NEC

User's Manual

V850E/RG3

Documentation

μPD70F3464
μPD70F3465
μPD70F3466
μPD70F3470
μPD70F3471
μPD70F3472

Document No. U19131EU3V0UM01
Date Published October 2009

Printed in Germany

Notes for CMOS Devices

(1) Precaution against ESD for semiconductors

Strong electric field, when exposed to a MOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Therefore, steps must be taken to stop generation of static electricity as much as possible, and quickly dissipate it when it has occurred.

- Avoid using insulators that easily build static electricity.
- Environmental control must be adequate. When it is dry, a humidifier should be used.
- Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material.
- All test and measurement tools including work bench and floor should be grounded.
- The operator should be grounded using wrist strap.
- Semiconductor devices must not be touched with bare hands.
- Similar precautions need to be taken for PW boards with semiconductor devices on it.

(2) Handling of unused input pins for CMOS

The lack of a connection for CMOS device inputs can cause malfunction. If no connection is provided to the input pins, an internal input level may be generated due to noise, etc., hence causing malfunction. CMOS devices behave differently than Bipolar or NMOS devices. Input levels of CMOS devices must be fixed high or low by using a pull-up or pull-down circuit. Each unused pin should be connected to VDD or GND with a resistor, if it could possibly be used as an output pin. All handling related to the unused pins must be judged device by device, based on related specifications governing the devices.

(3) Status before initialization of MOS devices

Power-on does not necessarily define the initial status of MOS device. The production process of MOS does not define the initial operation status of the device. Immediately after the power source is turned ON, the devices with a reset function have not yet been initialized. Hence, power-on does not guarantee out-pin levels, I/O settings, or contents of registers. A device is not initialized until a reset signal is received. The Reset operation must be executed immediately after power-on for devices that have a reset function.

(4) Voltage application waveform at input pin

Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between VIL (MAX) and VIH (MIN) due to noise, etc., the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between VIL (MAX) and VIH (MIN).

(5) Input of signal during power off state

Do not input signals or an I/O pull-up power supply while the device is not powered. The current injection that results from input of such a signal or I/O

pull-up power supply may cause malfunction, and the abnormal current that passes in the device at this time may cause degradation of internal elements. Input of signals during the power off state must be judged separately for each device and according to related specifications governing the device.

Legal Notes

- The information in this document is current as of July 2008. The information is subject to change without notice. For actual design-in, refer to the latest publications of NEC Electronics data sheets or data books, etc., for the most up-to-date specifications of NEC Electronics products. Not all products and/or types are available in every country. Please check with an NEC sales representative for availability and additional information.
- No part of this document may be copied or reproduced in any form or by any means without prior written consent of NEC Electronics. NEC Electronics assumes no responsibility for any errors that may appear in this document.
- NEC 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 NEC Electronics products listed in this document or any other liability arising from the use of such NEC Electronics products. No license, express, implied or otherwise, is granted under any patents, copyrights or other intellectual property rights of NEC Electronics or others.
- Descriptions of circuits, software and other related information in this document are provided for illustrative purposes in semiconductor product operation and application examples. The incorporation of these circuits, software and information in the design of customer's equipment shall be done under the full responsibility of customer. NEC Electronics assumes no responsibility for any losses incurred by customers or third parties arising from the use of these circuits, software and information.
- While NEC Electronics endeavors to enhance the quality, reliability and safety of NEC Electronics products, customers agree and acknowledge that the possibility of defects thereof cannot be eliminated entirely. To minimize risks of damage to property or injury (including death) to persons arising from defects in NEC Electronics products, customers must incorporate sufficient safety measures in their design, such as redundancy, fire-containment and anti-failure features.
- NEC Electronics products are classified into the following three quality grades: “Standard”, “Special” and “Specific”.

  The "Specific" quality grade applies only to NEC Electronics products developed based on a customer-designated “quality assurance program” for a specific application. The recommended applications of an NEC Electronics product depend on its quality grade, as indicated below. Customers must check the quality grade of each NEC Electronics product before using it in a particular application.

"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.
"Special":	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, life support systems and medical equipment for life support, etc.

The quality grade of NEC Electronics products is “Standard” unless otherwise expressly specified in NEC Electronics data sheets or data books, etc. If customers wish to use NEC Electronics products in applications not intended by NEC Electronics, they must contact an NEC Electronics sales representative in advance to determine NEC Electronics's willingness to support a given application.

Note 1. "NEC Electronics" as used in this statement means NEC Electronics Corporation and also includes its majority-owned subsidiaries.

2. "NEC Electronics products" means any product developed or manufactured by or for NEC Electronics (as defined above).

SuperFlash(R) SuperFlash(R) is a registered trademark of Silicon Storage Technology, Inc. in several countries including the United States and Japan.

This product uses SuperFlash(R) technology licensed from Silicon Technology, Inc.

Regional Information

Some information contained in this document may vary from country to country. Before using any NEC product in your application, please contact the NEC office in your country to obtain a list of authorized representatives and distributors. They will verify:

- Device availability
- Ordering information
- Product release schedule
- Availability of related technical literature
- Development environment specifications (for example, specifications for third-party tools and components, host computers, power plugs, AC supply voltages, and so forth)
- Network requirements

In addition, trademarks, registered trademarks, export restrictions, and other legal issues may also vary from country to country.

[America]

NEC Electronics America, Inc.
2880 Scott Blvd.
Santa Clara, CA 95050-2554
U.S.A.
Tel: 408-588-6000
http://www.am.necel.com/

NEC Electronics America, Inc.
Automotive Division
6555 N. State Highway 161,
9th Floor
Irving, TX 75039-2402
U.S.A.
Tel: 214-262-7850

[Europe]

NEC Electronics (Europe) GmbH
Arcadiastrasse 10
40472 Düsseldorf, Germany
Tel: 0211 65030
http://www.eu.necel.com/

United Kingdom Branch
Cygnus House, Sunrise Parkway
Linford Wood, Milton Keynes
MK14 6NP, U.K.
Tel: 01908 691133

Succursale Française
9, rue Paul Dautier, B.P. 52
78142 Velizy-Villacoublay Cédex
France
Tel: 01 30675800

Sucursal en España
Juan Esplandiu, 15
28007 Madrid, Spain
Tel: 091 5042787

Tyskland Filial
Täby Centrum
Entrance S (7th floor)
18322 Täby, Sweden
Tel: 08 6387200

Filiale Italiana
Via Fabio Filzi, 25/A
20124 Milano, Italy
Tel: 02 667541

Branch The Netherlands
Steijgerweg 6
5616 HS Eindhoven,
The Netherlands
Tel: 040 2654010

[Asia & Oceania]

NEC Electronics (China) Co., Ltd
7th Floor, Quantum Plaza, No. 27
ZhiChunLu Haidian District,
Beijing 100083, P.R.China
Tel: 010 82351155
http://www.cn.necel.com/

NEC Electronics Shanghai Ltd.
Room 2511-2512, Bank of China Tower,
200 Yincheng Road Central,
Pudong New Area,
Shanghai 200120, P.R. China
Tel: 021 58885400
http://www.cn.necel.com/

NEC Electronics Hong Kong Ltd.
12/F., Cityplaza 4,
12 Taikoo Wan Road, Hong Kong
Tel: 2886 9318
http://www.hk.necel.com/

NEC Electronics Taiwan Ltd.
7F, No. 363 Fu Shing North Road
Taipei, Taiwan, R.O.C.
Tel: 02 27192377

NEC Electronics Singapore Pte. Ltd.
238A Thomson Road,
#12-08 Novena Square,
Singapore 307684
Tel: 6253 8311
http://www.sg.necel.com/

NEC Electronics Korea Ltd.
11F., Samik Lavied'or Bldg., 720-2,
Yeoksam-Dong, Kangnam-Ku, Seoul,
135-080, Korea Tel: 02-558-3737
http://www.kr.necel.com/

Preface

Readers This manual is intended for users who want to understand the functions of the microcontrollers described in it.

Purpose This is a hardware manual presenting the system specifications for the microcontrollers described in it.

Preliminary draft This document is an incomplete, preliminary draft of the hardware manual for V850E/RG3. A more complete and comprehensive version is forthcoming in the first quarter of 2008.

Organization This system specification is organized by the following sections:

- Pin function
- CPU function
- Internal peripheral function

Module instances These microcontrollers may contain several instances of a dedicated module. In general, the different instances of such modules are identified by the index “n”, where “n” counts from 0 to the number of instances minus one.

Legend Symbols and notations are used as follows:

- Weight in data notation: Left is high order column, right is low order column
- Active low notation: $\overline{xxx}$ (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

Note An additional remark or tip.

Caution An item deserving extra attention.

Numeric notation:

- Binary: xxxx or xxx_B
- Decimal: xxxx
- Hexadecimal: $xxxx_H$ or 0x xxxx

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 : any example

Diagrams Block diagrams do not necessarily show the exact wiring in hardware; rather, they show the functional structure.

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

Further Information For further information see *http://www.necel.com*.

Table of Contents

Chapter 1 Introduction

The V850E/RG3 is a product line in NEC Electronics' V850 family of single-chip microcontrollers designed for automotive applications.

1.1 General

The V850E/RG3 single-chip microcontroller device makes the performance gains attainable with 32-bit RISC-based controllers available for embedded control applications. The integrated V850E CPU offers easy pipeline handling and programming, resulting in compact code size comparable to 16-bit CISC CPUs.

The V850E/RG3 devices provide an excellent combination of general purpose peripheral functions like serial communication interfaces, timers/counters, measurement and control functions, with full CAN network support.

Thus equipped, the V850E/RG3 product line is ideally suited for automotive safety applications. It is also an excellent choice for other applications where a combination of sophisticated peripheral functions and also CAN network support is required.

V850E CPU

The V850E CPU core is a 32-bit 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 supports 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, the devices are well suited for high-level real-time control applications.

On-chip flash memory

The V850E/RG3 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.

Reliability

Focus of the design was to achieve the utmost reliability and availability. Flash memory, RAM interface is equipped with error correction code (ECC). ECC makes it possible to correct single bit errors automatically and to detect and flag double bit errors.

Software development tools

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

1.2 Features Summary

The V850E/RG3 series includes the following microcontrollers:

V850E/RG3

- 3V devices
  - uPD70F3464GC(A9)-UEU
  - uPD70F3465GC(A9)-UEU
  - uPD70F3466GC(A9)-UEU
- 5V devices
  - uPD70F3470GC(A1)-UEU
  - uPD70F3471GC(A1)-UEU
  - uPD70F3472GC(A1)-UEU

V850E/RG3 family features

- 32-bit RISC CPU with Harvard Architecture
- Frequencies
  - Main CPU frequency up to 80 MHz
- Internal Flash: Up to 512 KB
- Internal RAM: Up to 32 KB
- Data Flash: 32 KB
- Full-CAN Interface
- Serial Interfaces:
  - 3-wire SPI : CSIF
  - Queued SPI : CSIE
  - UART (LIN compatible)
- Timers
  - 16-bit capture/compare timer
  - Internal timers
  - Watchdog timer
- AD converter
  - Up to 16 channels, 10-bit resolution
- I/O lines: 56 I/Os + 1 input
- Main oscillator:
  - 8 MHz for 5V devices
  - 16 MHz for 3V devices

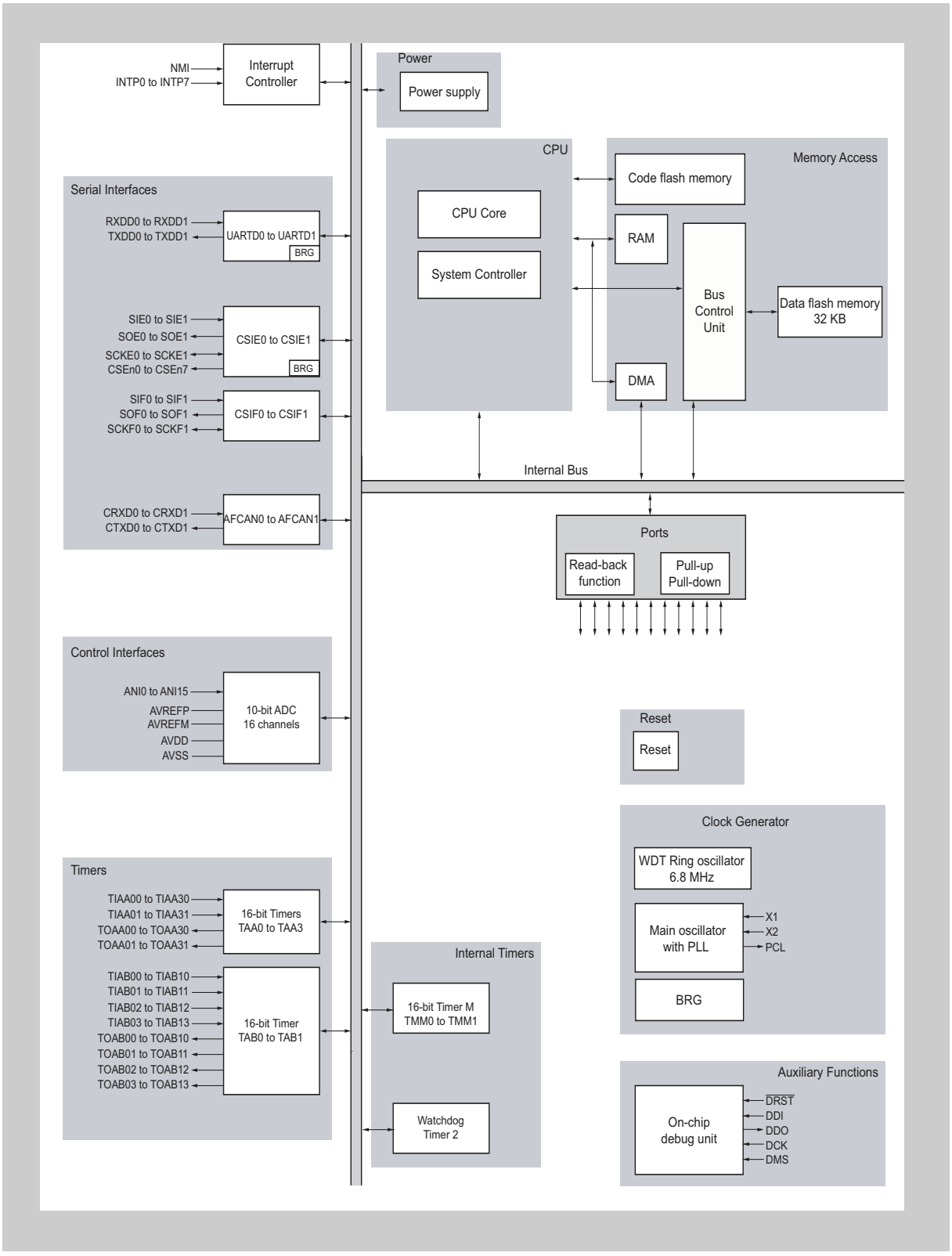
Table 1-1 V850E/RG3 features

Series name	Series name	V850E/RG3			V850E/RG3		
Product	Product	70F3464	70F3465	70F3466	70F3470	70F3471	70F3472
CPU	Core	V850E (32bit RISC)			V850E (32bit RISC)		
Internal Memory	Code Flash	256KB	384KB	512KB	256KB	384KB	512KB
	RAM	16KB	24KB	32KB	16KB	24KB	32KB
	DATA FLASH	32KB			32KB		
Operating Clock	Max CPU frequency	64MHz	64MHz	80MHz	80MHz		
	Main Osc	16MHz			8MHz		
	Programmable Clock Output	1ch			1ch		
Timers	TAA	4ch			4ch		
	TAB	2 ch	2 ch	2 ch	2 ch		
	TMM	2ch			2ch		
	WDT	1ch			1ch		
	WDT Ring Osc	212kHz			212kHz		
Serials interfaces	AFCAN (32msg buffer)	2 ch			2 ch		
	CSIE (queued SPI)	2 ch			2 ch		
	CSIF	2 ch			2 ch		
	UART/LIN	2 ch			2ch		
DMA		7 ch			7 ch		
AD converter	AD voltage	3.3V			5 V		
	Accuracy	10 bits			10 bits		
	Channels	16 ch			16 ch		
	Discharge & diagnostic mode	yes			yes		
I/Os	I/Os ports (I/O + input)	56+1	56+1	56+1	56+1	56+1	56+1
	Pull Up / Pull Down	Depend on pin function			Depend on pin function		
	Read-back function	All ports			All ports		
	I/O voltage	3.3 V			5 V		
On-Chip-Debug		yes			yes		
Package		100 pins QFP			100 pins QFP		
Temperature Range		-40°C to +105°C			-40°C to +110°C		

1.3 Description

The following figure provides a functional block diagram of the V850E/RG3 microcontroller.

Figure 1-1 V850E/RG3 block diagram



1.3.1 Internal units

CPU The CPU can execute almost all instruction processing, such as address calculation, arithmetic and logic operations, and data transfer, in one clock under control of a five-stage pipeline.

Dedicated hardware units such as a multiplier and a 32-bit barrel shifter are provided to speed up complicated instruction processing.

Bus Control Unit The Bus Control Unit (BCU) and Memory Controller (MEMC) control the access to on-chip peripheral I/Os, data flash, and the CRC.

ROM The ROM consists of an internal flash memory. It is divided into code flash and data flash.

RAM Internal RAM.

DMA Controller The V850E/RG3 has a seven channel DMA Controller that transfers data between the internal RAM and I/O.

Ports General-purpose port functions and control pin functions are available.

Clock Generator The Clock Generator generates the system clocks. It provides clock signal to CPU and peripherals.

On-chip Debug function An on-chip debug function that uses the N-Wire interface is provided.

Interrupt Controller The Interrupt Controller (INTC) processes non-maskable and maskable interrupt requests from the on-chip peripheral hardware and external sources. Eight levels of priorities can be specified for these interrupt requests, and multiple servicing control can be performed on interrupt sources.

UARTD The UARTs provide 2-wire Asynchronous Serial Interfaces that support LIN.

CSIF The Clocked Serial Interfaces are 3-wire variable-length serial interfaces.

CSIE The Enhanced Queued Clocked Serial Interfaces provide an internal 16-word FIFO buffer for queued operation, and automatically generated Chip Select signals with independent programmable timings.

CAN Controller The CAN Controller is a small-scale digital data transmission system that transfers data between units.

A/D Converter This is a high-speed, high-resolution 10-bit A/D Converter with 16 analog input pins. This converter is of the successive approximation type.

Timers/counters Four 16-bit timers/event counters TAA, two 16-bit timers/event counters TAB, and two 16-bit interval timer TMM are provided.

Watchdog Timer 2 The Watchdog Timer (WDT) is used to detect a program loop and system errors. When the Watchdog Timer overflows, it generates a non-maskable interrupt request signal or a system reset signal.

Chapter 2 Pin Functions

This chapter lists the ports of the microcontroller. It presents the configuration of the ports for control functions.

2.1 Overview

The microcontroller offers various pins for input/output functions, otherwise 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 *Section 2.2.1, Terms* on page 10.

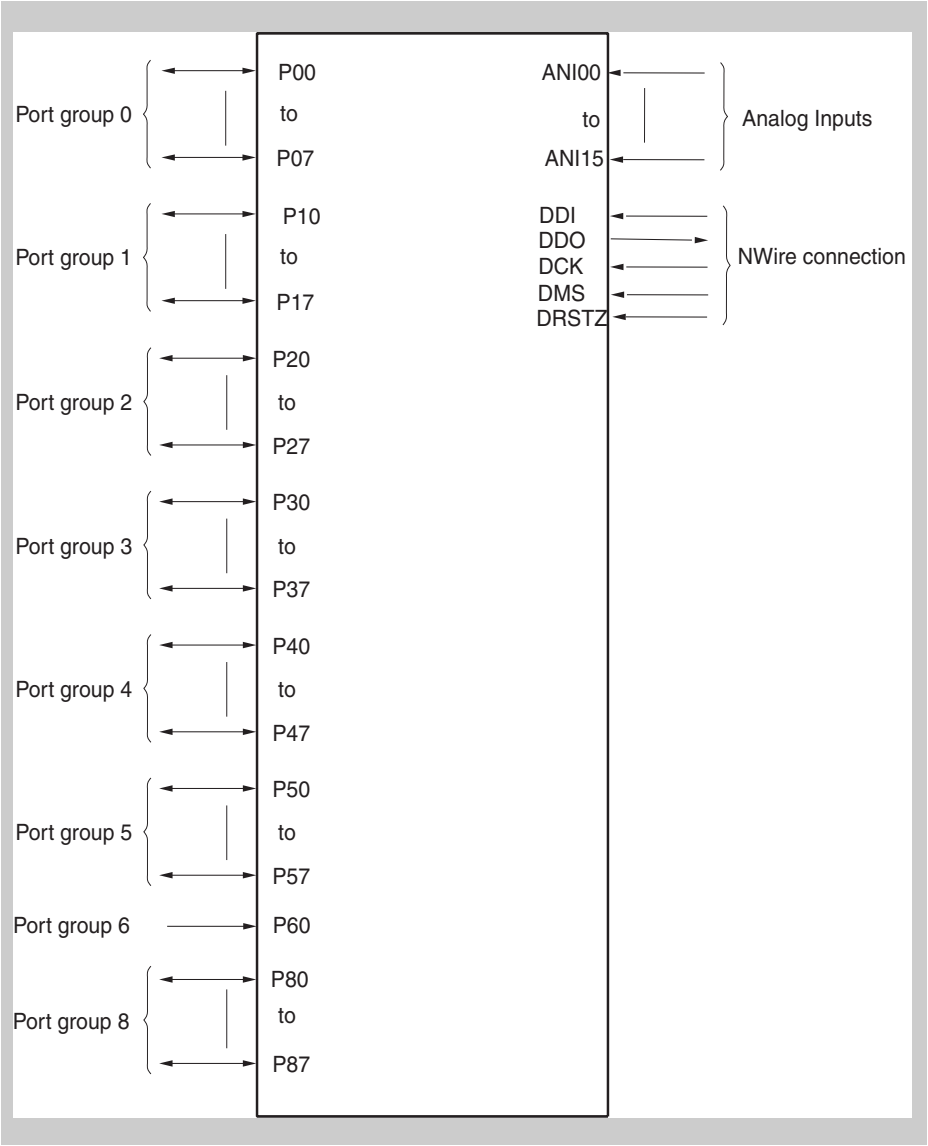
Features summary

- Number of ports and port groups:
  - Input/Output ports: 56
  - Input port: 1
  - Port groups: 8 + 1 analog
- Configuration possible for individual pins.
- For many pins, the internal connection of a pull-up or pull down resistor can be selected.
- The 3V devices (µPD70F3464, µPD70F3465 and µPD70F3466) allow connecting the pull-up or pull-down resistors during RESET mode.

2.2 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[a]	
	Port mode	Alternative mode
0	8-bit input/output	• Timer TAB0 channels • Timer TAB1 channels • PCL
1	8-bit input/output	• Timer TAB1 channels • CSIE0 Chip Select (4 to 7) • UARTD0 receive/transmit data
2	8-bit input/output	• Timer TAB0 channels • Timer TAB1 channels • CSIE0 and CSIE1 input, output, clock
3	8-bit input/output	• CSIE1 Chip Select
4	8-bit input/output	• CSIF0 input, output, clock • CSIF1 input, output, clock • CAN0 receive/transmit data
5	8-bit input/output	• Timer TAA0 channel • Timer TAA1 channel • Timer TAA2 channel • Timer TAA3 channel
6	1-bit input	• NMI • $\overline{\text{PUDSEL}}$[b]
8	8-bit input/output	• UARTD1 receive/transmit data • CAN1 receive/transmit data • CSIE0 Chip Select (0 to 3)
AN	16-bit analog input	• Analog inputs
N-Wire	5 input/output	• DDI : N-wire debug data input • DCK : N-wire interface clock • DMS : N-wire mode • DDO : N-wire debug data output • DRSTZ : N-wire RCU reset

a) Availability of some of the peripheral macros such as TAB, UARTD, aFCAN depends on the device.

b) Not available on µPD70F3470, µPD70F3471 and µPD70F3472

Pin configuration

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

- For a general description of the registers, see *Section 2.3, Port Group Configuration Registers* on page 11.
- For every port, detailed information on the configuration registers is given in *Section 2.4, Port Group Configuration* on page 17.

There are several types of control circuits, defined as port types. For a description of the port types, see *Section 2.4, Port Group Configuration* on page 17.

2.2.1 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 P01 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”.

2.3 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:

- *Section 2.3.2, Pin function configuration* on page 11
- *Section 2.3.3, Pin data input/output* on page 13

2.3.1 Overview

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

μPD70F3472

Table 2-2 Registers for port group configuration

Register name	Shortcut	Function
Port mode control register	PMCn	Pin function configuration
Port mode register	PMn	
Port function control register	PFCn	
Port register	Pn	Pin data input/output
Port pin read register	PPRn	
Port pull-up resistor option register[a]	PEn[a]	Pull-up/down enable/disable[a]
Port pull-up resistor option register [b]	PUn[b]	Pull-up enable/disable[b]
Port pull-down resistor option register[b]	PDn[b]	Pull-down enable/disable[b]

a) Available only on μPD70F3464, μPD70F3465 and μPD70F3466.
b) Available only on μPD70F3470, μPD70F3471 and μPD70F3472.

n = 0 to 6, 8

2.3.2 Pin function configuration

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

- port mode or alternative mode
- in port mode: input mode or output mode
- in alternative mode: selection of one of the alternative functions in alternative mode

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

Table 2-3 Pin function configuration (overview)

Function	Registers			I/O
	PMCn	PMn	PFCn	
Port mode (output)	0	0	X	O
Port mode (input)		1	X	I
Alternative mode (alternative function 1)	1	X	0	I/O[a]
Alternative mode (alternative function 2)			1	

a) In alternative mode, the corresponding port type defines whether a pin is in input mode or output mode.

(1) 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.

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

Address see *Section 2.4, Port Group Configuration* on page 17

Initial Value 00_H. This register is initialized by $\overline{\text{RESET}}$.

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

Table 2-4 PMCn register contents

Bit name	Function
PMCn[7:0]	Specifies the operation mode of the corresponding pin 0: Port mode 1: Alternative mode

Caution When changing the function of a port from port mode (PCMnm = 0) to external interrupt input (PCMnm = 1) an advertent interrupt may occur.

Therefore, it is recommended to follow the below procedure:

1. Set PMCnm = 1 to change to the alternative mode.
2. Wait until the delay of the noise rejection filter has passed.
3. Set INTnIC.INTnIF = 0 to clear the interrupt request.
4. Clear INTnIC.INTnMK (or clear INTMR.INTnMK) to enable the interrupt.

In step 3 you must wait for a certain time span because the external interrupt pins are equipped with noise rejection filters. The filters cause a delay in which the interrupt request flag INTnIC.INTnIF is set. This flag must be cleared (step 4).

(2) PMn - Port mode register

If a pin is in port mode (PMCn.PMCnm = 0), 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.

Note If a pin is in alternative mode (PMCn.PMCnm = 1) and the corresponding PMn bit is set (PMn.PMnm = 1), then the pin behaves as in input port mode: Reading Pn.Pmn reads the pin status.

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

Address See *Section 2.4, Port Group Configuration* on page 17

Initial Value FF_H. This register is initialized by $\overline{\text{RESET}}$.

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

Table 2-5 PMn register contents

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

(3) PFCn - Port function control register

If a pin is in alternative mode (PMCn.PMCnm = 1) some pins offer up to four alternative functions.

The 8-bit PFCn register together with the 8-bit register specifies which function of a pin is to be used. The corresponding port type defines whether a pin is in input or output mode.

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

Address see *Section 2.4, Port Group Configuration* on page 17

Initial Value 00_H

This register is initialized by $\overline{\text{RESET}}$.

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 name	Function
PFCn[7:0]	See *Table 2-3* on page 11 for details

2.3.3 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

If a pin is 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.

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

Address see *Section 2.4, Port Group Configuration* on page 17

Initial Value 00_H. This register is cleared by $\overline{\text{RESET}}$.

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

Table 2-7 Pn register contents

Bit name	Function
Pn[7:0]	Data, see *Table 2-8* on page 14 and *Table 2-9* on page 14 for details.

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

Port mode In port mode (PMCn.PMCnm = 0), register PMn specifies whether a pin is in input or in output mode. Data is written to or read from the Pn register as follows:

Table 2-8 Writing/reading register Pn in port mode (PMCn.PMCnm = 0)

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

Alternative mode In alternative mode (PMCn.PMCnm = 1), the corresponding port type defines whether a pin is in input or output mode. However, register PMn influences the writing/reading of register Pn.

In alternative mode, data is written to or read from the Pn register as follows.

Table 2-9 Writing/reading register Pn in alternative mode (PMCn.PMCnm = 1)

Function	PM	I/O
Write to Pn without affecting the pin status	X	–
Read from Pn…		
…and read the value of the alternative output function (for pins in alternative output function)	0	–
…and disregard the pin status (for pins in alternative input function)		
…and thus read the pin status	1	I

Caution Although 1-bit operations (read-modify-write operations) on Pn registers are intended to modify only a single bit, the entire Pn register is read. After the single bit has been modified, the contents of the complete register is written back.

If the ports of the register Pn contain both input and output ports Pnm, the read of Pn returns

- the contents of the register Pn for output ports
- the pin status of input ports, but not the Pn register bits

That means the read value of Pn may be different to the contents of the Pn register at bit positions, which are assigned to input ports.

Thus the contents of Pn may differ to the previous value not just in the bit that was to be modified, but also in other bits.

Example:

- Register P1 has the contents 00_H.
- Port P10 is configured as an output port, all other ports of port group 1 (ports P11 to P17) are configured as input ports.
- The port pins of ports P11 to P17 all have the level “1”.
- Bit P1.P10 is set to 1 by a 1-bit operation.

Afterwards, register P1 holds the value FF_H instead of the expected value 01_H, since bits P11 to P17 have be overwritten with the corresponding pin levels “1”.

(2) PPRn - Port pin read register

The PPRn register reflects the actual pin value, independent of the configuration of the control registers.

Access This register is read-only, in 8-bit and 1-bit units.

Address

PPR0 = FFFF $F700_H$	PPR4 = FFFF $F708_H$
PPR1 = FFFF $F702_H$	PPR5 = FFFF $F70A_H$
PPR2 = FFFF $F704_H$	PPR6 = FFFF $F70C_H$
PPR3 = FFFF $F706_H$	PPR8 = FFFF $F710_H$

Initial Value 00_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

Table 2-10 PPRn register contents

Bit name	Function
PPRn[7:0]	Actual pin value

2.3.4 Configuration of pull-up and pull-down resistors (3V devices)

This section applies to the following devices only:

- µPD70F3464
- µPD70F3465
- µPD70F3466

(1) Pull-up and pull-down activation in RESET state

With the devices µPD70F3464, µPD70F3465 and µPD70F3466, it is possible to enable the pull-up and pull-down resistors during the RESET mode. The state of the PU/PD is held after the RESET release.

The value after RESET of the registers PEn depends on the value of $\overline{\text{PUDSEL}}$ at RESET release.

Table 2-11 PU/PD mode selection

PUDSEL	Function	Reset value of PEn registers
0	PU/PD enabled during after reset	FF_H
1	PU/PD disabled during after reset	00_H

(2) PEn - Port pull-up and pull-down resistor option register

The PEn register specifies whether a pull-up or pull down resistor is connected to the pin.

Each port group comprises one or more ports. Most ports have either an internal pull-up or a pull-down resistor associated with them. For each port group, a configuration register is used to enable or disable the pull-up/pull-down resistor for each port of the port group.

The bit names of the PEn registers depend on whether the corresponding port has a pull-up or a pull-down. For ports with pull-ups, the corresponding bit in the PEn register is called PUnm. For ports with pull-downs, the corresponding bit in the PEn register is called PDnm.

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

Address See *Section 2.4, Port Group Configuration* on page 17

Initial Value 00_H. This register is cleared by $\overline{\text{RESET}}$.

7	6	5	4	3	2	1	0
Pxn7	Pxn6	Pxn5	Pxn4	Pxn3	Pxn2	Pxn1	Pxn0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Pxm stands for PUm in case of a port with a pull-up and PDm in case of a port with a pull-down (m = 0 to 7)

Table 2-12 PEn register contents

Bit name	Function
PUn[7:0]	Specifies whether a pull-up resistor is connected to the corresponding pin: 0: no pull-up resistor connected 1: pull-up resistor connected
PDn[7:0]	Specifies whether a pull-down resistor is connected to the corresponding pin: 0: no pull-down resistor connected 1: pull-down resistor connected

2.3.5 Configuration of pull-up and pull-down resistors (5V devices)

This section applies to the following devices only:

- µPD70F3470
- µPD70F3471
- µPD70F3472

(1) PUn - Port pull-up resistor option register

The PUn register specifies whether a pull-up resistor is connected to the pin.

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

Address See *Section 2.4, Port Group Configuration* on page 17

Initial Value 00_H. This register is cleared by $\overline{\text{RESET}}$.

7	6	5	4	3	2	1	0
PUn7	PUn6	PUn5	PUn4	PUn3	PUn2	PUn1	PUn0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 2-13 PUn register contents

Bit name	Function
PUn[7:0]	Specifies whether a pull-up resistor is connected to the corresponding pin: 0: no pull-up resistor connected 1: pull-up resistor connected

(2) PDn - Port pull-down resistor option register

The PDn register specifies whether a pull-down resistor is connected to the pin.

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

Address See *Section 2.4, Port Group Configuration* on page 17

Initial Value 00_H. This register is cleared by $\overline{\text{RESET}}$.

7	6	5	4	3	2	1	0
PDn7	PDn6	PDn5	PDn4	PDn3	PDn2	PDn1	PDn0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 2-14 PDn register contents

Bit name	Function
PDn[7:0]	Specifies whether a pull-down resistor is connected to the corresponding pin: 0: no pull-down resistor connected 1: pull-down resistor connected

2.4 Port Group Configuration

This section provides an overview of the port groups (P0 to P8) and of the pin functions (*Table 2-16* on page 19). *Table 2.5* on page 34 lists 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. These begin with *Section 2.4.4, Port group 0* on page 26.

2.4.1 Port group configuration lists

Table 2-15 provides an overview of the functions available at each port pin.

Table 2-15 Port group list[a] (1/2)

Port group name	Pin name	Alternative inputs	Alternative outputs	Buffer Type	
				3V devices[b]	5V devices[c]
0	P00	TIAB13	TOAB13	5-AP	5-AF
	P01	FLMD1	–	5-AP	5-W
	P02	INTP0	–	5-AP	5-AF
	P03	–	$\overline{\text{RESOUT}}$	5-AK	5-AF
	P04	INTP1	PCL	5-AP	5-AF
	P05	TIAB01	TOAB01	5-AP	5-AF
	P06	TIAB02 / INTP6	TOAB02	5-AP	5-AF
	P07	TIAB03 / INTP7	TOAB03	5-AP	5-AF
1	P10	TIAB11	TOAB11	5-AP	5-AF
	P11	TIAB12	TOAB12	5-AP	5-AF
	P12	–	CSE04	5-W	5-AF
	P13	–	CSE05	5-W	5-AF
	P14	–	CSE06	5-W	5-AF
	P15	–	CSE07	5-W	5-AF
	P16	INTP2	TXDD0	5-W	5-AF
	P17	RXDD0 / INTP3	–	5-W	5-AF
2	P20	TIAB10	TOAB10	5-AP	5-AF
	P21	TIAB00	TOAB00	5-AP	5-AF
	P22	SIE0	–	5-W	5-AF
	P23	SCKE0	–	5-W	5-AF
	P24	–	SOE0	5-W	5-AF
	P25	–	SOE1	5-W	5-AF
	P26	SCKE1	SCKE1	5-W	5-AF
	P27	SIE1	–	5-W	5-AF
3	P30	–	CSE10	5-W	5-AF
	P31	–	CSE11	5-W	5-AF
	P32	–	CSE12	5-W	5-AF
	P33	–	CSE13	5-W	5-AF
	P34	–	CSE14	5-W	5-AF
	P35	–	CSE15	5-W	5-AF
	P36	–	CSE16	5-W	5-AF
	P37	–	CSE17	5-W	5-AF
4	P40	SIF0	–	5-AP	5-AF
	P41	SCKF0	SCKF0	5-AP	5-AF
	P42	–	SOF0	5-AP	5-AF
	P43	SCKF0	SCKF0	5-AP	5-AP
	P44	–	SOF1	5-AP	5-AP
	P45	SIF1	–	5-AP	5-AP
	P46	–	CTXD0	5-W	5-AF
	P47	CRXD0	–	5-W	5-AF

Table 2-15 Port group list[a] (2/2)

Port group name	Pin name	Alternative inputs	Alternative outputs	Buffer Type	
				3V devices[b]	5V devices[c]
5	P50	TIAA00	TOAA00	5-AP	5-AF
	P51	TIAA01	TOAA01	5-AP	5-AF
	P52	TIAA10	TOAA10	5-AP	5-AF
	P53	TIAA11	TOAA11	5-AP	5-AF
	P54	TIAA21	TOAA21	5-AP	5-AF
	P55	TIAA20	TOAA20	5-AP	5-AF
	P56	TIAA30	TOAA30	5-AP	5-AF
	P57	TIAA31	TOAA31	5-AP	5-AF
6	P60	NMI / $\overline{\text{PUDSEL}}$[b]	–	2-B	2-G
8	P80	INTP4	TXDD1	5-W	5-AF
	P81	RXDD1 / INTP5	–	5-W	5-AF
	P82	–	CTXD1	5-W	5-AF
	P83	CRXD1	–	5-W	5-AF
	P84	–	CSE03	5-W	5-AF
	P85	–	CSE02	5-W	5-AF
	P86	–	CSE01	5-W	5-AF
	P87	–	CSE00	5-W	5-AF

a) Availability of some of the peripheral macros such as TAB, UART, aFCAN depends on the device.
b) Only valid for the devices: µPD70F3464, µPD70F3465 and µPD70F3466.
c) Only valid for the devices: µPD70F3470, µPD70F3471 and µPD70F3472.

2.4.2 Alphabetic pin function list

Table 2-16 provides a list of all pin function names in alphabetic order.

Table 2-16 Alphabetical pin function list (1/4)

Pin name[a]	# Pin	I/O	3V devices[b]		5V devices[c]		
			PU/PD	Buffer Type	PU	PD	Buffer Type
SCKE0	44	I/O	PU	5-W	tbd.	tbd.	5-W
SCKE1	49	I/O	PU	5-W	tbd.	tbd.	5-W
SCKF0	68	I/O	PD	5-AP	tbd.	tbd.	5-W
SCKF1	72	I/O	PD	5-AP	tbd.	tbd.	5-AP
ANI00	100	–	–	7-B	tbd.	tbd.	7-B
ANI01	99	–	–	7-B	tbd.	tbd.	7-B
ANI02	98	–	–	7-B	tbd.	tbd.	7-B
ANI03	97	–	–	7-B	tbd.	tbd.	7-B
ANI04	96	–	–	7-B	tbd.	tbd.	7-B
ANI05	95	–	–	7-B	tbd.	tbd.	7-B
ANI06	94	–	–	7-B	tbd.	tbd.	7-B
ANI07	93	–	–	7-B	tbd.	tbd.	7-B
ANI08	92	–	–	7-B	tbd.	tbd.	7-B

Table 2-16 Alphabetical pin function list (2/4)

Pin name[a]	# Pin	I/O	3V devices[b]		5V devices[c]		
			PU/PD	Buffer Type	PU	PD	Buffer Type
ANI09	91	–	–	7-B	tbd.	tbd.	7-B
ANI10	90	–	–	7-B	tbd.	tbd.	7-B
ANI11	89	–	–	7-B	tbd.	tbd.	7-B
ANI12	88	–	–	7-B	tbd.	tbd.	7-B
ANI13	87	–	–	7-B	tbd.	tbd.	7-B
ANI14	86	–	–	7-B	tbd.	tbd.	7-B
ANI15	85	–	–	7-B	tbd.	tbd.	7-B
CRXD0	76	I	PU	5-W	tbd.	tbd.	5-AF
CRXD1	36	I	PU	5-W	tbd.	tbd.	5-AF
CSE00	40	O	PU	5-W	tbd.	tbd.	5-AF
CSE01	39	O	PU	5-W	tbd.	tbd.	5-AF
CSE02	38	O	PU	5-W	tbd.	tbd.	5-AF
CSE03	37	O	PU	5-W	tbd.	tbd.	5-AF
CSE04	25	O	PU	5-W	tbd.	tbd.	5-W
CSE05	26	O	PU	5-W	tbd.	tbd.	5-W
CSE06	27	O	PU	5-W	tbd.	tbd.	5-W
CSE07	28	O	PU	5-W	tbd.	tbd.	5-W
CSE10	56	O	PU	5-W	tbd.	tbd.	5-W
CSE11	57	O	PU	5-W	tbd.	tbd.	5-W
CSE12	58	O	PU	5-W	tbd.	tbd.	5-W
CSE13	59	O	PU	5-W	tbd.	tbd.	5-W
CSE14	60	O	PU	5-W	tbd.	tbd.	5-W
CSE15	61	O	PU	5-W	tbd.	tbd.	5-W
CSE16	62	O	PU	5-W	tbd.	tbd.	5-W
CSE17	63	O	PU	5-W	tbd.	tbd.	5-W
CTXD0	75	O	PU	5-W	tbd.	tbd.	5-AF
CTXD1	35	O	PU	5-W	tbd.	tbd.	5-AF
DCK	52	I	PU	5-W	tbd.	tbd.	2-A
DDI	51	I	PU	5-W	tbd.	tbd.	2-A
DDO	54	O	PU	5-W	tbd.	tbd.	–
DMS	53	I	PU	2-A	tbd.	tbd.	2-A
$\overline{\text{DRST}}$	55	I	PD	2-I	tbd.	tbd.	2-I
FLMD0	17	I	–	2-G	tbd.	tbd.	–
FLMD1	6	I	PD	5-AP	tbd.	tbd.	5-W
INTP0	7	I	PD	5-AP	tbd.	tbd.	5-AF
INTP1	18	I	PD	5-AP	tbd.	tbd.	5-W
INTP2	31	I	PU	5-W	tbd.	tbd.	5-W
INTP3	32	I	PU	5-W	tbd.	tbd.	5-W
INTP4	33	I	PU	5-W	tbd.	tbd.	5-AF
INTP5	34	I	PU	5-W	tbd.	tbd.	5-AF
NMI	22	I	–	2-B	tbd.	tbd.	2-G
PCL	18	O	PD	5-AP	tbd.	tbd.	5-W
$\overline{\text{PUDSEL}}$[a]	22	I	–	2-B	tbd.	tbd.	2-G

Table 2-16 Alphabetical pin function list (3/4)

Pin name[a]	# Pin	I/O	3V devices[b]		5V devices[c]		
			PU/PD	Buffer Type	PU	PD	Buffer Type
$\overline{\text{RESET}}$	18	I	–	2-G	tbd.	tbd.	2-G
$\overline{\text{RESOUT}}$	8	O	–	5-AP	tbd.	tbd.	5-AF
RXDD0	32	I	PU	5-W	tbd.	tbd.	5-W
RXDD1	34	I	PU	5-W	tbd.	tbd.	5-AF
SIE0	43	I	PU	5-W	tbd.	tbd.	5-W
SIE1	50	I	PU	5-W	tbd.	tbd.	5-AF
SIF0	67	I	PD	5-AP	tbd.	tbd.	5-W
SIF1	74	I	PD	5-AP	tbd.	tbd.	5-AP
SOE0	45	O	PU	5-W	tbd.	tbd.	5-W
SOE1	48	O	PU	5-W	tbd.	tbd.	5-W
SOF0	69	O	PD	5-AP	tbd.	tbd.	5-W
SOF1	73	O	PD	5-AP	tbd.	tbd.	5-AP
TIAA00	77	I	PD	5-AP	tbd.	tbd.	5-AF
TIAA01	78	I	PD	5-AP	tbd.	tbd.	5-AF
TIAA10	79	I	PD	5-AP	tbd.	tbd.	5-AF
TIAA11	80	I	PD	5-AP	tbd.	tbd.	5-AF
TIAA20	82	I	PD	5-AP	tbd.	tbd.	5-AF
TIAA21	81	I	PD	5-AP	tbd.	tbd.	5-AF
TIAA30	83	I	PD	5-AP	tbd.	tbd.	5-AF
TIAA31	84	I	PD	5-AP	tbd.	tbd.	5-AF
TIAB00	42	I	PD	5-AP	tbd.	tbd.	5-W
TIAB01	19	I	PD	5-AP	tbd.	tbd.	5-W
TIAB02	20	I	PD	5-AP	tbd.	tbd.	5-W
TIAB03	21	I	PD	5-AP	tbd.	tbd.	5-W
TIAB10	41	I	PD	5-AP	tbd.	tbd.	5-W
TIAB11	23	I	PD	5-AP	tbd.	tbd.	5-W
TIAB12	24	I	PD	5-AP	tbd.	tbd.	5-W
TIAB13	5	I	PD	5-AP	tbd.	tbd.	5-AF
TOAA00	77	O	PD	5-AP	tbd.	tbd.	5-AF
TOAA01	78	O	PD	5-AP	tbd.	tbd.	5-AF
TOAA10	79	O	PD	5-AP	tbd.	tbd.	5-AF
TOAA11	80	O	PD	5-AP	tbd.	tbd.	5-AF
TOAA20	82	O	PD	5-AP	tbd.	tbd.	5-AF
TOAA21	81	O	PD	5-AP	tbd.	tbd.	5-AF
TOAA30	83	O	PD	5-AP	tbd.	tbd.	5-AF
TOAA31	84	O	PD	5-AP	tbd.	tbd.	5-AF
TOAB00	42	O	PD	5-AP	tbd.	tbd.	5-W
TOAB01	19	O	PD	5-AP	tbd.	tbd.	5-W
TOAB02	20	O	PD	5-AP	tbd.	tbd.	5-W
TOAB03	21	O	PD	5-AP	tbd.	tbd.	5-W
TOAB10	41	O	PD	5-AP	tbd.	tbd.	5-W

a) Availability of some of the pins is linked to the availability of the peripheral macros.
b) Devices: µPD70F3464, µPD70F3465 and µPD70F3466
c) Devices: µPD70F3470, µPD70F3471 and µPD70F3472

Table 2-16 Alphabetical pin function list (4/4)

Pin name[a]	# Pin	I/O	3V devices[b]		5V devices[c]		
			PU/PD	Buffer Type	PU	PD	Buffer Type
TOAB11	23	O	PD	5-AP	tbd.	tbd.	5-W
TOAB12	24	O	PD	5-AP	tbd.	tbd.	5-W
TOAB13	5	O	PD	5-AP	tbd.	tbd.	5-AF
TXDD0	31	O	PU	5-W	tbd.	tbd.	5-W
TXDD1	33	O	PU	5-W	tbd.	tbd.	5-AF
X1	14	I	–	16	tbd.	tbd.	16
X2	15	I	–	16	tbd.	tbd.	16

a) Availability of some of the pins is linked to the availability of the peripheral macros.
b) Devices: µPD70F3464, µPD70F3465 and µPD70F3466
c) Devices: µPD70F3470, µPD70F3471 and µPD70F3472

2.4.3 Port Buffers Diagrams

This chapter presents the block diagrams of all buffer types.

The tables in *Section 2.4, Port Group Configuration* on page 17 informs also about the buffer type, used for each port.

(1) Buffer type 2-A

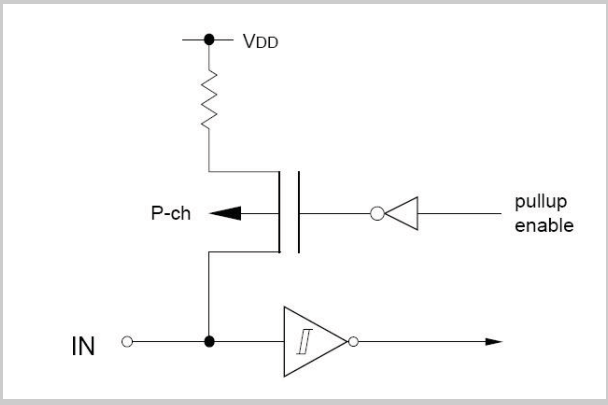


Figure 2-2 Block diagram: buffer type 2-A

(2) Buffer type 2-B

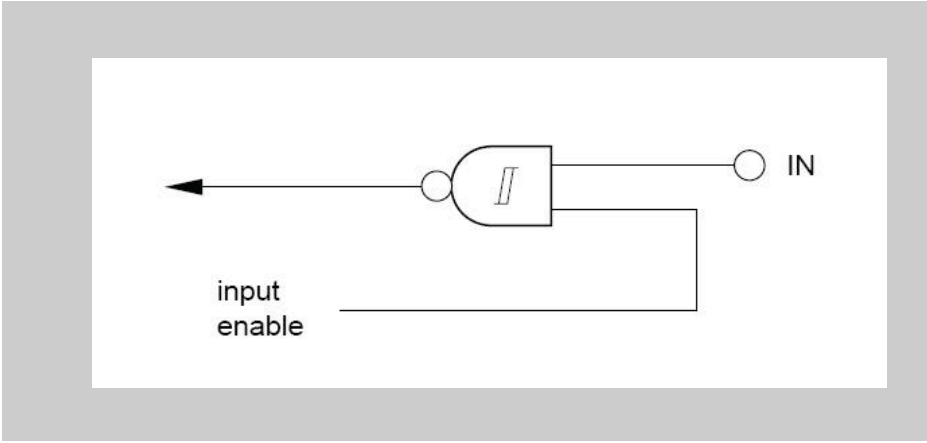


Figure 2-3 Block diagram: buffer type 2-B

(3) Buffer type 2-G

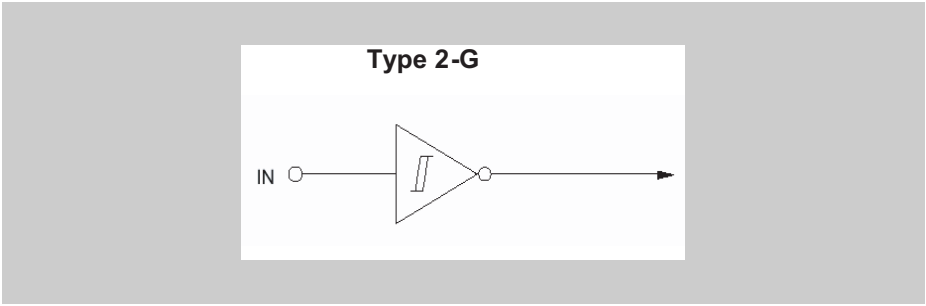


Figure 2-4 Block diagram: buffer type 2-G

(4) Buffer type 2-I

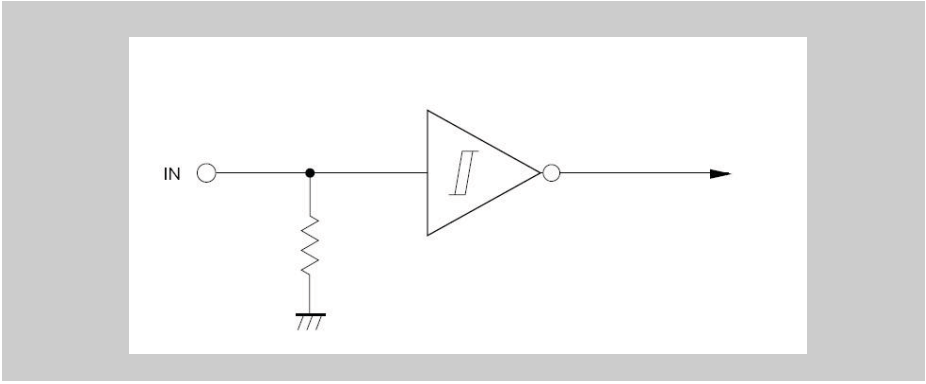


Figure 2-5 Block diagram: buffer type 2-I

(5) Buffer type 5-AF

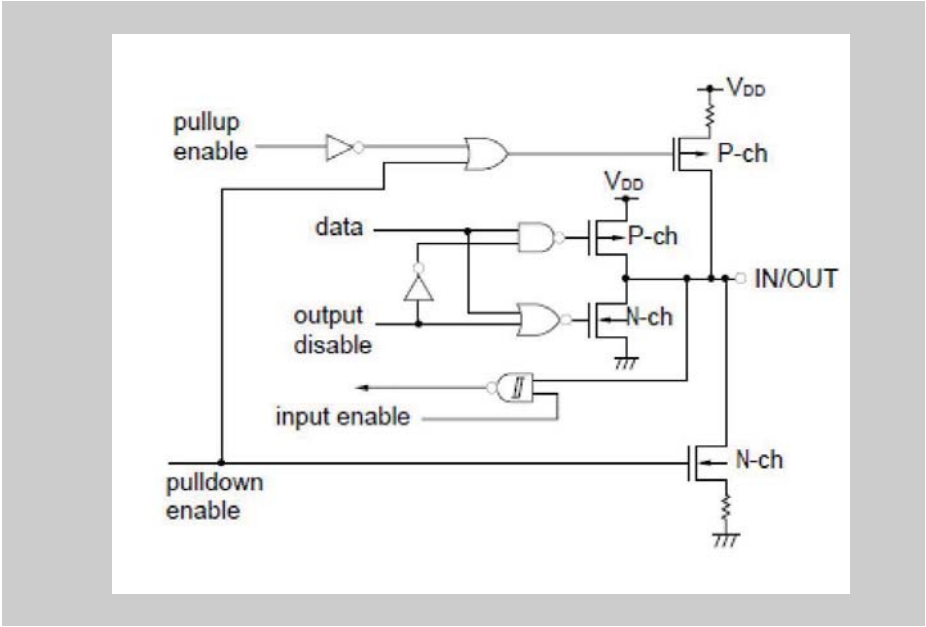


Figure 2-6 Block diagram: buffer type 5-AF

(6) Buffer type 5-K

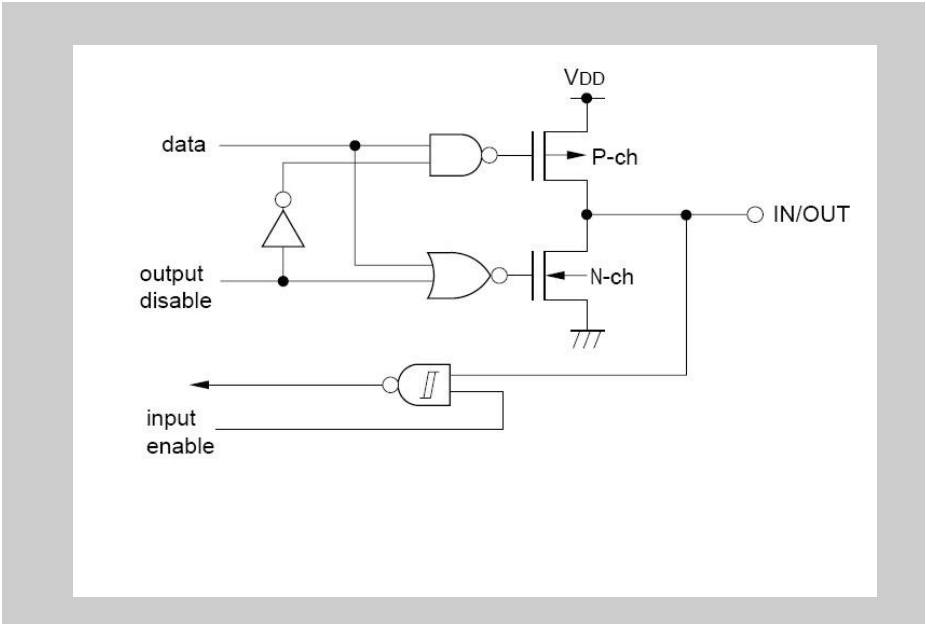


Figure 2-7 Block diagram: buffer type 5-K

(7) Buffer type 5-AP

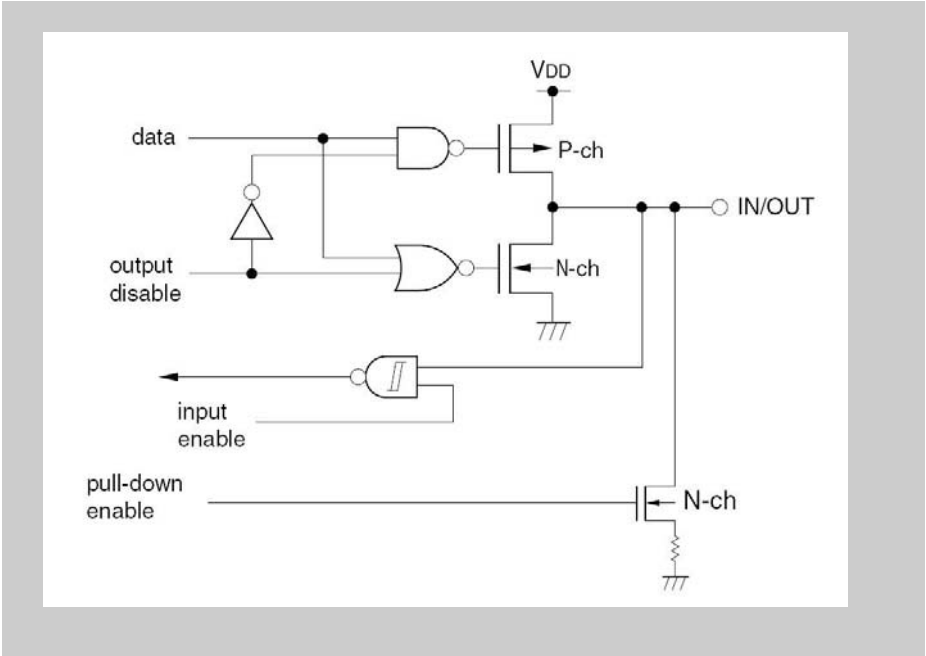


Figure 2-8 Block diagram: buffer type 5-AP

(8) Buffer type 5-W

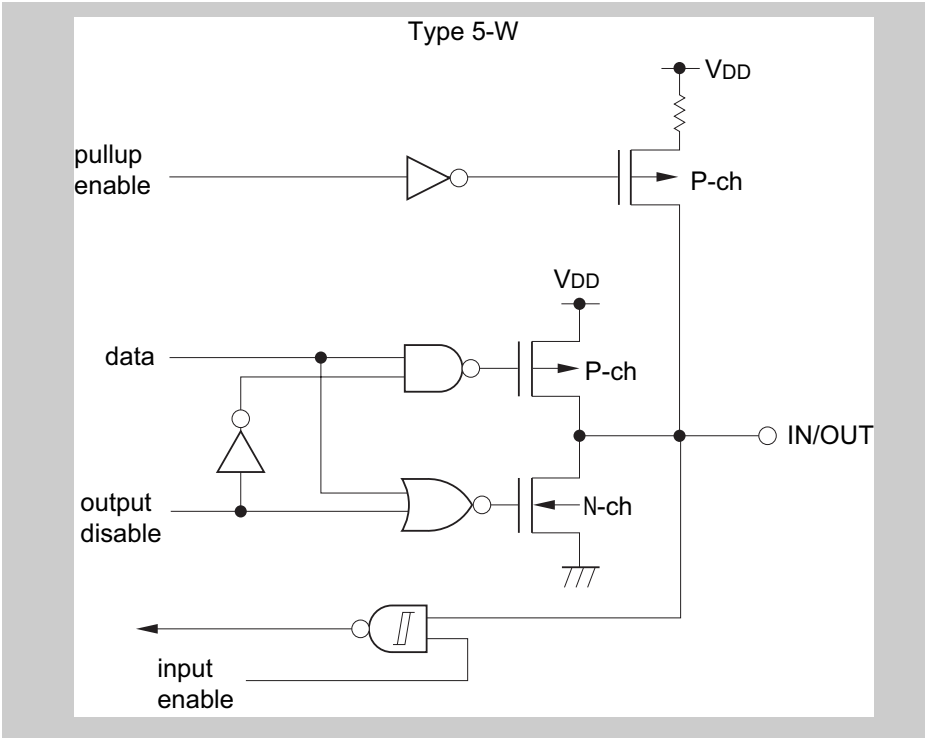


Figure 2-9 Block diagram: buffer type 5-W

(9) Buffer type 7-B

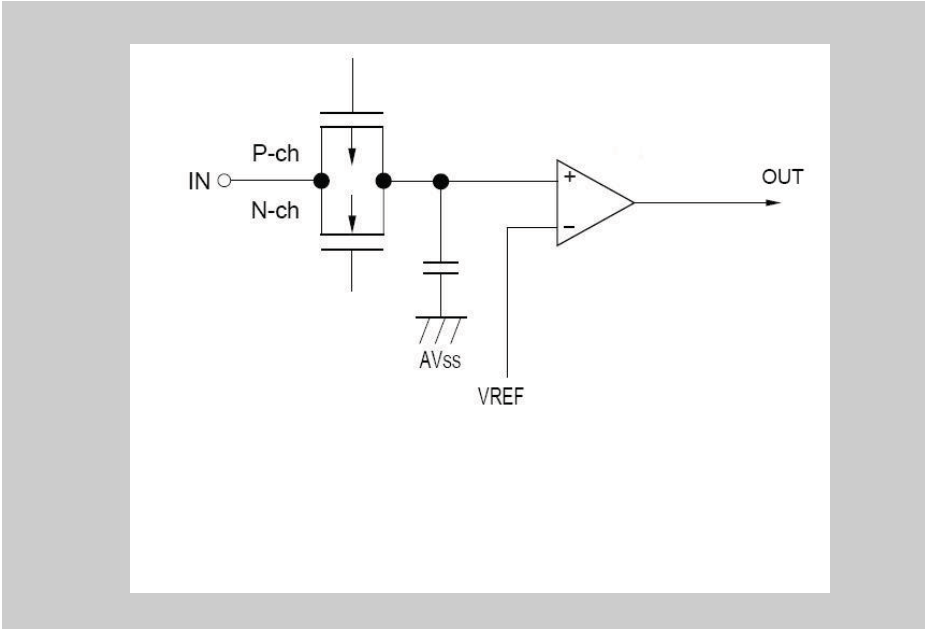


Figure 2-10 Block diagram: buffer type 7-B

(10) Buffer type 16

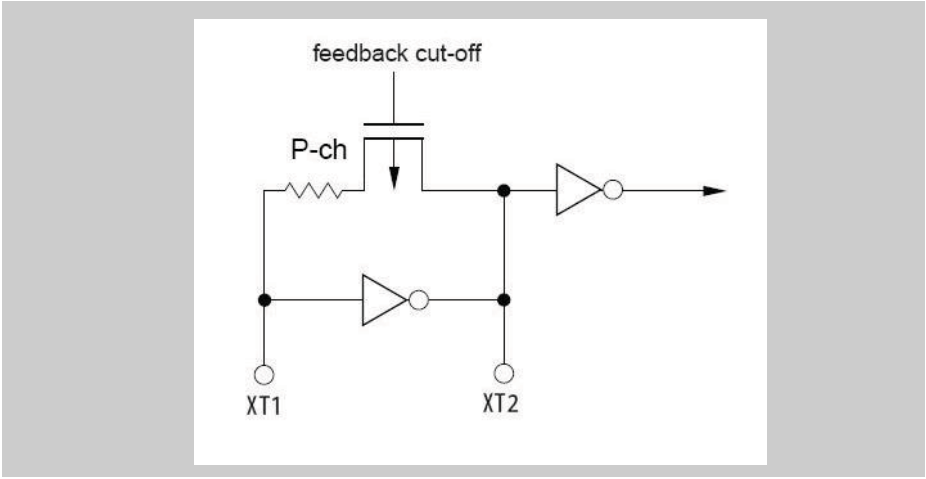


Figure 2-11 Block diagram: buffer type 16

2.4.4 Port group 0

Port group 0 is an 8-bit port group. In alternative mode, it comprises pins for the following functions:

- Timer TAB0 channels
  (TOAB01 to TOAB03, TIAB01 to TIAB03)
- Timer TAB1 channels
  (TOAB13 AND TIAB13)

Port group 0 includes the following pins.

Table 2-17 Port group 0: pin functions and port types

Pin functions in different modes			Pin function after reset	Pull-up / Pull down	
Port mode (PMCnm = 0)	Alternative mode (PMCnm = 1)				
	PFCnm = 0	PFCnm = 1		3V devices[a]	5V devices[b]
P00 (I/O)	TIAB13	TOAB13	P00 (I)	PD	PU/PD
P01 (I/O)	FLMD1	prohibited	P01 (I)	PD	PU
P02 (I/O)	INTP0	prohibited	P02 (I)	PD	PU/PD
P03 (I/O)	$\overline{\text{RESOUT}}$	prohibited	P03 (I)	-	PU/PD
P04 (I/O)	INTP1	PCL	P04 (I)	PD	PU/PD
P05 (I/O)	TIAB01	TOAB01	P05 (I)	PD	PU/PD
P06 (I/O)	TIAB02 INTP6	TOAB02	P06 (I)	PD	PU/PD
P07 (I/O)	TIAB03 INTP7	TOAB03	P07 (I)	PD	PU/PD

a) Only for devices: µPD70F3464, µPD70F3465 and µPD70F3466
b) Only for devices: µPD70F3470, µPD70F3471 and µPD70F3472

Table 2-18 Port group 0: configuration registers

Register	Address	Initial value	Used bits							
PMC0	FFFF F440$_H$	00$_H$	PMC07	PMC06	PMC05	PMC04	0	PMC02	PMC01	PMC00
PM0	FFFF F420$_H$	FF$_H$	PM07	PM06	PM05	PM04	PM03	PM02	PM01	PM00
PFC0	FFFF F460$_H$	00$_H$	PFC07	PFC06	PFC05	PFC04	0	0	0	PFC00
P0	FFFF F400$_H$	00$_H$	P07	P06	P05	P04	P03	P02	P01	P00
PE0[a]	FFFF FC40$_H$	xx$_H$[b]	PD07	PD06	PD05	PD04	0	PD02	PD01	PD00
PU0[c]	FFFF FC40$_H$	00$_H$	PU07	PU06	PU05	PU04	PU03	PU02	PU01	PU00
PD0[c]	FFFF FC60$_H$	00$_H$	PD07	PD06	PD05	PD04	PD03	PD02	0	PD00

a) Only for devices: µPD70F3464, µPD70F3465 and µPD70F3466
b) Reset value depends on value of $\overline{\text{PUDSEL}}$. See *Section 2.3.4, Configuration of pull-up and pull-down resistors (3V devices)*.
c) Only for devices: µPD70F3470, µPD70F3471 and µPD70F3472

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

2.4.5 Port group 1

Port group 1 is an 8-bit port group. In alternative mode, it comprises pins for the following functions:

- CSIE0 slave select output (SCSE04 to SCSE07)
- UARTD0 receive/transmit data (RXDD0, TXDD0)
- Timer TAB1 channels (TOAB11, TOAB11, TIAB12, TIAB12)

Port group 1 includes the following pins.

Table 2-19 Port group 1: pin functions and port types

Pin functions in different modes			Pin function after reset	Pull-up / Pull down	
Port mode (PMCnm = 0)	Alternative mode (PMCnm = 1)				
	PFCnm = 0	PFCnm = 1		3V devices[a]	5V devices[b]
P10 (I/O)	TIAB11	TOAB11	P10 (I)	PD	PU/PD
P11 (I/O)	TIAB12	TOAB12	P11 (I)	PD	PU/PD
P12 (I/O)	CS04	prohibited	P12 (I)	PU	PU/PD
P13 (I/O)	CS05	prohibited	P13 (I)	PU	PU/PD
P14 (I/O)	CS06	prohibited	P14 (I)	PU	PU/PD
P15 (I/O)	CS07	prohibited	P15 (I)	PU	PU/PD
P16 (I/O)	INTP2	TXDD0	P16 (I)	PU	PU/PD
P17 (I/O)	RXDD0 INTP3	prohibited	P17 (I)	PU	PU/PD

a) Only for devices: µPD70F3464, µPD70F3465 and µPD70F3466
b) Only for devices: µPD70F3470, µPD70F3471 and µPD70F3472

Table 2-20 Port group 1: configuration registers

Register	Address	Initial value	Used bits							
PMC1	FFFF F402 H	00_H	PMC17	PMC16	PMC15	PMC14	PMC13	PMC12	PMC11	PMC10
PM1	FFFF F422 H	FF_H	PM17	PM16	PM15	PM14	PM13	PM12	PM11	PM10
PFC1	FFFF F462 H	00_H	0	PFC16	0	0	0	0	PFC11	PFC10
P1	FFFF F402 H	00_H	P17	P16	P15	P14	P13	P12	P11	P10
PE1[a]	FFFF FC42 H	xx_H[b]	PU17	PU16	PU15	PU14	PU13	PU12	PD11	PD10
PU1[c]	FFFF FC42 H	00_H	PU17	PU16	PU15	PU14	PU13	PU12	PU11	PU10
PD1[c]	FFFF FC62 H	00_H	PD17	PD16	PD15	PD14	PD13	PD12	PD11	PD10

a) Only for devices: µPD70F3464, µPD70F3465 and µPD70F3466
b) Reset value depends on value of PUDSEL. See *Section 2.3.4, Configuration of pull-up and pull-down resistors (3V devices)*.
c) Only for devices: µPD70F3470, µPD70F3471 and µPD70F3472

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

2.4.6 Port group 2

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

- CSIE0 and CSIE1 clock and data outputs and data inputs (SCKE0, SOE0, SIE0, SCKE1, SOE1, SIE1)
- Timer TAB0 and TAB1 channels (TOAB10, TOAB10, TIAB00, TIAB00)

Port group 2 includes the following pins.

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

Pin functions in different modes			Pin function after reset	Pull-up / Pull down	
Port mode (PMCnm = 0)	Alternative mode (PMCnm = 1)				
	PFCnm = 0	PFCnm = 1		3V devices[a]	5V devices[b]
P20 (I/O)	TIAB10	TOAB10	P00 (I)	PD	PU/PD
P21 (I/O)	TIAB01	TOAB01	P01 (I)	PD	PU/PD
P22 (I/O)	SIE0	prohibited	P02 (I)	PU	PU/PD
P23 (I/O)	SCKE0	prohibited	P03 (I)	PU	PU/PD
P24 (I/O)	SOE0	prohibited	P04 (I)	PU	PU/PD
P25 (I/O)	SOE1	prohibited	P05 (I)	PU	PU/PD
P26 (I/O)	SCKE1	prohibited	P06 (I)	PU	PU/PD
P27 (I/O)	SIE1	prohibited	P07 (I)	PU	PU/PD

a) Only for devices: µPD70F3464, µPD70F3465 and µPD70F3466
b) Only for devices: µPD70F3470, µPD70F3471 and µPD70F3472

Table 2-22 Port group 2: configuration registers

Register	Address	Initial value	Used bits							
PMC2	FFFF F444$_H$	00$_H$	PMC27	PMC26	PMC25	PMC24	PMC23	PMC22	PMC21	PMC20
PM2	FFFF F424$_H$	FF$_H$	PM27	PM26	PM25	PM24	PM23	PM22	PM21	PM20
PFC2	FFFF F464$_H$	00$_H$	0	0	0	0	0	0	PFC21	PFC20
P2	FFFF F404$_H$	00$_H$	P27	P26	P25	P24	P23	P22	P21	P20
PE2[a]	FFFF FC44$_H$	xx$_H$[b]	PU27	PU26	PU25	PU24	PU23	PU22	PD21	PD20
PU2[c]	FFFF FC44$_H$	00$_H$	PU27	PU26	PU25	PU24	PU23	PU22	PU21	PU20
PD2[c]	FFFF FC64$_H$	00$_H$	PD27	PD26	PD25	PD24	PD23	PD22	PD21	PD20

a) Only for devices: µPD70F3464, µPD70F3465 and µPD70F3466
b) Reset value depends on value of PUDSEL. See *Section 2.3.4, Configuration of pull-up and pull-down resistors (3V devices)*.
c) Only for devices: µPD70F3470, µPD70F3471 and µPD70F3472

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

2.4.7 Port group 3

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

- CSIE1 slave select output (SCSE10 to SCSE17)

Port group 3 includes the following pins.

Table 2-23 Port group 3: pin functions and port types

Pin functions in different modes		Pin function after reset	Pull-up / Pull down	
Port mode (PMCnm = 0)	Alternative mode (PMCnm = 1)		3V devices[a]	5V devices[b]
P30 (I/O)	CSE10 (O)	P30 (I)	PU	PU/PD
P31 (I/O)	CSE11 (O)	P31 (I)	PU	PU/PD
P32 (I/O)	CSE12 (O)	P32 (I)	PU	PU/PD
P33 (I/O)	CSE13 (O)	P33 (I)	PU	PU/PD
P34 (I/O)	CSE14 (O)	P34 (I)	PU	PU/PD
P35 (I/O)	CSE15 (O)	P35 (I)	PU	PU/PD
P36 (I/O)	CSE16 (O)	P36 (I)	PU	PU/PD
P37 (I/O)	CSE17 (O)	P37 (I)	PU	PU/PD

a) Only for devices: µPD70F3464, µPD70F3465 and µPD70F3466
b) Only for devices: µPD70F3470, µPD70F3471 and µPD70F3472

Table 2-24 Port group 3: configuration registers

Register	Address	Initial value	Used bits							
PMC3	FFFF F446$_H$	00$_H$	PMC37	PMC36	PMC35	PMC34	PMC33	PMC32	PMC31	PMC30
PM3	FFFF F426$_H$	FF$_H$	PM37	PM36	PM35	PM34	PM33	PM32	PM31	PM30
P3	FFFF F406$_H$	00$_H$	P37	P36	P35	P34	P33	P32	P31	P30
PE3[a]	FFFF FC46$_H$	xx$_H$[b]	PU37	PU36	PU35	PU34	PU33	PU32	PU31	PU30
PU3[c]	FFFF FC46$_H$	00$_H$	PU37	PU36	PU35	PU34	PU33	PU32	PU31	PU30
PD3[c]	FFFF FC66$_H$	00$_H$	PD37	PD36	PD35	PD34	PD33	PD32	PD31	PD30

a) Only for devices: µPD70F3464, µPD70F3465 and µPD70F3466
b) Reset value depends on value of $\overline{\text{PUDSEL}}$. See *Section 2.3.4, Configuration of pull-up and pull-down resistors (3V devices)*.
c) Only for devices: µPD70F3470, µPD70F3471 and µPD70F3472

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

2.4.8 Port group 4

Port group 4 is an 7-bit port group. In alternative mode, it comprises pins for the following functions:

- CSIF0 and CSIF1 clock inputs/outputs and data inputs/outputs (SCKB0, SIB0, SOF0, SCKB1, SIB1, SOF1)
- CAN0 input output channels (CTXD0, CRXD0)

Port group 4 includes the following pins.

Table 2-25 Port group 4: pin functions and port types

Pin functions in different modes		Pin function after reset	Pull-up / Pull down	
Port mode (PMCnm = 0)	Alternative mode (PMCnm = 1)		3V devices[a]	5V devices[b]
P40 (I/O)	SIF0 (I/O)	P40 (I)	PD	PU/PD
P41 (I/O)	SCKF0 (I)	P41 (I)	PD	PU/PD
P42 (I/O)	SOF0 (O)	P42 (I)	PD	PU/PD
P43 (I/O)	SCKF1 (I/O)	P43 (I)	PD	PU/PD
P44 (I/O)	SOF1 (I)	P44 (I)	PD	PU/PD
P45 (I/O)	SIF1 (I)	P45 (I)	PD	PU/PD
P46 (I/O)	CTXD0 (O)	P46 (I)	PU	PU/PD
P47 (I/O)	CRXD0 (I)	P47 (I)	PU	PU/PD

a) Only for devices: µPD70F3464, µPD70F3465 and µPD70F3466
b) Only for devices: µPD70F3470, µPD70F3471 and µPD70F3472

Table 2-26 Port group 4: configuration registers

Register	Address	Initial value	Used bits							
PMC4	FFFF F448$_H$	00$_H$	PMC47	PMC46	PMC45	PMC44	PMC43	PMC42	PMC41	PMC40
PM4	FFFF F428$_H$	FF$_H$	PM47	PM46	PM45	PM44	PM43	PM42	PM41	PM40
P4	FFFF F408$_H$	00$_H$	P47	P46	P45	P44	P43	P42	P41	P40
PE4[a]	FFFF FC48$_H$	xx$_H$[b]	PU47	PU46	PD45	PD44	PD43	PD42	PD41	PD40
PU4[c]	FFFF FC48$_H$	00$_H$	PU47	PU46	PU45	PU44	PU43	PU42	PU41	PU40
PD4[c]	FFFF FC68$_H$	00$_H$	PD47	PD46	PD45	PD44	PD43	PD42	PD41	PD40

a) Only for devices: µPD70F3464, µPD70F3465 and µPD70F3466
b) Reset value depends on value of PUDSEL. See *Section 2.3.4, Configuration of pull-up and pull-down resistors (3V devices)*.
c) Only for devices: µPD70F3470, µPD70F3471 and µPD70F3472

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

2.4.9 Port group 5

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

- Timer AA 0 channels
  (TIAA00, TIAA01, TOAA00, TOAA01)
- Timer AA 1 channels
  (TIAA10, TIAA11, TOAA10, TOAA11)
- Timer AA 2channels
  (TIAA20, TIAA21, TOAA20, TOAA21)
- Timer AA 3 channels
  (TIAA30, TIAA31, TOAA30, TOAA31)

Port group 5 includes the following pins.

Table 2-27 Port group 5: pin functions and port types

Pin functions in different modes			Pin function after reset	Pull-up / Pull down	
Port mode (PMCnm = 0)	Alternative mode (PMCnm = 1)				
	PFCnm = 0	PFCnm = 1		3V devices[a]	5V devices[b]
P50 (I/O)	TIAA00 (I)	TOAA00 (O)	P50 (I)	PD	PU/PD
P51 (I/O)	TIAA01 (I)	TOAA01 (O)	P51 (I)	PD	PU/PD
P52 (I/O)	TIAA10 (I)	TOAA10 (O)	P52 (I)	PD	PU/PD
P53 (I/O)	TIAA11 (I)	TOAA11 (O)	P53 (I)	PD	PU/PD
P54 (I/O)	TIAA20 (I)	TOAA20 (O)	P54 (I)	PD	PU/PD
P55 (I/O)	TIAA21 (I)	TOAA21 (O)	P55 (I)	PD	PU/PD
P56 (I/O)	TIAA30 (I)	TOAA30 (O)	P56 (I)	PD	PU/PD
P57 (I/O)	TIAA31 (I)	TOAA31 (O)	P57 (I)	PD	PU/PD

a) Only for devices: µPD70F3464, µPD70F3465 and µPD70F3466
b) Only for devices: µPD70F3470, µPD70F3471 and µPD70F3472

Table 2-28 Port group 5: configuration registers

Register	Address	Initial value	Used bits							
PMC5	FFFF F44A$_H$	00$_H$	PMC57	PMC56	PMC55	PMC54	PMC53	PMC52	PMC51	PMC50
PM5	FFFF F42A$_H$	FF$_H$	PM57	PM56	PM55	PM54	PM53	PM52	PM51	PM50
PFC5	FFFF F46A$_H$	00$_H$	PFC57	PFC56	PFC55	PFC54	PFC53	PFC52	PFC51	PFC50
P5	FFFF F40A$_H$	00$_H$	P57	P56	P55	P54	P53	P52	P51	P50
PE5[a]	FFFF FC4A$_H$	xx$_H$[b]	PD57	PD56	PD55	PD54	PD53	PD52	PD51	PD50
PU5[c]	FFFF FC4A$_H$	00$_H$	PU57	PU56	PU55	PU54	PU53	PU52	PU51	PU50
PD5[c]	FFFF FC6A$_H$	00$_H$	PD57	PD56	PD55	PD54	PD53	PD52	PD51	PD50

a) Only for devices: µPD70F3464, µPD70F3465 and µPD70F3466
b) Reset value depends on value of PUDSEL. See *Section 2.3.4, Configuration of pull-up and pull-down resistors (3V devices)*.
c) Only for devices: µPD70F3470, µPD70F3471 and µPD70F3472

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

2.4.10 Port group 6

Port group 6 is a 4-bit port group. In alternative mode, it comprises the pin for External Non-maskable interrupt (NMI).

For the devices µPD70F3464, µPD70F3465 and µPD70F3466, P60 is also the pin for PUDSEL. This is **not an** alternative function, as it is only selected in RESET mode. See *Section 2.3.4, Configuration of pull-up and pull-down resistors (3V devices)* for further information on PUDSEL.

Port group 6 includes the following pins.

Table 2-29 Port group 6: pin functions and port types

Pin functions in different modes		Pin function after reset
Port mode (PMCnm = 0)	Alternative mode (PMCnm = 1)	
P60 (I)	NMI (I)	P60 (I)

Table 2-30 Port group CS: configuration registers

Register	Address	Initial value	Used bits							
PMC6	FFFF F44C$_H$	00$_H$	0	0	0	0	0	0	0	PMC60
P6	FFFF F40C$_H$	00$_H$	0	0	0	0	0	0	0	P60

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 a 5-bit port group. In alternative mode, it comprises pins for the following functions:

- UARTD1 channel (RXDD1, TXDD1)
- CAN1 channel (CTXD1, CRXD1)
- CSIE0 slave select output (SCSE00 to SCSE03)

Port group 8 includes the following pins.

Table 2-31 Port group 8: pin functions and port types

Pin functions in different modes			Pin function after reset	Pull-up / Pull down	
Port mode (PMCnm = 0)	Alternative mode (PMCnm = 1)				
	Function 1 PFCnm = 0	Function 3 PFCnm = 1		3V devices[a]	5V devices[b]
P80 (I)	TXDD1 (O)	INTP4 (I)	P80 (I)	PU	PU/PD
P81 (I/O)	RXDD1 (I) INTP5	prohibited	P81 (I)	PU	PU/PD
P82 (I/O)	CTXD1 (O)	prohibited	P82 (I)	PU	PU/PD
P83 (I/O)	CRXD1 (I)	prohibited	P83 (I)	PU	PU/PD
P84 (I/O)	CSE03 (O)	prohibited	P84 (I)	PU	PU/PD
P85 (I/O)	CSE02 (O)	prohibited	P84 (I)	PU	PU/PD
P86 (I/O)	CSE01 (O)	prohibited	P84 (I)	PU	PU/PD
P87 (I/O)	CSE00 (O)	prohibited	P84 (I)	PU	PU/PD

a) Only for devices: µPD70F3464, µPD70F3465 and µPD70F3466
b) Only for devices: µPD70F3470, µPD70F3471 and µPD70F3472

Table 2-32 Port group 8: configuration registers (1/2)

Register	Address	Initial value	Used bits							
PMC8	FFFF F450$_H$	00$_H$	PMC87	PMC86	PMC85	PMC84	PMC83	PMC82	PMC81	PMC80
PM8	FFFF F430$_H$	FF$_H$	PM87	PM86	PM85	PM84	PM83	PM82	PM81	PM80
PFC8	FFFF F470$_H$	00$_H$	0	0	0	0	0	0	0	PFC80
P8	FFFF F410$_H$	00$_H$	P87	P86	P85	P84	P83	P82	P81	P80

Table 2-32 Port group 8: configuration registers (2/2)

Register	Address	Initial value	Used bits							
PE8[a]	FFFF FC50$_H$	xx$_H$[b]	PD87	PD86	PD85	PD84	PD83	PD82	PD81	PD80
PU8[c]	FFFF FC50$_H$	00$_H$	PU87	PU86	PU85	PU84	PU83	PU82	PU81	PU80
PD8[c]	FFFF FC70$_H$	00$_H$	PD87	PD86	PD85	PD84	PD83	PD82	PD81	PD80

a) Only for devices: µPD70F3464, µPD70F3465 and µPD70F3466
b) Reset value depends on value of $\overline{\text{PUDSEL}}$. See *Section 2.3.4, Configuration of pull-up and pull-down resistors (3V devices)*.
c) Only for devices: µPD70F3470, µPD70F3471 and µPD70F3472

2.5 Pin Functions during and after Reset

See section 20.2“External Reset (RESET)” on page 595 for information on the status of pins during and after reset.

2.6 Recommended Connection of Unused Pins

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

Table 2-33 Recommended connection of unused pins

Pin	Recommended connection	
	3V devices[a]	5V devices[b]
Port pins		
pins of port group xx	tbd.	tbd.
Non-port pins		
ANI00 to ANI09	connect to AVDD or AVSS0	connect to AVDD or AVSS0
DCK, DDI, DMS	tbd.	connect to VDD3x via pull-up resistor
DDO	leave open	leave open
$\overline{\text{DRST}}$	tbd.	connect to VSS3x via pull-down resistor

a) Devices µPD70F3464, µPD70F3465 and µPD70F3466
b) Devices µPD70F3470, µPD70F3471 and µPD70F3472

Note 1. When connecting the unused pins with a power supply or ground, it is recommended to connect the pins through a resistance of 1 to 10 KΩ.

2. If the overall maximum output current 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 Electrical Target Specification.

2.7 Package Pins Assignment

The following figure shows the location of pins in top view. Every pin is labelled with its pin name. For port pins, only the pin name in port mode is given. For a list of all alternative pin names, refer to *Table 2-16* on page 19.

Figure 2-12 Pin configuration µPD70F346x – 100-pin plastic LQFP (fine pitch) (14x14)

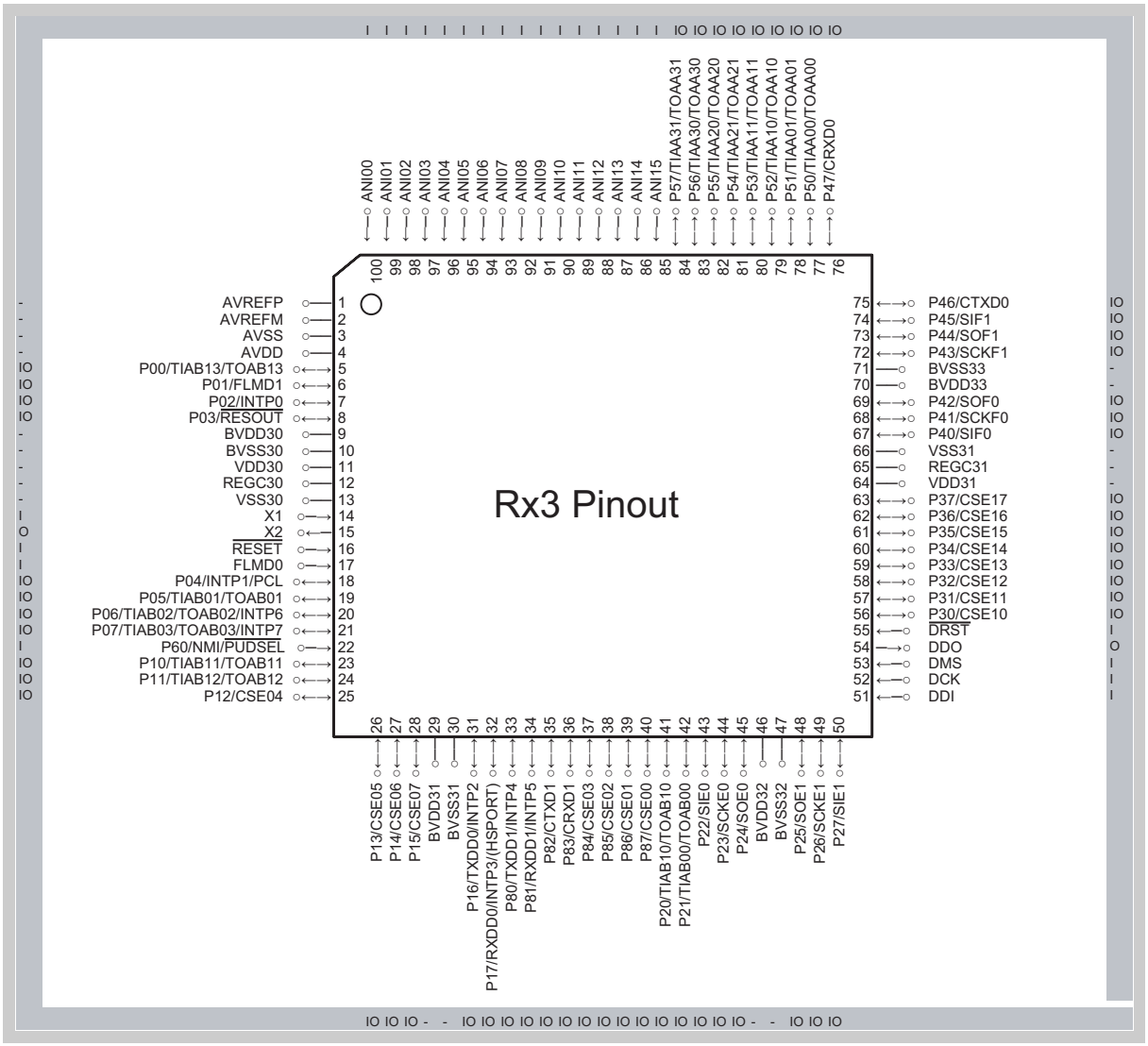
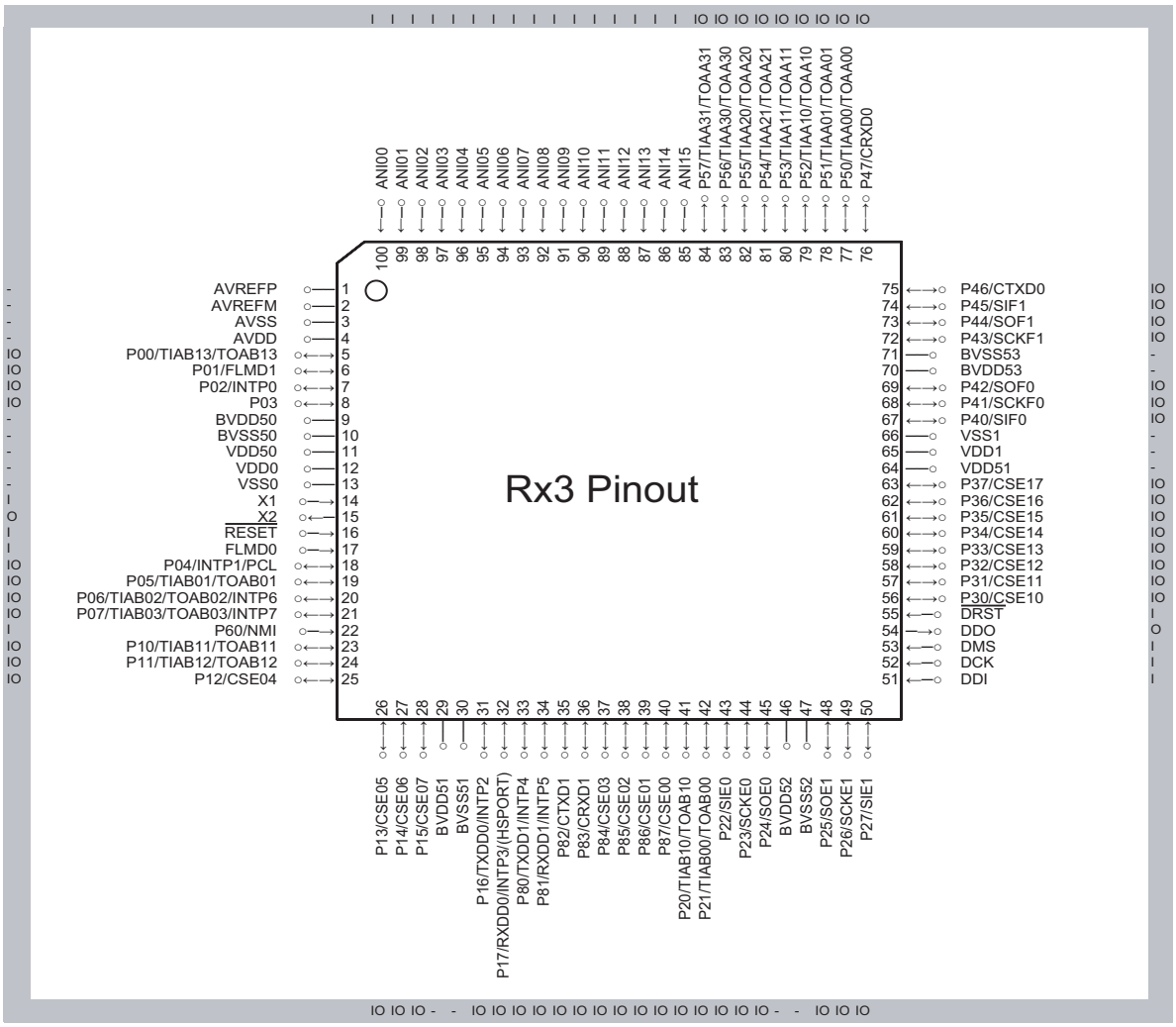


Figure 2-13 Pin Configuration µPD70F347x – 100-pin plastic LQFP (fine pitch) (14x14)



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 based 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.

The integrated V850E CPU offers simplified pipeline handling and programming, which results in compact code size comparable to 16-bit CISC CPUs. The RISC processor core of the V850E achieves marked improvements in instruction execution speed. The optimized pipeline architecture enables basic instructions to be executed in one clock period.

Because the V850 CPU uses two-byte basic instructions and instructions that are compatible with high-level languages, the object code efficiency in a C compiler is increased, and program size can be reduced. Furthermore, the on-chip interrupt controller provides high-speed interrupt response and processing. Thus, these devices are well suited for high-level real-time control applications.

Features summary

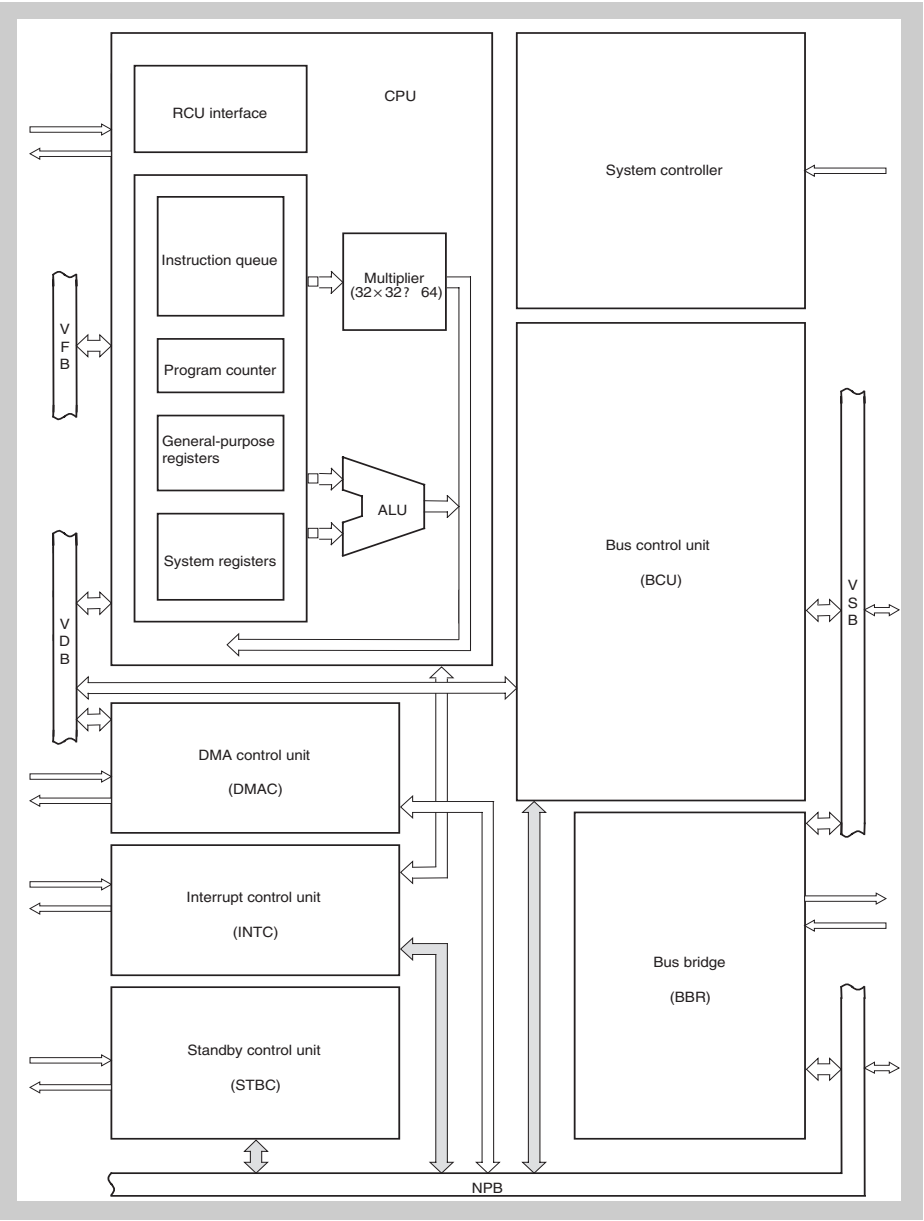
The CPU has the following special features:

- 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 buses 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 – System bus	Bus interface to the Memory Controller for access to the NPB bus bridge BBR.
VFB – Fetch bus	Interface to the internal ROM (mask ROM or flash ROM).
VDB – 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 table below. For details, refer to V850E1 User's Manual Architecture (U14559EJ3V1UM00).

Table 3-2 CPU register set

General Purpose Registers		System Registers
31	0	31 0
r0	zero register	EIPC status saving register during interrupt
r1	reserved for assembler	EIPSW status saving register during interrupt
r2		
r3	stack pointer (SP)	FEPC status saving register during NMI
r4	global pointer (GP)	FEPSW status saving register during NMI
r5	text pointer (TP)	
r6		ECR interrupt/execution source register
r7		
r8		PSW program status word
r9		
r10		CTPC status saving register during CALLT execution
r11		CTPSW status saving register during CALLT execution
r12		
r13		DBPC status saving register during exception/debug trap
r14		DBPCSW status saving register during exception/debug trap
r15		
r16		CTBP CALLT base pointer
r17		
r18		PC program counter
r19		
r20		
r21		
r22		
r23		
r24		
r25		
r26		
r27		
r28		
r29	zero register	
r30	element pointer (EP)	
r31	link pointer (LP	

Some registers are write protected. That means, writing to those registers is protected by a special sequence of instructions. Refer to *Section 3.8, Write Protected Registers* on page 66 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 3-3*). For details refer to the documentation of your assembler/compiler.

Table 3-3 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	Used to generate stack frame when function is called.[b]
r4	Global pointer	Used to access global variable in data area.[b]
r5	Text pointer	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	Base pointer when memory is accessed by means of instructions SLD (short format load) and SST (short format store).[a]
r31	Link pointer	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. After the registers have been used, their contents must be restored.

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.

Table 3-4 System register numbers

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

a) These registers can be accessed only when the DBTRAP instruction is executed. DBTRAP exceptions are generated upon ILGOP detections (refer to *Section Chapter 5, Interrupt Controller (INTC)* on page 81).

2) Since only one set of these registers is available, the program must save the contents of these registers when multiple interrupt servicing is permitted.

3) Since only one set of these registers is available, the program must save the contents of these registers when nested CALLT instructions are used.

Caution Even if EIPC or FEPC, or bit 0 of CTPC is set to 1 by the LDSR instruction, bit 0 is ignored when execution is returned to the main routine by the RETI instruction after interrupt servicing (this is because bit 0 of the PC is fixed to 0). Set an even value to EIPC, FEPC, and CTPC (bit 0 = 0).

(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 $\overline{\text{RESET}}$.

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 3-5*.

When one of the occasions listed in *Table 3-5* 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 3-10* on page 46 and to *Section Chapter 5, Interrupt Controller (INTC)* on page 81.

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

Table 3-5 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	• exception trap • debug trap • debug break • during a single-step operation
Status saving register during CALLT execution	CTPC	• execution of CALLT instruction

Note
1. Reading from DBPC is only enabled in debug mode. Otherwise, the read value is undefined.
2. When multiple interrupts are enabled, the contents of EIPC or FEPC must be saved by the program. This is because only one PC saving register is provided for maskable interrupts and non-maskable interrupts, respectively.
3. The values of EIPC are restored to PC during execution of a RETI instruction.

Cautions
1. 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.
2. 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).
3. If bit 0 is set to 1, the setting of this bit is ignored because bit 0 of the program counter is fixed to 0.

(3) PSW - Program status word register

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 $\overline{\text{RESET}}$.

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 3-6 PSW register contents (1/2)

Bit position	Flag	Function
31 to 8	RFU	Reserved Field. Fixed to 0.
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

Table 3-6 PSW register contents (2/2)

Bit position	Flag	Function
2	OV[1]	Overflow flag. Indicates whether an overflow occurred as a result of the operation. 0: Overflow did not occur 1: Overflow occurred
1	S[1]	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 3-7 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, CTPSW saving registers

The PSW saving registers save the contents of the program status word for different occasions (see *Table 3-5*).

When one of the occasions listed in *Table 3-5* 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 3-8 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 in debug mode. Otherwise, the read value is undefined.

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 3-10* on page 46.

Initial Value $0000\ 0000_H$. This register is cleared by $\overline{\text{RESET}}$.

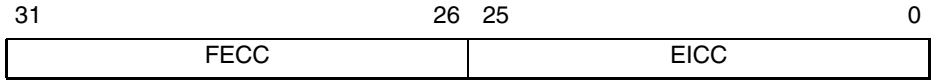


Table 3-9 ECR register contents

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 3-10 Interrupt/execution codes

Interrupt/Exception Source			Classification	Exception Code	Handler Address	Value restored to EIPC/FEPC
Name		Trigger				
Non-maskable interrupt (NMI)		refer to *Section Chapter 5, Interrupt Controller (INTC)* on page 81	Interrupt	0010_H	$0000\ 0010_H$	next PC (see Note)
Maskable interrupt			Interrupt	refer to *Section Chapter 5, Interrupt Controller (INTC)* on page 81	• 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
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, 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
- On-chip debug mode

After reset release, the microcontroller starts to fetch instructions from an internal boot ROM that contains the internal firmware. The firmware checks the pins FLMD0 and FLMD1 to set the operation mode after reset release according to *Table 3-11*.

Table 3-11 Selection of operation modes

Pins		Operation Mode
FLMD0	FLMD1	
0	X	Single chip mode
1	0	Flash programming mode
all other		Reserved

Note The FLMD1 pin function is shared with the PDL5 pin. See *Chapter 22, Section Chapter 22, On-Chip Debug Unit* for more information.

3.3.1 Normal operation mode

This is the user mode of the device. After reset release, the CPU starts op-code execution from the built-in Flash memory. The normal operation mode is always available.

3.3.2 Flash programming mode

In the Flash programming mode the device communicates with an external Flash Programmer. After reset release, the CPU starts op-code execution from the built-in BROM containing the Flash firmware.

For more information see section *Section Chapter 6, Flash Memory* on page 117.

3.3.3 On-chip debug mode

With on-chip debug mode, the microcomputer that is mounted on the target hardware controls the user program. Refer to *Chapter 22, Section Chapter 22, On-Chip Debug Unit* for details of operation.

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

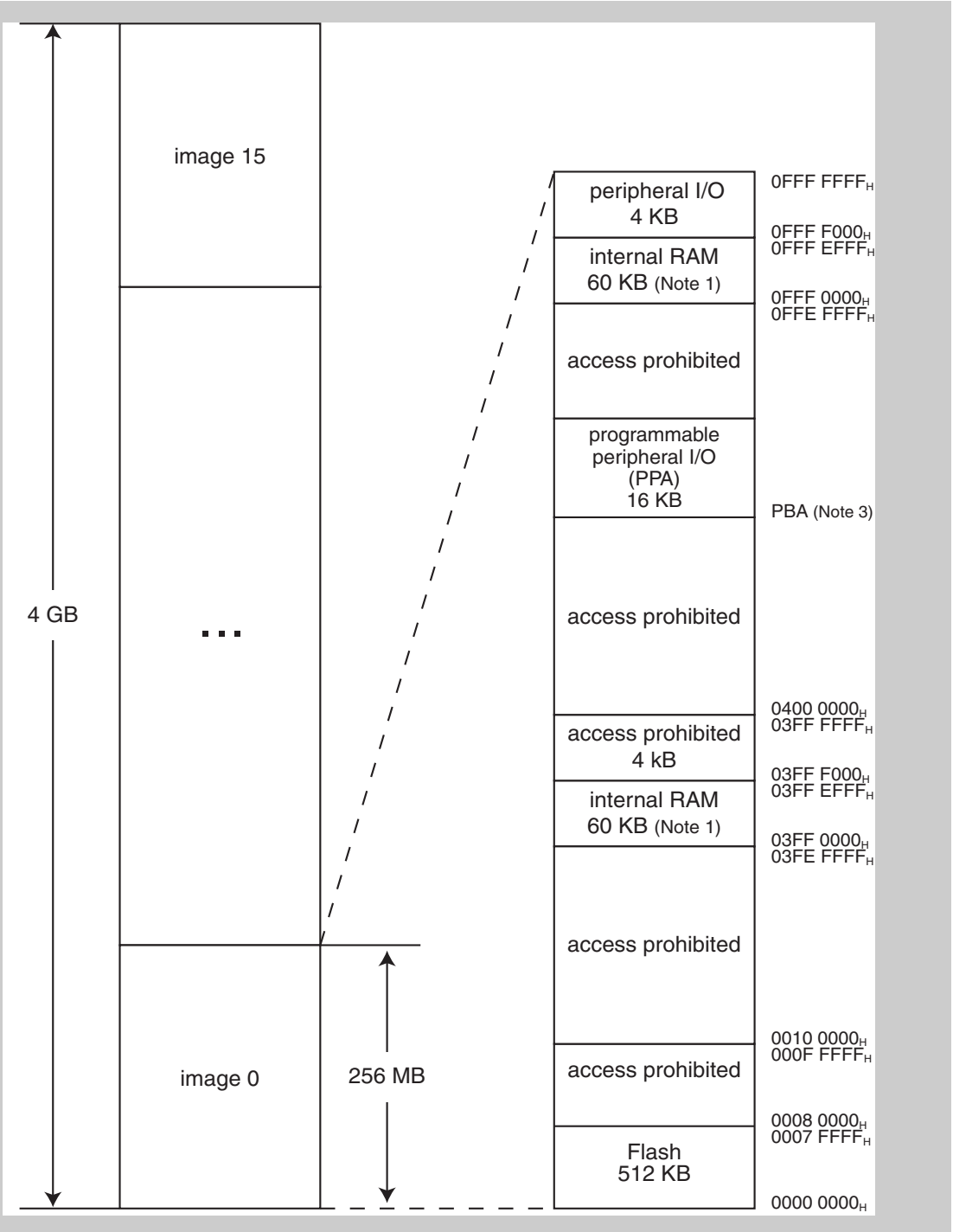
The CPU supports the following address space:

- 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.
- 256 MB physical address space
  The CPU provides 256 MB physical address space. That means that a maximum of 256 MB of internal memory can be accessed.

Any 32-bit address is translated to its corresponding physical address by ignoring bits 31 to 28 of the address. Thus, 16 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 $1000\ 0000_H$, $2000\ 0000_H$, …, $E000\ 0000_H$, or $F000\ 0000_H$.

The 256 MB physical address space is seen as 16 images in the 4 GB CPU address space:

Figure 3-2 Images in the CPU address space



Note 1. The internal RAM area (03FF 0000$_H$ and 03FF EFFF$_H$) is mirrored to the area 0FFF 0000$_H$ to 0FFF EFFF$_H$. If data is written in one area, it appears also in the other area.

2. The programmable peripheral base address is defined by the BPC register.

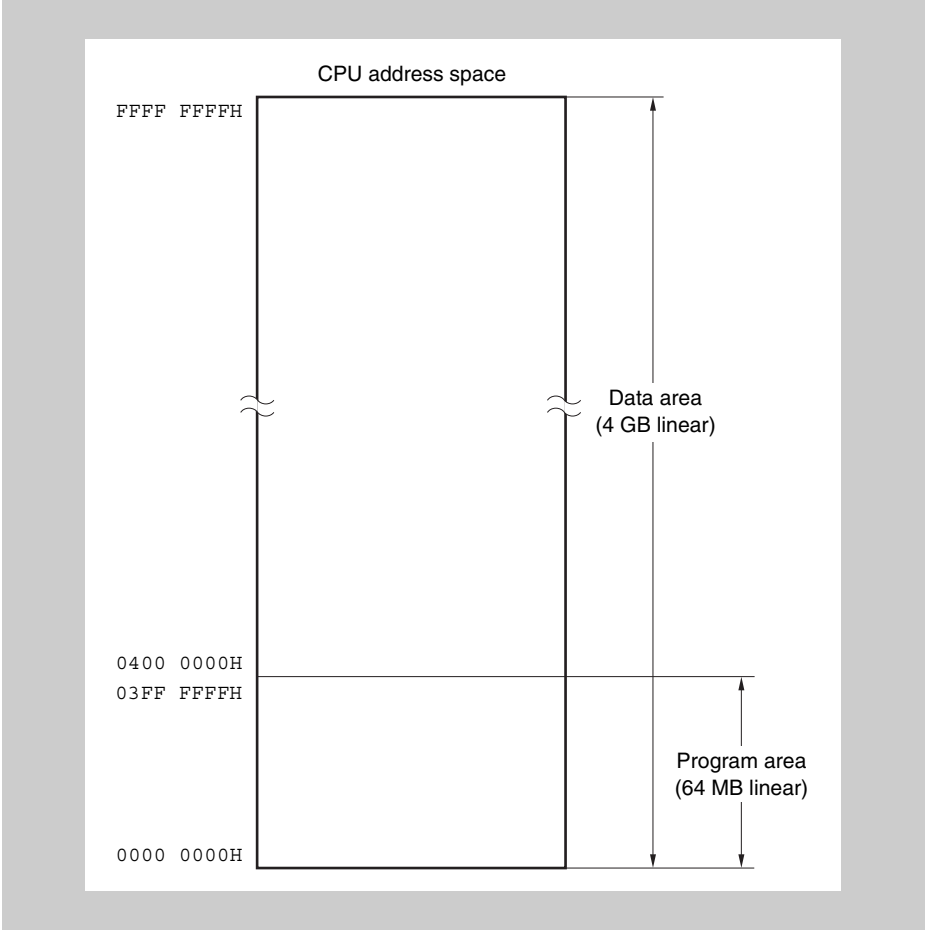
3.4.2 Program and data space

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

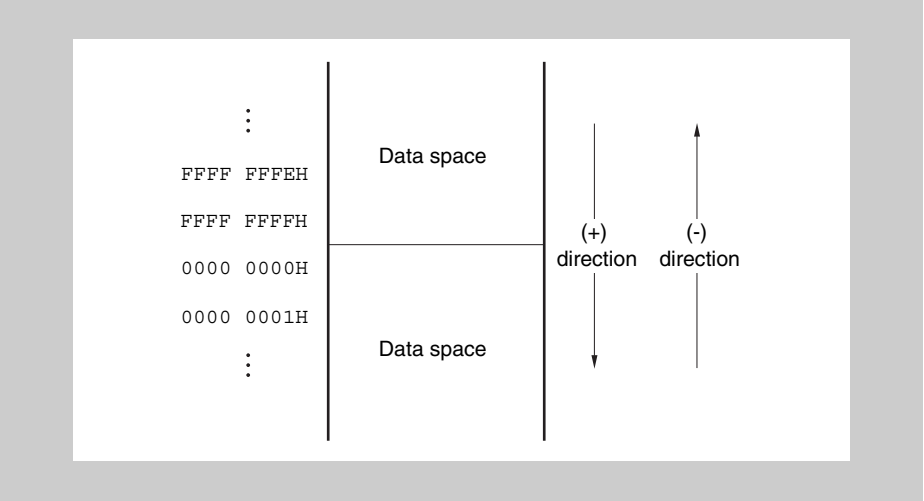
Figure 3-3 CPU address space



(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.

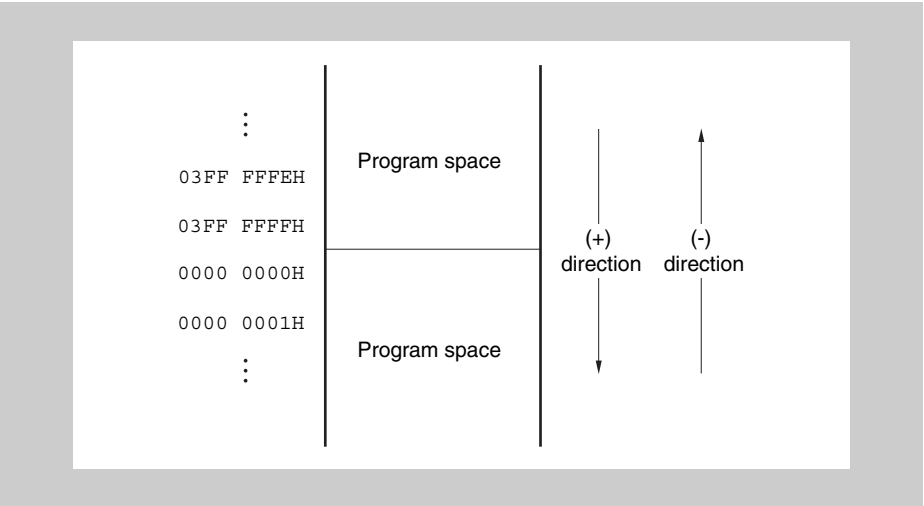
Figure 3-4 Wrap-around of data space



(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 3-5 Wrap-around of program space



Caution No instruction can be fetched from the 4 KB area of 0FFF $F000_H$ to 0FFF $FFFF_H$ because this area is defined as peripheral I/O area. Therefore, do not execute any branch to this area.

3.4.3 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 the pointer register. With relative

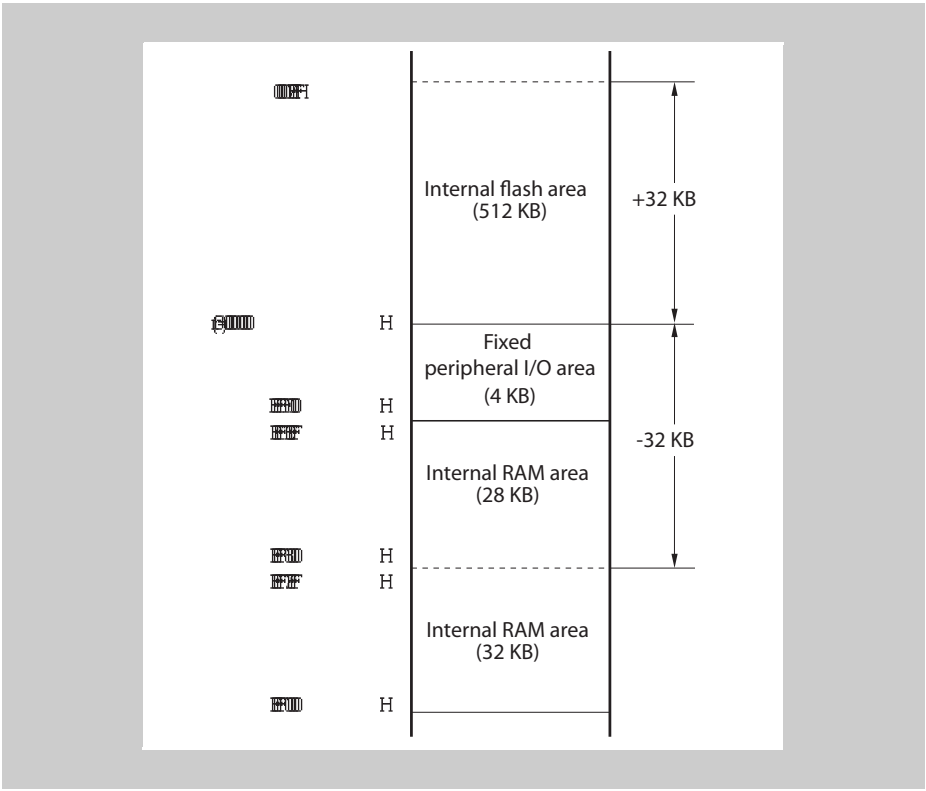
addressing, an instruction can access operand data at all addresses that lie in the range of ±32 KB relative to the address in the pointer register.

Using 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).
- Locate flash memory data segments in the area up to 0000 7FFFH, so that access to these constant data can utilize also relative addressing.
- Use the r0 register as a 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 3-6 Example application of wrap-around



3.5 Memory

The following sections introduce the CPU memory. They describe specific memory areas and give a recommendation for address space usage.

Memory areas The internal memory of the CPU provides several areas, which are listed here and then briefly described below:

- Internal flash area for flash memory devices
- Internal RAM area
- Internal fixed peripheral I/O area
- Programmable peripheral I/O area

Peripheral I/O areas Two areas of the address range are reserved for the registers of the on-chip peripheral functions. These areas are called “peripheral I/O areas”.

Table 3-12 Peripheral I/O areas

Name	Address range	Size
Fixed peripheral I/O area	0FFF $F000_H$ to 0FFF $FFFF_H$	4 KB
Programmable peripheral I/O area (PPA)	Can be allocated at arbitrary addresses. Base address is defined in the BPC register.	16 KB

Additionally, the bus which is used by the CPU to communicate to some of the internal memory (data flash) and peripheral I/O areas must be configured for optimum performance. The description of the registers and the appropriate configuration values are given in subsequent subsections.

Configuration registers The internal bus configuration registers includes the following.

- Internal peripheral function wait control register (VSWC)
- Bus cycle configuration register (BCT1)
- Data wait control register (DWC1)
- Bus cycle control register (BCC)

3.5.1 Internal flash area

Table 3-13 summarizes the size and addresses of the physical internal ROM (Flash Memory) for each device.

Table 3-13 Internal ROM (Flash Memory)

Device	ROM (Flash) Sizes	Address Range
µPD70F3464	256 KB	0000 0000_H - 0003 $FFFF_H$
µPD70F3465	384 KB	0000 0000_H - 0005 $FFFF_H$
µPD70F3466	512 KB	0000 0000_H - 0007 $FFFF_H$
µPD70F3470	256 KB	0000 0000_H - 0003 $FFFF_H$
µPD70F3471	384 KB	0000 0000_H - 0005 $FFFF_H$
µPD70F3472	512 KB	0000 0000_H - 0007 $FFFF_H$

3.5.2 Internal RAM area

The 32 KB area between addresses 03FF 7000_H and 03FF $EFFF_H$ is provided as the internal RAM area. *Table 3-14* summarizes the addresses of the internal RAM area for each device.

Table 3-14 Internal RAM size and address range

Device	RAM Size	Address Range
µPD70F3464	16 KB	03FF $B000_H$ - 03FF $EFFF_H$
µPD70F3465	24 KB	03FF 9000_H - 03FF $EFFF_H$
µPD70F3466	32 KB	03FF 7000_H - 03FF $EFFF_H$
µPD70F3470	16 KB	03FF $B000_H$ - 03FF $EFFF_H$
µPD70F3471	24 KB	03FF 9000_H - 03FF $EFFF_H$
µPD70F3472	32 KB	03FF 7000_H - 03FF $EFFF_H$

Note The internal RAM area is mirrored to the area 0FFF 0000_H to 0FFF $EFFF_H$. If data is written in one area, it appears also in the other area.

3.5.3 Internal fixed peripheral I/O area

The 4 KB area between addresses 0FFF $F000_H$ and 0FFF $FFFF_H$ 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, DMA, bus and Memory Controllers
- Configuration registers of the clock controller

For a list of all peripheral I/O registers, see *Section Appendix A, Special Function Registers* on page 625.

Note

1. Because the physical address space covers 256 MB, the address bits A[31:28] are not considered. Thus, this address space can also be addressed via the area FFFF $F000_H$ 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 0FFF $F000_H$ to 0FFF $FFFF_H$.
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, then the higher 8 bits become undefined.
   - During write operation, then the lower 8 bits of data are written to the register.

Cautions

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

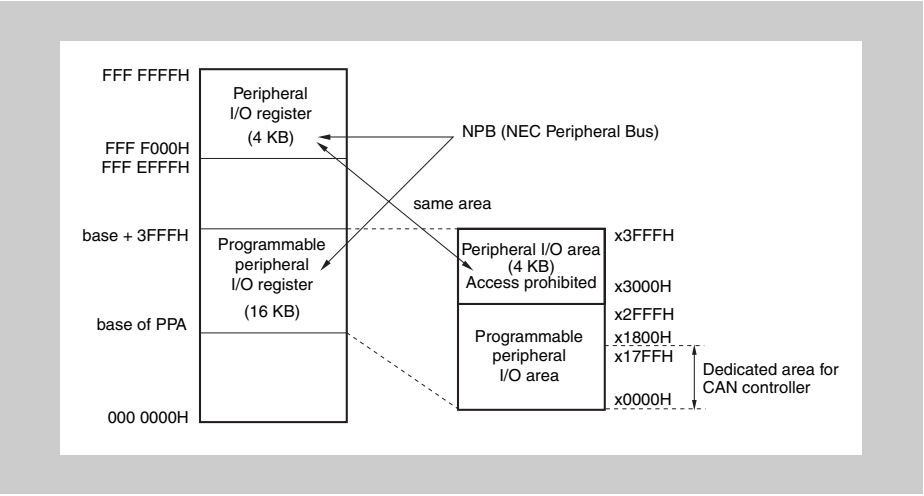
2. For DMA transfer, the fixed peripheral I/O area (0FFF $F000_H$ to 0FFF $FFFF_H$) cannot be specified as the source/destination address. Be sure to use the RAM area (0FFF 0000_H to 0FFF $EFFF_H$ or 03FF 0000_H to 03FF $EFFF_H$) for source/destination address of DMA transfer.

3.5.4 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).

Figure 3-7 Programmable peripheral I/O area



The CAN modules registers and message buffers are allocated to the PPA. Refer to *Section 17.5.1, CAN module register and message buffer addresses* on page 428 for information how to calculate the register and message buffer addresses of the CAN modules.

Access timing

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.

Cautions

1. 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 area
   - RAM area
2. 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. Access to this mirror of the *fixed* peripheral I/O area is prohibited.

Note All address definitions in this manual that refer to the programmable peripheral area assume that the base address of the PPA is 840 0000_H, that means recommended value for BPC is $A100_H$.

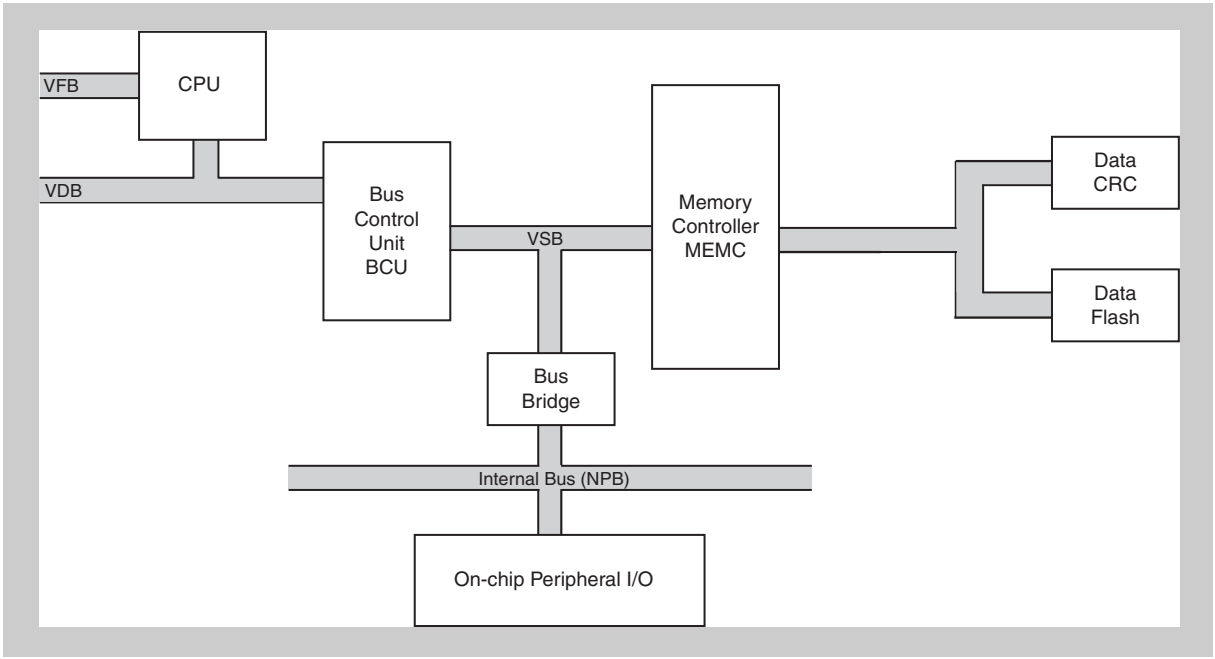
3.6 Bus and Memory Controller (BCU, MEMC)

The Bus Control Unit BCU and Memory Controller MEMC control the access to on-chip peripheral I/Os.

3.6.1 Overview

The figure below shows a block diagram of the modules that are necessary for accessing on-chip peripherals and data flash.

Figure 3-8 Bus and Memory Control block diagram



Busses The busses are abbreviated as follows:

- NPB: NEC peripheral bus
- VSB: V850 system bus
- VDB: V850 data bus
- VFB: V850 fetch bus

BCU The Bus Control Unit (BCU) controls the access to on-chip peripherals and to external I/O.

Memory Controller The 256 MB address range is divided into 2-MB memory banks and 64-MB memory areas. On V850E/RG3 devices, there are no external bus interface but the memory controller is still used to access the CRC and the internal data flash.

Consequently, the MEMC must be configured before accessing the Data Flash of the CRC macro.

3.6.2 Peripheral I/O areas access

The BCU controls the access to the Peripheral I/O areas via the registers VSWC and BPC.

For further information on the peripheral I/O areas, see *Chapter 3, Section 3.5.4, Programmable peripheral I/O area (PPA)*.

3.6.3 CRC module access

The CRC module is connected to the memory bus, although it is built into the microcontroller.

Table 3-15 shows all required register settings for the CRC module's access. Some settings are appropriate for the CRC module per default, others must be changed before the first access to the CRC module.

For details about the control settings refer to the description of the registers.

Table 3-15 Register settings for CRC module access

Control bit	Required setting	Comment
BCT1.ME4	1	• enable access • not default value, must be changed
DWC1.DWC4[2:0]	000_B	• no data wait states • not default value, must be changed
BCC.BC41	0	• no idle states • no default value, must be changed

Base address The base address of the CRC module is 0800 0300_H.

3.6.4 Data flash module access

The data flash module is connected to the memory bus, although is built into the microcontroller.

Table 3-16 shows all required register settings for the data flash module's access. Some settings are appropriate for the data flash module per default, others must be changed before the first access to the Data Flash module.

For details about the control settings refer to the description of the registers.

Table 3-16 Register settings for Data Flash module access

Control bit	Required setting	Comment
BCT1.ME7	1	• enable access • not default value, must be changed
DWC1.DWC7[2:0]	000_B	• no data wait states • not default value, must be changed
BCC.BC71	0	• no idle states • no default value, must be changed

Base address The base address of the data flash is 0FE0 0000_H.

3.6.5 BCU Control Registers

(1) BPC - Peripheral Area Selection Control Register

The BPC register specifies selection of the PPA.

Control registers for the on-chip CAN interface are implemented in the PPA. To use the CAN interface, set PA15 to 1 by writing the BPC register with a 16-bit memory manipulation instruction. To disable access to the CAN RAM and CAN registers, set PA15 to 0 by writing 0000_H to the BPC register with a 16-bit memory manipulation instruction.

Access The BPC register can be read or written only in 16-bit units.

Address $FFFF\ F064_H$

Initial Value 0000_H

15	14	13	12	11	10	9	8
PA15	0	PA13	PA12	PA11	PA10	PA09	PA08
R/W	R	R/W	R/W	R/W	R/W	R/W	R/W

7	6	5	4	3	2	1	0
PA07	PA06	PA05	PA04	PA03	PA02	PA01	PA00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

PA15	Usage of programmable peripheral I/O area
0	Usage of programmable peripheral I/O area is disabled
1	Usage of programmable peripheral I/O area is enabled

Bit name	Function
PA13 to PA00	Specify an address in programmable peripheral I/O area

Note The base address of the so called Programmable Peripheral Area (PPA) is allocated by the BPC register of the CPU core. The recommended setup value for BPC is 0xA100. This value allocates the PPA starting from 0x08400000 to 0x08402FFF. All CAN macros are located in this area.

(2) VSWC - Internal peripheral function wait control register

The VSWC register controls the bus access wait for the on-chip peripheral I/O registers. Both address setup and data wait states are based on the system clock.

Access to on-chip peripheral I/O registers is made in 3 clocks (without wait), however, waits may be required depending on the operation frequency. Set the values described below to the VSWC register in accordance with the operation frequency used.

For DMA transfers, the bus access wait and signal timing is controlled by DMAWC0 register. For a description of these registers see *Section Chapter 8, DMA Controller (DMAC)* on page 145.

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

SUWL2	SUWL1	SUWL0	Address setup wait for internal bus
0	0	0	0
0	0	1	1 CPU system clock (VBCLK)
0	1	0	2 CPU system clock (VBCLK)
0	1	1	3 CPU system clock (VBCLK)
1	0	0	4 CPU system clock (VBCLK)
1	0	1	5 CPU system clock (VBCLK)
1	1	0	6 CPU system clock (VBCLK)
1	1	1	7 CPU system clock (VBCLK)

VSWL2	VSWL1	VSWL0	Data wait for internal bus
0	0	0	0
0	0	1	1 CPU system clock (VBCLK)
0	1	0	2 CPU system clock (VBCLK)
0	1	1	3 CPU system clock (VBCLK)
1	0	0	4 CPU system clock (VBCLK)
1	0	1	5 CPU system clock (VBCLK)
1	1	0	6 CPU system clock (VBCLK)
1	1	1	7 CPU system clock (VBCLK)

The following setups are recommended for VSWC

System clock	Recommended VSWC value
64MHz	0x13
80MHz	0x24

3.6.6 MEMC Control Registers

(1) BCT1 - Bus cycle configuration register

The BCT1 register enables CPU access to the internal Data Flash area.

Access This register can be read/written in 16-bit units.

Address FFFF F482$_H$

Initial Value 4444$_H$

Bit Name	Data Flash Access
ME4	0: Disable access to CRC macro 1: Enable access to CRC macro
ME7	0: Disable access to Data Flash 1: Enable access to Data Flash

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ME7	1	0	0	0	1	0	0	0	1	0	0	ME4	1	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/W

Cautions

1. The bits marked with 0 must always be 0.
2. The bits marked with 1 must always be 1.
3. Before attempting to access the internal Data Flash or the CRC macro, the BCT1 register must be configured correctly. Do not change this register after initialization.

(2) DWC1 - Data wait control registers

The DWC1 register controls the number of wait states for memory access.

Access This register can be read/written in 16-bit units.

Address FFFF F486$_H$

Initial Value 3333$_H$

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	DW72	DW71	DW70	0	DW62	DW61	DW60	0	DW52	DW51	DW50	0	DW42	DW41	DW40
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

DWn2	DWn1	DWn0	Number of inserted wait states
0	0	0	No wait state inserted: recommended value
1	1	1	Reset state
Other than above			Prohibited

Caution The bits marked with 0 must always be 0.

Note For access to internal memory and peripheral I/O areas, programmable wait states are ***not*** carried out.

(3) BCC - Bus cycle control register

The BCC register controls insertion of idle state after any read access.These registers must be initialized as described below.

Access This register can be read/written in 16-bit units.

Address FFFF F48A$_H$

Initial Value AAAA$_H$:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BC71	0	BC61	0	BC51	0	BC41	0	BC31	0	BC21	0	BC11	0	BC01	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/W

Note The default value 0 of bits 14, 12, 10, 8, 6, 4, 0 and 0 must not be changed.

BCn1	Idle state insertion enable/disable
0	No idle state inserted: Recommended value
1	Idle state inserted after any read access (Reset value)

Caution Do not change this register after initialization.

Note For access to internal memory and peripheral I/O areas, *no* idle states should be inserted.

3.7 RAM ECC error detection

The RAM control circuit is equipped with an error detection function. A 5-bit ECC (Error Correction Coding) code is dedicated to each 8-bit word in the RAM.

The ECC can detect and correct single bit errors and detect double bit errors. In the latter case the double error detection interrupt INTDEDR is generated.

INTDEDR can generate two kinds of interrupts:

- an NMI via the NMI sharing function
- a maskable interrupt (exception code 0080_H), that is shared with double error detections of the flash memory function.

Besides generating an non-maskable or maskable interrupt, the address of the erroneous data and the location of the erroneous data byte of the 32-bit data are saved in registers.

Note A RAM access is always performed with 32-bit, i.e. 4-byte, data.
Thus a double error detection interrupt INTDEDR is also generated upon an 1-byte or 2-byte access, even if the read target byte(s) are not erroneous, but any other data byte that is read concurrently.

3.7.1 RAM ECC initialization

The RAM content and the 5-bit ECC values are undefined after device power-up, the 5-bit ECC values do not fit to their respective 8-bit data. Thus the entire RAM must be initialized, i.e. written, in order to fit the ECC values to the data. Since a double error detection interrupt INTDEDR may have been issued due to undefined data and ECC values after power-up, follow below procedure to start up the RAM ECC:

1. After device power-up NMI sharing is disabled (INTSEL.ISDR = 1) and the maskable interrupt is masked (ERRIC.ERRMK = 1). Don’t enable NMI sharing respectively unmask the maskable interrupt before the RAM has been completely initialized.
2. Initialize the entire RAM by writing to all RAM locations. The data written is not of concern.
3. Enable NMI sharing by INTSEL.ISDR = 0 or unmask the maskable interrupt by ERRIC.ERRMK = 0, whatever is desired.

For further information on interrupt sharing, please refer to *Section 5.5, Interrupt Sharing* on page 106.

Also when program code shall be executed from RAM, it is recommended to fill the entire RAM, preferably all data bytes with the NOP instruction code, before fetching the first instruction from RAM. This avoids ECC errors generated by instruction prefetches by the execution pipeline.

3.7.2 RAM ECC registers

The RAM ECC is operated by means of the following registers.

Table 3-17 RAM ECC registers overview

Register name	Shortcut	Address
RAM ECC error address register	RAMEAD	FFFF F8B0$_H$
RAM ECC error data location register	RAMEDLR	FFFF F8B2$_H$
RAM ECC control register	RAMECC	FFFF F8B4$_H$

(1) RAMEAD - RAM ECC error address register

The 16-bit RAMEAD register holds the lower 16 bit of the address of the RAM location, where an error was detected first.

Access This register can be read in 16-bit units.

Address FFFF F8B0$_H$

Initial Value 0000$_H$

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

Table 3-18 RAMEAD register contents

Bit Position	Bit Name	Function
15 - 2	RAMEADDR	16-bit address of first detection of a RAM error.

The address of the first error detection is stored in RAMEAD. At the same time the interrupt flag INTERRF.INTERR1 is set to 1. Additionally the erroneous byte location is indicated in the RAMEDLR register.

This register is not overwritten by a new error detection until:

- the INTERRF.INTERR1 is cleared to 0. Thus no new error is signaled by interrupt INTDEDR until the INTERRF.INTERR1 is cleared to 0.
- RAMEAD is cleared to 0000$_H$ by any reset.

After one of the above conditions has been fulfilled, the lower 16 bit of a new ECC detection address can be stored and a new INTDEDR interrupt can be generated.

Caution The RAMEAD register is undefined after the INTERR1 bit of the INTERRF register is cleared to 0. Therefore RAMEAD should be read prior to clearing the INTERR1 bit of the INTERRF register to 0.

For further information on the register INTERRF, please refer to chapter *Section 5.5.2, INTERRF Interrupt source flag register* on page 107.

(2) RAMEDLR - RAM ECC error data location register

The RAMEDLR indicates which byte of the 32-bit data byte at the address RAMEAD has generated an error.

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

Address FFFF F8B2$_H$

Initial Value 00$_H$

7	6	5	4	3	2	1	0
0	0	0	0	REBEN3	REBEN2	REBEN1	REBEN0
R	R	R	R	R	R	R	R

Table 3-19 RAMEDLR register contents

Bit Position	Bit Name	Function
3	REBEN3	ECC error status in data bits 31 to 24: 0: ECC error did not occur in bits 31 to 24 1: ECC error occurred in bits 31 to 24
2	REBEN2	ECC error status in data bits 23 to 16: 0: ECC error did not occur in bits 23 to 16 1: ECC error occurred in bits 23 to 16
1	REBEN1	ECC error status in data bits 15 to 8: 0: ECC error did not occur in bits 15 to 8 1: ECC error occurred in bits 15 to 8
0	REBEN0	ECC error status in data bits 7 to 0: 0: ECC error did not occur in bits 7 to 0 1: ECC error occurred in bits 7 to 0

The content of RAMEDLR is related to the error detection address in RAMEAD, thus only valid after an error detection interrupt INTDEDR, i.e. if INTERRF.INTERR1 = 1.

Caution The RAMEDLR register is undefined after the INTERR1 bit of the INTERRF register is cleared to 0. Therefore RAMEDLR should be read prior to clearing the INTERR1 bit of the INTERRF register to 0.

For further information on the register INTERRF, please refer to chapter *Section 5.5.2, INTERRF Interrupt source flag register* on page 107.

(3) RAMECC - RAM ECC control register

RAMECC control the operation of the RAM ECC function.

Writing to this register is only possible immediately after writing to the associated write protection register.
First write to the PRCMDIRA register. The contents is ignored. Then, you are permitted to write once to the RAMECC register. This must be done immediately after writing to the PRCMDIRA register. After the second write action, or if the second write action does not follow immediately, all protected registers are write-locked again.

For further details about write protected registers, please refer to *Section 3.8.1, Write protection control register* on page 67.

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

Address FFFF F8B4$_H$

Initial Value 00$_H$

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	RAEENB
R	R	R	R	R	R	R	R

Table 3-20 RAMECC register contents

Bit Position	Bit Name	Function
0	RAEENB	ECC function enable: 0: ECC function enabled 1: ECC function disabled

Cautions
1. Before modifying RAEENB make sure to stop any DMA transfers.
2. Confirm correct setting of RAEENB by re-reading this bit after its modification.

3.7.3 RAM ECC function check

In order to check the functionality of the RAM ECC error detection, you can proceed as described below:

1. enable ECC function by RAMECC.RAEENB = 0
5. write some data word (e.g. data_1) to the RAM: correct ECC will be generated
6. disable ECC function by RAMECC.RAEENB = 1
7. write another data word (e.g. data_2) to the RAM address of the first data word data_1: no new ECC will be generated, thus ECC does not fit to data_2
8. enable ECC function by RAMECC.RAEENB = 0
9. read data_2: a maskable or non-maskable ECC error interrupt should be generated

3.8 Write Protected Registers

Write protected registers are protected from inadvertent write access due to erroneous program execution. 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)
10. 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 3-21 Overview of write protected registers

Write protected register	Shortcut	Corresponding write enable register	Shortcut
Clock selection register 0	OCKS0	Command register	PRCMD
RAM ECC control register	RAMECC	RAM ECC Control Register	PRCMDIRA

Example Write to RAMECC:

```
PRCMDPIRA = 0x80;
RAMECC = 0x80;
```

Note
1. Make sure that the compiler generates two consecutive assembler “store” instructions to PRCMDPIRA and RAMECC from the associated C statements.
2. It is recommended to write the same value in both the enable register and the write protected register.

Since any action between writing to a “standard” register and writing to a protected register destroys this sequence, the effects of interrupts and DMA transfers have to be considered:

- Interrupts:
  After writing to PRCMDPIRA, no maskable interrupts will be acknowledged, until the subsequent instruction has been executed. Thus a maskable interrupt can not destroy the sequence.
  However, any non-maskable interrupt can still be acknowledged.
- DMA:
  In the above example, DMA transfers can still take place. They may destroy the sequence.

  If appropriate, you may disable DMA transfers in advance. Otherwise you must check whether writing to the protected register was successful. To do so, check the status via the status register, if available, or by reading back the protected register.

3.8.1 Write protection control register

The following section describes the register that controls access to write protected registers.

(1) PRCMD - Command register

The PRCMD register protects other registers from inadvertent write access, so that the system does not stop in case of a program hang-up.

After writing to the PRCMD register, it is permitted to write once to one of the protected registers. This must be done immediately after writing the PRCMD register. After the second write action all protected registers are write-locked again. Read access to any registers are permitted between write access to the PRCMD and the protected register.

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

Address FFFF F1FC$_H$

Initial Value The content 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

(2) PRCMDIRA - iRAM interface command register

PRCMDIRA is a write protection register for iRAM interface macros registers. Write to the iRAM control registers is only possible, if any data has been written into PRCMDIRA immediately before writing to the iRAM interface macro.

After writing to the PRCMDIRA register, it is permitted to write once to one of the protected registers. This must be done immediately after writing the PRCMDIRA register. After the second write action all protected registers are write-locked again. Read access to any registers are permitted between write access to the PRCMDIRA and the protected register.

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

Address FFFF FF2E$_H$

Initial Value The content 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

Chapter 4 Clock Generator

The Clock Generator provides the clock signals needed by the CPU and the on-chip peripherals. This chapter discusses the clock generator functions and control registers, and provides information about clock generator operation.

4.1 Overview

The Clock Generator can generate the required clock signals from the following sources:

- Main oscillator - a built-in oscillator that requires a crystal with a frequency of 16 MHz for 3V devices or 8 MHz for 5V devices.
- High-frequency internal ring oscillator - an internal oscillator without external components and a nominal frequency of 6.8 MHz (typ.).

Features summary

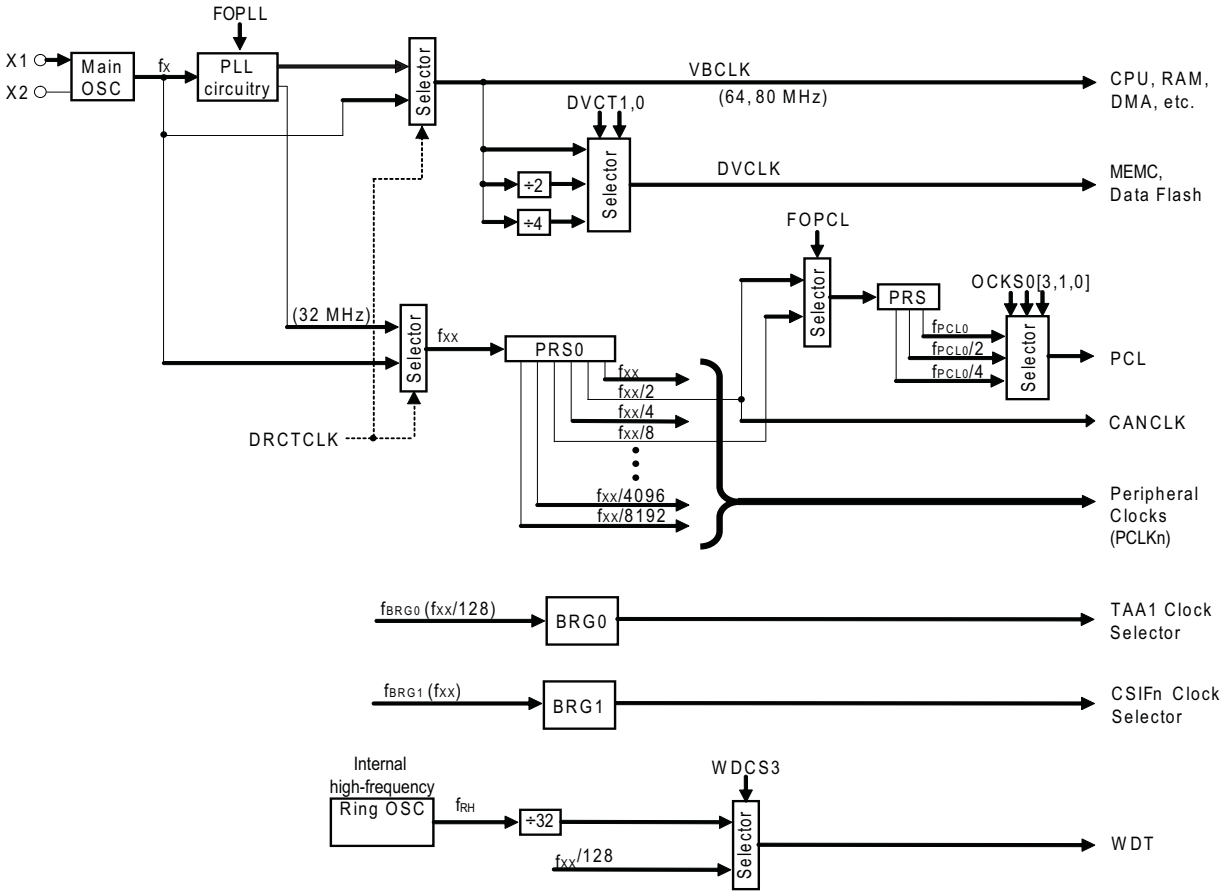
Special features of the clock generator are:

- PLL synthesizer for the main oscillator:
  - Clock oscillator multiplier which generates high speed internal CPU clock
  - Supplies 32 MHz source clock for peripheral clock prescaler
- Peripheral supply clock generation
  - 32 MHz to 3.906 KHz
- HALT mode
- Vital clock configuration registers are write-protected by a special write sequence
- Programmable clock output function (PCL)

4.1.1 Description

The following figure shows a simplified block diagram of the Clock Generator.

Figure 4-1 Clock Generator Block Diagram



* : 3V devices only

Note
- f_X: Main oscillator output clock (16 MHz or 8 MHz)
- f_{XX}: Source clock for Prescaler which supplies clock sources for all peripherals and programmable clock output. Normal operation: 32 MHz
- VBCLK: Clock to CPU (64 MHz or 80 MHz)

An external resonator or crystal must be connected to the X1 and X2 pins. Its frequency is multiplied by the PLL synthesizer. The PLL provides an internal system clock of 32 MHz, regardless of the frequency of VBCLK. The multiplication factor of the PLL is determined from the Flash mask option register.

The clock controller enables the PLL automatically and enables the clock supply to the system after the oscillation stabilization time has passed.

VBCLK The CPU clock is referred to as VBCLK. The frequency of VBCLK is determined by the settings of the Flash Mask option register. Depending on the device, this frequency can be set to 64MHz or 80MHz.

PCLKn The peripheral modules are supplied with a set of 14 peripheral clock sources, PCLK0 to PCLK13, derived from a 32 MHz source clock supplied by the PLL. The maximum frequency available is PCLK0 with a frequency of 32 MHz. The other PCLKn frequency values are binary derivatives of the source clock,

equal to half of its predecessor PCLK(n-1). See *Table 4-1* on page 71 for the frequencies associated with each peripheral clock derivative.

Table 4-1 PCLKn Frequency Table

Peripheral Clock	Frequency
PCLK0	32 MHz
PCLK1	16 MHz
PCLK2	8 MHz
PCLK3	4 MHz
PCLK4	2 MHz
PCLK5	1 MHz
PCLK6	500 KHz
PCLK7	250 KHz
PCLK8	125 KHz
PCLK9	62.5 KHz
PCLK10	31.25 KHz
PCLK11	15.63 KHz
PCLK12	7.81 KHz
PCLK13	3.91 KHz

DRCTCLK This is not accessible to the user. On startup, the main oscillator drives the clock circuitry. After the stabilization time passes, the DRCTCLK value automatically changes and the PLL drives the clock circuitry.

4.2 Clock Generator Registers

The clock generator is controlled and operated by means of the following registers.

Table 4-2 Clock generator registers overview

Register name	Shortcut	Address
Flash Mask Option Register	FMOP0	0000 007A$_H$
Bus Clock Divide Register	DVC	FFFF F48E$_H$
Clock Control Register	CKC	FFFF FD1A$_H$
Integer Prescaler Control Register	OCKS0	FFFF FF28$_H$

4.2.1 Clock Control Register (CKC)

The CKC register indicates the current system clock status.

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

Address FFFF FD1A$_H$

Initial Value 03$_H$. This register is cleared by any reset.

7	6	5	4	3	2	1	0
PLLSTAT	0	0	0	0	0	1	1
R	R	R	R	R	R	R	R

PLLSTAT	System clock resource
0	System is running on main oscillator
1	System is running on PLLC clock

The system starts on main oscillator and automatically switches to PLLC after PLL stabilization time.

4.2.2 Bus Clock Divide Register (DVC)

The DVC register is an 8-bit register used to define the operating frequency of DVCLK. The internal clock signal DVCLK is used to clock the MEMC interface and the data flash. The table below lists the proper configuration of the register.

Access This register can be read or written in 8-bit unit.

Address FFFF F48E$_H$

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	DVC1	DVC0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

CPU freq (MHz)	DVCT1	DVCT0	Bus clock divider selection
64	0	0	$f_{DVCLK} = f_{VBCLK} /2$
80	0	0	$f_{DVCLK} = f_{VBCLK} /2$
			Other Setting prohibited

4.2.3 Flash Mask Option Register (FMOP0)

The FMOP0 register is a 16-bit register used to configure the operation of the watchdog timer, determine the input clock frequency of the PCL prescaler, and selects the PLL multiplier settings that determine the resulting CPU frequency. This register cannot be modified by the user application.

Access The register is located in the flash memory in the device. Therefore it can only be changed via flash programming. Read access is possible in 8/16-bit units.

Address 0000 007A$_H$

Initial Value The reset value depends on the settings of the options written to the flash.

15	14	13	12	11	10	9	8
1	1	1	1	1	1	1	WDTMD
R	R	R	R	R	R	R	R

7	6	5	4	3	2	1	0
FOPCL	1	1	1	1	1	FOPLL1	FOPLL0
R	R	R	R	R	R	R	R

WDTMD	WDT Operation Mode		
	Count Operation	Input Clock	Operation Mode
0	Can be stopped by WDCS24 bit of WDTM register	Selectable by WDTM register: ring clock or main clock	Selectable by WDTM register: NMI interrupt mode or Reset mode
1	Cannot be stopped (WDTM fixed to 01101xxxB)	Fixed at ring clock	Fixed to Reset mode

FOPCL	PCL prescaler supply clock frequency (f_{PCL0})
0	f_{PCL0} = PCLK1 (16 MHz)
1	f_{PCL0} = PCLK3 (4 MHz)

FOPLL1[a]	FOPLL0[a]	CPU Frequency
0	1	64 MHz
0	0	80 MHz

a) Refer to *Table 4-3* for the specified value to use.

Caution Do not use other values for FOPPL1 and FOPPL0 than specified in *Table 4-3* below.

The following table indicates which value of FOPLL1 and FOPPL0 should be used on each V850E/RG3 device.

Table 4-3 Specified value of FOPLLn bits for each V850E/RG3 device

Device	Specified values		CPU Frequency
	FOPLL1	FOPLL0	
µPD70F3464	0	1	64 MHz
µPD70F3465	0	1	64 MHz
µPD70F3465	0	0	80 MHz
µPD70F3470	0	0	80 MHz
µPD70F3471	0	0	80 MHz
µPD70F3472	0	0	80 MHz

4.2.4 Integer Prescaler Control Register (OCKS0)

The OCKS0 register is an 8-bit register that controls the operation of the integer prescaler divider for the programmable clock (PCL) output.

Access This register can be read in 8-bit or 1-bit units. This register is write-protected and data must be written to it in a specific sequence.

Address FFFF FF28$_H$

Initial value 00$_H$

7	6	5	4	3	2	1	0
0	0	0	OCKSEN	OCKSTH	0	OCKS01	OCKS00
R	R	R	R/W	R/W	R	R/W	R/W

OCKSEN	Integer prescaler operation control
0	Operation is disabled
1	Operation is enabled

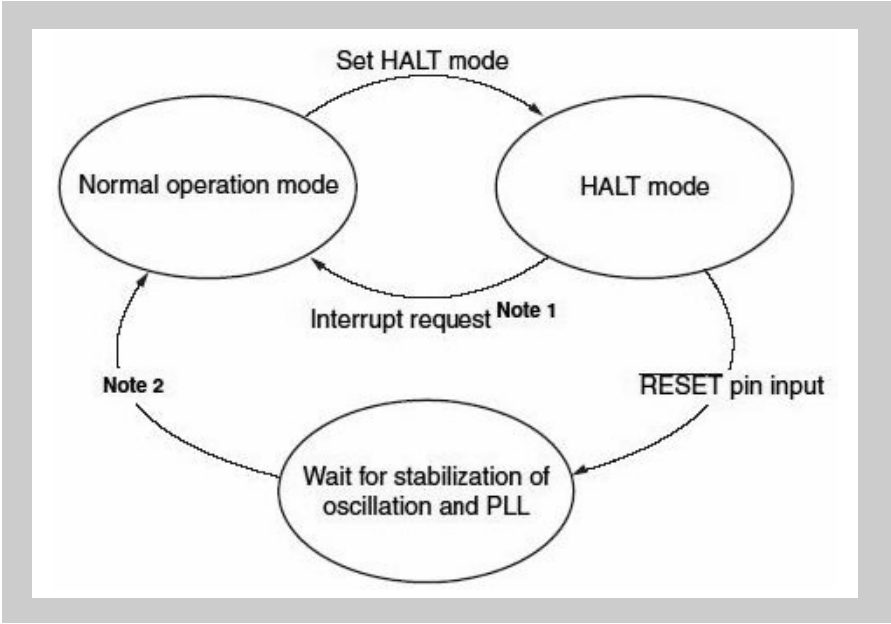
OCKSTH	OCKS01	OCKS00	PCL Output (divisor selector)
0	0	0	$f_{PCL0}/2$
0	1	0	$f_{PCL0}/4$
1	0	0	f_{PCL0}
Other than above			prohibited

4.3 HALT mode

The power save function of this device supports the HALT mode. In the HALT mode, the Clock Generator (oscillator and PLL synthesizers) continues to operate, but the CPU clock and hence program execution stops. The built-in RAM keeps the state it had before the HALT mode was set. The clock supply to the on-chip peripherals continues. Peripheral devices that do not require instruction processing remain operating.

The system is switched to HALT mode by a specific instruction (the HALT instruction). *Figure 4-2* shows the transitions between normal operation mode and HALT mode.

Figure 4-2 Power Save Mode State Transition Diagram



Note 1. Non-maskable interrupt request signal (NMI) or unmasked maskable interrupt request signal.

2. The oscillation stabilization time is necessary after release of reset because the PLL is initialized by a reset. The stabilization time is determined by hardware.

4.3.1 Setting and operation status

The HALT mode is set when a dedicated instruction (HALT) is executed in the normal operation mode.

In this mode the clock supllied to the CPU is stopped. The Clock Generator and PLL continue operating. Clock supply to the other on-chip peripheral functions continues.

As a result, program execution is stopped, and the content of the internal RAM is retained before the HALT mode was set. The on-chip peripheral functions that are independent of instruction processing by the CPU continue operating.

Table 4-4 shows the operation status in the HALT mode.

The average power consumption of the system can be reduced by using the HALT mode in combination with the normal operation mode for intermittent operation.

Cautions

1. Insert five or more NOP instructions after the HALT instruction.
2. If the HALT instruction is executed while an interrupt request is being held pending, the HALT mode is set but is released immediately by the pending interrupt request.

The following table shows the operation status in the HALT mode.

Table 4-4 Operation Status in HALT Mode

Function	Operation Status
Main oscillator	Operating
PLL	Operating
High speed internal oscillator	Operating
CPU System	Suspended
DMA	Operating
Interrupt controller	Operating
Watchdog Timer	Operating
Ports	Maintained
On-chip peripheral I/O (excluding ports)	Operating
Internal data	All internal data such as CPU registers, states, data, and the contents of internal RAM are retained in the state they were before HALT mode was set.

4.3.2 Releasing HALT mode

The HALT mode is released by a non-maskable interrupt request signal (NMI), an unmasked maskable interrupt request signal, or any reset.

After the HALT mode has been released, the normal operation mode is restored.

(1) Releasing HALT mode by non-maskable interrupt request signal or unmasked maskable interrupt request signal

The HALT mode is released by a non-maskable interrupt request signal or an unmasked maskable interrupt request signal, regardless of the priority of the interrupt request. If the HALT mode is set in an interrupt servicing routine, however, an interrupt request that is issued later is serviced as follows.

- If an interrupt request signal with a priority lower than or same as the interrupt currently being serviced is generated, the HALT mode is released, but the newly generated interrupt request signal is not acknowledged. The interrupt request signal itself is retained.
- If an interrupt request signal with a priority higher than that of the interrupt currently being serviced is issued (including a non-maskable interrupt request signal), the HALT mode is released and that interrupt request signal is acknowledged.

Table 4-5 Operation after releasing HALT mode by interrupt request signal

Release Source	Interrupt Enabled (EI) Status	Interrupt Disabled (DI) Status
Non-maskable interrupt request signal	Execution branches to the handler address	
Unmasked maskable interrupt request signal	Execution branches to the handler address or the next instruction is executed	The next instruction is executed

(2) Releasing HALT mode by reset

The same operation as the normal reset operation is performed.

4.4 Baud Rate Generator

4.4.1 Overview

Features The baud rate generator (BRG) has the following functions:

- Baud rate generator based on a 2-bit divider and an 8-bit compare register
- 1 interrupt output per BRG macro

Instances The V850E/RG3 microcontrollers has 2 instances of the baud rate generator BRG.

Table 4-6 Instances of BRGn

BRG	
Instances	2
Names	BRG0, BRG1

Throughout this chapter, the individual instances of baud rate generators are identified by “n”, for example BRGn, or BRGnPRSM for the pre-scalar mode register of BRGn.

Register addresses All BRGn register addresses are given as address offsets to the individual base addresses <base> of each BRGn.

The <base> addresses of each BRGn are listed in the following table:

Table 4-7 Register <base> addresses of BRGn

BRGn	<BRG_base> address
BRG0	FFFF $FD10_H$
BRG1	FFFF $FD14_H$

Clock supply The clock input connections are listed in following table:

Table 4-8 Clock input of BRG

BRGn clock input	Connected clock	
	BRG0	BRG1
BRGTCKI	PCLK7 (250 KHz)	PCLK0 (32 MHz)

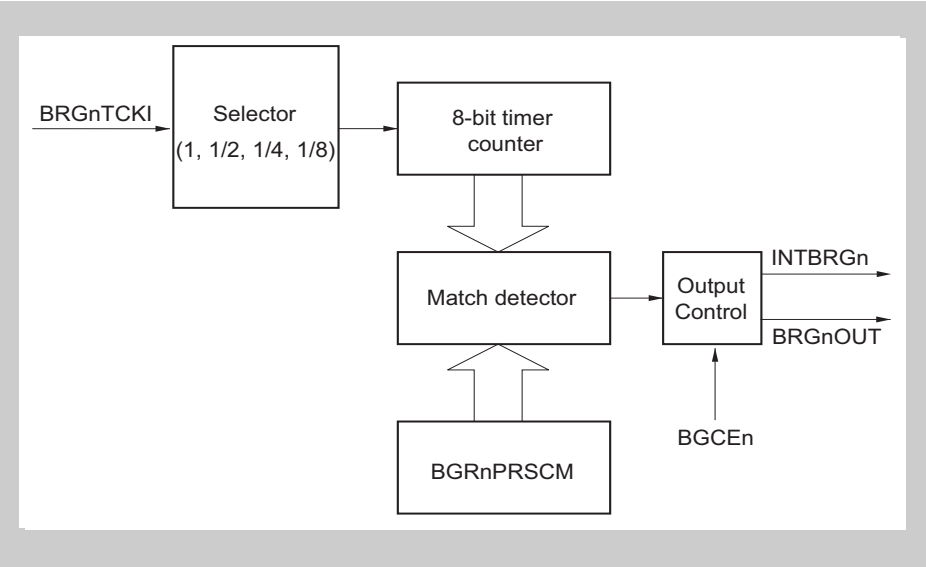
The BRG output connections are listed in following table:

Table 4-9 BRG output connections

BRGnOUT	Connected to...	
BRG0OUT	TAA1TCKI7	TAA1 input clock 7
BRG1OUT	CF0TCKI6	CSF0 input clock 6
	CF1TCKI6	CSF1 input clock 6

The following figure shows the block diagram of BRGn.

Figure 4-3 Baud Rate Generator block diagram



4.4.2 Control Registers

The Baud Rate Generators BRGn are controlled and operated by means of the following registers.

Table 4-10 BRGn registers overview

Register name	Shortcut	Address
BRGn prescaler mode register	BRGnPRSM	<BRG_base>
BRGn prescaler compare register	BRGnPRSCM	<BRG_base> + 1_H

(1) BRGnPRSM - Prescaler mode registers

The BRGnPRSM registers control generation of the baud rate signal.

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

Address BRGnPRSM : <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

BGCEn	Baud rate generator output
0	Disabled
1	Enabled

BGCSn1	BGCSn0	Input clock divider selection	Setting value k
0	0	BRGTCKI	0
0	1	BRGTCKI/2	1
1	0	BRGTCKI/4	2
1	1	BRGTCKI/8	3

Cautions

1. Do not rewrite the BRGnPRSM register during operation.
2. Set the BGCSn[1:0] bits before setting the BGCEn bit to 1.

(2) BRGnPRSCM - Prescaler compare registers

The BRGnPRSCM registers are 8-bit compare registers.

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

Address BRG0PRSCM : <BRG_base> + 1_H

BRG1PRSCM : <BRG_base> + 5_H

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

7	6	5	4	3	2	1	0
BRGnPRSCM7	BRGnPRSCM6	BRGnPRSCM5	BRGnPRSCM4	BRGnPRSCM3	BRGnPRSCM2	BRGnPRSCM1	BRGnPRSCM0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

When the counter value equals the value of BRGnPRSCM, a high-pulse will be output by INTBRGn and an interrupt can be generated.

Cautions

1. Do not rewrite the BRGnPRSCM register during operation.
2. Set the BRGnPRSCM register before setting the BRGnPRSM.BGCEn bit to 1.

4.4.3 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.

$$f_{BRGn} = \frac{BRGnTCKI}{2^{k+1} \times N}$$

Note

f_{BRGn}:	BRGn output clock
BRGnTCKI:	BRGn input clock frequency
k:	BRGnPRSM.BGCSn[1:0] register setting value ($0 \leq k \leq 3$)
N:	BRGnPRSCM.BRGnPRSCM[7:0] register value if BRGnPRSCM = 00_H, then N = 256.

Chapter 5 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 up to five 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 4 system clocks after the generation of an interrupt request.

5.1 Features

- Interrupts
  - Non-maskable interrupts: 4 sources
    (depending on whether interrupt sharing is enabled or not)
  - Maskable interrupts:
    - 69 sources on 3V devices
    - 70 sources on 5V devices
  - 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
    Interrupt sharing for maskable and non-maskable interrupts
- Exceptions
  - Software exceptions: 2 channels with each 16 sources
  - Exception traps: 1 source (illegal opcode exception)

The following table lists all the interrupts and excepts available of the V850E/RG3 devices.

Table 5-1 Interrupt/exception source list (1/3)

Type[a]	Default Priority	Name	Trigger	Generating Unit	Exception Code	Handler Address	Restored PC	Interrupt Control Register
Reset	–	RESET	External RESET, POC, LVI, WDT (reset mode), or software reset	Pin	0000_H	$0000\ 0000_H$	undef.	–
Non-maskable	–	INTDEDF[b]	Double error detection	Flash memory	0010_H	$0000\ 0010_H$	nextPC	–
		INTDEDR[b]	Double error detection	RAM				
Non-maskable	–	WDT	Watchdog Timer (interrupt mode)	WDT	0020_H	$0000\ 0020_H$	nextPC	–
Non-maskable	–	NMI	NMI Input	Pin	0030_H	$0000\ 0030_H$	nextPC	–
Software exception	–	TRAP0n (n = 0 to F_H)	TRAP instruction	–	$004n_H$ (n = 0 to F_H)	$0000\ 0040_H$	nextPC	–
	–	TRAP1n (n = 0 to F_H)	TRAP instruction	–	$005n_H$ (n = 0 to F_H)	$0000\ 0050_H$	nextPC	–
Exception trap	–	ILGOP	Illegal opcode	–	0060_H	$0000\ 0060_H$	nextPC	–
		DBTRAP	DBTRAP instruction	–				–
M	0	INTERR	Double error detection		0080_H	$0000\ 0080_H$	nextPC	ERRIC
M	1	INTAD	ADC0 end of conversion	ADC	0090_H	$0000\ 0090_H$	nextPC	AD0IC
M	2	INTLVI	Low voltage indicator	POCLVI interrupt	$00A0_H$	$0000\ 00A0_H$	nextPC	LVIIC
M	3	INTP0	External interrupt 0	Pin	$00B0_H$	$0000\ 00B0_H$	nextPC	P0IC
M	4	INTP1	External interrupt 1	Pin	$00C0_H$	$0000\ 00C0_H$	nextPC	P1IC
M	5	INTP2	External interrupt 2	Pin	$00D0_H$	$0000\ 00D0_H$	nextPC	P2IC
M	6	INTP3	External interrupt 3	Pin	$00E0_H$	$0000\ 00E0_H$	nextPC	P3IC
M	7	INTP4	External interrupt 4	Pin	$00F0_H$	$0000\ 00F0_H$	nextPC	P4IC
M	8	INTP5	External interrupt 5	Pin	0100_H	$0000\ 0100_H$	nextPC	P5IC
M	9	INTP6	External interrupt 6	Pin	0110_H	$0000\ 0110_H$	nextPC	P6IC
M	10	INTP7	External interrupt 7	Pin	0120_H	$0000\ 0120_H$	nextPC	P7IC
M	11	INTCE0C	CSIE 0 transfer end interrupt	CSIE 0	0130_H	$0000\ 0130_H$	nextPC	CE0CIC
M	12	INTCE0OF	CSIE 0 overflow interrupt	CSIE 0	0140_H	$0000\ 0140_H$	nextPC	CE0OFIC
M	13	INTCE1C	CSIE 1 transfer end interrupt	CSIE 1	0150_H	$0000\ 0150_H$	nextPC	CE1CIC
M	14	INTCE1OF	CSIE 1 overflow interrupt	CSIE 1	0160_H	$0000\ 0160_H$	nextPC	CE1OFIC
M	15	INTCF0R	CSIF 0 reception completion / reception error interrupt	CSIF 0	0170_H	$0000\ 0170_H$	nextPC	CF0RIC
M	16	INTCF0T	CSIF 0 transmit interrupt	CSIF 0	0180_H	$0000\ 0180_H$	nextPC	CF0TIC
M	17	INTCF1R	CSIF 1 reception completion / reception error interrupt	CSIF 1	0190_H	$0000\ 0190_H$	nextPC	CF1RIC
M	18	INTCF1T	CSIF 1 transmit interrupt	CSIF 1	$01A0_H$	$0000\ 01A0_H$	nextPC	CF1TIC
M	19	INTTMM0EQ0	TMM 0 compare interrupt	TMM0	$01B0_H$	$0000\ 01B0_H$	nextPC	TMM0EQ0IC
M	20	INTTAA0CC0	TAA 0 capture compare channel 0 interrupt	TAA0	$01C0_H$	$0000\ 01C0_H$	nextPC	TAA0CC0IC
M	21	INTTAA0CC1	TAA 0 capture compare channel 1 interrupt	TAA0	$01D0_H$	$0000\ 01D0_H$	nextPC	TAA0CC1IC

Table 5-1 Interrupt/exception source list (2/3)

Type[a]	Default Priority	Name	Trigger	Generating Unit	Exception Code	Handler Address	Restored PC	Interrupt Control Register
M	22	INTTAA0OV	TAA 0 overflow interrupt	TAA0	$01E0_H$	0000 $01E0_H$	nextPC	TAA0OVIC
M	23	INTTMM1EQ0	TMM 1 compare interrupt	TMM1	$01F0_H$	0000 $01F0_H$	nextPC	TMM1EQ0IC
M	24	INTTAA1CC0	TAA 1 capture compare channel 0 interrupt	TAA1	0200_H	0000 0200_H	nextPC	TAA1CC0IC
M	25	INTTAA1CC1	TAA 1capture compare channel 1 interrupt	TAA1	0210_H	0000 0210_H	nextPC	TAA1CC1IC
M	26	INTTAA1OV	TAA 1 overflow interrupt	TAA1	0220_H	0000 0220_H	nextPC	TAA1OVIC
M	27	INTTAA2CC0	TAA 2 capture compare channel 0 interrupt	TAA2	0230_H	0000 0230_H	nextPC	TAA2CC0IC
M	28	INTTAA2CC1	TAA 2 capture compare channel 1 interrupt	TAA2	0240_H	0000 0240_H	nextPC	TAA2CC1IC
M	29	INTTAA2OV	TAA 2 overflow interrupt	TAA2	0250_H	0000 0250_H	nextPC	TAA2OVIC
M	30	INTTAA3CC0	TAA 3 capture compare channel 0 interrupt	TAA3	0260_H	0000 0260_H	nextPC	TAA3CC0IC
M	31	INTTAA3CC1	TAA 3 capture compare channel 1 interrupt	TAA3	0270_H	0000 0270_H	nextPC	TAA3CC1IC
M	32	INTTAA3OV	TAA 3 overflow interrupt	TAA3	0280_H	0000 0280_H	nextPC	TAA3OVIC
M	33	INTTAB0CC0	TAB 0 capture compare channel 0 interrupt	TAB0	0290_H	0000 0290_H	nextPC	TAB0CC0IC
M	34	INTTAB0CC1	TAB 0 capture compare channel 1 interrupt	TAB0	$02A0_H$	0000 $02A0_H$	nextPC	TAB0CC1IC
M	35	INTTAB0CC2	TAB 0 capture compare channel 2 interrupt	TAB0	$02B0_H$	0000 $02B0_H$	nextPC	TAB0CC2IC
M	36	INTTAB0CC3	TAB 0 capture compare channel 3 interrupt	TAB0	$02C0_H$	0000 $02C0_H$	nextPC	TAB0CC3IC
M	37	INTTAB0OV	TAB 0 overflow interrupt	TAB0	$02D0_H$	0000 $02D0_H$	nextPC	TAB0OVIC
M	38	INTUD0T	UARTD 0 transmission interrupt	UARTD0	$02E0_H$	0000 $02E0_H$	nextPC	UD0TIC
M	39	INTUD1T	UARTD 1 transmission interrupt	UARTD1	$02F0_H$	0000 $02F0_H$	nextPC	UD1TIC
M	40	INTUD0R	UARTD 0 reception interrupt	UARTD0	0300_H	0000 0300_H	nextPC	UD0RIC
M	41	INTUD1R	UART 1 reception interrupt	UARTD1	0310_H	0000 0310_H	nextPC	UD1RIC
M	42	INTUD0S	UARTD 0 status interrupt	UARTD0	0320_H	0000 0320_H	nextPC	UD0SIC
M	43	INTUD1S	UART 1 status interrupt	UARTD1	0330_H	0000 0330_H	nextPC	UD1SIC
M	44	INTTAB1CC0	TAB 1 capture compare channel 0 interrupt	TAB1	0340_H	0000 0340_H	nextPC	TAB1CC0IC
M	45	INTTAB1CC1	TAB 1 capture compare channel 1 interrupt	TAB1	0350_H	0000 0350_H	nextPC	TAB1CC1IC
M	46	INTTAB1CC2	TAB 1 capture compare channel 2 interrupt	TAB1	0360_H	0000 0360_H	nextPC	TAB1CC2IC
M	47	INTTAB1CC3	TAB 1 capture compare channel 3 interrupt	TAB1	0370_H	0000 0370_H	nextPC	TAB1CC3IC
M	48	INTTAB1OV	TAB 1 overflow interrupt	TAB1	0380_H	0000 0380_H	nextPC	TAB1OVIC
M	49	INTBRG0	BRG 0 match detection	BRG0	0390_H	0000 0390_H	nextPC	BRG0IC
M	50	INTBRG1	BRG 1 match detection	BRG1	$03A0_H$	0000 $03A0_H$	nextPC	BRG1IC
M	51	INTC0ERR	AFCAN0 error interrupt	AFCAN0	$03B0_H$	0000 $03B0_H$	nextPC	C0ERRIC
M	52	INTC0REC	AFCAN0 receive interrupt	AFCAN0	$03C0_H$	0000 $03C0_H$	nextPC	C0RECIC

Table 5-1 Interrupt/exception source list (3/3)

Type[a]	Default Priority	Name	Trigger	Generating Unit	Exception Code	Handler Address	Restored PC	Interrupt Control Register
M	53	INTC0TRX	AFCAN0 transmit interrupt	AFCAN0	$03D0_H$	0000 $03D0_H$	nextPC	C0TRXIC
M	54	INTC0WUP	AFCAN0 wake up interrupt	AFCAN0	$03E0_H$	0000 $03E0_H$	nextPC	C0WUPIC
M	55	INTC1ERR	AFCAN1 error interrupt	AFCAN1	$03F0_H$	0000 $03F0_H$	nextPC	C1ERRIC
M	56	INTC1REC	AFCAN1 receive interrupt	AFCAN1	0400_H	0000 0400_H	nextPC	C1RECIC
M	57	INTC1TRX	AFCAN1 transmit interrupt	AFCAN1	0410_H	0000 0410_H	nextPC	C1TRXIC
M	58	INTC1WUP	AFCAN1 wake up interrupt	AFCAN1	0420_H	0000 0420_H	nextPC	C1WUPIC
M	59	INTAD0D	DMA end of transfer interrupt for ADC0	DMA	0430_H	0000 0430_H	nextPC	AD0DIC
M	60	INTCE0TXD	DMA end of transfer interrupt for CSIE0 (RAM to CSIE0)	DMA	0440_H	0000 0440_H	nextPC	CE0TDIC
M	61	INTCE0RXD	DMA end of transfer interrupt for CSIE0 (CSIE0 to RAM)	DMA	0450_H	0000 0450_H	nextPC	CE0RDIC
M	62	INTCE1TXD	DMA end of transfer interrupt for CSIE1 (RAM to CSIE0)	DMA	0460_H	0000 0460_H	nextPC	CE1TDIC
M	63	INTCE1RXD	DMA end of transfer interrupt for CSIE1 (CSIE0 to RAM)	DMA	0470_H	0000 0470_H	nextPC	CE1RDIC
M	64	INTCF0TD	DMA end of transfer interrupt for CSIF0 (RAM to CSIF0)	DMA	0480_H	0000 0480_H	nextPC	CF0TDIC
M	65	INTCF0RD	DMA end of transfer interrupt for CSIF0 (CSIF0 to RAM)	DMA	0490_H	0000 0490_H	nextPC	CF0RDIC
M	66	INTCF1TD	DMA end of transfer interrupt for CSIF1 (RAM to CSIF1)	DMA	$04A0_H$	0000 $04A0_H$	nextPC	CF1TDIC
M	67	INTCF1RD	DMA end of transfer interrupt for CSIF1 (CSIF1 to RAM)	DMA	$04B0_H$	0000 $04B0_H$	nextPC	CF1RDIC
M	115	INTFL	Flash programming completion interrupt	Flash	$07B0_H$	0000 $07B0_H$	nextPC	FLIC

a) *M* stands for Maskable interrupt

b) Depending on the setting of INTSEL.ISRD, this interrupt source generates an NMI or a maskable interrupt. For more information see *Section 5.5, Interrupt Sharing* on page 106. For details see *Section 5.5, Interrupt Sharing* on page 106.

Note 1. Default priority:
The priority order when two or more maskable interrupt requests are generated at the same time. The highest priority is 0.

2. 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).

3. nextPC:
The PC value that starts the processing following interrupt/exception processing.

4. The execution address of the illegal instruction when an illegal opcode exception occurs is calculated by (Restored PC – 4).

5.2 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 requests:

Table 5-2 Non-maskable interrupts

Interrupt Source	Generating Unit	Comment
NMI	Pin	When the valid edge specified by bits INTM0.ESN[1:0] is detected on the NMI pin, the interrupt occurs.
INTDEDF	Flash memory	These interrupt sources only generate a *non-maskable* interrupt if interrupt sharing is enabled for non-maskable interrupts. See *Section 5.5, Interrupt Sharing* on page 106.
INTDEDR	RAM	
WDTNMI	Watchdog timer	When the watchdog is in NMI mode

Note If a NMI is generated while NMI is being serviced, the service is executed as follows:

The new NMI request is held pending regardless of the value of the PSW.NP bit. The pending NMIVC request is acknowledged after servicing of the current NMI request has finished (after execution of the RETI instruction).

Caution If PSW.NP is cleared by the LDSR instruction while a non-maskable interrupt is being serviced, a following NMI interrupt cannot be acknowledged correctly.

Figure 5-1 Example of non-maskable interrupt request acknowledgement operation: multiple NMI requests generated at the same time

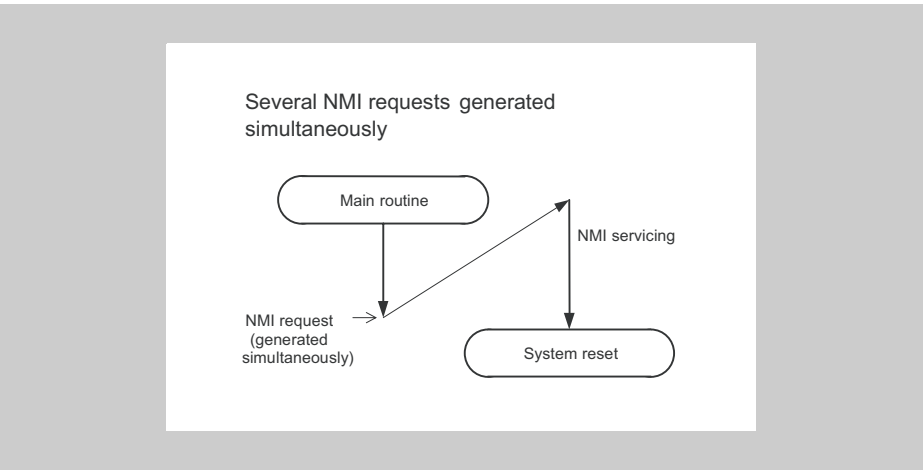
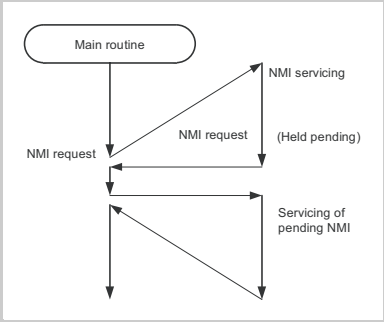


Figure 5-2 Example of non-maskable interrupt request acknowledgement operation: NMI request generated during NMI servicing



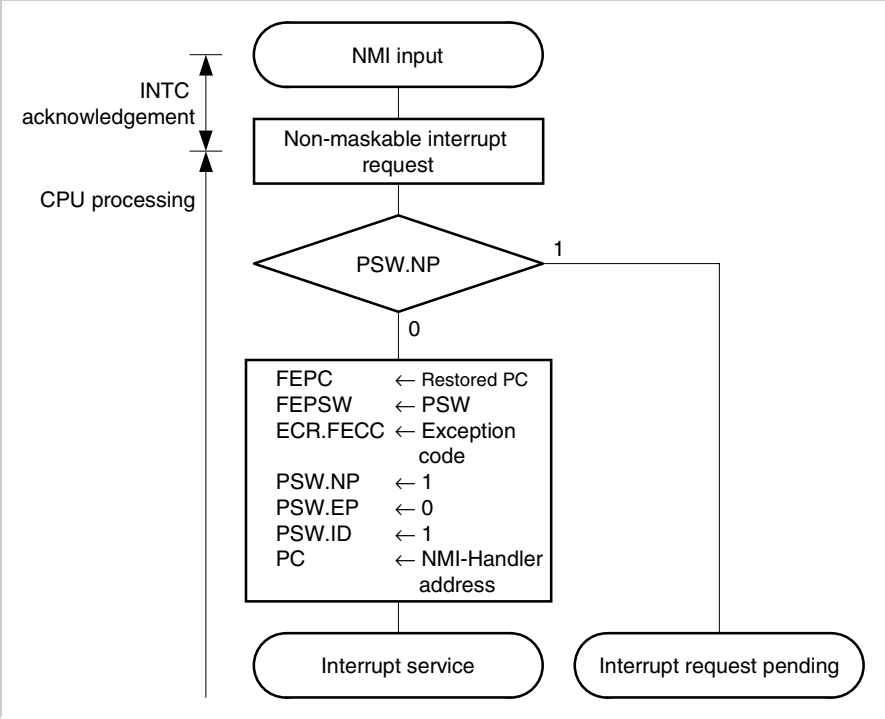
5.2.1 Operation

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.
11. Saves the current PSW to FEPSW.
12. Writes exception code 0010_H to the higher halfword (FECC) of ECR.
13. Sets the NP and ID bits of the PSW and clears the EP bit.
14. 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 5-3*.

Figure 5-3 Processing configuration of non-maskable interrupt



5.2.2 Restore

Execution is restored from the non-maskable interrupt (NMI) processing by the RETI instruction.

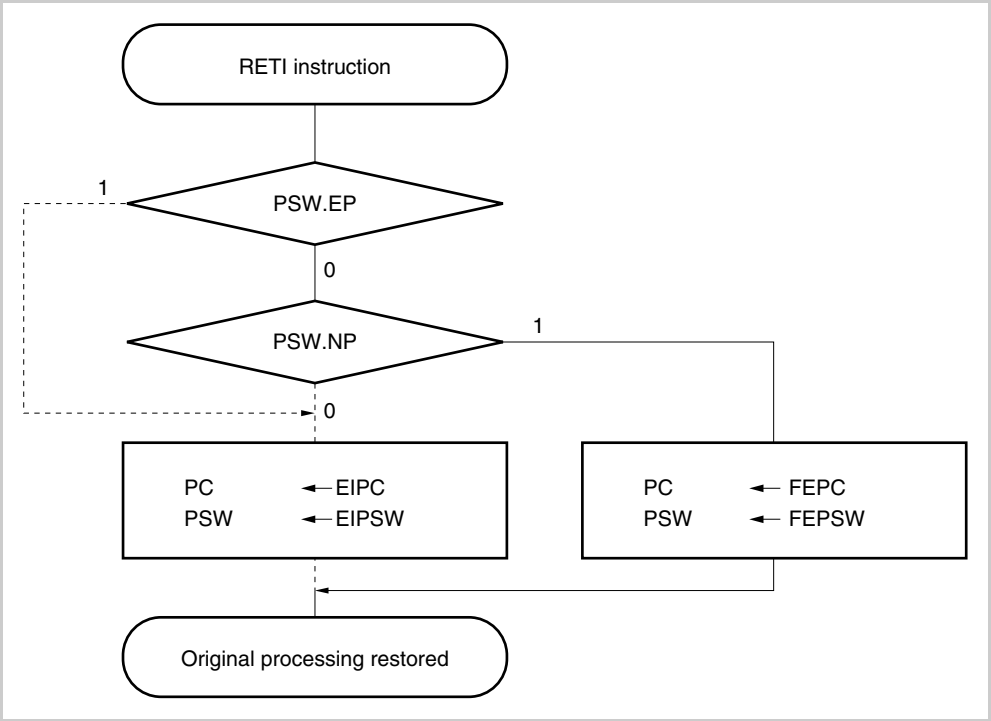
When the RETI instruction is executed, the CPU performs the following processing, and transfers control to the address of the restored PC.

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.

15. Transfers control back to the address of the restored PC and PSW.

Figure 5-4 illustrates how the RETI instruction is processed.

Figure 5-4 RETI instruction processing



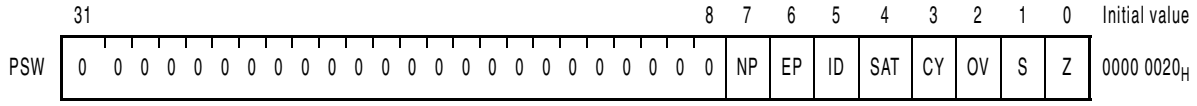
Caution When the PSW.EP bit and PSW.NP bit are changed by the LDSR instruction during non-maskable interrupt processing, in order to restore the PC and PSW correctly during recovery by the RETI instruction, it is necessary to set PSW.EP back to 0 and PSW.NP back to 1 using the LDSR instruction immediately before the RETI instruction.

Note The solid line indicates the CPU processing flow.

5.2.3 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.



NP	Function: indicate NMI interrupt processing
0	No NMI interrupt processing is in progress
1	NMI interrupt is currently being processed

5.2.4 External NMI control

The NMI can be configured to generate an NMI upon a rising, falling or both edges at the NMI pin. To enable respectively disable the NMI and to configure the edge refer to *Section 5.4.1, Edge and Level Detection Configuration* on page 103.

5.3 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.
16. Execute the DI instruction before executing the RETI instruction, then reset EIPC and EIPSW with the values saved in (1).

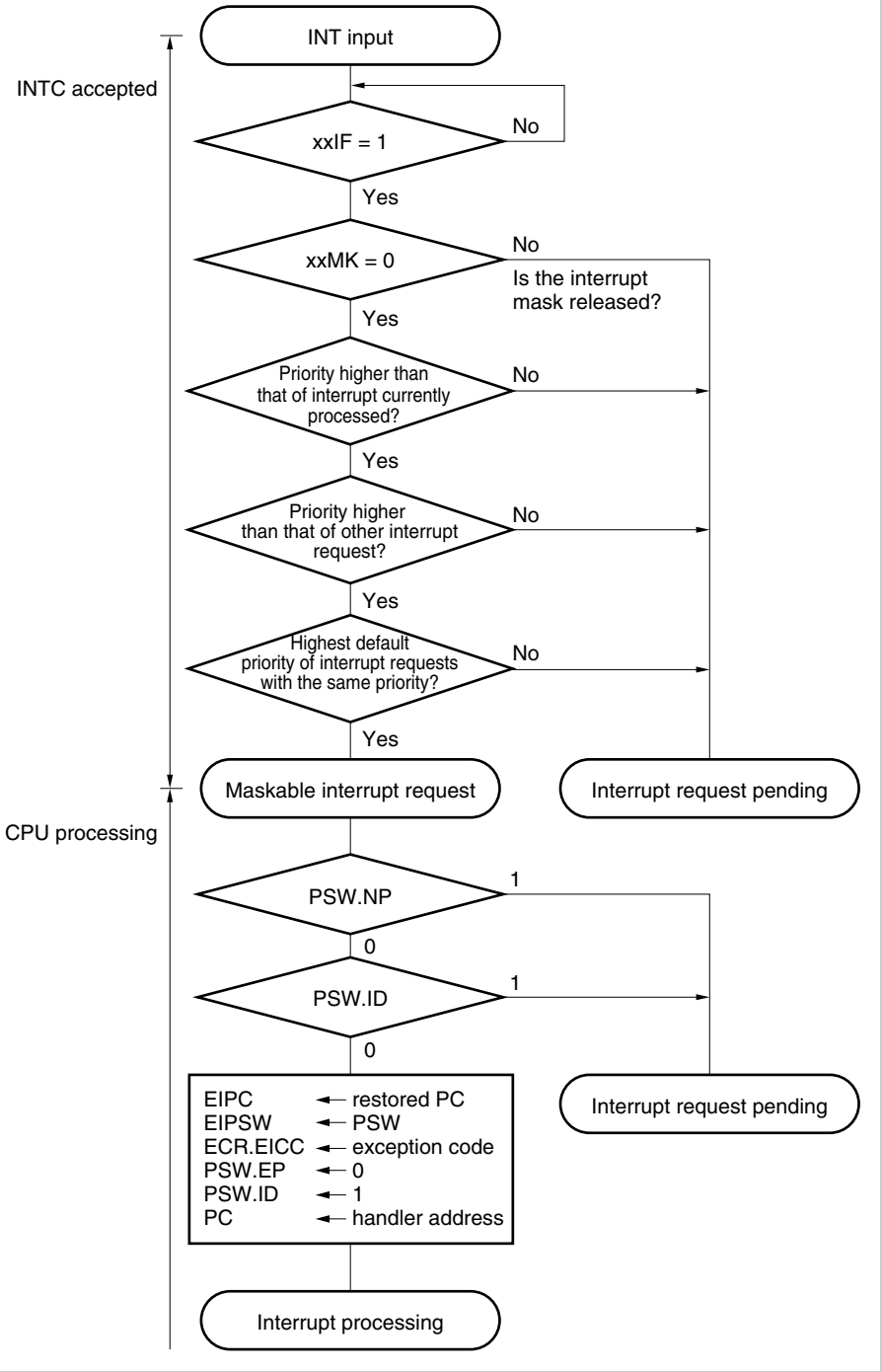
5.3.1 Operation

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.
17. Saves the current PSW to EIPSW.
18. Writes an exception code to the lower halfword of ECR (EICC).
19. Sets the ID bit of the PSW and clears the EP bit.
20. 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 5-5*.

Figure 5-5 Maskable interrupt processing



Note For the ISPR register, see *Section 5.3.6, ISPR - In-service priority register* on page 102.

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.

5.3.2 Restore

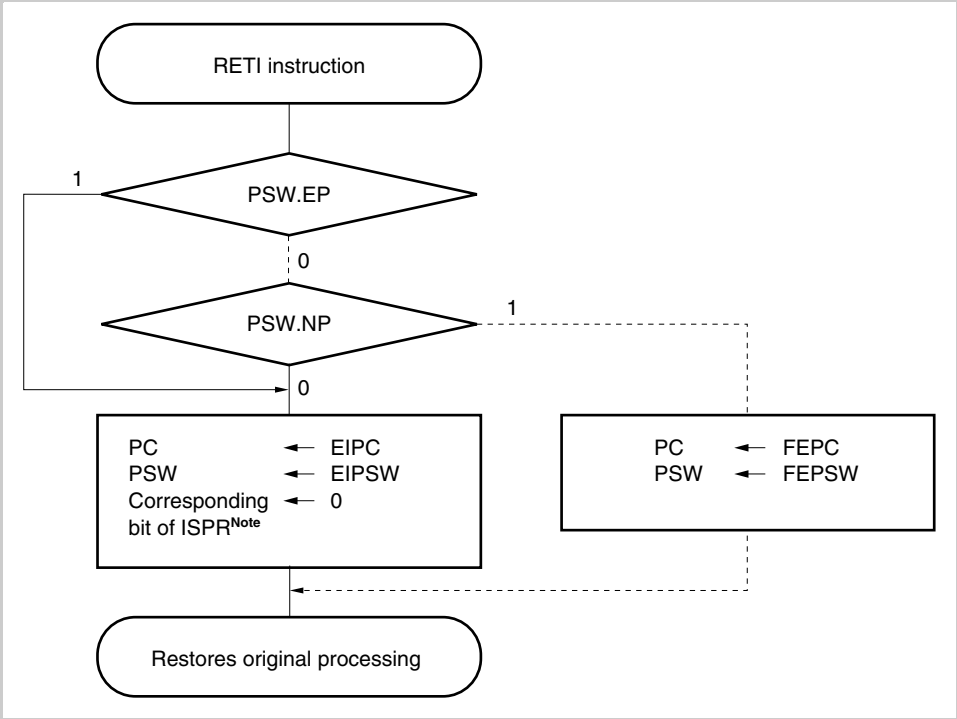
Recovery from maskable interrupt processing is carried out by the RETI instruction.

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.
21. Transfers control to the address of the restored PC and PSW.

Figure 5-6 illustrates the processing of the RETI instruction.

Figure 5-6 RETI instruction processing



Note 1. For the ISPR register, see *Section 5.3.6, ISPR - In-service priority register* on page 102.

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.

5.3.3 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.

Figure 5-7 Example of processing in which another interrupt request is issued while an interrupt is being processed (1/2)

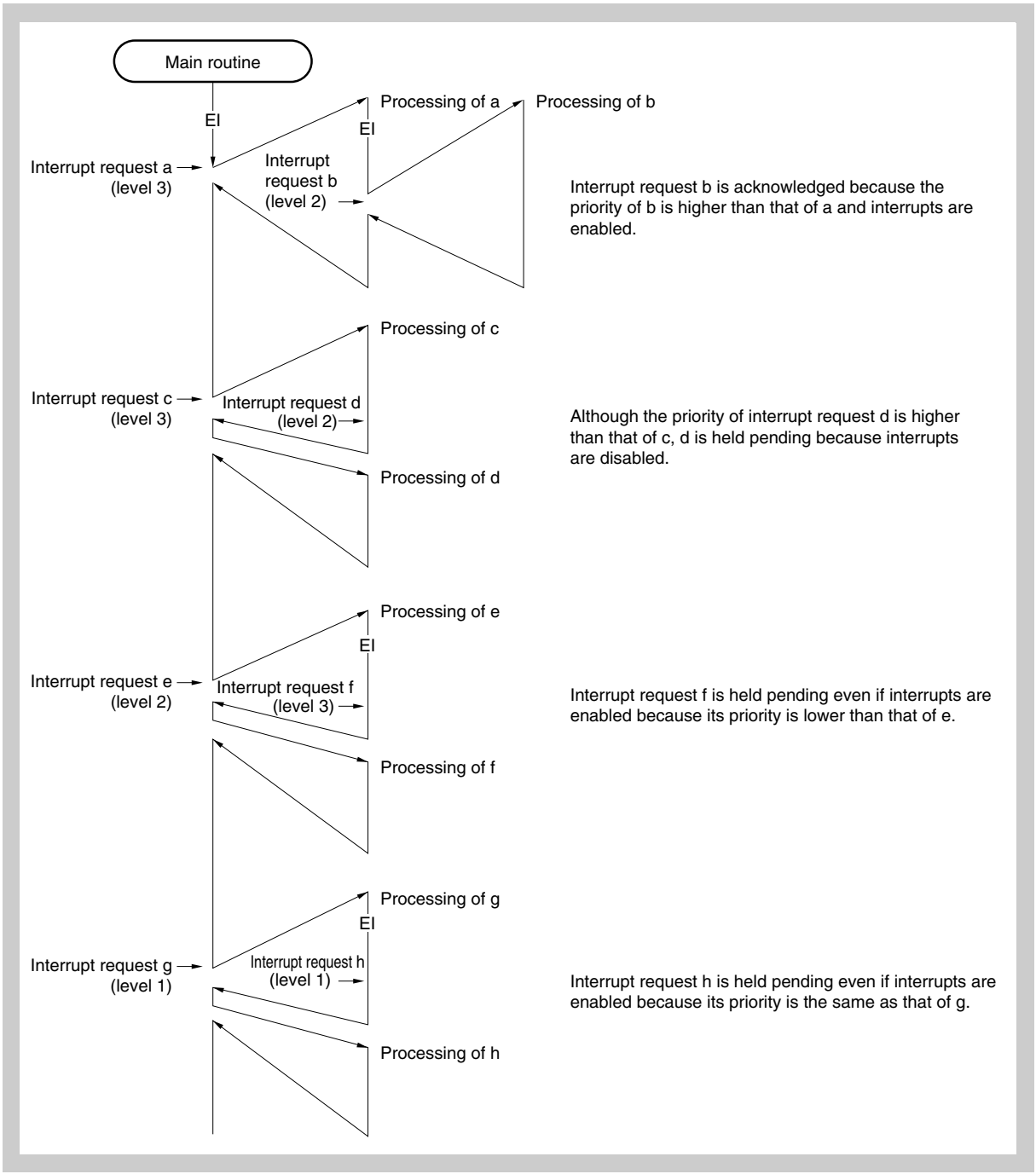
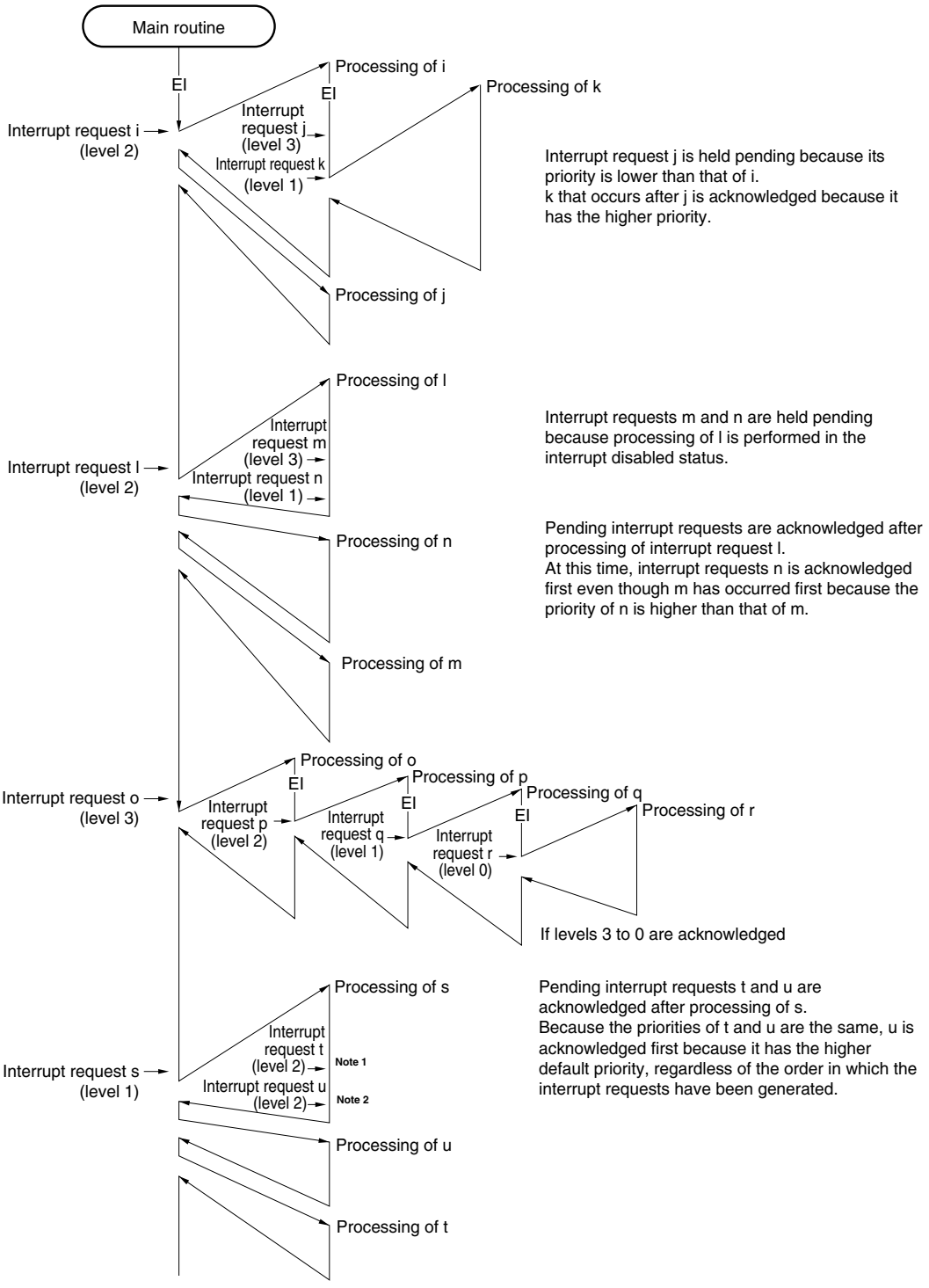


Figure 5-8 Example of processing in which another interrupt request is issued while an interrupt is being processed (2/2)

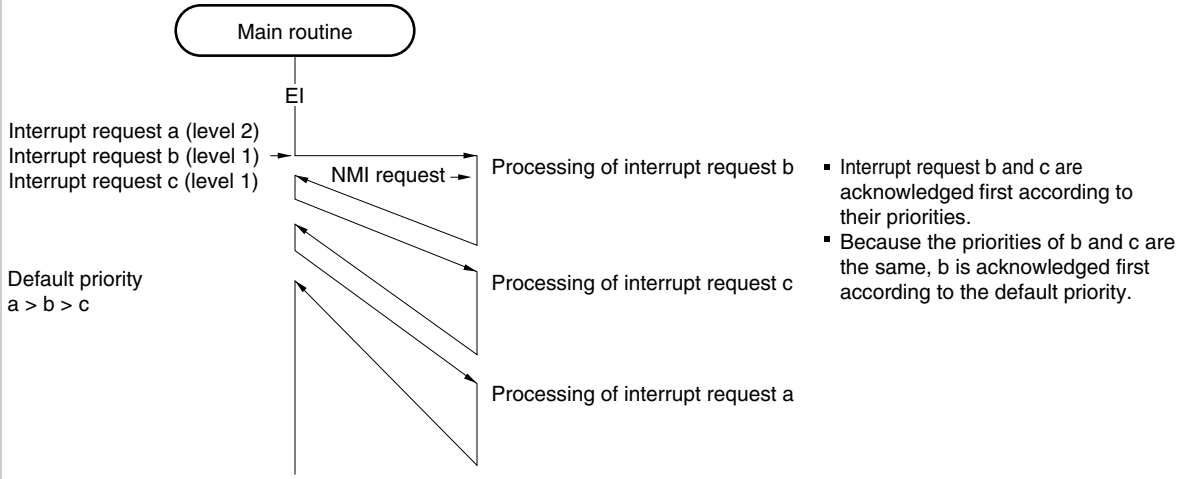


Caution The values of the EIPC and EIPSW registers must be saved before executing multiple interrupts. When returning from multiple interrupt servicing, restore the values of EIPC and EIPSW after executing the DI instruction.

Note 1. <a> to <u> in the figures 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.

Figure 5-9 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.

Note <a> to <c> in the figure are the temporary names of interrupt requests shown for the sake of explanation.

5.3.4 xxICn - 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.

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

Address FFFF F110$_H$ to FFFF F1F8$_H$

Initial value 47$_H$

7	6	5	4	3	2	1	0
xxIFn	xxMKn	0	0	0	xxPR2	xxPR1	xxPR0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

xxIFn	Interrupt request flag
0	Interrupt request not issued
1	Interrupt request issued

xxMKn	This is an interrupt mask flag.
0	Enables interrupt processing
1	Disables interrupt processing (pending)

xxPR2	xxPR1	xxPR0	Interrupt priority specification bit
0	0	0	Specifies level 0 (highest)
0	0	1	Specifies level 1
0	1	0	Specifies level 2
0	1	1	Specifies level 3
1	0	0	Specifies level 4
1	0	1	Specifies level 5
1	1	0	Specifies level 6
1	1	1	Specifies level 7 (lowest)

Note xx: identification name of each peripheral unit. For a complete list of all interrupt control registers see *Table 5-3* on page 97.

The address and bit of each interrupt control register is shown in the following table.

Table 5-3 Addresses and bits of interrupt control registers (1/3)

Address	Register	Bit								Associated interrupt
		7	6	5	4	3	2	1	0	
FFFF F110$_H$	ERRIC	ERRIF	ERRMK	0	0	0	ERRPR2	ERRPR1	ERRPR0	INTERR
FFFF F112$_H$	AD0IC	AD0IF	AD0MK	0	0	0	AD0PR2	AD0PR1	AD0PR0	INTAD0D
FFFF F114H	LVIIC	LVIIF	LVIMK	0	0	0	LVIPR2	LVIPR1	LVIPR0	INTLVI
FFFF F116$_H$	P0IC	P0IF	P0MK	0	0	0	P0PR2	P0PR1	P0PR0	INTP0
FFFF F118$_H$	P1IC	P1IF	P1MK	0	0	0	P1PR2	P1PR1	P1PR0	INTP1
FFFF F11A$_H$	P2IC	P2IF	P2MK	0	0	0	P2PR2	P2PR1	P2PR0	INTP2

Table 5-3 Addresses and bits of interrupt control registers (2/3)

Address	Register	Bit								Associated interrupt
		7	6	5	4	3	2	1	0	
FFFF F11C$_H$	P3IC	P3IF	P3MK	0	0	0	P3PR2	P3PR1	P3PR0	INTP3
FFFF F11E$_H$	P4IC	P4IF	P4MK	0	0	0	P4PR2	P4PR1	P4PR0	INTP4
FFFF F120$_H$	P5IC	P5IF	P5MK	0	0	0	P5PR2	P5PR1	P5PR0	INTP5
FFFF F122$_H$	P6IC	P6IF	P6MK	0	0	0	P6PR2	P6PR1	P6PR0	INTP6
FFFF F124$_H$	P7IC	P7IF	P7MK	0	0	0	P7PR2	P7PR1	P7PR0	INTP7
FFFF F126$_H$	CE0CIC	CE0CIF	CE0CMK	0	0	0	CE0CPR2	CE0CPR1	CE0CPR0	INTCE0C
FFFF F128$_H$	CE0OFIC	CE0OFIF	CE0OFMK	0	0	0	CE0OFPR2	CE0OFPR1	CE0OFPR0	INTCE0OF
FFFF F12A$_H$	CE1CIC	CE1CIF	CE1CMK	0	0	0	CE1CPR2	CE1CPR1	CE1CPR0	INTCE1C
FFFF F12C$_H$	CE1OFIC	CE1OFIF	CE1OFMK	0	0	0	CE1OFPR2	CE1OFPR1	CE1OFPR0	INTCE1OF
FFFF F12E$_H$	CF0RIC	CF0RIF	CF0RMK	0	0	0	CF0RPR2	CF0RPR1	CF0RPR0	INTCF0R
FFFF F130$_H$	CF0TIC	CF0TIF	CF0TMK	0	0	0	CF0TPR2	CF0TPR1	CF0TPR0	INTCF0T
FFFF F132$_H$	CF1RIC	CF1RIF	CF1RMK	0	0	0	CF1RPR2	CF1RPR1	CF1RPR0	INTCF1R
FFFF F134$_H$	CF1TIC	CF1TIF	CF1TMK	0	0	0	CF1TPR2	CF1TPR1	CF1TPR0	INTCF1T
FFFF F136$_H$	TMM0EQ0IC	TMM0EQ0IF	TMM0EQ0MK	0	0	0	TMM0EQ0PR2	TMM0EQ0PR1	TMM0EQ0PR0	INTTMM0EQ0
FFFF F138	TAA0CC0IC	TAA0CC0IF	TAA0CC0MK	0	0	0	TAA0CC0PR2	TAA0CC0PR1	TAA0CC0PR0	INTTAA0CC0
FFFF F13A$_H$	TAA0CC1IC	TAA0CC1IF	TAA0CC1MK	0	0	0	TAA0CC1PR2	TAA0CC1PR1	TAA0CC1PR0	INTTAA0CC1
FFFF F13C$_H$	TAA0OVIC	TAA0OVIF	TAA0OVMK	0	0	0	TAA0OVPR2	TAA0OVPR1	TAA0OVPR0	INTTAA0OV
FFFF F13E$_H$	TMM1EQ0IC	TMM1EQ0IF	TMM1EQ0MK	0	0	0	TMM1EQ0PR2	TMM1EQ0PR1	TMM1EQ0PR0	INTTMM1EQ0
FFFF F140$_H$	TAA1CC0IC	TAA1CC0IF	TAA1CC0MK	0	0	0	TAA1CC0PR2	TAA1CC0PR1	TAA1CC0PR0	INTTAA1CC0
FFFF F142$_H$	TAA1CC1IC	TAA1CC1IF	TAA1CC1MK	0	0	0	TAA1CC1PR2	TAA1CC1PR1	TAA1CC1PR0	INTTAA1CC1
FFFF F144$_H$	TAA1OVIC	TAA1OVIF	TAA1OVMK	0	0	0	TAA1OVPR2	TAA1OVPR1	TAA1OVPR0	INTTAA1OV
FFFF F146$_H$	TAA2CC0IC	TAA2CC0IF	TAA2CC0MK	0	0	0	TAA2CC0PR2	TAA2CC0PR1	TAA2CC0PR0	INTTAA2CC0
FFFF F148$_H$	TAA2CC1IC	TAA2CC1IF	TAA2CC1MK	0	0	0	TAA2CC1PR2	TAA2CC1PR1	TAA2CC1PR0	INTTAA2CC1
FFFF F14A$_H$	TAA2OVIC	TAA2OVIF	TAA2OVMK	0	0	0	TAA2OVPR2	TAA2OVPR1	TAA2OVPR0	INTTAA2OV
FFFF F14C$_H$	TAA3CC0IC	TAA3CC0IF	TAA3CC0MK	0	0	0	TAA3CC0PR2	TAA3CC0PR1	TAA3CC0PR0	INTTAA3CC0
FFFF F14E$_H$	TAA3CC1IC	TAA3CC1IF	TAA3CC1MK	0	0	0	TAA3CC1PR2	TAA3CC1PR1	TAA3CC1PR0	INTTAA3CC1
FFFF F150$_H$	TAA3OVIC	TAA3OVIF	TAA3OVMK	0	0	0	TAA3OVPR2	TAA3OVPR1	TAA3OVPR0	INTTAA3OV
FFFF F152$_H$	TAB0CC0IC	TAB0CC0IF	TAB0CC0MK	0	0	0	TAB0CC0PR2	TAB0CC0PR1	TAB0CC0PR0	INTTAB0CC0
FFFF F154$_H$	TAB0CC1IC	TAB0CC1IF	TAB0CC1MK	0	0	0	TAB0CC1PR2	TAB0CC1PR1	TAB0CC1PR0	INTTAB0CC1
FFFF F156$_H$	TAB0CC2IC	TAB0CC2IF	TAB0CC2MK	0	0	0	TAB0CC2PR2	TAB0CC2PR1	TAB0CC2PR0	INTTAB0CC2
FFFF F158$_H$	TAB0CC3IC	TAB0CC3IF	TAB0CC3MK	0	0	0	TAB0CC3PR2	TAB0CC3PR1	TAB0CC3PR0	INTTAB0CC3
FFFF F15A$_H$	TAB0OVIC	TAB0OVIF	TAB0OVMK	0	0	0	TAB0OVPR2	TAB0OVPR1	TAB0OVPR0	INTTAB0OV
FFFF F15C$_H$	UD0TIC	UD0TIF	UD0TMK	0	0	0	UD0TPR2	UD0TPR1	UD0TPR0	INTUD0T
FFFF F15E$_H$	UD1TIC	UD1TIF	UD1TMK	0	0	0	UD1TPR2	UD1TPR1	UD1TPR0	INTUD1T
FFFF F160$_H$	UD0RIC	UD0RIF	UD0RMK	0	0	0	UD0RPR2	UD0RPR1	UD0RPR0	INTUD0R
FFFF F162$_H$	UD1RIC	UD1RIF	UD1RMK	0	0	0	UD1RPR2	UD1RPR1	UD1RPR0	INTUD1R
FFFF F164$_H$	UD0SIC	UD0SIF	UD0SMK	0	0	0	UD0SPR2	UD0SPR1	UD0SPR0	INTUD0S
FFFF F166$_H$	UD1SIC	UD1SIF	UD1SMK	0	0	0	UD1SPR2	UD1SPR1	UD1SPR0	INTUD1S
FFFF F168$_H$	TAB1CC0IC	TAB1CC0IF	TAB1CC0MK	0	0	0	TAB1CC0PR2	TAB1CC0PR1	TAB1CC0PR0	INTTAB1CC0
FFFF F16A$_H$	TAB1CC1IC	TAB1CC1IF	TAB1CC1MK	0	0	0	TAB1CC1PR2	TAB1CC1PR1	TAB1CC1PR0	INTTAB1CC1
FFFF F16C$_H$	TAB1CC2IC	TAB1CC2IF	TAB1CC2MK	0	0	0	TAB1CC2PR2	TAB1CC2PR1	TAB1CC2PR0	INTTAB1CC2
FFFF F16E$_H$	TAB1CC3IC	TAB1CC3IF	TAB1CC3MK	0	0	0	TAB1CC3PR2	TAB1CC3PR1	TAB1CC3PR0	INTTAB1CC3
FFFF F170$_H$	TAB1OVIC	TAB1OVIF	TAB1OVMK	0	0	0	TAB1OVPR2	TAB1OVPR1	TAB1OVPR0	INTTAB1OV

Table 5-3 Addresses and bits of interrupt control registers (3/3)

Address	Register	Bit								Associated interrupt
		7	6	5	4	3	2	1	0	
FFFF F172$_H$	BRG0IC	BRG0IF	BRG0MK	0	0	0	BRG0PR2	BRG0PR1	BRG0PR0	INTBRG0
FFFF F174$_H$	BRG1IC	BRG1IF	BRG1MK	0	0	0	BRG1PR2	BRG1PR1	BRG1PR0	INTBRG1
FFFF F176$_H$	C0ERRIC	C0ERRIF	C0ERRMK	0	0	0	C0ERRPR2	C0ERRPR1	C0ERRPR0	INTC0ERR
FFFF F178$_H$	C0RECIC	C0RECIF	C0RECMK	0	0	0	C0RECPR2	C0RECPR1	C0RECPR0	INTC0REC
FFFF F17A$_H$	C0TRXIC	C0TRXIF	C0TRXMK	0	0	0	C0TRXPR2	C0TRXPR1	C0TRXPR0	INTC0TRX
FFFF F17C$_H$	C0WUPIC	C0WUPIF	C0WUPMK	0	0	0	C0WUPPR2	C0WUPPR1	C0WUPPR0	INTC0WUP
FFFF F17E$_H$	C1ERRIC	C1ERRIF	C1ERRMK	0	0	0	C1ERRPR2	C1ERRPR1	C1ERRPR0	INTC1ERR
FFFF F180$_H$	C1RECIC	C1RECIF	C1RECMK	0	0	0	C1RECPR2	C1RECPR1	C1RECPR0	INTC1REC
FFFF F182$_H$	C1TRXIC	C1TRXIF	C1TRXMK	0	0	0	C1TRXPR2	C1TRXPR1	C1TRXPR0	INTC1TRX
FFFF F184$_H$	C1WUPIC	C1WUPIF	C1WUPMK	0	0	0	C1WUPPR2	C1WUPPR1	C1WUPPR0	INTC1WUP
FFFF F186$_H$	AD0DIC	AD0DIF	AD0DMK	0	0	0	AD0DPR2	AD0DPR1	AD0DPR0	INTAD0D
FFFF F188$_H$	CE0TDIC	CE0TDIF	CE0TDMK	0	0	0	CE0TDPR2	CE0TDPR1	CE0TDPR0	INTCE0TXD
FFFF F18A$_H$	CE0RDIC	CE0RDIF	CE0RDMK	0	0	0	CE0RDPR2	CE0RDPR1	CE0RDPR0	INTCE0RXD
FFFF F18C$_H$	CE1TDIC	CE1TDIF	CE1TDMK	0	0	0	CE1TDPR2	CE1TDPR1	CE1TDPR0	INTCE1TXD
FFFF F18E$_H$	CE1RDIC	CE1RDIF	CE1RDMK	0	0	0	CE1RDPR2	CE1RDPR1	CE1RDPR0	INTCE1RXD
FFFF F190$_H$	CF0TDIC	CF0TDIF	CF0TDMK	0	0	0	CF0TDPR2	CF0TDPR1	CF0TDPR0	INTCF0TD
FFFF F192$_H$	CF0RDIC	CF0RDIF	CF0RDMK	0	0	0	CF0RDPR2	CF0RDPR1	CF0RDPR0	INTCF0RD
FFFF F194$_H$	CF1TDIC	CF1TDIF	CF1TDMK	0	0	0	CF1TDPR2	CF1TDPR1	CF1TDPR0	INTCF1TD
FFFF F196$_H$	CF1RDIC	CF1RDIF	CF1RDMK	0	0	0	CF1RDPR2	CF1RDPR1	CF1RDPR0	INTCF1RD
FFFF F1F6$_H$	FLIC	FLIF	FLMK	0	0	0	FLPR2	FLPR1	FLPR0	INTFL

5.3.5 IMR0 to IMR4 - Interrupt mask registers

These registers set the interrupt mask state for the maskable interrupts.

The IMKn bit of the IMRm (m = 0 to 4) registers is equivalent to the xxMK bit of the xxIC register.

IMRm registers can be read/written in 16- bit, 8-bit, and 1-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).

Caution Mask bits without function, indicated with “0”, must not be altered. Make sure to set them “0” when writing to the register.

Address FFFF F100$_H$

Initial value FFFF$_H$

	15	14	13	12	11	10	9	8
IMR0	CF0RMK	CE1OFMK	CE1CMK	CE0OFMK	CE0CMK	P7MK	P6MK	P5MK
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
	7	6	5	4	3	2	1	0
	P4MK	P3MK	P2MK	P1MK	P0MK	0	AD0MK	ERRMK
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Address FFFF F102$_H$

Initial value FFFF$_H$

	15	14	13	12	11	10	9	8
IMR1	TAA3CC1MK	TAA3CC0MK	TAA2OVMK	TAA2CC1MK	TAA2CC0MK	TAA1OVMK	TAA1CC1MK	TAA1CC0MK
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
	7	6	5	4	3	2	1	0
	TMM1EQ0MK	TAA0OVMK	TAA0CC1MK	TAA0CC0MK	TMM0EQ0MK	CF1TMK	CF1RMK	CF0TMK
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Address FFFF F104$_H$

Initial value FFFF$_H$

	15	14	13	12	11	10	9	8
IMR2	TAB1CC3MK	TAB1CC2MK	TAB1CC1MK	TAB1CC0MK	UD1SMK	UD0SMK	UD1RMK	UD0RMK
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
	7	6	5	4	3	2	1	0
	UD1TMK	UD0TMK	TAB0OVMK	TAB0CC3MK	TAB0CC2MK	TAB0CC1MK	TAB0CC0MK	TAA3OVMK
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Address FFFF F106$_H$

Initial value FFFF$_H$

	15	14	13	12	11	10	9	8
IMR3	CE1RDMK	CE1TDMK	CE0RDMK	CE0TDMK	AD0DMK	C1WUPMK	C1TRXMK	C1RECMK
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
	7	6	5	4	3	2	1	0
	C1ERRMK	C0WUPMK	C0TRXMK	C0RECMK	C0ERRMK	BRG1MK	BRG0MK	TAB1OVMK
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Address FFFF F108$_H$

Initial value FFFF$_H$

	15	14	13	12	11	10	9	8
IMR4	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
	1	1	1	1	CF1RDMK	CF1TDMK	CF0RDMK	CF0TDMK
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Caution Be sure to write 1 to bits 15-8 and 7-2.

xxMKn	Interrupt mask flag
0	Interrupt servicing enabled
1	Interrupt servicing disabled (pending)

5.3.6 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.

Address FFFF F1FA$_H$

Initial value 00$_H$

	7	6	5	4	3	2	1	0
ISPR	ISPR7	ISPR6	ISPR5	ISPR4	ISPR3	ISPR2	ISPR1	ISPR0
	R	R	R	R	R	R	R	R

Table 5-4 ISPR register contents

ISPRn	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)

5.3.7 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$_H$

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
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

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.

5.4 External maskable interrupts

This microcontroller provides 8 maskable external interrupts INTP0 to INTP7 with the following features:

- Analog input filter
- Interrupt detection selectable for each interrupt input:
  - Rising edge
  - Falling edge
  - Both edges: rising and falling edge
- 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 *Section 5.4.1, Edge and Level Detection Configuration* on page 103.

5.4.1 Edge and Level Detection Configuration

The registers for the configuration of interrupts specify, whether an interrupt is detected at the falling, rising, or on both falling and rising edge of the signal. For every maskable or non-maskable interrupt two registers define the type of edge:

- INTFn - enables falling edge
- INTRn - enables rising edge

If both bits are 0, no interrupt will be generated.

(1) INTF0 - External interrupt falling edge specification register 0

The 8-bit INTF0 register specifies if an interrupt is detected on falling edges.

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

Address FFFF FC00$_H$

Initial Value 00$_H$. This register is initialized by $\overline{\text{RESET}}$.

7	6	5	4	3	2	1	0
INTP7F	INTP6F	INTP5F	INTP4F	INTP3F	INTP2F	INTP1F	INTP0F
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 5-5 INTF0 register contents

Bit position	Bit name	Function
7	INTP7F	INTP7 falling edge selection
6	INTP6F	INTP6 falling edge selection
5	INTP5F	INTP5 falling edge selection
4	INTP4F	INTP4 falling edge selection
3	INTP3F	INTP3 falling edge selection
2	INTP2F	INTP2 falling edge selection
1	INTP1F	INTP1 falling edge selection
0	INTP0F	INTP0 falling edge selection

(2) INTF1 - External interrupt falling edge specification register 1

The 8-bit INTR1 register specifies if the NMI is detected on falling edges and enables the NMI.

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

Address FFFF FC02$_H$

Initial Value 00$_H$. This register is initialized by $\overline{\text{RESET}}$.

7	6	5	4	3	2	1	0
NMIF	0	0	0	0	0	0	NMIEN
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 5-6 INTF1 register contents

Bit position	Bit name	Function
7	NMIF	NMI falling edge selection
0	NMIEN	NMI enable bit

(3) INTR0 - External interrupt rising edge specification register 0

The 8-bit INTR0 register specifies if an interrupt is detected on rising edges.

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

Address FFFF FC04$_H$

Initial Value 00$_H$. This register is initialized by $\overline{\text{RESET}}$.

7	6	5	4	3	2	1	0
INTP7R	INTP6R	INTP5R	INTP4R	INTP3R	INTP2R	INTP1R	INTP0R
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 5-7 INTR0 register contents

Bit position	Bit name	Function
7	INTP7R	INTP7 rising edge selection
6	INTP6R	INTP6 rising edge selection
5	INTP5R	INTP5 rising edge selection
4	INTP4R	INTP4 rising edge selection
3	INTP3R	INTP3 rising edge selection
2	INTP2R	INTP2 rising edge selection
1	INTP1R	INTP1 rising edge selection
0	INTP0R	INTP0 rising edge selection

(4) INTR1 - External interrupt rising edge specification register 1

The 8-bit INTR1 register specifies if the NMI is detected on rising edges.

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

Address FFFF FC06$_H$

Initial Value 00$_H$. This register is initialized by $\overline{\text{RESET}}$.

7	6	5	4	3	2	1	0
NMIR	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 5-8 INTR1 register contents

Bit position	Bit name	Function
7	NMIR	NMI rising edge selection

5.5 Interrupt Sharing

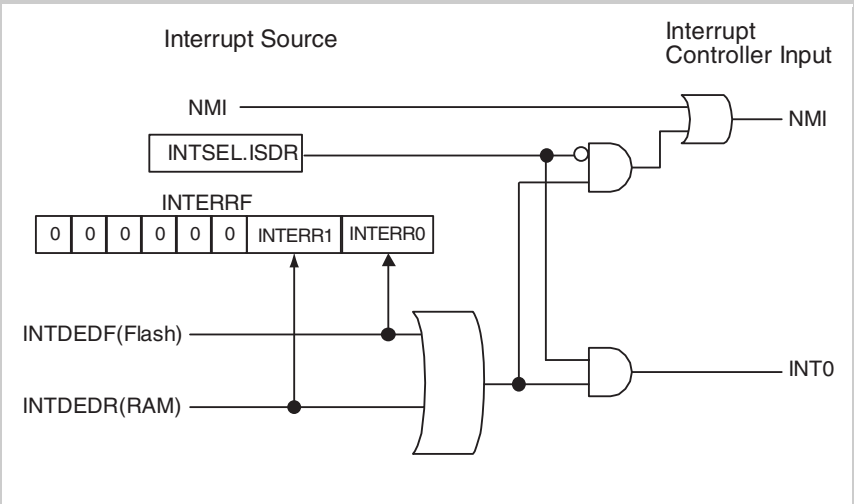
Interrupt sharing is supported for both non-maskable and maskable interrupts. The INTSEL register configures interrupt sharing.

Non-maskable interrupts

Interrupt sharing for non-maskable interrupts can be enabled by setting bit INTSEL.ISR = 1. If interrupt sharing is enabled, the interrupt signals INTDEDF and INTDEDR generate an NMI instead of a maskable interrupt.

Figure 5-10 shows the logic operations of interrupt sharing for NMI.

Figure 5-10 Interrupt sharing for NMI



Maskable interrupts

Some interrupt sources share the same maskable interrupt. There are the following options for interrupt sharing:

- Any of the interrupt sources can generate the corresponding interrupt request. All these interrupt sources are combined with logical “or”.
- An interrupt selection bit defines which interrupt source can generate the corresponding interrupt request.

For more details on the interrupt sources see the interrupt/exception source list in *Table 5-1* on page 82.

5.5.1 INTSEL - Interrupt share enable register

The 8-bit INTSEL register specifies

- which interrupt source generates a maskable interrupt with priority 75,
- whether or not the interrupt source for the NMI is shared.

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

Address FFFF FB00$_H$

Initial Value 00$_H$. This register is initialized by $\overline{\text{RESET}}$.

7	6	5	4	3	2	1	0
ISDR	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 5-9 INTSEL register contents

Bit position	Bit name	Function
7	ISDR	Enables/disables sharing of NMI 0: NMI sharing enabled 1: NMI sharing disabled

Caution If necessary, the bit INTSEL.ISDR must be set to “1” after reset release and must not be changed afterwards.

5.5.2 INTERRF Interrupt source flag register

The 8-bit INTERRF register identifies the interrupt source of NMI and INT0 (default priority 0) interrupts.

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

Address FFFF FB02$_H$

Initial Value 00$_H$. This register is initialized by $\overline{\text{RESET}}$.

7	6	5	4	3	2	1	0
0	0	0	0	0	0	INTERR1	INTERR0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 5-10 INTERRF register contents

Bit position	Bit name	Function
1	INTERR1	Displays the status of interrupt INTDEDR (RAM) 0: Interrupt not generated 1: Interrupt generated
0	INTERR0	Displays the status of interrupt INTDEDF (Flash) 0: Interrupt not generated 1: Interrupt generated

5.6 Software Exception

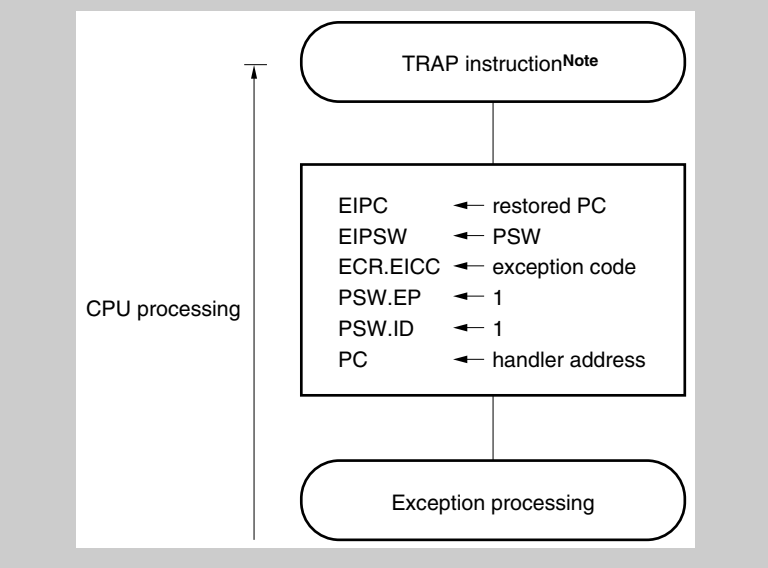
5.6.1 Operation

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.
22. Saves the current PSW to EIPSW.
23. Writes an exception code to the lower 16 bits (EICC) of ECR (interrupt source).
24. Sets the EP and ID bits of the PSW.
25. Sets the handler address (0000 0040_H or 0000 0050_H) corresponding to the software exception to the PC, and transfers control.

Figure 5-11 illustrates the processing of a software exception.

Figure 5-11 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 0000 0040_H, and if the vector is 10_H to $1F_H$, it becomes 0000 0050_H.

5.6.2 Restore

Recovery from software exception processing is carried out by the RETI instruction.

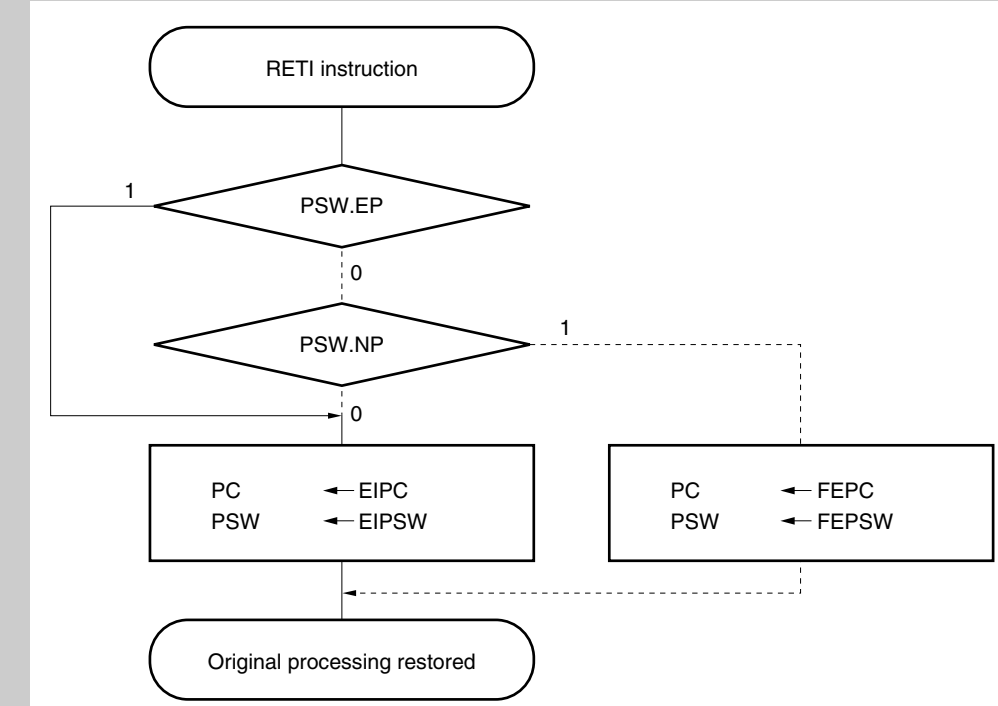
By executing the RETI instruction, the CPU carries out the following processing and shifts control to the restored PC's address.

1. Loads the restored PC and PSW from EIPC and EIPSW because the EP bit of the PSW is 1.

26. Transfers control to the address of the restored PC and PSW.

Figure 5-12 illustrates the processing of the RETI instruction.

Figure 5-12 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.

5.6.3 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.

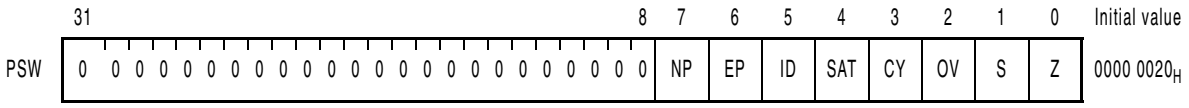


Table 5-11 INTSEL register contents

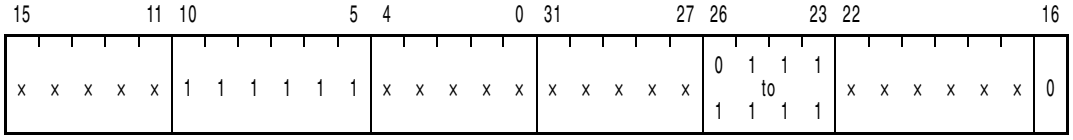
Bit position	Bit name	Function
6	EP	Shows that exception processing is in progress. 0: Exception processing not in progress. 1: Exception processing in progress.

5.7 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.

5.7.1 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.



Note ×: Arbitrary

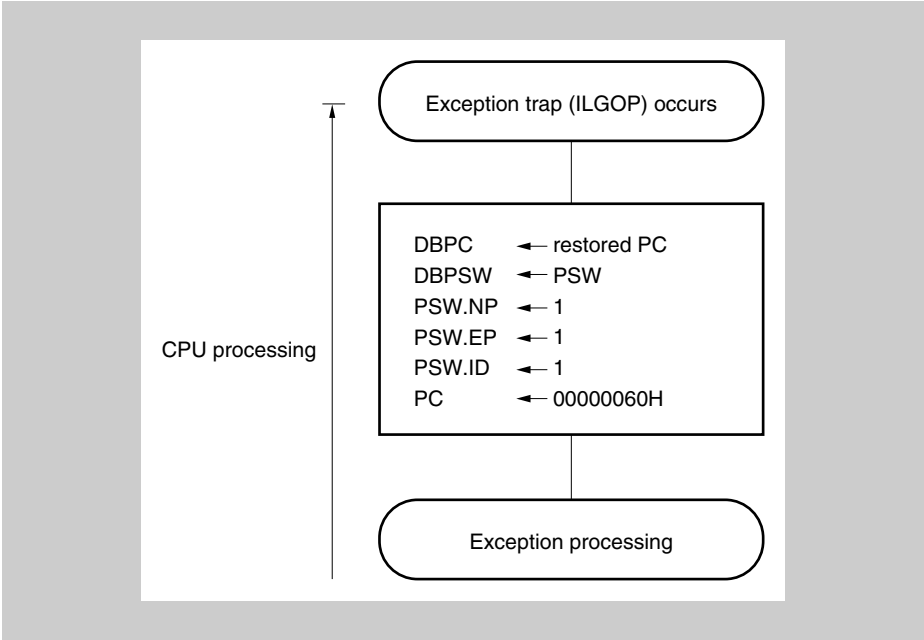
(1) Operation

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.
27. Saves the current PSW to DBPSW.
28. Sets the NP, EP, and ID bits of the PSW.
29. Sets the handler address (0000 0060_H) corresponding to the exception trap to the PC, and transfers control.

Figure 5-13 illustrates the processing of the exception trap.

Figure 5-13 Exception trap processing



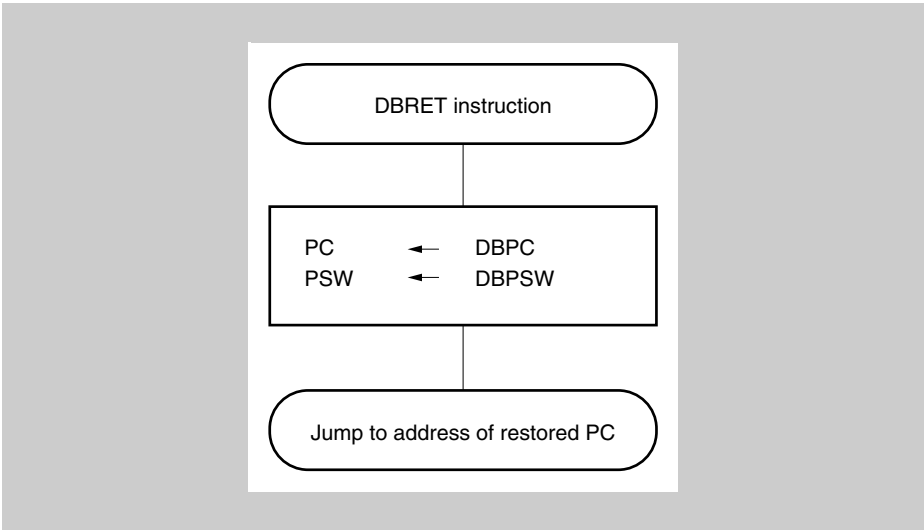
(2) Restore

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.
30. Transfers control to the address indicated by the restored PC and PSW.

Figure 5-14 illustrates the restore processing from an exception trap.

Figure 5-14 Restore processing from exception trap



5.7.2 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.

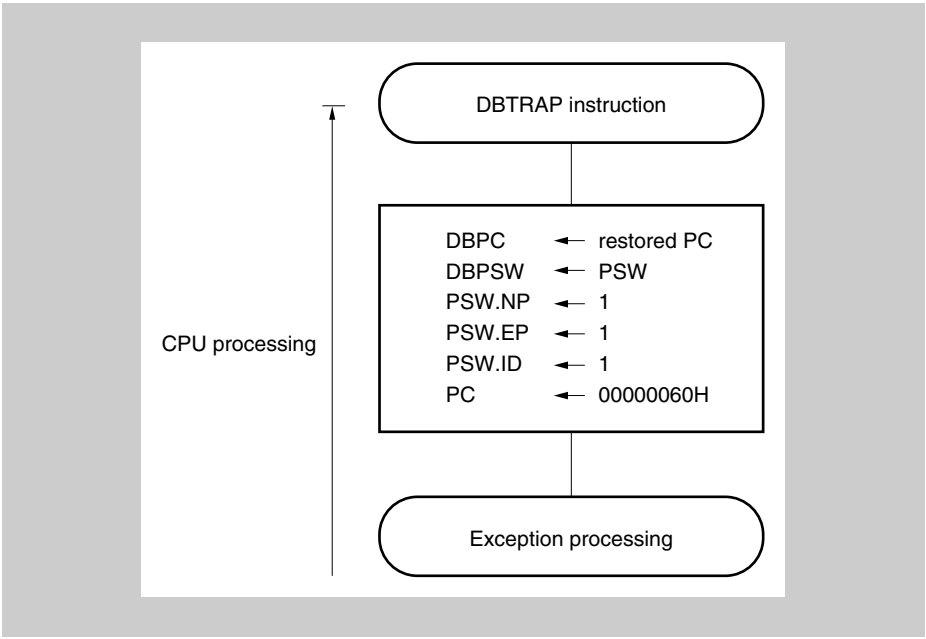
(1) Operation

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.
31. Saves the current PSW to DBPSW.
32. Sets the NP, EP and ID bits of the PSW.
33. Sets the handler address (0000 0060_H) corresponding to the debug trap to the PC and transfers control.

Figure 5-15 illustrates the processing of the debug trap.

Figure 5-15 Debug trap processing



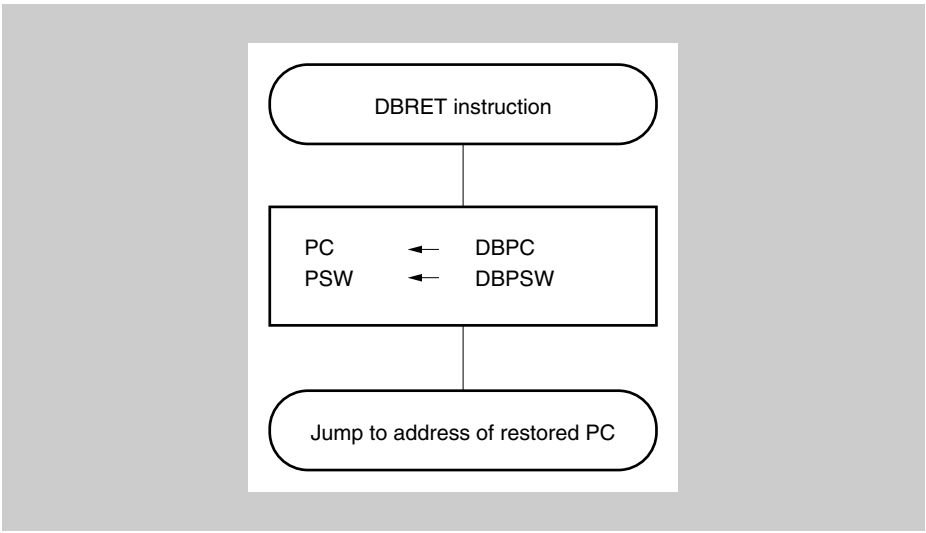
(2) Restore

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.
34. Transfers control to the address indicated by the restored PC and PSW.

Figure 5-16 illustrates the restore processing from a debug trap.

Figure 5-16 Restore processing from debug trap



5.8 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.

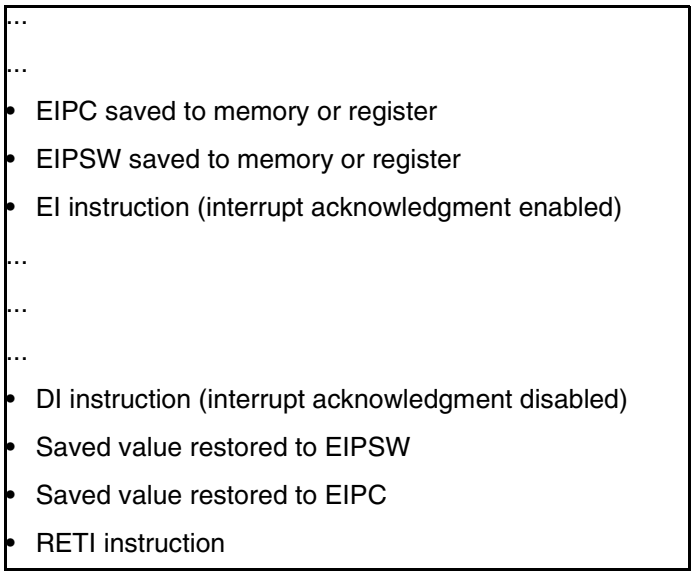
Maskable interrupt multiple processing control is executed when an interrupt has an enable status (ID = 0). Thus, if multiple interrupts are executed, it is necessary to have an interrupt enable status (ID = 0) even for an interrupt processing routine.

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.

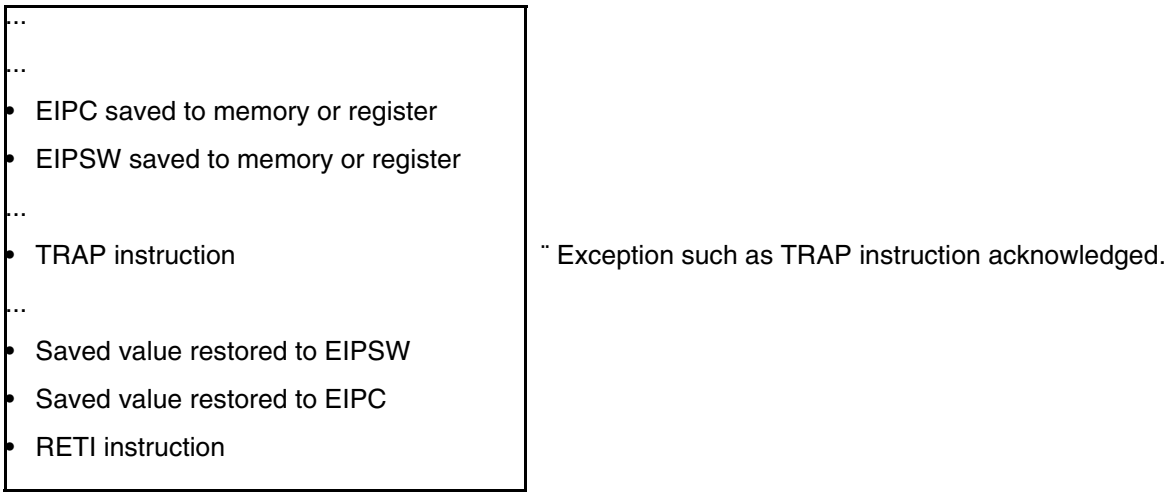
(1) Acknowledgment of maskable interrupts in service program

Service program of maskable interrupt or exception



(2) Generation of exception in service program

Service program of maskable interrupt or exception



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 (PlCn), 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.

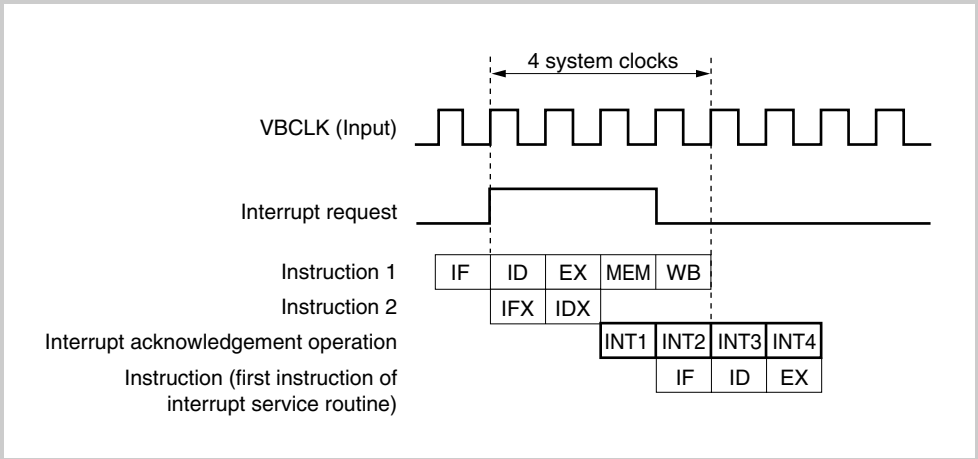
5.9 Interrupt Response Time

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 4 clocks. To input interrupt requests continuously, leave a space of at least 4 clocks between interrupt request inputs.

- During software or hardware STOP mode
- When there are two or more successive interrupt request non-sampling instructions (see *Section 5.10, Periods in Which Interrupts Are Not Acknowledged* on page 116).
- When the interrupt control register is accessed

Figure 5-17 Pipeline operation at interrupt request acknowledgment (outline)



Note INT1 to INT4:Interrupt acknowledgement processing
IFx:Invalid instruction fetch
IDx:Invalid instruction decode

Table 5-12 Interrupt response time

Interrupt response time (internal system clocks)			Condition
	Internal interrupt	External interrupt	
Minimum	4	4 + digital noise filter delay	The following cases are exceptions: • In IDLE/software STOP mode • Two or more interrupt request non-sample instructions are executed • Access to interrupt control register
Maximum	8	8 + digital noise filter delay	

5.10 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 interrupt control register (PICn), in-service priority register (ISPR), and command register (PRCMD).

Chapter 6 Flash Memory

The V850E/RG3 microcontroller is equipped with following internal flash memory:

- code flash: 512 KB
- data flash: 32 KB

The code flash memory is attached to the dedicated fetch bus interface of the V850 CPU core. It is used for non-volatile storage of program code and constant data.

The data flash memory is accessible via the memory interface bus. It holds nonvolatile user's data, which is subject to be altered during normal program operation.

The flash memory can be written in different ways:

- by a flash programmer equipped with a suitable adapter (off-board write)
- by a flash programmer connected to the target system where the device is mounted (on-board write)
- by the application software on the device (self-programming)

6.1 Overview

Code Flash:

- Flash size:
  - 256 KB (µPD70F3464, µPD70F3470
  - 384 KB (µPD70F3465, µPD70F3471)
  - 512 KB (µPD70F3466, µPD70F3472)
- ECC: 7 bits per 32-bit word
- Block size: 4 KB

Data Flash:

- Flash size: 32 KB
- ECC: 6 bits per 32-bit word
- Block size: 2 KB

6.2 Code Flash Memory

6.2.1 Code flash memory features

- Internal code flash memory: up to 512 KB
- Operation speed: up to 80 MHz by 2-way interleaved access
  - 4-byte/1 CPU clock cycle access for consecutive instruction fetches
  - 4-byte/5 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 two serial interfaces
- On-board and off-board programming
- Flash memory programming by self-programming
- 7-bit ECC (error correction code) for every 4 bytes
  - 1 bit error: automatic correction
  - 2 bit errors: detection
  - Specific interrupt generation in case of bit errors

Note Before starting any operation (e.g. write, erase, verify) in Code or Data Flash, it is mandatory to ensure that no other operation beside read in either Code or Data Flash is in progress

6.2.2 Code flash memory mapping

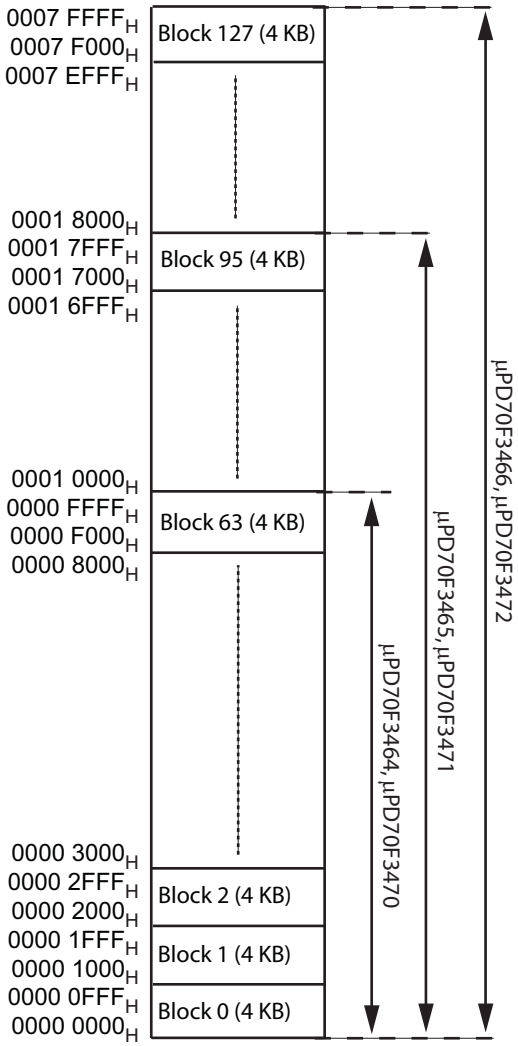
The V850E/RG3 microcontroller's internal code flash memory area is divided into blocks of 4 KB and can be programmed and erased in block units. All or some of the blocks can also be erased at once.

The following figures list the block structures and address assignments of the code flash memory.

Additional information:

- Boot swap cluster size
  Fixed 32 KB boot cluster for secure self-programming, refer to *Section 6.5.3, Secure self-programming (boot cluster swapping)* on page 136.
- Interleave
  Interleave configuration of the flash memory blocks.
- CPU branch latency
  Number of additional CPU clock cycles during instruction fetches of non-linear code.

Figure 6-1 Code flash memory configuration



Note
- Code flash size: 512 KB
- Boot swap cluster sizes: 32 KB
- Interleave: 2-way
- CPU branch latency: 4 cycles

6.2.3 Code flash memory functional outline

Serial programming

The internal flash memory of the V850E/RG3 microcontroller can be rewritten by using the rewrite function of a dedicated flash programmer, whether the device has already been mounted on the target system or not (on-board vs. off-board programming).

Since there is no functional difference between on-board and off-board programming by an external flash programmer, both will be referred to as “serial programming” in contrast to ”self-programming”.

Self-programming

The self-programming library, which facilitates rewriting of the flash memory by the user program, is ideal for program updates after production and shipment, since no additional programming equipment is required. During self-programming some software services as well as interrupt servicing can still be in operation, e.g to sustain communication with other devices.

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 *Section 6.4.4, Flash memory programming control* on page 129 for details on how to enter normal operation or serial flash programming mode.

Extra area

The flash memory contains an extra area, used to store the settings of security and protection functions and other flash relevant information.
The extra area is not mapped into the CPU’s address space, thus is not directly accessible by the user’s program. The extra area’s settings can only be read and modified by an external programmer or by the self-programming library.

Boot swap

A boot swap function makes safe re-programming of the flash memory possible and is used to maintain an operable software version, even if re-programming fails for any reason, e.g. in a power fail situation.

For further information concerning boot swapping refer to *Section 6.5.3, Secure self-programming (boot cluster swapping)* on page 136.

Protection

A set of protection flags can be configured during flash memory programming to prohibit access to the flash memory in different ways, including read-out, rewrite and erase protections. By these means the code flash memory can be protected against read-out and rewrite of the flash memory content by unauthorized persons.

Table 6-1 Flash memory write methods

Environment	Interface	Outline	Operation Mode
Serial programming	Serial I/F (UART, CSI)	Flash memory programming is done by an external flash programmer. The device may be mounted on the target system (on-board) or unmounted (off-board) by using a suitable programming adapter board. In either case the communication between the device and the flash progammer is using a serial interface. For details refer to *Section 6.4, Flash Programming with Flash Programmer* on page 125.	Flash memory programming mode
Self-programming	Self-programming library	Flash memory can be rewritten by executing a user program that has been written to the flash memory in advance by means of off-board/on-board programming. The self-programming library provides all necessary functions to be called by the user’s software. For details refer to *Section 6.5, Code Flash Self-Programming* on page 134.	Normal operation mode

Table 6-2 on page 121 summarizes the functions used to modify flash memory content.

Table 6-2 Basic functions for flash memory modifications

Function	Functional outline	Support (√: Supported, ×: Not supported)	
		Serial programming	Self-programming
Block erasure	The contents of specified memory blocks are erased.	√	√
Multiple block erasure	The contents of the specified successive multiple blocks are erased.	√	√
Chip erasure	The contents of the entire memory area is erased all at once. The extra area - except the boot block cluster protection flag - is also erased. **Caution:** The chip erase function erases also the data flash memory.	√	x [a]
Write	Writing to specified addresses, and a verify check to see if write level is secured are performed.	√	√
Verify	Data read from the flash memory is compared with data transferred from the flash programmer.	√	x [b]
Checksum	Microcontroller internally calculated checksum over the entire flash memory content is compared with the checksum calculated by the external programmer	√	x
Blank check	The erasure status of the entire memory is checked.	√	√
Protection settings	Following functions can be prohibited: • chip erase • block erase • write • read • rewriting of the boot block cluster	√	√ [c]

a) In self-programming mode all blocks can be specified to be erased at once by block erasure. Note that the extra area is not erased in this case.

b) Can be carried out by the user's program.

c) Except protection against rewriting of the boot block cluster all other protections have no effect in self-programming mode.
Protection settings can be activated in self-programming mode. Previously activated protection settings can not be deactivated.

The following table lists the available flash memory protection functions.

Table 6-3 Protection functions

Function	Functional outline	Applicable (√: applies, ×: doesn't apply)	
		Serial programming	Self-programming
Chip erase command prohibit	Erasure of the entire flash (including the extra area[a] and the data flash) or single blocks impossible.	√	×
Block erase command prohibit	Erasure of single blocks impossible.	√	×
Program command prohibit	Erasure and rewrite of single blocks impossible.	√	×
Read command prohibit	Read-out of any flash content impossible.	√	×
Rewriting boot area prohibit	Erasure (by block or chip erase) or writing of the boot block cluster impossible.	√	√

a) The boot block cluster protection flag is not erased.

6.2.4 Code flash memory erasure and rewrite

Erasure According to its block structure the flash memory can be erased in two different modes.

- All-blocks batch erasure
  All flash memory blocks are erased at the same time.
- Block erasure
  Each 4 KB flash memory block is erased separately.
  In self-programming mode any number of contiguous flash memory blocks can be erased all together.

Rewrite In self-programming and serial programming mode, it is possible to rewrite the flash memory in smaller units than one block. Once a complete block has been erased, it can be rewritten in units of 4 bytes. Each unit can be rewritten only once after erasure of the complete block.

6.2.5 Code flash memory error detection

The code flash memory control circuit is equipped with an error detection function. A 7-bit ECC (Error Correction Coding) code is dedicated to each 32-bit word in the flash memory.

The ECC can detect and correct single bit errors and detect double bit errors. In the latter case the double error detection interrupt INTDEDF is generated.

INTDEDF can generate two kinds of interrupts:

- an NMI via the NMI sharing function
- a maskable interrupt INTERR (exception code 0080_H)

Besides generating a non-maskable or maskable interrupt, the address of the erroneous data and the erroneous data byte of the 32-bit data are saved in registers.

Note As error detection is also performed on instructions, which are prefetched by the CPU's execution pipeline, an error may be detected and reported, even if the instruction is not attempted to be executed because of a program branch.

One register is provided for the error correction function.

(1) ROMEAD - Flash ROM ECC error register

The 32-bit ROMEAD register holds the address of the flash memory location, where an error was detected first.

Access This register can be read in 32-bit units.

Address FFFF FCEA$_H$

Initial Value 0000 0000$_H$

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

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

Table 6-4 ROMEAD register contents

Bit Name	Function
ROMEADDR	20-bit address the first flash ROM error detected.

The address of the first error detected is stored in ROMEAD. At the same time the interrupt flag INTERRF.INTERR0 is set to 1.

This register is not overwritten by a new error detection until

- the INTERRF.INTERR0 is cleared to 0. Thus no new error is signaled by interrupt INTDEDF until the INTERRF.INTERR0 is cleared to 0.
- ROMEAD is cleared to 0000 0000$_H$ by any reset.

Afterwards a new ECC detection address can be stored.

(2) Flash memory error detection check

For checking the functionality of the flash memory error detection, a dedicated address in the flash memory needs to be reserved for generating a mismatch between the flash memory data word and its associated ECC.

This flash memory address needs to be programmed twice:

2. The first data written to the address generates the correct ECC
3. The second data written to the same address must differ to the first data. Consider that during re-programming only "1" bits can be changed to "0", but not vice versa. Further the correct ECC of the second data must differ to the correct ECC of the first data.

Afterwards any read of the data (either as an instruction fetch or as a read of a constant) should generate a maskable or non-maskable flash memory error interrupt INTDEDF.

6.3 Data Flash Memory

The V850E/RG3 microcontroller contains a 32 KB data flash in addition to the code flash. The data flash is on-chip connected to the external memory bus.

6.3.1 Data flash memory features

The data flash has the following features:

- 32 KB of data flash memory in 2 KB blocks
- Write access in 32-bit steps
- Erase in 2 KB blocks
- Write, erase operations to the data flash while user's code can be executed from code flash

Note Before starting any operation (e.g. write, erase, verify) in Code or Data Flash, it is mandatory to ensure that no other operation beside read in either Code or Data Flash is in progress

6.3.2 Data flash memory map

The data flash is connected to the memory bus, but is built into the V850 microcontroller.

The data flash is mapped to the $\overline{CS7}$ area to address range FE0 0000_H to FE0 $7FFF_H$.

The memory interface has to be set up as follows:

Table 6-5 BCU/MEMC register settings for data flash access

Control bit	Required setting	Comment
CSC1.CS7[3:0]	0010_B	• area 0 assigned to $\overline{CS1}$ • default, don't change
BSC.BSC7[1:0]	10_B	• 32 bit bus width • default, don't change
BEC.BE70	0	• little endian • default, don't change
BCT1.ME7	1	• enable $\overline{CS7}$ • no default, must be changed
AWC.AHW7 AWC.ASW7	00_B	• no address setup/hold waits • default, don't change
DWC1.DWC7[2:0]	001_B	• 1 data wait state • no default, must be changed
BCC.BC71	0	• no idle states • no default, must be changed

6.3.3 Data flash reading

The data flash can be read via the external memory bus.

6.3.4 Data flash writing

The data flash can be written by using the data flash library or serial programming with an external flash programmer tool.

Programming during normal operation is achieved by using the data flash access layer software library. The data flash access layer is described in a separate User's Manual.

Note The chip (all blocks) erase command of an external programmer also erases the data flash.

6.4 Flash Programming with Flash Programmer

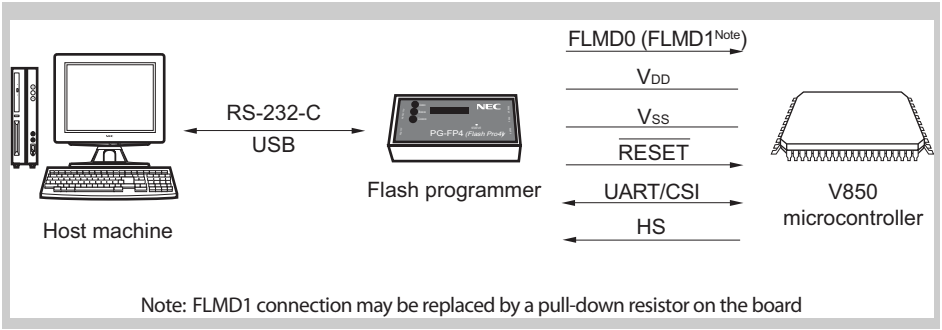
A dedicated flash programmer can be used for serial programming of the flash memory.

- On-board programming
  The contents of the flash memory can be rewritten with the device mounted on the target system. Mount a connector that connects the flash programmer to the target system.
- Off-board programming
  The flash memory of the device can be written before the device is mounted on the target system by using a dedicated programming adapter.

6.4.1 Programming environment

The necessary environment to write a program to the flash memory of the V850 microcontroller is shown below.

Figure 6-2 Environment to program flash memory using serial programming



A host PC is required for controlling the flash programmer.

The following V850 microcontroller serial interfaces can be used as the interface between the flash programmer and the device:

- asynchroneous serial interface UART
- clocked serial interface CSI

If a CSI interface is used with handshake, the flash programmer's HS signal is connected to a certain V850 port, in the following generally named as HSPORT. The port used as HSPORT for this product is given in *Table 6-7*.

Off-board flash memory programming off-board requires a dedicated programming adapter.

In this chapter the terms UART and CSI may be used generically for the dedicated interface types and channels the V850 microcontroller provides. UART and CSI signal names are used accordingly.

6.4.2 Communication mode

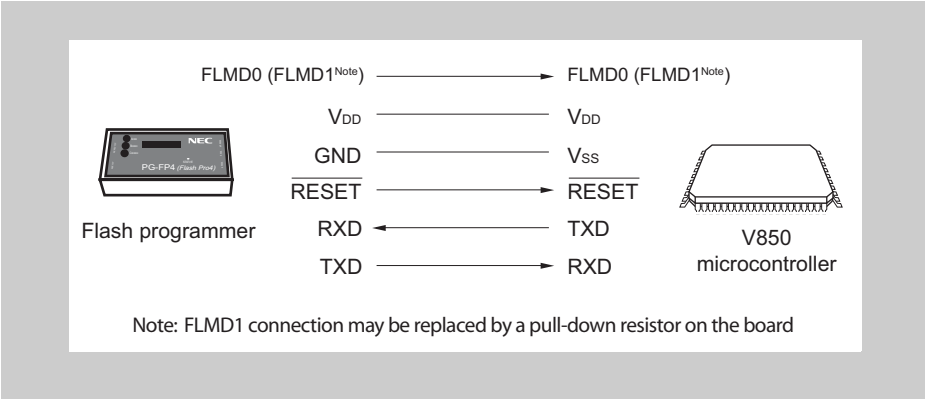
The communication between the flash programmer and the V850 microcontroller utilizes the asynchronous serial interface UART or the synchronous serial interface CSI.

If using the synchronous serial interface CSI, programming without handshake and with handshake modes are supported. In the latter mode, the port pin HSPORT is used for the programmer's handshake signal HS.

(1) UART

The external flash programmer offers various choices of available baud rates.

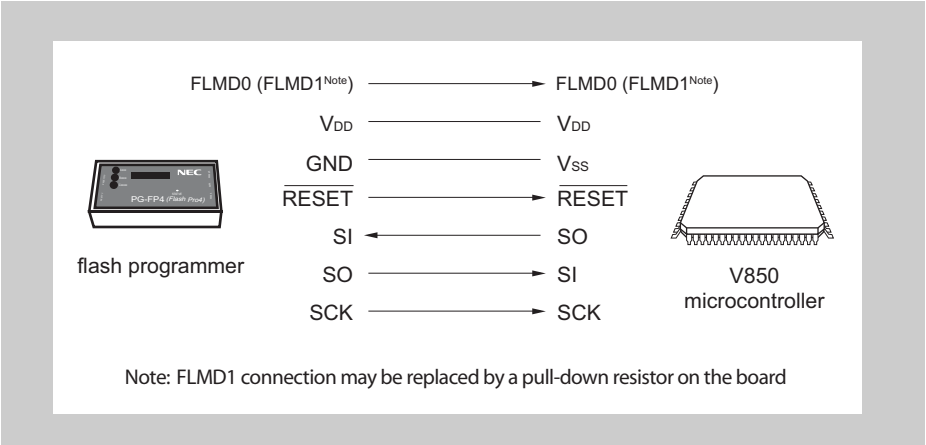
Figure 6-3 Communication with flash programmer via UART



(2) CSI without handshake

The external flash programmer offers various choices of available clock rates.

Figure 6-4 Communication with flash programmer via CSI without handshake

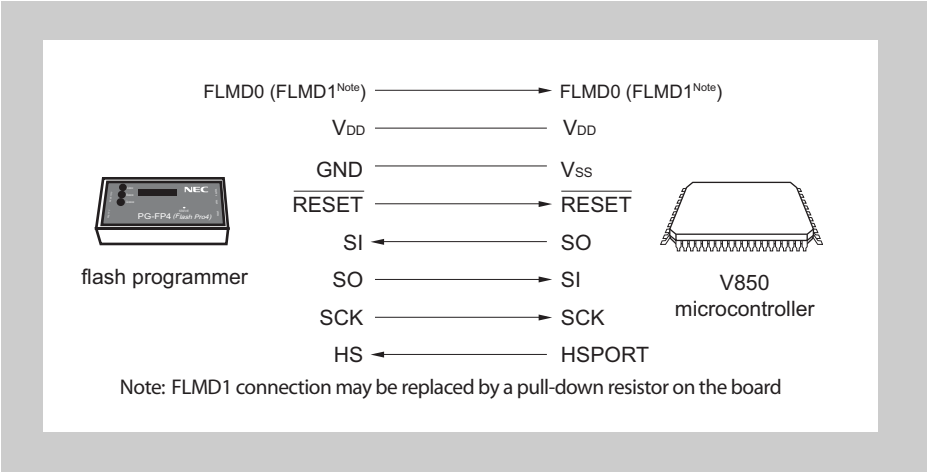


The flash programmer outputs a transfer clock and the V850 microcontroller operates as a slave.

(3) CSI with handshake (CSI + HS)

The external flash programmer offers various choices of available clock rates.

Figure 6-5 Communication with flash programmer via CSI with handshake



The flash programmer outputs a transfer clock and the V850 microcontroller operates as a slave.

6.4.3 Pin connections for flash programmer PG-FP4

A connector must be mounted on the target system to connect the flash programmer for on-board programming. In addition, functions to switch between the normal operation mode and flash memory programming mode and to control the V850 microcontroller's reset pin must be provided on the board.

When the flash memory programming mode is enabled, all the pins not used for flash memory programming are in the same status as immediately after reset.

If the PG-FP4 is used as the flash programmer, it generates the signals listed in *Table 6-6* for the V850 microcontroller. For details, refer to the PG-FP4 User's Manual (U15260E).

Table 6-6 Signals generated by flash programmer PG-FP4

PG-FP4			Controller	Connection		
Signal name	**I/O**	**Pin function**	**Pin name**	**UART**	**CSI**	**CSI + HS**
FLMD0	Output	Write enable/disable	FLMD0	√	√	√
FLMD1	Output	Write enable/disable	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	×	×	×
$\overline{RESET}$	Output	Reset signal	$\overline{RESET}$	√	√	√
SI/RXD	Input	Receive signal	SO/TXD	√	√	√
SO/TXD	Output	Transmit signal	SI/RXD	√	√	√
SCK	Output	Transfer clock	SCK	×	√	√
HS	Input	Handshake signal for CSI + HS communication	HSPORT	x	×	√

Note √: Must be connected.
×: Does not have to be connected.

Table 6-7 Wiring of V850E/RG3 flash writing adapters for CSIF

Flash programmer (FG-FP4) connection pin			Name of FA board pin	UARTD0	CSIF0 + HS	CSIF0
Signal name	I/O	Pin function		Pin name	Pin name	Pin name
SI/RxD	I	Receive signal	SI	TXDC0	SOB0	
SO/TxD	O	Transmit signal	SO	RXDC0	SIB0	
SCK	O	Transfer clock	SCK	Not needed	SCKB0	
CLK	O	Clock to V850 microcontroller	X1	Leave open		
			X2	Leave open		
$\overline{\text{RESET}}$	O	Reset signal	$\overline{\text{RESET}}$	$\overline{\text{RESET}}$ (please see caution bellow)		
FLMD0	I	Write voltage	FLMD0	FLMD0 (please see caution bellow)		
FLMD1	I	Write voltage	FLMD1	FLMD1		
HS	I	Handshake signal for CSI + HS	RESERVE/ HS	Not needed	HSPORT = P84	Not needed
VDD	–	VDD voltage generation/ voltage monitor	VDD	V_{DD3x}		
				BV_{DD3x}		
				AV_{REF0}		
GND	–	Ground	GND	V_{SS3x}		
				BV_{SS3x}		
				AV_{SS}		

Caution The programmer pins FLMD0 and $\overline{\text{RESET}}$ may not support a 5 V interface and may require a level shifter. Refer to the user's manual of the programmer to be used.

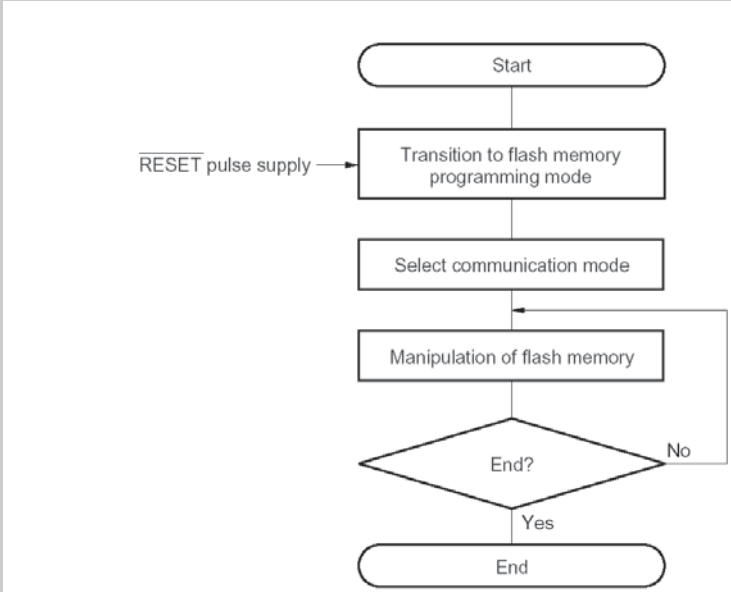
Table 6-8 V850E/RG3 pin numbers for serial programming

Pin name	Port	V850E/RG3 pin nr.
TXDD0	P84	Pin 61
RXDD0	P83	Pin 60
SIB0	P80	Pin 57
SOB0	P81	Pin 58
SCKB0	P82	Pin 59
$\overline{\text{RESET}}$	–	Pin 128
FLMD0	–	Pin 137
FLMD1	PCS1	Pin 115
HSPORT	P84	Pin 61

6.4.4 Flash memory programming control

The procedure to program the flash memory is illustrated below.

Figure 6-6 Flash memory programming procedure



Note: A reset pulse is required to initiate the selection of the flash programming mode.

(1) Operation mode control

To rewrite the contents of the flash memory by using the flash programmer, set the V850 microcontroller in the flash memory programming mode.

To set this mode, set the FLMD0 and FLMD1 pins as shown in *Table 6-9* on page 129 and release $\overline{\text{RESET}}$.

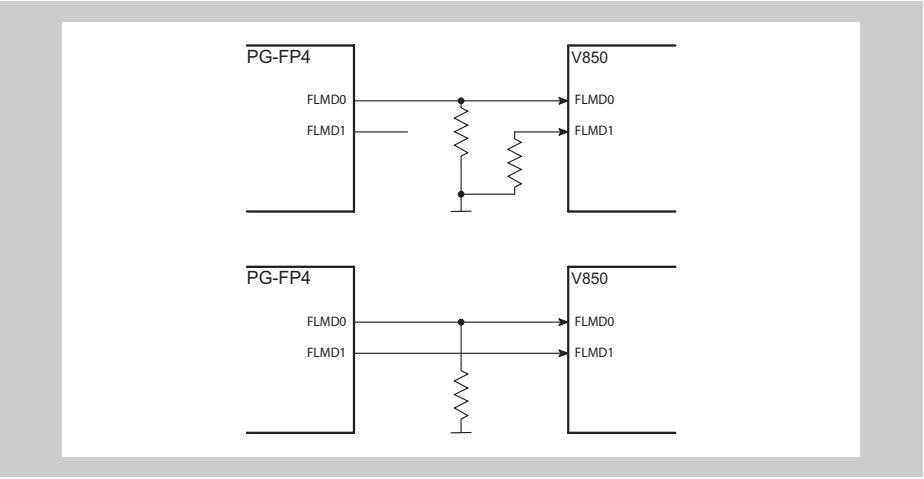
In the normal operation mode, 0 V is input to the FLMD0 pin. A pull-down resistor at the FLMD0 pin ensures normal operation mode if no flash programmer is connected. In the flash memory programming mode, the V_{DD} write voltage is supplied to the FLMD0 pin. Additionally the FLMD1 pin has to hold 0 V level.

Table 6-9 Operation mode setting

Pins		Operation mode
FLMD0	**FLMD1**	
V_{SS}	X	Normal operation mode
V_{DD}	V_{SS}	Flash programming mode
	V_{DD}	Setting prohibited

An example of connection of the FLMD0 and FLMD1 pins is shown below. FLMD1 can be connected to ground via a resistor. Alternatively the FLMD1 pin may also be connected directly to the FLMD1 signal of the flash programmer.

Figure 6-7 Example of connection to flash programmer PG-FP4

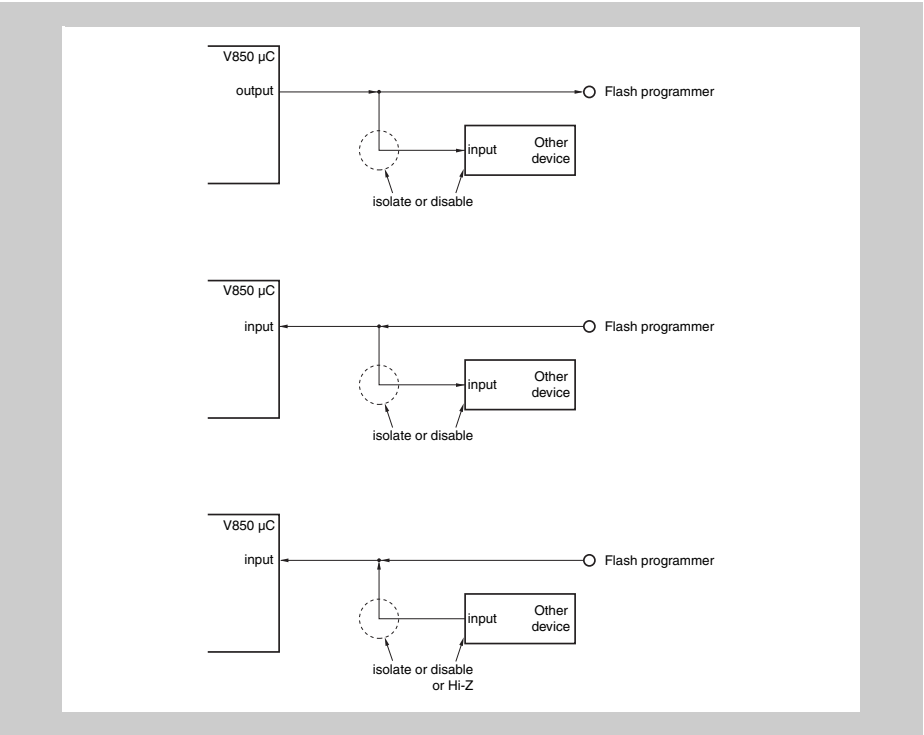


Once started in normal operation mode (FLMD0 = 0), FLMD0 pin is used for enabling self-programming. Refer also to *Section 6.5, Code Flash Self-Programming* on page 134.

(2) Potential conflicts with on-board signal connections

Serial I/O signals 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 external device signals do not conflict with the signals of the flash programmer and the V850 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.

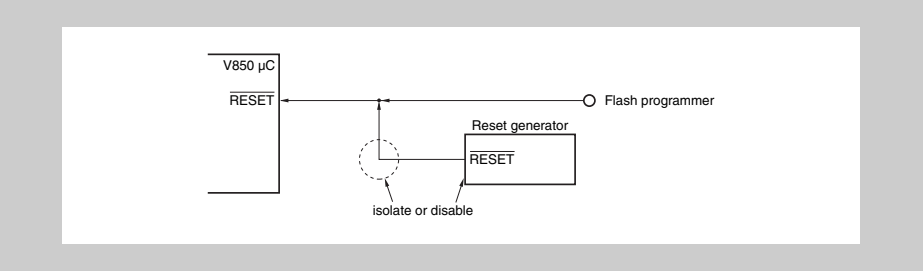
Figure 6-8 Potential conflicts with serial interfaces signals



$\overline{\text{RESET}}$ Pay attention in particular if the flash programmer's $\overline{\text{RESET}}$ signal is also connected to an on-board reset generation circuit. The reset output of the reset

generator may interfere with the flash programming process and may need to be isolated or disabled.

Figure 6-9 Potential conflict with $\overline{\text{RESET}}$



Ports The V850 port pins adopt the following status during serial programming:

Ports used for programming are configured as UART or CSI pins.

All other pins remain in their default state after reset release.

In case the default state of the pins not used for programming after reset is input port or high-impedance output port, pay attention to other devices connected to these pins. If these devices require defined levels at the pins, the ports may have to be connected to V_{DD} or V_{SS} via a resistor.

Oscillators Connect all oscillator pins in the same way as in the normal operation mode.

$\overline{\text{DRST}}$ During flash memory programming, input a low level to $\overline{\text{DRST}}$ or leave it open. Do not input a high level.

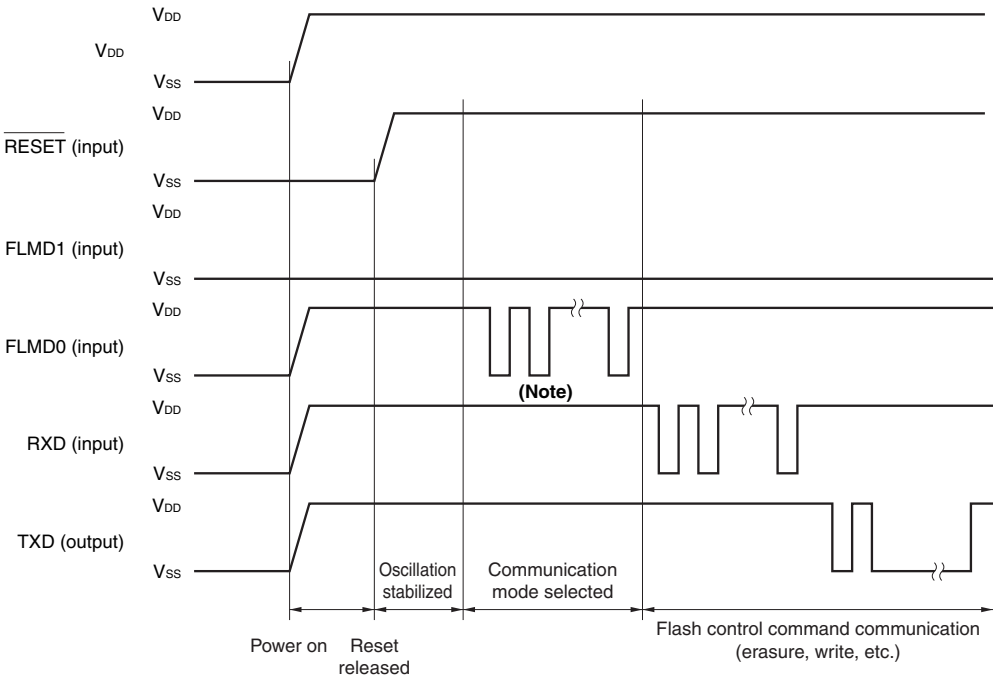
Power supply Supply the same power to all power supply pins, including reference voltages, power regulator pins, etc., as in the normal operation mode.

(3) Selection of the communication mode

The communication interface is chosen by applying a specified number of pulses to the FLMD0 pin after reset release. Note that this is handled by the flash programmer.

Figure 6-10 on page 132 gives an example how the UART is established for the communication between the flash programmer and the V850 microcontroller.

Figure 6-10 Selection of communication mode



Note The number of clocks to be inserted differs depending on the chosen communication mode. For details, refer to *Table 6-10* on page 132.

Table 6-10 FLMD0 pulses for communication mode setting

FLMD0 pulses	Communication Mode	Remarks
0	UART	Communication rate: 9600 bps (after reset), LSB first
8	CSI	V850E/RG3 performs slave operation, MSB first
11	CSI + HS	V850E/RG3 performs slave operation, MSB first
Other	–	Setting prohibited

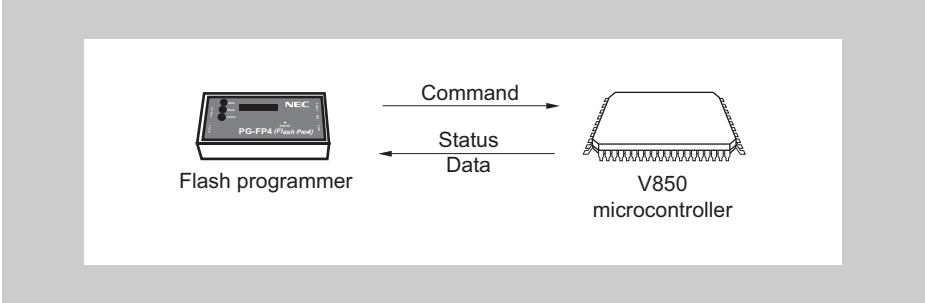
When UART has been selected after reception of the FLMD0 pulses with 9600 bps, the flash programmer changes the baud rate according to the user's choice via the flash programmer's user interface.

At first the programmer sends two 00_H bytes, which are used by the microcontroller to measure the baud rate and to set up it's own baud rate accordingly.

(4) Communication commands

The flash programmer sends commands to the V850 microcontroller. Depending on the commands, the V850 microcontroller returns status information or the requested data.

Figure 6-11 Communication commands exchange



The following table lists the flash memory control commands of the V850 microcontroller. All these commands are issued by the flash programmer, and the V850 microcontroller performs the corresponding processing.

Table 6-11 Flash memory control commands

Classification	Command name	Support			Function
		CSI	CSI + HS	UART	
Blank check	Block blank check command	√	√	√	Checks erasure status of entire memory.
Erase	Chip erase command	√	√	√	Erases all memory contents.
	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.
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.
	Protection setting command	√	√	√	Sets protection against chip erasure, block erasure, and writing.

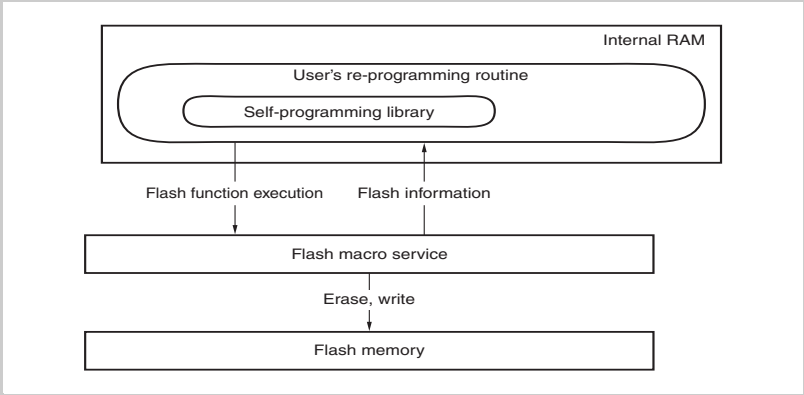
6.5 Code Flash Self-Programming

This V850 microcontroller supports a flash macro service that allows the user program to rewrite the internal flash memory by itself.

By using this flash macro service and a self-programming library, provided by NEC, the user's program is able to rewrite the flash memory with data, transferred in advance to the internal RAM.

Thus the user program can be upgraded and constant data can be rewritten in the field.

Figure 6-12 Concept of self-programming



During self-programming, access to the flash memory is not possible. Thus program execution is only possible by instruction fetching from internal RAM.

Consequently the user re-programming software routines, which must remain in operation during the self-programming procedure, must be copied from the flash memory to the internal RAM prior to activating the self-programming. Since interrupt processing by using the interrupt vectors in the flash memory is also impossible during self-programming, a special feature is provided to re-route interrupt acknowledges to the internal RAM (refer to *Section 6.5.4, Interrupt handling during flash self-programming* on page 137).

It is recommended to refer to the Application Note "Self-Programming" (document nr. U16929EE) for comprehensive information concerning flash self-programming. This document also explains the functions of the self-programming library. The latest version of this document and the library can be loaded via the URL

http://www.eu.necel.com

Caution The self-programming operation also employs internal firmware, which makes use of 9 KB of the internal RAM in the address range FFFF $CC00_H$ to FFFF EFFFH.
Thus this RAM area is overwritten and must be re-initialized after completion of the self-programming.
Also consider that the stack additionally occupies up to 1 KB of the RAM.

6.5.1 Self-programming enable

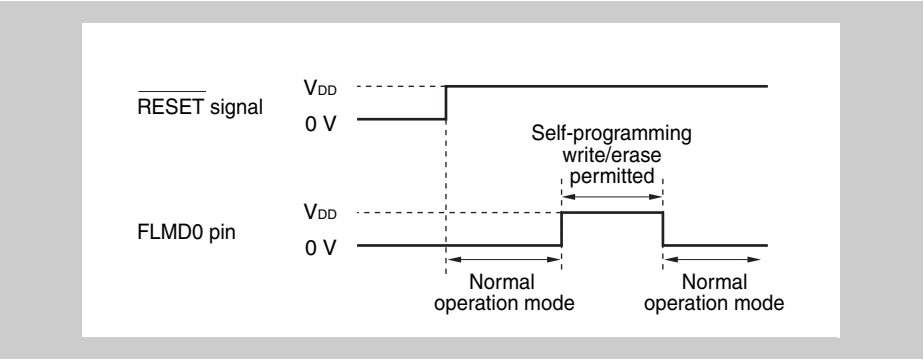
The self-programming functions can be started out of the normal user mode of the V850 microcontroller.

The V850 microcontroller must be set into self-programming mode via the self-programming library.

For security reasons, writing and erasing of the flash memory must also be permitted by setting the external FLMD0 pin to high level. Note that FLMD0 holds low level in normal operation mode after reset release.

This requires some external components or wiring, e.g. connecting an output port to FLMD0.

Figure 6-13 Self-programming enable



When self-programming has been completed, the voltage on the FLMD0 pin must be returned to 0 V.

6.5.2 Self-programming library functions

Code flash memory self-programming by the user's program is supported by the self-programming library.

This library provides a set of C function calls to carry out basic functions such as:

- blank-check/erase/rewrite/verify of the flash
- boot cluster swapping, including definition of boot block clusters
- setting of protection flags
- obtain various information concerning the code flash memory

Detailed information on how to use the library functions is given in the Application Note: "Self-Programming Library for embedded Single Voltage FLASH" (document no. U16929EE).

The up-to-date version of the self-programming library and the above mentioned Application Note can be obtained from *http://www.eu.necel.com*.

6.5.3 Secure self-programming (boot cluster swapping)

The V850 flash microcontrollers support a mechanism to swap a cluster of code flash memory blocks, starting from address 0000 0000_H, with another cluster of the same size, located immediately above the first one.

Boot swap cluster

The cluster of blocks starting at address 0000 0000_H is named active boot swap cluster, since it contains the entry point of the user's program at the default reset vector 0000 0000_H. The boot swap cluster size is fixed to 32 KB.

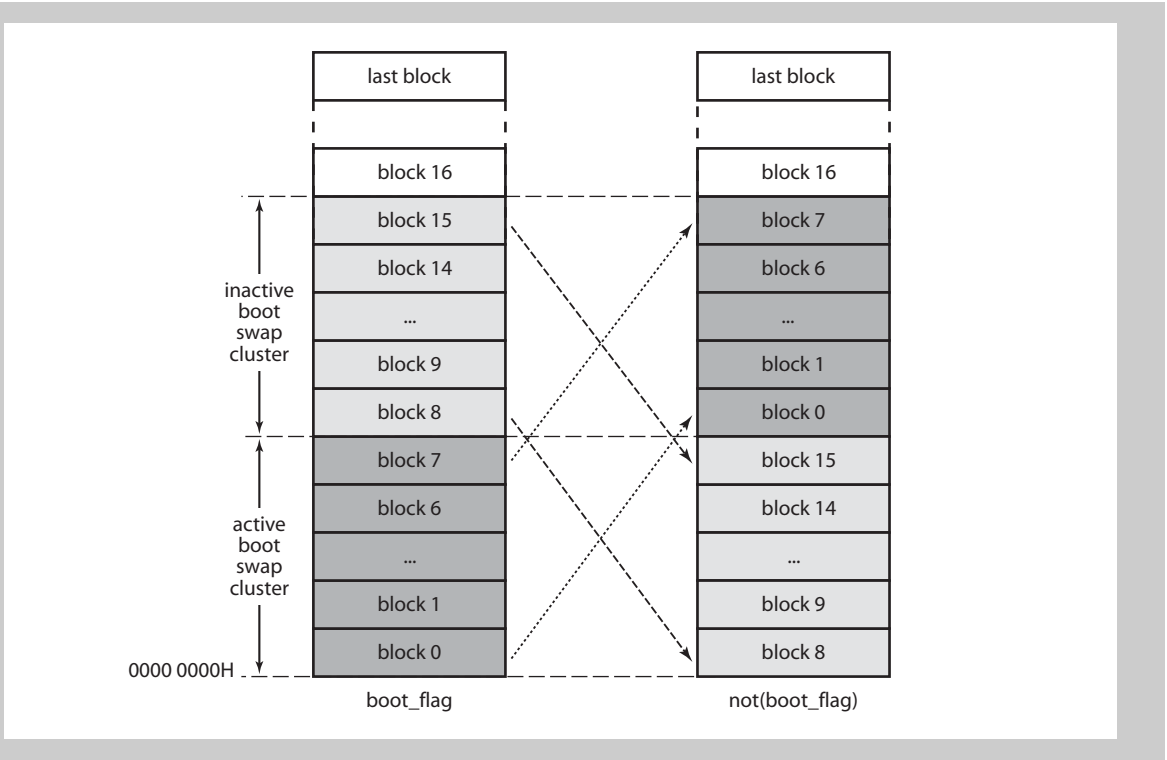
Boot swap flag

The boot swap flag controls which of the two clusters is the active boot block cluster, and can be defined during flash programming via the self-programming library.

The boot swap flag is stored in the flash memory extra area.

Figure 6-14 on page 136 shows an example of the boot block swapping function with a cluster size of 8 flash memory blocks. After inverting the boot_flag - it becomes not(boot_flag) - blocks 8 to 17 become the active boot block cluster. Thus, after the next reset release the user's program starts from the new boot swap cluster.

Figure 6-14 Boot swap cluster swapping function



Secure self-programming

The boot cluster swapping function enables secure self-programming. If a new boot block cluster needs to be created, the new code can be written to the inactive boot block cluster, while the boot_flag remains in its previous state. If writing the new boot block cluster is completed successfully, the boot_flag can be inverted, making the new boot code active.

If writing the new boot code fails for any reason, e.g., a power failure or unintended reset, the old boot code still remains active and rewriting can be started again.

Boot block protection

To prohibit rewriting of the boot blocks, the boot block cluster protection flag can be set during flash memory programming. When this flag is set, the blocks

of the active boot block cluster can neither be erased nor written. Boot cluster swapping is impossible as well.

Caution Once the boot block cluster protection has been activated, it can never be deactivated again.

6.5.4 Interrupt handling during flash self-programming

This V850 microcontroller provides functions to maintain interrupt servicing during the self-programming procedure.

Since neither the interrupt vector table nor the interrupt handler routines, which are normally located in the code flash memory, are accessible while self-programming is active, interrupt acknowledges have to be re-routed to non-flash memory, i.e. to the internal RAM.

Therefore two prerequisites are necessary to enable interrupt servicing during self-programming:

- The interrupt handler routine needs to be copied to the appropriate location in internal RAM. The user program has to perform this copy process.
- The interrupt acknowledge has to be re-routed to the relocated handler in RAM. Re-routing to the handler is done by the internal firmware. Thus the user program does not have to handle this process.

The internal firmware and the self-programming library provide functions to initialize and process such interrupts.

The interrupt handler routines can be copied from flash to the internal RAM by use of self-programming library functions.

The addresses of the interrupt handler routines are set up via the self-programming library as well.

Note
1. Note that this special interrupt handling adds some interrupt latency time.
2. Special interrupt handling is performed only while the flash self-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 *Section 3.2.2, System register set* on page 40)
- 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 *Section 3.2.2, System register set* on page 40).

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 servicing the interrupt needs to be compiled as an interrupt function (i.e., terminate with a RETI instruction, save/restore all used registers, etc.).

It is recommended to refer to the Application Note “Self-Programming” (document nr. U16929EE) for comprehensive information concerning flash self-programming. This document explains also the functions of the self-programming library. The latest version of this document can be loaded via the URL

http://www.eu.necel.com

Chapter 7 Data Protection and Security

7.1 Overview

The microcontroller supports various methods for securing safe (re-)programming of the internal flash memory and protecting the flash memory data against undesired access, such as illegal read-out or illegal reprogramming.

Security functions

Security functions enhance the reliability of the flash operation during normal operation. They also support countermeasures against unexpected failures during reprogramming processes. These are basically:

- Secure self-programming
- Secure bootloader update
- Boot swapping
- Boot block cluster protection

These functions are described in detail in *Section Chapter 6, Flash Memory* on page 117.

Protection functions

Protection functions provide a set of mechanisms to protect the internal flash memory data from being read, erased, or altered by unauthorized persons. These are basically:

- On-chip (N-Wire) debug interface protection
- Flash memory erase/write/read protection via the serial programming interface

Some interfaces offer general access to the internal flash memory: N-Wire debug interface, external flash programmer interfaces and self-programming facilities. All of these interfaces need to be considered for a proper protection concept.

The following sections give an overview about supported protection methods.

7.2 N-Wire Debug Interface Protection

In general, illegal read-out of the flash memory contents is possible via the N-Wire debug interface. To avoid this, the debug interface can be protected and it can be disabled. The debug interface is protected via a 10-byte ID code and an internal flag (N-Wire use enable flag).

When the debugger is started, the status of a flag is queried (N-Wire use enable flag). Set this flag to zero to disable the use of the N-Wire in-circuit emulator.

When debugging is enabled (N-Wire use enable flag is set to 1), you have to enter a 10-byte ID code via the debugger. The code is compared with the ID code stored in the internal flash memory. If the codes do not match, debugging is not possible.

The N-Wire use enable flag can be set or reset while reprogramming the flash by an external flash writer or with the self-programming feature. The flag is located at bit 7 at address 0000 0079_H.

You can specify your own 10-byte ID code and program it to the internal flash memory by an external flash writer or with the self-programming feature. The ID code is located in the address range 0000 0070_H to 0000 0079_H.

The protection levels are summarized in *Table 7-1*.

Table 7-1 Possible results of ID code comparison

N-Wire use enable flag	ID code	Protection Level
0	X[a]	Level 2: Full protection N-Wire debug interface cannot be used.[b]
1	user-specific ID code	Level 1: ID code protection user ID code N-Wire debug interface can only be used if the user enters the correct ID code.
	ID code is all ones[c]	Level 0: ID code protection with default ID code N-Wire debug interface can be used if the user enters the default ID code FF_H for all ID bytes.

a) Codes are not compared
b) Once the N-Wire debug interface has been set as “use-prohibited”, it cannot be used until the flash memory is re-programmed.
c) This is the default state after the flash memory has been erased.

Note 1. After you have set protection levels 1 or 2, set the “block erase disable flag” in the flash extra area. Otherwise, an unauthorized person could erase the block that contains the ID code or the “N-Wire use enable flag”, respectively, and thus suspend the protection.

7.2.1 Additional N-Wire disable via FMOP0 flash mask option

Additonal to bit 7 at address 0000 0079_H can be diasabled via the FMOP0.OCDMI bit. For description please refer to 4.2.3“Flash Mask Option Register (FMOP0)” on page 73.

For disableling the N-Wire functionaltity, both options should be used.

Remark To use the shared I/O functionality the FMOP0.OCDMI bit has to be set to 0.

7.3 Flash Programmer 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. The available flash memory protection methods are as follows.

Serial programming It is possible to prohibit any external access via the serial programming interface, e.g., by an external flash programmer. With maximum protection the internal flash memory can not be erased, read-out or written at all, neither in block units nor the entire flash memory.

Self-programming During self-programming all operations to erase, read or program the flash memory are under control of the user's program. Thus no further protection functions in self-programming mode are considered. One exception is the boot block protection, which applies also in self-programming mode.

Protection flags The protection flags can be reset by an external flash programmer, provided the effective protection level allows it to do so. In self-programming mode, the effective protection flags can not be reset, but otherflags can be set to enhance the protection level. The protection flags are stored in the flash memory extra area.

Each protection function can be used in combination with the others at the same time.

7.3.1 Write protection flag

Set this flag to disable the write function via external flash programmer interfaces.

No flash memory content can be written from external interfaces if this flag is set. Erasure of single blocks is prohibited as well.

This flag does not affect the self-programming interface. In self-programming mode, writing of the flash memory is still possible.

7.3.2 Chip erase protection flag

Set this flag to disable the chip erase function via external flash programmer interfaces.

No flash memory content can be erased - neither in single blocks nor the entire flash memory - from external interfaces if this flag is set.

Chip erase is not available in self-programming mode, though it is possible to erase the entire flash memory content by block erase of all blocks all together. Note that the contents of the extra area are not erased by this means. I.e., protection flags, variable reset vector, etc. are still valid.

7.3.3 Block erase protection flag

Set this flag to disable the feature to erase single blocks via external flash programmer interfaces.

Single blocks can not be erased. Chip erase is still possible, provided the chip erase protection flag is not set.

This flag does not affect the self-programming interface. In self-programming mode, erasure of single blocks or sets of contiguous blocks of the flash memory is still possible.

7.3.4 Read-out protection flag

Set this flag to disable the feature that allows reading back the flash memory via the serial programming interfaces. No flash content can be read out.

This flag does not affect the self-programming interface. In self-programming mode read-out of flash memory content is still possible.

7.3.5 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 it. Thus the boot block cluster content can not be changed any more.

For the explanation of the boot block cluster refer to *Section 6.5.3, Secure self-programming (boot cluster swapping)* on page 136.

All protection flags are reset after shipment of the device, thus no protection is enabled at all.

Once a protection flag has been set, i.e., the protection is effective, it can not be reset by any means, except after a chip erase, which erases the entire flash memory including the extra area.

Consequently, without prior chip erase the protection level can only be increased, but not decreased.

Table 7-2 Protection functions overview

Function	Functional outline	Applicable (√: applies, ×: doesn't apply)	
		Serial programming	Self-programming
Block erase command prohibit	Erasure of single blocks impossible. Once block erase protection is enabled, disable is only possible after chip erase.	√	×
Chip erase command prohibit	Erasure of the entire flash (including the extra area) or single blocks impossible. Once chip erase protection is enabled, all protection flag settings can not be changed any more.	√	×
Program command prohibit	Erasure and rewrite of single blocks impossible. Once write protection is enabled, disable is only possible after chip erase.	√	×
Read command prohibit	Read-out of any flash content impossible. Once read protection is enabled, disable is only possible after chip erase.	√	×
Rewriting boot block cluster prohibit	Erasure (by block or chip erase) or writing of the boot block cluster impossible. Once rewrite protection of the boot block cluster is enabled, it can not be disabled any more.	√	√

Table 7-3 Rewriting operation when erasing/writing is enabled/prohibited

Prohibition state		Programming mode	Block erasure		Chip erasure	Write	
			Boot area	None boot area		Boot area	None boot area
Rewriting boot area enabled	All enabled	Self-programming	yes		–	yes	
		Serial programming	yes		yes	yes	
	Block erase command prohibited	Self-programming	yes		–	yes	
		Serial programming	no		yes	yes	
	Chip erase command prohibited	Self-programming	yes		–	yes	
		Serial programming	no		no	yes	
	Write command prohibited	Self-programming	yes		–	yes	
		Serial programming	no		yes	no	
Rewriting boot area prohibited	All enabled	Self-programming	no	yes	–	no	yes
		Serial programming		yes	no		yes
	Block erase command prohibited	Self-programming		yes	–		yes
		Serial programming		no	yes		yes
	Chip erase command prohibited	Self-programming		yes	–		yes
		Serial programming		no	no		yes
	Write command prohibited	Self-programming		yes	–		yes
		Serial programming		no	yes		no

Note –: not supported

Table 7-4 Read operation when reading is enabled/prohibited

Prohibition State	Programming mode	Read
Read command enabled	Self-programming	√
	Serial programming	√
Read command prohibited	Self-programming	√
	Serial programming	×

Note √: execution enabled, ×: execution disabled, –: not supported

Chapter 8 DMA Controller (DMAC)

The microcontroller includes a direct memory access controller (DMAC) that executes and controls DMA transfers.

Note Throughout this chapter, the individual channels of the DMA Controller are identified by “n” (n = 0, 2 to 7)

8.1 Overview

The DMAC controls data transfer between internal RAM and I/O, based on DMA requests issued by the on-chip peripheral I/O.

Features summary The DMAC has the following features:

- 7 independently configurable channels
- Programmable access wait and signal timing
- Maximum DMA transfer count: 256 (2^8)

8.1.1 Principle of operation

A DMA transfer essentially transfers a predefined number of data units between the internal RAM (iRAM) and peripheral I/O registers. The CPU initiates the transfer and the transfer itself is performed by the DMAC.

DMA channels Each DMA channel has a fixed transfer direction to transfer data between the internal RAM and a peripheral I/O register:

- DMA channels 0, 4, 5: I/O → iRAM
- DMA channels 2, 3, 6, 7: iRAM → I/O

DMA priorities The DMA channel priorities are fixed as follows:
DMA channel 0 > DMA channel 2 > … > DMA channel 7

Transfer data units The size of the data unit to be transferred is either 8 bit or 16 bit:

- DMA channels 0: 16 bit
- DMA channels 2 to 3: n x 16 bit (n=1, 2)
- DMA channels 4 to 7: 8 bit or 16 bit selectable

Transfer addresses For a DMA transfer, the following addresses have to be defined:

- Source address

  The address of the location of the data that is to be copied. When data is being transferred from internal RAM to a peripheral I/O register, the MARn register contains the source address. When data is being transferred from a peripheral I/O register to internal RAM, the corresponding peripheral register contains the source address.
- Destination start address

  The address of the location where the data is to be copied. When data is being transferred from internal RAM to a peripheral I/O register, the corresponding peripheral I/O register contains the destination address.

  When data is being transferred from a peripheral I/O register to internal RAM, the MARn register contains the destination address.

Whether used to store a source address or destination address, the MARn register is automatically incremented after each DMA data transfer. The size of the transfer data unit, determines how much the register is incremented.

– 8-bit data: address increment by 1
– 16-bit data: address increment by 2

Transfer start/stop

The DMA transfer is initiated by defined interrupts of the on-chip peripheral I/Os. These interrupts are called DMA triggers.

To start a DMA transfer, the following conditions have to be fulfilled:

- DMA transfer is enabled (DMAMC.DEn = 1).
- The number of transfer actions that has to be performed (transfer count) is written to register DTRCn.
- The transfer status is set to “idle” (DMAS.DMASn = 0).
- DMA trigger is generated.

When the defined number of transfer actions is completed, a DMA completion interrupt (INTDMAn) is generated and the DMA transfer is stopped.

Transfer status

The transfer status can be monitored:

- Register DMAS displays the transfer status (idle/in progress/completed).
- Register DTCRn displays the current transfer count (it is decreased with every transfer action).

8.1.2 Forcible termination of DMA transfer

A DMA transfer that has been initiated can be forcibly terminated by clearing the bit DMAMC.DEn = 0.

If the DMAMC.DEn bit is cleared during a DMA transfer action, the current transfer action is finished before the DMA transfer is stopped (see *Section 8.3.3, Example of forcible termination of DMA transfer* on page 160).

8.1.3 DMA interrupt function

The DMA triggers are interrupts of

- A/D Converter (ADC)
- Serial interfaces (CSIE, CSIF)

These interrupts are shared with their corresponding DMA transfer completion interrupts (INTDMAn). If a DMA channel is enabled, the peripheral I/O interrupt corresponding to the DMA trigger can no longer generate an interrupt request. Instead, the DMA transfer completion interrupt can generate an interrupt with the corresponding interrupt handler address.

Table 8-1 shows the relations between DMA triggers and DMA completion interrupts.

Table 8-1 Relations between DMA triggers and DMA completion interrupts

DMA channel	DMA trigger	DMA completion interrupt		
		Name	Entry in interrupt list[a]	Handler address
0	INTAD0	INTDMA0	INTAD0D	0000 0430_H
1	not supported			
2	INTCE0C	INTDMA2	INTDMA2	0000 0440_H

Table 8-1 Relations between DMA triggers and DMA completion interrupts

DMA channel	DMA trigger	DMA completion interrupt		
		Name	Entry in interrupt list[a]	Handler address
3	INTCE1C	INTDMA3	INTDMA3	0000 0460$_H$
4, 5	INTCE0C	INTDMA4, INTDMA5	INTCE0C	0000 0450$_H$
	INTCE1C		INTCE1C	0000 0470$_H$
	INTCF0R		INTCF0R	0000 0490$_H$
	INTCF1R		INTCF1R	0000 04B0$_H$
6, 7	INTCF0T	INTDMA6, INTDMA7	INTCF0T	0000 0480$_H$
	INTCF1T		INTCF1T	0000 04A0$_H$

a) The interrupt/exception list is given in *Section Chapter 5, Interrupt Controller (INTC)* on page 81

8.2 Registers

The DMA Controller is controlled by means of the following registers:

Table 8-2 DMA Controller register base address

Unit	Base address
DMA Controller	FFFF F300$_H$

Table 8-3 DMA Controller register overview

Register name	Shortcut	Address
DMA wait control registers	DMAWC0	FFFF FE20$_H$
DMA transfer SFR start address registers	SAR2	<base> + 24$_H$
	SAR3	<base> + 26$_H$
DMA transfer memory start address registers	MARn (n = 0, 2 to 7)	<base> + n x 2$_H$
DMA trigger factor registers	DTFR4	<base> + 88$_H$
	DTFR5	<base> + 8A$_H$
	DTFR6	<base> + 8C$_H$
	DTFR7	<base> + 8E$_H$
DMA transfer count registers	DTCRn (n = 0, 2 to 7)	<base> + 40$_H$ + n x 2$_H$
DMA status register	DMAS	<base> + 62$_H$
DMA mode control register	DMAMC	<base> + 60$_H$
DMA data size control register	DMADSC	<base> + 64$_H$

8.2.1 DMAWCm - DMA wait control register

The 8-bit DMAWCm registers control the internal bus timing for DMA transfers (m= 0, 1).

Access These registers can be read/written in 8-bit or 1-bit units.

Address DMAWC0: FFFF $FE20_H$

DMAWC1: FFFF $FE22_H$

Initial Value DMAWC0 = 37_H

DMAWC1 = 07_H

DMAWC0 The DMAWC0 register must have the same setting as the internal peripheral function wait control register VSWC:

DMAWC0 = VSWC.

Note The contents of the VSWC register depend on the system clock.

DMAWC1 The DMAWC1 register must be set as follow :

DMAWC1 = 00_H

Caution The DMAWC0 register must be set up correctly before the first DMA transfer is started. Do not change this register afterwards.

8.2.2 SARn - DMA transfer SFR address register

The 8-bit SARn register specifies the CSIE register to which data will be transferred from RAM during a DMA transfer. It also configures the DMA for 16-bit or 32-bit data transfer.

When DMA channel 2 or 3 is configured for 16-bit transfers, the data from RAM is transferred to the CEnTX register for transmission by the CSIE. However, when DMA channel 2 or 3 is configured for 32-bit transfers, the first 16 bits of data from RAM are transferred to the CSIEn Chip Selection CSI Buffer register (CE0CS) in order to configure the chip select pins for the next CSIE transmission. Then, the second 16 bits of data are transferred to the CEnTX register for transmission.

This register is only valid for DMA transfers to CSIE.

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

Address SAR2: <base> + 24_H

SAR3: <base> + 26_H

Initial Value Undefined

7	6	5	4	3	2	1	0
0	0	0	0	SARn3	SARn2	SARn1	SARn0
R	R	R	R	R/W	R/W	R/W	R/W

SARn3	SARn2	SARn1	SARn0	Data unit size	Destination register			
					n = 2		n = 3	
					Register	Address	Register	Address
1	0	1	0	32-bit	CE0CS	FFFF FD44$_H$	CE1CS	FFFF FD84$_H$
1	0	1	1	16-bit	CE0TX	FFFF FD46$_H$	CE1TX	FFFF FD86$_H$
other setting than above					prohibited			

Note Set SARnm to 1010 for 32-bit transfer and to 1011 for 16-bit transfer to CSIEn.

8.2.3 MARn - DMA transfer RAM address register

The 16-bit MARn register specifies the lower 16 bits of the address of the location within the internal RAM to be used for a DMA transfer using DMA channel n. The address is considered the source address when data is transferred from RAM (transfer direction iRAM → I/O), and is considered the destination address when data is transferred to RAM (transfer direction I/O → iRAM).

within the internal RAM area for the DMA channel n.

Access This register can be read/written in 16-bit units.

Address MARn: <base> + n x 2_H

Initial Value Undefined.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MARn15	MARn14	MARn13	MARn12	MARn11	MARn10	MARn9	MARn8	MARn7	MARn6	MARn5	MARn4	MARn3	MARn2	MARn1	MARn0
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

Bit name	Function
MARn15 to MARn0	The lower 16 bits of the DMA transfer address (A15 to A0) for internal RAM.

Cautions

1. Due to the internal RAM size, the value written to the MARn register has to be in a specific range depending on the device:
   - $B000_H$ to $EFFF_H$ for 16KB RAM devices
   - 9000_H to $EFFF_H$ for 24KB RAM devices
   - 7000_H to $EFFF_H$. for 32KB RAM devices
2. For DMA channels that are configured for 16-bit transfer data units, the RAM source or destination area must be 16 bit aligned. Thus the memory address must be even, that is MARn.MARn0 = 0.

8.2.4 DTFRn - DMA trigger factor register

The 8-bit DTFRn register controls the DMA transfer start trigger of DMA channel n via interrupt requests from on-chip peripheral I/O (n = 4 to 7).

The interrupt request set by this register serves as DMA transfer start factor.

This register is only valid for DMA transfers to serial interfaces (CSIE, CSIF).

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

Address DTFR4: <base> + 88_H
DTFR5: <base> + $8A_H$
DTFR6: <base> + $8C_H$
DTFR7: <base> + $8E_H$

Initial Value 00_H. This register is cleared by $\overline{\text{RESET}}$.

7	6	5	4	3	2	1	0
0	0	0	0	0	IFCn2	IFCn1	IFCn0
R	R	R	R	R	R/W	R/W	R/W

Cautions

1. Do not set the same DMA trigger by different DTFRn registers.
2. Do not rewrite the DTFRn register until a started DMA transfer ends (corresponding DTCRn register value is 00_H).
3. Write the DTFRn register before setting the corresponding DTCRn register. Otherwise, if the DMA trigger is present, the DMA transfer might be started by writing the DTFRn register.

IFCn2	IFCn1	IFCn0	DMA transfer start factor	
			n = 4, 5	n = 6, 7
0	0	0	DMA request from on-chip peripheral I/O disabled	
0	0	1	n/a	
0	1	0	n/a	
0	1	1	INTCF0R	INTCF0T
1	0	0	INTCF1R	INTCF1T
1	0	1	INTCE0C	n/a
1	1	0	INTCE1C	n/a
1	1	1	n/a	

8.2.5 DTCRn - DMA transfer count registers

The 8-bit DTCRn register has two functions:

- Before DMA transfer, it sets the transfer count for DMA channel n
- During DMA transfer, it stores the remaining transfer count.

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

Address DTCRn: <base> + 40_H + n x 2_H

Initial Value Undefined.

7	6	5	4	3	2	1	0
DTCRn7	DTCRn6	DTCRn5	DTCRn4	DTCRn3	DTCRn2	DTCRn1	DTCRn0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

DTCRn7	DTCRn6	…	DTCRn2	DTCRn1	DTCRn0	Remaining DMA transfer counts
0	0	…	0	0	0	256
0	0	…	0	0	1	1
0	0	…	0	1	0	2
0	0	…	0	1	1	3
…	…	…	…	…	…	...
1	1	…	1	0	1	253
1	1	…	1	1	0	254
1	1	…	1	1	1	255

Cautions

1. The value set to the DTCRn register is decreased by each DMA transfer of channel n. It does not keep the initial value after the DMA transfer ends. Therefore, after DMA transfer end the DTCRn register value becomes 00_H.
2. A DMA request becomes only effective after the DTCRn register was written. Even if 00_H (means a transfer count of 256) is the initial value, the DTRCn register must be rewritten in order to enable a new DMA transfer.

8.2.6 DMAS - DMA status register

The 8-bit DMAS register displays the transfer status of the DMA channels.

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

Address <base> + 62_H

Initial Value 00_H. This register is cleared by $\overline{\text{RESET}}$.

7	6	5	4	3	2	1	0
DMAS7	DMAS6	DMAS5	DMAS4	DMAS3	DMAS2	DMAS1	DMAS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

DMASn	Displays the transfer status of DMA channel n
0	Transfer is idle or on progress
1	Transfer is completed

Note
1. The DMASn 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.
2. Since the DMASn bit is not cleared by the DMAC, it has to be cleared by software before DMA transfer is started.

8.2.7 DMAMC - DMA mode control register

The 8-bit DMAMC register enables/disables the operation of the DMA channels.

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

Address <base> + 60_H

Initial Value 00_H. This register is cleared by $\overline{\text{RESET}}$.

7	6	5	4	3	2	1	0
DE7	DE6	DE5	DE4	DE3	DE2	DE1	DE0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

DEn	Enables/disables DMA transfer of channel n
	Enables/disables DMA transfer of channel n: 0: Operation disabled 1: Operation enabled

Caution Writing of a DEn bit is prohibited if the corresponding peripheral function, that generates the DMA trigger, is operating.

8.2.8 DMADSC - DMA data size control register

The 8-bit DMADSC register controls the transfer data size of DMA channels 4 to 7. The transfer data size of DMA channels 0 to 3 is fixed to 16 bits.

This register is only valid for DMA transfers to serial interfaces.

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

Address <base> + 64_H

Initial Value 00_H. This register is cleared by $\overline{\text{RESET}}$.

7	6	5	4	3	2	1	0
DMADSC7	DMADSC6	DMADSC5	DMADSC4	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

DMADSCn	Sets the transfer data size of DMA channel n
n = 4 to 7	0: 8 bits 1: 16 bits

Note 32-bit transfer (2 x 16-bit) for DMA channels 2 and 3 has to be configured by *Section 8.2.2, SARn - DMA transfer SFR address register* on page 148.

8.3 Transfer Details

This section presents details and examples of all possible DMA transfers:

- *Section 8.3.1, DMA transfer from A/D Converter* on page 153
- *Section 8.3.2, DMA transfer to/from serial interfaces* on page 156

Further, an *Section 8.3.3, Example of forcible termination of DMA transfer* on page 160 is given.

8.3.1 DMA transfer from A/D Converter

The following table summarizes the DMA transfer ADCm → iRAM (m = 0).

Table 8-4 Overview of DMA transfer from A/D Converter

Channel n	Transfer unit	DMA trigger		Transfer			
				Size	Max. count	Source register	Destination start address MARn
0	ADC0	INTAD	End of conversion interrupt signal of A/D Converter	16 bit	256	ADDMA0 (fixed)	any even address in iRAM

For each DMA trigger, the data will be transferred from the A/D conversion result register for DMA (ADDMAm) to the internal RAM area specified by the destination start address in MARn.

Transferred data The data that is to be transferred is defined by the following:

- A/D Converter scan area

The number of ADC channels that are to be scanned can be defined in the A/D Converter.

The DMA transfer is performed for every finished A/D conversion within the A/D Converter scan area. Thus, for every ADC channel, a separate DMA transfer is required.

- DMA transfer count

  The number of DMA transfers has to be defined in the DTCRn register. The total number of DMA transfers can be calculated as

  scan area size × number of DMA triggers

Note The user has to take care that the number of DMA transfer counts complies with the A/D Converter scan area size and the number of A/D Converter start triggers.

For an example see *Section (2), Example of DMA transfer of A/D Converter result registers* on page 154.

(1) Principle of DMA transfer of A/D Converter result registers

CPU

1. Set A/D Converter scan area (ADMn2 register of the A/D converter)
2. Set destination start address in iRAM (MARn register)
3. Set DMA transfer count (1 to 256, DTCRn register)
4. Clear status bit of DMA channel n (DMAS.DMASn = 0)
5. Enable DMA channel n (DMAMC.DEn = 1)
6. Enable A/D Converter n

ADC

7. Waiting for A/D Converter start trigger
8. A/D sampling and conversion of next channel in ADCR register
9. A/D Converter generates DMA request and triggers step 12
10. If the complete scan area has not been converted yet, go to step 7
11. Go to step 6

DMA

12. Waiting for DMA trigger (INTAD signal)
13. Data transfer from ADDMA0 register to destination start address in iRAM
14. Increment MARn register by 2 (destination start address)
15. Decrement DTCRn by 1 (transfer count)
16. If DMA transfer is not terminated (DTCRn > 0), then go to step 11
17. Stop DMA transfer and disable DMA channel n
18. Set DMAS.DMASn = 1 (transfer status)
19. Generate DMA completion interrupt (INTDMAn)

(2) Example of DMA transfer of A/D Converter result registers

DMA transfer for A/D Converter 0 (ADC0):

- A/D Converter scan area is selected from channel 0 to channel 2.
- Destination start address in iRAM is 0FFF $C000_H$

- A/D scan area should be transferred twice, means trigger count is 2.

CPU

1. Set A/D Converter 0 scan area in register ADM02: channel 0 to channel
2. Set destination start address to 0FFF $C000_H$: MAR0 = $C000_H$
3. Set DMA transfer count (2 triggers x 3 channels = 6): DTCR0 = 06_H
4. Clear status bit of DMA channel 0 (DMAS.DMAS0 = 0)
5. Enable DMA channel 0: DMAMC.DE0 = 1
6. Enable A/D Converter 0

ADC0

7. Waiting for ADC0 start trigger
8. A/D sampling and conversion of channel 0
9. A/D Converter generates DMA request signal INTAD

DMA

10. Data transfer from ADDMA0 register to destination address 0FFF $C000_H$
11. Increment MAR0 register to 0FFF $C0002_H$
12. Decrement DTCR0 register to 05_H

ADC0

13. A/D sampling and conversion of channel 1
14. A/D Converter generates DMA request signal INTAD

DMA

15. Data transfer from ADDMA0 register to destination address 0FFF $C002_H$
16. Increment MAR0 register to 0FFF $C0004_H$
17. Decrement DTCR0 register to 04_H

ADC0

18. A/D sampling and conversion of channel 2
19. A/D Converter generates DMA request signal INTAD

DMA

20. Data transfer from ADDMA0 register to destination address 0FFF $C004_H$
21. Increment MAR0 register to 0FFF $C006_H$
22. Decrement DTCR0 register to 03_H

ADC0

23. Waiting for ADC0 start trigger (INTAD0)
24. A/D sampling and conversion of channel 0
25. A/D Converter generates DMA request signal INTAD

DMA

26. Data transfer from ADDMA0 register to destination address 0FFF $C006_H$
27. Increment MAR0 register to 0FFF $C008_H$
28. Decrement DTCR0 register to 02_H

ADC0 29. A/D sampling and conversion of channel 1

30. A/D Converter generates DMA request signal INTAD

DMA 31. Data transfer from ADDMA0 register to destination address 0FFF $C008_H$

32. Increment MAR0 register to 0FFF $C00A_H$

33. Decrement DTCR0 register to 01_H

ADC0 34. A/D sampling and conversion of channel 2

35. A/D Converter generates DMA request signal INTAD

DMA 36. Data transfer from ADDMA0 register to destination address 0FFF $C00A_H$

37. Increment MAR0 register to 0FFF $C00C_H$

38. Decrement DTCR0 register to 00_H

39. Stop DMA transfer and disable DMA channel 0, since DTCR0 = 00_H

40. Set DMAS.DMAS0 = 1

41. Generate DMA completion interrupt (INTDMA0)

8.3.2 DMA transfer to/from serial interfaces

(1) Serial data reception with DMA transfer for CSIFn and CSIEn

The DMA channels 4 and 5 are dedicated to transfer data from serial interfaces CSIE0, CSIE1, CSIF0, CSIF1 to the iRAM.

The following table summarizes the DMA transfer serial interface → iRAM.

Table 8-5 Overview of DMA transfer from serial interfaces

Channel n	Transfer unit	DMA trigger		Transfer			
				Size	Max. count	Source address	Destination start address (MARn)
4	CSIE0	INTCE0C	Receive complete interrupt	8 bit	255	CE0RXL0	Any iRAM address
				16 bit		CE0RX0	Any even iRAM address
	CSIE1	INTCE1C		8 bit		CE1RXL0	Any iRAM address
				16 bit		CE1RX0	Any even iRAM address
	CSIF0	INTCF0R		8 bit		CF0RX0L	Any iRAM address
				16 bit		CF0RX0	Any even iRAM address
	CSIF1	INTCF1R		8 bit		CF1RX0L	Any iRAM address
				16 bit		CF1RX0	Any even iRAM address
5	CSIE0	INTCE0C		8 bit		CE0RXL0	Any iRAM address
				16 bit		CE0RX0	Any even iRAM address
	CSIE1	INTCE1C		8 bit		CE1RXL0	Any iRAM address
				16 bit		CE1RX0	Any even iRAM address
	CSIF0	INTCF0R		8 bit		CF0RX0L	Any iRAM address
				16 bit		CF0RX0	Any even iRAM address
	CSIF1	INTCF1R		8 bit		CF1RX0L	Any iRAM address
				16 bit		CF1RX0	Any even iRAM address

Source address The source address is any of the listed serial interfaces' data receive registers. It is implicitly defined by the channel number n and the selection made by the DMA trigger factor register DTFRn register.

Destination address The destination address is any iRAM address (for 16-bit transfers an even iRAM address), specified by the DMA memory start address MARn.

DMA trigger Upon each DMA trigger 8-bit or 16-bit data is transferred from the corresponding serial reception register to the internal RAM.

The DMA trigger is the end-of-reception interrupt signal of the selected serial interface. Like the source address, it is defined by the DMA channel number and the DMA trigger factor register DTFRn register.

Transfer size The transfer data size can be set to 8- or 16-bit via the register DMADSC.

- In case of 8-bit transfer data size, the destination address is incremented by 1 for each DMA trigger.
- In case of 16-bit transfer data size, the destination address must be even, and is incremented by 2 for each DMA trigger.

DMA transfer procedure of serial data reception

CPU

1. Set destination start address in iRAM (MARn register)
2. Select demanded serial interface (DTFRn register)
3. Set up DMA transfer data size (8 or 16 bits) of channel n (DMADSC.DMADSCn)
4. Set DMA transfer count (1 to 255, DTCRn register)
5. Clear status bit of DMA channel n (DMAS.DMASn = 0)
6. Enable DMA channel n (DMAMC.DEn = 1)

DMA

7. Waiting for occurrence of DMA trigger
8. Data transfer from source in peripheral I/O register to iRAM
9. Decrement DTCRn register by 1
10. If DMA transfer is not terminated (DTCRn > 0), then go to step 7
11. Stop DMA transfer and disable DMA channel n
12. Set DMAS.DMASn = 1 (transfer status)
13. Generate DMA completion interrupt (INTDMAn)

(2) Serial data transmission with DMA transfer for CSIEn

The DMA channels 2 and 3 are dedicated to transfer data from the iRAM to the serial interfaces CSIE0, CSIE1. The following table summarizes the DMA transfer iRAM → CSIEn:

Table 8-6 Overview of DMA transfer from serial interfaces

Channel n	Transfer unit	DMA trigger		Transfer				
				Size	Max. count	Source address (MARn)	Destination start address	End address
2	CSIE0	INTCE0	Transmit interrupt	1x16 bit	256	Any iRAM address	CE0TX	CE0TX
				2x16 bit		Any even iRAM address	CE0CS	
3	CSIE1	INTCE1		1x16 bit		Any iRAM address	CE1TX	CE1TX
				2x16 bit		Any even iRAM address	CE1CS	

Source address The source address is any iRAM address (for 16-bit transfers an even iRAM address), specified by the DMA memory start address MARn.

Destination address Transfer data is copied to areas defined by the following:

- Start address is defined in the SARn registers. Possible start addresses are either CEnCS or CEnTX registers of the CSIEn macro.
- End address is always the CEnTX register of the CSIEn macro.

DMA trigger Upon each DMA trigger one or two 16-bit data are transferred from the internal RAM to the corresponding serial transmit register(s).

The DMA trigger is the transmission interrupt signal INTCEn of the selected CSIEn macro.

Transfer size Only 16-bit data can be transferred, but it is possible to send one or two 16-bit data. The number of 16-bits data to be transferred is defined by the start address as follows:

- One 16-bit data for start address set as CEnTX
- Two 16-bit data for start address set as CEnCS

DMA Transfer Procedure for 32-bit serial data transmission

CPU

1. Set CEnCS register as start address (SRAn = $0A_H$)
2. Set DMA source address in iRAM (MARn register)
3. Set DMA transfer count (1 to 255, DTCRn register)
4. Clear status bit of DMA channel n (DMAS.DMASn = 0)
5. Enable DMA channel n (DMAMC.DEn = 1)

DMA

6. Waiting for occurrence of DMA trigger (INTCEn)
7. Data transfer from source in iRAM to CEnCS register
8. Data transfer from source in iRAM to CEnTX register
9. Decrement DTCRn register by 1
10. If DMA transfer is not terminated (DTCRn > 0), then go to step 7
11. Stop DMA transfer and disable DMA channel n
12. Set DMAS.DMASn = 1 (transfer status)
13. Generate DMA completion interrupt (INTDMAn)

(3) Serial data transmission with DMA transfer for CSIFn

The DMA channels 6 and 7 are dedicated to transfer data from the iRAM to the serial interfaces CSIF0, CSIF1.

The following table summarizes the DMA transfer iRAM → serial interface:

Table 8-7 Overview of DMA transfer from serial interfaces

Channel n	Transfer unit	DMA trigger		Transfer			
				Size	Max. count	Source address (MARn)	Destination start address
6	CSIF0	INTCF0T	Transmit interrupt	8 bit	256	Any iRAM address	CF0TXL
				16 bit		Any even iRAM address	CF0TX
	CSIF1	INTCF1T		8 bit		Any iRAM address	CF1TXL
				16 bit		Any even iRAM address	CF1TX
7	CSIF0	INTCF0T		8 bit		Any iRAM address	CF0TXL
				16 bit		Any even iRAM address	CF0TX
	CSIF1	INTCF1T		8 bit		Any iRAM address	CF1TXL
				16 bit		Any even iRAM address	CF1TX

Source address The source address is any iRAM address (for 16-bit transfers an even iRAM address), specified by the DMA memory start address MARn.

Destination address The destination address is any of the listed serial interfaces' data transmit registers. It is implicitly defined by the channel number n and the selection made by the DMA trigger factor register DTFRn register.

DMA trigger Upon each DMA trigger 8-bit or 16-bit is transferred from the internal RAM to the corresponding serial transmit register.

The DMA trigger is the transmit-enable interrupt signal of the selected serial interface. Like the destination address, It is defined by the DMA channel number and the DMA trigger factor register DTFRn register.

Transfer size The transfer size can be set to 8- or 16- bit via the register DMADSC.

- In case of 8-bit transfer data size, the source address is incremented by 1 for each DMA trigger.
- In case of 16-bit transfer data size, the source address MARn must be even, and is incremented by 2 for each DMA trigger.

DMA Transfer Procedure of Serial Data Transmission

CPU

1. Set source address in iRAM (MARn register)
2. Select demanded serial interface (DTFRn register)
3. Set up DMA transfer data size (8 or 16 bits) of channel n (DMADSC.DMADSCn)
4. Set DMA transfer count (1 to 255, DTCRn register)
5. Clear status bit of DMA channel n (DMAS.DMASn = 0)
6. Enable DMA channel n (DMAMC.DEn = 1)

DMA

7. Waiting for occurrence of DMA trigger
8. Data transfer from source in iRAM to peripheral I/O register
9. Decrement DTCRn register by 1
10. If DMA transfer is not terminated (DTCRn > 0), then go to step 7
11. Stop DMA transfer and disable DMA channel n

12. Set DMAS.DMASn = 1 (transfer status)
13. Generate DMA completion interrupt (INTDMAn)

8.3.3 Example of forcible termination of DMA transfer

CPU
1. Set DMA transfer count (e.g. DTCRn = 10_H)
2. Start DMA transfer (DMAMC.DEn = 1)

DMA
3. Waiting for DMA trigger
4. Start first data transfer
5. Decrement DTCRn register by 1 (e.g. DTCRn = 09_H)
6. Waiting for DMA trigger
7. Start next data transfer

CPU
8. Stop DMA transfer forcibly (DMAMC.DEn = 0)

DMA
9. Decrement DTCRn register by 1 (e.g. DTCRn = 08_H)
10. DMA transfer is stopped completely

Chapter 9 16-Bit Timer/Event Counter AA (TAA)

The V850E/Rx3 includes four channels of the 16-bit Timer/Event Counter AA (TAA0 to TAA3). The timer is upward compatible to Timer P used in various other devices of the V850E and the V850ES family. It offers new additional features for enhanced output control and for 32-bit capture.

Instances The V850E/RG3 has four instances of this 16-bit timer/event counter AA:

Table 9-1 Instances of Timer AA

Timer AA	
Instances	4
Names	TAA0 to TAA3

Throughout this chapter, the individual instances of Timer AA are identified by “n” (n = 0 to 3), for example, TAAnCTL0 for the TAAn control register 0.

9.1 Features

Timer AA (TAA) is a 16-bit timer/event counter provided with general-purpose functions.

TAA can perform the following operations.

- 16-bit-accuracy PWM output timer
- Interval timer
- External event counter function
- Timer synchronised operation function
- One-shot pulse output
- Pulse interval and frequency measurement counter
- Free running function
- External trigger pulse output function
- 32-bit capture timer function by cascading 2 channels of TAA

9.2 Function Outline

- Capture trigger input signal × 2
- External trigger input signal × 1
- Clock select × 8
- External event count input × 1
- Readable counter × 1
- Capture/compare reload register × 2
- Capture/compare match interrupt × 2

- Timer output (TOAAn0, TOAAn1) × 2
- 32-bit capture by cascading two timer AA (TAA2+TAA3).

9.3 Configuration

TAA includes the following hardware.

Table 9-2 Timer TAA registers and external connections

Item	Configuration
Timer register	16-bit counter
Registers	• TAAn timer capture/compare registers 0, 1 (TAAnCCR0, TAAnCCR1) • TAAn timer read buffer register (TAAnCNT) • CCR0 buffer register, CCR1 buffer register
Input selection registers	• TAAn input selection registers (SELCNT1, SELCNT2)
Timer output	TOAAn0, TOAAn1
Timer input	TIAAn0, TIAAn1, TTRGAAn, TEVTAAn
Control registers	• TAAn control registers 0, 1 (TAAnCTL0, TAAnCTL1) • TAAn I/O control registers 0 to 3 (TAAnIOC0, TAAnIOC1, TAAnIOC2, TAAnIOC4) • TAAn option registers 0, 1 (TAAnOPT0, TAAnOPT1)

Timer AA (TAA) pins provide an alternate function of port pins. For information on how to set the alternate function, refer to the description of the registers in *Section Chapter 2, Pin Functions* on page 7.

Figure 9-1 and *Figure 9-2* show the block diagrams of the input circuits of the different timers TAAn.

Clock supply

Each timer AA provides 6 clock inputs. All are connected to the clock generator.

The following table lists the cross reference between TAAn clock input and the connected clocks:

Table 9-3 Clock inputs of TAAn

TAAn clock input	Connected clock		
	TAA0	TAA1	TAA2, TAA3
TAAnTCKI0	PCLK0 (32 MHz)	PCLK1 (16 MHz)	PCLK1 (16 MHz)
TAAnTCKI3	PCLK3 (4 MHz)	PCLK4 (2 MHz)	PCLK4 (2 MHz)
TAAnTCKI4	PCLK4 (2 MHz)	PCLK5 (1 MHz)	PCLK5 (1 MHz)
TAAnTCKI5	PCLK5 (1 MHz)	PCLK6 (500 KHz)	PCLK6 (500 KHz)
TAAnTCKI6	PCLK6 (500 KHz)	PCLK7 (250 KHz)	PCLK7 (250 KHz)
TAAnTCKI7	PCLK7 (250 KHz)	BRGOUT	PCLK8 (125 KHz)

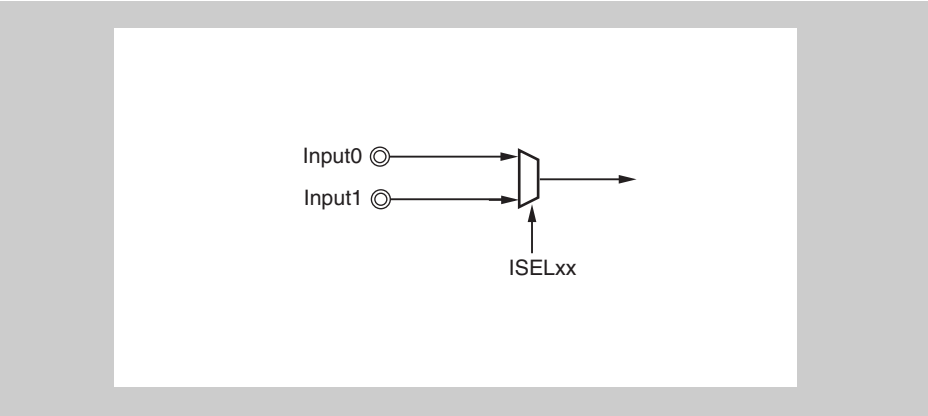
9.4 Input Selection Registers

To enhance the functionality of the timer macros, they are externally connected to different sources. The source being used can be configured with the input selection register.

The possible connections and configurations are described below.

9.4.1 TIAAn0 - Timer input connection block diagram

Figure 9-1 Input circuit



9.4.2 SEL0CNT - Selector Control Register 0

The SEL0CNT register is an 8-bit register that controls the external input pin source of the capture register 0 and 1 of TAA0.

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

Address FFFF F6E0$_H$

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

7	6	5	4	3	2	1	0
ISEL07	ISEL06	0	ISEL04	0	ISEL02	0	ISEL00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

ISEL07	Selection of ADC timer trigger input (TTRGADC)
0	INTTAA0CC0, trigger from TAA0 capture /compare match 0
1	INTTMM0EQ0 interrupt of TMM0

ISEL06	Selection of TIAA30 input signal (TAA3)
0	TIAA03 pin input
1	RXDD1 pin input

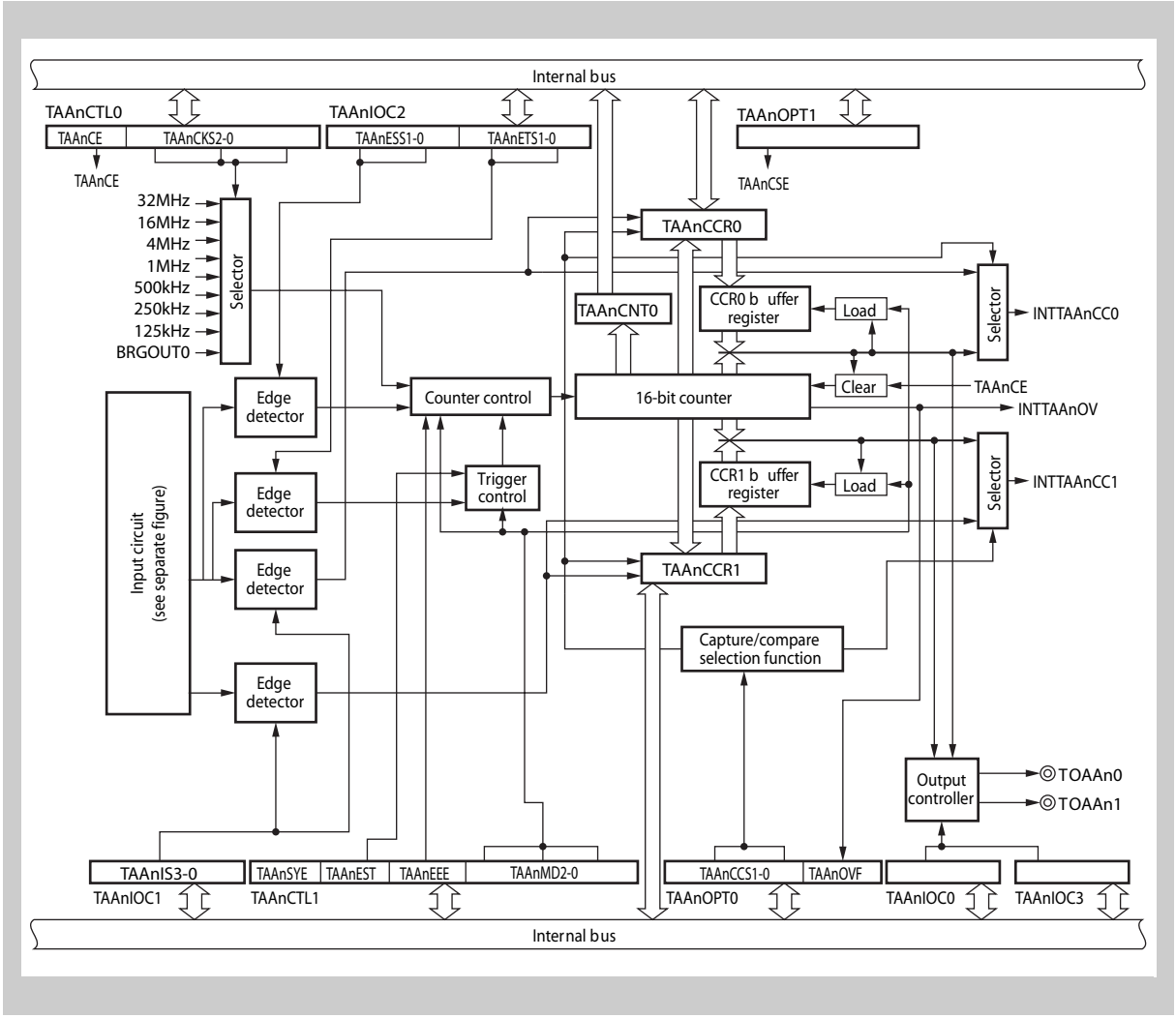
ISEL04	Selection of TIAA20 input signal (TAA2)
0	TIAA02 pin input
1	RXDD0 pin input

ISEL02	Selection of TIAA10 input signal (TAA1)
0	TIAA01 pin input
1	Trigger signal (TSOUT1)

ISEL00	Selection of TIAA00 input signal (TAA0)
0	TIAA00 pin input
1	Trigger signal (TSOUT0)

9.5 Timer AA Block Diagram

Figure 9-2 Block Diagram of Timer AA



9.6 Control Registers

Register addresses

All TAAn register addresses are given as address offsets from the individual base addresses <base> of each TAAn.

The <base> addresses of each TAAn are listed in the following table:

Table 9-4 Register <base> addresses of TAAn

TAAn	<base> address
TAA0	FFFF $F600_H$
TAA1	FFFF $F620_H$
TAA2	FFFF $F640_H$
TAA3	FFFF $F660_H$

The TAAn are controlled and operated by means of the following registers:

Table 9-5 TAAn registers overview

Register name	Shortcut	Address
TAAn control register 0	TAAnCTL0	<base>
TAAn control register 1	TAAnCTL1	<base> + 1_H
TAAn I/O control register 0	TAAnIOC0	<base> + 2_H
TAAn I/O control register 1	TAAnIOC1	<base> + 3_H
TAAn I/O control register 2	TAAnIOC2	<base> + 4_H
TAAn option register 0	TAAnOPT0	<base> + 5_H
TAAn capture/compare register 0	TAAnCCR0	<base> + 6_H
TAAn capture/compare register 1	TAAnCCR1	<base> + 8_H
TAAn counter read buffer register	TAAnCNT	<base> + A_H
TAAn I/O control register 4	TAAnIOC4	<base> + C_H
TAAn option register 1	TAAnOPT1	<base> + D_H

9.6.1 TAAnCTL0 - TAA control register 0

TAAn control register 0 is an 8-bit register that controls the operation of timer AA.

Access This register can be read/written in 8-bit or 1-bit units.
The TAAnCTL0 register is prohibited from writing during operation (TAAnCE=1). However, the TAAnCE bit can be rewritten.

Address <base>

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

7	6	5	4	3	2	1	0
TAAnCE	0	0	0	0	TAAnCKS2	TAAnCKS1	TAAnCKS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

TAAnCE	Timer AAn operation control
0	Disable internal operating clock operation (TAAn is asynchronously reset)
1	Enable internal operating clock operation
Internal operating clock control and TAAn asynchronous reset are performed with the TAAnCE bit. When TAAnCE bit is cleared to 0, the internal operating clock of TAAn stops (fixed to low level) and TAAn is reset asynchronously. When the TAAnCE bit is set to 1, the internal operating clock is enabled within 2 input clocks, and TAAn counts up.	

Note In the following modes TAAnCTL0.TAAnCE cannot be set to “1”:

- Slave timer in synchronous operation mode
  If the timer is operated as the slave timer in synchronous operation mode, i.e. TAAnCTL1.TAAnSYE = 1.
- Slave timer in 32-bit cascaded capture mode
  If timer TAAn is operated in 32-bit capture mode for capturing the upper 16 bit

TAAnCKS2	TAAnCKS1	TAAnCKS0	Internal count clock selection
0	0	0	TAAnTCKI0
0	0	1	TAAnTCKI0/2
0	1	0	TAAnTCKI0/4
0	1	1	TAAnTCKI3
1	0	0	TAAnTCKI4
1	0	1	TAAnTCKI5
1	1	0	TAAnTCKI6
1	1	1	TAAnTCKI7

Caution Set bits TAAnCKS2 to TAAnCKS0 only when TAAnCE = 0.
When TAAnCE bit setting is changed from 0 to 1, TAAnCKS2 to TAAnCKS0 bits can be set simultaneously.

9.6.2 TAAnCTL1 - TAA timer control register 1

TAAn control register 1 is an 8-bit register that controls the operation of timer AA.

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
TAAnSYE	TAAnEST	TAAnEEE	0	0	TAAnMD2	TAAnMD1	TAAnMD0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

TAAnSYE	Synchronous operation mode enable control
0	Independent operation mode (asynchronous operation mode)
1	Synchronous operation mode (specification of slave operation) In this mode, timer AA can operate in synchronization with a master timer. If TAAnSYE = 1, TAAnCTL0.TAAnCE cannot be set to “1”. For the synchronous operation mode, refer to *Section 9.8, Timer Synchronization Operation Function* on page 217. **Caution:** Be sure to clear the TAAnSYE to 0, if TAAn is used as the master timer. Respectively, set the TAAnSYE = 1, if TAAn is used as slave timer.

TAAnEST	Software trigger control
0	No operation
1	In one-shot pulse mode: One-shot pulse software trigger
	In external trigger pulse output mode: Pulse output software trigger
The TAAnEST bit functions as a software trigger in the one-shot pulse mode or external trigger pulse output mode (this bit is invalid in any other mode). By setting TAAnEST to 1 when TAAnCE = 1, a software trigger is issued. Therefore, be sure to set TAAnEST to 1 after setting TAAnCE = 1. The TIAAn0 pin is used for an external trigger. The read value of the TAAnEST bit is always 0.	

TAAnEEE	Count clock selection
0	Use the internal clock (clock selected with TAAnCKS2 to TAAnCKS0 bits of TAAnCTL0 register)
1	Use external clock (TEVTAAn input edge)
The valid edge is specified with TAAnEES1 and TAAnEES0 bits when TAAnEEE bit = 1 (external clock TEVTAAn).	

TAAnMD2	TAAnMD1	TAAnMD0	Timer mode selection
0	0	0	Interval timer mode
0	0	1	External event counter mode
0	1	0	External trigger pulse output mode
0	1	1	One-shot pulse mode
1	0	0	PWM mode
1	0	1	Free-running mode
1	1	0	Pulse width measurement mode
1	1	1	Setting prohibited

Cautions

1. Set bits TAAnEEE and TAAnMD2 to TAAnMD0 when TAAnCE = 0. (The same value can be written when TAAnCE = 1.) The operation is not guaranteed when rewriting is performed when TAAnCE = 1. If rewriting was mistakenly performed, clear TAAnCE to 0 and then set the bits again.
2. In the external event count mode the external event count input is selected regardless of the value of the TAAnEEE bit.
3. Set the count clock to internal clock (TAAnEEE = 0) when you use an external trigger pulse mode, the single shot pulse mode, and the pulse

length measurement mode.

4. Set the edge detection of the TIAAn0 capture input to no detection when you use an external event count mode (TAAnEES1 of the TAAnIOC2 register and TAAnEES0 = 00).

9.6.3 TAAnIOC0 - TAA dedicated I/O control register 0

The TAAnIOC0 register is an 8-bit register that controls the timer output.

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
0	0	0	0	TAAnOL1	TAAnOE1	TAAnOL0	TAAnOE0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

TAAnOLm	TOAAnm output level setting
0	Normal output
1	Inverted output
This bit can be used to invert the timer output	

TAAnOEm	TOAAnm output setting
0	Timer output is disabled (Low level and high level are output from TOAAnm pin when TAAnOLm = 0 and TAAnOLm = 1, respectively.)
1	Timer output is enabled (TOAAnm pin outputs pulses.)

Cautions

1. Rewrite bits TAAnOLm and TAAnOEm when TAAnCE = 0 (the same value can be written when TAAnCE = 1.). If rewriting was mistakenly performed, clear TAAnCE to 0 and then set the bits again.
2. To enable the timer output, be sure to set the corresponding alternate-function pins TAAnIS3 to TAAnIS0 of the TAAnIOC1 register to “No edge detection” and invalidate the capture operation. Then set the corresponding alternate-function port to output mode.

Note m = 0, 1

9.6.4 TAAnIOC1 - TAA dedicated I/O control register 1

The TAAnIOC1 register is an 8-bit register that controls the valid edge for the external input signals (TIAAn0 and TIAAn1).

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 cleared by any reset.

7	6	5	4	3	2	1	0
0	0	0	0	TAAnIS3	TAAnIS2	TAAnIS1	TAAnIS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

TAAnIS3	TAAnIS2	Capture input (TIAAn1) valid edge setting
0	0	No edge detection (capture operation is invalid)
0	1	Detection of rising edge
1	0	Detection of falling edge
1	1	Detection of both edges

TAAnIS1	TAAnIS0	Capture input (TIAAn0) valid edge detection
0	0	No edge detection (capture operation is invalid)
0	1	Detection of rising edge
1	0	Detection of falling edge
1	1	Detection of both edges

Cautions

1. Bits TAAnIS3 to TAAnIS0 are valid only in the free-running capture mode and pulse width measurement mode. In all the other modes, the capture operation is not performed.
2. When using this as the capture input, be sure to set the corresponding alternate function pins TAAnOE1 and TAAnOE0 of the TAAnIOC0 register to “Timer output is disabled” and set the capture input valid edge. Then set the corresponding alternate-function port to input mode.
3. In the external event count mode (TAAnCTL1.TAAnEEE = 1), set the TIAAn0 capture input to "No edge detection" (TAAnIS1 and TAAnISO bits = 00).

Rewrite during timer operation

If the edge specification for the capture operation will be changed while the timer remains in operation (TAAnCTL0.TAAnCE = 1), only a single bit of the edge specification bits TAAnIOC1.TAAnIS[k = 0 or 1, i = 0 or 1] of a dedicated capture input may be changed with a single write operation.

Consequently, proceed as follows (TIAAn0 is used as an example):

- Change from rising edge to falling edge:

  - current status is TAAnIOC1.TAAnIS[1:0] = 01_B: “rising edge”
  - set TAAnIOC1.TAAnIS[1:0] = 00_B: specify “no edge”
  - set TAAnIOC1.TAAnIS[1:0] = 10_B: specify “falling edge”
- Change from falling edge to rising edge:
  - current status is TAAnIOC1.TAAnIS[1:0] = 10_B: “falling edge”
  - set TAAnIOC1.TAAnIS[1:0] = 00_B: specify “no edge”
  - set TAAnIOC1.TAAnIS[1:0] = 01_B: specify “rising edge”
- Change from rising or falling edge to both edges:
  - current status is TAAnIOC1.TAAnIS[1:0] = 01_B or 10_B: “rising” or “falling edge”
  - set TAAnIOC1.TAAnIS[1:0] = 11_B: specify “both edges”

9.6.5 TAAnIOC2 - TAA I/O control register 2

The TAAnIOC2 register is an 8-bit register that controls the valid edge for external event count input signals (TEVTAAn) and external trigger input signal (TTRGAAn).

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 cleared by any reset.

7	6	5	4	3	2	1	0
0	0	0	0	TAAnEES1	TAAnEES0	TAAnETS1	TAAnETS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

TAAnEES1	TAAnEES0	External event count input valid edge setting (TEVTAAn)
0	0	No edge detection (external event is invalid)
0	1	Detection of rising edge
1	0	Detection of falling edge
1	1	Detection of both edges

TAAnETS1	TAAnETS0	External trigger input valid edge detection (TTRGAAn)
0	0	No edge detection (external trigger is invalid)
0	1	Detection of rising edge
1	0	Detection of falling edge
1	1	Detection of both edges

Cautions
1. Rewrite TAAnEES1, TAAnEES0, TAAnETS1, and TAAnETS0 bits when TAAnCE = 0 (the same value can be written when TAAnCE = 1). If rewriting was mistakenly performed, clear TAAnCE to 0 and then set the bits again.

2. TAAnEES1 and TAAnEES0 bits are valid only when TAAnEEE = 1 or when the external event count mode has been set (TAAnCTL1.TAAnMD[2:0] = 001_B).
3. TAAnETS1 and TAAnETS0 bits are only valid when the external trigger pulse output mode or one-shot pulse mode is set (TAAnMD[2:0] = 010_B or 011_B).

Rewrite during timer operation

If the edge specification for the external event count input and external trigger input shall be changed, while the timer remains in operation (TAAnCTL0.TAAnCE = 1), only a single bit of the edge specification bits TAAnIOC2.TAAnEES[k = 0 or 1, i = 0 or 1] / TAAnIOC2.TAAnETS[k = 0 or 1, i = 0 or 1] of a dedicated capture input may be changed with a single write operation.

Consequently proceed as follows (TIAAn0 is used in this example):

In external event counter mode:

- Change from rising edge to falling edge:
  - current status is TAAnIOC2.TAAnEES[1:0] = 01_B: “rising edge”
  - set TAAnIOC2.TAAnEES[1:0] = 00_B: specify “no edge”
  - set TAAnIOC2.TAAnEES[1:0] = 10_B: specify “falling edge”
- Change from falling edge to rising edge:
  - current status is TAAnIOC2.TAAnEES[1:0] = 10_B: “falling edge”
  - set TAAnIOC2.TAAnEES[1:0] = 00_B: specify “no edge”
  - set TAAnIOC2.TAAnEES[1:0] = 01_B: specify “rising edge”
- Change from rising or falling edge to both edges:
  - current status is TAAnIOC2.TAAnEES[1:0] = 01_B or 10_B: “rising” or “falling edge”
  - set TAAnIOC2.TAAnEES[1:0] = 11_B: specify “both edges”

In external trigger mode:

- Change from rising edge to falling edge:
  - current status is TAAnIOC2.TAAnETS[1:0] = 01_B: “rising edge”
  - set TAAnIOC2.TAAnETS[1:0] = 00_B: specify “no edge”
  - set TAAnIOC2.TAAnETS[1:0] = 10_B: specify “falling edge”
- Change from falling edge to rising edge:
  - current status is TAAnIOC2.TAAnETSS[1:0] = 10_B: “falling edge”
  - set TAAnIOC2.TAAnETS[1:0] = 00_B: specify “no edge”
  - set TAAnIOC2.TAAnEtS[1:0] = 01_B: specify “rising edge”
- Change from rising or falling edge to both edges:
  - current status is TAAnIOC2.TAAnETS[1:0] = 01_B or 10_B: “rising” or “falling edge”
  - set TAAnIOC2.TAAnETS[1:0] = 11_B: specify “both edges”

Ensure the input level is not changing while the TAAnIOC2 register is modified.

9.6.6 TAAnIOC4 - TAA I/O control register 3

The TAAnIOC4 register is an 8-bit register that controls the output function of Timer AA.

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

Address <base> + C_H

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

7	6	5	4	3	2	1	0
0	0	0	0	TAAnOS1	TAAnOR1	TAAnOS0	TAAnOR0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

TAAnOS1	TAAnOR1	Toggle Control of TOAAn1
0	0	Standard operation.
0	1	Force output level to inactive at next toggle event
1	0	Force output level to active at next toggle event
1	1	Freeze current output level.

Note
1. After forcing the output level to either active or inactive, the TOAAn1 maintains this level (i.e., no toggling afterwards) until the TAAnOS1 and TAAnOR1 are cleared to standard operation.
2. The forcing of an output level occurs at the time of the next toggle event, while the freeze becomes effective immediately.

TAAnOS0	TAAnOR0	Toggle Control of TOAAn0
0	0	Standard operation.
0	1	Force output level to inactive at next toggle event
1	0	Force output level to active at next toggle event
1	1	Freeze current output level.

Note
1. After forcing the output level to either active or inactive, the TOAAn1 (TOAAn0) maintains this level (i.e. no toggling afterwards) until the TAAnOS1 (TAAnOS0) and TAAnOR1 (TAAnOR0) are cleared to standard operation.
2. The forcing of an output level is executed at the time of the next upcoming toggle event, while the freeze becomes effective immediately.
3. Writing to TAAnIOC4 is also possible, when TAAnCTL0.TAAnCE = 1.
4. The TAAnIOC4 register can be used in the interval mode or the free running mode. In other modes set this register to 00_H.
5. In the free running mode, new settings for the TAAnIOC4 register become effective only if the compare function is selected. When the capture function is selected, this register is invalid.

9.6.7 TAAnOPT0 - TAA option register 0

The TAAnOPT0 register is an 8-bit register used to set the capture/compare operation and detect 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 cleared by any reset.

7	6	5	4	3	2	1	0
0	0	TAAnCCS1	TAAnCCS0	0	0	0	TAAnOVF
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

TAAnCCS1	TAAnCCR1 register capture/compare selection
0	Compare register selection
1	Capture register selection
The TAAnCCS1 bit setting is valid only in the free-running mode.	

TAAnCCS0	TAAnCCR0 register capture/compare selection
0	Compare register selection
1	Capture register selection
The TAAnCCS0 bit setting is valid only in the free-running mode.	

TAAnOVF	Timer AA overflow detection
Set (1)	Overflow occurrence
Reset (0)	TAAnOVF bit write or TAAnCE = 0
• The TAAnOVF bit is set when the 16-bit counter value overflows from $FFFF_H$ to 0000_H in the free-running mode and the pulse width measurement mode. • An interrupt request signal (INTTAAnOV) is generated as soon as TAAnOVF bit is set (1). The INTTAAnOV signal is not generated in any mode other than free-running mode and the pulse width measurement mode. • The TAAnOVF bit is not cleared even when the TAAnOVF bit and the TAAnOPT0 register are read when TAAnOVF = 1. • The TAAnOVF bit can be both read and written, but 1 cannot be written to the TAAnOVF bit from the CPU. Writing 1 has no influence on timer AA operation.	

Caution Rewrite TAAnCCS1 and TAAnCCS0 bits when TAAnCE = 0 (the same value can be written when TAAnCE = 1). If rewriting was mistakenly performed, clear TAAnCE to 0 and then set the bits again.

9.6.8 TAAnOPT1 - TAA option register 1

The TAAnOPT1 register is an 8-bit register used to set the 32-bit capture mode by cascading two Timer AA channels.

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

Address <base> + D_H

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

7	6	5	4	3	2	1	0
TAAnCSE	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

TAAnCSE	TAAnCCR1 register capture/compare selection
0	16-bit non-cascaded mode
1	Set 32-bit cascaded capture mode. Timer AAn becomes the upper 16-bit and slave. The master timer is TAAm with m=n-1.

Note
1. When setting TAAnCSE, the timer becomes the upper 16-bit of a 32-bit timer.
2. Cascading is only available for capture with free-running counter.
3. The following pair of timers can be cascaded:
   - TAA2 and TAA3
     (TAA2 will become master and will hold the lower 16-bit value)

The table below shows the effects of the TAAnCSE flag on the timer operation:

	TAAnCSE=0	TAnCSE=1
Operating clock	macro clock from clock tree	macro clock of TAAm
Count Enable	TAAnCE bit of TAAnCTL0	TAAmCE bit of TAAm
Count Clock	selected by TAAnCKS[2:0]	Counter overflow from TAAm
Capture Signal 0	TIAAn0 input with edge filter as selected by TAAnIS[1:0]	TIAAm0 with edge filter selected for TAAm
Capture Signal 1	TIAAn1 input with edge filter as selected by TAAnIS[3:2]	TIAAm1 with edge filter selected for TAAm
Capture Interrupt	INTTAAnCC0 or INTTAAnCC1	INTTAAmCC0 or INTTAAmCC1

n= 3; m= 2

For details on the 32-bit capture mode, please refer to *Section 9.7.9, 32-bit Capture in free-running cascade mode* on page 211.

9.6.9 TAAnCCR0 - TAA capture/compare register 0

The TAAnCCR0 register is a 16-bit register that operates either as capture register or as a compare register.

In free-running mode, this register can be used as a capture register or as a compare register specified by bit TAAnOPT0.TAAnCCS0.

In the pulse width measurement mode, this register can be used only as a capture register (the compare function cannot be used.)

In all modes other than free-running mode and pulse width measurement mode, this register is used as a compare register.

Access This register can be read/written in 16-bit units.

Address <base> + 6_H

Initial Value 0000_H. This register is cleared by any reset. After a $\overline{\text{RESET}}$, TAAnCCR0 register default status is compare register.

TAAnCCR0 is also cleared to 0000_H if the internal operation clock is disabled by TAAnCTL0.TAAnCE = 0.

Caution When external event counter mode is used, do not set TAAnCCR0 register to 0000_H.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	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/W

- When used as a compare register, TAAnCCR0 can be rewritten when TAAnCE = 1, as shown below:

TAA operation mode	TAAnCCR0 register writing method
PWM mode, external trigger pulse output mode	Reload
Free-running mode, external event count mode, one-shot pulse mode, interval timer mode	Any time write
Pulse width measurement mode	Not applicable (used as capture register)

- When used as capture register, the count value is stored in TAAnCCR0 upon capture trigger (TIAAn0) input edge detection.

Note The value of the TAAnCCR0 register can be read/written when TAAnCE bit of TAAn control register 0 (TAAnCTL0) equals 1.

9.6.10 TAAnCCR1 - TAA capture/compare register 1

The TAAnCCR1 register is a 16-bit register that operates either both as a capture register or as a compare register.

In free-running mode, this register can be used as a capture register or as a compare register specified by bit TAAnOPT0.TAAnCCS1.

In the pulse width measurement mode, this register can be used only as a capture register (the compare function cannot be used.)

In all modes other than free-running mode and pulse width measurement mode, this register is used as a compare register.

Access This register can be read/written in 16-bit units.

Address <base> + 8_H

Initial Value 0000_H. This register is cleared by any reset. After a $\overline{\text{RESET}}$, TAAnCCR1 register default status is compare register.

TAAnCCR1 is also cleared to 0000_H if the internal operation clock is disabled by TAAnCTL0.TAAnCE = 0.

Caution When external event counter mode is used, do not set TAAnCCR1 register to 0000_H.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	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/W

- When used as a compare register, TAAnCCR1 can be rewritten when TAAnCE = 1, as below mentioned.

TAA operation mode	TAAnCCR1 register writing method
PWM mode, external trigger pulse output mode	Reload
Free-running mode, external event count mode, one-shot pulse mode, interval timer mode	Any time write
Pulse width measurement mode	Not applicable (used as capture register)

- When used as a capture register

  The count value is stored in TAAnCCR1 upon capture trigger (TIAAn1) input edge detection.

Note The value of the TAAnCCR1 register can be read/written when TAAnCE bit of TAAn control register 0 (TAAnCTL0) equals 1.

9.6.11 TAAnCNT - TAA counter read buffer register

TAAnCNT register is a read buffer register that can read 16-bit counter values.

This register is read-only, using a 16-bit memory manipulation instruction.

When TAAnCE bit of TAAnCTL0 equals 0, 0000_H is read from this register.

The value of the register is read when TAAnCE bit = 1.

Access This register can be read in 16-bit units.

Address <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
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

9.7 Operation

Timer AA can perform the following operations when not in cascade mode:

Operation	TAAnEST Software trigger bit	TIAAn0 External trigger input	TAAnEEE Count clock selection	Capture/Compare Selection	Compare Write
Interval timer mode	Invalid	Invalid	Internal/TIAAn0 pin	Compare only	Any time write
External event counter mode	Invalid	Invalid	External only	Compare only	Any time write
External trigger pulse output mode[Note 1]	Valid	Valid	Internal only	Compare only	Reload
One-shot pulse output mode[Note 2]	Valid	Valid	Internal only	Compare only	Any time write
PWM mode	Invalid	Invalid	Internal/TIAAn0 pin	Compare only	Reload
Free-running mode	Invalid	Invalid	Internal/TIAAn0 pin	Capture/compare selectable	Any time write
Pulse width measurement mode[Note 1,2]	Invalid	Invalid	Internal only	Capture only	Not applicable

Note 1. To use the external trigger pulse output mode, one-shot pulse mode, or pulse width measurement mode, select a count clock by clearing the TAAnEEE bit of the TAAnCTL1 register to 0.

2. When the external event count function is used, set the edge detection of the TIAAn0 capture input to "No edge detection" (TAAnIS1 and TAAnIS0 bits of TAAnIOC1 register to "00").

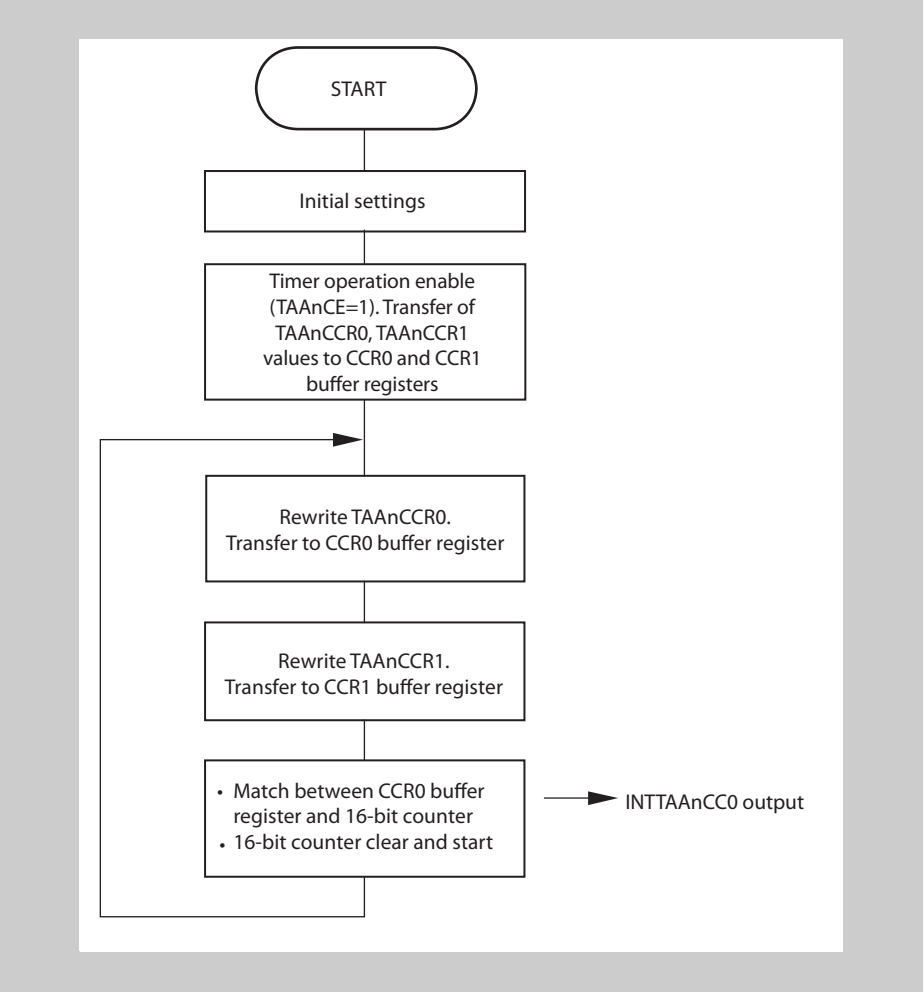
9.7.1 Anytime write and reload

TAAnCCR0 and TAAnCCR1 register rewrite is possible for timer AA during timer operation (TAAnCE = 1), but the write method (anytime write, reload) differs depending on the mode.

(1) Anytime write

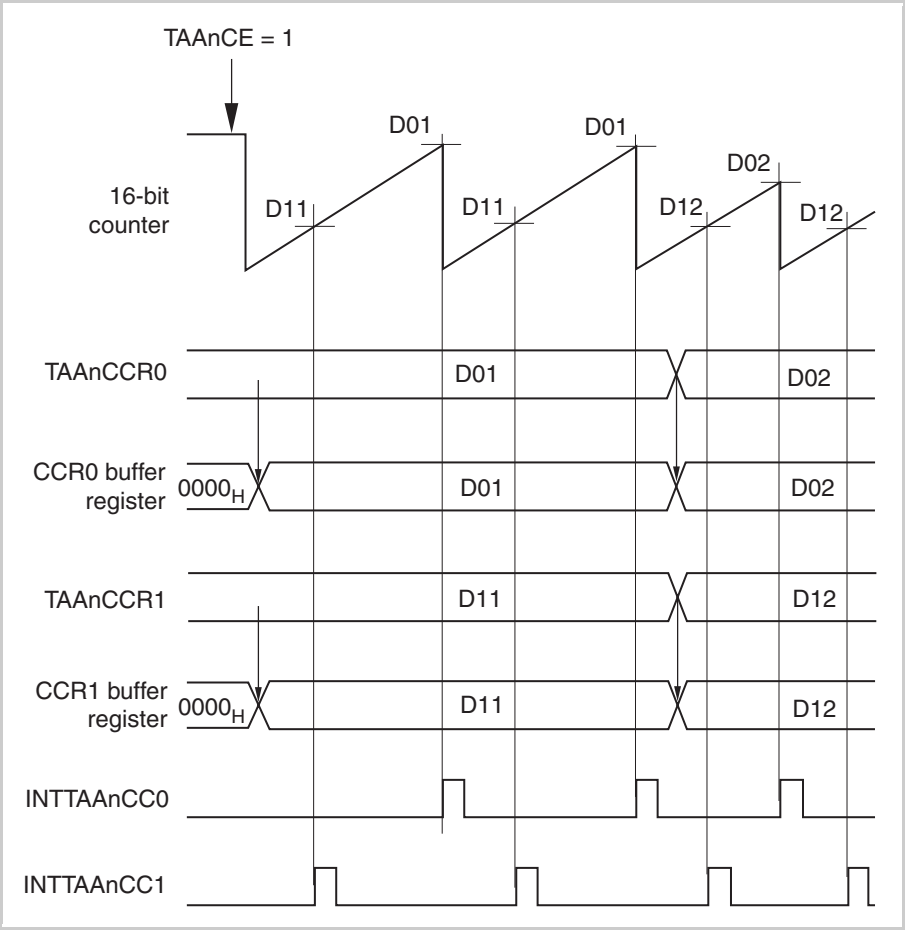
When data is written to the TAAnCCRm register during timer operation, it is transferred at any time to CCRm buffer register and used as the 16-bit counter comparison value.

Figure 9-3 Flowchart of basic operation for anytime write



Note 1. This flowchart illustrates an example of the operation in the interval timer mode.

Figure 9-4 Timing diagram for anytime write



D01, D02: Setting values of TAAnCCR0 register (0000_H to $FFFF_H$)
D11, D12: Setting values of TAAnCCR1 register (0000_H to $FFFF_H$)

This timing chart illustrates an example of the operation in the interval timer mode.

(2) Reload

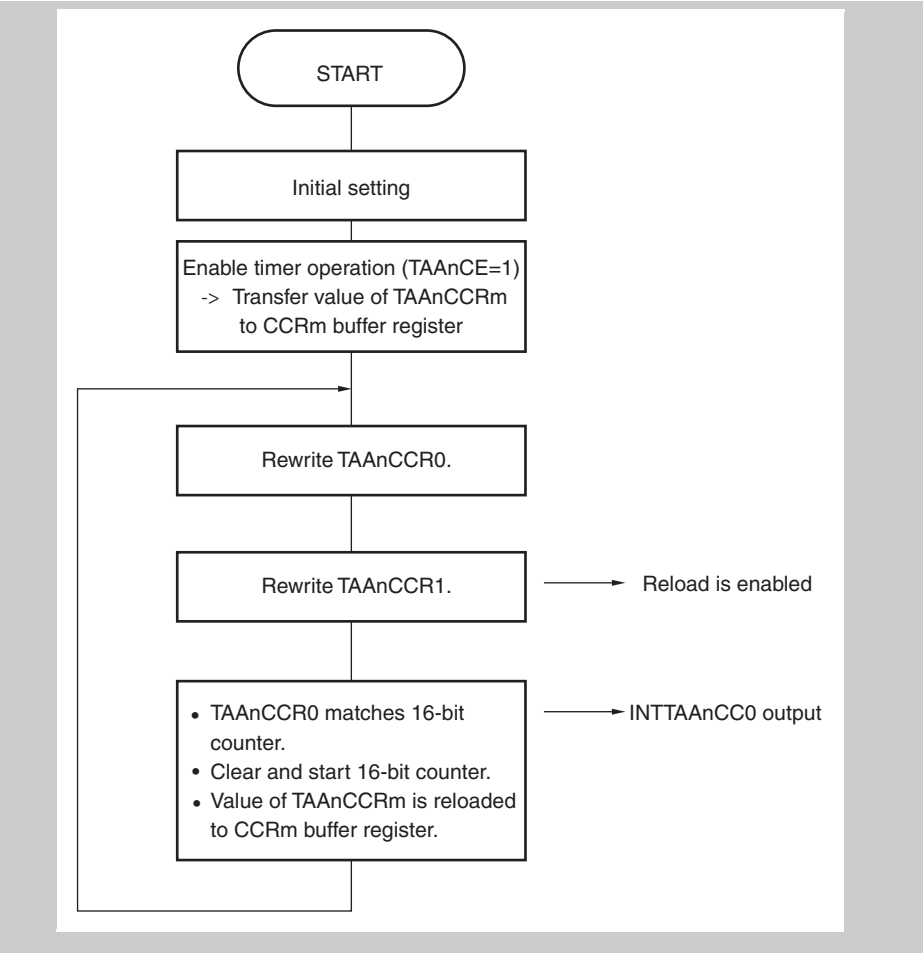
When data is written to the TAAnCCR0 and TAAnCCR1 registers during timer operation, it is compared with the value of the 16-bit counter via the CCRm buffer register. The values of the TAAnCCR0 and TAAnCCR1 registers can be rewritten when TAAnCE = 1.

So that the set values of the TAAnCCR0 and TAAnCCR1 registers are compared with the value of the 16-bit counter (the set values are reloaded to the CCRm buffer register), the value of the TAAnCCR0 register must be rewritten and then a value must be written to the TAAnCCR1 register before the value of the 16-bit counter matches the value of TAAnCCR0. When the value of the TAAnCCR0 register matches the value of the 16-bit counter, the values of the TAAnCCR0 and TAAnCCR1 registers are reloaded.

Whether the next reload timing is made valid is controlled by writing to the TAAnCCR1 register. Therefore, write the same value to the TAAnCCR1

register when it is necessary to rewrite the value of only the TAAnCCR0 register.

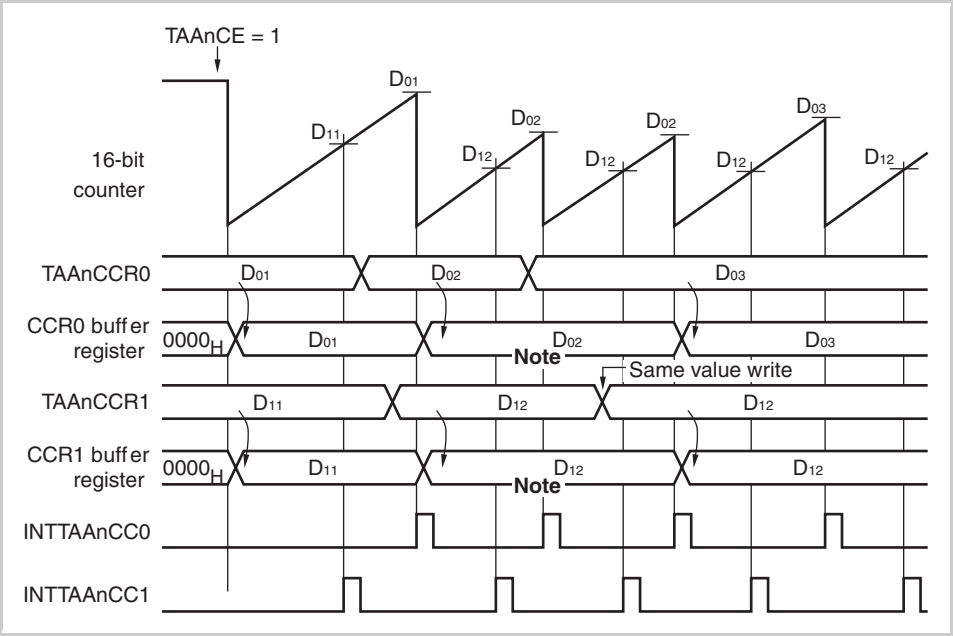
Figure 9-5 Flowchart of basic operation for reload



Caution Writing to the TAAnCCR1 register includes an operation to enable reload. Therefore, rewrite the TAAnCCR1 register after rewriting the TAAnCCR0 register.

Note 1. This flowchart illustrates an example of the PWM mode operation.

Figure 9-6 Timing chart for reload



Note Reload is not performed because TAAnCCR1 register is not written.

D01, D02, D03: Setting value of TAAnCCR0 register (0000_H to $FFFF_H$)
D11, D12: Setting value of TAAnCCR1 register (0000_H to $FFFF_H$)

This flowchart illustrates PWM mode operation.

9.7.2 Interval timer mode (TAAnMD2 to TAAnMD0 = 000_B)

In the interval timer mode, an interrupt request signal (INTTAAnCC0) is generated when the 16-bit counter is cleared and there is a match between the setting value of the TAAnCCR0 register and the value of the 16-bit counter. The TAAnCCR0 register can be rewritten when TAAnCE = 1, and when a value is set to TAAnCCR0 with a write instruction from the CPU, it is transferred to the CCR0 buffer register through anytime write mode, and is compared with the 16-bit counter value.

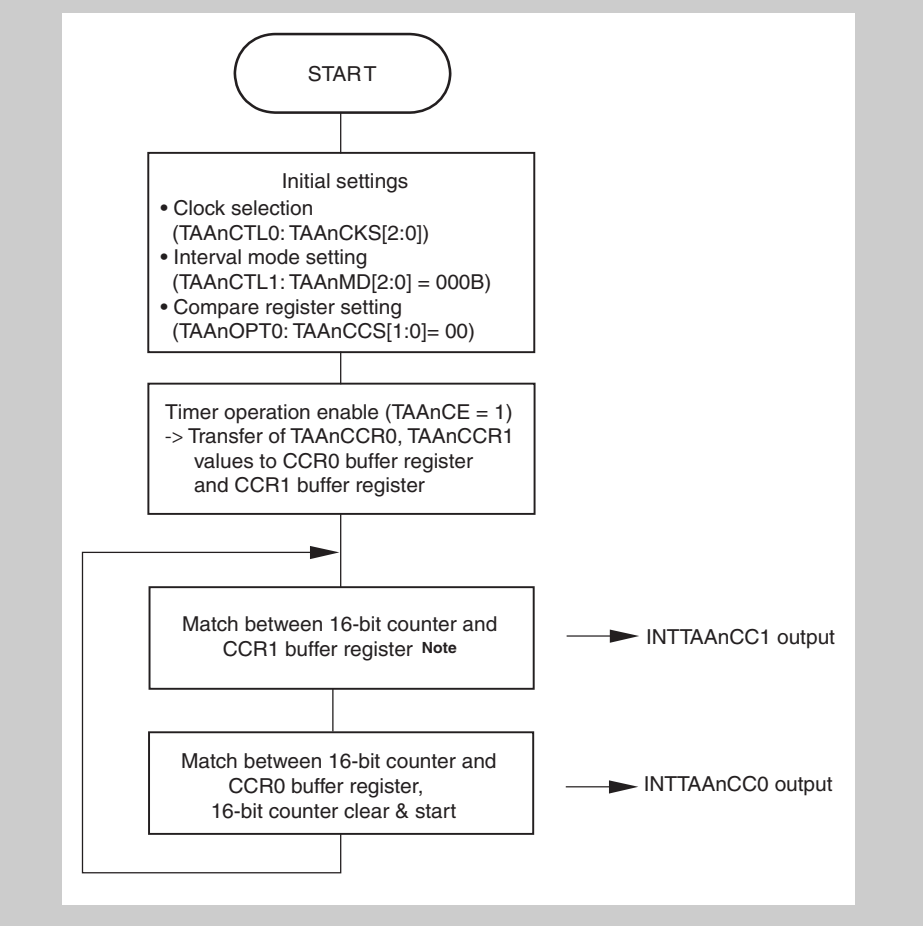
In the interval timer mode, the 16-bit counter is cleared only upon a match between the value of the 16-bit counter and the value of the CCR0 buffer register.

16-bit counter clearing using the TAAnCCR1 register is not performed. However, the setting value of the TAAnCCR1 register is transferred to the CCR1 buffer register and compared with the value of the 16-bit counter, and an interrupt request (INTTAAnCC1) is output if these values match. In addition, TOAA1n pin output is also possible by setting the TAAnOE1 bit to 1.

When the TAAnCCR1 register is not used, make $FFFF_H$ its setting value.

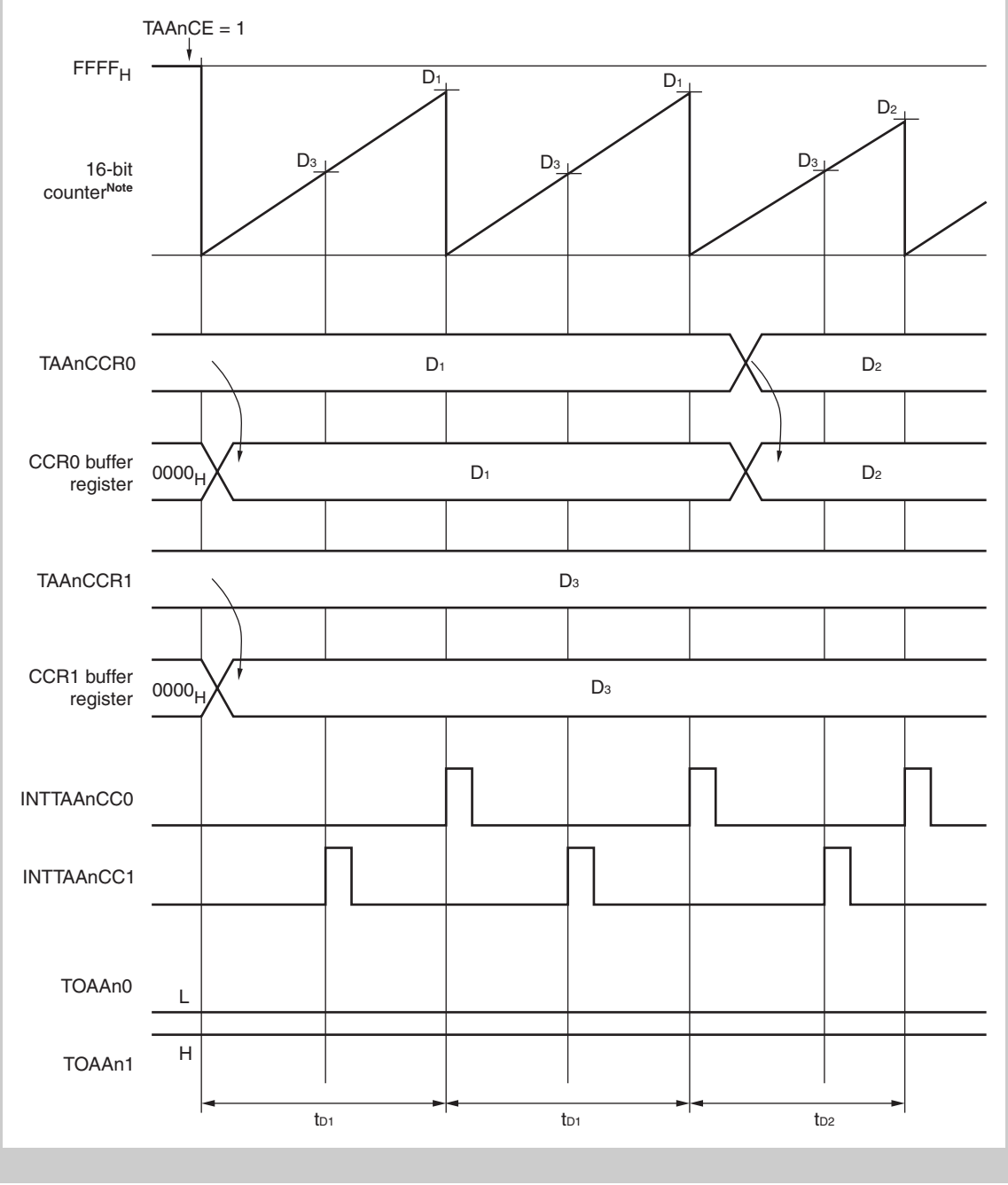
When performing timer output with the TOAAn1 pin, set the same values to both the TAAnCCR0 and the TAAnCCR1 registers, since the 16-bit timer counter cannot be cleared with the TAAnCCR1 register.

Figure 9-7 Flowchart of basic operation in interval timer mode



Note The 16-bit counter is not cleared when its value matches the value of TAAnCCR1.

Figure 9-8 Basic operation timing in interval timer mode When D1 > D2 > D3; only TAAnCCR0 register value is written, and TOAAn0 and TOAAn1 are not output (TAAnOE0 = 0, TAAnOE1 = 0, TAAnOL0 = 0, TAAnOL1 = 1)

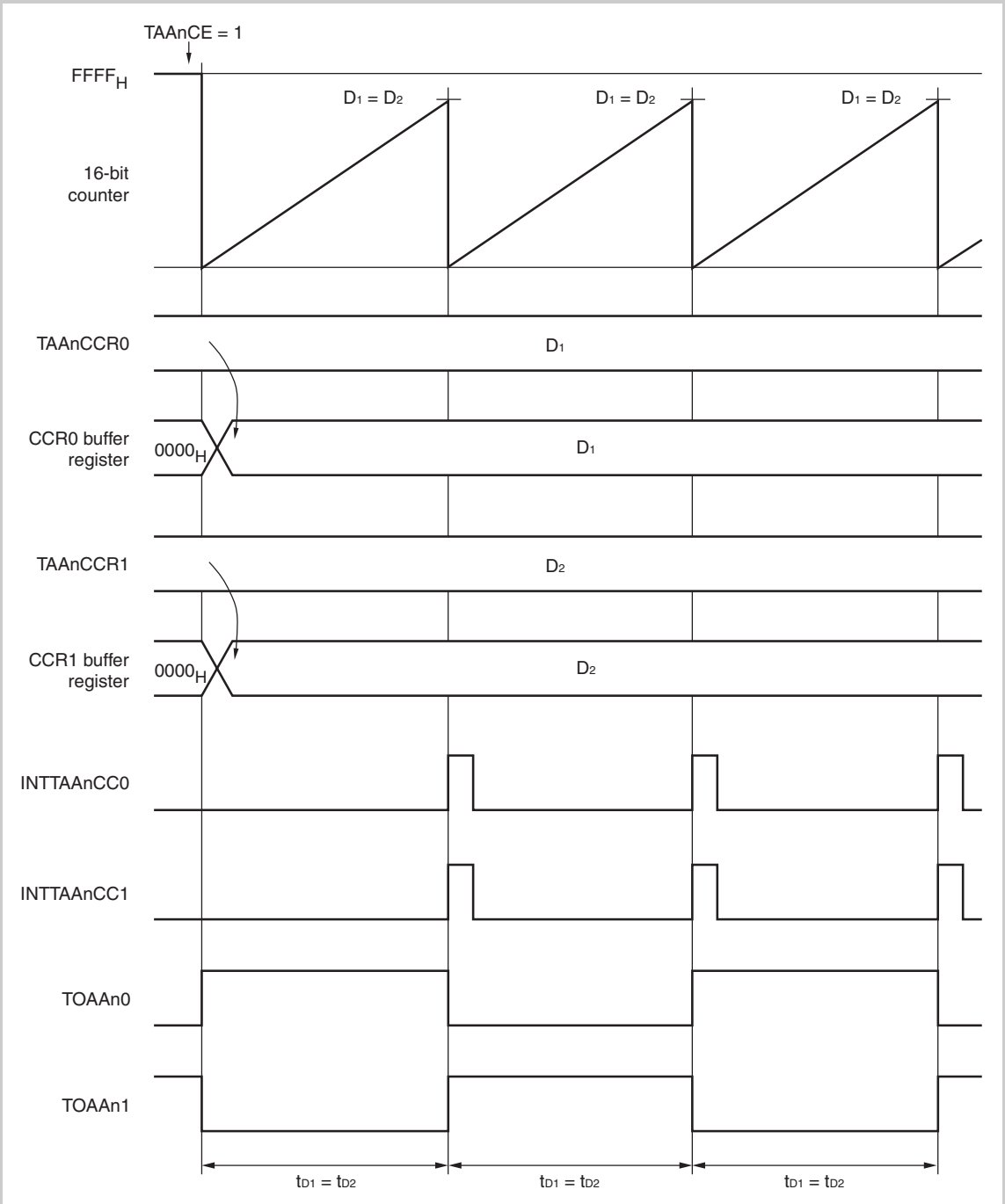


Note The 16-bit counter is not cleared when its value matches the value of TAAnCCR1.

D1, D2: Setting values of TAAnCCR0 register (0000_H to $FFFF_H$)
D3: Setting value of TAAnCCR1 register (0000_H to $FFFF_H$)

Interval time (t_{Dn})= (Dn + 1) × (count clock cycle)

Figure 9-9 Basic operation timing in interval timer mode When D1 = D2; TAAnCCR0 and TAAnCCR1 are not rewritten, and TOAAn0 and TOAAn1 are output (TAAnOE0 = 1, TAAnOE1 = 1, TAAnOL0 = 0, TAAnOL1 = 1)



D1: Setting value of TAAnCCR0 register (0000_H to $FFFF_H$)
D2: Setting value of TAAnCCR1 register (0000_H to $FFFF_H$)

Interval time (t_{Dn}) = (Dn + 1) × (count clock cycle)

9.7.3 External event counter mode (TAAnMD2 to TAAnMD0 = 001_B)

In the external event count mode, the external event count input (TEVTAAn pin input) is used as a count-up signal. Regardless of the setting of the TAAnEEE bit of the TAAnCTL0 register, the 16-bit timer/event counter AA counts up the external event count input (TEVTAAn pin input) when it is set in the external event count mode. In this mode, an interrupt request (INTTAAnCC0) is generated when the set value of the TAAnCCR0 register matches the value of the 16-bit counter, and the value of the 16-bit counter is cleared.

When a value is set to the TAAnCCR0 register with a write instruction from the CPU, it is transferred to the CCR0 buffer register through anytime write, and is compared with the 16-bit counter value.

In the external event counter mode, the 16-bit counter is cleared only upon a match between the value of the 16-bit counter and the value of the CCR0 buffer register.

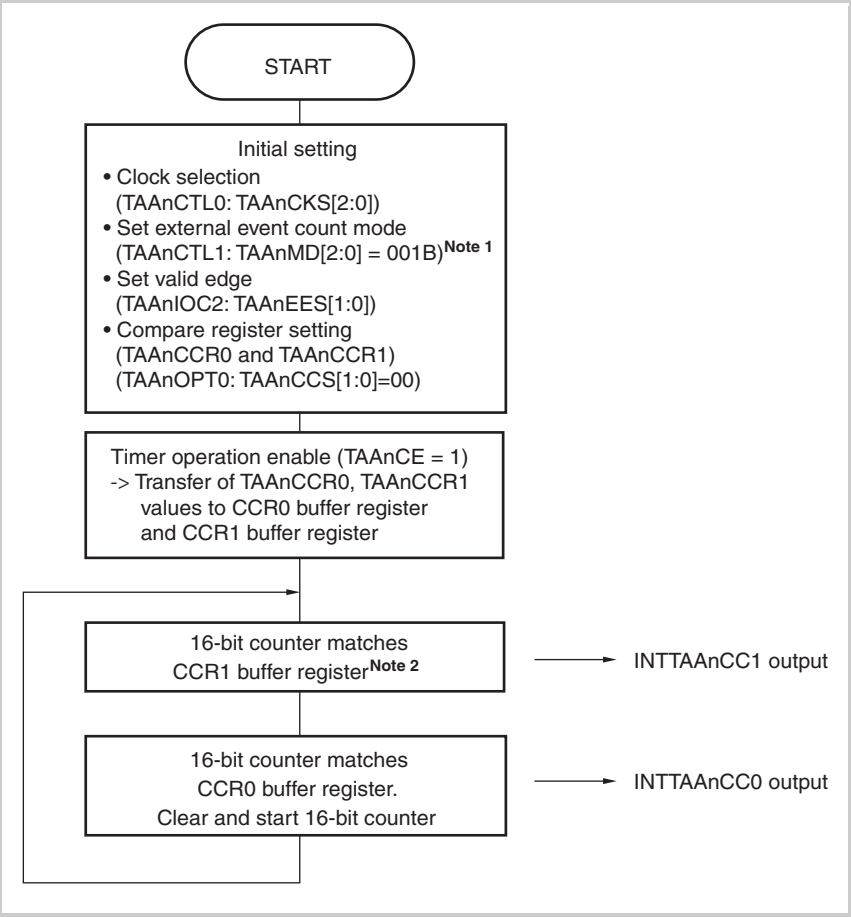
The 16-bit counter can not be cleared using TAAnCCR1 register. However, the setting value of the TAAnCCR1 register is transferred to the CCR1 buffer register and compared with the value of the 16-bit counter, and an interrupt request (INTTAAnCC1) is output if these values match.

Moreover, TOAAnm pin output is also possible by setting the TAAnOEm bit to 1.

When performing timer output with the TOAAn1 pin, set the same values to TAAnCCR0 register and TAAnCCR1 register since the 16-bit counter cannot be cleared with CCR1 buffer register.

The TAAnCCR0 register can be rewritten when TAAnCE = 1. When TAAnCCR1 register is not used, it is recommended to set TAAnCCR1 register to $FFFF_H$.

Figure 9-10 Flowchart of basic operation in external event counter mode

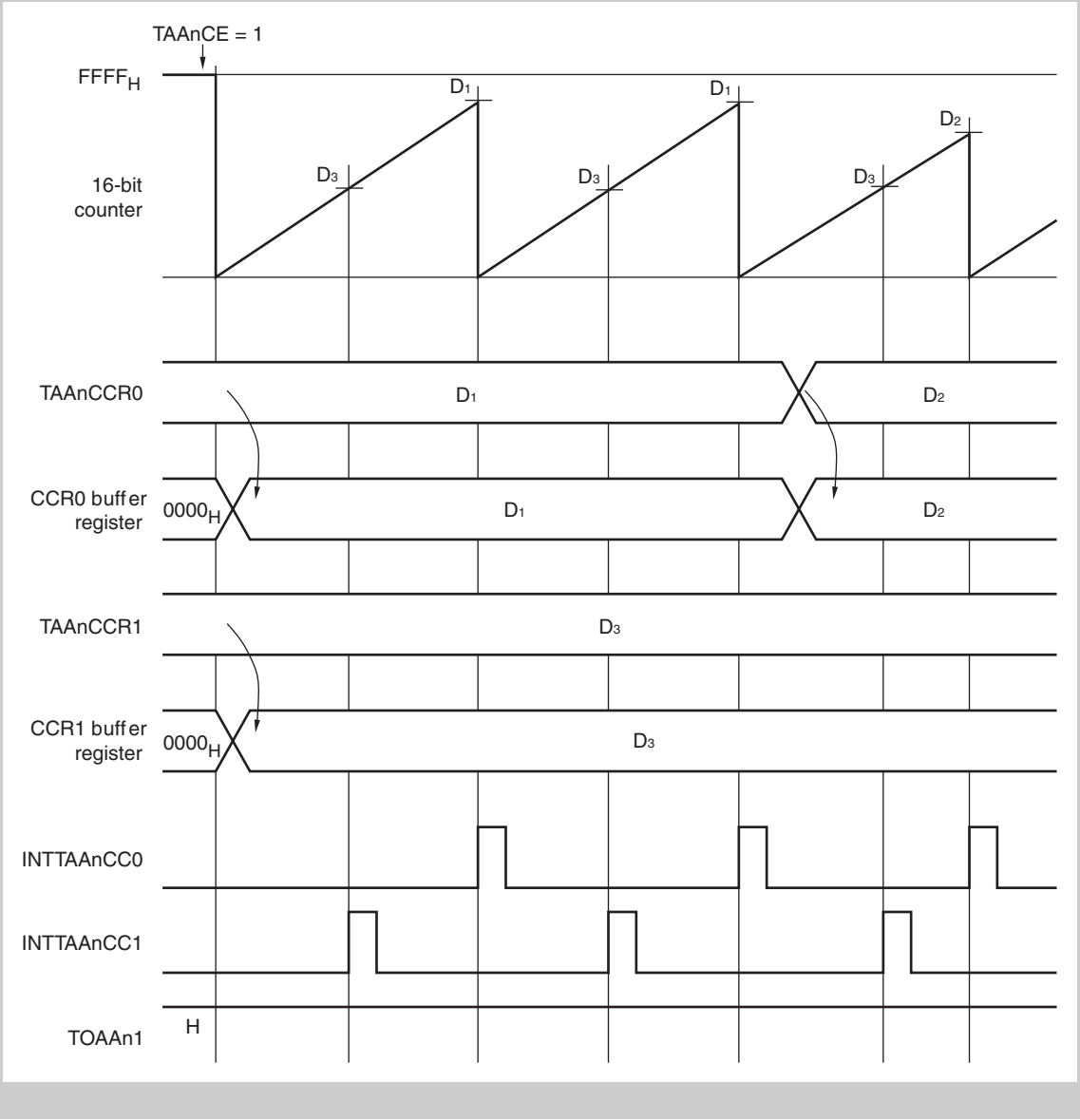


Note 1. Selection of the TAAnEEE bit has no influence.

2. The 16-bit counter is not cleared upon a match between the 16-bit counter and the CCR1 buffer register.

3. m=0,1

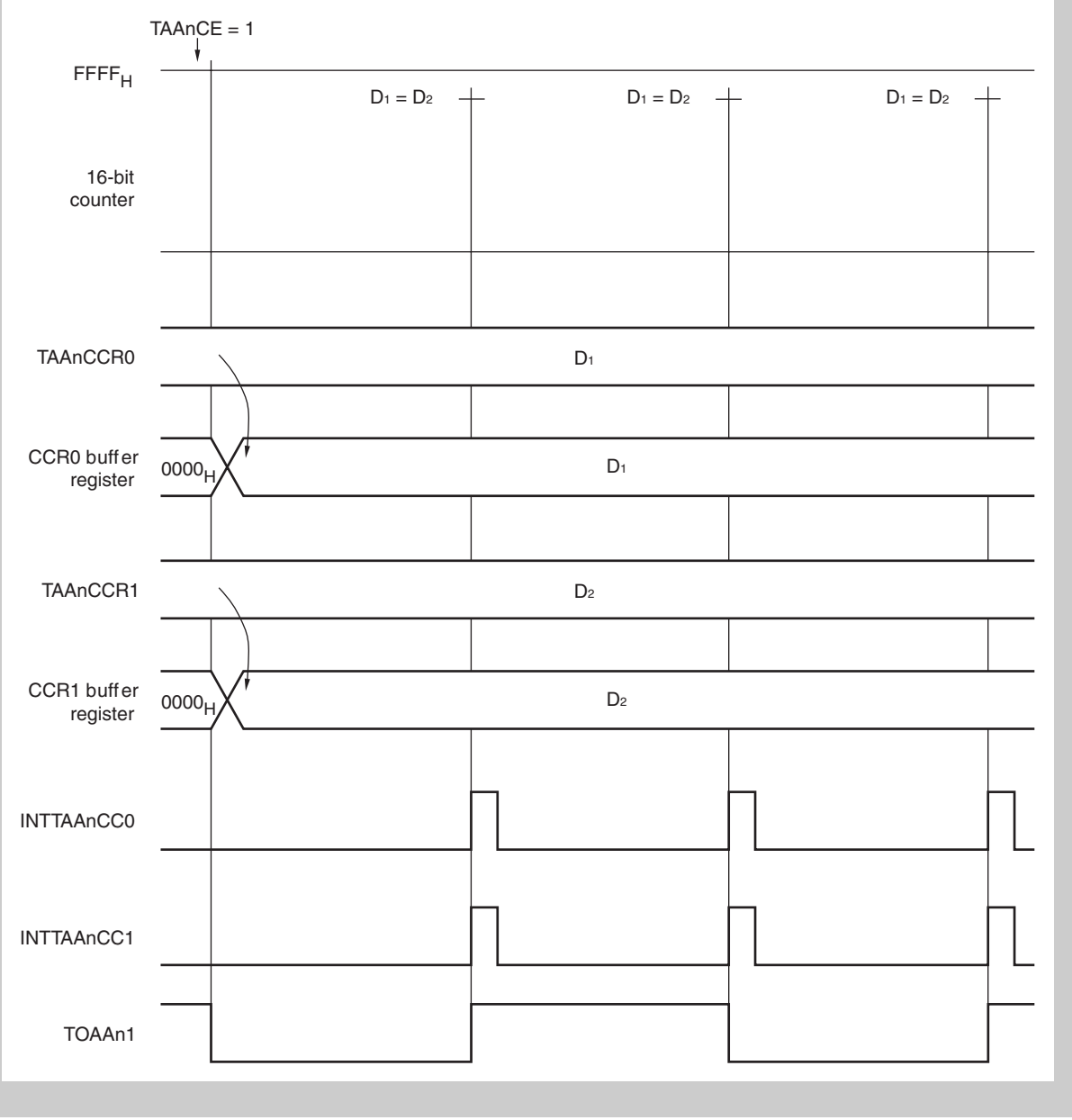
Figure 9-11 Basic operation timing in external event counter mode When D1 > D2 > D3; rewrite TAAnCCR0 only; TOAAn1 is not output (TAAnOE0 = 0, TAAnOE1 = 0, TAAnOL0 = 0, TAAnOL1 = 1)



D1, D2: Setting values of TAAnCCR0 register (0000_H to $FFFF_H$)
D3: Setting value of TAAnCCR1 register (0000_H to $FFFF_H$)

Number of event counts = (Dn + 1)

Figure 9-12 Basic operation timing in external event counter mode When D1 = D2; TAAnCCR0 and TAAnCCR1 are not rewritten, TOAAn1 is output (TAAnOE0 = 1, TAAnOE1 = 1, TAAnOL0 = 0, TAAnOL1 = 1)



D1: Setting value of TAAnCCR0 register (0000_H to $FFFF_H$)
D2: Setting value of TAAnCCR1 register (0000_H to $FFFF_H$)

Number of event count = (Dn + 1)

9.7.4 External trigger pulse mode (TAAnMD2 to TAAnMD0 = 010_B)

When TAAnCE = 1 in the external trigger pulse mode, the 16-bit counter stops at $FFFF_H$ and waits for a trigger condition (input of an external trigger (TIAAn0 pin input) or SW trigger by setting of TAAnEST bit). When the counter detects the trigger condition, it starts counting up. The duty factor of the signal output from the TOAAn1 pin is set by a reload register (TAAnCCR1) and the period is set by a compare register (TAAnCCR0).

Rewriting the TAAnCCR0 and TAAnCCR1 registers is enabled when TAAnCE = 1.

To ensure that the selected values of the TAAnCCR0 and TAAnCCR1 registers after rewriting are compared with the value of the 16-bit counter (reloaded to the CCRm buffer register), the TAAnCCR0 register and then the TAAnCCR1 register must be written before the value of the 16-bit counter matches the value of the TAAnCCR0 register.

When the value of the TAAnCCR0 register later matches the value of the 16-bit counter, the values of the TAAnCCR0 and TAAnCCR1 registers are reloaded to the CCRm buffer register.

Whether the next reload timing is made valid or not is controlled by writing to the TAAnCCR1 register. Therefore, write the same value to the TAAnCCR1 register when it is necessary to rewrite only the TAAnCCR0 register's value.

Reload is invalid when only the TAAnCCR0 register is rewritten. To stop timer AA, clear TAAnCE to 0. If the edge of the external trigger (TTRGAAn pin input) is detected more than once in the external trigger pulse mode, the 16-bit counter is cleared at the point of edge detection, and resumes counting up.

To realize the same function as the external trigger pulse mode by using a software trigger pulse mode instead of the external trigger input (TTRGAAn pin input), a software trigger is generated by setting the TAAnEST bit of the TAAnCTL1 register to 1.

When using a software trigger, a square wave that has one cycle of the PWM waveform as half of its own cycle can also be output from the TOAAn0 pin.

The waveform of the external trigger pulse is output from TOAAn1. A toggle output is produced from the TOAAn0 pin when the value of the TAAnCCR0 register matches the value of the 16-bit counter.

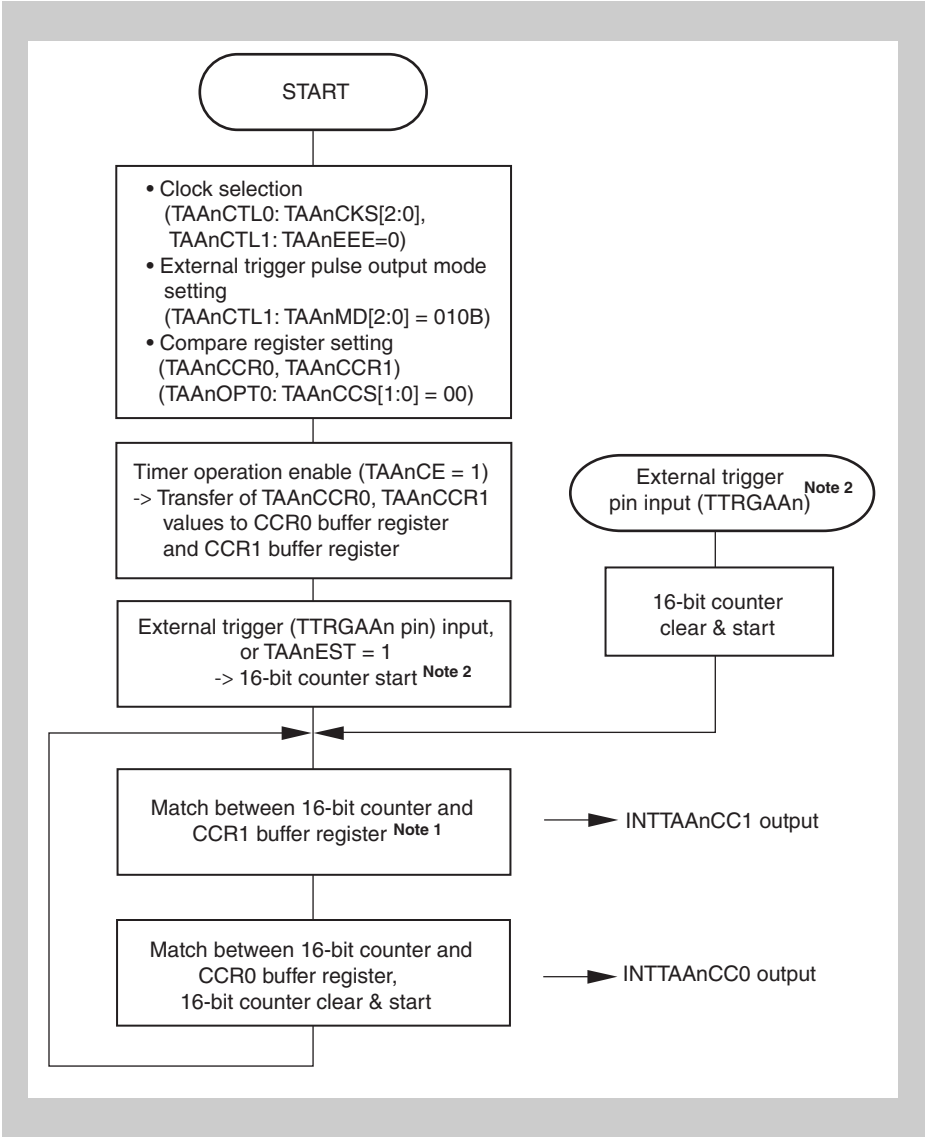
In the external trigger pulse mode, the capture function of the TAAnCCR0 and TAAnCCR1 registers cannot be used because they can only be used as compare registers.

Caution In the external trigger pulse mode, select the internal clock (TAAnEEE bit of TAAnCTL1 register = 0) for the count clock.

Note 1. For the reload operation when TAAnCCR0 and TAAnCCR1 are rewritten during timer operation, refer to *Section 9.7.6, PWM mode (TAAnMD2 to TAAnMD0 = 100_B)* on page 194.

2. m= 0,1

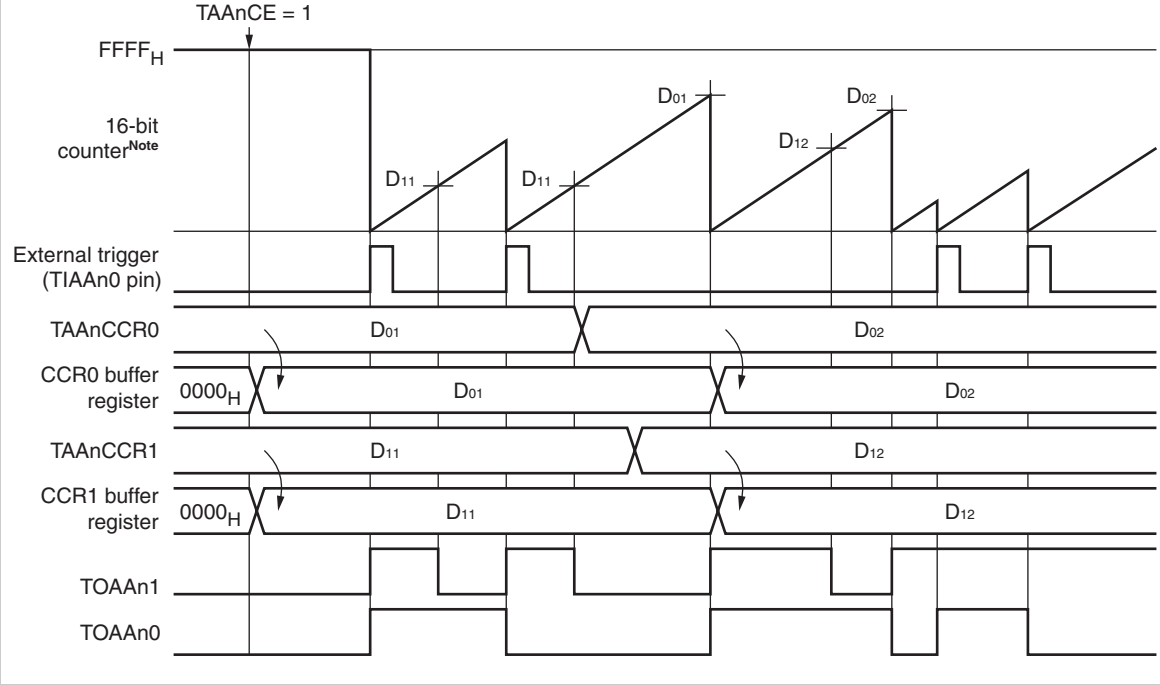
Figure 9-13 Flowchart of basic operation in external trigger pulse output mode



Note 1. The 16-bit counter is not cleared upon a match between the 16-bit counter and the CCR1 buffer register.

2. n = 0 to 3.

Figure 9-14 Basic operation timing in external trigger pulse output mode (TAAnOE0 = 1, TAAnOE1 = 1, TAAnOL0 = 0, TAAnOL1 = 0)



Note The 16-bit counter is not cleared when it matches the CCR1 buffer register.

D01, D02: Setting value of TAAnCCR0 register (0000_H to $FFFF_H$)
D11, D12: Setting value of TAAnCCR1 register (0000_H to $FFFF_H$)

Duty of TOAAn1 output =
(Set value of TAAnCCR1 register) / (Set value of TAA0CCR0 register+1)
Cycle of TOAAn1 output =
(Set value of TAAnCCR0 register + 1) x (Count clock cycle)

9.7.5 One-shot pulse mode (TAAnMD2 to TAAnMD0 = 011_B)

When TAAnCE is set to 1 in the one-shot pulse mode, the 16-bit counter waits for the setting of the TAAnEST bit (to 1) or a trigger that is input when the edge of the TTRGAAn pin is detected, while holding $FFFF_H$. When the trigger is input, the 16-bit counter starts counting up.

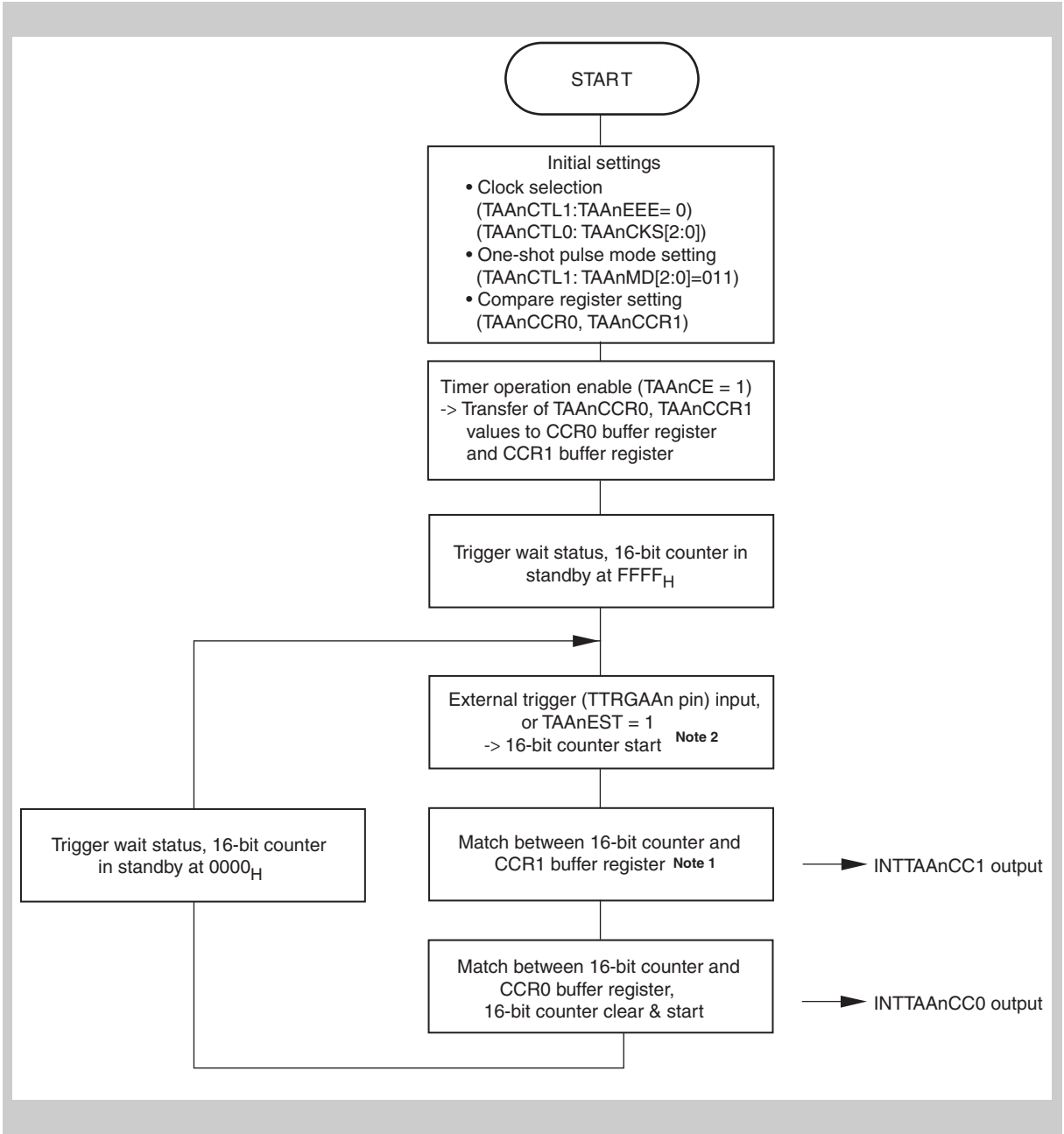
When the value of the 16-bit counter matches the value of the CCR1 buffer register that has been transferred from the TAAnCCR1 register, TOAAn1 goes high. When the value of the 16-bit counter matches the value of the CCR0 buffer register that has been transferred from the TAAnCCR0 register, TOAAn1 goes low, and the 16-bit counter is cleared to 0000_H and stops. Input of a second or subsequent trigger is ignored while the 16-bit counter is operating. Be sure to input a second trigger while the 16-bit counter is stopped at 0000_H. In the one shot pulse mode, rewriting the TAAnCCR0 and TAAnCCR1 registers is enabled when TAAnCE = 1. The set values of the TAAnCCR0 and TAAnCCR1 registers become valid after a write instruction from the CPU is executed. They are then transferred to the CCR0 and CCR1 buffer registers, and compared with the value of the 16-bit counter. The

waveform of the one-shot pulse is output from the TOAAn1 pin. The TOAAn0 pin produces a toggle output when the value of the 16-bit counter matches the value of the TAAnCCR0 register. In the one-shot pulse mode, the TAAnCCR0 and TAAnCCR1 registers function only as compare registers. They cannot be used as capture registers.

Cautions

1. In the one-shot pulse mode, select the internal clock (TAAnEEE bit of TAAnCTL1 register = 0) as the count clock.
2. In the one-shot pulse mode, it is prohibited to set the TAAnCCR1 register to 0000_H.

Figure 9-15 Flowchart of basic operation in one-shot pulse mode

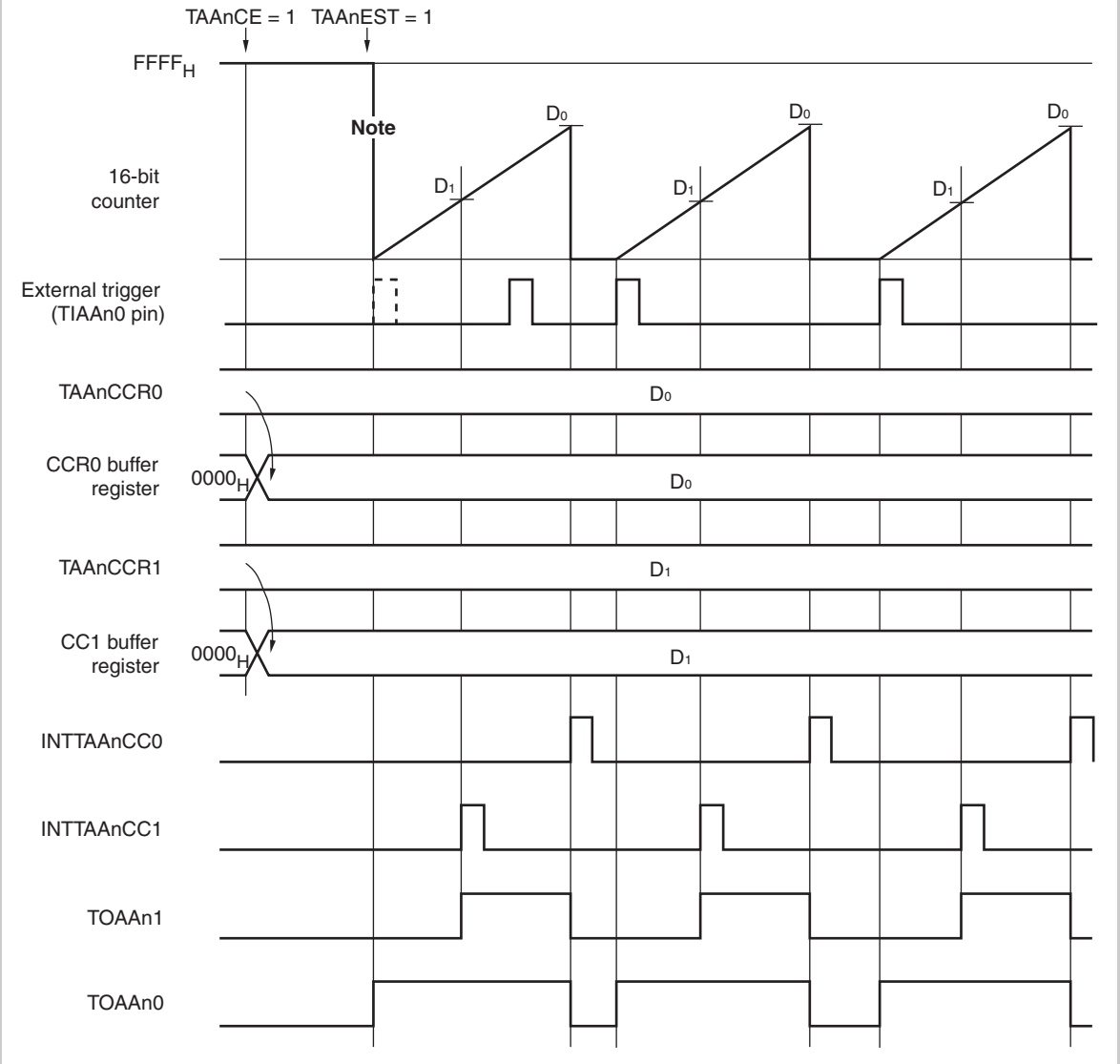


Note 1. Only the TAAnEST bit of the TAAnCTL1 register can be rewritten during the timer operation (TAAnCE = 1).

2. The 16-bit counter is not cleared upon a match between the values of the 16-bit counter and the CCR1 buffer register.

Caution The 16-bit counter is not cleared when a trigger input is performed during the count-up operation of the 16-bit counter.

Figure 9-16 Timing of basic operation in one-shot pulse mode (TAAnOE0 = 1, TAAnOE1 = 1, TAAnOL0 = 0, TAAnOL1 = 0)



Note The 16-bit counter starts counting up when either TAAnEST = 1 is set or the external trigger (TIAAn0) is input.

D0: Setting value of TAAnCCR0 register (0000_H to $FFFF_H$)
D1: Setting value of TAAnCCR1 register (0000_H to $FFFF_H$)

Active level period of TOAAn1 pin output is (setting value of TAAnCCR0 - setting value of TAAnCCR1 + 1) × count clock period

Output delay period = (setting value of TAAnCCR 1 register) × count clock period

9.7.6 PWM mode (TAAnMD2 to TAAnMD0 = 100_B)

In the PWM mode, TAAn capture/compare register 1 (TAAnCCR1) is used to set the duty factor and TAAn capture/compare register 0 (TAAnCCR0) is used to set the cycle. By using these two registers and operating the timer, variable-duty PWM is output.

Rewriting the TAAnCCR0 and TAAnCCR1 registers is enabled when TAAnCE = 1.

So that the set values of the TAAnCCR0 and TAAnCCR1 registers are compared with the value of the 16-bit counter (reloaded to the CCR0 and CCR1 buffer registers), the TAAnCCR0 register must be rewritten and then a value must be written to the TAAnCCR1 register before the value of the 16-bit counter matches the value of the TAAnCCR0 register.

The values of the TAAnCCR0 and TAAnCCR1 registers are reloaded when the value of the TAAnCCR0 register later matches the value of the 16-bit counter. Whether the next reload timing is made valid or not is controlled by writing to the TAAnCCR1 register. Therefore, write the same value to the TAAnCCR1 register even when only the value of the TAAnCCR0 register needs to be rewritten. Reload is invalid when only the value of the TAAnCCR0 register is rewritten.

To stop Timer AA, clear TAAnCE to 0.

The waveform of PWM is output from the TOAAn1 pin. The TOAAn0 pin produces a toggle output when the 16-bit counter matches the TAAnCCR0 register.

In the PWM mode, the TAAnCCR0 and TAAnCCR1 registers are used only as compare registers. They cannot be used as capture registers.

Figure 9-17 Flowchart of basic operation in PWM mode When values of TAAnCCR0, TAAnCCR1 registers are not rewritten during timer operation

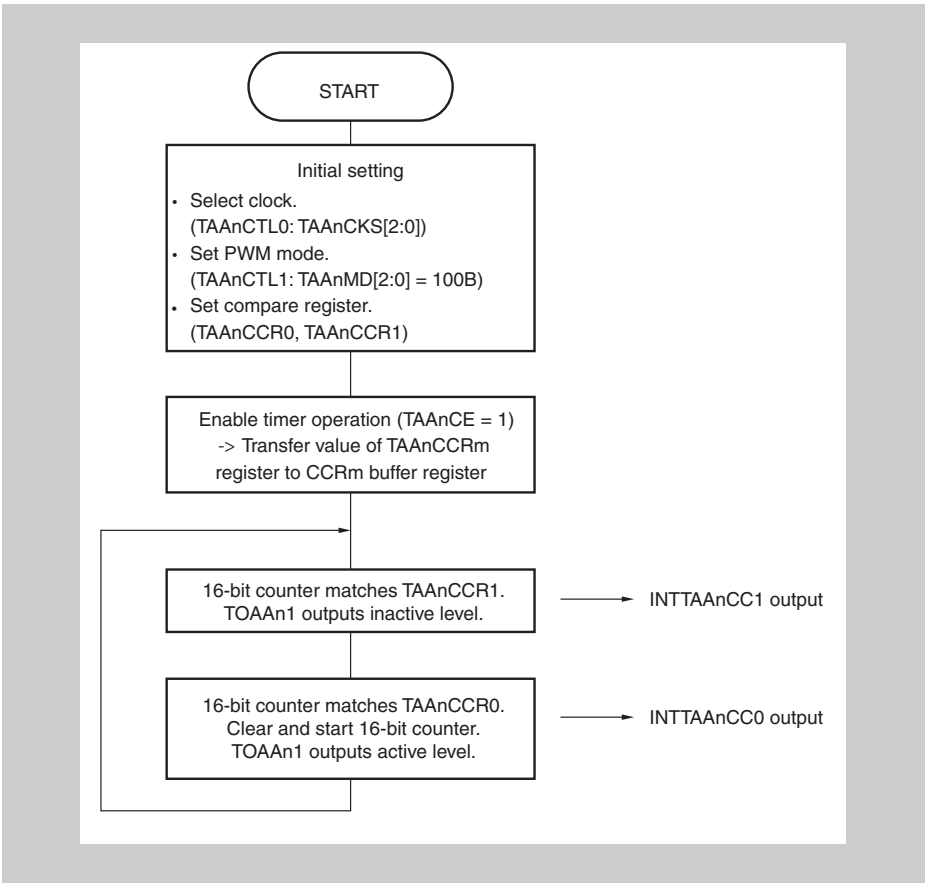
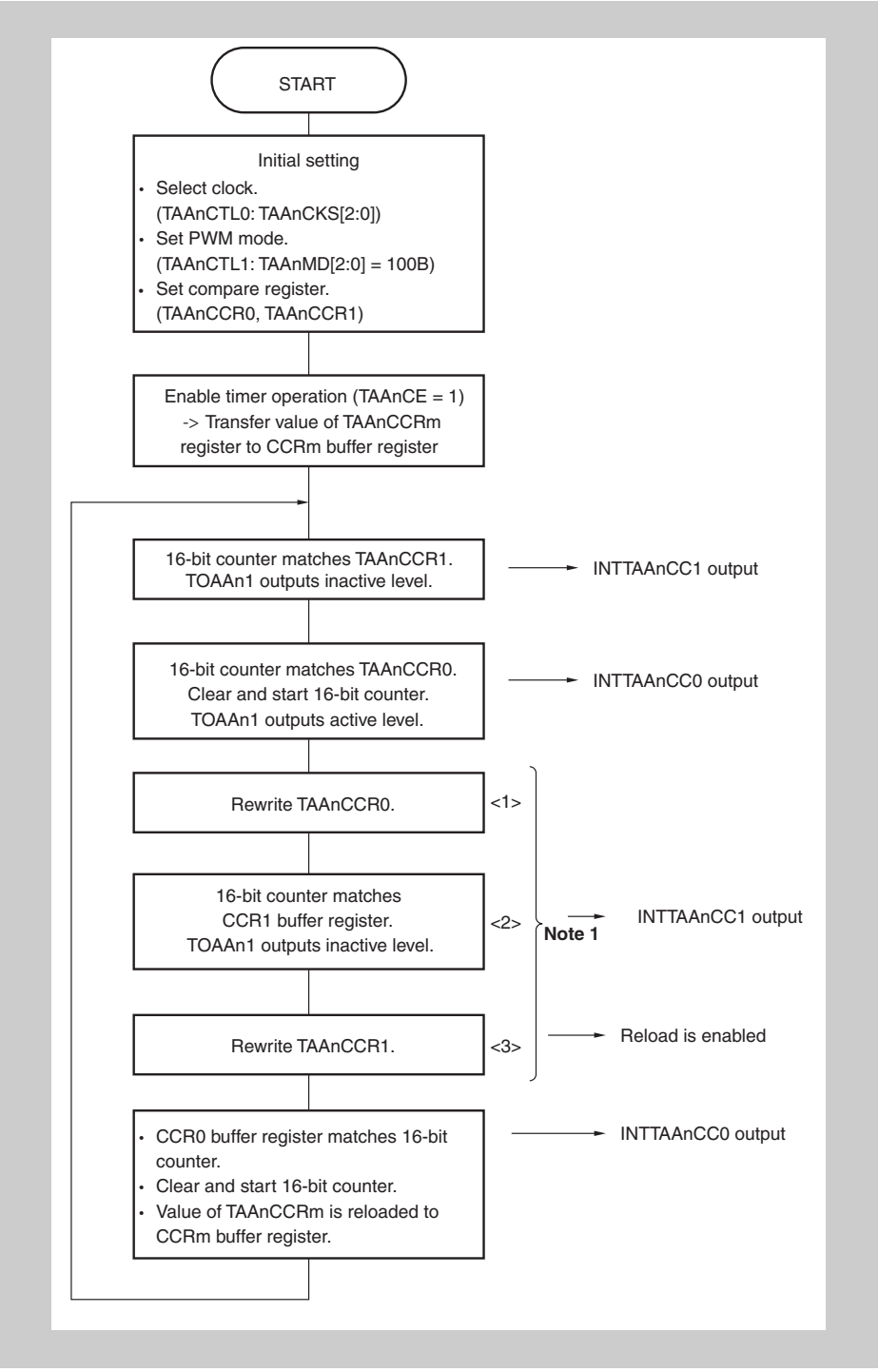
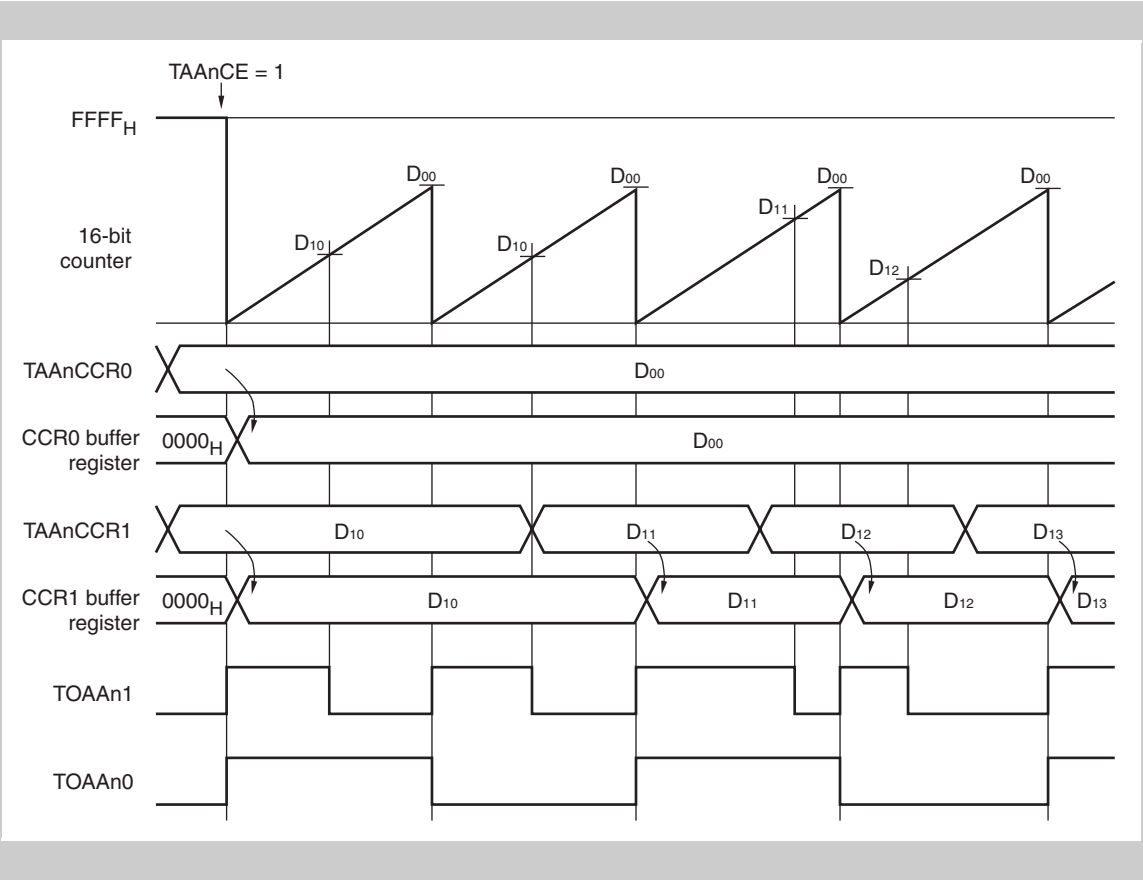


Figure 9-18 Flowchart of basic operation in PWM mode When values of TAAnCCR0, TAAnCCR1 registers are rewritten during timer operation



Note 1. The timing of <2> in the above flowchart may differ depending on the rewrite timing of steps <1> and <3> and the value of TAAnCCR1, but make sure that step <3> comes after step <1>.

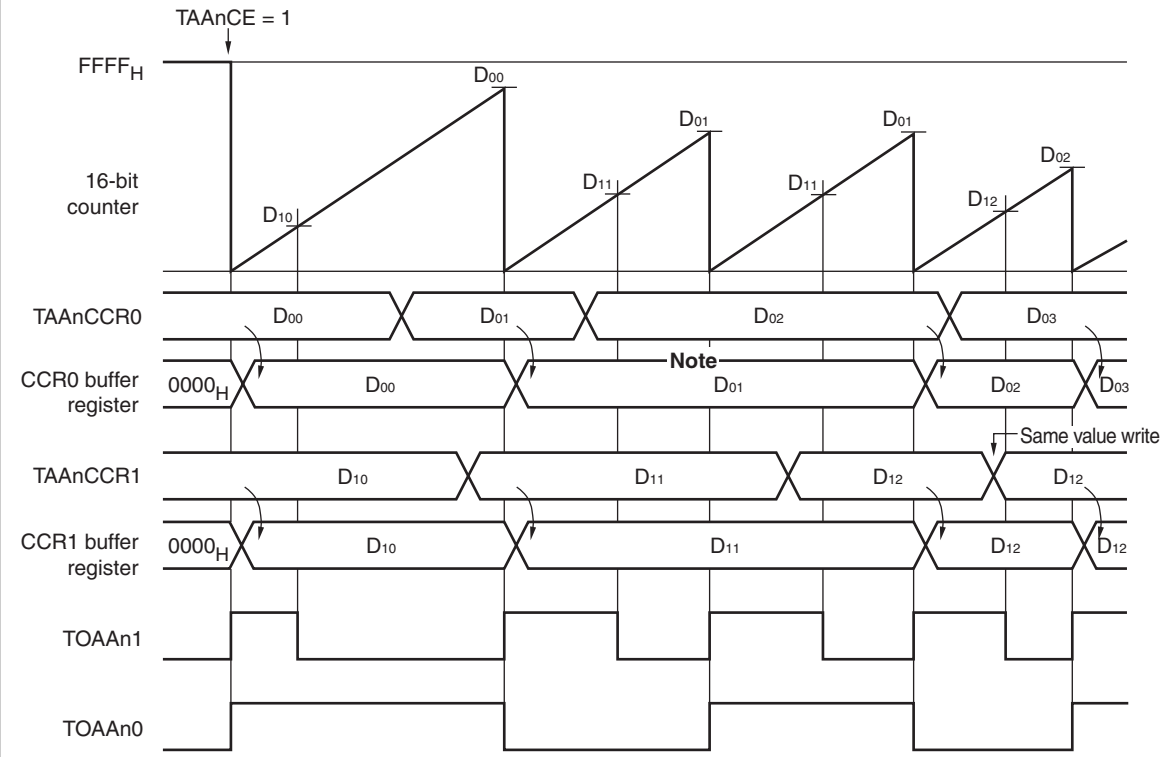
Figure 9-19 Basic operation timing in PWM mode When rewriting TAAnCCR1 value (TAAnOE0 = 1, TAAnOE1 = 1, TAAnOL0 = 0, TAAnOL1 = 0)



D00: Set value of TAAnCCR0 register (0000_H to $FFFF_H$)
D10, D11, D12, D13: Set value of TAAnCCR1 register (0000_H to $FFFF_H$)

Duty of TOAAn1 output =
(Set value of TAAnCCR1 register) / (Set value of TAA0CCR0 register + 1)
Cycle of TOAAn1 output =
(Set value of TAAnCCR0 register + 1) x (Count clock cycle)
Toggle width of TOAAn0 output =
(Set value of TAAnCCR0 register + 1) x (Count clock period)

Figure 9-20 Basic operation timing in PWM mode When TAAnCCR0, TAAnCCR1 values are rewritten (TAAnOE0 = 1, TAAnOE1 = 1, TAAnOL0 = 0, TAAnOL1 = 0)



Note 1. Reload is not performed because the TAAnCCR1 register was not rewritten.

D00, D01, D02, D03: Setting values of TAAnCCR0 register (0000_H to $FFFF_H$)
D10, D11, D12, D13: Setting values of TAAnCCR1 register (0000_H to $FFFF_H$)
Duty of TOAAn1 output =
(Set value of TAAnCCR1 register) / (Set value of TAA0CCR0 register + 1)
Cycle of TOAAn1 output =
(Set value of TAAnCCR0 register + 1) x (Count clock cycle)
Toggle width of TOAAn0 output =
(Set value of TAAnCCR0 register + 1) x (Count clock cycle)

2. To output a 0% duty PWM signal set the TAAnCCR1 register to 0.
To output a 100% duty PWM signal set the TAAnCCR1 register to the value of the CCR0 register +1. Do not set a value of $FFFF_H$ to the CCR1 register.

9.7.7 Free-running mode (TAAnMD2 to TAAnMD0 = 101_B)

In the free-running mode, both the interval function and the compare function can be realized by operating the 16-bit counter as a free-running counter and selecting capture/compare operation with the TAAnCCS1 and TAAnCCS0 bits.

The settings of the TAAnCCS1 and TAAnCCS0 bits of the TAAnOPT0 register are valid only in the free-running mode.

TAAnCCS1	Operation
0	Use TAAnCCR1 register as compare register
1	Use TAAnCCR1 register as capture register

TAAnCCS0	Operation
0	Use TAAnCCR0 register as compare register
1	Use TAAnCCR0 register as capture register

- Using TAAnCCR1 register as compare register

  An interrupt is output when the 16-bit counter matches the CCR1 buffer register in the free-running mode.

  Rewrite during compare timer operation is enabled and performed with anytime write mode. (Once the compare value has been written, the internal clock is synchronized and this value is used as the 16-bit counter comparison value.)

  When timer output (TOAAn1) has been enabled, TOAAn1 performs toggle output when the 16-bit counter matches the CCR1 buffer register.
- Using TAAnCCR1 register as capture register

  The value of the 16-bit counter is saved to the TAAnCCR1 register upon TIAAn1 pin edge detection.
- Using TAAnCCR0 register as compare register

  This register performs an interrupt when the 16-bit counter matches the CCR0 buffer register in the free-running mode (interval function).

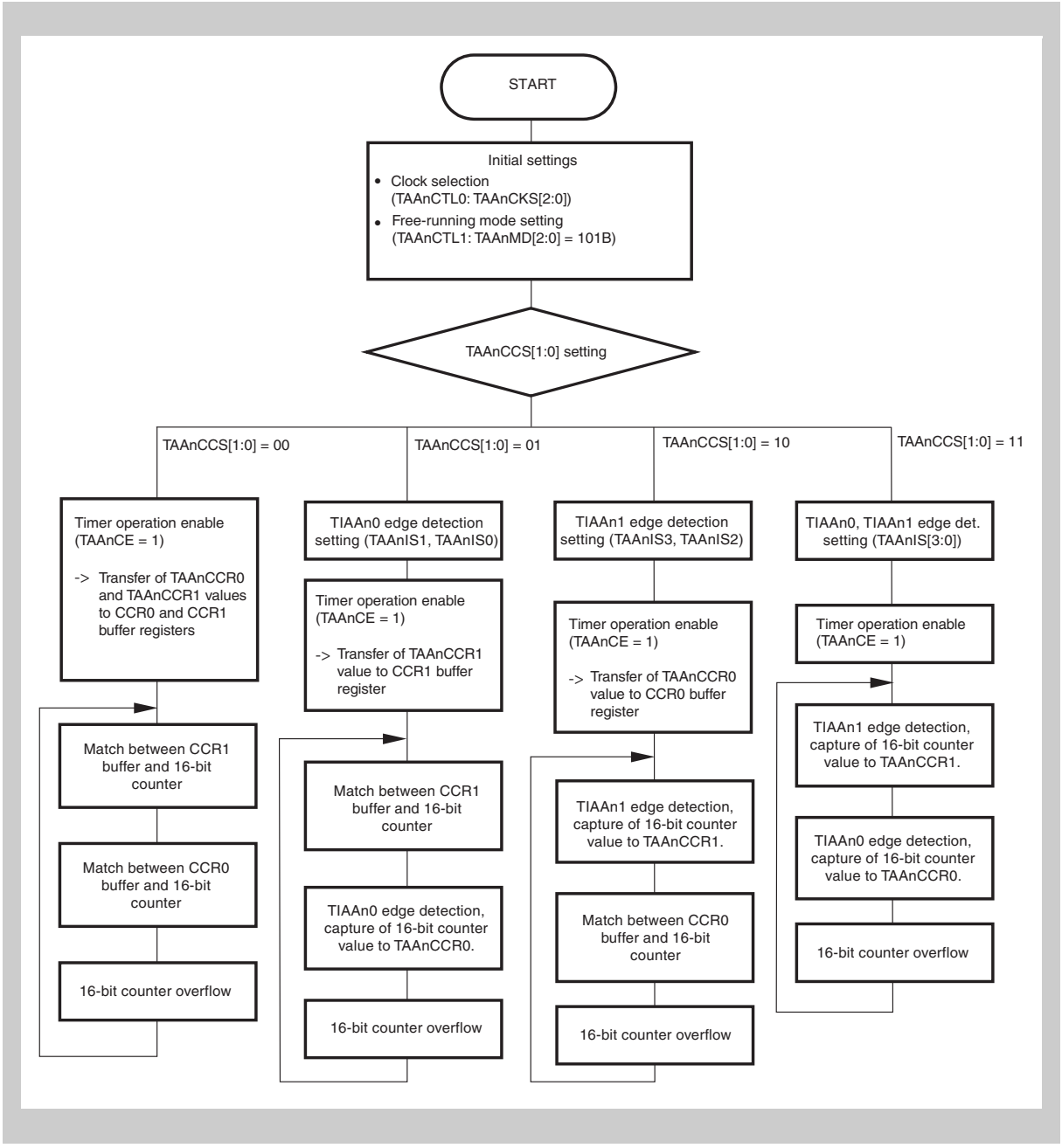
  Rewrite during compare timer operation is enabled and performed with anytime write mode. (Once the compare value has been written, the internal clock synchronizes and this value is used as the 16-bit counter comparison value.)

  When timer output (TOAAn0) has been enabled, TOAAn0 performs toggle output upon a match between the 16-bit counter and the CCR0 buffer register.
- Using TAAnCCR0 register as capture register

  The value of the 16-bit counter is saved to the TAAnCCR0 register upon TIAAn0 pin edge detection.

Caution When the TAAnEEE bit of TAAnCTL1 register is set to 1 and the count clock is set to the external event count input, the TAAnCCR0 register cannot be used as the capture register.

Figure 9-21 Flowchart of basic operation in free-running mode

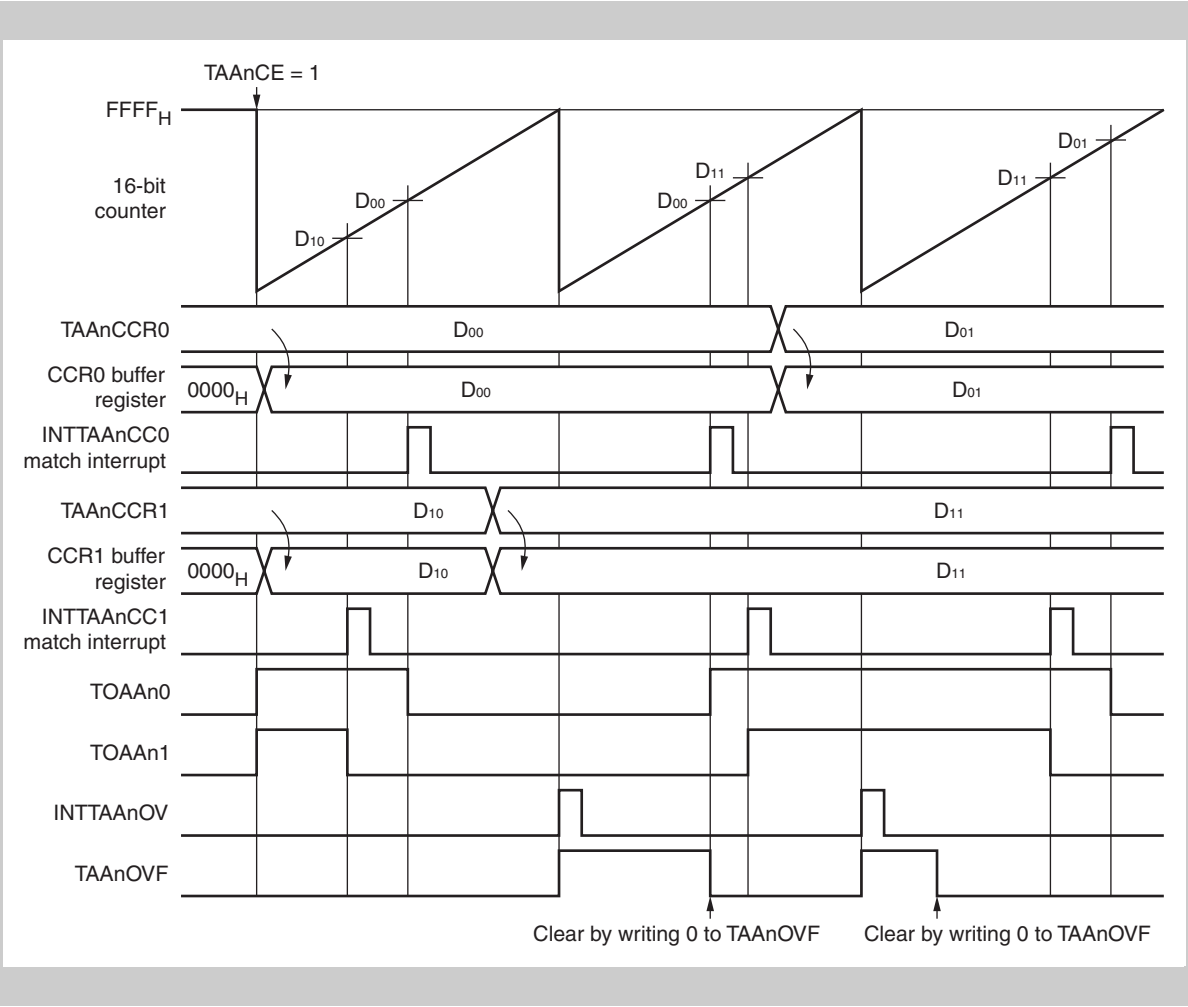


(1) When TAAnCCS1 = 0, and TAAnCCS0 = 0 settings (interval function description, compare function)

When TAAnCE = 1 is set, the 16-bit counter counts from 0000_H to $FFFF_H$ and the free-running count-up operation continues until TAAnCE = 0 is set.

In this mode, when a value is written to the TAAnCCR0 and TAAnCCR1 registers, they are transferred to the CCR0 buffer register and the CCR1 buffer register (any time write mode). In this mode, no one-shot pulse is output even when an one-shot pulse trigger is input. Moreover, when TAAnOEm = 1 is set, TOAAnm performs toggle output upon a match between the 16-bit counter and the CCRm buffer register.

Figure 9-22 Basic operation timing in free-running mode (TAAnCCS1 = 0, TAAnCCS0 = 0) (TAAnOE0 = 1, TAAnOE1 = 1, TAAnOL0 = 0, TAAnOL1 = 0)



D00, D01: Setting values of TAAnCCR0 register (0000_H to $FFFF_H$)
D10, D11: Setting values of TAAnCCR1 register (0000_H to $FFFF_H$)

TOAAn0 output toggle width = (Setting values of TAAnCCR0 register) × (count clock cycle)

TOAAn1 output toggle width = (Setting values of TAAnCCR1 register)

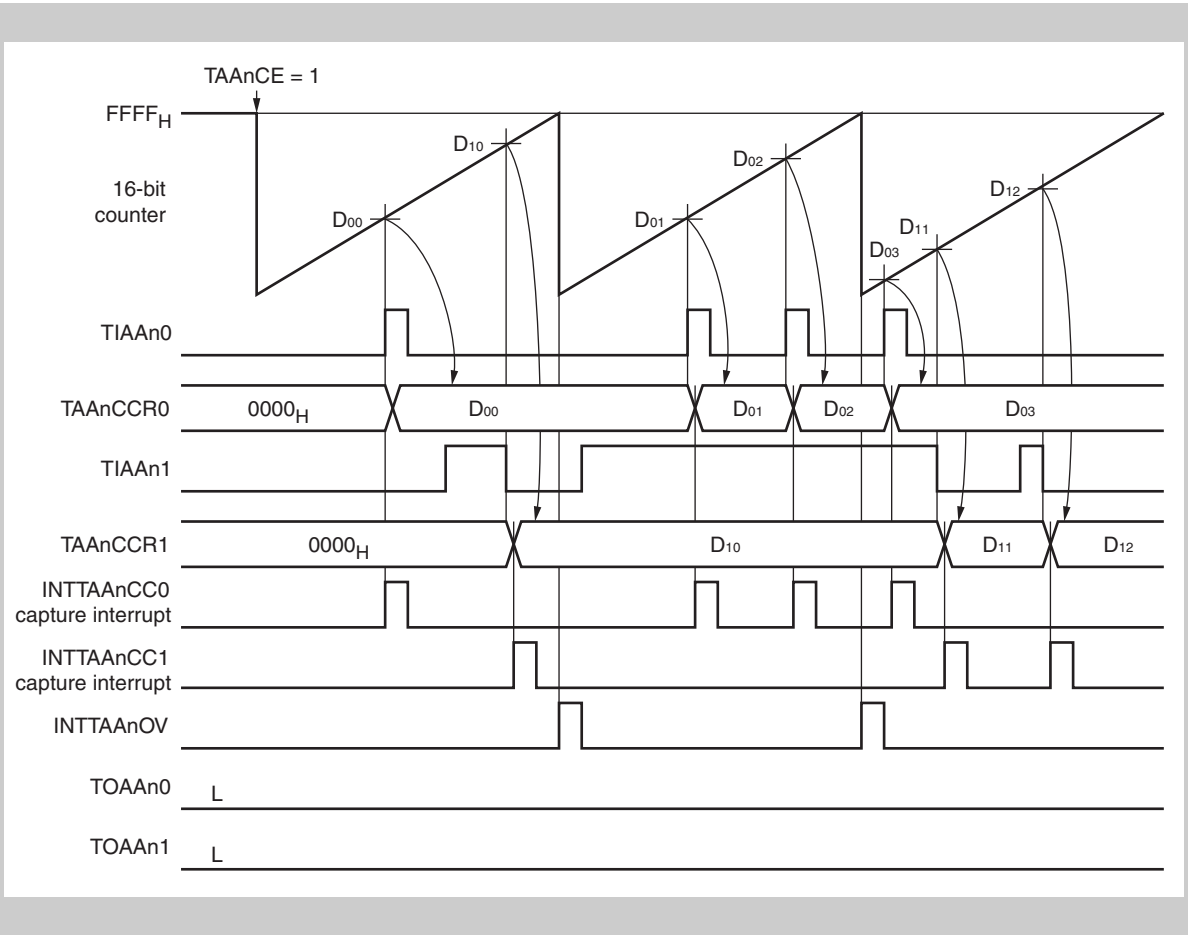
TOAAnm output goes high when counting is started.

(2) When TAAnCCS1 = 1 and TAAnCCS0 = 1 settings (capture function description)

When TAAnCE = 1, the 16-bit counter counts from 0000_H to $FFFF_H$ and free-running count-up operation continues until TAAnCE = 0 is set. During this time, values are captured by capture trigger operation and are written to the TAAnCCR0 and TAAnCCR1 registers.

Regarding capture close to the overflow ($FFFF_H$), judgment is made using the overflow flag (TAAnOVF). However, if overflow occurs twice (two or more free-running cycles), the capture trigger interval cannot be judged with the TAAnOVF flag. In this case, the system should be revised.

Figure 9-23 Basic operation timing in free-running mode (TAAnCCS1 = 1, TAAnCCS0 = 1) (TAAnOE0 = 1, TAAnOE1 = 1, TAAnOL0 = 0, TAAnOL1 = 0)



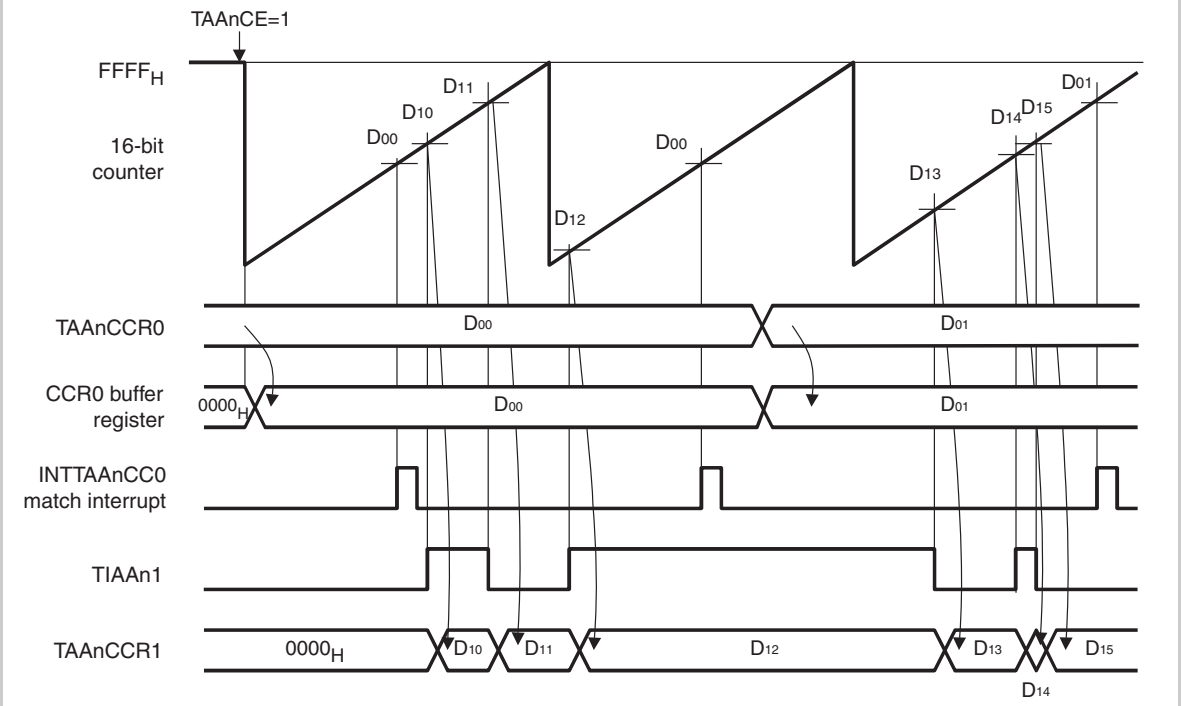
D00, D01, D02, D03:
Values captured to TAAnCCR0 register (0000_H to $FFFF_H$)
D10, D11, D12:
Values captured to TAAnCCR1 register (0000_H to $FFFF_H$)

TIAAn0: Set to rising edge detection (TAAnIS1, TAAnIS0 = 01)
TIAAn1: Set to falling edge detection (TAAnIS3, TAAnIS2 = 10)

(3) When TAAnCCS1 = 1 and TAAnCCS0 = 0

When TAAnCE = 1 is set, the counter counts from 0000_H to $FFFF_H$ and free-running count-up operation continues until TAAnCE = 0 is set. The TAAnCCR0 register is used as a compare register. An interrupt signal is output upon a match between the value of the 16-bit counter and the setting value transferred to the CCR0 buffer register from the TAAnCCR0 register as an interval function. Even if TAAnOE1 = 1 to realize the output function, TAAnCCR1 register cannot control TOAAn1 because it is used as capture register.

Figure 9-24 Basic operation timing in free-running mode (TAAnCCS1 = 1, TAAnCCS0 = 0) (TAAnOE0 = 1, TAAnOE1 = 1, TAAnOL0 = 0, TAAnOL1 = 0)



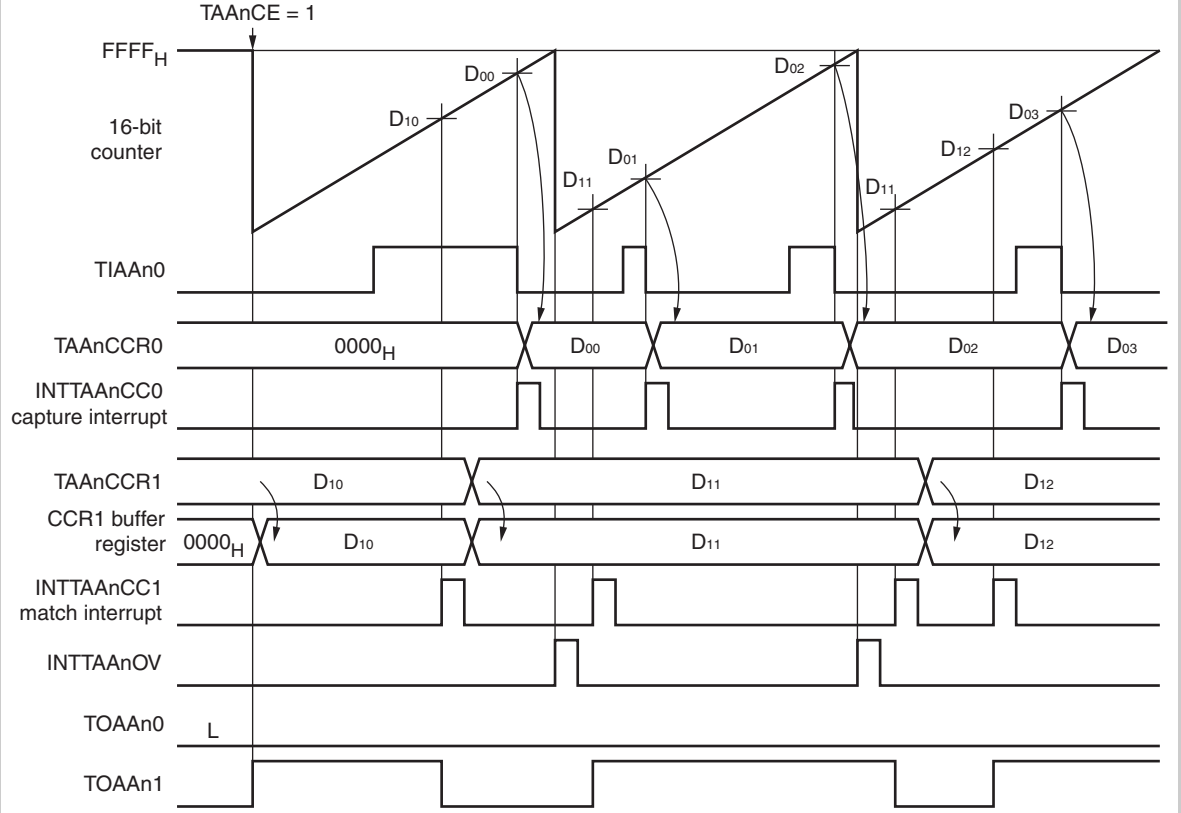
D00, D01:
Setting compare values of TAAnCCR0 register (0000_H to $FFFF_H$)
D10, D11, D12, D13, D14, D15:
Values captured to TAAnCCR1 register (0000_H to $FFFF_H$)
TIAAn1:
Set to detection of both rising and falling edges (TAAnIS3, TAAnIS2 = 11)

(4) When TAAnCCS1 = 0 and TAAnCS0 = 1

When TAAnCE is set to 1, the 16-bit counter counts from 0000_H to $FFFF_H$ and free-running count-up operation continues until TAAnCE = 0 is set. The TAAnCCR1 register is used as a compare register. An interrupt signal is output upon a match between the value of the 16-bit counter and the setting value of the TAAnCCR1 register as an interval function. When TAAnOE1 = 1 is set, TOAAn1 performs toggle output upon mach between the value of the 16-bit counter and the setting value of the TAAnCCR1 register.

Even if TAAnOE0 = 1 to realize the output function, TAAnCCR0 register cannot control TOAAn0 because it is used as capture register.

Figure 9-25 Basic operation timing in free-running mode (TAAnCCS1 = 0, TAAnCCS0 = 1) (TAAnOE0 = 1, TAAnOE1 = 1, TAAnOL0 = 0, TAAnOL1 = 0)



D00, D01, D02, D03:
Values captured to TAAnCCR0 register (0000_H to $FFFF_H$)
D10, D11, D12:
Setting compare value of TAAnCCR1 register (0000_H to $FFFF_H$)

TIAAn0: Set to falling edge detection (TAAnIS1, TAAnIS0 = 10)

(5) Overflow flag

When the counter overflows from $FFFF_H$ to 0000_H in the free-running mode, the overflow flag (TAAnOVF) is set to 1 and an overflow interrupt (INTTAAnOV) is output.

The overflow flag is cleared by the CPU when writing 0 to it.

9.7.8 Pulse width measurement mode (TAAnMD2 to TAAnMD0 = 11_{0B})

In the pulse width measurement mode, a free-running count is performed. The value of the 16-bit counter is saved to capture register 0 (TAAnCCR0) or capture register 1 (TAAnCCR1) respectively. Also, the 16-bit counter is cleared upon edge detection of the TIAAn0 pin, or TIAAn1 respectively. The external input pulse width can be measured as a result.

However, when measuring a large pulse width that exceeds 16-bit counter overflow, perform judgment with the overflow flag. Since measurement of pulses for which overflow occurs twice or more is not possible, adjust the operating frequency of the 16-bit counter.

Depending on the selected capture input sources and specified edge detection three different measurement methods can be applied.

1. Pulse period measurement
1. Alternating pulse width and pulse space measurement.
2. Simultaneous pulse width and pulse space measurement:
   Both capture inputs are required to measure pulse width and pulse space simultaneously.

The measurements methods are explained in the following sub-chapters.

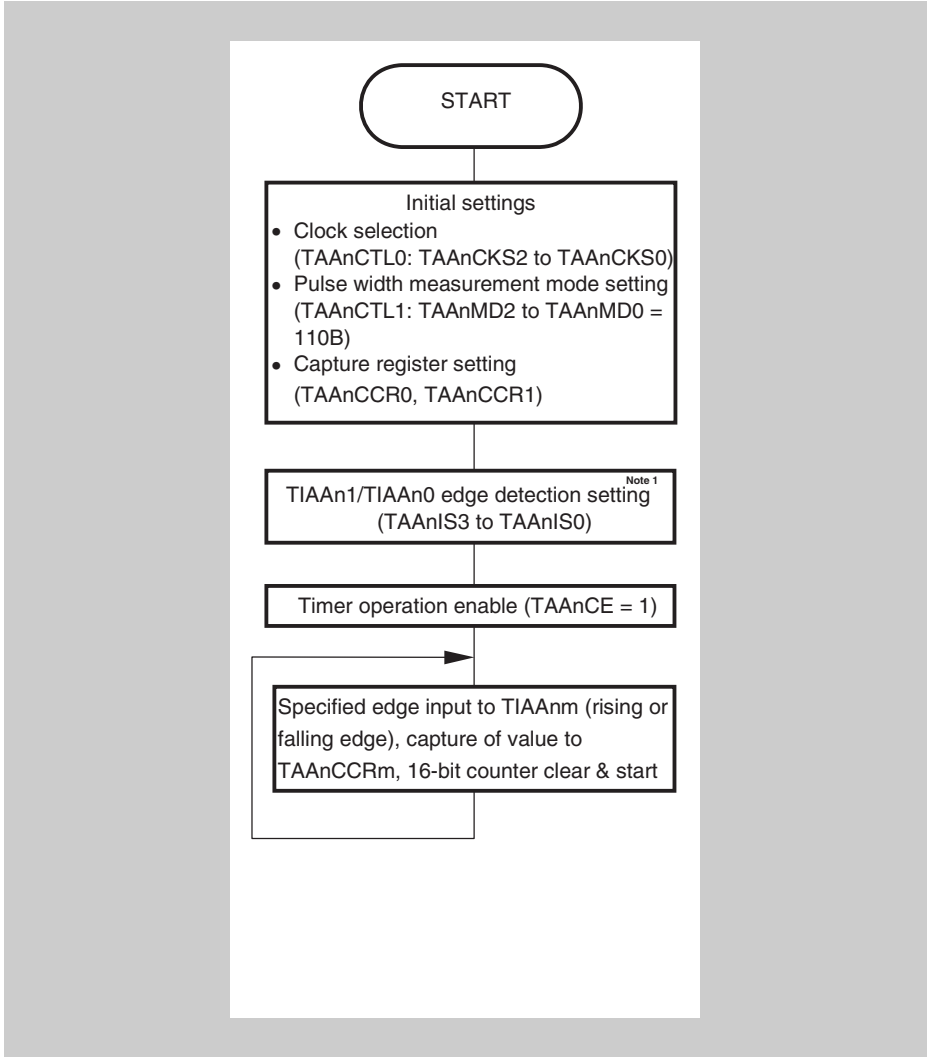
Caution In the pulse width measurement mode, select the internal clock (TAAnEEE of TAAnCTL1 register = 0).

(1) Pulse period measurement

The pulse period of a signal can be measured in the pulse width measurement mode, when the edge detection of one of the inputs TIAAn0 and TIAAn1 is set either to “rising edge” or “falling edge”. The detection of the other input should be set to “no edge detection”.

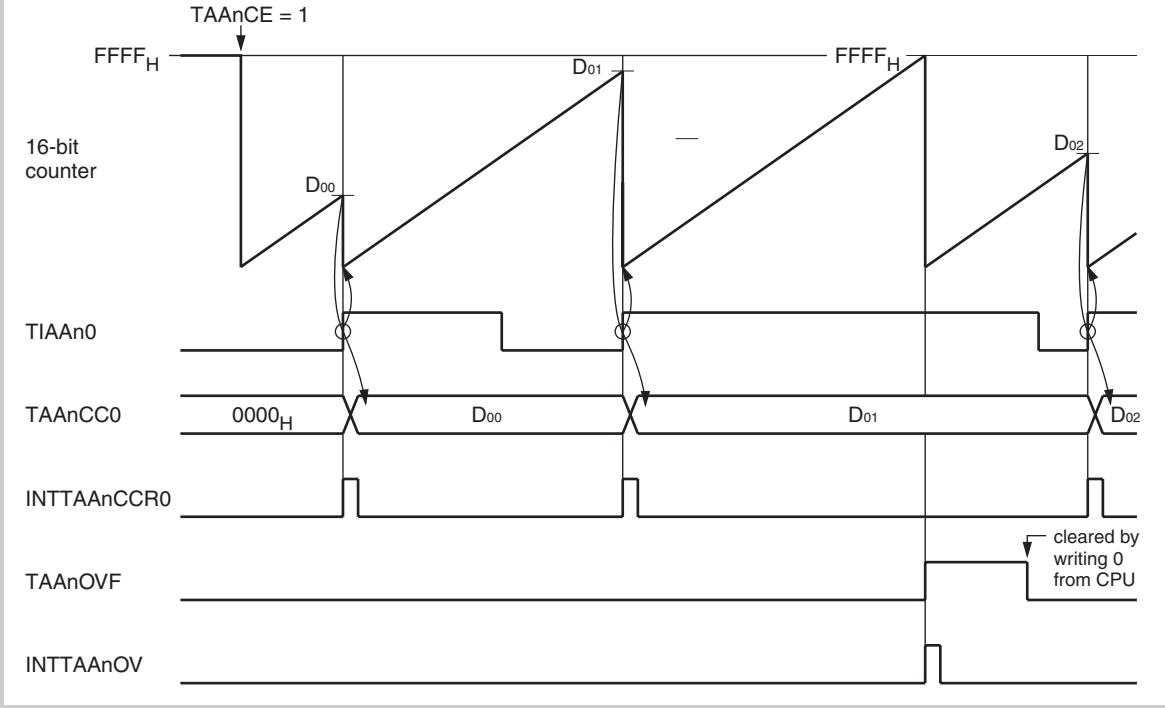
By detection of the specified edge the resulting value is captured in the corresponding capture register (TAAnCCR0 or TAAnCCR1), and the timer is cleared and restarts counting.

Figure 9-26 Flowchart of pulse period measurement



Note 1. External pulse input is possible for both TIAAn0 and TIAAn1, but only one should be selected for the pulse period measurement.
Specify either “rising edge” or “falling edge” for edge detection. Specify the edge of the external input pulse that is not used as “no edge detection”.

Figure 9-27 Basic operation timing of pulse period measurement (TAAnOE0 = 0, TAAnOE1 = 0, TAAnOL0 = 0, TAAnOL1 = 0)



D_{00}, D_{01}, D_{02}: Values captured to TAAnCCR0 register (0000_H to $FFFF_H$)

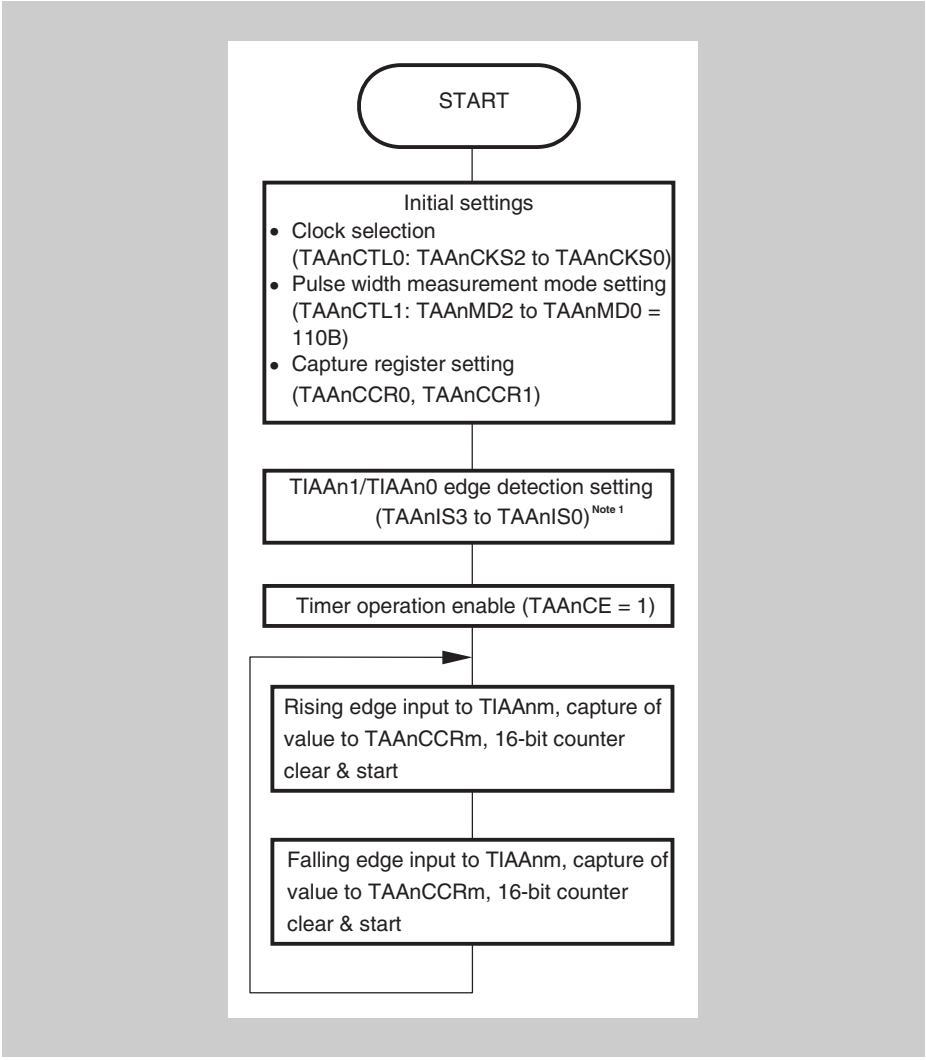
TIAAn0: Set to detection of rising edge (TAAnIS1, TAAnIS0 = 01B)
TIAAn1: Set to no edge detection (TAAnIS3, TAAnIS2 = 00_B)

(2) Alternating pulse width and pulse space measurement

The pulse period of a signal can be measured in the pulse width measurement mode alternating in one capture register, when the edge detection of one of the inputs TIAAn0 and TIAAn1 is set to “both rising and falling edges”. The detection of the other input should be set to “no edge detection”.

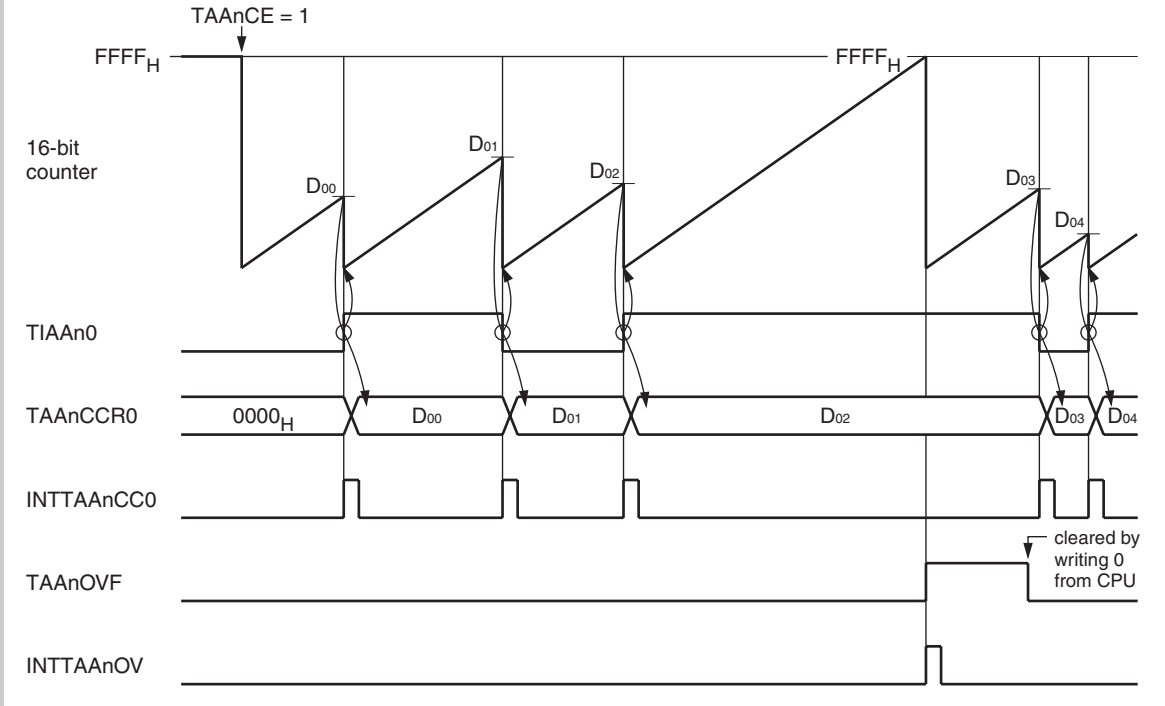
By detection of a falling or rising edge the resulting value is captured in the corresponding capture register (TAAnCCR0 or TAAnCCR1), and the timer is cleared and restarts counting.

Figure 9-28 Flowchart of alternating pulse width and pulse space measurement



Note 1. External pulse input is possible for both TIAAn0 and TIAAn1, but only one should be selected for the alternating pulse width and pulse space measurement.
Specify “both rising and the falling edges” for edge detection. Specify the edge of the external input pulse that is not used as “no edge detection”.

Figure 9-29 Basic operation timing of alternating pulse width and pulse space measurement (TAAnOE0 = 0, TAAnOE1 = 0, TAAnOL0 = 0, TAAnOL1 = 0)



D_{00}, D_{01}, D_{02}, D_{03}, D_{04}:
Values captured to TAAnCCR0 register (0000_H to $FFFF_H$)

TIAAn0:
Set to detection of both rising and falling edges (TAAnIS1, TAAnIS0 = 11_B)
TIAAn1:
Set to no edge detection (TAAnIS3, TAAnIS2 = 00_B)

Pulse width = Captured value × Count clock cycle

If the valid edge is not input even when the 16-bit counter counted up to $FFFF_H$, an overflow interrupt request signal (INTTAAnOV) is generated at the next count clock, and the counter is cleared to 0000_H and continues counting. At this time, the overflow flag (TAAnOPT0.TAAnOVF 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.

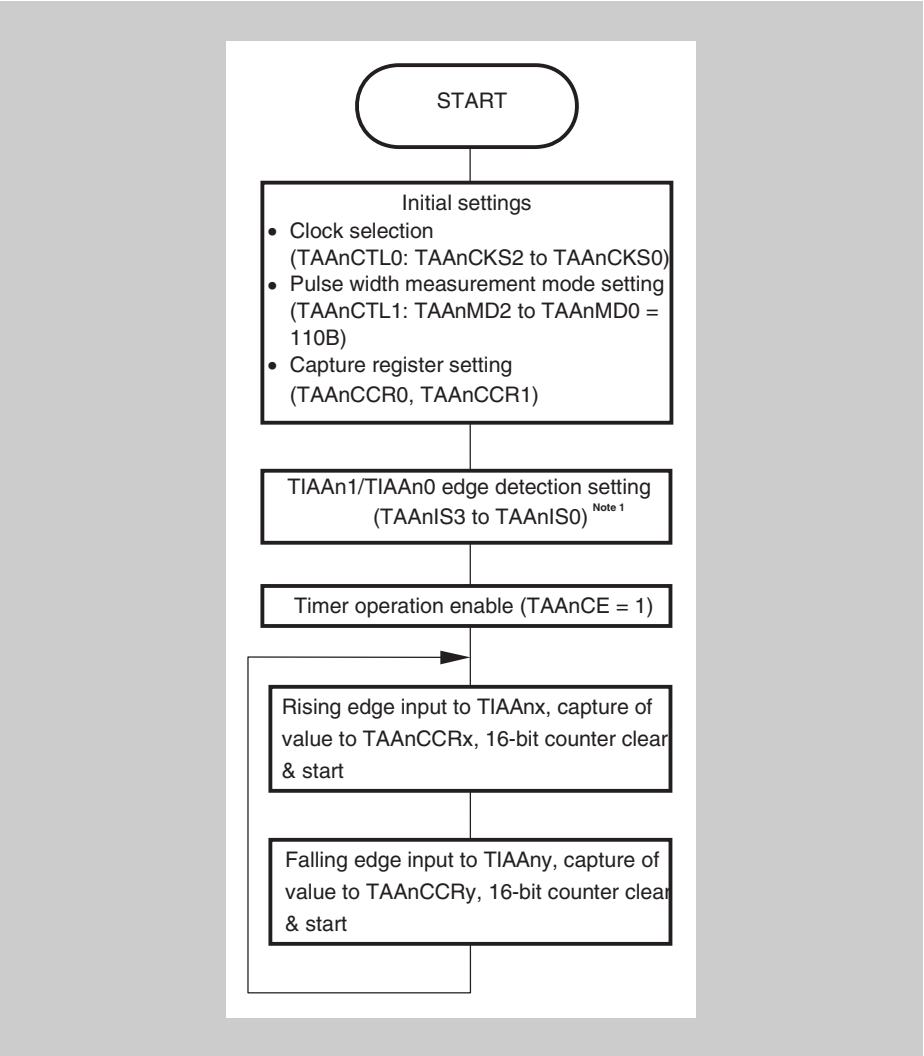
Pulse width = (10000_H × TAAnOVF bit set (1) count + Captured value) × Count clock cycle

(3) Simultaneous pulse width and pulse space measurement

Pulse width and pulse space can be measure simultaneously in the pulse width measurement mode, when the signal is input to both inputs TIAAn0 and TIAAn1, where both inputs detect opposite edges. Alternatively the signal can be input to TIAAn0 only, when the capture source input selection for capture register 1 is used. (refer to *Section 9.4.2, SEL0CNT - Selector Control Register 0* on page 163.

By detection of the specified edge the resulting values of pulse width or pulse space are captured in the corresponding capture registers (TAAnCCR0, TAAnCCR1), and the timer is cleared and restarts counting.

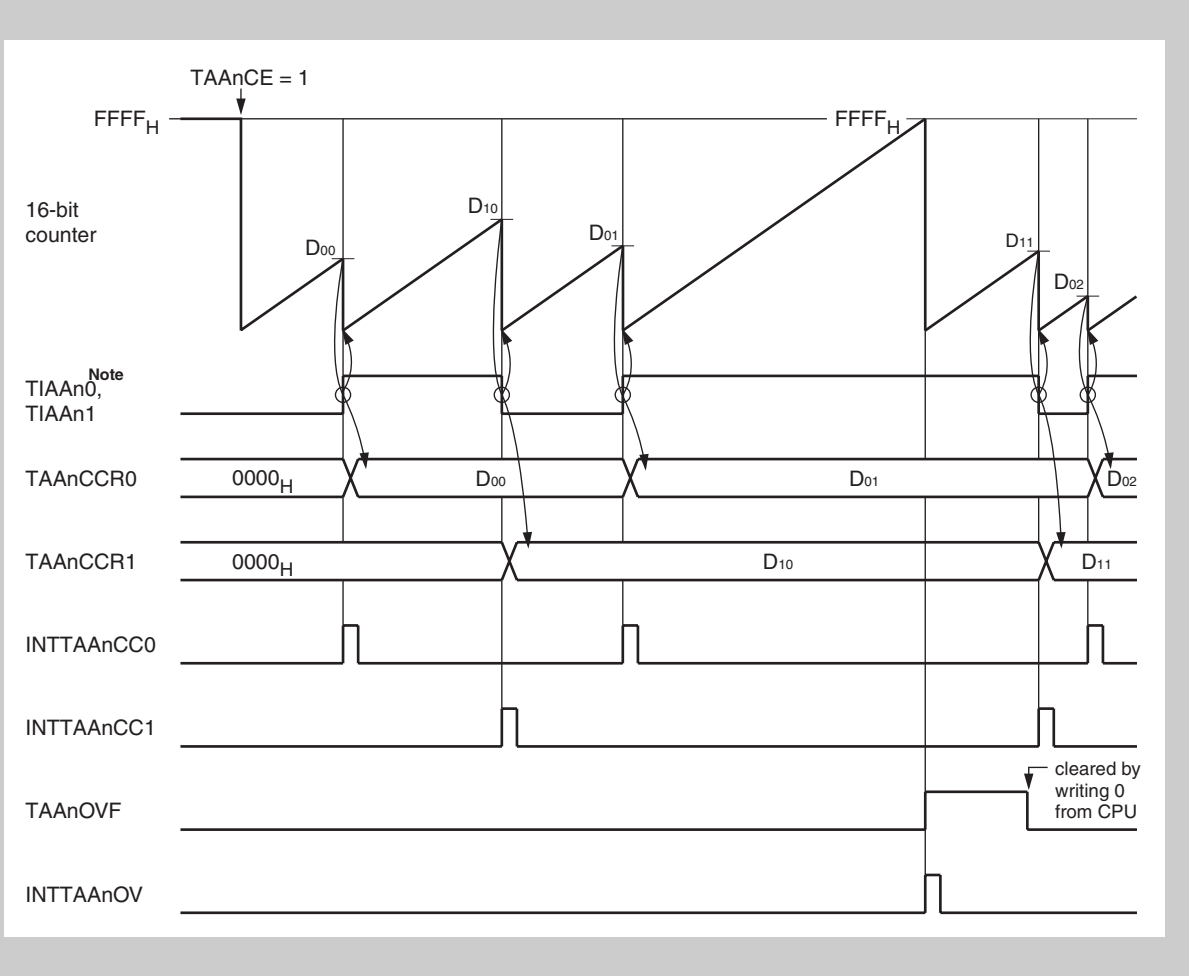
Figure 9-30 Flowchart of simultaneous pulse width and pulse space measurement



Note 1. External pulse input must be input to both TIAAn0 and TIAAn1.
Specify “rising edge” for edge detection of first input, and “falling edge” for the second input, or vice versa.

2. x = 0,1
y = 0 when x = 1; y = 1 when x = 0

Figure 9-31 Basic operation timing of simultaneous pulse width and pulse space measurement (TAAnOE0 = 0, TAAnOE1 = 0, TAAnOL0 = 0, TAAnOL1 = 0)



Note The signal to measure has to be assigned to both inputs, TIAAn0 and TIAAn1.

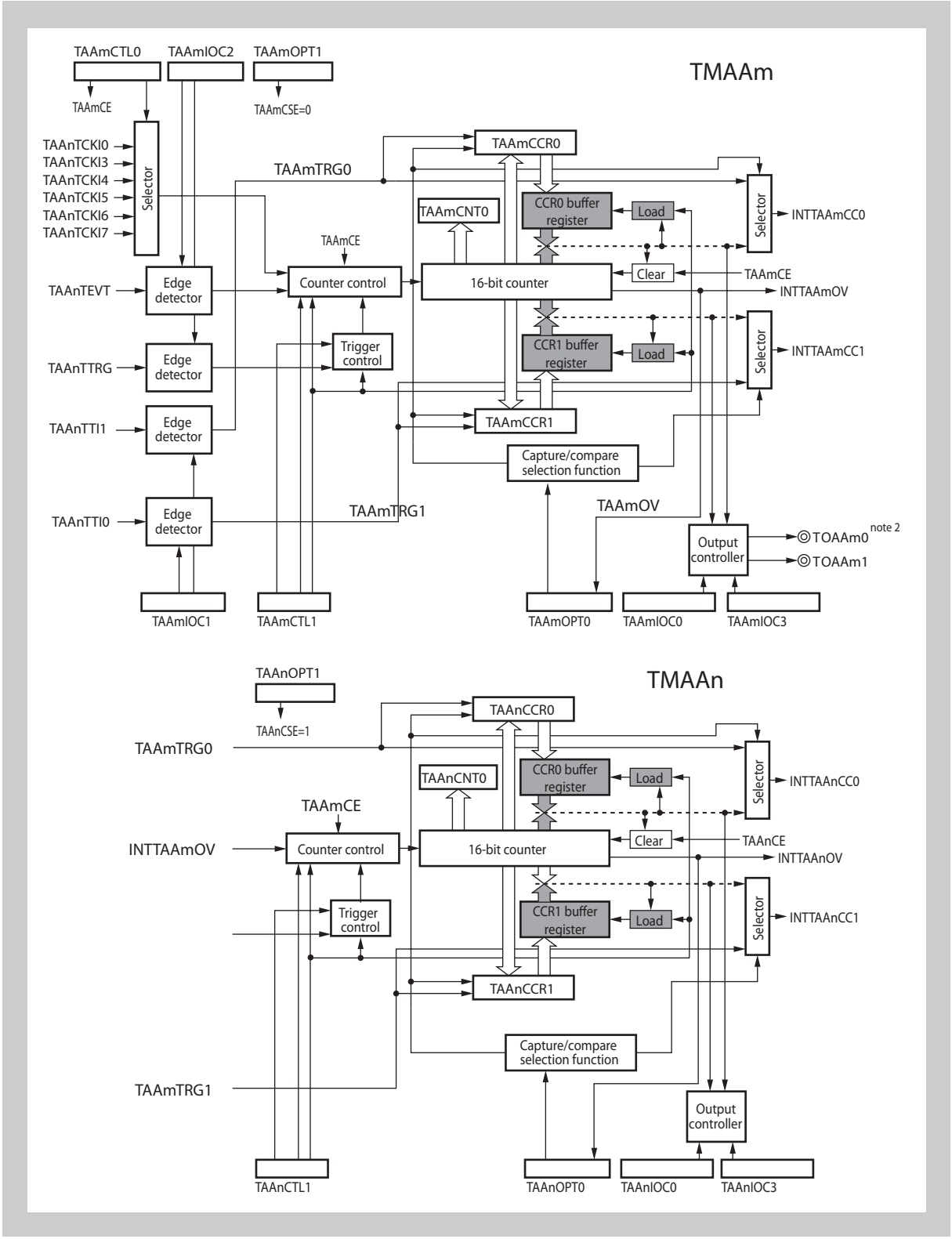
D_{00}, D_{01}, D_{02}: Values captured to TAAnCCR0 register (0000_H to $FFFF_H$)
D_{10}, D_{11}: Values captured to TAAnCCR1 register (0000_H to $FFFF_H$)

TIAAn0: Set detection to rising edge (TAAnIS1, TAAnIS0 = 01_B)
TIAAn1: Set detection to falling edge (TAAnIS3, TAAnIS2 = 10_B)

9.7.9 32-bit Capture in free-running cascade mode

Two Timer AA (TAAn in combination with TAAm, for details please check device details) can be cascaded to operate as a 32-bit capture timer. In cascade mode, the timer with the lower number (TAAm) is used to control the operation (master timer). Both cascaded timers have to be initialized as free-running timers.

Figure 9-32 Block diagram of TAAm and TAAn in 32-bit capture mode



Note Explanation of signals can be found in *Figure 9-2 on page 164*.

Figure 9-32 shows the block diagram of TAAm and TAAn in cascade mode. Signals that are irrelevant in cascade mode are not shown, the connections to the internal bus are also hidden for better readability of the image, as *Figure 9-2 on page 164* can be used for a in-depth look of each timer.

Note Cascading two TAA timers is only allowed for free-running mode with both capture/compare registers set to capture mode. Proper operation of TAAm and TAAn is not guaranteed for any other setting.

Figure 9-34 shows the recommended flow for setting up TAAm and TAAn in cascade mode. As TAAm is used for general control, TAAn is set up first and set in cascaded operation by setting the TAAnCSE bit to 1. Then TAAm is initialized by selecting the proper clock setting and capture trigger input. Only TIAAm0 and TIAAm1 can be used as external capture trigger.

Note When cascading TAAm and TAAn, set TAAnCSE=1 and TAAmCSE=0.

Operation starts when the count enable flag of TAAm (TAAmCE) is set to 1. The counter of TAAm is used for the lower 16-bit of the 32-bit count value, while the upper 16-bit are handled by TAAn.

Whenever the counter of TAAm overflows, the counter is cleared to 0, interrupt INTTAAmOV is generated and the counter of TAAn is incremented by 1. When the counter of TAAn overflows, the counter is also cleared to 0 and interrupt INTTAAnOV is generated.

When a capture trigger 0/1 is detected by TAAm, a capture of the lower 16-bit counter value to TAAmCCR0/1 and of the upper 16-bit counter value to TAAnCCR0/1 at the same time. The interrupts of the TAAm will indicate the capture (INTTAAmCC0/1).

Figure on page 215 shows an example of a 32-bit capture timing.

Figure 9-33 Basic flow of 32-bit capture mode

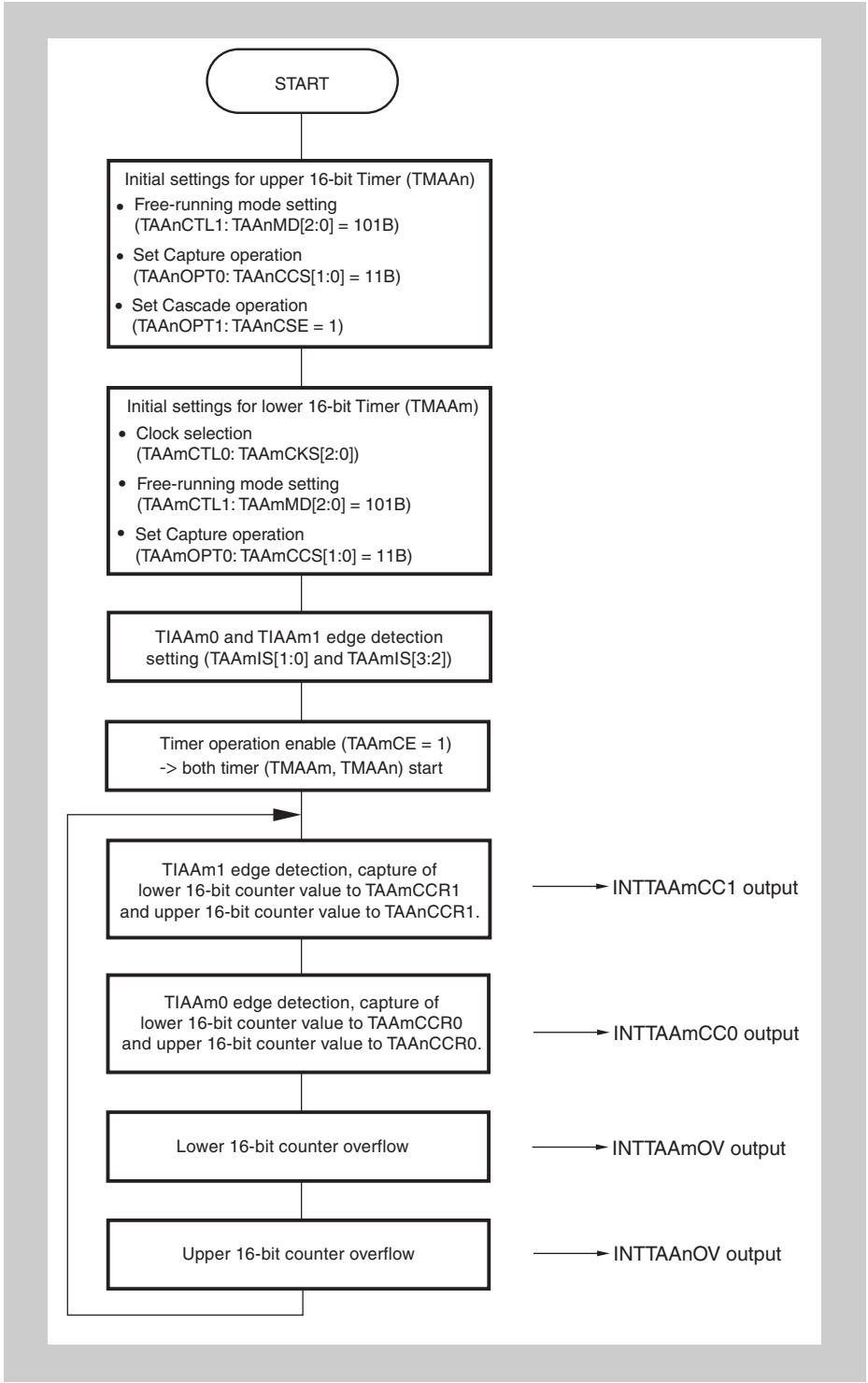
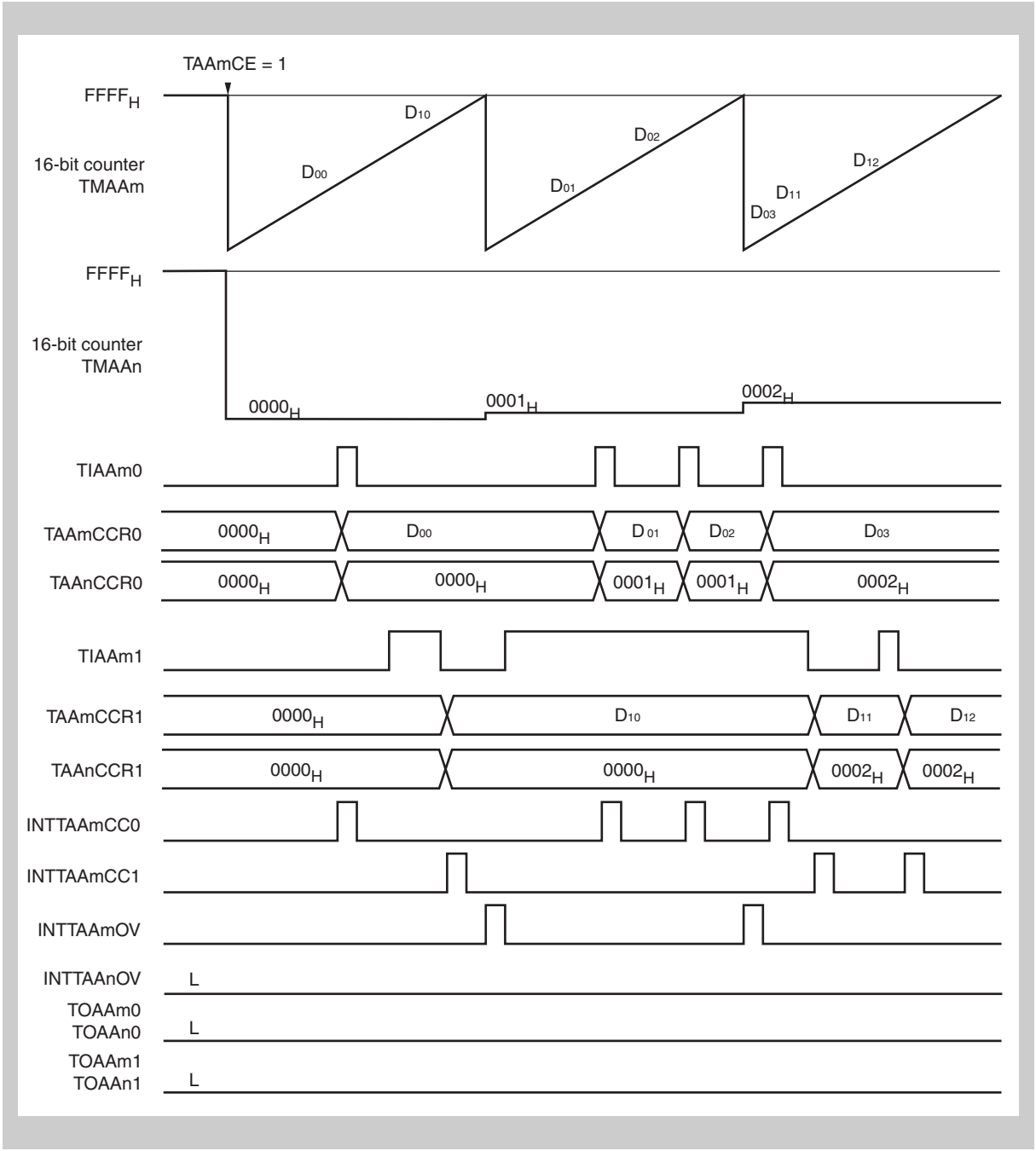
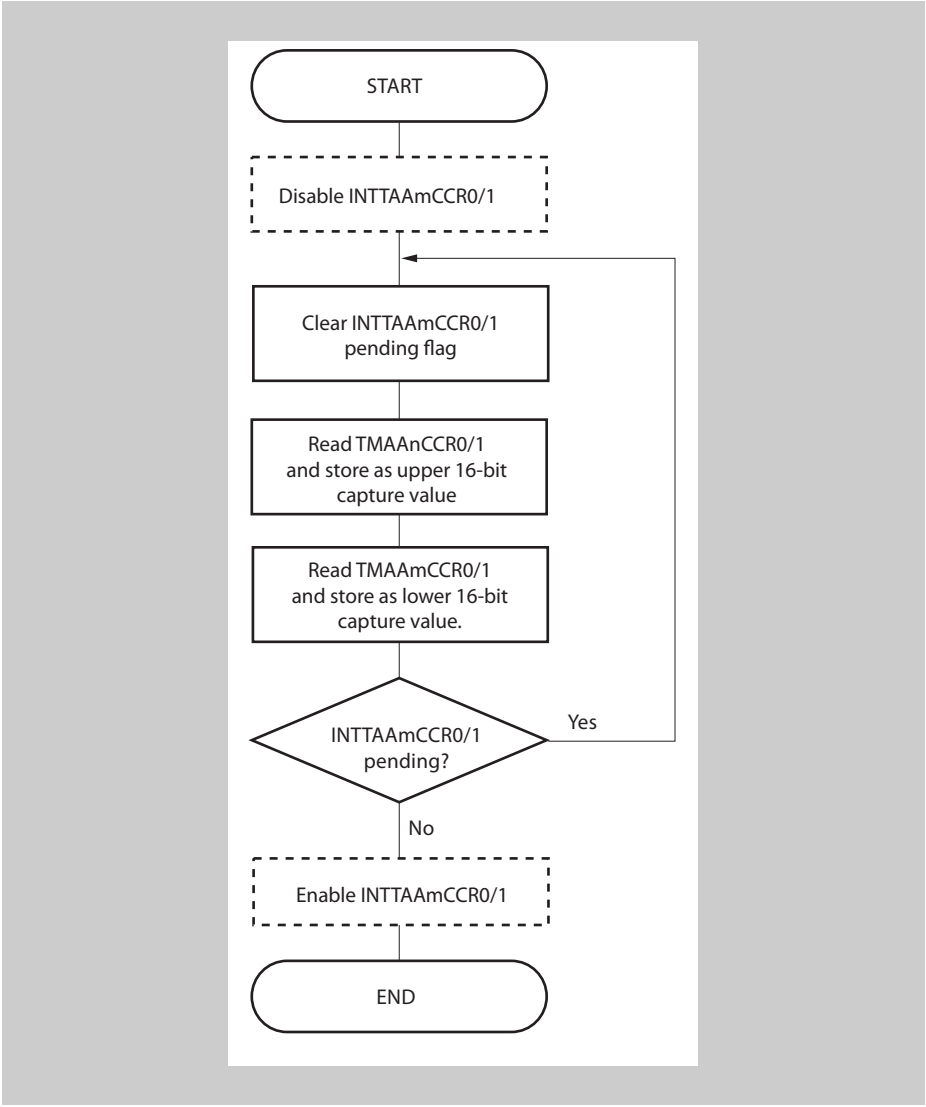


Figure 9-34 Basic timing of 32-bit capture mode



Because the 32-bit resolution is achieved by cascading two individual TAA timers, a direct read of the 32-bit capture value is not possible. To ensure that the data is not corrupted during read operation, the following procedure for reading needs to be followed:

Figure 9-35 Flow of 32-bit read (capture or counter value)



Disabling the capture interrupt (INTTAAmCCR0/1) is not required if the read sequence is done in the interrupt service routine, as nesting of the same interrupt is not possible. However, if the read operation is done in a normal routine while the interrupt signal is also assigned to a interrupt service routine, disabling the interrupt is mandatory; otherwise corrupted data might be read.

The same flow can be used for reading the timer counter value. In this case the relevant interrupt that needs to be cleared and checked is INTTAAmOV. Please note that you can either read the upper 16-bit counter (TAAnCNT) and then the lower 16-bit counter (TAAmCNT) or vice versa. While both methods work, the read values can be slightly different, as the count operation of the lower 16-bit counter continues while the upper 16-bit timer is read:

- When reading the upper 16-bit first, the lower 16-bit might be incremented during that read.
- When reading the lower 16-bit first, the value might be already “old” after reading the upper 16-bit.

The software programmer needs to decide which method is considered better for the application.

9.8 Timer Synchronization Operation Function

Timers AA have a timer synchronized operation function (tuned operation mode). Master timer and incorporated slave timers of the corresponding timer group (listed in *Table 9-6*) start and clock synchronously. When the master timer is cleared, the slave timers are cleared synchronously, too.

Table 9-6 Tuned Operation Mode of Timer

Master Timer	Slave Timer
TAA2	TAA3

Cautions

1. The tuned operation mode is enabled or disabled by the TAAnSYE bit of the TAAnCTL1 register and TABnSYE bit of the TABnCTL1 register.
2. Set the tuned operation mode using the following procedure.
   - Set the TAAnSYE bit of the TAAnCTL1 register of the slave timer to enable the tuned operation. Set the TAAnMD2 to TAAnMD0 bits of the TAAnCTL1 register of the slave timer to the free-running mode (101_B).
   - Set the timer mode of the master timer by using the TAAnMD2 to TAAnMD0 bits of the TAAnCTL1 register to PWM mode (110_B), external trigger pulse output mode (010_B), one-shot pulse output mode (011_B), or free-running mode (101_B). At this time, do not set the TAAnSYE bit of the TAAnCTL1 register of the master timer.
   - Set the compare register value of the master and slave timers.
   - Set the TAAnCE bit of the TAAnCTL0 register of the slave timer to enable operation on the internal operating clock.
   - Set the TAAnCE bit of the TAAnCTL0 register of the master timer to enable operation on the internal operating clock.

Table 9-7 and *Table 9-8* show the timer modes that can be used in the tuned operation mode

(√: settable, ×: not settable).

Table 9-7 Timer modes usable in tuned operation mode

Master Timer	Free-Running Mode	PWM Mode	External Trigger Pulse Output Mode	One-shot Pulse Output Mode
TAA2	√	√	√	√

Table 9-8 Timer output functions

Tuned Timer Group	Timer	Pin	Free-Running Mode		PWM Mode	
			Tuning OFF	Tuning ON	Tuning OFF	Tuning ON
0	TAA2 (master)	TOAA20	PPG	←	Toggle	←
		TOAA21	PPG	←	PWM	←
	TAA3 (slave)	TOAA30	PPG	←	Toggle	PWM
		TOAA31	PPG	←	PWM	←

Note The timing of transmitting data from the compare register of the master timer to the compare register of the slave timer is as follows.
PPG: CPU write timing (anytime write mode)
Toggle, PWM: Timing at which timer counter and compare register match TOAAn0 (n = 0 to 9) (reload mode)

Chapter 10 16-Bit Timer/Event Counter AB (TAB)

The V850E/Rx3 includes two channels of the 16-bit Timer/Event Counter AB (TAB0, TAB1). The timer is upward compatible to Timer Q used in other V850E and V850ES devices, including V850E/RS1. Timer AB offers new additional features for enhanced output control and for 32-bit capture.

Instances The V850E/Rx3 has two instances of this 16-bit timer/event counter AB.

Table 10-1 Instances of Timer AB

Timer AB	
Instances	2
Names	TAB0, TAB1

Throughout this chapter, the individual instances of Timer AB are identified by “n” (n = 0 to 1), for example, TABnCTL0 for the TABn control register 0.

10.1 Features

Timer AB (TAB) is a 16-bit timer/event counter provided with general-purpose functions.

TAB can perform the following operations.

- 16-bit-accuracy PWM output
- Interval timer
- External event counter (operation not possible when clock is stopped)
- One-shot pulse output
- Pulse width measurement function
- Triangular wave PWM output
- External trigger pulse output function
- Free-running function
- Timer synchronized operation function with Timers AA and Timers AB channels (refer to *Section Chapter 11, Timer AA/AB Synchronous Operation* on page 267)

10.2 Function Outline

- Capture trigger input signal × 4
- External trigger input signal × 1
- Clock select × 8
- External event input × 1
- Readable counter × 1
- Capture/compare reload register × 4

- Capture/compare match interrupt × 4
- Timer output (TOABn0 to TOABn3) × 4

10.3 Configuration

TAB includes the following hardware.

Table 10-2 Timer AB Configuration

Item	Configuration
Timer register	16-bit counter × 1
Registers	• TABn capture/compare registers 0 to 3 (TABnCCR0 to TABnCCR3) • TABn counter read buffer register (TABnCNT) • CCR0 to CCR3 buffer registers
Timer input	4 (TIABn0[Note] to TIABn3), TABnTTRG, TABnTEVT
Timer output	4 (TOABn0 toTOABn3)
Control registers	• TABn control registers 0, 1 (TABnCTL0, TABnCTL1) • TABn dedicated I/O control registers 0 to 2 and 4 (TABnIOC0 to TABnIOC2 and TABnIOC4) • TABn option registers 0 (TABnOPT0)

Note TIABn0 functions alternately as a capture trigger input signal, external trigger input signal, and external event input signal.

Clock supply Each Timer AB instance has 6 clock inputs. All are connected to the clock generator.

The table below lists the cross reference between TABn clock input and the connected clocks:

Table 10-3 Clock inputs of TABn

TABn clock input	Connected clock	
	TAB0	TAB1
TABnTCKI0	PCLK0 (32 MHz)	PCLK1 (16 MHz)
TABnTCKI3	PCLK3 (4 MHz)	PCLK4 (2 MHz)
TABnTCKI4	PCLK4 (2 MHz)	PCLK5 (1 MHz)
TABnTCKI5	PCLK5 (1 MHz)	PCLK6 (500 KHz)
TABnTCKI6	PCLK6 (500 KHz)	PCLK7 (250 KHz)
TABnTCKI7	PCLK7 (250 KHz)	PCLK8 (125 KHz)

Timer AB (TAB) pins are alternate functions of port pins. For guidance on setting the alternate function, refer to the description of the registers in *Section Chapter 2, Pin Functions* on page 7.

Input connection for 3V The following block diagrams show the connection of the TABn macro inputs.

Figure 10-1 Input circuit of TAB0

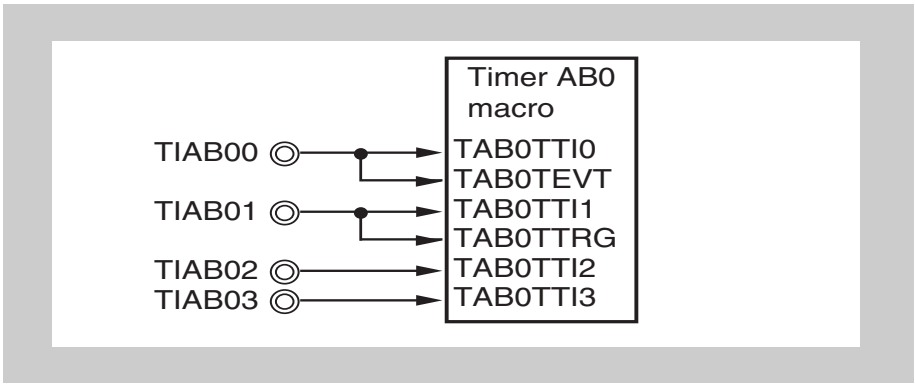
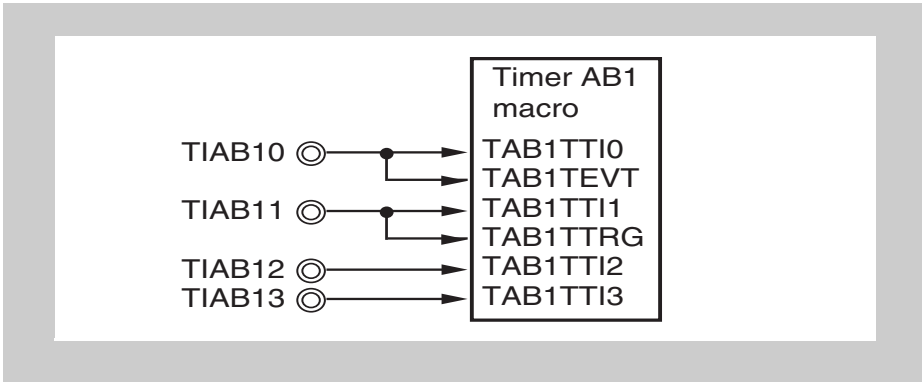


Figure 10-2 Input circuit of TAB1



Input connection for 5V

The following block diagrams show the connection of the TABn macro inputs.

Figure 10-3 Input circuit of TAB0

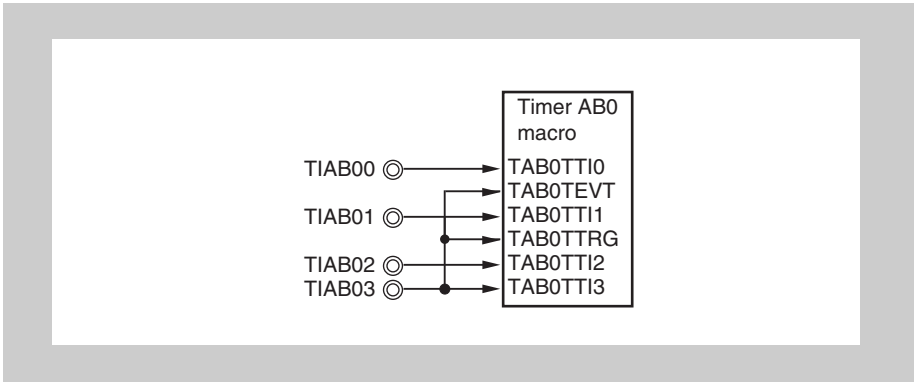
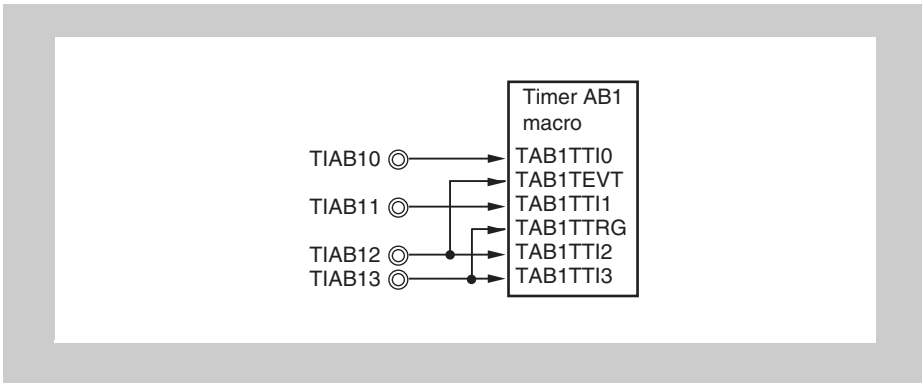
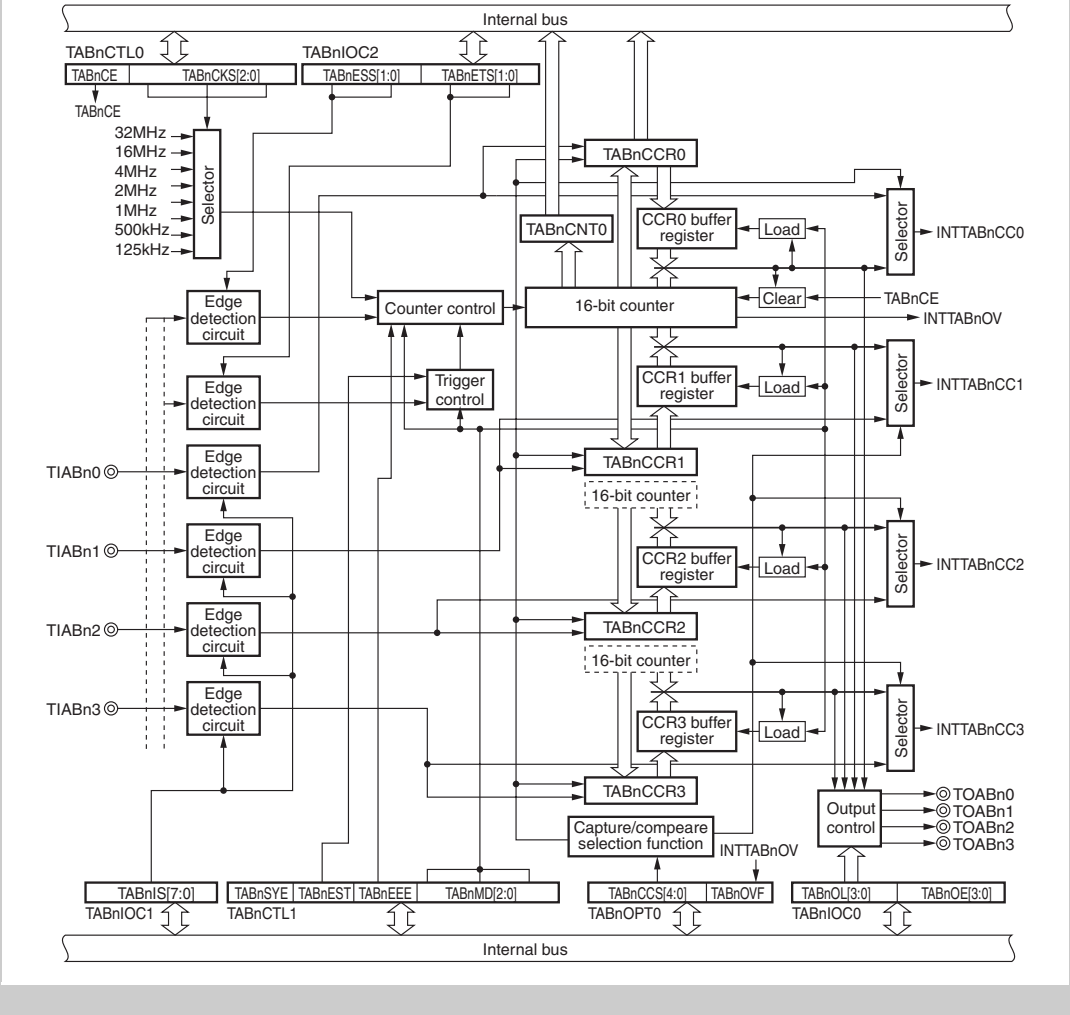


Figure 10-4 Input circuit of TAB1



10.4 Timer AB block diagram

Figure 10-5 Block diagram of Timer AB



10.5 Control Registers

Register addresses All TABn register addresses are given as address offsets to the individual base addresses <base> of each TABn.

The <base> addresses of each TABn are listed in the following table.

Table 10-4 Register <base> addresses of TABn

TABn	<base> address
TAB0	FFFF $F680_H$
TAB1	FFFF $F6A0_H$

The TABn are controlled and operated by means of the following registers.

Table 10-5 TABn registers overview

Register name	Shortcut	Address
TABn control register 0	TABnCTL0	<base>
TABn control register 1	TABnCTL1	<base> + 1_H
TABn I/O control register 0	TABnIOC0	<base> + 2_H
TABn I/O control register 1	TABnIOC1	<base> + 3_H
TABn I/O control register 2	TABnIOC2	<base> + 4_H
TABn option register 0	TABnOPT0	<base> + 5_H
TABn capture/compare register 0	TABnCCR0	<base> + 6_H
TABn capture/compare register 1	TABnCCR1	<base> + 8_H
TABn capture/compare register 2	TABnCCR2	<base> + A_H
TABn capture/compare register 3	TABnCCR3	<base> + C_H
TABn counter read buffer register	TABnCNT	<base> + E_H
TABn I/O control register 4	TABnIOC4	<base> + 10_H

10.5.1 TABnCTL0 - TAB control register 0

Timer AB control register 0 is an 8-bit register that controls the operation of timer AB.

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

Address <base>

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

7	6	5	4	3	2	1	0
TABnCE	0	0	0	0	TABnCKS2	TABnCKS1	TABnCKS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

TABnCE	Timer ABn operation control
0	Disable internal operating clock operation (asynchronously reset TABn).
1	Enable internal operating clock operation.
The TABnCE bit controls the internal operating clock and asynchronously resets TABn. When this bit is cleared to 0, the internal operating clock of TABn is stopped (fixed to the low level), and TABn is asynchronously reset. When the TABnCE bit is set to 1, the internal operating clock is enabled within 2 input clocks, and TABn counts up.	

Note If the timer is operated in synchronous operation mode, i.e., TABnCTL1.TABnSYE = 1, then TABnCTL0.TABnCE cannot be set to “1”.

TABnCKS2	TABnCKS1	TABnCKS0	Internal count clock selection
0	0	0	TABnTCKI0
0	0	1	TABnTCKI0/2
0	1	0	TABnTCKI0/4
0	1	1	TABnTCKI3
1	0	0	TABnTCKI4
1	0	1	TABnTCKI5
1	1	0	TABnTCKI6
1	1	1	TABnTCKI7

Caution Set bits TABnCKS[2:0} when TABnCE = 0. When the value of the TABnCE bit is changed from 0 to 1, bits TABnCKS[2:0] can be set simultaneously.

10.5.2 TABnCTL1 - Timer AB control register 1

The TABnCTL1 register is an 8-bit register that controls the operation of timer AB.

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
TABnSYE	TABnEST	TABnEEE	0	0	TABnMD2	TABnMD1	TABnMD0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

TABnSYE	Synchronous operation mode enable
0	Independent operation mode (asynchronous operation mode)
1	Synchronous operation mode (specification of slave operation) • In this mode, timer AB can operate in synchronization with a master timer. • If TABnSYE = 1, TABnCTL0.TABnCE cannot be set to “1”. • For the synchronous operation mode, refer to *Section Chapter 11, Timer AA/AB Synchronous Operation* on page 267.

Caution In the synchronous operation mode, the master timer can be used only in the PWM mode, external trigger pulse output mode and free-running mode.
The slave timers can be used in the free-running mode only.
Setting the external event count mode, one-shot pulse mode, and pulse width measurement mode is prohibited.

TABnEST	Software trigger control
0	No operation
1	In one-shot pulse mode: One-shot pulse software trigger
	In external trigger pulse output mode: Pulse output software trigger
The TABnEST bit functions as a software trigger in the one-shot pulse mode or external trigger pulse output mode (this bit is invalid in any other mode). By setting TABnEST to 1 when TABnCE = 1, a software trigger is issued. Therefore, be sure to set TABnEST to 1 when TABnCE = 1. The TIABn0 pin is used for an external trigger. The read value of the TABnEST bit is always 0.	

TABnEEE	Count clock selection
0	Use the internal clock (clock selected with bits TABnCKS2 to TABnCKS0)
1	Use the external clock from the TIABn0 input pin
The valid edge when TABnEEE = 1 (use the external clock from TIABn0 pin) is specified with bits TABnEES1 and TABnEES0.	

TABnMD2	TABnMD1	TABnMD0	Timer mode selection
0	0	0	Interval timer mode
0	0	1	External event counter mode
0	1	0	External trigger pulse output mode
0	1	1	One-shot pulse mode
1	0	0	PWM mode
1	0	1	Free-running mode
1	1	0	Pulse width measurement mode
1	1	1	Triangular wave PWM mode

Cautions

1. Set bits TABnEEE and TABnMD2 to TABnMD0 when TABnCE = 0. (The same value can be written when TABnCE = 1.) The operation is not guaranteed when rewriting is performed when TABnCE = 1. If rewriting was mistakenly performed, set TABnCE = 0 and then set the bits again.
2. TO (timer output) cannot be used in the external event count mode.
3. The external event count input is selected in the external event count mode regardless of the TABnEEE bit value.
4. When using the external trigger pulse output mode, one-shot pulse output mode or pulse width measurement mode, select the internal clock as the count clock (TABnEEE = 0).
5. To use the external event count mode, disable edge detection of the TOAAn0 capture input by clearing the TABnEES1 and TABnEES0 bits of the TABnIOC2 register to 00.

10.5.3 TABnIOC0 - TAB dedicated I/O control register 0

The TABnIOC0 register is an 8-bit register that controls the timer output.

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
TABnOL3	TABnOE3	TABnOL2	TABnOE2	TABnOL1	TABnOE1	TABnOL0	TABnOE0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

TABnOLm	Timer output level setting
0	Normal output
1	Inverted output

TABnOEm	Timer output setting
0	Disable timer output (TOABnm pin outputs low level when TABnOLm = 0, and high level when TABnOLm = 1).
1	Enable timer output (TOABnm pin outputs pulses).

Cautions

1. Rewrite bits TABnOLm and TABnOEm when TABnCE = 0. (The same value can be written when TABnCE = 1.) If rewriting was mistakenly performed, set TABnCE = 0 and then set the bits again.
2. To enable the timer output, be sure to set the corresponding alternate-function pins TABnIS7 to TABnIS0 of the TABnIOC1 register to “Detect no edge” and invalidate the capture operation. Then set the corresponding alternate-function port to output mode.
3. If the pin is used in control output mode, the output level of the TOABnm pin changes along with the TABnOLm bit manipulation even when TABnCE = 0 and TABnOEm = 0.

Note m = 0 to 3

10.5.4 TABnIOC1 - TAB dedicated I/O control register 1

The TABnIOC1 register is an 8-bit register that controls the valid edge of the external input signals (TIABn0 to TIABn3).

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 cleared by any reset.

7	6	5	4	3	2	1	0
TABnIS7	TABnIS6	TABnIS5	TABnIS4	TABnIS3	TABnIS2	TABnIS1	TABnIS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

TABnIS7	TABnIS6	Capture input (TIABn3) valid edge setting
0	0	Detect no edge (capture operation is invalid)
0	1	Detect rising edge.
1	0	Detection of falling edge
1	1	Detection of both edges

TABnIS5	TABnIS4	Capture input (TIABn2) valid edge setting
0	0	No edge detection (capture operation is invalid)
0	1	Detection of rising edge
1	0	Detection of falling edge
1	1	Detection of both edges

TABnIS3	TABnIS2	Capture input (TIABn1) valid edge setting
0	0	No edge detection (capture operation is invalid)
0	1	Detection of rising edge
1	0	Detection of falling edge
1	1	Detection of both edges

TABnIS1	TABnIS0	Capture input (TIABn0) valid edge setting
0	0	No edge detection (capture operation is invalid)
0	1	Detection of rising edge
1	0	Detection of falling edge
1	1	Detection of both edges

Cautions

1. Rewrite bits TABnIS3 to TABnIS0 when TABnCE = 0. (The same value can be written when TABnCE = 1.) If rewriting was mistakenly performed, set TABnCE = 0 and then set the bits again.
2. The TABnIS7 to TABnIS0 bits are valid only in the free-running capture mode and pulse width measurement mode. A capture operation is not performed in any other mode.

Rewrite during timer operation

If the edge specification for the capture operation will be changed while the timer remains in operation (TABnCTL0.TABnCE = 1), only a single bit of the edge specification bits TABnIOC1.TABnIS[k:i] of a dedicated capture input may be changed with a single write operation.

Consequently, proceed as follows (TIABn0 is used as an example):

- Change from rising edge to falling edge:
  - current status is TABnIOC1.TABnIS[1:0] = 01_B: “rising edge”
  - set TABnIOC1.TABnIS[1:0] = 00_B: specify “no edge”
  - set TABnIOC1.TABnIS[1:0] = 10_B: specify “falling edge”
- Change from falling edge to rising edge:
  - current status is TABnIOC1.TABnIS[1:0] = 10_B: “falling edge”
  - set TABnIOC1.TABnIS[1:0] = 00_B: specify “no edge”
  - set TABnIOC1.TABnIS[1:0] = 01_B: specify “rising edge”
- Change from rising or falling edge to both edges:

– current status is TABnIOC1.TABnIS[1:0] = 01_B or 10_B: “rising” or “falling edge”
– set TABnIOC1.TABnIS[1:0] = 11_B: specify “both edges”

Note [k:i] = 7:6, 5:4, 3:2, or 1:0.

10.5.5 TABnIOC2 - TAB dedicated I/O control register 2

The TABnIOC2 register is an 8-bit register that controls the valid edge of the external event count input signal (TIABn0) and external trigger input signal (TIABn0).

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 cleared by any reset.

7	6	5	4	3	2	1	0
0	0	0	0	TABnEES1	TABnEES0	TABnETS1	TABnETS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

TABnEES1	TABnEES0	Setting of valid edge of external event count input (TIABn0)
0	0	Detect no edge (external event count is invalid).
0	1	Detection of rising edge
1	0	Detection of falling edge
1	1	Detection of both edges

TABnETS1	TABnETS0	Setting of valid edge of external trigger input (TIABn0)
0	0	Detect no edge (external trigger is invalid).
0	1	Detection of rising edge
1	0	Detection of falling edge
1	1	Detection of both edges

Cautions

1. Rewrite bits TABnEES1, TABnEES0, TABnEST1, and TABnEST0 when TABnCE = 0. (The same value can be written when TABnCE = 1.) If rewriting was mistakenly performed, set TABnCE = 0 and then set the bits again.
2. The TABnEES1 and TABnEES0 bits are valid when TABnEEE = 1 or when the external event count mode is set (TABnMD[2:0]=001).
3. The TABnETS1 and TABnETS0 bits are valid when the external trigger pulse output mode or one-shot pulse mode is set (TABnMD[2:0]=010 or 011).

Rewrite during timer operation If the edge specification for the external event counter input signal or the external trigger input signal will be changed while the timer remains in operation (TABnCTL0.TABnCE = 1), only a single bit of the edge specification

bits TABnIOC2.TABnEES[k:i] / TABnIOC2.TABnETS[k:i] of a dedicated signal input may be changed with a single write operation.

Proceed as follows for the external event counter mode:

- Change from rising edge to falling edge:
  - current status is TABnIOC2.TABnEES[1:0] = 01_B: “rising edge”
  - set TABnIOC2.TABnEES[1:0] = 00_B: specify “no edge”
  - set TABnIOC2.TABnEES[1:0] = 10_B: specify “falling edge”
- Change from falling edge to rising edge:
  - current status is TABnIOC2.TABnEES[1:0] = 10_B: “falling edge”
  - set TABnIOC2.TABnEES[1:0] = 00_B: specify “no edge”
  - set TABnIOC2.TABnEES[1:0] = 01_B: specify “rising edge”

Proceed as follows for the external trigger pulse output mode:

- Change from rising edge to falling edge:
  - current status is TABnIOC2.TABnETS[1:0] = 01_B: “rising edge”
  - set TABnIOC2.TABnETS[1:0] = 00_B: specify “no edge”
  - set TABnIOC2.TABnETS[1:0] = 10_B: specify “falling edge”
- Change from falling edge to rising edge:
  - current status is TABnIOC2.TABnETS[1:0] = 10_B: “falling edge”
  - set TABnIOC2.TABnETS[1:0] = 00_B: specify “no edge”
  - set TABnIOC2.TABnETS[1:0] = 01_B: specify “rising edge”

Note [k:i] = 7:6, 5:4, 3:2, or 1:0

10.5.6 TABnIOC4 - TAB I/O control register 4

The TABnIOC4 register is an 8-bit register that controls the output function of Timer AB.

TABnIOC4 can be used only when the interval mode or the free-running compare mode is selected. In other modes, set this register to 00_H.

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

Address <base>+ 10_H

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

7	6	5	4	3	2	1	0
TABnOS3	TABnOR3	TABnOS2	TABnOR2	TABnOS1	TABnOR1	TABnOS0	TABnOR0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

TABnOSm	TABnORm	Toggle Control of TOABnm
0	0	Standard operation.
0	1	Force output level to inactive at next toggle event
1	0	Force output level to active at next toggle event
1	1	Freeze current output level.

Note
1. After forcing the output level to either active or inactive, the TOABnm maintains this level (=no toggling afterwards) until the TABnOSm and TABnORm are cleared to standard operation.
2. Forcing an output level to an active or inactive state occurs at the time of the next upcoming toggle event, while a freeze becomes effective immediately.
3. Writing to TABnIOC4 is also possible, when TABnCTL0.TABnCE = 1.
4. m = 0 to 3.

10.5.7 TABnOPT0 - TAB option register 0

The TABnOPT0 register is an 8-bit register that selects a capture or compare operation, and detects 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 cleared by any reset.

7	6	5	4	3	2	1	0
TABnCCS3	TABnCCS2	TABnCCS1	TABnCCS0	0	0	0	TABnOVF
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

TABnCCSm	Selection of capture or compare operation of TABnCCRm register
0	Compare register
1	Capture register
The TABnCCSm bit setting is valid only in the free-running mode.	

TABnOVF	Detection of Timer AB overflow
Set (1)	Overflow occurred
Reset (0)	0 written to TABnOVF bit or TABnCE = 0

- The TABnOVF bit is set when the 16-bit counter value overflows from $FFFF_H$ to 0000_H in the free-running mode or the pulse width measurement mode.
- As soon as the TABnOVF bit has been set to 1, an interrupt request signal (INTTABnOV) is generated. The INTTABnOV signal is not generated in any mode other than the free-running mode and pulse width measurement mode.
- The TABnOVF bit is not cleared even when the TABnOVF bit and the TABnOPT0 register are read when TABnOVF = 1.
- The TABnOVF bit can be both read and written, but 1 cannot be written to the TABnOVF bit from the CPU. Writing 1 has no influence on the operation of timer AB.

Cautions

1. Rewrite bits TABnCCS3 to TABnCCS0 when TABnCE = 0. (The same value can be written when TABnCE = 1.) If rewriting was mistakenly performed, set TABnCE = 0 and then set the bits again.
2. Be sure to clear bits 1, 2 and 3 to 0.

Note m = 0 to 3

10.5.8 TABnCCR0 - TAB capture/compare register 0

The TABnCCR0 register is a 16-bit register that has a capture function and a compare function.

Whether this register is used as a capture register or a compare register can be specified by using the TABnCCS0 bit of the TABnOPT0 register, but only in the free-running mode.

In the pulse width measurement mode, this register can be used only as a capture register (it cannot be used as a compare register).

In all the modes other than the free-running mode and pulse width measurement mode, this register functions as a compare register.

Access This register can be read/written in 16-bit units.

Address <base>+ 6_H

Initial Value 0000_H. This register is cleared by any reset. In the default status, the TABnCCR0 register functions as a compare register.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	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/W

- When used as a compare register

  TABnCCR0 can be rewritten when TABnCE = 1 as mentioned in this table:

TAB Operation Mode	Method of Writing TABnCCR0 Register
PWM output mode, external trigger pulse output mode, or triangular wave PWM mode	Reload
Free-running mode, external event count mode, one-shot pulse mode or interval timer mode	Any time write
Pulse width measurement mode	Cannot be used because used only as capture register

- When used as capture register

  The count value is stored in TABnCCR0 upon capture trigger (TIABn0) input edge detection.

Note The value of TABnCCR0 register can be read/written when TABnCE bit of TABn control register 0 (TABnCTL0) equals 1.

10.5.9 TABnCCR1 - TAB capture/compare register 1

The TABnCCR1 register is a 16-bit register that has a capture function and a compare function.

Whether this register is used as a capture register or a compare register can be specified by using the TABnCCS1 bit of the TABnOPT0 register, but only in the free-running mode.

In the pulse width measurement mode, this register can be used only as a capture register (it cannot be used as a compare register).

In all the modes other than the free-running mode and pulse width measurement mode, this register functions as a compare register.

Caution In the one-shot pulse mode, it is prohibited to set the TABnCCR1 register to 0000_H.

Access This register can be read/written in 16-bit units.

Address <base>+ 8_H

Initial Value 0000_H. This register is cleared by any reset. In the default status, the TABnCCR1 register functions as a reload register.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	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/W

- When used as a compare register

  TABnCCR1 can be rewritten when TABnCE = 1, as mentioned in this table:

TAB Operation Mode	Method of Writing TABnCCR1 Register
PWM output mode, external trigger pulse output mode, or triangular wave PWM mode	Reload
Free-running mode, external event count mode, one-shot pulse mode or interval timer mode	Any time write
Pulse width measurement mode	Cannot be used because used only as capture register

- When used as a capture register

  The count value is stored in TABnCCR1 upon capture trigger (TIABn1) input edge detection.

Note The value of TABnCCR1 register can be read/written when TABnCE bit of TABn control register 0 (TABnCTL0) equals 1.

10.5.10 TABnCCR2 - TAB capture/compare register 2

The TABnCCR2 register is a 16-bit register that has a capture function and a compare function.

Whether this register is used as a capture register or a compare register can be specified by using the TABnCCS2 bit of the TABnOPT0 register, but only in the free-running mode.

In the pulse width measurement mode, this register can be used only as a capture register (it cannot be used as a compare register).

In all the modes other than the free-running mode and pulse width measurement mode, this register functions as a compare register.

Access This register can be read/written in 16-bit units.

Address <base>+ 8_H

Initial Value 0000_H. This register is cleared by any reset. In the default status, the TABnCCR2 register functions as a compare register.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	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/W

- When used as a compare register

  TABnCCR2 can be rewritten when TABnCE = 1, as mentioned in this table:

TAB Operation Mode	Method of Writing TABnCCR2 Register
PWM output mode, external trigger pulse output mode, or triangular wave PWM mode	Reload
Free-running mode, external event count mode, one-shot pulse mode or interval timer mode	Any time write
Pulse width measurement mode	Cannot be used because used only as capture register

- When used as capture register

  The count value is stored in TABnCCR2 upon capture trigger (TIABn2) input edge detection.

Note The value of TABnCCR2 register can be read/written when TABnCE bit of TABn control register 0 (TABnCTL0) equals 1.

10.5.11 TABnCCR3 - TAB capture/compare register 3

The TABnCCR3 register is a 16-bit register that has a capture function and a compare function.

Whether this register is used as a capture register or a compare register can be specified by using the TABnCCS3 bit of the TABnOPT0 register, but only in the free-running mode.

In the pulse width measurement mode, this register can be used only as a capture register (it cannot be used as a compare register).

In all the modes other than the free-running mode and pulse width measurement mode, this register functions as a compare register.

Access This register can be read/written in 16-bit units.

Address <base>+ C_H

Initial Value 0000_H. This register is cleared by any reset. In the default status, the TABnCCR3 register functions as a compare register.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	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/W

- When used as a compare register

  TABnCCR3 can be rewritten when TABnCE = 1, as mentioned in this table:

TAB Operation Mode	Method of Writing TABnCCR2 Register
PWM output mode, external trigger pulse output mode, or triangular wave PWM mode	Reload
Free-running mode, external event count mode, one-shot pulse mode or interval timer mode	Any time write
Pulse width measurement mode	Cannot be used because used only as capture register

- When used as capture register

  The count value is stored in TABnCCR3 upon capture trigger (TIABn3) input edge detection.

Note The value of TABnCCR3 register can be read/written when TABnCE bit of TABn control register 0 (TABnCTL0) equals 1.

10.5.12 TABnCNT - TAB timer read buffer register

The TABnCNT register is a timer read buffer register that can read 16-bit counter values.

Access This register can be read in 16-bit units. The value of this register is read when TABnCE bit = 1.

Address <base>+ E_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
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

10.6 Operation

Timer AB can perform the following operations.

Operation	TABnEST Software trigger input	TIABn0 External trigger input	TABnEEE Count clock selection	Capture/ Compare Write	Compare Write
Interval timer mode	Invalid	Invalid	Internal/TIABn0 pin	Compare only	Anytime write
External event counter mode[Note 1]	Invalid	Invalid	TIABn0 pin only	Compare only	Anytime write
External trigger pulse output mode[Note 2]	Valid	Valid	Internal only	Compare only	Reload
One-shot pulse output mode[Note 2]	Valid	Valid	Internal only	Compare only	Any time write
PWM mode	Invalid	Invalid	Internal/TIABn0 pin	Compare only	Reload
Free-running mode	Invalid	Invalid	Internal/TIABn0 pin	Capture/compare switching enabled	Any time write
Pulse width measurement mode[Note 2]	Invalid	Invalid	Internal only	Capture only	Not applicable

Note
1. When using the external event count function, set TIABn0 capture input edge detection to “Detect no edge”. (Set TABnIS1 and TABnIS0 bits of TABnIOC1 register to “00”.)
2. When using the external trigger pulse output mode, one-shot pulse mode, or pulse width measurement mode, select the internal clock as the count clock (by setting the TABnEEE bit of the TABnCTL1 register to 1).

Caution Clearing the TABnCCR1 register to 0000_H is prohibited in the one-shot pulse mode.

10.6.1 Anytime write and reload

Timer AB allows rewriting of the TABnCCR0 to TABnCCR3 registers while the timer is operating (TABCE = 1). These registers are written differently (anytime write or reload) depending on the mode.

(1) Anytime write

When data is written to the TABnCCR0 to TABnCCR3 registers during timer operation, it is transferred at any time to the CCR0 buffer register and is compared with the value of the 16-bit counter.

This flowchart illustrates an example of the operation in the interval timer mode.

Figure 10-6 Flowchart of basic operation for anytime write

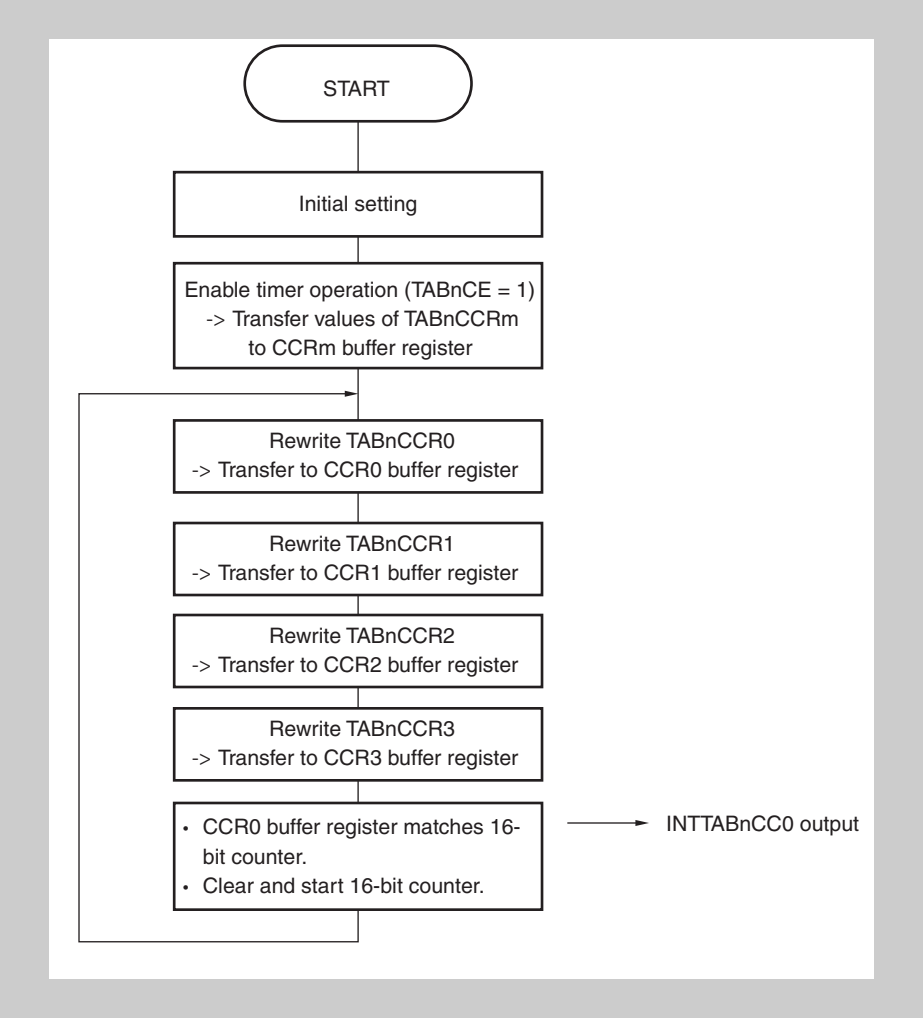
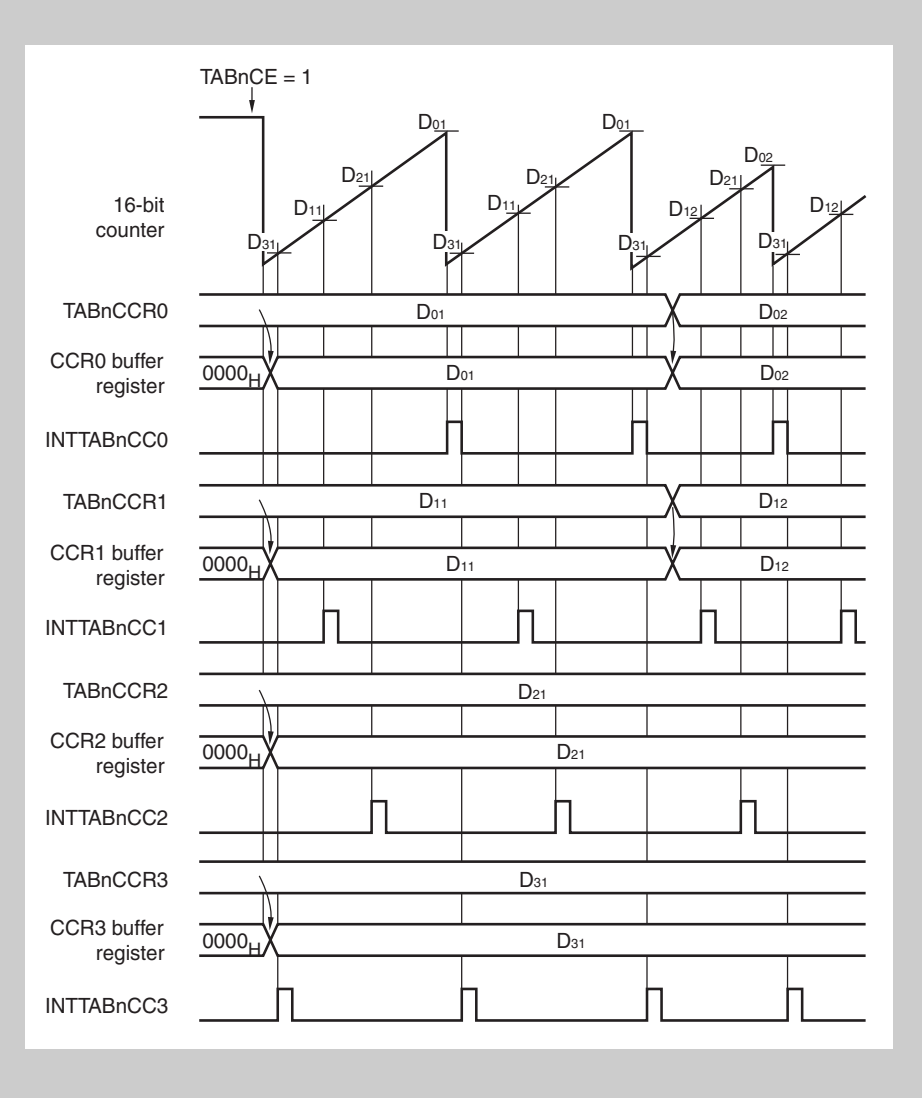


Figure 10-7 Timing chart of anytime write



Note 1. D01, D02: Setting values of TABnCCR0 register (0000_H to $FFFF_H$)
D11, D12: Setting values of TABnCCR1 register (0000_H to $FFFF_H$)
D21: Setting value of TABnCCR2 register (0000_H to $FFFF_H$)
D31: Setting value of TABnCCR3 register (0000_H to $FFFF_H$)

2. The above timing chart illustrates an example of interval timer mode operation.

(2) Reload

When data is written to the TABnCCRm register during timer operation, the written data is held until the specific conditions are met, then transferred to the CCRm buffer register to be compared with the value of the 16-bit counter.

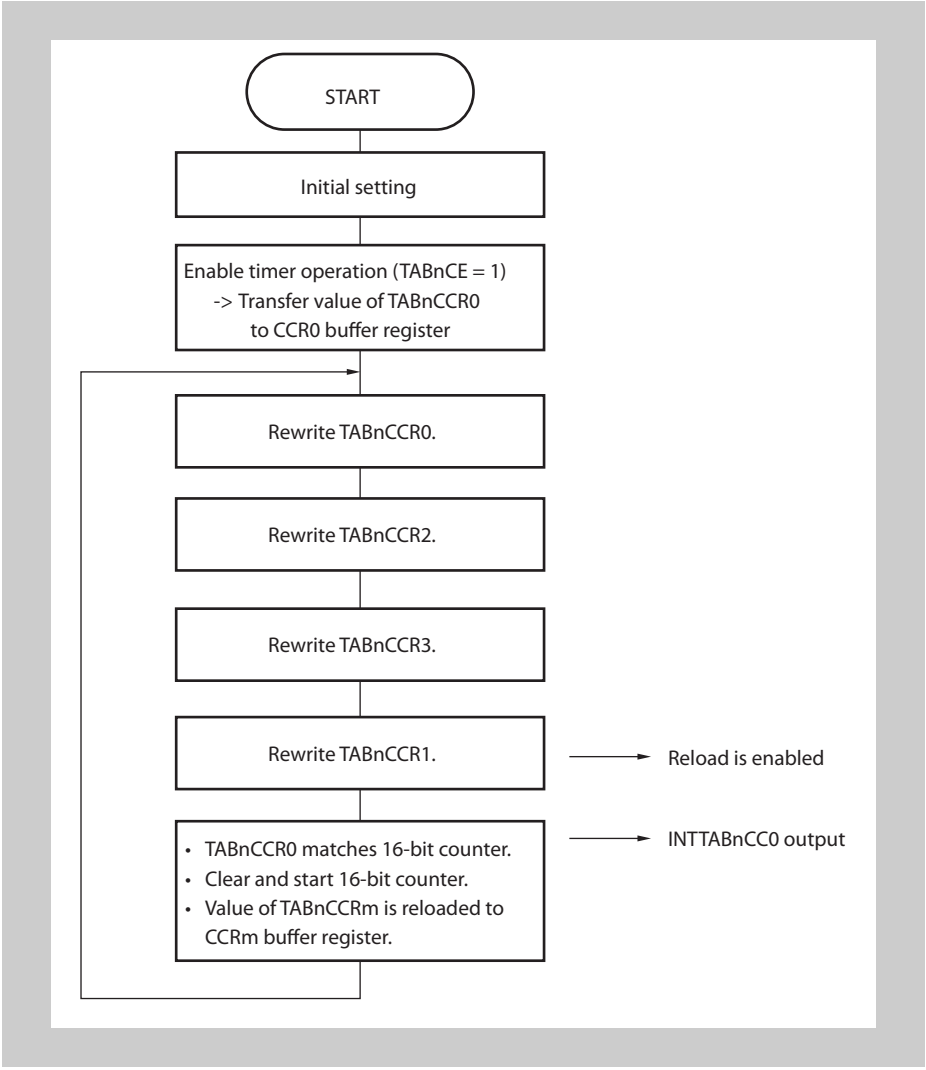
In order that the set values of the TABnCCRm register are compared with the value of the 16-bit counter (the set values are reloaded to the CCRm buffer register), the value of the TABnCCR0 register must be rewritten and then a value must be written to the TABnCCR1 register before the value of the 16-bit counter matches the value of the CCRm buffer register.

When the value of the CCRm buffer register matches the value of the 16-bit counter, the value of the TABnCCRm register is reloaded to the CCRm buffer register.

Whether the next reload timing is made valid or not is controlled by writing to the TABnCCR1 register.Therefore, to rewrite even only one of the TABnCCR0, TABnCCR2 and TABnCCR3 registers, the same value (the value already set to theTABnCCR1 register) must be set to the TABnCCR1 register.

This flowchart illustrates an example of PWM mode operation.

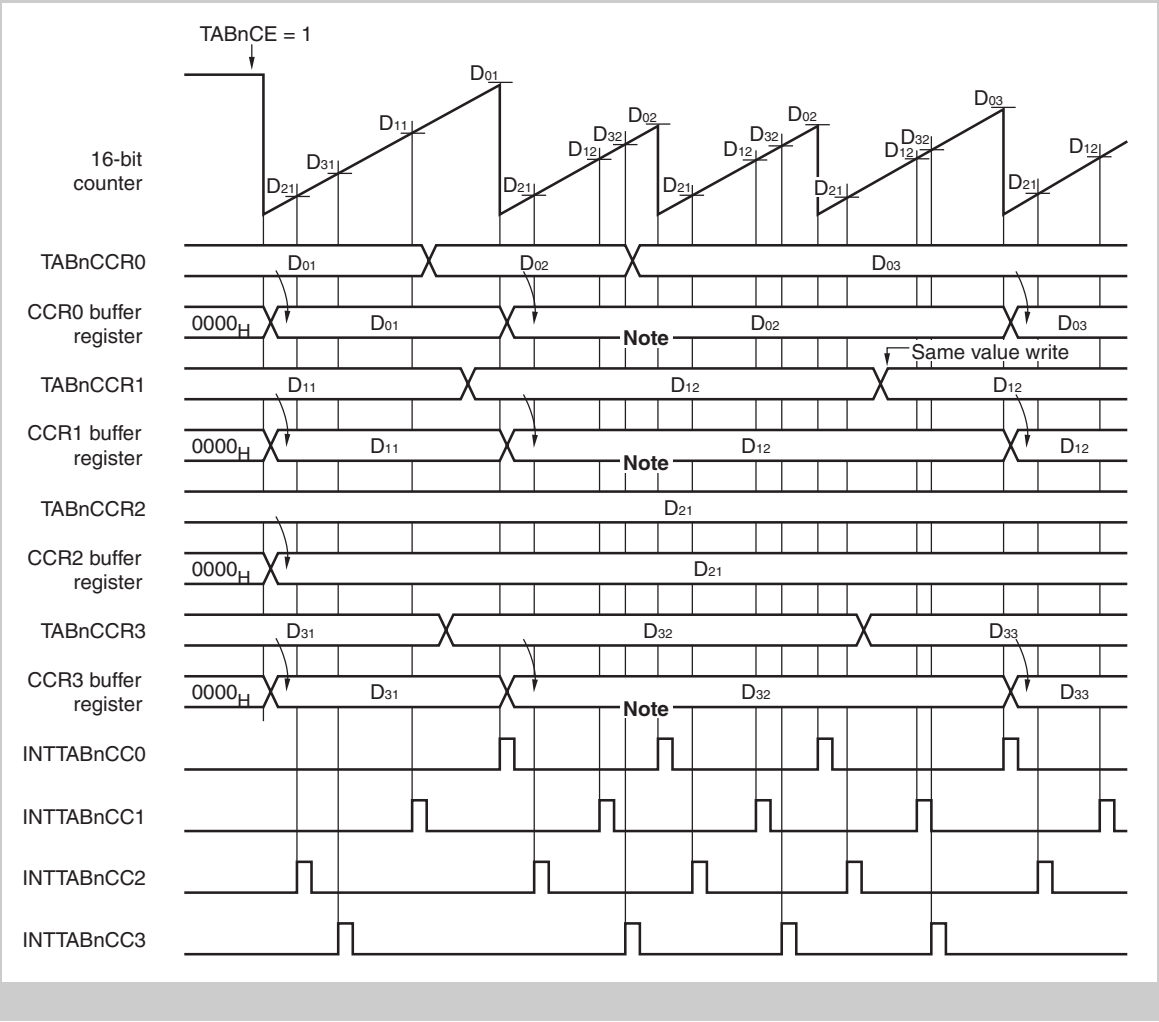
Figure 10-8 Flowchart of basic operation for reload



Caution Writing the TABnCCR1 register includes an operation to enable reload. Therefore, rewrite the TABnCCR1 register after rewriting other TABnCCR registers.

Note 1. m=0 to 3

Figure 10-9 Timing chart of reload



Note 1. Reload is not performed because TABnCCR1 register is not written.

2. D01, D02, D03: Setting values of TABnCCR0 register (0000_H to $FFFF_H$)
D11, D12: Setting values of TABnCCR1 register (0000_H to $FFFF_H$)
D21: Setting value of TABnCCR2 register (0000_H to $FFFF_H$)
D31, D32, D33: Setting values of TABnCCR3 register (0000_H to $FFFF_H$)

3. The flowchart above illustrates the operation in the PWM mode.

10.6.2 Interval timer mode (TABnMD2 to TABnMD0 = 000)

In the interval timer mode, an interrupt request signal (INTTABnCC0) is generated when the set value of the TABnCCR0 register matches the value of the 16-bit counter, and the 16-bit counter is cleared. Rewriting the TABnCCRm register is enabled when TABnCE = 1. When a value is set to the TABnCCRm register by a write instruction from the CPU, it is transferred to the CCRm buffer register by means of anytime write, and is compared with the value of the 16-bit counter.

In the interval timer mode, the 16-bit counter can be cleared only when its value matches the value of the CCR0 buffer register.

The 16-bit counter is not cleared by using the TABnCCRk register.

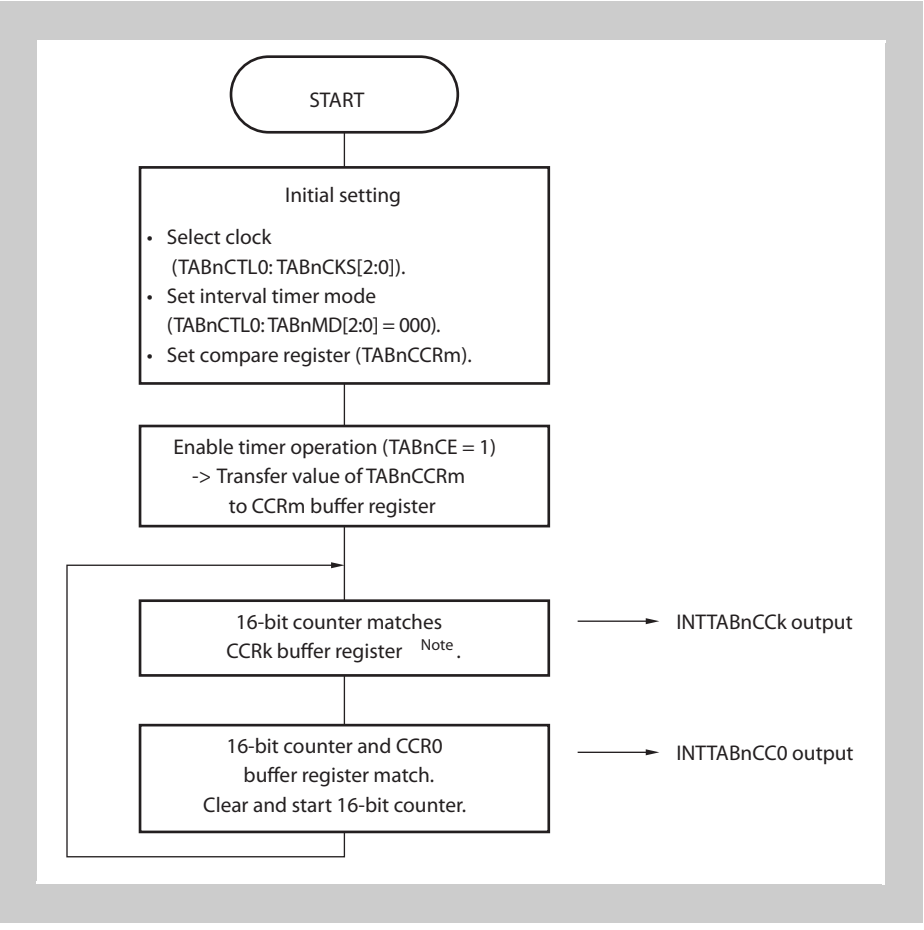
However, the set value of the TABnCCRk register is transferred to the CCRk buffer register and compared with the value of the 16-bit counter. As a result, an interrupt request (INTTABnCCk) is generated. The value can also be output from the TOABnm pin by setting the TABnOEm bit to 1.

When the TABnCCRk register is not used, it is recommended to set the TABnCCRk register to $FFFF_H$.

When performing timer output with the TOABnk pin, set the same values to the TABnCCR0 register and one of the TABnCCR1 to TABnCCR3 registers since the 16-bit timer counter cannot be cleared with the TABnCCRk register.

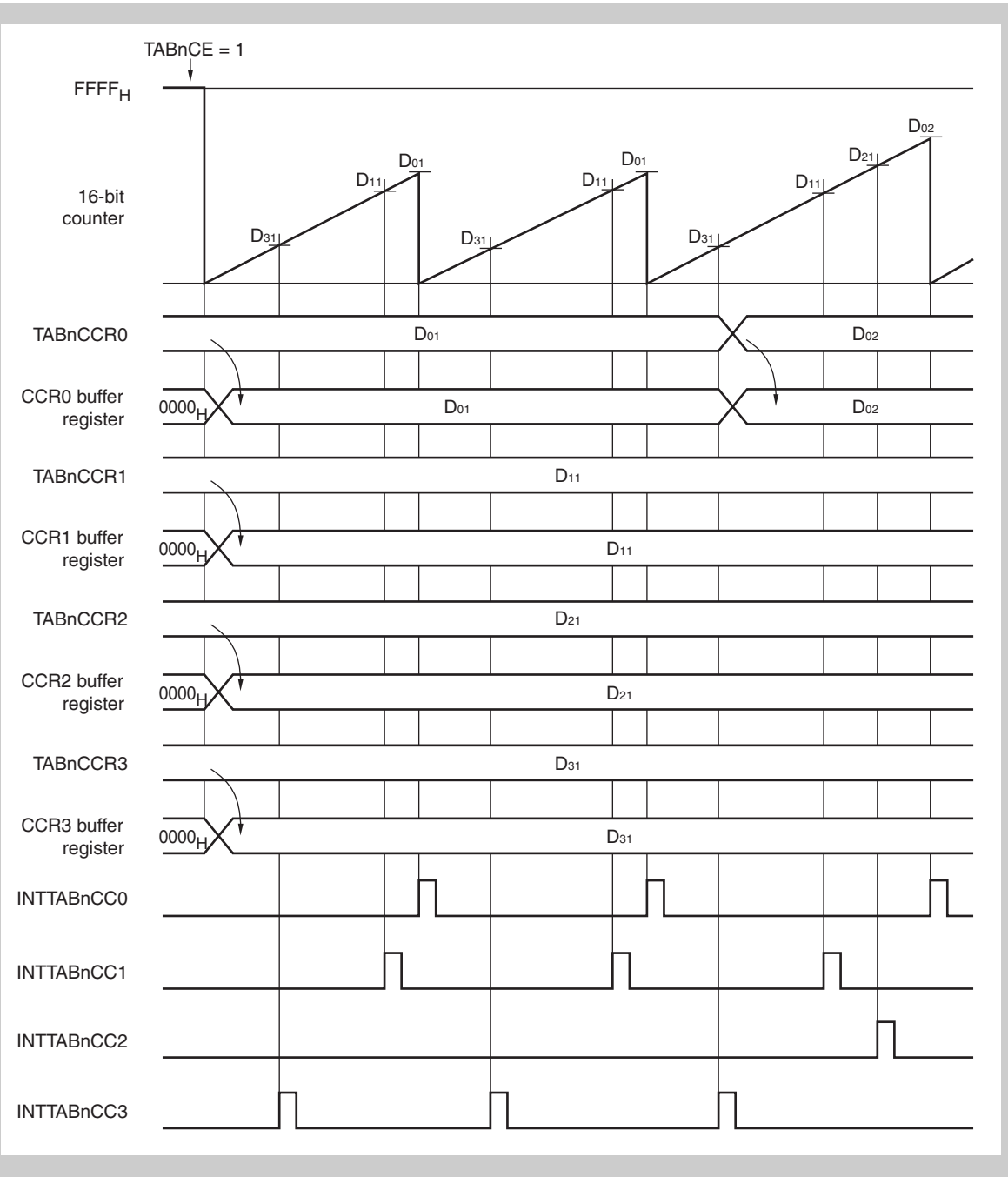
Note m = 0 to 3; k = 1 to 3

Figure 10-10 Flowchart of basic operation in interval timer mode



Note The 16-bit counter is not cleared upon a match between the 16-bit counter and TABnCCRk.

Figure 10-11 Basic operation timing in interval timer mode (1/2) (When only TABnCCR0 register value is rewritten and TOABnm is not output)

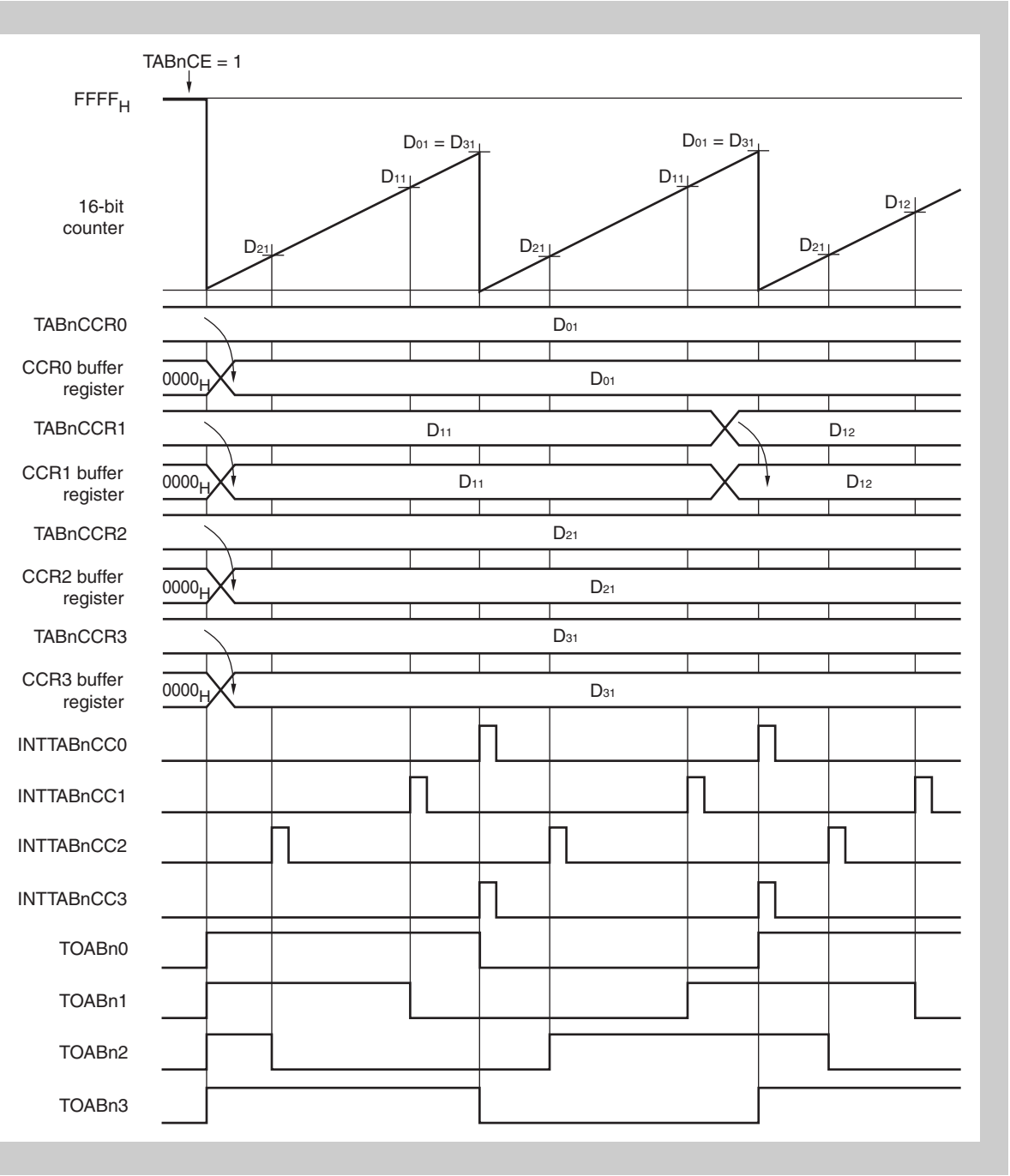


Note 1. D01, D02: Setting values of TABnCCR0 register (0000_H to $FFFF_H$)
D11: Setting value of TABnCCR1 register (0000_H to $FFFF_H$)
D21: Setting value of TABnCCR2 register (0000_H to $FFFF_H$)
D31: Setting value of TABnCCR3 register (0000_H to $FFFF_H$)

2. Interval time = (Dmk + 1) × (count clock cycle)

3. m = 0 to 3; k = 1 to 3

Figure 10-12 Basic operation timing in interval timer mode (2/2) (when D01 = D31, only TABnCCR1 register value is rewritten, and TOABnm is output)



Note 1. D01: Setting value of TABnCCR0 register (0000_H to $FFFF_H$)
D11, D12: Setting values of TABnCCR1 register (0000_H to $FFFF_H$)
D21: Setting value of TABnCCR2 register (0000_H to $FFFF_H$)
D31: Setting value of TABnCCR3 register (0000_H to $FFFF_H$)

2. Interval time = (Dmk + 1) × (count clock cycle)

3. m = 0 to 3; k = 1 to 3

10.6.3 External event counter mode (TABnMD2 to TABnMD0 = 001)

In the external event count mode, the external event count input (TIABn0 pin input) is used as a count-up signal.

Regardless of the setting of the TABnEEE bit of the TABnCTL0 register, 16-bit timer/event counter TAB counts up the external event count input (TIABn0 pin input) when it is set in the external event count mode.

In the external event count mode, an interrupt request (INTTABnCC0) is generated when the set value of the TABnCCR0 register matches the value of the 16-bit counter, and the value of the 16-bit counter is cleared.

When a value is set to the TABnCCRm register by a write instruction from the CPU, it is transferred to the CCRm buffer register, and is compared with the value of the 16-bit counter.

In the external event count mode, the 16-bit counter can be cleared only when its value matches the value of the CCR0 buffer register.

The 16-bit counter cannot be cleared by using the TABnCCRk register.

However, the set value of the TABnCCRk register is transferred to the CCRk buffer register and is compared with the value of the 16-bit counter. As a result, an interrupt request (INTTABnCCk) is generated.

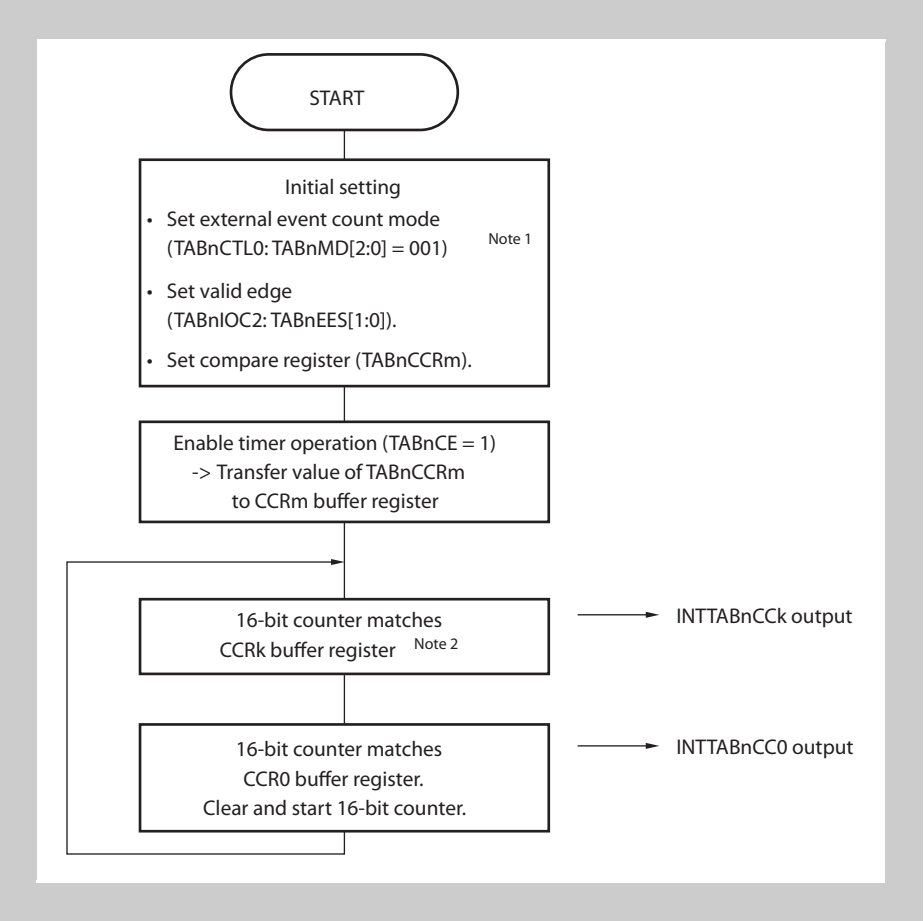
By setting the TABnOEk bit to 1, a signal can be output from the TOABnk pin.

When performing timer output with the TOABnk pin, set the same values to the TABnCCR0 register and the TABnCCRk register since the 16-bit counter cannot be cleared with the CCRk buffer register.

Rewriting the TABnCCR0 register is enabled when TABnCE = 1. When the TABnCCRk register is not used, it is recommended to set TABnCCRk to $FFFF_H$.

Note m = 0 to 3; k = 1 to 3

Figure 10-13 Flowchart of basic operation in external event counter mode

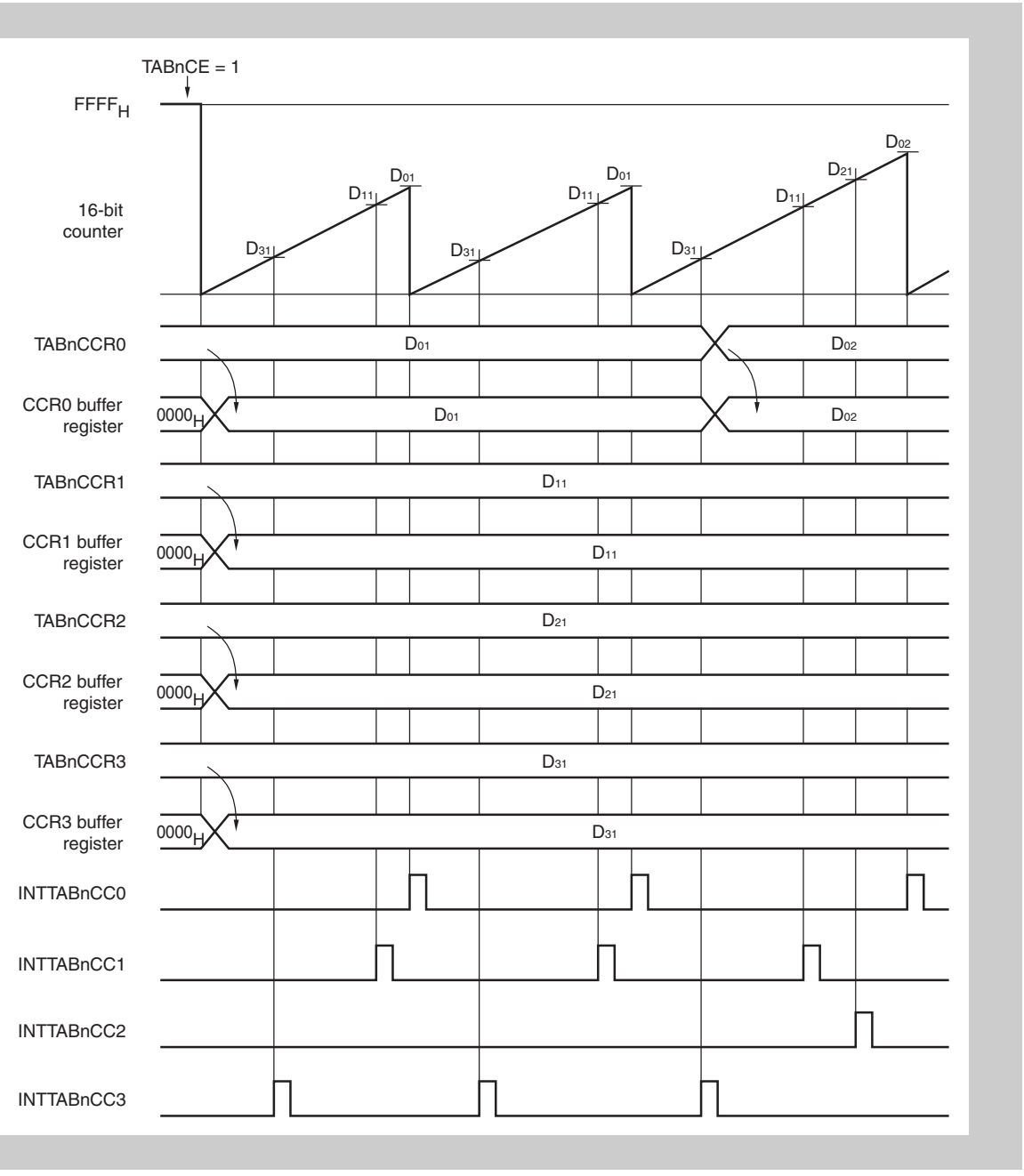


Note 1. Selection of the TABnEEE bit has no influence.

2. The 16-bit counter is not cleared when it matches the CCRk buffer register.

3. m = 0 to 3; k = 1 to 3

Figure 10-14 Basic operation timing in external event counter mode (1/2) (when only TABnCCR0 register value is rewritten and TOABnm is not output)

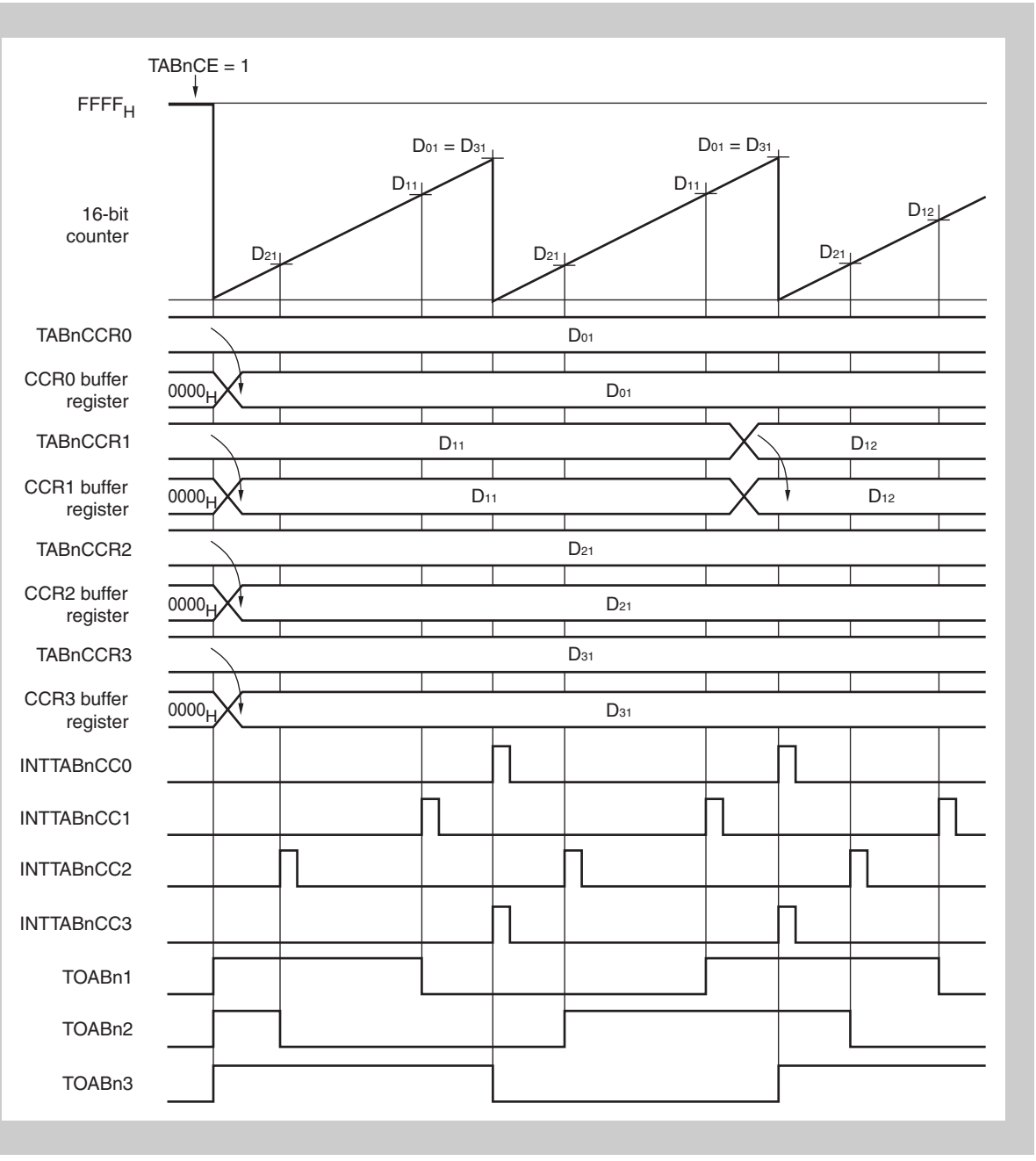


Note 1. LD01, D02: Setting values of TABnCCR0 register (0000_H to $FFFF_H$)
D11: Setting value of TABnCCR1 register (0000_H to $FFFF_H$)
D21: Setting value of TABnCCR2 register (0000_H to $FFFF_H$)
D31: Setting value of TABnCCR3 register (0000_H to $FFFF_H$)

2. Interval time = (Dmk + 1) × (count clock cycle)

3. m = 0 to 3; k = 1 to 3

Figure 10-15 Basic operation timing in external event counter mode (2/2) (when D01 = D31, only TABnCCR1 register is rewritten, and TOABnk is output)



Note 1. D01: Setting value of TABnCCR0 register (0000_H to $FFFF_H$)
D11, D12: Setting values of TABnCCR1 register (0000_H to $FFFF_H$)
D21: Setting value of TABnCCR2 register (0000_H to $FFFF_H$)
D31: Setting value of TABnCCR3 register (0000_H to $FFFF_H$)

2. Interval time = (Dmk + 1) × (count clock cycle)

3. m = 0 to 3; k = 1 to 3

10.6.4 External trigger pulse mode (TABnMD2 to TABnMD0 = 010)

When TABnCE = 1 in the external trigger pulse mode, the 16-bit counter stops at $FFFF_H$ and waits for input of an external trigger (TIABn0 pin input). When the counter detects the edge of the external trigger (TIABn0 pin input), it starts counting up.

The duty factor of the signal output from the TOABnk pin is set by a reload register (TABnCCRk) and the period is set by a compare register (TABnCCR0).

Rewriting the TABnCCRm register is enabled when TABnCE = 1. So that the set value of the TABnCCRm register after rewriting is compared with the value of the 16-bit counter (reloaded to the CCRm buffer register), the TABnCCR0 register must be rewritten and then a value is written to the TABnCCR1 register before the value of the 16-bit counter matches the value of the TABnCCR0 register. When the value of the TABnCCR0 register later matches the value of the 16-bit counter, the value of the TABnCCRm register is reloaded.

Whether the next reload timing is made valid or not is controlled by writing to the TABnCCR1 register. Therefore, write the same value to the TABnCCR1 register when it is necessary to rewrite the value of only the TABnCCR0 register.

Reload is invalid when only the TABnCCR0 register is rewritten.

To stop Timer AB, clear TABnCE to 0. If the edge of the external trigger (TIABn0 pin input) is detected more than once in the external trigger pulse mode, the 16-bit counter is cleared at the point of edge detection, and resumes counting up.

By setting the TABnEST bit of the TABnCTL1 register, we generate a software trigger to realize the same function as the external trigger pulse mode, but using a software trigger instead of external trigger input (TIABn0).

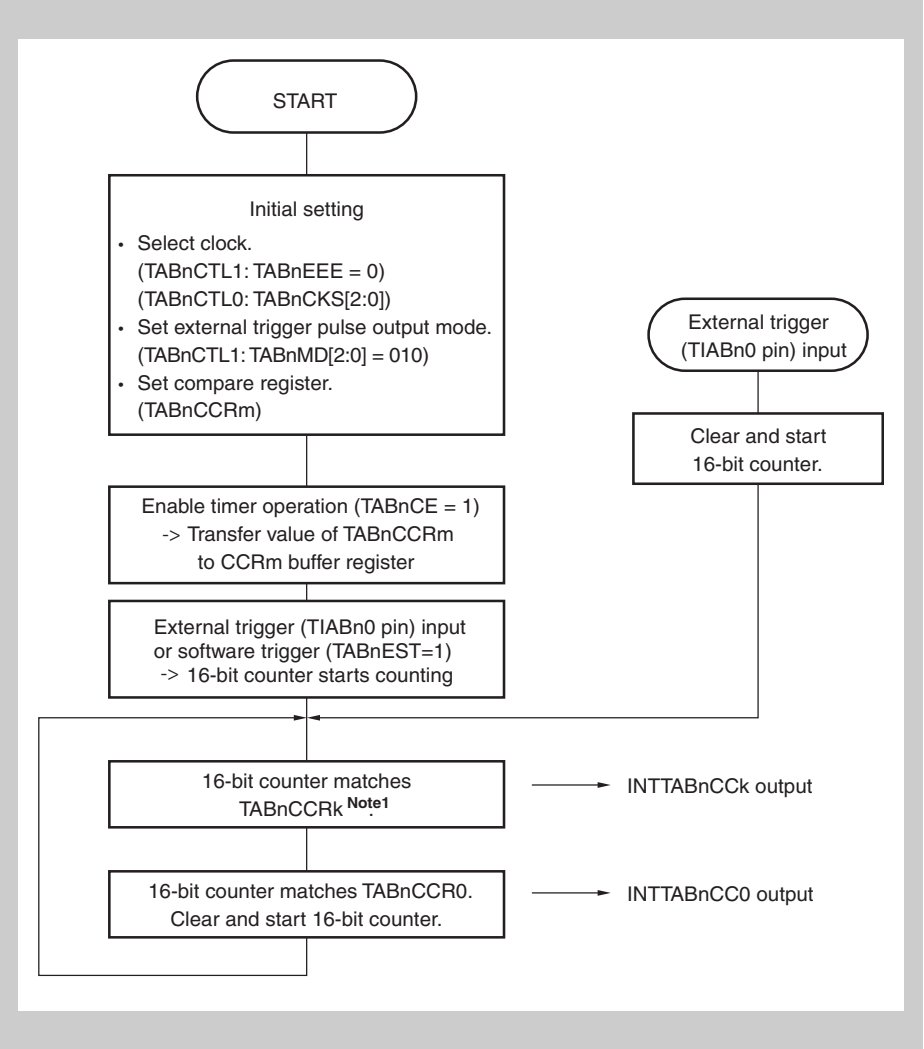
In the external trigger pulse mode, the capture function of the TABnCCRm register cannot be used because this register is used only as a compare register.

Caution In the external trigger pulse mode, select the internal clock (TABnCTL1.TABnEEE bit = 0) as the count clock.

Note 1. For the reload operation when TABnCCRm is rewritten during timer operation, refer to *Section 10.6.1, Anytime write and reload* on page 236.

2. m = 0 to 3; k = 1 to 3

Figure 10-16 Flowchart of basic operation in external trigger pulse output mode

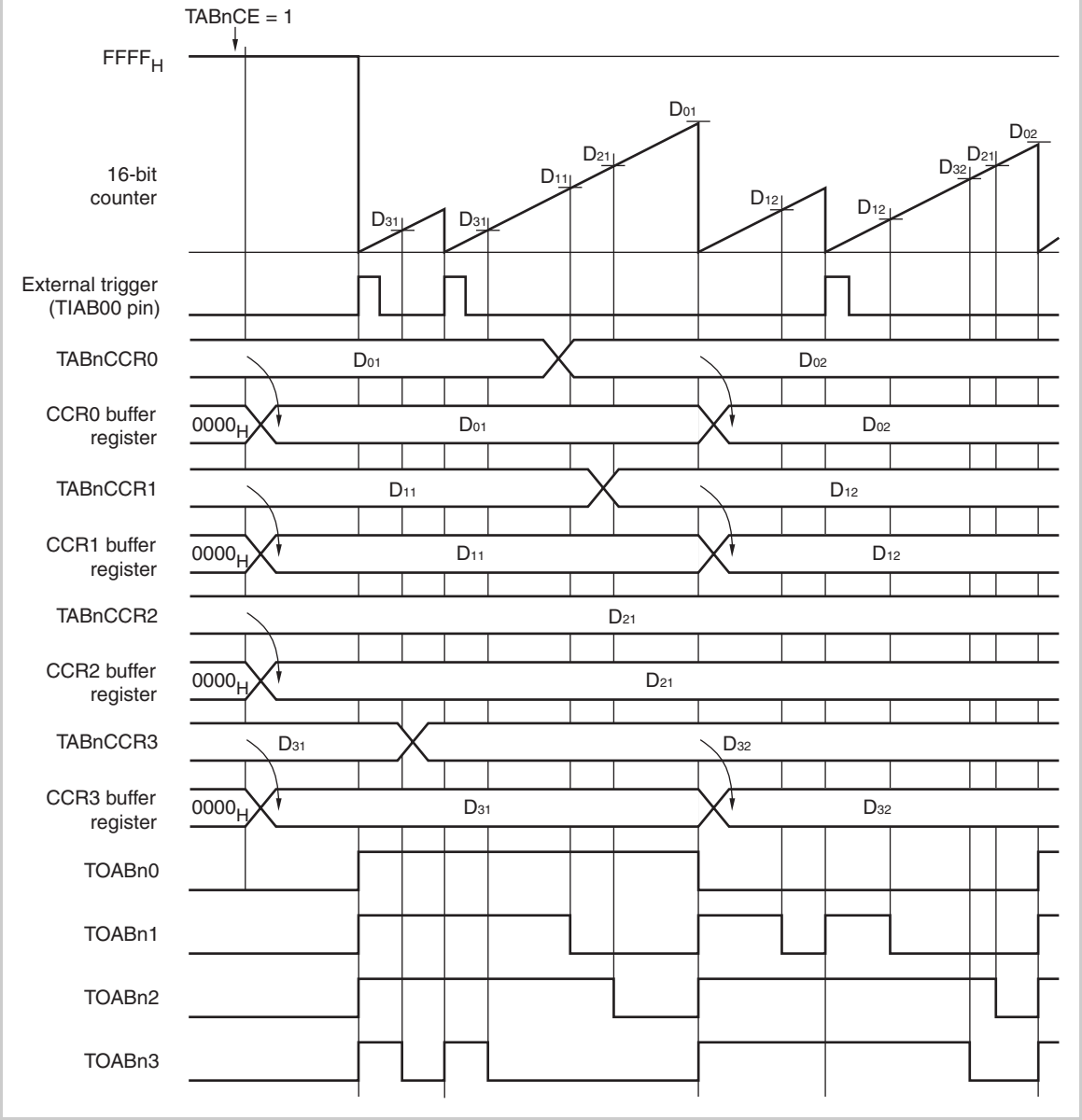


Note 1. The 16-bit counter is not cleared upon a match between the 16-bit counter and the CCRk buffer register.

2. The TABnEST bit of the TABnCTL register can be rewritten during timer operation (TABnCE = 1).

3. m = 0 to 3; k = 1 to 3

Figure 10-17 Basic operation timing in external trigger pulse output mode



Note 1. D01, D02: Setting values of TABnCCR0 register (0000_H to $FFFF_H$)
D11, D12: Setting values of TABnCCR1 register (0000_H to $FFFF_H$)
D21: Setting value of TABnCCR2 register (0000_H to $FFFF_H$)
D31, D32: Setting values of TABnCCR3 register (0000_H to $FFFF_H$)

2. Duty of TOABnk output = (Set value of TABnCCRk register) / (Set value of TABnCCR0 register + 1)
Cycle of TOABnk output = (Set value of TABnCCR0 register + 1) × (Count clock cycle)

3. k = 1 to 3

10.6.5 One-shot pulse mode (TABnMD2 to TABnMD0 = 011)

When TABnCE is set to 1 in the one-shot pulse mode, the 16-bit counter waits for the setting of the TABnEST bit (to 1) or a trigger that is input when the edge of the TIABn0 pin is detected, while holding $FFFF_H$. When the trigger is input, the 16-bit counter starts counting up. When the value of the 16-bit counter matches the value of the CCRk buffer register that has been transferred from the TABnCCR0 register, TOABnk goes high. When the value of the 16-bit counter matches the value of the CCR0 buffer register that has been transferred from the TABnCCR0 register, TOABnk goes low, and the 16-bit counter is cleared to 0000_H and stops. Input of a second or subsequent trigger is ignored while the 16-bit counter is operating. Be sure to input a second trigger while the 16-bit counter is stopped at 0000_H.

In the one-shot pulse mode, rewriting the TABnCCRm register is enabled when TABnCE = 1. The set value of the TABnCCRm register becomes valid after a write instruction from the CPU is executed. They are then transferred to the CCRm buffer register, and compared with the value of the 16-bit counter.

To realize the same function as the external trigger pulse mode by software (software pulse mode), a software trigger can be generated by setting the TABnEST bit of TABnCTL1 to 1.

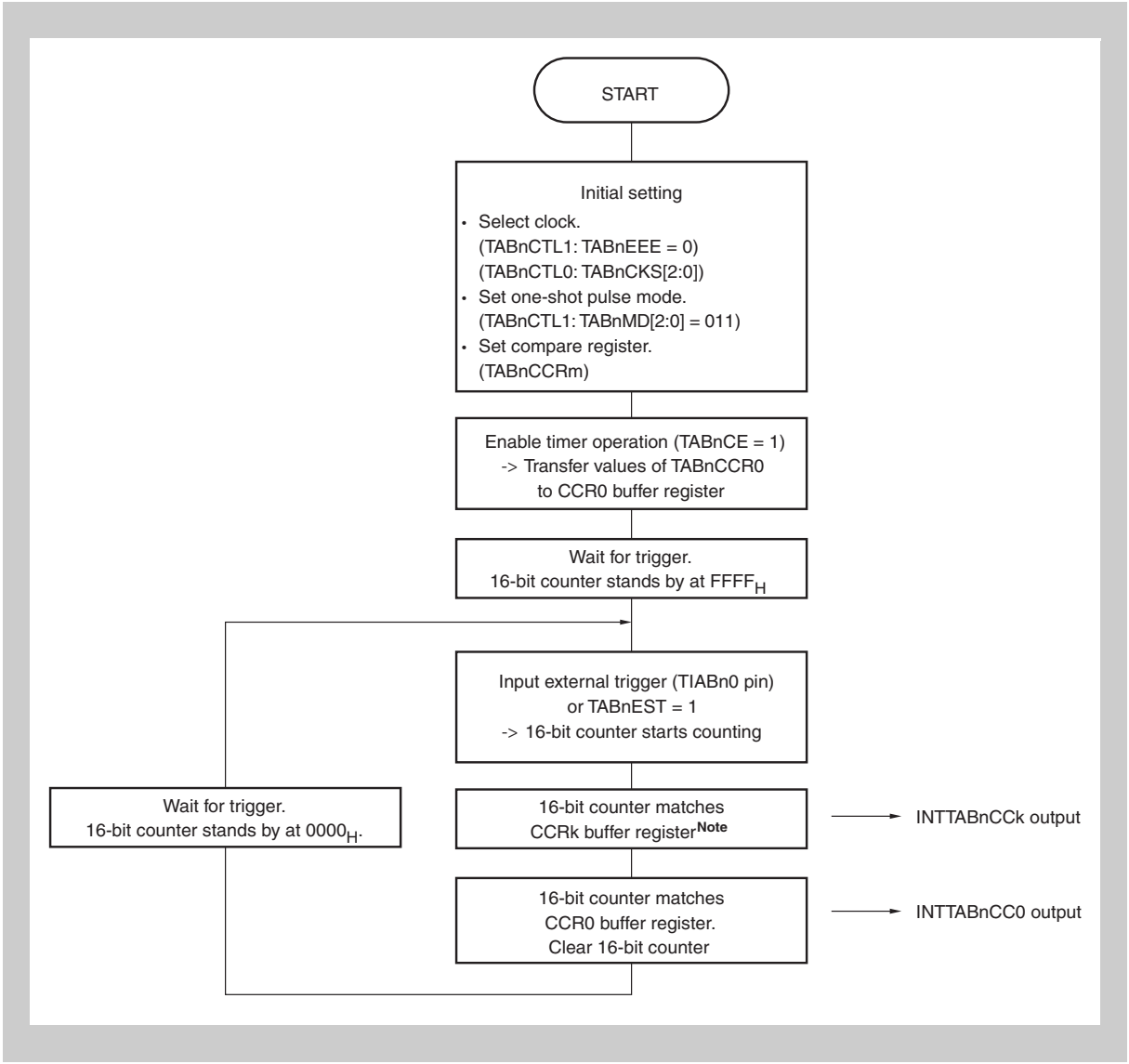
The waveform of the one-shot pulse is output from the TOABnk pin. The TOABnm pin produces a toggle output when the value of the 16-bit counter matches the value of the TABnCCR0 register.

In the one-shot pulse mode, the TABnCCRm register function is only used as a compare register. It cannot be used as a capture register.

Caution In the one-shot pulse mode, select the internal clock (TABnCTL1.TABnEEE bit = 0) for the count clock.

Note m = 0 to 3; k = 1 to 3

Figure 10-18 Flowchart of basic operation in one-shot pulse mode

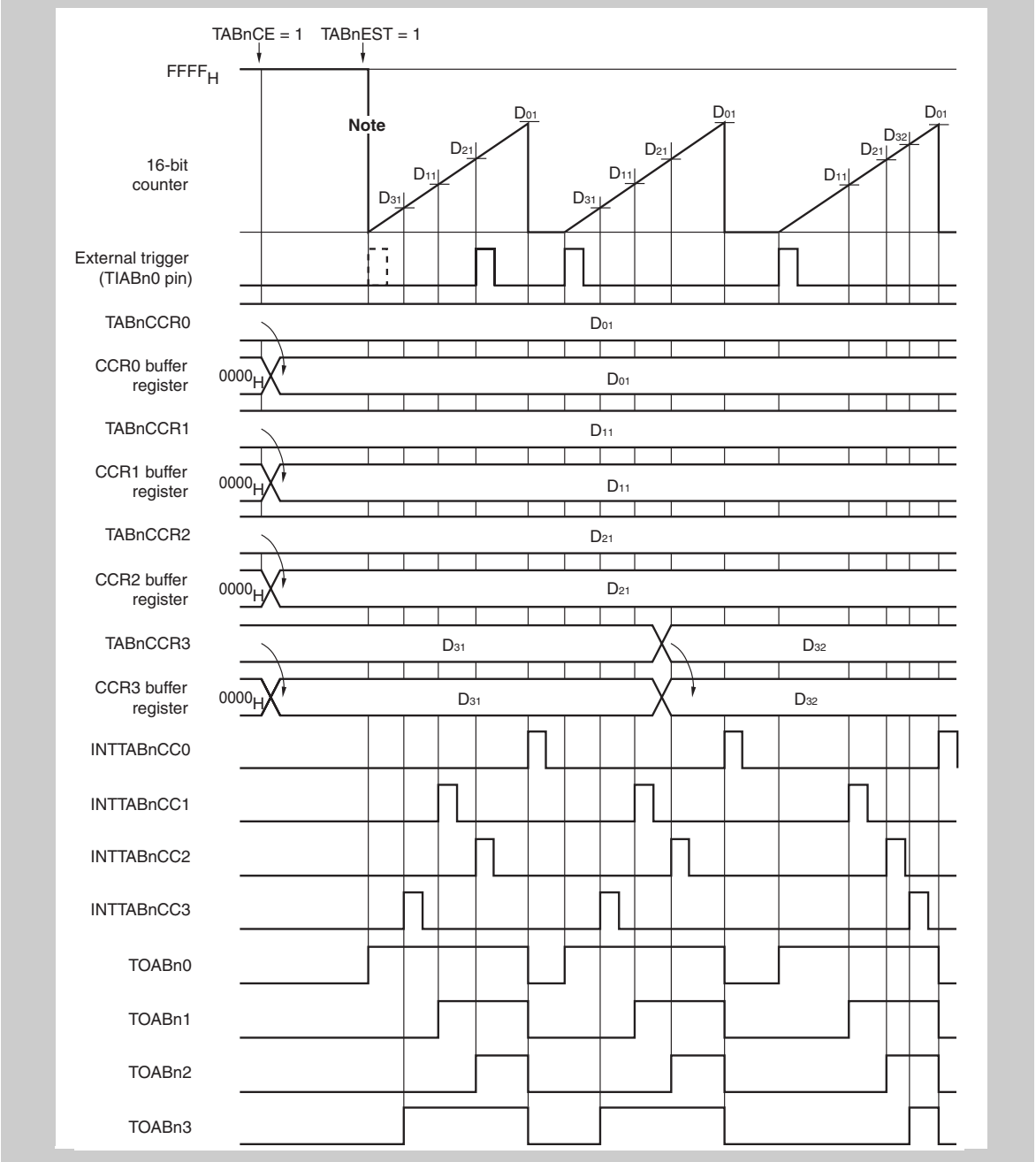


Caution The 16-bit counter is not cleared even if the trigger is input while the counter is counting up, and the trigger input is ignored.

Note 1. The 16-bit counter is not cleared upon a match between the 16-bit counter and the CCRk buffer register.

2. m = 0 to 3; k = 1 to 3

Figure 10-19 Timing of basic operation in one-shot pulse mode



Note

1. The 16-bit counter starts counting up when either TABnEST is set to 1 or external trigger (TIABn0 pin) is input.
2. Only the TABnEST bit of the TABnCTL register can be rewritten during timer operation (TABnCE = 1).
3. D01: Setting value of TABnCCR0 register (0000_H to $FFFF_H$)
   D11: Setting value of TABnCCR1 register (0000_H to $FFFF_H$)
   D21: Setting value of TABnCCR2 register (0000_H to $FFFF_H$)
   D31, D32: Setting value of TABnCCR3 register (0000_H to $FFFF_H$)
4. Output delay time = (Setting value of TABnCCRk register) × count clock cycle
5. Active level width = (Setting value of TABnCCR0 register - Setting value of TABnCCRk register + 1) × count clock cycle

10.6.6 PWM mode (TABnMD2 to TABnMD0 = 100)

In the PWM mode, TABn capture/compare register k (TABnCCRk) is used to set the duty factor and TABn capture/compare register 0 (TABnCCR0) is used to set the cycle.

By using these two registers and operating the timer, variable-duty PWM is output.

Rewriting the TABnCCRm register is enabled when TABnCE = 1.

So that the set value of the TABnCCRm register is compared with the value of the 16-bit counter (reloaded to the CCRm buffer register), a value must be written to the TABnCCR1 register before the value of the 16-bit counter matches the value of the TABnCCR0 register. The value of the TABnCCRm register is reloaded to the CCRm buffer registers when the value of the TABnCCR0 register later matches the value of the 16-bit counter.

Whether the next reload timing is made valid or not is controlled by writing to the TABnCCR1 register. Therefore, write the same value to the TABnCCR1 register even when only the value of the TABnCCR0 register needs to be rewritten. Reload is invalid when only the value of the TABnCCR0/TABnCCR2/TABnCCR3 register is rewritten.

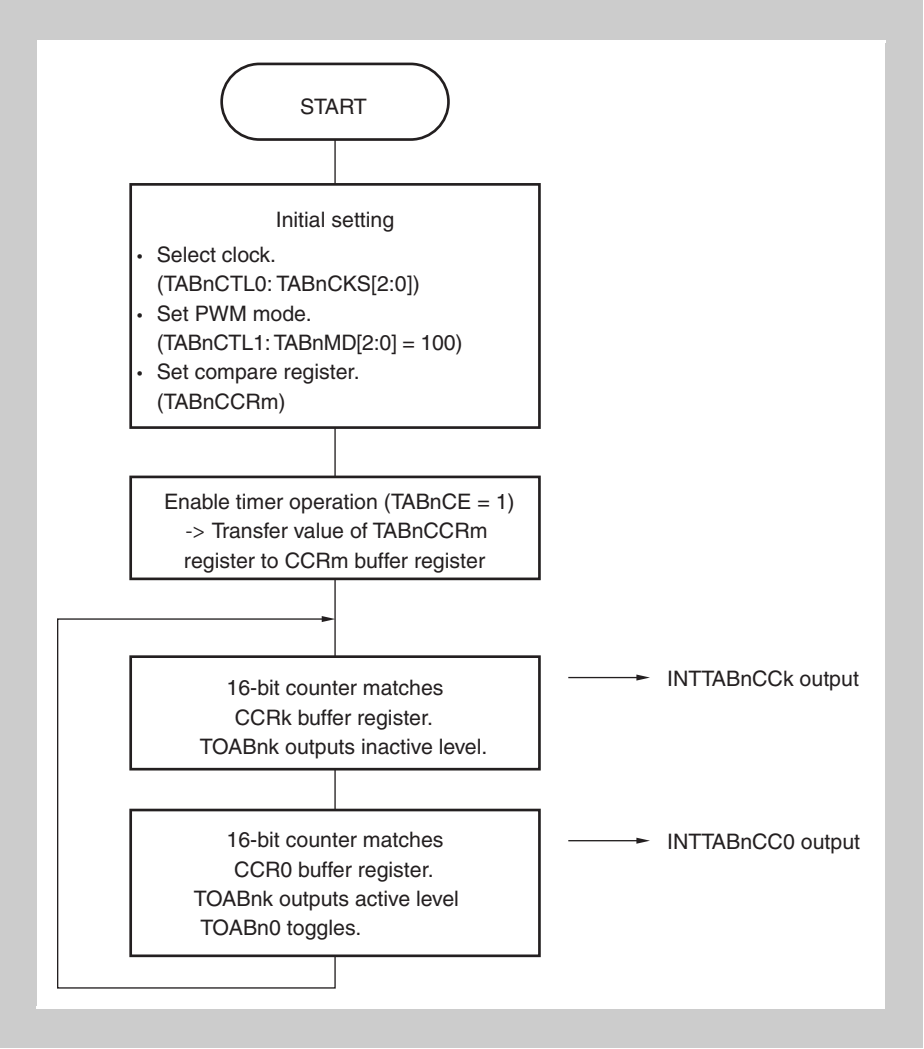
To stop Timer AB, set or clear TABnCE to 0.

The waveform of PWM is output from the TOABnk pin.
The TOABn0 pin produces a toggle output when the 16-bit counter matches the TABnCCR0 register.

In the PWM mode, the TABnCCRm register is used only as a compare register. It cannot be used as a capture register.

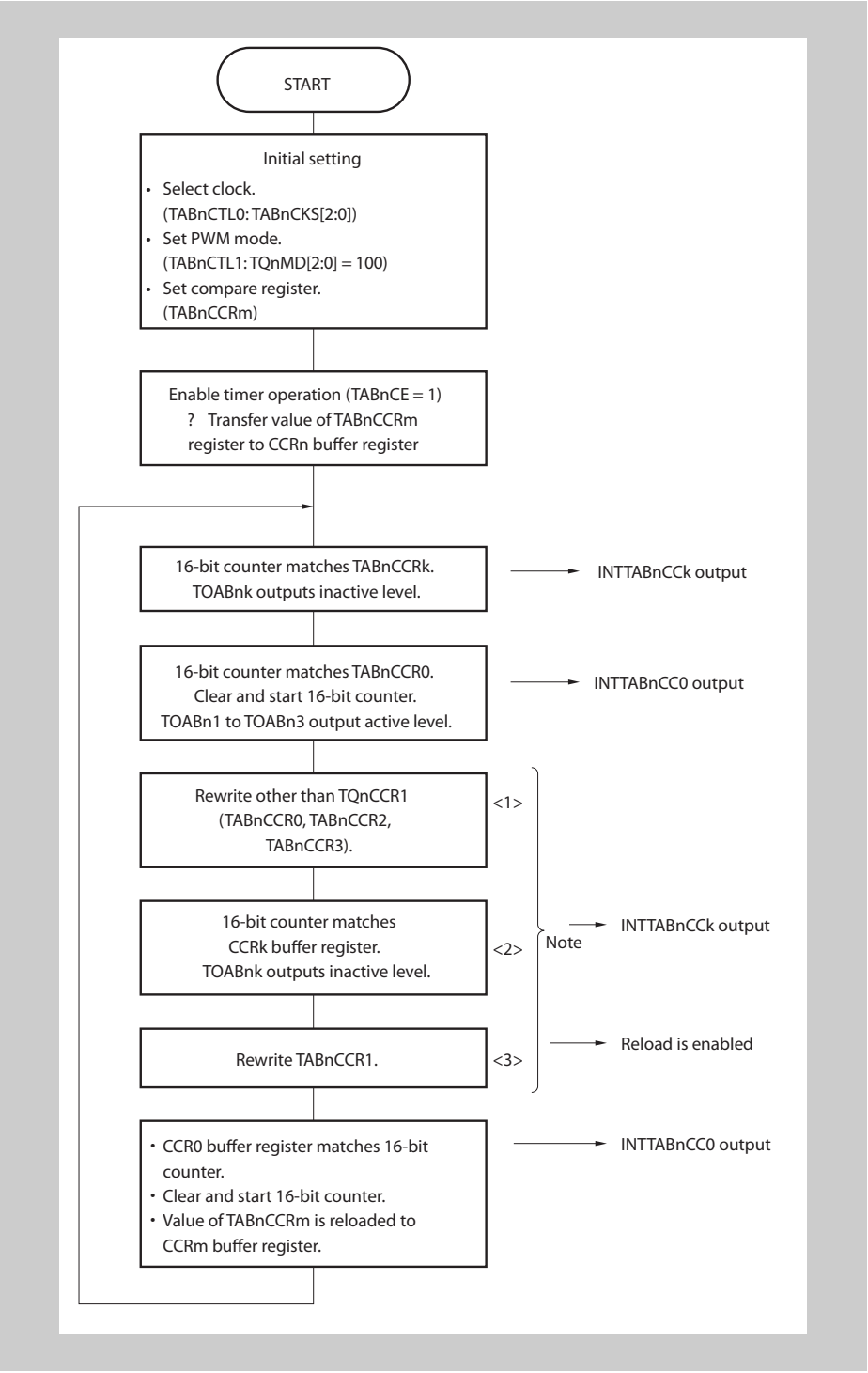
Note m = 0 to 3; k = 1 to 3

Figure 10-20 Flowchart of basic operation in PWM mode (1/2) (When values of TABnCCRm register is not rewritten during timer operation)



Note m = 0 to 3; k = 1 to 3

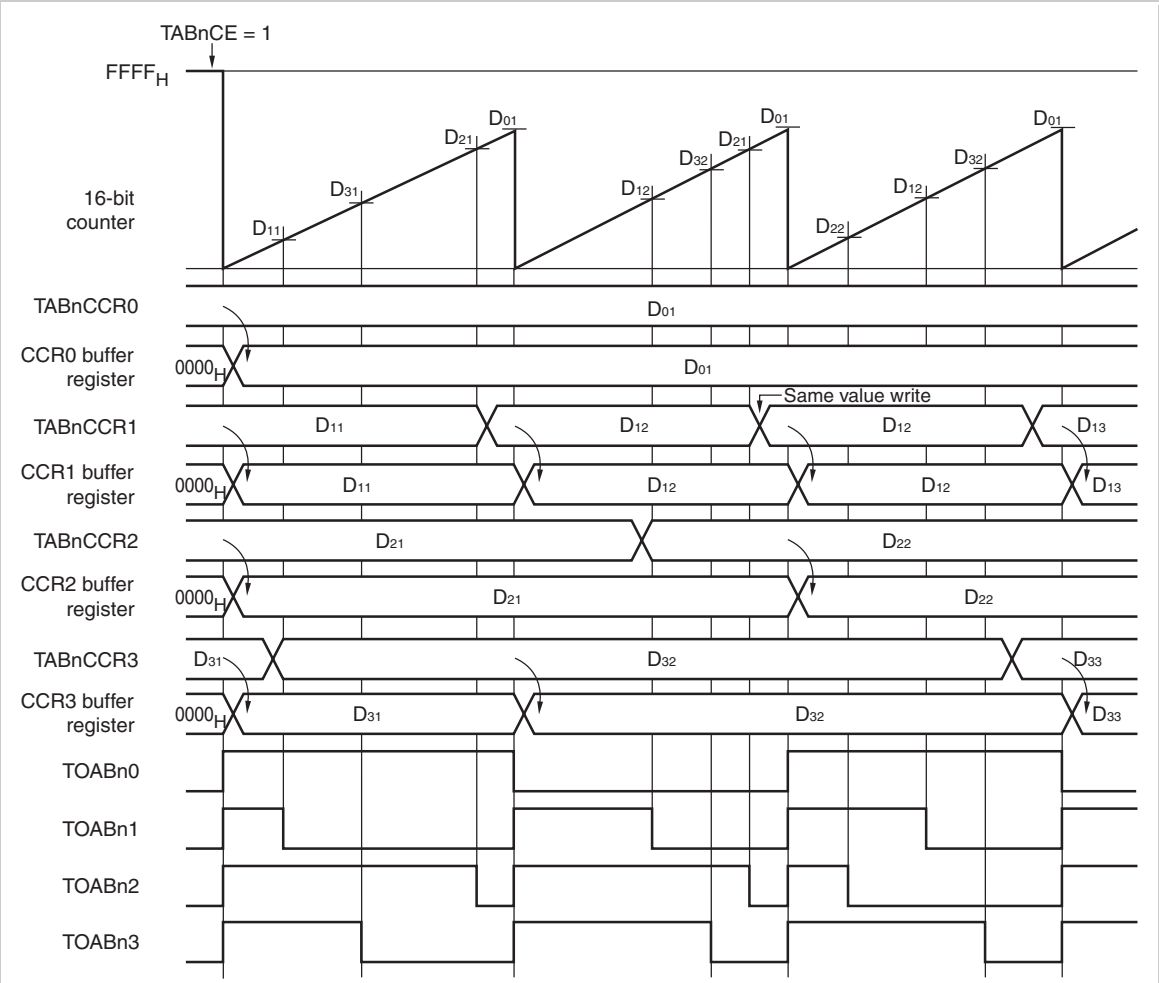
Figure 10-21 Flowchart of basic operation in PWM mode (2/2) (Value of TABnCCRm register rewritten during timer operation)



Note 1. The timing of <2> in the flowchart above may differ depending on the rewrite timing of steps <1> and <3> and the value of TABnCCRk, but make sure that step <3> comes after step <1>.

2. m = 0 to 3; k = 1 to 3

Figure 10-22 Basic operation timing in PWM mode (1/2) (when rewriting values of TABnCCR1 to TABnCCR3 registers)

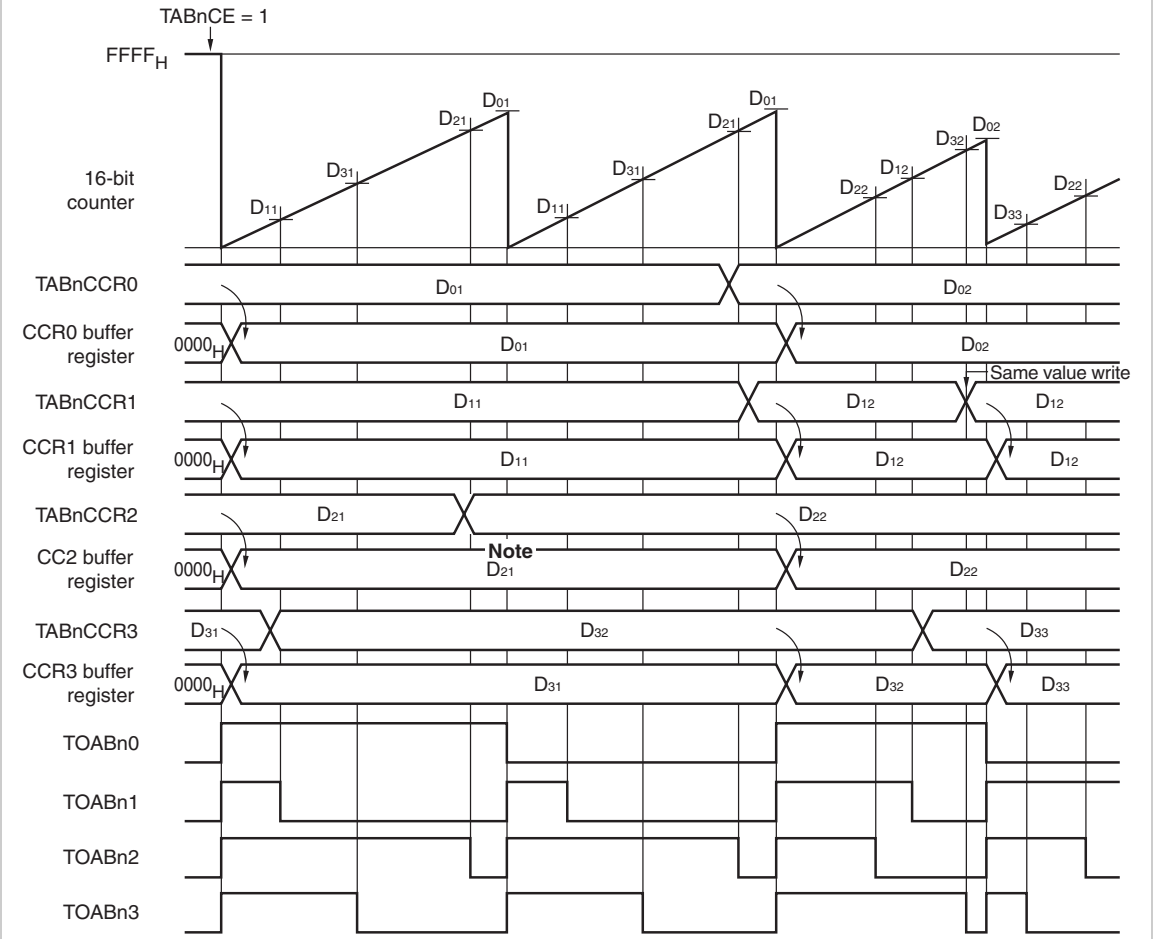


Note 1. D10: Setting value of TABnCCR0 register (0000_H to $FFFF_H$)
D11, D12, D13: Setting values of TABnCCR1 register (0000_H to $FFFF_H$)
D21, D22: Setting values of TABnCCR2 register (0000_H to $FFFF_H$)
D31, D32, D33: Setting values of TABnCCR3 register (0000_H to $FFFF_H$)

2. Duty of TOABnk output =
(Set value of TABnCCRk register) / (Set value of TABnCCR0 register + 1)
Cycle of TOABnk output =
(Set value of TABnCCR0 register + 1) × (Count clock cycle)
Toggle width of TOABn0 output =
(Set value of TABnCCR0 register + 1) x (Count clock cycle)

3. k = 1 to 3

Figure 10-23 Basic operation timing in PWM mode (2/2) (when rewriting values of TABnCCR0 to TABnCCR3 registers)



Note 1. Reload is not performed because the TABnCCR1 register was not rewritten.

2. D01, D02: Setting values of TABnCCR0 register (0000_H to $FFFF_H$)
D11, D12: Setting values of TABnCCR1 register (0000_H to $FFFF_H$)
D21, D22: Setting values of TABnCCR2 register (0000_H to $FFFF_H$)
D31, D32, D33: Setting values of TABnCCR3 register (0000_H to $FFFF_H$)

3. Duty of TOABnk output =
(Set value of TABnCCRk register) / (Set value of TABnCCR0 register + 1)
Cycle of TOABnk output =
(Set value of TABnCCR0 register + 1) × (Count clock cycle)
Toggle width of TOABn0 output =
(Set value of TABnCCR0 register + 1) × (Count clock cycle)

4. k = 1 to 3

5. To output a 0% duty PWM signal set the TABnCCRm register to 0.
To output a 100% duty PWM signal set the TABnCCRm register to the value of the TABnCCR0 register +1. Do not set a value of $FFFF_H$ to the TABnCCR1 register.

10.6.7 Free-running mode (TABnMD2 to TABnMD0 = 101)

In the free-running mode, the 16-bit counter is operating as a free-running counter and the capture/compare operation is selected with the TABnCCS3 to TABnCCS0 bits of the TABnOPT0 register.

The settings of the TABnCCS3 to TABnCCS0 bits of the TABnOPT0 register are valid only in the free-running mode.

TABnCCSm	Operation
0	Use TABnCCRm register as compare register
1	Use TABnCCRm register as capture register

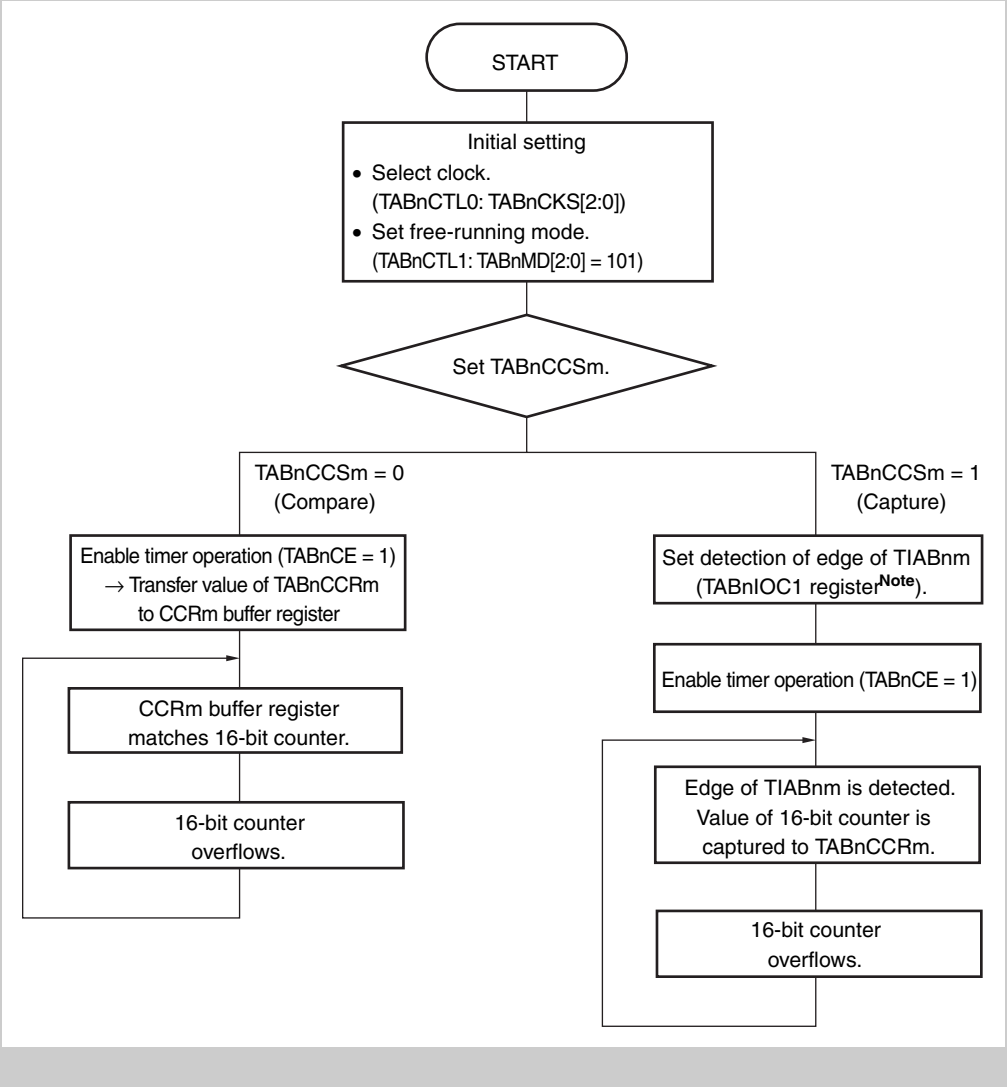
- When TABnCCRm register is used as compare register
  - When the value of the 16-bit counter matches the value of the CCRm buffer register in the free-running mode, an interrupt is generated (interval function).
  - Rewriting the value of the compare register is enabled during timer operation, and a value can be written to the register at any time (when the value to be compared is written to the register, it is synchronized with the internal clock and compared with the value of the 16-bit counter).
  - If timer output (TOABnm) is enabled, TOABnm produces a toggle output when the value of the 16-bit counter matches the value of the CCRm buffer register.

- When TABnCCRm register is used as capture register

  The value of the 16-bit counter is saved to the TABnCCRm register upon TIABnm pin edge detection.

Note
1. The TABnCCR0 register cannot be used as the capture register when the TABnEEE bit of the TABnCTL1 register is set to 1 and the external event count input is selected for the count clock.
2. For rewriting of the TABnCCR0 to TABnCCR3 registers during timer operation (TABnCE = 1), refer to *Section 10.6.1, Anytime write and reload* on page 236
3. n = 0 to 2, m = 0 to 3

Figure 10-24 Flowchart of basic operation in free-running mode



Note 1. TABCCR0 edge detection: TABnIS1 and TABnIS0 bits
TABCCR1 edge detection: TABnIS3 and TABnIS2 bits
TABCCR2 edge detection: TABnIS5 and TABnIS4 bits
TABCCR3 edge detection: TABnIS7 and TABnIS6 bits

2. n = 0 to 2, m = 0 to 3

(1) When TABnCCSn = 0 setting (compare function)

When TABnCE is set to 1, the 16-bit counter counts from 0000_H to $FFFF_H$, and continues counting up in the free-running mode until TABnCE is cleared to 0.

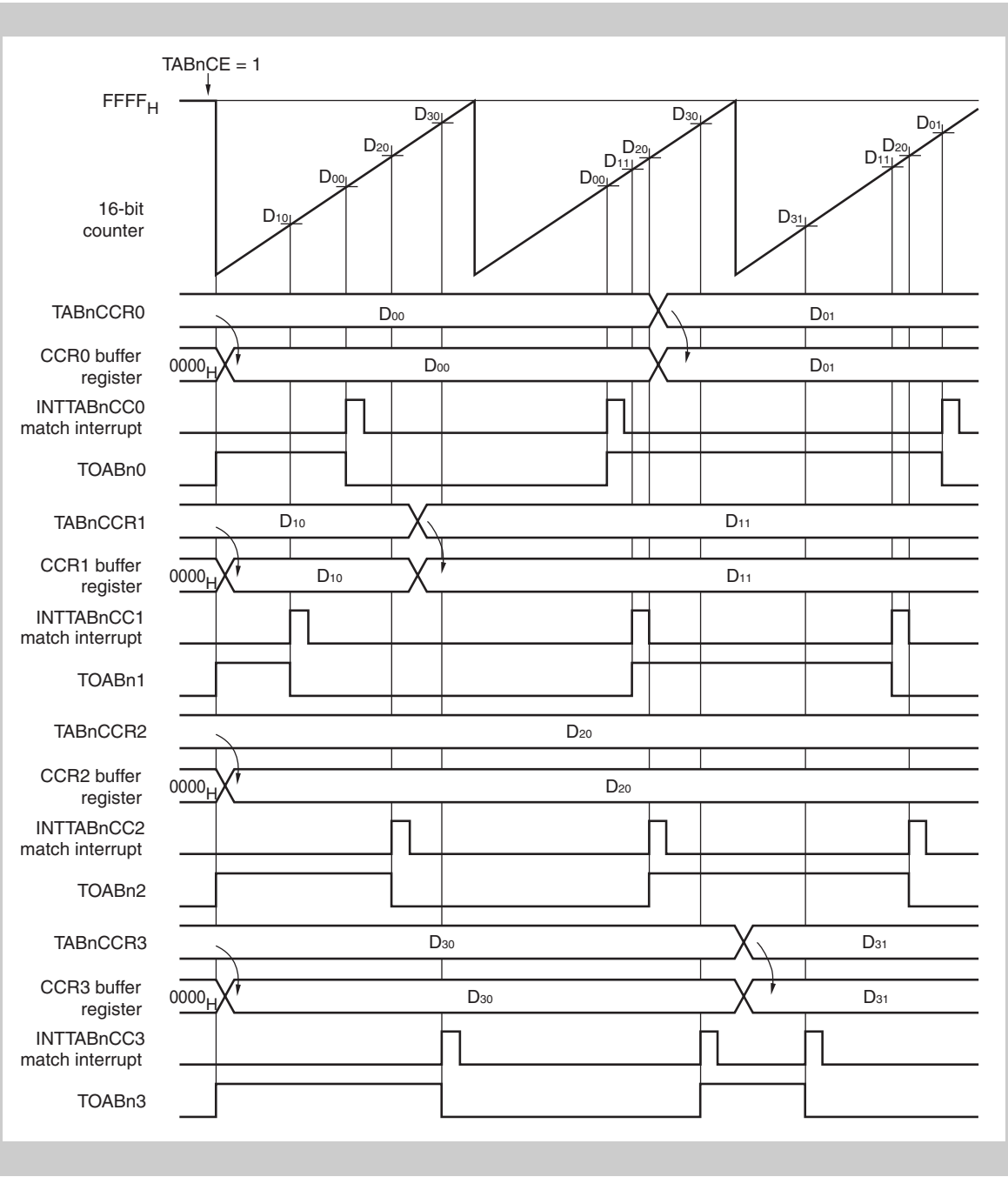
If a value is written to the TABnCCRm register in this mode, it is transferred to the CCRm buffer registers (anytime write). Even if an one-shot pulse trigger is input in this mode, an one-shot pulse is not generated. If TABnOEm is set to 1, TOABnm produces a toggle output when the value of the 16-bit counter matches the value of the CCRm buffer register.

(2) When TABnCCSn = 1 setting (capture function)

When TABnCE is set to 1, the 16-bit counter counts from 0000_H to $FFFF_H$, and continues counting up in the free-running mode until TABnCE is cleared to 0. The value captured by a capture trigger is written to the TABnCCRm registers.

Capturing before and after overflow ($FFFF_H$) is judged using the overflow flag (TABnOVF). However, if the interval of the capture trigger is such that the overflow occurs two times (two periods of more of free-running), the TABnOVF flag cannot be used for judgment.

Figure 10-25 Basic operation timing in free-running mode (1/4) (TABnCCS3 = 0, TABnCCS2 = 0, TABnCCS1 = 0, TABnCCS0 = 0)

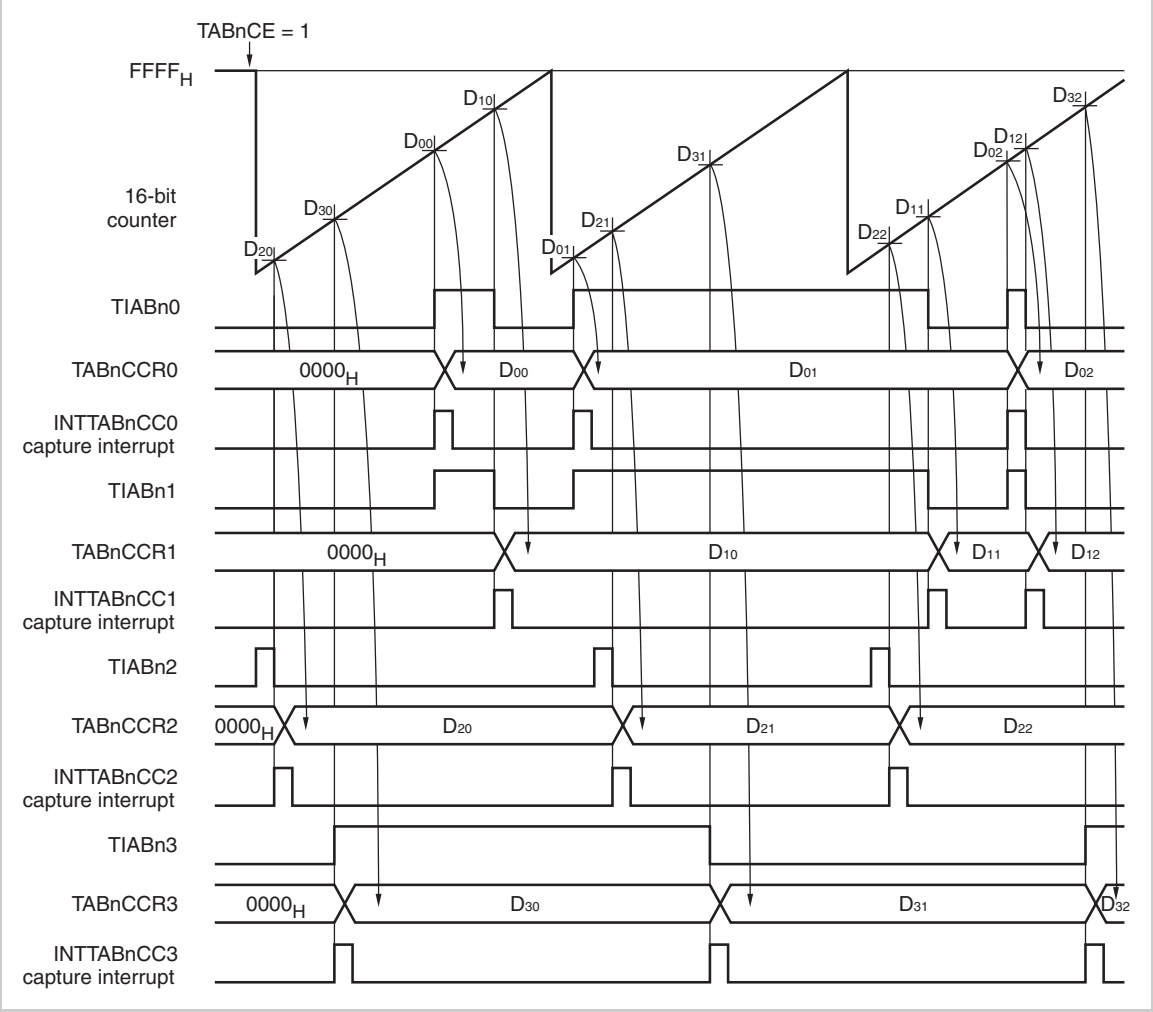


Note 1. D00, D01: Setting values of TABnCCR0 register (0000_H to $FFFF_H$)
D10, D11: Setting values of TABnCCR1 register (0000_H to $FFFF_H$)
D20: Setting value of TABnCCR2 register (0000_H to $FFFF_H$)
D30, D31: Setting values of TABnCCR3 register (0000_H to $FFFF_H$)

2. TOABnm output goes high when counting is started.

3. m = 0 to 3

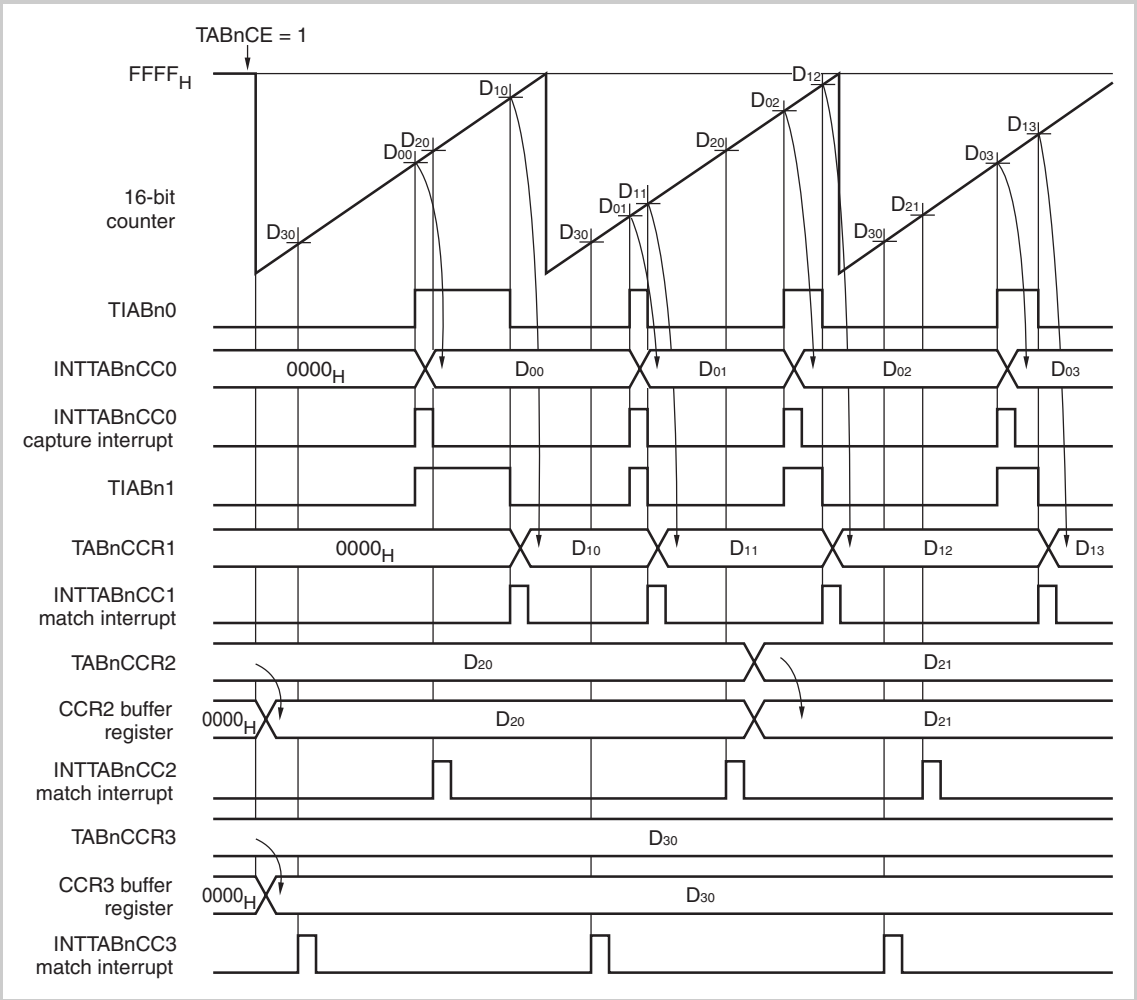
Figure 10-26 Basic operation timing in free-running mode (2/4) (TABnCCS3 = 1, TABnCCS2 = 1, TABnCCS1 = 1, TABnCCS0 = 1)



Note 1. D00, D01, D02: Values captured to TABnCCR0 register (0000_H to $FFFF_H$)
D10, D11, D12: Values captured to TABnCCR1 register (0000_H to $FFFF_H$)
D20, D21, D22: Values captured to TABnCCR2 register (0000_H to $FFFF_H$)
D30, D31, D32: Values captured to TABnCCR3 register (0000_H to $FFFF_H$)

2. TIABn0: Set to rising edge detection (TABnIS1, TABnIS0 = 01)
   TIABn01: Set to falling edge detection (TABnIS3, TABnIS2 = 10)
   TIABn2: Set to falling edge detection (TABnIS5, TABnIS4 = 10)
   TIABn3: Set to detection of both rising and falling edges
   (TABnIS7, TABnIS6 = 11)

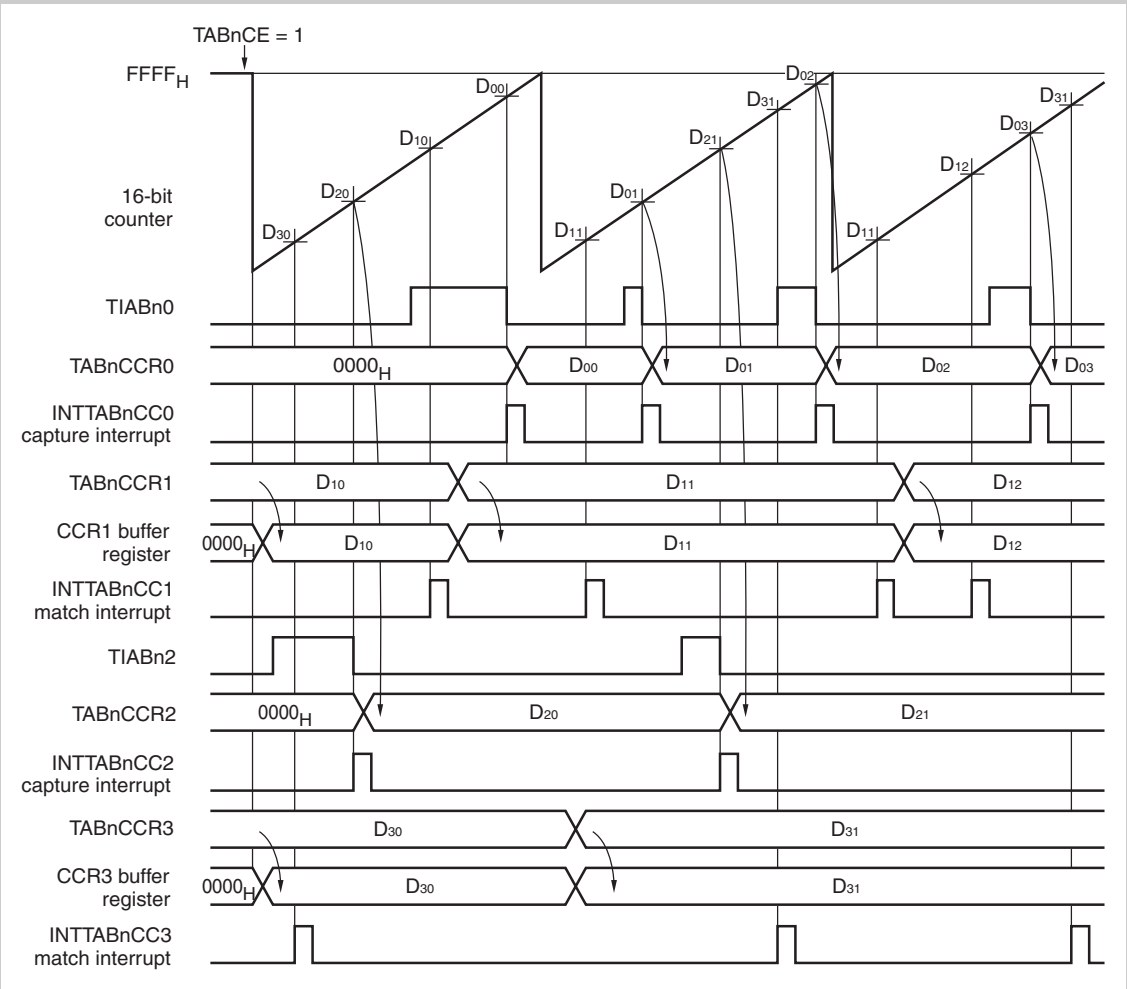
Figure 10-27 Basic operation timing in free-running mode (3/4)
(TABnCCS3 = 1, TABnCCS2 = 1, TABnCCS1 = 1, TABnCCS0 = 0)



Note 1. D00, D01, D02, D03: Values captured to TABnCCR0 register (0000_H to $FFFF_H$)
D10, D11, D12, D13: Values captured to TABnCCR1 register (0000_H to $FFFF_H$)
D20, D21: Setting values of TABnCCR2 register (0000_H to $FFFF_H$)
D30: Setting value of TABnCCR3 register (0000_H to $FFFF_H$)

2. TIABn0: Set to rising edge detection (TABnIS1, TABnIS0 = 01)
   TIABn1: Set to falling edge detection (TABnIS3, TABnIS2 = 10)

Figure 10-28 Basic operation timing in free-running mode /4/4) (TABnCCS3 = 0, TABnCCS2 = 1, TABnCCS1 = 0, TABnCCS0 = 1)



Note 1. D00, D01, D02, D03: Values captured to TABnCCR0 register (0000_H to $FFFF_H$)
D10, D11, D12: Setting values of TABnCCR1 register (0000_H to $FFFF_H$)
D20, D21: Values captured to TABnCCR2 register (0000_H to $FFFF_H$)
D30, D31: Setting values of TABnCCR3 register (0000_H to $FFFF_H$)

2. TIABn0: Set to falling edge detection (TABnIS1, TABnIS0 = 10)
TIABn2: Set to falling edge detection (TABnIS5, TABnIS4 = 10)

(3) Overflow flag

When the counter overflows from $FFFF_H$ to 0000_H in the free-running mode, the overflow flag (TABnOVF) is set to 1 and an overflow interrupt (INTTABnOV) is output.

The overflow flag is cleared by the CPU writing 0 to it.

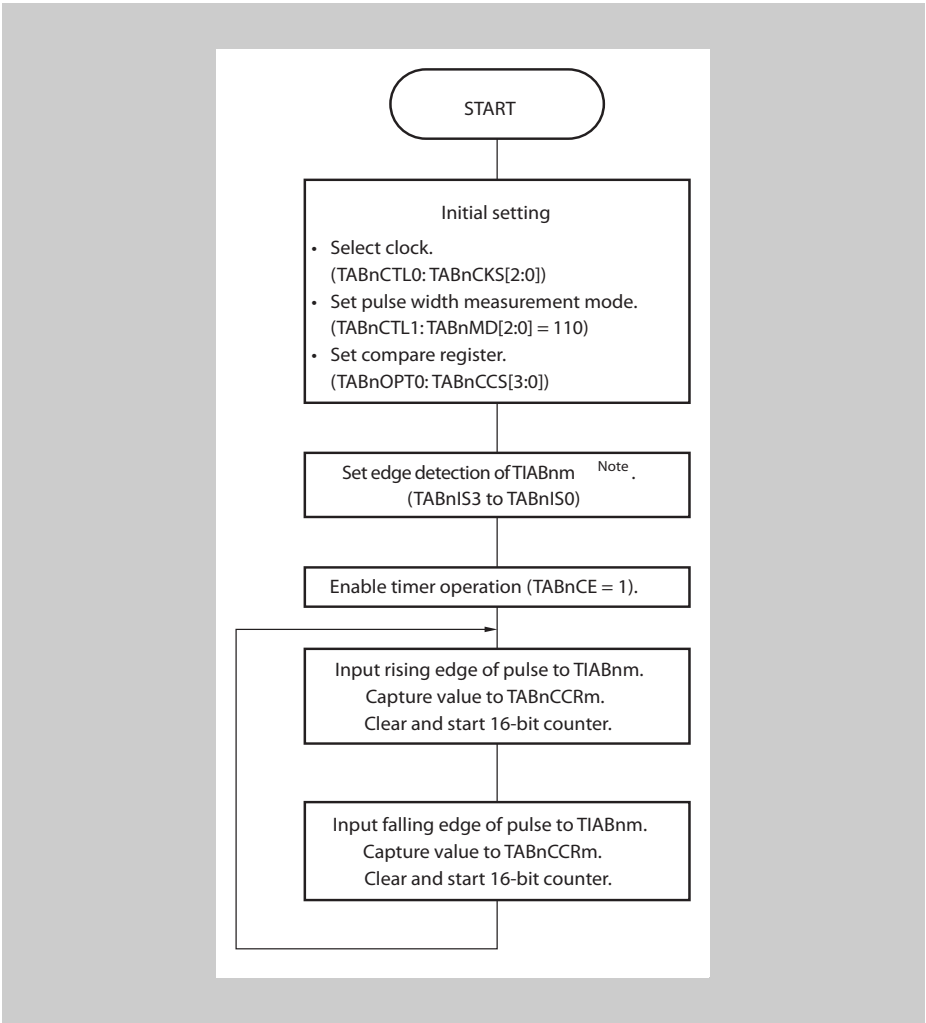
10.6.8 Pulse width measurement mode (TABnMD2 to TABnMD0 = 110)

In the pulse width measurement mode, free-running counting is performed. The value of the 16-bit counter is captured to capture register m (TABnCCRm) when both the rising and falling edges of the TIABnm pin are detected, and the 16-bit counter is cleared to 0000_H. In this way, the external input pulse width can be measured.

To measure a long pulse width that exceeds the overflow of the 16-bit counter, use the overflow flag for detection. A pulse width that causes overflow to occur twice or more cannot be measured. Adjust the operating frequency of the 16-bit counter.

Caution In the pulse width measurement mode, select the internal clock (TABnEEE of TABnCTL1 register = 0) as a count clock.

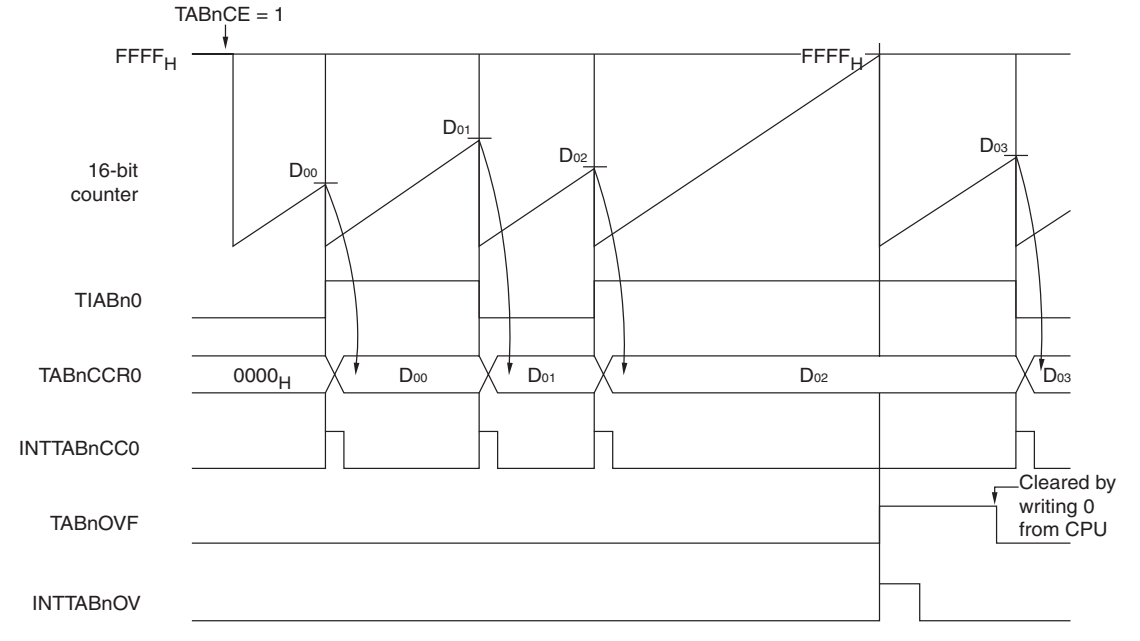
Figure 10-29 Flowchart of basic operation in pulse width measurement mode



Note m = 0 to 3

Caution An external pulse can be input from any of TIABn0 to TIABn3 but only one of them can be used. Specify that both the rising and falling edges are detected. Specify that the input edge of an external pulse input that is not used is not detected.

Figure 10-30 Basic operation timing in pulse width measurement mode



Note

1. D00, D01, D02, D03: Values captured to TABnCCR0 register (0000_H to $FFFF_H$)
2. TIABn0: Set to detection of both rising and falling edges
3. Pulse width = Captured value × Count clock cycle
   - If the valid edge is not input even when the 16-bit counter counted up to $FFFF_H$, an overflow interrupt request signal (INTTABnOV) is generated at the next count clock, and the counter is cleared to 0000_H and continues counting. At this time, the overflow flag (TTABnOVF bit) is also set to 1. After the overflow flag is read and confirmed, clear it 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: Pulse width = (10000_H × TABnOVF bit set (1) count + Captured value) × Count clock cycle

Chapter 11 Timer AA/AB Synchronous Operation

Timers AA and Timers AB have a timer synchronized operation function, also named tuned operation mode. Master timer and incorporated slave timers of the corresponding timer group (listed in *Table 11-1*) start and clock synchronously. When the master timer is cleared, the slave timers are cleared synchronously, too.

Table 11-1 Synchronous operation mode of timers

Master timer	Slave timer
TAA2	TAA3
TAB0	TAA0
TAB1	TAA1

Setup In the following the procedure is described how to set up the master and slave timer for synchronous operation. In this example, TABm is used as the master timer, TAAn is used as the slave timer.

- Slave timer setup
  - TAAnCTL1.TAAnSYE = 1: enable synchronous operation
  - TAAnCTL1.TAAnMD[2:0] = 101_B: free-running mode
  - TAAnCCR0/1: set compare value
- Master timer setup:
  - TABmCTL1.TABmMD[2:0]
    = 101_B: free-running mode
    = 100_B: PWM mode
    = 111_B: triangular wave PWM mode
  - TABmCCR0/1: set compare value
  - TABmCTL0.TABmC = 1: enable operation

Table 11-2 and *Table 11-3* show the timer modes that can be used in the synchronous operation mode.

Table 11-2 Timer modes usable in synchronous operation mode

Master timer	Slave timer	Free-running mode	PWM mode
TAA2	TAA3	√	√
TAB0	TAA0	√	√
TAB1	TAA1	√	√

Table 11-3 Timer output functions

Synch channel	Timer	Pin	Free-running mode		PWM mode	
			Synch OFF	Synch ON	Synch OFF	Synch ON
Ch1	TAA2 (master)	TOAA20	PPG	<-	Toggle	<-
		TOAA21	PPG	<-	PWM	<-
	TAA3 (slave)	TOAA30	PPG	<-	Toggle	PWM
		TOAA31	PPG	<-	PWM	<-
Ch2	TAB0 (master)	TOAB00	PPG	<-	Toggle	<-
		TOAB01 to TOAB03	PPG	<-	PWM	<-
	TAA0 (slave)	TOAA00	PPG	<-	Toggle	PWM
		TOAA01	PPG	<-	PWM	<-
Ch3	TAB1 (master)	TOAB10	PPG	<-	Toggle	<-
		TOAB11 to TOAB13	PPG	<-	PWM	<-
	TAA1 (slave)	TOAA10	PPG	<-	Toggle	PWM
		TOAA11	PPG	<-	PWM	<-

Note The timing of transmitting data from the compare register of the master timer to the compare register of the slave timer is as follows:

PPG:	CPU write timing (anytime write mode)
Toggle, PWM	Timing at which timer counter and compare register match TOAAn0 and TOABm0

Chapter 12 16-Bit Interval Timer M (TMM)

Instances The V850E/RG3 has two instances of this 16-bit timer/event counter M:

Table 12-1 Instances of Timer M

Timer M	
Instances	2
Names	TMM0, TMM1

Throughout this chapter, the individual instances of Timer M are identified by “n” (n = 0, 1).

12.1 Features

Timer M (TMM) supports only a clear & start mode. It does not support a free-running mode. To use Timer M in a manner equivalent to in the free-running mode, set the compare register to $FFFF_H$ and start the 16-bit counter. A match interrupt will occur when the timer overflows.

- Interval function
- Clock selection × 3
- Simple counter × 1

  (The simple counter is a counter that does not use a counter read buffer. This counter cannot be read during timer count operation.)
- Simple compare × 1

  (The simple compare register is a register that does not use a compare write buffer. No data can be written to this compare register during timer count operation.)
- Compare match interrupt × 1

Clock supply Each Timer M has three selectable clock inputs. All are connected to the clock generator.

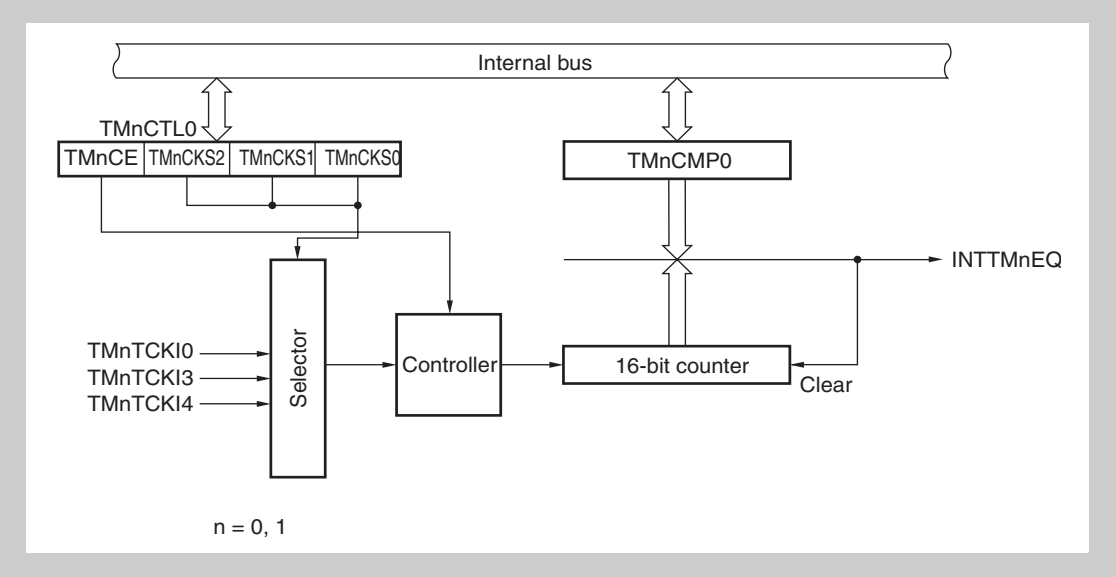
The table below lists the cross reference between TMMn clock input and the connected clocks:

Table 12-2 Clock inputs of TMMn

TMMn clock input	Connected clock
TMnTCKI0	PCLK1 (16 MHz)
TMnTCKI3	PCLK4 (2 MHz)
TMnTCKI4	PCLK5 (1 MHz)

12.2 Timer M Block diagram

Figure 12-1 Block diagram of Timer M



12.3 Timer M Registers

Register addresses

All TMMn register addresses are given as address offsets to the individual base addresses <base> of each TMMn.

The <base> addresses of each TMMn are listed in the following table:

Table 12-3 Register <base> addresses of TMMn

TMMn	<base> address
TMM0	FFFF F6C0$_H$
TMM1	FFFF F6D0$_H$

The TMMn are controlled and operated by means of the following registers.

Table 12-4 TMMn registers overview

Register name	Shortcut	Address
TMMn control register 0	TMnCTL0	<base>
TMMn compare register 0	TMnCMP0	<base> + 4$_H$

12.3.1 TMnCTL0 - TMMn control register 0

The TMnCTL0 register is an 8-bit register that controls the operation of TMM.

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

Address <base>

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

7	6	5	4	3	2	1	0
TMnCE	0	0	0	TMnCKS3	TMnCKS2	TMnCKS1	TMnCKS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Caution Changing the TMnCTL0.TMnCKS[2:0] bits is prohibited while the timer is operating (TMnCE = 1). Thus rewriting of these bits with the same value is permitted.

The TMnCTL.TMnCE bit can be changed at any time.

TMnCE	Control of operation of Timer M
0	Disable internal operating clock operation (asynchronously reset TMMn).
1	Enable internal operating clock operation.
The TMnCE bit controls the internal operating clock and asynchronously resets TMMn. When this bit is cleared to 0, the internal operating clock of TMM is stopped, and TMMn is asynchronously reset. When the TMnCE bit is set to 1, the internal operating clock is enabled within two input clocks, and the timer counts up.	

TMnCKS3	TMnCKS2	TMnCKS1	TMnCKS0	Internal count clock selection
0	0	0	0	TMnTCKI0
0	0	0	1	TMnTCKI0/2
0	0	1	0	TMnTCKI0/4
0	0	1	1	TMnTCKI3
0	1	0	0	TMnTCKI4
other than above				prohibited

Cautions
1. Set TMnCKS3 to TMnCKS0 bits at TMnCE = 0. When the TMnCE bits is set from 0 to 1, the TMnCKS3 to TMnCKS0 bits can be set at the same time.
2. Set bit 6-4 to 0.

12.3.2 TMnCMP0 - TMMn compare register 0

The TMnCMP0 register is a 16-bit compare register.

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.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	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/W

Caution Changing the TMnCMP0 register contents is prohibited while the timer is operating (TMnCE = 1). Thus rewriting with the same value is permitted.

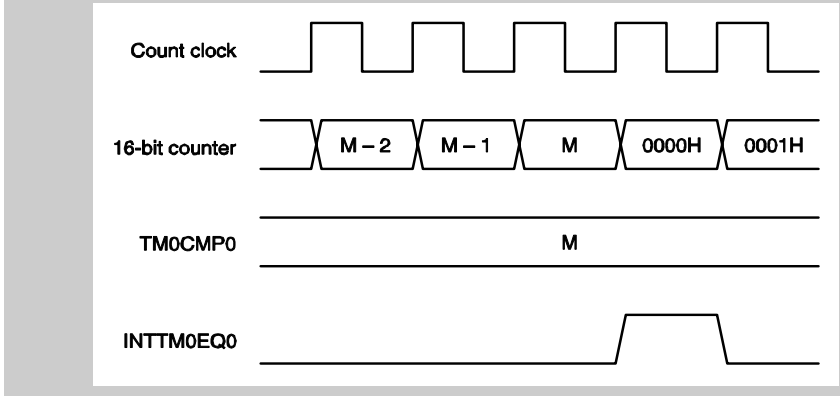
12.4 Operation

12.4.1 Interval timer mode

In the interval timer mode, a match interrupt signal (INTTMnEQ0) is output when the value of the 16-bit counter matches the value of TMMn compare register 0 (TMnCMP0). At the same time, the counter is cleared to 0000_H and starts counting up.

When the TMnCMP0 register is set to $FFFF_H$, Timer M performs an operation similar to that of a timer operating in free-running mode.

Figure 12-2 Timing of operation in interval timer mode



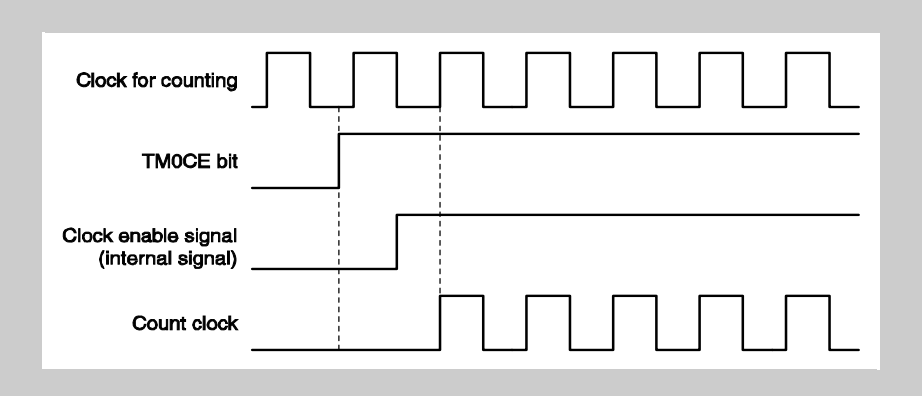
Caution To set M clocks as the interval period, set the TMnCMP0 register to M – 1.

12.4.2 Cautions

(1) Clock Generator and clock enable timing

Because the second clock is the first pulse of the timer count-up signal when the TMnCE bit is changed from 0 to 1, the timer counts one clock less.

Figure 12-3 Count operation start timing



(2) Changing register settings while TMM is operating

The values of TMnCMP0 and TMnCTL0 cannot be changed while the timer is operating. The user must first clear the TMnCTL0.TM0CE bit and stop the timer before changing the register settings. After the registers are updated, the timer can be restarted.

Chapter 13 Watchdog Timer (WDT)

13.1 Functions

Watchdog Timer has the following functions.

- Default-start Watchdog Timer
- Reset mode:
  Reset operation upon overflow of Watchdog Timer 2 (generation of WDTRES signal)
- Non-maskable interrupt request mode:
  NMI operation upon overflow of Watchdog Timer 2 (generation of INTWDT signal)
- Input selectable from main clock or internal ring oscillator as the source clock

Cautions

1. Watchdog Timer is automatically started after reset release. Source clock is the internal ring oscillator.
2. Using the flash mask option register, WDT can be fixed to operate from an internal low frequency ring oscillator (212 KHz typ.) and only in Reset mode. Only the interval time can be changed. Changing the clock source and operation mode is not possible.
3. Sometimes, WDT will not be used and the clock source and operation mode will be changed. In such cases, the flash mask option should not be set to fix the internal ring oscillator source clock and reset mode.
   In this case, after reset, the settings should be changed before the first WDT overflow. Alternatively WDT should be cleared once, and required changes should be performed within the next interval time.
4. The WDTM register can be written only once after reset. Even if the default setting of WDTM shall not be changed, it is recommended to write the default value to WDTM once in order to activate the write protection mechanism.
5. The RETI instruction can not be used to restore from the interrupt service routine of the non-maskable INTWDT. Therefore a system reset must be performed after completion of the INTWDT service routine.

For further information about flash mask option registers, see 4.2.3“Flash Mask Option Register (FMOP0)” on page 73.

13.2 Watchdog Timer 2 Block Diagram

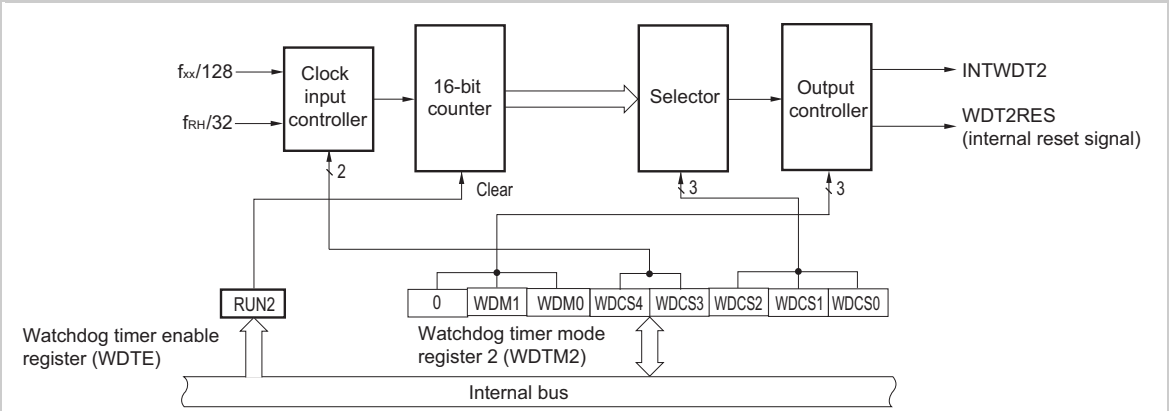


Figure 13-1 Block diagram of Watchdog Timer 2

Note f_{XX}: Oscillation frequency
f_{RH}: Internal-OSC clock frequency
INTWDT: Non-maskable interrupt request signal from Watchdog Timer 2
WDTRES: Watchdog Timer 2 reset signal

13.3 Control Registers

13.3.1 WDTM - Watchdog Timer mode register 2

The WDTM register sets the operation mode, operation clock and overflow time of Watchdog Timer 2.

Access The register can be read/written in 1 or 8-bit units.
This register can be read any number of times, but it can be written only once following reset release.

Address FFFF FDB8$_H$

Initial Value 67$_H$. The register is initialized by any reset.

7	6	5	4	3	2	1	0
0	WDM1	WDM0	WDCS4	WDCS3	WDCS2	WDCS1	WDCS0
R	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 13-1 Selection of operation mode

WDM1	WDM0	Function
0	0	Stops operation
0	1	Non-maskable interrupt request mode (generation of INTWDT)
1	x	Reset mode (generation of RESWDT)

Table 13-2 Watchdog Timer 2 Clock Selection

WDCS4	WDCS3	WDCS2	WDCS1	WDCS0	Selected clock period	6.8 MHz (typ.)	
0	0	0	0	0	$2^{14}/f_{RH}$	2.42 ms	
0	0	0	0	1	$2^{15}/f_{RH}$	4.83 ms	
0	0	0	1	0	$2^{16}/f_{RH}$	9.67 ms	
0	0	0	1	1	$2^{17}/f_{RH}$	19.3 ms	
0	0	1	0	0	$2^{18}/f_{RH}$	38.67 ms	
0	0	1	0	1	$2^{19}/f_{RH}$	77.33 ms	
0	0	1	1	0	$2^{20}/f_{RH}$	154.66 ms	
0	0	1	1	1	$2^{21}/f_{RH}$	309.3 ms	
						f_X = 8 MHz	f_X = 16 MHz
0	1	0	0	0	$2^{16}/f_X$	8.2 ms	4.1 ms
0	1	0	0	1	$2^{17}/f_X$	16.4 ms	8.2 ms
0	1	0	1	0	$2^{18}f_X$	32.8 ms	16.4 ms
0	1	0	1	1	$2^{19}/f_X$	65.5 ms	32.8 ms
0	1	1	0	0	$2^{20}/f_X$	131.1 ms	65.3 ms
0	1	1	0	1	$2^{21}/f_X$	262.1 ms	131.1 ms
0	1	1	1	0	$2^{22}/f_X$	524.3 ms	262.2 ms
0	1	1	1	1	$2^{23}/f_X$	1048.6 ms	524.3 ms
1	×	×	×	×	STOP		

Cautions

1. If the WDTM register is rewritten twice after reset, an overflow signal is forcibly generated. If the Watchdog Timer has stopped operation, WDTM can be written several times without generating an overflow.
2. To stop WDT securely, set WDTM = $1F_H$.
3. In order to ensure that the Watchdog Timer does not overflow and thus generate a watchdog event, write to the WDTE register first to reset the watchdog timer, then write the desired value to the WDTM register.

13.3.2 WDTE - Watchdog Timer enable register

The counter of Watchdog Timer 2 is cleared and counting restarted by writing AC_H to the WDTE register.

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

Address FFFF FDB9$_H$

Initial Value $9A_H$. The register is initialized by any reset.

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

Cautions

1. When a value other than AC_H is written to the WDTE register, an overflow signal is forcibly output.
2. When a 1-bit memory manipulation instruction is executed for the WDTE register, an overflow signal is forcibly output.
3. The read value of the WDTE register is $9A_H$ (which differs from written value AC_H).

13.4 Watchdog Timer Operation

Watchdog Timer 2 automatically starts after reset is released.

The WDTM register can be written only once following reset using byte access. To use watchdog timer 2, write the operation mode and the interval time to the WDTM register using an 8-bit memory manipulation instruction. After this, the operation of watchdog timer 2 cannot be stopped/changed again.

The WDCS4 to WDCS0 bits of the WDTM register are used to select the watchdog timer loop detection time interval.

Writing AC_H to the WDTE register clears the counter of watchdog timer and starts the count operation again.

After the count operation has started, write AC_H to WDTE within the loop detection time interval.

If the time interval expires without AC_H being written to the WDTE register, a reset signal (WDTRES) or a non-maskable interrupt request signal (INTWDT) is generated, depending on the set values of the WDM1 and WDM2 bits of the WDTM register.

When not using watchdog timer 2, write $1F_H$ to the WDTM register.

If the non-maskable interrupt request mode is set, execution cannot return from non-maskable interrupt servicing by using the RETI instruction. Therefore, execute a system reset after interrupt servicing ends.

Chapter 14 Clocked Serial Interface (CSIF)

Instances The V850E/RG3 microcontrollers has 2 instances of the clocked serial interface CSIF.

Table 14-1 Instances of CSIF

CSIF	
Instances	2
Names	CSIF0, CSIF1

Throughout this chapter, the individual instances of clocked serial interface are identified by “n”, for example CSIFn, or CFnCTL0 for the control register 0 of CSIFn.

14.1 Features

- Transfer rate: 8 Mbps to 2 kbps (using dedicated baud rate generator)
- Master mode and slave mode selectable
- 8-bit to 16-bit transfer, 3-wire serial interface
- 2 interrupt request signals (INTCFnT, INTCFnR)
- 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

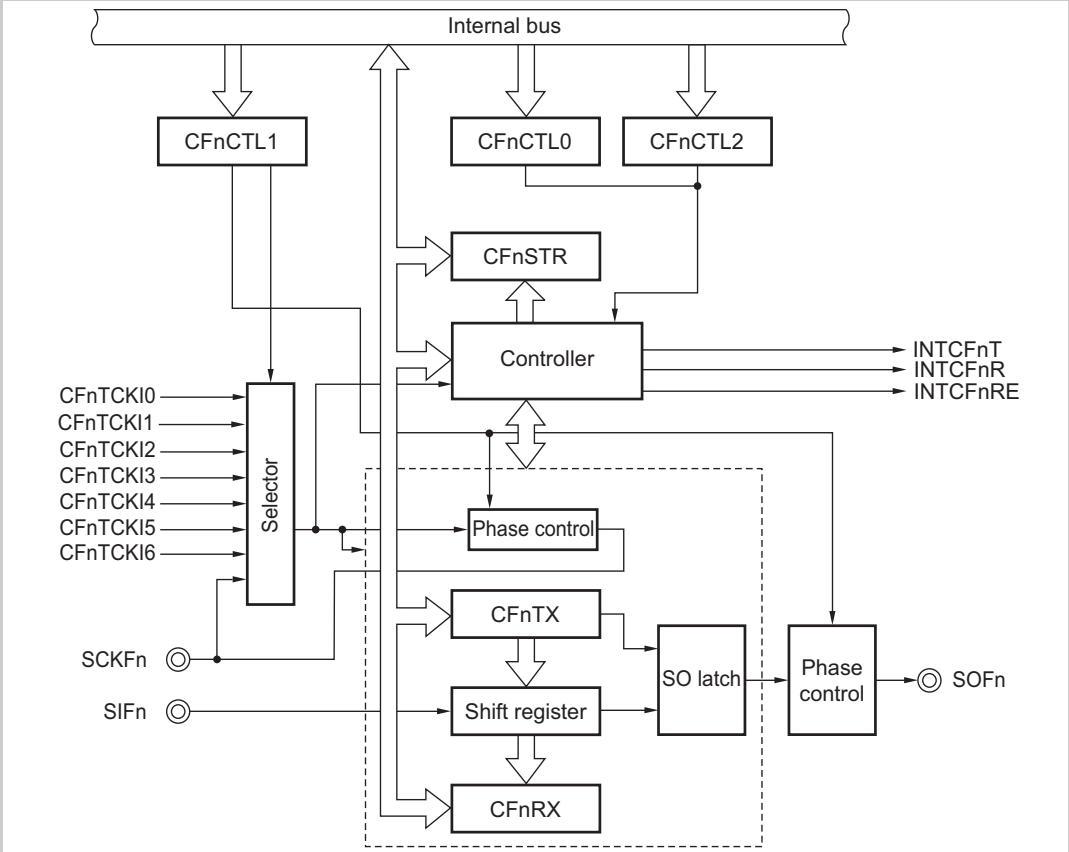
  SOFn: Serial data output
  SIFn: Serial data input
  SCKFn: Serial clock input/output

  Transmission mode, reception mode, and transmission/reception mode selectable
- Dedicated baud rate generator for the two interface instances
- Stable clock sources available

14.2 Configuration

The following shows the block diagram of CSIFn.

Figure 14-1 Block diagram of CSIFn



Clock supply Each Clocked Serial Interface provides 8 clock inputs.

- Six clock inputs CFTCKI0 to CFTCKI5 are connected to the clock generator output clocks (CSIFn in master operation mode).
- One clock input CFTCKI6 is supplied from an internal baud rate generator's output clock BRG1OUT (CSIFn in master mode operation).
- One clock input CFTSCI is provided for CSIFn slave operation from the external clock SCKFn (CSIFn in slave mode operation).

This table lists the cross reference between the CSIFn clock inputs and the connected clocks:

Table 14-2 Clock inputs of CSIFn

CSIFn clock input	CSIFn connected clock	Operation mode
CFnTCKI0	f_{XX}/2 (16 MHz)	Master
CFnTCKI1	f_{XX}/4 (8 MHz)	Master
CFnTCKI2	f_{XX}/8 (4 MHz)	Master
CFnTCKI3	f_{XX}/16 (2 MHz)	Master
CFnTCKI4	f_{XX}/32 (1 MHz)	Master

Table 14-2 Clock inputs of CSIFn

CSIFn clock input	CSIFn connected clock	Operation mode
CFnTCKI5	PCLK6 (500 KHz)	Master
CFnTCKI6	BRG1OUT	Master
CFnTSCI	External clock input SCKFn	Slave

14.3 CSIF Control Registers

Register addresses

All CSIFn register addresses are given as address offsets to the individual base addresses <base> of each CSIFn.

The <base> addresses of each CSIFn are listed in the following table:

Table 14-3 Register <base> addresses of CSIFn

CSIFn	<base> address
CSIF0	FFFF FD00$_H$
CSIF1	FFFF FD20$_H$

The clocked serial interfaces CSIFn are controlled and operated by means of the following registers:

Table 14-4

Register name	Shortcut	Address
CSIFn control register 0	CFnCTL0	<base>
CSIFn control register 1	CFnCTL1	<base> + 1$_H$
CSIFn control register 2	CFnCTL2	<base> + 2$_H$
CSIFn status register	CFnSTR	<base> + 3$_H$
CSIFn receive data register	CFnRX0	<base> + 4$_H$
CSIFn transmit data register	CFnTX0	<base> + 6$_H$

14.3.1 CFnCTL0 - CSIFn control register 0

CFnCTL0 is a register that controls the CSIFn 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
CFnPWR	CFnTXE[a]	CFnRXE[a]	CFnDIR[a]	0	0	CFnTMS[a]	CFnSCE[a]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

a) These bits can only be written when the CFnPWR bit = 0. However, CFnPWR bit = 1 can also be set at the same time as writing these bits.

Table 14-5

CFnPWR	CSIFn operation disable/enable
0	Disable CSIFn operation and reset the CSIFn registers
1	Enable CSIFn operation

The CFnPWR bit controls the CSIFn operation and resets the internal circuit.

Table 14-6

CFnTXE	Transmit operation disable/enable
0	Disable transmit operation
1	Enable transmit operation

The SOFn output is low level when the CFnTXE bit is 0.

Table 14-7

CFnRXE	Receive operation disable/enable
0	Disable receive operation
1	Enable receive operation

When the CFnRXE 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 (CFnRX0 register) is not updated.

Table 14-8

CFnDIR	Transfer direction mode specification (MSB/LSB)
0	MSB first transfer
1	LSB first transfer

Table 14-9

CFnTMS	Transfer mode specification
0	Single transfer mode
1	Continuous transfer mode

Table 14-10

CFnSCE	CSIFn communication trigger enable/disable
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 CFnSCE bit is 1.
  Set the CFnSCE bit to 1.
  (b) In single reception mode
  Clear the CFnSCE bit to 0 before reading the receive data (CFnRX0 register).
  If the CFnSCE bit is read while it is 1, the next communication operation is started.
  (c) In continuous reception mode
  Clear the CFnSCE bit to 0 one communication clock before reception of the last data is completed
  The CFnSCE 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 CFnSCE bit to 1.

14.3.2 CFnCTL1 - CSIFn control register 1

CFnCTL1 is an 8-bit register that controls the CSIFn 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	CFnCKP	CFnDAP	CFnCKS2	CFnCKS1	CFnCKS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Caution The CFnCTL1 register can be written only when the CFnCTL0.CFnPWR bit=0.

Table 14-11

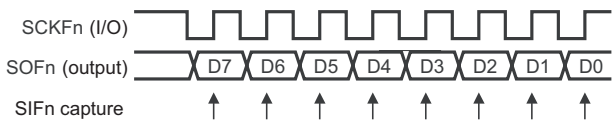
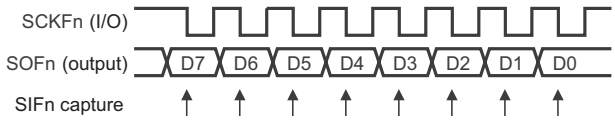
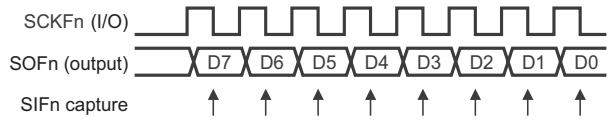
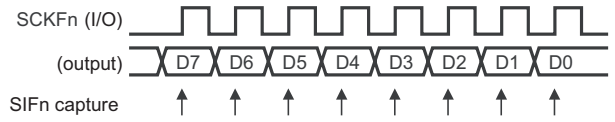
CFnCKP	CFnDAP	SIFn/SOFn timing in relation to SCKFn
0	0	
0	1	
1	0	
1	1	

Table 14-12

CFnCKS2	CFnCKS1	CFnCKS0	Communication clock	Mode
0	0	0	16 MHz	Master
0	0	1	8 MHz	Master
0	1	0	4 MHz	Master
0	1	1	2 MHz	Master
1	0	0	1 MHz	Master
1	0	1	500 KHz	Master
1	1	0	BRGOUT1	Master
1	1	1	SCKFn input	Slave

14.3.3 CFnCTL2 - CSIFn control register 2

CFnCTL2 is an 8-bit register that controls the number of CSIFn 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	CFnCL3	CFnCL2	CFnCL1	CFnCL0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Caution The CFnCTL2 register can be written only when the CFnCTL0.CFnPWR bit = 0 or when both the CFnTXE and CFnRXE bits = 0.

Table 14-13

CFnCL3	CFnCL2	CFnCL1	CFnCL0	Number of serial transfer bits
0	0	0	0	8 bits
0	0	0	1	9 bits
0	0	1	0	10 bits
0	0	1	1	11 bits
0	1	0	0	12 bits
0	1	0	1	13 bits
0	1	1	0	14 bits
0	1	1	1	15 bits
1	x	x	x	16 bits

Note If the number of transfer bits is not 8 or 16, prepare data so that it is justified to to the LSB of the CFnTX0 and CFnRX0 registers. E.g., 101010101010_B should be represented as 0000101010101010_B.

14.3.4 CFnSTR - CSIFn status register

CFnSTR is an 8-bit register that displays the CSIFn status.

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 cleared by any reset.
Also, the CFnSTR register can be initialized by clearing the CFnCTL0.CFnPWR bit to 0.

7	6	5	4	3	2	1	0
CFnTSF	0	0	0	0	0	0	CFnOVE
R	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 14-14

CFnTSF	Transfer operation status flag
0	Idle status
1	Operating status

During transmission, this bit is set when data is written to the CFnTX0 register, and during reception, it is set when a dummy read of the CFnRX0 register is performed.

When transfer ends, this flag is cleared to 0 at the last edge of the clock.

Table 14-15

CFnOVE	Overrun error flag
0	No overrun
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 CFnOVE flag displays the overrun error occurrence status in this case.
- The CFnOVE flag is cleared by writing 0 to it. It is not set when 1 is written to it.

Note In case of an overrun error, the reception error interrupt INTCFnRE behaves differently depending on the transfer mode:

- Continuous transfer mode
  The reception error interrupt INTCFnRE is generated instead of the reception completion interrupt INTCFnR.
- Single transfer mode
  No interrupt is generated.

In either case the overflow flag CFnSTR.CFnOVE is set to 1 and the previous data in CFnRX0 will be overwritten with the new data.

14.3.5 CFnRX0 - CSIFn receive data register

The CFnRX0 register is a 16-bit buffer register that holds receive data.

Access This register is 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 CFnRX0L register.

Address <base> + 4_H

Initial Value 0000_H. This register is cleared by any reset.
Also, the CFnRX0 register can be initialized by clearing the CFnCTL0.CFnPWR bit to 0.

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

The receive operation is started by reading the CFnRX0 register while in the reception enabled mode.

14.3.6 CFnTX0 - CSIF transmit data register

The CFnTX0 register is a 16-bit buffer register used to write the CSIFn 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/written in 8-bit units as the CFnTX0L register.

Address <base> + 6_H

Initial Value 0000_H. This register is cleared by any reset.
Also, the CFnTX0 register can be initialized by clearing the CFnCTL0.CFnPWR bit to 0.

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

The transmit operation is started by writing data to the CFnTX0 register while in the transmission enabled mode.

Note The communication start conditions are shown below:

- Transmission mode (CFnTXE bit = 1, CFnRXE bit = 0):
  Write to CFnTX0 register
- Transmission/reception mode (CFnTXE bit = 1, CFnRXE bit = 1):
  Write to CFnTX0 register
- Reception mode (CFnTXE bit = 0, CFnRXE bit = 1):
  Read from CFnRX0 register

14.4 Transfer data length change function

The CSIFn transfer data length can be set in 1-bit units between 8 and 16 bits using the CFnCTL2.CFnCL3 to CFnCTL2.CFnCL0 bits.

When the transfer bit length is set to a value other than 16 bits, the data written to the CFnTX0 or CFnRX0 register starts from the LSB, regardless of whether the transfer start bit is the MSB or LSB. Any data can be written to the higher bits, which are not used, but the receive data becomes 0 following the transmission of the designated number of transmit bits.

Figure 14-2 Transfer bit length = 10 bits, MSB first

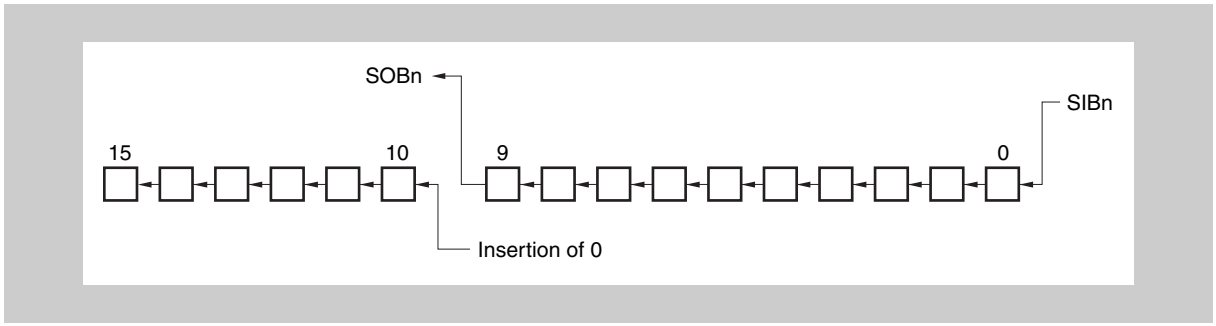
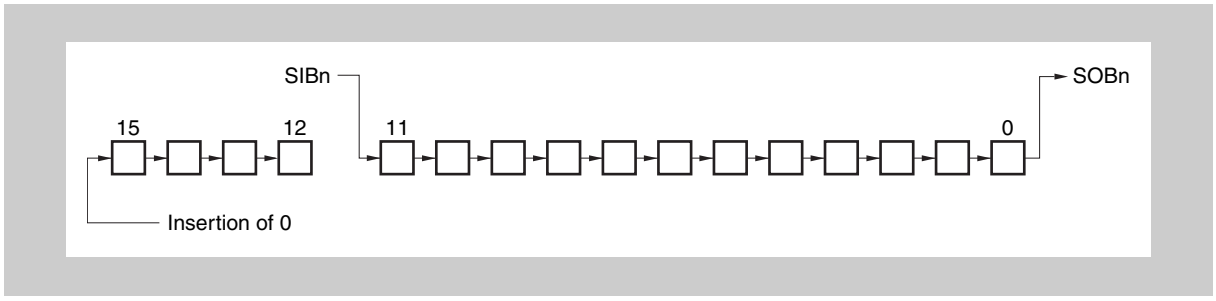


Figure 14-3 Transfer bit length = 12 bits, LSB first



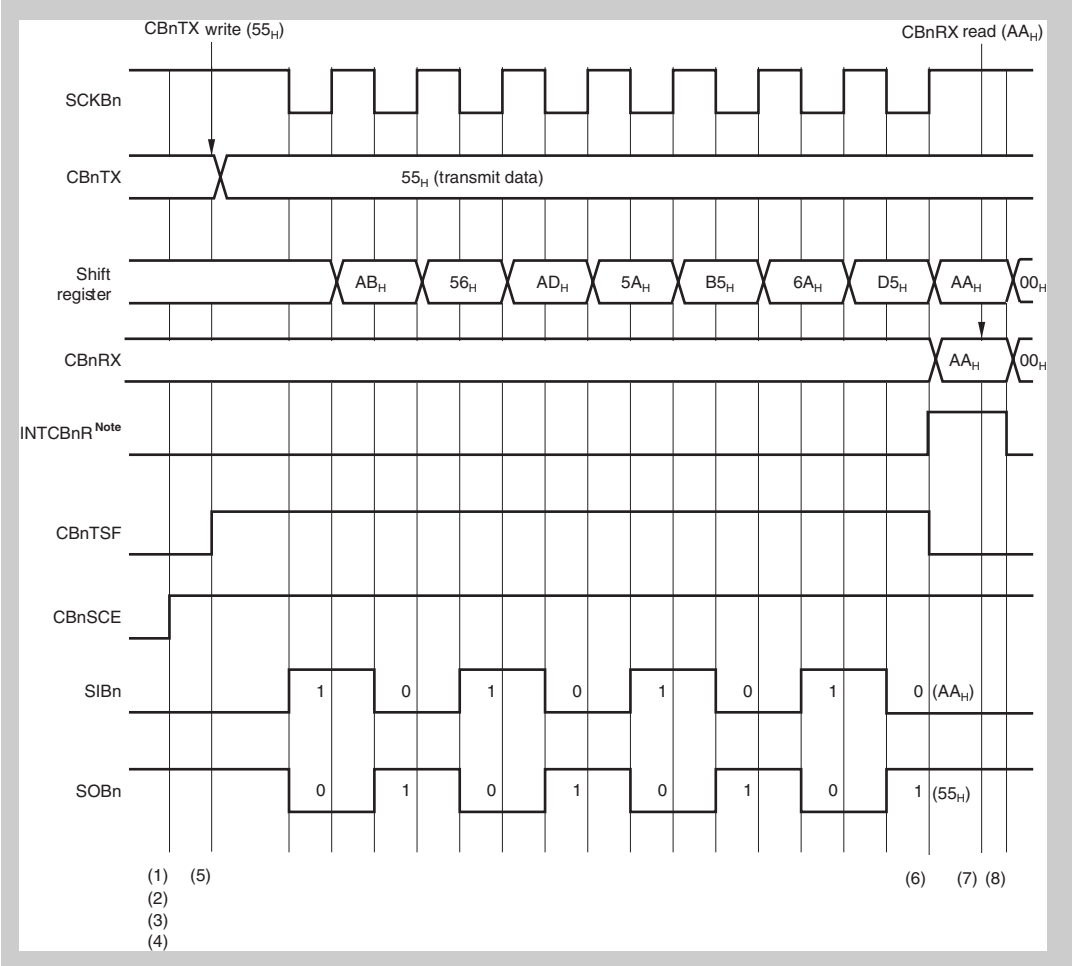
14.5 Operation

14.5.1 Single transfer mode

(1) Master mode, transmission/reception mode)

MSB first (CFnCTL0.CFnDIR bit = 0) communication type 1 (see 16.4 (2) CSIFn control register 1 (CFnCTL1), transfer data length = 8 bits (CFnCTL2.CFnCL3 to CFnCTL2.CFnCL0 bits = 0, 0, 0, 0)

Figure 14-4 Single Transfer Timing (Master, Transmission/Reception Mode)



2. Clear the CFnCTL0.CFnPWR bit to 0.
3. Set the CFnCTL1 and CFnCTL2 registers to specify the transfer mode.
4. Set the CFnTXE, CFnRXE, and CFnSCE bits of the CFnCTL0 register to 1 at the same time as specifying the transfer mode using the CFnDIR bit, to set the transmission/reception enabled status.
5. Set the CFnPWR bit to 1 to enable the CSIFn operation.
6. Write transfer data to the CFnTX0 register (transmission start).
7. The reception complete interrupt request signal (INTCFnR) is output.
8. Read the CFnRX0 register before clearing the CFnPWR bit to 0.
9. Check that the CFnSTR.CFnTSF bit = 0 and set the CFnPWR bit to 0 to stop operation of CSIFn (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 CFnRX0 register.

Note

1. In single transmission or single transmission/reception mode, the INTCFnT signal is not generated. When communication is complete, the INTCFnR signal is generated.
2. The processing of steps (3) and (4) can be set simultaneously.

Caution In case the CSIF interface is operating in

- single transmit/reception mode (CFnCTL0.CFnTMS = 0)
- communication type 2 respectively type 4 (CFnCTL1.CFnDAP = 1)

pay attention to following effect:

In case the next transmit should be initiated immediately after the occurrence of the reception completion interrupt INTCFnR any write to the CFnTX0 register is ignored as long as the communication status flag is still reflecting an ongoing communication (CFnTSF = 1). Thus the new transmission will not be started.

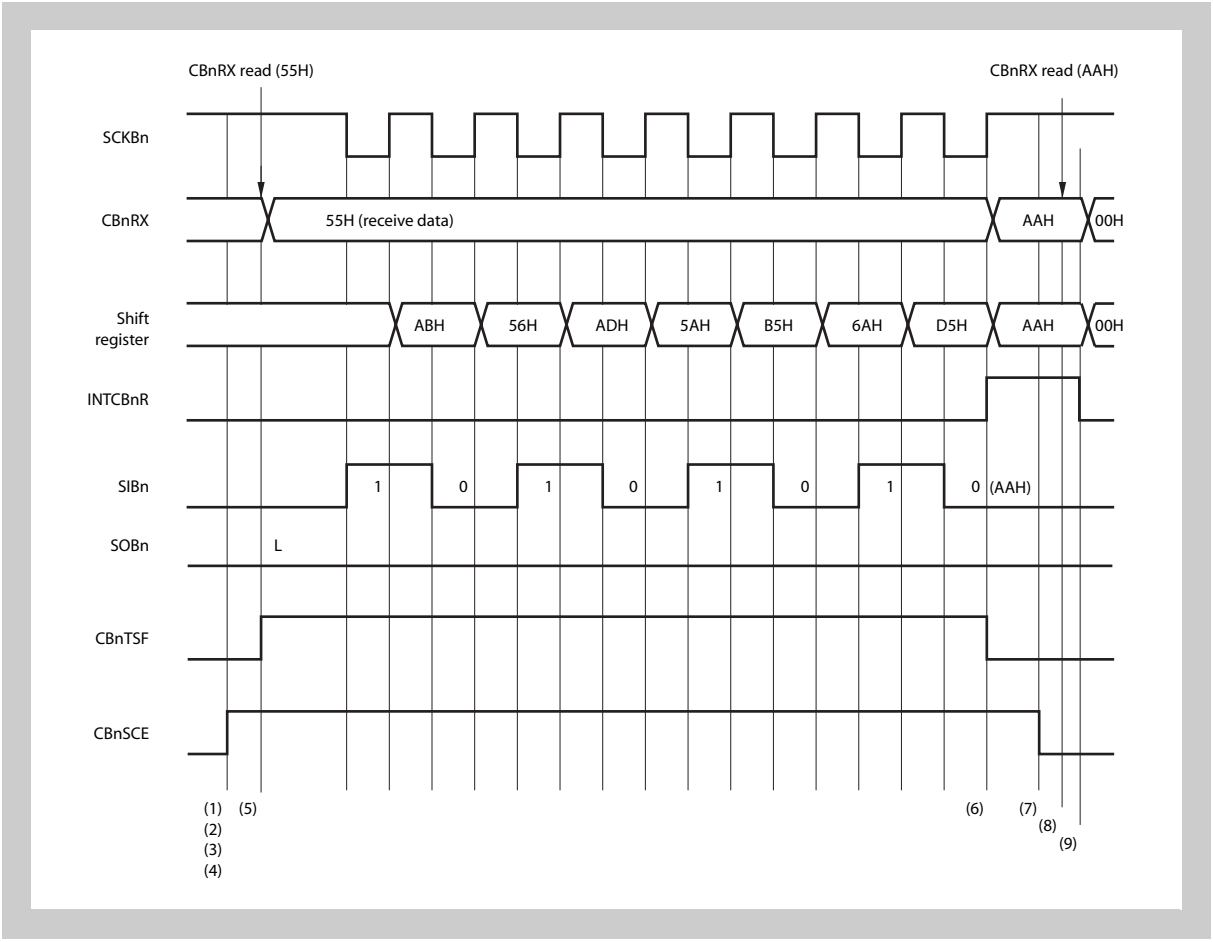
For transmitting data continuously use one of the following options:

- Use continuous transfer mode (CFnCTL0.CFnTMS = 1). This is the only usable mode for automatic transmission of data by the DMA controller.
- If single transfer mode (CFnCTL0.CFnTMS = 0) should be used, CFnSTR.CFnTSF = 0 needs to be verified before writing data to the CFnTX0 register.

(2) Master, Reception Mode

MSB first (CFnCTL0.CFnDIR bit = 0), communication type 1 (see 16.4 (2) CSIFn control register 1 (CFnCTL1), transfer data length = 8 bits (CFnCTL2.CFnCL3 to CFnCTL2.CFnCL0 bits = 0, 0, 0, 0)

Figure 14-5 Single Transfer Timing (Master, Reception Mode)



1. Clear the CFnCTL0.CFnPWR bit to 0.
2. Set the CFnCTL1 and CFnCTL2 registers to specify the transfer mode.
3. Set the CFnCTL0.CFnRXE and CFnCTL0.CFnSCE bits to 1, CFnCTL0.TXE to 0, at the same time as specifying the transfer mode using the CFnDIR bit, to set the reception enabled status.
4. Set the CFnPWR bit to 1 to enable the CSIFn operation.
5. Perform a dummy read of the CFnRX0 register (reception start trigger).
6. The reception complete interrupt request signal (INTCFnR) is output.
7. Set the CFnSCE bit to 0 to set the final receive data status.
8. Read the CFnRX0 register.
9. Check that the CFnSTR.CFnTSF bit = 0 and set the CFnPWR bit to 0 to stop the CSIFn 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.)

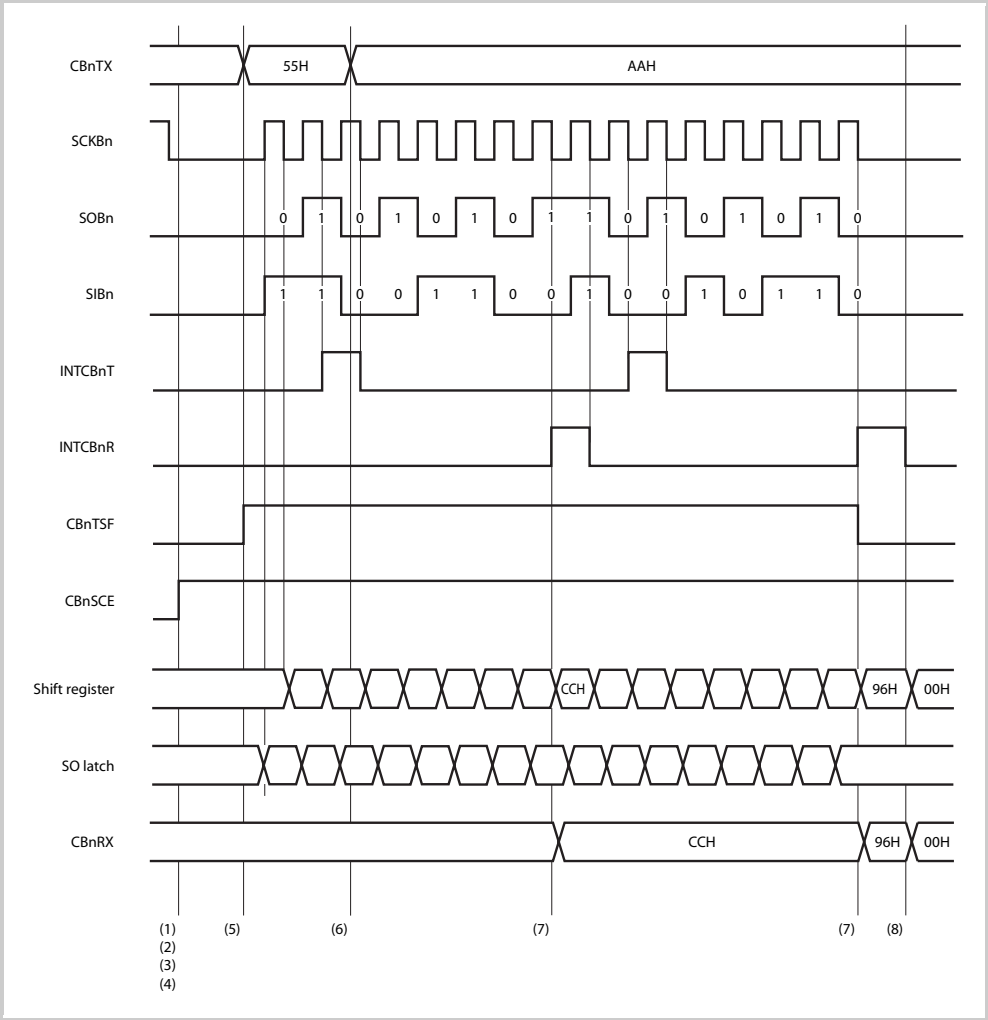
Note The processing of steps (3) and (4) can be set simultaneously.

14.5.2 Continuous mode

(1) Master, transmission/reception mode)

MSB first (CFnCTL0.CFnDIR bit = 0), communication type 3 (see 16.4 (2) CSIFn control register 1 (CFnCTL1)), transfer data length = 8 bits (CFnCTL2.CFnCL3 to CFnCTL2.CFnCL0 bits = 0, 0, 0, 0)

Figure 14-6 Continuous Transfer Timing (Master, Transmission/Reception Mode)



1. Clear the CFnCTL0.CFnPWR bit to 0.
2. Set the CFnCTL1 and CFnCTL2 registers to specify the transfer mode.
3. Set the CFnTXE, CFnRXE, and CFnSCE bits of the CFnCTL0 register to 1 at the same time as specifying the transfer mode using the CFnDIR bit, to set the transmission/reception enabled status.
4. Set the CFnPWR bit to 1 to enable the CSIFn operation.
5. Write transfer data to the CFnTX0 register (transmission start).
6. The transmission enable interrupt request signal (INTCFnT) is received and transfer data is written to the CFnTX0 register.
7. The reception complete interrupt request signal (INTCFnR) is output.

Read the CFnRX0 register before the next receive data arrives or before the CFnPWR bit is cleared to 0.

8. Check that the CFnSTR.CFnTSF bit = 0 and set the CFnPWR bit to 0 to stop the operation of CSIFn (end of transmission/reception).

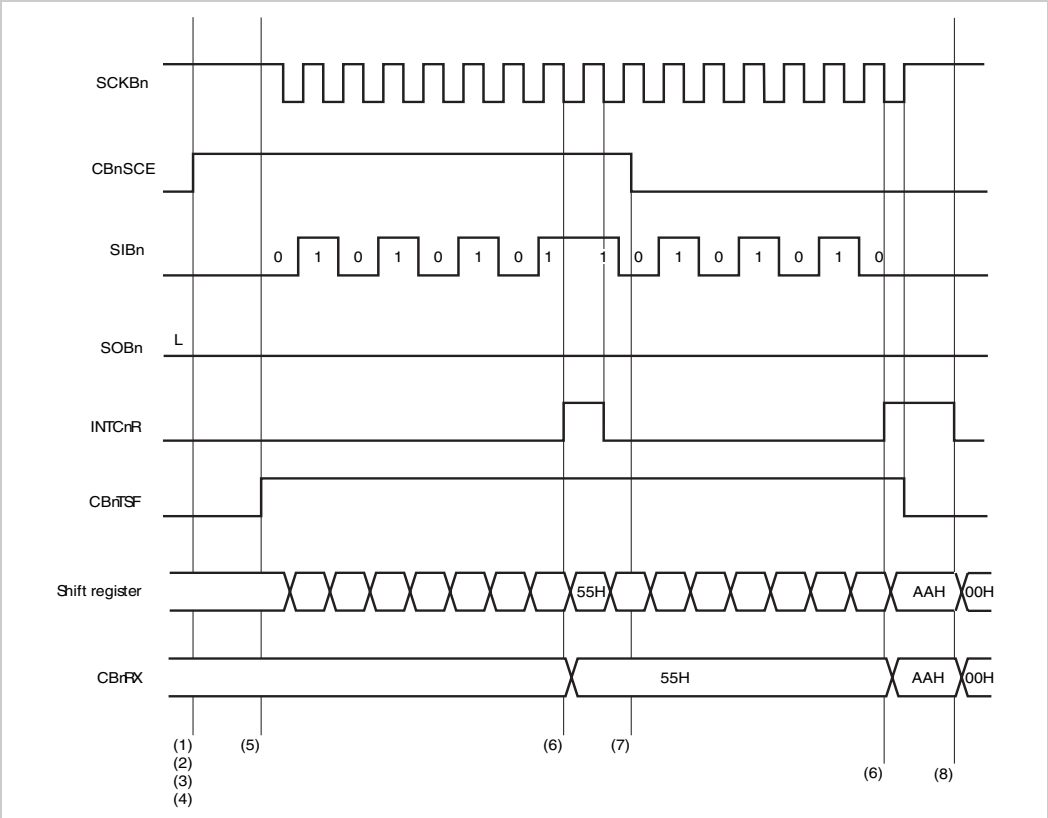
To continue transfer, repeat steps (5) to (7) before (8).

In transmission mode or transmission/reception mode, the communication is not started by reading the CFnRX0 register.

(2) Master, reception mode

MSB first (CFnCTL0.CFnDIR bit = 0), communication type 2 (see 16.4 (2) CSIFn control register 1 (CFnCTL1)), transfer data length = 8 bits (CFnCTL2.CFnCL3 to CFnCTL2.CFnCL0 bits = 0, 0, 0, 0)

Figure 14-7 Continuous transfer timing (master, reception mode)



1. Clear the CFnCTL0.CFnPWR bit to 0.
2. Set the CFnCTL1 and CFnCTL2 registers to specify the transfer mode.
3. Set the CFnCTL0.CFnRXE bit to 1 at the same time as specifying the transfer mode using the CFnDIR bit, to set the reception enabled status.
4. Set the CFnPWR bit to 1 to enable the CSIFn operation.
5. Perform a dummy read of the CFnRX0 register (reception start trigger).
6. The reception complete interrupt request signal (INTCFnR) is output.

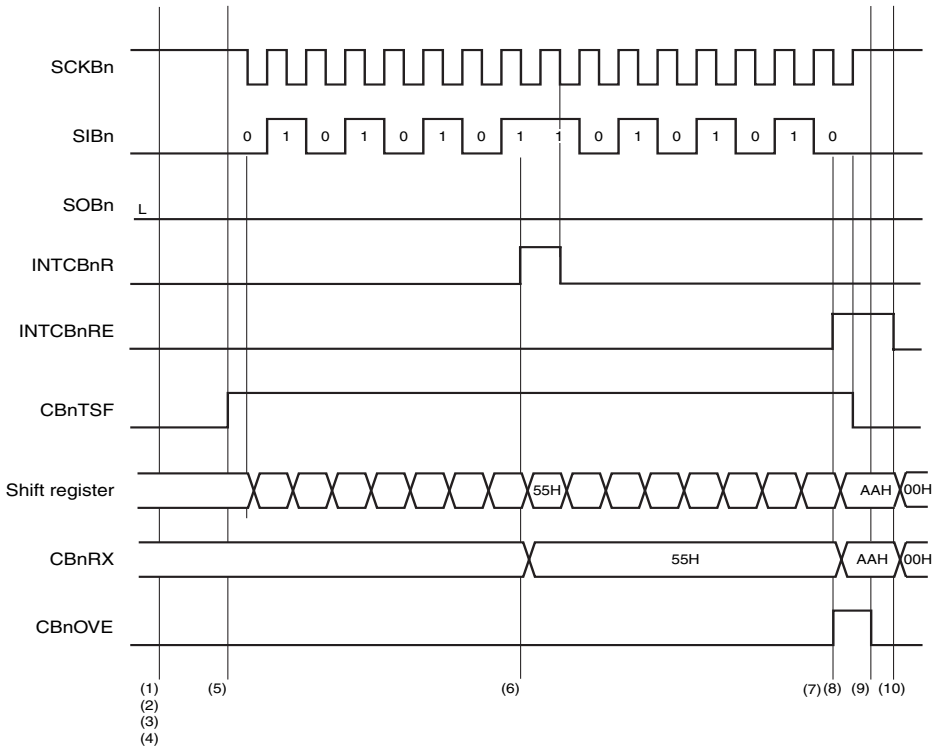
   Read the CFnRX0 register before the next receive data arrives or before the CFnPWR bit is cleared to 0.
7. Set the CFnCTL0.CFnSCE bit = 0 while the last data being received to set the final receive data status.
8. Check that the CFnSTR.CFnTSF bit = 0 and set the CFnPWR bit to 0 to stop the operation of CSIFn (end of reception).

To continue transfer, repeat steps (5) and (6) before (7).

(3) Transmission Error

MSB first (CFnCTL0.CFnDIR bit = 0), communication type 2 (see 16.4 (2) CSIFn control register 1 (CFnCTL1)), transfer data length = 8 bits (CFnCTL2.CFnCL3 to CFnCTL2.CFnCL0 bits = 0, 0, 0, 0)

Figure 14-8 Error during data transfer

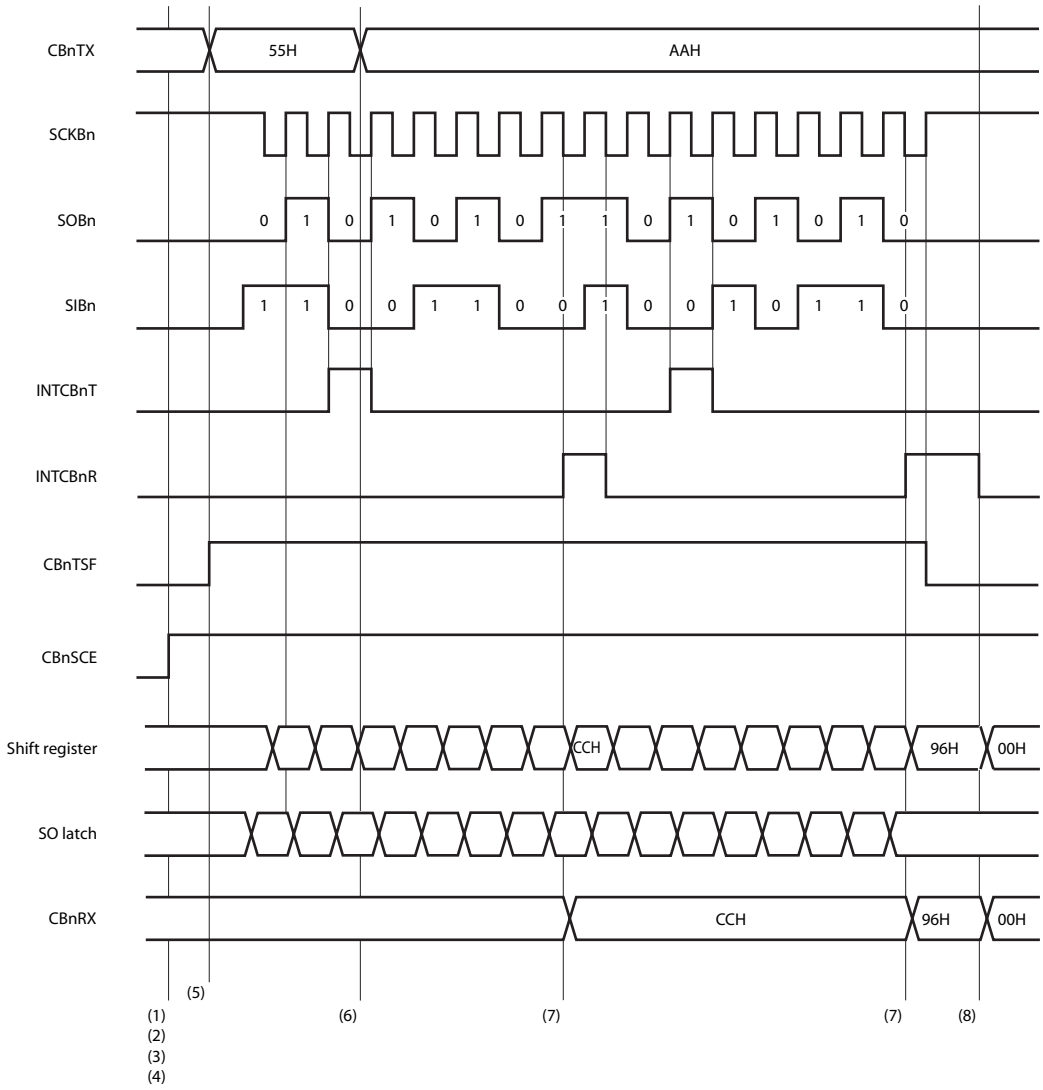


1. Clear the CFnCTL0.CFnPWR bit to 0.
2. Set the CFnCTL1 and CFnCTL2 registers to specify the transfer mode.
3. Set the CFnCTL0.CFnRXE bit to 1 at the same time as specifying the transfer mode using the CFnDIR bit, to set the reception enabled status.
4. Set the CFnPWR bit = 1 to enable CSIFn operation.
5. Perform a dummy read of the CFnRX0 register (reception start trigger).
6. The reception complete interrupt request signal (INTCFnR) is output.
7. If the data could not be read before the end of the next transfer, the CFnSTR.CFnOVE flag is set to 1 upon the end of reception and the reception error interrupt INTCFnRE is output.
8. Overrun error processing is performed after checking that the CFnOVE bit = 1 in the INTCFnRE interrupt servicing.
9. Clear CFnOVE bit to 0.
10. Check that the CFnSTR.CFnTSF bit = 0 and set the CFnPWR bit to 0 to stop the operation CSIFn (end of reception).

(4) Continuous mode (slave mode, transmission/reception mode)

MSB first (CFnCTL0.CFnDIR bit = 0), communication type 2 (see 16.4 (2) CSIFn control register 1 (CFnCTL1)), transfer data length = 8 bits (CFnCTL2.CSnCL3 to CFnCTL2.CFnCL0 bits = 0, 0, 0, 0)

Figure 14-9 Continuous transfer timing (slave, transmission/reception mode)



1. Clear the CFnCTL0.CFnPWR bit to 0.
2. Set the CFnCTL1 and CFnCTL2 registers to specify the transfer mode.
3. Set the CFnTXE, CFnRXE and CFnSCE bits of the CFnCTL0 register to 1 at the same time as specifying the transfer mode using the CFnDIR bit, to set the transmission/reception enabled status.
4. Set the CFnPWR bit to 1 to enable supply of the CSIFn operation.
5. Write the transfer data to the CFnTX0 register.
6. The transmission enable interrupt request signal (INTCFnT) is received and the transfer data is written to the CFnTX0 register.
7. The reception complete interrupt request signal (INTCFnR) is output.

Read the CFnRX0 register.

8. Check that the CFnSTR.CFnTSF bit = 0 and set the CFnPWR bit to 0 to stop the operation of CSIFn (end of transmission/reception).

To continue transfer, repeat steps (5) to (7) before (8).

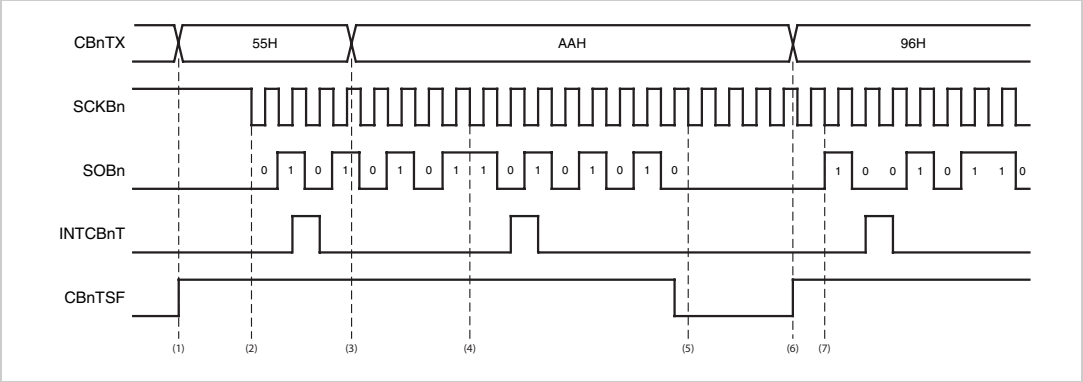
Note In order to start the entire data transfer the CFnTX0 register has to be written initially, as done in step (5) above. If this step is omitted also no data will be received.

Discontinued transmission

In case the CSIF is operating in continuous slave transmission mode (CFnCTL0.CFnTMS = 1, CFnCTL1.CFnCKS[2:0] = 111_B) and new data is not written to the CFnTX0 register the SOFn pin outputs the level of the last bit.

Figure 14-10 outlines this behavior.

Figure 14-10 Discontinued transmission timing (slave, transmission/reception mode)



The example shows the situation that two data bytes (55_H, AA_H) are transmitted correctly, but the third (96_H) fails.

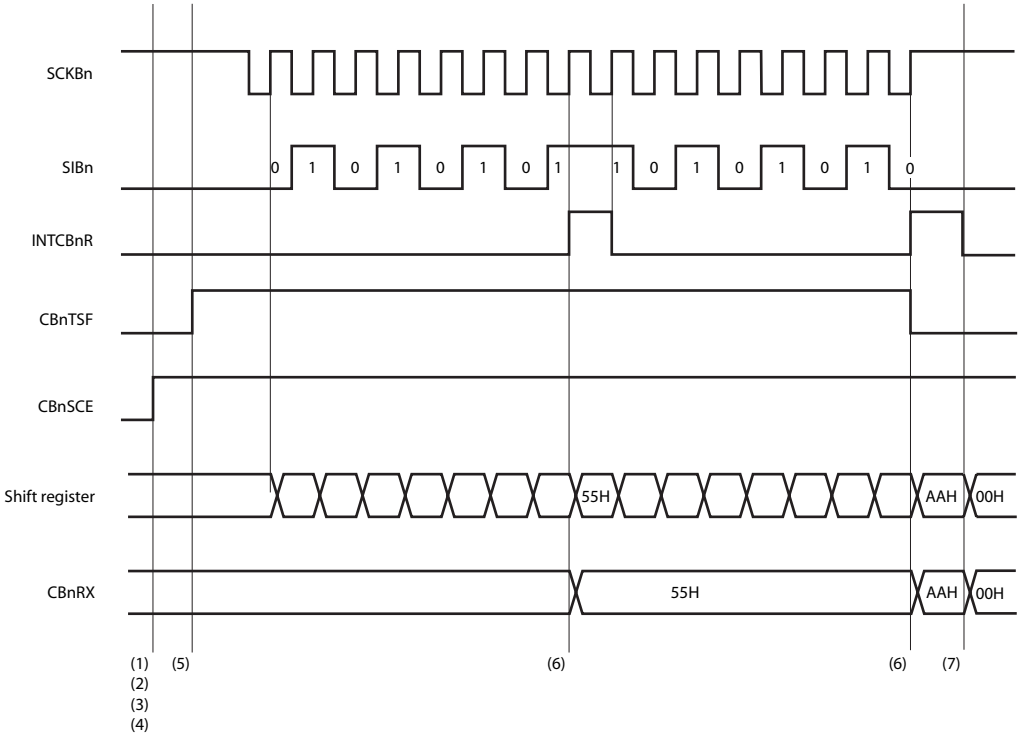
1. Data 55_H is written (by the CPU or DMA) to CFnTX0.
2. The master issues the clock SCKFn and transmission of 55_H starts.
3. INTCFnT is generated and the next data AA_H is written to CFnTX0 promptly, i.e. before the first data has been transmitted completely.
4. Transmission of the second data AA_H continues correctly and INTCFnT is generated. But this time the next data is not written to CFnTX0 in time.
5. Since there is no new data available in CFnTX0, but the master continuous to apply SCKFn clocks, SOFn remains at the level of the transmitted last bit.
6. New data (96_H) is written to CFnTX0.
7. With the next SCKFn 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.

(5) Slave, reception mode

MSB first (CFnCTL0.CFnDIR bit = 0), communication type 1 (see 16.4 (2) CSIFn control register 1 (CFnCTL1)), transfer data length = 8 bits (CFnCTL2.CFnCL3 to CFnCTL2.CFnCL0 bits = 0, 0, 0, 0)

Figure 14-11 Continuous transfer timing (slave, reception mode)



1. Clear the CFnCTL0.CFnPWR bit to 0.
2. Set the CFnCTL1 and CFnCTL2 registers to specify the transfer mode.
3. Set the CFnCTL0.CFnRXE and CFnCTL0.CFnSCE bits to 1 at the same time as specifying the transfer mode using the CFnDIR bit, to set the reception enabled status.
4. Set the CFnPWR bit = 1 to enable CSIFn operation.
5. Perform a dummy read of the CFnRX0 register (reception start trigger).
6. The reception complete interrupt request signal (INTCFnR) is output.
7. Read the CFnRX0 register.
8. Check that the CFnSTR.CFnTSF bit = 0 and set the CFnPWR bit to 0 to stop the operation of CSIFn (end of reception).

To continue transfer, repeat steps (5) and (6) before (7).

14.5.3 Clock timing

Figure 14-12 Clock timing: (CFnCKP = 0, CFnDAP = 0)

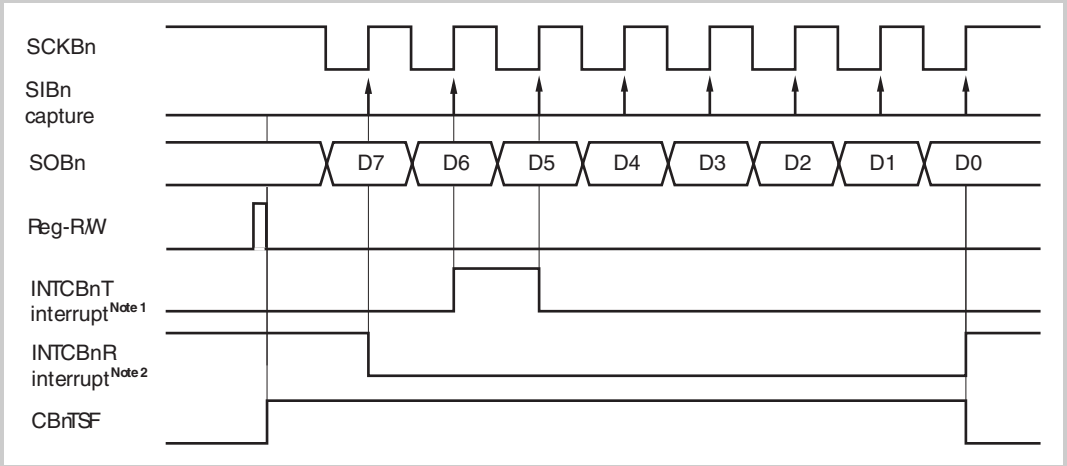


Figure 14-13 Clock timing: (CFnCKP = 1, CFnDAP = 0)

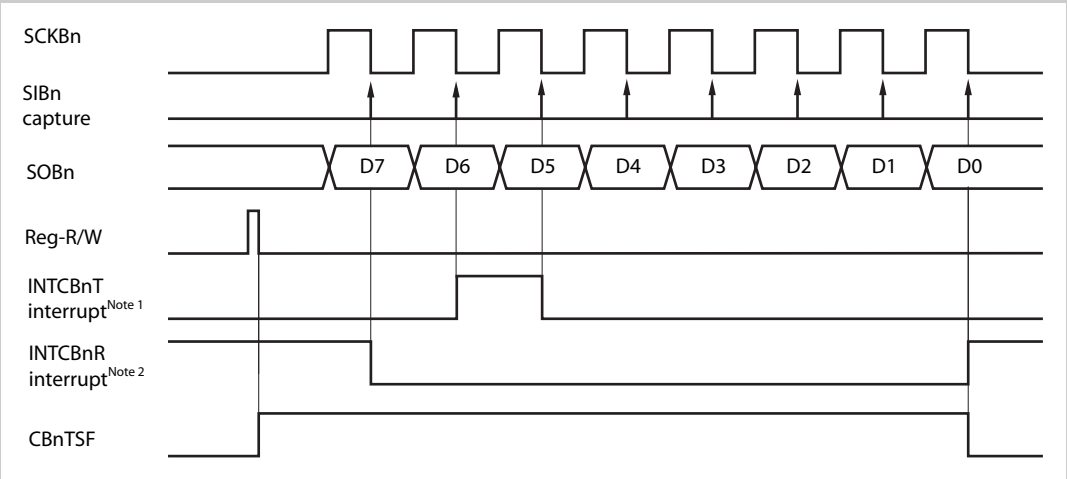


Figure 14-14 Clock timing: (CFnCKP = 0, CFnDAP = 1)

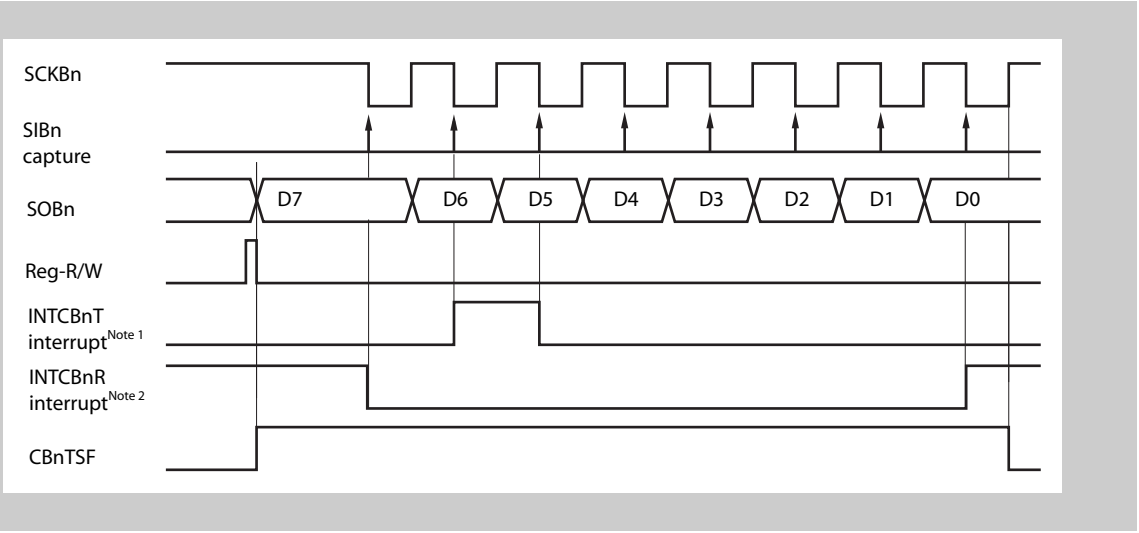
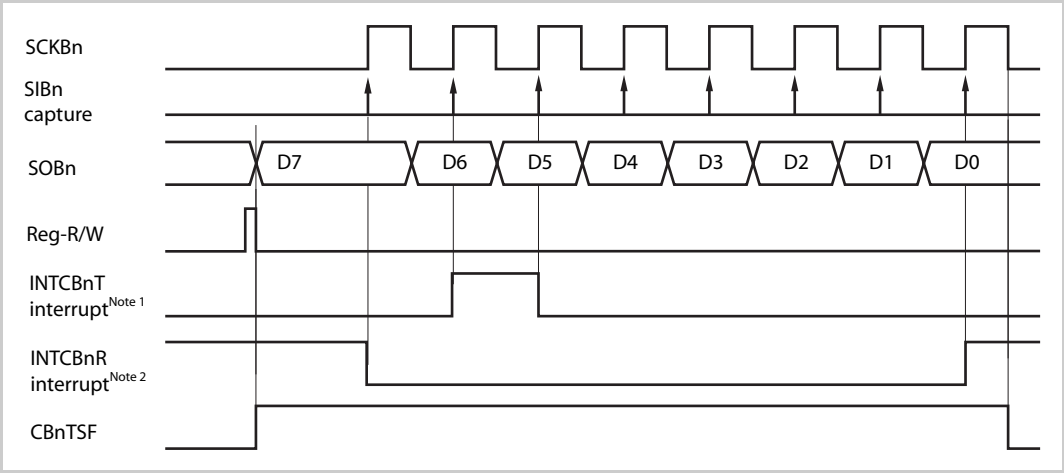


Figure 14-15 Clock timing: (CFnCKP = 1, CFnDAP = 1)

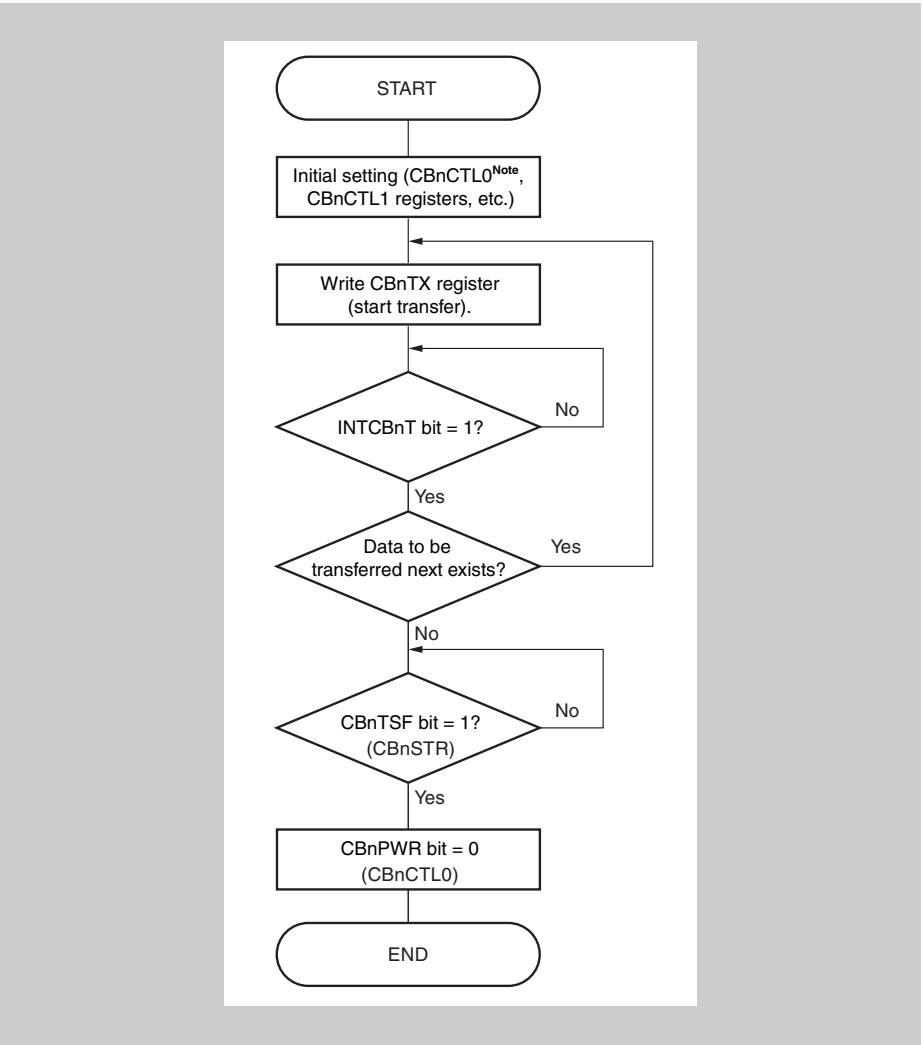


Note 1. The INTCFnT interrupt is set when the data written to the transmit buffer is transferred to the data shift register in the continuous transmission or continuous transmission/reception modes. In the single transmission or single transmission/reception modes, the INTCFnT interrupt request signal is not generated, but the INTCFnR interrupt request signal is generated upon completion of communication.

2. The INTCFnR interrupt occurs if reception is correctly completed and received data is ready in the CFnRX0 register while reception is enabled, and if an overrun error occurs. In the single mode, the INTCFnR interrupt request signal is generated even in the transmission mode, upon completion of communication.

14.6 Operation Flow

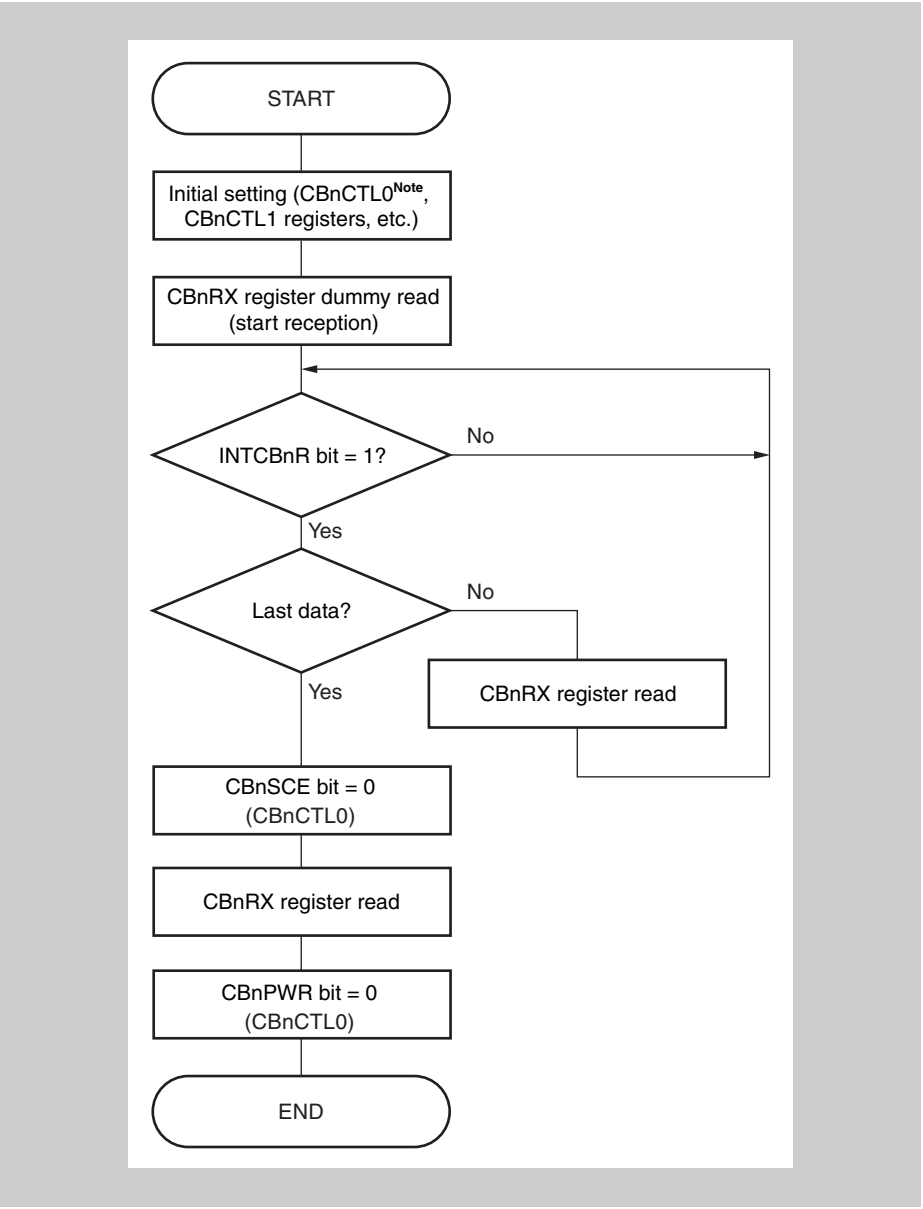
14.6.1 Single transmission



Note Set the CFnSCE 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 CFnTX0 register is written.

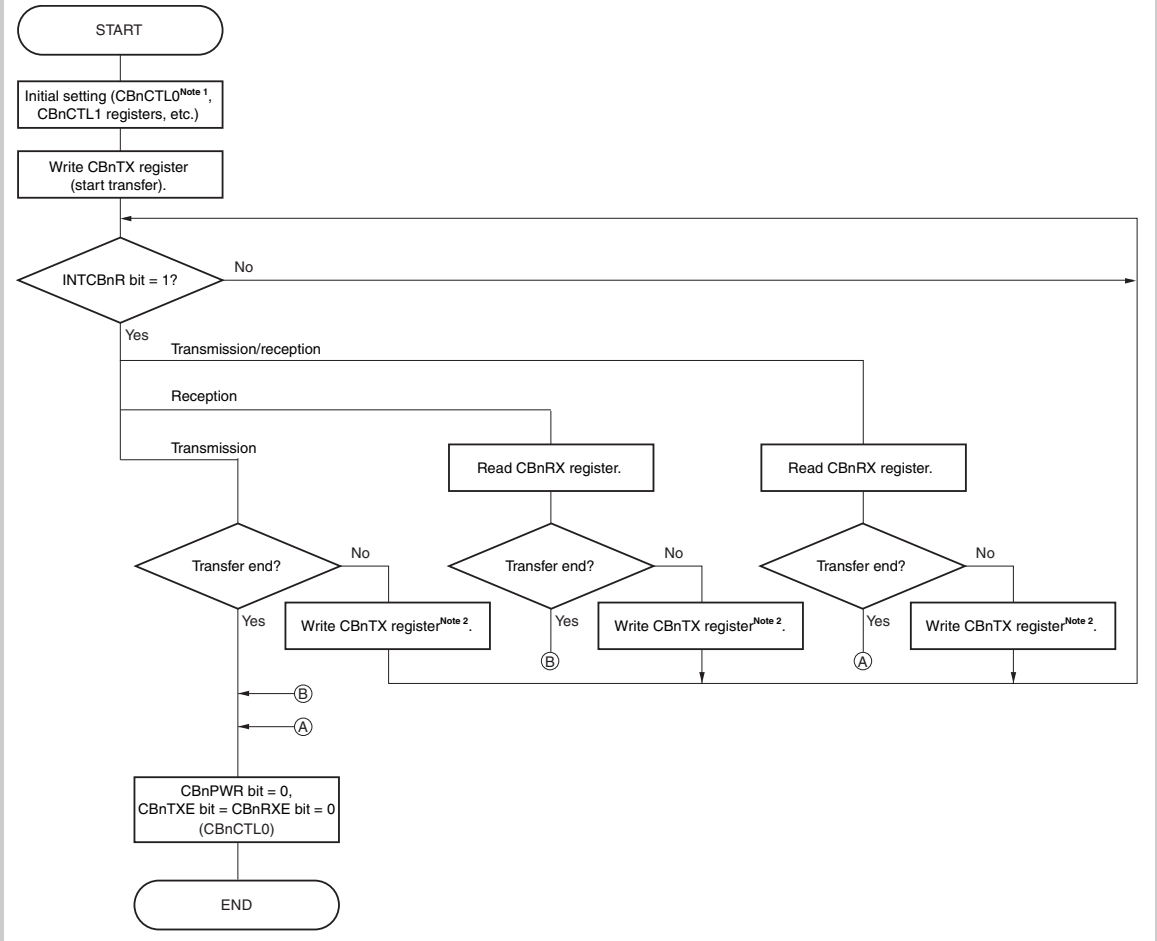
14.6.2 Single reception



Note Set the CFnSCE 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 CFnRX0 register is read.

14.6.3 Single transmission/reception

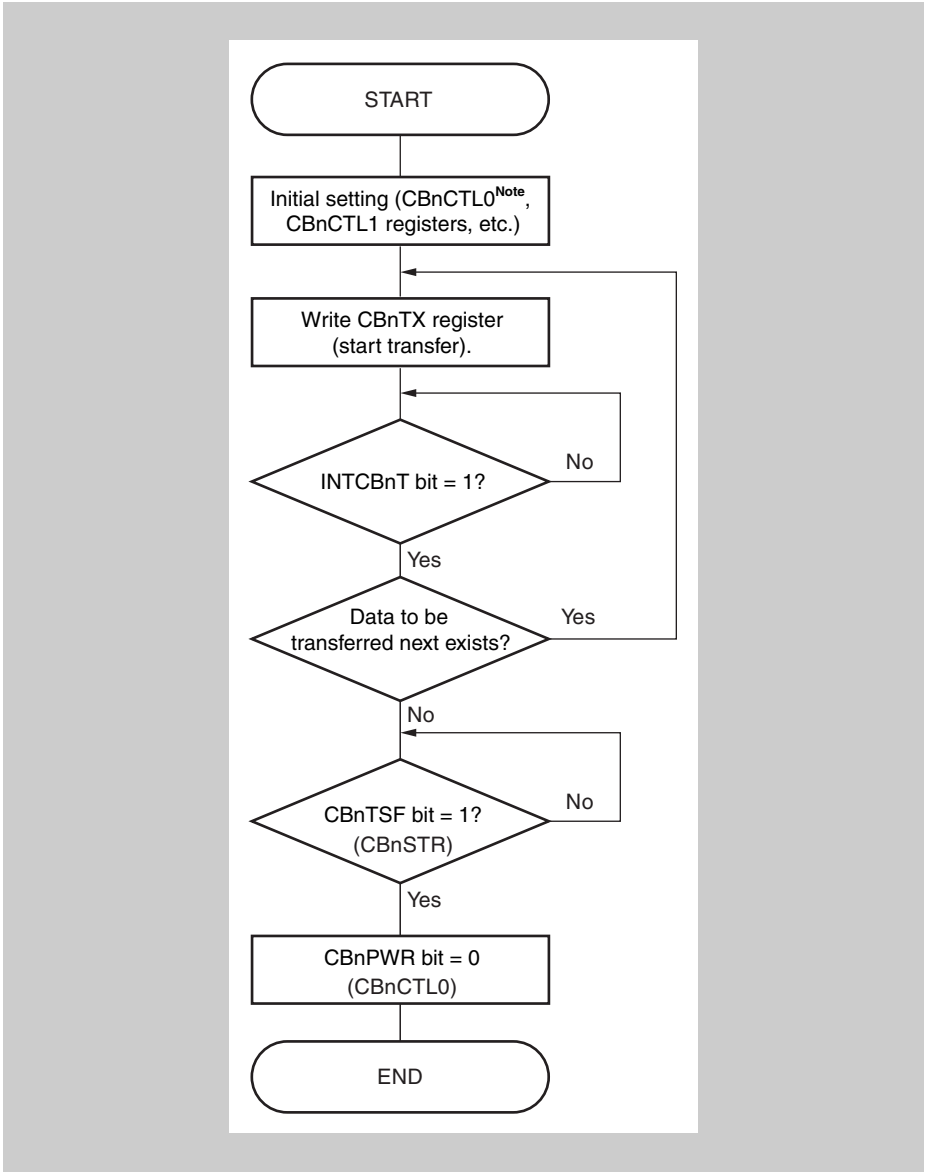


Note 1. Set the CFnSCE bit to 1 in the initial setting.

2. If the next transfer is reception only, dummy data is written to the CFnTX0 register.

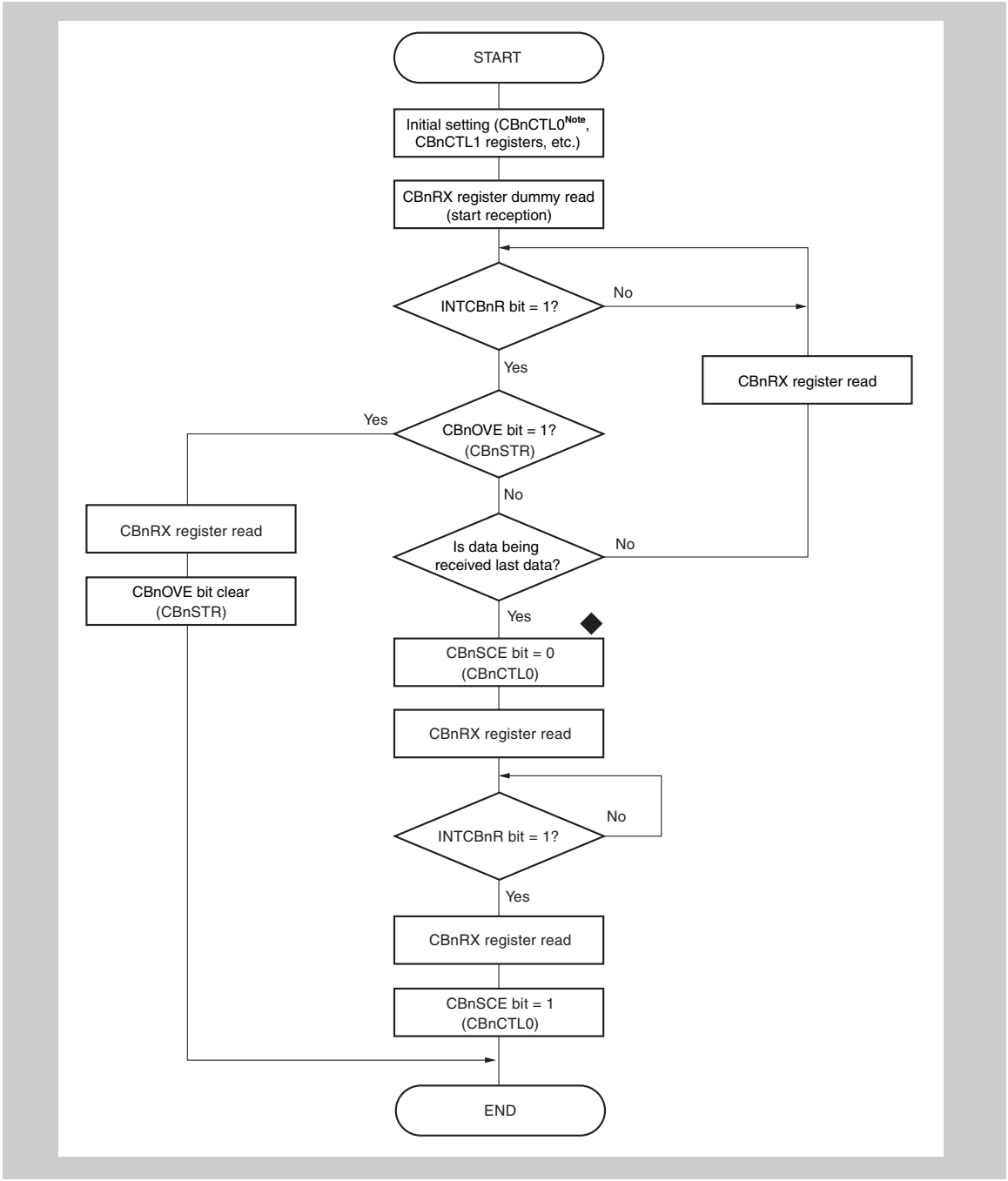
Caution Even in the single mode, the CFnSTR.CFnOVE flag is set to 1. If only transmission is used in the transmission/reception mode, therefore, programming without checking the CFnOVE flag is recommended.

14.6.4 Continuous transmission



Note Set the CFnSCE bit to 1 in the initial setting.

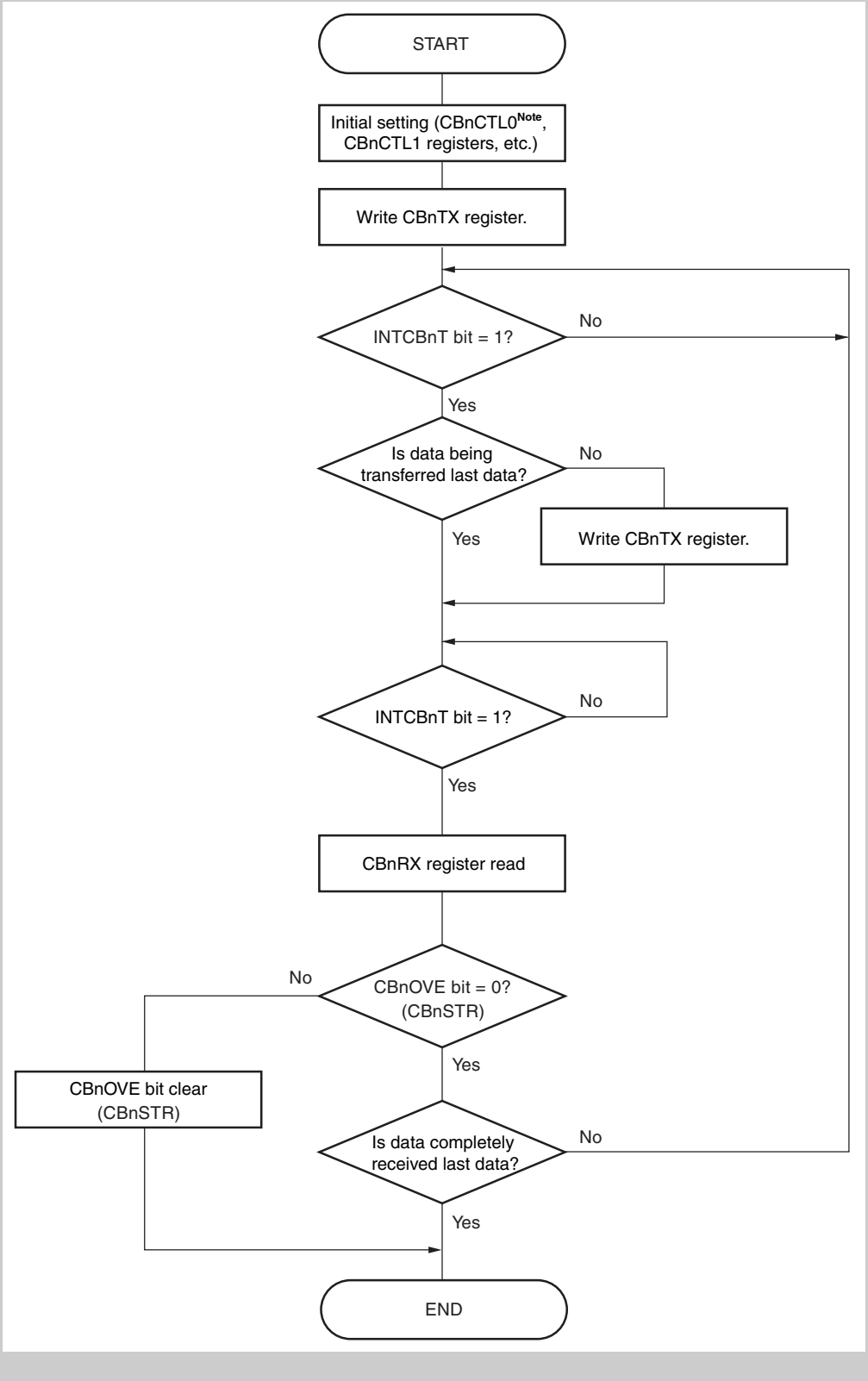
14.6.5 Continuous reception



Note Set the CFnSCE 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 CFnRX0 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 CFnCTL0.CFnSCE bit to 1, and read dummy data from the CFnRX0 register.

14.6.6 Continuous transmission/reception



Note Set the CFnSCE bit to 1 in the initial setting.

14.7 Output Pins

14.7.1 SCKFn pin

When CSIFn operation is disabled (CFnCTL0.CFnPWR bit = 0), the SCKFn pin output status is as follows.

Table 14-16

CFnCKP	CFnCKS2	CFnCKS1	CFnCKS0	SCKFn 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 SCKFn pin changes if any of the CFnCTL1.CFnCKP and CFnCKS2 to CFnCKS0 bits is rewritten.

14.7.2 SOFn pin

When CSIFn operation is disabled (CFnPWR bit = 0), the SOFn pin output status is as follows.

Table 14-17

CFnTXE	CFnDAP	CFnDIR	SOFn pin output
0	×	×	Fixed to low level
1	0	×	SOFn latch value (low level)
	1	0	CFnTXn value (MSB)
		1	CFnTXn value (LSB)

Note 1. The SOFn pin output changes when any one of the CFnCTL0.CFnTXE, CFnCTL0.CFnDIR bits, and CFnCTL1.CFnDAP bit is rewritten.

2. ×: don't care

14.8 Cautions

14.8.1 CSIFn behavior during debugger break

The CSIFn continues to operate in debugger break-mode, provided all clocks are operating.

- in continuous reception/transmission mode
- in slave reception/transmission mode

Reception

When the CSIFn is in reception mode and an external device is sending data during debugger break-mode the CSIFn may produce an overflow error as it continues to receive data during break-mode.

Thus the overflow error flag UCFnSTR.CFnOVE may be set and the reception error interrupt INTCFnRE may be generated. All additional received data is discarded.

Note that the reception error interrupt INTCFnRE will not be serviced during break-mode. After resuming run-mode, if the interrupt controller is configured accordingly, the interrupt will then be serviced.

Transmission

If the debugger break-mode is entered while the CSIFn is transmitting data, the transmission is completed, and a transmission enable interrupt request INTCFnT is generated.

The transmission enable interrupt INTCNnT will not be serviced during break-mode. After resuming run-mode, if the interrupt controller is configured accordingly, the interrupt will then be serviced.

DMA reception

If the DMA controller is used to fetch received data from the CFnRX0 register, the DMA controller continues to do so in debugger break-mode, but the data in CFnRX0 is not tagged as already read. Thus the next receive data during break-mode will generate an overflow error, as described above.

If the specified number of data units in the DBCn register has been fetched from the CSIFn, the DCHCn.TCn is set and the DMA completion interrupt INTDMAn is generated. The INTDMAn request is serviced after resuming run-mode.

Since the DMA trigger interrupt INTCFnR is not generated in case of an overflow (the INTCFnRE interrupt is generated instead), no further DMA triggers occur. Only the first received data in break-mode is transferred by the DMA, all additional data is discarded.

DMA transmission

If the DMA controller is used to write transmission data to the CFnTX0 register, the DMA controller continues to do so in debugger break-mode. If the specified number of data units in the DBCn register has been transferred to the CSIFn, the DCHCn.TCn is set and the DMA completion interrupt INTDMAn is generated. The INTDMAn request is served after resuming run-mode.

Chapter 15 Queued Clocked Serial Interface (CSIE)

Instances The V850E/RG3 microcontroller has two instances of the Enhanced Queued Clocked Serial Interface CSIE: CSIE0 and CSIE1.

Table 15-1 Instances of CSIE

CSIE	
Instances	2
Names	CSIE0, CSIE1

Throughout this chapter, the individual instances of CSIE are identified by “n” (n = 0 to 1), for example CEnCTL2 for the Queued CSIn control register 2.

The individual chip select signals are identified by “m” (m = 0 to 7 for CSIE0 and m = 0 to 3 for CSIE1), for example SCSE06 for the chip select pin 6 of CSIE0.

Clock supply Each Queued Clocked Serial Interface provides 1 clock input.

Table 15-2 Clock input of CSIEn

CSIEn clock input	Connected clock
CETCKI	PCLK0 (32 MHz)

Pinning The following pins are provided to enable a 3-wire serial interface:

Pin Function	CSIE0	CSIE1
Serial data output	SOE0	SOE1
Serial data input	SIE0	SIE1
Serial clock I/O	SCKE0	SCKE1
Chip select	SCSE00 to SCSE07	SCSE10 to SCSE17

15.1 Features

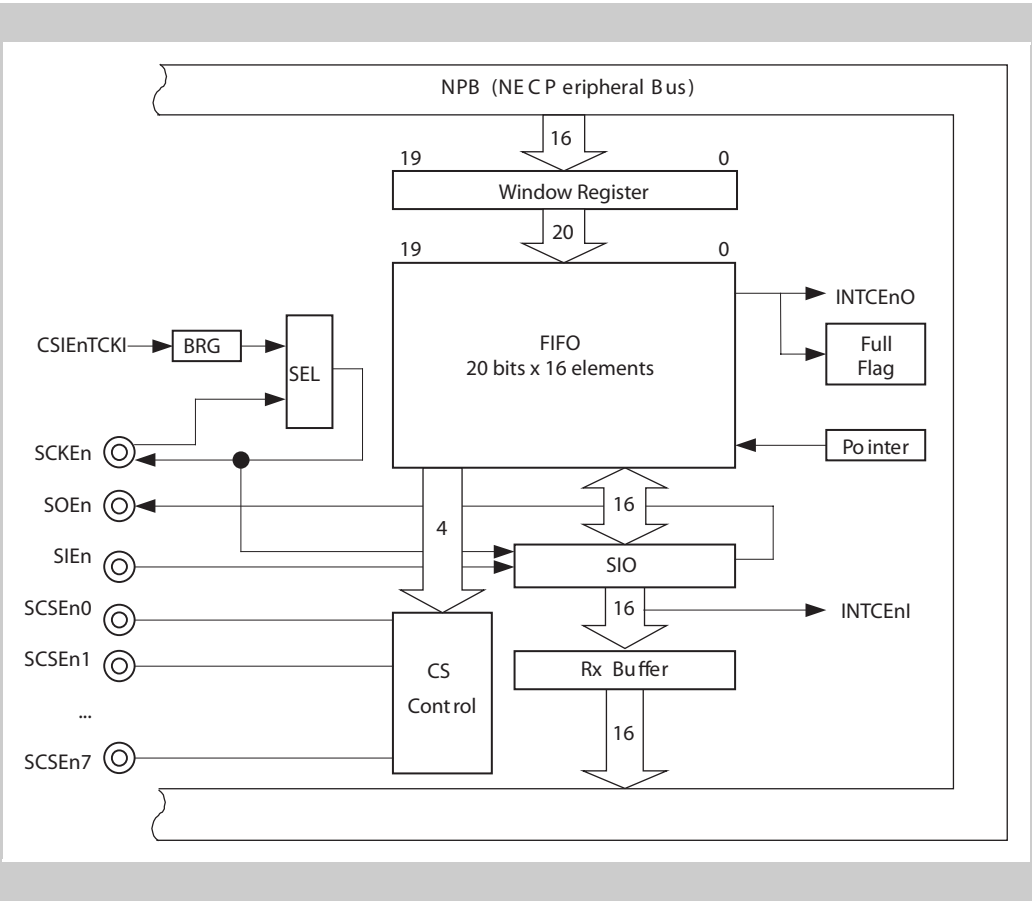
- 3-wire serial synchronous transfers
- Master mode and slave mode selectable
- Serial clock and data phase selectable
- Transfer date length selectable from 8 to 16 bits in 1-bit units
- Data transfer with MSB- or LSB-first selectable
- Three selectable transfer modes:
  - transmit only mode
  - receive only mode
  - transmit/receive mode
- Transmit and receive FIFO (16 elements)
- Selection between single transfer mode and block transfer mode
- Internal baud rate generator

- Programmable baud rate through BRG output (master) or slave clock
- DMA transfer of received data to memory available
- Maximum SCKE frequency: 8 MHz

Note In all figures of this chapter, except where explicitly mentioned, it is assumed that the chip select signals SCSEn0 to SCSEn7 are set as “active low”, although it is possible to define the active level for each chip select also to “active high”.

15.1.1 Queued CSI block diagram

Figure 15-1 Queued CSI Block Diagram



15.1.2 Input/Output pins

The table below shows the input/output pins of the CSIE.

Table 15-3 Input/Output Pins of the CSIE

Signal name	I/O	Active level	Disabled level	Function
SCKEn	I/O	–	H	Serial clock signal
SIEn	I	–	–	Input serial data signal
SOEn	O	–	–	Output serial data signal
SCSEn0	O	L[a]	H[a]	Serial peripheral chip select signal 0
SCSEn1	O	L[a]	H[a]	Serial peripheral chip select signal 1
SCSEn2	O	L[a]	H[a]	Serial peripheral chip select signal 2
SCSEn3	O	L[a]	H[a]	Serial peripheral chip select signal 3
SCSEn4	O	L[a]	H[a]	Serial peripheral chip select signal 4
SCSEn5	O	L[a]	H[a]	Serial peripheral chip select signal 5
SCSEn6	O	L[a]	H[a]	Serial peripheral chip select signal 6
SCSEn7	O	L[a]	H[a]	Serial peripheral chip select signal 7

a) The active level is programmable for each chip select.

15.2 Queued CSI Control Registers

Register addresses

All CSIEn register addresses are given as address offsets to the individual base addresses <base> of each CSIEn.

The <base> addresses of each CSIEn is listed in the following table:

Table 15-4 Register <base> addresses of CSIEn

CSIEn	<base> address
CSIE0	FFFF FD40$_H$
CSIE1	FFFF FD80$_H$

The Enhanced Queued CSIEs are controlled and operated by means of the following registers:

Table 15-5 CSIEn register overview (1/2)

Register name	Shortcut	Address
Queued CSI control register 0	CEnCTL0	<base>
Queued CSI control register 1	CEnCTL1	<base> + 01$_H$
Receive data buffer	CEnRX0	<base> + 02$_H$
Receive data buffer L	CEnRXL0	<base> + 02$_H$
Receive data buffer H	CEnRXH0	<base> + 03$_H$
Chip Select FIFO buffer	CEnCS	<base> + 04$_H$
Chip Select FIFO buffer L	CEnCSL	<base> + 04$_H$
Chip Select FIFO buffer H	CEnCSH	<base> + 05$_H$
Transmit data FIFO buffer	CEnTX0	<base> + 06$_H$
Transmit data FIFO buffer L	CEnTXL0	<base> + 06$_H$

Table 15-5 CSIEn register overview (2/2)

Register name	Shortcut	Address
Transmit data FIFO buffer H	CEnTXH0	<base> + 07_H
CSI Buffer Status Register	CEnSTR	<base> + 08_H
Queued CSI control register 2	CEnCTL2	<base> + 09_H
Queued CSI control register 3	CEnCTL3	<base> + $0C_H$
Queued CSI control register 4	CEnCTL4	<base> + $0D_H$
CSEn0 enhanced timing	CEnOPT0	<base> + 10_H
CSEn1 enhanced timing	CEnOPT1	<base> + 12_H
CSEn2 enhanced timing	CEnOPT2	<base> + 14_H
CSEn3 enhanced timing	CEnOPT3	<base> + 16_H
CSEn4 enhanced timing	CEnOPT4	<base> + 18_H
CSEn5 enhanced timing	CEnOPT5	<base> + $1A_H$
CSEn6 enhanced timing	CEnOPT6	<base> + $1C_H$
CSEn7 enhanced timing	CEnOPT7	<base> + $1E_H$
Chip Select Control	CSGLCTLn	*

* See *Section 15.2.6, CSGLCTLn - Chip Select Control Register* on page 321

15.2.1 CEnCTL0 - Queued CSI control register 0

The CEnCTL0 register controls Queued CSI operation.

Access This register can be read/written in 8-bit or 1-bit units. The CEnTMS, CEnDIR, CEnSIT, CEnWE, CEnCSM bits can only be written when CEnTXE = 0 and CEnRXE = 0.

Address <base>

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

7	6	5	4	3	2	1	0
CEnPWR	CEnTXE	CEnRXE	CEnTMS	CEnDIR	CEnSIT	CEnWE	CEnCSM
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

CEnPWR	Queued CSI operation clock control
0	Stop macro operation clock (Reset internal control circuits)
1	Provide macro operation clock
Clearing CEnPWR = “0” resets the internal circuits asynchronously, stops operation and sets the Queued CSI to standby state. Input clock is not provided to internal circuits. Set CEnPWR = “1” to activate the Queued CSI.	

Caution When changing the CEnPWR bit, do not change any other bit at the same time.

While CEnPWR=0, the only registers that can be accessed are CEnCTL0, CEnTX0, CEnTXL0, and CEnSTR.

Set the CEnPWR bit before writing any of the other bits of CEnCTL0.

CEnTXE	Transmission enable/disable
0	Transmission disabled
1	Transmission enabled

CEnRXE	Receive enable/disable
0	Receive disabled
1	Receive enabled

CEnTMS	Transfer mode select
0	Single transfer mode
1	Block transfer mode

Caution Write is permitted only when CEnTXE = 0 and CEnRXE = 0.

CEnDIR	Serial data direction selection
0	Data is sent/received with MSB first
1	Data is sent/received with LSB first

Caution Write is permitted only when CEnTXE = 0 and CEnRXE = 0.

See *Section 15.3.2, Serial data direction select function* on page 332 for further details on the CEnDIR bit setting.

CEnSIT	Interrupt delay mode select (INTCEnI signal)
0	No delay
1	Half clock delay

Caution Write is permitted only when CEnTXE = 0 and CEnRXE = 0.
This bit is only valid in master mode. In slave mode, no delay is generated.

CEnWE	Transmission wait enable/disable select
0	Transmission wait disable. Not insert half clock wait at transmission start.
1	Transmission wait enable. Insert 1 clock (SCKE) wait at transmission start.

Note This bit has no effect when CEnCTL4.CEnOPE = 1.

Caution Write is permitted only when CEnTXE = 0 and CEnRXE = 0.
This bit is only valid in master mode. In slave mode, no wait is generated.

CEnCSM	Chip Select mode select
0	Chip select inactive level output disable. Do not force chip select inactive state after each transfer of a data element.
1	Chip select inactive level output enable. Hold all chip selects inactive for half-length SCKE after each transfer of a data element.

Note This bit has no effect when CEnCTL4.CEnOPE = 1.

Caution Write is permitted only when CEnTXE = 0 and CEnRXE= 0.
This bit is only valid for CEnWE=1.
This bit is only valid in master mode. In slave mode, CS signals are always held at inactive level.

In combination:

CEnWE	CEnCSM	Transmission wait	Chip Select inactive level
0	0	None	Not output
0	1	None	Not output
1	0	One SCKE length clock wait	Not output
1	1	One SCKE length clock wait	Output inactive level of half-length SCKE at first half of transmission wait

See *Section 15.3.10, Additional timing and delay selections* on page 339 for further details on the timing selections by CEnSIT, CEnWE, CEnCSM bits.

15.2.2 CEnCTL1 - Queued CSI control register 1

The CEnCTL1 register is an 8-bit register that is used to control the serial transfer operations.

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

Caution This register can be written only while CEnTXE = 0 and CEnRXE = 0.

Address <base> + 1_H

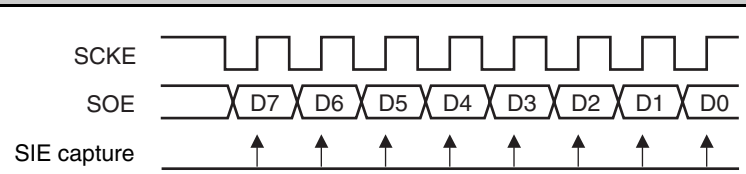
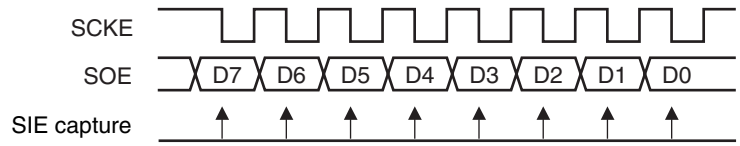
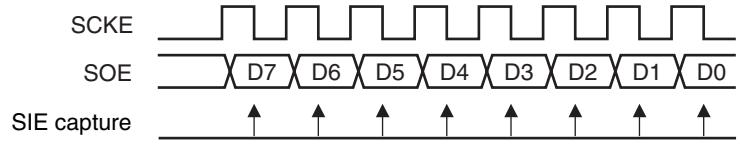
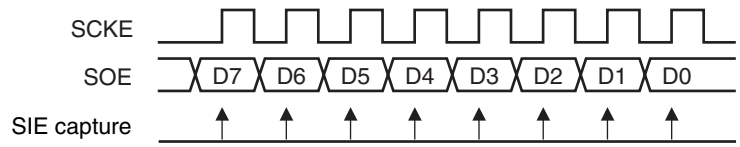
Initial Value 07_H. This register is cleared by any reset.

7	6	5	4	3	2	1	0
CEnMDL2	CEnMDL1	CEnMDL0	CEnCKP	CEnDAP	CEnCKS2	CEnCKS1	CEnCKS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

CEnMDL2	CEnMDL1	CEnMDL0	Operation clock	N	Comment
0	0	0	BRG disable	–	BRG stop for power saving
0	0	1	PRSOUT/2	1	See caution below
0	1	0	PRSOUT/4	2	
0	1	1	PRSOUT/6	3	
1	0	0	PRSOUT/8	4	
1	0	1	PRSOUT/10	5	
1	1	0	PRSOUT/12	6	
1	1	1	PRSOUT/14	7	
These bits are used to modulate the selected clock					

Caution Write is permitted only when CEnTXE = 0 and CEnRXE = 0.
CEnMDL[2:0] = [0,0,1] is prohibited when CEnCKS[2:0] = [0,0,0].

See *Section 15.3.6, Transmission clock select function* on page 334 for further explanation on the transfer clock selection.

CEnCKP	CEnDAP	Clock and Data Phase Selection
0	0	
0	1	
1	0	
1	1	

Caution Write is permitted only when CEnTXE = 0 and CEnRXE = 0.

Note CEnCKP: Clock phase selection bit
CEnDAP: Data phase selection bit

CEnCKS2	CEnCKS1	CEnCKS0	Prescaler output (PRSOUT)	Mode	K
0	0	0	CSIEnTCKI	Master Mode	0
0	0	1	CSIEnTCKI/2	Master Mode	1
0	1	0	CSIEnTCKI/4	Master Mode	2
0	1	1	CSIEnTCKI/8	Master Mode	3
1	0	0	CSIEnTCKI/16	Master Mode	4
1	0	1	CSIEnTCKI/32	Master Mode	5
1	1	0	CSIEnTCKI/64	Master Mode	6
1	1	1	SCKE (input)	Slave Mode	–

Note SCKE pin is configured as input mode for serial transfer clock.
f_{QCSI} is the clock supply of the Queued-CSI macro.

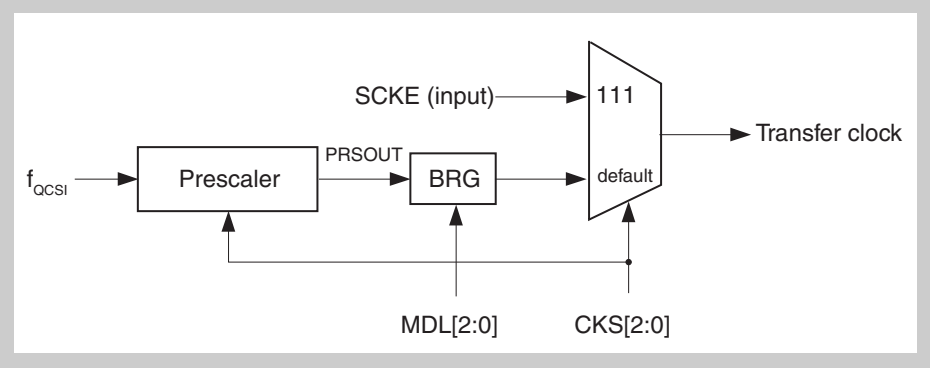
Caution Write is permitted only when CEnTXE = 0 and CEnRXE = 0.

See *Section 15.3.4, Slave mode* on page 333 and *Section 15.3.5, Master mode* on page 334 for further explanation on slave mode and master mode.

The baud rate for the transmission is calculated with the following formula:

$$\text{baudrate} = \frac{f_{QCSI}}{(2 \times N) \times 2^{K}} = \frac{f_{QCSI}}{N \times 2^{(K+1)}} = \frac{f_{QCSI}}{CEnMDL \times 2^{(CEnCKS+1)}}$$

Figure 15-2 Queued CSI Baud Rate Block Diagram



15.2.3 CEnCTL2 - Queued CSI control register 2

The CEnCTL2 register is an 8-bit register that specifies the active voltage level of the chip select pins SCSEn0 to SCSEn3 and the Queued CSI data length.

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

Address <base> + 9_H

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

Caution This register can be written only while CEnTXE = 0 and CEnRXE = 0.

7	6	5	4	3	2	1	0
CEnCSL3	CEnCSL2	CEnCSL1	CEnCSL0	CEnDL3	CEnDL2	CEnDL1	CEnDL0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

CEnCSLm	Chip Select active level selection
0	Chip Select signal SCSEnm is active low
1	Chip Select signal SCSEnm is active high

Note To select the active level of the remaining pins, see *Section 15.2.5, CEnCTL4 - Queued CSI control register 4* on page 319

Caution Write is permitted only when CEnTXE = 0 and CEnRXE = 0.

The chip select settings will be reflected at the outputs immediately after the register is set.

CEnDL3	CEnDL2	CEnDL1	CEnDL0	Transfer data length
0	0	0	0	Data length is 16 bits
0	0	0	1	Setting prohibited
0	0	1	x	
0	1	x	x	
1	0	0	0	Data length is 8 bits
1	0	0	1	Data length is 9 bits
1	0	1	0	Data length is 10 bits
1	0	1	1	Data length is 11 bits
1	1	0	0	Data length is 12 bits
1	1	0	1	Data length is 13 bits
1	1	1	0	Data length is 14 bits
1	1	1	1	Data length is 15 bits

Caution Write is permitted only when CEnTXE = 0 and CEnRXE = 0.

See *Section 15.3.3, Data length select function* on page 333 for further explanation on the data length selection.

15.2.4 CEnCTL3 - Queued CSI Control Register 3

The CEnCTL3 register is an 8-bit register that specifies the number of data elements to be transferred in block transfer mode.

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

Address <base> + C_H

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

7	6	5	4	3	2	1	0
0	0	0	0	CEnSFN3	CEnSFN2	CEnSFN1	CEnSFN0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Note Unused bits must be written as 0.

CEnSFN3	CEnSFN2	CEnSFN1	CEnSFN0	Transfer data number
0	0	0	0	Transfer 16 data elements
0	0	0	1	Transfer 1 data element
0	0	1	0	Transfer 2 data elements
0	0	1	1	Transfer 3 data elements
0	1	0	0	Transfer 4 data elements
0	1	0	1	Transfer 5 data elements
0	1	1	0	Transfer 6 data elements
0	1	1	1	Transfer 7 data elements
1	0	0	0	Transfer 8 data elements
1	0	0	1	Transfer 9 data elements
1	0	1	0	Transfer 10 data elements
1	0	1	1	Transfer 11 data elements
1	1	0	0	Transfer 12 data elements
1	1	0	1	Transfer 13 data elements
1	1	1	0	Transfer 14 data elements
1	1	1	1	Transfer 15 data elements

15.2.5 CEnCTL4 - Queued CSI control register 4

The CEnCTL4 register is an 8-bit register that specifies the active voltage level of the chip select pins SCSEn4 to SCSEn7, specifies the alternative clock and data phase setting, and controls the enhanced chip select timing.

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

Address <base> + D_H

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

Caution This register can be written only while CEnTXE = 0 and CEnRXE = 0.

7	6	5	4	3	2	1	0
CEnCSL7	CEnCSL6	CEnCSL5	CEnCSL4	CEnCPA	CEnDPA	CEnOPE	CEnMD
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

CEnCSLm	Chip Select active level selection
0	Chip Select signal SCSEnm is active low
1	Chip Select signal SCSEnm is active high

Caution Write is permitted only when CEnTXE = 0 and CEnRXE = 0.

The chip select settings will be reflected at the outputs immediately after the register is set.

CEnCPA	CEnDPA	Alternative Clock and Data phase settings
0	0	Clock and data phase as CKP=0, DAP=0
0	1	Clock and data phase as CKP=0, DAP=1
1	0	Clock and data phase as CKP=1, DAP=0
1	1	Clock and data phase as CKP=1, DAP=1

Caution Write is permitted only when CEnTXE = 0 and CEnRXE = 0.

The CPA/DPA bits define an alternative setting for the serial communication within the same Queued-CSI macro. Each chip select can be assigned to either the standard (CKP/DAP) or alternative (CPA/DPA) phase setting when the enhanced timing is active (CEnOPE=1). The CPA/DPA bits have no effect when CEnOPE=0.

Please refer to the description *Section 15.2.2, CEnCTL1 - Queued CSI control register 1* on page 314 for details on the timings for each clock and data phase.

CEnOPE	Enhanced timing enable/disable
0	Enhanced timing is disabled
1	Enhanced timing is enabled.

Caution Write is permitted only when CEnTXE = 0 and CEnRXE = 0.

When the enhanced timing is enabled (CEnOPE=1), the setup time, inter-data time, hold time and idle time for each chip select signal can be defined using the CEnOPTm registers. For CEnOPE=0, the standard timing applies to all chip select signals.

CEnMD	Chip select mode
0	4 chip select mode or special 8 chip select mode. • 4 chip select mode: - Setting this bit to 0 in combination to writing 0 to CSGLCTLn.CSGLCTLn0 sets the 4 chip select mode. - Each of the 4 bits of the CEnCS register (CEnCS[3:0]) represents a chip select line. When the bit is set to 1, the chip select line is active, when set to 0, it is inactive. • Special 8 chip select mode: Setting this bit to 0 in combination to writing 1 to CSGLCTLn.CSGLCTLn0 sets the special 8 chip select mode.
1	8 chip select mode. The combination of the lower 3 bits of the CEnCS line (CEnCS[2:0]) select one out of 8 available chip select lines.

Cautions
1. Write is permitted only when CEnTXE = 0 and CEnRXE = 0.
2. The setting CSGLCTLn.CSGLCTLn0 = 1 and CEnCTL4.CEnMD = 1 is forbidden.

Note Rx3 devices are equipped with additional logic to support a new chip select mode. Therefore the chip select mode also depends on the CSGLCTLn register.

4 chip select mode In 4 chip select mode, only 4 chip selects can be used for the communication, but multiple chip selects can be active for the same communication (“broadcast”), as each bit represents a chip select signal.

8 chip select mode In 8 chip select mode, all 8 chip selects are accessible, although only one out of the 8 chip selects can be selected as active for a communication.

Special 8 chip select mode In special 8 chip select mode, two out of the 8 chip selects can be active simultaneously: CS0 and one other, CSX, where X = 1 to 7.

Caution For macros offering only 4 chip selects as output pins, the setting of 8 chip select mode is prohibited.

15.2.6 CSGLCTLn - Chip Select Control Register

The CSGLCTLn register is an 8-bit register that allows the configuration of this extra mode for the Chip Select signals.

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

Address The addresses of the CSGLCTLn registers are listed in the table below.

Table 15-6 Addresses of the CSGLCTLn registers

CSGLCTLn	<base> address
CSGLCTL0	FFFF FD30$_H$
CSGLCTL1	FFFF FD34$_H$

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

Caution This register must be configured prior to using the CSIEn macro.

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	CSGLCTLn 0
R	R	R	R	R	R	R	R/W

CSGLCTLn0	Enhanced timing enable/disable
0	Special 8 Chip Select mode not activated
1	Special 8 Chip Select mode activated

Cautions

1. The setting CSGLCTLn0 = 1 and CEnCTL4.CEnMD = 1 is forbidden.
2. In special 8 chip select mode, the following cautions apply:
   - all chip select signals must be configured active low(CExCTL2 and CExCTL4 bits 7~4 = 0).
   - when not using the enhanced timing option, (CEnCTL4.CEnOPE=0), always insert 1 clock (SCKE) wait a transmission start with the wait enable bit (CExCTL0.CEnWE = 1).
   - when using the enhanced timing option (CEnCTL4.CEnOPE=1), always program at least 0.5 (SCKE) setup delay for each chip select (CEnOPTm:CEnSPm[2:0] > 000_b).
   - all chip selects must be programmed with the same timing.

15.2.7 CEnRX0 - Receive data buffer register

The CEnRX0 register is a 16-bit register or separated as upper 8 bits (CEnRXH0) and lower 8 bits (CEnRXL0), that is used to store receive data.

Access This register can be read/written in 8-bit or 16-bit units.

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
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

Caution The receive data buffer register is considered as emptied by the application software whenever the lower 8 bits of the register are read. It is therefore necessary to read CEnRXH0 before CEnRXL0 when 8-bit access is used.

15.2.8 CEnCS - Chip select data buffer register

The CEnCS register is a 16-bit register than can be also accessed as 8 bit register (CEnCSL) that stores Chip Select data.

Access This register can be read/written in 8-bit or 16-bit units.

Address <base> + 4_H

Initial Value $FFFF_H$. This register is cleared by any reset.

Following the FIFO write pointer, the value written to CEnCS is stored in the FIFO data buffer as Chip Select bits. The value is stored to these bits when the transmit data is written to its register (CEnTX0 or CEnTXL0).

CEnCS write is prohibited when CEnPWR = 1 and f_{QCSI} is stopped.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-	-	-	-	-	CEnCS3	CEnCS2	CEnCS1	CEnCS0
R	R	R	R	R	R	R	R	R	R	R	R	R/W	R/W	R/W	R/W

Cautions
1. The Chip Select register is stored in the FIFO buffer when the transmit data is written to its register. It is therefore necessary to write the CEnCS register before the CEnTX0 (CEnTXL0) register.
2. When the enhanced timing mode is enabled by setting CEn0PE = 1, setting of chip select CEnCS3-CEnCS0 = “1111” is prohibited.

Chip Select Modes Depending on the chip select mode (defined by the CEnMD and CSGLCTLn0 bits - see *Section 15.2.5, CEnCTL4 - Queued CSI control register 4* on page 319 and *Section 15.2.6, CSGLCTLn - Chip Select Control Register* on page 321 for details) the CEnCS[3:0] bits have a different meaning.

The table below summarizes the different possible Chip Select configurations.

CSGLCTLn0	CEnMD	Chip Select Mode
0	0	4 chip select mode
0	1	8 chip select mode
1	0	Special 8 chip select mode
1	1	Setting forbidden

(a) 4 chip select mode (CEnMD = 0 ad CSGLCTLn0 = 0)

CEnCS bits				Chip select signals			
CEnCS3	CEnCS2	CEnCS1	CEnCS0	SCSEn3	SCSEn2	SCSEn1	SCSEn0
0	0	0	0	selected	selected	selected	selected
0	0	0	1	selected	selected	selected	-
0	0	1	0	selected	selected	-	selected
0	0	1	1	selected	selected	-	-
0	1	0	0	selected	-	selected	selected
0	1	0	1	selected	-	selected	-
0	1	1	0	selected	-	-	selected
0	1	1	1	selected	-	-	-
1	0	0	0	-	selected	selected	selected
1	0	0	1	-	selected	selected	-
1	0	1	0	-	selected	-	selected
1	0	1	1	-	selected	-	-
1	1	0	0	-	-	selected	selected
1	1	0	1	-	-	selected	-
1	1	1	0	-	-	-	selected
1	1	1	1	-	-	-	-

(b) 8 chip select mode (CEnMD = 1 and CSGLCTLn0 = 0)

CEnCS3	CEnCS2	CEnCS1	CEnCS0	Active chip select signal
0	0	0	0	SCSEn0 selected
0	0	0	1	SCSEn1 selected
0	0	1	0	SCSEn2 selected
0	0	1	1	SCSEn3 selected
0	1	0	0	SCSEn4 selected
0	1	0	1	SCSEn5 selected
0	1	1	0	SCSEn6 selected
0	1	1	1	SCSEn7 selected
1	0	0	0	SCSEn0 selected
1	0	0	1	SCSEn1 selected
1	0	1	0	SCSEn1 selected
1	0	1	1	SCSEn1 selected
1	1	0	0	SCSEn1 selected

1	1	0	1	SCSEn1 selected
1	1	1	0	SCSEn1 selected
1	1	1	1	SCSEn7 selected

(c) Special 8 chip select mode (CEnMD = 1 and CSGLCTLn0 = 1)

CEnCS3	CEnCS2	CEnCS1	CEnCS0	Active chip select signal
0	0	0	0	SCSEn0 selected
0	0	0	1	SCSEn1 selected
0	0	1	0	SCSEn2 selected
0	0	1	1	SCSEn3 selected
0	1	0	0	SCSEn4 selected
0	1	0	1	SCSEn5 selected
0	1	1	0	SCSEn6 selected
0	1	1	1	SCSEn7 selected
1	0	0	0	SCSEn0 and SCEn7 selected
1	0	0	1	SCSEn0 and SCEn1 selected
1	0	1	0	SCSEn0 and SCEn2 selected
1	0	1	1	SCSEn0 and SCEn3 selected
1	1	0	0	SCSEn0 and SCEn4 selected
1	1	0	1	SCSEn0 and SCEn5 selected
1	1	1	0	SCSEn0 and SCEn6 selected
1	1	1	1	No chip SCSEnm selected

Note In 4 chip select mode and 8 chip select mode, the active level for each chip select is defined in the CEnCTL2 and CEnCTL4 register (CEnCSL[7:0] bits).

Caution When the enhanced timing mode is enabled by setting CEn0PE = 1, setting of chip select CEnCS3-CEnCS0 = “1111” is prohibited.

15.2.9 CEnTX0 - Transmission data buffer registers

The CEnTX0 register is a 16-bit buffer register, or separated as upper 8 bits (CEnTXH0) and lower 8 bits (CEnTXL0), that stores transmission data.

Access This register can be read/written in 8-bit or 16-bit units.

Address <base> + 6_H

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

Incrementing and following the FIFO buffer write pointer, the value written to CEnTX0 is stored to the FIFO buffer as transmission data.

CEnTX0 write is prohibited when CEnPWR = 1 and f_{QCSI} is stopped.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	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/W

Caution The transmit data buffer register is considered as written whenever the lower 8 bits of the register are written. It is therefore necessary to write CEnTXH0 before CEnTXL0 when 8-bit access is used.

15.2.10 CEnSTR - Queued-CSI status registers

The CEnSTR register is an 8-bit register that shows the status of the Queued CSI.

Access This register can be read/written in 8-bit or 1-bit units.
The bits CEnFLF, CEnEMF and CEnTSF are read only.

CEnSTR read is prohibited when CEnPWR = 1 and f_{QCSI} is stopped.

Address <base> + 8_H

Initial Value 20_H. This register is cleared by any reset.

7	6	5	4	3	2	1	0
CEnPCT	CEnFLF	CEnEMF	CEnTSF	CEnSFP3	CEnSFP2	CEnSFP1	CEnSFP0
R/W	R	R	R	R	R	R	R

CEnPCT	FIFO buffer pointer clear command
0	No operation
1	Clear all FIFO pointers to "0"

Note Read value is always “0”.

CEnFLF	FIFO buffer full status flag
0	FIFO buffer is not full
1	FIFO buffer is full

CEnEMF	FIFO buffer empty status flag
0	FIFO buffer is not empty
1	FIFO buffer is empty

CEnTSF	Transmission status flag
0	Idle state
1	Transmission in on going or preparing

Note 1. This bit is cleared to “0” by CEnPWR = 0 or (CEnTXE = 0 and CEnRXE = 0).

2. In single transfer mode, this bit holds “1” from transmission start to FIFO empty.

3. In block transfer mode, this bit holds “1” from transmission start until finish transferring all data to be sent.

CEnSFP3 -0	Transmission data count
0 - 15	In Single transfer mode, CEnSFP[3:0] indicates the number of remaining transfers in the FIFO. This value can be understood as (Write FIFO pointer) - (SIO load pointer)
	In FIFO transfer mode, CEnSFP[3:0] indicates the number of data transfers completed. In case of CEnSFP[3:0] = 0_H: • CEnEMF = 0 and CEnSFP[3:0] = 0_H: Number of receptions completed = 0 • CEnEMF = 1 and CEnSFP[3:0] = 0_H: Number of receptions completed = 16

Note CEnSFP[3:0] holds its value until RESET or CEnPCT = 1.

Caution CEnFLF, CEnEMF, CEnTSF and CEnSFP[3:0] are continuously updated with the current status of the Queued CSI. This means that a value read might be outdated shortly after the read was executed.
When writing accidentally a 17th data element in FIFO, an overflow interrupt (INTCEnO) will occur to indicate the error.

15.2.11 CEnOPTm - Queued CSI enhanced timing registers

The CEnOPTm registers are 16-bit register, that are used to define the timing for the chip select signal CEnm when enhanced timing is active (CEnOPE = 1).

Access This register can be read/written in 16-bit units.

Address
CEnOPT0 : <base> + 10_H
CEnOPT1 : <base> + 12_H
CEnOPT2 : <base> + 14_H
CEnOPT3 : <base> + 16_H
CEnOPT4 : <base> + 18_H
CEnOPT5 : <base> + $1A_H$
CEnOPT6 : <base> + $1C_H$
CEnOPT7 : <base> + $1E_H$

Initial Value 0002_H. This register is cleared by any reset.

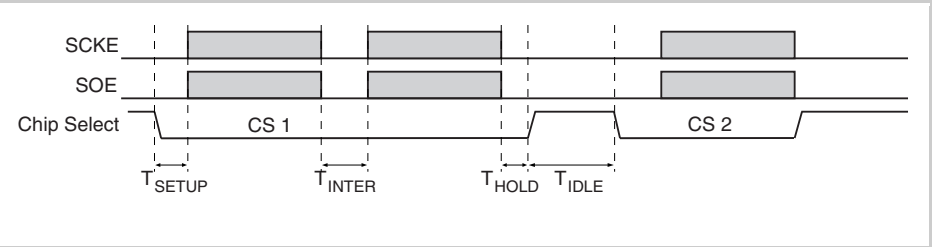
Caution These register can be written only while CEnTXE = 0 and CEnRXE = 0.

15	14	13	12	11	10	9	8
CEnPHSm	CEnIDLm	CEnIDm2	CEnIDm1	CEnIDm0	CEnHDm2	CEnHDm1	CEnHDm0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

7	6	5	4	3	2	1	0
CEnDLm1	CEnDLm0	CEnINm2	CEnINm1	CEnINm0	CEnSPm2	CEnSPm1	CEnSPm0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

The enhanced timing allows insertion of delays in the chip select timing:

Figure 15-3 General overview on enhanced chip select timing



CEnPHSm	Clock/data phase selection
0	Clock/data phase selection defined by CKP/DAP are used for chip select m
1	Alternative clock/data phase selection defined by CKA/DAP are used for chip select m

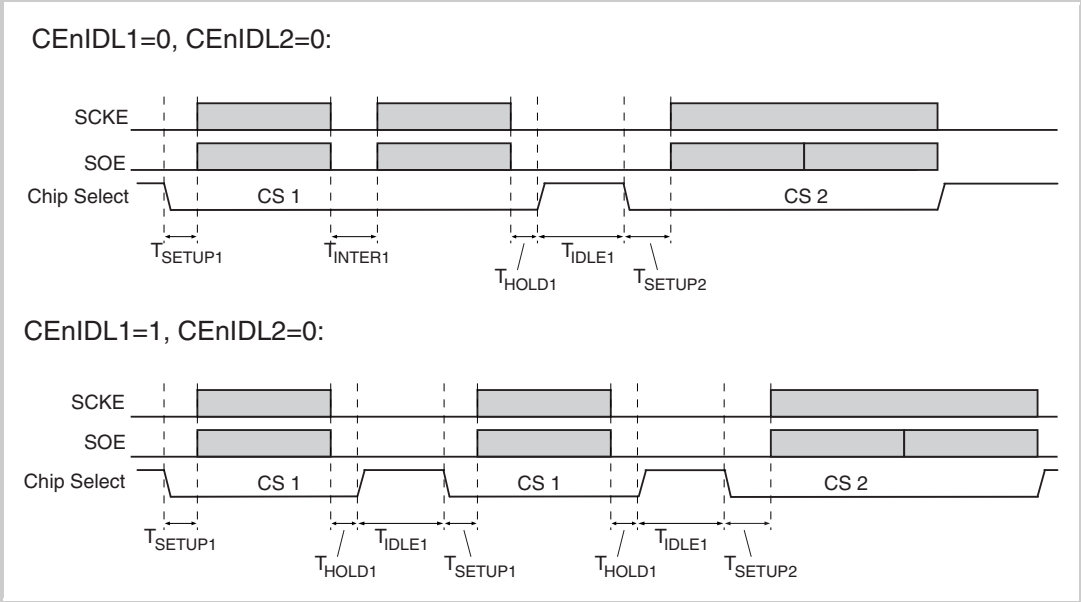
Caution Write is permitted only when CEnTXE = 0 and CEnRXE = 0.

CEnIDLm	Chip select idle enforcement
0	Chip select m becomes idle whenever a different chip select value is defined the CEnCS[3:0] bits. If the chip select value remains the same or the buffer becomes empty, the chip select stays active.
1	An idle state is always inserted after each transfer.

Caution Write is permitted only when CEnTXE = 0 and CEnRXE = 0.

When the multiple data is transferred using the same chip select setting, the chip select line remains active. The CEnIDLm bit forced the chip select line to become idle after each transfer, even if the same chip select setting is used for the next transfer.

Figure 15-4 Enforced chip select idle communication



CEnIDm2	CEnIDm1	CEnIDm0	Chip select m idle timing [Note]
0	0	0	chip select m is idle for 0 SCKE
0	0	1	chip select m is idle for 0.5 SCKE
0	1	0	chip select m is idle for 1 SCKE
0	1	1	chip select m is idle for 2 SCKE
1	0	0	chip select m is idle for 3 SCKE
1	0	1	chip select m is idle for 4 SCKE
1	1	0	chip select m is idle for 6 SCKE
1	1	1	chip select m is idle for 8 SCKE

Caution Write is permitted only when CEnTXE = 0 and CEnRXE = 0.

Note A minimum idle period of around 2-3 macro clock cycles (f_{QCSI}) is applied whenever a CS is set from active to inactive. This time is an addition to the time set by the CEnIDm[2:0] bits.

CEnHDm2	CEnHDm1	CEnHDm0	Chip select m hold timing	
			CEnSIT=0	CEnSIT=1
0	0	0	No hold time for chip select m	hold chip select m for 0.5 SCKE
0	0	1	hold chip select m for 0.5 SCKE	hold chip select m for 1 SCKE
0	1	0	hold chip select m for 1 SCKE	hold chip select m for 1.5 SCKE
0	1	1	hold chip select m for 2 SCKE	hold chip select m for 2.5 SCKE
1	0	0	hold chip select m for 3 SCKE	hold chip select m for 3.5 SCKE
1	0	1	hold chip select m for 4 SCKE	hold chip select m for 4.5 SCKE
1	1	0	hold chip select m for 6 SCKE	hold chip select m for 6.5 SCKE
1	1	1	hold chip select m for 8 SCKE	hold chip select m for 8.5 SCKE

Caution Write is permitted only when CEnTXE = 0 and CEnRXE = 0.

Note The hold times given above assume that a new transfer element is waiting in the transfer buffer. When the transfer queue becomes empty, the hold time will be longer than the times given above.

CEnDLm1	CEnDLm0	Chip select m data length selection
0	0	The data length defined by CEnDL[3:0] is used as data length chip select m
0	1	The data length defined by CEnDL[3:0] is used as data length chip select m
1	0	The data length for chip select m is 8 bits (regardless of CEnDL[3:0] setting)
1	1	The data length for chip select m is 16 bits (regardless of CEnDL[3:0] setting)

Caution Write is permitted only when CEnTXE = 0 and CEnRXE = 0.

CEnINm2	CEnINm1	CEnINm0	Chip select m inter-data delay timing	
			CEnSIT=0	CEnSIT=1
0	0	0	No inter-data delay	Inter-data delay is 0.5 SCKE
0	0	1	Inter-data delay is 0.5 SCKE	Inter-data delay is 0.5 SCKE
0	1	0	Inter-data delay is 1 SCKE	Inter-data delay is 1 SCKE
0	1	1	Inter-data delay is 2 SCKE	Inter-data delay is 2 SCKE
1	0	0	Inter-data delay is 3 SCKE	Inter-data delay is 3 SCKE
1	0	1	Inter-data delay is 4 SCKE	Inter-data delay is 4 SCKE
1	1	0	Inter-data delay is 6 SCKE	Inter-data delay is 6 SCKE
1	1	1	Inter-data delay is 8 SCKE	Inter-data delay is 8 SCKE

Caution Write is permitted only when CEnTXE = 0 and CEnRXE = 0.

Note 1. Inter-data delay only applies when no idle state is enforced between data (CEnIDLm=0).

2. The inter-data times given above assume that a new transfer element is waiting in the transfer buffer. When the transfer queue becomes empty, the hold time will be longer than the times given above.

CEnSPm2	CEnSPm1	CEnSPm0	Chip select m setup timing
0	0	0	No setup delay
0	0	1	Setup delay is 0.5 SCKE
0	1	0	Setup delay is 1 SCKE
0	1	1	Setup delay is 2 SCKE
1	0	0	Setup delay is 3 SCKE
1	0	1	Setup delay is 4 SCKE
1	1	0	Setup delay is 6 SCKE
1	1	1	Setup delay is 8 SCKE

15.3 Explanation of Queued CSI Functions

15.3.1 Transmit buffer

Chip select data and transmission data can be stored to the transmit FIFO buffer continuously by writing to the CEnCS register and CEnTX0 register. The Writing FIFO pointer is automatically incremented when data is written to CEnTX0. The size of the transmit FIFO buffer is 20 bits × 16 entries.

In slave mode, it is not necessary to set the chip select data.

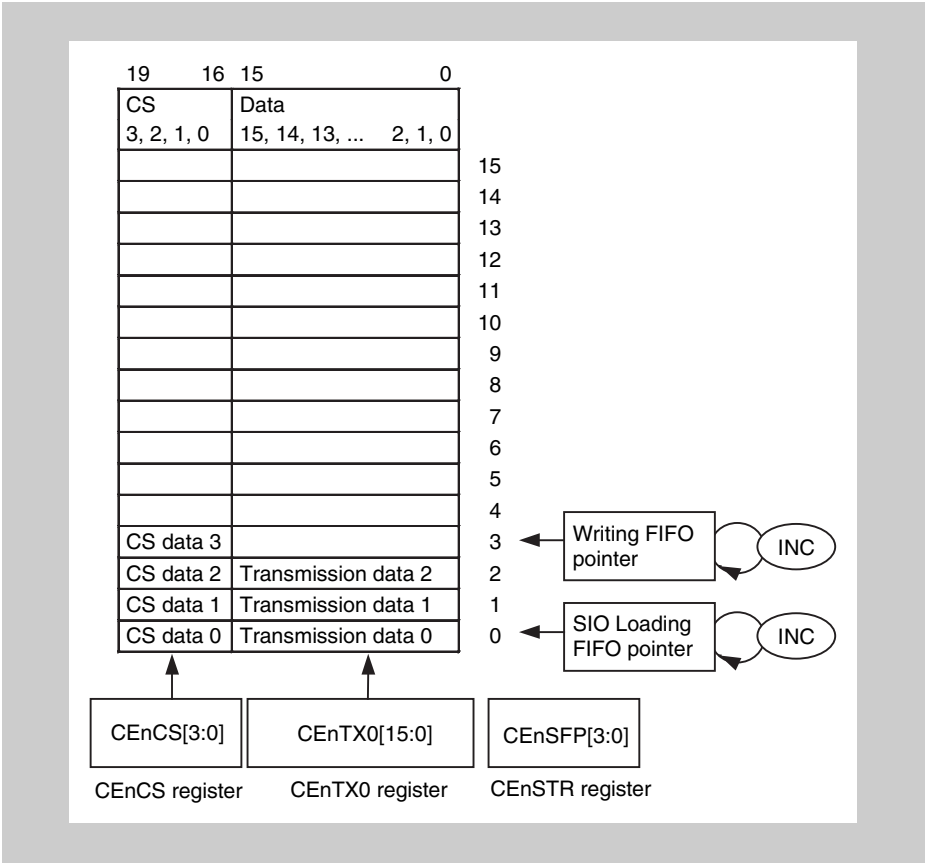
The transfer start condition (assuming that CEnEMF = 0) is to write to the lower bits of CEnTX0 register. If the transmission data length is 9 bits or more, data should be written by a 16-bit write to CEnTX0 or by two 8-bit writes with first CEnTXH0, then CEnTXL0 (in that order). When transmission data length is 8 bits, data should be written by one 8-bit write to CEnTXL0 or by one 16-bit write to CEnTX0. For the 16-bit write, the upper 8 bits are ignored in the 8-bit transmission.

The CEnFLF bit in the CEnSTR status register is set to “1” after 16 writes have been made to the transmit buffer, assuming the Writing FIFO pointer was reset previously.

When a transmission write is attempted while the FIFO is full (CEnFLF=1), the interrupt INTCEnO is generated to indicate an overflow. In that case, the transmission and chip select data are discarded and not stored.

When a transfer cycle is finished and the SIO Loading FIFO pointer is incremented, the FIFO buffer of the previous location is considered empty in case of single transfer mode. Refer to *Section 15.3.8, Description of the block transfer mode* on page 337 for more information on block transfer mode.

Figure 15-5 Transmit buffer



15.3.2 Serial data direction select function

The serial data direction is selectable using the CEnDIR bit in the CEnCTL0 register. The examples below show the communication for data length of 8 bit (CEnDL[3:0] = [1,0,0,0]):

Figure 15-6 Serial data direction select function - MSB first (CEnDIR = 0)

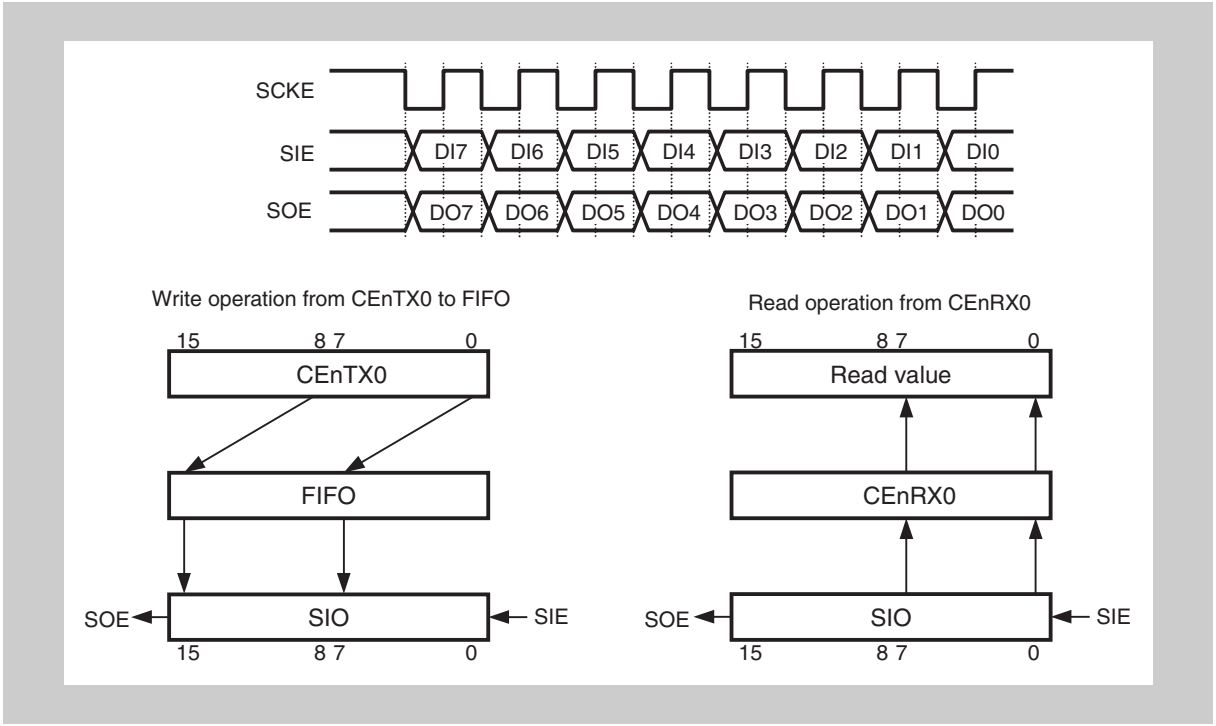
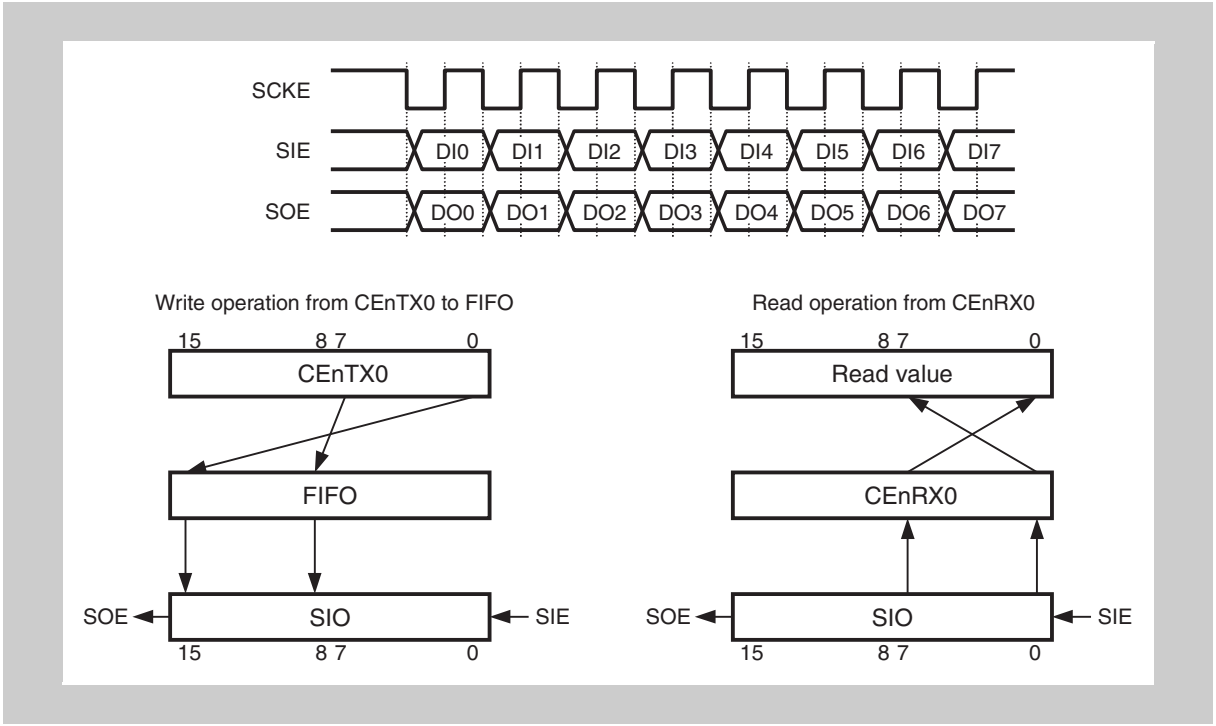


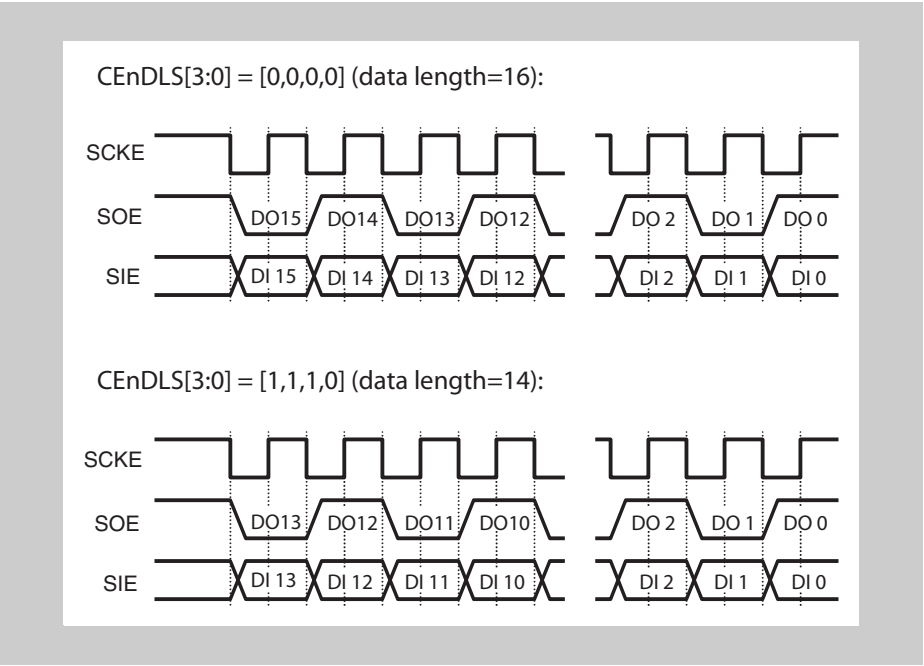
Figure 15-7 Serial data direction select function - LSB first (CEnDIR = 0)



15.3.3 Data length select function

Transmission data length is selectable from 8 bits to 16 bits using the CEnDL[3:0] bits in CEnCTL2 register. The examples below show the communication with MSB first (CEnDIR = 1):

Figure 15-8 Data length select function

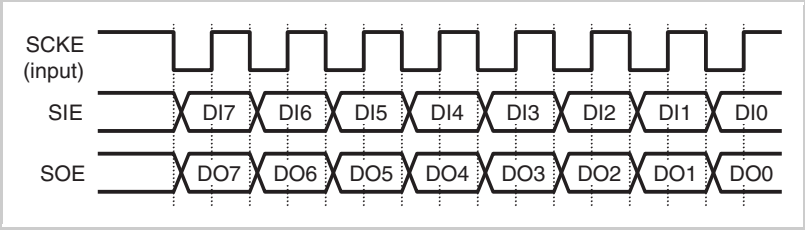


15.3.4 Slave mode

When the CEnCKS[2:0] bits in CEnCTL1 are set to [1,1,1], the Queued CSI operates in slave mode. In slave mode, the SCKE serial clock pin becomes input and another device is the CSI communication master. The baud rate generator “BRG” is recommended to be disabled by setting bits CEnMDL[2:0] to [0,0,0] when using slave mode. Also, the chip select pin SCSEn[3:0] outputs are not available in slave mode, as they are only available in master mode.

The example below shows the communication in slave mode for 8 data bits, CEnCKP=0, CEnDAP=0 and MSB first:

Figure 15-9 Slave mode

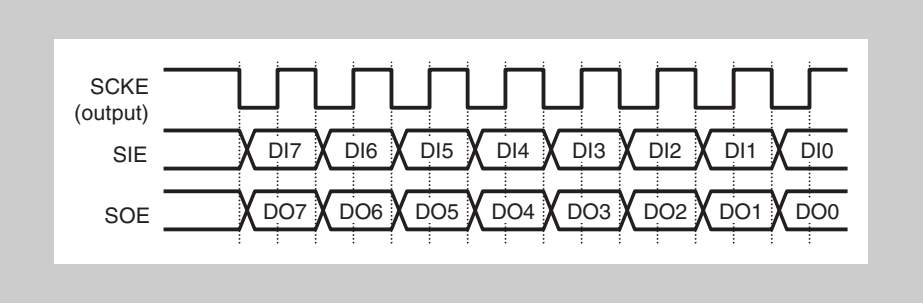


15.3.5 Master mode

When the CEnCKS[2:0] bits in CEnCTL1 are not set to [1,1,1], the Queued CSI operates in master mode. In master mode, the SCKE pin is configured as output and the serial communication clock is generated by the Queued CSI module. The SCKE pin's default value is "1" when CEnCKP = 1, and is default "0" when CEnCKP = 0. The SCSEn[3:0] pin outputs are available in master mode.

The example below shows the communication in master mode for 8 data bits, CEnCKP=0, CEnDAP=0 and MSB first:

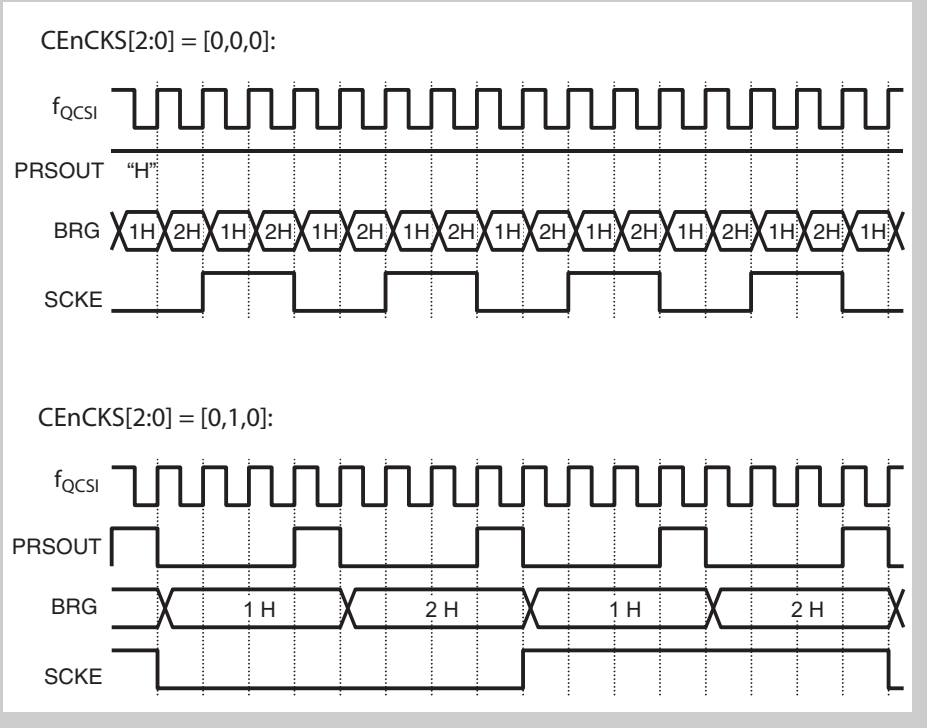
Figure 15-10 Master mode



15.3.6 Transmission clock select function

In Master Mode, the transfer baud rate is selectable using CEnCKS[2:0] bits and CEnMDL[2:0] bits in CEnCTL1 register. The baud rate generator "BRG" counts up at each rising edge of f_{QCSI}. The example below illustrates the baud rate generation for CEnMDL[2:0] = [0,1,0].

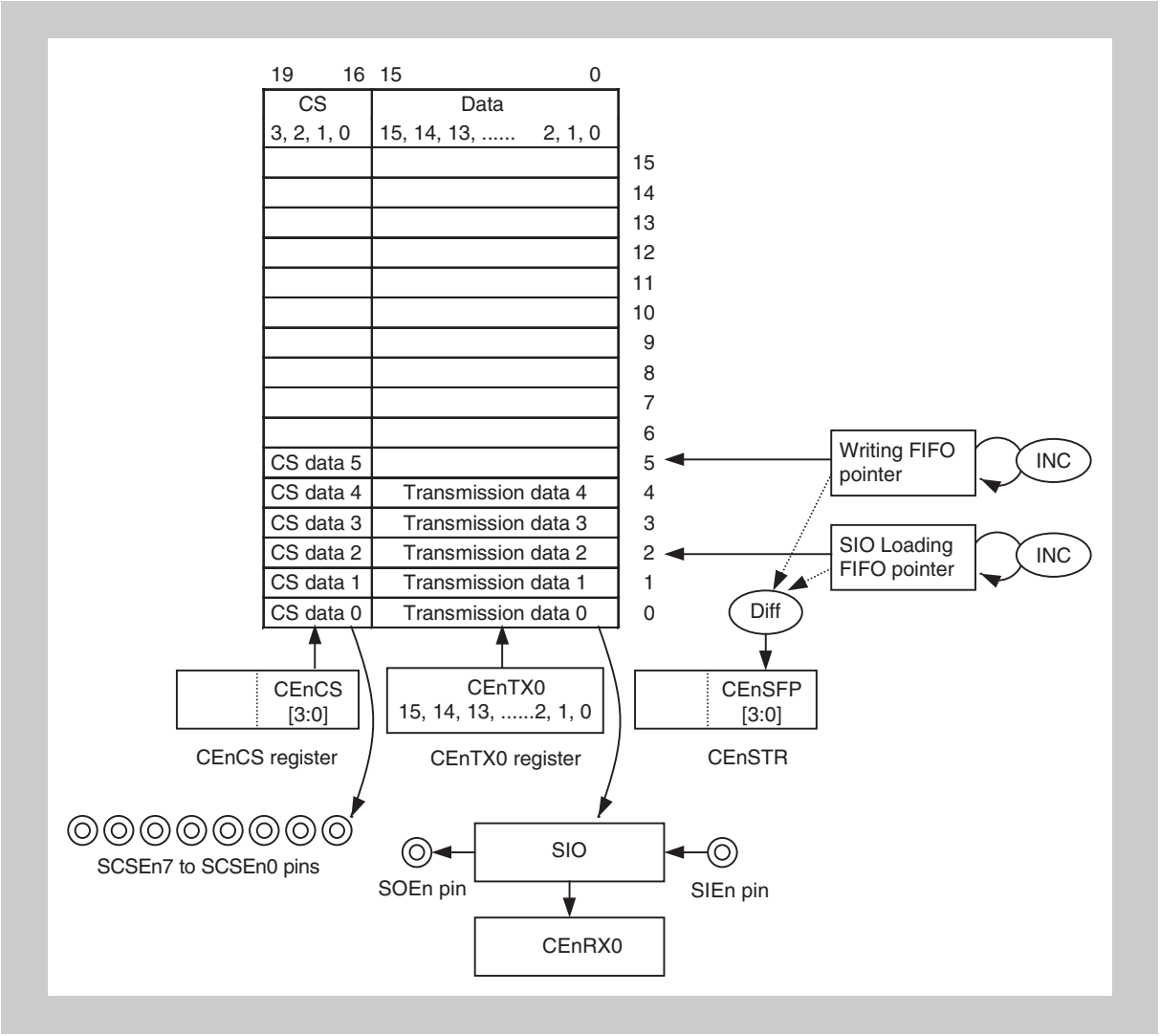
Figure 15-11 Transfer clock select function



15.3.7 Description of the single transfer mode

When the CEnTMS bit in the CEnCTL0 register is set to “0”, the Queued CSI operates in single transfer mode.

Figure 15-12 Single transfer mode data handling



Transfer start condition in single transfer mode:

[CEnTXE = 1 or CEnRXE = 1] and
[Data exists in FIFO (CEnEMF = 0)]

A transfer starts once the transmission data (pointed to by the SIO Loading FIFO pointer) is transferred from the FIFO buffer to the serial shift register SIO. At that time, the transfer status flag CEnTSF turns to “1”. The SCSEn[3:0] pins output the chip select data from the FIFO buffer.

At the end of the transfer:

1. If CEnRX0 is empty, the received data is stored from SIO to CEnRX0, and transfer end interrupt signal INTCEnI is generated (in transmit only mode, INTCEnI will be generated only when the FIFO buffer becomes empty). Finally, the SIO Loading FIFO pointer is incremented.
3. If CEnRX0 is not empty, the storing of receive data, INTCEnI generation and SIO Loading FIFO pointer incrementing wait for the CEnRX0 to be emptied by a software read operation.

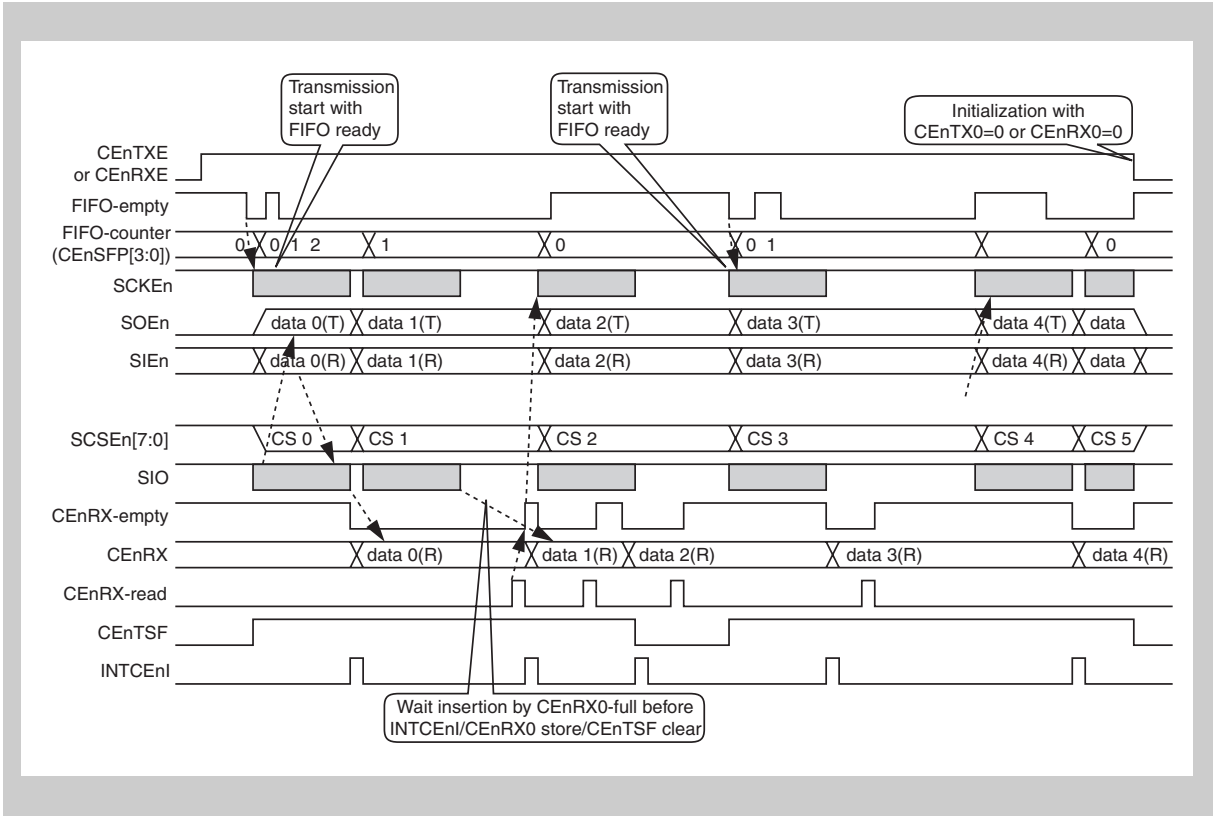
4. In transmit only mode, if transmission data is available in the FIFO buffer, the next transfer will start immediately, regardless of the CEnRX0 buffer condition.

When a transfer finishes and the FIFO buffer is empty (Writing FIFO pointer = SIO Loading FIFO pointer), CEnTSF is cleared “0”.

CEnSFP[3:0] always show the current value of: (Writing FIFO pointer) - (SIO Loading FIFO pointer).

It is recommended to check that CEnFLF = 0 just before data is written to the CEnTX0 register. If CEnFLF = 1 when a write to CEnTX0 is attempted, the overflow interrupt INTCEnO is generated and the written data is ignored.

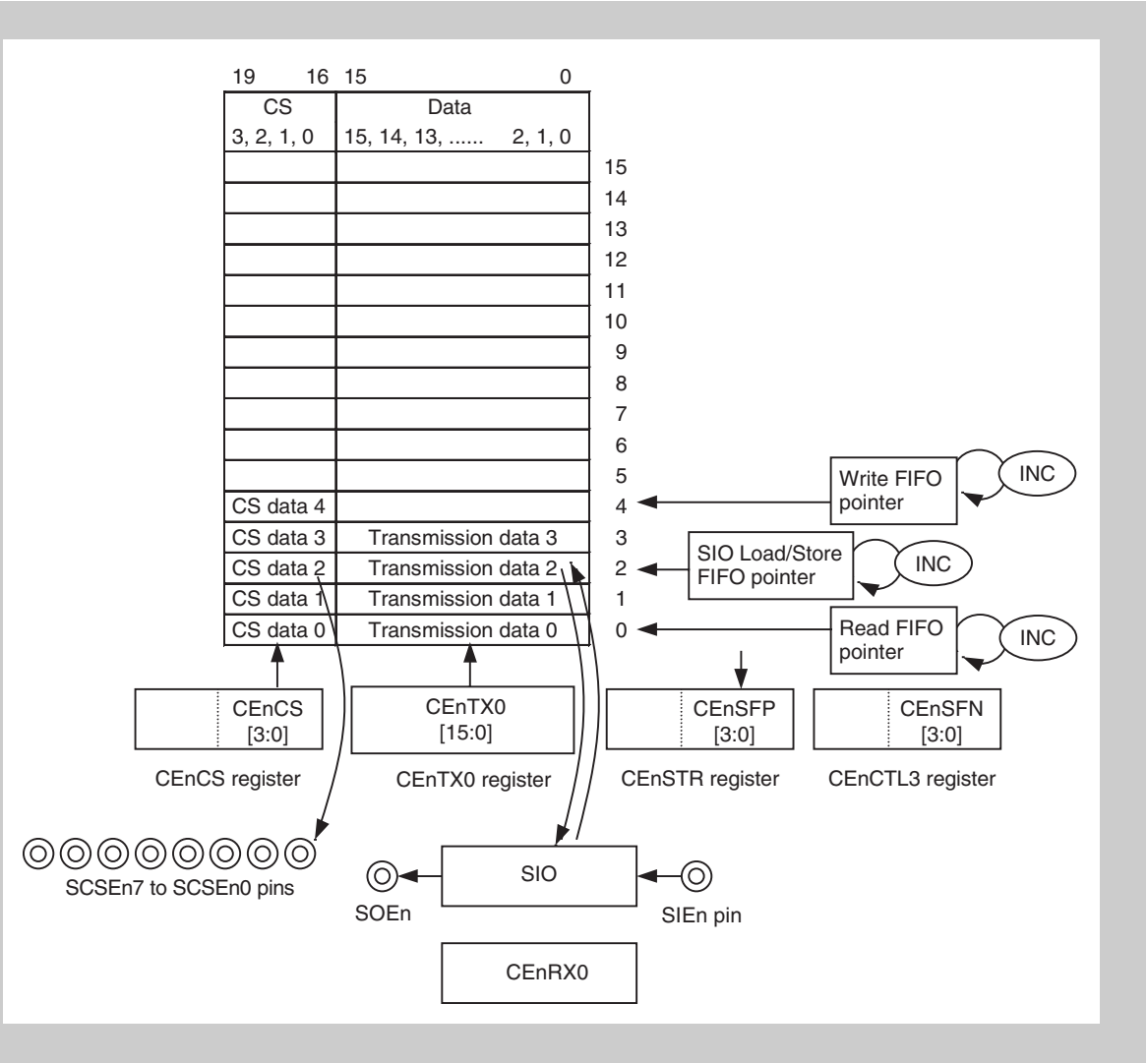
Figure 15-13 Single transfer mode (master, transmit/receive) timing



15.3.8 Description of the block transfer mode

When the CEnTMS bit in the CEnCTL0 register is set to “1”, the Queued CSI operates in block transfer mode.

Figure 15-14 Block transfer mode data handling



Transfer start condition in block transfer mode:

[CEnTXE = 1 or CEnRXE = 1] and

[Data exists in FIFO (CEnEMF = 0)]

The transmission data number must be set in CEnSFN[3:0]. Note that writing a value greater than 16 to the CEnCTL3 register is prohibited, as the FIFO buffer design can hold up to 16 elements only.

The transfer starts by copying the first data element - pointed to by SIO Load/Store FIFO pointer - to the SIO shift register. At that time the transmission status flag CEnTSF is set to “1”, and the SCSEn[3:0] pins output the CS value from the FIFO.

When the transfer of the data element is finished, the received data overwrites the location in the FIFO using the SIO Load/Store FIFO pointer, and the SIO Load/Store FIFO pointer is then incremented.

When the transmission/reception counter reaches the value set by CEnSFN[3:0], then CEnTSF is cleared to “0” and the transmission/reception end interrupt signal INTCEnI is generated.

After the interrupt occurs, the received data can be read from CEnRX0. The Read FIFO pointer is automatically incremented by the CEnRX0 read operation.

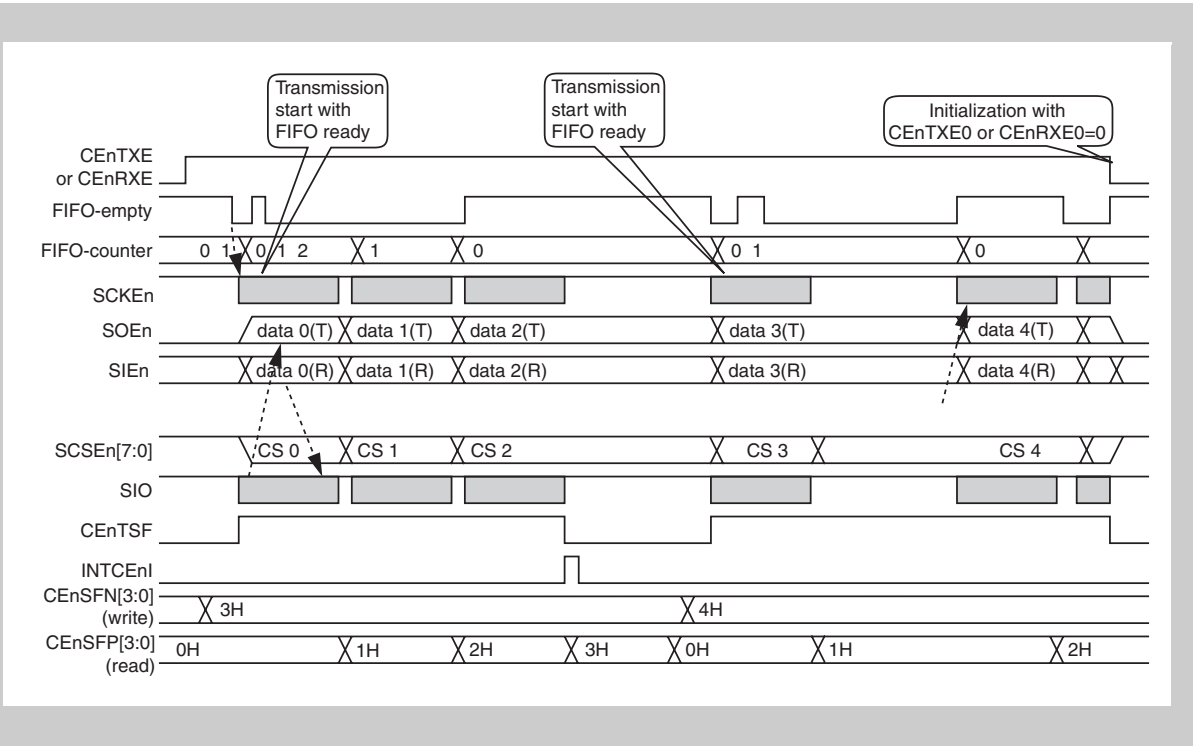
All FIFO pointers must be cleared by setting CEnPCT = 1 before the next transmit/receive cycle can start.

SFP[3:0] represents the [SIO Load/Store FIFO pointer] and shows the number of transmission/receptions completed. In case of SFP[3:0]=0_H, the numbers of transmissions/receptions depends on the setting of the SFEMP bit:

SFEMP=0: 0 transmissions/receptions completed

SFEMP=1: 16 transmissions/receptions completed

Figure 15-15 Block transfer mode (master, transmit/receive) timing



15.3.9 Description of the operation modes

(1) Transmit only mode

Setting the CEnCTL0 register's CEnTXE =1 and CEnRXE = 0 places the Queued CSI in transmit only mode. A transmission starts when transmit data is written in the CEnTX0 register. The current condition of the CEnRX0 buffer and SIO register has no effect. The data in the CEnRX0 and the SIO buffer is undefined after completion of the transmission.

(2) Receive only mode

Setting the CEnCTL0 register's CEnTXE = 0 and CEnRXE = 1 places the Queued CSI in receive only mode. A reception starts when dummy data is written in the CEnTX0 register. It is mandatory, though, that the CEnRX0 and SIO are empty. If a receive operation is terminated while the previous receive data remains unread in CEnRX0, the Queued CSI is placed on wait status until the previous data is completely read and CEnRX0 becomes empty.

(3) Transmit/receive mode

Setting the CEnCTL0 register's CEnTXE = 1 and CEnRXE = 1 places the Queued CSI in transmit/receive mode. A transfer (meaning transmission and reception) starts when transmit data is written in the CEnTX0 register. Note that an empty CEnRX0 or SIO is mandatory. If a receive operation is terminated while the previous receive data remains unread in CEnRX0, the Queued CSI is placed on wait status until the previous data is completely read and CEnRX0 becomes empty.

In block transfer mode with slave mode, only the first dummy data write operation is required. There is no need to write chip-select data, as these bits are ignored.

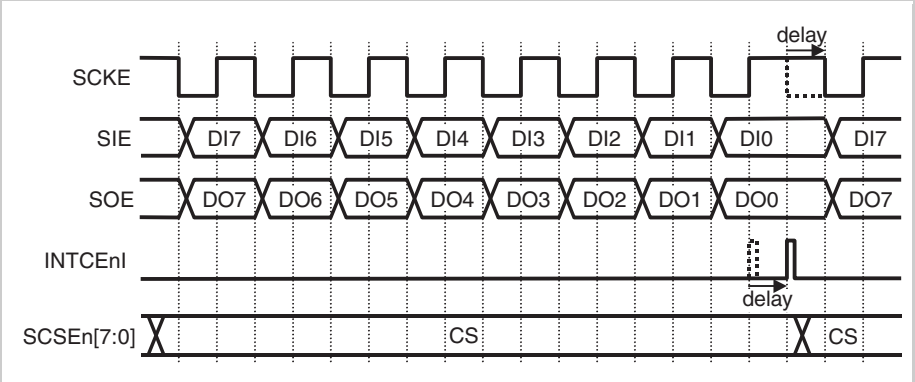
15.3.10 Additional timing and delay selections

(1) Delay selection of receive termination interrupt signal (INTCEnI)

In master mode, the CEnSIT bit of the CEnCTL0 register can be used to delay the generation of the receive termination interrupt signal (INTCEnI) by a half serial clock cycle (SCKE). The CEnSIT bit takes effect only in the master mode and is ignored in slave mode.

Figure 15-16 below illustrates the CEnSIT function, assuming a setting of CEnSIT=1, CEnWE=0, CEnCKP=0, CEnDAP=0 and CEnDL[3:0] = [1,0,0,0].

Figure 15-16 Delay selection of receive termination interrupt (INTCEnI)

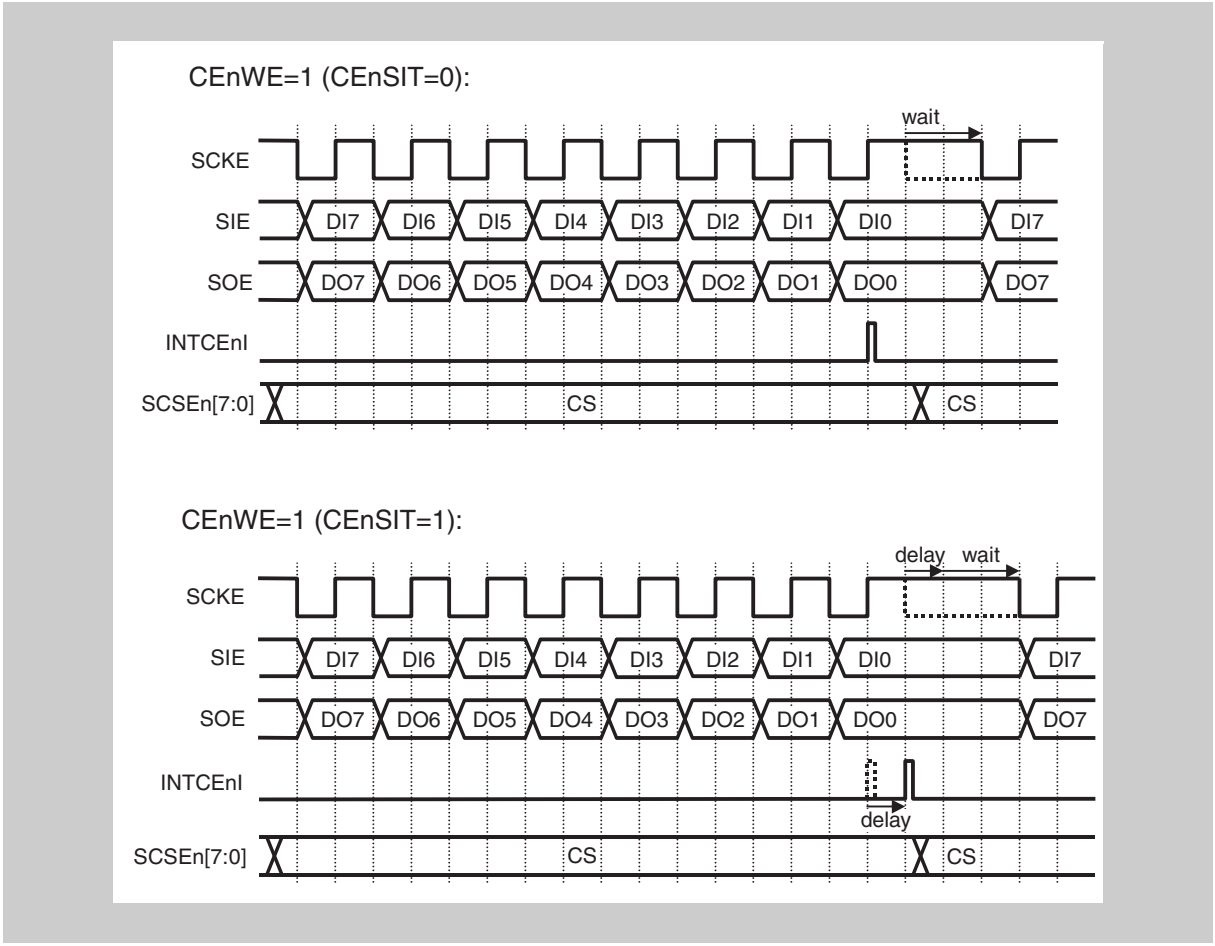


(2) Selection of transmit wait enable/disable

In master mode, while the enhanced timing is not enabled, the CEnCTL0 register's CEnWE bit setting can be used to delay the start of transmission by one SCKE clock cycle. The CEnWE bit takes effect only in the master mode and is ignored in slave mode.

Figure 15-17 below illustrates the CEnWE function, assuming a setting of CEnWE=1, CEnCSM=0, CEnOPE=0, CEnCKP=0, CEnDAP=0 and CEnDL[3:0] = [1,0,0,0].

Figure 15-17 Selection of transmit wait enable/disable



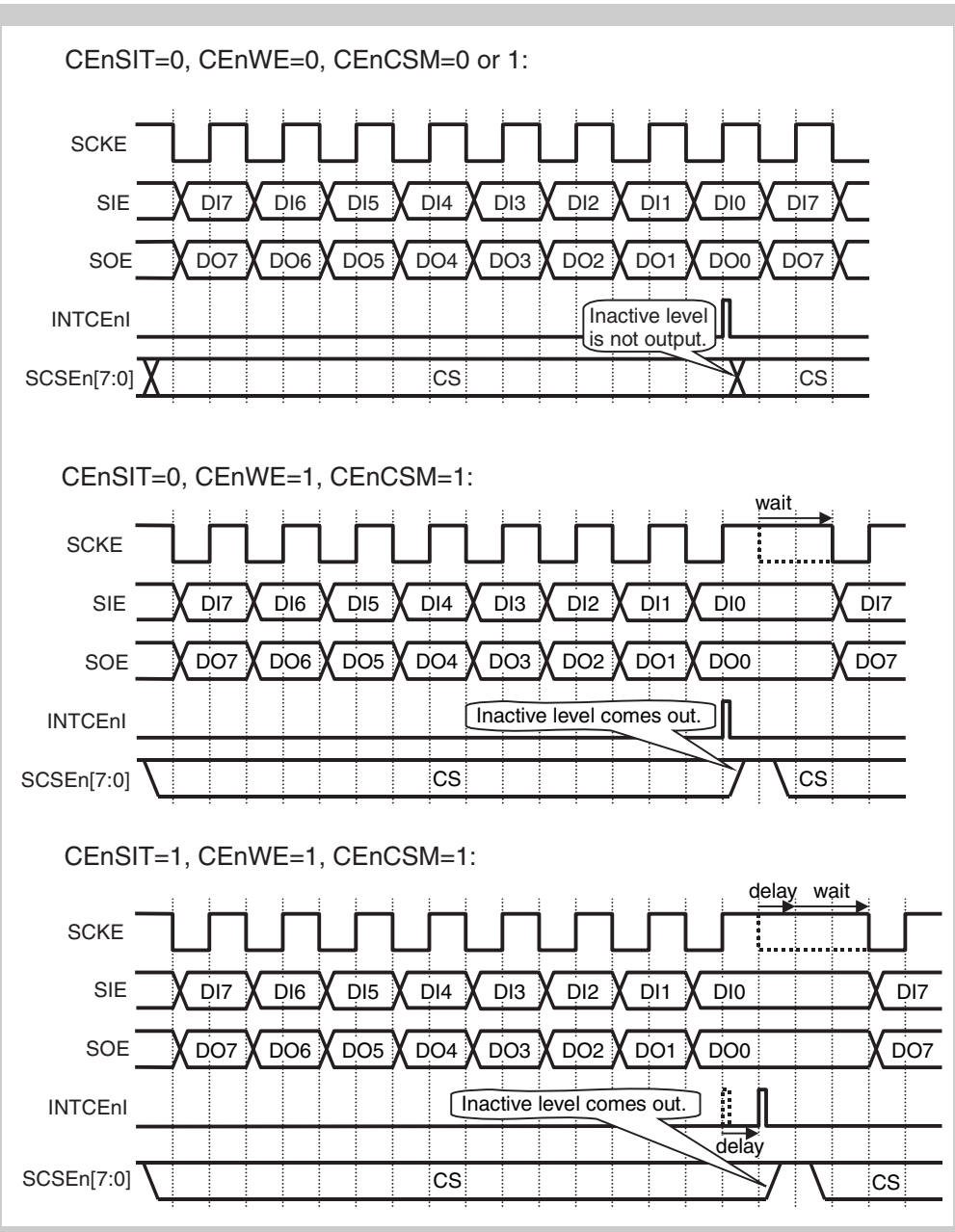
(3) Selection of chip-select mode

In master mode with CEnWE = 1 and CEnOPE=0, the CEnCSM bit setting can be used to output an inactive level at the chip select pins SCSEn[3:0] during the delay between data transmissions. The CEnCSM bit takes effect only in the master mode and is ignored in slave mode.

The CEnCSM bit has no effect while CEnWE is set to 0.

Figure 15-18 below illustrates the CEnCSM function, assuming a setting of CEnCKP=0, CEnDAP=0 and CEnDL[3:0] = [1,0,0,0].

Figure 15-18 Selection of chip-select mode



15.3.11 Default pin levels

(1) SCKE pin's default level

The SCKE pin's default level with the CEnCTL0 register settings CEnPWR=0 or (CEnTXE=0 and CEnRXE = 0) is as follows:

CEnCKP	CEnCKS[2:0]	SCKE default level	
0	1, 1, 1 (slave mode)*	1	← Initialization after reset
	Other than 1, 1, 1 (master mode)	1	
1	1, 1, 1 (slave mode)	1	
	Other than 1, 1, 1 (master mode)	0	

Note *: In slave mode, the SCKE pin is always set to “1”.

(2) SOE pin's default level

The SOE pin's default level with the CEnCTL0 register settings CEnPWR=0 or (CEnTXE=0 and CEnRXE=0) is as follows:

SOE pin's default level	
0	←Initialization after reset

(3) SCSEn0 to SCSEn7 pins' default level

The SCSEn0 to SCSEn7 pins' default level with the CEnCTL0 register settings CEnPWR=0 or (CEnTXE=0 and CEnRXE=0) is as follows:

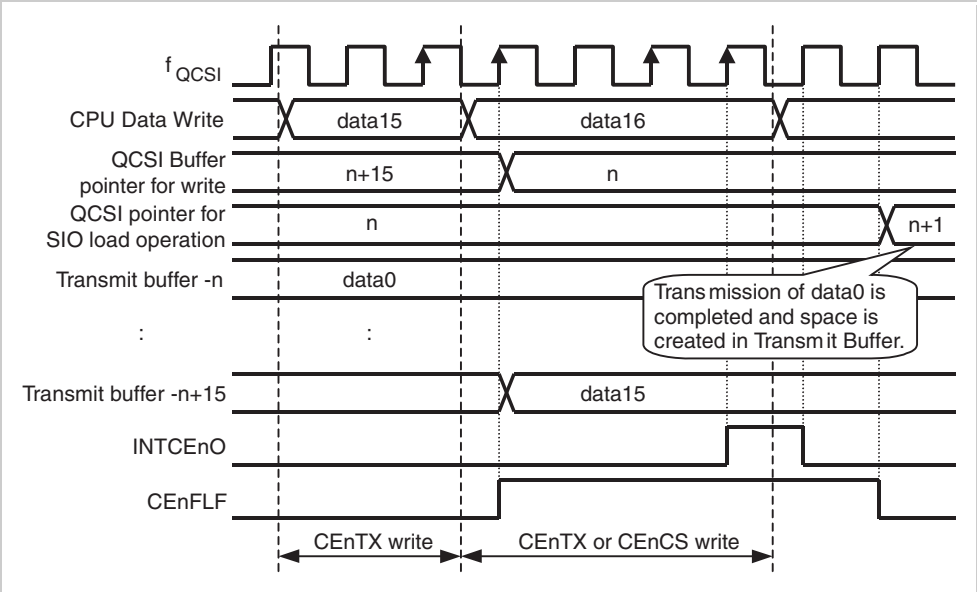
SCSEn0 to SCSEn7 pins' default level	
inactive	←Initialization after reset

15.3.12 Transmit buffer overflow interrupt signal (INTCEnO)

When the transmit FIFO buffer contains 16 elements, writing a 17th chip-select data (CEnCS write) or transfer data (CEnTX0 write) results in the generation of the overflow interrupt INTCEnO. For the 17th item, both chip-select and transfer data values are discarded.

The transmit FIFO buffer contains 16 elements if the FIFO pointer value for the write operation equals the FIFO pointer value for the SIO load operation plus 15. When the transfer is completed and the FIFO buffer pointer for the SIO load operation is incremented, space for one element is available again in transmit buffer.

Figure 15-19 Transmit buffer overflow interrupt signal (INTCEnO)

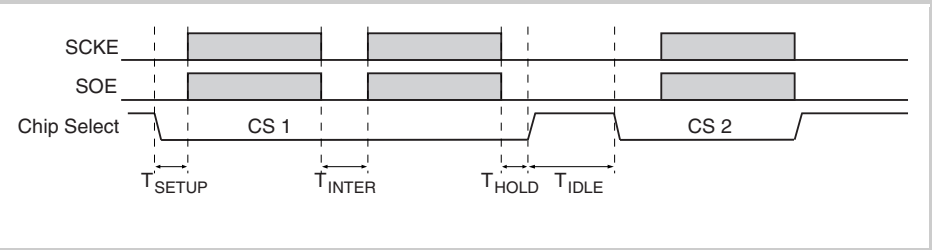


15.3.13 Enhanced chip select timing

(1) General

The Enhanced Queued-CSI CSIE allows specific timing settings for each chip select line. The figure below shows the general timings of the chip select lines:

Figure 15-20 General Overview on Enhanced Chip Select Timing



T_{SETUP} is the time between the chip select line becoming active and the first data bit to be transferred. T_{INTER} defines the delay between two transfers while the same chip select is chosen and active. After the last transfer, the chip select line remains active for a period set by T_{HOLD}. Finally, T_{IDLE} can be used to delay the start of a new transfer with a different chip select setting.

Each delay time is set relative to the serial communication clock SCKE and can range from 0 (no delay) up to 8 serial clocks.

(2) Chip select deactivation

A chip select line is selected by writing the corresponding value to the CEnCS[3:0] bits. The chip select line remains active until a value is written to the CEnCS[3:0] bits that is different from the previous value.

The chip select line remains active even when the transfer queue becomes empty. After a new element is written in the transfer queue, either the inter-data or the hold timing will be applied before the new element is transmitted. In other words, the queue idle time will be seen in addition to the defined chip select timing.

(3) Multiple chip select activation

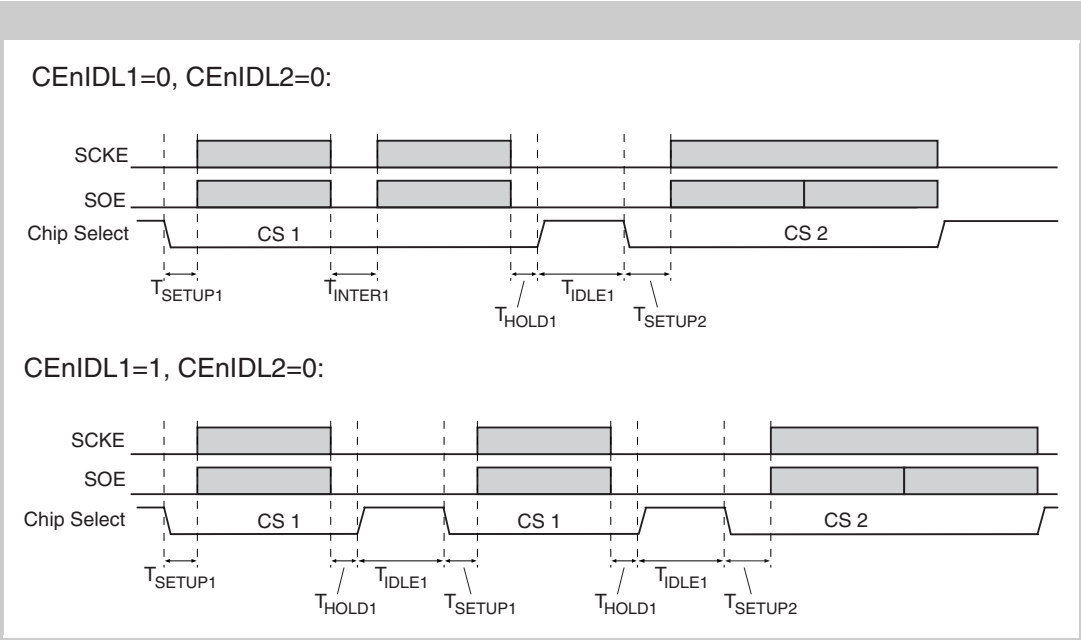
In 4 chip select mode, multiple chip selects can be active at the same time. When enhanced timing is enabled, the Queued-CSI macro will use the timing of the enabled chip select with the lowest number. But to ensure proper communication, it is mandatory that the timings of all chip selects that can be active at a same time are set to the same values.

(4) Enforced chip select idle setting

The chip select line is held active until the chip select setting of a transfer element changes. While this is suitable for most peripheral devices connected to the serial interface, some devices require an inactive state of the chip select line after each element transmitted.

The “Force Chip Select Idle” (CEnIDLm) bit is used to force this behaviour. *Figure 15-21* illustrates the difference between a standard communication and a communication with the CEnIDL1 bit set. It is assumed that the delay timings of CS2 are set as T_{INTER2}=0 and T_{HOLD2}=0, and that two elements are transferred to CS1 one at a time before a transfer of two elements to CS2.

Figure 15-21 Enforced chip select idle example

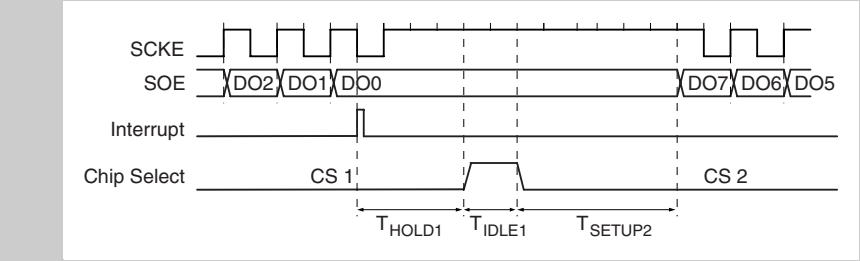


While the timing for CEnIDL1=0 uses T_{INTER1} as the delay between the two elements for CS1, the setting of CEnILD1=1 forces the T_{HOLD1}, T_{IDLE1} and T_{SETUP1} delays for the chip select line between the elements.

(5) Hold, idle and setup timing relative to $\overline{\text{SCKE}}$

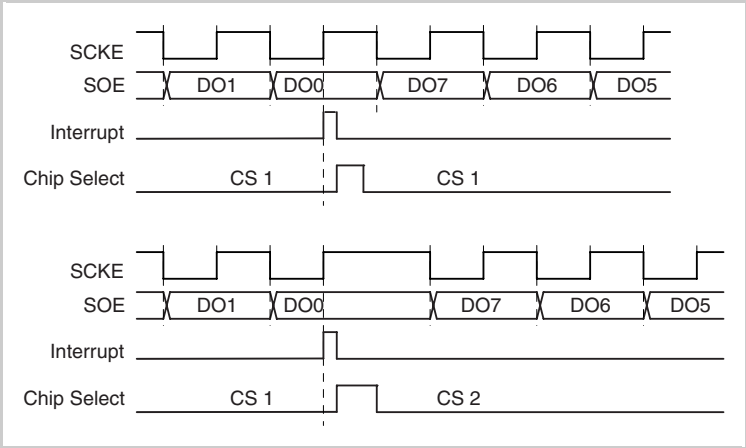
The start of any timing is at the last valid data signal of the SCKE, not at the end of the SCKE cycle. *Figure 15-22* illustrates the timing. It is assumed that CKP=0, DAP=1, T_{HOLD1}=2 SCKE, T_{IDLE1}=1 SCKE, T_{SETUP2}=3 SCKE, and that the data bits per element is 8:

Figure 15-22 Enhanced chip select timing extension relative to SCKE



With the setting of CKP=0 and DAP=1, data is output on the rising edge of SCKE and the data is valid at the falling edge. When the last data bit is sent, the interrupt is generated at the falling edge of the SCKE.

The next figure shows the timing with T_{HOLD1}=T_{IDLE1}=T_{SETUP2}=0, but CEnIDL1=1, CKP/DAP=0/0:



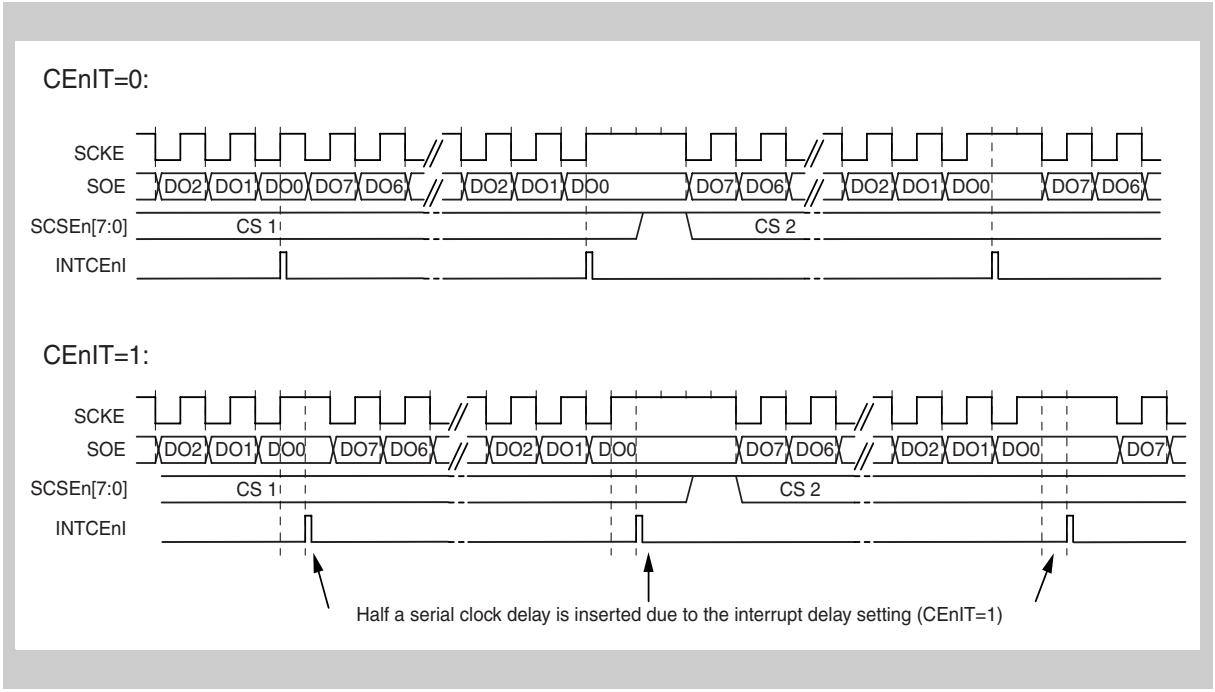
A small chip select idle pulse is generated after the “data valid” edge of SCKE, which is the rising edge in this example due to CKP=DAP=0.

(6) Impact of interrupt delay on enhanced timing

The CEnSIT bit allows to set a delay of half serial communication clock before the transfer complete interrupt is generated. This delay is inserted before any delay defined by the enhanced timing.

Figure 15-23 shows as example a communication with T_{INTER1}=0, T_{HOLD1}=0.5, T_{IDLE1}=1, T_{SETUP2}=0 and T_{INTER2}=1, using single buffer transfer mode and 8-bit data length.

Figure 15-23 Enforced chip select idle timing extension

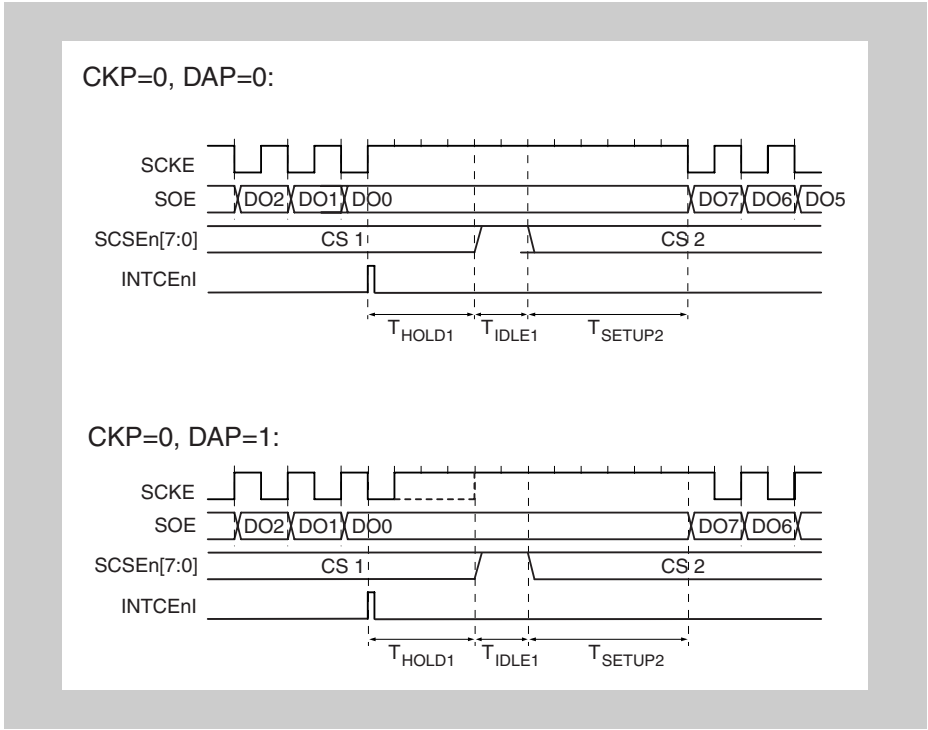


(7) Data phase details (dap=1) with enhanced timing

The DAP bit defines the phase of the data line relative to the serial clock. While the timing diagram for DAP=0 is straight forward, for DAP=1 there would be two options for the rising edge of the serial clock signal at the last data bit (DO0) when hold time is applied. Please use the figure below as reference.

The rising edge could be either done at the end of the DO0 bit, or it could be done at the end of the hold time (dashed line in the timing diagram). The Queued-CSI module places the rising edge at the end of the data bit.

Figure 15-24 Data phase details (DAP=1) with enhanced timing



(8) Change of clock phase and data phase

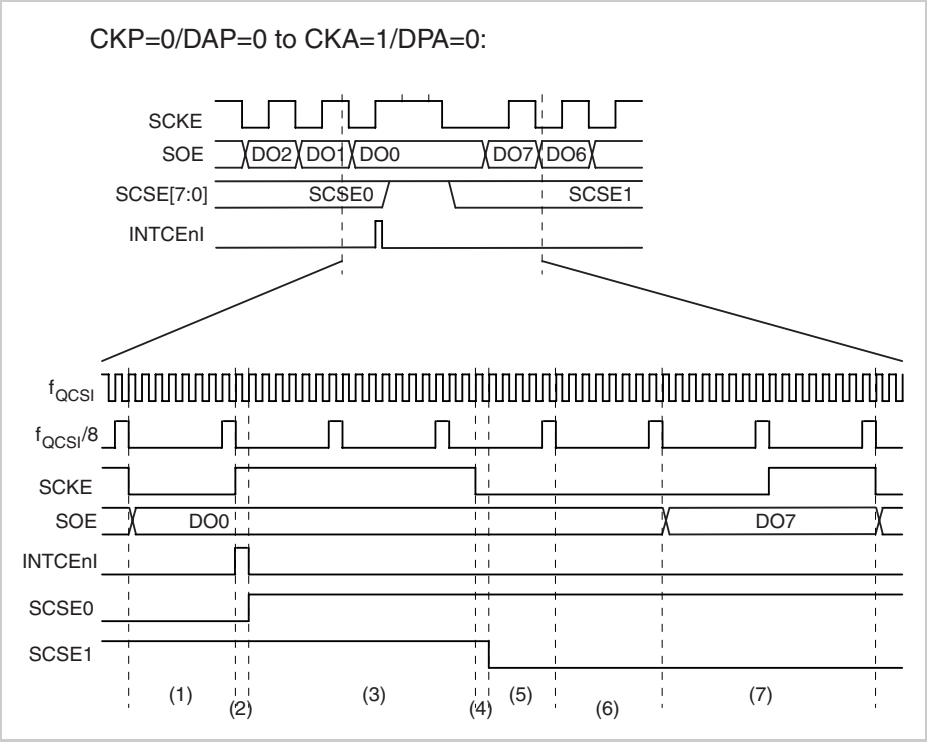
The enhanced timing also allows the definition of two clock polarity/data phase pairs. The following figures (*Figure 15-25* to *Figure 15-27*) illustrate the transition in polarity down to internal Queued-CSI operating clock level. In a nutshell, there is one macro clock delay between the change in polarity and the activation of the chip select signal.

Except when noted otherwise, it is assumed in the following examples that there is no setup delay defined for CS2. If a delay is defined, the standard rules would apply after the activation of the chip select signal.

Caution *Figure 15-25* shows the synchronisation delay after the new CS signal is activated, while in the other figures the synchronisation delay is ignored to simplify the drawings. The synchronization delay would apply, though!

Figure 15-25 shows the transition from SCSE0 with (CKP=0, DAP=0) to SCSE1 with (CKA=1, DPA=0), using the following settings: T_{HOLD0}=0, T_{IDLE0}=1, T_{SETUP1}=0.5, f_{SCKE}=f_{QCSI}/16:

Figure 15-25 CKP=0, DAP=0 to CKA=1, DPA=0 with enhanced timing



1. First half of the last data bit sent with SCSE0 active
2. At the rising edge of SCKE, the interrupt signal is generated.
3. Represents the idle time (T_{IDLE0}=1). One f_{QCSI} clock after the generation of the interrupt signal, SCSE0 is set to inactive. It is hold inactive for serial clock cycle plus one internal clock f_{QCSI}.
4. After the idle time, the serial clock polarity is changed due to the CKA=1 transition.

5. One internal clock f_{QCSI} after the change of the clock polarity SCSE1 is set to active. Then the timing is synchronized with the internal baud rate generator that operates continuously at $f_{QCSI}/8$.
6. Setup delay of SCSE1 (T_{SETUP1}=0.5).
7. Output of the first data bit with SCSEn1 active.

Figure 15-26 CKP=0, DAP=0 to CKA=1, DPA=1 with enhanced timing

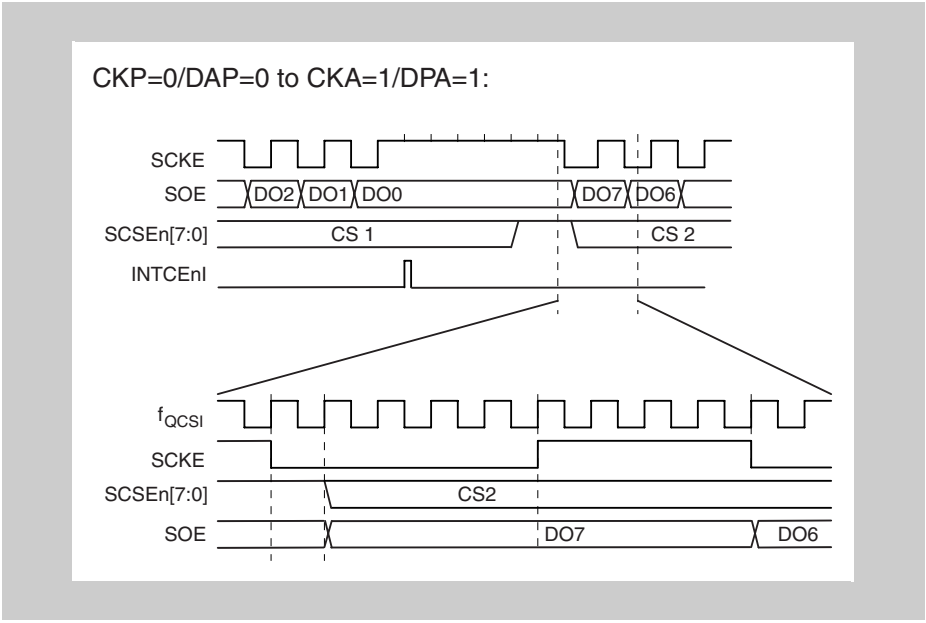


Figure 15-27 CKP=1, DAP=0 to CKA=0, DPA=1 with enhanced timing

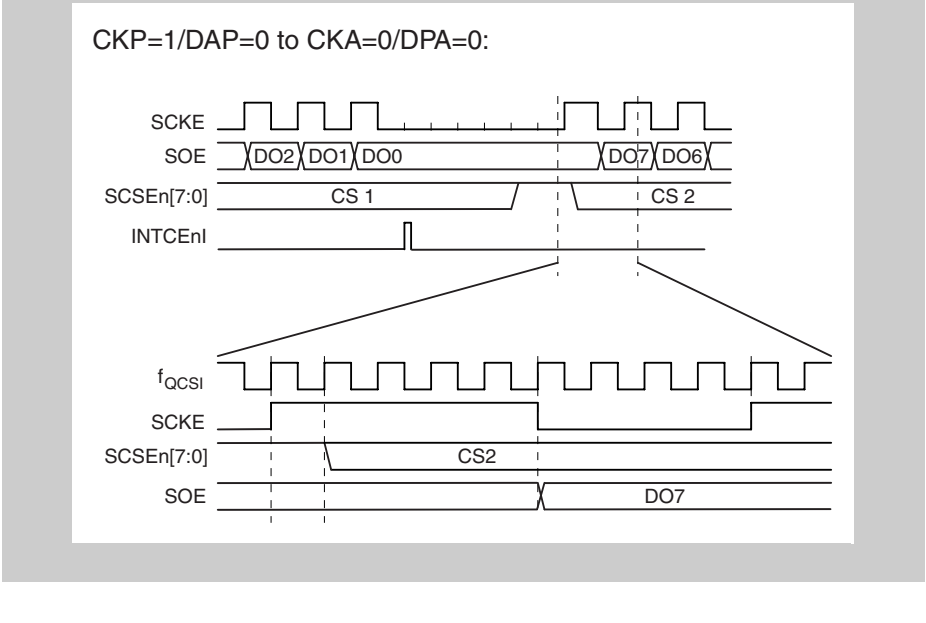
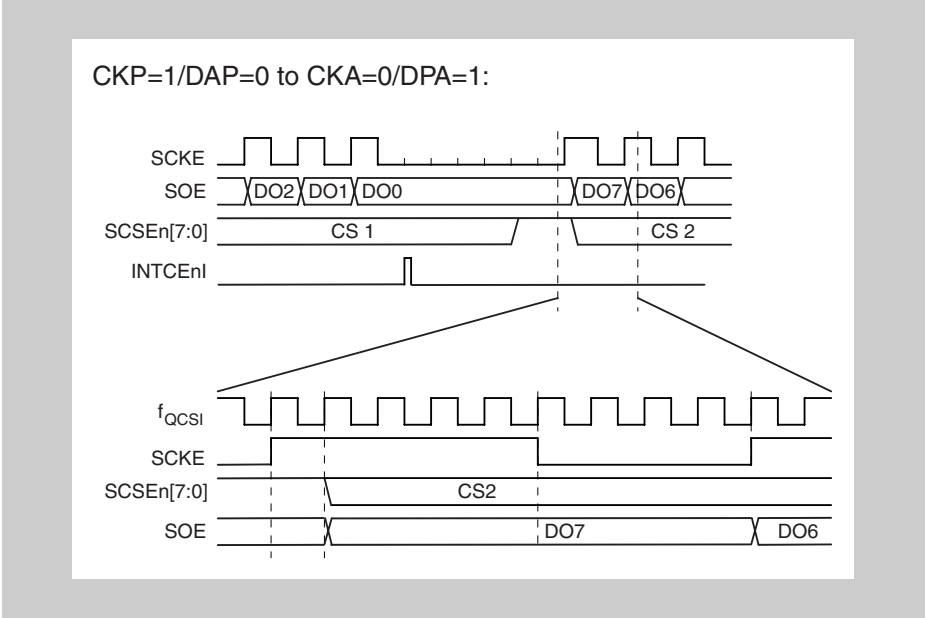


Figure 15-28 CKP=1, DAP=0 to CKA=0, DPA=0 with enhanced timing



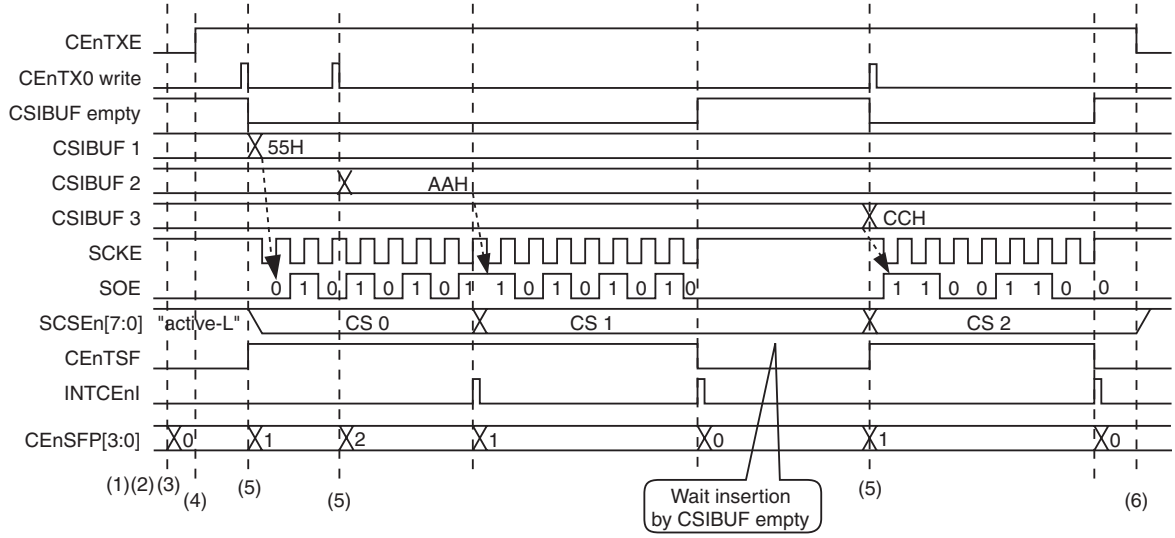
15.4 Operating Procedure

15.4.1 Single transfer mode (master mode, transmit only mode)

Figure 15-29 illustrates the timing of the following transfer:

- MSB first (CEnDIR = 0)
- No INTCEnI delay (CEnSIT=0)
- Transmission wait disabled (CEnWE = 0)
- CS inactive disabled (CEnCSM = 0)
- CEnCKP = 0, CEnDAP = 0
- Transmission data length of 8 bits (CEnDL[3:0] = [1,0,0,0])

Figure 15-29 Single transfer mode (master, transmit only) timing



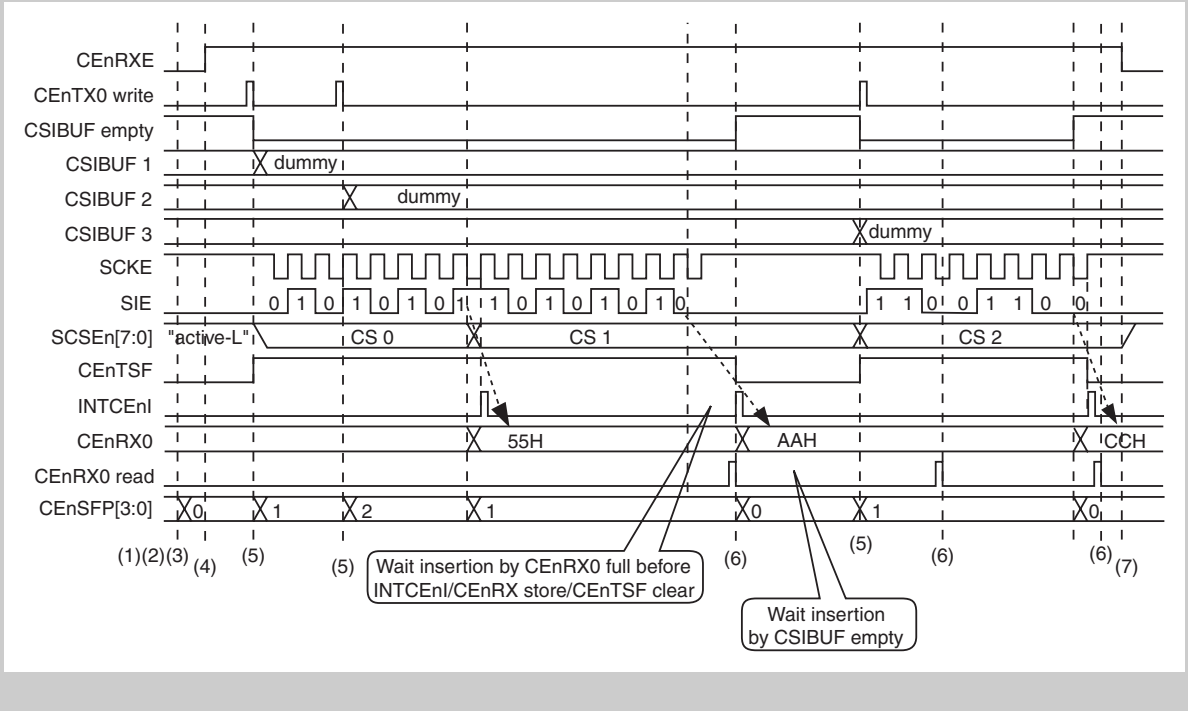
1. Set the CEnCTL0 register's CEnPWR bit to 1 to enable the supply of the Queued CSI operation clock.
2. Set the CEnCTL1 and CEnCTL2 registers to specify the transfer mode.
3. Write “1” in the CEnSTR register's CEnPCT bit to clear all FIFO pointers.
4. Specify the transfer mode using the CEnCTL0 register's CEnTMS, CEnDIR, and CEnSIT bits; at the same time, set the CEnTXE bit to 1 to enable transmission.
5. Make sure that the CEnSTR register's CEnFLF bit is set to 0, then write chip-select data and transmission data in the CEnCS and CEnTX0 registers in this order.
6. Repeat step (5) until the last element to be transmitted is written in the CEnCS/CEnTX0 registers.
7. Verify that CEnSTR.CEnTSF = 0 (CSIEn in idle state) and set the CEnCTL0 register's CEnTXE bit to 0 to disable transmission (end of transmission).

15.4.2 Single transfer mode (master mode, receive only mode)

Figure 15-30 illustrates the timing of the following transfer:

- MSB first (CEnDIR = 0)
- No INTCEnI delay (CEnSIT = 0)
- Transmission wait disabled (CEnWE = 0)
- CS inactive disabled (CEnCSM = 0)
- CEnCKP = 0, CEnDAP = 1
- Transmission data length of 8 bits (CEnDL[3:0] = [1,0,0,0])

Figure 15-30 Single transfer mode (master, receive only) timing



1. Set the CEnCTL0 register's CEnPWR bit to 1 to enable the supply of the Queued CSI operation clock.
2. Set the CEnCTL1 and CEnCTL2 registers to specify the transfer mode.
3. Write “1” in the CEnSTR register's CEnPCT bit to clear all FIFO pointers.
4. Specify the transfer mode using the CEnCTL0 register's CEnTMS, CEnDIR, and CEnSIT bits; at the same time, set the CEnRXE bit to 1 to enable the receive operation.
5. Make sure that the CEnSTR register's CEnFLF bit is set to 0, then write chip-select data and dummy transmission data in the CEnCS and CEnTX0 registers in this order (start-of-receive trigger).
6. Check for a reception to be completed (e.g. by monitoring the INTCEnI interrupt). If so, read the CEnRX0 register.
7. Repeat steps (5) and (6) until the last element is received and read from the CEnRX0 register.

8. Verify that CEnSTR.CEnTSF = 0 (CSIEn in idle state) and set the CEnCTL0 register's CEnRXE bit to 0 to disable the receive operation (end of receive operation).

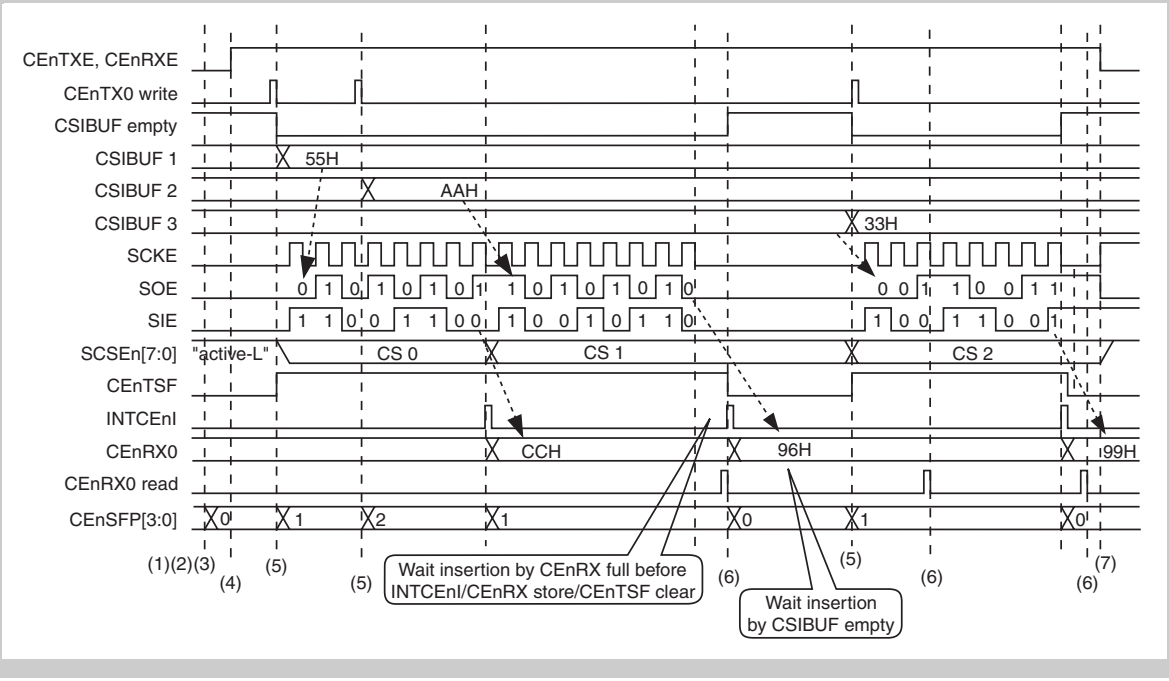
Note The SOE pin is invalid and maintains its signal level, as the output latch is disabled.

15.4.3 Single transfer mode (master mode, transmit/receive mode)

Figure 15-31 illustrates the timing of the following transfer:

- MSB first (CEnDIR = 0)
- No INTCEnI delay (CEnSIT = 0)
- Transmission wait disabled (CEnWE = 0)
- CS inactive disabled (CEnCSM = 0)
- CEnCKP = 1
- CEnDAP = 0
- Transmission data length of 8 bits (CEnDL[3:0] = [1,0,0,0])

Figure 15-31 Single transfer mode (master, transmit/receive) timing



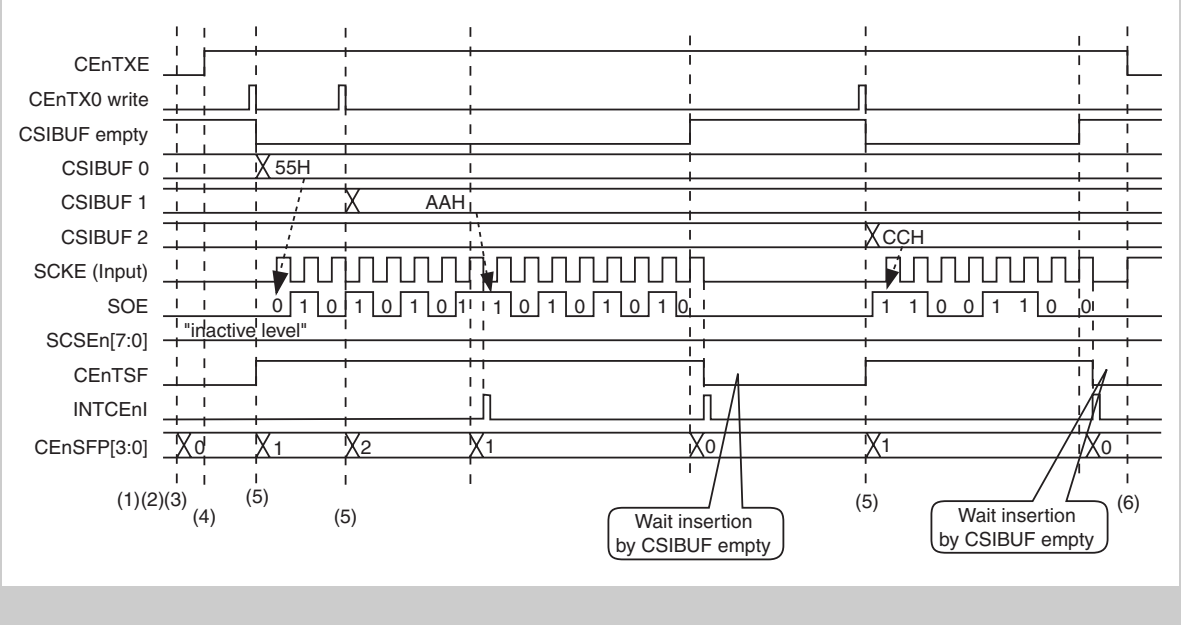
1. Set the CEnCTL0 register's CEnPWR bit to 1 to enable the supply of the Queued CSI operation clock.
2. Set the CEnCTL1 and CEnCTL2 registers to specify the transfer mode.
3. Write “1” in the CEnSTR register's CEnPCT bit to clear all FIFO pointers.
4. Specify the transfer mode using the CEnCTL0 register's CEnTMS, CEnDIR, and CEnSIT bits; at the same time, set the CEnTXE and CEnRXE bits to 1 to enable the transmit/receive operation.
5. Make sure that the CEnSTR register's CEnFLF bit is set to 0, then write chip-select data and transmission data in the CEnCS and CEnTX0 registers in this order.
6. Check for a transmission to be finished (e.g. by monitoring the INTCEnI interrupt). If so, read the CEnRX0 register.
7. Repeat steps (5) and (6) until the last element is send/received and read from the CEnRX0 register.
8. Set the CEnCTL0 register's CEnTXE and CEnRXE bits to 0 to disable the transmit/receive operation (end of transmit/receive operation).

15.4.4 Single transfer mode (slave mode, transmit only mode)

Figure 15-32 illustrates the timing of the following transfer:

- MSB first (CEnDIR = 0)
- No INTCEnI delay (CEnSIT = 0)
- Transmission wait disabled (CEnWE = 0)
- CS inactive disabled (CEnCSM = 0)
- CEnCKP = 1
- CEnDAP = 1
- Transmission data length of 8 bits (CEnDL[3:0] = [1,0,0,0])

Figure 15-32 Single transfer mode (slave, transmit only) timing



1. Set the CEnCTL0 register's CEnPWR bit to 1 to enable the supply of the Queued CSI operation clock.
2. Set the CEnCTL1 and CEnCTL2 registers to specify the transfer mode.
3. Write “1” in the CEnSTR register's CEnPCT bit to clear all FIFO pointers.
4. Specify the transfer mode using the CEnCTL0 register's CEnTMS, CEnDIR, and CEnSIT bits; at the same time, set the CEnTXE bit to 1 to enable transmission.
5. Make sure that the CEnSTR register's CEnFLF bit is set to 0, then write transmission data in the CEnTX0 register. (In the slave mode, there is no need to set data in the CEnCS register, as the chip select pins SCSEn[3:0] are not used.)
6. Repeat step (5) until the last element to be transmitted is written in the CEnTX0 register.
7. Set the CEnCTL0 register's CEnTXE bit to 0 to disable transmission (end of transmission).

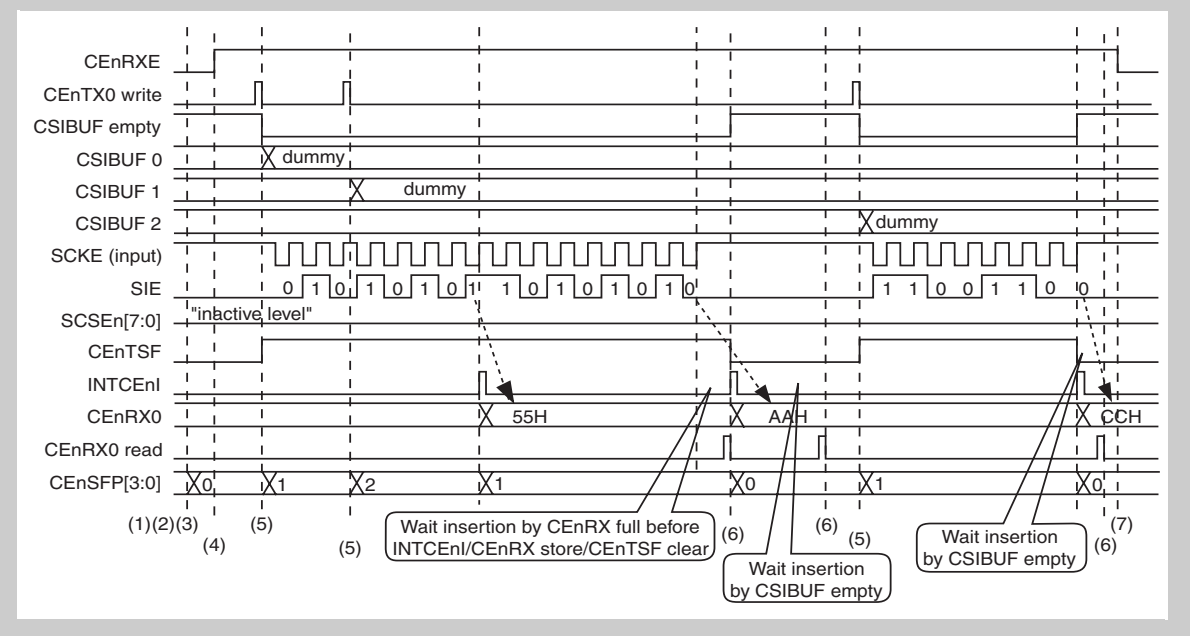
To continue transmission, repeat step (5) before executing step (6).

15.4.5 Single transfer mode (slave mode, receive only mode)

Figure 15-33 illustrates the timing of the following transfer:

- MSB first (CEnDIR = 0)
- No INTCEnI delay (CEnSIT = 0)
- Transmission wait disabled (CEnWE = 0)
- CS inactive disabled (CEnCSM = 0)
- CEnCKP = 0
- CEnDAP = 0
- Transmission data length of 8 bits (CEnDL[3:0] = [1,0,0,0])

Figure 15-33 Single transfer mode (slave, receive only) timing



1. Set the CEnCTL0 register's CEnPWR bit to 1 to enable the supply of the Queued CSI operation clock.
2. Set the CEnCTL1 and CEnCTL2 registers to specify the transfer mode.
3. Write “1” in the CEnSTR register's CEnPCT bit to clear all FIFO pointers.
4. Specify the transfer mode using the CEnCTL0 register's CEnTMS, CEnDIR, and CEnSIT bits; at the same time, set the CEnRXE bit to 1 to enable the receive operation.
5. Make sure that the CEnSTR register's CEnFLF bit is set to 0, then write dummy transmission data in the CEnTX0 register (start-of-receive trigger). (In slave mode, there is no need to set data in the CEnCS register)
6. Check for a reception to be completed (e.g. by monitoring the INTCEnI interrupt). If so, read the CEnRX0 register.
7. Repeat steps (5) and (6) until the last element is received and read from the CEnRX0 register.
8. Set the CEnCTL0 register's CEnRXE bit to 0 to disable the receive operation (end of receive operation).

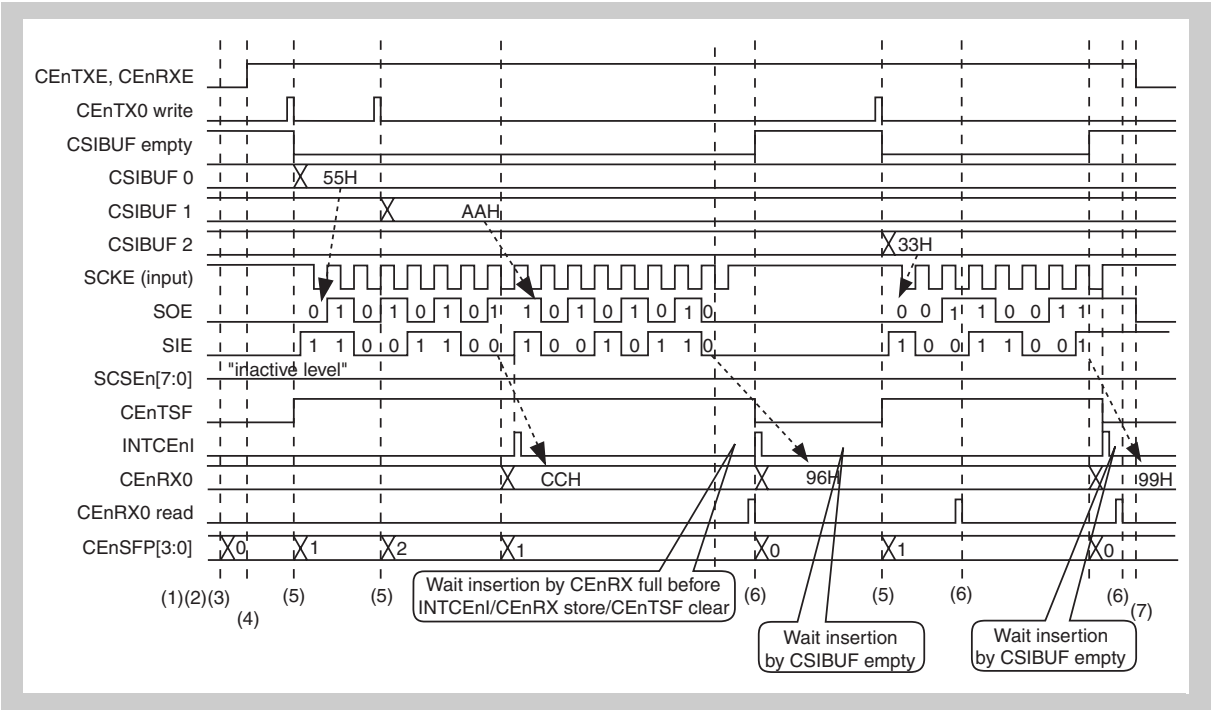
Note 1. The SOEn pin is invalid and maintaining its signal level, as the output latch is disabled.

15.4.6 Single transfer mode (slave mode, transmit/receive mode)

Figure 15-34 illustrates the timing of the following transfer:

- MSB first (CEnDIR = 0)
- No INTCEnI delay (CEnSIT = 0)
- Transmission wait disabled (CEnWE = 0)
- CS inactive disabled (CEnCSM = 0)
- CEnCKP = 0
- CEnDAP = 1
- Transmission data length of 8 bits (CEnDL[3:0] = [1,0,0,0])

Figure 15-34 Single transfer mode (slave, transmit/receive) timing



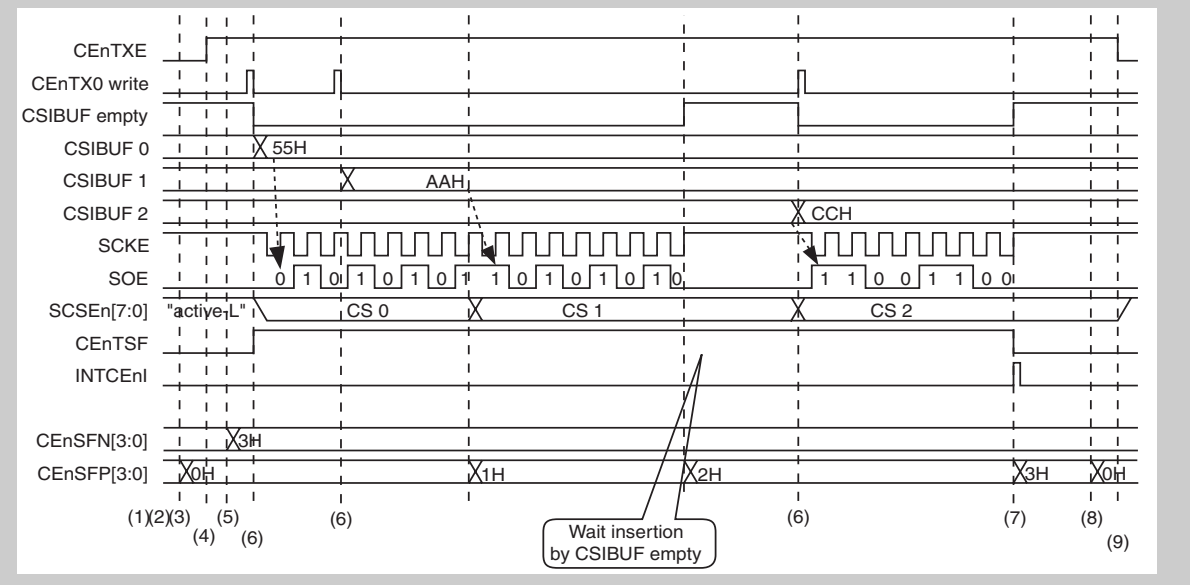
1. Set the CEnCTL0 register's CEnPWR bit to 1 to enable the supply of the Queued CSI operation clock.
2. Set the CEnCTL1 and CEnCTL2 registers to specify the transfer mode.
3. Write “1” in the CEnSTR register's CEnPCT bit to clear all FIFO pointers.
4. Specify the transmit mode using the CEnCTL0 register's CEnTMS, CEnDIR, and CEnSIT bits; at the same time, set the CEnTXE and CEnRXE bits to 1 to enable the transmit/receive operation.
5. Make sure that the CEnSTR register's CEnFLF bit is set to 0, then write transmission data in the CEnTX0 register. (In the slave mode, there is no need to set data in the CEnCS register)
6. Check for a reception to be completed (e.g. by monitoring the INTCEnI interrupt). If so, read the CEnRX0 register.
7. Repeat steps (5) and (6) until the last element is send/received and read from the CEnRX0 register.
8. Set the CEnCTL0 register's CEnTXE and CEnRXE bits to 0 to disable the transmit/receive operation (end of transmit/receive operation).

15.4.7 Block transfer mode (master mode, transmit only mode)

Figure 15-35 illustrates the timing of the following transfer:

- MSB first (CEnDIR = 0)
- No INTCEnI delay (CEnSIT = 0)
- Transmission wait disabled (CEnWE = 0)
- CS inactive disabled (CEnCSM = 0)
- CEnCKP = 0
- CEnDAP = 0
- Transmission data length of 8 bits (CEnDL[3:0] = [1,0,0,0])

Figure 15-35 Block transfer mode (master, transmit only) timing



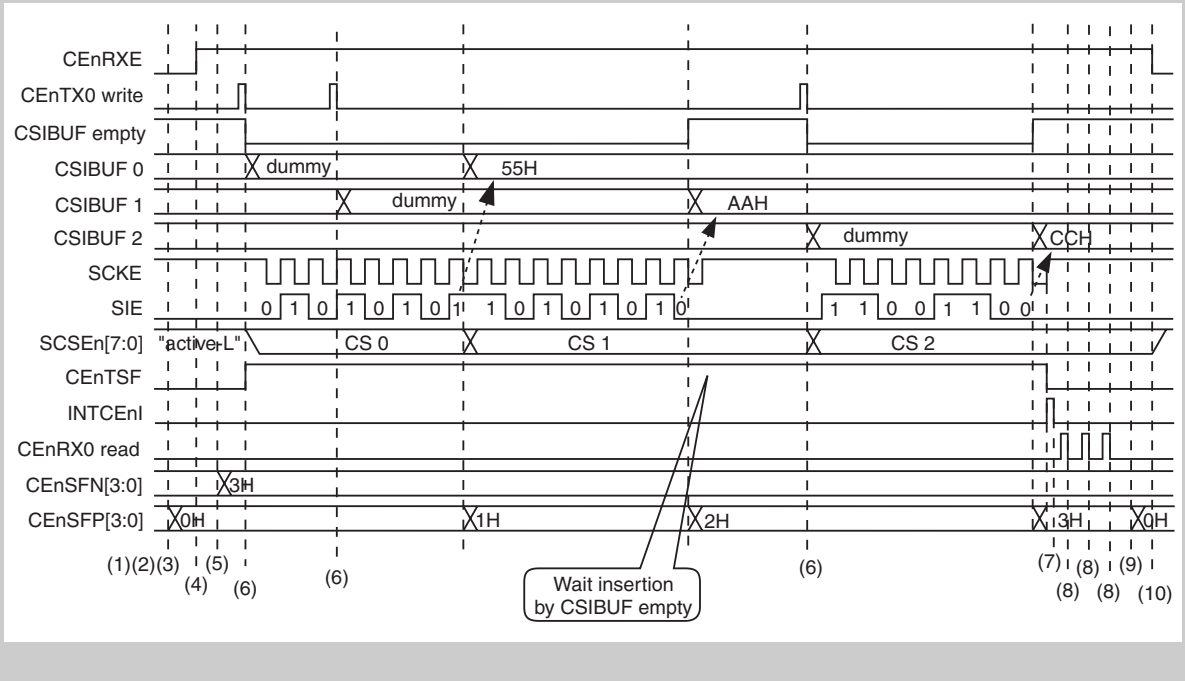
1. Set the CEnCTL0 register's CEnPWR bit to 1 to enable the supply of the Queued CSI operation clock.
2. Set the CEnCTL1 and CEnCTL2 registers to specify the transfer mode.
3. Write “1” in the CEnSTR register's CEnPCT bit to clear all FIFO pointers.
4. Specify the transfer mode using the CEnCTL0 register's CEnTMS, CEnDIR, and CEnSIT bits; at the same time, set the CEnTXE bit to 1 to enable transmission.
5. Set the number of send-data items in the CEnCTL3 register's CEnSFN[3:0] bits.
6. Make sure that the CEnSTR register's CEnFLF bit is set to 0, then write chip-select data and transmission data in the CEnCS and CEnTX0 registers in this order.
7. Wait for the transmissions to be completed (e.g. by monitoring the INTCEnI interrupt).
8. Write “1” in the CEnSTR register's CEnPCT bit and clear all FIFO pointers for the next transmission.
9. To continue transmission, repeat steps (5) - (8).
10. Set the CEnCTL0 register's CEnTXE bit to 0 to disable transmission (end of transmission).

15.4.8 Block transfer mode (master mode, receive only mode)

Figure 15-36 illustrates the timing of the following transfer:

- MSB first (CEnDIR = 0)
- No INTCEnI delay (CEnSIT = 0)
- Transmission wait disabled (CEnWE = 0)
- CS inactive disabled (CEnCSM = 0)
- CEnCKP = 0
- CEnDAP = 1
- Transmission data length of 8 bits (CEnDL[3:0] = [1,0,0,0])

Figure 15-36 Block transfer mode (master, receive only) timing



1. Set the CEnCTL0 register's CEnPWR bit to 1 to enable the supply of the Queued CSI operation clock.
2. Set the CEnCTL1 and CEnCTL2 registers to specify the transfer mode.
3. Write “1” in the CEnSTR register's CEnPCT bit to clear all FIFO pointers.
4. Specify the transfer mode using the CEnCTL0 register's CEnTMS, CEnDIR, and CEnSIT bits; at the same time, set the CEnRXE bit to 1 to enable the receive operation.
5. Set the number of receive-data items in the CEnCTL3 register's CEnSFN[3:0] bits.
6. Make sure that the CEnSTR register's CEnFLF bit is set to 0, then write chip-select data and dummy transmission data in the CEnCS and CEnTX0 registers in this order (start-of-receive trigger).
7. Wait for the receptions to be completed (e.g. by monitoring the INTCEnI interrupt).
8. Read the received data by multiple read of the CEnRX0 register (= sequential read from the FIFO).

9. Write “1” in the CEnSTR register's CEnPCT bit and clear all FIFO pointers for the next transmission.
10. To continue reception, repeat steps (5) - (9).
11. Set the CEnCTL0 register's CEnRXE bit to 0 to disable the receive operation (end of receive operation).

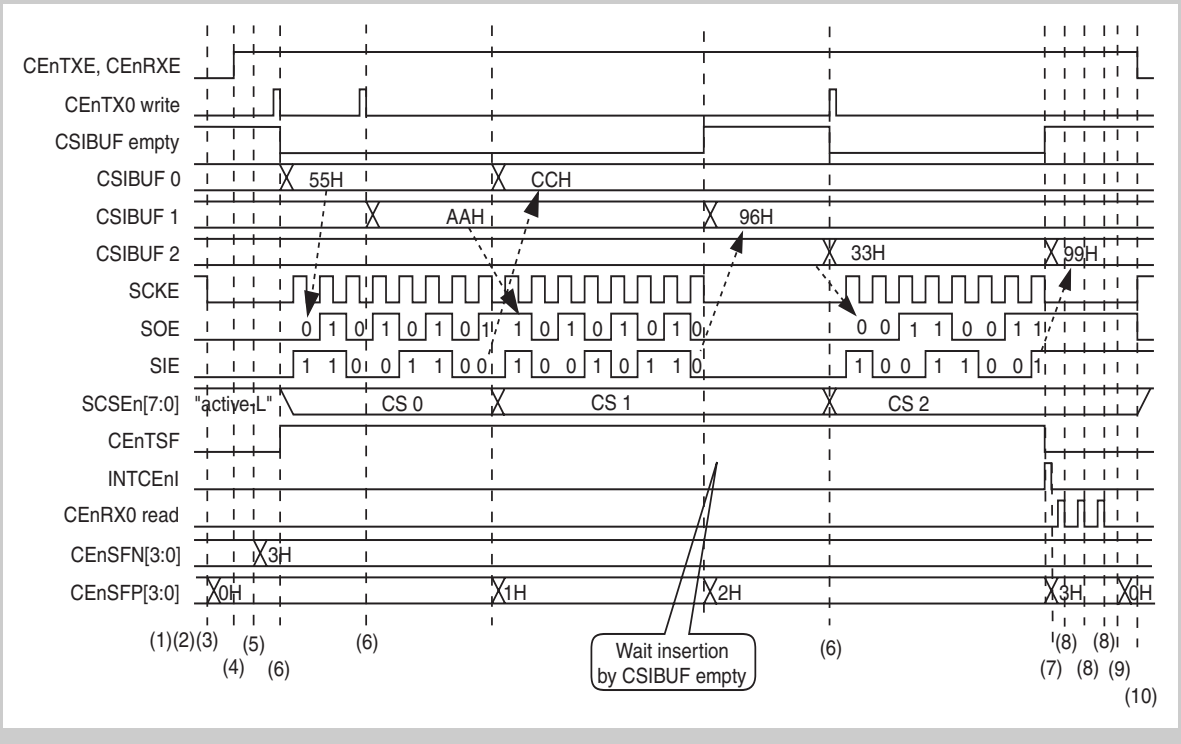
Note The SOEn pin is invalid and maintains its signal level, as the output latch is disabled.

15.4.9 Block transfer mode (master mode, transmit/receive mode)

Figure 15-37 illustrates the timing of the following transfer:

- MSB first (CEnDIR = 0)
- No INTCEnI delay (CEnSIT = 0)
- Transmission wait disabled (CEnWE = 0)
- CS inactive disabled (CEnCSM = 0)
- CEnCKP = 1
- CEnDAP = 0
- Transmission data length of 8 bits (CEnDL[3:0] = [1,0,0,0])

Figure 15-37 Block transfer mode (master, transmit/receive) timing



1. Set the CEnCTL0 register's CEnPWR bit to 1 to enable the supply of the Queued CSI operation clock.
2. Set the CEnCTL1 and CEnCTL2 registers to specify the transfer mode.
3. Write “1” in the CEnSTR register's CEnPCT bit to clear all FIFO pointers.

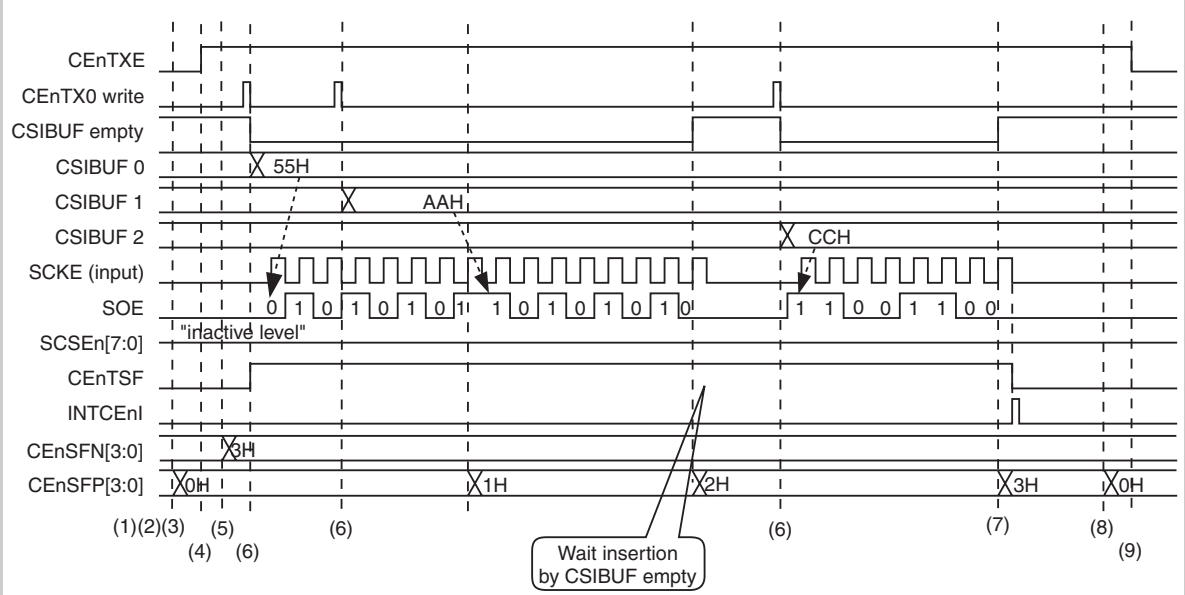
4. Specify the transfer mode using the CEnCTL0 register's CEnTMS, CEnDIR, and CEnSIT bits; at the same time, set the CEnTXE and CEnRXE bits to 1 to enable the transmit/receive operation.
5. Set the number of transmit/receive data items in the CEnCTL3 register's CEnSFN[3:0] bits.
6. Make sure that the CEnSTR register's CEnFLF bit is set to 0, then write chip-select data and transmission data in the CEnCS and CEnTX0 registers in this order.
7. Wait for the transmissions/receptions to be completed (e.g. by monitoring the INTCEnI interrupt).
8. Read the received data by multiple read of the CEnRX0 register (= sequential read from the FIFO).
9. Write “1” in the CEnSTR register's CEnPCT bit and clear all FIFO pointers for the next transmission.
10. To continue transmission/reception, repeat steps (5) - (9).
11. Set the CEnCTL0 register's CEnTXE and CEnRXE bits to 0 to disable the transmit/receive operation (end of transmit/receive operation).

15.4.10 Block transfer mode (slave mode, transmit only mode)

Figure 15-38 illustrates the timing of the following transfer:

- MSB first (CEnDIR = 0)
- No INTCEnI delay (CEnSIT = 0)
- Transmission wait disabled (CEnWE = 0)
- CS inactive disabled (CEnCSM = 0)
- CEnCKP = 1
- CEnDAP = 1
- Transmission data length of 8 bits (CEnDL[3:0] = [1,0,0,0])

Figure 15-38 Block transfer mode (slave, transmit only) timing



1. Set the CEnCTL0 register's CEnPWR bit to 1 to enable the supply of the Queued CSI operation clock.
2. Set the CEnCTL1 and CEnCTL2 registers to specify the transfer mode.
3. Write “1” in the CEnSTR register's CEnPCT bit to clear all FIFO pointers.
4. Specify the transfer mode using the CEnCTL0 register's CEnTMS, CEnDIR, and CEnSIT bits; at the same time, set the CEnTXE bit to 1 to enable transmission.
5. Set the number of send-data items in the CEnCTL3 register's CEnSFN[3:0] bits.
6. Make sure that the CEnSTR register's CEnFLF bit is set to 0, then write transmission data in the CEnTX0 register. (In the slave mode, there is no need to set data in the CEnCS register, as the chip select pins SCSEn[3:0] are not used.)
7. Wait for the transmissions to be completed (e.g. by monitoring the INTCEnI interrupt).

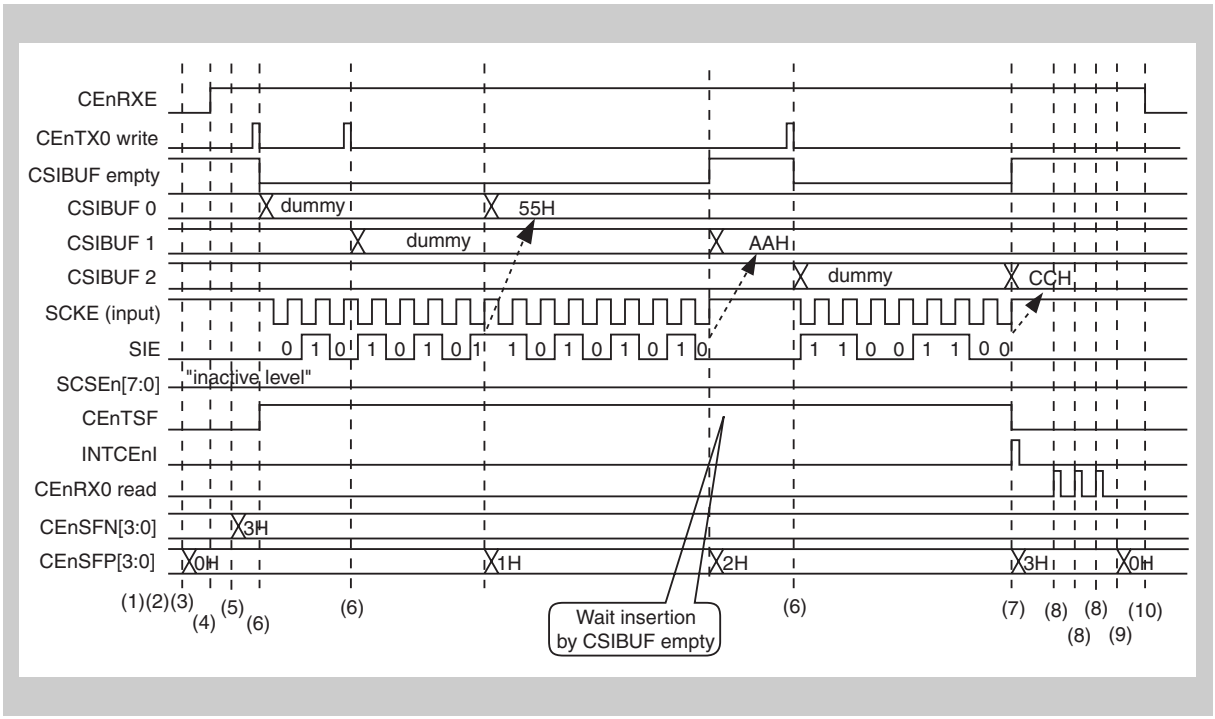
8. Write “1” in the CEnSTR register's CEnPCT bit and clear all FIFO pointers for the next transmission.
9. To continue transmission, repeat steps (5) - (8).
10. Set the CEnCTL0 register's CEnTXE bit to 0 to disable transmission (end of transmission).

15.4.11 Block transfer mode (slave mode, receive only mode)

Figure 15-39 illustrates the timing of the following transfer:

- MSB first (CEnDIR = 0)
- No INTCEnI delay (CEnSIT = 0)
- Transmission wait disabled (CEnWE = 0)
- CS inactive disabled (CEnCSM = 0)
- CEnCKP = 0
- CEnDAP = 0
- Transmission data length of 8 bits (CEnDL[3:0] = [1,0,0,0])

Figure 15-39 Block transfer mode (slave, receive only) timing



1. Set the CEnCTL0 register's CEnPWR bit to 1 to enable the supply of the Queued CSI operation clock.
2. Set the CEnCTL1 and CEnCTL2 registers to specify the transfer mode.
3. Write “1” in the CEnSTR register's CEnPCT bit to clear all FIFO pointers.
4. Specify the transfer mode using the CEnCTL0 register's CEnTMS, CEnDIR, and CEnSIT bits; at the same time, set the CEnRXE bit to 1 to enable the receive operation.
5. Set the number of receive-data items in the CEnCTL3 register's CEnSFN[3:0] bits.

6. Make sure that the CEnSTR register's CEnFLF bit is set to 0, then write dummy transmission data in the CEnTX0 register (start-of-receive trigger). (In the slave mode, there is no need to set data in the CEnCS register, as the chip select pins SCSEn[3:0] are not used.)
7. Wait for the receptions to be completed (e.g. by monitoring the INTCEnI interrupt).
8. Read the received data by multiple read of the CEnRX0 register (= sequential read from the FIFO).
9. Write “1” in the CEnSTR register's CEnPCT bit and clear all FIFO pointers for the next transmission.
10. To continue reception, repeat steps (5) - (9).
11. Set the CEnCTL0 register's CEnRXE bit to 0 to disable the receive operation (end of receive operation).

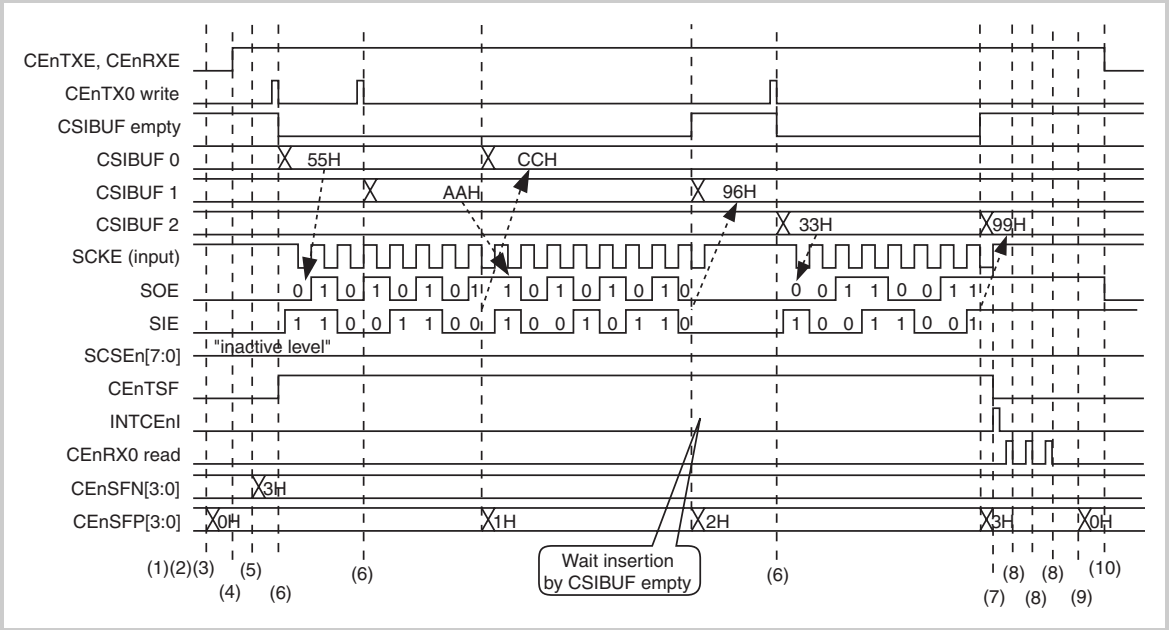
Note The SOEn pin is invalid and maintains its signal level, as the output latch is disabled.

15.4.12 Block transfer mode (slave mode, transmit/receive mode)

Figure 15-40 illustrates the timing of the following transfer:

- MSB first (CEnDIR = 0)
- No INTCEnI delay (CEnSIT = 0)
- Transmission wait disabled (CEnWE = 0)
- CS inactive disabled (CEnCSM = 0)
- CEnCKP = 0
- CEnDAP = 1
- Transmission data length of 8 bits (CEnDL[3:0] = [1,0,0,0])

Figure 15-40 Block transfer mode (slave, transmit/receive) timing



1. Set the CEnCTL0 register's CEnPWR bit to 1 to enable the supply of the Queued CSI operation clock.
2. Set the CEnCTL1 and CEnCTL2 registers to specify the transfer mode.
3. Write “1” in the CEnSTR register's CEnPCT bit to clear all FIFO pointers.
4. Specify the transfer mode using the CEnCTL0 register's CEnTMS, CEnDIR, and CEnSIT bits; at the same time, set the CEnTXE and CEnRXE bits to 1 to enable the transmit/receive operation.
5. Set the number of transmit/receive data items in the CEnCTL3 register's CEnSFN[3:0] bits.
6. Make sure that the CEnSTR register's CEnFLF bit is set to 0, then write transmission data in the CEnTX0 register. (In the slave mode, there is no need to set data in the CEnCS register, as the chip select pins SCSEn[3:0] are not used.)
7. Wait for the transmissions/receptions to be completed (e.g. by monitoring the INTCEnI interrupt).

8. Read the received data by multiple read of the CEnRX0 register (= sequential read from the FIFO).
9. Write “1” in the CEnSTR register's CEnPCT bit and clear all FIFO pointers for the next transmission.
10. To continue transmission, repeat steps (5) - (9).
11. Set the CEnCTL0 register's CEnTXE and CEnRXE bits to 0 to disable the transmit/receive operation (end of transmit/receive operation).

Chapter 16 Asynchronous Serial Interface (UARTD)

The V850E/Rx3 microcontroller includes two channels of the Asynchronous Serial Interface (UARTD).

16.1 Features

Instances

The V850E/RG3 microcontrollers has 2 instances of the clocked serial interface UARTD.

Table 16-1 Instances of UARTD

UARTD	
Instances	2
Names	UARTD0 to UARTD1

Throughout this chapter, the individual instances of clocked serial interface are identified by “n”, for example UARTDn, or UDnCTL0 for the control register 0 of UARTDn.

Clock supply

Each UARTDn provides 12 clock inputs.

- Twelve clock inputs UDnTCKI0 to UDnTCKIB are connected to the clock generator output clocks.
- Three clock inputs are not connected.

Table 16-2 Clock inputs of UARTDn

UARTDn clock input	Clock frequency
UDnTCKI0	16 MHz
UDnTCKI1	8 MHz
UDnTCKI2	4 MHz
UDnTCKI3	2 MHz
UDnTCKI4	1 MHz
UDnTCKI5	500 KHz
UDnTCKI6	125 KHz
UDnTCKI7	62.5 KHz
UDnTCKI8	31.25 KHz
UDnTCKI9	15.625 KHz
UDnTCKIA	7.812 KHz
UDnTCKIB	3.906 KHz

Baud rate setting example

The following tables summarizes some useful settings of the Baud Rate Generator for some common UARTD baud rates.

For detailed information concerning baud rate calculation refer to the section *Chapter 16, Section 16.7, Baud Rate Generator* in this chapter.

Table 16-3 Baud rate generator setting data

Target baud rate	UDnCTL1 Clock Selector		UDnCTL2 Clock Selector		Actual baud rate	Baud rate error
[Kbps]	hex	dec	hex	dec	[Kbps]	(%)
0.3	$0A_H$	10	$0D_H$	13	0.30	0.15
0.6	09_H	9	$0D_H$	13	0.60	0.16
1.2	08_H	8	$0D_H$	13	1.20	0.16
2.4	07_H	7	$0D_H$	13	2.40	0.16
4.8	06_H	6	$0D_H$	13	4.81	0.16
9.6	05_H	5	$1A_H$	26	9.62	0.16
19.2	05_H	5	$0D_H$	13	19.23	0.16
31.25	04_H	4	10_H	16	31.25	0.00
38.4	04_H	4	$0D_H$	13	38.46	0.16
57.6	00_H	0	$8B_H$	139	57.55	0.08
76.8	03_H	3	$0D_H$	13	76.92	0.16
115.2	00_H	0	45_H	69	115.94	0.64
153.6	02_H	2	$0D_H$	13	153.85	0.16
312.5	01_H	1	$0D_H$	13	307.69	1.54

16.2 UARTD Functional Outline

- Baud Rate Generator for flexible configuration of the baud rate
- Full-duplex communication:
  - UARTD receive data register n (UDnRX)
  - UARTD transmit data register n (UDnTX)
- 2-pin configuration:
  - TXDDn: Transmit data output pin
  - RXDDn: Receive data input pin
- Reception error and status output function
  - Parity error
  - Framing error
  - Overrun error
  - Data consistency error
  - SBF receive error
- Interrupt sources: 3
  - Reception complete interrupt (INTUDnR):

    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 (INTUDnT):

    This interrupt occurs upon transfer of transmit data from the transmit buffer register to the shift register in the transmission enabled status.

  - Communication error interrupt (INTUDnS):

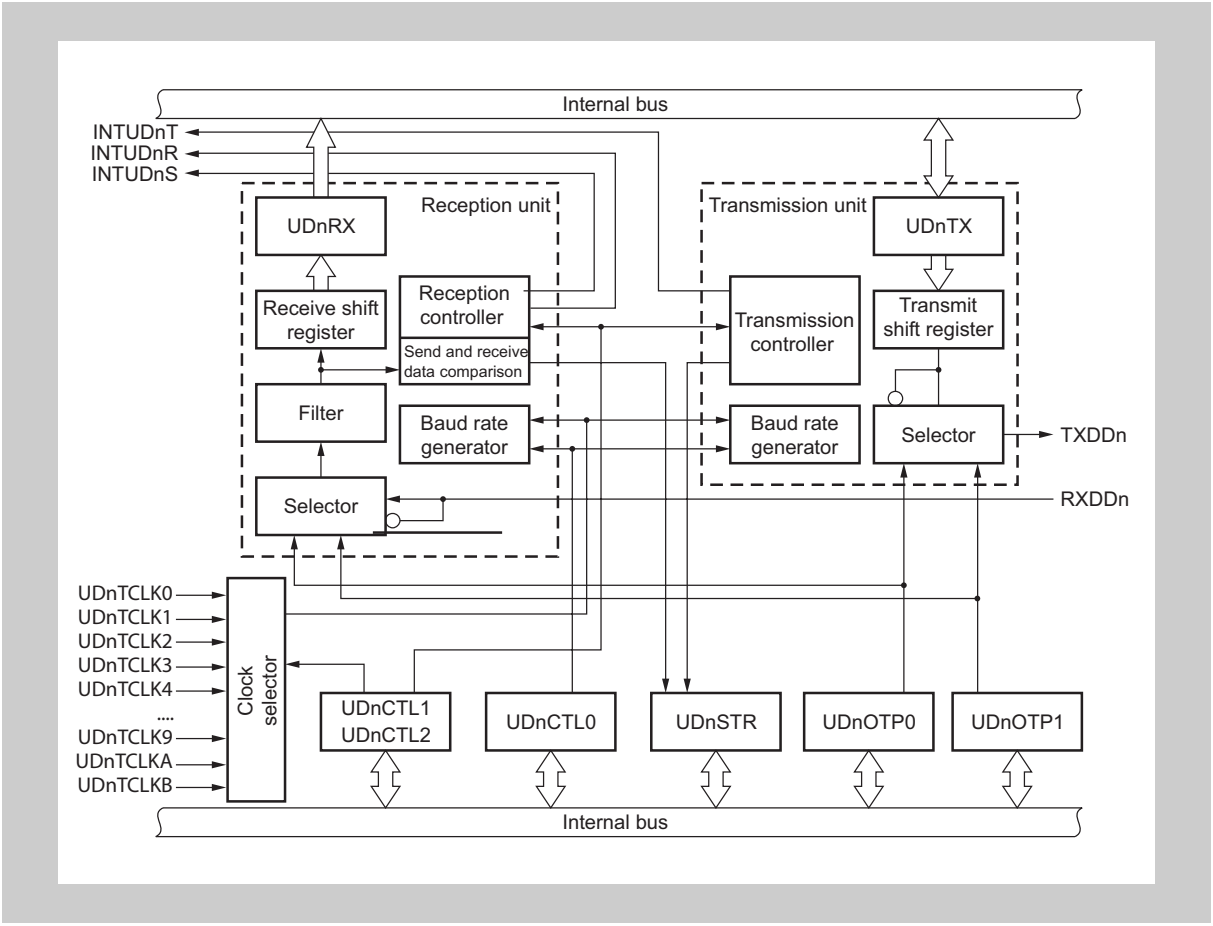
    This interrupt occurs upon reception of erroneous data and the data consistency and SBF reception during LIN communication.

- Character length: 7, 8 bits
- Parity function: odd, even, 0, none
- Transmission stop bit: 1, 2 bits
- On-chip dedicated baud rate generator for flexible baud rate generation
- MSB-/LSB-first transfer selectable
- Transmit/receive data inverted input/output option
- 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
- SBF reception can be detected during data communication.
- Bus monitor function to keep data consistency of the transmit data
- DMA support

16.3 UARTD Block Diagram

The block diagram of the UARTDn is shown below.

Figure 16-1 Block diagram of Asynchronous Serial Interface UARTDn



Note For the configuration of the baud rate generator, see *Figure 16-11 on page 398*.

16.4 UARTD Registers

Register addresses

All UARTDn register addresses are given as address offsets to the individual base addresses <base> of each UARTDn.

The <base> addresses of each UARTDn are listed in the following table:

Table 16-4 Register <base> addresses of UARTDn

UARTDn	<base> address
UARTD0	FFFF FA00$_H$
UARTD1	FFFF FA10$_H$

The UARTDn are controlled and operated by means of the following registers.

Table 16-5 UARTDn registers overview

Register name	Shortcut	Address
UARTDn control register 0	UDnCTL0	<base>
UARTDn control register 1	UDnCTL1	<base> + 1$_H$
UARTDn control register 2	UDnCTL2	<base> + 2$_H$
UARTDn option register 0	UDnOPT0	<base> + 3$_H$
UARTDn status register	UDnSTR	<base> + 4$_H$
UARTDn option register 1	UDnOPT1	<base> + 5$_H$
UARTDn receive register	UDnRX	<base> + 6$_H$
UARTDn transmit register	UDnTX	<base> + 7$_H$

16.4.1 UDnCTL0 - UARTDn control register 0

The UDnCTL0 register is an 8-bit register that controls the UARTDn 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
UDnPWR	UDnTXE	UDnRXE	UDnDIR	UDnPS1	UDnPS0	UDnCL	UDnSL
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

UDnPWR	UARTDn operation control
0	Disable UARTDn operation (UARTDn reset asynchronously)
1	Enable UARTDn operation

The UARTDn operation is controlled by the UDnPWR bit. The TXDDn pin output is fixed to high level by clearing the UDnPWR bit to 0 (fixed to low level if UDnOPT0.UDnTDL bit = 1).

UDnTXE	Transmission operation control
0	Disable transmission operation
1	Enable transmission operation

- To start transmission, set the UDnPWR bit to 1 and then set the UDnTXE bit to 1.To stop transmission clear the UDnTXE bit to 0 and then UDnPWR bit to 0.
- To initialize the transmission unit, clear the UDnTXE bit to 0, wait for two cycles of the base clock, then set the UDnTXE bit to 1 again.

UDnRXE	Transmission operation control
0	Disable transmission operation
1	Enable transmission operation

- To enable reception, set the UDnPWR bit to 1 and then set the UDnRXE bit to 1. The reception is enabled after the UDnRXE bit is set to 1 and two cycles of base clock have passed.
  The rising edge detection of the RXDD pin is enabled after the UDnRXE bit is set to 1 and four cycles of the base clock have passed.
- To stop reception, clear the UDnRXE bit to 0 and then UDnPWR bit to 0.
- To initialize the reception unit, clear the UDnRXE bit to 0, wait for two periods of the base clock, and then set the UDnRXE bit to 1 again.

UDnDIR	Transfer direction selection
0	MSB-first transfer
1	LSB-first transfer

UDnPS1	UDnPS0	Transmit/Receive parity
0	0	No parity
0	1	Zero (0) parity
1	0	Odd parity
1	1	Even parity

- This register is rewritten only when the UDnPWR bit = 0 or when the UDnTXE bit = UDnRXE bit = 0.
- If “No parity” or “Zero parity” is selected during reception, a parity check is not performed. Therefore, the UDnSTR.UDnPE bit will not be set if a parity error does occur, and no interrupt error will be output.
- When transmission and reception are performed in the LIN format, clear the UDnPS1 and UDnPS0 bits to 00.

Note For details of parity, see *Section 16.6.11, Parity types and operations* on page 396.

UDnCL	Data character length
0	7 bits
1	8 bits

This register can be rewritten only when the UDnPWR bit = 0 or the UDnTXE bit = UDnRXE bit = 0.

UDnSL	Number
0	1-bit
1	2-bit

This register can be rewritten only when the UDnPWR bit = 0 or the UDnTXE bit = UDnRXE bit = 0.

16.4.2 UDnCTL1 - UARTDn control register 1

The UDnCTL1 register is an 8-bit register that selects the UARTDn base clock.

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.

Caution Clear the UDnCTL0.UDnPWR bit to 0 before rewriting the UDnCTL1 register.

7	6	5	4	3	2	1	0
0	0	0	0	UDnCKS3	UDnCKS2	UDnCKS1	UDnCKS0
R	R	R	R	R/W	R/W	R/W	R/W

UDnCKS3	UDnCKS2	UDnCKS1	UDnCKS0	Input
0	0	0	0	UDnTCKI0
0	0	0	1	UDnTCKI1
0	0	1	0	UDnTCKI2
0	0	1	1	UDnTCKI3
0	1	0	0	UDnTCKI4
0	1	0	1	UDnTCKI5
0	1	1	0	UDnTCKI6
0	1	1	1	UDnTCKI7
1	0	0	0	UDnTCKI8
1	0	0	1	UDnTCKI9
1	0	1	0	UDnTCKIA
1	0	1	1	UDnTCKIB
Setting prohibited				none

16.4.3 UDnCTL2 - UARTDn control register 2

The UDnCTL2 register is an 8-bit register that selects the baud rate (serial transfer speed) clock of UARTDn.

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

Address <base> + 2_H

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

Caution Clear the UDnCTL0.UDnPWR bit to 0 or clear the UDnTXE and UDnRXE bits to 00 before rewriting the UDnCTL2 register.

7	6	5	4	3	2	1	0
UDnBRS7	UDnBRS6	UDnBRS5	UDnBRS4	UDnBRS3	UDnBRS2	UDnBRS1	UDnBRS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

UDnBRS7	UDnBRS6	UDnBRS5	UDnBRS4	UDnBRS3	UDnBRS2	UDnBRS1	UDnBRS0	Default [k]	Serial clock
0	0	0	0	0	0	x	x	x	setting prohibited
0	0	0	0	0	1	0	0	4	f_{UCLK} / 4
0	0	0	0	0	1	0	1	5	f_{UCLK} / 5
0	0	0	0	0	1	1	0	6	f_{UCLK} / 6
....									
1	1	1	1	1	1	0	0	252	f_{UCLK} / 252
1	1	1	1	1	1	0	1	253	f_{UCLK} / 253
1	1	1	1	1	1	1	0	254	f_{UCLK} / 254
1	1	1	1	1	1	1	1	255	f_{UCLK} / 255

Note f_{UCLK}: Clock frequency selected by UDnCTL1.UDnCKS[3:0]

16.4.4 UDnOPT0 - UARTDn option control register 0

The UDnOPT0 register is an 8-bit register that controls the serial transfer operation of the UARTDn 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
UDnSRF	UDnSRT	UDnSTT	UDnSLS2	UDnSLS1	UDnSLS0	UDnTDL	UDnRDL
R	R/W	R/W	R/W	R/W	R/W	R/W	R/W

UDnSRF	SBF reception flag
0	• After normal reception of SBF • When UDnCTL0.UDnPWR = UDnCTL0.UDnRXE = 0
1	• During reception of SBF • After SBF reception error occurs

Note
- The UDnSRF bit is held at 1 when an SBF reception error occurs. Once a new SBF is received correctly, the UDnSRF bit is cleared to 0. The bit cannot be cleared by writing to it.
- The validity of the UDnSRF bit is affected by the setting of the UDnSRS bit. If the UDnSRS bit = 0, an SBF cannot be correctly detected. Hence, the UDnSRF bit could incorrectly indicate an SBF reception error. If LIN communication is being executed and the UDnSRF bit is being used, then the UDnSRS bit should be set to 1.

UDnSRT	SBF reception mode trigger
0	No trigger
1	SBF reception trigger

- During LIN communication, the UDnSRT bit is used to put the UARTD into an SBF reception mode. In this mode, the UARTD analyzes the next data frame to confirm it is a valid SBF. If UDnSRS = 0, a reception complete interrupt (INTUDnR) is generated. Otherwise, if UDnSRS = 1, the SBF reception success flag (UDnSSF) is set and a status interrupt (INTUDnS) is generated. Also the error detection for overrun, parity, and framing is not performed, and the data in the receive shift register is not transferred to the receive data register (UDnRX).
- Set the UDnSRT bit after setting the UDnCTL0.UDnPWR bit and UDnCTL0.UDnRXE bit = 1.
- After the UDnSRT bit is set, re-setting of the UDnSRT bit is disabled until a successful SBF is received, UDnSRF is cleared, and the reception complete interrupt (INTUDnR) is generated.
- If the UDnSRT is set during an SBF reception, the SBF cannot be received. Hence, other reception operations are not performed until the next SBF reception is successful.
- This bit is always read as “0”.

UDnSTT	SBF transmission trigger
0	No trigger
1	SBF transmission trigger

- The UDnSTT bit is used to initiate the transmission of an SBF. When a “1” is written to this bit, an SBF transmission is started.
- Set the UDnSTT bit after setting the UDnPWR bit = UDnTXE bit = 1.
- This bit is always read as “0”.

UDnSLS2	UDnSLS1	UDnSLS0	SBF transmit length selection
1	0	1	13-bit output (default)
1	1	0	14-bit output
1	1	1	15-bit output
0	0	0	16-bit output
0	0	1	17-bit output
0	1	0	18-bit output
0	1	1	19-bit output
1	0	0	20-bit output

- These bits can only be written when the UDnPWR = 0 or UDnTXE = 0.

UDnTDL	Transmit data level bit
0	Normal output of transfer data
1	Inverted output of transfer data

Note: The UDnTDL bit control inverts the TXDDn output level regardless of the values of the UDnPWR and UDnTXE bits. Therefore, if the UDnTDL bit is set to 1 while the operation is disabled, the TXDDn outputs the low level.

UDnRDL	Receive data level bit
0	Normal input of transfer data
1	Inverted input of transfer data

- The input level of the TXDDn pin can be inverted using the UDnTDL bit.
- This bit can only be written when UDnPWR = 0 or UDnRXE bit = 0.

Note 1. To cancel the SBF reception enable status without receiving the SBF, set the UDnPWR bit = 0 or UDnRXE bit = 0.

2. The confirmation method of SBF receive completion while the UDnSRT bit is set depends on the values of the SBF reception mode selection bit (UDnSRS). If the UDnSRS bit is cleared to 0, it is confirmed by receive completion interrupt which is detected after the setting of the SBF reception trigger bit.
   If the UDnSRS bit is set to 1, it is confirmed by whether the SBF receive success flag (UDnSSF) is 1 when status interrupt is detected after the setting of the SBF reception trigger bit. It can also be confirmed by the UDnSRF bit = 0 after the receive interrupt or the status interrupt is detected. In any case, after the SBF reception is completed, the UART normal reception is operated at the next reception.

3. Data transmission while UDnDCS bit = 1 during UDnSRF bit = 1 is prohibited. However, the SBF transmission is enabled.

Caution Before starting the SBF transmission by UDnOPT0.UDnSTT=1 make sure that no data transfer is ongoing, that means check that UDnSTR.UDnTSF=0.

16.4.5 UDnOPT1 - UARTDn option control register 1

The UDnOPT1 register is an 8-bit register that controls the serial transfer operation of the UARTDn register.

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	UDnSRS	UDnDCS
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

UDnSRS	SBF reception mode selection bit
0	SBF cannot be detected
1	SBF can be detected

Note: **1.** This bit should only be set when the LIN communication is used. Otherwise set this bit to 0.

2. When this bit is set to 1, it is necessary to set UD0DCS to 1.

UDnDCS	Data consistency check selection bit
0	Data consistency is not checked
1	Data consistency is checked

When data is transmitted using the LIN protocol, this bit selects the handling of the consistency checking of data. When UDNDCS = 1 the transmitted data and received data are compared and the mismatch is detected. In that case a status interrupt request signal (UDTIS) is generated.

Note: **1.** This bit should only be set when the LIN communication is used. Otherwise set this bit to 0.

2. When this bit is used, the data bit length doesn't prohibits the eight bit fixation and the addition of the parity bit.

16.4.6 UDnSTR - UARTDn status register

The UDnSTR register is an 8-bit register that displays the UARTDn transfer status and reception error contents.

The UDnSTRn register consists of flags indicating the error contents when a reception error occurs, the inconsistency between transmit and receive data and successful SBF reception during LIN communication. 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 UDnSTR register.

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
UDnSTR register	• Reset • UDnCTL0.UDnPWR = 0
UDnSSF	• UDnCTL0.UDnRXE = 0 • UDnOPT1.UDnSRS = 0
UDnDCE	• UDnCTL0.UDnRXE = 0 • UDnOPT1.UDnDCS = 0
UDnTSF bit	• UDnCTL0.UDnTXE = 0
UDnPE, UDnFE, UDnOVE bits	• 0 write • UDnCTL0.UDnRXE = 0

Note To clear the status flag, use a 1-bit manipulation instruction or write the inverted value of the read value using a 8-bit manipulation instruction to clear all bits together.

7	6	5	4	3	2	1	0
UDnTSF	0	0	UDnSSF	UDnDCE	UDnPE	UDnFE	UDnOVE
R	R/W	R/W	R/W	R/W	R/W	R/W	R/W

UDnTSF	Transfer status flag
0	Following conditions - When the UDnPWR bit = 0 or the UDnTXE bit = 0 has been set. - When, following transfer completion, there was no next data transfer from UDnTX register - When there is no next transmit data at the UDnTX bit after the SBF transmission has been completed
1	Following conditions - Write to UDnTXB bit - When the SBF transmission trigger bit (UDnSST) is set

The UDnTSF bit is always 1 when performing continuous transmission. When initializing the transmission unit, check that the UDnTSF bit = 0 before performing initialization. The transmit data is not guaranteed when initialization is performed while the UDnTSF bit = 1.

During the communication the UDnTSF bit is cleared after 2 clocks.

UDnSSF	SBF receive successful flag
0	When the UDnPWR bit = 1or the UDnRXE bit = 0 or the UDnSRS bit = 0 or the UDnSSF bit = 0 has been set
1	When a consecutive low level (SBF) of 11 bits or more is received and the SBF reception mode bit UDnSRS has been set.

When the SBF receive mode selection bit is set in LIN communication mode, it is necessary to read this bit by the status interrupt processing and to confirm the beginning of a new frame slot.

This bit is cleared by writing a “0” to it. Writing a “1” has no effect.

UDnDCE	Data consistency error flag
0	No data consistency error
1	A data consistency error has occurred

The transmit data is compared with the receive data when data is transmitted in LIN communication mode. When the transmit data does not match the receive data, this is a data consistency error and this bit is set to 1.

This bit is cleared by writing a “0” to it. Writing a “1” has no effect.

UDnPE	Parity error flag
0	No parity error
1	A parity error has occurred

A parity error occurs when the parity of the received data and the parity bit do not match. The parity (none, 0, E, O) is determined by the UDnPS1/UDnPS0 bits

This bit is cleared by writing a “0” to it. Writing a “1” has no effect.

UDnFE	Framing error flag
0	No framing error
1	A framing error has occurred

- Only the first bit of the receive data stop bits is checked, regardless of the value of the UDnCTL0.UDnSL bit.

This bit is cleared by writing a “0” to it. Writing a “1” has no effect.

UDnOVE	Overrun error flag
0	No overrun error
1	An overrun error has occurred

- When an overrun error occurs, the data is discarded without the next receive data being written to the receive buffer.
- The UDnOVE 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

16.4.7 UDnRX - UARTDn receive data register

The UDnRX register is an 8-bit buffer register that stores parallel data converted by the receive shift register. 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 UARTDn receive shift register to the UDnRX register in synchronization with the completion of shift-in processing of 1 frame.

The data stored in the receive shift register is transferred to the UDnRX 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 UDnRX register and the MSB always becomes 0. During MSB-first reception, the receive data is transferred to bits 7 to 1 of the UDnRX register and the LSB always becomes 0.

When an overrun error (UDnOVE) occurs, the receive data at this time is not transferred to the UDnRX register and is discarded.

Transfer to the UDnRX register also causes the reception complete interrupt request signal (INTUDnR) to be output.

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 UDnRX register can be set to FFH by clearing the UDnCTL0.UDnPWR bit to 0.

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

16.4.8 UDnRXS - UARTDn receive shift register

This is a shift register used to convert the serial data input to the RXDDn pin into parallel data. Upon reception of 1 byte of data and detection of the stop bit, the receive data is transferred to the UDnRX register.

This register cannot be manipulated directly.

16.4.9 UDnTX - UARTDn transmit data register

The UDnTX register is an 8-bit transmit data buffer. Transmission starts when transmit data is written to the UDnTX register. When data can be written to the UDnTX register (when data of one frame is transferred from the UDnTX register to the UARTDn transmit shift register), the transmission enable interrupt request signal (INTUDnT) is generated.

If the next data is written before the transmission is completed the continuous transmission is enabled.

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.

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

16.4.10 UDnTXS - UARTDn transmit shift register

The transmit shift register is a shift register used to convert the parallel data transferred from the UDnTX register into serial data.

When 1 byte of data is transferred from the UDnTX register, the shift register data is output from the TXDDn pin.

This register cannot be manipulated directly.

16.5 Interrupt Request Signals

The following three interrupt request signals are generated from UARTDn.

- Reception complete interrupt request signal (INTUDnR)
- Status interrupt request signal (INTUDnS)
- Transmission enable interrupt request signal (INTUDnT)

16.5.1 Reception complete interrupt request signal (INTUDnR)

A reception complete interrupt request signal is output when data is shifted into the receive shift register and transferred to the UDnRX register in the reception enabled status.

In case of erroneous reception, the status interrupt INTUDnS is generated instead of INTUDnR.

No reception complete interrupt request signal is generated in the reception disabled status.

16.5.2 Status interrupt request signal (INTUDnS)

A status interrupt request is generated if an error condition occurred during reception, as reflected by UDnSTR.UDnPE (parity error flag), UDnSTR.UDnFE (framing error flag), UDnSTR.UDnOVE (overrun error flag), the data is not consistent between data transmit and data reception.

When the SBF reception mode selection bit is set in LIN communication mode (UDnSRS bit = 1), the status interrupt request signal is generated when a consecutive low level (SBF) of 11 bits or more is received.

16.5.3 Transmission enable interrupt request signal (INTUDnT)

If transmit data is transferred from the UDnTX register to the UARTDn transmit shift register with transmission enabled, the transmission enable interrupt request signal is generated.

16.6 Operation

16.6.1 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 UDnCTL0 register.

Moreover, control of UART output/inverted output for the TXDDn bit is performed using the UDnOPT0.UDnTDL bit.

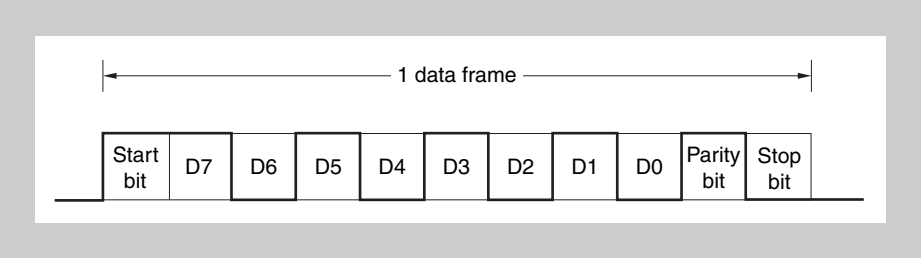
- Start bit 1 bit
- Character bits 7 bits/8 bits
- Parity bit Even parity/odd parity/0 parity/no parity
- Stop bit 1 bit/2 bits

16.6.2 UARTD transmit/receive data format

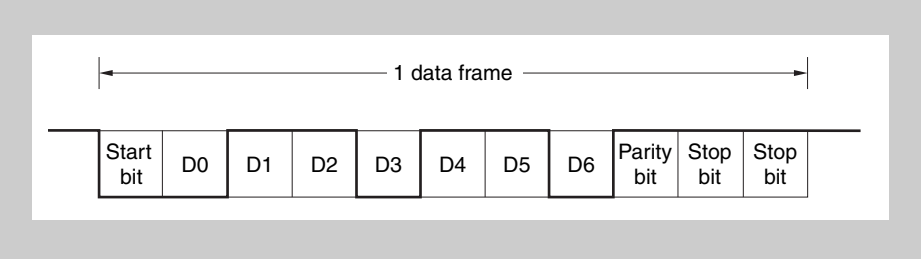
(1) 8-bit data length, LSB first, even parity, 1 stop bit, transfer data: 55_H

(2) 8-bit data length, MSB first, even parity, 1 stop bit, transfer data: 55_H

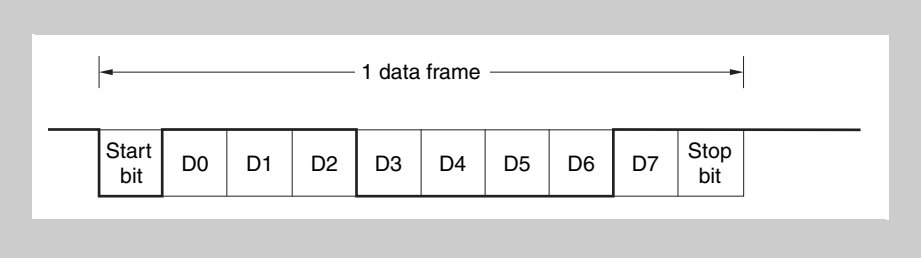
(3) 8-bit data length, MSB first, even parity, 1 stop bit, transfer data: 55_H, TXDDn inversion



(4) 7-bit data length, LSB first, odd parity, 2 stop bits, transfer data: 36_H



(5) 8-bit data length, LSB first, no parity, 1 stop bit, transfer data: 87_H



16.6.3 SBF transmission/reception format

The UARTD 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 communication 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.

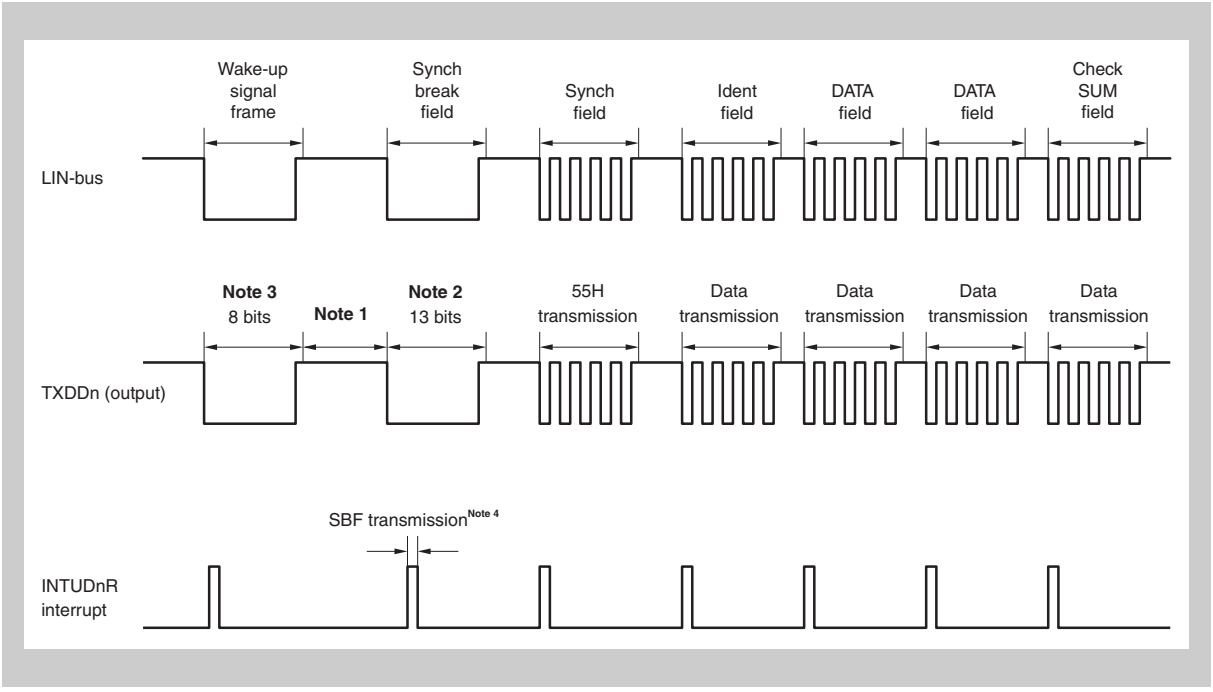
Normally, the LIN master is connected to a network such as CAN (Controller Area Network).

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 16-2 and *Figure 16-3* outline the transmission and reception manipulations of LIN.

Figure 16-2 LIN transmission manipulation outline

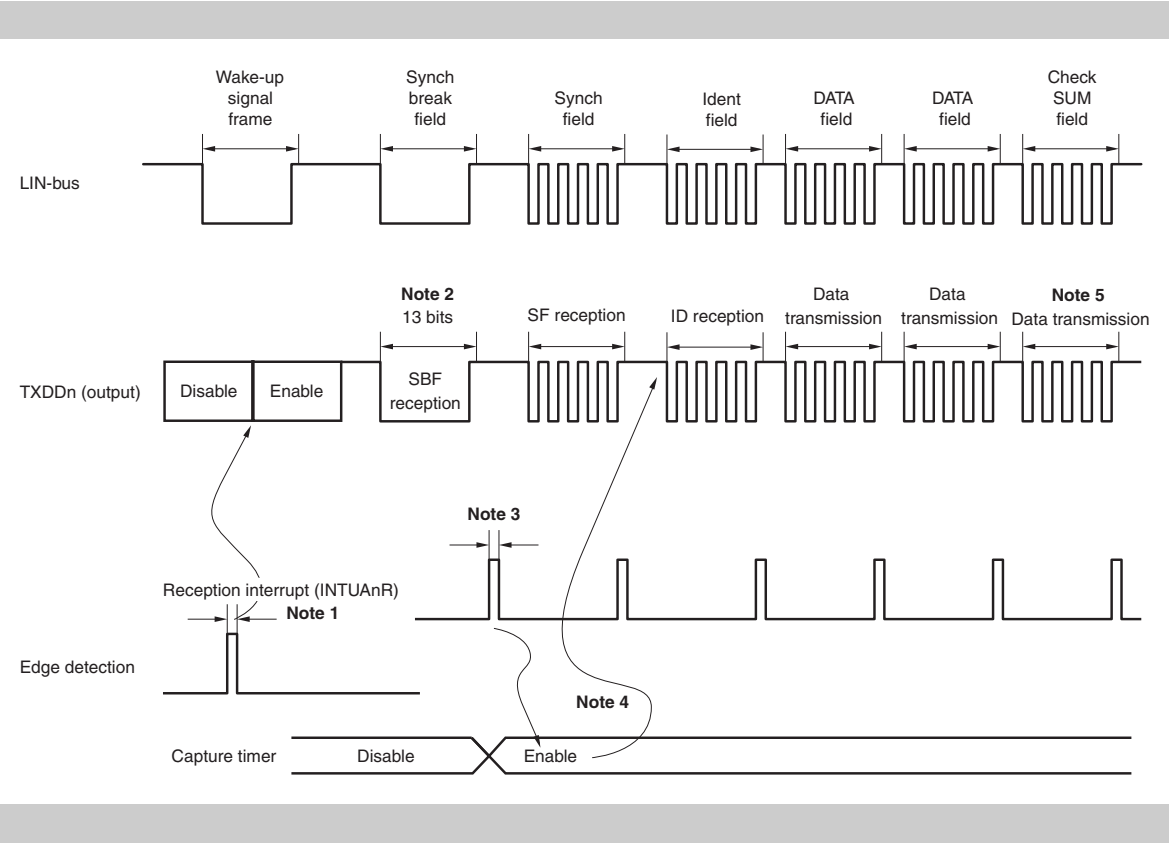


Note

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 UDnOPT0.UDnSBL2 to UDnOPT0.UDnSBL0 bits. If even finer output width adjustments are required, such adjustments can be performed using the UDnCTLn.UDnBRS7 to UDnCTLn.UDnBRS0 bits.
3. 80_H transfer in the 8-bit mode is substituted for the wakeup signal frame.

4. A transmission enable interrupt request signal (INTUDnT) is output at the start of each transmission. The INTUDnT signal is also output at the start of each SBF transmission.

Figure 16-3 LIN reception manipulation outline



Note 1. The wakeup signal is sent by the pin edge detector, UARTDn is enabled, and the SBF reception mode is set.

2. Upon detection of the SBF reception of 11 or more bits, normal SBF reception end is judged. When the SBF reception mode selection bit (UDnSRS) is set to "0", the receive completion interrupt request signal (INTUDnR) is generated, and when the UDnSRS is set to "1", the status interrupt request signal (INTUDnS) is generated. 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. When SBF reception ends normally, if the SBF reception mode selection bit (UDnSRS) is "0", the receive completion interrupt request signal (INTUDnR) is generated, and if the UDnSRS is "1", the status interrupt request signal (INTUDnS) is generated and the SBF reception success flag (UDnSSF) is set. If the SBF reception trigger bit (UDnSRT) is "1", the error detection for the overrun, parity, and framing (UDnOVE, UDnPE, UDnFE) is not performed during the SBF reception. Moreover, the data transfer from the receive shift register to the receive data register (UDnRX) is not performed, either. At this time, the UDnRX holds the prior value.

4. The RXDDn 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 UDnCTL2 register obtained by correcting the baud rate error after dropping UARTD enable is set again, causing the status to become the reception status.

5. Check-sum field distinctions are made by software. UARTDn is initialized following CSF reception, and the processing for setting the SBF reception mode again is performed by software. When the UDnSRS bit = 1, the SBF reception can be performed automatically without setting to the SBF reception mode again.

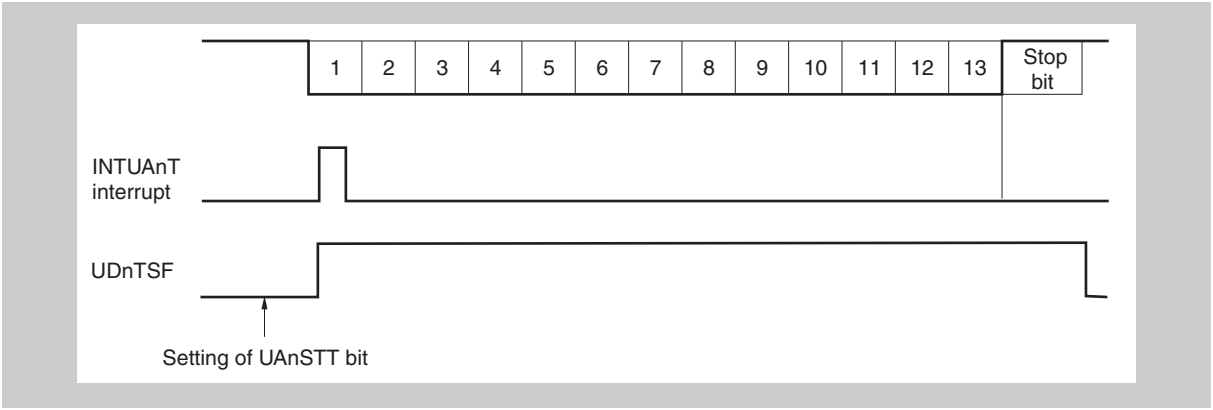
16.6.4 SBF transmission

When the UDnCTL0.UDnPWR bit = UDnCTL0.UDnTXE bit = 1, the transmission enabled status is entered, and SBF transmission is started by setting (to 1) the SBF transmission trigger (UDnOPT0.UDnSTT bit).

Thereafter, a low level the width of bits 13 to 20 specified by the UDnOPT0.UDnSLS2 to UDnOPT0.UDnSLS0 bits is output. A transmission enable interrupt request signal (INTUDnT) is generated upon SBF transmission start. Following the end of SBF transmission, the UDnSTT 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 UDnTX register, or until the SBF transmission trigger (UDnSTT bit) is set.

Figure 16-4 SBF transmission



16.6.5 SBF reception

The reception enabled status is achieved by setting the UDnCTL0.UDnPWR bit to 1 and then setting the UDnCTL0.UDnRX bit to 1.

The SBF reception wait status is set by setting the SBF reception trigger (UDnOPT0.UDnSTR bit) to 1.

In the SBF reception wait status, similarly to the UART reception wait status, the RXDDn pin is monitored and start bit detection is performed.

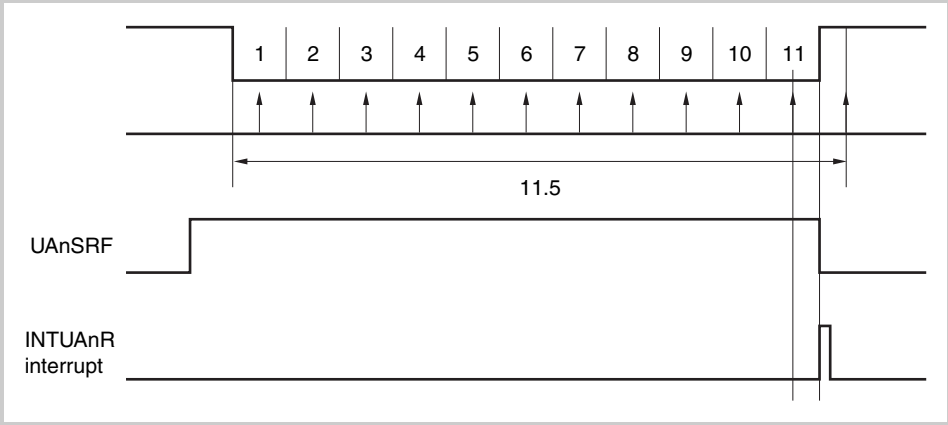
Following detection of the start bit, reception is started and the internal counter counts up according to the set baud rate.

When a high level is received and if the SBF width is 11 or more bits, when SBF receiving mode selection bit (UDnSRS) is “0” the reception completion interrupt request signal (INTUDnR) is generated. When the UDnSRS bit is “1” the SBF reception success flag (UDnSSF) is set at the same time as generating a status interrupt request signal (INTUDnS). The UDnOPT0.UDnSRF bit is automatically cleared and SBF reception ends. Error detection for the UDnSTR.UDnOVE, UDnSTR.UDnPE, and UDnSTR.UDnFE bits is suppressed and UART communication error detection processing is not

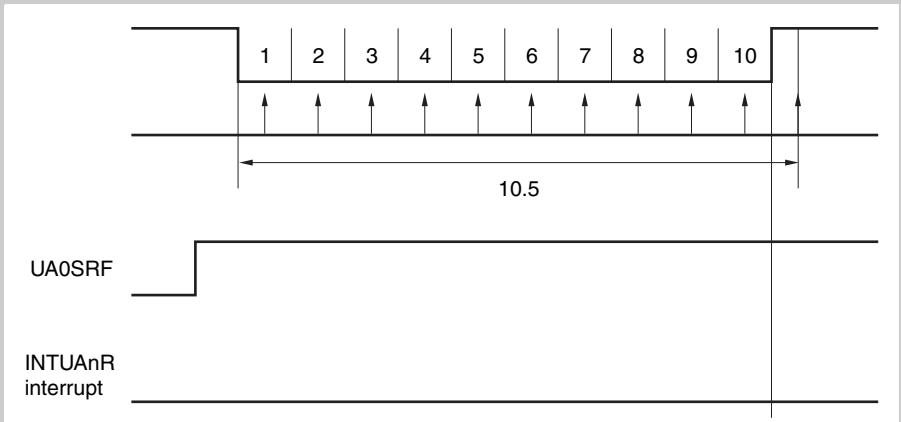
performed. Moreover, data transfer of the UARTDn reception shift register and UDnRX register is not performed and FFH, the initial value, is held. If the SBF width is 10 or fewer bits, reception is terminated as error processing without outputting an interrupt, and the SBF reception mode is returned to. The UDnSRF bit is not cleared at this time.

The SBF mode can be selected between a single SBF receive mode and an any time SBF receive mode in the UDnOPT1 register (UDnOPT1.UDnSRS). The status of a successful reception of the SBF is shown y the UDnOPT1.UDnSRS bit in the UDnOPT1 register.

(1) Normal SBF reception (detection of stop bit in more than 10.5 bits)



(2) SBF reception error (detection of stop bit in 10.5 or fewer bits)



Note The UDnSRF bit is reset by setting the UDnSRT bit to "1", and cleared by normal SBF reception.

16.6.6 Data consistency check

The UARTD incorporates a data consistency check function to detect a mismatch between the transmit data written to transmit register (UDnTX) and the data on the bus when the device operates in master mode.

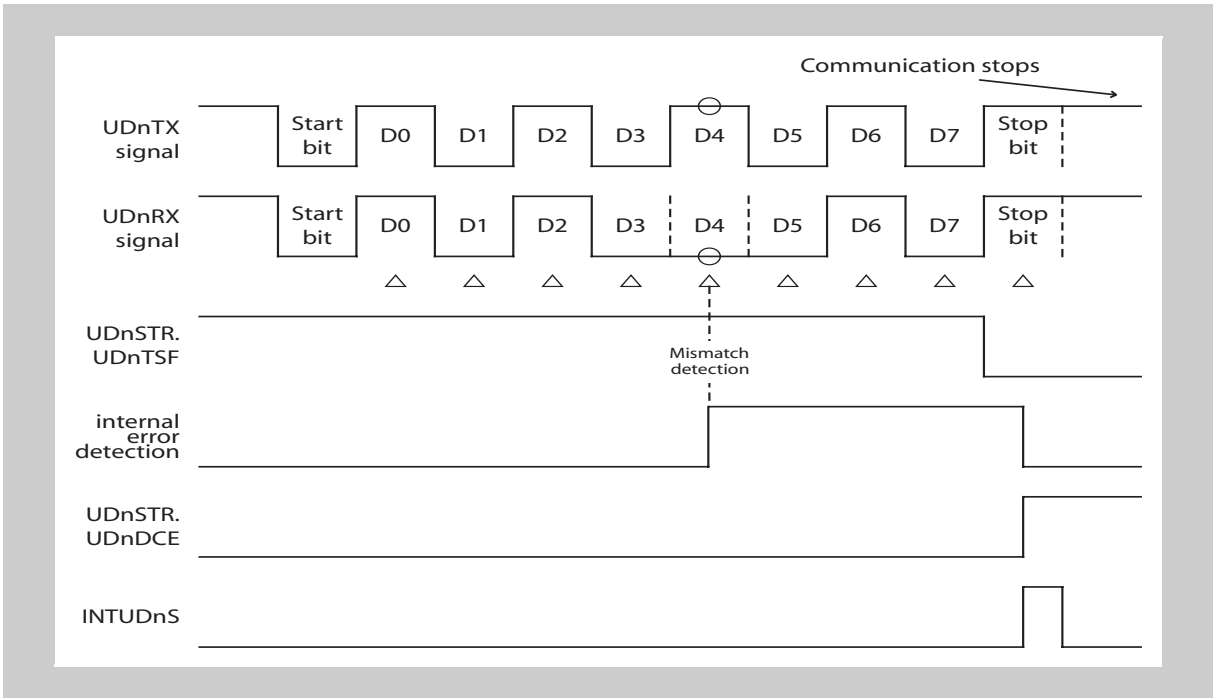
The data consistency is checked by comparing the transmit data in the transmit register (UDnTX) and the receive data in the receive register (UDnRX). In case of a mismatch the data consistency error flag (UDnSTR.UDnDCE) is set and a status interrupt request (INTUDnS) occurs.

The consistency check of data is not done in reception mode.

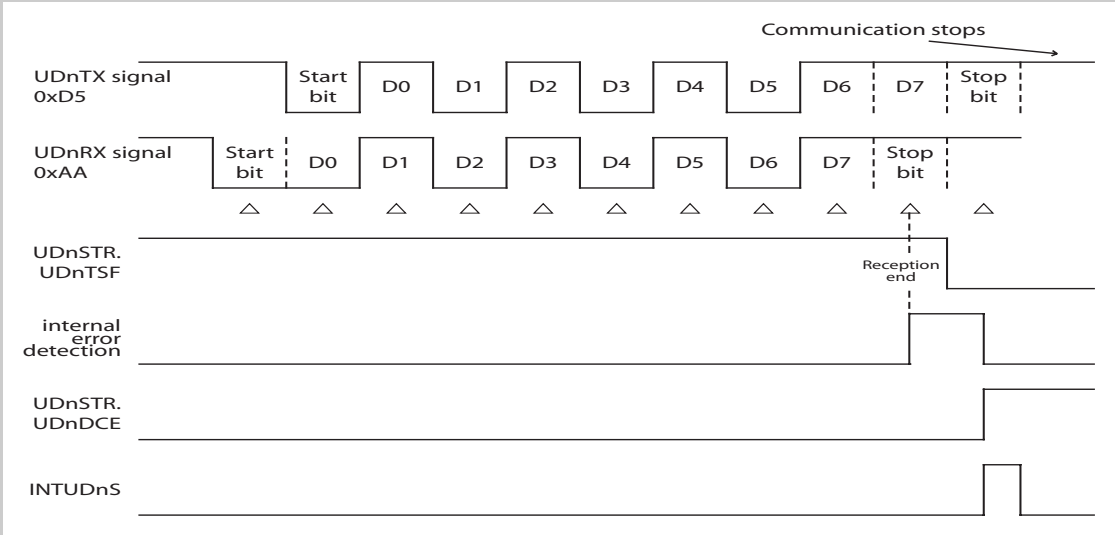
The consistency check of the send data and the input data terminal level is done even if the reception is disabled (UDnRXE = 0) during sending. In that case also the reception completion interrupt request signal (INTUDnR), the UDnSSF, UDnFE, UDnOVE and the status interrupt request signal (INTUDnS) will not be generated as well. Receive data does not need to be read.

Refer to *Section 16.4.6, UDnSTR - UARTDn status register* on page 380 for details.

(1) Timing example of data consistency error (UDnSRF = 0)



(2) Timing example of data consistency error when there is a delay between transmit and receive operation

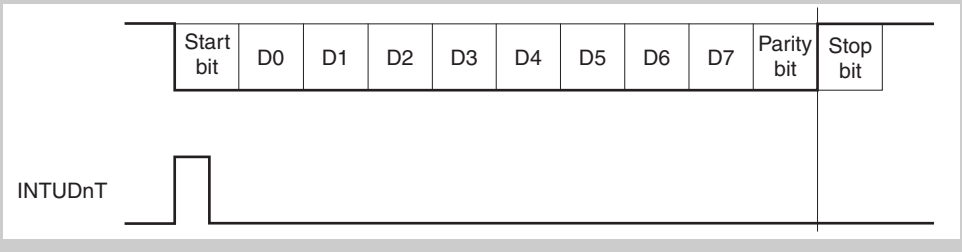


16.6.7 UART transmission

First, set the transmission enabled status by performing the following procedures.

- Specify the operation clock by the UARTD control register 1 (UDnCTL1)
- Specify the baud rate by the UARTD control register 2 (UDnCTL2)
- Specify the output logic level by the UARTD option control register 0 (UDnOPT0).
- Specify the transmit destination, parity, data character length, stop bit length by the UARTD control register 0 (UDnCTL0).
- Set the power bit and the transmission enabled bit (UDnPWR = 1, UDnTXE = 1)

Write of the transmit data to the transmission buffer register (UDnTX) starts transmission. The data which is saved in the UDnTX register is transferred to the transmit shift register (UDnTXS). Then, the start bit, parity bit, and stop bit are added and the data is output serially from the TXDDn to the data. Moreover, at the timing that the transfer to UDnTXS of the data stored in UDnTX is completed, a transmission interrupt request signal (INTUDnT) is generated. Once INTUDnT is generated, the next data can be written to UDnTX.



Note LSB first

16.6.8 Continuous transmission procedure

A continuous transmissions becomes enabled by writing the next transmit data after the transmission request interrupt (INTUDnT) is generated.

Caution If the value is written to the UDnTX register before the INTUDnT is generated, the transmit data set before is overwritten by the new transmit data.

To initialize the transmission unit, confirm that the transmission status flag is reset (UDnTSF = 0). If the initialization is performed when the UDnTSF = 1, the transmission is aborted.

During continuous transmission execution, there is a 2 clock interval after the transmission of the stop bit to the start of the next start bit transmission.

Figure 16-5 Continuous transmission processing flow

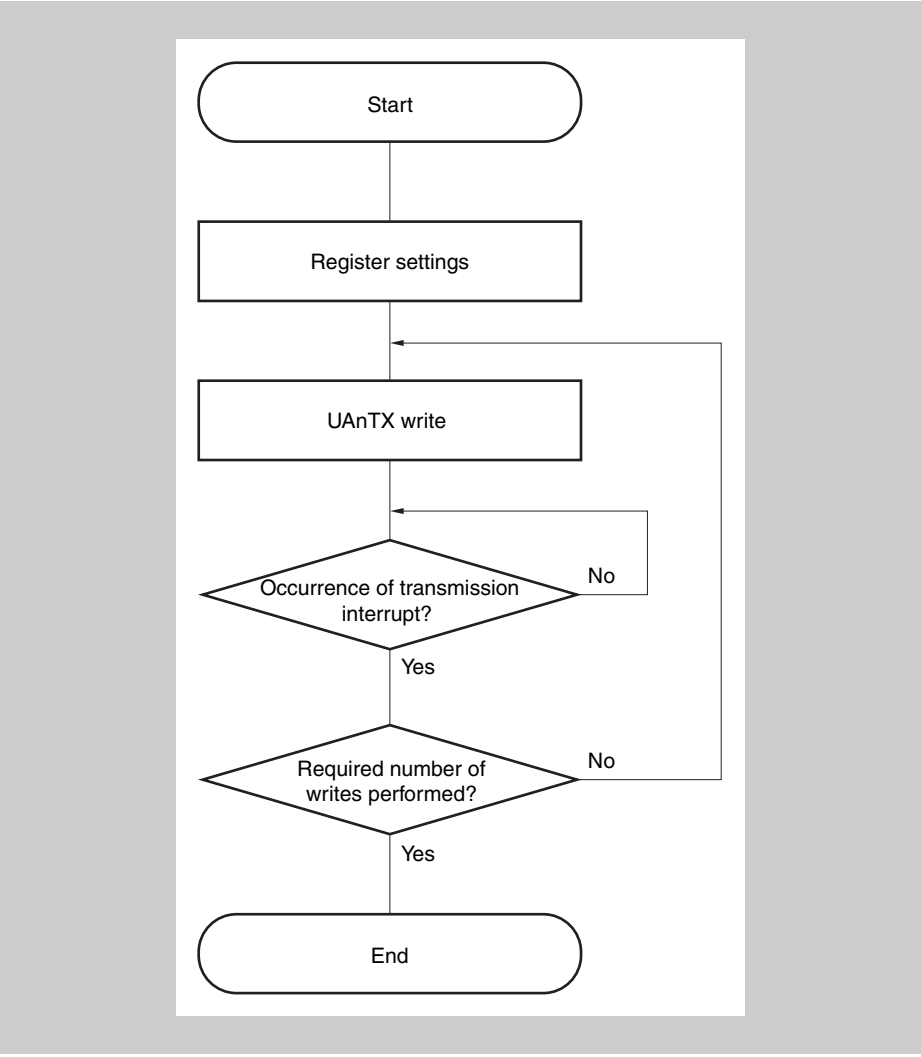


Figure 16-6 Continuous transmission operation timing —transmission start

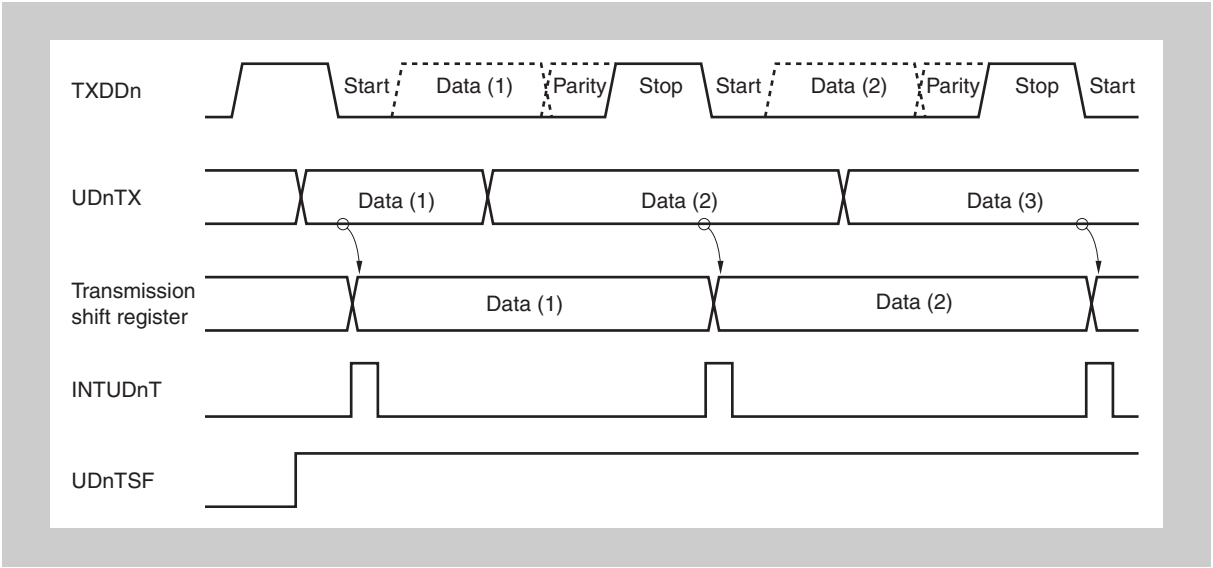
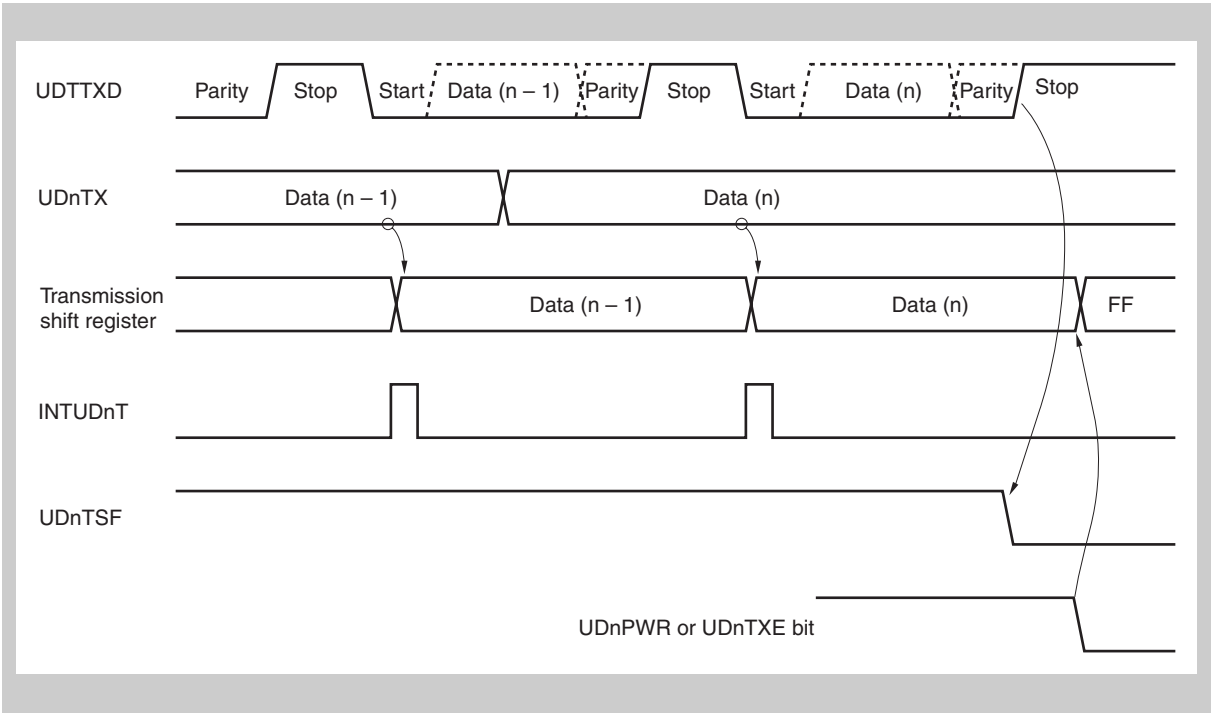


Figure 16-7 Continuous transmission operation timing—transmission end



16.6.9 UART reception

First, set the reception enabled status by the next operations to monitor the RXDDn input and perform the start bit detection.

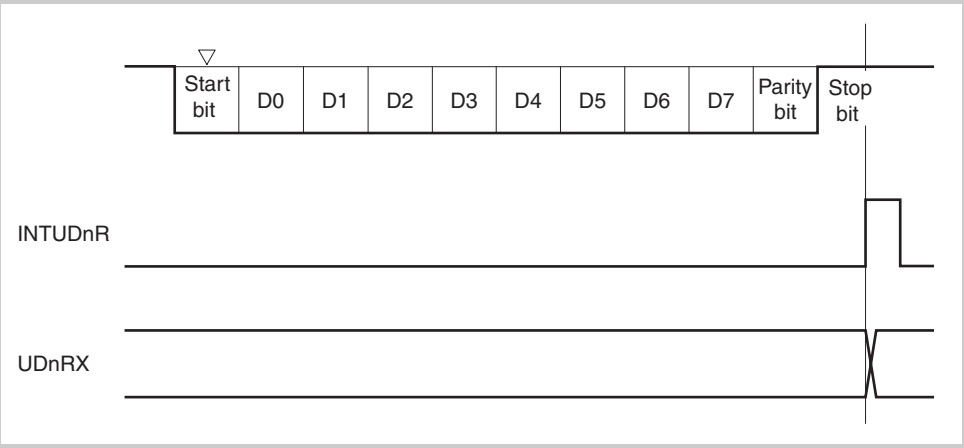
- Specify the operation clock by the UARTD control register 1 (UDnCTL1)
- Specify the baud rate by the UARTD control register 2 (UDnCTL2)
- Specify the output logic level by the UARTD option control register 0 (UDnOPT0)
- Specify the communication direction, parity, data character length, and stop bit length by the UARTD control register 0 (UDnCTL0).
- Set the power bit and the reception enabled bit (UDnPWR = 1, UDnRXE=1).

When the sampling of the input level of the RXDDn pin is performed and the falling edge is detected, the data sampling of the RXDDn input is started. The start bit is recognized if the RXDDn pin is low level after the time of a half bit is passed after the detection of the falling edge (shown in the figure below). After a start bit has been recognized, the receive operation starts, and serial data is stored in the receive shift register according to the set baud rate. When the reception complete interrupt request signal (INTUDnR) is output upon reception of the stop bit, the data stored in the receive shift register is written to the receive data register (UDnRX).

However, if an overrun error occurs (UDnOVE = 1), the receive data at this time is not transferred to the UDnRX register and is discarded. Even if a parity error (UDnPE = 1) or a framing error (UDnFE = 1) occurs during reception, reception continues until the reception position of the first stop bit, and the reception data is transferred to the UDnRX. In any case of the reception errors, INTUDnS is output after the following reception completion, but not INTUDnR.

When the communication direction, parity, data character length, and the stop bit length are changed, clear the power bit (UDnPWR = 0) or clear both the transmission enabled bit and the reception enabled bit (UDnTXE = 0, UDnRXE = 0), and then change the setting.

Figure 16-8 UART reception



Cautions

1. Be sure to read the UDnRX register even when a reception error occurs. If the UDnRX register is not read, an overrun error occurs during reception of the next data.

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 UDnRX register after the reception complete interrupt request signal (INTUDnR) has been generated, and clear the UDnPWR or UDnRXE bit to 0. If the UDnPWR or UDnRXE bit is cleared to 0 before the INTUDnR signal is generated, the read value of the UDnRX register cannot be guaranteed.
4. If receive completion processing (INTUDnR signal generation) of UARTDn and the UDnPWR bit = 0 or UDnRXE bit = 0 conflict, the INTUDnR signal may be generated in spite of these being no data stored in the UDnRX register.
   To complete reception without waiting INTUDnR signal generation, be sure to clear (0) the interrupt request flag (UDnRIF) of the UDnRIC register, after setting (1) the interrupt mask flag (UDnRMK) of the interrupt control register (UDnRIC) and then set (1) the UDnPWR bit = 0 or UDnRXE bit = 0.

Note
1. If the low level is always input to the RXDDn pin, it is not judged as the start bit.
2. In continuous reception, immediately after the stop bit is detected at the first reception bit (when the reception completion interrupt is generated), the next start bit can be detected.
3. If the UDnRDL = 1 (receive data inversion input) is selected, when the reception is started, change the data reception pin to the UART receive pin mode and then enable the reception. If the pin mode is changed after the reception is enabled, the start bit is detected faultily if the pin level at this time is high level.

16.6.10 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 UDnSTR register and a status interrupt request signal INTUDnS is output when an error occurs.

It is possible to ascertain which error occurred during reception by reading the contents of the UDnSTR register.

Clear the reception error flag by writing 0 to it after reading it.

Table 16-6 Reception error causes

Error flag	Reception error	Cause
UDnPE	Parity error	Received parity bit does not match the setting
UDnFE	Framing error	Stop bit not detected
UDnOVE	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 UDnRX. Consequently the data from UDnRX must be read. Otherwise an overrun error UDnSTR.UDnOVE will occur at reception of the next data.

In case of an overrun error, the receive shift register data is not transferred to UDnRX, thus the previous data is not overwritten.

16.6.11 Parity types and operations

Caution When using the LIN function, fix the UDnPS1 and UDnPS0 bits of the UDnCTL0 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.

16.6.12 Receive data noise filter

This filter samples the RXDDn pin using the base clock of the prescaler output.

When the same sampling value is read twice, the match detector output changes and the RXDDn 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 16-10*). See *Section (1), Base clock* on page 398 regarding the base clock.

Moreover, since the circuit is as shown in *Figure 16-9*, the processing that goes on within the receive operation is delayed by 3 clocks in relation to the external signal status.

Figure 16-9 Noise filter circuit

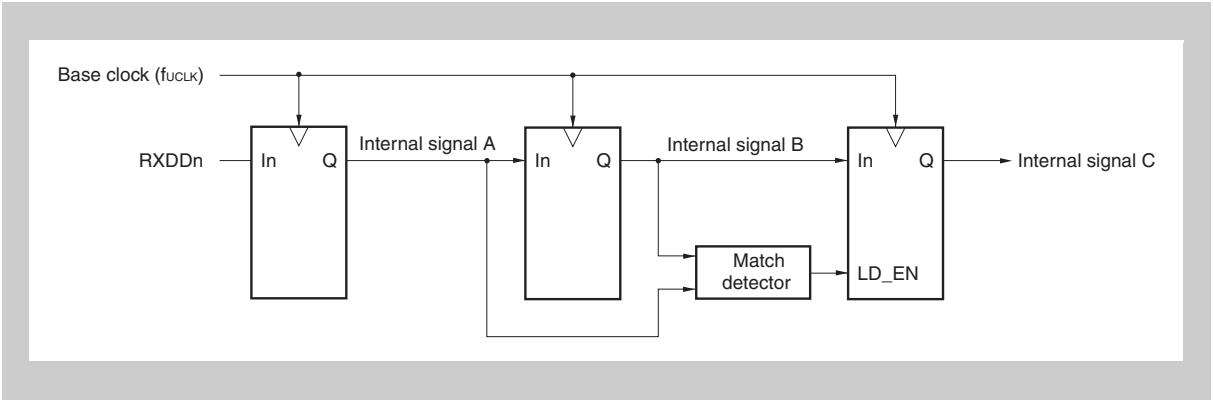
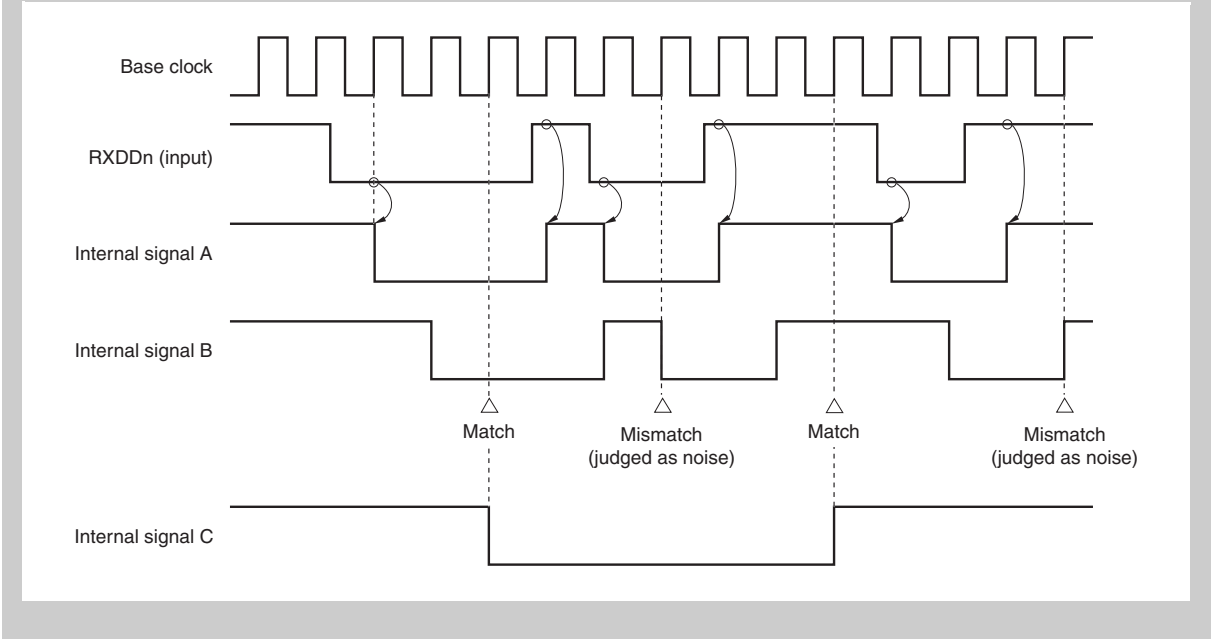


Figure 16-10 Timing of RXDDn signal judged as noise



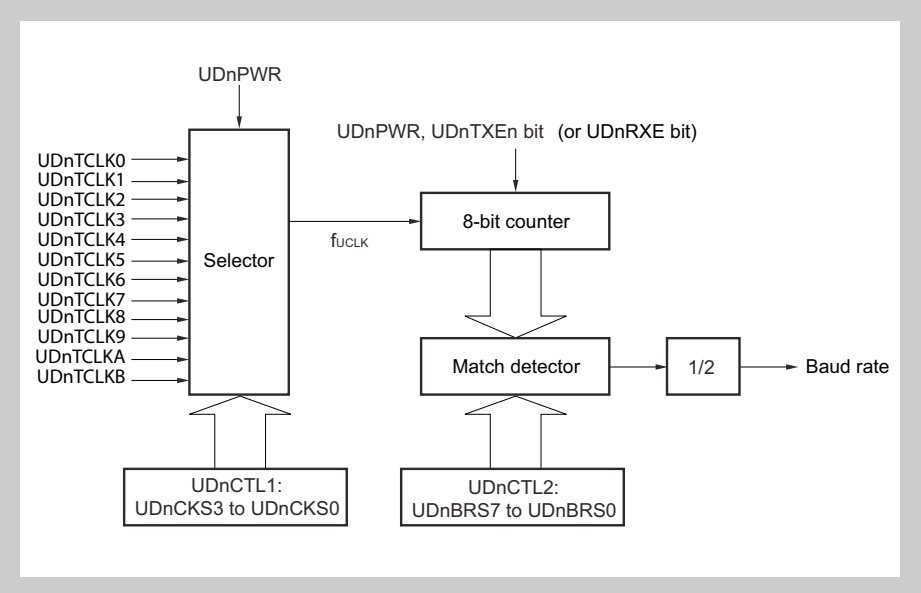
16.7 Baud Rate Generator

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 UARTDn. 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.

16.7.1 Baud rate generator configuration

Figure 16-11 Configuration of baud rate generator



(1) Base clock

When the UDnCTL0.UDnPWR bit is 1, the clock selected by the UDnCTL1.UDnCKS[3:0] bits are supplied to the 8-bit counter. This clock is called the base clock. When the UDnPWR bit = 0, f_{UCLK} is fixed to the low level.

(2) Serial clock generation

A serial clock can be generated by setting the UDnCTL1 register and the UDnCTL2 register.

The base clock is selected by UDnCTL1.UDnCKS3 to UDnCTL1.UDnCKS0.

The frequency division value for the 8-bit counter can be set using the UDnCTL2.UDnBRS[7:0] bits.

16.7.2 Baud rate

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 UDnCTL1.UDnCKS[3:0].

k = Value set using the UDnCTL2.UDnBRS[7:0] bits
(k = 4, 5, 6, …, 255)

16.7.3 Baud rate error

The baud rate error is obtained by the following equation.

$$\text{Error (\%)} = \left(\frac{\text{Actual baud rate (baud rate with error)}}{\text{Target baud rate (correct baud rate)}} - 1 \right) \times 100 \text{ [\%]}$$

Cautions

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 (7) Allowable baud rate range during reception.

Example

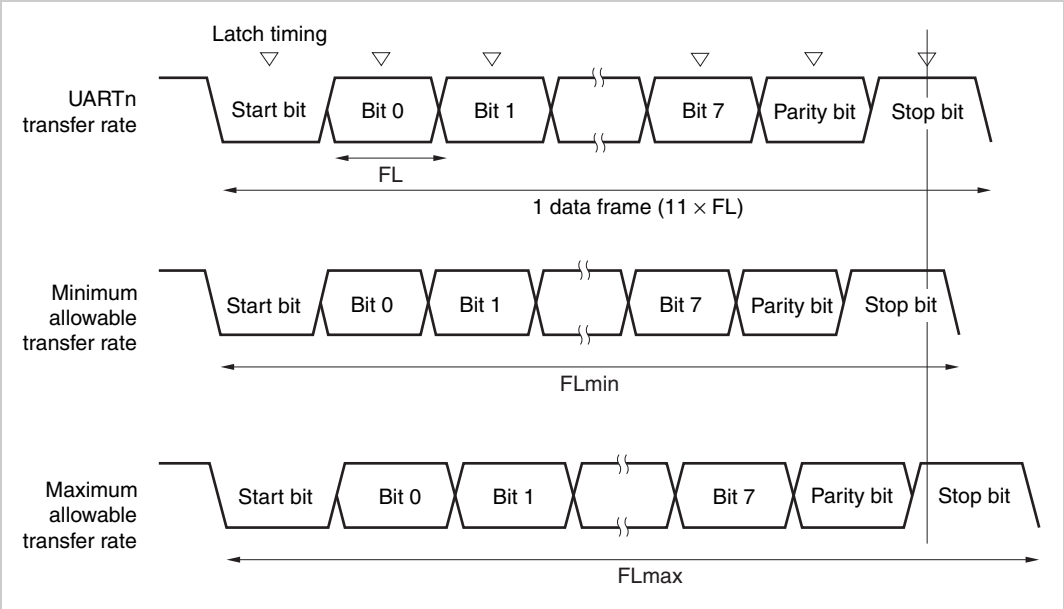
- Assumption: clock frequency of UDnTCKI1 fUDnTCKI1 = 16 MHz
- Setting value of
  - UDnCTL1.UDnCKS[3:0] = 0001_B: fUCLK = fUDnTCKI1 = 16 MHz
  - UDnCTL2.UDnBRS[7:0] = $0011\ 0100_B$: k = 52
- Target baud rate = 153,600 bps
- Actual Baud rate = 16 MHz / (2 × 52) = 153,846 bps
- Baud rate error = (153,846/153,600 – 1) × 100 = 0.160%

16.7.4 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 16-12 Allowable baud rate range during reception



As shown in *Figure 16-12*, the receive data latch timing is determined by the counter set using the UDnCTL2 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: UARTDn baud rate

k: Setting value of UDnCTL2.UDnBRS[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 UARTDn and the destination from the above-described equations for obtaining the minimum and maximum baud rate values yields the following.

Table 16-7 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%

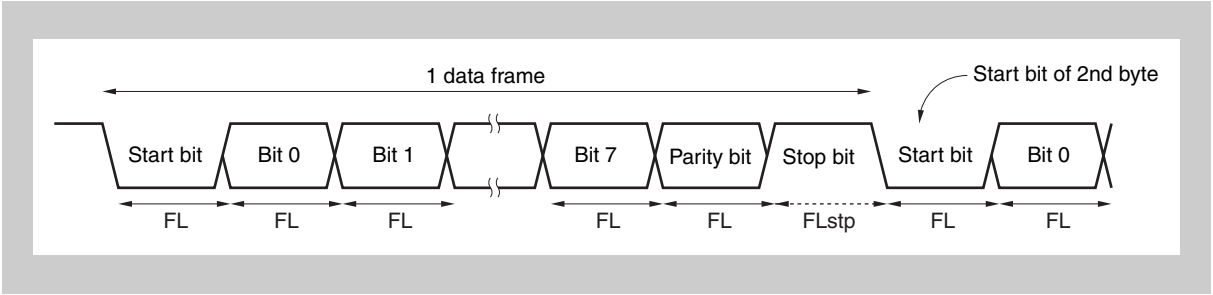
Note 1. The reception accuracy depends on the bit count in 1 frame, the input clock frequency, and the division ratio (k). The higher the input clock frequency and the larger the division ratio (k), the higher the accuracy.

2. k: Setting value of UDnCTL2.UDnBRS[7:0]

16.7.5 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 16-13 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.

$$\text{Transfer rate} = 11 \times FL + (2/f_{UCLK})$$

Chapter 17 CAN Controller (aFCAN)

The microcontroller features an on-chip 2-channel CAN (Controller Area Network) controller that complies with the CAN protocol as standardized in ISO 11898.

Instances

The V850E/RG3 has two instances of CAN macros:

Table 17-1 Instances of CAN

CAN macros	Instances	Names
aFCAN	2	CAN0 and CAN1

Throughout this chapter, the individual instances of aFCAN are identified by “n” (n = 0 to 1), for example, C0CTRL for the CAN 0 Control Register.

Register addresses

All CANn register addresses are given as address offsets to the individual base addresses <base> of each CANn.

The <base> addresses of each CANn are listed in the following table:

Table 17-2 Register <base> addresses of CANn

CANn	<base> address
CAN0	$0840\ 0000_H$
CAN1	$0840\ 0800_H$

Note

1. The actual value of the <base> address depends on the settings of the BPC-register. The above mentioned values are related to the recommended value $A100_H$.
2. Throughout this chapter, the individual CAN channels are identified by “n” (n = for example CANn, or CnGMCTRL for the CANn global control register.
3. Throughout this chapter, the CAN message buffer registers are identified by “m” (m = 0 to 3), for example C0MDATA4m for CAN0 message data byte 4 of message buffer register m.

17.1 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
- 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%
  - Baud rates 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 CAN channels 0 to n in collaboration with timers capture channels

17.1.1 Overview of functions

Table 17-3 presents an overview of the CAN Controller functions.

Table 17-3 Overview of functions

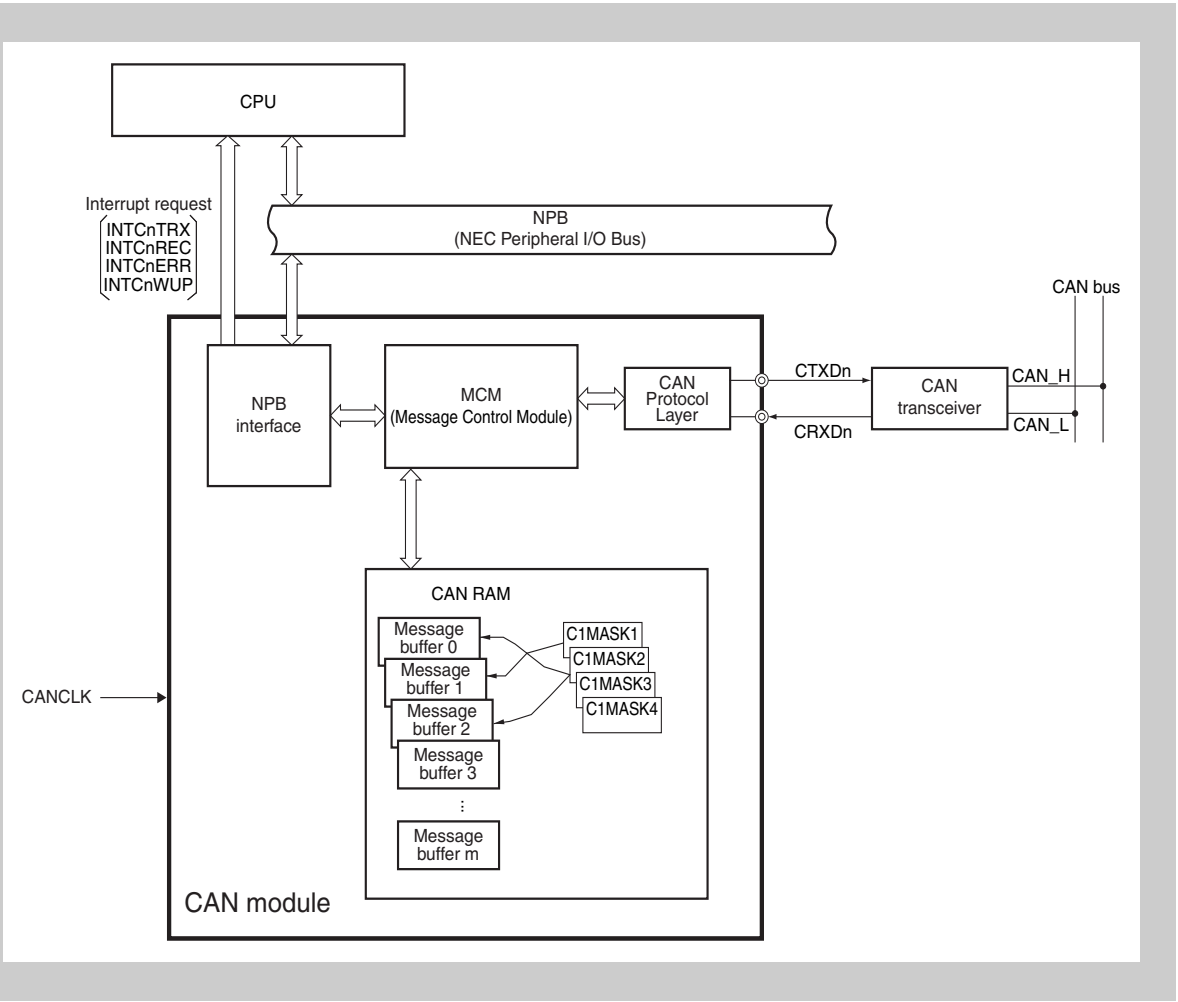
Function	Details
Protocol	CAN protocol ISO 11898 (standard and extended frame transmission/reception)
Baud rate	Maximum 1 Mbps (CAN clock input ≥ 8 MHz)
Data storage	Storing messages in the CAN RAM
Number of messages	• 32 message buffers per channel • Each message buffer can be set to be either a transmit 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)

17.1.2 Configuration

The CAN Controller is composed of the following four blocks.

- NPB interface
  This functional block provides an NPB (NEC Peripheral I/O Bus) interface and means of transmitting and receiving signals between the CAN module and the host CPU.
- MAC (Memory Access Controller)
  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 17-1 Block diagram of CAN module

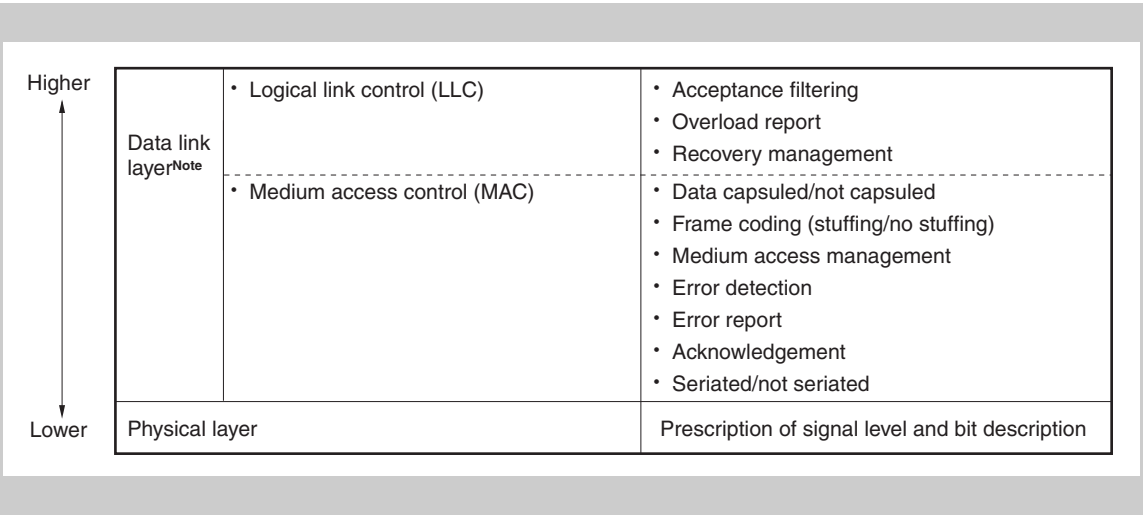


17.2 CAN Protocol

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.

Figure 17-2 Composition of layers



Note CAN Controller specification

17.2.1 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 $2{,}048 \times 2^{18}$ 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.

17.2.2 Frame types

The following four types of frames are used in the CAN protocol.

Table 17-4 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.

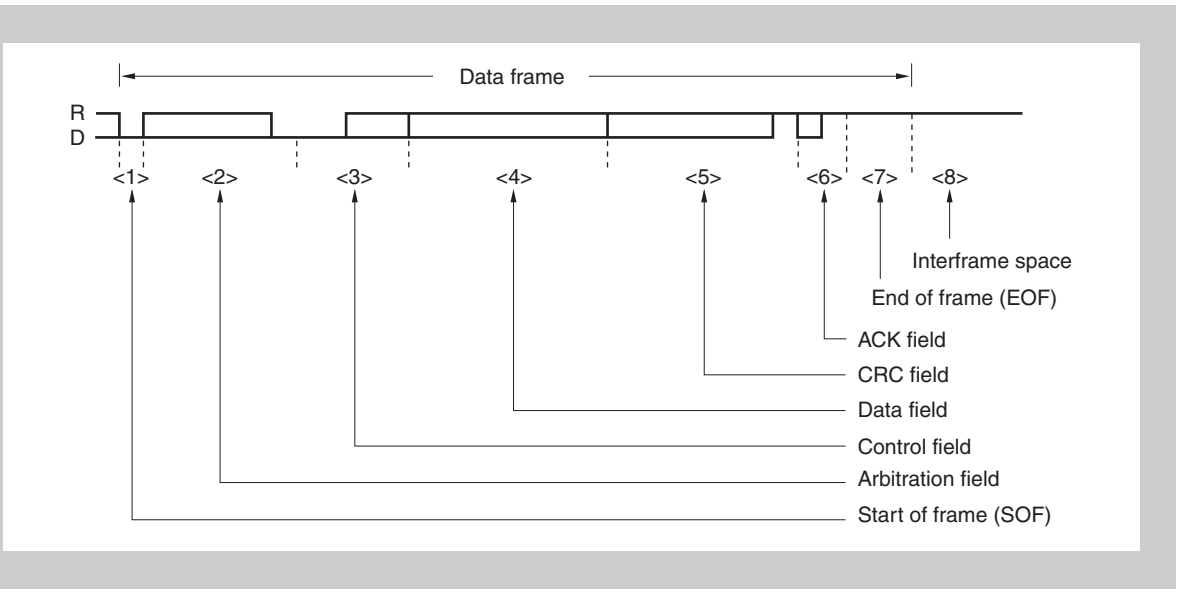
- Dominant level is indicated by logical 0.
- Recessive level is indicated by logical 1.
- When a dominant level and a recessive level are transmitted simultaneously, the bus value becomes dominant level.

17.2.3 Data frame and remote frame

(1) Data frame

A data frame is composed of seven fields.

Figure 17-3 Data frame

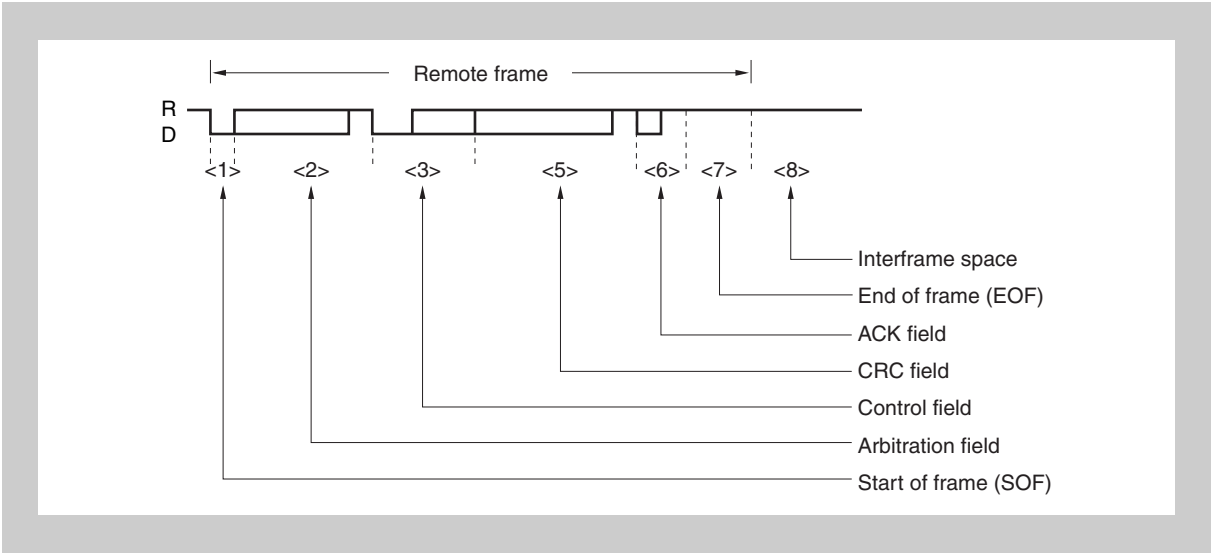


Note D: Dominant = 0
R: Recessive = 1

(2) Remote frame

A remote frame is composed of six fields.

Figure 17-4 Remote frame



Note 1. The data field is not transferred even if the control field's data length code is not "0000_B".

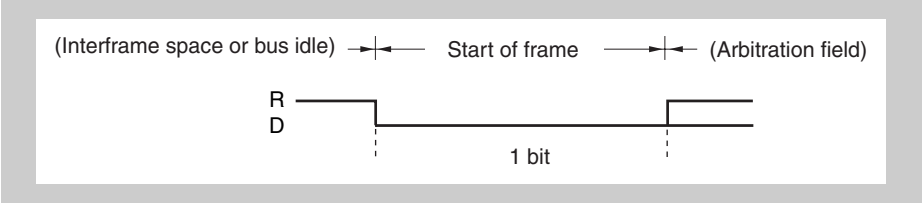
2. D: Dominant = 0
R: Recessive = 1

(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 17-5 Start of frame (SOF)



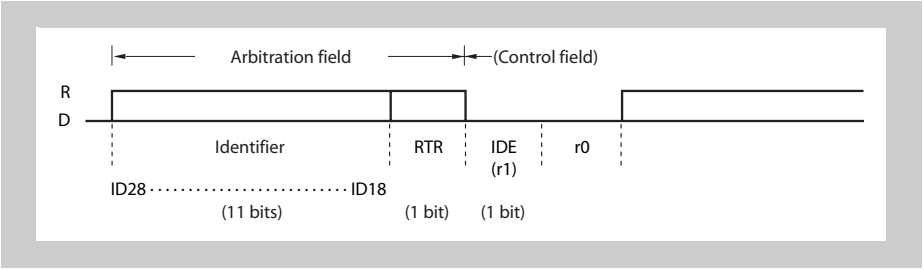
Note D: Dominant = 0
R: Recessive = 1

- 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 a case.

(b) Arbitration field

The arbitration field is used to set the priority, data frame/remote frame, and frame format.

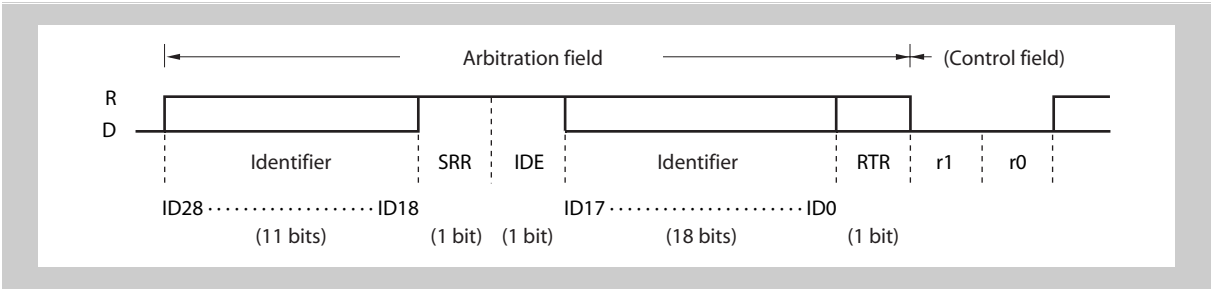
Figure 17-6 Arbitration field (in standard format mode)



Cautions
1. ID28 to ID18 are identifiers.
2. An identifier is transmitted MSB first.

Note D: Dominant = 0
R: Recessive = 1

Figure 17-7 Arbitration field (in extended format mode)



Cautions
1. ID28 to ID18 are identifiers.
2. An identifier is transmitted MSB first.

Note D: Dominant = 0
R: Recessive = 1

Table 17-5 RTR frame settings

Frame type	RTR bit
Data frame	0 (D)
Remote frame	1 (R)

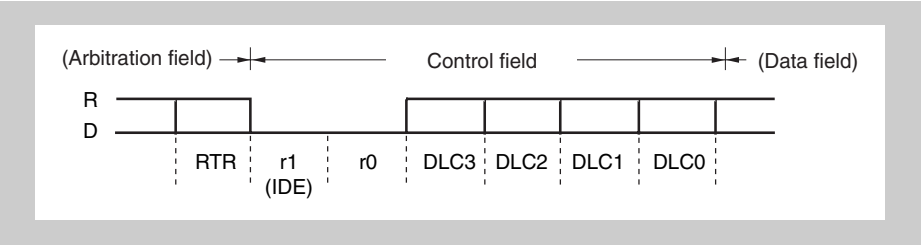
Table 17-6 Frame format setting (IDE bit) and number of identifier (ID) bits

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 17-8 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 17-7 Data length setting

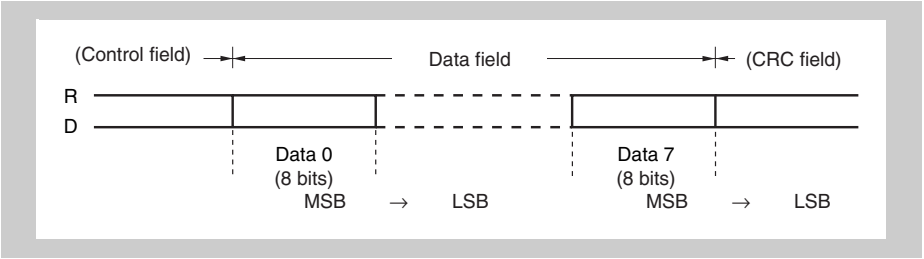
Data length code				Data byte count
DLC3	DLC2	DLC1	DLC0	
0	0	0	0	0 bytes
0	0	0	1	1 byte
0	0	1	0	2 bytes
0	0	1	1	3 bytes
0	1	0	0	4 bytes
0	1	0	1	5 bytes
0	1	1	0	6 bytes
0	1	1	1	7 bytes
1	0	0	0	8 bytes
Other than above				8 bytes regardless of the value of DLC3 to DLC0

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 17-9 Data field

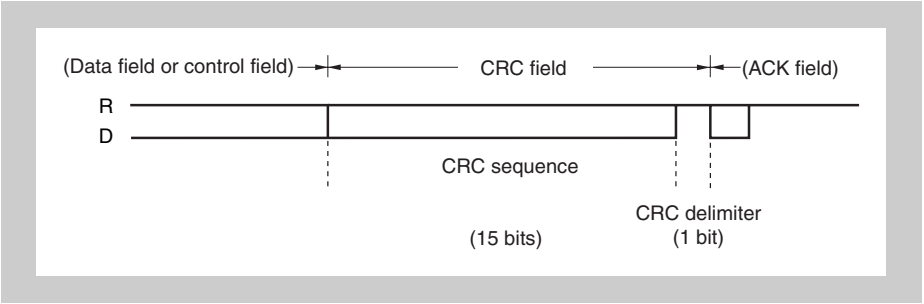


Note D: Dominant = 0
R: Recessive = 1

(e) CRC field

The CRC field is a 16-bit field that is used to check for errors in transmitted data.

Figure 17-10 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.

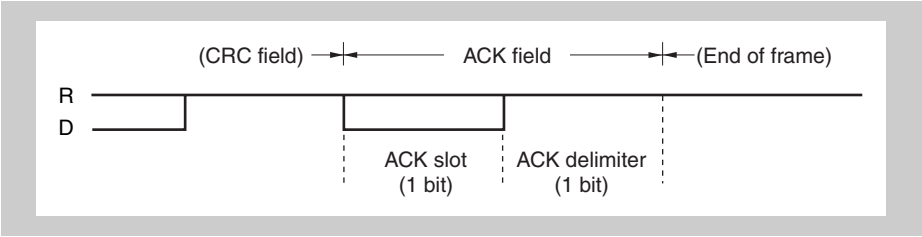
  $P(X) = X^{15} + X^{14} + X^{10} + X^8 + X^7 + X^4 + X^3 + 1$

- Transmitting node: Transmits the CRC sequence calculated from the data (before bit stuffing) at the start of the 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 received 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 17-11 ACK field



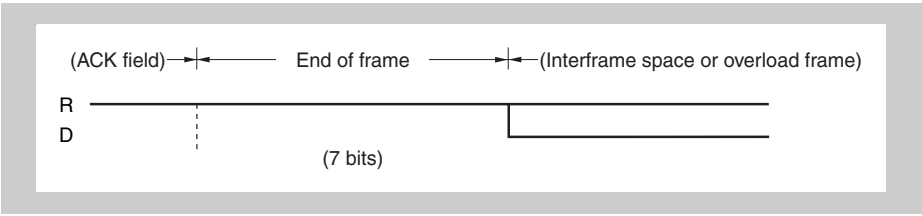
Note D: Dominant = 0
R: Recessive = 1

- 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 a data frame/remote frame.

Figure 17-12 End of frame (EOF)



Note D: Dominant = 0
R: Recessive = 1

(h) Interframe space

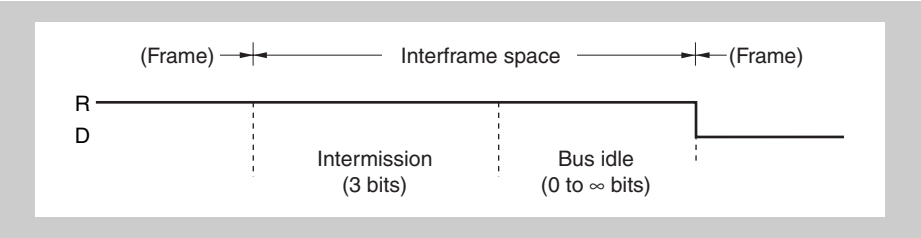
The interframe space is inserted after a data frame, remote frame, error frame, or overload frame to separate one frame from the next.

- 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 17-13 Interframe space (error active node)



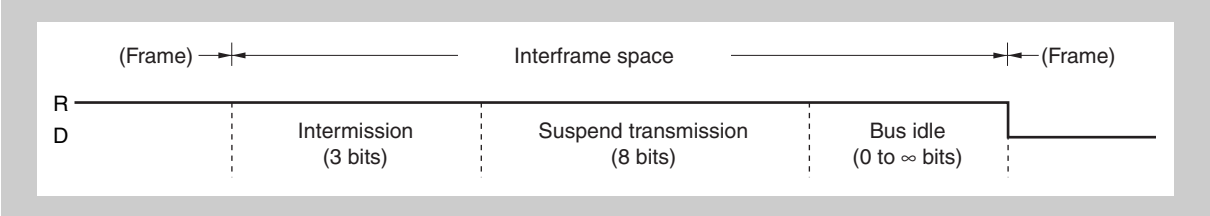
Note 1. Bus idle: State in which the bus is not used by any node.

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 17-14 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

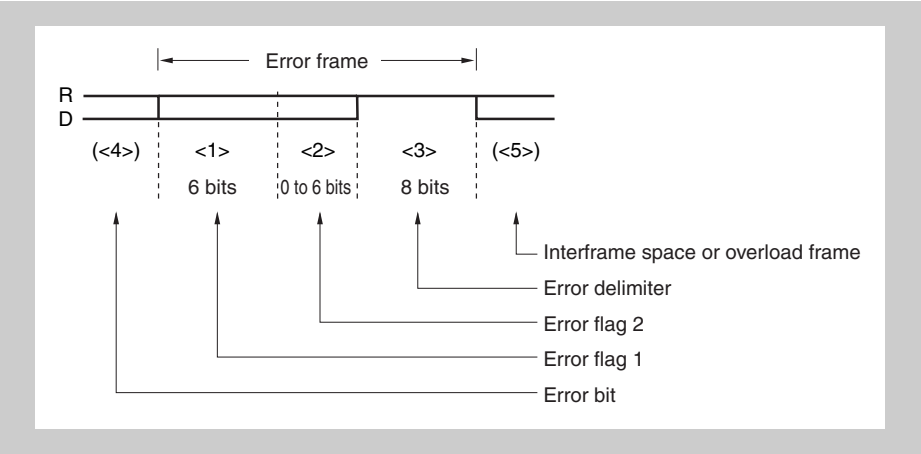
Table 17-8 Operation in error status

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.

17.2.4 Error frame

An error frame is output by a node that has detected an error.

Figure 17-15 Error frame



Note D: Dominant = 0
R: Recessive = 1

Table 17-9 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 issue 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.

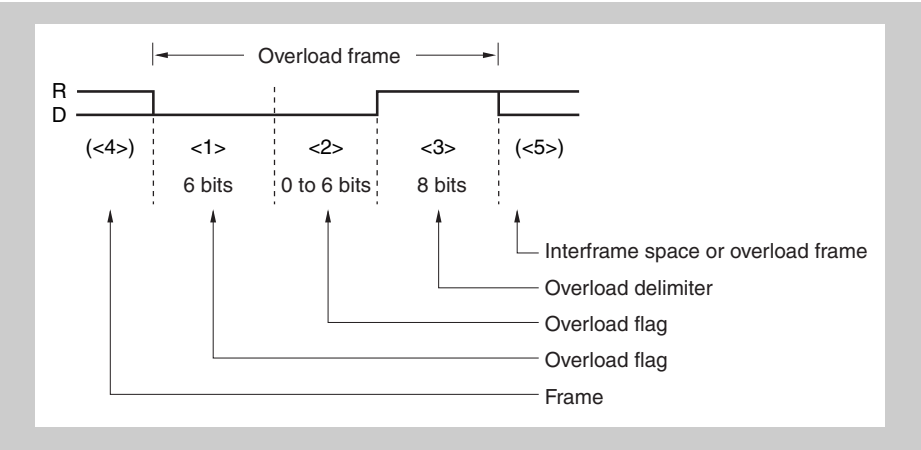
17.2.5 Overload frame

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 that are not generating overload frames.

Figure 17-16 Overload frame



Note D: Dominant = 0
R: Recessive = 1

Table 17-10 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.

17.3 Functions

17.3.1 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 17-11 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.

17.3.2 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 17-12 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.

17.3.3 Multi masters

Because the bus priority (a node acquiring transmit functions) is determined by the identifier, any node can be the bus master.

17.3.4 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.

17.3.5 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. However, it is not woken up from the CAN stop mode by bus operation; the CAN stop mode is controlled by CPU access.

17.3.6 Error control function

(1) Error types

Table 17-13 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 17-14 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.
CRC 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 17-15*.

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 the bus is considered to have 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 17-15 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 occurs 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 17-16 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.

12. **A request to enter the CAN initialization mode**

13. **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 to timing <1> in *Figure 17-17 on page 423*). 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 17-17 on page 423*). 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 17-17 on page 423*), 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.

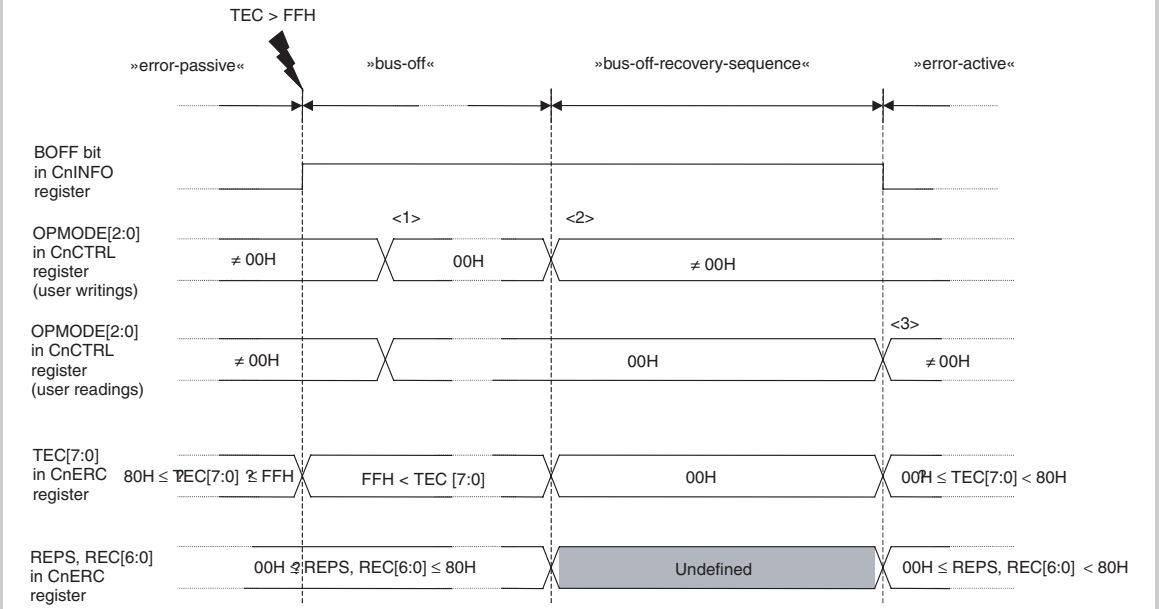
During the bus-off period and bus-off recovery sequence, the BOFF bit of the CnINFO register stays set (to 1). In the bus-off recovery sequence, the reception error counter (REC[6:0]) counts the number of times 11 consecutive recessive-level bits have been detected on the bus. Therefore, the recovery state can be checked by reading REC[6:0].

Cautions

1. 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.

2. During the bus-off recovery sequence, when the request to change the mode from the initialization mode to an operation mode is generated to execute the buss-off recovery sequence again, the reception error counter (REC [6:0]) is cleared. In this case, it is required to detect 11 consecutive recessive-level bits 128 times again on the bus.

Figure 17-17 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, see *Section (a), Recovery from bus-off state through normal recovery sequence* on page 422.

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 17-54 on page 523*.

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.

Cautions

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.

17.3.7 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 (*Section 17.7.9, CnBRP - CANn module bit rate prescaler register* on page 453).

(2) Data bit time (8 to 25 time quanta)

One data bit time is defined as shown in *Chapter 17, Section Figure 17-18, Segment setting*.

The CAN Controller sets time segment 1, time segment 2, and reSynchronization Jump Width (SJW) of data bit time, as shown in *Figure 17-18*. 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 17-18 Segment setting

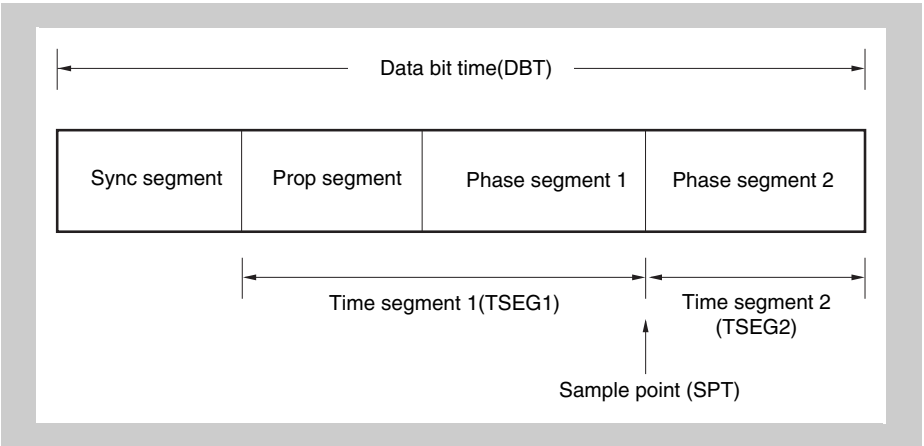


Table 17-17 Segment setting

Segment name	Settable range	Notes on setting to conform to CAN specification
Time segment 1 (TSEG1)	2TQ to 16TQ	-
Time segment 2 (TSEG2)	1TQ to 8TQ	IPT[1] of the CAN controller is 0TQ[2]. 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 17-19*.

Figure 17-19 Configuration of data bit time defined by CAN specification

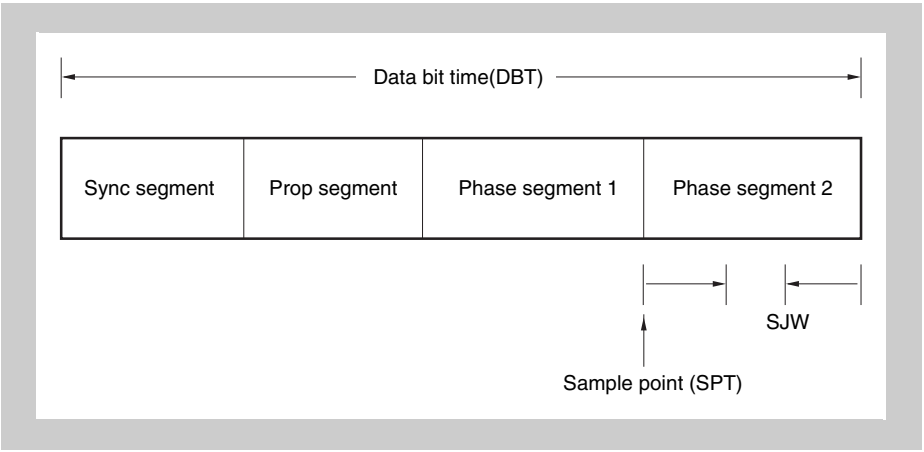


Table 17-18 Configuration of data bit time defined by CAN specification

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.
Phase segment 1	Programmable to 1 to 8	The length of this segment is set so that ACK is returned before the start of phase segment 1.
Phase segment 2	Phase segment 1 or IPT, whichever greater	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.
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

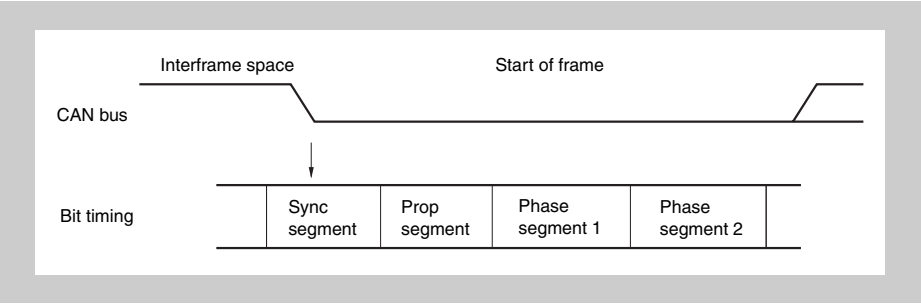
- The receiving node establishes synchronization by a level change on the bus because it does not have a sync signal.
- The transmitting node transmits data in synchronization with the bit timing of the transmitting node.

(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 indicates the sync segment and the next segment is the prop segment. In this case, synchronization is established regardless of SJW.

Figure 17-20 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)

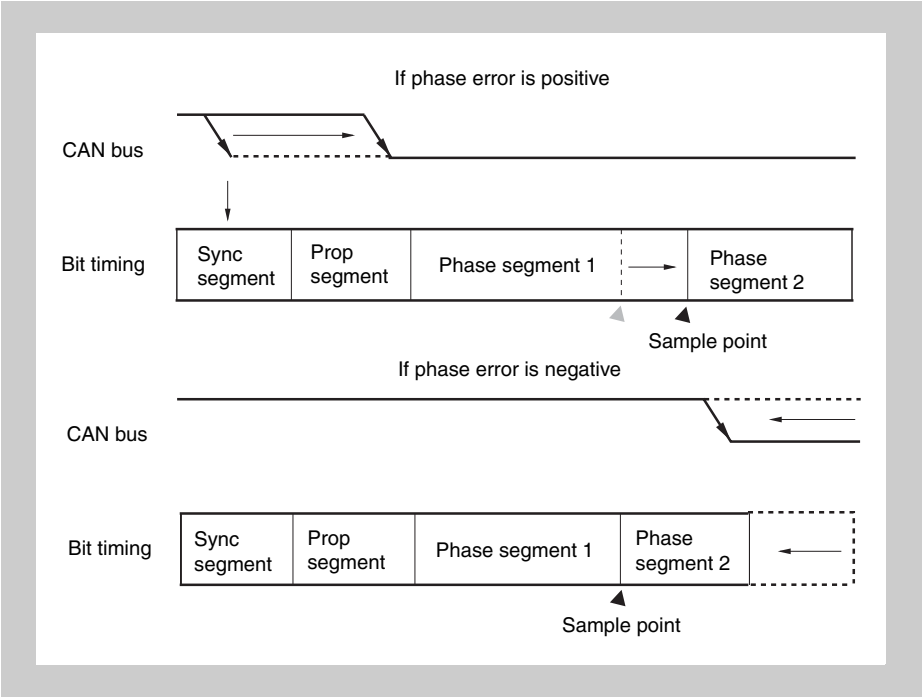
  Negative: If the edge is after the sample point (phase error)

  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.

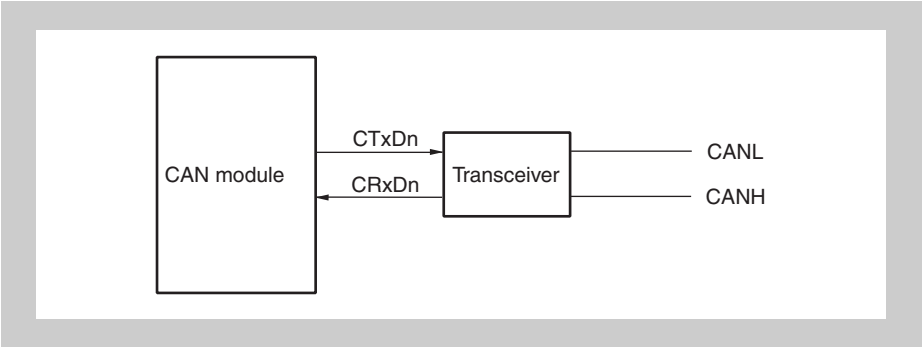
Figure 17-21 Resynchronization



17.4 Connection with Target System

The CAN module has to be connected to the CAN bus using an external transceiver.

Figure 17-22 Connection to CAN bus



17.5 Internal Registers of CAN Controller

17.5.1 CAN module register and message buffer addresses

In this chapter all register and message buffer addresses are defined as address offsets from different base addresses.

Since all registers are accessed via the programmable peripheral area the bottom address is defined by the BPC register (refer to *Section 3.5.4, Programmable peripheral I/O area (PPA)* on page 55).

The addresses given in the following tables are offsets from the programmable peripheral area base address PBA.

The recommended setting of BPC is $A100_H$. This setting would define the programmable peripheral area base address

PBA = $0840\ 0000_H$

Table 17-19 lists all base addresses used throughout this chapter.

Table 17-19 CAN module base addresses

Base address name	Base address of	Address	Address for BPC =$A100_H$
C0RBaseAddr	CAN0 registers	PBA + 0000_H	$0840\ 0000_H$
C0MBaseAddr	CAN0 message buffers	PBA + 0100_H	$0840\ 0100_H$
C1RBaseAddr	CAN1 registers	PBA + 0800_H	$0840\ 0800_H$
C1MBaseAddr	CAN1 message buffers	PBA + 0900_H	$0840\ 0900_H$

In the following <CnRBaseAddr> respectively <CnMBaseAddr> are used for the base address names for CAN channel n.

17.5.2 CAN Controller configuration

Table 17-20 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)

17.5.3 CAN registers overview

(1) CANn global and module registers

The following table lists the address offsets to the CANn register base address: CnRBaseAddr.

Table 17-21 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		–	–	√	0000_H
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 17-22 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

17.5.4 Register bit configuration

Table 17-23 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 17-24 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
52_H	CnLEC (W)	0	0	0	0	0	0	0	0
52_H	CnLEC (R)	0	0	0	0	0	LEC2	LEC1	LEC0

Table 17-24 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
53_H	CnINFO	0	0	0	BOFF	TECS1	TECS0	RECS1	RECS0
54_H	CnERC	TEC7 to TEC0							
55_H		REPS	REC6 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 17-25 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)							
4_H	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 *Section 17.5.3, CAN registers overview* on page 430.

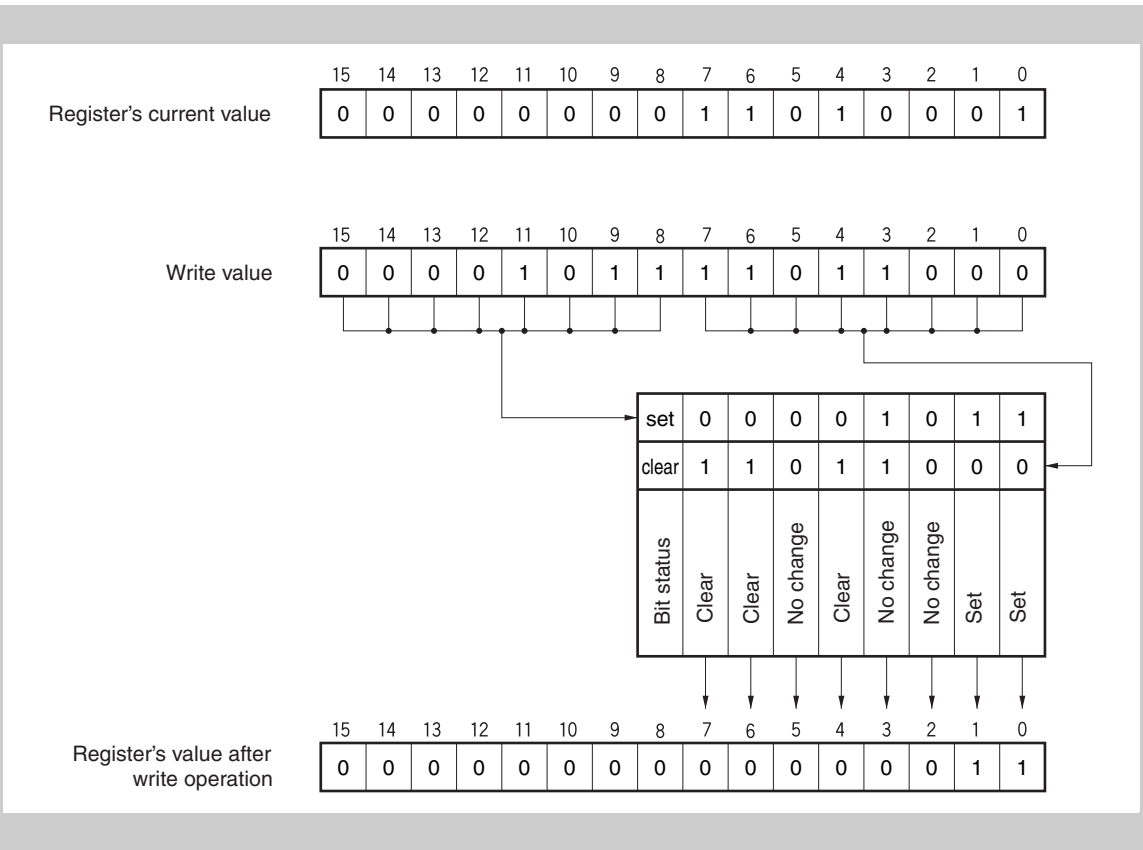
17.6 Bit Set/Clear Function

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 17-23* below to set or clear the lower 8 bits in these registers. Setting or clearing of the 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 17-26*). *Figure 17-23* shows how the values of set bits or clear bits relate to set/clear/no change operations in the corresponding register.

Figure 17-23 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

17.7 Control Registers

17.7.1 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.

(1) CnGMCTRL read

15	14	13	12	11	10	9	8
MBON	0	0	0	0	0	0	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
0	0	0	0	0	0	EFSD	GOM
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

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.

Cautions

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 by GOM bit = 0 disabled.
1	Forced shut down by GOM bit = 0 enabled.

Caution To request a forced shut down, the GOM bit must be cleared to 0 in a subsequent write, immediately following access after the EFSD bit has been set to 1. If access to another register (including reading the CnGMCTRL register) is executed 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.

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). Failure to clear the GOM in the latter case means that when access to another register occurs, the EFSD bit will forcibly clear to 0 and the forced shutdown request becomes invalid.

(2) CnGMCTRL write

15	14	13	12	11	10	9	8
0	0	0	0	0	0	Set EFSD	Set GOM
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	Clear GOM
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

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.

17.7.2 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
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

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

17.7.3 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.

(1) CnGMABT read

15	14	13	12	11	10	9	8
0	0	0	0	0	0	0	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
0	0	0	0	0	0	ABTCLR	ABTTRG
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

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 process is complete.

ABTTRG	Automatic block transmission status bit
0	Automatic block transmission is stopped.
1	Automatic block transmission is under execution.

Cautions

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.

(2) CnGMABT write

15	14	13	12	11	10	9	8
0	0	0	0	0	0	Set ABTCLR	Set ABTTRG
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	Clear ABTTRG
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

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.

(3) 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
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

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)

Cautions

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.

(4) 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
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

7	6	5	4	3	2	1	0
CMID7	CMID6	CMID5	CMID4	CMID3	CMID2	CMID1	CMID0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

CnMASK1H

15	14	13	12	11	10	9	8
0	0	0	CMID28	CMID27	CMID26	CMID25	CMID24
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

7	6	5	4	3	2	1	0
CMID23	CMID22	CMID21	CMID20	CMID19	CMID18	CMID17	CMID16
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

(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
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

7	6	5	4	3	2	1	0
CMID7	CMID6	CMID5	CMID4	CMID3	CMID2	CMID1	CMID0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

CnMASK2H

15	14	13	12	11	10	9	8
0	0	0	CMID28	CMID27	CMID26	CMID25	CMID24
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

7	6	5	4	3	2	1	0
CMID23	CMID22	CMID21	CMID20	CMID19	CMID18	CMID17	CMID16
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

(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
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

7	6	5	4	3	2	1	0
CMID7	CMID6	CMID5	CMID4	CMID3	CMID2	CMID1	CMID0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

CnMASK3H

15	14	13	12	11	10	9	8
0	0	0	CMID28	CMID27	CMID26	CMID25	CMID24
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

7	6	5	4	3	2	1	0
CMID23	CMID22	CMID21	CMID20	CMID19	CMID18	CMID17	CMID16
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

(d) CANn module mask 4 register (CnMASK4L, CnMASK4H)

Access These registers can be read/written in 16-bit units.

Address CnMASK4L: <CnRBaseAddr> + 04CH
CnMASK4H: <CnRBaseAddr> + 04EH

Initial Value Undefined.

CnMASK4L

15	14	13	12	11	10	9	8
CMID15	CMID14	CMID13	CMID12	CMID11	CMID10	CMID9	CMID8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

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
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

7	6	5	4	3	2	1	0
CMID23	CMID22	CMID21	CMID20	CMID19	CMID18	CMID17	CMID16
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

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.

17.7.4 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> + 050H

Initial Value 0000_H. The register is initialized by any reset.

(1) CnCTRL read

15	14	13	12	11	10	9	8
0	0	0	0	0	0	RSTAT	TSTAT
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

7	6	5	4	3	2	1	0
CCERC	AL	VALID	PSMODE1	PSMODE0	OPMODE2	OPMODE1	OPMODE0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

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 when 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

Cautions

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.

(a) 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
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

7	6	5	4	3	2	1	0
0	Clear AL	Clear VALID	Clear PSMODE1	Clear PSMODE0	Clear OPMODE2	Clear OPMODE1	Clear OPMODE0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

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.

17.7.5 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> + 052H

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
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

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

(1) 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
R	R	R	R	R	R	R	R

BOFF	Bus-off state bit
0	Not bus-off state (transmit error counter ≤ 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 (≥ 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 (≥ 128).

17.7.6 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
R	R	R	R	R	R	R	R

7	6	5	4	3	2	1	0
TEC7	TEC6	TEC5	TEC4	TEC3	TEC2	TEC1	TEC0
R	R	R	R	R	R	R	R

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).

17.7.7 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.

(1) CnIE read

15	14	13	12	11	10	9	8
0	0	0	0	0	0	0	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
0	0	CIE5	CIE4	CIE3	CIE2	CIE1	CIE0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

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.

(2) CnIE write

15	14	13	12	11	10	9	8
0	0	Set CIE5	Set CIE4	Set CIE3	Set CIE2	Set CIE1	Set CIE0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

7	6	5	4	3	2	1	0
0	0	Clear CIE5	Clear CIE4	Clear CIE3	Clear CIE2	Clear CIE1	Clear CIE0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

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.

17.7.8 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.

(1) CnINTS read

15	14	13	12	11	10	9	8
0	0	0	0	0	0	0	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
0	0	CINTS5	CINTS4	CINTS3	CINTS2	CINTS1	CINTS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

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.

(2) CnINTS write

15	14	13	12	11	10	9	8
0	0	0	0	0	0	0	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
0	0	Clear CINTS5	Clear CINTS4	Clear CINTS3	Clear CINTS2	Clear CINTS1	Clear CINTS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

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.

17.7.9 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 by the CnBTR register.

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

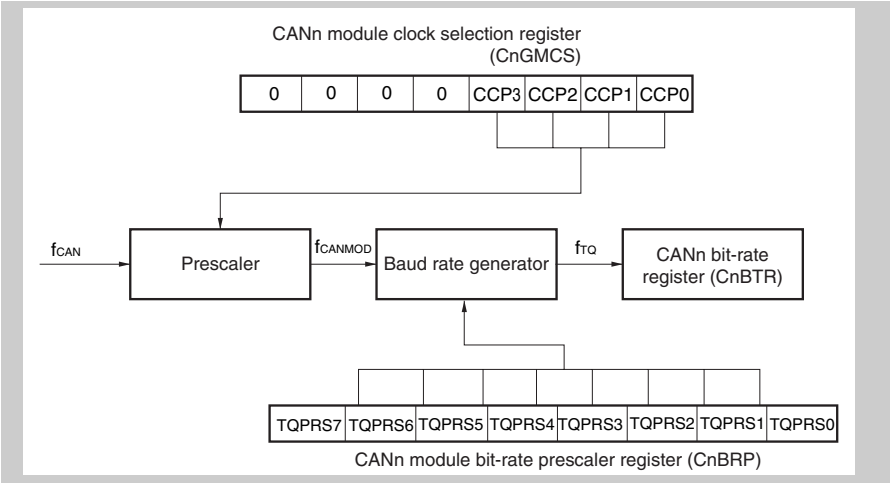
Address <CnRBaseAddr> + $05A_H$

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
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

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 17-24 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.

17.7.10 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.

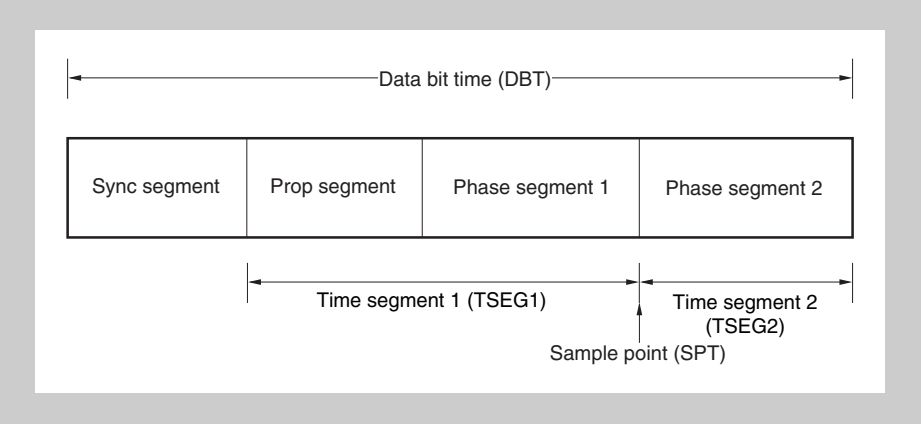
Address <CnRBaseAddr> + $05C_H$

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
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

7	6	5	4	3	2	1	0
0	0	0	0	TSEG13	TSEG12	TSEG11	TSEG10
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Figure 17-25 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)

17.7.11 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
R	R	R	R	R	R	R	R

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.

17.7.12 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.

(1) CnRGPT read

15	14	13	12	11	10	9	8
RGPT7	RGPT6	RGPT5	RGPT4	RGPT3	RGPT2	RGPT1	RGPT0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

7	6	5	4	3	2	1	0
0	0	0	0	0	0	RHPM	ROVF
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

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.

(2) CnRGPT write

15	14	13	12	11	10	9	8
0	0	0	0	0	0	0	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
0	0	0	0	0	0	0	Clear ROVF
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Clear ROVF	Setting of ROVF bit
0	ROVF bit is not changed.
1	ROVF bit is cleared to 0.

17.7.13 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
R	R	R	R	R	R	R	R

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.

17.7.14 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.

(1) CnTGPT read

15	14	13	12	11	10	9	8
TGPT7	TGPT6	TGPT5	TGPT4	TGPT3	TGPT2	TGPT1	TGPT0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

7	6	5	4	3	2	1	0
0	0	0	0	0	0	THPM	TOVF
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

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.

(2) CnTGPT write

15	14	13	12	11	10	9	8
0	0	0	0	0	0	0	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
0	0	0	0	0	0	0	Clear TOVF
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Clear TOVF	Setting of TOVF bit
0	TOVF bit is not changed.
1	TOVF bit is cleared to 0.

17.7.15 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.

(1) CnTS read

15	14	13	12	11	10	9	8
0	0	0	0	0	0	0	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
0	0	0	0	0	TSLOCK	TSSEL	TSEN
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

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.

Remark The TSOUT signal is output from the CAN controller to the timer. For details, refer to Chapter 9“16-Bit Timer/Event Counter AA (TAA)” on page 161.

(2) CnTS write

15	14	13	12	11	10	9	8
0	0	0	0	0	Set TSLOCK	Set TSSEL	Set TSEN
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

7	6	5	4	3	2	1	0
0	0	0	0	0	Clear TSLOCK	Clear TSSEL	Clear TSEN
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

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.

17.7.16 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 *Section 17.5.3, CAN registers overview* on page 430.

Initial Value Undefined.

CnMDATA01m

15	14	13	12	11	10	9	8
MDATA0115	MDATA0114	MDATA0113	MDATA0112	MDATA0111	MDATA0110	MDATA019	MDATA018
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

7	6	5	4	3	2	1	0
MDATA017	MDATA016	MDATA015	MDATA014	MDATA013	MDATA012	MDATA011	MDATA010
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

CnMDATA0m

7	6	5	4	3	2	1	0
MDATA07	MDATA06	MDATA05	MDATA04	MDATA03	MDATA02	MDATA01	MDATA00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

CnMDATA1m

7	6	5	4	3	2	1	0
MDATA17	MDATA16	MDATA15	MDATA14	MDATA13	MDATA12	MDATA11	MDATA1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

CnMDATA23m

15	14	13	12	11	10	9	8
MDATA2315	MDATA2314	MDATA2313	MDATA2312	MDATA2311	MDATA2310	MDATA239	MDATA238
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

7	6	5	4	3	2	1	0
MDATA237	MDATA236	MDATA235	MDATA234	MDATA233	MDATA232	MDATA231	MDATA230
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

CnMDATA2m

7	6	5	4	3	2	1	0
MDATA27	MDATA26	MDATA25	MDATA24	MDATA23	MDATA22	MDATA21	MDATA20
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

CnMDATA3m

7	6	5	4	3	2	1	0
MDATA37	MDATA36	MDATA35	MDATA34	MDATA33	MDATA32	MDATA31	MDATA30
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

CnMDATA45m

15	14	13	12	11	10	9	8
MDATA4515	MDATA4514	MDATA4513	MDATA4512	MDATA4511	MDATA4510	MDATA459	MDATA458
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

7	6	5	4	3	2	1	0
MDATA457	MDATA456	MDATA455	MDATA454	MDATA453	MDATA452	MDATA451	MDATA450
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

CnMDATA4m

7	6	5	4	3	2	1	0
MDATA47	MDATA46	MDATA45	MDATA44	MDATA43	MDATA42	MDATA41	MDATA40
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

CnMDATA5m

7	6	5	4	3	2	1	0
MDATA57	MDATA56	MDATA55	MDATA54	MDATA53	MDATA52	MDATA51	MDATA50
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

CnMDATA67m

15	14	13	12	11	10	9	8
MDATA6715	MDATA6714	MDATA6713	MDATA6712	MDATA6711	MDATA6710	MDATA679	MDATA678
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

7	6	5	4	3	2	1	0
MDATA677	MDATA676	MDATA675	MDATA674	MDATA673	MDATA672	MDATA671	MDATA670
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

CnMDATA6m

7	6	5	4	3	2	1	0
MDATA67	MDATA66	MDATA65	MDATA64	MDATA63	MDATA62	MDATA61	MDATA60
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

CnMDATA7m

7	6	5	4	3	2	1	0
MDATA77	MDATA76	MDATA75	MDATA74	MDATA73	MDATA72	MDATA71	MDATA70
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

17.7.17 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 *Section 17.5.3, CAN registers overview* on page 430.

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	

Cautions

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.

17.7.18 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 *Section 17.5.3, CAN registers overview* on page 430.

Initial Value Undefined.

7	6	5	4	3	2	1	0
OWS	RTR	MT2	MT1	MT0	0	0	MA0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

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.

17.7.19 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 *Section 17.5.3, CAN registers overview* on page 430.

Initial Value Undefined.

CnMIDLm

15	14	13	12	11	10	9	8
ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

7	6	5	4	3	2	1	0
ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

CnMIDHm

15	14	13	12	11	10	9	8
IDE	0	0	ID28	ID27	ID26	ID25	ID24
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

7	6	5	4	3	2	1	0
ID23	ID22	ID21	ID20	ID19	ID18	ID17	ID16
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

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)

Cautions 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 these registers. Note that for standard ID, the ID value must be shifted to fit into ID28 to ID18 bit positions.

17.7.20 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 *Section 17.5.3, CAN registers overview* on page 430.

Initial Value 00x0 0000 0000 0000_B. The register is initialized by any reset.

(1) CnMCTRLm read

15	14	13	12	11	10	9	8
0	0	MUC	0	0	0	0	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
0	0	0	MOW	IE	DN	TRQ	RDY
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

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) MOW is not set to 1 even if a remote frame is received and stored in the transmit message buffer with DN = 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.

(2) CnMCTRLm write

15	14	13	12	11	10	9	8
0	0	0	0	Set IE	0	Set TRQ	Set RDY
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

7	6	5	4	3	2	1	0
0	0	0	Clear MOW	Clear IE	Clear DN	Clear TRQ	Clear RDY
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

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.

Cautions

1. Always set the IE bit and RDY bit separately.
2. Do not set the DN bit to 1 by software. Be sure to write 0 to bit 10.
3. 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.
4. Do not clear the RDY bit (0) during message transmission. Follow the transmission abort process about clearing the RDY bit (0) to redefine the message buffer.
5. Clear again when RDY bit is not cleared even if this bit is cleared.
6. Be sure that RDY is cleared before writing to the other message buffer registers, by checking the status of the RDY bit.

17.8 CAN Controller Initialization

17.8.1 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 to initialize the CAN module, refer to *Section 17.16, Operation of CAN Controller* on page 504.

17.8.2 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.

17.8.3 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.

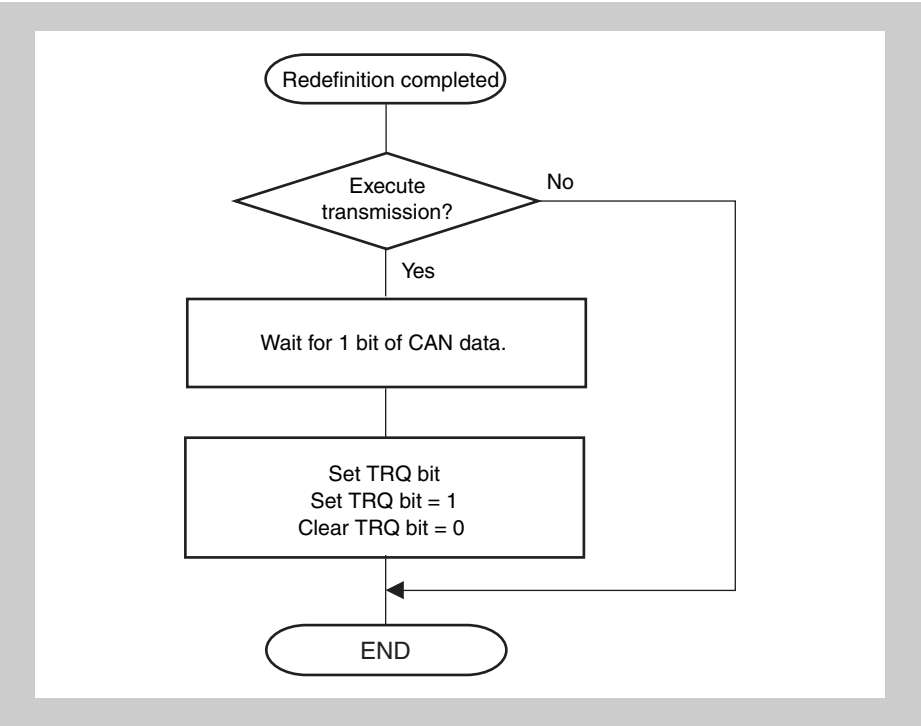
(2) To redefine message buffer during reception

Perform redefinition as shown in *Figure 17-38*.

(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 *Section (1), Transmission abort process except for in normal operation mode with automatic block transmission (ABT)* on page 484 and *Section (2), Transmission abort process except for ABT transmission in normal operation mode with automatic block transmission (ABT)* on page 484). 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 17-26 Setting transmission request (TRQ) to transmit message buffer after redefinition



Cautions

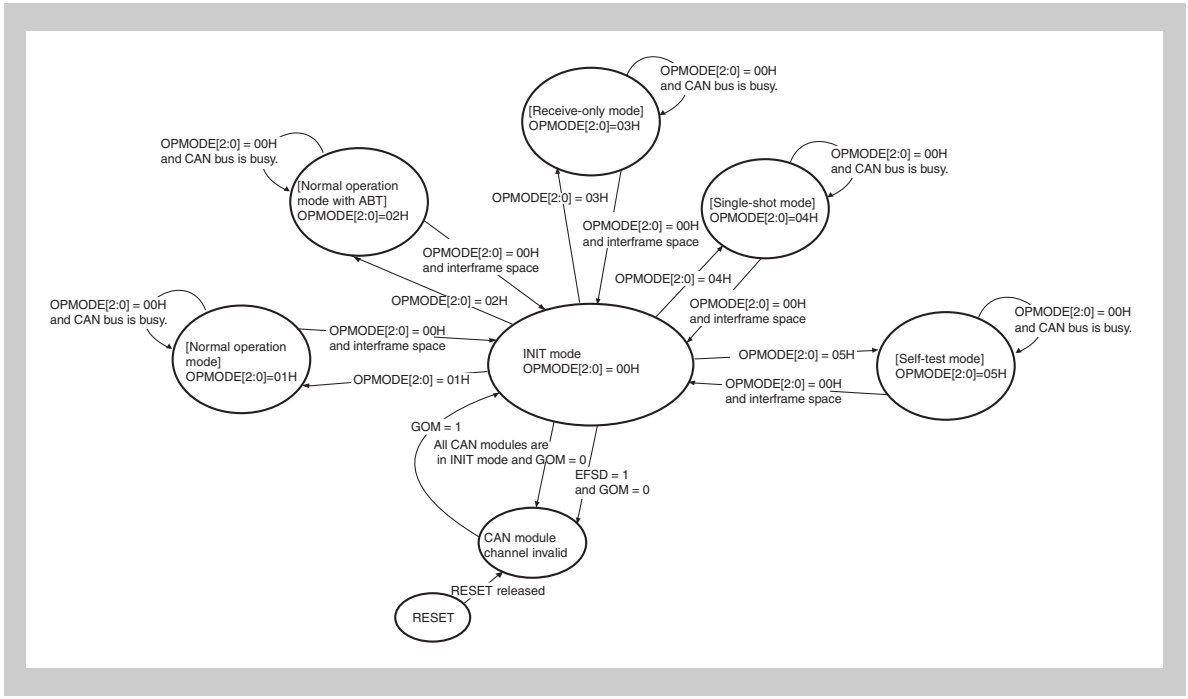
1. When a message is received, reception filtering is performed in accordance with the ID and mask set to each receive message buffer. If the procedure in *Figure 17-38 on page 507* is not observed, the contents of the message buffer after it has been redefined may contradict the result of reception (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 17-26 on page 470* is not observed, a message with an ID not having the highest priority may be transmitted after redefinition.

17.8.4 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 17-27 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 17-36 on page 504*).

17.8.5 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.

17.9 Message Reception

17.9.1 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 17-26 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

Table 17-26 MBRB priorities

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

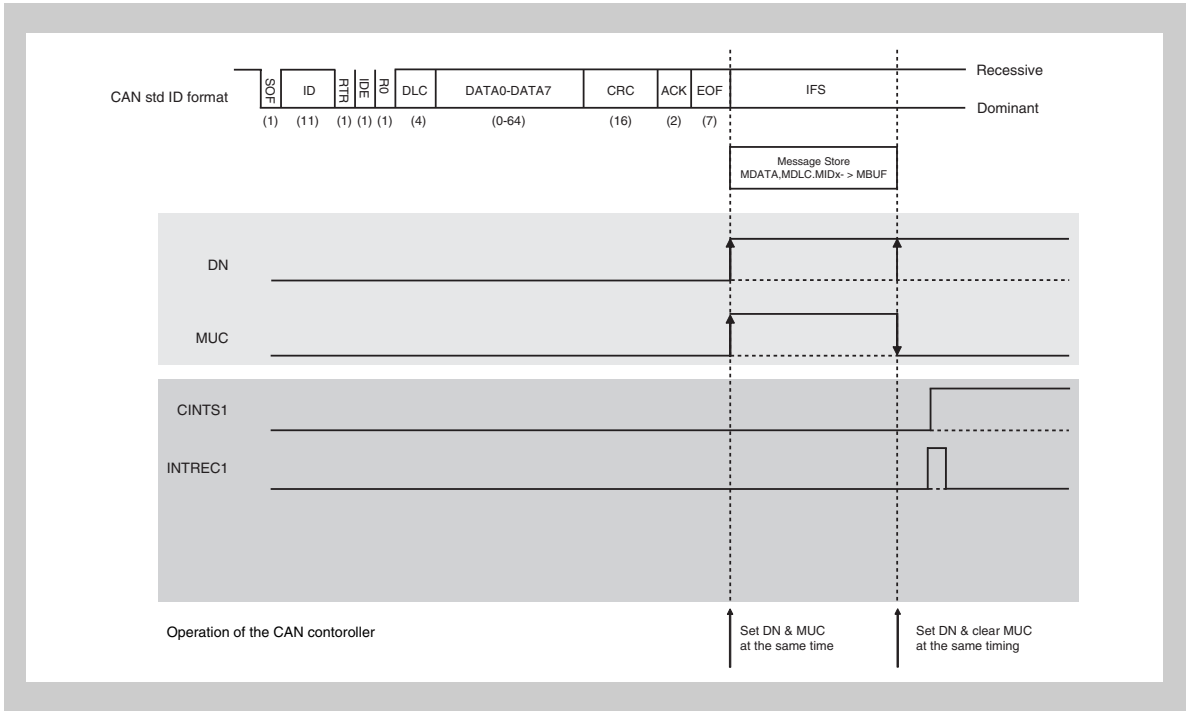
17.9.2 Receive data read

To keep data consistency when reading CAN message buffers, perform the data reading according to *Figure 17-49 on page 518* to *Figure 17-51 on page 520*.

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 17-28 on page 473*.)

The receive history list is also updated just before the storage process. In addition, during the 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 that the storage process may be disturbed (delayed) when the CPU accesses the message buffer.

Figure 17-28 DN and MUC bit setting period (for standard ID format)



17.9.3 Receive history list function

The receive history list function records in the receive history list (RHL) 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 used 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 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.

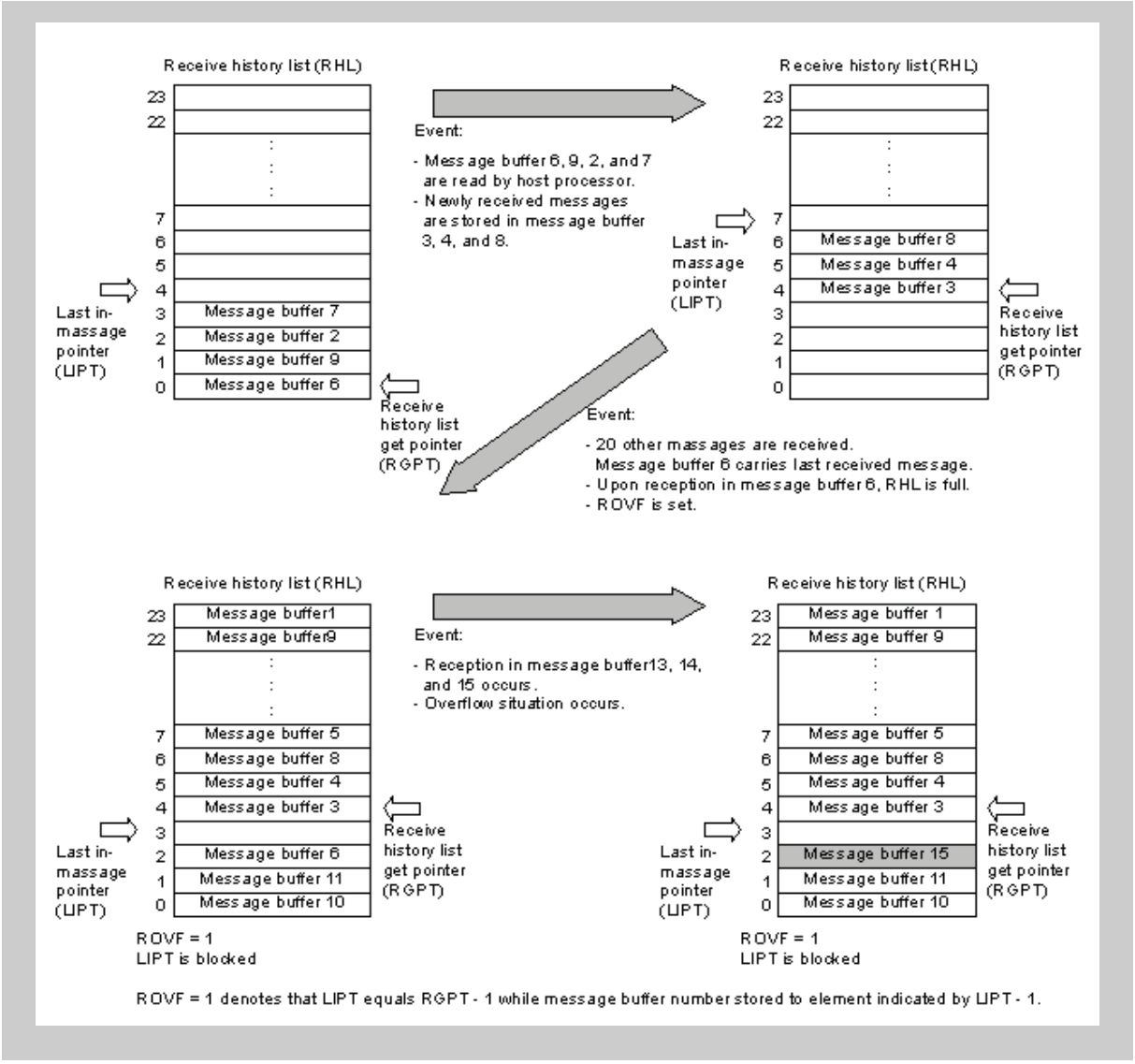
If the LIPT pointer is incremented and matches the value of the RGPT pointer minus 1, the ROVF bit (receive history list overflow) of the CnRGPT register is set to 1. This indicates that the RHL is full of numbers of message buffers that have not been read. When further message reception and storing 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 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 storage of a newly received data or remote frame. This may lead to the situation in which RHPM indicates an empty

history list although a frame has been received, 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.

Figure 17-29 Receive history list



17.9.4 Mask function

Any message buffer that is used for reception can be assigned to one of four global reception masks (or to no mask).

By using the mask function, the message ID comparison can be reduced by masked bits, allowing the reception of several different IDs into one buffer.

While the mask function is in effect, an identifier bit that is set to 1 by a mask in the received message will not be 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

1. 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.

17.9.5 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 for storing receive data in each message buffer for the MBRB are the same as the conditions of storing data in a single message buffer.

Cautions

1. MBRB can be configured for each of the same message buffer types. Therefore, even if a message buffer of another MBRB with a matching ID but a different buffer type 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 17-26*.

17.9.6 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 selected 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 stores 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.

17.10 Message Transmission

17.10.1 Message transmission

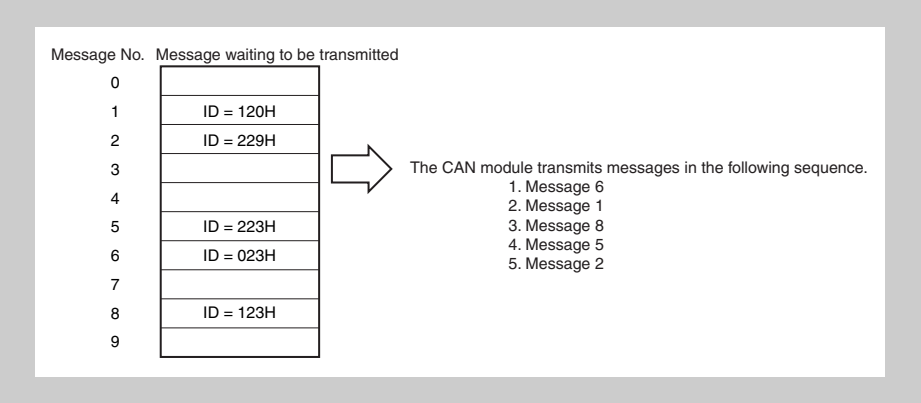
A message buffer with its TRQ bit set to 1 participates in the search for the most highly prioritized message when the following conditions are fulfilled. This behavior is valid for all operational modes.

- 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).

Figure 17-30 Message processing example



After the transmit message search, the CAN module transmits the 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).

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 lower priority has already started. If transmission of a message with a lower priority has already started, 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 two or more transmission-pending extended ID message frames have 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 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. Because the RDY flag may be locked temporarily during internal transfer actions, the status of RDY must be checked by software, after changing it.

17.10.2 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 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 numbers of the message buffers that have received and stored a frame will be recorded chronologically.

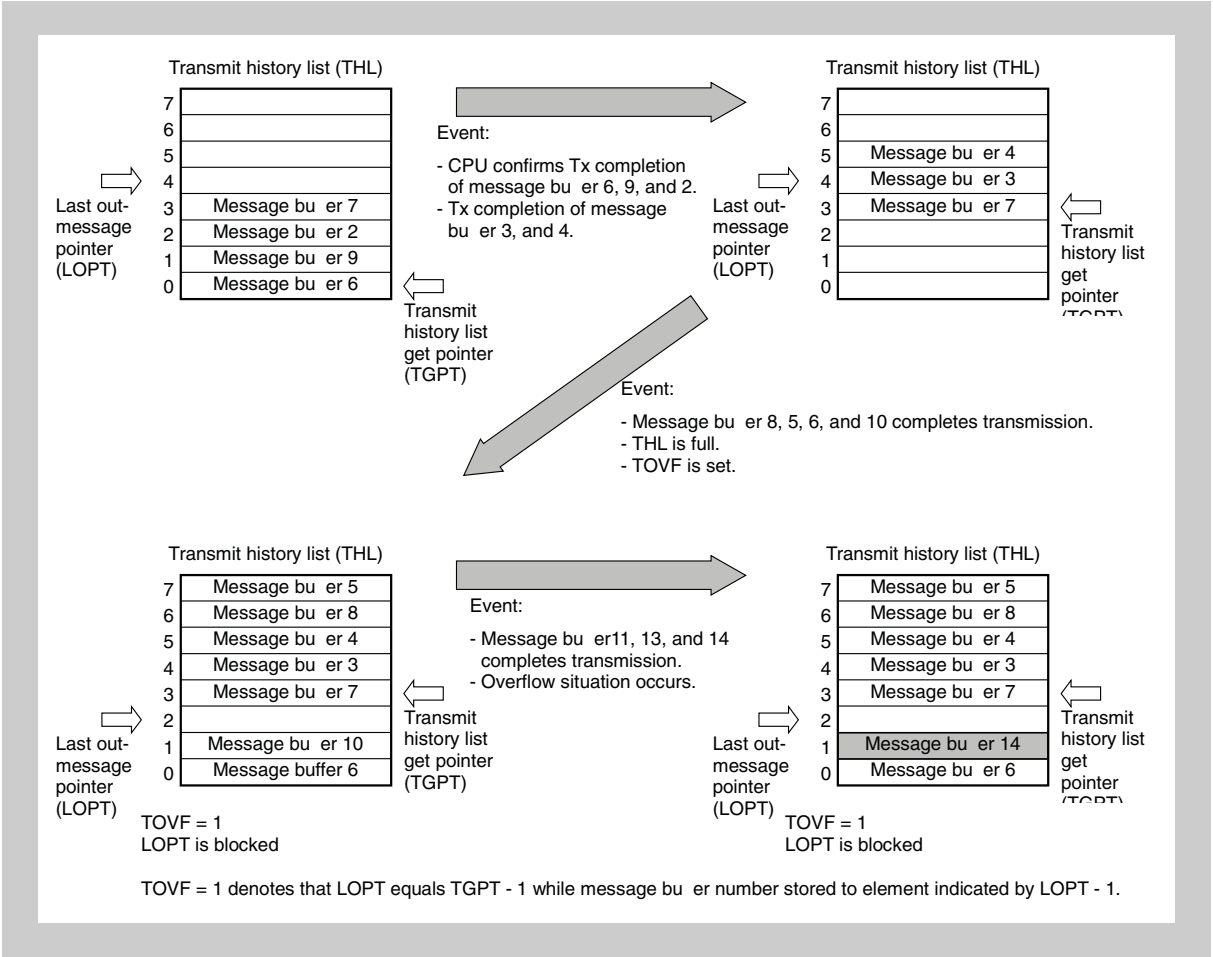
The TGPT pointer is utilized as a read pointer that reads a recorded message buffer number from the THL. This pointer indicates the first THL element that the CPU has not yet read. By reading the CnTGPT register by software, the number of a message buffer that has completed transmission can be read. 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. 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 THPM indicating 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).

Figure 17-31 Transmit history list



17.10.3 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 initializing message buffers for ABT, set the RDY bit (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).

Cautions

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.

17.10.4 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 17-45 on page 514*).

(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 17-46 on page 515*).

(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 17-47 on page 516*). 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 17-48 on page 517*).

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.

17.10.5 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.

17.11 Power Saving Modes

17.11.1 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.

1. 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

2. 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.

3. 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.

Similarly, if a sleep mode request is pending, and at the same time a message is transmitted in a message box, the sleep mode request is not cancelled, but is executed. This may result in CAN being in sleep mode, while the CPU would execute the transmit interrupt routine. Therefore, the interrupt routine must check the access to the message buffers as well as transmission 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.

Cautions

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.

17.11.2 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.

17.11.3 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).

17.12 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 17-27 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).

17.13 Diagnosis Functions and Special Operational Modes

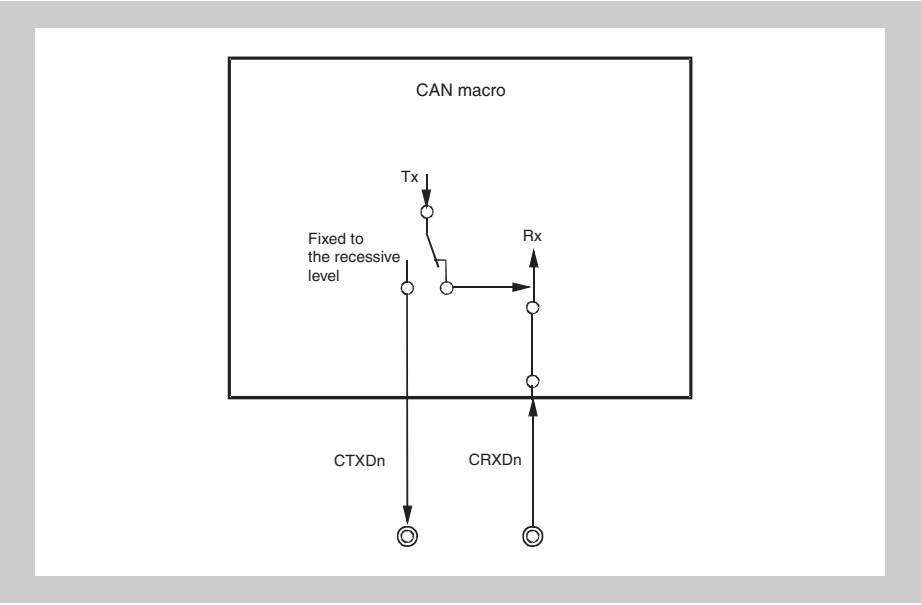
The CAN module provides a receive-only mode, single-shot mode, and self-test mode to support CAN bus diagnosis functions or the operation of special CAN communication methods.

17.13.1 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 17-32 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.

17.13.2 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.

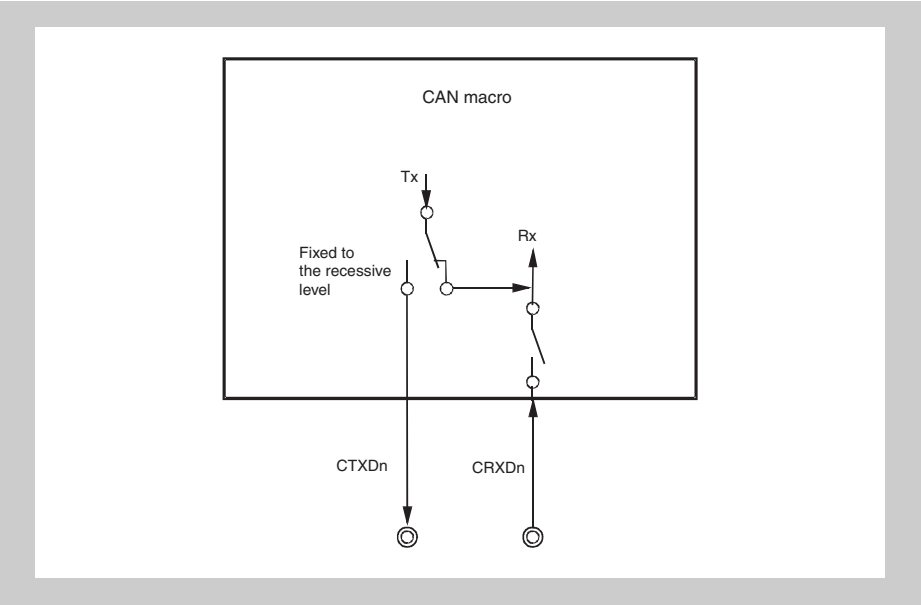
17.13.3 Self-test mode

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 17-33 CAN module terminal connection in self-test mode



17.13.4 Receive/transmit operation in each operation mode

The following table shows outline of the receive/transmit operation in each operation mode.

Table 17-28 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

Table 17-28 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
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.

17.14 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.

17.14.1 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.

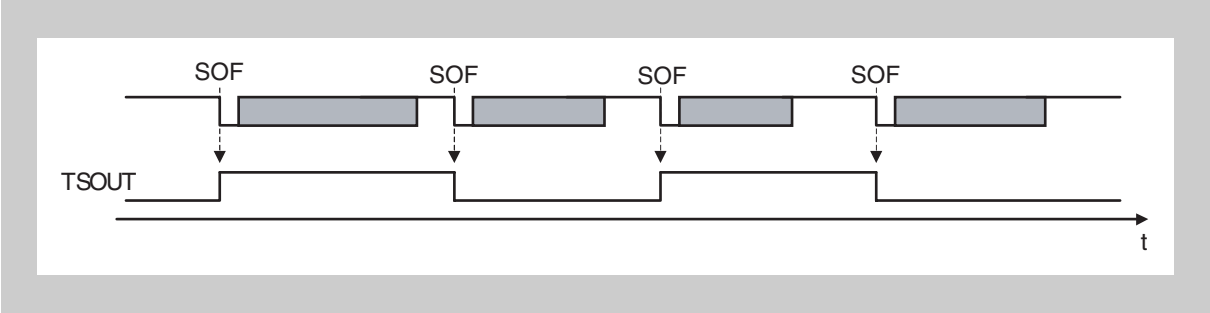


Figure 17-34 Timing diagram of capture signal TSOUT

The TSOUT signal toggles its level upon occurrence of the selected event during data frame reception (in *Figure 17-34*, 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.

17.15 Baud Rate Settings

17.15.1 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 17-29 shows the combinations of bit rates that satisfy the above conditions.

Table 17-29 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 17-29 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 17-29 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 17-29* 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.

17.15.2 Representative examples of baud rate settings

Table 17-30 and *Table 17-31* show representative examples of baud rate settings.

Table 17-30 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	0000 0000	8	1	1	3	3	0011	010	62.5
1000	1	0000 0000	8	1	3	2	2	0100	001	75.0
1000	1	0000 0000	8	1	5	1	1	0101	000	87.5
500	1	0000 0000	16	1	1	7	7	0111	110	56.3
500	1	0000 0000	16	1	3	6	6	1000	101	62.5
500	1	0000 0000	16	1	5	5	5	1001	100	68.8
500	1	0000 0000	16	1	7	4	4	1010	011	75.0
500	1	0000 0000	16	1	9	3	3	1011	010	81.3
500	1	0000 0000	16	1	11	2	2	1100	001	87.5
500	1	0000 0000	16	1	13	1	1	1101	000	93.8
500	2	0000 0001	8	1	1	3	3	0011	010	62.5
500	2	0000 0001	8	1	3	2	2	0100	001	75.0
500	2	0000 0001	8	1	5	1	1	0101	000	87.5
250	2	0000 0001	16	1	1	7	7	0111	110	56.3
250	2	0000 0001	16	1	3	6	6	1000	101	62.5
250	2	0000 0001	16	1	5	5	5	1001	100	68.8
250	2	0000 0001	16	1	7	4	4	1010	011	75.0
250	2	0000 0001	16	1	9	3	3	1011	010	81.3
250	2	0000 0001	16	1	11	2	2	1100	001	87.5
250	2	0000 0001	16	1	13	1	1	1101	000	93.8
250	4	0000 0011	8	1	3	2	2	0100	001	75.0
250	4	0000 0011	8	1	5	1	1	0101	000	87.5
125	4	0000 0011	16	1	1	7	7	0111	110	56.3
125	4	0000 0011	16	1	3	6	6	1000	101	62.5
125	4	0000 0011	16	1	5	5	5	1001	100	68.8
125	4	0000 0011	16	1	7	4	4	1010	011	75.0
125	4	0000 0011	16	1	9	3	3	1011	010	81.3
125	4	0000 0011	16	1	11	2	2	1100	001	87.5
125	4	0000 0011	16	1	13	1	1	1101	000	93.8
125	8	0000 0111	8	1	3	2	2	0100	001	75.0
125	8	0000 0111	8	1	5	1	1	0101	000	87.5
100	4	0000 0011	20	1	7	6	6	1100	101	70.0
100	4	0000 0011	20	1	9	5	5	1101	100	75.0
100	5	0000 0100	16	1	7	4	4	1010	011	75.0
100	5	0000 0100	16	1	9	3	3	1011	010	81.3

Table 17-30 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	0000 0111	10	1	3	3	3	0101	010	70.0
100	8	0000 0111	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	0000 0011	24	1	7	8	8	1110	111	66.7
83.3	4	0000 0011	24	1	9	7	7	1111	110	70.8
83.3	6	0000 0101	16	1	5	5	5	1001	100	68.8
83.3	6	0000 0101	16	1	7	4	4	1010	011	75.0
83.3	6	0000 0101	16	1	9	3	3	1011	010	81.3
83.3	6	0000 0101	16	1	11	2	2	1100	001	87.5
83.3	8	0000 0111	12	1	5	3	3	0111	010	75.0
83.3	8	0000 0111	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 17-30* 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 17-31 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	0000 0000	16	1	1	7	7	0111	110	56.3
1000	1	0000 0000	16	1	3	6	6	1000	101	62.5
1000	1	0000 0000	16	1	5	5	5	1001	100	68.8
1000	1	0000 0000	16	1	7	4	4	1010	011	75.0
1000	1	0000 0000	16	1	9	3	3	1011	010	81.3
1000	1	0000 0000	16	1	11	2	2	1100	001	87.5
1000	1	0000 0000	16	1	13	1	1	1101	000	93.8
1000	2	0000 0001	8	1	3	2	2	0100	001	75.0
1000	2	0000 0001	8	1	5	1	1	0101	000	87.5
500	2	0000 0001	16	1	1	7	7	0111	110	56.3
500	2	0000 0001	16	1	3	6	6	1000	101	62.5
500	2	0000 0001	16	1	5	5	5	1001	100	68.8
500	2	0000 0001	16	1	7	4	4	1010	011	75.0
500	2	0000 0001	16	1	9	3	3	1011	010	81.3
500	2	0000 0001	16	1	11	2	2	1100	001	87.5
500	2	0000 0001	16	1	13	1	1	1101	000	93.8
500	4	0000 0011	8	1	3	2	2	0100	001	75.0
500	4	0000 0011	8	1	5	1	1	0101	000	87.5
250	4	0000 0011	16	1	3	6	6	1000	101	62.5
250	4	0000 0011	16	1	5	5	5	1001	100	68.8
250	4	0000 0011	16	1	7	4	4	1010	011	75.0
250	4	0000 0011	16	1	9	3	3	1011	010	81.3
250	4	0000 0011	16	1	11	2	2	1100	001	87.5
250	8	0000 0111	8	1	3	2	2	0100	001	75.0
250	8	0000 0111	8	1	5	1	1	0101	000	87.5
125	8	0000 0111	16	1	3	6	6	1000	101	62.5
125	8	0000 0111	16	1	7	4	4	1010	011	75.0
125	8	0000 0111	16	1	9	3	3	1011	010	81.3
125	8	0000 0111	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	0000 0111	20	1	9	5	5	1101	100	75.0
100	8	0000 0111	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 17-31 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	0000 0111	24	1	7	8	8	1110	111	66.7
83.3	8	0000 0111	24	1	9	7	7	1111	110	70.8
83.3	12	0000 1011	16	1	7	4	4	1010	011	75.0
83.3	12	0000 1011	16	1	9	3	3	1011	010	81.3
83.3	12	0000 1011	16	1	11	2	2	1100	001	87.5
83.3	16	0000 1111	12	1	5	3	3	0111	010	75.0
83.3	16	0000 1111	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 17-31* 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.

17.16 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.

Figure 17-35 Initialization

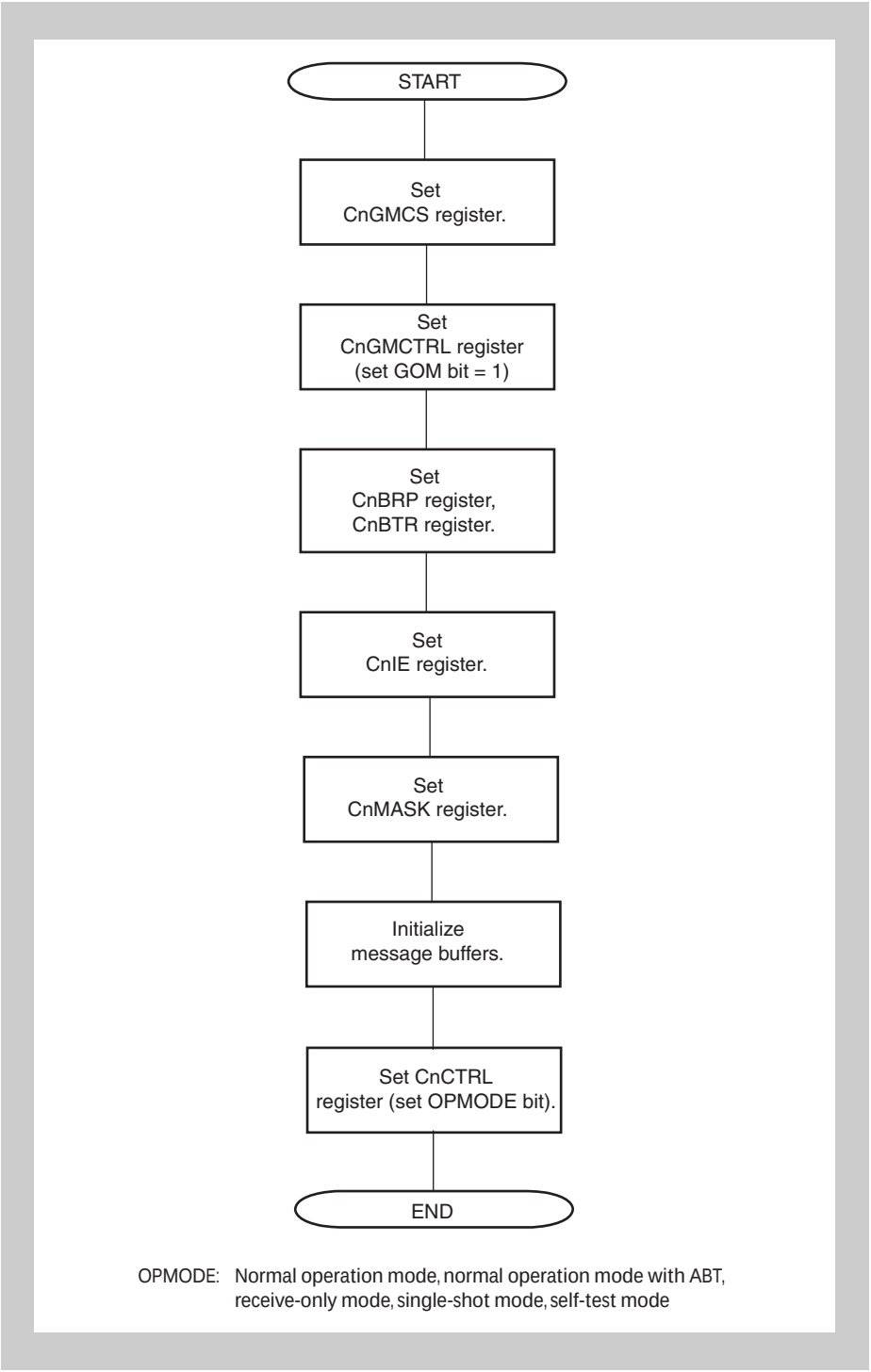
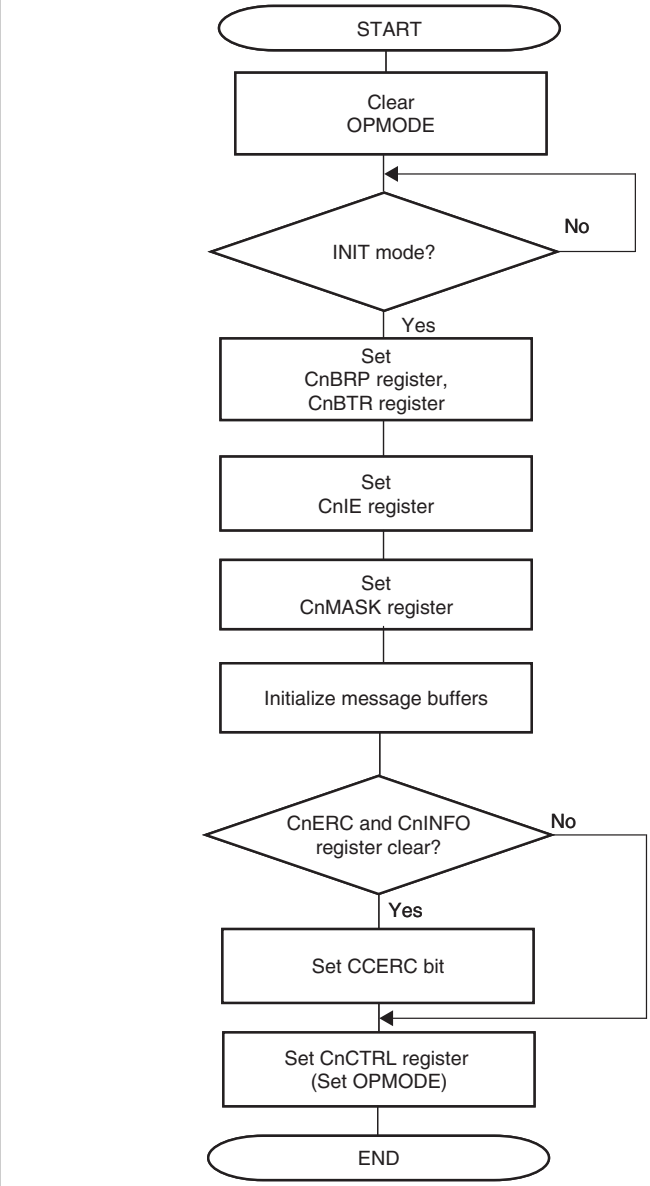


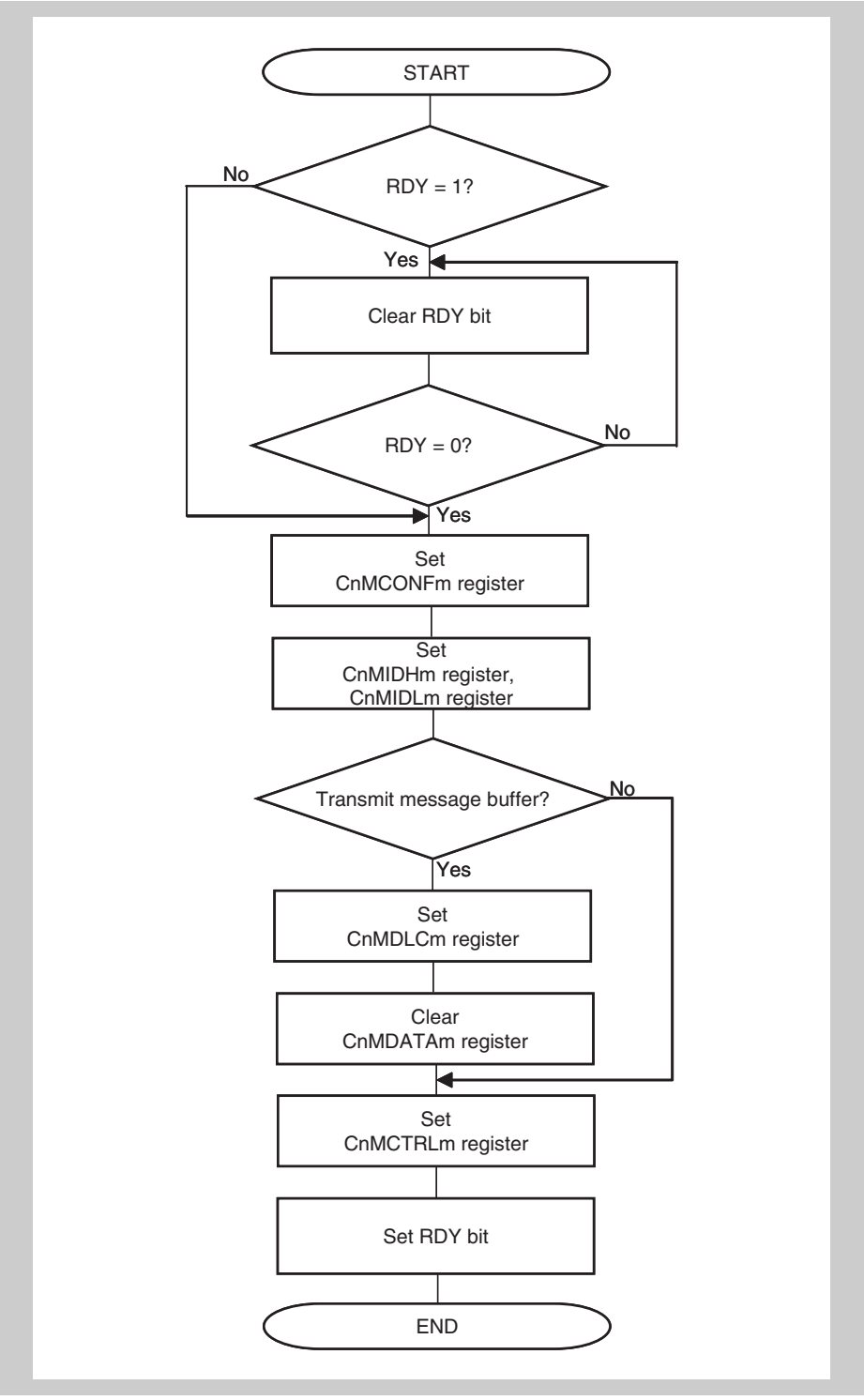
Figure 17-36 Re-initialization



OPMODE: Normal operation mode, normal operation mode with ABT, receive-only mode, single-shot mode, self-test mode

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 17-37 Message buffer initialization

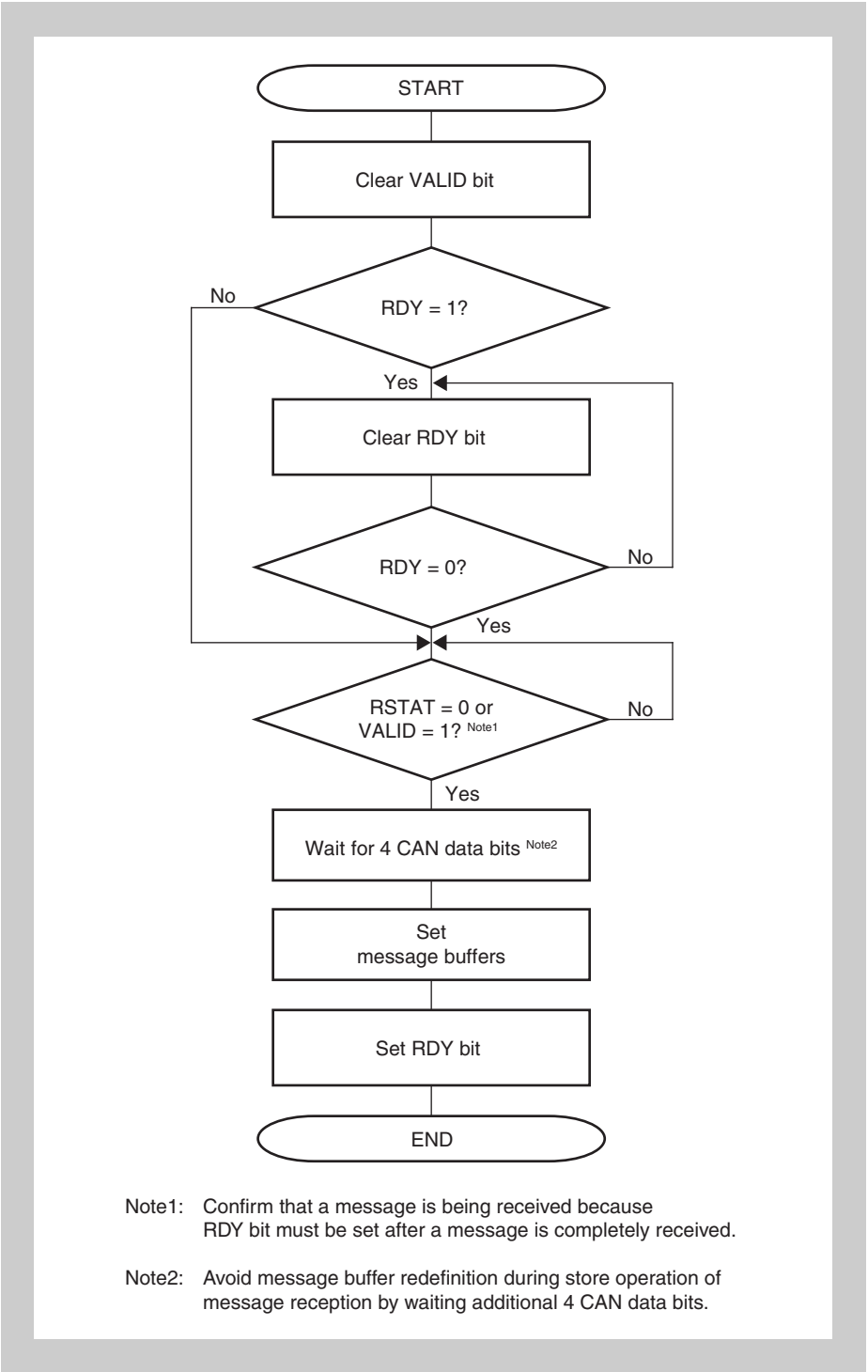


Cautions

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 17-38 shows the processing for a receive message buffer (MT[2:0] bits of CnMCONFm register = 001_B to 101_B).

Figure 17-38 Message buffer redefinition



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 17-39 shows the processing for a transmit message buffer during transmission (MT[2:0] bits of CnMCONFm register = 000_B).

Figure 17-39 Message buffer redefinition during transmission

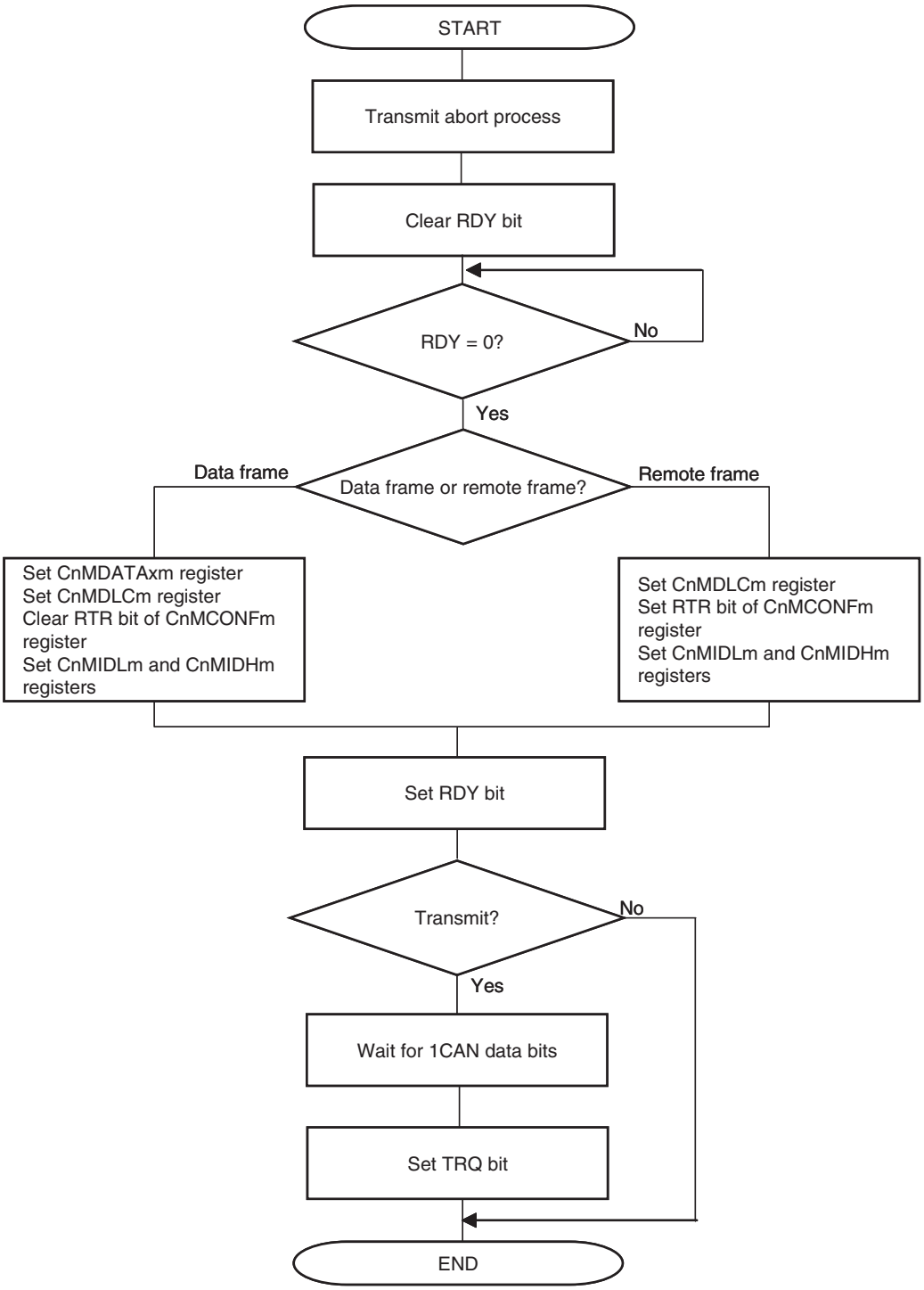
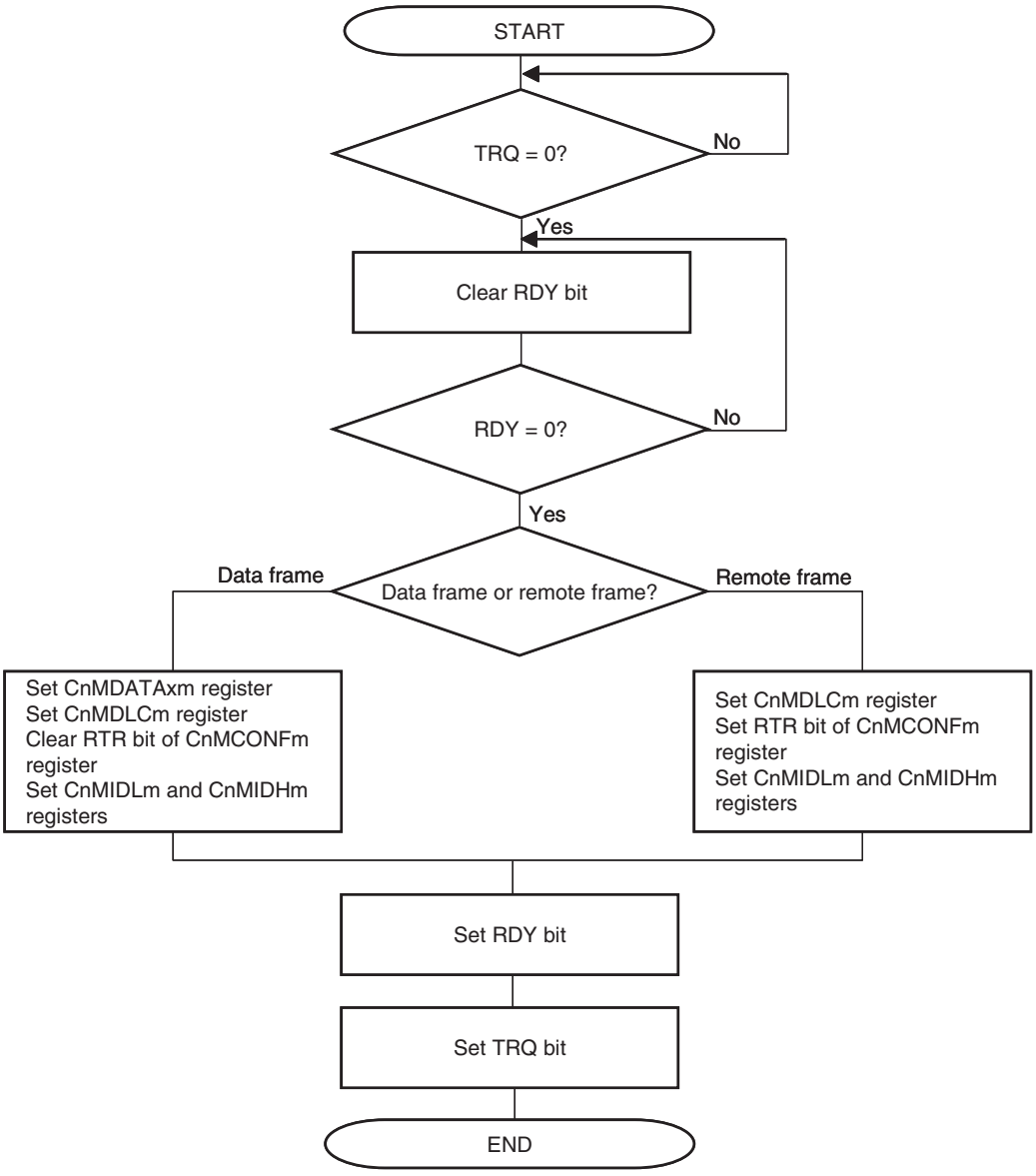


Figure 17-40 shows the processing for a transmit message buffer (MT[2:0] bits of CnMCONFm register = 000_B).

Figure 17-40 Message transmit processing

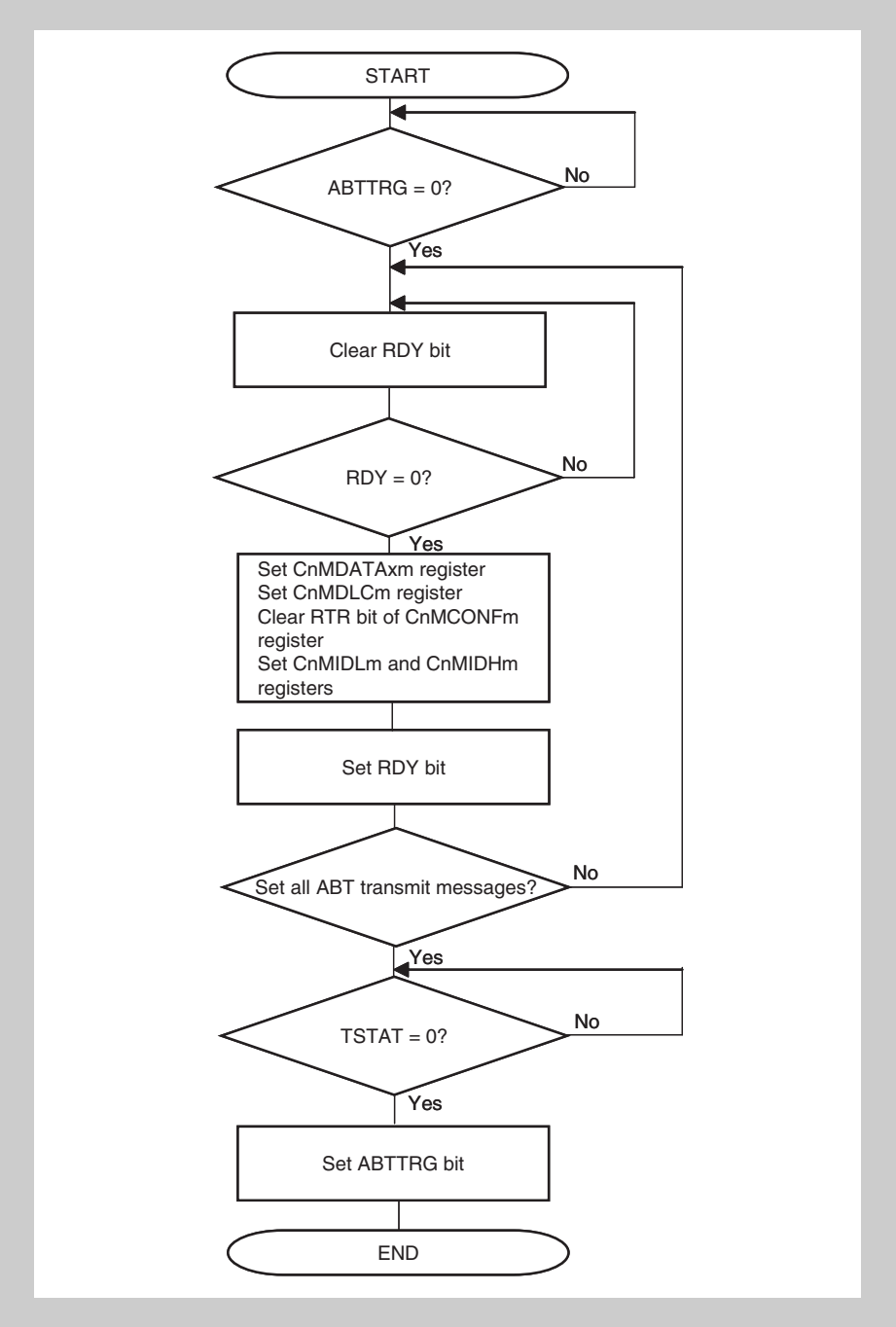


Cautions

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 17-41 shows the processing for a transmit message buffer (MT[2:0] bits of CnMCONFm register = 000_B)

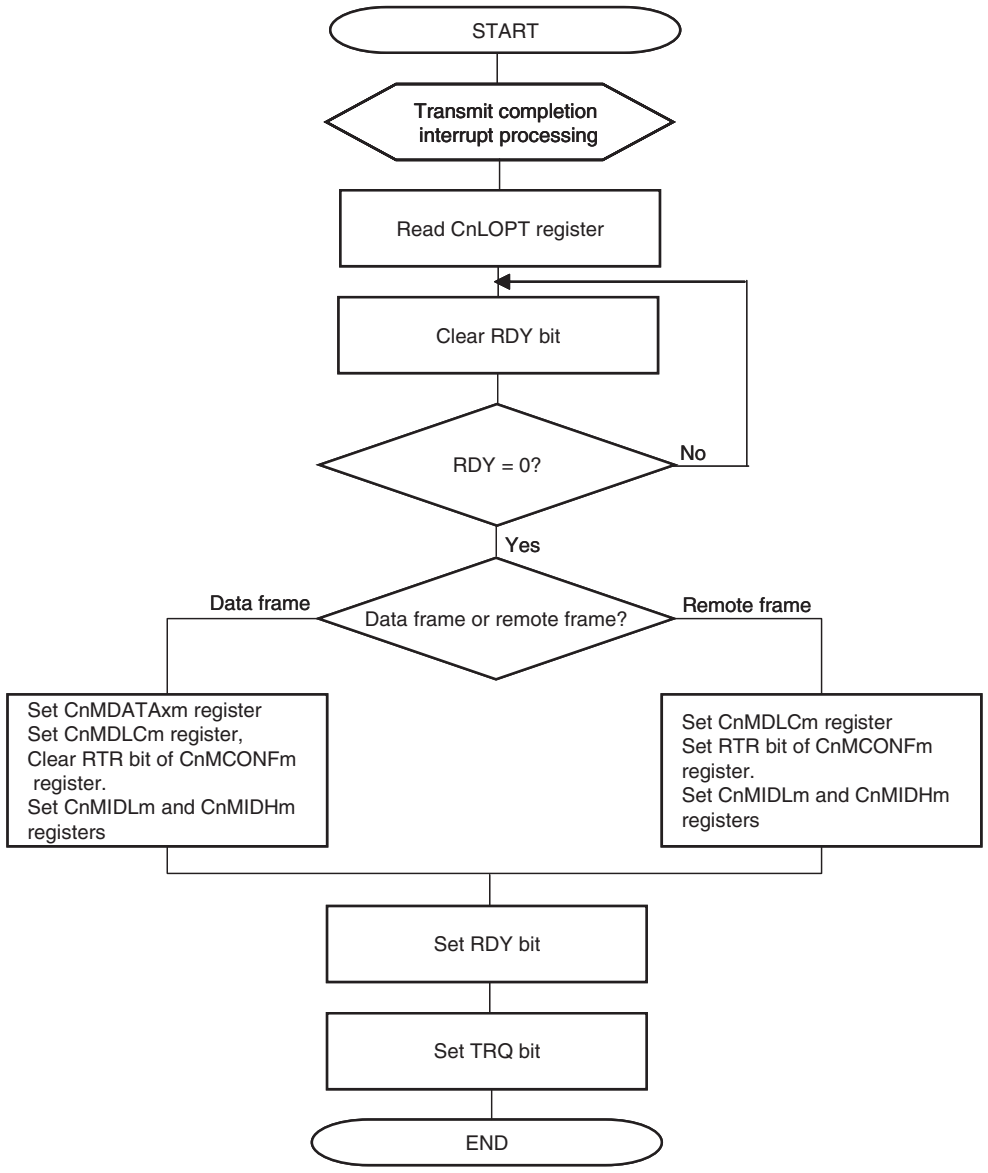
Figure 17-41 ABT message transmit processing



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 17-40 on page 509*.

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.

Figure 17-42 Transmission via interrupt (using CnLOPT register)



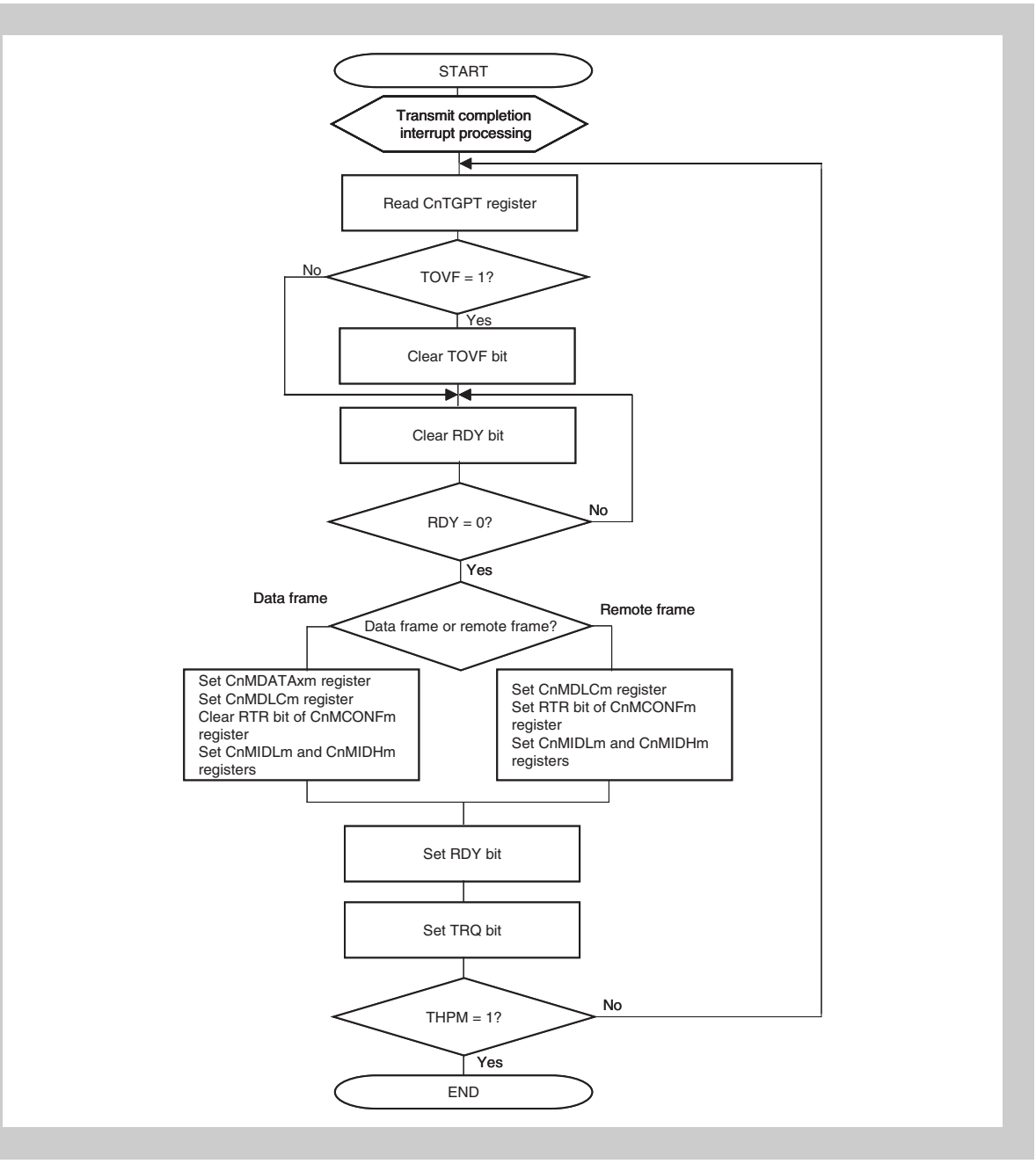
Cautions

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 17-43 Transmission via interrupt (using CnTGPT register)



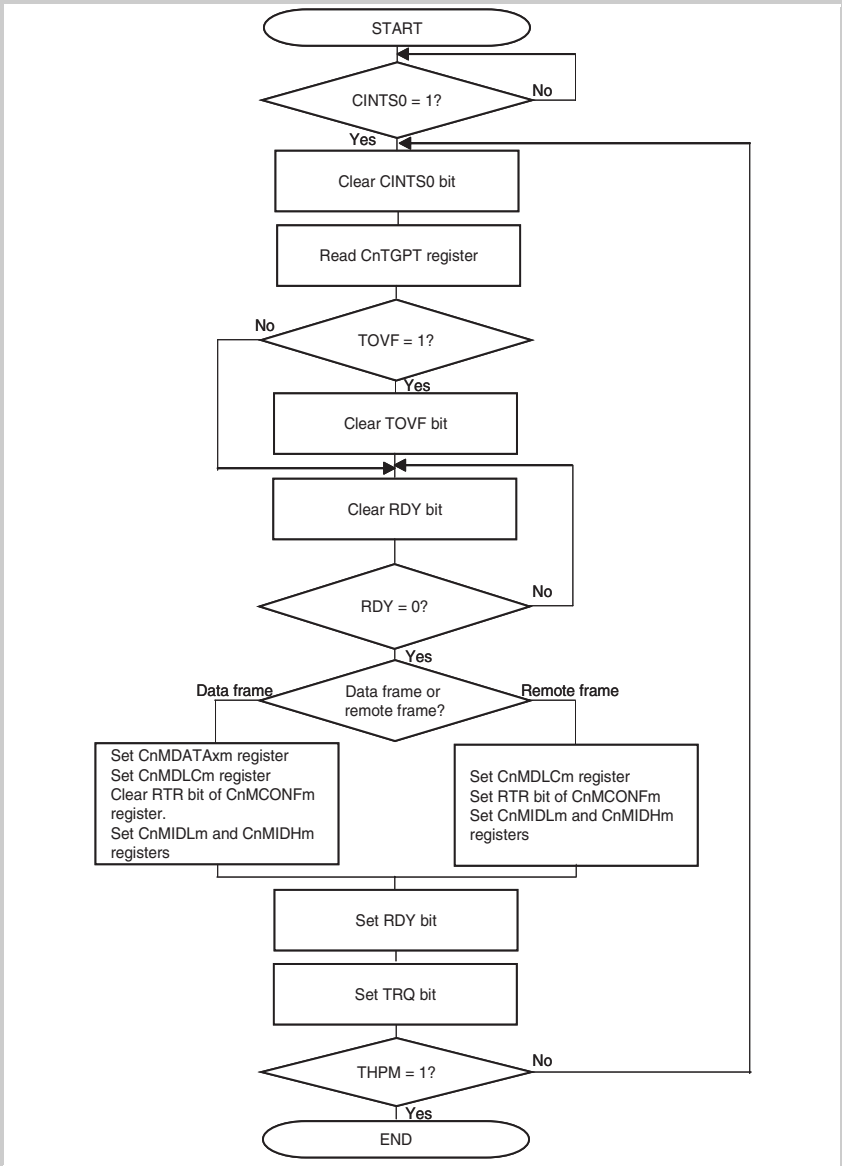
Cautions

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 17-44 Transmission via software polling



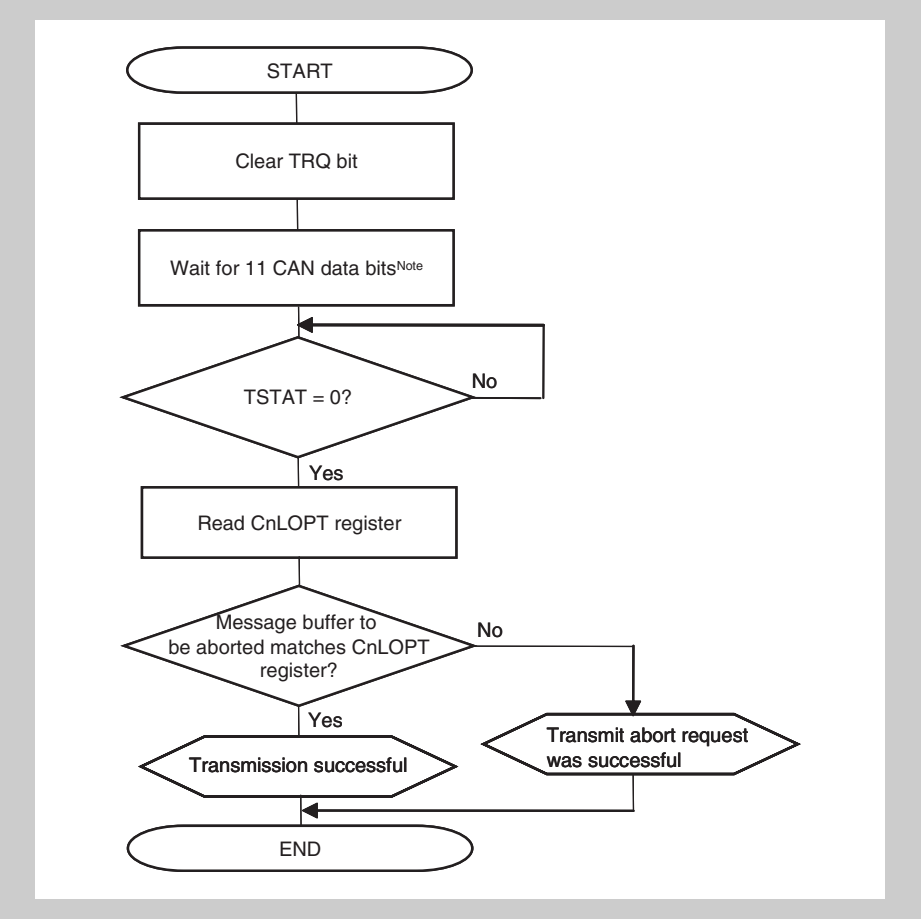
Cautions 1. The TRQ bit should be set after the RDY bit is set.

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.

Figure 17-45 Transmission abort processing (except normal operation mode with ABT)



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).

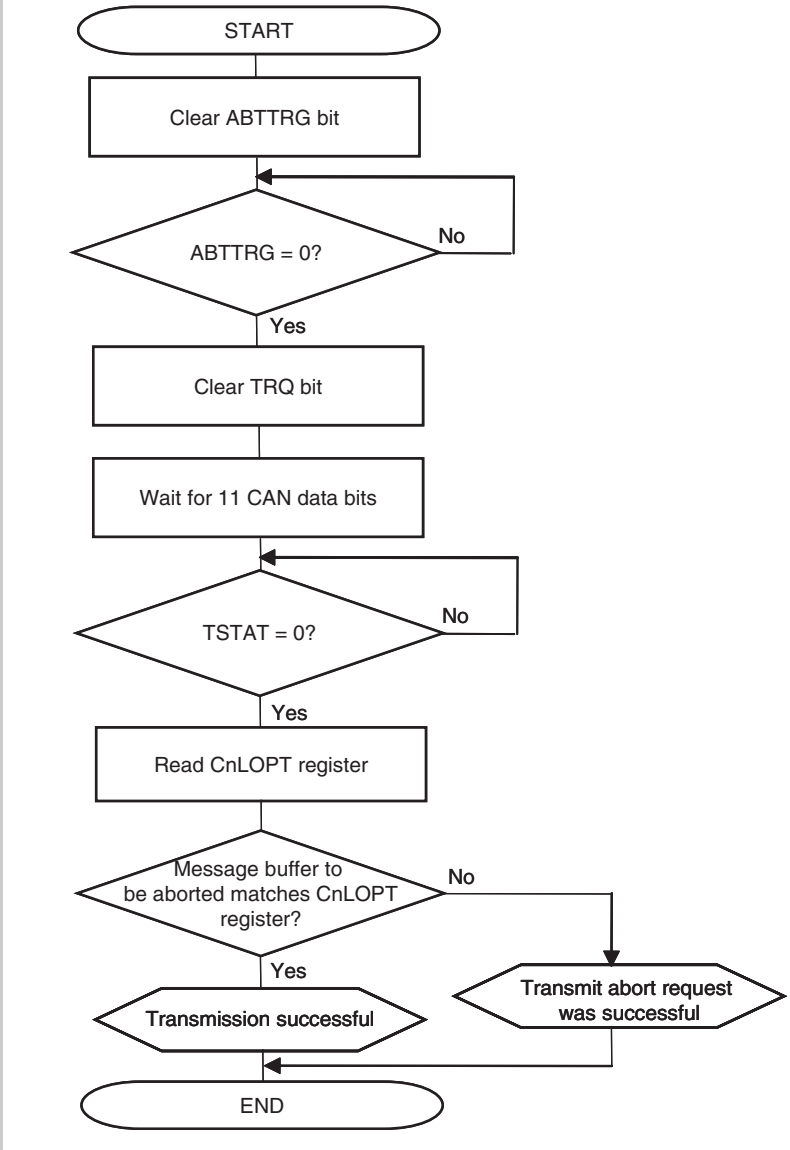
Cautions 1. Clear the TRQ bit for aborting transmission request, not the RDY bit.

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 17-46 Transmission abort processing except for ABT transmission (normal operation mode with ABT)

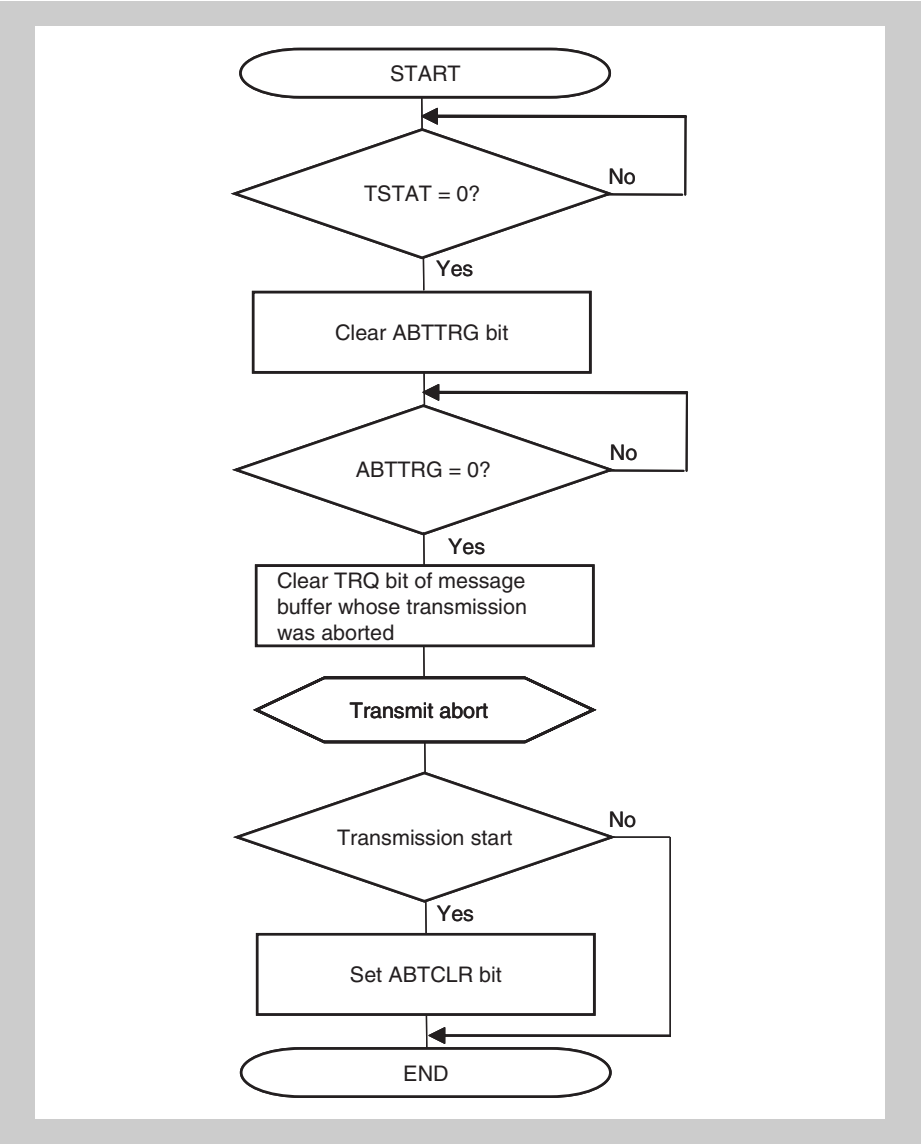


Cautions

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 17-47 shows the processing to skip resumption of transmitting a message that was stopped when transmission of an ABT message buffer was aborted.

Figure 17-47 Transmission abort processing (normal operation mode with ABT)

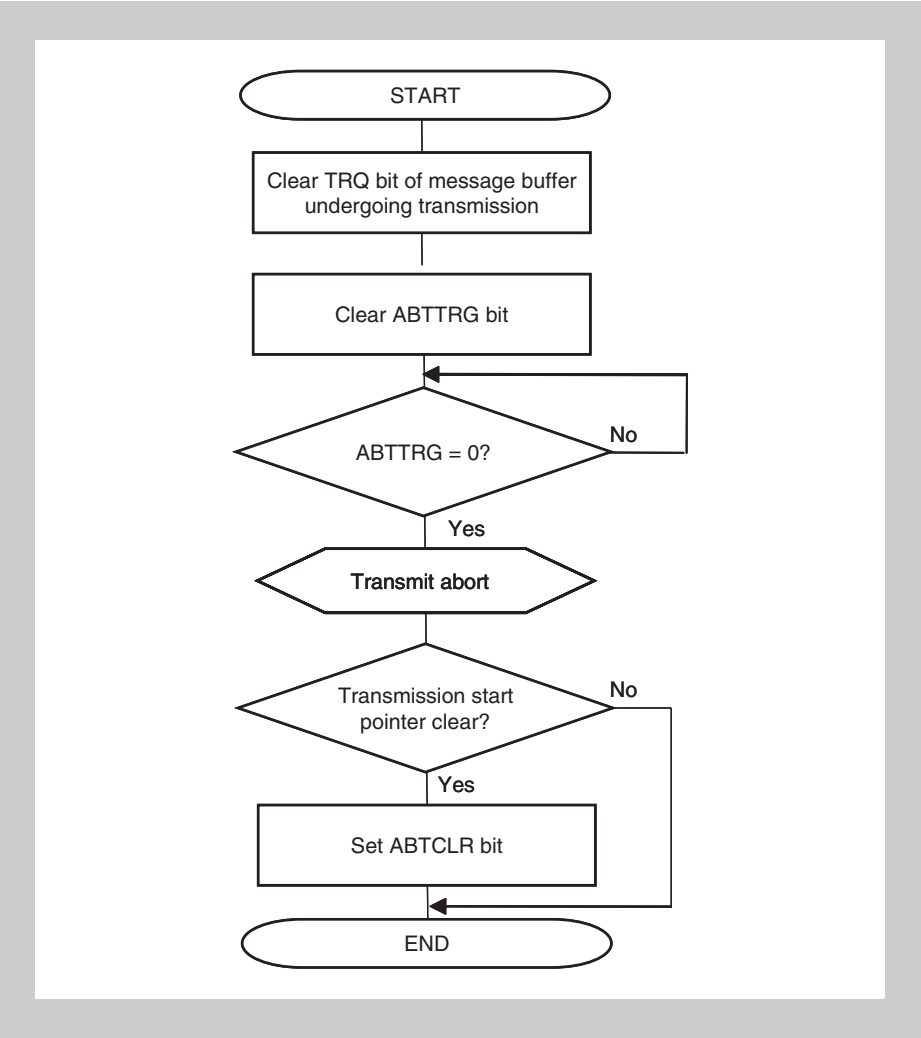


Cautions

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 17-47* or *Figure 17-48*. When clearing a transmission request in an area other than the ABT area, follow the procedure shown in *Figure 17-45 on page 514*.

Figure 17-48 shows the processing to not skip resumption of transmitting a message that was stopped when transmission of an ABT message buffer was aborted.

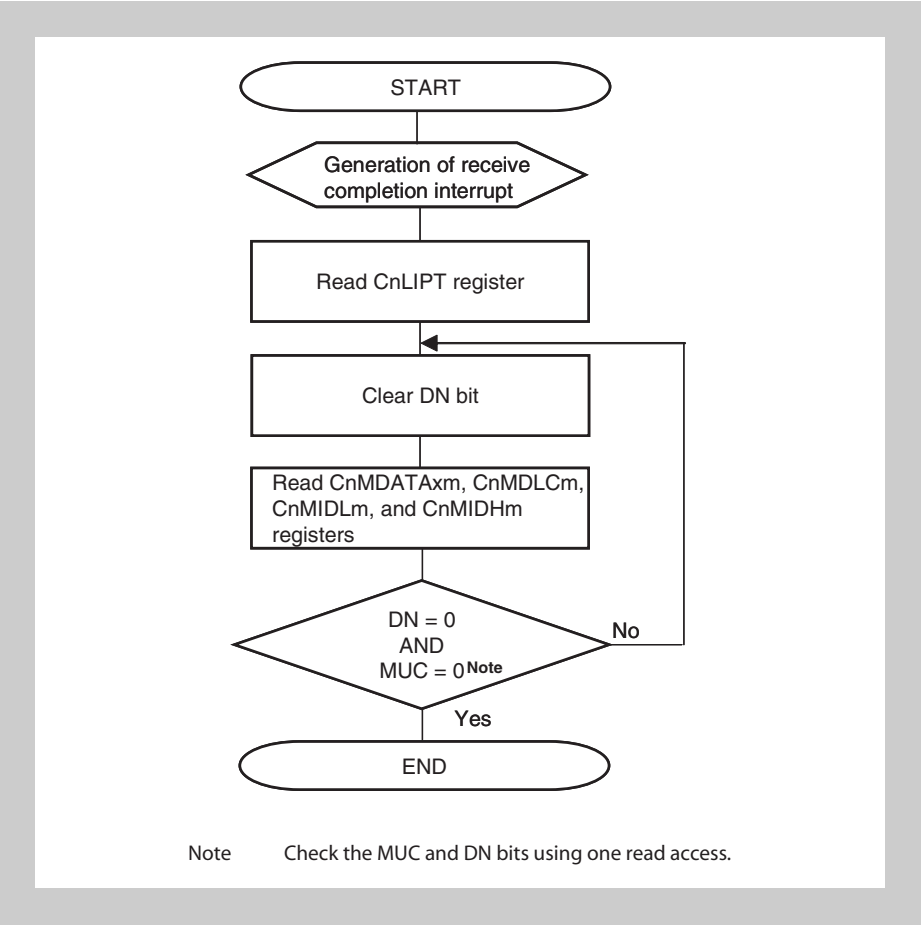
Figure 17-48 ABT transmission request abort processing (normal operation mode with ABT)



Cautions

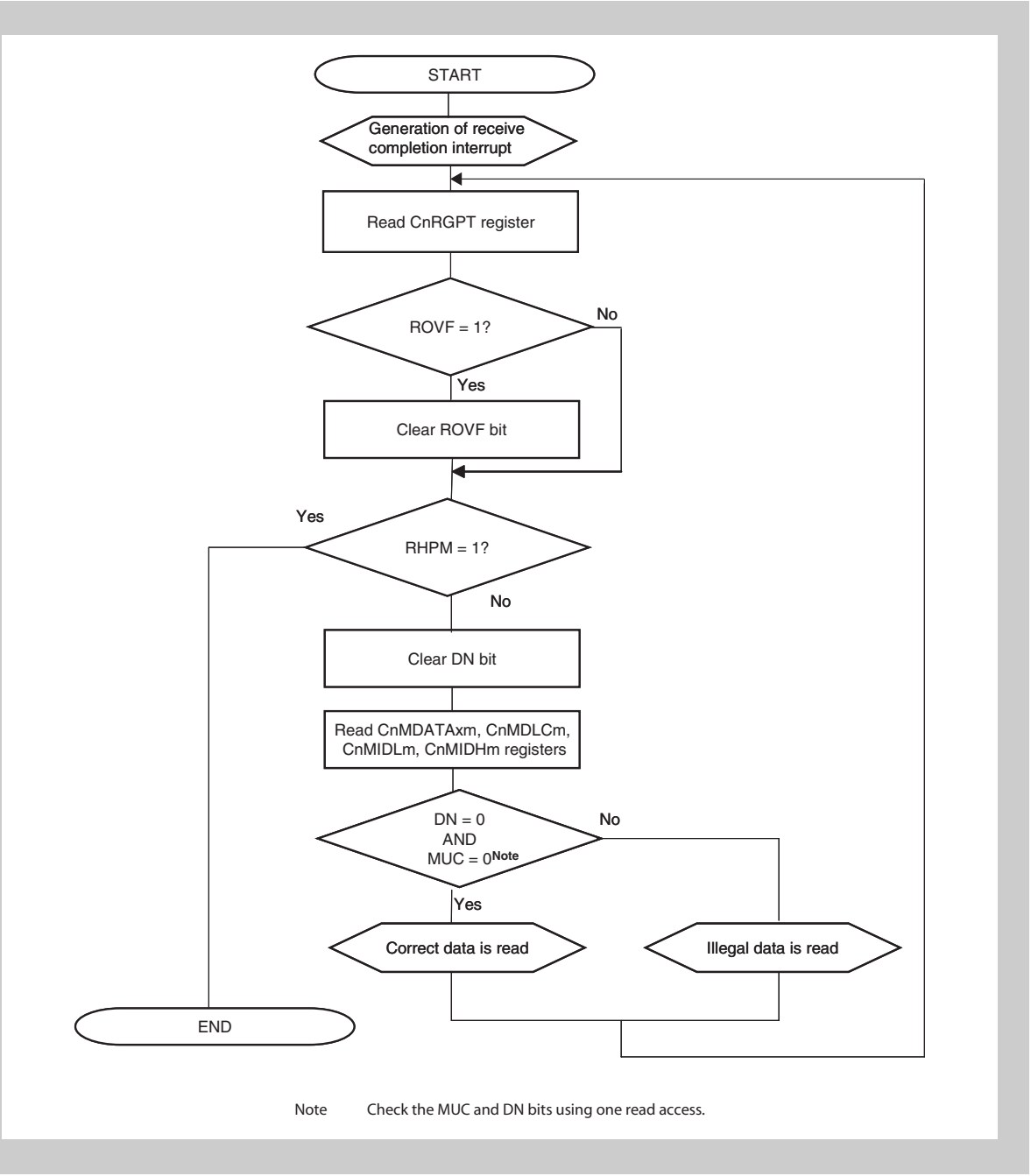
1. Do not set any transmission requests while ABT transmission abort processing is in progress.
2. Make a CAN sleep mode/CAN stop mode request after the ABTTRG bit is cleared (after ABT mode is stopped) following the procedure shown in *Figure 17-47* or *Figure 17-48*. When clearing a transmission request in an area other than the ABT area, follow the procedure shown in *Figure 17-45 on page 514*.

Figure 17-49 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.

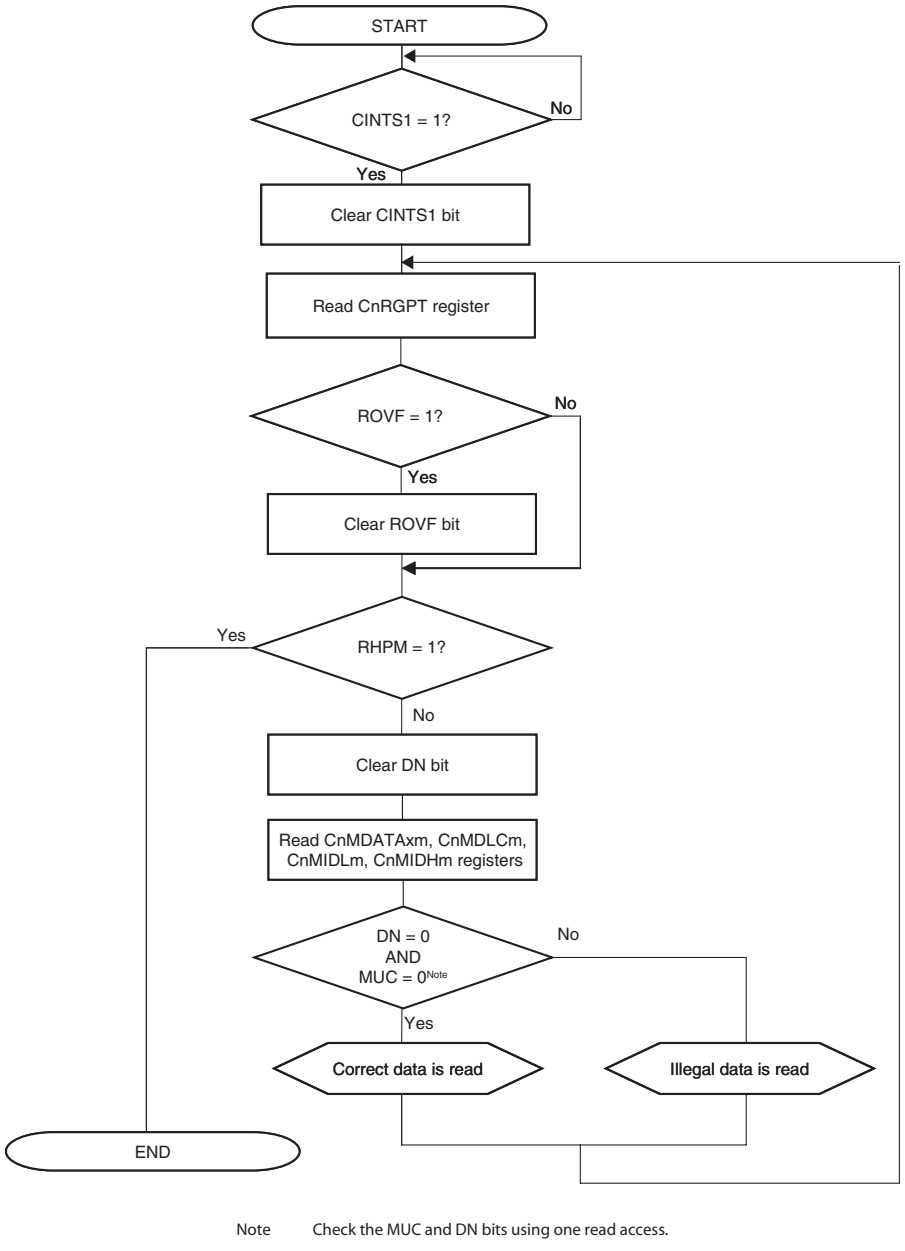
Figure 17-50 Reception via interrupt (using CnRGPT register)



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.

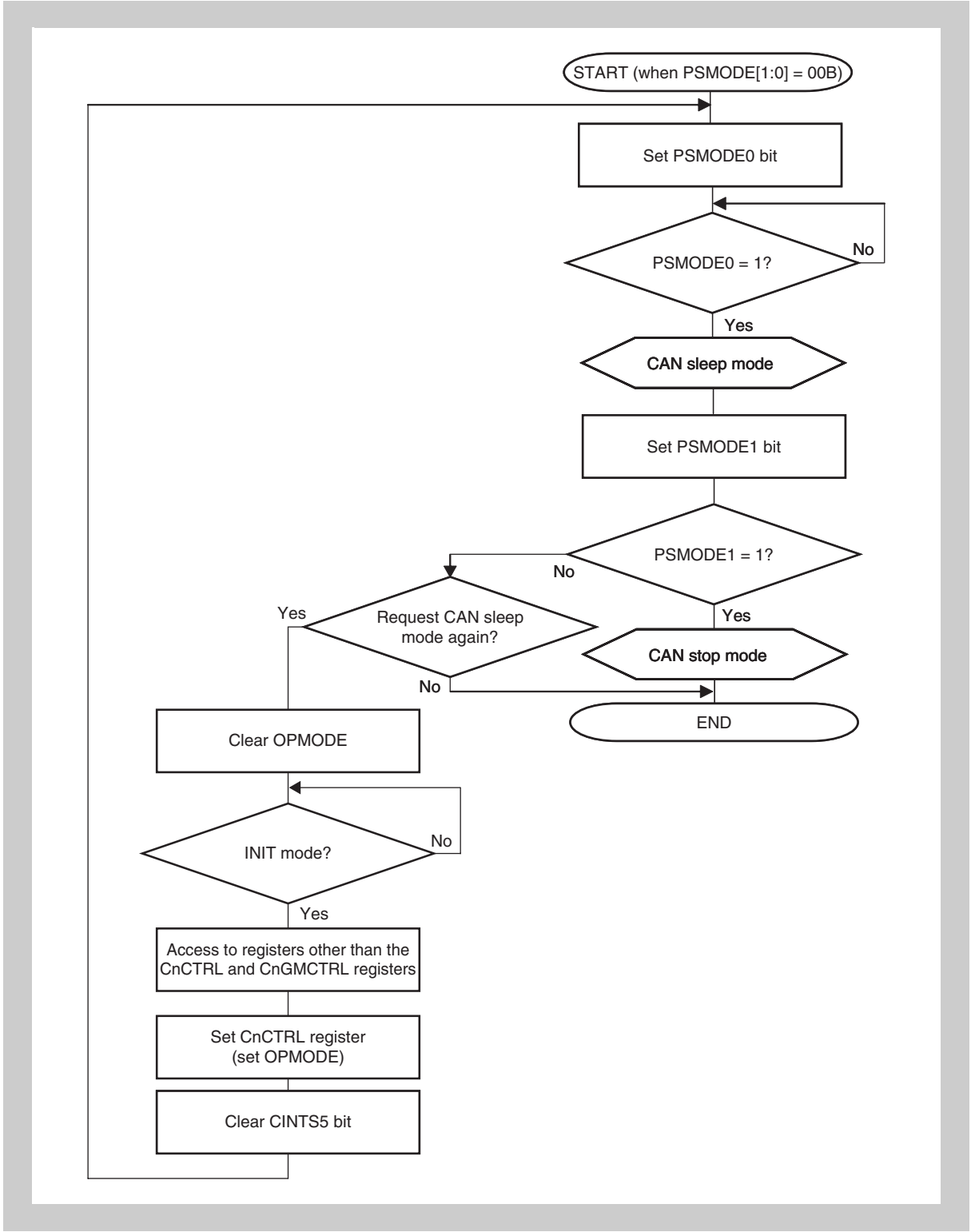
Figure 17-51 Reception via software polling



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.

Figure 17-52 Setting CAN sleep mode/stop mode



Caution To abort transmission before making a request for the CAN sleep mode, perform processing according to *Figure 17-45 on page 514* and *Figure 17-47 on page 516*.

Figure 17-53 Clear CAN sleep/stop mode

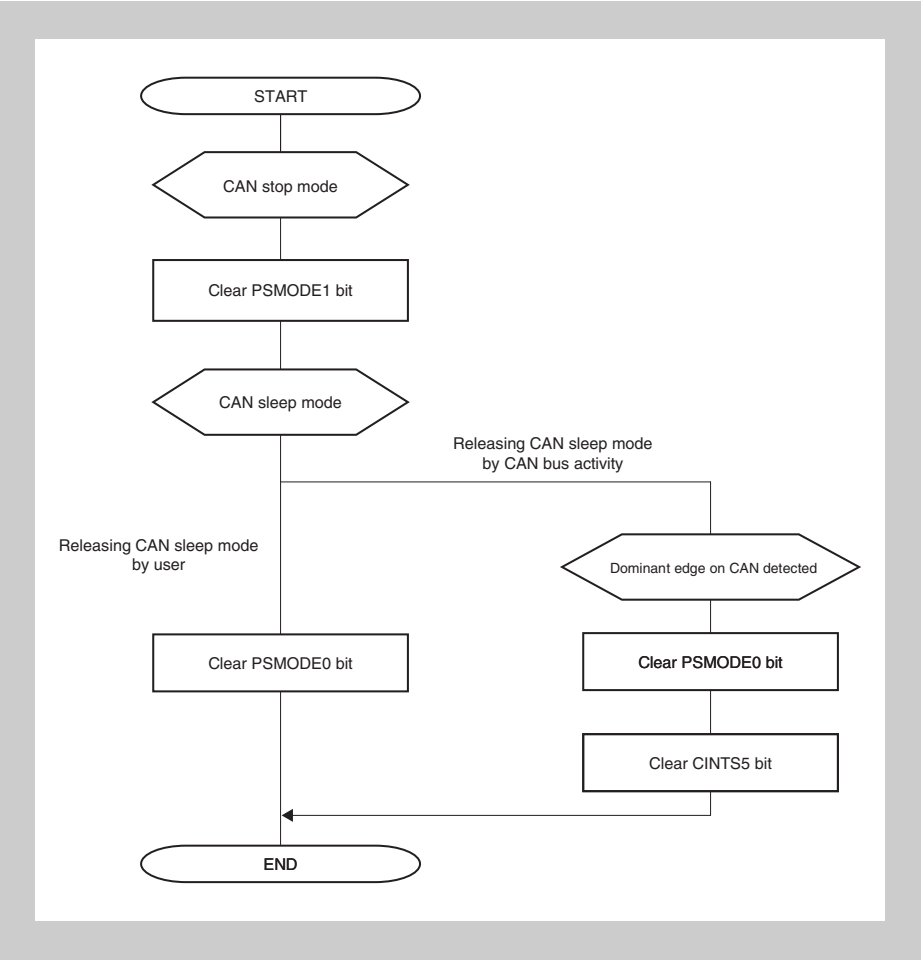
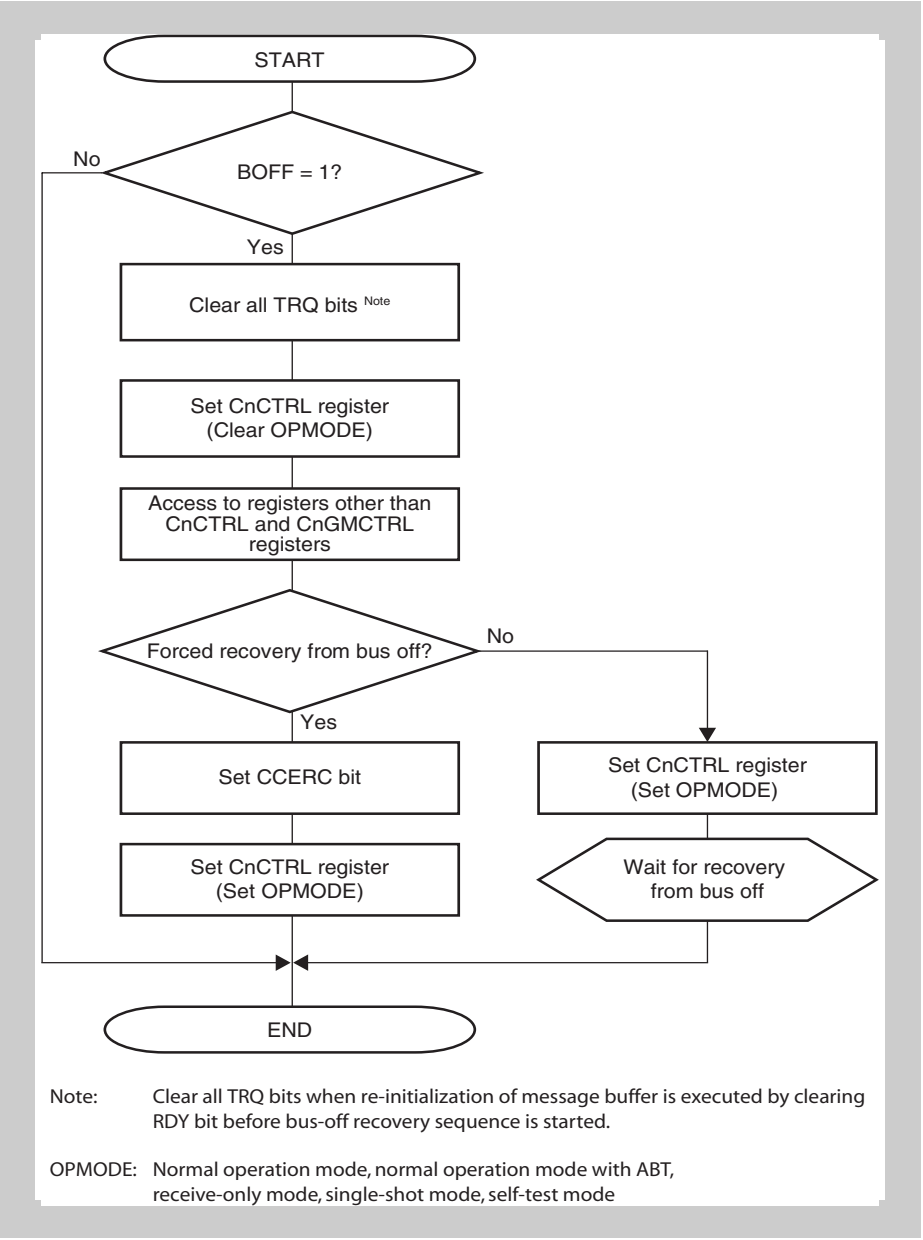
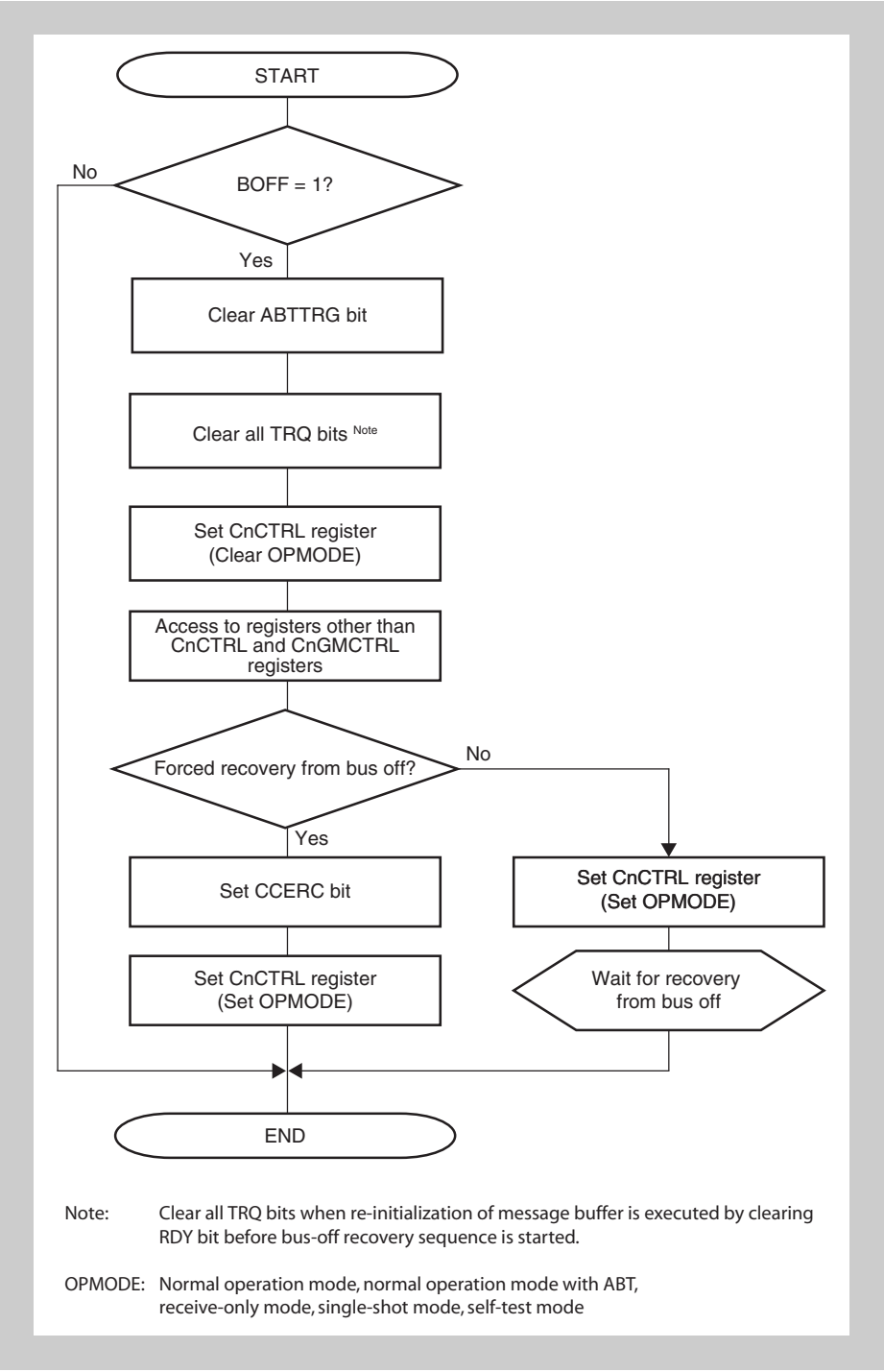


Figure 17-54 Bus-off recovery (except normal operation mode with ABT)



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.

Figure 17-55 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 17-56 Normal shutdown process

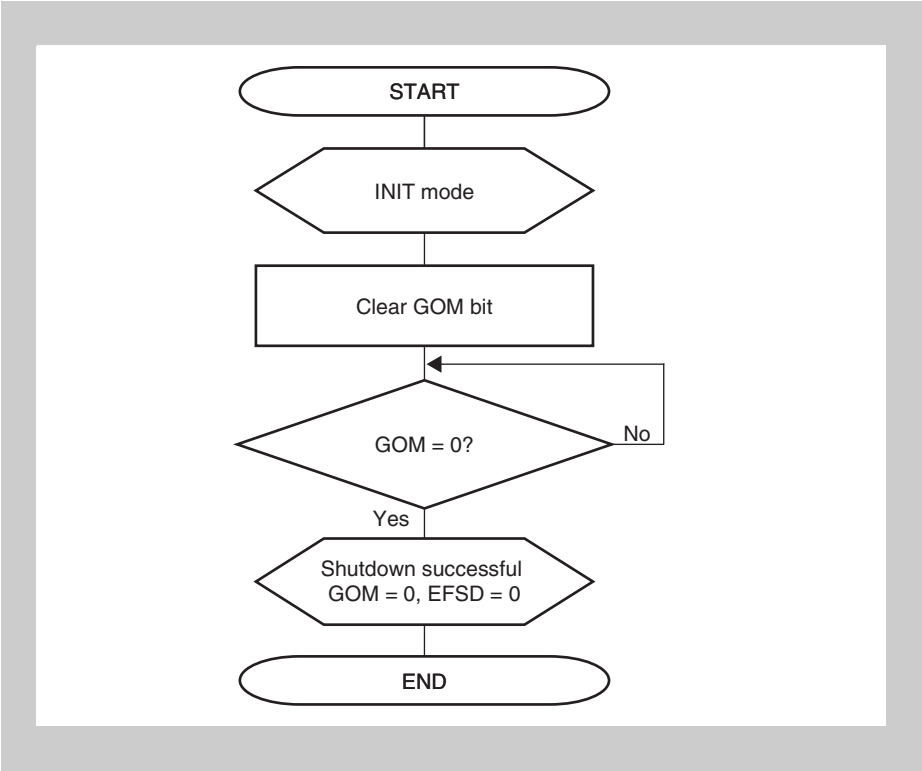
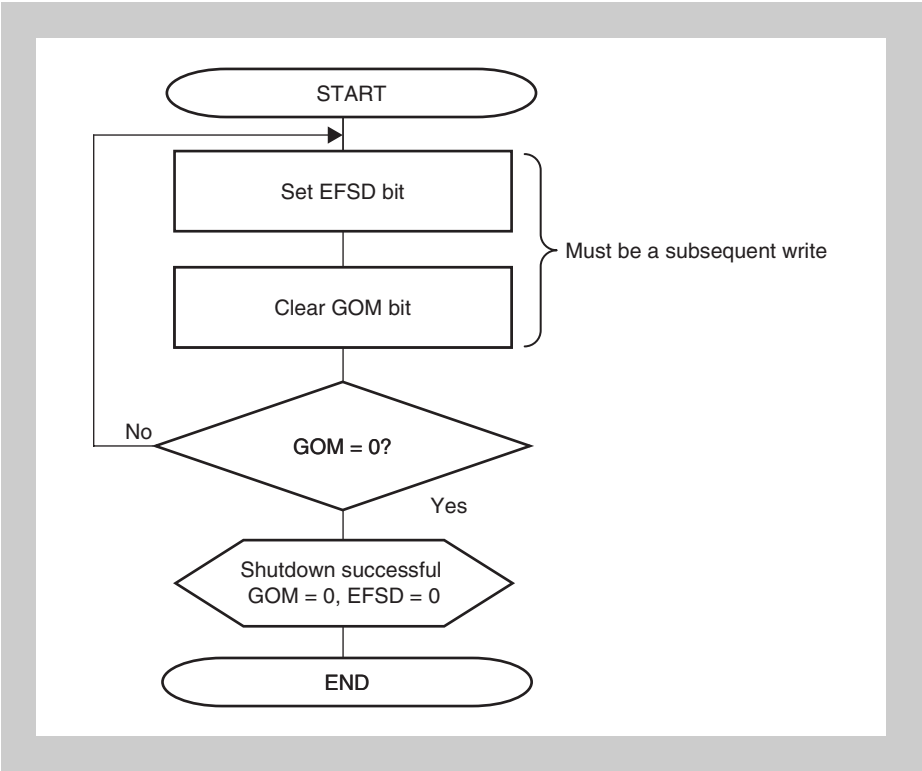


Figure 17-57 Forced shutdown process



Caution Do not read- or write-access any registers by software between setting the EFSD bit and clearing the GOM bit.

Figure 17-58 Error handling

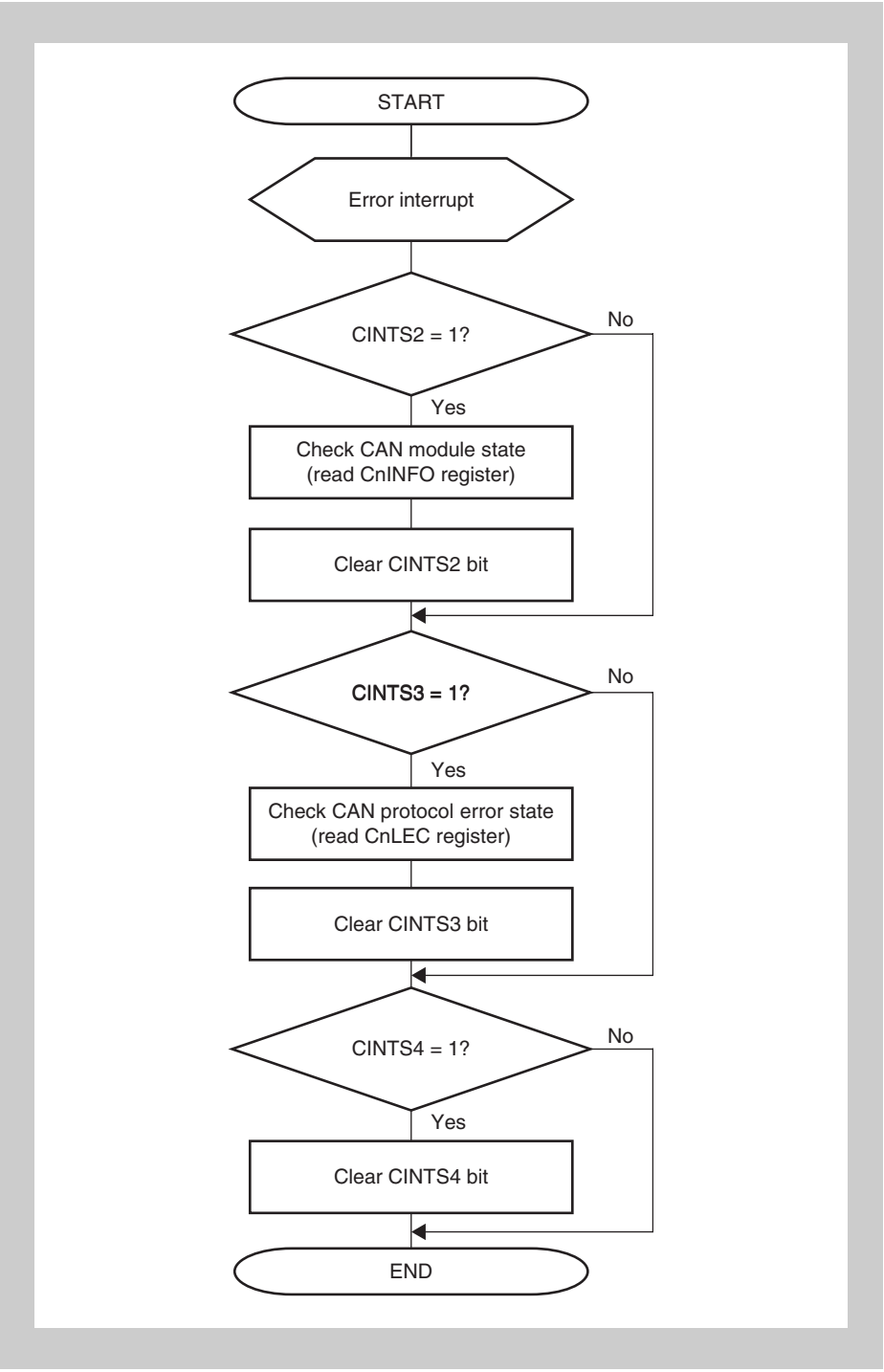
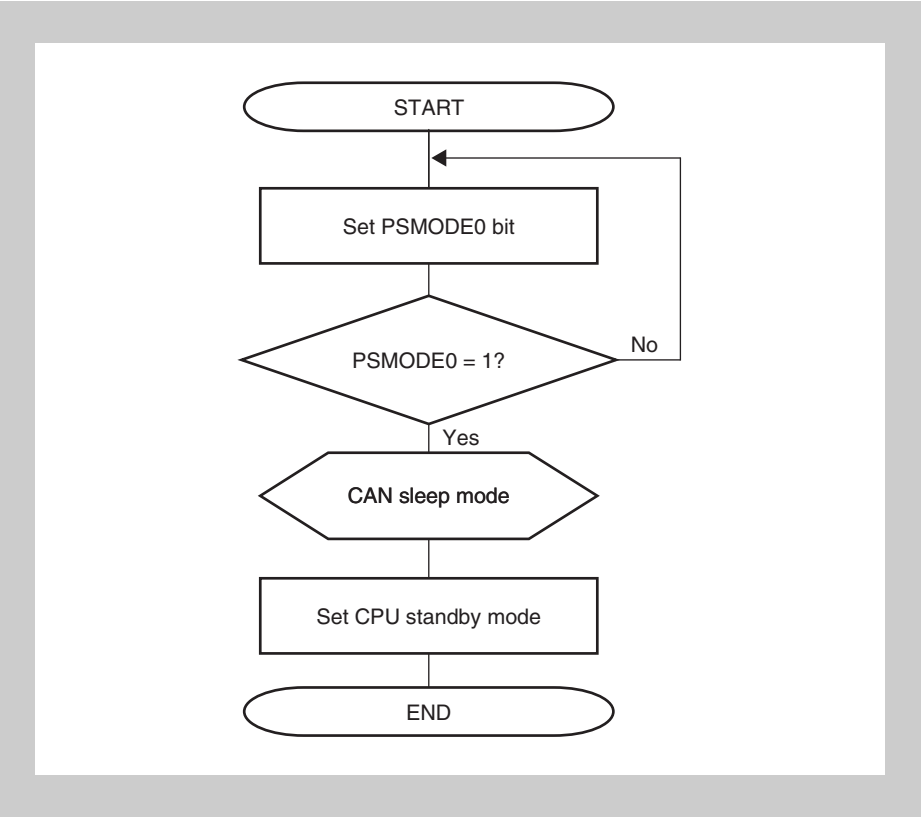
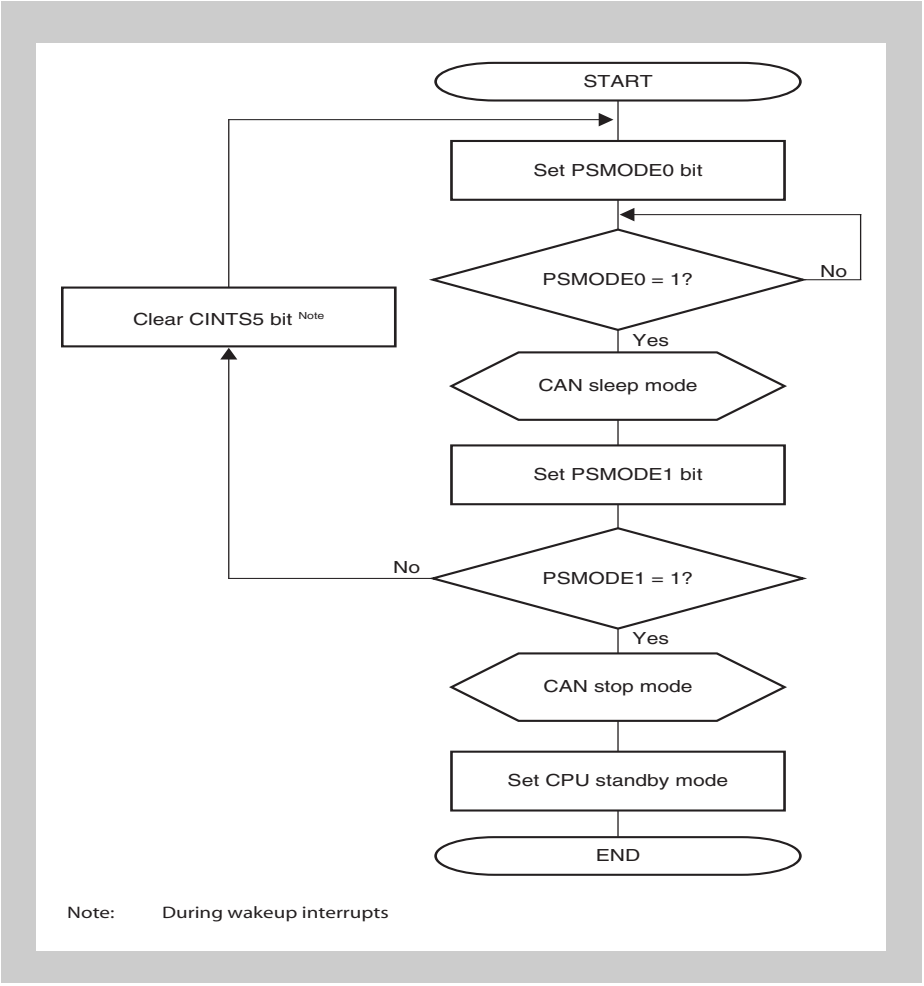


Figure 17-59 Setting CPU stand-by (from CAN sleep mode)



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 17-60 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 18 A/D Converter (ADC 3V)

Instances The 3V V850E/RG3 microcontrollers have one instance of the analog-digital converter ADC.

Table 18-1 Instances of ADC

ADCn	
Instances	1
Names	ADC0

Throughout this chapter, the individual instances of the A/D converter are identified by “n” (n = 0). Each A/D converter provides 16 channels, which are identified by “m” (m = 0 to 15).

18.1 Features

- Analog input: 16 channels (ANI00 to ANI15)
- 10-bit resolution
- Successive approximation method
- On-chip A/D conversion result register: ADCR0 to ADCR15, ADCRDDn0 to ADCRDDn2, ADCRSSn
- The following functions are provided as operation modes.
  - Single select mode with 1-buffer mode
  - Single select mode with 4-buffer mode
  - Single scan mode
  - Continuous select mode with 1-buffer mode
  - Continuous select mode with 4-buffer mode
  - Continuous scan mode
- The following functions are provided as trigger modes.
  - A/D trigger mode
  - Timer trigger mode
- The following extended functions are included.
  - Diagnostic mode
  - Discharge operation
  - DMA transfer support of A/D conversion result to internal RAM

18.2 Clock supply

Each ADC provides one clock input.

The clock connection is listed in following table.

Table 18-2 Clock input of ADC0

ADC0 clock input	Connected clock
VPCLK	32 MHz

Conversion time settings

The conversion time can be set via the ADMn1 register. Only the following settings are allowed.

Table 18-3 A/D Conversion time settings

Overall conversion time		Condition
Minimum [us]	Maximum [us]	
2.0	8.0	Discharge function disabled
5.625	22.5	
2.25	9.0	Discharge function enabled
5.875	23.5	

18.3 Input Selection Registers

To enhance the functionality of the ADC macro, it is externally connected to different trigger sources. The used source can be configured with the input selection register.

The possible connections and configurations are described in the following chapters.

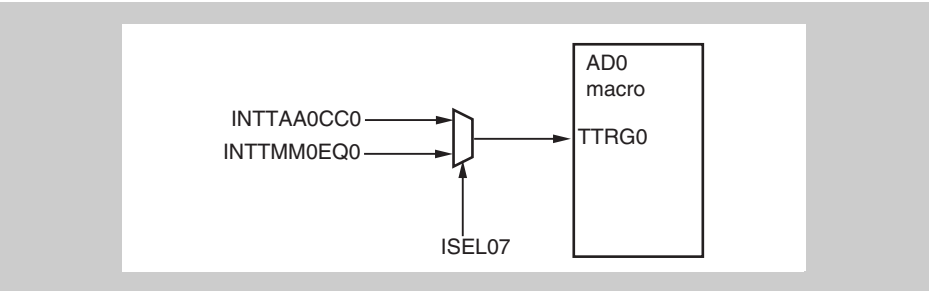


Figure 18-1 ADC0 - Timer trigger connection block diagram

18.3.1 SEL0CNT - Selector control register 1

The SEL0CNT register is an 8-bit register that controls the source selection for TAA input functions and the A/D Converter trigger input.
Bit seven of this register selects the ADC timer trigger input function.

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

Address 0xFFFF F6E0

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

7	6	5	4	3	2	1	0
ISEL07	ISEL06	ISEL05	ISEL04	ISEL03	ISEL02	ISEL01	ISEL00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

ISEL07	ADC timer trigger input TTRGADC
0	INTTAA0CC0, trigger from TAA0 capture /compare match 0
1	INTTMM0EQ0, from TMM0

18.4 Macro Overview

The A/D converter adopts the successive approximation method.

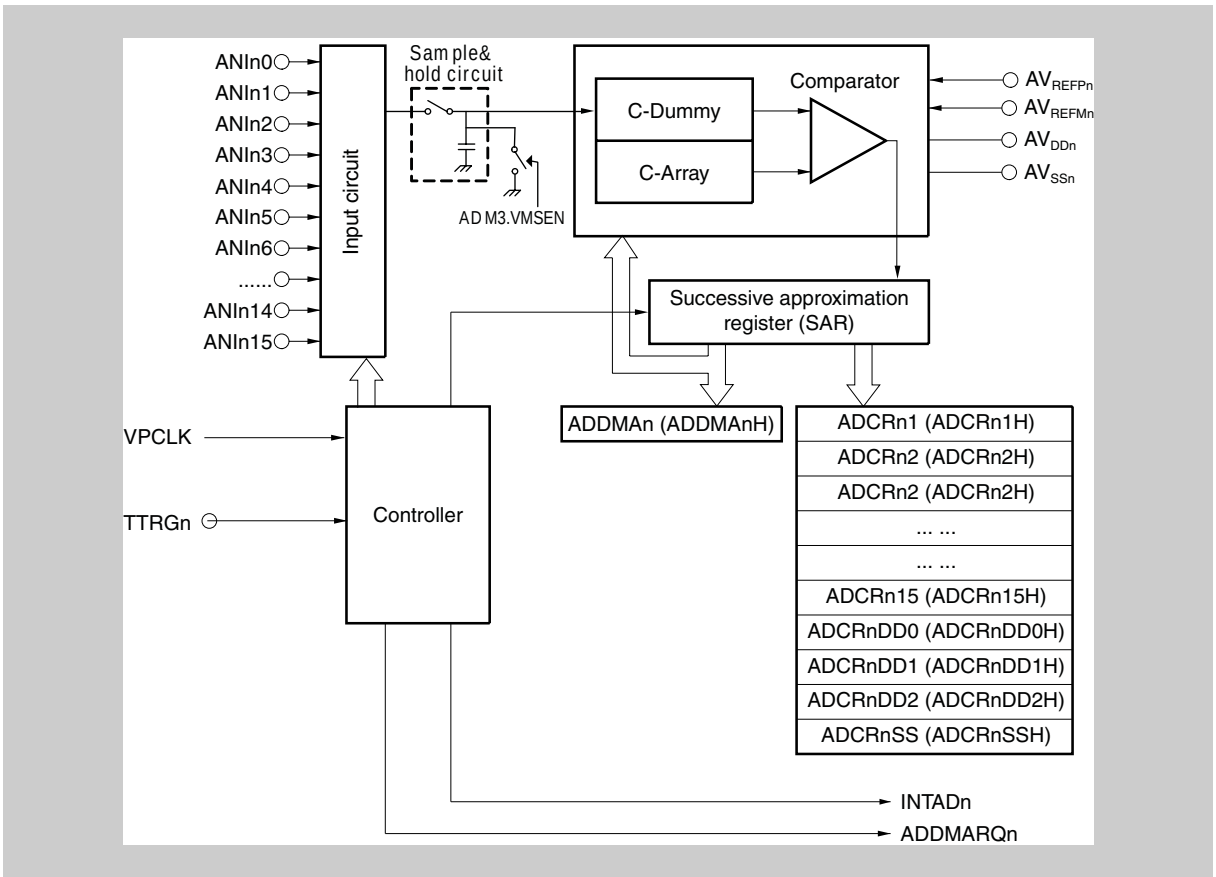


Figure 18-2 Block diagram of A/D converter (ADCn)

18.4.1 Input circuit

The input circuit selects the analog input (ANIn0 to ANIn15) according to the mode set by the ADMn0, ADMn1, ADMn2, and ADMn3 registers.

18.4.2 C-array

The C-array holds the charge of the differential voltage between the voltage input (from the analog input pins ANI0 to ANI15) and the reference voltage (AV_{DD}, $2/3AV_{DD}$, $1/2\ AV_{DD}$ and AV_{SS}), and redistributes the sampled charges.

18.4.3 C-dummy

This block holds the reference voltage (AV_{DD}, $2/3AV_{DD}$, $1/2\ AV_{DD}$ and AV_{SS}) and assigns the reference of the comparator input.

18.4.4 Sample & hold circuit

The sample and hold circuit sequentially samples each of the analog input signals sent from the input circuit and holds the sampled analog input signal for the voltage comparator during A/D conversion.

18.4.5 Voltage comparator

The voltage comparator compares the C-array comparison potential with the C-dummy reference potential.

18.4.6 Successive approximation register (SAR)

The voltage value of the analog input signal is compared with the compare voltage value from the C-dummy. The SAR register holds the result of this comparison starting from the most significant bit (MSB).

When the comparison result has reached the least significant bit (LSB) and A/D conversion has been completed, the contents of the SAR register are transferred to the A/D conversion result register (ADCRnm, ADCRDDnu [u=0 to 2] and ADCRSSn).

18.4.7 A/D conversion result register (ADCRnm, ADCRDDnu [u=0 to 2], ADCRSSn), A/D conversion result register H (ADCRnHm, ADCRDDnuH [u=0 to 2], ADCRSSnH)

ADCRnm, ADCRDDnu [u=0 to 2] and ADCRSSn are a 10-bit register that holds A/D conversion results. Each time A/D conversion is completed, the conversion results are loaded from the successive approximation register (SAR). RESET input makes this register undefined.

18.4.8 A/D conversion result register for DMA transfer (ADDMA0)

ADDMA0 is a 16-bit register that holds the last 10-bit A/D conversion result and an overrun flag for indicating a DMA transfer failure.

18.4.9 ANIn0 to ANIn15 pins

These are 16-channel analog input pins for the A/D converter. They input the analog signals to be converted.

Caution Make sure that the voltages input to ANIn0 to ANIn15 do not exceed the rated values. If a voltage higher than AV_{DD} or lower than AV_{SSn} (even within the range of the absolute maximum ratings) is input to a channel, the conversion value of the channel is undefined, and the conversion values of the other channels may also be affected.

18.4.10 AV_{REFP} pins

This is the pin for inputting the top reference voltage of the A/D converter. It converts signals input to the ANI0 to ANI15pins to digital signals based on the voltage applied between AVREFM and AVREFP.

18.4.11 AV_{REFM} pins

This is the pin for inputting the bottom reference voltage of the A/D converter. It converts signals input to the ANI0 to ANI15 pins to digital signals based on the voltage applied between AVREFM and AVREFP. Always use this pin at the same potential as the AVSS pin.

18.4.12 AV_{SS} pin

This is the ground pin of the A/D converter. Always use this pin at the same potential as the V_{SS} pin even when the A/D converter is not used.

18.4.13 AV_{DD} pin

This is the analog power supply pin of the A/D converter.

Cautions

1. If there is noise at the analog input pins (ANIn0 to ANIn15) or at the reference voltage input pin (AV_{REFP} or AV_{REFM}), the noise may generate an illegal conversion result.

   Software processing will be needed to avoid a negative effect on the system from this illegal conversion result. An example of this software processing is shown below.

   - Take the average result of a number of A/D conversions and use that as

the A/D conversion result.

- Execute a number of A/D conversions consecutively and use those results, omitting any exceptional results that may have been obtained.

2. Do not apply a voltage outside the AV_{REFM} to AV_{REFP} range to the pins that are used as A/D converter input pins.

18.5 Control Registers

Register addresses

The ADCn register addresses are given as address offsets to the individual base addresses <base> of each ADCn.

The <base> addresses of each ADCn are listed in the following table.

Table 18-4 Register <base> addresses of ADCn

ADCn	<base> address
ADC0	FFFF $F200_H$

The ADCn are controlled and operated by means of the following registers.

Table 18-5 ADCn registers overview (1/2)

Register name	Shortcut	Address
ADCn conversion result register 0	ADCRn0	<base>
ADCn conversion result register 1	ADCRn1	<base> + 2_H
ADCn conversion result register 2	ADCRn2	<base> + 4_H
ADCn conversion result register 3	ADCRn3	<base> + 6_H
ADCn conversion result register 4	ADCRn4	<base> + 8_H
ADCn conversion result register 5	ADCRn5	<base> + A_H
ADCn conversion result register 6	ADCRn6	<base> + C_H
ADCn conversion result register 7	ADCRn7	<base> + E_H
ADCn conversion result register 8	ADCRn8	<base> + 10_H
ADCn conversion result register 9	ADCRn9	<base> + 12_H
ADCn conversion result register 10	ADCRn10	<base> + 14_H
ADCn conversion result register 11	ADCRn11	<base> + 16_H
ADCn conversion result register 12	ADCRn12	<base> + 18_H
ADCn conversion result register 13	ADCRn13	<base> + $1A_H$
ADCn conversion result register 14	ADCRn14	<base> + $1C_H$
ADCn conversion result register 15	ADCRn15	<base> + $1E_H$
ADC result register for AVDD	ADCRDDn0	<base> + 20_H
ADC result register for 2/3AVDD	ADCRDDn1	<base> + 22_H
ADC result register for 1/2AVDD	ADCRDDn2	<base> + 24_H
ADC result register for AVSS	ADCRSSn	<base> + 26_H
ADC result register for DMA	ADDMA0	<base> + $2E_H$
ADCn mode register 0	ADMn0	<base> + 30_H

Table 18-5 ADCn registers overview (2/2)

Register name	Shortcut	Address
ADCn mode register 1	ADMn1	<base> + 31_H
ADCn mode register 2	ADMn2	<base> + 32_H
ADCn mode register 3	ADMn3	<base> + 33_H

18.5.1 ADMn0 - A/D Converter mode register 0

The ADMn0 register is an 8-bit register that specifies the operation mode, and executes conversion operations.

Access This register can be read/written in 8-bit or 1-bit units. However, bit 6 can only be read. Anything written to this bit is ignored.

Address <base> + 30_H

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

Cautions

1. In A/D trigger mode (ADMn1.TRG[1:0]=00), the conversion trigger is set by writing 1 to the ADCEn bit. When the ADCEn bit is 1 in the timer trigger mode (ADMn1.TRG[1:0]=01), the trigger signal standby state is set. To clear the ADCEn bit, write 0 or reset.
2. Changing the setting of the BSn bit is prohibited while A/D conversion is enabled (ADCEn bit = 1).
3. Changing the setting of the MSn bit is prohibited while A/D conversion is operating (ADCEn bit = 1 and ADCSn bit = 1).
4. When data is written to the ADMn0 register during an A/D conversion operation, the conversion operation is stopped and conversion is executed from the beginning.

7	6	5	4	3	2	1	0
ADCEn	ADCSn	BSn	MSn	0	0	0	0
R/W	R	R/W	R/W	R/W	R/W	R/W	R/W

ADCEn	A/D Conversion Operation Control of ADCn
0	Disables A/D conversion operation of ADCn
1	Enables A/D conversion operation ADCn

ADCSn	A/D Conversion Status Flag of ADCn
0	A/D conversion of ADCn is stopped
1	A/D conversion of ADCn is operating

BSn	ADCn Buffer Mode Specification
0	1-buffer mode
1	4-buffer mode

MSn	ADCn Operation Mode Specification
0	Scan mode
1	Select mode

18.5.2 ADMn1 - A/D Converter mode register 1

The ADMn1 register is an 8-bit register that specifies the conversion operation time and trigger mode.

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

Address <base> + 31_H

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

Cautions

1. Changing the setting of the TRGn1, TRGn0 and FRn3 to FRn0 bits is prohibited while A/D conversion is enabled (ADCEn bit of the ADMn0 register = 1).
2. When data is written to the ADMn1 register during an A/D conversion operation, the conversion operation is stopped.

7	6	5	4	3	2	1	0
0	0	TRGn1	TRGn0	FRn3	FRn2	FRn1	FRn0
R	R	R/W	R/W	R/W	R/W	R/W	R/W

TRGn1	TRGn0	ADCn Trigger Mode Specification
0	0	A/D trigger mode (conversion starts with ADMn0.ADCEn = 1)
0	1	Timer trigger mode
1	0	Setting prohibited
1	1	

Table 18-6 A/D conversion time without discharge mode (ADMn2.VMSEN=0)

FRn3	FRn2	FRn1	FRn0	SMP	ADHCLK	fxx=32MHz, Unit (μs)	
						Total Conversion Time[Note 1]	A/D Stabilization Time[Note 2]
0	0	0	0	0	fxx/4	2.000	1.0
				1		5.625	
0	0	0	1	0	fxx/8	4.000	1.0
				1		11.250	
0	0	1	0	0	fxx/12	6.000	1.0
				1		16.875	
0	0	1	1	0	fxx/16	8.000	1.0
				1		22.500	
Other than above					Prohibited		

Note 1. The conversion time must be set between 2.0 and 8.0us with SMP=0, 5.625 and 22.5us with SMP=1. Also it will take 2*ADHCLK (max) to start the conversion after the trigger is input. If the trigger comes during the conversion, it will take 3.25*ADHCLK (max) to restart the conversion.

2. After the ADCEn bit is set from 0 to 1 to secure the stabilization time of the A/D converter, conversion is started after the A/D stabilization time has elapsed only before the first A/D conversion is executed.

Table 18-7 A/D conversion time with discharge mode (ADMn2.VMSEN=1)

FRn3	FRn2	FRn1	FRn0	SMP	ADHCLK	fxx=32MHz, Unit (μs)	
						Total Conversion Time[Note 1]	A/D Stabilization Time[Note 2]
0	0	0	0	0	fxx/4	2.250	1.0
				1		5.875	
0	0	0	1	0	fxx/8	4.500	1.0
				1		11.750	
0	0	1	0	0	fxx/12	6.750	1.0
				1		17.625	
0	0	1	1	0	fxx/16	9.000	1.0
				1		23.500	
Other than above					Prohibited		

Note 1. The conversion time must be set between 2.25 and 9.0us with SMP=0, 5.875 and 23.5us with SMP=1. Also it will take 2*ADHCLK (max) to start the conversion after the trigger is input. If the trigger comes during the conversion, it will take 3.25*ADHCLK (max) to restart the conversion.

2. When the ADCE bit is set from 0 to 1 to secure the stabilization time of the A/D converter, conversion starts after the A/D stabilization time has elapsed but before the first conversion is executed.

18.5.3 ADMn2 - A/D Converter mode register 2

The ADMn2 register is an 8-bit register that specifies the analog input pin of the A/D Converter n.

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

Address <base> + 32_H

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

Cautions
1. If a channel for which no analog input pin exists is specified, the result of conversion is undefined.
2. Changing the setting of the ANISn3 to ANISn0 bits is prohibited while conversion is operating (ADCEn bit of the ADMn0 register = 1and ADCSn bit of the ADMn0 register = 1).
3. When data is written to the ADMn2 register during a conversion operation, the conversion operation is stopped.

7	6	5	4	3	2	1	0
0	0	0	0	ANISn3	ANISn2	ANISn1	ANISn0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 18-8 A/D Converter mode register 2 (ADMn2) without diagnostic mode

ANISn3	ANISn2	ANISn2	ANISn0	ADMn3.DIAGEN=0	
				Select Mode	Scan Mode
0	0	0	0	ANIn0	ANIn0
0	0	0	1	ANIn1	ANIn0, ANIn1
0	0	1	0	ANIn2	ANIn0 to ANIn2
0	0	1	1	ANIn3	ANIn0 to ANIn3
0	1	0	0	ANIn4	ANIn0 to ANIn4
0	1	0	1	ANIn5	ANIn0 to ANIn5
0	1	1	0	ANIn6	ANIn0 to ANIn6
0	1	1	1	ANIn7	ANIn0 to ANIn7
1	0	0	0	ANIn8	ANIn0 to ANIn8
1	0	0	1	ANIn9	ANIn0 to ANIn9
1	0	1	0	ANIn10	ANIn0 to ANIn10
1	0	1	1	ANIn11	ANIn0 to ANIn11
1	1	0	0	ANIn12	ANIn0 to ANIn12
1	1	0	1	ANIn13	ANIn0 to ANIn13
1	1	1	0	ANIn14	ANIn0 to ANIn14
1	1	1	1	ANIn15	ANIn0 to ANIn15

Table 18-9 A/D Converter mode register 2 (ADMn2) with diagnostic mode

ANISn3	ANISn2	ANISn2	ANISn0	ADMn3.DIAGEN=1	
				Select Mode	Scan Mode
0	0	0	0	AVDD	ANIn0->AVDD->2/3AVDD->1/2AVDD->AVSS
0	0	0	1	2/3AVDD	ANIn0->ANIn2->AVDD->2/3AVDD->1/2AVDD->AVSS
0	0	1	0	1/2AVDD	ANIn0->ANIn2->AVDD->2/3AVDD->1/2AVDD->AVSS
0	0	1	1	AVSS	ANIn0->ANIn3->AVDD->2/3AVDD->1/2AVDD->AVSS
0	1	0	0	setting prohibited	ANIn0->ANIn4->AVDD->2/3AVDD->1/2AVDD->AVSS
0	1	0	1	setting prohibited	ANIn0->ANIn5->AVDD->2/3AVDD->1/2AVDD->AVSS
0	1	1	0	setting prohibited	ANIn0->ANIn6->AVDD->2/3AVDD->1/2AVDD->AVSS
0	1	1	1	setting prohibited	ANIn0->ANIn7->AVDD->2/3AVDD->1/2AVDD->AVSS
1	0	0	0	setting prohibited	ANIn0->ANIn8->AVDD->2/3AVDD->1/2AVDD->AVSS
1	0	0	1	setting prohibited	ANIn0->ANIn9->AVDD->2/3AVDD->1/2AVDD->AVSS
1	0	1	0	setting prohibited	ANIn0->ANIn10->AVDD->2/3AVDD->1/2AVDD->AVSS
1	0	1	1	setting prohibited	ANIn0->ANIn11->AVDD->2/3AVDD->1/2AVDD->AVSS
1	1	0	0	setting prohibited	ANIn0->ANIn12->AVDD->2/3AVDD->1/2AVDD->AVSS

Table 18-9 A/D Converter mode register 2 (ADMn2) with diagnostic mode

ANISn3	ANISn2	ANISn2	ANISn0	ADMn3.DIAGEN=1	
				Select Mode	Scan Mode
1	1	0	1	setting prohibited	ANIn0->ANIn13->AVDD->2/3AVDD->1/2AVDD->AVSS
1	1	1	0	setting prohibited	ANIn0->ANIn14->AVDD->2/3AVDD->1/2AVDD->AVSS
1	1	1	1	setting prohibited	ANIn0->ANIn15->AVDD->2/3AVDD->1/2AVDD->AVSS

18.5.4 ADMn3 - A/D Converter mode register 3

The ADMn0 register is an 8-bit register that specifies the extended operation mode.

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

Address <base> + 33_H

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

Cautions

1. Changing the setting of the PLMn and VMSENn bits is prohibited while A/D conversion is enabled (ADCEn bit of the ADMn0 register = 1).
2. Changing the setting of the DIAGENn and SMPn bits is prohibited while A/D conversion is operating (ADCEn bit of the ADMn0 register = 1 and ADCSn bit of the ADMn0 register =1).
3. When data is written to the ADMn3 register during an A/D conversion operation, the conversion operation is stopped.
4. The conversion rate is longer when selecting the auto discharge function. *Chapter 18, Section Table 18-7, A/D conversion time with discharge mode (ADMn2.VMSEN=1)* shows the resulting times.

7	6	5	4	3	2	1	0
0	0	0	DIAGENn	PLMn	SMPn	0	VMSENn
R/W	R/W	R/W	R/W	R/W	R/W	0	R/W

DIAGENn	A/D Diagnostic mode selection bit of ADCn
0	Disables A/D conversion of AVDD, 2//3AVDD, 1/2AVDD and AVSS of ADCn
1	Enables A/D conversion of AVDD, 2//3AVDD, 1/2AVDD and AVSS of ADCn

PLMn	A/D Trigger Mode Selection bit of ADCn
0	Continuous mode
1	Single mode

SMPn	Sampling time selection bit
0	See *Chapter 18, Section Table 18-6, A/D conversion time without discharge mode (ADMn2.VMSEN=0)* and *Chapter 18, Section Table 18-7, A/D conversion time with discharge mode (ADMn2.VMSEN=1)*
1	

VMSENn	A/D Discharge function mode selection bit of ADCn
0	Standard mode
1	Discharge function mode enabled

18.5.5 ADCRn0 to ADCRn15, ADCRnH0 to ADCRnH15 - A/D conversion result registers

The ADCRnm register is a 10-bit register holding the A/D conversion results.

Access These registers are read-only in 16-bit or 8-bit units. When 16-bit access is performed, the ADCRnm register is specified, and when 8 bit access is performed, the ADCRnHm register holding the higher 8 bits of the conversion result is specified.

When reading the 10-bit data of the A/D conversion results from the ADCRnm register, only the higher 10 bits are valid and the lower 6 bits are always read as 0.

Address

ADCRn0 : <base>	ADCRnH0 : <base> + 1_H
ADCRn1 : <base> + 2_H	ADCRnH1 : <base> + 3_H
ADCRn2 : <base> + 4_H	ADCRnH2 : <base> + 5_H
ADCRn3 : <base> + 6_H	ADCRnH3 : <base> + 7_H
ADCRn4 : <base> + 8_H	ADCRnH4 : <base> + 9_H
ADCRn5 : <base> + A_H	ADCRnH5 : <base> + B_H
ADCRn6 : <base> + C_H	ADCRnH6 : <base> + D_H
ADCRn7 : <base> + E_H	ADCRnH7 : <base> + F_H
ADCRn8 : <base> + 10_H	ADCRnH8 : <base> + 11_H
ADCRn9 : <base> + 12_H	ADCRnH9 : <base> + 13_H
ADCRn10 : <base> + 14_H	ADCRnH10 : <base> + 15_H
ADCRn11 : <base> + 16_H	ADCRnH11 : <base> + 17_H
ADCRn12 : <base> + 18_H	ADCRnH12 : <base> + 19_H
ADCRn13 : <base> + $1A_H$	ADCRnH13 : <base> + $1B_H$
ADCRn14 : <base> + $1C_H$	ADCRnH14 : <base> + $1D_H$
ADCRn15 : <base> + $1E_H$	ADCRnH15 : <base> + $1F_H$

Initial Value Reset input causes an undefined register content.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADnm9	ADnm8	ADnm7	ADnm6	ADnm5	ADnm4	ADnm3	ADnm2	ADnm1	ADnm0	0	0	0	0	0	0
R	R	R	R	R	R	R	R	R	R	R/W	R/W	R/W	R/W	R/W	R/W

ADCRnHm

7	6	5	4	3	2	1	0
ADnm9	ADnm8	ADnm7	ADnm6	ADnm5	ADnm4	ADnm3	ADnm2
R	R	R	R	R	R	R	R

The correspondence between each analog input pin and the ADCRnm register is shown in Table 18-10 below.

Table 18-10 Assignment of A/D conversion result registers to analog input pins

Analog Input Pin	Assignment of A/D Conversion Result Registers	
	Select 1 Buffer Mode/ Scan Mode	Select 4 Buffer Mode
ANIn0	ADCRn0, ADCRnH0	ADCRn0 to ADCRn3, ADCRnH0 to ADCRnH3
ANIn1	ADCRn1, ADCRnH1	
ANIn2	ADCRn2, ADCRnH2	
ANIn3	ADCRn3, ADCRnH3	
ANIn4	ADCRn4, ADCRnH4	ADCRn4 to ADCRn7, ADCRnH4 to ADCRnH7
ANIn5	ADCRn5, ADCRnH5	
ANIn6	ADCRn6, ADCRnH6	
ANIn7	ADCRn7, ADCRnH7	
ANIn8	ADCRn8, ADCRnH8	ADCRn8 to ADCRn11, ADCRnH8 to ADCRnH11
ANIn9	ADCRn9, ADCRnH9	
ANIn10	ADCRn10, ADCRnH10	
ANIn11	ADCRn11, ADCRnH11	
ANIn12	ADCRn12, ADCRnH12	ADCRn12 to ADCRn15, ADCRnH12 to ADCRnH15
ANIn13	ADCRn13, ADCRnH13	
ANIn14	ADCRn14, ADCRnH14	
ANIn15	ADCRn15, ADCRnH15	
AVDDn	ADCRDDn0, ADCRDDn0H	ADCRDDn0, ADCRDDn1, ADCRDDn2, ADCRSSn, ADCRDDn0H ADCRDDn1H, ADCRDDn2H, ADCRSSnH,
2/3AVDDn	ADCRDDn1, ADCRDDn1H	
1/2AVDDn	ADCRDDn2, ADCRDDn2H	
AVSSn	ADCRSSn, ADCRSSnH	

The relationship between the analog voltage input to the analog input pins (ANIn0 to ANIn15) and the A/D conversion result (of the A/D conversion result register (ADCRnm)) is as follows:

$$ADCR = INT\left(\frac{V_{IN}}{AV_{REF}} \times 1024 + 0{,}5\right)$$

or,

$$(ADCR - 0{,}5) \times \frac{AV_{REF}}{1024} \leq V_{IN} < (ADCR + 0{,}5) \times \frac{AV_{REF}}{1024}$$

INT( ): Function that returns the integer value

VIN: Analog input voltage

AVREF: AVREF pin voltage

ADCR: Value of A/D conversion result register (ADCRnm)

Figure 18-3 shows the relationship between the analog input voltage and the A/D conversion results.

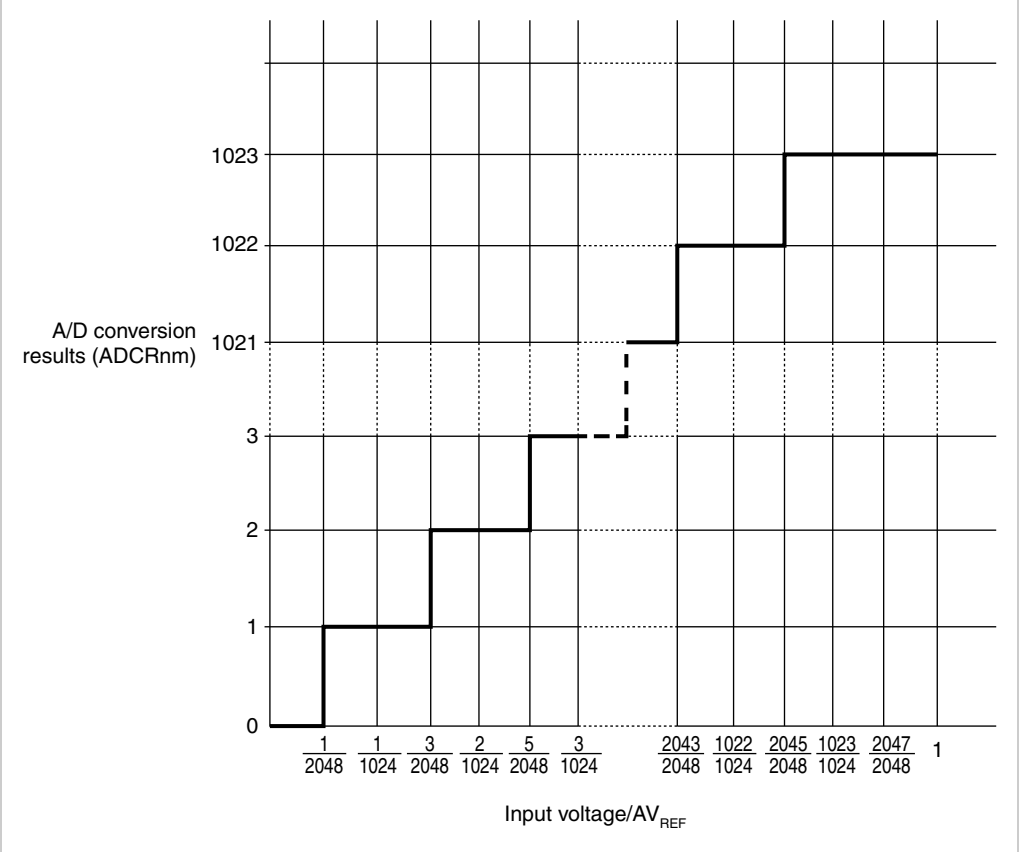


Figure 18-3 Relationship between analog input voltage and A/D conversion results

18.5.6 ADDMA0- A/D conversion result register for DMA

The ADDMA0 register is a 16-bit register holding the result of the latest A/D conversion operation, and is used for DMA transfer of ADC results into the internal RAM. It has an overrun detection flag indicating an overrun situation of the DMA transfer mechanism.

Access This register is read-only in 16-bit units or 8-bit units.

Address ADDMA0 : <base> + $2E_H$

Initial Value Reset input causes an undefined register content.

Cautions

1. This register is read-only in 16-bit units or 8-bit units. When 16-bit access is performed, the ADDMA0 register is specified, and when 8-bit access is performed, the ADDMAHn register holding the higher 8 bits of the conversion result is specified.
2. Do not read the ADDMA0 register by CPU during DMA transfer activities. If this register is read by CPU, overflow detection cannot be ensured.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AD9	AD8	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0	0	0	0	0	0	ODF
R	R	R	R	R	R	R	R	R	R	R/W	R/W	R/W	R/W	R/W	R

7	6	5	4	3	2	1	0
AD9	AD8	AD7	AD6	AD5	AD4	AD3	AD2
R	R	R	R	R	R	R	R

AD9 to AD0	A/D Conversion Result for DMA Transfer
000_H to $3FF_H$	Latest A/D conversion result value

ODFn	Overrun Detection Flag
0	No A/D conversion result overrun was detected.
1	At least one A/D conversion result was overrun since the last read of the ADDMA0 register.
• The ODFn flag is used for indicating a DMA transfer failure of the A/D conversion results. • The ODFn flag is cleared (0) when the conversion is stopped (ADCEn bit of the ADMn0 register is cleared to 0).	

18.6 Operation Overview

18.6.1 Basic operation

A/D conversion is executed by the following procedure.

1. The selection of the analog input and specification of the operation mode, trigger mode, etc. should be specified using the ADMn0, ADMn1, ADMn2 or ADMn3 registers[Note 1].
   When the ADCEn bit of the ADMn0 register is set to 1, A/D conversion starts in the A/D trigger mode. In the timer trigger mode the trigger standby state[Note 2] is set.
2. When conversion is started, the C-array voltage on the analog input side and the C-array voltage on the reference side are compared by the comparator.
3. When the comparison of the 10 bits ends, the conversion results are stored in the ADCRnm register. When conversion has been performed the specified number of times, the A/D conversion end interrupt (INTADn) is generated.

Note 1. If the setting of the ADMn0, ADMn1, ADMn2 or ADMn3 registers is changed during conversion, the operation immediately before is stopped, and the result of the conversion is not stored in the ADCRnm register. The conversion operation is then initialized, and conversion is executed from the beginning again.

2. If the ADCEn bit of the ADMn0 register is set to 1 during the timer trigger mode, the mode changes to the trigger standby state.
The conversion operation is started by the trigger signal (ADCSn bit in the ADMn0 register = 1), and the trigger standby state (ADCSn bit = 0) is returned when the conversion operation ends.

18.6.2 Operation mode and trigger mode

Various conversion operations can be specified for the A/D converter by specifying the operation mode and trigger mode. The operation mode and trigger mode are set by the ADMn0, ADMn1 and ADMn3 registers.

The following table shows the relationship between the operation mode and trigger mode.

Table 18-11 Relationship between operation mode and trigger mode

Trigger Mode		Operation Mode		Register Set Value		
				ADMn0	ADMn1	ADMn3
A/D trigger	Continuous	Select	1 buffer	$xx01\ 0000_B$	$xx00\ 0xxx_B$	$000x\ 0x0x_B$
			4 buffers	$xx11\ 0000_B$		
		Scan		$xx00\ 0000_B$		
	Single	Select	1 buffer	$xx01\ 0000_B$		$000x\ 1x0x_B$
			4 buffers	$xx11\ 0000_B$		
		Scan		$xx00\ 0000_B$		
Timer trigger		Select	1 buffer	$xx01\ 0000_B$	$xx010\ xxx_B$	$000x\ 0x0x_B$
			4 buffers	$xx11\ 0000_B$		
		Scan		$xx00\ 0000_B$		

(1) Trigger mode

There are two types of trigger modes that serve as the start timing of A/D conversion processing: A/D trigger mode and timer trigger mode. These trigger modes are set by the TRGn1 and TRGn0 bits of the ADMn1 register.

- **A/D trigger (continuous) mode**

  In the continuous mode, this mode starts the conversion timing of the analog input set to the ANI0 to ANI15 pins, and by setting the ADCE bit of the ADMn0 register to 1, starts the A/D conversion. Unless the ADCE bit is cleared to 0 after the conversion, the next conversion operation is repeated.

  If the data is written to the ADMn0 register during the conversion, the conversion is stopped and then executed from the beginning again. If the data is written to the ADMn1 to ADMn3 registers during the conversion, the conversion is stopped.

- **A/D trigger (single) mode**

  In the single mode, this mode starts the conversion timing of the analog input set to the ANIn0 to ANIn15 pins, and by setting

  the ADCE bit of the ADMn0 register to 1, starts the A/D conversion. To restart the A/D conversion after the INTAD interrupt (A/D conversion completed, ADCSn bit = 0), the ADCE bit has to be overwritten to 1 to start the next conversion.

  If the data is written to the ADMn0 register during the conversion, the conversion is stopped and then executed from the beginning again. If the

data is written to the ADMn1 to ADMn3 registers during the conversion, the conversion is stopped.

Note: The ADCE bit of the ADMn0 register stays at 1 in the single mode when the conversion is finished.

- **Timer trigger mode**

  This mode specifies the conversion timing of the analog input set for the ANIn0 to ANIn15 pins using signals from the TTRGn macro input.

  The ISEL07 bit selects which timer output is connected to the TTRGn macro input.

  If the ADCEn bit of the ADMn0 register is set to 1, the A/D Converter waits for an event input, and starts conversion when the event occurs (ADCSn bit of the ADMn0 register = 1). When conversion has finished, the converter waits for an event input again (ADCSn bit = 0).

  If the data is written to the ADMn0 register during the conversion, the conversion operation is stopped and the converter waits for a timer signal again. *If the data is written to the ADMn1 to ADMn3 registers during the conversion, the conversion operation is stopped.*

(2) Operation mode

There are two operation modes that set the ANIn0 to ANIn15 pins: select mode and scan mode. The select mode has sub-modes that consist of 1-buffer mode and 4-buffer mode. These modes are set by the BSn and MSn bits of the ADMn0 register.

- **Select mode**

  In this mode, one analog input specified by the ADMn2 register is A/D converted. The conversion results are stored in the ADCRnm register corresponding to the analog input (ANInm). For this mode, the 1-buffer mode and 4-buffer mode are provided for storing the A/D conversion results.

- **Scan mode**

  In this mode, the analog inputs specified by the ADMn2 register are selected sequentially from the ANIn0 pin, and A/D conversion is executed. The A/D conversion results are stored in the ADCRnm register corresponding to the analog input. When the conversion of the specified analog input ends, the A/D conversion end interrupt (INTADn) is generated.

  After conversion has finished, the next conversion operation is stopped in single mode of A/D trigger mode and repeated in both single mode except for A/D trigger mode and the continuous mode, unless the ADCE bit of the ADMn0 register is cleared to 0.

18.6.3 Standby

When the ADCEn bit of the ADMn0 register is cleared to 0, the DC current path between AVDD and AVSS and the DC current path between AVREFP and AVREFM are cut.

18.7 Operation in A/D Trigger Mode

When the ADCEn bit of the ADMn0 register is set to 1, A/D conversion is started.

18.7.1 Select mode operation

In this mode, the analog input specified by the ADMn2 register is converted. The conversion results are stored in the ADCRnm register corresponding to the analog input. In the select mode, the 1-buffer mode and 4-buffer mode are supported according to the storing method of the conversion results.

(1) 1-buffer mode (A/D trigger single select: 1 buffer)

In this mode, one analog input, AVDD (AVDD, 2/3AVDD, 1/2AVDD) or AVSS is converted once. The conversion results are stored in one ADCRnm, ADCRDDnu [u=0 to 2], or ADCRSSn register. The analog input, AVDD (AVDD, 2/3AVDD, 1/2AVDD) or AVSS and ADCRnm, ADCRDDnu [u=0 to 2] or ADCRSSn register correspond one to one.

Each time a conversion is executed, an A/D conversion end interrupt (INTAD) is generated and conversion ends.

This mode is most appropriate for applications in which the results of each first-time conversion are read.

(2) 1-buffer mode (A/D trigger continuous select: 1 buffer)

This mode is similar to the above, except that the conversion process does not stop after the first conversion. It will start again until the ADCE bit of the ADMn0 register is cleared to 0.

Each time an conversion is executed, an A/D conversion end interrupt (INTAD) is generated.

Table 18-12 Correspondence between analog input pins and ADCRnm register (A/D trigger select: 1 buffer)

Analog Input	A/D Conversion Result Register
ANInm	ADCRnm
AVDD Note	ADCRDDn0
2/3AVDD Note	ADCRDDn1
1/2AVDD Note	ADCRDDn2
AVSS Note	ADCRSSn

Note When the diagnostic mode is selected (ADMn3.DIAGEN=1).

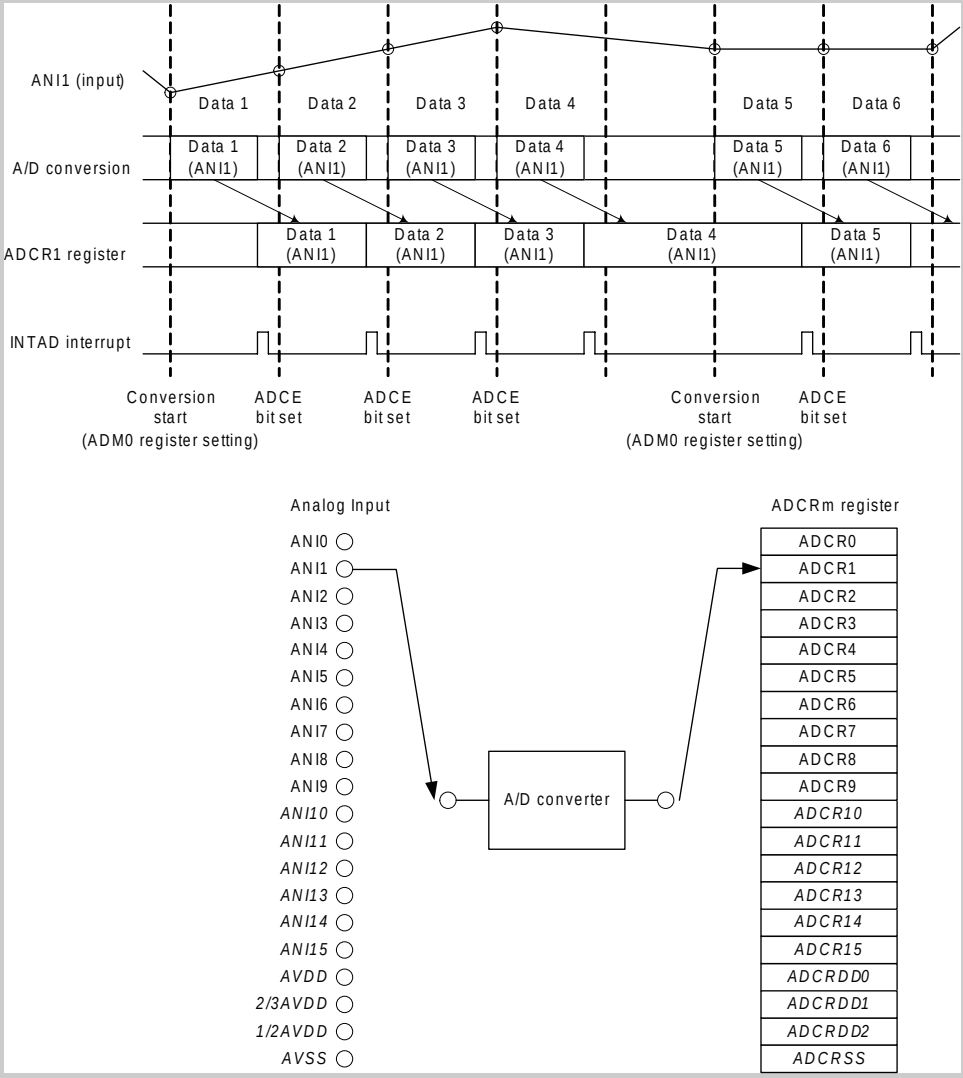


Figure 18-4 Single select mode operation timing: 1-buffer mode (ANIn1)

1. The ADCEn bit of ADMn0 register is set to 1 (enable)
4. ANIn1 is converted and the conversion result is stored in ADCRn1 register
5. The INTAD interrupt is generated
6. The ADCEn bit of ADMn0 register has to be overwritten to 1 to start the next conversion.

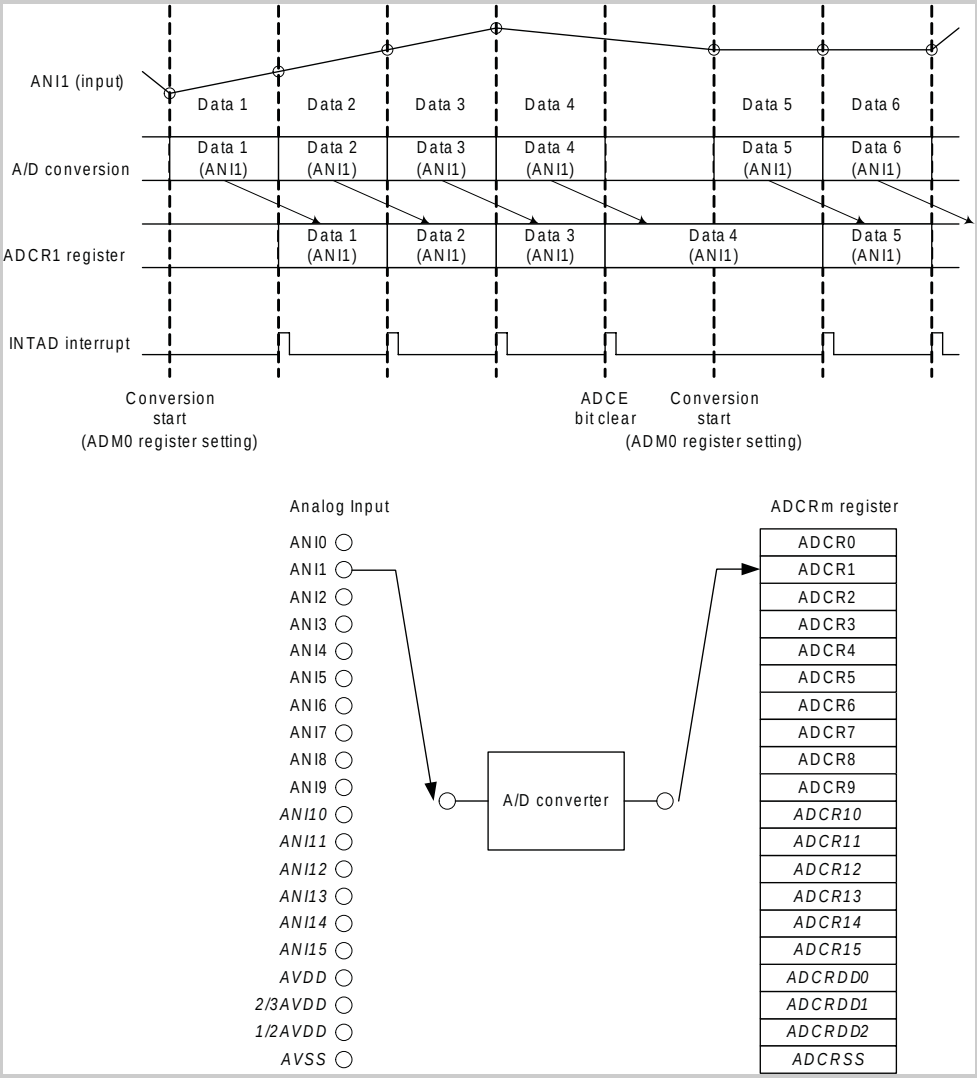


Figure 18-5 Continuous select mode operation timing: 1-buffer mode (ANIn1)

1. The ADCEn bit of ADMn0 register is set to 1 (enable)
7. ANIn1 is converted and the conversion result is stored in ADCRn1 register
8. The INTAD interrupt is generated
9. Unless the ADCEn bit of ADMn0 register is cleared to 0, the conversion operation is repeated.

(3) 4-buffer mode (A/D trigger single select: 4 buffers)

In this mode, one analog input, AVDD (AVDD, 2/3AVDD, 1/2AVDD) or AVSS is converted four times and the results are stored in the ADCRnm, ADCRDDnu [u=0 to 2] or ADCRSSn register. When the fourth conversion ends, an A/D conversion end interrupt (INTAD) is generated and the conversion is stopped.

(4) 4-buffer mode (A/D trigger continuous select: 4 buffers)

In this mode, one analog input, AVDD (AVDD, 2/3AVDD, 1/2AVDD) or AVSS is A/D converted four times and the results are stored in the ADCRnm, ADCRDDnu [u=0 to 2] or ADCRSSn register. When the fourth conversion ends, an A/D conversion end interrupt (INTAD) is generated and the

conversion is started again. It will start as long as the ADCEn bit of the ADMn0 register is not cleared to 0.

This mode is suitable for applications in which the average of the A/D conversion results is calculated.

Table 18-13 Correspondence between analog input pins and ADCRnm register (A/D trigger select: 4 buffers)

Analog Input	A/D Conversion Result Register
ANI0 to ANI3	ADCRn0 (1st time)
	ADCRn1 (2nd time)
	ADCRn2 (3rd time)
	ADCRn3 (4th time)
ANI4 to ANI7	ADCRn4 (1st time)
	ADCRn5 (2nd time)
	ADCRn6 (3rd time)
	ADCRn7 (4th time)
ANI8 to ANI11	ADCRn8 (1st time)
	ADCRn9 (2nd time)
	ADCRn10 (3rd time)
	ADCRn11 (4th time)
ANI12 to ANI15	ADCRn12 (1st time)
	ADCRn13 (2nd time)
	ADCRn14 (3rd time)
	ADCRn15 (4th time)
AVDD Note 2/3AVDD Note 1/3AVDD Note AVSS Note	ADCRDDn0 (1st time)
	ADCRDDn1 (2nd time)
	ADCRDDn2 (3rd time)
	ADCRSSn (4th time)

Note When the diagnostic mode is selected (ADMn3.DIAGEN=1).

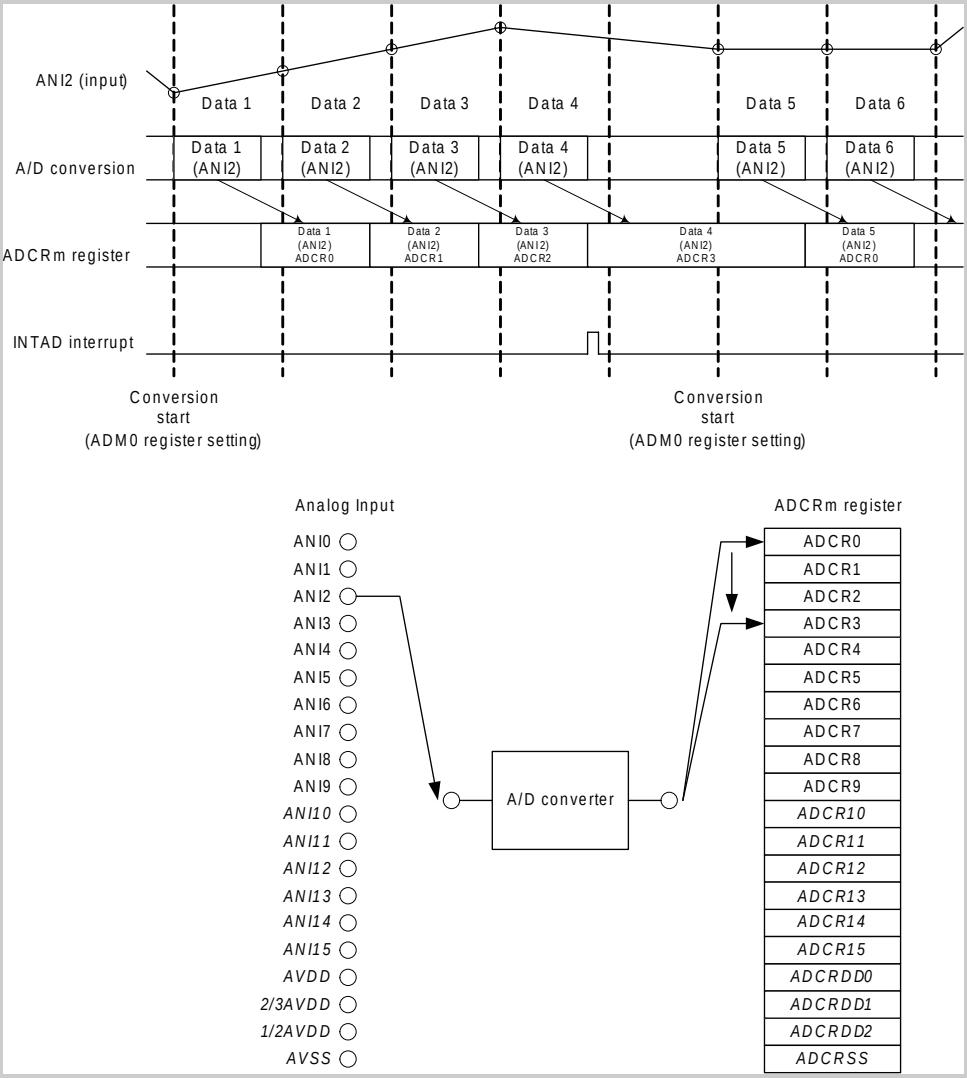


Figure 18-6 Example of 4-buffer mode operation (A/D trigger single select: 4 buffers)

1. The ADCEn bit of ADMn0 register is set to 1 (enable)
10. ANIn2 is converted and the conversion result is stored in ADCRn0 register
11. ANIn2 is converted and the conversion result is stored in ADCRn1 register
12. ANIn2 is converted and the conversion result is stored in ADCRn2 register
13. ANIn2 is converted and the conversion result is stored in ADCRn3 register
14. The INTAD interrupt is generated
15. The ADCEn bit of ADMn0 register has to be overwritten to 1 to start the next conversion.

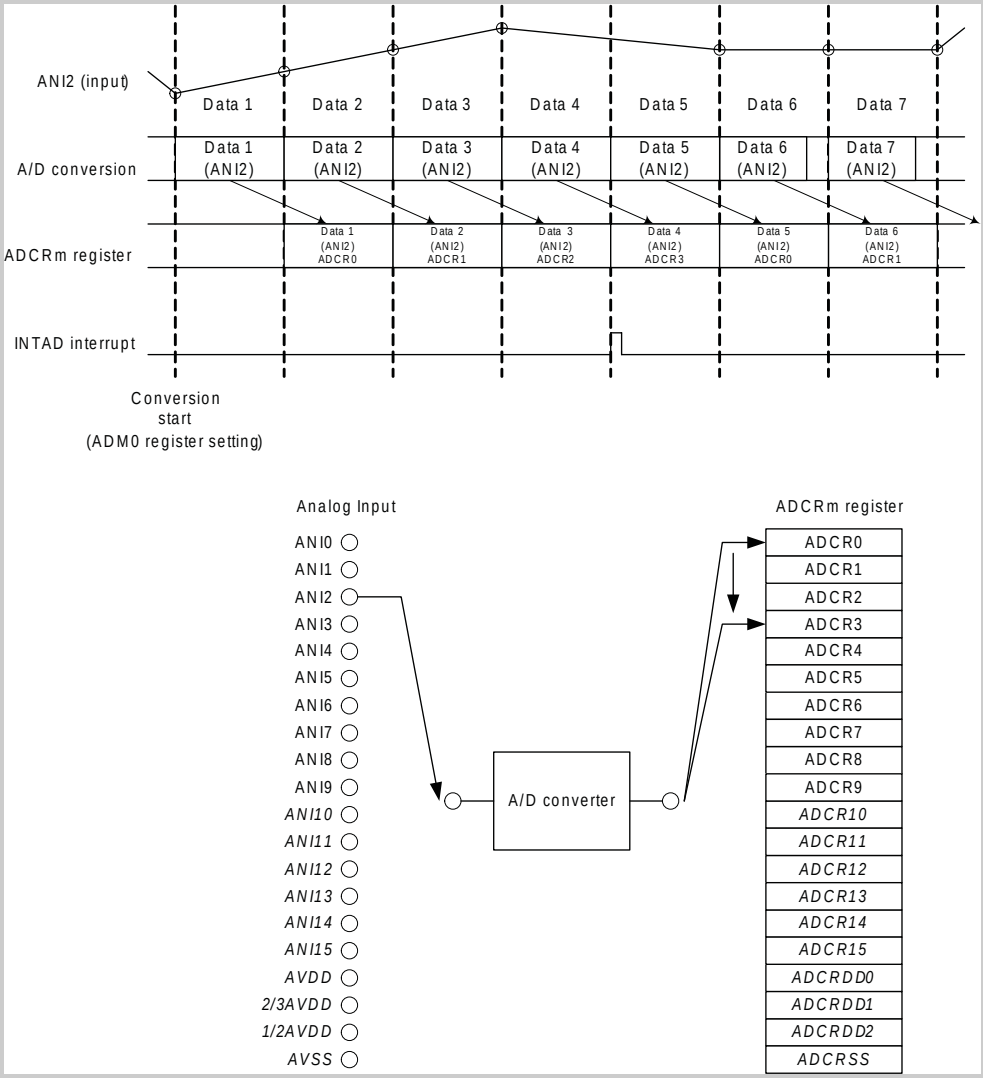


Figure 18-7 Example of 4-buffer mode operation (A/D trigger continuous select: 4 buffers)

1. The ADCE nbit of ADMn0 register is set to 1 (enable)

16. ANIn2 is converted and the conversion result is stored in ADCRn0 register
17. ANIn2 is converted and the conversion result is stored in ADCRn1 register
18. ANIn2 is converted and the conversion result is stored in ADCRn2 register
19. ANIn2 is converted and the conversion result is stored in ADCRn3 register
20. The INTAD interrupt is generated
21. Unless the ADCEn bit of ADMn0 register is cleared to 0, the conversion operation is repeated

18.7.2 Scan mode operation

In this mode, the analog inputs, AVDD (AVDD, 2/3AVDD, 1/2AVDD) or AVSS specified by the ADMn2 or ADMn3 register are selected sequentially from the ANIn0 pin, and A/D conversion is executed. The A/D conversion results are stored in the ADCRnm, ADCRDDnu, ADCRSSn register corresponding to the analog input.

(1) A/D trigger single scan

When conversion of all the specified analog input ends, the A/D conversion end interrupt (INTAD) is generated, and A/D conversion is stopped.

(2) A/D trigger continuous scan

When conversion of all the specified analog input ends, the A/D conversion end interrupt (INTAD) is generated and the A/D conversion is started again. It will start as long as the ADCEn bit of the ADMn0 register is not cleared to 0.

This mode is most appropriate for applications in which multiple analog inputs are constantly monitored.

Table 18-14 Correspondence between analog input pins and ADCRnm register (A/D trigger scan)

Analog Input	A/D Conversion Result Register
ANI0	ADCRn0
:	:
:	:
ANIm [m=0 to 15] **Note1**	ADCRn15
AVDD **Note 2**	ADCRDDn0
2/3AVDD **Note 2**	ADCRDDn1
1/2AVDD **Note 2**	ADCRDDn2
AVSS **Note 2**	ADCRSSn

Note 1. Set by the ANISn3 to ANISn0 bits of the ADMn2 register.

2. 2.When the diagnostic mode is selected (ADMn3.DIAGEN=1).

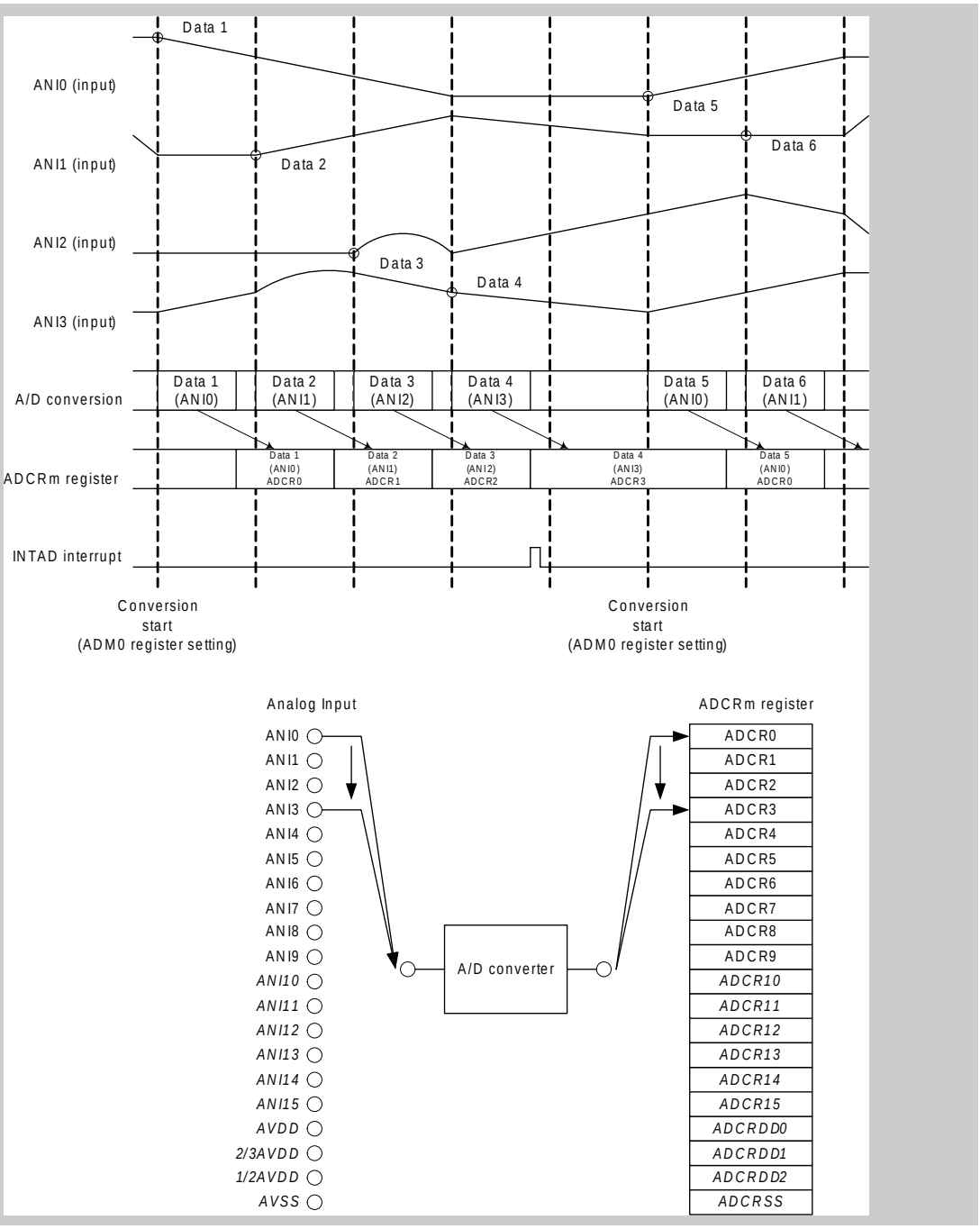


Figure 18-8 Example of single scan mode operation (A/D trigger single scan)

1. The ADCEn bit of ADMn0 register is set to 1 (enable)
22. ANnI0 is converted and the conversion result is stored in ADCRn0
23. ANIn1 is converted and the conversion result is stored in ADCRn1
24. ANIn2 is converted and the conversion result is stored in ADCRn2
25. ANIn3 is converted and the conversion result is stored in ADCRn3
26. The INTAD interrupt is generated
27. The ADCEn bit of ADMn0 register has to be overwritten to 1 to start the next conversion.

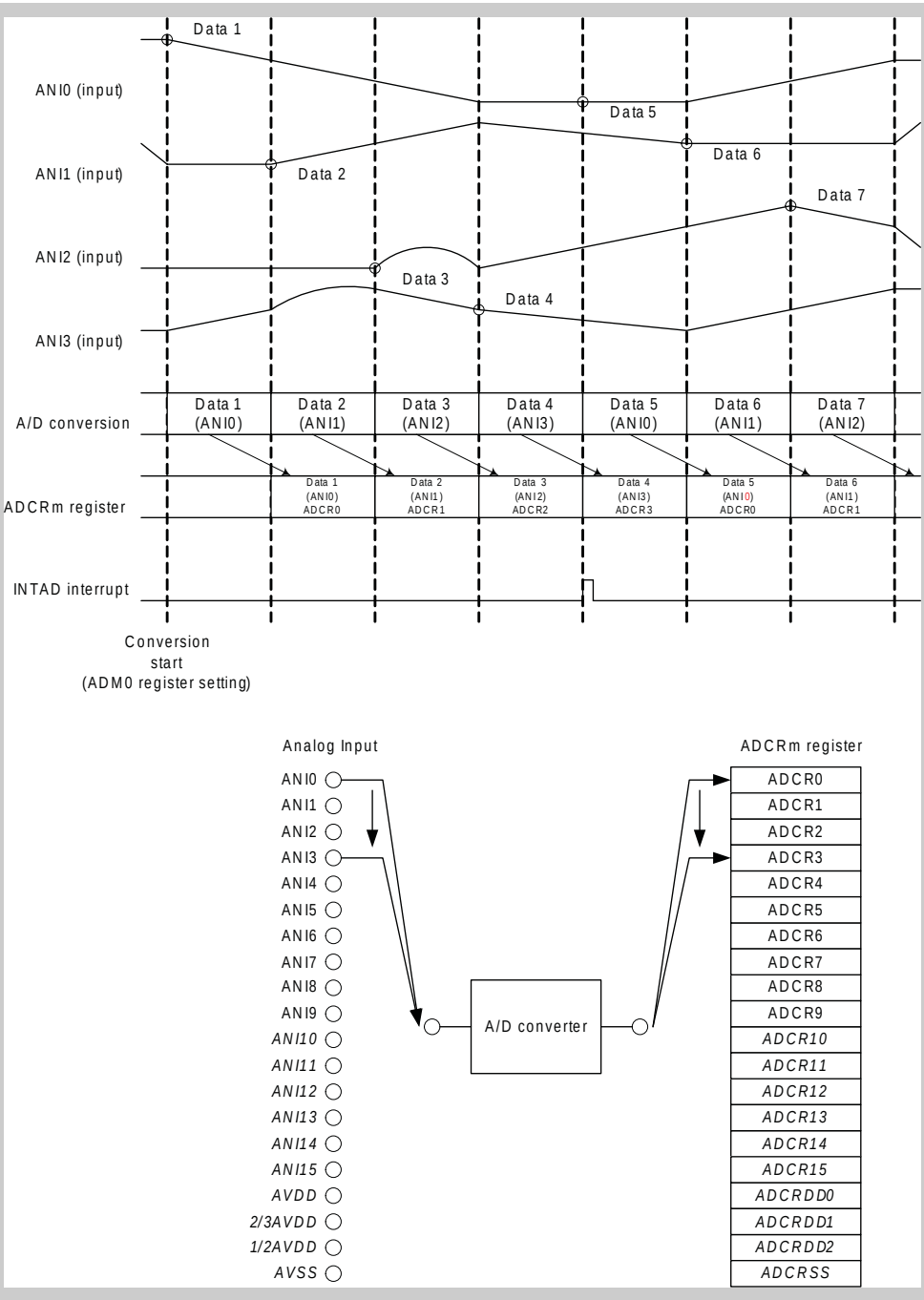


Figure 18-9 Example of continuous scan mode operation (A/D trigger continuous scan)

1. The ADCEn bit of ADMn0 register is set to 1 (enable)
28. ANIn0 is converted and the conversion result is stored in ADCRn0
29. ANIn1 is converted and the conversion result is stored in ADCRn1
30. ANIn2 is converted and the conversion result is stored in ADCRn2
31. ANIn3 is converted and the conversion result is stored in ADCRn3
32. The INTAD interrupt is generated
33. Unless the ADCEn bit of ADMn0 register is cleared to 0, the conversion operation is repeated

18.8 Operation in Timer Trigger Mode

In this mode, the conversion timing of the analog input signal set by the ANIn0 to ANIn15 pins is defined by a timer event signal.

The analog input conversion timing is generated when an A/D converter trigger signal from the timers is generated.

When the ADCEn bit of the ADMn0 register is set to 1, the A/D converter waits for the timer signal and starts conversion when the timer event occurs (ADCSn bit of the ADMn0 register = 1). When conversion is finished, the converter waits for a timer event signal again (ADCSn bit = 0).

If the timer event signal occurs during conversion, the conversion operation is executed from the beginning again.

If the data is written to the ADMn0 register during the conversion, the conversion operation is stopped and the converter waits for a timer signal again. If the data is written to the ADMn1 to ADMn3 registers during the conversion, the conversion operation is stopped.

18.8.1 Select mode operation

In this mode, an analog input (ANIn0 to ANIn15) specified by the ADMn2 register is converted. The conversion results are stored in the ADCRnm register corresponding to the analog input. In the select mode, the 1-buffer mode and 4-buffer mode are provided according to the storing method of the A/D conversion results.

(1) 1-buffer mode operation (timer trigger select: 1 buffer)

In this mode, one analog input, AVDD (AVDD, 2/3AVDD, 1/2AVDD) or AVSS is A/D converted once and the conversion results are stored in one ADCRnm, ADCRDDnu [u=0 to 2] or ADCRSSn register.

One analog input is converted one time, using the trigger of the timer event signals, and the results are stored in one ADCRnm register. A conversion end interrupt (INTAD) is generated for each conversion.

Unless the ADCEn bit of the ADMn0 register is cleared to 0, conversion is repeated each time a timer event signal is generated.

This mode is most appropriate for applications in which the results of each first-time conversion are read.

Table 18-15 Correspondence between analog input pins and ADCRnm register (1-buffer mode (timer trigger select: 1 buffer))

Trigger	Analog Input	A/D Conversion Result Register
Timer event signal	ANInm	ADCRnm
	AVDD Note	ADCRDDn0
	2/3AVDD Note	ADCRDDn1
	1/2AVDD Note	ADCRDDn2
	AVSS Note	ADCRSSn

Note When the diagnostic mode is selected (ADMn3.DIAGEN=1).

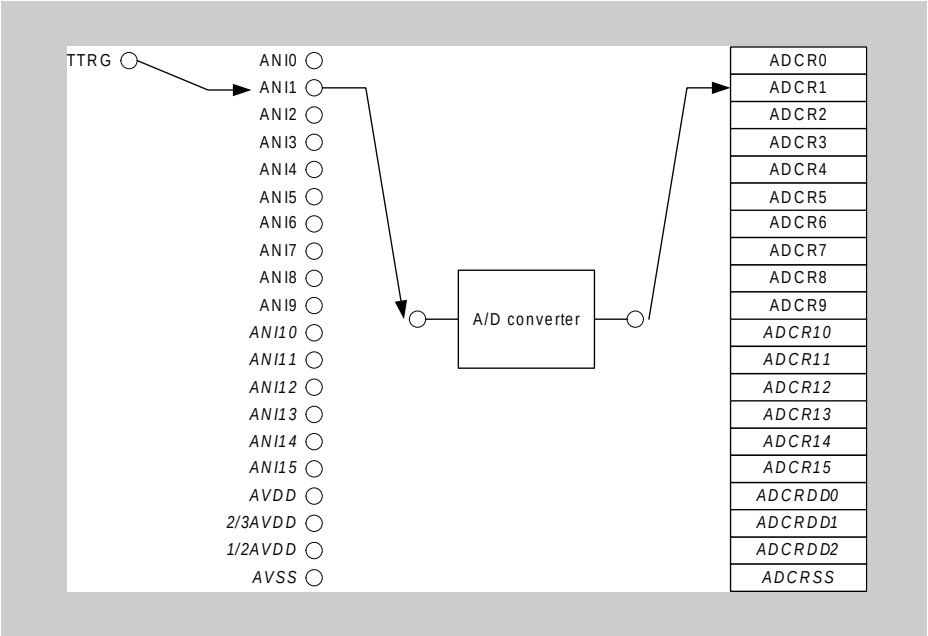


Figure 18-10 Example of 1-buffer mode operation (timer trigger select: 1 buffer) (ANIn1)

1. The ADCEn bit of ADMn0 register is set to 1 (enable)
34. The TS0ADT0 signal is generated
35. ANIn1 is converted and the conversion result is stored in ADCRn1
36. The INTADn interrupt is generated

(2) 4-buffer mode operation (timer trigger select: 4 buffers)

In this mode, A/D conversion of one analog input, AVDD (AVDD, 2/3AVDD, 1/2AVDD) or AVSS is executed four times, and the results are stored in the ADCRnm, ADCRDDnu [u=0 to 2] or ADCRSSn register.

One analog input is A/D converted four times using the timer event signals as a trigger, and the results are stored in four ADCRnm registers. The A/D conversion end interrupt (INTAD) is generated when the four A/D conversions end.

After conversion has finished, the next conversion is repeated when a timer event signal is generated, unless the ADCEn bit of the ADMn0 register is cleared to 0.

This mode is suitable for applications in which the average of the A/D conversion results is calculated.

Table 18-16 Correspondence between analog input pins and ADCRnm register (4-buffer mode (timer trigger select: 4 buffers))

Trigger	Analog Input	A/D Conversion Result Register
Timer event signal	ANI0 to ANI3	ADCRn0 (1st time)
		ADCRn1 (2nd time)
		ADCRn2 (3rd time)
		ADCRn3 (4th time)
	ANI4 to ANI7	ADCRn4 (1st time)
		ADCRn5 (2nd time)
		ADCRn6 (3rd time)
		ADCRn7 (4th time)
	ANI8 to ANI11	ADCRn8 (1st time)
		ADCRn9 (2nd time)
		ADCRn10 (3rd time)
		ADCRn11 (4th time)
	ANI12 to ANI15	ADCRn12 (1st time)
		ADCRn13 (2nd time)
		ADCRn14 (3rd time)
		ADCRn15 (4th time)
	AVDD Note 2/3AVDD Note 1/3AVDD Note AVSS Note	ADCRDDn0 (1st time)
		ADCRDDn1 (2nd time)
		ADCRDDn2 (3rd time)
		ADCRSSn (4th time)

Note When the diagnostic mode is selected (ADMn3.DIAGEN=1).

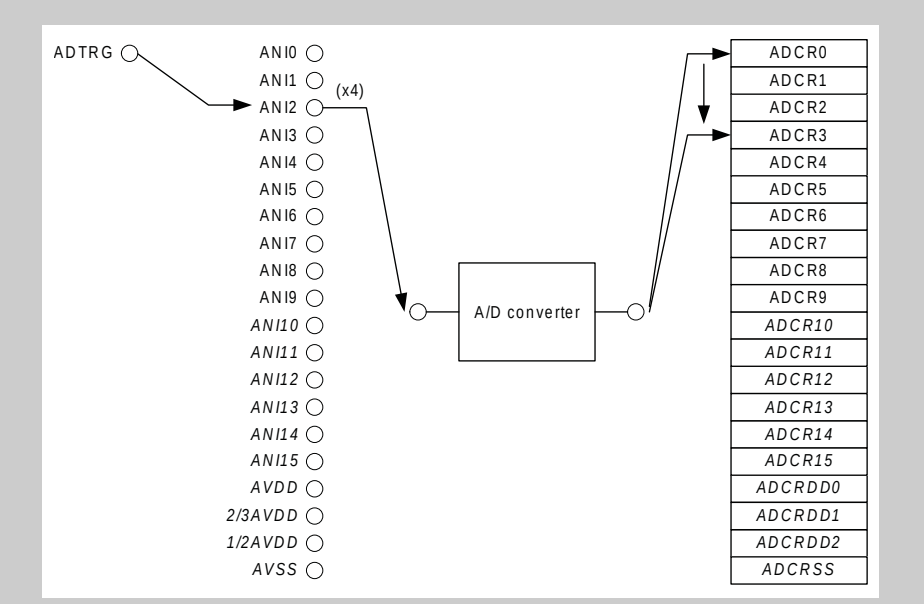


Figure 18-11 Example of 4-buffer mode operation (timer trigger select: 4 buffers) (ANIn3)

1. The ADCEn bit of ADMn0 register is set to 1 (enable)

37. TheTS0ADT0 signal is generated

38. ANIn3 is converted and the conversion result is stored in ADCRn0
39. TheTS0ADT0 signal is generated
40. ANIn3 is converted and the conversion result is stored in ADCRn1
41. TheTS0ADT0 signal is generated
42. ANIn3 is converted and the conversion result is stored in ADCRn2
43. TheTS0ADT0 signal is generated
44. ANIn3 is converted and the conversion result is stored in ADCRn3
45. The INTADn interrupt is generated

18.8.2 Scan mode operation

In this mode, the analog inputs specified by the ADMn2 register, AVDD (AVDD, 2/3AVDD, 1/2AVDD) or AVSS, are selected sequentially from the ANIn0 pin and are converted the specified number of times using the timer event signal as a trigger.

The result of conversion is stored in the ADCRnm, ADCRDDnu [u=0 to 2] or ADCRSSn register corresponding to the analog input. When all the specified analog input signals have been converted, an A/D conversion end interrupt (INTAD) occurs.

After conversion has finished, the A/D converter waits for a trigger unless the ADCEn bit of the ADMn0 register is cleared to 0. When a timer event occurs again, conversion starts again, from the ANIn0 input.

This mode is most appropriate for applications in which multiple analog inputs are constantly monitored.

Table 18-17 Correspondence between analog input pins and ADCRnm register (scan mode (timer trigger scan))

Trigger	Analog Input	A/D Conversion Result Register
Timer event signal	ANI0	ADCR0
	:	:
	:	:
	ANIm [m=0 to 15]	ADCRnm
	Note1	ADCRDDn0
	AVDD **Note 2**	ADCRDDn1
	2/3AVDD **Note 2**	ADCRDDn2
	1/2AVDD **Note 2**	ADCRSSn
	AVSS **Note 2**	

Note 1. Set by the ANISn3 to ANISn0 bits of the ADMn2 register.

2. When the diagnostic mode is selected (ADMn3.DIAGEN=1).

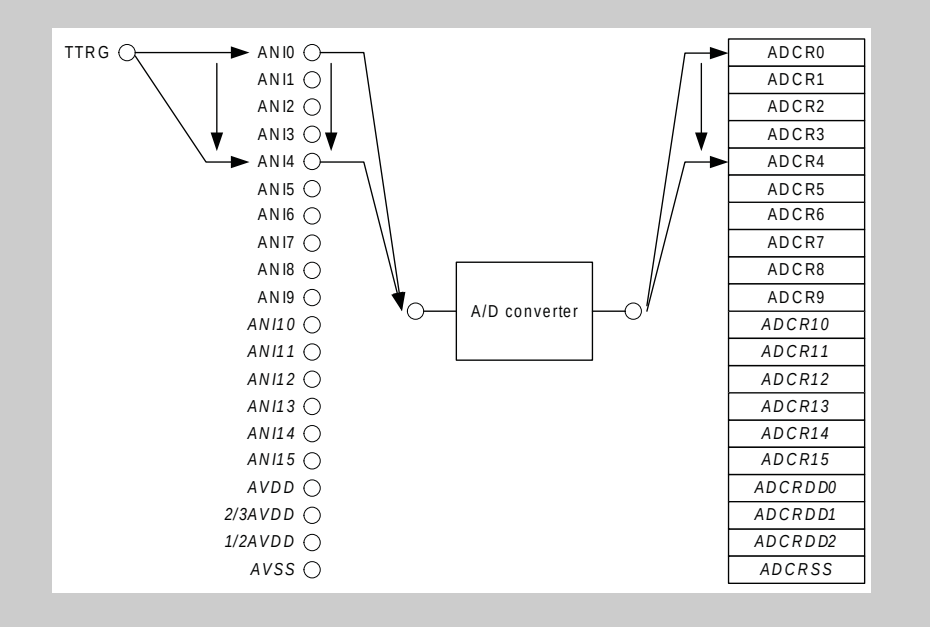


Figure 18-12 Example of scan mode operation (timer trigger scan) (ANIn0 to ANIn4)

1. The ADCEn bit of ADMn0 register is set to 1 (enable)
46. The TS0ADT0 signal is generated
47. ANIn0 is converted and the conversion result is stored in ADCRn0
48. ANIn1 is converted and the conversion result is stored in ADCRn1
49. ANIn2 is converted and the conversion result is stored in ADCRn2
50. ANIn3 is converted and the conversion result is stored in ADCRn3
51. ANIn4 is converted and the conversion result is stored in ADCRn4
52. The INTADn interrupt is generated

18.9 Extended Functions

18.9.1 Automatic Discharge Operation

The automatic discharge operation internally connects the sample and hold circuit to GND in order to open inputs and other failure modes. This function is disabled by default and is controlled by the VMSENn bit of the ADMn3 register.

18.9.2 Diagnostic Mode

The diagnostic mode configures the A/D converter to perform a conversion operation on the AVDD (AVDD, 2/3AVDD, 1/2AVDD) and AVSS terminals. Using this mode allows the user to implement a software algorithm in the application program to calibrate conversion results and thereby reduce the effect of certain error factors such as full scale error and zero offset error. This mode is disabled by default, and is controlled by the DIAGENn bit of the ADMn3 register.

(1) Operation in Select Mode

In select mode, AVDD (AVDD, 2/3AVDD, 1/2AVDD) or AVSS terminals can be selected using the ANISn3 to ANISn0 bits of register ADMn2. The conversion result is stored in the appropriate conversion result register and interrupt (INTAD0) is output.

(2) Operation in Scan Mode

In scan mode, AVDD (AVDD, 2/3AVDD, 1/2AVDD) and AVSS terminals are scanned immediately after conversion of the Scan mode conversion end channel specified in register ADMn2. The conversion result is stored in the appropriate conversion result register and interrupt (INTAD0) is output.

18.10 Precautions

18.10.1 Stopping conversion operation

When the ADCEn bit of the ADMn0 register is cleared to 0 during a conversion operation, the conversion operation stops and the conversion results are not stored in the ADCRnm, ADCRDDnu [u=0 to 2] or ADCRSSn register.

18.10.2 Writing the ADMn0 register during the conversion operation

If the data is written to the ADMn0 register during the conversion operation, the conversion operation is immediately stopped and the conversion results are not stored in the ADCRnm, ADCRDDnu [u=0 to 2] or ADCRSSn register. The conversion operation is initialized and the conversion operation is executed from the beginning again. Modifying the value of some bits is forbidden during the conversion. Please refer to the register description in section 18.5.1“ADMn0 - A/D Converter mode register 0” on page 535.

18.10.3 Writing the ADMn1, ADMn2 or ADMn3 registers during the conversion operation

If the data is written to the ADMn0, ADMn1 or ADMn2 register during the conversion operation, the conversion operation is immediately stopped and the conversion results are not stored in the ADCRnm, ADCRDDnu [u=0 to 2] or ADCRSSn register. Modifying the value of some bits is forbidden during the conversion. Please refer to the register descriptions in section 18.5“Control Registers” on page 534.

18.10.4 Timer trigger interval

Set the input time interval of the trigger in the timer trigger mode longer than the conversion time specified by the FRn3 to FRn0 bits of the ADMn1 register.

18.10.5 When 0 < interval ≤ conversion operation time

When the following timer trigger is input during a conversion operation, the conversion operation is aborted and the conversion starts according to the last timer trigger input.

When conversion operations are aborted, the conversion results are not stored in the ADCRnm, ADCRDDnu [u=0 to 2] or ADCRSSn register. However, the number of times the trigger has been input is counted. When an interrupt occurs, the values that have been converted are stored in the ADCRnm, ADCRDDnu [u=0 to 2] or ADCRSSn register.

18.10.6 Input range of ANIn0 to ANIn15

Use the input voltage at ANIn0 to ANIn15 within the specified range. If a voltage outside the range of AVREFM is input to any of these pins (even within the absolute maximum rating range), the converted value of the channel is undefined. In addition, the converted value of the other channels may also be affected.

18.10.7 A/D stabilization time

When the ADCEn bit of the ADMn0 register is set from 0 to 1, the conversion operation is started after the A/D stabilization time has elapsed.

18.10.8 Starting the conversion operation after the ADCEn bit of the ADMn0 register is set to 1

It can take up to 2*ADHCLK (max.) to start the first channel conversion each time the trigger is received or the ADCEn bit is set to 1. This has the following consequences:

- In single select 1-buffer mode, each conversion can take up to 2*ADHCLK (max.) to start.
- In any other mode, receiving the trigger or setting ADCEn to 1 will start more than one conversion and the extra 2*ADHCLK (max.) only apply for the first of these conversions.

18.10.9 Changing the configuration while the conversion operation is in idle state

The MSn bit of the ADMn0 register, the ANISn3-0 bits of the ADMn2 register, and the DIAGn and SMPn bits of the ADMn3 register can be changed while the conversion is in idle state (ADMn0.ADCE =1 and ADMn0.ADCS=0). Therefore these bits can be changed without clearing the ADCEn bit of the ADMn0 register to 0 in the single select mode.

18.10.10 Conflicts

- **Conflict between writing A/D conversion result register (ADCRnm, ADCRDDnu [u=0 to 2] or ADCRSSn) at end of conversion and reading ADCRnm, ADCRDDnu [u=0 to 2] or ADCRSSn register by instruction**

  Reading the ADCRnm, ADCRDDnu [u=0 to 2] or ADCRSSn register takes precedence. After the register has been read, the new conversion results are written to the ADCRnm, ADCRDDnu [u=0 to 2] or ADCRSSn register.

Chapter 19 A/D Converter (ADC 5V)

Instances The V850E/RG3 microcontrollers has 1 instance of the analog digital converter ADC.

Table 19-1 Instances of ADC

ADCn	
Instances	1
Names	ADC0

Throughout this chapter, the individual instances of the A/D Converter are identified by “n” (n = 0). The A/D Converter provides 16 channels, which are identified by “m” (m = 0 to 15).

19.1 Features

- Analog input: 16 channels (ANIn0 to ANIn15)
- 10-bit resolution
- On-chip A/D conversion result register (ADCRn0 to ADCRn15, ADCRDDn0 to ADCRDDn2, ADCRSSn)
- Operating voltage/upper reference voltage:

  AVDD = AVREFP = 4.5 to 5.5V
- ADC ground/lower reference voltage:

  AVSS = AVREFM = 0V
- Operation modes:
  - Single select mode with 1-buffer mode
  - Single select mode with 4-buffer mode
  - Single scan mode
  - Continuous select mode with 1-buffer mode
  - Continuous select mode with 4-buffer mode
  - Continuous scan mode
- Trigger modes:
  - A/D trigger mode
  - Timer trigger mode
- Successive approximation method
- Extended functions
  - Diagnostic mode
  - Discharge operation
  - DMA transfer support of A/D conversion result to internal RAM

19.2 Configuration

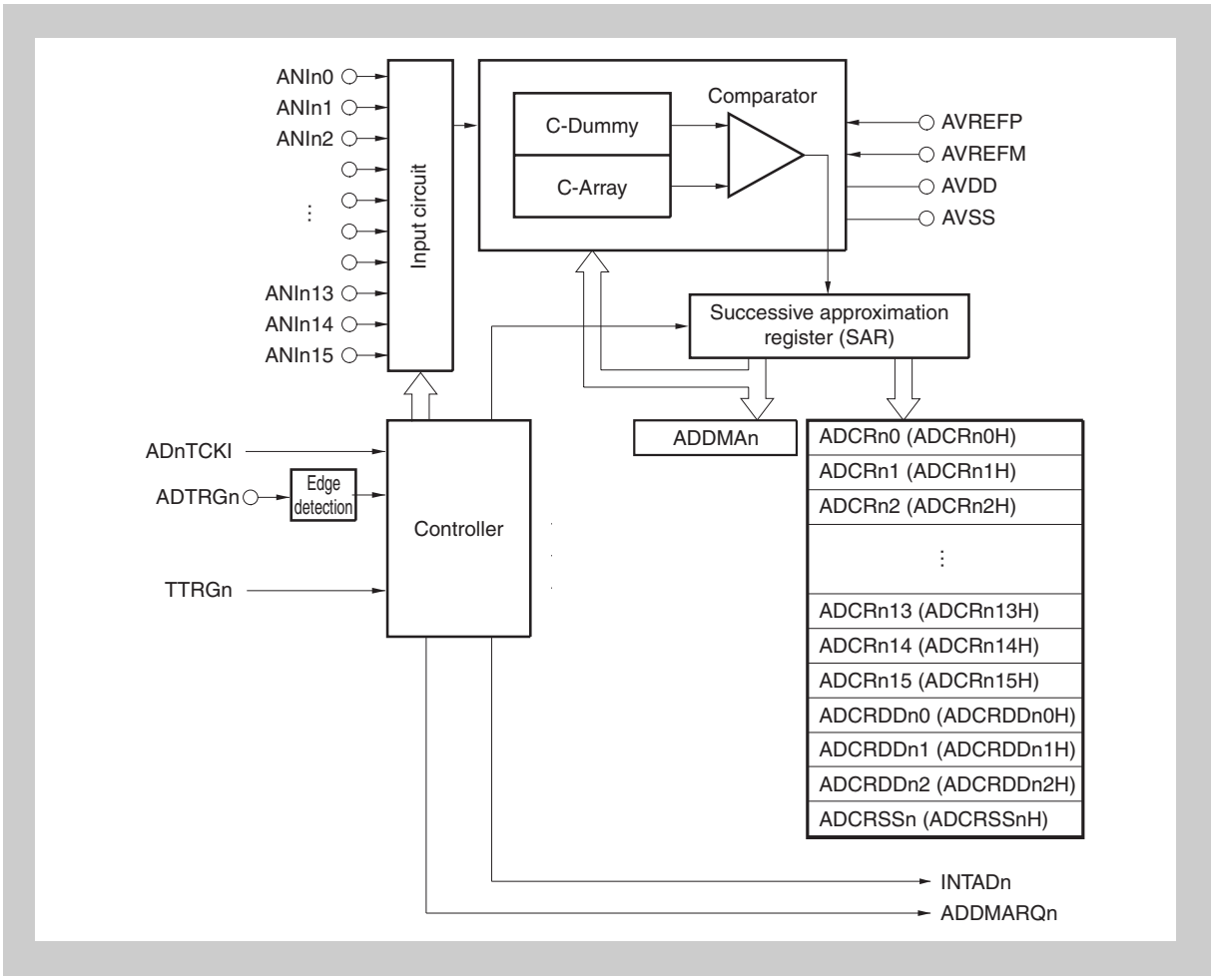


Figure 19-1 Block diagram of A/D converter (ADCn)

19.2.1 Input circuit

The input circuit selects the analog input (ANIn0 to ANIn15) according to the mode set by the ADMn0, ADMn1, ADMn2, and ADMn3 registers.

19.2.2 C-Array

Holds the charge of the differential voltage between the voltage input from the analog input pins (ANIn0 to ANIn15) and the reference voltage (1/2 AVDD), and redistributes the sampled charges.

19.2.3 C-Dummy

This block holds the reference voltage (1/2 AVDD) and assigns the reference of the comparator input.

19.2.4 Voltage Comparator

The Voltage Comparator compares the C-Array comparison potential with the C-Dummy reference potential.

19.2.5 A/D conversion result register (ADCRnm), A/D conversion result register nH (ADCRnHm)

ADCRnm is a 10-bit register that holds A/D conversion results. Each time A/D conversion is completed, the conversion results are loaded from the successive approximation register (SAR).

RESET input makes this register undefined.

19.2.6 A/D conversion result register for DMA transfer (ADDMA0)

ADDMA0 is a 16-bit register that holds the last 10-bit A/D conversion result and an overrun flag for indicating a DMA transfer failure.

19.2.7 ANIn0 to ANIn15 pins

There are 16 analog input pins for the A/D Converter n. They provide the analog signals to the ADC for conversion.

Caution Make sure that the voltages input to ANIn0 to ANIn15 do not exceed the rated values. If a voltage higher than AVDD or lower than AVSS (even within the range of the absolute maximum ratings) is input to a channel, the conversion value of the channel is undefined, and the conversion values of the other channels may also be affected.

19.2.8 AVREFM, AVREFP pins

These are the pins that provide the upper and lower reference voltages to the A/D converter. The ADC converts signals that are input to the ANIn0 to ANIn15 pins to digital signals based on the voltage applied between AVREFM and AVREFP.

19.2.9 AVSS pin

This is the ground pin of the A/D Converter. Always use this pin at the same potential as that of the V_{SS} pin even when the A/D Converter is not used.

19.2.10 AVDD pin

This is the analog power supply pin of both A/D Converters.

Cautions

1. If there is noise at the analog input pins (ANIn0 to ANIn15) or at the reference voltage input pin (AVREFP), that noise may generate an illegal conversion result.
   Software processing will be needed to avoid a negative effect on the system from this illegal conversion result.
   An example of this software processing is shown below.
   - Take the average result of a number of A/D conversions and use that as the A/D conversion result.
   - Execute a number of A/D conversions consecutively and use those results, omitting any exceptional results that may have been obtained.
2. Do not apply a voltage outside the AVREFM to AVREFP range to the pins that are used as A/D Converter input pins.

19.2.11 Clock supply

Each ADC includes 1 clock input.

- Input ADnTCKI is connected to the clock generator.

The clock connection is listed in following table:

Table 19-2 Clock input of ADC0

ADC0 clock input	Connected clock
AD0TCKI	ADCLK (32 MHz)

Conversion time settings

The conversion time can be set via the ADMn1 and ADMn3 registers. See *Table 19-6* and *Table 19-7* for the allowable register settings and the corresponding conversion times.

19.3 Control Registers

Register addresses

The ADCn register addresses are given as address offsets to the individual base addresses <base> of each ADCn.

The <base> addresses of each ADCn are listed in the following table:

Table 19-3 Register <base> addresses of ADCn

ADCn	<base> address
ADC0	FFFF $F200_H$

The ADCn are controlled and operated by means of the following registers:

Table 19-4 ADCn register list (1/2)

Register name	Shortcut	Address
ADCn channel 0 result register	ADCRn0	<base>
ADCn channel 1 result register	ADCRn1	<base> + 2_H
ADCn channel 2 result register	ADCRn2	<base> + 4_H
ADCn channel 3 result register	ADCRn3	<base> + 6_H

Table 19-4 ADCn register list (2/2)

Register name	Shortcut	Address
ADCn channel 4 result register	ADCRn4	<base> + 8_H
ADCn channel 5 result register	ADCRn5	<base> + A_H
ADCn channel 6 result register	ADCRn6	<base> + C_H
ADCn channel 7 result register	ADCRn7	<base> + E_H
ADCn channel 8 result register	ADCRn8	<base> + 10_H
ADCn channel 9 result register	ADCRn9	<base> + 12_H
ADCn channel 10 result register	ADCRn10	<base> + 14_H
ADCn channel 11 result register	ADCRn11	<base> + 16_H
ADCn channel 12 result register	ADCRn12	<base> + 18_H
ADCn channel 13 result register	ADCRn13	<base> + $1A_H$
ADCn channel 14 result register	ADCRn14	<base> + $1C_H$
ADCn channel 15 result register	ADCRn15	<base> + $1E_H$
ADCn result register for AVDD	ADCRDDn0	<base> + 20_H
ADCn result register for 2/3 AVDD	ADCRDDn1	<base> + 22_H
ADCn result register for 1/2 AVDD	ADCRDDn2	<base> + 24_H
ADCn result register for AVSS	ADCRSSn	<base> + 26_H
ADCn result register for DMA	ADDMAn	<base> + $2E_H$
ADCn mode control register 0	ADMn0	<base> + 30_H
ADCn mode control register 1	ADMn1	<base> + 31_H
ADCn mode control register 2	ADMn2	<base> + 32_H
ADCn mode control register 3	ADMn3	<base> + 33_H

19.3.1 A/D conversion result registers (ADCRn0–ADCRn15, ADCRDDn0–ADCRDDn2, ADCRSSn)

The ADCRnm register is a 10-bit register holding the A/D conversion results.

Access These registers are read-only in 16-bit or 8-bit units. When 16-bit access is performed, the ADCRnm register is specified, and when 8 bit access is performed, the ADCRnHm register holding the higher 8 bits of the conversion result is specified.

When reading the 10-bit data of the A/D conversion results from the ADCRnm register, only the higher 10 bits are valid and the lower 6 bits are always read as 0.

Address

ADCRn0 : <base>
ADCRn1 : <base> + 2_H
ADCRn2 : <base> + 4_H
ADCRn3 : <base> + 6_H
ADCRn4 : <base> + 8_H
ADCRn5 : <base> + A_H
ADCRn6 : <base> + C_H
ADCRn7 : <base> + E_H
ADCRn8 : <base> + 10_H
ADCRn9 : <base> + 12_H
ADCRn10 : <base> + 14_H
ADCRn11 : <base> + 16_H

ADCRnH0 : <base> + 1_H
ADCRnH1 : <base> + 3_H
ADCRnH2 : <base> + 5_H
ADCRnH3 : <base> + 7_H
ADCRnH4 : <base> + 9_H
ADCRnH5 : <base> + B_H
ADCRnH6 : <base> + D_H
ADCRnH7: <base> + F_H
ADCRnH8 : <base> + 11_H
ADCRnH9 : <base> + 13_H
ADCRnH10 : <base> + 15_H
ADCRnH11 : <base> + 17_H

ADCRn12 : <base> + 18_H
ADCRn13 : <base> + $1A_H$
ADCRn14 : <base> + $1C_H$
ADCRn15 : <base> + $1E_H$
ADCRDDn0: <base> + 20_H
ADCRDDn1: <base> + 22_H
ADCRDDn2: <base> + 24_H
ADCRSSn: <base> + 26_H

ADCRnH12 : <base> + 19_H
ADCRnH13 : <base> + $1B_H$
ADCRnH14 : <base> + $1D_H$
ADCRnH15 : <base> + $1F_H$
ADCRDDn0H: <base> + 21_H
ADCRDDn1H: <base> + 23_H
ADCRDDn2H: <base> + 25_H
ADCRSSnH <base> + 27_H

Initial Value Reset input causes an undefined register content.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADnm9	ADnm8	ADnm7	ADnm6	ADnm5	ADnm4	ADnm3	ADnm2	ADnm1	ADnm0	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
ADnm9	ADnm8	ADnm7	ADnm6	ADnm5	ADnm4	ADnm3	ADnm2
R	R	R	R	R	R	R	R

The correspondence between each analog input pin and the ADCRnm register is shown in Table 19-5 below.

Table 19-5 Assignment of A/D conversion result registers to analog input pins

Analog Input Pin	Assignment of A/D Conversion Result Registers	
	Select 1 Buffer Mode/ Scan Mode	Select 4 Buffer Mode
ANIn0	ADCRn0, ADCRnH0	ADCRn0 to ADCRn3, ADCRnH0 to ADCRnH3
ANIn1	ADCRn1, ADCRnH1	
ANIn2	ADCRn2, ADCRnH2	
ANIn3	ADCRn3, ADCRnH3	
ANIn4	ADCRn4, ADCRnH4	ADCRn4 to ADCRn7, ADCRnH4 to ADCRnH7
ANIn5	ADCRn5, ADCRnH5	
ANIn6	ADCRn6, ADCRnH6	
ANIn7	ADCRn7, ADCRnH7	
ANIn8	ADCRn8, ADCRnH8	ADCRn8 to ADCRn11, ADCRnH8 to ADCRnH11
ANIn9	ADCRn9, ADCRnH9	
ANIn10	ADCRn10, ADCRnH10	
ANIn11	ADCRn11, ADCRnH11	
ANIn12	ADCRn12, ADCRnH12	ADCRn12 to ADCRn15, ADCRnH12 to ADCRnH15
ANIn13	ADCRn13, ADCRnH13	
ANIn14	ADCRn14, ADCRnH14	
ANIn15	ADCRn15, ADCRnH15	
AVDD	ADCRDDn0, ADCRDDn0H	ADCRDDn0 to ADCRDDn2 w/ ADCRSSn ADCRDDn0H to ADCRDDn2H w/ ADCRSSnH
AVSS	ADCRSSn, ADCRSSnH	

The relationship between the analog voltage input to the analog input pins (ANIn0 to ANIn15) and the A/D conversion result (of the A/D conversion result register (ADCRnm)) is as follows:

$$ADCR = INT\left(\frac{V_{IN}}{AVREFP} \times 1024 + 0{,}5\right)$$

or,

$$(ADCR - 0{,}5) \times \frac{AVREFP}{1024} \leq V_{IN} < (ADCR + 0{,}5) \times \frac{AVREFP}{1024}$$

INT( ): Function that returns the integer value

V_{IN}: Analog input voltage

AVREFP: AV_{REF} pin voltage

ADCR: Value of A/D conversion result register (ADCRnm)

Figure 19-2 on page 569 shows the relationship between the analog input voltage and the A/D conversion results.

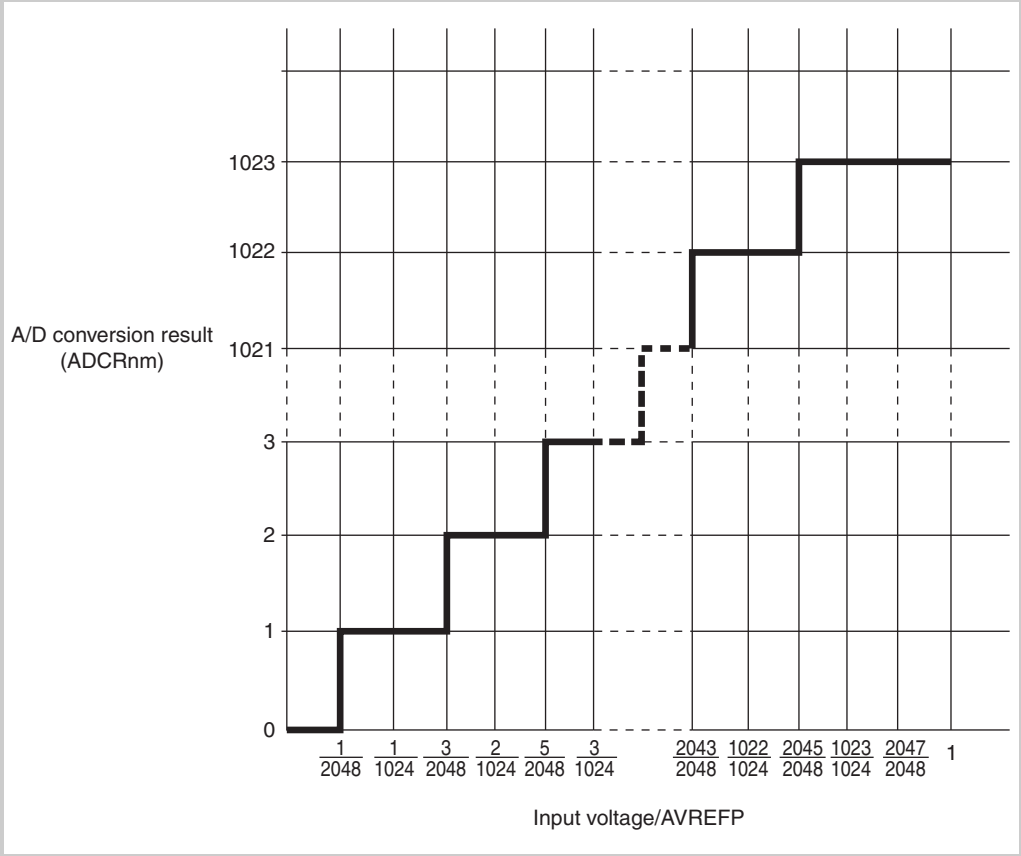


Figure 19-2 Relationship between analog input voltage and A/D conversion results

19.3.2 ADDMAn - A/D conversion result register for DMA

The ADDMAn register is a 16-bit register holding the result of the latest A/D conversion operation, and is used for DMA transfer of ADCn results into the internal RAM. It has an overrun detection flag indicating an overrun situation of the DMA transfer mechanism.

Access This register is read-only in 16-bit units.

Address <base> + $2E_H$

Initial Value Reset input causes an undefined register content.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDMAn9	ADDMAn8	ADDMAn7	ADDMAn6	ADDMAn5	ADDMAn4	ADDMAn3	ADDMAn2	ADDMAn1	ADDMAn0	0	0	0	0	0	ODFn
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

ADDMAn9 to ADDMAn0: A/D conversion result for DMA transfer

ODFn	Overrun Detection Flag
0	No A/D conversion result overrun was detected.
1	At least one A/D conversion result was overrun since the last read of the ADDMAn register.
• The ODFn flag is used for indicating a DMA transfer failure of the A/D conversion results. • The ODFn flag is cleared (0), when the A/D conversion is stopped (ADCEn bit of the ADMn0 register is cleared to 0).	

Caution Do not read the ADDMAn register by CPU during DMA transfer activities. If this register is read by CPU, overflow detection cannot be ensured.

19.3.3 ADMn0 - A/D Converter mode register 0

The ADMn0 register is an 8-bit register that specifies the operation mode, and executes conversion operations.

Access This register can be read/written in 8-bit or 1-bit units.However, bit 6 can only be read. Writing this bit is ignored.

Address <base> + 30_H

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

7	6	5	4	3	2	1	0
ADCEn	ADCSn	BSn	MSn	0	0	0	0
R/W	R	R/W	R/W	R/W	R/W	R/W	R/W

ADCEn	A/D Conversion Operation Control of ADCn
0	Disables A/D conversion operation of ADCn
1	Enables A/D conversion operation ADCn

ADCSn	A/D Conversion Status Flag of ADCn
0	A/D conversion of ADCn is stopped
1	A/D conversion of ADCn is operating

BSn	ADCn Buffer Mode Specification
0	1-buffer mode
1	4-buffer mode

MSn	ADCn Operation Mode Specification
0	Scan mode
1	Select mode

Cautions

1. When the ADCEn bit is 1 in the timer trigger mode, the trigger signal standby state is set. To clear the ADCEn bit, write 0 or reset.
   In A/D trigger mode (ADMn1TRGn[1:0] = 00), the conversion trigger is set by writing 1 to the ADCEn bit. After the operation, when the mode is changed to the timer trigger mode without clearing the ADCEn bit, the trigger input standby state is set immediately after changing the register.
2. Changing the setting of the BSn and MSn bits is prohibited while A/D conversion is enabled (ADCEn bit = 1).
3. When data is written to the ADMn0 register during an A/D conversion operation, the conversion in progress is stopped, the ADC is initialized, and a new conversion is initiated.

19.3.4 ADMn1 - A/D Converter mode register 1

The ADMn1 register is an 8-bit register that specifies the conversion operation time and trigger mode.

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

Address <base> + 31_H

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

7	6	5	4	3	2	1	0
0	0	TRGn1	TRGn0	FRn3	FRn2	FRn1	FRn0
R	R	R/W	R/W	R/W	R/W	R/W	R/W

TRGn1	TRGn0	ADCn Trigger Mode Specification
0	0	A/D trigger mode (conversion starts with ADMn0.ADCEn = 1)
0	1	Timer trigger mode
1	x	Setting prohibited

FRn3	FRn2	FRn1	FRn0	ADC Clock Selection[Note]
0	0	0	0	16 MHz
0	0	0	1	8 MHz
0	0	1	0	4 MHz
Other than above				prohibited

Note See *Table 19-6* on page 574 and *Table 19-7* on page 574 for absolute conversion times.

Cautions

1. Do not change the set value of the A/D conversion time (FRn3 to FRn0 bits) Changing the setting of the FRn3 to FRn0 bits is prohibited while A/D conversion is enabled (ADCEn bit of the ADMn0 register = 1).
2. If data is written to the ADMn1 register during an A/D conversion operation, the conversion operation is initialized and conversion is executed from the beginning.
3. during an A/D conversion operation (ADCEn bit = 1). To change the value, clear the ADCEn bit to 0.
4. When the trigger mode (TRGn1 and TGRn0 bits) is changed midway, A/D conversion can be started immediately without having to secure the A/D stabilization time by re-setting the ADCE bit to 1.
5. The Conversion operation time must be in a range from 2.0 to 10.0 µs. Settings which are not in this range are not allowed.

19.3.5 ADMn2 - A/D Converter mode register 2

The ADMn2 register is an 8-bit register that specifies the analog input pin of the A/D Converter n.

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

Address <base> + 32_H

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

7	6	5	4	3	2	1	0
0	0	0	0	ANISn3	ANISn2	ANISn1	ANIn0
R	R	R	R	R/W	R/W	R/W	R/W

ANISn3	ANISn2	ANISn2	ANISn0	ADMn.DIAGEN = 0		ADMn.DIAGEN = 1	
				Select Mode	Scan Mode	Select Mode	Scan Mode
0	0	0	0	ANIn0	ANIn0	AVDD	ANIn0-AVDD-AVSS
0	0	0	1	ANIn1	ANIn0 - ANIn1	setting prohibited	ANIn0-ANIn1-AVDD-AVSS
0	0	1	0	ANIn2	ANIn0 - ANIn2		ANIn0-ANIn2-AVDD-AVSS
0	0	1	1	ANIn3	ANIn0 - ANIn3	AVSS	ANIn0-ANIn3-AVDD-AVSS

ANISn3	ANISn2	ANISn2	ANISn0	ADMn.DIAGEN = 0		ADMn.DIAGEN = 1	
				Select Mode	Scan Mode	Select Mode	Scan Mode
0	1	0	0	ANIn4	ANIn0 - ANIn4	setting prohibited	ANIn0-ANIn4-AVDD-AVSS
0	1	0	1	ANIn5	ANIn0 - ANIn5		ANIn0-ANIn5-AVDD-AVSS
0	1	1	0	ANIn6	ANIn0 - ANIn6		ANIn0-ANIn6-AVDD-AVSS
0	1	1	1	ANIn7	ANIn0 - ANIn7		ANIn0-ANIn7-AVDD-AVSS
1	0	0	0	ANIn8	ANIn0 - ANIn8		ANIn0-ANIn8-AVDD-AVSS
1	0	0	1	ANIn9	ANIn0 - ANIn9		ANIn0-ANIn9-AVDD-AVSS
1	0	1	0	ANIn10	ANIn0 - ANIn10		ANIn0-ANIn10-AVDD-AVSS
1	0	1	1	ANIn11	ANIn0 - ANIn11		ANIn0-ANIn11-AVDD-AVSS
1	1	0	0	ANIn12	ANIn0 - ANIn12		ANIn0-ANIn12-AVDD-AVSS
1	1	0	1	ANIn13	ANIn0 - ANIn13		ANIn0-ANIn13-AVDD-AVSS
1	1	1	0	ANIn14	ANIn0 - ANIn14		ANIn0-ANIn14-AVDD-AVSS
1	1	1	1	ANIn15	ANIn0 - ANIn15		ANIn0-ANIn15-AVDD-AVSS

Cautions

1. If a channel for which no analog input pin exists is specified, the result of A/D conversion is undefined.
2. Changing the setting of the ANISn3 to ANISn0 bits is prohibited while A/D conversion is enabled (ADMn0.ADCEn = 1).
3. iF data is written to the ADMn2 register during an A/D conversion operation, the conversion operation is initialized and conversion is executed from the beginning.

19.3.6 ADMn3 - ADC Converter mode register 3

This register is used to select the extended mode operation of the A/D converter.

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

Address <base> + 33_H

Initial Value 00_H.

7	6	5	4	3	2	1	0
0	0	0	DIAGENn	PLMn	SMP1n	SMP0n	VMSENn
R	R	R	R/W	R/W	R/W	R/W	R/W

DIAGENn	Diagnostic mode selection bit
0	Conversion of AVDD and AVSS is disabled
1	Conversion of AVDD and AVSS is enabled

PLMn	A/D conversion mode selection bit
0	Continuous mode
1	Single mode

SMP1n	SMP0n	Sampling time selection bit
x	x	See *Table 19-6* on page 574 and *Table 19-7* on page 574 for conversion times.

VMSENn	Discharge function mode selection bit
0	Standard mode
1	Discharge function mode enabled

Cautions

1. Changing the setting of the PLMn and VMSENn bits is prohibited while A/D conversion is enabled (ADCEn bit of the ADMn0 register = 1).
2. Changing the setting of the DIAGENn and *SMP1-0* bits is prohibited while A/D conversion is operating (ADCEn bit of the ADMn0 register = 1 and ADCSn bit of the ADMn0 register = 1).
3. When data is written to the ADMn3 register during an A/D conversion operation, the conversion operation is stopped.
4. The conversion rate is longer when selecting the auto discharge function. *Table 19-7* on page 574 shows the resulting times.

Table 19-6 A/D conversion time without discharge mode (ADMn3.VMSEN = 0)

FR3	FR2	FR1	FR0	SMP1	SMP0	ADCLK	Total conversion time
0	0	0	0	0	0	16 MHz	2.0000
				0	1		2.5000
				1	0		3.5000
				1	1		5.5000
0	0	0	1	0	0	8 MHz	4.0000
				0	1		5.0000
				1	0		7.0000
				1	1		11.0000
0	0	1	0	0	0	4 MHz	6.0000
				0	1		7.5000
				1	0		10.5000
				1	1		16.5000

Table 19-7 A/D conversion time with discharge mode (ADMn3.VMSEN = 1)

FR3	FR2	FR1	FR0	SMP1	SMP0	ADCLK	Total conversion time
0	0	0	0	0	0	16 MHz	2.1250
				0	1		2.6250
				1	0		3.6250
				1	1		5.6250
0	0	0	1	0	0	8 MHz	4.2500
				0	1		5.2500
				1	0		7.2500
				1	1		11.2500

Table 19-7 A/D conversion time with discharge mode (ADMn3.VMSEN = 1)

FR3	FR2	FR1	FR0	SMP1	SMP0	ADCLK	Total conversion time
0	0	1	0	0	0	4 MHz	6.3750
				0	1		7.8750
				1	0		10.8750
				1	1		16.8750

19.4 Operation

The A/D Converter can perform several different types of conversion operations by specifying the trigger and operation mode. These modes are determined by the configuration of the A/D conversion mode registers ADMn0, ADMn1, and ADMn3.

There are two trigger modes and six operation modes available for A/D converter operation. The following sections describe the basic operation of the ADC and the various trigger and operation modes.

19.4.1 Basic operation

A/D conversion is executed by the following procedure.

1. The selection of the analog inputs to be used and the specification of the operation mode, trigger mode, and conversion timing should be specified using the ADMn0, ADMn1, ADMn2, and ADMn3 registers.

   When the ADCEn bit of the ADMn0 register is set to 1, A/D conversion starts in the A/D trigger mode. In the timer trigger mode, the trigger standby state is set.

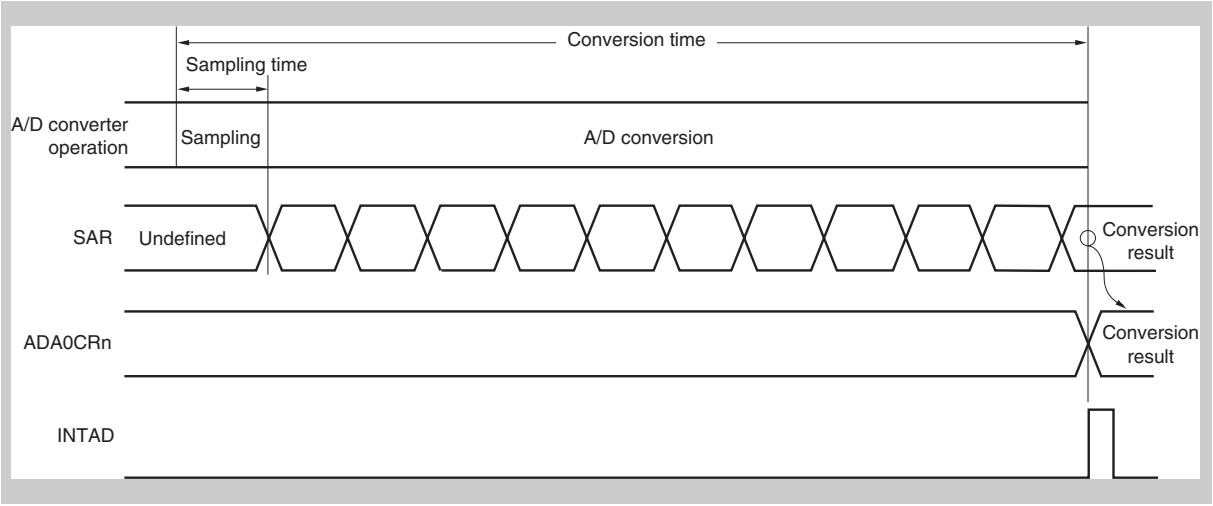
53. When A/D conversion is started, the C-array voltage on the analog input side and the C-array voltage on the reference side are compared by the comparator.

54. When the comparison of the 10 bits ends, the conversion results are stored in the ADCRnm register. When A/D conversion has been performed the specified number of times, the A/D conversion end interrupt (INTADn) is generated.

Note 1. If the setting of the ADMn0, ADMn1, ADMn2, or ADMn3 registers is changed during A/D conversion, the conversion in progress is stopped, and the result of the conversion is not stored in the ADCRnm register. The ADC is initialized, and a new conversion is initiated.

2. During the timer trigger mode, if the ADCEn bit of the ADMn0 register is set to 1, the mode changes to the standby state. The A/D conversion operation is started by the timer trigger signal (ADCSn bit in the ADMn0 register = 1), and the standby state (ADCSn bit = 0) is returned when the A/D conversion operation ends.

Figure 19-1 A/D Converter Basic Operation



19.4.2 Trigger mode

There are two types of trigger mode that serve as the start timing of A/D conversion processing: A/D trigger mode and timer trigger mode. These trigger modes are set by the TRGn1 and TRGn0 bits of the ADMn1 register.

(1) A/D trigger mode

In the A/D trigger mode, the A/D conversion is started by setting the ADCEn bit of the ADMn0 register to 1. If interrupts are enabled, an INTAD interrupt will be generated (A/D conversion completed, ADCSn bit = 0). In order to initiate the next A/D conversion, the ADCEn bit has to be rewritten to 1. However, if the ADC is configured for continuous mode, the converter will continue to process A/D conversions unless the ADCEn bit is set to 0 following the last conversion.

If data is written to the ADMn0 register during an A/D conversion operation, the conversion in progress is stopped, the ADC is initialized, and a new conversion is initiated. If data is written to the ADMn1 to ADMn3 registers during the conversion, the conversion is stopped.

(2) Timer trigger mode

This mode configures the A/D converter to initiate a conversion of the analog inputs upon receipt of a signal (trigger) from the timer.

If the ADCEn bit of the ADMn0 register is set to 1, the A/D converter waits for an event input from the timer (TTRG), and starts the conversion when the rising edge of TTRG is detected (ADCS bit of the ADMn0 register = 1).

When the conversion has finished, the converter waits for the next event input (ADCSn bit = 0).

If another event input is received during the conversion, the conversion is restarted from the beginning.

If data is written to the ADMn0 register during an A/D conversion operation, the conversion in progress is stopped, the ADC is initialized, and the converter waits for the next event input. If data is written to theADMn1 to ADMn3 registers during the conversion, the conversion is stopped.

19.4.3 Operation mode

There are two operation modes that set the ANIn0 to ANIn15 pins: select mode and scan mode. The select mode has sub-modes that consist of 1-buffer mode and 4-buffer mode. These modes are set by the BSn and MSn bits of the ADMn0 register.

(1) Select mode

In this mode, one analog input specified by the ADMn2 register is A/D converted. The conversion results are stored in the ADCRnm register corresponding to the analog input (ANInm). For this mode, the 1-buffer mode and 4-buffer mode are provided for storing the A/D conversion results.

- **1-buffer mode**

  In this mode, one analog input specified by the ADMn2 register is A/D converted. The conversion results are stored in the ADCRnm register corresponding to the analog input (ANInm). The ANInm and ADCRnm register correspond one to one, and an A/D conversion end interrupt (INTADn) is generated each time one A/D conversion ends. After conversion has finished, the next conversion operation is repeated, unless the ADCEn bit of the ADMn0 register is cleared to 0.

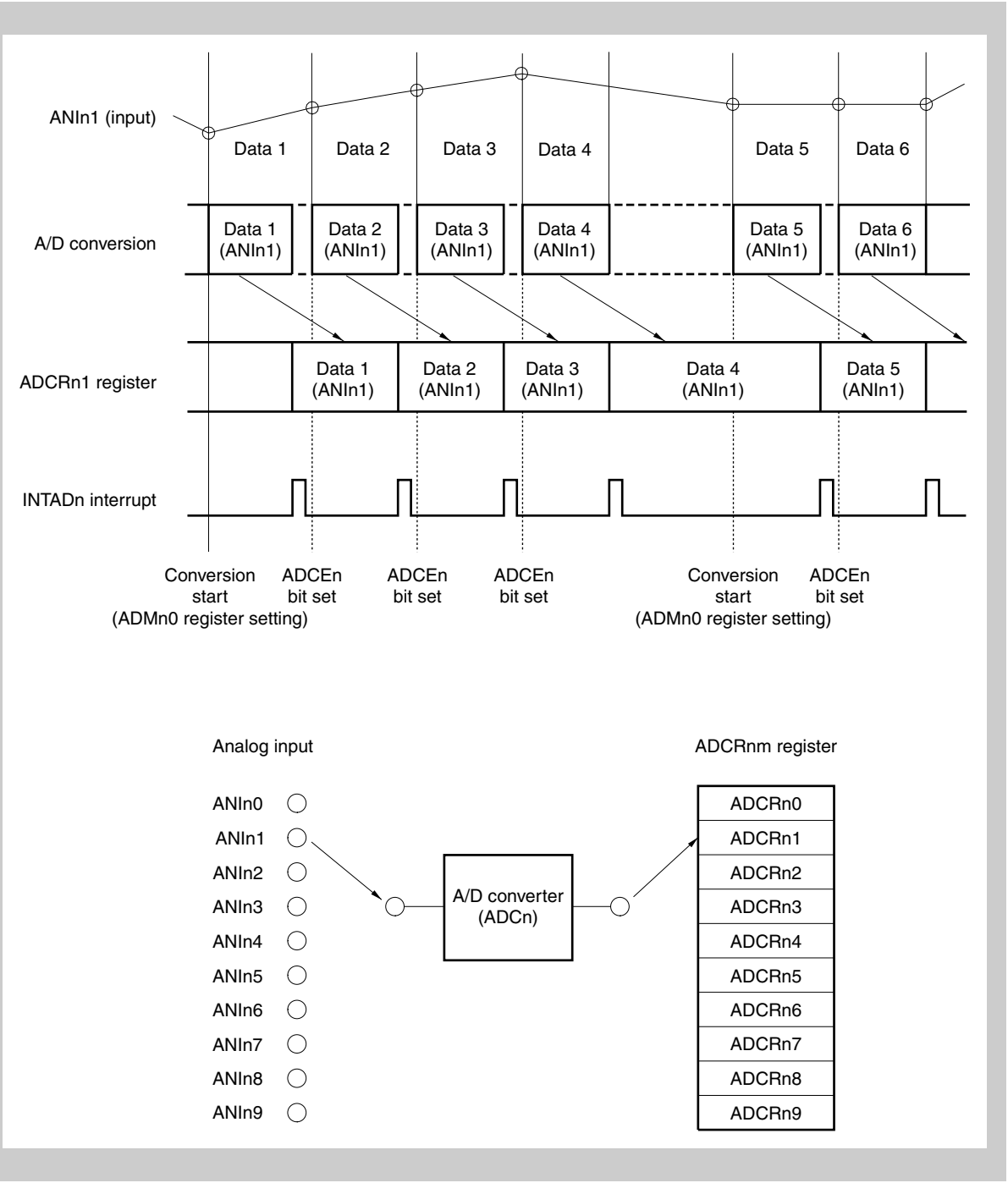


Figure 19-3 Select mode operation timing: 1-buffer mode (ANIn1)

- **4-buffer mode**

  In this mode, one analog input is A/D converted and the results are stored in the ADCRnm registers. The A/D conversion end interrupt (INTADn) is generated when the four A/D conversions end (m = 0 to 3 when one of the analog input channels ANIn0 to ANIn3 is specified, m = 4 to 7 when one of analog input channels ANIn4 to ANIn7 is specified, and m = 8 to 9 when one of the analog input channels ANIn8 or ANIn9 is specified).

  After conversion has finished, the next conversion operation is repeated, unless the ADCEn bit of the ADMn0 register is cleared to 0.

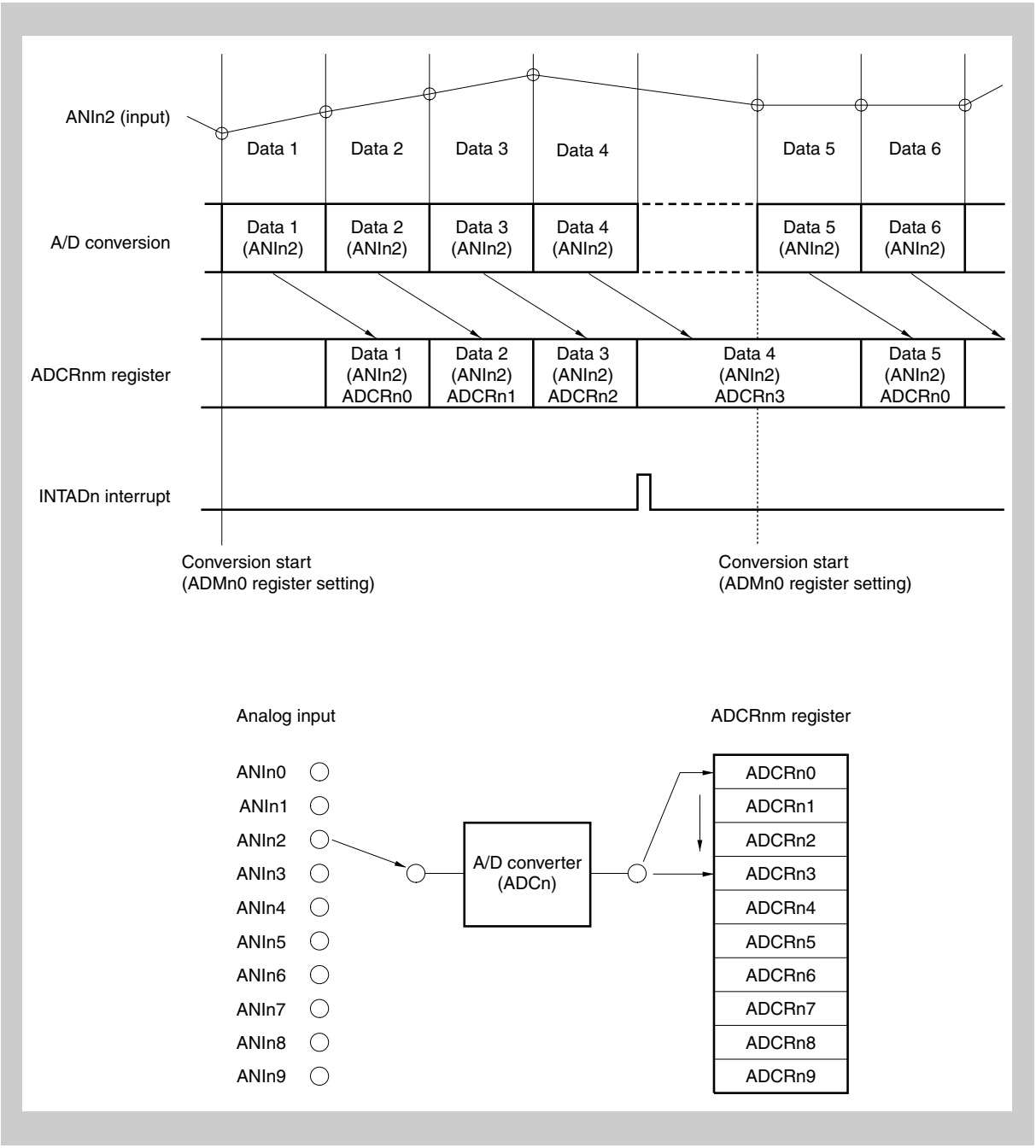


Figure 19-4 Select mode operation timing: 4-buffer mode (ANIn2)

- **Scan mode**

  In this mode, the analog inputs specified by the ADMn2 register are selected sequentially from the ANIn0 pin, and A/D conversion is executed. The A/D conversion results are stored in the ADCRnm register corresponding to the analog input. When the conversion of the specified analog input ends, the A/D conversion end interrupt (INTADn) is generated. After conversion has finished, the next conversion operation is repeated, unless the ADCEn bit of the ADMn0 register is cleared to 0.

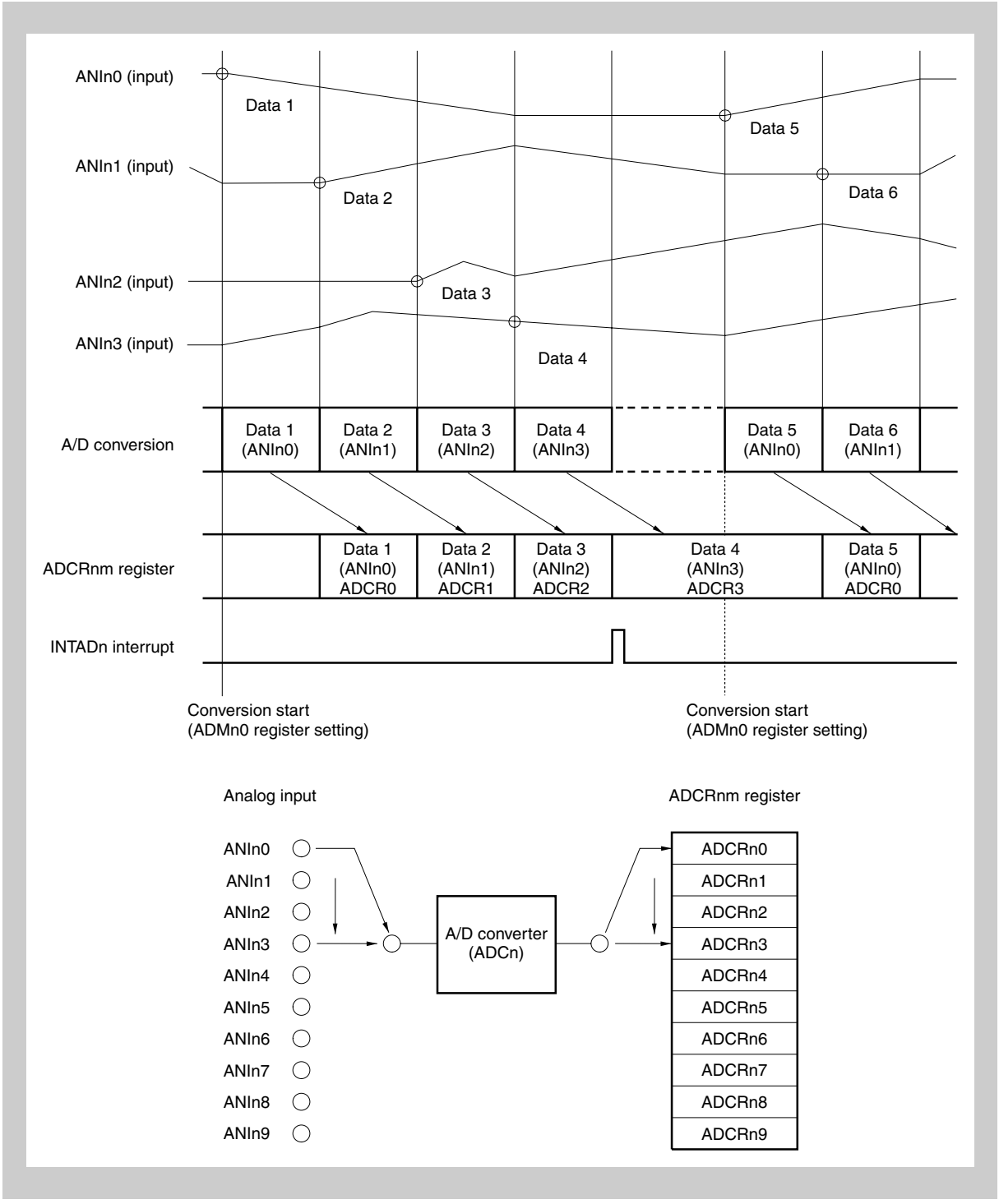


Figure 19-5 Scan mode operation timing: 4-channel scan (ANI0 to ANI3)

19.5 Operation in A/D Trigger Mode

When the ADCEn bit of the ADMn0 register is set to 1, A/D conversion is started.

19.5.1 Select mode operation

In this mode, the analog input specified by the ADMn2 register is A/D converted. The conversion results are stored in the ADCRnm register corresponding to the analog input. In the select mode, the 1-buffer mode and 4-buffer mode are supported according to the storing method of the A/D conversion results.

(1) 1-buffer mode (A/D trigger select: 1 buffer)

In this mode, one analog input is A/D converted once. The conversion results are stored in one ADCRn register. The analog input and ADCRn register correspond one to one.

Each time an A/D conversion is executed, an A/D conversion end interrupt (INTAD) is generated and A/D conversion ends. The next conversion operation is repeated, unless the ADCEn bit of the ADMn0 register is cleared to 0.

Table 19-8 Correspondence between analog input pins and ADCRnm register (A/D trigger select: 1 buffer)

Analog Input	A/D Conversion Result Register
ANInm	ADCRnm

This mode is most appropriate for applications in which the results of each first-time A/D conversion are read.

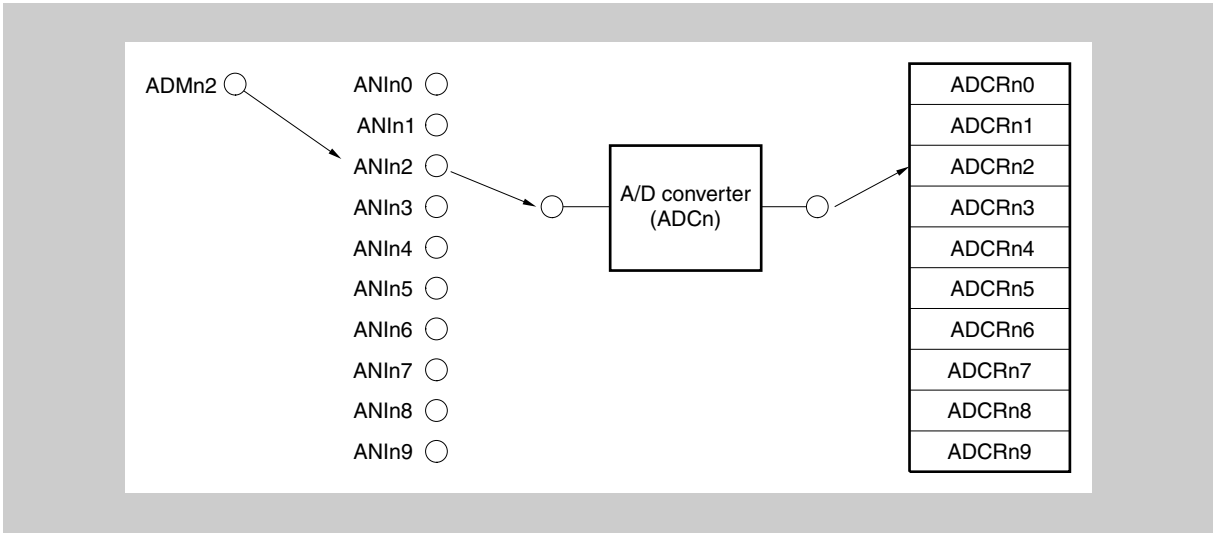


Figure 19-6 Example of 1-buffer mode operation (A/D trigger select: 1 buffer)

1. The ADCEn bit of ADMn0 register is set to 1 (enable)
55. ANIn2 is converted and the conversion result is stored in the ADCRn2 register
56. The INTAD interrupt is generated

(2) 4-buffer mode (A/D trigger select: 4 buffers)

In this mode, one analog input is A/D converted four times (two times for analog input ANIn8 or ANIn9) and the results are stored in the ADCRnm register. When the 4th A/D conversion ends, an A/D conversion end interrupt (INTADn) is generated and the A/D conversion is stopped. The next conversion operation is repeated, unless the ADCEn bit of the ADMn0 register is cleared to 0.

Table 19-9 Correspondence between analog input pins and ADCRnm register (A/D trigger select: 4 buffers)

Analog Input	A/D Conversion Result Register
ANI0 to ANI3	ADCRn0 (1st time)
	ADCRn1 (2nd time)
	ADCRn2 (3rd time)
	ADCRn3 (4th time)
ANI4 to ANI7	ADCRn4 (1st time)
	ADCRn5 (2nd time)
	ADCRn6 (3rd time)
	ADCRn7 (4th time)
ANIn8, ANIn9	ADCRn8 (1st time)
	ADCRn9 (2nd time)

This mode is suitable for applications in which the average of the A/D conversion results is calculated.

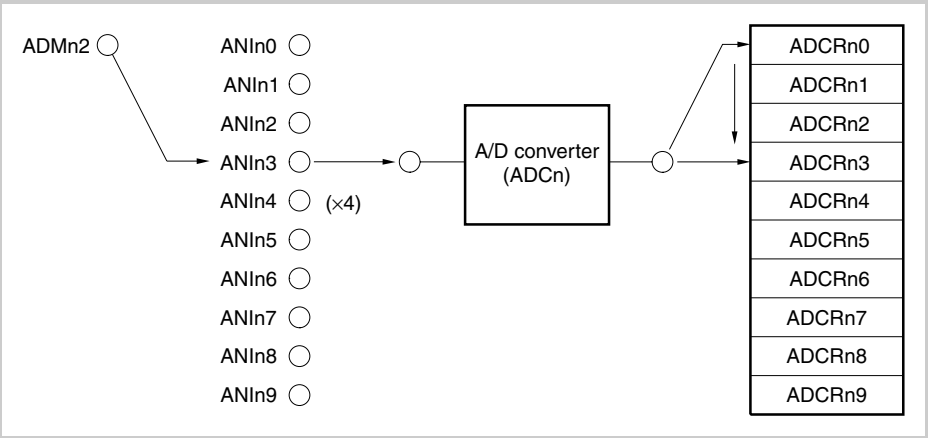


Figure 19-7 Example of 4-buffer mode operation (A/D trigger select: 4 buffers)

1. The ADCEn bit of ADMn0 register is set to 1 (enable)
57. ANIn3 is converted and the conversion result is stored in the ADCRn0 register
58. ANIn3 is converted and the conversion result is stored in the ADCRn1 register
59. ANIn3 is converted and the conversion result is stored in the ADCRn2 register
60. ANIn3 is converted and the conversion result is stored in the ADCRn3 register
61. The INTAD interrupt is generated

19.5.2 Scan mode operation

In this mode, the analog inputs specified by the ADMn2 register are selected sequentially from the ANIn0 pin, and A/D conversion is executed. The A/D conversion results are stored in the ADCRnm register corresponding to the analog input.

When conversion of all the specified analog input ends, the A/D conversion end interrupt (INTADn) is generated, and A/D conversion is stopped. The next conversion operation is repeated, unless the ADCEn bit of the ADMn0 register is cleared to 0.

Table 19-10 Correspondence between analog input pins and ADCRnm register (A/D trigger scan)

Analog Input	A/D Conversion Result Register
ANIn0 . . . ANInm[Note]	ADCRn0 . . . ADCRnm

Note Set by the ANISn3 to ANISn0 bits of the ADMn2 register.

This mode is most appropriate for applications in which multiple analog inputs are constantly monitored.

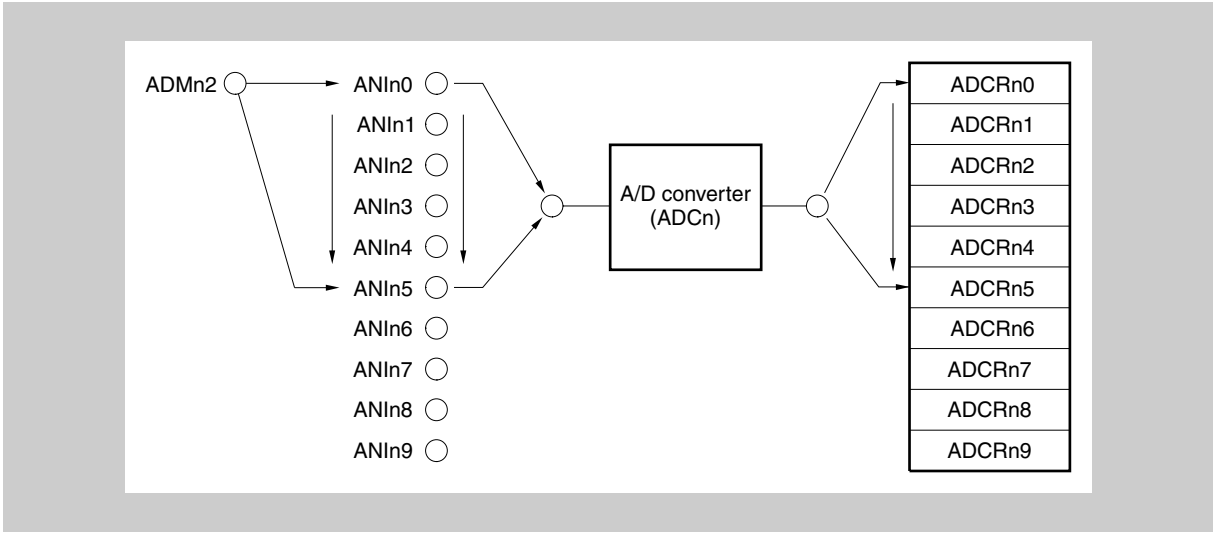


Figure 19-8 Example of scan mode operation (A/D trigger scan)

1. The ADCEn bit of ADMn0 register is set to 1 (enable)
62. ANIn0 is converted and the conversion result is stored in ADCRn0
63. ANIn1 is converted the conversion result is stored in ADCRn1
64. ANIn2 is converted the conversion result is stored in ADCRn2
65. ANIn3 is converted the conversion result is stored in ADCRn3
66. ANIn4 is converted the conversion result is stored in ADCRn4
67. ANIn5 is converted the conversion result is stored in ADCRn5
68. The INTAD interrupt is generated

19.6 Operation in Timer Trigger Mode

In this mode, the conversion timing of the analog input signal set by the ANIn0 to ANIn9 pins is defined by a timer event signal.

The analog input conversion timing is generated when an A/D Converter trigger signal from the timers is generated.

When the ADCEn bit of the ADMn0 register is set to 1, the A/D Converter waits for the timer signal and starts conversion when the timer event occurs (ADCSn bit of the ADMn0 register = 1). When conversion is finished, the converter waits for a timer event signal again (ADCSn bit = 0).

If the timer event signal occurs during conversion, the conversion operation is executed from the beginning again.

If data is written to the ADMn0 to ADMn2 registers during conversion, the conversion operation is stopped and executed from the beginning again.

19.6.1 Select mode operation

In this mode, an analog input (ANIn0 to ANIn15) specified by the ADMn2 register is A/D converted. The conversion results are stored in the ADCRnm register corresponding to the analog input. In the select mode, the 1-buffer mode and 4-buffer mode are provided according to the storing method of the A/D conversion results.

(1) 1-buffer mode operation (timer trigger select: 1 buffer)

In this mode, one analog input is A/D converted once and the conversion results are stored in one ADCRnm register.

One analog input is A/D converted once using the trigger of the timer event signals and the results are stored in one ADCRnm register. An A/D conversion end interrupt (INTADn) is generated for each A/D conversion.

Unless the ADCEn bit of the ADMn0 register is cleared to 0, A/D conversion is repeated each time a timer event signal is generated.

Table 19-11 Correspondence between analog input pins and ADCRnm register (1-buffer mode (timer trigger select: 1 buffer))

Trigger	Analog Input	A/D Conversion Result Register
Timer event signal	ANIn0	ADCRn0
	ANIn1	ADCRn1
	ANIn2	ADCRn2
	ANIn3	ADCRn3
	ANIn4	ADCRn4
	ANIn5	ADCRn5
	ANIn6	ADCRn6
	ANIn7	ADCRn7
	ANIn8	ADCRn8
	ANIn9	ADCRn9
	ANIn10	ADCRn10
	ANIn11	ADCRn11
	ANIn12	ADCRn12
	ANIn13	ADCRn13
	ANIn14	ADCRn14
	ANIn15	ADCRn15

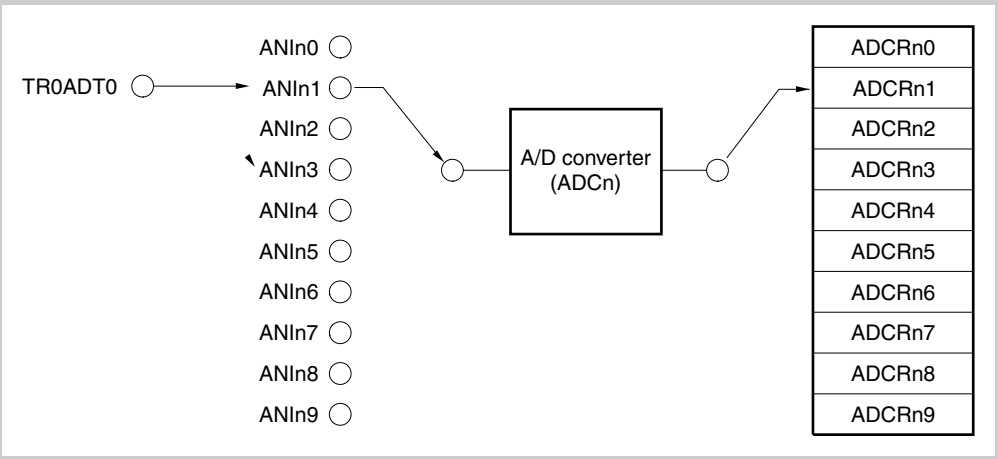


Figure 19-9 Example of 1-buffer mode operation (timer trigger select: 1 buffer) (ANIn1)

1. The ADCEn bit of ADMn0 register is set to 1 (enable)
69. The TS0ADT0 signal is generated
70. ANIn1 is converted and the conversion result is stored in ADCRn1
71. The INTADn interrupt is generated

(2) 4-buffer mode operation (timer trigger select: 4 buffers)

In this mode, A/D conversion of one analog input is executed four times, and the results are stored in the ADCRnm register.

One analog input is A/D converted four times using the timer event signals as a trigger, and the results are stored in four ADCRnm registers. The A/D

conversion end interrupt (INTADn) is generated when the four A/D conversions end.

After conversion has finished, the next conversion is repeated when a timer event signal is generated, unless the ADCEn bit of the ADMn0 register is cleared to 0.

This mode is suitable for applications in which the average of the A/D conversion results is calculated.

Table 19-12 Correspondence between analog input pins and ADCRnm register (4-buffer mode (timer trigger select: 4 buffers))

Trigger	Analog Input	A/D Conversion Result Register
Timer event signal	ANI0 to ANI3	ADCRn0 (1st time)
		ADCRn1 (2nd time)
		ADCRn2 (3rd time)
		ADCRn3 (4th time)
	ANI4 to ANI7	ADCRn4 (1st time)
		ADCRn5 (2nd time)
		ADCRn6 (3rd time)
		ADCRn7 (4th time)
	ANIn8 to ANI11	ADCRn8 (1st time)
		ADCRn9 (2nd time)
		ADCRn10 (3rd time)
		ADCRn11 (4th time)
	ANIn12 to ANIn15	ADCRn12 (1st time)
		ADCRn13 (2nd time)
		ADCRn14 (3rd time)
		ADCRn15 (4th time)

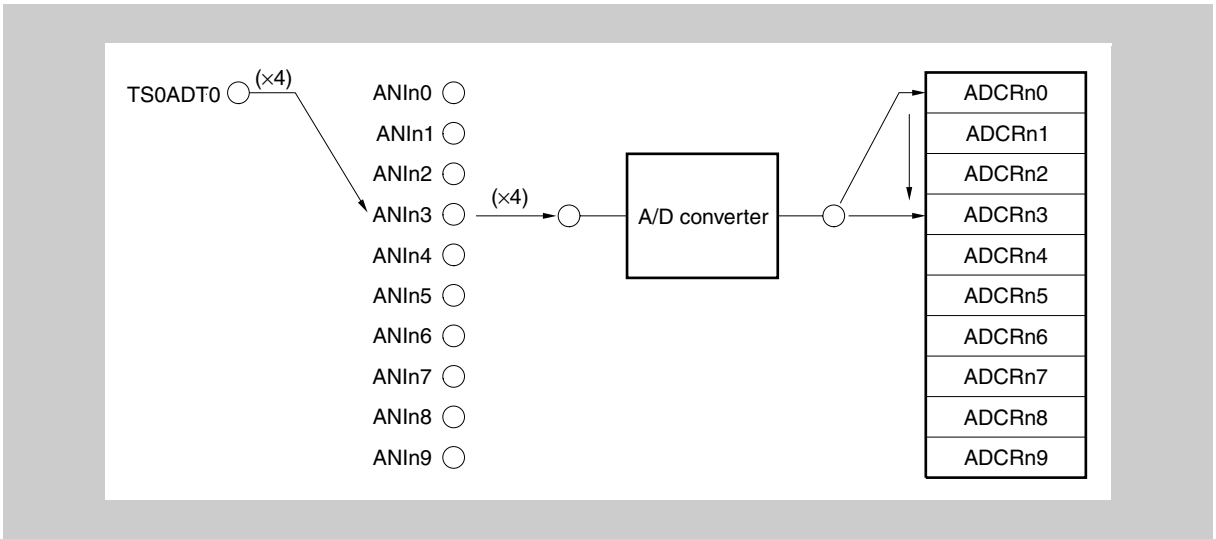


Figure 19-10 Example of 4-buffer mode operation (timer trigger select: 4 buffers) (ANIn3)

1. The ADCEn bit of ADMn0 register is set to 1 (enable)
72. TheTS0ADT0 signal is generated
73. ANIn3 is converted and the conversion result is stored in ADCR0

74. TheTS0ADT0 signal is generated
75. ANIn3 is converted and the conversion result is stored in ADCR1
76. TheTS0ADT0 signal is generated
77. ANIn3 is converted and the conversion result is stored in ADCR2
78. TheTS0ADT0 signal is generated
79. ANIn3 is converted and the conversion result is stored in ADCR3
80. The INTADn interrupt is generated

19.6.2 Scan mode operation

In this mode, the analog inputs specified by the ADMn2 register are selected sequentially from the ANIn0 pin and are A/D converted the specified number of times using the timer event signal as a trigger.

The result of conversion is stored in the ADCRnm register corresponding to the analog input. When all the specified analog input signals have been converted, an A/D conversion end interrupt (INTADn) occurs.

After conversion has finished, the A/D Converter waits for a trigger unless the ADCEn bit of the ADMn0 register is cleared to 0. When a timer event occurs again, the converter starts A/D conversion again, starting from the ANIn0 input.

This mode is most appropriate for applications in which multiple analog inputs are constantly monitored.

Table 19-13 Correspondence between analog input pins and ADCRnm register (scan mode (timer trigger scan))

Trigger	Analog Input	A/D Conversion Result Register
Timer event signal	ANIn0	ADCRn0
	ANIn1	ADCRn1
	ANIn2	ADCRn2
	ANIn3	ADCRn3
	ANIn4	ADCRn4
	ANIn5	ADCRn5
	ANIn6	ADCRn6
	ANIn7	ADCRn7
	ANIn8	ADCRn8
	ANIn9	ADCRn9
	ANIn10	ADCRn10
	ANIn11	ADCRn11
	ANIn12	ADCRn12
	ANIn13	ADCRn13
	ANIn14	ADCRn14
	ANIn15	ADCRn15

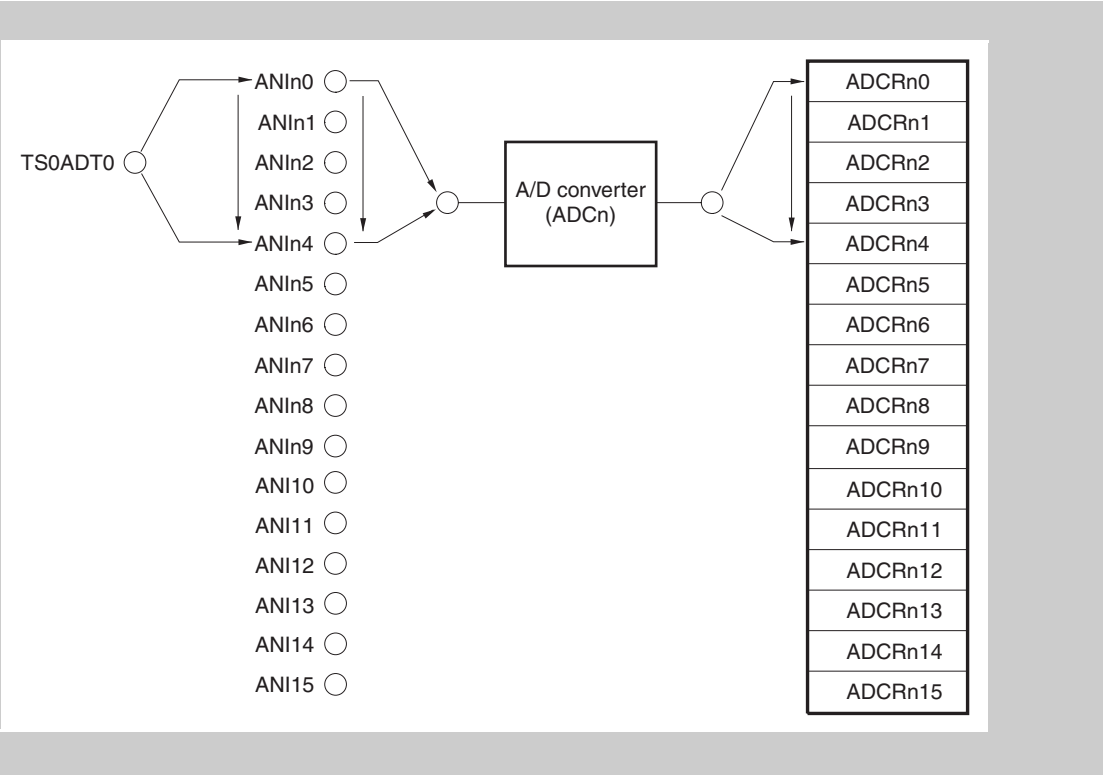


Figure 19-11 Example of scan mode operation (timer trigger scan) (ANIn0 to ANIn4)

1. The ADCEn bit of ADMn0 register is set to 1 (enable)
81. The TS0ADT0 signal is generated
82. ANIn0 is converted the conversion result is stored in ADCRn0
83. ANIn1 is converted and the conversion result is stored in ADCRn1
84. ANIn2 is converted and the conversion result is stored in ADCRn2
85. ANIn3 is converted and the conversion result is stored in ADCRn3
86. ANIn4 is converted and the conversion result is stored in ADCRn4
87. The INTADn interrupt is generated

19.7 Extended Functions

19.7.1 Automatic Discharge Operation

The automatic discharge operation internally connects the sample and hold circuit to GND in order to open inputs and other failure modes. This function is disabled by default and is controlled by the VMSENn bit of the ADMn3 register.

19.7.2 4.4.2 Diagnostic Mode

The diagnostic mode configures the A/D converter to perform a conversion operation on the *AVDD and AVSS* terminals. Using this mode allows the user to implement a software algorithm in the application program to calibrate conversion results and thereby reduce the effect of certain error factors such as full scale error and zero offset error. This mode is disabled by default, and is controlled by the DIAGENn bit of the ADMn3 register.

(1) Operation in Select Mode

In select modes, *AVDD or AVSS* terminals can be selected using the ANISn3 to ANISn0 bits of register ADMn2. The A/D conversion result is stored in the appropriate conversion result register and interrupt (INTAD) is output. Operation is performed in the same manner described in section *Chapter 19, Section 19.5.1, Select mode operation*.

(2) Operation in Scan Mode

In scan modes, *AVDD and AVSS* terminals are scanned immediately after A/D conversion of the Scan mode conversion end channel specified in register ADMn2. The A/D conversion result is stored in the appropriate conversion result register and interrupt (INTAD) is output. Operation is performed in the same manner described in *Chapter 19, Section 19.5.2, Scan mode operation*.

19.8 Precautions

19.8.1 Stopping conversion operation

When the ADCEn bit of the ADMn0 register is cleared to 0 during a conversion operation, the conversion operation stops and the conversion results are not stored in the ADCRnm register.

19.8.2 Timer trigger interval

Set the interval (input time interval) of the trigger in the timer trigger mode longer than the conversion time specified by the FRn3 to FRn0 bits of the ADMn1 register.

When 0 < interval ≤ conversion operation time

When the following timer trigger is input during a conversion operation, the conversion operation is aborted and the conversion starts according to the last timer trigger input.

When conversion operations are aborted, the conversion results are not stored in the ADCRnm register. However, the number of times the trigger has been input is counted. When an interrupt occurs, the values that have been converted are stored in the ADCRnm register.

19.8.3 Operation in HALT mode

A/D conversion continues in the HALT mode. When this mode is released by NMI input or unmasked maskable interrupt input, the ADMn0, ADMn1, and ADMn2 registers as well as the ADCRnm register hold the value.

19.8.4 Input range of ANIn0 to ANIn15

Use the input voltage at ANIn0 to ANIn15 within the specified range. If a voltage outside the range of AV_{REF} is input to any of these pins (even within the absolute maximum rating range), the converted value of the channel is undefined. In addition, the converted value of the other channels may also be affected.

19.8.5 Conflicts

- **Conflict between writing A/D conversion result registers (ADCRnm, ADCRnHm) at end of conversion and reading ADCRnm and ADCRnHm registers by instruction**

  Reading the ADCRnm and ADCRnHm registers takes precedence. After these registers have been read, the new conversion result is written to the ADCRnm and ADCRnHm registers.

- **Conflict between writing ADCRnm and ADCRnHm at end of conversion and writing ADMn1 or ADMn2 register**

  If ADMn1 or ADMn2 register is written immediately after ADCRnm and ADCRnHm have been written on completion of A/D conversion, the conversion result is written to the ADCRnm and ADCRnHm registers, but the A/D conversion end interrupt (INTADn) may not occur depending on the timing.

19.9Reading the A/D Converter Characteristics Table

This section explains the terms that are used in reading the A/D Converter Characteristics Table.

19.9.1 Resolution

Resolution refers to the minimum analog input voltage that can be recognized (refer to the Electrical Specification). 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

$= (AV_{REF0} – 0)/100$

$= AV_{REF0} /100$

When the resolution is 10 bits, 1LSB is as follows:

$1LSB = 1/2^{10} = 1/1024$

= 0.098 “%FSR”

The accuracy is determined by the overall error, independently of the resolution.

19.9.2 Overall Error

Overall Error 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.

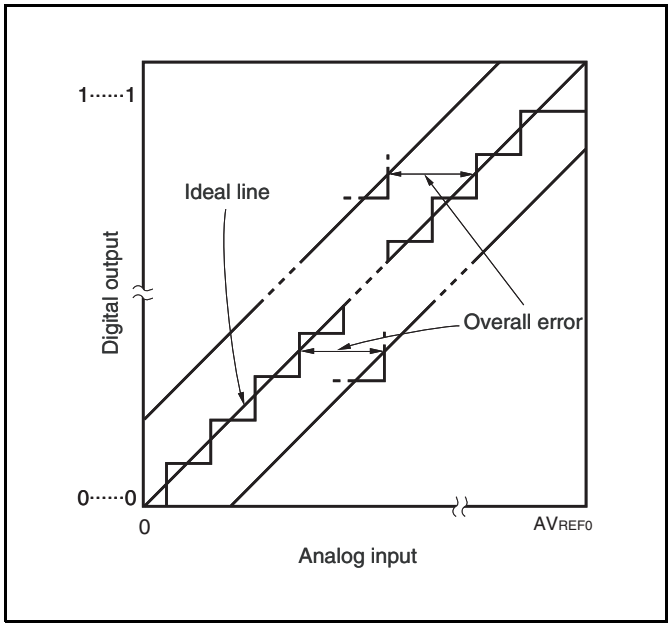


Figure 19-1 Overall Error

19.9.3 Quantization Error

Quantization Error 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 within 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.

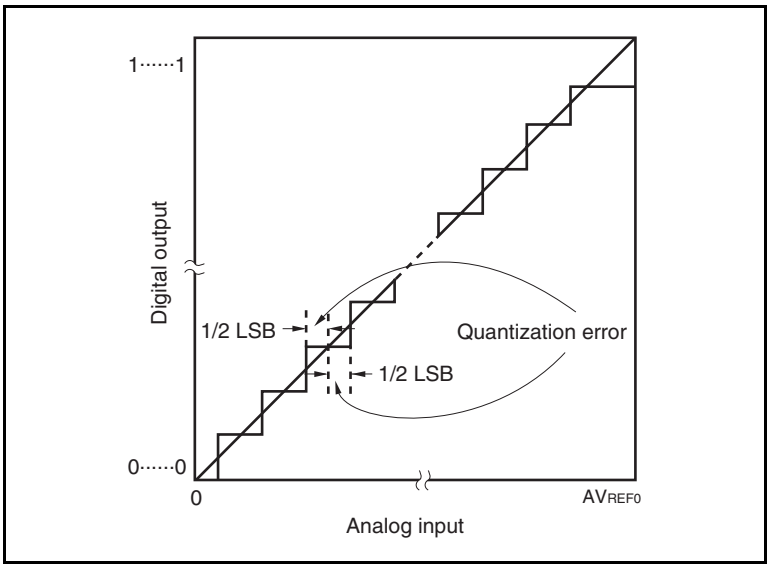


Figure 19-2 Quantization Error

19.9.4 Zero-Scale Error

Zero-Scale Error 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).

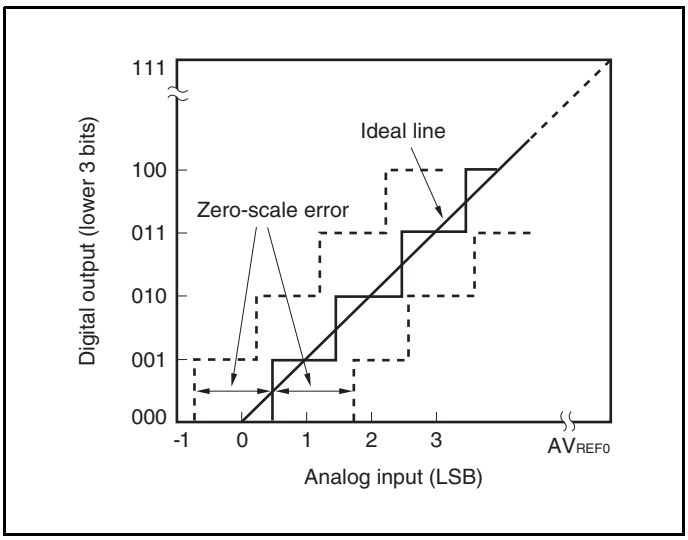


Figure 19-3 Zero-Scale Error

19.9.5 Full-Scale Error

Full-Scale Error 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).

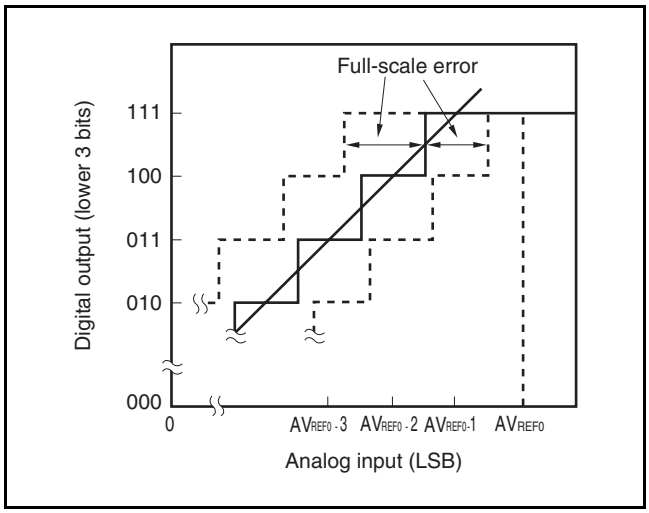


Figure 19-4 Full-Scale Error

19.9.6 Differential Linearity Error

Ideally, the step to output a specific code is 1 LSB. The *Differential Linearity Error* indicates the difference between the actually measured value and its theoretical value when a specific code is output.

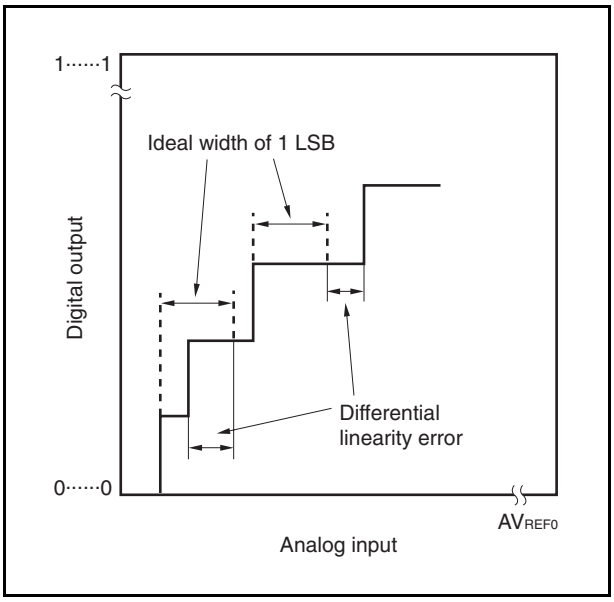


Figure 19-5 Differential Linearity Error

19.9.7 Integral Linearity Error

The *Integral Linearity 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.

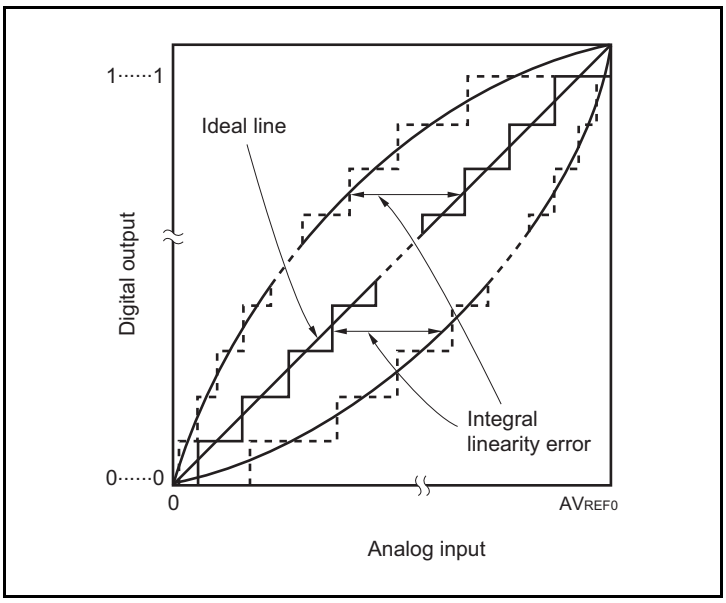


Figure 19-6 Integral Linearity Error

19.9.8 Conversion Time

Conversion Time 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.

19.9.9 Sampling Time

Sampling Time is the time for which the analog switch is ON to load an analog voltage to the sample & hold circuit.

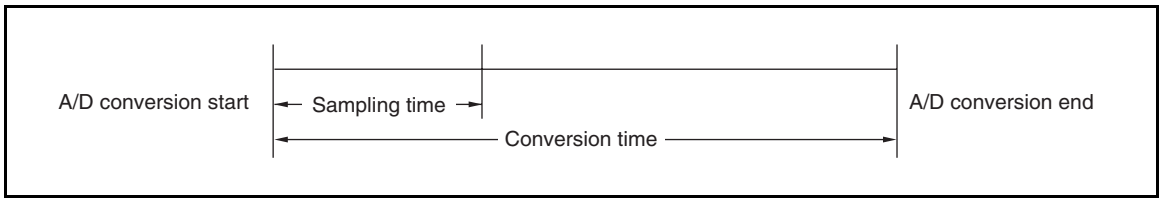


Figure 19-7 Sampling Time

Chapter 20 RESET Function

The V850E/Rx3 microcontroller has several system reset functions, which are provided to ensure a proper start up of the device.

20.1 Overview

Features A reset can be caused by the following events:

- External reset signal through $\overline{\text{RESET}}$ pin
- Overflow of the Watchdog Timer (WDT)
- Power-on clear (POC) [5V devices only]
- Comparison of the supply voltage and the low-voltage indicator (LVI) reference voltage [5V devices only]
- Software reset

A reset output function ($\overline{\text{RESOUT}}$) is also available, but is slightly different on on 3V and 5V devices.

20.2 External Reset ($\overline{\text{RESET}}$)

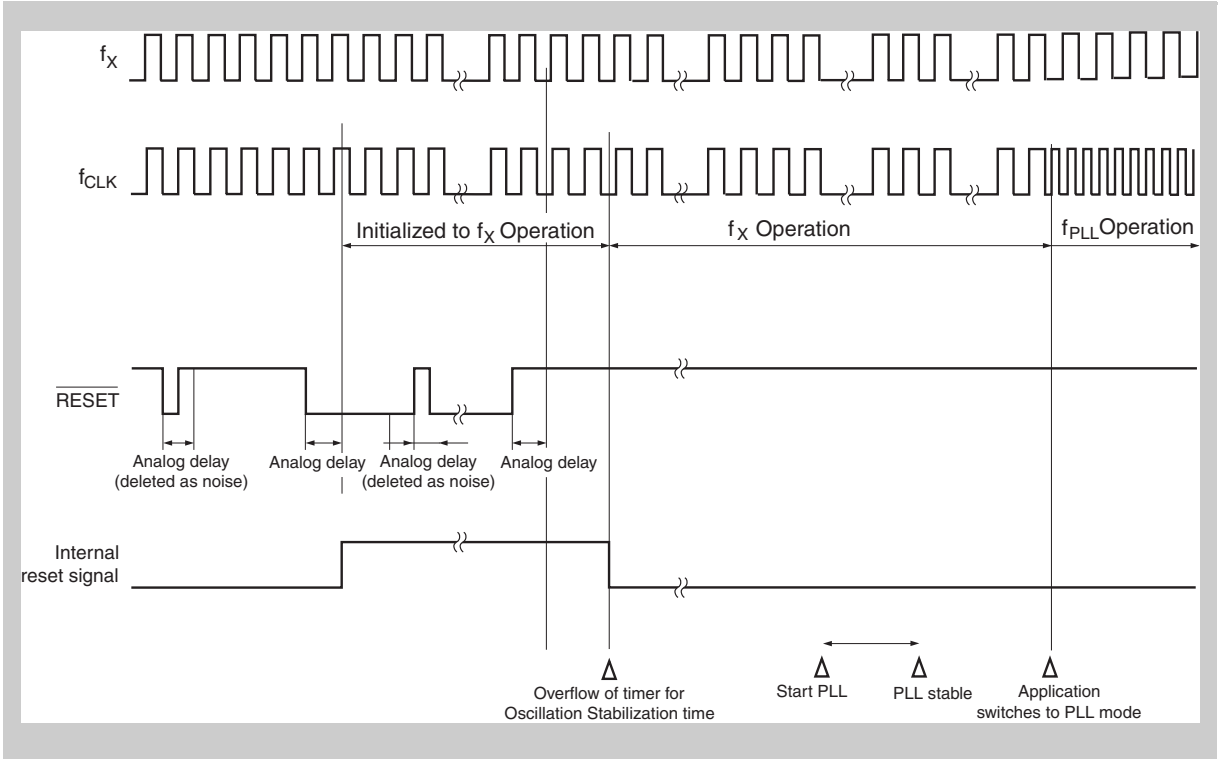
When a low level signal is input to the $\overline{\text{RESET}}$ pin, the system is reset, and each hardware unit is initialized.

When the level of the $\overline{\text{RESET}}$ pin is changed from low to high, the reset status is released, and the oscillation stabilization timer (OSTS) begins counting. When the OSTS timer has elapsed, the CPU starts program execution.

The $\overline{\text{RESET}}$ pin incorporates a noise eliminator, which is an analog filter applied to the reset signal. Even if no clock is active in the controller, the external $\overline{\text{RESET}}$ can keep the controller in the reset state.

The following figure illustrates the timing when an external reset is performed. It shows the effect of the noise elimination through the analog delay. The analog filter input measures pulses up to a certain width as noise and suppresses them. For the minimum $\overline{\text{RESET}}$ pulse width, refer to the Electrical Specification.

Figure 20-1 Timing of Reset Operation by $\overline{\text{RESET}}$ Pin Input



Caution The on-chip debug mode (flash memory products only) may be set depending on the pin status after reset has been released. For details, see *Chapter 22.*

The table below shows the hardware status during an external reset or POC.

Table 20-1 Hardware Status During/After External Reset

Item	During Reset	After Reset
Main clock oscillator (f_X)	Continues oscillation	
Internal Ring OSC (f_{RH})	Stops oscillation	Starts oscillation
Peripheral clock (f_{XX1} or divided)	Stops operation	• Operation starts after oscillation stabilization time • Operating on main oscillator after FMOP read
Internal system clock (f_{CLK}), CPU clock (f_{CPU})	Stops operation	Starts operation after oscillation stabilization time
CPU	Initialized	Program execution starts after oscillation stabilization time
Watchdog timer 2	Stops operation	Starts operation
Internal RAM	Undefined	

Table 20-1 Hardware Status During/After External Reset (Continued)

Item			During Reset	After Reset
I/O lines (ports/ alternate-function pins)	3V devices[a]	$\overline{\text{RESOUT}}$	Driving low level	
		$\overline{\text{PUDSEL}}$	Input	High impedance
		other I/O, $\overline{\text{PUDSEL}}$=0	Pull-up / -down	
		other I/O, $\overline{\text{PUDSEL}}$=1	High impedance	
	5V devices[b]		High impedance	
On-chip peripheral I/O registers			Initialized to specified status	
Other on-chip peripheral functions			Operation stops	Operation can be started after oscillation stabilization time

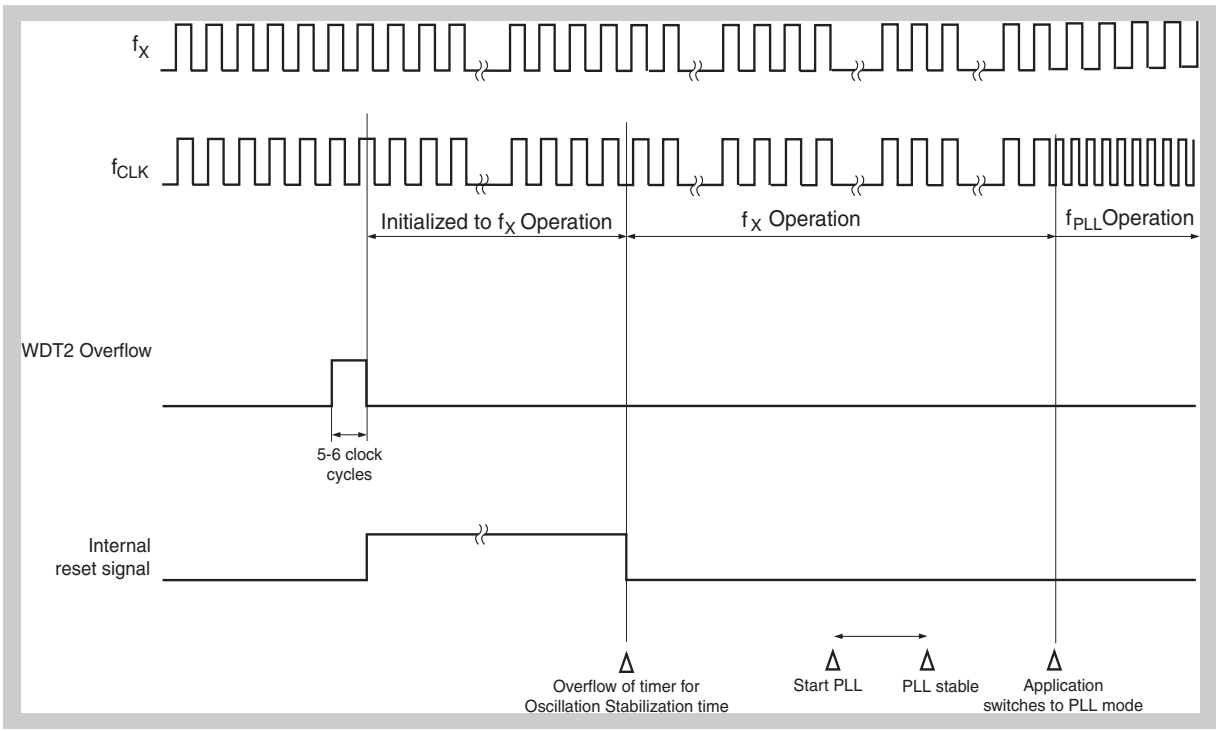
a) µPD70F3464, µPD70F3465 and µPD70F3466
b) µPD70F3470, µPD70F3471 and µPD70F3472

20.3 Reset by Watchdog Timer

Watchdog Timer (WDT) may be configured to generate a system reset if Watchdog Timer overflows. After the system reset is executed, the hardware is initialized to the initial status.

The reset period continues as determined by the analog delay, and then the reset status is automatically released. Following reset release, the CPU starts program execution after securing the oscillation stabilization time (initial value of OSTS register: $2^{13}/f_X$) of the main clock oscillator. The main clock oscillator is not stopped during the reset period.

Figure 20-2 Timing of Reset Operation by WDT Overflow



The following table shows the hardware status during and after WDT overflow.

Table 20-2 Hardware Status During/After Reset by WDT Overflow

Item			During Reset	After Reset
Main clock oscillator (f_X)			Continues oscillation	
Internal Ring OSC (f_{RH})			Stops oscillation	Starts oscillation
Peripheral clock (f_{XX1} or divided)			Stops operation	• Operation starts after oscillation stabilization time • Operating on main oscillator after FMOP read
Internal system clock (f_{CLK}), CPU clock (f_{CPU})			Stops operation	Starts operation after oscillation stabilization time
CPU			Initialized	Program execution starts after oscillation stabilization time
Watchdog timer 2			Stops operation	Starts operation
Internal RAM			Undefined	
I/O lines (ports/ alternate-function pins)	3V devices[a]	$\overline{\text{RESOUT}}$	Driving low level	
		$\overline{\text{PUDSEL}}$	Input	High impedance
		other I/O, $\overline{\text{PUDSEL}}$=0	Pull-up / -down	
		other I/O, $\overline{\text{PUDSEL}}$=1	High impedance	
	5V devices[b]		High impedance	
On-chip peripheral I/O registers			Initialized to specified status (OCDM is unchanged)	
Other on-chip peripheral functions			Operation stops	Operation can be started after oscillation stabilization time

a) µPD70F3464, µPD70F3465 and µPD70F3466
b) µPD70F3470, µPD70F3471 and µPD70F3472

20.4 Reset at Power-On-Clear (POC)

The Power-On-Clear (POC) feature is only available for 5V device.

The POC feature compares the power supply voltage (V_{DD}) with an internal reference voltage (V_{POC}). It ensures that the microcontroller only operates as long as the power supply exceeds a specified limit. When $V_{DD} < V_{POC}$, the POC feature generates an internal reset signal.

After POC reset, the RESF register is cleared to 00_H.

The following diagram illustrates the POC circuit.

Figure 20-3 POC Circuit

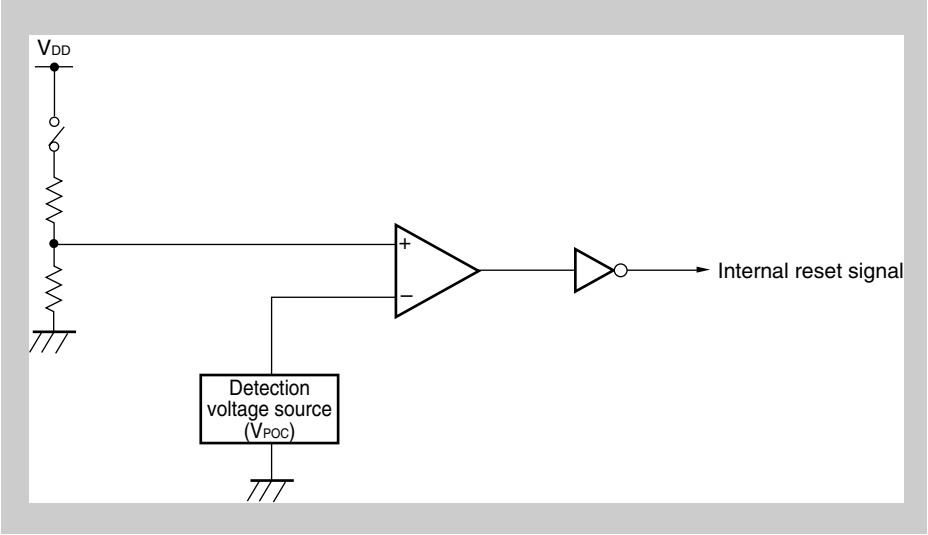


Figure 20-4 shows the timing of the POC reset.

Figure 20-4 Power-on-Clear Reset Operation

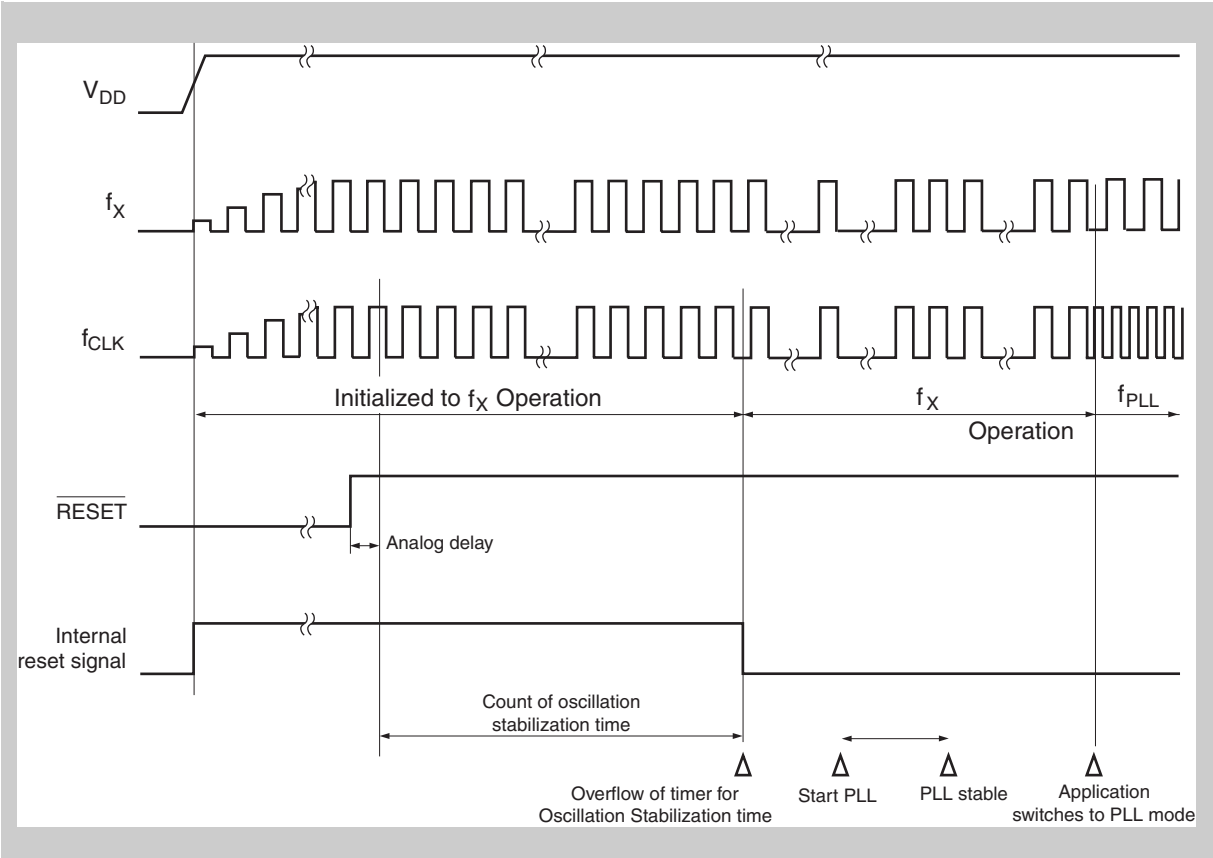


Table 20-3 Hardware Status During/After Reset POC reset[a]

Item	During Reset	After Reset
Main clock oscillator (f_X)	Running, worst case stabilization time secured by counter	
Peripheral clock (f_{XX1} or divided)	Operation stops	• Operation starts after oscillation stabilization time • Operating on main oscillator after FMOP read
Internal system clock (f_{CLK}), CPU clock (f_{CPU})	Operation stops	Operation starts after securing of oscillation stabilization time
CPU	Initialized	Program execution starts after securing of oscillation stabilization time
Watchdog timer 2	Operation stops	Operation starts
Internal RAM	Undefined	
I/O lines (ports/alternate-function pins)	High impedance	
On-chip peripheral I/O registers	Initialized to specified status, OCDM register is cleared (00_H).	
Other on-chip peripheral functions	Operation stops	Operation can be started after oscillation stabilization time

a) Only for 5V devices

20.5 Reset Operation by Low Voltage Indicator (LVI)

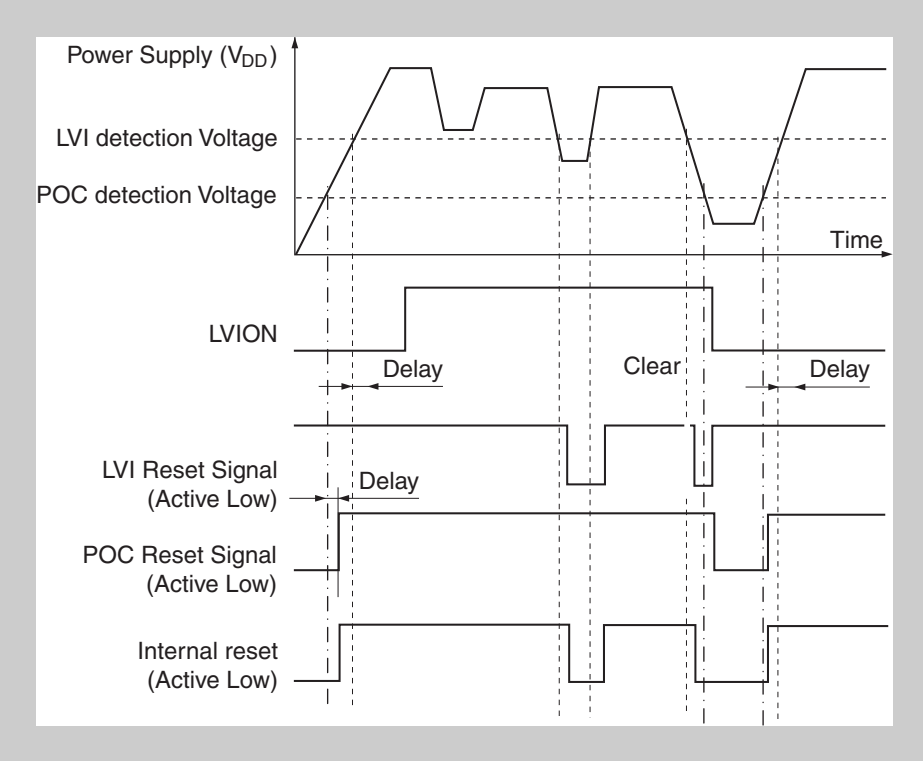
The Low Voltage Indicator (LVI) feature is only available on 5V devices.

If the LVI operation is enabled (LVIMD bit of the LVIM register is set to 1), a system reset is generated if the V_{DD1} power supply falls below a user-selected reference voltage level (V_{LVI}).

The reset status lasts from detection of the supply voltage drop until the supply voltage rises above the LVI detection level. After reset release, the CPU starts boot processing execution after the oscillation stabilization time elapses (initial value of OSTS register: $2^{13}/f_X$).

The main clock oscillator is not stopped during the reset period.

Figure 20-5 Timing of Reset Operation by Low-Voltage Indicator



Note 1. The time period between a dashed line and the rising or falling edge of a control signal represents the minimum analog delay in the circuit.

2. The LVION and LVI reset signal are set to the inactive state by the POC reset signal active level.

The following table shows the hardware status during/after LVI reset.

Table 20-4 Hardware Status During/After LVI Reset[a]

Item	During Reset	After Reset
Main clock oscillator (f_X)	Continues oscillation	
Internal Ring OSC (f_{RH})	Stops oscillation	Starts oscillation
Peripheral clock (f_{XX1} or divided)	Operation stops	Operation starts after oscillation stabilization time
Internal system clock (f_{CLK}), CPU clock (f_{CPU})	Operation stops	Operation starts after oscillation stabilization time
CPU	Initialized	Program execution starts after oscillation stabilization time
Watchdog timer 2	Operation stops	Operation starts
Internal RAM	Undefined	
I/O lines (ports/alternate-function pins)	High impedance	
On-chip peripheral I/O registers	Initialized to specified status, OCDM register is unchanged.	
Other on-chip peripheral functions	Operation stops	Operation can be started after oscillation stabilization time

a) Only 5V devices

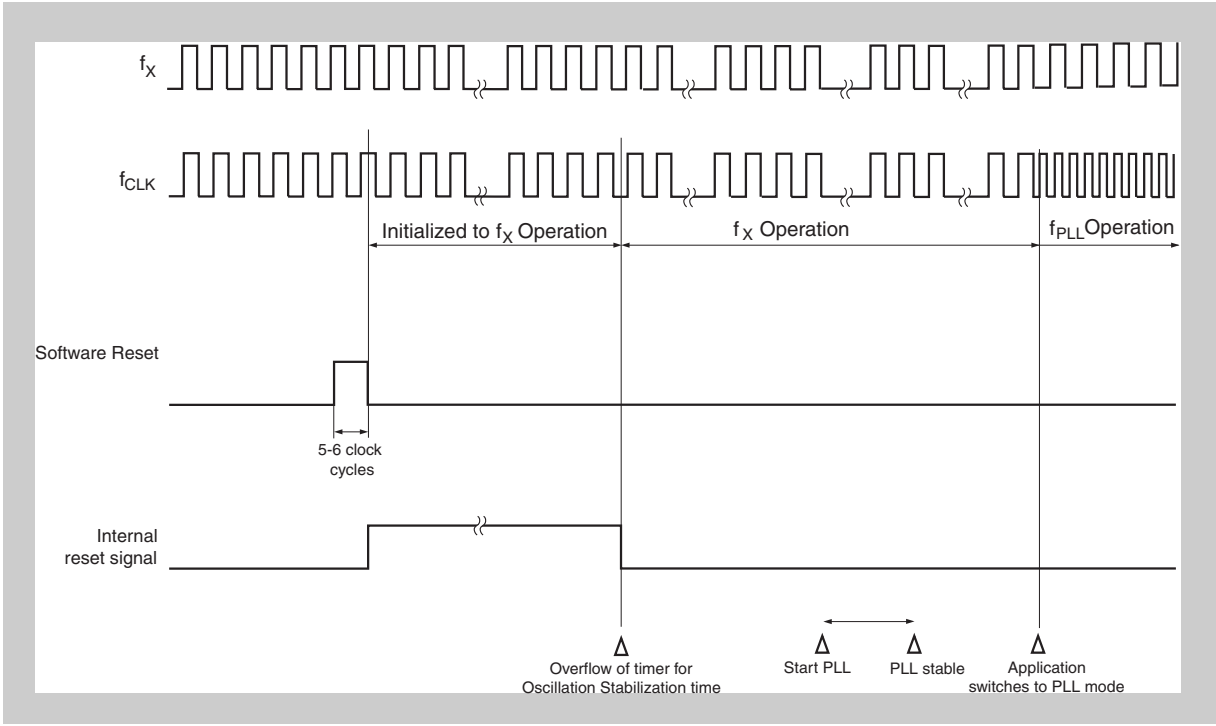
20.6 Software Reset

Writing in the Software Reset Trigger Register (RESSWT) with any data will generate a software reset of the device. The corresponding flag in RESSTAT is then set.

Caution When a software reset is generated, the watchdog timer is not reset.

The main clock oscillator is not stopped during the reset period.

Figure 20-6 Timing of Reset Operation by Software Reset



The following table shows the hardware status during and after a software reset.

Table 20-5 Hardware Status During/After a Software Reset

Item	During Reset	After Reset
Main clock oscillator (f_X)	Continues oscillation	
Internal Ring OSC (f_{RH})	Stops oscillation	Starts oscillation
Peripheral clock (f_{XX1} or divided)	Stops operation	• Operation starts after oscillation stabilization time • Operating on main oscillator after FMOP read
Internal system clock (f_{CLK}), CPU clock (f_{CPU})	Stops operation	Starts operation after oscillation stabilization time
CPU	Initialized	Program execution starts after oscillation stabilization time
Watchdog timer 2	Stops operation	Starts operation

Table 20-5 Hardware Status During/After a Software Reset (Continued)

Item			During Reset	After Reset
Internal RAM			Undefined	
I/O lines (ports/ alternate-function pins)	3V devices[a]	$\overline{\text{RESOUT}}$	Driving low level	
		$\overline{\text{PUDSEL}}$	Input	High impedance
		other I/O, $\overline{\text{PUDSEL}}$=0	Pull-up / -down	
		other I/O, $\overline{\text{PUDSEL}}$=1	High impedance	
	5V devices[b]		High impedance	
On-chip peripheral I/O registers			Initialized to specified status (OCDM is unchanged)	
Other on-chip peripheral functions			Operation stops	Operation can be started after oscillation stabilization time

a) µPD70F3464, µPD70F3465 and µPD70F3466
b) µPD70F3470, µPD70F3471 and µPD70F3472

20.7 $\overline{\text{RESOUT}}$ function

20.7.1 $\overline{\text{RESOUT}}$ function on 3V devices

This chapter describes the $\overline{\text{RESOUT}}$ function only for 3V devices.

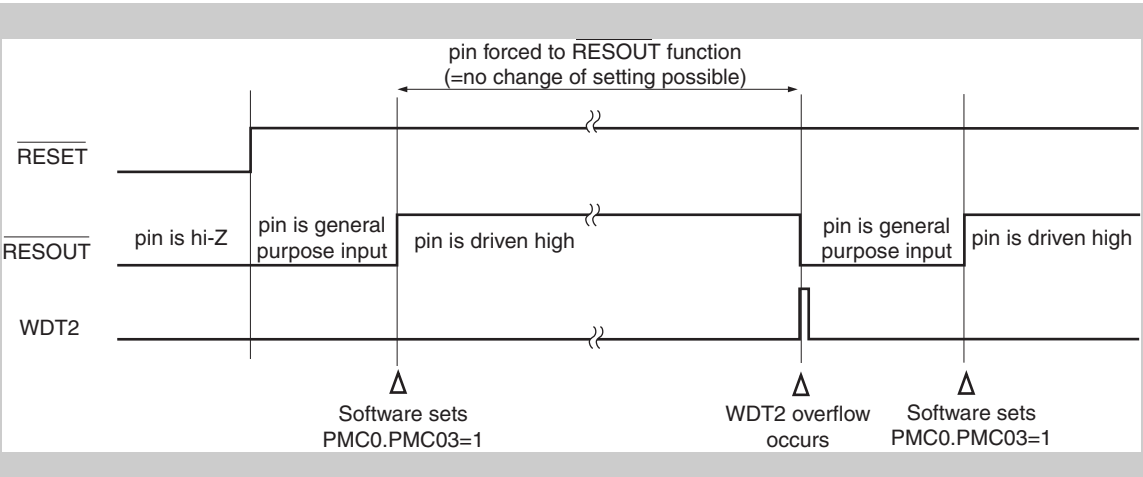
This function can inform the system when a reset occurs. The $\overline{\text{RESOUT}}$ pin drives low level during and after system reset. The software application can set the $\overline{\text{RESOUT}}$ pin to high level by normal GPIO port control function.

The $\overline{\text{RESOUT}}$ pin output drives a low level regardless of the reset source.

The $\overline{\text{RESOUT}}$ pin has a dedicated I/O buffer cell. It drives its output low when $\overline{\text{RESET}}$ pin is at a low level by global line control; for other reset sources, the port logic activates this function.

By connecting the pull-up resistor, the $\overline{\text{RESOUT}}$ pin is forced to output a high level that cannot be changed until either an internal or external reset occurs. *Figure 20-7* illustrates the timing of the $\overline{\text{RESOUT}}$ pin.

Figure 20-7 Timing and State of $\overline{\text{RESOUT}}$

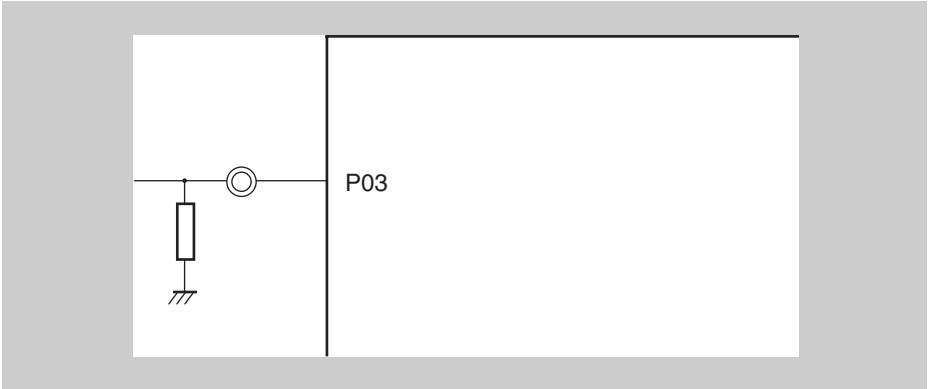


20.7.2 $\overline{\text{RESOUT}}$ function on 5V devices

There is no dedicated $\overline{\text{RESOUT}}$ function on the 5V devices. However, port P03 is provided without any alternative functions in order that $\overline{\text{RESOUT}}$ can easily and safely be emulated on this pin using application software outputting a high level to the P03 pin, mirroring the state of RESET to the output.

If P03 is used to emulate the $\overline{\text{RESOUT}}$ function, consider connecting a pull-down resistor externally to the pin as shown in the following figure. By connecting a pull-down resistor to the pin, the $\overline{\text{RESOUT}}$ signal line will be pulled low during RESET and kept low until the application software outputs a high level on the P03 pin.

Figure 20-8 Connection of P03 when emulating RESOUT



20.8 Reset Controller Registers

20.8.1 Reset Source Flag Register (RESSTAT)

The RESSTAT register indicates the source of a reset whether it is internal or external. Individual flags are set to indicate the source of the most recent reset.

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

Address FFFF FDA8$_H$

Initial Value $\overline{\text{RESET}}$ sets this register to 02$_H$. The default value differs if the reset stems from an internal source.

When a reset is executed by the WDTRES signal, RAM parity check, low-voltage detector (LVI), POC, or software reset, the reset flag of the corresponding register (RESWDT bit, RESLVI bit, RESPOC bit, or RESSW bit) is set to 1 (the other sources are held).

The contents of the register are cleared when any data is written to the register by software.

Note Data can only be written to this register following a specific sequence (refer to *Chapter 3, Section 3.8, Write Protected Registers on page 66*).

7	6	5	4	3	2	1	0
0	0	RESSW	RESWDT	0	RESLVI[a]	RESEXT	RESPOC[a]
R	R	R/W	R/W	R	R/W	R/W	R/W

a) Only available on 5V devices: µPD70F3470, µPD70F3471 and µPD70F3472

RESSW	Set by activation of the Software Reset
0	Normal operation
1	Reset occurred

RESWDT	Set by activation of the Watchdog Timer Reset
0	Normal operation
1	Reset occurred

RESLVI	Set by activation of the LVI Reset
0	Normal operation
1	Reset occurred[a]

a) Only available on 5V devices: µPD70F3470, µPD70F3471 and µPD70F3472

RESEXT	External reset
0	Normal operation
1	• 3V devices: External reset occurred • 5V devices: External Reset was released after the Power-On Reset was released. Set at release of the External Reset.

RESPOC	Set if Power-On Reset was released after External Reset was released
0	Normal operation
1	Reset occurred[a]

a) Only available on 5V devices: µPD70F3470, µPD70F3471 and µPD70F3472

20.8.2 Software Reset Trigger Register (RESSWT)

The software reset trigger register allows the user to reset the microcontroller by software instruction. Writing any data to this register generates a software reset of the device.

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

Address FFFF FDAA$_H$

Initial Value Undefined

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

Note This register is a write protected register. It can only be written after a write access to the RESCMD register.

20.8.3 Reset Protection Register (RESCMD)

RESCMD protects registers that may have a significant influence on the application system from an inadvertent write access.

Writing to the Reset Protection Register with any data enables access to the protected registers with the next NPB bus cycle. In this way, the protected register can only be rewritten in a specified sequence.

Any write access to RESCMD must immediately be followed by an access to the protected register. Any access to another NPB register inside this module or any access to the protected registers without prior access to RESCMD is illegal and will set the RERR flag of RES register.

Table 20-6 Registers protected by RESCMD

Register name	SFR	For register details see
Software Reset Trigger Register	RESSWT	*Section 20.8.2, Software Reset Trigger Register (RESSWT)* on page 605
On Chip Debug Mode Control Register	OCDM	*Section 20.8.5, On-Chip Debug Mode Register (OCDM)* on page 607

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

Address FFFF FDAC$_H$

Initial Value Undefined

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

Note The read value of the RESSWT register will always be 00$_H$.

20.8.4 Reset Status Register (RES)

This register indicates the status of a write instruction to a protected register. Any write access to RESCMD must immediately be followed by an access to a protected register. Any access to another NPB register or any access to the protected registers without prior access to RESCMD is illegal and will set the RERR flag of the RES register.

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

Address FFFF FDAE$_H$

Initial Value 00$_H$

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	RERR
R	R	R	R	R	R	R	R/W

Table 20-1

RERR	Reset Error
0	Write access was successful
1	Write access failed

20.8.5 On-Chip Debug Mode Register (OCDM)

The OCDM register enables the On Chip Debug Mode. It can be set by releasing the external Reset after the Power-On Reset is released or by software. It will be reset either by setting the external Reset and the Power-On Reset simultaneously, or by software.

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

Address FFFF F9FC$_H$

Initial Value The value after reset depends on the device and the reset source. See the table below.

Reset release condition		OCDM0 reset value
3V devices	External reset release	1
	Other reset release conditions	unchanged
5V devices	External reset release	1
	POC reset is released after the external reset	0
	Other reset release conditions	unchanged

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	OCDM0
R	R	R	R	R	R	R	R/W

Note This register is a write protected register. It can only be written after an access to the RESCMD register. See *Section 20.8.3, Reset Protection Register (RESCMD)*.

OCDM0	On-Chip Debug Mode Enable
0	Normal operation, DRSTZ doesn't influence the device operation.
1	On-Chip Debug Mode can be enabled by DRSTZ = '1'

Chapter 21 Low-Voltage Indicator

The 5V versions of the V850E/Rx3 microcontroller (µPD79F3470, µPD79F3471, and µPD79F3472) include a programmable low-voltage detector macro that helps to increase the reliability of system designs for automotive applications. The low-voltage detector continuously monitors and compares the supply voltage (V_{DD}) to an internal reference voltage (V_{LVI}), and generates an internal interrupt or internal reset signal when $V_{DD} < V_{LVI}$.

21.1 Low-Voltage Indicator Functions

The low-voltage indicator (LVI) has the following functions:

- Software selectable reference voltage level.
- Software selectable operation mode: Interrupt or Reset.
- The low-voltage indicator can operate in STOP mode.
- Operation can be stopped by software.

If the low-voltage indicator is used to generate a reset signal, bit 0 (LVIRF) of the reset source flag register (RESF) is set to 1 when the reset signal is generated. Refer to *Chapter 20* RESET Function.

21.2 Configuration of Low-Voltage Indicator

Figure 21-1 shows the block diagram of the low-voltage indicator.

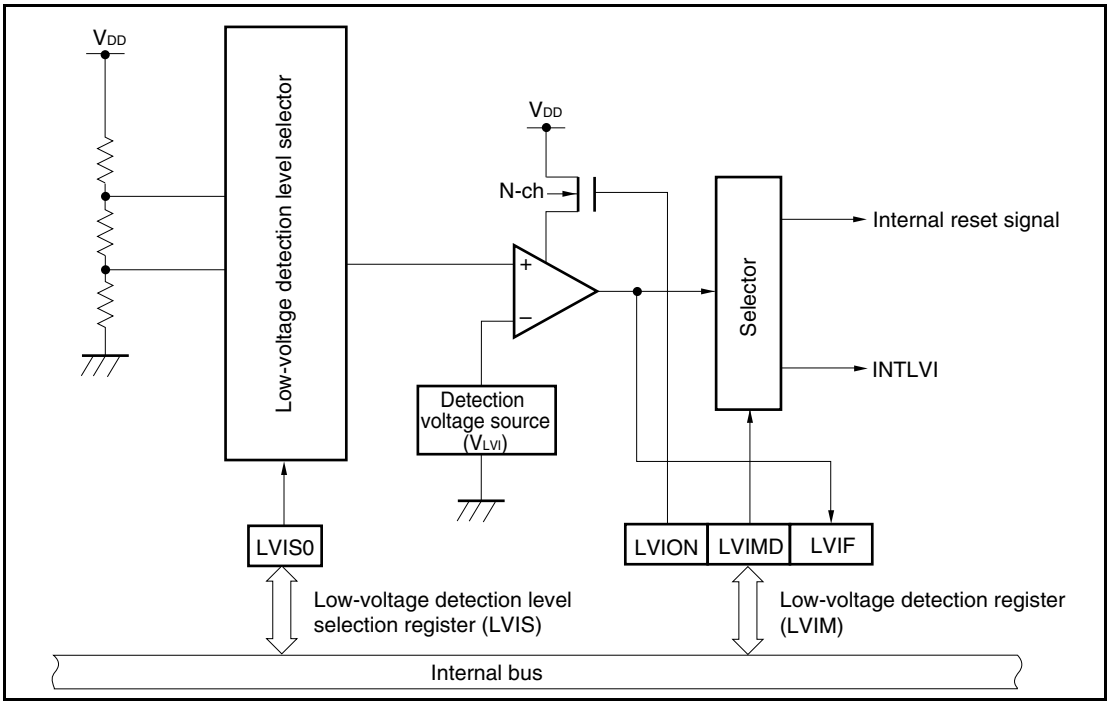


Figure 21-1 Block Diagram of Low-Voltage Indicator

21.3 Low-Voltage Indicator Control Registers

The low-voltage indicator is controlled by the following registers.

- Low-Voltage Indicator Mode register (LVIM)
- Low-Voltage Indicator Level Selection register (LVIS)
- Low-Voltage Indicator Protection register (PHCMDLVI)

21.3.1 Low-Voltage Indicator Mode Register (LVIM)

The LVIM register is a write-protected register (see *Section 3.8.1, Write protection control register* on page 67). It enables or disables low-voltage detection and sets the operation mode of the low-voltage indicator.

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

Address FFFF F890$_H$

Initial Value 00$_H$

7	6	5	4	3	2	1	0
LVION	0	0	0	0	0	LVIMD	LVIF
R/W	R	R	R	R	R	R/W	R

LVION	Enables or disables low-voltage detection operation
0	Disables operation
1	Enables operation

Table 21-1

LVIMD	Low-voltage detection operation mode selection
0	Generates interrupt request signal INTLVI when supply voltage (V_{DD}) < internal reference voltage (V_{LVI})
1	Generates internal reset signal LVIRES when supply voltage (V_{DD}) < internal reference voltage (V_{LVI})

Table 21-2

LVIF	Low-voltage detection flag
0	When supply voltage (V_{DD}) > reference voltage (V_{LVI}), or when operation is disabled
1	When supply voltage (V_{DD}) < reference voltage (V_{LVI})

Cautions

- After setting the LVION bit to 1, wait for 0.2ms before checking the voltage using the LVIF bit.
- The value of LVIF flag is output as the interrupt request signal INTLVI when LVION bit = 1 and LVIMD bit = 0.
- The LVIF bit is read-only.
- To stop LVI when using 8-bit manipulation instruction, write 00$_H$ to LVIM.

- Always write 0 to bits 6-2.

21.3.2 Low-voltage Indicator Level Selection Register (LVIS)

The LVIS register selects the low-voltage level to be detected.

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

Address FFFF F891$_H$

Initial Value 00$_H$

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	LVIS0
R	R	R	R	R	R	R	R/W

Table 21-3

LVIS0	Detection level
0	V_{LVI0} (4.4 V ±0.2 V)
1	V_{LVI1} (4.2 V ±0.2 V)

21.3.3 LVI Protection Register (PHCMDLVI)

The PHCMDLVI is a protection register used to prevent unintended write operations to LVIM. It is a write-only register. Any value written to PHCMDLVI is ignored.

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

Table 21-4

PHCMDLVI	Detection level
0	V_{LVI0} (4.4 V ±0.2 V)
1	V_{LVI1} (4.2 V ±0.2 V)

Note Reading PHCMDLVI results to 0x00. Do not open read access of PHCMDLVI to customers.

For writing to LVIM following sequence has to be respected:

1. Write any value to PHCMDLVI.
2. Write value to LVIM. Read-Modify-Write operation is possible.
3. Check new contents of LVIM.

Notes 1. One or more NPB read cycles in between writing to PHCMDLVI and writing to LVIM do not violate the special sequence, i.e. writing to LVIM is permitted.

2. One or more NPB write cycles in between writing to PHCMDLVI and writing to LVIM violate the special sequence, i.e. writing to LVIM is prohibited.
3. If LVIM = LVIMD = ’1’, writing to LVIM or LVIS is prohibited.
4. On 16 bit write access to LVIM_LVIS register, write protection by PHCMD protects only write access to LVIM register.

21.4 Low-Voltage Indicator Operation

Depending on the setting of the LVIMD bit, an interrupt signal (INTLVI) or an internal reset signal is generated.

The following sections explain how to specify each operation. Timing charts are included.

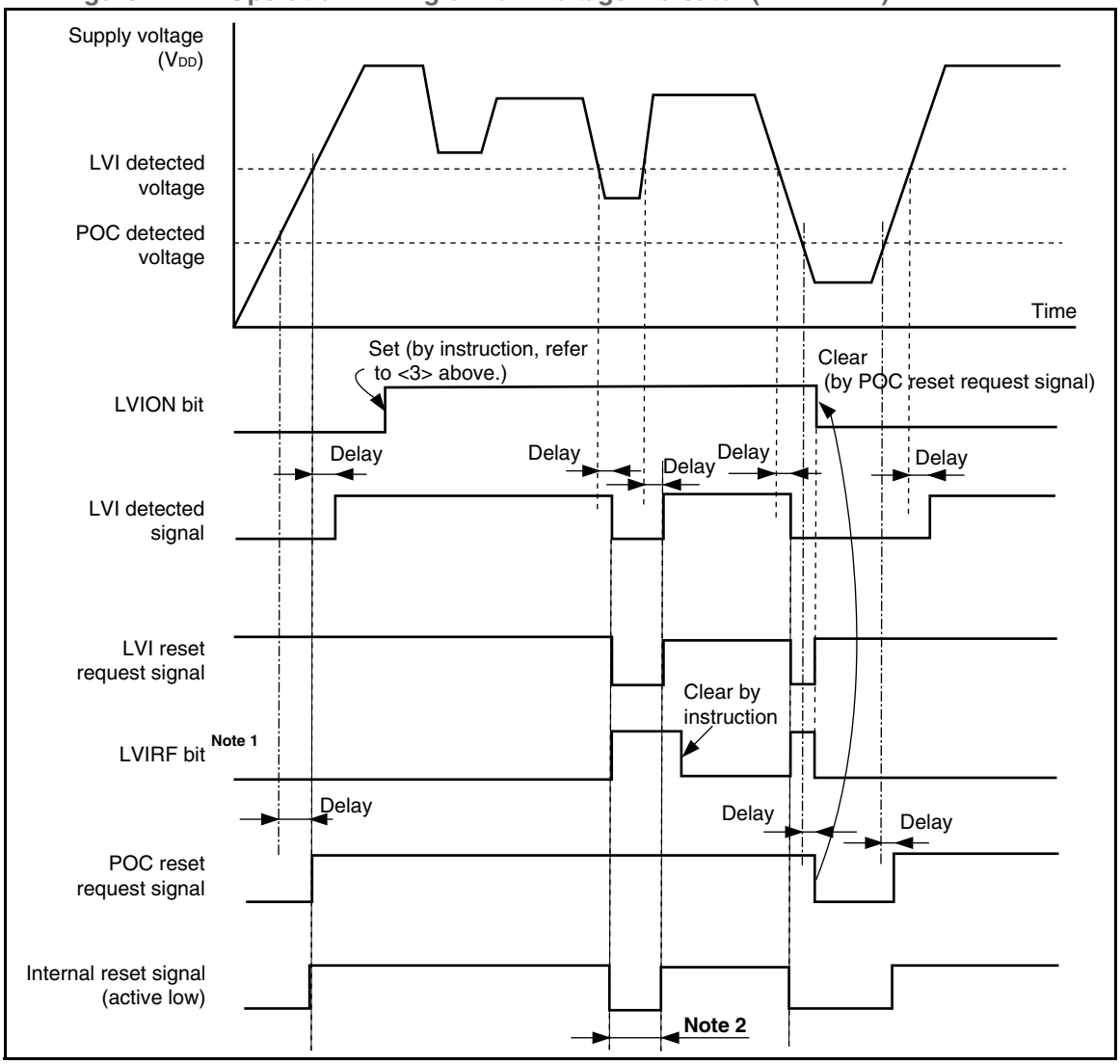
21.4.1 Use for Internal Reset Signal

(a) To start operation:

1. Mask the LVI interrupt.
2. Select the reference voltage level to be used by setting the LVIS0 bit accordingly.
3. Set the LVION bit to 1 (to enable operation).
4. Insert a wait cycle of 0.2 ms (TYP) (target value) or more by software.
5. Use the LVIF bit to check if the supply voltage > reference voltage.
6. Set the LVIMD bit to 1 (to enable Reset mode operation).

Caution If LVIMD is set to 1, the contents of the LVIM and LVIS registers cannot be changed until a reset request other than LVI is generated.

Figure 21-2 Operation Timing of Low-Voltage Indicator (LVIMD = 1)



Notes

1. The LVIRF bit is bit 0 of the reset source flag register (RESF). For details of RESF, refer to *Section 20.8.1, Reset Source Flag Register (RESSTAT)* on page 604.
2. During the period when the supply voltage is equal to or below the LVI trip voltage, the internal reset signal is retained (internal reset state).

21.4.2 Use for Interrupt

(a) To start operation:

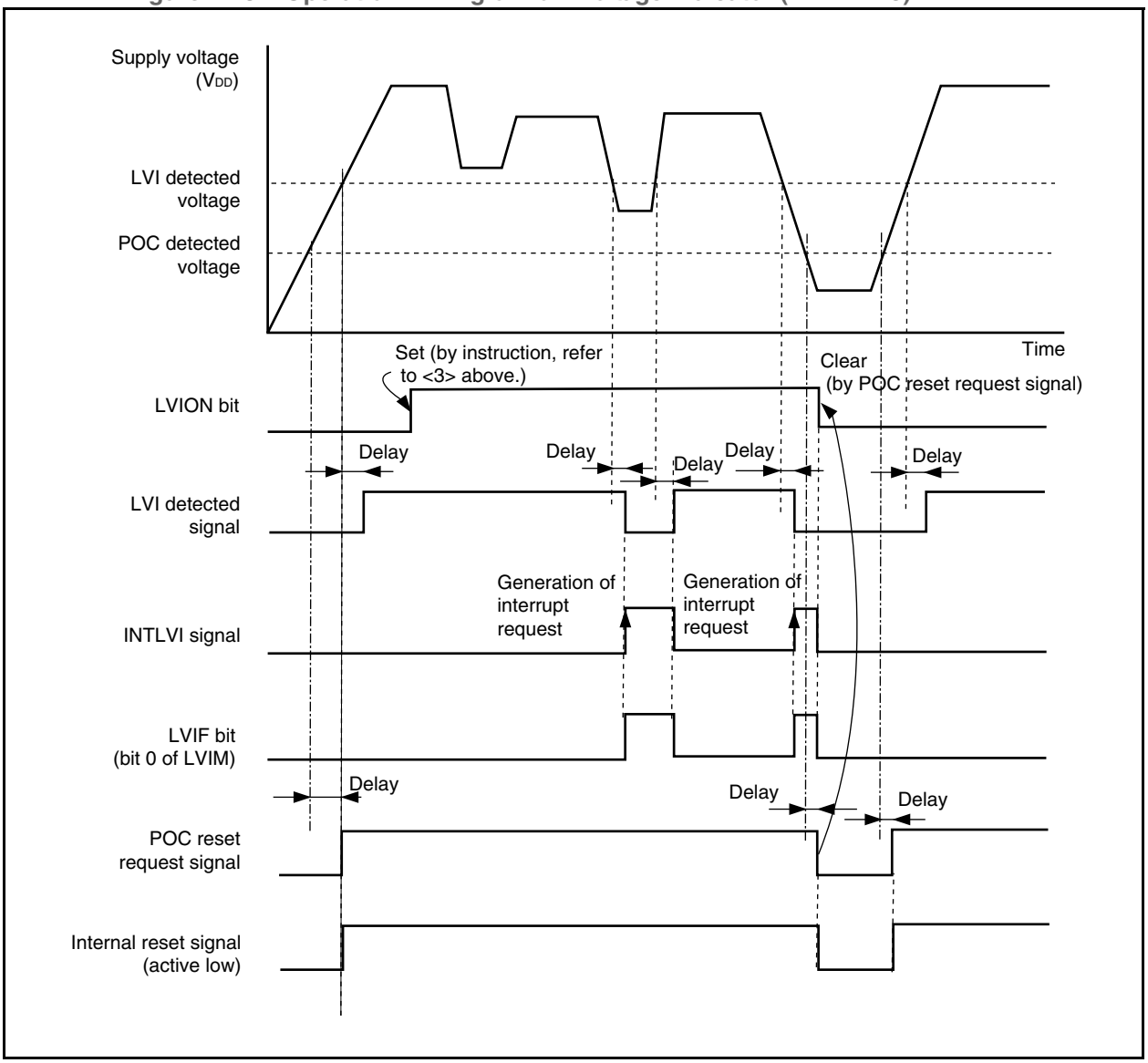
1. Mask the LVI interrupt.
2. Select the reference voltage level to be used by setting the LVIS0 bit accordingly.
3. Set the LVION bit to 1 (to enable operation).
4. Insert a wait cycle of 0.2 ms (TYP) (target value) or more by software.
5. Use the LVIF bit to check if the supply voltage > reference voltage.

6. Clear the interrupt request flag of LVI.
7. Unmask the LVI interrupt.

(b) To stop operation:

Set the LVION bit to 0.

Figure 21-3 Operation Timing of Low-Voltage Indicator (LVIMD = 0)



Chapter 22 On-Chip Debug Unit

The microcontroller includes an on-chip debug unit. By connecting an N-Wire emulator, on-chip debugging can be executed.

22.1 Functional Outline

22.1.1 Debug functions

(1) Debug interface

Communication with the host machine is established by using the $\overline{\text{DRST}}$, DCK, DMS, DDI, and DDO signals via an on-chip debug emulator. The communication specifications of N-Wire are used for the interface.

(2) On-chip debug

On-chip debugging can be executed by preparing wiring and a connector for on-chip debugging on the target system. An on-chip debug emulator is used to connect the host PC to the on-chip debug unit.

(3) Forced reset function

The microcontroller can be forcibly reset.

(4) Break reset function

The CPU can be started in the debug mode immediately after reset of the CPU is released.

(5) Forced break function

Execution of the user program can be forcibly aborted.

(6) Hardware break function

Two breakpoints for instruction and data access can be used. The instruction breakpoint can abort program execution at any address. The access breakpoint can abort program execution by data access to any address.

(7) Software break function

Up to eight software breakpoints can be set in the internal flash memory area. The number of software breakpoints that can be set in the RAM area differs depending on the debugger to be used.

(8) Debug monitor function

A memory space for debugging that is different from the user memory space is used during debugging (background monitor mode). The user program can be executed starting from any address.

While execution of the user program is aborted, the user resources (such as memory and I/O) can be read and written, and the user program can be downloaded.

(9) Mask function

Each of the following signals can be masked. That means these signals will not be effective during debugging.

The correspondence with the mask functions of the debugger (ID850NWC) for the N-Wire emulator (IE-V850E1-CD-NW) of NEC Electronics is shown below.

NMI0 mask function: Flash double-bit error, RAM double-bit error

(10) Peripheral macro operation/stop selection function during break

Depending on the debugger to be used, certain peripheral macros can be configured to continue or to stop operation upon a breakpoint hit.

- Functions that are always stopped during break
  - Watchdog Timer
- Functions that can operate or be stopped during break (however, each function cannot be selected individually)
  - Timers AA
  - Timers AB
  - Timer M
- Peripheral functions that continue operating during break (functions that cannot be stopped)
  - Peripheral functions other than above (e.g. CAN, UARTD, ...)

22.1.2 Security function

This microcontroller has an N-Wire security function, which requires the user to input an ID code upon start of the debugger. The ID code is compared to a predefined ID code, written in advance to the internal flash memory by an external flash programmer. This function prevents unauthorized persons to operate the microcontroller in N-Wire debug mode and to read the internal flash memory area.

The ID code in the internal flash memory can only be written by an external flash programmer. It cannot be changed in self-programming mode and also not in N-Wire debugging mode.

ID code

Be sure to write an ID code when writing a program to the internal flash memory.

The area of the ID code is 10 bytes wide and in the range of addresses $0000\ 0070_H$ to $0000\ 0079_H$.

The ID code when the memory is erased is shown below.

Address	ID code
$0000\ 0079_H$	FF_H
$0000\ 0078_H$	FF_H
$0000\ 0077_H$	FF_H
$0000\ 0076_H$	FF_H
$0000\ 0075_H$	FF_H
$0000\ 0074_H$	FF_H
$0000\ 0073_H$	FF_H

Address	ID code
0000 0072_H	FF_H
0000 0071_H	FF_H
0000 0070_H	FF_H

Security bit Bit 7 of address 0000 0079_H enables or disables use of the N-Wire emulator.

- Bit 7 of address 0000 0079_H

  0: disabledN-Wire emulator cannot connect to the on-chip debug unit.

  1: enabledN-Wire emulator can connect to the on-chip debug unit if the 10-byte ID code input matches the ID code stored in the flash memory

This security bit can only be modified by programming the flash memory via an external flash programmer. It is not possible to modify the security bit in self-programming mode, and therefore also not in N-Wire debugging mode.

After reset, the entire ID code area is set to FF_H. This means that

- N-Wire debugging is generally enabled
- the ID code is FF_H for all ID code bytes

Consequently, controller access is possible without any restriction.

Caution If access via the N-Wire interface should be disabled "block erase disabled" should be configured as well. Otherwise the flash memory blocks containing the ID code could be erased and N-Wire access could be enabled.

22.2 Controlling the N-Wire Interface

The N-wire interface pins DRST, DDI, DDO, DCK, DMS are dedicated (i.e. not shared with GPIO functions); see Table 33-1 for their function description. Note that N-Wire debugging must be generally permitted by the security bit in the ID code region (*0x0000'0079[bit7]=1) of the Flash memory.

Table 22-1 N-wire interface pins

Pin	Description	OCDM = 0	OCDM = 1	
			$\overline{DRST}$ = 0	$\overline{DRST}$ = 1
DRST	N-wire RCU reset	pull-down	pull-down + input	
DDI	N-wire debug data in	pull-up	pull-up + input	input
DDO	N-wire debug data out	pull-up	pull-up	output
DCK	N-wire interface clock	pull-up	pull-up + input[a]	
DMS	N-wire mode	pull-up	pull-up + input	input

a) Pull-up should always be enabled for improved security. The device stays in Reset if DRSTZ is accidently set to a high level.

22.2.1 OCDM - On-chip debug mode register

The OCDM register is used to select the normal operation mode or on-chip debug mode.

Writing to this register is protected by a special sequence of instructions. Please refer to section *20.8.3“Reset Protection Register (RESCMD)” on page 606* for details.

Access The register can be read or written in 8-bit and 1-bit units.

Address FFF F9FC$_H$

Initial Value 01_H if normal external Reset occurs
00_H if POC Reset occurs

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	OCDM0
R	R	R	R	R	R	R	R/W

OCDM0	On-chip debug mode enable bit
0	On-chip debug mode is disabled
1	On-chip debug mode is enabled

The reset value of OCDM.OCDM0 depends on the reset source.

The $\overline{\text{DRST}}$ signal depicts the N-wire interface reset signal. If $\overline{\text{DRST}}$ = 0, the on-chip debug unit is kept in reset state and does not affect normal controller operation. $\overline{\text{DRST}}$ is driven by the debugger, if one is connected. The debugger may start communication with the controller by setting $\overline{\text{DRST}}$ = 1.

$\overline{\text{DRST}}$ has an internal permanent pull-down resistor to make sure that the device can start even if the emulator is not connected.

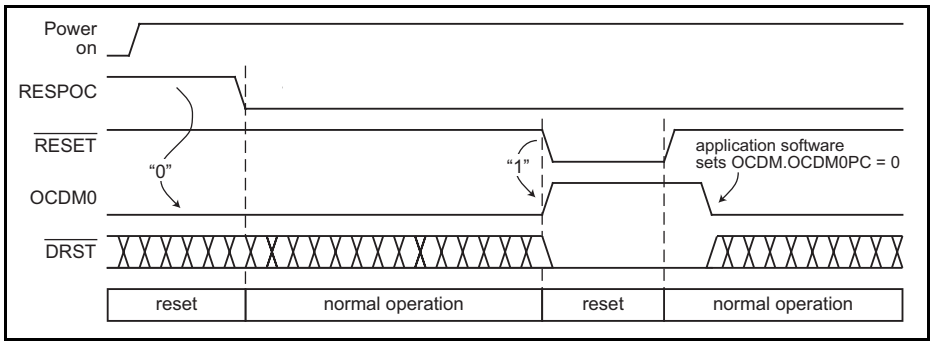
It is assumed that the same initialization S/W is performed, independently of the reset source.

22.3 N-Wire Enabling Methods

22.3.1 Starting Normal Operation after RESET and RESPOC

For “normal operation,” pins P97-P911 must be available as port pins. Therefore, the software has to set OCDM.OCDM0 = 0 to make the pins available as port pins after RESET.

Figure 22-1 Start Normal Operation without N-Wire Activation



22.3.2 Starting Debugger after RESPOC and $\overline{\text{RESET}}$

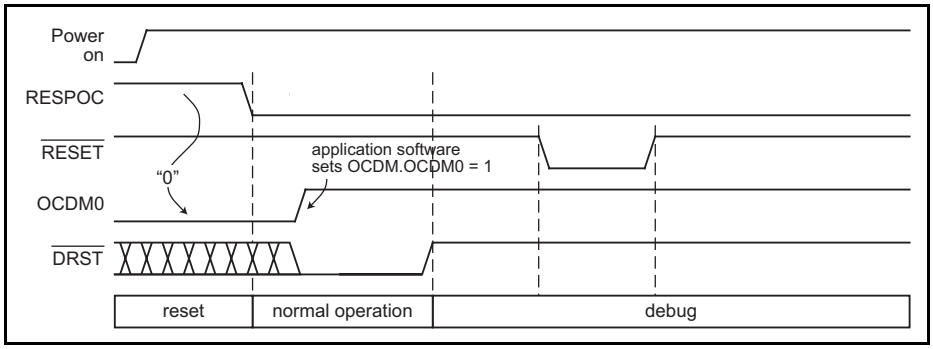
The software has to set OCDM.OCDM0 = 1 to enable the N-Wire interface after RESPOC occurs. This setting allows the debugger to establish communication with the controller by setting the $\overline{\text{DRST}}$ pin to “high” and assuming control of the CPU.

When the debugger starts operation, the entire controller is reset, i.e. all registers are set to their default states and the CPU's program counter is set to the reset vector $0000\ 0000_H$.

Notes

- After RESPOC, CPU instructions cannot be debugged before the software sets OCDM.OCDM0 = 1. To restart the user’s program from beginning under the debugger’s control, apply an external $\overline{\text{RESET}}$ after the debugger starts as shown in *Figure 22-2*. This will cause the program to restart. However, the status of the controller might not be the same as immediately after RESPOC, since the internal RAM may have already been initialized when the external $\overline{\text{RESET}}$ was applied.
- To use the on-chip debug function in a product with a power-on clear function, input a low level to the $\overline{\text{RESET}}$ input pin for 2000 ms or longer after power application.

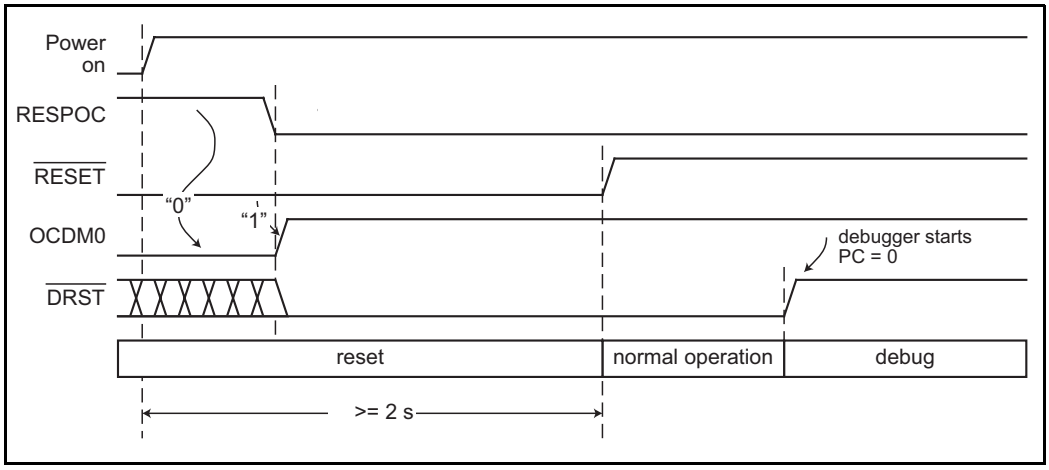
Figure 22-2 Start Operation with N-Wire Activation



22.3.3 Activate N-Wire by $\overline{\text{RESET}}$

The N-Wire interface can be activated after power-up by keeping $\overline{\text{RESET}}$ active for at least 2 seconds after RESPOC release. This sets OCDM.OCDM0 = 1, enabling the N-Wire interface and eliminating the need for software to perform OCDM.OCDM0 = 1.

Figure 22-3 N-Wire activation by $\overline{\text{RESET}}$ pin



22.4 Connection to N-Wire Emulator

To connect the N-Wire emulator, a connector for emulator connection and a connection circuit must be mounted on the target system.

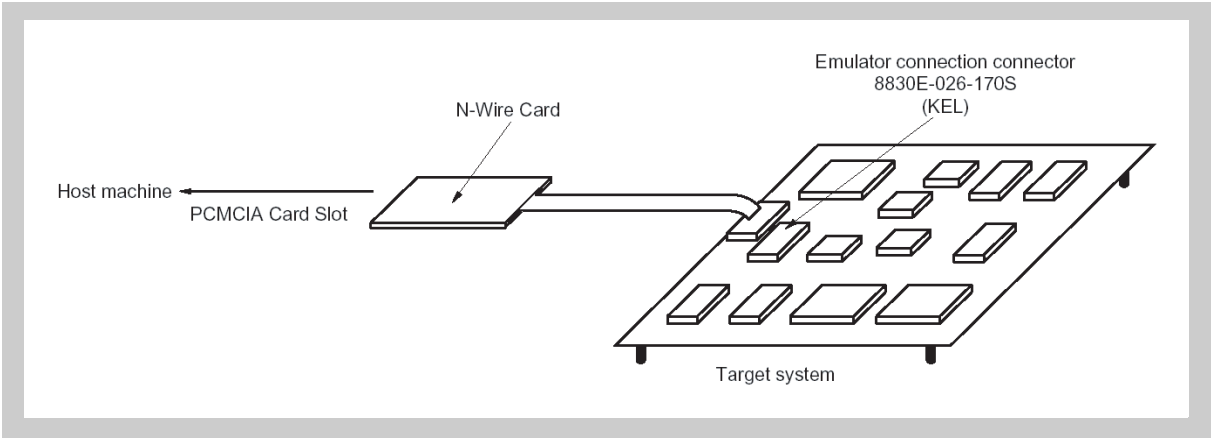
As a connector example, the KEL connector is described in more detail. Other connectors, such as the MICTOR connector (product name: 2-767004-2, Tyco Electronics AMP K.K.), are available as well. For the mechanical and electrical specification of these connectors, refer to their user's manuals.

22.4.1 KEL connector

KEL connector product names:

- 8830E-026-170S (KEL): straight type
- 8830E-026-170L (KEL): right-angle type

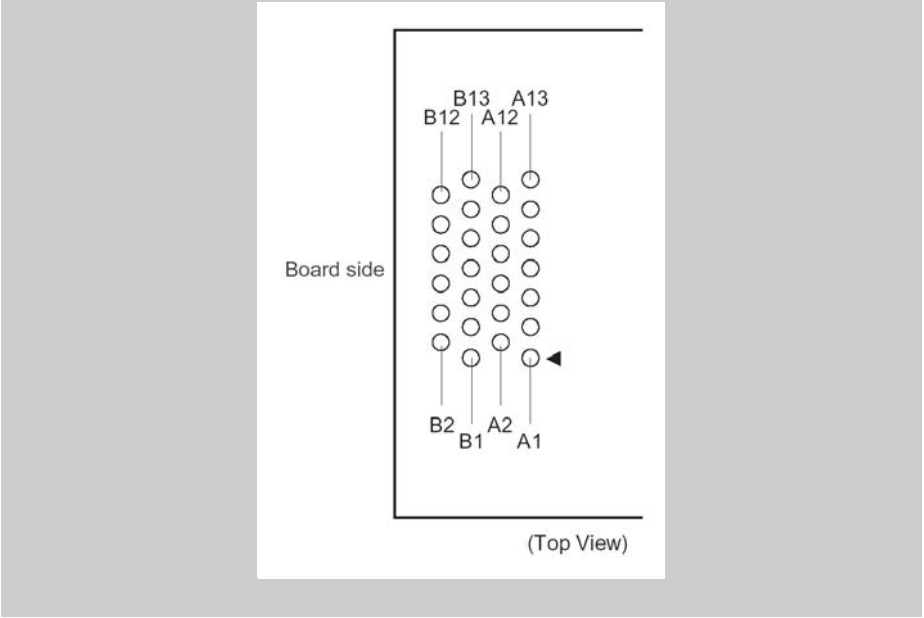
Figure 22-4 Connection to N-Wire emulator (NEC Electronics IE-V850E1-CD-NW: N-Wire Card)



(1) Pin configuration

Figure 22-5 shows the pin configuration of the connector for emulator connection (target system side), and *Table 22-2* on page 622 shows the pin functions.

Figure 22-5 Pin configuration of connector for emulator connection (target system side)



Caution Evaluate the dimensions of the connector when actually mounting the connector on the target board.

(2) Pin functions

The following table shows the pin functions of the connector for emulator connection (target system side). “I/O” indicates the direction viewed from the device.

Table 22-2 Pin functions of connector for emulator connection (target system side)

Pin no.	Pin name	I/O	Pin function
A1	(Reserved 1)	–	(Connect to GND)
A2	(Reserved 2)	–	(Connect to GND)
A3	(Reserved 3)	–	(Connect to GND)
A4	(Reserved 4)	–	(Connect to GND)
A5	(Reserved 5)	–	(Connect to GND)
A6	(Reserved 6)	–	(Connect to GND)
A7	DDI	Input	Data input for N-Wire interface
A8	DCK	Input	Clock input for N-Wire interface
A9	DMS	Input	Transfer mode select input for N-Wire interface
A10	DDO	Output	Data output for N-Wire interface
A11	$\overline{\text{DRST}}$	Input	On-chip debug unit reset input
A12	$\overline{\text{RESET}}$	Input	Reset input.
A13	FLMD0	Input	Control signal for flash download (flash memory versions only)
B1	GND	–	–
B2	GND	–	–
B3	GND	–	–
B4	GND	–	–
B5	GND	–	–
B6	GND	–	–
B7	GND	–	–
B8	GND	–	–
B9	GND	–	–
B10	GND	–	–
B11	(Reserved 8)	–	(Connect to GND)
B12	(Reserved 9)	–	(Connect to GND)
B13	V_{DD}	–	3.3 V input (for monitoring power supply to target)

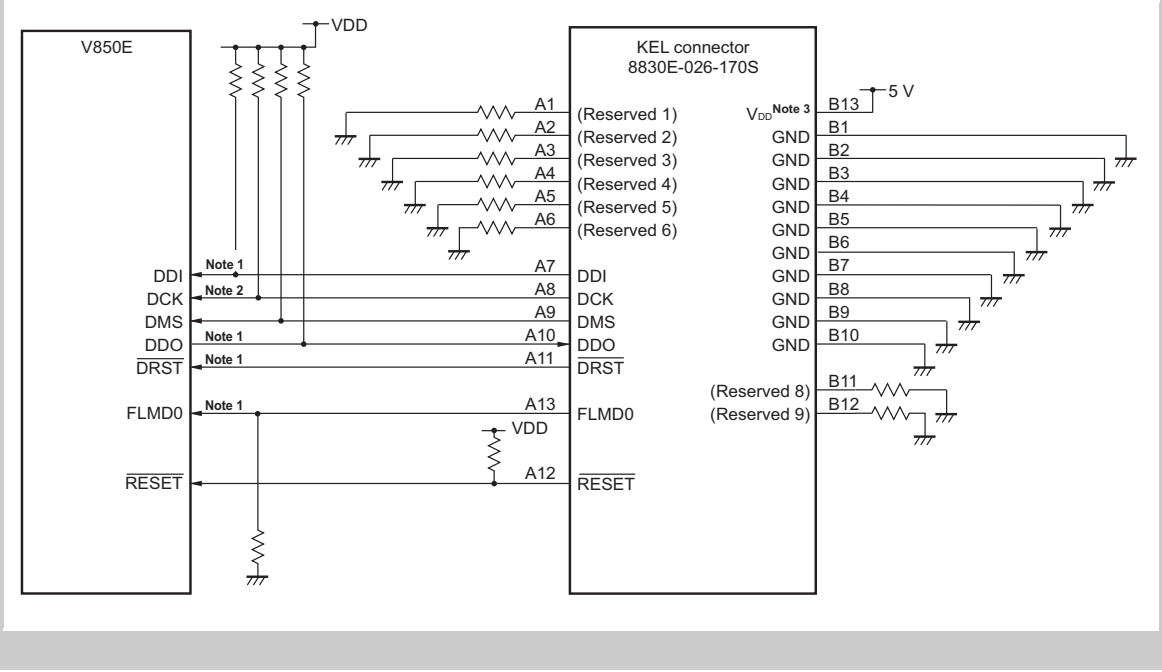
Cautions

1. The connection of the pins not supported by the microcontroller is dependent upon the emulator to be used.
2. The pattern of the target board must satisfy the following conditions.
   - The pattern length must be 100 mm or less.
   - The clock signal must be shielded by GND.

(3) Example of recommended circuit

An example of the recommended circuit of the connector for emulator connection (target system side) is shown below.

Figure 22-6 Example of recommended emulator connection circuit



Note
1. The pattern length must be 100 mm or less.
2. Shield the DCK signal by enclosing it with GND.
3. This pin is used to detect power to the target board. Connect the voltage of the N-Wire interface to this pin.

Caution The N-Wire emulator pins FLMD0 and $\overline{\text{RESET}}$ may not support a 5 V interface and may require a level shifter. Refer to the user's manual of the emulator to be used.

22.5 Restrictions and Cautions on On-Chip Debug Function

- Do not mount a device that was used for debugging on a mass-produced product (this is because the flash memory was rewritten during debugging and the number of rewrites of the flash memory cannot be guaranteed).
- If a reset signal (reset input from the target system or reset by an internal reset source) is input during RUN (program execution), the break function may malfunction.
- Even if reset is masked by using a mask function, the I/O buffer (port pin, etc.) is reset when a pin reset signal is input.
- With a debugger that can set software breakpoints in the internal flash memory, the breakpoints temporarily become invalid when pin reset or internal reset is effected. The breakpoints become valid again if a break such as a hardware break or forced break is executed. Until then, no software break occurs.
- The $\overline{\text{RESET}}$ signal input is masked during a break.
- The on-chip debugging unit uses the exception vector address 60_H for software breakpoint (DBTRAP, refer to *Section Chapter 5, Interrupt Controller (INTC)* on page 81). Thus the debugger takes over control when one of the following exceptions occur:
  – debug trap (DBTRAP)
  – illegal op-code detection (ILGOP)

  The debugger executes its own exception handler. Therefore, the user's exception handler at address 60_H will not be executed.
- When executing on-chip debugging, pin reset must be input to set the OCDM0 bit of the OCDM register to 1. For details, refer to
- When the break command is started in on-chip debug (OCD) mode and the application software accesses to the UARTD/CSIF/CAN peripheral I/O registers, CSIF, UARTD and CAN do not operate normally if on-chip debugging is restarted without executing reset.

Caution If the flash memory is programmed during a debug session and the options bytes have been changed, a target reset command has to be issued in order to make the new option byte settings effective.

Appendix A Special Function Registers

The following tables list all registers that are accessed via the NPB (NEC peripheral bus). The registers are called “special function registers” (SFR).

Table A-1 lists all CAN special function registers. The addresses are given as offsets to the programmable peripheral base address (refer to *Section 17.5.1, CAN module register and message buffer addresses* on page 428 of this manual).

The tables list all registers and do not distinguish between the different derivatives.

A.1 CAN Registers

The CAN registers are accessible via the programmable peripheral area.

Table A-1 CAN special function registers (1/3)

Address offset	Register name	SFR	1	8	16	32	Reset Value
0x000	CAN0 global control register	C0GMCTRL	-	-	R/W	-	0x0000
0x000	CAN0 global control register Low Byte	C0GMCTRLL	-	R/W	-	-	0x00
0x001	CAN0 global control register High Byte	C0GMCTRLH	-	R/W	-	-	0x00
0x002	CAN0 global clock selection register	C0GMCS	-	R/W	-	-	0x00
0x006	CAN0 global automatic block transmission register	C0GMABT	-	-	R/W	-	0x0000
0x006	CAN0 global automatic block transmission register Low Byte	C0GMABTL	-	R/W	-	-	0x00
0x007	CAN0 global automatic block transmission register High Byte	C0GMABTH	-	R/W	-	-	0x00
0x008	CAN0 global automatic block transmission delay register	C0GMABTD	-	R/W	-	-	0x00
0x040	CAN0 module mask 1 register	C0MASK1L	-	-	R/W	-	0x0000
0x042		C0MASK1H	-	-	R/W	-	0x0000
0x044	CAN0 module mask 2 register	C0MASK2L	-	-	R/W	-	0x0000
0x046		C0MASK2H	-	-	R/W	-	0x0000
0x048	CAN0 module mask 3 register	C0MASK3L	-	-	R/W	-	0x0000
0x04A		C0MASK3H	-	-	R/W	-	0x0000
0x04C	CAN0 module mask 4 register	C0MASK4L	-	-	R/W	-	0x0000
0x04E		C0MASK4H	-	-	R/W	-	0x0000
0x050	CAN0 module control register	C0CTRL	-	-	R/W	-	0x0000
0x052	CAN0 module last error code register	C0LEC	-	R/W	-	-	0x00
0x053	CAN0 module information register	C0INFO	-	R	-	-	0x00
0x054	CAN0 module error counter register	C0ERC	-	-	R	-	0x0000
0x056	CAN0 module interrupt enable register	C0IE	-	-	R/W	-	0x0000
0x056	CAN0 module interrupt enable register	C0IEL	-	R/W	-	-	0x00

Table A-1 CAN special function registers (2/3)

Address offset	Register name	SFR	1	8	16	32	Reset Value
0x057	CAN0 module interrupt enable register	C0IEH	-	R/W	-	-	0x00
0x058	CAN0 module interrupt status register	C0INTS	-	-	R/W	-	
0x05A	CAN0 module bit-rate prescaler register	C0BRP	-	R/W	-	-	0x00
0x05C	CAN0 module bit-rate register	C0BTR	-	-	R/W	-	0x0000
0x05E	CAN0 module last in-pointer register	C0LIPT	-	R	-	-	0x00
0x060	CAN0 module receive history list register	C0RGPT	-	-	R/W	-	0x0000
0x062	CAN0 module last out-pointer register	C0LOPT	-	R	-	-	0x00
0x064	CAN0 module transmit history list register	C0TGPT	-	-	R/W	-	0x0000
0x066	CAN0 module time stamp register	C0TS	-	-	R/W	-	
0x100 to 0x4EF	CAN0 Message Buffer registers, see *Table 17-22* on page 431						
0x800	CAN1 global control register	C1GMCTRL	-	-	R/W	-	0x0000
0x800	CAN1 global control register Low Byte	C1GMCTRLL	-	R/W	-	-	0x00
0x801	CAN1 global control register High Byte	C1GMCTRLH	-	R/W	-	-	0x00
0x802	CAN1 global clock selection register	C1GMCS	-	R/W	-	-	0x00
0x806	CAN1 global automatic block transmission register	C1GMABT	-	-	R/W	-	0x0000
0x806	CAN1 global automatic block transmission register Low Byte	C1GMABTL	-	R/W	-	-	0x00
0x807	CAN1 global automatic block transmission register High Byte	C1GMABTH	-	R/W	-	-	0x00
0x808	CAN1 global automatic block transmission delay register	C1GMABTD	-	R/W	-	-	0x00
0x840	CAN1 module mask 1 register	C1MASK1L	-	-	R/W	-	0x0000
0x842		C1MASK1H	-	-	R/W	-	0x0000
0x844	CAN1 module mask 2 register	C1MASK2L	-	-	R/W	-	0x0000
0x846		C1MASK2H	-	-	R/W	-	0x0000
0x848	CAN1 module mask 3 register	C1MASK3L	-	-	R/W	-	0x0000
0x84A		C1MASK3H	-	-	R/W	-	0x0000
0x84C	CAN1 module mask 4 register	C1MASK4L	-	-	R/W	-	0x0000
0x84E		C1MASK4H	-	-	R/W	-	0x0000
0x850	CAN1 module control register	C1CTRL	-	-	R/W	-	0x0000
0x852	CAN1 module last error code register	C1LEC	-	R/W	-	-	0x00
0x853	CAN1 module information register	C1INFO	-	R	-	-	0x00
0x854	CAN1 module error counter register	C1ERC	-	-	R	-	0x0000
0x856	CAN1 module interrupt enable register Low Byte	C1IEL	-	R/W	-	-	0x00
0x856	CAN1 module interrupt enable register	C1IE	-	-	R/W	-	0x0000
0x857	CAN1 module interrupt enable register High Byte	C1IEH	-	R/W	-	-	0x00
0x858	CAN1 module interrupt status register	C1INTS	-	-	R/W	-	0x0000
0x85A	CAN1 module bit-rate prescaler register	C1BRP	-	R/W	-	-	0x00
0x85C	CAN1 module bit-rate register	C1BTR	-	-	R/W	-	0x0000
0x85E	CAN1 module last in-pointer register	C1LIPT	-	R	-	-	0x00
0x860	CAN1 module receive history list register	C1RGPT	-	-	R/W	-	0x0000

Table A-1 CAN special function registers (3/3)

Address offset	Register name	SFR	1	8	16	32	Reset Value
0x862	CAN1 module last out-pointer register	C1LOPT	-	R	-	-	0x00
0x864	CAN1 module transmit history list register	C1TGPT	-	-	R/W	-	0x0000
0x866	CAN1 module time stamp register	C1TS	-	-	R/W	-	0x0000
0x900 to 0xCEF	CAN1 Message Buffer registers, see *Table 17-22* on page 431						

A.2 Other Special Function Registers

Table A-2 Other special function registers (1/10)

Address	Register name	SFR	1	8	16	32	Reset Value
0xFFFF F064	Peripheral I/O area select control register	BPC	-	-	R/W	-	0x0000
0xFFFF F06E	System wait control register	VSWC	R/W	R/W	-	-	0x77
0xFFFF F100	Interrupt mask control register 0	IMR0	-	-	R/W	-	0xFFFF
0xFFFF F100	Interrupt mask control register 0L	IMR0L	R/W	R/W	-	-	0xFF
0xFFFF F101	Interrupt mask control register 0H	IMR0H	R/W	R/W	-	-	0xFF
0xFFFF F102	Interrupt mask control register 1	IMR1	-	-	R/W	-	0xFFFF
0xFFFF F102	Interrupt mask control register 1L	IMR1L	R/W	R/W	-	-	0xFF
0xFFFF F103	Interrupt mask control register 1H	IMR1H	R/W	R/W	-	-	0xFF
0xFFFF F104	Interrupt mask control register 2	IMR2	-	-	R/W	-	0xFFFF
0xFFFF F104	Interrupt mask control register 2L	IMR2L	R/W	R/W	-	-	0xFF
0xFFFF F105	Interrupt mask control register 2H	IMR2H	R/W	R/W	-	-	0xFF
0xFFFF F106	Interrupt mask control register 3	IMR3	-	-	R/W	-	0xFFFF
0xFFFF F106	Interrupt mask control register 3L	IMR3L	R/W	R/W	-	-	0xFF
0xFFFF F107	Interrupt mask control register 3H	IMR3H	R/W	R/W	-	-	0xFF
0xFFFF F108	Interrupt mask control register 4	IMR4	-	-	R/W	-	0xFFFF
0xFFFF F108	Interrupt mask control register 4L	IMR4L	R/W	R/W	-	-	0xFF
0xFFFF F109	Interrupt mask control register 4H	IMR4H	R/W	R/W	-	-	0xFF
0xFFFF F110	Interrupt control register	ERRIC	R/W	R/W	-	-	0x47
0xFFFF F112	Interrupt control register	AD0IC	R/W	R/W	-	-	0x47
0xFFFF F114	Interrupt control register	LVIIC	R/W	R/W	-	-	0x47
0xFFFF F116	Interrupt control register	P0IC	R/W	R/W	-	-	0x47
0xFFFF F118	Interrupt control register	P1IC	R/W	R/W	-	-	0x47
0xFFFF F11A	Interrupt control register	P2IC	R/W	R/W	-	-	0x47
0xFFFF F11C	Interrupt control register	P3IC	R/W	R/W	-	-	0x47
0xFFFF F11E	Interrupt control register	P4IC	R/W	R/W	-	-	0x47
0xFFFF F120	Interrupt control register	P5IC	R/W	R/W	-	-	0x47
0xFFFF F122	Interrupt control register	P6IC	R/W	R/W	-	-	0x47
0xFFFF F124	Interrupt control register	P7IC	R/W	R/W	-	-	0x47
0xFFFF F126	Interrupt control register	CE0CIC	R/W	R/W	-	-	0x47
0xFFFF F128	Interrupt control register	CE0OFIC	R/W	R/W	-	-	0x47
0xFFFF F12A	Interrupt control register	CE1CIC	R/W	R/W	-	-	0x47
0xFFFF F12C	Interrupt control register	CE1OFIC	R/W	R/W	-	-	0x47
0xFFFF F12E	Interrupt control register	CF0RIC	R/W	R/W	-	-	0x47
0xFFFF F130	Interrupt control register	CF0TIC	R/W	R/W	-	-	0x47
0xFFFF F132	Interrupt control register	CF1RIC	R/W	R/W	-	-	0x47
0xFFFF F134	Interrupt control register	CF1TIC	R/W	R/W	-	-	0x47
0xFFFF F136	Interrupt control register	TMM0EQ0IC	R/W	R/W	-	-	0x47
0xFFFF F138	Interrupt control register	TAA0CC0IC	R/W	R/W	-	-	0x47
0xFFFF F13A	Interrupt control register	TAA0CC1IC	R/W	R/W	-	-	0x47
0xFFFF F13C	Interrupt control register	TAA0OVIC	R/W	R/W	-	-	0x47
0xFFFF F13E	Interrupt control register	TMM1EQ0IC	R/W	R/W	-	-	0x47

Table A-2 Other special function registers (2/10)

Address	Register name	SFR	1	8	16	32	Reset Value
0xFFFF F140	Interrupt control register	TAA1CC0IC	R/W	R/W	-	-	0x47
0xFFFF F142	Interrupt control register	TAA1CC1IC	R/W	R/W	-	-	0x47
0xFFFF F144	Interrupt control register	TAA1OVIC	R/W	R/W	-	-	0x47
0xFFFF F146	Interrupt control register	TAA2CC0IC	R/W	R/W	-	-	0x47
0xFFFF F148	Interrupt control register	TAA2CC1IC	R/W	R/W	-	-	0x47
0xFFFF F14A	Interrupt control register	TAA2OVIC	R/W	R/W	-	-	0x47
0xFFFF F14C	Interrupt control register	TAA3CC0IC	R/W	R/W	-	-	0x47
0xFFFF F14E	Interrupt control register	TAA3CC1IC	R/W	R/W	-	-	0x47
0xFFFF F150	Interrupt control register	TAA3OVIC	R/W	R/W	-	-	0x47
0xFFFF F152	Interrupt control register	TAB0CC0IC	R/W	R/W	-	-	0x47
0xFFFF F154	Interrupt control register	TAB0CC1IC	R/W	R/W	-	-	0x47
0xFFFF F156	Interrupt control register	TAB0CC2IC	R/W	R/W	-	-	0x47
0xFFFF F158	Interrupt control register	TAB0CC3IC	R/W	R/W	-	-	0x47
0xFFFF F15A	Interrupt control register	TAB0OVIC	R/W	R/W	-	-	0x47
0xFFFF F15C	Interrupt control register	UD0TIC	R/W	R/W	-	-	0x47
0xFFFF F15E	Interrupt control register	UD1TIC	R/W	R/W	-	-	0x47
0xFFFF F160	Interrupt control register	UD0RIC	R/W	R/W	-	-	0x47
0xFFFF F162	Interrupt control register	UD1RIC	R/W	R/W	-	-	0x47
0xFFFF F164	Interrupt control register	UD0SIC	R/W	R/W	-	-	0x47
0xFFFF F166	Interrupt control register	UD1SIC	R/W	R/W	-	-	0x47
0xFFFF F168	Interrupt control register	TAB1CC0IC	R/W	R/W	-	-	0x47
0xFFFF F16A	Interrupt control register	TAB1CC1IC	R/W	R/W	-	-	0x47
0xFFFF F16C	Interrupt control register	TAB1CC2IC	R/W	R/W	-	-	0x47
0xFFFF F16E	Interrupt control register	TAB1CC3IC	R/W	R/W	-	-	0x47
0xFFFF F170	Interrupt control register	TAB1OVIC	R/W	R/W	-	-	0x47
0xFFFF F172	Interrupt control register	BRG0IC	R/W	R/W	-	-	0x47
0xFFFF F174	Interrupt control register	BRG1IC	R/W	R/W	-	-	0x47
0xFFFF F176	Interrupt control register	C0ERRIC	R/W	R/W	-	-	0x47
0xFFFF F178	Interrupt control register	C0RECIC	R/W	R/W	-	-	0x47
0xFFFF F17A	Interrupt control register	C0TRXIC	R/W	R/W	-	-	0x47
0xFFFF F17C	Interrupt control register	C0WUPIC	R/W	R/W	-	-	0x47
0xFFFF F17E	Interrupt control register	C1ERRIC	R/W	R/W	-	-	0x47
0xFFFF F180	Interrupt control register	C1RECIC	R/W	R/W	-	-	0x47
0xFFFF F182	Interrupt control register	C1TRXIC	R/W	R/W	-	-	0x47
0xFFFF F184	Interrupt control register	C1WUPIC	R/W	R/W	-	-	0x47
0xFFFF F186	Interrupt control register	AD0DIC	R/W	R/W	-	-	0x47
0xFFFF F188	Interrupt control register	CE0TDIC	R/W	R/W	-	-	0x47
0xFFFF F18A	Interrupt control register	CE0RDIC	R/W	R/W	-	-	0x47
0xFFFF F18C	Interrupt control register	CE1TDIC	R/W	R/W	-	-	0x47
0xFFFF F18E	Interrupt control register	CE1RDIC	R/W	R/W	-	-	0x47
0xFFFF F190	Interrupt control register	CF0TDIC	R/W	R/W	-	-	0x47
0xFFFF F192	Interrupt control register	CF0RDIC	R/W	R/W	-	-	0x47
0xFFFF F194	Interrupt control register	CF1TDIC	R/W	R/W	-	-	0x47
0xFFFF F196	Interrupt control register	CF1RDIC	R/W	R/W	-	-	0x47

Table A-2 Other special function registers (3/10)

Address	Register name	SFR	1	8	16	32	Reset Value
0xFFFF F1F6	Interrupt control register	FLIC	R/W	R/W	-	-	0x47
0xFFFF F1FA	In-service priority register	ISPR	R	R	-	-	0X00
0xFFFF F1FC	Command register	PRCMD	-	W	-	-	0x00
0xFFFF F200	ADC conversion result register 0	ADCR00	-	-	R	-	0x0000
0xFFFF F201	ADC conversion result register 0H	ADCR0H0	-	R	-	-	0x00
0xFFFF F202	ADC conversion result register 1	ADCR01	-	-	R	-	0x0000
0xFFFF F203	ADC conversion result register 1H	ADCR0H1	-	R	-	-	0x00
0xFFFF F204	ADC conversion result register 2	ADCR02	-	-	R	-	0x0000
0xFFFF F205	ADC conversion result register 2H	ADCR0H2	-	R	-	-	0x00
0xFFFF F206	ADC conversion result register 3	ADCR03	-	-	R	-	0x0000
0xFFFF F207	ADC conversion result register 3H	ADCR0H3	-	R	-	-	0x00
0xFFFF F208	ADC conversion result register 4	ADCR4	-	-	R	-	0x0000
0xFFFF F209	ADC conversion result register 4H	ADCR0H4	-	R	-	-	0x00
0xFFFF F20A	ADC conversion result register 5	ADCR05	-	-	R	-	0x0000
0xFFFF F20B	ADC conversion result register 5H	ADCR0H5	-	R	-	-	0x00
0xFFFF F20C	ADC conversion result register 6	ADCR06	-	-	R	-	0x0000
0xFFFF F20D	ADC conversion result register 6H	ADCR0H6	-	R	-	-	0x00
0xFFFF F20E	ADC conversion result register 7	ADCR07	-	-	R	-	0x0000
0xFFFF F20F	ADC conversion result register 7H	ADCR0H7	-	R	-	-	0x00
0xFFFF F210	ADC conversion result register 8	ADCR08	-	-	R	-	0x0000
0xFFFF F211	ADC conversion result register 8H	ADCR0H8	-	R	-	-	0x00
0xFFFF F212	ADC conversion result register 9	ADCR09	-	-	R	-	0x0000
0xFFFF F213	ADC conversion result register 9H	ADCR0H9	-	R	-	-	0x00
0xFFFF F214	ADC conversion result register 10	ADCR010	-	-	R	-	0x0000
0xFFFF F215	ADC conversion result register 10H	ADCR0H10	-	R	-	-	0x00
0xFFFF F216	ADC conversion result register 11	ADCR011	-	-	R	-	0x0000
0xFFFF F217	ADC conversion result register 11H	ADCR0H11	-	R	-	-	0x00
0xFFFF F218	ADC conversion result register 12	ADCR012	-	-	R	-	0x0000
0xFFFF F219	ADC conversion result register 12H	ADCR0H12	-	R	-	-	0x00
0xFFFF F21A	ADC conversion result register 13	ADCR013	-	-	R	-	0x0000
0xFFFF F21B	ADC conversion result register 13H	ADCR0H13	-	R	-	-	0x00
0xFFFF F21C	ADC conversion result register 14	ADCR014	-	-	R	-	0x0000
0xFFFF F21D	ADC conversion result register 14H	ADCR0H14	-	R	-	-	0x00
0xFFFF F21E	ADC conversion result register 15	ADCR015	-	-	R	-	0x0000
0xFFFF F21F	ADC conversion result register 15H	ADCR0H15	-	R	-	-	0x00
0xFFFF F220	ADC conversion result register for AVDD	ADCRDD00	-	-	R	-	0x0000
0xFFFF F221	ADC conversion result register AVDDH	ADCRDD0H0	-	R	-	-	0x00
0xFFFF F222	ADC conversion result register for 2/3AVDD	ADCRDD01	-	-	R	-	0x0000
0xFFFF F223	ADC conversion result register 2/3AVDDH	ADCRDD0H1	-	R	-	-	0x00
0xFFFF F224	ADC conversion result register for 1/2AVDD	ADCRDD02	-	-	R	-	0x0000
0xFFFF F225	ADC conversion result register 1/2AVDDH	ADCRDD0H2	-	R	-	-	0x00
0xFFFF F226	ADC conversion result register AVSS	ADCRSS0	-	-	R	-	0x0000
0xFFFF F227	ADC conversion result register AVSSH	ADCRSS0H	-	R	-	-	0x00
0xFFFF F22E	ADC result register for DMA	ADDMA0	-	-	R	-	0x0000

Table A-2 Other special function registers (4/10)

Address	Register name	SFR	1	8	16	32	Reset Value
0xFFFF F230	ADC mode register 0	ADM00	R/W	R/W	-	-	0x00
0xFFFF F231	ADC mode register 1	ADM01	R/W	R/W	-	-	0x00
0xFFFF F232	ADC mode register 2	ADM02	R/W	R/W	-	-	0x00
0xFFFF F233	ADC mode register 3	ADM03	R/W	R/W	-	-	0x00
0xFFFF F300	DMA transfer memory start address register 0	MAR0	-	-	R/W	-	0x0000
0xFFFF F304	DMA transfer memory start address register 2	MAR2	-	-	R/W	-	0x0000
0xFFFF F306	DMA transfer memory start address register 3	MAR3	-	-	R/W	-	0x0000
0xFFFF F308	DMA transfer memory start address register 4	MAR4	-	-	R/W	-	0x0000
0xFFFF F30A	DMA transfer memory start address register 5	MAR5	-	-	R/W	-	0x0000
0xFFFF F30C	DMA transfer memory start address register 6	MAR6	-	-	R/W	-	0x0000
0xFFFF F30E	DMA transfer memory start address register 7	MAR7	-	-	R/W	-	0x0000
0xFFFF F324	DMA transfer SFR start address register 2	SAR2	-	R/W	-	-	0x00
0xFFFF F326	DMA transfer SFR start address register 3	SAR3	-	R/W	-	-	0x00
0xFFFF F340	DMA transfer count register 0	DTCR0	-	R/W	-	-	0x00
0xFFFF F344	DMA transfer count register 2	DTCR2	-	R/W	-	-	0x00
0xFFFF F346	DMA transfer count register 3	DTCR3	-	R/W	-	-	0x00
0xFFFF F348	DMA transfer count register 4	DTCR4	-	R/W	-	-	0x00
0xFFFF F34A	DMA transfer count register 5	DTCR5	-	R/W	-	-	0x00
0xFFFF F34C	DMA transfer count register 6	DTCR6	-	R/W	-	-	0x00
0xFFFF F34E	DMA transfer count register 7	DTCR7	-	R/W	-	-	0x00
0xFFFF F360	DMA mode control register	DMAMC	-	R/W	-	-	0x00
0xFFFF F362	DMA status register	DMAS	-	R/W	-	-	0x00
0xFFFF F364	DMA data size control register	DMADSC	-	R/W	-	-	0x00
0xFFFF F388	DMA trigger factor register 4	DTFR4	-	R/W	-	-	0x00
0xFFFF F38A	DMA trigger factor register 5	DTFR5	-	R/W	-	-	0x00
0xFFFF F38C	DMA trigger factor register 6	DTFR6	-	R/W	-	-	0x00
0xFFFF F38E	DMA trigger factor register 7	DTFR7	-	R/W	-	-	0x00
0xFFFF F400	P0 port register	P0	R/W	R/W	-	-	0x00
0xFFFF F402	P1 port register	P1	R/W	R/W	-	-	0x00
0xFFFF F404	P2 port register	P2	R/W	R/W	-	-	0x00
0xFFFF F406	P3 port register	P3	R/W	R/W	-	-	0x00
0xFFFF F408	P4 port register	P4	R/W	R/W	-	-	0x00
0xFFFF F40A	P5 port register	P5	R/W	R/W	-	-	0x00
0xFFFF F40C	P6 port register	P6	R/W	R/W	-	-	0x00
0xFFFF F410	P8 port register	P8	R/W	R/W	-	-	0x00
0xFFFF F420	P0 Port mode register	PM0	R/W	R/W	-	-	0xF7
0xFFFF F422	P1 Port mode register	PM1	R/W	R/W	-	-	0xFF
0xFFFF F424	P2 Port mode register	PM2	R/W	R/W	-	-	0xFF
0xFFFF F426	P3 Port mode register	PM3	R/W	R/W	-	-	0xFF
0xFFFF F428	P4 Port mode register	PM4	R/W	R/W	-	-	0xFF
0xFFFF F42A	P5 Port mode register	PM5	R/W	R/W	-	-	0xFF
0xFFFF F430	P8 Port mode register	PM8	R/W	R/W	-	-	0xFF
0xFFFF F440	P0 Port mode control register	PMC0	R/W	R/W	-	-	0x00
0xFFFF F442	P1 Port mode control register	PMC1	R/W	R/W	-	-	0x00

Table A-2 Other special function registers (5/10)

Address	Register name	SFR	1	8	16	32	Reset Value
0xFFFF F444	P2 Port mode control register	PMC2	R/W	R/W	-	-	0x00
0xFFFF F446	P3 Port mode control register	PMC3	R/W	R/W	-	-	0x00
0xFFFF F448	P4 Port mode control register	PMC4	R/W	R/W	-	-	0x00
0xFFFF F44A	P5 Port mode control register	PMC5	R/W	R/W	-	-	0x00
0xFFFF F44C	P6 Port mode control register	PMC6	R/W	R/W	-	-	0x00
0xFFFF F450	P8 Port mode control register	PMC8	R/W	R/W	-	-	0x00
0xFFFF F460	P0 Port function control register	PFC0	R/W	R/W	-	-	0x00
0xFFFF F462	P1 Port function control register	PFC1	R/W	R/W	-	-	0x00
0xFFFF F464	P2 Port function control register	PFC2	R/W	R/W	-	-	0x00
0xFFFF F46A	P5 Port function control register	PFC5	R/W	R/W	-	-	0x00
0xFFFF F470	P8 Port function control register	PFC8	R/W	R/W	-	-	0x00
0xFFFF F482	Bus cycle type configuration register 1	BCT1	-	-	R/W	-	0x4444
0xFFFF F486	Data wait control register 1	DWC1	-	-	R/W	-	0x3333
0xFFFF F48A	Bus cycle control register	BCC	-	-	R/W	-	0xAAAA
0xFFFF F48E	Bus clock divide register	DVC	-	R/W	-	-	0x00
0xFFFF F600	TAA0 control register 0	TAA0CTL0	R/W	R/W	-	-	0x00
0xFFFF F601	TAA0 control register 1	TAA0CTL1	R/W	R/W	-	-	0x00
0xFFFF F602	TAA0 I/O control register 0	TAA0IOC0	R/W	R/W	-	-	0x00
0xFFFF F603	TAA0 I/O control register 1	TAA0IOC1	R/W	R/W	-	-	0x00
0xFFFF F604	TAA0 I/O control register 2	TAA0IOC2	R/W	R/W	-	-	0x00
0xFFFF F605	TAA0 option register 0	TAA0OPT0	R/W	R/W	-	-	0x00
0xFFFF F606	TAA0 capture/compare register 0	TAA0CCR0	-	-	R/W	-	0x0000
0xFFFF F608	TAA0 capture/compare register 1	TAA0CCR1	-	-	R/W	-	0x0000
0xFFFF F60A	TAA0 counter read buffer register	TAA0CNT	-	-	R	-	0x0000
0xFFFF F60C	TAA0 I/O control register 4	TAA0IOC4	R/W	R/W	-	-	0x00
0xFFFF F620	TAA1 control register 0	TAA1CTL0	R/W	R/W	-	-	
0xFFFF F621	TAA1 control register 1	TAA1CTL1	R/W	R/W	-	-	0x00
0xFFFF F622	TAA1 I/O control register 0	TAA1IOC0	R/W	R/W	-	-	0x00
0xFFFF F623	TAA1 I/O control register 1	TAA1IOC1	R/W	R/W	-	-	0x00
0xFFFF F624	TAA1 I/O control register 2	TAA1IOC2	R/W	R/W	-	-	0x00
0xFFFF F625	TAA1 option register 0	TAA1OPT0	R/W	R/W	-	-	0x00
0xFFFF F626	TAA1 capture/compare register 0	TAA1CCR0	-	-	R/W	-	0x0000
0xFFFF F628	TAA1 capture/compare register 1	TAA1CCR1	-	-	R/W	-	0x0000
0xFFFF F62A	TAA1 counter read buffer register	TAA1CNT	-	-	R	-	0x0000
0xFFFF F62C	TAA1 I/O control register 4	TAA1IOC4	R/W	R/W	-	-	0x00
0xFFFF F640	TAA2 control register 0	TAA2CTL0	R/W	R/W	-	-	0x00
0xFFFF F641	TAA2 control register 1	TAA2CTL1	R/W	R/W	-	-	0x00
0xFFFF F642	TAA2 I/O control register 0	TAA2IOC0	R/W	R/W	-	-	0x00
0xFFFF F643	TAA2 I/O control register 1	TAA2IOC1	R/W	R/W	-	-	0x00
0xFFFF F644	TAA2 I/O control register 2	TAA2IOC2	R/W	R/W	-	-	0x00
0xFFFF F645	TAA2 option register 0	TAA2OPT0	R/W	R/W	-	-	0x00
0xFFFF F646	TAA2 capture/compare register 0	TAA2CCR0	-	-	R/W	-	0x0000
0xFFFF F648	TAA2 capture/compare register 1	TAA2CCR1	-	-	R/W	-	0x0000
0xFFFF F64A	TAA2 counter read buffer register	TAA2CNT	-	-	R	-	0x0000

Table A-2 Other special function registers (6/10)

Address	Register name	SFR	1	8	16	32	Reset Value
0xFFFF F64C	TAA2 I/O control register 4	TAA2IOC4	R/W	R/W	-	-	0x00
0xFFFF F660	TAA3 control register 0	TAA3CTL0	R/W	R/W	-	-	0x00
0xFFFF F661	TAA3 control register 1	TAA3CTL1	R/W	R/W	-	-	0x00
0xFFFF F662	TAA3 I/O control register 0	TAA3IOC0	R/W	R/W	-	-	0x00
0xFFFF F663	TAA3 I/O control register 1	TAA3IOC1	R/W	R/W	-	-	0x00
0xFFFF F664	TAA3 I/O control register 2	TAA3IOC2	R/W	R/W	-	-	0x00
0xFFFF F665	TAA3 option register 0	TAA3OPT0	R/W	R/W	-	-	0x00
0xFFFF F666	TAA3 capture/compare register 0	TAA3CCR0	-	-	R/W	-	0x0000
0xFFFF F668	TAA3 capture/compare register 1	TAA3CCR1	-	-	R/W	-	0x0000
0xFFFF F66A	TAA3 counter read buffer register	TAA3CNT	-	-	R	-	0x00
0xFFFF F66C	TAA3 I/O control register 4	TAA3IOC4	R/W	R/W	-	-	0x00
0xFFFF F66D	TAA3 option register 1	TAA3OPT1	R/W	R/W	-	-	0x00
0xFFFF F680	TAB0 control register 0	TAB0CTL0	R/W	R/W	-	-	0x00
0xFFFF F681	TAB0 control register 1	TAB0CTL1	R/W	R/W	-	-	0x00
0xFFFF F682	TAB0 I/O control register 0	TAB0IOC0	R/W	R/W	-	-	0x00
0xFFFF F683	TAB0 I/O control register 1	TAB0IOC1	R/W	R/W	-	-	0x00
0xFFFF F684	TAB0 I/O control register 2	TAB0IOC2	R/W	R/W	-	-	0x00
0xFFFF F685	TAB0 option register 0	TAB0OPT0	R/W	R/W	-	-	0x00
0xFFFF F686	TAB0 capture/compare register 0	TAB0CCR0	-	-	R/W	-	0x0000
0xFFFF F688	TAB0 capture/compare register 1	TAB0CCR1	-	-	R/W	-	0x0000
0xFFFF F68A	TAB0 capture/compare register 2	TAB0CCR2	-	-	R/W	-	0x0000
0xFFFF F68C	TAB0 capture/compare register 3	TAB0CCR3	-	-	R/W	-	0x0000
0xFFFF F68E	TAB0 counter read buffer register	TAB0CNT	-	-	R	-	0x0000
0xFFFF F690	TAB0 I/O control register 4	TAB0IOC4	R/W	R/W	-	-	0x00
0xFFFF F6A0	TAB1 timer control register0	TAB1CTL0	R/W	R/W	-	-	0x00
0xFFFF F6A1	TAB1 timer control register1	TAB1CTL1	R/W	R/W	-	-	0x00
0xFFFF F6A2	TAB1 I/O control register 0	TAB1IOC0	R/W	R/W	-	-	0x00
0xFFFF F6A3	TAB1 I/O control register 1	TAB1IOC1	R/W	R/W	-	-	0x00
0xFFFF F6A4	TAB1 I/O control register 2	TAB1IOC2	R/W	R/W	-	-	0x00
0xFFFF F6A5	TAB1 option register 0	TAB1OPT0	R/W	R/W	-	-	0x00
0xFFFF F6A6	TAB1 capture/compare register 0	TAB1CCR0	-	-	R/W	-	0x0000
0xFFFF F6A8	TAB1 capture/compare register 1	TAB1CCR1	-	-	R/W	-	0x0000
0xFFFF F6AA	TAB1 capture/compare register 2	TAB1CCR2	-	-	R/W	-	0x0000
0xFFFF F6AC	TAB1 capture/compare register 3	TAB1CCR3	-	-	R/W	-	0x0000
0xFFFF F6AE	TAB1 counter read buffer register	TAB1CNT	-	-	R	-	0x0000
0xFFFF F6B0	TAB1 I/O control register 4	TAB1IOC4	R/W	R/W	-	-	0x00
0xFFFF F6C0	TMM0 timer control register0	TM0CTL0	R/W	R/W	-	-	0x00
0xFFFF F6C4	TMM0 compare register 0	TM0CMP0	-	-	R/W	-	0x0000
0xFFFF F6D0	TMM1 timer control register0	TM1CTL0	R/W	R/W	-	-	0x00
0xFFFF F6D4	TMM1 compare register 0	TM1CMP0	-	-	R/W	-	0x0000
0xFFFF F6E0	Selector control register	SEL0CNT	R/W	R/W	-	-	0x00
0xFFFF F700	P0 Pin Status Read Register	PPR0	R/W	R/W	-	-	0x00
0xFFFF F702	P1 Pin Status Read Register	PPR1	R/W	R/W	-	-	0x00
0xFFFF F704	P2 Pin Status Read Register	PPR2	R/W	R/W	-	-	0x00

Table A-2 Other special function registers (7/10)

Address	Register name	SFR	1	8	16	32	Reset Value
0xFFFF F706	P3 Pin Status Read Register	PPR3	R/W	R/W	-	-	0x00
0xFFFF F708	P4 Pin Status Read Register	PPR4	R/W	R/W	-	-	0x00
0xFFFF F70A	P5 Pin Status Read Register	PPR5	R/W	R/W	-	-	0x00
0xFFFF F70C	P6 Pin Status Read Register	PPR6	R/W	R/W	-	-	0x00
0xFFFF F710	P8 Pin Status Read Register	PPR8	R/W	R/W	-	-	0x00
0xFFFF F890	16 bit access to LVIM/LVIS[a]	LVIM_LVIS	-	-	R/W	-	0x0000
0xFFFF F890	Low voltage detection register[b]	LVIM	R/W	R/W	-	-	0x00
0xFFFF F891	Low voltage detection level select register[b]	LVIS	R/W	R/W	-	-	0x00
0xFFFF F8B0	RAM ECC error address register	RAMEAD	-	-	R	-	0x0000
0xFFFF F8B2	RAM ECC error data location register	RAMEDLR	-	R	-	-	0x00
0xFFFF F8B4	RAM ECC control register	RAMECC	-	R/W	-	-	0x00
0xFFFF F9FC	On Chip debug mode enable register	OCDM	R/W	R/W	-	-	0x01
0xFFFF FA00	UARTD0 control register 0	UD0CTL0	R/W	R/W	-	-	0x10
0xFFFF FA01	UARTD0 control register 1	UD0CTL1	-	R/W	-	-	0x00
0xFFFF FA02	UARTD0 control register 2	UD0CTL2	-	R/W	-	-	0xFF
0xFFFF FA03	UARTD0 option control register 0	UD0OPT0	R/W	R/W	-	-	0x14
0xFFFF FA04	UARTD0 status register	UD0STR	R/W	R/W	-	-	0x00
0xFFFF FA05	UARTD0 option control register 1	UD0OPT1	-	R/W	-	-	0x00
0xFFFF FA06	UARTD0 receive data register	UD0RX	-	R	-	-	0xFF
0xFFFF FA07	UARTD0 transmit data register	UD0TX	-	R/W	-	-	0xFF
0xFFFF FA10	UARTD1 control register 0	UD1CTL0	R/W	R/W	-	-	0x10
0xFFFF FA11	UARTD1 control register 1	UD1CTL1	-	R/W	-	-	0x00
0xFFFF FA12	UARTD1 control register 2	UD1CTL2	-	R/W	-	-	0xFF
0xFFFF FA13	UARTD1 option control register 0	UD1OPT0	R/W	R/W	-	-	0x14
0xFFFF FA14	UARTD1 status register	UD1STR	R/W	R/W	-	-	0x00
0xFFFF FA15	UARTD1 option control register 1	UD1OPT1	-	R/W	-	-	0x00
0xFFFF FA16	UARTD1 receive data register	UD1RX	-	R	-	-	0xFF
0xFFFF FA17	UARTD1 transmit data register	UD1TX	-	R/W	-	-	0xFF
0xFFFF FB00	Interrupt factor selection register	INTSEL	R/W	R/W			0x80
0xFFFF FB02	Interrupt error factor hold register	INTERRF	R/W	R/W			0x00
0xFFFF FC00	External interrupt falling edge specification register 0	INTF0	R/W	R/W	-	-	0x00
0xFFFF FC02	External interrupt falling edge specification register 1	INTF1	R/W	R/W	-	-	0x00
0xFFFF FC04	External interrupt rising edge specification register 0	INTR0	R/W	R/W	-	-	0x00
0xFFFF FC06	External interrupt rising edge specification register 1	INTR1	R/W	R/W	-	-	0x00
0xFFFF FC40	P0 Pull-up/down resistor control register[b]	PE0	R/W	R/W	-	-	0xF7[note]
0xFFFF FC40	P0 Pull-up control register[b]	PU0	R/W	R/W	-	-	0x00
0xFFFF FC42	P1 Pull-up/down resistor control register[b]	PE1	R/W	R/W	-	-	0xFF[note]
0xFFFF FC42	P1 Pull-up control register[c]	PU1	R/W	R/W	-	-	0x00
0xFFFF FC44	P2 Pull-up/down resistor control register[b]	PE2	R/W	R/W	-	-	0xFF[note]

Note a) This register is for 5V devices only.
b) Reset value of pull up/down registers depends on the setting of P60/$\overline{\text{PUDSEL}}$ pin

Table A-2 Other special function registers (8/10)

Address	Register name	SFR	1	8	16	32	Reset Value
0xFFFF FC44	P2 Pull-up control register[c]	PU2	R/W	R/W	-	-	0x00
0xFFFF FC46	P3 Pull-up/down resistor control register[b]	PE3	R/W	R/W	-	-	0xFF[note]
0xFFFF FC46	P3 Pull-up control register[c]	PU3	R/W	R/W	-	-	0x00
0xFFFF FC48	P4 Pull-up/down resistor control register[b]	PE4	R/W	R/W	-	-	0xFF[note]
0xFFFF FC48	P4 Pull-up control register[c]	PU4	R/W	R/W	-	-	0x00
0xFFFF FC4A	P5 Pull-up/down resistor control register[b]	PE5	R/W	R/W	-	-	0xFF[note]
0xFFFF FC4A	P5 Pull-up control register[c]	PU5	R/W	R/W	-	-	0x00
0xFFFF FC50	P8 Pull-up/down resistor control register[b]	PE8	R/W	R/W	-	-	0xFF[note]
0xFFFF FC50	P8 Pull-up control register[c]	PU8	R/W	R/W	-	-	0x00
0xFFFF FC60	P0 Pull-down control register[c]	PD0	R/W	R/W	-	-	0x00
0xFFFF FC62	P1 Pull-down control register[c]	PD1	R/W	R/W	-	-	0x00
0xFFFF FC64	P2 Pull-down control register[c]	PD2	R/W	R/W	-	-	0x00
0xFFFF FC66	P3 Pull-down control register[c]	PD3	R/W	R/W	-	-	0x00
0xFFFF FC68	P4 Pull-down control register[c]	PD4	R/W	R/W	-	-	0x00
0xFFFF FC6A	P5 Pull-down control register[c]	PD5	R/W	R/W	-	-	0x00
0xFFFF FC70	P6 Pull-down control register[c]	PD8	R/W	R/W	-	-	0x00
0xFFFF FCEC	Flash ROM ECC error register	ROMEAD	-	-	-	R	0x0000 0000
0xFFFF FD00	CSIF0 control register 0	CF0CTL0	R/W	R/W	-	-	0x01
0xFFFF FD01	CSIF0 control register 1	CF0CTL1	R/W	R/W	-	-	0x00
0xFFFF FD02	CSIF0 control register 2	CF0CTL2	-	R/W	-	-	0x00
0xFFFF FD03	CSIF0 status register	CF0STR	R/W	R/W	-	-	0x00
0xFFFF FD04	CSIF0 receive data register low byte	CF0RX0L	-	R	-	-	0x00
0xFFFF FD04	CSIF0 receive data register	CF0RX0	-	-	R	-	0x0000
0xFFFF FD06	CSIF0 transmit data register low byte	CF0TX0L	-	R/W	-	-	0x00
0xFFFF FD06	CSIF0 transmit data register	CF0TX0	-	-	R/W	-	0x0000
0xFFFF FD10	Prescaler mode register	BRG0PRSM	R/W	R/W	-	-	0x00
0xFFFF FD11	Prescaler compare register	BRG0PRSCM	-	R/W	-	-	0x00
0xFFFF FD14	Prescaler mode register	BRG1PRSM	R/W	R/W	-	-	0x00
0xFFFF FD15	Prescaler compare register	BRG1PRSCM	-	R/W	-	-	0x00
0xFFFF FD1A	Clock control register	CKC	R/W	R/W	-	-	0x03
0xFFFF FD20	CSIF1 control register 0	CF1CTL0	R/W	R/W	-	-	0x01
0xFFFF FD21	CSIF1 control register 1	CF1CTL1	R/W	R/W	-	-	0x00
0xFFFF FD22	CSIF1 control register 2	CF1CTL2	-	R/W	-	-	0x00
0xFFFF FD23	CSIF1 status register	CF1STR	R/W	R/W	-	-	0x00
0xFFFF FD24	CSIF1 receive data register low byte	CF1RX0L	-	R	-	-	0x00
0xFFFF FD24	CSIF1 receive data register	CF1RX0	-	-	R	-	0x0000
0xFFFF FD26	CSIF1 transmit data register low byte	CF1TX0L	-	R/W	-	-	0x00
0xFFFF FD26	CSIF1 transmit data register	CF1TX0	-	-	R/W	-	0x0000
0xFFFF FD30	CSIE0 Chip Select Glue Control	CSGLCTL0	R/W	R/W	-	-	0x00

Note b) Only valid for these devices: µPD70F3464, µPD70F3465, µPD70F3466. Reset value of pull up/down registers depends on the setting of P60/$\overline{\text{PUDSEL}}$ pin.

c) Only valid for these devices: µPD70F3470, µPD70F3471, µPD70F3472

Table A-2 Other special function registers (9/10)

Address	Register name	SFR	1	8	16	32	Reset Value
0xFFFF FD34	CSIE1 Chip Select Glue Control	CSGLCTL1	R/W	R/W	-	-	0x00
0xFFFF FD40	CSIE0 control register 0	CE0CTL0	R/W	R/W	-	-	0x00
0xFFFF FD41	CSIE0 control register 1	CE0CTL1	R/W	R/W	-	-	0x07
0xFFFF FD42	CSIE0 receive data buffer	CE0RX0	-	-	R		0x0000
0xFFFF FD42	CSIE0 receive data buffer L	CE0RXL0	-	R	-	-	0x00
0xFFFF FD43	CSIE0 receive data buffer H	CE0RXH0	-	R	-	-	0x00
0xFFFF FD44	CSIE0 chip selection CSI buffer	CE0CS	-	-	R/W		0xFFFF
0xFFFF FD44	CSIE0 chip selection CSI buffer L	CE0CSL	-	R/W	-	-	0xFF
0xFFFF FD45	CSIE0 chip selection CSI buffer H	CE0CSH	-	R/W	-	-	0xFF
0xFFFF FD46	CSIE0 transmit data CSI buffer	CE0TX0	-	-	R/W		0x0000
0xFFFF FD46	CSIE0 transmit data CSI buffer L	CE0TXL0	-	R/W	-	-	0x00
0xFFFF FD47	CSIE0 transmit data CSI buffer H	CE0TXH0	-	R/W	-	-	0x00
0xFFFF FD48	CSIE0 FIFO buffer status buffer	CE0STR	R/W	R/W	-	-	0x20
0xFFFF FD49	CSIE0 control register 2	CE0CTL2	R/W	R/W	-	-	0x00
0xFFFF FD4C	CSIE0 control register 3	CE0CTL3	R/W	R/W	-	-	0x00
0xFFFF FD4D	CSIE0 control register 4	CE0CTL4	R/W	R/W	-	-	0x00
0xFFFF FD50	CSIE0 chip select timing register 0	CE0OPT0	-	-	R/W		0x0002
0xFFFF FD52	CSIE0 chip select timing register 1	CE0OPT1	-	-	R/W		0x0002
0xFFFF FD54	CSIE0 chip select timing register 2	CE0OPT2	-	-	R/W		0x0002
0xFFFF FD56	CSIE0 chip select timing register 3	CE0OPT3	-	-	R/W		0x0002
0xFFFF FD58	CSIE0 chip select timing register 4	CE0OPT4	-	-	R/W		0x0002
0xFFFF FD5A	CSIE0 chip select timing register 5	CE0OPT5	-	-	R/W		0x0002
0xFFFF FD5C	CSIE0 chip select timing register 6	CE0OPT6	-	-	R/W		0x0002
0xFFFF FD5E	CSIE0 chip select timing register 7	CE0OPT7	-	-	R/W		0x0002
0xFFFF FD80	CSIE1 control register 0	CE1CTL0	R/W	R/W	-	-	0x00
0xFFFF FD81	CSIE1 control register 1	CE1CTL1	R/W	R/W	-	-	0x07
0xFFFF FD82	CSIE1 receive data buffer	CE1RX0	-	-	R	-	0x0000
0xFFFF FD82	CSIE1 receive data buffer L	CE1RXL0	-	R	-	-	0x00
0xFFFF FD83	CSIE1 receive data buffer H	CE1RXH0	-	R	-	-	0x00
0xFFFF FD84	CSIE1 chip selection CSI buffer	CE1CS	-	-	R/W	-	0xFFFF
0xFFFF FD84	CSIE1 chip selection CSI buffer L	CE1CSL	-	R/W	-	-	0xFF
0xFFFF FD85	CSIE1 chip selection CSI buffer H	CE1CSH	-	R/W	-	-	0xFF
0xFFFF FD86	CSIE1 transmit data CSI buffer	CE1TX0	-	-	R/W	-	0x0000
0xFFFF FD86	CSIE1 transmit data CSI buffer L	CE1TXL0	-	R/W	-	-	0x00
0xFFFF FD87	CSIE1 transmit data CSI buffer H	CE1TXH0	-	R/W	-	-	0x00
0xFFFF FD88	CSIE1 FIFO buffer status buffer	CE1STR	R/W	R/W	-	-	0x20
0xFFFF FD89	CSIE1 control register 2	CE1CTL2	R/W	R/W	-	-	0x00
0xFFFF FD8C	CSIE1 control register 3	CE1CTL3	R/W	R/W	-	-	0x00
0xFFFF FD8D	CSIE1 control register 4	CE1CTL4	R/W	R/W	-	-	0x00
0xFFFF FD90	CSIE1 chip select timing register 0	CE1OPT0	-	-	R/W	-	0x0002
0xFFFF FD92	CSIE1 chip select timing register 1	CE1OPT1	-	-	R/W	-	0x0002

[c)] Only valid for these devices: µPD70F3470, µPD70F3471, µPD70F3472.

Table A-2 Other special function registers (10/10)

Address	Register name	SFR	1	8	16	32	Reset Value
0xFFFF FD94	CSIE1 chip select timing register 2	CE1OPT2	-	-	R/W	-	0x0002
0xFFFF FD96	CSIE1 chip select timing register 3	CE1OPT3	-	-	R/W	-	0x0002
0xFFFF FD98	CSIE1 chip select timing register 4	CE1OPT4	-	-	R/W	-	0x0002
0xFFFF FD9A	CSIE1 chip select timing register 5	CE1OPT5	-	-	R/W	-	0x0002
0xFFFF FD9C	CSIE1 chip select timing register 6	CE1OPT6	-	-	R/W	-	0x0002
0xFFFF FD9E	CSIE1 chip select timing register 7	CE1OPT7	-	-	R/W	-	0x0002
0xFFFF FDA8	Reset Source Flag Register	RESSTAT	-	R/W	-	-	0x01
0xFFFF FDAA	Software Reset Trigger Register	RESSWT	-	W	-	-	0x00
0xFFFF FDAC	Reset Protection Register	RESCMD	-	R/W	-	-	0x00
0xFFFF FDAE	Reset Status Register	RES	-	R/W	-	-	0x00
0xFFFF FDB8	Watchdog Timer mode register	WDTM	R/W	R/W	-	-	0x67
0xFFFF FDB9	Watchdog Timer enable register	WDTE	-	R/W	-	-	0x9A
0xFFFF FDC8	LVIM and LVIS write protection[d]	PHCMDLVI	-	W			0x00
0xFFFF FE20	DMA Wait Control Register	DMAWC0	-	R/W	-	-	0x37
0xFFFF FF28	Integer Prescaler Control Register	OCKS0	-	R/W	-	-	0x00
0xFFFF FF2E	RAM control macro write protection register	PRCMDIRA	-	R/W	-	-	0x00

d) This register is for 5V devices only.

Revision History

The following changes were made between June 2008 and July 2009.

What was revised	Page
Corrected the CPU chapter by removing information about the SYS register.	65
Updated the document number and status (no longer “preliminary”).	all

The following changes are done in UM version 3.0

What was revised	Page or location
Value of BPC - error on the note in the bottom of the page	56
Removed Duplicate entries : There are two locations showing the PRCMDIRA register with different descriptions-p.62, sect. 3.7.2.4 and p.64, sect. 3.8.1.2.	62, 64
Update the values of DVC	Sect 4.2.2
Add reference to OCKS0 register	Tbl 4-2
Address section: change BRG0PRSCM : <BRG_base> + 1H / BRG1PRSCM : <BRG_base> + 5H to BRGnPRSCM : <BRG_base> + 1H	Sect. 4.4.2.2 -
TAA1OVC changed to TAA1OVIC; TAB0CCOVIC changed to TAB0OVIC	Tbl 5-1 -
Changed "IMR7" to "IMR4"; change "m = 0 to 7" to "m = 0 to 4"	Sect. 5.3.5
Changed INTF0 in title to ISPR	Tbl 5-4
Changed INTR0 in title to INTR1	Tbl 5-8
Changed INTTERF in title to INTERRF	Tbl 5-10
Changed DMASL to DMAS, DMAMCL to DMAMC, DMADSCL to DMADSC	Tbl 8-3
Changed DMAS1 to "0" in the bit table; there is no channel 1.	Sect. 8.2.6
Changed DMAMCL to DMAMC; 2 instances	Sect. 8.2.7
Changed DE1 to "0" in the bit table; there is no channel 1.	Sect. 8.2.7
Changed "in 1 or 8-bit units" to "in 8-bit units"	Sect. 8.2.8
Change DMAMCL to DMAMC; 7 instances. Change DMASL to DMAS; 10 instances Change DMADSCL to DMADSC; 2 instances	sect. 8.3
SEL0CNT register be added to the Timer AA register table	Tbl 9-5
Fix typo; change second TABnCCR1 to TABnCCR3	Tbl 10-5
There was a question mark ('?') in one of the flowchart boxes that has been replaced with the proper character (arrow)	Fig. 10-24
Change WDTM2 to WDTM; 15 instances including fig. 13-1	Ch. 13
WDTM access should indicate also 1-bit access	Sect. 13.3.1
Change CFnRX to CFnRX0; 31 instances including fig. 14-1 Change CFnRXL to CFnRX0L; 1 instance Change CFnTX to CFnTX0; 28 instances including fig. 14-1 Change CFnTXL to CFnTX0L; 1 instance	Ch. 14
Change CEnRX0L to CEnRXL0; 3 instances Change CEnRX0H to CEnRXH0; 3 instances Change CEnTX0L to CEnTXL0; 8 instances Change CEnTX0H to CEnTXH0; 4 instances	Ch. 15
add CEnCSH to table.	Tbl 15-5

What was revised	Page or location
Change ADCRnmH to ADCRnHm; 12 instances Change ADCRn0H to ADCRnH0; 8 instances Change ADCRn1H to ADCRnH1; 6 instances Change ADCRn2H to ADCRnH2; 6 instances Change ADCRn3H to ADCRnH3; 6 instances Change ADCRn4H to ADCRnH4; 6 instances Change ADCRn5H to ADCRnH5; 4 instances Change ADCRn6H to ADCRnH6; 4 instances Change ADCRn7H to ADCRnH7; 6instances Change ADCRn8H to ADCRnH8; 6 instances Change ADCRn9H to ADCRnH9; 4 instances Change ADCRn10H to ADCRnH10; 4 instances (one instance is a typo- "ADCRn0H") Change ADCRn11H to ADCRnH11; 6 instances Change ADCRn12H to ADCRnH12; 6 instances Change ADCRn13H to ADCRnH13; 5 instances Change ADCRn14H to ADCRnH14; 5 instances Change ADCRn15H to ADCRnH15; 9 instances Change ADCRnDDu to ADCRDDnu; 20 instances Change ADCRnDDuH to ADCRDDnHu; 1 instances Change ADCRnDD0 to ADCRDDn0; 10 instances Change ADCRnDD0H to ADCRDDnH0; 3 instances Change ADCRnDD1 to ADCRDDn1; 9 instances Change ADCRnDD1H to ADCRDDnH1; 3 instances Change ADCRnDD2 to ADCRDDn2; 10 instances Change ADCRnDD2H to ADCRDDnH2; 3 instances Change ADCRnSS to ADCRSSn; 31 instances Change ADCRnSSH to ADCRSSnH; 4 instances Change ADDMA to ADDMAn; 4 instances REMOVE reference to ADDMAnH; 4 instances	Ch. 18, 19 (incl. figures and tables)
Register bit table - ANIn0 replaced by ANISn0	Section 18.5.3
"ACD" should be "ADC"; also for PLMn bit label, change "trigger mode" to "conversion mode"	Sect 19.3.6
"RESSW" should be "RESSWT"	Sect. 20.6
RESSTAT access should say read AND write	Sect. 20.8.1
"without discharge mode" should be "with discharge mode"	Tbl 19-7
Add LVIIC	Table A-2
DTCR0-DTCR7, DMAMC, DMAS, DMADSC, and DTFR4-DTFR7 should NOT have bit R/W capability; should be '-'. Remove entry for ADDMA0H.	Tbl A-2 (4/11)
DVC should NOT have bit R/W capability; should be '-'	Tbl A-2 (5/11)
Remove entries for TM0CTL1, TM0IOC0, TM1CTL1, TM1IOC0; R/W; no bit access. RESSTAT access should be R/W. WDTM should have R/W bit access	Tbl A-2 (7/11)
modify entries for INTSEL & INTERRF to indicate 1-bit & 8-bit R/W ability	Tbl A-2 (8/11)
CF0CTL2, CF1CTL2, BRG0PRSCM, BRG1PRSCM, CE0RXH0, CE0CSL, CE0CSH do NOT have bit access capability. CE0TX0 should be R/W; CE0TXL0 should be R/W - 8-bit only. Change address for ROMEAD to 0xFFFF FCEA.	Tbl A-2 (9/11)
CE1RXH0, CE1CSL, CE1CSH should NOT have bit access capability; should be '-'. CE0TXH0 should be R/W - 8-bit only. CE1TX0, CE1TXL0, CE1TXH0 should be R/W; no bit access. RESSTAT access should be R/W. WDTM should have R/W bit access.-	Tbl A-2 (10/11)
Change the description of SMP bit from “ADCn Buffer Mode Specification” to “**Sampling time selection bit**”	Section 18.5.4

What was revised	Page or location