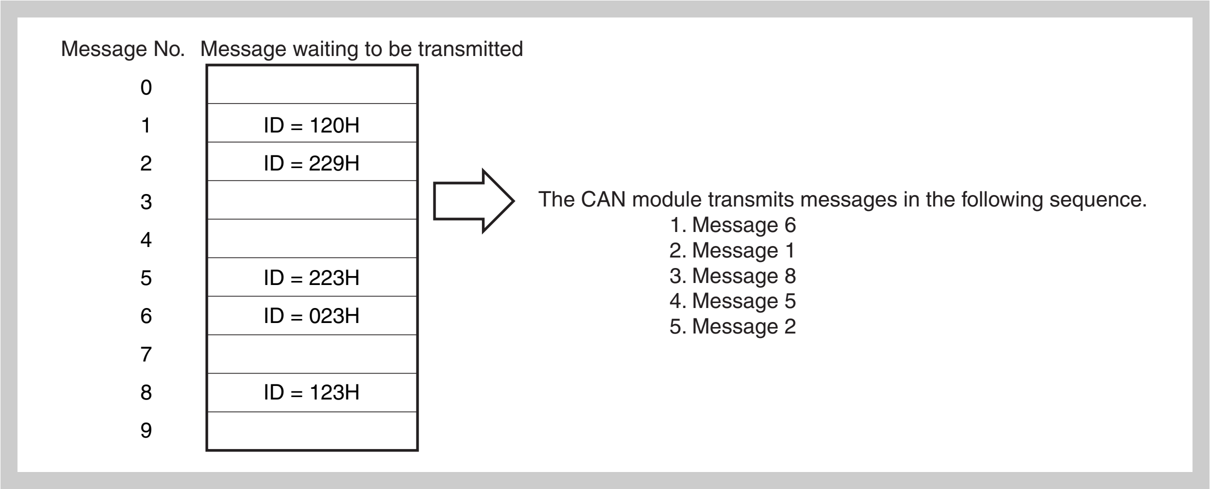
18.10.1 Message transmission

A message buffer with its TRQ bit set to 1 participates in the search for the most high-prioritized message when the following conditions are fulfilled. This behavior is valid for all operational modes.

- Used as a message buffer
(MA0 bit of CnMCONFm register set to 1.)
- Set as a transmit message buffer
(MT[2:0] bits of CnMCONFm register set to 000_B.)
- Ready for transmission
(RDY bit of CnMCTRLm register set to 1.)

The CAN system is a multi-master communication system. In a system like this, the priority of message transmission is determined based on message identifiers (IDs). To facilitate transmission processing by software when there are several messages awaiting transmission, the CAN module uses hardware to check the ID of the message with the highest priority and automatically identifies that message. This eliminates the need for software-based priority control.

Transmission priority is controlled by the identifier (ID).



Message No.	Message waiting to be transmitted
0	
1	ID = 120H
2	ID = 229H
3	
4	
5	ID = 223H
6	ID = 023H
7	
8	ID = 123H
9	

The CAN module transmits messages in the following sequence.
1. Message 6
2. Message 1
3. Message 8
4. Message 5
5. Message 2

Figure 18-30 Message processing example

After the transmit message search, the transmit message with the highest priority of the transmit message buffers that have a pending transmission request (message buffers with the TRQ bit set to 1 in advance) is transmitted.

If a new transmission request is set, the transmit message buffer with the new transmission request is compared with the transmit message buffer with a pending transmission request. If the new transmission request has a higher priority, it is transmitted, unless transmission of a message with a low priority has already started. If transmission of a message with a low priority has already started, however, the new transmission request is transmitted later. To solve this priority inversion effect, the software can perform a transmission abort request for the lower priority message. The highest priority is determined according to the following rules.

Priority	Conditions	Description
1 (high)	Value of first 11 bits of ID [ID28 to ID18]:	The message frame with the lowest value represented by the first 11 bits of the ID is transmitted first. If the value of an 11-bit standard ID is equal to or smaller than the first 11 bits of a 29-bit extended ID, the 11-bit standard ID has a higher priority than a message frame with a 29-bit extended ID.
2	Frame type	A data frame with an 11-bit standard ID (RTR bit is cleared to 0) has a higher priority than a remote frame with a standard ID and a message frame with an extended ID.
3	ID type	A message frame with a standard ID (IDE bit is cleared to 0) has a higher priority than a message frame with an extended ID.
4	Value of lower 18 bits of ID [ID17 to ID0]:	If one or more transmission-pending extended ID message frame has equal values in the first 11 bits of the ID and the same frame type (equal RTR bit values), the message frame with the lowest value in the lower 18 bits of its extended ID is transmitted first.
5 (low)	Message buffer number	If two or more message buffers request transmission of message frames with the same ID, the message from the message buffer with the lowest message buffer number is transmitted first.

Note **1.** If the automatic block transmission request bit ABTTRG is set to 1 in the normal operation mode with ABT, the TRQ bit is set to 1 only for one message buffer in the ABT message buffer group.

If the ABT mode was triggered by ABTTRG bit (1), one TRQ bit is set to 1 in the ABT area (buffer 0 through 7). Beyond this TRQ bit, the application can request transmissions (set TRQ bit to 1) for other TX-message buffers that do not belong to the ABT area. In that case an interval arbitration process (TX-search) evaluates all TX-message buffers with TRQ bit set to 1 and chooses the message buffer that contains the highest prioritized identifier for the next transmission. If there are 2 or more identifiers that have the highest priority (i.e. identical identifiers), the message located at the lowest message buffer number is transmitted at first.

Upon successful transmission of a message frame, the following operations are performed.

- The TRQ flag of the corresponding transmit message buffer is automatically cleared to 0.
- The transmission completion status bit CINTS0 of the CnINTS register is set to 1 (if the interrupt enable bit (IE) of the corresponding transmit message buffer is set to 1).
- An interrupt request signal INTCnTRX is output (if the CIE0 bit of the CnIE register is set to 1 and if the interrupt enable bit (IE) of the corresponding transmit message buffer is set to 1).

2. When changing the contents of a transmit buffer, the RDY flag of this buffer must be cleared before updating the buffer contents. As during internal transfer actions, the RDY flag may be locked temporarily, the status of RDY must be checked by software, after changing it.

18.10.2 Transmit history list function

The transmit history list (THL) function records in the transmit history list the number of the transmit message buffer from which data or remote frames have been were sent. The THL consists of storage elements equivalent to up to seven messages, the last out-message pointer (LOPT) with the corresponding CnLOPT register, and the transmit history list get pointer (TGPT) with the corresponding CnTGPT register.

The THL is undefined immediately after the transition of the CAN module from the initialization mode to one of the operation modes.

The CnLOPT register holds the contents of the THL element indicated by the value of the LOPT pointer minus 1. By reading the CnLOPT register, therefore, the number of the message buffer that transmitted a data frame or remote frame first can be checked. The LOPT pointer is utilized as a write pointer that indicates to what part of the THL a message buffer number is recorded. Any time a data frame or remote frame is transmitted, the corresponding message buffer number is recorded to the THL element indicated by the LOPT pointer. Each time recording to the THL has been completed, the LOPT pointer is automatically incremented. In this way, the number of the message buffer that has received and stored a frame will be recorded chronologically.

The TGPT pointer is utilized as a read pointer that reads a recorded message buffer number from the THL. This pointer indicates the first THL element that the CPU has not yet read. By reading the CnTGPT register by software, the number of a message buffer that has completed transmission can be read. Each time a message buffer number is read from the CnTGPT register, the TGPT pointer is automatically incremented.

If the value of the TGPT pointer matches the value of the LOPT pointer, the THPM bit (transmit history list pointer match) of the CnTGPT register is set to 1. This indicates that no message buffer numbers that have not been read remain in the THL. If a new message buffer number is recorded, the LOPT pointer is incremented and because its value no longer matches the value of the TGPT pointer, the THPM bit is cleared. In other words, the numbers of the unread message buffers exist in the THL.

If the LOPT pointer is incremented and matches the value of the TGPT pointer minus 1, the TOVF bit (transmit history list overflow) of the CnTGPT register is set to 1. This indicates that the THL is full of message buffer numbers that have not been read. If a new message is received and stored, the message buffer number recorded last is overwritten by the message buffer number that transmitted its message afterwards. In this case, after the TOVF bit has been set (1), therefore, the recorded message buffer numbers in the THL do not completely reflect the chronological order. However the other transmitted messages can be found by a CPU search applied to all transmit message buffers unless the CPU has not overwritten a transmit object in one of these buffers beforehand. In total up to six transmission completions can occur without overflowing the THL.

Caution If the history list is in the overflow condition (TOVF is set), reading the history list contents is still possible, until the history list is empty (indicated by THPM flag set). Nevertheless, the history list remains in the overflow condition, until TOVF is cleared by software. If TOVF is not cleared, the THPM flag will also not be updated (cleared) upon successful transmission of a new message. This may lead to the situation, that THPM indicates an empty history list, although a successful transmission has taken place, while the history list is in the overflow state (TOVF and THPM are set).

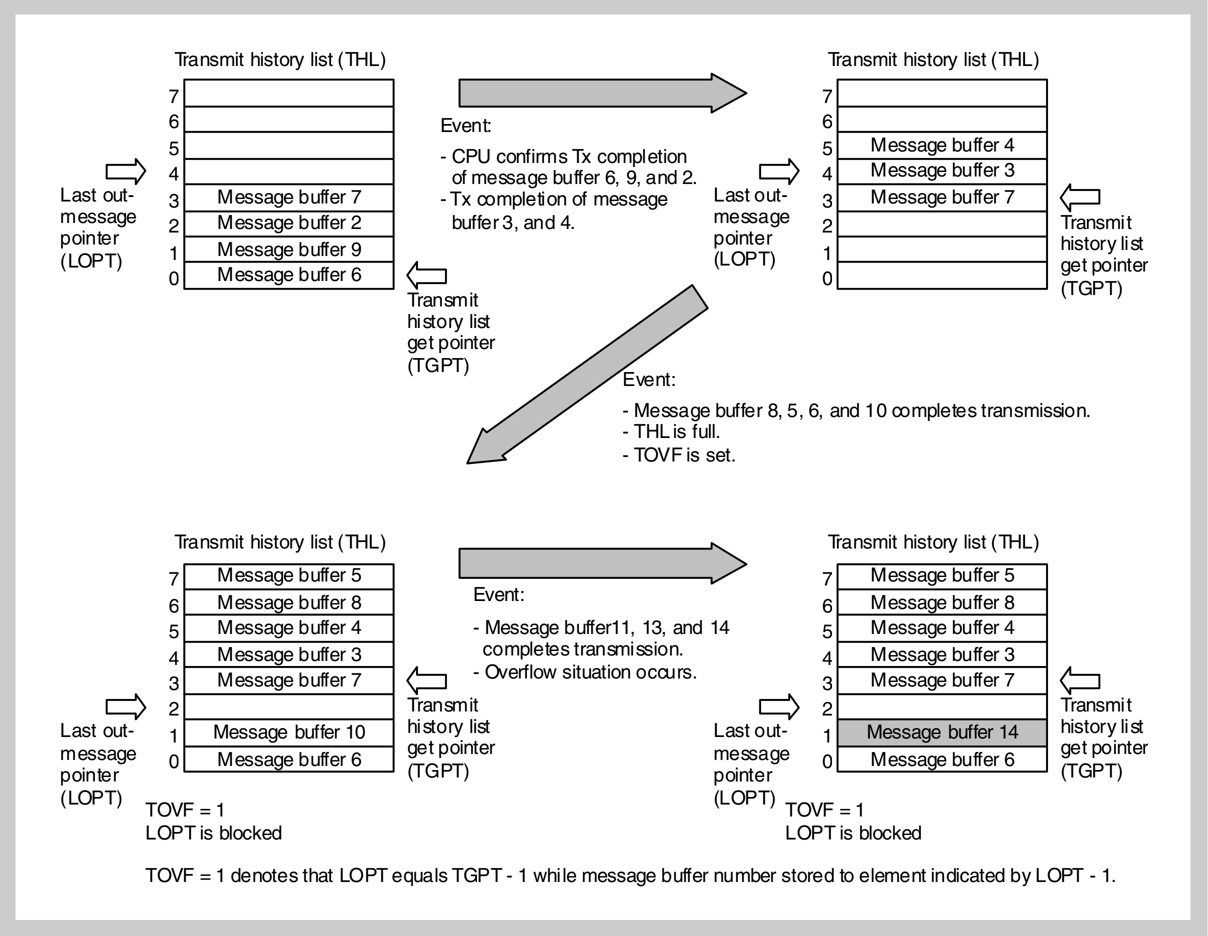

Transmit history list (THL)
7
6
5
4
3 Message buffer 7
2 Message buffer 2
1 Message buffer 9
0 Message buffer 6
Last out-message pointer (LOPT)
Transmit history list get pointer (TGPT)
Event:
- CPU confirms Tx completion of message buffer 6, 9, and 2.
- Tx completion of message buffer 3, and 4.
Transmit history list (THL)
7
6
5 Message buffer 4
4 Message buffer 3
3 Message buffer 7
2
1
0
Last out-message pointer (LOPT)
Transmit history list get pointer (TGPT)
Event:
- Message buffer 8, 5, 6, and 10 completes transmission.
- THL is full.
- TOVF is set.
Transmit history list (THL)
7 Message buffer 5
6 Message buffer 8
5 Message buffer 4
4 Message buffer 3
3 Message buffer 7
2
1 Message buffer 10
0 Message buffer 6
Last out-message pointer (LOPT)
Transmit history list get pointer (TGPT)
TOVF = 1
LOPT is blocked
Event:
- Message buffer11, 13, and 14 completes transmission.
- Overflow situation occurs.
Transmit history list (THL)
7 Message buffer 5
6 Message buffer 8
5 Message buffer 4
4 Message buffer 3
3 Message buffer 7
2
1 Message buffer 14
0 Message buffer 6
Last out-message pointer (LOPT)
Transmit history list get pointer (TGPT)
TOVF = 1
LOPT is blocked
TOVF = 1 denotes that LOPT equals TGPT - 1 while message buffer number stored to element indicated by LOPT - 1.

Figure 18-31 Transmit history list

18.10.3 Automatic block transmission (ABT)

The automatic block transmission (ABT) function is used to transmit two or more data frames successively with no CPU interaction. The maximum number of transmit message buffers assigned to the ABT function is eight (message buffer numbers 0 to 7).

By setting the OPMODE[2:0] bits of the CnCTRL register to 010_B, “normal operation mode with automatic block transmission function” (hereafter referred to as ABT mode) can be selected.

To issue an ABT transmission request, define the message buffers by software first. Set the MA0 bit (1) in all the message buffers used for ABT, and define all the buffers as transmit message buffers by setting the MA[2:0] bits to 000_B. Be sure to set the same ID for the message buffers for ABT even when that ID is being used for all the message buffers. To use two or more IDs, set the ID of each message buffer by using the CnMIDLm and CnMIDHm registers. Set the CnMDLCm and CnMDATA0m to CnMDATA7m registers before issuing a transmission request for the ABT function.

After initialization of message buffers for ABT is finished, the RDY bit needs to be set (1). In the ABT mode, the TRQ bit does not have to be manipulated by software.

After the data for the ABT message buffers has been prepared, set the ABTTRG bit to 1. Automatic block transmission is then started. When ABT is started, the TRQ bit in the first message buffer (message buffer 0) is automatically set to 1. After transmission of the data of message buffer 0 is finished, the TRQ bit of the next message buffer, message buffer 1, is set automatically. In this way, transmission is executed successively.

A delay time can be inserted by program in the interval in which the transmission request (TRQ) is automatically set while successive transmission is being executed. The delay time to be inserted is defined by the CnGMABTD register. The unit of the delay time is DBT (data bit time). DBT depends on the setting of the CnBRP and CnBTR registers.

Among transmit objects within the ABT-area, the priority of the transmission ID is not evaluated. The data of message buffers 0 to 7 are sequentially transmitted. When transmission of the data frame from message buffer 7 has been completed, the ABTTRG bit is automatically cleared to 0 and the ABT operation is finished.

If the RDY bit of an ABT message buffer is cleared during ABT, no data frame is transmitted from that buffer, ABT is stopped, and the ABTTRG bit is cleared. After that, transmission can be resumed from the message buffer where ABT stopped, by setting the RDY and ABTTRG bits to 1 by software. To not resume transmission from the message buffer where ABT stopped, the internal ABT engine can be reset by setting the ABTCLR bit to 1 while ABT mode is stopped and the ABTTRG bit is cleared to 0. In this case, transmission is started from message buffer 0 if the ABTCLR bit is cleared to 0 and then the ABTTRG bit is set to 1.

An interrupt can be used to check if data frames have been transmitted from all the message buffers for ABT. To do so, the IE bit of the CnMCTRLm register of each message buffer except the last message buffer needs to be cleared (0).

If a transmit message buffer other than those used by the ABT function (message buffers 8 to 31) is assigned to a transmit message buffer, the message to be transmitted next is determined by the priority of the transmission ID of the ABT message buffer whose transmission is currently held pending and the transmission ID of the message buffers other than those used by the ABT function.

Transmission of a data frame from an ABT message buffer is not recorded in the transmit history list (THL).

Caution

1. Set the ABTCLR bit to 1 while the ABTTRG bit is cleared to 0 in order to resume ABT operation at buffer No.0. If the ABTCLR bit is set to 1 while the ABTTRG bit is set to 1, the subsequent operation is not guaranteed.
2. If the automatic block transmission engine is cleared by setting the ABTCLR bit to 1, the ABTCLR bit is automatically cleared immediately after the processing of the clearing request is completed.
3. Do not set the ABTTRG bit in the initialization mode. If the ABTTRG bit is set in the initialization mode, the proper operation is not guaranteed after the mode is changed from the initialization mode to the ABT mode.
4. Do not set the TRQ bit of the ABT message buffers to 1 by software in the normal operation mode with ABT. Otherwise, the operation is not guaranteed.
5. The CnGMABTD register is used to set the delay time that is inserted in the period from completion of the preceding ABT message to setting of the TRQ bit for the next ABT message when the transmission requests are set in the order of message numbers for each message for ABT that is successively transmitted in the ABT mode. The timing at which the messages are actually transmitted onto the CAN bus varies depending on the status of transmission from other stations and the status of the setting of the transmission request for messages other than the ABT messages (message buffers 8 to 31).
6. If a transmission request is made for a message other than an ABT message and if no delay time is inserted in the interval in which transmission requests for ABT are automatically set (CnGMABTD register = 00_H), messages other than ABT messages may be transmitted not depending on their priority compared to the priority of the ABT message.
7. Do not clear the RDY bit to 0 when the ABTTRG bit = 1.
8. If a message is received from another node while normal operation mode with ABT is active, the TX-message from the ABT-area may be transmitted with delay of one frame although CnGMABTD register was set up with 00_H.

18.10.4 Transmission abort process

(1) Transmission abort process except for in normal operation mode with automatic block transmission (ABT)

The user can clear the TRQ bit of the CnMCTRLm register to 0 to abort a transmission request. The TRQ bit will be cleared immediately if the abort was successful. Whether the transmission was successfully aborted or not can be checked using the TSTAT bit of the CnCTRL register and the CnTGPT register, which indicate the transmission status on the CAN bus (for details, refer to the processing in *Figure 18-45 on page 668*).

(2) Transmission abort process except for ABT transmission in normal operation mode with automatic block transmission (ABT)

The user can clear the ABTTRG bit of the CnGMABT register to 0 to abort a transmission request. After checking the ABTTRG bit of the CnGMABT register = 0, clear the TRQ bit of the CnMCTRLm register to 0. The TRQ bit will be cleared immediately if the abort was successful. Whether the transmission was successfully aborted or not can be checked using the TSTAT bit of the CnCTRL register and the CnTGPT register, which indicate the transmission status on the CAN bus (for details, refer to the processing in *Figure 18-46 on page 669*).

(3) Transmission abort process for ABT transmission in normal operation mode with automatic block transmission (ABT)

To abort ABT that is already started, clear the ABTTRG bit of the CnGMABT register to 0. In this case, the ABTTRG bit remains 1 if an ABT message is currently being transmitted and until the transmission is completed (successfully or not), and is cleared to 0 as soon as transmission is finished. This aborts ABT.

If the last transmission (before ABT) was successful, the normal operation mode with ABT is left with the internal ABT pointer pointing to the next message buffer to be transmitted.

In the case of an erroneous transmission, the position of the internal ABT pointer depends on the status of the TRQ bit in the last transmitted message buffer. If the TRQ bit is set to 1 when clearing the ABTTRG bit is requested, the internal ABT pointer points to the last transmitted message buffer (for details, refer to the process in *Figure 18-47 on page 670*). If the TRQ bit is cleared to 0 when clearing the ABTTRG bit is requested, the internal ABT pointer is incremented (+1) and points to the next message buffer in the ABT area (for details, refer to the process in *Figure 18-48 on page 671*).

Caution Be sure to abort ABT by clearing ABTTRG bit to 0. The operation is not guaranteed if aborting transmission is requested by clearing RDY.

When the normal operation mode with ABT is resumed after ABT has been aborted and the ABTTRG bit is set to 1, the next ABT message buffer to be transmitted can be determined from the following table.

Status of TRQ of ABT message buffer	Abort after successful transmission	Abort after erroneous transmission
Set (1)	Next message buffer in the ABT area[a]	Same message buffer in the ABT area
Cleared (0)	Next message buffer in the ABT area[a]	Next message buffer in the ABT area[a]

a) The above resumption operation can be performed only if a message buffer ready for ABT exists in the ABT area. For example, an abort request that is issued while ABT of message buffer 7 is in progress is regarded as completion of ABT, rather than abort, if transmission of message buffer 7 has been successfully completed, even if the ABTTRG bit is cleared to 0. If the RDY bit in the next message buffer in the ABT area is cleared to 0, the internal ABT pointer is retained, but the resumption operation is not performed even if the ABTTRG bit is set to 1, and ABT ends immediately.

18.10.5 Remote frame transmission

Remote frames can be transmitted only from transmit message buffers. Set whether a data frame or remote frame is transmitted via the RTR bit of the CnMCONFm register. Setting (1) the RTR bit sets remote frame transmission.

18.11 Power Saving Modes

18.11.1 CAN sleep mode

The CAN sleep mode can be used to set the CAN Controller to stand-by mode in order to reduce power consumption. The CAN module can enter the CAN sleep mode from all operation modes. Release of the CAN sleep mode returns the CAN module to exactly the same operation mode from which the CAN sleep mode was entered.

In the CAN sleep mode, the CAN module does not transmit messages, even when transmission requests are issued or pending.

(1) Entering CAN sleep mode

The CPU issues a CAN sleep mode transition request by writing 01_B to the PSMODE[1:0] bits of the CnCTRL register.

This transition request is only acknowledged only under the following conditions.

3. The CAN module is already in one of the following operation modes
   - Normal operation mode
   - Normal operation mode with ABT
   - Receive-only mode
   - Single-shot mode
   - Self-test mode
   - CAN stop mode in all the above operation modes
4. The CAN bus state is bus idle (the 4th bit in the interframe space is recessive).
   If the CAN bus is fixed to dominant, the request for transition to the CAN sleep mode is held pending.Also the transition from CAN stop mode to CAN sleep mode is independent of the CAN bus state.
5. No transmission request is pending

Note If a sleep mode request is pending, and at the same time a message is received in a message box, the sleep mode request is not cancelled, but is executed right after message storage has been finished. This may result in AFCAN being in sleep mode, while the CPU would execute the RX interrupt routine. Therefore, the interrupt routine must check the access to the message buffers as well as reception history list registers by using the MBON flag, if sleep mode is used.

If any one of the conditions mentioned above is not met, the CAN module will operate as follows.

- If the CAN sleep mode is requested from the initialization mode, the CAN sleep mode transition request is ignored and the CAN module remains in the initialization mode.
- If the CAN bus state is not bus idle (i.e., the CAN bus state is either transmitting or receiving) when the CAN sleep mode is requested in one of the operation modes, immediate transition to the CAN sleep mode is not possible. In this case, the CAN sleep mode transition request has to be held pending until the CAN bus state becomes bus idle (the 4th bit in the interframe space is recessive). In the time from the CAN sleep mode request to successful transition, the PSMODE[1:0] bits remain 00_B. When

the module has entered the CAN sleep mode, the PSMODE[1:0] bits are set to 01_B.

- If a request for transition to the initialization mode and a request for transition to the CAN sleep mode are made at the same time while the CAN module is in one of the operation modes, the request for the initialization mode is enabled. The CAN module enters the initialization mode at a predetermined timing. At this time, the CAN sleep mode request is not held pending and is ignored.
- Even when initialization mode and sleep mode are not requested simultaneously (i.e the first request has not been granted while the second request is made), the request for initialization has priority over the sleep mode request. The sleep mode request is cancelled when the initialization mode is requested. When a pending request for initialization mode is present, a subsequent request for Sleep mode request is cancelled right at the point in time where it was submitted.

(2) Status in CAN sleep mode

The CAN module is in the following state after it enters the CAN sleep mode:

- The internal operating clock is stopped and the power consumption is minimized.
- The function to detect the falling edge of the CAN reception pin (CRXDn) remains in effect to wake up the CAN module from the CAN bus.
- To wake up the CAN module from the CPU, data can be written to the PSMODE[1:0] bits of the CAN module control register (CnCTRL), but nothing can be written to other CAN module registers or bits.
- The CAN module registers can be read, except for the CnLIPT, CnRGPT, CnLOPT, and CnTGPT registers.
- The CAN message buffer registers cannot be written or read.
- MBON bit of the CAN Global Control register (CnGMCTRL) is cleared.
- A request for transition to the initialization mode is not acknowledged and is ignored.

(3) Releasing CAN sleep mode

The CAN sleep mode is released by the following events:

- When the CPU writes 00_B to the PSMODE[1:0] bits of the CnCTRL register
- A falling edge at the CAN reception pin (CRXDn) (i.e. the CAN bus level shifts from recessive to dominant)

Caution Even if the falling edge belongs to the SOF of a receive message, this message will not be received and stored. If the CPU has turned off the clock supply to the CAN module while the CAN module was in sleep mode, even subsequently the CAN sleep mode will not be released and PSMODE [1:0] will remain 01_B unless the clock to the CAN module is supplied again. In addition to this, the receive message will not be received after that.

After releasing the sleep mode, the CAN module returns to the operation mode from which the CAN sleep mode was requested and the PSMODE[1:0] bits of the CnCTRL register must be reset by software to 00_B. If the CAN sleep mode is released by a change in the CAN bus state, the CINTS5 bit of the CnINTS register is set to 1, regardless of the CIE bit of the CnIE register. After the CAN module is released from the CAN sleep mode, it participates in the CAN bus again by automatically detecting 11 consecutive recessive-level bits on the CAN bus. The user application has to wait until MBON = 1, before accessing message buffers again.

When a request for transition to the initialization mode is made while the CAN module is in the CAN sleep mode, that request is ignored; the CAN module has to be released from sleep mode by software first before entering the initialization mode.

Caution

1. Be aware that the release of CAN sleep mode by CAN bus event, and thus the wake up interrupt may happen at any time, even right after requesting sleep mode, if a CAN bus event occurs.
2. Always reset the PSMODE[1:0] bits to 00_B, when waking up from CAN sleep mode, before accessing any other registers of the CAN module.
3. Always clear the interrupt flag CINTS5, when waking up from CAN sleep mode.

18.11.2 CAN stop mode

The CAN stop mode can be used to set the CAN Controller to stand-by mode to reduce power consumption. The CAN module can enter the CAN stop mode only from the CAN sleep mode. Release of the CAN stop mode puts the CAN module in the CAN sleep mode.

The CAN stop mode can only be released (entering CAN sleep mode) by writing 01_B to the PSMODE[1:0] bits of the CnCTRL register and not by a change in the CAN bus state. No message is transmitted even when transmission requests are issued or pending.

(1) Entering CAN stop mode

A CAN stop mode transition request is issued by writing 11_B to the PSMODE[1:0] bits of the CnCTRL register.

A CAN stop mode request is only acknowledged when the CAN module is in the CAN sleep mode. In all other modes, the request is ignored.

Caution To set the CAN module to the CAN stop mode, the module must be in the CAN sleep mode. To confirm that the module is in the sleep mode, check that the PSMODE[1:0] bits = 01_B, and then request the CAN stop mode. If a bus change occurs at the CAN reception pin (CRXDn) while this process is being performed, the CAN sleep mode is automatically released. In this case, the CAN stop mode transition request cannot be acknowledged.

(2) Status in CAN stop mode

The CAN module is in the following state after it enters the CAN stop mode.

- The internal operating clock is stopped and the power consumption is minimized.
- To wake up the CAN module from the CPU, data can be written to the PSMODE[1:0] bits of the CAN module control register (CnCTRL), but nothing can be written to other CAN module registers or bits.
- The CAN module registers can be read, except for the CnLIPT, CnRGPT, CnLOPT, and CnTGPT registers.
- The CAN message buffer registers cannot be written or read.
- MBON bit of the CAN Global Control register (CnGMCTRL) is cleared.
- An initialization mode transition request is not acknowledged and is ignored.

(3) Releasing CAN stop mode

The CAN stop mode can only be released by writing 01_B to the PSMODE[1:0] bits of the CnCTRL register. After releasing the CAN stop mode, the CAN module enters the CAN sleep mode.

When the initialization mode is requested while the CAN module is in the CAN stop mode, that request is ignored; the CPU has to release the stop mode and subsequently CAN sleep mode before entering the initialization mode. It is impossible to enter the other operation mode directly from the CAN stop mode not entering the CAN sleep mode, that request is ignored.

18.11.3 Example of using power saving modes

In some application systems, it may be necessary to place the CPU in a power saving mode to reduce the power consumption. By using the power saving mode specific to the CAN module and the power saving mode specific to the CPU in combination, the CPU can be woken up from the power saving status by the CAN bus.

Here is an example for using the power saving modes.

- First, put the CAN module in the CAN sleep mode (PSMODE[1:0] = 01_B). Next, put the CPU in the power saving mode. If an edge transition from recessive to dominant is detected at the CAN reception pin (CRXDn) in this status, the CINTS5 bit in the CAN module is set to 1. If the CIE5 bit of the CnCTRL register is set to 1, a wakeup interrupt (INTWUPn) is generated.
- The CAN module is automatically released from CAN sleep mode (PSMODE = 00_B) and returns to normal operation mode.
- The CPU, in response to INTWUPn, can release its own power saving mode and return to normal operation mode.

  To further reduce the power consumption of the CPU, the internal clock - including that of the CAN module - may be stopped. In this case, the operating clock supplied to the CAN module is stopped after the CAN module has been put in CAN sleep mode. Then the CPU enters a power saving mode in which the clock supplied to the CPU is stopped.
- If an edge transition from recessive to dominant is detected at the CAN reception pin (CRXDn) in this status, the CAN module can set the CINTS5 bit to 1 and generate the wakeup interrupt (INTWUPn) even if it is not supplied with the clock.
- The other functions, however, do not operate, because clock supply to the CAN module is stopped, and the module remains in CAN sleep mode.
- The CPU, in response to INTWUPn
  - releases its power saving mode,
  - resumes supply of the internal clocks - including the clock to the CAN module - after the oscillation stabilization time has elapsed, and
  - starts instruction execution.
- The CAN module is immediately released from the CAN sleep mode when clock supply is resumed, and returns to the normal operation mode (PSMODE = 00_B).

18.12 Interrupt Function

The CAN module provides 6 different interrupt sources.

The occurrence of these interrupt sources is stored in interrupt status registers. Four separate interrupt request signals are generated from the six interrupt sources. When an interrupt request signal that corresponds to two or more interrupt sources is generated, the interrupt sources can be identified by using an interrupt status register. After an interrupt source has occurred, the corresponding interrupt status bit must be cleared to 0 by software.

Table 18-25 List of CAN module interrupt sources

No.	Interrupt status bit		Interrupt enable bit		Interrupt request signal	Interrupt source description
	Name	Register	Name	Register		
1	CINTS0	CnINTS	CIE0[a]	CnIE	INTCnTRX	Message frame successfully transmitted from message buffer m
2	CINTS1	CnINTS	CIE1[a]	CnIE	INTCnREC	Valid message frame reception in message buffer m
3	CINTS2	CnINTS	CIE2	CnIE	INTCnERR	CAN module error state interrupt (Supplement 1)
4	CINTS3	CnINTS	CIE3	CnIE		CAN module protocol error interrupt (Supplement 2)
5	CINTS4	CnINTS	CIE4	CnIE		CAN module arbitration loss interrupt
6	CINTS5	CnINTS	CIE5	CnIE	INTCnWUP	CAN module wakeup interrupt from CAN sleep mode (Supplement 3)

[a] The IE bit (message buffer interrupt enable bit) in the CnMCTRL register of the corresponding message buffer has to be set to 1 for that message buffer to participate in the interrupt generation process.

Supplements

1. This interrupt is generated when the transmission/reception error counter is at the warning level, or in the error passive or bus-off state.
2. This interrupt is generated when a stuff error, form error, ACK error, bit error, or CRC error occurs.
3. This interrupt is generated when the CAN module is woken up from the CAN sleep mode because a falling edge is detected at the CAN reception pin (CAN bus transition from recessive to dominant).

18.13 Diagnosis Functions and Special Operational Modes

The CAN module provides a receive-only mode, single-shot mode, and self-test mode to support CAN bus diagnosis functions or the operation of special CAN communication methods.

18.13.1 Receive-only mode

The receive-only mode is used to monitor receive messages without causing any interference on the CAN bus and can be used for CAN bus analysis nodes.

For example, this mode can be used for automatic baud-rate detection. The baud rate in the CAN module is changed until “valid reception” is detected, so that the baud rates in the module match (“valid reception” means a message frame has been received in the CAN protocol layer without occurrence of an error and with an appropriate ACK between nodes connected to the CAN bus). A valid reception does not require message frames to be stored in a receive message buffer (data frames) or transmit message buffer (remote frames). The event of valid reception is indicated by setting the VALID bit of the CnCTRL register (1).

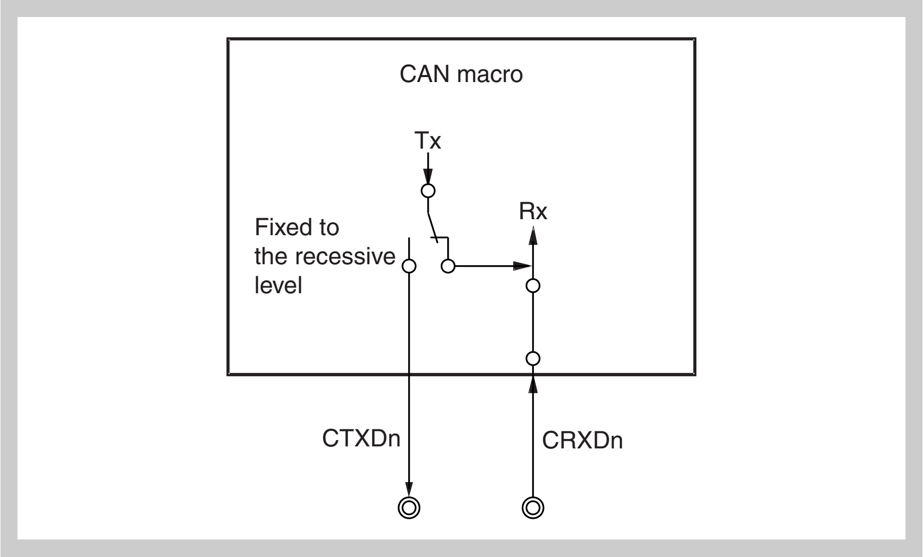


Figure 18-32 CAN module terminal connection in receive-only mode

In the receive-only mode, no message frames can be transmitted from the CAN module to the CAN bus. Transmit requests issued for message buffers defined as transmit message buffers are held pending.

In the receive-only mode, the CAN transmission pin (CTXDn) in the CAN module is fixed to the recessive level. Therefore, no active error flag can be transmitted from the CAN module to the CAN bus even when a CAN bus error is detected while receiving a message frame. Since no transmission can be issued from the CAN module, the transmission error counter the CnERC.TEC7 to CnERC.TEC0 bits are never updated. Therefore, a CAN module in the receive-only mode does not enter the bus-off state.

Furthermore, in the receive-only mode ACK is not returned to the CAN bus in this mode upon the valid reception of a message frame. Internally, the local node recognizes that it has transmitted ACK. An overload frame cannot be transmitted to the CAN bus.

Caution If only two CAN nodes are connected to the CAN bus and one of them is operating in the receive-only mode, there is no ACK on the CAN bus. Due to the missing ACK, the transmitting node will transmit an active error flag, and repeat transmitting a message frame. The transmitting node becomes error passive after transmitting the message frame 16 times (assuming that the error counter was 0 in the beginning and no other errors have occurred). After the message frame for the 17th time is transmitted, the transmitting node generates a passive error flag. The receiving node in the receive-only mode detects the first valid message frame at this point, and the VALID bit is set to 1 for the first time.

18.13.2 Single-shot mode

In the single-shot mode, automatic re-transmission as defined in the CAN protocol is switched off. (According to the CAN protocol, a message frame transmission that has been aborted by either arbitration loss or error occurrence has to be repeated without control by software.) All other behavior of single shot mode is identical to normal operation mode. Features of single shot mode can not be used in combination with normal mode with ABT.

The single-shot mode disables the re-transmission of an aborted message frame transmission according to the setting of the AL bit of the CnCTRL register. When the AL bit is cleared to 0, re-transmission upon arbitration loss and upon error occurrence is disabled. If the AL bit is set to 1, re-transmission upon error occurrence is disabled, but re-transmission upon arbitration loss is enabled. As a consequence, the TRQ bit in a message buffer defined as a transmit message buffer is cleared to 0 by the following events:

- Successful transmission of the message frame
- Arbitration loss while sending the message frame
- Error occurrence while sending the message frame

The events arbitration loss and error occurrence can be distinguished by checking the CINTS4 and CINTS3 bits of the CnINTS register respectively, and the type of the error can be identified by reading the LEC[2:0] bits of the CnLEC register.

Upon successful transmission of the message frame, the transmit completion interrupt bit CINTS0 of the CnINTS register is set to 1. If the CIE0 bit of the CnIE register is set to 1 at this time, an interrupt request signal is output.

The single-shot mode can be used when emulating time-triggered communication methods (e.g., TTCAN level 1).

Caution The AL bit is only valid in single-shot mode. It does not influence the operation of re-transmission upon arbitration loss in the other operation modes.

18.13.3 Self-test mode

In the self-test mode, message frame transmission and message frame reception can be tested without connecting the CAN node to the CAN bus or without affecting the CAN bus.

In the self-test mode, the CAN module is completely disconnected from the CAN bus, but transmission and reception are internally looped back. The CAN transmission pin (CTXDn) is fixed to the recessive level.

If the falling edge on the CAN reception pin (CRXDn) is detected after the CAN module has entered the CAN sleep mode from the self-test mode, however, the module is released from the CAN sleep mode in the same manner as the other operation modes. To keep the module in the CAN sleep mode, use the CAN reception pin (CRXDn) as a port pin.

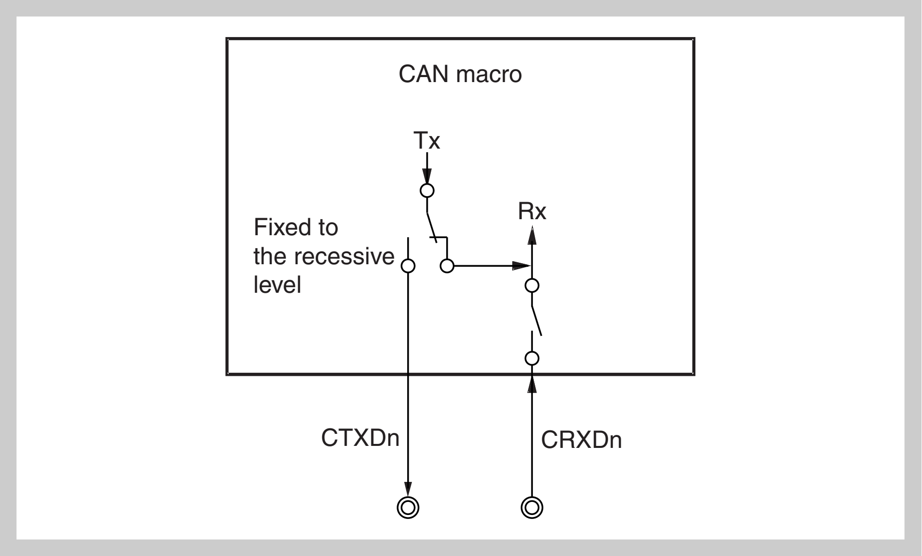


Figure 18-33 CAN module terminal connection in self-test mode

18.13.4 Receive/transmit operation in each operation mode

The following table shows outline of the receive/transmit operation in each operation mode.

Table 18-26 Outline of the receive/transmit in each operation mode

Operation mode	Transmission of data/ remote frame	Transmission of ACK	Transmission of error/ overload frame	Transmission retry	Automatic block transmission (ABT)	Set of VALID bit	Store data to message buffer
Initialization mode	No	No	No	No	No	No	No
Normal operation mode	Yes	Yes	Yes	Yes	No	Yes	Yes
Normal operation mode with ABT	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Receive only mode	No	No	No	No	No	Yes	Yes
Single-shot mode	Yes	Yes	Yes	No[a]	No	Yes	Yes
Self-test mode	Yes[b]	Yes[b]	Yes[b]	Yes[b]	No	Yes[b]	Yes[b]

[a] When the arbitration lost occurs, control of re-transmission is possible by the AL bit of CnCTRL register.

[b] Each signals are not generated to outside, but generated into the CAN module.

18.14 Time Stamp Function

CAN is an asynchronous, serial protocol. All nodes connected to the CAN bus have a local, autonomous clock. As a consequence, the clocks of the nodes have no relation (i.e., the clocks are asynchronous and may have different frequencies).

In some applications, however, a common time base over the network (= global time base) is needed. In order to build up a global time base, a time stamp function is used. The essential mechanism of a time stamp function is the capture of timer values triggered by signals on the CAN bus.

18.14.1 Time stamp function

The CAN Controller supports the capturing of timer values triggered by a specific frame. An on-chip 16-bit capture timer unit in a microcontroller system is used in addition to the CAN Controller. The 16-bit capture timer unit captures the timer value according to a trigger signal (TSOUT) for capturing that is output when a data frame is received from the CAN Controller. The CPU can retrieve the time of occurrence of the capture event, i.e., the time stamp of the message received from the CAN bus, by reading the captured value. The TSOUT signal can be selected from the following two event sources and is specified by the TSSEL bit of the CnTS register.

- SOF event (start of frame) (TSSEL = 0)
- EOF event (last bit of end of frame) (TSSEL = 1)

The TSOUT signal is enabled by setting the TSEN bit of the CnTS register to 1.

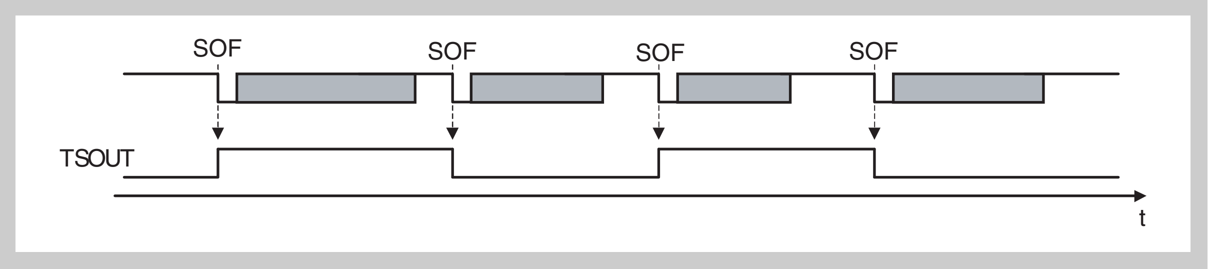


Figure 18-34 Timing diagram of capture signal TSOUT

The TSOUT signal toggles its level upon occurrence of the selected event during data frame reception (in *Figure 18-34*, the SOF is used as the trigger event source). To capture a timer value by using the TSOUT signal, the capture timer unit must detect the capture signal at both the rising edge and falling edge.

This time stamp function is controlled by the TSLOCK bit of the CnTS register. When TSLOCK is cleared to 0, the TSOUT signal toggles upon occurrence of the selected event. If TSLOCK is set to 1, the TSOUT signal toggles upon occurrence of the selected event, but the toggle is stopped as the TSEN bit is automatically cleared to 0 as soon as the message storing to the message buffer 0 starts. This suppresses the subsequent toggle occurrence by the TSOUT signal, so that the time stamp value toggled last (= captured last) can be saved as the time stamp value of the time at which the data frame was received in message buffer 0.

Caution The time stamp function using the TSLOCK bit stops toggle of the TSOUT signal by receiving a data frame in message buffer 0. Therefore, message buffer 0 must be set as a receive message buffer. Since a receive message buffer cannot receive a remote frame, toggle of the TSOUT signal cannot be stopped by reception of a remote frame. Toggle of the TSOUT signal does not stop when a data frame is received in a message buffer other than message buffer 0.
For these reasons, a data frame cannot be received in message buffer 0 when the CAN module is in the normal operation mode with ABT, because message buffer 0 must be set as a transmit message buffer. In this operation mode, therefore, the function to stop toggle of the TSOUT signal by the TSLOCK bit cannot be used.

18.15 Baud Rate Settings

18.15.1 Baud rate setting conditions

Make sure that the settings are within the range of limit values for ensuring correct operation of the CAN Controller, as follows.

- 5TQ ≤ SPT (sampling point) ≤ 17 TQ

  SPT = TSEG1 + 1
- 8 TQ ≤ DBT (data bit time) ≤ 25 TQ

  DBT = TSEG1 + TSEG2 + 1TQ = TSEG2 + SPT
- 1 TQ ≤ SJW (synchronization jump width) ≤ 4TQ

  SJW ≤ DBT – SPT
- 4 ≤ TSEG1 ≤ 16 [3 ≤ Setting value of TSEG1[3:0] ≤ 15]
- 1 ≤ TSEG2 ≤ 8 [0 ≤ Setting value of TSEG2[2:0] ≤ 7]

Note
1. TQ = 1/f_{TQ} (f_{TQ}: CAN protocol layer basic system clock)
2. TSEG1[3:0] (Bits 3 to 0 of CAN bit rate register (CnBTR))
3. TSEG2[2:0] (Bits 10 to 8 of CAN bit rate register (CnBTR))

Table 18-27 shows the combinations of bit rates that satisfy the above conditions.

Table 18-27 Settable bit rate combinations (1/3)

Valid bit rate setting					CnBTR register setting value		Sampling point (unit %)
DBT length	SYNC SEGMENT	PROP SEGMENT	PHASE SEGMENT1	PHASE SEGMENT2	TSEG1 [3:0]	TSEG2 [2:0]	
25	1	8	8	8	1111	111	68.0
24	1	7	8	8	1110	111	66.7
24	1	9	7	7	1111	110	70.8
23	1	6	8	8	1101	111	65.2
23	1	8	7	7	1110	110	69.6
23	1	10	6	6	1111	101	73.9
22	1	5	8	8	1100	111	63.6
22	1	7	7	7	1101	110	68.2
22	1	9	6	6	1110	101	72.7
22	1	11	5	5	1111	100	77.3
21	1	4	8	8	1011	111	61.9
21	1	6	7	7	1100	110	66.7
21	1	8	6	6	1101	101	71.4
21	1	10	5	5	1110	100	76.2
21	1	12	4	4	1111	011	81.0
20	1	3	8	8	1010	111	60.0
20	1	5	7	7	1011	110	65.0
20	1	7	6	6	1100	101	70.0
20	1	9	5	5	1101	100	75.0
20	1	11	4	4	1110	011	80.0
20	1	13	3	3	1111	010	85.0
19	1	2	8	8	1001	111	57.9
19	1	4	7	7	1010	110	63.2
19	1	6	6	6	1011	101	68.4
19	1	8	5	5	1100	100	73.7
19	1	10	4	4	1101	011	78.9
19	1	12	3	3	1110	010	84.2
19	1	14	2	2	1111	001	89.5
18	1	1	8	8	1000	111	55.6
18	1	3	7	7	1001	110	61.1
18	1	5	6	6	1010	101	66.7
18	1	7	5	5	1011	100	72.2
18	1	9	4	4	1100	011	77.8
18	1	11	3	3	1101	010	83.3
18	1	13	2	2	1110	001	88.9
18	1	15	1	1	1111	000	94.4
17	1	2	7	7	1000	110	58.8

Table 18-27 Settable bit rate combinations (2/3)

Valid bit rate setting					CnBTR register setting value		Sampling point (unit %)
DBT length	SYNC SEGMENT	PROP SEGMENT	PHASE SEGMENT1	PHASE SEGMENT2	TSEG1 [3:0]	TSEG2 [2:0]	
17	1	4	6	6	1001	101	64.7
17	1	6	5	5	1010	100	70.6
17	1	8	4	4	1011	011	76.5
17	1	10	3	3	1100	010	82.4
17	1	12	2	2	1101	001	88.2
17	1	14	1	1	1110	000	94.1
16	1	1	7	7	0111	110	56.3
16	1	3	6	6	1000	101	62.5
16	1	5	5	5	1001	100	68.8
16	1	7	4	4	1010	011	75.0
16	1	9	3	3	1011	010	81.3
16	1	11	2	2	1100	001	87.5
16	1	13	1	1	1101	000	93.8
15	1	2	6	6	0111	101	60.0
15	1	4	5	5	1000	100	66.7
15	1	6	4	4	1001	011	73.3
15	1	8	3	3	1010	010	80.0
15	1	10	2	2	1011	001	86.7
15	1	12	1	1	1100	000	93.3
14	1	1	6	6	0110	101	57.1
14	1	3	5	5	0111	100	64.3
14	1	5	4	4	1000	011	71.4
14	1	7	3	3	1001	010	78.6
14	1	9	2	2	1010	001	85.7
14	1	11	1	1	1011	000	92.9
13	1	2	5	5	0110	100	61.5
13	1	4	4	4	0111	011	69.2
13	1	6	3	3	1000	010	76.9
13	1	8	2	2	1001	001	84.6
13	1	10	1	1	1010	000	92.3
12	1	1	5	5	0101	100	58.3
12	1	3	4	4	0110	011	66.7
12	1	5	3	3	0111	010	75.0

Table 18-27 Settable bit rate combinations (3/3)

Valid bit rate setting					CnBTR register setting value		Sampling point (unit %)
DBT length	SYNC SEGMENT	PROP SEGMENT	PHASE SEGMENT1	PHASE SEGMENT2	TSEG1 [3:0]	TSEG2 [2:0]	
12	1	7	2	2	1000	001	83.3
12	1	9	1	1	1001	000	91.7
11	1	2	4	4	0101	011	63.6
11	1	4	3	3	0110	010	72.7
11	1	6	2	2	0111	001	81.8
11	1	8	1	1	1000	000	90.9
10	1	1	4	4	0100	011	60.0
10	1	3	3	3	0101	010	70.0
10	1	5	2	2	0110	001	80.0
10	1	7	1	1	0111	000	90.0
9	1	2	3	3	0100	010	66.7
9	1	4	2	2	0101	001	77.8
9	1	6	1	1	0110	000	88.9
8	1	1	3	3	0011	010	62.5
8	1	3	2	2	0100	001	75.0
8	1	5	1	1	0101	000	87.5
7[a]	1	2	2	2	0011	001	71.4
7[a]	1	4	1	1	0100	000	85.7
6[a]	1	1	2	2	0010	001	66.7
6[a]	1	3	1	1	0011	000	83.3
5[a]	1	2	1	1	0010	000	80.0
4[a]	1	1	1	1	0001	000	75.0

[a] Setting with a DBT value of 7 or less is valid only when the value of the CnBRP register is other than 00_H.

Caution The values in *Table 18-27* do not guarantee the operation of the network system. Thoroughly check the effect on the network system, taking into consideration oscillation errors and delays of the CAN bus and CAN transceiver.

18.15.2 Representative examples of baud rate settings

Table 18-28 and *Table 18-29* show representative examples of baud rate settings.

Table 18-28 Representative examples of baud rate settings (f_{CANMOD} = 8 MHz) (1/2)

Set baud rate value (unit: kbps)	Division ratio of CnBRP register	CnBRP register set value	Valid bit rate setting (unit: kbps)					CnBTR register setting value		Sampling point (unit: %)
			Length of DBT	SYNC SEGMENT	PROP SEGMENT	PHASE SEGMENT1	PHASE SEGMENT2	TSEG1 [3:0]	TSEG2 [2:0]	
1000	1	00000000	8	1	1	3	3	0011	010	62.5
1000	1	00000000	8	1	3	2	2	0100	001	75.0
1000	1	00000000	8	1	5	1	1	0101	000	87.5
500	1	00000000	16	1	1	7	7	0111	110	56.3
500	1	00000000	16	1	3	6	6	1000	101	62.5
500	1	00000000	16	1	5	5	5	1001	100	68.8
500	1	00000000	16	1	7	4	4	1010	011	75.0
500	1	00000000	16	1	9	3	3	1011	010	81.3
500	1	00000000	16	1	11	2	2	1100	001	87.5
500	1	00000000	16	1	13	1	1	1101	000	93.8
500	2	00000001	8	1	1	3	3	0011	010	62.5
500	2	00000001	8	1	3	2	2	0100	001	75.0
500	2	00000001	8	1	5	1	1	0101	000	87.5
250	2	00000001	16	1	1	7	7	0111	110	56.3
250	2	00000001	16	1	3	6	6	1000	101	62.5
250	2	00000001	16	1	5	5	5	1001	100	68.8
250	2	00000001	16	1	7	4	4	1010	011	75.0
250	2	00000001	16	1	9	3	3	1011	010	81.3
250	2	00000001	16	1	11	2	2	1100	001	87.5
250	2	00000001	16	1	13	1	1	1101	000	93.8
250	4	00000011	8	1	3	2	2	0100	001	75.0
250	4	00000011	8	1	5	1	1	0101	000	87.5
125	4	00000011	16	1	1	7	7	0111	110	56.3
125	4	00000011	16	1	3	6	6	1000	101	62.5
125	4	00000011	16	1	5	5	5	1001	100	68.8
125	4	00000011	16	1	7	4	4	1010	011	75.0
125	4	00000011	16	1	9	3	3	1011	010	81.3
125	4	00000011	16	1	11	2	2	1100	001	87.5
125	4	00000011	16	1	13	1	1	1101	000	93.8
125	8	00000111	8	1	3	2	2	0100	001	75.0
125	8	00000111	8	1	5	1	1	0101	000	87.5
100	4	00000011	20	1	7	6	6	1100	101	70.0
100	4	00000011	20	1	9	5	5	1101	100	75.0
100	5	00000100	16	1	7	4	4	1010	011	75.0
100	5	00000100	16	1	9	3	3	1011	010	81.3

Table 18-28 Representative examples of baud rate settings (f_{CANMOD} = 8 MHz) (2/2)

Set baud rate value (unit: kbps)	Division ratio of CnBRP register	CnBRP register set value	Valid bit rate setting (unit: kbps)					CnBTR register setting value		Sampling point (unit: %)
			Length of DBT	SYNC SEGMENT	PROP SEGMENT	PHASE SEGMENT1	PHASE SEGMENT2	TSEG1 [3:0]	TSEG2 [2:0]	
100	8	00000111	10	1	3	3	3	0101	010	70.0
100	8	00000111	10	1	5	2	2	0110	001	80.0
100	10	00001001	8	1	3	2	2	0100	001	75.0
100	10	00001001	8	1	5	1	1	0101	000	87.5
83.3	4	00000011	24	1	7	8	8	1110	111	66.7
83.3	4	00000011	24	1	9	7	7	1111	110	70.8
83.3	6	00000101	16	1	5	5	5	1001	100	68.8
83.3	6	00000101	16	1	7	4	4	1010	011	75.0
83.3	6	00000101	16	1	9	3	3	1011	010	81.3
83.3	6	00000101	16	1	11	2	2	1100	001	87.5
83.3	8	00000111	12	1	5	3	3	0111	010	75.0
83.3	8	00000111	12	1	7	2	2	1000	001	83.3
83.3	12	00001011	8	1	3	2	2	0100	001	75.0
83.3	12	00001011	8	1	5	1	1	0101	000	87.5
33.3	10	00001001	24	1	7	8	8	1110	111	66.7
33.3	10	00001001	24	1	9	7	7	1111	110	70.8
33.3	12	00001011	20	1	7	6	6	1100	101	70.0
33.3	12	00001011	20	1	9	5	5	1101	100	75.0
33.3	15	00001110	16	1	7	4	4	1010	011	75.0
33.3	15	00001110	16	1	9	3	3	1011	010	81.3
33.3	16	00001111	15	1	6	4	4	1001	011	73.3
33.3	16	00001111	15	1	8	3	3	1010	010	80.0
33.3	20	00010011	12	1	5	3	3	0111	010	75.0
33.3	20	00010011	12	1	7	2	2	1000	001	83.3
33.3	24	00010111	10	1	3	3	3	0101	010	70.0
33.3	24	00010111	10	1	5	2	2	0110	001	80.0
33.3	30	00011101	8	1	3	2	2	0100	001	75.0
33.3	30	00011101	8	1	5	1	1	0101	000	87.5

Caution The values in *Table 18-28* do not guarantee the operation of the network system. Thoroughly check the effect on the network system, taking into consideration oscillation errors and delays of the CAN bus and CAN transceiver.

Table 18-29 Representative examples of baud rate settings (f_{CANMOD} = 16 MHz) (1/2)

Set baud rate value (unit: kbps)	Division ratio of CnBRP register	CnBRP register set value	Valid bit rate setting (unit: kbps)					CnBTR register setting value		Sampling point (unit: %)
			Length of DBT	SYNC SEGMENT	PROP SEGMENT	PHASE SEGMENT1	PHASE SEGMENT2	TSEG1 [3:0]	TSEG2 [2:0]	
1000	1	00000000	16	1	1	7	7	0111	110	56.3
1000	1	00000000	16	1	3	6	6	1000	101	62.5
1000	1	00000000	16	1	5	5	5	1001	100	68.8
1000	1	00000000	16	1	7	4	4	1010	011	75.0
1000	1	00000000	16	1	9	3	3	1011	010	81.3
1000	1	00000000	16	1	11	2	2	1100	001	87.5
1000	1	00000000	16	1	13	1	1	1101	000	93.8
1000	2	00000001	8	1	3	2	2	0100	001	75.0
1000	2	00000001	8	1	5	1	1	0101	000	87.5
500	2	00000001	16	1	1	7	7	0111	110	56.3
500	2	00000001	16	1	3	6	6	1000	101	62.5
500	2	00000001	16	1	5	5	5	1001	100	68.8
500	2	00000001	16	1	7	4	4	1010	011	75.0
500	2	00000001	16	1	9	3	3	1011	010	81.3
500	2	00000001	16	1	11	2	2	1100	001	87.5
500	2	00000001	16	1	13	1	1	1101	000	93.8
500	4	00000011	8	1	3	2	2	0100	001	75.0
500	4	00000011	8	1	5	1	1	0101	000	87.5
250	4	00000011	16	1	3	6	6	1000	101	62.5
250	4	00000011	16	1	5	5	5	1001	100	68.8
250	4	00000011	16	1	7	4	4	1010	011	75.0
250	4	00000011	16	1	9	3	3	1011	010	81.3
250	4	00000011	16	1	11	2	2	1100	001	87.5
250	8	00000111	8	1	3	2	2	0100	001	75.0
250	8	00000111	8	1	5	1	1	0101	000	87.5
125	8	00000111	16	1	3	6	6	1000	101	62.5
125	8	00000111	16	1	7	4	4	1010	011	75.0
125	8	00000111	16	1	9	3	3	1011	010	81.3
125	8	00000111	16	1	11	2	2	1100	001	87.5
125	16	00001111	8	1	3	2	2	0100	001	75.0
125	16	00001111	8	1	5	1	1	0101	000	87.5
100	8	00000111	20	1	9	5	5	1101	100	75.0
100	8	00000111	20	1	11	4	4	1110	011	80.0
100	10	00001001	16	1	7	4	4	1010	011	75.0
100	10	00001001	16	1	9	3	3	1011	010	81.3
100	16	00001111	10	1	3	3	3	0101	010	70.0
100	16	00001111	10	1	5	2	2	0110	001	80.0
100	20	00010011	8	1	3	2	2	0100	001	75.0

Table 18-29 Representative examples of baud rate settings (f_{CANMOD} = 16 MHz) (2/2)

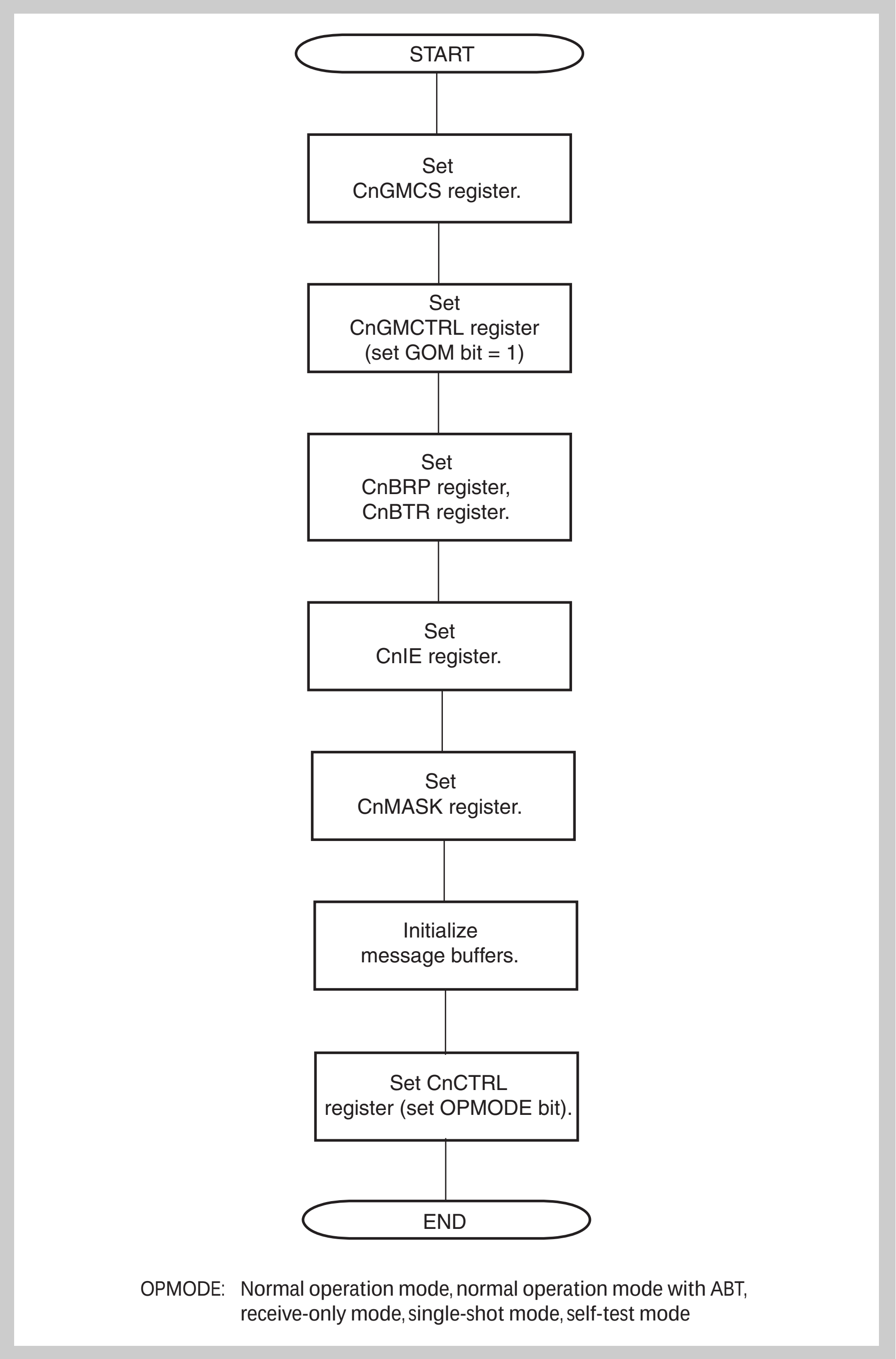
Set baud rate value (unit: kbps)	Division ratio of CnBRP register	CnBRP register set value	Valid bit rate setting (unit: kbps)					CnBTR register setting value		Sampling point (unit: %)
			Length of DBT	SYNC SEGMENT	PROP SEGMENT	PHASE SEGMENT1	PHASE SEGMENT2	TSEG1 [3:0]	TSEG2 [2:0]	
83.3	8	00000111	24	1	7	8	8	1110	111	66.7
83.3	8	00000111	24	1	9	7	7	1111	110	70.8
83.3	12	00001011	16	1	7	4	4	1010	011	75.0
83.3	12	00001011	16	1	9	3	3	1011	010	81.3
83.3	12	00001011	16	1	11	2	2	1100	001	87.5
83.3	16	00001111	12	1	5	3	3	0111	010	75.0
83.3	16	00001111	12	1	7	2	2	1000	001	83.3
83.3	24	00010111	8	1	3	2	2	0100	001	75.0
83.3	24	00010111	8	1	5	1	1	0101	000	87.5
33.3	30	00011101	24	1	7	8	8	1110	111	66.7
33.3	30	00011101	24	1	9	7	7	1111	110	70.8
33.3	24	00010111	20	1	9	5	5	1101	100	75.0
33.3	24	00010111	20	1	11	4	4	1110	011	80.0
33.3	30	00011101	16	1	7	4	4	1010	011	75.0
33.3	30	00011101	16	1	9	3	3	1011	010	81.3
33.3	32	00011111	15	1	8	3	3	1010	010	80.0
33.3	32	00011111	15	1	10	2	2	1011	001	86.7
33.3	37	00100100	13	1	6	3	3	1000	010	76.9
33.3	37	00100100	13	1	8	2	2	1001	001	84.6
33.3	40	00100111	12	1	5	3	3	0111	010	75.0
33.3	40	00100111	12	1	7	2	2	1000	001	83.3
33.3	48	00101111	10	1	3	3	3	0101	010	70.0
33.3	48	00101111	10	1	5	2	2	0110	001	80.0
33.3	60	00111011	8	1	3	2	2	0100	001	75.0
33.3	60	00111011	8	1	5	1	1	0101	000	87.5

Caution The values in *Table 18-29* do not guarantee the operation of the network system. Thoroughly check the effect on the network system, taking into consideration oscillation errors and delays of the CAN bus and CAN transceiver.

18.16 Operation of CAN Controller

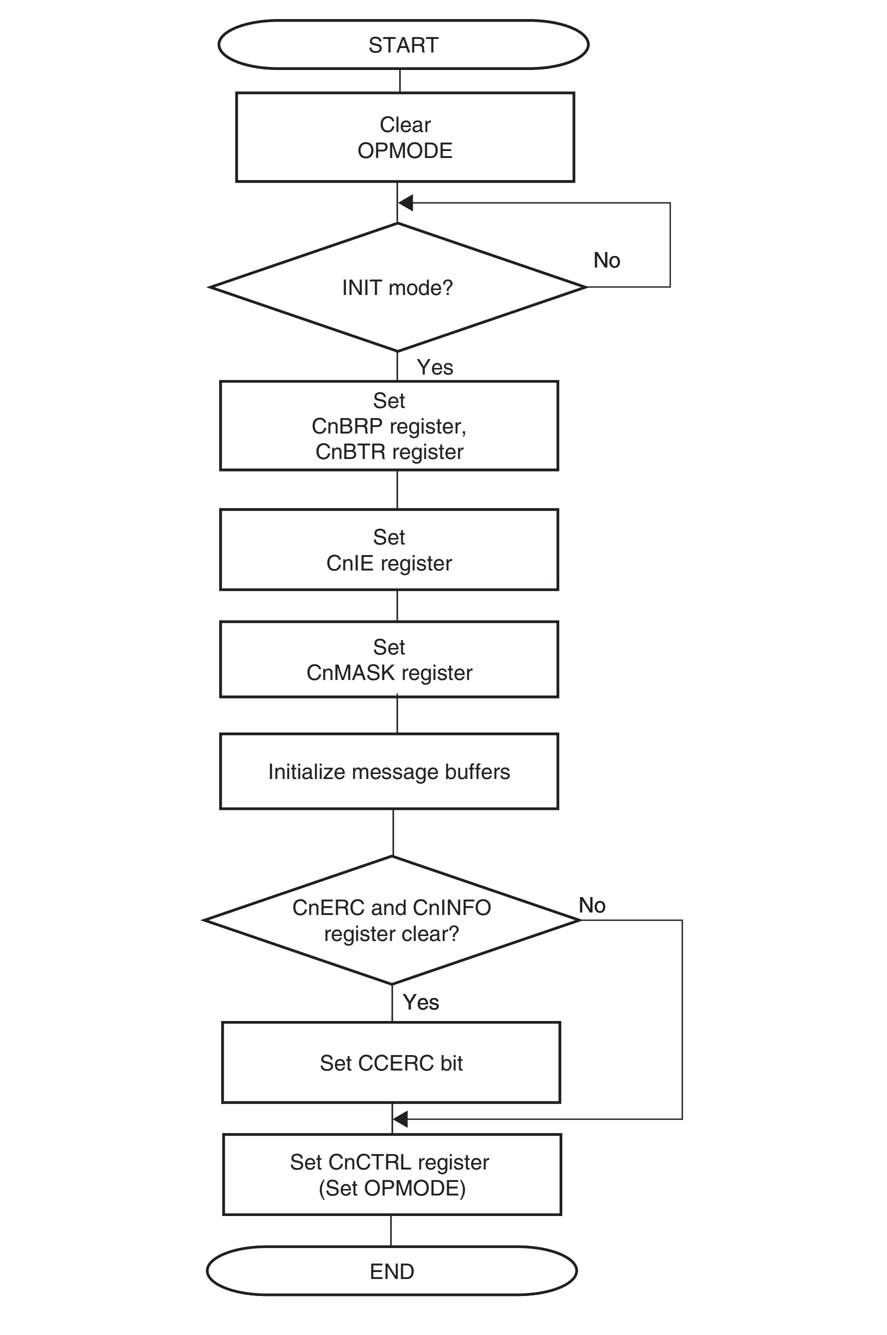
The processing procedure for showing in this chapter is recommended processing procedure to operate CAN controller.

Develop the program referring to recommended processing procedure in this chapter.



OPMODE: Normal operation mode, normal operation mode with ABT, receive-only mode, single-shot mode, self-test mode

Figure 18-35 Initialization


START
Clear OPMODE
INIT mode?
No
Yes
Set CnBRP register, CnBTR register
Set CnIE register
Set CnMASK register
Initialize message buffers
CnERC and CnINFO register clear?
No
Yes
Set CCERC bit
Set CnCTRL register (Set OPMODE)
END

OPMODE: Normal operation mode, normal operation mode with ABT, receive-only mode, single-shot mode, self-test mode

Figure 18-36 Re-initialization

Caution After setting the CAN module to the initialization mode, avoid setting the module to another operation mode immediately after. If it is necessary to immediately set the module to another operation mode, be sure to access registers other than the CnCTRL and CnGMCTRL registers (e.g., set a message buffer).

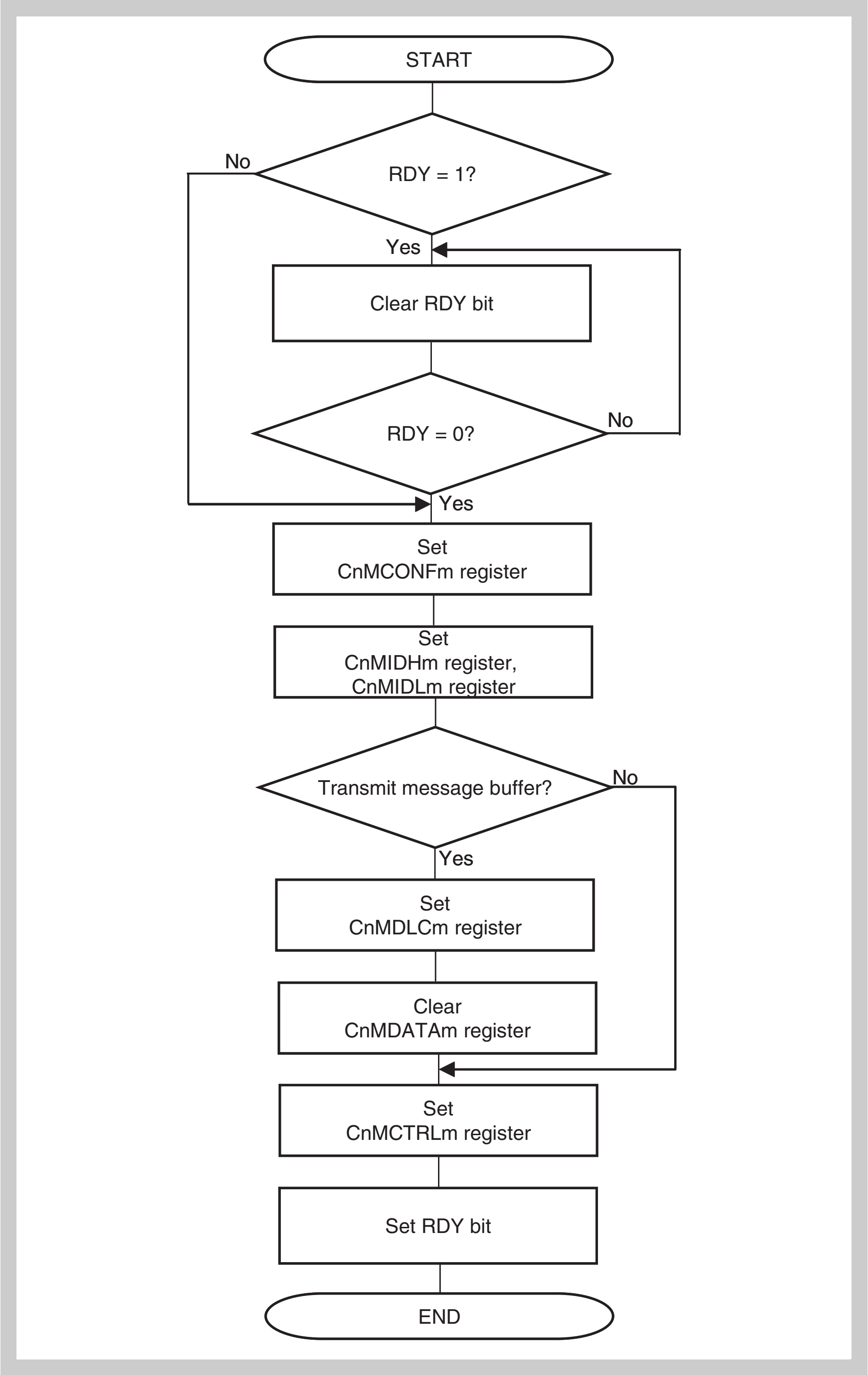
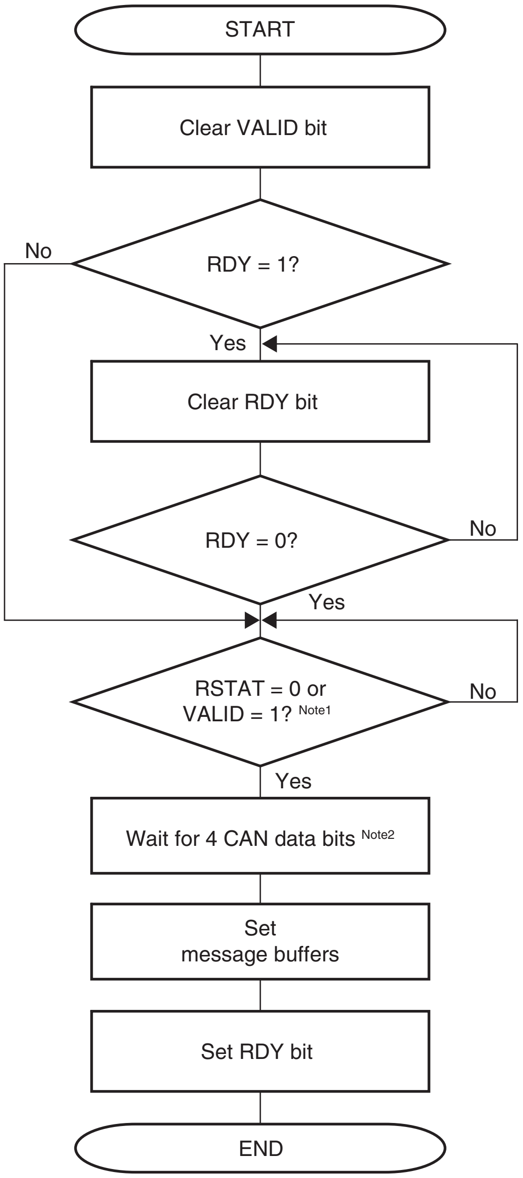


Figure 18-37 Message buffer initialization

Caution

1. Before a message buffer is initialized, the RDY bit must be cleared.
2. Make the following settings for message buffers not used by the application.
   - Clear the RDY, TRQ, and DN bits of the CnMCTRLm register to 0.
   - Clear the MA0 bit of the CnMCONFm register to 0.

Figure 18-38 shows the processing for a receive message buffer (MT[2:0] bits of CnMCONFm register = 001_B to 101_B).



Note1: Confirm that a message is being received because RDY bit must be set after a message is completely received.

Note2: Avoid message buffer redefinition during store operation of message reception by waiting additional 4 CAN data bits.

Figure 18-38 Message buffer redefinition

Figure 18-39 shows the processing for a transmit message buffer during transmission (MT[2:0] bits of CnMCONFm register = 000_B).

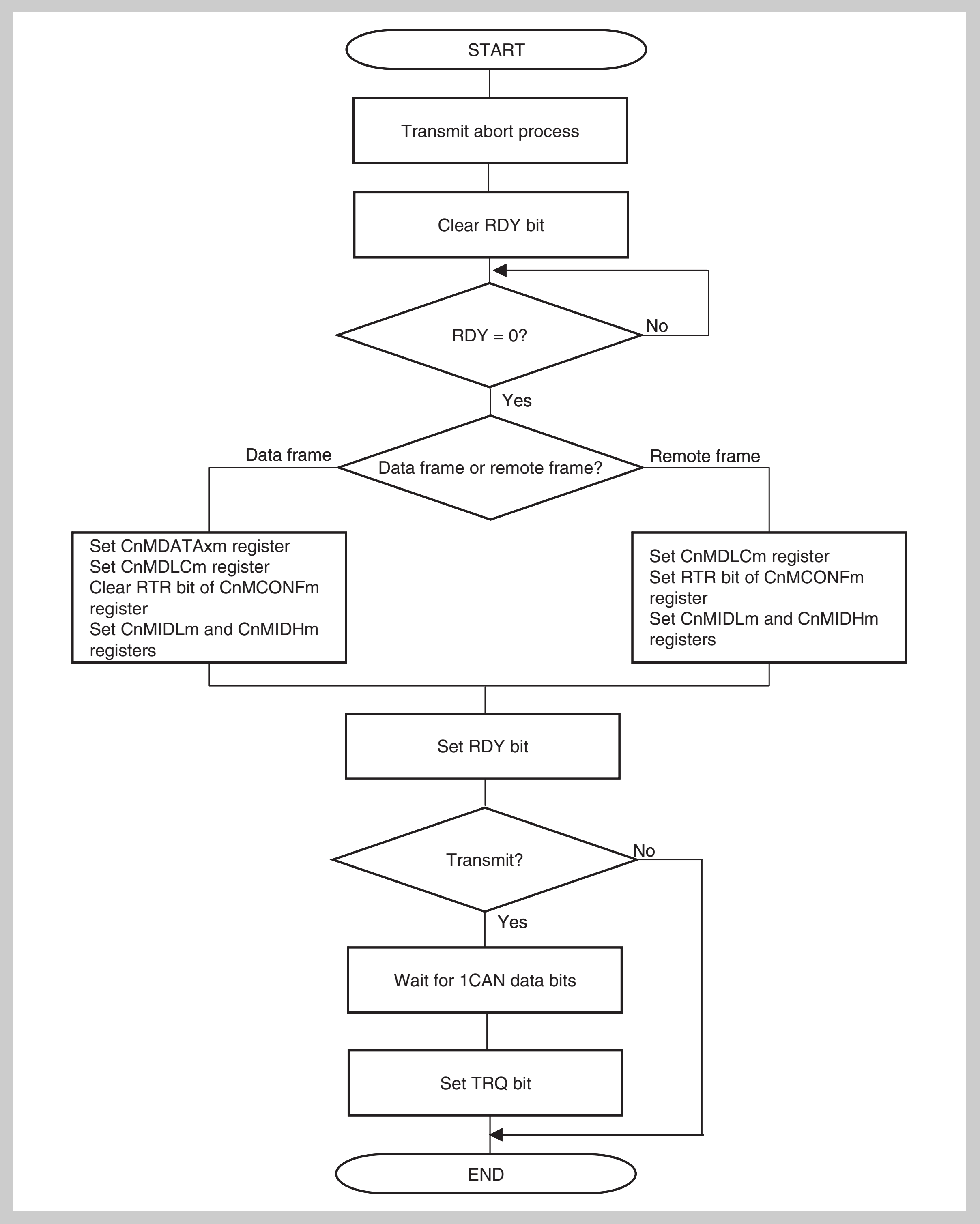


Figure 18-39 Message buffer redefinition during transmission

Figure 18-40 shows the processing for a transmit message buffer (MT[2:0] bits of CnMCONFm register = 000_B).

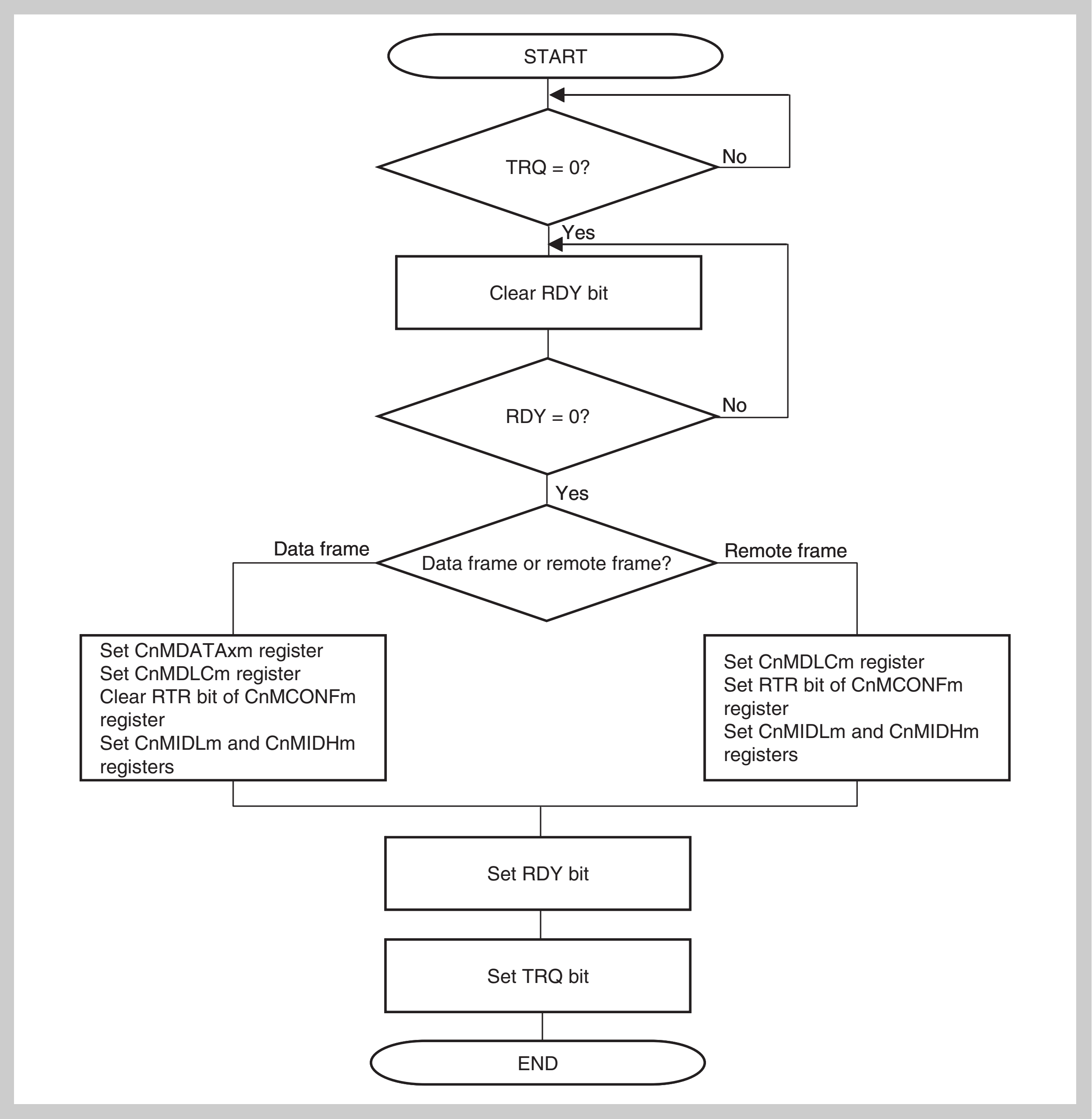

START
TRQ = 0?
No
Yes
Clear RDY bit
RDY = 0?
No
Yes
Data frame
Data frame or remote frame?
Remote frame
Set CnMDATAxm register
Set CnMDLCm register
Clear RTR bit of CnMCONFm register
Set CnMIDLm and CnMIDHm registers
Set CnMDLCm register
Set RTR bit of CnMCONFm register
Set CnMIDLm and CnMIDHm registers
Set RDY bit
Set TRQ bit
END

Figure 18-40 Message transmit processing

Caution

1. The TRQ bit should be set after the RDY bit is set.
2. The RDY bit and TRQ bit should not be set at the same time.

Figure 18-41 shows the processing for a transmit message buffer (MT[2:0] bits of CnMCONFm register = 000_B)

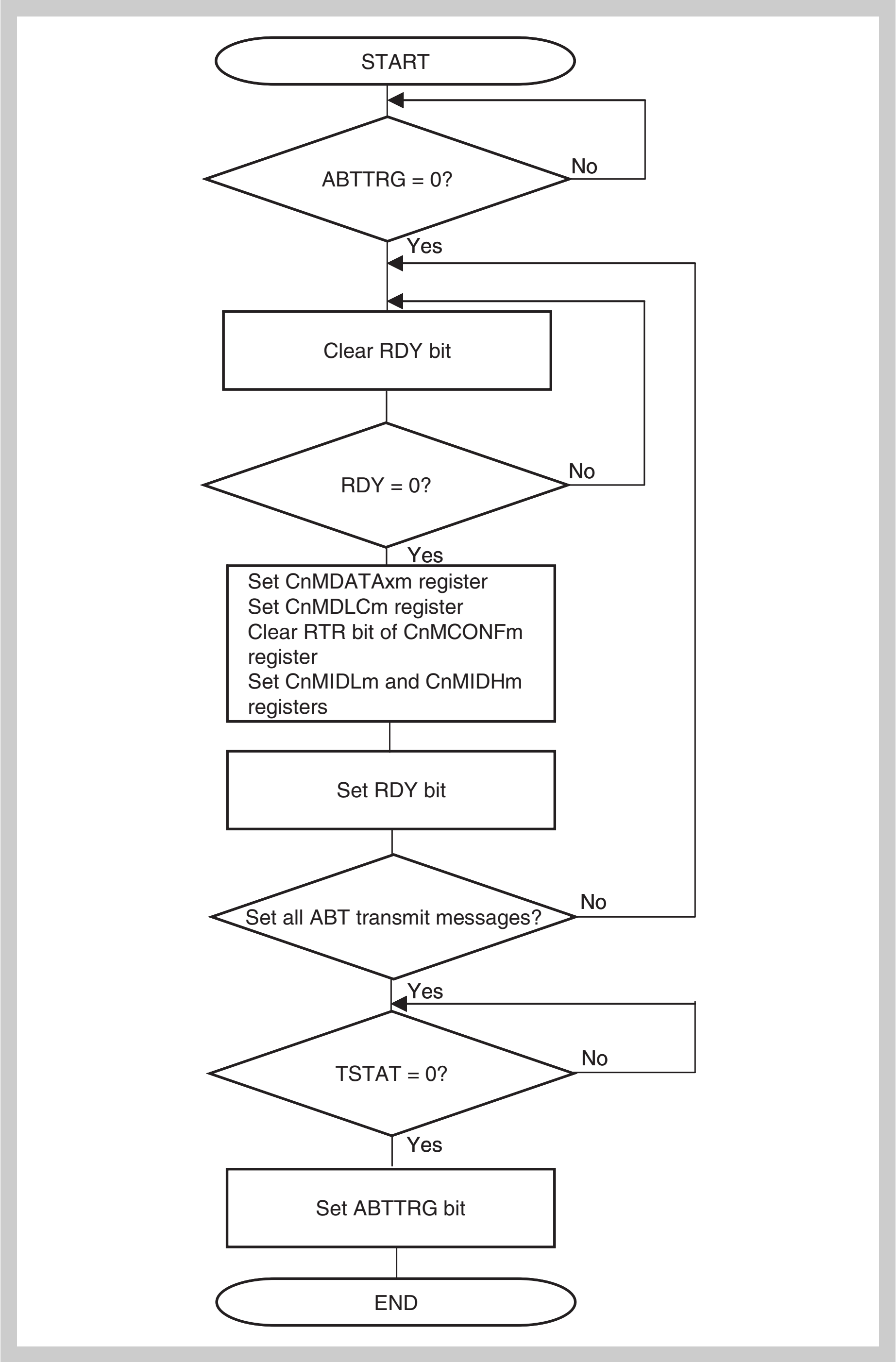


Figure 18-41 ABT message transmit processing

Caution The ABTTRG bit should be set to 1 after the TSTAT bit is cleared to 0. Checking the TSTAT bit and setting the ABTTRG bit to 1 must be processed consecutively.

Note This processing (normal operation mode with ABT) can only be applied to message buffers 0 to 7. For message buffers other than the ABT message buffers, see *Figure 18-40 on page 661*.

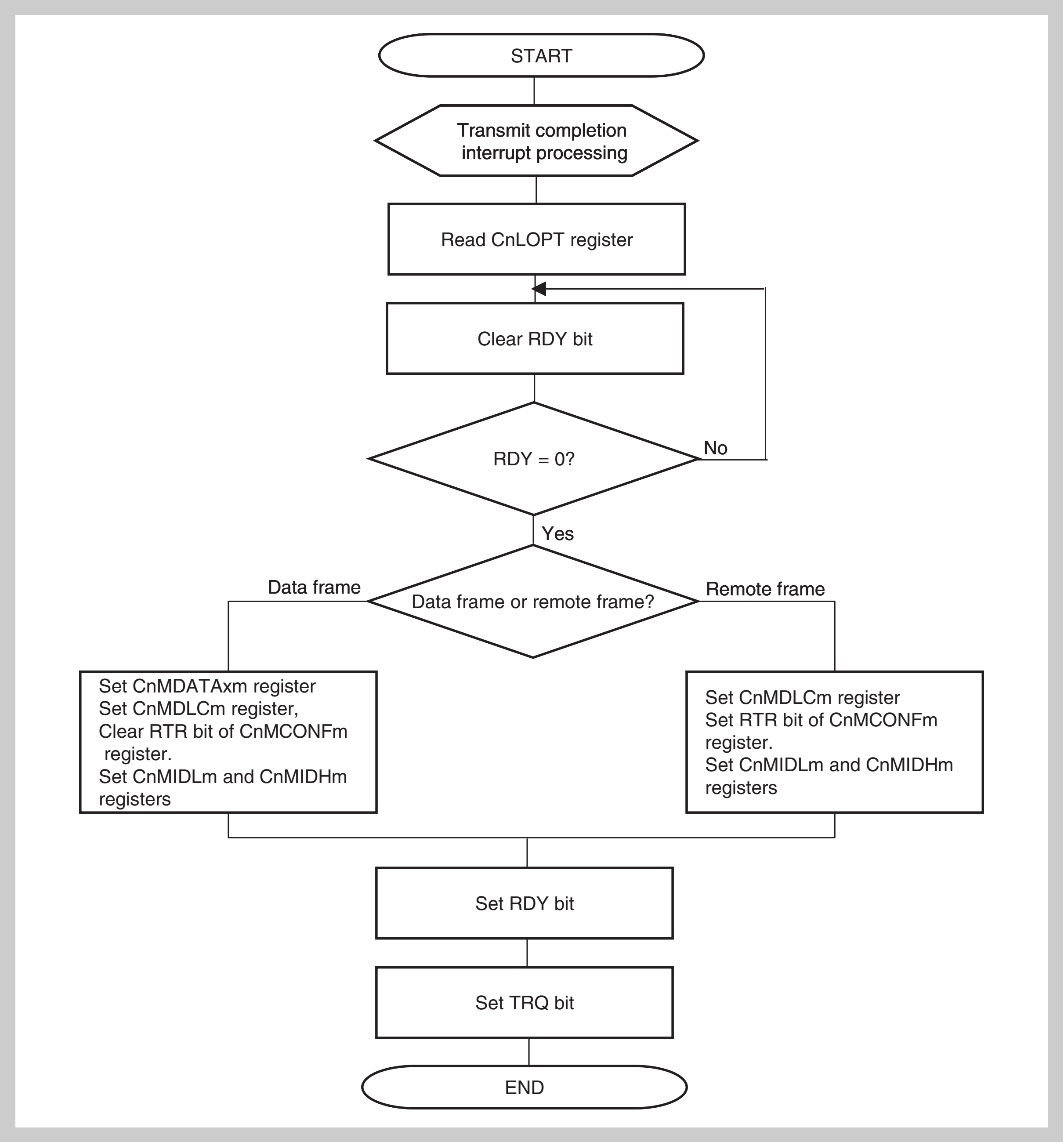

START
Transmit completion interrupt processing
Read CnLOPT register
Clear RDY bit
RDY = 0?
No
Yes
Data frame
Data frame or remote frame?
Remote frame
Set CnMDATAxm register
Set CnMDLCm register,
Clear RTR bit of CnMCONFm register.
Set CnMIDLm and CnMIDHm registers
Set CnMDLCm register
Set RTR bit of CnMCONFm register.
Set CnMIDLm and CnMIDHm registers
Set RDY bit
Set TRQ bit
END

Figure 18-42 Transmission via interrupt (using CnLOPT register)

Caution 1. The TRQ bit should be set after the RDY bit is set.

2. The RDY bit and TRQ bit should not be set at the same time.

Note Also check the MBON flag at the beginning and at the end of the interrupt routine, in order to check the access to the message buffers as well as TX history list registers, in case a pending sleep mode had been executed. If MBON is detected to be cleared at any check, the actions and results of the processing have to be discarded and processed again, after MBON is set again.
It is recommended to cancel any sleep mode requests, before processing TX interrupts.

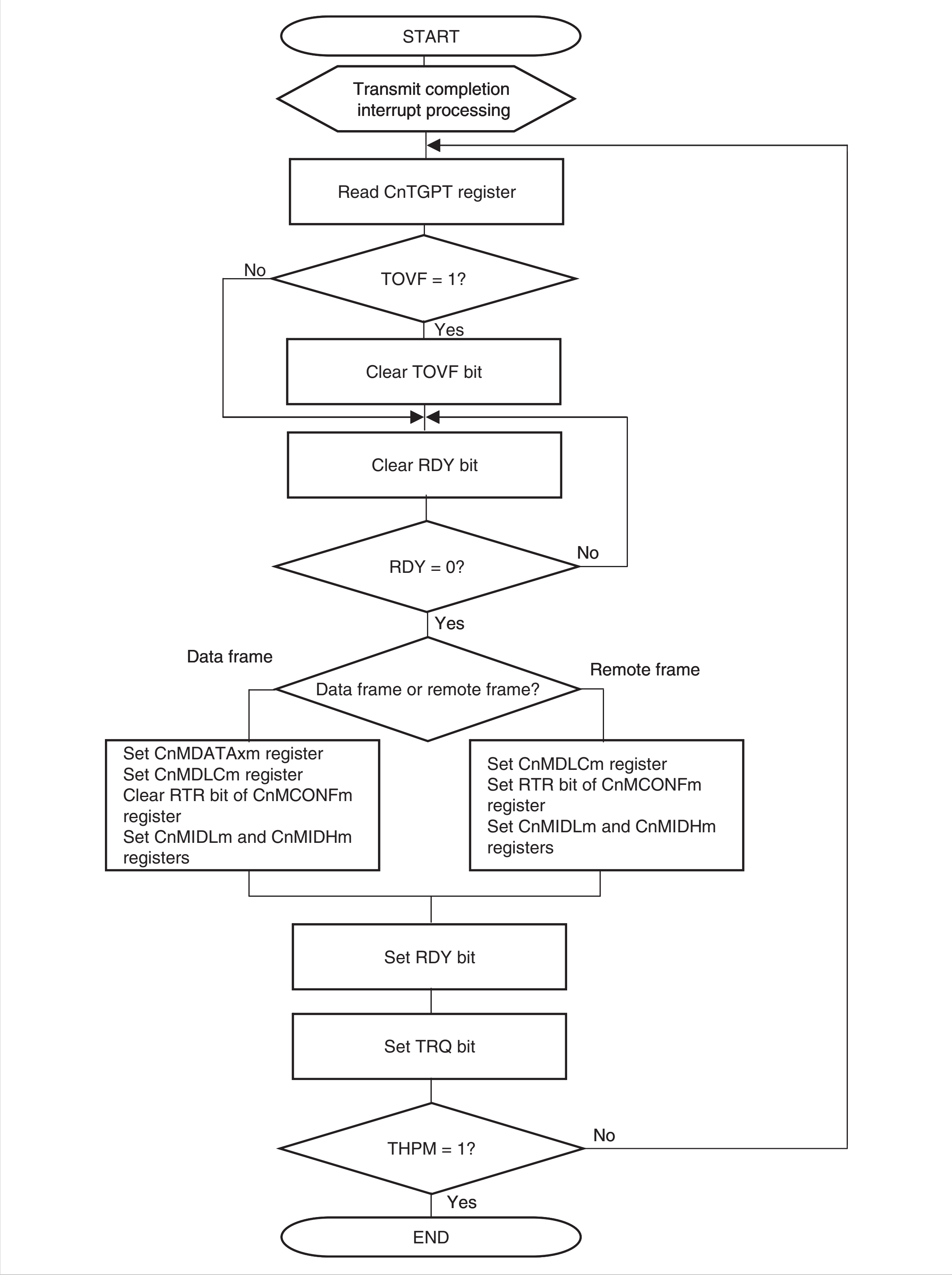


Figure 18-43 Transmission via interrupt (using CnTGPT register)

Caution

1. The TRQ bit should be set after the RDY bit is set.
2. The RDY bit and TRQ bit should not be set at the same time.

Note

1. Also check the MBON flag at the beginning and at the end of the interrupt routine, in order to check the access to the message buffers as well as TX history list registers, in case a pending sleep mode had been executed. If MBON is detected to be cleared at any check, the actions and results of the processing have to be discarded and processed again, after MBON is set again.
   It is recommended to cancel any sleep mode requests, before processing TX interrupts.

2. If TOVF was set once, the transmit history list is inconsistent. Consider to scan all configured transmit buffers for completed transmissions.

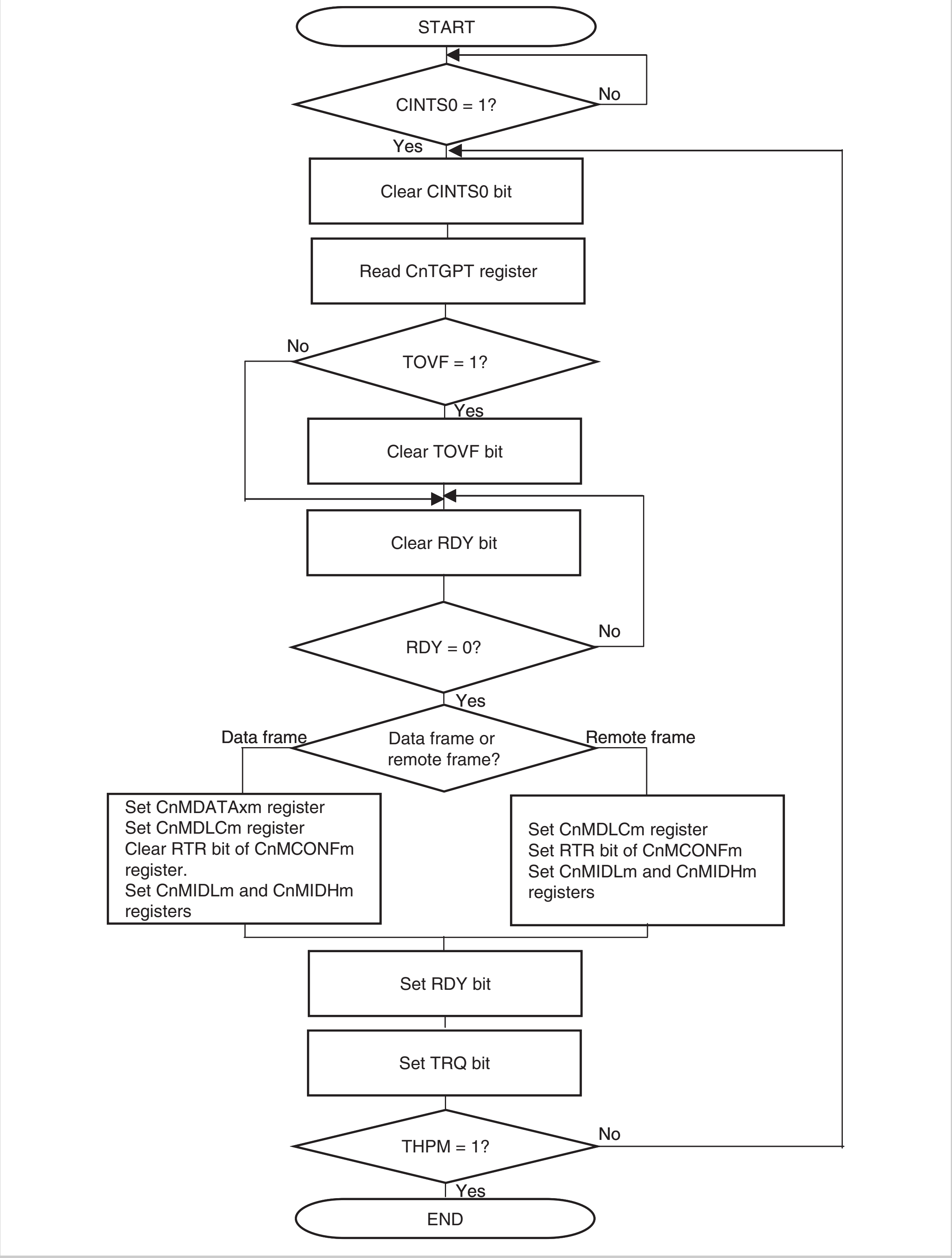


Figure 18-44 Transmission via software polling

Caution
1. The TRQ bit should be set after the RDY bit is set.
2. The RDY bit and TRQ bit should not be set at the same time.

Note

1. Also check the MBON flag at the beginning and at the end of the polling routine, in order to check the access to the message buffers as well as TX history list registers, in case a pending sleep mode had been executed. If MBON is detected to be cleared at any check, the actions and results of the processing have to be discarded and processed again, after MBON is set again.
2. If TOVF was set once, the transmit history list is inconsistent. Consider to scan all configured transmit buffers for completed transmissions.

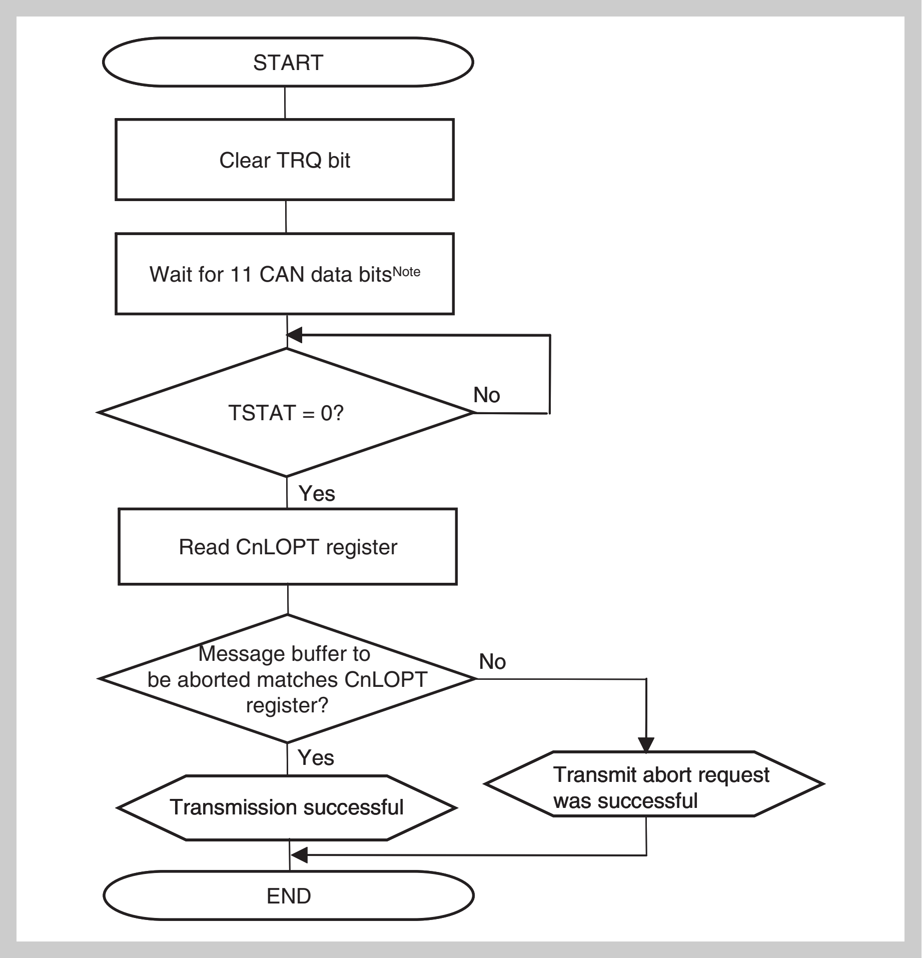

START
Clear TRQ bit
Wait for 11 CAN data bitsNote
TSTAT = 0?
No
Yes
Read CnLOPT register
Message buffer to be aborted matches CnLOPT register?
No
Yes
Transmission successful
Transmit abort request was successful
END

Figure 18-45 Transmission abort processing (except normal operation mode with ABT)

Caution

1. Clear the TRQ bit for aborting transmission request, not the RDY bit.
2. Before making a sleep mode transition request, confirm that there is no transmission request left using this processing.
3. The TSTAT bit can be periodically checked by a user application or can be checked after the transmit completion interrupt.
4. Do not execute any new transmission request including in the other message buffers while transmission abort processing is in progress.

Note There is a possibility of starting the transmission without being aborted even if TRQ bit is cleared, because the transmission request to protocol layer might already been accepted between 11 bits, total of interframe space (3 bits) and suspend transmission (8 bits).

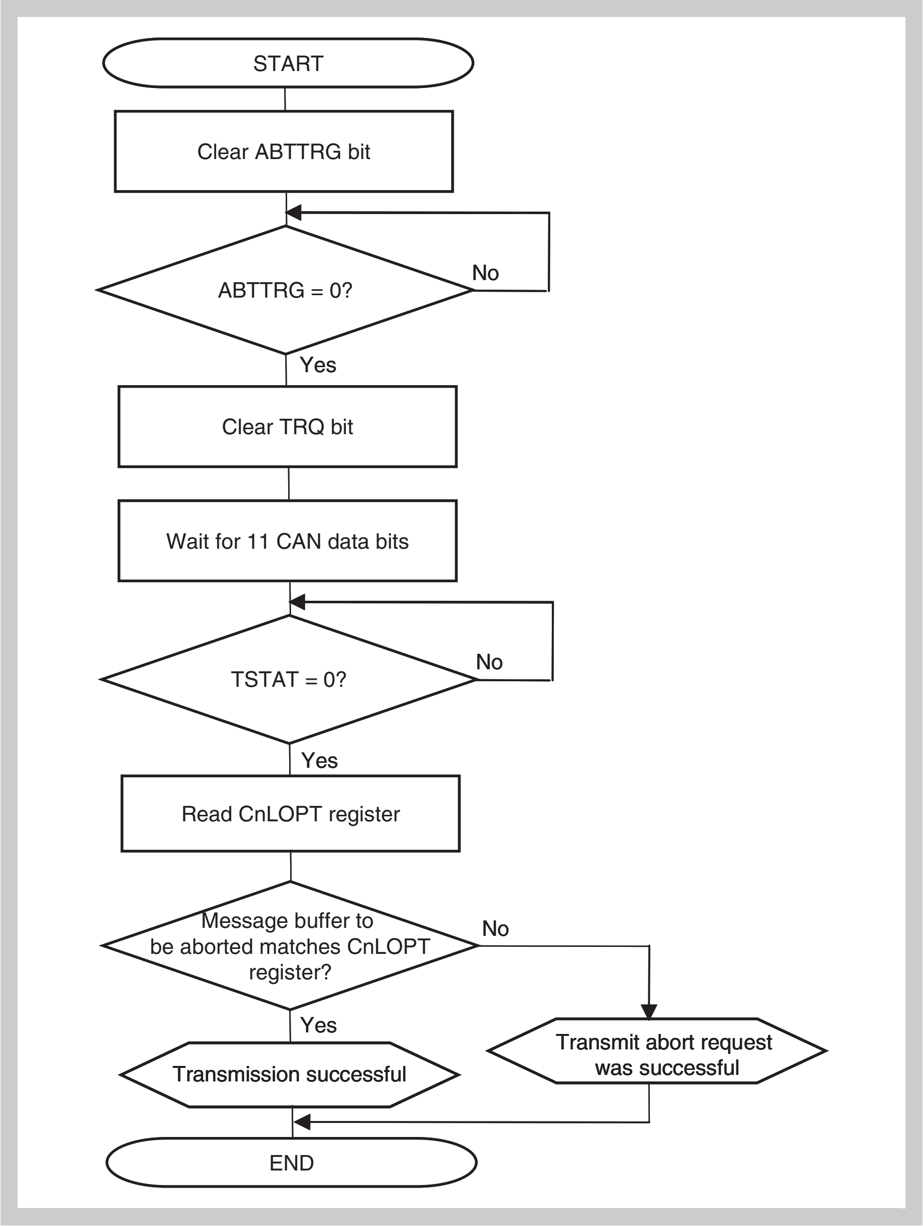


Figure 18-46 **Transmission abort processing except for ABT transmission (normal operation mode with ABT)**

Caution

1. Clear the TRQ bit for aborting transmission request, not the RDY bit.
2. Before making a sleep mode transition request, confirm that there is no transmission request left using this processing.
3. The TSTAT bit can be periodically checked by a user application or can be checked after the transmit completion interrupt.
4. Do not execute any new transmission request including in the other message buffers while transmission abort processing is in progress.

Figure 18-47 shows the processing to skip resumption of transmitting a message that was stopped when transmission of an ABT message buffer was aborted.

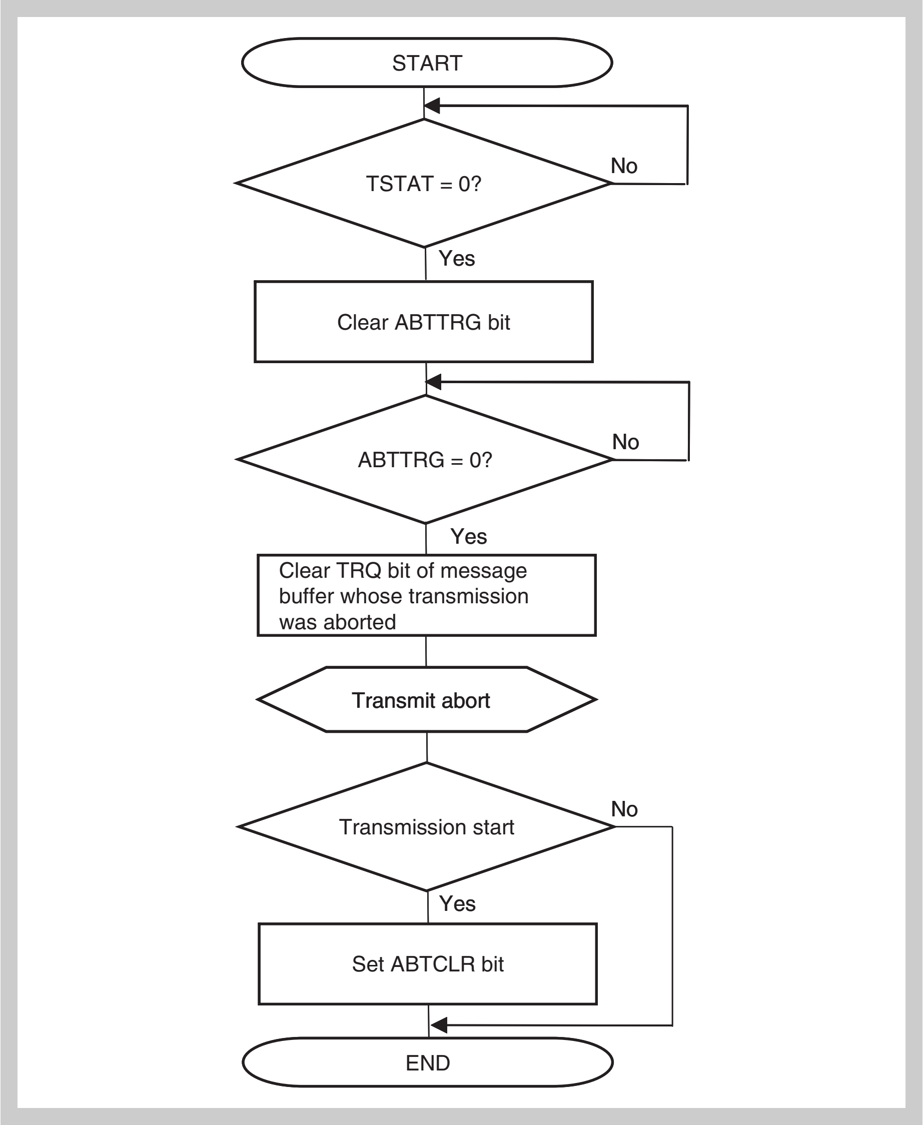


Figure 18-47 Transmission abort processing (normal operation mode with ABT)

Caution

1. Do not set any transmission requests while ABT transmission abort processing is in progress.
2. Make a CAN sleep mode/CAN stop mode transition request after the ABTTRG bit is cleared (after ABT mode is aborted) following the procedure shown in *Figure 18-47* or *Figure 18-48*. When clearing a transmission request in an area other than the ABT area, follow the procedure shown in *Figure 18-45 on page 668*.

Figure 18-48 shows the processing to not skip resumption of transmitting a message that was stopped when transmission of an ABT message buffer was aborted.

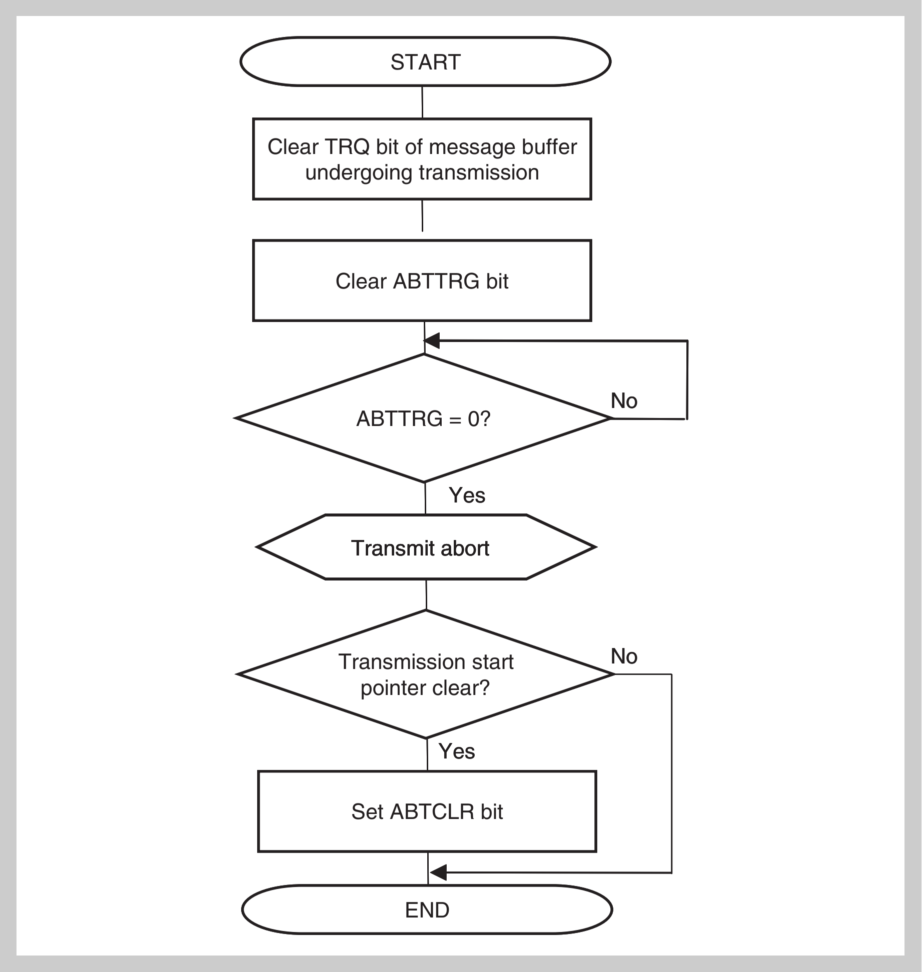
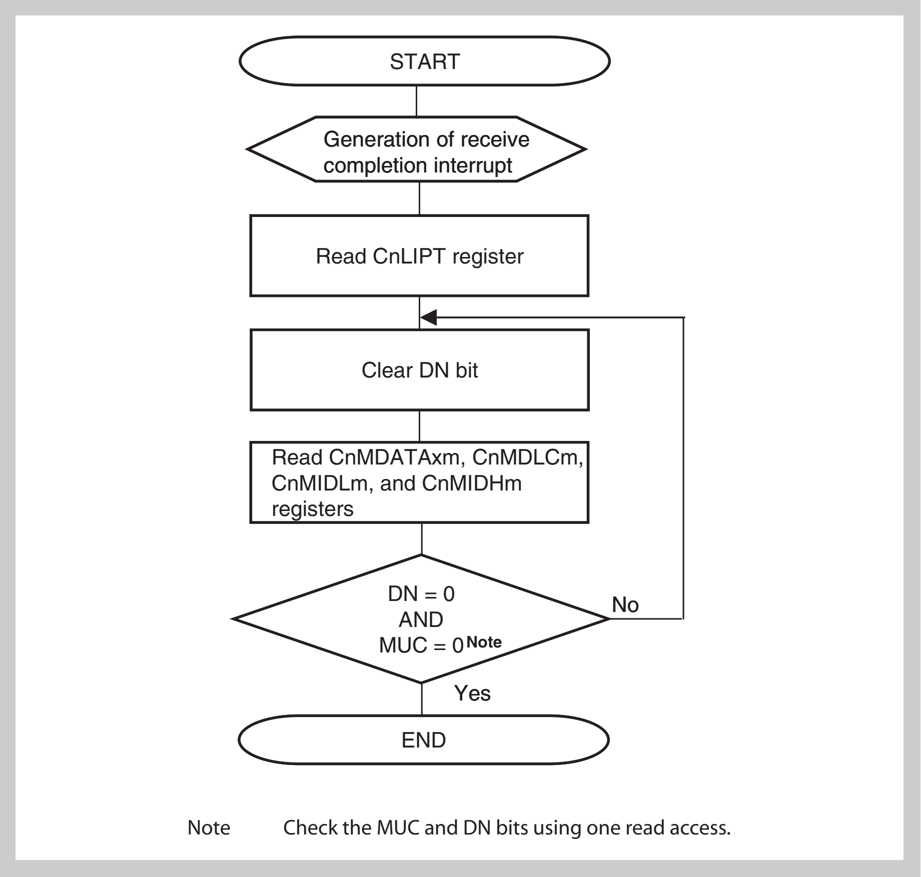

START
Clear TRQ bit of message buffer undergoing transmission
Clear ABTTRG bit
ABTTRG = 0?
No
Yes
Transmit abort
Transmission start pointer clear?
No
Yes
Set ABTCLR bit
END

Figure 18-48 ABT transmission request abort processing (normal operation mode with ABT)

Caution

1. Do not set any transmission requests while ABT transmission abort processing is in progress.
2. Make a CAN sleep mode/CAN stop mode request after the ABTTRG bit is cleared (after ABT mode is stopped) following the procedure shown in *Figure 18-47* or *Figure 18-48*. When clearing a transmission request in an area other than the ABT area, follow the procedure shown in *Figure 18-45 on page 668*.


START
Generation of receive completion interrupt
Read CnLIPT register
Clear DN bit
Read CnMDATAxm, CnMDLCm, CnMIDLm, and CnMIDHm registers
DN = 0 AND MUC = 0 Note
No
Yes
END

Note Check the MUC and DN bits using one read access.

Figure 18-49 Reception via interrupt (using CnLIPT register)

Note Also check the MBON flag at the beginning and at the end of the interrupt routine, in order to check the access to the message buffers as well as reception history list registers, in case a pending sleep mode had been executed. If MBON is detected to be cleared at any check, the actions and results of the processing have to be discarded and processed again, after MBON is set again.
It is recommended to cancel any sleep mode requests, before processing RX interrupts.

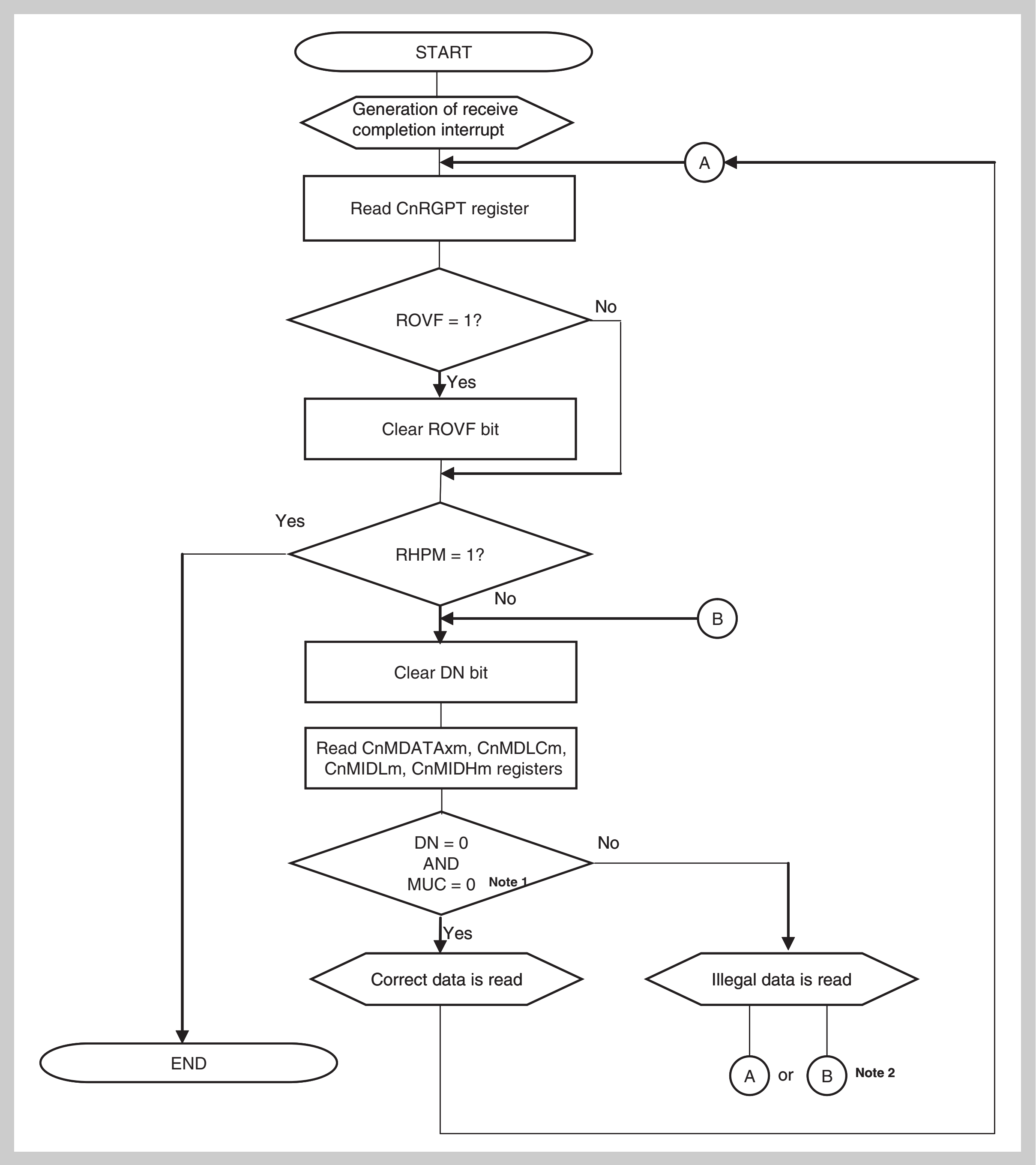


Figure 18-50 Reception via interrupt (using CnRGPT register)

Note
1. Check the MUC and DN bits using one read access.
2. Depending of the processing target of the application, two ways are possible:
– Way A: The message is not processed within this pass, but with the next pass, depending on the timing this can happen latest with the next Receive Interrupt.
Other messages will be processed earlier.
– Way B: The message is processed within this pass, the loop waits on this message.
Other messages will be processed later.

3. Also check the MBON flag at the beginning and at the end of the interrupt routine, in order to check the access to the message buffers as well as reception history list registers, in case a pending sleep mode had been executed. If MBON is detected to be cleared at any check, the actions and results of the processing have to be discarded and processed again, after MBON is set again.
   It is recommended to cancel any sleep mode requests, before processing RX interrupts.
4. If ROVF was set once, the receive history list is inconsistent. Consider to scan all configured receive buffers for receptions.

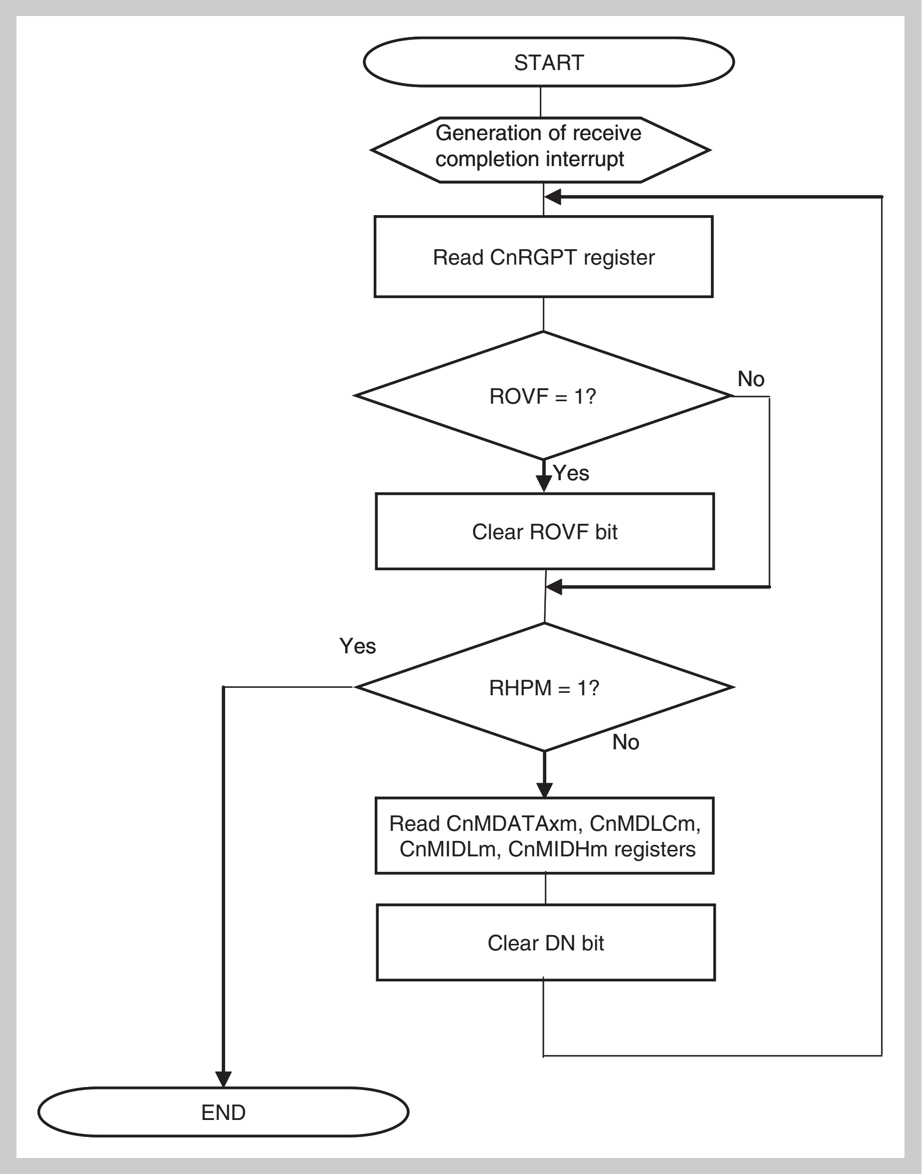
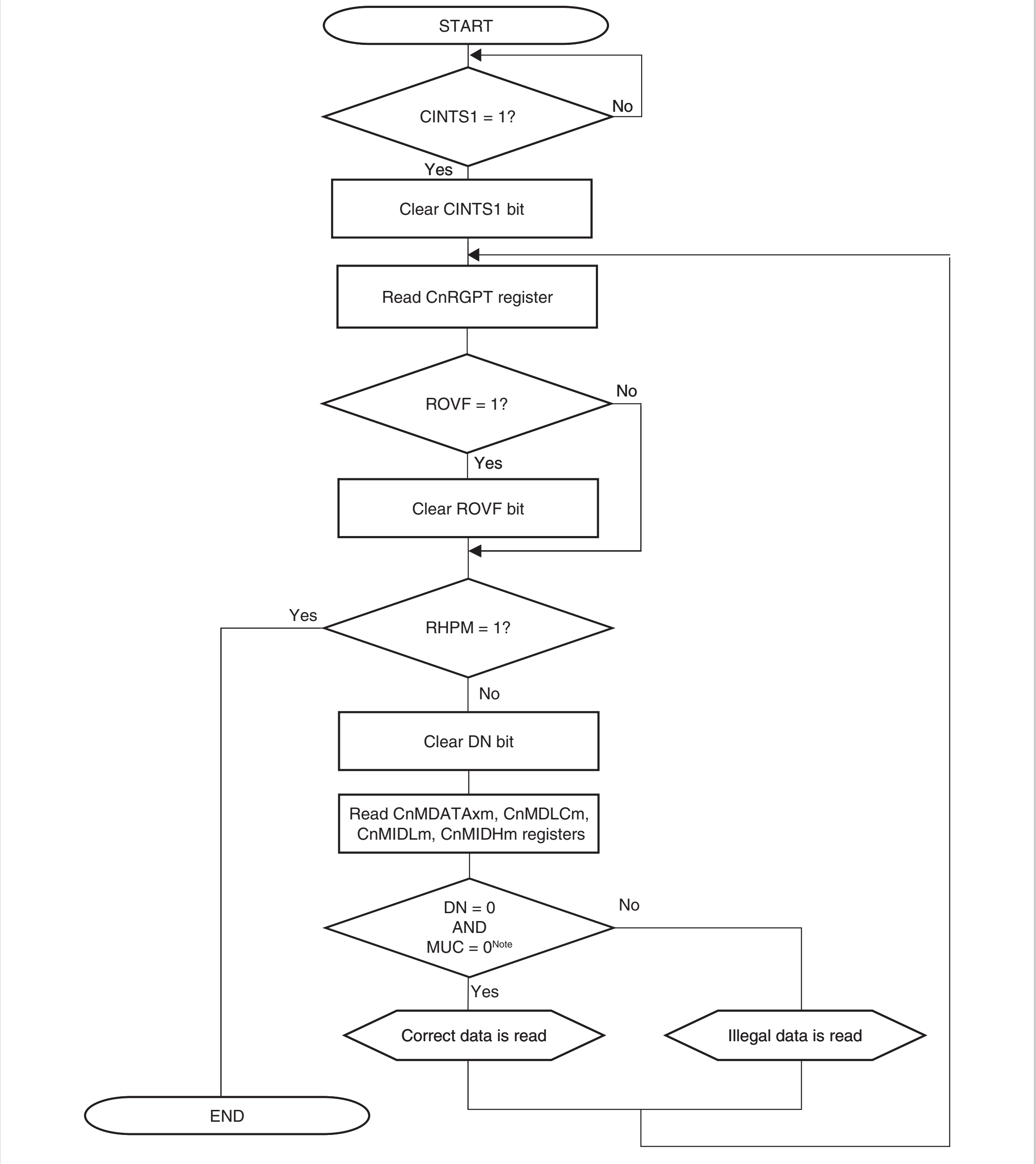

START
Generation of receive completion interrupt
Read CnRGPT register
ROVF = 1?
No
Yes
Clear ROVF bit
Yes
RHPM = 1?
No
Read CnMDATAxm, CnMDLCm, CnMIDLm, CnMIDHm registers
Clear DN bit
END

Figure 18-51 Reception via interrupt (using CnRGPT register), alternative way

Note

1. Also check the MBON flag at the beginning and at the end of the interrupt routine, in order to check the access to the message buffers as well as reception history list registers, in case a pending sleep mode had been executed. If MBON is detected to be cleared at any check, the actions and results of the processing have to be discarded and processed again, after MBON is set again.
   It is recommended to cancel any sleep mode requests, before processing RX interrupts.
2. If ROVF was set once, the receive history list is inconsistent. Consider to scan all configured receive buffers for receptions.
3. This flow will not provide most recently received data for the application. However, due to less effort on processing, it reduces interrupt load.
4. The overwrite function (CnMCONFm.OWS=1) must not be used with this flow - data inconsistency could occur.
5. It can be used alternatively to *Figure 18-50 on page 673*.



Note Check the MUC and DN bits using one read access.

Figure 18-52 Reception via software polling

Note

1. Also check the MBON flag at the beginning and at the end of the polling routine, in order to check the access to the message buffers as well as reception history list registers, in case a pending sleep mode had been executed. If MBON is detected to be cleared at any check, the actions and results of the processing have to be discarded and processed again, after MBON is set again.
2. If ROVF was set once, the receive history list is inconsistent. Consider to scan all configured receive buffers for receptions.

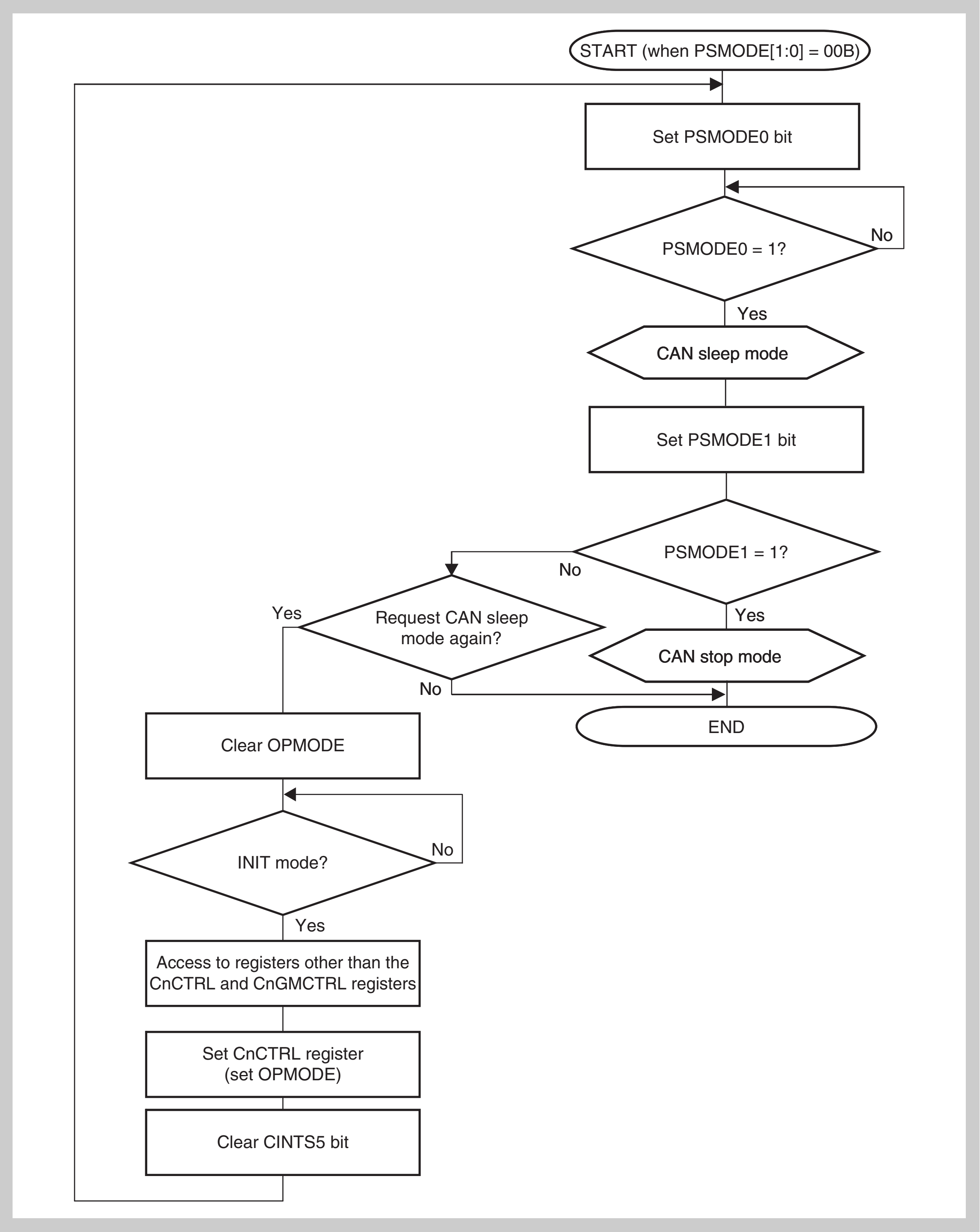

START (when PSMODE[1:0] = 00B)
Set PSMODE0 bit
PSMODE0 = 1?
No
Yes
CAN sleep mode
Set PSMODE1 bit
PSMODE1 = 1?
No
Yes
Request CAN sleep mode again?
Yes
No
CAN stop mode
END
Clear OPMODE
INIT mode?
No
Yes
Access to registers other than the CnCTRL and CnGMCTRL registers
Set CnCTRL register (set OPMODE)
Clear CINTS5 bit

Figure 18-53 Setting CAN sleep mode/stop mode

Caution To abort transmission before making a request for the CAN sleep mode, perform processing according to *Figure 18-45 on page 668* and *Figure 18-47 on page 670*.

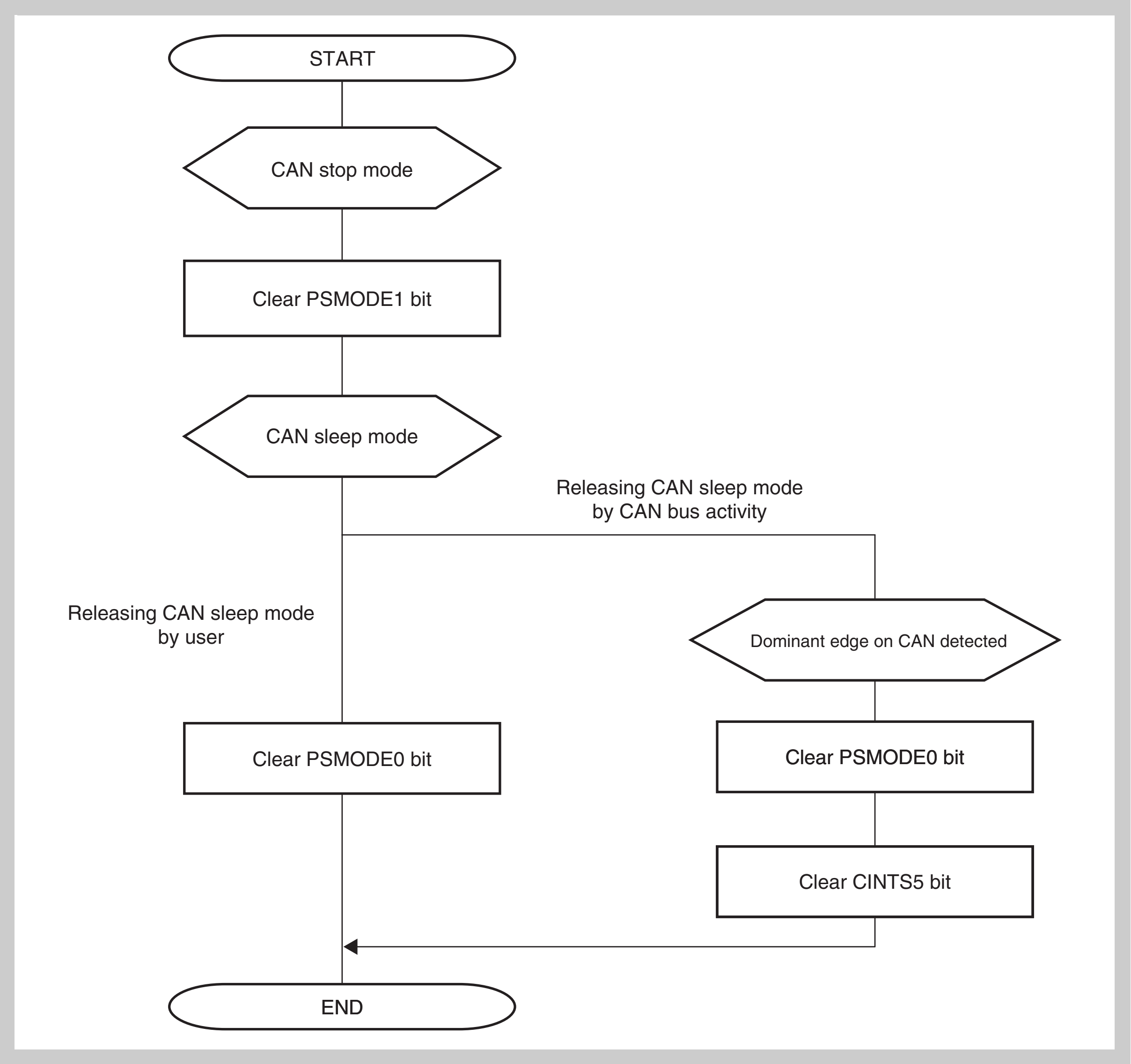
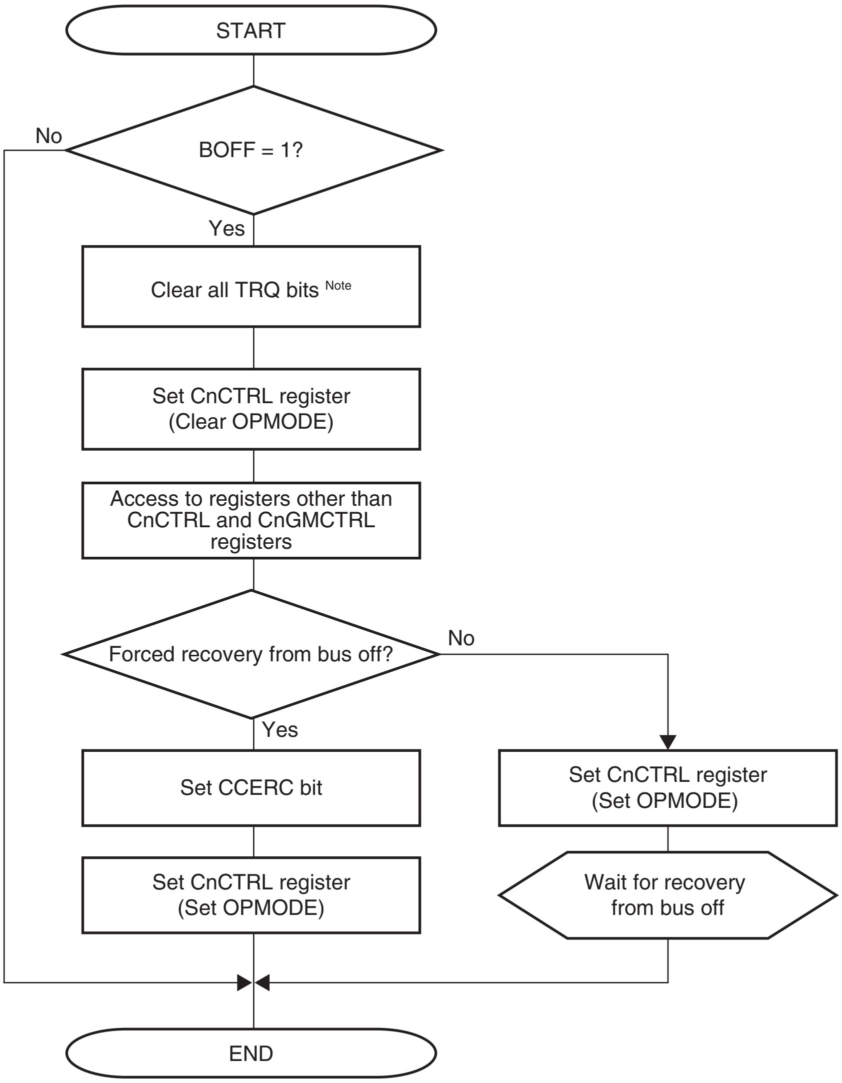

START
CAN stop mode
Clear PSMODE1 bit
CAN sleep mode
Releasing CAN sleep mode by CAN bus activity
Releasing CAN sleep mode by user
Dominant edge on CAN detected
Clear PSMODE0 bit
Clear PSMODE0 bit
Clear CINTS5 bit
END

Figure 18-54 Clear CAN sleep/stop mode

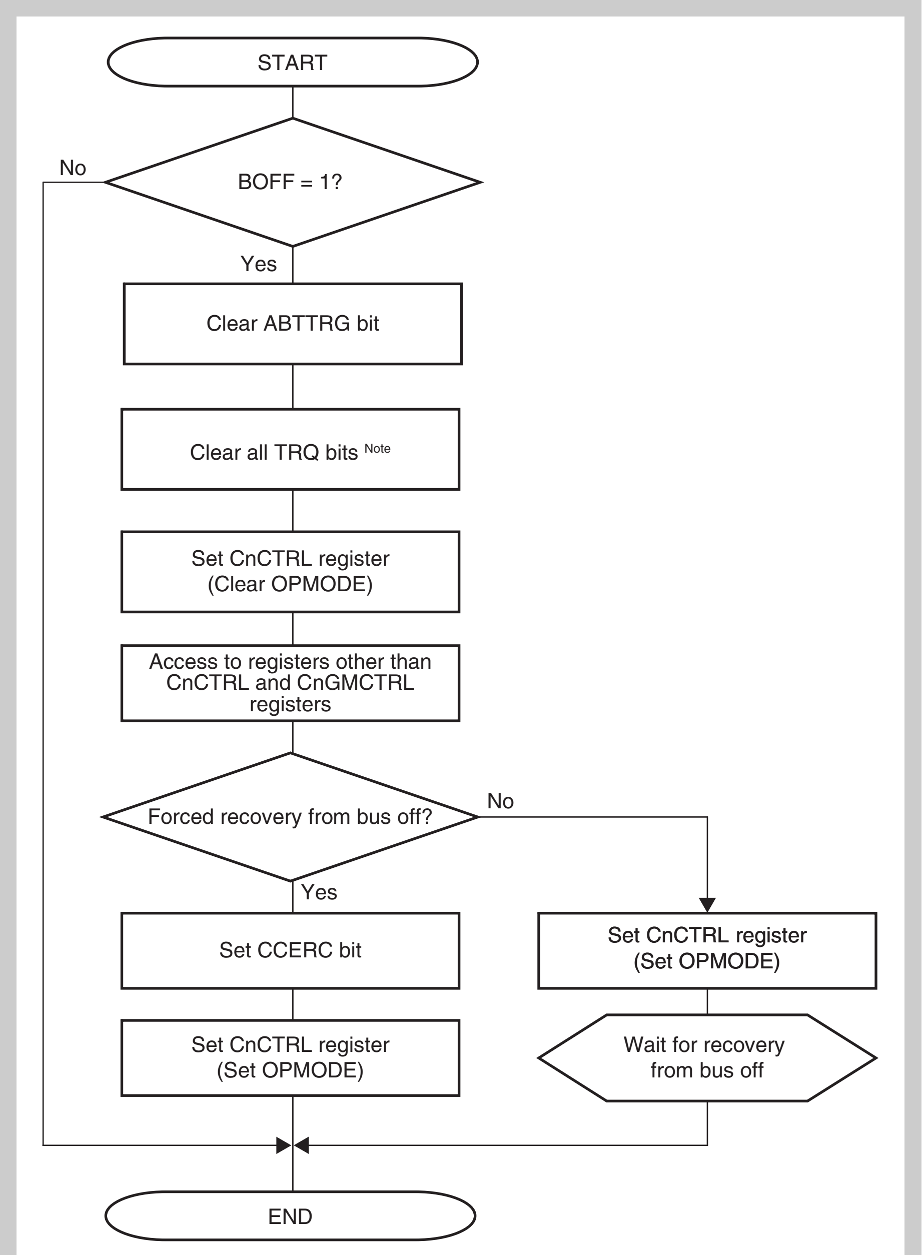


Note: Clear all TRQ bits when re-initialization of message buffer is executed by clearing RDY bit before bus-off recovery sequence is started.

OPMODE: Normal operation mode, normal operation mode with ABT, receive-only mode, single-shot mode, self-test mode

Figure 18-55 Bus-off recovery (except normal operation mode with ABT)

Caution When the transmission from the initialization mode to any operation modes is requested to execute bus-off recovery sequence again in the bus-off recovery sequence, reception error counter is cleared.
Therefore it is necessary to detect 11 consecutive recessive-level bits 128 times on the bus again.



Note: Clear all TRQ bits when re-initialization of message buffer is executed by clearing RDY bit before bus-off recovery sequence is started.

OPMODE: Normal operation mode, normal operation mode with ABT, receive-only mode, single-shot mode, self-test mode

Figure 18-56 Bus-off recovery (Normal Operation Mode with ABT)

Caution When the transmission from the initialization mode to any operation modes is requested to execute bus-off recovery sequence again in the bus-off recovery sequence, reception error counter is cleared.
Therefore it is necessary to detect 11 consecutive recessive-level bits 128 times on the bus again.

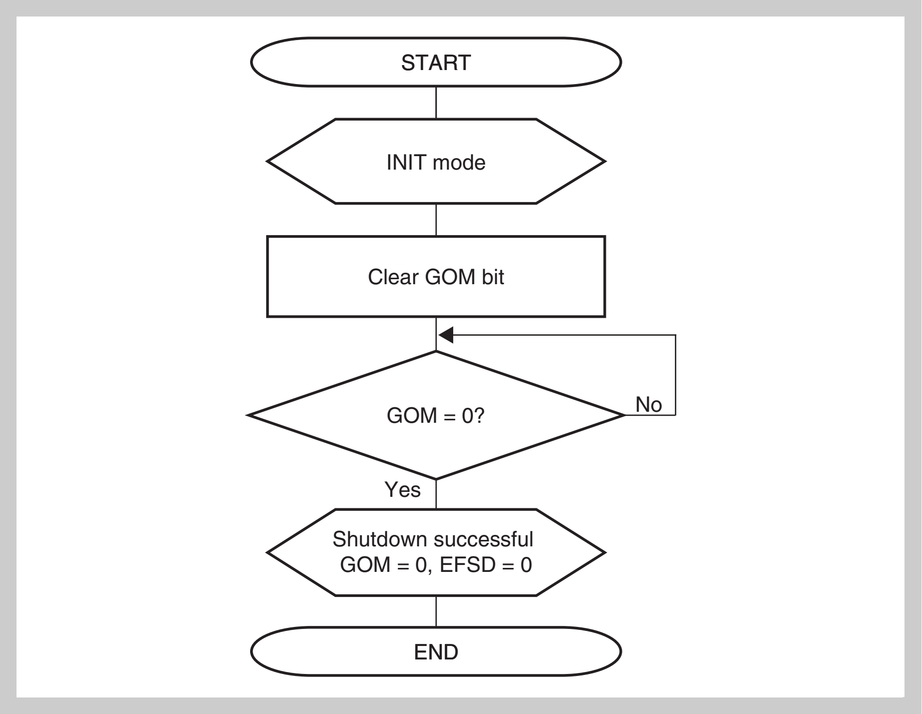


Figure 18-57 Normal shutdown process

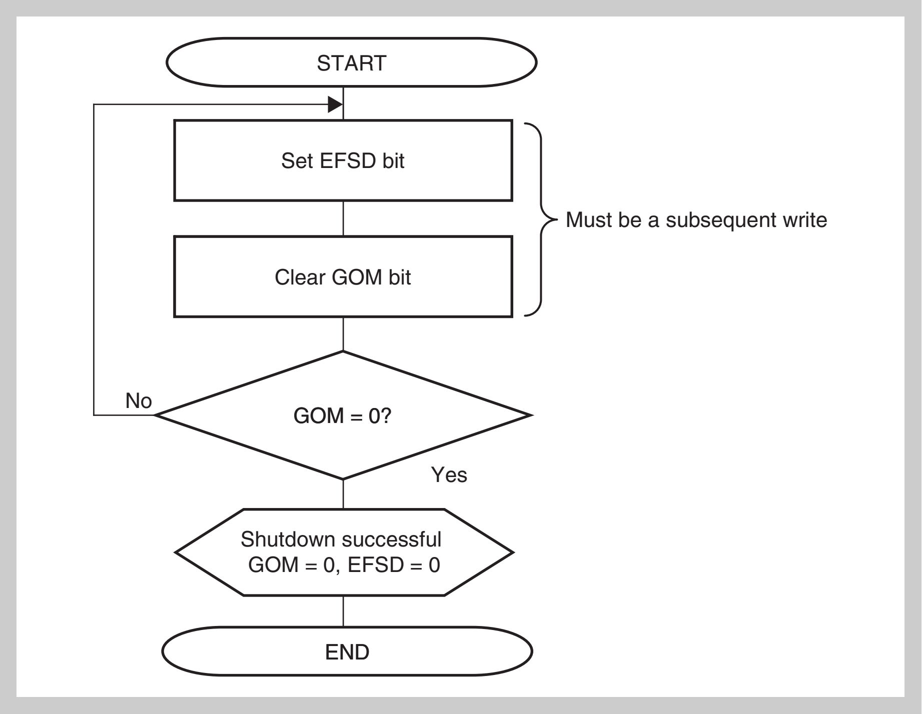


Figure 18-58 Forced shutdown process

Caution Do not read- or write-access any registers by software between setting the EFSD bit and clearing the GOM bit.

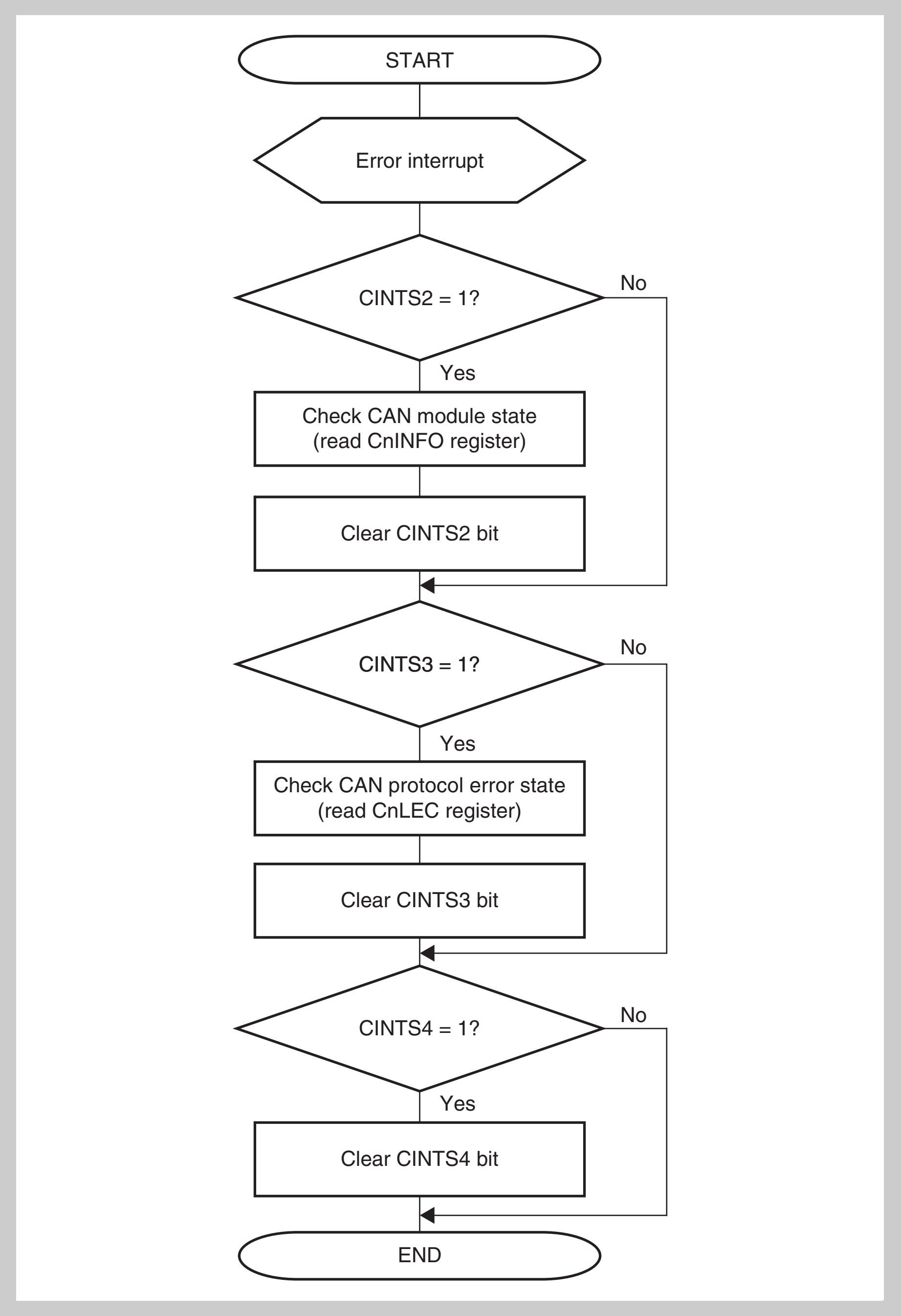


Figure 18-59 Error handling

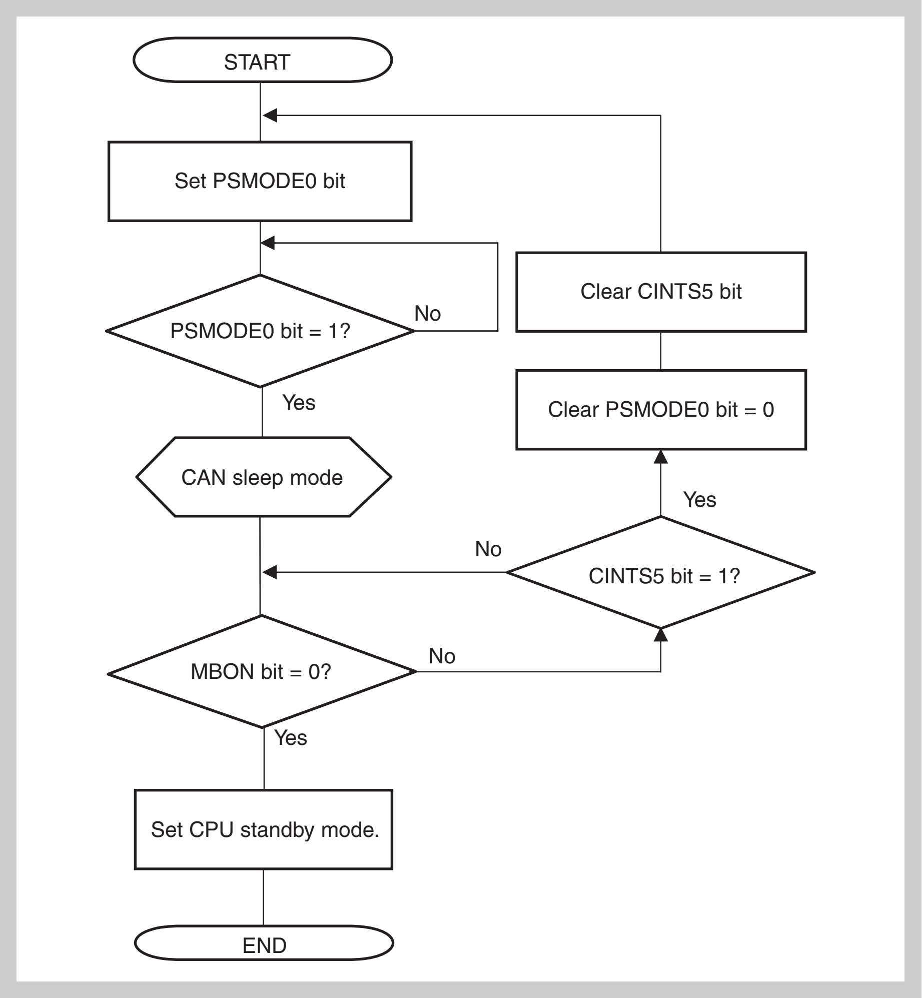


Figure 18-60 Setting CPU stand-by (from CAN sleep mode)

Caution Before the CPU is set in the CPU standby mode, please check if the CAN sleep mode has been reached.
However, after check of the CAN sleep mode, until the CPU is set in the CPU standby mode, the CAN sleep mode may be cancelled by wakeup from CAN bus.

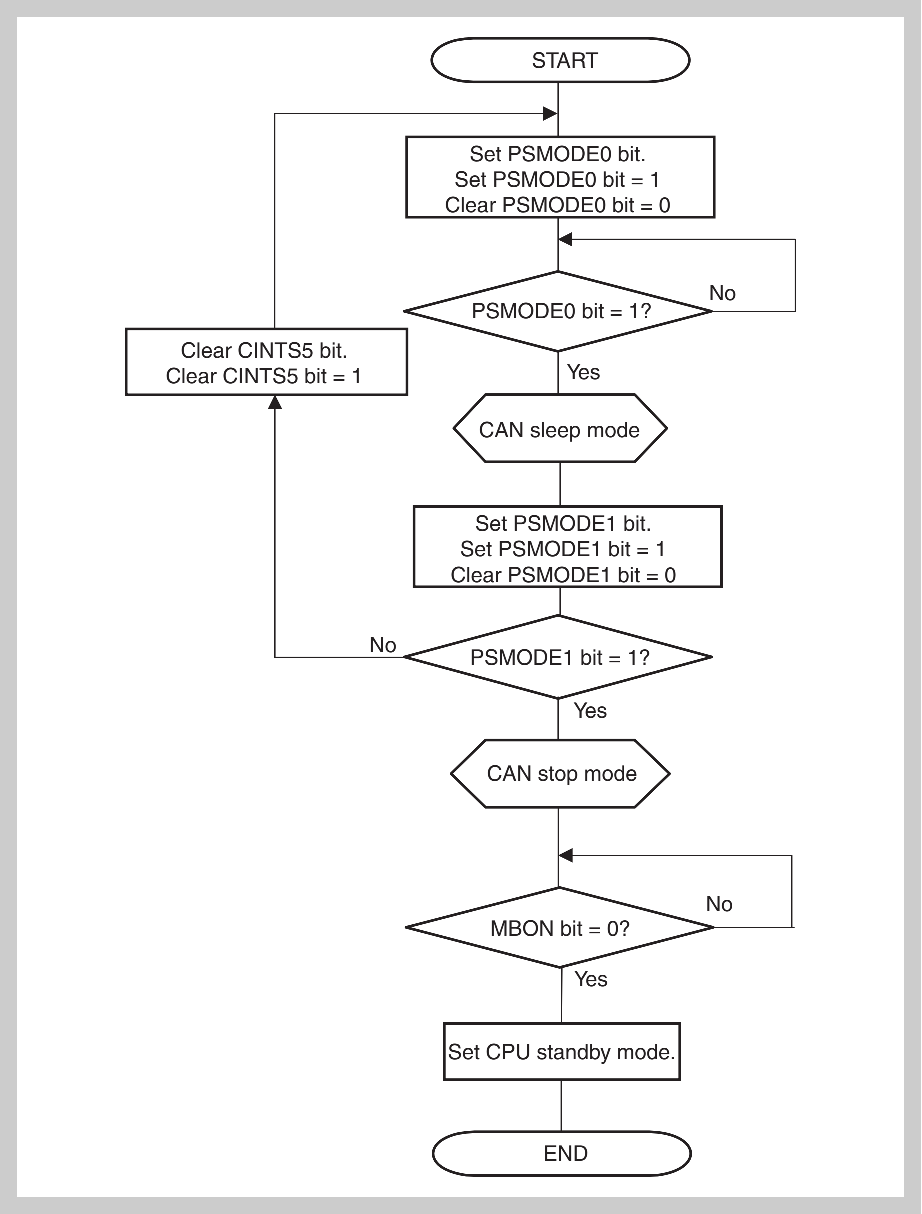


Figure 18-61 Setting CPU stand-by (from CAN stop mode)

Caution The CAN stop mode can only be released by writing 01_B to the PSMODE[1:0] bit of the CnCTRL register and not by a change in the CAN bus state.

Chapter 19 A/D Converter (ADC)

These microcontrollers contain an n-channel 10-bit A/D Converter.

The V850E/Dx3 - DG3 microcontrollers feature the following number of analog input channels:

ADC	All devices
Channels	8

Throughout this chapter, the individual channels of the A/D Converter are identified by “n”, for example ADCR0n for the A/D conversion result register of channel n.

19.1 Functions

The A/D Converter converts analog input signals into digital values.

Features summary

The A/D Converter has the following features.

- 10-bit resolution
- Successive approximation method
- The following functions are provided as operation modes.
  - Continuous select mode
  - Continuous scan mode
- The following functions are provided as trigger modes.
  - Software trigger mode
  - Timer trigger mode
- Power-fail monitor function (conversion result compare function)

The block diagram of the A/D Converter is shown below.

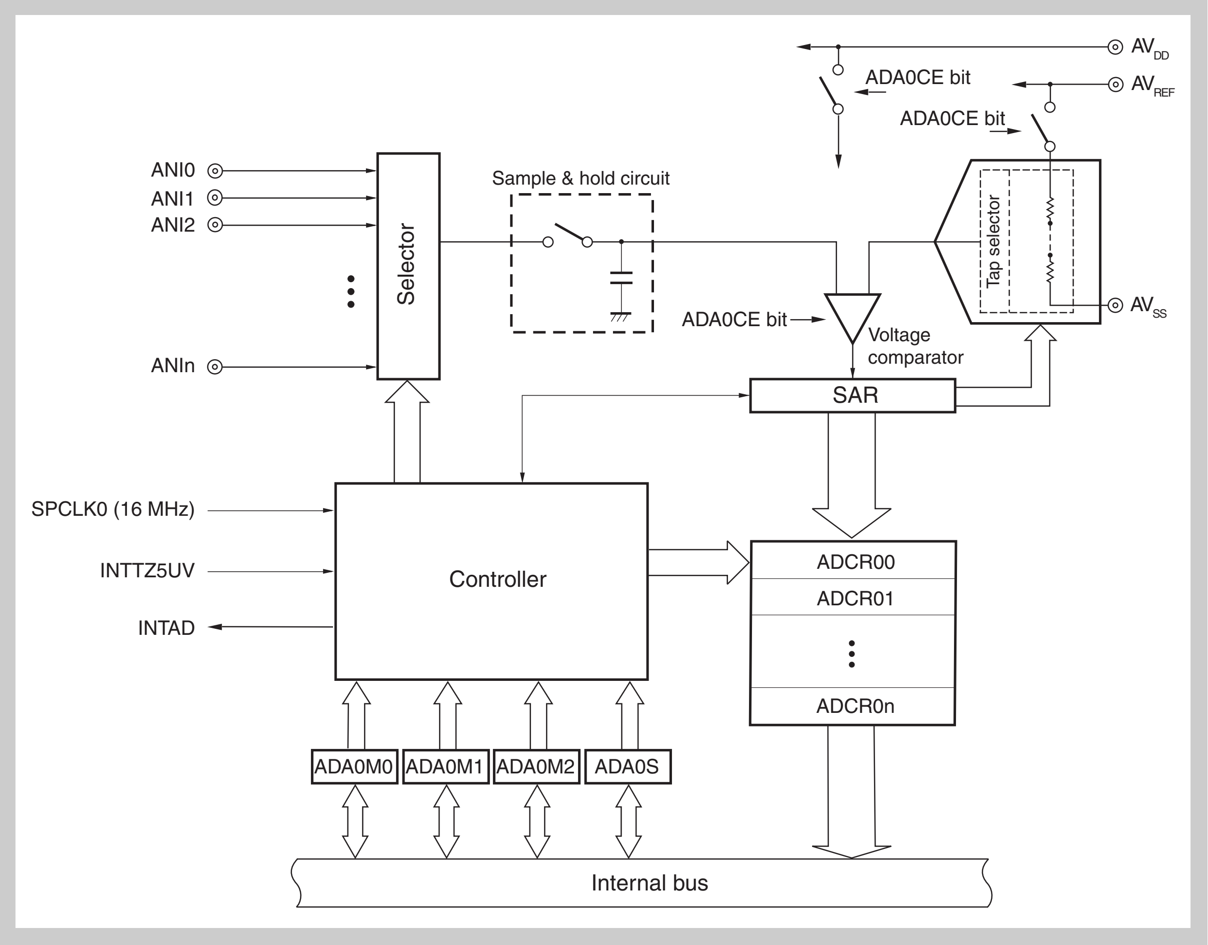


Figure 19-1 Block diagram of A/D Converter

19.2 Configuration

The A/D Converter includes the following hardware.

Table 19-1 Configuration of A/D Converter

Item	Configuration
Analog inputs	ANI0 to ANIn pins
Registers	Successive approximation register (SAR) A/D conversion result registers ADCR00 to ADCR0n A/D conversion result registers ADCR0H0 to ADCR0Hn: only higher 8 bits can be read
Control registers	A/D Converter mode registers 0 to 2 (ADA0M0 to ADA0M2) A/D Converter channel specification register 0 (ADA0S)

Caution It is mandatory to enable the A/D Converter after any reset and to perform a first conversion within a time period of maximum 1 s after reset release. With the execution of the first conversion, the A/D Converter circuit is initialized.

The execution of a first conversion is mandatory independently of whether the A/D Converter is used later on by the user application.

(1) Successive approximation register (SAR)

The SAR register compares the voltage value of the analog input signal with the voltage tap (compare voltage) value from the series resistor string, and holds the comparison result starting from the most significant bit (MSB).

When the comparison result has been held down to the least significant bit (LSB) (i.e., when A/D conversion is complete), the contents of the SAR register are transferred to the ADCR0n register.

(2) A/D conversion result register n (ADCR0n), A/D conversion result register Hn (ADCR0Hn)

The ADCR0n register is a 16-bit register that stores the A/D conversion result. ADCR0n consist of 16 registers and the A/D conversion result is stored in the 10 higher bits of the ADCR0n register corresponding to analog input. (The lower 6 bits are fixed to 0.)

The ADCR0n register is read-only, in 16-bit units.

When using only the higher 8 bits of the A/D conversion result, the ADCR0Hn register is read-only, in 8-bit units.

Caution A write operation to the ADA0M0 and ADA0S registers may cause the contents of the ADCR0n register to become undefined. After the conversion, read the conversion result before writing to the ADA0M0 and ADA0S registers. Correct conversion results may not be read if a sequence other than the above is used.

(3) Power-fail compare threshold value register (ADA0PFT)

The ADA0PFT register sets a threshold value that is compared with the value of A/D conversion result register nH (ADCR0Hn). The 8-bit data set to the ADA0PFT register is compared with the higher 8 bits of the A/D conversion result register (ADCR0Hn).

This register can be read or written in 8-bit or 1-bit units.

Reset input clears this register to 00_H.

(4) Sample & hold circuit

The sample & hold circuit samples each of the analog input signals selected by the input circuit and sends the sampled data to the voltage comparator. This circuit also holds the sampled analog input signal voltage during A/D conversion.

(5) Voltage comparator

The voltage comparator compares a voltage value that has been sampled and held with the voltage value of the series resistor string.

(6) Series resistor string

This series resistor string is connected between AV_{REF} and AV_{SS} and generates a voltage for comparison with the analog input signal.

(7) ANIn pins

These are analog input pins for the 16 A/D Converter channels and are used to input analog signals to be converted into digital signals. Pins other than the one selected as the analog input by the ADA0S register can be used as input port pins.

Caution

1. Make sure that the voltages input to the ANIn pins do not exceed the rated values. In particular if a voltage of AV_{REF} or higher is input to a channel, the conversion value of that channel becomes undefined, and the conversion values of the other channels may also be affected.
2. The analog input pins ANIn function also as input port pins. If any of ANIn is selected and A/D converted, do not execute an input instruction this ports during conversion. If executed, the conversion resolution may be degraded.

(8) AV_{REF} pin

This is the pin used to input the reference voltage of the A/D Converter. The signals input to the ANIn pins are converted to digital signals based on the voltage applied between the AV_{REF} and AV_{SS} pins.

(9) AV_{SS} pin

This is the ground pin of the A/D Converter. Always make the potential at this pin the same as that at the V_{SS} pin even when the A/D Converter is not used.

19.3 ADC Registers

The A/D Converter is controlled by the following registers:

- ADC mode registers 0, 1, 2 (ADA0M0, ADA0M1, ADA0M2)
- ADC channel specification register 0 (ADA0S)
- Power-fail compare mode register (ADA0PFM)

The following registers are also used:

- A/D conversion result register n (ADCR0n)
- A/D conversion result register nH (ADCR0Hn)
- Power-fail compare threshold value register (ADA0PFT)

(1) ADA0M0 - ADC mode register 0

The ADA0M0 register is an 8-bit register that specifies the operation mode and controls conversion operations.

Access This register can be read/written in 8-bit or 1-bit units. However, bit 0 is read-only.

Address FFFF $F200_H$

Initial Value 00_H. This register is cleared by any reset.

7	6	5	4	3	2	1	0
ADA0CE	0	ADA0MD1	ADA0MD0	0	0	ADA0TMD	ADA0EF
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R

Caution

1. If ADA0EF bit (bit 0) is written, this is ignored.
2. Changing the ADA0FR3 to ADA0FR0 bits of the ADA0M1 register during conversion (ADA0CE0 bit = 1) is prohibited.
3. When the A/D Converter is not used, stop the operation by setting the ADA0CE bit to 0 to reduce the current consumption.

(2) ADA0M1 - ADC mode register 1

The ADA0M1 register is an 8-bit register that controls the conversion time specification.

Access This register can be read/written in 8-bit or 1-bit units.

Address FFFF $F201_H$

Initial Value 00_H. This register is cleared by any reset.

7	6	5	4	3	2	1	0
1	0	0	0	ADA0FR3	ADA0FR2	ADA0FR1	ADA0FR0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Caution

1. The bit 7 must be changed to “1” after reset and must not be changed afterwards.
2. Be sure to clear bits 5 and 4 (set to 0).

Table 19-2 Conversion time settings

ADA0FR				Divider	f_{SPCLK0} = 16 MHz		f_{SPCLK0} = 4 MHz		Stabilization time[a]
3	2	1	0	div	conversion time[b]	sampling time[c]	conversion time[b]	sampling time[c]	
0	0	0	0	1	prohibited		7.75 µs	4.13 µs	16/f_{SPCLK0}
0	0	0	1	2	3.88 µs	2.06 µs	15.50 µs	8.25 µs	31/f_{SPCLK0}
0	0	1	0	3	5.81 µs	3.09 µs	prohibited		47/f_{SPCLK0}
0	0	1	1	4	7.75 µs	4.13 µs	prohibited		50/f_{SPCLK0}
0	1	0	0	5	9.69 µs	5.16 µs	prohibited		50/f_{SPCLK0}
0	1	0	1	6	11.63 µs	6.12 µs	prohibited		50/f_{SPCLK0}
0	1	1	0	7	13.56 µs	7.22 µs	prohibited		50/f_{SPCLK0}
0	1	1	1	8	15.50 µs	8.25 µs	prohibited		50/f_{SPCLK0}
1	x	x	x	prohibited					

a) **When A/D conversion is started by ADA0M0.ADA0CE = 0 → 1 the first sampling of the ANIn input is delayed by the given stabilization time. This ensures compliance with the necessary stabilization time. The stabilization time applies only prior to the first sampling.**

b) **The conversion time is calculated by (31 x div) / f_{SPCLK0}.**

c) **The sampling time is calculated by (16.5 x div) / f_{SPCLK0}.**

Note Note that the given times in *Table 19-2* do not regard the dithering of the A/D converter supply clock. Using a dithering supply clock does not impact the A/D converter’s operation.

(3) ADA0M2 - ADC mode register 2

The ADA0M2 register specifies the hardware trigger mode.

Access This register can be read/written in 8-bit or 1-bit units.

Address FFFF $F203_H$

Initial Value 00_H. This register is cleared by any reset.

7	6	5	4	3	2	1	0
0	0	0	0	0	0	ADA0TMD1	ADA0TMD0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Caution Be sure to clear bits 7 to 1.

(4) ADA0S - ADC channel specification register 0

The ADA0S register specifies the pin that inputs the analog voltage to be converted into a digital signal.

Access This register can be read/written in 8-bit or 1-bit units.

Address FFFF $F202_H$

Initial Value 00_H. This register is cleared by any reset.

7	6	5	4	3	2	1	0
0	0	0	0	ADA0S3	ADA0S2	ADA0S1	ADA0S0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 19-3 ADA0S register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>4 to 0</td><td>ADA0S[4:0]</td><td>A/D converter channel specification:
<table>
<tr><th>ADA0S3</th><th>ADA0S2</th><th>ADA0S1</th><th>ADA0S0</th><th>Select mode</th><th>Scan mode</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>ANI0</td><td>ANI0</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>ANI1</td><td>ANI0, ANI1</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>ANI2</td><td>ANI0 to ANI2</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>ANI3</td><td>ANI0 to ANI3</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td><td>ANI4</td><td>ANI0 to ANI4</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1</td><td>ANI5</td><td>ANI0 to ANI5</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td><td>ANI6</td><td>ANI0 to ANI6</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>1</td><td>ANI7</td><td>ANI0 to ANI7</td></tr>
<tr><td colspan="4">Other than above</td><td colspan="2">Setting prohibited</td></tr>
</table>
</td></tr>
</table>

(5) ADCR0n, ADCR0Hn - ADC conversion result registers

The ADCR0n and ADCR0Hn registers store the A/D conversion results.

Access These registers are read-only, in 16-bit or 8-bit units. However, specify the ADCR0n register for 16-bit access and the ADCR0Hn register for 8-bit access. The 10 bits of the conversion result are read from the higher 10 bits of the ADCR0n register, and 0 is read from the lower 6 bits. The higher 8 bits of the conversion result are read from the ADCR0Hn register.

Address

ADCR00:	FFFF F210$_H$	ADCR0H0:	FFFF F211$_H$
ADCR01:	FFFF F212$_H$	ADCR0H1:	FFFF F213$_H$
ADCR02:	FFFF F214$_H$	ADCR0H2:	FFFF F215$_H$
ADCR03:	FFFF F216$_H$	ADCR0H3:	FFFF F217$_H$
ADCR04:	FFFF F218$_H$	ADCR0H4:	FFFF F219$_H$
ADCR05:	FFFF F21A$_H$	ADCR0H5:	FFFF F21B$_H$
ADCR06:	FFFF F21C$_H$	ADCR0H6:	FFFF F21D$_H$
ADCR07:	FFFF F21E$_H$	ADCR0H7:	FFFF F21F$_H$

Initial Value undefined

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADCR0n	AD9	AD8	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0	0	0	0	0	0	0
	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

	7	6	5	4	3	2	1	0
ADCR0Hn	AD9	AD8	AD7	AD6	AD5	AD4	AD3	AD2
	R	R	R	R	R	R	R	R

The relationship between the analog voltage input to the analog input pins (ANI0 to ANI11) and the A/D conversion result (of A/D conversion result register n (ADCR0n)) is as follows:

$$\mathrm{ADCR0} = \mathrm{INT}(\frac{V_{IN}}{AV_{REF}} \bullet 1024 + 0{,}5)$$

or

$$(\mathrm{ADCR0} - 0{,}5) \bullet \frac{AV_{REF}}{1024} \le V_{IN} < (\mathrm{ADCR0} + 0{,}5) \bullet \frac{AV_{REF}}{1024}$$

INT( ): Function that returns the integer of the value in ( )

V_{IN}: Analog input voltage

AV_{REF}: AV_{REF} pin voltage

ADCR0: Value of A/D conversion result register n (ADCR0n)

Figure 19-2 shows the relationship between the analog input voltage and the A/D conversion results.

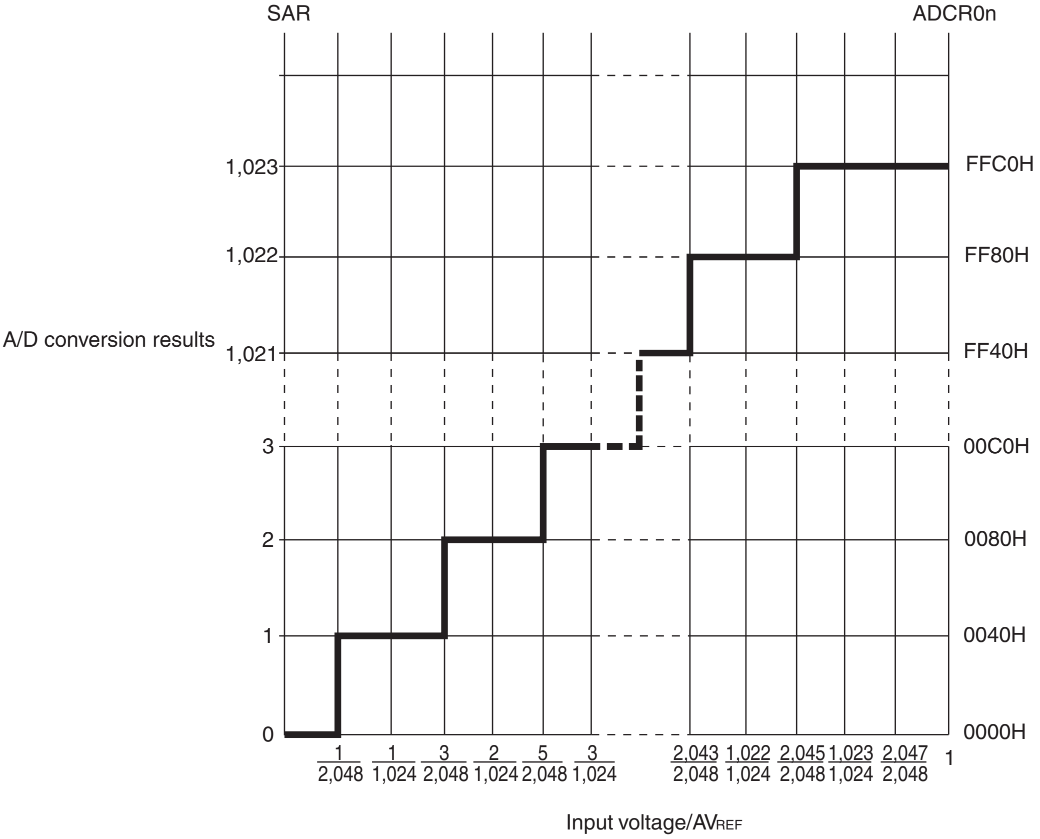


Figure 19-2 Relationship between analog input voltage and A/D conversion results

(6) ADA0PFM - ADC power-fail compare mode register

The ADA0PFM register is an 8-bit register that sets the power-fail compare mode.

Access This register can be read/written in 8-bit or 1-bit units.

Address FFFF F204$_H$

Initial Value 00$_H$. This register is cleared by any reset.

7	6	5	4	3	2	1	0
ADA0PFE	ADA0PFC0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 19-4 ADA0PFM register contents

Bit position	Bit name	Function
7	ADA0PFE	Power-fail compare enable/disable: 0: Power-fail compare disabled 1: Power-fail compare enabled
6	ADA0PFC0	Power-fail compare mode: 0: Generates interrupt request INTAD if ADA0CRnH ≥ ADA0PFT 1: Generates interrupt request INTAD if ADA0CRnH < ADA0PFT

Note In continuous select mode the conversion result of ADC channel ANIn, selected by ADA0S, is observed.
In continuous scan mode the conversion result of ADC channel ANI0 is observed.
For further details, refer to *“Power-fail compare mode“ on page 701*.

(7) ADA0PFT - ADC power-fail compare threshold value register

The ADA0PFT register sets the compare value in the power-fail compare mode.

Access This register can be read/written in 8-bit or 1-bit units.

Address FFFF F204$_H$

Initial Value 00$_H$. This register is cleared by any reset.

7	6	5	4	3	2	1	0
ADA0PFT7	ADA0PFT6	ADA0PFT5	ADA0PFT4	ADA0PFT3	ADA0PFT2	ADA0PFT1	ADA0PFT0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 19-5 ADA0PFT register contents

Bit position	Bit name	Function
7 to 0	ADA0PFT[7:0]	Compare value in power-fail compare mode.

19.4 Operation

19.4.1 Basic operation

1. Set the operation mode, trigger mode, and conversion time for executing A/D conversion by using the ADA0M0, ADA0M1, ADA0M2, and ADA0S registers. When the ADA0CE bit of the ADA0M0 register is set, conversion is started in the software trigger mode and the A/D Converter waits for a trigger in the external or timer trigger mode.
2. When A/D conversion is started, the voltage input to the selected analog input channel is sampled by the sample & hold circuit.
3. When the sample & hold circuit samples the input channel for a specific time, it enters the hold status, and holds the input analog voltage until A/D conversion is complete.
4. Set bit 9 of the successive approximation register (SAR). The tap selector selects (1/2) AV_{REF} as the voltage tap of the series resistor string.
5. The voltage difference between the voltage of the series resistor string and the analog input voltage is compared by the voltage comparator. If the analog input voltage is higher than (1/2) AV_{REF}, the MSB of the SAR register remains set. If it is lower than (1/2) AV_{REF}, the MSB is reset.
6. Next, bit 8 of the SAR register is automatically set and the next comparison is started. Depending on the value of bit 9, to which a result has been already set, the voltage tap of the series resistor string is selected as follows:

   –Bit 9 = 1: (3/4) AV_{REF}

   –Bit 9 = 0: (1/4) AV_{REF}

   This voltage tap and the analog input voltage are compared and, depending on the result, bit 8 is manipulated as follows.

   Analog input voltage ≥ Voltage tap: Bit 8 = 1

   Analog input voltage ≤ Voltage tap: Bit 8 = 0
7. This comparison is continued to bit 0 of the SAR register.
8. When comparison of the 10 bits is complete, the valid digital result is stored in the SAR register, which is then transferred to and stored in the ADCR0n register. At the same time, an A/D conversion end interrupt request signal (INTAD) is generated.

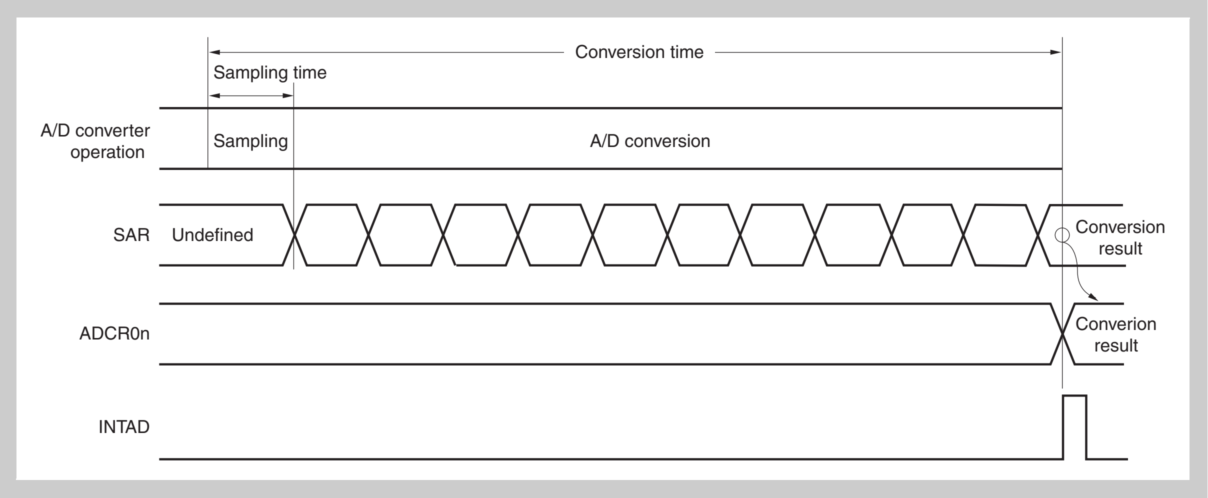


Figure 19-3 A/D Converter basic operation

19.4.2 Trigger mode

The timing of starting the conversion operation is specified by setting a trigger mode. The trigger mode includes a software trigger mode and hardware trigger modes. The hardware trigger modes include timer trigger modes 0 and 1, and external trigger mode. The ADA0TMD bit of the ADA0M0 register is used to set the trigger mode. In timer trigger mode set ADA0M2.ADA0TMD[1:0] = 01.

(1) Software trigger mode

When the ADA0CE bit of the ADA0M0 register is set to 1, the signal of the analog input pin ANIn specified by the ADA0S register is converted. When conversion is complete, the result is stored in the ADCR0n register. At the same time, the A/D conversion end interrupt request signal (INTAD) is generated.

If the operation mode specified by the ADA0MD1 and ADA0MD0 bits of the ADA0M0 register is the continuous select/scan mode, the next conversion is started, unless the ADA0CE bit is cleared to 0 after completion of the first conversion.

When conversion is started, the ADA0EF bit is set to 1 (indicating that conversion is in progress).

If the ADA0M0, ADA0M2, ADA0S, ADA0PFM, or ADA0PFT register is written during conversion, the conversion is aborted and started again from the beginning.

(2) Timer trigger mode

In this mode, converting the signal of the analog input pin ANIn specified by the ADA0S register is started by the Timer Z underflow interrupt signal.

Make sure to set ADA0M2.ADA0TMD[1:0] = 01_B.

When conversion is completed, the result of the conversion is stored in the ADCR0n register. At the same time, the A/D conversion end interrupt request signal (INTAD) is generated, and the A/D Converter waits for the trigger again.

When conversion is started, the ADA0EF bit is set to 1 (indicating that conversion is in progress). While the A/D Converter is waiting for the trigger, however, the ADA0EF bit is cleared to 0 (indicating that conversion is stopped). If the valid trigger is input during the conversion operation, the conversion is aborted and started again from the beginning.

If the ADA0M0, ADA0M2, ADA0S, ADA0PFM, or ADA0PFT register is written during conversion, the conversion is stopped and the A/D Converter waits for the trigger again.

19.4.3 Operation modes

Two operation modes are available as the modes in which to set the ANIn pins: continuous select mode and continuous scan mode.

The operation mode is selected by the ADA0MD1 and ADA0MD0 bits of the ADA0M0 register.

(1) Continuous select mode

In this mode, the voltage of one analog input pin selected by the ADA0S register is continuously converted into a digital value.

The conversion result is stored in the ADCR0n register corresponding to the analog input pin. In this mode, an analog input pin corresponds to an ADCR0n register on a one-to-one basis. Each time A/D conversion is completed, the A/D conversion end interrupt request signal (INTAD) is generated. After completion of conversion, the next conversion is started, unless the ADA0CE bit of the ADA0M0 register is cleared to 0.

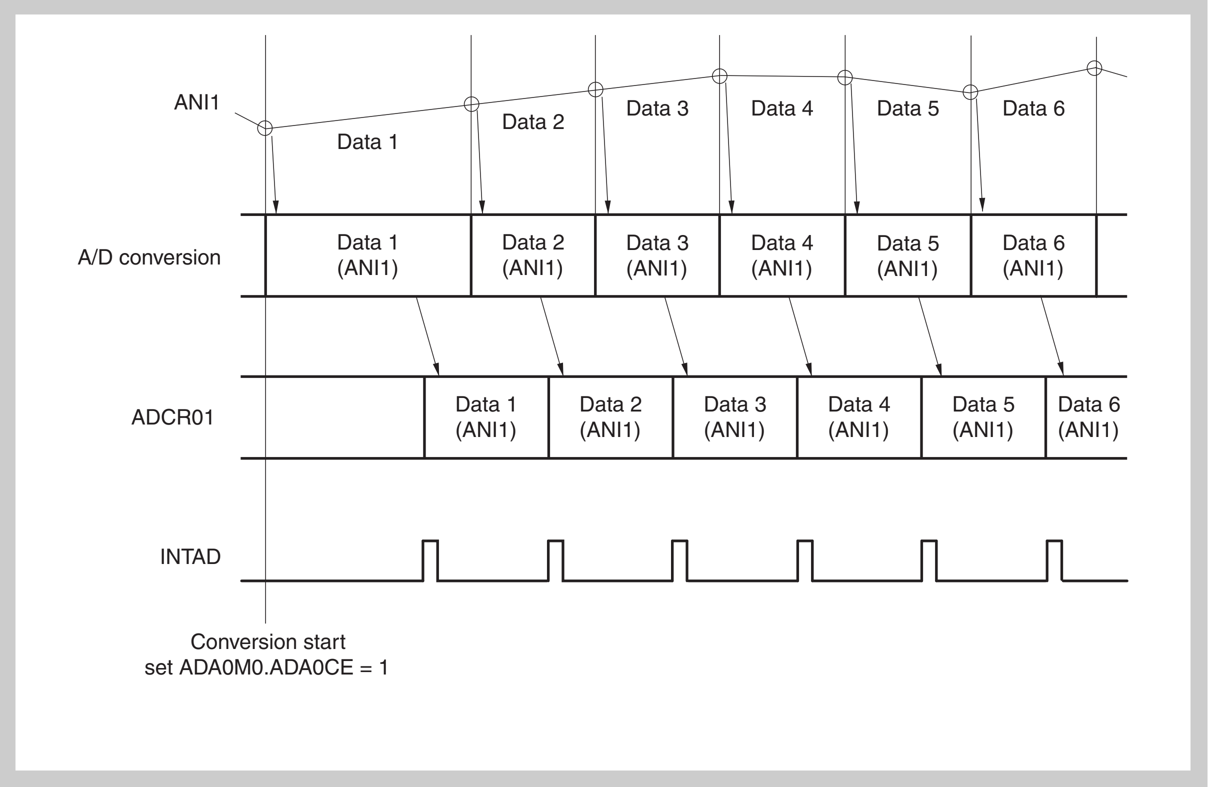


Figure 19-4 Timing example of continuous select mode operation (ADA0S = 01_H)

(2) Continuous scan mode

In this mode, analog input pins are sequentially selected, from the ANI0 pin to the pin specified by the ADA0S register, and their values are converted into digital values.

The result of each conversion is stored in the ADCR0n register corresponding to the analog input pin. When conversion of the analog input pin specified by the ADA0S register is complete, the A/D conversion end interrupt request signal (INTAD) is generated, and A/D conversion is started again from the ANI0 pin, unless the ADA0CE bit of the ADA0M0 register is cleared to 0.

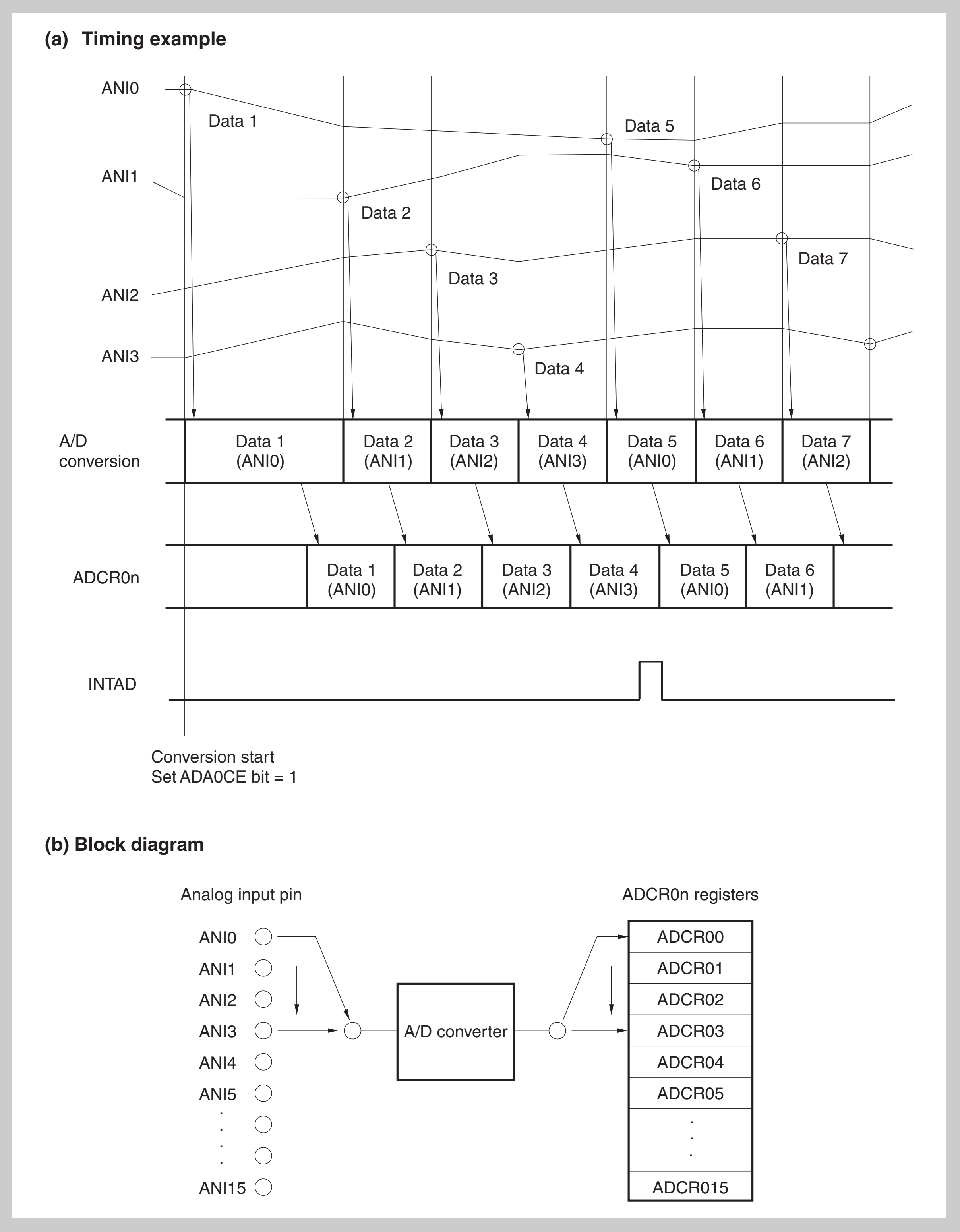


Figure 19-5 Timing example of continuous scan mode operation (ADA0S register = 03_H)

19.4.4 Power-fail compare mode

The A/D conversion end interrupt request signal (INTAD) can be controlled as follows by the ADA0PFM and ADA0PFT registers.

- If the power-fail compare mode is disabled (ADA0PFM.ADA0PFE = 0), the INTAD signal is generated each time conversion is completed.
- If the power-fail compare mode is enabled (ADA0PFM.ADA0PFE = 1) and ADA0PFM.ADA0PFC = 0, the value of the ADCR0Hn register is compared with the value of the ADA0PFT register when conversion is completed, and the INTAD signal is generated only if ADCR0H0 ≥ ADA0PFT.
- If the power-fail compare mode is enabled (ADA0PFM.ADA0PFE = 1) and ADA0PFM.ADA0PFC = 1, the value of the ADCR0Hn register is compared with the value of the ADA0PFT register when conversion is completed, and the INTAD signal is generated only if ADCR0H0 < ADA0PFT.

In the power-fail compare mode, two modes are available as modes in which to set the ANIn pins: continuous select mode and continuous scan mode.

(1) Continuous select mode

In this mode, the higher 8 bits of conversion result of the ANIn channel in ADA0CR0Hn, specified by ADA0S, is compared with the value of the ADA0PFT register.

If the result of power-fail comparison matches the condition set by the ADA0PFM.ADA0PFC bit, INTAD is generated.

In any case the next conversion is started.

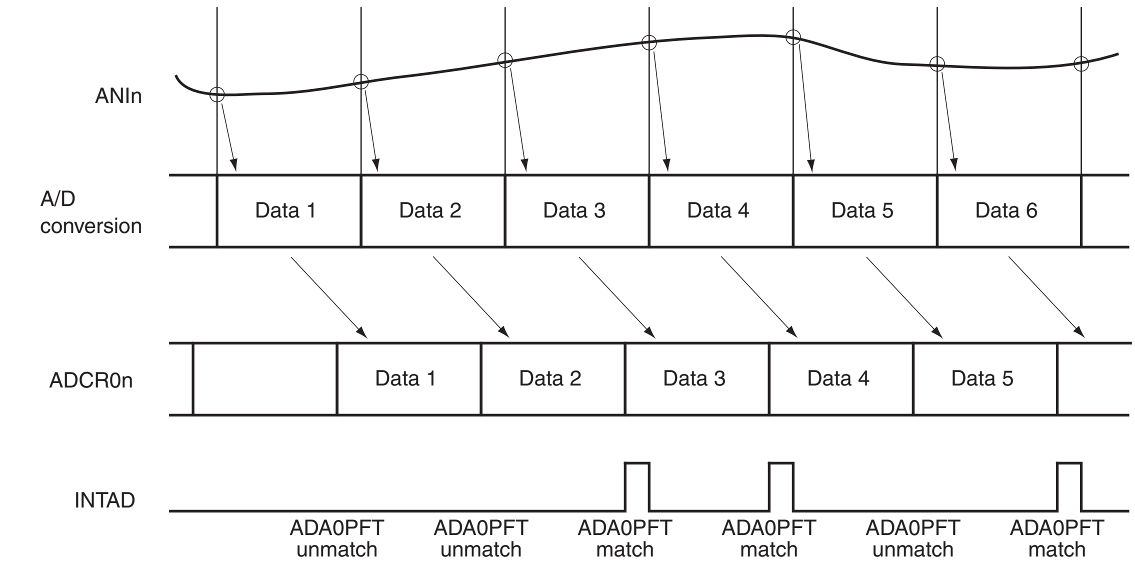


Figure 19-6 Timing example of continuous select mode operation whit power-fail comparison

(2) Continuous scan mode

In this mode, the ADC channels starting from ANI0 to the one specified by the ADA0S register are sequentially converted and the conversion results are stored in the ADCR0n registers.

Note In continuous scan mode power-fail comparison is performed only on ANI0.

After each conversion of ANI0, the higher 8 bits of conversion result in ADA0CR0H0 is compared with the value of the ADA0PFT register.

If the result of power-fail comparison matches the condition set by the ADA0PFM.ADA0PFC bit, INTAD is generated.

In any case conversion of the remaining ADC channels continuous.

Thus it is possible to catch a snapshot of the other analog inputs ANIn in case of power-fail.

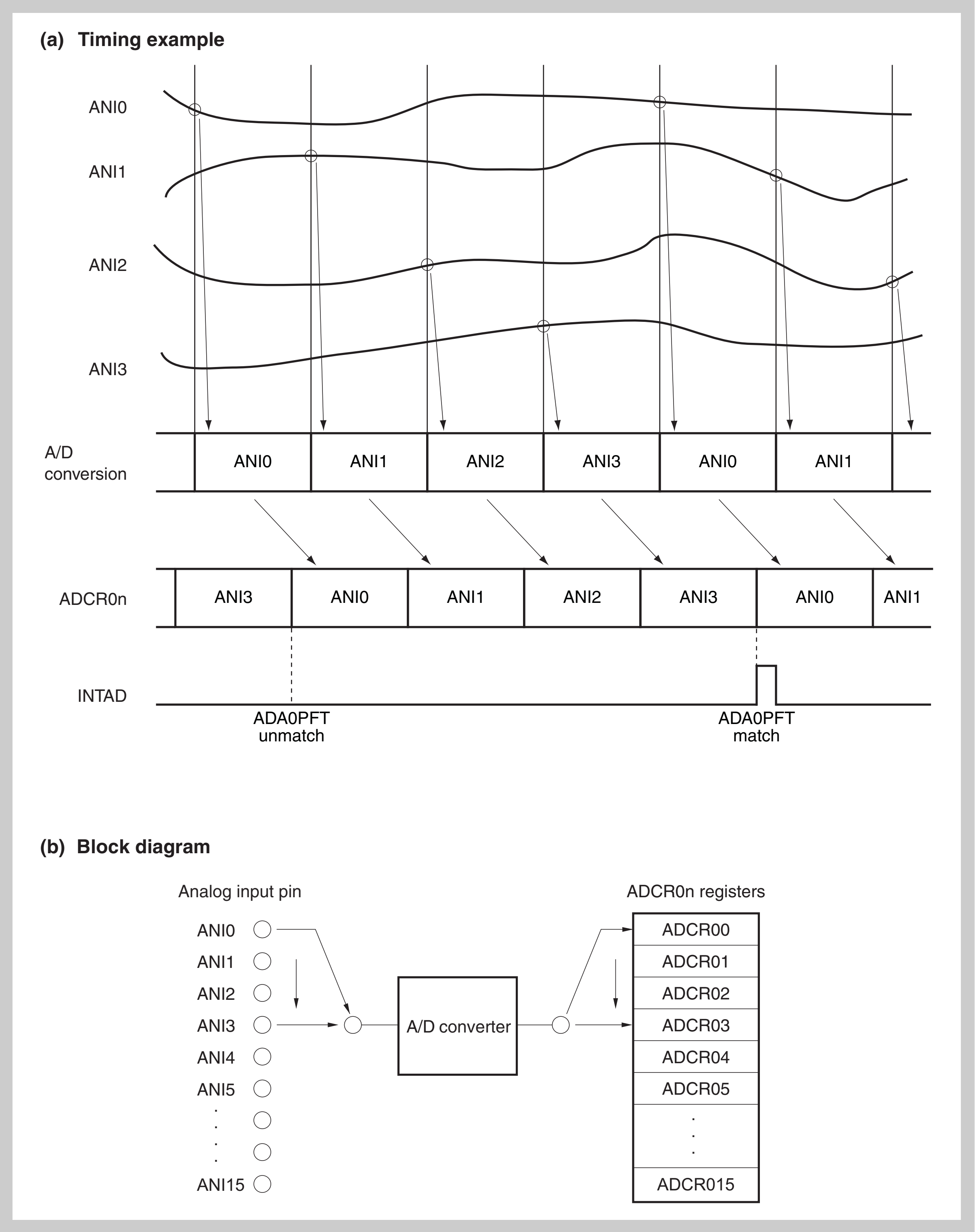


Figure 19-7 Timing example of continuous scan mode operation with power-fail comparison (ADA0S = 03_H)

19.5 Cautions

(1) When A/D Converter is not used

When the A/D Converter is not used, the power consumption can be reduced by clearing the ADA0CE bit of the ADA0M0 register to 0.

(2) Input range of ANIn pins

Input the voltage within the specified range to the ANIn pins. If a voltage equal to or higher than AV_{REF} or equal to or lower than AV_{SS} (even within the range of the absolute maximum ratings) is input to any of these pins, the conversion value of that channel is undefined.

(3) Countermeasures against noise

To maintain the 10-bit resolution, the ANIn pins must be effectively protected from noise. The influence of noise increases as the output impedance of the analog input source becomes higher. To lower the noise, connecting an external capacitor as shown in *Figure 19-8* is recommended.

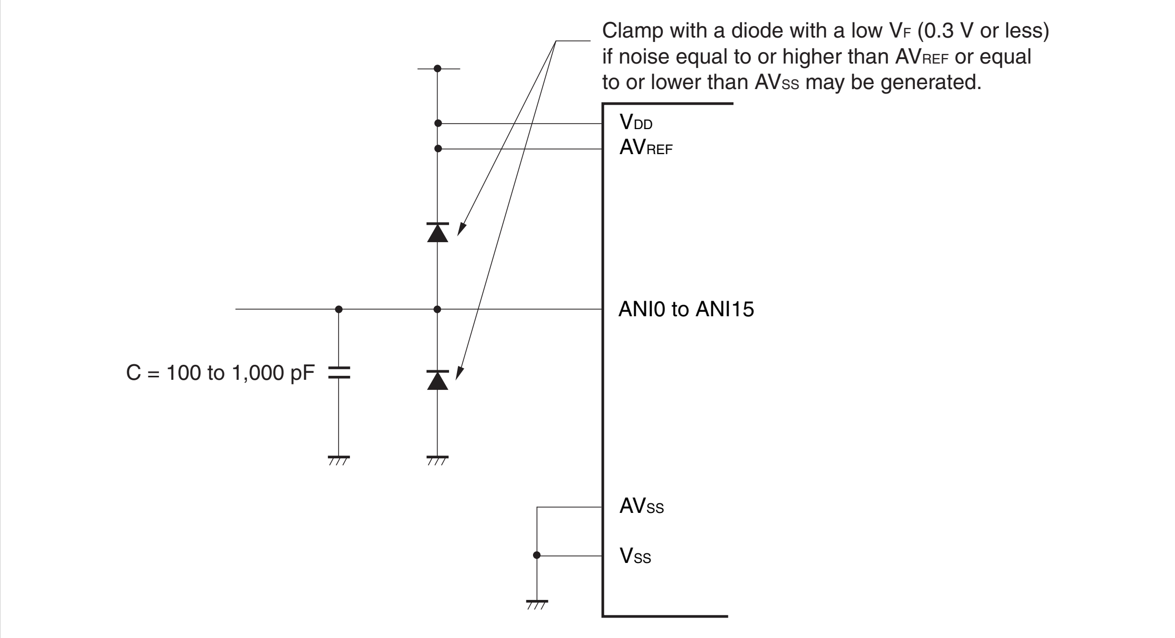

Clamp with a diode with a low V_F (0.3 V or less) if noise equal to or higher than AV_{REF} or equal to or lower than AV_{SS} may be generated.
V_{DD}
AV_{REF}
ANI0 to ANI15
C = 100 to 1,000 pF
AV_{SS}
V_{SS}

Figure 19-8 Processing of analog input pin

(4) Alternate I/O

The analog input pins ANIn function alternately as port pins. When selecting one of the ANIn pins to execute A/D conversion, do not execute an instruction to read an input port or write to an output port during conversion as the conversion resolution may drop.

If a digital pulse is applied to a pin adjacent to the pin whose input signal is being converted, the A/D conversion value may not be as expected due to the influence of coupling noise. Therefore, do not apply a pulse to a pin adjacent to the pin undergoing A/D conversion.

(5) Interrupt request flag (ADIF)

The interrupt request flag (ADIF) is not cleared even if the contents of the ADA0S register are changed. If the analog input pin is changed during A/D conversion, therefore, the result of converting the previously selected analog input signal may be stored and the conversion end interrupt request flag may be set immediately before the ADA0S register is rewritten. If the ADIF flag is read immediately after the ADA0S register is rewritten, the ADIF flag may be set even though the A/D conversion of the newly selected analog input pin has not been completed. When A/D conversion is stopped, clear the ADIF flag before resuming conversion.

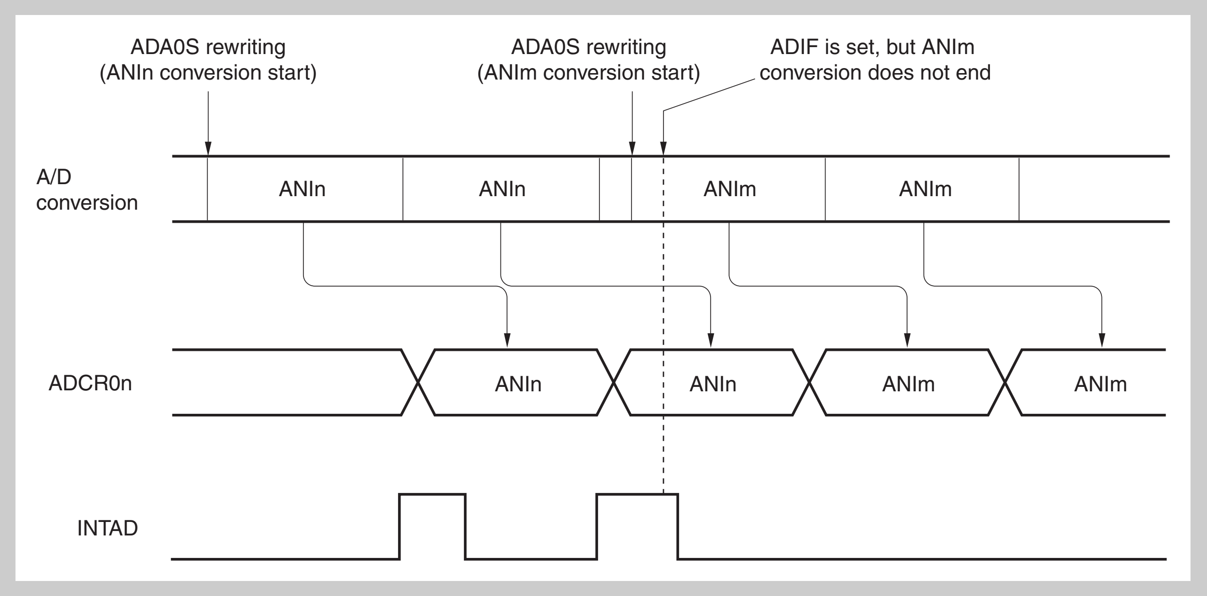


Figure 19-9 Generation timing of A/D conversion end interrupt request

(6) Reading ADCR0n register

When the ADA0M0 to ADA0M2 or ADA0S register is written, the contents of the ADCR0n register may be undefined. Read the conversion result after completion of conversion and before writing to the ADA0M0 to ADA0M2 and ADA0S registers. The correct conversion result may not be read at a timing different from the above.

19.6 How to Read A/D Converter Characteristics Table

This section describes the terms related to the A/D Converter.

(1) Resolution

The minimum analog input voltage that can be recognized, i.e., the ratio of an analog input voltage to 1 bit of digital output is called 1 LSB (least significant bit). The ratio of 1 LSB to the full scale is expressed as %FSR (full-scale range). %FSR is the ratio of a range of convertible analog input voltages expressed as a percentage, and can be expressed as follows, independently of the resolution.

1%FSR = (Maximum value of convertible analog input voltage – Minimum value of convertible analog input voltage)/100
= $(AV_{REF} - 0)/100$
= $AV_{REF}/100$

When the resolution is 10 bits, 1 LSB is as follows:

1 LSB = $1/2^{10} = 1/1{,}024$
= 0.098%FSR

The accuracy is determined by the overall error, independently of the resolution.

(2) Overall error

This is the maximum value of the difference between an actually measured value and a theoretical value. It is a total of zero-scale error, full-scale error, linearity error, and a combination of these errors.

The overall error in the characteristics table does not include the quantization error.

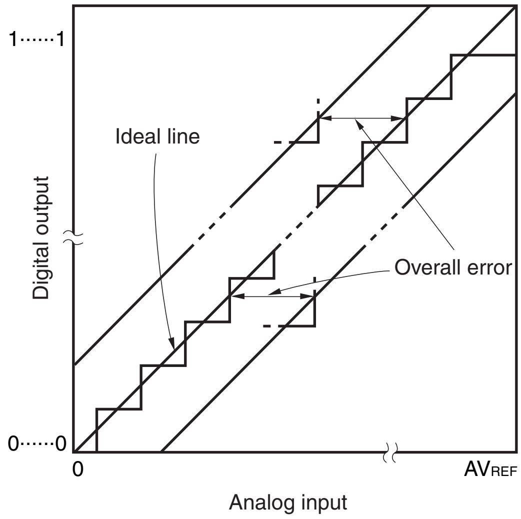


Figure 19-10 Overall error

(3) Quantization error

This is an error of ±1/2 LSB that inevitably occurs when an analog value is converted into a digital value. Because the A/D Converter converts analog input voltages in a range of ±1/2 LSB into the same digital codes, a quantization error is unavoidable.

This error is not included in the overall error, zero-scale error, full-scale error, integral linearity error, or differential linearity error in the characteristics table.

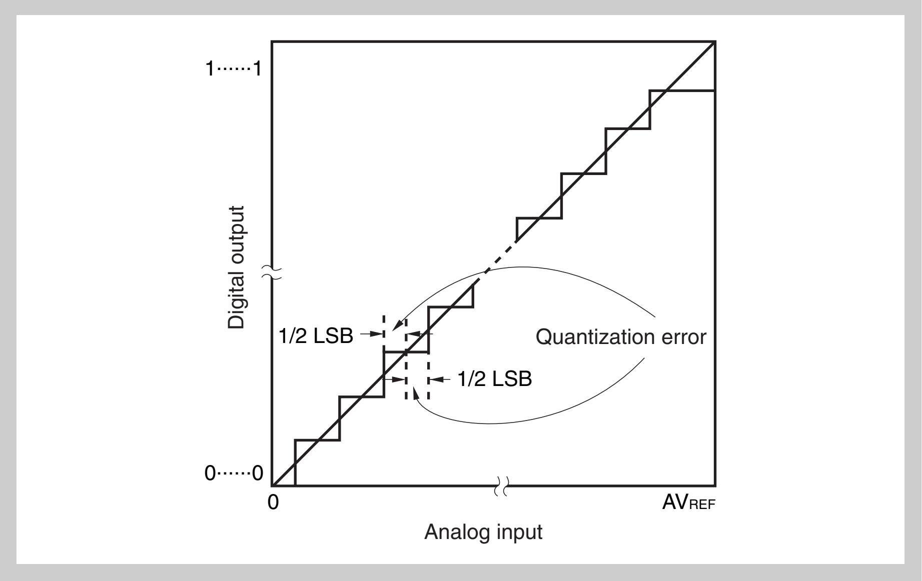


Figure 19-11 Quantization error

(4) Zero-scale error

This is the difference between the actually measured analog input voltage and its theoretical value when the digital output changes from 0…000 to 0…001 (1/2 LSB).

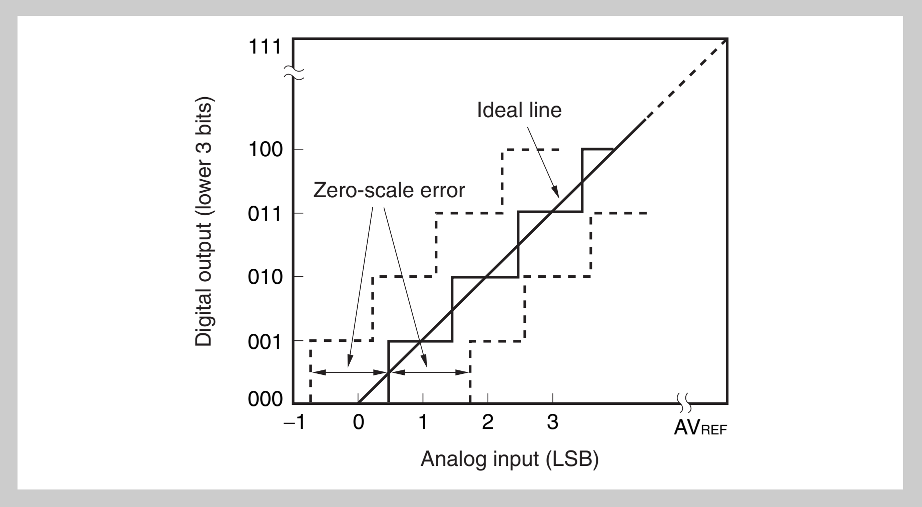


Figure 19-12 Zero-scale error

(5) Full-scale error

This is the difference between the actually measured analog input voltage and its theoretical value when the digital output changes from 1…110 to 0…111 (full scale - 3/2 LSB).

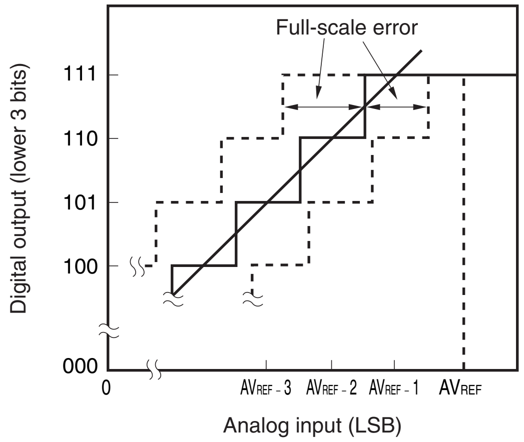


Figure 19-13 Full-scale error

(6) Differential linearity error

Ideally, the width to output a specific code is 1 LSB. This error indicates the difference between the actually measured value and its theoretical value when a specific code is output.

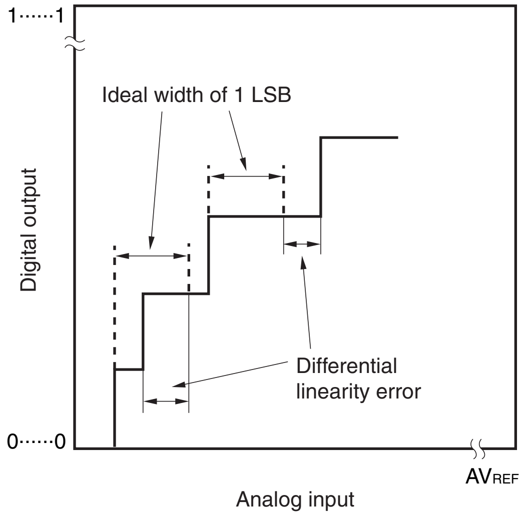


Figure 19-14 Differential linearity error

(7) Integral linearity error

This error indicates the extent to which the conversion characteristics differ from the ideal linear relationship. It indicates the maximum value of the difference between the actually measured value and its theoretical value where the zero-scale error and full-scale error are 0.

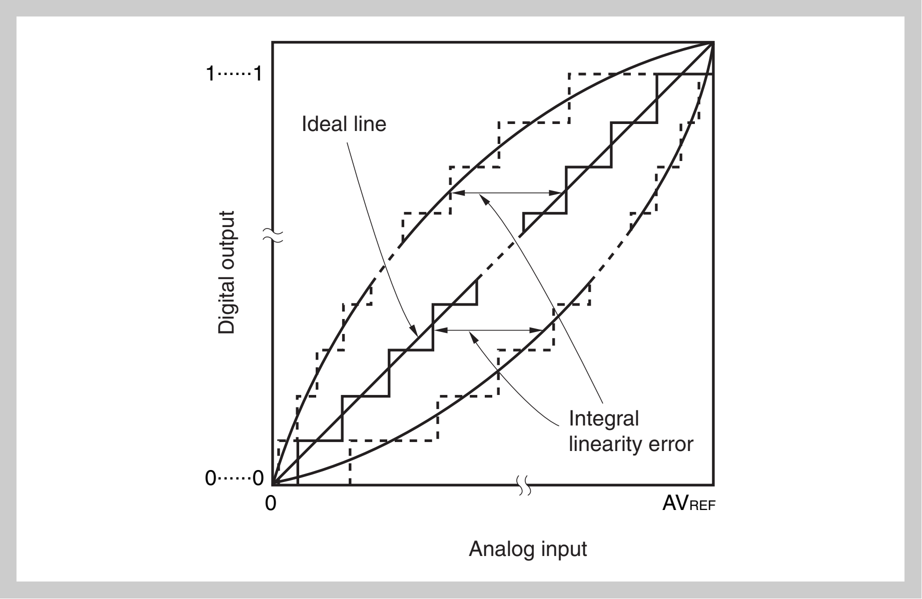


Figure 19-15 Integral linearity error

(8) Conversion time

This is the time required to obtain a digital output after an analog input voltage has been assigned.

The conversion time in the characteristics table includes the sampling time.

(9) Sampling time

This is the time for which the analog switch is ON to load an analog voltage to the sample & hold circuit.

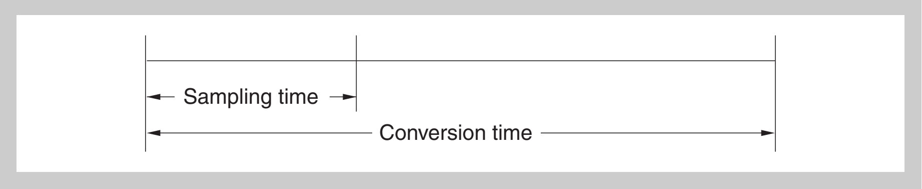


Figure 19-16 Sampling time

Chapter 20 Stepper Motor Controller/Driver (Stepper-C/D)

The Stepper Motor Controller/Driver module is comprised of four drivers (k = 3 to 6) for external 360° type meters or for bipolar and unipolar stepper motors.

The V850E/Dx3 - DG3 microcontrollers have following instances of the Stepper Motor Controller/Driver:

Stepper-C/D	All devices
Instances	1

Throughout this chapter, the individual instances of Stepper-C/D are identified by “n”, for example MCNTCn0, or MCNTCn1 for the timer mode control registers.

The Stepper Motor Controller/Driver module can be separated into two sub-modules. Throughout this chapter, the individual sub-modules are identified by “m” (m = 0, 1).

20.1 Overview

The Stepper Motor Controller/Driver module generates pulse width modulated (PWM) output signals. Each driver generates up to four output signals.

Features summary

The generated output signals have the following features:

- Pulse width of 8 bits precision
- 1-bit addition function enables an average pulse width precision of 1/2 bit, resulting in a pseudo 9-bit precision
- PWM frequency up to 32 KHz
- automatic PWM phase shift for reducing fluctuation on power supply and for reducing the susceptibility to electromagnetic interference

20.1.1 Driver overview

A stepper motor is driven by PWM signals. The PWM signals are generated by comparing the contents of compare registers with the actual value of a free running up counter.

The Stepper Motor Controller/Driver module can be separated into two sub-modules - each sub-module contains one counter and assigned compare registers and control registers. In the following, the two sub-modules are called Stepper Motor Controller/Driver 0 sub-module and Stepper Motor Controller/Driver 1 sub-module.

The following figures show the main components of the Stepper Motor Controller/Driver 0 sub-module (*Figure 20-1*) and of the Stepper Motor Controller/Driver 1 sub-module (*Figure 20-2*).

The Stepper Motor Controller/Driver 0 sub-module is comprised of 2 drivers (k = 3 to 4), Stepper Motor Controller/Driver 1 sub-module is comprised of 2 drivers (k = 5 to 6). Each Stepper Motor Controller/Driver sub-module includes a free running up counter (CNTm). The counter is controlled by a timer mode control register (MCNTCnm).

Each of the four drivers consists of two compare registers, MCMPnk0 and MCMPnk1, respectively. Their contents define the pulse widths for the sine and the cosine side of the meters. The MCMPnk0/MCMPnk1 registers comprise a master-slave register combination. This allows to re-write the master register while the slave register is currently used for comparison with the counter CNTm.

The compare control register MCMPCnk defines whether or not enhanced pulse width precision by one-bit addition is enabled, and it routes the output signals to the corresponding output pins (SMk1 to SMk4).

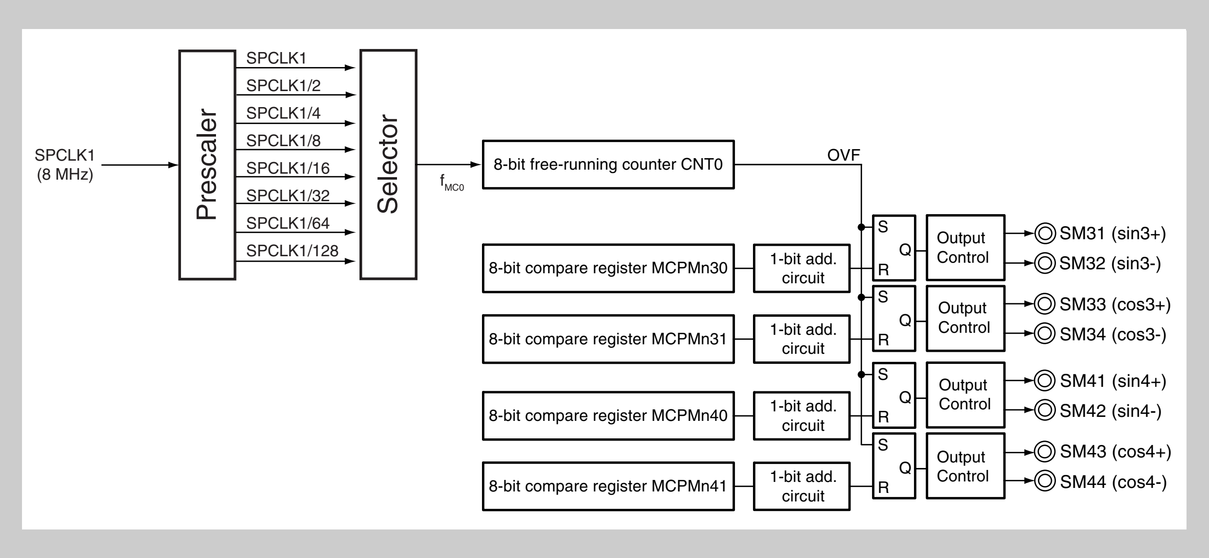


Figure 20-1 Stepper Motor Controller/Driver 0 block diagram

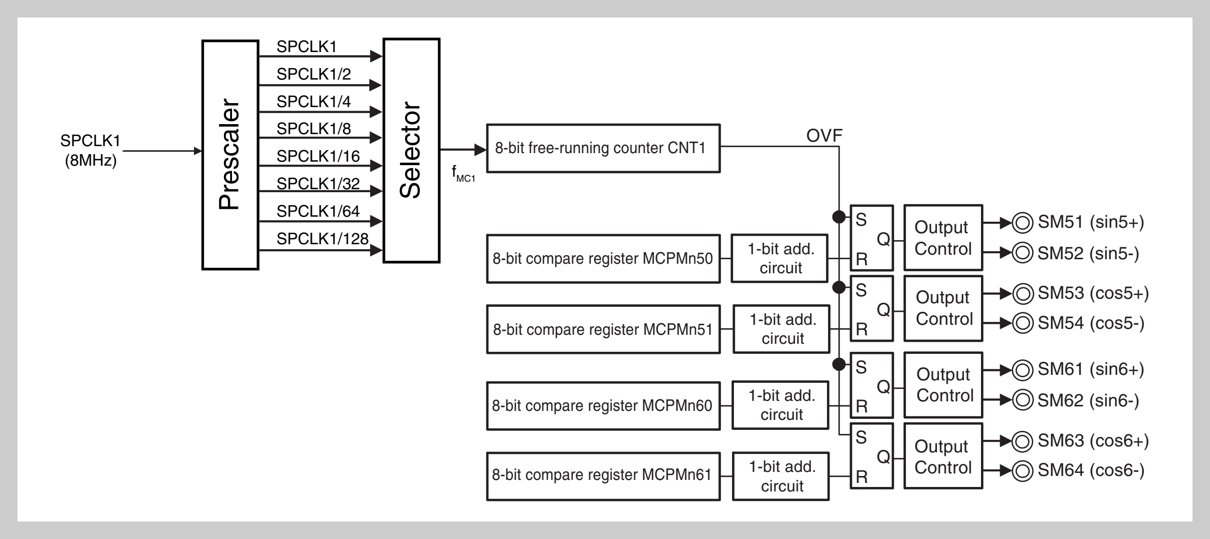


Figure 20-2 Stepper Motor Controller/Driver 1 block diagram

The external signals are listed in the following table.

Table 20-1 Stepper Motor Controller/Driver external connections

Signal name	I/O	Active level	Reset level	Pins	Function
SM[3:6]1	O	–	L	SM31 to SM61	driver signal, sine side (+)
SM[3:6]2	O	–	L	SM32 to SM62	driver signal, sine side (–)
SM[3:6]3	O	–	L	SM33 to SM63	driver signal, cosine side (+)
SM[3:6]4	O	–	L	SM34 to SM64	driver signal, cosine side (–)

20.2 Stepper Motor Controller/Driver Registers

The Stepper Motor Controller/Driver is controlled and operated by means of the following registers:

Table 20-2 Stepper Motor Controller/Driver registers overview

Register name	Shortcut	Address
Timer mode control registers	MCNTCn0	<base>
	MCNTCn1	<base> + 14_H
Compare registers	MCMPnk0 (k = 3 to 6)	<base> + 6_H, 8_H, 16_H, 18_H
	MCMPnk1 (k = 3 to 6)	<base> + 7_H, 9_H, 17_H, 19_H
	MCMPnkHW (k = 3 to 6)	<base> + 6_H, 8_H, 16_H, 18_H
Compare control registers	MCMPCnk (k = 3 to 6)	<base> + E_H, 10_H, $1A_H$, $1C_H$

The base address of the Stepper Motor Controller/Driver is <base> = FFFF $F5C0_H$.

(1) MCNTCn0, MCNTCn1 - Timer mode control registers

The 8-bit MCNTCnm registers control the operation of the free running up counters CNTm.

Access These registers can be read/written in 8-bit or 1-bit units.

Address MCNTCn0: <base>
MCNTCn1: <base> + 14_H

Initial Value 00_H. This register is cleared by any reset.

7	6	5	4	3	2	1	0
CAE[a]	0	FULL	PCE	0	SMCL2	SMCL1	SMCL0
R/W[b]	R	R/W	R/W	R	R/W	R/W	R/W

[a] Bit CAE refers only to register MCNTCn0. In register MCNTCn1, this bit is set to 0.
[b] In register MCNTCn1, this bit is read only (R)

Table 20-3 MCNTCnm register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>7</td><td>CAE[a]</td><td>Stepper Motor Controller/Driver control
0: Stepper Motor Controller/Driver operation is disabled.
1: Stepper Motor Controller/Driver operation is enabled.
This bit switches both Stepper Motor Controller/Driver 0 and Stepper Motor Controller/Driver 1.</td></tr>
<tr><td>5</td><td>FULL</td><td>Sets the count range of the timer counter
0: count range from 01_H to FF_H
1: count range from 00_H to FF_H
The initial start value is 00_H in both cases. For the impact of this bit on duty factor and PWM cycle time, see also <i>“Duty Factor“ on page 719</i>.</td></tr>
<tr><td>4</td><td>PCE</td><td>Timer operation control
0: Timer counter is stopped.
1: Timer counter is enabled.</td></tr>
<tr><td>2 to 0</td><td>SMCL[2:0]</td><td>Sets the timer count clock for the timer counter
<table>
<tr><th>SMCL2</th><th>SMCL1</th><th>SMCL0</th><th>Selected timer count clock</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>SPCLK1</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>SPCLK1 / 2</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>SPCLK1 / 4</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>SPCLK1 / 8</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>SPCLK1 / 16</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>SPCLK1 / 32</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>SPCLK1 / 64</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>SPCLK1 / 128</td></tr>
</table></td></tr>
</table>

[a] Bit CAE refers only to register MCNTCn0. In register MCNTCn1, this bit is set to 0.

Caution In register MCNTCn0, bits 3 and 6 must be 0.
In register MCNTCn1, bits 3, 6 and 7 must be 0.

Power save mode preparation Before entering any power save mode the Stepper-C/D must be shut down in advance in order to minimize power consumption.

Apply following sequence to shut down the Stepper-C/D:

1. Stop the counter CNT1 by setting MCNTCn1.PCE = 0.
2. Stop the counter CNT0 by setting MCNTCn0.PCE = 0.
3. Disable the Stepper-C/D operation by setting MCNTCn0.CAE = 0.

Note that the MCNTCn0.PCE and MCNTCn0.CAE bits must not be cleared to 0 by a single write instruction. Perform two write instructions as shown above.

(2) MCMPnk0 - Compare registers for sine side (k = 3 to 6)

The 8-bit MCMPnk0 registers hold the values that define the PWM pulse width for the sine side of the connected meters.

The contents of the registers are continuously compared to the timer counter value:

- Registers MCMPn**3**0 to MCMPn40 are compared to CNT0.
- Registers MCMPn50 to MCMPn60 are compared to CNT1.

When the register contents match the timer counter contents, a match signal is generated. Thus a PWM pulse with a pulse width corresponding to the MCMPnk0 register contents is output to the sine side of the connected meter.

Access These registers can be read/written in 8-bit units.

Address <base> + 6_H, 8_H, 16_H, 18_H

Initial Value 00_H. This register is cleared by any reset.

7	6	5	4	3	2	1	0
sine DATA							
R/W							

Note

1. New data must only be written to registers MCMPnk0 if the corresponding bit MCMPCnk.TEN = 0.
2. Don't write to the compare register MCMPnk0, until the corresponding bit MCMPCnk.TEN has been reset to 0 automatically.
3. To enable master-to-slave register copy upon next CNTm overflow set MCMPCnk.TEN = 1.

(3) MCMPnk1 - Compare registers for cosine side (k = 3 to 6)

The 8-bit MCMPnk1 registers hold the values that define the PWM pulse width for the cosine side of the connected meters.

The contents of the registers are continuously compared to the timer counter value:

- Registers MCMPn**31** to MCMPn41 are compared to CNT0.
- Registers MCMPn51 to MCMPn61 are compared to CNT1.

When the register contents match the timer counter contents, a match signal is generated. Thus a PWM pulse with a pulse width corresponding to the MCMPnk1 register contents is output to the sine side of the connected meter.

Access These registers can be read/written in 8-bit units.

Address <base> + 7_H, 9_H, 17_H, 19_H

Initial Value 00_H. This register is cleared by any reset.

7	6	5	4	3	2	1	0
cosine DATA							
R/W							

Note

1. New data must only be written to registers MCMPnk1 if the corresponding bit MCMPCnk.TEN = 0.
2. Don't write to the compare register MCMPnk1, until the corresponding bit MCMPCnk.TEN has been reset to 0 automatically.
3. To enable master-to-slave register copy upon next CNTm overflow set MCMPCnk.TEN = 1.

(4) MCMPnkHW - Combined compare registers (k = 3 to 6)

The 16-bit MCPMnkHW registers combine the sine and cosine registers MCMPnk0 and MCMPnk1. Via these registers it is possible to read or write the contents of MCMPnk0 and MCMPnk1 in a single instruction.

Access These registers can be read/written in 16-bit units.

Address <base> + 6_H, 8_H, 16_H, 18_H

Initial Value 0000_H. This register is cleared by any reset.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cosine DATA								sine DATA							
R/W															

Note

1. New data must only be written to registers MCMPnk1 if the corresponding bit MCMPCnk.TEN = 0.
2. Don't write to the compare register MCMPnk1, until the corresponding bit MCMPCnk.TEN has been reset to 0 automatically.
3. To enable master-to-slave register copy upon next CNTm overflow set MCMPCnk.TEN = 1.

(5) MCMPCnk - Compare control registers (k = 3 to 6)

The 8-bit MCMPCnk registers control the operation of the corresponding compare registers and the output direction of the PWM pin.

Access These registers can be read/written in 8-bit units.

Address <base> + E_H, 10_H, $1A_H$, $1C_H$

Initial Value 00_H. This register is cleared by any reset.

7	6	5	4	3	2	1	0
AOUT	0[a]	0[b]	TEN	ADB1	ADB0	DIR1	DIR0
R/W	R/W	R	R/W	R/W	R/W	R/W	R/W

[a] Do not change this bit.

[b] This bit may be written, but writing is ignored.

Table 20-4 MCMPCnk register contents

Bit position	Bit name	Function
7	AOUT	Selects the output pins for sine and cosine signals 0: The PWM signals for sine and cosine side are output to those pins that are selected by bits DIR0 and DIR1. At all other pins, the output signal is 0 (SMV_{SS} level). 1: The PWM signal for the sine side is output to pins SMk1 and SMk2. The PWM signal for the cosine side is output to pins SMk3 and SMk4.
4	TEN	Transfer enable control bit 0: MCMPnk0/MCMPnk1 master-to-slave register copy is disabled. New data can be written to compare registers MCMPnk0 or MCMPnk1. 1: MCMPnk0/MCMPnk1 master-to-slave register copy is enabled. The copy process will take place when CNT0 or CNT1, respectively, overflows. Don't write to compare registers MCMPnk0 or MCMPnk1 while MCMPCnk.TEN = 1. **Note:** This bit functions as a control bit and status flag. It is automatically reset to zero upon the next timer counter overflow.
3	ADB1	Sets 1-bit addition function for cosine side 0: no 1-bit addition to PWM signal 1: 1-bit addition to PWM signal
2	ADB0	Sets 1-bit addition function for sine side 0: no 1-bit addition to PWM signal 1: 1-bit addition to PWM signal
1 to 0	DIR[1:0]	Selects the output pins for the PWM signals. Bits DIR1 and DIR0 address the quadrant to be activated by sine and cosine. The PWM signal is routed to the specific pin with respect to the sin/cos of each quadrant. At the other output pins, the output level is SMV_{SS}. **Note:** These bits are only considered if bit AOUT is set to 0.

DIR1	DIR0	Selected output pins
0	0	Quadrant 1: SMk1 (sin +), SMk3 (cos +)
0	1	Quadrant 2: SMk1 (sin +), SMk4 (cos –)
1	0	Quadrant 3: SMk2 (sin –), SMk4 (cos –)
1	1	Quadrant 4: SMk2 (sin –), SMk3 (cos +)

20.3 Operation

In the following, the operation of the Stepper Motor Controller/Driver module as a driver for external meters is described.

20.3.1 Stepper Motor Controller/Driver operation

This section describes the generation of PWM signals of the driver k for driving external meters. Further, the achievable duty factor is explained and how advanced precision can be gained by 1-bit addition.

(1) Driving Meters

External meters can be driven both in H-bridge configuration and in half bridge configuration:

- Driving meters in H-bridge configuration

  Deflection of the needle of a meter in H-bridge configuration is determined by the sine and cosine value of its desired angle. Since the PWM signals do not inherit a sign, separate signals for positive and negative sine and cosine values are generated.
  The four signals at pins SMk1 to SMk4 of the driver k are:
  – sine side, positive (sin +)
  – sine side, negative (sin –)
  – cosine side, positive (cos +)
  – cosine side, negative (cos –)

  Two output control circuits select which signal (sign) for sine side and cosine side is output (bits MCMPCnk.DIR[1:0]). At the remaining two output pins, the signal is set to low level.
  To drive meter k in full bridge mode, set bit MCMPCnk.AOUT to 0.

- Driving meters in half bridge configuration

  In this mode, the same signal is sent to both sine pins (SMk1 and SMk2) and both cosine pins (SMk3 and SMk4), respectively. The setting of output control bits MCMPCnk.DIR[1:0] is neglected.
  To drive meter k in half bridge mode, set bit MCMPCnk.AOUT to 1.

(2) Generation of PWM signals

Bit data corresponding to the length of the PWM pulses has to be written to the compare registers MCMPnk0 (sine side) and MCMPnk1 (cosine side).

A timer counter is counting up. The rising edge of the PWM pulse is initiated at the overflow of the counter. The falling edge of the PWM pulse is initiated when the counter value equals the contents of the compare register.

The absolute pulse length in seconds is defined by the timer count clock (f_{MC0} and f_{MC1}, respectively). Various cycle times can be set via the timer mode control registers MCNTCn0 and MCNTCn1.

Instruction When writing data to compare registers, proceed as follows:

1. Confirm that MCMPCnk.TEN = 0.
2. Write 8-bit PWM data to MCMPnk0 and MCMPnk1.
3. Set MCMPCnk.ADB0 and MCMPCnk.ADB1 as desired.
4. Set MCMPCnk.TEN = 1 to start the counting operation.

   The data in MCMPnk0/MCMPnk1 will automatically be copied to the compare slave register when the counter overflows. The new pulse width is valid immediately.

   Bit MCMPCnk.TEN is automatically cleared to 0 by hardware.

(3) Duty Factor

The minimum pulse width that can be generated is zero (output signal is low) and the maximum pulse width is 255 clock cycles (maximum value of 8-bit compare registers).

The count range of the timer counter defines the duty factor. It can be set by bit MCNTCnm.FULL:

- count range 01_H to FF_H (MCNTCnm.FULL = 0)

  Formula for the duty cycle:
  PWM duty = MCMPki / 255 with k = 3 to 6 and i = 0, 1

  One count cycle is comprised of 255 clock cycles. A PWM signal with maximum pulse length is a steady high level signal. The duty factor is 100%.

- count range 00_H to FF_H (MCNTCnm.FULL = 1)

  Formula for the duty cycle:
  PWM duty = MCMPki / 256 with k = 3 to 6 and i = 0, 1

  One count cycle is comprised of 256 clock cycles. A PWM signal with maximum pulse length is comprised of 255 clock cycles at high level and one clock cycle at low level. The duty factor is 255/256 *100% = 99.6%.

(4) Advanced precision by 1-bit addition

The precision of the angle of a needle is implicitly defined by the number of bits of the compare registers MCMPnk0 and MCMPnk1 (8 bit).

If the 1-bit addition circuit is enabled, every second pulse of the PWM signal is extended by one bit (one clock cycle). In average, a pulse width precision of 1/2 bit (1/2 clock) can be achieved.

The following figures show the timing of PWM output signals with 1-bit addition disabled and enabled.

Note

1. The PWM pulse is not generated until the first overflow occurs after the counting operation has been started.
2. The PWM signal is two cycle counts delayed compared to the overflow signal and the match signal. This is not depicted in the figures.

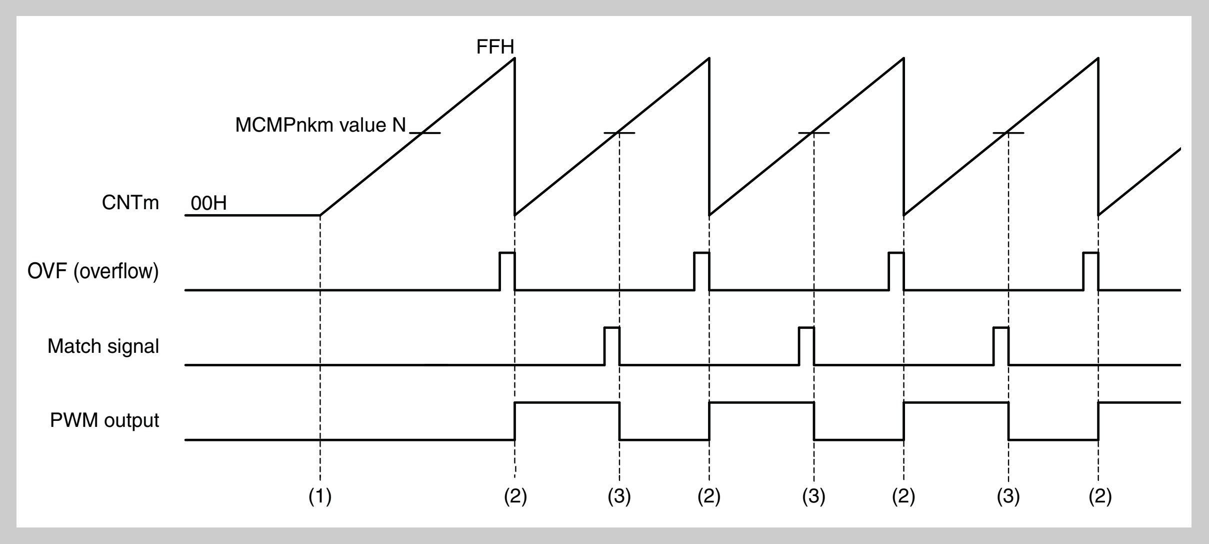


Figure 20-3 Output timing without 1-bit addition

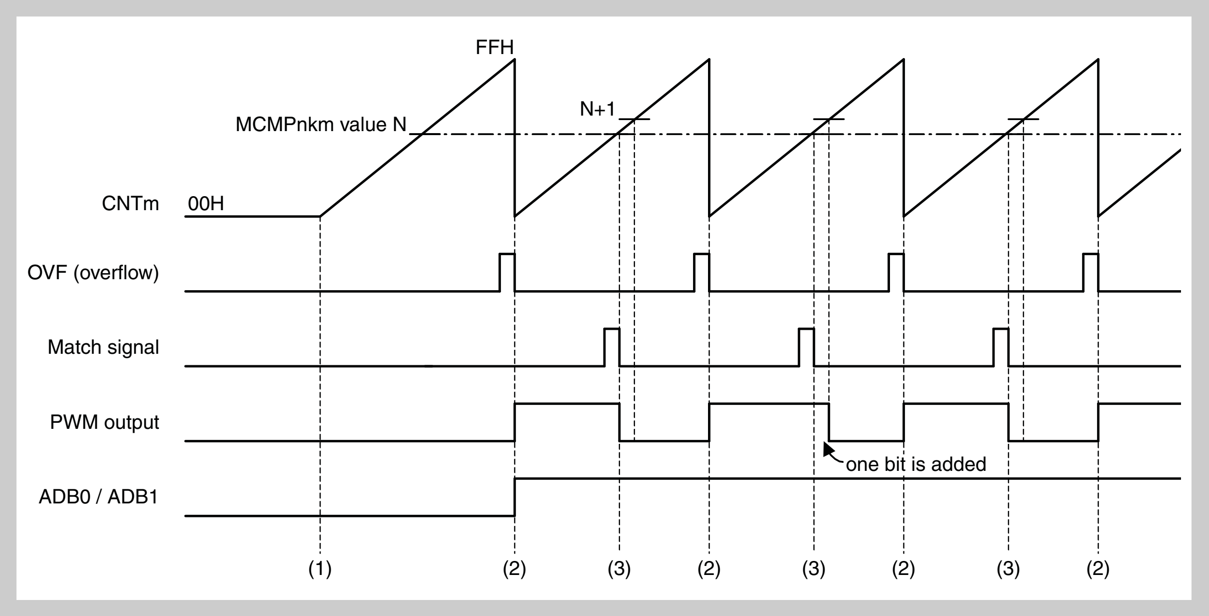


Figure 20-4 Output timing with 1-bit addition

Sequence

1. Start of counting (MCNTCnm.PCE is set to 1)
2. Generation of overflow signal (start of PWM pulse)
3. Generation of match signal (timer counter CNTm matches compare register, end of PWM pulse)

20.4 Timing

This section starts with the timing of the timer counter and general output timing behaviour. Then, examples of output signal generation with and without 1-bit addition are presented.

20.4.1 Timer counter

The free running up counter is clocked by the timer count clock selected in register MCNTCnm.

The counting operation is enabled or disabled by the MCNTCnm.PCE bit.

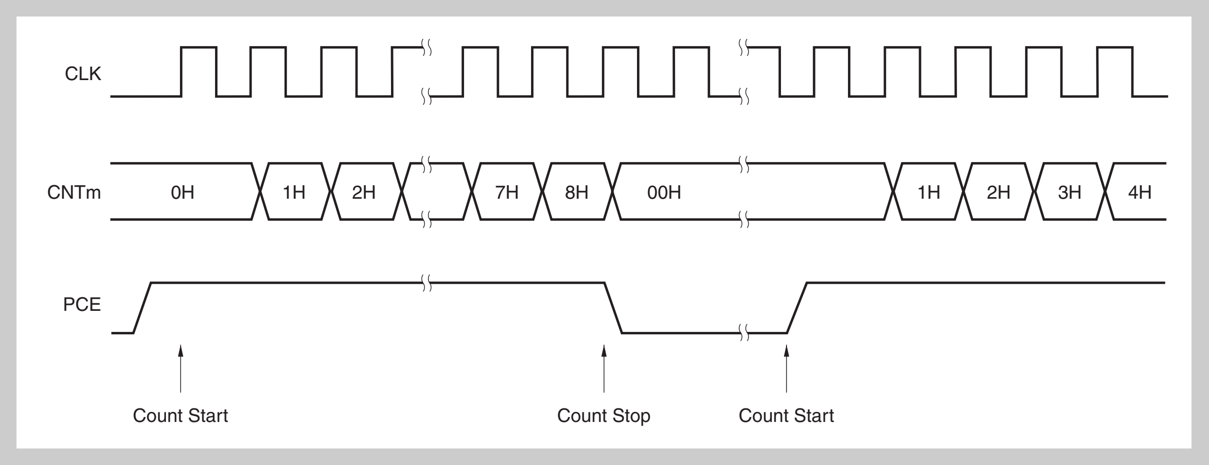


Figure 20-5 Restart Timing after Count Stop (Count Start—Count Stop—Count Start)

Sequence

- Count Start:
  - Enable counting operation (MCNTCnm.PCE = 1)
  - Timer counter starts with value 00_H. Depending on bit MCNTCnm.FULL, all following counter cycles start with 00_H or 01_H, respectively.
- Count Stop:
  - Disable counting operation (MCNTCnm.PCE = 0)
  - Counting is stopped and timer counter is set to 00_H.

20.4.2 Automatic PWM phase shift

Simultaneous switching of sine and cosine output could lead to a fluctuation of the power supply and increase the susceptibility to electromagnetic interference. To prevent this for drivers 3 to 4, the output signals are automatically shifted by one timer count clock cycle defined in MCNTCn0.

The same accounts for the output signals of drivers 5 and 6. They are controlled by the timer count clock defined in MCNTCn1.

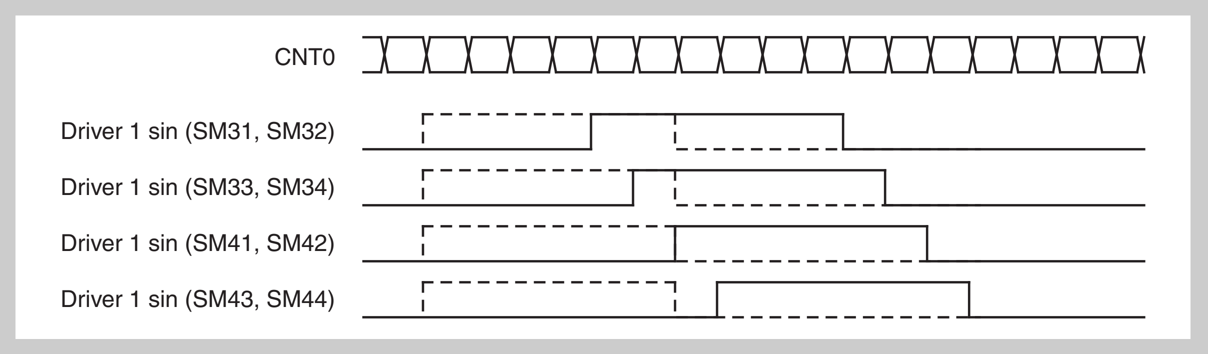

CNT0
Driver 1 sin (SM31, SM32)
Driver 1 sin (SM33, SM34)
Driver 1 sin (SM41, SM42)
Driver 1 sin (SM43, SM44)

Figure 20-6 Output timing of signals SM31 to SM44

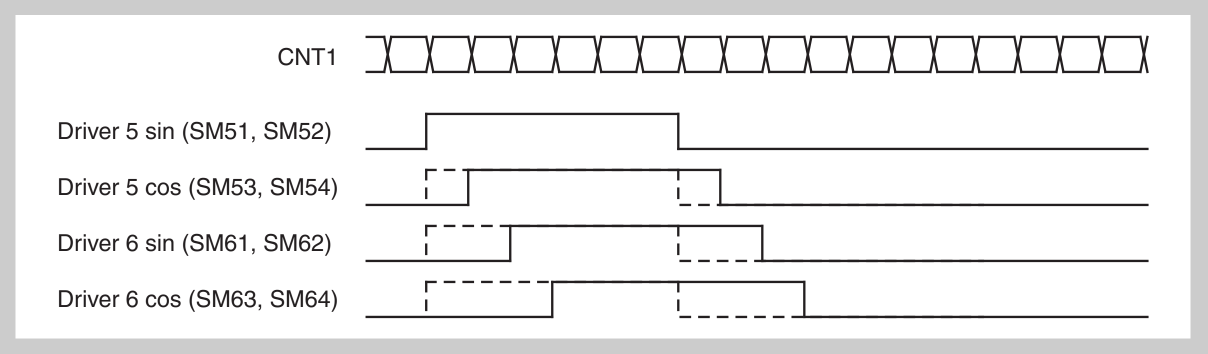

CNT1
Driver 5 sin (SM51, SM52)
Driver 5 cos (SM53, SM54)
Driver 6 sin (SM61, SM62)
Driver 6 cos (SM63, SM64)

Figure 20-7 Output timing of signals SM51 to SM64

Chapter 21 LCD Controller/Driver (LCD-C/D)

This LCD Controller/Driver is suitable for LC displays with up to 160 segments. The supported addressing method of the LCD is multiplex addressing.

21.1 Overview

The LCD Controller/Driver generates the signals that are necessary for driving an LCD panel.

Features summary The LCD Controller/Driver provides:

- Maximum of 40 segment signal outputs (SEG0 to SEG39)
- 4 common signal outputs (COM0 to COM3)
- Display mode: 1/4 duty (1/3 bias)
- Wide range of selectable frame frequencies
- Edge enhancement

21.1.1 Description

The following figure shows the main components of the LCD Controller/Driver:

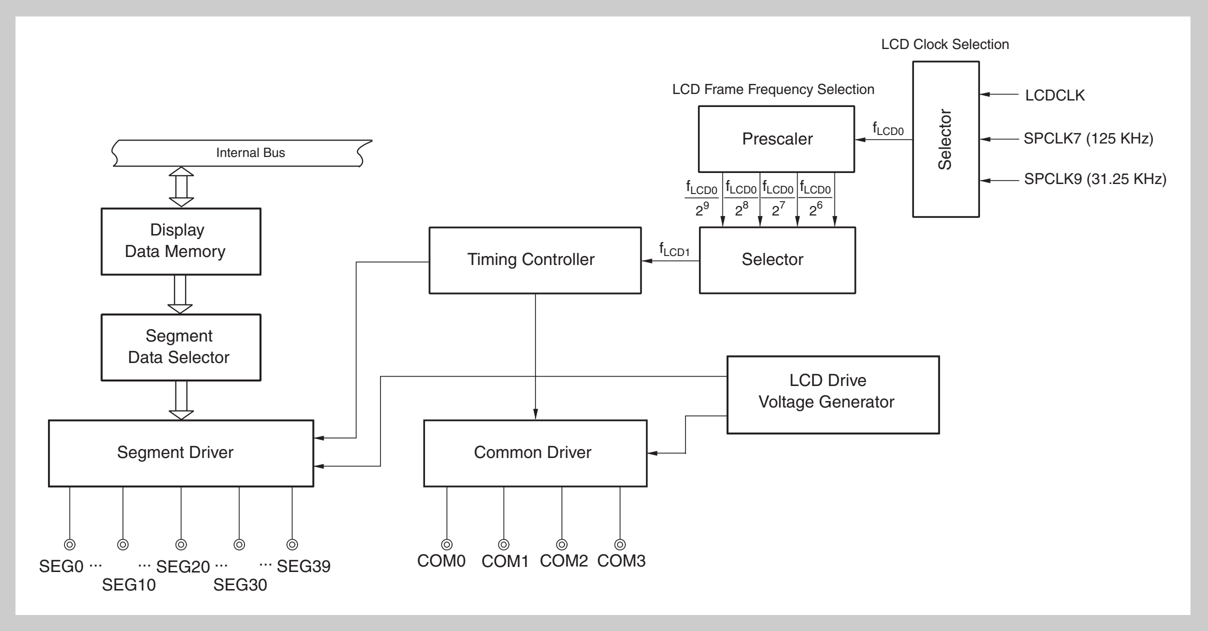


Figure 21-1 LCD Controller/Driver block diagram

The pattern that is to be displayed on the LCD panel has to be mapped to bit data. The bit data is stored in the display control registers SEGREG0k (k = 00 to 39). The LCD Controller/Driver generates the corresponding output signals for driving the LCD panel.

The update rate of the LC display is determined by the frame frequency. It can be adjusted via the clock control register LCDC.

The external signals are listed in the following table.

Table 21-1 LCD Controller/Driver external connections

Signal name	I/O	Pins	Function
SEG[0:39]	O	SEG0 to SEG39	Segment signals
COM[0:3]	O	COM0 to COM3	Common signals

21.1.2 LCD panel addressing

Each individual segment of an LCD panel is addressed by a signal pair: a segment signal and a common signal. The segment becomes visible when the potential difference of the corresponding common signal and the segment signal reaches or exceeds the LCD drive voltage V_{LCD}.

Example *Figure 21-2* shows how the eight LCD segments of a digit are allocated to

- two segment signals (SEG_{2n} and $SEG_{2n+1,}$ n = 0 to 19)
- four common signals

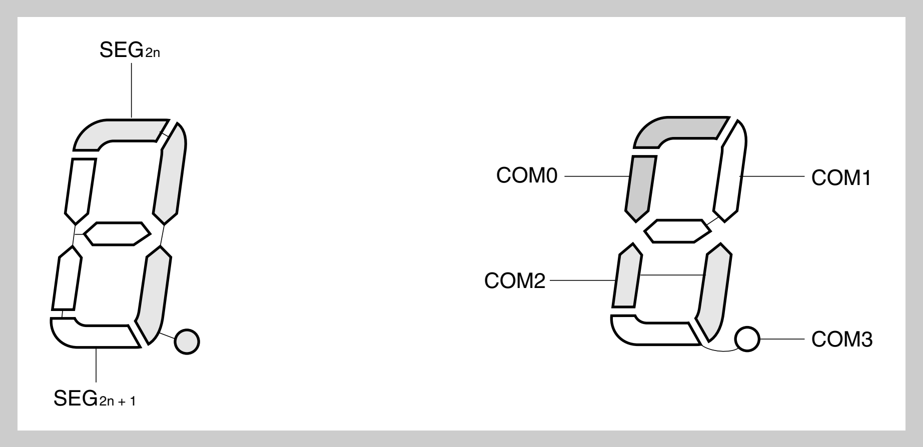


Figure 21-2 **Allocation of segment signals and common signals to LCD segments (4-time-division)**

Every combination of a segment and a common signal addresses a single element. The middle horizontal bar, for example, becomes visible if the potential difference of signals SEG_{2n+1} and COM1 exceeds V_{LCD}.

To display a desired pattern on the LCD panel:

1. Check what combination of segment and common signals form the desired display pattern.
2. Write bit data with the pattern to be displayed to registers SEGREG0k.

The LCD Controller/Driver generates the corresponding segment and common signals.

See also the *"Display Example" on page 733*.

Connections At the LCD panel, the signals are connected as follows:

Table 21-2 **Signals and connections of LCD Controller/Driver**

Signals	Connection at LCD panel
segment signals	front surface electrodes
common signals	rear surface electrodes

Caution The LCD panel is driven by AC voltage. The performance of the LCD deteriorates if DC voltage is applied in the common and segment signals. That means contrast and brightness of the display may decrease. The display may even be damaged.

21.2 LCD-C/D Registers

The LCD Controller/Driver is controlled by means of the following registers:

Table 21-3 LCD Controller/Driver registers overview

Register name	Shortcut	Address
LCD clock control register	LCDC0	FFFF $FB00_H$
LCD mode control register	LCDM0	FFFF $FB01_H$
LCD display control registers	SEGREG0k, k= 00 to 39	FFFF $FB20_H$ to FFFF $FB47_H$

(1) LCDC0 - LCD clock control register

The 8-bit LCDC0 register determines the duty cycle frequency f_{LCD1}.

Access This register can be read/written in 8-bit or 1-bit units.

Address FFFF FB00$_H$

Initial Value 00_H. This register is cleared by any reset.

7	6	5	4	3	2	1	0
0	0	0	0	LCDC03	LCDC02	LCDC01	LCDC00
R	R	R	R/W	R/W	R/W	R/W	R/W

Table 21-4 LCDC0 register contents

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>3 to 2</td><td>LCDC0[3:2]</td><td>Selects the LCD clock
<table>
<tr><th>LCDC03</th><th>LCDC02</th><th>Selected LCD clock (f_{LCD0})</th></tr>
<tr><td>0</td><td>0</td><td>LCDCLK</td></tr>
<tr><td>0</td><td>1</td><td>SPCLK7</td></tr>
<tr><td>1</td><td>0</td><td>SPCLK9</td></tr>
<tr><td>1</td><td>1</td><td>reserved</td></tr>
</table></td></tr>
<tr><td>1 to 0</td><td>LCDC0[1:0]</td><td>Selects the duty cycle frequency
<table>
<tr><th>LCDC01</th><th>LCDC00</th><th>Selected duty cycle frequency (f_{LCD1})</th></tr>
<tr><td>0</td><td>0</td><td>LCD clock (f_{LCD0}) divided by 2^6</td></tr>
<tr><td>0</td><td>1</td><td>LCD clock (f_{LCD0}) divided by 2^7</td></tr>
<tr><td>1</td><td>0</td><td>LCD clock (f_{LCD0}) divided by 2^8</td></tr>
<tr><td>1</td><td>1</td><td>LCD clock (f_{LCD0}) divided by 2^9</td></tr>
</table></td></tr>
</table>

Caution

1. Bit 4 must always be 0.
2. Changing the root clock source for LCDLCK will also change the Watch Timer clock WTCLK. For details refer to the *“TCC - Watch Timer clock control register“ on page 124*.

Note The frequency of LCDCLK is determined in the Clock Generator.
The root clock for LCDCLK can be selected from the main, sub, or internal oscillator. It can be identical with the clock source or it can be a fraction thereof.

Possible frame frequencies

Table 21-5 lists the possible frame frequencies. The values in *Table 21-5* are only examples. Check *“Clock Generator“ on page 100* for details.

Selection of the following LCD clocks is provided:

- LCDC0.LCDC0[3:2] = 00_B
  LCD clock (f_{LCD0}) = LCDCLK = f_0 / d, with
  - f_0 = root clock for LCDCLK
    can be selected from main oscillator ((f_{MOCLK} = 4 MHz), sub oscillator (f_{SOCLK} = 32.768 KHz), or internal oscillator (f_{ROCLK} ~ 240 KHz).
  - d = divider
    LCDCLK is gained by dividing the root clock by d. Divider d can be selected from 2^0 to 2^7.

  For details refer to the *“TCC - Watch Timer clock control register“ on page 124*.
- LCDC0.LCDC0[3:2] = 01_B
  LCD clock (f_{LCD0}) = SPCLK7 = SPCLK0 / 2^7 = 125 KHz
- LCDC0.LCDC0[3:2] = 10_B
  LCD clock (f_{LCD0}) = SPCLK9 = SPCLK0 / 2^9 = 31.25 KHz

Table 21-5 Example settings for frame frequency and duty cycle

LCDC03	LCDC02	LCDC01	LCDC00	LCD clock (f_{LCD0})[a]	Duty cycle frequency (f_{LCD1})	Frame frequency
0	1	0	1	SPCLK7 = 125 KHz	977 Hz	244 Hz
0	1	1	0		488 Hz	122 Hz
0	1	1	1		244 Hz	61 Hz
1	0	0	0	SPCLK9 = 31.25 KHz	488 Hz	122 Hz
1	0	0	1		244 Hz	61 Hz
0	0	0	0	LCDCLK = 32.768 KHz (with f_0 = f_{SOCLK} and d = 2^0)	512 Hz	128 Hz
0	0	0	1		256 Hz	64 Hz
0	0	0	1	LCDCLK ~ 120 KHz (with f_0 = f_{ROCLK} and d = 2^1)	~938 Hz	~234 Hz
0	0	1	0		~469 Hz	~117 Hz

[a] The frequency of the LCD clock (f_{LCD0}) is determined bv the setting of the Clock Generator. For details refer to the *“Clock Generator“ on page 100*.

(2) LCDM0 - LCD mode control register

The 8-bit LCDM0 register enables/disables the LCD operation, activates edge enhancement and selects the power supply.

Access This register can be read/written in 8-bit or 1-bit units.

Address FFFF FB01$_H$

Initial Value 00$_H$. This register is cleared by any reset.

7	6	5	4	3	2	1	0
LCDON0	0	0	LIPS0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 21-6 LCDM0 register contents

Bit position	Bit name	Function
7	LCDON0	Enables/disables LCD display 0: Display disabled No segment of the display is visible. The contents of the SEGREG0k registers are disregarded. The output is at non-selection level. 1: Display enabled
4	LIPS0	Selects the power supply 0: LCD Controller/Driver is not powered 1: LCD Controller/Driver is powered

Caution Bits 0, 1, 2, 3, 5, 6 must always be 0.

(3) SEGREG0k - LCD display control register (k = 00 to 39)

The 8-bit registers contain the data that is displayed on the LCD. Each register contains the data for one of the 40 segments.

Access These registers can be read/written in 8-bit or 1-bit units.

Address FFFF FB20$_H$ to FFFF FB47$_H$

Initial Value 00$_H$. This register is cleared by any reset.

7	6	5	4	3	2	1	0
0	0	0	0	DATA			
R/W							

Table 21-7 SEGREG0k register contents (k = 0 to 39)

Bit position	Bit name	Function
3 to 0	SEGREG0k[3:0]	Status of the LCD segment that is controlled by segment signal k and the common signal, that corresponds to the bit position. 0: Display off 1: Display on, if corresponding common signal is active

The bits 4 to 7 are ignored. They should be set to zero.

21.3 Operation

The following describes the timing of common and segment signals, the activation of an LCD segment and how edge enhancement can be applied.

21.3.1 Common signals and segment signals

This section describes the timing of common signals and segment signals and at which conditions an individual LCD segment becomes visible.

(1) Common Signals

Common signals COM0 to COM3 are generated internally. Together with the segment signals, they define which LCD segment is activated in the current cycle.

Figure 21-3 shows the common signal wave form for COM0, 1/4 duty (1/3 bias). 1/4 duty means each signal COMn is in selection level for one quarter of a frame.

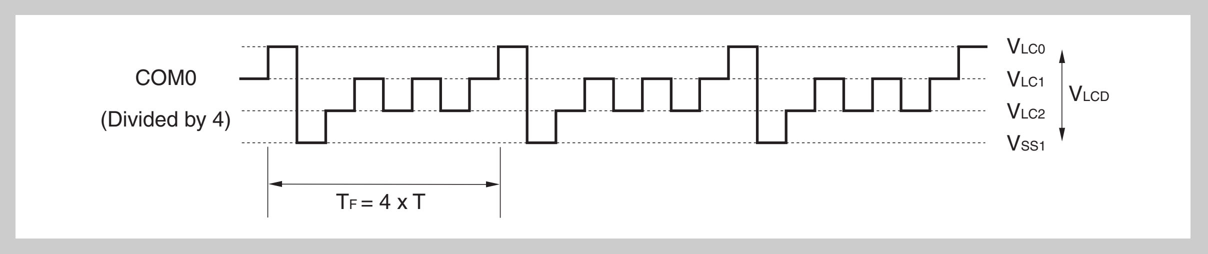


Figure 21-3 Common signal wave form (1/4 duty, 1/3 bias)

- T_F = frame cycle time.
  $T_F = 4 \times T$
  T corresponds to the duty cycle frequency f_{LCD1} and is thus determined by register LCDC.
- T = duty cycle time.
  Each frame cycle T_F is comprised of 4 duty cycles (1/4 duty), one duty cycle for each signal COMn.

Each LCD segment is allocated to one of the common signals. The LCD segment can only be activated in a duty cycle, in which the common signal is at selection level.

Figure 21-4 shows the selection and non-selection level of common signals.

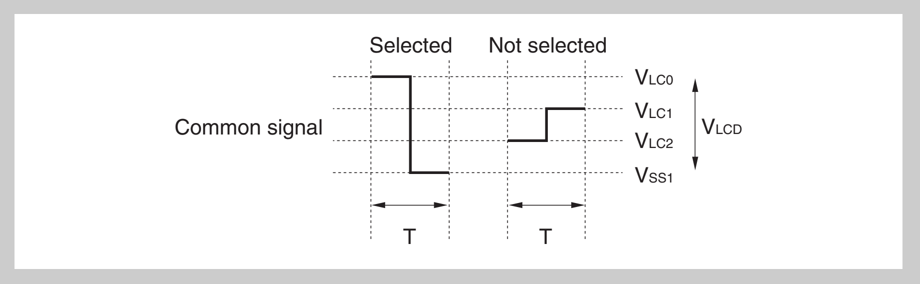


Figure 21-4 Selection level and non-selection level of common signals

T = duty cycle time.

(2) Segment Signals

Segment signals correspond to the contents of the 40 LCD display control registers SEGREG0k. Bits 0 to 3 of these registers are read in synchronization with the common signals COM0 to COM3, this means bit 0 is read in synchronization with common signal COM0 and so on.

- If the value of the bit is 1 while the common signal is at selection level, the corresponding segment signal is set to selection level.
- If the value of the bit is 0 while the common signal is at selection level, the corresponding segment signal is set to non-selection level.

Figure 21-5 shows the selection and non-selection level of segment signals.

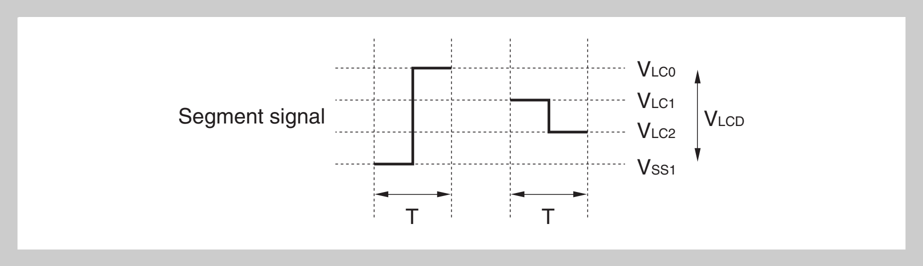


Figure 21-5 Selection level and non-selection level of segment signals

T = duty cycle time.

The Figure below shows the relation of the bits in registers SEGREG0k (k = 00 to 39) with common signals COM0 to COM3 and segment output signals SEG00 to SEG39.

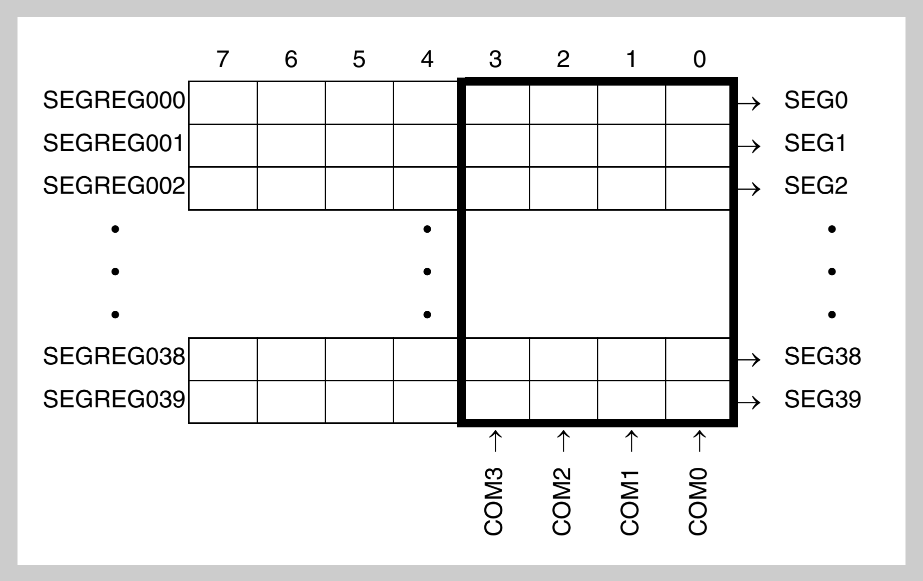


Figure 21-6 Relation between LCD display control registers and segment and common lines

Each of the bits 0 to 4 represents the status of one LCD segment. Setting the bit to 1 will make the LCD segment visible.

For example, setting bit SEGREG002[3] to 1 will make the LCD segment visible, that is controlled by the signal pair SEG2 and COM3.

21.3.2 Activation of LCD segments

An LCD segment becomes visible when the potential difference of the corresponding common signal and segment signal reaches or exceeds the LCD drive voltage V_{LCD}. This is achieved if common and segment signal are at their selection levels.

Within one frame cycle T_F , each LCD segment can be activated once. Activation lasts for one duty cycle T. LCD segments corresponding to common signals COM0 to COM3 are not activated simultaneously, but consecutively.

21.4 Display Example

As a display example, register contents and output signals for a 20-digit LCD display are presented in this section.

(1) LCD panel

The display pattern of a single digit is given below. Each digit is addressed by two segment signals and four common signals.

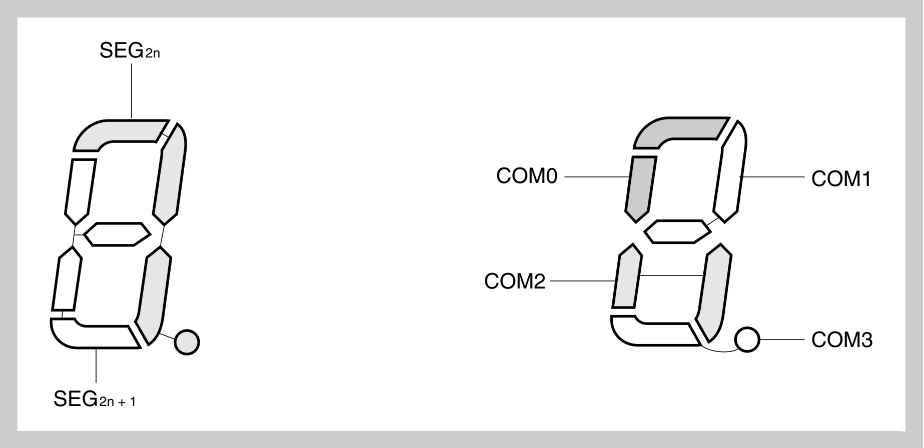


Figure 21-7 4-time-division LCD pattern and electrode connections

Figure 21-8 on page 734 shows the whole LCD panel and its connection to the segment signals and common signals. The display example is “123456.78901234567890,” and the register contents of SEGREG0k (k = 00 to 39) correspond to this.

An explanation is given here taking the example of the 6th digit with point: “6.”. The corresponding segment signals are output to pins SEG28 and SEG29 with the selection levels at the COM0 to COM3 common signal timings as shown in the table below:

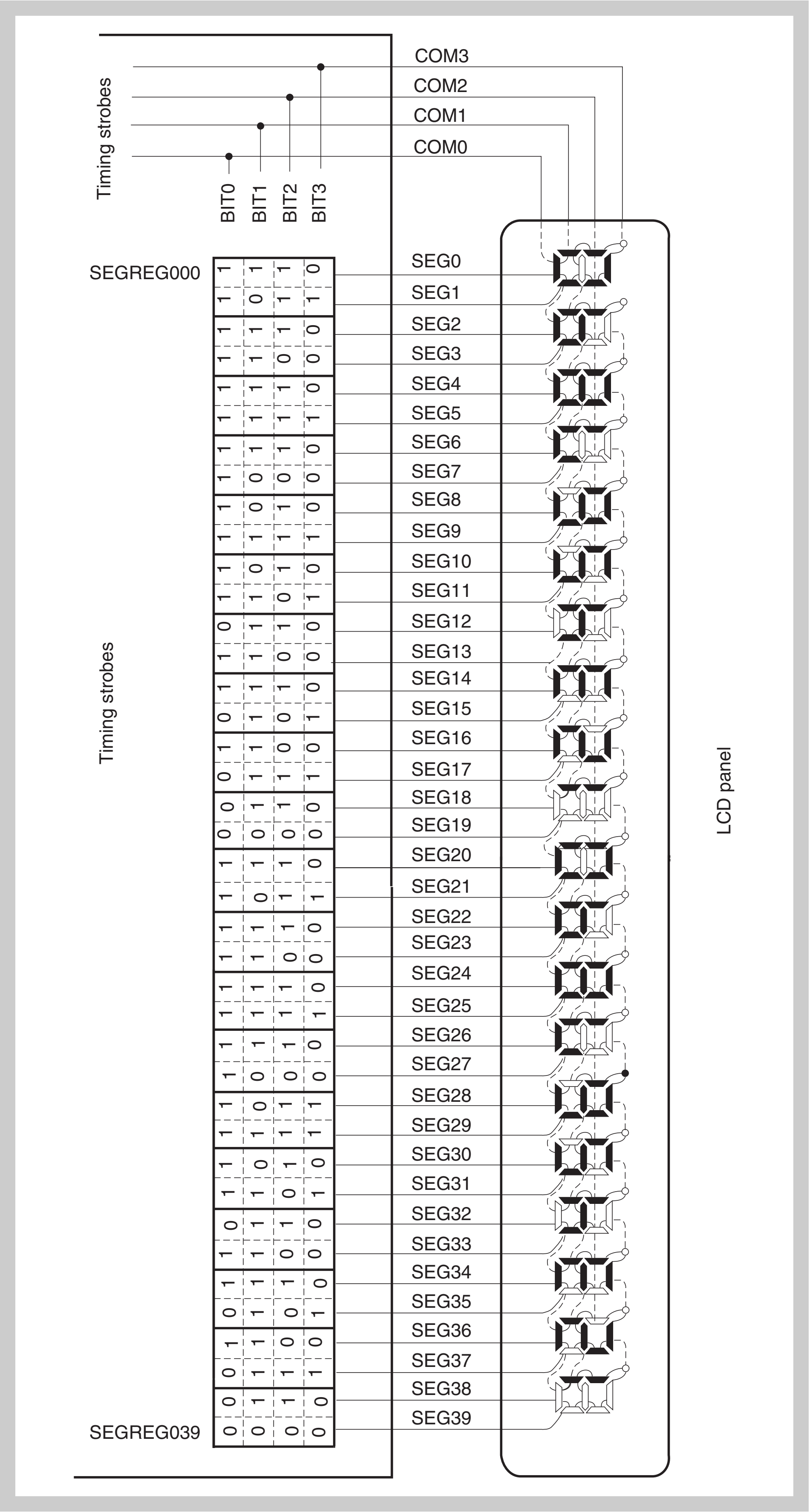
Table 21-8 Selection and non-selection levels of example

Common signal	Segment signal SEG28	Segment signal SEG29
COM0	selected	selected
COM1	not selected	selected
COM2	selected	selected
COM3	selected	selected

From this, it can be seen that 1101_B must be prepared in the display control register SEGREG028 and 1111_B must be prepared in SEGREG029.

Examples of the LCD drive waveforms between SEG28 and the COM0 and COM1 signals are shown in *Figure 21-9 on page 735* (for the sake of simplicity, waveforms for COM2 and COM3 have been omitted).

When SEG28 is at the selection level at the COM0 selection timing, it can be seen that the $+V_{LCD}/–V_{LCD}$ AC square wave, which is the LCD illumination (ON) level, is generated.



	BIT0	BIT1	BIT2	BIT3
SEG0	1	1	1	0
SEG1	1	0	1	1
SEG2	1	1	1	0
SEG3	1	1	0	0
SEG4	1	1	1	0
SEG5	1	1	1	1
SEG6	1	1	1	0
SEG7	1	0	0	0
SEG8	1	0	1	0
SEG9	1	1	1	1
SEG10	1	0	1	0
SEG11	1	1	0	1
SEG12	0	1	1	0
SEG13	1	1	0	0
SEG14	1	1	1	0
SEG15	0	1	0	1
SEG16	1	1	0	0
SEG17	0	1	1	1
SEG18	0	1	1	0
SEG19	0	0	0	0
SEG20	1	1	1	0
SEG21	1	0	1	1
SEG22	1	1	1	0
SEG23	1	1	0	0
SEG24	1	1	1	0
SEG25	1	1	1	1
SEG26	1	1	1	0
SEG27	1	0	0	0
SEG28	1	0	1	1
SEG29	1	1	1	1
SEG30	1	0	1	0
SEG31	1	1	0	1
SEG32	0	1	1	0
SEG33	1	1	0	0
SEG34	1	1	1	0
SEG35	0	1	0	1
SEG36	1	1	0	0
SEG37	0	1	1	1
SEG38	0	1	1	0
SEG39	0	0	0	0

Figure 21-8 4-time-division LCD panel connection example

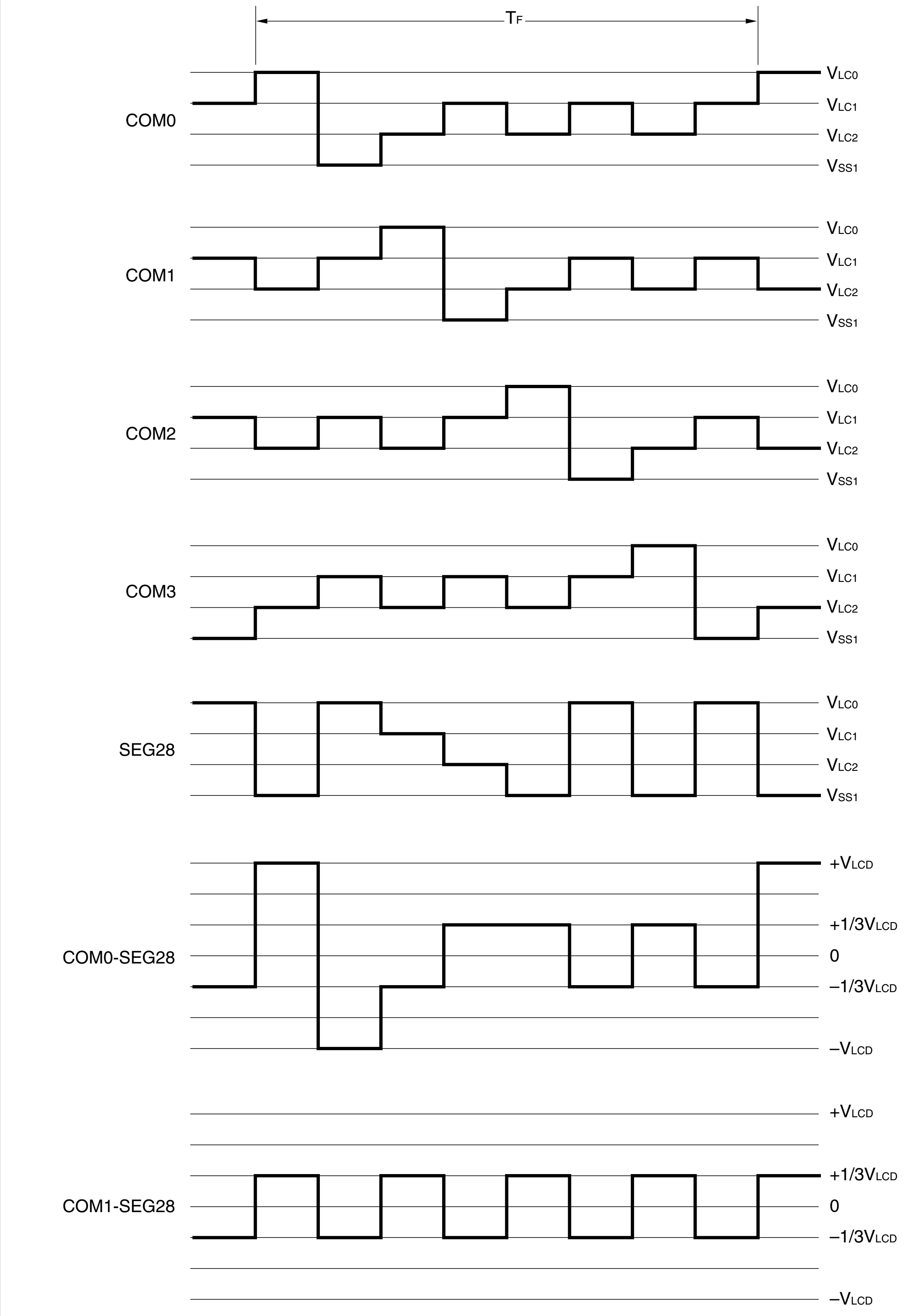


Figure 21-9 4-time-division LCD drive waveforms – examples

Chapter 22 Sound Generator (SG)

The Sound Generator (SG0) generates an audio-frequency tone signal and a high-frequency pulse-width modulated (PWM) signal. The duty cycle of the PWM signal defines the volume.

By default, the two signal components are routed to separate pins. But both signals can also be combined to generate a composite signal that can be used to drive a loudspeaker circuit.

22.1 Overview

The Sound Generator consists of a programmable square wave tone generator and a programmable pulse-width modulator.

The PWM includes an internal automatic logarithmic decrement unit (ALD). The ALD can be used to reduce the tone volume over time without CPU intervention.

Features summary

Special features of the Sound Generator are:

- Programmable tone frequency (100 Hz to 6 KHz with a minimum step size of 20 Hz)
- Programmable volume level (9 bit resolution)
- Automatic logarithmic volume decrement function (ALD):
  - Volume reduction without CPU interaction
  - Programmable sound duration (256 steps)
  - Sound duration associated with tone frequency (gong effect)
  - Interrupt generation when programmable volume low level is reached
- Wide range of PWM signal frequency (32 KHz to 64 KHz)
- Sound can be stopped or retriggered (even if the ALD is switched on)
- Composite or separated frequency/volume output for external circuitry variation
- Hardware-optimized update of frequency and volume to avoid audible artifacts

22.1.1 Description

The following figure provides a functional block diagram of the Sound Generator.

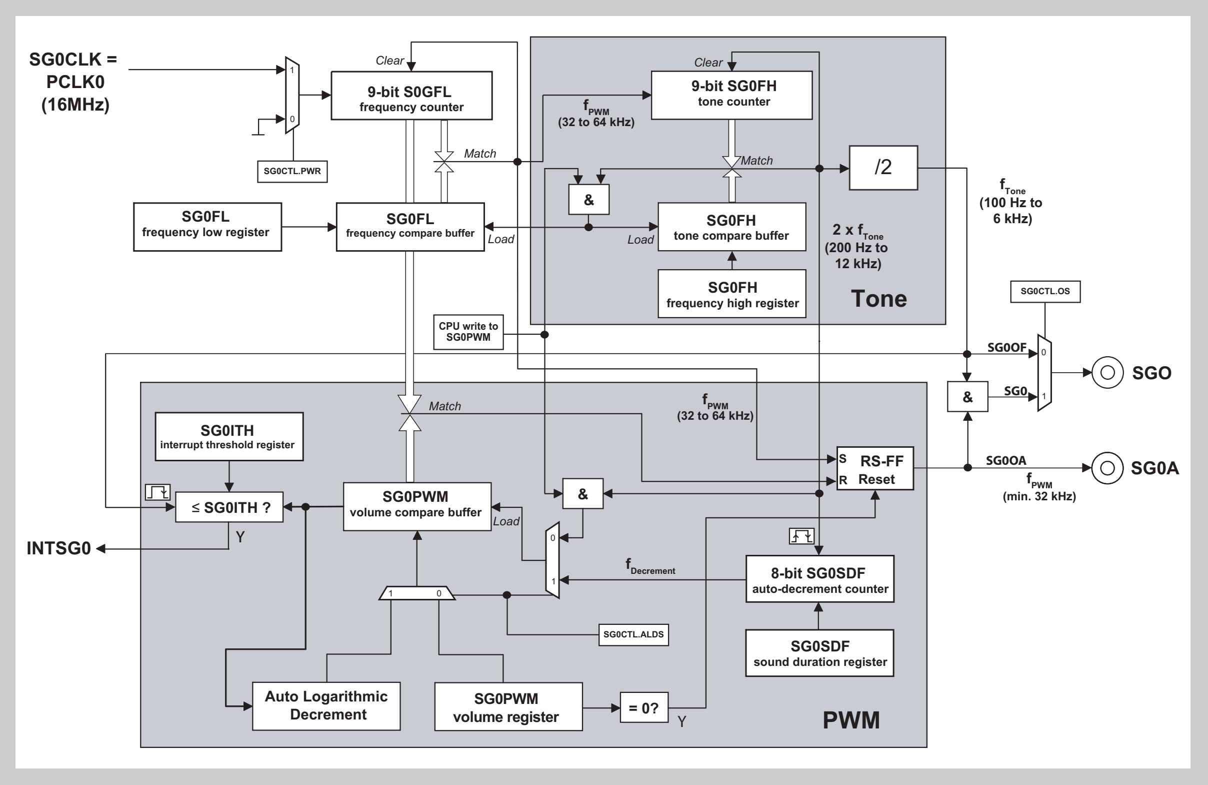


Figure 22-1 Sound Generator block diagram

The Sound Generator's input clock SG0CLK is the 16 MHz clock PCLK0.

Tone generator

The tone generator consists of two up-counters with compare registers. The values written to the frequency registers are automatically copied to compare buffers. The counters are reset to zero when their values match the contents of the associated compare buffers.

The 9-bit counter SG0FL generates a clock with a frequency between 32 KHz and 64 KHz. This clock constitutes the PWM frequency.

It is also the input of the second 9-bit counter SG0FH. The resulting tone signal behind the by-two-divider has a frequency between 100 Hz and 6 KHz and a 50 % duty cycle.

PWM

The PWM modulates the duty cycle according to the desired volume. It is controlled by the volume register SG0PWM. The value written to this register is automatically copied to the associated volume compare buffer.

The PWM continually compares the value of the counter SG0FL with the contents of its volume compare buffer.

The RS flipflop of the PWM is set by the pulses generated by the counter SG0FL. It is reset when the SG0FL counter value matches the contents of the volume buffer. Thus, the PWM output signal can have a duty cycle between 0 % (null volume) and 100 % (maximum volume).

The PWM frequency is above 32 KHz and hence outside the audible range.

Outputs

The Sound Generator is connected to the pins SGO and SGOA. By default, pin SGO provides the tone signal SG0OF and pin SGOA the PWM signal SG0OA that holds the volume (“amplitude”) information.

If bit SG0CTL.OS is set, pin SGO provides the composite signal SG0O that can directly control a speaker circuit.

22.1.2 Principle of operation

The software-controlled registers SG0FL, SG0FH, and SG0PWM are equipped with hardware buffers. The Sound Generator operates on these buffers.

This approach eliminates audible artifacts, because the buffers are only updated in synchronization with the generated tone waveform.

Note This section provides an overview. For details please refer to *“Sound Generator Operation“ on page 746*.

(1) Generation of the tone frequency

The tone frequency is determined by two counters and their associated compare register values. Two counters are necessary to keep the tone pulse and the PWM signal synchronized.

The first counter (SG0FL) provides the input to the second (SG0FH) and also to the PWM. It is used to keep the PWM frequency outside the audio range (above 30 KHz) and within the signal bandwidth of the external sound system (usually below 64 KHz). Its match value defines also the 100 % volume level.

The second counter (SG0FH) generates the tone frequency (100 Hz to 6 KHz).

Note If the target values of the counters SG0FL/SG0FH are changed to generate a different tone frequency, the volume register SG0PWM has to be adjusted to keep the same volume.

(2) Generation of the volume information

The volume information (the “amplitude” of the audible signal) is provided as a high-frequency PWM signal. In composite mode, the PWM signal is ANDed with the tone signal, as illustrated in the following figure.

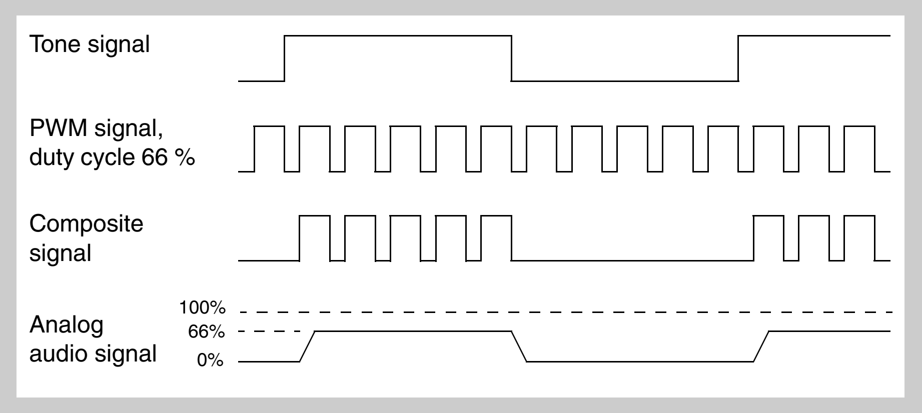


Figure 22-2 Generation of the composite output signal

After low-pass filtering, the analog signal amplitude corresponds to the duty cycle of the PWM signal. Low-pass filtering (averaging) is an inherent characteristic of a loudspeaker system.

The duty cycle can vary between 0 % and 100 %. Its generation is controlled by the counter register SG0FL and the volume register SG0PWM.

When the volume register SG0PWM is cleared, the sound stops immediately.

(3) Automatic fading

The automatic logarithmic decrement function (ALD) provides an automatic volume reduction without CPU interaction.

In regular intervals (related to the tone frequency, selectable via register SG0SDF), the ALD divides the present contents of the volume buffer by 32 (truncated) and subtracts the result from the buffer value. The logarithmic reduction creates the impression of a linear volume reduction in the human ear.

The initial volume that is defined by the contents of the volume register SG0PWM remains unchanged.

(4) Interrupt generation

When the ALD is switched on, the Sound Generator generates the interrupt request INTSG0.

This interrupt signals that the sound volume has decreased to a certain level (set in register SG0ITH). Because the sound duration depends on the tone frequency and the contents of the sound duration register SG0SDF, INTSG0 can be used to indicate “sound is low” or “sound has ended”.

This interrupt is generated only once after the start volume level has been written to SG0PWM.

22.2 Sound Generator Registers

The Sound Generator is controlled by means of the following registers:

Table 22-1 Sound Generator registers overview

Register name	Shortcut	Address
SG0 frequency low register	SG0FL	<base>
SG0 frequency high register	SG0FH	<base> + 2_H
SG0 volume register	SG0PWM	<base> + 4_H
SG0 sound duration factor register	SG0SDF	<base> + 6_H
SG0 control register	SG0CTL	<base> + 7_H
SG0 interrupt threshold register	SG0ITH	<base> + 8_H

Table 22-2 Sound Generator register base address

Module	Base address
SG0	FFFF F5A0_H

(1) SG0CTL - SG0 control register

The 8-bit SG0CTL register controls the operation of the Sound Generator.

Access This register can be read/written in 8-bit or 1-bit units.

Address <base> + 7_H

Initial Value 00_H. This register is cleared by any reset.

7	6	5	4	3	2	1	0
0	0	0	PWR	0	0	OS	ALDS
R	R	R	R/W	R	R	R/W	R/W

Table 22-3 SG0CTL register contents

Bit position	Bit name	Function
4	PWR	Power save mode selection: 0: Clock input switched off (the Sound Generator is disabled and does not operate). 1: Clock input switched on (the Sound Generator is enabled and ready to use).
1	OS	SG0 output mode selection: 0: Selects SGOF and SGOA outputs (frequency and amplitude separated). 1: Selects SGO output (frequency and amplitude mixed).
0	ALDS	Automatic logarithmic decrement of volume (ALD) selection: 0: ALD switched off. 1: ALD activated.

Note Change the contents of this register only when the sound is stopped (register SG0PWM cleared).

(2) SG0FL - SG0 frequency low register

The 16-bit SG0FL register is used to specify the target value for the PWM frequency. It holds the target value for the 9-bit counter SG0FL.

Access This register is can be read/written in 16-bit units. It cannot be written if bit SG0CTL.PWR = 0.

The SG0FL register can also be read/written together with the SG0FH register by 32-bit access via the SG0F register.

Address <base>

Initial Value 0000_H. This register is cleared by any reset.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	Counter SG0FL target value								
R	R	R	R	R	R	R	R/W								

For the calculation of the resulting PWM frequency refer to *“PWM calculations“ on page 749*.

The value written to SG0FL defines also the reference value for the maximum sound amplitude (100% PWM duty cycle). A 100 % duty cycle (continually high) will be generated if the SG0PWM value is higher than the SG0FL value. For details see *“PWM calculations“ on page 749*).

Note

1. The bits SG0FL[15:9] are not used.
2. The maximum value to be written is 510 ($01FE_H$). This yields a PWM frequency of 31.3 KHz. The minimum value to be written depends on the capability of the external circuit. A value of 255 ($00FF_H$) would yield a PWM frequency of 62.5 KHz.
3. The value read from this register does not necessarily reflect the current PWM frequency, because this frequency is determined by the frequency compare buffer value. The buffer might not be updated yet.
   For details see *“Updating the frequency buffer values“ on page 746*.

(3) SG0FH - SG0 frequency high register

The 16-bit SG0FH register is used to specify the final tone frequency. It holds the target value for the 9-bit counter SG0FH.

Access This register is can be read/written in 16-bit units. It cannot be written if bit SG0CTL.PWR = 0.

The SG0FH register can also be read/written together with the SG0FL register by 32-bit access to the SG0FL register via the SG0F register.

Address <base> + 2_H

Initial Value 0000_H. This register is cleared by any reset.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	Counter SG0FH target value								
R	R	R	R	R	R	R	R/W								

For the calculation of the resulting tone frequency refer to *“Tone frequency calculation“ on page 747*.

Note

1. The bits SG0FH[15:9] are not used.
2. Legal values depend on the contents of register SG0FL which defines the frequency of the input pulse. For example: If the counter SG0FL generates a frequency of 32.4 KHz, a value of 161 would generate a tone frequency of 100 Hz.
3. The value read from this register does not necessarily reflect the current tone frequency, because this frequency is determined by the frequency compare buffer value. The buffer might not be updated yet.
For details see *“Updating the frequency buffer values“ on page 746*.

(4) SG0F - SG0 frequency register

The 32-bit register SG0F combines access to the 16-bit registers SG0FL and SG0H. This makes it possible to change the values for the PWM and tone frequency with one write access.

Access This register is can be read/written in 32-bit units. It cannot be written if bit SG0CTL.PWR = 0.

Address <base>

Initial Value $0000\ 0000_H$. This register is cleared by any reset.

31	16	15	0
SG0FH		SG0FL	

(5) SG0PWM - SG0 volume register

The 16-bit register SG0PWM is used to specify the sound volume. It holds the target value for the sound amplitude that is given by the duty cycle of the PWM signal. When the ALD is switched on, this is the start value.

Access This register is can be read/written in 16-bit units. It cannot be written if bit SG0CTL.PWR = 0.

Address <base> + 4_H

Initial Value 0000_H. This register is cleared by any reset.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	Sound volume target value								
R	R	R	R	R	R	R	R/W								

The value written to this register must be considered in conjunction with the contents of register SG0FL. The register SG0FL specifies the maximum value of the counter SG0FL.

For the calculation of the resulting duty cycle refer to *“PWM calculations“ on page 749*.

The setting takes effect after the SG0PWM buffer has been updated (see *“Updating the volume buffer value“ on page 748*).

Note

1. The bits 15:9 are not used.
2. The value read from this register does not necessarily reflect the current volume, because the value of counter SG0FL is compared with the contents of the volume buffer. The buffer might not be updated yet or changed by the ALD function.
3. The value of this register remains unchanged when the ALD is switched on.
4. The sound stops immediately when this register is cleared.

(6) SG0SDF - SG0 sound duration factor register

The 8-bit register SG0SDF is used to specify the duration of the sound when the ALD is switched on. It defines the number of tone signal edges between two successive volume reductions.

Access This register can be read/written in 8-bit or 1-bit units.

Address <base> + 6_H

Initial Value 00_H. This register is cleared by any reset.

7	6	5	4	3	2	1	0
No. of edges between two volume reductions							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

The ALD is synchronized with the tone signal. With this register, the ALD is instructed to reduce the volume at every nth edge (falling or rising) of the tone signal.

The correspondence between the value written to SG0SDF and n is shown in the following table.

Table 22-4 ALD cycle rate

SG0SDF value	n
$0000\ 0000_B$	1
$0000\ 0001_B$	2
...	...
$1111\ 1111_B$	256

Because both edges are counted, the maximum time between two successive volume reductions is 128 times the tone period.

Note Change the contents of this register only when the sound is stopped (register SG0PWM cleared).

(7) SG0ITH - SG0 interrupt threshold register

The 16-bit register SG0ITH is used to specify the volume level for the interrupt request INTSG0.

When the ALD is switched on, the sound volume is stepwise reduced. This is done by reducing the value of the volume buffer. INTSG0 is generated when the value of the volume buffer is equal to or less than the value written to SG0ITH.

INTSG0 is never generated when the ALD is switched off.

Access This register is can be read/written in 16-bit units.

Address <base> + 8_H

Initial Value 0000_H. This register is cleared by any reset.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	Interrupt threshold value								
R	R	R	R	R	R	R	R/W								

To avoid glitches, the INTSG0 interrupt is only generated at a falling edge of the tone signal. If the condition is met at a rising edge, the interrupt will be generated at the next falling edge of the tone signal.

Note

1. The bits 15:9 are not used.
2. Change the contents of this register only when the sound is stopped (register SG0PWM cleared).

When the ALD is switched on, a write access to the SG0PWM volume register starts the comparison of the SG0ITH register contents with the volume buffer. The comparison ends after INTSG0 has been generated.

To revive the comparison, you must first write to SG0PWM. This restarts the tone.

22.3 Sound Generator Operation

This section explains the details of the Sound Generator.

22.3.1 Generating the tone

The tone signal is generated by the compare match signal of the SG0FH counter value with the value of the SG0FH buffer, followed by a by-two-divider. At each compare match, the counter is reset to zero.

Remember that the SG0FH counter is clocked by the output of the SG0FL counter.

(1) Updating the frequency buffer values

The values of the frequency buffers can be changed by writing to the associated frequency registers SG0FL and SG0FH. Both registers can be written together via SG0F.

Changing the value of the SG0FL (equivalent to SG0F[15:0]) register would also yield a change of the PWM frequency, i.e. the sound volume. Therefore it is obligatory to write the correct PWM value to SG0PWM before a new SG0F value is copied to the frequency buffers.

The SG0F register contents is copied to the buffers when the following sequence is detected:

1. CPU write access to SG0PWM register occurred.
2. SG0FH counter value and SG0FH buffer value have matched.
   This match is equivalent to the next edge (rising or falling) of the tone signal.

The following figure shows an example (not to scale).

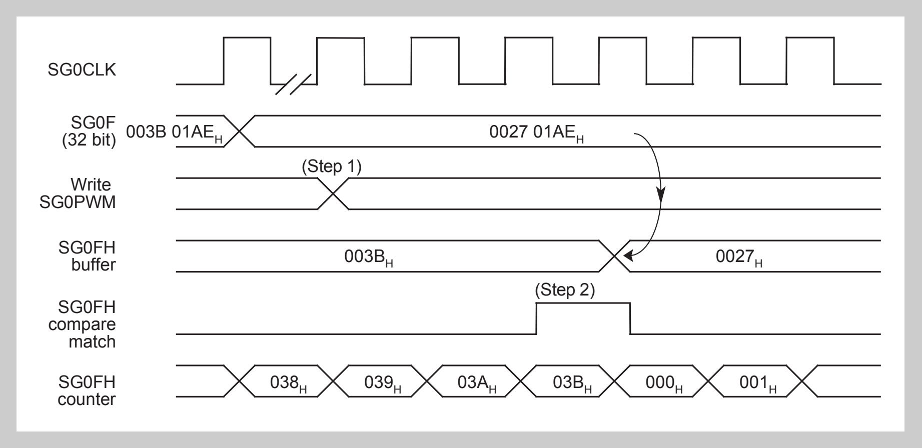


Figure 22-3 Update timing of the frequency buffers

Up to the next match, frequency registers and associated buffers can hold different values. If a 309 Hz tone is generated, as in the above example, the time span between writing to the SG0PWM register and updating the buffer can be up to 3.24 ms.

(2) Tone frequency calculation

The tone frequency can be calculated as:

$$f_{tone} = f_{SG0CLK} / (([\text{SG0FL buffer}] + 1) \times ([\text{SG0FH buffer}] + 1) \times 2)$$

where:

f_{SG0CLK} = frequency of the SG0 input clock

[SG0FL buffer] = contents of the SG0FL buffer

[SG0FH buffer] = contents of the SG0FH buffer

Example If:

- f_{SG0CLK} = 16 MHz
- [SG0FL buffer] = 255 ($00FF_H$) (this yields a PWM frequency of 62.5 KHz)
- [SG0FH buffer] = 32 (0020_H)

then:

- f_{tone} = 947 Hz

Note Note that the buffer contents can differ from the contents of the associated register until the next compare match.

22.3.2 Generating the volume information

The sound volume information is generated by comparing the SG0FL counter value with the contents of the SG0PWM volume buffer. An RS flipflop is set when the counter matches the SG0FL buffer and reset when the counter reaches the value of the volume buffer SG0PWM.

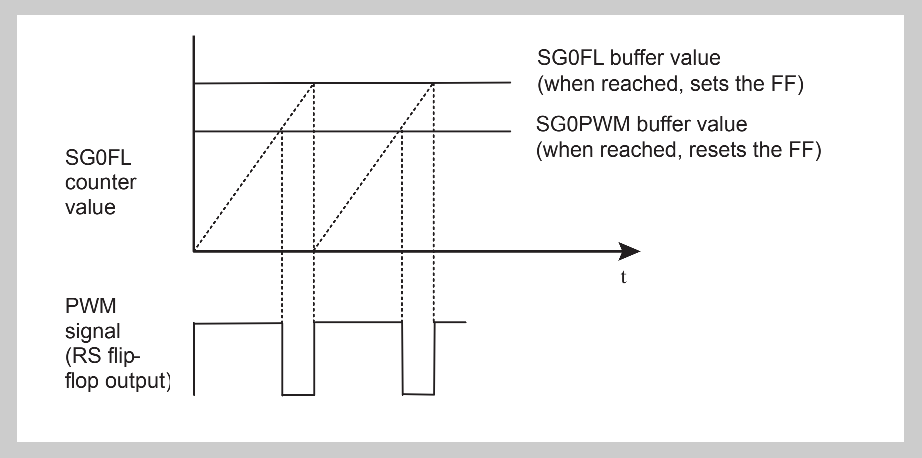

SG0FL buffer value
(when reached, sets the FF)
SG0PWM buffer value
(when reached, resets the FF)
SG0FL counter value
t
PWM signal (RS flip-flop output)

Figure 22-4 PWM signal generation

The duty cycle of the PWM signal is determined by the difference between the contents of the SG0FL counter buffer and the contents of the SG0PWM volume buffer. The larger the difference, the smaller the duty cycle.

The PWM signal is continually high when the value of the volume buffer is higher than the value of the frequency compare buffer.

Note To achieve 100 % duty cycle for all PWM frequencies, SGOFL must not be set to a value above $1FE_H$.

The PWM signal is continually low when the value of the volume buffer is zero — the sound has stopped.

(1) Updating the volume buffer value

The value of the volume compare buffer can be changed by writing to the volume register SG0PWM. It is also changed by the ALD function.

- If the register is cleared by writing 0000_H, the register value is copied to the volume compare buffer with the next rising edge of SG0CLK.
- As a result, the sound stops at the latest after one period of SG0CLK.
- If a non-zero value is written to the register, the buffer is updated with the next falling or rising edge of the tone frequency (match between SG0FH counter value and SG0FH buffer value).

When the ALD is switched on (SG0CTL.ALDS = 1) and no write access to the SG0PWM register occurred, the ALD reduces the contents of the volume buffer gradually.

If SG0PWM is written between two reductions, the new value takes precedence over the ALD, and the volume buffer is updated.

(2) PWM calculations

PWM frequency The PWM frequency is generated by the counter SG0FL. It can be calculated as:

$f_{PWM} = f_{SG0CLK}$ / (([SG0FL buffer] + 1)

where:

f_{SG0CLK} = frequency of the SG0 input clock

[SG0FL buffer] = contents of the SG0FL buffer

Duty cycle The duty cycle of the PWM signal is calculated as follows:

- If [SG0PWM buffer] > [SG0FL buffer]:
  Duty cycle = 100 %
- If 0 ≤ [SG0PWM buffer] ≤ [SG0FL buffer]:
  Duty cycle = [SG0PWM buffer] / ([SG0FL buffer] + 1)

where:

[SG0PWM buffer] = contents of SG0PWM buffer

[SG0FL buffer] = contents of SG0FL buffer

Example If [SG0FL] is set to 240 ($00F0_H$), the following table applies:

Table 22-5 Duty cycle calculation example

[SG0PWM]	Calculation	Duty cycle [%]
$01FF_H$		100
...		100
$00F1_H$	241 / 241	100
$00F0_H$	240 / 241	99.6
$00EF_H$	239 / 241	99.2
...	...	...
0001_H	1 / 241	0.41
0000_H	0 / 241	0

The table shows, how the contents of register SG0FL affects the achievable volume resolution.

(3) Automatic fading

The built-in automatic logarithmic decrement function (ALD) can be used to reduce the volume gradually to zero without CPU intervention. The logarithmic decrease matches the sensitivity of the human ear and creates the impression of a linearly decaying sound.

A sound started with SG0CTL.ALDS = 1 will automatically fade away. The fading can be stopped by writing to the SG0PWM register.

The speed of the volume reduction is controlled by the sound duration factor register SG0SDF. Depending on the value written to SG0SDF, a new amplitude value is calculated at every nth edge (rising or falling) of the tone signal.

The range of n is 1 to 256.

The calculation of the volume reduction uses 13-bit arithmetic and follows below procedure:

```
PWM8[0] = SG0PWM;                       // initial PWM output
OV13[0] = SG0PWM << 5 + 31;             // in 13-bit
for (n=1, n< N+1; n++){
    NV13[n] = OV13[n-1]-(OV13[n-1]>>5); // decrement in 13-bit
    PWM8[n] = NV13[n] >> 5;             // new PWM output
}
```

where:

PWM8[n]:	8-bit PWM output value
OV13[n]:	internal old value in 13-bit
NV13[n]:	internal new value in 13-bit
N:	number of volume reduction steps

Because the SG0PWM register is not affected, the present volume value cannot be read from that register.

The sound stops when the volume buffer value becomes zero.

The number of repetitions depends on the start value set in register SG0PWM. To avoid an initial delay with apparently no effect, the start value shall not exceed the value of register SG0FL by more than 1.

The total sound duration depends on

- the start value,
- the sound duration factor set in register SG0SDF,
- the tone frequency.

Example The subsequent table shows two examples of the sound duration for minimum and maximum tone frequency.

The following settings are assumed:

- f_{SG0CLK} = 16 MHz
- [SG0FL] = 332 (this yields a PWM frequency of 48.048 KHz)
- [SG0PWM] = 333 (100 % volume)
- a) [SG0FH] = 3 (this yields a tone frequency of 6.006 KHz)
- b) [SG0FH] = 240 (this yields a tone frequency of 99.69 Hz)

Table 22-6 Sound duration example

	Tone frequency	Reduced at every tone edge	Reduced at every 2nd tone edge	...	Reduced at every 256th tone edge
Sound duration [roughly sec]	6006 Hz	0.018	0.035	...	4.58
	99.69 Hz	1.08	2.15	...	275

22.4 Sound Generator Application Hints

This section provides supplementary programming information.

22.4.1 Initialization

To enable the Sound Generator, set SG0CTL.PWR to 1. This connects the SG0 to the clock SG0CLK.

Check bit SG0CTL.OS.

When SG0CTL.OS is 0, the signal at pin SGO is a symmetrical square waveform with the frequency f_{tone}. When SG0CTL.OS is 1, the signal at pin SGO is composed of the tone signal and PWM pulses.

The frequency data registers SG0FL and SG0FH provide the buffer values for the counters. The combined value represents the frequency of the tone.

22.4.2 Start and stop sound

The sound is started by writing a non-zero value to the volume register SG0PWM.

Before starting the sound, all other register settings must be made.

The sound is stopped by writing 0000_H to the volume register SG0PWM. The sound is stopped regardless of the current value of amplitude output or frequency output. Thus, the sound can be stopped quickly, even if a very low sound frequency is chosen.

When the ALD is switched on, the sound stops automatically when the contents of the volume buffer reaches zero.

22.4.3 Change sound volume

The sound volume is changed by writing a new value to register SG0PWM.

The new volume takes effect with the next edge of the tone pulse (rising or falling).

Note When the ALD is switched on, the current volume value cannot be read from register SG0PWM.

22.4.4 INTSG0 interrupt

The interrupt INTSG0 is only generated when the ALD is active (SG0CTL.ALDS = 1). INTSG0 is generated when the value of the volume buffer is equal to or less than the value written to SG0ITH.

This can be used to reconfigure the Sound Generator when a certain volume level is reached.

The interrupt does not stop the ALD sound.

22.4.5 Constant sound volume

A sound started with SG0CTL.ALDS = 0 is output with the volume value written to SG0PWM. The sound is output continually and does not stop automatically. It has to be stopped by writing 0000_H to the SG0PWM register.

22.4.6 Generate special sounds

To generate special sounds (like blinker clicks etc.), frequency and volume can be changed simultaneously.

To change the frequency of a sound that has already started:

1. Write to frequency register SG0FL in 32-bit mode (or to SG0FL and SG0FH separately in 16-bit mode).
2. Write to volume register SG0PWM.

Chapter 23 Power Supply Scheme

The microcontroller has general power supply pins for its core, internal memory and peripherals. These pins are connected to internal voltage regulators. The microcontroller also has dedicated power supply pins for certain I/O modules. These pins provide the power for the I/O operations.

23.1 Overview

The following table gives the naming convention of the pins:

Table 23-1 Naming convention of power supply pins

Dedicated function		V_{DD} or V_{SS}	5	n
<none>	CPU core, internal memory and peripherals	• VDD: Voltage Drain Drain • VSS: Voltage for Substrate and Source	level 5 V (nominative)	instance number
A	A/D Converter			
B	Standard I/O buffer			
SM	Stepper Motor Controller/Driver I/O			

The following pins belong to the Power Supply Scheme:

Table 23-2 Power supply pins

Pin	Connected to
VDD50 / VSS50	CPU core Pin pair is connected to voltage regulator 0.
REGC0	Capacitor for voltage regulator 0 for pin pair VDD50 / VSS50.
AVDD / AVSS	A/D Converter (power supply)
AVREF	A/D Converter (reference input level)
BVDD5n / BVSS5n	I/O buffer LCD Controller/Driver I/O (n = 0, 1)
SMVDD5n / SMVSS5n	Stepper Motor Controller/Driver (n = 0, 1)

Note For electrical characteristics refer to the Data Sheet.

23.2 Description

Figure 23-1 gives an overview of the allocation of power supply pins. Their functional assignment is depicted in more detail in *Figure 23-2*.

Note The diagrams do not show the exact pin location.

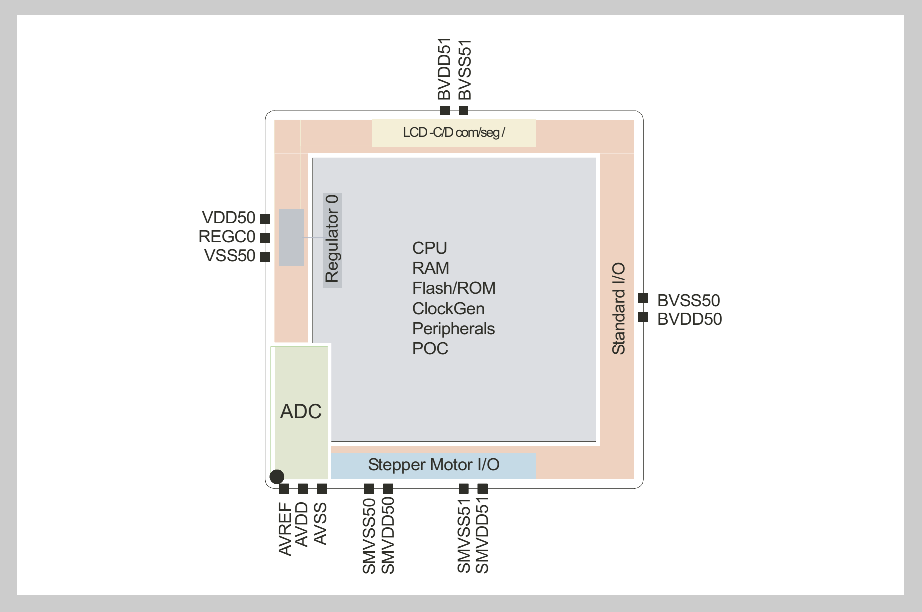


Figure 23-1 Power supply pins

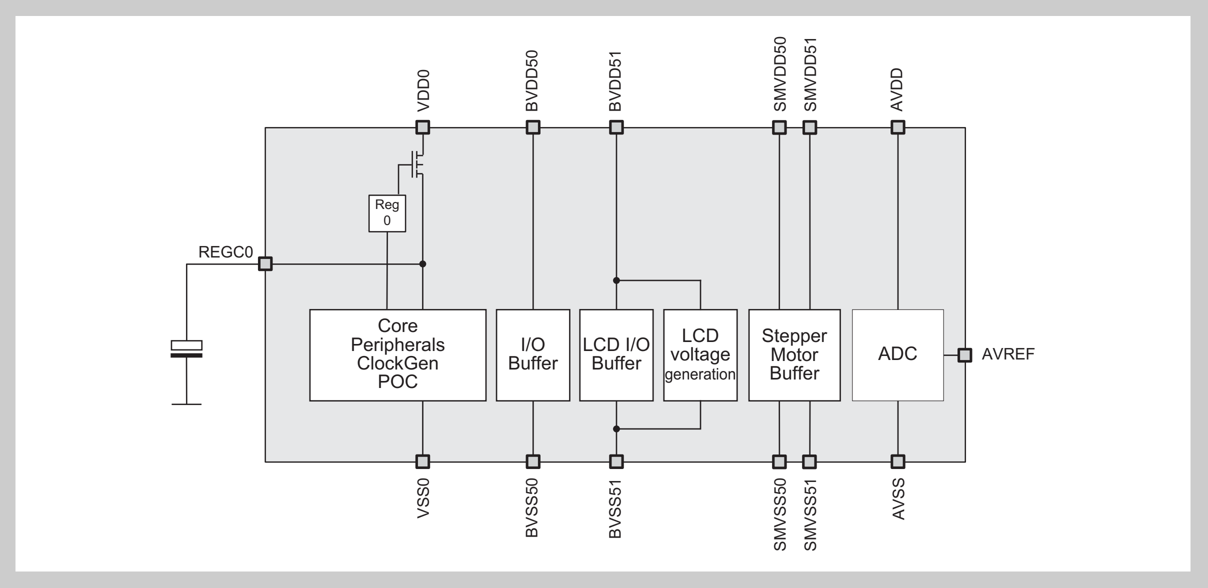


Figure 23-2 Functional assignment of power supply pins

23.3 Voltage regulators

The on-chip voltage regulators generate the voltages for the internal circuitry (CPU core, clock generation circuit and peripherals), refer to *Figure 23-2*.

The regulators operate per default in all operation modes (normal operation, HALT, IDLE, STOP, WATCH, Sub-WATCH, and during RESET).

During power save modes the voltage regulators can be optionally disabled by setting the STBCTL register (refer to *“Control registers for power save modes“ on page 130*).

Note To stabilize the output voltage of the regulator, connect a capacitor to the REGCn pin. Refer to the Data Sheet.

Chapter 24 Reset

Several system reset functions are provided in order to initialize hardware and registers.

24.1 Overview

Features summary

A reset can be caused by the following events:

- External reset signal $\overline{\text{RESET}}$
  Noise in the external reset signal is eliminated by an analog filter.
- Power-On-Clear (internal signal RESPOC)
- Overflow of the Watchdog Timer (internal signal RESWDT)
- Main or sub-oscillator fails (internal signals RESCMM, RESCMS)
- Software reset (internal signal RESSW)

As output, the reset function provides two internal reset signals:

- SYSRES (system reset)
- SYSRESWDT (Watchdog Timer reset)

24.1.1 General reset performance

The following figure shows the signals involved in the reset function:

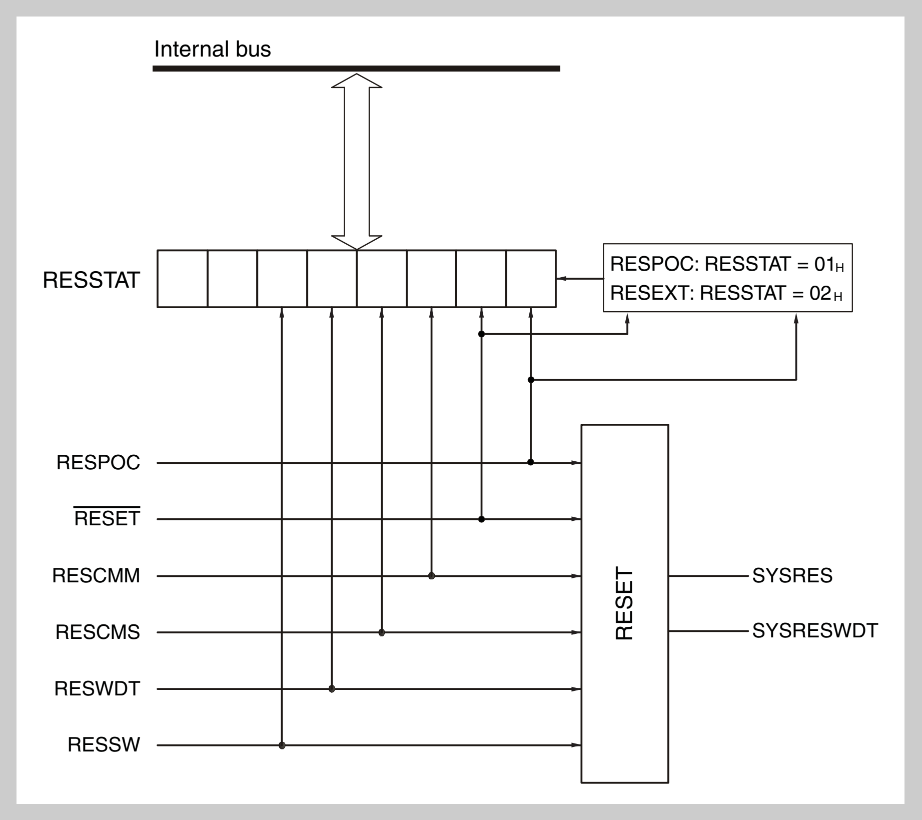


Figure 24-1 Reset function signal diagram

All resets are applied asynchronously. That means, resets are not synchronized to any internal clock. This ensures that the microcontroller can be kept in reset state even if all internal clocks fail to operate.

The reset function provides two internal reset signals:

- System reset SYSRES
  SYSRES is activated by all reset sources.
- Watchdog reset SYSRESWDT
  SYSRESWDT is activated by Power-On-Clear and external $\overline{\text{RESET}}$ only.

Both resets provoke different reset behaviour of the Watchdog Timer. For details refer to the *“Watchdog Timer (WDT)“ on page 395*.

(1) Variable reset vector(flash memory devices only)

The flash memory devices allow to program the start address of the user’s program, instead of starting at address 0000 0000_H. The variable reset vector is stored in the extra area of the flash memory and can be written by an external flash programmer or in self-programming mode.

(2) Hardware status

With each reset function the hardware is initialized (including the watchdog). When the reset status is released, program execution is started.

The following table describes the status of the clocks during reset and after reset release. Note that the clock status "operates" does not inevitably mean that any function using this clock source operates as well. The function may additionally require to be enabled by other means.

Table 24-1 Hardware status during and after reset

Item	During reset	After reset
Main oscillator	Stops oscillation	Stopped[a]
Sub oscillator	Operates	Starts oscillation
Internal oscillator	Operates	Starts oscillation The internal oscillator clock is the default clock source after reset release.
SSCG clock	Stops operation	Stopped[a]
PLL clock	Stops operation	Stopped[a]
CPU system clock (VBCLK)	Stops operation	Starts oscillation based on the internal oscillator clock.
CPU	Initialized	Program execution starts after oscillation stabilization time.
Watchdog Timer (WDTCLK)	Stops operation	Starts operation based on internal oscillator clock
Watch Timer (WTCLK)	Stops operation	Starts operation based on internal oscillator clock
Peripheral clocks	Stop operation	• PCLK0–2: operating based on internal-osc • PCLK3-15: stopped • SPCLK0–2: operating based on internal-osc • SPCLK3-15: stopped
On-chip peripheral functions	Stop operation	Depends on availability of peripheral clock and default status of the peripheral function.
I/O pins (port/alternative function pins)	All pins are in input port mode. See chapter *“Pin Functions“ on page 29* for a description.	

[a] The main oscillator is started by the internal firmware. However the application software has to ensure stable main oscillation before utilizing this clock for any purpose. SSCG and PLL must be started by the application software. Assure also here that the stabilization time has passed. See chapter *“Clock Generator“ on page 100* for details.

(3) Register status

With each reset function the registers of the CPU, internal RAM, and on-chip peripheral I/Os are initialized.

Since after reset the internal firmware is processed, some resources hold a different value as after reset, when the user's program is started. After a reset, make sure to set the registers to the values needed within your program.

Table 24-2 Initial values of CPU and internal RAM after reset

On-chip hardware		Register name	Initial value	
			After Reset	**At start of user's program**
CPU	Program registers	General-purpose register (r0)	$0000\ 0000_H$	$0000\ 0000_H$
		General-purpose registers (r1 to r31)	Undefined	Undefined
		Program counter (PC)	$0000\ 0000_H$	Variable reset vector programmed to flash extra area
	System registers	Status save registers during interrupt (EIPC, EIPSW)	Undefined	Undefined
		Status save registers during non-maskable interrupt (NMI) (FEPC, FEPSW)	Undefined	Undefined
		Interrupt cause register (ECR)	$0000\ 0000_H$	$0000\ 0000_H$
		Program status word (PSW)	$0000\ 0020_H$	• $0000\ 0020_H$: if no security flags or variable reset vector are set • $0000\ 0021_H$: else
		Status save registers during CALLT execution (CTPC, CTPSW)	Undefined	Undefined
		Status save registers during exception/debug trap (DBPC, DBPSW)	Undefined	Undefined
		CALLT base pointer (CTBP)	Undefined	Undefined
Internal RAM	After power-on	After Power-On-Clear reset the entire RAM contents is undefined.	Undefined	Undefined
	After $\overline{\text{RESET}}$	If a $\overline{\text{RESET}}$ occurs while writing to a RAM memory block, the contents of that RAM memory block may be corrupted. All other RAM memory blocks are not affected. Refer also to the note below the table.	All data in previous state	• $03FF\ 0000_H$ - $03FF\ 07FF_H$: undefined All other data in previous state or undefined (refer to note below).
	After any other reset	Any internal generated reset does not change the RAM contents.	All data in previous state	• $03FF\ 0000_H$ - $03FF\ 07FF_H$: undefined All other data in previous state.
Peripherals		Macro internal registers	The reset values of the various registers are given in the chapters of the peripheral functions	

Note In the table above, “Undefined” means either undefined at the time of a power-on reset, or undefined due to data destruction when the falling edge of the external $\overline{\text{RESET}}$ signal corrupts an ongoing RAM write access.
The internal RAM of the microcontroller comprises several separate RAM blocks. In case writing to one RAM block while a reset occurs the contents of only this RAM block may be corrupted. The other RAM blocks remain unchanged.

24.1.2 Reset at power-on

The Power-On-Clear circuit (POC) permanently compares the power supply voltage V_{DD} with an internal reference voltage (V_{IP}). It ensures that the microcontroller only operates as long as the power supply exceeds a well-defined limit.

When the power supply voltage falls below the internal reference voltage ($V_{DD} < V_{IP}$), the internal reset signal RESPOC is generated.

After Power-On-Clear reset, the RESSTAT register is cleared and the RESSTAT.RESPOC bit is set (RESSTAT = 01_H, refer also to *“RESSTAT - Reset source flag register“ on page 764* for the interaction between Power-On-Clear and external $\overline{\text{RESET}}$). The system reset signals SYSRES and SYSRESWDT are generated.

Note
1. Depending on the supply voltage drop rate it may be required to apply an external $\overline{\text{RESET}}$ signal additionally in order to avoid microcontroller operation out of the specified operating conditions. For detailed electrical characteristics refer to the Data Sheet.
2. POC shares the reference voltage supply with the power regulators.

The following figure shows the timing when a reset is performed at power-on.

The Power-On-Clear function holds the microcontroller in reset state as long as the power supply voltage does not exceed the threshold level V_{IP}.

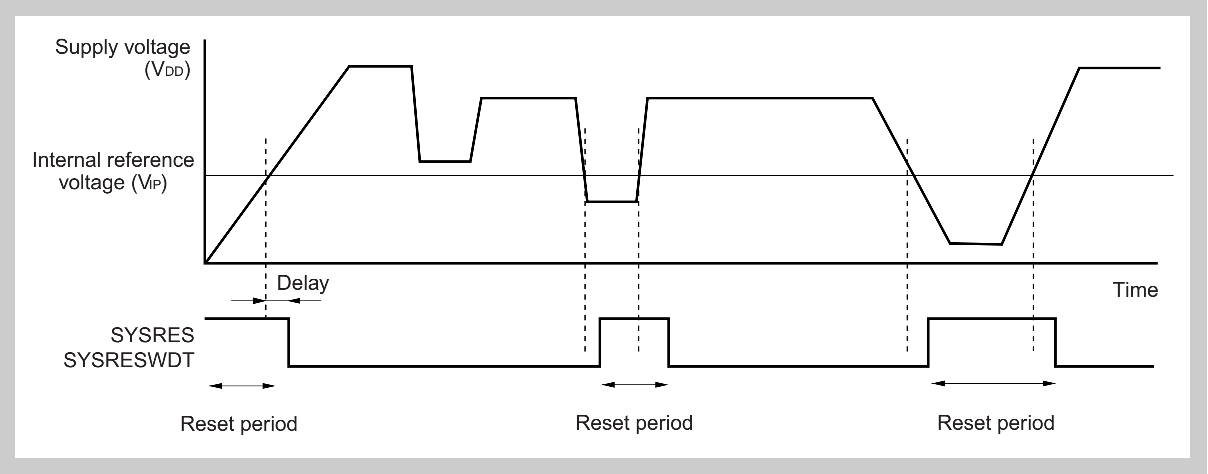


Figure 24-2 Timing of internal reset signal generation by Power-On-Clear circuit

24.1.3 External $\overline{\text{RESET}}$

Reset is performed when a low level signal is applied to the $\overline{\text{RESET}}$ pin.

The reset status is released when the signal applied to the $\overline{\text{RESET}}$ pin changes from low to high.

After the external $\overline{\text{RESET}}$ is released, the RESSTAT register is cleared and the RESSTAT.RESEXT bit is set (RESSTAT = 02_H, refer also to *“RESSTAT - Reset source flag register“ on page 764* for the interaction between Power-On-Clear and external $\overline{\text{RESET}}$). The system reset signals SYSRES and SYSRESWDT are generated.

The $\overline{\text{RESET}}$ pin incorporates a noise eliminator, which is applied to the reset signal $\overline{\text{RESET}}$. To prevent erroneous external reset due to noise, it uses an analog filter. Even if no clock is active in the controller the external $\overline{\text{RESET}}$ can keep the controller in reset state.

Note The internal system reset signals SYSRES and SYSRESWDT keep their active level for at least four system clock cycles after the $\overline{\text{RESET}}$ pin is released.

The following figure shows the timing when an external reset is performed. It explains the effect of the noise eliminator. The noise eliminator uses the analog delay to prevent the generation of an external reset due to noise.

The analog delay is caused by the analog input filter. The filter regards pulses up to a certain width as noise and suppresses them. For the minimum $\overline{\text{RESET}}$ pulse width refer to the Data Sheet.

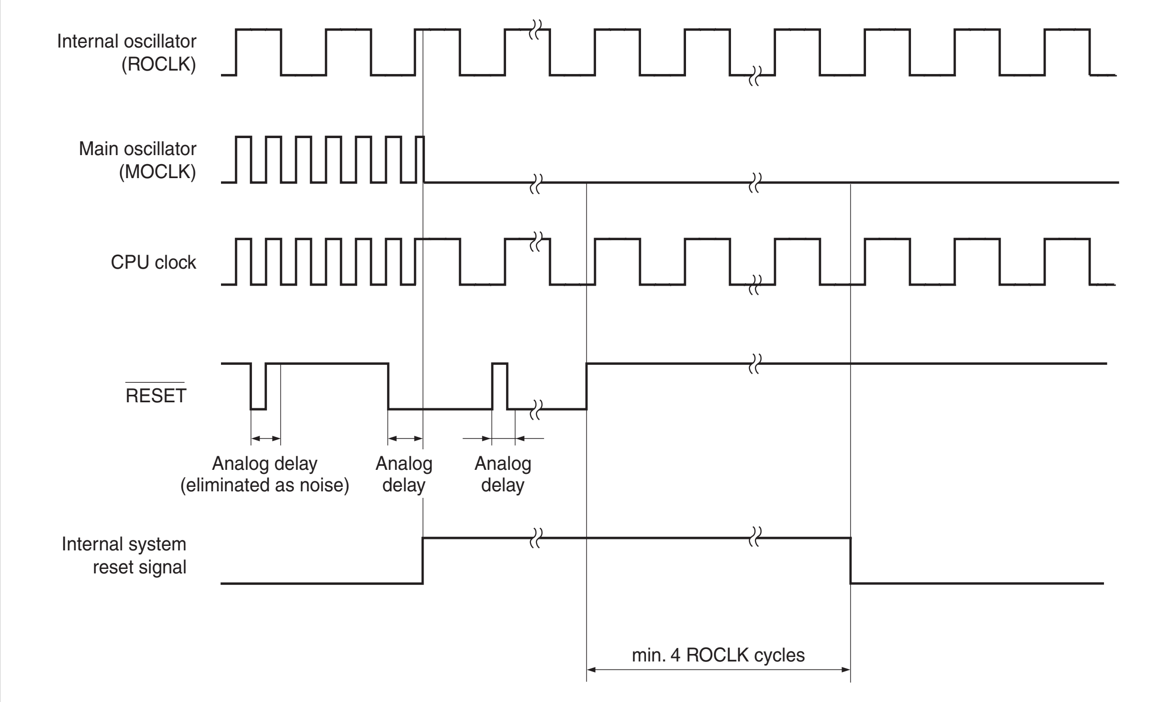

Internal oscillator (ROCLK)
Main oscillator (MOCLK)
CPU clock
RESET
Analog delay (eliminated as noise)
Analog delay
Analog delay
Internal system reset signal
min. 4 ROCLK cycles

Figure 24-3 External $\overline{\text{RESET}}$ timing

24.1.4 Reset by Watchdog Timer

The Watchdog Timer can be configured to generate a reset if the watchdog time expires. After watchdog reset, the RESSTAT.RESWDT bit is set. The system reset signal SYSRES is generated.

After Watchdog Timer overflow, the reset status lasts for a specific time. Then the reset status is automatically released.

24.1.5 Reset by Clock Monitor

The two Clock Monitors generate a reset when either the main oscillator or the sub-oscillator fails. After a Clock Monitor reset, the corresponding bit (RESSTAT.RESCMM or RESSTAT.RESCMS) is set. The system reset signal SYSRES is generated.

After a Clock Monitor reset, the reset status lasts for a specific time. Then the reset status is automatically released.

24.1.6 Software reset

Software reset is generated by two consecutive write accesses:

1. Suspend write protection of RESSWT register:
   byte write access to register RESCMD (content of the data is not relevant)
2. Generate software reset:
   byte write access to register RESSWT (content of the data is not relevant)

These two steps are required in order to prevent an unintentional software reset.

The registers RESCMD and RESSWT are always read as 00_H.

After software reset, the RESSTAT.RESSW bit is set. The system reset signal SYSRES is generated.

24.2 Reset Registers

The reset functions are controlled and operated by means of the following registers:

Table 24-3 Reset function registers overview

Register name	Shortcut	Address
Reset source flag register	RESSTAT	FFFF FF20$_H$
Software reset register	RESSWT	FFFF FF22$_H$
Software reset enable register	RESCMD	FFFF FF24$_H$
Reset status register	RES	FFFF FF26$_H$

(1) RESSTAT - Reset source flag register

The 8-bit RESSTAT register contains information about which type of resets occurred since the last Power-On-Clear or external $\overline{\text{RESET}}$ or after the last software clear of the register.

Each following reset condition sets the corresponding flag in the register. For example, if a Power-On-Clear reset is finished and then a Watchdog Timer reset occurs, the RESSTAT reads xxx1 0001$_B$.

Access The register can be read/written in 8-bit units.

Address FFFF FF20$_H$

Initial Value Power-On-Clear reset sets this register to 01$_H$.
External $\overline{\text{RESET}}$ sets this register to 02$_H$.

7	6	5	4	3	2	1	0
X	X	RESSW	RESWDT	RESCM2	RESCM1	RESEXT	RESPOC
R	R	R/W[a]	R/W[a]	R/W[a]	R/W[a]	R/W[a]	R/W[a]

[a] Any write clears this register, independent of the data written.

Table 24-4 RESSTAT register contents (1/2)

Bit position	Bit name	Function
5	RESSW	Software reset 0: Not generated. 1: Generated.
4	RESWDT	Reset by Watchdog Timer 0: Not generated. 1: Generated.
3	RESCM2	Reset by Clock Monitor of sub oscillator 0: Not generated. 1: Generated.

Table 24-4 RESSTAT register contents (2/2)

Bit position	Bit name	Function
2	RESCM1	Reset by Clock Monitor of main oscillator 0: Not generated. 1: Generated.
1	RESEXT	External $\overline{\text{RESET}}$ 0: Not generated. 1: Generated.
0	RESPOC	Reset at Power-On-Clear 0: Not generated. 1: Generated.

Note If clearing this register by writing and flag setting (occurrence of reset) conflict, flag setting takes precedence.

RESPOC and RESEXT Both Power-On-Clear and external $\overline{\text{RESET}}$ set RESSTAT to different initial states.

- Power-On-Clear reset sets RESSTAT = 01_H
- External $\overline{\text{RESET}}$ sets RESSTAT = 02_H

Special caution is required if both reset events are active concurrently:

- If the Power-On-Clear reset is longer active than the external $\overline{\text{RESET}}$: RESSTAT = 01_H. That means RESSTAT indicates only the occurrence of the Power-On-Clear reset.
- If the external $\overline{\text{RESET}}$ is longer active than the Power-On-Clear reset: RESSTAT = 02_H.That means RESSTAT indicates only the occurrence of the external $\overline{\text{RESET}}$.
- If the Power-On-Clear reset and external $\overline{\text{RESET}}$ has been released simultaneously: RESSTAT = 03_H. That means RESSTAT indicate the occurrence of both reset events.

All other reset events just set their respective bit in RESSTAT and do not change the others.

(2) RESSWT - Software reset register

Write operation to the 8-bit RESSWT register generates a software reset. The content of data written to RESSWT is not relevant.

Writing to this register is protected by a special sequence of instructions. To enable write access to RESSWT, first write to RESCMD. Please refer to *“Write Protected Registers“ on page 96* for details.

The register is always read as 00_H.

Access This register can only be written in 8-bit units.

Address FFFF FF22$_H$

Initial Value 00_H

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0
W	W	W	W	W	W	W	W

(3) RESCMD - Software reset enable register

Immediately after writing data to the 8-bit RESCMD register, write access to the RESSWT register is enabled. The content of data written to RESCMD register is not relevant.

The register is always read as 00_H.

Access This register can only be written in 8-bit units.

Address FFFF $FF24_H$

Initial Value 00_H

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0
W	W	W	W	W	W	W	W

Caution In case a high level programming language is used, make sure that the compiler translates the two write instructions to RESCMD and RESSWT into two consecutive assembler “store” instructions.

(4) RES - Reset status register

The 8-bit RES register indicates the status of a write attempt to a register protected by RESCMD (see also *“RESCMD - Software reset enable register“ on page 766*).

The register is always read as 00_H.

Access This register can be read/written in 8-bit units.

Address FFFF $FF26_H$

Initial Value 00_H

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	RERR
R[a]	R[a]	R[a]	R[a]	R[a]	R[a]	R[a]	R/W

[a] These bits may be written, but write is ignored.

Table 24-5 RES register contents

Bit position	Bit name	Function
0	RERR	Write error status: 0: Write access was successful. 1: Write access failed. You can clear this register by writing 0 to it. Setting this register to 1 by software is not possible.

Note RES.RERR is set, if a write access to register RESMD is not directly followed by a write access to one of the write-protected registers.

Appendix A Registers Access Times

This chapter provides formulas to calculate the access time to registers, which are accessed via the peripheral I/O areas.

All accesses to the peripheral I/O areas are passed over to the NPB bus via the VSB - NPB bus bridge BBR. Read and write access times to registers via the NPB depend on the register, the system clock VBCLK and the setting of the VSWC register.

The CPU operation during an access to a register via the NPB depends also on the kind of peripheral I/O area:

- Fixed peripheral I/O area

  During a read or write access the CPU operation stops until the access via the NPB is completed.

- Programmable peripheral I/O area

  During a read access the CPU operation stops until the read access via the NPB is completed.

  During a write access the CPU operation continues operation, provided any preceded NPB access is already finished. If a preceded NPB access is still ongoing the CPU stops until this access is finished and the NPB is cleared.

The following formulas are given to calculate the access times T_a, when the CPU reads from or writes to special function registers via the NPB bus.

The access time depends

- on the CPU system clock frequency f_{VBCLK}
- on the setting of the internal peripheral function wait control register VSWC, which determines the address set up wait SUWL = VSWC.SUWL and data wait VSWL = VSWC.VSWL (refer to *"VSWC - Internal peripheral function wait control register" on page 271* for the correct values for a certain CPU system clock VBCLK)
- for some registers on the clock frequency applied to the module

Note "ru[...]" in the formulas mean "round up" the calculated value of the term in squared brackets.

All formulas calculate the maximum access time.

CPU access For calculating the access times for CPU accesses 1 VBLCK period time 1/ f_{VBCLK} has to be added to the results of the formulas.

A.1 Timer P

Register **TPnCCR0, TPnCCR1**

Access R

Formula $T_a = \left\{ SUWL + VSWL + 3 + ru\left[\frac{f_{VBCLK}}{(2 + VSWL) \cdot f_{PCLK0}} + 1\right] \cdot (2 + VSWL) \right\} \cdot \frac{1}{f_{VBCLK}}$

Access W

Formula $T_a = \left\{ SUWL + VSWL + 3 + ru\left[\frac{5 \cdot f_{VBCLK}}{(2 + VSWL) \cdot f_{PCLK0}} + 1\right] \cdot (2 + VSWL) \right\} \cdot \frac{1}{f_{VBCLK}}$

Register **TPnCNT**

Access R

Formula $T_a = \left\{ SUWL + VSWL + 3 + ru\left[\frac{f_{VBCLK}}{(2 + VSWL) \cdot f_{PCLK0}} + 1\right] \cdot (2 + VSWL) \right\} \cdot \frac{1}{f_{VBCLK}}$

Access W

Formula $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

Register **all other**

Access R/W (no write access during timer operation)

Formula $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

A.2 Timer Z

Register **TZnCNT0**

Access R

Formula $T_a = (SUWL + 3 \cdot VSWL + 6) \cdot \frac{1}{f_{VBCLK}} + \frac{\dot{4}, 5}{f_{PCLK2}}$

Register **TZnCNT1**

Access R

Formula $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

Register **TZnR**

Access R

Formula $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

Access W

Formula $T_a = (SUWL + 3 \cdot VSWL + 6) \cdot \frac{1}{f_{VBCLK}} + \frac{4, 5}{f_{PCLK2}}$

Register **TZnCTL**

Access R/W

Formula $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

A.3 Timer G

Register **TMGn0, TMGn1**

Access R

Formula $T_a = \left\{SUWL + VSWL + 3 + ru\left[\frac{f_{VBCLK}}{(2 + VSWL) \cdot f_{SPCLK0}} + 1\right] \cdot (2 + VSWL)\right\} \cdot \frac{1}{f_{VBCLK}}$

Access W (no write access during timer operation)

Formula $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

Register **GCCn[5:0]**

Access R

Formula $T_a = \left\{SUWL + VSWL + 3 + ru\left[\frac{f_{VBCLK}}{(2 + VSWL) \cdot f_{SPCLK0}} + 1\right] \cdot (2 + VSWL)\right\} \cdot \frac{1}{f_{VBCLK}}$

Access W (for GCCn0 and GCCn5 no write access during timer operation)

Formula

- for multiple write within 7 SPCLK0 periods

  $T_a = \left\{SUWL + VSWL + 3 + ru\left[\frac{f_{VBCLK}}{(2 + VSWL) \cdot f_{SPCLK0}} + 1\right] \cdot (2 + VSWL)\right\} \cdot \frac{1}{f_{VBCLK}}$

- for single write within 7 SPCLK0 periods

  $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

Register **all other**

Access R/W (no write access during timer operation)

Formula $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

A.4 Watch Timer

Register **WTnCNT1**

Access R

Formula $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

Register **WTnR**

Access R

Formula $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

Access W

Formula $T_a = (SUWL + 3 \cdot VSWL + 7) \cdot \frac{1}{f_{VBCLK}}$

Register **CR00**

Access Read-Modify-Write

Formula $T_a = (SUWL + 3 \cdot VSWL + 7) \cdot \frac{1}{f_{VBCLK}}$

Register **all other**

Access R/W

Formula $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

A.5 Watch Calibration Timer

Register **CR01**

Access R/W

Formula $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

Access Read-Modify-Write

Formula $T_a = (SUWL + 3 \cdot VSWL + 7) \cdot \frac{1}{f_{VBCLK}}$

Register **all other**

Access R/W

Formula $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

A.6 Watchdog Timer

Register all

Access R/W

Formula $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

A.7 Asynchronous Serial Interface (UARTA)

Register **all**

Access R/W

Formula $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

A.8 Clocked Serial Interface (CSIB)

Register **all**

Access R/W

Formula $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

A.9 I^{2}C Bus

Register **IICSn**

Access R

Formula $T_a = (SUWL + 3 \cdot VSWL + 7) \cdot \frac{1}{f_{VBCLK}}$

Register **all other**

Access R/W

Formula $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

A.10 CAN Controller

Register **CnMDATA[7:0]m**

Access R

Formula

$$T_a = \left\{SUWL + VSWL + 3 + ru\left[4 \cdot \frac{\frac{f_{VBCLK}}{f_{CANMOD}} + 1}{2 + VSWL}\right] \cdot (2 + VSWL)\right\} \cdot \frac{1}{f_{VBCLK}}$$

Access 8-bit Write

Formula

$$T_a = \left\{SUWL + VSWL + 3 + ru\left[5 \cdot \frac{\frac{f_{VBCLK}}{f_{CANMOD}} + 1}{2 + VSWL}\right] \cdot (2 + VSWL)\right\} \cdot \frac{1}{f_{VBCLK}}$$

Access 16-bit Write

Formula

$$T_a = \left\{SUWL + VSWL + 3 + ru\left[3 \cdot \frac{\frac{f_{VBCLK}}{f_{CANMOD}} + 1}{2 + VSWL}\right] \cdot (2 + VSWL)\right\} \cdot \frac{1}{f_{VBCLK}}$$

Register **CnRGPT, CnTGPT, CnLIPT, CnLOPT**

Access R

Formula

$$T_a = \left\{SUWL + VSWL + 3 + ru\left[4 \cdot \frac{\frac{f_{VBCLK}}{f_{CANMOD}} + 1}{2 + VSWL}\right] \cdot (2 + VSWL)\right\} \cdot \frac{1}{f_{VBCLK}}$$

Register **all other**

Access R/W

Formula

$$T_a = \left\{SUWL + VSWL + 3 + ru\left[2 \cdot \frac{\frac{f_{VBCLK}}{f_{CANMOD}} + 1}{2 + VSWL}\right] \cdot (2 + VSWL)\right\} \cdot \frac{1}{f_{VBCLK}}$$

A.11 A/D Converter

Register **ADAM0[2:0], ADACR0n**

Access R

Formula $$T_a = \left\{SUWL + VSWL + 3 + ru\left[\frac{2 \cdot f_{VBCLK}}{(2 + VSWL) \cdot f_{SPCLK0}} + 1\right] \cdot (2 + VSWL)\right\} \cdot \frac{1}{f_{VBCLK}}$$

Access W

Formula $$T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$$

Register **all other**

Access R/W

Formula $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

A.12 Stepper Motor Controller/Driver

Register **MCNTCn[1:0], MCMPCnk**

Access R

Formula $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

Access W

Formula $T_a = \left\{SUWL + VSWL + 3 + ru\left[\frac{2 \cdot f_{VBCLK}}{(2 + VSWL) \cdot f_{SPCLK1}} + 1\right] \cdot (2 + VSWL)\right\} \cdot \frac{1}{f_{VBCLK}}$

Register **all other**

Access R/W

Formula $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

A.13 LCD Controller/Driver

Register **all**

Access R/W

Formula $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

A.14 Sound Generator

Register **SG0FL, SG0FH, SG0PWM**

Access R

Formula $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

Access W

Formula $T_a = (SUWL + 3 \cdot VSWL + 6) \cdot \frac{1}{f_{VBCLK}} + \frac{2}{f_{PCLK0}}$

Register **all other**

Access R/W

Formula $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

A.15 Clock Generator

Register **CGSTAT**

Access R

Formula $T_a = (SUWL + 3 \cdot VSWL + 7) \cdot \frac{1}{f_{VBCLK}}$

Access W

Formula $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

Register **all other**

Access R/W

Formula $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

A.16 All other Registers

Register **all**

Access R/W

Formula $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

Appendix B Special Function Registers

The following tables list all registers that are accessed via the NPB (Peripheral bus). The registers are called “special function registers” (SFR).

Table B-1 lists all CAN special function registers. The addresses are given as offsets to programmable peripheral base address (refer to *“CAN module register and message buffer addresses“ on page 572*.

The tables list all registers and do not distinguish between the different derivatives.

B.17 CAN Registers

The CAN registers are accessible via the programmable peripheral area.

Table B-1 CAN special function registers (1/2)

Address offset	Register name	Shortcut	1	8	16	32	Initial value
0x000	CAN0 Global Macro Control register	C0GMCTRL	-	-	R/W	-	0x0000
0x000	CAN0 Global Macro Control register low byte	C0GMCTRLL	R/W	R/W	-	-	0x00
0x001	CAN0 Global Macro Control register high byte	C0GMCTRLH	R/W	R/W	-	-	0x00
0x002	CAN0 Global Macro Clock Selection register	C0GMCS	R/W	R/W	-	-	0x0F
0x006	CAN0 Global Macro Automatic Block Transmission register	C0GMABT	-	-	R/W	-	0x0000
0x006	CAN0 Global Macro Automatic Block Transmission register low byte	C0GMABTL	R/W	R/W	-	-	0x00
0x007	CAN0 Global Macro Automatic Block Transmission register high byte	C0GMABTH	R/W	R/W	-	-	0x00
0x008	CAN0 Global Macro Automatic Block Transmission Delay register	C0GMABTD	R/W	R/W	-	-	0x00
0x040	CAN0 Module Mask 1 register lower half word	C0MASK1L	-	-	R/W	-	undefined
0x042	CAN0 Module Mask 1 register upper half word	C0MASK1H	-	-	R/W	-	undefined
0x044	CAN0 Module Mask 2 register lower half word	C0MASK2L	-	-	R/W	-	undefined
0x046	CAN0 Module Mask 2 register upper half word	C0MASK2H	-	-	R/W	-	undefined
0x048	CAN0 Module Mask 3 register lower half word	C0MASK3L	-	-	R/W	-	undefined
0x04A	CAN0 Module Mask 3 register upper half word	C0MASK3H	-	-	R/W	-	undefined
0x04C	CAN0 Module Mask 4 register lower half word	C0MASK4L	-	-	R/W	-	undefined
0x04E	CAN0 Module Mask 4 register upper half word	C0MASK4H	-	-	R/W	-	undefined
0x050	CAN0 Module Control register	C0CTRL	-	-	R/W	-	0x0000
0x052	CAN0 Module Last Error Code register	C0LEC	R/W	R/W	-	-	0x00
0x053	CAN0 Module Information register	C0INFO	R	R	-	-	0x00
0x054	CAN0 Module Error Counter	C0ERC	-	-	R/W	-	0x0000
0x056	CAN0 Module Interrupt Enable register	C0IE	-	-	R/W	-	0x0000

Table B-1 CAN special function registers (2/2)

Address offset	Register name	Shortcut	1	8	16	32	Initial value
0x056	CAN0 Module Interrupt Enable register low byte	C0IEL	R/W	R/W	-	-	0x00
0x057	CAN0 Module Interrupt Enable register high byte	C0IEH	R/W	R/W	-	-	0x00
0x058	CAN0 Module Interrupt Status register	C0INTS	-	-	R/W	-	0x0000
0x058	CAN0 Module Interrupt Status register low byte	C0INTSL	R/W	R/W	-	-	0x00
0x05A	CAN0 Module Bit-Rate Prescaler register	C0BRP	R/W	R/W	-	-	0xFF
0x05C	CAN0 Bit Rate register	C0BTR	-	-	R/W	-	0x370F
0x05E	CAN0 Module Last In-Pointer register	C0LIPT	-	R/W	-	-	undefined
0x060	CAN0 Module Receive History List Get Pointer register	C0RGPT	-	-	R/W	-	0x??02 (undefined)
0x060	CAN0 Module Receive History List Get Pointer register low byte	C0RGPTL	R/W	R/W	-	-	0x02
0x062	CAN0 Module Last Out-Pointer register	C0LOPT	-	R	-	-	undefined
0x064	CAN0 Module Transmit History List Get Pointer register	C0TGPT	-	-	R/W	-	0x??02 (undefined)
0x064	CAN0 Module Transmit History List Get Pointer register low byte	C0TGPTL	R/W	R/W	-	-	0x02
0x066	CAN0 Module Time Stamp register	C0TS	-	-	R/W	-	0x0000
0x066	CAN0 Module Time Stamp register low byte	C0TSL	R/W	R/W	-	-	0x00
0x067	CAN0 Module Time Stamp register high byte	C0TSH	R/W	R/W	-	-	0x00
0x100 to 0x4EF	CAN0 Message Buffer registers, see *Table 18-20 on page 575*.						

B.18 Other Special Function Registers

Table B-2 Other special function registers (1/13)

Address	Register name	Shortcut	1	8	16	32	Initial value
0xFFFFF064	CPU: Peripheral Area Select Control register	BPC	-	-	R/W	-	0x0000
0xFFFFF100	Interrupt Mask register 0	IMR0	-	-	R/W	-	0xFFFF
0xFFFFF100	Interrupt Mask register 0L	IMR0L	R/W	R/W	-	-	0xFF
0xFFFFF101	Interrupt Mask register 0H	IMR0H	R/W	R/W	-	-	0xFF
0xFFFFF102	Interrupt Mask register 1	IMR1	-	-	R/W	-	0xFFFF
0xFFFFF102	Interrupt Mask register 1L	IMR1L	R/W	R/W	-	-	0xFF
0xFFFFF103	Interrupt Mask register 1H	IMR1H	R/W	R/W	-	-	0xFF
0xFFFFF104	Interrupt Mask register 2	IMR2	-	-	R/W	-	0xFFFF
0xFFFFF104	Interrupt Mask register 2L	IMR2L	R/W	R/W	-	-	0xFF
0xFFFFF105	Interrupt Mask register 2H	IMR2H	R/W	R/W	-	-	0xFF
0xFFFFF106	Interrupt Mask register 3	IMR3	-	-	R/W	-	0xFFFF
0xFFFFF106	Interrupt Mask register 3L	IMR3L	R/W	R/W	-	-	0xFF
0xFFFFF107	Interrupt Mask register 3H	IMR3H	R/W	R/W	-	-	0xFF
0xFFFFF108	Interrupt Mask register 4	IMR4	-	-	R/W	-	0xFFFF
0xFFFFF108	Interrupt Mask register 4L	IMR4L	R/W	R/W	-	-	0xFF
0xFFFFF109	Interrupt Mask register 4H	IMR4H	R/W	R/W	-	-	0xFF
0xFFFFF10A	Interrupt Mask register 5	IMR5	-	-	R/W	-	0xFFFF
0xFFFFF10A	Interrupt Mask register 5L	IMR5L	R/W	R/W	-	-	0xFF
0xFFFFF10B	Interrupt Mask register 5H	IMR5H	R/W	R/W	-	-	0xFF
0xFFFFF114	Interrupt control register of INTWT0UV	WT0UVIC	R/W	R/W	-	-	0x47
0xFFFFF116	Interrupt control register of INTWT1UV	WT1UVIC	R/W	R/W	-	-	0x47
0xFFFFF11A	Interrupt control register of INTTM01	TM01IC	R/W	R/W	-	-	0x47
0xFFFFF11C	Interrupt control register of INTP0	P0IC	R/W	R/W	-	-	0x47
0xFFFFF11E	Interrupt control register of INTP1	P1IC	R/W	R/W	-	-	0x47
0xFFFFF120	Interrupt control register of INTP2	P2IC	R/W	R/W	-	-	0x47
0xFFFFF122	Interrupt control register of INTP3	P3IC	R/W	R/W	-	-	0x47
0xFFFFF12A	Interrupt control register of INTTZ0UV	TZ0UVIC	R/W	R/W	-	-	0x47
0xFFFFF12C	Interrupt control register of INTTZ1UV	TZ1UVIC	R/W	R/W	-	-	0x47
0xFFFFF12E	Interrupt control register of INTTZ2UV	TZ2UVIC	R/W	R/W	-	-	0x47
0xFFFFF130	Interrupt control register of INTTZ3UV	TZ3UVIC	R/W	R/W	-	-	0x47
0xFFFFF136	Interrupt control register of INTTP0OV	TP0OVIC	R/W	R/W	-	-	0x47
0xFFFFF138	Interrupt control register of INTTP0CC0	TP0CC0IC	R/W	R/W	-	-	0x47
0xFFFFF13A	Interrupt control register of INTTP0CC1	TP0CC1IC	R/W	R/W	-	-	0x47
0xFFFFF14E	Interrupt control register of INTTG0OV0	TG0OV0IC	R/W	R/W	-	-	0x47
0xFFFFF150	Interrupt control register of INTTG0OV1	TG0OV1IC	R/W	R/W	-	-	0x47
0xFFFFF152	Interrupt control register of INTTG0CC0	TG0CC0IC	R/W	R/W	-	-	0x47
0xFFFFF154	Interrupt control register of INTTG0CC1	TG0CC1IC	R/W	R/W	-	-	0x47
0xFFFFF156	Interrupt control register of INTTG0CC2	TG0CC2IC	R/W	R/W	-	-	0x47
0xFFFFF158	Interrupt control register of INTTG0CC3	TG0CC3IC	R/W	R/W	-	-	0x47

Table B-2 Other special function registers (2/13)

Address	Register name	Shortcut	1	8	16	32	Initial value
0xFFFFF15A	Interrupt control register of INTTG0CC4	TG0CC4IC	R/W	R/W	-	-	0x47
0xFFFFF15C	Interrupt control register of INTTG0CC5	TG0CC5IC	R/W	R/W	-	-	0x47
0xFFFFF15E	Interrupt control register of INTTG1OV0	TG1OV0IC	R/W	R/W	-	-	0x47
0xFFFFF160	Interrupt control register of INTTG1OV1	TG1OV1IC	R/W	R/W	-	-	0x47
0xFFFFF162	Interrupt control register of INTTG1CC0	TG1CC0IC	R/W	R/W	-	-	0x47
0xFFFFF164	Interrupt control register of INTTG1CC1	TG1CC1IC	R/W	R/W	-	-	0x47
0xFFFFF166	Interrupt control register of INTTG1CC2	TG1CC2IC	R/W	R/W	-	-	0x47
0xFFFFF168	Interrupt control register of INTTG1CC3	TG1CC3IC	R/W	R/W	-	-	0x47
0xFFFFF16A	Interrupt control register of INTTG1CC4	TG1CC4IC	R/W	R/W	-	-	0x47
0xFFFFF16C	Interrupt control register of INTTG1CC5	TG1CC5IC	R/W	R/W	-	-	0x47
0xFFFFF172	Interrupt control register of INTAD	ADIC	R/W	R/W	-	-	0x47
0xFFFFF174	Interrupt control register of INTC0ERR	C0ERRIC	R/W	R/W	-	-	0x47
0xFFFFF176	Interrupt control register of INTC0WUP	C0WUPIC	R/W	R/W	-	-	0x47
0xFFFFF178	Interrupt control register of INTC0REC	C0RECIC	R/W	R/W	-	-	0x47
0xFFFFF17A	Interrupt control register of INTC0TRX	C0TRXIC	R/W	R/W	-	-	0x47
0xFFFFF17C	Interrupt control register of INTCB0RE	CB0REIC	R/W	R/W	-	-	0x47
0xFFFFF17E	Interrupt control register of INTCB0R	CB0RIC	R/W	R/W	-	-	0x47
0xFFFFF180	Interrupt control register of INTCB0T	CB0TIC	R/W	R/W	-	-	0x47
0xFFFFF182	Interrupt control register of INTUA0RE	UA0REIC	R/W	R/W	-	-	0x47
0xFFFFF184	Interrupt control register of INTUA0R	UA0RIC	R/W	R/W	-	-	0x47
0xFFFFF186	Interrupt control register of INTUA0T	UA0TIC	R/W	R/W	-	-	0x47
0xFFFFF188	Interrupt control register of INTUA1RE	UA1REIC	R/W	R/W	-	-	0x47
0xFFFFF18A	Interrupt control register of INTUA1R	UA1RIC	R/W	R/W	-	-	0x47
0xFFFFF18C	Interrupt control register of INTUA1T	UA1TIC	R/W	R/W	-	-	0x47
0xFFFFF18E	Interrupt control register of INTIIC0	IIC0IC	R/W	R/W	-	-	0x47
0xFFFFF192	Interrupt control register of INTSG0	SG0IC	R/W	R/W	-	-	0x47
0xFFFFF19C	Interrupt control register of INTSW0	SW0IC	R/W	R/W	-	-	0x47
0xFFFFF19E	Interrupt control register of INTSW1	SW11IC	R/W	R/W	-	-	0x47
0xFFFFF1C2	Interrupt control register of INTCB1RE	CB1REIC	R/W	R/W	-	-	0x47
0xFFFFF1C4	Interrupt control register of INTCB1R	CB1RIC	R/W	R/W	-	-	0x47
0xFFFFF1C6	Interrupt control register of INTCB1T	CB1TIC	R/W	R/W	-	-	0x47
0xFFFFF1FA	In-service Priority register	ISPR	R	R	-	-	0x00
0xFFFFF1FC	Command register	PRCMD	-	W	-	-	undefined
0xFFFFF1FE	Power Save Control register	PSC	R/W	R/W	-	-	0x00
0xFFFFF200	ADC mode register 0	ADA0M0	R/W	R/W	-	-	0x00
0xFFFFF201	ADC mode register 1	ADA0M1	R/W	R/W	-	-	0x00
0xFFFFF202	ADC channel select register	ADA0S	R/W	R/W	-	-	0x00
0xFFFFF203	ADC mode register 2	ADA0M2	R/W	R/W	-	-	0x00
0xFFFFF204	ADC power fail comparison mode register	ADA0PFM	R/W	R/W	-	-	0x00
0xFFFFF205	ADC power fail threshold register	ADA0PFT	R/W	R/W	-	-	0x00
0xFFFFF210	ADC result register channel 0	ADCR00	-	-	R	-	undefined

Table B-2 Other special function registers (3/13)

Address	Register name	Shortcut	1	8	16	32	Initial value
0xFFFFF211	ADC result register high byte channel 0	ADCR0H0	R	R	-	-	undefined
0xFFFFF212	ADC result register channel 1	ADCR01	-	-	R	-	undefined
0xFFFFF213	ADC result register high byte channel 1	ADCR0H1	R	R	-	-	undefined
0xFFFFF214	ADC result register channel 2	ADCR02	-	-	R	-	undefined
0xFFFFF215	ADC result register high byte channel 2	ADCR0H2	R	R	-	-	undefined
0xFFFFF216	ADC result register channel 3	ADCR03	-	-	R	-	undefined
0xFFFFF217	ADC result register high byte channel 3	ADCR0H3	R	R	-	-	undefined
0xFFFFF218	ADC result register channel 4	ADCR04	-	-	R	-	undefined
0xFFFFF219	ADC result register high byte channel 4	ADCR0H4	R	R	-	-	undefined
0xFFFFF21A	ADC result register channel 5	ADCR05	-	-	R	-	undefined
0xFFFFF21B	ADC result register high byte channel 5	ADCR0H5	R	R	-	-	undefined
0xFFFFF21C	ADC result register channel 6	ADCR06	-	-	R	-	undefined
0xFFFFF21D	ADC result register high byte channel 6	ADCR0H6	R	R	-	-	undefined
0xFFFFF21E	ADC result register channel 7	ADCR07	-	-	R	-	undefined
0xFFFFF21F	ADC result register high byte channel 7	ADCR0H7	R	R	-	-	undefined
0xFFFFF300	Port Drive strength control register P0	PDSC0	R/W	R/W	-	-	0x00
0xFFFFF302	Port Drive strength control register P1	PDSC1	R/W	R/W	-	-	0x00
0xFFFFF304	Port Drive strength control register P2	PDSC2	R/W	R/W	-	-	0x00
0xFFFFF306	Port Drive strength control register P3	PDSC3	R/W	R/W	-	-	0x00
0xFFFFF308	Port Drive strength control register P4	PDSC4	R/W	R/W	-	-	0x00
0xFFFFF30A	Port Drive strength control register P5	PDSC5	R/W	R/W	-	-	0x00
0xFFFFF30C	Port Drive strength control register P6	PDSC6	R/W	R/W	-	-	0x00
0xFFFFF310	Port Drive strength control register P8	PDSC8	R/W	R/W	-	-	0x00
0xFFFFF312	Port Drive strength control register P9	PDSC9	R/W	R/W	-	-	0x00
0xFFFFF314	Port Drive strength control register P10	PDSC10	R/W	R/W	-	-	0x00
0xFFFFF344	Port LCD control register P2	PLCDC2	R/W	R/W	-	-	0x00
0xFFFFF346	Port LCD control register P3	PLCDC3	R/W	R/W	-	-	0x00
0xFFFFF348	Port LCD control register P4	PLCDC4	R/W	R/W	-	-	0x00
0xFFFFF34C	Port LCD control register port 6	PLCDC6	R/W	R/W	-	-	0x00
0xFFFFF350	Port LCD control register port 8	PLCDC8	R/W	R/W	-	-	0x00
0xFFFFF352	Port LCD control register port 9	PLCDC9	R/W	R/W	-	-	0x00
0xFFFFF354	Port LCD control register port 10	PLCDC10	R/W	R/W	-	-	0x00
0xFFFFF360	Port open drain control register P0	PODC0	R/W	R/W	-	-	0x00
0xFFFFF362	Port open drain control register P1	PODC1	R/W	R/W	-	-	0x00
0xFFFFF364	Port open drain control register P2	PODC2	R/W	R/W	-	-	0x00
0xFFFFF366	Port open drain control register P3	PODC3	R/W	R/W	-	-	0x00
0xFFFFF368	Port open drain control register P4	PODC4	R/W	R/W	-	-	0x00
0xFFFFF36A	Port open drain control register P5	PODC5	R/W	R/W	-	-	0x00
0xFFFFF36C	Port open drain control register P6	PODC6	R/W	R/W	-	-	0x00
0xFFFFF370	Port open drain control register P8	PODC8	R/W	R/W	-	-	0x00
0xFFFFF372	Port open drain control register P9	PODC9	R/W	R/W	-	-	0x00

Table B-2 Other special function registers (4/13)

Address	Register name	Shortcut	1	8	16	32	Initial value
0xFFFFF374	Port open drain control register P10	PODC10	R/W	R/W	-	-	0x00
0xFFFFF378	Port open drain control register P12	PODC12	R/W	R/W	-	-	0x00
0xFFFFF37A	Port open drain control register P13	PODC13	R/W	R/W	-	-	0x00
0xFFFFF380	Port input characteristic control register P0	PICC0	R/W	R/W	-	-	0xFF
0xFFFFF382	Port input characteristic control register P1	PICC1	R/W	R/W	-	-	0xFF
0xFFFFF384	Port input characteristic control register P2	PICC2	R/W	R/W	-	-	0xFF
0xFFFFF386	Port input characteristic control register P3	PICC3	R/W	R/W	-	-	0xFF
0xFFFFF388	Port input characteristic control register P4	PICC4	R/W	R/W	-	-	0xFF
0xFFFFF38A	Port input characteristic control register P5	PICC5	R/W	R/W	-	-	0xFF
0xFFFFF38C	Port input characteristic control register P6	PICC6	R/W	R/W	-	-	0xFF
0xFFFFF390	Port input characteristic control register P8	PICC8	R/W	R/W	-	-	0xFF
0xFFFFF392	Port input characteristic control register P9	PICC9	R/W	R/W	-	-	0xFF
0xFFFFF394	Port input characteristic control register P10	PICC10	R/W	R/W	-	-	0xFF
0xFFFFF398	Port input characteristic control register P12	PICC12	R/W	R/W	-	-	0xFF
0xFFFFF39A	Port input characteristic control register P13	PICC13	R/W	R/W	-	-	0xFF
0xFFFFF3A0	Port input level control register P0	PILC0	R/W	R/W	-	-	0x00
0xFFFFF3A2	Port input level control register P1	PILC1	R/W	R/W	-	-	0x00
0xFFFFF3A4	Port input level control register P2	PILC2	R/W	R/W	-	-	0x00
0xFFFFF3A6	Port input level control register P3	PILC3	R/W	R/W	-	-	0x00
0xFFFFF3A8	Port input level control register P4	PILC4	R/W	R/W	-	-	0x00
0xFFFFF3AA	Port input level control register P5	PILC5	R/W	R/W	-	-	0x00
0xFFFFF3AC	Port input level control register P6	PILC6	R/W	R/W	-	-	0x00
0xFFFFF3AE	Port input level control register P7	PILC7	R/W	R/W	-	-	0x00
0xFFFFF3B0	Port input level control register P8	PILC8	R/W	R/W	-	-	0x00
0xFFFFF3B2	Port input level control register P9	PILC9	R/W	R/W	-	-	0x00
0xFFFFF3B4	Port input level control register P10	PILC10	R/W	R/W	-	-	0x00
0xFFFFF3B8	Port input level control register P12	PILC12	R/W	R/W	-	-	0x00
0xFFFFF3BA	Port input level control register P13	PILC13	R/W	R/W	-	-	0x00
0xFFFFF3C0	Port pin read register P0	PPR0	R	R	-	-	0x00
0xFFFFF3C2	Port pin read register P1	PPR1	R	R	-	-	0x00
0xFFFFF3C4	Port pin read register P2	PPR2	R	R	-	-	0x00
0xFFFFF3C6	Port pin read register P3	PPR3	R	R	-	-	0x00
0xFFFFF3C8	Port pin read register P4	PPR4	R	R	-	-	0x00
0xFFFFF3CA	Port pin read register P5	PPR5	R	R	-	-	0x00
0xFFFFF3CC	Port pin read register P6	PPR6	R	R	-	-	0x00
0xFFFFF3D0	Port pin read register P8	PPR8	R	R	-	-	0x00
0xFFFFF3D2	Port pin read register P9	PPR9	R	R	-	-	0x00
0xFFFFF3D4	Port pin read register P10	PPR10	R	R	-	-	0x00
0xFFFFF3D8	Port pin read register P12	PPR12	R	R	-	-	0x00
0xFFFFF3DA	Port pin read register P13	PPR13	R	R	-	-	0x00
0xFFFFF3E0	Port read control register P0	PRC0	R/W	R/W	-	-	0x00

Table B-2 Other special function registers (5/13)

Address	Register name	Shortcut	1	8	16	32	Initial value
0xFFFFF3E2	Port read control register P1	PRC1	R/W	R/W	-	-	0x00
0xFFFFF3E4	Port read control register P2	PRC2	R/W	R/W	-	-	0x00
0xFFFFF3E6	Port read control register P3	PRC3	R/W	R/W	-	-	0x00
0xFFFFF3E8	Port read control register P4	PRC4	R/W	R/W	-	-	0x00
0xFFFFF3EA	Port read control register P5	PRC5	R/W	R/W	-	-	0x00
0xFFFFF3EC	Port read control register P6	PRC6	R/W	R/W	-	-	0x00
0xFFFFF3F0	Port read control register P8	PRC8	R/W	R/W	-	-	0x00
0xFFFFF3F2	Port read control register P9	PRC9	R/W	R/W	-	-	0x00
0xFFFFF3F4	Port read control register P10	PRC10	R/W	R/W	-	-	0x00
0xFFFFF3F8	Port read control register P12	PRC12	R/W	R/W	-	-	0x00
0xFFFFF3FA	Port read control register P13	PRC13	R/W	R/W	-	-	0x00
0xFFFFF400	Port register port 0	P0	R/W	R/W	-	-	0x00
0xFFFFF402	Port register port 1	P1	R/W	R/W	-	-	0x00
0xFFFFF404	Port register port 2	P2	R/W	R/W	-	-	0x00
0xFFFFF406	Port register port 3	P3	R/W	R/W	-	-	0x00
0xFFFFF408	Port register port 4	P4	R/W	R/W	-	-	0x00
0xFFFFF40A	Port register port 5	P5	R/W	R/W	-	-	0x00
0xFFFFF40C	Port register port 6	P6	R/W	R/W	-	-	0x00
0xFFFFF40E	Port register port 7	P7	R/W	R/W	-	-	0x00
0xFFFFF410	Port register port 8	P8	R/W	R/W	-	-	0x00
0xFFFFF412	Port register port 9	P9	R/W	R/W	-	-	0x00
0xFFFFF414	Port register port 10	P10	R/W	R/W	-	-	0x00
0xFFFFF418	Port register port 12	P12	R/W	R/W	-	-	0x00
0xFFFFF41A	Port register port 13	P13	R/W	R/W	-	-	0x00
0xFFFFF420	Port mode register port 0	PM0	R/W	R/W	-	-	0xFF
0xFFFFF422	Port mode register port 1	PM1	R/W	R/W	-	-	0xFF
0xFFFFF424	Port mode register port 2	PM2	R/W	R/W	-	-	0xFF
0xFFFFF426	Port mode register port 3	PM3	R/W	R/W	-	-	0xFF
0xFFFFF428	Port mode register port 4	PM4	R/W	R/W	-	-	0xFF
0xFFFFF42A	Port mode register port 5	PM5	R/W	R/W	-	-	0xFF
0xFFFFF42C	Port mode register port 6	PM6	R/W	R/W	-	-	0xFF
0xFFFFF430	Port mode register port 8	PM8	R/W	R/W	-	-	0xFF
0xFFFFF432	Port mode register port 9	PM9	R/W	R/W	-	-	0xFF
0xFFFFF434	Port mode register port 10	PM10	R/W	R/W	-	-	0xFF
0xFFFFF438	Port mode register port 12	PM12	R/W	R/W	-	-	0xFF
0xFFFFF43A	Port mode register port 13	PM13	R/W	R/W	-	-	0xFF
0xFFFFF440	Port mode control register port 0	PMC0	R/W	R/W	-	-	0x00
0xFFFFF442	Port mode control register port 1	PMC1	R/W	R/W	-	-	0x00
0xFFFFF444	Port mode control register port 2	PMC2	R/W	R/W	-	-	0x00
0xFFFFF446	Port mode control register port 3	PMC3	R/W	R/W	-	-	0x00
0xFFFFF448	Port mode control register port 4	PMC4	R/W	R/W	-	-	0x00

Table B-2 Other special function registers (6/13)

Address	Register name	Shortcut	1	8	16	32	Initial value
0xFFFFF44A	Port mode control register port 5	PMC5	R/W	R/W	-	-	0x00
0xFFFFF44C	Port mode control register port 6	PMC6	R/W	R/W	-	-	0x00
0xFFFFF44E	Port mode control register port 7	PMC7	R/W	R/W	-	-	0x00
0xFFFFF450	Port mode control register port 8	PMC8	R/W	R/W	-	-	0x00
0xFFFFF452	Port mode control register port 9	PMC9	R/W	R/W	-	-	0x00
0xFFFFF454	Port mode control register port 10	PMC10	R/W	R/W	-	-	0x00
0xFFFFF458	Port mode control register port 12	PMC12	R/W	R/W	-	-	0x00
0xFFFFF45A	Port mode control register port 13	PMC13	R/W	R/W	-	-	0x00
0xFFFFF46A	Port function control register port 5	PFC5	R/W	R/W	-	-	0x00
0xFFFFF470	Port function control register port 8	PFC8	R/W	R/W	-	-	0x00
0xFFFFF472	Port function control register port 9	PFC9	R/W	R/W	-	-	0x00
0xFFFFF47A	Port function control register port 13	PFC13	R/W	R/W	-	-	0x00
0xFFFFF560	Synchronized counter read register WT0	WT0CNT0	-	-	R	-	0x0000
0xFFFFF562	Non-synchronized counter read register WT0	WT0CNT1	-	-	R	-	0x0000
0xFFFFF564	Counter reload register WT0	WT0R	-	-	R/W	-	0x0000
0xFFFFF566	Control register WT0	WT0CTL	R/W	R/W	-	-	0x00
0xFFFFF570	Synchronized counter read register WT1	WT1CNT0	-	-	R	-	0x0000
0xFFFFF572	Non-synchronized counter read register WT1	WT1CNT1	-	-	R	-	0x0000
0xFFFFF574	Counter reload register WT1	WT1R	-	-	R/W	-	0x0000
0xFFFFF576	Control register WT1	WT1CTL	R/W	R/W	-	-	0x00
0xFFFFF590	Watchdog timer Frequency select register	WDCS	R/W	R/W	-	-	0x07
0xFFFFF592	Watchdog timer security register	WCMD	R/W	R/W	-	-	undefined
0xFFFFF594	Watchdog timer mode register	WDTM	R/W	R/W	-	-	0x00
0xFFFFF596	Watchdog timer error register	WPHS	R/W	R/W	-	-	0x00
0xFFFFF5A0	SG0 Frequency register	SG0F	-	-	-	R/W	0x00000000
0xFFFFF5A0	SG0 Frequency register low	SG0FL	-	-	R/W	-	0x0000
0xFFFFF5A2	SG0 Frequency register high	SG0FH	-	-	R/W	-	0x0000
0xFFFFF5A4	SG0 Amplitude register	SG0PWM	-	-	R/W	-	0x0000
0xFFFFF5A6	SG0 Duration factor register	SG0SDF	R/W	R/W	-	-	0x00
0xFFFFF5A7	SG0 Control register	SG0CTL	R/W	R/W	-	-	0x00
0xFFFFF5A8	SG0 Interrupt threshold register	SG0ITH	-	-	R/W	-	0x0000
0xFFFFF5C0	Timer Mode Control register 0	MCNTC00	R/W	R/W	-	-	0x00
0xFFFFF5C6	Compare register 3HW	MCMP03HW	-	-	R/W	-	0x0000
0xFFFFF5C6	Compare register 30	MCMP030	-	R/W	-	-	0x00
0xFFFFF5C7	Compare register 31	MCMP031	-	R/W	-	-	0x00
0xFFFFF5C8	Compare register 4HW	MCMP04HW	-	-	R/W	-	0x0000
0xFFFFF5C8	Compare register 40	MCMP040	-	R/W	-	-	0x00
0xFFFFF5C9	Compare register 41	MCMP041	-	R/W	-	-	0x00
0xFFFFF5CA	Compare Control register 1	MCMPC01	R/W	R/W	-	-	0x00
0xFFFFF5CC	Compare Control register 2	MCMPC02	R/W	R/W	-	-	0x00
0xFFFFF5CE	Compare Control register 3	MCMPC03	R/W	R/W	-	-	0x00

Table B-2 Other special function registers (7/13)

Address	Register name	Shortcut	1	8	16	32	Initial value
0xFFFFF5D0	Compare Control register 4	MCMPC04	R/W	R/W	-	-	0x00
0xFFFFF5D4	Timer Mode Control register 1	MCNTC01	R/W	R/W	-	-	0x00
0xFFFFF5D6	Compare register 5HW	MCMP05HW	-	-	R/W	-	0x0000
0xFFFFF5D6	Compare register 50	MCMP050	-	R/W	-	-	0x00
0xFFFFF5D7	Compare register 51	MCMP051	-	R/W	-	-	0x00
0xFFFFF5D8	Compare register 6HW	MCMP06HW	-	-	R/W	-	0x0000
0xFFFFF5D8	Compare register 60	MCMP060	-	R/W	-	-	0x00
0xFFFFF5D9	Compare register 61	MCMP061	-	R/W	-	-	0x00
0xFFFFF5DA	Compare Control register 5	MCMPC05	R/W	R/W	-	-	0x00
0xFFFFF5DC	Compare Control register 6	MCMPC06	R/W	R/W	-	-	0x00
0xFFFFF5E4	TM00 16-bit capture/compare register 0	CR001	-	-	R/W	-	0x0000
0xFFFFF5E6	TM00 Control register	TMC00	R/W	R/W	-	-	0x00
0xFFFFF5E7	TM00 Prescaler mode register	PRM00	R/W	R/W	-	-	0x00
0xFFFFF5E8	TM00 Capture/Compare Control register	CRC00	R/W	R/W	-	-	0x00
0xFFFFF600	TMZ0 Synchronized counter read register	TZ0CNT0	-	-	R	-	0x0000
0xFFFFF602	TMZ0 non-synchronized counter read register	TZ0CNT1	-	-	R	-	0x0000
0xFFFFF604	TMZ0 counter reload register	TZ0R	-	-	R/W	-	0x0000
0xFFFFF606	TMZ0 control register	TZ0CTL	R/W	R/W	-	-	0x00
0xFFFFF608	TMZ1 Synchronized counter read register	TZ1CNT0	-	-	R	-	0x0000
0xFFFFF60A	TMZ1 non-synchronized counter read register	TZ1CNT1	-	-	R	-	0x0000
0xFFFFF60C	TMZ1 counter reload register	TZ1R	-	-	R/W	-	0x0000
0xFFFFF60E	TMZ1 control register	TZ1CTL	R/W	R/W	-	-	0x00
0xFFFFF610	TMZ2 Synchronized counter read register	TZ2CNT0	-	-	R	-	0x0000
0xFFFFF612	TMZ2 non-synchronized counter read register	TZ2CNT1	-	-	R	-	0x0000
0xFFFFF614	TMZ2 counter reload register	TZ2R	-	-	R/W	-	0x0000
0xFFFFF616	TMZ2 control register	TZ2CTL	R/W	R/W	-	-	0x00
0xFFFFF618	TMZ3 Synchronized counter read register	TZ3CNT0	-	-	R	-	0x0000
0xFFFFF61A	TMZ3 non-synchronized counter read register	TZ3CNT1	-	-	R	-	0x0000
0xFFFFF61C	TMZ3 counter reload register	TZ3R	-	-	R/W	-	0x0000
0xFFFFF61E	TMZ3 control register	TZ3CTL	R/W	R/W	-	-	0x00
0xFFFFF620	TMZ4 Synchronized counter read register	TZ4CNT0	-	-	R	-	0x0000
0xFFFFF622	TMZ4 non-synchronized counter read register	TZ4CNT1	-	-	R	-	0x0000
0xFFFFF624	TMZ4 counter reload register	TZ4R	-	-	R/W	-	0x0000
0xFFFFF626	TMZ4 control register	TZ4CTL	R/W	R/W	-	-	0x00
0xFFFFF628	TMZ5 Synchronized counter read register	TZ5CNT0	-	-	R	-	0x0000
0xFFFFF62A	TMZ5 non-synchronized counter read register	TZ5CNT1	-	-	R	-	0x0000
0xFFFFF62C	TMZ5 counter reload register	TZ5R	-	-	R/W	-	0x0000
0xFFFFF62E	TMZ5 control register	TZ5CTL	R/W	R/W	-	-	0x00
0xFFFFF660	TMP0 timer control register 0	TP0CTL0	R/W	R/W	-	-	0x00
0xFFFFF661	TMP0 timer control register 1	TP0CTL1	R/W	R/W	-	-	0x00
0xFFFFF662	TMP0 timer-specific I/O control register 0	TP0IOC0	R/W	R/W	-	-	0x00

Table B-2 Other special function registers (8/13)

Address	Register name	Shortcut	1	8	16	32	Initial value
0xFFFFF663	TMP0 timer-specific I/O control register 1	TP0IOC1	R/W	R/W	-	-	0x00
0xFFFFF664	TMP0 timer-specific I/O control register 2	TP0IOC2	R/W	R/W	-	-	0x00
0xFFFFF665	TMP0 option register	TP0OPT0	R/W	R/W	-	-	0x00
0xFFFFF666	TMP0 capture/compare register 0	TP0CCR0	-	-	R/W	-	0x0000
0xFFFFF668	TMP0 capture/compare register 1	TP0CCR1	-	-	R/W	-	0x0000
0xFFFFF66A	TMP0 count register	TP0CNT	-	-	R	-	0x0000
0xFFFFF6A0	Timer mode register TMG 0	TMGM0	-	-	R/W	-	0x0000
0xFFFFF6A0	Timer mode register TMG 0 low byte	TMGM0L	R/W	R/W	-	-	0x00
0xFFFFF6A1	Timer mode register TMG 0 high byte	TMGM0H	R/W	R/W	-	-	0x00
0xFFFFF6A2	Channel mode register TMG 0	TMGCM0	-	-	R/W	-	0x0000
0xFFFFF6A2	Channel mode register TMG 0 low byte	TMGCM0L	R/W	R/W	-	-	0x00
0xFFFFF6A3	Channel mode register TMG 0 high byte	TMGCM0H	R/W	R/W	-	-	0x00
0xFFFFF6A4	Output control register TMG 0	OCTLG0	-	-	R/W	-	0x4444
0xFFFFF6A4	Output control register TMG 0 low byte	OCTLG0L	R/W	R/W	-	-	0x44
0xFFFFF6A5	Output control register TMG 0 high byte	OCTLG0H	R/W	R/W	-	-	0x44
0xFFFFF6A6	Time base status register TMG 0	TMGST0	R	R	-	-	0x00
0xFFFFF6A8	Timer count register 0 TMG 0	TMG00	-	-	R	-	0x0000
0xFFFFF6AA	Timer count register 1 TMG 0	TMG01	-	-	R	-	0x0000
0xFFFFF6AC	Capture / Compare register 0 TMG 0	GCC00	-	-	R/W	-	0x0000
0xFFFFF6AE	Capture / Compare register 1 TMG 0	GCC01	-	-	R/W	-	0x0000
0xFFFFF6B0	Capture / Compare register 2 TMG 0	GCC02	-	-	R/W	-	0x0000
0xFFFFF6B2	Capture / Compare register 3 TMG 0	GCC03	-	-	R/W	-	0x0000
0xFFFFF6B4	Capture / Compare register 4 TMG 0	GCC04	-	-	R/W	-	0x0000
0xFFFFF6B6	Capture / Compare register 5 TMG 0	GCC05	-	-	R/W	-	0x0000
0xFFFFF6C0	Timer mode register TMG 1	TMGM1	-	-	R/W	-	0x0000
0xFFFFF6C0	Timer mode register TMG 1 low byte	TMGM1L	R/W	R/W	-	-	0x00
0xFFFFF6C1	Timer mode register TMG 1 high byte	TMGM1H	R/W	R/W	-	-	0x00
0xFFFFF6C2	Channel mode register TMG 1	TMGCM1	-	-	R/W	-	0x0000
0xFFFFF6C2	Channel mode register TMG 1 low byte	TMGCM1L	R/W	R/W	-	-	0x00
0xFFFFF6C3	Channel mode register TMG 1 high byte	TMGCM1H	R/W	R/W	-	-	0x00
0xFFFFF6C4	Output control register TMG 1	OCTLG1	-	-	R/W	-	0x4444
0xFFFFF6C4	Output control register TMG 1 low byte	OCTLG1L	R/W	R/W	-	-	0x44
0xFFFFF6C5	Output control register TMG 1 high byte	OCTLG1H	R/W	R/W	-	-	0x44
0xFFFFF6C6	Time base status TMG 1	TMGST1	R	R	-	-	0x00
0xFFFFF6C8	Timer count register 0 TMG 1	TMG10	-	-	R	-	0x0000
0xFFFFF6CA	Timer count register 1 TMG 1	TMG11	-	-	R	-	0x0000
0xFFFFF6CC	Capture / Compare register 0 TMG 1	GCC10	-	-	R/W	-	0x0000
0xFFFFF6CE	Capture / Compare register 1 TMG 1	GCC11	-	-	R/W	-	0x0000
0xFFFFF6D0	Capture / Compare register 2 TMG 1	GCC12	-	-	R/W	-	0x0000
0xFFFFF6D2	Capture / Compare register 3 TMG 1	GCC13	-	-	R/W	-	0x0000
0xFFFFF6D4	Capture / Compare register 4 TMG 1	GCC14	-	-	R/W	-	0x0000

Table B-2 Other special function registers (9/13)

Address	Register name	Shortcut	1	8	16	32	Initial value
0xFFFFF6D6	Capture / Compare register 5 TMG 1	GCC15	-	-	R/W	-	0x0000
0xFFFFF700	Interrupt mode register 0	INTM0	R/W	R/W	-	-	0x00
0xFFFFF702	Interrupt mode register 1	INTM1	R/W	R/W	-	-	0x00
0xFFFFF710	Digital filter enable register 0	DFEN0	-	-	R/W	-	0x0000
0xFFFFF710	Digital filter enable register 0 low byte	DFEN0L	R/W	R/W	-	-	0x00
0xFFFFF711	Digital filter enable register 0 high byte	DFEN0H	R/W	R/W	-	-	0x00
0xFFFFF712	Digital filter enable register 1	DFEN1	-	-	R/W	-	0x0000
0xFFFFF712	Digital filter enable register 1 low byte	DFEN1L	R/W	R/W	-	-	0x00
0xFFFFF713	Digital filter enable register 1 high byte	DFEN1H	R/W	R/W	-	-	0x00
0xFFFFF71A	Sub oscillator clock monitor control register	CLMCS	R/W	R/W	-	-	0x00
0xFFFFF720	Peripheral Function Select register 0	PFSR0	R/W	R/W	-	-	0x01
0xFFFFF724	Peripheral function select register 2	PFSR2	R/W	R/W	-	-	0x01
0xFFFFF726	Peripheral Function Select register 3	PFSR3	R/W	R/W	-	-	0x01
0xFFFFF800	Protection register	PHCMD	-	R/W	-	-	undefined
0xFFFFF802	Peripheral status	PHS	R/W	R/W	-	-	0x00
0xFFFFF820	Power Save Mode	PSM	R/W	R/W	-	-	0x08/0x00
0xFFFFF822	Clock Control	CKC	-	R/W	-	-	0x00
0xFFFFF824	Clock Generator Status	CGSTAT	-	R	-	-	0x0D
0xFFFFF826	Watch Dog Clock Control	WCC	-	R/W	-	-	0x00
0xFFFFF828	Processor Clock Control	PCC	-	R/W	-	-	0x10
0xFFFFF82A	Frequency Modulation Control	SCFMC	R/W	R/W	-	-	0x00
0xFFFFF82C	Frequency Control 0	SCFC0	R/W	R/W	-	-	0x52
0xFFFFF82E	Frequency Control 1	SCFC1	R/W	R/W	-	-	0xEB
0xFFFFF830	SSCG Postscaler Control	SCPS	R/W	R/W	-	-	0x21
0xFFFFF832	SPCLK Control	SCC	R/W	R/W	-	-	0x00
0xFFFFF834	FOUTCLK Control	FCC	R/W	R/W	-	-	0x00
0xFFFFF836	Watch Timer Clock Control	TCC	R/W	R/W	-	-	0x00
0xFFFFF838	IIC Clock Control	ICC	R/W	R/W	-	-	0x00
0xFFFFF83C	Set Default Clock	SDC	-	R/W	-	-	0x00
0xFFFFF870	Main oscillator clock monitor mode register	CLMM	R/W	R/W	-	-	0x00
0xFFFFF878	Sub oscillator clock monitor mode register	CLMS	R/W	R/W	-	-	0x00
0xFFFFF900	VFB flash/ROM correction control register 0	CORCTL0	-	R/W	-	-	0x00
0xFFFFF901	VFB flash/ROM correction control register 1	CORCTL1	-	R/W	-	-	0x00
0xFFFFF910	VFB flash/ROM correction address register 0L	CORADR0L	-	-	R/W	-	0x0000
0xFFFFF910	VFB flash/ROM correction address register 0LL	CORADR0LL	-	R/W	-	-	0x00
0xFFFFF911	VFB flash/ROM correction address register 0LH	CORADR0LH	-	R/W	-	-	0x00
0xFFFFF912	VFB flash/ROM correction address register 0H	CORADR0H	-	-	R/W	-	0x0000
0xFFFFF912	VFB flash/ROM correction address register 0HL	CORADR0HL	-	R/W	-	-	0x00
0xFFFFF913	VFB flash/ROM correction address register 0HH	CORADR0HH	-	R/W	-	-	0x00
0xFFFFF914	VFB flash/ROM correction address register 1L	CORADR1L	-	-	R/W	-	0x0000
0xFFFFF914	VFB flash/ROM correction address register 1LL	CORADR1LL	-	R/W	-	-	0x00

Table B-2 Other special function registers (10/13)

Address	Register name	Shortcut	1	8	16	32	Initial value
0xFFFFF915	VFB flash/ROM correction address register 1LH	CORADR1LH	-	R/W	-	-	0x00
0xFFFFF916	VFB flash/ROM correction address register 1H	CORADR1H	-	-	R/W	-	0x0000
0xFFFFF916	VFB flash/ROM correction address register 1HL	CORADR1HL	-	R/W	-	-	0x00
0xFFFFF917	VFB flash/ROM correction address register 1HH	CORADR1HH	-	R/W	-	-	0x00
0xFFFFF918	VFB flash/ROM correction address register 2L	CORADR2L	-	-	R/W	-	0x0000
0xFFFFF918	VFB flash/ROM correction address register 2LL	CORADR2LL	-	R/W	-	-	0x00
0xFFFFF919	VFB flash/ROM correction address register 2LH	CORADR2LH	-	R/W	-	-	0x00
0xFFFFF91A	VFB flash/ROM correction address register 2H	CORADR2H	-	-	R/W	-	0x0000
0xFFFFF91A	VFB flash/ROM correction address register 2HL	CORADR2HL	-	R/W	-	-	0x00
0xFFFFF91B	VFB flash/ROM correction address register 2HH	CORADR2HH	-	R/W	-	-	0x00
0xFFFFF91C	VFB flash/ROM correction address register 3L	CORADR3L	-	-	R/W	-	0x0000
0xFFFFF91C	VFB flash/ROM correction address register 3LL	CORADR3LL	-	R/W	-	-	0x00
0xFFFFF91D	VFB flash/ROM correction address register 3LH	CORADR3LH	-	R/W	-	-	0x00
0xFFFFF91E	VFB flash/ROM correction address register 3H	CORADR3H	-	-	R/W	-	0x0000
0xFFFFF91E	VFB flash/ROM correction address register 3HL	CORADR3HL	-	R/W	-	-	0x00
0xFFFFF91F	VFB flash/ROM correction address register 3HH	CORADR3HH	-	R/W	-	-	0x00
0xFFFFF920	VFB flash/ROM correction address register 4L	CORADR4L	-	-	R/W	-	0x0000
0xFFFFF920	VFB flash/ROM correction address register 4LL	CORADR4LL	-	R/W	-	-	0x00
0xFFFFF921	VFB flash/ROM correction address register 4LH	CORADR4LH	-	R/W	-	-	0x00
0xFFFFF922	VFB flash/ROM correction address register 4H	CORADR4H	-	-	R/W	-	0x0000
0xFFFFF922	VFB flash/ROM correction address register 4HL	CORADR4HL	-	R/W	-	-	0x00
0xFFFFF923	VFB flash/ROM correction address register 4HH	CORADR4HH	-	R/W	-	-	0x00
0xFFFFF924	VFB flash/ROM correction address register 5L	CORADR5L	-	-	R/W	-	0x0000
0xFFFFF924	VFB flash/ROM correction address register 5LL	CORADR5LL	-	R/W	-	-	0x00
0xFFFFF925	VFB flash/ROM correction address register 5LH	CORADR5LH	-	R/W	-	-	0x00
0xFFFFF926	VFB flash/ROM correction address register 5H	CORADR5H	-	-	R/W	-	0x0000
0xFFFFF926	VFB flash/ROM correction address register 5HL	CORADR5HL	-	R/W	-	-	0x00
0xFFFFF927	VFB flash/ROM correction address register 5HH	CORADR5HH	-	R/W	-	-	0x00
0xFFFFF930	VFB flash/ROM correction value register 0L	CORVAL0L	-	-	R/W	-	0x0000
0xFFFFF932	VFB flash/ROM correction value register 0H	CORVAL0H	-	-	R/W	-	0x0000
0xFFFFF934	VFB flash/ROM correction value register 1L	CORVAL1L	-	-	R/W	-	0x0000
0xFFFFF936	VFB flash/ROM correction value register 1H	CORVAL1H	-	-	R/W	-	0x0000
0xFFFFF938	VFB flash/ROM correction value register 2L	CORVAL2L	-	-	R/W	-	0x0000
0xFFFFF93A	VFB flash/ROM correction value register 2H	CORVAL2H	-	-	R/W	-	0x0000
0xFFFFF93C	VFB flash/ROM correction value register 3L	CORVAL3L	-	-	R/W	-	0x0000
0xFFFFF93E	VFB flash/ROM correction value register 3H	CORVAL3H	-	-	R/W	-	0x0000
0xFFFFF940	VFB flash/ROM correction value register 4L	CORVAL4L	-	-	R/W	-	0x0000
0xFFFFF942	VFB flash/ROM correction value register 4H	CORVAL4H	-	-	R/W	-	0x0000
0xFFFFF944	VFB flash/ROM correction value register 5L	CORVAL5L	-	-	R/W	-	0x0000
0xFFFFF946	VFB flash/ROM correction value register 5H	CORVAL5H	-	-	R/W	-	0x0000
0xFFFFFA00	UARTA0 Control register 0	UA0CTL0	R/W	R/W	-	-	0x10

Table B-2 Other special function registers (11/13)

Address	Register name	Shortcut	1	8	16	32	Initial value
0xFFFFFA01	UARTA0 Control register 1	UA0CTL1	R/W	R/W	-	-	0x00
0xFFFFFA02	UARTA0 Control register 2	UA0CTL2	R/W	R/W	-	-	0xFF
0xFFFFFA03	UARTA0 Option register	UA0OPT0	R/W	R/W	-	-	0x14
0xFFFFFA04	UARTA0 Status register	UA0STR	R/W	R/W	-	-	0x00
0xFFFFFA06	UARTA0 Reception data register	UA0RX	-	R	-	-	0xFF
0xFFFFFA07	UARTA0 Transfer data register	UA0TX	R/W	R/W	-	-	0xFF
0xFFFFFA10	UARTA1 Control register 0	UA1CTL0	R/W	R/W	-	-	0x10
0xFFFFFA11	UARTA1 Control register 1	UA1CTL1	R/W	R/W	-	-	0x00
0xFFFFFA12	UARTA1 Control register 2	UA1CTL2	R/W	R/W	-	-	0xFF
0xFFFFFA13	UARTA1 Option register	UA1OPT0	R/W	R/W	-	-	0x14
0xFFFFFA14	UARTA1 Status register	UA1STR	R/W	R/W	-	-	0x00
0xFFFFFA16	UARTA1 Reception data register	UA1RX	-	R	-	-	0xFF
0xFFFFFA17	UARTA1 Transfer data register	UA1TX	R/W	R/W	-	-	0xFF
0xFFFFFB00	LCD clock control	LCDC0	R/W	R/W	-	-	0x00
0xFFFFFB01	LCD display mode control	LCDM0	R/W	R/W	-	-	0x00
0xFFFFFB20	LCD RAM data	SEGREG000	R/W	R/W	-	-	0x00
0xFFFFFB20	LCD RAM data	SEGREG020	R/W	R/W	-	-	0x00
0xFFFFFB21	LCD RAM data	SEGREG001	R/W	R/W	-	-	0x00
0xFFFFFB21	LCD RAM data	SEGREG021	R/W	R/W	-	-	0x00
0xFFFFFB22	LCD RAM data	SEGREG002	R/W	R/W	-	-	0x00
0xFFFFFB22	LCD RAM data	SEGREG022	R/W	R/W	-	-	0x00
0xFFFFFB23	LCD RAM data	SEGREG003	R/W	R/W	-	-	0x00
0xFFFFFB23	LCD RAM data	SEGREG023	R/W	R/W	-	-	0x00
0xFFFFFB24	LCD RAM data	SEGREG004	R/W	R/W	-	-	0x00
0xFFFFFB24	LCD RAM data	SEGREG024	R/W	R/W	-	-	0x00
0xFFFFFB25	LCD RAM data	SEGREG005	R/W	R/W	-	-	0x00
0xFFFFFB25	LCD RAM data	SEGREG025	R/W	R/W	-	-	0x00
0xFFFFFB26	LCD RAM data	SEGREG006	R/W	R/W	-	-	0x00
0xFFFFFB26	LCD RAM data	SEGREG026	R/W	R/W	-	-	0x00
0xFFFFFB27	LCD RAM data	SEGREG007	R/W	R/W	-	-	0x00
0xFFFFFB27	LCD RAM data	SEGREG027	R/W	R/W	-	-	0x00
0xFFFFFB28	LCD RAM data	SEGREG008	R/W	R/W	-	-	0x00
0xFFFFFB28	LCD RAM data	SEGREG028	R/W	R/W	-	-	0x00
0xFFFFFB29	LCD RAM data	SEGREG009	R/W	R/W	-	-	0x00
0xFFFFFB29	LCD RAM data	SEGREG029	R/W	R/W	-	-	0x00
0xFFFFFB30	LCD RAM data	SEGREG010	R/W	R/W	-	-	0x00
0xFFFFFB30	LCD RAM data	SEGREG030	R/W	R/W	-	-	0x00
0xFFFFFB31	LCD RAM data	SEGREG011	R/W	R/W	-	-	0x00
0xFFFFFB31	LCD RAM data	SEGREG031	R/W	R/W	-	-	0x00
0xFFFFFB32	LCD RAM data	SEGREG012	R/W	R/W	-	-	0x00
0xFFFFFB33	LCD RAM data	SEGREG013	R/W	R/W	-	-	0x00

Table B-2 Other special function registers (12/13)

Address	Register name	Shortcut	1	8	16	32	Initial value
0xFFFFFB34	LCD RAM data	SEGREG014	R/W	R/W	-	-	0x00
0xFFFFFB35	LCD RAM data	SEGREG015	R/W	R/W	-	-	0x00
0xFFFFFB36	LCD RAM data	SEGREG016	R/W	R/W	-	-	0x00
0xFFFFFB37	LCD RAM data	SEGREG017	R/W	R/W	-	-	0x00
0xFFFFFB38	LCD RAM data	SEGREG018	R/W	R/W	-	-	0x00
0xFFFFFB39	LCD RAM data	SEGREG019	R/W	R/W	-	-	0x00
0xFFFFFB40	LCD RAM data	SEGREG032	R/W	R/W	-	-	0x00
0xFFFFFB41	LCD RAM data	SEGREG033	R/W	R/W	-	-	0x00
0xFFFFFB42	LCD RAM data	SEGREG034	R/W	R/W	-	-	0x00
0xFFFFFB43	LCD RAM data	SEGREG035	R/W	R/W	-	-	0x00
0xFFFFFB44	LCD RAM data	SEGREG036	R/W	R/W	-	-	0x00
0xFFFFFB45	LCD RAM data	SEGREG037	R/W	R/W	-	-	0x00
0xFFFFFB46	LCD RAM data	SEGREG038	R/W	R/W	-	-	0x00
0xFFFFFB47	LCD RAM data	SEGREG039	R/W	R/W	-	-	0x00
0xFFFFFCA0	Self-programming enable control register	SELFEN	R/W	R/W	-	-	0x00
0xFFFFFCA2	Stand-by control register	STBCTL	R/W	R/W			0x00
0xFFFFFCA8	Self-programming enable protection register	SELFENP	-	W	-	-	undefined
0xFFFFFCAA	Stand-by control protection register	STBCTLP	-	W			undefined
0xFFFFFCB0	CLMM write protection register	PRCMDCMM	-	W	-	-	undefined
0xFFFFFCB2	CLMS write protection register	PRCMDCMS	-	W	-	-	undefined
0xFFFFFD00	CSIB0 control register 0	CB0CTL0	R/W	R/W	-	-	0x01
0xFFFFFD01	CSIB0 control register 1	CB0CTL1	R/W	R/W	-	-	0x00
0xFFFFFD02	CSIB0 control register 2	CB0CTL2	-	R/W	-	-	0x00
0xFFFFFD03	CSIB0 status register	CB0STR	R/W	R/W	-	-	0x00
0xFFFFFD04	CSIB0 received data register	CB0RX0	-	-	R	-	0x0000
0xFFFFFD04	CSIB0 received data register low byte	CB0RX0L	-	R	-	-	0x00
0xFFFFFD06	CSIB0 send data register	CB0TX0	-	-	R/W	-	0x0000
0xFFFFFD06	CSIB0 send data register low byte	CB0TX0L	-	R/W	-	-	0x00
0xFFFFFD10	CSIB1 control register 0	CB1CTL0	R/W	R/W	-	-	0x01
0xFFFFFD11	CSIB1 control register 1	CB1CTL1	R/W	R/W	-	-	0x00
0xFFFFFD12	CSIB1 control register 2	CB1CTL2	-	R/W	-	-	0x00
0xFFFFFD13	CSIB1 status register	CB1STR	R/W	R/W	-	-	0x00
0xFFFFFD14	CSIB1 received data register	CB1RX0	-	-	R	-	0x0000
0xFFFFFD14	CSIB1 received data register low byte	CB1RX0L	-	R	-	-	0x00
0xFFFFFD16	CSIB1 send data register	CB1TX0	-	-	R/W	-	0x0000
0xFFFFFD16	CSIB1 send data register low byte	CB1TX0L	-	R/W	-	-	0x00
0xFFFFFD80	IIC0 shift register	IIC0	-	R/W	-	-	0x00
0xFFFFFD82	IIC0 control register	IICC0	R/W	R/W	-	-	0x00
0xFFFFFD83	IIC0 Slave address register	SVA0	-	R/W	-	-	0x00
0xFFFFFD84	IIC0 combined IICCL0 and IICX0 register	IICCL0IICX0	-	-	R/W	-	0x0000
0xFFFFFD84	IIC0 clock selection register	IICCL0	R/W	R/W	-	-	0x00

Table B-2 Other special function registers (13/13)

Address	Register name	Shortcut	1	8	16	32	Initial value
0xFFFFFD85	IIC0 function expansion register	IICX0	R/W	R/W	-	-	0x00
0xFFFFFD86	IIC0 state register	IICS0	R	R	-	-	0x00
0xFFFFFD87	IIC0 state register (for emulation only)	IICSE0	R	R	-	-	0x00
0xFFFFFD8A	IIC0 flag register	IICF0	R/W	R/W	-	-	0x00
0xFFFFFDA0	Clock selection register odd prescaler 0	OCKS0	R/W	R/W	-	-	0x00
0xFFFFFDB0	Clock selection register odd prescaler 1	OCKS1	R/W	R/W	-	-	0x00
0xFFFFFDC0	Pre-scalar mode register	PRSM0	R/W	R/W	-	-	0x00
0xFFFFFDC1	Pre-scalar compare register	PRSCM0	R/W	R/W	-	-	0x00
0xFFFFFDE0	Pre-scalar mode register	PRSM1	R/W	R/W	-	-	0x00
0xFFFFFDE1	Pre-scalar compare register	PRSCM1	R/W	R/W	-	-	0x00
0xFFFFFDF0	Pre-scalar mode register	PRSM2	R/W	R/W	-	-	0x00
0xFFFFFDF1	Pre-scalar compare register	PRSCM2	R/W	R/W	-	-	0x00
0xFFFFFF20	Reset Source Flag register	RESSTAT	R/W	R/W	-	-	0x02/0x01
0xFFFFFF22	Software reset register	RESSWT	W	W	-	-	0x00
0xFFFFFF24	Software reset enable register	RESCMD	W	W	-	-	0x00
0xFFFFFF26	Reset status register	RES	-	R/W	-	-	0x00

Revision History

The following revision list shows all functional changes of this document R01UH0027ED0420 compared to the previous manual version R01UH0027ED0300.

Chapter	Page	Description
4	120	sub chapter of SSCG control registers corrected (mistakenly inserted twice; former sub chapter 4.2.2 removed)
4	135	Bit position number in STBCTL register contents table corrected
4	148	status of WTCLK/LCDCLK in clock generator status table for STOP mode changed
4	148	INTWT0UV, INTWT1UV added to list of maskable interrupts, which can release the STOP mode
5	170	description of prerequisites to enable interrupt servicing during self-programming simplified
17	545	data bit names of SDA0n data stream corrected in figure
21	724	LCD display control register name corrected
21	725	
21	731	

The following revision list shows all functional changes of this document R01UH0027ED0420 compared to the previous manual version R01UH0027ED0400.

Chapter	Page	Description
1	28	document number of data sheet updated
6	200	addresses of higher 8-bit registers IMRmH corrected
6	200	IMR6 register from address list removed
7	225	VSWC setting changed for system clock higher than 16 MHz
18	549	MAC (memory acces controller) replaced by MCM (message control module) according to block diagram of CAN module

The following revision list shows all functional changes of this document R01UH0027ED0420 compared to the previous manual version R01UH0027ED0410.

Chapter	Page	Description
2	72	assignment of pins 87 and 88 corrected (pin 87 = FLMD0, pin 88 = X1)
3	98	mistakenly inserted instruction and data access times of V850E/DL3, V850E/DL3 removed

Index

V850E/Dx3 - DG3 User Manual

Publication Date: Rev. 4.20 Feb 15, 2013

Published by: Renesas Electronics Corporation

SALES OFFICES

Renesas Electronics Corporation

http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2880 Scott Boulevard Santa Clara, CA 95050-2554, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-65030, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 204, 205, AZIA Center, No.1233 Lujiazui Ring Rd., Pudong District, Shanghai 200120, China
Tel: +86-21-5877-1818, Fax: +86-21-6887-7858 / -7898

Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2886-9318, Fax: +852 2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.
7F, No. 363 Fu Shing North Road Taipei, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
1 harbourFront Avenue, #06-10, keppel Bay Tower, Singapore 098632
Tel: +65-6213-0200, Fax: +65-6278-8001

Renesas Electronics Malaysia Sdn.Bhd.
Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics Korea Co., Ltd.
11F., Samik Lavied' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 1.0

V850E/Dx3 - DG3

Renesas Electronics Corporation

R01UH0027ED0420, Rev. 4.20