User Manual

32

Hardware

V850ES/Fx3

32-bit Single-Chip Microcontroller

μPD70F3370A	μPD70F3377A	μPD70F3384
μPD70F3371	μPD70F3378	μPD70F3385
μPD70F3372	μPD70F3379	
μPD70F3373	μPD70F3380	
μPD70F3374	μPD70F3381	
μPD70F3375	μPD70F3382	
μPD70F3376A	μPD70F3383	

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

Renesas Electronics
www.renesas.com

R01UH0237ED0320, Rev. 3.20
January 15, 2014

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation ofsemiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by youor third parties arising from the use of these circuits, software, or information.
2. 2. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
3. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The recommended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.

“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 etc.
“High Quality”:	Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anticrime systems; and safety equipment etc.

   Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (nuclear reactor control systems, military equipment etc.). You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application for which it is not intended. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.
6. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the even t of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or systems manufactured by you.

8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You should not use Renesas Electronics products or technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. When exporting the Renesas Electronics products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
11. This document may not be reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

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

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

(2012.4)

General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU products from Renesas. For detailed usage notes on theproducts covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Handling of unused Pins

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

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

2. Processing at power-on

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

   – The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.
   In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.
   In a similar way, the states of pins in a product that is reset by an on-chip power-on reset functionare not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.

3. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited.

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

4. Clock signals

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

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

5. Differences between products

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

   – The characteristics of an MPU or MCU in the same group but having a different part number may differ in terms of the internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Table of Contents

Preface

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

Purpose This manual presents the hardware manual for the concerned microcontrollers.

Organization This system specification describes 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 notation are used as follows:

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

Note Additional remark or tip

Caution 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 = don’t care

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

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

Further Information For further information see http://www.renesas.eu/.

Chapter 1 Introduction

The V850ES/Fx3 is a product line in NEC Electronics' V850 family of single-chip microcontrollers designed for automotive applications.

1.1 General

The V850ES/Fx3 single-chip microcontroller devices make the performance gains attainable with 32-bit RISC-based controllers available for embedded control applications. The integrated V850ES CPU offers easy pipeline handling and programming, resulting in compact code size comparable to 16-bit CISC CPUs.

The V850ES/Fx3 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.

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

Thus equipped, the V850ES/Fx3 product line is ideally suited for automotive body applications. It is also an excellent choice for other applications where a combination of sophisticated peripheral functions and CAN network support is required.

(1) V850ES CPU

The V850ES 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.

(2) On-chip flash memory

The V850ES/Fx3 microcontrollers have on-chip flash memory. It is possible to program the controllers directly in the target environment where they are mounted.

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

(3) A full range of software development tools

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

1.2 Features Summary

The V850ES/Fx3 series includes the following microcontrollers:

- V850ES/FE3
  - μPD70F3370A
  - μPD70F3371
- V850ES/FF3
  - μPD70F3372
  - μPD70F3373
- V850ES/FG3
  - μPD70F3374
  - μPD70F3375
  - μPD70F3376A
  - μPD70F3377A
- V850ES/FJ3
  - μPD70F3378
  - μPD70F3379
  - μPD70F3380
  - μPD70F3381
  - μPD70F3382
- V850ES/FK3
  - μPD70F3383
  - μPD70F3384
  - μPD70F3385

The common CPU core provides:

- 83 instructions
- 32 general registers (32 bits each)
- Comprehensive instruction set:
  - V850ES (compatible with V850 plus added powerful instructions for reducing code and increasing execution speed)
  - Signed multiplication (16 bits × 16 bits → 32 bits or 32 bits × 32 bits → 64 bits) in 1 to 5 clocks
  - Saturated operation instructions (with overflow/underflow detection)
  - 32-bit shift instructions in 1 clock cycle
  - Bit manipulation instructions
  - Load/store instructions with long/short format
  - Signed load instructions

The following table gives an overview of the most outstanding controller features.

Table 1-1 V850ES/Fx3 features (1/2)

Series name		**V850ES/FE3**		**V850ES/FF3**		**V850ES/FG3**				**V850ES/FJ3**					**V850ES/FK3**		
Product		**‘F3370A**	**‘F3371**	**‘F3372**	**‘F3373**	**‘F3374**	**‘F3375**	**‘F3376A**	**‘F3377A**	**‘F3378**	**‘F3379**	**‘F3380**	**‘F3381**	**‘F3382**	**‘F3383**	**‘F3384**	**‘F3385**
CPU		V850ES (32 bit RISC)															
Internal memory	Code Flash	128 KB	256 KB	128 KB	256 KB	128 KB	256 KB	384 KB	512 KB	256 KB	384 KB	512 KB	768 KB	1024 KB	512 KB	768 KB	1024 KB
	RAM	8 KB	16 KB	8 KB	16 KB	8 KB	16 KB	24 KB	32 KB	16 KB	24 KB	32 KB	40 KB	48 KB	32 KB	48 KB	60 KB
	Data Flash	32 KB															
External memory interface		–								16-bit multiplexed address/data bus, 4 chips selects							
Operating clock	max. CPU frequency	32 MHz						48 MHz		32 MHz	48 MHz						
	PLL ratio	x 8															
	SSCG ratio	x 96															
	MainOSC	operates on 4 MHz to 16 MHz crystal															
	SubOSC	operates on RC or crystal															
	Low speed internal oscillator	typ. 240 KHz															
	High speed internal oscillator	typ. 8 MHz															
I/O ports		51		67		84				128					152		
Timers	TAA	5 ch													8 ch		
	TAB	1 ch				2 ch				3 ch							
	TMM	1 ch															
	Motor control	1 ch															
	Watch	1 ch															
	WDT2	1 ch															

Table 1-1 V850ES/Fx3 features (2/2)

Series name		V850ES/FE3		V850ES/FF3		V850ES/FG3				V850ES/FJ3					V850ES/FK3		
Product		**‘F3370A**	**‘F3371**	**‘F3372**	**‘F3373**	**‘F3374**	**‘F3375**	**‘F3376A**	**‘F3377A**	**‘F3378**	**‘F3379**	**‘F3380**	**‘F3381**	**‘F3382**	**‘F3383**	**‘F3384**	**‘F3385**
A/D Converter		10 bits x 10 ch		10 bits x 12 ch		10 bits x 16 ch				10 bits x 24 ch					10 bits x 24 ch 10 bits x 16 ch		
Serial interfaces	UART/LIN	2 ch				3 ch		5 ch		3 ch	6 ch				8 ch		
	CSI	2 ch								3 ch			4 ch				
	IIC	1 ch															
	CAN	1 ch				2 ch				3 ch	4 ch				5 ch		
DMA		4 ch															
Interrupts	External (incl. NMI)	9 ch				12 ch		13 ch		16 ch					17 ch		
	Internal	48 ch				60 ch		65 ch		71 ch	81 ch		83 ch		101 ch		
Other functions	Power save modes	HALT, IDLE1, IDLE2, Sub_IDLE, STOP															
	Key return input	8 ch															
	Clock Monitor	Yes															
	POC	Power-On-Clear typical below 3.5 V[a]															
	LVI	Low-Voltage Detection typical below 3.7 V / 4.0 V (selectable by software)[a]															
	On-chip debug	Yes															
Operating voltage		3.3 V to 5.5 V[a]															
Package		64-pin QFP		80-pin QFP		100-pin QFP				144-pin QFP					176-pin QFP		

a) Refer to Datasheet

1.3 Description

The following figure provides a functional block diagram of the V850ES/FE3, V850ES/FF3, and V850ES/FG3 microcontrollers.

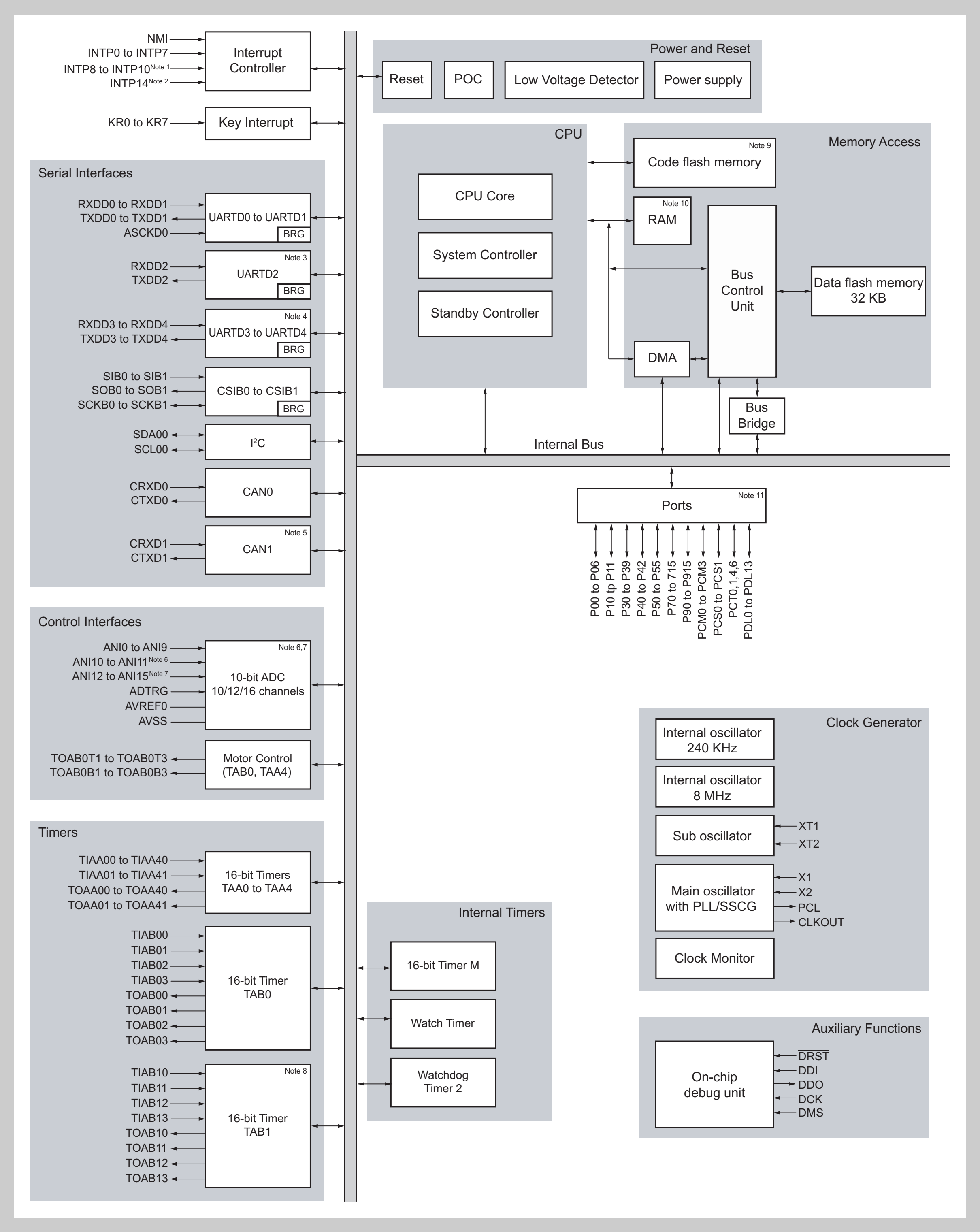


Figure 1-1 V850ES/FE3, V850ES/FF3, V850ES/FG3 block diagram

Table 1-2 on page 23 summarizes the different features of the V850ES/FE3, V850ES/FF3, V850ES/FG3 series devices, marked as “Notes” in *Figure 1-1 on page 22*.

Table 1-2 V850ES/FE3, V850ES/FF3, V850ES/FG3 feature set differences

Note	Feature	V850ES/FE3		V850ES/FF3		V850ES/FG3			
		‘F3370A	‘F3371	‘F3372	‘F3373	‘F3374	‘F3375	‘F3376A	‘F3377A
1	INTP8 to INTP10	–	–	–	–	√	√	√	√
2	INTP14	–	–	–	–	–	–	√	√
3	UARTD2	–	–	–	–	√	√	√	√
4	UARTD3 to UARTD4	–	–	–	–	–	–	√	√
5	CAN1	–	–	–	–	√	√	√	√
6	ANI10 to ANI11	–	–	√	√	√	√	√	√
7	ANI12 to ANI15	–	–	–	–	√	√	√	√
8	TAB1	–	–	–	–	√	√	√	√
9	Code flash	128 KB	256 KB	128 KB	256 KB	128 KB	256 KB	384 KB	512 KB
10	RAM	8 KB	16 KB	8 KB	16 KB	8 KB	16 KB	24 KB	32 KB
11	Ports	refer to *“Pin Functions” on page 32*							

The following figure provides a functional block diagram of the V850ES/FJ3 and V850ES/FK3 microcontrollers.

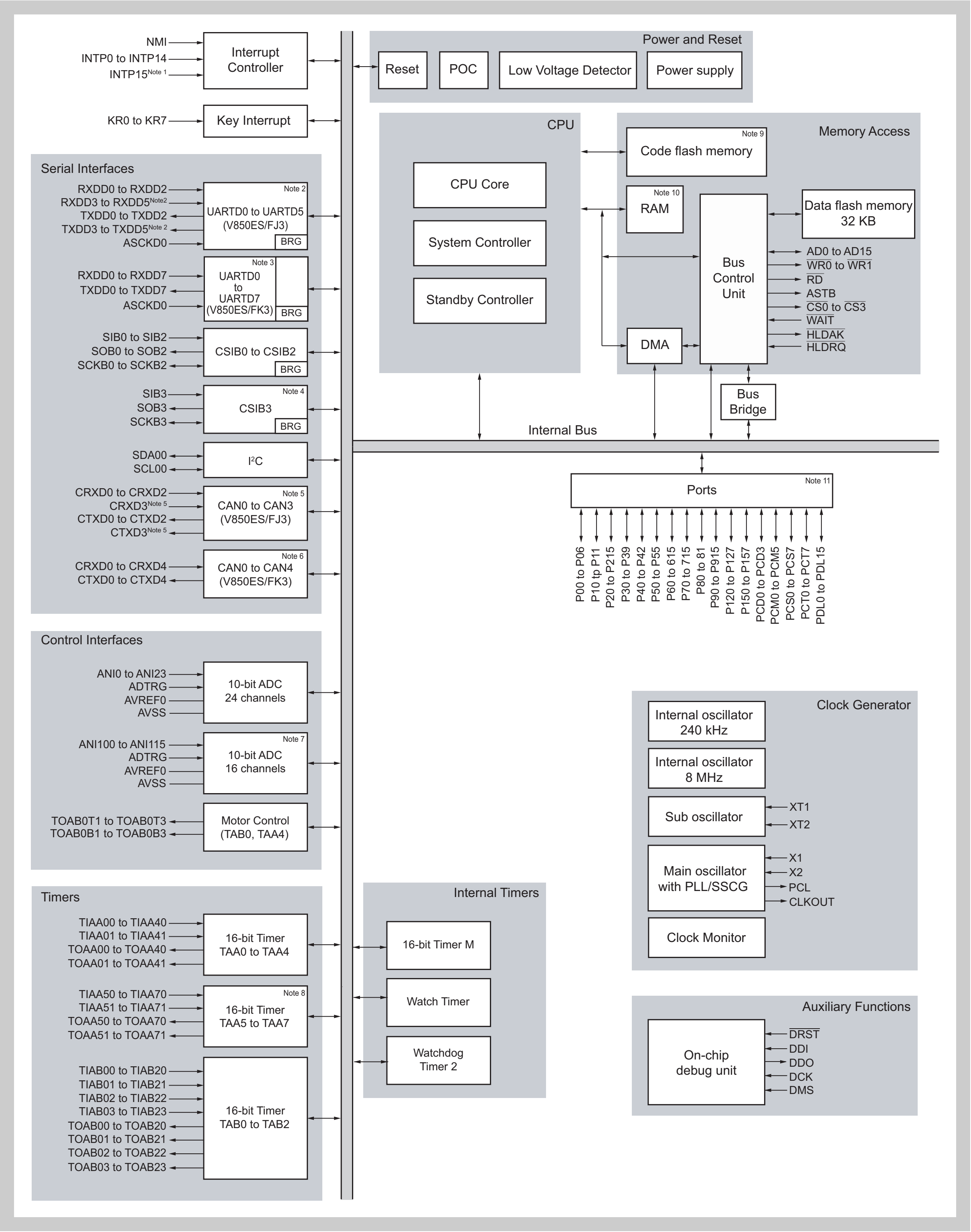


Figure 1-2 V850ES/FJ3 and V850ES/FK3 block diagram

Table 1-3 on page 25 summarizes the different features of the V850ES/FJ3, V850ES/FK3 series devices, marked as “Notes” in *Figure 1-2 on page 24*.

Table 1-3 V850ES/FJ3, V850ES/FK3 feature set differences

Note	Feature	V850ES/FJ3					V850ES/FK3		
		‘F3378	‘F3379	‘F3380	‘F3381	‘F3382	‘F3383	‘F3384	‘F3385
1	INTP15	–	–	–	–	–	√	√	√
2	UARTD3 to UARTD5	–	√	√	√	√	–	–	–
3	UARTD0 to UARTD7	–	–	–	–	–	√	√	√
4	CSIB3	–	–	–	√	√	√	√	√
5	CAN3	–	√	√	√	√	√	√	√
6	CAN4	–	–	–	–	–	√	√	√
7	ANI100 to ANI115	–	–	–	–	–	√	√	√
8	TAA5 to TAA7	–	–	–	–	–	√	√	√
9	Code flash	256 KB	384 KB	512 KB	768 KB	1024 KB	512 KB	768 KB	1024 KB
10	RAM	16 KB	24 KB	32 KB	40 KB	48 KB	32 KB	48 KB	60 KB
11	Ports	refer to *“Pin Functions” on page 32*							

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, to the data flash, and to external memory.

ROM The ROM consists of an internal flash memory. It is divided into code flash and data flash. For the available sizes, refer to *Table 1-1 on page 20*.

RAM For the available RAM sizes, refer to *Table 1-1 on page 20*.

DMA Controller The V850ES/Fx3 has a four-channel DMA Controller that transfers data between the internal RAM, on-chip peripheral I/O, data flash memory, and external memory, in response to interrupt requests from the on-chip peripheral I/O and external interrupts.

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

Clock Generator The Clock Generator generates the system clocks. It has four independent oscillators to ensure system operability if the main oscillator should fail and to provide low-speed clocks in power-save modes.

Clock Monitor The Clock Monitor monitors the main oscillator. In case of failure, it can switch the system to a different oscillator.

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.

Key Interrupt Function A key interrupt request signal can be generated by applying a falling edge to key input pins on eight channels.

UARTD The UARTs provide 2-wire Asynchronous Serial Interfaces.

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

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. This converter is of successive approximation type.

Motor Controller The Motor Controller uses the timers TAA4 and TAB0 to generate 3- or 6-phase pulse width modulated (PWM) signals for motor control.

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

Watch Timer The Watch Timer (WT) output forms the reference for the bookkeeping of daytime and calendar.

Watchdog Timer 2 The Watchdog Timer (WDT2) 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.

1.3.2 Structure of the manual

This manual explains how to use the V850ES/Fx3 microcontroller devices. It provides comprehensive information about the building blocks, their features, and how to set registers in order to enable or disable specific functions.

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

- Core functions

  "Pin Functions" on page 32
  "CPU System Functions" on page 155
  "Clock Generator" on page 179
  "Interrupt Controller (INTC)" on page 248
  "Key Interrupt Function" on page 296

- Memory access

  "Flash Memory" on page 298
  "Bus and Memory Control (BCU, MEMC)" on page 339
  "DMA Function (DMA Controller)" on page 373

- Timers

  "16-Bit Timer/Event Counter AA" on page 400
  "16-Bit Timer/Event Counter AB" on page 468
  "16-Bit Interval Timer M" on page 519
  "Watch Timer Functions" on page 527
  "Watchdog Timer 2" on page 533

- Serial interfaces

  "Asynchronous Serial Interface (UARTD)" on page 539
  "Clocked Serial Interface (CSIB)" on page 578
  "I^{2}C Bus (IIC)" on page 608
  "CAN Controller (CAN)" on page 674

- Control interfaces

  "A/D Converter (ADC)" on page 817
  "Motor Control Function" on page 856

- Power and reset

  "Power Supply Scheme" on page 916
  "Reset" on page 942
  "Low-Voltage Detector" on page 920

- Auxiliary functions

  "On-Chip Debug Unit" on page 930

1.4 Ordering Information

1.4.1 V850ES/FE3 ordering information

Part number	Package	On-chip flash memory	Quality grade[a]	Remark
UPD70F3370AM1GBA-GAH-AX	64-pin plastic LQFP (fine pitch) (10 x 10 mm^2)	128 KB	A	without Power-On-Clear circuit
UPD70F3370AM1GBA1-GAH-AX			A1	
UPD70F3370AM1GBA2-GAH-AX			A2	
UPD70F3370AM2GBA-GAH-AX			A	with Power-On-Clear circuit
UPD70F3370AM2GBA1-GAH-AX			A1	
UPD70F3370AM2GBA2-GAH-AX			A2	
UPD70F3371M1GBA-GAH-AX		256 KB	A	without Power-On-Clear circuit
UPD70F3371M1GBA1-GAH-AX			A1	
UPD70F3371M1GBA2-GAH-AX			A2	
UPD70F3371M2GBA-GAH-AX			A	with Power-On-Clear circuit
UPD70F3371M2GBA1-GAH-AX			A1	
UPD70F3371M2GBA2-GAH-AX			A2	

[a] The operating ambient temperature of each quality grades is as follows:
A: -40 to +85 °C, A1: -40 to +110 °C, A2: -40 to +125 °C

1.4.2 V850ES/FF3 ordering information

Part number	Package	On-chip flash memory	Quality grade[a]	Remark
UPD70F3372M1GKA-GAK-AX	80-pin plastic LQFP (fine pitch) (12 x 12 mm^2)	128 KB	A	without Power-On-Clear circuit
UPD70F3372M1GKA1-GAK-AX			A1	
UPD70F3372M1GKA2-GAK-AX			A2	
UPD70F3372M2GKA-GAK-AX			A	with Power-On-Clear circuit
UPD70F3372M2GKA1-GAK-AX			A1	
UPD70F3372M2GKA2-GAK-AX			A2	
UPD70F3373M1GKA-GAK-AX		256 KB	A	without Power-On-Clear circuit
UPD70F3373M1GKA1-GAK-AX			A1	
UPD70F3373M1GKA2-GAK-AX			A2	
UPD70F3373M2GKA-GAK-AX			A	with Power-On-Clear circuit
UPD70F3373M2GKA1-GAK-AX			A1	
UPD70F3373M2GKA2-GAK-AX			A2	

[a] The operating ambient temperature of each quality grades is as follows:
A: -40 to +85 °C, A1: -40 to +110 °C, A2: -40 to +125 °C

1.4.3 V850ES/FG3 ordering information

Part number	Package	On-chip flash memory	Quality grade[a]	Remark
UPD70F3374M1GCA)-UEU-AX	100-pin plastic LQFP (fine pitch) (14 x 14 mm^2)	128 KB	A	without Power-On-Clear circuit
UPD70F3374M1GCA1)-UEU-AX			A1	
UPD70F3374M1GCA2-UEU-AX			A2	
UPD70F3374M2GCA-UEU-AX			A	with Power-On-Clear circuit
UPD70F3374M2GCA1-UEU-AX			A1	
UPD70F3374M2GCA2-UEU-AX			A2	
UPD70F3375M1GCA-UEU-AX		256 KB	A	without Power-On-Clear circuit
UPD70F3375M1GCA1-UEU-AX			A1	
UPD70F3375M1GCA2-UEU-AX			A2	
UPD70F3375M2GCA-UEU-AX			A	with Power-On-Clear circuit
UPD70F3375M2GCA1-UEU-AX			A1	
UPD70F3375M2GCA2-UEU-AX			A2	
UPD70F3376AM1GCA-UEU-AX		384 KB	A	without Power-On-Clear circuit
UPD70F3376AM1GCA1-UEU-AX			A1	
UPD70F3376AM1GCA2-UEU-AX			A2	
UPD70F3376AM2GCA-UEU-AX			A	with Power-On-Clear circuit
UPD70F3376AM2GCA1-UEU-AX			A1	
UPD70F3376AM2GCA2-UEU-AX			A2	
UPD70F3377AM1GCA-UEU-AX		512 KB	A	without Power-On-Clear circuit
UPD70F3377AM1GCA1-UEU-AX			A1	
UPD70F3377AM1GCA2-UEU-AX			A2	
UPD70F3377AM2GCA-UEU-AX			A	with Power-On-Clear circuit
UPD70F3377AM2GCA1-UEU-AX			A1	
UPD70F3377AM2GCA2-UEU-AX			A2	

[a] The operating ambient temperature of each quality grades is as follows:
A: -40 to +85 °C, A1: -40 to +110 °C, A2: -40 to +125 °C

1.4.4 V850ES/FJ3 ordering information

Part number	Package	On-chip flash memory	Quality grade[a]	Remark
UPD70F3378M1GJA-GAE-AX	144-pin plastic LQFP (fine pitch) (20 x 20 mm^2)	256 KB	A	without Power-On-Clear circuit
UPD70F3378M1GJA1-GAE-AX			A1	
UPD70F3378M1GJA2-GAE-AX			A2	
UPD70F3378M2GJA-GAE-AX			A	with Power-On-Clear circuit
UPD70F3378M2GJA1-GAE-AX			A1	
UPD70F3378M2GJA2-GAE-AX			A2	
UPD70F3379M1GJA-GAE-AX		384 KB	A	without Power-On-Clear circuit
UPD70F3379M1GJA1-GAE-AX			A1	
UPD70F3379M1GJA2-GAE-AX			A2	
UPD70F3379M2GJA-GAE-AX			A	with Power-On-Clear circuit
UPD70F3379M2GJA1-GAE-AX			A1	
UPD70F3379M2GJA2-GAE-AX			A2	
UPD70F3380M1GJA-GAE-AX		512 KB	A	without Power-On-Clear circuit
UPD70F3380M1GJA1-GAE-AX			A1	
UPD70F3380M1GJA2-GAE-AX			A2	
UPD70F3380M2GJA-GAE-AX			A	with Power-On-Clear circuit
UPD70F3380M2GJA1-GAE-AX			A1	
UPD70F3380M2GJA2-GAE-AX			A2	
UPD70F3381M1GJA-GAE-AX		768 KB	A	without Power-On-Clear circuit
UPD70F3381M1GJA1-GAE-AX			A1	
UPD70F3381M1GJA2-GAE-AX			A2	
UPD70F3381M2GJA-GAE-AX			A	with Power-On-Clear circuit
UPD70F3381M2GJA1-GAE-AX			A1	
UPD70F3381M2GJA2-GAE-AX			A2	
UPD70F3382M1GJA-GAE-AX		1024 KB	A	without Power-On-Clear circuit
UPD70F3382M1GJA1-GAE-AX			A1	
UPD70F3382M1GJA2-GAE-AX			A2	
UPD70F3382M2GJA-GAE-AX			A	with Power-On-Clear circuit
UPD70F3382M2GJA1-GAE-AX			A1	
UPD70F3382M2GJA2-GAE-AX			A2	

[a] The operating ambient temperature of each quality grades is as follows:
A: -40 to +85 °C, A1: -40 to +110 °C, A2: -40 to +125 °C

1.4.5 V850ES/FK3 ordering information

Part number	Package	On-chip flash memory	Quality grade[a]	Remark
UPD70F3383M1GMA-GAR-AX	176-pin plastic LQFP (fine pitch) (24 x 24 mm^2)	512 KB	A	without Power-On-Clear circuit
UPD70F3383M1GMA1-GAR-AX			A1	
UPD70F3383M1GMA2-GAR-AX			A2	
UPD70F3383M2GMA-GAR-AX			A	with Power-On-Clear circuit
UPD70F3383M2GMA1-GAR-AX			A1	
UPD70F3383M2GMA2-GAR-AX			A2	
UPD70F3384M1GMA-GAR-AX		768 KB	A	without Power-On-Clear circuit
UPD70F3384M1GMA1-GAR-AX			A1	
UPD70F3384M1GMA2-GAR-AX			A2	
UPD70F3384M2GMA-GAR-AX			A	with Power-On-Clear circuit
UPD70F3384M2GMA1-GAR-AX			A1	
UPD70F3384M2GMA2-GAR-AX			A2	
UPD70F3385M1GMA-GAR-AX		1024 KB	A	without Power-On-Clear circuit
UPD70F3385M1GMA1-GAR-AX			A1	
UPD70F3385M1GMA2-GAR-AX			A2	
UPD70F3385M2GMA-GAR-AX			A	with Power-On-Clear circuit
UPD70F3385M2GMA1-GAR-AX			A1	
UPD70F3385M2GMA2-GAR-AX			A2	

[a] The operating ambient temperature of each quality grades is as follows:
A: -40 to +85 °C, A1: -40 to +110 °C, A2: -40 to +125 °C

Chapter 2 Pin Functions

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

2.1 Overview

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

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

For a description of the terms pin, port or port group, see *"Terms" on page 37*.

Features summary

- Number of ports and port groups:

V850ES/	FE3	FF3	FG3	FJ3	FK3
Port groups	8	10	11	15	17
I/O ports	51	67	84	128	152

- Configuration possible for individual pins.
- For many pins, the connection of a pull-up resistor can be selected.

2.1.1 Description

The V850ES/FE3, V850ES/FF3, and V850ES/FG3 microcontrollers have the port groups shown below.

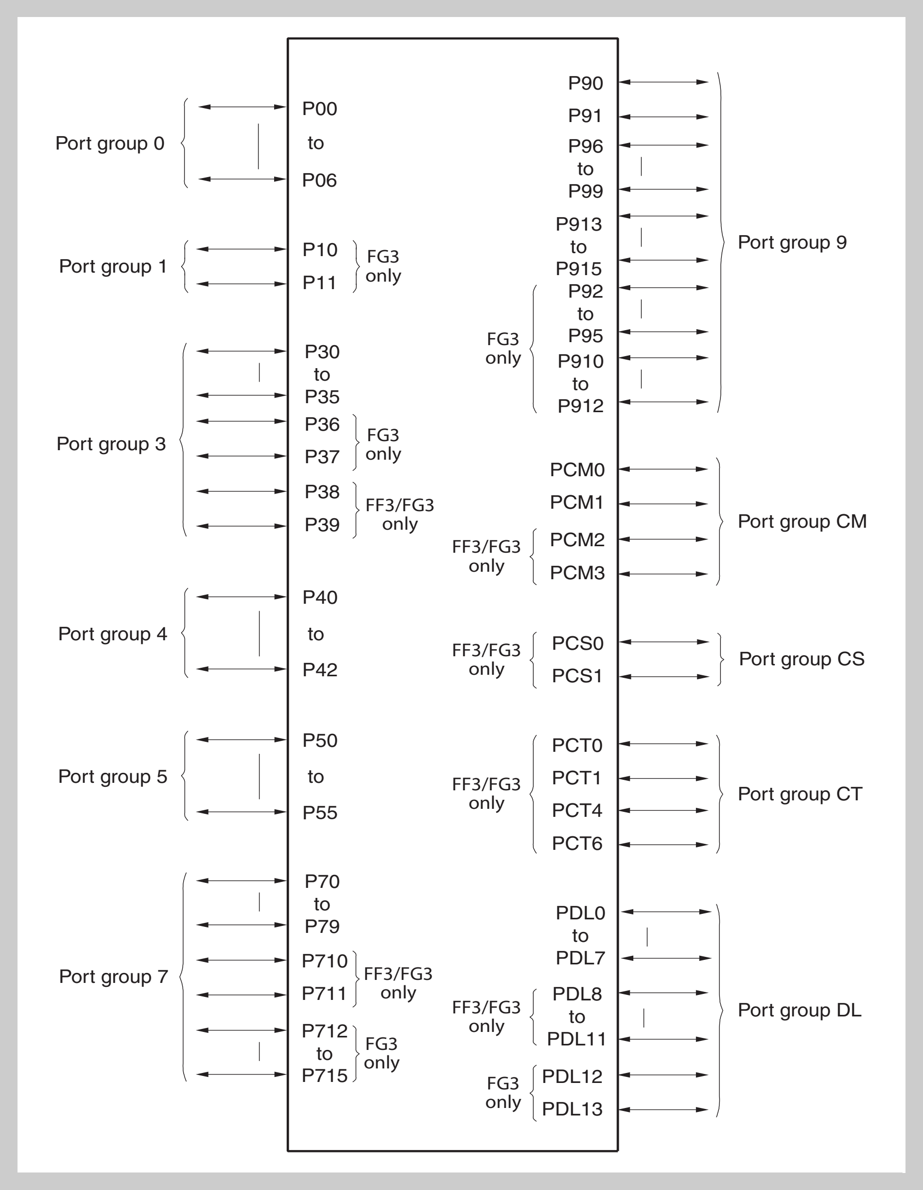


Figure 2-1 V850ES/FE3, V850ES/FF3, V850ES/FG3 port groups

The V850ES/FJ3 and V850ES/FK3 microcontrollers have the port groups shown below.

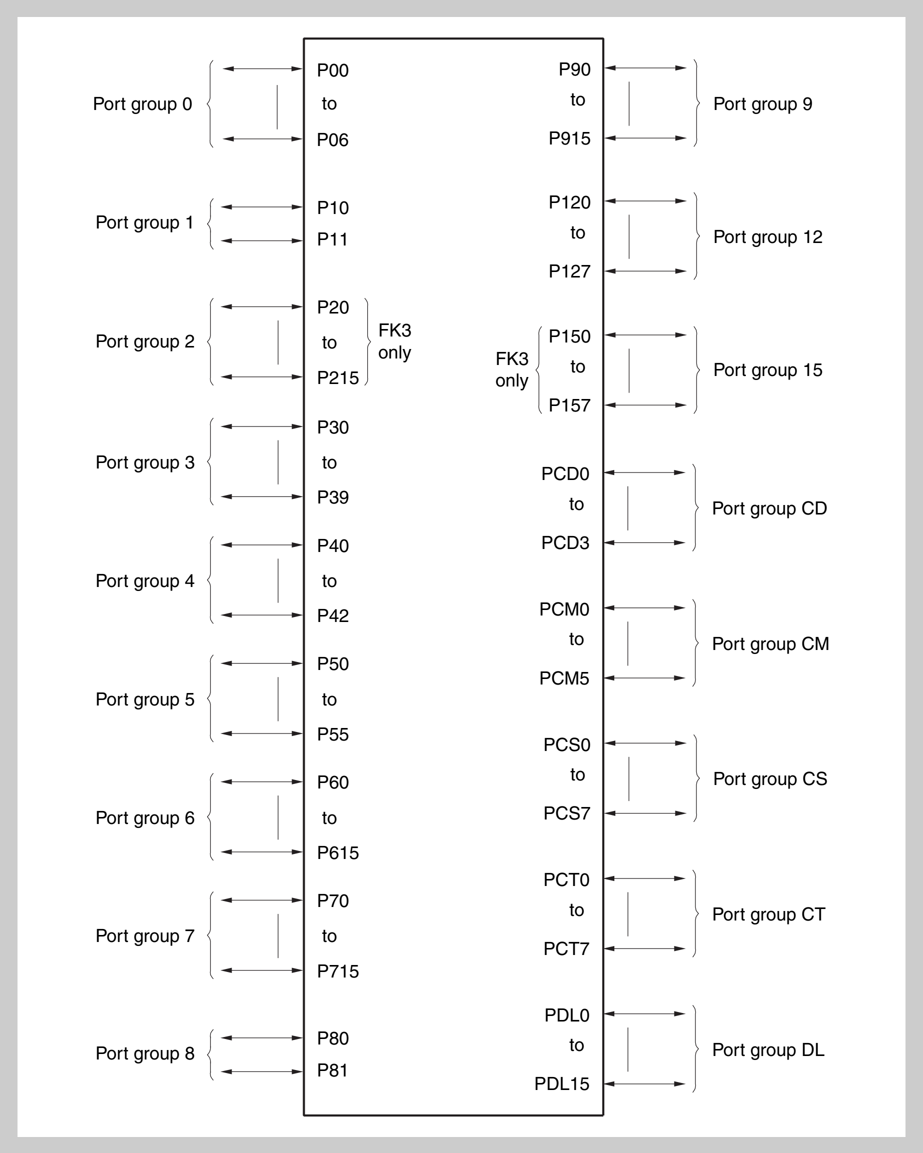


Figure 2-2 V850ES/FJ3, V850ES/FK3 port groups

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.

Note Not all port groups and functions in *Table 2-1* are available for all products of the V850ES/Fx3 product line. For detailed information which port groups and functions are available for a dedicated product, refer to *“2.4 Port Type Diagrams”* .

Table 2-1 Functions of each port group (1/2)

Port group name	Function	
	Port mode	Alternative mode
0	7-bit input/output	• External interrupt 0 to 3 • Non-maskable interrupt • N-Wire debug interface reset • A/D Converter 0 external trigger input • Timer TAA3 channels • Timer TAA4 channels • CAN0 transmit/receive data
1	2-bit input/output	• External interrupt 9 and 10
2	16-bit input/output	• A/D Converter 1 inputs
3	10-bit input/output	• External interrupt 7 and 8 • Timer TAA0 channels • Timer TAA1 channels • CAN0 transmit/receive data • CAN1 transmit/receive data • UARTD0 transmit/receive data • UARTD0 baud rate clock input • UARTD2 transmit/receive data
4	3-bit input/output	• External interrupt 14 • Key interrupt input 0 to 2 • Clocked Serial Interface CSIB0 data/clock line • UARTD3 transmit/receive data
5	6-bit input/output	• Key interrupt input 0 to 5 • N-Wire debug interface signals • Timer TAB0 channels • Motor control channels
6	16-bit input/output	• External interrupt 11 to 13, 15 • Timer TAB2 channels • Clocked Serial Interface CSIB3 data/clock line • CAN2 transmit/receive data • CAN3 transmit/receive data • UARTD6 transmit/receive data • UARTD7 transmit/receive data • A/D Converter 1 external trigger input
7	16-bit input/output	• A/D Converter 0 inputs
8	2-bit input/output	• External interrupt 14 • UARTD3 transmit/receive data

Table 2-1 Functions of each port group (2/2)

Port group name	Function	
	Port mode	Alternative mode
9	16-bit input/output	• External interrupt 4 to 6 • Key interrupt input 6 to 7 • Timer TAA2 channels • Timer TAB0 channels • Timer TAB1 channels • Clocked Serial Interface CSIB1 data/clock line • Clocked Serial Interface CSIB2 data/clock line • UARTD1 transmit/receive data • UARTD4 transmit/receive data • UARTD5 transmit/receive data • CAN2 transmit/receive data • I^2C data/clock line • Programmable clock output
12	8-bit input/output	• A/D Converter 0 inputs
15	8-bit input/output	• Timer TAA5 channels • Timer TAA6 channels • Timer TAA7 channels • CAN4 transmit/receive data
CD	4-bit input/output	–
CM	6-bit input/output	• External memory interface data wait request • CPU system clock output • External memory interface bus hold request input • External memory interface bus hold acknowledge output
CS	8-bit input/output	• External memory interface chip select signals
CT	8-bit input/output	• External memory interface read/write/address strobe
DL	16-bit input/output	• External memory interface address/data lines 0 to 15

Pin configuration

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

- For a general description of the registers, see *“Port Group Configuration Registers” on page 38*.
- For every port, detailed information on the configuration registers is given in *“Port Type Diagrams” on page 52*.

2.1.2 Terms

In this section, the following terms are used:

- **Pin**

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

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

- **Port group**

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

- **Port mode / Port**

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

  The corresponding name is Pnm. For example, P04 denotes port 4 of port group 0. It is referenced as “port P04”.

- **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 P03 and INTP0 denote the same physical pin. The different names indicate the function in which the pin is being operated.

- **Port type**

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

2.1.3 Noise elimination

The input signals at some pins are passing a filter to remove noise and glitches. The microcontroller supports both analog and digital filters.

See *“Noise Elimination” on page 144* for a detailed description.

2.2 Port Group Configuration Registers

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

- *“Pin function configuration” on page 39*
- *“Pin data input/output” on page 45*
- *“Configuration of pull-up resistors” on page 47*
- *“Open drain configuration” on page 48*

2.2.1 Overview

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

Table 2-2 Registers for port group configuration

Register name	Shortcut	Function
Port mode control register	PMCn	Pin function configuration
Port mode register	PMn	
Port function control register	PFCn	
Port function control expansion register	PFCEn	
On-chip debug mode register	OCDM	
Port register	Pn	Pin data input/output
Pull-up resistor option register	PUn	Configuration of pull-up resistors
Port function register	PFn	Open drain configuration

n = 0 to 9, 12, 15, CD, CM, CS, CT, DL

2.2.2 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
- normal mode or on-chip debug mode (N-Wire interface)

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

Table 2-3 Pin function configuration (overview)

Function	Registers					I/O
	OCDM	PMC	PM	PFCE	PFC	
Port mode (output)	0	0	0	X	X	O
Port mode (input)			1	X	X	I
Alternative mode (alternative function 1)		1	X	0	0	I/O[a]
Alternative mode (alternative function 2)					1	
Alternative mode (alternative function 3)				1	0	
Alternative mode (alternative function 4)					1	
On-chip debug mode[b]	1	X	X	X	X	I/O

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

[b] In on-chip debug mode, the corresponding pins are automatically set as input or output pins to provide the N-Wire interface. In this mode, the configuration of these pins can not be changed by the pin configuration registers. Refer to chapter *“On-Chip Debug Unit” on page 930* for details.

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

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

Address see *“Port Type Diagrams” on page 52*

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

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

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

Table 2-4 PMCn register contents

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

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

Therefore, it is recommended to follow the below procedure:

1. To select the alternative input function INTPn (I), set PFCE.PFCEnm and PFC.PFCnm accordingly.
2. Set PMCnm = 1 to change to the alternative mode.
3. Wait until the delay of the noise elimination filter has passed.
4. Set INTnIC.INTnIF = 0 to clear the interrupt request.
5. 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 elimination 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

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

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

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.
16-bit registers can also be read/written in 16-bit units.

Address see *"Port Type Diagrams" on page 52*

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

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

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

Table 2-5 PMn register contents

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

(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 PFCn register together with the PFCEn register specifie 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.
16-bit registers can also be read/written in 16-bit units.

Address see *“Port Type Diagrams” on page 52*

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

PFCn

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

PFCn

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

Table 2-6 PFCn register contents

Bit position	Bit name	Function
7 to 0	PFCn[7:0]	See *“Pin function configuration (overview)” on page 39* for details
15 to 0	PFCn[15:0]	See *“Pin function configuration (overview)” on page 39* for details

(4) PFCEn - Port function control expansion register

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

The PFCEn together with the PFCn 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.
16-bit registers can also be read/written in 16-bit units.

Address see *“Port Type Diagrams” on page 52*

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

PFCEn

7	6	5	4	3	2	1	0
PFCEn7	PFCEn6	PFCEn5	PFCEn4	PFCEn3	PFCEn2	PFCEn1	PFCEn0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

PFCEn

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

Table 2-7 PFCEn register contents

Bit position	Bit name	Function
7 to 0	PFCEn[7:0]	See *“Pin function configuration (overview)” on page 39* for details
15 to 0	PFCEn[15:0]	See *“Pin function configuration (overview)” on page 39* for details

(5) OCDM - On-chip debug mode register

The 8-bit OCDM register specifies whether dedicated pins of the microcontroller operate in normal operation mode or can be used for on-chip debugging (N-Wire interface). The setting of this register concerns only those pins that can be used for the N-Wire interface: P05/$\overline{\text{DRST}}$, P52/DDI, P53/DDO, P54/DCK, and P55/DMS.

To make these pins available for on-chip debugging, bit OCDM.OCDM0 must be set while pin $\overline{\text{DRST}}$ is high. If the on-chip debug mode is selected, the corresponding pins are automatically set as input or output pins, respectively. Setting of bits PMn.PMnm is not necessary.
For more details refer to *“On-Chip Debug Unit” on page 930*.

Writing to this register is protected by a special sequence of instructions. Please refer to *“CPU System Functions” on page 155* for details.

Access This register can be read/written in 8-bit and 1-bit units.
The register can only be written if a low level (’0’) is input to the P05/$\overline{\text{DRST}}$ pin.

Address FFFF F9FC$_H$

Initial Value 00$_H$/01$_H$:

- After Power-On-Clear reset, the normal operation mode is selected (OCDM.OCDM0 = 0).
- After external $\overline{\text{RESET}}$, the dedicated pins are available for on-chip debugging (OCDM.OCDM0 = 1).
- After any other reset, bit OCDM0 holds the same value as before the reset

	7	6	5	4	3	2	1	0
OCDM	0	0	0	0	0	0	0	OCDM0
	R	R	R	R	R	R	R	R/W

Table 2-8 OCDM register contents

Bit position	Bit name	Function
0	OCDM0	Enables/disables N-Wire interface: 0: Pins are used in normal operation mode (port mode or alternative mode). $\overline{\text{DRST}}$ pull-down resistor not connected 1: Pins are used in on-chip debug mode. $\overline{\text{DRST}}$ pull-down resistor connected

Note If the pins P05/$\overline{\text{DRST}}$, P52/DDI, P53/DDO, P54/DCK, and P55/DMS are used as N-Wire interface pins their configuration can not be changed by the pin configuration registers.

$\overline{\text{DRST}}$ pull-down resistor $\overline{\text{DRST}}$ (P05) is equipped with an internal pull-down resistor. Connection of the resistor is controlled by OCDM.OCDM0:

0: resistor detached from P05/$\overline{\text{DRST}}$

1: resistor attached to P05/$\overline{\text{DRST}}$

This ensures that the microcontroller is operating correctly, even if the pins are in N-Wire mode, but no debugger is connected.

2.2.3 Pin data input/output

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

(1) Pn - Port register

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

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

Address see *“Port Type Diagrams” on page 52*

Initial Value Undefined.

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
Pn	Pn7	Pn6	Pn5	Pn4	Pn3	Pn2	Pn1	Pn0
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

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

Table 2-9 Pn register contents

Bit position	Bit name	Function
7 to 0 or 15 to 0	Pn[7:0] or Pn[15:0]	Data, see *Table 2-10 on page 45* and *Table 2-11 on page 46* 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 2-10 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 2-11 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”.

2.2.4 Configuration of pull-up resistors

(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.
16-bit registers can also be read/written in 16-bit units.

Address see *"Port Type Diagrams" on page 52*

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

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PUn	PUn15	PUn14	PUn13	PUn12	PUn11	PUn10	PUn9	PUn8	PUn7	PUn6	PUn5	PUn4	PUn3	PUn2	PUn1	PUn0
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 2-12 PUn register contents

Bit position	Bit name	Function
7 to 0 or 15 to 0	PUn[7:0] or PUn[15:0]	Specifies whether a pull-up resistor is connected to the corresponding pin: 0: no pull-up resistor connected 1: pull-up resistor connected

Caution In Port mode, (PMCnm bit = 0), the PUnm bit of the PUn register is valid only when PMnm bit of PMn register is 1 (input mode). If PMnm bit = 0 (output mode), the setting value of PUn register is invalid (pull-up resistor is detached).

2.2.5 Open drain configuration

(1) PFn - Port function register

If a pin is in alternative mode (PMCn.PMCnm = 1), the PFn register specifies normal output or open-drain output.

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

Note The settings of PFn are only valid in alternative mode.

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

Address see *“Port Type Diagrams” on page 52*

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

	7	6	5	4	3	2	1	0
PFn	PFn7	PFn6	PFn5	PFn4	PFn3	PFn2	PFn1	PFn0
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PFn	PFn15	PFn14	PFn13	PFn12	PFn11	PFn10	PFn9	PFn8	PFn7	PFn6	PFn5	PFn4	PFn3	PFn2	PFn1	PFn0
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 2-13 PFn register contents

Bit position	Bit name	Function
7 to 0 or 15 to 0	PFn[7:0] or PFn[15:0]	Specifies normal output or open-drain output 0: Normal output 1: Open-drain output

2.3 Port Buffers Diagrams

This chapter presents the block diagrams of all buffer types.

The tables in *"Port group configuration lists" on page 99* informs also about the buffer type, used for each port.

(1) Buffer type 2

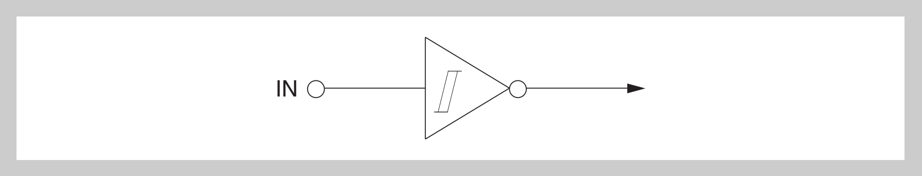


Figure 2-3 Block diagram: buffer type 2

(2) Buffer type 5

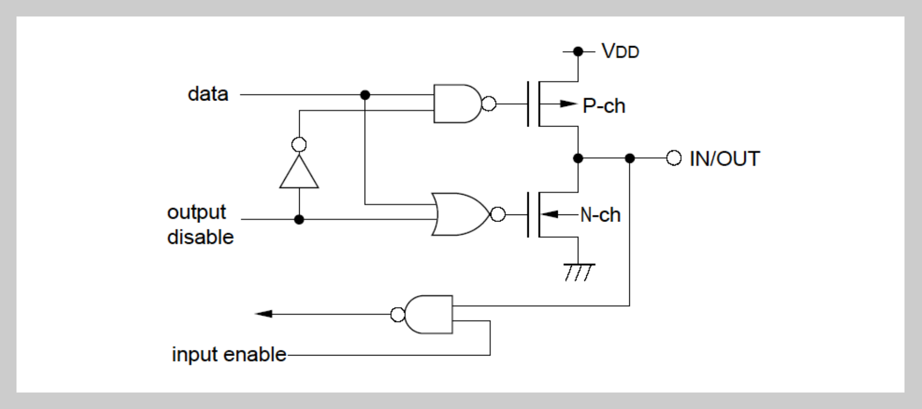


Figure 2-4 Block diagram: buffer type 5

(3) Buffer type 5-AF

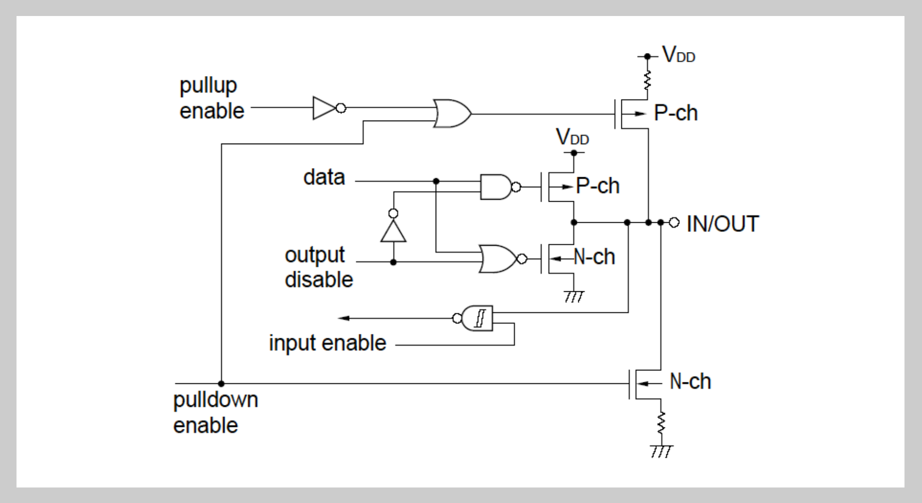


Figure 2-5 Block diagram: buffer type 5-AF

(4) Buffer type 5-K

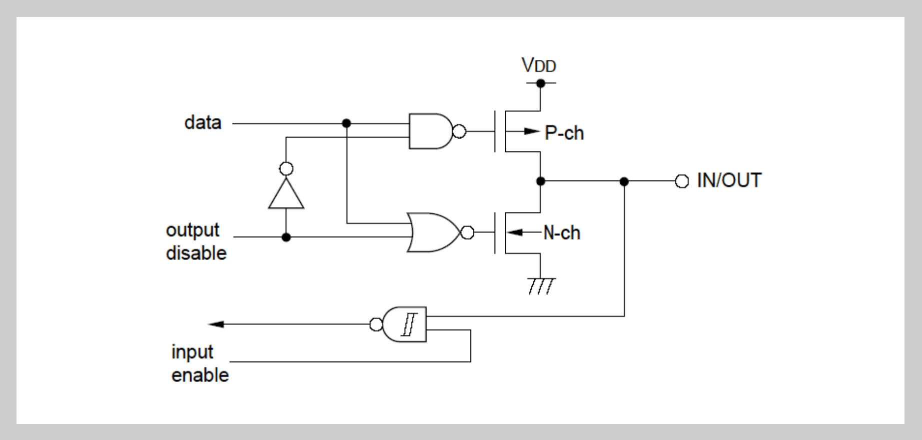


Figure 2-6 Block diagram: buffer type 5-K

(5) Buffer type 5-W

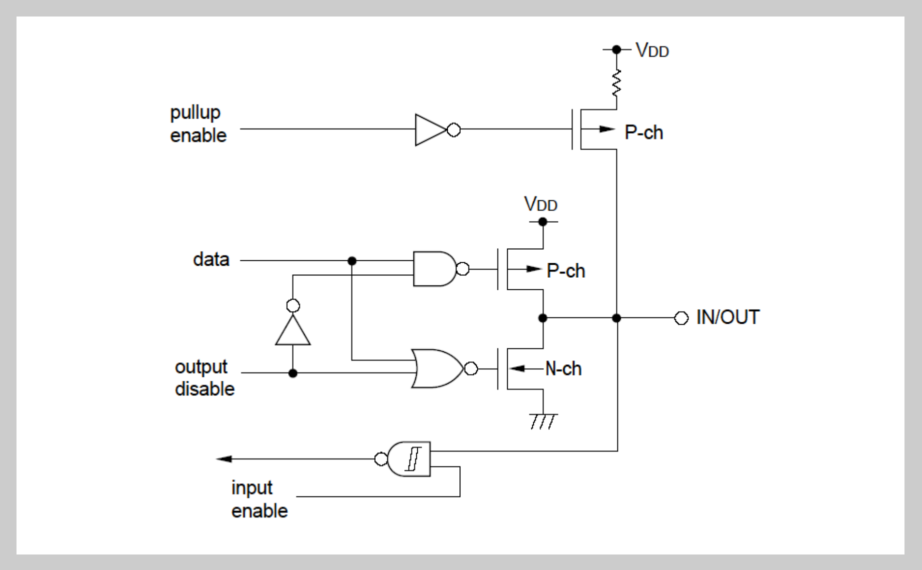


Figure 2-7 Block diagram: buffer type 5-W

(6) Buffer type 11-G

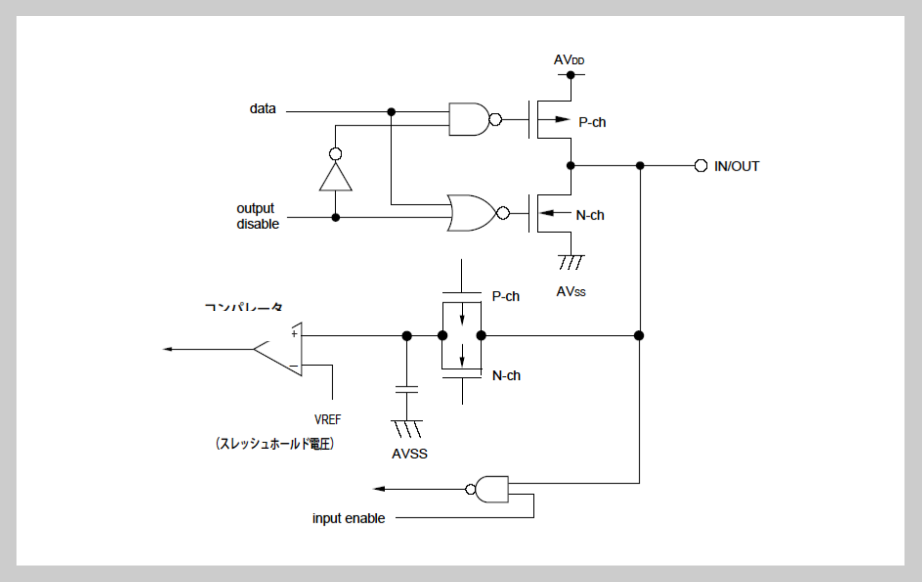


Figure 2-8 Block diagram: buffer type 11-G

(7) Buffer type 16

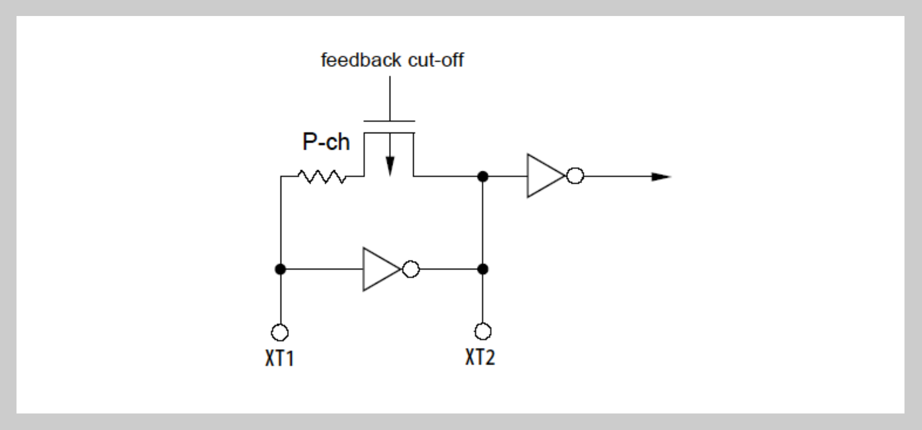


Figure 2-9 Block diagram: buffer type 16

2.4 Port Type Diagrams

This chapter presents the block diagrams of all port types.

The tables in the detailed descriptions of each port group from *“Port group 0” on page 114* onwards informs also about the port type, used for each port.

2.4.1 Port type C

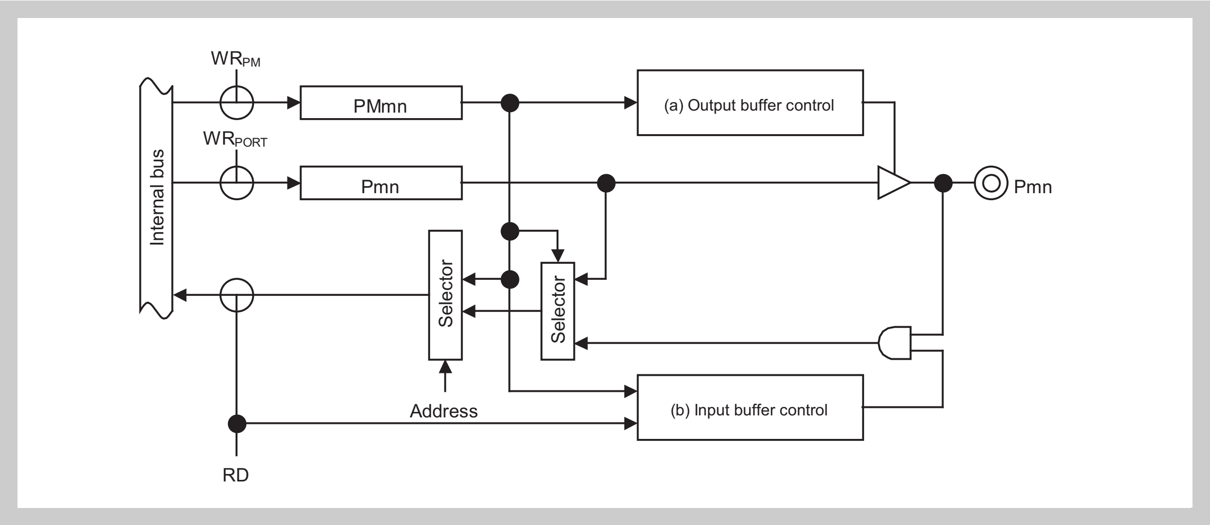


Figure 2-10 Port type C block diagram

2.4.2 Port type C-U

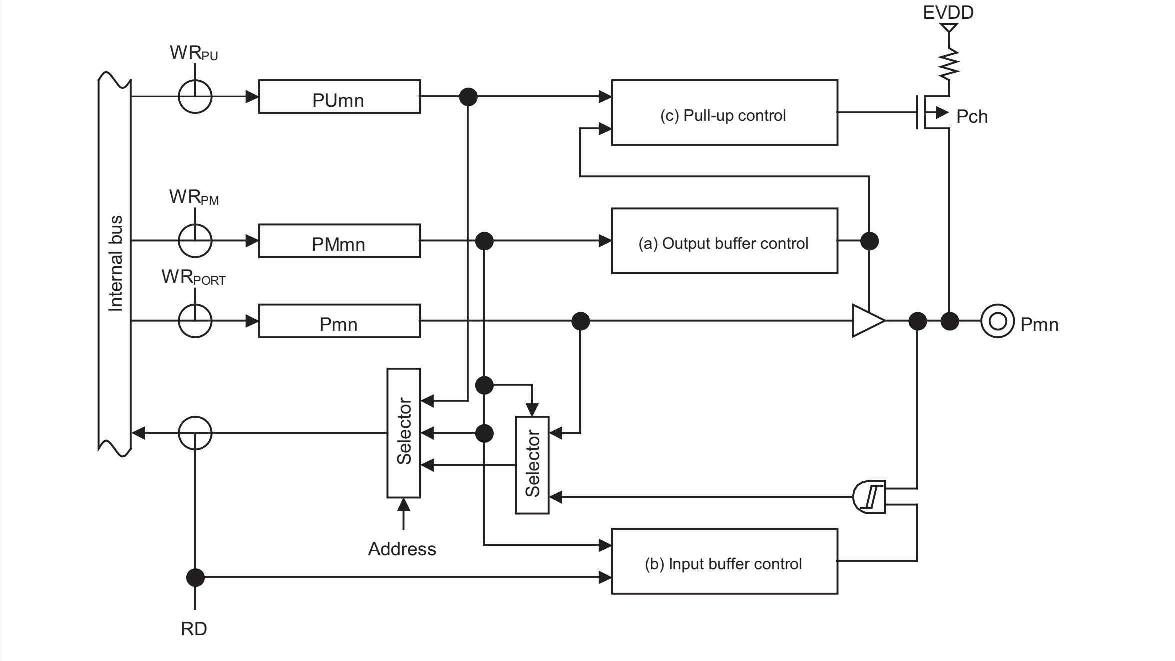


Figure 2-11 Port type C-U block diagram

2.4.3 Port type D0

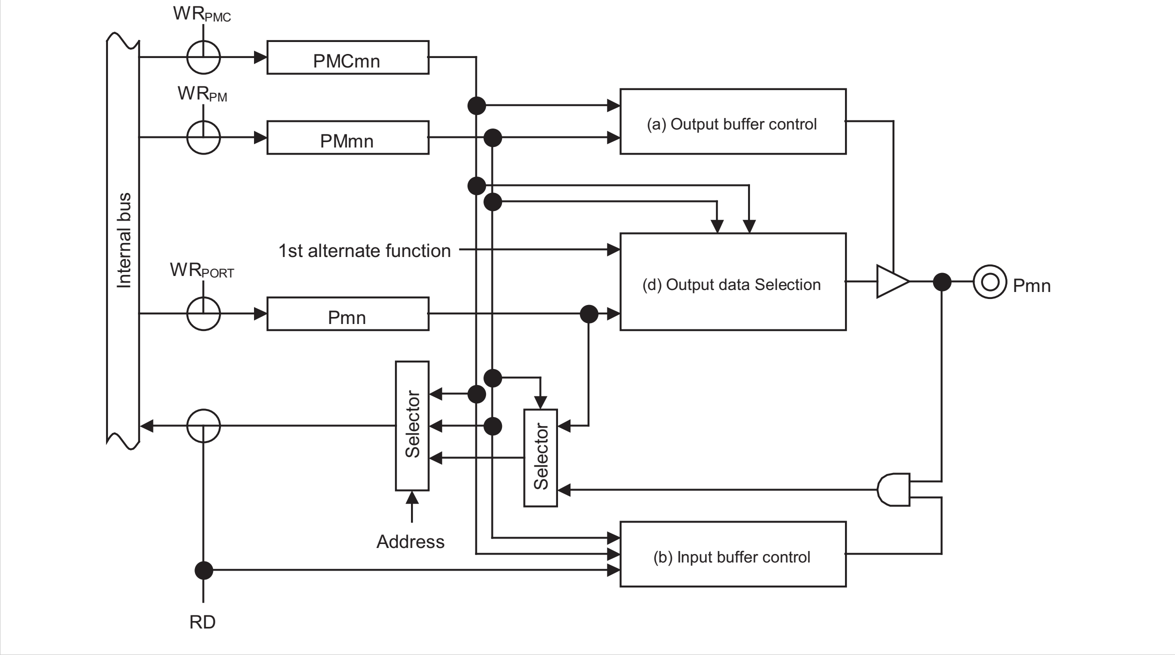


Figure 2-12 Port type D0 block diagram

2.4.4 Port type D0-U

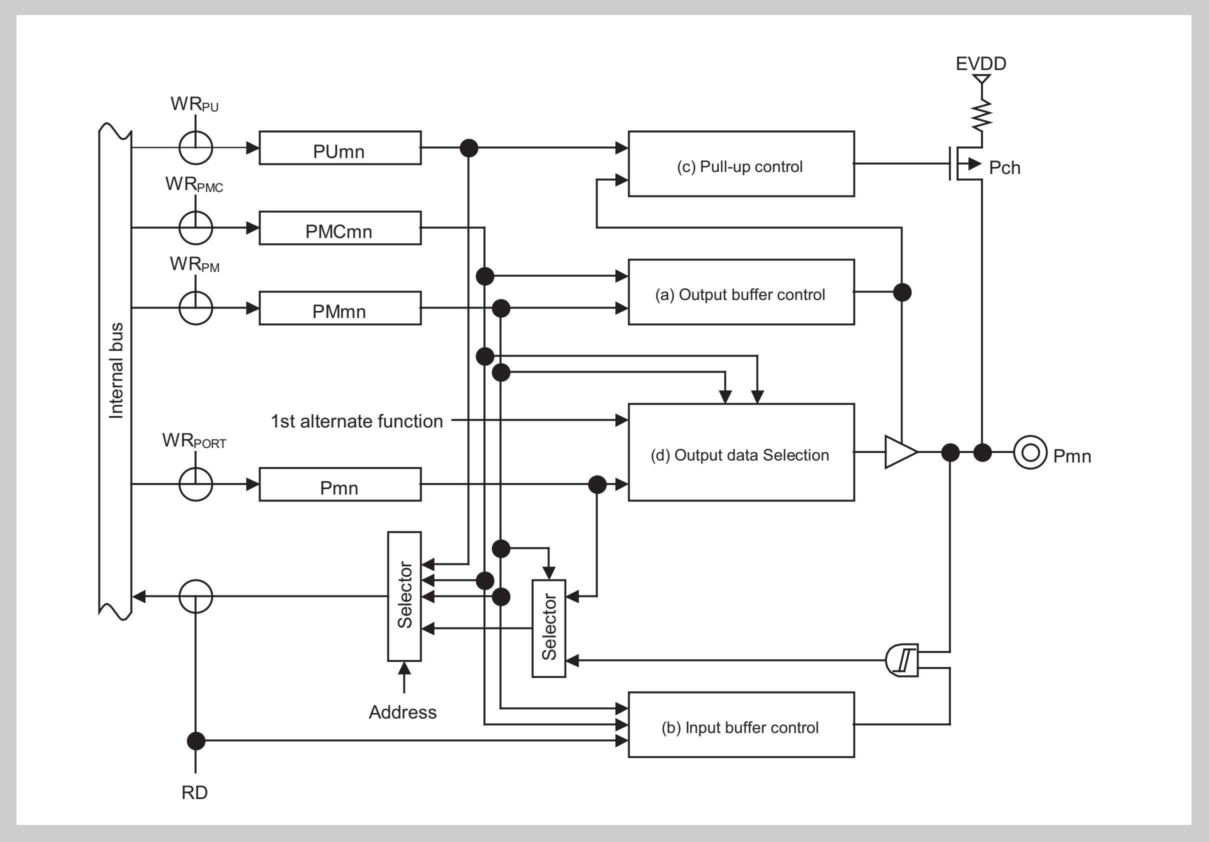


Figure 2-13 Port type D0-U block diagram

2.4.5 Port type D1

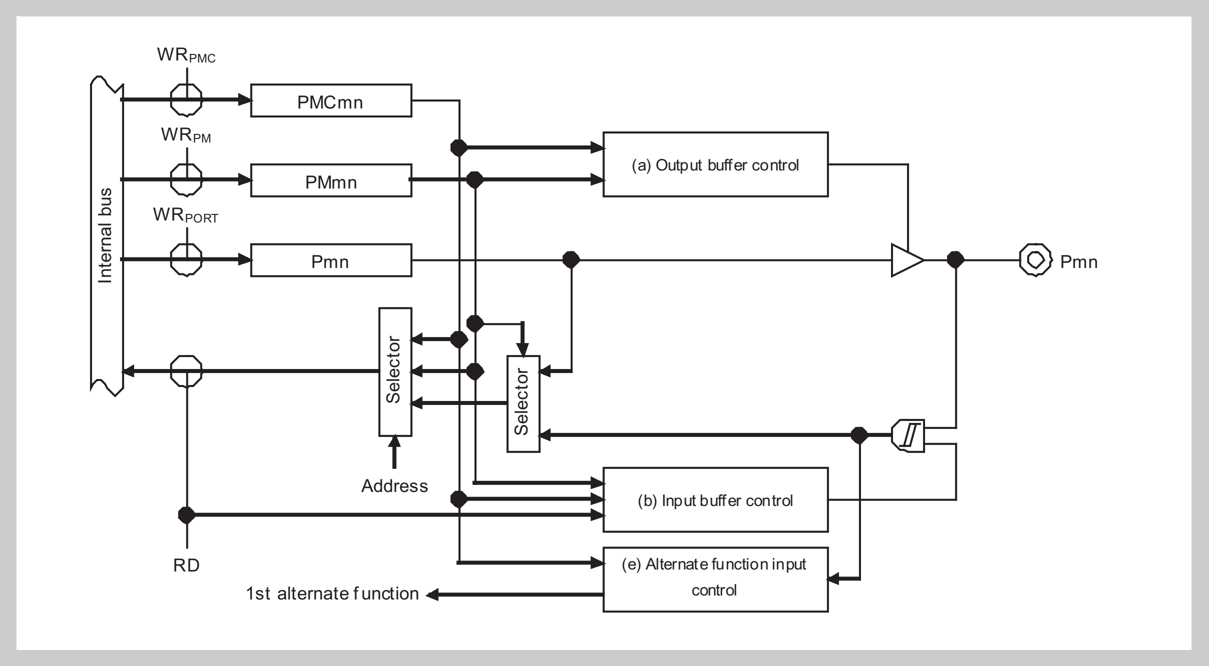


Figure 2-14 Port type D1 block diagram

2.4.6 Port type D1-U

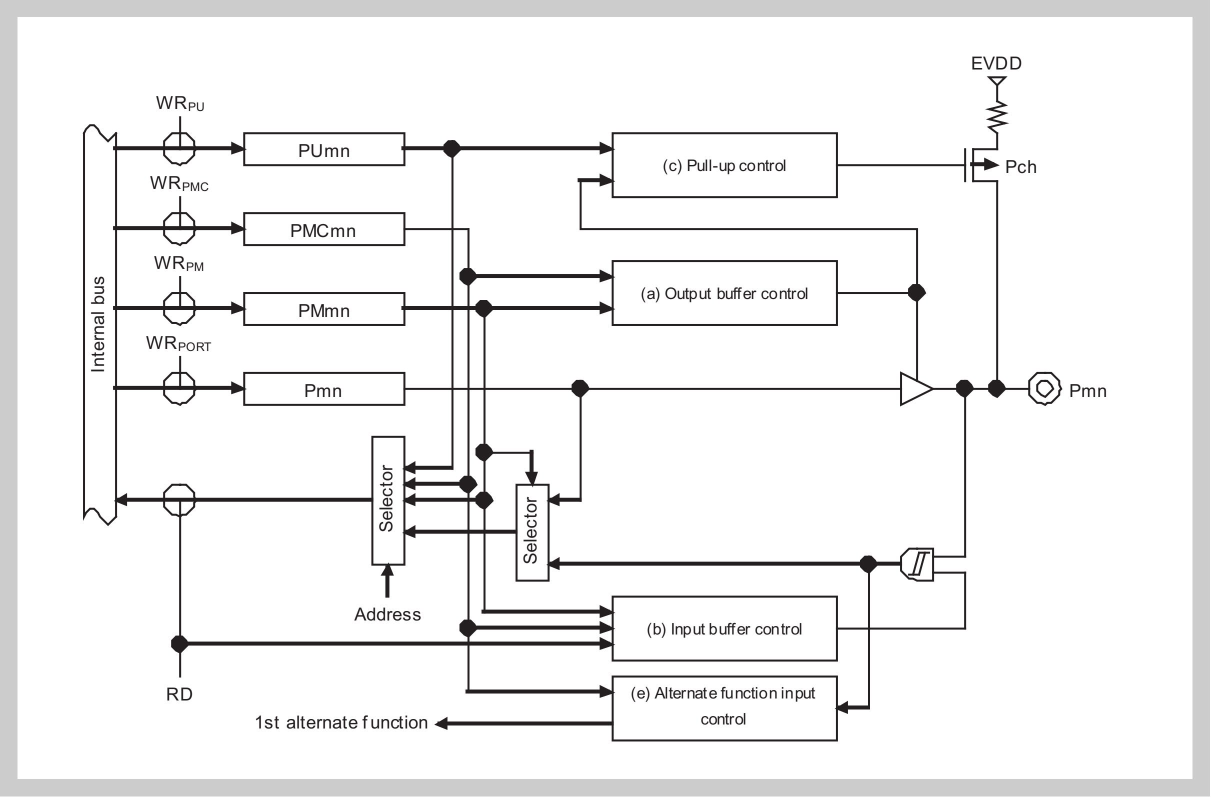


Figure 2-15 Port type D1-U block diagram

Note For V850ES/FK3 products the ADTRG1 input features an analog noise rejection filter.

2.4.7 Port type D1-UI

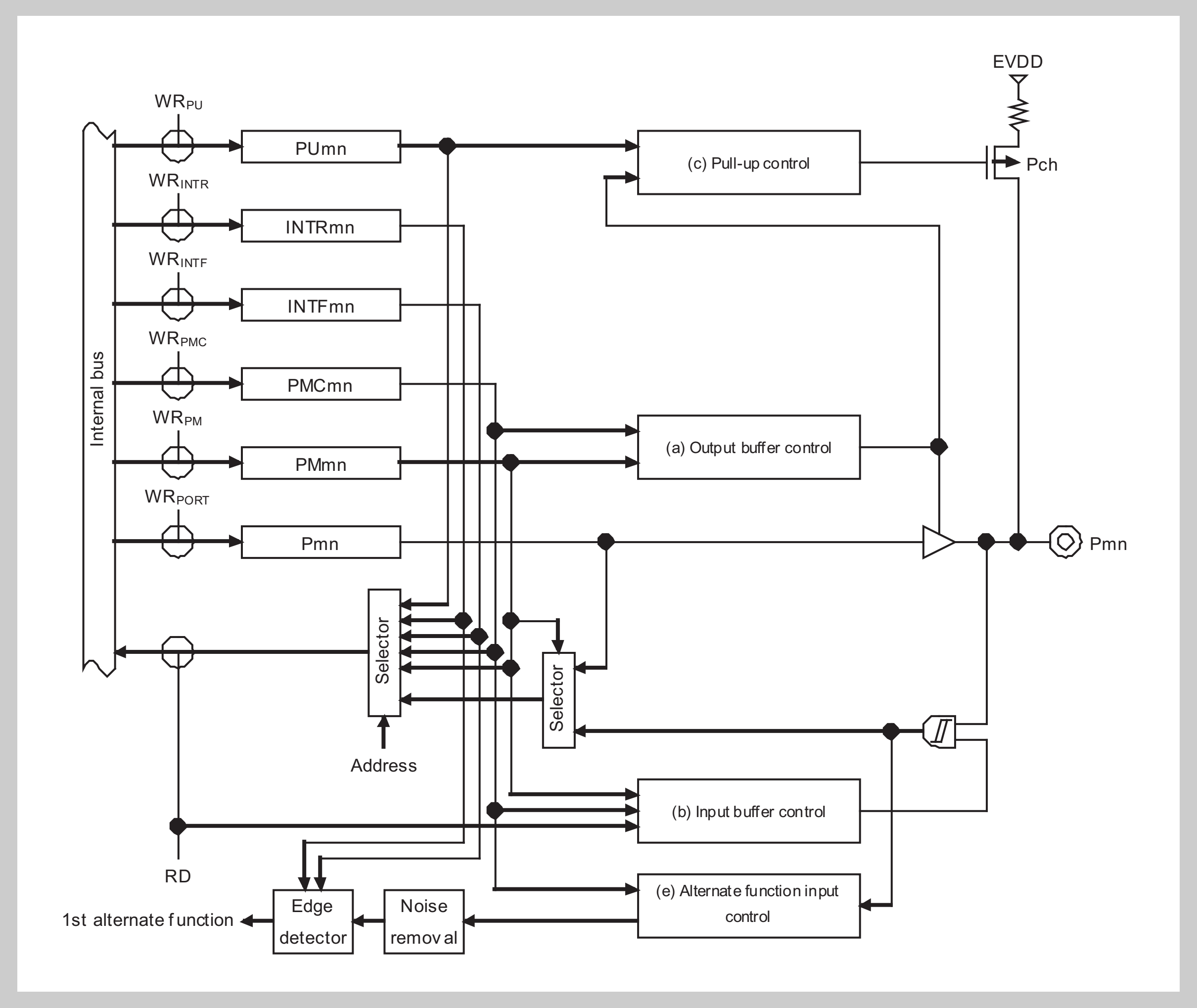


Figure 2-16 Port type D1-UI block diagram

2.4.8 Port type D3-UI

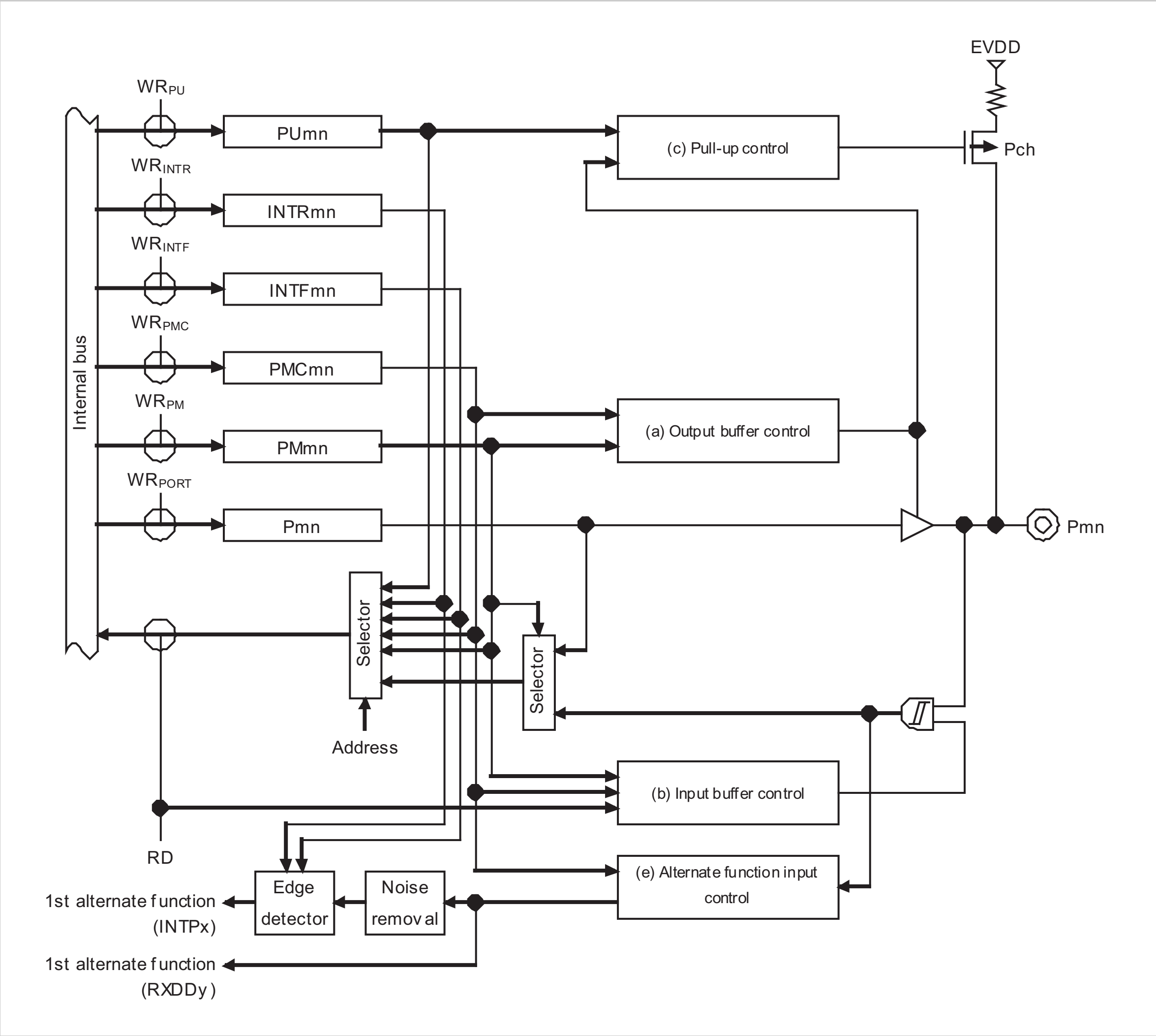


Figure 2-17 Port type D3-UI block diagram

2.4.9 Port type D1A

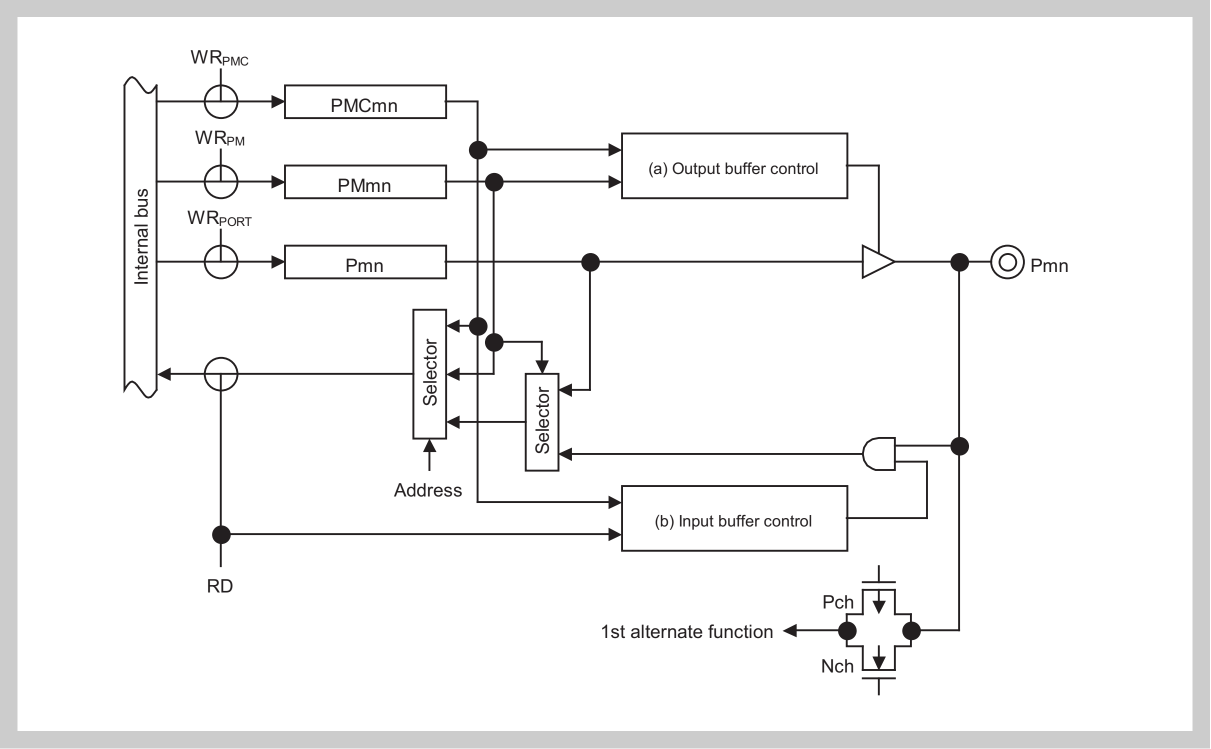


Figure 2-18 Port type D1A block diagram

2.4.10 Port type D1O1-UI

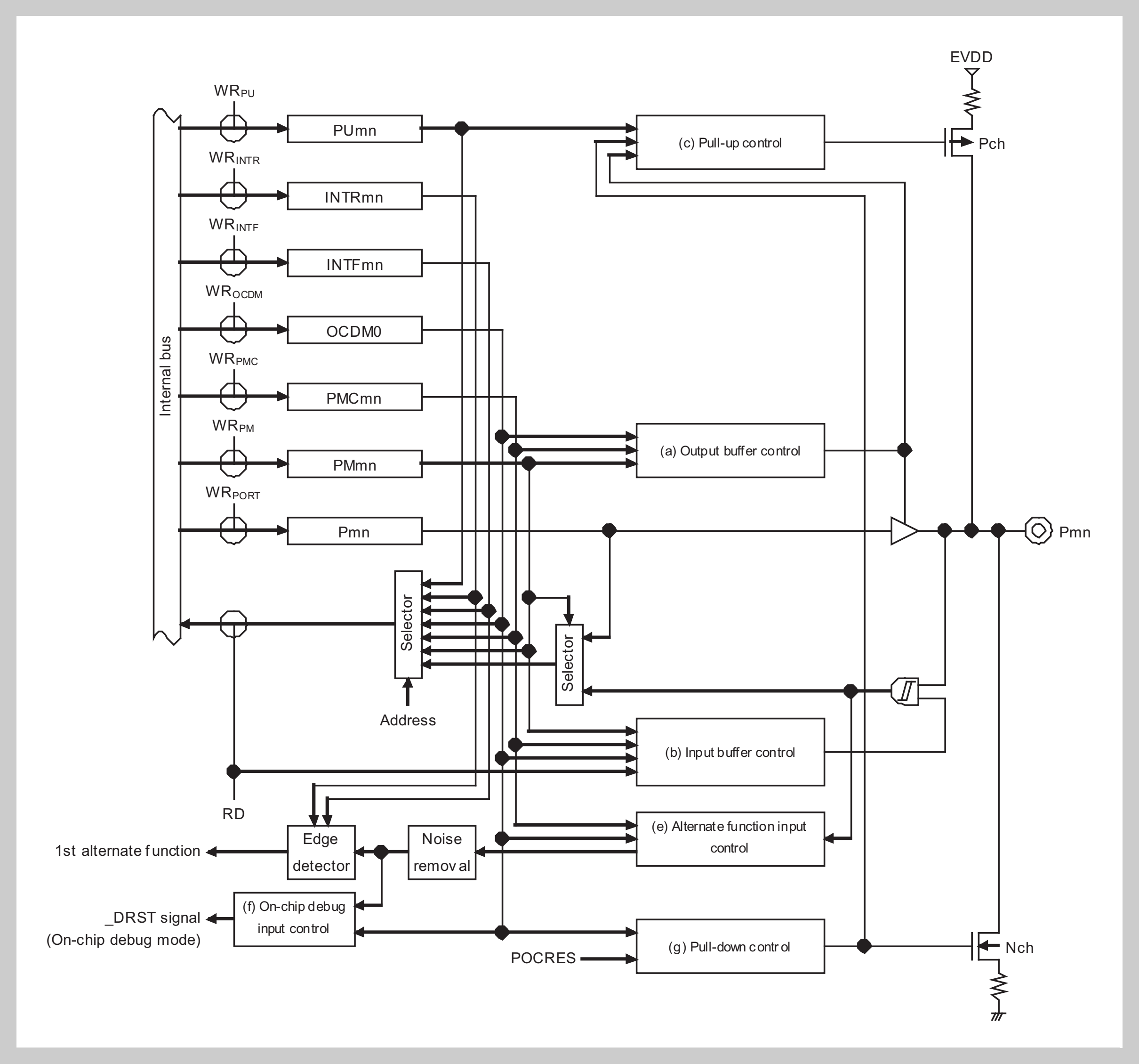


Figure 2-19 Port type D1O1-UI block diagram

2.4.11 Port type D2

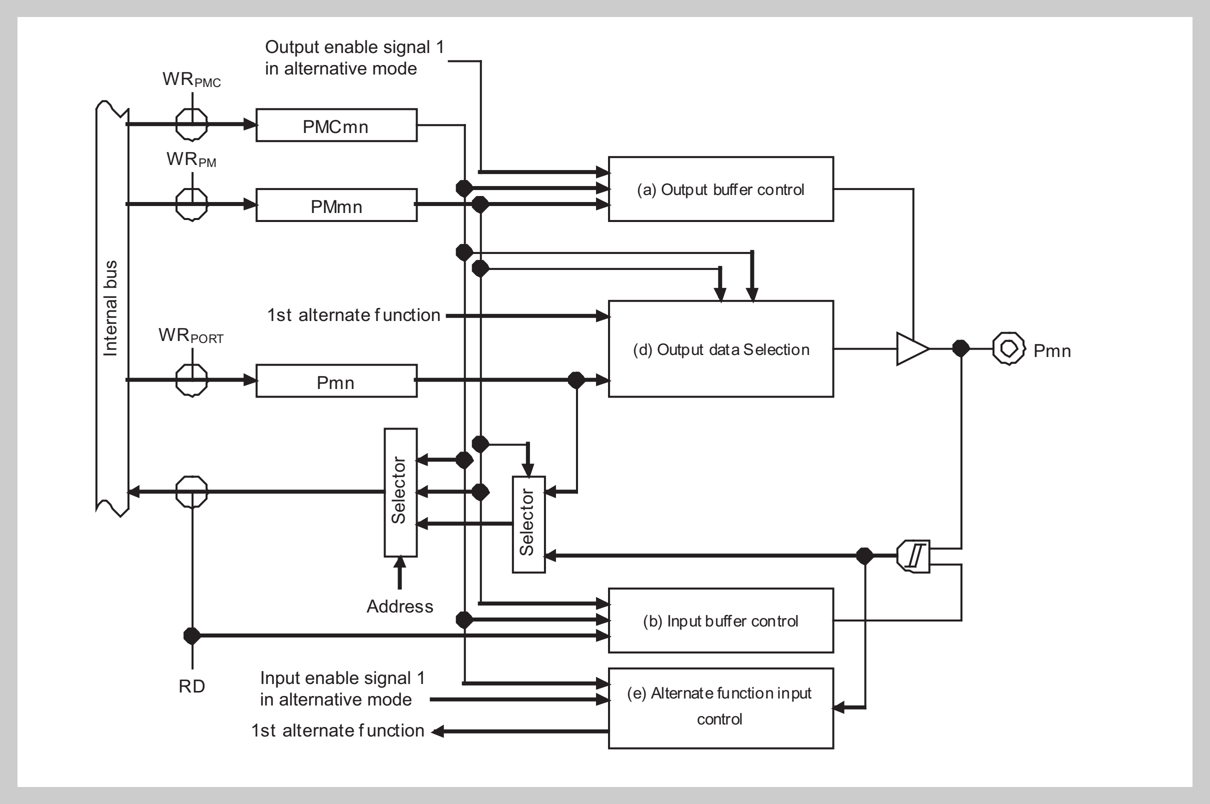


Figure 2-20 Port type D2 block diagram

2.4.12 Port type E01-U

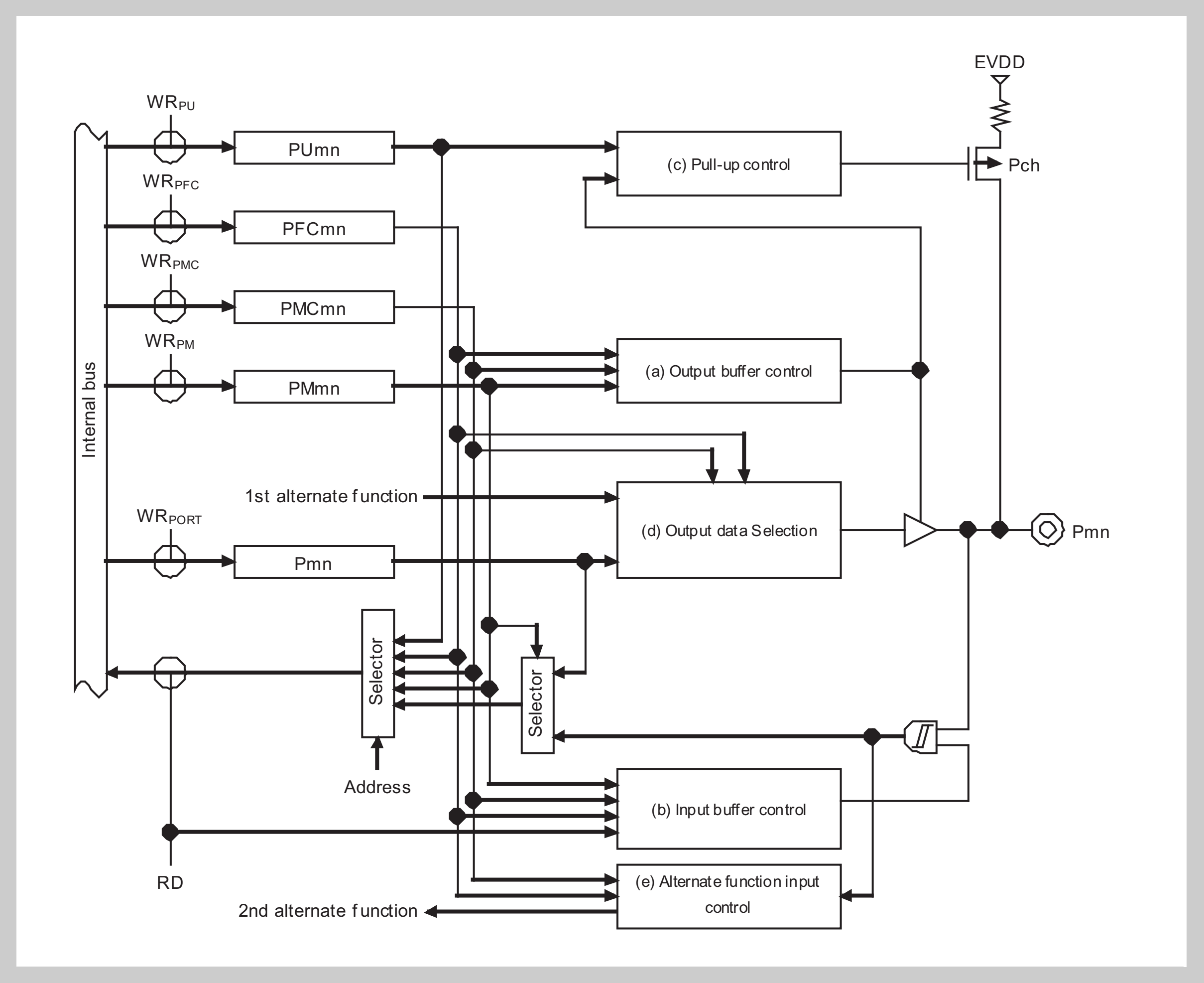


Figure 2-21 Port type E01-U block diagram

2.4.13 Port type E10-U

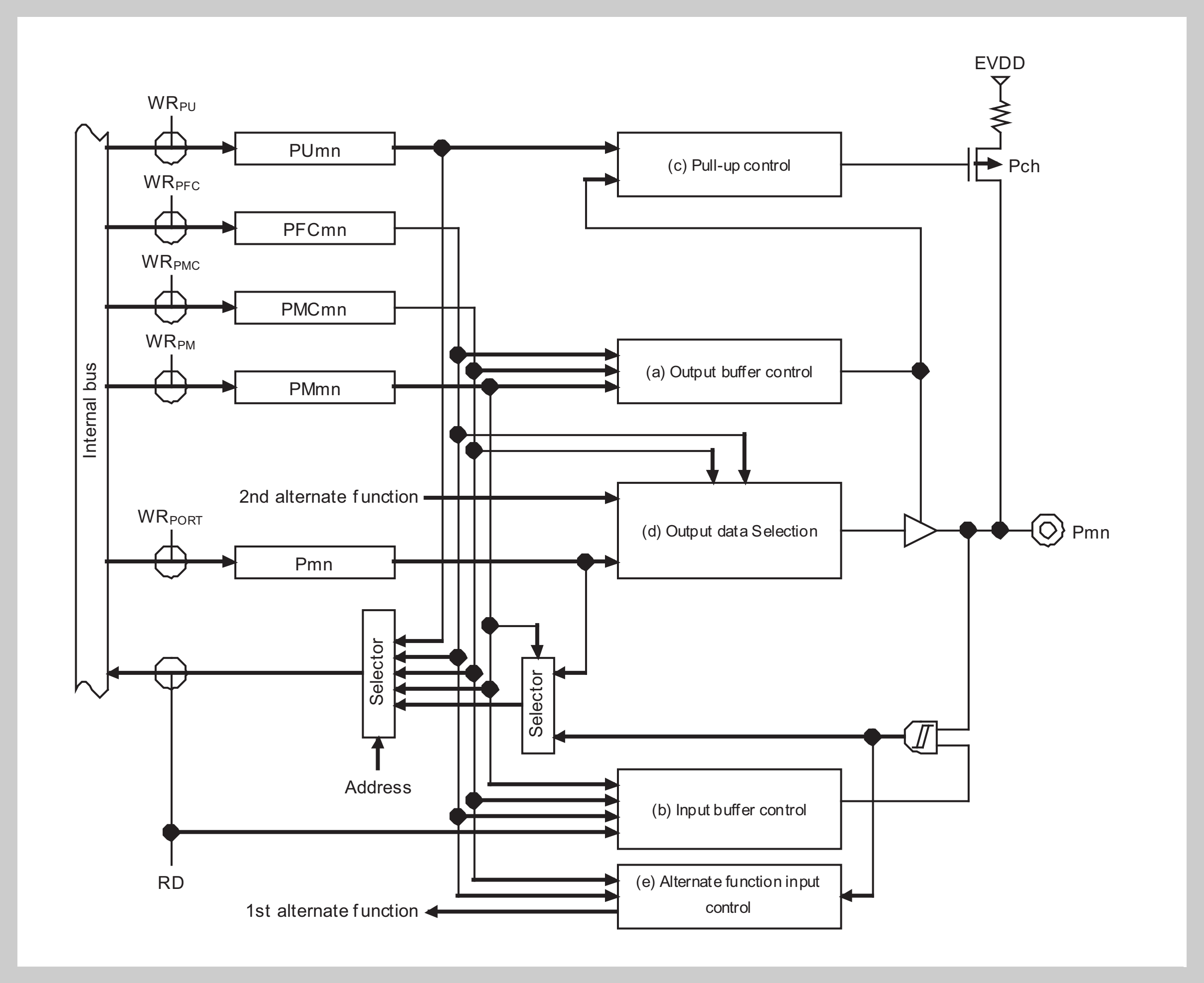


Figure 2-22 Port type E10-U block diagram

2.4.14 Port type E10-UI

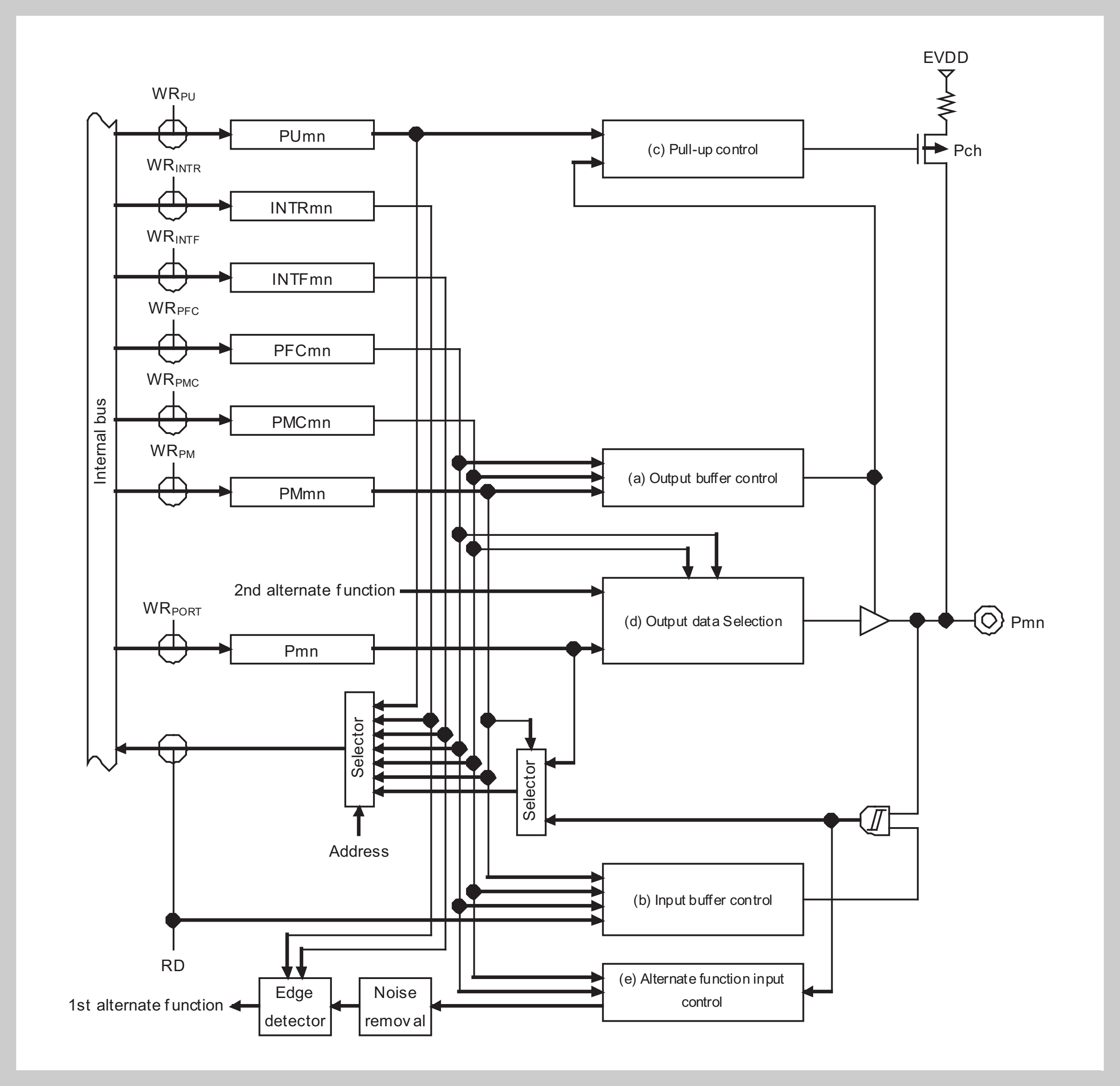


Figure 2-23 Port type E10-UI block diagram

2.4.15 Port type E11-U

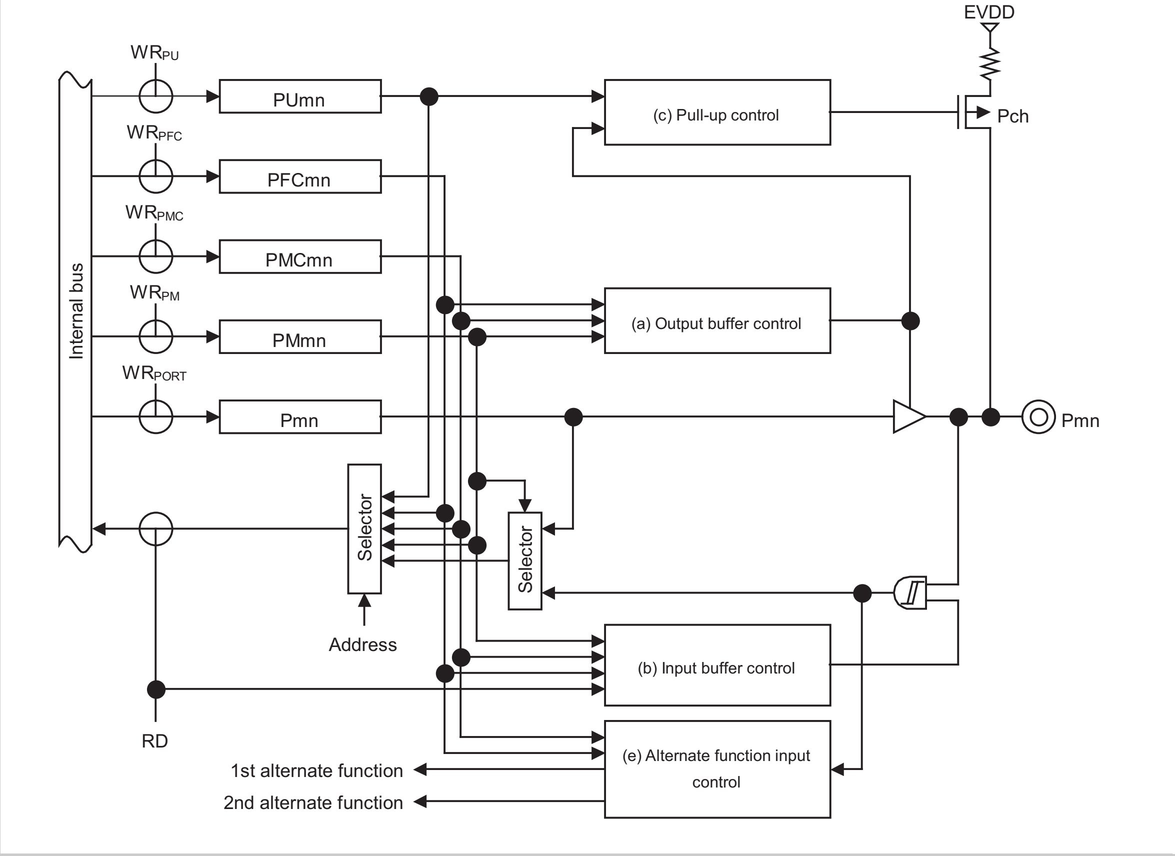


Figure 2-24 Port type E11-U block diagram

2.4.16 Port type E11-UI

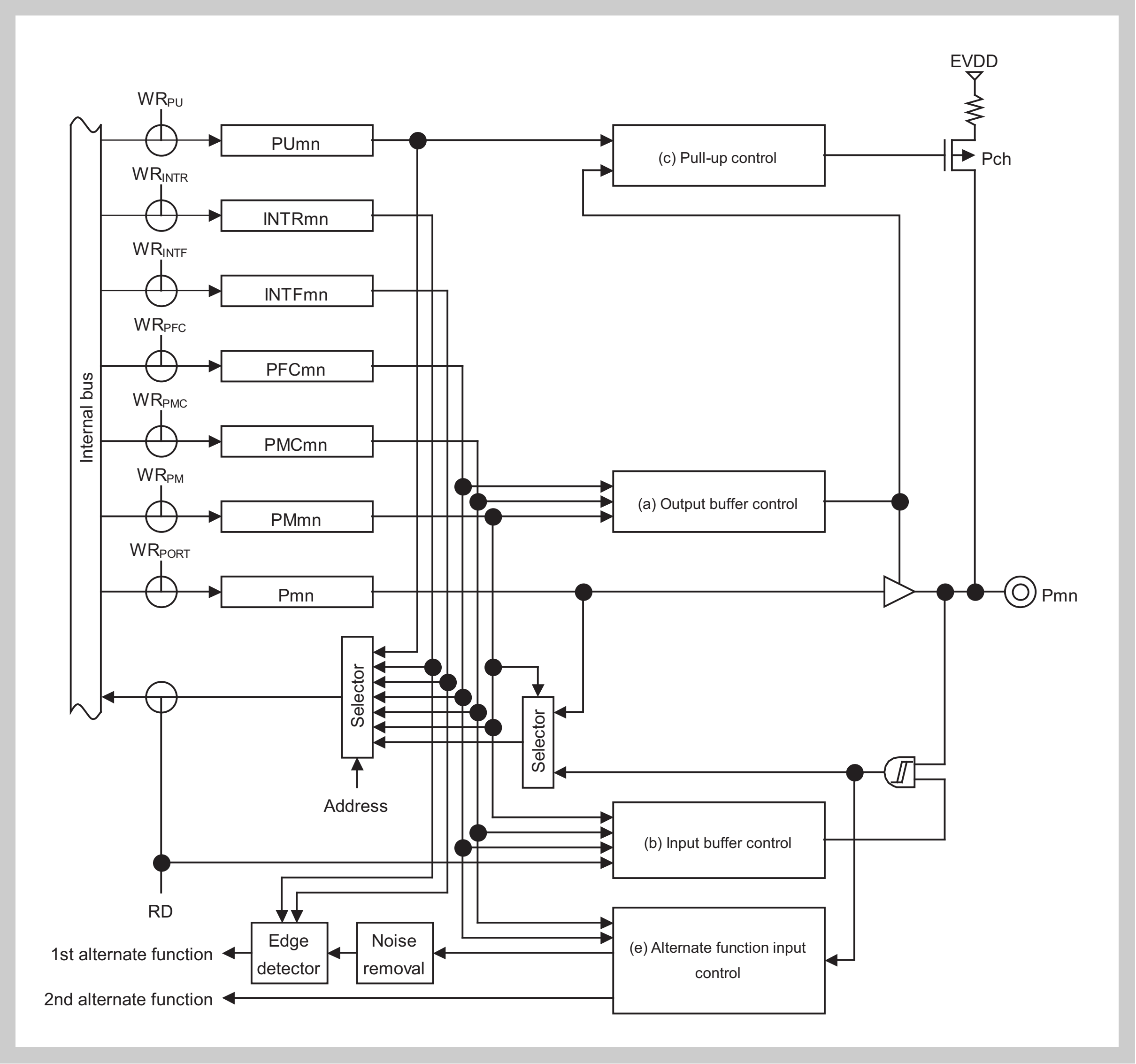


Figure 2-25 Port type E11-UI block diagram

2.4.17 Port type E21-U

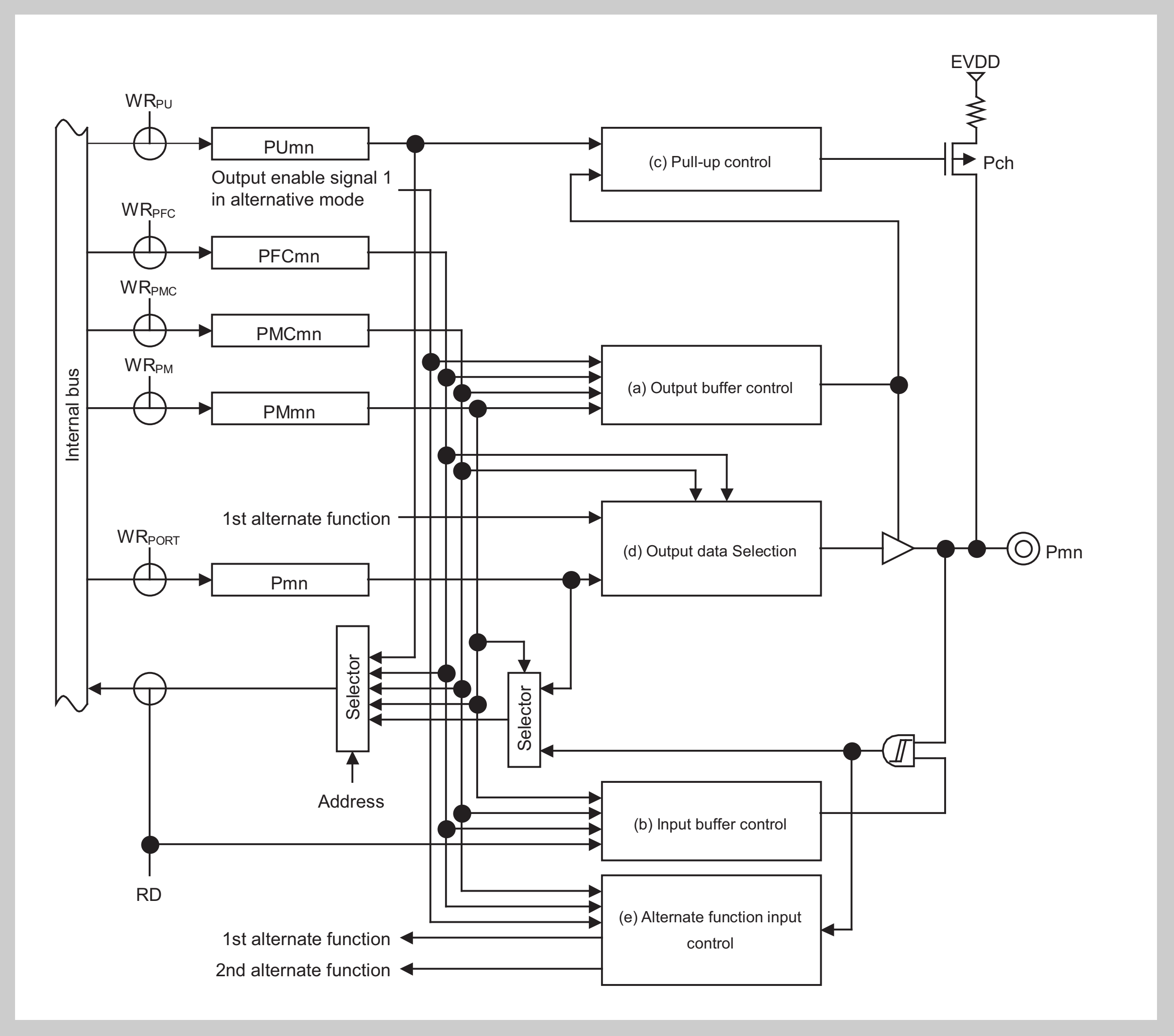


Figure 2-26 Port type E21-U block diagram

2.4.18 Port type Ex0-U

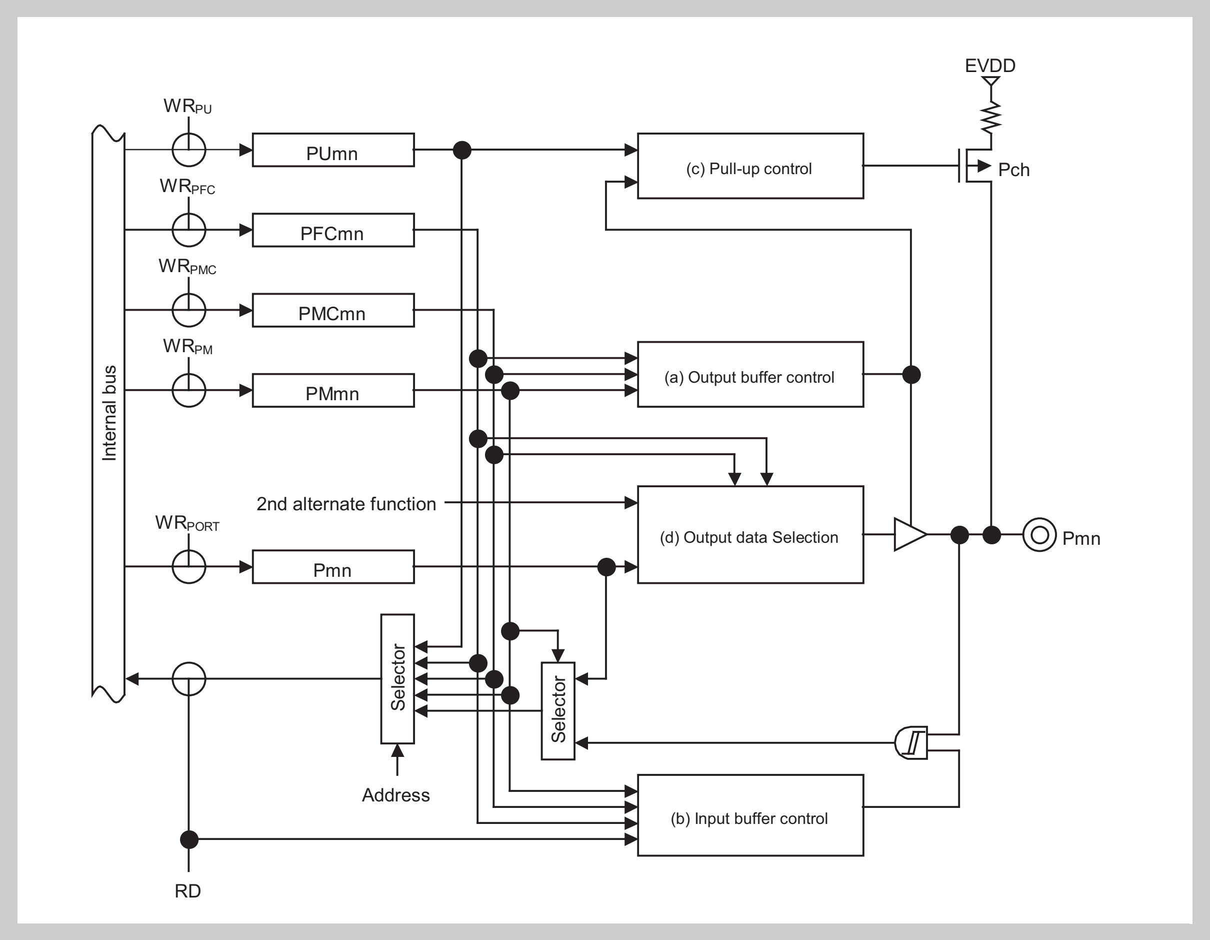


Figure 2-27 Port type Ex0-U block diagram

2.4.19 Port type Ex1-U

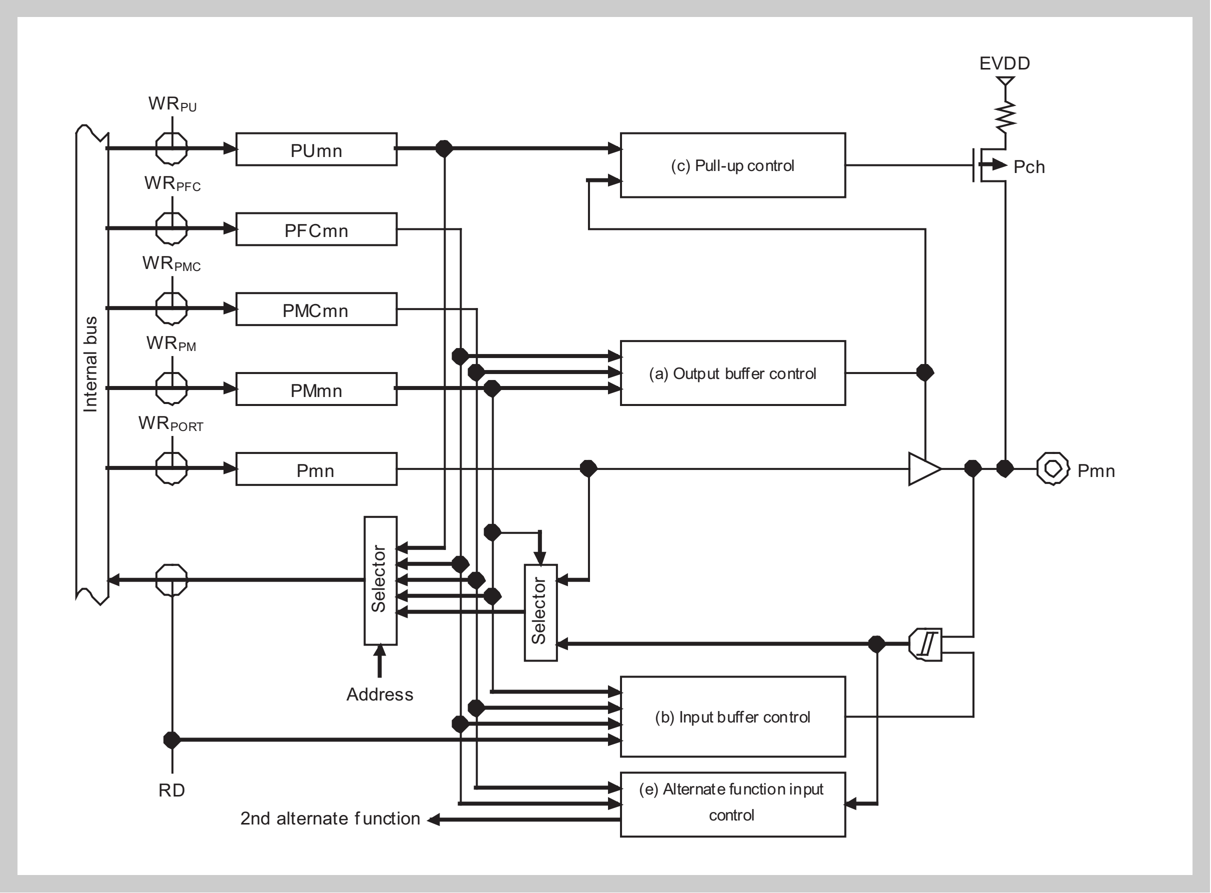


Figure 2-28 Port type Ex1-U block diagram

2.4.20 Port type Ex1-UI

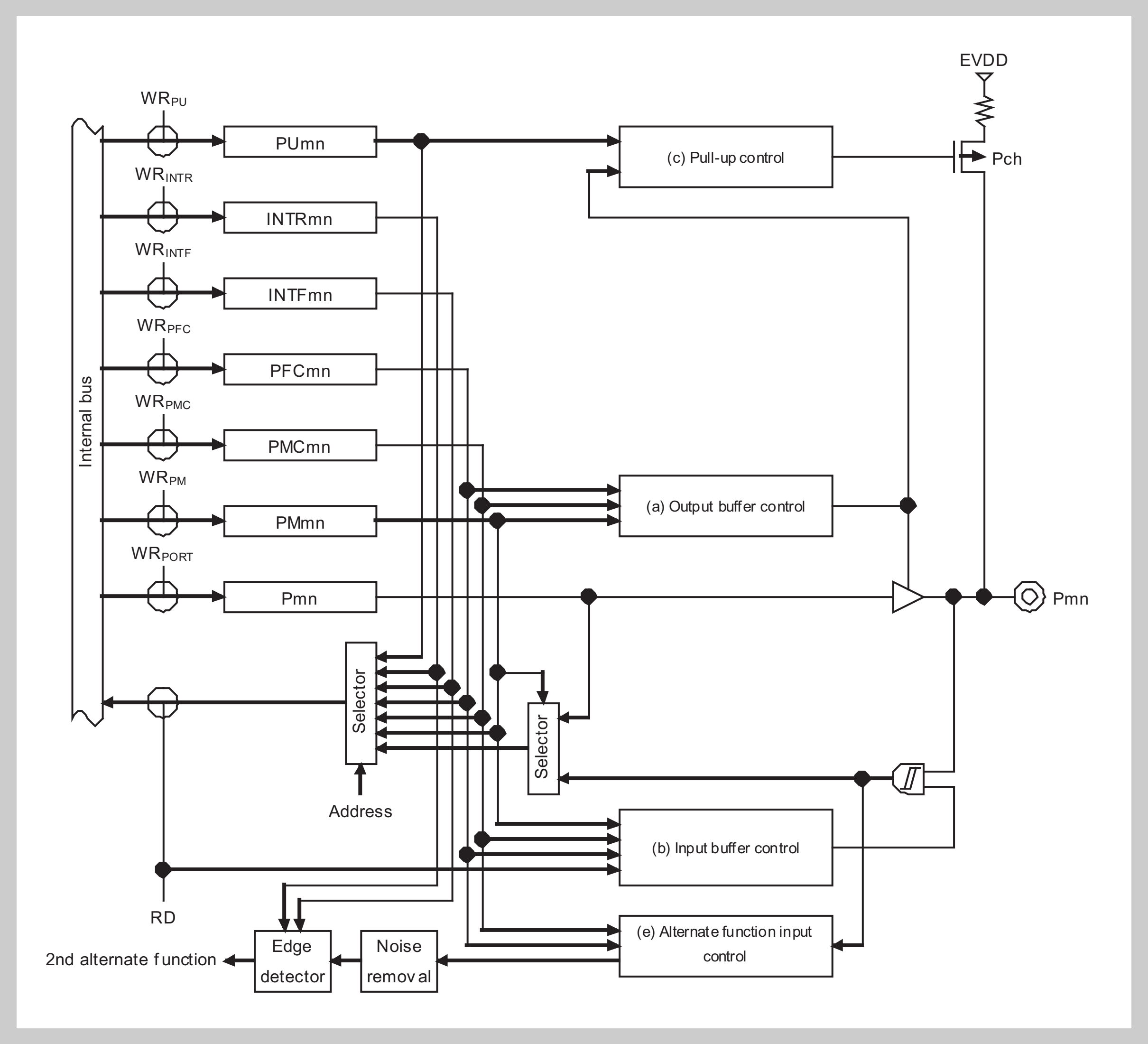


Figure 2-29 Port type Ex1-UI block diagram

2.4.21 Port type Ex2-U

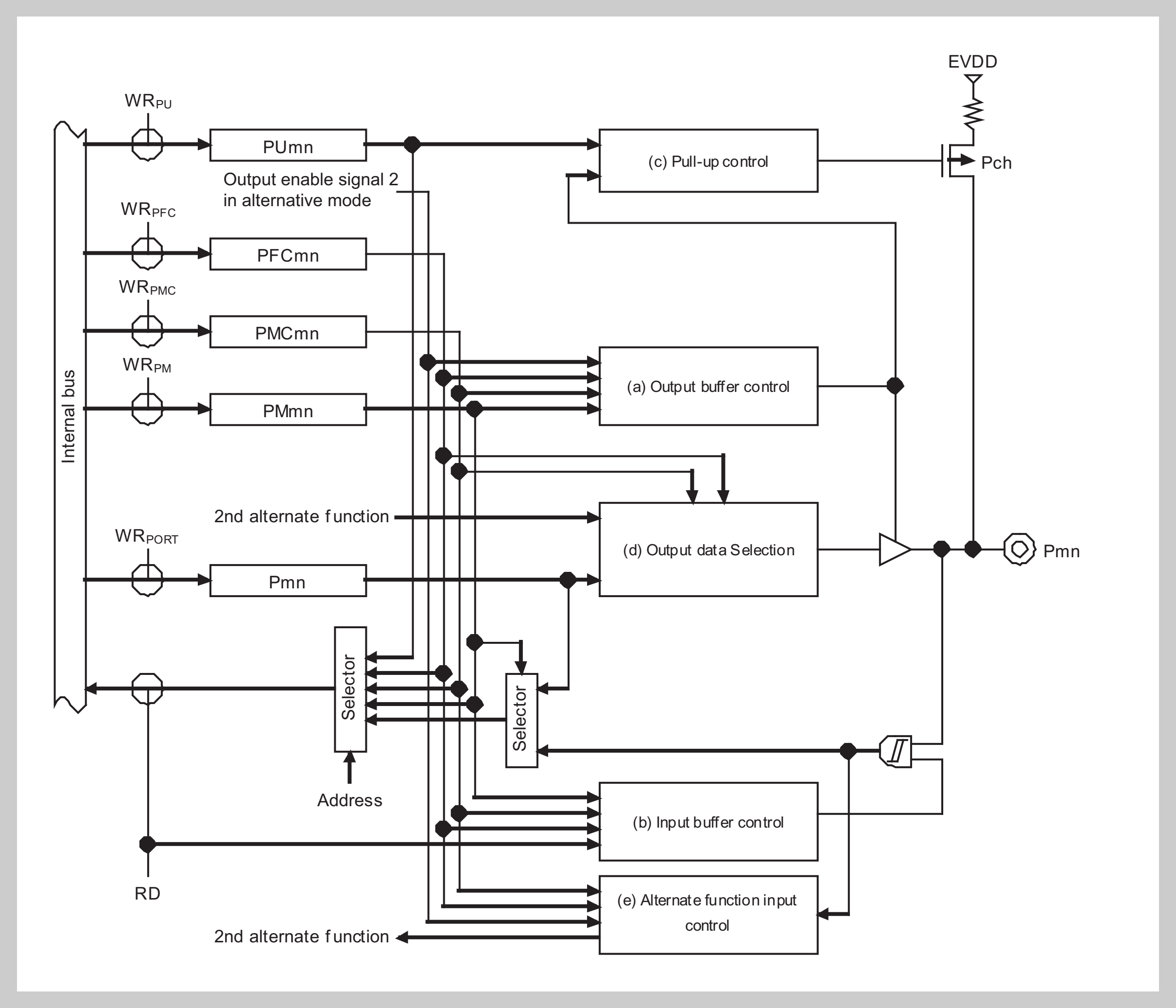


Figure 2-30 Port type Ex2-U block diagram

2.4.22 Port type F010x-U

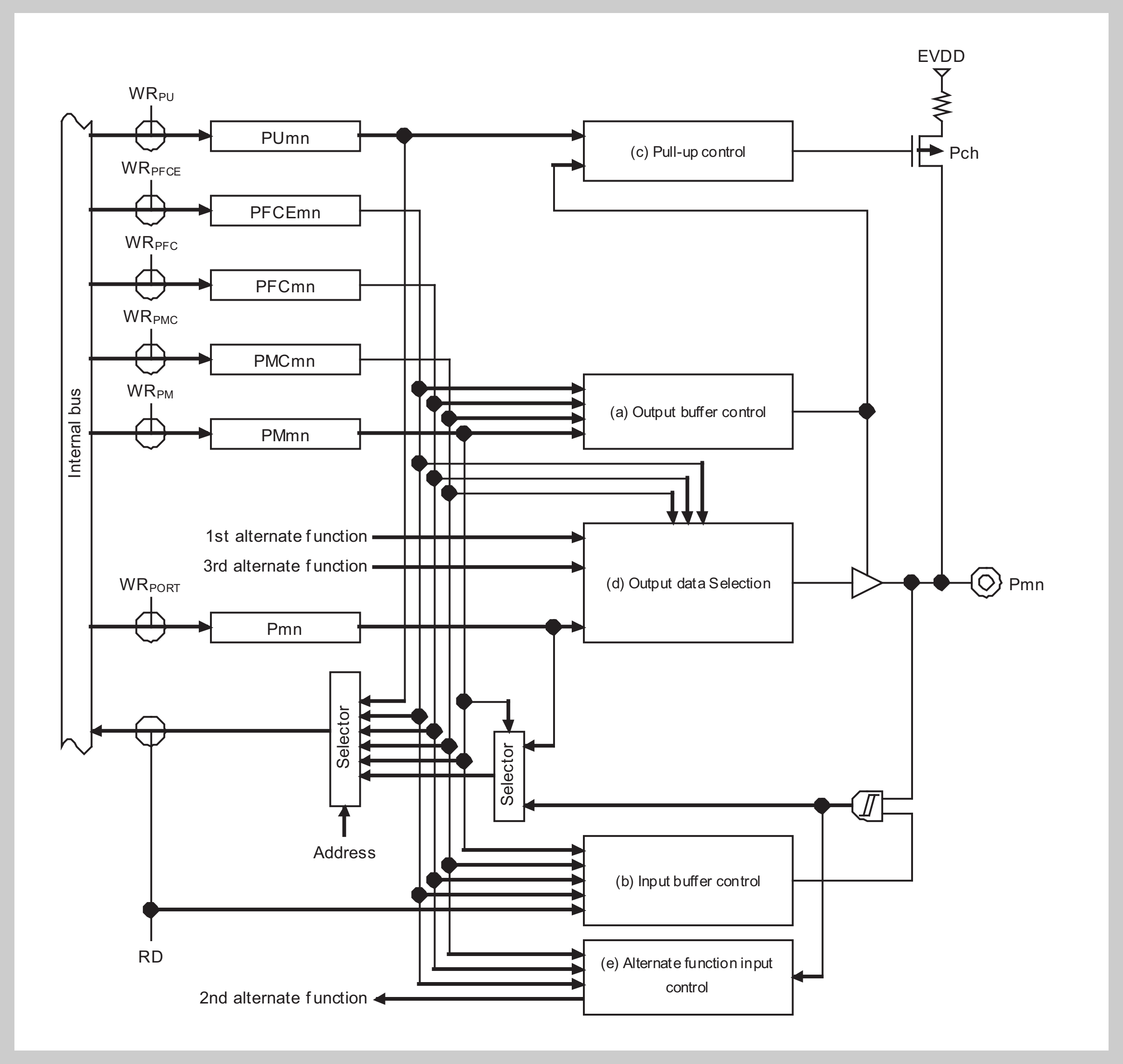


Figure 2-31 Port type F010x-U block diagram

2.4.23 Port type F010x-UI

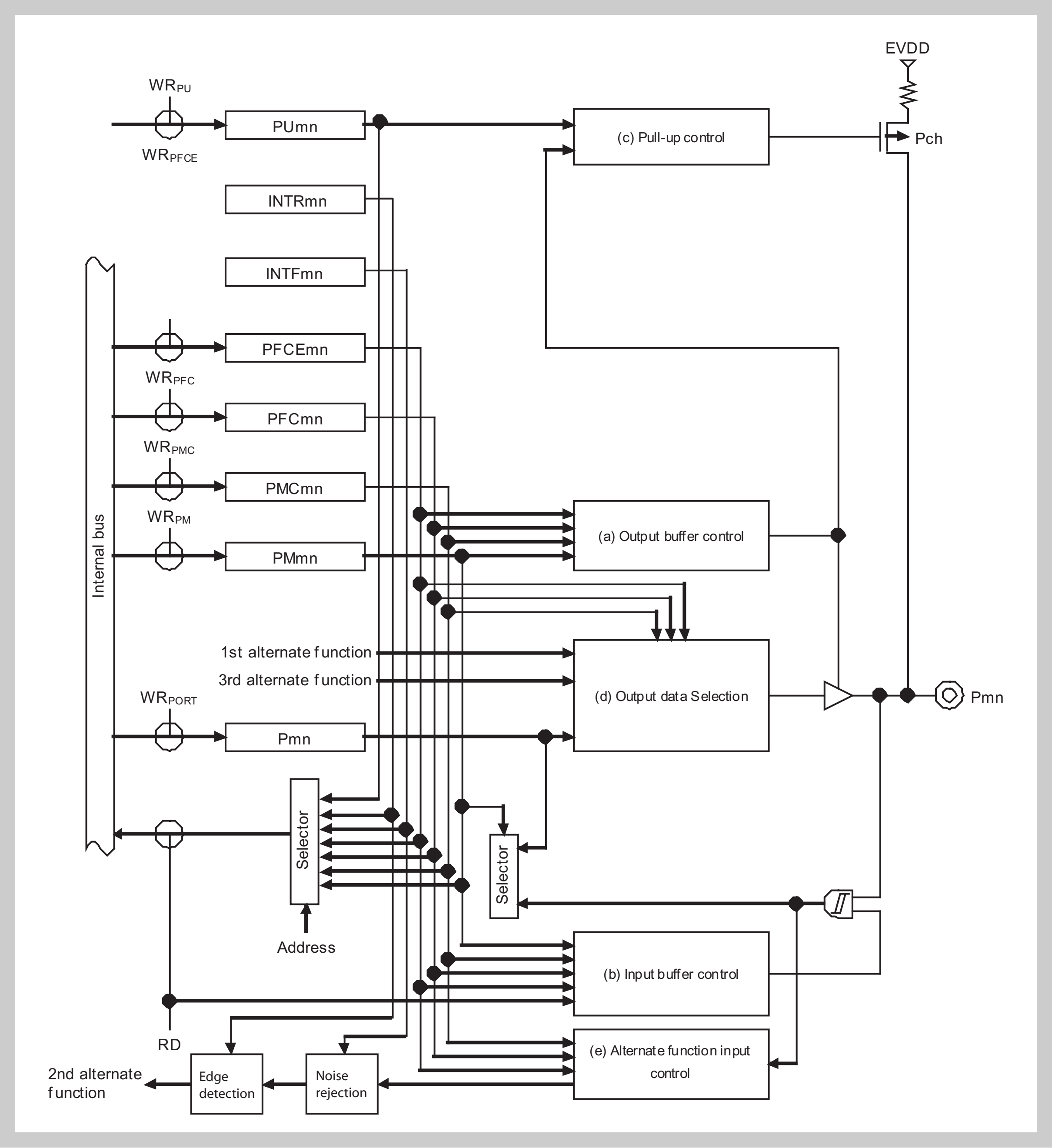


Figure 2-32 Port type F010x-UI block diagram

2.4.24 Port type F100x-U

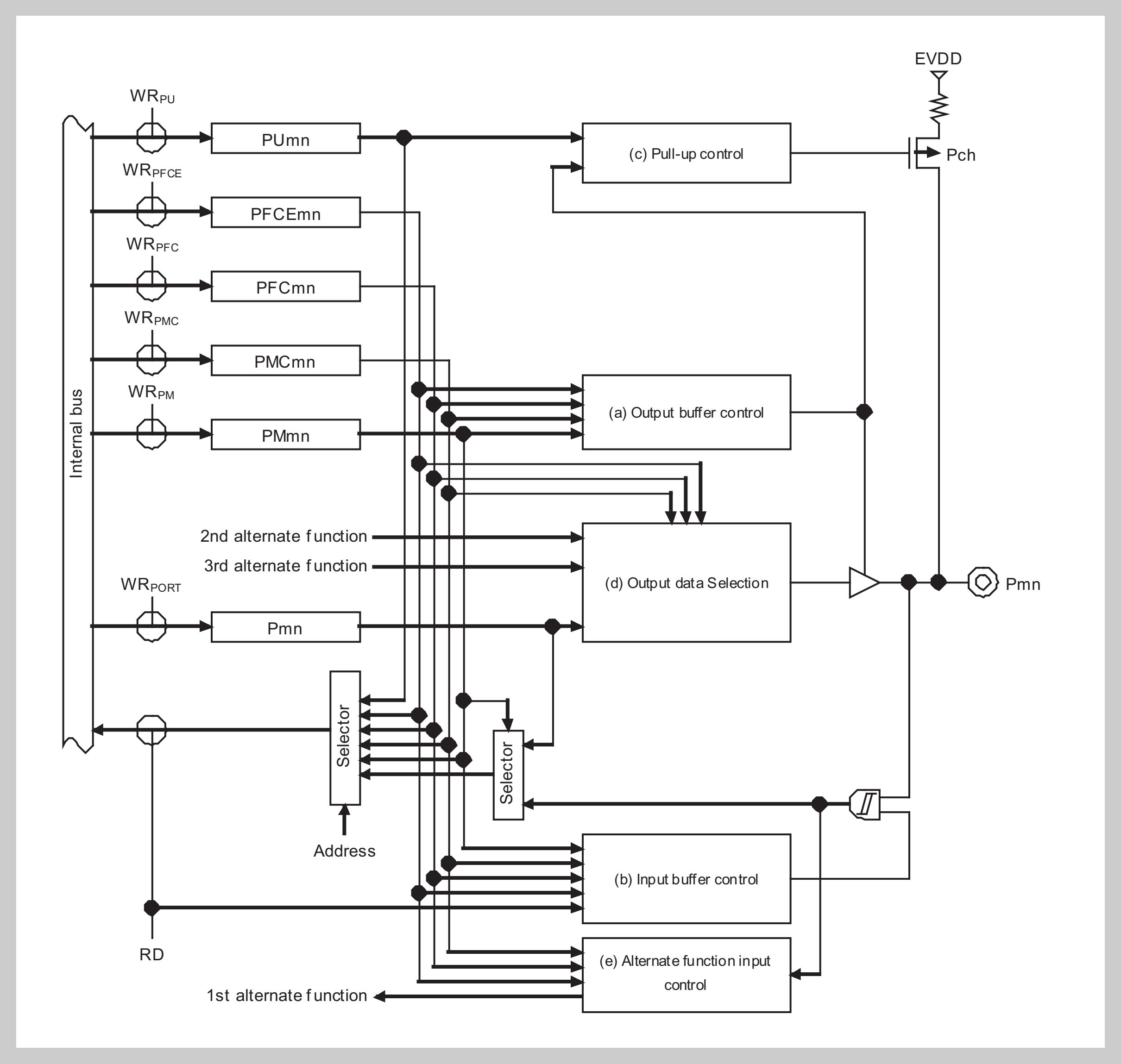


Figure 2-33 Port type F100x-U block diagram

2.4.25 Port type F1010-U

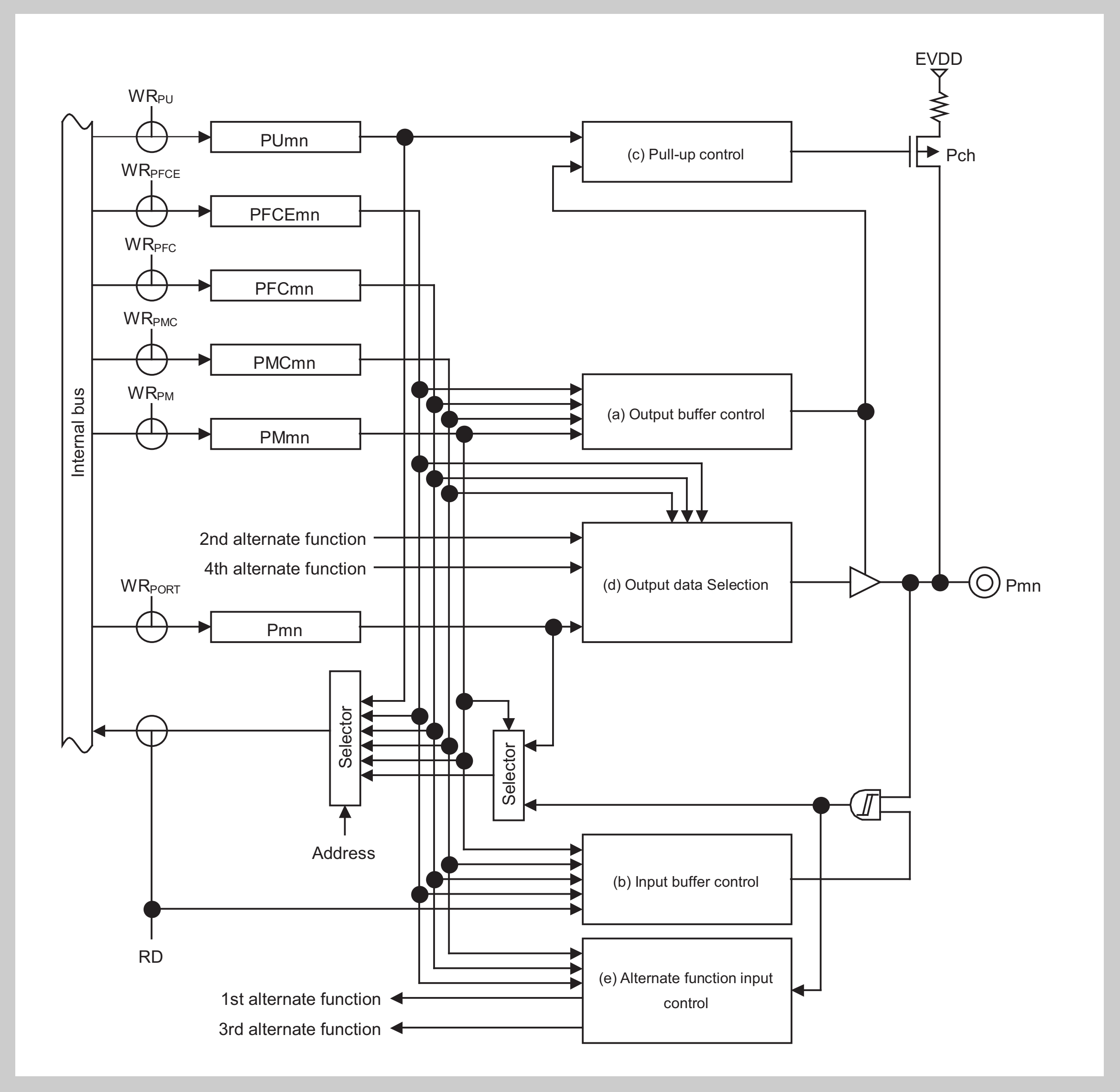


Figure 2-34 Port type F1010-U block diagram

2.4.26 Port type F101x-U

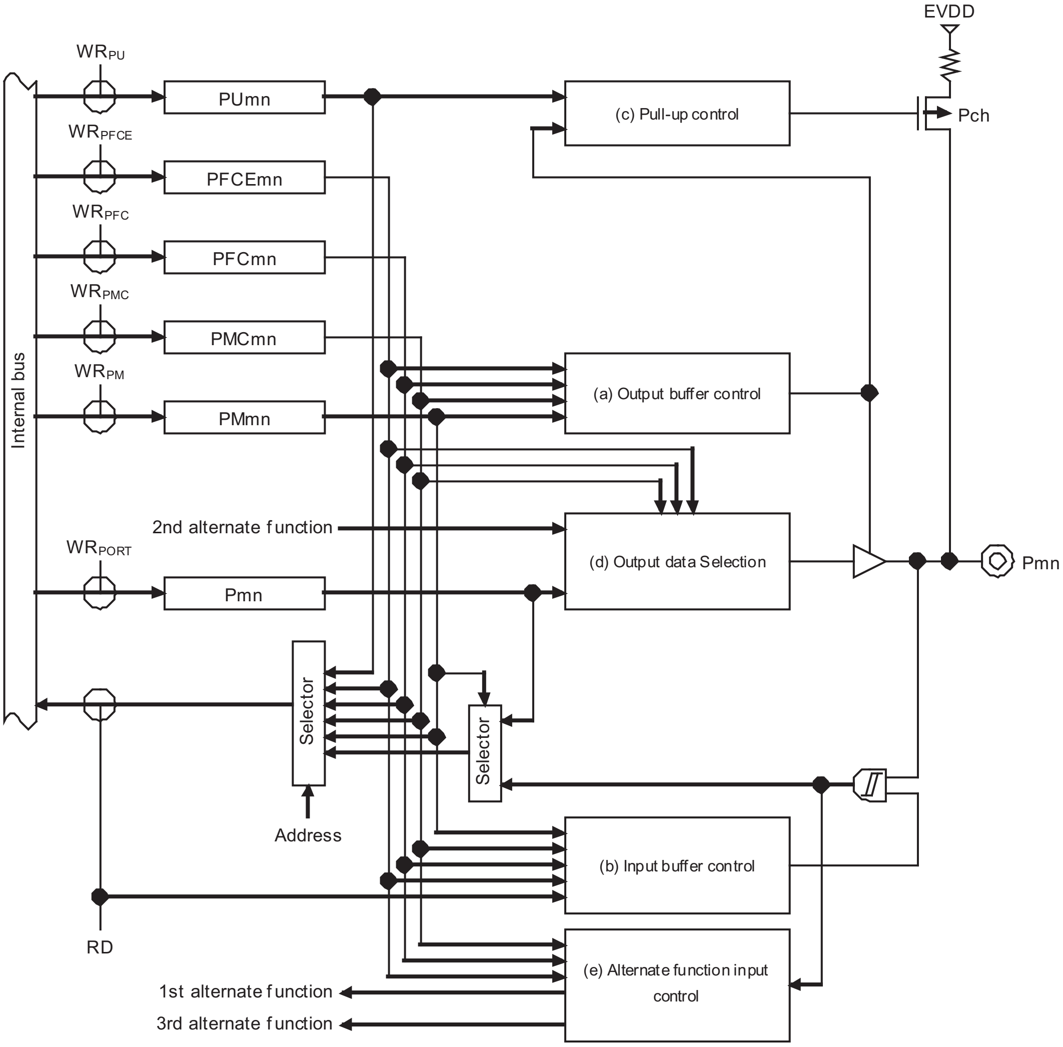


Figure 2-35 Port type F101x-U block diagram

2.4.27 Port type F1100O0-U

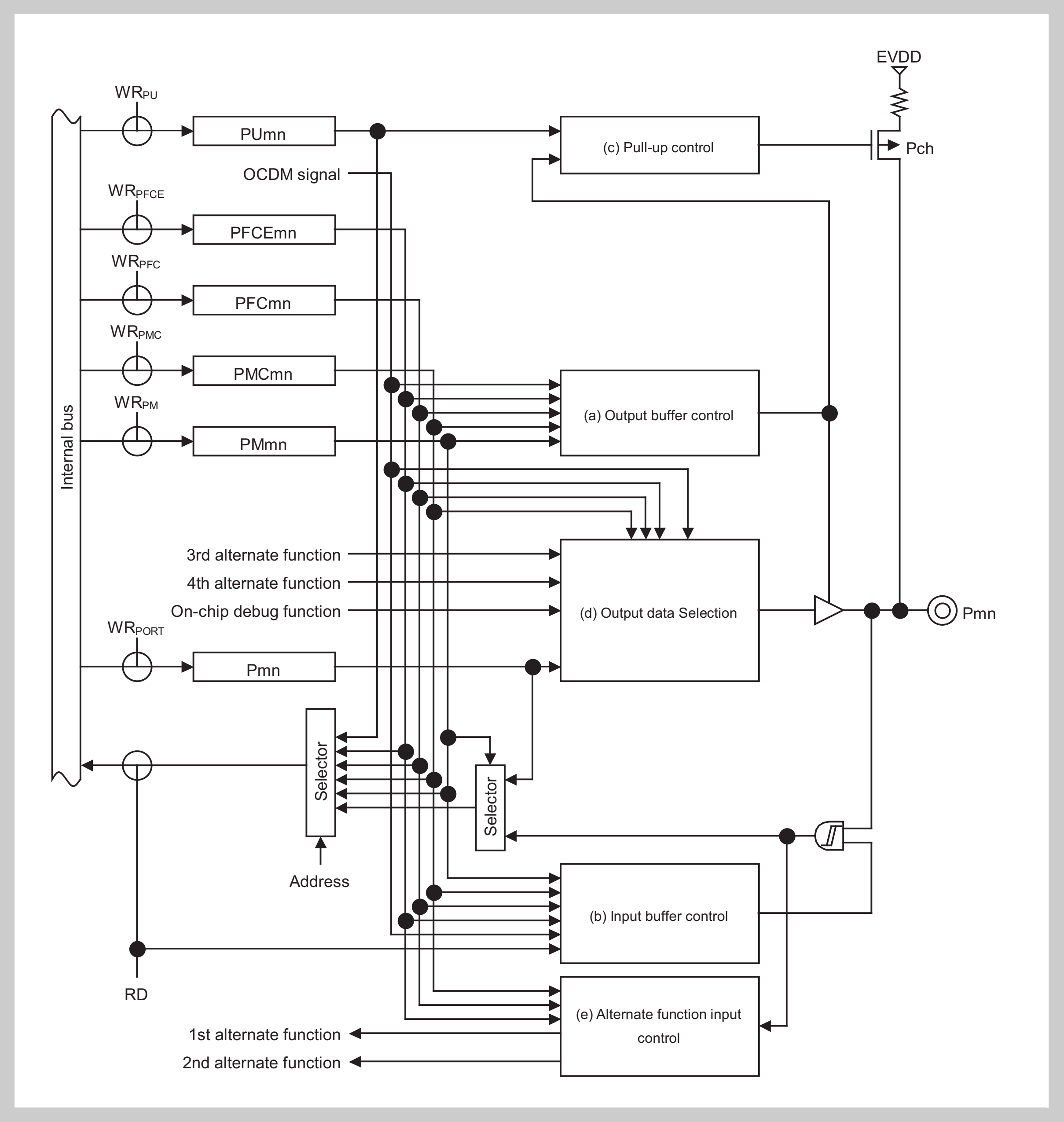


Figure 2-36 Port type F1100O0-U block diagram

2.4.28 Port type F1100O1-U

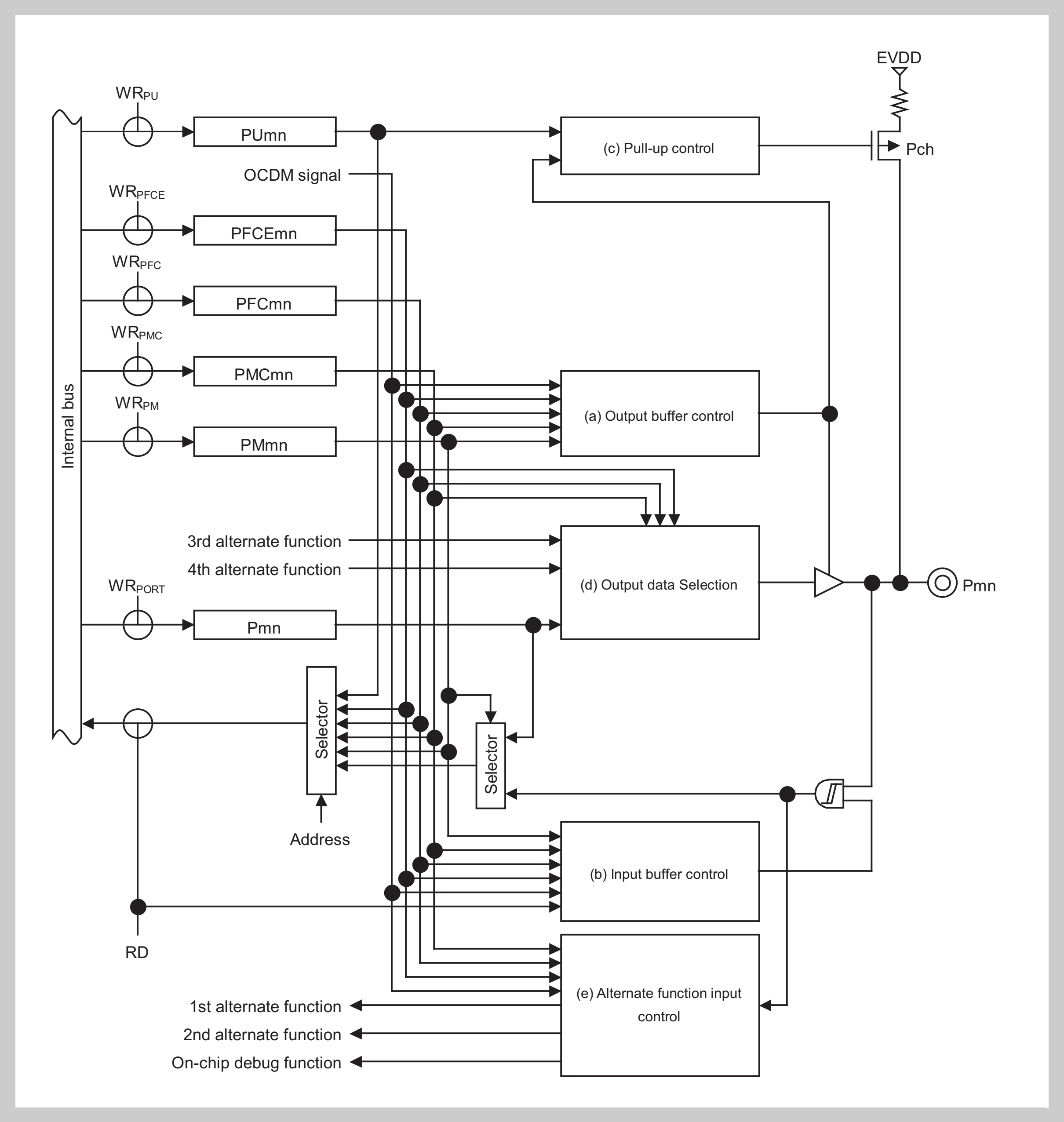


Figure 2-37 Port type F1100O1-U block diagram

2.4.29 Port type F1100-U

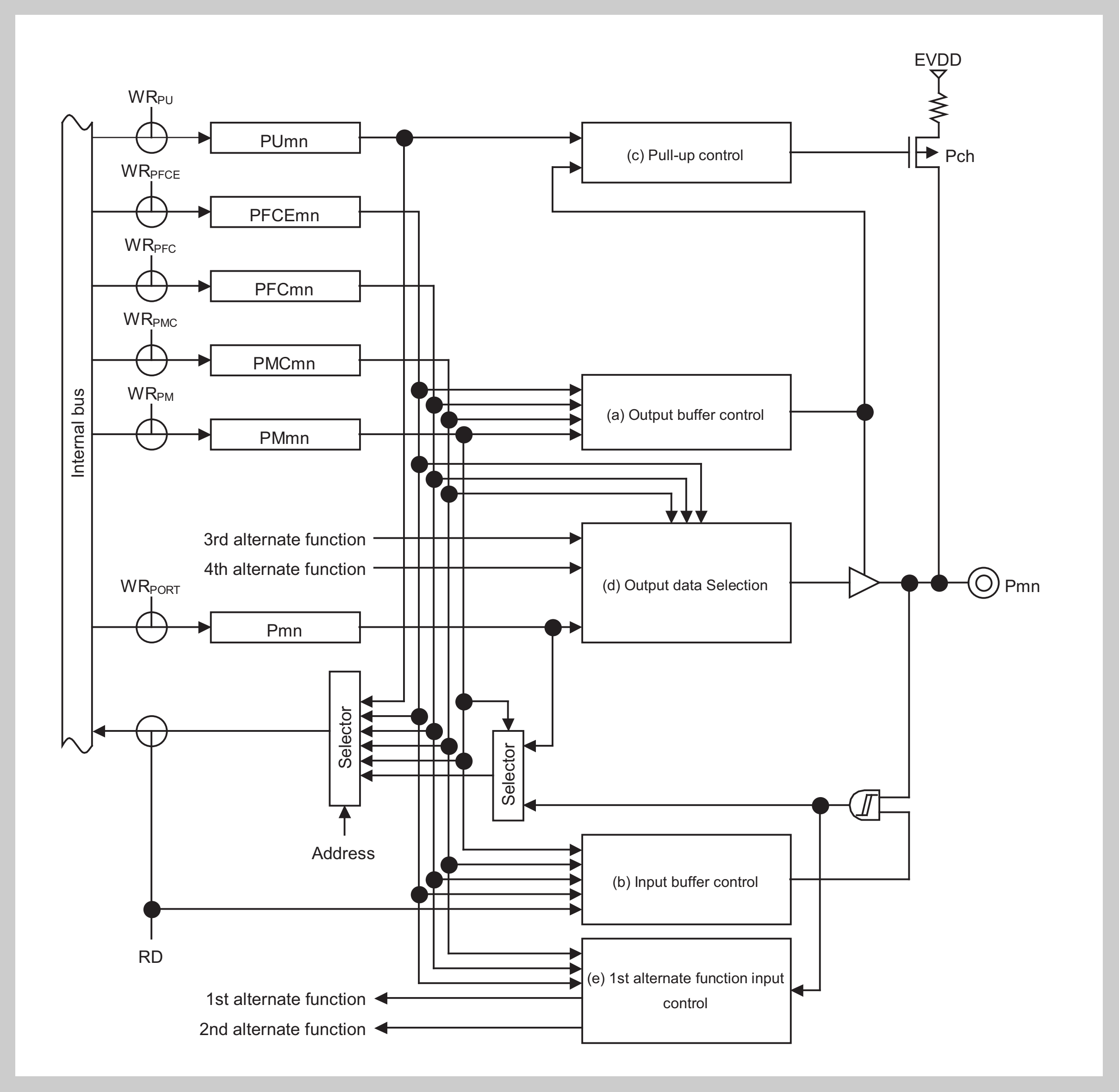


Figure 2-38 Port type F1100-U block diagram

2.4.30 Port type F1110-UI

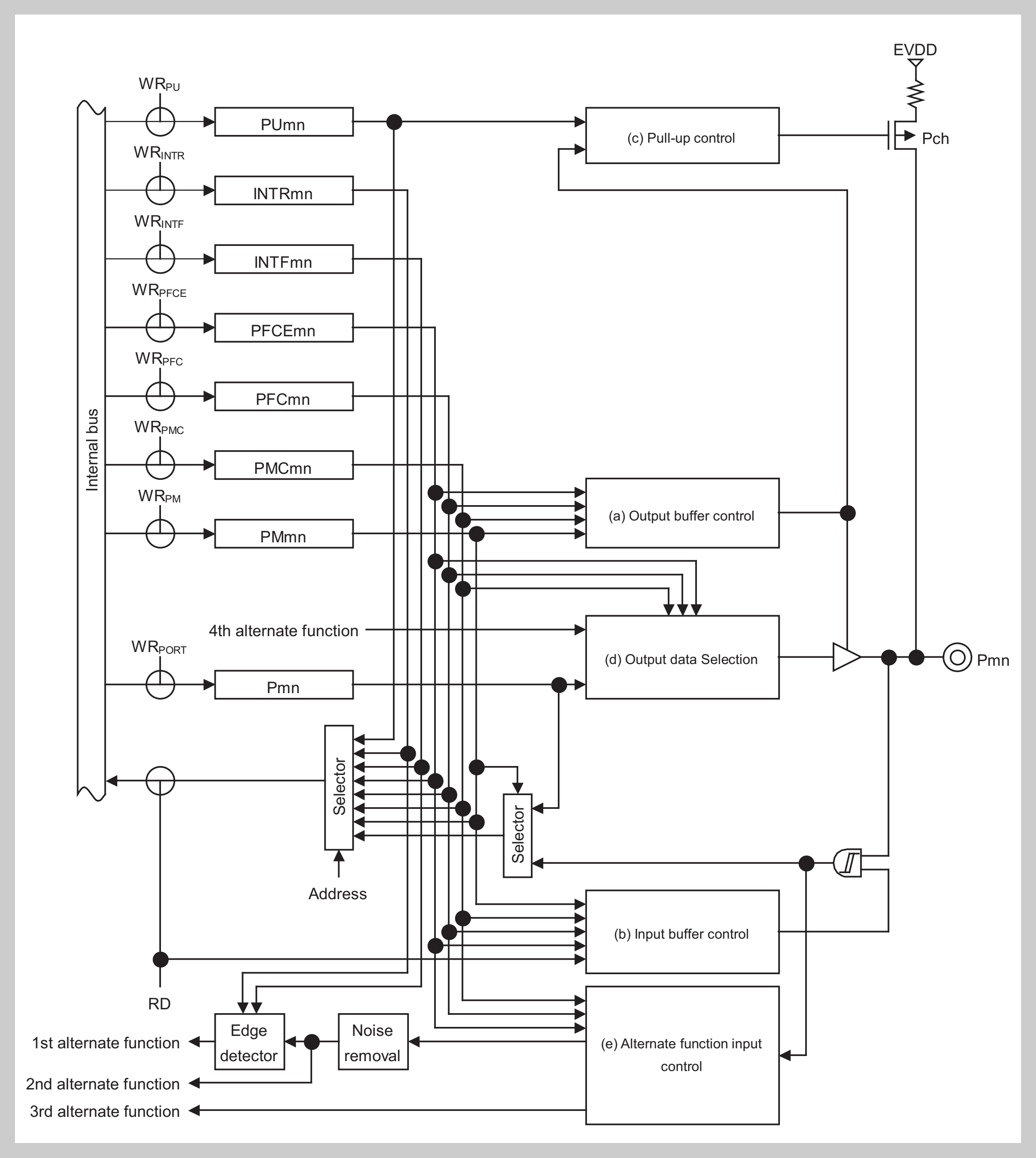


Figure 2-39 Port type F1110-UI block diagram

2.4.31 Port type F113x-UI

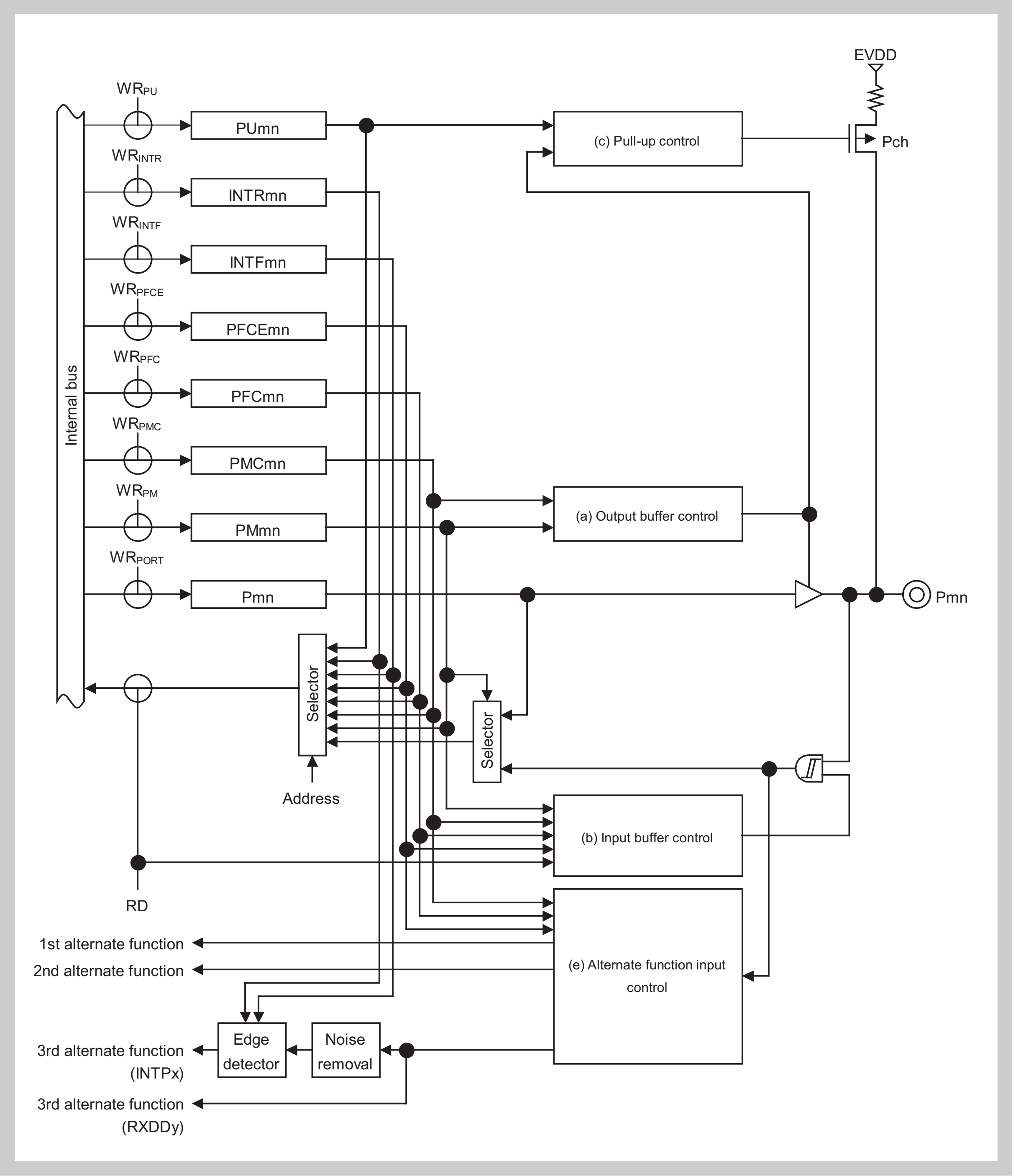


Figure 2-40 Port type F113x-UI block diagram

2.4.32 Port type F1x10-UI

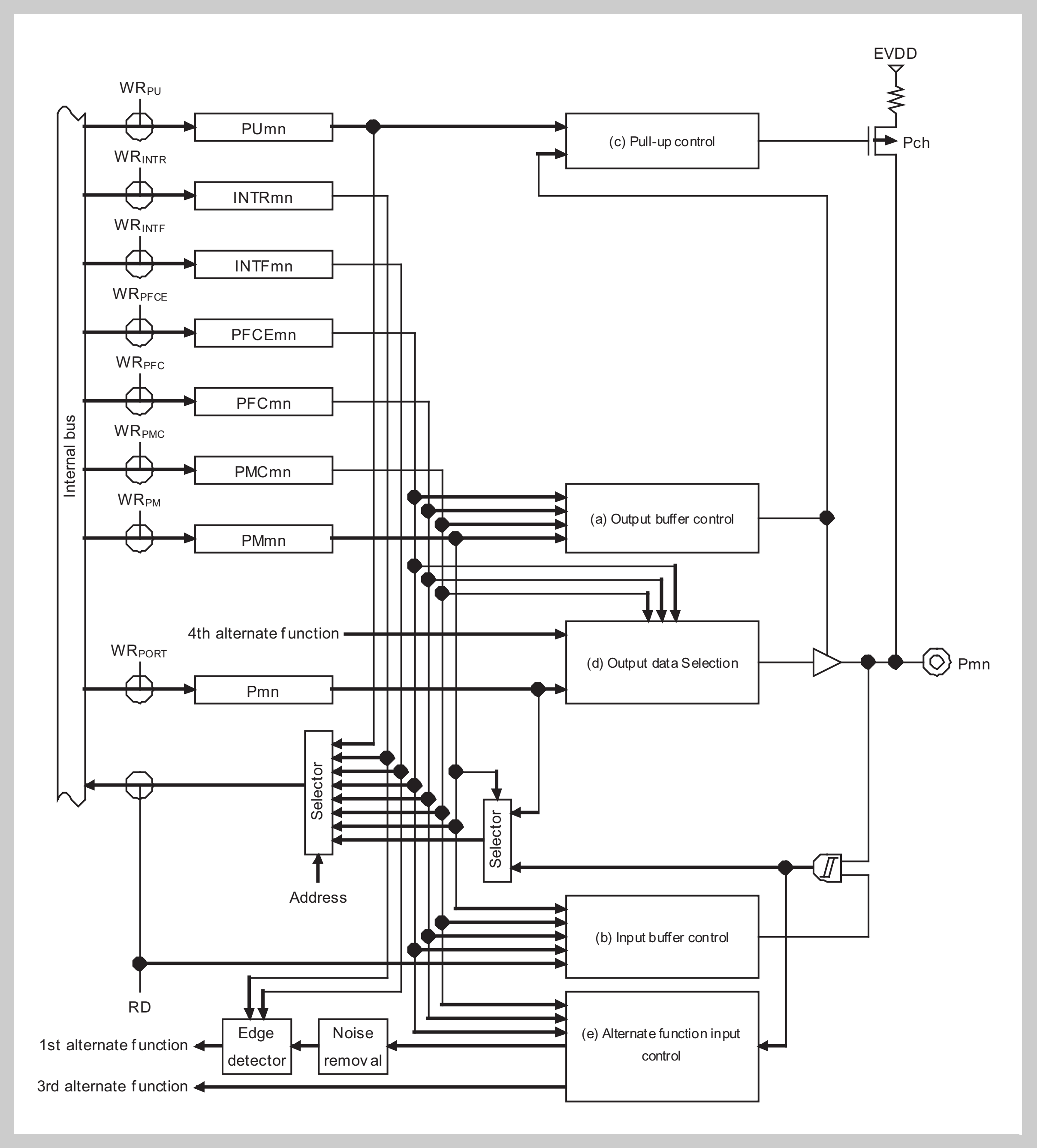


Figure 2-41 Port type F1x10-UI block diagram

2.4.33 Port type F3x1x-UI

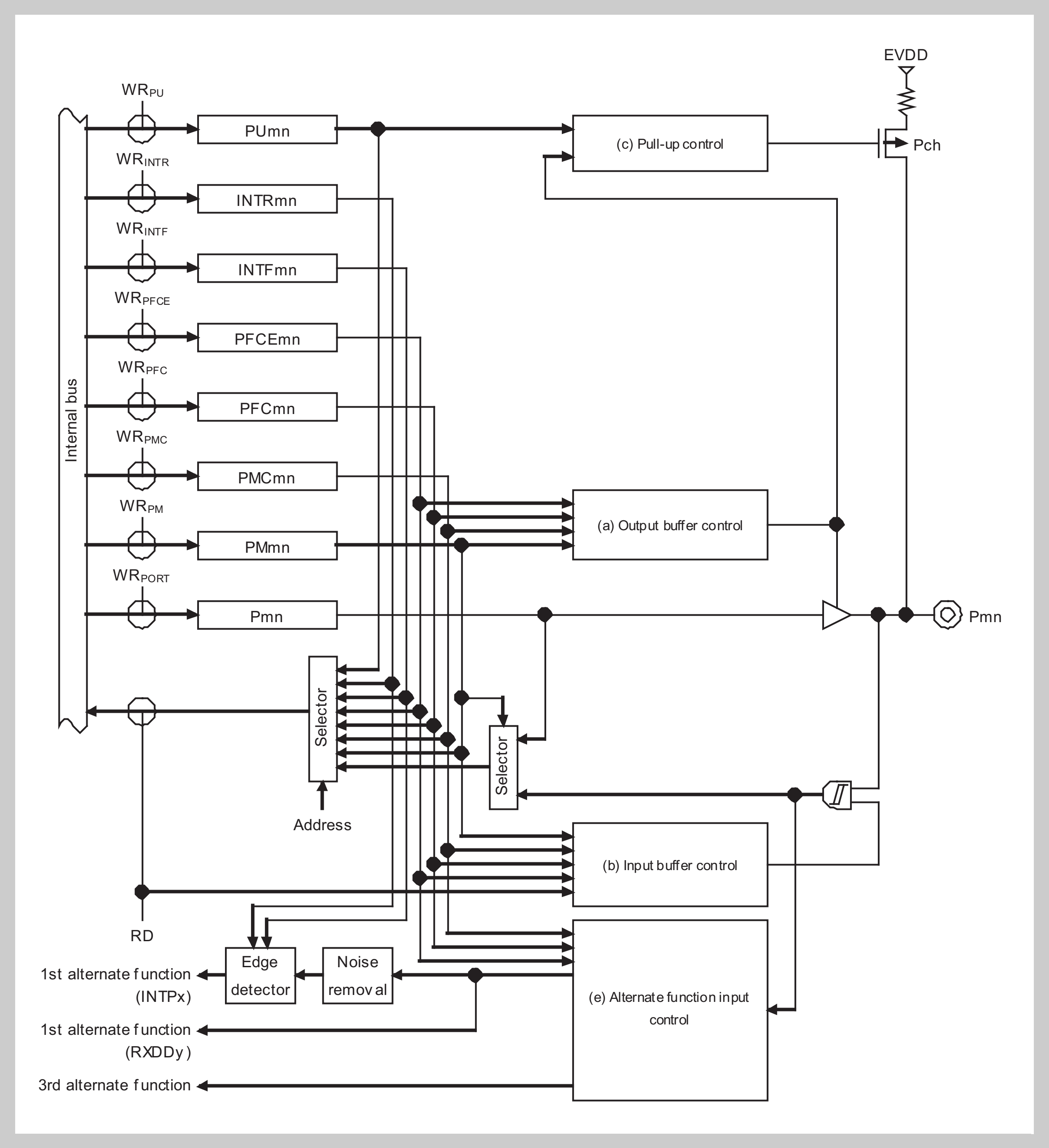


Figure 2-42 Port type F1x1x-UI block diagram

2.4.34 Port type F1xx0O1-U

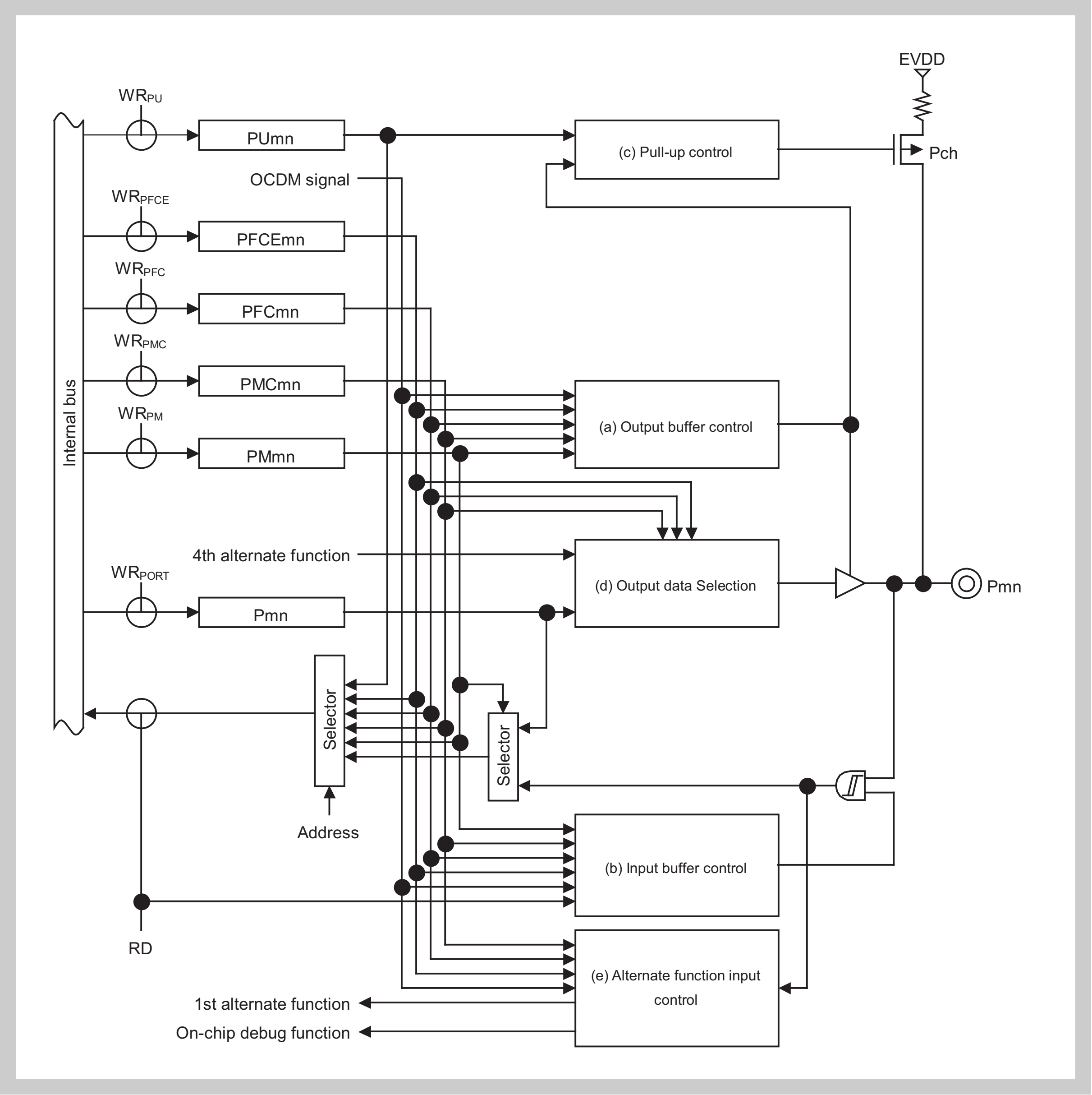


Figure 2-43 Port type F1xx0O1-U block diagram

2.4.35 Port type Fx010-U

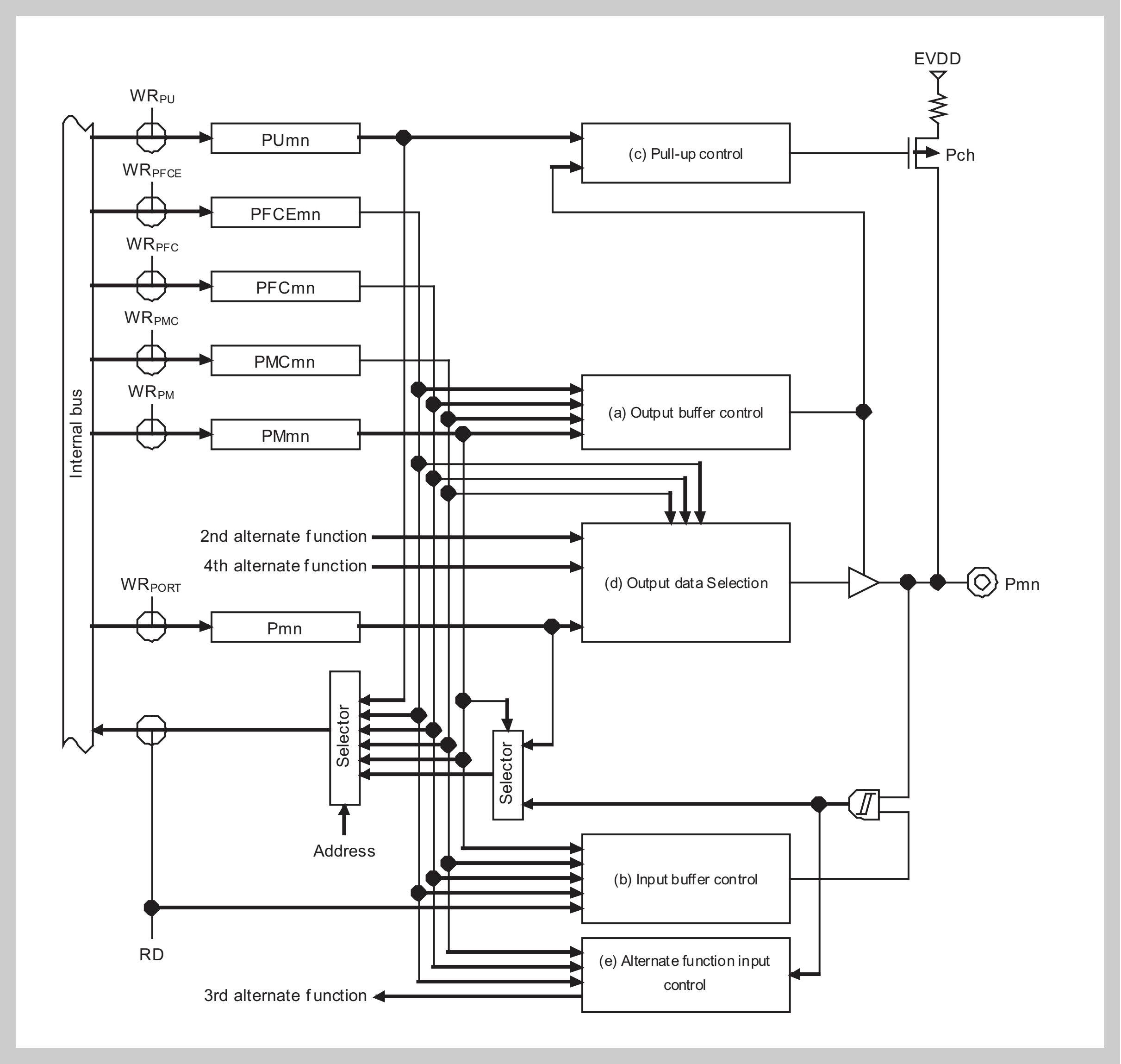


Figure 2-44 Port type Fx010-U block diagram

2.4.36 Port type Fx01x-U

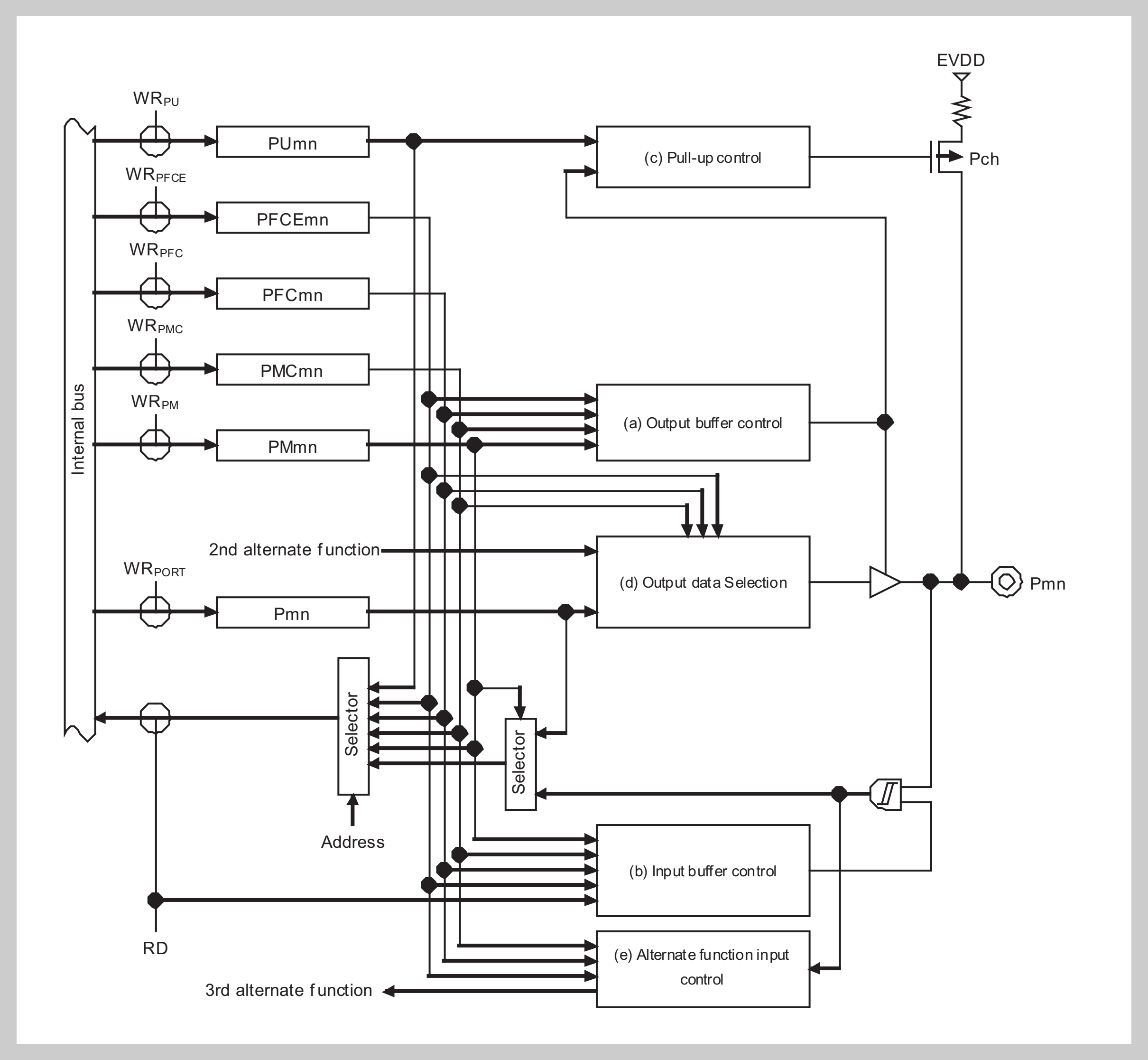


Figure 2-45 Port type Fx01x-U block diagram

2.4.37 Port type Fx103-UI

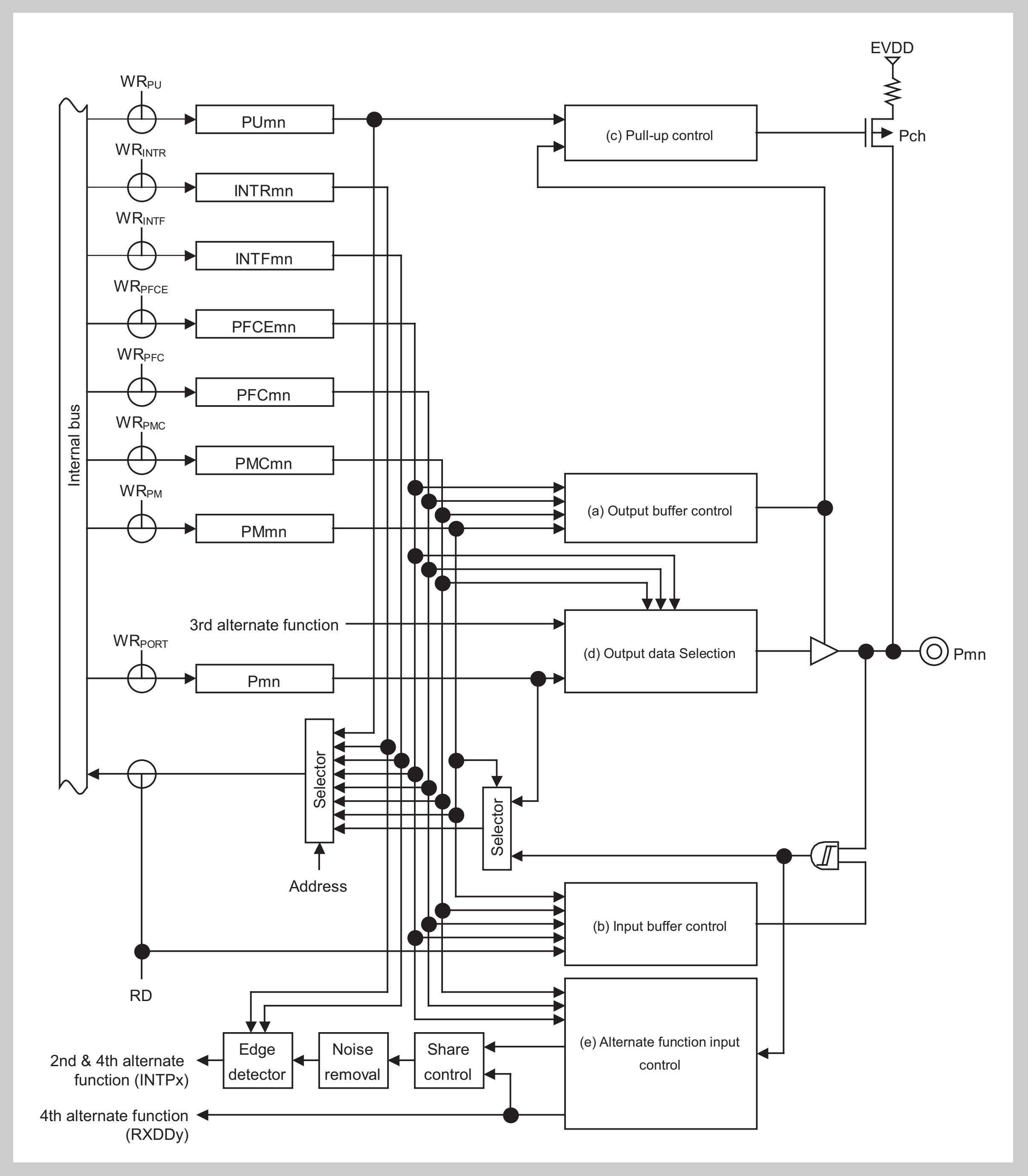


Figure 2-46 Port type Fx103-UI block diagram

2.4.38 Port type Fx10x-U

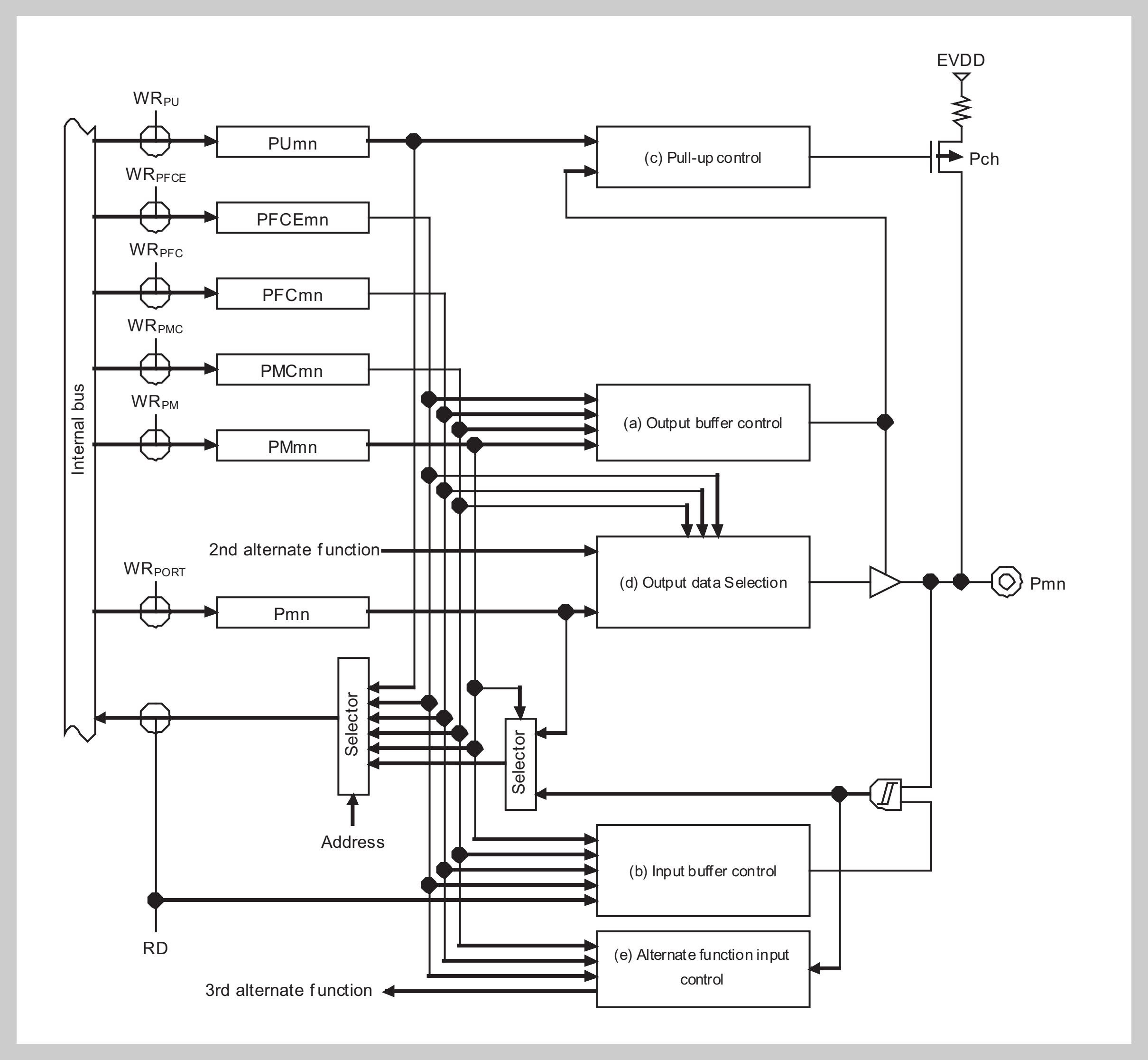


Figure 2-47 Port type Fx10x-U block diagram

2.4.39 Port type Fx10x-UI

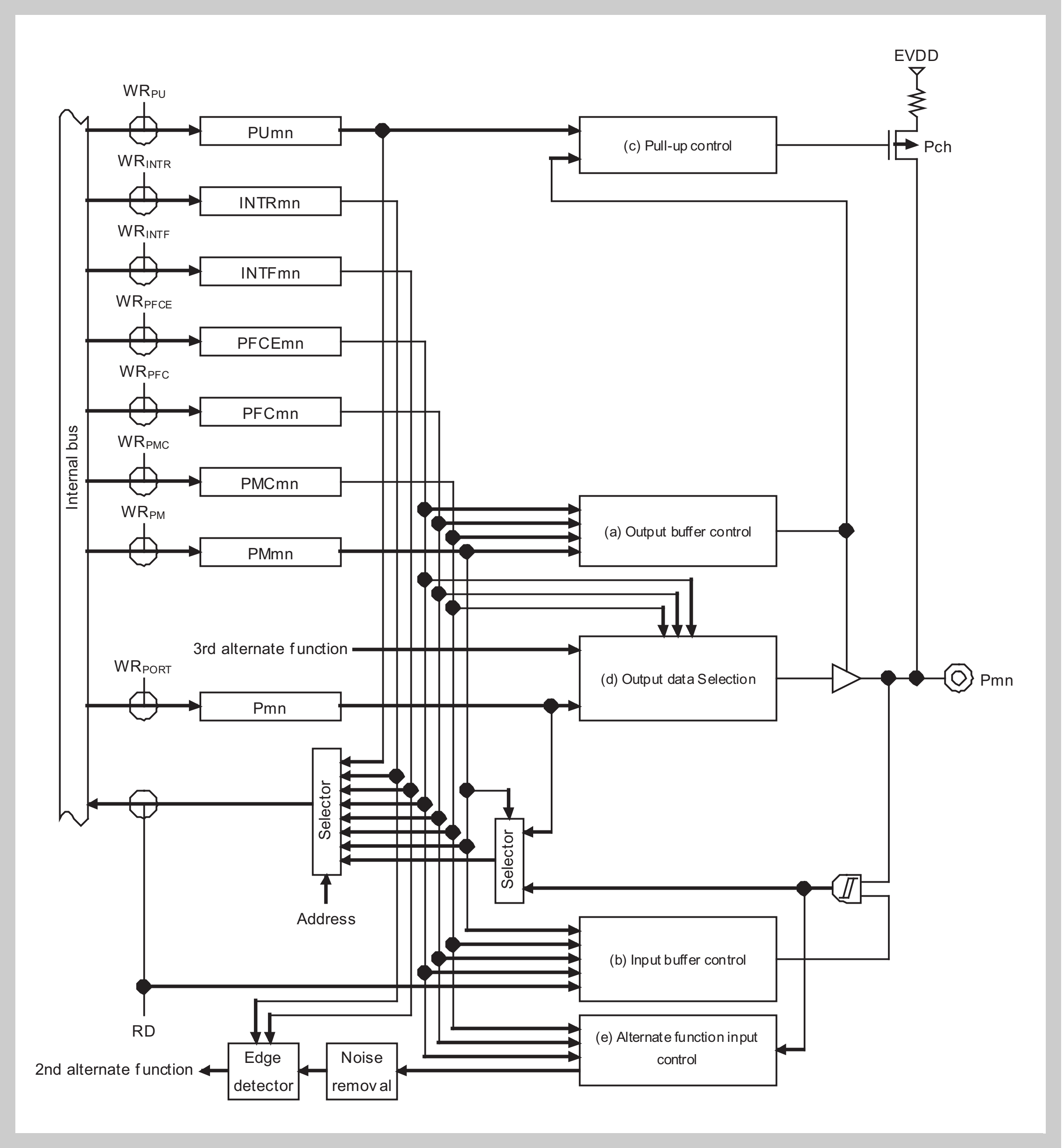


Figure 2-48 Port type Fx10x-UI block diagram

2.4.40 Port type Fx110-U

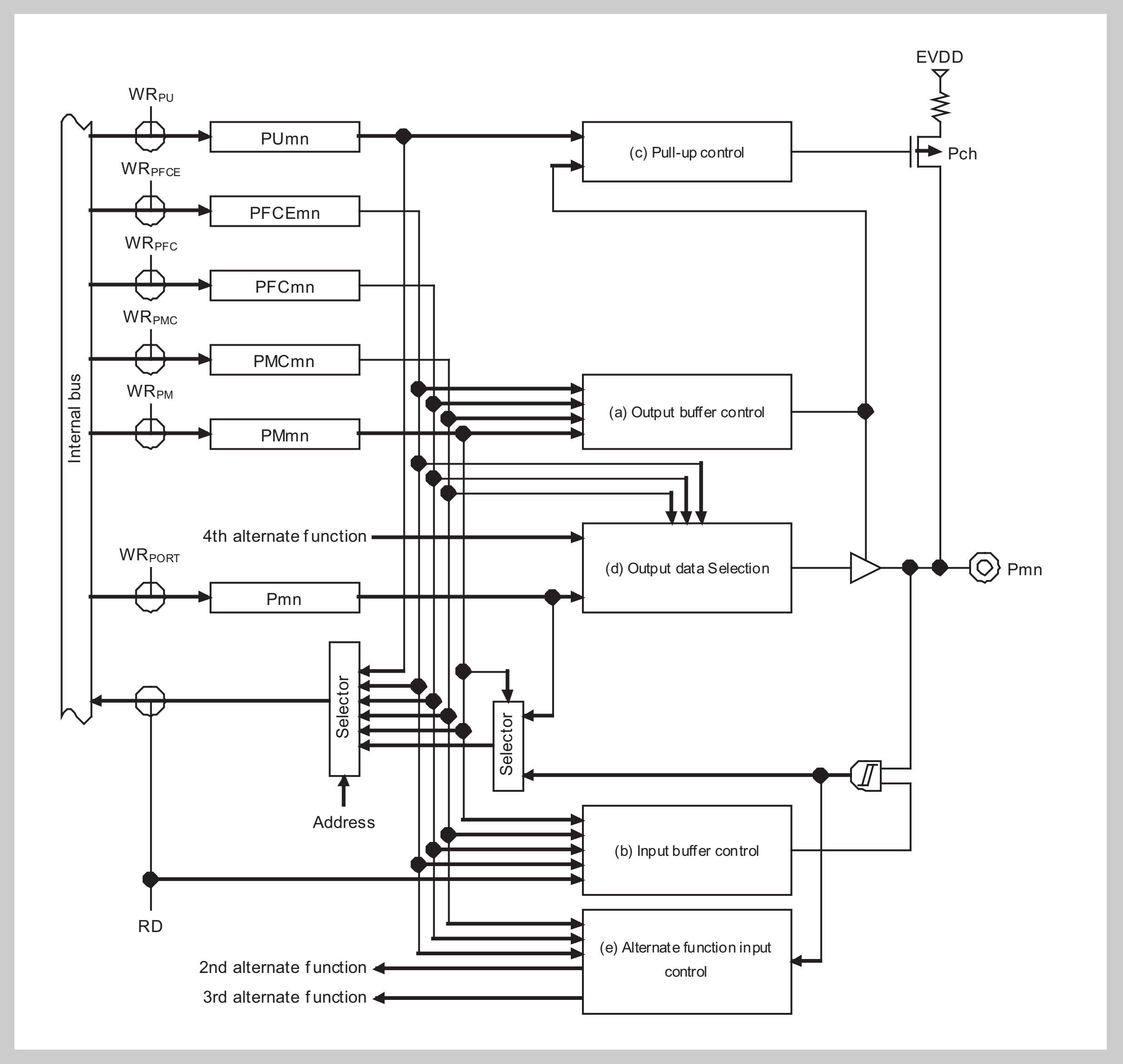


Figure 2-49 Port type Fx110-U block diagram

2.4.41 Port type Fx120-UFI

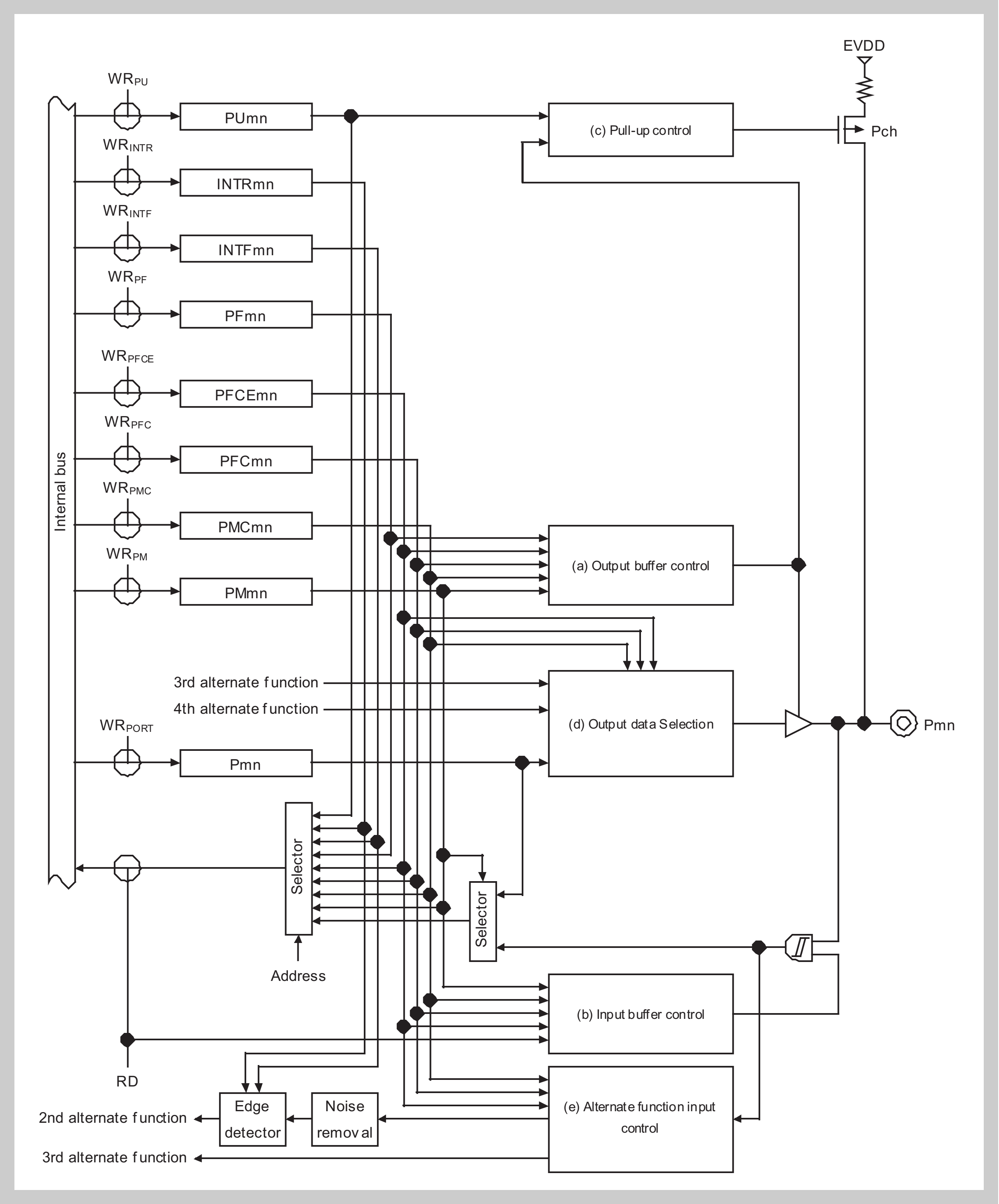


Figure 2-50 Port type Fx120-UFI block diagram

2.4.42 Port type Fx123-UFI

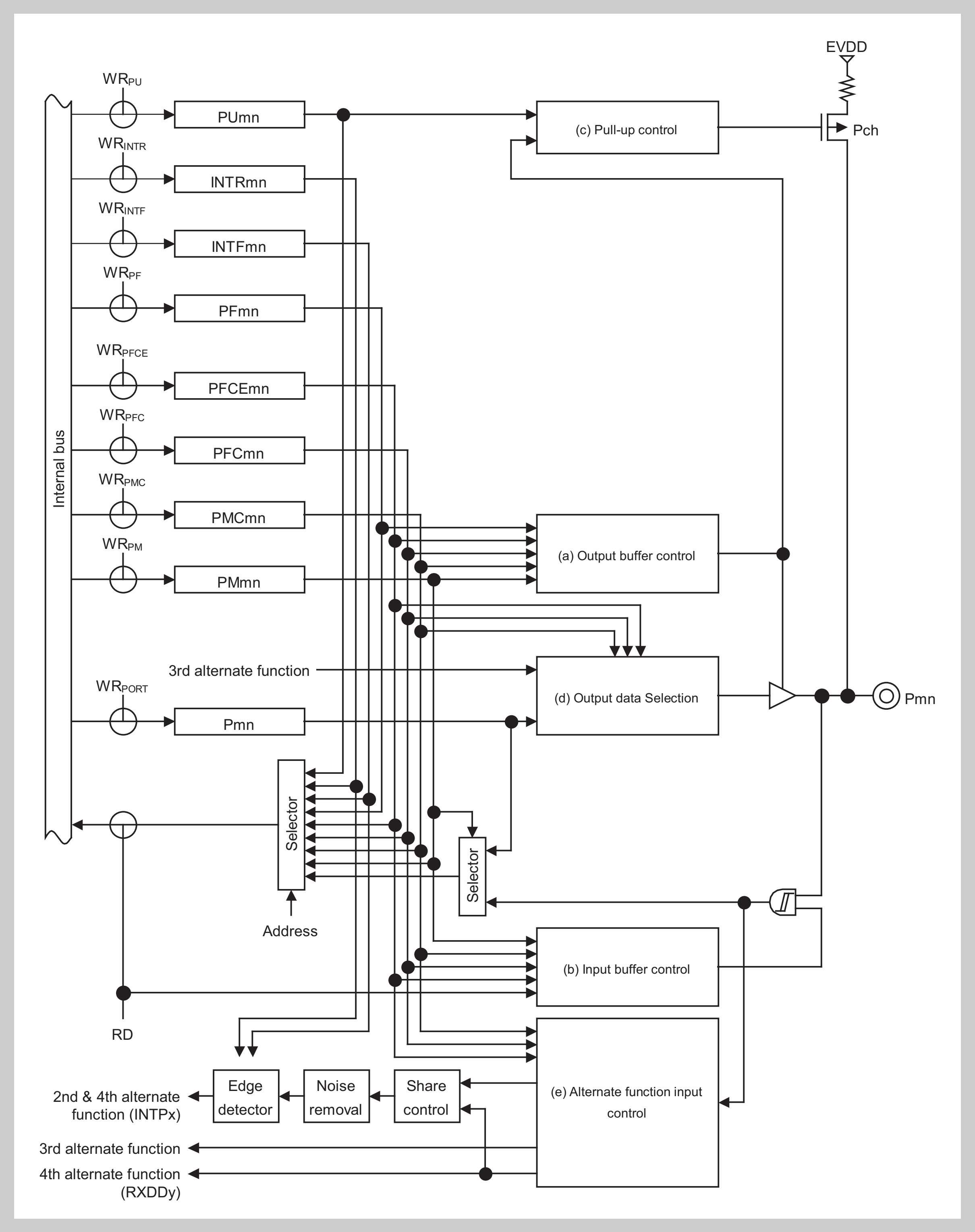


Figure 2-51 Port type Fx123-UFI block diagram

2.4.43 Port type Fx12x-UFI

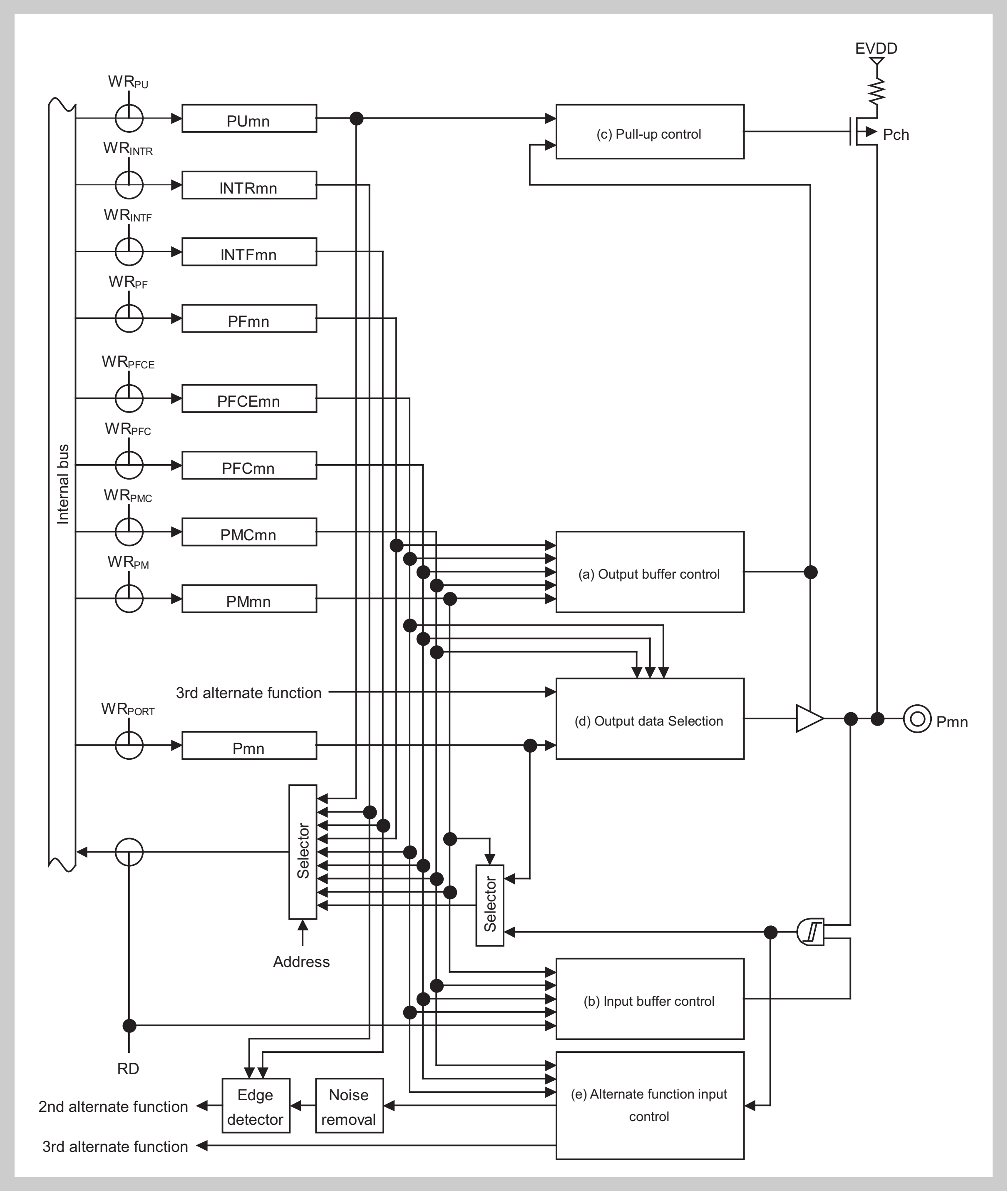


Figure 2-52 Port type Fx12x-UFI block diagram

2.4.44 Port type Fx13x-U

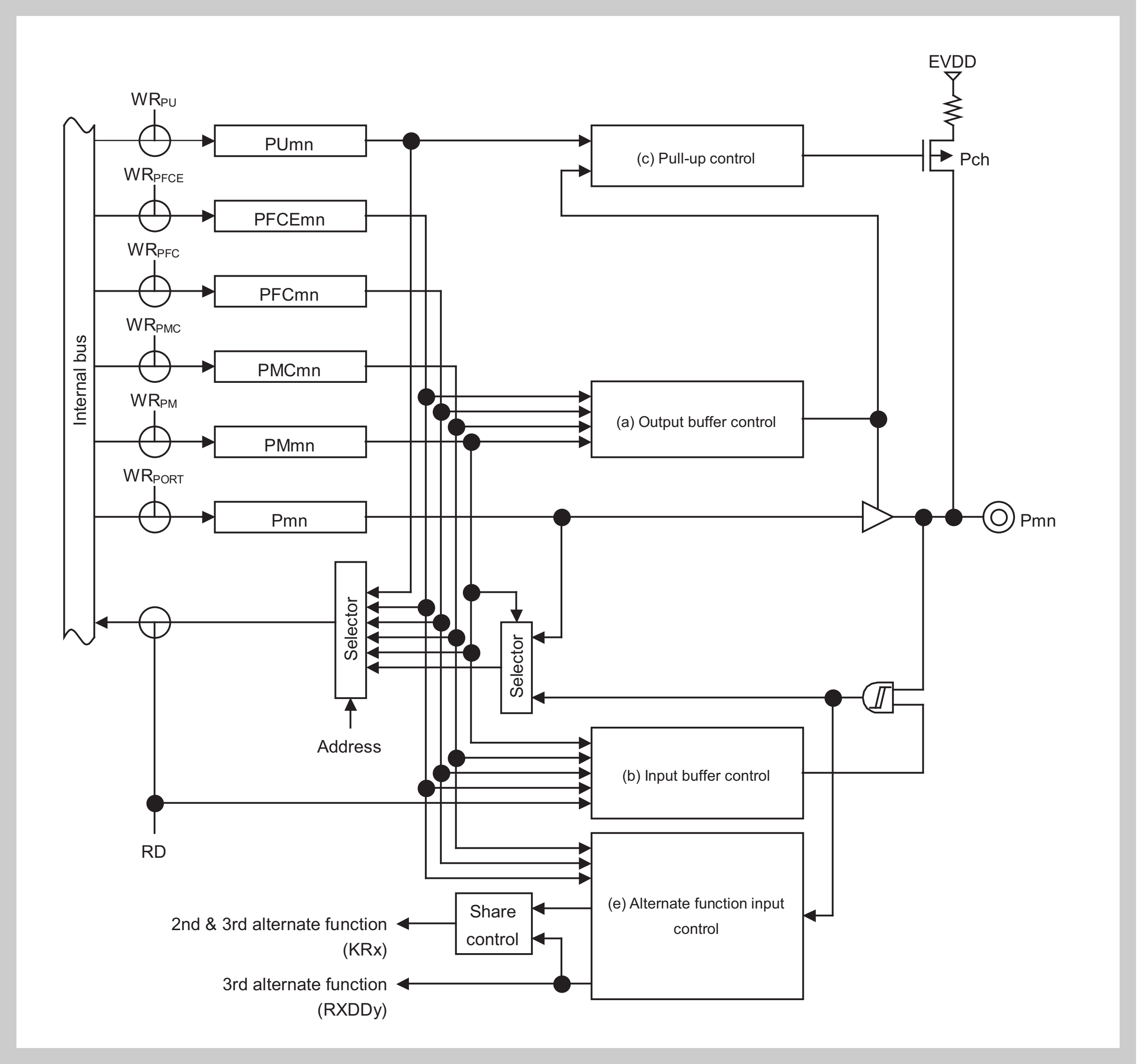


Figure 2-53 Port type Fx13x-U block diagram

2.4.45 Port type Fx210-U

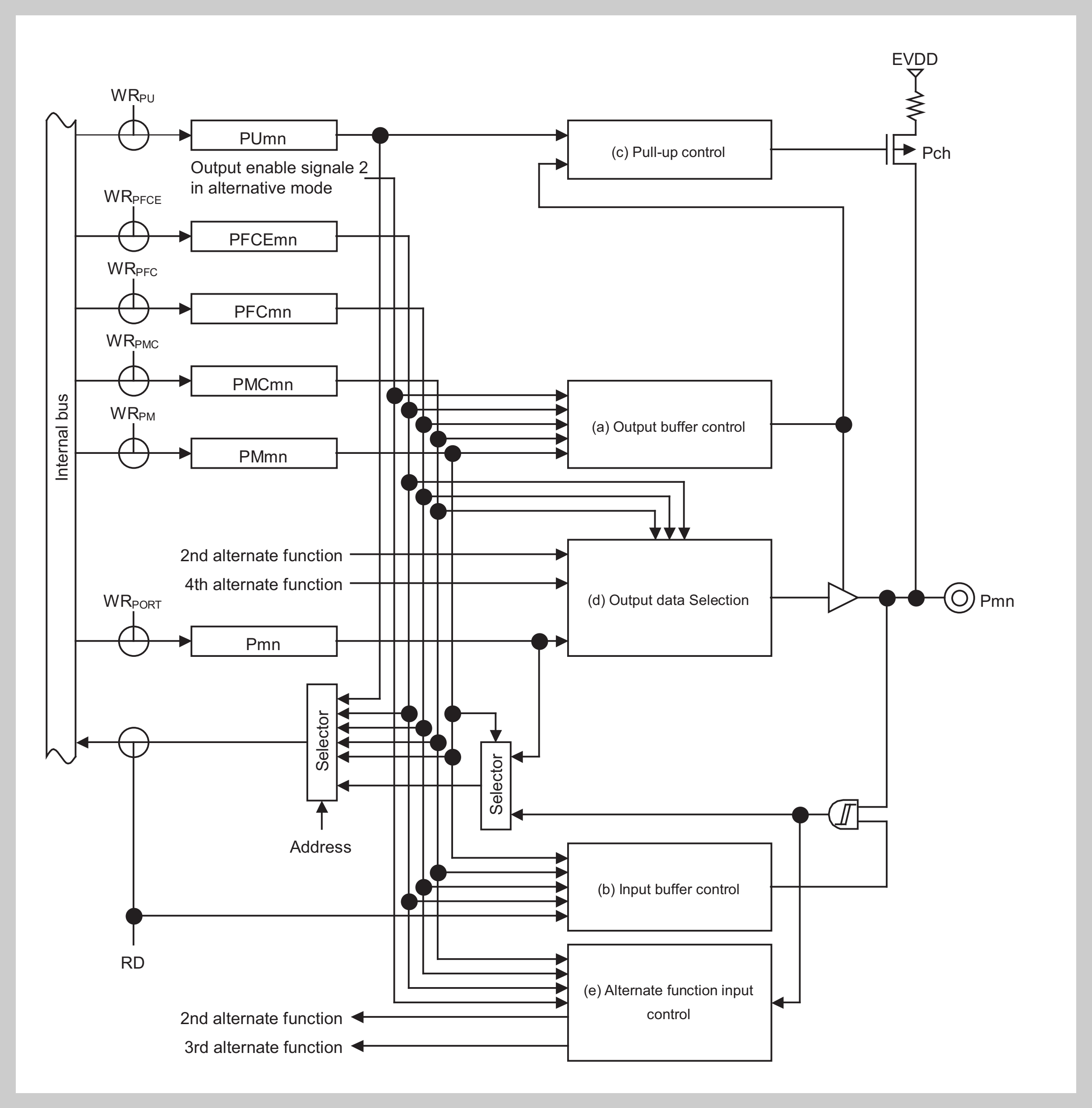


Figure 2-54 Port type Fx210-U block diagram

2.4.46 Port type Fx2x0-U

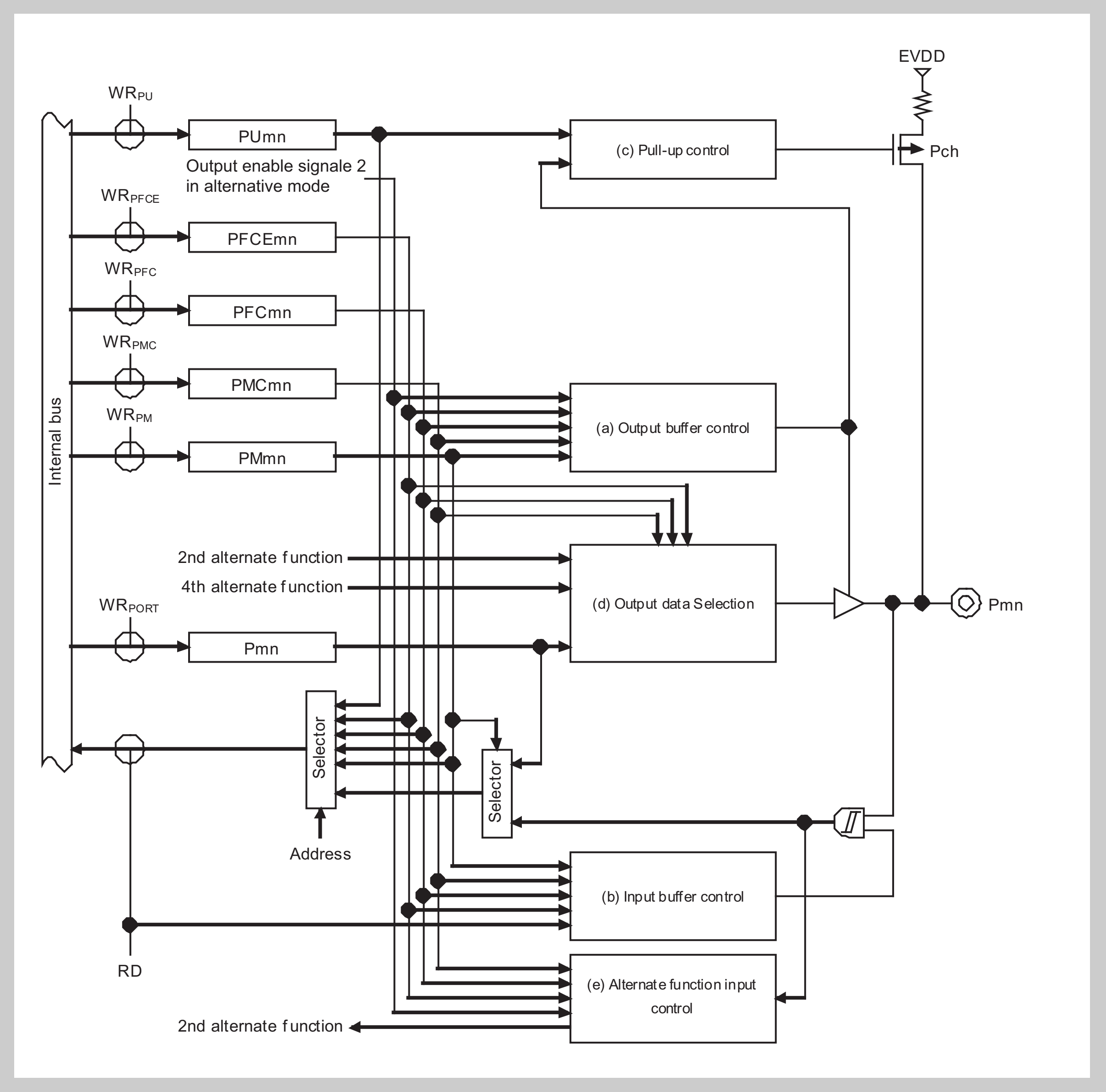


Figure 2-55 Port type Fx2x0-U block diagram

2.4.47 Port type Fxx10-U

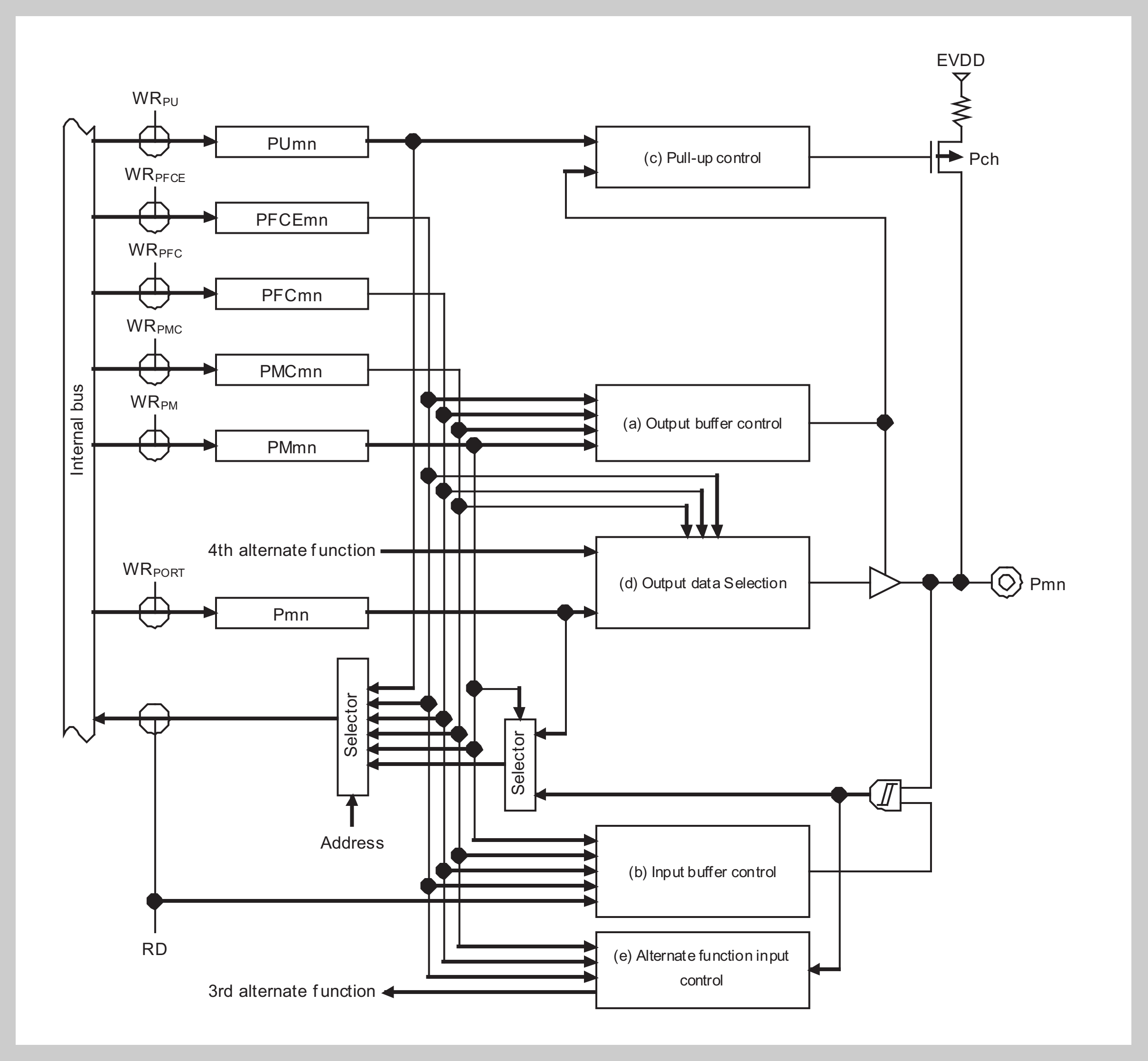


Figure 2-56 Port type Fxx10-U block diagram

2.4.48 Port type Fxx1x-U

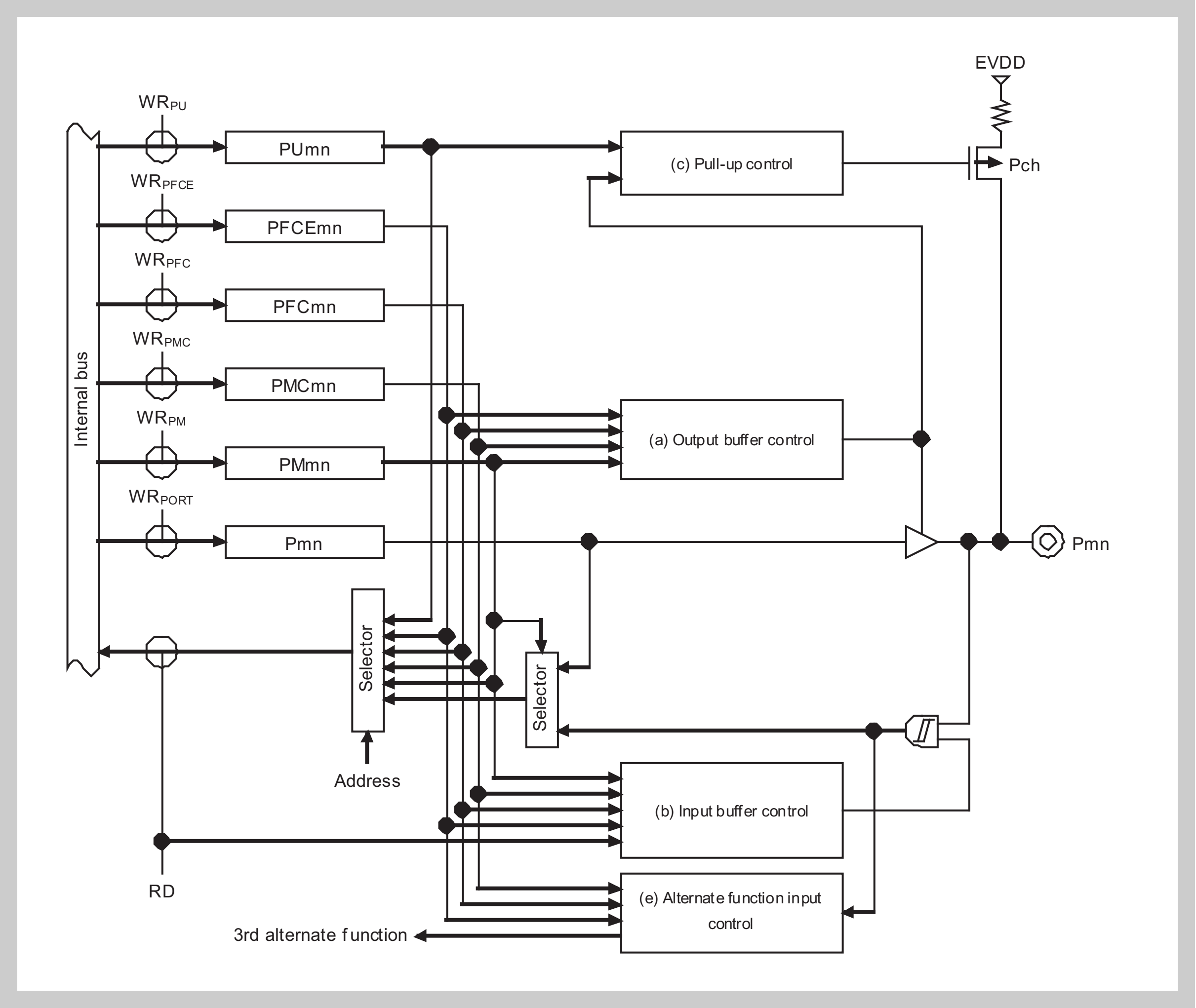


Figure 2-57 Port type Fxx1x-U block diagram

2.4.49 Port type Fxx2x-U

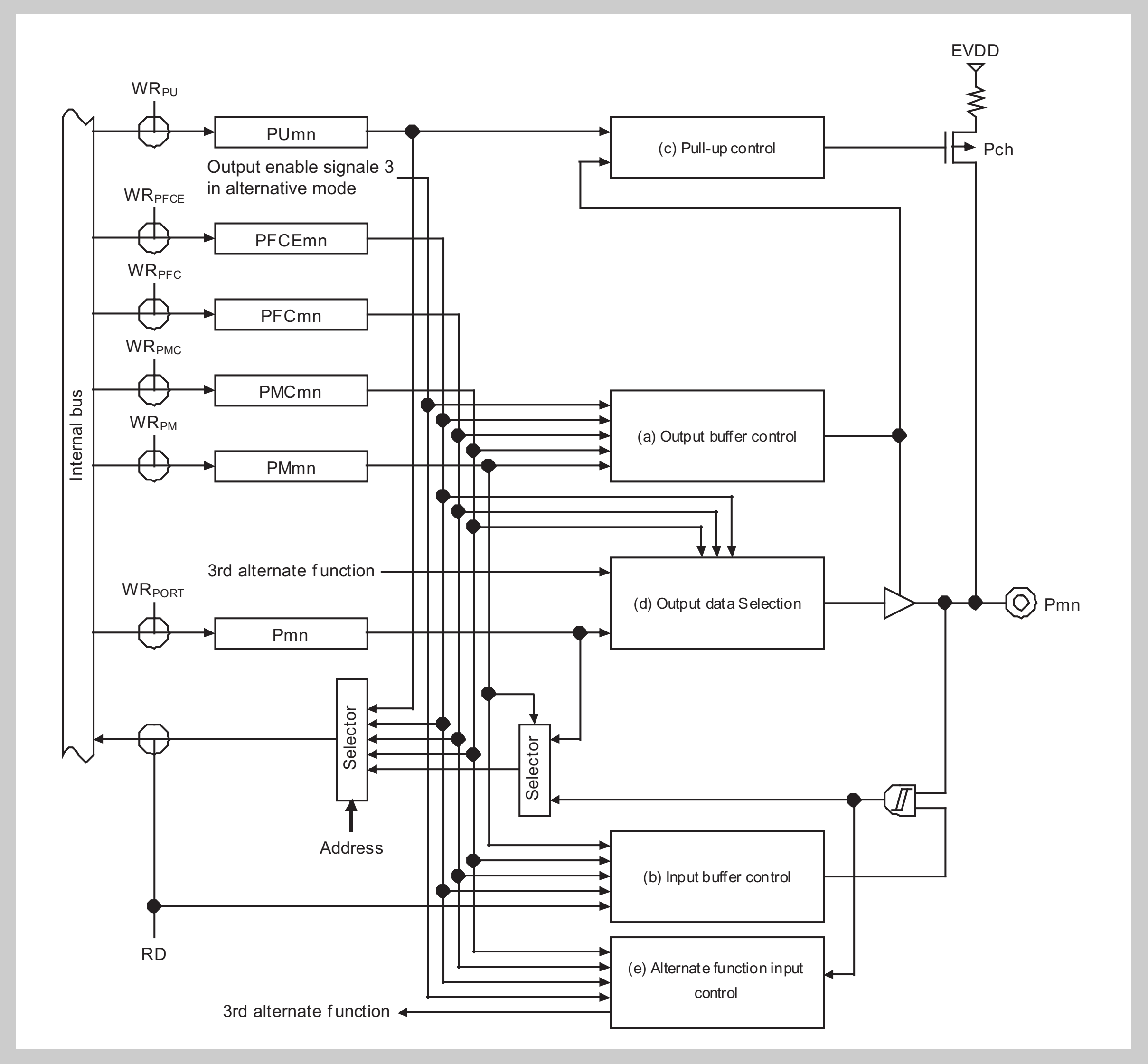


Figure 2-58 Port type Fxx2x-U block diagram

2.5 Port Group Configuration

This section provides an overview of the port groups (*Table 2-14*) and of the pin functions (*Table 2-14 on page 99*). In *Table 2-55 on page 148* it is listed how the pin functions change if the microcontroller is reset.

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. See *“Port group 0” on page 114* to *“Port group DL” on page 142*.

2.5.1 Port group configuration lists

Table 2-14 and *Table 2-15* provide overviews of the functions available at each port pin.

Table 2-14 V850ES/FE3, V850ES/FF3, V850ES/FG3 port group list (1/3)

Port group name	Port name	Alternative outputs	Alternative inputs	Buffer type
0	P00	TOAA31	TIAA31	5-W
	P01	TOAA30	TIAA30	5-W
	P02	TOAA40	NMI/TIAA40	5-W
	P03	TOAA41	INTP0/TIAA41/ADTRG	5-W
	P04	–	INTP1/CRXD0	5-W
	P05	–	INTP2/$\overline{\text{DRST}}$	5-AF
	P06	CTXD0	INTP3	5-W
1[a]	P10	–	INTP9	5-W
	P11	–	INTP10	5-W
3	P30	TXDD0	–	5-W
	P31	–	RXDD0/INTP7	5-W
	P32	TOAA00/TOAA01	ASCKD0/TIAA00	5-W
	P33	TOAA01/CTXD0	TIAA01	5-W
	P34	TOAA10	TIAA10/CRXD0	5-W
	P35	TOAA11	TIAA11	5-W
	P36[a]	CTXD1	–	5-W
	P37[a]	–	CRXD1	5-W
	P38[b]	TXDD2[a]	–	5-W
	P39[b]	–	RXDD2[a]/INTP8[a]	5-W
4	P40	–	SIB0/KR0/ RXDD3[c]/INTP14[c]	5-W
	P41	SOB0/TXDD3[c]	KR1	5-W
	P42	SCKB0	SCKB0/KR2	5-W

Table 2-14 V850ES/FE3, V850ES/FF3, V850ES/FG3 port group list (2/3)

Port group name	Port name	Alternative outputs	Alternative inputs	Buffer type
5	P50	TOAB01/TOAB0T1	KR0/TIAB01	5-W
	P51	TOAB02/TOAB0B1	KR1/TIAB02	5-W
	P52	TOAB03/TOAB0T2	KR2/TIAB03/DDI	5-W
	P53	TOAB00/TOAB0B2/ DDO	KR3/TIAB00	5-W
	P54	TOAB0T3	KR4/DCK	5-W
	P55	TOAB0B3	KR5/DMS	5-W
7	P70	–	ANI0	11-G
	P71	–	ANI1	11-G
	P72	–	ANI2	11-G
	P73	–	ANI3	11-G
	P74	–	ANI4	11-G
	P75	–	ANI5	11-G
	P76	–	ANI6	11-G
	P77	–	ANI7	11-G
	P78	–	ANI8	11-G
	P79	–	ANI9	11-G
	P710[b]	–	ANI10	11-G
	P711[b]	–	ANI11	11-G
	P712[a]	–	ANI12	11-G
	P713[a]	–	ANI13	11-G
	P714[a]	–	ANI14	11-G
	P715[a]	–	ANI15	11-G
9	P90	TXDD1	KR6	5-W
	P91	–	KR7/RXDD1	5-W
	P92[a]	TOAB11	TIAB11	5-W
	P93[a]	TOAB12	TIAB12	5-W
	P94[a]	TOAB13	TIAB13	5-W
	P95[a]	TOAB10	TIAB10	5-W
	P96	TOAA21	TIAA21	5-W
	P97	TOAA20	SIB1/TIAA20	5-W
	P98	SOB1/TOAB03	TIAB03	5-W
	P99	SCKB1/TOAB00	SCKB1/TIAB00	5-W
	P910[a]	–	–	5-W
	P911[a]	–	–	5-W
	P912[a]	–	–	5-W
	P913	PCL	INTP4	5-W
	P914	SDA00	SDA00/INTP5/RXDD4[c]	5-W
	P915	SCL00/TXDD4[c]	SCL00/INTP6	5-W

Table 2-14 V850ES/FE3, V850ES/FF3, V850ES/FG3 port group list (3/3)

Port group name	Port name	Alternative outputs	Alternative inputs	Buffer type
CM	PCM0	–	–	5
	PCM1	CLKOUT	–	5
	PCM2[b]	–	–	5
	PCM3[b]	–	–	5
CS[b]	PCS0	–	–	5
	PCS1	–	–	5
CT[b]	PCT0	–	–	5
	PCT1	–	–	5
	PCT4	–	–	5
	PCT6	–	–	5
DL	PDL0	–	–	5-K
	PDL1	–	–	5-K
	PDL2	–	–	5-K
	PDL3	–	–	5-K
	PDL4	–	–	5-K
	PDL5	–	FLMD1	5-K
	PDL6	–	–	5-K
	PDL7	–	–	5-K
	PDL8[b]	–	–	5-K
	PDL9[b]	–	–	5-K
	PDL10[b]	–	–	5-K
	PDL11[b]	–	–	5-K
	PDL12[a]	–	–	5-K
	PDL13[a]	–	–	5-K

a) V850ES/FG3 only

b) V850ES/FF3, V850ES/FG3 only

c) µPD70F3376A, µPD70F3377A of V850ES/FG3 only

Table 2-15 V850ES/FJ3, V850ES/FK3 port group list (1/4)

Port group name	Port name	Alternative outputs	Alternative inputs	Buffer type
0	P00	TOAA31	TIAA31	5-W
	P01	TOAA30	TIAA30	5-W
	P02	TOAA40	NMI/TIAA40	5-W
	P03	TOAA41	INTP0/TIAA41/ADTRG	5-W
	P04	–	INTP1/CRXD0	5-W
	P05	–	INTP2/$\overline{\text{DRST}}$	5-AF
	P06	CTXD0	INTP3	5-W
1	P10	–	INTP9	5-W
	P11	–	INTP10	5-W
2[a]	P20	–	ANI100	11-G
	P21	–	ANI101	11-G
	P22	–	ANI102	11-G
	P23	–	ANI103	11-G
	P24	–	ANI104	11-G
	P25	–	ANI105	11-G
	P26	–	ANI106	11-G
	P27	–	ANI107	11-G
	P28	–	ANI108	11-G
	P29	–	ANI109	11-G
	P210	–	ANI110	11-G
	P211	–	ANI111	11-G
	P212	–	ANI112	11-G
	P213	–	ANI113	11-G
	P214	–	ANI114	11-G
	P215	–	ANI115	11-G
3	P30	TXDD0	–	5-W
	P31	–	RXDD0/INTP7	5-W
	P32	TOAA00/TOAA01	ASCKD0/TIAA00	5-W
	P33	TOAA01/CTXD0	TIAA01	5-W
	P34	TOAA10	TIAA10/CRXD0	5-W
	P35	TOAA11	TIAA11	5-W
	P36	CTXD1	–	5-W
	P37	–	CRXD1	5-W
	P38	TXDD2	–	5-W
	P39	–	RXDD2/INTP8	5-W
4	P40	–	SIB0/KR0/ RXDD3[b]/INTP14[b]	5-W
	P41	SOB0/TXDD3[b]	KR1	5-W
	P42	SCKB0	SCKB0/KR2	5-W

Table 2-15 V850ES/FJ3, V850ES/FK3 port group list (2/4)

Port group name	Port name	Alternative outputs	Alternative inputs	Buffer type
5	P50	TOAB01/TOAB0T1	KR0/TIAB01	5-W
	P51	TOAB02/TOAB0B1	KR1/TIAB02	5-W
	P52	TOAB03/TOAB0T2	KR2/TIAB03/DDI	5-W
	P53	TOAB00/TOAB0B2/ DDO	KR3/TIAB00	5-W
	P54	TOAB0T3	KR4/DCK	5-W
	P55	TOAB0B3	KR5/DMS	5-W
6	P60	–	INTP11	5-W
	P61	–	INTP12	5-W
	P62	SOB3[c]/TXDD6[a]	INTP13	5-W
	P63	–	SIB3[c]/RXDD6[a]/ INTP13[a]	5-W
	P64	SCKB3[c]	SCKB3[c]	5-W
	P65	CTXD2	–	5-W
	P66	–	CRXD2	5-W
	P67	CTXD3[b]	–	5-W
	P68	–	CRXD3[b]	5-W
	P69	–	ADTRG1[a]	5-W
	P610	TOAB20	TIAB20	5-W
	P611	TOAB21	TIAB21	5-W
	P612	TOAB22	TIAB22	5-W
	P613	TOAB23	TIAB23	5-W
	P614	TXDD7[a]	–	5-W
	P615	–	RXDD7[a]/INTP15[a]	5-W
7	P70	–	ANI0	11-G
	P71	–	ANI1	11-G
	P72	–	ANI2	11-G
	P73	–	ANI3	11-G
	P74	–	ANI4	11-G
	P75	–	ANI5	11-G
	P76	–	ANI6	11-G
	P77	–	ANI7	11-G
	P78	–	ANI8	11-G
	P79	–	ANI9	11-G
	P710	–	ANI10	11-G
	P711	–	ANI11	11-G
	P712	–	ANI12	11-G
	P713	–	ANI13	11-G
	P714	–	ANI14	11-G
	P715	–	ANI15	11-G

Table 2-15 V850ES/FJ3, V850ES/FK3 port group list (3/4)

Port group name	Port name	Alternative outputs	Alternative inputs	Buffer type
8	P80	–	RXDD3[b]/INTP14	5-W
	P81	TXDD3[b]	–	5-W
9	P90	TXDD1	KR6	5-W
	P91	–	KR7/RXDD1	5-W
	P92	TOAB11	TIAB11	5-W
	P93	TOAB12	TIAB12	5-W
	P94	TOAB13	TIAB13	5-W
	P95	TOAB10	TIAB10	5-W
	P96	TOAA21	TIAA21	5-W
	P97	TOAA20	SIB1/TIAA20	5-W
	P98	SOB1/TOAB03	TIAB03	5-W
	P99	SCKB1/TOAB00	SCKB1/TIAB00	5-W
	P910	CTXD2	SIB2	5-W
	P911	SOB2	CRXD2	5-W
	P912	SCKB2/TXDD5[b]	SCKB2	5-W
	P913	PCL	INTP4/RXDD5[b]	5-W
	P914	SDA00	SDA00/INTP5/RXDD4[b]	5-W
	P915	SCL00/TXDD4[b]	SCL00/INTP6	5-W
12	P120	–	ANI16	11-G
	P121	–	ANI17	11-G
	P122	–	ANI18	11-G
	P123	–	ANI19	11-G
	P124	–	ANI20	11-G
	P125	–	ANI21	11-G
	P126	–	ANI22	11-G
	P127	–	ANI23	11-G
15[a]	P150	TOAA50	TIAA50	5-W
	P151	TOAA51	TIAA51	5-W
	P152	TOAA60	TIAA60	5-W
	P153	TOAA61	TIAA61	5-W
	P154	TOAA70	TIAA70	5-W
	P155	TOAA71	TIAA71	5-W
	P156	CTXD4	–	5-W
	P157	–	CRXD4	5-W
CD	PCD0	–	–	5
	PCD1	–	–	5
	PCD2	–	–	5
	PCD3	–	–	5

Table 2-15 V850ES/FJ3, V850ES/FK3 port group list (4/4)

Port group name	Port name	Alternative outputs	Alternative inputs	Buffer type
CM	PCM0	–	$\overline{\text{WAIT}}$	5
	PCM1	CLKOUT	–	5
	PCM2	$\overline{\text{HLDAK}}$	–	5
	PCM3	–	$\overline{\text{HLDRQ}}$	5
	PCM4	–	–	5
	PCM5	–	–	5
CS	PCS0	$\overline{\text{CS0}}$	–	5
	PCS1	$\overline{\text{CS1}}$	–	5
	PCS2	$\overline{\text{CS2}}$	–	5
	PCS3	$\overline{\text{CS3}}$	–	5
	PCS4	–	–	5
	PCS5	–	–	5
	PCS6	–	–	5
	PCS7	–	–	5
CT	PCT0	$\overline{\text{WR0}}$	–	5
	PCT1	$\overline{\text{WR1}}$	–	5
	PCT2	–	–	5
	PCT3	–	–	5
	PCT4	$\overline{\text{RD}}$	–	5
	PCT5	–	–	5
	PCT6	ASTB	–	5
	PCT7	–	–	5
DL	PDL0	AD0	AD0	5-K
	PDL1	AD1	AD1	5-K
	PDL2	AD2	AD2	5-K
	PDL3	AD3	AD3	5-K
	PDL4	AD4	AD4	5-K
	PDL5	AD5	AD5/FLMD1	5-K
	PDL6	AD6	AD6	5-K
	PDL7	AD7	AD7	5-K
	PDL8	AD8	AD8	5-K
	PDL9	AD9	AD9	5-K
	PDL10	AD10	AD10	5-K
	PDL11	AD11	AD11	5-K
	PDL12	AD12	AD12	5-K
	PDL13	AD13	AD13	5-K
	PDL14	AD14	AD14	5-K
	PDL15	AD15	AD15	5-K

[a] V850ES/FK3 only

[b] not available with µPD70F3378 of V850ES/FJ3

[c] not available with µPD70F3378, µPD70F3379, µPD70F3380 of V850ES/FJ3

2.5.2 Alphabetic pin function list

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

The table does not list differences between the various devices of the V850ES/ Fx3. These are listed in *Table 2-14 on page 99* and *Table 2-15 on page 102*.

Table 2-16 Alphabetic pin functions list (1/7)

Pin name	I/O	Pin function	Port	Pin number				
				FE3	FF3	FG3	FJ3	FK3
AD0	I/O	External memory interface address/data lines	PDL0	–	–	–	105	116
AD1			PDL1	–	–	–	106	117
AD2			PDL2	–	–	–	107	118
AD3			PDL3	–	–	–	108	119
AD4			PDL4	–	–	–	109	120
AD5			PDL5	–	–	–	110	121
AD6			PDL6	–	–	–	111	122
AD7			PDL7	–	–	–	112	123
AD8			PDL8	–	–	–	113	129
AD9			PDL9	–	–	–	114	130
AD10			PDL10	–	–	–	115	131
AD11			PDL11	–	–	–	116	132
AD12			PDL12	–	–	–	117	133
AD13			PDL13	–	–	–	118	134
AD14			PDL14	–	–	–	119	135
AD15			PDL15	–	–	–	120	136
ADTRG	I	A/D Converter 0 external trigger input	P03	15	6	18	18	19
ADTRG1	I	A/D Converter 1 external trigger input	P69	–	–	–	–	71

Table 2-16 Alphabetic pin functions list (2/7)

Pin name	I/O	Pin function	Port	Pin number				
				FE3	FF3	FG3	FJ3	FK3
ANI0	I	A/D Converter 0 input 0 to 23	P70	64	80	100	144	176
ANI1			P71	63	79	99	143	175
ANI2			P72	62	78	98	142	174
ANI3			P73	61	77	97	141	173
ANI4			P74	60	76	96	140	172
ANI5			P75	59	75	95	139	171
ANI6			P76	58	74	94	138	170
ANI7			P77	57	73	93	137	169
ANI8			P78	56	72	92	136	168
ANI9			P79	55	71	91	135	167
ANI10			P710	–	70	90	134	166
ANI11			P711	–	69	89	133	165
ANI12			P712	–	–	88	132	164
ANI13			P713	–	–	87	131	163
ANI14			P714	–	–	86	130	162
ANI15			P715	–	–	85	129	161
ANI16			P120	–	–	–	128	160
ANI17			P121	–	–	–	127	159
ANI18			P122	–	–	–	126	158
ANI19			P123	–	–	–	125	157
ANI20			P124	–	–	–	124	156
ANI21			P125	–	–	–	123	155
ANI22			P126	–	–	–	122	154
ANI23			P127	–	–	–	121	153
ANI100	I	A/D Converter 1 input 0 to 15	P20	–	–	–	–	44
ANI101			P21	–	–	–	–	43
ANI102			P22	–	–	–	–	42
ANI103			P23	–	–	–	–	41
ANI104			P24	–	–	–	–	40
ANI105			P25	–	–	–	–	39
ANI106			P26	–	–	–	–	38
ANI107			P27	–	–	–	–	37
ANI108			P28	–	–	–	–	36
ANI109			P29	–	–	–	–	35
ANI110			P210	–	–	–	–	34
ANI111			P211	–	–	–	–	33
ANI112			P212	–	–	–	–	32
ANI113			P213	–	–	–	–	31
ANI114			P214	–	–	–	–	30
ANI115			P215	–	–	–	–	29

Table 2-16 Alphabetic pin functions list (3/7)

Pin name	I/O	Pin function	Port	Pin number				
				FE3	FF3	FG3	FJ3	FK3
ASCKD0	I	UARTD0 baud rate clock input	P32	24	24	27	27	48
ASTB	O	External memory interface address strobe	PCT6	–	–	–	101	143
AVREF0	–	A/D Converter 0 reference voltage input	–	1	1	1	1	1
AVREF1	–	A/D Converter 1 reference voltage input	–	–	–	–	–	45
AVSS	–	A/D Converter 0 ground	–	2	2	2	2	2
AVSS1	–	A/D Converter 1 ground	–	–	–	–	–	46
BVDD	–	I/O buffer supply voltage	–	–	–	70	104	128
BVSS	–	I/O buffer supply ground	–	–	–	69	103	127
CLKOUT	O	CPU system clock output	PCM1	46	50	62	86	111
CRXD0	I	CAN0 to CAN4 receive data	P04	16	7	19	19	20
			P34	26	26	29	29	50
CRXD1			P37	–	–	32	32	53
CRXD2			P66	–	–	–	49	68
			P911				72	93
CRXD3			P68	–	–	–	51	70
CRXD4			P157	–	–	–	–	105
$\overline{\text{CS0}}$	O	External memory interface chip select signals	PCS0	–	–	–	81	106
$\overline{\text{CS1}}$			PCS1	–	–	–	82	107
$\overline{\text{CS2}}$			PCS2	–	–	–	83	108
$\overline{\text{CS3}}$			PCS3	–	–	–	84	109
CTXD0	O	CAN0 to CAN4 transmit data	P06	18	18	21	21	22
			P33	25	25	28	28	49
CTXD1			P36	–	–	31	31	52
CTXD2			P65	–	–	–	48	67
			P910				71	92
CTXD3			P67	–	–	–	50	69
CTXD4			P156	–	–	–	–	104
DCK	I	N-Wire interface clock	P54	34	36	41	41	60
DDI	I	N-Wire interface debug data input	P52	30	34	39	39	58
DDO	O	N-Wire interface debug data output	P53	31	35	40	40	59
DMS	I	N-Wire interface debug mode select input	P55	35	37	42	42	61
$\overline{\text{DRST}}$	I	N-Wire debug interface reset	P05	17	17	20	20	21
EVDD	–	Port buffer supply voltage	–	33	31	5, 34	5, 34,	5, 47, 77
EVSS	–	Port buffer supply voltage	–	32	30	33	33	28, 76
FLMD0	–	Flash programming mode setting pin	–	3	8	8	8	8
FLMD1	I	Flash programming mode setting pin	PDL5	52	62	76	110	121
$\overline{\text{HLDAK}}$	O	Bus hold acknowledge output	PCM2	–	–	–	87	112
$\overline{\text{HLDRQ}}$	I	Bus hold request input	PCM3	–	–	–	88	113

Table 2-16 Alphabetic pin functions list (4/7)

Pin name	I/O	Pin function	Port	Pin number				
				FE3	FF3	FG3	FJ3	FK3
INTP0	I	External interrupts INTP0 - INTP15	P03	15	6	18	18	19
INTP1			P04	16	7	19	19	20
INTP2			P05	17	17	20	20	21
INTP3			P06	18	18	21	21	22
INTP4			P913	42	44	56	74	95
INTP5			P914	43	45	57	75	96
INTP6			P915	44	46	58	76	97
INTP7			P31	23	23	26	26	27
INTP8			P39	–	–	36	36	55
INTP9			P10	–	–	3	3	3
INTP10			P11	–	–	4	4	4
INTP11			P60	–	–	–	43	62
INTP12			P61	–	–	–	44	63
INTP13			P62	–	–	–	45	64
			P63				–	65
INTP14			P40	–	–	22	22	23
			P80			–	59	80
INTP15			P615	–	–	–	–	79
KR0	I	Key interrupt KR0 - KR7	P40	19	19	22	22	23
			P50	28	32	37	37	56
KR1			P41	20	20	23	23	24
			P51	29	33	38	38	57
KR2			P42	21	21	24	24	25
			P52	30	34	39	39	58
KR3			P53	31	35	40	40	59
KR4			P54	34	36	41	41	60
KR5			P55	35	37	42	42	61
KR6			P90	36	38	43	61	82
KR7			P91	37	39	44	62	83
NMI	I	Non-maskable interrupt	P02	14	5	17	17	18
PCL	O	Programmable clock output	P913	42	44	56	74	95
$\overline{\text{RD}}$	O	External memory interface read strobe	PCT4	–	–	–	99	141
REGC	–	External voltage regulator capacitor connection	–	5	10	10	10	10
REGC1	–	External voltage regulator capacitor connection	–	–	–	–	–	125
$\overline{\text{RESET}}$	I	Reset input	–	9	14	14	14	14

Table 2-16 Alphabetic pin functions list (5/7)

Pin name	I/O	Pin function	Port	Pin number				
				FE3	FF3	FG3	FJ3	FK3
RXDD0	I	UARTD0-7 receive data	P31	23	23	26	26	27
RXDD1			P91	37	39	44	62	83
RXDD2			P39	–	–	36	36	55
RXDD3			P40	–	–	22	22	23
			P80			–	59	80
RXDD4			P914	–	–	57	75	96
RXDD5			P913	–	–	–	74	95
RXDD6			P63	–	–	–	–	65
RXDD7			P615	–	–	–	–	79
SCKB0	I/O	Clocked Serial Interface 0-3 clock lines	P42	21	21	24	24	25
SCKB1			P99	41	43	52	70	91
SCKB2			P912	–	–	–	73	94
SCKB3			P64	–	–	–	47	66
SCL00	I/O	I^{2}C0 clock line	P915	44	46	58	76	97
SDA00	I/O	I^{2}C0 data line	P914	43	45	57	75	96
SIB0	I	Clocked Serial Interface 0-3 data input	P40	19	19	22	22	23
SIB1			P97	39	41	50	68	89
SIB2			P910	–	–	–	71	92
SIB3			P63	–	–	–	46	65
SOB0	O	Clocked Serial Interface 0-3 data output	P41	20	20	23	23	24
SOB1			P98	40	42	51	69	90
SOB2			P911	–	–	–	72	93
SOB3			P62	–	–	–	45	64
TIAA00	I	Timer TAA0-7 channel 0 capture trigger input	P32	24	24	27	27	48
TIAA10			P34	26	26	29	29	50
TIAA20			P97	39	41	50	68	89
TIAA30			P01	13	4	7	7	7
TIAA40			P02	14	5	17	17	18
TIAA50			P150	–	–	–	–	98
TIAA60			P152	–	–	–	–	100
TIAA70			P154	–	–	–	–	102
TIAA01	I	Timer TAA0-7 channel 1 capture trigger input	P33	25	25	28	28	49
TIAA11			P35	27	27	30	30	51
TIAA21			P96	38	40	49	67	88
TIAA31			P00	12	3	6	6	6
TIAA41			P03	15	6	18	18	19
TIAA51			P151	–	–	–	–	99
TIAA61			P153	–	–	–	–	101
TIAA71			P155	–	–	–	–	103

Table 2-16 Alphabetic pin functions list (6/7)

Pin name	I/O	Pin function	Port	Pin number				
				FE3	FF3	FG3	FJ3	FK3
TIAB00	I	Timer TAB0-2 channel 0 capture trigger input	P53	31	35	40	40	59
			P99	41	43	52	70	91
TIAB10			P95	–	–	48	66	87
TIAB20			P610	–	–	–	53	72
TIAB01	I	Timer TAB0-2 channel 1 capture trigger input	P50	28	32	37	37	56
TIAB11			P92	–	–	45	63	84
TIAB21			P611	–	–	–	54	73
TIAB02	I	Timer TAB0-2 channel 2 capture trigger input	P51	29	33	38	38	57
TIAB12			P93	–	–	46	64	85
TIAB22			P612	–	–	–	55	74
TIAB03	I	Timer TAB0-2 channel 3 capture trigger input	P52	30	34	39	39	58
			P98	40	42	51	69	90
TIAB13			P94	–	–	47	65	86
TIAB23			P613	–	–	–	56	75
TOAA00	O	Timer TAA0-7 channel 0 signal output	P32	24	24	27	27	48
TOAA10			P34	26	26	29	29	50
TOAA20			P97	39	41	50	68	89
TOAA30			P01	13	4	7	7	7
TOAA40			P02	14	5	17	17	18
TOAA50			P150	–	–	–	–	98
TOAA60			P152	–	–	–	–	100
TOAA70			P154	–	–	–	–	102
TOAA01	O	Timer TAA0-7 channel 1 signal output	P32	24	24	27	27	48
			P33	25	25	28	28	49
TOAA11			P35	27	27	30	30	51
TOAA21			P96	38	40	49	67	88
TOAA31			P00	12	3	6	6	6
TOAA41			P03	15	6	18	18	19
TOAA51			P151	–	–	–	–	99
TOAA61			P153	–	–	–	–	101
TOAA71			P155	–	–	–	–	103
TOAB00	O	Timer TAB0-2 channel 0 capture trigger output	P53	31	35	40	40	59
			P99	41	43	52	70	91
TOAB10			P95	–	–	48	66	87
TOAB20			P610	–	–	–	53	72
TOAB01	O	Timer TAB0-2 channel 1 capture trigger output	P50	28	32	37	37	56
TOAB11			P92	–	–	45	63	84
TOAB21			P611	–	–	–	54	73

Table 2-16 Alphabetic pin functions list (7/7)

Pin name	I/O	Pin function	Port	Pin number				
				FE3	FF3	FG3	FJ3	FK3
TOAB02	O	Timer TAB0-2 channel 2 capture trigger output	P51	29	33	38	38	57
TOAB12			P93	–	–	46	64	85
TOAB22			P612	–	–	–	55	74
TOAB03	O	Timer TAB0-2 channel 3 capture trigger output	P52	30	34	39	39	58
			P98	40	42	51	69	90
TOAB13			P94	–	–	47	65	86
TOAB23			P613	–	–	–	56	75
TOAB0B1	O	Motor Control output signal	P51	29	33	38	38	57
TOAB0B2			P53	31	35	40	40	59
TOAB0B3			P55	35	37	42	42	61
TOAB0T1	O	Motor Control output signal	P50	28	32	37	37	56
TOAB0T2			P52	30	34	39	39	58
TOAB0T3			P54	34	36	41	41	60
TXDD0	O	UARTD0-7 transmit data	P30	22	22	25	25	26
TXDD1			P90	36	38	43	61	82
TXDD2			P38	–	–	35	35	54
TXDD3			P41	–	–	23	23	24
			P81			–	60	81
TXDD4			P915	–	–	58	76	97
TXDD5			P912	–	–	–	73	94
TXDD6			P62	–	–	–	–	64
TXDD7			P614	–	–	–	–	78
VDD	–	Core supply voltage	–	4	9	9	9	9
VDD1	–	Core supply voltage	–	–	–	70	104	126
VSS	–	Core supply ground	–	6	11	11	11	11
VSS1	–	Core supply ground	–	–	–	69	103	124
$\overline{\text{WAIT}}$	I	External memory interface data wait request	PCM0	–	–	–	85	110
$\overline{\text{WR0}}$	O	External memory interface write strobe (lower 8 bits)	PCT0	–	–	–	95	137
$\overline{\text{WR1}}$	O	External memory interface write strobe (higher 8 bits)	PCT1	–	–	–	96	138
X1	I	Main clock resonator connection	–	7	12	12	12	12
X2	–	Main clock resonator connection	–	8	13	13	13	13
XT1	I	Sub oscillator resonator connection	–	10	15	15	15	15
XT2	–	Sub oscillator resonator connection	–	11	16	16	16	16

Note The following alternative functions are provided on two pins each:

Unit	Alternative function	I/O	Port 1	Port 2
Timer	TOAA01	O	P33	P32
	TIAB00	I	P53	P99
	TOAB00	O	P53	P99
	TIAB03	I	P52	P98
	TOAB03	O	P52	P98
UARTD	RXDD3	I	P40	P80[a]
	TXDD3	O	P41	P81[a]
CAN	CTXD0	O	P06	P33
	CRXD0	I	P04	P34
	CTXD2[b]	O	P65	P910
	CRXD2[b]	I	P66	P911
External Interrupt	INTP13	I	P62	P63
	INTP14	I	P40[a]	P80[b]
Key interrupt	KR0	I	P40	P50
	KR1	I	P41	P51
	KR2	I	P42	P52

[a] not available for V850ES/FE3, V850ES/FF3, μPD70F3374 and μPD70F3375 of V850ES/FG3 and μPD70F3378 of V850ES/FJ3

[b] available on V850ES/FJ3 and V850ES/FK3

Thus you can select on which pin the alternative function should appear. Refer to *“Pin function configuration” on page 39*.

Caution Make sure an alternative input function is only supplied from a single pin at the same time. An alternative output function can be output on several pins concurrently.
For example, if P40 operates as key interrupt KR0, P50 must not operate as key interrupt KR0.

2.5.3 Port group 0

Port group 0 is a 7-bit port group. In alternative mode, it comprises pins for the following functions:

- External interrupt (INTP0 to INTP3)
- Non-maskable interrupt (NMI)
- N-Wire debug interface reset ($\overline{\text{DRST}}$)
- A/D Converter 0 external trigger input (ADTRG)
- Timer TAA3 channels (TIAA30, TIAA31 and TOAA30, TOAA31)
- Timer TAA4 channels (TIAA40, TIAA41 and TOAA40, TOAA41)
- CAN0 transmit/receive data (CTXD0, CRXD0)

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	Port type	Noise filter[a]	Input charact.[b]
Port mode (PMC = 0)	Alternative mode (PMCnm = 1)				On-chip debug mode (OCDM0 = 1)				
	PFCE = 0		PFCE = 1						
	Function 1 PFC = 0	Function 2 PFC = 1	Function 3 PFC = 0	Function 4 PFC = 1					
P00	TIAA31 (I)	TOAA31 (O)	–	–	–	P00 (I)	E10-U	A	S2
P01	TIAA30 (I)	TOAA30 (O)	–	–	–	P01 (I)	E10-U	A	S2
P02	NMI (I)	prohibited	TIAA40 (I)	TOAA40 (O)	–	P02 (I)	F1x10-UI	A	S2
P03	INTP0 (I)	ADTRG (I)	TIAA41 (I)	TOAA41 (O)	–	P03 (I)	F1110-UI	A	S2
P04	INTP1 (I)	CRXD0 (I)	–	–	–	P04 (I)	E11-UI	A	S1
P05	INTP2 (I)	–	–	–	$\overline{\text{DRST}}$ (I)	P05 (I) or $\overline{\text{DRST}}$ (I)[c]	D101-UI	A	S2
P06	INTP3 (I)	CTXD0 (O)	–	–	–	P06 (I)	E10-UI	B	S2

[a] A: analog noise filter only for TIAAnm, NMI, INTPn, $\overline{\text{DRST}}$, ADTRG inputs
B: analog and digital noise filter
–: no noise filter

[b] S1: Schmitt trigger (30/70%); S2: Schmitt trigger (40/80%); x: CMOS

[c] The pin function after reset depends on the reset source, that means on bit OCDM.OCDM0. Refer to *“OCDM - On-chip debug mode register” on page 44* and to *“On-Chip Debug Unit” on page 930*.

Note

1. Alternative functions CRXD0 and CTXD0 are provided on two pins each. Thus you can select on which pin the alternative function should appear. Refer to *“Alphabetic pin function list” on page 106*.
2. Setting of PU05 is valid only when the OCDM0 bit of the OCDM register = 0. It is not pulled up when the OCDM bit = 1.

Table 2-18 Port group 0: configuration registers

Register	Address	Initial value	Used bits							
PMC0	FFFF F440$_H$	00$_H$	X	PMC06	PMC05	PMC04	PMC03	PMC02	PMC01	PMC00
PM0	FFFF F420$_H$	FF$_H$	X	PM06	PM05	PM04	PM03	PM02	PM01	PM00
PFC0	FFFF F460$_H$	00$_H$	X	PFC06	X	PFC04	PFC03	PFC02	PFC01	PFC00
PFCE0	FFFF F700$_H$	00$_H$	X	X	X	X	PFCE03	PFCE02	X	X
OCDM	FFFF F9FC$_H$	00$_H$ / 01$_H$[a]	0	0	0	0	0	0	0	OCDM0
P0	FFFF F400$_H$	undefined	X	P06	P05	P04	P03	P02	P01	P00
PU0	FFFF FC40$_H$	00$_H$	X	PU06	PU05	PU04	PU03	PU02	PU01	PU00

[a] Depends on the reset source (Refer to *"OCDM - On-chip debug mode register" on page 44* and to *"On-Chip Debug Unit" on page 930*)

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

2.5.4 Port group 1 (V850ES/FG3, V850ES/FJ3, V850ES/FK3)

Note Port group 1 is available only for V850ES/FG3, V850ES/FJ3, and V850ES/FK3.

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

- External interrupt (INTP9 and INTP10)

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	Port type	Noise filter[a]	Input charact.[b]
Port mode (PMCnm = 0)	Alternative mode (PMCnm = 1)				
P10	INTP9 (I)	P10 (I)	D1-UI	A	S2
P11	INTP10 (I)	P11 (I)	D1-UI	A	S2

[a] A: analog noise filter only for INTPn inputs
B: analog and digital noise filter
–: no noise filter

[b] S1: Schmitt trigger (30/70%); S2: Schmitt trigger (40/80%); x: CMOS

Table 2-20 Port group 1: configuration registers

Register	Address	Initial value	Used bits							
PMC1	FFFF F442$_H$	00$_H$	X	X	X	X	X	X	PMC11	PMC10
PM1	FFFF F422$_H$	FF$_H$	X	X	X	X	X	X	PM11	PM10
P1	FFFF F402$_H$	undefined	X	X	X	X	X	X	P11	P10
PU1	FFFF FC42$_H$	00$_H$	X	X	X	X	X	X	PU11	PU10

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

2.5.5 Port group 2 (V850ES/FK3)

Note Port group 2 is available only for V850ES/FK3.

Port group 2 is a 16-bit port group. In alternative mode, it comprises pins for the following functions:

- A/D Converter 1 inputs

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	Port type	Noise filter[a]	Input charact.[b]
Port mode (PMCnm = 0)	Alternative mode (PMCnm = 1)				
P20	ANI100 (I)	P20 (I)	D1A	–	x
P21	ANI101 (I)	P21 (I)	D1A	–	x
P22	ANI102 (I)	P22 (I)	D1A	–	x
P23	ANI103 (I)	P23 (I)	D1A	–	x
P24	ANI104 (I)	P24 (I)	D1A	–	x
P25	ANI105 (I)	P25 (I)	D1A	–	x
P26	ANI106 (I)	P26 (I)	D1A	–	x
P27	ANI107 (I)	P27 (I)	D1A	–	x
P28	ANI108 (I)	P28 (I)	D1A	–	x
P29	ANI109 (I)	P29 (I)	D1A	–	x
P210	ANI110 (I)	P210 (I)	D1A	–	x
P211	ANI111 (I)	P211 (I)	D1A	–	x
P212	ANI112 (I)	P212 (I)	D1A	–	x
P213	ANI113 (I)	P213 (I)	D1A	–	x
P214	ANI114 (I)	P214 (I)	D1A	–	x
P215	ANI115 (I)	P215 (I)	D1A	–	x

a) A: analog noise filter; B: analog and digital noise filter; –: no noise filter

b) S1: Schmitt trigger (30/70%); S2: Schmitt trigger (40/80%); x: CMOS

Table 2-22 Port group 2: configuration registers

Register	Address	Initial value	Used bits							
PMC2L	FFFF $F444_H$	00_H	PMC27	PMC26	PMC25	PMC24	PMC23	PMC22	PMC21	PMC20
PMC2H	FFFF $F445_H$	00_H	PMC215	PMC214	PMC213	PMC212	PMC211	PMC210	PMC29	PMC28
PM2L	FFFF $F424_H$	FF_H	PM27	PM26	PM25	PM24	PM23	PM22	PM21	PM20
PM2H	FFFF $F425_H$	FF_H	PM215	PM214	PM213	PM212	PM211	PM210	PM29	PM28
P2L	FFFF $F404_H$	undefined	P27	P26	P25	P24	P23	P22	P21	P20
P2H	FFFF $F405_H$	undefined	P215	P214	P213	P212	P211	P210	P29	P28

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

2.5.6 Port group 3

Port group 3 is a 10-bit port group. In alternative mode, it comprises pins for the following functions:

- External interrupt (INTP7 and INTP8)
- Timer TAA0 channels (TIAA00, TIAA01 and TOAA00, TOAA01)
- Timer TAA1 channels (TIAA10, TIAA11 and TOAA10, TOAA11)
- CAN0 transmit/receive data (CTXD0, CRXD0)
- CAN1 transmit/receive data (CTXD1, CRXD1)
- UARTD0 transmit/receive data (TXDD0, RXDD0)
- UARTD0 baud rate clock input (ASCKD0)
- UARTD2 transmit/receive data (TXDD2, RXDD2)

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	Port type	Noise filter[a]	Input charact.[b]
Port mode (PMCnm = 0)	Alternative mode (PMCnm = 1)							
	PFCE = 0		PFCE = 1					
	Function 1 PFC = 0	Function 2 PFC = 1	Function 3 PFC = 0	Function 4 PFC = 1				
P30	TXDD0 (O)	–	–	–	P30 (I)	D0-U	–	S1
P31	RXDD0 (I) INTP7 (I)	–	–	–	P31 (I)	D3-UI	A	S1
P32	ASCKD0 (I)	TOAA01 (O)	TIAA00 (I)	TOAA00 (O)	P32 (I)	F1010-U	A	S2
P33	TIAA01 (I)	TOAA01 (O)	CTXD0 (O)	prohibited	P33 (I)	F100x-U	A	S2
P34	TIAA10 (I)	TOAA10 (O)	CRXD0 (I)	prohibited	P34 (I)	F101x-U	A	S1
P35	TIAA11 (I)	TOAA11 (O)	–	–	P35 (I)	E10-U	A	S2
P36[c]	CTXD1 (O)	–	–	–	P36 (I)	D0-U	–	S1
P37[c]	CRXD1 (I)	–	–	–	P37 (I)	D1-U	–	S1
P38[d]	TXDD2 (O)[c]	–	–	–	P38 (I)	C-U[e] D0-U	–	S1
P39[d]	RXDD2 (I)[c]/ INTP8 (I)[c]	–	–	–	P39 (I)	C-U[e] D3-UI	A	S1

[a] A: analog noise filter only for INTPn, TIAAnm inputs
B: analog and digital noise filter
–: no noise filter

[b] S1: Schmitt trigger (30/70%); S2: Schmitt trigger (40/80%); x: CMOS

[c] not available for V850ES/FE3, V850ES/FF3

[d] not available for V850ES/FE3

[e] for V850ES/FF3

Note Alternative functions CRXD0, CTXD0, and TOAA01 are provided on two pins each. Thus you can select on which pin the alternative function should appear. Refer to *“Pin function configuration” on page 39*.

Table 2-24 Port group 3: configuration registers

Register	Address	Initial value	Used bits							
V850ES/FE3										
PMC3L	FFFF F446$_{H}$	00$_{H}$	X	X	PMC35	PMC34	PMC33	PMC32	PMC31	PMC30
PM3L	FFFF F426$_{H}$	FF$_{H}$	X	X	PM35	PM34	PM33	PM32	PM31	PM30
PFC3L	FFFF F466$_{H}$	00$_{H}$	X	X	PFC35	PFC34	PFC33	PFC32	X	X
PFCE3L	FFFF F706$_{H}$	00$_{H}$	X	X	X	PFEC34	PFCE33	PFCE32	X	X
P3L	FFFF F406$_{H}$	undefined	X	X	P35	P34	P33	P32	P31	P30
PU3L	FFFF FC46$_{H}$	00$_{H}$	X	X	PU35	PU34	PU33	PU32	PU31	PU30
V850ES/FF3										
PMC3L	FFFF F446$_{H}$	00$_{H}$	X	X	PMC35	PMC34	PMC33	PMC32	PMC31	PMC30
PM3L	FFFF F426$_{H}$	FF$_{H}$	X	X	PM35	PM34	PM33	PM32	PM31	PM30
PM3H	FFFF F427$_{H}$	FF$_{H}$	X	X	X	X	X	X	PM39	PM38
PM3 (16 bit)	FFFF F426$_{H}$	FFFF$_{H}$	PM315 to PM38 (PM3H)				PM37 to PM30 (PM3L)			
PFC3L	FFFF F466$_{H}$	00$_{H}$	X	X	PFC35	PFC34	PFC33	PFC32	X	X
PFCE3L	FFFF F706$_{H}$	00$_{H}$	X	X	X	PFEC34	PFCE33	PFCE32	X	X
P3L	FFFF F406$_{H}$	undefined	X	X	P35	P34	P33	P32	P31	P30
P3H	FFFF F407$_{H}$	undefined	X	X	X	X	X	X	P39	P38
P3 (16 bit)	FFFF F406$_{H}$	undefined	P315 to P38 (P3H)				P37 to P30 (P3L)			
PU3L	FFFF FC46$_{H}$	00$_{H}$	X	X	PU35	PU34	PU33	PU32	PU31	PU30
PU3H	FFFF FC47$_{H}$	00$_{H}$	X	X	X	X	X	X	PU39	PU38
PU3 (16 bit)	FFFF FC46$_{H}$	0000$_{H}$	PU315 to PU38 (PU3H)				PU37 to PU30 (PU3L)			
V850ES/FG3, V850ES/FJ3, V850ES/FK3										
PMC3L	FFFF F446$_{H}$	00$_{H}$	PMC37	PMC36	PMC35	PMC34	PMC33	PMC32	PMC31	PMC30
PMC3H	FFFF F447$_{H}$	00$_{H}$	X	X	X	X	X	X	PMC39	PMC38
PMC3 (16 bit)	FFFF F446$_{H}$	0000$_{H}$	PMC315 to PMC38 (PMC3H)				PMC37 to PMC30 (PMC3L)			
PM3L	FFFF F426$_{H}$	FF$_{H}$	PM37	PM36	PM35	PM34	PM33	PM32	PM31	PM30
PM3H	FFFF F427$_{H}$	FF$_{H}$	X	X	X	X	X	X	PM39	PM38
PM3 (16 bit)	FFFF F426$_{H}$	FFFF$_{H}$	PM315 to PM38 (PM3H)				PM37 to PM30 (PM3L)			
PFC3L	FFFF F466$_{H}$	00$_{H}$	X	X	PFC35	PFC34	PFC33	PFC32	X	X
PFCE3L	FFFF F706$_{H}$	00$_{H}$	X	X	X	PFEC34	PFCE33	PFCE32	X	X
P3L	FFFF F406$_{H}$	undefined	P37	P36	P35	P34	P33	P32	P31	P30
P3H	FFFF F407$_{H}$	undefined	X	X	X	X	X	X	P39	P38
P3 (16 bit)	FFFF F406$_{H}$	undefined	P315 to P38 (P3H)				P37 to P30 (P3L)			
PU3L	FFFF FC46$_{H}$	00$_{H}$	PU37	PU36	PU35	PU34	PU33	PU32	PU31	PU30
PU3H	FFFF FC47$_{H}$	00$_{H}$	X	X	X	X	X	X	PU39	PU38
PU3 (16 bit)	FFFF FC46$_{H}$	0000$_{H}$	PU315 to PU38 (PU3H)				PU37 to PU30 (PU3L)			

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

All 16-bit registers can be accessed in 16-bit units.

2.5.7 Port group 4

Port group 4 is a 3-bit port group. In alternative mode, it comprises pins for the following functions:

- External interrupt (INTP14)
- Key interrupt input (KR0 to KR2)
- Clocked Serial Interface CSIB0 data/clock line (SIB0, SOB0, SCKB0)
- UARTD3 transmit/receive data (TXDD3, RXDD3)

Port group 4 includes the following pins:

Table 2-25 Port group 4: pin functions and buffer

Pin functions in different modes					Pin function after reset	Port type	Noise filter[a]	Input charact.[b]
Port mode (PMCnm = 0)	Alternative mode (PMCnm = 1)							
	PFCE = 0		PFCE = 1					
	Function 1 PFC = 0	Function 2 PFC = 1	Function 3 PFC = 0	Function 4 PFC = 1				
P40	SIB0 (I)	KR0 (I)	RXDD3 (I)[c] INTP14 (I)[c]	prohibited	P40 (I)	E11-U F113x-UI[c]	A	S1
P41	SOB0 (O)	KR1 (I)	TXDD3 (O)[c]	prohibited	P41 (I)	E01-U F010x-U[c]	A	S2
P42	SCKB0 (I/O)	KR2 (I)	–	–	P42 (I)	E21-U	A	S2

[a] A: analog noise filter only for KRn, INTPn inputs
B: analog and digital noise filter
–: no noise filter

[b] S1: Schmitt trigger (30/70%); S2: Schmitt trigger (40/80%); x: CMOS

[c] not available for V850ES/FE3, V850ES/FF3, µPD70F3374 and µPD70F3375 of V850ES/FG3, µPD70F3378 of V850ES/FJ3

Note Alternative functions RXDD3, TXDD3, INTP14, and KR0 to KR2 are provided on two pins each. Thus you can select on which pin the alternative function should appear. Refer to *“Pin function configuration” on page 39*.

Table 2-26 Port group 4: configuration registers

Register	Address	Initial value	Used bits							
PMC4	FFFF F448$_H$	00$_H$	X	X	X	X	X	PMC42	PMC41	PMC40
PM4	FFFF F428$_H$	FF$_H$	X	X	X	X	X	PM42	PM41	PM40
PFC4	FFFF F468$_H$	00$_H$	X	X	X	X	X	PFC42	PFC41	PFC40
PFCE4[a]	FFFF F708$_H$	00$_H$	X	X	X	X	X	X	PFCE41	PFCE40
P4	FFFF F408$_H$	undefined	X	X	X	X	X	P42	P41	P40
PU4	FFFF FC48$_H$	00$_H$	X	X	X	X	X	PU42	PU41	PU40

[a] not available with V850ES/FE3, V850ES/FF3, µPD70F3374 and µPD70F3375 of V850ES/FG3, µPD70F3378 of V850ES/FJ3

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

2.5.8 Port group 5

Port group 5 is an 8-bit port group. In alternative mode, it comprises pins for the following functions:

- Key interrupt input 0 to 5 (KR0 to KR5)
- N-Wire debug interface signals (DDI, DDO, DCK, DMS)
- Timer TAB0 channels (TIAB00 to TIAB03, TOAB00 to TOAB03)
- Motor control channels (TOAB0B1 to TOAB0B3, TOAB0T1 to TOAB0T3)

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	Port type	Noise filter[a]	Input charact.[b]
Port mode (PMC = 0)	Alternative mode (PMCnm = 1)				On-chip debug mode (OCDM0 = 1)				
	PFCE = 0		PFCE = 1						
	Function 1 PFC = 0	Function 2 PFC = 1	Function 3 PFC = 0	Function 4 PFC = 1					
P50	KR0 (I)	TIAB01 (I)	TOAB01 (O)	TOAB0T1 (O)	–	P50 (I)	F1100-U	A	S2
P51	KR1 (I)	TIAB02 (I)	TOAB02 (O)	TOAB0B1 (O)	–	P51 (I)	F1100-U	A	S2
P52	KR2 (I)	TIAB03 (I)	TOAB03 (O)	TOAB0T2 (O)	DDI (I)	P52 (I) or DDI (I)[c]	F1100O1-U	A	S2
P53	KR3 (I)	TIAB00 (I)	TOAB00 (O)	TOAB0B2 (O)	DDO (O)	P53 (I) or DDO (O)[c]	F1100O0-U	A	S2
P54	KR4 (I)	prohibited	prohibited	TOAB0T3 (O)	DCK (I)	P54 (I) or DCK (I)[c]	F1xx0O1-U	A	S2
P55	KR5 (I)	prohibited	prohibited	TOAB0B3 (O)	DMS (I)	P55 (I) or DMS (I)[c]	F1xx0O1-U	A	S2

a) A: analog noise filter only for KRn, TIABnm inputs
B: analog and digital noise filter
–: no noise filter

b) S1: Schmitt trigger (30/70%); S2: Schmitt trigger (40/80%); x: CMOS

c) The pin function after reset depends on the reset source, that means on bit OCDM.OCDM0. Refer to *"OCDM - On-chip debug mode register" on page 44* and to *"On-Chip Debug Unit" on page 930*.

Note Alternative functions TIAB00, TIAB03, TOAB00, TOAB03 and KR0 to KR2 are provided on two pins each. Thus you can select on which pin the alternative function should appear. Refer to *"Pin function configuration" on page 39*.

Table 2-28 Port group 5: configuration registers

Register	Address	Initial value	Used bits							
PMC5	FFFF F44A$_H$	00$_H$	X	X	PMC55	PMC54	PMC53	PMC52	PMC51	PMC50
PM5	FFFF F42A$_H$	FF$_H$	X	X	PM55	PM54	PM53	PM52	PM51	PM50
PFC5	FFFF F46A$_H$	00$_H$	X	X	PFC55	PFC54	PFC53	PFC52	PFC51	PFC50
PFCE5	FFFF F70A$_H$	00$_H$	X	X	PFCE55	PFCE54	PFCE53	PFCE52	PFCE51	PFCE50
OCDM	FFFF F9FC$_H$	00$_H$ / 01$_H$[a]	0	0	0	0	0	0	0	OCDM0
P5	FFFF F40A$_H$	undefined	X	X	P55	P54	P53	P52	P51	P50
PU5	FFFF FC4A$_H$	00$_H$	X	X	PU55	PU54	PU53	PU52	PU51	PU50

[a] Depends on the reset source (Refer to *“OCDM - On-chip debug mode register” on page 44* and to *“On-Chip Debug Unit” on page 930*)

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

2.5.9 Port group 6 (V850ES/FJ3, V850ES/FK3)

Note Port group 6 is available only for V850ES/FJ3, V850ES/FK3.

Port group 6 is a 16-bit port group. In alternative mode, it comprises pins for the following functions:

- External interrupt (INTP11 to INTP13, INTP15)
- Timer TAB2 channels (TIAB20 to TIAB23 and TOAB20 to TOAB23)
- Clocked Serial Interface CSIB3 data/clock line (SIB3, SOB3, SCKB3)
- CAN2 transmit/receive data (CTXD2, CRXD2)
- CAN3 transmit/receive data (CTXD3, CRXD3)
- UARTD6 transmit/receive data (TXDD6, RXDD6)
- UARTD7 transmit/receive data (TXDD7, RXDD7)
- A/D Converter 1 external trigger input (ADTRG1)

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 type	Noise filter[a]	Input charact.[b]
Port mode (PMC = 0)	Alternative mode (PMCnm = 1)							
	PFCE = 0		PFCE = 1					
	Function 1 PFC = 0	Function 2 PFC = 1	Function 3 PFC = 0	Function 4 PFC = 1				
P60	prohibited	INTP11 (I)	–	–	P60 (I)	refer to *Table 2-30* *on* *page 126*	A	S2
P61	prohibited	INTP12 (I)	–	–	P61 (I)		A	S2
P62	TXDD6 (O)[c]	INTP13 (I)	SOB3 (O)[d]	prohibited	P62 (I)		A	S2
P63	RXDD6 (I)[c] INTP13 (I)[c]	prohibited	SIB3 (I)[d]	prohibited	P63 (I)		–	S1
P64	prohibited	prohibited	SCKB3 (I/O)[d]	prohibited	P64 (I)		–	S1
P65	prohibited	CTXD2 (O)	–	–	P65 (I)		–	S1
P66	prohibited	CRXD2 (I)	–	–	P66 (I)		–	S1
P67	prohibited	CTXD3 (O)[e]	–	–	P67 (I)		–	S1
P68	prohibited	CRXD3 (I)[e]	–	–	P68 (I)		–	S1
P69	ADTRG1 (I)[c]	–	–	–	P69 (I)		–/A[f]	S1
P610	TIAB20 (I)	TOAB20 (O)	–	–	P610 (I)		A	S2
P611	TIAB21 (I)	TOAB21 (O)	–	–	P611 (I)		A	S2
P612	TIAB22 (I)	TOAB22 (O)	–	–	P612 (I)		A	S2
P613	TIAB23 (I)	TOAB23 (O)	–	–	P613 (I)		A	S2
P614	TXDD7 (O)[c]	–	–	–	P614 (I)		–	S1
P615	RXDD7 (I)[c] INTP15[c]	–	–	–	P615 (I)		–	S1

[a] A: analog noise filter only for INTPn, TIABnm inputs
B: analog and digital noise filter
–: no noise filter

[b] S1: Schmitt trigger (30/70%); S2: Schmitt trigger (40/80%); x: CMOS

[c] only available for V850ES/FK3

[d] not available for µPD70F3378, µPD70F3379, and µPD70F3380 of V850ES/FJ3

[e] not available for µPD70F3378 of V850ES/FJ3

[f] ADTRG1 input for V850ES/FK3 features an analog noise filter

Table 2-30 **Port group 6: port types**

Port	V850ES/FJ3			V850ES/FK3
	μPD70F3378	μPD70F3379, μPD70F3380	μPD70F3381, μPD70F3382	all devices
P60	EX1-UI	EX1-UI	EX1-UI	EX1-UI
P61	EX1-UI	EX1-UI	EX1-UI	EX1-UI
P62	EX1-UI	EX1-UI	Fx10x-UI	F010x-UI
P63	C-U	C-U	Fxx1x-U	F3x1x-UI
P64	C-U	C-U	Fxx2x-U	Fxx2x-U
P65	Ex0-U	Ex0-U	Ex0-U	Ex0-U
P66	Ex1-U	Ex1-U	Ex1-U	Ex1-U
P67	C-U	Ex0-U	Ex0-U	Ex0-U
P68	C-U	Ex1-U	Ex1-U	Ex1-U
P69	C-U	C-U	C-U	D1-U
P610	E10-U	E10-U	E10-U	E10-U
P611	E10-U	E10-U	E10-U	E10-U
P612	E10-U	E10-U	E10-U	E10-U
P613	E10-U	E10-U	E10-U	E10-U
P614	C-U	C-U	C-U	D0-U
P615	C-U	C-U	C-U	D3-UI

Note Alternative functions CRXD2 and CTXD2 are provided on two pins each. Thus you can select on which pin the alternative function should appear. Refer to *“Pin function configuration” on page 39*.

Table 2-31 **Port group 6: configuration registers**

Register	Address	Initial value	Used bits							
PMC6L	FFFF F44C$_H$	00$_H$	PMC67[a]	PMC66	PMC65	PMC64[b]	PMC63[b]	PMC62	PMC61	PMC60
PMC6H	FFFF F44D$_H$	00$_H$	PMC615[c]	PMC614[c]	PMC613	PMC612	PMC611	PMC610	PMC69[c]	PMC68[a]
PMC6 (16 bit)	FFFF F44C$_H$	0000$_H$	PMC615 to PMC68 (PMC6H)				PMC67 to PMC60 (PMC6L)			
PM6L	FFFF F42C$_H$	FF$_H$	PM67	PM66	PM65	PM64	PM63	PM62	PM61	PM60
PM6H	FFFF F42D$_H$	FF$_H$	PM615	PM614	PM613	PM612	PM611	PM610	PM69	PM68
PM6 (16 bit)	FFFF F42C$_H$	FFFF$_H$	PM615 to PM68 (PM6H)				PM67 to PM60 (PM6L)			
PFC6L	FFFF F46C$_H$	00$_H$	PFC67[a]	PFC66[a]	PFC65	PFC64	PFC63[b]	PFC62	PFC61	PFC60
PFC6H	FFFF F46D$_H$	00$_H$	X	X	PFC613	PFC612	PFC611	PFC610	X	PFC68[a]
PFC6 (16 bit)	FFFF F46C$_H$	0000$_H$	PFC613 to PFC68 (PFC6H)				PFC67 to PFC60 (PFC6L)			
PFCE6L[b]	FFFF F70C$_H$	00$_H$	X	X	X	PFCE64	PFCE63	PFCE62	X	X
P6L	FFFF F40C$_H$	undefined	P67	P66	P65	P64	P63	P62	P61	P60
P6H	FFFF F40D$_H$	undefined	P615	P614	P613	P612	P611	P610	P69	P68
P6 (16 bit)	FFFF F40C$_H$	undefined	P615 to P68 (P6H)				P67 to P60 (P6L)			
PU6L	FFFF FC4C$_H$	00$_H$	PU67	PU66	PU65	PU64	PU63	PU62	PU61	PU60
PU6H	FFFF FC4D$_H$	00$_H$	PU615	PU614	PU613	PU612	PU611	PU610	PU69	PU68
PU6 (16 bit)	FFFF FC4C$_H$	0000$_H$	PU615 to PU68 (PU6H)				PU67 to PU60 (PU6L)			

a) not available for µPD70F3378 of V850ES/FJ3

b) not available for µPD70F3378, µPD70F3379, µPD70F3380 of V850ES/FJ3

c) only available for V850ES/FK3

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

All 16-bit registers can be accessed in 16-bit units.

2.5.10 Port group 7

Port group 7 is a 16-bit port group. It includes pins for the following functions:

- A/D Converter 0 inputs

Port group 7 includes the following pins:

Table 2-32 Port group 7: pin functions and port types

Pin functions in different modes		Pin function after reset	Port type	Noise filter[a]	Input charact.[b]
Port mode (PMCnm = 0)	Alternative mode (PMCnm = 1)				
P70	ANI0 (I)	P70 (I)	D1A	–	x
P71	ANI1 (I)	P71 (I)	D1A	–	x
P72	ANI2 (I)	P72 (I)	D1A	–	x
P73	ANI3 (I)	P73 (I)	D1A	–	x
P74	ANI4 (I)	P74 (I)	D1A	–	x
P75	ANI5 (I)	P75 (I)	D1A	–	x
P76	ANI6 (I)	P76 (I)	D1A	–	x
P77	ANI7 (I)	P77 (I)	D1A	–	x
P78	ANI8 (I)	P78 (I)	D1A	–	x
P79	ANI9 (I)	P79 (I)	D1A	–	x
P710[c]	ANI10 (I)	P710 (I)	D1A	–	x
P711[c]	ANI11 (I)	P711 (I)	D1A	–	x
P712[d]	ANI12 (I)	P712 (I)	D1A	–	x
P713[d]	ANI13 (I)	P713 (I)	D1A	–	x
P714[d]	ANI14 (I)	P714 (I)	D1A	–	x
P715[d]	ANI15 (I)	P715 (I)	D1A	–	x

[a] A: analog noise filter; B: analog and digital noise filter; –: no noise filter
[b] S1: Schmitt trigger (30/70%); S2: Schmitt trigger (40/80%); x: CMOS
[c] not available for V850ES/FE3
[d] not available for V850ES/FE3 and V850ES/FF3

Table 2-33 Port group 7: configuration registers

Register	Address	Initial value	Used bits							
PMC7L	FFFF F44E$_H$	00$_H$	PMC77	PMC76	PMC75	PMC74	PMC73	PMC72	PMC71	PMC70
PMC7H	FFFF F44F$_H$	00$_H$	PMC715[a]	PMC714[a]	PMC713[a]	PMC712[a]	PMC711[b]	PMC710[b]	PMC79	PMC78
PM7L	FFFF F42E$_H$	FF$_H$	PM77	PM76	PM75	PM74	PM73	PM72	PM71	PM70
PM7H	FFFF F42F$_H$	FF$_H$	PM715[a]	PM714[a]	PM713[a]	PM712[a]	PM711[b]	PM710[b]	PM79	PM78
P7L	FFFF F40E$_H$	undefined	P77	P76	P75	P74	P73	P72	P71	P70
P7H	FFFF F40F$_H$	undefined	P715[a]	P714[a]	P713[a]	P712[a]	P711[b]	P710[b]	P79	P78

a) not available for V850ES/FE3 and V850ES/FF3

b) not available for V850ES/FE3

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

2.5.11 Port group 8 (V850ES/FJ3, V850ES/FK3)

Note Port group 8 is available only for V850ES/FJ3, V850ES/FK3.

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

- External interrupt (INTP14)
- UARTD3 transmit/receive data (TXDD3, RXDD3)

Port group 8 includes the following pins:

Table 2-34 Port group 8: pin functions and port types

Pin functions in different modes		Pin function after reset	Port type	Noise filter[a]	Input charact.[b]
Port mode (PMCnm = 0)	Alternative mode (PMCnm = 1)				
P80	RXDD3 (I)[c]/INTP14 (I)	P80 (I)	D1-UI D3-UI[c]	A	S1
P81	TXDD3 (O)[c]	P81 (I)	C-U D0-U[c]	–	S1

[a] A: analog noise filter only for INTPn inputs
B: analog and digital noise filter
–: no noise filter

[b] S1: Schmitt trigger (30/70%); S2: Schmitt trigger (40/80%); x: CMOS

[c] not available for µPD70F3378 of V850ES/FJ3

Note Alternative functions INTP14, RXDD3, and TXDD3 are provided on two pins each. Thus you can select on which pin the alternative function should appear. Refer to *“Pin function configuration” on page 39*.

Table 2-35 Port group 8: configuration registers

Register	Address	Initial value	Used bits							
PMC8	FFFF F450$_H$	00$_H$	X	X	X	X	X	X	PMC81[a]	PMC80
PM8	FFFF F430$_H$	FF$_H$	X	X	X	X	X	X	PM81	PM80
P8	FFFF F410$_H$	undefined	X	X	X	X	X	X	P81	P80
PU8	FFFF FC50$_H$	00$_H$	X	X	X	X	X	X	PU81	PU80

[a] not available for µPD70F3378 of V850ES/FJ3

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

2.5.12 Port group 9

Port group 9 is an 16-bit port group. In alternative mode, it comprises pins for the following functions:

- External interrupt (INTP4 to INTP6)
- Key interrupt input 6 to 7 (KR6 to KR7)
- Timer TAA2 channels (TIAA20, TIAA21 and TOAA20, TOAA21)
- Timer TAB0 channels (TIAB00, TIAB03 and TOAB00, TOAB03)
- Timer TAB1 channels (TIAB10 to TIAB13 and TOAB10 to TOAB13)
- Clocked Serial Interface CSIB1 data/clock line (SOB1, SIB1, SCKB1)
- Clocked Serial Interface CSIB2 data/clock line (SOB2, SIB2, SCKB2)
- UARTD1 transmit/receive data (TXDD1, RXDD1)
- UARTD4 transmit/receive data (TXDD4, RXDD4)
- UARTD5 transmit/receive data (TXDD5, RXDD5)
- CAN2 transmit/receive data (CTXD2, CRXD2)
- I^{2}C data/clock line (SDA00, SCL00)
- Programmable clock output (PCL)

Note If P914 and P915 are in output port mode (PMC9.PMC9m = 0 and PM9.PM9m = 0), the PF9H register specifies normal output or open-drain output.

Table 2-36 Port group 9: pin functions and port types

Pin functions in different modes					Pin function after reset	Port type	Noise filter[a]	Input charact.[b]
Port mode (PMCnm = 0)	Alternative mode (PMCnm = 1)							
	PFCE = 0		PFCE = 1					
	Function 1 PFC = 0	Function 2 PFC = 1	Function 3 PFC = 0	Function 4 PFC = 1				
P90	prohibited	KR6 (I)	TXDD1 (O)	prohibited	P90 (I)	*Table 2-37 on page 133*	A	S2
P91	prohibited	KR7 (I)	RXDD1 (I) KR7 (I)	prohibited	P91 (I)		A	S1
P92[c]	prohibited	TIAB11 (I)	TOAB11 (O)	prohibited	P92 (I)		A	S2
P93[c]	prohibited	TIAB12 (I)	TOAB12 (O)	prohibited	P93 (I)		A	S2
P94[c]	prohibited	TIAB13 (I)	TOAB13 (O)	prohibited	P94 (I)		A	S2
P95[c]	prohibited	TIAB10 (I)	TOAB10 (O)	prohibited	P95 (I)		A	S2
P96	prohibited	prohibited	TIAA21 (I)	TOAA21 (O)	P96 (I)		A	S2
P97	prohibited	SIB1 (I)	TIAA20 (I)	TOAA20 (O)	P97 (I)		A	S2
P98	prohibited	SOB1 (O)	TIAB03 (I)	TOAB03 (O)	P98 (I)		A	S2
P99	prohibited	SCKB1 (I/O)	TIAB00 (I)	TOAB00 (O)	P99 (I)		A	S2
P910[c]	prohibited	SIB2 (I)[d]	CTXD2 (O)[d]	prohibited	P910 (I)		–	S2
P911[c]	prohibited	SOB2 (O)[d]	CRXD2 (I)[d]	prohibited	P911 (I)		–	S1
P912[c]	prohibited	SCKB2 (I/O)[d]	prohibited	TXDD5 (O)[e]	P912 (I)		–	S2
P913	prohibited	INTP4	PCL (O)	RXDD5 (I)[e] INTP4 (I)[e]	P913 (I)		A	S1
P914	prohibited	INTP5	SDA00 (I/O)	RXDD4 (I)[f] INTP5 (I)[e]	P914 (I)		A	S1
P915	prohibited	INTP6	SCL00 (I/O)	TXDD4[f]	P915 (I)		A	S1

[a] A: analog noise filter only for KRn, TIABnm inputs
B: analog and digital noise filter
–: no noise filter

[b] S1: Schmitt trigger (30/70%); S2: Schmitt trigger (40/80%); x: CMOS

[c] not available for V850ES/FE3 and V850ES/FF3

[d] not available for V850ES/FE3, V850ES/FF3 and V850ES/FG3

[e] not available for V850ES/FE3, V850ES/FF3, V850ES/FG3 and μPD70F3378 of V850ES/FJ3

[f] not available for V850ES/FE3, V850ES/FF3, μPD70F3374, μPD70F3375 of V850ES/FG3 and μPD70F3378, μPD70F3379 of V850ES/FJ3

Note Alternative functions TIAB00, TIAB03, TOAB00, TOAB03, CRXD2, and CTXD2 are provided on two pins each. Thus you can select on which pin the alternative function should appear. Refer to *"Pin function configuration" on page 39*.

Table 2-37 Port group 9: port types

Port	V850ES/FE3	V850ES/FF3	V850ES/FG3		V850ES/FJ3		V850ES/FK3
	all devices	all devices	µPD70F3374 µPD70F3375	µPD70F3376A µPD70F3377A	µPD70F3378	µPD70F3379 µPD70F3380 µPD70F3381 µPD70F3382	all devices
P90	Fx10x-U	Fx10x-U	Fx10x-U	Fx10x-U	Fx10x-U	Fx10x-U	Fx10x-U
P91	Fx13x-U	Fx13x-U	Fx13x-U	Fx13x-U	Fx13x-U	Fx13x-U	Fx13x-U
P92	–	–	Fx10x-U	Fx10x-U	Fx10x-U	Fx10x-U	Fx10x-U
P93	–	–	Fx10x-U	Fx10x-U	Fx10x-U	Fx10x-U	Fx10x-U
P94	–	–	Fx10x-U	Fx10x-U	Fx10x-U	Fx10x-U	Fx10x-U
P95	–	–	Fx10x-U	Fx10x-U	Fx10x-U	Fx10x-U	Fx10x-U
P96	Fxx10-U	Fxx10-U	Fxx10-U	Fxx10-U	Fxx10-U	Fxx10-U	Fxx10-U
P97	Fx110-U	Fx110-U	Fx110-U	Fx110-U	Fx110-U	Fx110-U	Fx110-U
P98	Fx010-U	Fx010-U	Fx010-U	Fx010-U	Fx010-U	Fx010-U	Fx010-U
P99	Fx210-U	Fx210-U	Fx210-U	Fx210-U	Fx210-U	Fx210-U	Fx210-U
P910	–	–	C-U	C-U	Fx10x-U	Fx10x-U	Fx10x-U
P911	–	–	C-U	C-U	Fx01x-U	Fx01x-U	Fx01x-U
P912	–	–	C-U	C-U	Ex2-U	Fx2x0-U	Fx2x0-U
P913	Fx10x-UI	Fx10x-UI	Fx10x-UI	Fx10x-UI	Fx10x-UI	Fx103-UI	Fx103-UI
P914	Fx12x-UFI	Fx12x-UFI	Fx12x-UFI	Fx123-UFI	Fx12x-UFI	Fx123-UFI	Fx123-UFI
P915	Fx12x-UFI	Fx12x-UFI	Fx12x-UFI	Fx120-UFI	Fx12x-UFI	Fx120-UFI	Fx120-UFI

Table 2-38 Port group 9: V850ES/FE3, V850ES/FF3 configuration registers

Register	Address	Initial value	Used bits							
PMC9L	FFFF F452$_H$	00$_H$	PMC97	PMC96	X	X	X	X	PMC91	PMC90
PMC9H	FFFF F453$_H$	00$_H$	PMC915	PMC914	PMC913	X	X	X	PMC99	PMC98
PMC9 (16 bit)	FFFF F452$_H$	0000$_H$	PMC915 to PMC98 (PMC9H)				PMC97 to PMC90 (PMC9L)			
PM9L	FFFF F432$_H$	FF$_H$	PM97	PM96	X	X	X	X	PM91	PM90
PM9H	FFFF F433$_H$	FF$_H$	PM915	PM914	PM913	X	X	X	PM99	PM98
PM9 (16 bit)	FFFF F432$_H$	FFFF$_H$	PM915 to PM98 (PM9H)				PM97 to PM90 (PM9L)			
PFC9L	FFFF F472$_H$	00$_H$	PFC97	PFC96	X	X	X	X	PFC91	PFC90
PFC9H	FFFF F473$_H$	00$_H$	PFC915	PFC914	PFC913	X	X	X	PFC99	PFC98
PFC9 (16 bit)	FFFF F472$_H$	0000$_H$	PFC915 to PFC98 (PFC9H)				PFC97 to PFC90 (PFC9L)			
PFCE9L	FFFF F712$_H$	00$_H$	PFCE97	PFCE96	X	X	X	X	PFCE91	PFCE90
PFCE9H	FFFF F713$_H$	00$_H$	PFCE915	PFCE914	PFCE913	X	X	X	PFCE99	PFCE98
PFCE9 (16 bit)	FFFF F712$_H$	0000$_H$	PFCE915 to PFCE98 (PFCE9H)				PFCE97 to PFCE90 (PFCE9L)			
P9L	FFFF F412$_H$	undefined	P97	P96	X	X	X	X	P91	P90
P9H	FFFF F413$_H$	undefined	P915	P914	P913	X	X	X	P99	P98
P9 (16 bit)	FFFF F412$_H$	undefined	P915 to P98 (P9H)				P97 to P90 (P9L)			
PU9L	FFFF FC52$_H$	00$_H$	PU97	PU96	X	X	X	X	PU91	PU90
PU9H	FFFF FC53$_H$	00$_H$	PU915	PU914	PU913	X	X	X	PU99	PU98
PU9 (16 bit)	FFFF FC52$_H$	0000$_H$	PU915 to PU98 (PU9H)				PU97 to PU90 (PU9L)			
PF9H	FFFF FC73$_H$	00$_H$	PF915	PF914	X	X	X	X	X	X

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

All 16-bit registers can be accessed in 16-bit units.

Table 2-39 Port group 9: V850ES/FG3, V850ES/FJ3, V850ES/FK3 configuration registers

Register	Address	Initial value	Used bits							
PMC9L	FFFF F452$_H$	00$_H$	PMC97	PMC96	PMC95	PMC94	PMC93	PMC92	PMC91	PMC90
PMC9H	FFFF F453$_H$	00$_H$	PMC915	PMC914	PMC913	PMC912[a]	PMC911[a]	PMC910[a]	PMC99	PMC98
PMC9 (16 bit)	FFFF F452$_H$	0000$_H$	PMC915 to PMC98 (PMC9H)				PMC97 to PMC90 (PMC9L)			
PM9L	FFFF F432$_H$	FF$_H$	PM97	PM96	PM95	PM94	PM93	PM92	PM91	PM90
PM9H	FFFF F433$_H$	FF$_H$	PM915	PM914	PM913	PM912	PM911	PM910	PM99	PM98
PM9 (16 bit)	FFFF F432$_H$	FFFF$_H$	PM915 to PM98 (PM9H)				PM97 to PM90 (PM9L)			
PFC9L	FFFF F472$_H$	00$_H$	PFC97	PFC96	PFC95	PFC94	PFC93	PFC92	PFC91	PFC90
PFC9H	FFFF F473$_H$	00$_H$	PFC915	PFC914	PFC913	PFC912[a]	PFC911[a]	PFC910[a]	PFC99	PFC98
PFC9 (16 bit)	FFFF F472$_H$	0000$_H$	PFC915 to PFC98 (PFC9H)				PFC97 to PFC90 (PFC9L)			
PFCE9L	FFFF F712$_H$	00$_H$	PFCE97	PFCE96	PFCE95	PFCE94	PFCE93	PFCE92	PFCE91	PFCE90
PFCE9H	FFFF F713$_H$	00$_H$	PFCE915	PFCE914	PFCE913	PFCE912[b]	PFCE911[a]	PFCE910[a]	PFCE99	PFCE98
PFCE9 (16 bit)	FFFF F712$_H$	0000$_H$	PFCE915 to PFCE98 (PFCE9H)				PFCE97 to PFCE90 (PFCE9L)			
P9L	FFFF F412$_H$	undefined	P97	P96	P95	P94	P93	P92	P91	P90
P9H	FFFF F413$_H$	undefined	P915	P914	P913	P912	P911	P910	P99	P98
P9 (16 bit)	FFFF F412$_H$	undefined	P915 to P98 (P9H)				P97 to P90 (P9L)			
PU9L	FFFF FC52$_H$	00$_H$	PU97	PU96	PU95	PU94	PU93	PU92	PU91	PU90
PU9H	FFFF FC53$_H$	00$_H$	PU915	PU914	PU913	PU912	PU911	PU910	PU99	PU98
PU9 (16 bit)	FFFF FC52$_H$	0000$_H$	PU915 to PU98 (PU9H)				PU97 to PU90 (PU9L)			
PF9H	FFFF FC73$_H$	00$_H$	PF915	PF914	X	X	X	X	X	X

a) not available for V850ES/FG3

b) not available for V850ES/FG3 and µPD70F3378 of V850ES/FJ3

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

All 16-bit registers can be accessed in 16-bit units.

2.5.13 Port group 12 (V850ES/FJ3, V850ES/FK3)

Note Port group 8 is available only for V850ES/FJ3, V850ES/FK3.

Port group 12 is an 8-bit port group. It includes pins for the following functions:

- A/D Converter channel 0 inputs

Port group 12 includes the following pins:

Table 2-40 Port group 12: pin functions and port types

Pin functions in different modes		Pin function after reset	Port type	Noise filter[a]	Input charact.[b]
Port mode (PMCnm = 0)	Alternative mode (PMCnm = 1)				
P120	ANI16 (I)	P120 (I)	D1A	–	x
P121	ANI17 (I)	P121 (I)	D1A	–	x
P122	ANI18 (I)	P122 (I)	D1A	–	x
P123	ANI19 (I)	P123 (I)	D1A	–	x
P124	ANI20 (I)	P124 (I)	D1A	–	x
P125	ANI21 (I)	P125 (I)	D1A	–	x
P126	ANI22 (I)	P126 (I)	D1A	–	x
P127	ANI23 (I)	P127 (I)	D1A	–	x

[a] A: analog noise filter; B: analog and digital noise filter; –: no noise filter
[b] S1: Schmitt trigger (30/70%); S2: Schmitt trigger (40/80%); x: CMOS

Table 2-41 Port group 12: configuration registers

Register	Address	Initial value	Used bits							
PMC12	FFFF F458$_H$	00$_H$	PMC127	PMC126	PMC125	PMC124	PMC123	PMC122	PMC121	PMC120
PM12	FFFF F438$_H$	FF$_H$	PM127	PM126	PM125	PM124	PM123	PM122	PM121	PM120
P12	FFFF F418$_H$	undefined	P127	P126	P125	P124	P123	P122	P121	P120

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

2.5.14 Port group 15 (V850ES/FK3)

Note Port group 15 is available only for V850ES/FK3.

Port group 15 is an 8-bit port group. It includes pins for the following functions:

- Timer TAA5 channels (TIAA50, TIAA51 and TOAA50, TOAA51)
- Timer TAA6 channels (TIAA60, TIAA61 and TOAA60, TOAA61)
- Timer TAA7 channels (TIAA70, TIAA71 and TOAA70, TOAA71)
- CAN4 transmit/receive data (CTXD4, CRXD4)

Port group 15 includes the following pins:

Table 2-42 Port group 15: pin functions and port types

Pin functions in different modes			Pin function after reset	Port type	Noise filter[a]	Input charact.[b]
Port mode (PMCnm = 0)	Alternative mode (PMCnm = 1)					
	Function 1 PFC = 0	Function 2 PFC = 1				
P150	TIAA50 (I)	TOAA50 (I)	P150 (I)	E10-U	–	S2
P151	TIAA51 (I)	TOAA51 (I)	P151 (I)	E10-U	A	S2
P152	TIAA60 (I)	TOAA60 (I)	P152 (I)	E10-U	A	S2
P153	TIAA61 (I)	TOAA61 (I)	P153 (I)	E10-U	A	S2
P154	TIAA70 (I)	TOAA70 (I)	P154 (I)	E10-U	A	S2
P155	TIAA71 (I)	TOAA71 (I)	P155 (I)	E10-U	A	S2
P156	CTXD4 (O)	–	P156 (I)	D0-U	–	S1
P157	CRXD4 (I)	–	P157 (I)	D1-U	–	S1

[a] A: analog noise filter only for TIAAnm inputs
B: analog and digital noise filter
–: no noise filter

[b] S1: Schmitt trigger (30/70%); S2: Schmitt trigger (40/80%); x: CMOS

Table 2-43 Port group 15: configuration registers

Register	Address	Initial value	Used bits							
PMC15	FFFF F45E$_H$	00$_H$	PMC157	PMC156	PMC155	PMC154	PMC153	PMC152	PMC151	PMC150
PM15	FFFF F43E$_H$	FF$_H$	PM157	PM156	PM155	PM154	PM153	PM152	PM151	PM150
PFC15	FFFF F47E$_H$	00$_H$	X	X	PFC155	PFC154	PFC153	PFC152	PFC151	PFC150
P15	FFFF F41E$_H$	undefined	P157	P156	P155	P154	P153	P152	P151	P150
PU15	FFFF FC5E$_H$	00$_H$	PU157	PU156	PU155	PU154	PU153	PU152	PU151	PU150

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

2.5.15 Port group CD (V850ES/FJ3, V850ES/FK3)

Note Port group CD is available only for V850ES/FJ3 and V850ES/FK3.

Port group CD is a 4-bit port group. The pins of this port group only work in port mode.

Port group CD includes the following pins:

Table 2-44 Port group CD: pin functions and port types

Pin function in port mode	Pin function after reset	Port type	Noise filter[a]	Input charact.[b]
PCD0	PCD0 (I)	C	–	x
PCD1	PCD1 (I)	C	–	x
PCD2	PCD2 (I)	C	–	x
PCD3	PCD3 (I)	C	–	x

[a] A: analog noise filter; B: analog and digital noise filter; –: no noise filter

[b] S1: Schmitt trigger (30/70%); S2: Schmitt trigger (40/80%); x: CMOS

Table 2-45 Port group CD: configuration registers

Register	Address	Initial value	Used bits							
PMCD	FFFF F02E$_{\text{H}}$	FF$_{\text{H}}$	X	X	X	X	PMCD3	PMCD2	PMCD1	PMCD0
PCD	FFFF F00E$_{\text{H}}$	undefined	X	X	X	X	PCD3	PCD2	PCD1	PCD0

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

2.5.16 Port group CM

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

- External memory interface data wait request ($\overline{\text{WAIT}}$)
- CPU system clock output (CLKOUT)
- External memory interface bus hold request input ($\overline{\text{HLDRQ}}$)
- External memory interface bus hold acknowledge output ($\overline{\text{HLDAK}}$)

Port group CM includes the following pins:

Table 2-46 Port group CM: pin functions and port types

Pin functions in different modes		Pin function after reset	Port type	Noise filter[a]	Input charact.[b]
Port mode (PMCnm = 0)	Alternative mode (PMCnm = 1)				
PCM0	$\overline{\text{WAIT}}$ (I)[c]	PCM0 (I)	C D1[c]	–	x
PCM1	CLKOUT (O)	PCM1 (I)	D0	–	x
PCM2[d]	$\overline{\text{HLDAK}}$ (O)[c]	PCM2 (I)	C D0[c]	–	x
PCM3[d]	$\overline{\text{HLDRQ}}$ (I)[c]	PCM3 (I)	C D1[c]	–	x
PCM4[c]	–	PCM4 (I)	C	–	x
PCM5[c]	–	PCM5 (I)	C	–	x

[a] A: analog noise filter; B: analog and digital noise filter; –: no noise filter
[b] S1: Schmitt trigger (30/70%); S2: Schmitt trigger (40/80%); x: CMOS
[c] not available for V850ES/FE3, V850ES/FF3, V850ES/FG3
[d] not available for V850ES/FE3

Table 2-47 Port group CM: configuration registers

Register	Address	Initial value	Used bits							
PMCCM	FFFF F04C$_H$	00$_H$	X	X	X	X	PMCCM3[a]	PMCCM2[a]	PMCCM1	PMCCM0[a]
PMCM	FFFF F02C$_H$	FF$_H$	X	X	PMCM5[a]	PMCM4[a]	PMCM3[b]	PMCM2[b]	PMCM1	PMCM0
PCM	FFFF F00C$_H$	undefined	X	X	PCM5[a]	PCM4[a]	PCM3[b]	PCM2[b]	PCM1	PCM0

[a] not available for V850ES/FE3, V850ES/FF3, V850ES/FG3
[b] not available for V850ES/FE3

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

2.5.17 Port group CS (V850ES/FF3, V850ES/FG3, V850ES/FJ3, V850ES/FK3)

Note Port group CS is available only for V850ES/FF3, V850ES/FG3, V850ES/FJ3 and V850ES/FK3.

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

- External memory interface chip select signals ($\overline{\text{CS0}}$ to $\overline{\text{CS3}}$)

Port group CS includes the following pins:

Table 2-48 Port group CS: pin functions and port types

Pin functions in different modes		Pin function after reset	Port type	Noise filter[a]	Input charact.[b]
Port mode (PMCnm = 0)	Alternative mode (PMCnm = 1)				
PCS0	$\overline{\text{CS0}}$ (O)[c]	PCS0 (I)	C D0[c]	–	x
PCS1	$\overline{\text{CS1}}$ (O)[c]	PCS1 (I)	C D0[c]	–	x
PCS2[c]	$\overline{\text{CS2}}$ (O)	PCS2 (I)	D0	–	x
PCS3[c]	$\overline{\text{CS3}}$ (O)	PCS3 (I)	D0	–	x
PCS4[c]	–	PCS4 (I)	C	–	x
PCS5[c]	–	PCS5 (I)	C	–	x
PCS6[c]	–	PCS6 (I)	C	–	x
PCS7[c]	–	PCS7 (I)	C	–	x

[a] A: analog noise filter; B: analog and digital noise filter; –: no noise filter
[b] S1: Schmitt trigger (30/70%); S2: Schmitt trigger (40/80%); x: CMOS
[c] not available for V850ES/FF3, V850ES/FG3

Table 2-49 Port group CS: configuration registers

Register	Address	Initial value	Used bits							
PMCCS	FFFF F048$_H$	00$_H$	X	X	X	X	PMCCS3[a]	PMCCS2[a]	PMCCS1[a]	PMCCS0[a]
PMCS	FFFF F028$_H$	FF$_H$	PMCS7[a]	PMCS6[a]	PMCS5[a]	PMCS4[a]	PMCS3[a]	PMCS2[a]	PMCS1	PMCS0
PCS	FFFF F008$_H$	undefined	PCS7[a]	PCS6[a]	PCS5[a]	PCS4[a]	PCS3[a]	PCS2[a]	PCS1	PCS0

[a] not available for V850ES/FF3, V850ES/FG3

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

2.5.18 Port group CT (V850ES/FF3, V850ES/FG3, V850ES/FJ3, V850ES/FK3)

Note Port group CT is available only for V850ES/FF3, V850ES/FG3, V850ES/FJ3 and V850ES/FK3.

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

- External memory interface read strobe ($\overline{\text{RD}}$)
- External memory interface write strobes ($\overline{\text{WR0}}$, $\overline{\text{WR1}}$)
- External memory interface address strobe (ASTB)

Port group CT includes the following pins:

Table 2-50 Port group CT: pin functions and port types

Pin functions in different modes		Pin function after reset	Port type	Noise filter[a]	Input charact.[b]
Port mode (PMCnm = 0)	Alternative mode (PMCnm = 1)				
PCT0	$\overline{\text{WR0}}$ (O)[c]	PCT0 (I)	C D0[c]	–	x
PCT1	$\overline{\text{WR1}}$ (O)[c]	PCT1 (I)	C D0[c]	–	x
PCT2[c]	–	PCT2 (I)	C	–	x
PCT3[c]	–	PCT3 (I)	C	–	x
PCT4	$\overline{\text{RD}}$ (O)[c]	PCT4 (I)	C D0[c]	–	x
PCT5[c]	–	PCT5 (I)	C	–	x
PCT6	ASTB (O)[c]	PCT6 (I)	C D0[c]	–	x
PCT7[c]	–	PCT7 (I)	C	–	x

[a] A: analog noise filter; B: analog and digital noise filter; –: no noise filter
[b] S1: Schmitt trigger (30/70%); S2: Schmitt trigger (40/80%); x: CMOS
[c] not available for V850ES/FF3, V850ES/FG3

Table 2-51 Port group CT: configuration registers

Register	Address	Initial value	Used bits							
PMCCT[a]	FFFF $F04A_H$	00_H	X	PMCCT6	X	PMCCT4	X	X	PMCCT1	PMCCT0
PMCT	FFFF $F02A_H$	FF_H	PMCT7[a]	PMCT6	PMCT5[a]	PMCT4	PMCT3[a]	PMCT2[a]	PMCT1	PMCT0
PCT	FFFF $F00A_H$	undefined	PCT7[a]	PCT6	PCT5[a]	PCT4	PCT3[a]	PCT2[a]	PCT1	PCT0

[a] not available for V850ES/FF3, V850ES/FG3

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

2.5.19 Port group DL

Port group DL is an 16-bit input/output port group. In alternative mode, it comprises pins for the following functions:

- External memory interface address/data lines 0 to 15 (AD0 to AD15)

Port group DL includes the following pins:

Table 2-52 Port group DL: pin functions and port types

Pin functions in different modes		Pin function after reset	Port type	Noise filter[a]	Input charact.[b]
Port mode (PMCnm = 0)	Alternative mode (PMCnm = 1)				
PDL0	AD0 (I/O)[f]	PDL0 (I)	C D2[f]	–	x/S2[c]
PDL1	AD1 (I/O)[f]	PDL1 (I)	C D2[f]	–	x/S2[c]
PDL2	AD2 (I/O)[f]	PDL2 (I)	C D2[f]	–	x/S2[c]
PDL3	AD3 (I/O)[f]	PDL3 (I)	C D2[f]	–	x/S2[c]
PDL4	AD4 (I/O)[f]	PDL4 (I)	C D2[f]	–	x/S2[c]
PDL5	AD5 (I/O)[f]/FLMD1 (I)	PDL5 (I)	C D2[f]	–	x/S2[c]
PDL6	AD6 (I/O)[f]	PDL6 (I)	C D2[f]	–	x/S2[c]
PDL7	AD7 (I/O)[f]	PDL7 (I)	C D2[f]	–	x/S2[c]
PDL8[d]	AD8 (I/O)[f]	PDL8 (I)	C D2[f]	–	x/S2[c]
PDL9[d]	AD9 (I/O)[f]	PDL9 (I)	C D2[f]	–	x/S2[c]
PDL10[d]	AD10 (I/O)[f]	PDL10 (I)	C D2[f]	–	x/S2[c]
PDL11[d]	AD11 (I/O)[f]	PDL11 (I)	C D2[f]	–	x/S2[c]
PDL12[e]	AD12 (I/O)[f]	PDL12 (I)	C D2[f]	–	x/S2[c]
PDL13[e]	AD13 (I/O)[f]	PDL13 (I)	C D2[f]	–	x/S2[c]
PDL14[f]	AD14 (I/O)[f]	PDL14 (I)	D2	–	x/S2[c]
PDL15[f]	AD15 (I/O)[f]	PDL15 (I)	D2	–	x/S2[c]

a) A: analog noise filter; B: analog and digital noise filter; –: no noise filter

b) S1: Schmitt trigger (30/70%); S2: Schmitt trigger (40/80%); x: CMOS

c) x: for V850ES/FJ3 and V850ES/FK3
S2: for V850ES/FE3, V850ES/FF3, V850ES/FG3

d) not available for V850ES/FE3

e) not available for V850ES/FE3, V850ES/FF3

f) not available for V850ES/FE3, V850ES/FF3, V850ES/FG3

Table 2-53 Port group DL: configuration registers

Register	Address	Initial value	Used bits							
PMCDLL[a]	FFFF F044$_H$	00$_H$	PMCDL7	PMCDL6	PMCDL5	PMCDL4	PMCDL3	PMCDL2	PMCDL1	PMCDL0
PMCDLH[a]	FFFF F045$_H$	00$_H$	PMCDL15[a]	PMCDL14[a]	PMCDL13[b]	PMCDL12[b]	PMCDL11	PMCDL10	PMCDL9	PMCDL8
PMCDL (16 bit)[a]	FFFF F044$_H$	0000$_H$	PMCDL15 to PMCDL8 (PMCDLH)				PMCDL7 to PMCDL0 (PMCDLL)			
PMDLL	FFFF F024$_H$	FF$_H$	PMDL7	PMDL6	PMDL5	PMDL4	PMDL3	PMDL2	PMDL1	PMDL0
PMDLH[c]	FFFF F025$_H$	FF$_H$	PMDL15[a]	PMDL14[a]	PMDL13[b]	PMDL12[b]	PMDL11	PMDL10	PMDL9	PMDL8
PMDL (16 bit)[c]	FFFF F024$_H$	FFFF$_H$	PMDL15 to PMDL8 (PMDLH)				PMDL7 to PMDL0 (PMDLL)			
PDLL	FFFF F004$_H$	undefined	PDL7	PDL6	PDL5	PDL4	PDL3	PDL2	PDL1	PDL0
PDLH[c]	FFFF F005$_H$	undefined	PDL15[a]	PDL14[a]	PDL13[b]	PDL12[b]	PDL11	PDL10	PDL9	PDL8
PDL (16 bit)[c]	FFFF F004$_H$	undefined	PDL15 to PDL8 (PDLH)				PDL7 to PDL0 (PDLL)			

[a] not available for V850ES/FE3, V850ES/FF3, V850ES/FG3
[b] not available for V850ES/FE3, V850ES/FF3
[c] not available for V850ES/FE3

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

All 16-bit registers can be accessed in 16-bit units.

2.6 Noise Elimination

The input signals at some pins are passing a filter to remove noise and glitches. The microcontroller supports both analog and digital filters.

In *Table 2-17 on page 114* and in the following tables it is listed whether a pin is equipped with an analog filter, a digital filter, both analog and digital filter, or no filter at all.

2.6.1 Analog filtered inputs

The following input signals are passed through an analog filter to remove noise and glitches:

- Non-maskable interrupt (NMI)
- External interrupts (INTPn)
- Key interrupt inputs (KRn)
- Timer TAA trigger inputs (TIAAnm)
- Timer TAB trigger inputs (TIABnm)
- A/D converter external input triggers (ADTRG, ADTRG1)
- N-Wire debug interface reset (DRST)

The analog filter suppresses input pulses that are shorter than a specified pulse width (refer to the Datasheet). This assures the hold time for the external interrupt signals.

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

2.6.2 Digitally filtered inputs

The input signal INTP3 is passed through both an analog and a digital filter.

The digital filter operates in all modes, in which f_{XX} is available. Thus, it does not operate in standby modes (if f_{XT} is used as the sampling clock, it can operate in standby modes). The digital filter is only effective if the corresponding pin works in alternative input mode and not as a general purpose I/O port.

Filter operation

The input terminal signal is sampled with the sampling frequency $f_{s.}$ Spikes shorter than N-1 sampling cycles are suppressed and no internal signal is generated. Pulses longer than N sampling cycles are recognized as valid pulses and an internal signal is generated. The pulses between N-1 and N sampling cycles are eliminated as noise, or detected as a valid edge. The characteristics of the digital filter can be set by the NFC register.

The characteristic of the digital noise filter is determined by the register NFC:

- f_s is defined by NFC.NFC[2:0]

  f_s is the sampling frequency. Together with N it defines the minimum input terminal pulse width to be validated.
- N is defined by NFC.NFSTS
  Possible values for N are 2 or 3.

The filter operation is illustrated in *Figure 2-59* for NFC.NFSTS = 0 (N = 3).

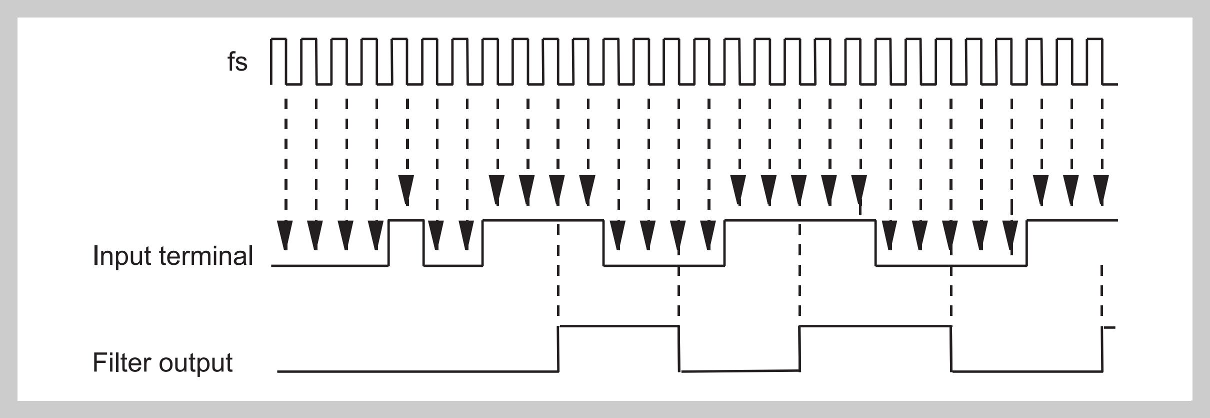


Figure 2-59 Digital noise removal example for NFC.NFSTS = 0 (N = 3)

(1) NFC - Digital noise filter control register

The 8-bit NFC register specifies the noise elimination circuit for signal INTP3.

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

Address FFFF $F318_H$

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

	7	6	5	4	3	2	1	0
NFC	NFEN	NFSTS	0	0	0	NFC2	NFC1	NFC0
	R/W	R/W	R	R	R	R/W	R/W	R/W

Table 2-54 NFC register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>7</td><td>NFEN</td><td>Enables/disables digital noise elimination at pin INTP3:
0: Digital noise elimination is disabled.
1: Digital noise elimination is enabled.</td></tr>
<tr><td>6</td><td>NFSTS</td><td>Defines the number of sampling periods N of f_S to validate the external signal:
0: N = 3
1: N = 2</td></tr>
<tr><td>2 to 0</td><td>NFC[2:0]</td><td>Defines the sampling frequency f_s for digital noise removal:
<table>
<tr><th rowspan="2">NFC2</th><th rowspan="2">NFC1</th><th rowspan="2">NFC0</th><th colspan="2">Sampling frequency f_S</th></tr>
<tr><th>Option bit PRSI = 0</th><th>Option bit PRSI = 1</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>$f_{XX}/64$</td><td>$f_{XX}/128$</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>$f_{XX}/128$</td><td>$f_{XX}/256$</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>$f_{XX}/256$</td><td>$f_{XX}/512$</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>$f_{XX}/512$</td><td>$f_{XX}/1024$</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>$f_{XX}/1024$</td><td>$f_{XX}/2048$</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>f_{XT}</td><td>f_{XT}</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td rowspan="2">setting prohibited</td><td rowspan="2">setting prohibited</td></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
</table>
</td></tr>
</table>

Note

1. f_{XX} = system clock
   f_{XT} = Sub oscillator frequency
2. Specification of PRSI is an option byte (refer to *"Flash Memory" on page 298*).

Remark If f_S is set to $f_{XX}/64$, $f_{XX}/128$, $f_{XX}/256$, $f_{XX}/512$, $f_{XX}/1024$, or $f_{XX}/2048$, it cannot be used to release the standby mode, because the sampling clock stops in the IDLE1, IDLE2 mode, or STOP mode. In this case, set fs to f_{XT} or connect the analog noise elimination circuit (setting not to execute digital noise elimination) to release the standby mode.

Caution After the sampling clock has been changed, it takes N sampling clocks (defined sampling frequency N = 3 or 2) to initialize the digital noise eliminator. Therefore, if an INTP3 valid edge is input within these N sampling clocks time after the sampling clock has been changed, an interrupt request signal may be generated.
Therefore, be careful about the following points when using the interrupt and DMA functions.

When using the interrupt function, after the N sampling clocks (selected sampling frequency N = 3 or 2) have elapsed, enable interrupts after the interrupt request flag (PIC3.PIF3 bit) has been cleared.
When using the DMA function (started by INTP3), enable DMA after the N sampling clocks have elapsed.

2.7 Pin Functions in Reset and Power Save Modes

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

The reset source makes a difference concerning the N-Wire debugger interface pins $\overline{\text{DRST}}$, DDI, DDO, DCK and DMS after reset release. An external $\overline{\text{RESET}}$ or an internal Power-On-Clear switches all pins to input port mode, while all other internal reset sources make the pins available for the debugger.

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

Table 2-55 Pin functions and reset / power save modes

Operating status		Pin status
external $\overline{\text{RESET}}$	during	• P05/$\overline{\text{DRST}}$: P05 port input with internal pull-down resistor • all other pins: Hi-impedence
	after	• P05/$\overline{\text{DRST}}$: $\overline{\text{DRST}}$ input with internal pull-down resistor • P52/DDI, P54/DCK, P55/DMS: DDI, DCK, DMS inputs • P53/DDO: DDO output • all other pins: input port mode
Power-On-Clear (POC)	during	• P05/$\overline{\text{DRST}}$: P05 port input with internal pull-down resistor • all other pins: Hi-impedence
	after	input port mode
all other reset sources	during	• P05/$\overline{\text{DRST}}$, P52/DDI, P53/DDO, P54/DCK, P55/DMS: same as before reset • all other pins: input port mode
	after	
HALT mode	during	same as before HALT mode
	after	
IDLE 1, IDLE 2, STOP mode	during	same as before power save mode: • Output signals are valid and output levels are remained. • Input signals with wake-up capability[a] are valid. • Input signals without wake-up capability are ignored.
	after	same as before power save mode

a) Inputs with wake-up capability: external interrupts (INTP0 to INTP14, NMI) and CAN0 to CAN3 receive data (CRXD0, CRXD1, CRXD2, CRXD3)

2.8 Recommended Connection of unused Pins

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

Table 2-56 Recommended connection of unused pins

Pin	Recommended connection
Port pins	
pins of port groups 0, 1, 3 to 6, 8, 9, 15 (except P05 of Port 0)	• output pins: leave open • input pins: connect to EVDD or EVSS via a resistor, or use the internal pull-up resistor
P05 of port group 0	• output pins: leave open • input pins: connect to EVSS via a resistor
pins of port groups 7, 12	• output pins: leave open • input pins: connect to AVREF0 or AVSS via a resistor
pins of port groups CD, CM, CS, CT, DL	• output pins: leave open • input pins: connect to BVDD or BVSS via a resistor
Non-port pins	
AVREF0, AVREF1	connect to VDD
FLMD0	connect to VSS
REGC, REGC2	connect to regulator output stability capacity
XT1	connect to VSS via a resistor
XT2	leave open
Internally connected pins	
IC	connect to VSS via a resistor

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 Datasheet.

2.9 Package Pins Assignment

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

2.9.1 V850ES/FE3 package pins assignment

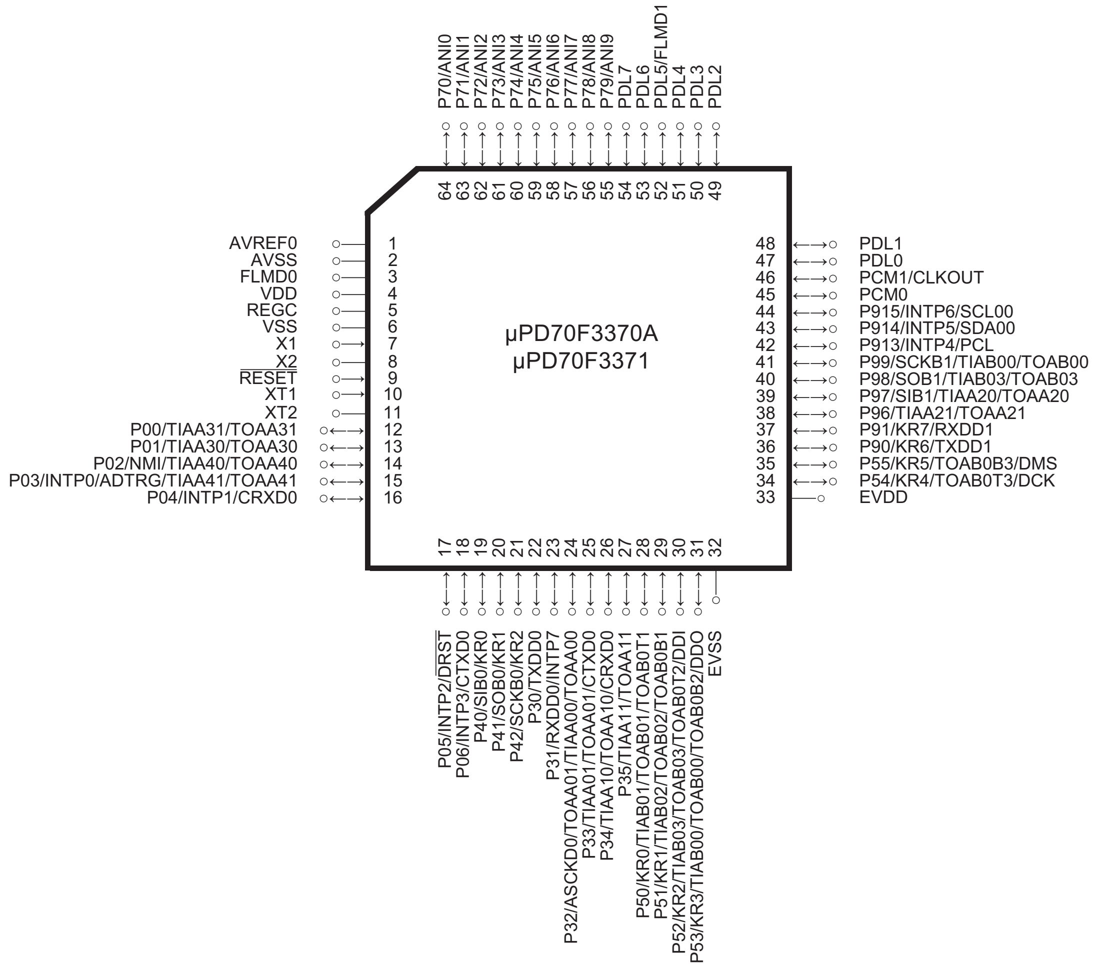


Figure 2-60 V850ES/FE3 package pin assignment

2.9.2 V850ES/FF3 package pins assignment

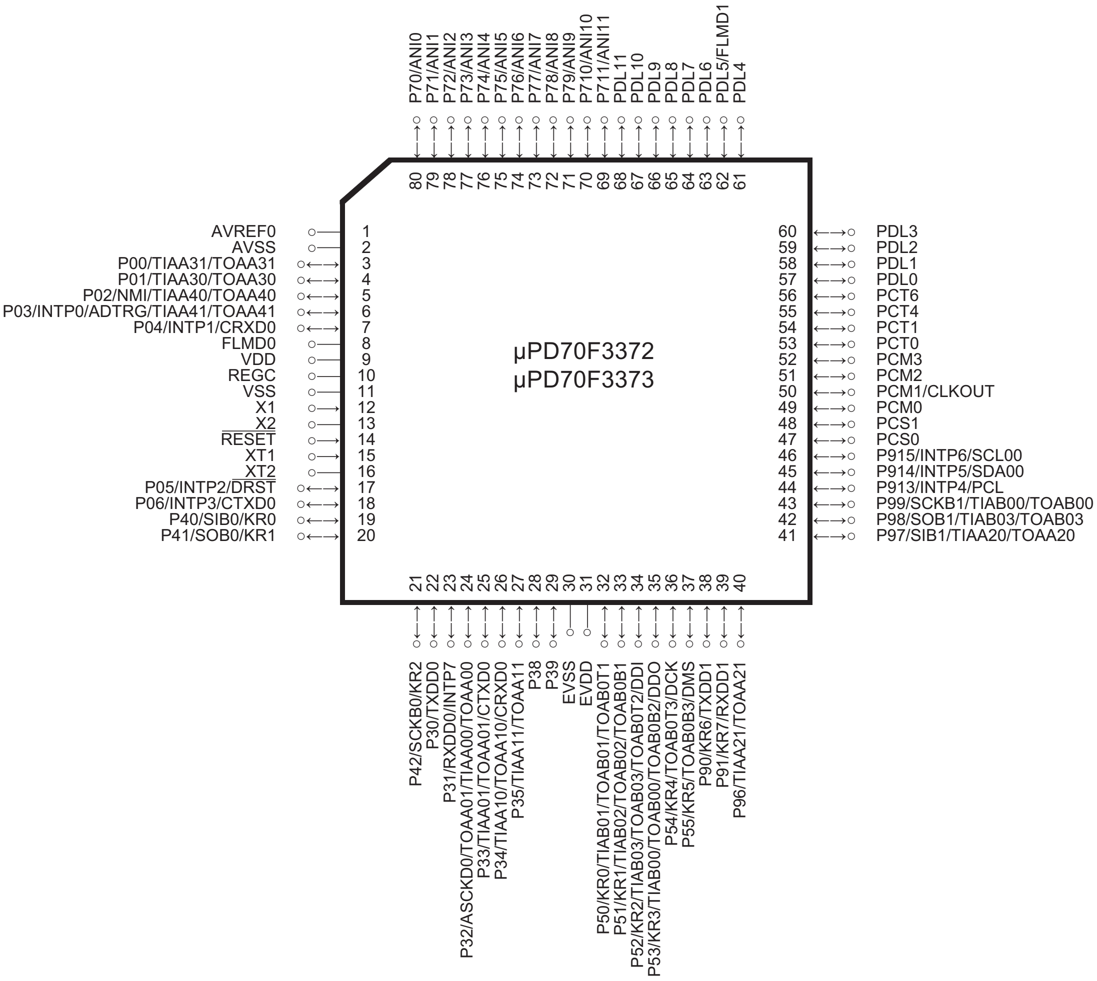
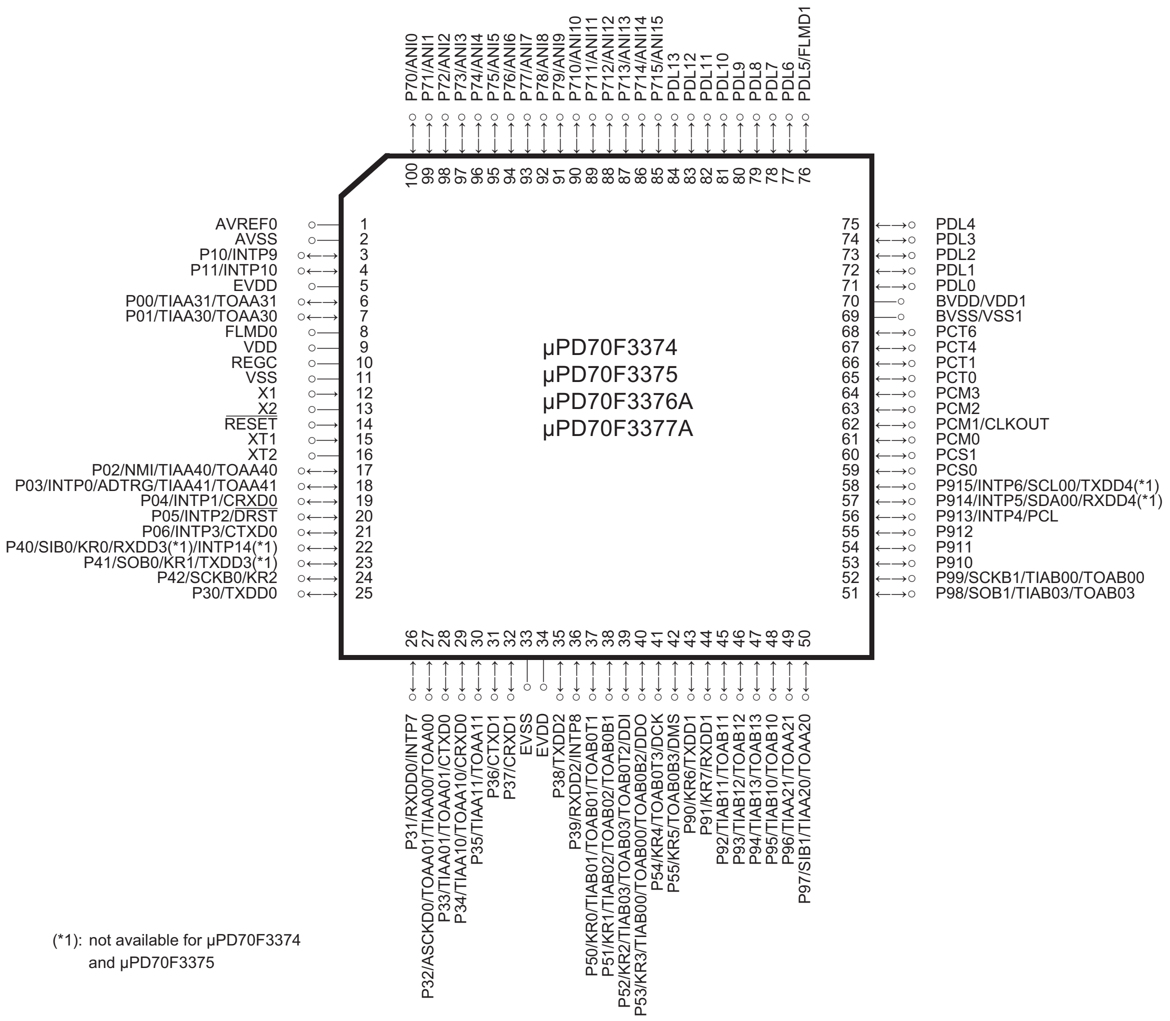


Figure 2-61 V850ES/FF3 package pin assignment

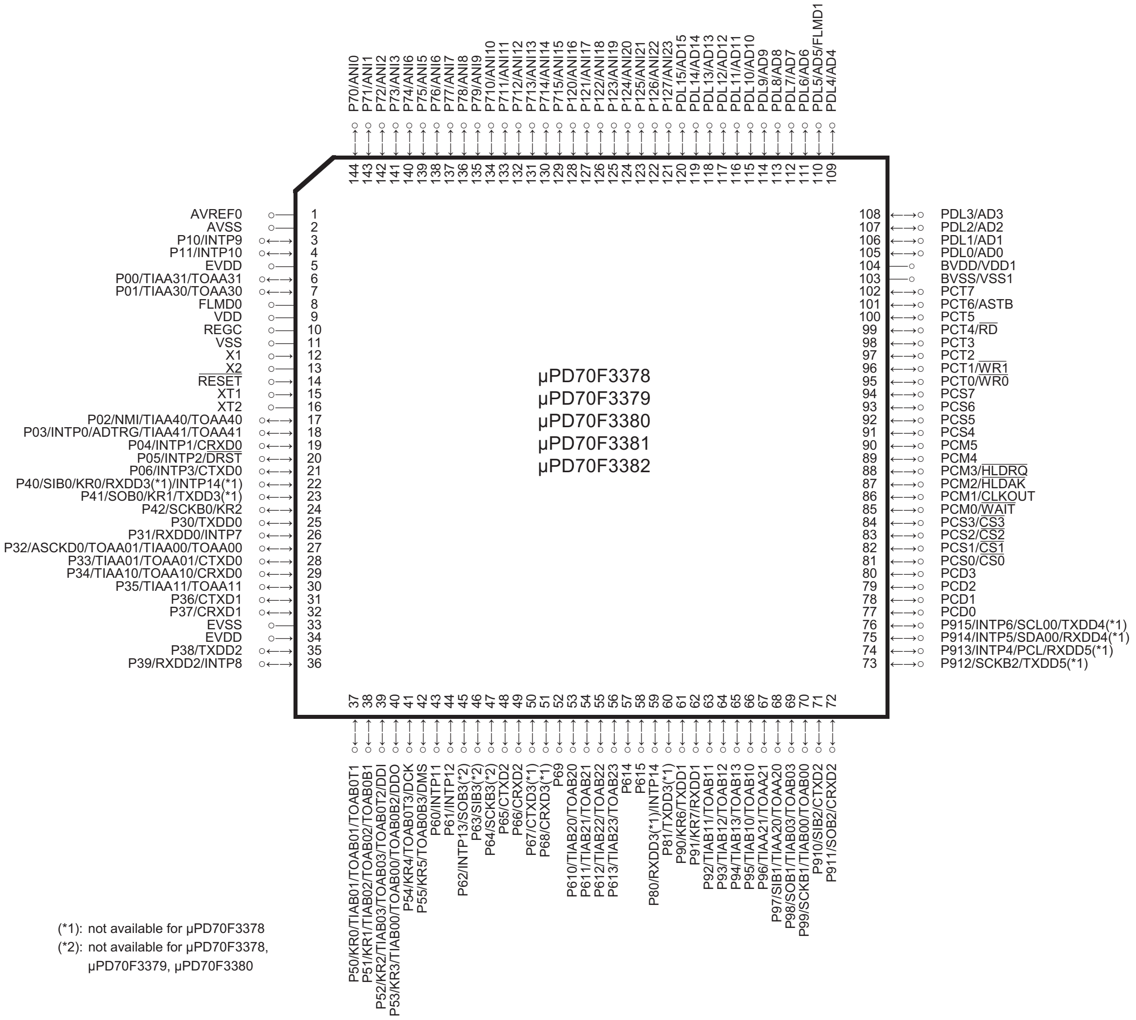
2.9.3 V850ES/FG3 package pins assignment



(*1): not available for µPD70F3374
and µPD70F3375

Figure 2-62 V850ES/FG3 package pin assignment

2.9.4 V850ES/FJ3 package pins assignment



(*1): not available for μPD70F3378

(*2): not available for μPD70F3378, μPD70F3379, μPD70F3380

Figure 2-63 V850ES/FJ3 package pin assignment

2.9.5 V850ES/FK3 package pins assignment

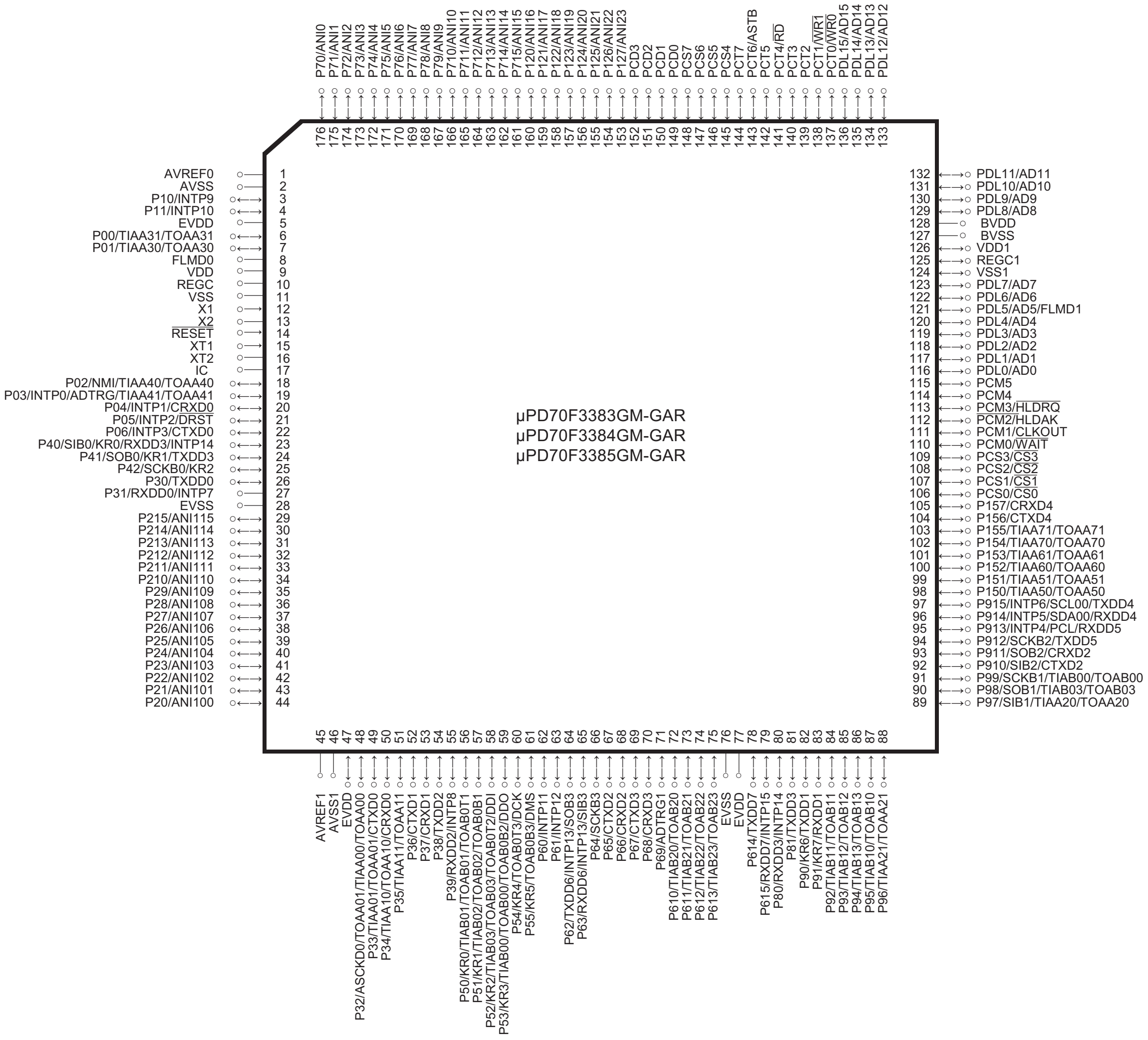


Figure 2-64 V850ES/FK3 package pin assignment

Chapter 3 CPU System Functions

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

3.1 Overview

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

In order to make the microcontroller ideal for use in digital control applications, a 16-bit hardware multiplier enables this CPU to support word/half-word multiply instructions, saturated multiply instructions, bit operation instructions, etc.

Features summary The CPU has the following special features:

- Memory space:
  - 64 MB linear program space
  - 4 GB linear data space
- 32 general purpose registers
- Internal 32-bit architecture
- Five-stage pipeline
- Efficient multiplication and division instructions
- Saturation logic (saturated operation instructions)
- Barrel shifter (32-bit shift in one clock cycle)
- Instruction formats: long and short
- Four types of bit manipulation instructions: set1, clear1, not1, test1

3.1.1 Description

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

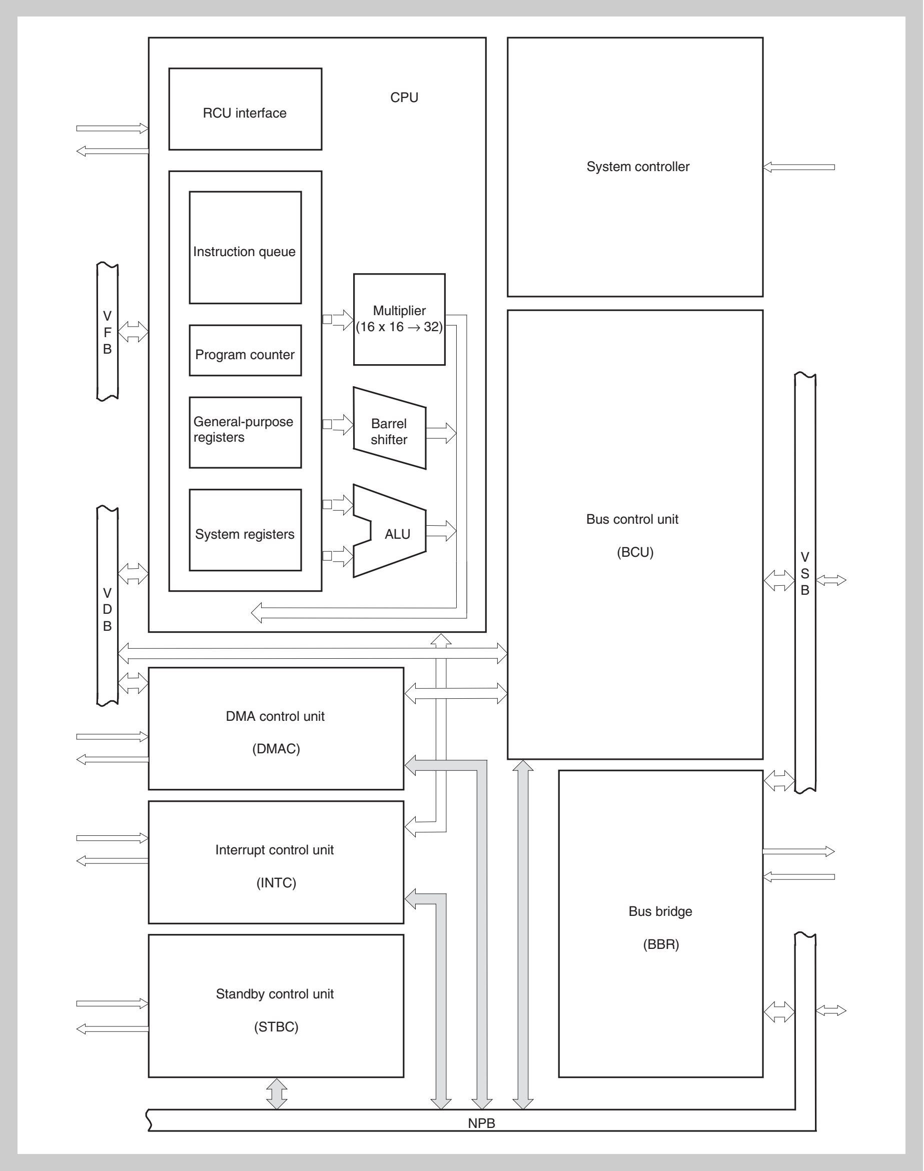


Figure 3-1 CPU system

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

Table 3-1 Bus types

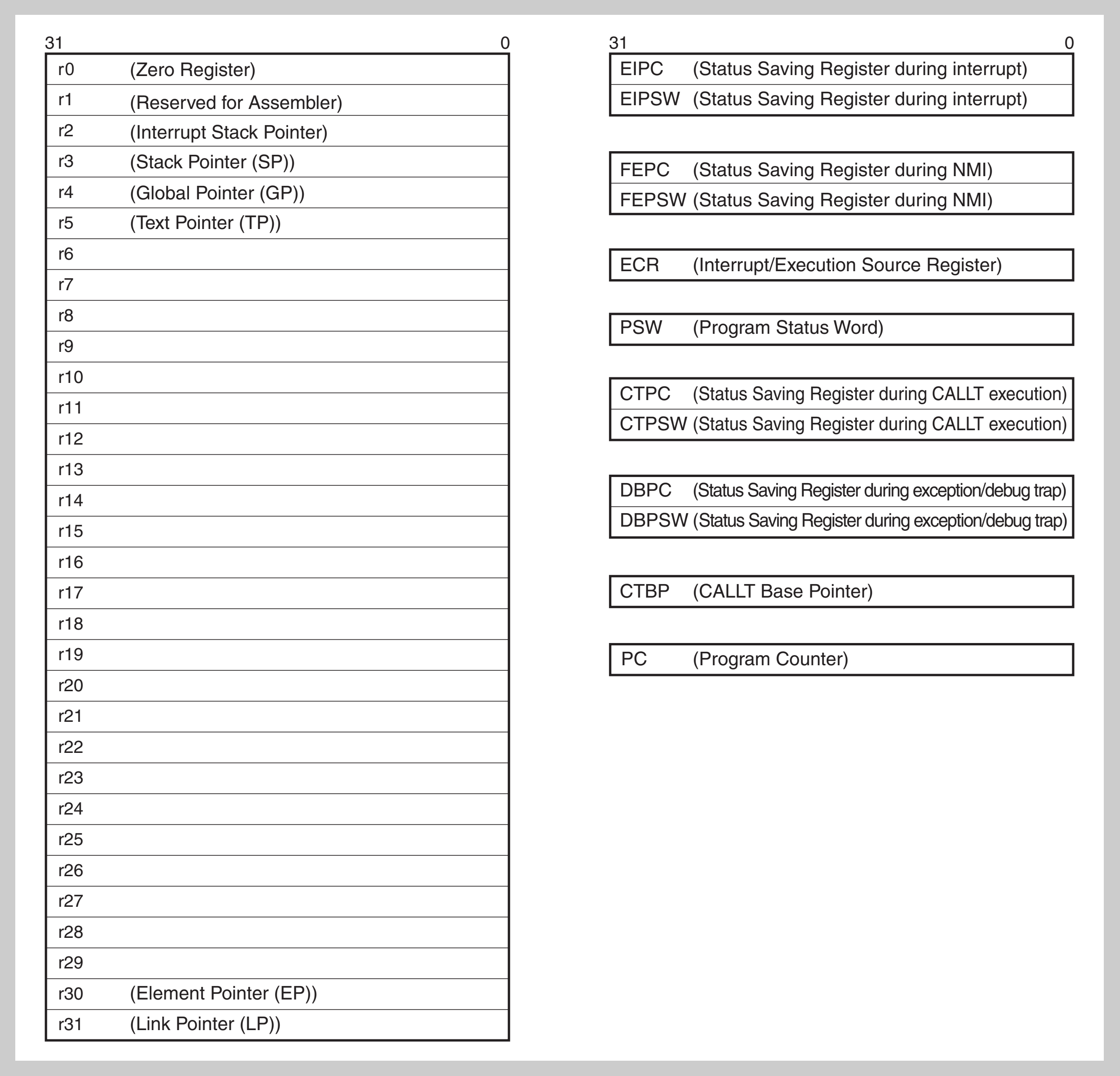
Bus type	Function
NPB – Peripheral bus	Bus interface to the peripherals (internal bus).
VSB – System bus	Bus interface to the Memory Controller for access to external memory and to the NPB bus bridge BBR.
VFB – Fetch bus	Interface to the internal ROM (mask ROM or code flash).
VDB – Data bus	Interface 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 figure below. For details, refer to V850ES User’s Manual Architecture.



r0	(Zero Register)
r1	(Reserved for Assembler)
r2	(Interrupt Stack Pointer)
r3	(Stack Pointer (SP))
r4	(Global Pointer (GP))
r5	(Text Pointer (TP))
r6	
r7	
r8	
r9	
r10	
r11	
r12	
r13	
r14	
r15	
r16	
r17	
r18	
r19	
r20	
r21	
r22	
r23	
r24	
r25	
r26	
r27	
r28	
r29	
r30	(Element Pointer (EP))
r31	(Link Pointer (LP))

EIPC	(Status Saving Register during interrupt)
EIPSW	(Status Saving Register during interrupt)
FEPC	(Status Saving Register during NMI)
FEPSW	(Status Saving Register during NMI)
ECR	(Interrupt/Execution Source Register)
PSW	(Program Status Word)
CTPC	(Status Saving Register during CALLT execution)
CTPSW	(Status Saving Register during CALLT execution)
DBPC	(Status Saving Register during exception/debug trap)
DBPSW	(Status Saving Register during exception/debug trap)
CTBP	(CALLT Base Pointer)
PC	(Program Counter)

Figure 3-2 CPU register set

3.2.1 General purpose registers (r0 to r31)

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

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

Table 3-2 General purpose registers

Register name	Usage	Operation
r0	Zero register	Always holds 0. It is used for operations using 0 and offset 0 addressing.[a]
r1	Assembler-reserved register	Used for 32-bit direct addressing.[b]
r2	User address/data variable register	
r3	Stack pointer	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. The contents must be restored to the registers after the registers have been used.

3.2.2 System register set

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

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

Example STSR 0, r2

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

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

Table 3-3 System register numbers

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

[a] Reading from this register is only enabled between a DBTRAP exception (exception handler address 0000 0060_H) and the exception handler terminating DBRET instruction. DBTRAP exceptions are generated upon ILGOP detections (refer to *“Interrupt Controller (INTC)” on page 248*).

(1) PC - Program counter

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

Access This register can not be accessed by any instruction.

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

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

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

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

When one of the occasions listed in *Table 3-4* occurs, except for some instructions, the address of the instruction following the one being executed is saved to the saving registers.
For more details refer to *Table 3-9 on page 164* and to the *“Interrupt Controller (INTC)” on page 248*.

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

Table 3-4 PC saving registers

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

a) Reading from this register is only enabled between a DBTRAP exception (exception handler address 0000 0060_H) and the exception handler terminating DBRET instruction. DBTRAP exceptions are generated upon ILGOP detections (refer to *“Interrupt Controller (INTC)” on page 248*).

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

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

(3) PSW - Program status word

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

If the bits in the register are modified by the LDSR instruction, the PSW will take on the new value immediately after the LDSR instruction has been executed.
When the ID flag is set to 1, however, acknowledgment of an interrupt request is disabled from when the LDSR instruction is still under execution.

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

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

Table 3-5 PSW register contents (1/2)

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

Table 3-5 **PSW register contents (2/2)**

Bit position	Flag	Function
2	OV[a]	Overflow flag. Indicates whether an overflow occurred as a result of the operation. 0: Overflow did not occur. 1: Overflow occurred.
1	S[a]	Sign flag. Indicates whether the result of the operation is negative. 0: Result is positive or zero. 1: Result is negative.
0	Z	Zero flag. Indicates whether the result of the operation is zero. 0: Result is not zero. 1: Result is zero.

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

Saturated operation instructions

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

Table 3-6 **Saturation-processed operation result**

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

[a] Retains the value before operation.

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

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

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

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

Table 3-7 PSW saving registers

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

[a] Reading from this register is only enabled between a DBTRAP exception (exception handler address 0000 0060_H) and the exception handler terminating DBRET instruction. DBTRAP exceptions are generated upon ILGOP detections (refer to *"Interrupt Controller (INTC)" on page 248*).

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

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

(5) ECR - Interrupt/exception source register

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

For a list of interrupts/exceptions and corresponding exception codes, see *Table 3-9 on page 164*.

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

31 … 26	25 … 0
FECC	EICC

Table 3-8 ECR register contents

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

The following table lists the exception codes.

Table 3-9 Interrupt/execution codes

Interrupt/Exception Source			Classification	Exception Code	Handler Address	Value restored to EIPC/FEPC
Name		Trigger				
Non-maskable interrupts (NMI)		NMI0 input	Interrupt	0010_H	$0000\ 0010_H$	next PC (see Note)
		NMI1 input	Interrupt	0020_H	$0000\ 0020_H$	next PC (see Note)
		NMI2 input	Interrupt	0030_H	$0000\ 0030_H$	next PC (see Note)
Maskable interrupt		refer to *“Interrupt Controller (INTC)” on page 248*	Interrupt	refer to *“Interrupt Controller (INTC)” on page 248*	• 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, generally, the address of the instruction *following* the one being executed is saved to the saving registers, except when an interrupt is acknowledged during execution of one of the following instructions:

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

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

(6) CTBP - CALLT base pointer

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

Initial Value Undefined

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

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

3.3 Operation Modes

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

The following operation modes are available:

- Normal operation mode
- Flash programming mode
- On-chip debug mode

After reset release, the microcontroller starts to fetch instructions from an internal boot ROM which contains the internal firmware. The firmware checks the pins FLMD0 and FLMD1 (PDL5) to set the operation mode after reset release according to *Table 3-10*:

Table 3-10 Selection of operation modes

Pins		Operation Mode
FLMD0	**FLMD1 (PDL5)**	
0	X	Normal operation mode (fetch from code flash)
1	0	Flash programming mode
	1	Setting prohibited

Note The FLMD1 pin function is shared with the PDL5 pin.

3.3.1 Normal operation mode

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

Note The lower 1 MB of the memory area is always mapped to the internal code flash memory. Thus, external memory mapped to this area can not be addressed in normal operation mode. See also *"Bus and Memory Control (BCU, MEMC)" on page 339*.

3.3.2 Flash programming mode

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

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

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

For more information see section *"Flash Memory" on page 298*.

3.3.3 On-Chip debug mode

By connecting an N-Wire emulator, on-chip debugging can be executed. The N-Wire emulator is connected through JTAG type signals.
In On-Chip debug mode user's code can be programmed into the flash. Afterwards the software can be evaluated using breakpoints and the user resources (such as memory and I/O can be read or written.

For more information see *Chapter 25 on page 930*.

3.4 Address Space

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

3.4.1 CPU address space and physical address space

The CPU supports the following address space:

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

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

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

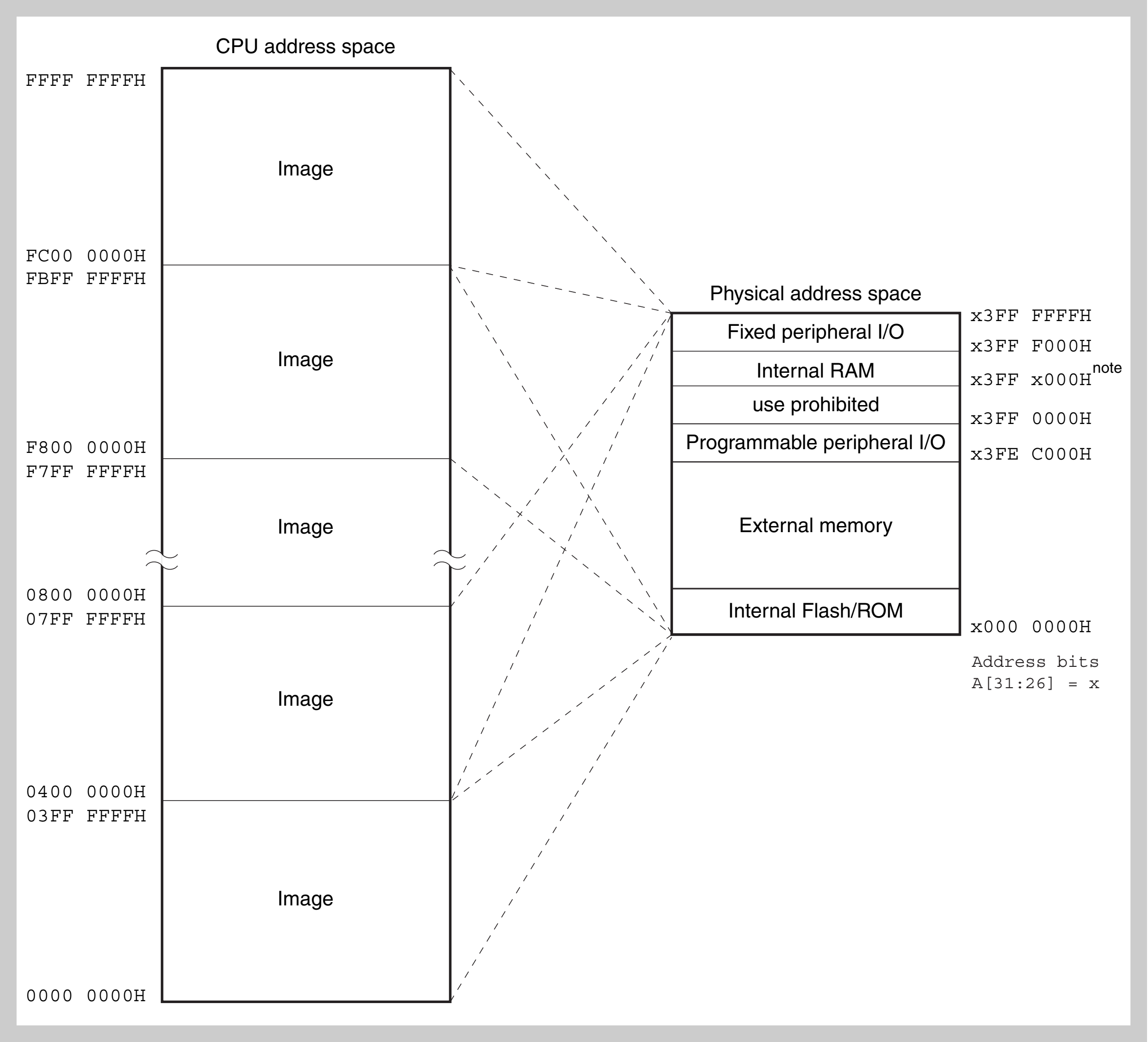


Figure 3-3 Images in the CPU address space

Note The start address of the internal RAM area depends on the product derivative. See *"Internal RAM area" on page 171* for details.

3.4.2 Program and data space

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

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

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

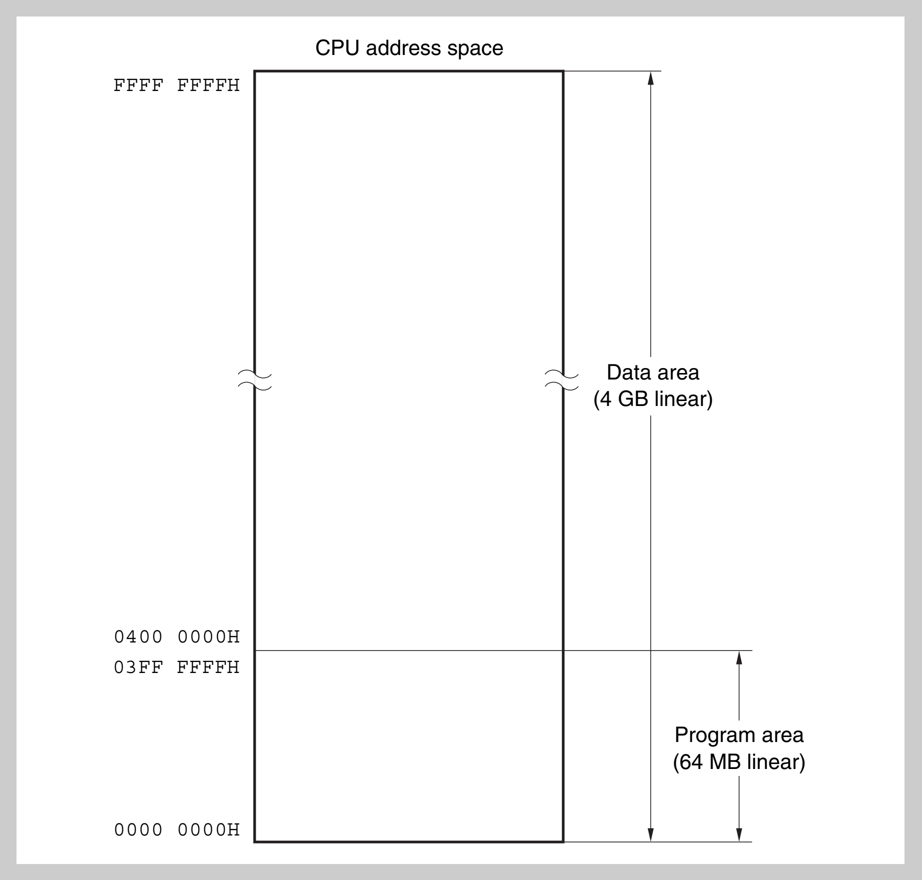


Figure 3-4 CPU address space

(1) Wrap-around of data space

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

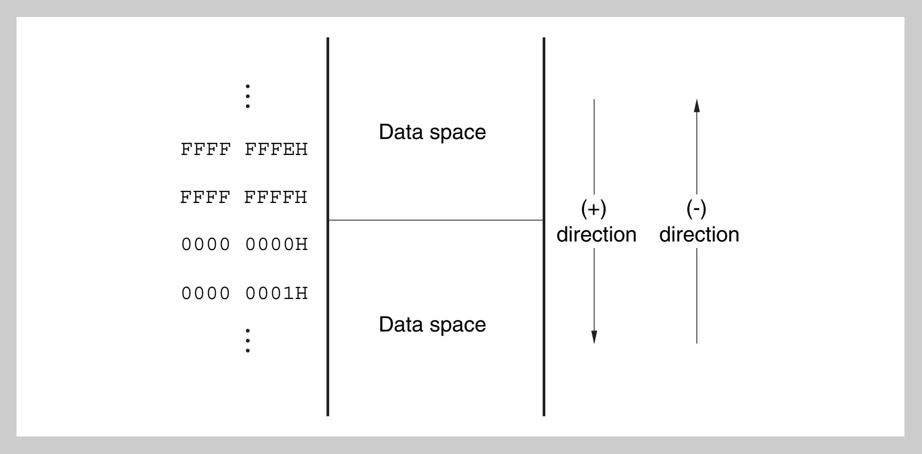


Figure 3-5 Wrap-around of data space

(2) Wrap-around of program space

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

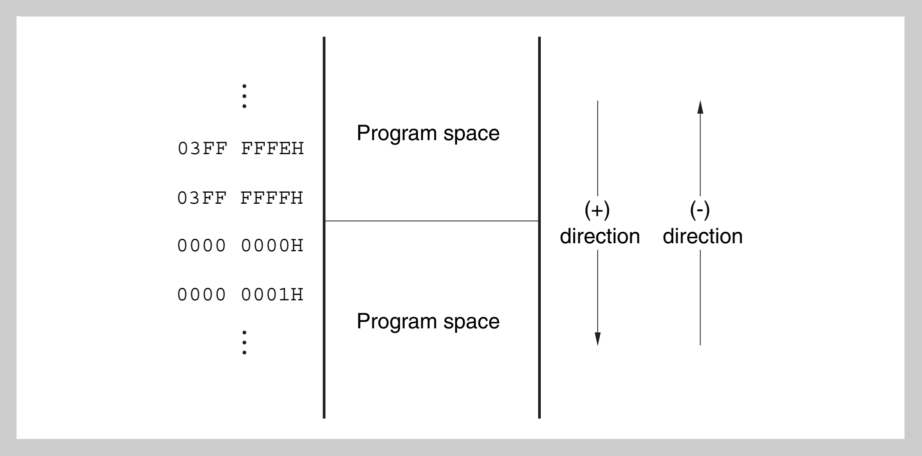


Figure 3-6 Wrap-around of program space

Caution No instruction can be fetched from the 4 KB area of 03FF F000$_H$ to 03FF FFFF$_H$ because this area is defined as peripheral I/O area. Therefore, do not execute any branch to this area.

3.5 Memory

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

3.5.1 Memory areas

The internal memory of the CPU provides several areas:

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

The areas are briefly described below.

(1) Internal code flash area

Table 3-11 shows the size and address range of internal code flash area. The internal code flash area cannot be used for access to external memory.

Table 3-11 Internal code flash areas V850ES/Fx3

Product	Device	Code flash size	Address range
V850ES/FE3	µPD70F3370A	128 KB	0000 0000_H to 0001 $FFFF_H$
	µPD70F3371	256 KB	0000 0000_H to 0003 $FFFF_H$
V850ES/FF3	µPD70F3372	128 KB	0000 0000_H to 0001 $FFFF_H$
	µPD70F3373	256 KB	0000 0000_H to 0003 $FFFF_H$
V850ES/FG3	µPD70F3374	128 KB	0000 0000_H to 0001 $FFFF_H$
	µPD70F3375	256 KB	0000 0000_H to 0003 $FFFF_H$
	µPD70F3376A	384 KB	0000 0000_H to 0005 $FFFF_H$
	µPD70F3377A	512 KB	0000 0000_H to 0007 $FFFF_H$
V850ES/FJ3	µPD70F3378	256 KB	0000 0000_H to 0003 $FFFF_H$
	µPD70F3379	384 KB	0000 0000_H to 0005 $FFFF_H$
	µPD70F3380	512 KB	0000 0000_H to 0007 $FFFF_H$
	µPD70F3381	768 KB	0000 0000_H to 000B $FFFF_H$
	µPD70F3382	1 MB	0000 0000_H to 000F $FFFF_H$
V850ES/FK3	µPD70F3383	512 KB	0000 0000_H to 0007 $FFFF_H$
	µPD70F3384	768 KB	0000 0000_H to 000B $FFFF_H$
	µPD70F3385	1 MB	0000 0000_H to 000F $FFFF_H$

(2) Internal RAM area

Table 3-12 shows the size and address range of internal RAM area.

Table 3-12 **Internal RAM areas V850ES/Fx3**

Product	Device	RAM size	Address range
V850ES/FE3	µPD70F3370A	8 KB	03FF D000$_H$ – 03FF EFFF$_H$
	µPD70F3371	16 KB	03FF B000$_H$ – 03FF EFFF$_H$
V850ES/FF3	µPD70F3372	8 KB	03FF D000$_H$ – 03FF EFFF$_H$
	µPD70F3373	16 KB	03FF B000$_H$ – 03FF EFFF$_H$
V850ES/FG3	µPD70F3374	8 KB	03FF D000$_H$ – 03FF EFFF$_H$
	µPD70F3375	16 KB	03FF B000$_H$ – 03FF EFFF$_H$
	µPD70F3376A	24 KB	03FF 9000$_H$ – 03FF EFFF$_H$
	µPD70F3377A	32 KB	03FF 7000$_H$ – 03FF EFFF$_H$
V850ES/FJ3	µPD70F3378	16 KB	03FF B000$_H$ – 03FF EFFF$_H$
	µPD70F3379	24 KB	03FF 9000$_H$ – 03FF EFFF$_H$
	µPD70F3380	32 KB	03FF 7000$_H$ – 03FF EFFF$_H$
	µPD70F3381	40 KB	03FF 5000$_H$ – 03FF EFFF$_H$
	µPD70F3382	48 KB	03FF 3000$_H$ – 03FF EFFF$_H$
V850ES/FK3	µPD70F3383	32 KB	03FF 7000$_H$ – 03FF EFFF$_H$
	µPD70F3384	48 KB	03FF 3000$_H$ – 03FF EFFF$_H$
	µPD70F3385	60 KB	03FF 0000$_H$ – 03FF EFFF$_H$

Note that the internal firmware, which is processed after reset, uses some RAM (refer to *“Reset” on page 942*).

(3) Fixed peripheral I/O area

The 4 KB area between addresses 03FF F000$_H$ and 03FF FFFF$_H$ is provided as the internal 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 this 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 *“Special Function Registers” on page 950*.

Note
1. Because the physical address space covers 64 MB, the address bits A[31:26] are not considered. Thus, this 4 KB address space can also be addressed via the area FFFF 0000$_H$ to FFFF FFFF$_H$. This has the advantage that the area can be indirectly addressed by an offset and the zero base r0.
Therefore, in this manual, all addresses of peripheral I/O registers in the 4 KB peripheral I/O area are given in the range FFFF F000$_H$ to FFFF FFFF$_H$ instead of 03FF F000$_H$ to 03FF FFFF$_H$.
2. The *fixed* peripheral I/O area is mirrored to the upper 4 KB of the *programmable* peripheral I/O area PPA. If data is written to one area, it appears also in the other area.
3. Program fetches cannot be executed from any peripheral I/O area.

4. Word registers, that means 32-bit registers, are accessed in two half word accesses. The lower two address bits are ignored.
5. For registers in which byte access is possible, if half word access is executed:
   - During read operation: The higher 8 bits become undefined.
   - During write operation: The lower 8 bits of data are written to the register.

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

(4) Programmable peripheral I/O area

The 12 KB area between addresses 03FE C000$_H$ and 03FE EFFF$_H$ is provided as a programmable peripheral I/O area (PPA). Within the microcontroller, the usage and address range of the PPA are *not* configurable.

The CAN modules registers and message buffers are allocated to the PPA. Refer to *"CAN module register and message buffer addresses" on page 700* for information on how to calculate the register and message buffer addresses of the CAN modules.

The base address of the PPA is specified by the peripheral area selection control register (BPC).

- For the *microcontroller*, the base address of the PPA is fixed to 03FE C000$_H$. Thus writing to BPC.PA[13:0] does not change the PPA base address. Nevertheless the PPA must be enabled by setting BPC.PA15 = 1.
- For the *emulation tool*, the PPA has to be enabled and the base address has to be set up by writing 8FFB$_H$ to the BPC register.

To make software suitable for both microcontroller and emulation tool, it is recommended to include the set up of the PPA with BPC = 8FFB$_H$ in the software.

(5) External memory area

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

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

For access to external memory, see *“Bus and Memory Control (BCU, MEMC)” on page 339*.

Note Access to external memory is provided only with the V850ES/FJ3 and V850ES/FK3 products.

Data flash The data flash area can be allocated via the external memory area. For access to the data flash area, see *“Flash Memory” on page 298* and *“Bus and Memory Control (BCU, MEMC)” on page 339*.

3.5.2 Recommended use of data address space

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

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

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

For efficient use of the relative addressing feature, the data segments should be located in the address range FFFF F800$_H$ to 0000 0000$_H$ and 0000 0000$_H$ to 0000 7FFF$_H$. The peripheral I/O registers and the internal RAM is aligned to the upper bound, thus the registers and a part of the RAM can be addressed via relative addressing, with base address 0 (r0).

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

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

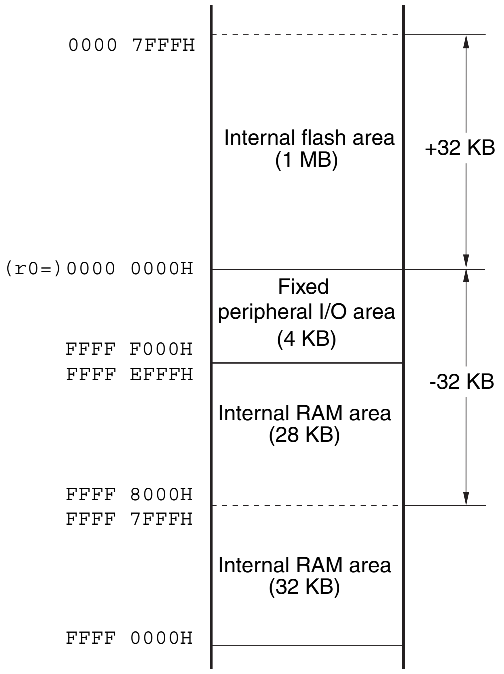

0000 7FFFH
Internal flash area (1 MB)
+32 KB
(r0=)0000 0000H
Fixed peripheral I/O area (4 KB)
FFFF F000H
FFFF EFFFH
-32 KB
Internal RAM area (28 KB)
FFFF 8000H
FFFF 7FFFH
Internal RAM area (32 KB)
FFFF 0000H

Figure 3-7 Example application of wrap-around

3.6 Write Protected Registers

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

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

Incorrect store operations can be checked by a flag of the system status register (SYS).

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.

Table 3-13 Overview of write protected registers

Write protected register	Shortcut	Corresponding write enable register	Shortcut	For details see
Processor clock control register	PCC	Command register	PRCMD	*"Clock Generator Registers" on page 187*
Main system clock mode register	MCM			
SSCG frequency control register 0	SFC0			
SSCG frequency control register 1	SFC1			
Clock Monitor mode register	CLM			
Power save control register	PSC			
Data flash control register	DFLCTL			*"Flash Memory" on page 298*
Reset source flag register	RESF			*"Reset" on page 942*
Internal RAM data status register	RAMS			*"Low-Voltage Detector" on page 920*
Low-voltage detection register	LVIM			
On-chip debug mode register	OCDM			*"On-Chip Debug Unit" on page 930*

Example Enable Clock Monitor

The following example shows how to write to the write protected register CLM. The example enables the Clock Monitor.

```
do {
     _PRERR = 0;
     DI();
     PRCMD = 0x5A;
     CLM = 0x01;
     EI();
} while (_PRERR != 0)
```

Note

1. Make sure that the compiler generates two consecutive assembler "store" instructions to PRCMD and CLM from the associated C statements.
2. Special care must be taken when writing to registers PSC and PRCMD. Please refer to *"Clock Generator" on page 179* for details.
3. Any value can be written to the PRMCD register.

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

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

3.6.1 Write protection control registers

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

(1) PRCMD - Command register

The 8-bit 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, you are permitted to write once to one of the protected registers. This must be done immediately after writing to the PRCMD register. After the second write action all protected registers are write-locked again. Read accesses to any register are permitted between write access to the PRCMD and the protected register.

Any value can be written to the PRCMD register. Nevertheless, writing ’00_H’ or the value to be stored to the protected register in the next instruction minimizes the use of CPU register and the program size.

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

Address FFFF $F1FC_H$.

Initial Value The contents of this register is undefined.

	7	6	5	4	3	2	1	0
PRCMD	×	×	×	×	×	×	×	×
	W	W	W	W	W	W	W	W

An invalid write attempt to one of write protected registers sets the error flag SYS.PRERR.

(2) SYS - System register

The 8-bit SYS register indicates the status of a write attempt to a write protected register.

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

Address FFFF $F802_H$.

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

	7	6	5	4	3	2	1	0
SYS	0	0	0	0	0	0	0	PRERR
	R	R	R	R	R	R	R	R/W

Table 3-14 SYS register contents

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

Chapter 4 Clock Generator

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

4.1 Overview

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

- Main oscillator—a built-in oscillator that requires an external crystal with a frequency between 4 MHz and 16 MHz
- Sub oscillator—a built-in oscillator that requires an external crystal (32,768 KHz) or an external RC resonator (20 KHz)
- Low-frequency internal oscillator—an internal oscillator without external components and a nominal frequency of 240 KHz
- High-frequency internal oscillator—an internal oscillator without external components and a nominal frequency of 8 MHz

Features summary

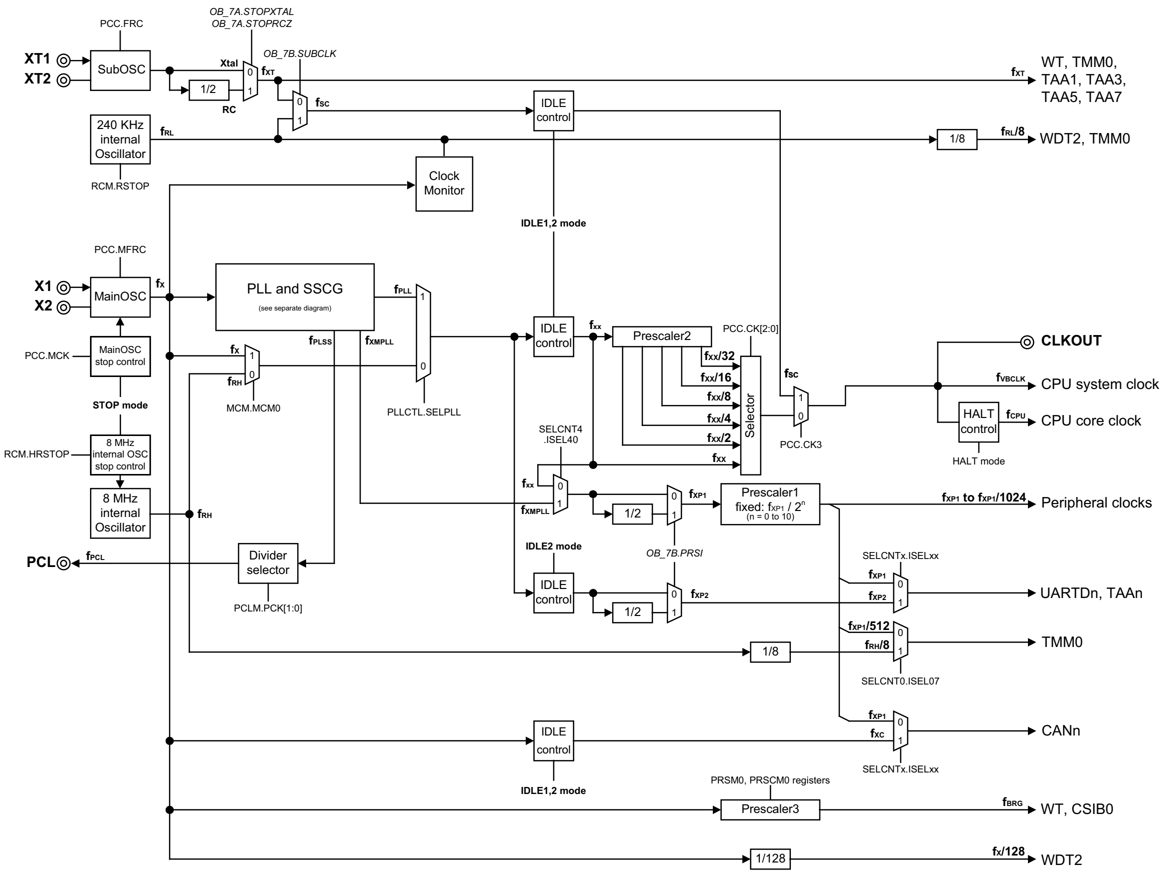
Special features of the Clock Generator are:

- Choice of oscillators to reduce power consumption in stand-by mode
- PLL synthesizer for the main oscillator:
  - In clock-through mode: 4 MHz to 16 MHz main system clock f_{XX}
  - In PLL (Phase Locked Loop) mode main system clock f_{XX}:
    - 12 MHz to 48 MHz with fixed frequency from PLL
    - 12 MHz to ~<48 MHz with modulated frequency from a Spread Spectrum Clock Generator SSCG
- Sub oscillator can be crystal controlled (f_{XT})
- Two internal internal oscillators (f_{RL} = 240 KHz and f_{RH} = 8 MHz)
- Internal main system clock generation:
  - f_{XX}, $f_{XX}/2$, $f_{XX}/4$, $f_{XX}/8$, $f_{XX}/16$, $f_{XX}/32$, f_{XT}, f_{RL}, f_{RH}
  - Subclock mode: f_{XT} or f_{RL} selectable by an option byte in the code flash memory
- Peripheral clock generation
  - f_{XP1} to f_{XP1} / 1024
  - $f_{XP1} = f_{XX}$ or $f_{XP1} = f_{XX}/2$ selectable by an option byte
  - f_{XP1} selectable from fixed (PLL) or modulated (SSCG) clock
- Clock output function (CLKOUT pin)
- Programmable clock output function (PCL pin)
- Individual clock source selection for CPU and groups of peripherals
- Specific power save modes
- Vital system registers are write-protected by a special write sequence

- Direct main oscillator clock feed-through for Watch Timer, Watchdog Timer, CSIB0, and CAN support
- Clock Monitor for main oscillator

4.1.1 Description

The Clock Generator is built up as illustrated in the following figure.



Note: *OB_7A.bitname* and *OB7B.bitname* describe control bits of option byte $007A_H$ and $007B_H$

Figure 4-1 Block diagram of the Clock Generator

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

MainOSC The main oscillator (MainOSC) oscillates at frequencies f_X = 4 MHz to 16 MHz. After reset release, the main oscillator is stopped. Starting the oscillation must be set via software.

The main oscillator is equipped with a stop control.
Oscillation of the main clock oscillator is stopped in the STOP mode or controlled by the PCC register.

SubOSC The sub oscillator (SubOSC) oscillates at a frequency f_{XT} of 32.768 KHz (crystal connected) or typically 20 KHz with an external RC circuit.

240 KHz internal OSC The low speed internal oscillator generates a clock f_{RL} with a frequency of 240 KHz typically. The oscillation can be stopped by means of the RCM register. The oscillation cannot be stopped, if this is disabled by option byte $007A_H$.

8 MHz internal OSC

The high speed internal oscillator generates a clock f_{RH} with a frequency of typically 8 MHz. After reset release, the 8 MHz internal oscillator is activated.

The high speed internal oscillator is equipped with a stop control. The oscillation can be stopped by means of the RCM register.

Main system clock f_{XX}

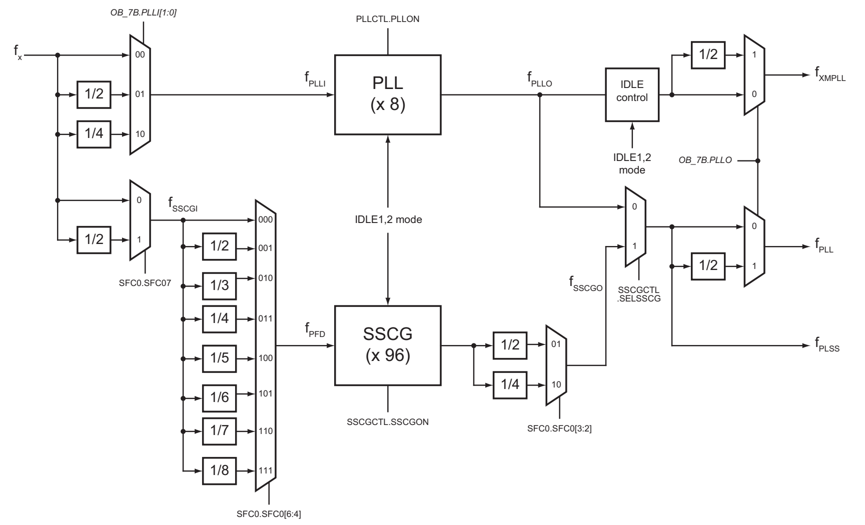
The main system clock f_{XX} can be derived from different sources:

- Clock-through mode: main system clock f_{XX} derived from MainOSC f_X or internal OSC f_{RH}.
- PLL mode, main system clock f_{XX} derived from f_{PLLO} (PLL output) or f_{SSCGO} (SSCG output).

These modes can be selected by the bit PLLCTL.SELPLL.

PLL and SSCG

The PLL and SSCG circuit generates the base frequency f_{PLL}, which can be used as the main system clock f_{xx}. Below figure shows the block diagram of the PLL and SSCG circuit.



Note: *OB7B.bitname* describe control bits of option byte $007B_H$

Figure 4-2 PLL and SSCG block diagram

f_{PLL} can be chosen from the the PLL or the SSCG ouput by a selector, controlled by SSCGCTL.SELSSCG.

A flash memory option byte allows to chose $f_{PLL} = f_{PLLO}$ (f_{SSCGO}) or $f_{XX} = f_{PLLO}/2$ ($f_{SSCGO}/2$) .

Input clock to both PLL and SSCG is the MainOSC f_X.

- PLL

The PLL is preceded by a frequency divider. The input of the PLL (f_{PLLI}) can be set to f_X, $f_X/2$, or $f_X/4$. The divider is set through an option byte in the code flash memory.

The phase-locked loop circuit (PLL) multiplies the MainOSC clock f_X or a fraction of it by eight. Its input clock is called f_{PLLI}, its output is f_{PLLO}.

The PLL is started or stopped by PLLCTL.PLLON. For details on the PLL see also *“Controlling the PLL“ on page 243*.

- SSCG

The SSCG input clock f_{SSCGI} is the MainOSC clock f_X or a fraction of it via two divider stages, controlled by SFC0.SFC07 and SFC0.SFC0[6:4].

The SSCG multiplies f_{SSCGI} by 96 and applies additionally a modulation to its output clock f_{SSCGO}.

The output selector, controlled by SCF0.SCF0[3:2], sets f_{SSCGO} to a half or a quarter of the SSCG’s output frequency.

(1) System and CPU clocks

The CPU system is clocked by two clocks:

- f_{VBCLK} supplies all remaining parts of the CPU system, like DMAC, BCU, MEMC, INTC.
- f_{CPU} is derived from f_{VBCLK} supplies the CPU core and is subject to HALT mode control.

Clock source for both clocks can be the output of the PLL/SSCG or any of the oscillators.

The following table gives an overview of the available CPU clock sources.

Table 4-1 Clock sources for the CPU

Clock source	Frequency	Description
8 MHz internal OSC	~8 MHz	Default clock source after reset release.
240 KHz internal OSC	~240 KHz	Default clock source if MainOSC has stopped.
SubOSC	32 KHz or 20 KHz	Selectable as clock source.
MainOSC	4 to 16 MHz	CPU system clocks in clock-through mode
PLL	up to 48 MHz[a]	For maximum performance
SSCG	up to 48 MHz[a b]	For reduced EMI

[a)] The maximum clock is limited to 32 MHz for the following devices:
– V850ES/FE3
– V850ES/FF3
– µPD70F3374, µPD70F3375 of V850ES/FG3
– µPD70F3378 of V850ES/FJ3

[b)] The max. SSCG output frequency indicates the case without modulation. If modulation is enabled the average SSCG frequency has to be set lower.

The clock sources MainOSC, PLL/SSCG and 8 MHz internal OSC can generate the master clock f_{XX}. This clock forms the input to Prescaler2. Prescaler2 can divide the master clock f_{XX} by 1, 2, 4, 8, 16 or 32. Its operation is set in the PCC register.

Prescaler2, the SubOSC, or the 240 KHz internal OSC can generate the CPU core clock (f_{CPU}) and the CPU system clock (f_{VBCLK}). The only difference between f_{VBCLK} and f_{CPU} is that the latter can be stopped in HALT mode.

f_{VBCLK} is the clock supplied to the DMA, INTC, ROM, and RAM blocks. It is directly available at the CLKOUT pin.

(2) Peripheral clocks

The middle and right-hand side of *Figure 4-1 on page 180* shows how the clocks for the peripheral modules are generated and distributed.

Peripheral base clock f_{XP1} By use of the SELCNT4.ISEL40 bit either f_{XX} or f_{XMPLL} can be selected as the clock source for the peripheral base clock f_{XP1}. f_{XMPLL} is derived from the fixed PLL frequency f_{PLLO} ($f_{XMPLL} = f_{PLLO}$ or $f_{PLLO}/2$ by a code flash memory option byte). Therefore the unmodulated f_{PLLO} clock can be supplied to the peripherals even if a modulated f_{XX} clock derived from the SSCG is supplied to the CPU.

Prescaler1 General purpose peripheral clocks are provided by fixed Prescaler1.

This prescaler generates the peripheral clocks (f_{XP1} to $f_{XP1}/1024$) to be supplied to on-chip peripheral functions such as timers, serial interfaces and A/D Converter.

(3) Special clocks

The Clock Generator provides special clocks for certain peripherals.

Clock for UARTDn, TAAn This clock can be derived from f_{XP1} or f_{XP2}. Both f_{XP1} and f_{XP2} have the same frequency which is either f_{XX} or f_{XX} /2, depending on the setting of bit PRSI in the option bytes.

Note that f_{XP1} stops in all IDLE modes while f_{XP2} stops only in IDLE2 mode.

The timers TAA1 and TAA3 can also be supplied with the SubOSC clock f_{XT}.

Clock for TMM0 Clock source for timer TMM can be any of the oscillators. The selection between f_{XP1} or f_{RH} is made by bit SELCNT0.ISEL07.

Clock for CANn The CAN interfaces can be clocked by f_{XP1} or by f_{XC}, as chosen by a bit in the SELCNTx register. Select f_{XC} when directly supplying the clock generated by a clock oscillator to the CAN controller.

Clock for WT After reset, the Watch Timer is clocked by the SubOSC (f_{XT}). This can be changed when the main oscillator has stabilized. WT can then be clocked by the output of Prescaler3 that supplies also the CSIB0 block.

Prescaler3 serves as a baud rate generator. It is controlled by the registers PRSM0 and PRSCM0. For details see also *“Operation of Prescaler3“ on page 244*.

PCL The Clock Generator has a programmable clock (PCL) output.This output can deliver a fraction of f_{PLSS} (f_{PLSS} divided by 4, 8, 16, or 32), which is either f_{PLLO} or f_{SSCGO}. It is controlled by register PCLM and must be enabled by setting PCLM.PCLE.

CLKOUT This output provides the CPU system clock f_{VBCLK}. During the oscillation stability period, its state becomes Hi-Z.

Clock for WDT2 This is the clock for the Watchdog Timer. The clock for WDT2 is available (and hence the Watchdog Timer running) as long as the chosen clock source (240 KHz internal OSC or MainOSC) is active.

Note that the WDT2 operation is defined in option byte $007A_H$.

(4) Stand-by control

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

For an introduction, see *“Power save modes overview“ on page 185*.

(5) Summary of clock signals

f_{X}:	MainOSC clock is input clock to PLL and SSCG
f_{XT}:	SubOSC clock
f_{RL}:	240 KHz internal OSC clock
f_{RH}:	8 MHz internal OSC clock
f_{PLLI}:	PLL input clock. Can be f_{X} or a fraction of f_{X}
f_{PLLO}:	PLL output clock
f_{SSCGI}:	SSCG input clock. Can be f_{X} or a fraction of f_{X}
f_{SSCGO}:	SSCG output clock
f_{PLL}:	PLL or SSCG output clock
f_{XMPLL}:	f_{PLLO} or $f_{PLLO}/2$ for f_{XP1}
f_{XX}:	Main system clock
f_{VBCLK}:	CPU system clock
f_{CPU}:	CPU core clock (same clock as f_{VBCLK}, but stops in HALT mode)
f_{XP1}:	Peripheral clock 1 (output of Prescaler1, stops in IDLE1 mode)
f_{XP2}:	Peripheral clock 2 (same frequency as f_{XP1}, but continues in IDLE1 mode)
f_{XC}:	MainOSC clock for CANn interfaces (same frequency as f_{X}, stops in IDLE1 and IDLE2 mode)
f_{SC}:	Sub clock

4.1.2 Clock Monitor

The Clock Monitor supervises the operation of the MainOSC. In case of malfunction, the Clock Monitor can generate a system reset.

The monitor requires that the built-in 240 KHz internal oscillator is active. For details see *“Operation of the Clock Monitor“ on page 245*.

4.1.3 Power save modes overview

The power consumption of the system can be effectively reduced by using the stand-by modes and selecting the appropriate mode for the application. The available stand-by modes are listed below.

The following explanations provide a general overview. For details, please refer to *“Power save modes description“ on page 222* and the register descriptions.

HALT mode Mode in which only the operating clock of the CPU (f_{CPU}) is stopped. All other clocks remain active.

This mode is entered by executing the HALT instruction. All other power save modes are entered by setting registers.

This mode allows quick recovery to the normal operating mode, because it is not necessary to wait for oscillators to be stable or the PLL/SSCG to be locked.

IDLE1 mode Mode in which all the internal operations of the chip except the oscillators, PLL/SSCG, and flash memory are stopped. The PLL holds the previous operating status.
This mode allows quick return to the normal operating mode in response to a release signal, because it is not necessary to wait for oscillators to settle or the PLL/SSCG to lock.

IDLE2 mode Mode in which all the internal operations of the chip except the oscillators are stopped.

STOP Mode in which all the internal operations of the chip except the Sub oscillator are stopped.

Subclock operation Mode in which the subclock is used as the CPU system clock f_{VBCLK}. Subclock source can be the SubOSC (f_{XT}) or the 240 KHz internal OSC (f_{RL}). The selection is made by the SUBCLK bit of the option byte $007B_H$.

Sub-IDLE mode A mode that can be entered during subclock operation. All the internal operations of the chip except the oscillator, PLL/SSCG, and flash memory are stopped. The PLL holds the previous operating status.

4.1.4 Start conditions

After securing the setup time of the 8 MHz internal OSC, the CPU begins program execution. The oscillation stabilization time for the internal oscillator is ensured by hardware.

The table below shows the state during reset and after reset release.

Table 4-2 Oscillation during reset period or after reset release

Item	During the reset period	After releasing reset
MainOSC (f_X)	Stopped	
SubOSC (f_{XT})	Continues oscillation	
240 KHz internal-OSC (f_{RL})	Stopped	Starts oscillation
8 MHz internal-OSC (f_{RH})	Stopped	Starts oscillation
PLL (f_{PLLO})	Stopped	
SSCG (f_{SSCGO})	Stopped	
CPU system clock (f_{VBCLK})	Stopped	Starts operation on 8 MHz internal OSC f_{RH} after internal OSC is stable.
Peripheral clocks f_{XP1} (and fractions thereof), f_{XP2}	Stopped	Starts operation on 8 MHz internal OSC f_{RH} after internal OSC is stable.
Programmable clock output PCL (f_{PCL})	Disabled (low level)	
System clock output CLKOUT (f_{VBCLK})	Stopped	Output of 8 MHz internal oscillator f_{RH} after internal oscillator is stable. Must be enabled by software.

4.2 Clock Generator Registers

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

Table 4-3 Clock Generator register overview

Register name	Shortcut	Address	Write-protected by register
Power save control register	PSC	FFFF F1FE$_H$	PRCMD
Selector control register 0	SELCNT0	FFFF F308$_H$	
Selector control register 1	SELCNT1	FFFF F30A$_H$	
Selector control register 2	SELCNT2	FFFF F30C$_H$	
Selector control register 3	SELCNT3	FFFF F30E$_H$	
Selector control register 4	SELCNT4	FFFF F3F8$_H$	
Selector control register 5	SELCNT5	FFFF F3FA$_H$	
SSCG control register	SSCGCTL	FFFF F3F0$_H$	
SSCG frequency control register 0	SFC0	FFFF F3F1$_H$	
SSCG frequency control register 1	SFC1	FFFF F3F2$_H$	
Oscillation stabilization time select register	OSTS	FFFF F6C0$_H$	
PLL lockup time specification register	PLLS	FFFF F6C1$_H$	
Oscillation stabilization timer status register	OSTC	FFFF F6C2$_H$	
Internal oscillator mode register	RCM	FFFF F80C$_H$	
Power save mode control register	PSMR	FFFF F820$_H$	
PLL lock status register	LOCKR	FFFF F824$_H$	
Processor clock control register	PCC	FFFF F828$_H$	PRCMD
PLL control register	PLLCTL	FFFF F82C$_H$	
CPU operation clock status register	CCLS	FFFF F82E$_H$	
Programmable clock mode register	PCLM	FFFF F82F$_H$	
Main system clock mode register	MCM	FFFF F860$_H$	PRCMD
Clock Monitor mode register	CLM	FFFF F870$_H$	PRCMD
Prescaler3 mode register	PRSM0	FFFF F8B0$_H$	
Prescaler3 compare register	PRSCM0	FFFF F8B1$_H$	

Note
1. Some registers are write-protected to avoid inadvertent changes. Data can be written to these registers only in a special sequence of instructions, so that the register contents is not easily rewritten in case of a program hang-up.
   Writing to a protected register is only possible immediately after writing to the associated write protection register. For details please refer to *“CPU System Functions“ on page 155*.
2. In addition to the registers, control bits must be set in the code flash memory option bytes. For details see *“Option Bytes“ on page 215*.

The subsequent register descriptions are grouped as follows:

- **General clock generator registers:**
  - *“CCLS - CPU operation clock status register“ on page 189*
  - *“MCM - Main system clock mode register“ on page 190*
  - *“OSTC - Oscillation stabilization timer status register“ on page 191*
  - *“OSTS - Oscillation stabilization time select register“ on page 192*
  - *“PCC - Processor clock control register“ on page 194*
  - *“PCLM - Programmable clock mode register“ on page 197*
- **PLL control registers:**
  - *“LOCKR - PLL lock status register“ on page 199*
  - *“PLLCTL - PLL control register“ on page 200*
  - *“PLLS - PLL lockup time specification register“ on page 201*
- **SSCG control registers**
  - *“SSCGCTL - SSCG control register“ on page 202*
  - *“SFC0 - SSCG frequency control register 0“ on page 203*
  - *“SFC1 - SSCG frequency control register 1“ on page 204*
- **Stand-by control registers**
  - *“PSC - Power save control register“ on page 205*
  - *“PSMR - Power save mode control register“ on page 206*
- **Prescaler3 control registers:**
  - *“PRSM0 - Prescaler3 mode register“ on page 207*
  - *“PRSCM0 - Prescaler3 compare register“ on page 208*
- **Clock Monitor registers:**
  - *“CLM - Main oscillator Clock Monitor mode register“ on page 208*
- **Selector control registers:**
  - *“SELCNT0 - Selector control register 0“ on page 209*
  - *“SELCNT1 - Selector control register 1“ on page 210*
  - *“SELCNT2 - Selector control register 2“ on page 211*
  - *“SELCNT3 - Selector control register 3“ on page 212*
  - *“SELCNT4 - Selector control register 4“ on page 213*
  - *“SELCNT5 - Selector control register 5“ on page 214*

4.2.1 General Clock Generator registers

The general clock generator registers control and reflect the operation of the clock generator.

(1) CCLS - CPU operation clock status register

The CCLS register indicates the CPU operation clock status..

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

Address FFFF $F82E_H$.

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

	7	6	5	4	3	2	1	0
CCLS	0	0	0	0	0	0	0	CCLSF
	R	R	R	R	R	R	R	R

Table 4-4 CCLS register contents

Bit position	Bit name	Function
0	CCLSF	CPU operating clock status: 0: Operates on main system clock f_{XX} or subclock f_{SC}[a]. 1: Operates on 240 KHz internal oscillator f_{RL}.

[a] Subclock f_{SC} is either f_{XT} or f_{RL}, depending on SUBCLK bit of option byte $007B_H$.

Note If the Watchdog Timer WDT2 overflows before the oscillation stabilization time of the MainOSC has elapsed, this is judged as an abnormal oscillation of the MainOSC f_X. Thus the CPU system clock f_{VBCLK} is changed to internal oscillator f_{RL}.

(2) MCM - Main system clock mode register

The 8-bit MCM register specifies the main system clock (f_{XX}) source in clock-through mode and informs about its status.

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

Writing to this register is protected by a special sequence of instructions. Please refer to *“CPU System Functions“ on page 155* for details.

Address FFFF $F860_H$.

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

	7	6	5	4	3	2	1	0
MCM	0	0	0	0	0	0	MCS	MCM0
	R	R	R	R	R	R	R	R/W

Table 4-5 MCM register contents

Bit position	Bit name	Function
1	MCS	Status of the main system clock f_{XX} (in clock-through mode, if PLLCTL.SELPLL = 0): 0: Operating on 8 MHz internal oscillator clock f_{RH}. 1: Operating on MainOSC clock f_X.
0	MCM0	Clock selection of main system clock f_{XX}: 0: Clock source is the 8 MHz internal oscillator f_{RH} (in clock-through mode). 1: Clock source is – MainOSC f_X (in clock-through mode, if PLLCTL.SELPLL = 0) – PLL output f_{PLL} (in PLL mode, if PLLCTL.SELPLL = 1) **Caution:** 1. When the oscillation of a previous clock switch is not steady, rewriting of this bit is prohibited. 2. The MCM0 can be set to 0 only, if the current mode is clock-through, i.e. PLLCTL.SELPLL = 0. Do not change from PLL mode or subclock operation mode directly to 8 MHz internal oscillator clock-through mode or vice versa.

(3) OSTC - Oscillation stabilization timer status register

The 8-bit OSTC register indicates the status of the main oscillator.

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

Address FFFF F6C2$_H$.

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

	7	6	5	4	3	2	1	0
OSTC	0	0	0	0	0	0	0	MSTS
	R	R	R	R	R	R	R	R

Table 4-6 OSTC register contents

Bit position	Bit name	Function
0	MSTS	Oscillation stabilization status of MainOSC: 0: MainOSC stopped or waiting for oscillation stabilization. 1: MainOSC oscillation stabilization ended.

Remarks

1. The OSTC register does not monitor the main clock status but indicates the process status, based on the oscillation stabilization time specified by the OSTS register.
2. When the main clock oscillator is stopped by the software (PCC.MCK bit = 1) or entered into STOP mode, the OSTC register is set to 00$_H$. If it is stopped due to abnormal oscillation, the status is maintained.

(4) OSTS - Oscillation stabilization time select register

The 8-bit OSTS register specifies the oscillation stabilization time following reset release or release of the STOP mode.

The oscillation stabilization time and setup time are required when the STOP mode and IDLE mode are released, respectively.

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

Address FFFF $F6C0_H$.

Initial Value 06_H. The register is initialized by any reset.

	7	6	5	4	3	2	1	0
OSTS	0	0	0	OSTS4	OSTS3	OSTS2	OSTS1	OSTS0
	R	R	R	R/W	R/W	R/W	R/W	R/W

Table 4-7 OSTS register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th colspan="6">Function</th></tr>
<tr><td rowspan="24">4 to 0</td><td rowspan="24">OSTS[4:0]</td><td colspan="6">Selection of oscillation stabilization time and setup time:</td></tr>
<tr><th>OSTS4[a]</th><th>OSTS3</th><th>OSTS2</th><th>OSTS1</th><th>OSTS0</th><th>Oscillation stabilization time[b]</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>2^{10}/fx</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>2^{11}/fx</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>2^{12}/fx</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>2^{13}/fx</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>2^{14}/fx</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td><td>2^{15}/fx</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>2^{16}/fx</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>2^{17}/fx</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>2^{18}/fx</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>2^{19}/fx</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>2^{20}/fx</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td><td>2^{21}/fx</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</td><td>×</td><td>Setting prohibited</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>2^{4}/fx</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td><td>2^{5}/fx</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td><td>2^{6}/fx</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>2^{7}/fx</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>2^{8}/fx</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>1</td><td>1</td><td>2^{9}/fx</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>2^{10}/fx</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>0</td><td>1</td><td>2^{11}/fx</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>2^{12}/fx</td></tr>
<tr><td></td><td></td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td>2^{13}/fx</td></tr>
</table>

[a] Bit OSTS4 is only valid during IDLE2 mode release. In case of shifting to the STOP mode at OSTS4 bit = 1, the oscillation stabilization time after STOP mode release is the set period of the OSTS3-0 bits (OSTS4 bit is considered as 0).

[b] For minimum oscillation stabilization / setup times refer to the Datasheet.

Note

1. When IDLE2 mode is released, set the stabilization time to the following requirements:
   - In case of PLL mode: PLL lockup time requirements
   - In case of clock-through mode: flash set up time requirement

   For the exact timing values, refer to the Datasheet.
2. When STOP mode is released, set the stabilization time to the following requirements:
   - In case of PLL mode: PLL lockup time requirement
   - In case of clock-through mode:flash set up time requirement

   For the exact timing values, refer to the Datasheet.
3. If the required oscillation stabilization time of the MainOSC exceeds the above times, set the value to the required oscillation stabilization time of the MainOSC.

(5) PCC - Processor clock control register

The 8-bit PCC register controls the CPU system clock f_{VBCLK}.

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

Writing to this register is protected by a special sequence of instructions. Please refer to *"CPU System Functions" on page 155* for details.

Address FFFF $F828_H$.

Initial Value 40_H. The register is initialized by any reset.

	7	6	5	4	3	2	1	0
PCC	FRC	MCK	MFRC	CLS	CK3	CK2	CK1	CK0
	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W

Table 4-8 PCC register contents (1/2)

Bit position	Bit name	Function
7	FRC	Use of built-in Sub oscillator feedback resistor: 0: Feedback resistor connected. 1: Feedback resistor not connected.
6	MCK	Operation of MainOSC: 0: Oscillation enabled. 1: Oscillation stopped. **Note:** **1.** When the MCK bit is set to 1 while the system is operating with the main system clock as the CPU clock, the operation of the main system clock does not stop. It stops after the CPU clock has been changed to the subclock. **2.** When the main system clock is stopped and the device is operating on the subclock, clear the MCK bit to 0 and wait until the oscillation stabilization time has elapsed before switching back to the main system clock.
5	MFRC	Use of main oscillator on-chip feedback resistor: 0: Feedback resistor connected. 1: Feedback resistor not connected.
4	CLS	Status of CPU system clock f_{VBCLK}: 0: Main system clock f_{XX} operation. 1: Subclock f_{SC} operation.

Table 4-8 PCC register contents (2/2)

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>3 to 0</td><td>CK[3:0]</td><td>Clock selection:
<table>
<tr><th>CK3</th><th>CK2</th><th>CK1</th><th>CK0</th><th>Clock selection</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>f_{XX}</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>$f_{XX}/2$</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>$f_{XX}/4$</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>$f_{XX}/8$</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td><td>$f_{XX}/16$</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1</td><td>$f_{XX}/32$</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>×</td><td>Setting prohibited</td></tr>
<tr><td>1</td><td>×</td><td>×</td><td>×</td><td>Subclock f_{SC} (f_{XT} or f_{RL})[a]</td></tr>
</table>
Note: **1.** Do not change the CPU clock (by using the CK[3:0] bits) while CLKOUT is being output.
2. Use a bit manipulation instruction to manipulate the CK3 bit. When using an 8-bit manipulation instruction to change the CK3 bit, do not change the set values of the CK[2:0] bits simultaneously.</td></tr>
</table>

[a] Preset in option byte $007B_H$.

Examples:

main to subclock

1. Confirmation of operating clock: Confirm that the current clock is in main clock (MCS = 1). Switching from the high speed internal oscillator clock operation to low-speed internal oscillator clock operation is prohibited. In the high-speed internal oscillation clock operation (MCS = 0), set the MCM.MCM0 bit = 1 and then confirm that the MCM.MCM0 bit = 1 again.
2. Confirmation of CPU clock (f_{CPU}) frequency:
   Confirm that f_{CPU} satisfies either of the following conditions.
   • When OB7B.SUBLCK = 0, f_{CPU} > subclock oscillation frequency (f_{XT}) (32.768 kHz) × 4
   • When OB7B.SUBCLK = 1, f_{CPU} > low-speed internal oscillation clock frequency (f_{RL}) (TYP.240 kHz) × 4
   If the above conditions are not satisfied, change the CK2 to CK0 bits setting so as to satisfy the condition. At this time, do not change the CK3 bit.
3. Setting the CK3 bit to “1”: Set via bit manipulation instruction. Do not change the CK2-CK0 bits.
4. Subclock operation: The maximum time required for switching to subclock operation or to low-speed internal oscillation clock operation after the CK3 bit is set to 1, is as follows:
   • When OB7B.SUBCLK = 0: 1 / Subclock oscillation frequency (f_{XT})
   • When OB7B.SUBCLK = 1: 1 / low-speed internal oscillation frequency (f_{RL})
   Read the CLS bit and confirm that the operation has been switched to the subclock or low-speed internal oscillation operation.
5. Setting the MCK bit to "1": Set the MCK bit = 1 to stop the main oscillator operation.
   Caution: Stop PLL/SSCG before stopping the main oscillator operation. In addition, stop the operation of internal peripheral functions which operate at the main clock frequency.

subclock to main

1. Setting the MCK bit to "0": Enables main clock oscillation.
2. Software wait: Insert wait status via program to wait until the oscillation stabilization time of the main clock oscillator (OSTC.MSTS = 1) is elapsed.
3. Setting the CK3 bit to "0": Set via a bit manipulation instruction. Do not change the CK2 to CK0 bits.
4. Main clock operation: The maximum time required for switching to the main clock operation which is specified by the CK2 to CK0 bits after the CK3 bit is set, is as follows.
   - When OB7B.SUBCLK = 0: 1 / Subclock oscillation frequency (f_{XT})
   - When OB7B.SUBCLK = 1: 1 / low-speed internal oscillation frequency (f_{RL})

   Read the CLS bit and confirm that the operation has been switched to the main clock operation.

Caution Do not change to a different clock selection until the previous one has entered a stable status.

(6) PCLM - Programmable clock mode register

The 8-bit PCLM register specifies the setting the programmable clock output PCL.

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

Address FFFF $F82F_H$.

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

	7	6	5	4	3	2	1	0
PCLM	0	0	0	PCLE	0	0	PCK1	PCK0
	R	R	R	R/W	R	R	R/W	R/W

Table 4-9 PCLM register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>4</td><td>PCLE</td><td>PCL enable:
0: PCL disabled (PCL pin is fixed to low level).
1: PCL enabled.</td></tr>
<tr><td>1 to 0</td><td>PCK[1:0]</td><td>PCL clock frequency selection:
<table>
<tr><th>PCK1</th><th>PCK0</th><th>PCL output clock</th></tr>
<tr><td>0</td><td>0</td><td>$f_{PCL} = f_{PLLO}/4$</td></tr>
<tr><td>0</td><td>1</td><td>$f_{PCL} = f_{PLLO}/8$</td></tr>
<tr><td>1</td><td>0</td><td>$f_{PCL} = f_{PLLO}/16$</td></tr>
<tr><td>1</td><td>1</td><td>$f_{PCL} = f_{PLLO}/32$</td></tr>
</table></td></tr>
</table>

Note A PCL clock is only output when the PLL is in locked status.

(7) RCM - Internal oscillator mode register

The 8-bit RCM register specifies the operation and informs about the status of the low-speed and high-speed internal oscillators.

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

Address FFFF $F80C_H$.

Initial Value 80_H. The register is initialized by any reset.

	7	6	5	4	3	2	1	0
RCM	RSTS	0	0	0	0	0	HRSTOP	RSTOP
	R	R	R	R	R	R	R/W	R/W

Table 4-10 RCM register contents

Bit position	Bit name	Function
7	RSTS	Oscillation stability status of 8 MHz internal oscillator: 0: 8 MHz internal oscillator stopped or waiting for oscillation stability. 1: 8 MHz internal oscillator operating.
1	HRSTOP	Operation/stop of 8 MHz internal oscillator: 0: 8 MHz internal oscillator operating. 1: 8 MHz internal oscillator stopped. **Caution:** When the CPU clock source is the 8 MHz internal oscillator, do not set this bit to 1.
0	RSTOP	Operation/stop of 240 KHz internal oscillator: 0: 240 KHz internal oscillator operating. 1: 240 KHz internal oscillator stopped. **Note:** Setting this bit is ignored if bit RMOPIN of option byte $007A_H$ is set. **Caution:** When the CPU clock source is the 240 KHz internal oscillator, do not set this bit to 1.

4.2.2 PLL control registers

The PLL registers control and reflect the operation of the PLL.

(1) LOCKR - PLL lock status register

Phase lock occurs at a given frequency following power application or immediately after the STOP mode is released, and the time required for stabilization is the lockup time (frequency stabilization time). This time until stabilization is called the lockup status, and the stabilized state is called the locked status.

The lock register LOCKR includes a LOCK bit that reflects the PLL frequency stabilization status.

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

Address FFFF F824$_H$.

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

	7	6	5	4	3	2	1	0
LOCKR	0	0	0	0	0	0	0	LOCK
	R	R	R	R	R	R	R	R

Table 4-11 LOCKR register contents

Bit position	Bit name	Function
0	LOCK	PLL lock status check: 0: Locked status. 1: Unlocked status.

The LOCK register does not reflect the lock status of the PLL in real time. The set/reset conditions are as follows:

Set conditions

- Upon system reset. This register is set to 01$_H$ by reset and cleared to 00$_H$ after the reset has been released and the oscillation stabilization time has elapsed.
- In STOP and IDLE2 mode.
- Upon setting the PLL to stop (clearing bit PLLCTL.PLLON).
- Upon stopping the main system clock and using the CPU with subclock (setting bits PCC.CK3 and PCC.MCK to 1).

Clear conditions

- After reset release and overflow of oscillation stabilization time counter (OSTS register default time).
- When bit PLLCTL.PLLON is changed from 0 to 1 after PLL lockup timer overflow (time set by PLLS register).
- After STOP mode release and oscillation stabilization time counter overflow (time set by OSTS register), when the STOP mode was set while the PLL was in PLL mode.
- After IDLE2 mode release and oscillation stabilization timer overflow (time set by OSTS register), when the IDLE2 mode was set while the PLL was in PLL mode.

Note The PLL can enter the locked status only, if the MainOSC is enabled, i.e. PCC.MCLK = 0.

(2) PLLCTL - PLL control register

The 8-bit PLLCTL register controls the PLL function.

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

Address FFFF $F82C_H$.

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

	7	6	5	4	3	2	1	0
PLLCTL	0	0	0	0	0	0	SELPLL	PLLON
	R	R	R	R	R	R	R/W	R/W

Table 4-12 PLLCTL register contents

Bit position	Bit name	Function
1	SELPLL	Main system clock f_{XX} mode selection: 0: Clock-through mode (f_{XX} is MainOSC f_X or 8 MHz internal oscillator f_{RH} clock, depending on MCM.MCM0). 1: PLL mode (f_{XX} is PLL output f_{PLL}, if MCM.MCM0 = 1 as well).
0	PLLON	Control of PLL operation/stop: 0: PLL stopped. 1: PLL started. (After PLL operation starts, a lockup time is required for frequency stabilization).

Note

1. The SELPLL bit can be set to 1 only

   –if the PLL clock frequency has stabilized

   –and current mode is clock-through with MainOSC f_X as main system clock f_{XX}, i.e. MCM.MCM0 = 1

   If the PLL is unlocked or MCM.MCM0 = 0 (clock-through mode with internal oscillator f_{RH}), SELPLL can not be changed to 1. Thus you can not change from 8 MHz internl oscillator clock-through mode directly to PLL mode.

2. When the PLLON bit is cleared to 0, the SELPLL bit is automatically cleared to 0 (clock-through mode).

3. When the PLLON bit = 1 and the main clock is stopped, PLL stops the operation.

(3) PLLS - PLL lockup time specification register

The 8-bit PLLS register specifies the settling time of the PLL.

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

Address FFFF F6C1$_H$.

Initial Value 03$_H$. The register is initialized by any reset.

	7	6	5	4	3	2	1	0
PLLS	0	0	0	0	0	PLLS2	PLLS1	PLLS0
	R	R	R	R	R	R/W	R/W	R/W

Table 4-13 PLLS register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>2 to 0</td><td>PLLS[2:0]</td><td>PLL lockup time selection:
<table>
<tr><th>PLLS2</th><th>PLLS1</th><th>PLLS0</th><th>Lockup time</th></tr>
<tr><td>0</td><td>1</td><td>0</td><td>$2^{12}/f_X$</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>$2^{13}/f_X$ (default value)</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>$2^{14}/f_X$</td></tr>
</table></td></tr>
</table>

Note For the exact lockup time, refer to the Datasheet.

Caution Do not change the setting of the PPLS register during the PLL lock-up time.

4.2.3 SSCG control registers

This section describes the registers used for controlling the Spread Spectrum Clock Generator SSCG.

(1) SSCGCTL - SSCG control register

The 8-bit SSCGCTL register controls the SSCG operation and the source select the f_{PLL} clock.

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

Address FFFF $F3F0_H$.

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

	7	6	5	4	3	2	1	0
SSCG	0	0	0	0	0	0	SELSSCG	SSCGON
	R	R	R	R	R	R	R/W	R/W

Table 4-14 SSCGCTL register contents

Bit position	Bit name	Function
1	SELSSCG	PLL/SSCG output clock selection: 0: PLL output selected ($f_{PLL} = f_{PLLO}$ or $f_{PLLO}/2$) 1: SSCG output selected ($f_{PLL} = f_{SSCGO}$ or $f_{SSCGO}/2$)
0	SSCGON	SSCG enable/disable control 0: SSCG disabled/stopped 1: SSCG enabled

Note

1. The SSCG output clock can only be selected (SELSSCG = 1), if the SSCG is enabled (SSCGON = 1).
2. The clock selection is automatically set to PLL clock (SELSSCG = 0) if the SSCG is stopped (SSCGON = 0).
3. If the PLL is disabled (PLLCTL.PLLON bit = 0) or the MainOSC clock f_X stops, the SSCG stops operating.

SSCG start-up When the SSCG is started a SSCG lock-up time is needed: This can be assured by any of the following implementations:

1. Write the required PLL and SSCG lock-up time to the PLLS register and set the PLLCLTL.PLLON = 1 after setting SSCGCTL.SSCGON = 1.
2. Count the lock-up time of the SSCG by software after setting the SSCGON = 1 (PLL must be enabled before).
3. Set twice or more of the required PLL and SSCG lock-up time to the OSTS register.

Caution

1. The default value “0” of bit 7 of the SSCGCTL register must not be altered.
2. The SELSSCG bit must only be written if the PLL is disabled or locked.
3. Secure a setup time for at least 1 µs via software after the SFC0 and SFC1 registers are set and until the SSCGON bit is changed from 0 to 1.

(2) SFC0 - SSCG frequency control register 0

The 8-bit SFC0 register controls the frequency multiplication of the SSCG. It determines the SSCG output frequency f_{SSCGO}.

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

Writing to this register is protected by a special sequence of instructions. Please refer to *"CPU System Functions" on page 155* for details.

Address FFFF $F3F1_H$.

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

	7	6	5	4	3	2	1	0
SFC0	SFC07	SFC06	SFC05	SFC04	SFC03	SFC02	SFC01	SFC00
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Note This register can only be written when the SSCG enable bit SSCGCTL.SSCGON is cleared and the SSCG is safely switched off.
Please refer to *"SSCGCTL - SSCG control register" on page 202* for additional information.

Table 4-15 SFC0 register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>7</td><td>SFC07</td><td>SSCG input frequency divider selector:
0: $f_{SSCGI} = f_x$
1: $f_{SSCGI} = f_x/2$</td></tr>
<tr><td>6 to 4</td><td>SFC0[6:4]</td><td>SSCG input clock divider selector
<table>
<tr><th>SFC06</th><th>SFC05</th><th>SFC04</th><th>SSCG input clock f_{PFD}</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>f_{SSCGI}</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>$f_{SSCGI}/2$</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>$f_{SSCGI}/3$</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>$f_{SSCGI}/4$</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>$f_{SSCGI}/5$</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>$f_{SSCGI}/6$</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>$f_{SSCGI}/7$</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>$f_{SSCGI}/8$</td></tr>
</table></td></tr>
<tr><td>3 to 2</td><td>SFC0[3:2]</td><td>SSCG output clock divider selector
<table>
<tr><th>SFC03</th><th>SFC02</th><th>SSCG output clock divider</th></tr>
<tr><td>0</td><td>0</td><td>prohibited</td></tr>
<tr><td>0</td><td>1</td><td>Division by 2</td></tr>
<tr><td>1</td><td>0</td><td>Division by 4</td></tr>
<tr><td>1</td><td>1</td><td>prohibited</td></tr>
</table></td></tr>
<tr><td>1 to 0</td><td>SFC0[1:0]</td><td>SSCG input frequency range specification selector
<table>
<tr><th>SFC01</th><th>SFC00</th><th>SSCG frequency range specification</th></tr>
<tr><td>0</td><td>0</td><td>$0.87\text{MHz} \leq f_{PFD} < 1.00\text{MHz}$</td></tr>
<tr><td>0</td><td>1</td><td>$1.00\text{MHz} \leq f_{PFD} < 1.22\text{MHz}$</td></tr>
<tr><td>1</td><td>0</td><td>$1.22\text{MHz} \leq f_{PFD} < 1.45\text{MHz}$</td></tr>
<tr><td>1</td><td>1</td><td>$1.45\text{MHz} \leq f_{PFD} \leq 1.74\text{MHz}$</td></tr>
</table></td></tr>
</table>

(3) SFC1 - SSCG frequency control register 1

The 8-bit SFC1 register controls the frequency modulation of the SSCG in dithering mode.

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

Writing to this register is protected by a special sequence of instructions. Please refer to *“CPU System Functions“ on page 155* for details.

Address FFFF F3F2$_H$.

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

	7	6	5	4	3	2	1	0
SFC1	SFC17	SFC16	SFC15	SFC14	0	0	SFC11	SFC10
	R/W	R/W	R/W	R/W	R	R	R/W	R/W

Note This register can only be written when the SSCG enable bit SSCGCTL.SSCGON is cleared and the SSCG is safely switched off.
Please refer to *“SSCGCTL - SSCG control register“ on page 202* for additional information.

Table 4-16 SFC1 register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>7</td><td>SFC17</td><td>Frequency modulation enable control:
0: Modulation disabled
1: Modulation enabled</td></tr>
<tr><td>6 to 4</td><td>SFC1[6:4]</td><td>Frequency modulation range control:
<table>
<tr><th>SFC16</th><th>SFC15</th><th>SFC14</th><th>FM range</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>± 0.5 % (typical value)</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>± 1.0 % (typical value)</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>± 2.0 % (typical value)</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>± 3.0 % (typical value)</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>± 4.0 % (typical value)</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>± 5.0 % (typical value)</td></tr>
<tr><td colspan="3">other settings</td><td>prohibited</td></tr>
</table></td></tr>
<tr><td>1 to 0</td><td>SFC1[1:0]</td><td>Frequency modulation frequency control:
<table>
<tr><th>SFC11</th><th>SFC10</th><th>Modulation frequency</th></tr>
<tr><td>0</td><td>0</td><td>40 KHz (typical value)</td></tr>
<tr><td>0</td><td>1</td><td>50 KHz (typical value)</td></tr>
<tr><td>1</td><td>0</td><td>60 KHz (typical value)</td></tr>
<tr><td>1</td><td>1</td><td>prohibited</td></tr>
</table></td></tr>
</table>

Note The given modulation ranges and frequencies are typical values. Refer also to the related chapter in the Datasheet.

In dithering mode, the SSCG output frequency f_{SSCG} varies according to the FM range, specified by SFC1[6:4], around it’s center value:

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

The time of one full cycle is given by the period of the modulation frequency specified in SFC1[1:0].

4.2.4 Stand-by control registers

These registers control and reflect the various stand-by modes that can be entered for saving power.

(1) PSC - Power save control register

The 8-bit PSC register controls the stand-by function. The STP bit of this register specifies the stand-by mode.

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

Writing to this register is protected by a special sequence of instructions. Please refer to *“Write Protected Registers“ on page 176* for details.

Address FFFF F1FE$_H$.

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

	7	6	5	4	3	2	1	0
PSC	0	NMI1M	NMI0M	INTM	0	0	STP	0
	R	R/W	R/W	R/W	R	R	R/W	R

Table 4-17 PSC register contents

Bit position	Bit name	Function
6	NMI1M	Stand-by mode release control by occurrence of INTWDT2 signal: 0: Enable releasing stand-by mode by INTWDT2 signal. 1: Disable releasing stand-by mode by INTWDT2 signal.
5	NMI0M	Stand-by mode release control by NMI pin input: 0: Enable releasing stand-by mode by NMI pin input. 1: Disable releasing stand-by mode by NMI pin input.
4	INTM	Stand-by mode release control by maskable interrupt request signal: 0: Enable releasing stand-by mode by maskable interrupt request signal. 1: Disable releasing stand-by mode by maskable interrupt request signal.
1	STP	Setting of stand-by mode: 0: Normal mode. 1: Stand-by mode. **Note:** **1.** Stand-by modes that can be set by the STP bit: IDLE1 mode, IDLE2 mode, STOP mode, and sub-IDLE mode. **2.** Before setting this bit, set the bits PSMR.PSM[1:0].

When writing to this register, follow the instructions given in *“CPU System Functions“ on page 155*.

Entering a power save mode requires some attention, refer to “*“Power save mode activation“ on page 241*

Caution Entering a power save mode requires special attention, refer to *“Power save mode activation“ on page 241*.

(2) PSMR - Power save mode control register

The 8-bit PSMR register is used to specify one of the power save modes. The setting becomes effective when the mode is entered by setting PSC.STP to 1.

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

Address FFFF $F820_H$.

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

	7	6	5	4	3	2	1	0
PSMR	0	0	0	0	0	0	PSM1	PSM0
	R	R	R	R	R	R	R/W	R/W

Table 4-18 PSMR register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>1 to 0</td><td>PSM[1:0]</td><td>Specification of operation in software stand-by mode:
<table>
<tr><th>PSM1</th><th>PSM0</th><th>Power save mode</th></tr>
<tr><td>0</td><td>0</td><td>IDLE1 mode</td></tr>
<tr><td>0</td><td>1</td><td>STOP mode</td></tr>
<tr><td>1</td><td>0</td><td>IDLE2 mode or sub-IDLE mode[a]</td></tr>
<tr><td>1</td><td>1</td><td>STOP mode</td></tr>
</table>
Note: The PSM0 and PSM1 bits take effect after PSC.STP = 1.</td></tr>
</table>

a) Sub-IDLE mode is entered if the processor is in subclock mode (clocked by f_{XT} or f_{RL}).

For information on these modes, refer to *“Power save modes description“ on page 222*.

4.2.5 Prescaler3 control registers

These registers control the Prescaler3 that generates f_{BRG} which can be applied to the Watch Timer and the Clocked Serial Interface CSIB0. Prescaler3 includes a clock divider, a counter, and a comparator. For details see *"Operation of Prescaler3" on page 244*.

(1) PRSM0 - Prescaler3 mode register

The PRSM0 register controls the Prescaler3 operation.

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

Address FFFF $F8B0_H$.

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

	7	6	5	4	3	2	1	0
PRSM0	0	0	0	BGCE0	0	0	BGCS01	BGCS00
	R	R	R	R/W	R	R	R/W	R/W

Table 4-19 **PRSM0 register contents**

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>4</td><td>BGCE0</td><td>Prescaler3 output:
0: Disabled.
1: Enabled.</td></tr>
<tr><td>1 to 0</td><td>BGCS0[1:0]</td><td>Selection of counter clock:
<table>
<tr><th>BGCS01</th><th>BGCS00</th><th>Prescaler clock selection</th></tr>
<tr><td>0</td><td>0</td><td>f_X</td></tr>
<tr><td>0</td><td>1</td><td>f_X /2</td></tr>
<tr><td>1</td><td>0</td><td>f_X /4</td></tr>
<tr><td>1</td><td>1</td><td>f_X /8</td></tr>
</table></td></tr>
</table>

Note

1. Do not change the values of BGCS0[1:0] during Watch Timer operation.
2. Set the BGCS0[1:0] bits before setting the BGCE0 bit to 1.
3. Set the PRSM0 and PRSCM0 registers according to the main clock frequency that is used to obtain an f_{BRG} frequency of 32,768 KHz.

(2) PRSCM0 - Prescaler3 compare register

The PRSCM0 register specifies the compare value and hence the output frequency of f_{BRG}.

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

Address FFFF F8B1$_H$.

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

	7	6	5	4	3	2	1	0
PRSCM0	PRSCM7	PRSCM6	PRSCM5	PRSCM4	PRSCM3	PRSCM2	PRSCM1	PRSCM0
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Note

1. Do not rewrite the PRSCM0 register during Watch Timer operation.
2. Set the PRSM0 and PRSCM0 registers according to the main clock frequency that is used to obtain an f_{BRG} frequency of 32,768 KHz.

For details and a calculation example, please refer to *“Operation of Prescaler3“ on page 244*.

4.2.6 Clock Monitor control registers

These registers control and reflect the operation of the Clock Monitor.

(1) CLM - Main oscillator Clock Monitor mode register

The 8-bit CLM register is used to enable the monitor for the main oscillator clock.

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

Writing to this register is protected by a special sequence of instructions. Please refer to *“CPU System Functions“ on page 155* for details.

Address FFFF F870$_H$.

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

	7	6	5	4	3	2	1	0
CLM	0	0	0	0	0	0	0	CLME
	R	R	R	R	R	R	R	R/W

Table 4-20 CLM register contents

Bit position	Bit name	Function
0	CLME	Clock Monitor enable: 0: Clock Monitor for main oscillator disabled. 1: Clock Monitor for main oscillator enabled. This bit can only be cleared by reset.

Note

1. CLM.CLME can be set at any time. However, the Clock Monitor is only activated after the main oscillator has stabilized, indicated by OSTC.MSTS = 1.
2. When reset is generated by the clock monitor, CLM.CLME is cleared to 0 and RESF.CLMRF is set to 1.

4.2.7 Selector control registers

These registers are used to select the clocks and functions of timers TAAn, TMM0 and serial interfaces UARTDn, CANn.

Note In this section, only the bits that refer to clock generation and distribution are described. For further information please refer to the descriptions of the on-chip peripherals.

(1) SELCNT0 - Selector control register 0

The 8-bit SELCNT0 register is used to specify the clock for timer TMM0.

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

Address FFFF F308$_H$.

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

- V850ES/FE3
- V850ES/FF3

	7	6	5	4	3	2	1	0
SELCNT0	ISEL07	0	0	ISEL04	ISEL03	ISEL02	0	ISEL00
	R/W	R	R	R/W	R/W	R/W	R	R/W

- µPD70F3374, µPD70F3375 of V850ES/FG3
- µPD70F3378 of V850ES/FJ3

	7	6	5	4	3	2	1	0
SELCNT0	ISEL07	0	ISEL05	ISEL04	ISEL03	ISEL02	ISEL01	ISEL00
	R/W	R	R/W	R/W	R/W	R/W	R/W	R/W

- µPD70F3376A, µPD70F3377A of V850ES/FG3
- µPD70F3379, µPD70F3380, µPD70F3381, µPD70F3382 of V850ES/FJ3
- V850ES/FK3

	7	6	5	4	3	2	1	0
SELCNT0	ISEL07	ISEL06	ISEL05	ISEL04	ISEL03	ISEL02	ISEL01	ISEL00
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Note “R” bits marked with “0” must not be changed from their default value “0”.

Table 4-21 SELCNT0 register contents

Bit position	Bit name	Function
7	ISEL07	Selection of count clock for TMM0: 0: Clock = f_{XP1}/512. 1: Clock = f_{RH}/8.
6 to 0	ISEL0[6 :0]	Refer to TAAn chapter: *“SELCNT0 - Selector control register 0“ on page 408*

(2) SELCNT1 - Selector control register 1

The 8-bit SELCNT1 register is used to specify the clock for UARTD5 and CAN2, CAN3.

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

Address FFFF $F30A_H$.

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

- µPD70F3378, µPD70F3379, µPD70F3380, µPD70F3381, µPD70F3382 of V850ES/FJ3
- V850ES/FK3

	7	6	5	4	3	2	1	0
SELCNT1	0	0	ISEL15[a]	ISEL14[a]	ISEL13	ISEL12[a]	ISEL11[a]	ISEL10
	R	R	R/W	R/W	R/W	R/W	R/W	R/W

[a] Not available on µPD70F3378

Note "R" bits marked with "0" must not be changed from their default value "0".

Table 4-22 SELCNT1 register contents

Bit position	Bit name	Function
5	ISEL15	Selection of UARTD5 clock: 0: Clock = f_{XP1}. The clock that stops in the IDLE1 mode. 1: Clock = f_{XP2}. The clock that does not stop in the IDLE1 mode.
4	ISEL14	Selection of CAN3 clock: 0: Clock = f_{XP1}. 1: Clock = f_{XC}.
3	ISEL13	Selection of CAN2 clock: 0: Clock = f_{XP1}. 1: Clock = f_{XC}.
2 to 0	ISEL1[2:0]	Refer to TAAn chapter: *"SELCNT1 - Selector control register 1" on page 409*

(3) SELCNT2 - Selector control register 2

The 8-bit SELCNT2 register is used to specify the clock for UARTD0, UARTD1, CAN0 and TAAn.

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

Address FFFF F30C$_H$.

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

	7	6	5	4	3	2	1	0
SELCNT2	ISEL27	ISEL26	ISEL25	ISEL24	ISEL23	ISEL22	ISEL21	ISEL20
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 4-23 SELCNT2 register contents

Bit position	Bit name	Function
7	ISEL27	Selection of UARTD1 clock: 0: Clock = f_{XP1}. The clock that stops in the IDLE1 mode. 1: Clock = f_{XP2}. The clock that does not stop in the IDLE1 mode.
6	ISEL26	Selection of UARTD0 clock: 0: Clock = f_{XP1}. 1: Clock = f_{XP2}.
5	ISEL25	Selection of CAN0 clock: 0: Clock = f_{XP1}. 1: Clock = f_{XC}.
4	ISEL24	Selection of TAA4 counter clock: 0: Clock = f_{XP1}. 1: Clock = f_{XP2}.
3	ISEL23	Selection of TAA3 counter clock: 0: Clock = f_{XP1}. 1: Clock = f_{XP2}.
2	ISEL22	Selection of TAA2 counter clock: 0: Clock = f_{XP1}. 1: Clock = f_{XP2}.
1	ISEL21	Selection of TAA1 counter clock: 0: Clock = f_{XP1}. 1: Clock = f_{XP2}.
0	ISEL20	Selection of TAA0 counter clock: 0: Clock = f_{XP1}. 1: Clock = f_{XP2}.

(4) SELCNT3 - Selector control register 3

The 8-bit SELCNT3 register is used to specify the clocks for UARTD2 to UARTD4 and CAN1.

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

Address FFFF $F30E_H$.

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

- µPD70F3374, µPD70F3375 of V850ES/FG3
- µPD70F3378 of V850ES/FJ3

	7	6	5	4	3	2	1	0
SELCNT3	0	0	0	0	0	ISEL32	ISEL31	0
	R	R	R	R	R	R/W	R/W	R

- µPD70F3376A, µPD70F3377A of V850ES/FG3
- µPD70F3379, µPD70F3380, µPD70F3381, µPD70F3382 of V850ES/FJ3

	7	6	5	4	3	2	1	0
SELCNT3	0	0	0	ISEL34	ISEL33	ISEL32	ISEL31	ISEL30
	R	R	R	R/W	R/W	R/W	R/W	R/W

- V850ES/FK3

	7	6	5	4	3	2	1	0
SELCNT3	0	ISEL36	0	ISEL34	ISEL33	ISEL32	ISEL31	ISEL30
	R	R/W	R	R/W	R/W	R/W	R/W	R/W

Note “R” bits marked with “0” must not be changed from their default value “0”.

Table 4-24 SELCNT3 register contents

Bit position	Bit name	Function
6	ISEL36	Refer to timer TAAn chapter: *“SELCNT3 - Selector control register 3“ on page 410*
4	ISEL34	Selection of UARTD4 clock: 0: Clock = f_{XP1}. 1: Clock = f_{XP2}.
3	ISEL33	Selection of UARTD3 clock: 0: Clock = f_{XP1}. 1: Clock = f_{XP2}.
2	ISEL32	Selection of UARTD2 clock: 0: Clock = f_{XP1}. 1: Clock = f_{XP2}.
1	ISEL31	Selection of CAN1 clock: 0: Clock = f_{XP1}. 1: Clock = f_{XC}.
0	ISEL30	Refer to timer TAAn chapter: *“SELCNT3 - Selector control register 3“ on page 410*

(5) SELCNT4 - Selector control register 4

The 8-bit SELCNT4 register specifies the peripheral clocks.

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

Address FFFF F3F8$_H$.

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

	7	6	5	4	3	2	1	0
SELCNT4	0	0	0	0	0	0	0	ISEL40
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 4-25 SELCNT4 register contents

Bit position	Bit name	Function
0	ISEL40	Selection of the clock source for f_{XP1} clock: 0: $f_{XP1} = f_{XX}$ (8 MHz internal oscillator. MainOsc, PLL or SSCG). 1: $f_{XP1} = f_{XMPLL}$ (PLL).

Note

1. If the PLL is stopped (PLLCTL.PLLON = 0), ISEL40 can not be set to 1. Be sure to set ISEL40 to 0 before stopping the PLL.
2. When the SSCG is not used (SSCGON bit = 0), set the ISEL40 bit to 0.

(6) SELCNT5 - Selector control register 5

The 8-bit SELCNT5 register specifies the clocks for TAA5 to TAA7, UARTD6, UARTD7 and CAN4.

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

Address FFFF $F3FA_H$.

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

- V850ES/FK3

	7	6	5	4	3	2	1	0
SELCNT5	ISEL57	ISEL56	ISEL55	ISEL54	ISEL53	ISEL52	ISEL51	ISEL50
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 4-26 SELCNT5 register contents

Bit position	Bit name	Function
7 - 6	ISEL5[7:6]	Refer to timer TAAn chapter: *“SELCNT5 - Selector control register 5“ on page 411*
5	ISEL55	Selection of UARTD7 clock: 0: Clock = f_{XP1}. 1: Clock = f_{XP2}.
4	ISEL54	Selection of UARTD6 clock: 0: Clock = f_{XP1}. 1: Clock = f_{XP2}.
3	ISEL53	Selection of CAN4 clock: 0: Clock = f_{XP1}. 1: Clock = f_{XC}.
2	ISEL52	Selection of TAA7 counter clock: 0: Clock = f_{XP1}. 1: Clock = f_{XP2}.
1	ISEL51	Selection of TAA6 counter clock: 0: Clock = f_{XP1}. 1: Clock = f_{XP2}.
0	ISEL50	Selection of TAA5 counter clock: 0: Clock = f_{XP1}. 1: Clock = f_{XP2}.

4.3 Option Bytes

The code flash memory versions in this product series have an option data area where a block subject to mask options is specified. When writing a program to a code flash memory version, be sure to set the option data area corresponding to the following option bytes.

The option bytes are used for:

- Enable or disable stopping the 240 KHz internal oscillator by software
- Specifying the WDT2 operation mode
- Selection of SubOSC external connection (crystal or RC resonator)
- Selection of clock source in subclock operation mode (SubOSC or 240 KHz internal oscillator)
- Selection of PLL input clock
- Selection of PLL output clock
- Selection of peripheral clock

The option bytes are stored as 16-bit data at addresses 0000 007A$_H$ and 0000 007B$_H$ of the internal code flash memory.

Note In the following only the Clock generator related option bytes settings are described. For a complete overview refer to *“Flash Mask Options“ on page 330*.

4.3.1 Option byte 0000 007A$_H$

Address 0000 007A$_H$.

7	6	5	4	3	2	1	0
STOPXTAL	STOPRCZ	0	0	0	X	WDTMD1	RMOPIN

Note Bits marked with “0” must not be changed from their value “0”.

Table 4-27 Setting of option byte 0000 007A$_H$

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>7 to 6</td><td>STOPXTAL,
STOPRCZ</td><td>Selection of SubOSC mode:
<table>
<tr><th>STOPXTAL</th><th>STOPRCZ</th><th>Sub oscillator setting</th></tr>
<tr><td>0</td><td>0</td><td>Crystal oscillator mode (32,768 KHz)</td></tr>
<tr><td>1</td><td>1</td><td>RC oscillator mode (20 KHz)</td></tr>
<tr><td colspan="2">other than above</td><td>Setting prohibited</td></tr>
</table></td></tr>
<tr><td>1</td><td>WDTMD1</td><td>Specifies WDT2 operation mode:
0: <i>Count operation</i>: Can be stopped by WDM2.WDCS24.
<i>Input clock</i>: Selectable by WDTM2 register. 240 KHz internal oscillator or MainOSC.
<i>Operation mode</i>: Selectable by WDTM2 register. NMI interrupt (INTWDT2) or reset mode (WDT2RES) selectable.
1: <i>Count operation</i>: Cannot be stopped.
<i>Input clock</i>: Fixed to 240 KHz internal oscillator.
<i>Operation mode</i>: Fixed to reset mode (WDT2RES).</td></tr>
<tr><td>0</td><td>RMOPIN</td><td>Option that the 240 KHz internal oscillator can be stopped by software:
0: Can be stopped by software.
1: Cannot be stopped.</td></tr>
</table>

4.3.2 Option byte 0000 $007B_H$

Address 0000 $007B_H$.

7	6	5	4	3	2	1	0
SUBCLK	0	0	LATENCY	PLLO	PRSI	PLLI1	PLLI0

Note Bits marked with “0” must not be changed from their value “0”.

Table 4-28 Setting of option byte 0000 $007B_H$

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>7</td><td>SUBCLK</td><td>Clock source in subclock operating mode:
0: SubOSC selection.
1: 240 KHz internal oscillator selection.</td></tr>
<tr><td>4</td><td>LATENCY</td><td>refer to <i>“Flash Mask Options“ on page 330</i></td></tr>
<tr><td>3</td><td>PLLO</td><td>PLL output clock $f_{PLL}$and f_{XMPLL} selection:
<table>
<tr><th>SSCGCTL.SELSSCG</th><th>PLLO</th><th>f_{XMPLL}</th><th>f_{PLL}</th></tr>
<tr><td>0</td><td>0</td><td>f_{PLLO}</td><td>f_{PLLO}</td></tr>
<tr><td></td><td>1</td><td>$f_{PLLO}/2$</td><td>$f_{PLLO}/2$</td></tr>
<tr><td>1</td><td>0</td><td>f_{PLLO}</td><td>f_{SSCGO}</td></tr>
<tr><td></td><td>1</td><td>$f_{PLLO}/2$</td><td>$f_{SSCGO}/2$</td></tr>
</table></td></tr>
<tr><td>2</td><td>PRSI</td><td>Divider Setting for peripheral clocks f_{XP1} and f_{XP2}:
0: f_{XP1}, $f_{XP2} = f_{XX}$ (for $f_{XX} \leq 32$ MHz)
1: f_{XP1}, $f_{XP2} = f_{XX}$ /2 (for 32 MHz $< f_{XX} \leq 48$ MHz, can also be set if $f_{XX} \leq$ 32MHz)</td></tr>
<tr><td>1 to 0</td><td>PLLI[1:0]</td><td>PLL input clock frequency selection:
<table>
<tr><th>PLLI1</th><th>PLLI0</th><th>PLL input clock</th></tr>
<tr><td>0</td><td>0</td><td>$f_{PLLI} = f_X$</td></tr>
<tr><td>0</td><td>1</td><td>$f_{PLLI} = f_X/2$</td></tr>
<tr><td>1</td><td>x</td><td>$f_{PLLI} = f_X/4$</td></tr>
</table></td></tr>
</table>

4.4 Clock Generator Operation

This chapter describes the specific features of the Clock Generator. For details see:

- *“Overview of clock operation control settings“ on page 218*
- *“Operation state transitions“ on page 219*
- *“Power save modes description“ on page 222*
- *“Available clocks in power save modes“ on page 239*
- *“Controlling the PLL“ on page 243*
- *“Watch Dog Timer Clock“ on page 243*
- *“CLKOUT function“ on page 243*
- *“Operation of Prescaler3“ on page 244*
- *“Operation of the Clock Monitor“ on page 245*

4.4.1 Overview of clock operation control settings

The following table gives an overview of the settings that specify the CPU system clock f_{VBCLK}. It identifies the register bits that must be set or cleared to generate specific f_{VBCLK}.

Table 4-29 CPU system clock settings

CCLS. CCLSF	PCC.CLS	PLLCTL. SELPLL	SSCGCTL. SELSSCG	MCM. MCM0	Option byte 007B: SUBCLK bit	Operation Clock
0	0 (Main system clock operation mode)	0 (Clock-through mode)	-	0 (8 MHz internal oscillator mode)	x[a]	8 MHz internal oscillator clock operation
				1 (MainOSC mode)		MainOSC clock operation
		1 (PLL/SSCG mode)	0 (PLL)			PLL clock operation
			1 (SSCG)			SSCG clock operation
	1 (Subclock operation mode)	x	x		0 (SubOSC mode)	SubOSC clock operation
					1 (240 KHz internal oscillator mode 2)	240 KHz internal oscillator clock operation (Sub)
1	-					240 KHz internal oscillator clock operation (Security)
Other than above						Setting prohibited

[a] x = don’t care

4.4.2 Operation state transitions

The following figure illustrates the various state transitions.

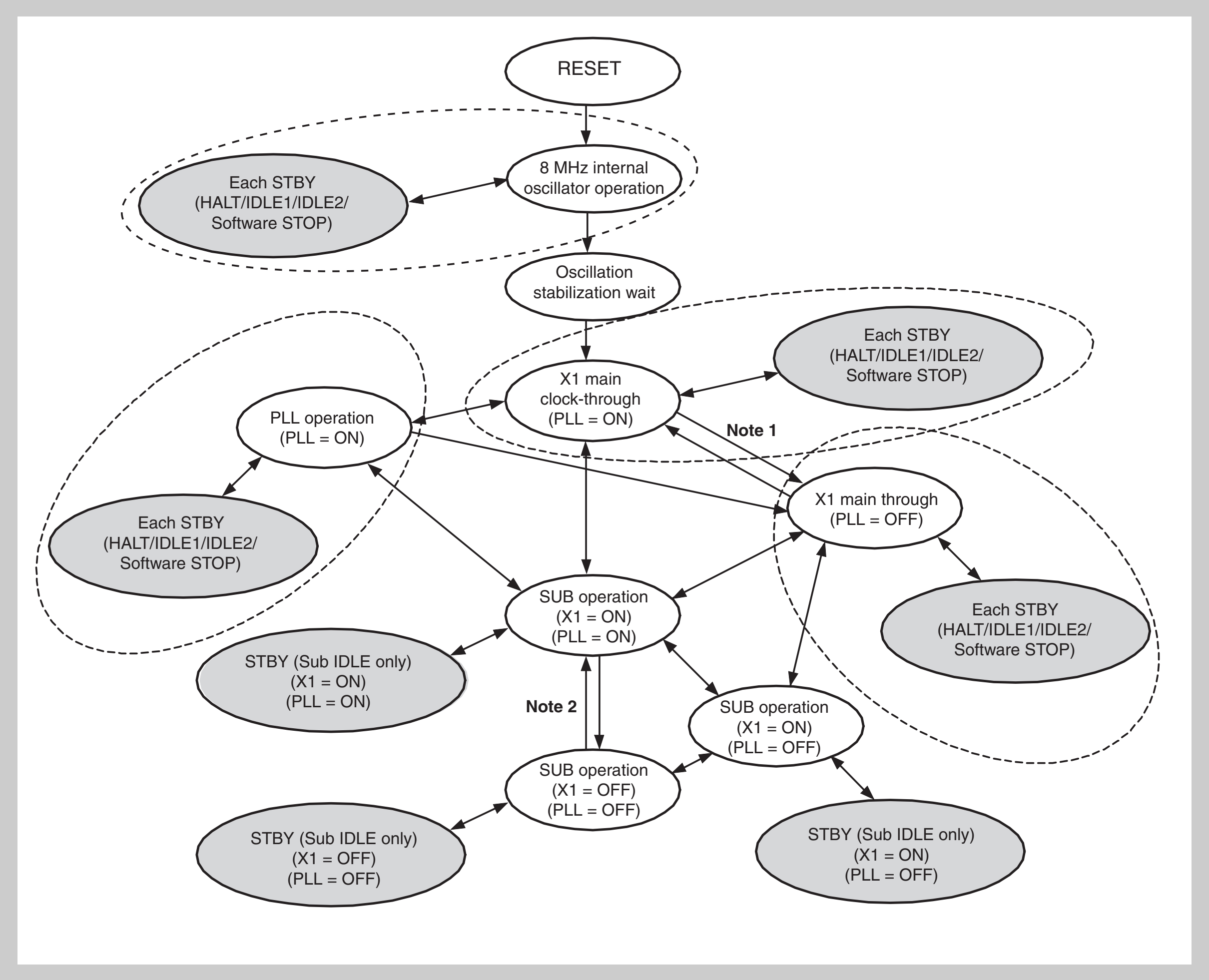

RESET
8 MHz internal oscillator operation
Each STBY (HALT/IDLE1/IDLE2/ Software STOP)
Oscillation stabilization wait
Each STBY (HALT/IDLE1/IDLE2/ Software STOP)
X1 main clock-through (PLL = ON)
PLL operation (PLL = ON)
Note 1
Each STBY (HALT/IDLE1/IDLE2/ Software STOP)
X1 main through (PLL = OFF)
SUB operation (X1 = ON) (PLL = ON)
Each STBY (HALT/IDLE1/IDLE2/ Software STOP)
STBY (Sub IDLE only) (X1 = ON) (PLL = ON)
Note 2
SUB operation (X1 = ON) (PLL = OFF)
SUB operation (X1 = OFF) (PLL = OFF)
STBY (Sub IDLE only) (X1 = OFF) (PLL = OFF)
STBY (Sub IDLE only) (X1 = ON) (PLL = OFF)

Figure 4-3 Operation state transition diagram

Note

1. When the PLL operation mode is entered, secure the lockup time by using software and check the PLL lock status by using the LOCKR.LOCK bit.
2. When changing the operation mode to the main clock oscillator mode, secure the oscillation stabilization time by using software and check the oscillation stabilization status by using the OSTC.OSTS bit. Enable the PLL operation before the main clock oscillator is enabled or after the oscillation is stabilized.

(1) Status transition from PLL operation

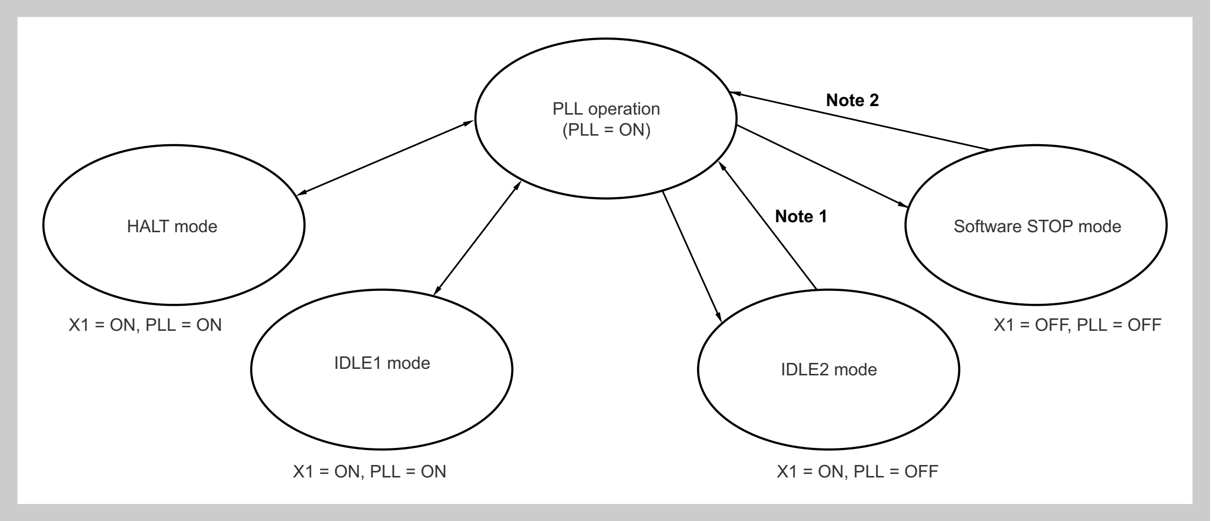


Figure 4-4 Stand-by transition from PLL operation (PLL = ON)

Note
1. After the time set by the OSTS register has elapsed, the CPU returns to the PLL mode.
2. After the time set by the OSTS register has elapsed, the CPU returns to the PLL mode. If the Watchdog Timer overflows (reset) while the oscillation stabilization time is being counted, the CPU starts clock operation with the internal oscillator.

(2) Status transition from main clock-through operation (with PLL on)

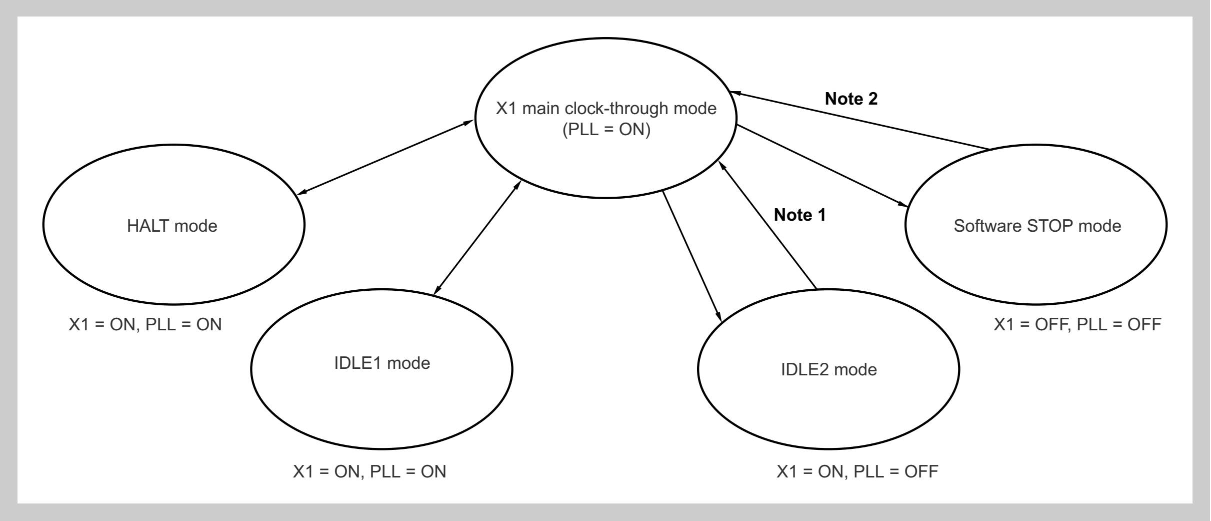


Figure 4-5 Stand-by transition from main clock-through operation (PLL = ON)

Note
1. After the time set by the OSTS register has elapsed, the CPU returns to the through mode.
2. After the time set by the OSTS register has elapsed, the CPU returns to the through mode. If the Watchdog Timer overflows (reset) while the oscillation stabilization time is counted, the CPU starts its clock operation with the internal oscillator.

(3) Status transition from clock-through operation (with PLL off)

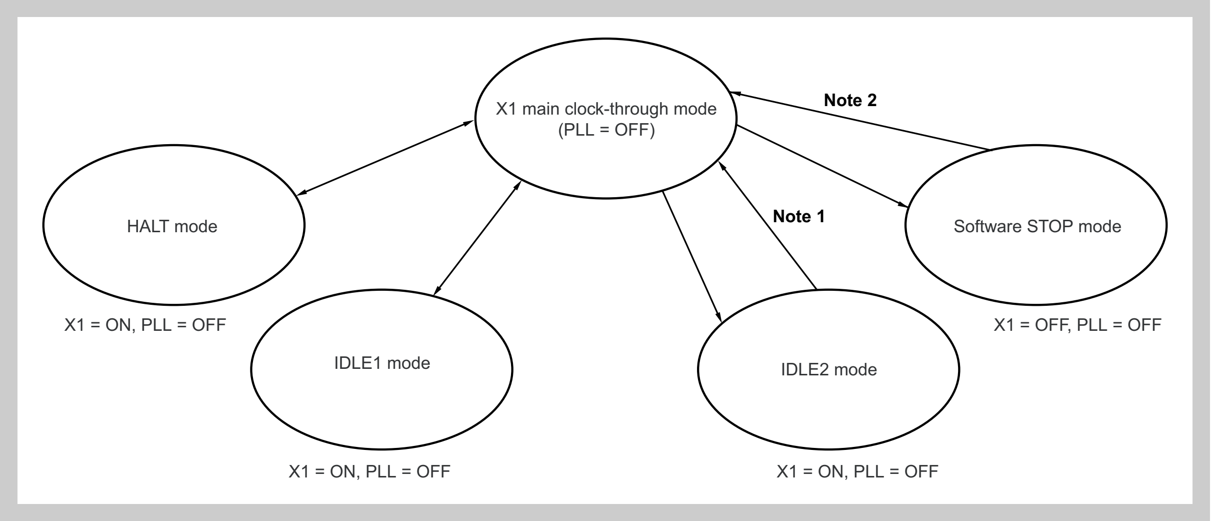


Figure 4-6 Stand-by transition from x1 main clock-through operation (PLL = OFF)

Note 1. After the time set by the OSTS register has elapsed, the CPU returns to the through mode.

2. After the time set by the OSTS register has elapsed, the CPU returns to the through mode. If the Watchdog Timer overflows (reset) while the oscillation stabilization time is counted, the CPU starts its clock operation with the internal oscillator.

(4) Status transition to / from subclock operation

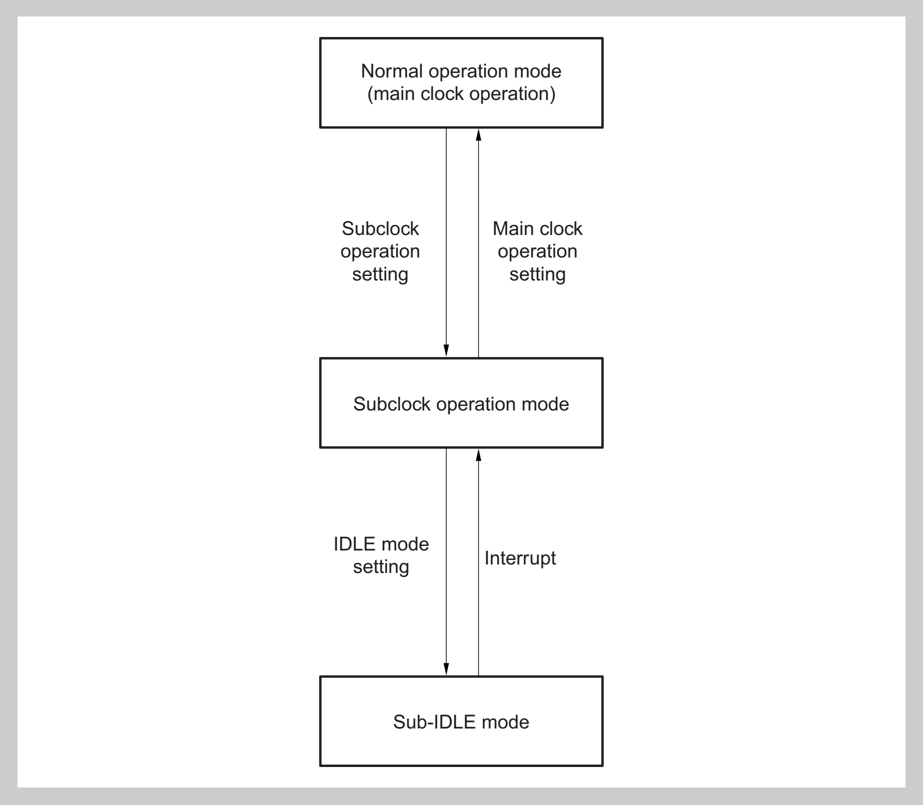


Figure 4-7 Status transition diagram (during subclock operation)

4.4.3 Power save modes description

This section explains the various power save modes in detail.

Table 4-30 Stand-by modes

Mode	Functional Outline
HALT mode	Mode in which only the operating clock of the CPU is stopped
IDLE1 mode	Mode in which all the internal operations of the chip except the oscillator, PLL/SSCG, and flash memory are stopped
IDLE2 mode	Mode in which all the internal operations of the chip except the oscillator are stopped
STOP mode	Mode in which all the internal operations of the chip except the subclock oscillator are stopped
Subclock operation mode	Mode in which the subclock is used as the CPU system clock
Sub-IDLE mode	Mode in which all the internal operations of the chip except the oscillator, PLL/SSCG, and flash memory are stopped, in the subclock operation mode

Caution Before entering any power save mode make sure that any access to the data flash is completed.

During power save mode

During all power save modes, the pins behave as follows:

- All output pins retain their function. That means all outputs are active, provided the required clock source is available.
- All input pins remain as input pins.
- All input pins with stand-by wake-up capability remain active, the function of all others is disabled.

During all power save modes, the main oscillator Clock Monitor remains active, provided that the oscillator is operating. If the oscillator is switched off during stand-by, the Clock Monitor enters stand-by as well.

Wake-up signals

The following signals can awake the controller from power save modes:

- Reset signals
  - external $\overline{\text{RESET}}$
  - Power-On-Clear reset RESPOC
  - Watchdog Timer reset RESWDT2
The Watchdog Timer must be configured to generate the reset in case of overflow and its input clock must be active during stand-by.
  - Clock Monitor reset RESCLM
The main oscillator must be active during stand-by.
- Non maskable interrupts
  - NMI0
The appropriate port must be configured correctly.
  - NMIWDT2
The Watchdog Timer must be configured to generate the interrupt in case of overflow and its input clock must be active during stand-by.
- Maskable interrupts
  - any unmasked maskable interrupt

Note that not all these signals are available in all power save modes.

Note In the following tables the clock status “operates” does not necessarily mean that the functions that use this clock source are operating as well.

(1) HALT mode

In this mode, the clock oscillators continue operating, but clock supply to the CPU is stopped. Clock supply to the other on-chip peripheral functions continues.

As a result, program execution is stopped, and the contents of the internal RAM before the HALT mode was set are retained. The on-chip peripheral functions that are not dependent upon the instruction processing of the CPU continue operating.

The HALT mode can reduce the average current consumption of the system if it is used with the normal operation mode for intermittent operation.

Entering HALT mode When the HALT instruction is executed in the normal operation mode, the HALT mode is set.
Insert five or more NOP instructions after the HALT instruction.

Note If the HALT instruction is executed while an interrupt request signal is held pending, the HALT mode is set but is released immediately by the pending interrupt request.

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

Table 4-31 Controller status in HALT mode (1/2)

		Working condition	
		Without Subclock	**With Subclock**
MainOSC (f_X)		Oscillation enabled	
SubOSC (f_{XT})		-	Oscillation enabled
240 KHz internal oscillator (f_{RL})		Oscillation enabled	
8 MHz internal oscillator (f_{RH})		Oscillation enabled	
PLL (f_{PLLO})		Operable	
SSCG (f_{SSCGO})		Operable	
CPU		Stops operation	
Port function		Holds status before HALT mode is set	
External bus interface		Refer to *“Bus and Memory Control (BCU, MEMC)“ on page 339*	
Timer/counter	TAA0 -TAA7	TAA0, 2, 4 and 6: Operable TAA1,3 ,5 and 7: Operable, when other than f_{XT} is selected as the count clock	Operable
	TAB0 -TAB2	Operable	
	TMM0	Operable, when other than f_{XT} is selected as the count clock	Operable
Watch Timer (WT)		Operable	
Watchdog Timer (WDT2)		Operable	
AD converter		Operable	
Serial Interface	UARTD0-7	Operable	
	CSIB0-3	Operable	
	IIC00	Operable	

Table 4-31 Controller status in HALT mode (2/2)

	Working condition	
	Without Subclock	With Subclock
CAN Controller (CAN0-4)	Operable	
DMA Controller	Operable	
Interrupt Controller	Operable	
Key interrupting function	Operable	
Clock Monitor	Operable	
Power-On-Clear circuit	Operable	
Low-Voltage Detector	Operable	
Voltage Regulator	Operation continues	
Internal data	The CPU registers, states, data and all other internal data such as the contents of the internal RAM are retained as they were before HALT mode was set	

Leaving HALT mode The HALT mode is released by a non-maskable interrupt request signal (NMI pin input or INTWDT2 signal), unmasked external interrupt request signal, unmasked internal interrupt request of a peripheral function that can operate in the HALT mode, or reset signal (reset by RESET pin input, WDT2RES signal, Low-Voltage Detector (LVI), or Clock Monitor (CLM)).

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

(a) Release by non-maskable interrupt request or unmasked maskable interrupt request

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 signal. If the HALT mode is set in an interrupt routine, however, the operation is performed as follows:

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

Table 4-32 Operation after HALT mode is released by interrupt request signal

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

(b) Releasing by $\overline{\text{RESET}}$ input

The operation is the same as the normal reset operation.

(2) IDLE1 mode

In the IDLE1 mode, the main oscillator, PLL/SSCG, and flash memory continue operating, but clock supply to the CPU and the other on-chip peripheral functions is stopped.

As a result, program execution is stopped, and the contents of the internal RAM before the IDLE1 mode was set are retained. The CPU and other on-chip peripheral functions stop operating. However, the on-chip peripheral functions that can operate on the subclock or external clock continue operating.

The IDLE1 mode can reduce current consumption more than the HALT mode because the operations of the on-chip peripheral functions are stopped. Because the main oscillator is not stopped, however, the normal mode can be restored without securing the oscillation stabilization time, in the same manner as in the HALT mode.

Entering IDLE1 mode

The IDLE1 mode is set when the PSM1 and PSM0 bits of the PSMR register are cleared to “00” and the STP bit of the PSC register is set to 1 in the normal operation mode.
Insert five or more NOP instructions after the store instruction that manipulates the PSC register to set the IDLE1 mode.

IDLE1 mode status

The following table shows the operation status in the IDLE1 mode.

Table 4-33 Controller status in IDLE1 mode (1/2)

		Working condition	
		Without Subclock	**With Subclock**
MainOSC (f_X)		Oscillation enabled	
SubOSC (f_{XT})		-	Oscillation enabled
240 KHz internal oscillator (f_{RL})		Oscillation enabled	
8 MHz internal oscillator (f_{RH})		Oscillation enabled	
PLL (f_{PLLO})		Operable	
SSCG (f_{SSCGO})		Operable	
CPU		Stops operation	
Port function		Holds status before IDLE1 mode is set	
External bus interface		Refer to *“Bus and Memory Control (BCU, MEMC)“ on page 339*	
Timer/counter	TAA0 -TAA7	Operable, if f_{XP2} is selected as the count clock	TAA0, 2, and 4: Operable, if f_{XP2} is selected as the count clock. TAA1 and 3: Operable, if f_{XP2} or f_{XT} is selected as the count clock[a]
	TAB0 -TAB2	Operation stops	
	TMM0	Operable, if $f_{RH}/8$, $f_{RL}/8$ or INTWT is selected as the count clock	Operable if $f_{RH}/8$, $f_{RL}/8$, INTWT or f_{XT} is selected as count clock.
Watch Timer (WT)		Operable, if clocked by Prescaler3	Operable
Watchdog Timer (WDT2)		Operable	
AD converter[b]		Stops operation	
Serial Interface	UARTD0-7	UARTD0: Operable if either f_{XP2} or ASCKD0 is selected input clock UARTD1-7: Operable if f_{XP2} is selected as operation clock.	
	CSIB0-3	Operable, if SCKBn is selected as input clock.	
	IIC00	Stops operation	
CAN Controller (CAN0-3)		Stops operation	

Table 4-33 Controller status in IDLE1 mode (2/2)

	Working condition	
	Without Subclock	With Subclock
DMA Controller	Stops operation	
Interrupt Controller	Stops operation (But it is possible to leave IDLE1 Mode)	
Key interrupting function	Operable	
Clock Monitor	Operable	
Power-On-Clear circuit	Operable	
Low-Voltage Detector	Operable	
Voltage Regulator	Operation continues	
Internal data	The CPU registers, states, data and all other internal data such as the contents of the internal RAM are retained as they were before IDLE1 mode was set	

[a] Only when setting the ISELxx bit =1 (f_{XP2}), the count operation by f_{XT} is also possible.
[b] To achieve low power consumption, stop the A/D Converter before shifting to the IDLE1 mode.

Leaving IDLE1 mode The IDLE1 mode is released by a non-maskable interrupt request signal (NMI pin input or INTWDT2 signal), unmasked external interrupt request signal, unmasked internal interrupt request signal of a peripheral function that can operate in the IDLE1 mode, or reset signal.

Note Interrupt request signals that are disabled by the NMI1M, NMI0M, and INTM bits of the PSC register are invalid and do not release the IDLE1 mode.

When digital noise elimination is enabled for INTP3, the power save mode cannot be released using INTP3 pin. For details, refer to *“Pin Functions“ on page 32*.

When the IDLE1 mode has been released, the normal operation mode is restored.

(a) Release by non-maskable interrupt request or unmasked maskable interrupt request

The IDLE1 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 signal. If the IDLE1 mode is set in an interrupt routine, however, the operation is performed as follows:

- If an interrupt request signal having a priority lower than that of the interrupt request currently being serviced is generated, the IDLE1 mode is released, but the interrupt request with the lower priority is not acknowledged. The interrupt request signal itself is held.
- If an interrupt request signal (including a non-maskable interrupt request signal) has a priority higher than that of the interrupt request currently being serviced is generated, the IDLE1 mode is released, and this interrupt request signal is acknowledged.

Table 4-34 Operation after IDLE1 mode is released by interrupt request signal

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

(b) Releasing by $\overline{\text{RESET}}$ input

The operation is the same as the normal reset operation.

(3) IDLE2 mode

In the IDLE2 mode, the main clock oscillator continues operating, but clock supply to the CPU, PLL/SSCG, flash memory, and the other on-chip peripheral functions is stopped.

As a result, program execution is stopped, and the contents of the internal RAM before the IDLE2 mode was set are retained. Not only the CPU but also the other on-chop peripheral functions stop operating. However, the on-chip peripheral functions that can operate on the subclock or external clock continue operating.

The IDLE2 mode can reduce current consumption more than the IDLE1 mode because the operations of the on-chip peripheral functions and flash memory are stopped. Because the PLL/SSCG and flash memory are stopped, however, setup times for the PLL/SSCG and flash memory must be maintained after the IDLE2 mode is released.

Entering IDLE2 mode The IDLE2 mode is set when the PSM1 and PSM0 bits of the PSMR register are set to “10” and the STP bit of the PSC register is set to 1 in the normal operation mode.

Note Insert five or more NOP instructions after the store instruction that manipulates the PSC register to set the IDLE2 mode.

IDLE2 mode status The following table shows the operation status in the IDLE2 mode.

Table 4-35 Controller status in IDLE2 mode (1/2)

		Working condition	
		Without Subclock	**With Subclock**
MainOSC (f_X)		Oscillation enabled	
SubOSC (f_{XT})		-	Oscillation enabled
240 KHz internal oscillator (f_{RL})		Oscillation enabled	
8 MHz internal oscillator (f_{RH})		Oscillation enabled	
PLL (f_{PLLO})		Stops operation	
SSCG (f_{SSCGO})		Stops operation	
CPU		Stops operation	
Port function		Holds status before IDLE2 mode is set	
External bus interface		Refer to *“Bus and Memory Control (BCU, MEMC)“ on page 339*	
Timer/counter	TAA0 -TAA7	Stops operation	
	TAB0 -TAB2	Stops operation	
	TMM0	Operable if f_{RH}/8, f_{RL}/8 or INTWT is selected as count clock.	Operable if f_{RH}/8, f_{RL}/8, INTWT or f_{XT} is selected as count clock.
Watch Timer (WT)		Operable, if clocked by Prescaler3	Operable
Watchdog Timer (WDT2)		Operable	
AD convertor[a]		Stops operation	
Serial Interface	UARTD0-7	UARTD0: Operable if ASCKD0 is selected as input clock UARTD1-7: Operation stops	
	CSIB0-3	Operable, if SCKBn is selected as input clock.	
	IIC00	Stops operation	
CAN Controller (CAN0-3)		Stops operation	
DMA Controller		Stops operation	

Table 4-35 **Controller status in IDLE2 mode (2/2)**

	Working condition	
	Without Subclock	With Subclock
Interrupt Controller	Stops operation (But it is possible to leave IDLE2 Mode)	
Key interrupting function	Operable	
Clock Monitor	Operable	
Power-On-Clear circuit	Operable	
Low-Voltage Detector	Operable	
Voltage Regulator	Operation continuous	
Internal data	The CPU registers, states, data and all other internal data such as the contents of the internal RAM are retained as they were before IDLE2 mode was set	

[a] To achieve low power consumption, stop the A/D Converter before shifting to the IDLE2 mode.

Leaving IDLE2 mode

The IDLE2 mode is released by a non-maskable interrupt request signal (NMI pin input or INTWDT2 signal), unmasked external interrupt request signal, unmasked internal interrupt request of a peripheral function that can operate in the IDLE2 mode, or reset signal.

When the IDLE2 mode has been released, the normal operation mode is restored.

Note

1. Interrupt request signals that are disabled by the NMI1M, NMI0M, and INTM bits of the PSC register are invalid and do not release the IDLE2 mode.
2. When digital noise elimination is enabled for INTP3, the power save mode cannot be released using INTP3 pin. For details, refer to *“Pin Functions“ on page 32*.

(a) Release by non-maskable interrupt request or unmasked maskable interrupt request

The IDLE2 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 signal. If the IDLE2 mode is set in an interrupt routine, however, the operation is performed as follows:

- If an interrupt request signal having a priority lower than that of the interrupt request currently being serviced is generated, the IDLE2 mode is released, but the interrupt request with the lower priority is not acknowledged. The interrupt request signal itself is held.
- If an interrupt request signal (including a non-maskable interrupt request signal) has a priority higher than that of the interrupt request currently being serviced is generated, the IDLE2 mode is released, and this interrupt request signal is acknowledged.

Table 4-36 **Operation after IDLE2 mode is released by interrupt request signal**

Releasing Source	Interrupt Enabled (EI) Status	Interrupt Disabled (DI) Status
Non-maskable interrupt request signal	Execution branches to the handler address after the specified setup time has elapsed.	
Maskable interrupt request signal	Execution branches to the handler address, or the next instruction is executed after the specified setup time has elapsed.	The next instruction is executed after the specified setup time has elapsed.

(b) Releasing by $\overline{\text{RESET}}$ input

The operation is the same as the normal reset operation.

(c) Securing setup time after release of IDLE2 mode

Secure the setup time of ROM (flash memory) after releasing the IDLE2 mode.

- Releasing by non-maskable interrupt request signal or unmasked maskable interrupt request signal:

  The setup time is secured by setting the OSTS register.
  When a source that releases the IDLE2 mode occurs, an internal dedicated timer starts counting in accordance with the setting of the OSTS register.
  When this counter overflows, the normal operation mode is restored.

- Releasing by reset input ($\overline{\text{RESET}}$ pin input or WDT2RES occurrence)

  The operation is the same as the normal reset operation.
  The oscillation stabilization time is the default value of the OSTS register, $2^{16} / f_X$.

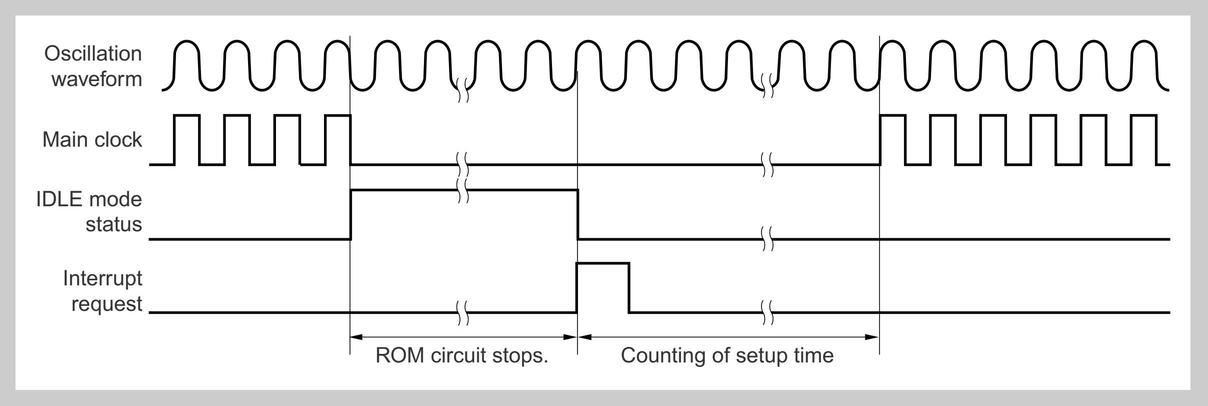


Figure 4-8 IDLE2 mode timing

(4) STOP mode

In the STOP mode, the subclock oscillator continues operating, but the main clock oscillator stops operating. Moreover, clock supply to the CPU and the other on-chip peripheral functions is stopped.

As a result, program execution is stopped, and the contents of the internal RAM before the STOP mode was set are retained. Not only the CPU but also the other on-chip peripheral functions stop operating. However, the on-chip peripheral functions that can operate on the subclock or external clock continue operating.

The STOP mode can reduce current consumption more than the IDLE2 mode because the operation of the main clock oscillator is stopped. When the subclock oscillator, internal oscillators and external clock are not used, the current consumption can be substantially reduced with only a leakage current flowing.

Entering STOP mode

The STOP mode is set when the PSM1 and PSM0 bits of the PSMR register are set to “01_B” or “11_B”, and the STP bit of the PSC register is set to 1 in the normal operation mode.

Insert five or more NOP instructions after the store instruction that manipulates the PSC register to set the STOP mode.

STOP mode status

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

Table 4-37 Controller status in STOP mode (1/2)

		Working condition	
		Without Subclock	With Subclock
MainOSC (f_X)		Stops operation	
SubOSC (f_{XT})		-	Oscillation enabled
240 KHz internal oscillator (f_{RL})		Oscillation enabled	
8 MHz internal oscillator (f_{RH})		Stops operation	
PLL (f_{PLLO})		Stops operation	
SSCG (f_{SSCGO})		Stops operation	
CPU		Stops operation	
Port function		Holds status before STOP mode is set	
External bus interface		Refer to *“Bus and Memory Control (BCU, MEMC)“ on page 339*	
Timer/counter	TAA0 -TAA7	Stops operation	
	TAB0 -TAB2	Stops operation	
	TMM0	Operable if f_{RL}/8 is selected as count clock.	Operable if f_{RL}/8, INTWT or f_{XT} is selected as count clock.
Watch Timer (WT)		Stops operation	Operable if f_{XT} is selected as count clock.
Watchdog Timer (WDT2)		Operable if f_{RL} is selected as count clock.	
AD convertor		Stops operation	
Serial Interface	UARTD0-7	UARTD0: Operable if ASCKD0 is selected input clock UARTD1-7: Operation stops.	
	CSIB0-3	Operable, if SCKBn is selected as input clock.	
	IIC00	Stops operation	
CAN Controller (CAN0-3)		Stops operation	

Table 4-37 Controller status in STOP mode (2/2)

	Working condition	
	Without Subclock	With Subclock
DMA Controller	Stops operation	
Interrupt Controller	Stops operation (But it is possible to leave STOP Mode)	
Key interrupting function	Operable	
Clock Monitor	Stops operation	
Power-On-Clear circuit	Operable	
Low-Voltage Detector	Operable	
Voltage Regulator	Operation continuous	
Internal data	The CPU registers, states, data and all other internal data such as the contents of the internal RAM are retained as they were before STOP mode was set	

Note 1. If the STOP mode is set while the A/D Converter is operating, the A/D Converter is automatically stopped and starts operating again after the STOP mode is released. However, in that case, the A/D conversion results up to the second conversion after the STOP mode is released are invalid (the third or later conversion results are valid). All the A/D conversion results before the STOP mode was set are invalid.

2. The power consumption in STOP mode is the same, no matter whether the A/D Converter was operating or stopped before the STOP mode was set.

Leaving STOP mode The STOP mode is released by a non-maskable interrupt request signal (NMI pin input or INTWDT2 signal), unmasked external interrupt request signal, unmasked internal interrupt request signal of a peripheral function that can operate in the STOP mode, or reset signal.

When the STOP mode has been released, the normal operation mode is restored.

Note 1. Interrupt request signals that are disabled by the NMI1M, NMI0M, and INTM bits of the PSC register are invalid and do not release the STOP mode.

2. When digital noise elimination is enabled for INTP3, the power save mode cannot be released using INTP3 pin. For details, refer to *“Pin Functions“ on page 32*.

(a) Release by non-maskable interrupt request or unmasked maskable interrupt request

The STOP 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 signal. If the STOP mode is set in an interrupt routine, however, the operation is performed as follows:

- If an interrupt request signal with a priority lower than that the interrupt request currently being serviced is generated, the STOP mode is released, but the interrupt request with the lower priority is not acknowledged. The interrupt request signal itself is held.
- If an interrupt request signal (including a non-maskable interrupt request signal) with a priority higher than that of the interrupt request currently being serviced is generated, the STOP mode is released, and this interrupt request signal is acknowledged.

Table 4-38 Operation after STOP mode is released by interrupt request signal

Releasing Source	Interrupt Enabled (EI) Status	Interrupt Disabled (DI) Status
Non-maskable interrupt request signal	Execution branches to the handler address after the specified oscillation stabilization time has elapsed.	
Maskable interrupt request signal	Execution branches to the handler address, or the next instruction is executed after the oscillation stabilization time has elapsed.	The next instruction is executed after the oscillation stabilization time has elapsed.

(b) Securing setup time after release of STOP mode

The main clock / 8MHz internal oscillator stop operating when the STOP mode is set. Therefore, secure the oscillation stabilization time of the clock oscillator(s) after releasing the STOP mode.

Releasing by non-maskable interrupt request signal or unmasked maskable interrupt request signal:

- The setup time is secured by setting the OSTS register.
- When a source that releases the STOP mode occurs, an internal dedicated timer starts counting in accordance with the setting of the OSTS register. When this counter overflows, the normal operation mode is restored.

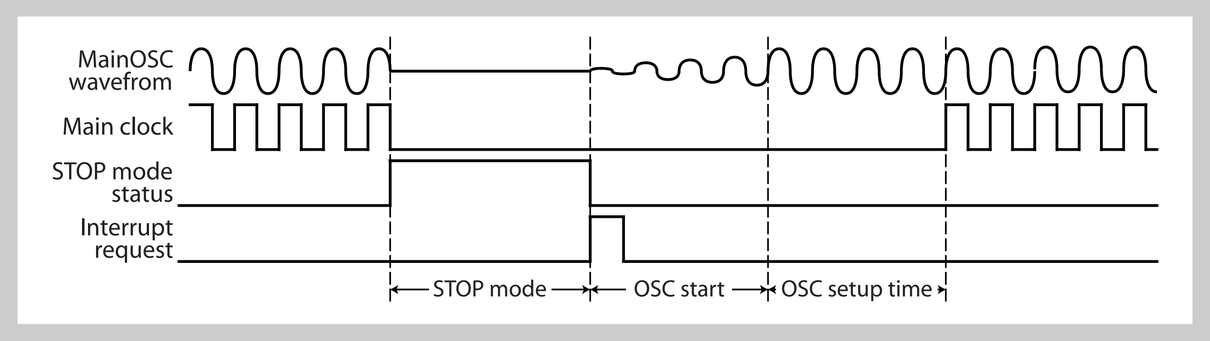


Figure 4-9 STOP mode timing for main clock operation

(c) Releasing by $\overline{\text{RESET}}$ input

The operation is the same as the normal reset operation.

(5) Subclock operation mode

When the subclock operation mode is set, the CPU system clock f_{VBCLK} is changed from the main system clock to the subclock. Subclock can be f_{XT} or f_{RL}. The selection is made by the SUBCLK bit of the option byte $007B_H$.

Check that the CPU system clock has been changed by using the CLS bit of the PCC register.

When the MCK bit of the PCC register is set to 1, the operation of the main clock oscillator is stopped. Consequently, the entire system operates on the subclock.

In the subclock operation mode, the subclock is used as the CPU system clock, so that the current consumption can be reduced from that in the normal operation mode. In addition, a current consumption close to that in the STOP mode can be achieved by stopping the operation of the main clock oscillator.

Entering subclock mode The subclock operation mode is set when the CK3 bit of the PCC register is set to 1 in the normal operation mode.

Note

1. Changing the value of the CK2 to CK0 bits of the PCC register is prohibited when the CK3 bit is manipulated (0 to 1 or 1 to 0). Set the CK3 bit by using a bit manipulation instruction. For details of the PCC register, refer to *"PCC - Processor clock control register" on page 194*.
2. If the following condition is not satisfied, change the CK2 to CK0 bits so as to satisfy the condition and move to subclock operation mode.
   Internal system clock (fCLK) > subclock (fSC) × 4

Subclock mode status The following table shows the operation status in subclock mode.

The table is shown for FJ3 ≥ 384 KB devices (maximum specification).

Table 4-39 Controller status in subclock mode (1/2)

		Working condition	
		With MainOSC operating	With MainOSC stopped
MainOSC (f_X)		Oscillation enabled	
SubOSC (f_{XT})		Oscillation enabled	
240 KHz internal oscillator (f_{RL})		Oscillation enabled	
8 MHz internal oscillator (f_{RH})		Oscillation enabled	
PLL (f_{PLLO})		Operable	Stops operation[a]
SSCG (f_{SSCGO})		Operable	Stops operation[a]
CPU		Operable	
Port function		Settable	
External bus interface		Refer to *"Bus and Memory Control (BCU, MEMC)" on page 339*	
Timer/counter	TAA0 -TAA7	Operable	Stops operation
	TAB0 -TAB2	Operable	Stops operation
	TMM0	Operable	Operable if f_{RH}/8, f_{RL}/8, INTWT or f_{XT} is selected as count clock.
Watch Timer (WT)		Operable	Operable if f_{XT} is selected as count clock.
Watchdog Timer (WDT2)		Operable	Operable if f_{RL} is selected as count clock.

Table 4-39 Controller status in subclock mode (2/2)

		Working condition	
		With MainOSC operating	With MainOSC stopped
AD convertor		Operable	Stops operation
Serial Interface	UARTD0-7	Operable	UARTD0: Operable if ASCKD0 is selected input clock UARTD1-7: Operation stops
	CSIB0-3	Operable	Operable if SCKBn input clock is selected as operation clock.
	IIC00	Operable	Stops operation
CAN Controller (CAN0-3)		Operable	Stops operation
DMA Controller		Operable	
Interrupt Controller		Operable	
Key interrupting function		Operable	
Clock Monitor		Operable	
Power-On-Clear circuit		Operable	
Low-Voltage Detector		Operable	
Voltage Regulator		Operation continuous	
Internal data		Settable[b]	

a) Set PLL to stop (PLLCTL.PLLON = 0) when you stop the main clock oscillation circuit.

b) The data from the data flash cannot be read. Refer to *“Flash Memory“ on page 298* for details.

Note

1. When stopping the main clock, be sure to stop the PLL (by clearing the PLLON bit of the PLLCTL register to 0).
2. When the CPU is operating on the subclock and main clock oscillation is stopped, accessing a register in which a wait occurs is disabled. If a wait is generated, it can be released only by $\overline{\text{RESET}}$.

Leaving subclock mode

The subclock operation mode is released by clearing the CK3 bit to 0 or by a reset signal.

Note

1. Changing the set value of the CK2 to CK0 bits of the PCC register is prohibited when the CK3 bit is manipulated (set the CK3 bit by using a bit manipulation instruction). For details of the PCC register, refer to *“PCC - Processor clock control register“ on page 194*.
2. When digital noise elimination is enabled for INTP3, the power save mode cannot be released using INTP3 pin. For details, refer to *“Pin Functions“ on page 32*.

When the main clock is stopped (PCC.MCK = 1), clear the MCK bit to 0, secure the oscillation stabilization time of the main clock by software, and then clear the CK3 bit to 0.

When the subclock operation mode is released, the normal operation mode is restored.

(6) Sub-IDLE mode

In the sub-IDLE mode, the clock oscillator continues operating, but clock supply to the CPU, flash memory, and the other on-chip peripheral functions is stopped.

As a result, program execution is stopped, and the contents of the internal RAM before the sub-IDLE mode was set is retained. Not only the CPU but also the other on-chip peripheral functions stop operating. However, the on-chip peripheral functions that can operate on the subclock continue operating.

The sub-IDLE mode can reduce current consumption more than the subclock operation mode because the operations of the CPU, flash memory, and other on-chip peripheral functions are stopped.

If the sub-IDLE mode is set after the main clock is stopped, a current consumption close to that in the STOP mode can be achieved.

Entering sub-IDLE mode The sub-IDLE mode is set when the PSM1 and PSM0 bits of the PSMR register are set to “10” and the STP bit of the PSC register is set to 1 while the processor is in the subclock operation mode.

Note Insert five or more NOP instructions after the store instruction that manipulates the PSC register to set the sub-IDLE mode.

Sub-IDLE mode status The following table shows the operation status in sub-IDLE mode.

Table 4-40 Controller status in sub-IDLE mode (1/2)

		Working condition	
		When main clock oscillator oscillates	When main clock oscillator stops
240 KHz internal oscillator (f_{RL})		Oscillation enabled	
8 MHz internal oscillator (f_{RH})		Oscillation enabled	
SubOSC (f_{XT})		Oscillation enabled	
PLL (f_{PLLO})		Operable	Stops operation[a]
SSCG (f_{SSCGO})		Operable	Stops operation[a]
CPU		Stops operation	
Port function		The settings of the previous mode are maintained	
External bus interface		Refer to *“Bus and Memory Control (BCU, MEMC)“ on page 339*	
Timer/counter	TAA0-TAA7	Stops operation	
	TAB0 -TAB2	Stops operation	
	TMM0	Operable if $f_{RH}/8$, $f_{RL}/8$ or f_{XT} is selected as count clock.	
Watch Timer (WT)		Operable	Operable if f_{XT} is selected as count clock.
Watchdog Timer (WDT2)		Operable	Operable if f_{RL} is selected as count clock.
AD convertor		Stops operation	
Serial Interface	UARTD0-7	UARTD0: Operable, if ASCKD0 is selected as input clock UARTD1-7: Operation stops	
	CSIB0-3	Operable if SCKBn input clock is selected as operation clock.	
	IIC00	Stops operation	
CAN Controller (CAN0-3)		Stops operation	

Table 4-40 Controller status in sub-IDLE mode (2/2)

	Working condition	
	When main clock oscillator oscillates	When main clock oscillator stops
DMA Controller	Stops operation	
Interrupt Controller	Stops operation (but it is possible to leave Sub Idle Mode)	
Key interrupting function	Operable	
Clock Monitor	Operable	
Power-On-Clear circuit	Operable	
Low-Voltage Detector	Operable	
Voltage Regulator	Operation continuous	
Internal data	The CPU registers, statuses, data and all other internal data such as the contents of the internal RAM are retained as they were before Sub IDLE mode was set	

[a] Stop the PLL (PLLCTL.PLLON = 0) when you stop the main clock oscillation circuit.

Leaving sub-IDLE mode The sub-IDLE mode is released by a non-maskable interrupt request signal (NMI pin input or INTWDT2 signal), unmasked external interrupt request signal, unmasked internal interrupt request of a peripheral function that can operate in the sub-IDLE mode, or reset signal.

The PLL returns to the operation status before the sub-IDLE mode was set.

When the sub-IDLE mode is released by an interrupt request signal, the subclock operation mode is restored. When the sub-IDLE mode is released by $\overline{\text{RESET}}$, the normal operation mode is restored.

Note

1. Interrupt request signals that are disabled by the NMI1M, NMI0M, and INTM bits of the PSC register are invalid and do not release the sub-IDLE mode.
2. When digital noise elimination is enabled for INTP3, the power save mode cannot be released using INTP3 pin. For details, refer to *“Pin Functions“ on page 32*.

(a) Release by non-maskable interrupt request or unmasked maskable interrupt request

The sub-IDLE mode is released by a non-maskable interrupt signal or an unmasked maskable interrupt request signal, regardless of the priority of the interrupt request signal.

If the sub-IDLE mode is set in an interrupt routine, however, the operation is performed as follows:

- Interrupt request signals that are set (disabled) by the NMI1M, NMI0M, and INTM bits of the PSC register are invalid and do not release the sub-IDLE mode.
- If an interrupt request signal having a priority lower than that of the interrupt request currently being serviced is generated, the sub-IDLE mode is released, but the interrupt request with the lower priority is not acknowledged. The interrupt request signal itself is held.
- If an interrupt request signal (including a non-maskable interrupt request signal) having a priority higher than that of the interrupt request currently being serviced is generated, the sub-IDLE mode is released, and this interrupt request signal is acknowledged.

Table 4-41 Operation after sub-IDLE mode is released by interrupt request signal

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

(b) Releasing by $\overline{\text{RESET}}$ input

The operation is the same as the normal reset operation.

4.4.4 Available clocks in power save modes

The following table gives an overview of the clock signals available in the various stand-by modes.

Table 4-42 Clock operation in power save modes

Operation status		f_X/f_{PLLI} Note2	f_{XT} Note2	f_{RL} Note2	f_{RH} Note2	f_{PLL} f_{SSCGO} f_{PCL}	f_{XX}	f_{XP1}	f_{VBCLK}	f_{CPU}	f_{XP2}	f_{XC}
Reset period		x	O	x	x	x	x	x	x	x	x	x
From reset release to 8 MHz internal oscillator setup		x	O	O	O	x	O	x	x	x	x	x
8 MHz internal oscillator Note1	Run	enable	O	enable	O	enable	O	O	O	O	O	enable
	HALT mode	enable	O	enable	O	enable	O	O	O	x	O	enable
	IDLE1 mode	enable	O	enable	O	enable	x	x	x	x	O	x
	STOP mode	x	O	enable	x	x	x	x	x	x	x	x
	From STOP release to oscillation stabilization	enable	O	enable	O	x	O	x	x	x	O	enable
MainOSC Note1	Run	O	O	enable	enable	enable	O	O	O	O	O	O
	HALT mode	O	O	enable	enable	enable	O	O	O	x	O	O
	IDLE1 mode	O	O	enable	enable	enable	x	x	x	x	O	x
	IDLE2 mode	O	O	enable	enable	x	x	x	x	x	x	x
	From IDLE2 release to setup	O	O	enable	enable	x	x	x	x	x	x	x
	STOP mode	x	O	enable	x	x	x	x	x	x	x	x
	From STOP release to oscillation stabilization	O	O	enable	enable	x	x	x	x	x	x	x
PLL/SSCG Note1	Run	O	O	enable	enable	O	O	O	O	O	O	O
	HALT mode	O	O	enable	enable	O	O	O	O	x	O	O
	IDLE1 mode	O	O	enable	enable	O	x	x	x	x	O	x
	IDLE2 mode	O	O	enable	enable	x	x	x	x	x	x	x
	From IDLE2 release to setup	O	O	enable	enable	x	x	x	x	x	x	x
	STOP mode	x	O	enable	x	x	x	x	x	x	x	x
	From STOP release to oscillation stabilization	O	O	enable	enable	x	x	x	x	x	x	x
SubOSC Note1	Run	enable	O	enable	enable	enable	enable	enable	O	O	enable	enable
	IDLE mode	enable	O	enable	enable	enable	x	x	x	x	x	x
240 KHz internal oscillator-Sub Note1	Run	enable	O	O	enable	enable	enable	enable	O	O	enable	enable
	IDLE mode	enable	O	O	enable	enable	x	x	x	x	x	x
240 KHz internal oscillator-Security Note1	Run	-	O	O	enable	-	enable	enable	O	O	enable	enable
	HALT mode	-	O	O	enable	-	enable	enable	O	x	enable	enable

O: Operating x: Stopped
Enable: Operation enable (by control register and option bytes setting)

Note **1.** The working conditions are the following:

- 8 MHz internal oscillator:	8 MHz internal oscillator clock operation
- MainOSC:	MainOSC clock operation
- PLL/SSCG:	PLL/SSCG clock operation
- SubOSC:	SubOSC clock operation
- 240 KHz internal oscillator-Sub:	240 KHz internal oscillator clock operation for Sub
- 240 KHz internal oscillator-Security:	240 KHz internal oscillator clock operation for Security

2. The clock signals are:

f_{X}:	MainOSC clock
f_{XT}:	SubOSC clock
f_{RL}:	240 KHz internal oscillator clock
f_{RH}:	8 MHz internal oscillator clock
f_{PLL}:	PLL/SSCG output clock
f_{PCL}:	Programmable clock output
f_{XX}:	Main system clock
f_{VBCLK}:	CPU system clock
f_{CPU}:	CPU core clock
f_{XP1}:	Peripheral clock (Prescaler1)
f_{XP2}:	Clock for UARTD, TAA
f_{XC}:	Clock for CAN

4.4.5 Power save mode activation

In the following procedures are described how to securely entering a power save mode.

Caution Before entering any power save mode make sure that any access to the data flash is completed.

(1) HALT mode

For entering the HALT mode proceed as follows:

1. Mask all interrupts which shall not have wake-up capability by xxIC.xxMK = 0 and discard all possibly pending interrupts by xxIC.xxIF = 0.
2. Unmask all interrupts which shall have wake-up capability by xxIC.xxMK = 1.
3. Execute the “halt” instruction.

(2) IDLE1, IDLE2 and STOP mode

For entering these power save mode proceed as follows:

1. In case maskable interrupts shall be used for wake-up unmask these interrupts by IMRm.xxMK = 0 (refer to *“IMRm - Interrupt mask registers“ on page 275*).
2. Mask all other interrupts, i.e.
   – none wake-up capable interrupts
   – wake-up capable interrupts which shall not be used for wake-up

   by IMRm.xxMK = 1. This prevents the power save mode entry procedure from being interrupted by these interrupts.
3. It is recommended to disable interrupt acknowledgement by the “di” instruction.
4. Specify the desired power save mode in PSM.PSM[1:0].
5. Enable writing to the write-protected register PSC by writing to PRCMD.
6. Write to PSC for specifying permitted wake-up events and activate the power save mode by setting PSC.STP to 1.

Example The following example shows how to initialize and enter a IDLE1, IDLE2 or STOP power save mode.

First the desired power save mode is specified (IDLE2 mode in this example, that means PSMR.PSM[1:0] = 10_B).

The PSC register is a write-protected register, and the PRCMD register is the corresponding write-enable register. PRCMD has to be written immediately before writing to PSC.

In this example, maskable interrupts are permitted to leave the power save mode.

```
1.    // xxIC.xxMK = 0                // mask all none wake-up interrupts
2.    // xxIC.xxMK = 1                // unmask all wake-up interrupts
3.    di
4.    mov     0x02,r10
5.    st.b    10,PSMR[r0]             // PSMR.PSM[1:0] = 10B: IDLE2 mode
6.    mov     0x62,r10
7.    st.b    r10,PRCMD[r0]           // enable write to PSC
8.    st.b    r10,PSC[r0]             // wake up by maskable interrupts
                                      // and enter power save mode
9.    nop
10.   nop
11.   nop
12.   nop
13.   nop
14.                                   // after wake-up
15.   // xxIC.xxIF = 0                // discard all unwanted pending interrupts
16.   ei
```

Be aware of the following notes when entering power save mode using the above sequence:

Note

1. It is recommended to disable maskable interrupt acknowledgement in general by the “di” instruction (step 3.) to prevent any pending interrupt from being served during the power save mode set-up procedure. This makes it also possible to completely control the process after wake-up, since no pending interrupt will be unintentional acknowledged. Before enabling interrupt acknowledgement by the “ei” instruction (step 16.) after wake-up, all unwanted interrupts can be discarded by setting xxIC.xxIF = 0 (step 15.).
Since the wake-up capability of the unmasked wake-up interrupts is not affected by “di”, such interrupts shall be masked (step 1.) by IMRm.xxMK = 1.

2. The store instruction to PRCMD will not allow to acknowledge any interrupt until processing of the subsequent instruction is complete. That means, an interrupt will not be acknowledged before the store to PSC. This presupposes that both store instructions are performed consecutively, as shown in the above example.
If another instruction is placed between steps 7 and 8, an interrupt request may be acknowledged in between, and the power save mode may not be entered.
However if the “di” instruction was executed before (step 3.) none interrupt will be acknowledged anyway.

3. At least 5 “nop” instructions must follow the power down mode setting, that means after the write to PSC. The microcontroller requires this time to enter power down mode.

4. Any data can be written to the PRCMD register.
In the example the same data is written, minimizing the number of used registers.

5. Make sure that all DMA channels are disabled. Otherwise a DMA could happen between steps 7 and 8, and the power down mode may not be entered at all.
   Further on do not perform write operations to PRCMD and write-protected registers by DMA transfers.
6. No special sequence is required for reading the PSC register.

4.4.6 Controlling the PLL

Using the PLL After the $\overline{\text{RESET}}$ signal has been released, the PLL has to be started by PLLCTL.PLLON = 1, after the main oscillator has stabilized (OSTC.MSTS = 1).
Since the default mode is the clock-through mode (PLLCTL.SELPLL = 0), select the PLL mode (PLLCTL.SELPLL = 1).

- To operate the PLL from the stopped status, set PLLCTL.PLLON = 1, and then set PLLCTL.SELPLL = 1 after the LOCKR.LOCK bit = 0 (the lockup time can be counted by setting the lockup time to the PLLS register and monitoring the LOCK flag of the LOCKR register).
- To stop the PLL, first select the clock-through mode (PLLCTL.SELPLL = 0), wait for 8 clocks or more, and then stop the PLL (set PLLCTL.PLLON = 0).

When shifting to the IDLE2 or STOP mode while remaining in the PLL operation mode, set the OSTS register as follows:

- STOP mode: Oscillation stabilization time > PLL lockup time
- IDLE2 mode: Setup time > PLL lockup time

When shifting to the IDLE1 mode, the PLL does not stop. Stop the PLL if necessary.

Not using the PLL The clock-through mode (PLLCTL.SELPLL = 0) is selected after the $\overline{\text{RESET}}$ signal has been released. The PLL is stopped by default.

4.4.7 Watch Dog Timer Clock

After reset release, the Watchdog Timer WDT2 is operating on the 240 KHz internal oscillator (f_{RL}/8 = 30 KHz approx.).

When the MainOSC has stabilized, the Watchdog Timer can be clocked by the MainOSC (f_X/128).

4.4.8 CLKOUT function

The clock output function is used to output the CPU system clock (f_{VBCLK}) from the CLKOUT pin.

The status of the CLKOUT pin is the same as the CPU system clock. The pin can output the clock when it is in the operable status. It outputs a low level in the stopped status.

4.4.9 Operation of Prescaler3

Prescaler3 generates the clock f_{BRG} by dividing the main oscillator output signal f_X.

(1) Description

Prescaler3 consists of a clock divider, a counter, and a comparator.

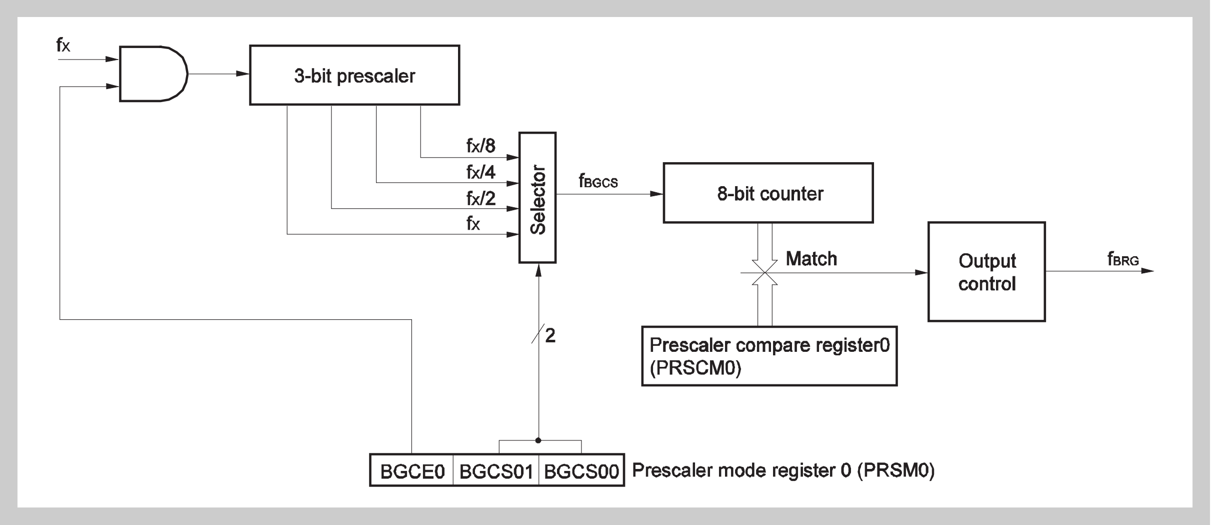


Figure 4-10 Prescaler3 Block Diagram

(2) Calculation

The relation between the main oscillator clock (f_X), prescaler clock divider selection PRSM0.BGCS0[1:2], PRSCM0 compare register value, and output clock f_{BRG} is as follows:

$$f_{BRG} = f_X / (2^m \times N \times 2)$$

where

f_{BRG} = output clock frequency

f_X = input clock frequency

m = BGCS0[1:0] value (0 to 3)

N = PRSCM0 register value (1 to FF_H). If PRSCM0 = 00_H: N = 256

Example If

f_X = 4 MHz
m = 0
N = $3D_H$

then

f_{BRG} = 32,787 KHz

4.4.10 Operation of the Clock Monitor

The Clock Monitor samples the main clock by using the internal 240 KHz internal oscillator. It generates a reset request signal when the oscillation of the main clock has stopped.

(1) Description

The functional block diagram is shown below.

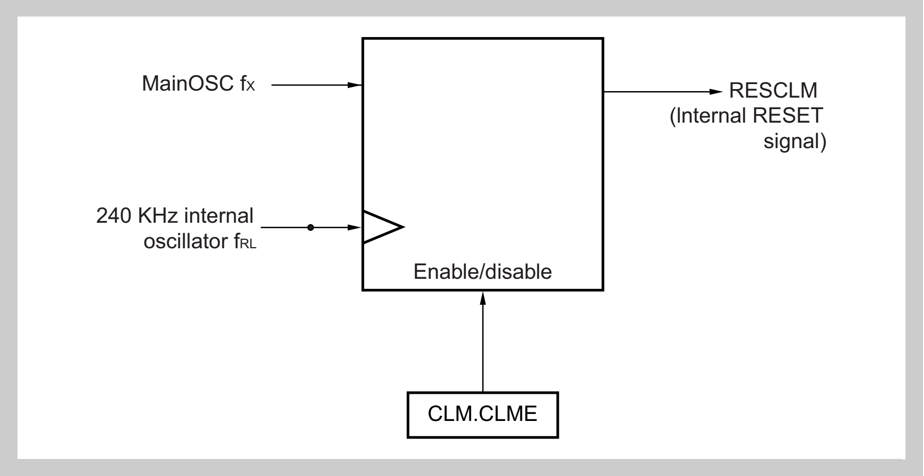

MainOSC fX
240 KHz internal oscillator fRL
Enable/disable
RESCLM (Internal RESET signal)
CLM.CLME

Figure 4-11 Clock Monitor Block Diagram

The Clock Monitor samples the main oscillator signal f_X. The Clock Monitor is clocked by the internal 240 KHz internal oscillator (f_{RL}).

The RESCLM reset signal is generated when the MainOSC clock fails.

Table 4-43 Operation status of Clock Monitor (when CLM.CLME Bit = 1, during internal oscillator operation)

CPU system clock f_{VBCLK}	Operation mode	Status of MainOSC	Status of internal oscillator Clock	Status of Clock Monitor
Main clock	HALT mode	Oscillates	Oscillates[a]	Operates[b]
	IDLE1 mode, IDLE2 mode	Oscillates	Oscillates[a]	Operates[b]
	STOP mode	Stops	Oscillates[a]	Stops
Subclock (PCC.MCK = 0)	Sub-IDLE mode	Oscillates	Oscillates[a]	Operates[b]
Subclock (PCC.MCK = 1)	Sub-IDLE mode	Stops	Oscillates[a]	Stops
Internal oscillator clock	–	Stops	Stops[a]	Stops
During reset	–	Stops	Stops	Stops

[a] Internal oscillator can be stopped by setting the RSTOP bit of the RCM register to 1 only when “Internal oscillator can be stopped” is specified by an option function.

[b] The Clock Monitor is stopped when the internal-OSC is stopped.

(2) Start and stop

The Clock Monitor operation must be enabled by setting bit CLM.CLME to 1. Once this bit has been set, it cannot be cleared to 0 by any means other than reset.

The Clock Monitor is automatically started as soon as the main oscillator is stable, indicated by OSTC.MSTS = 1.

The Clock Monitor automatically stops under the following conditions:

- While oscillation stabilization time is being counted after STOP mode is released
- When the main clock is stopped (PCC.MCK bit = 1 during subclock operation, or PCC.CLS bit = 0 during main clock operation)
- When the sampling clock is stopped (240 KHz internal oscillator)
- When the CPU operates with 8 MHz internal oscillator
- When the CPU operates with 240 KHz internal oscillator

(3) Operation when main clock oscillation is stopped (CLME bit = 1)

If oscillation of the main clock is stopped when the CLME bit = 1, an internal reset signal is generated as shown in the following figure.

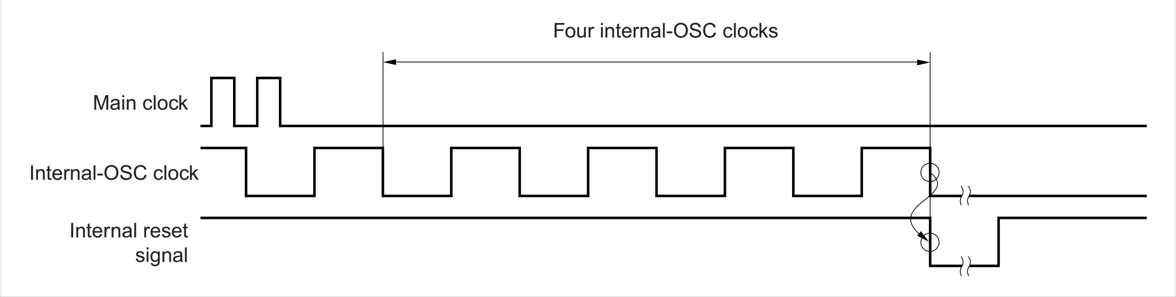


Figure 4-12 When oscillation of main clock is stopped

(4) Operation in STOP mode or after STOP mode is released

If the STOP mode is set with the CLME bit = 1, the monitor operation is stopped in the STOP mode and while the oscillation stabilization time is being counted. After the oscillation stabilization time, the monitor operation is automatically started.

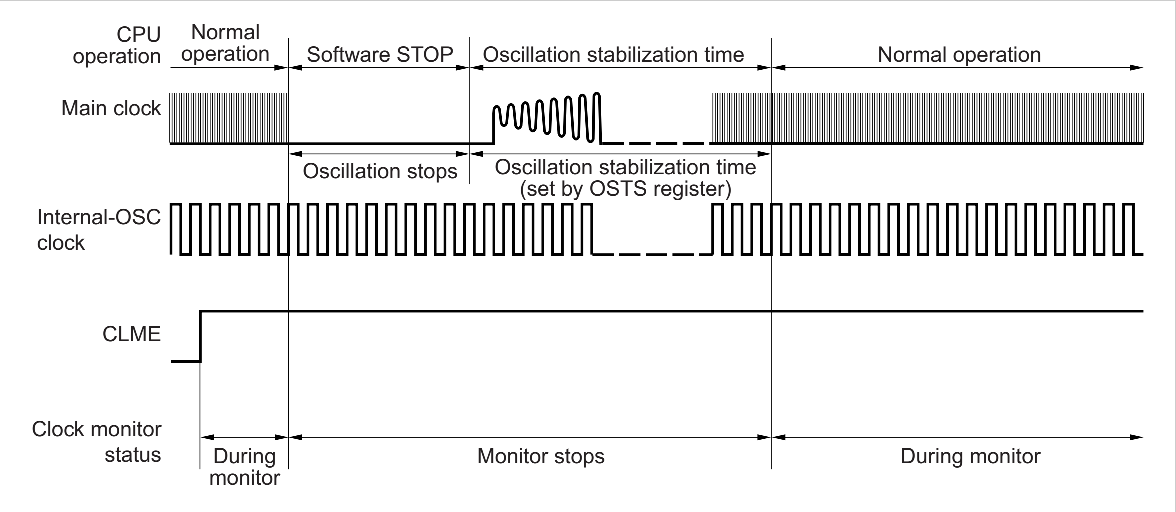


Figure 4-13 Operation in STOP mode or after STOP mode is released

(5) Operation when main clock is stopped

During subclock operation (CLS bit of the PCC register = 1) or when the main clock is stopped by setting the MCK bit of the PCC register to 1, the monitor operation is stopped until the main clock operation is started (CLS bit of PCC register = 0). The monitor operation is automatically started when the main clock operation is started.

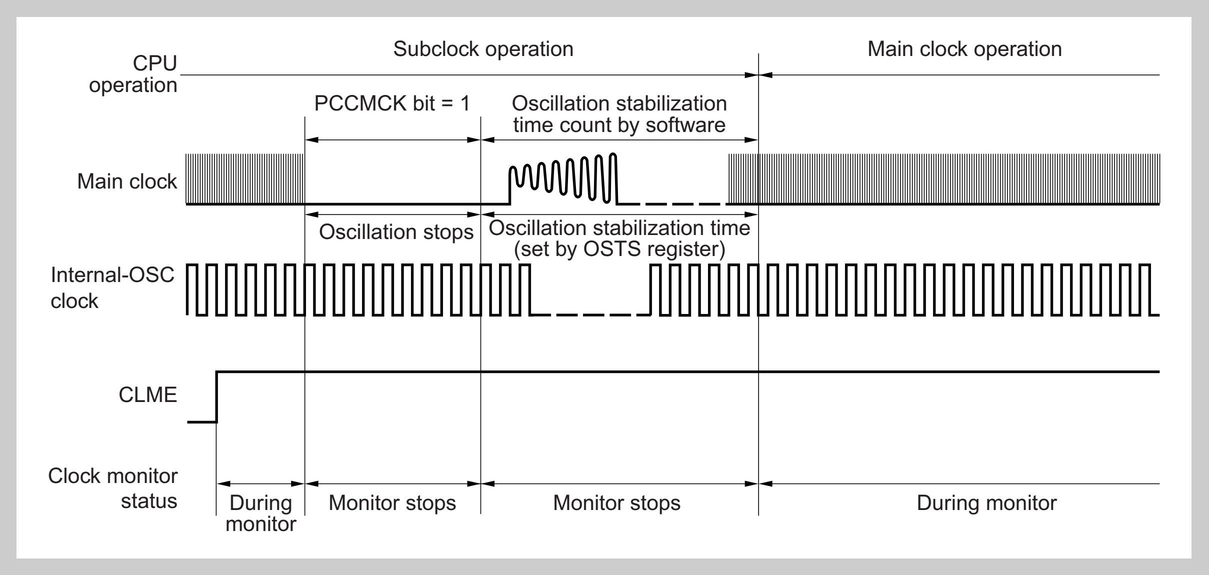


Figure 4-14 Operation When Main Clock Is Stopped (Arbitrary)

(6) Operation during and after power save modes

Main oscillator stopped

If the main oscillator is stopped, the Clock Monitor changes to stand-by. When the main oscillator is restarted after power save mode release, the Clock Monitor restarts automatically.

Internal oscillator stopped

When the 240 KHz internal oscillator is stopped, the Clock Monitor's operation is suspended. Operation is automatically resumed as soon as the internal oscillator is restarted.

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 two non-maskable interrupt requests.

An interrupt is an event that occurs independently of program execution, and an exception is an event whose occurrence is dependent on program execution. Generally, an exception takes precedence over an interrupt.

This controller can process interrupt requests from the on-chip peripheral hardware and external sources. Moreover, exception processing can be started by the TRAP instruction (software exception) or by generation of an exception event (i.e. fetching of an illegal opcode) (exception trap).

Eight levels of software-programmable priorities can be specified for each interrupt request. Starting of interrupt servicing takes no fewer than 5 system clocks after the generation of an interrupt request.

5.1 Features

- Interrupts
  - Non-maskable interrupts: 2 sources
  - Maskable interrupts:

Maskable interrupts	V850ES FE3/FF3	V850ES/FG3		V850ES/FJ3			V850ES/ FK3
		‘F3374 ‘F3375	‘F3376A ‘F3377A	‘F3378	‘F3379 ‘F3380	‘F3381 ‘F3382	
Internal	48	60	65	71	81	83	101
External	8	11	12	15	15	15	16

  - 8 levels of programmable priorities (maskable interrupts)
  - Multiple interrupt control according to priority
  - Masks can be specified for each maskable interrupt request
  - Noise elimination, edge detection and valid edge specification, level detection for external interrupt request signals
  - Wake-up capable
    (analogue noise elimination for external interrupt request signals)
- Exceptions
  - Software exceptions: 2 channels with each 16 sources
  - Exception traps: 2 sources (illegal opcode exception and debug trap)

Table 5-1 V850ES/FE3, V850ES/FF3, V850ES/FG3 interrupt/exception source list (1/3)

Type	Interrupt/Exception Source				Default Priority	Exception Code	Handler Address	Restored PC
	Name	Control Register	Generating Source	Generating Unit				
Reset	RESET	–	Reset input by internal source	RESET	–	0000_H	00000000_H	undef.
Non–maskable	NMI	–	NMI pin valid edge input	Pin	–	0010_H	00000010_H	nextPC
	INTWDT2	–	WDT2 overflow	WDT2	–	0020_H	00000020_H	nextPC
Software exception	TRAP0n (n = 0 to F_H)	–	TRAP instruction	–	–	$004n_H$	00000040_H	nextPC
	TRAP1n (n = 0 to F_H)	–	TRAP instruction	–	–	$005n_H$	00000050_H	nextPC
Exception trap	ILGOP/ DBG0	–	Illegal opcode/DBTRAP instruction	–	–	0060_H	00000060_H	nextPC
Maskable	INTLVIL	LVILIC	Low voltage detection (voltage falling below reference level)	POCLVI	0	0080_H	00000080_H	nextPC
	INTLVIH	LVIHIC	Low voltage detection (voltage rising above reference level)	POCLVI	1	0090_H	00000090_H	nextPC
	INTP0	PIC0	External interrupt 0	Pin	2	$00A0_H$	$000000A0_H$	nextPC
	INTP1	PIC1	External interrupt 1	Pin	3	$00B0_H$	$000000B0_H$	nextPC
	INTP2	PIC2	External interrupt 2	Pin	4	$00C0_H$	$000000C0_H$	nextPC
	INTP3	PIC3	External interrupt 3	Pin	5	$00D0_H$	$000000D0_H$	nextPC
	INTP4	PIC4	External interrupt 4	Pin	6	$00E0_H$	$000000E0_H$	nextPC
	INTP5	PIC5	External interrupt 5	Pin	7	$00F0_H$	$000000F0_H$	nextPC
	INTP6	PIC6	External interrupt 6	Pin	8	0100_H	00000100_H	nextPC
	INTP7	PIC7	External interrupt 7	Pin	9	0110_H	00000110_H	nextPC
	INTTAB0OV[b]	TAB0OVIC	TAB0 overflow	TAB0	10	0120_H	00000120_H	nextPC
	INTTAB0CC0[b]	TAB0CCIC0	TAB0 capture 0 / compare 0 match	TAB0	11	0130_H	00000130_H	nextPC
	INTTAB0CC1[b]	TAB0CCIC1	TAB0 capture 1 / compare 1 match	TAB0	12	0140_H	00000140_H	nextPC
	INTTAB0CC2[b]	TAB0CCIC2	TAB0 capture 2 / compare 2 match	TAB0	13	0150_H	00000150_H	nextPC
	INTTAB0CC3[b]	TAB0CCIC3	TAB0 capture 3 / compare 3 match	TAB0	14	0160_H	00000160_H	nextPC
	INTTAA0OV	TAA0OVIC	TAA0 overflow	TAA0	15	0170_H	00000170_H	nextPC
	INTTAA0CC0	TAA0CCIC0	TAA0 capture 0 / compare 0 match	TAA0	16	0180_H	00000180_H	nextPC
	INTTAA0CC1	TAA0CCIC1	TAA0 capture 1 / compare 1 match	TAA0	17	0190_H	00000190_H	nextPC
	INTTAA1OV	TAA1OVIC	TAA1 overflow	TAA1	18	$01A0_H$	$000001A0_H$	nextPC
	INTTAA1CC0	TAA1CCIC0	TAA1 capture 0 / compare 0 match	TAA1	19	$01B0_H$	$000001B0_H$	nextPC
	INTTAA1CC1	TAA1CCIC1	TAA1 capture 1 / compare 1 match	TAA1	20	$01C0_H$	$000001C0_H$	nextPC
	INTTAA2OV	TAA2OVIC	TAA2 overflow	TAA2	21	$01D0_H$	$000001D0_H$	nextPC
	INTTAA2CC0	TAA2CCIC0	TAA2 capture 0 / compare 0 match	TAA2	22	$01E0_H$	$000001E0_H$	nextPC
	INTTAA2CC1	TAA2CCIC1	TAA2 capture 1 / compare 1 match	TAA2	23	$01F0_H$	$000001F0_H$	nextPC
	INTTAA3OV	TAA3OVIC	TAA3 overflow	TAA3	24	0200_H	00000200_H	nextPC
	INTTAA3CC0	TAA3CCIC0	TAA3 capture 0 / compare 0 match	TAA3	25	0210_H	00000210_H	nextPC
	INTTAA3CC1	TAA3CCIC1	TAA3 capture 1 / compare 1 match	TAA3	26	0220_H	00000220_H	nextPC
	INTTAA4OV	TAA4OVIC	TAA4 overflow	TAA4	27	0230_H	00000230_H	nextPC

Table 5-1 V850ES/FE3, V850ES/FF3, V850ES/FG3 interrupt/exception source list (2/3)

Type	Interrupt/Exception Source				Default Priority	Exception Code	Handler Address	Restored PC
	Name	Control Register	Generating Source	Generating Unit				
Maskable	INTTAA4CC0	TAA4CCIC0	TAA4 capture 0 / compare 0 match	TAA4	28	0240_H	00000240_H	nextPC
	INTTAA4CC1	TAA4CCIC1	TAA4 capture 1 / compare 1 match	TAA4	29	0250_H	00000250_H	nextPC
	INTTM0EQ0	TM0EQIC0	TMM0 compare match	TMM0	30	0260_H	00000260_H	nextPC
	INTCB0R	CB0RIC	CSIB0 reception completion / reception error	CSIB0	31	0270_H	00000270_H	nextPC
	INTCB0T	CB0TIC	CSIB0 consecutive transmission write enable	CSIB0	32	0280_H	00000280_H	nextPC
	INTCB1R	CB1RIC	CSIB1 reception completion / reception error	CSIB1	33	0290_H	00000290_H	nextPC
	INTCB1T	CB1TIC	CSIB1 consecutive transmission write enable	CSIB1	34	$02A0_H$	$000002A0_H$	nextPC
	INTUD0S	UD0SIC	UARTD0 status interrupt	UARTD0	35	$02B0_H$	$000002B0_H$	nextPC
	INTUD0R	UD0RIC	UARTD0 reception completion	UARTD0	36	$02C0_H$	$000002C0_H$	nextPC
	INTUD0T	UD0TIC	UARTD0 consecutive transmission enable	UARTD0	37	$02D0_H$	$000002D0_H$	nextPC
	INTUD1S	UD1SIC	UARTD1 status interrupt	UARTD1	38	$02E0_H$	$000002E0_H$	nextPC
	INTUD1R	UD1RIC	UARTD1 reception completion	UARTD1	39	$02F0_H$	$000002F0_H$	nextPC
	INTUD1T	UD1TIC	UARTD1 consecutive transmission enable	UARTD1	40	0300_H	00000300_H	nextPC
	INTIIC0	IIC0IC	IIC0 transfer completion	IIC0	41	0310_H	00000310_H	nextPC
	INTUD4S[a]	UD4SIC	UARTD4 status interrupt	UARTD4				
	INTAD	ADIC	A/D conversion completion	AD	42	0320_H	00000320_H	nextPC
	INTC0ERR	C0ERRIC	CAN0 error	CAN0	43	0330_H	00000330_H	nextPC
	INTC0WUP	C0WUPIC	CAN0 wake–up	CAN0	44	0340_H	00000340_H	nextPC
	INTC0REC	C0RECIC	CAN0 reception	CAN0	45	0350_H	00000350_H	nextPC
	INTC0TRX	C0TRXIC	CAN0 transmission	CAN0	46	0360_H	00000360_H	nextPC
	INTDMA0	DMAIC0	DMA0 transfer completion	DMAC	47	0370_H	00000370_H	nextPC
	INTDMA1	DMAIC1	DMA1 transfer completion	DMAC	48	0380_H	00000380_H	nextPC
	INTDMA2	DMAIC2	DMA2 transfer completion	DMAC	49	0390_H	00000390_H	nextPC
	INTDMA3	DMAIC3	DMA3 transfer completion	DMAC	50	$03A0_H$	$000003A0_H$	nextPC
	INTKR	KRIC	Key return interrupt	KR	51	$03B0_H$	$000003B0_H$	nextPC
	INTWTI	WTIIC	Watch Timer interval	WT	52	$03C0_H$	$000003C0_H$	nextPC
	INTWT	WTIC	Watch Timer reference time	WT	53	$03D0_H$	$000003D0_H$	nextPC
	Reserved	–	–	–	54	$03E0_H$	$000003E0_H$	nextPC
	INTFL	FLIC	Flash programming completion	FLASH	55	$03F0_H$	$000003F0_H$	nextPC
	INTP8[b]	PIC8	External interrupt 8	Pin	56	0400_H	00000400_H	nextPC
	INTP9[b]	PIC9	External interrupt 9	Pin	57	0410_H	00000410_H	nextPC
	INTP10[b]	PIC10	External interrupt 10	Pin	58	0420_H	00000420_H	nextPC
	INTTAB1OV[b]	TAB1OVIC	TAB1 overflow	TAB1	59	0430_H	00000430_H	nextPC
	INTTAB1CC0[b]	TAB1CCIC0	TAB1 capture 0 / compare 0 match	TAB1	60	0440_H	00000440_H	nextPC
	INTTAB1CC1[b]	TAB1CCIC1	TAB1 capture 1 / compare 1 match	TAB1	61	0450_H	00000450_H	nextPC

Table 5-1 V850ES/FE3, V850ES/FF3, V850ES/FG3 interrupt/exception source list (3/3)

Type	Interrupt/Exception Source				Default Priority	Exception Code	Handler Address	Restored PC
	Name	Control Register	Generating Source	Generating Unit				
Maskable	INTTAB1CC2[b]	TAB1CCIC2	TAB1 capture 2 / compare 2 match	TAB1	62	0460_H	00000460_H	nextPC
	INTTAB1CC3[b]	TAB1CCIC3	TAB1 capture 3 / compare 3 match	TAB1	63	0470_H	00000470_H	nextPC
	INTUD2S[b]	UD2SIC	UARTD2 status interrupt	UARTD2	64	0480_H	00000480_H	nextPC
	INTUD2R[b]	UD2RIC	UARTD2 reception completion	UARTD2	65	0490_H	00000490_H	nextPC
	INTUD2T[b]	UD2TIC	UARTD2 consecutive transmission enable	UARTD2	66	$04A0_H$	$000004A0_H$	nextPC
	INTC1ERR[b]	C1ERRIC	CAN1 error	CAN1	67	$04B0_H$	$000004B0_H$	nextPC
	INTC1WUP[b]	C1WUPIC	CAN1 wake-up	CAN1	68	$04C0_H$	$000004C0_H$	nextPC
	INTC1REC[b]	C1RECIC	CAN1 reception	CAN1	69	$04D0_H$	$000004D0_H$	nextPC
	INTC1TRX[b]	C1TRXIC	CAN1 transmission	CAN1	70	$04E0_H$	$000004E0_H$	nextPC
	Reserved	–	–	–	71	$04F0_H$	$000004F0_H$	nextPC
	Reserved	–	–	–	72	0500_H	00000500_H	nextPC
	Reserved	–	–	–	73	0510_H	00000510_H	nextPC
	INTP14[c]	PIC14	External interrupt 14	Pin	74	0520_H	00000520_H	nextPC
	INTUD3S[a]	UD3SIC	UARTD3 status interrupt	UARTD3	75	0530_H	00000530_H	nextPC
	INTUD3R[a]	UD3RIC	UARTD3 reception completion	UARTD3	76	0540_H	00000540_H	nextPC
	INTUD3T[a]	UD3TIC	UARTD3 consecutive transmission enable	UARTD3	77	0550_H	00000550_H	nextPC
	INTUD4R[a]	UD4RIC	UARTD4 reception completion	UARTD4	78	0560_H	00000560_H	nextPC
	INTUD4T[a]	UD4TIC	UARTD4 consecutive transmission enable	UARTD4	79	0570_H	00000570_H	nextPC

[a] not available for
- V850ES/FE3
- V850ES/FF3
- μPD70F3374, μPD70F3375 of V850ES/FG3
- μPD70F3378 of V850ES/FJ3

[b] not available for V850ES/FE3, V850ES/FF3

[c] not available for
- V850ES/FE3
- V850ES/FF3
- μPD70F3374, μPD70F3375 of V850ES/FG3

Shared interrupts Some interrupt sources share the same maskable interrupt (see *Table 5-2*).

Table 5-2 V850ES/FE3, V850ES/FF3, V850ES/FG3 shared maskable interrupts

Interrupt Source				Default Priority
Name	Generating Unit	Name	Generating Unit	
INTIIC0	IIC0	INTUD4S	UARTD4	41

Caution The interrupt sources in *Table 5-2* must not be used concurrently.

Table 5-3 V850ES/FJ3, V850ES/FK3 interrupt/exception source list (1/4)

Type	Interrupt/Exception Source				Default Priority	Exception Code	Handler Address	Restored PC
	Name	Control Register	Generating Source	Generating Unit				
Reset	RESET	–	Reset input by internal source	RESET	–	0000$_H$	00000000$_H$	undef.
Non-maskable	NMI	–	NMI pin valid edge input	Pin	–	0010$_H$	00000010$_H$	nextPC
	INTWDT2	–	WDT2 overflow	WDT2	–	0020$_H$	00000020$_H$	nextPC
Software exception	TRAP0n (n = 0 to F$_H$)	–	TRAP instruction	–	–	004n$_H$	00000040$_H$	nextPC
	TRAP1n (n = 0 to F$_H$)	–	TRAP instruction	–	–	005n$_H$	00000050$_H$	nextPC
Exception trap	ILGOP/ DBG0	–	Illegal opcode/DBTRAP instruction	–	–	0060$_H$	00000060$_H$	nextPC
Maskable	INTLVIL	LVILIC	Low voltage detection (voltage falling below reference level)	POCLVI	0	0080$_H$	00000080$_H$	nextPC
	INTLVIH	LVIHIC	Low voltage detection (voltage rising above reference level)	POCLVI	1	0090$_H$	00000090$_H$	nextPC
	INTP0	PIC0	External interrupt 0	Pin	2	00A0$_H$	000000A0$_H$	nextPC
	INTP1	PIC1	External interrupt 1	Pin	3	00B0$_H$	000000B0$_H$	nextPC
	INTP2	PIC2	External interrupt 2	Pin	4	00C0$_H$	000000C0$_H$	nextPC
	INTP3	PIC3	External interrupt 3	Pin	5	00D0$_H$	000000D0$_H$	nextPC
	INTP4	PIC4	External interrupt 4	Pin	6	00E0$_H$	000000E0$_H$	nextPC
	INTP5	PIC5	External interrupt 5	Pin	7	00F0$_H$	000000F0$_H$	nextPC
	INTP6	PIC6	External interrupt 6	Pin	8	0100$_H$	00000100$_H$	nextPC
	INTP7	PIC7	External interrupt 7	Pin	9	0110$_H$	00000110$_H$	nextPC
	INTTAB0OV	TAB0OVIC	TAB0 overflow	TAB0	10	0120$_H$	00000120$_H$	nextPC
	INTTAB0CC0	TAB0CCIC0	TAB0 capture 0 / compare 0 match	TAB0	11	0130$_H$	00000130$_H$	nextPC
	INTTAB0CC1	TAB0CCIC1	TAB0 capture 1 / compare 1 match	TAB0	12	0140$_H$	00000140$_H$	nextPC
	INTTAB0CC2	TAB0CCIC2	TAB0 capture 2 / compare 2 match	TAB0	13	0150$_H$	00000150$_H$	nextPC
	INTTAB0CC3	TAB0CCIC3	TAB0 capture 3 / compare 3 match	TAB0	14	0160$_H$	00000160$_H$	nextPC
	INTTAA0OV	TAA0OVIC	TAA0 overflow	TAA0	15	0170$_H$	00000170$_H$	nextPC
	INTTAA0CC0	TAA0CCIC0	TAA0 capture 0 / compare 0 match	TAA0	16	0180$_H$	00000180$_H$	nextPC
	INTTAA0CC1	TAA0CCIC1	TAA0 capture 1 / compare 1 match	TAA0	17	0190$_H$	00000190$_H$	nextPC
	INTTAA1OV	TAA1OVIC	TAA1 overflow	TAA1	18	01A0$_H$	000001A0$_H$	nextPC
	INTTAA1CC0	TAA1CCIC0	TAA1 capture 0 / compare 0 match	TAA1	19	01B0$_H$	000001B0$_H$	nextPC
	INTTAA1CC1	TAA1CCIC1	TAA1 capture 1 / compare 1 match	TAA1	20	01C0$_H$	000001C0$_H$	nextPC
	INTTAA2OV	TAA2OVIC	TAA2 overflow	TAA2	21	01D0$_H$	000001D0$_H$	nextPC
	INTTAA2CC0	TAA2CCIC0	TAA2 capture 0 / compare 0 match	TAA2	22	01E0$_H$	000001E0$_H$	nextPC
	INTTAA2CC1	TAA2CCIC1	TAA2 capture 1 / compare 1 match	TAA2	23	01F0$_H$	000001F0$_H$	nextPC
	INTTAA3OV	TAA3OVIC	TAA3 overflow	TAA3	24	0200$_H$	00000200$_H$	nextPC
	INTTAA3CC0	TAA3CCIC0	TAA3 capture 0 / compare 0 match	TAA3	25	0210$_H$	00000210$_H$	nextPC
	INTTAA3CC1	TAA3CCIC1	TAA3 capture 1 / compare 1 match	TAA3	26	0220$_H$	00000220$_H$	nextPC
	INTTAA4OV	TAA4OVIC	TAA4 overflow	TAA4	27	0230$_H$	00000230$_H$	nextPC

Table 5-3 V850ES/FJ3, V850ES/FK3 interrupt/exception source list (2/4)

Type	Interrupt/Exception Source				Default Priority	Exception Code	Handler Address	Restored PC
	Name	Control Register	Generating Source	Generating Unit				
Maskable	INTTAA4CC0	TAA4CCIC0	TAA4 capture 0 / compare 0 match	TAA4	28	0240$_H$	00000240$_H$	nextPC
	INTTAA4CC1	TAA4CCIC1	TAA4 capture 1 / compare 1 match	TAA4	29	0250$_H$	00000250$_H$	nextPC
	INTTM0EQ0	TM0EQIC0	TMM0 compare match	TMM0	30	0260$_H$	00000260$_H$	nextPC
	INTCB0R	CB0RIC	CSIB0 reception completion / reception error	CSIB0	31	0270$_H$	00000270$_H$	nextPC
	INTCB0T	CB0TIC	CSIB0 consecutive transmission write enable	CSIB0	32	0280$_H$	00000280$_H$	nextPC
	INTCB1R	CB1RIC	CSIB1 reception completion / reception error	CSIB1	33	0290$_H$	00000290$_H$	nextPC
	INTCB1T	CB1TIC	CSIB1 consecutive transmission write enable	CSIB1	34	02A0$_H$	000002A0$_H$	nextPC
	INTUD0S	UD0SIC	UARTD0 status interrupt	UARTD0	35	02B0$_H$	000002B0$_H$	nextPC
	INTUD0R	UD0RIC	UARTD0 reception completion	UARTD0	36	02C0$_H$	000002C0$_H$	nextPC
	INTUD0T	UD0TIC	UARTD0 consecutive transmission enable	UARTD0	37	02D0$_H$	000002D0$_H$	nextPC
	INTUD1S	UD1SIC	UARTD1 status interrupt	UARTD0	38	02E0$_H$	000002E0$_H$	nextPC
	INTUD1R	UD1RIC	UARTD1 reception completion	UARTD0	39	02F0$_H$	000002F0$_H$	nextPC
	INTUD1T	UD1TIC	UARTD1 consecutive transmission enable	UARTD0	40	0300$_H$	00000300$_H$	nextPC
	INTIIC0	IIC0IC	IIC0 transfer completion	IIC0	41	0310$_H$	00000310$_H$	nextPC
	INTUD4S	UD4SIC	UARTD4 status interrupt	UARTD4				
	INTAD	ADIC	A/D conversion completion	AD	42	0320$_H$	00000320$_H$	nextPC
	INTC0ERR	C0ERRIC	CAN0 error	CAN0	43	0330$_H$	00000330$_H$	nextPC
	INTC0WUP	C0WUPIC	CAN0 wake-up	CAN0	44	0340$_H$	00000340$_H$	nextPC
	INTC0REC	C0RECIC	CAN0 reception	CAN0	45	0350$_H$	00000350$_H$	nextPC
	INTC0TRX	C0TRXIC	CAN0 transmission	CAN0	46	0360$_H$	00000360$_H$	nextPC
	INTDMA0	DMAIC0	DMA0 transfer completion	DMA	47	0370$_H$	00000370$_H$	nextPC
	INTDMA1	DMAIC1	DMA1 transfer completion	DMA	48	0380$_H$	00000380$_H$	nextPC
	INTDMA2	DMAIC2	DMA2 transfer completion	DMA	49	0390$_H$	00000390$_H$	nextPC
	INTDMA3	DMAIC3	DMA3 transfer completion	DMA	50	03A0$_H$	000003A0$_H$	nextPC
	INTKR	KRIC	Key return interrupt	KR	51	03B0$_H$	000003B0$_H$	nextPC
	INTWTI	WTIIC	Watch Timer interval	WT	52	03C0$_H$	000003C0$_H$	nextPC
	INTWT	WTIC	Watch Timer reference time	WT	53	03D0$_H$	000003D0$_H$	nextPC
	Reserved	–	–	–	54	03E0$_H$	000003E0$_H$	nextPC
	INTFL	FLIC	Flash programming completion	FLASH	55	03F0$_H$	000003F0$_H$	nextPC
	INTP8	PIC8	External interrupt 8	Pin	56	0400$_H$	00000400$_H$	nextPC
	INTP9	PIC9	External interrupt 9	Pin	57	0410$_H$	00000410$_H$	nextPC
	INTP10	PIC10	External interrupt 10	Pin	58	0420$_H$	00000420$_H$	nextPC
	INTTAB1OV	TAB1OVIC	TAB1 overflow	TAB1	59	0430$_H$	00000430$_H$	nextPC
	INTTAB1CC0	TAB1CCIC0	TAB1 capture 0 / compare 0 match	TAB1	60	0440$_H$	00000440$_H$	nextPC
	INTTAB1CC1	TAB1CCIC1	TAB1 capture 1 / compare 1 match	TAB1	61	0450$_H$	00000450$_H$	nextPC

Table 5-3 V850ES/FJ3, V850ES/FK3 interrupt/exception source list (3/4)

Type	Interrupt/Exception Source				Default Priority	Exception Code	Handler Address	Restored PC
	Name	Control Register	Generating Source	Generating Unit				
Maskable	INTTAB1CC2	TAB1CCIC2	TAB1 capture 2 / compare 2 match	TAB1	62	0460$_H$	00000460$_H$	nextPC
	INTTAB1CC3	TAB1CCIC3	TAB1 capture 3 / compare 3 match	TAB1	63	0470$_H$	00000470$_H$	nextPC
	INTUD2S	UD2SIC	UARTD2 status interrupt	UARTD2	64	0480$_H$	00000480$_H$	nextPC
	INTUD2R	UD2RIC	UARTD2 reception completion	UARTD2	65	0490$_H$	00000490$_H$	nextPC
	INTUD2T	UD2TIC	UARTD2 consecutive transmission enable	UARTD2	66	04A0$_H$	000004A0$_H$	nextPC
	INTC1ERR	C1ERRIC	CAN1 error	CAN1	67	04B0$_H$	000004B0$_H$	nextPC
	INTC1WUP	C1WUPIC	CAN1 wake-up	CAN1	68	04C0$_H$	000004C0$_H$	nextPC
	INTC1REC	C1RECIC	CAN1 reception	CAN1	69	04D0$_H$	000004D0$_H$	nextPC
	INTC1TRX	C1TRXIC	CAN1 transmission	CAN1	70	04E0$_H$	000004E0$_H$	nextPC
	INTP11	PIC11	External interrupt 11	Pin	71	04F0$_H$	000004F0$_H$	nextPC
	INTP12	PIC12	External interrupt 12	Pin	72	0500$_H$	00000500$_H$	nextPC
	INTP13	PIC13	External interrupt 13	Pin	73	0510$_H$	00000510$_H$	nextPC
	INTP14	PIC14	External interrupt 14	Pin	74	0520$_H$	00000520$_H$	nextPC
	INTUD3S[a]	UD3SIC	UARTD3 status interrupt	UARTD3	75	0530$_H$	00000530$_H$	nextPC
	INTUD3R[a]	UD3RIC	UARTD3 reception completion	UARTD3	76	0540$_H$	00000540$_H$	nextPC
	INTUD3T[a]	UD3TIC	UARTD3 consecutive transmission enable	UARTD3	77	0550$_H$	00000550$_H$	nextPC
	INTUD4R[a]	UD4RIC	UARTD4 reception completion	UARTD4	78	0560$_H$	00000560$_H$	nextPC
	INTUD4T[a]	UD4TIC	UARTD4 consecutive transmission enable	UARTD4	79	0570$_H$	00000570$_H$	nextPC
	INTTAB2OV	TAB2OVIC	TAB2 overflow	TAB2	80	0580$_H$	00000580$_H$	nextPC
	INTTAB2CC0	TAB2CCIC0	TAB2 capture 0 / compare 0 match	TAB2	81	0590$_H$	00000590$_H$	nextPC
	INTTAB2CC1	TAB2CCIC1	TAB2 capture 1 / compare 1 match	TAB2	82	05A0$_H$	000005A0$_H$	nextPC
	INTTAB2CC2	TAB2CCIC2	TAB2 capture 2 / compare 2 match	TAB2	83	05B0$_H$	000005B0$_H$	nextPC
	INTTAB2CC3	TAB2CCIC3	TAB2 capture 3 / compare 3 match	TAB2	84	05C0$_H$	000005C0$_H$	nextPC
	INTUD5S[a]	UD5SIC	UARTD5 status interrupt	UARTD5	85	05D0$_H$	000005D0$_H$	nextPC
	INTCB2R	CB2RIC	CSIB2 reception completion / reception error	CSIB2	86	05E0$_H$	000005E0$_H$	nextPC
	INTUD5R[a]	UD5RIC	UARTD5 reception completion	UARTD5				
	INTCB2T	CB2TIC	CSIB2 consecutive transmission write enable	CSIB2	87	05F0$_H$	000005F0$_H$	nextPC
	INTUD5T[a]	UD5TIC	UARTD5 consecutive transmission enable	UARTD5				
	INTC2ERR	C2ERRIC	CAN2 error	CAN2	88	0600$_H$	00000600$_H$	nextPC
	INTC2WUP	C2WUPIC	CAN2 wake-up	CAN2	89	0610$_H$	00000610$_H$	nextPC
	INTC2REC	C2RECIC	CAN2 reception	CAN2	90	0620$_H$	00000620$_H$	nextPC
	INTC2TRX	C2TRXIC	CAN2 transmission	CAN2	91	0630$_H$	00000630$_H$	nextPC
	INTC3ERR[a]	C3ERRIC	CAN3 error	CAN3	92	0640$_H$	00000640$_H$	nextPC
	INTC3WUP[a]	C3WUPIC	CAN3 wake-up	CAN3	93	0650$_H$	00000650$_H$	nextPC
	INTC3REC[a]	C3RECIC	CAN3 reception	CAN3	94	0660$_H$	00000660$_H$	nextPC
	INTC3TRX[a]	C3TRXIC	CAN3 transmission	CAN3	95	0670$_H$	00000670$_H$	nextPC

Table 5-3 V850ES/FJ3, V850ES/FK3 interrupt/exception source list (4/4)

Type	Interrupt/Exception Source				Default Priority	Exception Code	Handler Address	Restored PC
	Name	Control Register	Generating Source	Generating Unit				
Maskable	INTP15[b]	PIC15	External interrupt 15	Pin	96	0680_H	00000680_H	nextPC
	INTTAA5OV[b]	TAA5OVIC	TAA5 overflow	TAA5	97	0690_H	00000690_H	nextPC
	INTTAA5CC0[b]	TAA5CCIC0	TAA5 capture 0 / compare 0 match	TAA5	98	$06A0_H$	$000006A0_H$	nextPC
	INTTAA5CC1[b]	TAA5CCIC1	TAA5 capture 1 / compare 1 match	TAA5	99	$06B0_H$	$000006B0_H$	nextPC
	INTTAA6OV[b]	TAA6OVIC	TAA6 overflow	TAA6	100	$06C0_H$	$000006C0_H$	nextPC
	INTTAA6CC0[b]	TAA6CCIC0	TAA6 capture 0 / compare 0 match	TAA6	101	$06D0_H$	$000006D0_H$	nextPC
	INTTAA6CC1[b]	TAA6CCIC1	TAA6 capture 1 / compare 1 match	TAA6	102	$06E0_H$	$000006E0_H$	nextPC
	INTTAA7OV[b]	TAA7OVIC	TAA7 overflow	TAA7	103	$06F0_H$	$000006F0_H$	nextPC
	INTTAA7CC0[b]	TAA7CCIC0	TAA7 capture 0 / compare 0 match	TAA7	104	0700_H	00000700_H	nextPC
	INTTAA7CC1[b]	TAA7CCIC1	TAA7 capture 1 / compare 1 match	TAA7	105	0710_H	00000710_H	nextPC
	INTUD6S[b]	UD6SIC	UARTD6 status interrupt	UARTD6	106	0720_H	00000720_H	nextPC
	INTCB3R[c]	CB3RIC	CSIB3 reception completion / reception error	CSIB3	107	0730_H	00000730_H	nextPC
	INTUD6R[b]	UD6RIC	UARTD6 reception completion	UARTD6				
	INTCB3T[c]	CB3TIC	CSIB3 consecutive transmission write enable	CSIB3	108	0740_H	00000740_H	nextPC
	INTUD6T[b]	UD6TIC	UARTD6 consecutive transmission enable	UARTD6				
	INTUD7S[b]	UD7SIC	UARTD7 status interrupt	UARTD7	109	0750_H	00000750_H	nextPC
	INTUD7R[b]	UD7RIC	UARTD7 reception completion	UARTD7	110	0760_H	00000760_H	nextPC
	INTUD7T[b]	UD7TIC	UARTD7 consecutive transmission enable	UARTD7	111	0770_H	00000770_H	nextPC
	INTAD1[b]	AD1IC	A/D1 conversion completion	AD1	112	0780_H	00000780_H	nextPC
	INTC4ERR[b]	C4ERRIC	CAN4 error	CAN4	113	0790_H	00000790_H	nextPC
	INTC4WUP[b]	C4WUPIC	CAN4 wake-up	CAN4	114	$07A0_H$	$000007A0_H$	nextPC
	INTC4REC[b]	C4RECIC	CAN4 reception	CAN4	115	$07B0_H$	$000007B0_H$	nextPC
	INTC4TRX[b]	C4TRXIC	CAN4 transmission	CAN4	116	$07C0_H$	$000007C0_H$	nextPC

a) not available for µPD70F3378 of V850ES/FJ3

b) not available for V850ES/FJ3

c) not available for µPD70F3378, µPD70F3379, µPD70F3380 of V850ES/FJ3

Shared interrupts Some interrupt sources share the same maskable interrupt (see *Table 5-4*).

Table 5-4 V850ES/FJ3, V850ES/FK3 shared maskable interrupts

Interrupt Source				Default Priority
Name	Generating Unit	Name	Generating Unit	
INTIIC0	IIC0	INTUD4S	UARTD4	41
INTCB2R	CSIB2	INTUD5R	UARTD5	86
INTCB2T		INTUD5T		87
INTCB3R	CSIB3	INTUD3R	UARTD6	107
INTCB3T		INTUD3T		108

Caution The interrupt sources in *Table 5-4* must not be used concurrently.

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:

- NMI: NMI pin input
- INTWDT2: Non-maskable Watchdog Timer interrupt request

When the valid edge, specified by the INTR0.INTR02 and INTF0.INTF02, is detected atthe NMI pin, the NMI interrupt occurs.

The Watchdog Timer interrupt request is only effective as non-maskable interrupt if WDTM2.WDM2[1:0] = 01_B is chosen in the Watchdog Timer mode register.

If multiple non-maskable interrupts are generated at the same time, the highest priority servicing is executed according to the following priority order (the lower priority interrupt is ignored):

INTWDT2 > NMI

Note that if a NMI from port pin or INTWDT2 request is generated while NMI from port pin is being serviced, the service is executed as follows.

(1) If a NMI is generated while NMI is being serviced

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).

(2) If a INTWDT2 request is generated while NMI is being serviced

If the PSW.NP bit remains set (1) while NMI is being serviced, the new INTWDT2 request is held pending. The pending INTWDT2 request is acknowledge after servicing of the current NMI request has finished (after execution of the RETI instruction).

If the PSW.NP bit is cleared (0) while NMI is being serviced, the newly generated INTWDT2 request is executed (NMI servicing is halted).

Caution

1. Although the values of the PC and PSW are saved to an NMI status save register (FEPC, FEPSW) when a non-maskable interrupt request is generated, only the NMI can be restored by the RETI instruction at this time. Because INTWDT2 cannot be restored by the RETI instruction, the system must be reset after servicing this interrupt.
2. If PSW.NP is cleared to 0 by the LDSR instruction during non-maskable interrupt servicing, a NMI interrupt afterwards cannot be acknowledged correctly.

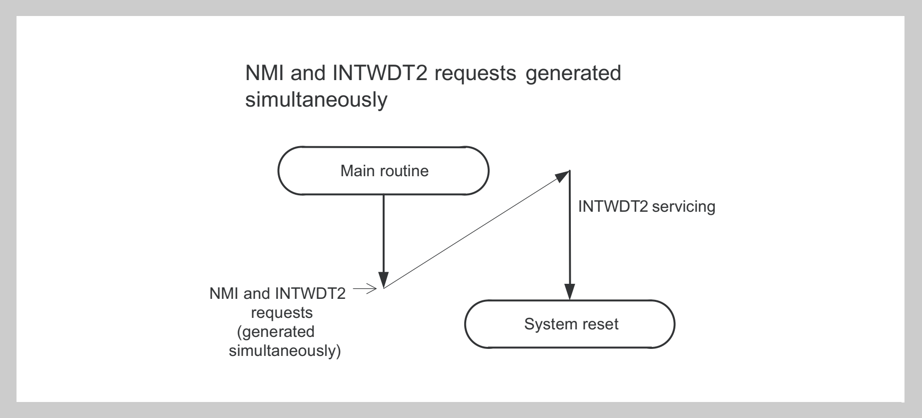

NMI and INTWDT2 requests generated simultaneously
Main routine
INTWDT2 servicing
NMI and INTWDT2 requests (generated simultaneously)
System reset

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

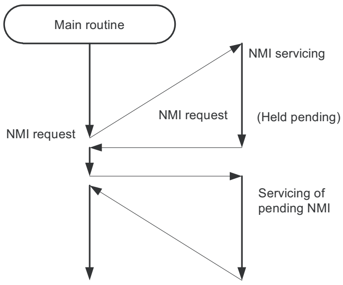
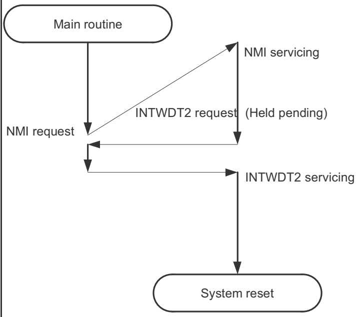
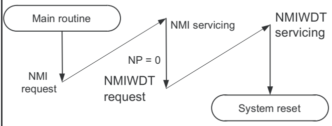
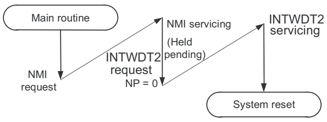
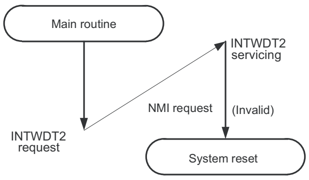
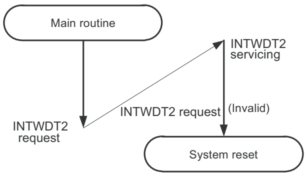
NMI being serviced	NMI request generated during NMI servicing	
	NMI	INTWDT2
NMI	NMI request generated during NMI servicing  Main routine NMI servicing NMI request (Held pending) NMI request Servicing of pending NMI	INTWDT2 request generated during NMI servicing (NP = 1 retained before NMI1 request)  Main routine NMI servicing INTWDT2 request (Held pending) NMI request INTWDT2 servicing System reset INTWDT2 request generated during NMI servicing (NP=0 set before INTWDT2 request)  Main routine NMI servicing NMIWDT servicing NP = 0 NMI request NMIWDT request System reset INTWDT2 request generated during NMI servicing (NP=0 set after INTWDT2 request)  Main routine NMI servicing (Held pending) INTWDT2 servicing INTWDT2 request NP = 0 NMI request System reset
INTWDT2	NMI request generated during INTWDT2 servicing  Main routine INTWDT2 servicing NMI request (Invalid) INTWDT2 request System reset	NMI request generated during INTWDT2 servicing  Main routine INTWDT2 servicing INTWDT2 request (Invalid) INTWDT2 request System reset

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.
2. Saves the current PSW to FEPSW.
3. Writes exception code 0010_H to the higher halfword (FECC) of ECR.
4. Sets the NP and ID bits of the PSW and clears the EP bit.
5. Sets the handler address corresponding to the non-maskable interrupt to the PC, and transfers control.

The processing configuration of a non-maskable interrupt is shown in *Figure 5-3*.

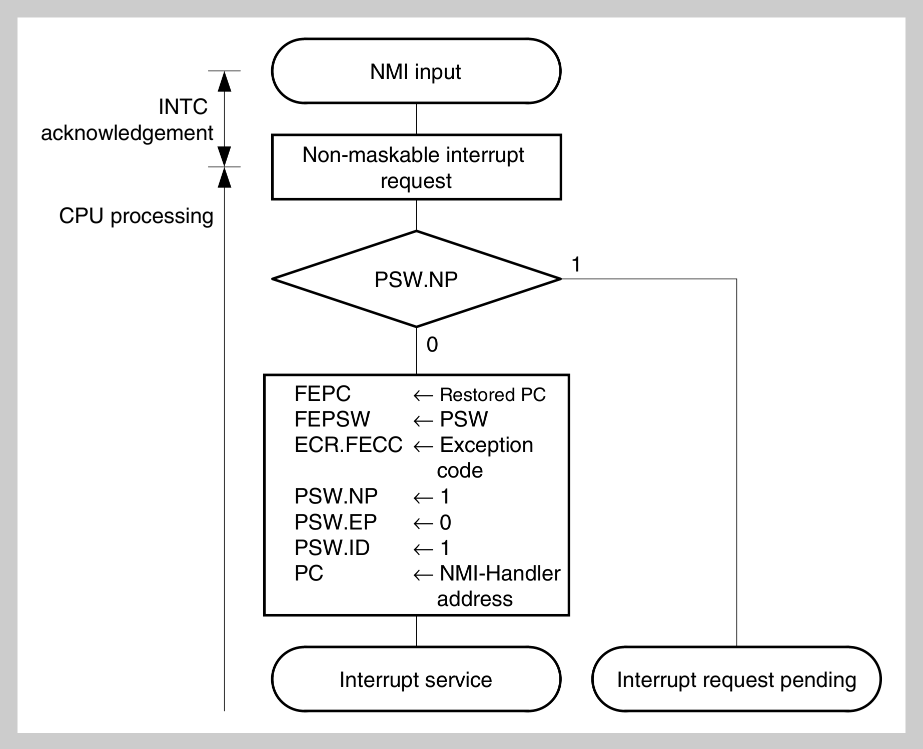


Figure 5-3 Processing configuration of non-maskable interrupt

5.2.2 Restore

(1) NMI

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

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

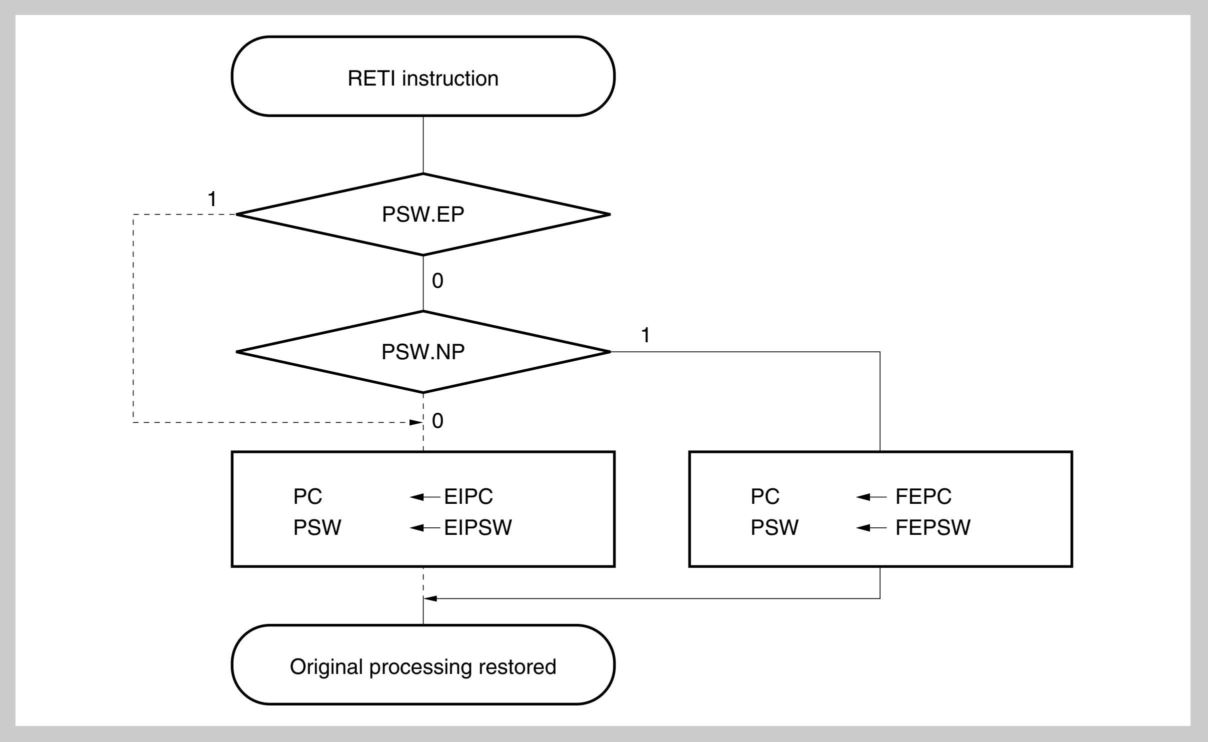


Figure 5-4 RETI instruction processing

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.

(2) INTWDT2

Restoring by RETI instruction is not possible. Perform a system reset after interrupt servicing.

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.

Initial Value 00000020$_H$

	31																								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

Bit position	Bit name	Function
7	NP	Indicates whether NMI interrupt processing is in progress. 0: No NMI interrupt processing 1: NMI interrupt currently being processed

5.2.4 NMI control

The NMI can be configured to generate a non-maskable interrupt upon a rising, falling or both edges at the NMI pin. To enable respectively disable the NMI and to configure the edge refer to *“External Interrupts Edge Detection Configuration” on page 280*.

5.3 Maskable Interrupts

Maskable interrupt requests can be masked by interrupt control registers. This microcontroller has up to 116 maskable interrupt sources.

If two or more maskable interrupt requests are generated at the same time, they are acknowledged according to the default priority. In addition to the default priority, eight levels of priorities can be specified by using the interrupt control registers (programmable priority control).

When an interrupt request has been acknowledged, the acknowledgement of other maskable interrupt requests is disabled and the interrupt disabled (DI) status is set.

When the EI instruction is executed in an interrupt processing routine, the interrupt enabled (EI) status is set, which enables servicing of interrupts having a higher priority than the interrupt request in progress (specified by the interrupt control register). Note that only interrupts with a higher priority will have this capability; interrupts with the same priority level cannot be nested.

However, if multiple interrupts are executed, the following processing is necessary.

1. Save EIPC and EIPSW in memory or a general-purpose register before executing the EI instruction.
2. Execute the DI instruction before executing the RETI instruction, then reset EIPC and EIPSW with the values saved in (1).

5.3.1 Operation

If a maskable interrupt occurs, the CPU performs the following processing, and transfers control to a handler routine:

1. Saves the restored PC to EIPC.
2. Saves the current PSW to EIPSW.
3. Writes an exception code to the lower halfword of ECR (EICC).
4. Sets the ID bit of the PSW and clears the EP bit.
5. Sets the handler address corresponding to each interrupt to the PC, and transfers control.

The processing configuration of a maskable interrupt is shown in *Figure 5-5*.

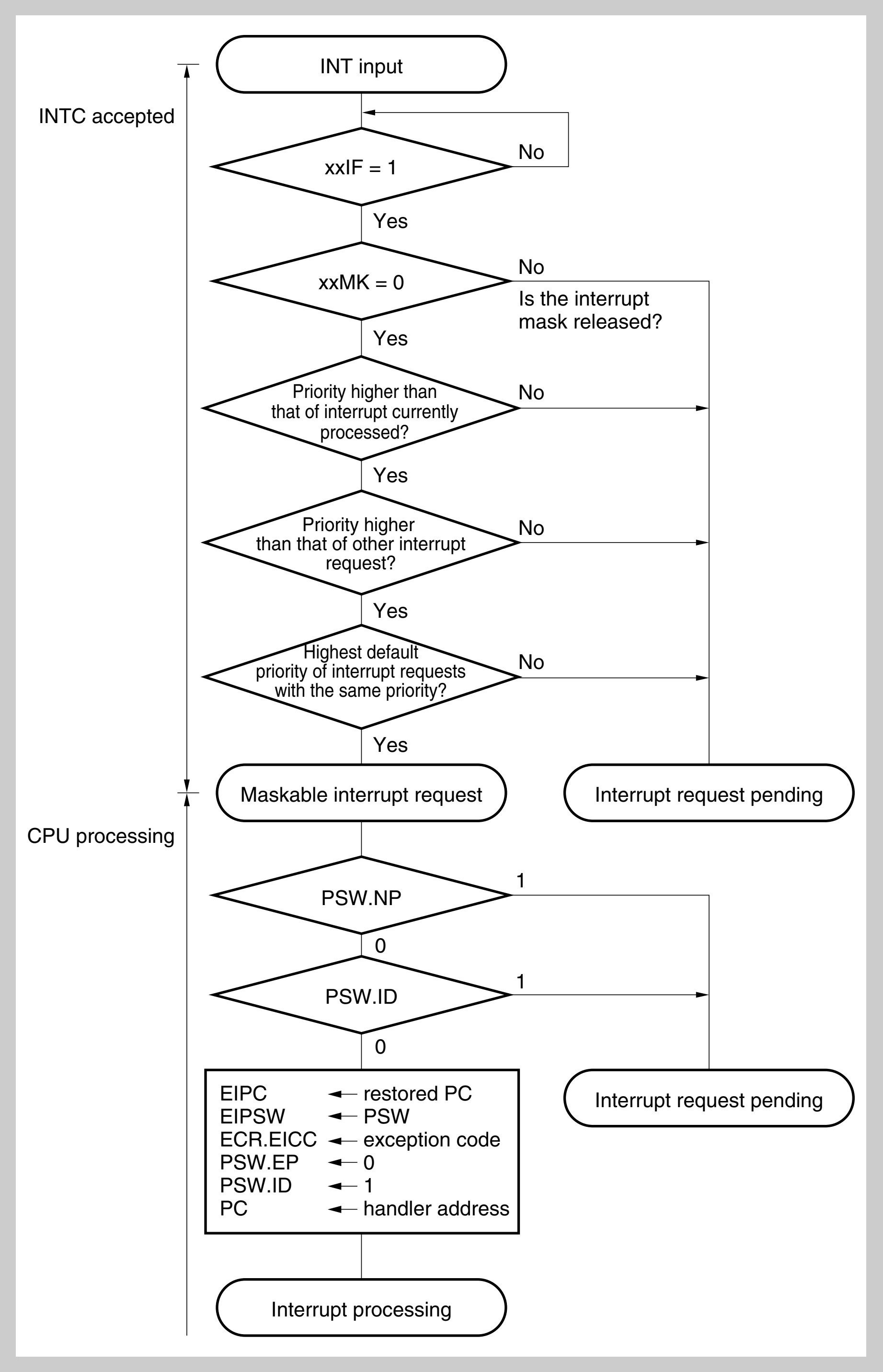

INTC accepted
INT input
xxIF = 1
No
Yes
xxMK = 0
No
Is the interrupt mask released?
Yes
Priority higher than that of interrupt currently processed?
No
Yes
Priority higher than that of other interrupt request?
No
Yes
Highest default priority of interrupt requests with the same priority?
No
Yes
Maskable interrupt request
Interrupt request pending
CPU processing
PSW.NP
1
0
PSW.ID
1
0
EIPC ← restored PC
EIPSW ← PSW
ECR.EICC ← exception code
PSW.EP ← 0
PSW.ID ← 1
PC ← handler address
Interrupt request pending
Interrupt processing

Figure 5-5 Maskable interrupt processing

Note For the ISPR register, see *"ISPR - In-service priority register" on page 279*.

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

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

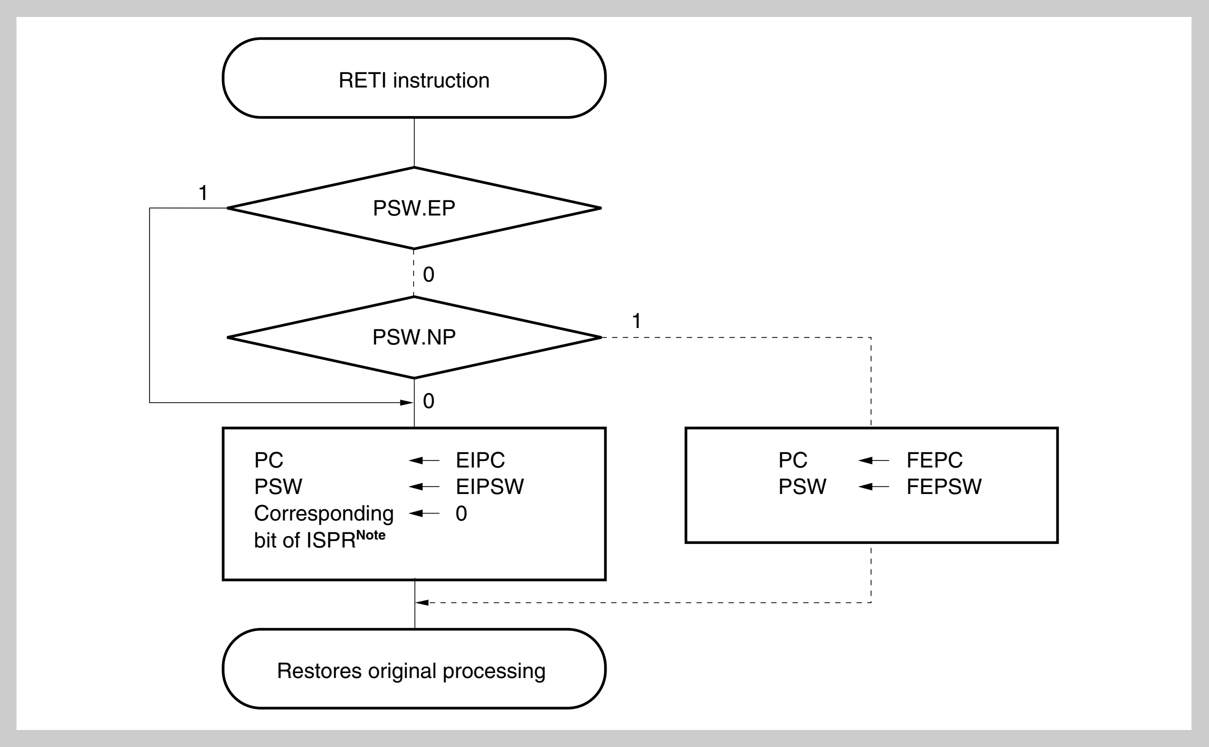


Figure 5-6 RETI instruction processing

Note
1. For the ISPR register, see *“ISPR - In-service priority register” on page 279.*
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.

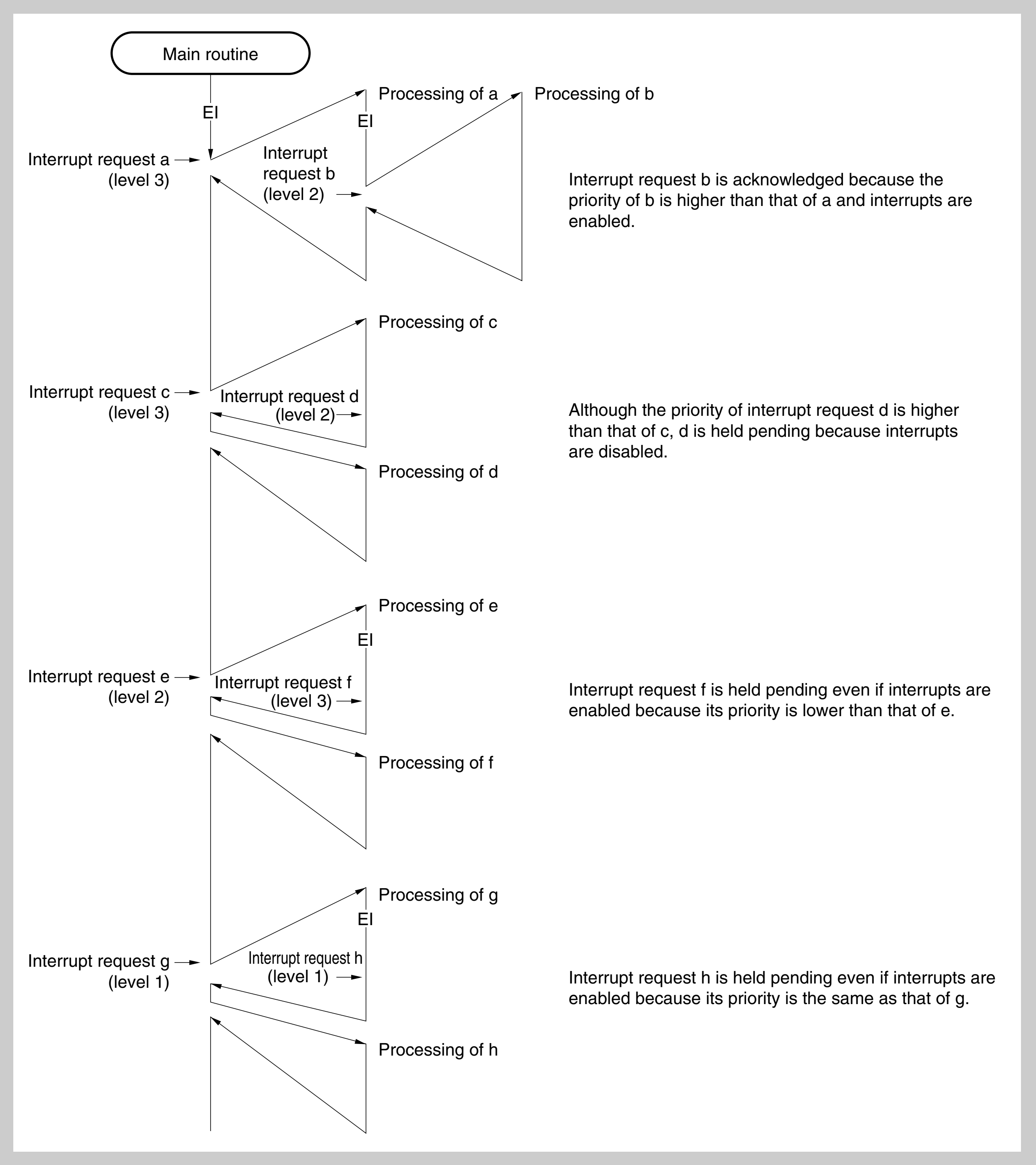

Main routine
EI
Processing of a
Processing of b
EI
Interrupt request a (level 3)
Interrupt request b (level 2)
Interrupt request b is acknowledged because the priority of b is higher than that of a and interrupts are enabled.
Processing of c
Interrupt request c (level 3)
Interrupt request d (level 2)
Although the priority of interrupt request d is higher than that of c, d is held pending because interrupts are disabled.
Processing of d
Processing of e
EI
Interrupt request e (level 2)
Interrupt request f (level 3)
Interrupt request f is held pending even if interrupts are enabled because its priority is lower than that of e.
Processing of f
Processing of g
EI
Interrupt request g (level 1)
Interrupt request h (level 1)
Interrupt request h is held pending even if interrupts are enabled because its priority is the same as that of g.
Processing of h

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

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

Note

1. <a> to <u> in the figure are the temporary names of interrupt requests shown for the sake of explanation.
2. The default priority in the figure indicates the relative priority between two interrupt requests.

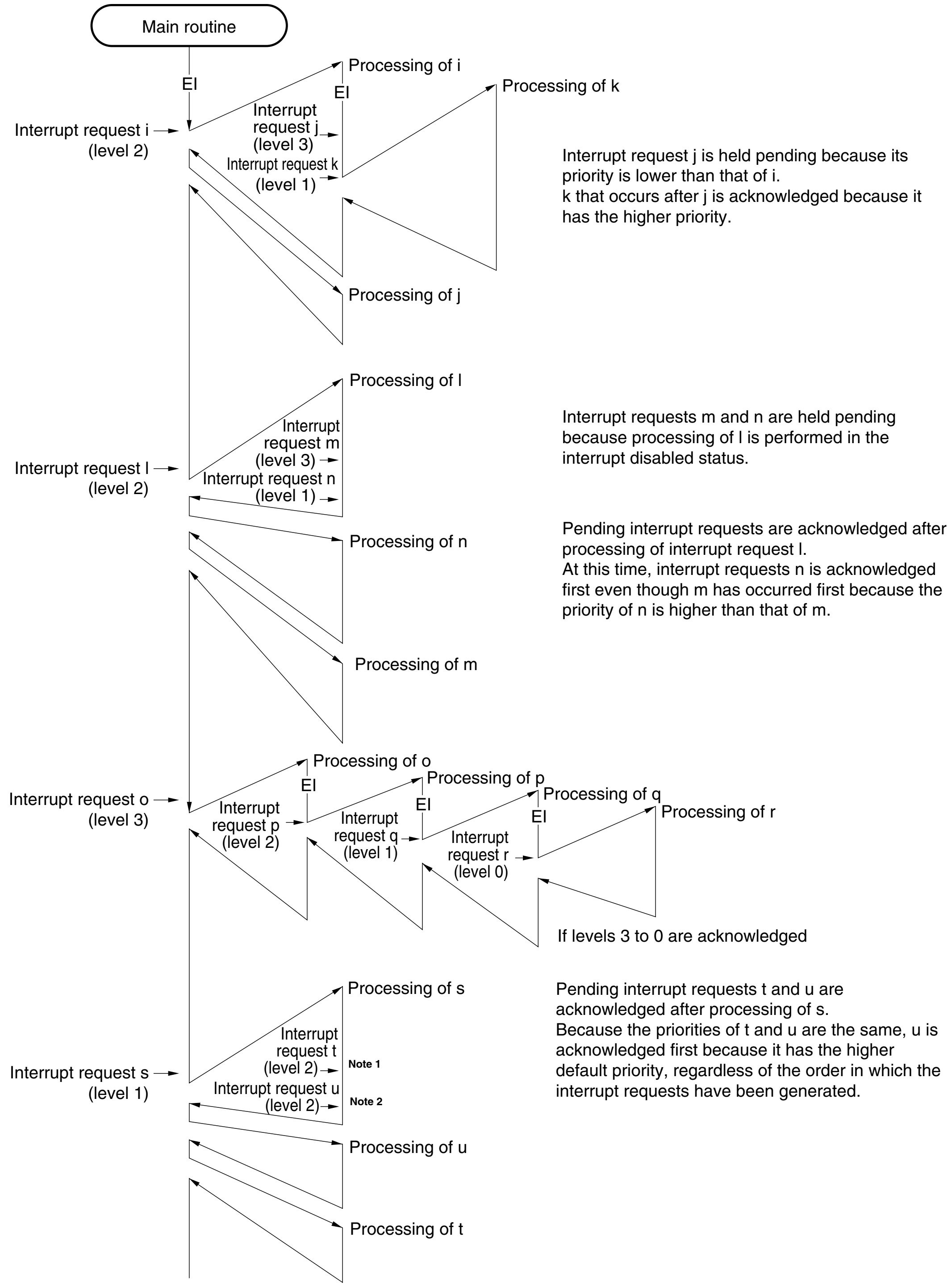

Main routine
EI
Processing of i
EI
Processing of k
Interrupt request i
(level 2)
Interrupt request j
(level 3)
Interrupt request k
(level 1)
Interrupt request j is held pending because its priority is lower than that of i.
k that occurs after j is acknowledged because it has the higher priority.
Processing of j
Processing of l
Interrupt request m
(level 3)
Interrupt request l
(level 2)
Interrupt request n
(level 1)
Interrupt requests m and n are held pending because processing of l is performed in the interrupt disabled status.
Processing of n
Pending interrupt requests are acknowledged after processing of interrupt request l.
At this time, interrupt requests n is acknowledged first even though m has occurred first because the priority of n is higher than that of m.
Processing of m
Processing of o
EI
Processing of p
EI
Processing of q
EI
Processing of r
Interrupt request o
(level 3)
Interrupt request p
(level 2)
Interrupt request q
(level 1)
Interrupt request r
(level 0)
If levels 3 to 0 are acknowledged
Processing of s
Interrupt request t
(level 2)
Note 1
Interrupt request s
(level 1)
Interrupt request u
(level 2)
Note 2
Pending interrupt requests t and u are acknowledged after processing of s.
Because the priorities of t and u are the same, u is acknowledged first because it has the higher default priority, regardless of the order in which the interrupt requests have been generated.
Processing of u
Processing of t

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. Lower default priority
2. Higher default priority

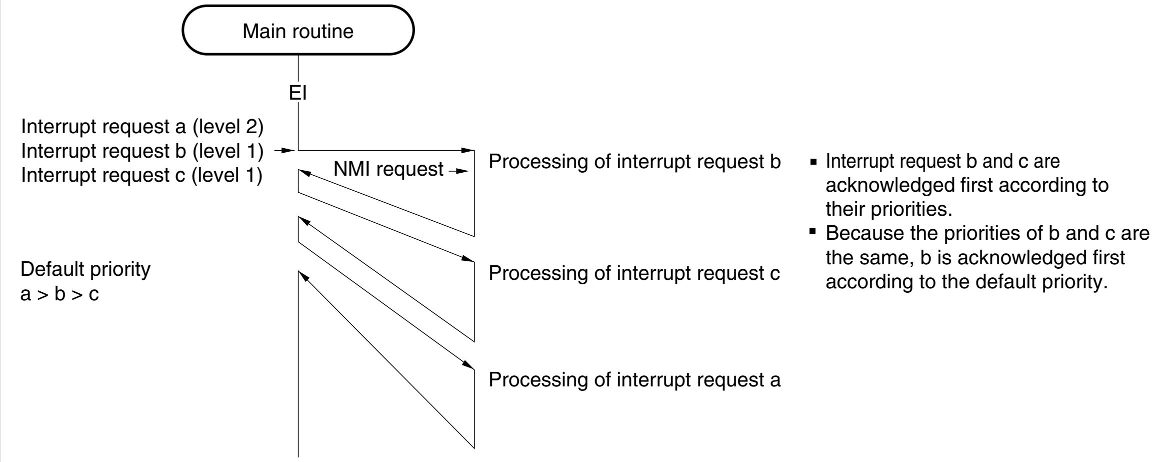

Main routine
EI
Interrupt request a (level 2)
Interrupt request b (level 1)
Interrupt request c (level 1)
NMI request
Processing of interrupt request b
Processing of interrupt request c
Processing of interrupt request a
Default priority
a > b > c
▪ Interrupt request b and c are acknowledged first according to their priorities.
▪ Because the priorities of b and c are the same, b is acknowledged first according to the default priority.

Figure 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 interrupt control registers

An interrupt control register is assigned to each interrupt request (maskable interrupt) and sets the control conditions for each maskable interrupt request.

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

Address FFFF F110$_{H}$ to FFFF F1F8$_{H}$

Initial Value 47$_{H}$. The register is initialized by any reset

	7	6	5	4	3	2	1	0
xxICn	xxIFn	xxMKn	0	0	0	xxPR2	xxPR1	xxPR0

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>7</td><td>xxIFn</td><td>This is an interrupt request flag.
0: Interrupt request not issued
1: Interrupt request issued
The flag xxIFn is reset automatically by the hardware if an interrupt request is acknowledged.</td></tr>
<tr><td>6</td><td>xxMKn</td><td>This is an interrupt mask flag.
0: Enables interrupt processing
1: Disables interrupt processing (pending)</td></tr>
<tr><td>2 to 0</td><td>xxPR2 to xxPR0</td><td>8 levels of priority order are specified for each interrupt.
<table>
<tr><th>xxPR2</th><th>xxPR1</th><th>xxPR0</th><th>Interrupt priority specification bit</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>Specifies level 0 (highest)</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Specifies level 1</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>Specifies level 2</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>Specifies level 3</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>Specifies level 4</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Specifies level 5</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>Specifies level 6</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>Specifies level 7 (lowest)</td></tr>
</table></td></tr>
</table>

Note xx: identification name of each peripheral unit (LVIL, LVIH, P, TAB0OV-TAB2OV, TAB0CC-TAB2CC, TAA0OV-TAA7OV, TAA0CC-TAA7CC, TM0EQ, CB0R-CB3R, CB0T-CB3T, UD0S-UD7S, UD0R-UD7R, UD0T-UD7T, IIC0, AD, AD1, C0ERR-C4ERR, C0WUP-C4WUP, C0REC-C4REC, C0TRX-C4TRX, DMA, KR, WTI, WT, FL)

Caution Do not modify an interrupt control register by a read-modify-write operation when the interrupt is active and its occurrence cannot be excluded, or ensure that bit maniplulation is done by SET1 or CLR1 instructions.
To mask/unmask the interrupts during operating it is recommended to use interrupt mask registers (refer to *"IMRm - Interrupt mask registers" on page 275*) rather than the interrupt control registers.

The address and the availability of each interrupt control register for each device is shown in the following table.

Note The symbols used in the table mean:
√ : register available for the device
–: register not available for the device

Table 5-5 V850ES/Fx3 addresses of interrupt control registers (1/4)

Address	Register	V850ES/ FE3/FF3	V850ES/FG3		V850ES/FJ3			V850ES/ FK3
			µPD70F3374 µPD70F3375	µPD70F3376A µPD70F3377A	µPD70F3378	µPD70F3379 µPD70F3380	µPD70F3381 µPD70F3382	
FFFFF110$_H$	LVILIC	√	√	√	√	√	√	√
FFFFF112$_H$	LVIHIC	√	√	√	√	√	√	√
FFFFF114$_H$	PIC0	√	√	√	√	√	√	√
FFFFF116$_H$	PIC1	√	√	√	√	√	√	√
FFFFF118$_H$	PIC2	√	√	√	√	√	√	√
FFFFF11A$_H$	PIC3	√	√	√	√	√	√	√
FFFFF11C$_H$	PIC4	√	√	√	√	√	√	√
FFFFF11E$_H$	PIC5	√	√	√	√	√	√	√
FFFFF120$_H$	PIC6	√	√	√	√	√	√	√
FFFFF122$_H$	PIC7	√	√	√	√	√	√	√
FFFFF124$_H$	TAB0OVIC	√	√	√	√	√	√	√
FFFFF126$_H$	TAB0CCIC0	√	√	√	√	√	√	√
FFFFF128$_H$	TAB0CCIC1	√	√	√	√	√	√	√
FFFFF12A$_H$	TAB0CCIC2	√	√	√	√	√	√	√
FFFFF12C$_H$	TAB0CCIC3	√	√	√	√	√	√	√
FFFFF12E$_H$	TAA0OVIC	√	√	√	√	√	√	√
FFFFF130$_H$	TAA0CCIC0	√	√	√	√	√	√	√
FFFFF132$_H$	TAA0CCIC1	√	√	√	√	√	√	√
FFFFF134$_H$	TAA1OVIC	√	√	√	√	√	√	√
FFFFF136$_H$	TAA1CCIC0	√	√	√	√	√	√	√
FFFFF138$_H$	TAA1CCIC1	√	√	√	√	√	√	√
FFFFF13A$_H$	TAA2OVIC	√	√	√	√	√	√	√
FFFFF13C$_H$	TAA2CCIC0	√	√	√	√	√	√	√
FFFFF13E$_H$	TAA2CCIC1	√	√	√	√	√	√	√
FFFFF140$_H$	TAA3OVIC	√	√	√	√	√	√	√
FFFFF142$_H$	TAA3CCIC0	√	√	√	√	√	√	√
FFFFF144$_H$	TAA3CCIC1	√	√	√	√	√	√	√
FFFFF146$_H$	TAA4OVIC	√	√	√	√	√	√	√
FFFFF148$_H$	TAA4CCIC0	√	√	√	√	√	√	√
FFFFF14A$_H$	TAA4CCIC1	√	√	√	√	√	√	√
FFFFF14C$_H$	TM0EQIC0	√	√	√	√	√	√	√
FFFFF14E$_H$	CB0RIC	√	√	√	√	√	√	√
FFFFF150$_H$	CB0TIC	√	√	√	√	√	√	√
FFFFF152$_H$	CB1RIC	√	√	√	√	√	√	√
FFFFF154$_H$	CB1TIC	√	√	√	√	√	√	√

Table 5-5 V850ES/Fx3 addresses of interrupt control registers (2/4)

Address	Register	V850ES/ FE3/FF3	V850ES/FG3		V850ES/FJ3			V850ES/ FK3
			μPD70F3374 μPD70F3375	μPD70F3376A μPD70F3377A	μPD70F3378	μPD70F3379 μPD70F3380	μPD70F3381 μPD70F3382	
FFFFF156$_{H}$	UD0SIC	√	√	√	√	√	√	√
FFFFF158$_{H}$	UD0RIC	√	√	√	√	√	√	√
FFFFF15A$_{H}$	UD0TIC	√	√	√	√	√	√	√
FFFFF15C$_{H}$	UD1SIC	√	√	√	√	√	√	√
FFFFF15E$_{H}$	UD1RIC	√	√	√	√	√	√	√
FFFFF160$_{H}$	UD1TIC	√	√	√	√	√	√	√
FFFFF162$_{H}$	IIC0IC	√	√	√	√	√	√	√
	UD4SIC	–	–	√	–	√	√	√
FFFFF164$_{H}$	ADIC	√	√	√	√	√	√	√
FFFFF166$_{H}$	C0ERRIC	√	√	√	√	√	√	√
FFFFF168$_{H}$	C0WUPIC	√	√	√	√	√	√	√
FFFFF16A$_{H}$	C0RECIC	√	√	√	√	√	√	√
FFFFF16C$_{H}$	C0TRXIC	√	√	√	√	√	√	√
FFFFF16E$_{H}$	DMAIC0	√	√	√	√	√	√	√
FFFFF170$_{H}$	DMAIC1	√	√	√	√	√	√	√
FFFFF172$_{H}$	DMAIC2	√	√	√	√	√	√	√
FFFFF174$_{H}$	DMAIC3	√	√	√	√	√	√	√
FFFFF176$_{H}$	KRIC	√	√	√	√	√	√	√
FFFFF178$_{H}$	WTIIC	√	√	√	√	√	√	√
FFFFF17A$_{H}$	WTIC	√	√	√	√	√	√	√
FFFFF17E$_{H}$	FLIC	√	√	√	√	√	√	√
FFFFF180$_{H}$	PIC8	–	√	√	√	√	√	√
FFFFF182$_{H}$	PIC9	–	√	√	√	√	√	√
FFFFF184$_{H}$	PIC10	–	√	√	√	√	√	√
FFFFF186$_{H}$	TAB1OVIC	–	√	√	√	√	√	√
FFFFF188$_{H}$	TAB1CCIC0	–	√	√	√	√	√	√
FFFFF18A$_{H}$	TAB1CCIC1	–	√	√	√	√	√	√
FFFFF18C$_{H}$	TAB1CCIC2	–	√	√	√	√	√	√
FFFFF18E$_{H}$	TAB1CCIC3	–	√	√	√	√	√	√
FFFFF190$_{H}$	UD2SIC	–	√	√	√	√	√	√
FFFFF192$_{H}$	UD2RIC	–	√	√	√	√	√	√
FFFFF194$_{H}$	UD2TIC	–	√	√	√	√	√	√
FFFFF196$_{H}$	C1ERRIC	–	√	√	√	√	√	√
FFFFF198$_{H}$	C1WUPIC	–	√	√	√	√	√	√
FFFFF19A$_{H}$	C1RECIC	–	√	√	√	√	√	√
FFFFF19C$_{H}$	C1TRXIC	–	√	√	√	√	√	√
FFFFF19E$_{H}$	PIC11	–	–	–	√	√	√	√
FFFFF1A0$_{H}$	PIC12	–	–	–	√	√	√	√
FFFFF1A2$_{H}$	PIC13	–	–	–	√	√	√	√
FFFFF1A4$_{H}$	PIC14	–	–	√	√	√	√	√

Table 5-5 V850ES/Fx3 addresses of interrupt control registers (3/4)

Address	Register	V850ES/ FE3/FF3	V850ES/FG3		V850ES/FJ3			V850ES/ FK3
			µPD70F3374 µPD70F3375	µPD70F3376A µPD70F3377A	µPD70F3378	µPD70F3379 µPD70F3380	µPD70F3381 µPD70F3382	
FFFFF1A6$_H$	UD3SIC	–	–	√	–	√	√	√
FFFFF1A8$_H$	UD3RIC	–	–	√	–	√	√	√
FFFFF1AA$_H$	UD3TIC	–	–	√	–	√	√	√
FFFFF1AC$_H$	UD4RIC	–	–	√	–	√	√	√
FFFFF1AE$_H$	UD4TIC	–	–	√	–	√	√	√
FFFFF1B0$_H$	TAB2OVIC	–	–	–	√	√	√	√
FFFFF1B2$_H$	TAB2CCIC0	–	–	–	√	√	√	√
FFFFF1B4$_H$	TAB2CCIC1	–	–	–	√	√	√	√
FFFFF1B6$_H$	TAB2CCIC2	–	–	–	√	√	√	√
FFFFF1B8$_H$	TAB2CCIC3	–	–	–	√	√	√	√
FFFFF1BA$_H$	UD5SIC	–	–	–	–	√	√	√
FFFFF1BC$_H$	CB2RIC	–	–	–	√	√	√	√
	UD5RIC	–	–	–	–	√	√	√
FFFFF1BE$_H$	CB2TIC	–	–	–	√	√	√	√
	UD5TIC	–	–	–	–	√	√	√
FFFFF1C0$_H$	C2ERRIC	–	–	–	–	√	√	√
FFFFF1C2$_H$	C2WUPIC	–	–	–	–	√	√	√
FFFFF1C4$_H$	C2RECIC	–	–	–	–	√	√	√
FFFFF1C6$_H$	C2TRXIC	–	–	–	–	√	√	√
FFFFF1C8$_H$	C3ERRIC	–	–	–	–	√	√	√
FFFFF1CA$_H$	C3WUPIC	–	–	–	–	√	√	√
FFFFF1CC$_H$	C3RECIC	–	–	–	–	√	√	√
FFFFF1CE$_H$	C3TRXIC	–	–	–	–	√	√	√
FFFFF1D0$_H$	PIC15	–	–	–	–	–	–	√
FFFFF1D2$_H$	TAA5OVIC	–	–	–	–	–	–	√
FFFFF1D4$_H$	TAA5CCIC0	–	–	–	–	–	–	√
FFFFF1D6$_H$	TAA5CCIC1	–	–	–	–	–	–	√
FFFFF1D8$_H$	TAA6OVIC	–	–	–	–	–	–	√
FFFFF1DA$_H$	TAA6CCIC0	–	–	–	–	–	–	√
FFFFF1DC$_H$	TAA6CCIC1	–	–	–	–	–	–	√
FFFFF1DE$_H$	TAA7OVIC	–	–	–	–	–	–	√
FFFFF1E0$_H$	TAA7CCIC0	–	–	–	–	–	–	√
FFFFF1E2$_H$	TAA7CCIC1	–	–	–	–	–	–	√
FFFFF1E4$_H$	UD6SIC	–	–	–	–	–	–	√
FFFFF1E6$_H$	CB3RIC	–	–	–	–	–	√	√
	UD6RIC	–	–	–	–	–	–	√
FFFFF1E8$_H$	CB3TIC	–	–	–	–	–	√	√
	UD6TIC	–	–	–	–	–	–	√
FFFFF1EA$_H$	UD7SIC	–	–	–	–	–	–	√
FFFFF1EC$_H$	UD7RIC	–	–	–	–	–	–	√

Table 5-5 V850ES/Fx3 addresses of interrupt control registers (4/4)

Address	Register	V850ES/ FE3/FF3	V850ES/FG3		V850ES/FJ3			V850ES/ FK3
			µPD70F3374 µPD70F3375	µPD70F3376A µPD70F3377A	µPD70F3378	µPD70F3379 µPD70F3380	µPD70F3381 µPD70F3382	
FFFFF1EE_H	UD7TIC	–	–	–	–	–	–	√
FFFFF1F0_H	AD1IC	–	–	–	–	–	–	√
FFFFF1F2_H	C4ERRIC	–	–	–	–	–	–	√
FFFFF1F4_H	C4WUPIC	–	–	–	–	–	–	√
FFFFF1F6_H	C4RECIC	–	–	–	–	–	–	√
FFFFF1F8_H	C4TRXIC	–	–	–	–	–	–	√

5.3.5 IMRm - Interrupt mask registers

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

The xxMKn bit of the IMRm registers is equivalent to the xxMKn bit of the xxICn register.

- 16 bit IMRm registers are accessible through
  - 16 bit IMRm via the given <Address> and can be read/written in 16-bit units
  - 8 bit IMRmL = IMRm[7:0] registers via the given <Address> and can be read/written in 8- and 1-bit units
  - 8 bit IMRmH = IMRm[15:8] registers via <Address> + 1 and can be read/written in 8- and 1-bit units
- 8 bit IMRm registers are accessible through
  - 8 bit IMRm or IMRmL registers via the given <Address> and can be read/written in 8- and 1-bit units

Caution

1. Mask bits without function, indicated with “1”, must not be altered. Make sure to set them “1” when writing to the register.
2. The device file defines the xxMKn bit of the xxICn register as a reserved word. If a bit is manipulated using the name of xxMKn, the contents of the xxICn register, instead of the IMRm register, are rewritten (as a result, the contents of the IMRm register are also rewritten).

Bit position	Bit name	Function
15 to 0	xxMKn	Interrupt mask flag. 0: Interrupt servicing enabled 1: Interrupt servicing disabled (pending)

xx: identification name of each peripheral unit (see the note in *“xxICn - Maskable interrupt control registers” on page 270*

(1) IMR0 - Interrupt mask register 0

Address FFFF F100$_H$

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

IMR0

15	14	13	12	11	10	9	8
TAA0OVMK	TAB0CCMK3	TAB0CCMK2	TAB0CCMK1	TAB0CCMK0	TAB0OVMK	PMK7	PMK6
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PMK5	PMK4	PMK3	PMK2	PMK1	PMK0	LVIHMK	LVILMK

(2) IMR1 - Interrupt mask register 1

Address FFFF F102$_H$

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

IMR1

15	14	13	12	11	10	9	8
CB0RMK	TM0EQMK0	TAA4CCMK1	TAA4CCMK0	TAA4OVMK	TAA3CCMK1	TAA3CCMK0	TAA3OVMK
7	6	5	4	3	2	1	0
TAA2CCMK1	TAA2CCMK0	TAA2OVMK	TAA1CCMK1	TAA1CCMK0	TAA1OVMK	TAA0CCMK1	TAA0CCMK0

(3) IMR2 - Interrupt mask register 2

Address FFFF F104$_H$

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

- V850ES/FE3
- V850ES/FF3
- µPD70F3374, µPD70F3375 of V850ES/FG3
- µPD70F3378 of V850ES/FJ3

IMR2

15	14	13	12	11	10	9	8
DMAMK0	C0TRXMK	C0RECMK	C0WUPMK	C0ERRMK	ADMK	IIC0MK	UD1TMK
7	6	5	4	3	2	1	0
UD1RMK	UD1SMK	UD0TMK	UD0RMK	UD0SMK	CB1TMK	CB1RMK	CB0TMK

- µPD70F3376A, µPD70F3377A of V850ES/FG3
- µPD70F3379, µPD70F3380, µPD70F3381, µPD70F3382 of V850ES/FJ3
- V850ES/FK3

IMR2

15	14	13	12	11	10	9	8
DMAMK0	C0TRXMK	C0RECMK	C0WUPMK	C0ERRMK	ADMK	IIC0MK UD4SMK	UD1TMK
7	6	5	4	3	2	1	0
UD1RMK	UD1SMK	UD0TMK	UD0RMK	UD0SMK	CB1TMK	CB1RMK	CB0TMK

(4) IMR3 - Interrupt mask register 3

Address FFFF F106$_H$

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

- V850ES/FE3
- V850ES/FF3

IMR3

15	14	13	12	11	10	9	8
1	1	1	1	1	1	1	1
7	6	5	4	3	2	1	0
FLMK	1	WTMK	WTIMK	KRMK	DMAMK3	DMAMK2	DMAMK1

- V850ES/FG3
- V850ES/FJ3
- V850ES/FK3

	15	14	13	12	11	10	9	8
IMR3	TAB1CCMK3	TAB1CCMK2	TAB1CCMK1	TAB1CCMK0	TAB1OVMK	PMK10	PMK9	PMK8
	7	6	5	4	3	2	1	0
	FLMK	1	WTMK	WTIMK	KRMK	DMAMK3	DMAMK2	DMAMK1

(5) IMR4 - Interrupt mask register 4

Address FFFF F108$_H$

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

- µPD70F3374, µPD70F3375 of V850ES/FG3

	15	14	13	12	11	10	9	8
IMR4	1	1	1	1	1	1	1	1
	7	6	5	4	3	2	1	0
	1	C1TRXMK	C1RECMK	C1WUPMK	C1ERRMK	UD2TMK	UD2RMK	UD2SMK

- µPD70F3376A, µPD70F3376A, µPD70F3377A of V850ES/FG3

	15	14	13	12	11	10	9	8
IMR4	UD4TMK	UD4RMK	UD3TMK	UD3RMK	UD3SMK	PMK14	1	1
	7	6	5	4	3	2	1	0
	1	C1TRXMK	C1RECMK	C1WUPMK	C1ERRMK	UD2TMK	UD2RMK	UD2SMK

- µPD70F3378 of V850ES/FJ3

	15	14	13	12	11	10	9	8
IMR4	1	1	1	1	1	PMK14	PMK13	PMK12
	7	6	5	4	3	2	1	0
	PMK11	C1TRXMK	C1RECMK	C1WUPMK	C1ERRMK	UD2TMK	UD2RMK	UD2SMK

- µPD70F3379, µPD70F3380, µPD70F3381, µPD70F3382 of V850ES/FJ3
- V850ES/FK3

	15	14	13	12	11	10	9	8
IMR4	UD4TMK	UD4RMK	UD3TMK	UD3RMK	UD3SMK	PMK14	PMK13	PMK12
	7	6	5	4	3	2	1	0
	PMK11	C1TRXMK	C1RECMK	C1WUPMK	C1ERRMK	UD2TMK	UD2RMK	UD2SMK

(6) IMR5 - Interrupt mask register 5

Address FFFF F10A$_{\text{H}}$

Initial Value FFFF$_{\text{H}}$. The register is initialized by any reset

- µPD70F3378 of V850ES/FJ3

	15	14	13	12	11	10	9	8
IMR5	1	1	1	1	C2TRXMK	C2RECMK	C2WUPMK	C2ERRMK
	7	6	5	4	3	2	1	0
	CB2TMK	CB2RMK	1	TAB2CCMK3	TAB2CCMK2	TAB2CCMK1	TAB2CCMK0	TAB2OVMK

- µPD70F3379, µPD70F3380, µPD70F3381, µPD70F3382 of V850ES/FJ3
- V850ES/FK3

	15	14	13	12	11	10	9	8
IMR5	C3TRXMK	C3RECMK	C3WUPMK	C3ERRMK	C2TRXMK	C2RECMK	C2WUPMK	C2ERRMK
	7	6	5	4	3	2	1	0
	CB2TMK UD5TMK	CB2RMK UD5RMK	UD5SMK	TAB2CCMK3	TAB2CCMK2	TAB2CCMK1	TAB2CCMK0	TAB2OVMK

(7) IMR6 - Interrupt mask register 6

Address FFFF F10C$_{\text{H}}$

Initial Value FFFF$_{\text{H}}$. The register is initialized by any reset

- µPD70F3381, µPD70F3382 of V850ES/FJ3

	15	14	13	12	11	10	9	8
IMR6	1	1	1	CB3TMK	CB3RMK	1	1	1
	7	6	5	4	3	2	1	0
	1	1	1	1	1	1	1	1

- V850ES/FK3

	15	14	13	12	11	10	9	8
IMR6	UD7TMK	UD7RMK	UD7SMK	CB3TMK UD6TMK	CB3RMK UD6RMK	UD6SMK	TAA7CCMK1	TAA7CCMK0
	7	6	5	4	3	2	1	0
	TAA7OVMK	TAA6CCMK1	TAA6CCMK0	TAA6OVMK	TAA5CCMK1	TAA5CCMK0	TAA5OVMK	PMK15

(8) IMR7 - Interrupt mask register 7

Address FFFF F10E$_{\text{H}}$

Initial Value FFFF$_{\text{H}}$. The register is initialized by any reset

- V850ES/FK3 only

	15	14	13	12	11	10	9	8
IMR7	1	1	1	1	1	1	1	1
	7	6	5	4	3	2	1	0
	1	1	1	C4TRXMK	C4RECMK	C4WUPMK	C4ERRMK	AD1MK

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.

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

Address FFFFF1FA$_H$

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

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

Bit position	Bit name	Function
7 to 0	ISPR7 to ISPR0	Indicates priority of interrupt currently acknowledged 0: Interrupt request with priority n not acknowledged 1: Interrupt request with priority n acknowledged

Note n = 0 to 7 (priority level)

Caution If an interrupt is acknowledged while the ISPR register is being read in the interrupt enabled (EI) state, the value of the ISPR register after the bits of the register have been set by acknowledging the interrupt may be read.
To accurately read the value of the ISPR register before an interrupt is acknowledged, read the register while interrupts are disabled (DI).

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 00000020_H

	31																								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

Bit position	Bit name	Function
5	ID	Indicates whether maskable interrupt processing is enabled or disabled. 0: Maskable interrupt request acknowledgement enabled 1: Maskable interrupt request acknowledgement disabled (pending) This bit is set to 1 by the DI instruction and reset to 0 by the EI instruction. Its value is also modified by the RETI instruction or LDSR instruction when referencing to PSW. Non-maskable interrupt requests and exceptions are acknowledged regardless of this flag. when a maskable interrupt is acknowledged, the ID flag is automatically set to 1 by hardware. The interrupt request generated during the acknowledgement disabled period (ID = 1) is acknowledged when the PIFn bit of PICn register is set to 1, and the ID flag is reset to 0.

5.3.8 External maskable interrupts

This microcontroller provides maskable external interrupts INTPn with the following features:

- Analog input filter (refer to *"Analog filtered inputs" on page 144*)
- Digital input filter for INTP3 (refer to *"Digitally filtered inputs" on page 145*)
- Interrupt detection selectable for each interrupt input:
  - Rising edge
  - Falling edge
  - Both edges: rising and falling edge

For configuration of the external interrupt events refer to *"External Interrupts Edge Detection Configuration" on page 280*.

5.4 External Interrupts Edge Detection Configuration

The microcontroller provides the maskable external interrupts INTPn and one non-maskable interrupt (NMI).

INTPn and NMI can be configured to generate interrupts upon rising, falling or both edges. Two register sets are provided to specify edges and levels for each external interrupt.

INTRm The INTRm registers specify the rising edge for edge detection of corresponding external interrupt signals.

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

Bit position	Bit name	Function
15 to 0	INTRm[15:0]	Specifies the edge detection for external interrupt signals 0: no detection at rising edge 1: detection at rising edge

INTFm The INTFm registers specify the falling edge for edge detection of corresponding external interrupt signals.

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

Bit position	Bit name	Function
15 to 0	INTFm[15:0]	Specifies the edge detection for external interrupt signals 0: no detection at falling edge 1: detection at falling edge

Caution When the function of the dedicated pin is changed from the external interrupt function (alternate function) to the port function, an edge may be detected. Therefore, first clear INTRm.INTRmk / INTFm.INTFmk (k = 0 to 15) to 0, and then set the port mode.

(1) INTR0/INTF0 - External interrupt edge specification register 0

Address FFFFFC20$_H$

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

INTR0

7	6	5	4	3	2	1	0
0	INTR06	INTR05	INTR04	INTR03	INTR02	0	0
	INTP3	INTP2	INTP1	INTP0	NMI		

Address FFFFFC00$_H$

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

INTF0

7	6	5	4	3	2	1	0
0	INTF06	INTF05	INTF04	INTF03	INTF02	0	0
	INTP3	INTP2	INTP1	INTP0	NMI		

(2) INTR1/INTF1 - External interrupt edge specification register 1

Address FFFFFC22$_H$

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

	7	6	5	4	3	2	1	0
INTR1	0	0	0	0	0	0	INTR11	INTR10
							INTP10	INTP9

Address FFFFFC02$_H$

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

	7	6	5	4	3	2	1	0
INTF1	0	0	0	0	0	0	INTF11	INTF10
							INTP10	INTP9

(3) INTR3/INTF3 - External interrupt edge specification register 3

- V850ES/FE3
- V850ES/FF3

Address FFFFFC26$_H$

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

	7	6	5	4	3	2	1	0
INTR3L	0	0	0	0	0	0	INTR31	0
							INTP7	

Address FFFFFC06$_H$

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

	7	6	5	4	3	2	1	0
INTF3L	0	0	0	0	0	0	INTF31	0
							INTP7	

- V850ES/FG3
- V850ES/FJ3
- V850ES/FK3

Address FFFFFC26$_H$

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

INTR3[a]

15	14	13	12	11	10	9	8
0	0	0	0	0	0	INTR39	0
						INTP8	

7	6	5	4	3	2	1	0
0	0	0	0	0	0	INTR31	0
						INTP7	

a) Both bytes of this 16-bit register can also be accessed bytewise with
– INTR3L = INTR3[7:0] under the address FFFF FC26$_H$
– INTR3H = INTR3[15:8] under the address FFFF FC27$_H$

Address FFFFFC06$_H$

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

INTF3[a]

15	14	13	12	11	10	9	8
0	0	0	0	0	0	INTF39	0
						INTP8	

7	6	5	4	3	2	1	0
0	0	0	0	0	0	INTF31	0
						INTP7	

a) Both bytes of this 16-bit register can also be accessed bytewise with
– INTF3L = INTF3[7:0] under the address FFFF FC06$_H$
– INTF3H = INTF3[15:8] under the address FFFF FC07$_H$

(4) INTR4/INTF4 - External interrupt edge specification register 4

- µPD70F3376A, µPD70F3377A of V850ES/FG3
- µPD70F3379, µPD70F3380, µPD70F3381, µPD70F3382 of V850ES/FJ3
- V850ES/FK3

Address FFFFFC28$_H$

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

INTR4

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	INTR40
							INTP14

Address FFFFFC08$_H$

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

INTF4

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	INTF40
							INTP14

(5) INTR6/INTF6 - External interrupt edge specification register 6

- V850ES/FJ3

Address FFFFFC2C$_H$

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

	7	6	5	4	3	2	1	0
INTR6L	0	0	0	0	0	INTR62	INTR61	INTR60
						INTP13	INTP12	INTP11

Address FFFFFC0C$_H$

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

	7	6	5	4	3	2	1	0
INTF6L	0	0	0	0	0	INTF62	INTF61	INTF60
						INTP13	INTP12	INTP11

- V850ES/FK3

Address FFFFFC2C$_H$

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

	15	14	13	12	11	10	9	8
INTR6[a]	INTR615	0	0	0	0	0	0	0
	INTP15							

	7	6	5	4	3	2	1	0
	0	0	0	0	0	INTR62	INTR61	INTR60
						INTP13	INTP12	INTP11

[a] Both bytes of this 16-bit register can also be accessed bytewise with
– INTR6L = INTR6[7:0] under the address FFFF FC2C$_H$
– INTR6H = INTR6[15:8] under the address FFFF FC2D$_H$

Address FFFFFC0C$_H$

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

	15	14	13	12	11	10	9	8
INTF6[a]	INTF615	0	0	0	0	0	0	0
	INTP15							

	7	6	5	4	3	2	1	0
	0	0	0	0	0	INTF62	INTF61	INTF60
						INTP13	INTP12	INTP11

[a] Both bytes of this 16-bit register can also be accessed bytewise with
– INTF6L = INTF6[7:0] under the address FFFF FC0C$_H$
– INTF6H = INTF6[15:8] under the address FFFF FC0D$_H$

(6) INTR8/INTF8 - External interrupt edge specification register 8

- V850ES/FJ3
- V850ES/FK3

Address FFFFFC30$_H$

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

	7	6	5	4	3	2	1	0
INTR8	0	0	0	0	0	0	0	INTR80
								INTP14

Address FFFFFC10$_H$

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

	7	6	5	4	3	2	1	0
INTF8	0	0	0	0	0	0	0	INTF80
								INTP14

(7) INTR9/INTF9 - External interrupt edge specification register 9

Address FFFFFC33$_H$

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

	7	6	5	4	3	2	1	0
INTR9H	INTR915	INTR914	INTR913	0	0	0	0	0
	INTP6	INTP5	INTP4					

Address FFFFFC13$_H$

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

	7	6	5	4	3	2	1	0
INTF9H	INTF915	INTF914	INTF913	0	0	0	0	0
	INTP6	INTP5	INTP4					

5.5 Software Exception

A software exception is generated when the CPU executes the TRAP instruction, and can be always acknowledged.

5.5.1 Operation

If a software exception occurs, the CPU performs the following processing, and transfers control to the handler routine:

1. Saves the restored PC to EIPC.
2. Saves the current PSW to EIPSW.
3. Writes an exception code to the lower 16 bits (EICC) of ECR (interrupt source).
4. Sets the EP and ID bits of the PSW.
5. Sets the handler address (00000040_H or 00000050_H) corresponding to the software exception to the PC, and transfers control.

Figure 5-10 illustrates the processing of a software exception.

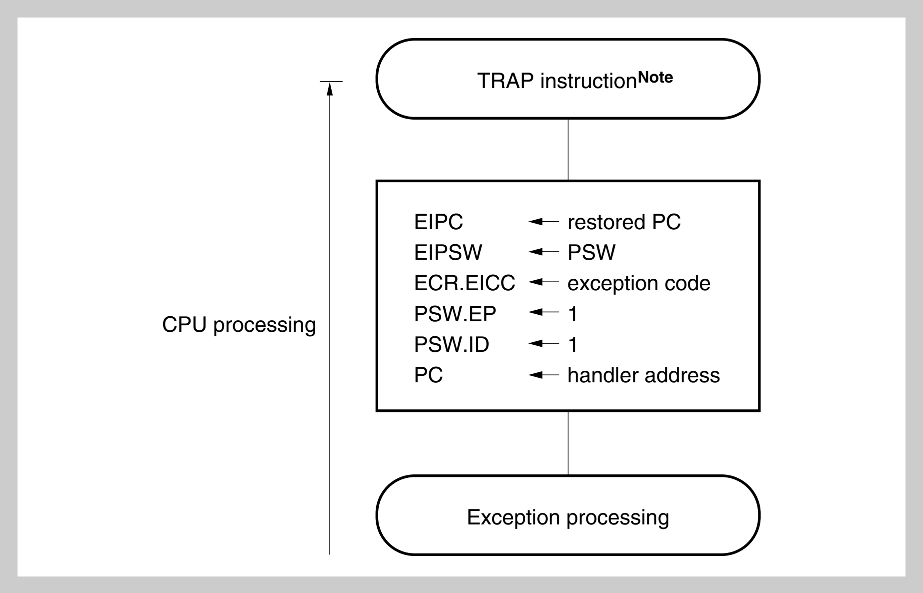


Figure 5-10 Software exception processing

Note TRAP Instruction Format: TRAP vector (the vector is a value from 0 to 1FH.)

The handler address is determined by the TRAP instruction's operand (vector). If the vector is 0 to $0F_H$, it becomes 00000040_H, and if the vector is 10_H to $1F_H$, it becomes 00000050_H.

5.5.2 Restore

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

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

1. Loads the restored PC and PSW from EIPC and EIPSW because the EP bit of the PSW is 1.
2. Transfers control to the address of the restored PC and PSW.

Figure 5-11 illustrates the processing of the RETI instruction.

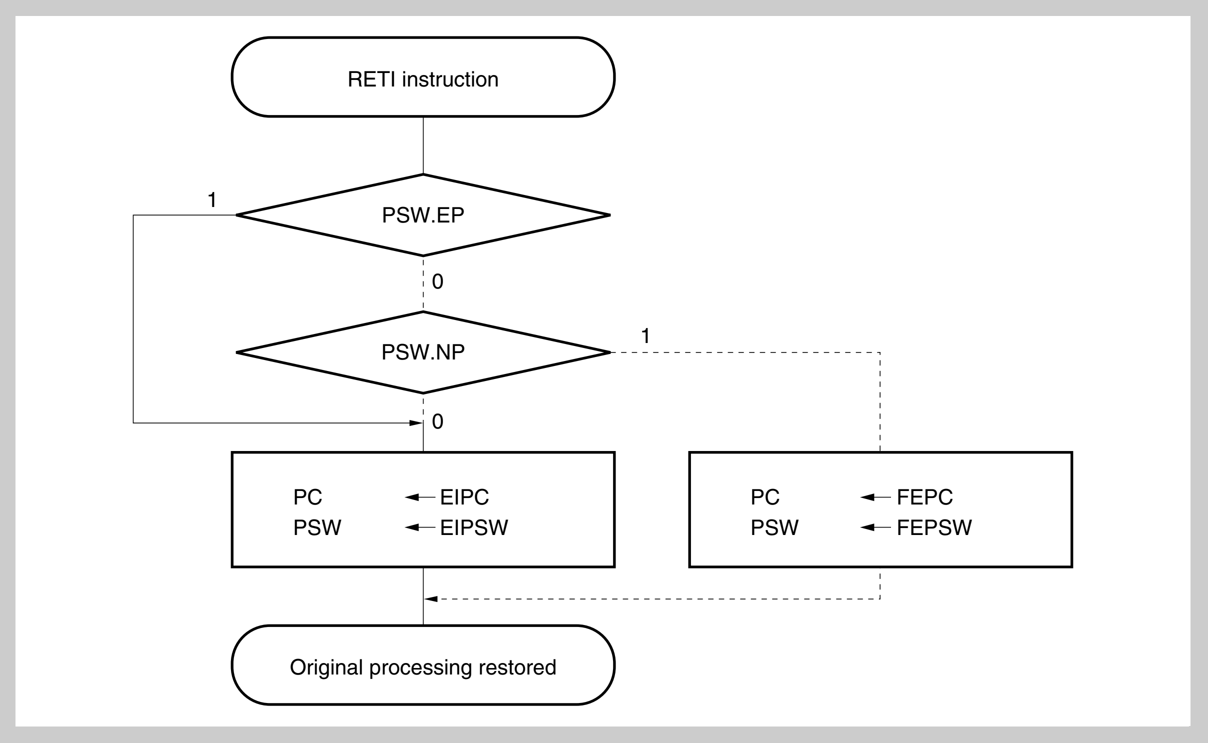


Figure 5-11 RETI instruction processing

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

Note The solid lines show the CPU processing flow.

5.5.3 Exception status flag (EP)

The EP flag is bit 6 of PSW, and is a status flag used to indicate that exception processing is in progress. It is set when an exception occurs.

Initial Value 00000020_H

	31																								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

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.

5.6 Exception Trap

An exception trap is an interrupt that is requested when an illegal execution of an instruction takes place. For this microcontroller, an illegal opcode exception (ILGOP: Illegal Opcode Trap) is considered as an exception trap.

5.6.1 Illegal opcode definition

The illegal instruction has an opcode (bits 10 to 5) of 111111_B, a sub-opcode (bits 23 to 26) of 0111_B to 1111_B, and a sub-opcode (bit 16) of 0_B. An exception trap is generated when an instruction applicable to this illegal instruction is executed.

15 11	10 5	4 0	31 27	26 23	22	16
x x x x x	1 1 1 1 1 1	x x x x x	x x x x x	0 1 1 1 to 1 1 1 1	x x x x x x	0

Note ×: Arbitrary

(1) Operation

If an exception trap occurs, the CPU performs the following processing, and transfers control to the handler routine:

1. Saves the restored PC to DBPC.
2. Saves the current PSW to DBPSW.
3. Sets the NP, EP, and ID bits of the PSW.
4. Sets the handler address (00000060_H) corresponding to the exception trap to the PC, and transfers control.

Figure 5-12 illustrates the processing of the exception trap.

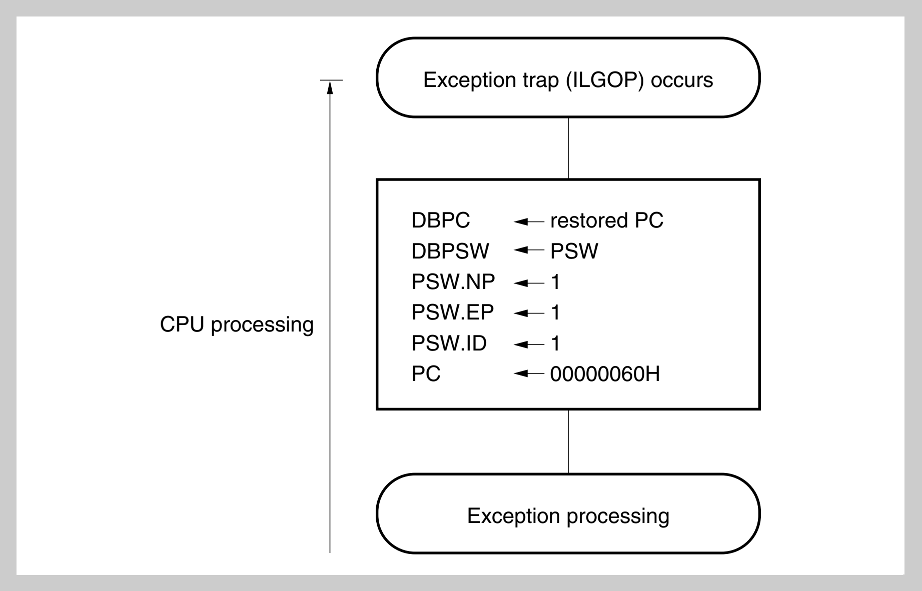


Figure 5-12 Exception trap processing

(2) Restore

Recovery from an exception trap is carried out by the DBRET instruction. By executing the DBRET instruction, the CPU carries out the following processing and controls the address of the restored PC.

1. Loads the restored PC and PSW from DBPC and DBPSW.
2. Transfers control to the address indicated by the restored PC and PSW.

Figure 5-13 illustrates the restore processing from an exception trap.

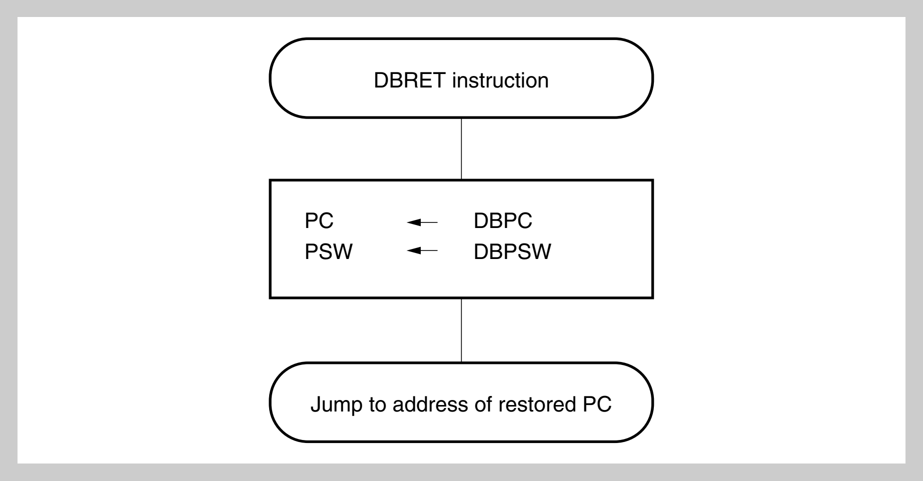


Figure 5-13 Restore processing from exception trap

Note The DBPC and DBPSW registers can be accessed only when the DBTRAP instruction is executed.

5.6.2 Debug trap

The debug trap is an exception that can be acknowledged every time and is generated by execution of the DBTRAP instruction.

When the debug trap is generated, the CPU performs the following processing.

(1) Operation

When the debug trap is generated, the CPU performs the following processing, transfers control to the debug monitor routine, and shifts to debug mode.

1. Saves the restored PC to DBPC.
2. Saves the current PSW to DBPSW.
3. Sets the NP, EP and ID bits of the PSW.
4. Sets the handler address (00000060_H) corresponding to the debug trap to the PC and transfers control.

Figure 5-14 illustrates the processing of the debug trap.

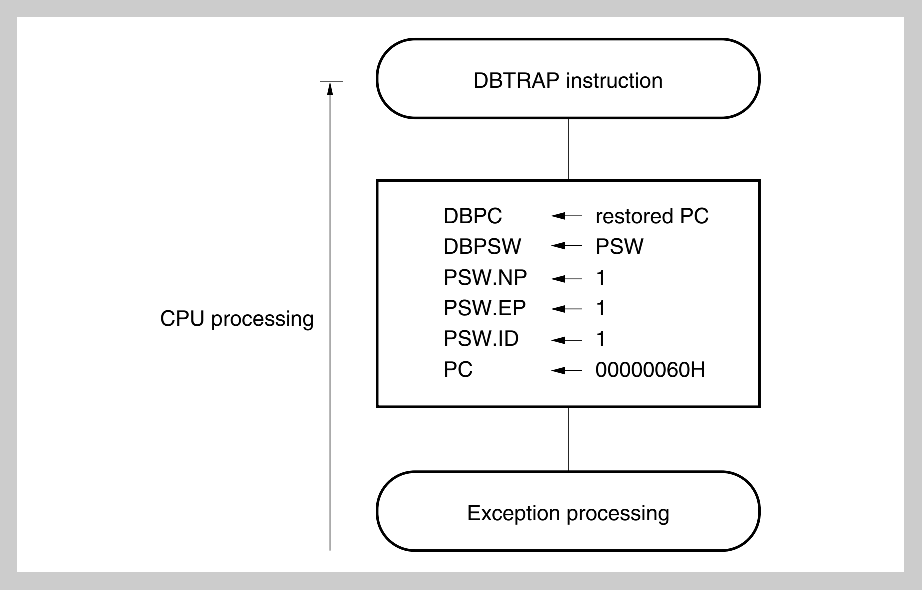


Figure 5-14 Debug trap processing

(2) Restore

Recovery from a debug trap is carried out by the DBRET instruction. By executing the DBRET instruction, the CPU carries out the following processing and controls the address of the restored PC.

1. Loads the restored PC and PSW from DBPC and DBPSW.
2. Transfers control to the address indicated by the restored PC and PSW.

Figure 5-15 illustrates the restore processing from a debug trap.

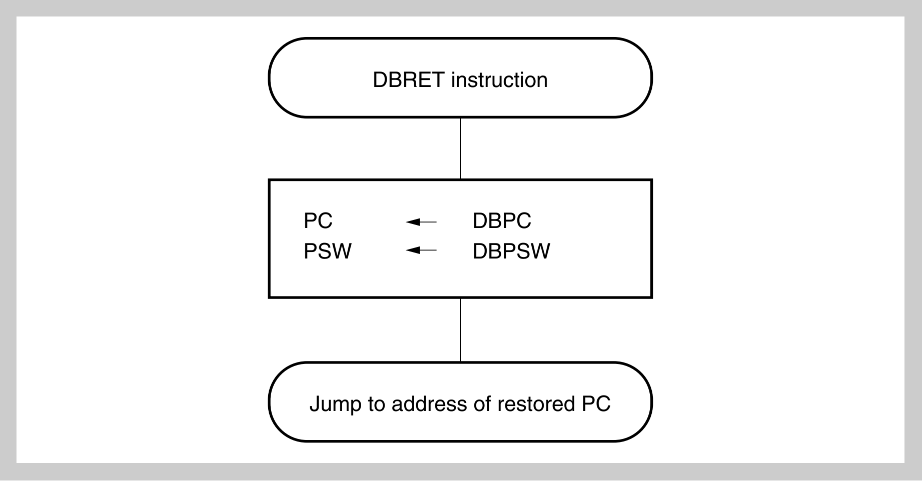


Figure 5-15 Restore processing from debug trap

5.7 Multiple Interrupt Processing Control

Multiple interrupt processing control is a process by which an interrupt request that is currently being processed can be interrupted during processing if there is an interrupt request with a higher priority level, and the higher priority interrupt request is received and processed first.

If there is an interrupt request with a lower priority level than the interrupt request currently being processed, that interrupt request is held pending.

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

If a maskable interrupt enable or a software exception is generated in a maskable interrupt or software exception service program, it is necessary to save EIPC and EIPSW.

This is accomplished by the following procedure.

(1) Acknowledgment of maskable interrupts in service program

Service program of maskable interrupt or exception

```
...
...
•EIPC saved to memory or register
•EIPSW saved to memory or register
•EI instruction (interrupt acknowledgment enabled)
...
...
...
•DI instruction (interrupt acknowledgment disabled)
•Saved value restored to EIPSW
•Saved value restored to EIPC
•RETI instruction
```

¨ Maskable interrupt acknowledgment

(2) Generation of exception in service program

Service program of maskable interrupt or exception

...

...

•EIPC saved to memory or register

•EIPSW saved to memory or register

...

•TRAP instruction ¨ Exception such as TRAP instruction acknowledged.

...

•Saved value restored to EIPSW

•Saved value restored to EIPC

•RETI instruction

The priority order for multiple interrupt processing control has 8 levels, from 0 to 7 for each maskable interrupt request (0 is the highest priority), but it can be set as desired via software. Setting of the priority order level is done using the PPRn0 to PPRn2 bits of the interrupt control request register (PICn), which is provided for each maskable interrupt request. After system reset, an interrupt request is masked by the PMKn bit and the priority order is set to level 7 by the PPRn0 to PPRn2 bits.

The priority order of maskable interrupts is as follows.

(High) Level 0 > Level 1 > Level 2 > Level 3 > Level 4 >
Level 5 > Level 6 > Level 7 (Low)

Interrupt processing that has been suspended as a result of multiple processing control is resumed after the processing of the higher priority interrupt has been completed and the RETI instruction has been executed.

A pending interrupt request is acknowledged after the current interrupt processing has been completed and the RETI instruction has been executed.

Caution In a non-maskable interrupt processing routine (time until the RETI instruction is executed), maskable interrupts are suspended and not acknowledged.

5.8 Interrupt Response Time

The following table describes the interrupt response time (from interrupt generation to start of interrupt processing).

Except in the following cases, the interrupt response time is a minimum of 5 clocks.

- During software or hardware STOP mode
- When an external bus is accessed
- When there are two or more successive interrupt request non-sampling instructions (see *"Periods in which interrupts are not acknowledged" on page 295*).
- When the interrupt control register is accessed

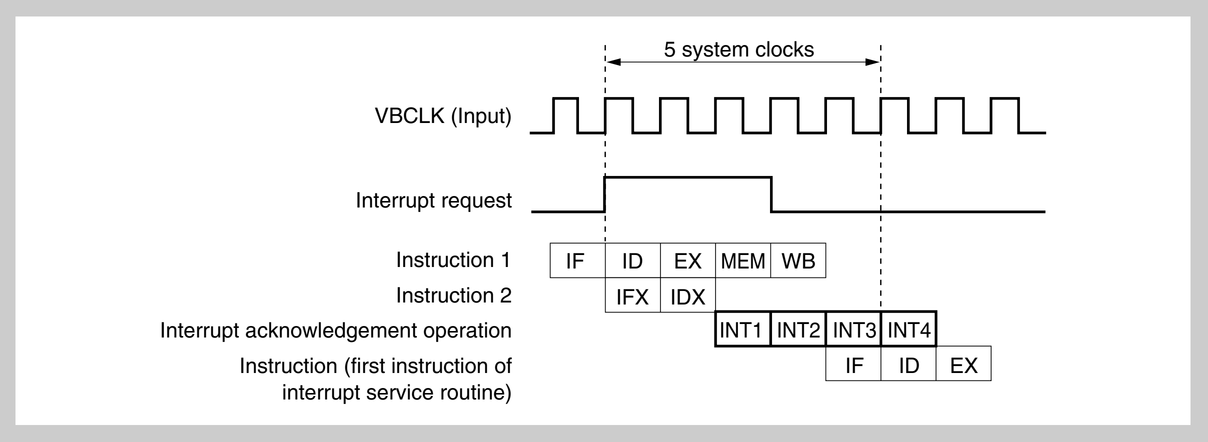


Figure 5-16 Pipeline operation at interrupt request acknowledgment (outline)

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

Note If the same interrupt occurs during the interrupt acknowledge time of 5 cycles, this new interrupt will discarded. The next interrupt of the same source will only be registered after these 5 cycles.

Table 5-6 Interrupt response time

Interrupt response time (internal system clocks)			Condition
	Internal interrupt	External interrupt	
Minimum	4	4 + analog delay time	The following cases are exceptions: • In IDLE/software STOP mode • External bit access • Two or more interrupt request non-sample instructions are executed • Access to interrupt control register
Maximum	6 (in case of latency = 2) 7 (in case of latency = 3)	6 + analog delay time (in case of latency = 2) 7 + analog delay time (in case of latency = 3)	

5.9 Periods in which interrupts are not acknowledged

An interrupt is acknowledged while an instruction is being executed. However, no interrupt will be acknowledged between an interrupt non-sample instruction and the next instruction.

The interrupt request non-sampling instructions are as follows:

- EI instruction
- DI instruction
- LDSR reg2, 0x5 instruction (for PSW)
- The store instruction for the interrupt control register (PICn), in-service priority register (ISPR), and command register (PRCMD).• The store instruction for the following registers and SET1, NOT1, CLR1 instruction.
- Interrupt registers:
  Interrupt control register (xxICn), interrupt mask registers 0 to 7 (IMR0 to IMR7)
- In-service priority register (ISPR)
- Command register (PRCMD)
- Power save control register (PSC)
- On-chip debug mode register (OCDM)
- Peripheral emulation register 1 (PEMU1)

Chapter 6 Key Interrupt Function

6.1 Function

A key interrupt request signal (INTKR) can be generated by inputting a falling edge to the eight key input pins (KR0 to KR7) by setting the key return mode register (KRM).

Table 6-1 Assignment of Key Return Detection Pins

Flag	Pin Description
KRM0	Controls KR0 signal in 1-bit units
KRM1	Controls KR1 signal in 1-bit units
KRM2	Controls KR2 signal in 1-bit units
KRM3	Controls KR3 signal in 1-bit units
KRM4	Controls KR4 signal in 1-bit units
KRM5	Controls KR5 signal in 1-bit units
KRM6	Controls KR6 signal in 1-bit units
KRM7	Controls KR7 signal in 1-bit units

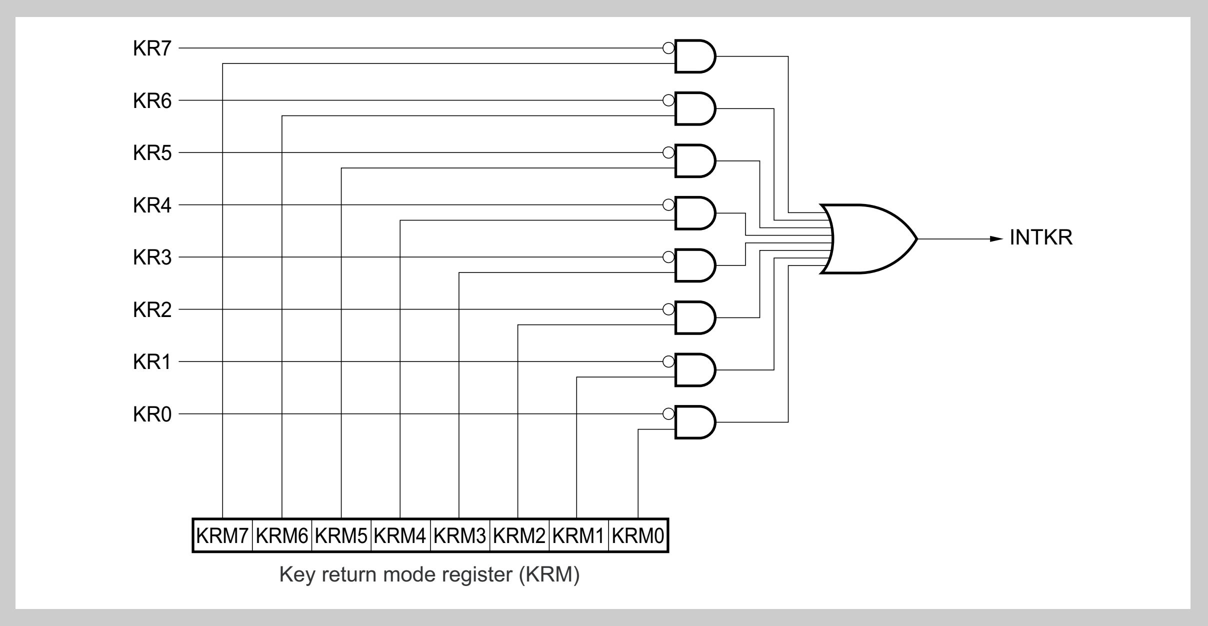


Figure 6-1 Key Return Block Diagram

6.2 Control Register

(1) KRM - Key return mode register

The KRM register controls the KRM0 to KRM7 bits using the KR0 to KR7 signals.

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

Address FFFF F300$_H$.

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

	7	6	5	4	3	2	1	0
KRM	KRM7	KRM6	KRM5	KRM4	KRM3	KRM2	KRM1	KRM0
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 6-2 KRM register contents

Bit position	Bit name	Function
7 to 0	KRMn	Control of key return mode: 0: Does not detect key return signal. 1: Detects key return signal.

Note For the alternate-function pin settings, see *"Pin Functions" on page 32*.

6.3 Cautions

1. If a low level is input to any of the KR0 to KR7 pins, the INTKR signal is not generated even if the falling edge of another pin is input.
2. If the KRM register is changed, an interrupt request signal (INTKR) may be generated. To prevent this, change the KRM register after disabling interrupts (DI), and then enable interrupts (EI) after clearing the interrupt request flag (KRIC.KRIF bit) to 0.
3. To use the Key Interrupt Function, be sure to set the port pin to the key return pin and then enable the operation with the KRM register. To switch from the key return pin to the port pin, disable the operation with the KRM register and then set the port pin.
4. Before writing a new value to the KRM register write a value of 0x00 to the KRM register first.

Chapter 7 Flash Memory

The following V850ES/Fx3 devices are equipped with internal flash memory:

Product	Product name	Code flash	Data flash
V850ES/FE3	µPD70F3370A	128 KB	32 KB
	µPD70F3371	256 KB	32 KB
V850ES/FF3	µPD70F3372	128 KB	32 KB
	µPD70F3373	256 KB	32 KB
V850ES/FG3	µPD70F3374	128 KB	32 KB
	µPD70F3375	256 KB	32 KB
	µPD70F3376A	384 KB	32 KB
	µPD70F3377A	512 KB	32 KB
V850ES/FJ3	µPD70F3378	256 KB	32 KB
	µPD70F3379	384 KB	32 KB
	µPD70F3380	512 KB	32 KB
	µPD70F3381	768 KB	32 KB
	µPD70F3382	1 MB	32 KB
V850ES/FK3	µPD70F3383	512 KB	32 KB
	µPD70F3384	768 KB	32 KB
	µPD70F3385	1 MB	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 are subject to be altered during normal program operation.

Flash memory is commonly used in the following development environments and applications:

- For altering software after solder-mounting of the microcontroller on the target system.
- For differentiating software in small-scale production of various models.
- For data adjustment when starting mass production.
- For facilitating inventory management.
- For updating software after shipment.

The flash memory can be written in different ways:

- by a flash programmer equipped with a suitable adapter (off-board write)
- mounted on the target board by connecting a dedicated flash programmer to the target system (on-board write)
- by the microcontroller’s application software (self-programming)

Additionally a flash memory address space is provided to hold various configuration settings, called option bytes. Via the option bytes start-up configurations can be set for e.g. the Clock Generator and the Watchdog

Timer. The option bytes can be written by use of an external flash programmer and in self-programming mode.

7.1 Code Flash Memory Overview

7.1.1 Code flash memory features

- 4-byte/1 CPU clock access during instruction fetch
- All-blocks or multiple blocks batch erase or single block erase
- Erase/write with single power supply
- Communication with dedicated flash programmer via various serial interfaces
- On-board and off-board programming
- Flash memory programming by self-programming

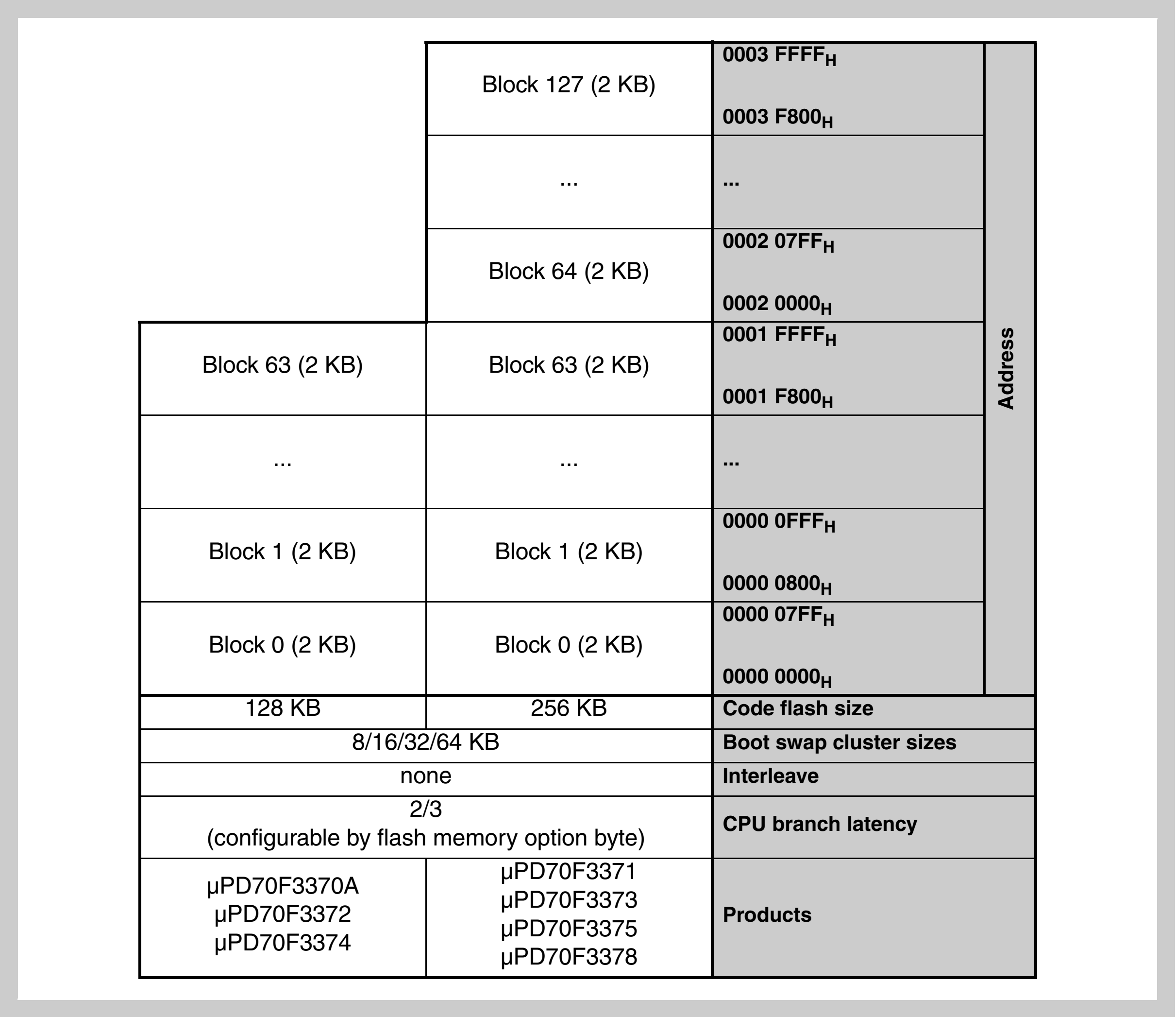
7.1.2 Code flash memory mapping

The microcontroller’s internal code flash memory area is divided into blocks of 2 KB respectively 4 KB blocks and can be programmed/erased in block units. All or some of the blocks can also be erased at once.

Following figures list the block structures and address assignments for all V850ES/Fx3 devices with code flash memory.

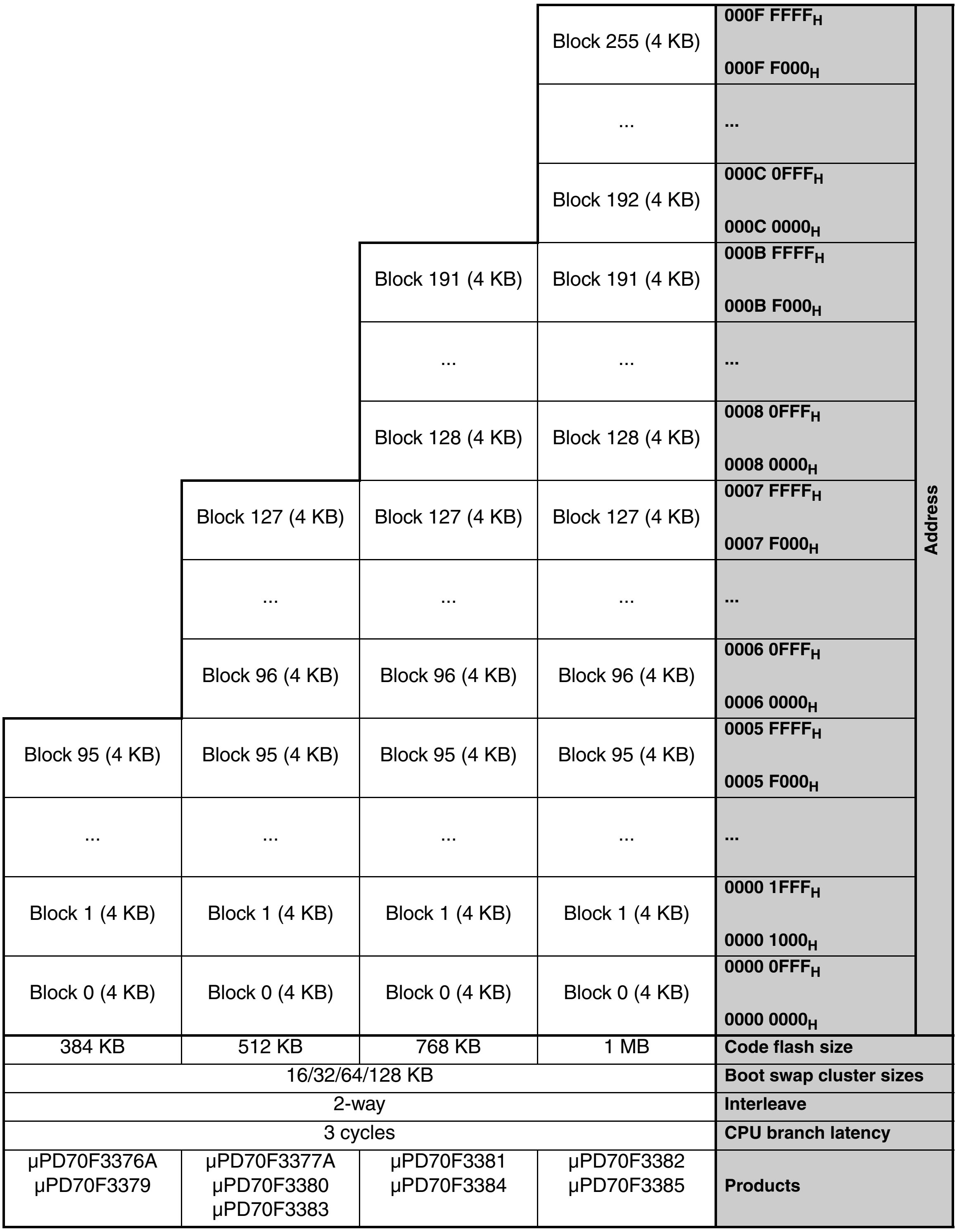
Additional information comprise:

- Boot swap cluster size
  Configurable size of boot cluster for secure self-programming, refer to *“Secure self-programming (boot cluster swapping)” on page 323*.
- Interleave
  Interleave configuration of the flash memory blocks.
- CPU branch latency
  Number of additional CPU clock cycles during instruction fetches of non-linear code. The CPU branch latency may be configurable by the LATENCY control bit of the option byte at address 0000 007B$_H$.



		Address
	Block 127 (2 KB)	0003 FFFF$_H$ 0003 F800$_H$
	...	...
	Block 64 (2 KB)	0002 07FF$_H$ 0002 0000$_H$
Block 63 (2 KB)	Block 63 (2 KB)	0001 FFFF$_H$ 0001 F800$_H$
...	...	...
Block 1 (2 KB)	Block 1 (2 KB)	0000 0FFF$_H$ 0000 0800$_H$
Block 0 (2 KB)	Block 0 (2 KB)	0000 07FF$_H$ 0000 0000$_H$
128 KB	256 KB	**Code flash size**
8/16/32/64 KB		**Boot swap cluster sizes**
none		**Interleave**
2/3 (configurable by flash memory option byte)		**CPU branch latency**
µPD70F3370A µPD70F3372 µPD70F3374	µPD70F3371 µPD70F3373 µPD70F3375 µPD70F3378	**Products**

Figure 7-1 Code flash memory configuration for V850ES/Fx3 devices with up to 256 KB code flash



				Address
			Block 255 (4 KB)	000F FFFF$_H$ 000F F000$_H$
			...	...
			Block 192 (4 KB)	000C 0FFF$_H$ 000C 0000$_H$
		Block 191 (4 KB)	Block 191 (4 KB)	000B FFFF$_H$ 000B F000$_H$
		...	...	...
		Block 128 (4 KB)	Block 128 (4 KB)	0008 0FFF$_H$ 0008 0000$_H$
	Block 127 (4 KB)	Block 127 (4 KB)	Block 127 (4 KB)	0007 FFFF$_H$ 0007 F000$_H$
	...	...	...	...
	Block 96 (4 KB)	Block 96 (4 KB)	Block 96 (4 KB)	0006 0FFF$_H$ 0006 0000$_H$
Block 95 (4 KB)	Block 95 (4 KB)	Block 95 (4 KB)	Block 95 (4 KB)	0005 FFFF$_H$ 0005 F000$_H$
...	...	...	...	...
Block 1 (4 KB)	Block 1 (4 KB)	Block 1 (4 KB)	Block 1 (4 KB)	0000 1FFF$_H$ 0000 1000$_H$
Block 0 (4 KB)	Block 0 (4 KB)	Block 0 (4 KB)	Block 0 (4 KB)	0000 0FFF$_H$ 0000 0000$_H$
384 KB	512 KB	768 KB	1 MB	**Code flash size**
16/32/64/128 KB				**Boot swap cluster sizes**
2-way				**Interleave**
3 cycles				**CPU branch latency**
µPD70F3376A µPD70F3379	µPD70F3377A µPD70F3380 µPD70F3383	µPD70F3381 µPD70F3384	µPD70F3382 µPD70F3385	**Products**

Figure 7-2 Code flash memory configuration for V850ES/Fx3 devices with more than 256 KB code flash

7.1.3 Code flash memory functional outline

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

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

Self-programming The self-programming facility, 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 serving 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 *“Flash memory programming control” on page 316* 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, the variable reset vector 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 self-programming.

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 *“Secure self-programming (boot cluster swapping)” on page 323*.

Protection A set of protection flags can be specified during flash memory programming to prohibit access the flash memory in different ways, implying 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.
For further information concerning data protection refer to *“Data Protection and Security” on page 334*.

Variable reset vector The variable reset vector function allows flexible assignment of the application program start by redefinition of the reset vector.

For further information concerning the variable reset vector refer to *“Variable Reset Vector” on page 329*.

Table 7-1 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 *“Flash Programming with Flash Programmer” on page 311*.	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 application software. For details refer to *“Code Flash Self-Programming” on page 321*.	Normal operation mode

Table 7-2 on page 305 summarizes the functions used to modify flash memory content.

Table 7-2 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 cluster protection flag - is also erased. **Caution:** The chip erase function erases also the data flash memory.	√	×[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.	√	×[b]
Checksum	Microcontroller internally calculated checksum over the entire flash memory content is compared with the checksum calculated by the serial programmer	√	×
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 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 cluster all other protections have no effect in self-programming mode.
Protection settings can be activated in self-programming mode. Already activate protection settings can not be deactivated.

The following table lists the available flash memory protection functions.

For details refer to *"Data Protection and Security" on page 334*.

Table 7-3 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 cluster impossible.	√	√

[a] The boot cluster protection flag is not erased.

7.1.4 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 (chip erase)
  All blocks are erased all together.
- Block erasure

  Each 2 KB respectively 4 KB flash memory block can be erased separately.

  In self-programming mode any number of contiguous flash memory blocks can be erased all together.

Rewrite In self- 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 16 byte. Each unit can be rewritten only once after erasure of the complete block.

7.2 Data Flash Memory

The V850ES/Fx3 Series products contain a 32 KB data flash in addition to the code flash. The data flash is on-chip connected to the external memory bus.

Caution Before entering any power save mode make sure that any access to the data flash is completed.

7.2.1 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 application code can be executed from code flash

7.2.2 Data flash memory map

The data flash can be mapped by software to different memory address locations.

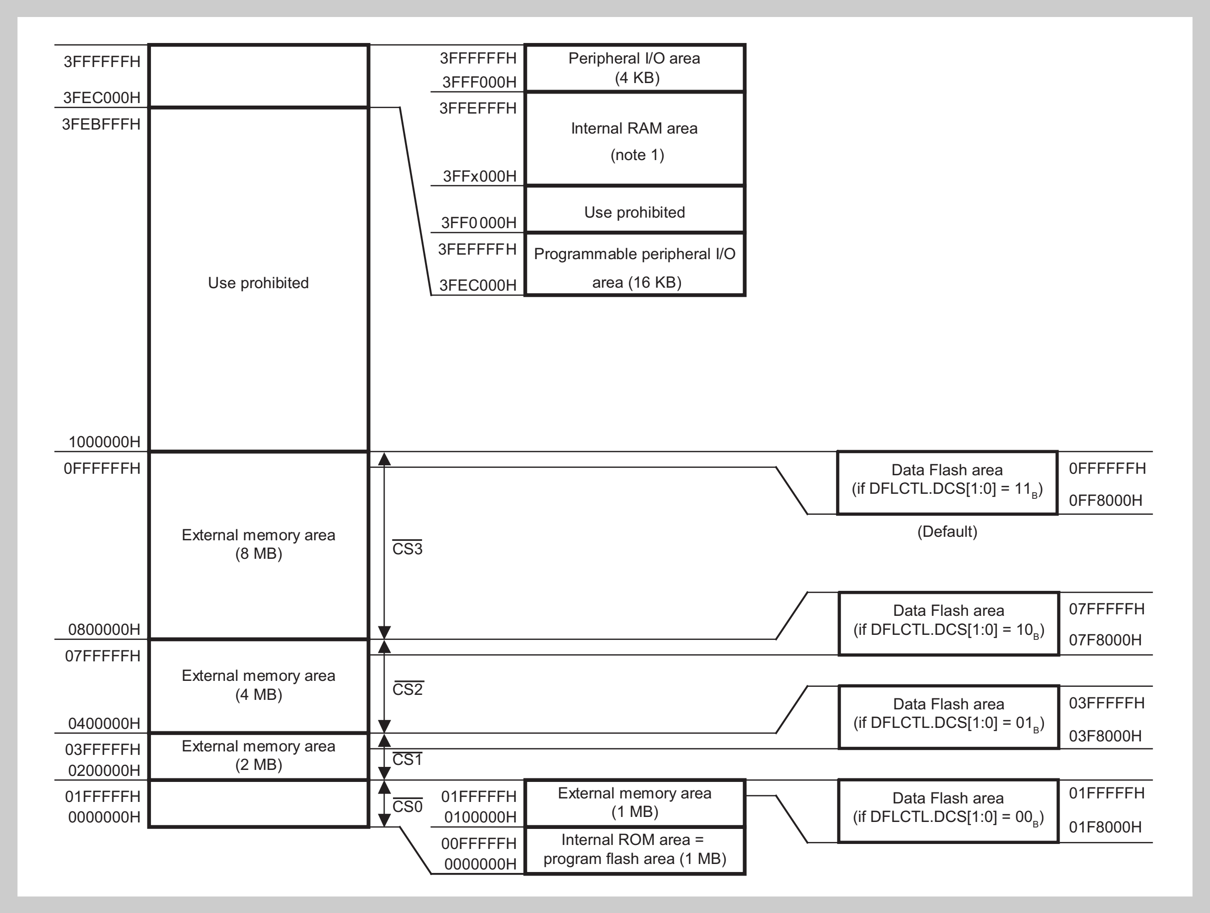


Figure 7-3 Memory mapping of data flash

The memory interface has to be set up as follows for the chosen chip select area:

Control	Set to	Comment
BSC.BSn0	1	Bus size: 16 bit
DWC0.DWn[2:0]	001_B	For $f_{xx} \leq 40$ MHz: 1 data wait state
	010_B	For 40 MHz $< f_{xx} \leq 48$ MHz: 2 data wait states
AWC.AHWn	0	No address hold wait states
AWC.ASWn	0	For $f_{xx} \leq 24$ MHz: no address setup wait states
	1	For 24 MHz $< f_{xx} \leq 48$ MHz: 1 address setup wait state
BCC.BCn1	0	No idle state insertion

For further information about the memory interface configuration refer to *“Bus and Memory Control (BCU, MEMC)” on page 339*.

7.2.3 Data flash control register

(1) DFLCTL - Data flash control register

The data flash is controlled with the data flash control register DFLCTL to enable the access to the data flash and to define the memory address location.

Writing to this register is protected by a special sequence of instructions. Please refer to *"Write Protected Registers" on page 176*.

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

Address FFFF FCF8$_H$

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

	7	6	5	4	3	2	1	0
DFLCTL	DFLEN	0	0	0	0	0	DCS1	DCS0
	R/W	R	R	R	R	R	R/W	R/W

Table 7-4 DFLCTL register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>7</td><td>DFLEN</td><td>Read access control of data flash:
0: Disable
1: Enable</td></tr>
<tr><td>1 to 0</td><td>DCS[1:0]</td><td>Selection of mapping area of data flash:
<table>
<tr><th>DCS1</th><th>DCS0</th><th>Chip select address range</th></tr>
<tr><td>0</td><td>0</td><td>CS0 area (01F 8000$_H$ – 01F FFFF$_H$)</td></tr>
<tr><td>0</td><td>1</td><td>CS1 area (03F 8000$_H$ – 03F FFFF$_H$)</td></tr>
<tr><td>1</td><td>0</td><td>CS2 area (07F 8000$_H$ – 07F FFFF$_H$)</td></tr>
<tr><td>1</td><td>1</td><td>CS3 area (0FF 8000$_H$ – 0FF FFFF$_H$) (default)</td></tr>
</table></td></tr>
</table>

Note

1. Do not set the external memory range in the same address area as the data flash when a read access is done by CPU and DMA.
2. Do not write to this register when the external access is being executed by using the external bus interface function.

7.2.4 Data flash reading

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

Reading of the data flash is performed with the following procedure:

1. Select the Chip select area for the data flash to work in by configuring the DFLCTL register.
1. Initialize the memory interface as described in *7.2.2 on page 308* for the selected Chip Select area.
2. Enable the read access to the data flash by setting the DFLCTL.DFLEN = 1.
3. Execute read operation

7.2.5 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 erase command of an external programmer erases also the data flash.

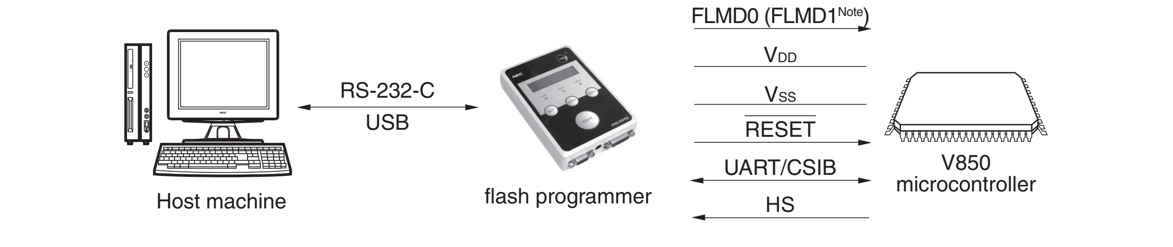
7.3 Flash Programming with Flash Programmer

A dedicated flash programmer can be used for external writing of the flash memory.

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

7.3.1 Programming environment

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



Note: FLMD1 connection may be replaced by a pull-down resistor on the board

Figure 7-4 Environment to write program to flash memory

A host machine is required for controlling the flash programmer.

Following microcontroller serial interfaces can be used as the interface between the flash programmer and the microcontroller:

- asynchronous serial interface UART
- clocked serial interface 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 7-6*.

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 microcontroller provides. UART and CSI signal names are used accordingly.

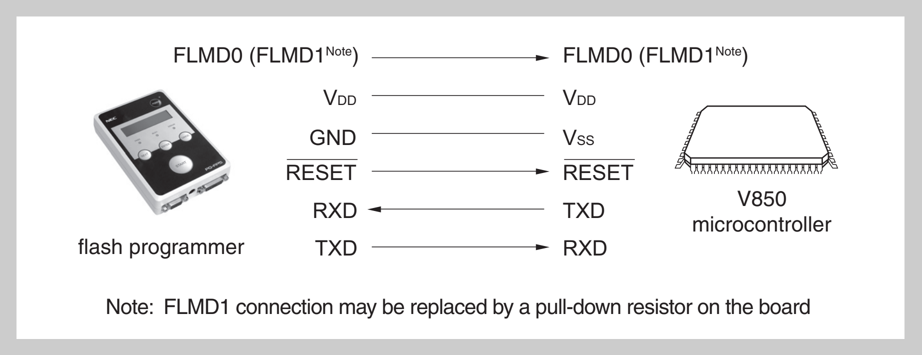
7.3.2 Communication mode

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

For programming via the synchronous serial interface CSI 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.

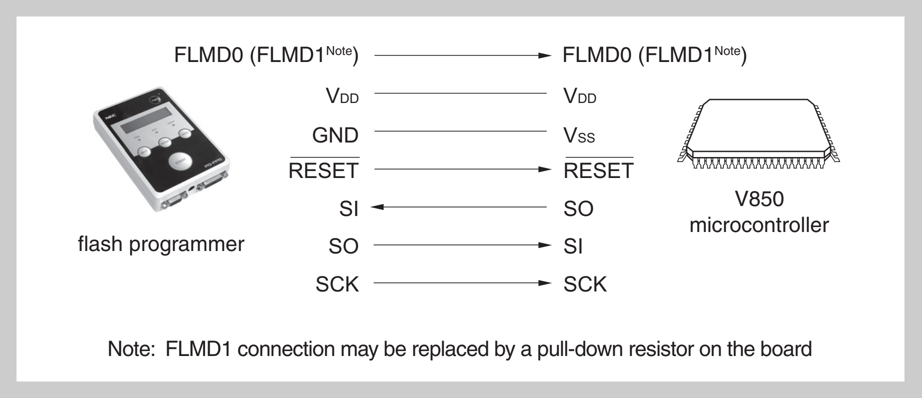


Note: FLMD1 connection may be replaced by a pull-down resistor on the board

Figure 7-5 Communication with flash programmer via UART

(2) CSI without handshake

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



Note: FLMD1 connection may be replaced by a pull-down resistor on the board

Figure 7-6 Communication with flash programmer via CSI without handshake

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

(3) CSI with handshake (CSI + HS)

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

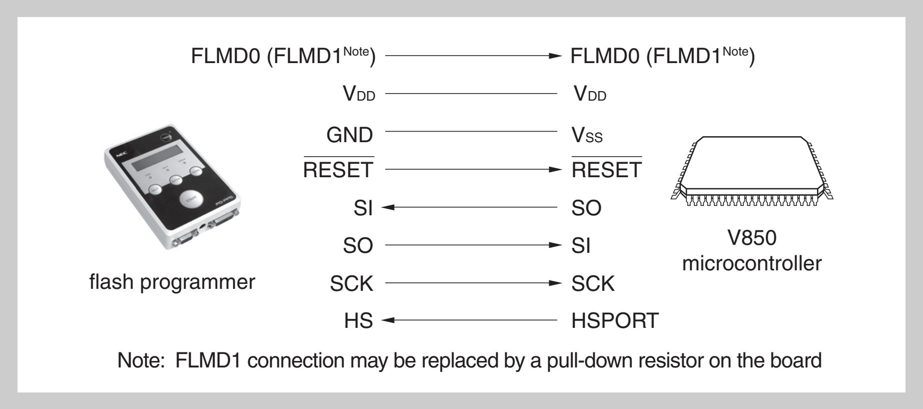


Figure 7-7 Communication with flash programmer via CSI with handshake

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

7.3.3 Pin connection with flash programmer PG-FP5

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

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

If the PG-FP5 is used as the flash programmer, it generates the signals listed in *Table 7-5* for the microcontroller. For details, refer to the PG-FP5 User's Manual (U15260E).

Table 7-5 Signals generated by flash programmer PG-FP5

PG-FP5			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	´×	´×	´×
$\overline{\text{RESET}}$	Output	Reset signal	$\overline{\text{RESET}}$	√	√	√
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	´×	´×	√

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

Table 7-6 Wiring of V850ES/Fx3 flash writing adapters

Flash programmer (FG-FP5) connection pin			Name of FA board[a] pin	Name of Serial I/F pin		
Signal name	I/O	Pin function		UARTD0	CSIB0 + HS	CSIB0
SI/RxD	I	Receive signal	SI	TXDD0	SOB0	
SO/TxD	O	Transmit signal	SO	RXDD0	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	$\overline{\text{RESET}}$	$\overline{\text{RESET}}$		
FLMD0	I	Write voltage	FLMD0	FLMD0		
FLMD1	I	Write voltage	FLMD1	FLMD1		
HS	I	Handshake signal for CSI + HS	RESERVE/ HS	Not needed	HSPORT = PCM0	Not needed
VDD	–	VDD voltage generation/ voltage monitor	VDD	V_{DD}		
				V_{DD1}		
				BV_{DD}		
				EV_{DD}		
				AV_{REF0}		
				AV_{REF1}		
GND	–	Ground	GND	V_{SS}		
				V_{SS1}		
				BV_{SS}		
				EV_{SS}		
				AV_{SS}		
				AV_{SS1}		

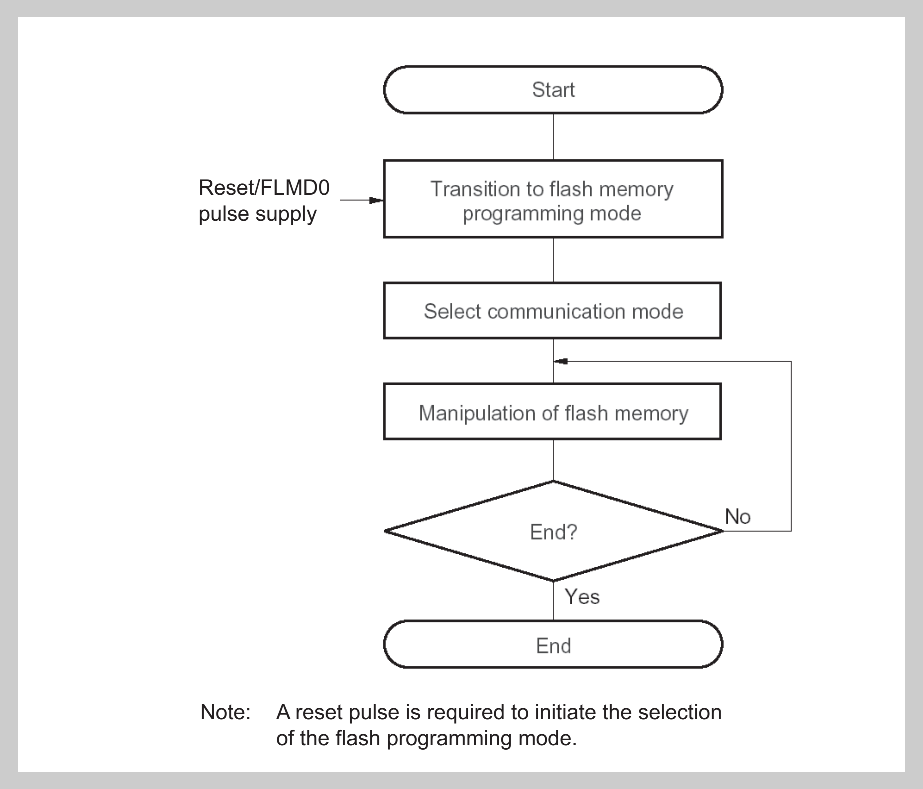
[a] FA board: flash programming adadpter board

Table 7-7 V850ES/Fx3 pin numbers for serial programming

Pin name	Port	V850ES/FE3 pin no.	V850ES/FF3 pin no.	V850ES/FG3 pin no.	V850ES/FJ3 pin no.	V850ES/FK3 pin no.
TXDD0	P30	22	22	25	25	26
RXDD0	P31	23	23	26	26	27
SIB0	P40	19	19	22	22	23
SOB0	P41	20	20	23	23	24
SCKB0	P42	21	21	24	24	25
$\overline{\text{RESET}}$	–	9	14	14	14	14
FLMD0	–	3	8	8	8	8
FLMD1	PDL5	52	62	76	110	121
PCM0	PCM0	45	49	61	85	110

7.3.4 Flash memory programming control

The procedure to program the flash memory is illustrated below.


Start
Reset/FLMD0 pulse supply
Transition to flash memory programming mode
Select communication mode
Manipulation of flash memory
End?
No
Yes
End

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

Figure 7-8 Flash memory programming procedure

(1) Operation mode control

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

To set this mode, set the FLMD0 and FLMD1 pins as shown in *Table 7-8 on page 316* and release $\overline{\text{RESET}}$.

In the normal operation mode, VSS 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 VSS level.

Table 7-8 Operation mode setting

Pins		Operation mode
FLMD0	**FLMD1**	
V_{SS}	Don’t care	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.

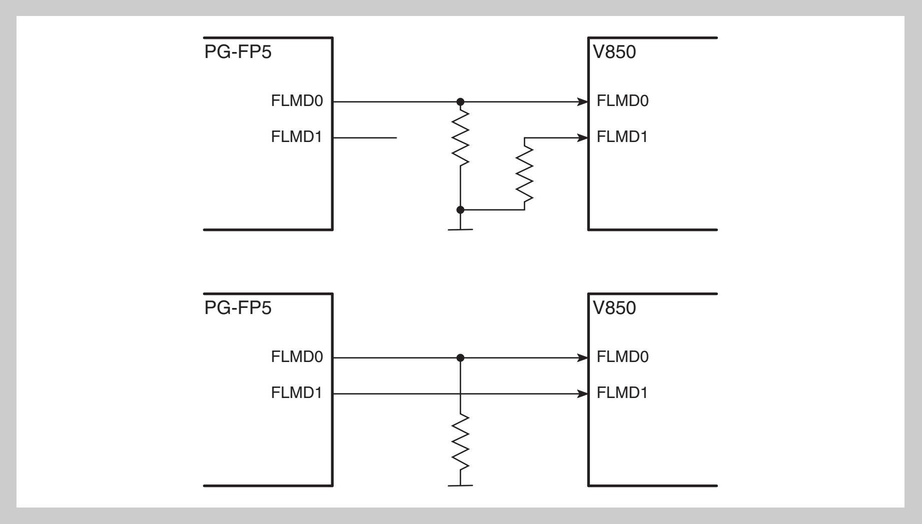


Figure 7-9 **Example of connection to flash programmer PG-FP5**

Once started in normal operation mode (FLMD0 = 0), FLMD0 pin is used for enabling self-programming. Refer also to *7.4 on page 321*.

(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 concerned signals do not conflict with the signals of the flash programmer and the microcontroller. Output pins of the other devices must be isolated or set in high impedance state. Ensure that the other devices do not malfunction because of flash programmer signals.

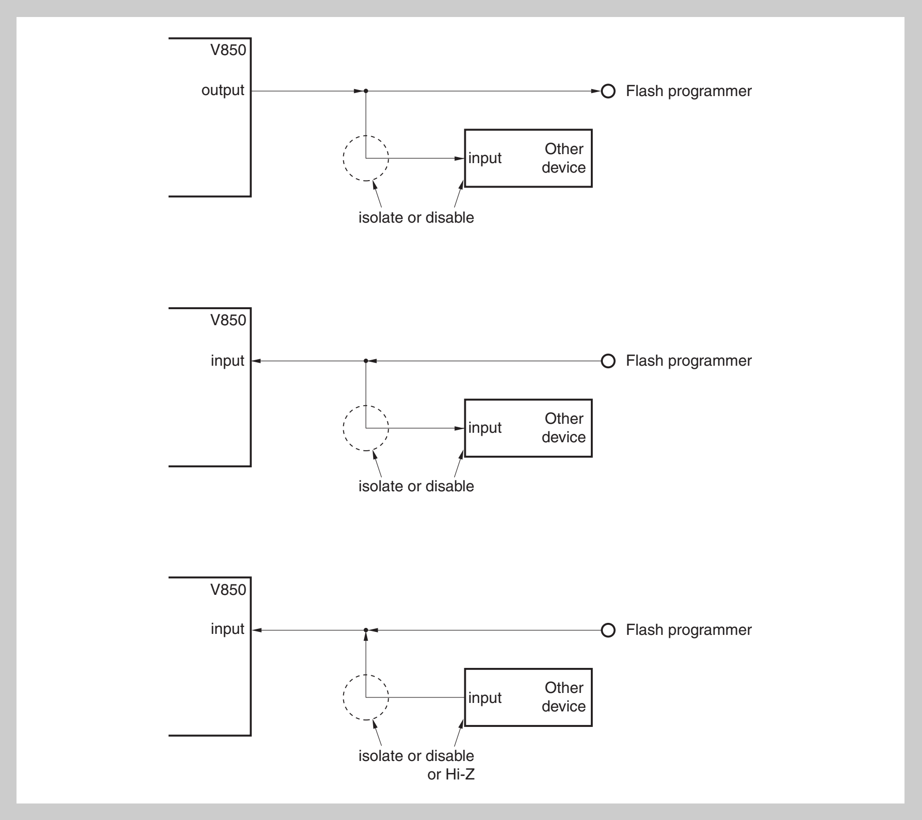


Figure 7-10 **Potential conflicts with serial interfaces signals**

$\overline{\text{RESET}}$ Pay attention in particular if the flash programmer's $\overline{\text{RESET}}$ signal is connected also to an on-board reset generation circuit. The reset output of the reset generator may ruin the flash programming process and may need to be isolated or disabled.

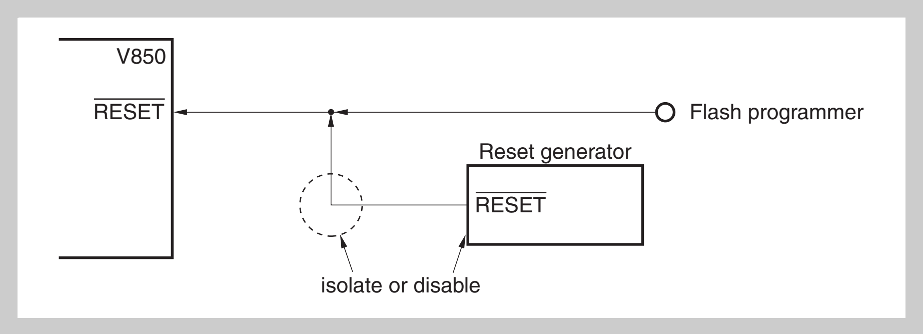


Figure 7-11 Potential conflict with $\overline{\text{RESET}}$

Ports The V850 port pins adopts following status during serial programming:

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

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

In case the default state after reset of the pins not used for programming is inport 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 resistors.

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 7-12 on page 319 gives an example how the UART is established for the communication between the flash programmer and the microcontroller.

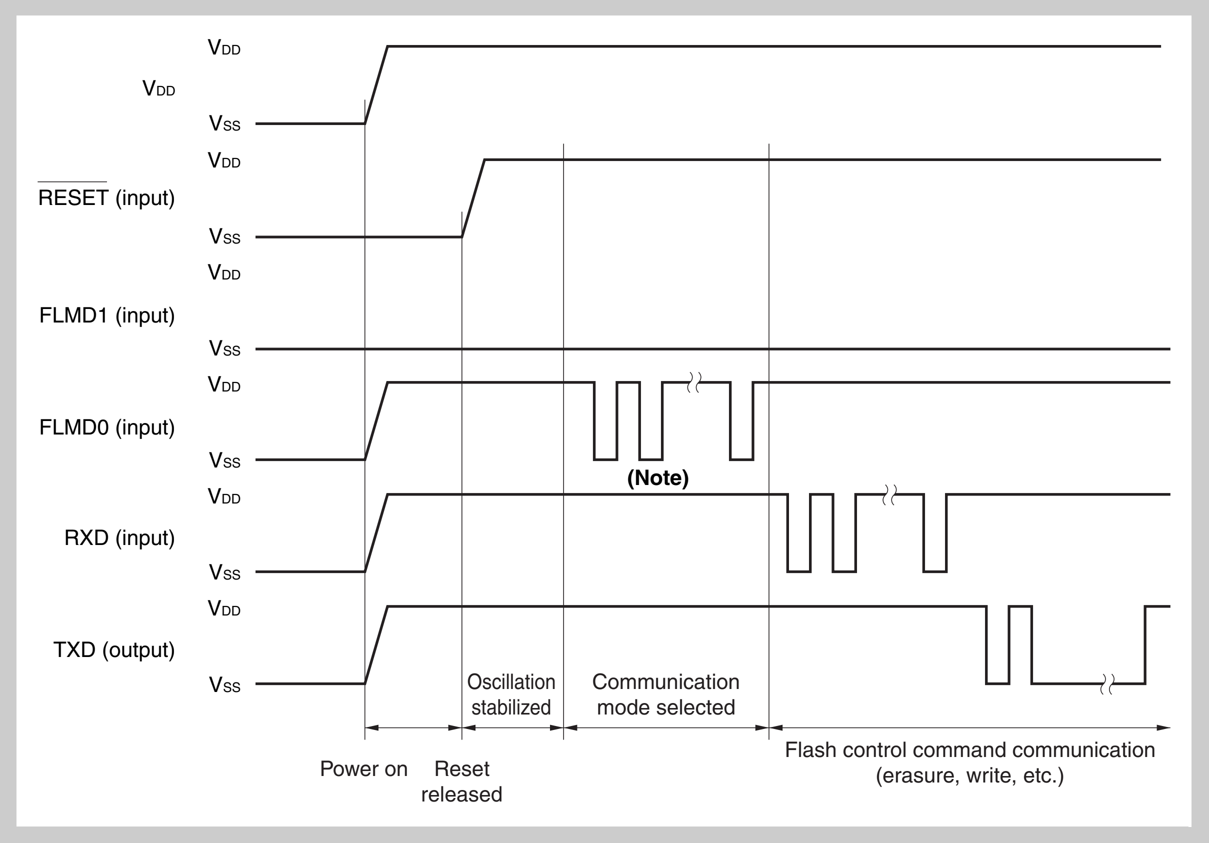


Figure 7-12 Selection of communication mode

Note The number of clocks to be inserted differs depending on the chosen communication mode. For details, refer to *Table 7-9 on page 319*.

Table 7-9 FLMD0 pulses for communication mode setting

FLMD0 pulses	Communication Mode	Remarks
0	UART	Communication rate: 9600 bps (after reset), LSB first
8	CSI	V850ES/Fx3 performs slave operation, MSB first
11	CSI + HS	V850ES/Fx3 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 microcontoller 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 microcontroller. Depending on the commands, the microcontroller returns status information or the requested data.

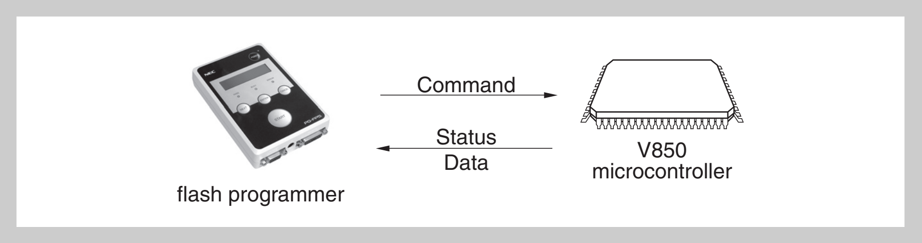


Figure 7-13 Communication commands exchange

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

Table 7-10 Flash memory control commands

Classification	Command name	Support			Function
		CSIB	CSIB + HS	UARTD	
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.

7.4 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 Renesas, the user's program is able to rewrite the flash memory with data, transferred in advance to the internal RAM or the external memory.

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

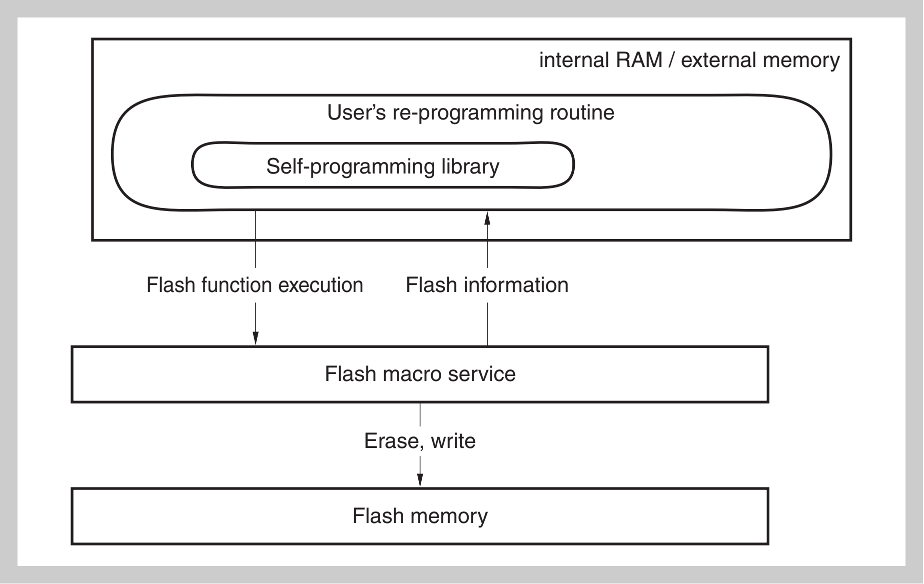


Figure 7-14 Concept of self-programming

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

Consequently the instructions of user re-programming software routines, which shall remain in operation during the self-programming procedure, must be copied from the flash memory to the internal RAM or external memory 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 *"Interrupt handling during flash self-programming" on page 328*).

It is recommended to refer to the application note "Self-Programming" (document no. 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 and the library can be loaded via the URL

http://www.renesas.eu/updates

7.4.1 Self-programming enable

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

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

For security reasons writing and erasing of the flash memory must be additionally 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.

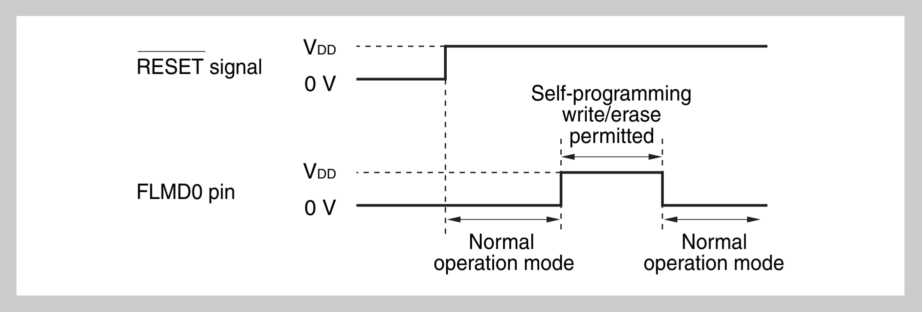


Figure 7-15 Self-programming enable

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

7.4.2 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 like

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

Detailed information 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.renesas.eu/updates

7.4.3 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.

Caution Boot cluster swapping is only supported, if the variable reset vector remains in its default state 0000 0000_H.
If the reset vector is changed to other values, boot cluster swapping is not possible.

Boot swap cluster A group of boot blocks to be swapped. 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.

Boot swap flag Which of the two clusters is the active boot cluster is controlled by the boot swap flag, that can be defined during flash programming via the self-programming library.
The boot swap flag is stored in the flash memory extra area.

Figure 7-16 on page 323 shows an example of the boot block swapping function with a cluster size of 4 flash memory blocks. After inverting the boot_flag - it becomes not(boot_flag) - blocks 4 to 7 become the active boot cluster. Thus after next reset release the user's program starts from the new boot swap cluster.

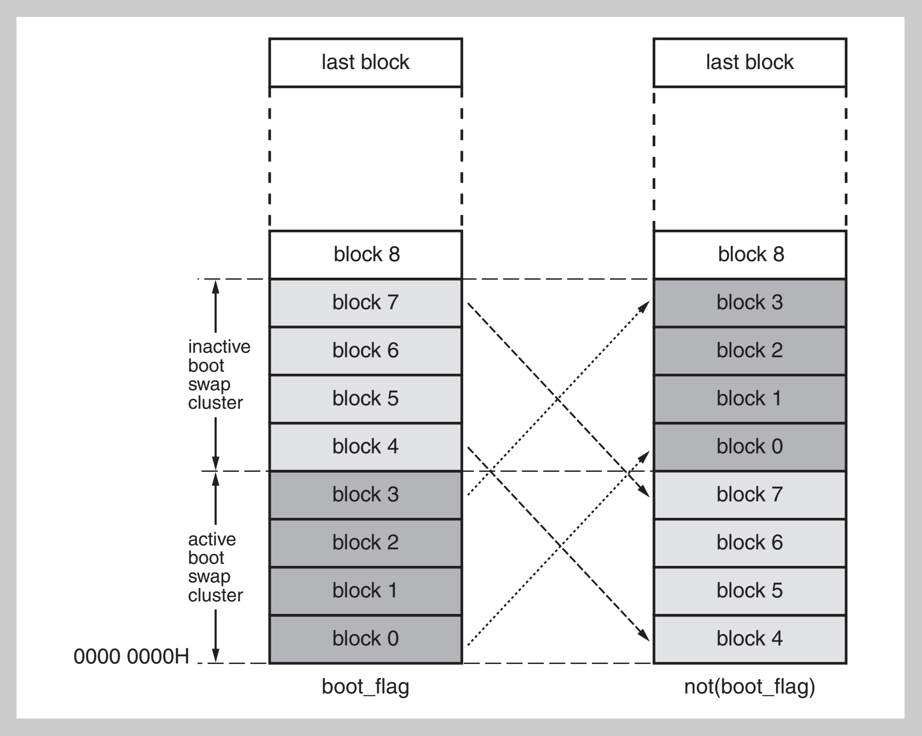


Figure 7-16 Boot swap cluster swapping function

Secure self-programming

The boot cluster swapping function enables secure self-programming. In case the boot code shall be rewritten, the new code can be written to the inactive boot cluster, while the boot_flag remains in its previous state.
If rewriting of the boot cluster has been completed successfully, the boot_flag can be inverted, making the new boot code active.
If rewriting of the new boot code fails for any reason, e.g. power fail or unintended reset, the old boot code still remains active and rewriting can be started again.

Boot cluster

The boot code size itself may be smaller than the boot swap cluster size.

The number of flash memory blocks, which are part of the boot code, are named boot cluster. The number of boot blocks, which are member of the cluster, can be defined during self-programming via the self-programming library.

The boot cluster size determines the boot swap cluster size. This is automatically evaluated from the number of boot blocks, defined during self-programming.

Table 7-11 on page 325 shows the relation between the number of boot blocks, the boot cluster size and the boot swap cluster.

Number of boot blocks

The number of boot blocks has to be defined by the user during self-programming. It determines the blocks, which are subject to the boot cluster protection, that allows to protect the boot blocks from any erase or write process.

Table 7-11 Relation between boot block and boot swap cluster

Number of boot blocks[a]	Devices with 2 KB blocks (≤ 256 KB code flash)		Devices with 4 KB blocks (≥ 384 KB code flash)	
	Boot cluster	**Boot swap cluster**	**Boot cluster**	**Boot swap cluster**
00_H	0000 0000$_H$ - 0000 07FF$_H$ (2 KB)	0000 0000$_H$ - 0000 1FFF$_H$ (8 KB)	0000 0000$_H$ - 0000 0FFF$_H$ (4 KB)	0000 0000$_H$ - 0000 3FFF$_H$ (16 KB)
01_H	RESV - 0000 0FFF$_H$H (4 KB)		RESV - 0000 1FFF$_H$ (8 KB)	
02_H	RESV - 0000 17FF$_H$ (6 KB)		RESV - 0000 2FFF$_H$ (12 KB)	
03_H	RESV - 0000 1FFF$_H$ (8 KB)		RESV - 0000 3FFF$_H$ (16 KB)	
04_H	RESV - 0000 27FF$_H$ (10 KB)	0000 0000$_H$ - 0000 3FFF$_H$ (16 KB)	RESV - 0000 4FFF$_H$ (20 KB)	0000 0000$_H$ - 0000 7FFF$_H$ (32 KB)
...	...		...	
07_H	RESV - 0000 3FFF$_H$ (16 KB)		RESV - 0000 7FFF$_H$ (32 KB)	
08_H	RESV - 0000 47FF$_H$ (18 KB)	0000 0000$_H$ - 0000 7FFF$_H$ (32 KB)	RESV - 0000 8FFF$_H$ (36 KB)	0000 0000$_H$ - 0000 FFFF$_H$ (64 KB)
...	...		...	
$0F_H$	RESV - 0000 7FFF$_H$ (max. 32 KB)		RESV - 0000 FFFF$_H$ (64 KB)	
10_H	RESV - 0000 87FF$_H$ (34 KB)	0000 0000$_H$ - 0000 7FFF$_H$ (64 KB)	RESV - 0001 0FFF$_H$ (68 KB)	0000 0000$_H$ - 0001 FFFF$_H$ (128 KB)
...	...		...	
$1F_H$	RESV - 0000 FFFF$_H$ (64 KB)		RESV - 0001 FFFF$_H$ (128 KB)	
20_H	RESV - 0001 07FF$_H$ (66 KB)		RESV - 0002 0FFF$_H$ (132 KB)	
...	...		...	
$7F_H$	RESV - 0003 FFFF$_H$ (256 KB)		RESV - 0007 FFFF$_H$ (512 KB)	
80_H	Setting prohibited			
...				
FF_H				

[a] The number of boot blocks has to be defined by the user during self-programming or via the external programmer during serial programming.

RESV: Start address of the block including the boot vector.

Figure 7-17 on page 327 illustrates an example with following settings:

Table 7-12

Number of Boot blocks[a]	Boot cluster	Boot swap cluster
00$_H$	0000 0000$_H$ - 0000 07FF$_H$ (2 KB)	0000 0000$_H$ - 0000 1FFF$_H$ (8 KB)
01$_H$	RESV - 0000 0FFF$_H$H (4 KB)	
02$_H$	RESV - 0000 17FF$_H$ (6 KB)	
03$_H$	RESV - 0000 1FFF$_H$ (8 KB)	
04$_H$	RESV - 0000 27FF$_H$ (10 KB)	0000 0000$_H$ - 0000 3FFF$_H$ (16 KB)
...	...	
07$_H$	RESV - 0000 3FFF$_H$ (16 KB)	
08$_H$	RESV - 0000 47FF$_H$ (18 KB)	0000 0000$_H$ - 0000 7FFF$_H$ (32 KB)
...	...	
0F$_H$	RESV - 0000 7FFF$_H$ (max. 32 KB)	
10$_H$	RESV - 0000 87FF$_H$ (34 KB)	0000 0000$_H$ - 0000 7FFF$_H$ (64 KB)
...	...	
1F$_H$	RESV - 0000 FFFF$_H$ (64 KB)	
20$_H$	RESV - 0001 07FF$_H$ (66 KB)	
...	...	
7F$_H$	RESV - 0003 FFFF$_H$ (256 KB)	
80$_H$	Setting prohibited	
...		
FF$_H$		

[a] The number of boot blocks has to be defined by the user during self-programming or via the external programmer during serial programming.

RESV: Start address of the block including the boot vector.

- number of boot blocks: 2 (boot cluster contains 2 blocks), thus the active boot cluster comprises
  - if boot_flag: blocks 0and 1
  - if not(boot_flag): blocks 4 and 5
- active boot swap clusters comprises

- if boot_flag: blocks 0 to 3
- if not(boot_flag): blocks 4 to 7

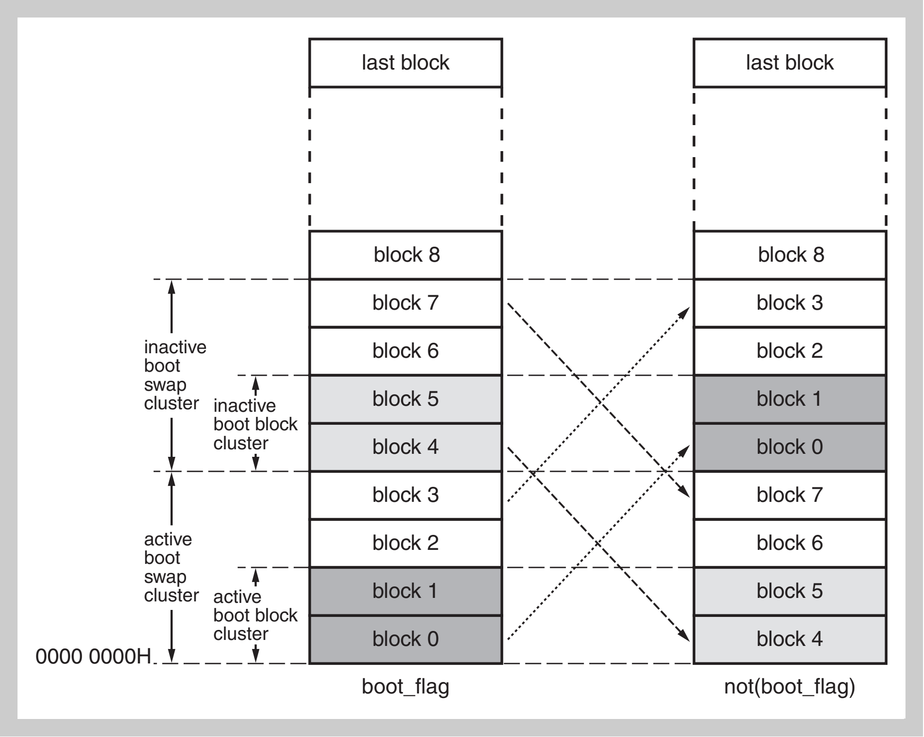


Figure 7-17 Boot cluster swapping function

Boot block protection

To prohibit rewriting of the boot blocks, the boot cluster protection flag can be set during flash memory programming. When this flag is set, the blocks of the active boot cluster can neither be erased nor written. Boot cluster swapping is impossible as well.
Note that only the blocks of the active boot cluster are protected. In the example according to *Figure 7-17 on page 327*, for instance, blocks 0 and 1 would be prohibited, while blocks 2 and 3 could still be erased and written.

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

For further information concerning flash memory protection flags refer to *“Data Protection and Security” on page 334*.

7.4.4 Interrupt handling during flash self-programming

This 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 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 concerned interrupt handler routine needs to be copied to the internal RAM, respectively external memory. The user has to initiate this copy process.
- The concerned interrupt acknowledge has to be re-routed to that handler. Re-routing to the handler is done by the internal firmware. Thus the user doesn't have to care about.

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 done only during the flash programming environment is activated. If self-programming is deactivated, the normal interrupt vector table in the flash memory is used.

All interrupt vectors are relocated to one entry point in the internal RAM:

- New entry point of *all* maskable interrupts is the 1st address of the internal RAM. A handler routine must check the interrupt source. The interrupt request source can be identified via the interrupt/exception source register ECR.EICC (refer to *"System register set" on page 159*)
- New entry point of *all* non maskable interrupts is the word address following the maskable interrupt entry, i.e. the second address of the internal RAM. The interrupt request source can be identified via the interrupt/exception source register ECR.FECC (refer to*"System register set" on page 159*).

In general a jump to a special handler routine will be placed at the 1st and 2nd internal RAM address, which identifies the interrupt sources and branches to the correct interrupt service routine.

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

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.renesas.eu/updates

7.5 Variable Reset Vector

This microcontroller provides a facility to specify the address of the first user software instruction to be executed after reset release.

By default the first user's instruction to be executed after reset, i.e. the reset vector, is the one stored at address 0000 0000_H. During flash programming another reset vector address can be specified, the so called variable reset vector.

The variable reset vector is stored in the flash memory extra area.

The variable reset vector can be modified in all flash programming modes. The self-programming library supports this function.

Note The variable reset vector only determines the user's program start after reset. The vector table is not affected. It is always located at address 0000 0000_H.

7.6 Flash Mask Options

In the option data area, a block subject to mask options is specified. Make sure to set the option data area corresponding to the following option bytes in the program at address 0000 007A_H/0000 007B_H as default data.

Caution If the flash memory is programmed during a debug session with the on-chip debugger 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.

(1) Option Byte 0000 0007A_H

Table 7-13 Option byte 0000 0007A_H settings

Address	Set Value	Setting	
007A_H	00$_H$	Internal-OSC:	Can be stopped.
		WDT2:	Count clock can be selected. Overflow signal can be selected from INTWDT2 or WDT2RES.
		Sub oscillator:	Crystal resonator connection
	01$_H$	Internal-OSC:	Cannot be stopped.
		WDT2:	Count clock can be selected. Overflow signal can be selected from INTWDT2 or WDT2RES.
		Sub oscillator:	Crystal resonator connection
	02$_H$	Internal-OSC:	Can be stopped.
		WDT2:	Count clock is fixed to Internal-OSC. Overflow signal is fixed to WDT2RES.
		Sub oscillator:	Crystal resonator connection
	03$_H$	Internal-OSC:	Cannot be stopped.
		WDT2:	Count clock is fixed to Internal-OSC. Overflow signal is fixed to WDT2RES.
		Sub oscillator:	Crystal resonator connection
	C0$_H$	Internal-OSC:	Can be stopped.
		WDT2:	Count clock can be selected. Overflow signal can be selected from INTWDT2 or WDT2RES.
		Sub oscillator:	RC oscillation connection
	C1$_H$	Internal-OSC:	Cannot be stopped.
		WDT2:	Count clock can be selected. Overflow signal can be selected from INTWDT2 or WDT2RES.
		Sub oscillator:	RC oscillation connection
	C2$_H$	Internal-OSC:	Can be stopped.
		WDT2:	Count clock is fixed to Internal-OSC. Overflow signal is fixed to WDT2RES.
		Sub oscillator:	RC oscillation connection
	C3$_H$	Internal-OSC:	Cannot be stopped.
		WDT2:	Count clock is fixed to Internal-OSC. Overflow signal is fixed to WDT2RES.
		Sub oscillator:	RC oscillation

(2) Option Byte 0000 0007B$_H$

	7	6	5	4	3	2	1	0
0000 007B$_H$	SUBCLK	0	0	LATENCY	PLLO	PRSI	PLLI1	PLLI0

Table 7-14 Option byte 0000 0007B$_H$ contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>7</td><td>SUBCLK</td><td>Clock source at subclock operating mode.
0: SubOSC selection
1: 240 KHz internal oscillator selection</td></tr>
<tr><td>4</td><td>LATENCY</td><td>Selection of CPU branch latency.
0: CPU Branch latency: 2
1: CPU Branch latency: 3
<b>Note:</b> Setting of the LATENCY bit is fixed to a branch latency of 3 for the following devices:
• µPD70F3376A, µPD70F3377A devices of V850ES/FG3 products
• µPD70F3379, µPD70F3380, µPD70F3381, µPD70F3382 devices of V850ES/FJ3 products
• V850ES/FK3 products.</td></tr>
<tr><td>3</td><td>PLLO</td><td>PLL output clock selection.
0: $f_{PLL} = f_{PLLO}$
1: $f_{PLL} = f_{PLLO}/2$</td></tr>
<tr><td>2</td><td>PRSI</td><td>Peripheral clock selection.
0: f_{XP1}, $f_{XP2} = f_{XX}$
1: f_{XP1}, $f_{XP2} = f_{XX}/2$
<b>Note:</b> Set the PRSI bit = “1” in case of 32 MHz < $f_{XX} \leq$ 48 MHz.</td></tr>
<tr><td>1, 0</td><td>PN[1:0]</td><td>Selection of PLL input clock to PLL.
<table>
<tr><th>PLLI1</th><th>PLLI0</th><th>PLL input clock to PLL</th></tr>
<tr><td>0</td><td>0</td><td>$f_{PLLI} = f_X$</td></tr>
<tr><td>0</td><td>1</td><td>$f_{PLLI} = f_X/2$</td></tr>
<tr><td>1</td><td>×</td><td>$f_{PLLI} = f_X/4$</td></tr>
</table></td></tr>
</table>

7.7 Device Information

7.7.1 PRDSELL register - Product selection code register

The 16-bit PRDSELL register specifies the product name of the device.

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

Address FFFFFCC8$_H$

Initial Value Device depending
(for details refer to *Table 7-15*)

	15	14	13	12	11	10	9	8
PRDSELL	PN15	PN14	PN13	PN12	PN11	PN10	PN9	PN8
	R	R	R	R	R	R	R	R

	7	6	5	4	3	2	1	0
	PN7	PN6	PN5	PN4	0	0	PN1	PN0
	R	R	R	R	R	R	R	R

Table 7-15 SELCNT0 register contents (1/2)

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>15 to 4</td><td>PN[15:4]</td><td>Specifies the product name.
<table>
<tr><th colspan="2">PN[15:4]</th><th>Product name</th></tr>
<tr><td>0011 0111 0000$_B$</td><td>370$_H$</td><td>µPD70F3370</td></tr>
<tr><td>0011 0111 0001$_B$</td><td>371$_H$</td><td>µPD70F3371</td></tr>
<tr><td>0011 0111 0010$_B$</td><td>372$_H$</td><td>µPD70F3372</td></tr>
<tr><td>0011 0111 0011$_B$</td><td>373$_H$</td><td>µPD70F3373</td></tr>
<tr><td>0011 0111 0100$_B$</td><td>374$_H$</td><td>µPD70F3374</td></tr>
<tr><td>0011 0111 0101$_B$</td><td>375$_H$</td><td>µPD70F3375</td></tr>
<tr><td>0011 0111 0110$_B$</td><td>376$_H$</td><td>µPD70F3376</td></tr>
<tr><td>0011 0111 0111$_B$</td><td>377$_H$</td><td>µPD70F3377</td></tr>
<tr><td>0011 0111 1000$_B$</td><td>378$_H$</td><td>µPD70F3378</td></tr>
<tr><td>0011 0111 1001$_B$</td><td>379$_H$</td><td>µPD70F3379</td></tr>
<tr><td>0011 1000 0000$_B$</td><td>380$_H$</td><td>µPD70F3380</td></tr>
<tr><td>0011 1000 0001$_B$</td><td>381$_H$</td><td>µPD70F3381</td></tr>
<tr><td>0011 1000 0010$_B$</td><td>382$_H$</td><td>µPD70F3382</td></tr>
<tr><td>0011 1000 0011$_B$</td><td>383$_H$</td><td>µPD70F3383</td></tr>
<tr><td>0011 1000 0100$_B$</td><td>384$_H$</td><td>µPD70F3384</td></tr>
<tr><td>0011 1000 0101$_B$</td><td>385$_H$</td><td>µPD70F3385</td></tr>
</table>
</td></tr>
</table>

Table 7-15 SELCNT0 register contents (2/2)

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>1, 0</td><td>PN[1:0]</td><td>Specifies the suffix code.
<table>
<tr><th>PN1</th><th>PN0</th><th>Suffix code</th></tr>
<tr><td>0</td><td>0</td><td>None</td></tr>
<tr><td>0</td><td>1</td><td>A</td></tr>
<tr><td>1</td><td>0</td><td>B</td></tr>
<tr><td>1</td><td>1</td><td>H</td></tr>
</table></td></tr>
</table>

7.7.2 PRDSELH register - Product selection code register

The 16-bit PRDSELH register specifies the RAM start address of the device.

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

Address FFFFFCCA$_H$

Initial Value Device depending
(for details refer to *Table 7-16*)

	15	14	13	12	11	10	9	8
PRDSELH	×	×	×	×	×	×	×	×
	R	R	R	R	R	R	R	R

7	6	5	4	3	2	1	0
×	×	×	×	RAM3	RAM2	RAM1	RAM0
R	R	R	R	R	R	R	R

Table 7-16 SELCNT0 register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>3 to 0</td><td>RAM[3:0]</td><td>Specifies the start address of the internal RAM.
<table>
<tr><th>RAM3</th><th>RAM2</th><th>RAM1</th><th>RAM0</th><th>RAM start address</th></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>03FFD000H</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td><td>03FFB000H</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td><td>03FF9000H</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>1</td><td>03FF7000H</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</td><td>03FF5000H</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>1</td><td>03FF3000H</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>0</td><td>03FF0000H</td></tr>
</table></td></tr>
</table>

Chapter 8 Data Protection and Security

8.1 Overview

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

Security functions

Security functions provide countermeasures against unexpected failures during reprogramming processes. These are basically:

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

These functions are described in detail in *“Flash Memory” on page 298*.

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 in 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.

8.2 N-Wire Debug Interface Protection

In general read-out of the flash memory contents is possible via the N-Wire debug interface, but protection against illegal read-out can be enabled. For protection of the flash memory, the usage of 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), 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 8-1*

Table 8-1 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 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.

8.3 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 access from external 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 is 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 set respectively reset by an external flash programmer, provided the effective protection level allows to do so.
In self-programming mode the effective protection flags can not be reset, but other ones 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.

(1) Program protection flag

Set this flag to disable the programming function via external flash programmer interfaces.
No flash memory content can be written from external, 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 further on possible.

(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, 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 is not erased by this means. I.e. protection flags, variable reset vector, etc. are still valid.

(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 further on possible.

(4) Read-out protection flag

Set this flag to disable the feature that allows reading back the flash memory via external flash programmer 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 further on possible.

(5) Boot cluster protection flag

Set this flag to disable erasure and rewrite of the boot cluster.
The boot cluster can not be manipulated in any way (no erase/write).

This applies in serial and self-programming mode.

Once this flag is set, it is impossible to reset this flag. Thus the boot cluster content can not be changed any more.

For the explanation of the boot cluster refer to *“Secure self-programming (boot cluster swapping)” on page 323*.

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 8-2 Protection functions overview

Function	Functional outline	Programming method	
		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 cluster prohibit	Erasure (by block or chip erase) or writing of the boot cluster impossible. Once rewrite protection of the boot cluster is enabled, it can not be disabled any more.	√	√

Note √: applicable, ×: not applicable

Table 8-3 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 8-4 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

Chapter 9 Bus and Memory Control (BCU, MEMC)

Besides providing access to on-chip peripheral I/Os, the microcontroller products V850ES/FJ3 and V850ES/FK3 support access to external memory devices (such as external ROM and RAM) and external I/O. The Bus Control Unit BCU and Memory Controller MEMC control the access to on-chip peripheral I/Os and to external devices.

Furthermore, the data flash area, available with all V850ES/Fx3 products can be allocated via the external memory area.

Note Though the V850ES/FE3, V850ES/FF3 and V850ES/FG3 products do not provide access to external memory, the configuration registers for external memory access have to be set up correctly for proper access to the internal data flash.

9.1 Overview

The following external devices can be connected to the microcontroller device:

- SRAM
- ROM
- External I/O

Features summary The bus and memory control of the microcontroller device provides:

- 16 address/data signals (AD0 to AD15)
- Selectable data bus width for each chip select area (8 bits and 16 bits)
- 4 chip select signals externally available ($\overline{\text{CS0}}$ to $\overline{\text{CS3}}$)
- Access to memory takes a minimum of three CPU clock cycles
- Address setup/hold wait state can be inserted for each chip select area
- Up to 7 data wait states can be inserted for each chip select area (programmable wait)
- External data wait function through $\overline{\text{WAIT}}$ pin
- Idle state can be inserted for each chip select area
- Bus hold function
- Fixed to little endian format

9.2 Description

The figure below shows a block diagram of the modules that are necessary for accessing on-chip peripherals, external memory, external I/O, or data flash.

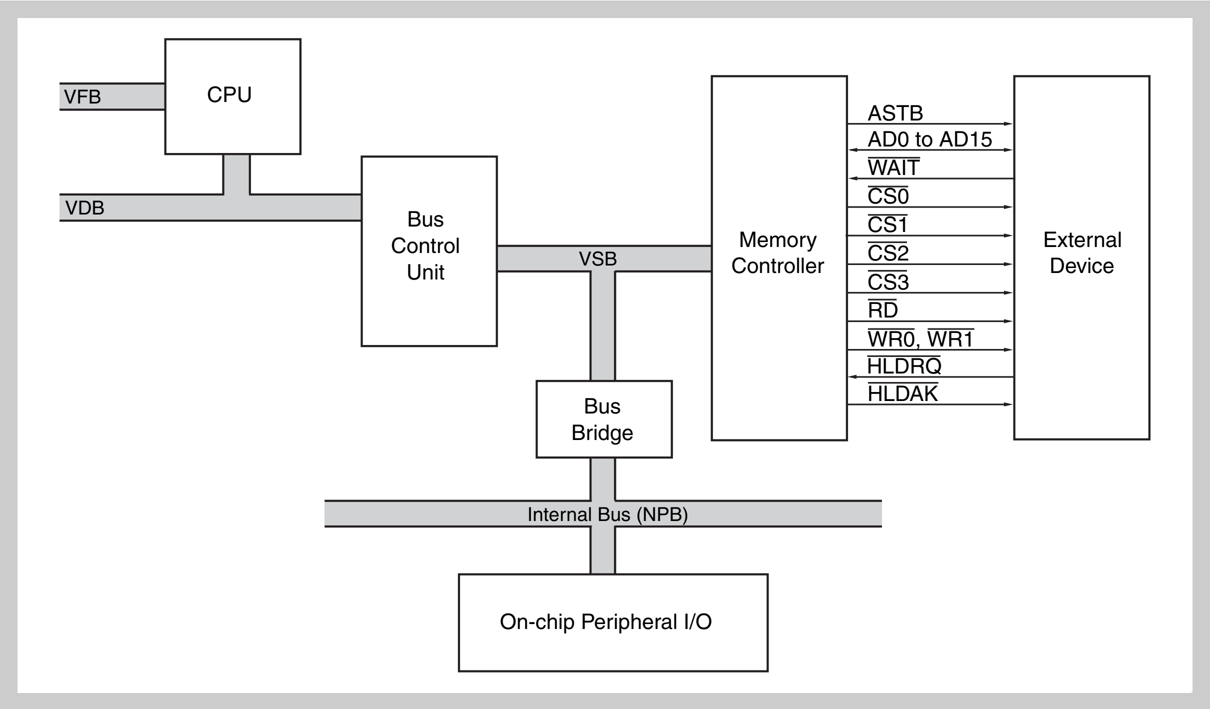


Figure 9-1 Bus Control Unit and Memory Control block diagram

Busses The busses are abbreviated as follows:

- NPB: Peripheral bus
- VSB: V850 system bus
- VDB: V850 data bus
- VFB: V850 fetch bus

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

For access to external devices, the BCU generates the necessary control signals (chip select signals) for the Memory Controller.

Memory Controller The 64 MB address range is divided into 2-MB, 4-MB and 8-MB memory blocks. Each of the memory blocks can be assigned to an external device.

If an instruction uses such an address, a chip select signal is generated. The device supports four chip select signals ($\overline{\text{CS0}}$ to $\overline{\text{CS3}}$). Each chip select signal covers a certain address range, also called “chip select area”. For details see *“Memory blocks and chip select signals” on page 342*.

The Memory Controller generates the control signals for access to the external devices. For example, it generates the read strobe ($\overline{\text{RD}}$) and the write strobes ($\overline{\text{WR0}}$, $\overline{\text{WR1}}$). From the 26 bit address of the CPU, the lower 16 bits are passed to the external device.

The external signals of the Memory Controller are listed in the following table:

Table 9-1 Memory Controller external connections

Signal name	I/O	Active level	Pins	Function
$\overline{\text{CS0}}$	O	L	$\overline{\text{CS0}}$	Chip select signal
$\overline{\text{CS1}}$	O	L	$\overline{\text{CS1}}$	Chip select signal
$\overline{\text{CS2}}$	O	L	$\overline{\text{CS2}}$	Chip select signal
$\overline{\text{CS3}}$	O	L	$\overline{\text{CS3}}$	Chip select signal
AD[0:15]	I/O	–	AD0 to AD15	Address/data bus
ASTB	O	–	ASTB	Address strobe
$\overline{\text{WAIT}}$	I	L	$\overline{\text{WAIT}}$	Data wait
$\overline{\text{WR0}}$	O	L	$\overline{\text{WR0}}$	Write strobe (lower 8 bits)
$\overline{\text{WR1}}$	O	L	$\overline{\text{WR1}}$	Write strobe (higher 8 bits)
$\overline{\text{RD}}$	O	L	$\overline{\text{RD}}$	Read strobe
$\overline{\text{HLDRQ}}$	I	L	$\overline{\text{HLDRQ}}$	Bus hold control
$\overline{\text{HLDAK}}$	O	L	$\overline{\text{HLDAK}}$	

All pins are in input port mode after reset. Refer to *“Pin Functions” on page 32*.

Note If the concerned pins are configured as external memory bus pins, change between input and output is performed automatically by Memory Controller’s read and write operations.

Configuration The microcontroller device supports interfacing with various memory devices. To make the Bus and Memory Controller suitable for the connected device, the wait functions and idle state insertions can be configured.

For a detailed description, see *“Configuration of Memory Access” on page 359*.

9.2.1 Memory blocks and chip select signals

The 64 MB address range is divided into memory blocks. Each memory block is assigned to a chip select ($\overline{\text{CS}}$) signal. If a memory block is configured for external access, access to that memory block generates the corresponding chip select signal (see *Figure 9-2 on page 342*). The memory block that activates a chip select signal is also called chip select area.

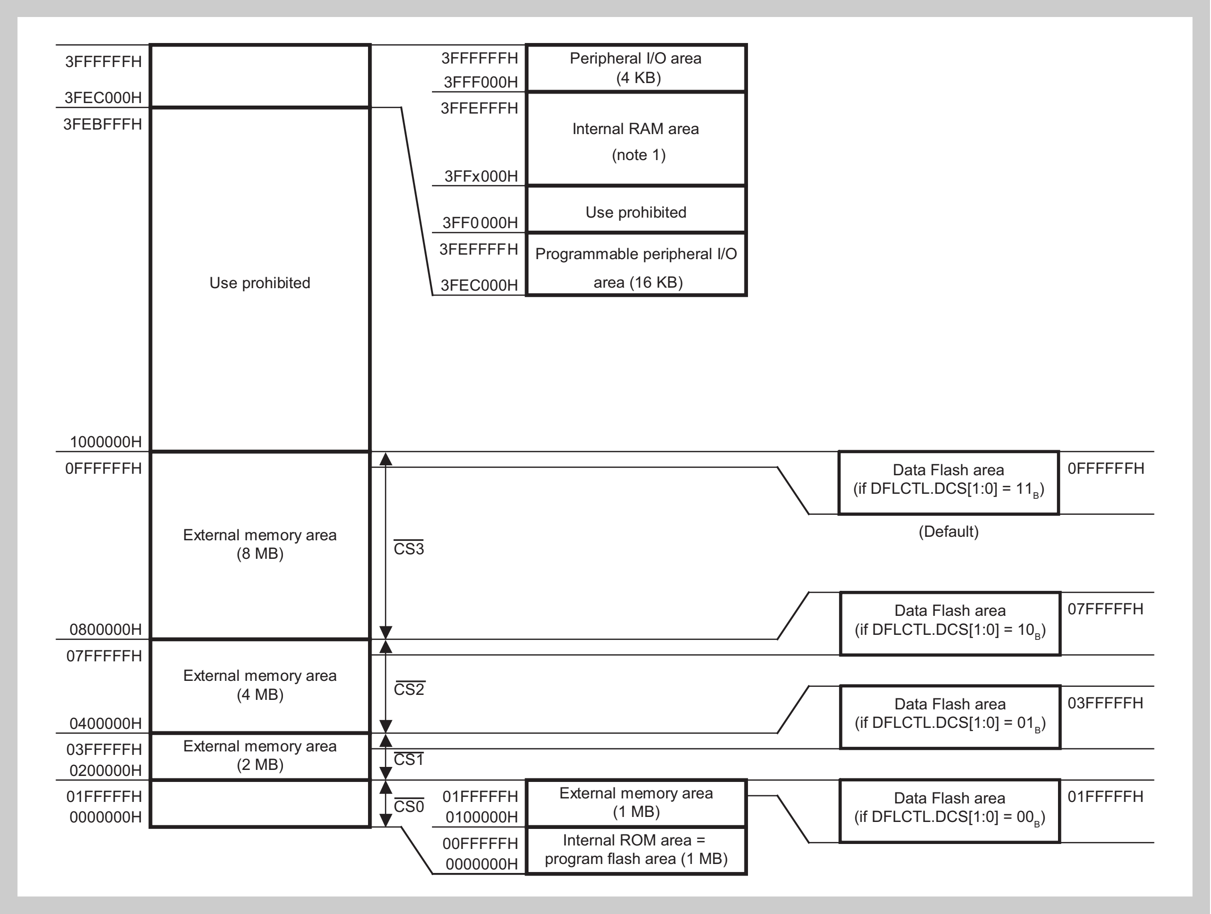


Figure 9-2 Memory blocks and chip select signals

Note

1. Size and start address of the internal RAM area depend on the product derivative. See *"CPU System Functions" on page 155* for details.
2. Throughout this chapter, the individual chip select areas are identified by "k" (k = 0 to 3), for example $\overline{\text{CSk}}$ for the chip select signal k or BSC.BSCk[1:0] for setting the data bus width of chip select area k.
3. The lower 1 MB of the memory area is always mapped to the internal code flash memory. Thus, external memory mapped to this area can not be addressed in normal operation mode.
4. The data flash area can be mapped to the upper boundary of any of the chip select areas. For more details on the data flash, see *"Flash Memory" on page 298*.

9.2.2 Peripheral I/O area

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 9-2 Peripheral I/O areas

Name	Address range	Size
Fixed peripheral I/O area	03FF $F000_H$ to 03FF $FFFF_H$	4 KB
Programmable peripheral I/O area (PPA)	03FE $C000_H$ to 03FE $EFFF_H$	16 KB

(1) Fixed peripheral I/O area

The fixed peripheral I/O area holds the registers of the on-chip peripheral I/O functions.

Note Because the address space covers 64 MB, the address bits A[31:26] are not considered. Therefore, in this manual, all addresses of peripheral I/O registers in the 4 KB peripheral I/O area are given in the range FFFF $F000_H$ to FFFF $FFFF_H$ instead of 03FF $F000_H$ to 03FF $FFFF_H$.

(2) Programmable peripheral I/O area (PPA)

With this microcontroller, usage and address range of the PPA are *not* 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).

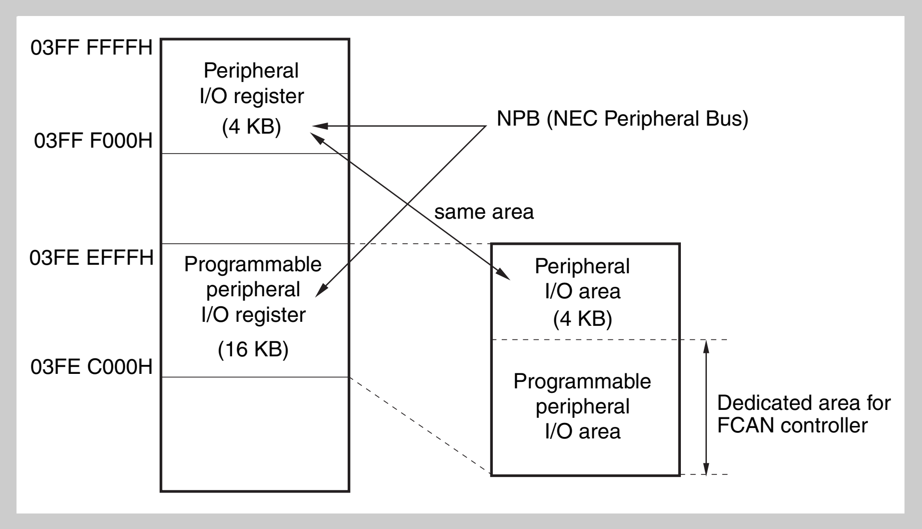

03FF FFFFH
03FF F000H
03FE EFFFH
03FE C000H
Peripheral I/O register (4 KB)
Programmable peripheral I/O register (16 KB)
NPB (NEC Peripheral Bus)
same area
Peripheral I/O area (4 KB)
Programmable peripheral I/O area
Dedicated area for FCAN controller

Figure 9-3 Programmable peripheral I/O area

The CAN modules registers and message buffers are allocated to the PPA. Refer to *"CAN module register and message buffer addresses" on page 700* for information how to calculate the register and message buffer addresses of the CAN modules.

Note
1. The *fixed* peripheral I/O area is mirrored to the upper 4 KB of the *programmable* peripheral I/O area. If data is written in one area, data having the same contents is also written in the other area.
2. To make software suitable for both microcontroller and emulation tool, it is recommended to include the set up of the base address in the software. See *"BPC - Peripheral area selection control register" on page 352*.

9.2.3 NPB access timing

All accesses to the peripheral I/O areas are passed over to the NPB bus via the VSB - NPB bus bridge BBR. Read and write access times to registers via the NPB depend on the register (refer to *“Registers Access Times” on page 971*), the system clock VBCLK and the setting of the VSWC register.

The CPU operation during an access to a register via the NPB depends also on the kind of peripheral I/O area:

- Fixed peripheral I/O area
  During a read or write access the CPU operation stops until the access via the NPB is completed.
- Programmable peripheral I/O area
  During a read access the CPU operation stops until the read access via the NPB is completed.
  During a write access the CPU operation continues operation, provided any preceded NPB access is already finished. If a preceded NPB access is still ongoing the CPU stops until this access is finished and the NPB is cleared.

Caution Pay attention at write accesses to NPB peripheral I/O registers via the programmable peripheral I/O area.

Since the CPU may continue operation, even though the data has not yet been transferred to its destination register, inconsistencies may occur between the program flow and the status of the registers.

In particular register set-ups which change an operational status of a certain module require special notice, like, for instance, masking/unmasking of interrupts via maskable interrupt control registers xxIC, enabling/disabling timers, etc.

9.2.4 Bus properties

This section summarizes the properties of the internal and external bus.

(1) Bus width

The microcontroller device accesses external memory and external I/O in 8-bit or 16-bit units.

The data bus size for each chip select area is specified in the bus size configuration register (BSC).

The operation for each type of access is given in *“Access to 8-bit data busses” on page 367* and in *“Access to 16-bit data busses” on page 370*.

(2) Bus priority order

There are several kinds of external bus cycles as shown below. The bus hold has the highest priority, followed by the DMA cycle, the operand data access, and instruction fetch, in that order.

Table 9-3 Bus priority order

Priority	External bus cycle	Bus master
High	Bus hold	External device
	DMA cycle	DMA Controller
	Operand data access	CPU
	Instruction fetch (branch)	CPU
Low	Instruction fetch (successive)	CPU

(3) Bus access

The number of CPU clocks necessary for accessing each resource is as follows:

Table 9-4 Number of bus access clocks

Bus cycle configuration		Internal ROM (32 bits)		Internal RAM (32 bits)	External memory (16 bits)
		with branch latency 2	with branch latecancy 3		
Instruction fetch	Normal access	1	1	1[a]	3 + n[b]
	Branch	2	3	2[a]	3 + n[b]
Operand data access		3	4	1	3 + n[b]

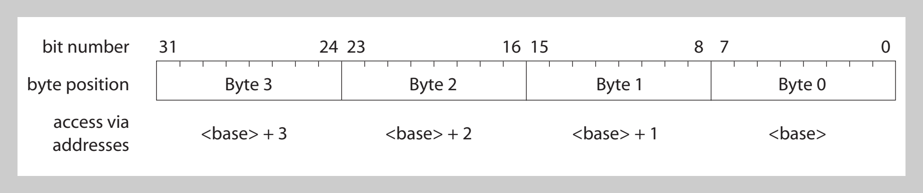
[a] In case of contention with data access, one cycle is added.
[b] n is the number of inserted wait states

Note Unit: Clocks/access

(4) Endian format

The endian format is fixed to little endian format.

The endian format defines the byte order in which word data is stored. “Little Endian” means that the low-order byte of the word is stored in memory at the lowest address, and the high-order byte at the highest address. Therefore, the base address of the word addresses the low-order byte:



bit number	31 24	23 16	15 8	7 0
byte position	Byte 3	Byte 2	Byte 1	Byte 0
access via addresses	<base> + 3	<base> + 2	<base> + 1	<base>

Figure 9-4 Little endian addresses within a word

9.2.5 Boundary operation conditions

The microcontroller device has the following boundary operation conditions:

(1) Program space

Instruction fetches from the internal peripheral I/O area are inhibited and yield NOP operations.

If a branch instruction exists at the upper limit of the internal RAM area, a pre-fetch operation (invalid fetch) that straddles over the internal peripheral I/O area does not occur.

(2) Data space

The microcontroller device is provided with an address misalign function.

By this function, data of any format (word: 32 bit, halfword: 16 bit, byte: 8 bit) can be placed to any address in memory, even though the address is not aligned to the data format (that means address 4n for words, address 2n for halfwords).

- Unaligned halfword data access
  When the LSB of the address is A0 = 1, two byte accesses are performed.
- Unaligned word data access
  - When the LSB of the address is A0 = 1, two byte and one halfword accesses are performed. In total it takes 3 bus cycles.
  - When the LSBs of the address are A[1:0] =10_B, two halfword accesses are performed.

Note Accessing data on misaligned addresses takes more than one bus cycle to complete data read/write. Consequently, the bus efficiency will drop.

9.2.6 Initialization for access to external devices

To enable access to external devices, initialize the following registers after any reset.

1. Bus size configuration register BSC
   Set the data bus width for the active chip select areas.
2. Data wait control registers DWCn
   Set the number of data wait states with respect to the starting bus cycle.
3. Bus cycle control register BCC
   Set the number of idle states for each chip select area k = 0 to 3.

Caution

1. Do not change these registers after initialization.
2. Do not access external devices before initialization is finished.

9.2.7 Bus hold function

The bus hold function enables the configuration of multi-processor type systems in which two or more bus masters exist.

During bus hold, the external address/data bus is released. Execution of the program in the internal ROM and internal RAM is continued until a peripheral I/O register or the external memory is accessed.

(1) Entering/releasing bus hold state

The bus hold state is entered when a low level signal is applied to the HLDRQ pin. The microcontroller sets HLDAK to low level, which indicates the release of the external bus.

Exceptions:

- The bus hold state is not entered in STOP, IDLE1, IDLE2, Sub IDLE mode. This is due to the stopped internal system clock.
- The bus hold state is not entered in idle state.
- The bus hold state is not entered during a multiple-access cycle initiated by the bus sizing function or by a bit manipulation instruction. *Table 9-5* lists the timing at which HLDRQ is not acknowledged.

Table 9-5 Timing at which HLDRQ is not acknowledged

Status	**Data bus width**	**Access type**	**Timing at which HLDRQ is not acknowledged**
CPU bus lock	16 bits	Word access to even address	Between first and second access
		Word access to odd address	Between first and second access
			Between second and third access
		Halfword access to odd address	Between first and second access
	8 bit	Word access	Between first and second access
			Between second and third access
			Between third and fourth access
		Halfword access	Between first and second access
Ready-modify-write access of bit manipulation instruction	–	–	Between read access and write access

The bus hold state is released when a high level signal is applied to the HLDRQ pin. The HLDAK pin returns to high level.

(2) Monitoring bus hold state

At the HLDAK pin, the bus hold state can be monitored:

- low level at HLDAK: bus is released (bus hold)
- high level HLDAK: microcontroller is bus master (no bus hold)

(3) Bus hold procedure

The bus hold transition procedure is shown in *Figure 9-5*:

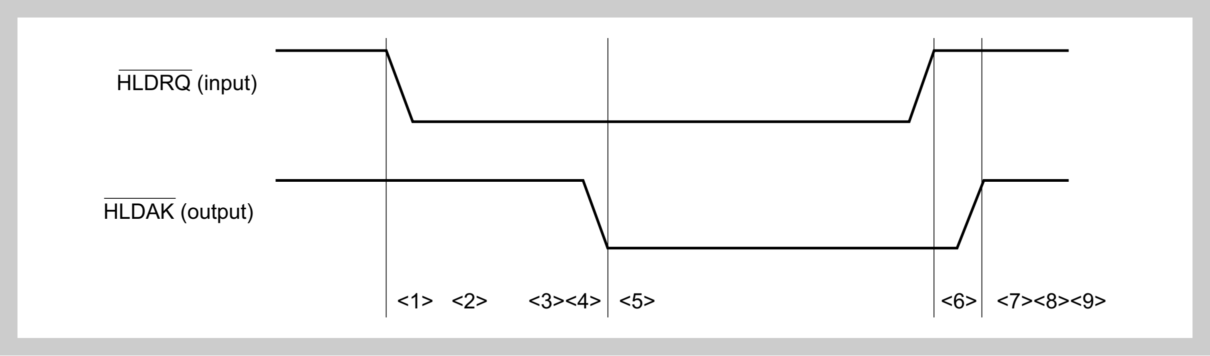

$\overline{\text{HLDRQ}}$ (input)
$\overline{\text{HLDAK}}$ (output)
<1>
<2>
<3><4>
<5>
<6>
<7><8><9>

Figure 9-5 Bus hold state transition

The procedure steps are described below:

1. $\overline{\text{HLDRQ}}$ = 0 acknowledged 2. All bus cycle start requests inhibited 3. End of current bus cycle 4. Shift to bus idle status	normal status
5. $\overline{\text{HLDAK}}$ = 0 … 6. $\overline{\text{HLDRQ}}$ = 1 acknowledged	bus hold state
7. $\overline{\text{HLDAK}}$ = 1 8. Bus cycle start request inhibition released 9. Bus cycle starts	normal status

Operation in power save mode

Because the internal system clock is stopped in STOP, IDLE1, IDLE2 and Sub IDLE modes, the bus hold status is not entered even if the $\overline{\text{HLDRQ}}$ pin is asserted.

In the HALT mode, the bus hold status is entered immediately after $\overline{\text{HLDRQ}}$ is set to low level. Thus $\overline{\text{HLDAK}}$ pin is set to low level also.

When the $\overline{\text{HLDRQ}}$ pin is deasserted, the $\overline{\text{HLDAK}}$ pin returns to high level, and the bus hold status is cleared.

9.2.8 Pin status

This section presents the pin status during access to internal memory, in idle state and during bus hold.

For the pin status after reset and in power save modes, see *“Pin Functions” on page 32*.

(1) Pin status during access to internal memory

Then following table shows the pin status when internal ROM, internal RAM, or on-chip peripheral I/O is accessed:

Table 9-6 Pin status during access to internal memory

Access destination	Address bus AD[15:0] with ASTB high	Data bus AD[15:0] with ASTB low	Control signals
Internal ROM	undefined	Hi-Z	Inactive
Internal RAM	undefined	Hi-Z	Inactive
On-chip peripheral I/O	see Note	Hi-Z	Inactive

Note When an on-chip peripheral I/O is accessed, the address of the on-chip peripheral I/O being accessed is output via the address bus.

(2) Pin status in idle state and during bus hold

The pin status in idle state and during bus hold is given in the following table:

Table 9-7 Pin status in idle state and during bus hold

Pin	Status in idle state[a]	Status during bus hold	Status during IDLE mode and STOP mode
AD[15:0]	Held	Hi-Z	Hi-Z
$\overline{\text{WR0}}$, $\overline{\text{WR1}}$, $\overline{\text{RD}}$, ASTB	H	Hi-Z	H
CLKOUT	Operating	Operating	L
$\overline{\text{HLDAK}}$	H	L	H
$\overline{\text{HLDRQ}}$	–	Operating	–
$\overline{\text{CS0}}$ to $\overline{\text{CS3}}$	Held	Hi-Z	H

[a] Idle state is the bus state TI between two bus access cycles, when set up by BCC.BCk1 = 1.

9.3 Registers

Access to on-chip peripherals, to external memory, and to external I/O is controlled and operated by registers of the Bus Control Unit (BCU) and the Memory Controller:

Table 9-8 Bus and memory control register overview

Module	Register name	Shortcut	Address
Bus Control Unit (BCU)	Peripheral area selection control register	BPC	FFFF F064$_H$
	Bus size configuration register	BSC	FFFF F066$_H$
	Internal peripheral function wait control register	VSWC	FFFF F06E$_H$
Memory Controller	Address setup wait control register	AWC	FFFF F488$_H$
	Data wait control registers	DWC0	FFFF F484$_H$
	Bus cycle control register	BCC	FFFF F48A$_H$

9.3.1 BCU registers

The following registers are part of the BCU. They define the usage of the programmable peripheral I/O area (PPA) and the data bus width.

(1) BPC - Peripheral area selection control register

The 16-bit BPC register enables/disables the PPA and it determines the starting address of the PPA.

- For the microcontroller, the base address of the PPA is fixed to 03FE C000$_H$. Thus writing to BPC.PA[13:0] does not change the PPA base address. Nevertheless the PPA must be enabled by setting BPC.PA15 = 1.
- For the emulation tool, the PPA has to be enabled and the base address has to be set up by writing 8FFB$_H$ to the BPC register.

To make software suitable for both microcontroller and emulation tool, it is recommended to include the set up of the PPA with BPC = 8FFB$_H$ in the software.

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

Address FFFF F064$_H$

Initial Value 0000$_H$

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BPC	PA15	0	PA13	PA12	PA11	PA10	PA9	PA8	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0

Table 9-9 BPC register contents

Bit Position	Bit Name	Function
15	PA15	Select usage of programmable peripheral I/O area (PPA). 0: PPA disabled 1: PPA enabled
11 to 0	PA[13:0]	Bits PA[13:0] specify bits 27 to 14 of the starting address of the PPA. The other bits of the address are fixed to 0.

Caution The bits marked with 0 must always be 0.
The base address PBA of the programmable peripheral area sets the start address of the 16 KB PPA in a range of 256 MB. The 256 MB page is mirrored 16 times to the entire 32-bit address range.

The base address PBA is calculated by

$$PBA = BPC.PA[13:0] \text{ x } 2^{14}$$

Table 9-10 shows how the addresses of the programmable peripheral area are assembled. The base address PBA is highlighted.

Table 9-10 Address range of programmable peripheral area (12 KB)

31	…	28	27 … 14	13	…	1	0	bit
0	…	0	BPC.PA[13:0]	1	…	1	1	
	…		…		…			
0	…	0	BPC.PA[13:0]	0	…	0	1	
0	…	0	BPC.PA[13:0]	0	…	0	0	PBA

(2) BSC - Bus size configuration register

The 16-bit BSC register controls the data bus width for each chip select area.

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

Address FFFF $F066_H$

Initial Value 5555_H (must be initialized correctly, refer to the Caution below)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BSC	0	1	0	1	0	1	0	1	0	BS30	0	BS20	0	BS10	0	BS00
										$\overline{CS3}$		$\overline{CS2}$		$\overline{CS1}$		$\overline{CS0}$

Table 9-11 BSC register contents

Bit position	Bit name	Function
6, 4, 2, 0	BSk0 (k = 0 to 3)	Sets the data bus width for each chip select area k: 0: 8 bit 1: 16 bit

Caution

1. The bits marked with 0 must always be 0.
   The bits marked with 1 must always be 1.
2. To initialize an external memory area after a reset, register BSC has to be set. Do not change this register after initialization. Do not access external devices before initialization is finished.

Data flash access To access the data flash via chip select area n, set BSC.BSn0 = 1 (bus size 16 bit).

(3) VSWC - Internal peripheral function wait control register

The 8-bit VSWC register controls the bus access wait for the on-chip peripheral I/O registers. The 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.

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

Address FFFF F06E$_H$

Initial Value 77$_H$

	7	6	5	4	3	2	1	0
VSWC	0	SUWL2	SUWL1	SUWL0	0	VSWL2	VSWL1	VSWL0
	R	R/W	R/W	R/W	R	R/W	R/W	R/W

Table 9-12 VSWC register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th colspan="4">Function</th></tr>
<tr><td rowspan="10">6 to 4</td><td rowspan="10">SUWL[2:0]</td><td colspan="4">Address setup wait for internal bus:</td></tr>
<tr><th>SUWL2</th><th>SUWL1</th><th>SUWL0</th><th>Number of address setup wait states</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1 CPU system clock (VBCLK)</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>2 CPU system clock (VBCLK)</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>3 CPU system clock (VBCLK)</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>4 CPU system clock (VBCLK)</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>5 CPU system clock (VBCLK)</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>6 CPU system clock (VBCLK)</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>7 CPU system clock (VBCLK)</td></tr>
<tr><td rowspan="10">2 to 0</td><td rowspan="10">VSWL[2:0]</td><td colspan="4">Data wait for internal bus:</td></tr>
<tr><th>VSWL2</th><th>VSWL1</th><th>VSWL0</th><th>Number of data wait states</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1 CPU system clock (VBCLK)</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>2 CPU system clock (VBCLK)</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>3 CPU system clock (VBCLK)</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>4 CPU system clock (VBCLK)</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>5 CPU system clock (VBCLK)</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>6 CPU system clock (VBCLK)</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>7 CPU system clock (VBCLK)</td></tr>
</table>

The following setups are recommended for VSWC:

Table 9-13 Recommended timing for internal bus

System clock (f_{CPU})	≤16 MHz	≤25 MHz	≤33 MHz	≤48 MHz
SUWL	0	0	1	1
VSWL	0	1	1	2
VSWC	00_H	01_H	11_H	12_H

Note

1. The bits marked with 0 must always be 0.
2. This register must be initialized after $\overline{\text{RESET}}$.

9.3.2 Memory Controller registers

The following registers are part of the Memory Controller. They specify the number of data wait states, the number of address wait states, and the number of idle states.

(1) AWC - Address setup wait control register

The 16-bit AWC register controls the insertion of an address setup wait before and address hold wait state after the T1 cycle. The address setup wait and address hold wait state can be enabled for each chip select area.

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

Address FFFF F488$_H$

Initial Value FFFF$_H$: After system setup, by default, address hold and wait states insertion is enabled for each chip select area.
(This register must be initialized correctly, refer to the Caution below.)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AWC	1	1	1	1	1	1	1	1	AHW3	ASW3	AHW2	ASW2	AHW1	ASW1	AHW0	ASW0
									$\overline{\text{CS3}}$		$\overline{\text{CS2}}$		$\overline{\text{CS1}}$		$\overline{\text{CS0}}$	

Table 9-14 AWC register contents

Bit position	Bit name	Function
1, 3, 5, 7	AHWk (k = 0 to 3)	Enables/disables the address hold wait insertion for each chip select area k: 0: No wait state inserted 1: Address hold wait state inserted after T1 cycle
0, 2, 4, 6	ASWk (k = 0 to 3)	Enables/disables the address setup wait insertion for each chip select area k: 0: No wait state inserted 1: Address setup wait state inserted before T1 cycle

Caution

1. The bits marked with 1 must always be 1.
2. To initialize an external memory area after a reset, register AWC has to be set. Do not change this register after initialization. Do not access external devices before initialization is finished.

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

Data flash access To access the data flash via chip select area n, use the following settings:

- AWC.AHWn = 0 (no address hold wait state)
- for fxx ≤ 24 MHz: AWC.ASWn = 0 (no address setup wait state)
  for 24 MHz < fxx ≤ 48 MHz: AWC.ASWn = 1 (one address setup wait state)

(2) DWC0 - Data wait control register

The 16-bit DWC0 register controls the number of wait states after the T2 cycle. Each chip select area is controlled separately. A maximum of seven data wait states is possible.

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

Address FFFF F484$_H$

Initial Value 7777$_H$: After system setup, by default, seven data wait states are inserted for each chip select area.
(This register must be initialized correctly, refer to the Caution below.)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DWC0	0	DW32	DW31	DW30	0	DW22	DW21	DW20	0	DW12	DW11	DW10	0	DW02	DW01	DW00
		$\overline{CS3}$				$\overline{CS2}$				$\overline{CS1}$				$\overline{CS0}$		

Table 9-15 DWC0 register contents

Bit position	Bit name	Function	
14 to 12, 10 to 8, 6 to 4, 2 to 0	DWk[2:0] (k = 0 to 3)	Sets the number of wait states after the T2 cycle for each chip select area k.	
		DWk[2:0]	**Number of inserted wait states**
		000$_B$	No wait state inserted
		001$_B$	1 wait state
		010$_B$	2 wait states
		011$_B$	3 wait states
		...	...
		111$_B$	7 wait states

Note For access to internal memory, programmable waits are *not* carried out.

Caution
1. The bits marked with 0 must always be 0.
2. To initialize an external memory area after a reset, this register has to be set. Do not access external devices before initialization is finished. Do not change this register while an external device is accessed.

Data flash access To access the data flash via chip select area n, set
for fxx ≤ 40 MHz: DWC.DW[2:0] =001$_B$ (one data wait state)
for fxx ≤ 48 MHz: DWC.DW[2:0] =010$_B$ (two data wait states)

(3) BCC - Bus cycle control register

The 16-bit BCC register controls the insertion of an idle state after the T3 cycle. Each chip select area is controlled separately.

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

Address FFFF F48A$_H$

Initial Value AAAA$_H$ After system reset, an idle state is inserted.
(This register must be initialized correctly, refer to the Caution below.)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BCC	0	1	0	1	0	1	0	1	BC31	0	BC21	0	BC11	0	BC01	0
									$\overline{\text{CS3}}$		$\overline{\text{CS2}}$		$\overline{\text{CS1}}$		$\overline{\text{CS0}}$	

Table 9-16 BCC register contents

Bit position	Bit name	Function
7, 5, 3, 1	BCk1 (k = 0 to 3)	Enables/disables the idle state insertion for each chip select area k: 0: No idle state inserted 1: Idle state inserted after T3 cycle

Note For access to internal memory, no idle states are inserted.

Caution

1. The bits marked with 0 must always be 0.
   The bits marked with 1 must always be 1.
2. To initialize an external memory area after a reset, this register has to be set. Do not access external devices before initialization is finished. Do not change this register while an external device is accessed.

Data flash access To access the data flash via chip select area n, set BCC.BCn1 = 0 (no idle state inserted).

9.4 Configuration of Memory Access

The microcontroller device supports interfacing with various memory devices. Therefore, the wait functions and idle state insertions can be configured.

9.4.1 Wait function

Several wait functions are supported:

(1) Address setup wait

The microcontroller device allows insertion of an address setup wait state before the first access cycle (T1 state).

The address setup wait state can be enabled by AWC.ASWk = 1 individually for each chip select area.

(2) Address hold wait

The microcontroller device allows insertion of an address hold wait state after the first access cycle (T1 state).

The address hold wait state can be enabled by AWC.AHWk = 1 individually for each chip select area

(3) Programmable wait function

With the purpose of realizing easy interfacing with low-speed memory or with I/Os, it is possible to insert up to seven data wait states after the second access cycle (T2 state).

The number of wait states can be specified by data wait control register DWC0.

(4) External wait function

Each read or write operation takes at least three cycles (T1, T2 and T3). To stretch the access cycle for accessing slow external devices, any number of wait states (TW) can be inserted under external control of the $\overline{\text{WAIT}}$ signal.

The $\overline{\text{WAIT}}$ signal can be set asynchronously from the system clock. The $\overline{\text{WAIT}}$ signal is sampled at the falling edge of the clock in the T2 and TW states. Depending on the level of the $\overline{\text{WAIT}}$ signal at sampling timing, a wait state is inserted or not.

(5) Relationship between programmable wait and external wait

If both programmable wait and external wait ($\overline{\text{WAIT}}$) are applied, an OR relation gives the resulting number of wait cycles. *Figure 9-6* shows that as long as any of the two waits is active, a wait cycle will be performed.

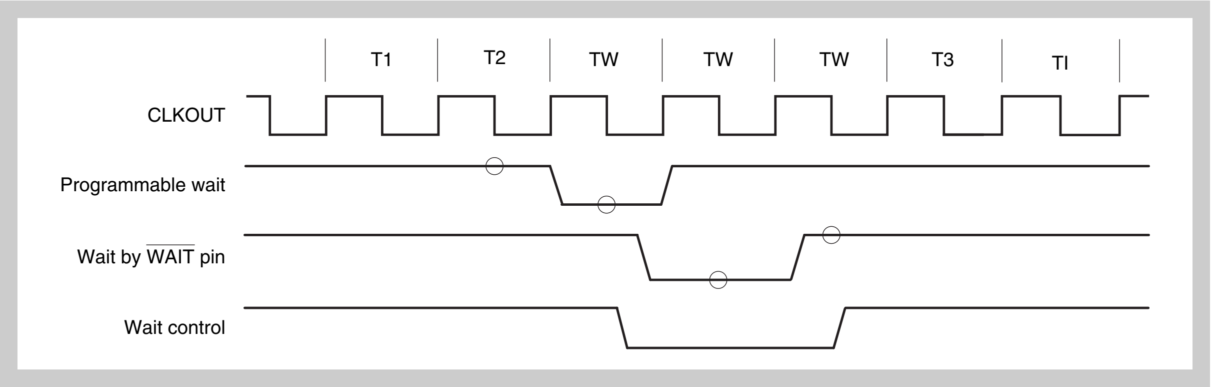


Figure 9-6 Example of wait insertion

Note The circles indicate the sampling timing.

9.4.2 Idle state insertion

To facilitate interfacing with low-speed memory devices, an idle state (TI) can be inserted between two bus cycles, that means after the T3 state. Idle states are inserted to meet the data output flow delay on memory read or write accesses for each chip select area. The next bus cycle is started after the idle state.

The idle state is specified by BCC.BCk1 = 1.

9.5 External Devices Interface Timing

This section presents examples of write and read operations. The states are abbreviated as:

- T1, T2 and T3 states: Basic states for access.
- TW state: Wait state that is inserted according to the DWC0 register settings and according to the $\overline{\text{WAIT}}$ input.
- TASW state: Address setup wait state that is inserted according to the AWC register settings.
- TAHW state: Address hold wait state that is inserted according to the AWC register settings.
- TI state: Idle state that is inserted according to the BCC register settings.
- TH state: Bus hold state that is entered according to the $\overline{\text{HLDRQ}}$ input.

9.5.1 Writing to external devices

This section shows typical sequences of writing data to external devices.

The microcontroller uses both low and high byte write strobes:

Table 9-17 Write access with low and high byte write strobes

WR1	WR0	Write access
0	0	16-bit write
0	1	8-bit odd address write
1	0	8-bit even address write
1	1	–

(1) Write with wait cycles and idle state insertion (bus size: 16-bit)

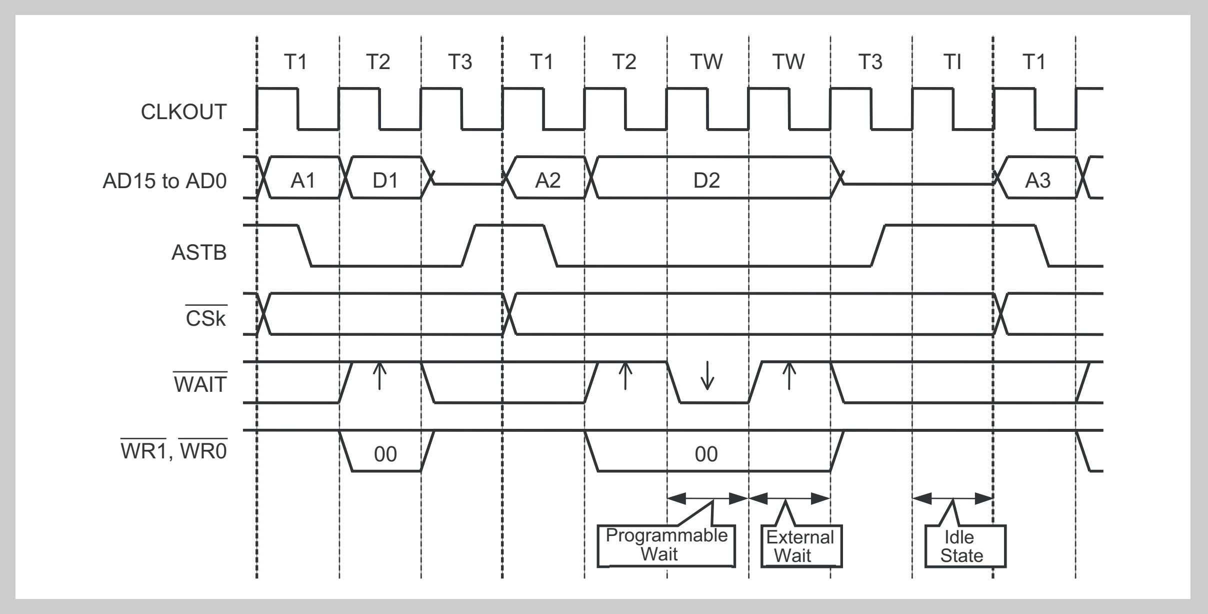


Figure 9-7 Timing: write data with external and programmable wait cycles and idle state insertion (bus size: 16-bit)

Register settings:

- BSC.BSk0 = 1_B (16 bit data bus size)
- AWC.AHWk = AWC.ASWk = 0 (no address setup/hold wait states inserted)
- DWC0.DWk[2:0] = 001_B (one programmable data wait state inserted)
- BCC.BCk1 = 1_B (one idle state inserted)

Note

1. The arrows indicate the sampling timing.
2. AD[7:0] holds the address for accessing the odd address byte
   AD[15:8] holds the address for accessing the even address byte
3. $\overline{\text{CSk}}$ with k = 0 to 3.

(2) Write with wait cycles and idle state insertion (bus size: 8-bit)

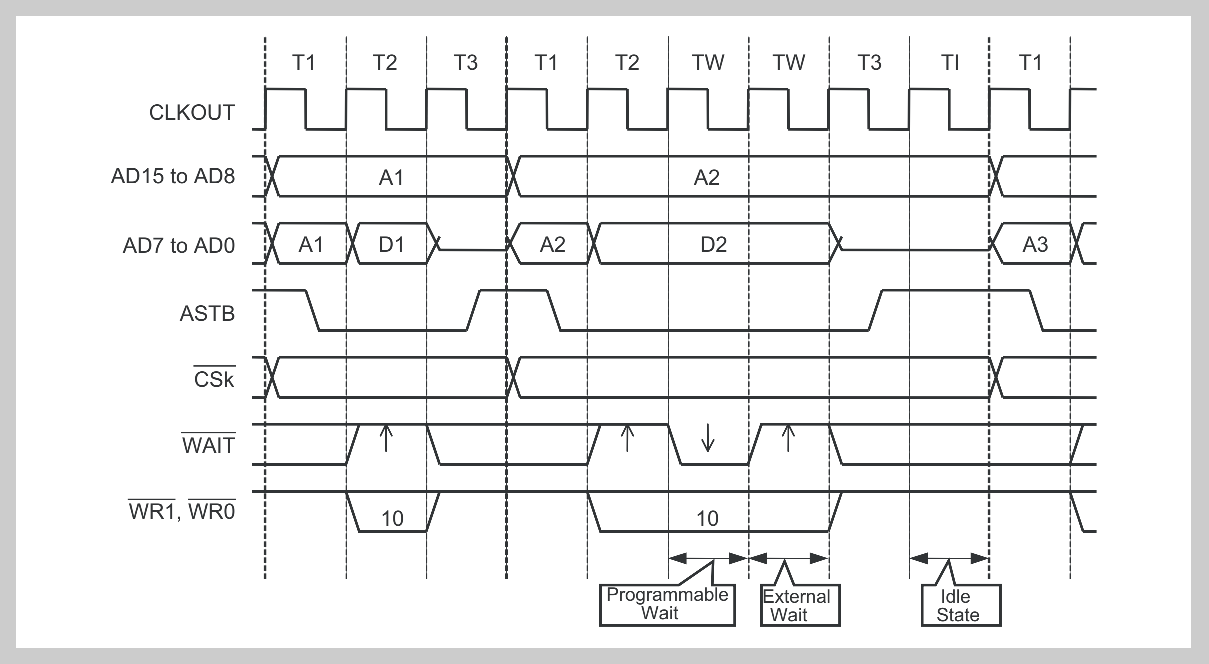


Figure 9-8 Timing: write data with external and programmable wait cycles and idle state insertion (bus size: 8-bit)

Register settings:

- BSC.BSk0 = 0_B (8 bit data bus size)
- AWC.AHWk = AWC.ASWk = 0 (no address setup/hold wait states inserted)
- DWC0.DWk[2:0] = 001_B (one programmable data wait state inserted)
- BCC.BCk1 = 1_B (one idle state inserted)

Note

1. The arrows indicate the sampling timing.
2. AD[7:0] holds the address for accessing the odd address byte
   AD[15:8] holds the address for accessing the even address byte
3. $\overline{\text{CSk}}$ with k = 0 to 3.

The data has to be stable at the rising edge of the $\overline{\text{WR}}$ signal. For details refer to the Electrical Target Specification.

(3) Write with address setup/hold wait (bus size: 16-bit)

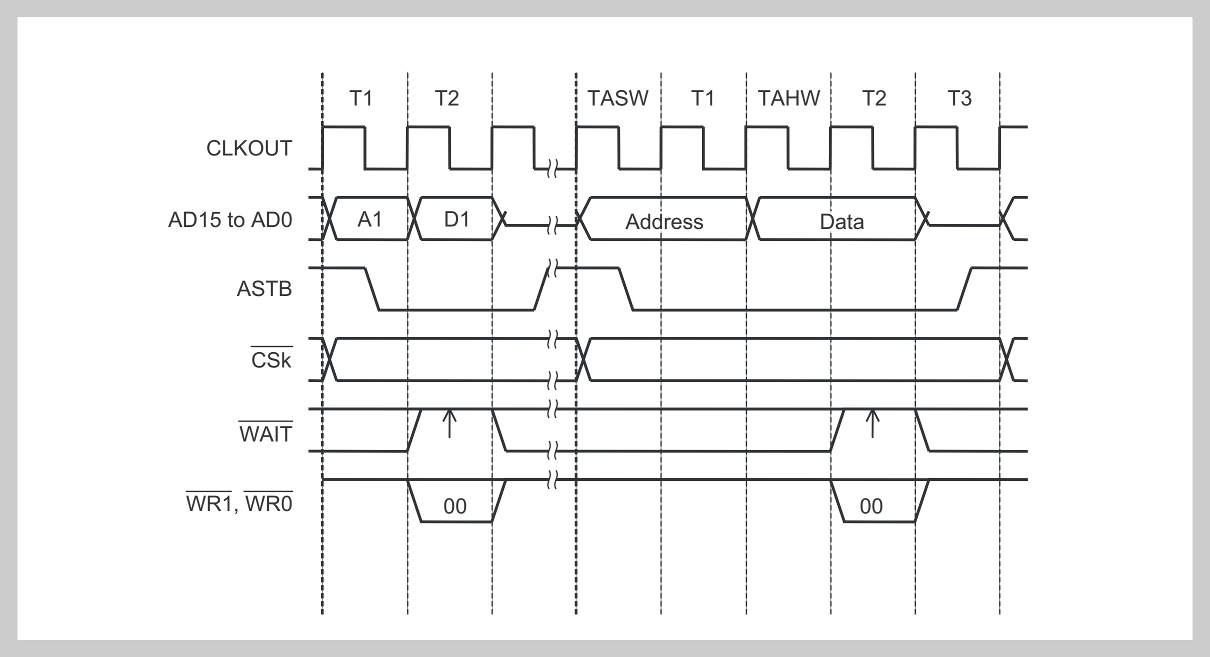


Figure 9-9 Timing: write data with address setup/hold wait (bus size: 16-bit)

Register settings:

- BSC.BSk0 = 1_B (16 bit data bus size)
- AWC.AHWk = AWC.ASWk = 1 (one address setup/hold wait state inserted)
- DWC0.DWk[2:0] = 000_B (no programmable data wait states inserted)
- BCC.BCk1 = 0_B (no idle states inserted)

Note

1. The arrows indicate the sampling timing.
2. AD[7:0] holds the address for accessing the odd address byte
   AD[15:8] holds the address for accessing the even address byte
3. $\overline{\text{CSk}}$ with k = 0 to 3.

The data has to be stable at the rising edge of the $\overline{\text{WR}}$ signal. For details refer to the Electrical Target Specification.

9.5.2 Reading from external devices

This section shows typical sequences of reading data from external devices.

(1) Read with wait cycles and idle state insertion (bus size: 16 bit)

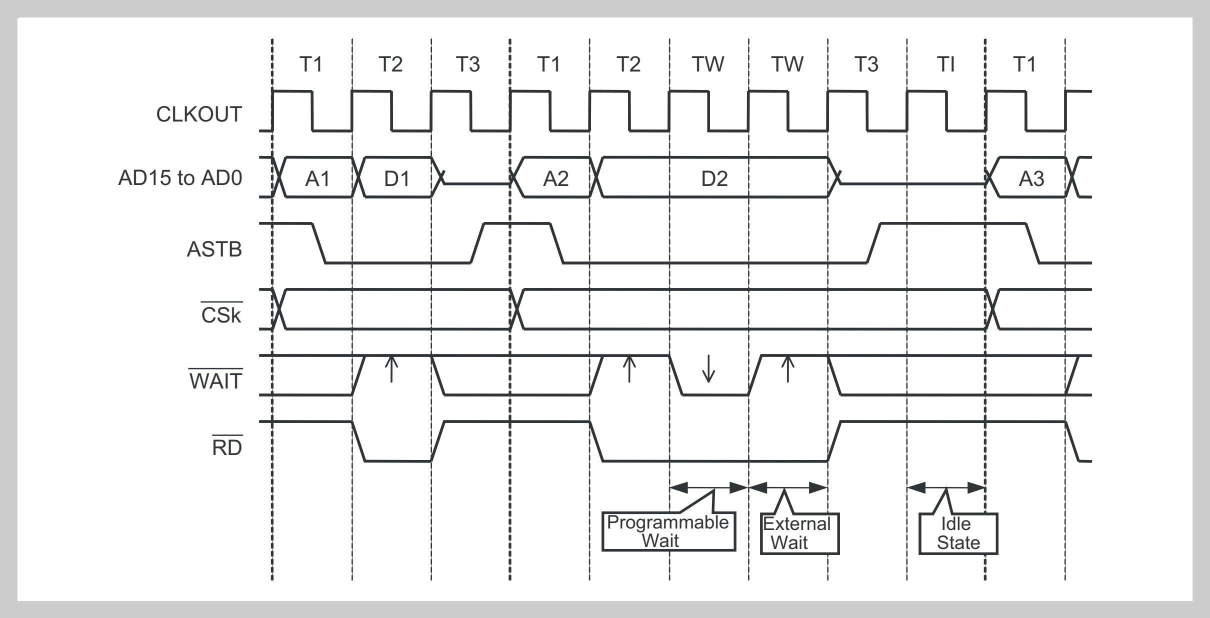


Figure 9-10 Timing: read data with external and programmable wait cycles and idle state insertion (bus size: 16 bit)

Register settings:

- BSC.BSk0 = 1_B (16 bit data bus size)
- AWC.AHWk = AWC.ASWk = 0 (no address setup/hold wait states inserted)
- DWC0.DWk[2:0] = 001_B (one programmable data wait state inserted)
- BCC.BCk1 = 1_B (one idle state inserted)

Note

1. The arrows indicate the sampling timing.
2. AD[7:0] holds the address for accessing the odd address byte
   AD[15:8] holds the address for accessing the even address byte
3. $\overline{\text{CSk}}$ with k = 0 to 3.

(2) Read with wait cycles and idle state insertion (bus size: 8 bit)

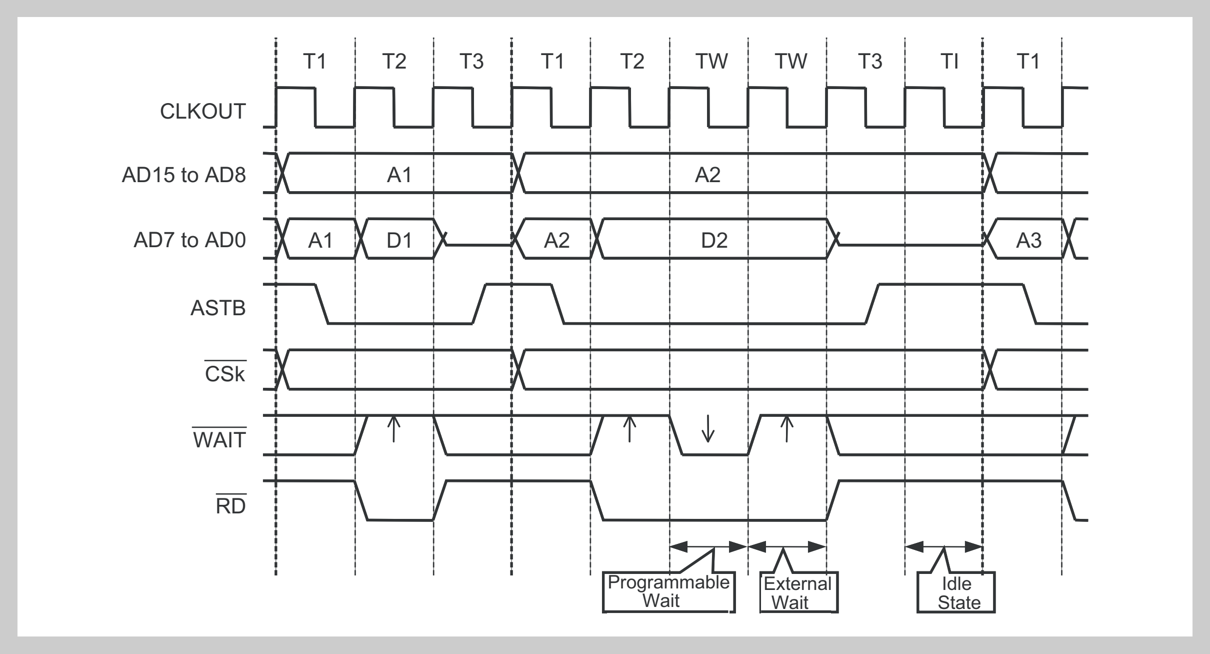


Figure 9-11 Timing: read data with external and programmable wait cycles and idle state insertion (bus size: 8 bit)

Register settings:

- BSC.BSk0 = 0_B (8 bit data bus size)
- AWC.AHWk = AWC.ASWk = 0 (no address setup/hold wait states inserted)
- DWC0.DWk[2:0] = 001_B (one programmable data wait state inserted)
- BCC.BCk1 = 1_B (one idle state inserted)

Note

1. The arrows indicate the sampling timing.
2. AD[7:0] holds the address for accessing the odd address byte
   AD[15:8] holds the address for accessing the even address byte
3. $\overline{\text{CSk}}$ with k = 0 to 3.

(3) Read with bus hold state and idle state insertion (bus size: 16 bits)

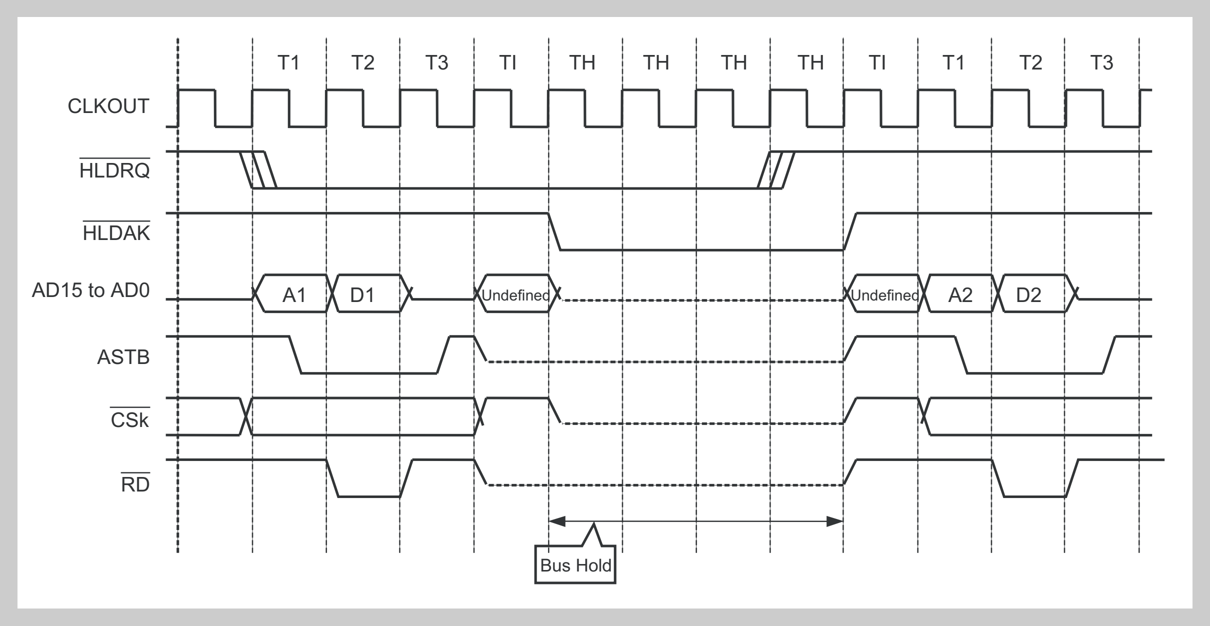


Figure 9-12 Timing: read data with bus hold state and idle state insertion (bus size: 16 bits)

Register settings:

- BSC.BSk0 = 1_B (16 bit data bus size)
- AWC.AHWk = AWC.ASWk = 0 (no address setup/hold wait states inserted)
- DWC0.DWk[2:0] = 001_B (one programmable data wait state inserted)
- BCC.BCk1 = 0_B (no idle state inserted, see also the Note below)

Note

1. The arrows indicate the sampling timing.
2. AD[7:0] holds the address for accessing the odd address byte
   AD[15:8] holds the address for accessing the even address byte
3. The idle state (TI) is independent of the setting of BCC.BCk1.
4. $\overline{\text{CSk}}$ with k = 0 to 3.

9.6 Data Access Order

9.6.1 Access to 8-bit data busses

This section shows how byte, half word and word accesses are performed for an 8-bit data bus. The endian format for all accesses is little endian.

(1) Byte access (8 bits)

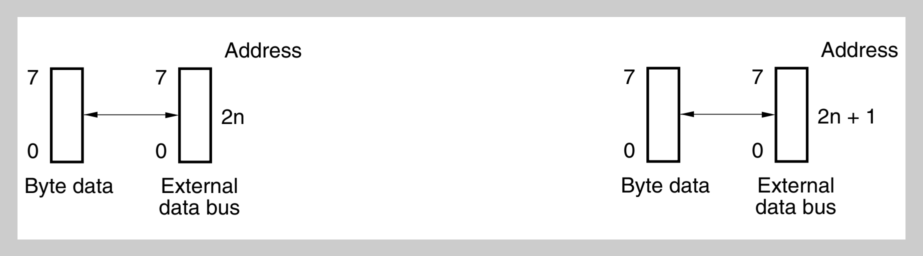


Figure 9-13 Left: Access to even address (2n)
Right: Access to odd address (2n + 1)

(2) Halfword access (16 bits)

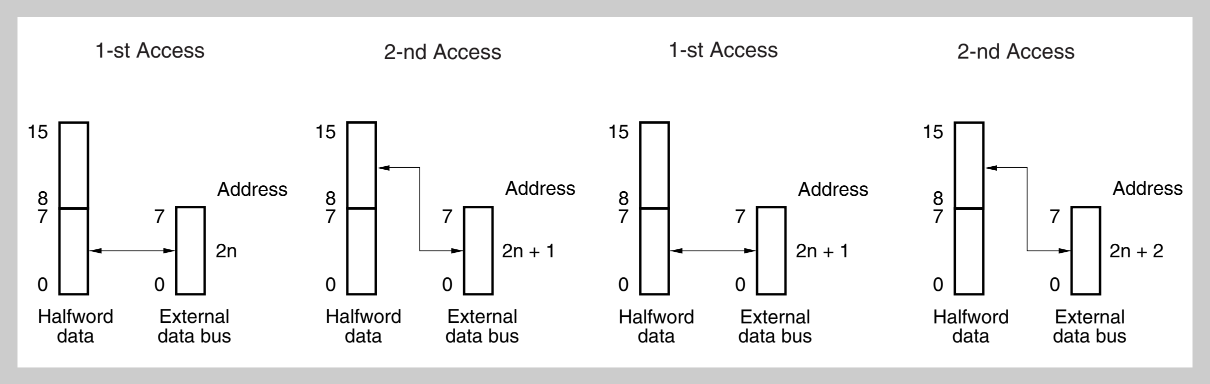


Figure 9-14 Left: Access to even address (2n)
Right: Access to odd address (2n + 1)

(3) Word access (32 bits)

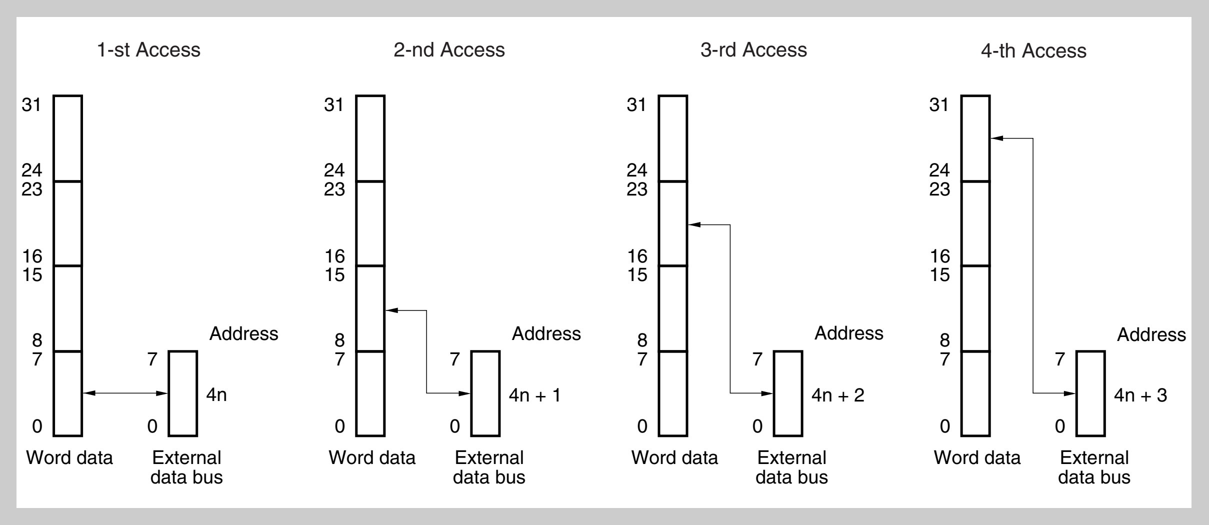


Figure 9-15 Access to address 4n

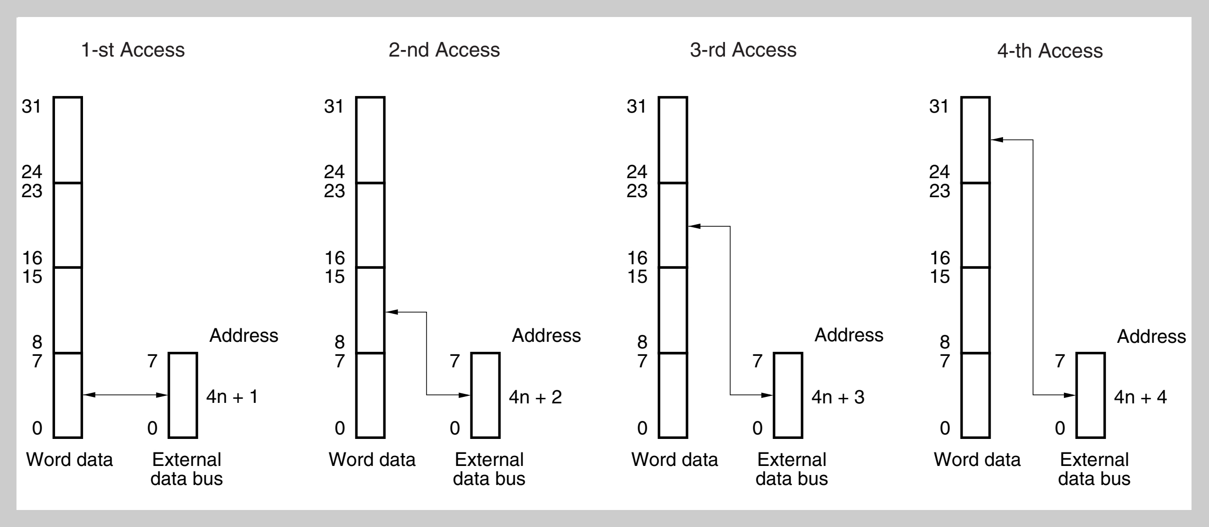


Figure 9-16 Access to address 4n + 1

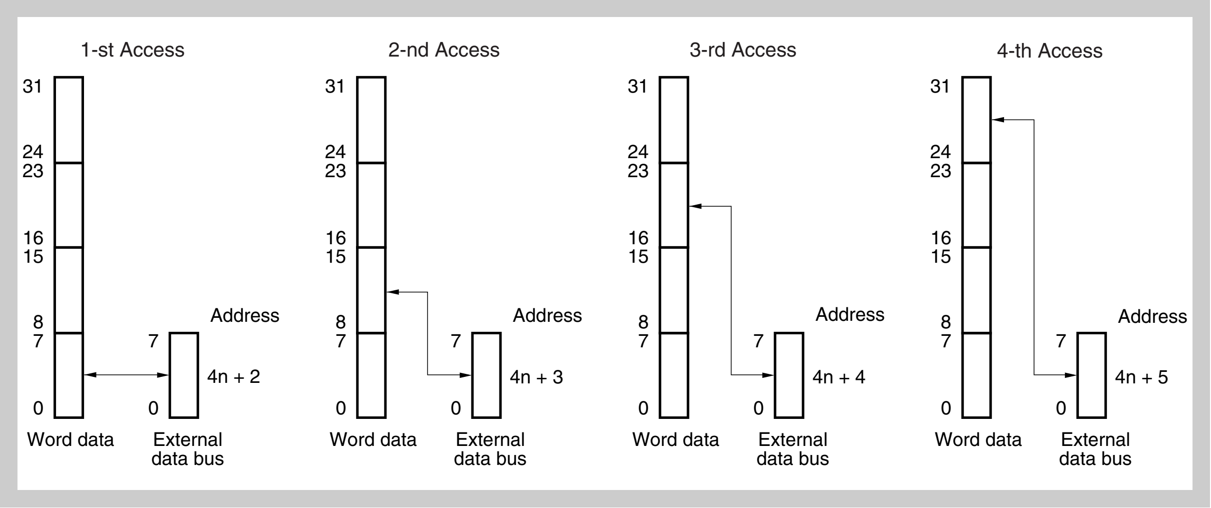


Figure 9-17 Access to address 4n + 2

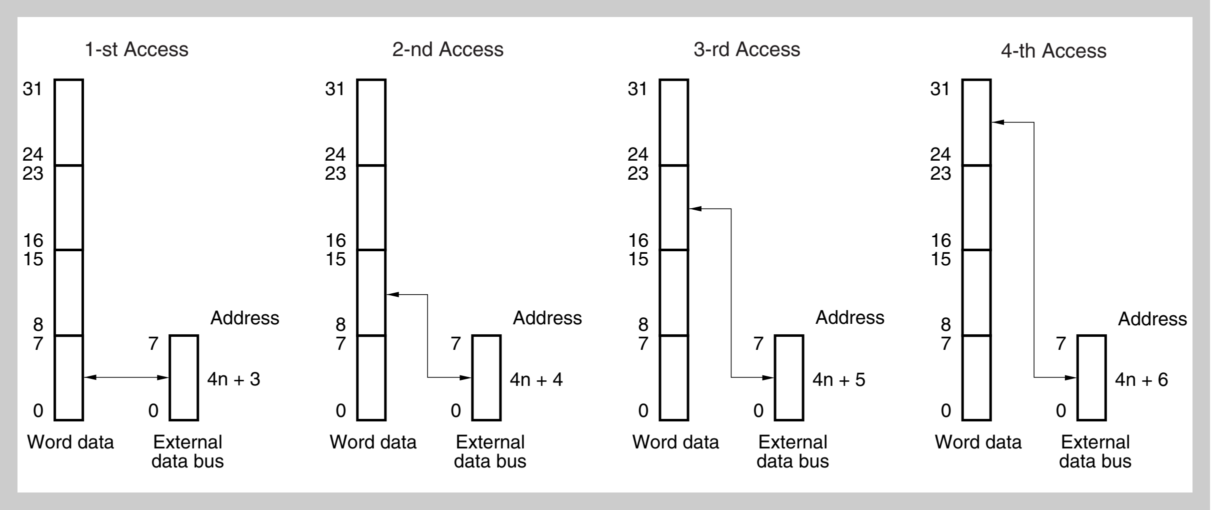


Figure 9-18 Access to address 4n + 3

9.6.2 Access to 16-bit data busses

This section shows how byte, half word and word accesses are performed for a 16 bit data bus. The endian format for all accesses is little endian.

Access all data in order starting from the lower order side.

(1) Byte access (8 bits)

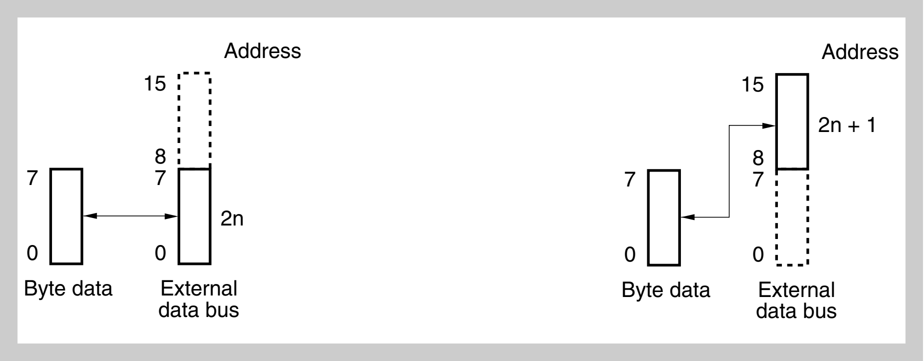


Figure 9-19 Left: Access to even address (2n)
Right: Access odd address (2n + 1)

(2) Halfword access (16 bits)

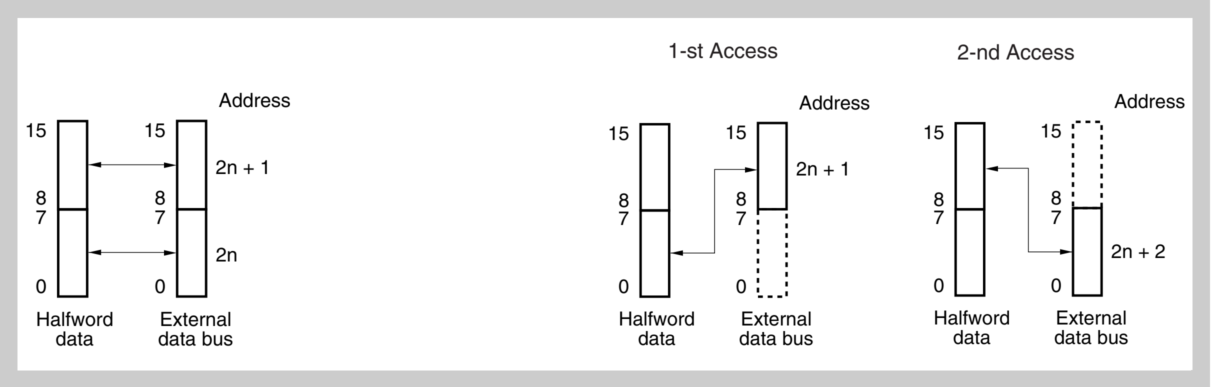


Figure 9-20 Left: Access to even address (2n)
Right: Access to odd address (2n + 1)

(3) Word access (32 bits)

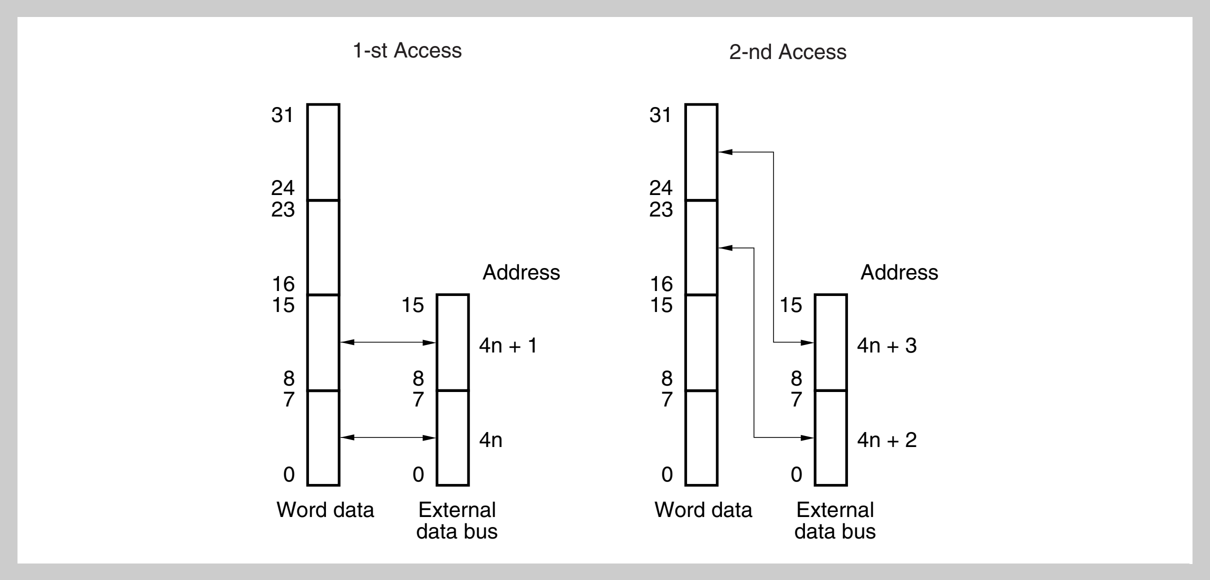


Figure 9-21 Access to address 4n

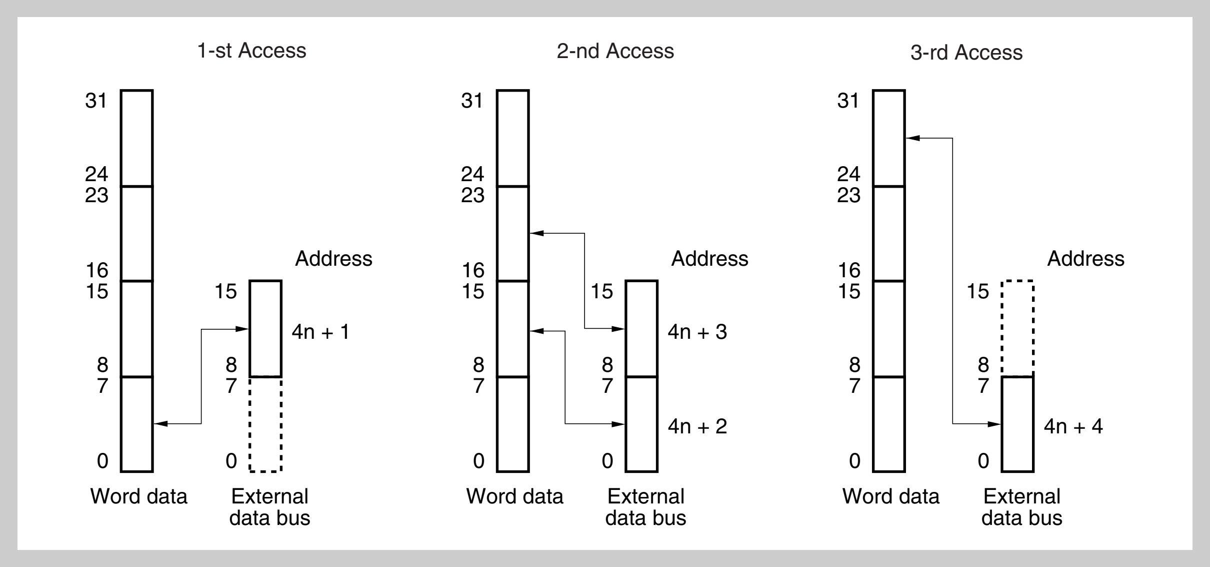


Figure 9-22 Access to address 4n + 1

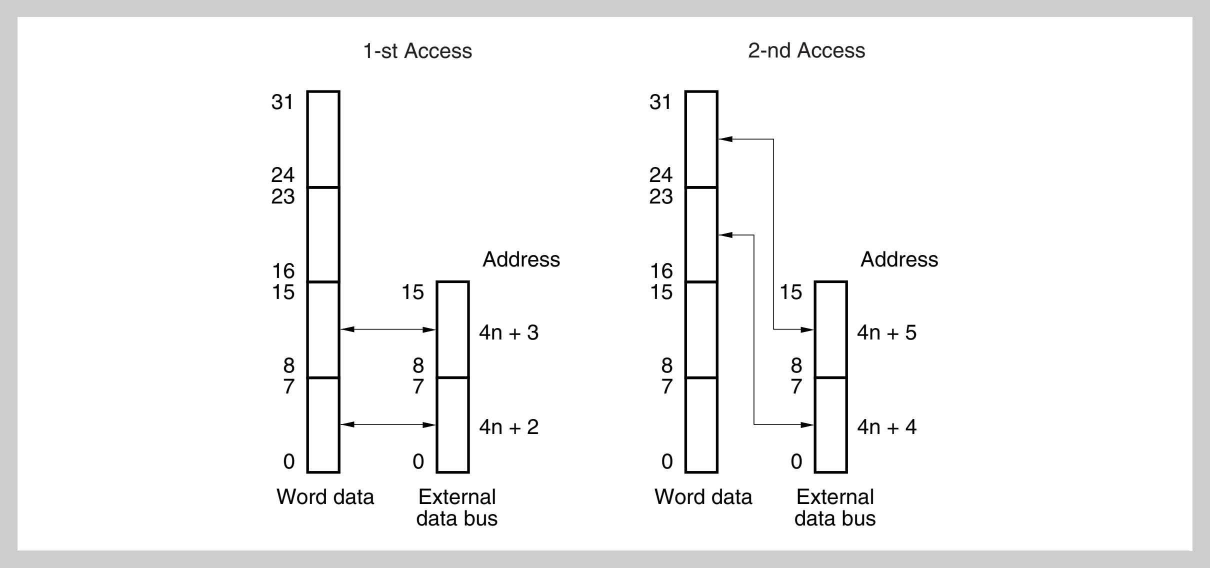


Figure 9-23 Access to address 4n + 2

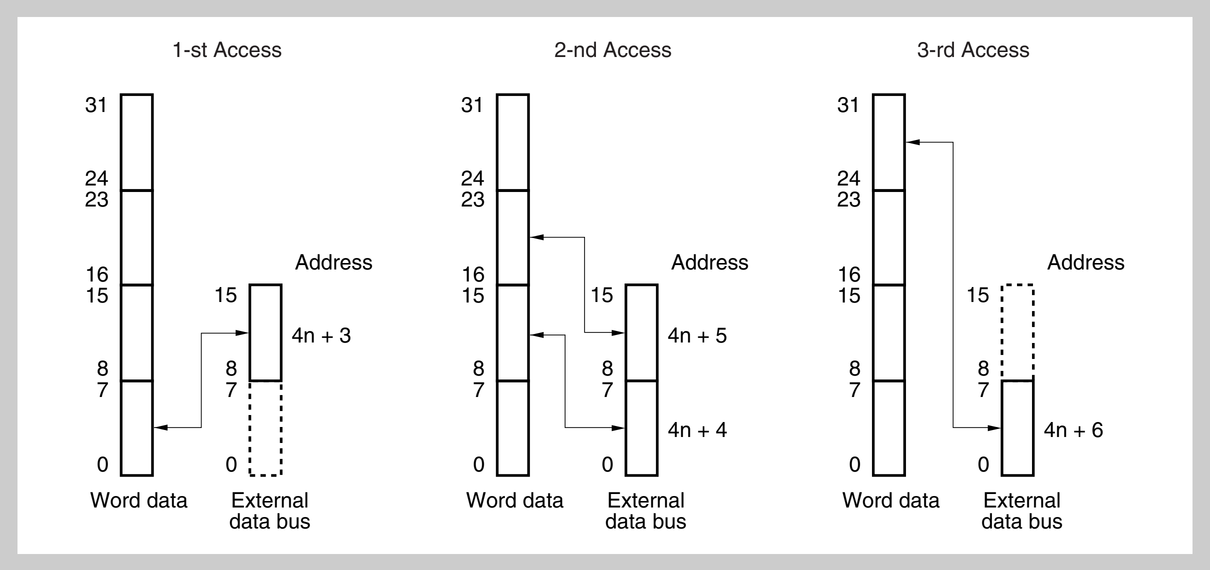


Figure 9-24 Access to address 4n + 3

Chapter 10 DMA Function (DMA Controller)

The microcontroller includes a direct memory access controller (DMAC) that executes and controls DMA transfer.

The DMAC controls data transfer between memory and I/O, between memories, or between I/Os based on DMA requests issued by the on-chip peripheral I/O (serial interface, timer/counter, and A/D Converter), interrupts from external input pins, or software triggers (memory refers to internal RAM, data flash, or external memory).

10.1 Features

- 4 independent DMA channels
- Transfer unit: 8/16 bits
- Maximum transfer count: 65536 (2^{16})
- Transfer type: Two-cycle transfer
- Transfer mode: Single transfer mode
- Transfer requests
- Request by interrupts from on-chip peripheral I/O (serial interface, timer/counter, A/D Converter) or interrupts from external input pin
- Requests by software trigger
- Transfer targets
  The following table specifies the available sources and destination for DMA transfers.

Table 10-1 Relationship between transfer targets

		Transfer destination				
		On-chip Peripheral I/O	External memory	Data flash	Internal RAM	Internal ROM
Source	On-chip peripheral I/O	√	√	×	√	×
	External memory	√	√	×	√	×
	Data flash	√	√	×	√	×
	Internal RAM	√	√	×	×	×
	Internal ROM	×	×	×	×	×

Note √: Transfer enabled, ×: Transfer disabled

10.2 Configuration

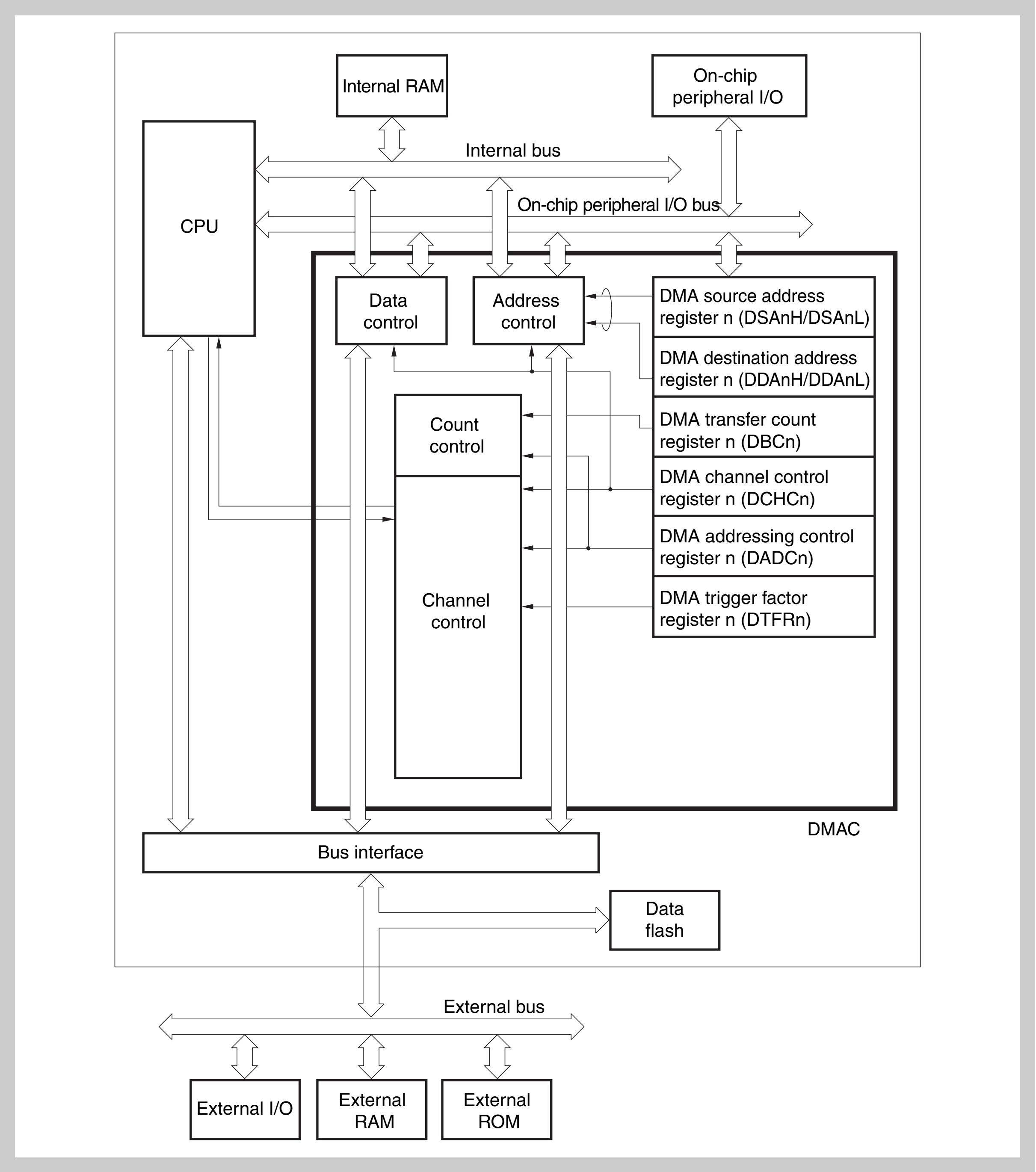

Internal RAM
On-chip peripheral I/O
Internal bus
On-chip peripheral I/O bus
CPU
Data control
Address control
DMA source address register n (DSAnH/DSAnL)
DMA destination address register n (DDAnH/DDAnL)
Count control
DMA transfer count register n (DBCn)
DMA channel control register n (DCHCn)
DMA addressing control register n (DADCn)
Channel control
DMA trigger factor register n (DTFRn)
DMAC
Bus interface
Data flash
External bus
External I/O
External RAM
External ROM

Figure 10-1 Block Diagram of DMA Controller

Note n = 0 to 3

10.3 Registers

(1) DSA0 to DSA3 - DMA source address registers 0 to 3

The DSA0 to DSA3 registers set the DMA source addresses (26 bits each) for DMA channel n (n = 0 to 3). These registers are divided into two 16-bit registers, DSAnH and DSAnL.

Access These registers can be read/written in 16-bit units.

Address

DSA0L:	FFFF $F080_H$	DSA0H:	FFFF $F082_H$
DSA1L:	FFFF $F088_H$	DSA1H:	FFFF $F08A_H$
DSA2L:	FFFF $F090_H$	DSA2H:	FFFF $F092_H$
DSA3L:	FFFF $F098_H$	DSA3H:	FFFF $F09A_H$

Initial Value undefined

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DSAnH	IR	0	0	0	0	0	SAn25	SAn24	SAn23	SAn22	SAn21	SAn20	SAn19	SAn18	SAn17	SAn16
	R/W															

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DSAnL	SAn15	SAn14	SAn13	SAn12	SAn11	SAn10	SAn9	SAn8	SAn7	SAn6	SAn5	SAn4	SAn3	SAn2	SAn1	SAn0
	R/W															

IR	Specification of DMA transfer source
0	External memory, on-chip peripheral or data flash
1	Internal RAM

SAn25 to SAn16	Set the address (A25 to A16) of the DMA transfer source (default value is undefined). During DMA transfer the next DMA transfer source address is held. When DMA transfer is completed the DMA address set first is held.

SAn15 to SAn0	Set the address (A15 to A0) of the DMA transfer source (default value is undefined). During DMA transfer the next DMA transfer source address is held. When DMA transfer is completed the DMA address set first is held.

Caution

1. Be sure to clear bits 14 to 10 of the DSAnH register to 0.
2. Set the DSAnH and DSAnL registers at the following timing when DMA transfer is disabled (DCHCn.Enn bit = 0).
   - Period from after reset to start of first DMA transfer
   - Period from after channel initialization by DCHCn.INITn bit to start of DMA transfer
   - Period from after completion of DMA transfer (DCHCn.TCn bit = 1) to start of the next DMA transfer
3. When the value of the DSAn register is read, two 16-bit registers, DSAnH and DSAnL, are read. If reading and updating conflict, the value being updated may be read (see *"Cautions" on page 395*).
4. Following reset, set the DSAnH, DSAnL, DDAnH, DDAnL, and DBCn registers before starting DMA transfer. If these registers are not set, the operation when DMA transfer is started is not guaranteed.

(2) DDA0 to DDA3 - DMA destination address registers 0 to 3

The DDA0 to DDA3 registers set the DMA destination address (26 bits each) for DMA channel n (n = 0 to 3). These registers are divided into two 16-bit registers, DDAnH and DDAnL.

Access These registers can be read/written in 16-bit units.

Address

DDAL0:	FFFF $F084_H$	DDAH0:	FFFF $F086_H$
DDAL1:	FFFF $F08C_H$	DDAH1:	FFFF $F08E_H$
DDAL2:	FFFF $F094_H$	DDAH2:	FFFF $F096_H$
DDAL3:	FFFF $F09C_H$	DDAH3:	FFFF $F09E_H$

Initial Value undefined

DDAnH

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IR	0	0	0	0	0	DAn25	DAn24	DAn23	DAn22	DAn21	DAn20	DAn19	DAn18	DAn17	DAn16

R/W

DDAnL

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAn15	DAn14	DAn13	DAn12	DAn11	DAn10	DAn9	DAn8	DAn7	DAn6	DAn5	DAn4	DAn3	DAn2	DAn1	DAn0

R/W

IR	Specification of DMA transfer source
0	External memory, on-chip peripheral or data flash
1	Internal RAM

DAn25 to DAn16	Set the address (A25 to A16) of the DMA transfer destination (default value is undefined). During DMA transfer the next DMA transfer destination address is held. When DMA transfer is completed the DMA address set first is held.

Dn15 to DAn0	Set the address (A15 to A0) of the DMA transfer destination (default value is undefined). During DMA transfer the next DMA transfer destination address is held. When DMA transfer is completed the DMA address set first is held.

Caution

1. Be sure to clear bits 14 to 10 of the DDAnH register to 0.
2. Set the DDAnH and DDAnL registers at the following timing when DMA transfer is disabled (DCHCn.Enn bit = 0).
   - Period from after reset to start of first DMA transfer
   - Period from after channel initialization by DCHCn.INITn bit to start of DMA transfer
   - Period from after completion of DMA transfer (DCHCn.TCn bit = 1) to start of the next DMA transfer
3. When the value of the DDAn register is read, two 16-bit registers, DDAnH and DDAnL, are read. If reading and updating conflict, a value being updated may be read (see *“Cautions” on page 395*).
4. Following reset, set the DSAnH, DSAnL, DDAnH, DDAnL, and DBCn registers before starting DMA transfer. If these registers are not set, the operation when DMA transfer is started is not guaranteed.

(3) DBC0 to DBC3 - DMA byte count registers 0 to 3

The DBC0 to DBC3 registers are 16-bit registers that set the byte transfer count for DMA channel n (n = 0 to 3). These registers hold the remaining transfer count during DMA transfer.

These registers are decremented by 1 per one transfer regardless of the transfer data unit (8/16 bits), and the transfer is terminated if a borrow occurs.

Access These registers can be read/written in 16-bit units.

Address DBC0: FFFF $F0C0_H$
DBC1: FFFF $F0C2_H$
DBC2: FFFF $F0C4_H$
DBC3: FFFF $F0C6_H$

Initial Value undefined

	15	14	13	12	11	10	9	8
DBCn	BCn15	BCn14	BCn13	BCn12	BCn11	BCn10	BCn9	BCn8
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

	7	6	5	4	3	2	1	0
	BCn7	BCn6	BCn5	BCn4	BCn3	BCn2	BCn1	BCn0
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

DBCn15 to DBCn0	Byte transfer count setting or remaining byte transfer count during DMA transfer
0000H	Byte transfer count 1 or remaining byte transfer count.
0001H	Byte transfer count 2 or remaining byte transfer count.
..	..
FFFFH	Byte transfer count 65536 (2^{16}) or remaining byte transfer count.
The number of tranfer data set first is held when DMA transfer is complete.	

Caution

1. Set the DBCn register at the following timing when DMA transfer is disabled (DCHCn.Enn bit = 0).
   - Period from after reset to start of first DMA transfer
   - Period from after channel initialization by DCHCn.INITn bit to start of DMA transfer
   - Period from after completion of DMA transfer (DCHCn.TCn bit = 1) to start of the next DMA transfer
2. Following reset, set the DSAnH, DSAnL, DDAnH, DDAnL, and DBCn registers before starting DMA transfer. If these registers are not set, the operation when DMA transfer is started is not guaranteed.

Remark If the DBCn register is read during DMA transfer after the terminal count is reached the value set immediately before the DMA transfer will be read out (0000H will not be read, even if DMA transfer has ended).

(4) DADC0 to DADC3 - DMA addressing control registers 0 to 3

These 16-bit registers are used to control the DMA transfer modes for DMA channel n (n= 0 to 3).

Access These registers can be read/written in 16-bit units.

Address DADC0: FFFF F0D0$_H$
DADC1: FFFF F0D2$_H$
DADC2: FFFF F0D4$_H$
DADC3: FFFF F0D6$_H$

Initial Value 0000$_H$

	15	14	13	12	11	10	9	8
DADCn	0	DSn0	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
	SADn1	SADn0	DADn1	DADn0	0	0	0	0
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 10-2 DADCn register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>14</td><td>DSn0</td><td>Sets the transfer data size for DMA transfer.
<table>
<tr><th>DSn0</th><th>Transfer data size</th></tr>
<tr><td>0</td><td>8 bits</td></tr>
<tr><td>1</td><td>16 bits</td></tr>
</table>
For the peripheral I/O and programmable peripheral I/O registers, ensure the transfer size matches the access size.</td></tr>
<tr><td>7, 6</td><td>SADn1,
SADn0</td><td>Sets the count direction of the source address for DMA channel n.
<table>
<tr><th>SADn1</th><th>SADn0</th><th>Count direction</th></tr>
<tr><td>0</td><td>0</td><td>Increment</td></tr>
<tr><td>0</td><td>1</td><td>Decrement</td></tr>
<tr><td>1</td><td>0</td><td>Fixed</td></tr>
<tr><td>1</td><td>1</td><td>Setting prohibited</td></tr>
</table></td></tr>
<tr><td>5, 4</td><td>DADn1,
DADn0</td><td>Sets the count direction of the destination address for DMA channel n.
<table>
<tr><th>DAD1</th><th>DAD0</th><th>Count direction</th></tr>
<tr><td>0</td><td>0</td><td>Increment</td></tr>
<tr><td>0</td><td>1</td><td>Decrement</td></tr>
<tr><td>1</td><td>0</td><td>Fixed</td></tr>
<tr><td>1</td><td>1</td><td>Setting prohibited</td></tr>
</table></td></tr>
</table>

Caution

1. Be sure to clear bits 15, 13 to 8, and 3 to 0 of the DADCn register to 0.
2. Set the DADCn register at the following timing when DMA transfer is disabled (DCHCn.Enn bit = 0).
   - Period from after reset to start of first DMA transfer
   - Period from after channel initialization by DCHCn.INITn bit to start of DMA transfer
   - Period from after completion of DMA transfer (DCHCn.TCn bit = 1) to start of the next DMA transfer
3. The DSn0 bit specifies the size of the transfer data, and does not control bus sizing. If 8-bit data (DSn0 bit = 0) is set, therefore, the lower data bus is not always used.
4. If the transfer data size is set to 16 bits (DSn0 bit = 1), transfer cannot be started from an odd address. Transfer is always started from an address with the first bit of the lower address aligned to 0.
5. If DMA transfer is executed on an on-chip peripheral I/O register (as the transfer source or destination), be sure to specify the same transfer size as the register size. For example, to execute DMA transfer on an 8-bit register, be sure to specify 8-bit transfer.

(5) DCHC0 to DCHC3 - DMA channel control registers 0 to 3

These 8-bit registers are used to control the DMA transfer operating mode for DMA channel n.

Access These registers can be read/written in 8-bit or 1-bit units. (However, bit 7 is read only and bits 2 and 1 are write only. If bits 2 and 1 are read, the read value is always 0.)

Address DCHC0: FFFF F0E0$_H$
DCHC1: FFFF F0E2$_H$
DCHC2: FFFF F0E4$_H$
DCHC3: FFFF F0E6$_H$

Initial Value 00$_H$

	7	6	5	4	3	2	1	0
DCHCn	TCn	0	0	0	0	INITn	STGn	ENn
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 10-3 DCHCn register contents

Bit position	Bit name	Function
7	TCn	The Terminal Count status bit TC indicates whether DMA transfer through DMA channel n has ended or not. It is read-only, and is set to 1 when DMA transfer ends and cleared (0) when it is read. 0: DMA transfer has not ended. 1: DMA transfer has ended.
2	INITn	If the INITn bit is set to 1 with DMA transfer disabeld (Enn bit = 0), the DMA transfer status can be initialized. When re-setting the DMA transfer status (re-setting the DDAnH, DDAnL, DSAnH, DSAnL, DBCn and DADCn registers) be sure to initialize the DMA channel. The TCn bit must be 0 before setting the INITn bit to 1. When intializing the DMA controller be sure to observe the procedure described in the sub-chapter *“Cautions” on page 395*.
1	STGn	This is a software startup trigger of the DMA transfer. If this bit is set to 1 in the DMA transfer enable state (TCn bit = 0, Enn = 1) the DMA transfer is started.
0	ENn	Specifies whether DMA transfer through DMA channel n is to be enabled or disabled. 0: DMA transfer disabled 1: DMA transfer enabled The DMA transfer is enabled when the Enn bit is set to 1. When the DMA transfer is completed (when a terminal count is generated) this bitis automatically cleared to 0. To abort a DMA transfer, clear the Enn bit to 0 by software. To resume, set the Enn bit to 1 again. When aborting or resuming the DMA transfer be sure to observe the procedure described in the sub-chapter *“Cautions” on page 395*

Caution

1. Be sure to clear bits 6 to 3 of the DCHCn register to 0.
2. When DMA transfer is completed (when a terminal count is generated), the Enn bit is cleared to 0 and then the TCn bit is set to 1. If the DCHCn register is read while its bits are being updated, a value indicating “transfer not completed and transfer is disabled” (TCn bit = 0 and Enn bit = 0) may be read.
3. When generating a DMA transfer request via software, confirm that the TCn bit is set (1) before clearing the TCn bit (0).
4. Initialization may not be executed if setting of the INITn bit and DMA transfer from other channels conflict.

(6) DTFR0 to DTFR3 - DMA trigger factor registers 0 to 3

The DTFR0 to DTFR3 registers are 8-bit registers that control the DMA transfer trigger factor via interrupt request signals from on-chip peripheral I/O.

The interrupt request signals set by these registers serve as DMA transfer start factors.

Access These registers can be read or written in 8-bit units. However, DFn bit can be read or written in 1-bit units.

Address DTFR0: FFFF F810$_H$
DTFR1: FFFF F812$_H$
DTFR2: FFFF F814$_H$
DTFR3: FFFF F816$_H$

Initial Value 00$_H$

	7	6	5	4	3	2	1	0
DTFRn	DFn	0	IFCn5	IFCn4	IFCn3	IFCn2	IFCn1	IFCn0
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 10-4 DTFRn register contents

Bit position	Bit name	Function
7	DFn	DMA transfer request flag 0: No DMA transfer request 1: DMA transfer request **Note:** **1.** Only 0 can be written to the DFn bit. **2.** Write 0 to this bit to clear a DMA transfer request if an interrupt that is specified as the cause of starting DMA transfer occurs while DMA transfer is disabled.
5 to 0	IFCn5 to IFCn0	DMA trigger factor selection Specifies, which interrupt is used as transfer trigger factor for DMA channel n. For more details refer to *Table 10-5.*

Table 10-5 DMA trigger factors (1/3)

IFCn5	IFCn4	IFCn3	IFCn2	IFCn1	IFCn0	Interrupt Source
0	0	0	0	0	0	DMA request by interrupt disabled
0	0	0	0	0	1	INTLVIL
0	0	0	0	1	0	INTP0
0	0	0	0	1	1	INTP1
0	0	0	1	0	0	INTP2
0	0	0	1	0	1	INTP3
0	0	0	1	1	0	INTP4
0	0	0	1	1	1	INTP5
0	0	1	0	0	0	INTP6
0	0	1	0	0	1	INTP7
0	0	1	0	1	0	INTTAB0OV
0	0	1	0	1	1	INTTAB0CC0
0	0	1	1	0	0	INTTAB0CC1

Table 10-5 DMA trigger factors (2/3)

IFCn5	IFCn4	IFCn3	IFCn2	IFCn1	IFCn0	Interrupt Source
0	0	1	1	0	1	INTTAB0CC2
0	0	1	1	1	0	INTTAB0CC3
0	0	1	1	1	1	INTTAA0OV
0	1	0	0	0	0	INTTAA0CC0
0	1	0	0	0	1	INTTAA0CC1
0	1	0	0	1	0	INTTAA1OV
0	1	0	0	1	1	INTTAA1CC0
0	1	0	1	0	0	INTTAA1CC1
0	1	0	1	0	1	INTTAA2OV
0	1	0	1	1	0	INTTAA2CC0
0	1	0	1	1	1	INTTAA2CC1
0	1	1	0	0	0	INTTAA3OV
0	1	1	0	0	1	INTTAA3CC0
0	1	1	0	1	0	INTTAA3CC1
0	1	1	0	1	1	INTTM0EQ0
0	1	1	1	0	0	INTCB0R
0	1	1	1	0	1	INTCB0T
0	1	1	1	1	0	INTCB1R
0	1	1	1	1	1	INTCB1T
1	0	0	0	0	0	INTUD0R
1	0	0	0	0	1	INTUD0T
1	0	0	0	1	0	INTUD1R
1	0	0	0	1	1	INTUD1T
1	0	0	1	0	0	INTAD
1	0	0	1	0	1	INTTAA4OV
1	0	0	1	1	0	INTTAA4CC0
1	0	0	1	1	1	INTTAA4CC1
1	0	1	0	0	0	INTIIC0
1	0	1	0	0	1	INTKR
1	0	1	0	1	0	INTTAB1OV[a]
1	0	1	0	1	1	INTTAB1CC0[a]
1	0	1	1	0	0	INTTAB1CC1[a]
1	0	1	1	0	1	INTTAB1CC2[a]
1	0	1	1	1	0	INTTAB1CC3[a]
1	0	1	1	1	1	INTUD2R[a]
1	1	0	0	0	0	INTUD2T[a]
1	1	0	0	0	1	INTLVIH
1	1	0	0	1	0	INTUD3R[b]
1	1	0	0	1	1	INTUD3T[b]
1	1	0	1	0	0	INTTAB2OV[c]
1	1	0	1	0	1	INTTAB2CC0[c]

Table 10-5 DMA trigger factors (3/3)

IFCn5	IFCn4	IFCn3	IFCn2	IFCn1	IFCn0	Interrupt Source
1	1	0	1	1	0	INTTAB2CC1[c]
1	1	0	1	1	1	INTTAB2CC2[c]
1	1	1	0	0	0	INTTAB2CC3[c]
1	1	1	0	0	1	INTCB2R[c]
1	1	1	0	1	0	INTCB2T[c]
1	1	1	0	1	1	INTUD4R[b]
1	1	1	1	0	0	INTUD4T[b]
1	1	1	1	0	1	INTUD5R[d]
1	1	1	1	1	0	INTUD5T[d]
1	1	1	1	1	1	INTAD1[e]

a) not available for V850ES/FE3, V850ES/FF3

b) not available for
– V850ES/FE3
– V850ES/FF3
– μPD70F3374, μPD70F3375 of V850ES/FG3
– μPD70F3378 of V850ES/FJ3

c) not available for V850ES/FE3, V850ES/FF3, V850ES/FG3

d) not available for
– V850ES/FE3
– V850ES/FF3
– V850ES/FG3
– μPD70F3378 of V850ES/FJ3

e) not available for V850ES/FE3, V850ES/FF3, V850ES/FG3, V850ES/FJ3

Caution

1. Set the IFCn5 to IFCn0 bits at the following timing when DMA transfer is disabled (DCHCn.Enn bit = 0).
   - Period from after reset to start of first DMA transfer
   - Period from after channel initialization by DCHCn.INITn bit to start of DMA transfer
   - Period from after completion of DMA transfer (DCHCn.TCn bit = 1) to start of the next DMA transfer
2. An interrupt request that is generated in the standby mode (IDEL1, IDLE2, STOP, or sub-IDLE mode) does not start the DMA transfer cycle (nor is the DFn bit set to 1).
3. If a DMA trigger factor is selected by the IFCn5 to IFCn0 bits, the DFn bit is set to 1 when an interrupt occurs from the selected on-chip peripheral I/O, regardless of whether the DMA transfer is enabled or disabled. If DMA is enabled in this status, DMA transfer is immediately started.
4. When changing the DTFRn register disable all DMA transfers in lower priority DMA channels

10.4 Transfer Targets

Table 10-6 shows the relationship between the transfer targets (√: Transfer enabled, ×: Transfer disabled).

Table 10-6 Relationship between transfer targets

		Transfer Destination				
		On-Chip peripheral I/O	External memory	Data flash	Internal RAM	Internal ROM
Source	On-chip peripheral I/O	√	√	×	√	×
	External memory	√	√	×	√	×
	Data Flash	√	√	×	√	×
	Internal RAM	√	√	×	×	×
	Internal ROM	×	×	×	×	×

Caution The operation is not guaranteed for combinations of transfer destination and source marked with “×” in *Table 10-6*

10.5 Transfer Modes

Single transfer is supported as the transfer mode.

In single transfer mode, the bus is released at each byte/halfword transfer. If there is a subsequent DMA transfer request, transfer is performed again once. This operation continues until a terminal count occurs.

When the DMAC has released the bus, if another higher priority DMA transfer request is issued, the higher priority DMA request always takes precedence.

If a new transfer request of the same channel and a transfer request of another channel with a lower priority are generated in a transfer cycle, DMA transfer of the channel with the lower priority is executed after the bus is released to the CPU (the new transfer request of the same channel is ignored in the transfer cycle).

10.6 Transfer Types

As a transfer type, the 2-cycle transfer is supported.

In two-cycle transfer, data transfer is performed in two cycles, a read cycle and a write cycle.

In the read cycle, the transfer source address is output and reading is performed from the source to the DMAC. In the write cycle, the transfer destination address is output and writing is performed from the DMAC to the destination.

An idle cycle of one clock is always inserted between a read cycle and a write cycle. If the data bus width differs between the transfer source and destination for DMA transfer of two cycles, the operation is performed as follows.

<16-bit data transfer>

<1> Transfer from 32-bit bus → 16-bit bus
A read cycle (the higher 16 bits are in a high-impedance state) is generated, followed by generation of a write cycle (16 bits).

<2> Transfer from 16-/32-bit bus to 8-bit bus
A 16-bit read cycle is generated once, and then an 8-bit write cycle is generated twice.

<3> Transfer from 8-bit bus to 16-/32-bit bus
An 8-bit read cycle is generated twice, and then a 16-bit write cycle is generated once.

<4> Transfer between 16-bit bus and 32-bit bus
A 16-bit read cycle is generated once, and then a 16-bit write cycle is generated once.

For DMA transfer executed to an on-chip peripheral I/O register (transfer source/destination), be sure to specify the same transfer size as the register size. For example, for DMA transfer to an 8-bit register, be sure to specify byte (8-bit) transfer.

Note The bus width of each transfer target (transfer source/destination) is as follows.

- On-chip peripheral I/O: 16-bit bus width
- Internal RAM: 32-bit bus width
- External memory: 8-bit or 16-bit bus width
- Internal data flash: 16-bit bus width

10.7 DMA Channel Priorities

The DMA channel priorities are fixed as follows
DMA channel 0 > DMA channel 1 > DMA channel 2 > DMA channel 3

The priorities are checked for every transfer cycle.

Caution If DMA transfer at two or more DMA channels are activated by the same factor, transfer via a DMA channel with a lower priority may be acknowledged prior to transfer via DMA channels with higher priorities.

10.8 Time Related to DMA Transfer

The time required to respond to a DMA request, and the minimum number of clocks required for DMA transfer are shown below.

Single transfer: DMA response time (<1>) + Transfer source memory access (<2>) + 1[Note 1] + Transfer destination memory access (<2>)

DMA Cycle		Minimum Number of Execution Clocks
<1> DMA request response time		4 clocks (MIN.) + Noise elimination time[Note 2]
<2> Memory access	External memory access	Depends on connected memory.
	Data flash	• for $f_{XX} \leq 24$ MHz: 4 clocks • for 24 MHz < $f_{XX} \leq 48$ MHz: 5 clocks • for 40 MHz < $f_{XX} \leq 48$ MHz: 6clocks
	Internal RAM access	2 clocks[Note 3]
	Peripheral I/O register access	3 clocks + Number of wait cycles specified by VSWC register[Note 4]

Note

1. One clock is always inserted between a read cycle and a write cycle in DMA transfer.
2. If an external interrupt (INTPn) is specified as the trigger to start DMA transfer, noise elimination time is added (n = 0 to 7).
3. Two clocks are required for a DMA cycle.
4. More wait cycles are necessary for accessing a specific peripheral I/O register (for details, see *“CPU System Functions” on page 155*).

10.9 DMA Transfer Start Factors

There are two types of DMA transfer start factors, as shown below.

(1) Request by software

If the STGn bit is set to 1 while the DCHCn.TCn bit = 1 and Enn bit = 1 (DMA transfer enabled), DMA transfer is started.

To request the next DMA transfer cycle immediately after that, confirm, by using the DBCn register, that the preceding DMA transfer cycle has been completed, and set the STGn bit to 1 again (n = 0 to 3).

TCn bit = 0, Enn bit = 1

↓

STGn bit = 1 … Starts the first DMA transfer.

↓

Confirm that the contents of the DBCn register have been updated.
STGn bit = 1 … Starts the second DMA transfer.

↓

:

↓

Generation of terminal count … Enn bit = 0, TCn bit = 1, and INTDMAn signal is generated.

(2) Request by on-chip peripheral I/O

If an interrupt request is generated from the on-chip peripheral I/O set by the DTFRn register when the DCHCn.TCn bit = 0 and Enn bit = 1 (DMA transfer enabled), DMA transfer is started.

Caution

1. Two trigger factors (software trigger and hardware trigger) cannot be used for one DMA channel. If two trigger factors are simultaneously generated for one DMA channel, only one of them is valid. The trigger factor that is valid cannot be identified.
2. A new transfer request that is generated after the preceding DMA transfer request was generated or in the preceding DMA transfer cycle is ignored (cleared).
3. The transfer request interval of the same DMA channel varies depending on the setting of bus wait in the DMA transfer cycle, the start status of the other channels, or the external bus hold request. In particular, as described in Caution 2, a new transfer request that is generated for the same channel before the DMA transfer cycle or during the DMA transfer cycle is ignored. Therefore, the transfer request intervals for the same DMA channel must be sufficiently separated by the system. When the software trigger is used, completion of the DMA transfer cycle that was generated before can be checked by updating the DBCn register.

10.10 DMA Abort Factors

DMA transfer is aborted if a bus hold occurs.

The same applies if transfer is executed between the internal memory/on-chip peripheral I/O and internal memory/on-chip peripheral I/O.

When the bus hold is cleared, DMA transfer is resumed.

10.11 End of DMA Transfer

When DMA transfer has been completed the number of times set to the DBCn register and when the DCHCn.Enn bit is cleared to 0 and TCn bit is set to 1, a DMA transfer end interrupt request signal (INTDMAn) is generated for the Interrupt Controller (INTC) (n = 0 to 3).

The microcontroller does not output a terminal count signal to an external device. Therefore, confirm completion of DMA transfer by using the DMA transfer end interrupt or polling the TCn bit.

10.12 Operation Timing

Figure 10-2 to *Figure 10-5* show DMA operation timing.

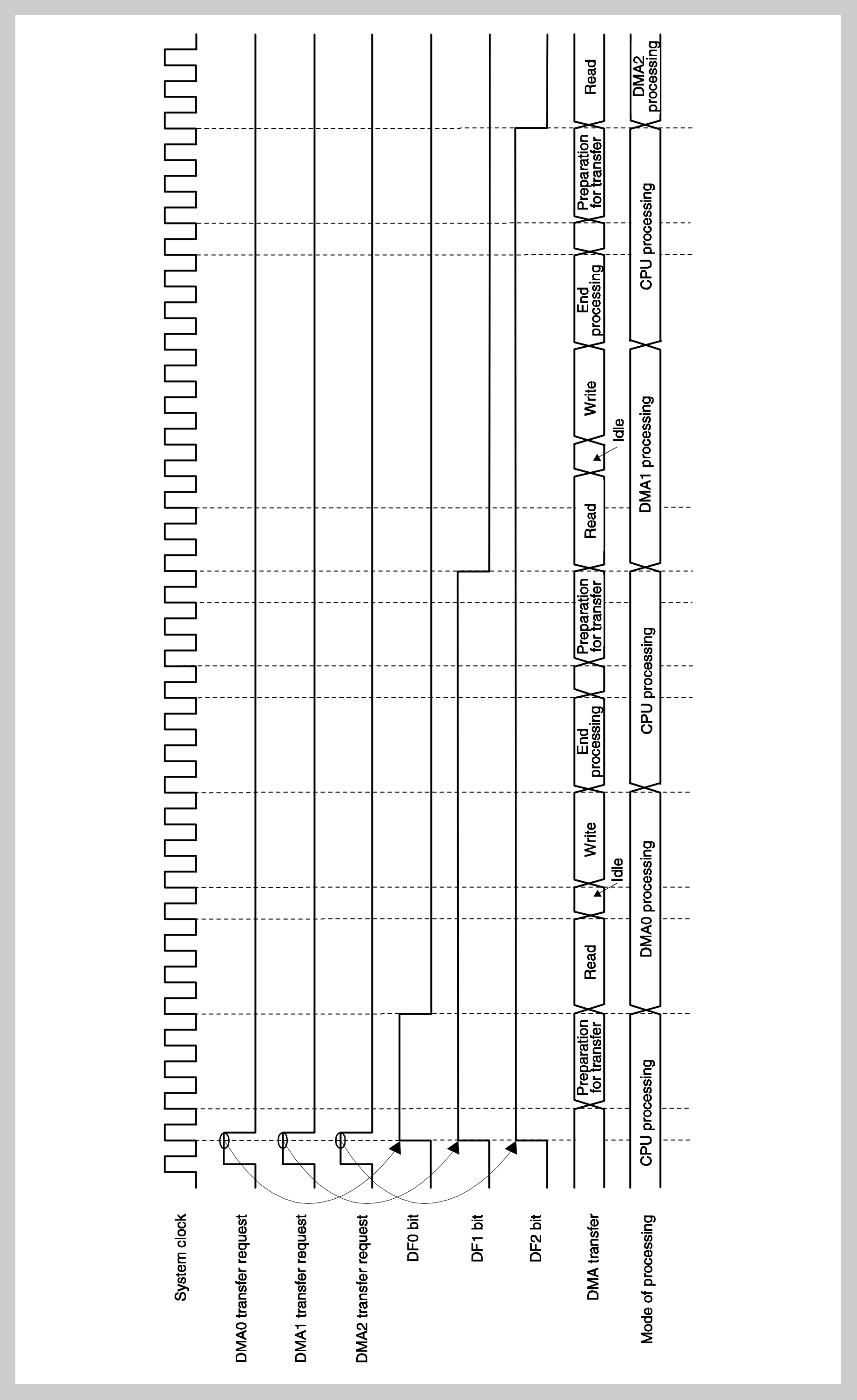


Figure 10-2 Priority of DMA (1)

Note **1.** Transfer in the order of DMA0 → DMA1 → DMA2

2. In the case of transfer between external memory spaces (multiplexed bus, no wait)

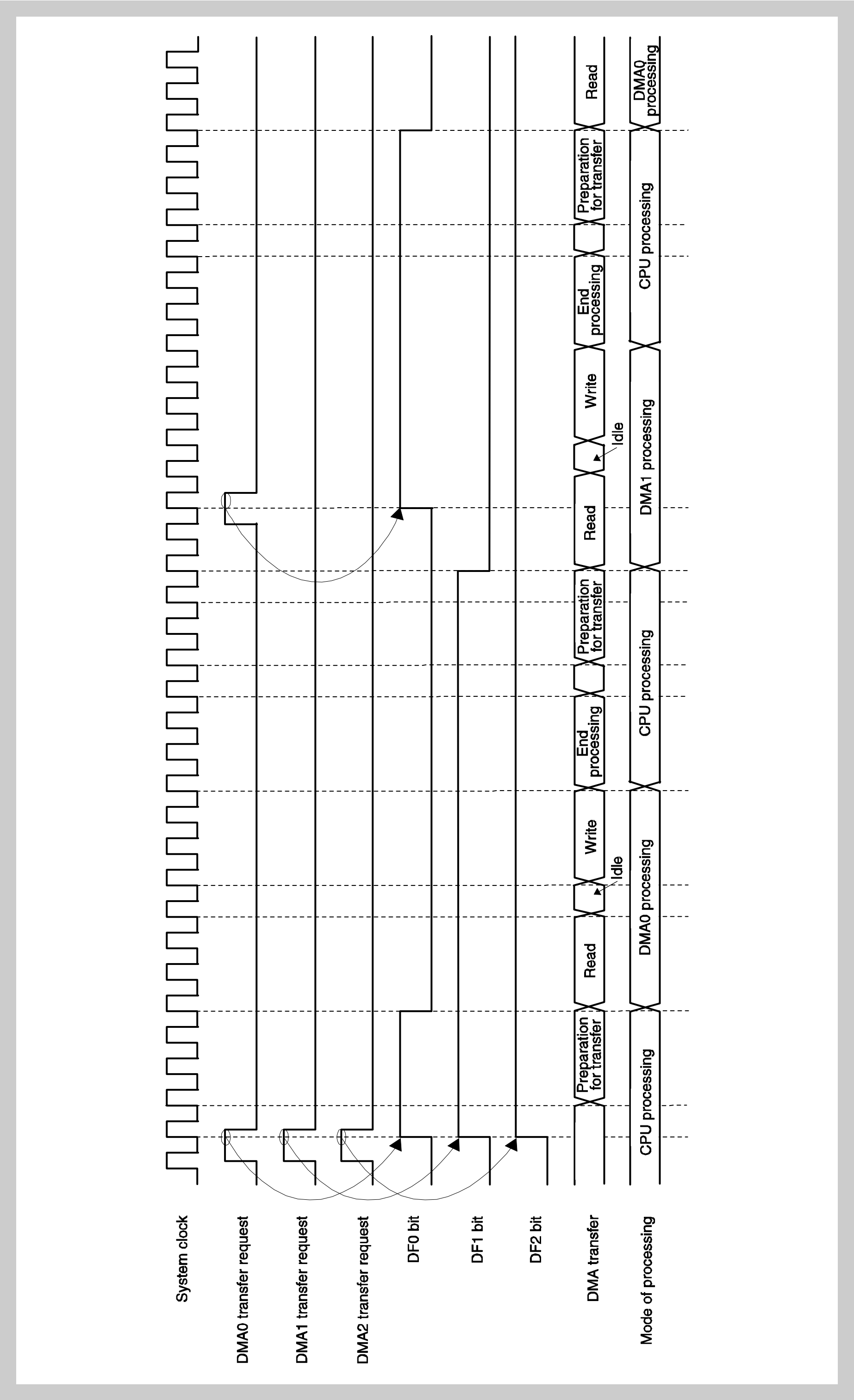


Figure 10-3 Priority of DMA (2)

Note 1. Transfer in the order of DMA0 → DMA1 → DMA0 (DMA2 is held pending.)

2. In the case of transfer between external memory spaces (multiplexed bus, no wait)

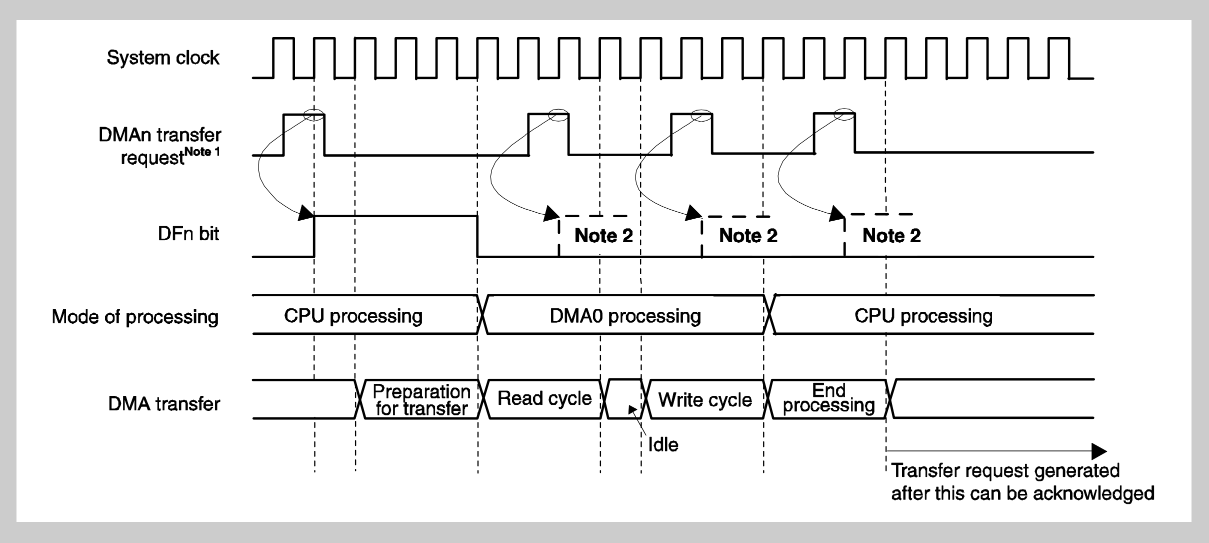


Figure 10-4 Period in Which DMA Transfer Request Is Ignored (1)

Note
1. Interrupt from on-chip peripheral I/O, or software trigger (STGn bit)
2. New DMA request of the same channel is ignored between when the first request is generated and the end processing is complete.
3. In the case of transfer between external memory spaces (multiplexed bus, no wait)

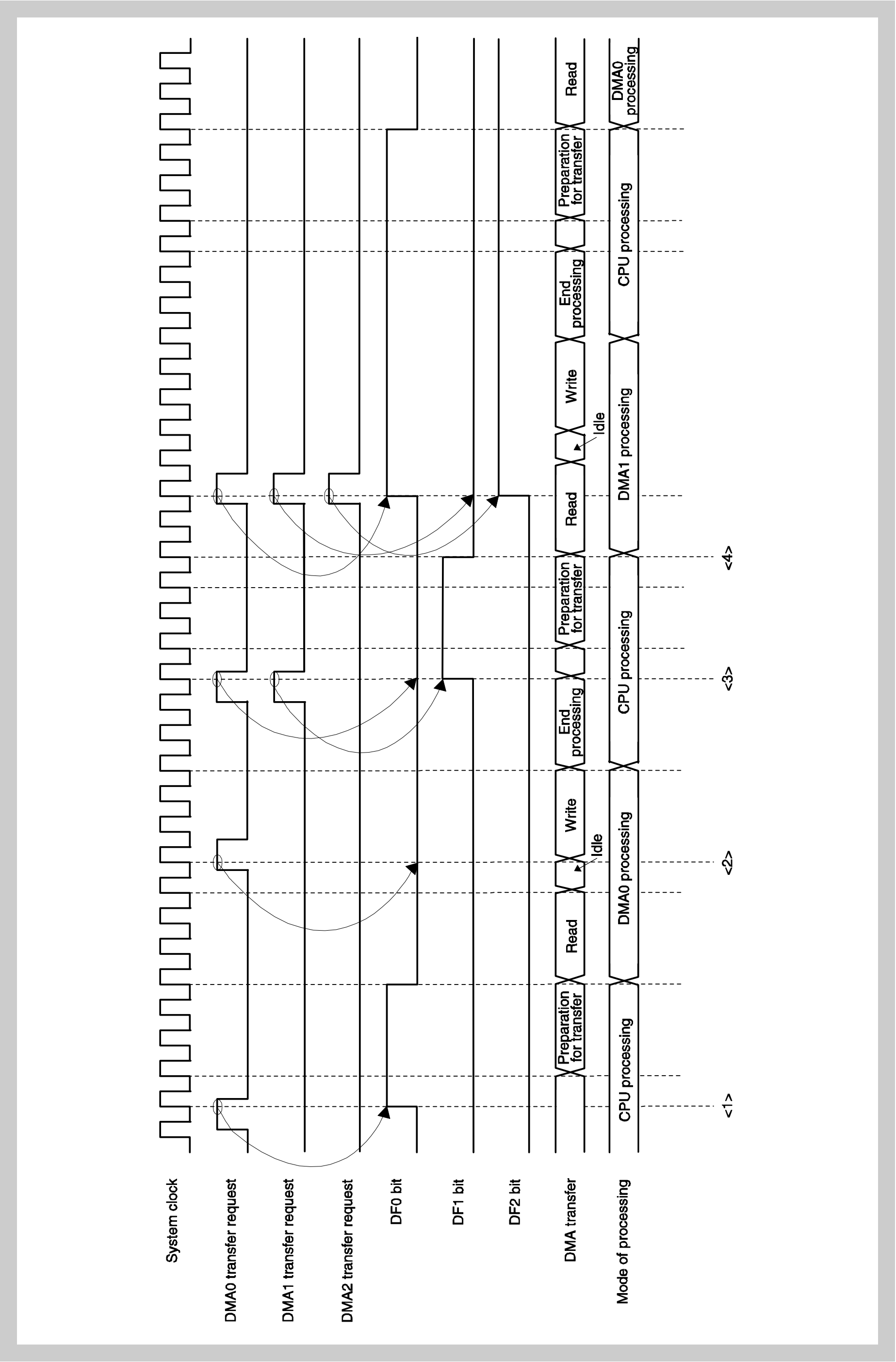


Figure 10-5 Period in Which DMA Transfer Request Is Ignored (2)

<1> DMA0 transfer request

<2> New DMA0 transfer request is generated during DMA0 transfer.
→ A DMA transfer request of the same channel is ignored during DMA transfer.

<3> New DMA0 transfer request is generated during DMA0 transfer.
→ A DMA transfer request of the same channel is ignored during DMA transfer.
→ DMA1 request is acknowledged.

<4> Requests for DMA0, DMA1, and DMA2 are generated at the same time.
→ DMA1 request is ignored (a DMA transfer request of the same channel during transfer is ignored).
→ DMA0 request is acknowledged according to priority. DMA2 request is held pending (transfer of DMA2 occurs next).

10.13 Cautions

(1) Caution for VSWC register

When using the DMAC, be sure to set an appropriate value, in accordance with the operating frequency, to the VSWC register.

When the default value (77H) of the VSWC register is used, or if an inappropriate value is set to the VSWC register, the operation is not correctly performed (for details of the VSWC register, see *“Bus and Memory Control (BCU, MEMC)” on page 339*.

(2) Caution for DMA transfer executed on internal RAM

When executing the following instructions located in the internal RAM, do not execute a DMA transfer that transfers data to/from the internal RAM (transfer source/destination), because the CPU may not operate correctly afterward.

- Bit manipulation instruction located in internal RAM (SET1, CLR1, or NOT1)
- Data access instruction to misaligned address located in internal RAM

Conversely, when executing a DMA transfer to transfer data to/from the internal RAM (transfer source/destination), do not execute the above two instructions.

(3) Caution for reading DCHCn.TCn bit (n = 0 to 3)

The TCn bit is cleared to 0 when it is read, but it is not automatically cleared even if it is read at a specific timing. To accurately clear the TCn bit, add the following processing.

(a) When waiting for completion of DMA transfer by polling TCn bit

Confirm that the TCn bit has been set to 1 (after TCn bit = 1 is read), and then read the TCn bit three more times.

(b) When reading TCn bit in interrupt servicing routine

Execute reading the TCn bit three times.

(4) DMA transfer initialization procedure (setting DCHCn.INITn bit to 1)

Even if the INITn bit is set to 1 when the channel executing DMA transfer is to be initialized, the channel may not be initialized. To accurately initialize the channel, execute either of the following two procedures.

(a) Temporarily stop transfer of all DMA channels

Initialize the channel executing DMA transfer using the procedure in <1> to <7> below.

Note, however, that TCn bit is cleared to 0 when step <5> is executed. Make sure that the other processing programs do not expect that the TCn bit is 1.

<1> Disable interrupts (DI).

<2> Read the DCHCn.Enn bit of DMA channels other than the one to be forcibly terminated, and transfer the value to a general-purpose register.

<3> Clear the Enn bit of the DMA channels used (including the channel to be forcibly terminated) to 0. To clear the Enn bit of the last DMA channel, execute the clear instruction twice. If the target of DMA transfer (transfer source/destination) is the internal RAM, execute the instruction three times.

Example:
Execute instructions in the following order if channels 0, 1, and 2 are used (if the target of transfer is not the internal RAM).

- Clear DCHC0.E00 bit to 0.
- Clear DCHC1.E11 bit to 0.
- Clear DCHC2.E22 bit to 0.
- Clear DCHC2.E22 bit to 0 again.

<4> Set the INITn bit of the channel to be forcibly terminated to 1.

<5> Read the TCn bit of each channel not to be forcibly terminated. If both the TCn bit and the Enn bit read in <2> are 1 (logical product (AND) is 1), clear the saved Enn bit to 0.

<6> After the operation in <5>, write the Enn bit value to the DCHCn register.

<7> Enable interrupts (EI).

Caution Be sure to execute step <5> above to prevent illegal setting of the Enn bit of the channels whose DMA transfer has been normally completed between <2> and <3>.

(b) Repeatedly execute setting INITn bit until transfer is forcibly terminated correctly

<1> Suppress a request from the DMA request source of the channel to be forcibly terminated (stop operation of the on-chip peripheral I/O).

<2> Check that the DMA transfer request of the channel to be forcibly terminated is not held pending, by using the DTFRn.DFn bit. If a DMA transfer request is held pending, wait until execution of the pending request is completed.

<3> When it has been confirmed that the DMA request of the channel to be forcibly terminated is not held pending, clear the Enn bit to 0.

<4> Again, clear the Enn bit of the channel to be forcibly terminated.
If the target of transfer for the channel to be forcibly terminated (transfer source/ destination) is the internal RAM, execute this operation once more.

<5> Copy the initial number of transfers of the channel to be forcibly terminated to a general-purpose register.

<6> Set the INITn bit of the channel to be forcibly terminated to 1.

<7> Read the value of the DBCn register of the channel to be forcibly terminated, and compare it with the value copied in <5>. If the two values do not match, repeat operations <6> and <7>.

Note 1. When the value of the DBCn register is read in <7>, the initial number of transfers is read if forced termination has been correctly completed. If not, the remaining number of transfers is read.

2. Note that method (b) may take a long time if the application frequently uses DMA transfer for a channel other than the DMA channel to be forcibly terminated.

(5) Procedure of temporarily stopping DMA transfer (clearing Enn bit)

Stop and resume the DMA transfer under execution using the following procedure.

<1> Suppress a transfer request from the DMA request source (stop the operation of the on-chip peripheral I/O).

<2> Check the DMA transfer request is not held pending, by using the DFn bit (check if the DFn bit = 0).
If a request is pending, wait until execution of the pending DMA transfer request is completed.

<3> If it has been confirmed that no DMA transfer request is held pending, clear the Enn bit to 0 (this operation stops DMA transfer).

<4> Set the Enn bit to 1 to resume DMA transfer.

<5> Resume the operation of the DMA request source that has been stopped (start the operation of the on-chip peripheral I/O).

(6) Memory boundary

The operation is not guaranteed if the address of the transfer source or destination exceeds the area of the DMA target (external memory, internal RAM, or on-chip peripheral I/O) during DMA transfer.

(7) Transferring misaligned data

DMA transfer of misaligned data with a 16-bit bus width is not supported.

If an odd address is specified as the transfer source or destination, the least significant bit of the address is forcibly assumed to be 0.

(8) Bus arbitration for CPU

Because the DMA Controller has a higher priority bus mastership than the CPU, a CPU access that takes place during DMA transfer is held pending until the DMA transfer cycle is completed and the bus is released to the CPU.

However, the CPU can access the external memory, on-chip peripheral I/O, and internal RAM to/from which DMA transfer is not being executed.

- The CPU can access the internal RAM when DMA transfer is being executed between the external memory and on-chip peripheral I/O.
- The CPU can access the internal RAM and on-chip peripheral I/O when DMA transfer is being executed between the external memory and external memory.

(9) Registers/bits that must not be rewritten during DMA operation

Set the following registers at the following timing when a DMA operation is not under execution.

[Registers]

- DSAnH, DSAnL, DDAnH, DDAnL, DBCn, and DADCn registers
- DTFRn.IFCn5 to DTFRn.IFCn0 bits

[Timing of setting]

- Period from after reset to start of the first DMA transfer
- Time after channel initialization to start of DMA transfer
- Period from after completion of DMA transfer (TCn bit = 1) to start of the next DMA transfer

(10) Be sure to set the following register bits to 0.

- Bits 14 to 10 of DSAnH register
- Bits 14 to 10 of DDAnH register
- Bits 15, 13 to 8, and 3 to 0 of DADCn register
- Bits 6 to 3 of DCHCn register

(11) DMA trigger factor

Do not start two or more DMA channels with the same trigger factor. If two or more channels are started with the same factor, a DMA channel with a lower priority may be acknowledged earlier than a DMA channel with a higher priority.

(12) Read values of DSAn and DDAn registers

Values in the middle of updating may be read from the DSAn and DDAn registers during DMA transfer (n = 0 to 3).

For example, if the DSAnH register and then the DSAnL register are read when the DMA transfer source address (DSAn register) is 0000FFFFH and the count direction is incremental (DADCn.SAD1 and DADCn.SAD0 bits = 00), the value of the DSAn register differs as follows, depending on whether DMA transfer is executed immediately after the DSAnH register is read.

(a) If DMA transfer does not occur while DSAn register is read

<1> Read value of DSAnH register: DSAnH = 0000H

<2> Read value of DSAnL register: DSAnL = FFFFH

(b) If DMA transfer occurs while DSAn register is read

<1> Read value of DSAnH register: DSAnH = 0000H

<2> Occurrence of DMA transfer

<3> Incrementing DSAn register: DSAn = 00100000H

<4> Read value of DSAnL register: DSAnL = 0000H

(13) Break command in On-Chip debug mode

When the break command is started in the on-chip debug (OCD) mode, peripheral devices that support DMA operation do not acknowledge the read operation triggered by DMA.

Chapter 11 16-Bit Timer/Event Counter AA

The V850ES/Fx3 microcontrollers have following instances of the 16-bit timer/ event counter AA:

TAA	V850ES/ FE3	V850ES/ FF3	V850ES/ FG3	V850ES/ FJ3	V850ES/ FK3
Instances	5				8
Names	TAA0 to TAA4				TAA0 to TAA7

Throughout this chapter, the individual instances of Timer AA are identified by "n", for example, TAAnCTL0 for the TAAn control register 0.

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

11.1 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 with Timers AA and Timers AB channels (refer to *"Timer AA/AB Synchronous Operation" on page 525*)
- 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

11.2 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 (TAA0+TAA1, TAA2+TAA3, TAA5+TAA6).

11.3 Configuration

TAA includes the following hardware.

Table 11-1 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	Selector control registers (SELCNT0, SELCNT1, SELCNT3, SELCNT5)
Timer output	TOAAn0, TOAAn1
Timer input	TIAAn0, TIAAn1
Control registers	• TAAn control registers 0, 1 (TAAnCTL0, TAAnCTL1) • TAAn I/O control registers 0 to 2 and 4 (TAAnIOC0 to TAAnIOC2, TAAnIOC4) • TAAn option registers 0, 1 (TAAnOPT0, TAAnOPT1)

Timer AA (TAA) pins are alternate function of port pins. For how to set the alternate function, refer to the description of the registers in *“Pin Functions” on page 32*.

The block diagram of the timer TAA is shown below. *Figure 11-2* to *Figure 11-5* show the block diagrams of the input circuits of the different timers TAAn.

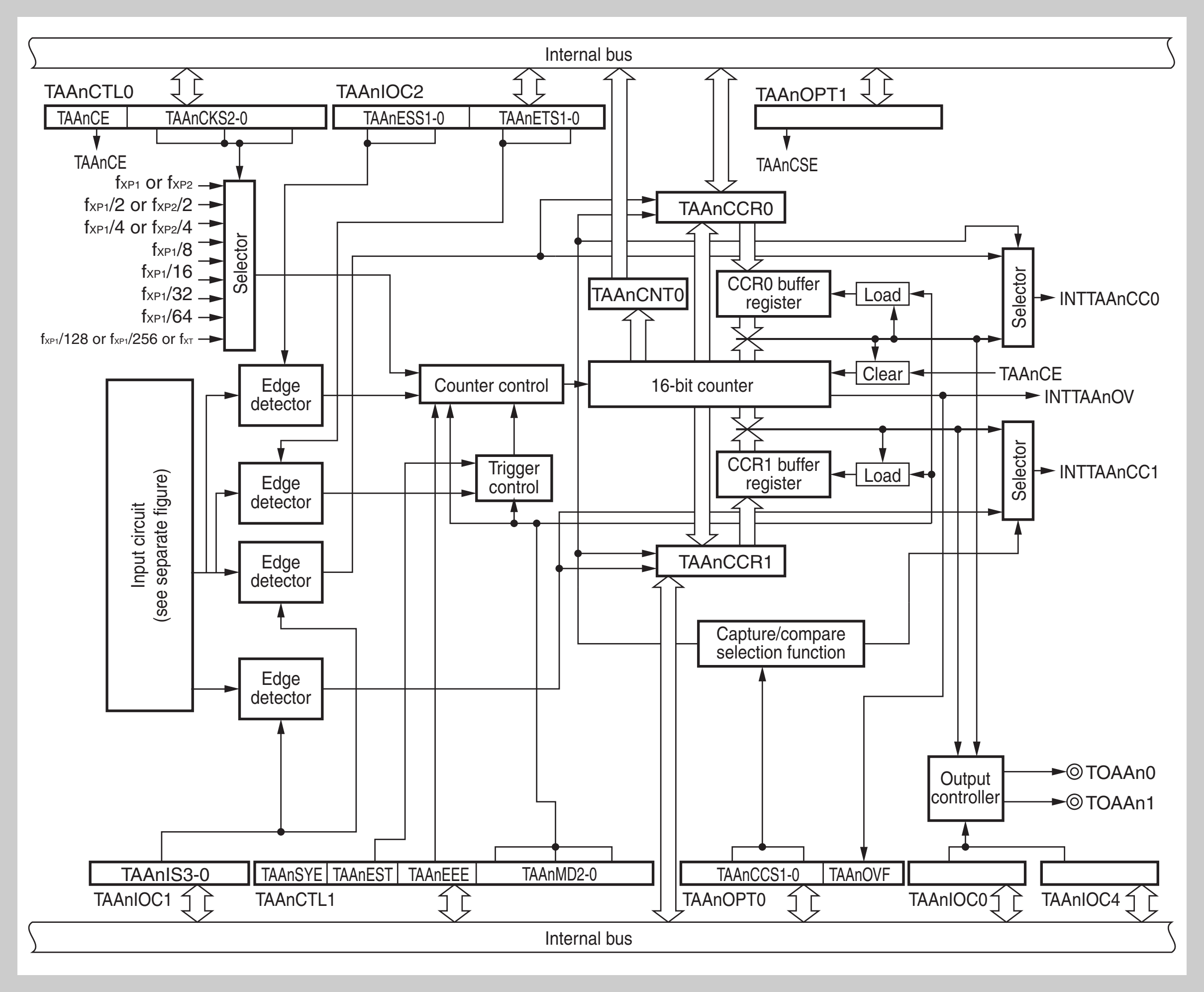


Figure 11-1 Block diagram of Timer AA

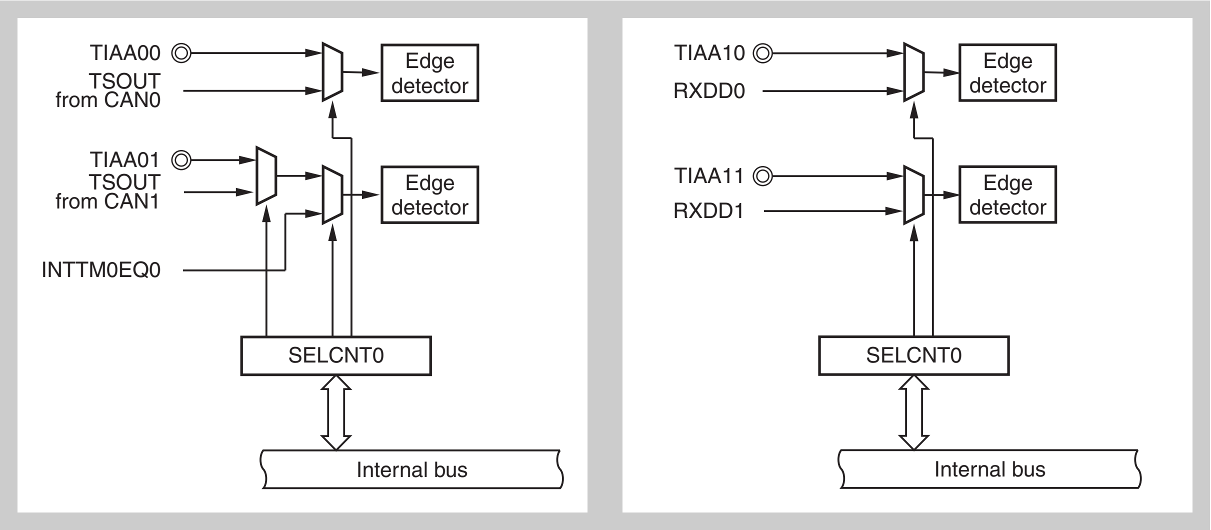


Figure 11-2 Input circuit of TAA0 (left) and TAA1 (right)

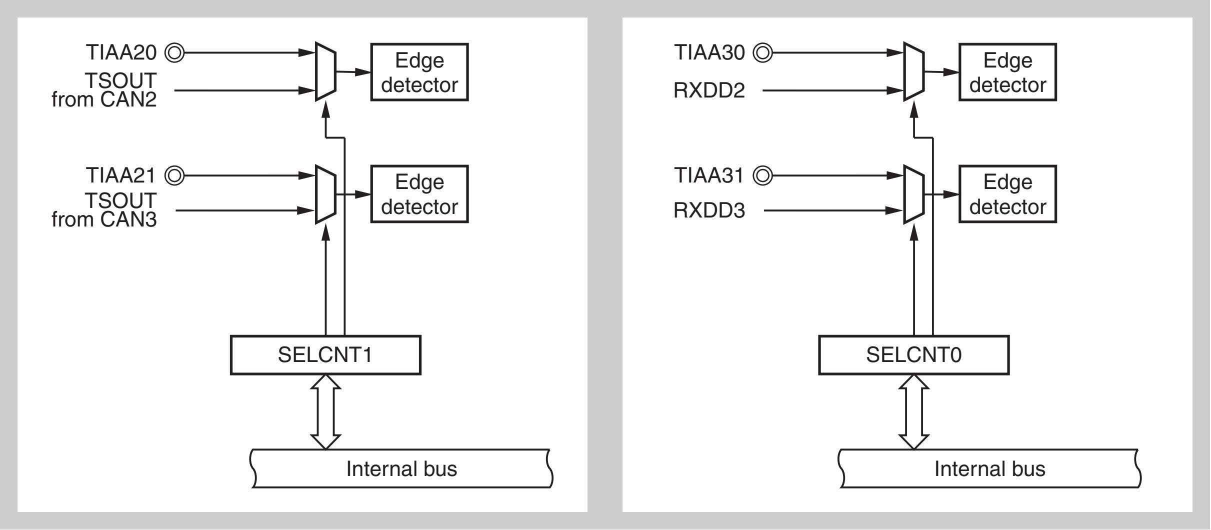


Figure 11-3 Input circuit of TAA2 (left) and TAA3 (right)

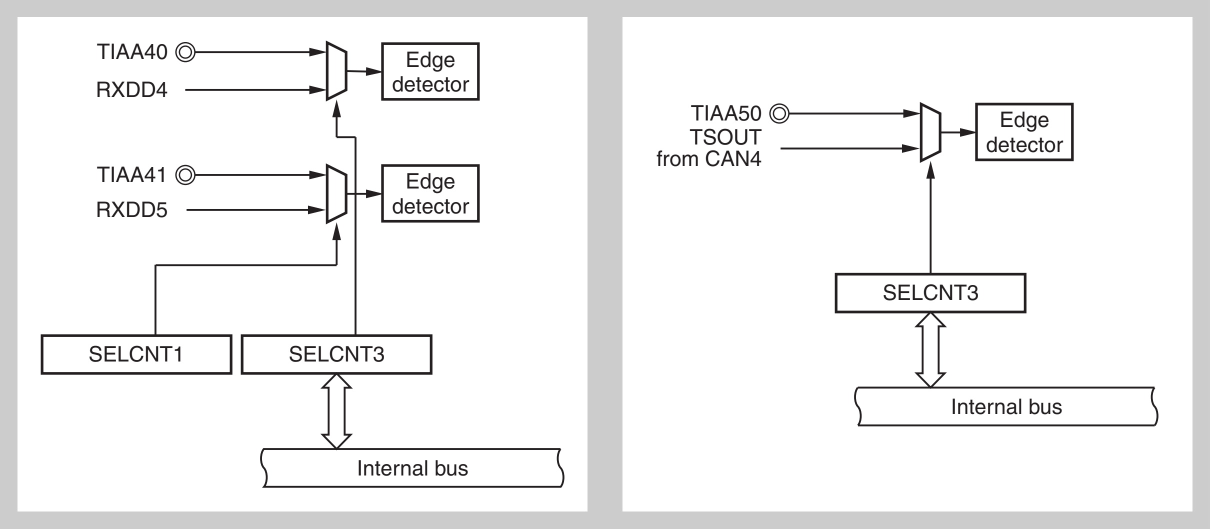


Figure 11-4 Input circuit of TAA4 (left) and TAA5 (right)

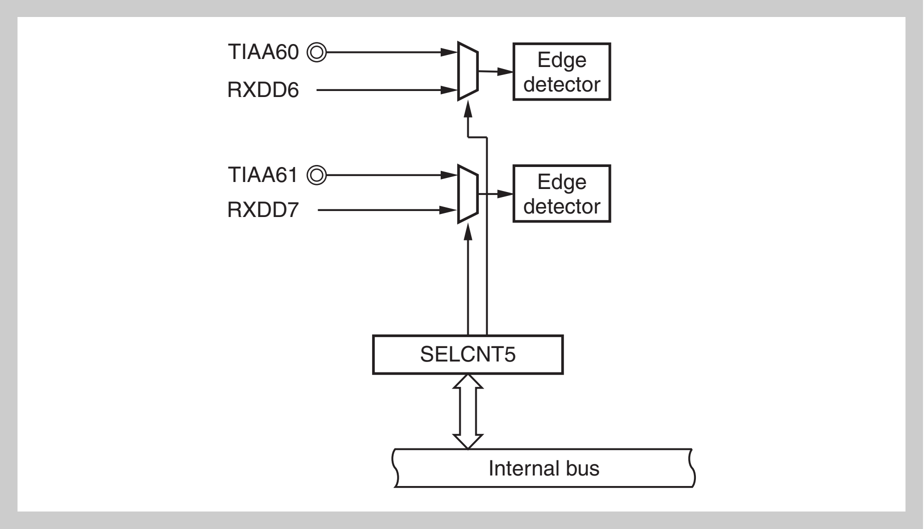


Figure 11-5 Input circuit of TAA6

(1) 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.

After a $\overline{\text{RESET}}$, TAAnCCR0 register default status is compare register.

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

Address

TAA0CCR0: FFFFF596_H	TAA1CCR0: FFFFF5A6_H
TAA2CCR0: FFFFF5B6_H	TAA3CCR0: FFFFF5C6_H
TAA4CCR0: FFFFF5D6_H	TAA5CCR0: FFFFF5E6_H
TAA6CCR0: FFFFF5F6_H	TAA7CCR0: FFFFF606_H

Initial Value 0000_H. This registers is cleared by any reset, or if the internal operation clock is disabled by TAAnCTL0.TAAnCE = 0.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TAAnCCR0	Capture/compare value 0															
	R/W															

Caution When external event counter mode is used, do not set TAAnCCR0 register to 0000_H.

Use as compare register When used as a compare register, TAAnCCR0 can be rewritten when TAAnCTL0.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)

Use as capture register When used as capture register, the count value is stored in TAAnCCR0 upon capture trigger (TIAAn0) input edge detection.

Note

1. The value of TAAnCCR0 register can be read/written when TAAnCTL0.TAAnCE = 1.
2. Access to the TAAnCCR0 register is prohibited when the main clock is stopped in the subclock mode.

(2) 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.

After $\overline{\text{RESET}}$, TAAnCCR1 register default status is compare register.

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

Address

TAA0CCR1: $FFFFF598_H$	TAA1CCR1: $FFFFF5A8_H$
TAA2CCR1: $FFFFF5B8_H$	TAA3CCR1: $FFFFF5C8_H$
TAA4CCR1: $FFFFF5D8_H$	TAA5CCR1: $FFFFF5E8_H$
TAA6CCR1: $FFFFF5F8_H$	TAA7CCR1: $FFFFF608_H$

Initial Value 0000_H. This register is cleared by any reset, or if the internal operation clock is disabled by TAAnCTL0.TAAnCE = 0.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TAAnCCR1	Capture/compare value 1															
	R/W															

Caution When external event counter mode is used, do not set TAAnCCR1 register to 0000_H.

Use as compare register When used as a compare register TAAnCCR1 can be rewritten when TAAnCTL0.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)

Use as capture register When used as a capture register the count value is stored in TAAnCCR1 upon capture trigger (TIAAn1) input edge detection.

Note

1. The value of TAAnCCR1 register can be read/written when TTAAnCTL0.TAAnCE = 1.
2. Access to the TAAnCCR1 register is prohibited when the main clock is stopped in the subclock mode.

(3) TAAnCNT - TAA counter read buffer register

TAAnCNT register is a read buffer register that can read 16-bit counter values.

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

Address

TAA0CNT:	FFFFF59A$_H$	TAA1CNT:	FFFFF5AA$_H$
TAA2CNT:	FFFFF5BA$_H$	TAA3CNT:	FFFFF5CA$_H$
TAA4CNT:	FFFFF5DA$_H$	TAA5CNT:	FFFFF5EA$_H$
TAA6CNT:	FFFFF5FA$_H$	TAA7CNT:	FFFFF60A$_H$

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
TAAnCNT	Counter value															
	R															

Note 0000$_H$ is read from this register, when TAAnCTL0.TAAnCE = 0. The current counter value is read when TAAnCE bit = 1.

11.4 Input Selection Registers

These registers are used to select the inputs to timers.

Note
1. In this section, only the bits that refer to Timer AA input selections are described. For further information concerning the other bits please refer to *"Clock Generator" on page 179*.
2. Enable the related peripheral function only after setting/changing the SELCNTn registers.

(1) SELCNT0 - Selector control register 0

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

Address FFFF F308$_H$.

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

- V850ES/FE3
- V850ES/FF3

	7	6	5	4	3	2	1	0
SELCNT0	ISEL07	0	0	ISEL04	ISEL03	ISEL02	0	ISEL00
	R/W	R	R	R/W	R/W	R/W	R	R/W

- µPD70F3374, µPD70F3375 of V850ES/FG3
- µPD70F3378 of V850ES/FJ3

	7	6	5	4	3	2	1	0
SELCNT0	ISEL07	0	ISEL05	ISEL04	ISEL03	ISEL02	ISEL01	ISEL00
	R/W	R	R/W	R/W	R/W	R/W	R/W	R/W

- µPD70F3376A, µPD70F3377A of V850ES/FG3
- µPD70F3379, µPD70F3380, µPD70F3381, µPD70F3382 of V850ES/FJ3
- V850ES/FK3

	7	6	5	4	3	2	1	0
SELCNT0	ISEL07	ISEL06	ISEL05	ISEL04	ISEL03	ISEL02	ISEL01	ISEL00
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Note "R" bits marked with "0" must not be changed from their default value "0".

Table 11-2 SELCNT0 register contents (1/2)

Bit position	Bit name	Function
7	ISEL07	Refer to Clock Generator: *"SELCNT0 - Selector control register 0" on page 209*
6	ISEL06	Selection of TIAA31: 0: TIAA31 pin 1: RXDD3 pin
5	ISEL05	Selection of TIAA30: 0: TIAA30 pin 1: RXDD2 pin
4	ISEL04	Selection of TIAA11: 0: TIAA11 pin 1: RXDD1 pin

Table 11-2 SELCNT0 register contents (2/2)

Bit position	Bit name	Function
3	ISEL03	Selection of TIAA10: 0: TIAA10 pin 1: RXDD0 pin
2, 1	ISEL0[2:1]	Selection of TIAA01: 00_B:TIAA01 pin 01_B:TSOUT signal from CAN1 $1\times_B$:INTTM0EQ0 signal from TMM **Note:** If the INTTM0EQ0 interrupt signal is used for the TIAA01 input signal, use it in the following range. TMM operation clock period ≥ TAA operation clock period × 4
0	ISEL00	Selection of TIAA00: 0: TIAA00 pin 1: TSO UT signal from CAN0

(2) SELCNT1 - Selector control register 1

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

Address FFFF $F30A_H$.

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

- µPD70F3381, µPD70F3382 of V850ES/FJ3
- V850ES/FK3

	7	6	5	4	3	2	1	0
SELCNT1	0	0	ISEL15	ISEL14	ISEL13	ISEL12	ISEL11	ISEL10
	R	R	R/W	R/W	R/W	R/W	R/W	R/W

Note “R” bits marked with “0” must not be changed from their default value “0”.

Table 11-3 SELCNT1 register contents

Bit position	Bit name	Function
5 to 3	ISEL1[5:3]	Refer to Clock Generator: *“SELCNT1 - Selector control register 1” on page 210*
2	ISEL12	Selection of TIAA41: 0: TIAA41 pin 1: RXDD5 pin
1	ISEL11	Selection of TIAA21: 0: TIAA21 pin 1: TSOUT signal from CAN3
0	ISEL10	Selection of TIAA20: 0: TIAA20 pin 1: TSOUT signal from CAN2

(3) SELCNT3 - Selector control register 3

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

Address FFFF F30E$_H$.

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

- µPD70F3374, µPD70F3375 of V850ES/FG3
- µPD70F3378 of V850ES/FJ3

	7	6	5	4	3	2	1	0
SELCNT3	0	0	0	0	0	ISEL32	ISEL31	0
	R	R	R	R	R	R/W	R/W	R

- µPD70F3379, µPD70F3380, µPD70F3381, µPD70F3382 of V850ES/FJ3
- V850ES/FK3

	7	6	5	4	3	2	1	0
SELCNT3	0	0	0	ISEL34	ISEL33	ISEL32	ISEL31	ISEL30
	R	R	R	R/W	R/W	R/W	R/W	R/W

- V850ES/FK3

	7	6	5	4	3	2	1	0
SELCNT3	0	ISEL36	0	ISEL34	ISEL33	ISEL32	ISEL31	ISEL30
	R/W	R/W	R	R/W	R/W	R/W	R/W	R

Note “R” bits marked with “0” must not be changed from their default value “0”.

Table 11-4 SELCNT3 register contents

Bit position	Bit name	Function
6	ISEL36	Selection of TIAA50: 0: TIAA50 pin 1: TSOUT signal from CAN4
4 to 1	ISEL3[4:1]	Refer to Clock Generator: *“SELCNT3 - Selector control register 3” on page 212*
0	ISEL30	Selection of TIAA40: 0: TIAA40 pin 1: RXDD4 pin

(4) SELCNT5 - Selector control register 5

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

Address FFFF F3FA$_H$.

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

- V850ES/FK3

	7	6	5	4	3	2	1	0
SELCNT5	ISEL57	ISEL56	ISEL55	ISEL54	ISEL53	ISEL52	ISEL51	ISEL50
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 11-5 SELCNT5 register contents

Bit position	Bit name	Function
7	ISEL57	Selection of TIAA61: 0: TIAA61 pin 1: RXDD7 pin
6	ISEL56	Selection of TIAA60: 0: TIAA60 pin 1: RXDD6 pin
5 to 0	ISEL5[5:0]	Refer to Clock Generator: *“SELCNT5 - Selector control register 5” on page 214*

11.5 Control Registers

(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.

Address

TAA0CTL0: FFFFF590$_H$	TAA1CTL0: FFFFF5A0$_H$
TAA2CTL0: FFFFF5B0$_H$	TAA3CTL0: FFFFF5C0$_H$
TAA4CTL0: FFFFF5D0$_H$	TAA5CTL0: FFFFF5E0$_H$
TAA6CTL0: FFFFF5F0$_H$	TAA6CTL0: FFFFF600$_H$

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

	7	6	5	4	3	2	1	0
TAAnCTL0	TAAnCE	0	0	0	0	TAAnCKS2	TAAnCKS1	TAAnCKS0
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Caution The TAAnCTL0 register is prohibited from writing during operation (TAAnCE=1). However, the TAAnCE bit can be rewritten.

Table 11-6 TAAnCTL0 register contents (1/2)

Bit position	Bit name	Function
7	TAAnCE	Controls the timer TAAn operation. 0: Internal operating clock operation disabled (TAAn is asynchronously reseted) 1: Internal operating clock operation enabled 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, i.e. TAAnOPT1.TAAnCSE = 1 (n = 1, 3, 5).

Table 11-6 TAAnCTL0 register contents (2/2)

Bit position	Bit name	Function
2 to 0	TAAnCKS [2:0]	Selects the count clock of timer TAAn.

SELCNT2.ISEL2[4:0] or SELCNT5.ISEL5[2:0][a]	TAAnCKS2	TAAnCKS1	TAAnCKS0	Selection of internal count clock				
				Input	n = 0, 2, 4, 6		n = 1, 3, 5, 7	
					PRSI =		PRSI =	
					0	1	0	1
0	0	0	0	f_{XP1}	f_{XX}	$f_{XX}/2$	f_{XX}	$f_{XX}/2$
1				f_{XP2}[b]	f_{XX}	$f_{XX}/2$	f_{XX}	$f_{XX}/2$
0	0	0	1	$f_{XP1}/2$	$f_{XX}/2$	$f_{XX}/4$	$f_{XX}/2$	$f_{XX}/4$
1				$f_{XP2}/2$	$f_{XX}/2$	$f_{XX}/4$	$f_{XX}/2$	$f_{XX}/4$
0	0	1	0	$f_{XP1}/4$	$f_{XX}/4$	$f_{XX}/8$	$f_{XX}/4$	$f_{XX}/8$
1				$f_{XP2}/4$	$f_{XX}/4$	$f_{XX}/8$	$f_{XX}/4$	$f_{XX}/8$
×	0	1	1	$f_{XP1}/8$	$f_{XX}/8$	$f_{XX}/16$	$f_{XX}/8$	$f_{XX}/16$
×	1	0	0	$f_{XP1}/16$	$f_{XX}/16$	$f_{XX}/32$	$f_{XX}/16$	$f_{XX}/32$
×	1	0	1	$f_{XP1}/32$	$f_{XX}/32$	$f_{XX}/64$	$f_{XX}/32$	$f_{XX}/64$
×	1	1	0	$f_{XP1}/64$	$f_{XX}/64$	$f_{XX}/128$	$f_{XX}/64$	$f_{XX}/128$
×	1	1	1	$f_{XP1}/128$	$f_{XX}/128$	$f_{XX}/256$	–	–
				f_{XT}	–	–	f_{XT}	f_{XT}

[a] available only for V850ES/FK3

[b] f_{XP2} doesn't stop in IDLE1 mode. Refer to *“Selector control registers” on page 209* for details.

Caution: Do not set to SELCNT2.ISEL[4:0] = 1 at 32 MHz < f_{XX} ≤ 48 MHz.

Note: PRSI can be set by the option bytes (refer to *“Flash Mask Options” on page 330* for details.):

- PRSI = 0: f_{XX} ≤ 32 MHz
- PRSI = 1: 32 MHz < f_{XX} ≤ 48 MHz

Caution

1. Set bits TAAnCKS[2:0] only when TAAnCE = 0. When TAAnCE bit setting is changed from 0 to 1, TAAnCKS[2.0] bits can be set simultaneously.
2. When the main clock is stopped, the count operation with the subclock is not available.

(2) TAAnCTL1 - TAA timer control register 1

TAAn control register 1 is an 8-bit register that controls the operation of timer AA.

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

$\overline{\text{RESET}}$ input clears this register to 00H.

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

Address		
	TAA0CTL1: FFFFF591$_H$	TAA1CTL1: FFFFF5A1$_H$
	TAA2CTL1: FFFFF5B1$_H$	TAA3CTL1: FFFFF5C1$_H$
	TAA4CTL1: FFFFF5D1$_H$	TAA5CTL1: FFFFF5E1$_H$
	TAA6CTL1: FFFFF5F1$_H$	TAA6CTL1: FFFFF601$_H$

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

	7	6	5	4	3	2	1	0
TAAnCTL1	TAAnSYE	TAAnEST	TAAnEEE	0	0	TAAnMD2	TAAnMD1	TAAnMD0
	R/W	R/W	R/W	R	R	R/W	R/W	R/W

Table 11-7 TAAnCTL1 register contents (1/2)

Bit position	Bit name	Function
7	TAAnSYE	Controls the tuned operation mode of timer TAAn. 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 *“Timer AA/AB Synchronous Operation” on page 525*. **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.
6	TAAnEST	Controls the software trigger of timer TAAn. 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. **Note:** The read value of the TAAnEST bit is always 0.
5	TAAnEEE	Selects the count clock input of timer TAAn. 0: Use the internal clock (clock selected by TAAnCTL0.TAAnCKS[2:0] bits) 1: Use external clock (input edge of TIAAn0) The valid edge is specified with TAAnEES1 and TAAnEES0 bits when TAAnEEE bit = 1 (external clock TIAAn0).

Table 11-7 TAAnCTL1 register contents (2/2)

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>2 to 0</td><td>TAAnMD [2:0]</td><td>Selects the operation mode of timer TAAn.
<table>
<tr><th>TAAnMD2</th><th>TAAnMD1</th><th>TAAnMD0</th><th>Timer mode selection</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>Interval timer mode</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>External event counter mode</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>External trigger pulse output mode</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>One-shot pulse mode</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>PWM mode</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Free-running mode</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>Pulse width measurement mode</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>Setting prohibited</td></tr>
</table>

Caution: 1. Set bits TAAnEEE and TAAnMD[2:0] 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 using the 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 (TAAnIOC2.TAAnEES[1:0] = 00B) when using the external event count mode.</td></tr>
</table>

(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

TAA0IOC0:	FFFFF592$_H$	TAA1IOC0:	FFFFF5A2$_H$
TAA2IOC0:	FFFFF5B2$_H$	TAA3IOC0:	FFFFFC2$_H$
TAA4IOC0:	FFFFF5D2$_H$	TAA5IOC0:	FFFFF5E2$_H$
TAA6IOC0:	FFFFF5F2$_H$	TAA7IOC0:	FFFFF602$_H$

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

	7	6	5	4	3	2	1	0
TAAnIOC0	0	0	0	0	TAAnOL1	TAAnOE1	TAAnOL0	TAAnOE0
	R	R	R	R	R/W	R/W	R/W	R/W

Caution

1. Rewrite bits TAAnOLm and TAAnOEm when TAAnCTL0.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 TAAnIOC1.TAAnIS[3:0] to “No edge detection” and invalidate the capture operation. Then set the corresponding alternate-function port to output mode.

Table 11-8 TAAnIOC0 register contents

Bit position	Bit name	Function
3, 1	TAAnOL1 TAAnOL0	Specifies the TOAAnm output level. 0: Normal output (inactive level = L, active level = H) 1: Inverted output (inactive level = H, active level = L) This bit can be used to invert the timer output
2, 0	TAAnOE1 TAAnOE0	Controls the TOAAnm output. 0: Timer output is disabled (inactive level is output depending on the TAAnOLm bit) 1: Timer output is enabled (TOAAnm pin outputs pulses.)

Note m = 0, 1

(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

TAA0IOC1:	$FFFFF593_H$	TAA1IOC1:	$FFFFF5A3_H$
TAA2IOC1:	$FFFFF5B3_H$	TAA3IOC1:	$FFFFF5C3_H$
TAA4IOC1:	$FFFFF5D3_H$	TAA5IOC1:	$FFFFF5E3_H$
TAA6IOC1:	$FFFFF5F3_H$	TAA7IOC1:	$FFFFF603_H$

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

	7	6	5	4	3	2	1	0
TAAnIOC1	0	0	0	0	TAAnIS3	TAAnIS2	TAAnIS1	TAAnIS0
	R	R	R	R	R/W	R/W	R/W	R/W

Caution

1. Bits TAAnIS[3:0] are valid only in the free-running capture mode and pulse width measurement mode. In all the other modes, capture operation is not performed.
2. If used as the capture input, be sure to set the corresponding bits TAAnIOC0. TAAnOE[1:0] 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" (TAAnIS[1:0] = 00_B).

Table 11-9 TAAnIOC1 register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>3, 2</td><td>TAAIS3
TAAnIS2</td><td>Specifies the capture input (TIAAn1) valid edge.
<table>
<tr><th>TAAnIS3</th><th>TAAnIS2</th><th>Capture input (TIAAn1) valid edge setting</th></tr>
<tr><td>0</td><td>0</td><td>No edge detection (capture operation invalid)</td></tr>
<tr><td>0</td><td>1</td><td>Rising edge detection</td></tr>
<tr><td>1</td><td>0</td><td>Falling edge detection</td></tr>
<tr><td>1</td><td>1</td><td>Both, rising and falling edge detection</td></tr>
</table>
Capture operation is performed and capture interrupt (INTTAAnCC1) is output upon edge detection.</td></tr>
<tr><td>1, 0</td><td>TAAnIS1
TAAnIS0</td><td>Specifies the capture input (TIAAn0) valid edge.
<table>
<tr><th>TAAnIS1</th><th>TAAnIS0</th><th>Capture input (TIAAn0) valid edge setting</th></tr>
<tr><td>0</td><td>0</td><td>No edge detection (capture operation invalid)</td></tr>
<tr><td>0</td><td>1</td><td>Rising edge detection</td></tr>
<tr><td>1</td><td>0</td><td>Falling edge detection</td></tr>
<tr><td>1</td><td>1</td><td>Both, rising and falling edge detection</td></tr>
</table>
Capture operation is performed and capture interrupt (INTTAAnCC0) is output upon edge detection.</td></tr>
</table>

Rewrite during timer operation

If the edge specification for the capture operation shall be changed, while the timer remains in operation (TAAnCTL0.TAAnCE = 1), only a single bit of the edge specification bits TAAnIOC1.TAAnIS[k:i] of a dedicated capture input may be changed with a single write operation.

Consequently proceed as follows (TIAAn0 is used exemplarily):

- 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”

(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 (TIAAn0) and external trigger input signal (TIAAn0).

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

Address

TAA0IOC2:	$FFFFF594_H$	TAA1IOC2:	$FFFFF5A4_H$
TAA2IOC2:	$FFFFF5B4_H$	TAA3IOC2:	$FFFFF5C4_H$
TAA4IOC2:	$FFFFF5D4_H$	TAA5IOC2:	$FFFFF5E4_H$
TAA6IOC2:	$FFFFF5F4_H$	TAA7IOC2:	$FFFFF604_H$

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

	7	6	5	4	3	2	1	0
TAAnIOC2	0	0	0	0	TAAnEES1	TAAnEES0	TAAnETS1	TAAnETS0
	R	R	R	R	R/W	R/W	R/W	R/W

Caution Rewrite TAAnEES[1:0] and TAAnETS[1:0] bits when TAAnCTL0.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.

Table 11-10 TAAnIOC2 register contents

Bit position	Bit name	Function
3, 2	TAAnEES1 TAAnEES0	Specifies the external event counter input (TEVTAAn) valid edge.
		TAAnEES1 | TAAnEES0 | External event counter input (TEVTAAn) valid edge setting
		0 | 0 | No edge detection (external event count is invalid)
		0 | 1 | Rising edge detection
		1 | 0 | Falling edge detection
		1 | 1 | Both, rising and falling edge detection
		Caution: 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).
1, 0	TAAnETS1 TAAnETS0	Specifies the external trigger input (TTRGAAn) valid edge.
		TAAnETS1 | TAAnETS0 | External trigger input (TTRGAAn) valid edge setting
		0 | 0 | No edge detection (external trigger is invalid)
		0 | 1 | Rising edge detection
		1 | 0 | Falling edge detection
		1 | 1 | Both, rising and falling edge detection
		Caution: 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:i] / TAAnIOC2.TAAnETS[k:i] of a dedicated capture input may be changed with a single write operation.

Consequently proceed as follows (TIAAn0 is used exemplarily):

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.

(6) TAAnIOC4 - TAA I/O control register 4

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

TAA0IOC4:	FFFFF59C$_H$	TAA1IOC4	FFFFF5AC$_H$
TAA2IOC4:	FFFFF5BC$_H$	TAA3IOC4:	FFFFF5CC$_H$
TAA4IOC4:	FFFFF5DC$_H$	TAA5IOC4:	FFFFF5EC$_H$
TAA6IOC4:	FFFFF5FC$_H$	TAA7IOC4:	FFFFF60C$_H$

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

	7	6	5	4	3	2	1	0
TAAnIOC4	0	0	0	0	TAAnOS1	TAAnOR1	TAAnOS0	TAAnOR0
	R	R	R	R	R/W	R/W	R/W	R/W

Caution The TAAnIOC4 register can only be used when interval mode or free-running compare mode is selected. In all other modes the TAAnIOC4 register has to be set to 00$_H$.

Note

1. Writing to TAAnIOC4 is also possible, when TAAnCTL0.TAAnCE = 1.
2. In the free running mode, the setting's of the TAAnIOC4 register becomes effective only if the compare function is selected. When the capture function is selected, it is invalid.

Table 11-11 TAAnIOC4 register contents (1/2)

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>3, 2</td><td>TAAnOS1
TAAnOR1</td><td>Controls toggling of the timer output TOAAn1.
<table>
<tr><th>TAAnOS1</th><th>TAAnOR1</th><th>Toggle Control of TOAAn1</th></tr>
<tr><td>0</td><td>0</td><td>Standard operation.</td></tr>
<tr><td>0</td><td>1</td><td>Force output level to inactive at next toggle event</td></tr>
<tr><td>1</td><td>0</td><td>Force output level to active at next toggle event</td></tr>
<tr><td>1</td><td>1</td><td>Freeze current output level.</td></tr>
</table>
<b>Note:</b> 1. After forcing the output level to either active or inactive, the TOAAn1 output maintains this level (= no toggling afterwards) until the TAAnOS1 and TAAnOR1 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.</td></tr>
</table>

Table 11-11 TAAnIOC4 register contents (2/2)

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>1, 0</td><td>TAAnOS0
TAAnOR0</td><td>Controls toggling of the timer output TOAAn0.
<table>
<tr><th>TAAnOS0</th><th>TAAnOR0</th><th>Toggle Control of TOAAn0</th></tr>
<tr><td>0</td><td>0</td><td>Standard operation.</td></tr>
<tr><td>0</td><td>1</td><td>Force output level to inactive at next toggle event</td></tr>
<tr><td>1</td><td>0</td><td>Force output level to active at next toggle event</td></tr>
<tr><td>1</td><td>1</td><td>Freeze current output level.</td></tr>
</table>
<b>Note:</b> <b>1.</b> After forcing the output level to either active or inactive, the TOAAn0 output maintains this level (= no toggling afterwards) until the TAAnOS0 and TAAnOR0 are cleared to standard operation.
<b>2.</b> The forcing of an output level is executed at the time of the next upcoming toggle event, while the freeze becomes effective immediately.</td></tr>
</table>

(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

TAA0OPT0: FFFFF595$_H$	TAA1OPT0: FFFFF5A5$_H$
TAA2OPT0: FFFFF5B5$_H$	TAA3OPT0: FFFFF5C5$_H$
TAA4OPT0: FFFFF5D5$_H$	TAA5OPT0: FFFFF5E5$_H$
TAA6OPT0: FFFFF5F5$_H$	TAA7OPT0: FFFFF605$_H$

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

	7	6	5	4	3	2	1	0
TAAnOPT0	0	0	TAAnCCS1	TAAnCCS0	0	0	0	TAAnOVF
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Caution Rewrite TAAnCCS1 and TAAnCCS0 bits when TAAnCTL0.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.

Table 11-12 TAAnOPT0 register contents

Bit position	Bit name	Function
5	TAAnCCS1	Specifies the operation mode of register TAAnCCR1 0: Operation as compare register 1: Operation as capture register **Note:** The setting of bit TAAnCCS1 is valid in the free-running mode only.
4	TAAnCCS0	Specifies the operation mode of register TAAnCCR0 0: Operation as compare register 1: Operation as capture register **Note:** The setting of bit TAAnCCS0 is valid in the free-running mode only.
0	TAAnOVF	Indicates timer TAAn overflow 0: No overflow occurrence after timer restart or flag reset 1: Overflow occurrence • The TAAnOVF bit is set when the 16-bit counter value overflows from FFFFH to 0000H 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 cleared by writing 0 to it, or if TAAnCTL0.TAAnCE is set to 0. **Caution:** 1. When TAAnOVF = 1, the TAAnOVF flag is not cleared even if the TAAnOVF flag and TAAnOPT0 register are read. 2. The TAAnOVF flag can be read and written, but writing 1 to TAAnOVF does not set it and has no influence on the operation of timer AA.

(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.

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

Address TAA1OPT1: FFFFF5AD$_H$
TAA3OPT1: FFFFF5CD$_H$
TAA6OPT1: FFFFF5FD$_H$

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

	7	6	5	4	3	2	1	0
TAAnOPT1	TAAnCSE	0	0	0	0	0	0	0
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 11-13 TAAnOPT0 register contents

Bit position	Bit name	Function
7	TAAnCSE	Controls the cascade mode of timers TAAn/TAAm. 0: 16-bit non-cascaded mode 1: 32-bit cascaded capture mode. Timer AAn becomes the upper 16-bit and slave. The master timer is TAAm (where m = n -1). **Note:** **1.** When setting TAAnCSE, the timer becomes the upper 16-bit of a 32-bit timer. **2.** If TAAnCSE = 1, TAAnCTL0.TAAnCE is forced to “0”. **3.** Cascading is only available for capture with free-running counter. **4.** The following pairs of timers can be cascaded: • TAA0 and TAA1 (TAA0 will become master and will hold the lower 16-bit value) • TAA2 and TAA3 (TAA2 will become master and will hold the lower 16-bit value) • TAA5 and TAA6 (TAA5 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:

Table 11-14 Timer Operation in Non-cascaded/Cascaded Capture Mode

	TAAnCSE = 0	TAAnCSE = 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

Note n=1, 3 or 6; m= (n-1)

For details on the 32-bit capture mode, please refer to *“32-bit Capture in Free-Running Cascade Mode” on page 462*.

11.6 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[a]	Invalid	Invalid	External only	Compare only	Any time write
External trigger pulse output mode[b]	Valid	Valid	Internal only	Compare only	Reload
One-shot pulse output mode[b]	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[b]	Invalid	Invalid	Internal only	Capture only	Not applicable

a) When the external event count function is used, set the edge detection of the TIAAn0 capture input to "No edge detection" (TAAnIOC1.TAAnIS[1:0] = 00_B).

b) To use the external trigger pulse output mode, one-shot pulse mode, or pulse width measurement mode, select a count clock by clearing the TAAnCTL1.TAAnEEE bit to 0.

11.6.1 Anytime write and reload

TAAnCCR0 and TAAnCCR1 register rewrite is possible for timer AA during timer operation (TAAnCE = 1), but the write method (any time 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.

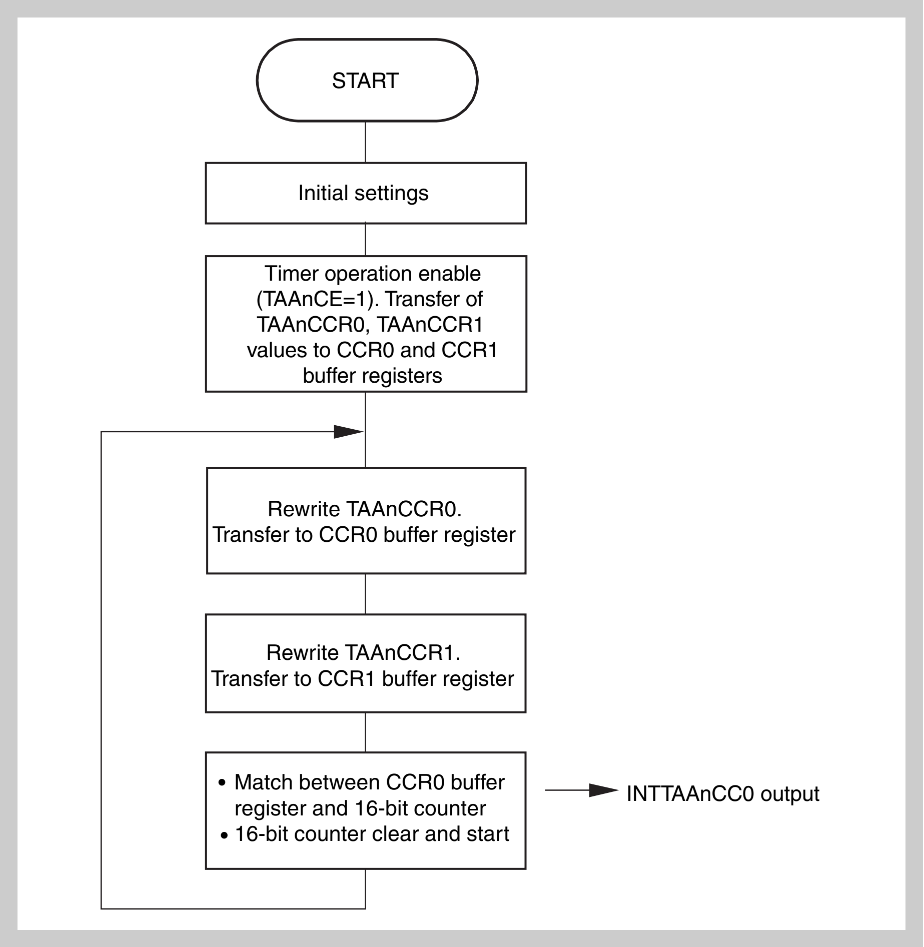

START
Initial settings
Timer operation enable (TAAnCE=1). Transfer of TAAnCCR0, TAAnCCR1 values to CCR0 and CCR1 buffer registers
Rewrite TAAnCCR0. Transfer to CCR0 buffer register
Rewrite TAAnCCR1. Transfer to CCR1 buffer register
• Match between CCR0 buffer register and 16-bit counter
• 16-bit counter clear and start
INTTAAnCC0 output

Figure 11-6 Flowchart of basic operation for anytime write

Note 1. The above flowchart illustrates an example of the operation in the interval timer mode.

2. m = 0, 1

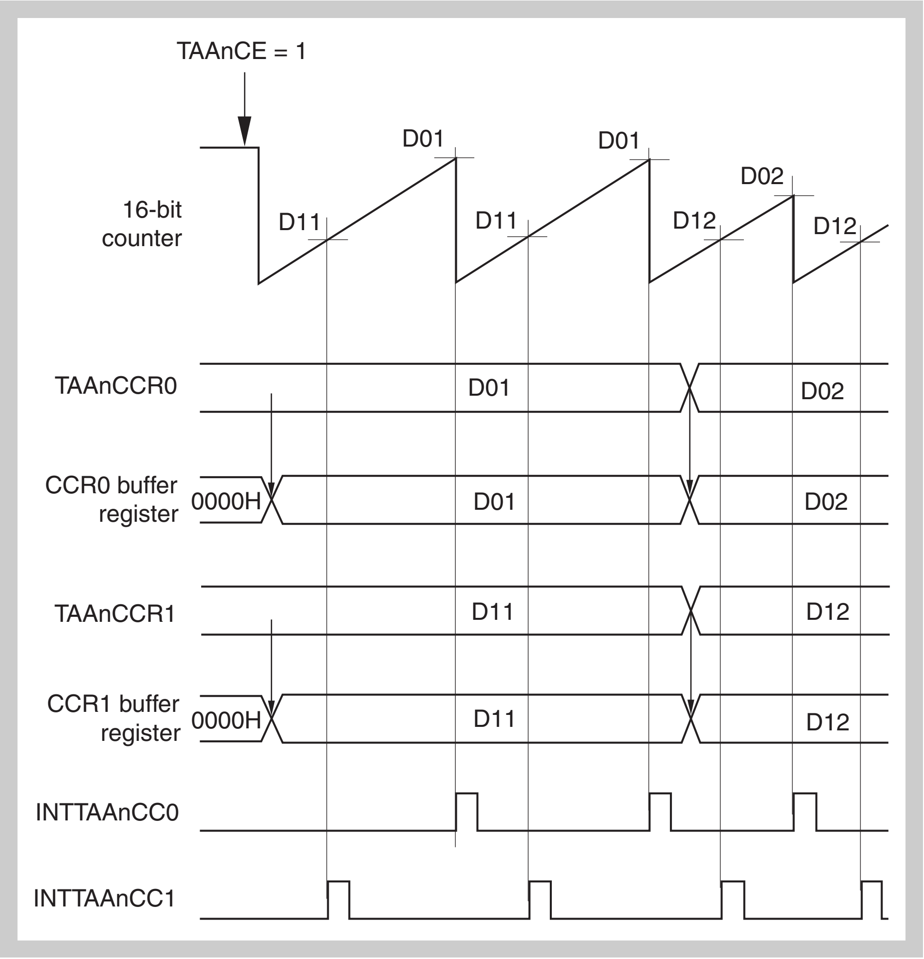


Figure 11-7 Timing diagram for anytime write

D01, D02: Setting values of TAAnCCR0 register (0000_H to $FFFF_H$)
D11, D12: Setting values of TAAnCCR1 register (0000_H to $FFFF_H$)

The above timing chart illustrates an example of the operation in the interval timer mode.

Caution Though the compare registers can be written at any time, the write access will be synchronized with the internal count clock, depending on the corresponding ISEL bit (of the SELCNT2 or SELCNT5 register), TAAnCTL0.TAAnCKS[2:0] bits and PRSI bit (of the option byte 0000 $007B_H$). Due to this synchronization a delay has to be taken into account.
Particularly when the dedicated interrupt request flag of a capture/compare interrupt is cleared directly after rewriting the capture/compare register, an unexpected interrupt request may occur, since the new compare value is still not synchronized and accepted. The occurrence of the accidental interrupt can be avoided by a certain delay between capture/compare register write and clearing of the interrupt request flag. An applicable delay can be achieved by a consecutive write of the same capture/compare register.

Example

1. Write capture/compare register
2. Write same capture/compare register again
   (delays program execution until the synchronization takes effect)
3. Clear dedicated interrupt request flag

(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 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 the value of only the TAAnCCR0 register.

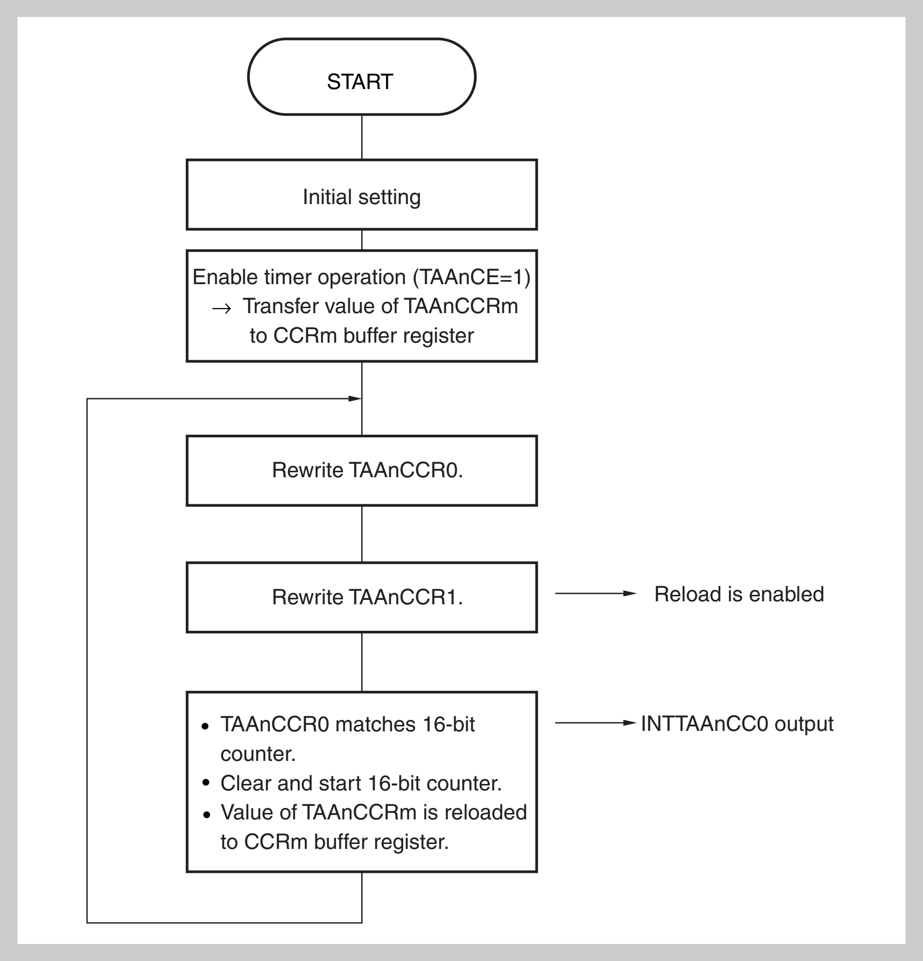

START
Initial setting
Enable timer operation (TAAnCE=1)
→ Transfer value of TAAnCCRm to CCRm buffer register
Rewrite TAAnCCR0.
Rewrite TAAnCCR1.
Reload is enabled
• TAAnCCR0 matches 16-bit counter.
• Clear and start 16-bit counter.
• Value of TAAnCCRm is reloaded to CCRm buffer register.
INTTAAnCC0 output

Figure 11-8 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. Above flowchart illustrates an example of the PWM mode operation.
2. m = 0, 1

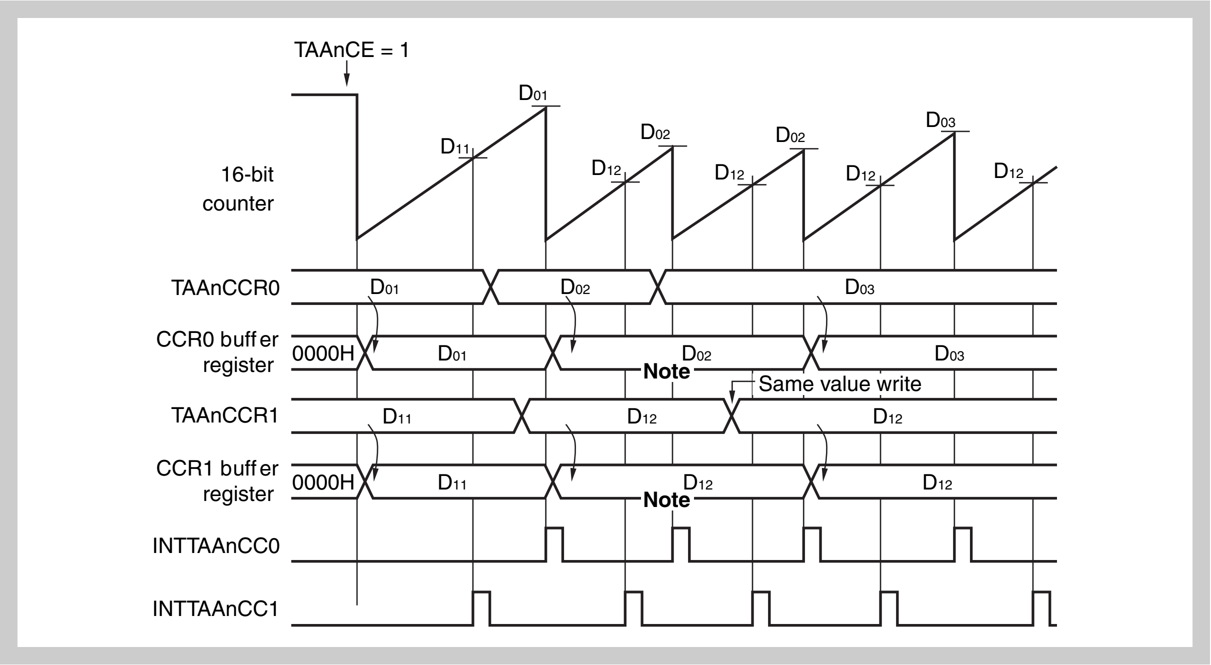


Figure 11-9 Timing chart for reload

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 $FFFF_H$)

Above flowchart illustrates PWM mode operation.

11.6.2 Interval timer mode (TAAnMD2 to TAAnMD0 = 000_B)

In the interval timer mode, an interrupt request signal (INTTAAnCC0) is generated upon a match between the setting value of the TAAnCCR0 register and the value of the 16-bit counter, and the 16-bit counter is cleared. 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 any time 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, it is recommended to set FFFF_H as the setting value for the TAAnCCR1 register.

When performing timer output with the TOAAn1 pin, set the same values to the TAAnCCR0 register and the TAAnCCR1 register since the 16-bit timer counter cannot be cleared with the TAAnCCR1 register.

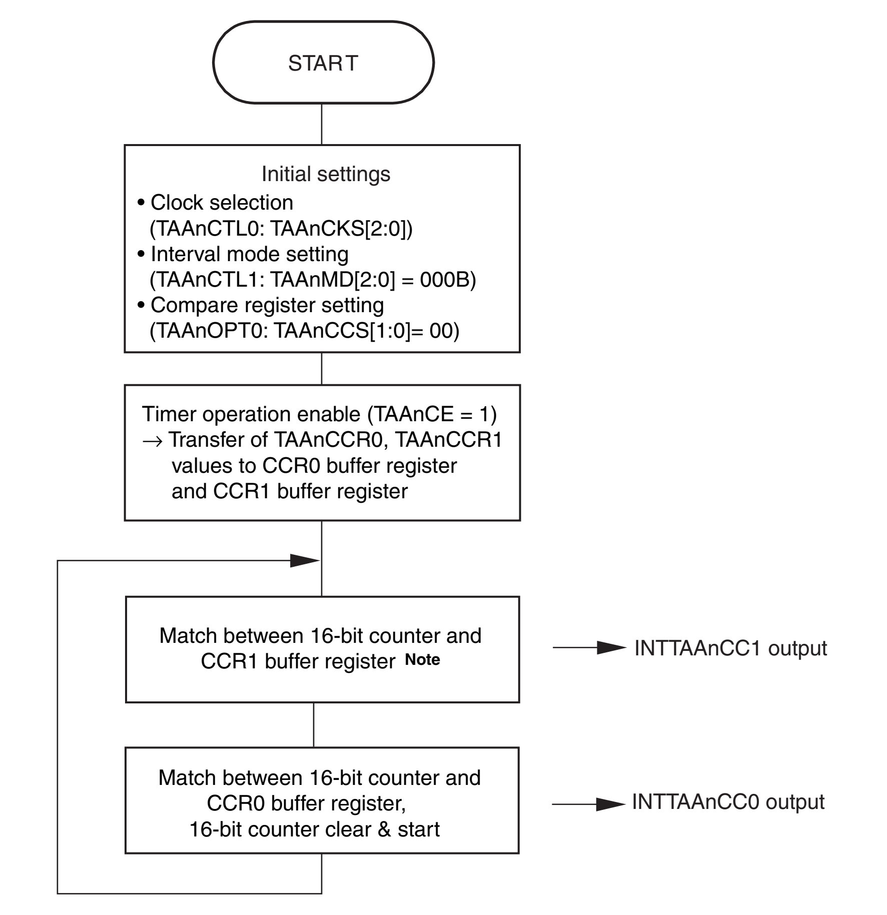

START
Initial settings
• Clock selection
(TAAnCTL0: TAAnCKS[2:0])
• Interval mode setting
(TAAnCTL1: TAAnMD[2:0] = 000B)
• Compare register setting
(TAAnOPT0: TAAnCCS[1:0]= 00)
Timer operation enable (TAAnCE = 1)
→ Transfer of TAAnCCR0, TAAnCCR1 values to CCR0 buffer register and CCR1 buffer register
Match between 16-bit counter and CCR1 buffer register Note
INTTAAnCC1 output
Match between 16-bit counter and CCR0 buffer register, 16-bit counter clear & start
INTTAAnCC0 output

Figure 11-10 Flowchart of basic operation in interval timer mode

Note The 16-bit counter is not cleared when its value matches the value of TAAnCCR1.

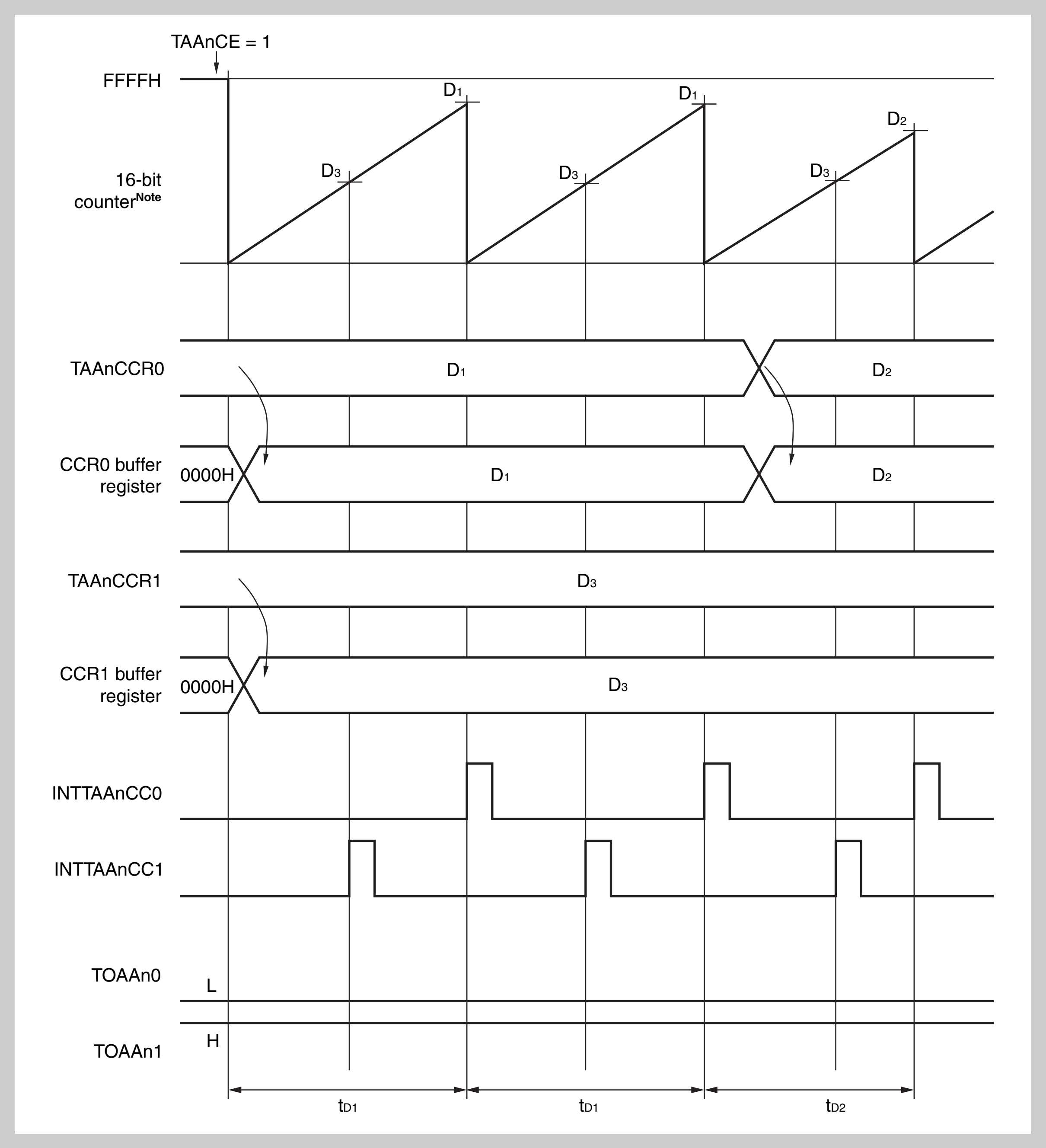


Figure 11-11 Basic operation timing in interval timer mode when D1 > D2 > D3; only TAAnCCR0 register value is written, and TOAAn0 and TOAAn1 are not output (TAAnOE0 = 0, TAAnOE1 = 0, TAAnOL0 = 0, TAAnOL1 = 1)

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)

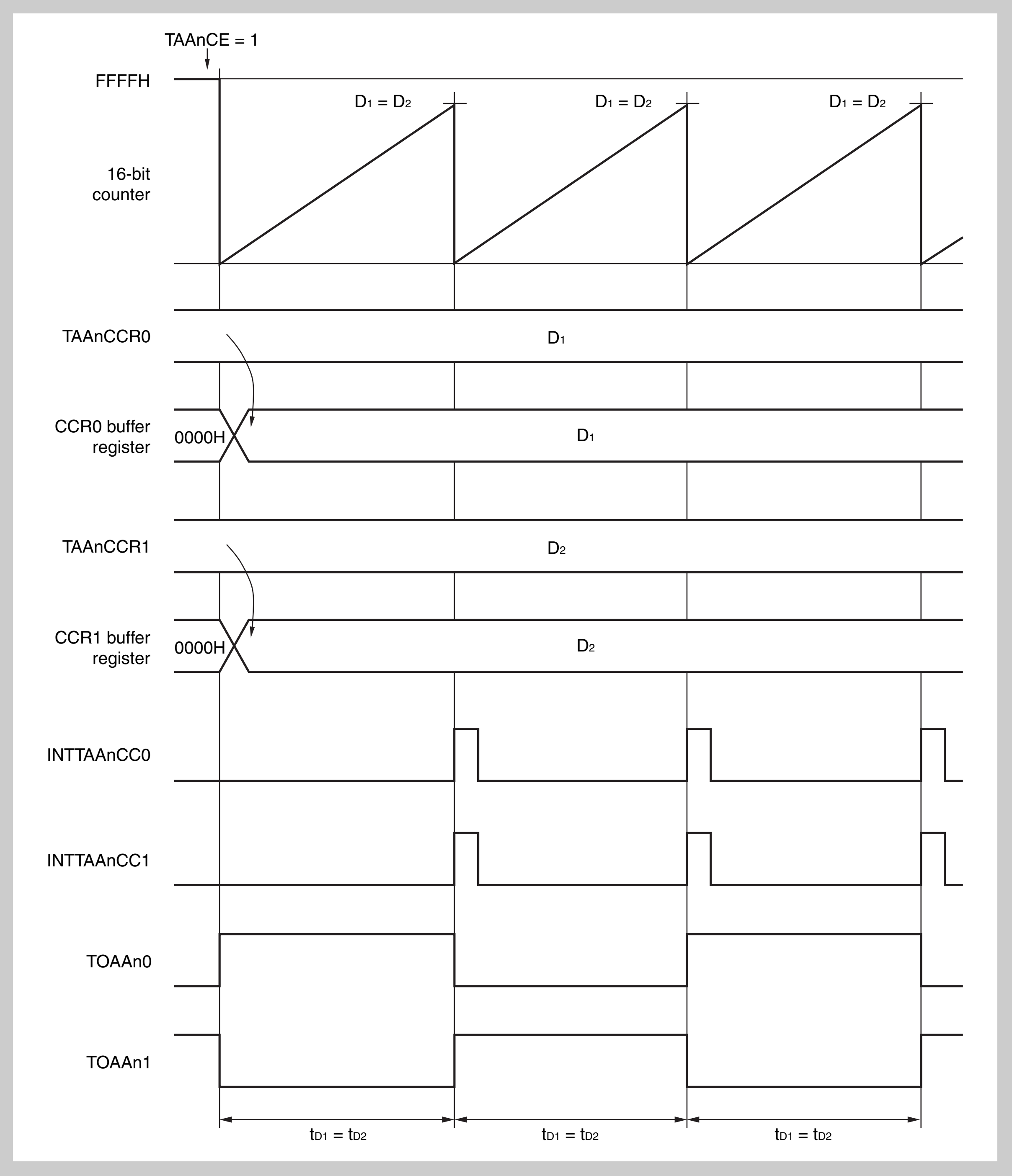


Figure 11-12 Basic operation timing in interval timer mode when D1 = D2; TAAnCCR0 and TAAnCCR1 are not rewritten, and TOAAn0 and TOAAn1 are output (TAAnOE0 = 1, TAAnOE1 = 1, TAAnOL0 = 0, TAAnOL1 = 1)

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)

When a new value is written to the TAAnCCR0 register that is smaller than the TAAn counter value at that moment, the counter will run to up to FFFF$_H$ and restart counting at 0000$_H$. When the value of the counter then matches the TAAnCCR0 register a compare event will occur..

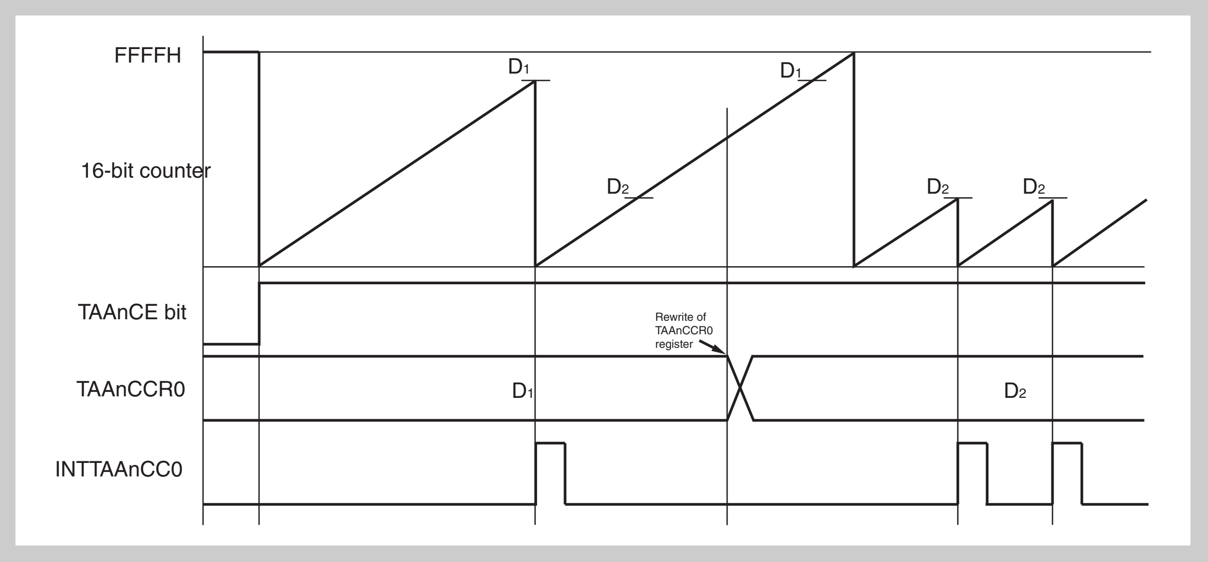


Figure 11-13 Rewriting of the compare register with a smaller value than the current counter value

11.6.3 External event counter mode (TAAnMD2 to TAAnMD0 = 001_B)

In the external event count mode, the external event count input (TIAAn0 pin input) is used as a count-up signal. Regardless of the setting of the TAAnCTL1.TAAnEEE bit, 16-bit timer/event counter AA counts up the external event count input (TIAAn0 pin input) when it is set in the external event count mode. In the external event count 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 any time 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 $FFFF_H$.

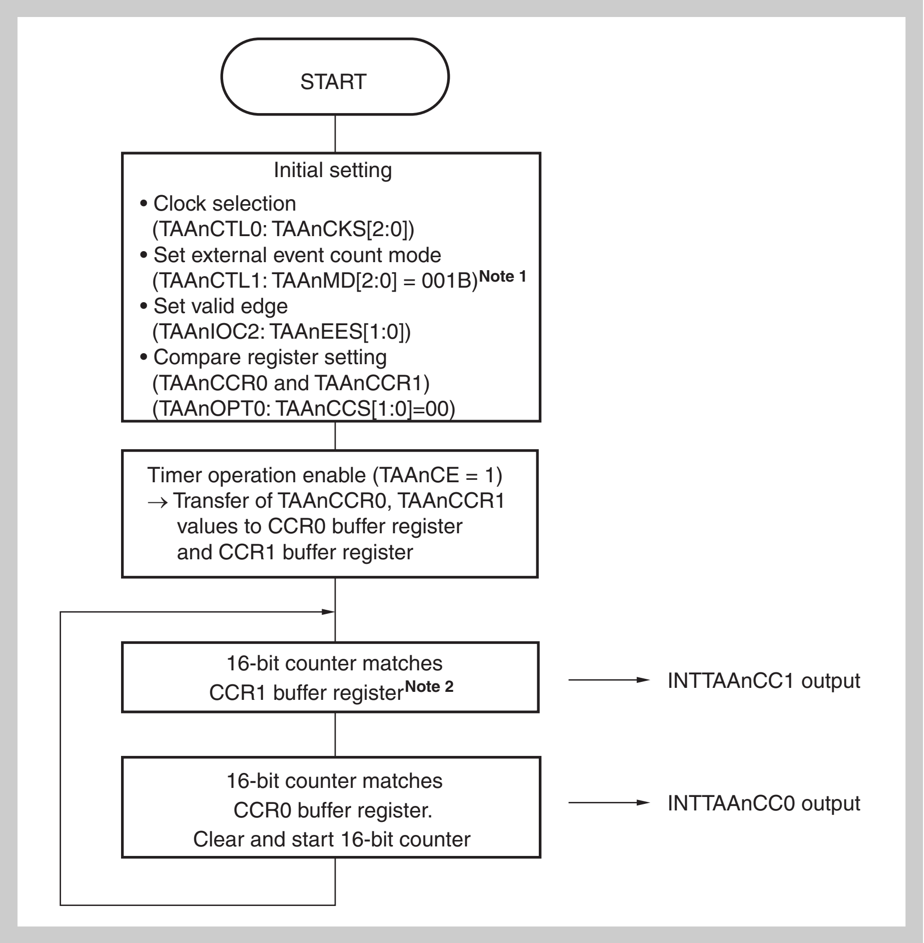

START
Initial setting
• Clock selection
(TAAnCTL0: TAAnCKS[2:0])
• Set external event count mode
(TAAnCTL1: TAAnMD[2:0] = 001B)Note 1
• Set valid edge
(TAAnIOC2: TAAnEES[1:0])
• Compare register setting
(TAAnCCR0 and TAAnCCR1)
(TAAnOPT0: TAAnCCS[1:0]=00)
Timer operation enable (TAAnCE = 1)
→ Transfer of TAAnCCR0, TAAnCCR1
values to CCR0 buffer register
and CCR1 buffer register
16-bit counter matches
CCR1 buffer registerNote 2
INTTAAnCC1 output
16-bit counter matches
CCR0 buffer register.
Clear and start 16-bit counter
INTTAAnCC0 output

Figure 11-14 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

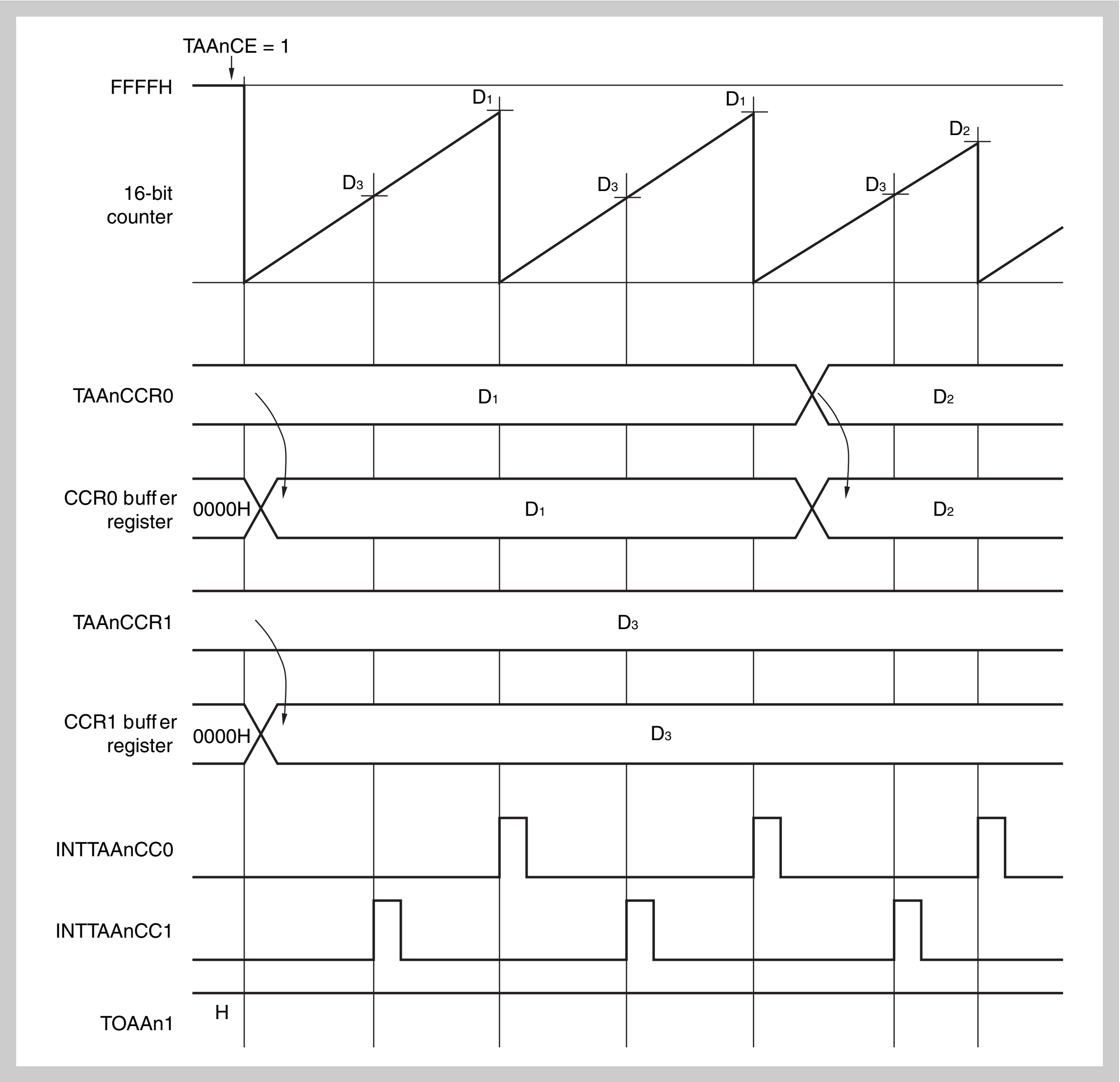


Figure 11-15 Basic Operation Timing in External Event Counter Mode When D1 > D2 > D3; rewrite TAAnCCR0 only; TOAAn1 is not output (TAAnOE0 = 0, TAAnOE1 = 0, TAAnOL0 = 0, TAAnOL1 = 1)

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)

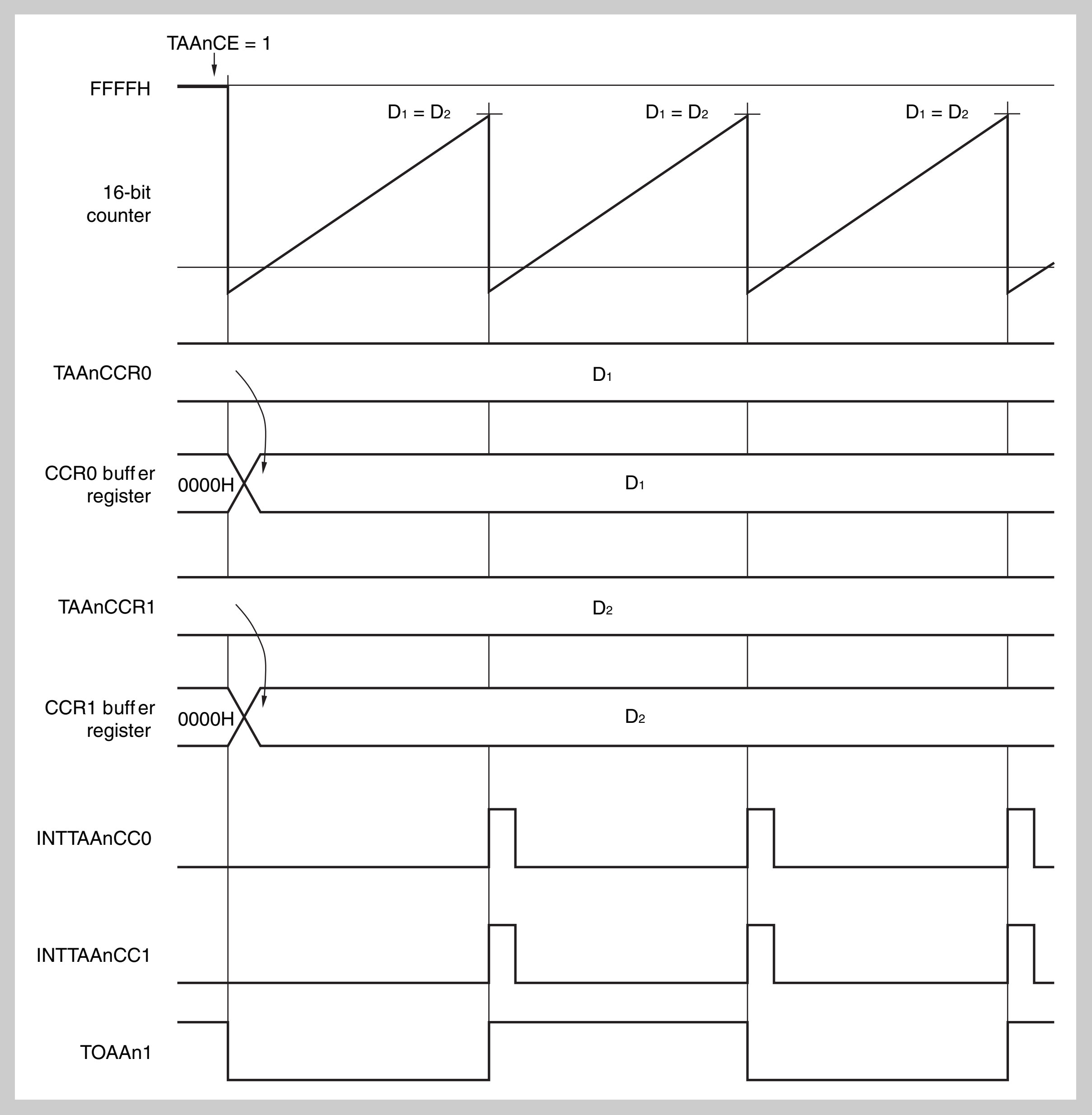


Figure 11-16 Basic Operation Timing in External Event Counter Mode When D1 = D2; TAAnCCR0 and TAAnCCR1 are not rewritten, TOAAn1 is output (TAAnOE0 = 1, TAAnOE1 = 1, TAAnOL0 = 0, TAAnOL1 = 1)

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)

11.6.4 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 of 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 the value of only the TAAnCCR0 register.

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 (TIAAn0 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 instead of the external trigger input (TIAAn0 pin input) (software trigger pulse mode), a software trigger is generated by setting the TAAnCTL1.TAAnEST bit 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 outputed 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 these registers can be used only 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 *“PWM mode (TAAnMD2 to TAAnMD0 = 100_B)” on page 444*.
2. m = 0, 1

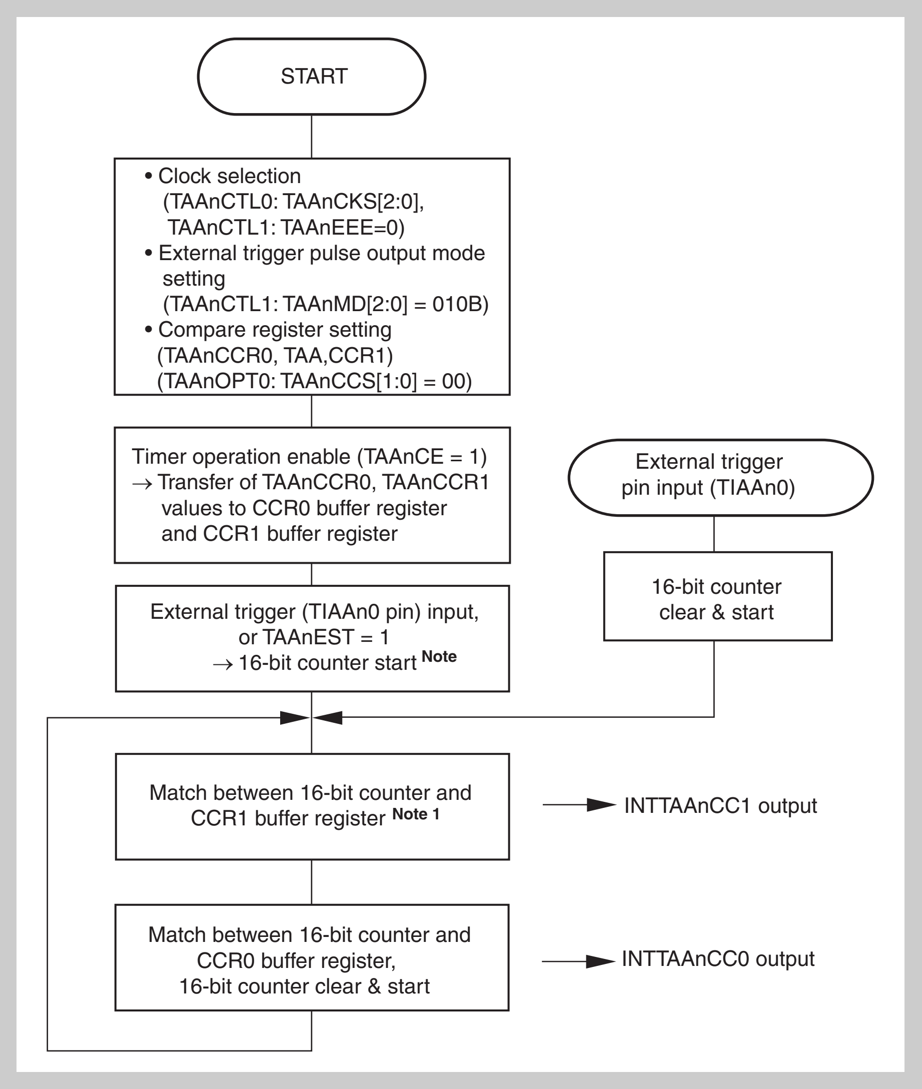

START
• Clock selection
(TAAnCTL0: TAAnCKS[2:0],
TAAnCTL1: TAAnEEE=0)
• External trigger pulse output mode setting
(TAAnCTL1: TAAnMD[2:0] = 010B)
• Compare register setting
(TAAnCCR0, TAA,CCR1)
(TAAnOPT0: TAAnCCS[1:0] = 00)
Timer operation enable (TAAnCE = 1)
→ Transfer of TAAnCCR0, TAAnCCR1 values to CCR0 buffer register and CCR1 buffer register
External trigger (TIAAn0 pin) input, or TAAnEST = 1
→ 16-bit counter start Note
External trigger pin input (TIAAn0)
16-bit counter clear & start
Match between 16-bit counter and CCR1 buffer register Note 1
INTTAAnCC1 output
Match between 16-bit counter and CCR0 buffer register, 16-bit counter clear & start
INTTAAnCC0 output

Figure 11-17 Flowchart of Basic Operation in External Trigger Pulse Output Mode

Note The 16-bit counter is not cleared upon a match between the 16-bit counter and the CCR1 buffer register.

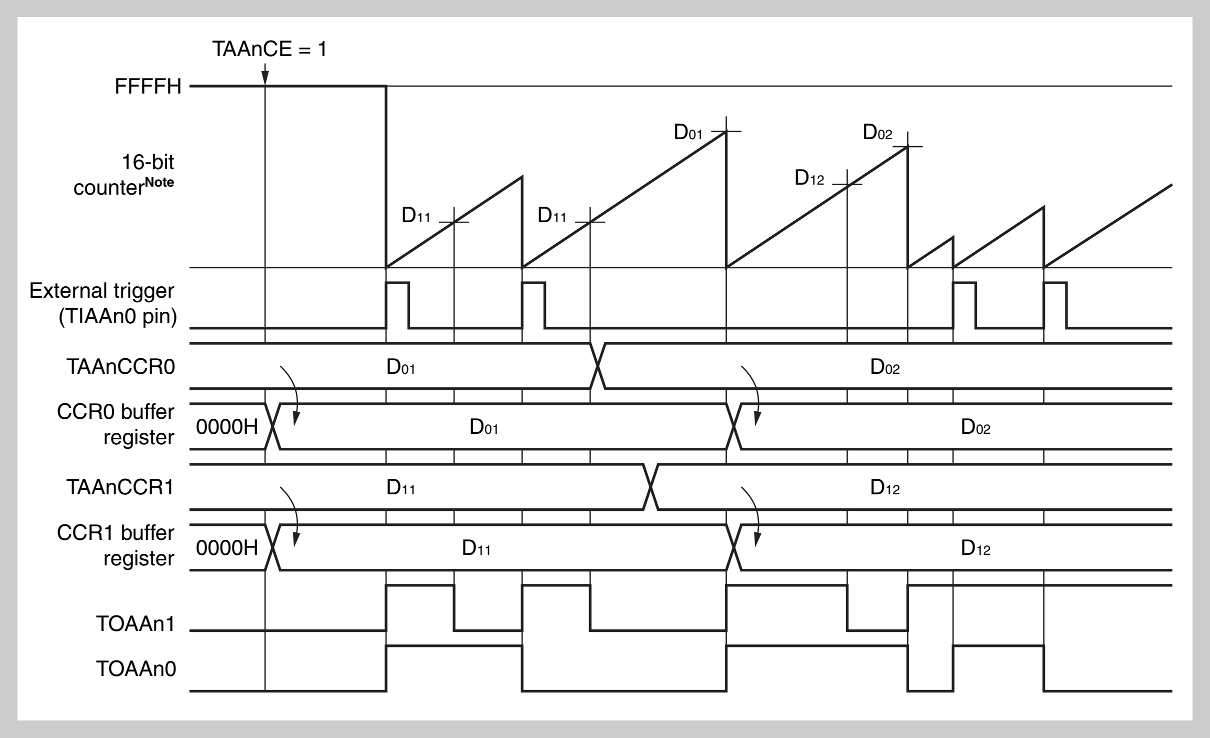


Figure 11-18 Basic Operation Timing in External Trigger Pulse Output Mode (TAAnOE0 = 1, TAAnOE1 = 1, TAAnOL0 = 0, TAAnOL1 = 0)

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)
Cycle of TOAAn1 output =
(Set value of TAAnCCR0 register + 1) · (Count clock cycle)

11.6.5 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 TIAAn0 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 0000H 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 0000H. 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.

Caution

1. In the one-shot pulse mode, select the internal clock (TAAnCTL1.TAAnEEE = 0) as the count clock.
2. In the one-shot pulse mode, it is prohibited to set the TAAnCCR1 register to 0000_H.

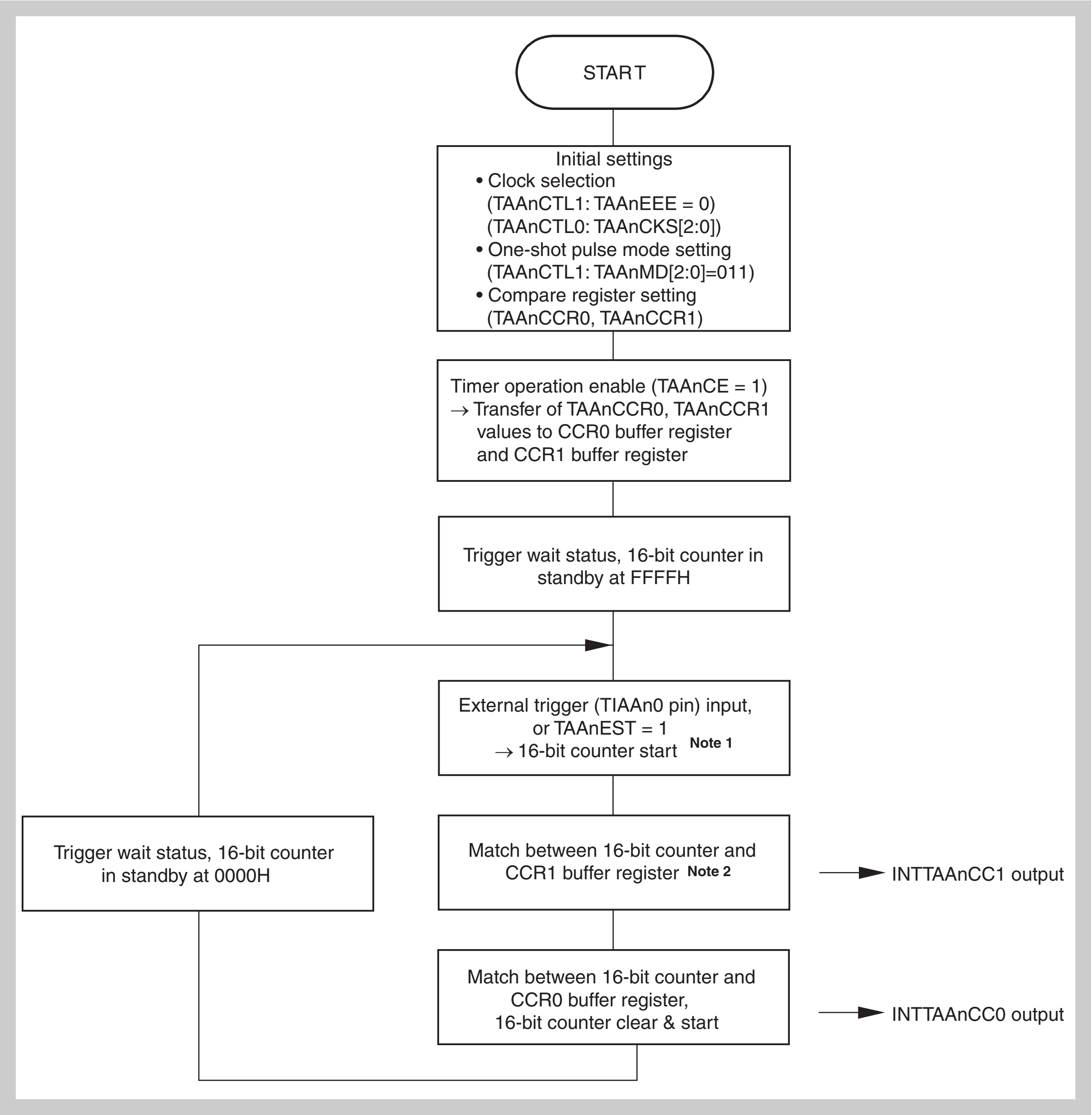

START
Initial settings
• Clock selection
(TAAnCTL1: TAAnEEE = 0)
(TAAnCTL0: TAAnCKS[2:0])
• One-shot pulse mode setting
(TAAnCTL1: TAAnMD[2:0]=011)
• Compare register setting
(TAAnCCR0, TAAnCCR1)
Timer operation enable (TAAnCE = 1)
→ Transfer of TAAnCCR0, TAAnCCR1 values to CCR0 buffer register and CCR1 buffer register
Trigger wait status, 16-bit counter in standby at FFFFH
External trigger (TIAAn0 pin) input, or TAAnEST = 1
→ 16-bit counter start Note 1
Match between 16-bit counter and CCR1 buffer register Note 2
INTTAAnCC1 output
Match between 16-bit counter and CCR0 buffer register, 16-bit counter clear & start
INTTAAnCC0 output
Trigger wait status, 16-bit counter in standby at 0000H

Figure 11-19 Flowchart of Basic Operation in One-Shot Pulse Mode

Note

1. Only the TAAnCTL1.TAAnEST bit 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.

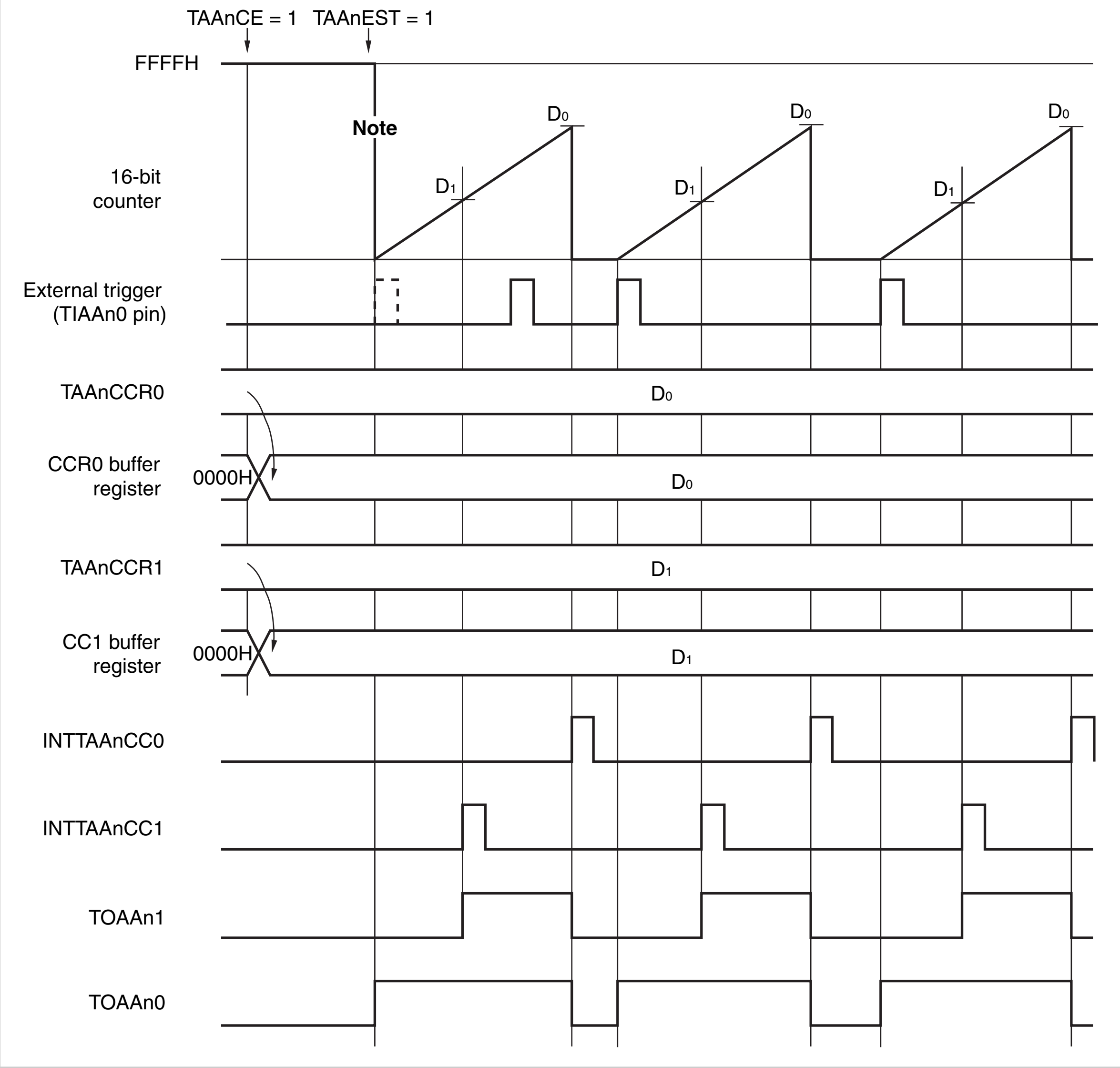


Figure 11-20 Timing of Basic Operation in One-Shot Pulse Mode (TAAnOE0 = 1, TAAnOE1 = 1, TAAnOL0 = 0, TAAnOL1 = 0)

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

11.6.6 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.

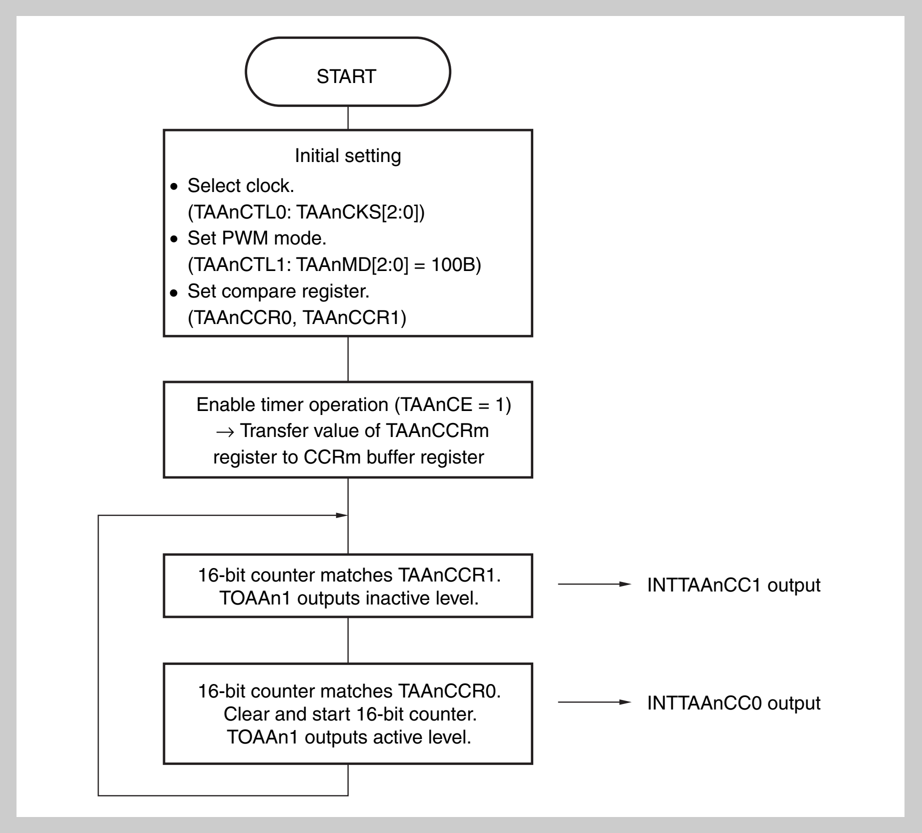

START
Initial setting
• Select clock.
(TAAnCTL0: TAAnCKS[2:0])
• Set PWM mode.
(TAAnCTL1: TAAnMD[2:0] = 100B)
• Set compare register.
(TAAnCCR0, TAAnCCR1)
Enable timer operation (TAAnCE = 1)
→ Transfer value of TAAnCCRm
register to CCRm buffer register
16-bit counter matches TAAnCCR1.
TOAAn1 outputs inactive level.
INTTAAnCC1 output
16-bit counter matches TAAnCCR0.
Clear and start 16-bit counter.
TOAAn1 outputs active level.
INTTAAnCC0 output

Figure 11-21 Flowchart of Basic Operation in PWM Mode When values of TAAnCCR0, TAAnCCR1 registers are not rewritten during timer operation

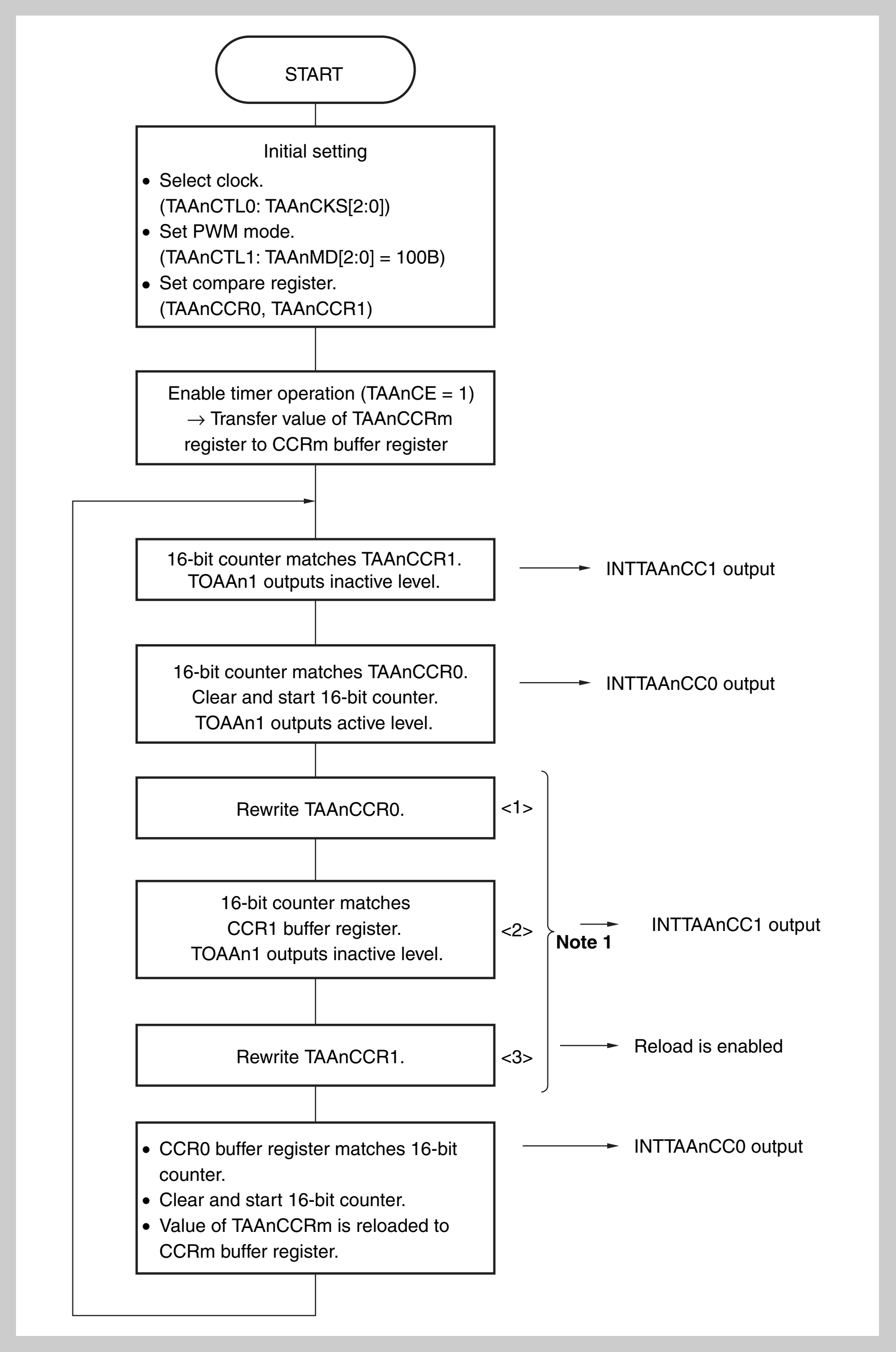

START
Initial setting
• Select clock.
(TAAnCTL0: TAAnCKS[2:0])
• Set PWM mode.
(TAAnCTL1: TAAnMD[2:0] = 100B)
• Set compare register.
(TAAnCCR0, TAAnCCR1)
Enable timer operation (TAAnCE = 1)
→ Transfer value of TAAnCCRm
register to CCRm buffer register
16-bit counter matches TAAnCCR1.
TOAAn1 outputs inactive level.
INTTAAnCC1 output
16-bit counter matches TAAnCCR0.
Clear and start 16-bit counter.
TOAAn1 outputs active level.
INTTAAnCC0 output
Rewrite TAAnCCR0.
<1>
16-bit counter matches
CCR1 buffer register.
TOAAn1 outputs inactive level.
<2>
Note 1
INTTAAnCC1 output
Rewrite TAAnCCR1.
<3>
Reload is enabled
• CCR0 buffer register matches 16-bit counter.
• Clear and start 16-bit counter.
• Value of TAAnCCRm is reloaded to CCRm buffer register.
INTTAAnCC0 output

Figure 11-22 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>.
2. m = 0, 1

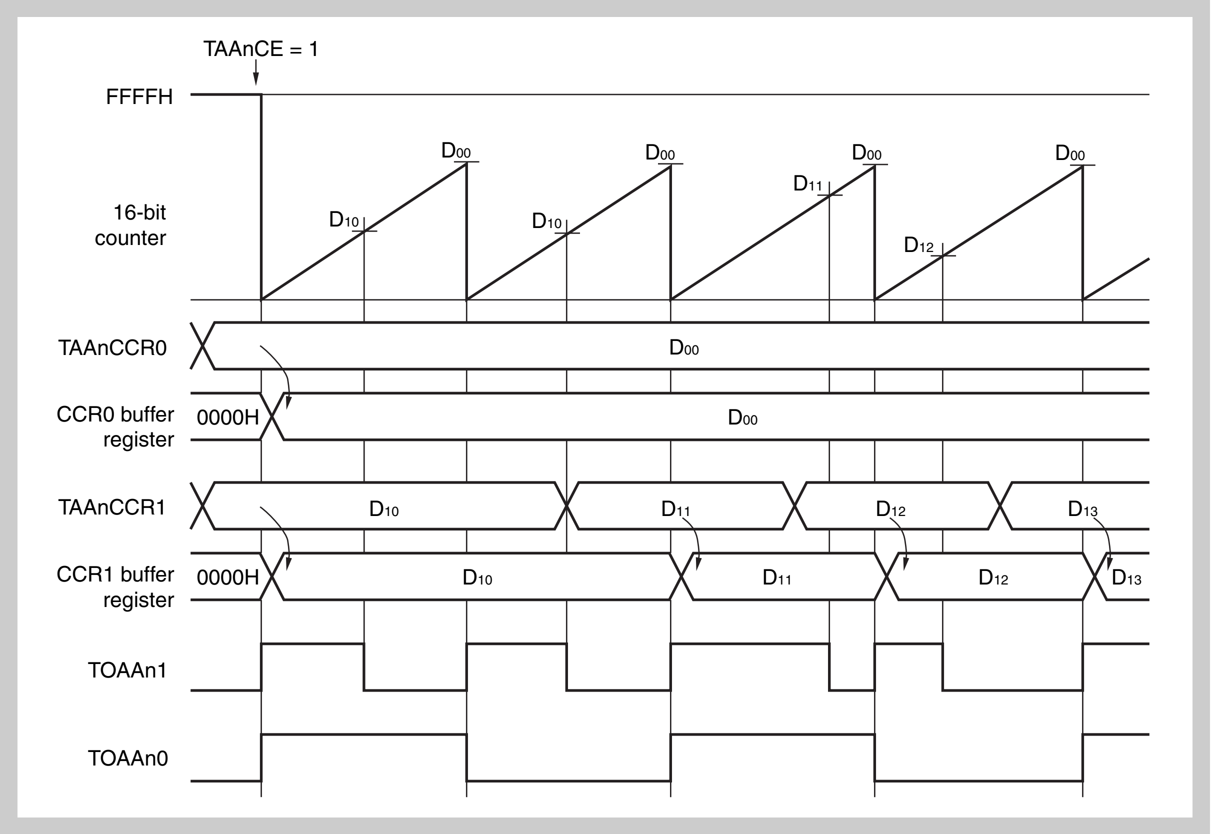


Figure 11-23 Basic Operation Timing in PWM Mode When rewriting TAAnCCR1 value (TAAnOE0 = 1, TAAnOE1 = 1, TAAnOL0 = 0, TAAnOL1 = 0)

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 factor 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)

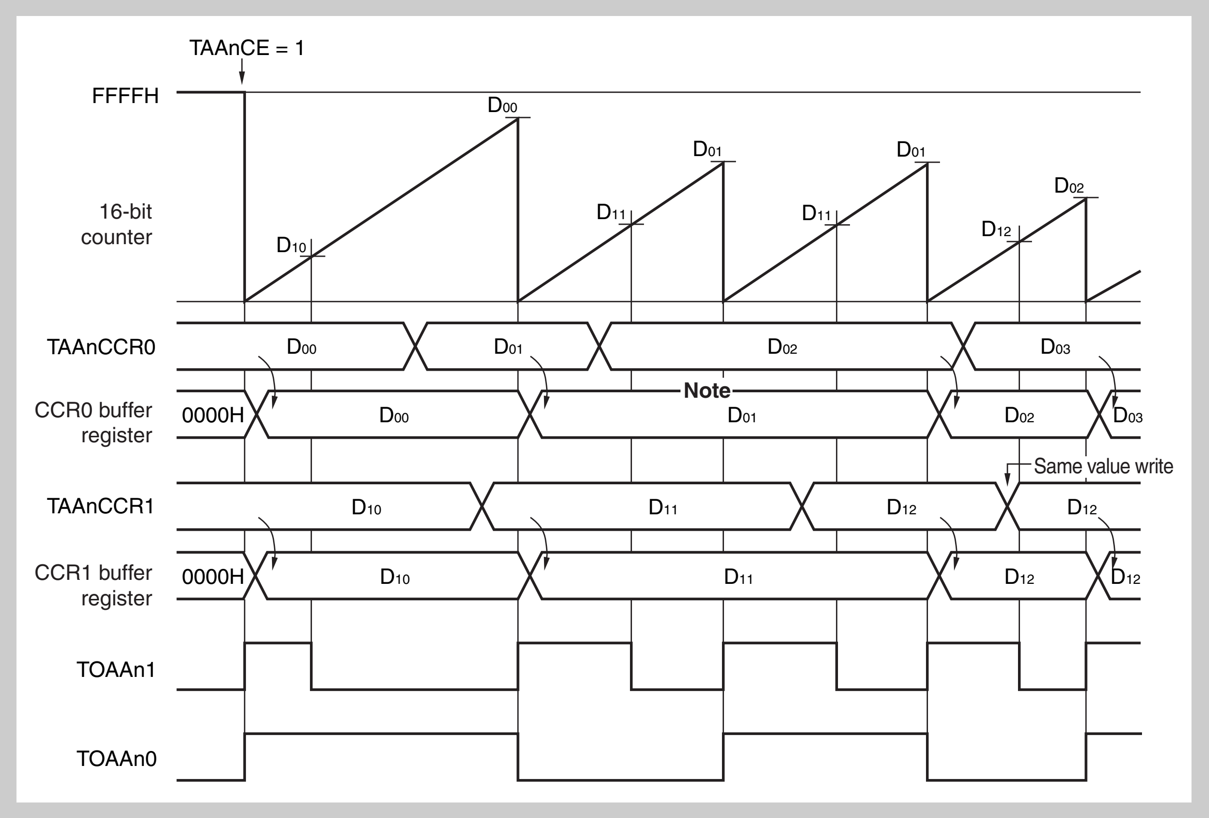


Figure 11-24 Basic Operation Timing in PWM Mode When TAAnCCR0, TAAnCCR1 values are rewritten (TAAnOE0 = 1, TAAnOE1 = 1, TAAnOL0 = 0, TAAnOL1 = 0)

Note 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 factor 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)

Note 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.

11.6.7 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 TAAnOPT0.TAAnCCS[1:0] bits.

The settings of the TAAnOPT0.TAAnCCS[1:0] bits 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 upon a match between the 16-bit counter and the CCR1 buffer register in the free-running mode.

  Rewrite during compare timer operation is enabled and performed with any time write mode. (Once the compare value has been written, synchronization with the internal clock is done and this value is used as the 16-bit counter comparison value.)

  When timer output (TOAAn1) has been enabled, TOAAn1 performs toggle output upon a match between the 16-bit counter and 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

  An interrupt is output upon a match between the 16-bit counter and the CCR0 buffer register in the free-running mode.

  Rewrite during compare timer operation is enabled and performed with any time write mode. (Once the compare value has been written, synchronization with the internal clock is done 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 TAAnCTL1.TAAnEEE bit 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.

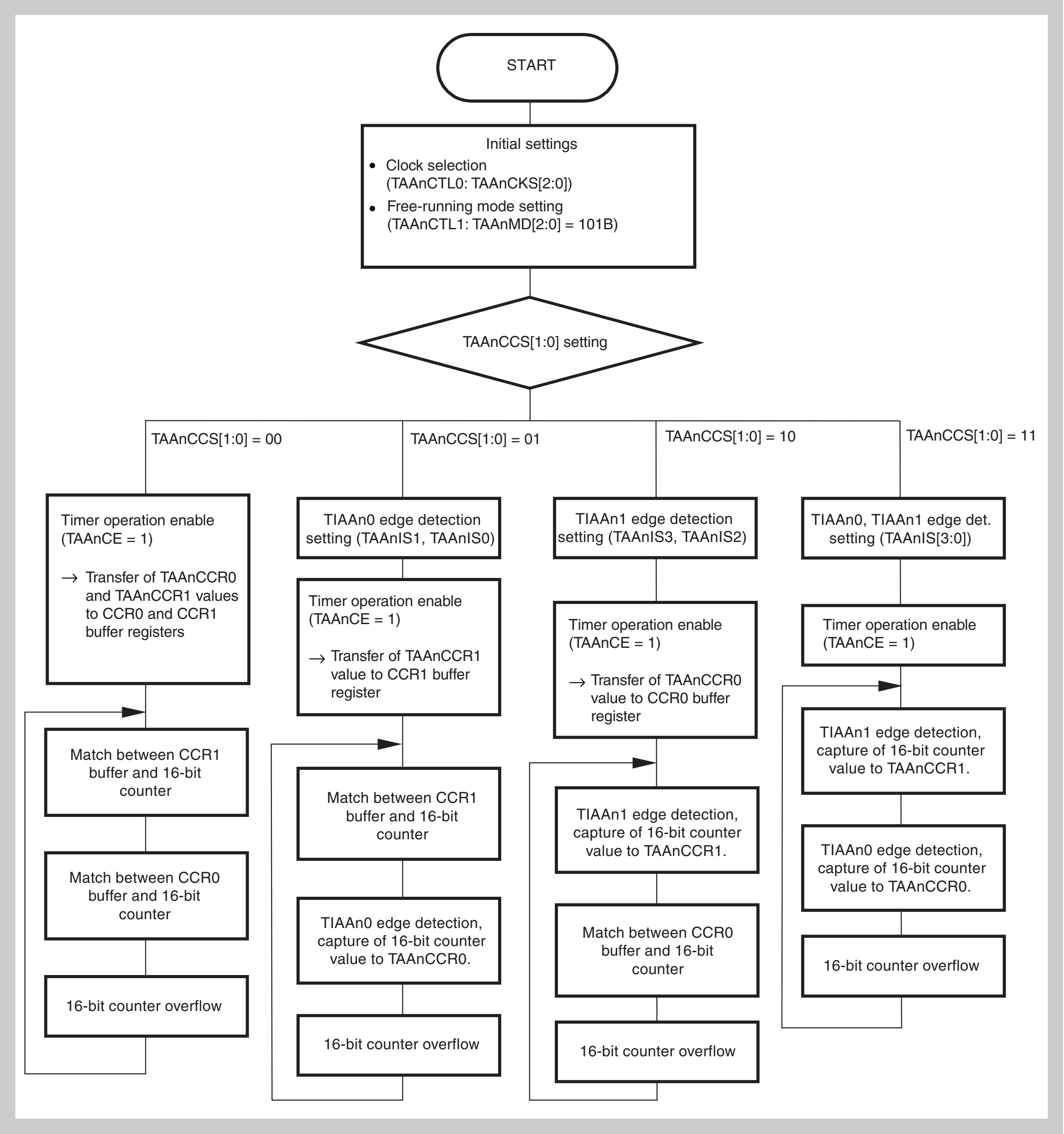

START
Initial settings
• Clock selection (TAAnCTL0: TAAnCKS[2:0])
• Free-running mode setting (TAAnCTL1: TAAnMD[2:0] = 101B)
TAAnCCS[1:0] setting
TAAnCCS[1:0] = 00
TAAnCCS[1:0] = 01
TAAnCCS[1:0] = 10
TAAnCCS[1:0] = 11
Timer operation enable (TAAnCE = 1) → Transfer of TAAnCCR0 and TAAnCCR1 values to CCR0 and CCR1 buffer registers
Match between CCR1 buffer and 16-bit counter
Match between CCR0 buffer and 16-bit counter
16-bit counter overflow
TIAAn0 edge detection setting (TAAnIS1, TAAnIS0)
Timer operation enable (TAAnCE = 1) → Transfer of TAAnCCR1 value to CCR1 buffer register
Match between CCR1 buffer and 16-bit counter
TIAAn0 edge detection, capture of 16-bit counter value to TAAnCCR0.
16-bit counter overflow
TIAAn1 edge detection setting (TAAnIS3, TAAnIS2)
Timer operation enable (TAAnCE = 1) → Transfer of TAAnCCR0 value to CCR0 buffer register
TIAAn1 edge detection, capture of 16-bit counter value to TAAnCCR1.
Match between CCR0 buffer and 16-bit counter
16-bit counter overflow
TIAAn0, TIAAn1 edge det. setting (TAAnIS[3:0])
Timer operation enable (TAAnCE = 1)
TIAAn1 edge detection, capture of 16-bit counter value to TAAnCCR1.
TIAAn0 edge detection, capture of 16-bit counter value to TAAnCCR0.
16-bit counter overflow

Figure 11-25 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.

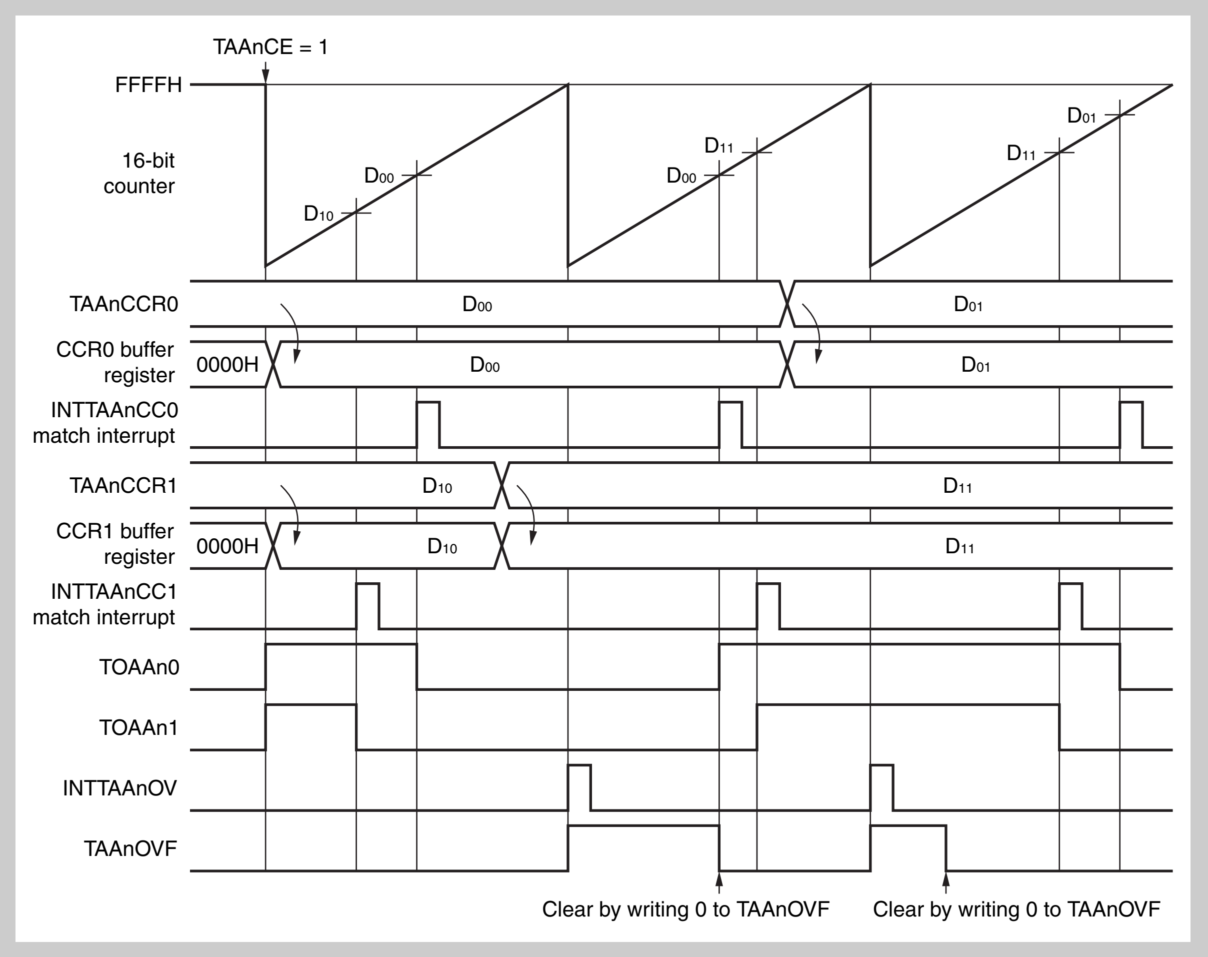


Figure 11-26 Basic Operation Timing in Free-Running Mode (TAAnCCS1 = 0, TAAnCCS0 = 0) (TAAnOE0 = 1, TAAnOE1 = 1, TAAnOL0 = 0, TAAnOL1 = 0)

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.

Note m = 0, 1

(2) When TAAnCCS1 = 1 and TAAnCCS0 = 1 settings (capture function description)

When TAAnCE = 1, the 16-bit counter counts from 0000H to FFFFH 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 (FFFFH), 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.

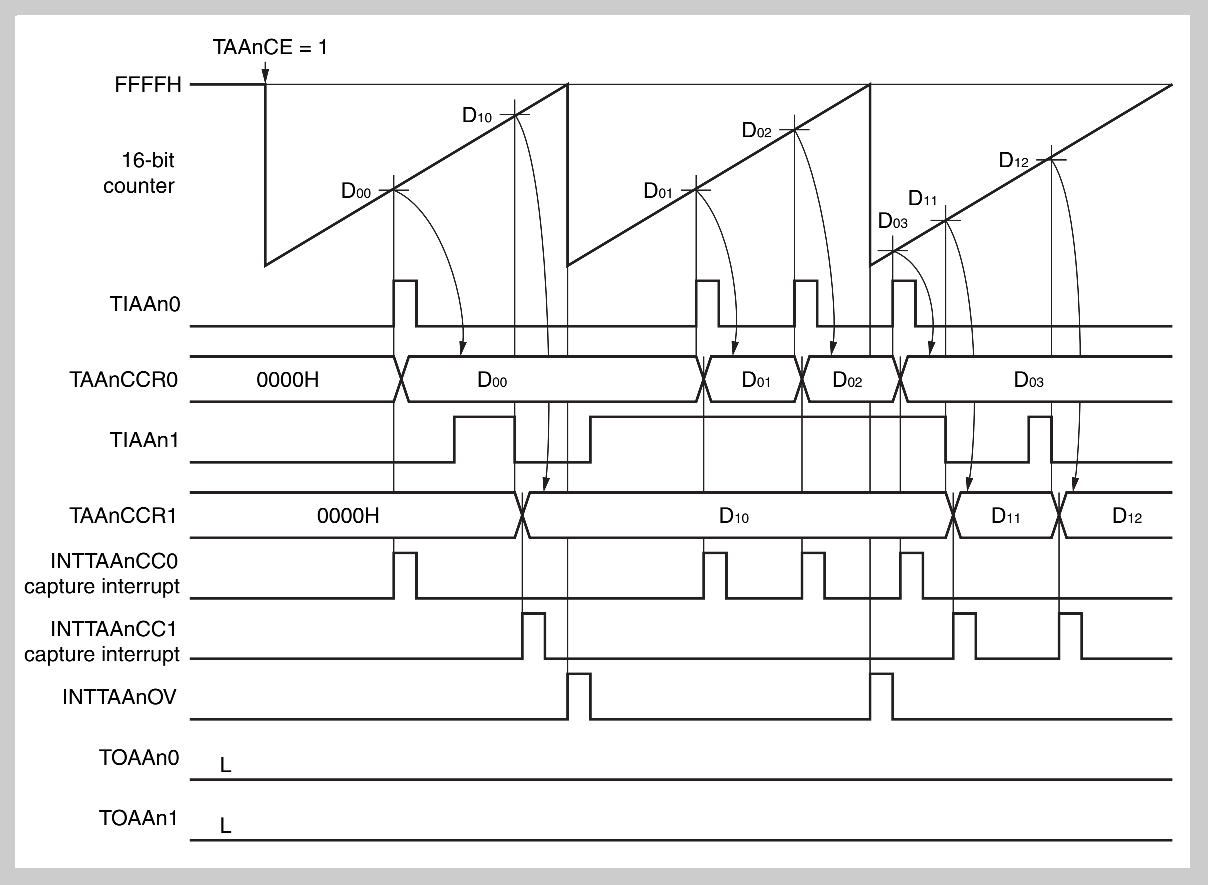


Figure 11-27 Basic Operation Timing in Free-Running Mode (TAAnCCS1 = 1, TAAnCCS0 = 1) (TAAnOE0 = 1, TAAnOE1 = 1, TAAnOL0 = 0, TAAnOL1 = 0)

D00, D01, D02, D03:
Values captured to TAAnCCR0 register (0000$_H$ to FFFF$_H$)
D10, D11, D12:
Values captured to TAAnCCR1 register (00000$_H$ to FFFF$_H$)

TIAAn0: Set to rising edge detection (TAAnIS[1:0] = 01$_B$)
TIAAn1: Set to falling edge detection (TAAnIS[3:2] = 10$_B$)

(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.

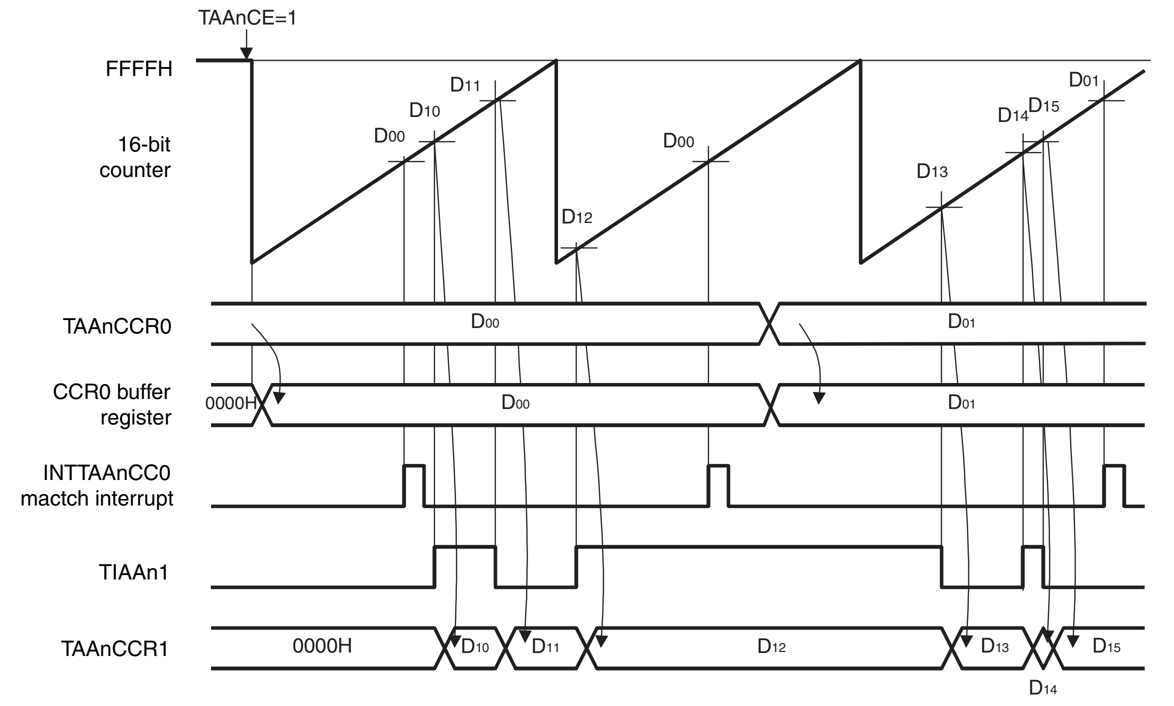


Figure 11-28 Basic Operation Timing in Free-Running Mode (TAAnCCS1 = 1, TAAnCCS0 = 0) (TAAnOE0 = 1, TAAnOE1 = 1, TAAnOL0 = 0, TAAnOL1 = 0)

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 (TAAnIS[3:2] = 11_B)

(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.

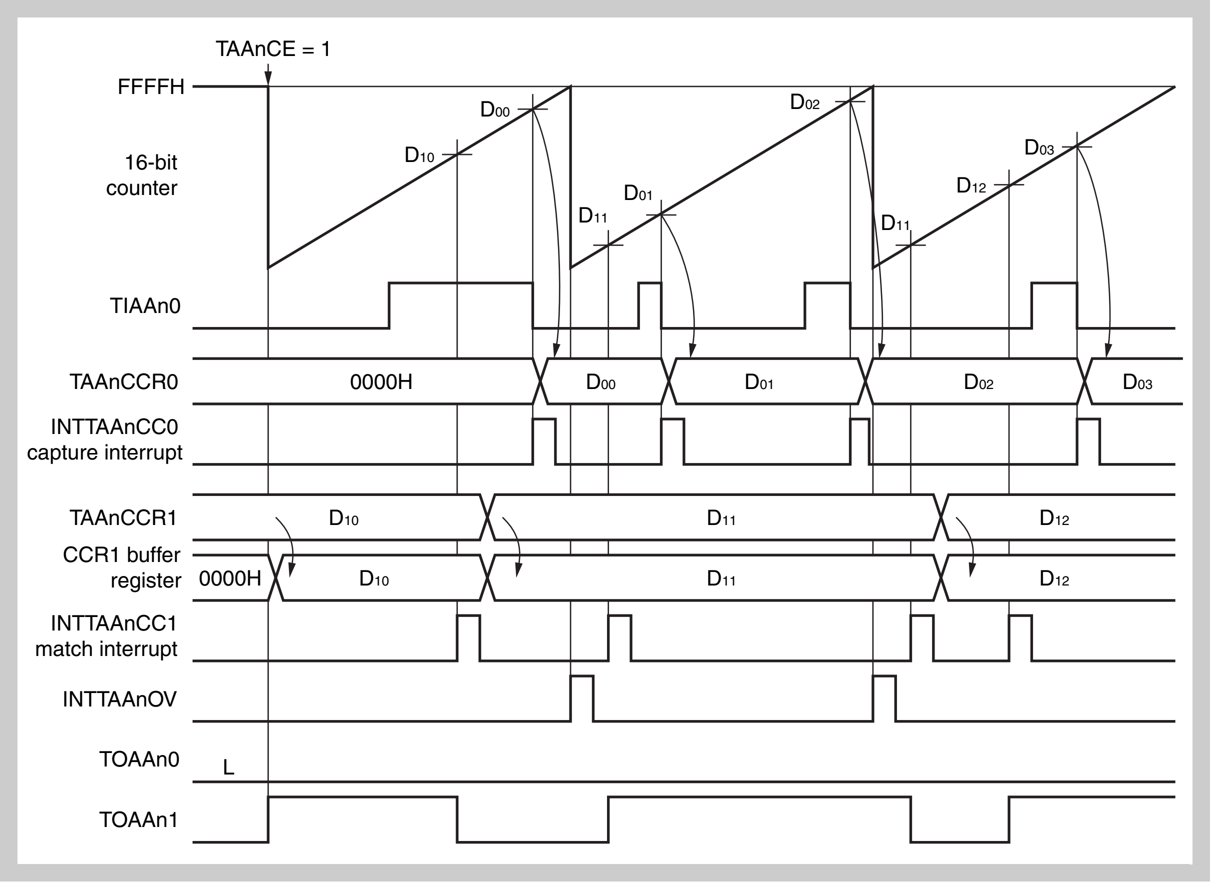


Figure 11-29 Basic Operation Timing in Free-Running Mode (TAAnCCS1 = 0, TAAnCCS0 = 1) (TAAnOE0 = 1, TAAnOE1 = 1, TAAnOL0 = 0, TAAnOL1 = 0)

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 (TAAnIS[1:0] = 10_B)

(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.

11.6.8 Pulse width measurement mode (TAAnMD2 to TAAnMD0 = 110B)

In the pulse width measurement mode, 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, and 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, e.g by counting the overflow count by using the overflow interrupt.

Depending on the selected capture input sources and specified edge detection three different measurement methods can be applied.

1. Pulse period measurement
2. Alternating pulse width and pulse space measurement.
3. 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 (TAAnCTL1.TAAnEEE = 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.

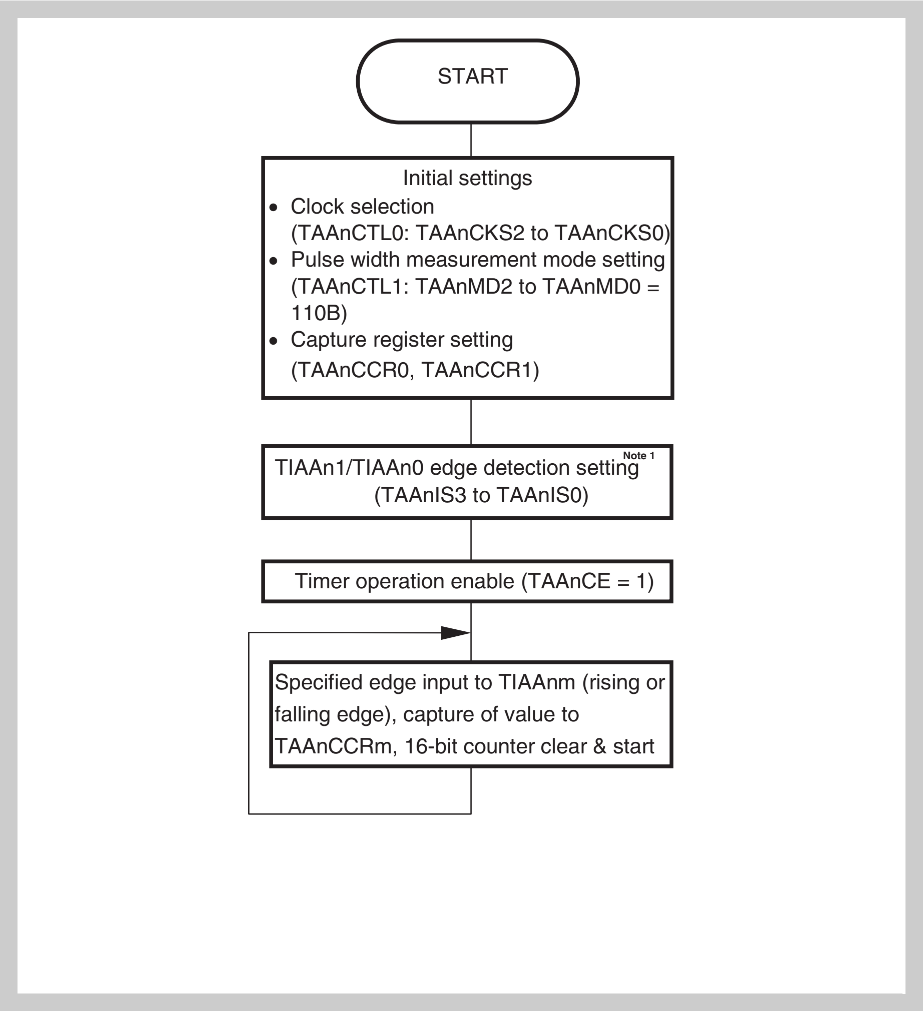

START
Initial settings
• Clock selection
(TAAnCTL0: TAAnCKS2 to TAAnCKS0)
• Pulse width measurement mode setting
(TAAnCTL1: TAAnMD2 to TAAnMD0 = 110B)
• Capture register setting
(TAAnCCR0, TAAnCCR1)
TIAAn1/TIAAn0 edge detection setting Note 1
(TAAnIS3 to TAAnIS0)
Timer operation enable (TAAnCE = 1)
Specified edge input to TIAAnm (rising or falling edge), capture of value to TAAnCCRm, 16-bit counter clear & start

Figure 11-30 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”.
2. m = 0, 1

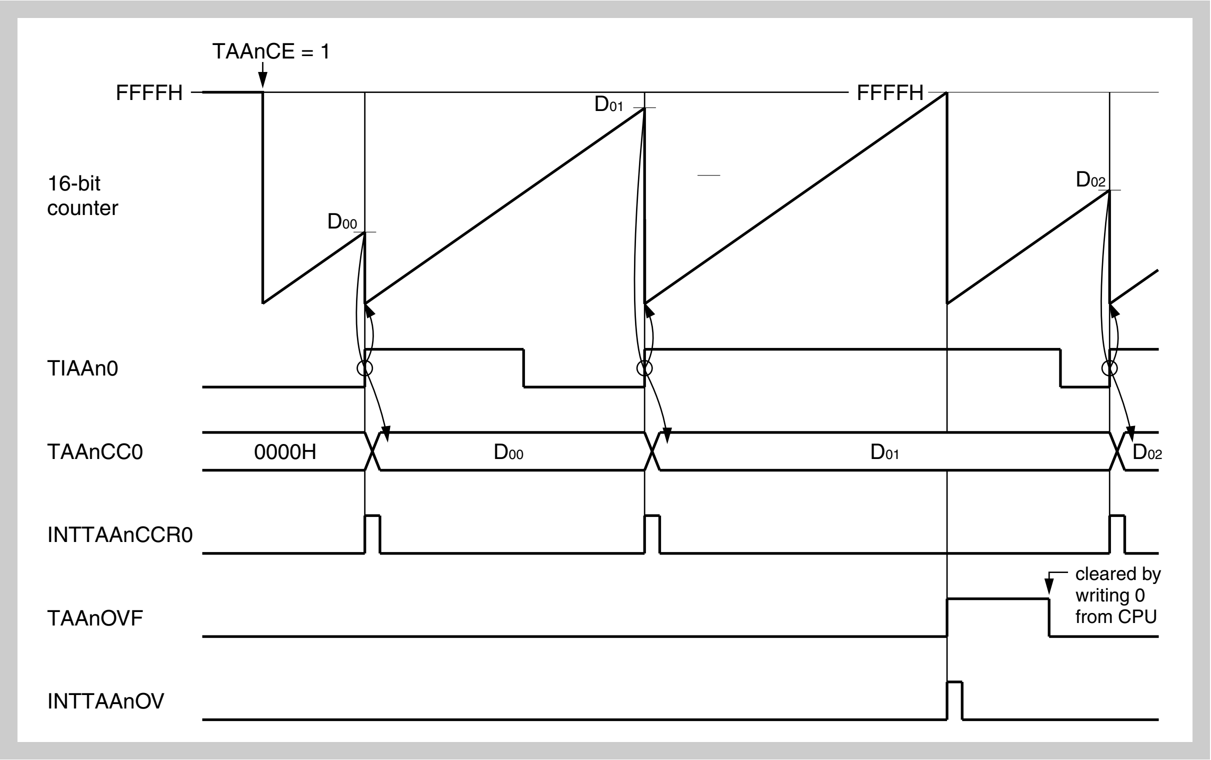


Figure 11-31 Basic Operation Timing of Pulse Period Measurement (TAAnOE0 = 0, TAAnOE1 = 0, TAAnOL0 = 0, TAAnOL1 = 0)

D_{00}, D_{01}, D_{02}: Values captured to TAAnCCR0 register (0000_H to $FFFF_H$)

TIAAn0: Set to detection of rising edge (TAAnIS[1:0] = 01_B)
TIAAn1: Set to no edge detection (TAAnIS[3:2] = 00_B)

(2) Alternating pulse width and pulse space measurement

The pulse width and space 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.

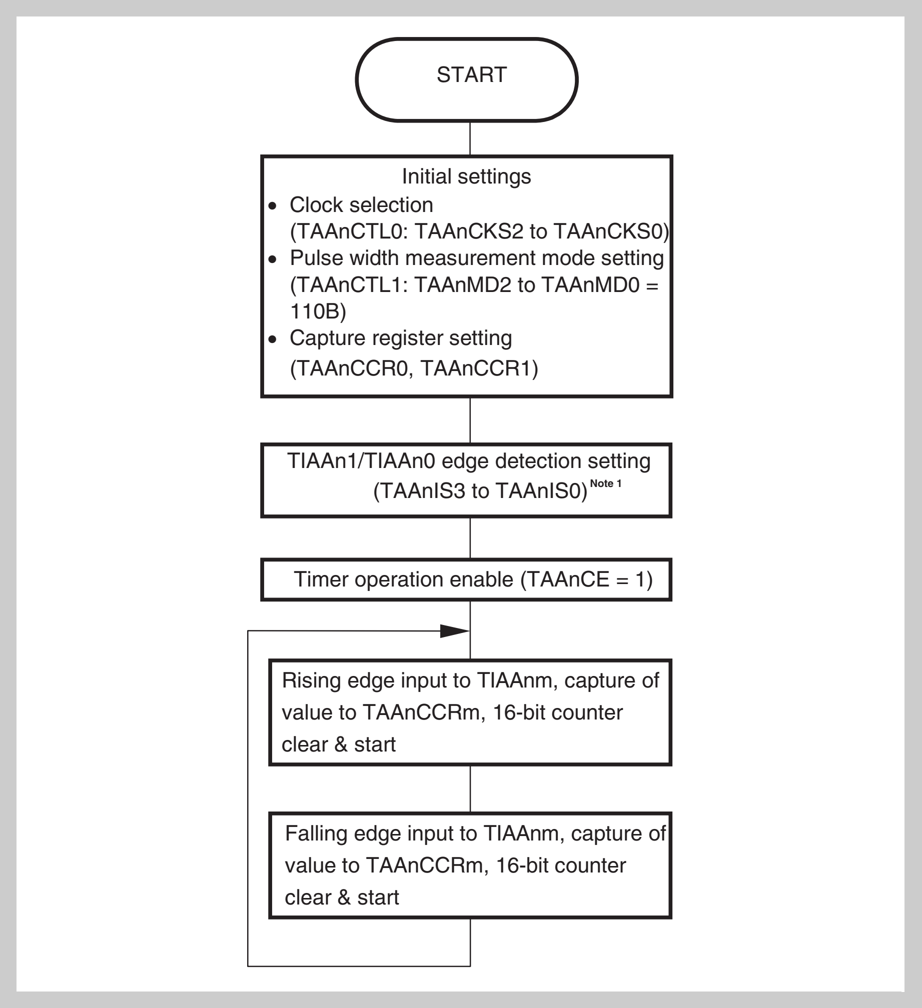

START
Initial settings
• Clock selection
(TAAnCTL0: TAAnCKS2 to TAAnCKS0)
• Pulse width measurement mode setting
(TAAnCTL1: TAAnMD2 to TAAnMD0 = 110B)
• Capture register setting
(TAAnCCR0, TAAnCCR1)
TIAAn1/TIAAn0 edge detection setting
(TAAnIS3 to TAAnIS0) Note 1
Timer operation enable (TAAnCE = 1)
Rising edge input to TIAAnm, capture of value to TAAnCCRm, 16-bit counter clear & start
Falling edge input to TIAAnm, capture of value to TAAnCCRm, 16-bit counter clear & start

Figure 11-32 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”.
2. m = 0, 1

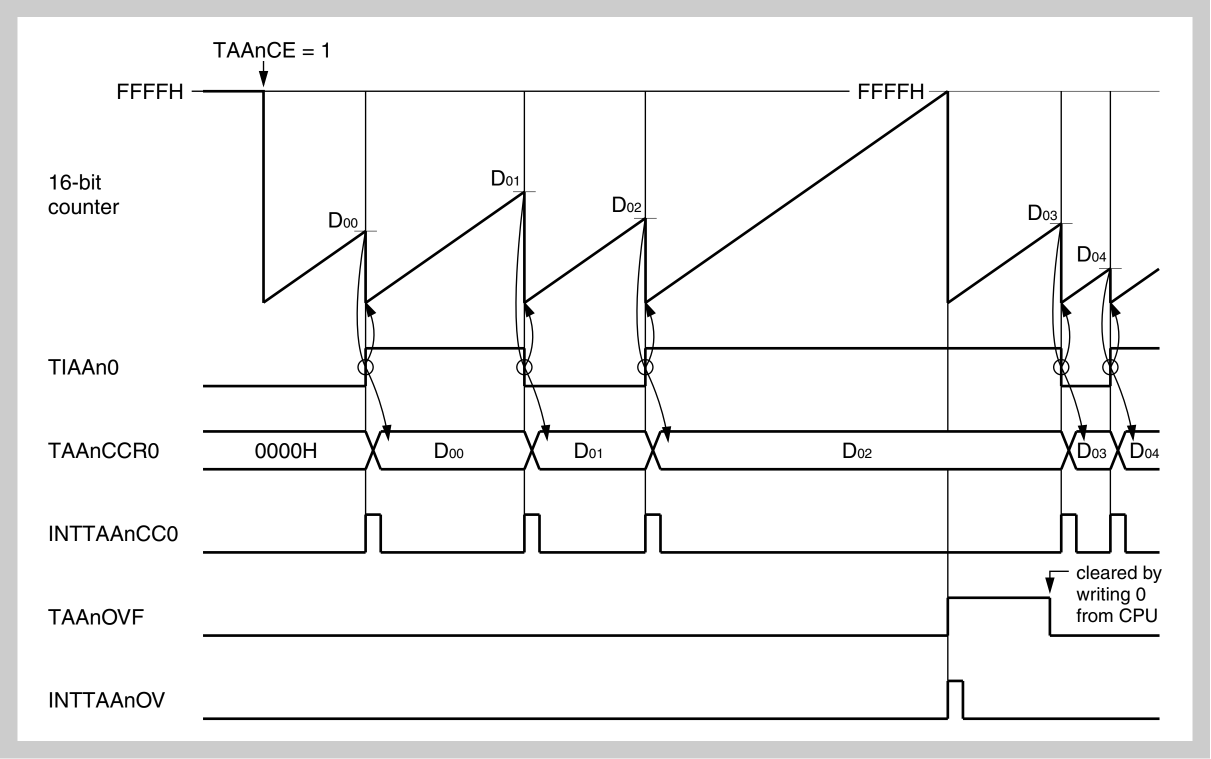


Figure 11-33 Basic Operation Timing of Alternating Pulse Width and Pulse Space Measurement (TAAnOE0 = 0, TAAnOE1 = 0, TAAnOL0 = 0, TAAnOL1 = 0)

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 (TAAnIS[1:0] = 11_B)
TIAAn1: Set to no edge detection (TAAnIS[3:2] = 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 (NTTAAnOV) is generated at the next count clock, and the counter is cleared to 0000H 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.

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.

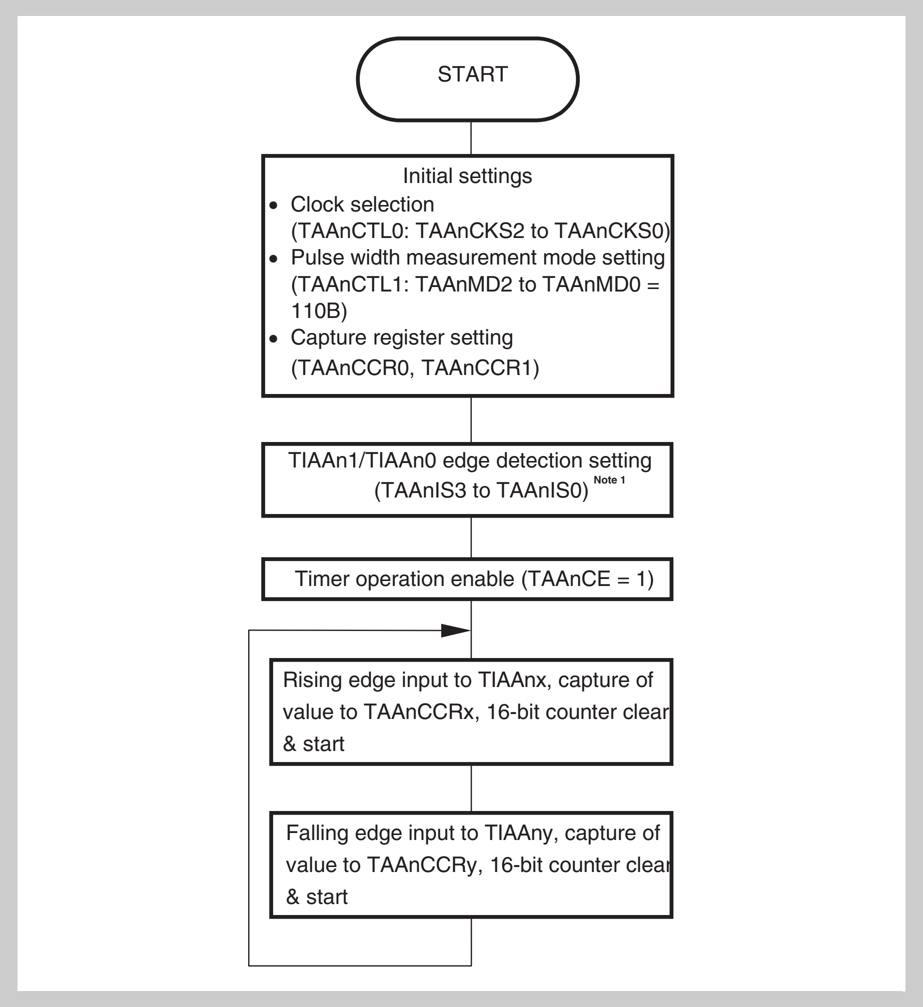

START
Initial settings
• Clock selection
(TAAnCTL0: TAAnCKS2 to TAAnCKS0)
• Pulse width measurement mode setting
(TAAnCTL1: TAAnMD2 to TAAnMD0 = 110B)
• Capture register setting
(TAAnCCR0, TAAnCCR1)
TIAAn1/TIAAn0 edge detection setting
(TAAnIS3 to TAAnIS0) Note 1
Timer operation enable (TAAnCE = 1)
Rising edge input to TIAAnx, capture of value to TAAnCCRx, 16-bit counter clear & start
Falling edge input to TIAAny, capture of value to TAAnCCRy, 16-bit counter clear & start

Figure 11-34 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

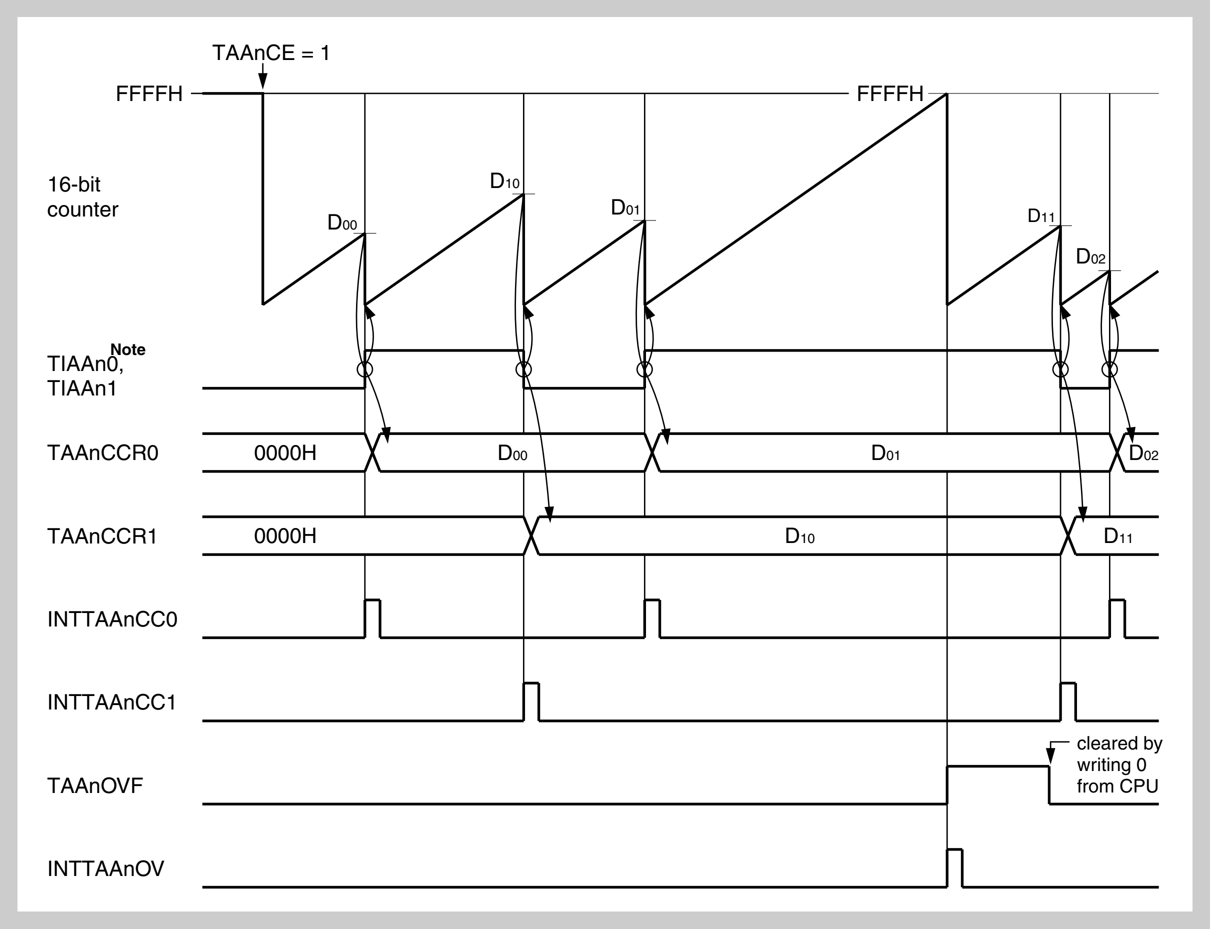


Figure 11-35 Basic Operation Timing of Simultaneous Pulse Width and Pulse Space Measurement (TAAnOE0 = 0, TAAnOE1 = 0, TAAnOL0 = 0, TAAnOL1 = 0)

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 (TAAnIS[1:0] = 01_B)
TIAAn1: Set detection to falling edge (TAAnIS[3:2] = 10_B)

11.6.9 32-bit Capture in Free-Running Cascade Mode

Two Timer AA (TAA0 in combination with TAA1, or TAA2 in combination with TAA3) can be cascaded to operate as a 32-bit capture timer. In cascade mode, the timer with the lower number (TAA0 or TAA2) is used to control the operation (master timer). Both cascaded timers have to be initialized as free-running timers.

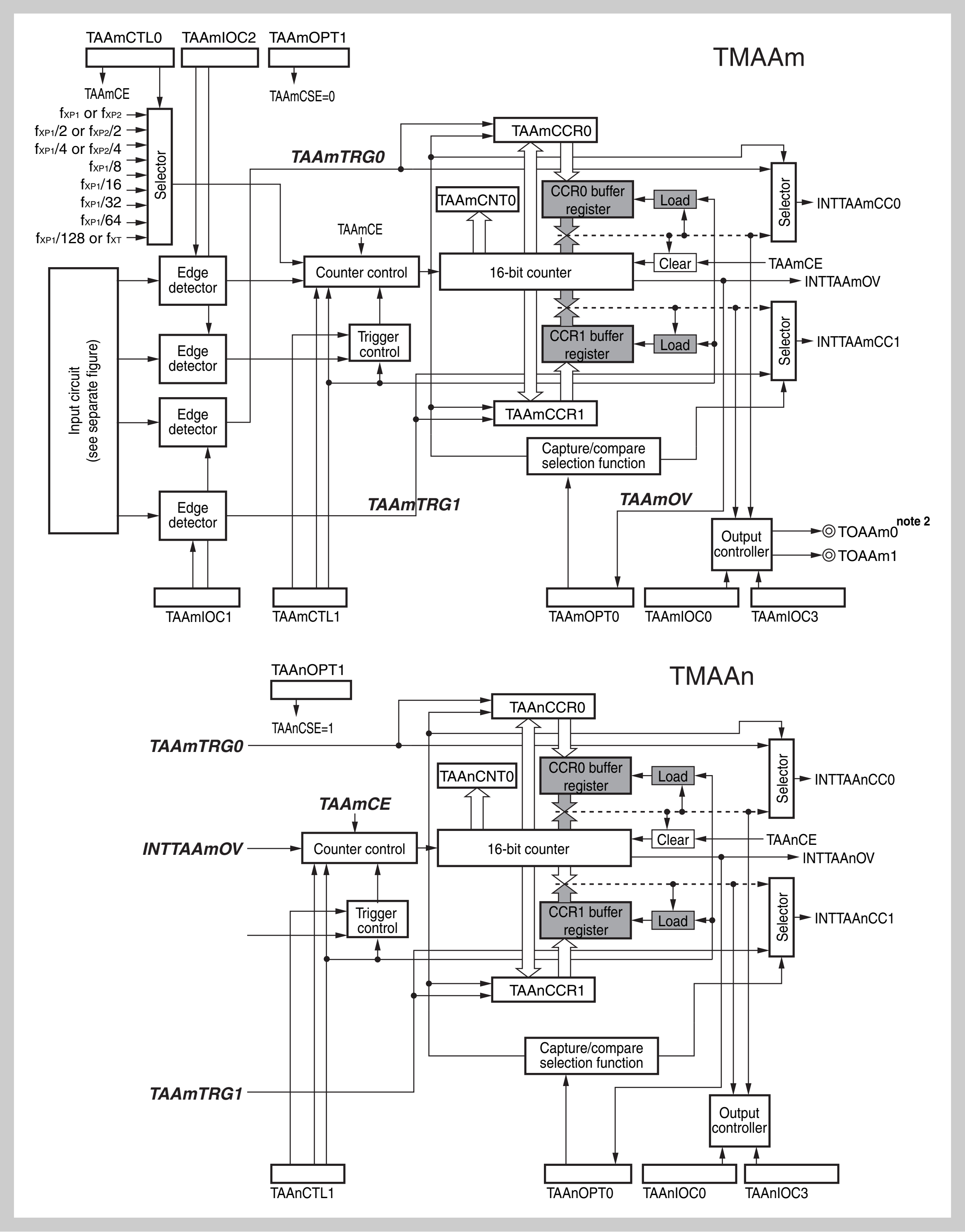


Figure 11-36 Block Diagram of TAAm and TAAn in 32-bit Capture Mode

Note 1. m = 0, 2
n = 1, 3

2. The 32-bit capture in cascade free-running mode is not available for TAA4.

3. Explanation of signals can be found in *Figure 11-1 on page 402*.

4. Block diagrams of the input circuits can be found in *Figure 11-2 on page 403* and *Figure 11-3 on page 403*.

Figure 11-36 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 11-1 on page 402* can be used for a in-depth look of each timer.

Note Cascading of two TAA 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 11-37 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 11-38 on page 465 shows an example of a 32-bit capture timing.

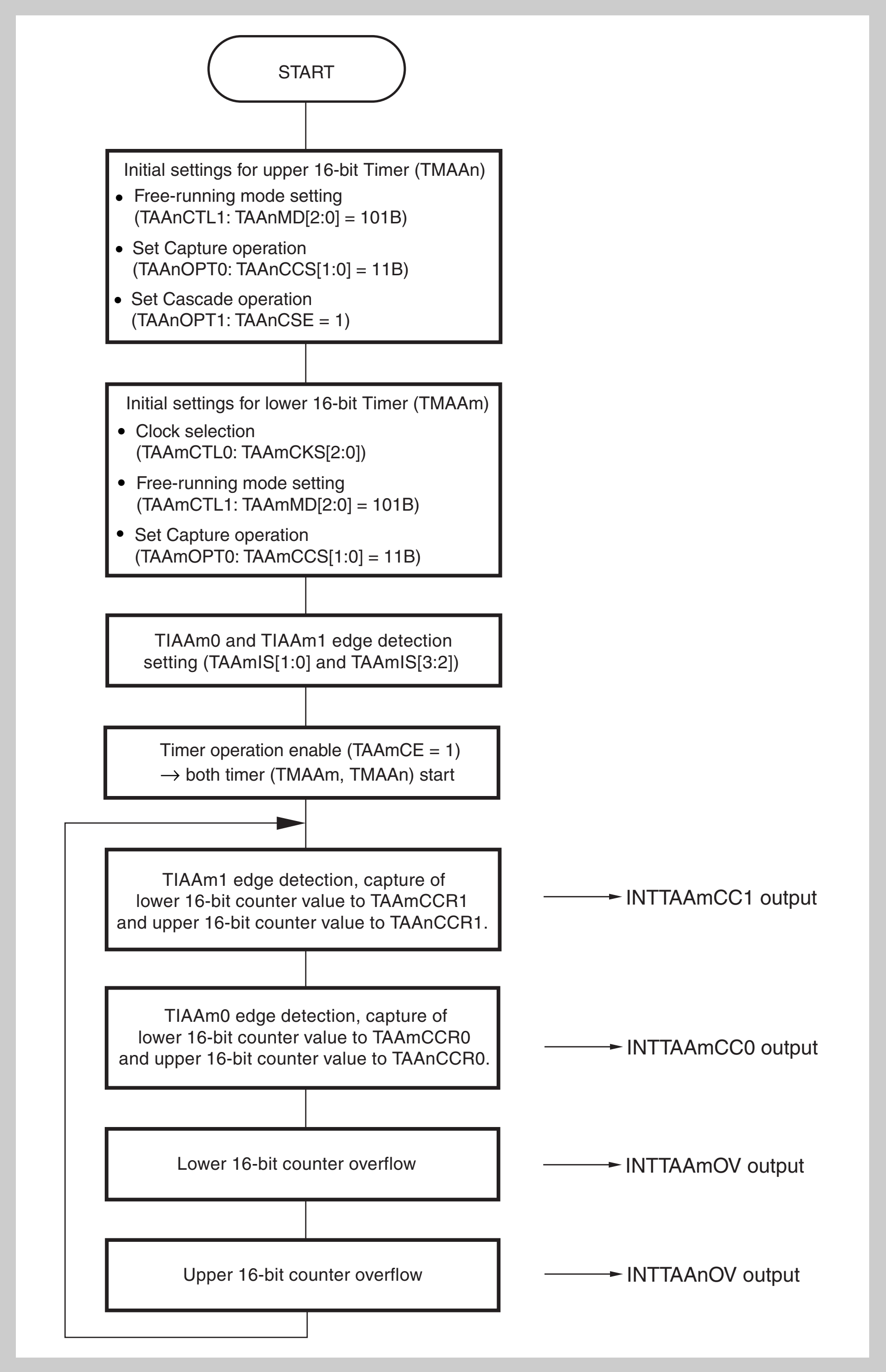

START
Initial settings for upper 16-bit Timer (TMAAn)
• Free-running mode setting
(TAAnCTL1: TAAnMD[2:0] = 101B)
• Set Capture operation
(TAAnOPT0: TAAnCCS[1:0] = 11B)
• Set Cascade operation
(TAAnOPT1: TAAnCSE = 1)
Initial settings for lower 16-bit Timer (TMAAm)
• Clock selection
(TAAmCTL0: TAAmCKS[2:0])
• Free-running mode setting
(TAAmCTL1: TAAmMD[2:0] = 101B)
• Set Capture operation
(TAAmOPT0: TAAmCCS[1:0] = 11B)
TIAAm0 and TIAAm1 edge detection setting (TAAmIS[1:0] and TAAmIS[3:2])
Timer operation enable (TAAmCE = 1)
→ both timer (TMAAm, TMAAn) start
TIAAm1 edge detection, capture of lower 16-bit counter value to TAAmCCR1 and upper 16-bit counter value to TAAnCCR1.
INTTAAmCC1 output
TIAAm0 edge detection, capture of lower 16-bit counter value to TAAmCCR0 and upper 16-bit counter value to TAAnCCR0.
INTTAAmCC0 output
Lower 16-bit counter overflow
INTTAAmOV output
Upper 16-bit counter overflow
INTTAAnOV output

Figure 11-37 Basic Flow of 32-bit Capture Mode

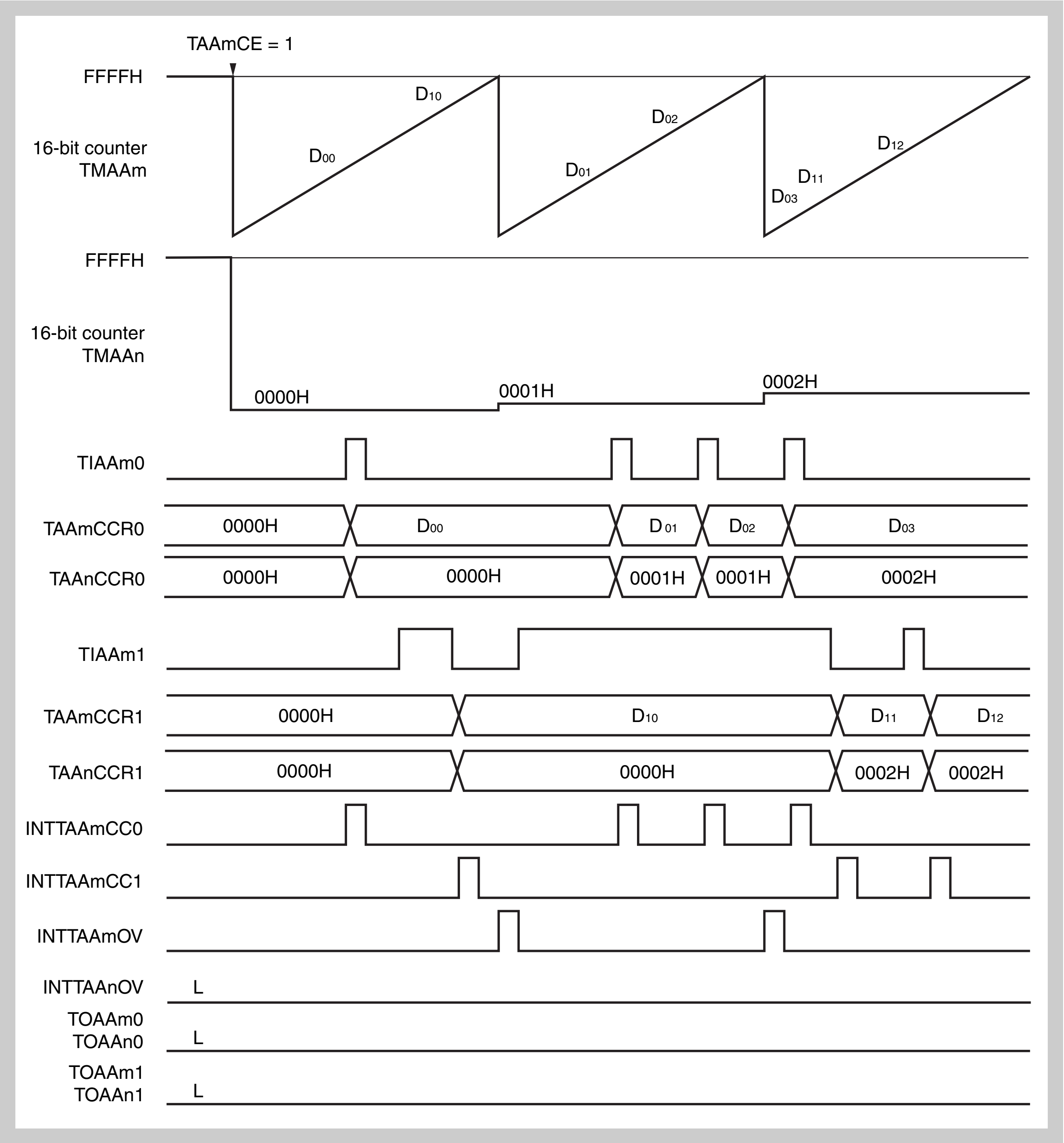


Figure 11-38 Basic Timing of 32-bit Capture Mode

Note m = 0, 2
n = 1,3

As the 32-bit resolution is achieved by cascading two individual TAA, 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 in *Figure 11-39* needs to be followed for reading.

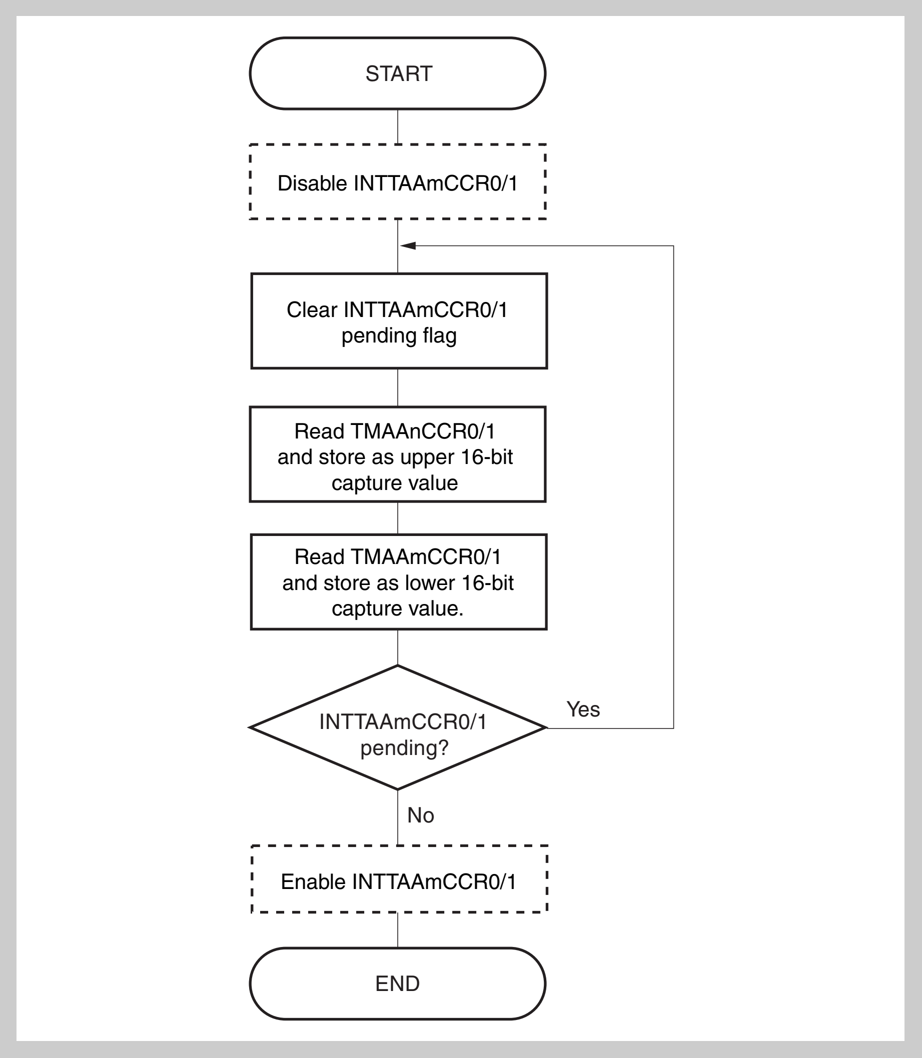

START
Disable INTTAAmCCR0/1
Clear INTTAAmCCR0/1 pending flag
Read TMAAnCCR0/1 and store as upper 16-bit capture value
Read TMAAmCCR0/1 and store as lower 16-bit capture value.
INTTAAmCCR0/1 pending?
Yes
No
Enable INTTAAmCCR0/1
END

Figure 11-39 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 which pending flags 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.

11.6.10 Capture operation on delayed input clock

If during capture operation the first capture event triggers before the first edge of the count clock occurs a value of $\mathrm{FFFF_H}$ and not a value of $\mathrm{0000_H}$ may be stored in the TAAnCCRm registers.

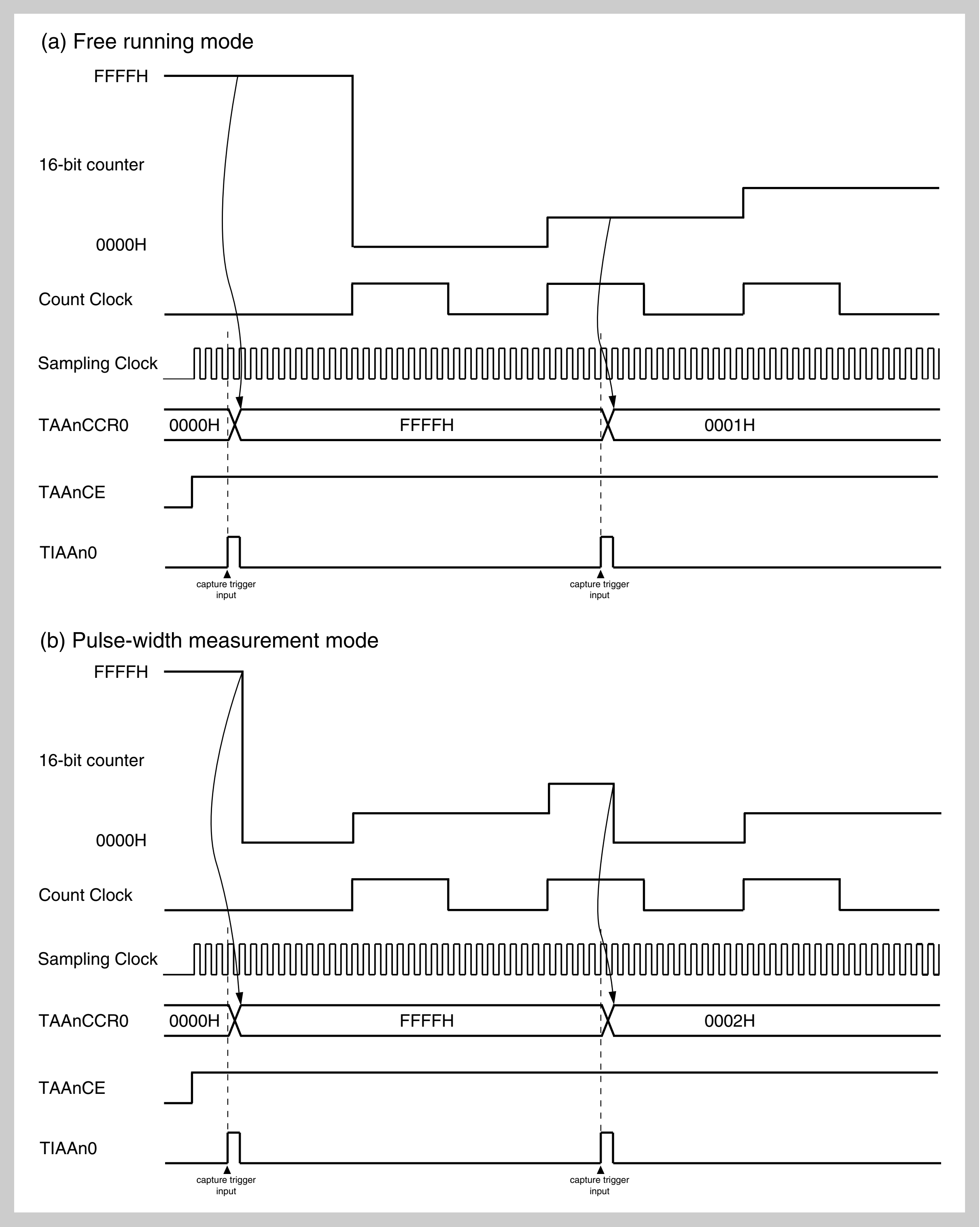


Figure 11-40 Capture operation on delayed input clock

Chapter 12 16-Bit Timer/Event Counter AB

The V850ES/Fx3 microcontroller have following instances of the 16-bit timer/ event counter AB:

TAB	V850ES/ FE3	V850ES/ FF3	V850ES/ FG3	V850ES/ FJ3	V850ES/ FK3
Instances	1		2	3	
Names	TAB0		TAB0 to TAB1	TAB0 to TAB2	

Throughout this chapter, the individual instances of Timer AB are identified by “n”, for example, TABnCTL0 for the TABn control register 0.

12.1 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 synchronised operation function with Timers AA and Timers AB channels (refer to *“Timer AA/AB Synchronous Operation” on page 525*)

12.2 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

12.3 Configuration

TAB includes the following hardware.

Table 12-1 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 buffers registers
Timer input	4 (TIABn0^{Note} to TIABn3)
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.

Timer AB (TAB) pins are alternate function of port pins. For how to set the alternate function, refer to the description of the registers in *“Pin Functions” on page 32*.

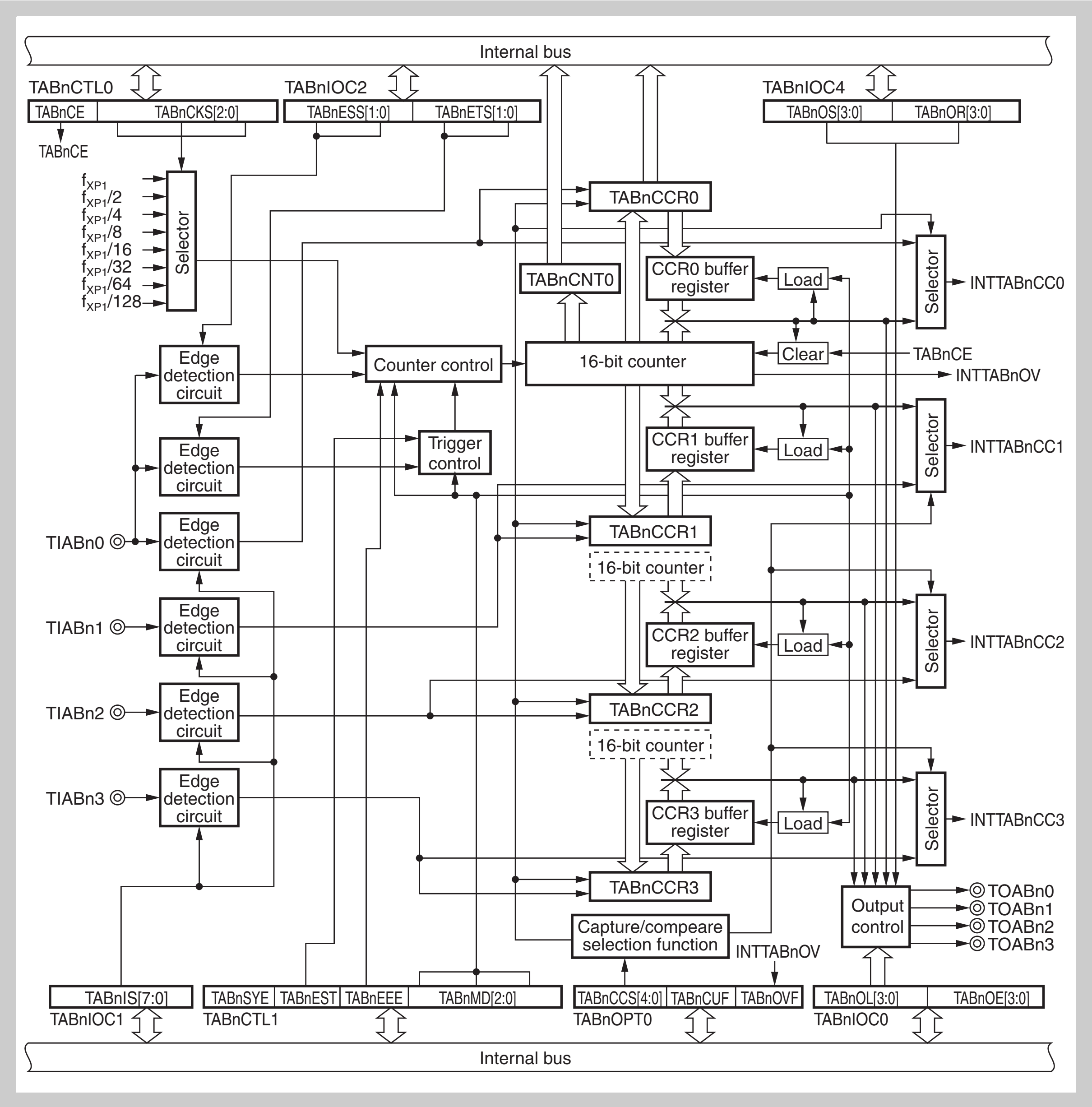


Figure 12-1 Block diagram of Timer AB

(1) 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.

In the default status, the TABnCCR0 register functions as a compare register.

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

Address TAB0CCR0: FFFFF546$_H$ TAB1CCR0: FFFFF616$_H$
TAB2CCR0: FFFFF626$_H$

Initial Value 0000$_H$. This registers is cleared by any reset, or if the internal operation clock is disabled by TABnCTL0.TABnCE = 0.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TABnCCR0	Capture/compare value 0															
	R/W															

Use as compare register When used as a compare register TABnCCR0 can be rewritten when TABnCE = 1, as below mentioned:

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	Not applicable (used as capture register)

Use as capture register When used as capture register the count value is stored in TABnCCR0 upon capture trigger (TIABn0) input edge detection.

Note

1. The value of TABnCCR0 register can be read/written when TABnCTL0.TABnCE = 1.
2. Access to the TABnCCR0 register is prohibited when the main clock is stopped in the subclock mode.

(2) 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 TABnOPT0.TABnCCS1 bit, 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.

In the default status, the TABnCCR1 register functions as a reload register.

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

Address TAB0CCR1: FFFFF548_H TAB1CCR1: FFFFF618_H
TAB2CCR1: FFFFF628_H

Initial Value 0000_H. This register is cleared by any reset, or if the internal operation clock is disabled by TABnCTL0.TABnCE = 0.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TABnCCR1	Capture/compare value 1															
	R/W															

Caution In the one-shot pulse mode, it is prohibited to set the TABnCCR1 register to 0000_H.

Use as compare register When used as a compare register TABnCCR1 can be rewritten when TABnCE = 1, as below mentioned:

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	Not applicable (used as capture register)

Use as capture register When used as a capture register the count value is stored in TABnCCR1 upon capture trigger (TIABn1) input edge detection.

Note

1. The value of TABnCCR1 register can be read/written when TABnCE bit of TABn control register 0 (TABnCTL0) equals 1.
2. Access to the TABnCCR1 register is prohibited when the main clock is stopped in the subclock mode.

(3) 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 TABnOPT0.TABnCCS2 bit , 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.

In the default status, the TABnCCR2 register functions as a compare register.

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

Address TAB0CCR2: FFFFF54A$_H$ TAB1CCR2: FFFFF61A$_H$
TAB2CCR2: FFFFF62A$_H$

Initial Value 0000$_H$. This register is cleared by any reset, or if the internal operation clock is disabled by TABnCTL0.TABnCE = 0.

<table>
<tr><th></th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td>TABnCCR2</td><td colspan="16">Capture/compare value 2</td></tr>
<tr><td></td><td colspan="16">R/W</td></tr>
</table>

Use as compare register When used as a compare register TABnCCR2 can be rewritten when TABnCE = 1, as below mentioned:

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	Not applicable (used as capture register)

Use as capture register When used as capture register the count value is stored in TABnCCR2 upon capture trigger (TIABn2) input edge detection.

Note

1. The value of TABnCCR2 register can be read/written when TABnCTL0.TABnCE bit = 1.
2. Access to the TABnCCR2 register is prohibited when the main clock is stopped in the subclock mode.

(4) 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 TABnOPT0.TABnCCS3 bit, 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.

In the default status, the TABnCCR3 register functions as a compare register.

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

Address TAB0CCR3: FFFFF54C$_H$ TAB1CCR3: FFFFF61C$_H$
TAB2CCR3: FFFFF62C$_H$

Initial Value 0000$_H$. This register is cleared by any reset, or if the internal operation clock is disabled by TABnCTL0.TABnCE = 0.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TABnCCR3	Capture/compare value 3															
	R/W															

Use as compare register When used as a compare register TABnCCR3 can be rewritten when TABnCE = 1, as below mentioned:

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	Not applicable (used as capture register)

Use as capture register When used as capture register the count value is stored in TABnCCR3 upon capture trigger (TIABn3) input edge detection.

Note

1. The value of TABnCCR3 register can be read/written when TABnCTL0.TABnCE = 1.
2. Access to the TABnCCR3 register is prohibited when the main clock is stopped in the subclock mode.

(5) 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 only in 16-bit units.

Address TAB0CNT: FFFFF54E_H TAB1CNT: FFFFF61E_H
TAB2CNT: FFFFF62E_H

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

<table>
<tr><th></th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td>TABnCNT</td><td colspan="16">Counter value</td></tr>
<tr><td></td><td colspan="16">R</td></tr>
</table>

Note 0000_H is read from this register, when TABnCTL0.TABnCE = 0. The current counter value is read when TABnCE bit = 1.

12.4 Control Registers

(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 TAB0CTL0: $FFFFF540_H$ TAB1CTL0: $FFFFF610_H$
TAB2CTL0: $FFFFF620_H$

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

	7	6	5	4	3	2	1	0
TABnCTL0	TABnCE	0	0	0	0	TABnCKS2	TABnCKS1	TABnCKS0
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 12-2 TABnCTL0 register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>7</td><td>TABnCE</td><td>Controls the timer TABn operation.
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.
Note: If the timer is operated in synchronous operation mode, i.e. TABnCTL1.TABnSYE = 1, TABnCTL0.TABnCE cannot be set to “1”.</td></tr>
<tr><td>2 to 0</td><td>TABnCKS [2:0]</td><td>**Caution:** Selects the count clock of timer TABn.
<table>
<tr><th rowspan="2">TABnCKS2</th><th rowspan="2">TABnCKS1</th><th rowspan="2">TABnCKS0</th><th rowspan="2">Input</th><th colspan="2">Selection of internal count clock</th></tr>
<tr><th>PRSI = 0</th><th>PRSI = 1</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>f_{XP1}</td><td>f_{XX}</td><td>$f_{XX}/2$</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>$f_{XP1}/2$</td><td>$f_{XX}/2$</td><td>$f_{XX}/4$</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>$f_{XP1}/4$</td><td>$f_{XX}/4$</td><td>$f_{XX}/8$</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>$f_{XP1}/8$</td><td>$f_{XX}/8$</td><td>$f_{XX}/16$</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>$f_{XP1}/16$</td><td>$f_{XX}/16$</td><td>$f_{XX}/32$</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>$f_{XP1}/32$</td><td>$f_{XX}/32$</td><td>$f_{XX}/64$</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>$f_{XP1}/64$</td><td>$f_{XX}/64$</td><td>$f_{XX}/128$</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>$f_{XP1}/128$</td><td>$f_{XX}/128$</td><td>$f_{XX}/256$</td></tr>
</table>
Caution: Writing of TABnCKS[2:0] bits is prohibited during operation (TABnCE = 1). When the value of the TABnCE bit is changed from 0 to 1, bits TABnCKS[2:0] can be set simultaneously.

Note: PRSI can be set by the option bytes:
• PRSI = 0: $f_{XX} \leqq 32$ MHz
• PRSI = 1: 32 MHz $< f_{XX} \leqq 48$ MHz
Refer to *“Flash Memory” on page 298* for details.</td></tr>
</table>

(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 TAB0CTL1: FFFFF541$_H$ TAB1CTL1: FFFFF611$_H$
TAB2CTL1: FFFFF621$_H$

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

TABnCTL1 (n =0, 1)

7	6	5	4	3	2	1	0
0	TABnEST	TABnEEE	0	0	TABnMD2	TABnMD1	TABnMD0
R/W	R/W	R/W	R	R	R/W	R/W	R/W

TAB2CTL1

7	6	5	4	3	2	1	0
TAB2SYE	TAB2EST	TAB2EEE	0	0	TAB2MD2	TAB2MD1	TAB2MD0
R/W	R/W	R/W	R	R	R/W	R/W	R/W

Caution Set bits TAB2SYE, TABnEEE and TABnMD[2:0] 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.

Table 12-3 TABnCTL1 register contents (1/2)

Bit position	Bit name	Function
7	TAB2SYE	Controls the tuned operation mode of timer TAB2. 0: Independent operation mode (asynchronous operation mode 1: Synchronous operation mode (specification of slave operation) In this mode, timer TAB2 can operate in synchronization with a master timer. For the synchronous operation mode, refer to *“Timer AA/AB Synchronous Operation” on page 525* **Note:** If TAB2SYE = 1, TAB2CTL0.TAB2CE cannot be set to “1”. **Caution:** 1. Be sure to clear the TAB2SYE to 0, if TAB2 is used as the master timer. Respectively, set the TAB2SYE = 1, if TAB2 is used as slave timer. 2. 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 and PWM mode only. Setting the external event count mode, one-shot pulse mode, and pulse width measurement mode is prohibited.

Table 12-3 TABnCTL1 register contents (2/2)

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>6</td><td>TABnEST</td><td>Controls the software trigger of timer TABn.
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.
Note: The read value of the TABnEST bit is always 0.</td></tr>
<tr><td>5</td><td>TABnEEE</td><td>Selects the count clock input of timer TABn.
0: Use the internal clock (clock selected with bits TABnCKS[2:0])
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 TABnEES[1:0].</td></tr>
<tr><td>2 to 0</td><td>TABnMD [2:0]</td><td>Selects the operation mode of timer TABn.
<table>
<tr><th>TABnMD2</th><th>TABnMD1</th><th>TABnMD0</th><th>Timer mode selection</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>Interval timer mode</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>External event counter mode</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>External trigger pulse output mode</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>One-shot pulse mode</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>PWM mode</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Free-running mode</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>Pulse width measurement mode</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>Triangular wave PWM mode</td></tr>
</table>
Caution:
1. TO (timer output) cannot be used in the external event count mode.
2. The external event count input is selected in the external event count mode regardless of the TABnEEE bit value.
3. 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) .
4. To use the external event count mode, disable edge detection of the TOABn0 capture input by clearing the TABnIOC2.TABnEES[1:0] to 00_B.</td></tr>
</table>

(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 TAB0IOC0: FFFFF542$_H$ TAB1IOC0: FFFFF612$_H$
TAB2IOC0: FFFFF622$_H$

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

	7	6	5	4	3	2	1	0
TABnIOC0	TABnOL3	TABnOE3	TABnOE2	TABnOE2	TABnOL1	TABnOE1	TABnOL0	TABnOE0
	R	R	R	R	R/W	R/W	R/W	R/W

Caution

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 TABnIOC1.TABnIS[7:0] of the register to “no edge detection” (00$_B$) 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.

Table 12-4 TABnIOC0 register contents

Bit position	Bit name	Function
7, 5, 3, 1	TABnOLm (m = 0 to 3)	Specifies the TOABnm output level. 0: Normal output (inactive level = L, active level = H) 1: Inverted output (inactive level = H, active level = L) This bit can be used to invert the timer output
6, 4, 2, 0	TABnOEm (m = 0 to 3)	Controls the TOABnm output. 0: Timer output is disabled (inactive level is output depending on the TABnOLm bit) 1: Timer output is enabled (TOABnm pin outputs pulses.)

(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 TAB0IOC1: FFFFF543_H TAB1IOC1: FFFFF613_H
TAB2IOC1: FFFFF623_H

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

	7	6	5	4	3	2	1	0
TABnIOC1	TABnIS7	TABnIS6	TABnIS5	TABnIS4	TABnIS3	TABnIS2	TABnIS1	TABnIS0
	R	R	R	R	R/W	R/W	R/W	R/W

Caution

1. Rewrite bits TABnIS[7:0] 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 TABnIS[7:0] 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.

Table 12-5 TABnIOC1 register contents (1/2)

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>7, 6</td><td>TABnIS[7:6]</td><td>Specifies the capture input (TIABn3) valid edge.
<table>
<tr><th>TABnIS7</th><th>TABnIS6</th><th>Capture input (TIABn3) valid edge setting</th></tr>
<tr><td>0</td><td>0</td><td>No edge detection (capture operation invalid)</td></tr>
<tr><td>0</td><td>1</td><td>Rising edge detection</td></tr>
<tr><td>1</td><td>0</td><td>Falling edge detection</td></tr>
<tr><td>1</td><td>1</td><td>Both, rising and falling edge detection</td></tr>
</table></td></tr>
<tr><td>5, 4</td><td>TABnIS[5:4]</td><td>Specifies the capture input (TIABn2) valid edge.
<table>
<tr><th>TABnIS5</th><th>TABnIS4</th><th>Capture input (TIABn2) valid edge setting</th></tr>
<tr><td>0</td><td>0</td><td>No edge detection (capture operation invalid)</td></tr>
<tr><td>0</td><td>1</td><td>Rising edge detection</td></tr>
<tr><td>1</td><td>0</td><td>Falling edge detection</td></tr>
<tr><td>1</td><td>1</td><td>Both, rising and falling edge detection</td></tr>
</table></td></tr>
<tr><td>3, 2</td><td>TABnIS[3:2]</td><td>Specifies the capture input (TIABn1) valid edge.
<table>
<tr><th>TABnIS3</th><th>TABnIS2</th><th>Capture input (TIABn1) valid edge setting</th></tr>
<tr><td>0</td><td>0</td><td>No edge detection (capture operation invalid)</td></tr>
<tr><td>0</td><td>1</td><td>Rising edge detection</td></tr>
<tr><td>1</td><td>0</td><td>Falling edge detection</td></tr>
<tr><td>1</td><td>1</td><td>Both, rising and falling edge detection</td></tr>
</table></td></tr>
</table>

Table 12-5 TABnIOC1 register contents (2/2)

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>1, 0</td><td>TABnIS[1:0]</td><td>Specifies the capture input (TIAAn0) valid edge.
<table>
<tr><th>TABnIS1</th><th>TABnIS0</th><th>Capture input (TIAAn0) valid edge setting</th></tr>
<tr><td>0</td><td>0</td><td>No edge detection (capture operation invalid)</td></tr>
<tr><td>0</td><td>1</td><td>Rising edge detection</td></tr>
<tr><td>1</td><td>0</td><td>Falling edge detection</td></tr>
<tr><td>1</td><td>1</td><td>Both, rising and falling edge detection</td></tr>
</table>
</td></tr>
</table>

Rewrite during timer operation

If the edge specification for the capture operation shall 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 exemplarily):

- 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”

(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).

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

$\overline{\text{RESET}}$ input clears this register to 00H.

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

Address TAB0IOC2: FFFFF544$_H$ TAB1IOC2: FFFFF614$_H$
TAB2IOC2: FFFFF624$_H$

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

	7	6	5	4	3	2	1	0
TABnIOC2	0	0	0	0	TABnEES1	TABnEES0	TABnETS1	TABnETS0
	R	R	R	R	R/W	R/W	R/W	R/W

Caution Rewrite TABnEES[1:0] and TABnETS[1:0] bits 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.

Table 12-6 TABnIOC2 register contents

Bit position	Bit name	Function
3, 2	TABnEES1 TABnEES0	Specifies the external event counter input (TIABn0) valid edge.

TABnEES1	TABnEES0	External event counter input (TIABn0) valid edge setting
0	0	No edge detection (external event count is invalid)
0	1	Rising edge detection
1	0	Falling edge detection
1	1	Both, rising and falling edge detection

Caution: The TABnEES[1:0] bits are valid when TABnEEE = 1, or when the external event count mode is set (TABnMD[2:0]=001$_B$).

Bit position	Bit name	Function
1, 0	TABnETS1 TABnETS0	Specifies the external trigger input (TIABn0) valid edge.

TABnETS1	TABnETS0	External trigger input (TIABn0) valid edge setting
0	0	No edge detection (external trigger is invalid)
0	1	Rising edge detection
1	0	Falling edge detection
1	1	Both, rising and falling edge detection

Caution: The TABnETS[1:0] bits are valid when the external trigger pulse output mode or one-shot pulse mode is set (TABnMD[2:0]=010$_B$ or 011$_B$).

Rewrite during timer operation

If the edge specification for the external event counter input signal or the external trigger input signal shall 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”

Ensure the input level is not changing while the TABnIOC2 register is modified.

(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 00H.

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

Address TAB0IOC4: FFFFF59C$_H$ TAB1IOC4 FFFFF5AC$_H$
TAB2IOC4: FFFFF5BC$_H$

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

	7	6	5	4	3	2	1	0
TABnIOC4	TABnOS3	TABnOR3	TABnOS2	TABnOR2	TABnOS1	TABnOR1	TABnOS0	TABnOR0
	R	R	R	R	R/W	R/W	R/W	R/W

Note Writing to TABnIOC4 is also possible, when TABTABnCTL0.TABnCE = 1.

Table 12-7 TABnIOC4 register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>7 to0</td><td>TABnOSm
TABnORm
(m = 0 to 3)</td><td>Controls toggling of the timer output TOABnm.
<table>
<tr><th>TABnOSm</th><th>TABnORm</th><th>Toggle Control of TOABnm</th></tr>
<tr><td>0</td><td>0</td><td>Standard operation.</td></tr>
<tr><td>0</td><td>1</td><td>Force output level to inactive at next toggle event</td></tr>
<tr><td>1</td><td>0</td><td>Force output level to active at next toggle event</td></tr>
<tr><td>1</td><td>1</td><td>Freeze current output level.</td></tr>
</table>
<b>Note:</b> <b>1.</b> After forcing the output level to either active or inactive, the TOABnm output maintains this level (= no toggling afterwards) until the TABnOSm and TABnORm are cleared to standard operation.
<b>2.</b> The forcing of an output level is executed at the time of the next upcoming toggle event, while the freeze becomes effective immediately.</td></tr>
</table>

(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 TAB0OPT0: FFFFF545$_H$ TAB1OPT0: FFFFF615$_H$
TAB2OPT0: FFFFF625$_H$

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

TAB0OPT0

7	6	5	4	3	2	1	0
TAB0CCS3	TAB0CCS2	TAB0CCS1	TAB0CCS0	0	TAB0CMS	TAB0CUF	TAB0OVF
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

TABnOPT0 (m = 1, 2)

7	6	5	4	3	2	1	0
TABnCCS3	TABnCCS2	TABnCCS1	TABnCCS0	0	0	TABnCUF	TABnOVF
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Caution

1. Rewrite TABnCCS[3:0] bits when TABnCE = 0 (the same value can be written when TABnCE = 1.). If rewriting was mistakenly performed, clear TABnCE to 0 and then set the bits again.
2. Be sure to clear bit 3 to 0, and bit 2 of registers TAB1OPT0 and TAB2OPT0 to 0 as well.

Table 12-8 TABnOPT0 register contents

Bit position	Bit name	Function
7 to 4	TABnCCSm (m = 0 to 3)	Specifies the operation mode of register TABnCCRm 0: Operation as compare register 1: Operation as capture register **Note:** The setting of bit TABnCCS1 is valid in the free-running mode only.
2	TAB0CMS	For details refer to *“Motor Control Function” on page 856*.
1	TABnCUF	For details refer to *“Motor Control Function” on page 856*.
0	TABnOVF	Indicates timer TABn overflow 0: No overflow occurrence after timer restart or flag reset 1: Overflow occurrence • 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. • An interrupt request signal (INTTABnOV) is generated as soon as TABnOVF bit is set (1). The INTTABnOV signal is not generated in any mode other than free-running mode and the pulse width measurement mode. • The TABnOVF bit is cleared by writing 0 to it, or if TABnCTL0.TABnCE is set to 0. **Caution:** 1. The TABnOVF bit is not cleared even when the TABnOVF bit and the TABnOPT0 register are read when TABnOVF = 1. 2. The TABnOVF bit can be read and written, but writing 1 to TABnOVF does not set it and has no influence on the operation of timer AB.

12.5 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	Any time write
External event counter mode[a]	Invalid	Invalid	TIABn0 pin only	Compare only	Any time write
External trigger pulse output mode[b]	Valid	Valid	Internal only	Compare only	Reload
One-shot pulse output mode[b]	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[b]	Invalid	Invalid	Internal only	Capture only	Not applicable

a) When using the external event count function, set TIABn0 capture input edge detection to “no edge detection” (set TABnIOC1.TABnIS[1:0] bits to 00_B.)

b) 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 TABnCTL1.TABnEEE bit to 1).

Caution Clearing the TABnCCR1 register to 0000_H is prohibited in the one-shot pulse mode.

12.5.1 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.

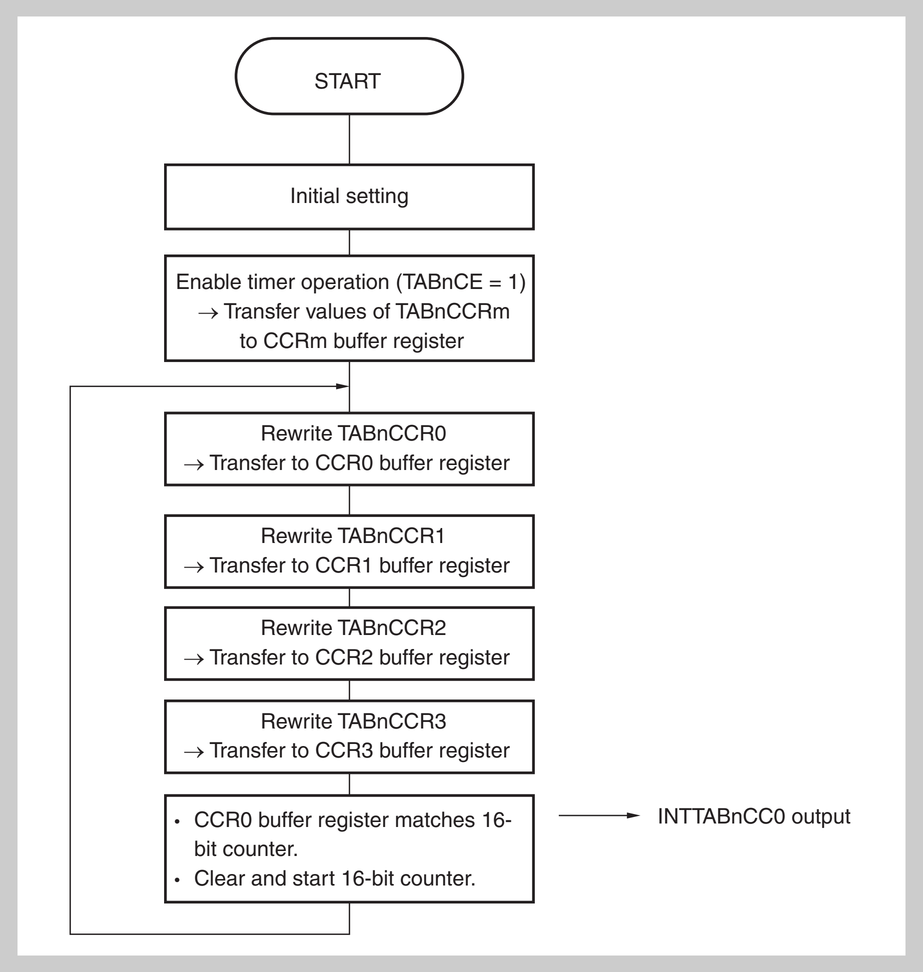

START
Initial setting
Enable timer operation (TABnCE = 1)
→ Transfer values of TABnCCRm
to CCRm buffer register
Rewrite TABnCCR0
→ Transfer to CCR0 buffer register
Rewrite TABnCCR1
→ Transfer to CCR1 buffer register
Rewrite TABnCCR2
→ Transfer to CCR2 buffer register
Rewrite TABnCCR3
→ Transfer to CCR3 buffer register
• CCR0 buffer register matches 16-bit counter.
• Clear and start 16-bit counter.
INTTABnCC0 output

Figure 12-2 Flowchart of basic operation for anytime write

Note The above flowchart illustrates an example of the operation in the interval timer mode.

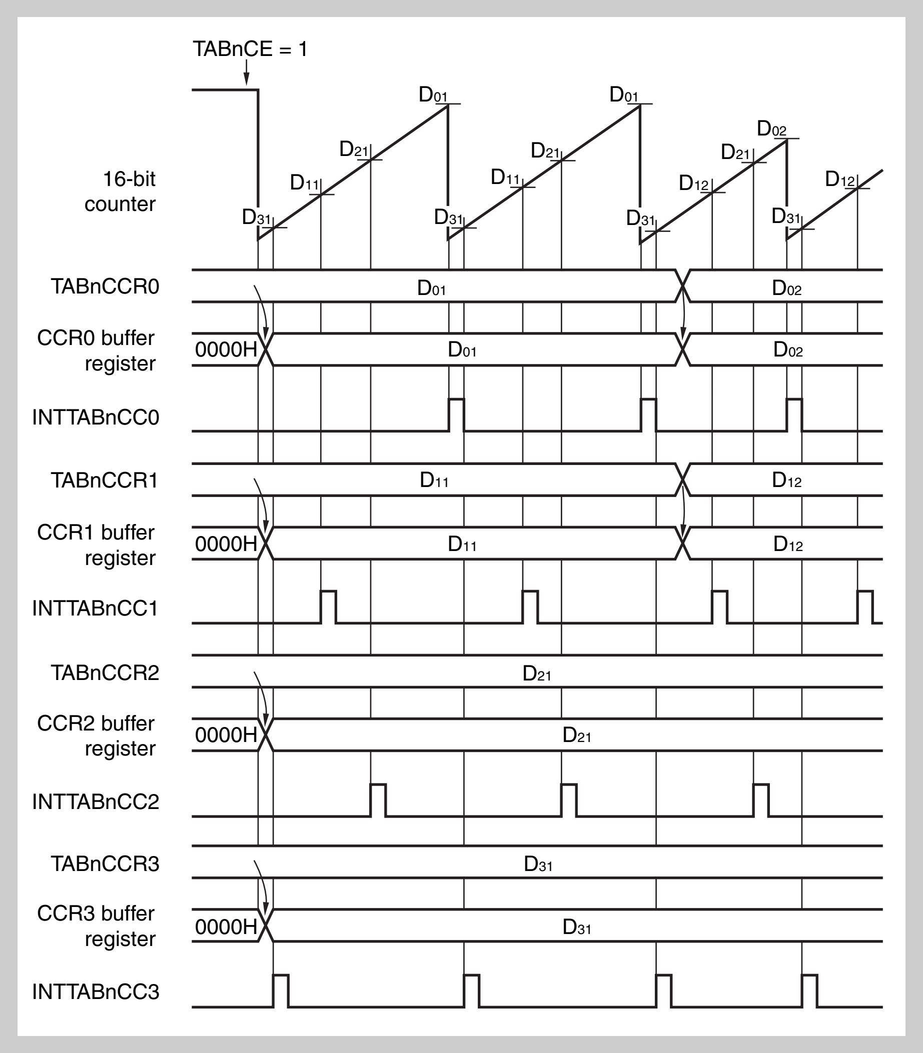


Figure 12-3 Timing chart of anytime write

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.

Caution Though the compare registers can be written at any time, the write access will be synchronized with the internal count clock, depending on setting of the SELCNT4.SEL40 bit, TABnCTL0.TABnCKS[2:0] bits and PRSI bit (of the option byte 0000 007B$_H$). Due to this synchronization a delay has to be taken into account.
Particularly when the dedicated interrupt request flag of a capture/compare interrupt is cleared directly after rewriting the capture/compare register, an unexpected interrupt request may occur, since the new compare value is still not synchronized and accepted. The occurrence of the accidental interrupt can be avoided by a certain delay between capture/compare register write and clearing of the interrupt request flag. An applicable delay can be achieved by a consecutive write of the same capture/compare register.

Example

1. Write capture/compare register
2. Write same capture/compare register again
   (delays program execution until the synchronization takes effect)
3. Clear dedicated interrupt request flag

(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.

So that the set values of the TABnCCRm register is 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.

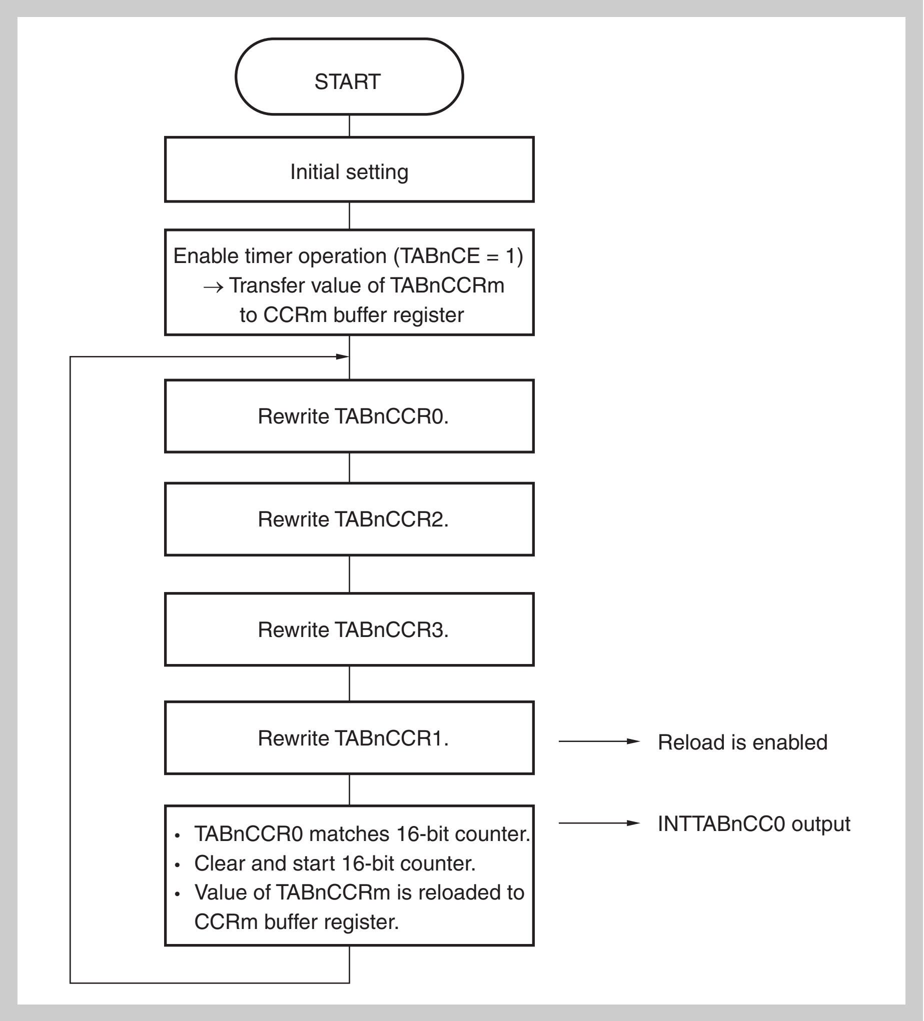

START
Initial setting
Enable timer operation (TABnCE = 1)
→ Transfer value of TABnCCRm
to CCRm buffer register
Rewrite TABnCCR0.
Rewrite TABnCCR2.
Rewrite TABnCCR3.
Rewrite TABnCCR1.
Reload is enabled
• TABnCCR0 matches 16-bit counter.
• Clear and start 16-bit counter.
• Value of TABnCCRm is reloaded to CCRm buffer register.
INTTABnCC0 output

Figure 12-4 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. The above flowchart illustrates an example of PWM mode operation.
2. m = 0 to 3

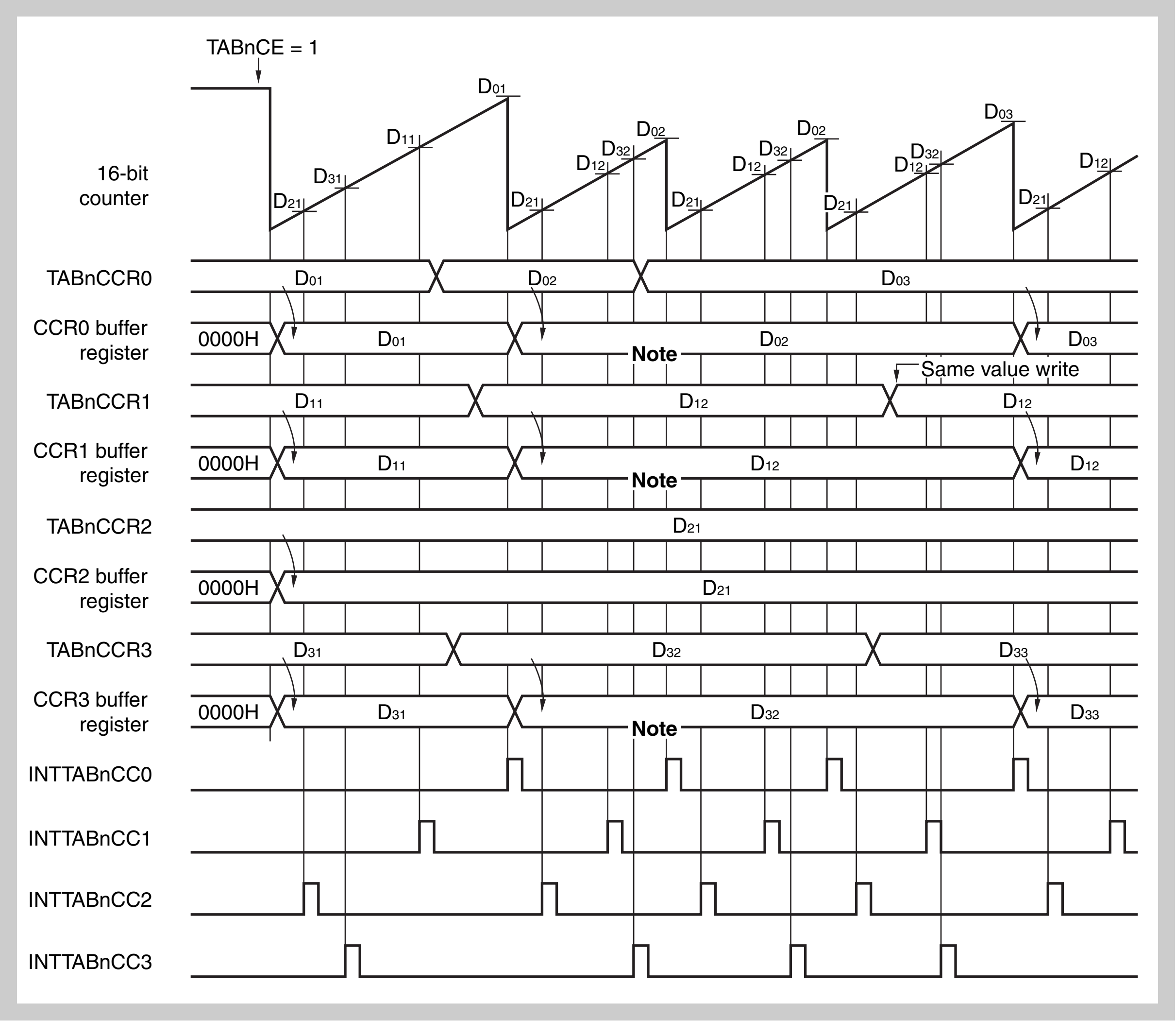


Figure 12-5 Timing chart of reload

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 above flowchart illustrates the operation in the PWM mode operation.

12.5.2 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 TOABnk pin by setting the TABnOEk 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.

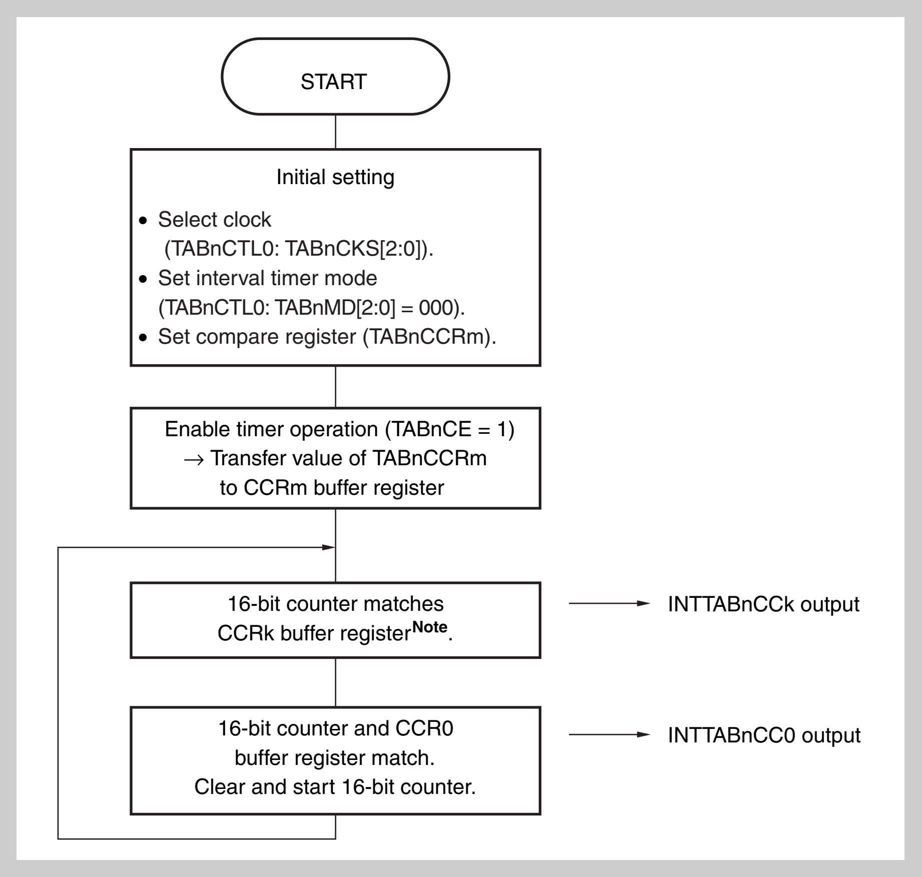

START
Initial setting
• Select clock (TABnCTL0: TABnCKS[2:0]).
• Set interval timer mode (TABnCTL0: TABnMD[2:0] = 000).
• Set compare register (TABnCCRm).
Enable timer operation (TABnCE = 1) → Transfer value of TABnCCRm to CCRm buffer register
16-bit counter matches CCRk buffer register[Note].
INTTABnCCk output
16-bit counter and CCR0 buffer register match. Clear and start 16-bit counter.
INTTABnCC0 output

Figure 12-6 Flowchart of Basic operation in interval timer mode

Note

1. The 16-bit counter is not cleared upon a match between the 16-bit counter and TABnCCRk.
2. m = 0 to 3; k = 1 to 3

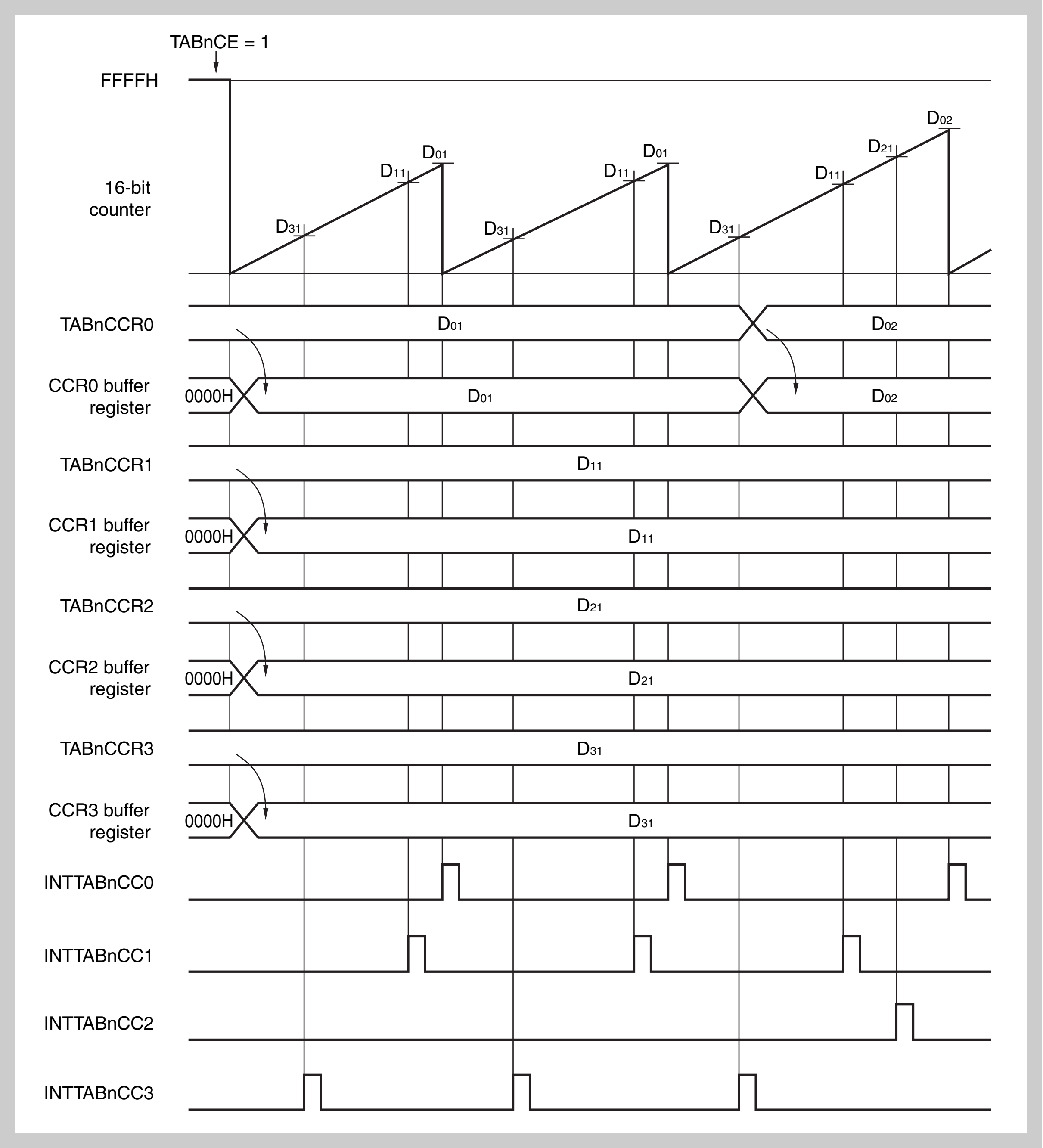


Figure 12-7 Basic operation timing in interval timer mode (1/2) (When only TABnCCR0 register value is rewritten and TOABnm is not output)

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

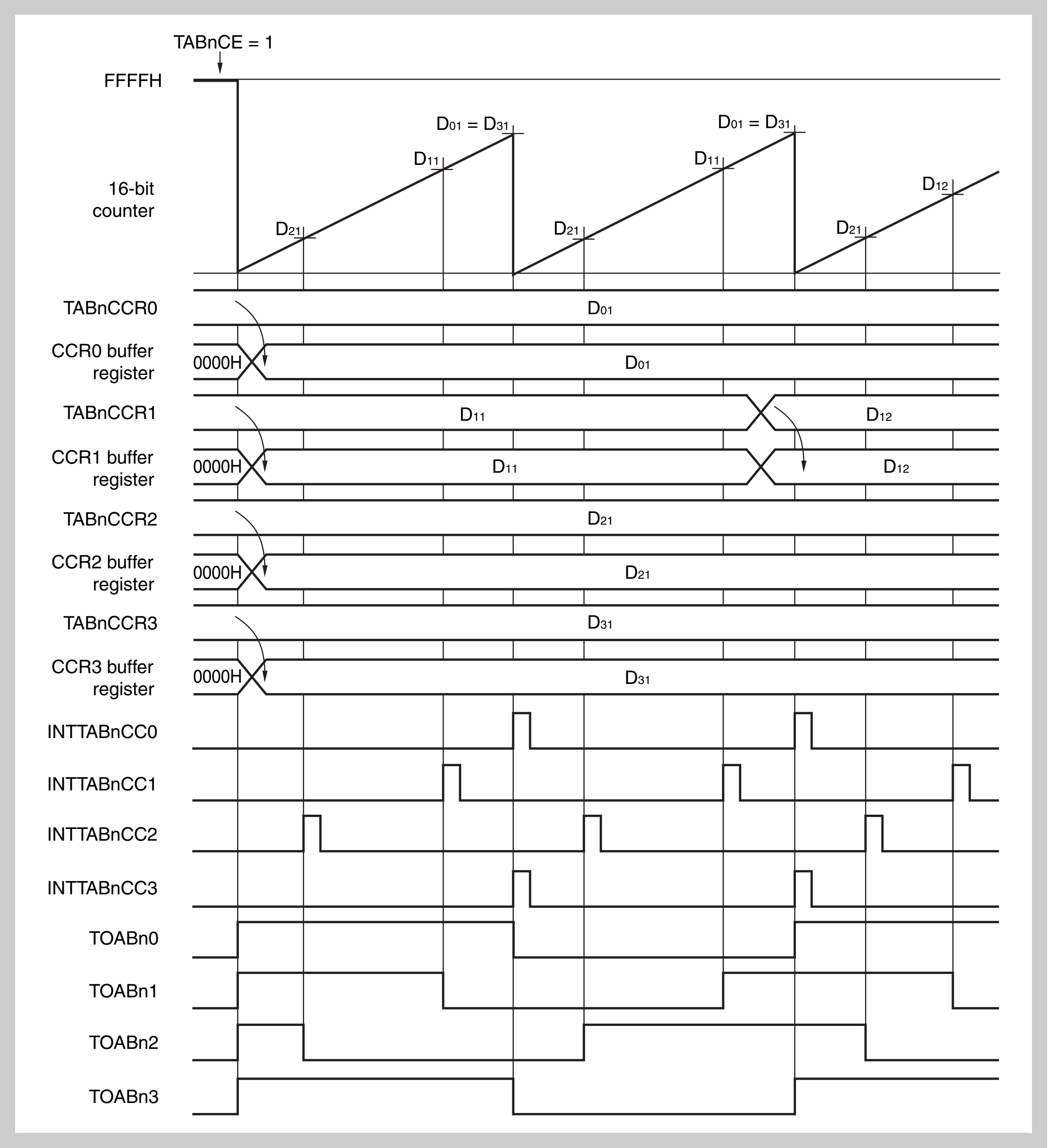


Figure 12-8 Basic operation timing in interval timer mode (2/2) (when D01 = D31, only TABnCCR1 register value is rewritten, and TOABnm is output)

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

12.5.3 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 TABnCTL0TABnEEE bit, 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

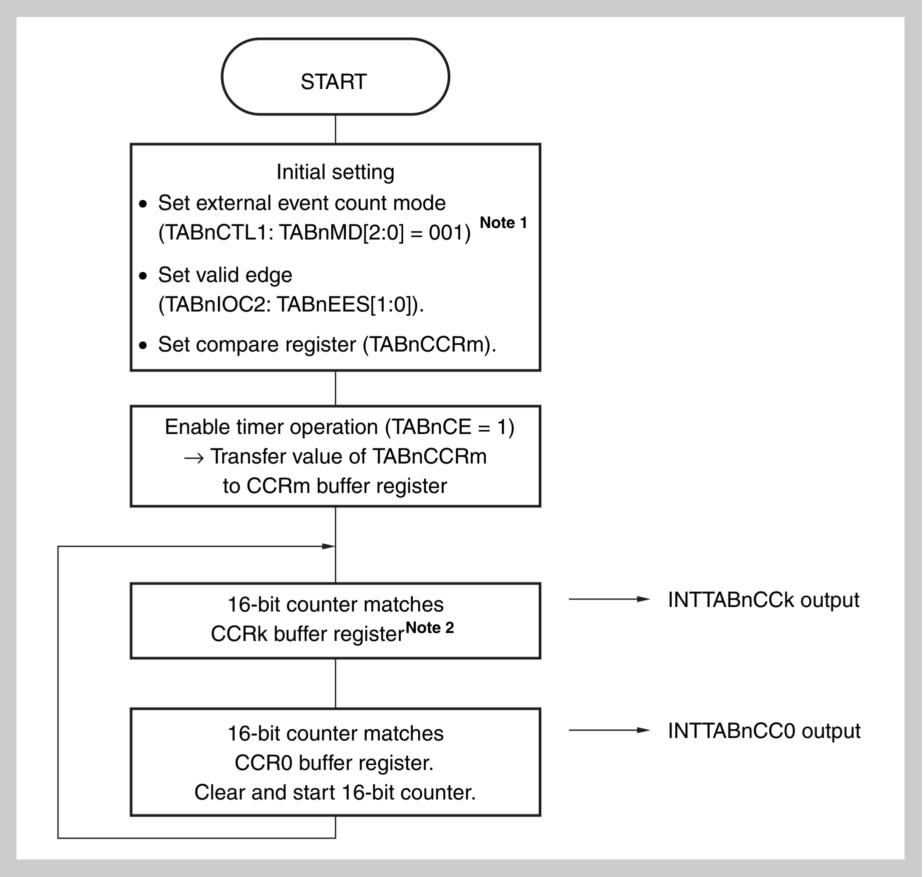

START
Initial setting
• Set external event count mode (TABnCTL1: TABnMD[2:0] = 001) Note 1
• Set valid edge (TABnIOC2: TABnEES[1:0]).
• Set compare register (TABnCCRm).
Enable timer operation (TABnCE = 1)
→ Transfer value of TABnCCRm to CCRm buffer register
16-bit counter matches CCRk buffer register Note 2
INTTABnCCk output
16-bit counter matches CCR0 buffer register. Clear and start 16-bit counter.
INTTABnCC0 output

Figure 12-9 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.

Note m = 0 to 3; k = 1 to 3

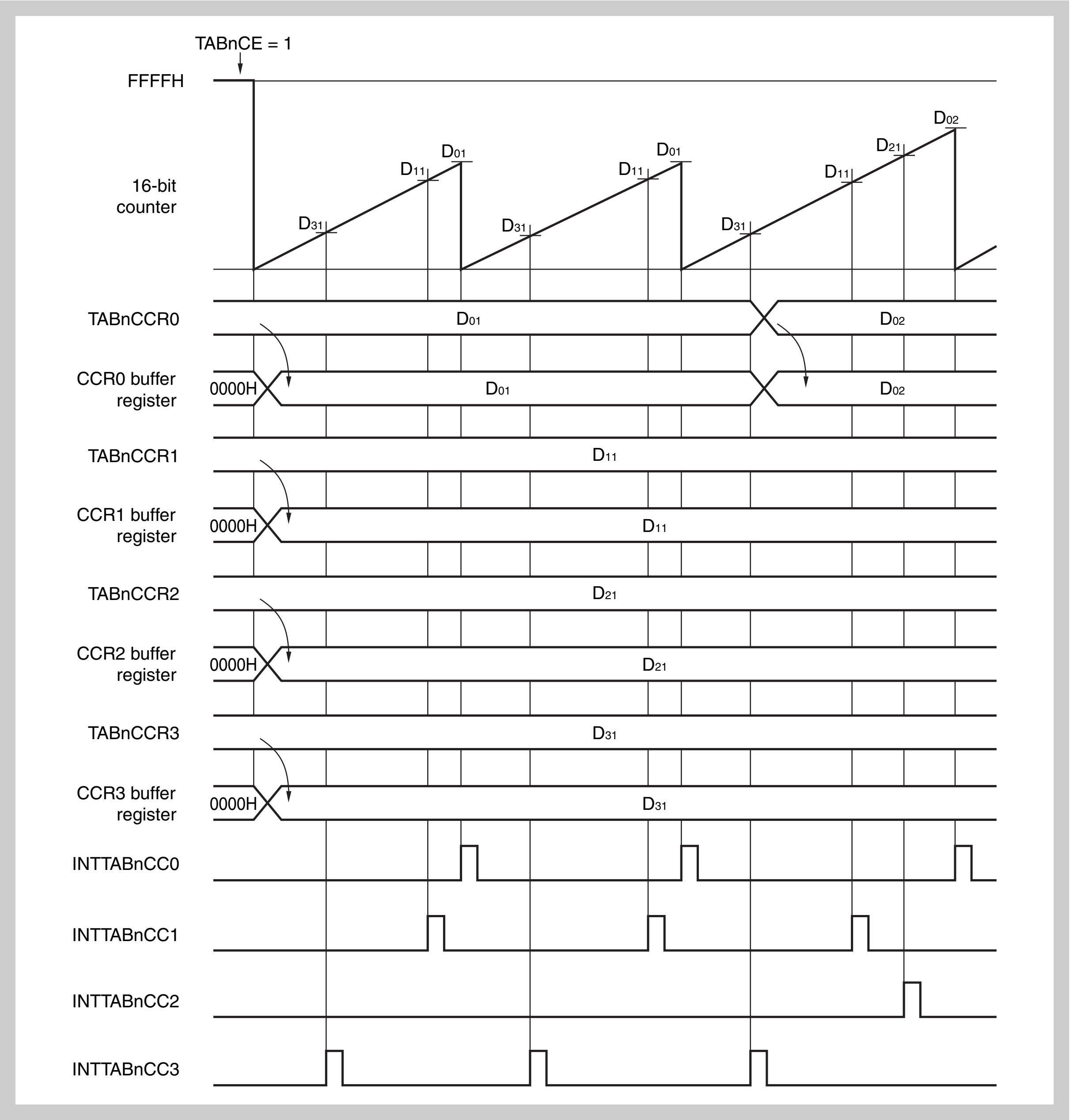


Figure 12-10 Basic operation timing in external event counter mode (1/2) (when only TABnCCR0 register value is rewritten and TOABnm is not output)

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

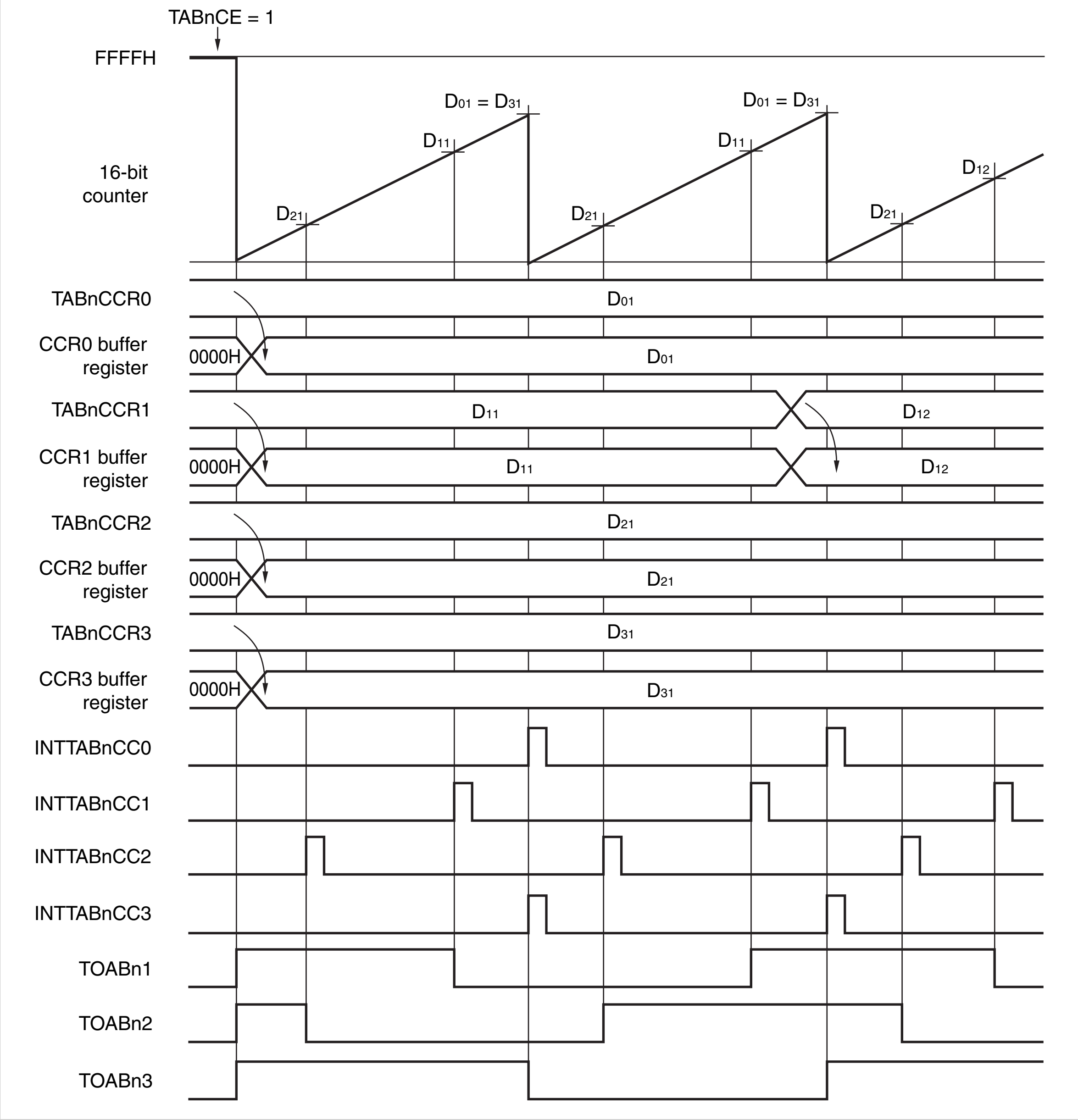


Figure 12-11 Basic operation timing in external event counter mode (2/2) (when D01 = D31, only TABnCCR1 register is rewritten, and TOABnk is output)

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

12.5.4 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.

To realize the same function as the external trigger pulse mode by using a software trigger instead of the external trigger input (TIABn0 pin input) (software trigger pulse mode), a software trigger is generated by setting the TABnEST bit of the TABnCTL1 regist.

In the external trigger pulse mode, the capture function of the TABnCCRm register cannot be used because this register can be 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 *“Anytime write and reload” on page 487*.
2. m = 0 to 3; k = 1 to 3

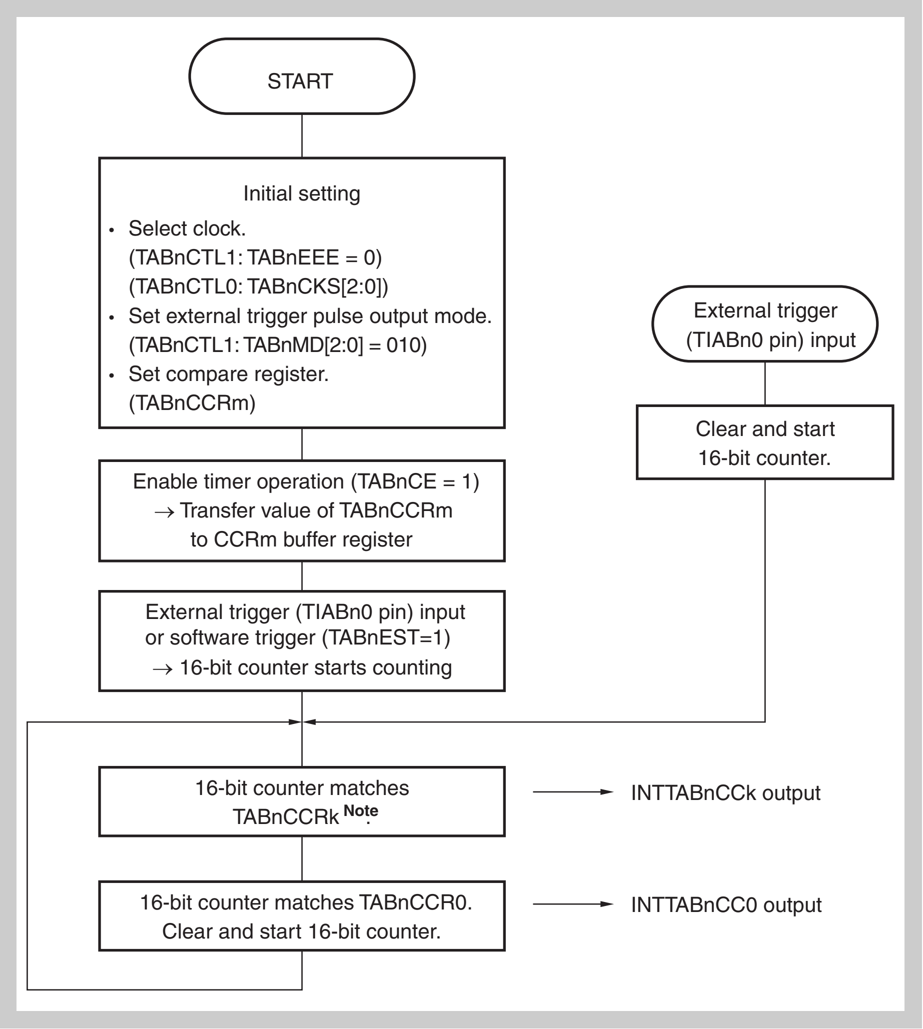

START
Initial setting
• Select clock.
(TABnCTL1: TABnEEE = 0)
(TABnCTL0: TABnCKS[2:0])
• Set external trigger pulse output mode.
(TABnCTL1: TABnMD[2:0] = 010)
• Set compare register.
(TABnCCRm)
Enable timer operation (TABnCE = 1)
→ Transfer value of TABnCCRm
to CCRm buffer register
External trigger (TIABn0 pin) input
or software trigger (TABnEST=1)
→ 16-bit counter starts counting
External trigger
(TIABn0 pin) input
Clear and start
16-bit counter.
16-bit counter matches
TABnCCRk Note.
INTTABnCCk output
16-bit counter matches TABnCCR0.
Clear and start 16-bit counter.
INTTABnCC0 output

Figure 12-12 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 TABnCTL .TABnEST bit can be rewritten during timer operation (TABnCE = 1).
3. m = 0 to 3; k = 1 to 3

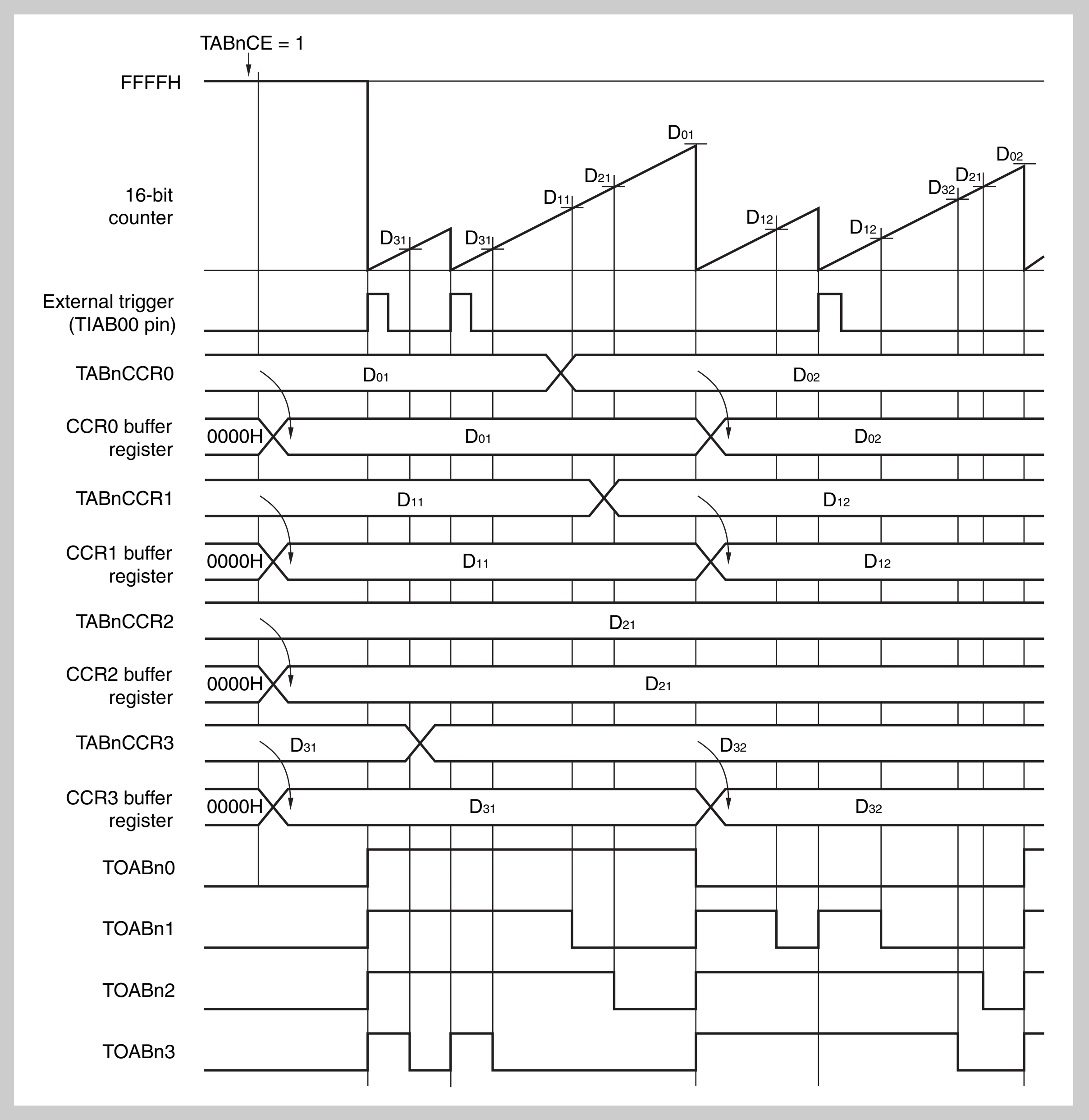


Figure 12-13 Basic operation timing in external trigger pulse output mode

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

12.5.5 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 TOABn0 pin outputs an active level while the 16-bit counter is counting, and an inactive level when the counter is stopped (waiting for a trigger).

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 only as a compare register. It cannot be used as a capture register.

Caution In the one-shot pulse mode, select the internal clock for the count clock (TABnCTL1.TABnEEE bit = 0).

Note m = 0 to 3; k = 1 to 3

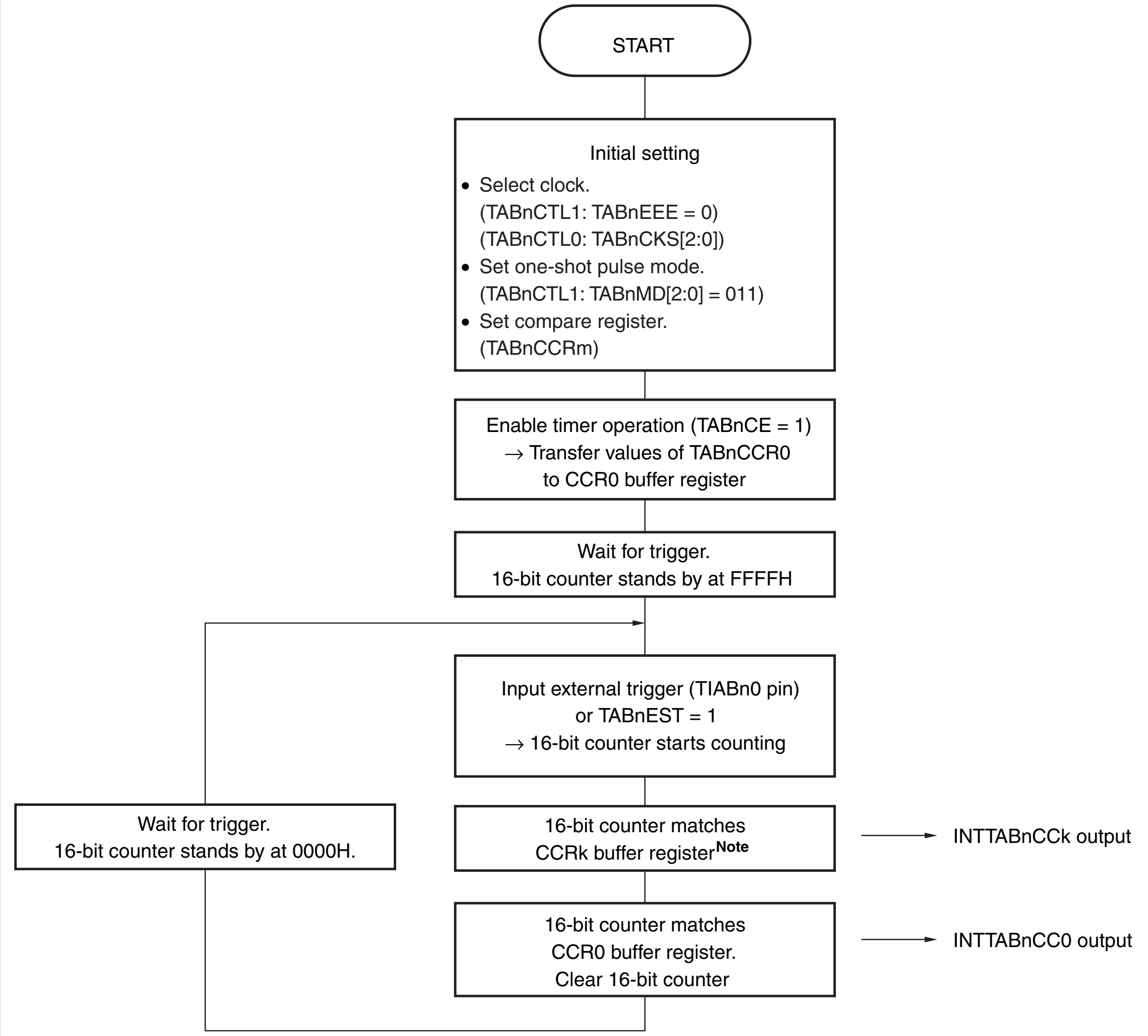


Figure 12-14 Flowchart of basic operation in one-shot pulse mode

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

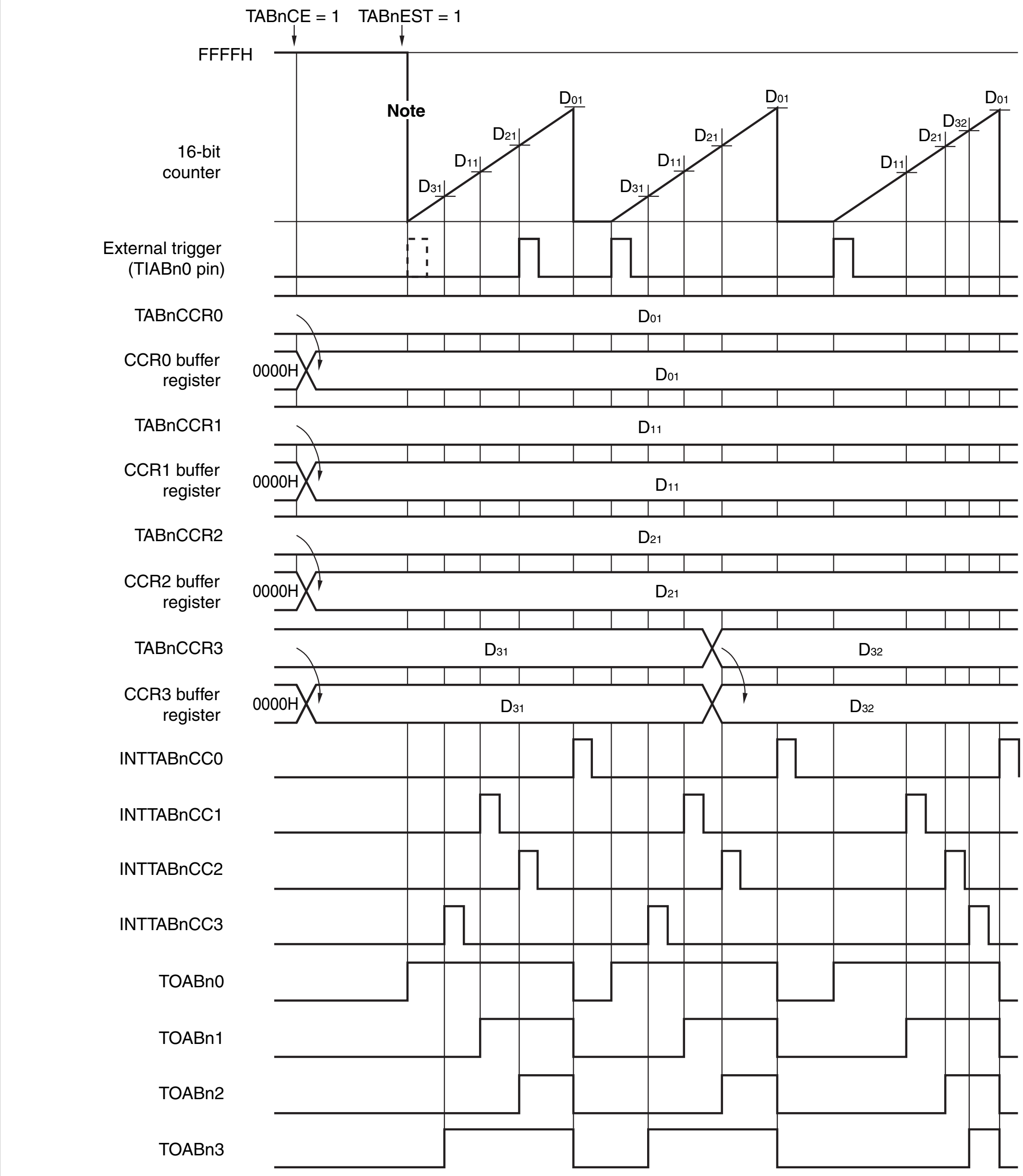


Figure 12-15 Timing of basic operation in one-shot pulse mode

Note

1. The 16-bit counter starts counting up when either TABnCTL.TABnEST is set to 1 or external trigger (TIABn0 pin) is input.
2. Only the TABnEST bit 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

12.5.6 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 four 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, 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

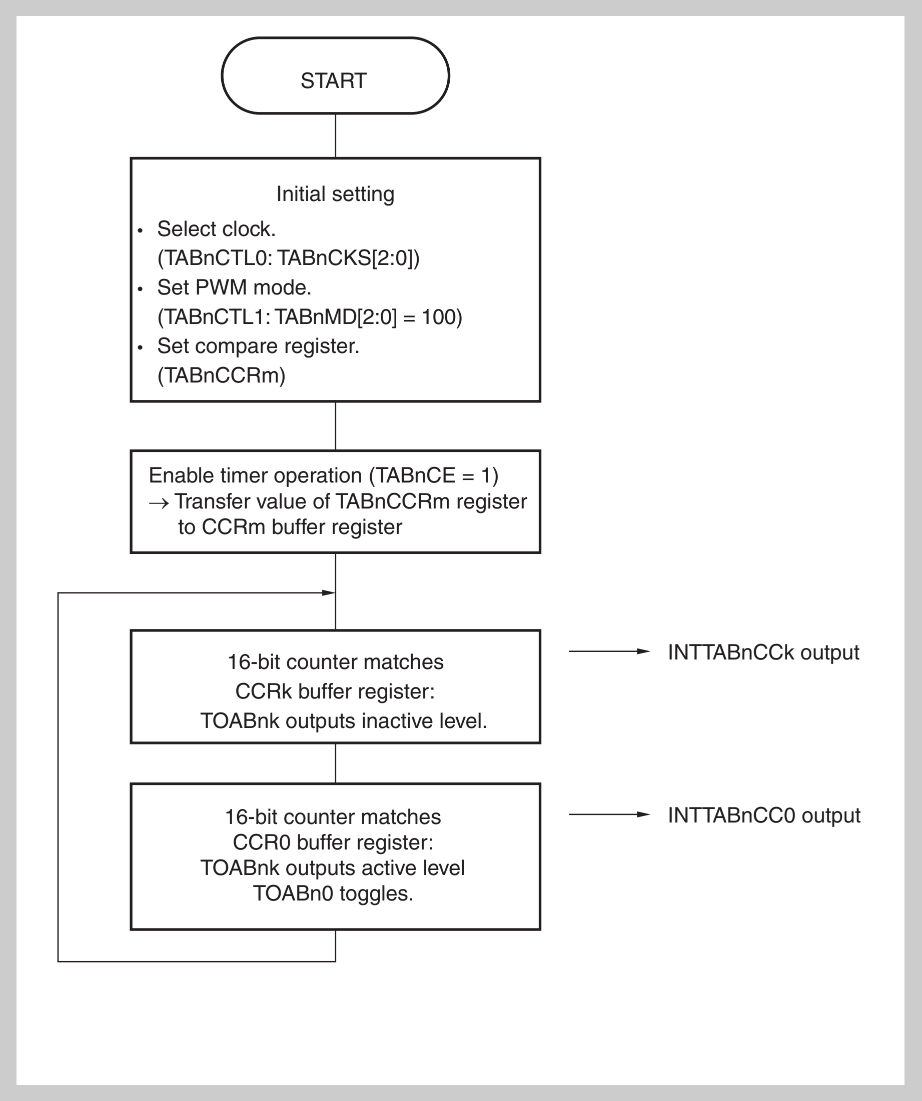

START
Initial setting
• Select clock.
(TABnCTL0: TABnCKS[2:0])
• Set PWM mode.
(TABnCTL1: TABnMD[2:0] = 100)
• Set compare register.
(TABnCCRm)
Enable timer operation (TABnCE = 1)
→ Transfer value of TABnCCRm register to CCRm buffer register
16-bit counter matches CCRk buffer register: TOABnk outputs inactive level.
INTTABnCCk output
16-bit counter matches CCR0 buffer register: TOABnk outputs active level TOABn0 toggles.
INTTABnCC0 output

Figure 12-16 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

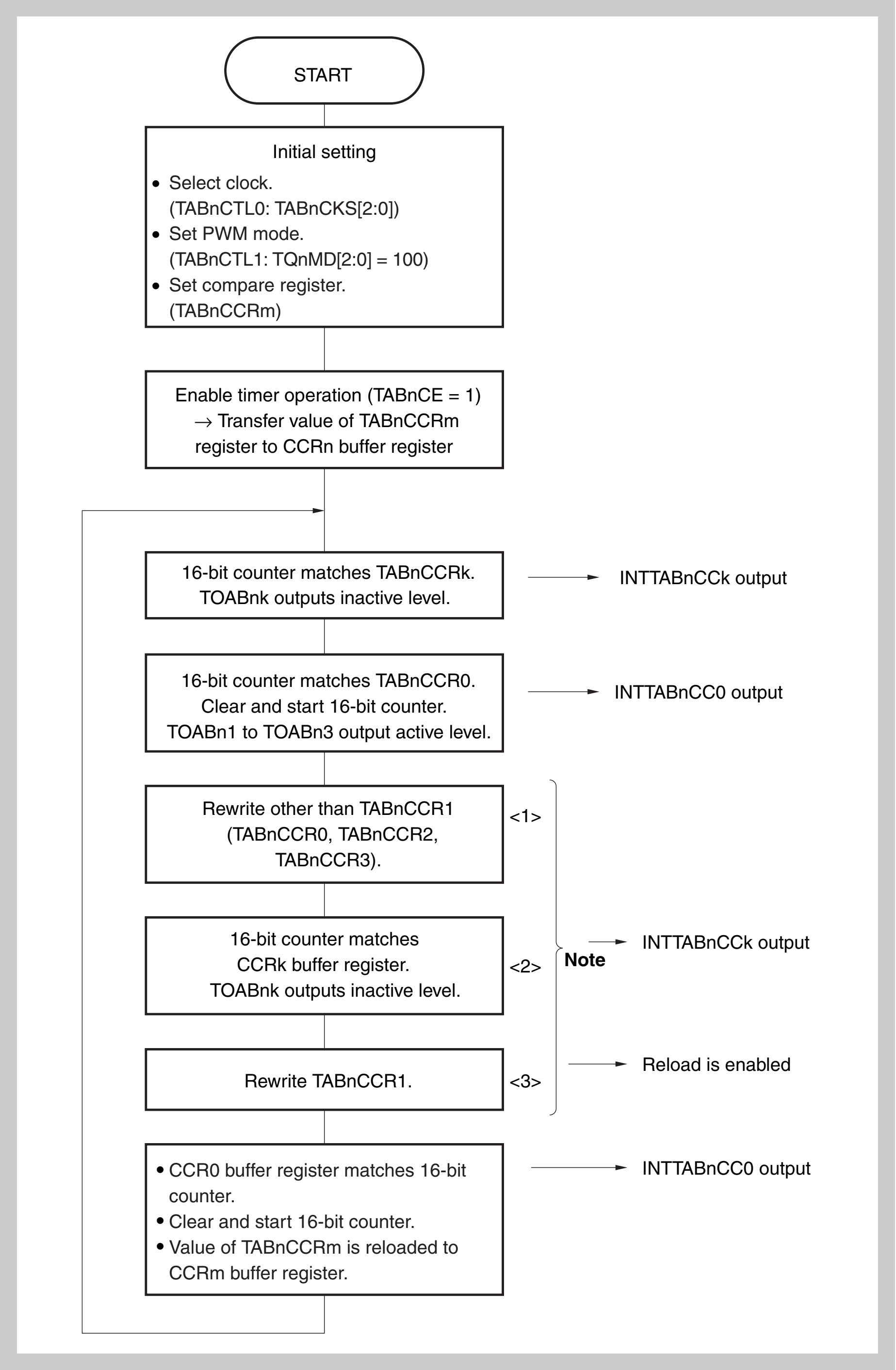

START
Initial setting
• Select clock.
(TABnCTL0: TABnCKS[2:0])
• Set PWM mode.
(TABnCTL1: TQnMD[2:0] = 100)
• Set compare register.
(TABnCCRm)
Enable timer operation (TABnCE = 1)
→ Transfer value of TABnCCRm
register to CCRn buffer register
16-bit counter matches TABnCCRk.
TOABnk outputs inactive level.
INTTABnCCk output
16-bit counter matches TABnCCR0.
Clear and start 16-bit counter.
TOABn1 to TOABn3 output active level.
INTTABnCC0 output
Rewrite other than TABnCCR1
(TABnCCR0, TABnCCR2,
TABnCCR3).
<1>
16-bit counter matches
CCRk buffer register.
TOABnk outputs inactive level.
<2>
Note
INTTABnCCk output
Rewrite TABnCCR1.
<3>
Reload is enabled
• CCR0 buffer register matches 16-bit
counter.
• Clear and start 16-bit counter.
• Value of TABnCCRm is reloaded to
CCRm buffer register.
INTTABnCC0 output

Figure 12-17 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 above flowchart 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

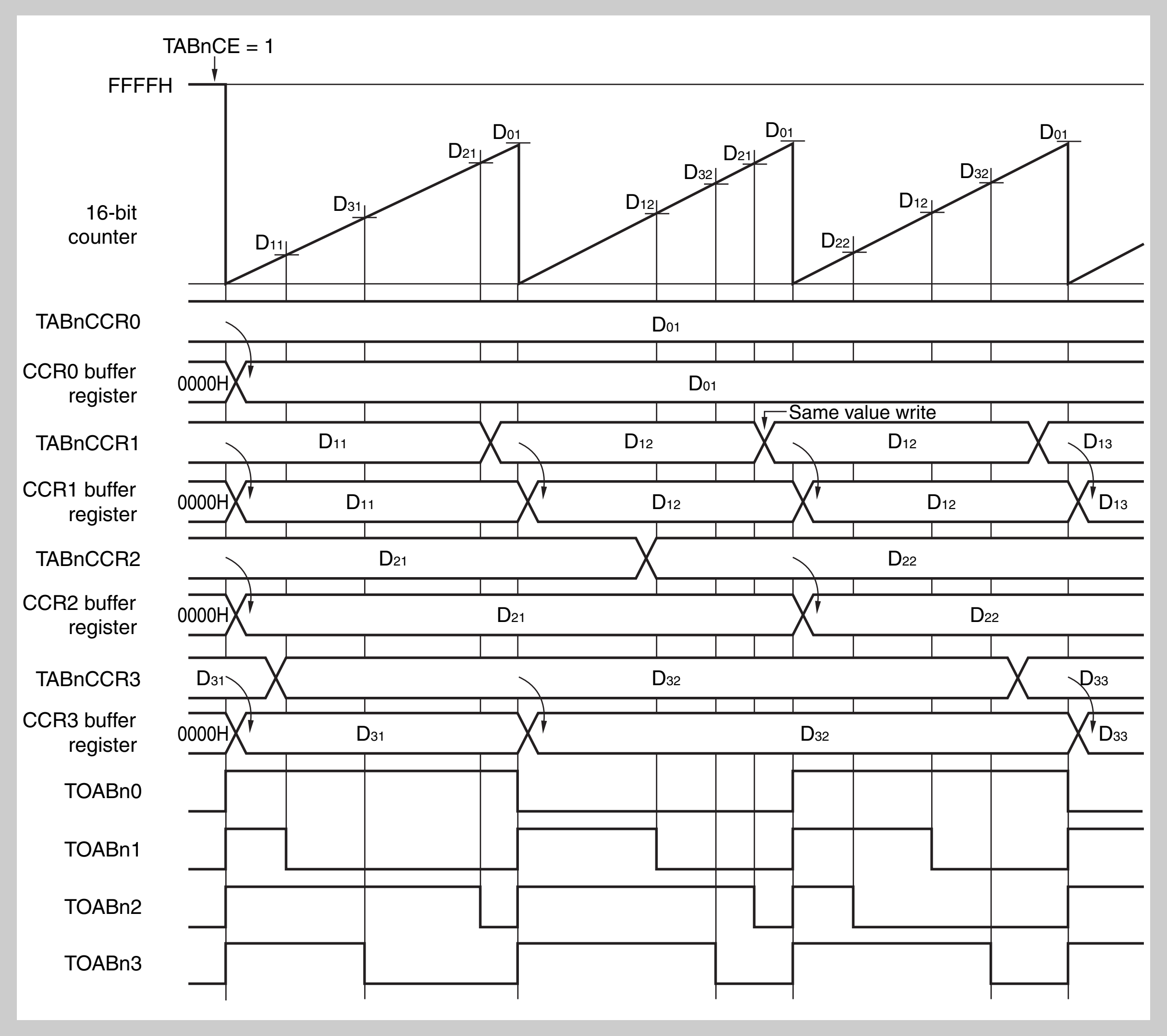


Figure 12-18 Basic operation timing in PWM mode (1/2) (when rewriting values of TABnCCR1 to TABnCCR3 registers)

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 factor 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)
3. k = 1 to 3

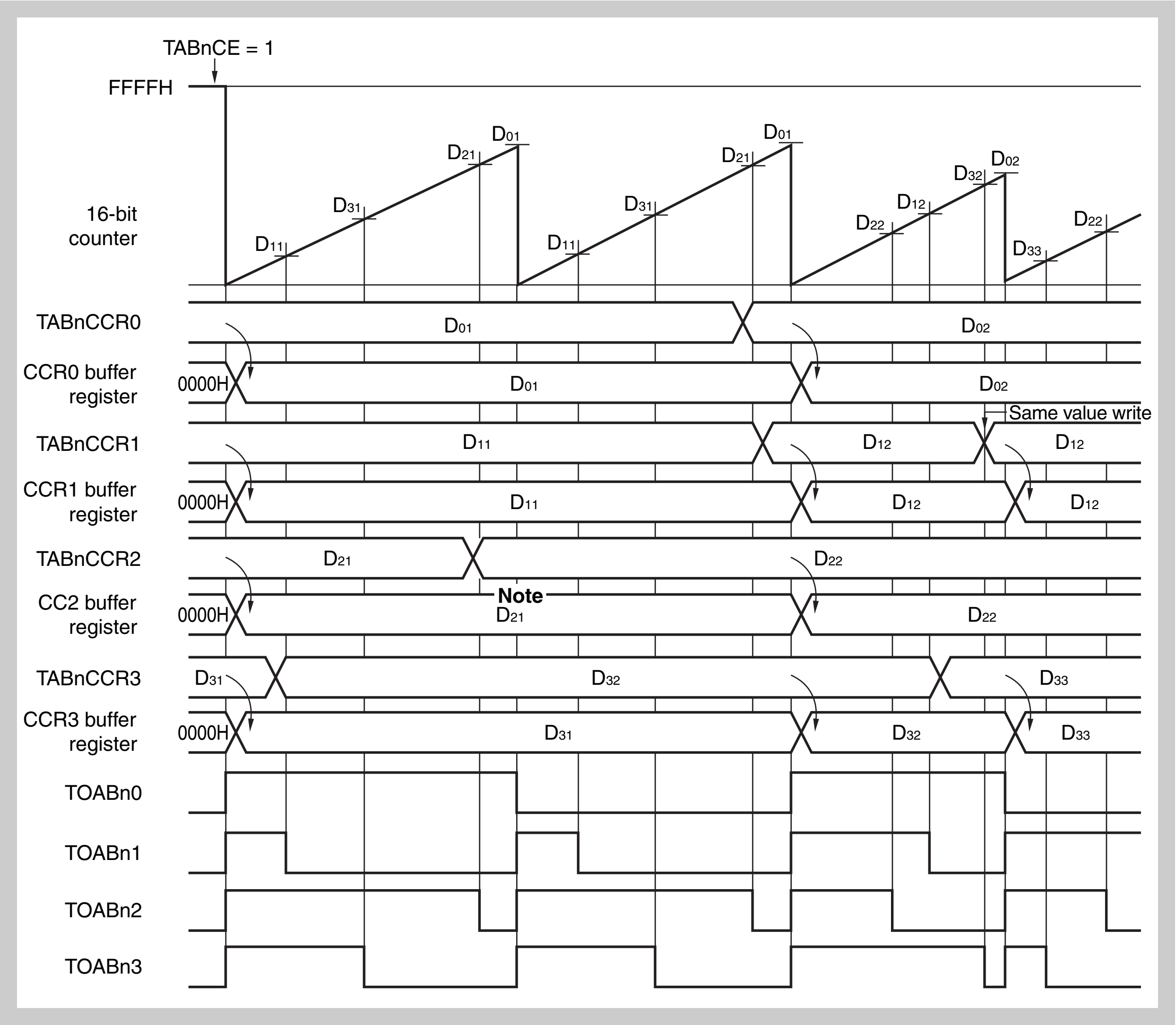


Figure 12-19 Basic operation timing in PWM mode (2/2) (when rewriting values of TABnCCR0 to TABnCCR3 registers)

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 factor 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.

12.5.7 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 TABnOPT0.TABnCCS[3:0] bits.

The settings of the TABnOPT0.TABnCCS[3:0] bits of the 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. Caution The TABnCCR0 register cannot be used as the capture register when the TABnCTL1.TABnEEE bit 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 *“Anytime write and reload” on page 487*.
3. n = 0 to 2, m = 0 to 3

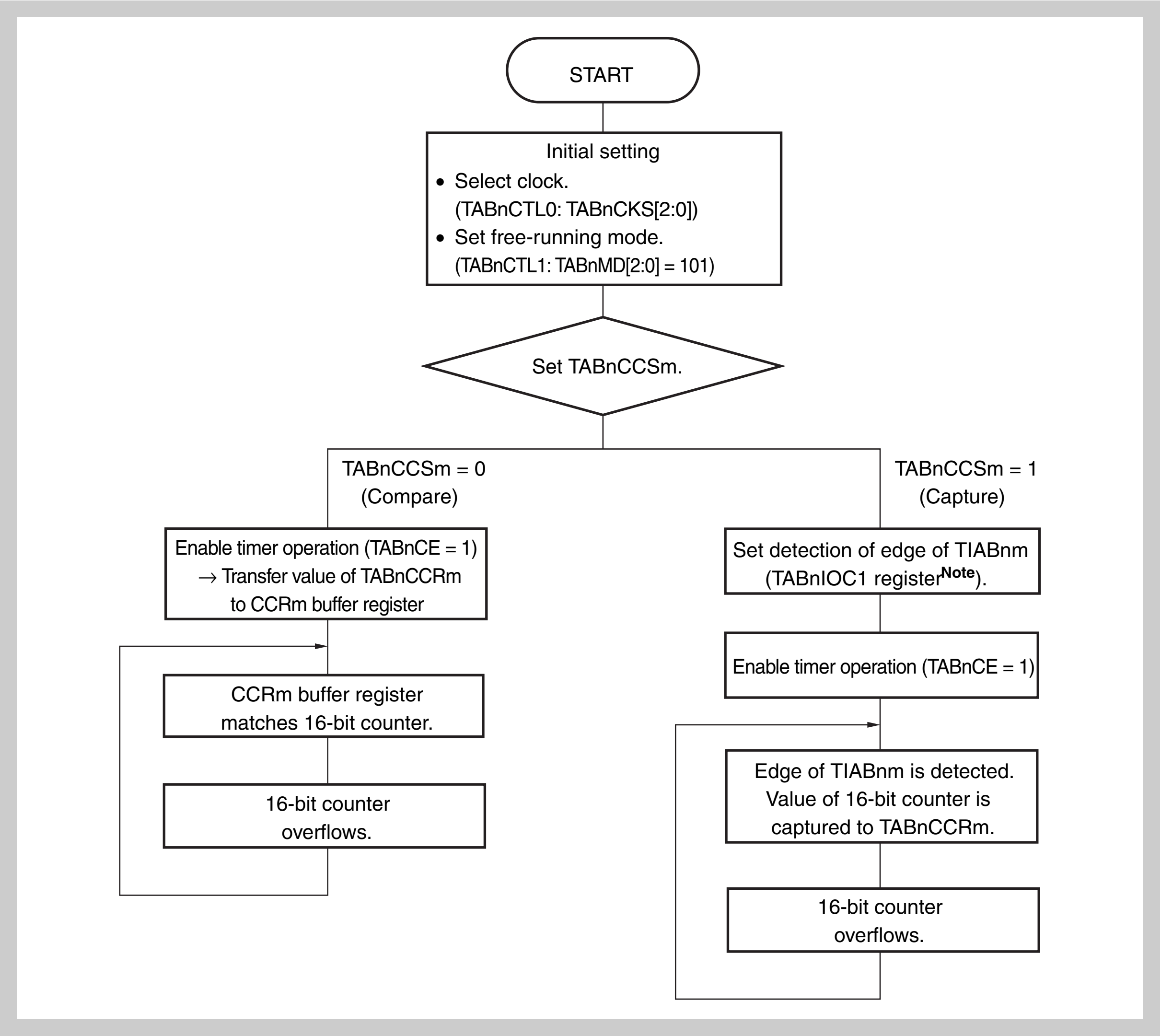


Figure 12-20 Flowchart of basic operation in free-running mode

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.

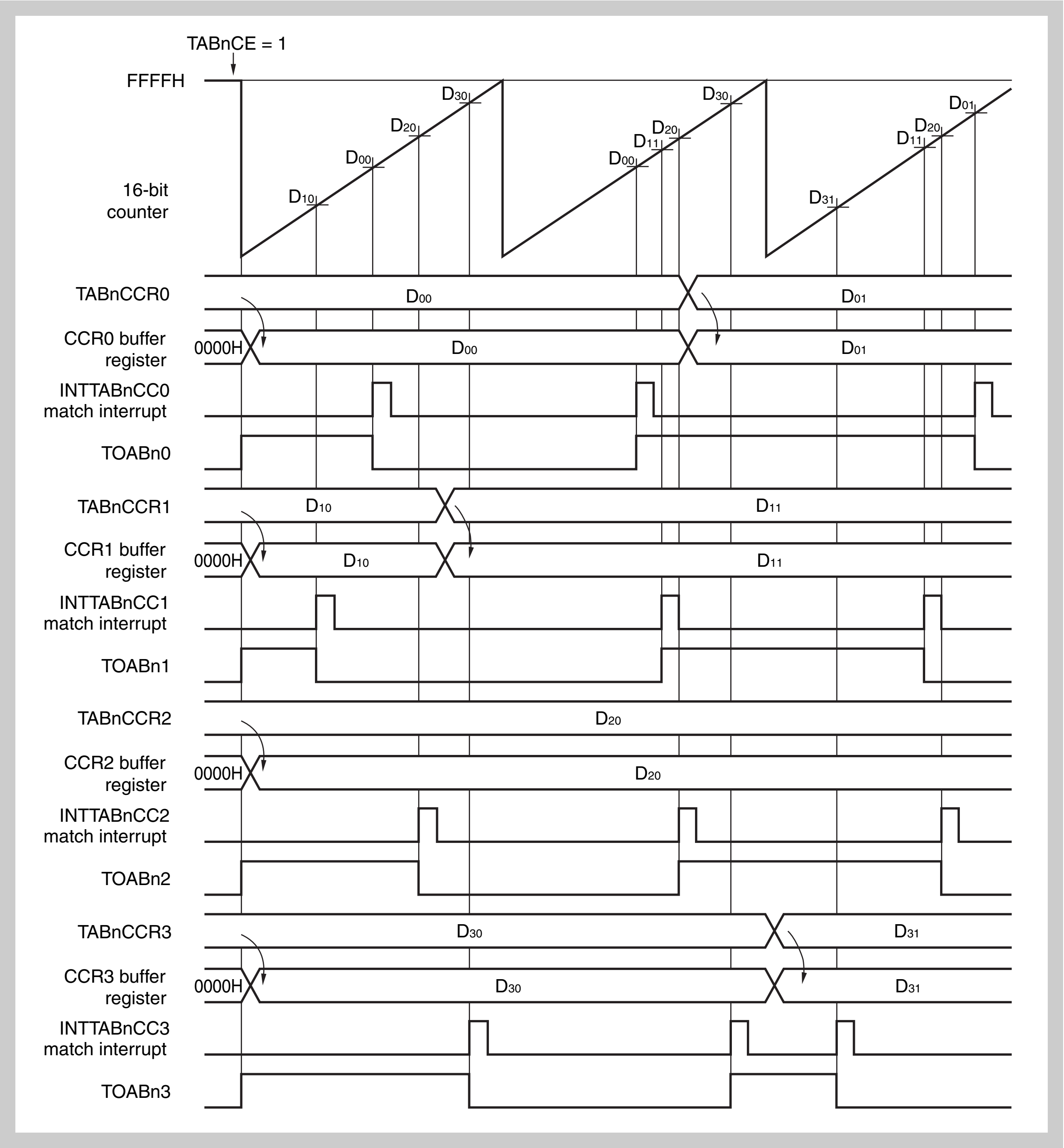


Figure 12-21 Basic operation timing in free-running mode (1/4) (TABnCCS3 = 0, TABnCCS2 = 0, TABnCCS1 = 0, TABnCCS0 = 0)

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

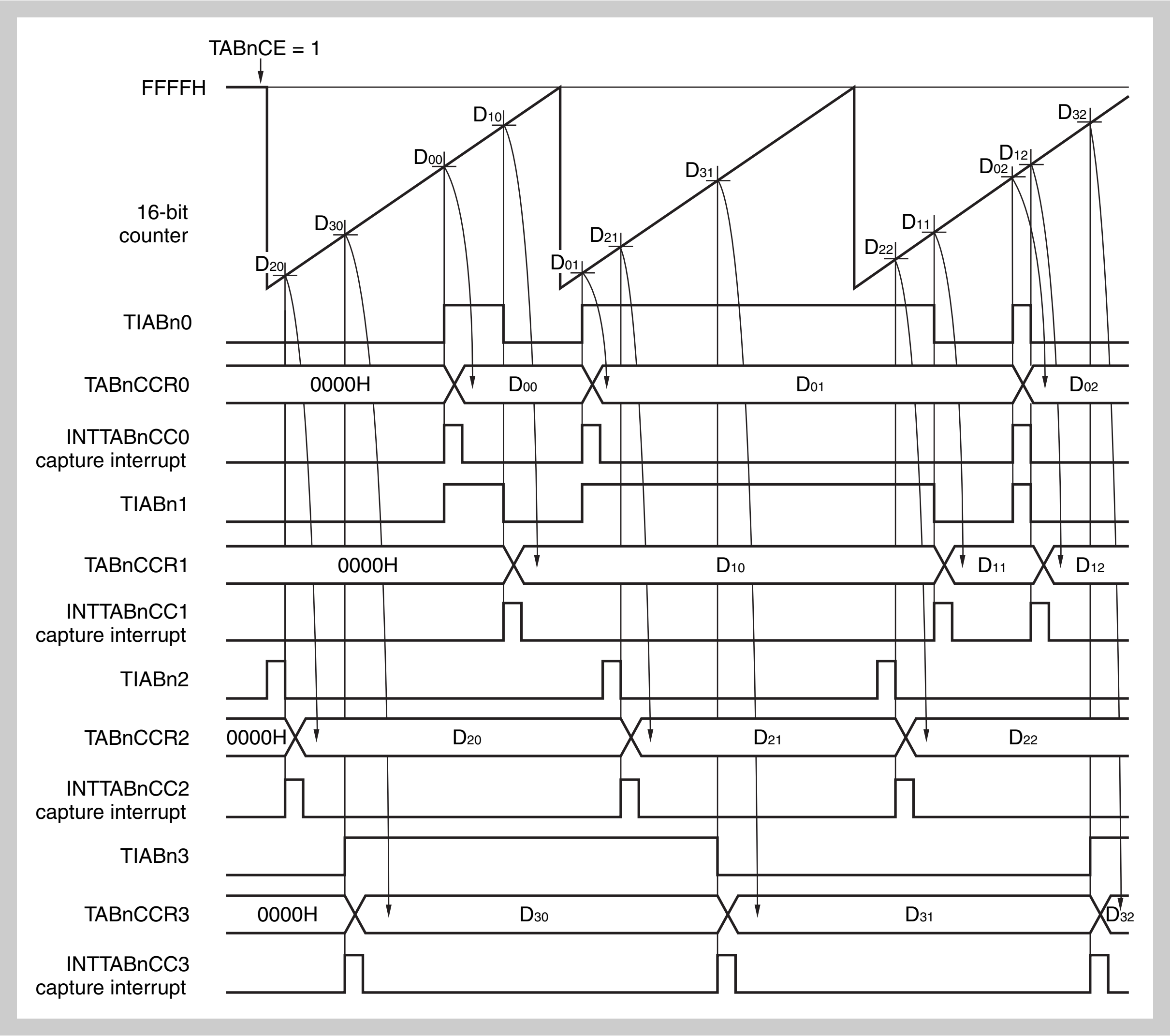


Figure 12-22 Basic operation timing in free-running mode (2/4)
(TABnCCS3 = 1, TABnCCS2 = 1, TABnCCS1 = 1, TABnCCS0 = 1)

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)

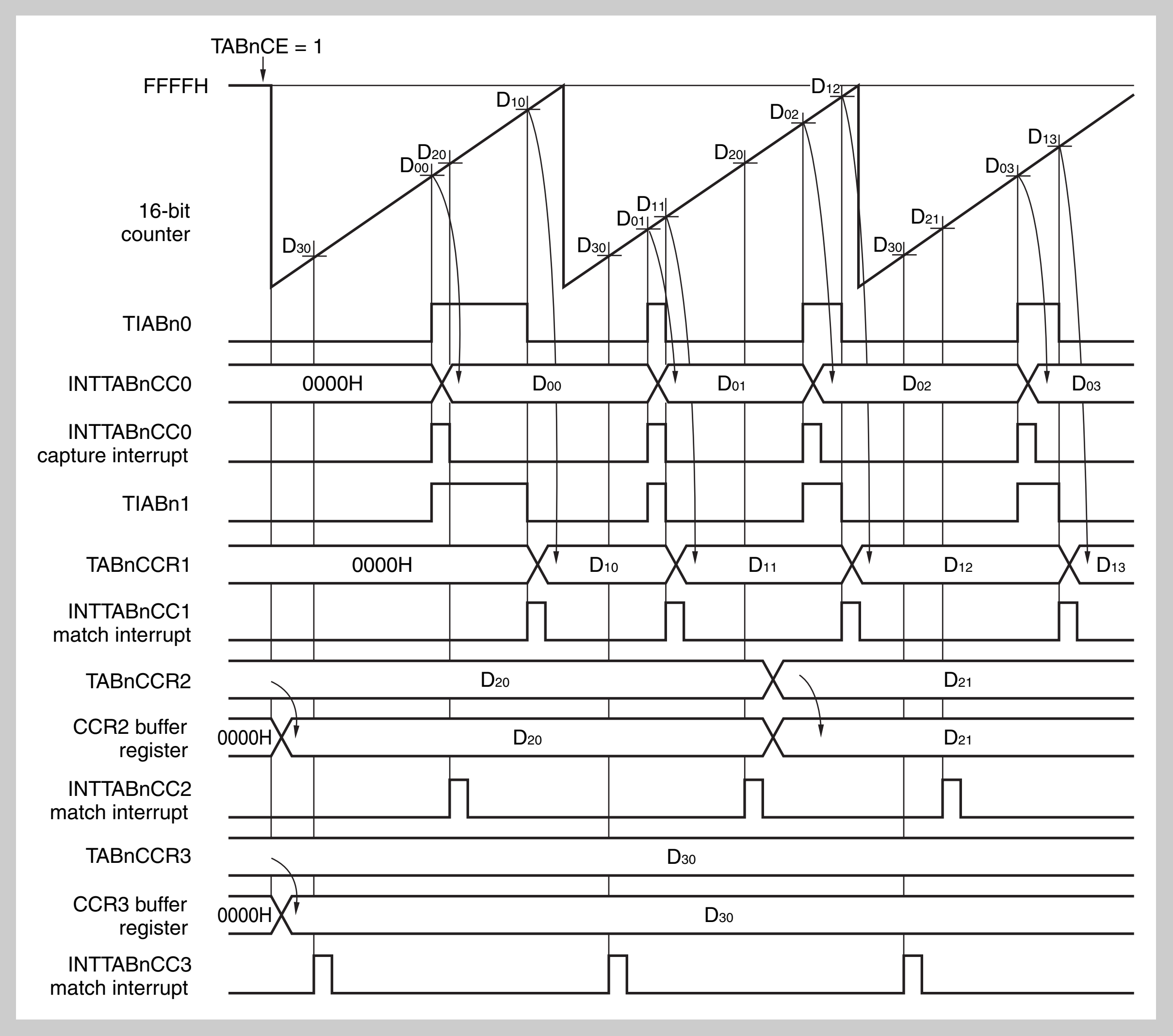


Figure 12-23 **Basic operation timing in free-running mode (3/4)**
(TABnCCS3 = 1, TABnCCS2 = 1, TABnCCS1 = 1, TABnCCS0 = 0)

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)

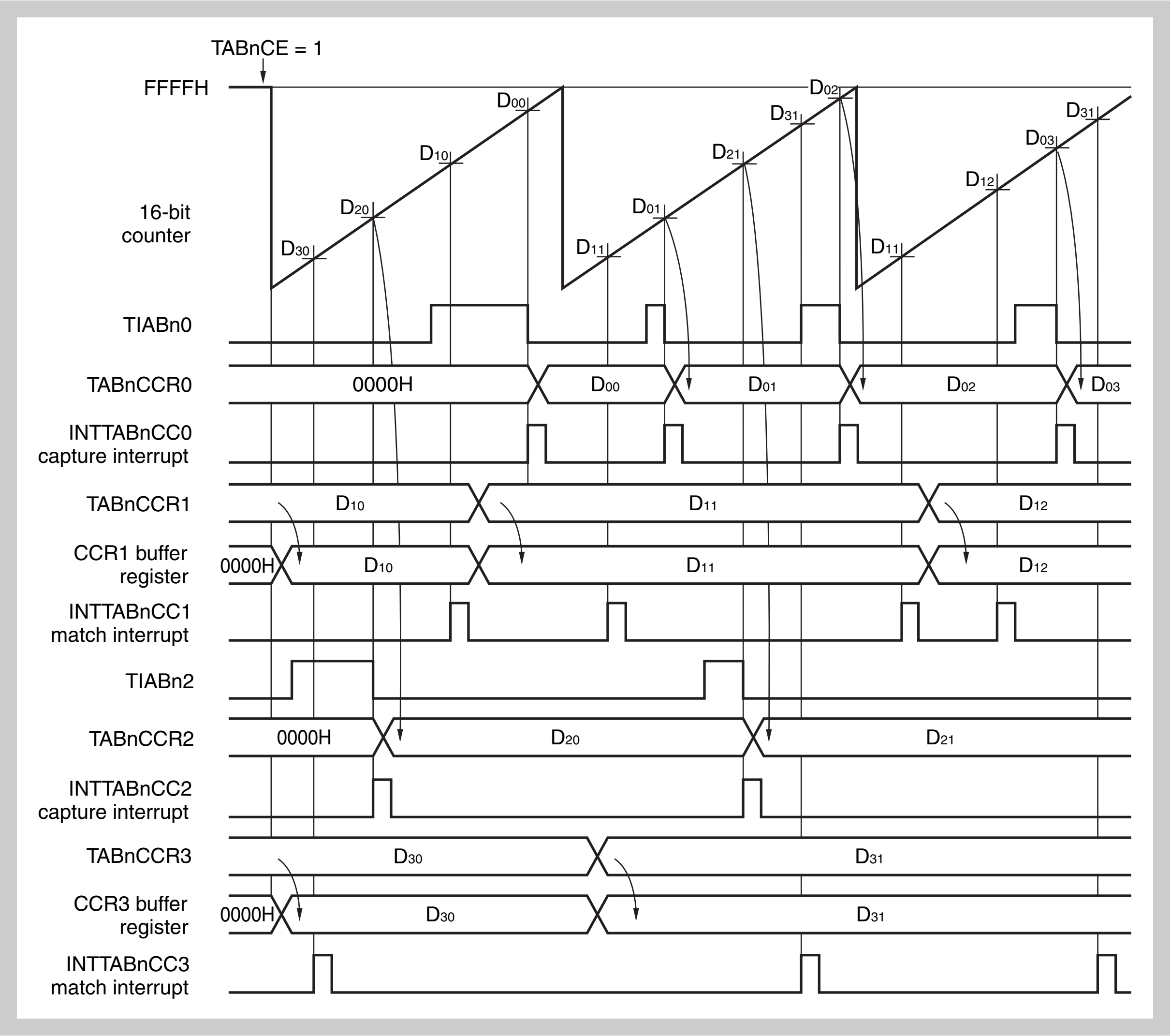


Figure 12-24 Basic operation timing in free-running mode /4/4) (TABnCCS3 = 0, TABnCCS2 = 1, TABnCCS1 = 0, TABnCCS0 = 1)

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.

12.5.8 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 (TABnCTL1.TABnEEE = 0) as a count clock.

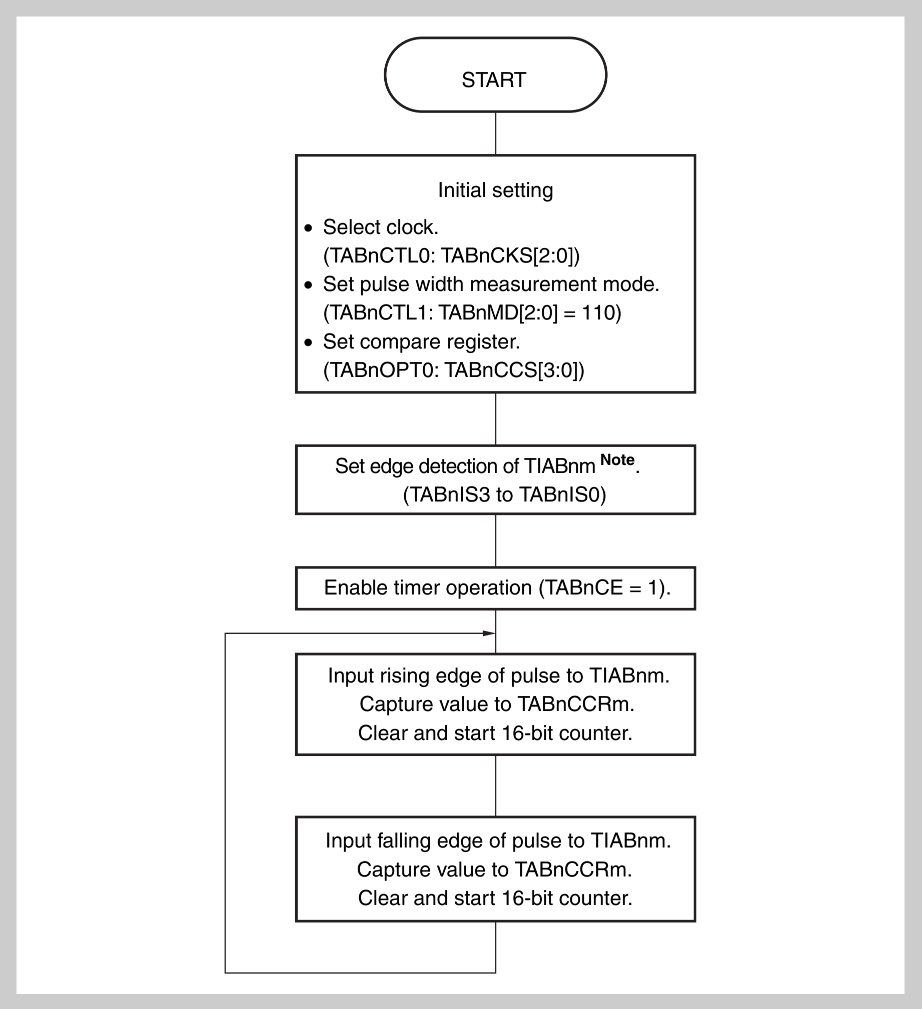

START
Initial setting
• Select clock.
(TABnCTL0: TABnCKS[2:0])
• Set pulse width measurement mode.
(TABnCTL1: TABnMD[2:0] = 110)
• Set compare register.
(TABnOPT0: TABnCCS[3:0])
Set edge detection of TIABnm Note.
(TABnIS3 to TABnIS0)
Enable timer operation (TABnCE = 1).
Input rising edge of pulse to TIABnm.
Capture value to TABnCCRm.
Clear and start 16-bit counter.
Input falling edge of pulse to TIABnm.
Capture value to TABnCCRm.
Clear and start 16-bit counter.

Figure 12-25 Flowchart of basic operation in pulse width measurement mode

Note
1. 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.
2. m = 0 to 3

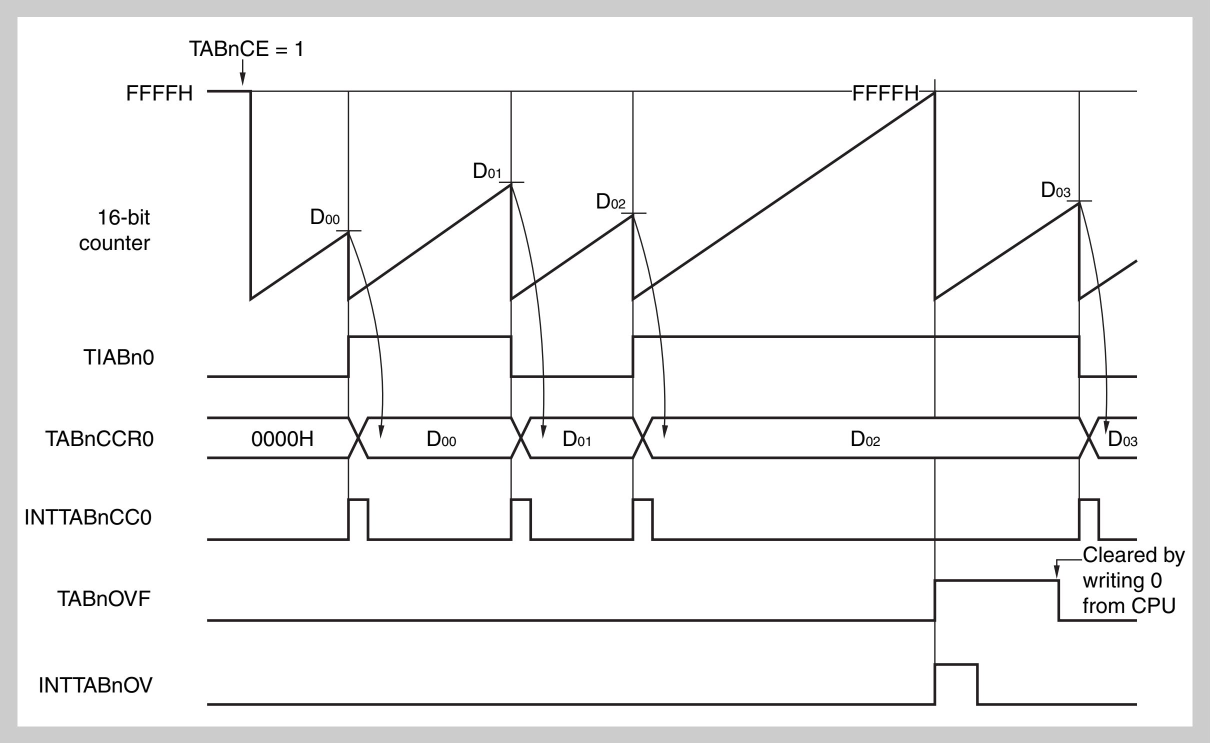


Figure 12-26 Basic operation timing in pulse width measurement mode

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 13 16-Bit Interval Timer M

The microcontroller includes a 16-bit interval Timer M (TMM0).

13.1 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 × 8
- 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

13.2 Configuration

TMM consists of the following hardware.

Table 13-1 Configuration of TMM

Item	Configuration
Timer register	16-bit counter
Register	TMM0 compare register 0 (TM0CMP0)
Control register	TMM0 control register 0 (TM0CTL0)

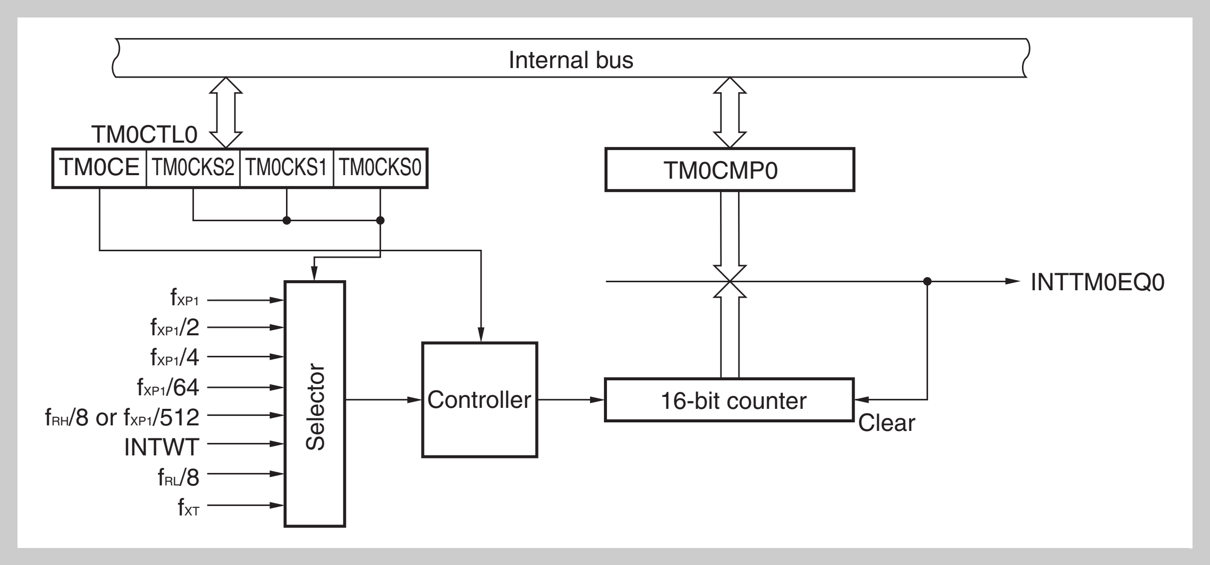


Figure 13-1 Block diagram of Timer M

13.3 Timer M Registers

(1) TM0CMP0 - TMM0 compare register 0

The TM0CMP0 register is a 16-bit compare register.

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

Address FFFFF694$_H$

Initial Value 0000$_H$. This registers is cleared by any reset, or if the internal operation clock is disabled by TM0CTL0.TM0nCE = 0.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TM0CMP0	Compare value 0															
	R/W															

Caution Changing the TM0CMP0 register contents is prohibited while the timer is operating (TM0CE = 1). Thus rewriting with the same value is permitted.

(2) TM0CTL0 - TMM0 control register 0

The TM0CTL0 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 FFFFF690$_H$

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

	7	6	5	4	3	2	1	0
TM0CTL0	TM0CE	0	0	0	0	TM0CKS2	TM0CKS1	TM0CKS0
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Caution

1. Changing the TM0CTL0.TM0CKS[2:0] bits is prohibited while the timer is operating (TM0CE = 1). Thus rewriting of these bits with the same value is permitted.
   The TM0CE bit can be changed at any time.
2. When writing to TM0CTL0 register, bits 6 to 3 must be set to 0.

Table 13-2 TM0CTL0 register contents (1/2)

Bit position	Bit name	Function
7	TM0CE	Controls the timer TM0 operation. 0: Disable internal operating clock operation (asynchronously reset TMM0). 1: Enable internal operating clock operation. The TM0CE bit controls the internal operating clock and asynchronously reset of TMM0. When this bit is cleared to 0, the internal operating clock of TMM is stopped, and TMM0 is asynchronously reset. When the TM0CE bit is set to 1, the internal operating clock is enabled within two input clocks, and the timer counts up.

Table 13-2 TM0CTL0 register contents (2/2)

Bit position	Bit name	Function
2 to 0	TM0CKS [2:0]	Selects the count clock of timer TM0.

SELCNT0. SEL07[a]	TM0CKS2	TM0CKS1	TM0CKS0	Selection of internal count clock		
				Input	PRSI =	
					0	1
×	0	0	0	f_{XP1}	f_{XX}	$f_{XX}/2$
×	0	0	1	$f_{XP1}/2$	$f_{XX}/2$	$f_{XX}/4$
×	0	1	0	$f_{XP1}/4$	$f_{XX}/4$	$f_{XX}/8$
×	0	1	1	$f_{XP1}/64$	$f_{XX}/64$	$f_{XX}/128$
0	1	0	0	$f_{XP1}/512$	$f_{XX}/512$	$f_{XX}/1024$
1				$f_{RH}/8$		
×	1	0	1	INTWT		
×	1	1	0	$f_{RL}/8$		
×	1	1	1	f_{XT}		

[a] Refer to chapter *“Selector control registers” on page 209* for details of SELCNT0 register.

Note: **1.** PRSI can be set by the option bytes (refer to *“Flash Mask Options” on page 330* for details.):

- PRSI = 0: $f_{XX} \leqq 32$ MHz
- PRSI = 1: 32 MHz < $f_{XX} \leqq 48$ MHz

2. f_{XX}: Main system clock frequency
f_{RL}: Low frequency internal oscillator clock frequency (240 KHz)
f_{RH}: High frequency internal oscillator clock frequency (8 MHz)
f_{XT}: Sub oscillator frequency

13.4 Operation

13.4.1 Interval timer mode

In the interval timer mode, a match interrupt signal (INTTM0EQ0) is output when the value of the 16-bit counter matches the value of TMM0 compare register 0 (TM0CMP0). At the same time, the counter is cleared to 0000_H and starts counting up.

When $FFFF_H$ is set to the TM0CMP0 register, Timer M performs an operation similar to that in the free-running mode.

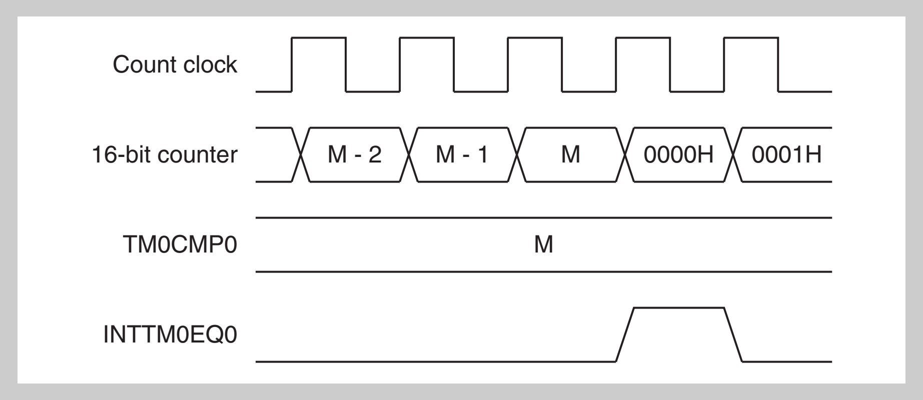


Figure 13-2 Timing of operation in interval timer mode

Caution To set M clocks as the interval period, set the TM0CMP0 register to M – 1.

13.4.2 Cautions

(1) Clock Generator and clock enable timing

Because the second clock is the first pulse of the timer count-up signal when the TM0CE bit is changed from 0 to 1, the timer counts one clock less.

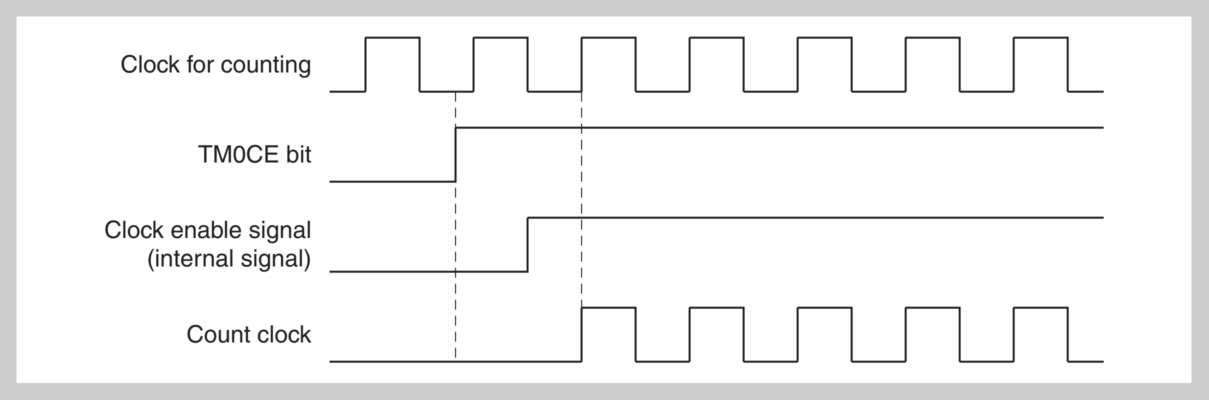


Figure 13-3 Count operation start timing

Chapter 14 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 14-1*) start and clock synchronously. When the master timer is cleared, the slave timers are cleared synchronously, too.

Table 14-1 Synchronous operation mode of timers

Master timer	Slave timer	V850ES/FE3	V850ES/FF3	V850ES/FG3	V850ES/FJ3	V850ES/FK3
TAA0	TAA1	√	√	√	√	√
TAA2	TAA3	√	√	√	√	√
TAB0	TAA4	√	√	√	√	√
TAB1	TAB2	–	–	–	√	√
TAA5	TAA6	–	–	–	–	√
	TAA7	–	–	–	–	√

Setup In the following the procedure is described how to set up the master and slave timer for synchronous operation. Exemplarily 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.TABmCE = 1: enable operation

Table 14-2 and *Table 14-3* show the timer modes that can be used in the synchronous operation mode.

Table 14-2 Timer modes usable in synchronous operation mode

Master timer	Slave timer	Free-running mode	PWM mode	Triangular wave PWM mode
TAA0	TAA1	√	√	×
TAA2	TAA3	√	√	×
TAB0	TAA4	√	√	×
TAB1	TAB2	√	√	√
TAA5	TAA6	√	√	×
	TAA7	√	√	×

Table 14-3 Timer output functions

Synch channel	Timer	Pin	Free-running mode (compare function)		PWM mode		Triangular wave PWM mode	
			Synch OFF	Synch ON	Synch OFF	Synch ON	Synch OFF	Synch ON
Ch0	TAA0 (master)	TOAA00	Toggle	Toggle	Toggle	Toggle	N/A	N/A
		TOAA01	Toggle	Toggle	PWM	PWM	N/A	N/A
	TAA1 (slave)	TOAA10	Toggle	Toggle	Toggle	PWM	N/A	N/A
		TOAA11	Toggle	Toggle	PWM	PWM	N/A	N/A
Ch1	TAA2 (master)	TOAA20	Toggle	Toggle	Toggle	Toggle	N/A	N/A
		TOAA21	Toggle	Toggle	PWM	PWM	N/A	N/A
	TAA3 (slave)	TOAA30	Toggle	Toggle	Toggle	PWM	N/A	N/A
		TOAA31	Toggle	Toggle	PWM	PWM	N/A	N/A
Ch2	TAB0 (master)	TOAB00	Toggle	Toggle	Toggle	Toggle	N/A	N/A
		TOAB01 to TOAB03	Toggle	Toggle	PWM	PWM	N/A	N/A
	TAA4 (slave)	TOAA40	Toggle	Toggle	Toggle	PWM	N/A	N/A
		TOAA41	Toggle	Toggle	PWM	PWM	N/A	N/A
Ch3	TAB1 (master)	TOAB10	Toggle	Toggle	Toggle	Toggle	Toggle	Toggle
		TOAB11 to TOAB13	Toggle	Toggle	PWM	PWM	Triangular wave PWM	Triangular wave PWM
	TAB2 (slave)	TOAB20	Toggle	Toggle	Toggle	PWM	Toggle	Triangular wave PWM
		TOAB21 to TOAB23	Toggle	Toggle	PWM	PWM	Triangular wave PWM	Triangular wave PWM
Ch4	TAA5 (master)	TOAA50	Toggle	Toggle	Toggle	Toggle	N/A	N/A
		TOAA51	Toggle	Toggle	PWM	PWM	N/A	N/A
	TAA6 (slave 1)	TOAA60	Toggle	Toggle	Toggle	PWM	N/A	N/A
		TOAA61	Toggle	Toggle	PWM	PWM	N/A	N/A
	TAA7 (slave 2)	TOAA70	Toggle	Toggle	Toggle	PWM	N/A	N/A
		TOAA71	Toggle	Toggle	PWM	PWM	N/A	N/A

The timing of transmitting data from the TAAnCCRm/TABnCCRm compare register to the CCRm registers is as follows:

Free-Running mode:	Timing at which the CPU writes the registers (anytime write).
Triangular wave PWM mode, PWM mode:	Timing at which timer counter CCR0 and compare register TOAAn0/TABn0 of master timer match.

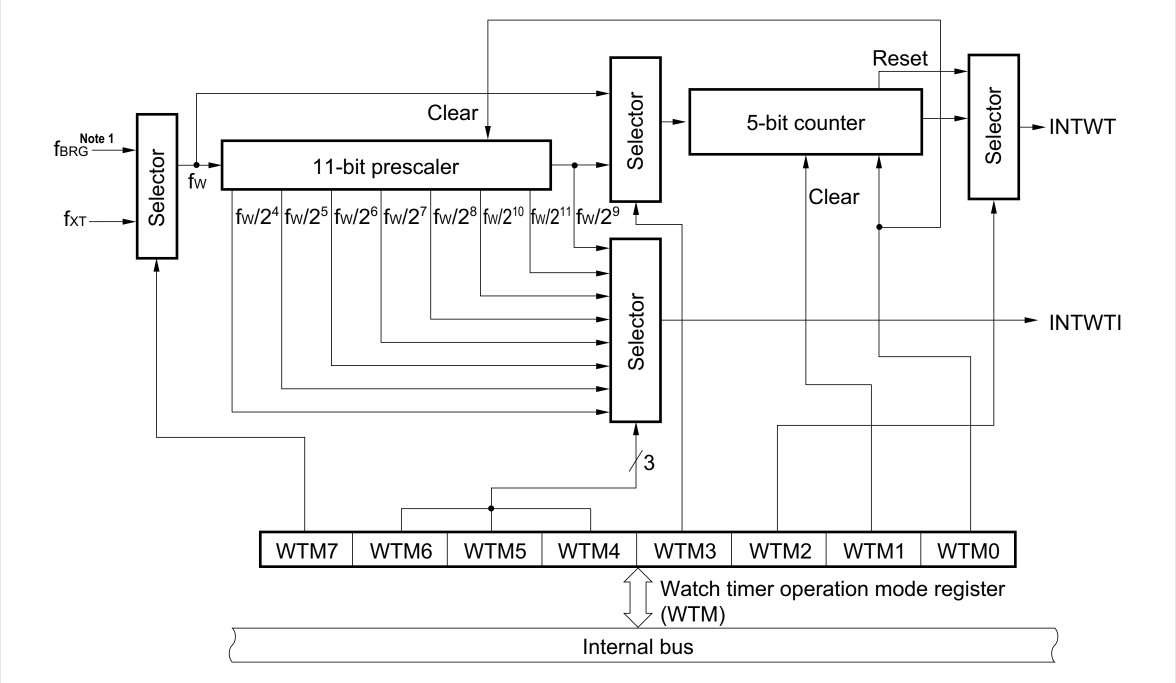
Chapter 15 Watch Timer Functions

15.1 Functions

The Watch Timer has the following functions.

- Watch Timer
- Interval timer

The Watch Timer and interval timer functions can be used at the same time.



Reset
5-bit counter
Selector
INTWT
Clear
Note 1
fBRG
Selector
fW
11-bit prescaler
Selector
fXT
fW/2^4 fW/2^5 fW/2^6 fW/2^7 fW/2^8 fW/2^10 fW/2^11 fW/2^9
Clear
Selector
INTWTI
3
WTM7 WTM6 WTM5 WTM4 WTM3 WTM2 WTM1 WTM0
Watch timer operation mode register (WTM)
Internal bus

Figure 15-1 Block diagram of Watch Timer

Note

1. The Prescaler3 output f_{BRG} is also used for CSIB0.
   For details refer to *"Clock Generator" on page 179*.
2. f_{XT}: Sub oscillator frequency
   f_W: Watch Timer clock frequency
   INTWT: Watch Timer interrupt
   INTWTI: Interval timer interrupt

(1) Watch Timer

The Watch Timer generates interrupt requests (INTWT) at time intervals of 0.5 or 0.25 seconds by using the Sub oscillator (nominal f_{XT} = 32.768 KHz).

Caution When using a clock f_{BRG} obtained by dividing the main clock f_X by Prescaler3 as the Watch Timer count clock f_W, set the PRSM0 and PRSCM0 registers according to the main clock frequency that is used so as to obtain a divided clock frequency of 32.768 KHz.
If 32.768 KHz cannot be generated, a clock correction software is necessary to realize the watch function.

(2) Interval timer

The Watch Timer generates an interrupt request (INTWTI) at time intervals specified in advance.

Table 15-1 Interval time of interval imer

Interval Time	Operation at $f_W = f_{XT}$ = 32.768 KHz
$2^4 \times 1/f_W$	488 µs
$2^5 \times 1/f_W$	977 µs
$2^6 \times 1/f_W$	1.95 ms
$2^7 \times 1/f_W$	3.91 ms
$2^8 \times 1/f_W$	7.81 ms
$2^9 \times 1/f_W$	15.6 ms
$2^{10} \times 1/f_W$	31.2 ms
$2^{11} \times 1/f_W$	62.5 ms

Note f_W: Watch Timer clock frequency
f_{XT}: Sub oscillator frequency

15.2 Configuration

The Watch Timer consists of the following hardware.

Table 15-2 Configuration of Watch Timer

Item	Configuration
Counter	5 bits × 1
Prescaler	11 bits × 1
Control register	Watch Timer operation mode register (WTM)

15.3 Control Registers

The Watch Timer operation mode register (WTM) controls the Watch Timer. Before operating the Watch Timer, set the count clock and the interval time.

(1) WTM - Watch Timer operation mode register

The WTM register enables or disables the count clock and operation of the Watch Timer, sets the interval time of the prescaler, controls the operation of the 5-bit counter, and sets the set time of the watch flag.

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

Address $FFFFF680_H$

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

	7	6	5	4	3	2	1	0
WTM	WTM7	WTM6	WTM5	WTM4	WTM3	WTM2	WTM1	WTM0
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Caution Rewrite the WTM[7:2] bits while both the WTM0 and WTM1 bits are 0.

Table 15-3 TAAnCTL1 register contents (1/2)

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>7 to 4</td><td>WTM[7:4]</td><td>Selects the watch timer interrupt time.
<table>
<tr><th>WTM7</th><th>WTM6</th><th>WTM5</th><th>WTM4</th><th>Watch timer interrupt time</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>$2^4/f_W$ (488 µs: $f_W = f_{XT}$)</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>$2^5/f_W$ (977µs: $f_W = f_{XT}$)</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>$2^6/f_W$ (1.95 ms: $f_W = f_{XT}$)</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>$2^7/f_W$ (3.91 ms: $f_W = f_{XT}$)</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td><td>$2^8/f_W$ (7.81 ms: $f_W = f_{XT}$)</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1</td><td>$2^9/f_W$ (15.63 ms: $f_W = f_{XT}$)</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td><td>$2^{10}/f_W$ (31.25 ms: $f_W = f_{XT}$)</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>1</td><td>$2^{11}/f_W$ (62.5 ms: $f_W = f_{XT}$)</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</td><td>$2^4/f_W$ (488 µs: $f_W = f_{BRG}$)</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>1</td><td>$2^5/f_W$ (977µs: $f_W = f_{BRG}$)</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>0</td><td>$2^6/f_W$ (1.95 ms: $f_W = f_{BRG}$)</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>1</td><td>$2^7/f_W$ (3.91 ms: $f_W = f_{BRG}$)</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>0</td><td>$2^8/f_W$ (7.81 ms: $f_W = f_{BRG}$)</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>1</td><td>$2^9/f_W$ (15.63 ms: $f_W = f_{BRG}$)</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>0</td><td>$2^{10}/f_W$ (31.25 ms: $f_W = f_{BRG}$)</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>$2^{11}/f_W$ (62.5 ms: $f_W = f_{BRG}$)</td></tr>
</table>
</td></tr>
</table>

Table 15-3 TAAnCTL1 register contents (2/2)

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>7, 3, 2</td><td>WTM7,
WTM[3:2]</td><td>Selects the set time of watch flag.
<table>
<tr><th>WTM7</th><th>WTM3</th><th>WTM3</th><th>Set time of watch flag</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>$2^{14}/f_W$ (0.5 s: $f_W = f_{XT}$)</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>$2^{13}/f_W$ (0.25 s: $f_W = f_{XT}$)</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>$2^{5}/f_W$ (977μs: $f_W = f_{XT}$)</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>$2^{4}/f_W$ (488 μs: $f_W = f_{XT}$)</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>$2^{14}/f_W$ (0.5 s: $f_W = f_{BRG}$)</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>$2^{13}/f_W$ (0.25 s: $f_W = f_{BRG}$)</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>$2^{5}/f_W$ (977μs: $f_W = f_{BRG}$)</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>$2^{4}/f_W$ (488 μs: $f_W = f_{BRG}$)</td></tr>
</table>
</td></tr>
<tr><td>1</td><td>WTM1</td><td>Controls the 5-bit counter operation.
0: Clears counter after operation stops.
1: Starts counter.</td></tr>
<tr><td>0</td><td>WTM0</td><td>Controls watch timer operation.
0: Stops operation (clears both, prescaler and 5-bit counter)
1: Enables operation</td></tr>
</table>

Note **1.** f_W: f_{XT}: Sub oscillator frequency
f_{BRG}: Prescaler3 output frequency
f_W: Watch timer clock frequency

2. Values in parentheses apply to operation with f_W = 32.768 KHz

15.4 Operation

15.4.1 Operation as Watch Timer

The Watch Timer generates an interrupt request at fixed time intervals. The Watch Timer operates using time intervals of 0.5 or 0.25 seconds with the Sub oscillator (32.768 KHz).

The count operation starts when the WTM[1:0] bits are set to 11_B. When the WTM0 bit is cleared to 0, the 11-bit prescaler and 5-bit counter are cleared and the count operation stops.

The time of the Watch Timer can be adjusted by clearing the WTM1 bit to 0 and then the 5-bit counter. At this time, an error of up to 15.6 ms may occur.

The interval timer may be cleared by clearing the WTM0 bit to 0. However, because the 5-bit counter is cleared at the same time, an error of up to 0.5 seconds may occur when the Watch Timer overflows (INTWT).

15.4.2 Operation as interval timer

The Watch Timer can also be used as an interval timer that repeatedly generates an interrupt at intervals specified by a preset count value.

The interval time can be selected by the WTM[7:4] bits.

Table 15-4 Interval time of itimer

WTM7	WTM6	WTM5	WTM4	Interval Time	
0	0	0	0	$2^4 \times 1/f_w$	488 µs (operating at $f_W = f_{XT} = 32.768$ KHz)
0	0	0	1	$2^5 \times 1/f_w$	977 µs (operating at $f_W = f_{XT} = 32.768$ KHz)
0	0	1	0	$2^6 \times 1/f_w$	1.95 ms (operating at $f_W = f_{XT} = 32.768$ KHz)
0	0	1	1	$2^7 \times 1/f_w$	3.91 ms (operating at $f_W = f_{XT} = 32.768$ KHz)
0	1	0	0	$2^8 \times 1/f_w$	7.81 ms (operating at $f_W = f_{XT} = 32.768$ KHz)
0	1	0	1	$2^9 \times 1/f_w$	15.6 ms (operating at $f_W = f_{XT} = 32.768$ KHz)
0	1	1	0	$2^{10} \times 1/f_w$	31.3 ms (operating at $f_W = f_{XT} = 32.768$ KHz)
0	1	1	1	$2^{11} \times 1/f_w$	62.5 ms (operating at $f_W = f_{XT} = 32.768$ KHz)
1	0	0	0	$2^4 \times 1/f_w$	488 µs (operating at $f_W = f_{BRG} = 32.768$ KHz)
1	0	0	1	$2^5 \times 1/f_w$	977 µs (operating at $f_W = f_{BRG} = 32.768$ KHz)
1	0	1	0	$2^6 \times 1/f_w$	1.95 ms (operating at $f_W = f_{BRG} = 32.768$ KHz)
1	0	1	1	$2^7 \times 1/f_w$	3.91 ms (operating at $f_W = f_{BRG} = 32.768$ KHz)
1	1	0	0	$2^8 \times 1/f_w$	7.81 ms (operating at $f_W = f_{BRG} = 32.768$ KHz)
1	1	0	1	$2^9 \times 1/f_w$	15.6 ms (operating at $f_W = f_{BRG} = 32.768$ KHz)
1	1	1	0	$2^{10} \times 1/f_w$	31.3 ms (operating at $f_W = f_{BRG} = 32.768$ KHz)
1	1	1	1	$2^{11} \times 1/f_w$	62.5 ms (operating at $f_W = f_{BRG} = 32.768$ KHz)

Note fw: Watch Timer clock frequency
f_{XT}: Sub oscillator frequency
f_{BRG}: Prescaler3 output frequency

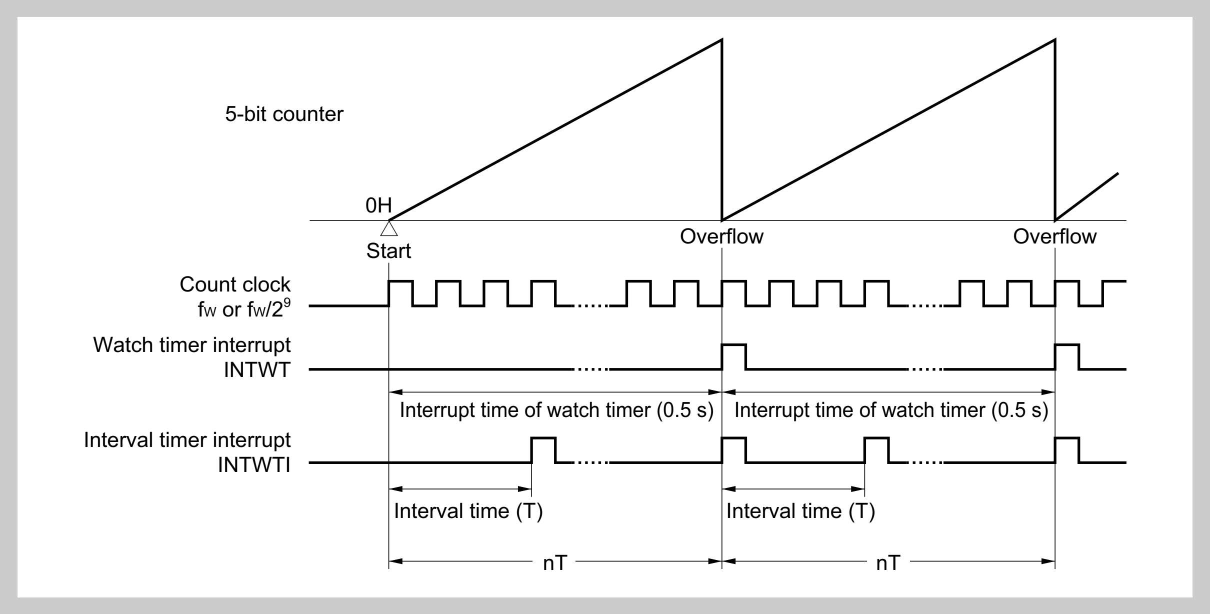


Figure 15-2 Operation Timing of Watch Timer/Interval Timer

Note

1. fw: Watch Timer clock frequency
2. Values in parentheses apply to operation with count clock fw = 32.768 KHz.
3. n: Number of interval timer operations

15.4.3 Cautions

The following time is required before the first Watch Timer interrupt request signal (INTWT) is generated after operation is enabled (WTM1 and WTM0 bits of WTM register = 1).

It takes 0.515625 seconds for the first INTWT signal to be generated
(2^9 × 1/32768 = 0.015625 s longer).
The INTWT signal is then generated every 0.5 seconds.

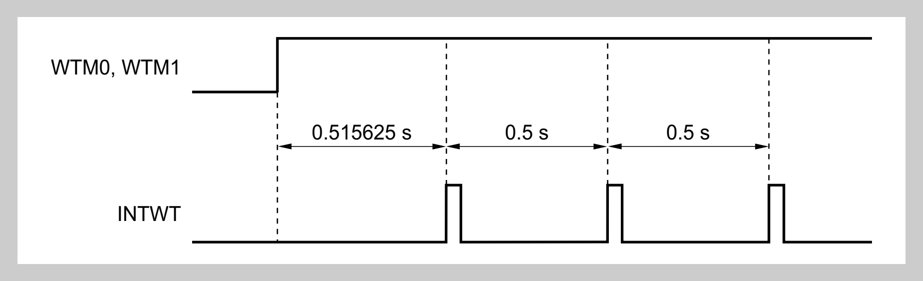


Figure 15-3 Example of generation of Watch Timer interrupt request signal (INTWT) (when interrupt period = 0.5 s)

Chapter 16 Watchdog Timer 2

16.1 Functions

Watchdog Timer 2 has the following functions.

- Default-start Watchdog Timer
- Reset mode:
  Reset operation upon overflow of Watchdog Timer 2 (generation of WDT2RES signal)
- Non-maskable interrupt request mode:
  NMI operation upon overflow of Watchdog Timer 2 (generation of INTWDT2 signal)
- Input selectable from main clock and 240 KHz internal oscillator as the source clock

Caution

1. Watchdog Timer 2 is automatically started after reset release. Source clock is a 240 KHz internal oscillator.
2. By flash mask option, operation of WDT2 can be set fixed to 240 KHz internal oscillator source clock and reset mode. Only the interval time can be changed. Changing of clock source and operation mode is not possible.
3. In case WDT2 shall not be used or clock source and operation mode shall be changed, flash mask option should not be set for fixing 240 KHz internal oscillator source clock and reset mode.
   In this case, after reset, the settings should be changed before the first WDT2 overflow. Alternatively WDT2 should be cleared once, and required changes should be performed within the next interval time.
4. The WDTM2 register can be written only once after reset. Even if the default setting of WDTM2 shall not be changed, it is recommended to once write the default value to WDTM2 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 INTWDT2. Therefore a system reset must be performed after completion of the INTWDT2 service routine.

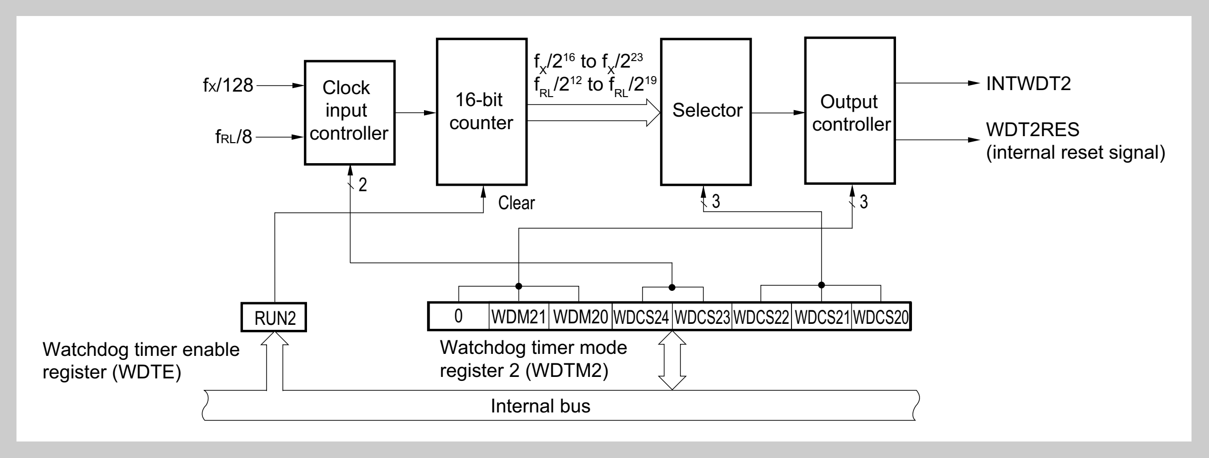


Figure 16-1 Block diagram of Watchdog Timer 2

Note		
	f_X:	Oscillation frequency
	f_{RL}:	Internal-OSC clock frequency
	INTWDT2:	Non-maskable interrupt request signal from Watchdog Timer 2
	WDT2RES:	Watchdog Timer 2 reset signal

16.2 Configuration

Watchdog Timer 2 consists of the following hardware.

Table 16-1 Configuration of Watchdog Timer 2

Item	Configuration
Control registers	Watchdog Timer mode register 2 (WDTM2) Watchdog Timer enable register (WDTE)

16.3 Control Registers

(1) WDTM2 - Watchdog Timer 2 mode register

The WDTM2 register sets the operation mode, operation clock and overflow time of Watchdog Timer 2.

Access The register can be read/written in 1-bit and 8-bit units.
This register can be read any number of times, but it can be written only once following reset release.

Address FFFF F6D0$_H$

Initial Value 67$_H$. The register is initialized by any reset.

	7	6	5	4	3	2	1	0
WDTM2	0	WDM21	WDM20	WDCS24	WDCS23	WDCS22	WDCS21	WDCS20
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Caution

1. If the WDTM2 register is rewritten twice after reset, an overflow signal is forcibly generated. If the Watchdog Timer has stopped operation, WDTM2 can be written several times without generating an overflow.
2. To stop WDT2 securely,
   - stop the internal oscillator by RCM.RSTOP = 1 (must be permitted by flash mask options)
   - set WDTM2 = 1F$_H$
3. In order to ensure that the Watchdog Timer does not overflow, and thus generate a watchdog event, during the register settings are changed, write to WDTE first for restarting the timer.

Table 16-2 WDTM2 register contents (1/2)

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>6,5</td><td>WDM2[1:0]</td><td>Selects the operation mode of watchdog timer 2.
<table>
<tr><th>WDM21</th><th>WDM20</th><th>Operation mode of watchdog timer 2</th></tr>
<tr><td>0</td><td>0</td><td>Stops operation</td></tr>
<tr><td>0</td><td>1</td><td>Non-maskable interrupt request mode (generation of INTWDT2)</td></tr>
<tr><td>1</td><td>×</td><td>Reset mode (generation of RESWDT2)</td></tr>
</table></td></tr>
</table>

Table 16-2 WDTM2 register contents (2/2)

Bit position	Bit name	Function
4 to 0	WDCS2 [4:0]	Selects the count clock of watchdog timer 2.

WDCS24	WDCS23	WDCS22	WDCS21	WDCS20	Selected clock period	240 KHz (typ.)	
0	0	0	0	0	$2^{12}/f_{RL}$	17.1 ms	
0	0	0	0	1	$2^{13}/f_{RL}$	34.1 ms	
0	0	0	1	0	$2^{14}/f_{RL}$	68.3 ms	
0	0	0	1	1	$2^{15}/f_{RL}$	136.5 ms	
0	0	1	0	0	$2^{16}/f_{RL}$	273.1 ms	
0	0	1	0	1	$2^{17}/f_{RL}$	546.1 ms	
0	0	1	1	0	$2^{18}/f_{RL}$	1,092.3 ms	
0	0	1	1	1	$2^{19}/f_{RL}$ (default)	2,184.5 ms	
						f_X = 4 MHz	f_X = 16 MHz
0	1	0	0	0	$2^{16}/f_X$	16.4 ms	4.1 ms
0	1	0	0	1	$2^{17}/f_X$	32.8 ms	8.2 ms
0	1	0	1	0	$2^{18}/f_X$	65.5 ms	16.4 ms
0	1	0	1	1	$2^{19}/f_X$	131.1 ms	32.8 ms
0	1	1	0	0	$2^{20}/f_X$	262.1 ms	65.3 ms
0	1	1	0	1	$2^{21}/f_X$	524.3 ms	131.1 ms
0	1	1	1	0	$2^{22}/f_X$	1,048.6 ms	262.2 ms
0	1	1	1	1	$2^{23}/f_X$	2,097.2 ms	524.3 ms
1	×	×	×	×	Stop		

(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 $F6D1_H$

Initial Value $9A_H$. The register is initialized by any reset.

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

Caution

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).

16.4 Watchdog Timer Operation

Watchdog Timer 2 automatically starts in the reset mode after reset is released.

The WDTM2 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 WDTM2 register using an 8-bit memory manipulation instruction. After this, the operation of watchdog timer 2 cannot be stopped/changed again.

The WDCS24 to WDCS20 bits of the WDTM2 register are used to select the watchdog timer 2 loop detection time interval.

Writing AC_H to the WDTE register clears the counter of watchdog timer 2 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 (WDT2RES) or a non-maskable interrupt request signal (INTWDT2) is generated, depending on the set values of the WDM21 and WDM20 bits of the WDTM2 register.

When not using watchdog timer 2, write $1F_H$ to the WDTM2 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 system reset after completion of interrupt servicing.

16.5 Watchdog Timer Operation in Power Save Mode

If the Watchdog Timer overflows while the device is in power save mode, following procedures take place:

- Watchdog Timer in reset operation mode (WDTM2.WDM21 = 1):
  A device RESET is executed.
- Watchdog Timer in NMI operation mode (WDTM2.WDM2[1:0] = 01_B):
  The NMI is not served, the device wakes up from power save mode and continues with normal operation.

Chapter 17 Asynchronous Serial Interface (UARTD)

The V850ES/Fx3 microcontrollers have following instances of the Universal Asynchronous Serial Interface UARTD:

UARTD	V850ES/FE3	V850ES/FF3	V850ES/FG3		V850ES/FJ3		V850ES/FK3
			µPD70F3374 µPD70F3375	µPD70F3376A µPD70F3377A	µPD70F3378	µPD70F3379 µPD70F3380 µPD70F3381 µPD70F3382	
Instances	2		3	5	3	6	8
Names	UARTD0 to UART1		UARTD0 to UART2	UARTD0 to UART4	UARTD0 to UART2	UARTD0 to UART5	UARTD0 to UART7

Throughout this chapter, the individual instances of UARTD are identified by “n”, for example, UDnCTL0 for the UARTDn control register 0.

17.1 Features

- Transfer rate: 300 bps to 1500 kbps (using dedicated baud rate generator)
- Full-duplex communication:
  - Internal UARTD receive data register n (UDnRX)
  - Internal 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.
  - Status 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
- MSB-/LSB-first transfer selectable
- Transmit/receive data inverted input/output possible
- 13 to 20 bits selectable for the SBF (Sync Break Field) in the LIN (Local Interconnect Network) communication format
  - Recognition of 11 bits or more possible for SBF reception in LIN communication format
  - SBF reception flag provided
- SBF reception can be detected during data communication.
- Bus monitor function to keep data consistency of the transmit data
- DMA support

17.2 Configuration

The block diagram of the UARTDn is shown below.

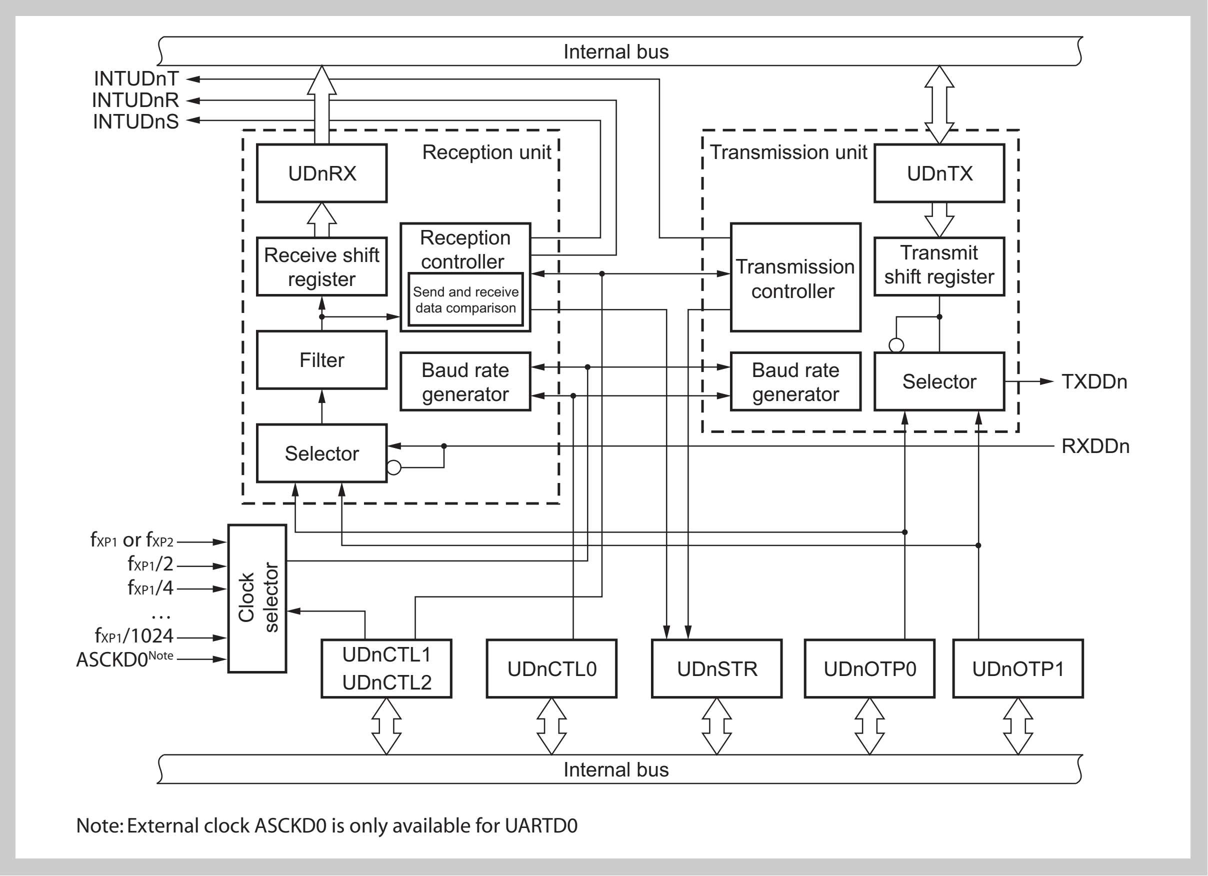


Figure 17-1 Block diagram of Asynchronous Serial Interface UARTDn

Note For the configuration of the baud rate generator, see *Figure 17-11 on page 570*.

UARTDn consists of the following hardware units.

Table 17-1 Configuration of UARTDn

Item	Configuration
Registers	UARTDn control register 0 (UDnCTL0) UARTDn control register 1 (UDnCTL1) UARTDn control register 2 (UDnCTL2) UARTDn option control register 0 (UDnOPT0) UARTDn status register (UDnSTR) UARTDn receive shift register UARTDn receive data register (UDnRX) UARTDn transmit shift register UARTDn transmit data register (UDnTX)

(1) UARTDn control register 0 (UDnCTL0)

The UDnCTL0 register is an 8-bit register used to specify the UARTDn operation.

(2) UARTDn control register 1 (UDnCTL1)

The UDnCTL1 register is an 8-bit register used to select the input clock for the UARTDn.

(3) UARTDn control register 2 (UDnCTL2)

The UDnCTL2 register is an 8-bit register used to control the baud rate for the UARTDn.

(4) UARTDn option control register 0 (UDnOPT0)

The UDnOPT0 register is an 8-bit register used to control the serial transfer for the UARTDn.

(5) UARTDn option control register 1 (UDnOPT1)

The UDnOPT1 register is an 8-bit register used to control the serial transfer for the UARTDn.

(6) UARTDn status register (UDnSTR)

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.

(7) 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.

(8) UARTDn receive data register (UDnRX)

The UDnRX register is an 8-bit register that holds receive data. When 7 characters are received, 0 is stored in the highest bit (when data is received LSB first).

In the reception enabled status, receive data is transferred from the UARTDn receive shift register to the UDnRX register in synchronization with the completion of shift-in processing of 1 frame.

Transfer to the UDnRX register also causes the reception complete interrupt request signal (INTUDnR) to be output.

(9) 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.

(10) UARTDn transmit data register (UDnTX)

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.

17.3 UARTD Registers

(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

UD0CTL0:	FFFFFA00$_H$	UD1CTL0:	FFFFFA10$_H$
UD2CTL0:	FFFFFA20$_H$	UD3CTL0:	FFFFFA30$_H$
UD4CTL0:	FFFFFA40$_H$	UD5CTL0:	FFFFFA50$_H$
UD6CTL0:	FFFFFA60$_H$	UD7CTL0:	FFFFFA70$_H$

Initial Value 10$_H$. This register is initialized by any reset.

	7	6	5	4	3	2	1	0
UDnCTL0	UDnPWR	UDnTXE	UDnRXE	UDnDIR	UDnPS1	UDnPS0	UDnCL	UDnSL
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 17-2 UCnCTL0 register contents (1/2)

Bit position	Bit name	Function
7	UDnPWR	Controls UARTDn operation. 0: Stops clock operation (UARTDn asynchronously reset) 1: Enable UARTDn operation **Note:** 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).
6	UDnTXE	Enables transmission operation of UARTDn. 0: Stops transmission operation 1: Enables transmission operation **Note:** **1.** To start transmission, set UDnPWR bit to 1 and then set the UDnTXE bit to 1. **2.** To stop transmission, clear the UDnTXE bit to 0 and then UDnPWR bit to 0. **3.** To initialize the transmission unit, clear the UDnTXE bit to 0, wait for two cycles of the base clock , and the set then UDnTXE bit to 1 again.
5	UDnRXE	Enables reception operation of UARTDn. 0: Stops reception operation 1: Enables reception operation **Note:** **1.** To start reception, set UDnPWR bit to 1 and then set the UDnRXE bit to 1. **2.** To stop reception, clear the UDnRXE bit to 0 and then UDnPWR bit to 0. **3.** To initialize the reception unit, clear the UDnRXE bit to 0, wait for two cycles of the base clock , and then set the UDnRXE bit to 1 again. 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 RXDDn pin is enabled after the UDnRXE bit is set to 1 and 4 cycles of the base clock have passed.

Table 17-2 UCnCTL0 register contents (2/2)

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>4</td><td>UDnDIR</td><td>Selects the transfer direction.
0: MSB-first transfer
1: Data is sent/received with LSB first
Note: **1.** This bit can be rewritten only when UCnPWR = 0 or UCnTXE = UCnRXE = 0.
2. When the transmission/reception is performed in the LIN format, set the UDnDIR bit to 1.</td></tr>
<tr><td>3, 2</td><td>UDnPS[1:0]</td><td>Selects the parity function.
<table>
<tr><th rowspan="2">UCnPS1</th><th rowspan="2">UCnPS0</th><th colspan="2">Parity Selection</th></tr>
<tr><th>During Transmission</th><th>During Reception</th></tr>
<tr><td>0</td><td>0</td><td>No parity output</td><td>Reception with no parity</td></tr>
<tr><td>0</td><td>1</td><td>0 parity output</td><td>Reception with 0 parity</td></tr>
<tr><td>1</td><td>0</td><td>Odd parity output</td><td>Odd parity check</td></tr>
<tr><td>1</td><td>1</td><td>Even parity output</td><td>Even parity check</td></tr>
</table>
For details of parity, see *“Parity types and operations” on page 568*.
Note: **1.** This bit can be rewritten only when UCnPWR = 0 or UCnTXE = UCnRXE = 0.
2. If “Reception with 0 parity” is selected during reception, a parity check is not performed. Therefore, since the UDnSTR.UCnPE bit is not set, no error interrupt is output.
3. When transmission and reception are performed in the LIN format, set the UDnPS[1:0] bits to 00_B.</td></tr>
<tr><td>1</td><td>UDnDL</td><td>Specifies the data character length.
0: 7 bits
1: 8 bits
Note: **1.** This bit can be rewritten only when UCnPWR = 0 or UCnTXE = UCnRXE = 0.
2. When the transmission/reception is performed in the LIN format, set the UDnDIR bit to 1.</td></tr>
<tr><td>0</td><td>UDnSL</td><td>Specifies the stop bit length.
0: 1 bit
1: 2 bits
Note: This bit can be rewritten only when UCnPWR = 0 or UCnTXE = UCnRXE = 0.</td></tr>
</table>

(2) UDnCTL1- UARTDn control register 1

For details, see *“UDnCTL1 - UARTDn control register 1” on page 571*.

(3) UDnCTL2 - UARTDn control register 2

For details, see *“UDnCTL2 - UARTDn control register 2” on page 572*.

(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

UD0OPT0:	FFFFFA03$_H$	UD1OPT0:	FFFFFA13$_H$
UD2OPT0:	FFFFFA23$_H$	UD3OPT0:	FFFFFA33$_H$
UD4OPT0:	FFFFFA43$_H$	UD5OPT0:	FFFFFA53$_H$
UD6OPT0:	FFFFFA63$_H$	UD7OPT0:	FFFFFA73$_H$

Initial Value 14$_H$. This register is initialized by any reset.

	7	6	5	4	3	2	1	0
UDnOPT0	UDnSRF	UDnSRT	UDnSTT	UDnSLS2	UDnSLS1	UDnSLS0	UDnTDL	UDnRDL
	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 17-3 UDnOPT0 register contents (1/3)

Bit position	Bit name	Function
7	UDnSRF	SBF Reception Flag 0: When UDnCTL0.UDnPWR = 0 and UDnCTL0.UDnRXE = 0 are set to 1. Also upon normal end of SBF reception. 1: During SBF reception • SBF (Sync Brake Field) reception is judged during LIN communication. • The UDnSRF bit is held at 1 when an SBF reception error occurs, and then if the SBF reception is started again and ended normally, the UDnSRF bit is cleared to 0. Clearing by the instruction is disabled. • UDnSRF bit is read-only. When the UDnSRF = 1, the judgment process that SBF reception ended normally differs depending on the values of the SBF reception mode selection bit (UDnSRS). If the UDnSRS bit = 0, when any high level inputs including noises are applied to the reception input data even only for a second, the judgment of whether the low level period is more than 11 bits or not is executed. If the UDnSRS bit = 1, the received input data is sampled along with the set baud rate and when the low level period is 11 bits or more, it is judged as the successful SBF reception.

Table 17-3 UDnOPT0 register contents (2/3)

Bit position	Bit name	Function
6	UDnSRT	SBF Reception Trigger 0: – 1: SBF reception trigger • This is the SBF reception trigger bit during LIN communication, and when read, “0” is always read. For SBF reception, set the UDnSRT bit (to 1) to enable SBF reception. • The UDnSRT bit can be set during the reception but the reception is aborted. The updating of the status flag, output of the interrupt request flag, and the data saving are not performed so the receive data set during the reception is not guaranteed. • After the UDnSRT bit is set, re-setting of the UDnSRT bit is disabled until the SBF reception is succeeded, UDnSRF is cleared, and the interrupt request signal is fallen. • The detection of the SBF reception starts at the next falling edge of the reception input data. If the UDnSRT is set during the SBF reception, the SBF cannot be received, so other reception operations are not performed until the next SBF reception is succeeded. **Note:** **1.** When this bit is read, always “0” is returned. **2.** Set the UCnSRT bit when UCnCTL0.UCnPWR = 1 and UCnCTL0.UCnRXE = 1. **3.** To cancel the SBF reception enable status without receiving the SBF, set the UDnPWR bit = 0 or UDnRXE bit = 0.
5	UDnSTT	SBF Transmission Trigger 0: – 1: SBF transmission trigger • This is the SBF transmittion trigger bit during LIN communication. **Note:** **1.** When this bit is read, always “0” is returned. **2.** Set the UDnSTT bit when UDnCTL0.UCnPWR = 1 and UDnCTL0.UDnTXE = 1. **3.** Before starting the SBF transmission by UDnSTT = 1 make sure that no data transfer is ongoing, thus check if UCnSTR.UCnTSF = 0. **4.** 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. **5.** Data transmission is prohibited while UDnDCS bit = 1 during UDnSRF bit = 1. However, the SBF transmission is enabled.

Table 17-3 UDnOPT0 register contents (3/3)

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>4 to 2</td><td>UDnSLS[2:0]</td><td>Selects the SBF length.
<table>
<tr><th>UDnSLS2</th><th>UDnSLS1</th><th>UDnSLS0</th><th>SBF transmission length</th></tr>
<tr><td>1</td><td>0</td><td>1</td><td>13-bit output (default value)</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>14-bit output</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>15-bit output</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>16-bit output</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>17-bit output</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>18-bit output</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>19-bit output</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>20-bit output</td></tr>
</table>

Note: Setting of the UDnSLS[2:0] bits is permitted only when UDnCTL0.UDnPWR = 0, or UDnCTL0.UDnTXE = 0.</td></tr>
<tr><td>1</td><td>UDnTDL</td><td>Specifies the transmit data level
0: Normal output of transfer data
1: Inverted output of transfer data
• The value of the TXDDn pin can be inverted using the UDnTDL bit.
Note: **1.** Setting of the UDnTDL bit is permitted only when UDnCTL0.UDnPWR = 0, or UDnCTL0.UDnTXE = 0.
2. The UDnTDL bit 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 pin outputs the low level.</td></tr>
<tr><td>0</td><td>UDnRDL</td><td>Receive Data Level
0: Normal input of transfer data
1: Inverted input of transfer data
• The value of the RXDDn pin can be inverted using the UDnRDL bit.
Note: Setting of the UDnRDL bit is permitted only when UDnCTL0.UDnPWR = 0 ,or UDnCTL0.UDnRXE = 0.</td></tr>
</table>

(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

UD0OPT1:	FFFFFA05$_H$	UD1OPT1:	FFFFFA15$_H$
UD2OPT1:	FFFFFA25$_H$	UD3OPT1:	FFFFFA35$_H$
UD4OPT1:	FFFFFA45$_H$	UD5OPT1:	FFFFFA55$_H$
UD6OPT1:	FFFFFA65$_H$	UD7OPT1:	FFFFFA75$_H$

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

	7	6	5	4	3	2	1	0
UDnOPT1	0	0	0	0	0	0	UDnSRS	UDnDCS
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 17-4 UDnOPT1 register contents

Bit position	Bit name	Function
1	UDnSRS	Selects the SBF reception mode. 0: A new SBF can't be detected while the communication is in progress. When the low level is detected at the stop bit position, it is recognized as framing error. 1: A new SBF can be detected while the communication is in progress.When the low level is detected at the stop bit position a waiting state is generated until high level is detected. When the width of the low level is 11 bits or more, it is recognized as new SBF. **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 UDnDCS to 1.
0	UDnDCSL	Enables data consistency check. 0: Data consistency is not checked. 1: Data consistency is checked. When data is transmitted using the LIN protocoll, 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.

(6) UDnSTR - UARTDn status register

The UDnSTR register is an 8-bit register that displays the UARTDn transfer status and reception error contents.

Access This register can be read/written in 8-bit or 1-bit units.
Though the UDnTSF bit is a read-only bit, the UCnPE, UCnFE, and UCnOVE bits can be read and written. However, these bits can only be cleared by writing 0 to it; but cannot be set by writing 1 to it (even if 1 is written to them, the value is retained).

Address

UD0STR:	FFFFFA04$_H$	UD1STR:	FFFFFA14$_H$
UD2STR:	FFFFFA24$_H$	UD3STR:	FFFFFA34$_H$
UD4STR:	FFFFFA44$_H$	UD5STR:	FFFFFA54$_H$
UD6STR:	FFFFFA64$_H$	UD7STR:	FFFFFA74$_H$

Initial Value 00$_H$. This register is cleared by any reset, and when UDnCTL0.PWR = 0 is set.

The initialization conditions of the various bits 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 a 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
UDnSTR	UDnTSF	0	0	UDnSSF	UDnDCE	UDnPE	UDnFE	UDnOVE
	R	R	R	R	R	R/W	R/W	R/W

Table 17-5 UDnSTR register contents (1/3)

Bit position	Bit name	Function
7	UDnTSF	Transfer status flag 0: When the UDnPWR bit = 0, or when the UDnTXE bit = 0 has been set, or when the current transfer is completed and no next data was written to be transferred from UDnTX, or when the SBF has been finished after SBF transmission trigger was set. 1: When data to be transferred is written to UDnTX register, or when the SBF transmission trigger bit (UDnSST) is set. **Note:** **1.** 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. **2.** During the communication the UDnTSF bit is cleared after 2 clocks.

Table 17-5 UDnSTR register contents (2/3)

Bit position	Bit name	Function
4	UDnSSF	SBF receive successful flag 0: When the UDnPWR bit = 1, or when the UDnRXE bit = 0, or when the UDnSRS bit = 0, or when 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. **Note:** **1.** 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. **2.** The UDnSSF bit is maintained until 0 is written. It is always 0 for UD0SRS = 0.
3	UDnDCE	Data consistency error flag 0: When the UDnPWR bit = 0, or when the UDnTXE bit = 0, or when the UDnDCS bit = 0 has been set, or when the UDnDCE bit = 0 has been written. 1: This bit is set when the transmit data is not consistent to receive data in LIN communication mode. The send data is compared with the receive data when data is transmitted in LIN communication mode. When a mismatch is detected, this bit becomes 1. **Note:** The UDnDCE bit is maintained until 0 is written. It is always 0 for UD0DCS = 0. When 1 is written to this bit, the value is retained.
2	UDnPE	Parity Error Flag 0: When UDnCTL0.UDnPWR = 0, or when UDnCTL0.UDnRXE = 0 has been set (reception disabled), or when 0 has been written 1: When parity of data and parity bit do not match during reception. **Note:** **1.** The operation of the UDnPE bit is controlled by the settings of the UDnCTL0.UDnPS[1:0] bits. **2.** The UDnPE bit can be read and written, but it can only be cleared by writing 0 to it, but it cannot be set by writing 1 to it. When 1 is written to this bit, the value is retained.
1	UDnFE	Framing Error Flag 0: When UDnCTL0.UDnPWR = 0, or when UDnCTL0.UDnRXE = 0 has been set (reception disabled), or when 0 has been written 1: When no stop bit is detected during reception. **Note:** **1.** Only the first bit of the receive data stop bits is checked, regardless of the value of the UDnCTL0.UDnSL bit. **2.** The UDnFE bit can be read and written, but it can only be cleared by writing 0 to it, but it cannot be set by writing 1 to it. When 1 is written to this bit, the value is retained.

Table 17-5 UDnSTR register contents (3/3)

Bit position	Bit name	Function
0	UDnOVE	Overrun Error Flag 0: When UDnCTL0.UDnPWR = 0, or when UDnCTL0.UDnRXE = 0 has been set (reception disabled), or when 0 has been written 1: When data has been received into the UDnRX register and the next receive operation is completed before that receive data has been read. **Note:** **1.** When an overrun error occurs, the data is discarded without the next receive data being written to the receive buffer. **2.** The UDnOVE bit can be read and written, but it can only be cleared by writing 0 to it, but it cannot be set by writing 1 to it. When 1 is written to this bit, the value is retained.

(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.

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.

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

Address

UD0RX:	FFFFFA06_H	UD1RX:	FFFFFA16_H
UD2RX:	FFFFFA26_H	UD3RX:	FFFFFA36_H
UD4RX:	FFFFFA46_H	UD5RX:	FFFFFA56_H
UD6RX:	FFFFFA66_H	UD7RX:	FFFFFA76_H

Initial Value FF_H. This register is cleared by any reset, and when UDnCTL0.PWR = 0 is set.

	7	6	5	4	3	2	1	0
UDnRX	Receive data							
	R							

(8) UDnTX - UARTDn transmit data register

The UDnTX register is an 8-bit register used to set transmit data.

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

Reset input sets this register to FFH.

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

Address

UD0TX:	FFFFFA07_H	UD1TX:	FFFFFA17_H
UD2TX:	FFFFFA27_H	UD3TX:	FFFFFA37_H
UD4TX:	FFFFFA47_H	UD5TX:	FFFFFA57_H
UD6TX:	FFFFFA67_H	UD7TX:	FFFFFA77_H

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

	7	6	5	4	3	2	1	0
UDnTX	Receive data							
	R/W							

Note

1. When the transmission is enabled (UDnPWR = 1 and UDnTXE = 1) the write to the UDnTX register triggers the start of the transmission.
2. Be sure to execute the transmit data write during transmission after the transmission interrupt request (INTUDnT) is generated.
3. If the the next data is written before the transmission is completed the continuous transmission is enabled.

17.4 Interrupt Request Signals

The following three interrupt request signals are generated from UARTDn.

- Reception complete interrupt request signal (INTUDnR)
- Transmission enable interrupt request signal (INTUDnT)
- Status interrupt request signal (INTUDnS)

(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.

(2) 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.

(3) 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.

17.5 Operation

17.5.1 Data format

Full-duplex serial data reception and transmission is performed.

As shown in the figures below, one data frame of transmit/receive data consists of a start bit, character bits, parity bit, and stop bit(s).

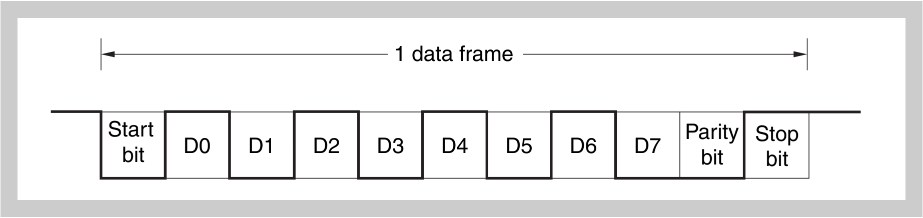
Specification of the character bit length within 1 data frame, parity selection, specification of the stop bit length, and specification of MSB/LSB-first transfer are performed using the UDnCTL0 register.

Moreover, control of UART output/inverted output for the TXDDn bit is performed using the UDnOPT0.UDnTDL bit.

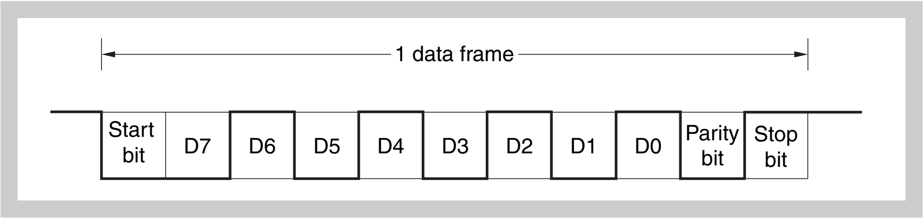
• Start bit	1 bit
• Character bits	7 bits/8 bits
• Parity bit parity/no parity	Even parity/odd parity/0
• Stop bit	1 bit/2 bits

(1) UARTD transmit/receive data format

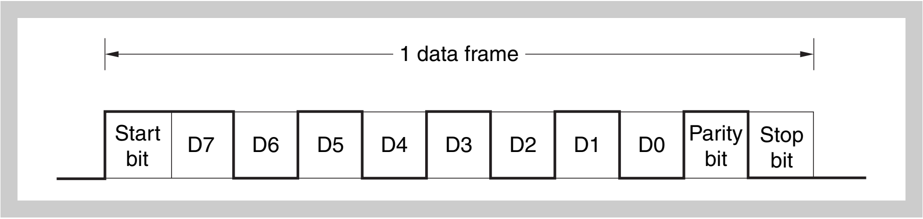
(a) 8-bit data length, LSB first, even parity, 1 stop bit, transfer data: 55H



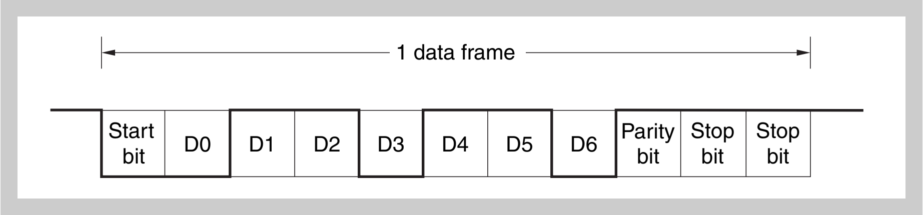
(b) 8-bit data length, MSB first, even parity, 1 stop bit, transfer data: 55H



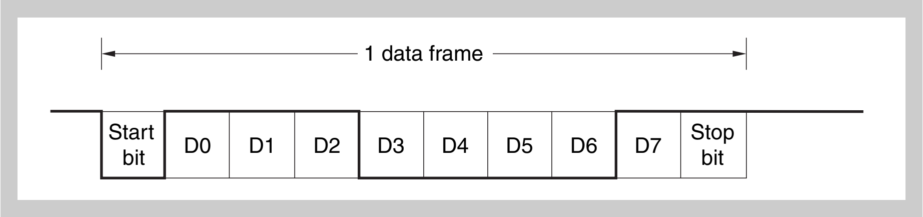
(c) 8-bit data length, MSB first, even parity, 1 stop bit, transfer data: 55H, TXDDn inversion



(d) 7-bit data length, LSB first, odd parity, 2 stop bits, transfer data: 36H



(e) 8-bit data length, LSB first, no parity, 1 stop bit, transfer data: 87H



17.5.2 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 17-2 and *Figure 17-3* outline the transmission and reception manipulations of LIN.

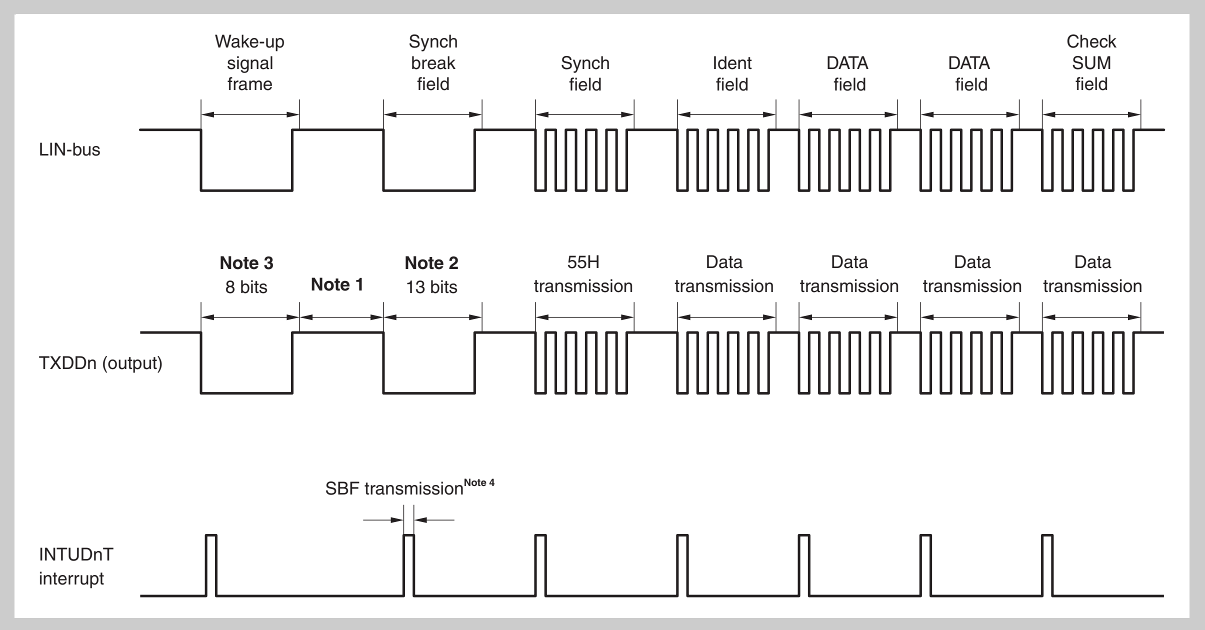


Figure 17-2 LIN transmission manipulation outline

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. 80H 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.

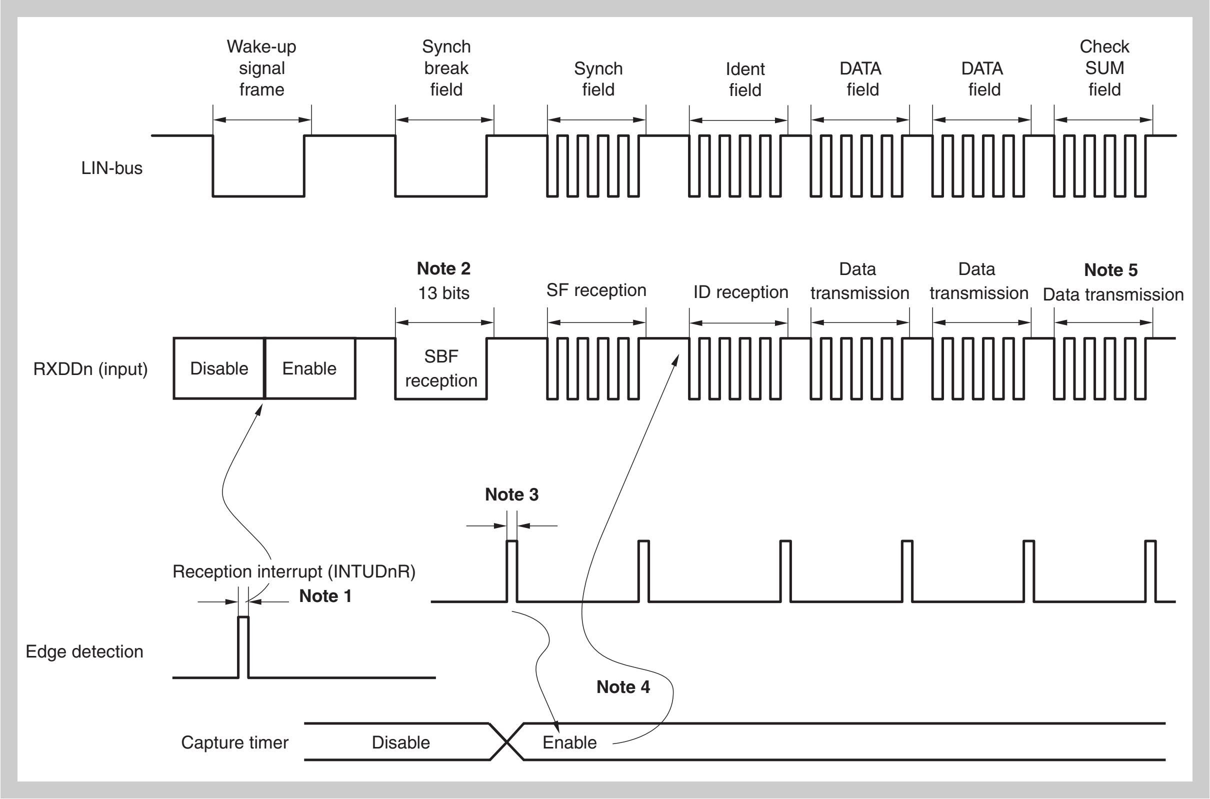


Figure 17-3 LIN reception manipulation outline

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.

17.5.3 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.

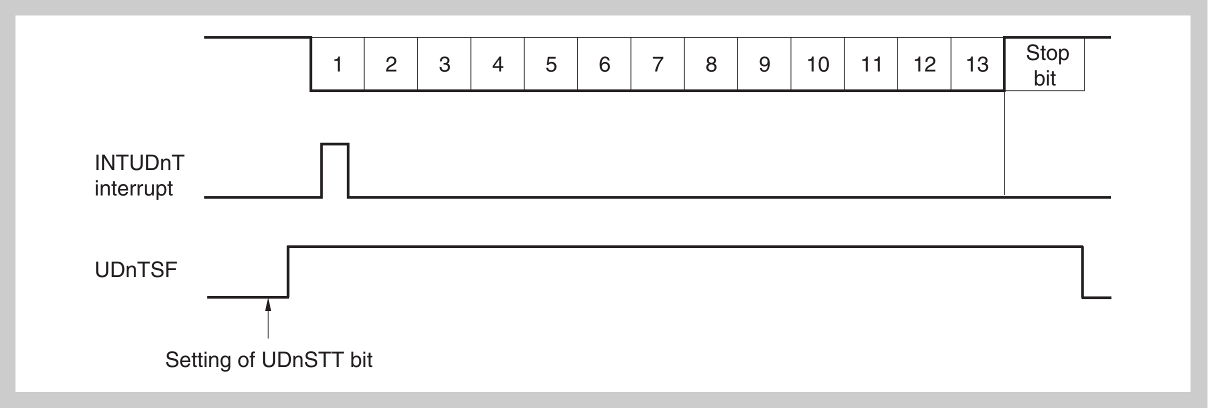


Figure 17-4 SBF transmission

17.5.4 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.

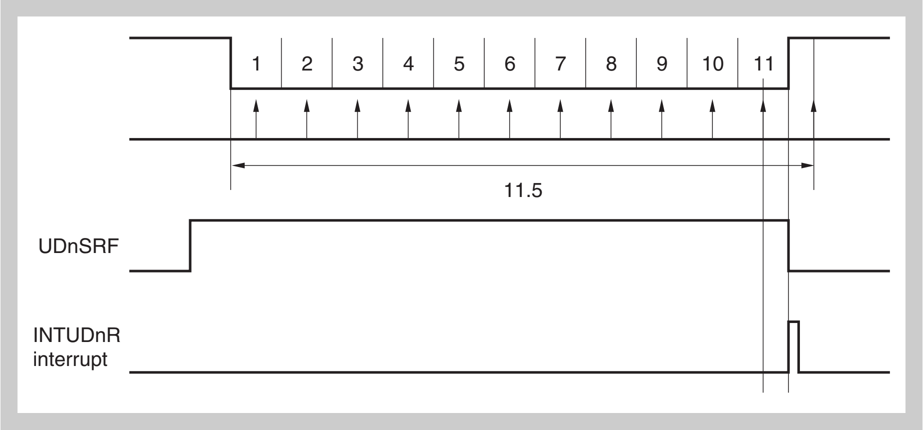
In the SBF reception wait status, similarly to the UART reception wait status, the RXDDn pin is monitored and start bit detection is performed.

Following detection of the low level, reception is started and the internal counter counts up according to the set baud rate.

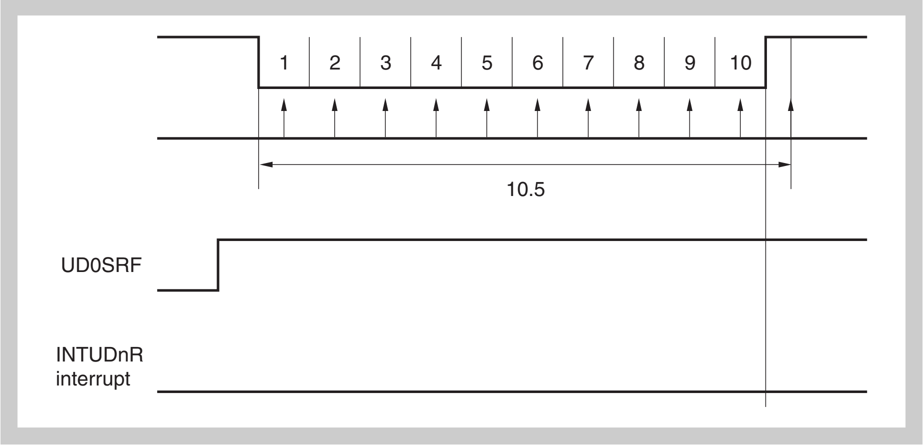
When a high level is received and if the SBF width is 11 or more bits, when SBF receiving mode selection bit (UDnSRS) is “0” the reception completion interrupt request signal (INTUDnR) is generated. When the UDnSRS bit is “1” the SBF reception success flag (UDnSSF) is set at the same time as generating 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.

(a) Normal SBF reception (detection of stop bit in more than 10.5 bits)



(b) SBF reception error (detection of stop bit in 10.5 or fewer bits)



Note The UDnSRF bit is reset by setting the UDnSRT bit to "1", and cleared by normal SBF reception.

17.5.5 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.

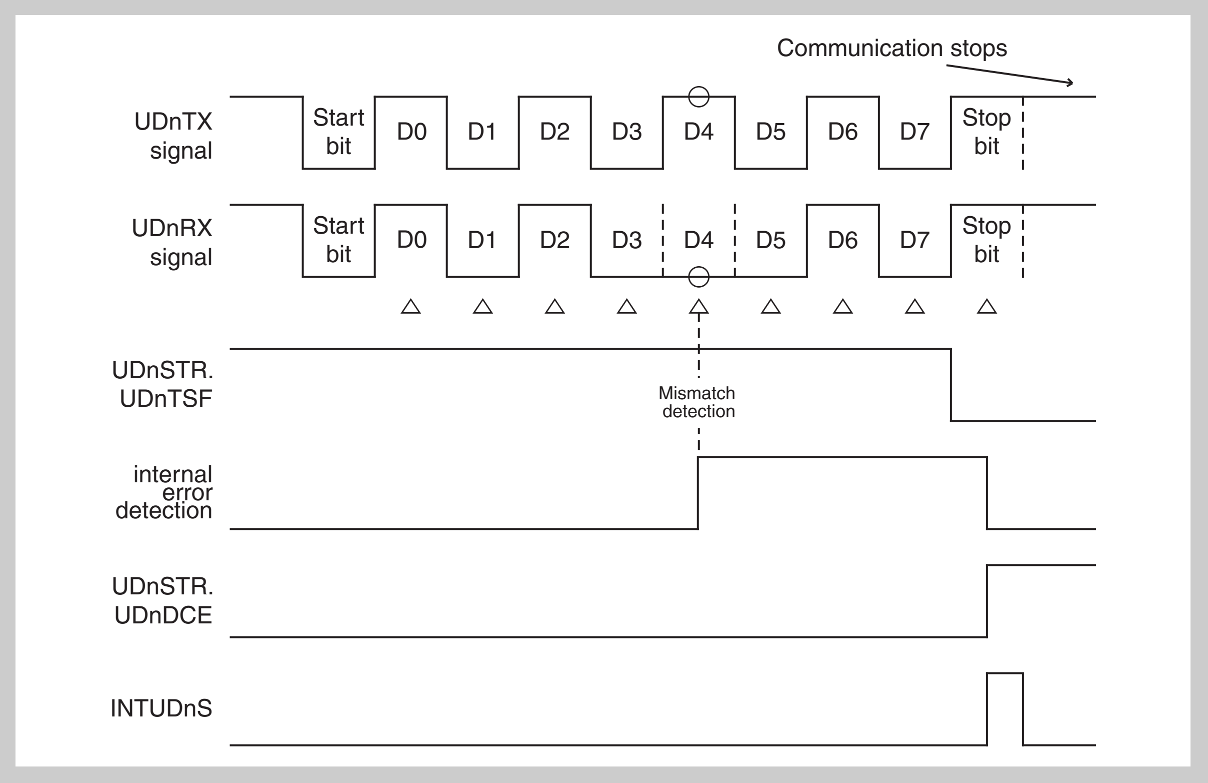
The data consistency is checked by comparing the transmit data in the transmit register (UDnTX) and the receive data in the receive register (UDnRX). In case of a mismatch the data consistency error flag (UDnSTR.UDnDCE) is set and a status interrupt request (INTUDnS) occurs.

The consistency check of data is not done in reception mode

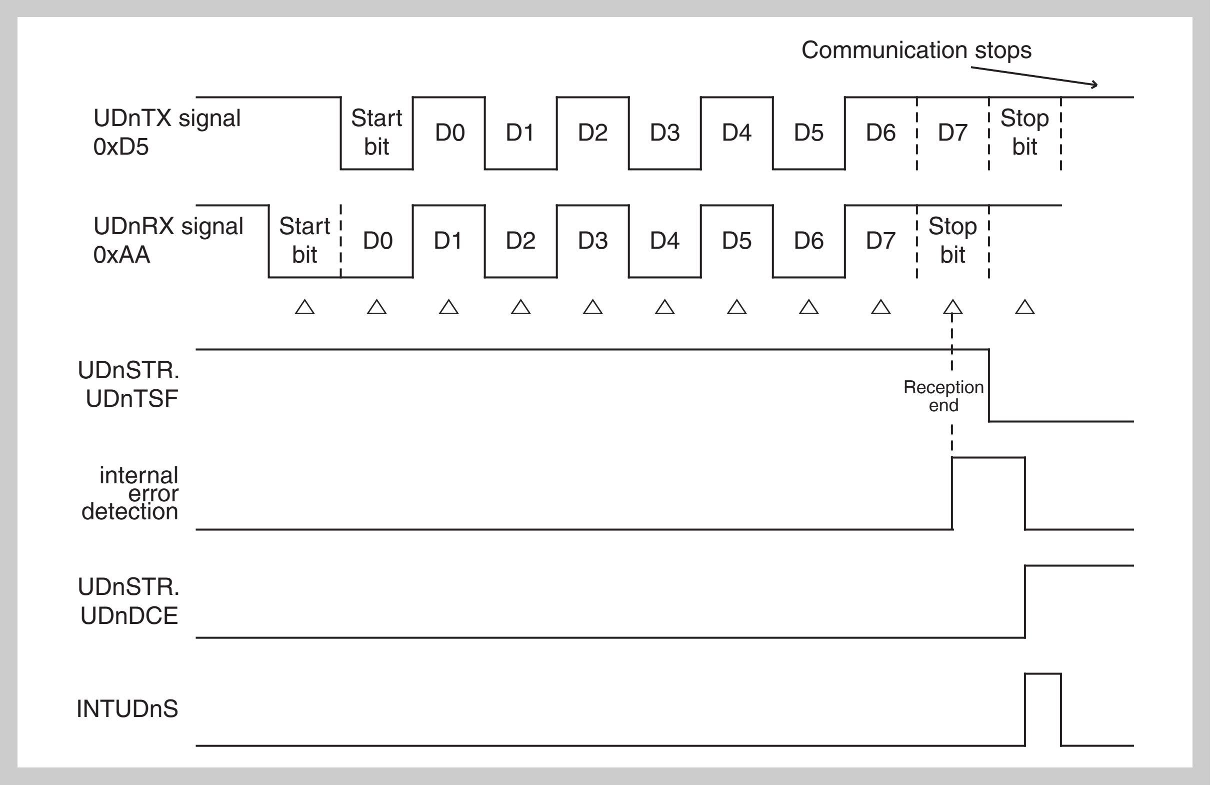
The consistency check of the send data and the input data terminal level is done even if the reception is disabled (UDnRXE = 0) during sending. In that case also the reception completion interrupt request signal (INTUDnR), the UDnSSF, UDnFE, UDnOVE and the status interrupt request signal (INTUDnS) will not be generated as well. Receive data does not need to be read.

Refer to *“UARTDn status register (UDnSTR)” on page 542* for details.

(a) Timing example of data consistency error (UDnSRF = 0)



(b) Timing example of data consistency error when there is a delay between transmit and receive operation

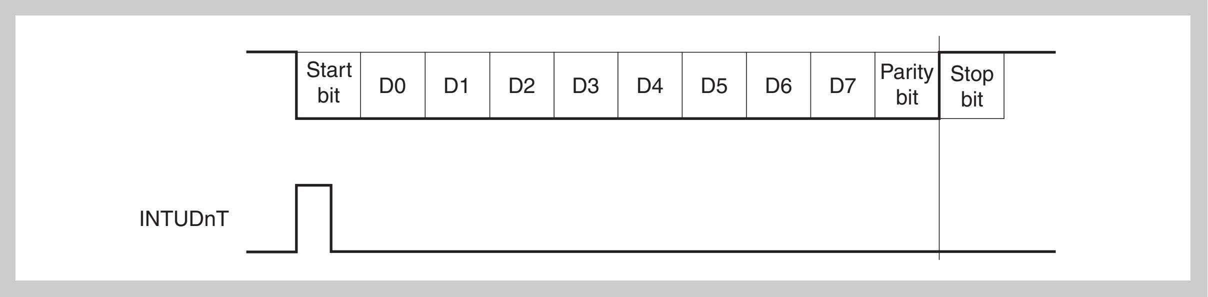


17.5.6 UART transmission

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

17.5.7 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.

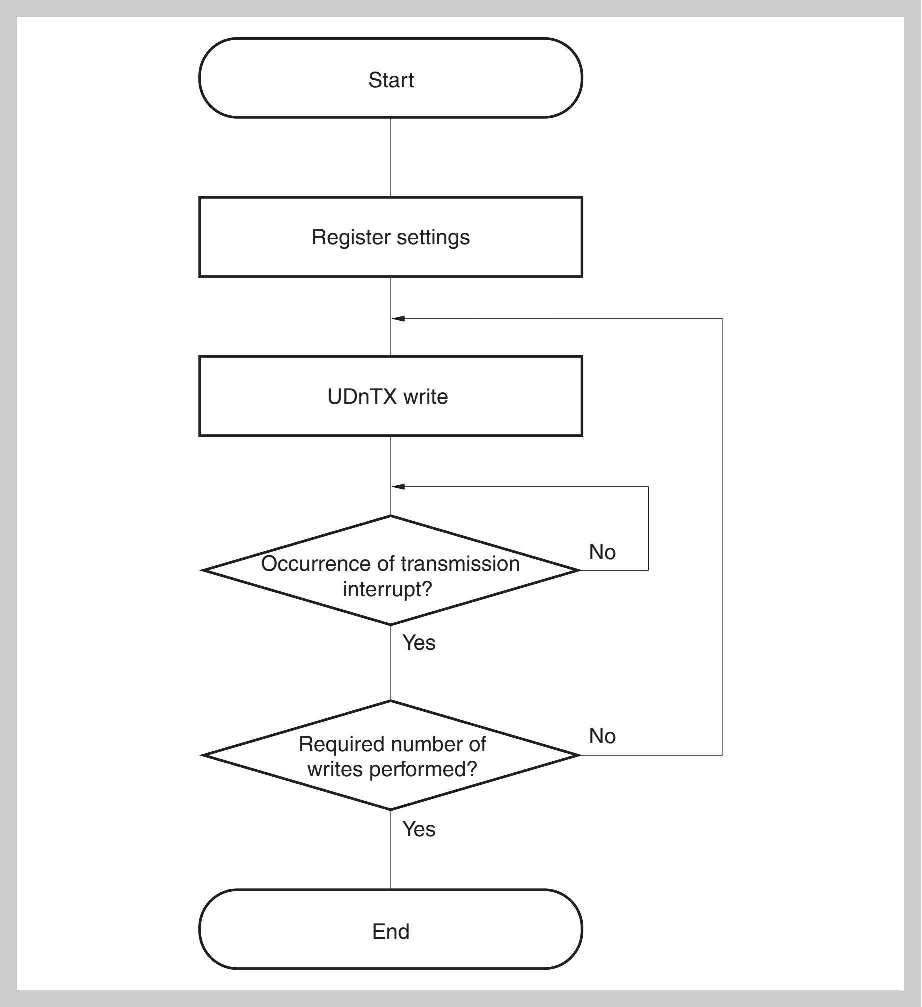

Start
Register settings
UDnTX write
Occurrence of transmission interrupt?
No
Yes
Required number of writes performed?
No
Yes
End

Figure 17-5 Continuous transmission processing flow

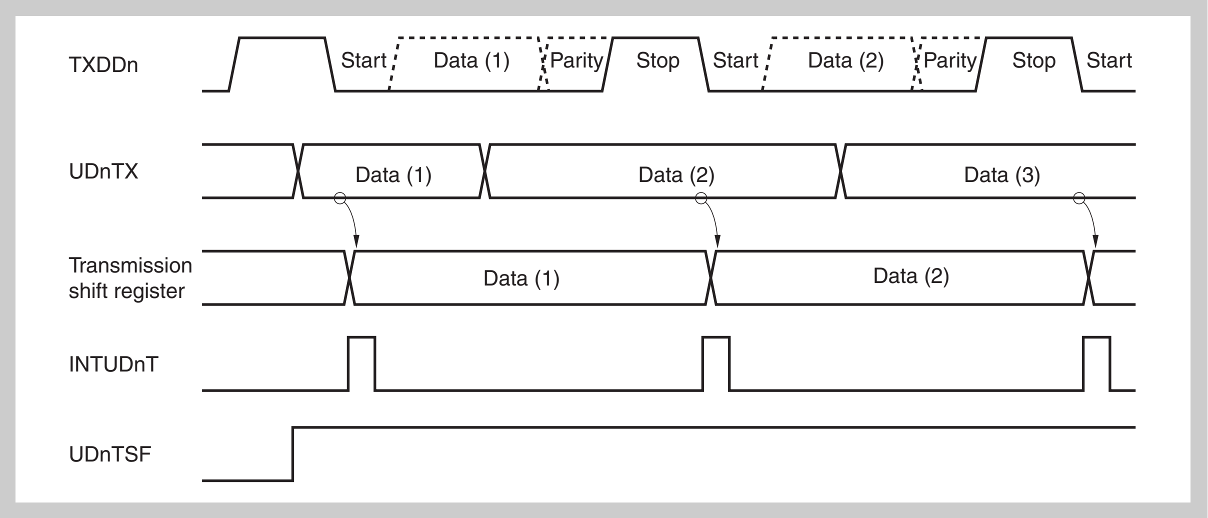


Figure 17-6 Continuous transmission operation timing —transmission start

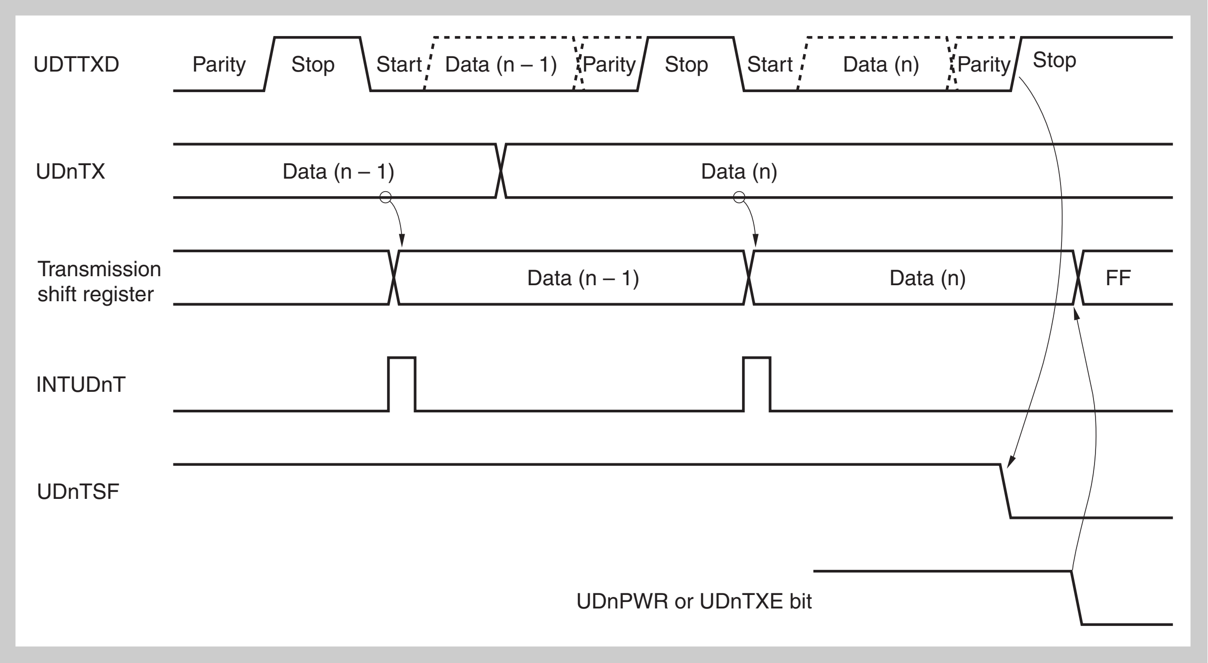


Figure 17-7 Continuous transmission operation timing—transmission end

17.5.8 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.

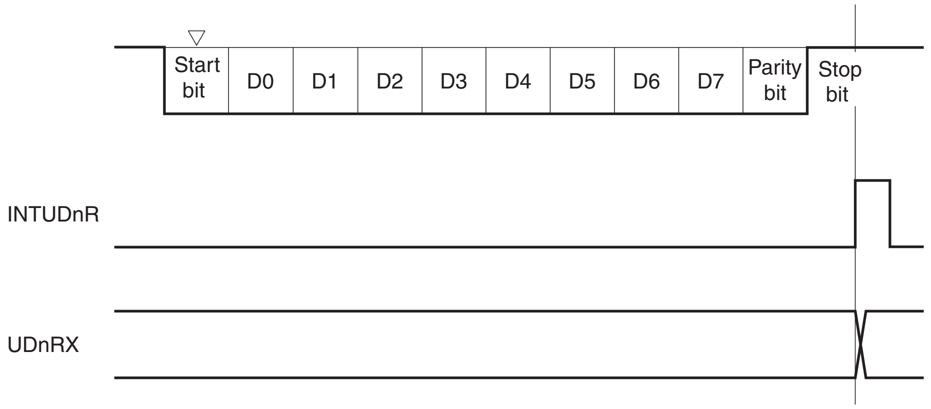


Figure 17-8 UART reception

Caution

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.

17.5.9 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 17-6 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.

17.5.10 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.

17.5.11 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 17-10*). See *"Base clock" on page 570* regarding the base clock.

Moreover, since the circuit is as shown in *Figure 17-9*, the processing that goes on within the receive operation is delayed by 3 clocks in relation to the external signal status.

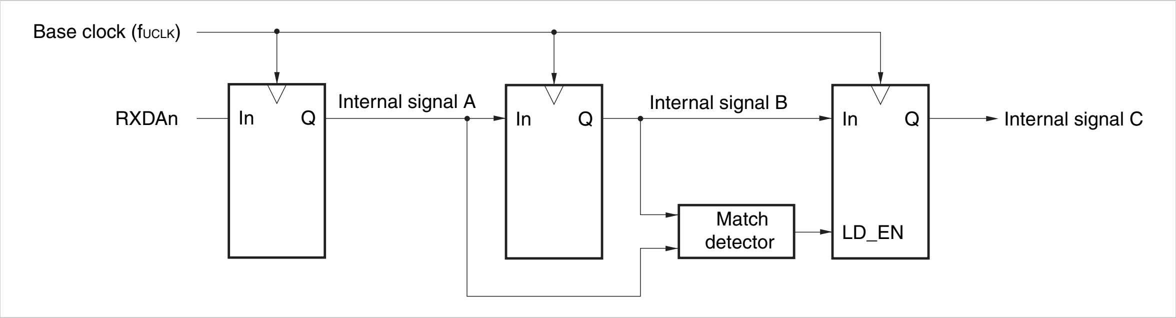


Figure 17-9 Noise filter circuit

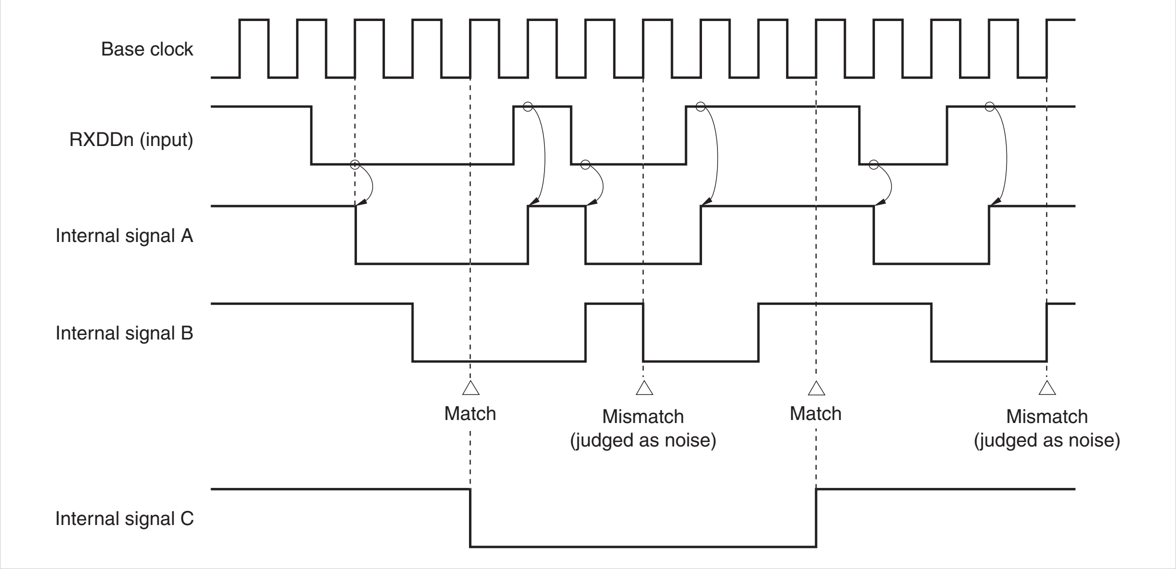


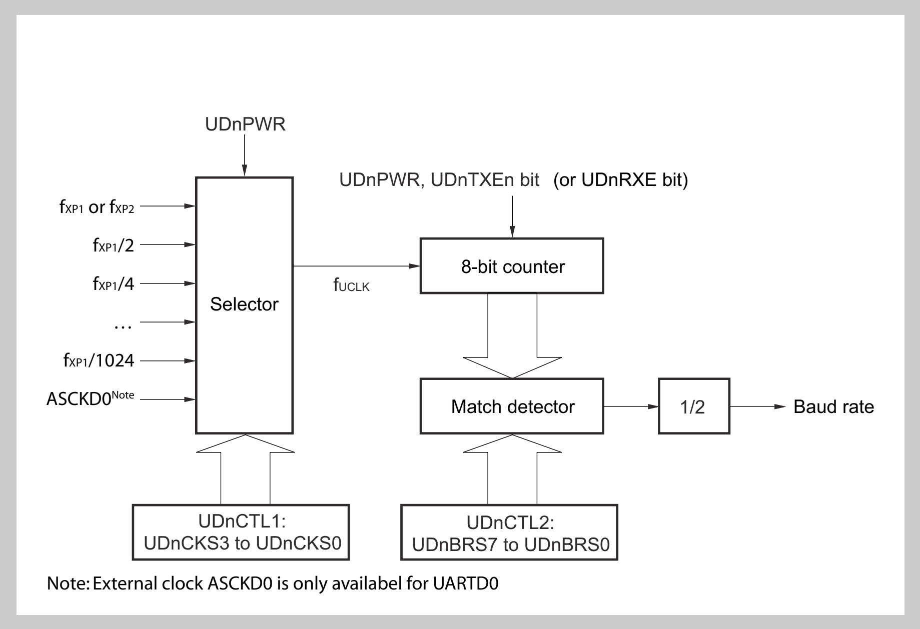
Figure 17-10 Timing of RXDDn signal judged as noise

17.6 Baud Rate Generator

The dedicated baud rate generator consists of a source clock selector block and an 8-bit programmable counter, and generates a serial clock during transmission and reception with 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.

(1) Baud rate generator configuration



Note: External clock ASCKD0 is only availabel for UARTD0

Figure 17-11 Configuration of baud rate generator

(a) Base clock

When the UDnCTL0.UDnPWR bit is 1, the clock selected by the UDnCTL1.UDnCKS[3:0] bits and SELCNT1.ISEL15 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.

(b) 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 and SELCNT1.ISEL15 bits.

The frequency division value for the 8-bit counter can be set using the UDnCTL2.UDnBRS[7:0] bits.

(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 units.

Address

UD0CTL1:	$FFFFFA01_H$	UD1CTL1:	$FFFFFA11_H$
UD2CTL1:	$FFFFFA21_H$	UD3CTL1:	$FFFFFA31_H$
UD4CTL1:	$FFFFFA41_H$	UD5CTL1:	$FFFFFA51_H$
UD6CTL1:	$FFFFFA61_H$	UD7CTL1:	$FFFFFA71_H$

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

	7	6	5	4	3	2	1	0
UDnCTL1	0	0	0	0	UDnCKS3	UDnCKS2	UDnCKS1	UDnCKS0
	R	R	R	R	R/W	R/W	R/W	R/W

Caution Clear the UDnCTL0.UDnPWR bit to 0 before rewriting the UDnCTL1 register.

Table 17-7 UDnCTL1 register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>3 to 0</td><td>UDnCKS [3:0]</td><td>Selects the base clock of UARTDn.
<table>
<tr><th>SELCNTm register[a]</th><th rowspan="2">UDnCKS3</th><th rowspan="2">UDnCKS2</th><th rowspan="2">UDnCKS1</th><th rowspan="2">UDnCKS0</th><th colspan="3">Input clock (f_{CLK})</th></tr>
<tr><th>ISELn</th><th>Input</th><th colspan="2">PRSI =</th></tr>
<tr><th></th><th></th><th></th><th></th><th></th><th></th><th>0</th><th>1</th></tr>
<tr><td>0</td><td rowspan="2">0</td><td rowspan="2">0</td><td rowspan="2">0</td><td rowspan="2">0</td><td>f_{XP1}</td><td>f_{XX}</td><td>$f_{XX}/2$</td></tr>
<tr><td>1</td><td>f_{XP2}[b]</td><td>f_{XX}</td><td>$f_{XX}/2$</td></tr>
<tr><td>×</td><td>0</td><td>0</td><td>0</td><td>1</td><td>$f_{XP1}/2$</td><td>$f_{XX}/2$</td><td>$f_{XX}/4$</td></tr>
<tr><td>×</td><td>0</td><td>0</td><td>1</td><td>0</td><td>$f_{XP1}/4$</td><td>$f_{XX}/4$</td><td>$f_{XX}/8$</td></tr>
<tr><td>×</td><td>0</td><td>0</td><td>1</td><td>1</td><td>$f_{XP1}/8$</td><td>$f_{XX}/8$</td><td>$f_{XX}/16$</td></tr>
<tr><td>×</td><td>0</td><td>1</td><td>0</td><td>0</td><td>$f_{XP1}/16$</td><td>$f_{XX}/16$</td><td>$f_{XX}/32$</td></tr>
<tr><td>×</td><td>0</td><td>1</td><td>0</td><td>1</td><td>$f_{XP1}/32$</td><td>$f_{XX}/32$</td><td>$f_{XX}/64$</td></tr>
<tr><td>×</td><td>0</td><td>1</td><td>1</td><td>0</td><td>$f_{XP1}64$</td><td>$f_{XX}/64$</td><td>$f_{XX}/128$</td></tr>
<tr><td>×</td><td>0</td><td>1</td><td>1</td><td>1</td><td>$f_{XP1}/128$</td><td>$f_{XX}/128$</td><td>$f_{XX}/256$</td></tr>
<tr><td>×</td><td>1</td><td>0</td><td>0</td><td>0</td><td>$f_{XP1}/256$</td><td>$f_{XX}/256$</td><td>$f_{XX}/512$</td></tr>
<tr><td>×</td><td>1</td><td>0</td><td>0</td><td>1</td><td>$f_{XP1}/512$</td><td>$f_{XX}/512$</td><td>$f_{XX}/1024$</td></tr>
<tr><td>×</td><td>1</td><td>0</td><td>1</td><td>0</td><td>$f_{XP1}/1024$</td><td>$f_{XX}/1024$</td><td>$f_{XX}/2048$</td></tr>
<tr><td>×</td><td>1</td><td>0</td><td>1</td><td>1</td><td>–</td><td colspan="2">ASCKD0[c]</td></tr>
<tr><td colspan="5">Other than above</td><td>–</td><td colspan="2">Setting prohibited</td></tr>
</table>

[a] For detailed information concerning the SELCNTm register refer to *“Clock Generator” on page 179*.
[b] f_{XP2} has the same frequency as f_{XP1}, but does not stop in IDLE1 mode.
[c] ASCKD0 is an external clock for UARTD0. For UARTD1 to UARTD5 the setting is prohibited.

Note: PRSI can be set by the option bytes (refer to *“Flash Mask Options” on page 330* for details):
• PRSI = 0: $f_{XX} \leq 32$ MHz
• PRSI = 1: 32 MHz $< f_{XX} \leq 48$ MHz</td></tr>
</table>

(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 units.

Address

UD0CTL2:	$FFFFFA02_H$	UD1CTL2:	$FFFFFA12_H$
UD2CTL2:	$FFFFFA22_H$	UD3CTL2:	$FFFFFA32_H$
UD4CTL2:	$FFFFFA42_H$	UD5CTL2:	$FFFFFA52_H$
UD6CTL2:	$FFFFFA62_H$	UD7CTL2:	$FFFFFA72_H$

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

	7	6	5	4	3	2	1	0
UDnCTL1	UDnBRS7	UDnBRS6	UDnBRS5	UDnBRS4	UDnBRS3	UDnBRS2	UDnBRS1	UDnBRS0
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Caution Clear the UDnCTL0.UDnPWR bit to 0 or clear the UDnTXE and UDnRXE bits to 00 before rewriting the UDnCTL2 register.

Table 17-8 UDnCTL1 register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>3 to 0</td><td>UDnBRS
[7:0]</td><td>Selects the baud rate clock of UARTDn.
<table>
<tr><th>UDn BRS7</th><th>UDn BRS6</th><th>UDn BRS5</th><th>UDn BRS4</th><th>UDn BRS3</th><th>UDn BRS2</th><th>UDn BRS1</th><th>UDn BRS0</th><th>Default (k)</th><th>Serial clock[a]</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>×</td><td>×</td><td colspan="2">Setting prohibited</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>4</td><td>$f_{UCLK}/4$</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td><td>5</td><td>$f_{UCLK}/5$</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>6</td><td>$f_{UCLK}/6$</td></tr>
<tr><td colspan="8">⋮</td><td>⋮</td><td>⋮</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>252</td><td>$f_{UCLK}/252$</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>253</td><td>$f_{UCLK}/253$</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>254</td><td>$f_{UCLK}/254$</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>255</td><td>$f_{UCLK}/255$</td></tr>
</table>
[a] f_{UCLK}: Clock frequency selected by UDnCTL1.UDnCKS[3:0]</td></tr>
</table>

(4) 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] and SELCNT1.ISEL15

k = Value set using the UDnCTL2.UDnBRS[7:0] bits (k = 4, 5, 6, …, 255)

(5) Baud rate error

The baud rate error is obtained by the following equation.

$$\text{Error (\%)} = \left(\frac{\text{Actual baud rate (baud rate with error)}}{\text{Target baud rate (correct baud rate)}} - 1\right) \times 100\ [\%]$$

Caution

1. The baud rate error during transmission must be within the error tolerance on the receiving side.
2. The baud rate error during reception must satisfy the range indicated in (7) Allowable baud rate range during reception.

Example

- Base clock frequency f_{xx} = 32 MHz
- Setting value of
  - PRSI = 0: $f_{XP1} = f_{xx}$ = 32 MHz
  - UDnCTL1.UDnCKS[3:0] = 0001_B: $f_{UCLK} = f_{XP1}/2$ = 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 [%]

(6) Baud rate setting example

Table 17-9 Baud rate generator setting data (normal operation, f_{XP1} = 24 MHz, PRSI = 1)

Target baud rate	UDnCTL1		UDnCTL2		Actual baud rate [bps]	Baud rate error (%)
[bps]	Selector	Divider	Divider k			
300	$0A_H$	2048	27_H	39	300.48	0.16
600	09_H	1024	27_H	39	600.96	0.16
1,200	08_H	512	27_H	39	1,201.92	0.16
2,400	07_H	256	27_H	39	2,403.85	0.16
4,800	06_H	128	27_H	39	4,807.69	0.16
9,600	05_H	64	27_H	39	9,615.38	0.16
19,200	04_H	32	27_H	39	19,230.77	0.16
31,250	05_H	64	$0C_H$	12	31,250.00	0.00
38,400	03_H	16	27_H	39	38,461.54	0.16
76,800	02_H	8	27_H	39	76,923.08	0.16
153,600	01_H	4	27_H	39	153,846.15	0.16
312,500	00_H	2	27_H	39	307,692.31	–1.54

Table 17-10 Baud rate generator setting data (normal operation, f_{XP1} = 32 MHz, PRSI = 0)

Target baud rate	UDnCTL1		UDnCTL2		Actual baud rate [bps]	Baud rate error (%)
[bps]	Selector	Divider	Divider k			
300	08_H	256	$D0_H$	208	300.48	0.16
600	08_H	256	68_H	104	600.96	0.16
1,200	08_H	256	34_H	52	1,201.92	0.16
2,400	07_H	128	34_H	52	2,403.85	0.16
4,800	06_H	64	34_H	52	4,807.69	0.16
9,600	05_H	32	34_H	52	9,615.38	0.16
19,200	04_H	16	34_H	52	19,230.77	0.16
31,250	05_H	32	10_H	16	31,250.00	0.00
38,400	03_H	8	34_H	52	38,461.54	0.16
76,800	02_H	4	34_H	52	76,923.08	0.16
153,600	01_H	2	34_H	52	153,846.15	0.16
312,500	00_H	1	34_H	52	307,692.31	–1.54

Table 17-11 Baud rate generator setting data (normal operation, f_{XP1} = 16 MHz, PRSI = 0)

Target baud rate	UDnCTL1		UDnCTL2		Actual baud rate [bps]	Baud rate error (%)
[bps]	Selector	Divider	Divider k			
300	08_H	256	68_H	104	300.48	0.16
600	08_H	256	34_H	52	600.96	0.16
1,200	07_H	128	34_H	52	1,201.92	0.16
2,400	06_H	64	34_H	52	2,403.85	0.16
4,800	05_H	32	34_H	52	4,807.69	0.16
9,600	04_H	16	34_H	52	9,615.38	0.16
19,200	03_H	8	34_H	52	19,230.77	0.16
31,250	03_H	8	20_H	32	31,250.00	0.00
38,400	02_H	4	34_H	52	38,461.54	0.16
76,800	01_H	2	34_H	52	76,923.08	0.16
153,600	00_H	1	34_H	52	153,846.15	0.16
312,500	00_H	1	$1A_H$	26	307,692.31	–1.54

(7) 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.

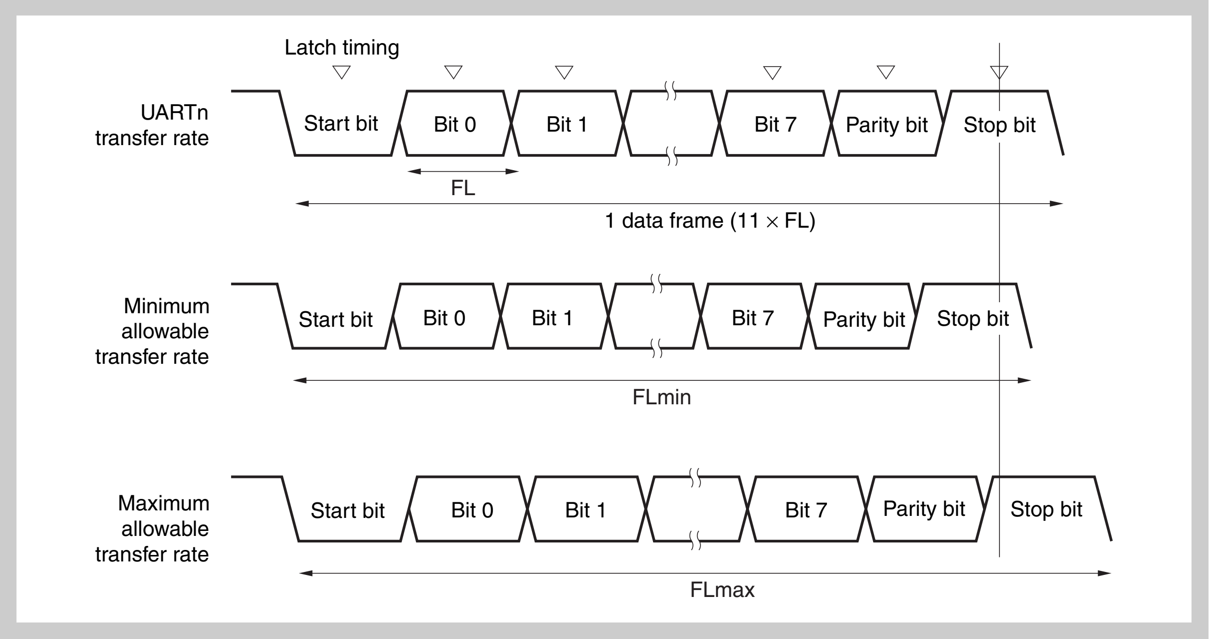


Figure 17-12 Allowable baud rate range during reception

As shown in *Figure 17-12*, 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.

$$FL = \langle Brate \rangle^{-1}$$

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.

$$BR_{max} = (FL_{min}/11)^{-1} = \frac{22k}{21k+2} \times Brate$$

Similarly, obtaining the following maximum allowable transfer rate yields the following.

$$\frac{10}{11} \times FL_{max} = 11 \times FL - \frac{k+2}{2k} \times FL = \frac{21k-2}{2k} \times FL$$

$$FL_{max} = \frac{21k-2}{20k} \times FL \times 11$$

Therefore, the minimum baud rate that can be received by the destination is as follows.

$$BRmin = (FL_{max}/11)^{-1} = \frac{20k}{21k-2} \times Brate$$

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 17-12 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]

(8) 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.

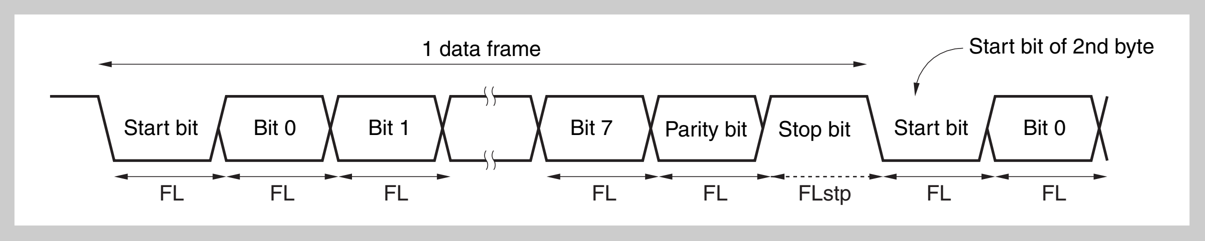


Figure 17-13 Transfer rate during continuous transfer

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.

$$\text{Transfer rate} = 11 \times FL + (2/f_{UCLK})$$

17.7 Cautions

When the clock supply to UARTDn is stopped (for example, in IDLE1, IDLE2, or STOP mode), the operation stops with each register retaining the value it had immediately before the clock supply was stopped. The TXDDn pin output also holds and outputs the value it had immediately before the clock supply was stopped. However, the operation is not guaranteed after the clock supply is resumed. Therefore, after the clock supply is resumed, the circuits should be initialized by setting the UDnCTL0.UDnPWR, UDnCTL0.UDnRXEn, and UDnCTL0.UDnTXEn bits to 0_B.

Chapter 18 Clocked Serial Interface (CSIB)

The V850ES/Fx3 microcontrollers have following instances of the Clocked Serial Interface CSIB:

CSIB	V850ES/FE3	V850ES/FF3	V850ES/FG3	V850ES/FJ3		V850ES/FK3
				µPD70F3378 µPD70F3379 µPD70F3380	µPD70F3381 µPD70F3382	
Instances	2			3	4	
Names	CSIB0 to CSIB1			CSIB0 to CSIB2	CSIB0 to CSIB3	

Throughout this chapter, the individual instances of CSIB are identified by “n”, for example, CBnCTL0 for the CSIBn control register 0.

18.1 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 (INTCBnT and INTCBnR)
- 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 SOBn: Serial data output
  SIBn: Serial data input
  SCKBn: Serial clock input/output

  Transmission mode, reception mode, and transmission/reception mode specifiable
- DMA support
  Two different DMA trigger events in transmission mode for CSIB0, CSIB1 and CSIB2 (refer to *“DMA Function (DMA Controller)” on page 373*)
- Baud rate generator input for CSIB0

18.2 Configuration

The following shows the block diagram of CSIBn.

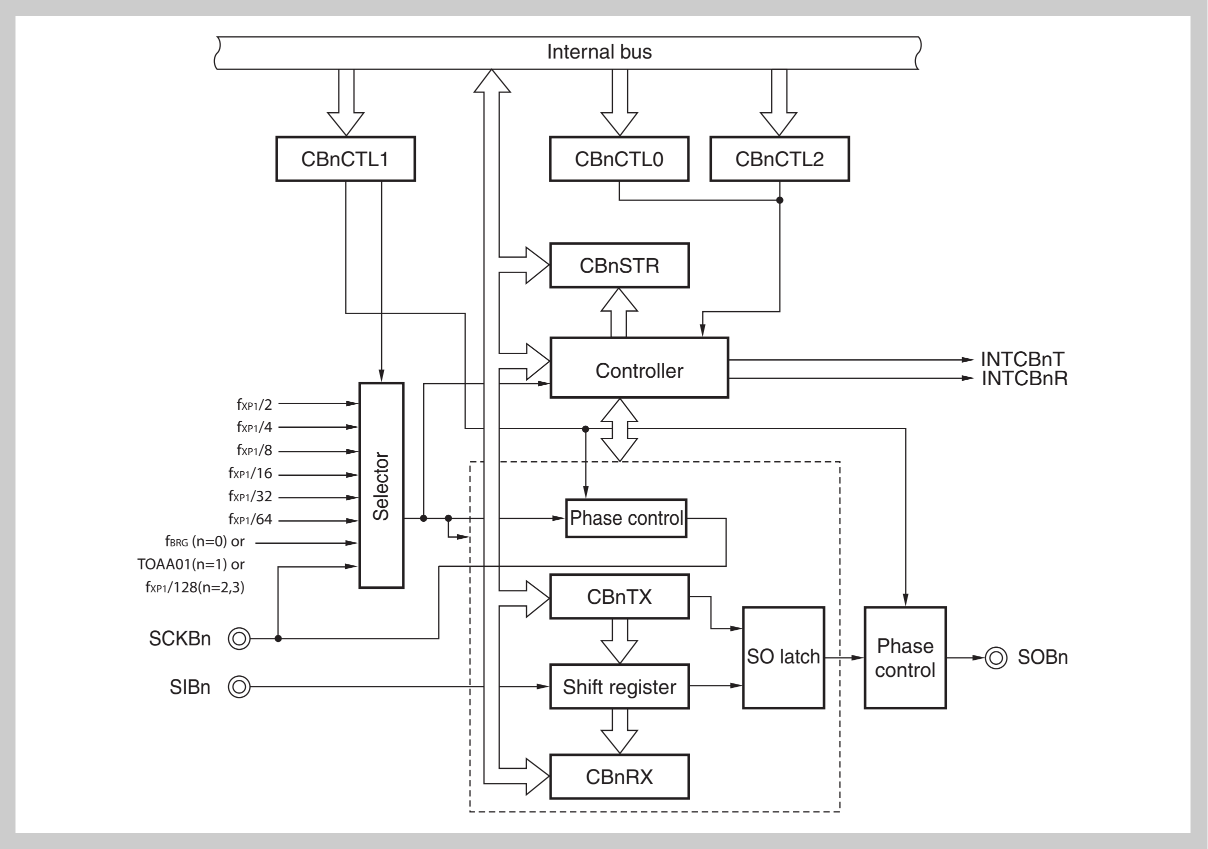


Figure 18-1 Block diagram of CSIBn

Note For details on the setting refer to *“CBnCTL1 - CSIBn control register 1” on page 583*.

CSIBn includes the following hardware.

Table 18-1 Configuration of CSIBn

Item	Configuration
Registers	CSIBn receive data register (CBnRX) CSIBn transmit data register (CBnTX)
Control registers	CSIBn control register 0 (CBnCTL0) CSIBn control register 1 (CBnCTL1) CSIBn control register 2 (CBnCTL2) CSIBn status register (CBnSTR)

(1) CBnRX - CSIBn receive data register

The CBnRX register is a 16-bit buffer register that holds receive data.The receive operation is started by reading the CBnRX register in the reception enabled status.

Access This register can be read-only in 16-bit units.
If the transfer data length is 8 bits, the lower 8 bits of this register are read-only in 8-bit units as the CBnRXL register.

Address CB0RX: FFFFFD04$_H$
CB1RX: FFFFFD14$_H$
CB2RX: FFFFFD24$_H$
CB3RX: FFFFFD34$_H$

Initial Value 0000$_H$. This register is cleared by any reset.
In addition to reset input, the CBnRX register can be initialized by clearing (to 0) the CBnPWR bit of the CBnCTL0 register.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CBnRX	Receive data															
	R															

(2) CBnTX - CSIB transmit data register

The CBnTX register is a 16-bit buffer register used to write the CSIBn transfer data.The transmit operation is started by writing data to the CBnTX register in the transmission enabled status.

Access This register can be read/written in 16-bit units.
If the transfer data length is 8 bits, the lower 8 bits of this register are read/write in 8-bit units as the CBnTXL register.

Address CB0TX: FFFFFD06$_H$
CB1TX: FFFFFD16$_H$
CB2TX: FFFFFD26$_H$
CB3TX: FFFFFD36$_H$

Initial Value 0000$_H$. This register is cleared by any reset.
In addition to reset input, the CBnTX register can be initialized by clearing (to 0) the CBnPWR bit of the CBnCTL0 register.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CBnTX	Transmit data															
	R/W															

Note The communication start conditions are shown below:

- Transmission mode (CBnTXE bit = 1, CBnRXE bit = 0):
  Write to CBnTX register
- Transmission/reception mode (CBnTXE bit = 1, CBnRXE bit = 1):
  Write to CBnTX register
- Reception mode (CBnTXE bit = 0, CBnRXE bit = 1):
  Read from CBnRX register

18.3 CSIB Control Registers

The following registers are used to control CSIBn.

- CSIBn control register 0 (CBnCTL0)
- CSIBn control register 1 (CBnCTL1)
- CSIBn control register 2 (CBnCTL2)
- CSIBn status register (CBnSTR)

(1) CBnCTL0 - CSIBn control register 0

CBnCTL0 is a register that controls the CSIBn serial transfer operation.

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

Address CB0CTL0: FFFFFD00$_H$
CB1CTL0: FFFFFD10$_H$
CB2CTL0: FFFFFD20$_H$
CB3CTL0: FFFFFD30$_H$

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

	7	6	5	4	3	2	1	0
CBnCTL0	CBnPWR	CBnTXE[a]	CBnRXE[a]	CBnDIR[a]	0	0	CBnTMS[a]	CBnSCE
	R/W	R/W	R/W	R/W	R	R	R/W	R/W

[a] These bits can only be rewritten when the CBnPWR bit = 0. However, CBnPWR bit = 1 can also be set at the same time as rewriting these bits.

Table 18-2 CBnCTL0 register contents (1/2)

Bit position	Bit name	Function
7	CBnPWR	CSIBn operation disable/enable: 0: Disable CSIBn operation and reset the CSIBn registers 1: Enable CSIBn operation The CBnPWR bit controls the CSIBn operation and resets the internal circuit. **Note:** To abort reception/transmission forcibly, clear the CBnPWR bit (not the CBnTXE bit or CBnRXE bit) to 0. The clock output stops at this time.
6	CBnTXE	Transmit operation disable/enable: 0: Disable transmit operation 1: Enable transmit operation The SOBn output is low level and transmission is disabled when the CBnTXE bit is 0.
5	CBnRXE	Receive operation disable/enable: 0: Disable receive operation 1: Enable receive operation When the CBnRXE bit is cleared to 0, no reception complete interrupt is output even when the prescribed data is transferred in order to disable the receive operation, and the receive data (CBnRX register) is not updated.
4	CBnDIR	Transfer direction mode specification (MSB/LSB): 0: MSB first transfer 1: LSB first transfer
1	CBnTMS	Transfer mode specification (MSB/LSB): 0: Single transfer mode 1: Continuous transfer mode

Table 18-2 CBnCTL0 register contents (2/2)

Bit position	Bit name	Function
0	CBnSCE	Specification of start transfer disable/enable: 0: Communication start trigger invalid 1: Communication start trigger valid This bit controls the behaviour upon a communication start trigger in master/slave single/continuous reception mode. To start the reception operation set the bit to 1 before performing a dummy read to the CBnRX register. To stop the reception operation in • Single reception mode clear the CBnSCE bit before reading the final data from the CBnRX register. • Continuous reception mode clear the CBnSCE bit at least one communication clock before the completion of the last data reception. • For details refer to chapter 18.4“Operation” on page 588.

(2) CBnCTL1 - CSIBn control register 1

CBnCTL1 is an 8-bit register that controls the CSIBn serial transfer operation.

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

Address CB0CTL1: FFFFFD01_H
CB1CTL1: FFFFFD11_H
CB2CTL1: FFFFFD21_H
CB3CTL1: FFFFFD31_H

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

	7	6	5	4	3	2	1	0
CBnCTL1	0	0	0	CBnCKP	CBnDAP	CBnCKS2	CBnCKS1	CBnCKS0
	R	R	R	R/W	R/W	R/W	R/W	R/W

Caution The CBnCTL1 register can be rewritten only when the CBnCTL0.CBnPWR bit = 0.

Table 18-3 CBnCTL1 register contents

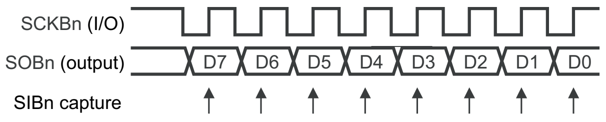
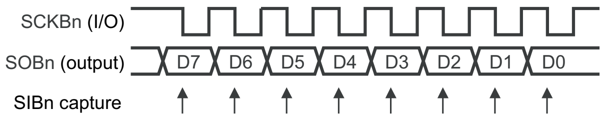
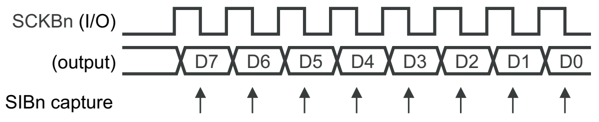
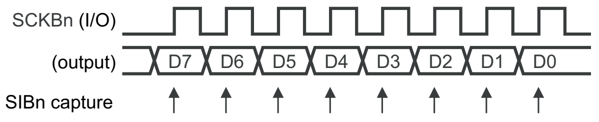
<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>4
3</td><td>CBnCKP
CBnDAP</td><td>Specification of data transmission/reception timing in relation to SCKBn.
<table>
<tr><th>CBnCKP</th><th>CBnDAP</th><th>SIBn/SOBN timing in relation to SCKBn</th></tr>
<tr><td>0</td><td>0</td><td>Communication type 1

</td></tr>
<tr><td>0</td><td>1</td><td>Communication type 2

</td></tr>
<tr><td>1</td><td>0</td><td>Communication type 3

</td></tr>
<tr><td>1</td><td>1</td><td>Communication type 4

</td></tr>
</table></td></tr>
<tr><td>2 to 0</td><td>CBnCKS[2:0]</td><td>Communication clock setting.
Refer to<i>Table 18-4</i></td></tr>
</table>

Table 18-4 Communication clock setting

CBn CKS2	CBn CKS1	CBn CKS0	Input	Input clock						Mode[a]
				n = 0		n = 1		n = 2, 3		
				PRSI = 0	PRSI = 1	PRSI = 0	PRSI = 1	PRSI = 0	PRSI = 1	
0	0	0	$f_{XP1}/2$[b]	$f_{xx}/2$	$f_{xx}/4$	$f_{xx}/2$	$f_{xx}/4$	$f_{xx}/2$	$f_{xx}/4$	M
0	0	1	$f_{XP1}/4$[b]	$f_{xx}/4$	$f_{xx}/8$	$f_{xx}/4$	$f_{xx}/8$	$f_{xx}/4$	$f_{xx}/8$	M
0	1	0	$f_{XP1}/8$[b]	$f_{xx}/8$	$f_{xx}/16$	$f_{xx}/8$	$f_{xx}/16$	$f_{xx}/8$	$f_{xx}/16$	M
0	1	1	$f_{XP1}/16$[b]	$f_{xx}/16$	$f_{xx}/32$	$f_{xx}/16$	$f_{xx}/32$	$f_{xx}/16$	$f_{xx}/32$	M
1	0	0	$f_{XP1}/32$[b]	$f_{xx}/32$	$f_{xx}/64$	$f_{xx}/32$	$f_{xx}/64$	$f_{xx}/32$	$f_{xx}/64$	M
1	0	1	$f_{XP1}/64$[b]	$f_{xx}/64$	$f_{xx}/128$	$f_{xx}/64$	$f_{xx}/128$	$f_{xx}/64$	$f_{xx}/128$	M
1	1	0	f_{BRG}[c]	f_{BRG}		–		–		M
			TOAA01[d]	–		TOAA01		–		M
			$f_{XP1}/128$[b]	–		–		$f_{xx}/128$	$f_{xx}/256$	M
1	1	1	External clock SCKBn							S

a) M: master mode; S: slave mode

b) Do not use the CSIBn if $f_{XP1} = f_{RH}$ (high speed internal oscillator clock)

c) The baud rate generator output is also used for the Watch Timer.

d) Output of TAA0

Note PRSI can be set by the option bytes. Refer to *"Flash Mask Options" on page 330* for details.

(3) CBnCTL2 - CSIBn control register 2

CBnCTL2 is an 8-bit register that controls the number of CSIBn serial transfer bits.

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

Address CB0CTL2: FFFFFD02$_H$
CB1CTL2: FFFFFD12$_H$
CB2CTL2: FFFFFD22$_H$
CB3CTL2: FFFFFD32$_H$

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

	7	6	5	4	3	2	1	0
CBnCTL2	0	0	0	0	CBnCL3	CBnCL2	CBnCL1	CBnCL0
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Caution The CBnCTL2 register can be rewritten only when the CBnCTL0.CBnPWR bit = 0 or when both the CBnTXE and CBnRXE bits = 0.

Table 18-5 CBnCTL2 register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>3 to 0</td><td>CBnCL[3:0]</td><td>Number of serial transfer bits
<table>
<tr><th>CBnCL3</th><th>CBnCL2</th><th>CBnCL1</th><th>CBnCL0</th><th>Number of serial transfer bits</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>8 bits</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>9 bits</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>10 bits</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>11 bits</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td><td>12 bits</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1</td><td>13 bits</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td><td>14 bits</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>1</td><td>15 bits</td></tr>
<tr><td>1</td><td>x</td><td>x</td><td>x</td><td>16 bits</td></tr>
</table></td></tr>
</table>

Note If the number of transfer bits is other than 8 or 16, prepare and use data stuffed from the LSB of the CBnTX and CBnRX registers.

Transfer data length change function The CSIBn transfer data length can be set in 1-bit units between 8 and 16 bits using the CBnCTL2.CBnCL3 to CBnCTL2.CBnCL0 bits.

When the transfer bit length is set to a value other than 16 bits, set the data to the CBnTX or CBnRX register starting from the LSB, regardless of whether the transfer start bit is the MSB or LSB. Any data can be set for the higher bits that are not used, but the receive data becomes 0 following serial transfer.

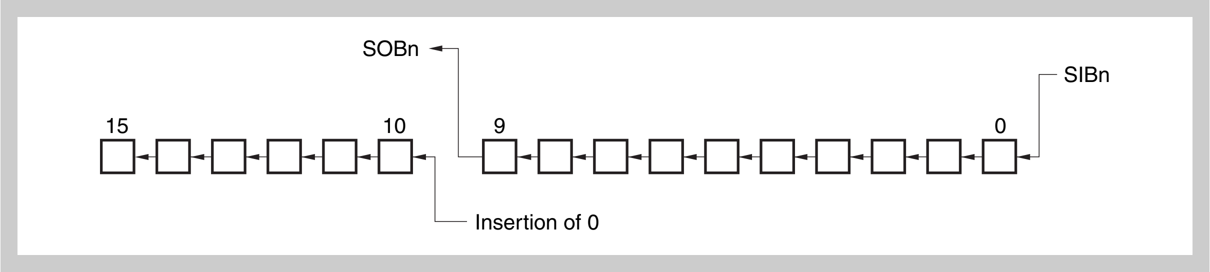


Figure 18-2 (i) Transfer bit length = 10 bits, MSB first

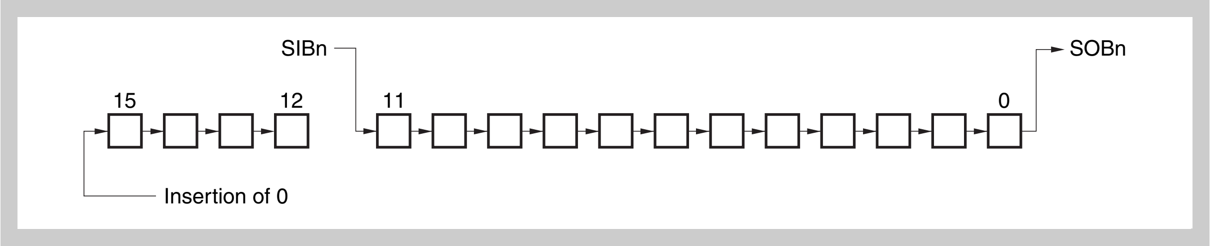


Figure 18-3 (ii) Transfer bit length = 12 bits, LSB first

(4) CBnSTR - CSIBn status register

CBnSTR is an 8-bit register that displays the CSIBn status.

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

Address CB0CTL2: FFFFFD03$_H$
CB1CTL2: FFFFFD13$_H$
CB2CTL2: FFFFFD23$_H$
CB3CTL2: FFFFFD33$_H$

Initial Value 00$_H$. This register is cleared by any reset.
In addition to reset input, the CBnSTR register can be initialized by clearing the CBnCTL0.CBnPWR bit to 0.

	7	6	5	4	3	2	1	0
CBnSTR	CBnTSF	0	0	0	0	0	0	CBnOVE
	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 18-6 CBnSTR register contents

Bit position	Bit name	Function
7	CBnTSF	Communication status flag 0: Communication stopped 1: Communicating During transmission, this register is set when data is prepared in the CBnTX register, and during reception, it is set when a dummy read of the CBnRX register is performed. When transfer ends, this flag is cleared to 0 at the last edge of the clock.
0	CBnOVE	Overrun error flag 0: No overrrun 1: Overrun • An overrun error occurs when the next reception starts without performing a CPU read of the value of the receive buffer, upon completion of the receive operation. The CBnOVE flag displays the overrun error occurrence status in this case. • The CBnOVE flag is cleared by writing 0 to it. It cannot be set even by writing 1 to it.

Note In case of an overrun error, the reception error interrupt INTCBnRE behaves different, depending on the transfer mode:

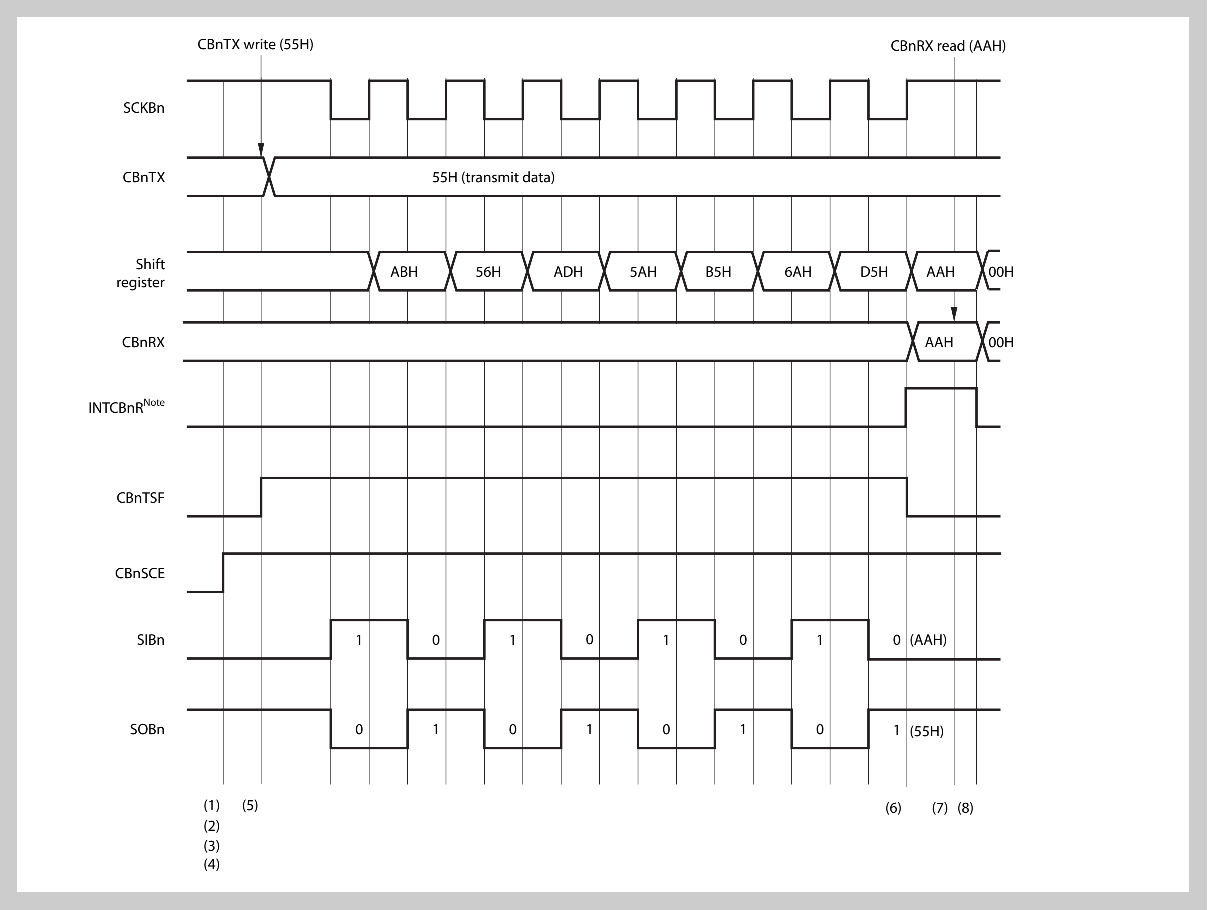
- Continuous transfer mode
  The reception error interrupt INTCBnRE is generated instead of the reception completion interrupt INTCBnR.
- Single transfer mode
  No interrupt is generated.

In either case the overflow flag CBnSTR.CBnOVE is set to 1 and the previous data in CBnRX will be overwritten with the new data.

18.4 Operation

18.4.1 Single transfer mode (master mode, transmission/reception mode)

MSB first (CBnCTL0.CBnDIR bit = 0), communication type 1 (see 18.3 (2) CBnCTL1 - CSIBn control register 1), transfer data length = 8 bits (CBnCTL2.CBnCL[3:0] bits = 0000_B)



(1) Clear the CBnCTL0.CBnPWR bit to 0.

(2) Set the CBnCTL1 and CBnCTL2 registers to specify the transfer mode.

(3) Set the CBnTXE and CBnRXE bits of the CBnCTL0 register to 1 at the same time as specifying the transfer mode using the CBnDIR bit, to set the transmission/reception enabled status.

(4) Set the CBnPWR bit to 1 to enable the CSIBn operation.

(5) Write transfer data to the CBnTX register (transmission start).

(6) The reception complete interrupt request signal (INTCBnR) is output.

(7) Read the CBnRX register before clearing the CBnPWR bit to 0.

(8) Check that the CBnSTR.CBnTSF bit = 0 and set the CBnPWR bit to 0 to stop operation of CSIBn (end of transmission/reception).

To continue transfer, repeat steps (5) to (7) before (8).

In transmission mode or transmission/reception mode, communication is not started by reading the CBnRX register.

Note

1. In single transmission mode the INTCBnT signal is generated. When communication is complete, the INTCBnR signal is generated.
2. The processing of steps (3) and (4) can be set simultaneously.

Caution

In case the CSIB interface is operating in

- single transmit/reception mode (CBnCTL0.CBnTMS = 0)
- communication type 2 respectively type 4 (CBnCTL1.CBnDAP = 1)

pay attention to following effect:

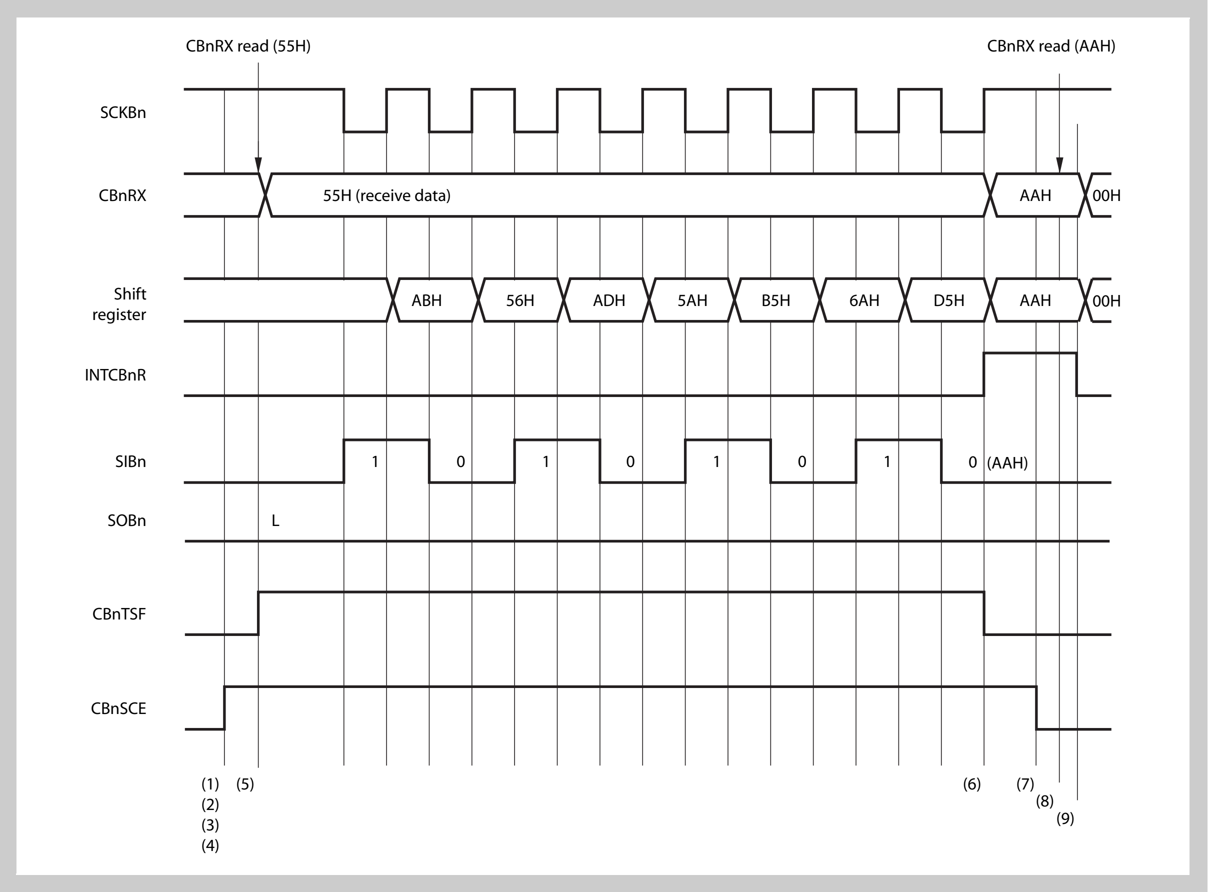
In case the next transmit should be initiated immediately after the occurrence of the reception completion interrupt INTCBnR any write to the CBnTX register is ignored as long as the communication status flag is still reflecting an ongoing communication (CBnTSF = 1). Thus the new transmission will not be started.

For transmitting data continuously use one of the following options:

- Use continuous transfer mode (CBnCTL0.CBnTMS = 1). This is the only usable mode for automatic transmission of data by the DMA Controller.
- If single transfer mode (CBnCTL0.CBnTMS = 0) should be used, CBnSTR.CBnTSF = 0 needs to be verified before writing data to the CBnTX register.

18.4.2 Single transfer mode (master mode, reception mode)

MSB first (CBnCTL0.CBnDIR bit = 0), communication type 1 (see 18.3 (2) CBnCTL1 - CSIBn control register 1), transfer data length = 8 bits (CBnCTL2.CBnCL[3:0] bits = 0000_B)



(1) Clear the CBnCTL0.CBnPWR bit to 0.

(2) Set the CBnCTL1 and CBnCTL2 registers to specify the transfer mode.

(3) Set the CBnCTL0.CBnRXE and CBnCTL0.CBnSCE bits to 1, CBnCTL0.TXE to 0, at the same time as specifying the transfer mode using the CBnDIR bit, to set the reception enabled status.

(4) Set the CBnPWR bit to 1 to enable the CSIBn operation.

(5) Perform a dummy read of the CBnRX register (reception start trigger).

(6) The reception complete interrupt request signal (INTCBnR) is output.

(7) Set the CBnSCE bit to 0 to set the final receive data status.

(8) Read the CBnRX register before setting the CBnPWR bit to "0".

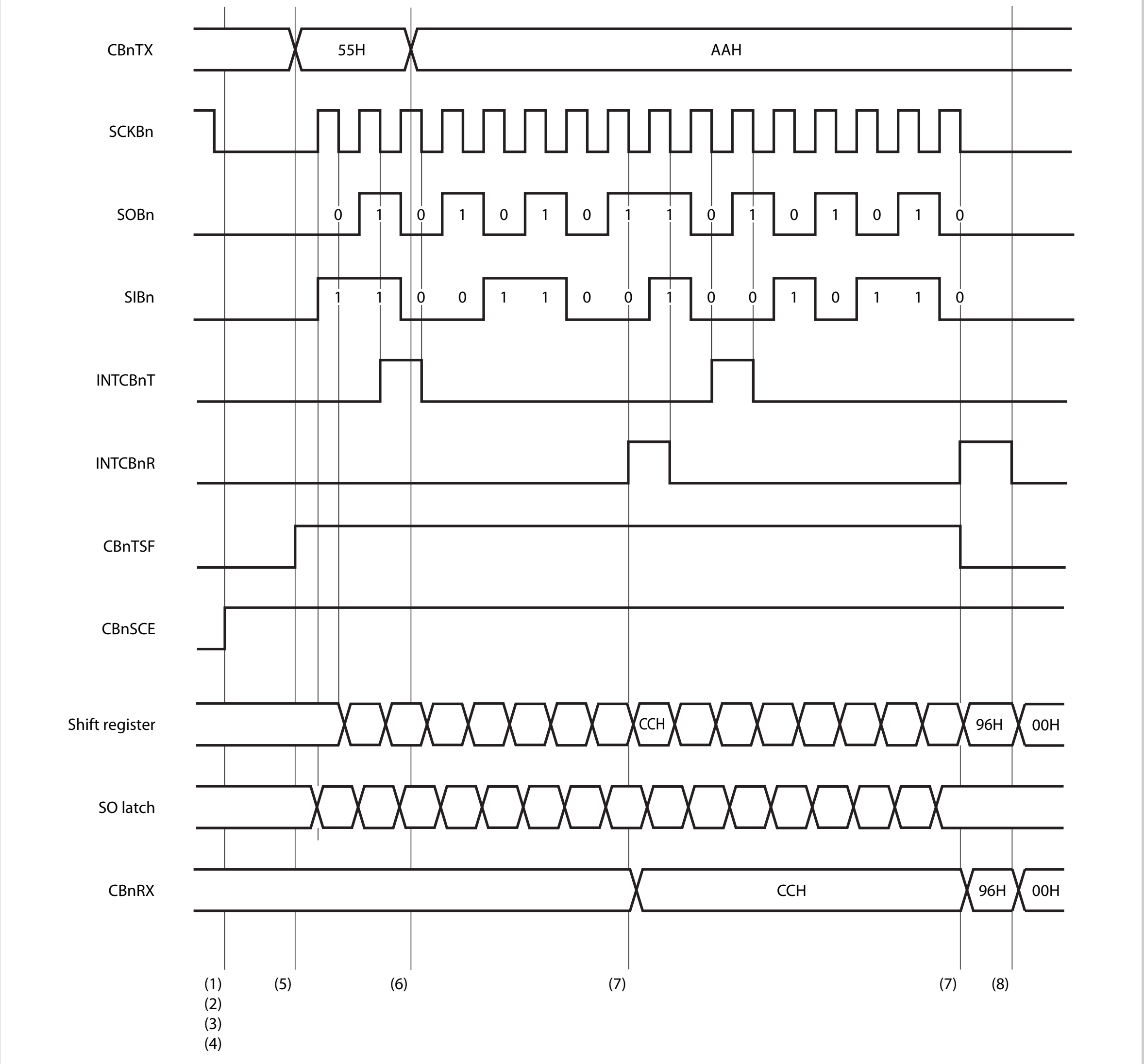
(9) Check that the CBnSTR.CBnTSF bit = 0 and set the CBnPWR bit to 0 to stop the CSIBn operation (end of reception).

To continue transfer, repeat steps (5) and (6) before (7). (At this time, (5) is not a dummy read, but a receive data read combined with the reception trigger.)

Note The processing of steps (3) and (4) can be set simultaneously.

18.4.3 Continuous mode (master mode, transmission/reception mode)

MSB first (CBnCTL0.CBnDIR bit = 0), communication type 3 (see 18.3 (2) CBnCTL1 - CSIBn control register 1), transfer data length = 8 bits (CBnCTL2.CBnCL[3:0] bits = 0000_B)



(1) Clear the CBnCTL0.CBnPWR bit to 0.

(2) Set the CBnCTL1 and CBnCTL2 registers to specify the transfer mode.

(3) Set the CBnTXE, CBnRXE, and CBnSCE bits of the CBnCTL0 register to 1 at the same time as specifying the transfer mode using the CBnDIR bit, to set the transmission/reception enabled status.

(4) Set the CBnPWR bit to 1 to enable the CSIBn operation.

(5) Write transfer data to the CBnTX register (transmission start).

(6) The transmission enable interrupt request signal (INTCBnT) is received and transfer data is written to the CBnTX register.

(7) The reception complete interrupt request signal (INTCBnR) is output.

Read the CBnRX register before the next receive data arrives or before the CBnPWR bit is cleared to 0.

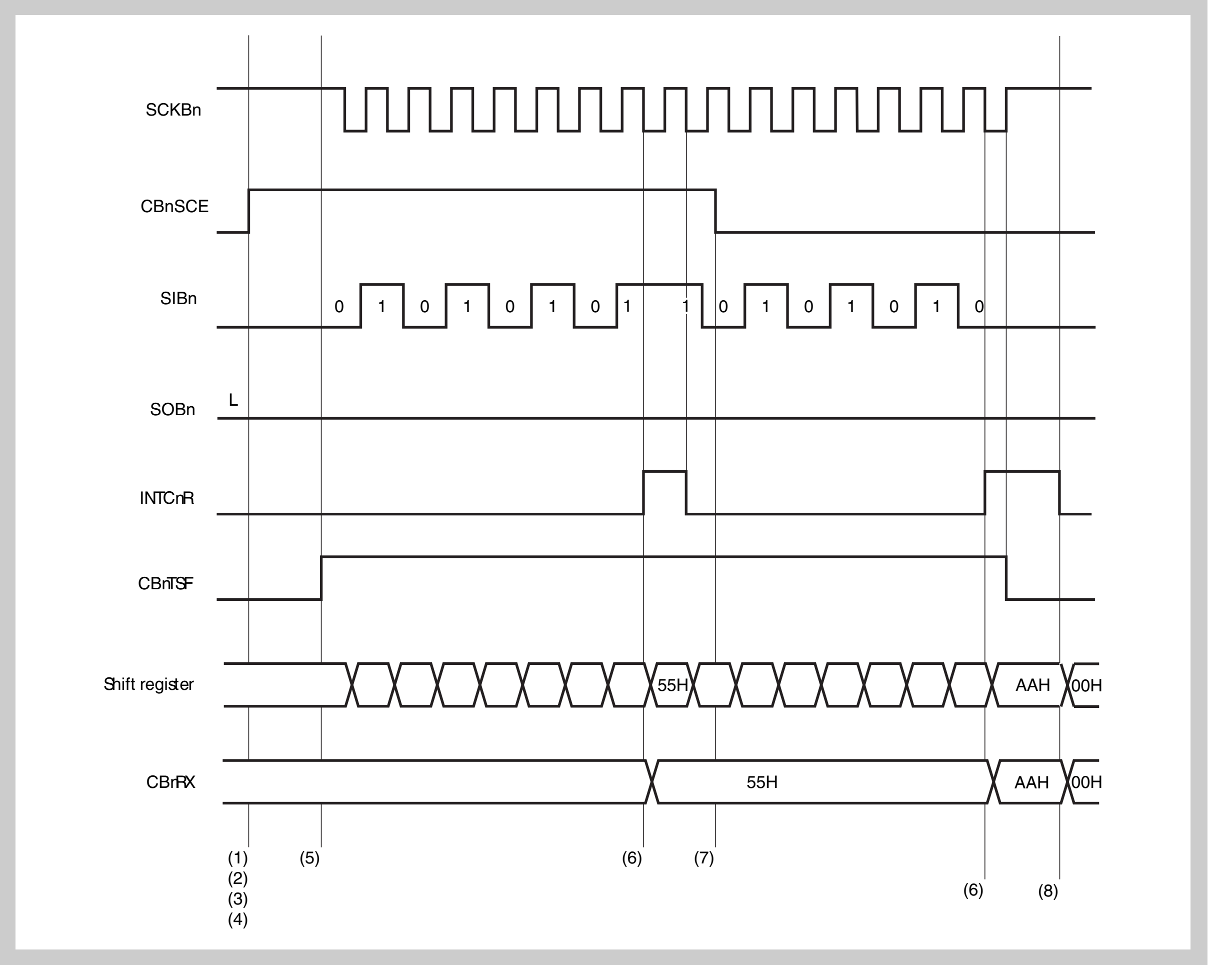
(8) Check that the CBnSTR.CBnTSF bit = 0 and set the CBnPWR bit to 0 to stop the operation of CSIBn (end of transmission/reception).

To continue transfer, repeat steps (5) to (7) before (8).

In transmission mode or transmission/reception mode, the communication is not started by reading the CBnRX register.

18.4.4 Continuous mode (master mode, reception mode)

MSB first (CBnCTL0.CBnDIR bit = 0), communication type 2 (see 18.3 (2) CBnCTL1 - CSIBn control register 1), transfer data length = 8 bits (CBnCTL2.CBnCL[3:0] bits = 0000_B)



(1) Clear the CBnCTL0.CBnPWR bit to 0.

(2) Set the CBnCTL1 and CBnCTL2 registers to specify the transfer mode.

(3) Set the CBnCTL0.CBnRXE bit to 1 at the same time as specifying the transfer mode using the CBnDIR bit, to set the reception enabled status.

(4) Set the CBnPWR bit to 1 to enable the CSIBn operation.

(5) Perform a dummy read of the CBnRX register (reception start trigger).

(6) The reception complete interrupt request signal (INTCBnR) is output.

Read the CBnRX register before the next receive data arrives or before the CBnPWR bit is cleared to 0.

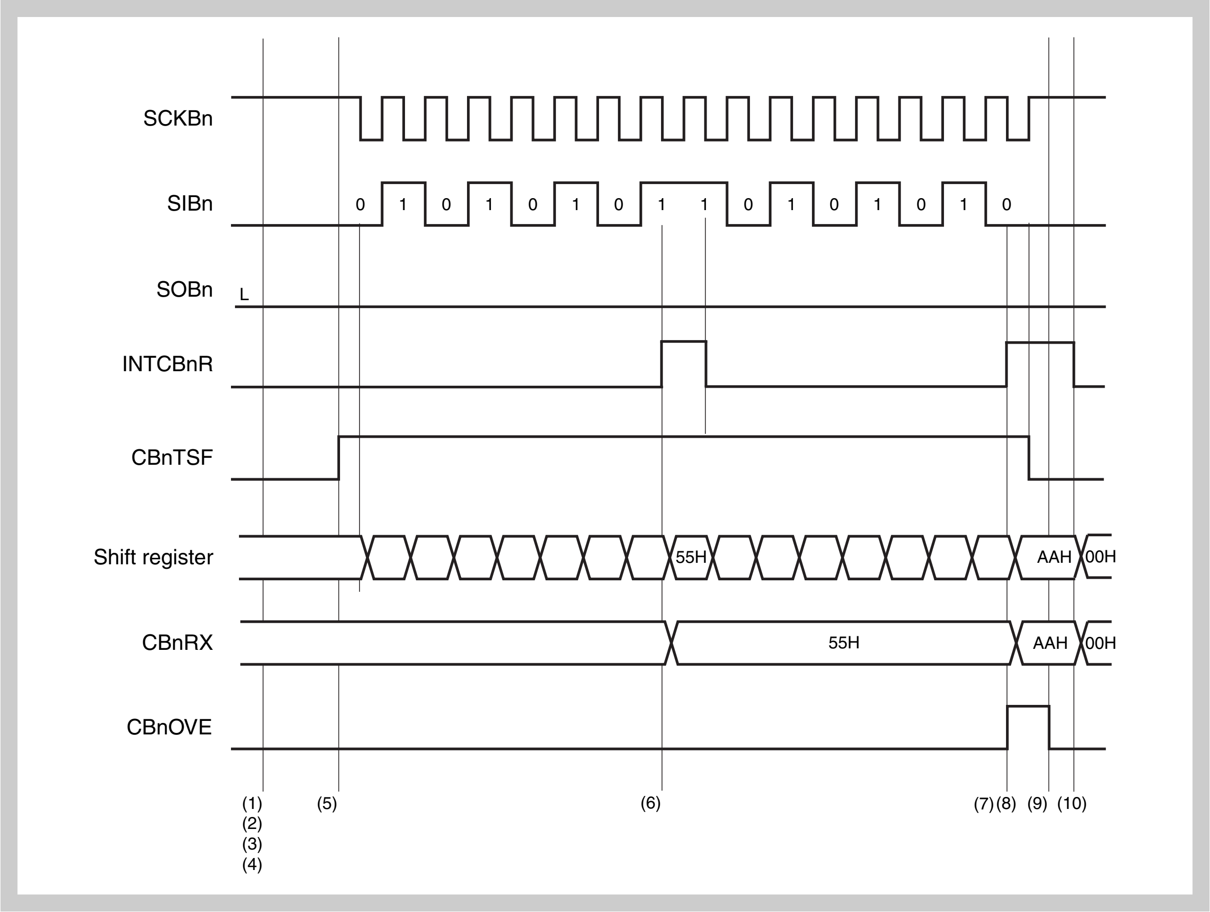
(7) Set the CBnCTL0.CBnSCE bit = 0 while the last data being received to set the final receive data status.

(8) Check that the CBnSTR.CBnTSF bit = 0 and set the CBnPWR bit to 0 to stop the operation of CSIBn (end of reception).

To continue transfer, repeat steps (5) and (6) before (7). At this time, (5) is not a dummy read but read of the received data also functions as a reception trigger.

18.4.5 Continuous reception mode (error)

MSB first (CBnCTL0.CBnDIR bit = 0), communication type 2 (see 18.3 (2) CBnCTL1 - CSIBn control register 1), transfer data length = 8 bits (CBnCTL2.CBnCL[3:0] bits = 0000_B)



(1) Clear the CBnCTL0.CBnPWR bit to 0.

(2) Set the CBnCTL1 and CBnCTL2 registers to specify the transfer mode.

(3) Set the CBnCTL0.CBnRXE bit to 1 at the same time as specifying the transfer mode using the CBnDIR bit, to set the reception enabled status.

(4) Set the CBnPWR bit = 1 to enable CSIBn operation.

(5) Perform a dummy read of the CBnRX register (reception start trigger).

(6) The reception complete interrupt request signal (INTCBnR) is output.

(7) If the data could not be read before the end of the next transfer, the CBnSTR.CBnOVE flag is set to 1 upon the end of reception and the reception interrupt INTCBnR is output again. Read the CBnRX register before the next data is received.

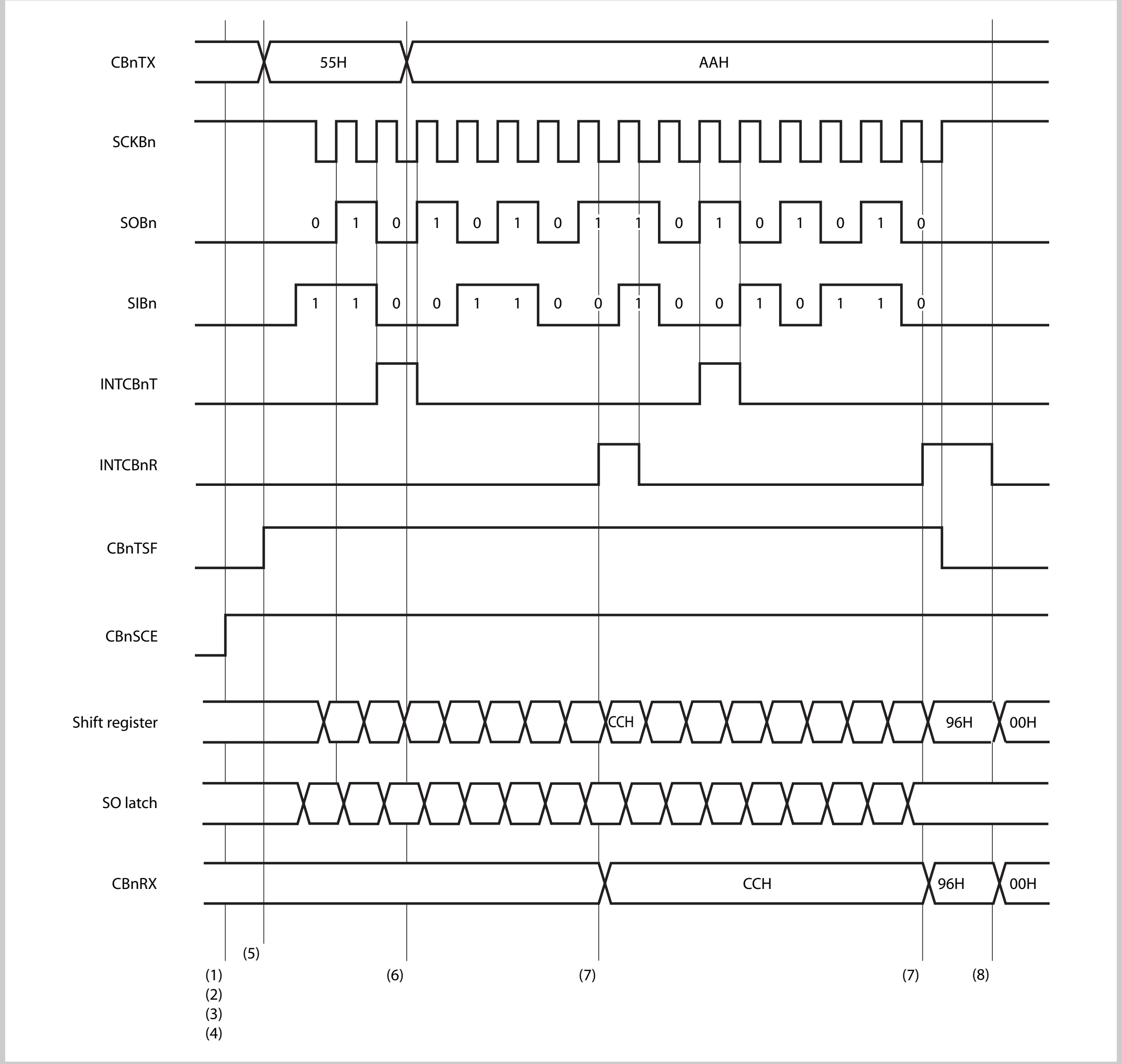
(8) Overrun error processing is performed after checking that the CBnOVE bit = 1 in the INTCBnR interrupt servicing.

(9) Clear CBnOVE bit to 0.

(10)Check that the CBnSTR.CBnTSF bit = 0 and set the CBnPWR bit to 0 to stop the operation CSIBn (end of reception).

18.4.6 Continuous mode (slave mode, transmission/reception mode)

MSB first (CBnCTL0.CBnDIR bit = 0), communication type 2 (see 18.3 (2) CBnCTL1 - CSIBn control register 1), transfer data length = 8 bits (CBnCTL2.CBnCL[3:0] bits = 0000_B)



(1) Clear the CBnCTL0.CBnPWR bit to 0.

(2) Set the CBnCTL1 and CBnCTL2 registers to specify the transfer mode.

(3) Set the CBnTXE, CBnRXE and CBnSCE bits of the CBnCTL0 register to 1 at the same time as specifying the transfer mode using the CBnDIR bit, to set the transmission/reception enabled status.

(4) Set the CBnPWR bit to 1 to enable supply of the CSIBn operation.

(5) Write the transfer data to the CBnTX register (waiting for the serial clock input).

(6) The transmission enable interrupt request signal (INTCBnT) is received and the transfer data is written to the CBnTX register.

(7) The reception complete interrupt request signal (INTCBnR) is output. Read the CBnRX register.

(8) Check that the CBnSTR.CBnTSF bit = 0 and set the CBnPWR bit to 0 to stop the operation of CSIBn (end of transmission/reception).

To continue transfer, repeat steps (5) to (7) before (8).

Note In order to start the entire data transfer the CBnTX register has to be written initially, as done in step (5) above. If this step is omitted also no data will be received.

Discontinued transmission In case the CSIB is operating in continuous slave transmission mode (CBnCTL0.CBnTMS = 1, CBnCTL1.CBnCKS[2:0] = 111_B) and new data is not written to the CBnTX register the SOBn pin outputs the level of the last bit.

Figure 18-4 outlines this behaviour.

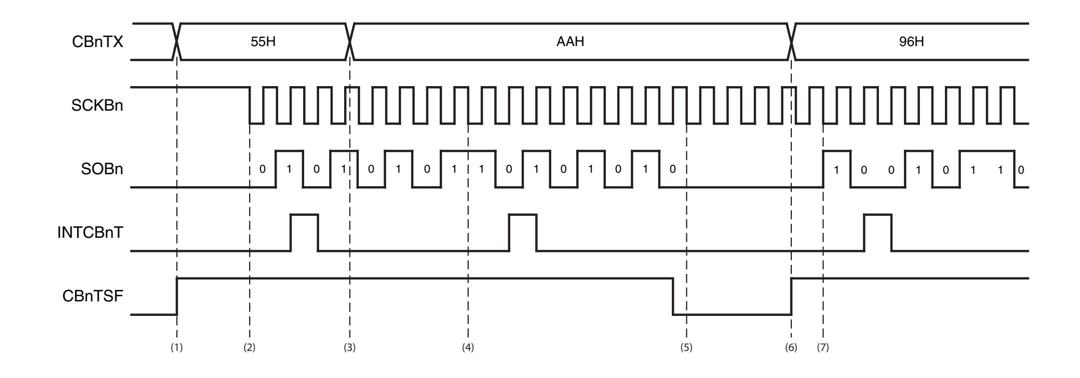


Figure 18-4 Discontinued slave transmission

The example shows the situation that two data bytes (55_H, AA_H) are transmitted correctly, but the third (96_H) fails.

(1) Data 55_H is written (by the CPU or DMA) to CBnTX.

(2) The master issues the clock SCKBn and transmission of 55_H starts.

(3) INTCBnT is generated and the next data AA_H is written to CBnTX promptly, i.e. before the first data has been transmitted completely.

(4) Transmission of the second data AA_H continues correctly and INTCBnT is generated. But this time the next data is not written to CBnTX in time.

(5) Since there is no new data available in CBnTX, but the master continuous to apply SCKBn clocks, SOBn remains at the level of the transmitted last bit.

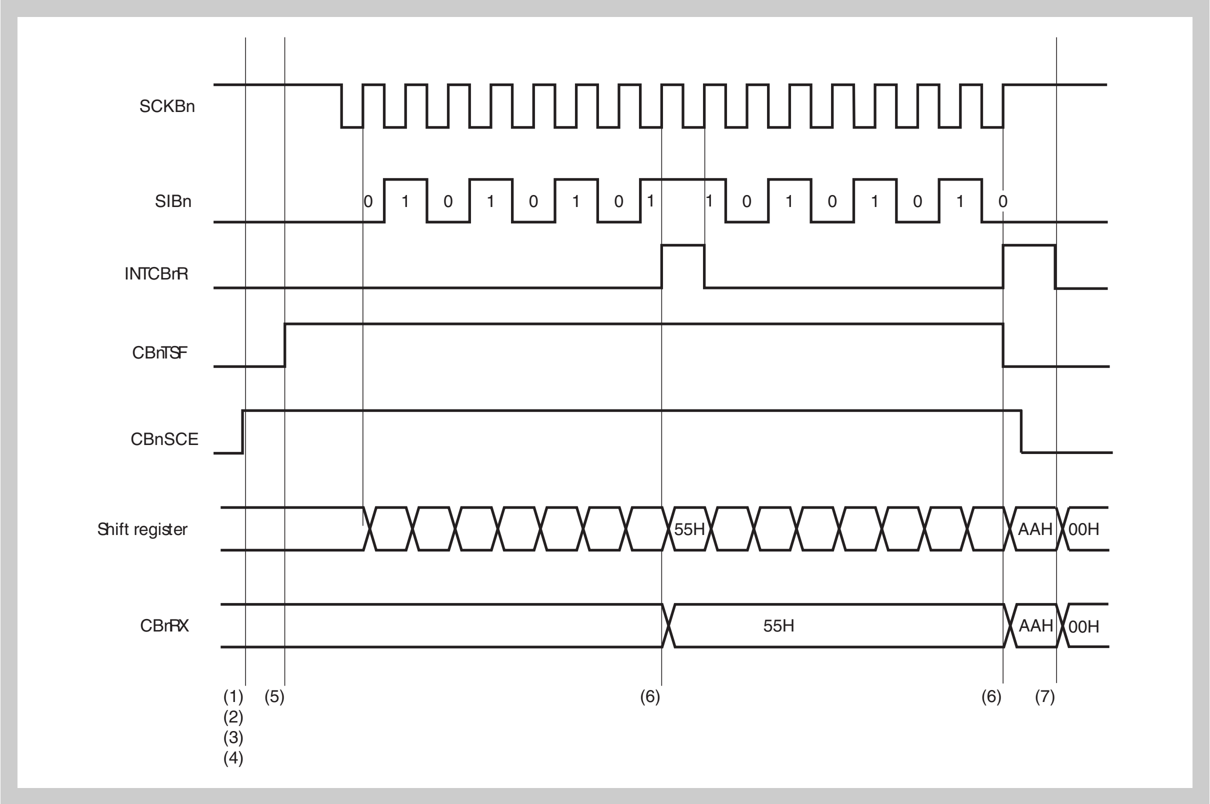
(6) New data (96_H) is written to CBnTX.

(7) With the next SCKBn cycle transmission of the new data (96_H) starts.

As a consequence the master receives a corrupted data byte from (5) onwards, which is made up of a random number of the repeated last bit of the former data and some first bits of the new data.

18.4.7 Continuous mode (slave mode, reception mode)

MSB first (CBnCTL0.CBnDIR bit = 0), communication type 1 (see 18.3 (2) CBnCTL1 - CSIBn control register 1), transfer data length = 8 bits (CBnCTL2.CBnCL[3:0] bits = 0000_B)



(1) Clear the CBnCTL0.CBnPWR bit to 0.

(2) Set the CBnCTL1 and CBnCTL2 registers to specify the transfer mode.

(3) Set the CBnCTL0.CBnRXE and CBnCTL0.CBnSCE bits to 1 and specify the transfer mode using the CBnDIR bit, to set the reception enabled status.

(4) Set the CBnPWR bit = 1 to enable CSIBn operation.

(5) Perform a dummy read of the CBnRX register (waiting for serial clock input).

(6) The reception complete interrupt request signal (INTCBnR) is output.
Read the CBnRX register.
If it is the last data, set the CBnSCE bit to 0, then read the CBnRX register.

(7) Check that the CBnSTR.CBnTSF bit = 0 and set the CBnPWR bit to 0 to stop the operation of CSIBn (end of reception).

To continue transfer, repeat steps (5) and (6) before (7).

18.4.8 Clock timing

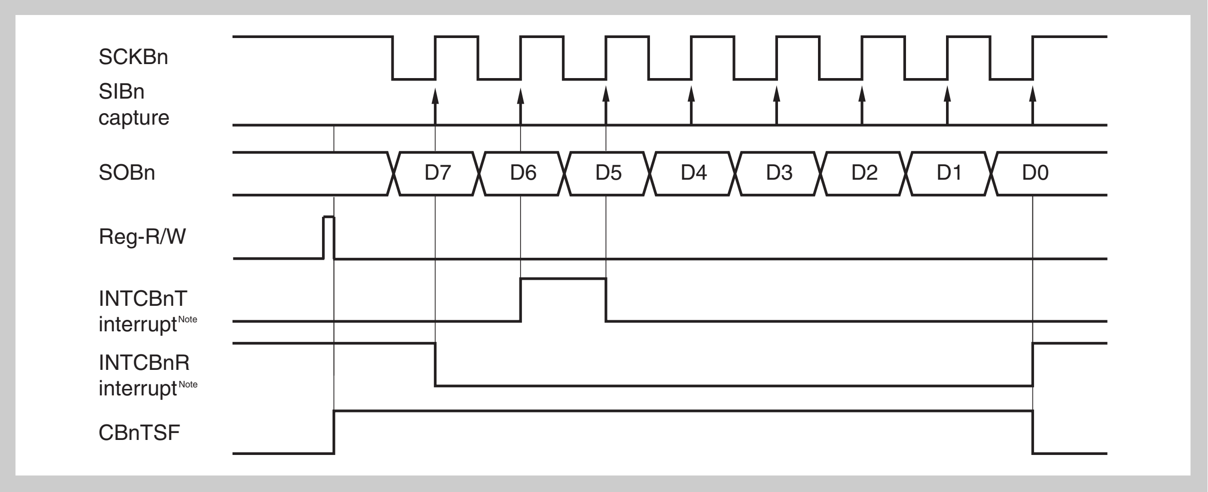


Figure 18-5 (i) Communication type 1 (CBnCKP = 0, CBnDAP = 0)

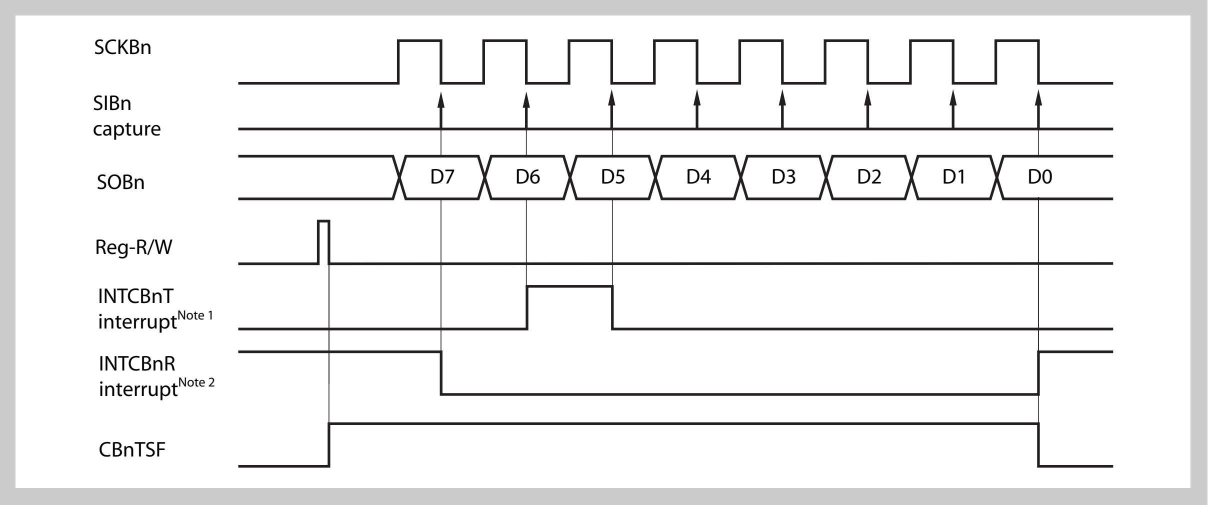


Figure 18-6 (ii) Communication type 3 (CBnCKP = 1, CBnDAP = 0)

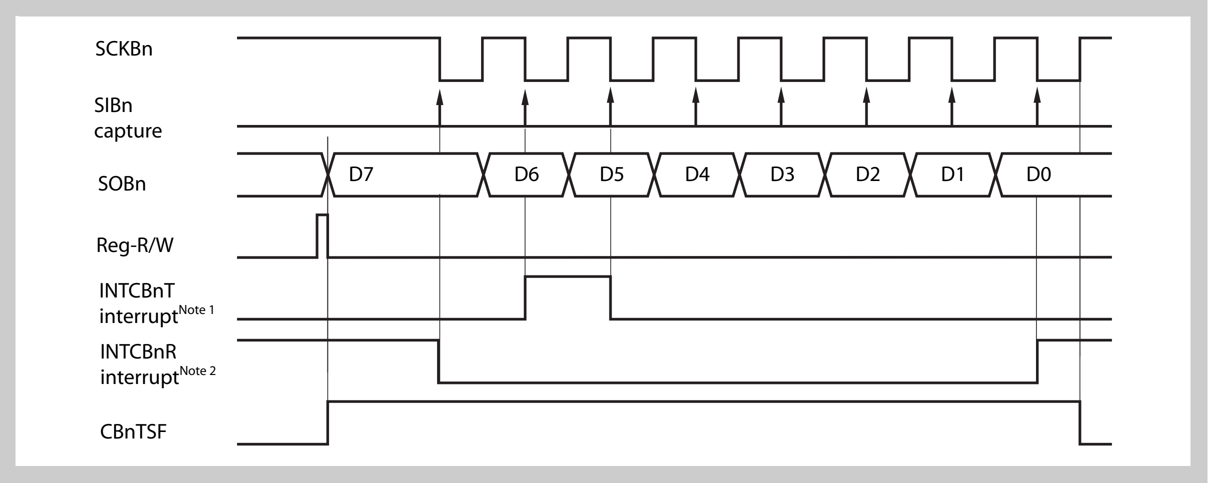


Figure 18-7 (iii) Communication type 2 (CBnCKP = 0, CBnDAP = 1)

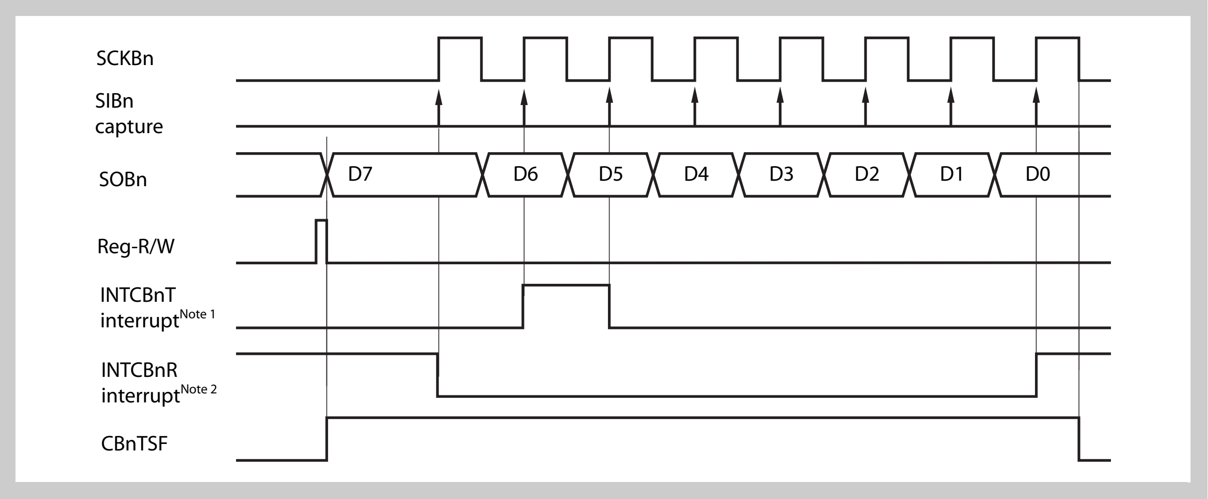


Figure 18-8 (iv) Communication type 4 (CBnCKP = 1, CBnDAP = 1)

Note

1. The INTCBnT interrupt is set when the data written to the transmit buffer is transferred to the data shift register in the continuous transmission or continuous transmission/reception modes. In the single transmission or single transmission/reception modes, the INTCBnT interrupt request signal is not generated, but the INTCBnR interrupt request signal is generated upon completion of communication.
2. The INTCBnR interrupt occurs if reception is correctly completed and receive data is ready in the CBnRX register while reception is enabled, and if an overrun error occurs. In the single mode, the INTCBnR interrupt request signal is generated even in the transmission mode, upon completion of communication.

18.5 Output Pins

(1) SCKBn pin

When CSIBn operation is disabled (CBnCTL0.CBnPWR bit = 0), the SCKBn pin output status is as follows.

CBnCKP	CBnCKS2	CBnCKS1	CBnCKS0	SCKBn pin output
0	Don’t care	Don’t care	Don’t care	Fixed to high level
1	1	1	1	High impedance
	Other than above			Fixed to low level

Note The output level of the SCKBn pin changes if any of the CBnCTL1.CBnCKP and CBnCKS2 to CBnCKS0 bits is rewritten.

(2) SOBn pin

When CSIBn operation is disabled (CBnPWR bit = 0), the SOBn pin output status is as follows.

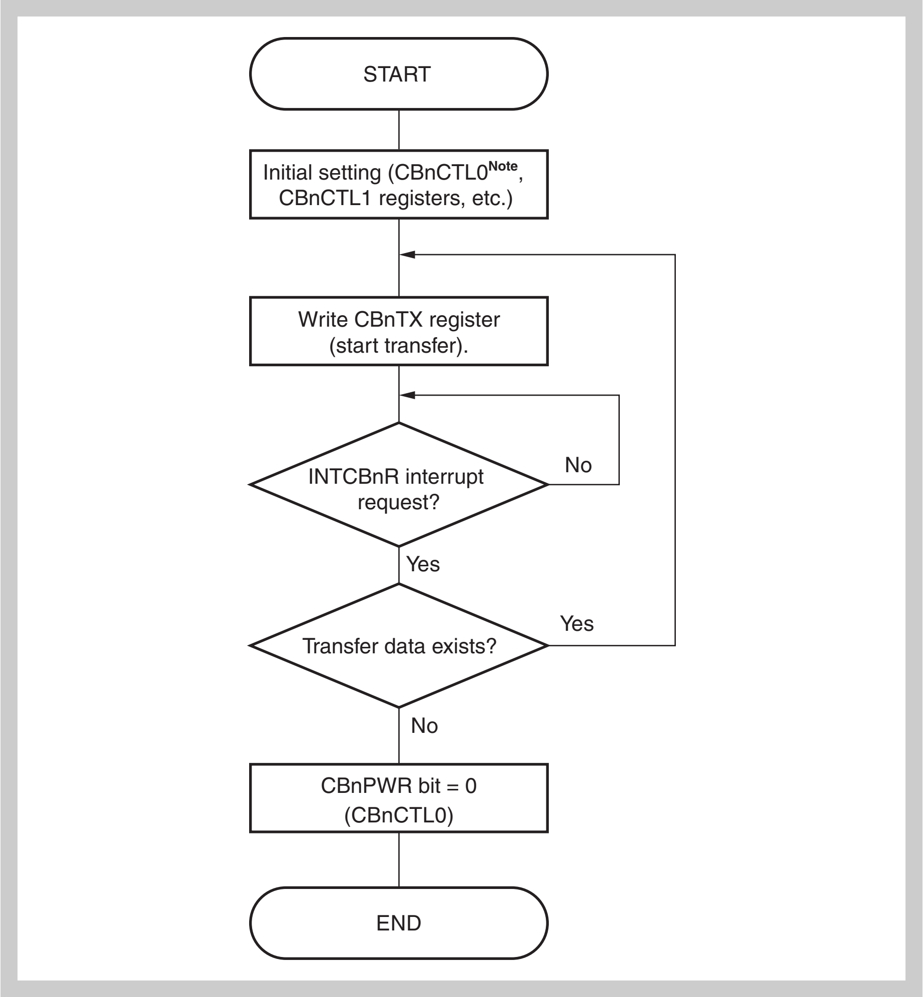
CBnTXE	CBnDAP	CBnDIR	SOBn pin output
0	×	×	Fixed to low level
1	0	×	SOBn latch value (low level)
	1	0	CBnTXn value (MSB)
		1	CBnTXn value (LSB)

Note 1. The SOBn pin output changes when any one of the CBnCTL0.CBnTXE, CBnCTL0.CBnDIR bits, and CBnCTL1.CBnDAP bit is rewritten.

2. ×: don’t care

18.6 Operation Flow

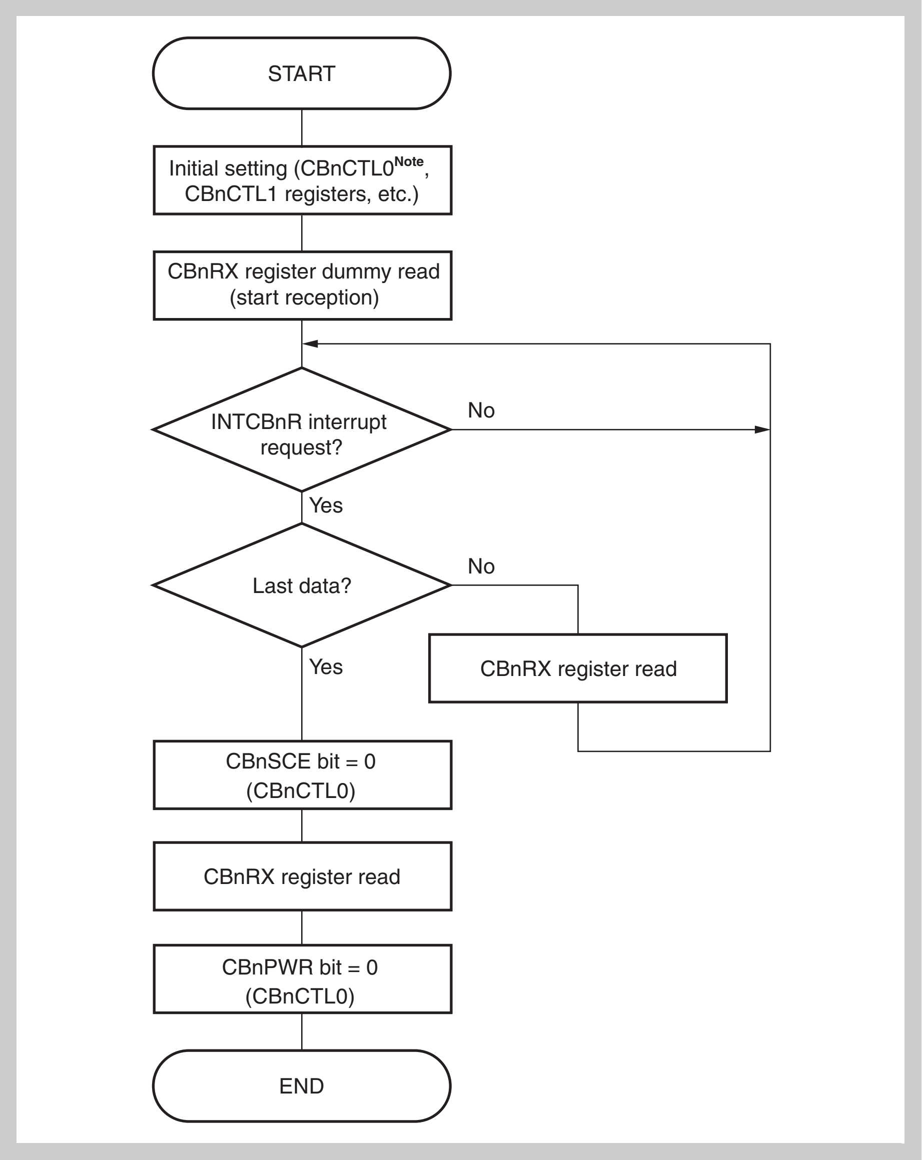
(1) Single transmission


START
Initial setting (CBnCTL0Note, CBnCTL1 registers, etc.)
Write CBnTX register (start transfer).
INTCBnR interrupt request?
No
Yes
Transfer data exists?
Yes
No
CBnPWR bit = 0 (CBnCTL0)
END

Note Set the CBnSCE bit to 1 in the initial setting.

Caution In the slave mode, data cannot be correctly transmitted if the next transfer clock is input earlier than the CBnTX register is written.

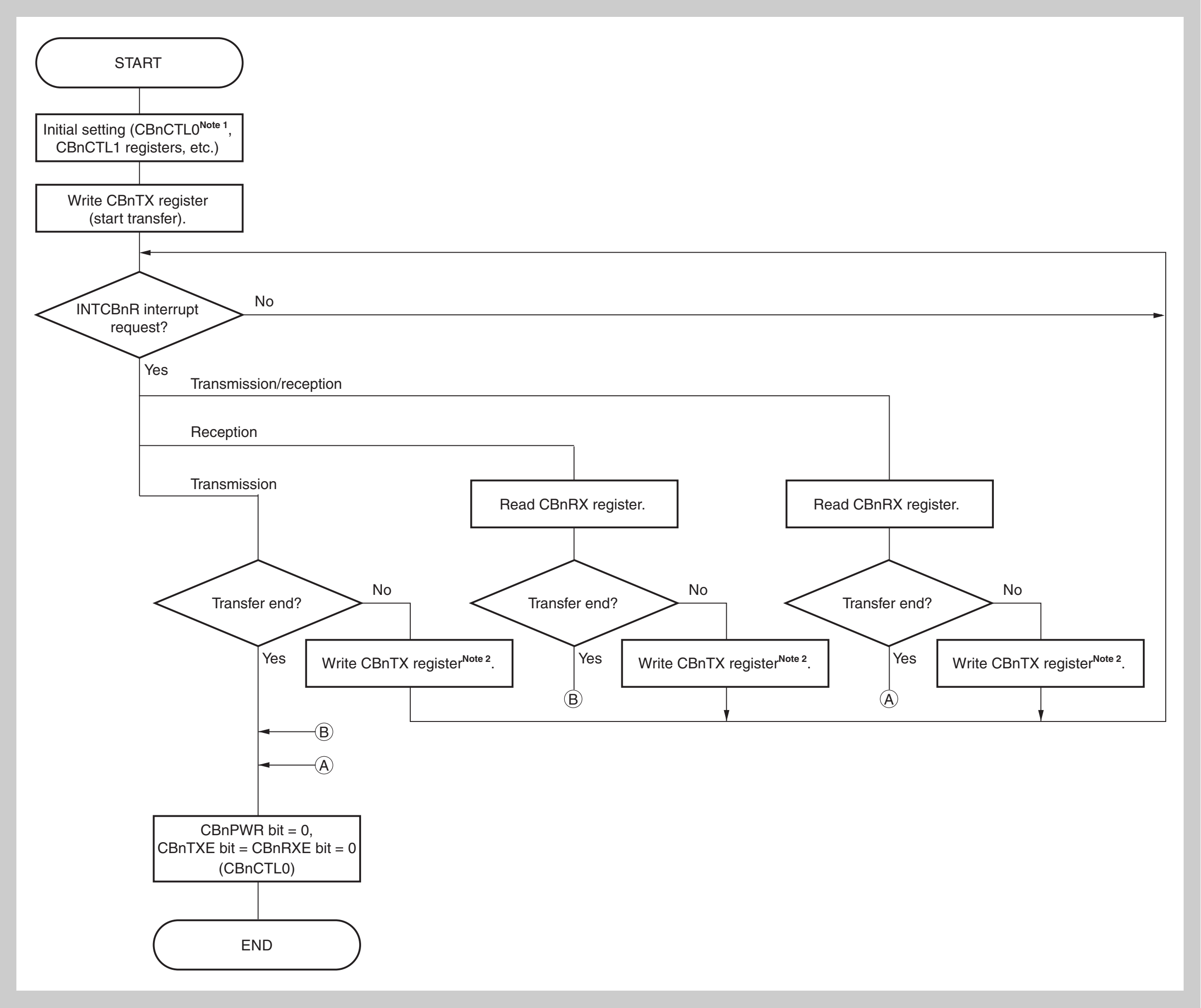
(2) Single reception


START
Initial setting (CBnCTL0Note, CBnCTL1 registers, etc.)
CBnRX register dummy read (start reception)
INTCBnR interrupt request?
No
Yes
Last data?
No
CBnRX register read
Yes
CBnSCE bit = 0 (CBnCTL0)
CBnRX register read
CBnPWR bit = 0 (CBnCTL0)
END

Note Set the CBnSCE bit to 1 in the initial setting.

Caution In the single mode, data cannot be correctly received if the next transfer clock is input earlier than the CBnRX register is read.

(3) Single transmission/reception


START
Initial setting (CBnCTL0Note 1, CBnCTL1 registers, etc.)
Write CBnTX register (start transfer).
INTCBnR interrupt request?
No
Yes
Transmission/reception
Reception
Transmission
Read CBnRX register.
Read CBnRX register.
Transfer end?
No
Yes
Write CBnTX registerNote 2.
B
A
CBnPWR bit = 0, CBnTXE bit = CBnRXE bit = 0 (CBnCTL0)
END

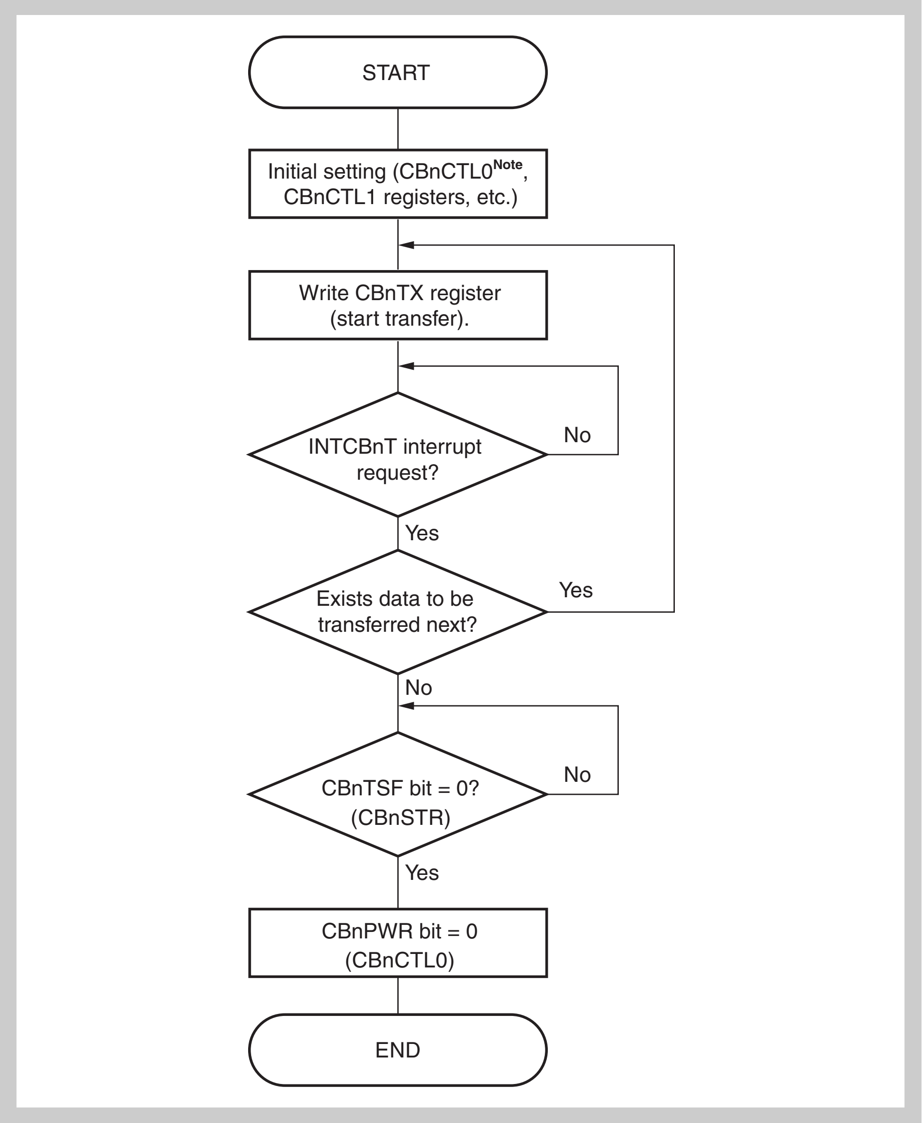
Note 1. Set the CBnSCE bit to 1 in the initial setting.

2. If the next transfer is reception only, dummy data is written to the CBnTX register.

Caution 1. Even in the single mode, the CBnSTR.CBnOVE flag is set to 1. If only transmission is used in the transmission/reception mode, therefore, programming without checking the CBnOVE flag is recommended.

2. In the slave mode, data cannot be correctly transmitted if the next transfer clock is input earlier than the CBnTX register is written.

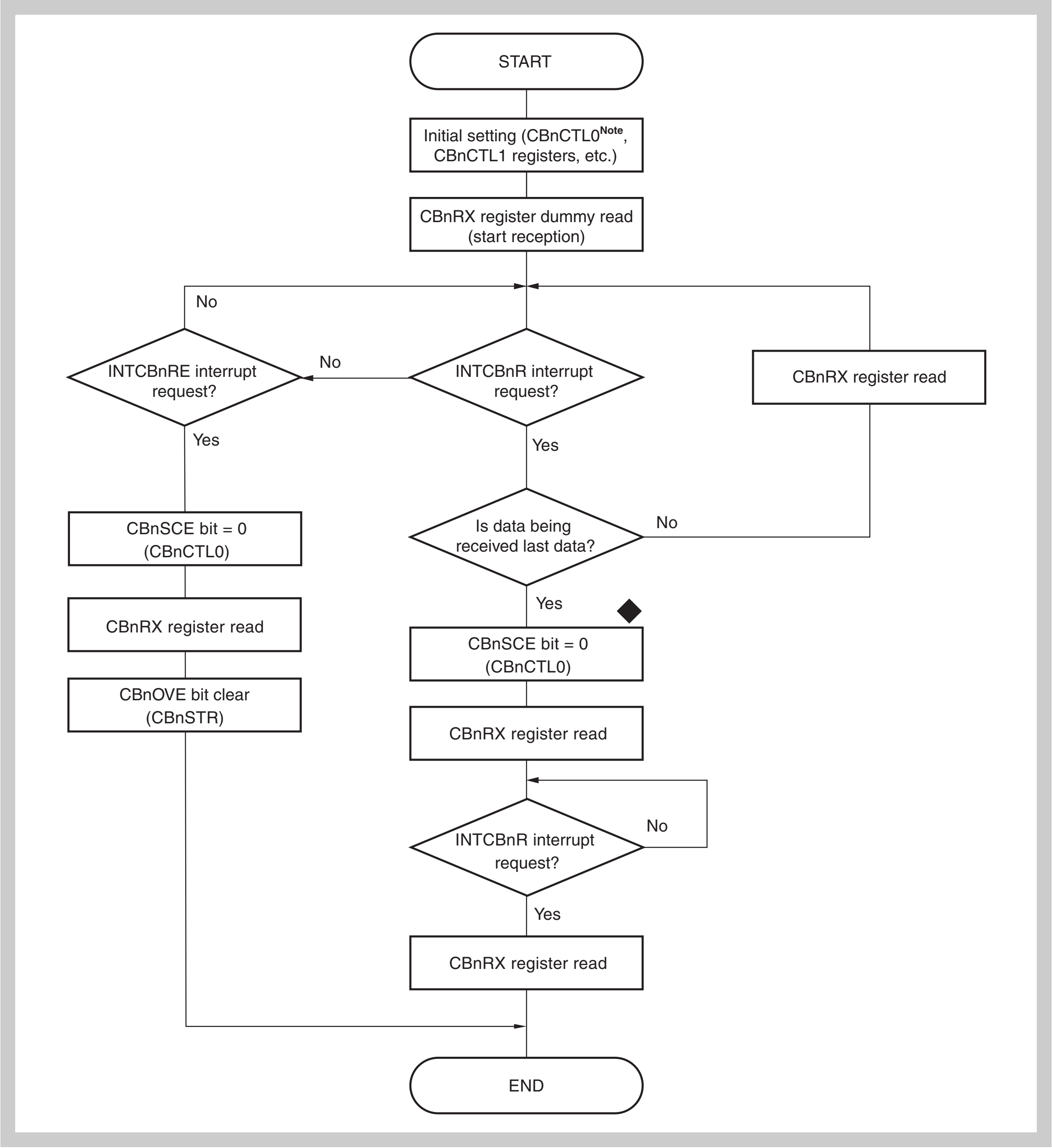
(4) Continuous transmission


START
Initial setting (CBnCTL0Note, CBnCTL1 registers, etc.)
Write CBnTX register (start transfer).
INTCBnT interrupt request?
No
Yes
Exists data to be transferred next?
Yes
No
CBnTSF bit = 0? (CBnSTR)
No
Yes
CBnPWR bit = 0 (CBnCTL0)
END

Note Set the CBnSCE bit to 1 in the initial setting.

Caution In the slave mode, data cannot be correctly transmitted if the next transfer clock is input earlier than the CBnTX register is written.

(5) Continuous reception

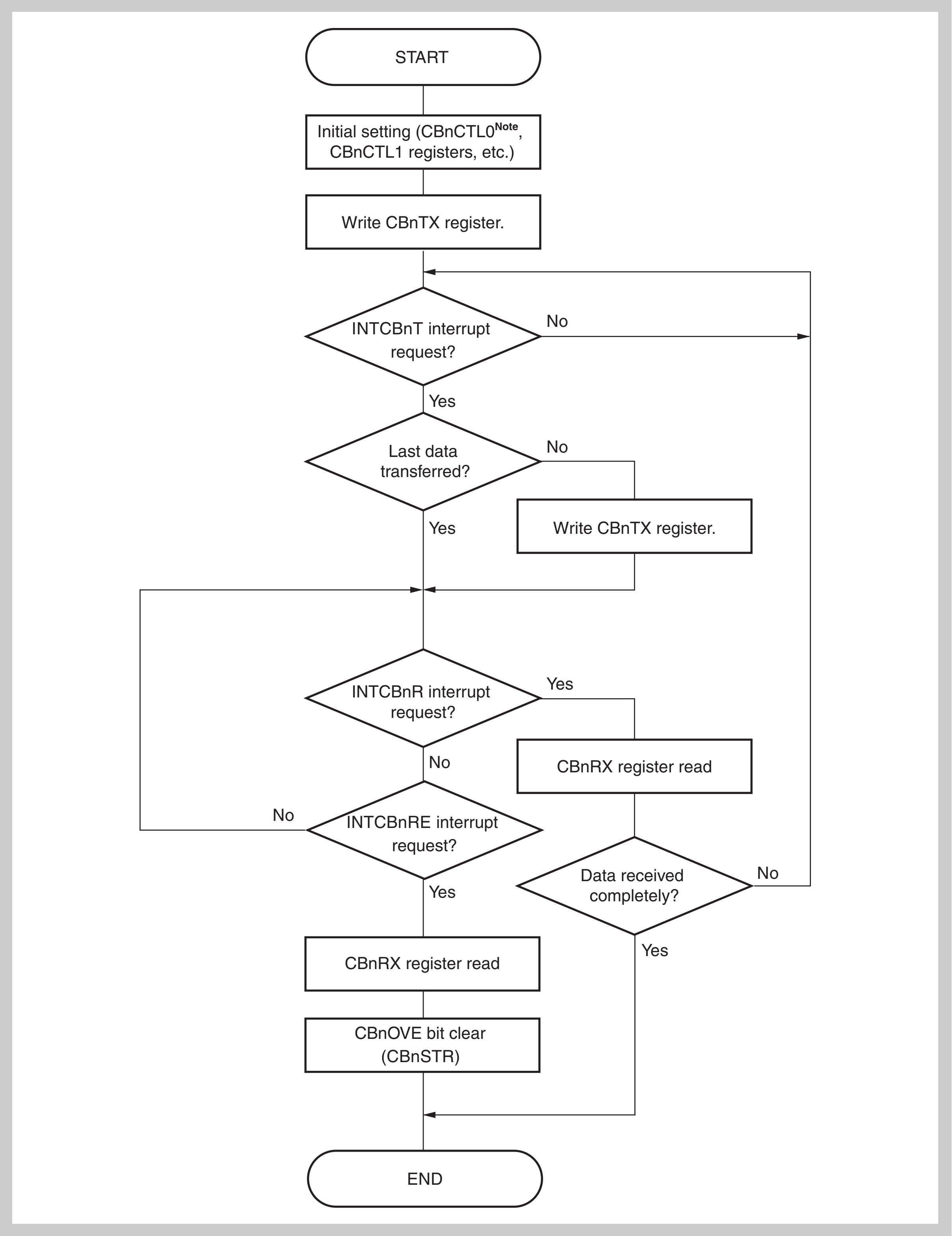

START
Initial setting (CBnCTL0Note, CBnCTL1 registers, etc.)
CBnRX register dummy read (start reception)
No
INTCBnRE interrupt request?
No
INTCBnR interrupt request?
CBnRX register read
Yes
Yes
CBnSCE bit = 0 (CBnCTL0)
Is data being received last data?
No
CBnRX register read
Yes
◆
CBnSCE bit = 0 (CBnCTL0)
CBnOVE bit clear (CBnSTR)
CBnRX register read
INTCBnR interrupt request?
No
Yes
CBnRX register read
END

Note Set the CBnSCE bit to 1 in the initial setting.

Caution

1. In the master mode, the clock is output without limit when dummy data is read from the CBnRX register. To stop the clock, execute the flow marked ◆ in the above flowchart.
   In the slave mode, malfunction due to noise during communication can be prevented by executing the flow marked ◆ in the above flowchart.
   Before resuming communication, set the CBnCTL0.CBnSCE bit to 1, and read dummy data from the CBnRX register.
2. In the slave mode, data cannot be correctly transmitted if the next transfer clock is input earlier than the CBnTX register is written.

(6) Continuous transmission/reception


START
Initial setting (CBnCTL0Note, CBnCTL1 registers, etc.)
Write CBnTX register.
INTCBnT interrupt request?
No
Yes
Last data transferred?
No
Yes
Write CBnTX register.
INTCBnR interrupt request?
Yes
No
CBnRX register read
No
INTCBnRE interrupt request?
Yes
Data received completely?
No
Yes
CBnRX register read
CBnOVE bit clear (CBnSTR)
END

Note 1. Set the CBnSCE bit to 1 in the initial setting.

2. In the slave mode, data cannot be correctly transmitted if the next transfer clock is input earlier than the CBnTX register is written.

Caution

1. When transferring transmit data and receive data using DMA transfer, error processing cannot be performed even if an overrun error occurs during serial transfer. Check that the no overrun error has occurred by reading the CBnSTR.CBnOVE bit after DMA transfer has been completed.
2. In regards to registers that are forbidden from being rewritten during operations (CBnCTL0.CBnPWR bit is 1), if rewriting has been carried out by mistake during operations, set the CBnCTL0.CBnPWR bit to 0 once, then initialize CSIBn. Registers to which rewriting during operation are prohibited are shown below.
   · CBnCTL0 register: CBnTXE, CBnRXE, CBnDIR, CBnTMS bit
   · CBnCTL1 register: CBnCKP, CBnDAP, CBnCKS[2:0] bits
   · CBnCTL2 register: CBnCL[3:0] bits
3. In the single transfer mode (CBnCTL0.CBnTMS bit = 0), when the CBnCTL1.CBnDAP bit is set to 1 and the next reception/transmission are started by using the receive completion interrupt INTCBnR, the reception/ transmission operations from the second time are not performed for 0.5 clocks of the SCKBn after the receive completion interrupt INTCBnR is generated. To perform the continuous transfer, use the continuous transfer mode.
4. When CSIBn is operated in slave mode input an external clock via SCKBn pin only after the transmission -/and/or reception process is started.

Chapter 19 I^{2}C Bus (IIC)

This microcontroller has one instance of this I^{2}C Bus interface.

Note Throughout this chapter, the individual instances of this I^{2}C Bus interface identified by “n” (IICn, n = 0).

19.1 Features

The I^{2}C Bus interface provides a synchronous serial interface with the following features:

- Supports Master and Slave mode
- 8-bit data transfer
- Transfer speed
  - up to 100 kbit/s (Standard Mode)
  - up to 400 kbit/s (Fast Mode)
- Two wire interface
  - SCL0n: serial clock
  - SDA0n: serial data
- Noise filter on SCL0n and SDA0n input

19.2 I^{2}C Pin Configuration

The I^{2}C function requires to define the pins SCL0n and SDA0n as input and open drain output pins simultaneously. In the following, the pin configuration registers are listed to be set up properly for IICn:

- PMC9.PMC914, PMC9.PMC915 = 1: alternative mode
- PFCE9.PFCE914 = PFCE9.PFCE915 = 1, together with
  PFC9.PFC914 = PFC9.PFC915 = 0: select alternative function 3
- PF9.PFC914 = PF9.PFC915 = 1: open drain output

19.3 Configuration

The block diagram of the IICn is shown below.

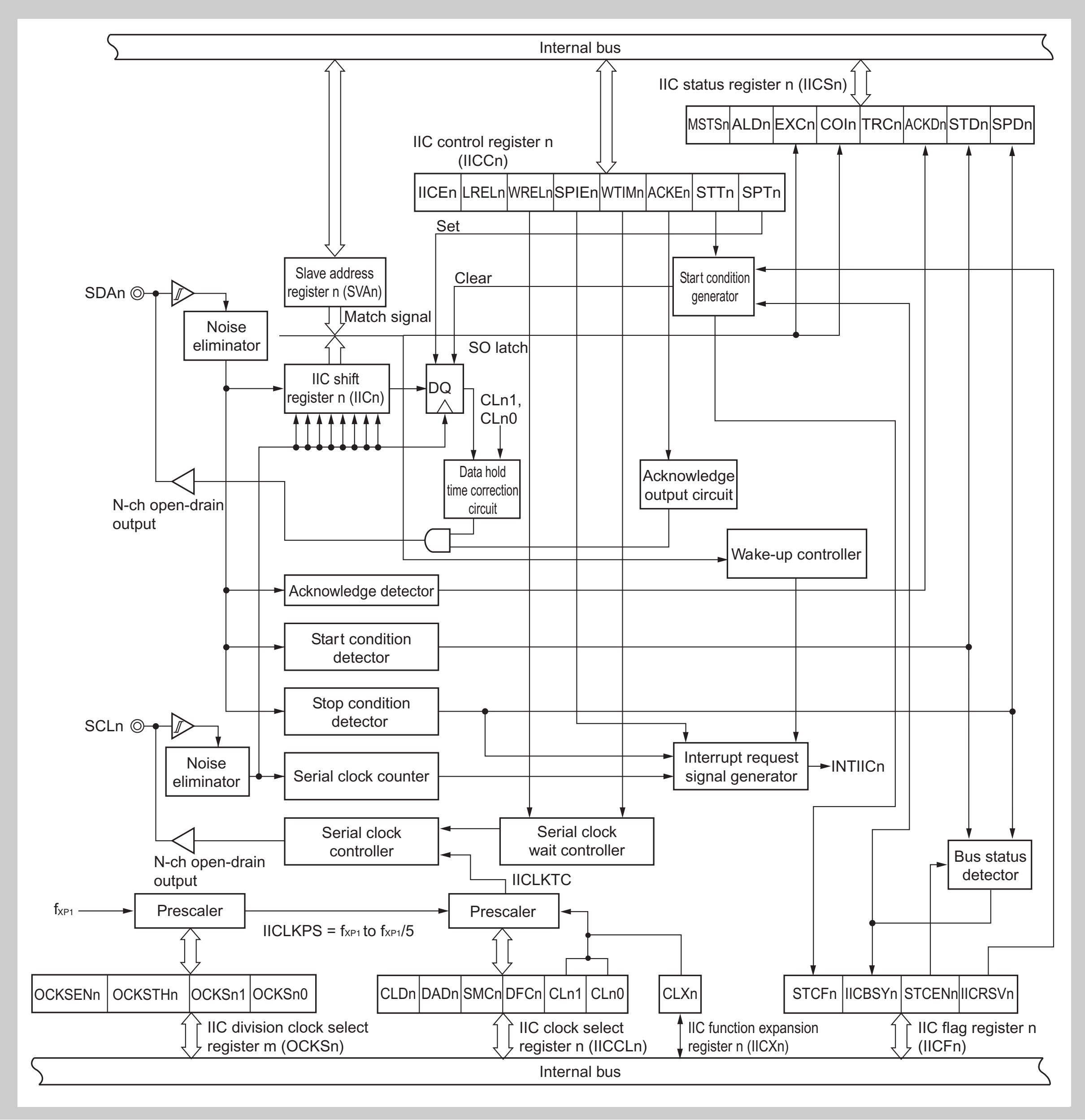


Figure 19-1 Block diagram of IICn

A serial bus configuration example is shown below.

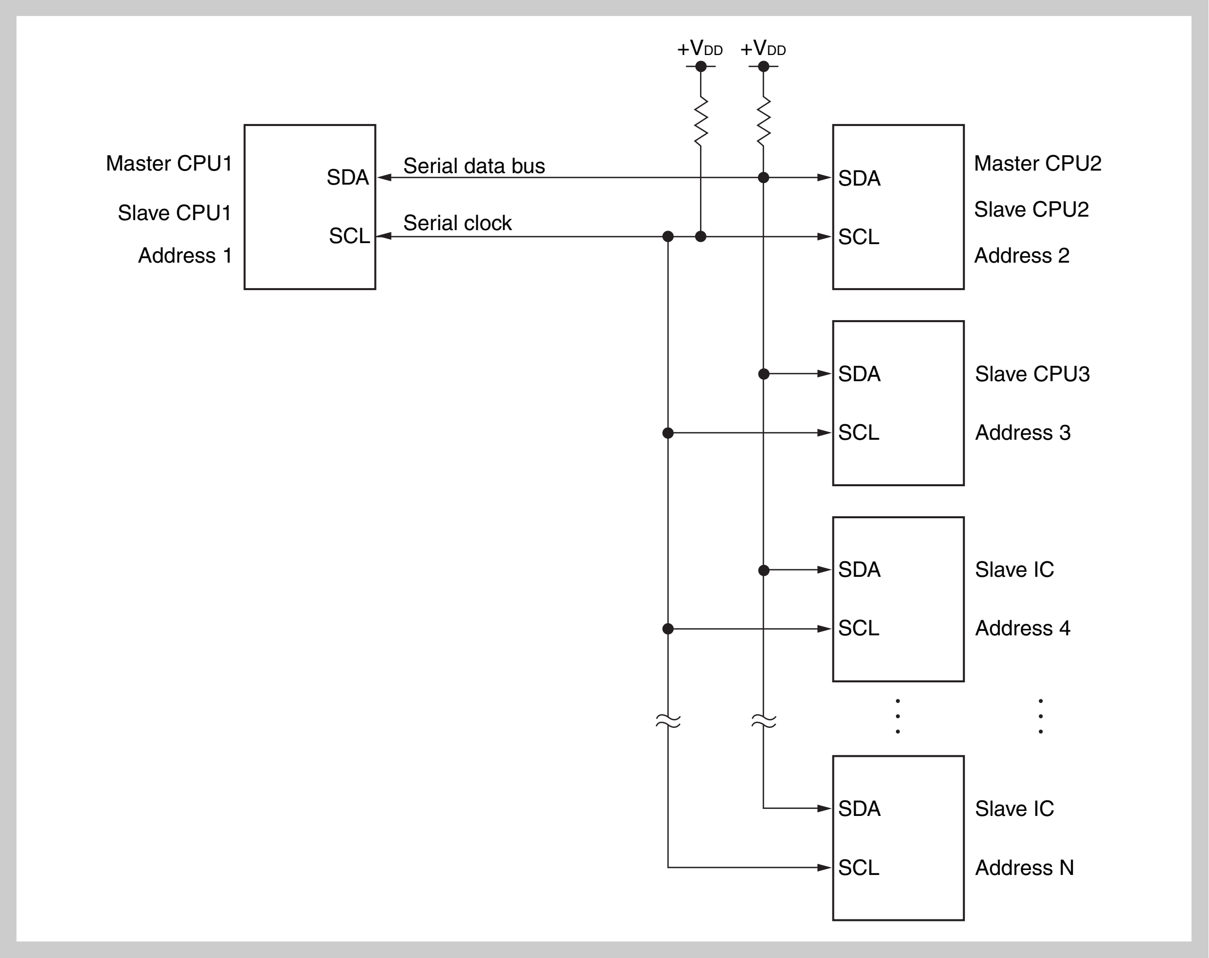


Figure 19-2 Serial bus configuration example using I^{2}C bus

IICn includes the following hardware.

Table 19-1 Configuration of IICn

Item	Configuration
Registers	IIC shift register n (IICn) Slave address register n (SVAn)
Control registers	IIC control register n (IICCn) IIC status register n (IICSn) IIC flag register n (IICF0n) IIC clock select register n (IICCLn) IIC function expansion register n (IICXn) IIC division clock select registers (OCKSn)

(1) IIC shift register n (IICn)

The IICn register converts 8-bit serial data into 8-bit parallel data and vice versa, and can be used for both transmission and reception.

Write and read operations to the IICn register are used to control the actual transmit and receive operations.

(2) Slave address register n (SVAn)

The SVAn register sets local addresses when in slave mode.

(3) SO latch

The SO latch is used to retain the output level of the SDA0n pin.

(4) Wakeup controller

This circuit generates an interrupt request when the address received by this register matches the address value set to the SVAn register or when an extension code is received.

(5) Prescaler

This selects the sampling clock to be used.

(6) Serial clock counter

This counter counts the serial clocks that are output and the serial clocks that are input during transmit/receive operations and is used to verify that 8-bit data was transmitted or received.

(7) Interrupt request signal generator

This circuit controls the generation of interrupt request signals (INTIICn).

An I^{2}C interrupt is generated following either of two triggers:

- Falling edge of eighth or ninth clock of the serial clock (set by IICCn.WTIMn bit)
- Interrupt occurrence due to stop condition detection (set by IICCn.SPIEn bit)

(8) Serial clock controller

In master mode, this circuit generates the clock output via the SCL0n pin from the sampling clock.

(9) Serial clock wait controller

This circuit controls the wait timing.

(10) $\overline{\text{ACK}}$ output circuit, stop condition detector, start condition detector, and $\overline{\text{ACK}}$ detector

These circuits are used to output and detect various control signals.

(11) Data hold time correction circuit

This circuit generates the hold time for data corresponding to the falling edge of the SCL0n pin.

(12) Start condition generator

A start condition is issued when the IICCn.STTn bit is set.

However, in the communication reservation disabled status (IICFn.IICRSVn = 1), this request is ignored and the IICFn.STCFn bit is set if the bus is not released (IICFn.IICBSYn = 1).

(13) Bus status detector

Whether the bus is released or not is ascertained by detecting a start condition and stop condition.

However, the bus status cannot be detected immediately after operation, so set the bus status detector to the initial status by using the IICFn.STCENn bit.

(14) Stop condition generator

A stop condition is generated when the IICC0.SPT0 bit is set.

19.4 IIC Registers

The I^2C interfaces are controlled by the following registers.

- IIC control register IICCn
- IIC status register IICSn
- IIC flag register IICFn
- IIC clock select register IICCLn
- IIC function expansion register IICXn
- IIC division clock select register OCKSn

The following registers are also used.

- IIC shift register IICn
- Slave address register SVAn

(1) IICCn - IICn control registers

The IICCn register enables/stops IICn operations, sets the wait timing, and sets other I^{2}C operations.

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

Address FFFFD82$_H$

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

	7	6	5	4	3	2	1	0
IICCn	IICEn	LRELn	WRELn	SPIEn	WTIMn	ACKEn	STTn	SPTn
	R/W	R/W	R/W	R/W	R	R	R/W	R/W

Caution Set the SPIEn, WTIMn, and ACKEn bits when the IICEn bit is 0 or during the wait period. When setting the IICEn bit from "0" to "1", these bits can also be set at the same time.

Table 19-2 IICCn register contents (1/4)

Bit position	Bit name	Function
7	IICEn	Specification of I2Cn operation enable/disable: 0: Operation stopped. IICSn register reset[Note]. Internal operation stopped. 1: Enable CSIBn operation **Caution:** Be sure to set the IICn bit to 1 when the SCL0n and SDA0n lines are high level. **Note:** Whenn IICEn is cleared to 0, the IICS register, IICFn.STCFn and IICFn.IICBSYn bits, and IICCLn.CLDn and IICCLn.DADn bits are reset.
6	LRELn	Exit from communications: 0: Normal operation 1: Exits from the current communication operation and sets stand-by mode. This setting is automatically cleared after being executed. Its uses include cases in which a locally irrelevant extension code has been received. The SCL0n and SDA0n lines are set to high impedance. The STTn and SPTn bits and the MSTSn, EXCn, COIn, TRCn, ACKDn, and STDn bits of the IICSn register are cleared. The stand-by mode following exit from communications remains in effect until the following communication entry conditions are met. • After a stop condition is detected, restart is in master mode. • An address match occurs or an extension code is received after the start condition. **Caution:** If the IICn operation is enabled (IICEn bit = 1) when the SCL0n line is high level and the SDA0n line is low level, the start condition is detected immediately. To avoid this, after enabling the II2Cn operation, immediately set the LRELn bit to 1 with a bit manipulation instruction. **Note:** **1.** The IICEn bit is automatically cleared to 0 after execution. **2.** The LRELn bit is invalid when the IICEn = 0. **3.** The LRELn bit is 0 when read after the data has been set.

Table 19-2 IICCn register contents (2/4)

Bit position	Bit name	Function
5	WRELn	Wait cancellation control: 0: Wait not cancelled 1: Wait cancelled. **Caution:** When TRCn bit = 1, the WRELn bit is set during the ninth clock and wait is cancelled, after which the TRCn bit is cleared and the SDA0n line is set to high impedance. **Note:** **1.** The WRELn bit is automatically cleared to 0 after wait is cancelled. **2.** The WRELn bit is 0 when read after the data has been set.
4	SPIEn	Enable/disable generation of interrupt request when stop condition is detected. 0: Disabled 1: Enabled **Note:** The SPIEn bit is invalid when the IICEn = 0.
3	WTIMn	Control of wait and interrupt request generation. 0: Interrupt request is generated at the eighth clock's falling edge. Master mode: After output of eight clocks, clock output is set to low level and wait is set. Slave mode: After input of eight clocks, the clock is set to low level and wait is set for the master device. 1: Interrupt request is generated at the ninth clock's falling edge. Master mode: After output of nine clocks, clock output is set to low level and wait is set. Slave mode: After input of nine clocks, the clock is set to low level and wait is set for the master device. In order to generate the ninth clock on SCL0n, the wait status must be cancelled by writing to IICn or setting IICCn.WRELn = 1. Consequently the ninth clock will be delayed until the wait status is cancelled **Note:** During address transfer, an interrupt occurs at the falling edge of the ninth clock regardless of this bit setting. This bit setting becomes valid when the address transfer is completed. In master mode, a wait is inserted at the falling edge of the ninth clock during address transfer. For a slave device that has received a local address, a wait is inserted at the falling edge of the ninth clock after an $\overline{\text{ACK}}$ signal is issued. When the slave device has received an extension code, however, a wait is inserted at the falling edge of the eighth clock.
2	ACKEn	Acknowledgement control: 0: Acknowledgment disabled. 1: Acknowledgment enabled. During the ninth clock period, the SDA0n line is set to low level. **Note:** **1.** The ACKEn bit setting is invalid for address reception. In this case, ACK is generated when the addresses match. However, the ACKEn bit setting is valid for reception of the extension code. **2.** The ACKEn bit is invalid when the IICEn = 0.

Table 19-2 IICCn register contents (3/4)

Bit position	Bit name	Function
1	STTn	Start condition trigger: 0: Single transfer mode 1: When bus is released (in STOP mode): A start condition is generated (for starting as master). The SDA0n line is changed from high level to low level and then the start condition is generated. Next, after the rated amount of time has elapsed, the SCL0n line is changed to low level. During communication with a third party: If the communication reservation function is enabled (IICFn.IICRSVn = 0) • This trigger functions as a start condition reserve flag. When set, it releases the bus and then automatically generates a start condition. • If the communication reservation function is disabled (IICRSVn = 1) • The IICFn.STCFn bit is set. This trigger does not generate a start condition. In the wait state (when master device): A restart condition is generated after the wait is released. **Caution:** 1. For master reception: The STTn bit cannot be set during transfer. It can be set only when the ACKEn bit has been set to 0 and the slave has been notified of final reception. 2. For master transmission: A start condition cannot be generated normally during the $\overline{\text{ACK}}$ period. Set the STTn bit during the wait period after the ninth clock output. 3. For slave: Even when the communication reservation function is disabled (IICRSVn bit = 1), the communication reservation status is entered. Setting the STTn bit to 1 at the same time as the SPTn bit is prohibited. When the STTn bit is set to 1, setting of the STTn bit to 1 again is disabled until the bit is cleared to 0. **Note:** **1.** The STTn bit is cleared (STTn = 0) under the following conditions: • when the STT0 bit is set to 1 in the communication reservation disabled status Cleared by loss in arbitration • cleared after start condition is generated by master device • when LRELn = 1 (communication save) • when IICEn= 0 (operation stop) • after reset **2.** The STTn bit is 0 if it is read immediately after data setting.

Table 19-2 IICCn register contents (4/4)

Bit position	Bit name	Function
0	SPTn	Stop condition trigger: 0: Stop condition is not generated. 1: Stop condition is generated (termination of master device’s transfer). After the SDA0n line goes to low level, either set the SCL0n line to high level or wait until it goes to high level. Next, after the rated amount of time has elapsed, the SDA0n line is changed from low level to high level and a stop condition is generated. Cautions concerning set timing **Caution:** 1. For master reception: The SPTn bit cannot be set during transfer. It can be set only when the ACKEn bit has been set to 0 and during the wait period after the slave has been notified of final reception. 2. For master transmission: A stop condition cannot be generated normally during the $\overline{\text{ACK}}$ period. Set the SPTn to 1 during the wait period. **Note:** **1.** SPTn cannot be set at the same time as the STTn bit. **2.** The SPTn bit can be set only in master mode. However, when the IICRSVn bit is 0, the SPTn bit must be set and a stop condition generated before the first stop condition is detected following the switch to the operation enabled status. For details, see *“Cautions” on page 660*. **3.** When the WTIMn bit has been set to 0 and the SPTn bit is set during the wait period that follows output of eight clocks, note that a stop condition will be generated during the high-level period of the ninth clock. When the ninth clock must be output to apply the ACK on the bus by the receiving device, proceed as follows: • Change IICCn.WTIMn from 0 to 1 in order to receive an additional interrupt after the ninth clock. • Cancel the wait state by IICCn.WRELn = 1 or by writing to the IICn register. • Upon the interrupt after the ninth clock require to set the stop condition by IICCn.STPn = 1. By this the wait status will be cancelled and the stop condition will be generated on the bus. **4.** When the SPTn bit is set to 1, setting the SPTn bit to 1 again is disabled until the bit is cleared to 0. **5.** The SPTn bit is 0 when read after the data has been set.

(2) IICSn - IICn status registers

The IICSn register indicates the status of the I^{2}Cn bus.

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

Address FFFFD86$_H$

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

	7	6	5	4	3	2	1	0
IICSn	MSTSn	ALDn	EXCn	COIn	TRCn	ACKDn	STDn	SPDn
	R	R	R	R	R	R	R	R

Caution The IICSn register can only be read when the IICCn.STTn bit is 1 or during the wait period.

Accessing the IICSn register is prohibited in the following statuses:

- When the CPU operates with the subclock and the main clock oscillation is stopped.
- When the CPU operates with the low speed internal oscillation clock.

Table 19-3 IICSn register contents (1/3)

Bit position	Bit name	Function
7	MSTSn	Master device status: 0: Slave device status or communication stand-by status. This status is entered • when a start condition is generated. 1: Master device communication status. This status is entered • when a stop condition is detected. • when ALDn bit = 1 (arbitration loss). • when LRELn bit is set to 1 (communication save). • when the IICEn bit changes from 1 to 0 (operation stop). • after reset.
6	ALDn	Arbitration loss detection: 0: This status means either that there was no arbitration or that the arbitration result was a “win”. 1: This status indicates the arbitration result was a “loss”. The MSTSn bit is cleared. **Caution:** Any bit manipulation instruction targeting this register also clears the ALDn bit **Note:** The ALDn bit is cleared • automatically when the IICSn register is read. • when the IICEn bit changes from 1 to 0 (operation stop). • after reset.

Table 19-3 IICSn register contents (2/3)

Bit position	Bit name	Function
5	EXCn	Detection of extension code reception: 0: Extension code was not received. 1: Extension code was received: when the higher four bits of the received address data are either 0000_B or 1111_B (set at the rising edge of the eighth clock). **Note:** The EXCn bit is cleared • when a start condition is detected. • when a stop condition is detected. • when LRELn bit is set to 1 (communication save). • when the IICEn bit changes from 1 to 0 (operation stop). • after reset.
4	COIn	Matching address detection: 0: Received address does not match. 1: Received address matches the local address (SVAn register) (set at the rising edge of the eighth clock). **Note:** The COIn bit is cleared • when a start condition is detected. • when a stop condition is detected. • when LRELn bit is set to 1 (communication save). • when the IICEn bit changes from 1 to 0 (operation stop). • after reset.
3	TRCn	Transmit/receive status detection: 0: Receive status (other than transmit status). The SDA0n line is set to high impedance. 1: Transmit status. The value in the SO latch is enabled for output to the SDA0n line (valid starting at the falling edge of the first byte’s ninth clock). **Note:** **1.** The TRCn bit is cleared • when a start condition is detected (as slave). • when a stop condition is detected. • when “1” is output to the first byte’s LSB (transfer direction specification bit) (as master). • when LRELn bit is set to 1 (communication save). • when the IICEn bit changes from 1 to 0 (operation stop). • when WRELn = 1 is set (wait cancellation). • when the ALDn bit changes from 0 to 1 (arbitration loss). • after reset. **2.** The TRCn bit is set as master • when a start condition is generated • when "0" is input by the first byte’s LSB (transfer direction specification bit) **3.** The TRCn bit is set as slave • when “1” is input by the first byte’s LSB (transfer direction specification bit)

Table 19-3 IICSn register contents (3/3)

Bit position	Bit name	Function
2	ACKDn	ACK detection: 0: ACK was not detected. 1: ACK was detected: after the SDA0n line is set to low level at the rising edge of the SCL0n pin's ninth clock **Note:** The ACKDn bit is cleared • when a stop condition is detected. • at the rising edge of the next byte's first clock. • when LRELn bit is set to 1 (communication save). • when the IICEn bit changes from 1 to 0 (operation stop). • after reset.
1	STDn	Start condition detection: 0: Start condition was not detected. 1: Start condition was detected. This indicates that the address transfer period is in effect. **Note:** The STDn bit is cleared • when a stop condition is detected. • at the rising edge of the next byte's first clock. • when LRELn bit is set to 1 (communication save). • when the IICEn bit changes from 1 to 0 (operation stop). • after reset.
0	SPDn	Stop condition detection: 0: Stop condition was not detected. 1: Stop condition was detected. The master device's communication is terminated and the bus is released. **Note:** The SPDn bit is cleared • at the rising edge of the next byte's first clock. • when the IICEn bit changes from 1 to 0 (operation stop). • after reset.

(3) IICFn - IICn flag registers

The IICFn register sets the I^2Cn operation mode and indicates the I^2C bus status.

Access This register can be read/written in 8-bit or 1-bit units. However, the STCFn and IICBSYn bits are read-only.

Address $FFFFD8A_H$

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

IICFn	7	6	5	4	3	2	1	0
	STCFn	IICBSYn	0	0	0	0	STCENn	IICRSVn
	R	R	R	R	R	R	R/W	R/W

Table 19-4 IICFn register contents (1/2)

Bit position	Bit name	Function
7	STCFn	Clear flag STTn: 0: Start condition issued. 1: Start condition cannot be issued: when start condition is not issued and STTn flag is cleared while communication reservation is disabled (IICRSVn = 1). **Note:** The STCFn bit is cleared • by IICCn.STTn = 1. • when the IICEn bit changes from 1 to 0 (operation stop). • after reset.
6	IICBSYn	I^2Cn bus status: 0: Bus released status. 1: Bus communication status: by setting the IICCn.IICEn bit when STCENn = 0, or when start condition is detected. **Note: 1.** The IICBSYn bit is cleared • when a stop condition is detected. • when the IICEn bit changes from 1 to 0 (operation stop). • after reset. **2.** The initial value of the IICBSYn bit is set by using the STCENn bit (see *"Cautions" on page 660*).
1	STCENn	Initial start enable trigger: 0: Start conditions cannot be generated until a stop condition is detected following operation enable (IICEn bit = 1). 1: Start conditions can be generated even if a stop condition is not detected following operation enable (IICEn = 1). **Caution: 1.** Write the STCENn bit only when operation is stopped (IICEn = 0). **2.** When the STCENn = 1, the bus released status (IICBSYn = 0) is recognized regardless of the actual bus status immediately after the I^2Cn bus operation is enabled. Therefore, to issue the first start condition (STTn = 1), it is necessary to confirm that the bus has been released, so as to not disturb other communications. **Note:** The STCENn bit is cleared • when a start condition is detected. • after reset.

Table 19-4 IICFn register contents (2/2)

Bit position	Bit name	Function
0	IICRSVn	Communication reservation function disable bit: 0: Communication reservation enabled. 1: Communication reservation disabled. **Caution:** Write the IICRSVn bit only when operation is stopped (IICEn = 0).

(4) IICCLn - IICn clock select registers

The IICCLn register sets the transfer clock for the I^2Cn bus.

Access This register can be read/written in 8-bit or 1-bit units. However, the CLDn and DADn bits are read-only.

Address $FFFFD84_H$

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

	7	6	5	4	3	2	1	0
IICCLn	0	0	CLDn	DADn	SMCn	DFCn	CLn1	CLn0
	R	R	R	R	R/W	R/W	R/W	R/W

Table 19-5 IICCLn register contents

Bit position	Bit name	Function
5	CLDn	Detection of SCL0n pin level (valid only when IICCn.IICEn = 1): 0: The SCL0n pin was detected at low level. 1: The SCL0n pin was detected at high level. **Note:** The CLDn bit is cleared • when the SCL0n pin is at low level. • when the IICEn bit changes from 1 to 0 (operation stop). • after reset.
4	DADn	Detection of SDA0n pin level (valid only when IICCn.IICEn = 1): 0: The SDA0n pin was detected at low level. 1: The SDA0n pin was detected at high level. **Note:** The CLDn bit is cleared • when the SDA0n pin is at low level. • when the IICEn bit changes from 1 to 0 (operation stop). • after reset.
3	SMCn	Specification of operation mode: 0: Operation in standard mode. 1: Operation in fast-speed mode.
2	DFCn	Digital filter operation control: 0: Digital filter off. 1: Digital filter on. The digital filter is used to eliminate noise in fast-speed mode. **Note:** The digital filter can be used only in fast-speed mode.In fast-speed mode, the transfer clock does not vary regardless of the DFCn bit setting (on/off).
1, 0	CLn[1:0]	The CLn[1:0] bits are set in combination with the SMC bit, IICXn.CLXn bit and the OCKSn register (see *"Transfer rate setting" on page 625*).

(5) IICXn - IICn function expansion registers

The IICXn register provides additional transfer data rate configuration in fast-speed mode.

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

Address FFFFD85_H

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

	7	6	5	4	3	2	1	0
IICXn	0	0	0	0	0	0	0	CLXn
	R	R	R	R	R	R	R	R/W

Table 19-6 IICXn register contents

Bit position	Bit name	Function
0	CLXn	The CLXn bit is set in combination with the IICCLn.SMCn bit, IICCLn.CLn[1:0] bits and the OCKSn register (see *"Transfer rate setting" on page 625*).

(6) OCKSn - IICn division clock select registers

The OCKSn register controls the IICn division clock.

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

Address FFFF340_H

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

	7	6	5	4	3	2	1	0
OCKSn	0	0	0	OCKSENn	OCKSTHn	0	OCKSn1	OCKSn0
	R	R	R	R/W	R/W	R	R/W	R/W

Table 19-7 OCKSn register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>4</td><td>OCKSENn</td><td>Operation setting of IICn division clock:
0: Disable IICn division clock operation.
1: Enable IICn division clock operation.</td></tr>
<tr><td>3, 1, 0</td><td>OCKSTHn,
OCKSn[1:0]</td><td>Selection of IICn division clock IICLKPS:
<table>
<tr><th rowspan="2">OCKSTHn</th><th rowspan="2">OCKSn1</th><th rowspan="2">OCKSn0</th><th colspan="2">IICn division clock IICLKPS</th></tr>
<tr><th>PRSI = 0</th><th>PRSI = 1</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>$f_{XX}/2$</td><td>$f_{XX}/4$</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>$f_{XX}/3$</td><td>$f_{XX}/6$</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>$f_{XX}/4$</td><td>$f_{XX}/8$</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>$f_{XX}/5$</td><td>$f_{XX}/10$</td></tr>
<tr><td>1</td><td>×</td><td>×</td><td>f_{XX}</td><td>$f_{XX}/2$</td></tr>
</table>
<b>Note:</b> PRSI can be set by the option bytes (refer to <i>"Flash Mask Options" on page 330</i> for details):
• PRSI = 0: $f_{XX} \leq 32$ MHz
• PRSI = 1: 32 MHz < $f_{XX} \leq 48$ MHz</td></tr>
</table>

(7) Transfer rate setting

The nominal transfer rate of the I^2C interface is determined by the root clock source f_{XP1}. The frequency of f_{XP1} can be set to f_{XP1}or $f_{XP1}/2$ by the PRSI bit of the option byte (007BH).

- The f_{XP1} can be divided by 1 to 5, configured by OCKSn.OCKSTHn and OCKSn.OCKSTn[1:0] (refer to *"OCKSn - IICn division clock select registers" on page 624*). The output clock of this divider is named IICLKPS.
- IICLKPS is passed through another configurable divider that finally outputs the clock for the serial transfer IICLKTC. This divider is configured by IICCLn.CL[1:0] and IICXn.CLX0 according to the following tables:

Note The I^2C interface input clock IICLKPS must be in the range of 1 MHz to 10 MHz.

The following tables summarize the transfer rate settings:

PRSI	IICCLn.SMCn	Mode	Table
0	0	standard	*Table 19-8 on page 625*
0	1	fast-speed	*Table 19-9 on page 626*
1	0	standard	*Table 19-10 on page 626*
1	1	fast-speed	*Table 19-11 on page 627*

Note PRSI can be set by the option bytes (refer to *"Flash Mask Options" on page 330* for details):

- PRSI = 0: $f_{XX} \leqq 32$ MHz
- PRSI = 1: 32 MHz < $f_{XX} \leqq 48$ MHz

Table 19-8 PRSI = 0: Transfer rate settings in standard mode (IICCLn.SMCn = 0)

IICXn. CLXn	IICCLn. CLn1	IICCLn. CLn0	OCKSn	IICLKPS	Transfer Clock	Possible Main System Clock Range (fxx)		(Reference) Transfer speed
						from	to	
0	0	0	10_H	fxx/2	fxx/88	4 MHz	8.38 MHz	45.5kHz ~ 95.2kHz
			11_H	fxx/3	fxx/132	6 MHz	12.57 MHz	45.5kHz ~ 95.2kHz
			12_H	fxx/4	fxx/176	8 MHz	16.76 MHz	45.5kHz ~ 95.2kHz
			13_H	fxx/5	fxx/220	10 MHz	20.95 MHz	45.5kHz ~ 95.2kHz
			18_H	fxx	fxx/44	4 MHz	4.19 MHz	90.9kHz ~ 95.2kHz
0	0	1	10_H	fxx/2	fxx/172	8.38 MHz	16.76 MHz	48.7kHz ~ 97.4kHz
			11_H	fxx/3	fxx/258	12.57 MHz	25.14 MHz	48.7kHz ~ 97.4kHz
			12_H	fxx/4	fxx/344	16.76 MHz	32 MHz	48.7kHz ~ 93.0kHz
			13_H	fxx/5	fxx/430	20.95 MHz	32 MHz	48.7kHz ~ 74.4kHz
0	1	0	×	fxx	fxx/86	4.19 MHz	8.38 MHz	48.7kHz ~ 97.4kHz
0	1	1	10_H	fxx/2	fxx/132	12.80 MHz		97.0kHz
			11_H	fxx/3	fxx/198	19.20 MHz		97.0kHz
			12_H	fxx/4	fxx/264	25.60 MHz		97.0kHz
			13_H	fxx/5	fxx/330	32 MHz		97.0kHz
			18_H	fxx	fxx/66	6.40 MHz		97.0kHz
Other than above				Setting Prohibited				

Table 19-9 PRSI = 0: Transfer rate settings in fast-speed mode (IICCLn.SMCn = 1)

IICXn. CLXn	IICCLn. CLn1	IICCLn. CLn0	OCKSn	Selected Clock	Transfer Clock	Possible Main System Clock Range (fxx)		(Reference) Transfer speed
						from	to	
0	0	×	10_H	fxx/2	fxx/48	8 MHz	16.76 MHz	166.7kHz ~ 349.2kHz
			11_H	fxx/3	fxx/72	12 MHz	25.14 MHz	166.7kHz ~ 349.2kHz
			12_H	fxx/4	fxx/96	16 MHz	32 MHz	166.7kHz ~ 333.3kHz
			13_H	fxx/5	fxx/120	20 MHz	32 MHz	166.7kHz ~ 266.7kHz
0	1	0	×	fxx	fxx/24	4 MHz	8.38 MHz	166.7kHz ~ 349.2kHz
0	1	1	10_H	fxx/2	fxx/36	12.80 MHz		355.6kHz
			11_H	fxx/3	fxx/54	19.20 MHz		355.6kHz
			12_H	fxx/4	fxx/72	25.60 MHz		355.6kHz
			13_H	fxx/5	fxx/90	32 MHz		355.6kHz
			18_H	fxx	fxx/18	6.4 MHz		355.6kHz
1	0	×	10_H	fxx/2	fxx/24	8 MHz	8.38 MHz	333.3kHz ~ 349.2kHz
			11_H	fxx/3	fxx/36	12 MHz	12.57 MHz	333.3kHz ~ 349.2kHz
			12_H	fxx/4	fxx/48	16 MHz	16.67 MHz	333.3kHz ~ 349.2kHz
			13_H	fxx/5	fxx/60	20 MHz	20.95 MHz	333.3kHz ~ 349.2kHz
			18_H	fxx	fxx/12	4 MHz	4.19 MHz	333.3kHz ~ 349.2kHz
1	1	0	×	fxx	fxx/12	4 MHz	4.19 MHz	333.3kHz ~ 349.2kHz
Other than above				Setting Prohibited				

Table 19-10 PRSI = 1: Transfer rate settings in standard mode (IICCLn.SMCn = 0)

IICXn. CLXn	IICCLn. CLn1	IICCLn. CLn0	OCKSn	Selected Clock	Transfer Clock	Possible Main System Clock Range (fxx)		(Reference) Transfer speed
						from	to	
0	0	0	10_H	fxx/4	fxx/176	8 MHz	16.76 MHz	45.5kHz ~ 95.2kHz
			11_H	fxx/6	fxx/264	12 MHz	25.14 MHz	45.5kHz ~ 95.2kHz
			12_H	fxx/8	fxx/352	16 MHz	33.52 MHz	45.5kHz ~ 95.2kHz
			13_H	fxx/10	fxx/440	20 MHz	41.90 MHz	45.5kHz ~ 95.2kHz
			18_H	fxx/2	fxx/88	4 MHz	8.38 MHz	90.9kHz ~ 95.2kHz
0	0	1	10_H	fxx/4	fxx/344	16.76 MHz	33.52 MHz	48.7kHz ~ 97.4kHz
			11_H	fxx/6	fxx/516	25.14 MHz	48 MHz	48.7kHz ~ 93.0kHz
			12_H	fxx/8	fxx/688	33.52 MHz	48 MHz	48.7kHz ~ 69.8kHz
			13_H	fxx/10	fxx/860	41.90 MHz	48 MHz	48.7kHz ~ 55.8kHz
			18_H	fxx/2	fxx/172	8.38 MHz	16.76 MHz	48.7kHz ~ 97.4kHz
0	1	0	×	fxx/2	fxx/172	8.38 MHz	16.76 MHz	48.7kHz ~ 97.4kHz
0	1	1	10_H	fxx/4	fxx/264	25.60 MHz		97.0kHz
			11_H	fxx/6	fxx/396	38.40 MHz		97.0kHz
			18_H	fxx/2	fxx/132	12.80 MHz		97.0kHz
Other than above				Setting Prohibited				

Table 19-11 PRSI = 1: Transfer rate settings in fast-speed mode (IICCLn.SMCn = 1)

IICXn. CLXn	IICCLn. CLn1	IICCLn. CLn0	OCKSn	Selected Clock	Transfer Clock	Possible Main System Clock Range (fxx)		(Reference) Transfer speed
						from	to	
0	0	×	10_H	fxx/4	fxx/96	16 MHz	33.52 MHz	166.7kHz ~ 349.2kHz
			11_H	fxx/6	fxx/144	24 MHz	48 MHz	166.7kHz ~ 333.3kHz
			12_H	fxx/8	fxx/192	32 MHz	48 MHz	166.7kHz ~ 250.0kHz
			13_H	fxx/10	fxx/240	40 MHz	48 MHz	166.7kHz ~ 200.0kHz
0	1	0	×	fxx/2	fxx/48	8 MHz	8.38 MHz	166.7kHz ~ 349.2kHz
0	1	1	10_H	fxx/4	fxx/72	25.60 MHz		355.6kHz
			11_H	fxx/6	fxx/108	38.40 MHz		355.6kHz
			18_H	fxx/2	fxx/36	12.80 MHz		355.6kHz
1	0	×	10_H	fxx/4	fxx/48	16 MHz	16.76 MHz	333.3kHz ~ 349.2kHz
			11_H	fxx/6	fxx/72	24 MHz	25.14 MHz	333.3kHz ~ 349.2kHz
			12_H	fxx/8	fxx/96	32 MHz	33.52 MHz	333.3kHz ~ 349.2kHz
			13_H	fxx/10	fxx/120	40 MHz	41.90 MHz	333.3kHz ~ 349.2kHz
1	1	0	×	fxx/2	fxx/24	8 MHz	8.38 MHz	333.3kHz ~ 349.2kHz
Other than above				Setting Prohibited				

Clock Stretching Heavy capacitive load and the dimension of the external pull-up resistor on the I^2C bus pins may yield extended rise times of the rising edge of SCL0n and SDA0n. Since the controller senses the level of the I^2C bus signals it recognizes such situation and takes countermeasures by stretching the clock SCL0n in order to ensure proper high level time t_{SCLH} of SCL0n.

After the microcontroller releases the (open-drain) SCL0n pin it waits until the SCL0n level exceeds the valid high level threshold V_{thH}. Then it does not pull SCL0n to low level before the nominal high level time t_{SCLH_nom} has elapsed.

This mechanism is the same used, when a slow I^2C slave device is pulling down SCL0n to low level to initiate a wait state.

Note It is assumed that the rise time f_r is much bigger than the fall time f_f.

Figure 19-3 shows an example.

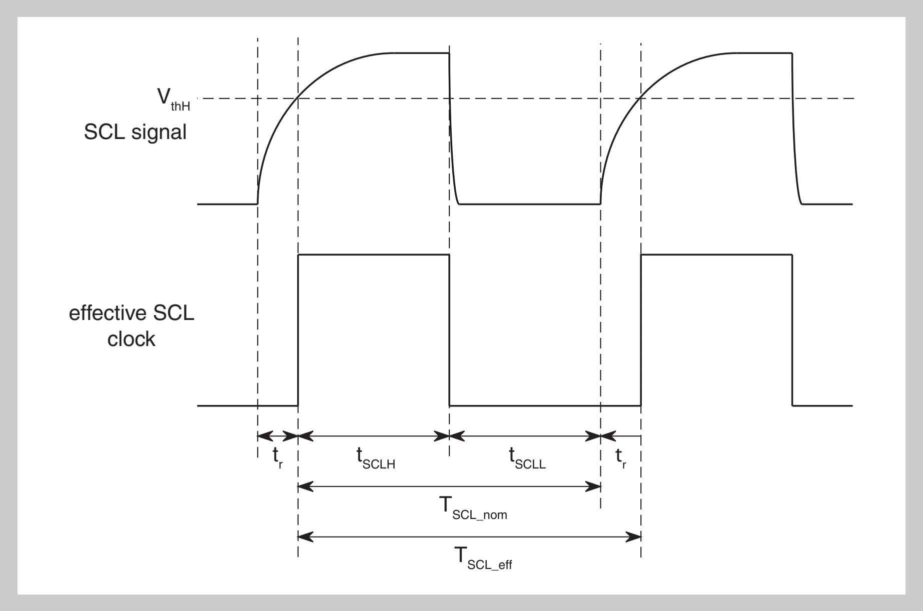


Figure 19-3 Clock Stretching of SCL0n

The effective clock frequency appearing at the SCL0n pin calculates to

$f_{SCL\_eff} = 1 / (T_{SCL\_nom} + t_r)$

With a nominal frequency of f_{SCL_nom} = 355.6 KHz (T_{SCL_nom} = 2.812 µs and a rise time of t_r = 135 ns the effective frequency is f_{eff} = 339.31 KHz.

(8) IICn - IICn shift registers

The IICn registers are used for serial transmission/reception (shift operations) synchronized with the serial clock.

A wait state is released by writing the IICn register during the wait period, and data transfer is started.

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

Address FFFFD80$_H$

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

IICn

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

Caution Access (read/write) this register only during the wait period. Accessing this register in communication states other than the wait period is prohibited. However, for the master device, this register can be written once only after the transmission trigger bit (IICC0.STT0 bit) has been set to 1.

(9) SVAn - IICn slave address registers

The SVAn registers hold the I^{2}C bus's slave addresses.

Reset input sets this register to 00H.

Access This register can be read/written in 8-bit units, but bit 0 should be fixed to 0.

Address FFFFD83$_H$

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

SVAn

7	6	5	4	3	2	1	0
slave address							0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Caution Rewriting of the SVAn register is prohibited when the start condition is detected (IICS0.STD0 = 1).

19.5 I^{2}C Bus Mode Functions

19.5.1 Pin functions

The serial clock pin (SCL0n) and serial data bus pin (SDA0n) are configured as follows.

SCL0n The SCL0n pin is used for serial clock input and output.
It is equipped with an N-ch open-drain output for both master and slave devices. As input it has Schmitt input characteristics.

SDA0n The SDA0n pin is used for serial data input and output.
It is equipped with an N-ch open-drain output for both master and slave devices. As input it has Schmitt input characteristics.

Since outputs from the serial clock line and the serial data bus line are N-ch open-drain outputs, an external pull-up resistor is required.

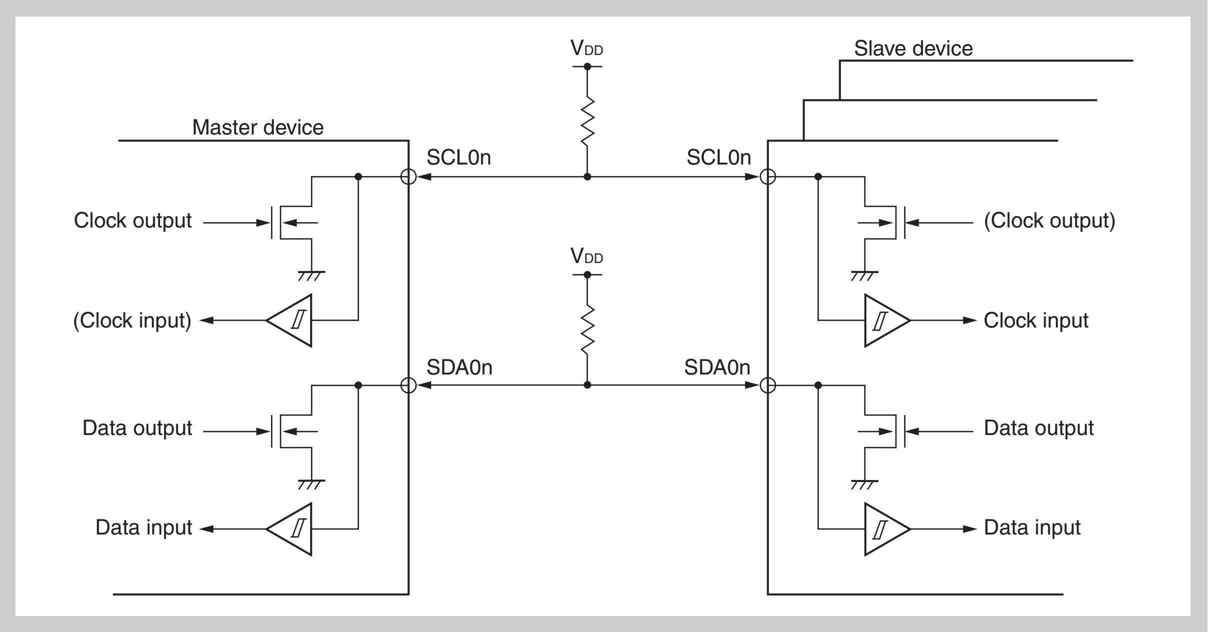


Figure 19-4 Pin configuration diagram

19.6 I²C Bus Definitions and Control Methods

The following section describes the I²C bus's serial data communication format and the signals used by the I²C bus. The transfer timing for the "start condition", "address", "transfer direction specification", "data" and "stop condition" output via the I²C bus's serial data bus is shown below.

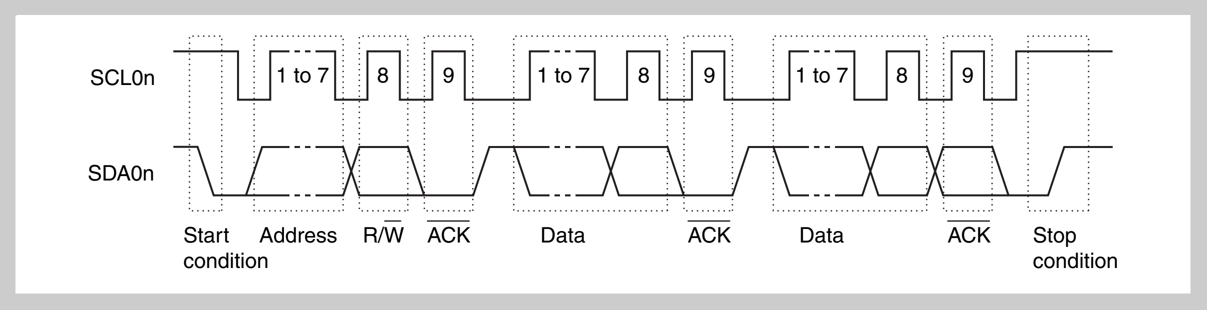


Figure 19-5 I²C bus serial data transfer timing

The master device outputs the start condition, slave address, and stop condition.

The acknowledge signal (ACK) can be output by either the master or slave device (normally, it is output by the device that receives 8-bit data).

The serial clock (SCL0n) is continuously output by the master device. However, in the slave device, the SCL0n pin's low-level period can be extended and a wait can be inserted.

19.6.1 Start condition

A start condition is met when the SCL0n pin is high level and the SDA0n pin changes from high level to low level. The start condition for the SCL0n and SDA0n pins is a signal that the master device outputs to the slave device when starting a serial transfer. The slave device can defect the start condition.

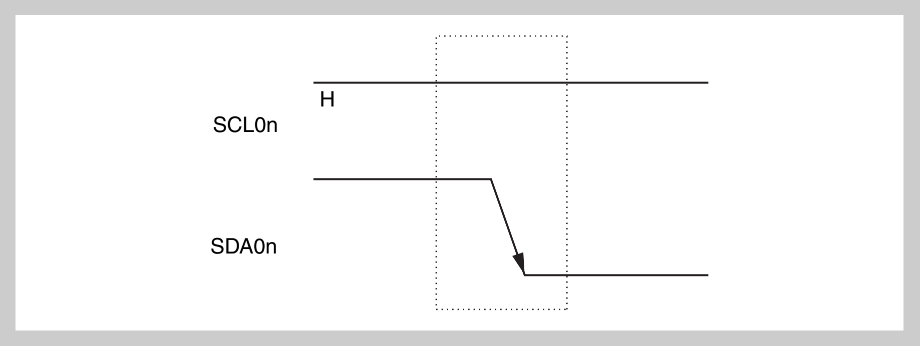


Figure 19-6 Start condition

A start condition is output when the IICCn.STTn bit is set (1) after a stop condition has been detected (IICSn.SPDn bit = 1). When a start condition is detected, the IICSn.STDn bit is set (1).

Caution When the IICC0.IICE0 bit of the microcontroller is set to 1 while communications with other devices are in progress, the start condition may be detected depending on the status of the communication line. Be sure to set the IICC0.IICE0 bit to 1 when the SCL00 and SDA00 lines are high level.

19.6.2 Addresses

The 7 bits of data that follow the start condition are defined as an address.

An address is a 7-bit data segment that is output in order to select one of the slave devices that are connected to the master device via the bus lines. Therefore, each slave device connected via the bus lines must have a unique address.

The slave devices include hardware that detects the start condition and checks whether or not the 7-bit address data matches the data values stored in the SVAn register. If the address data matches the values of the SVAn register, the slave device is selected and communicates with the master device until the master device transmits a start condition or stop condition.

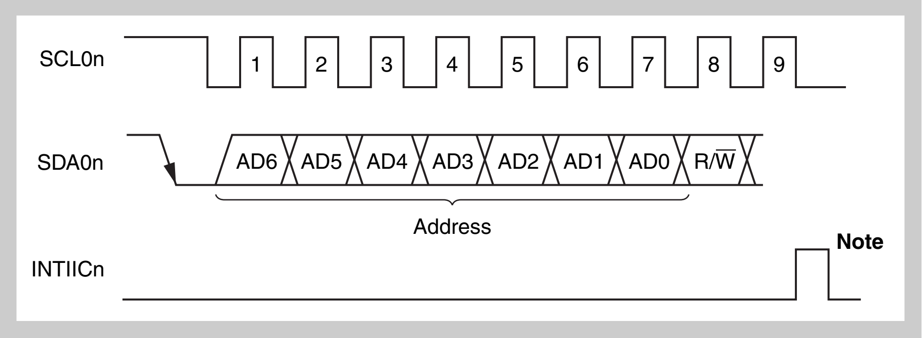


Figure 19-7 Address

Note The interrupt request signal (INTIICn) is generated if a local address or extension code is received during slave device operation.

The slave address and the eighth bit, which specifies the transfer direction as described in *"Transfer direction specification" on page 633*, are written together to IIC shift register n (IICn) and then output. Received addresses are written to the IICn register.

The slave address is assigned to the higher 7 bits of the IICn register.

19.6.3 Transfer direction specification

In addition to the 7-bit address data, the master device sends 1 bit that specifies the transfer direction. When this transfer direction specification bit has a value of 0, it indicates that the master device is transmitting data to a slave device. When the transfer direction specification bit has a value of 1, it indicates that the master device is receiving data from a slave device.

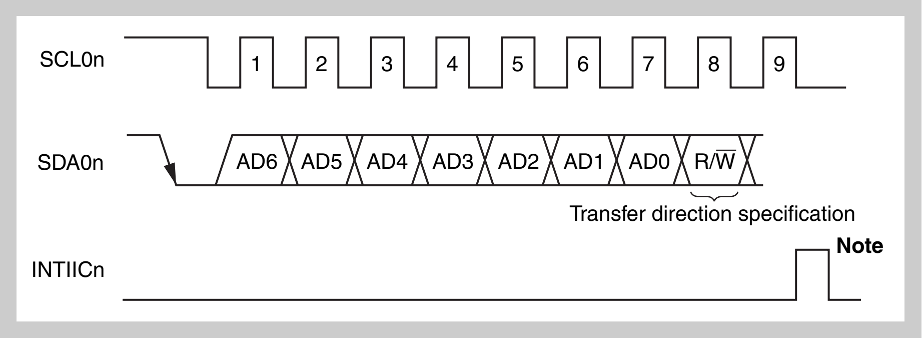


Figure 19-8 Transfer direction specification

Note The INTIICn signal is generated if a local address or extension code is received during slave device operation.

19.6.4 Acknowledge signal ($\overline{ACK}$)

The acknowledge signal ($\overline{ACK}$) is used by the transmitting and receiving devices to confirm serial data reception.

The receiving device returns one $\overline{ACK}$ signal for each 8 bits of data it receives. The transmitting device normally receives an $\overline{ACK}$ signal after transmitting 8 bits of data. The detection of ACK is confirmed with the IICS0.ACKD0 bit. However, when the master device is the receiving device, it does not output an $\overline{ACK}$ signal after receiving the final data to be transmitted. The transmitting device detects whether or not an $\overline{ACK}$ signal is returned after it transmits 8 bits of data. When an $\overline{ACK}$ signal is returned, the reception is judged as normal and processing continues. If the slave device does not return an $\overline{ACK}$ signal, the master device outputs either a stop condition or a restart condition and then stops the current transmission. Failure to return an $\overline{ACK}$ signal may be caused by the following three factors:

- Reception was not performed normally.
- The final data was received.
- The receiving device (slave) does not exist for the specified address.

When the receiving device sets the SDA0n line to low level during the ninth clock, the $\overline{ACK}$ signal becomes active (normal receive response).

When the IICCn.ACKEn bit is set to 1, automatic $\overline{ACK}$ signal generation is enabled.

Transmission of the eighth bit following the 7 address data bits causes the IICSn.TRCn bit to be set. When this TRCn bit's value is 0, it indicates receive mode. Therefore, the ACKEn bit should be set to 1.

When the slave device is receiving (when TRCn bit = 0), if the slave device does not need to receive any more data after receiving several bytes, clearing the ACKEn bit to 0 will prevent the master device from starting transmission of the subsequent data.

Similarly, when the master device is receiving (when TRCn bit = 0) and the subsequent data is not needed and when either a restart condition or a stop condition should therefore be output, clearing the ACKEn bit to 0 will prevent the $\overline{\text{ACK}}$ signal from being returned. This prevents the MSB from being output via the SDA0n line (i.e., stops transmission) during transmission from the slave device.

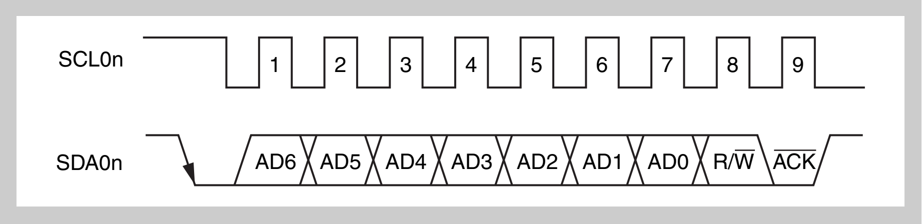


Figure 19-9 $\overline{\text{ACK}}$ signal

When the local address is received, an $\overline{\text{ACK}}$ signal is automatically output in synchronization with the falling edge of the SCL0n pin's eighth clock regardless of the value of the ACKEn bit. No $\overline{\text{ACK}}$ signal is output if the received address is not a local address.

The $\overline{\text{ACK}}$ signal output method during data reception is based on the wait timing setting, as described below.

When 8-clock wait is selected (IICCn.WTIMn bit = 0):

> The $\overline{\text{ACK}}$ signal is output at the falling edge of the SCL0n pin's eighth clock if the ACKEn bit is set to 1 before wait cancellation.

When 9-clock wait is selected (IICCn.WTIMn bit = 1):

> The $\overline{\text{ACK}}$ signal is automatically output at the falling edge of the SCL0n pin's eighth clock if the ACKEn bit has already been set to 1.

19.6.5 Stop condition

When the SCL0n pin is high level, changing the SDA0n pin from low level to high level generates a stop condition.

A stop condition is a signal that the master device outputs to the slave device when serial transfer has been completed. When used as the slave device, the start condition can be detected.

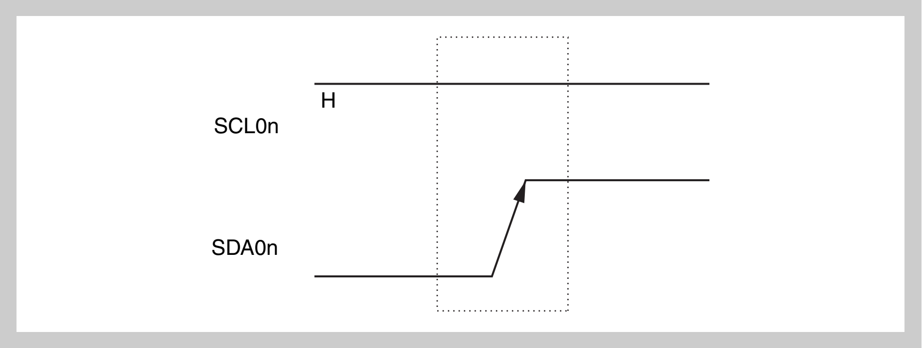


Figure 19-10 Stop condition

A stop condition is generated when the IICCn.SPTn bit is set to 1. When the stop condition is detected, the IICSn.SPDn bit is set to 1 and the INTIICn signal is generated when the IICCn.SPIEn bit is set to 1.

19.6.6 Wait signal ($\overline{\text{WAIT}}$)

The wait signal ($\overline{\text{WAIT}}$) is used to notify the communication partner that a device (master or slave) is preparing to transmit or receive data (i.e., is in a wait state).

Setting the SCL0n pin to low level notifies the communication partner of the wait status. When the wait status has been cancelled for both the master and slave devices, the next data transfer can begin.

(1) When master device has a nine-clock wait and slave device has an eight-clock wait (master: transmission, slave: reception, and IICCn.ACKEn bit = 1)

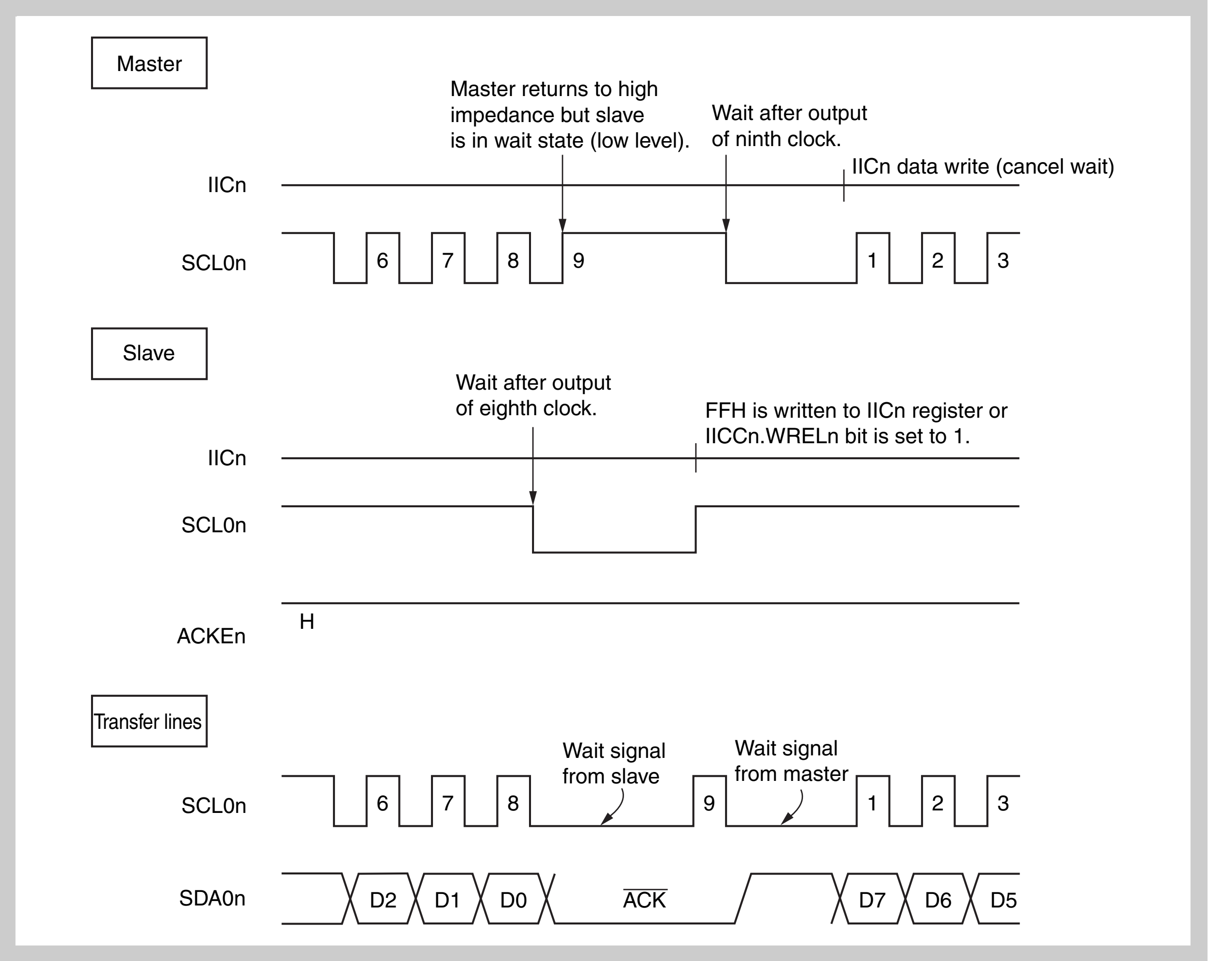

Master
Master returns to high impedance but slave is in wait state (low level).
Wait after output of ninth clock.
IICn data write (cancel wait)
IICn
SCL0n
6
7
8
9
1
2
3
Slave
Wait after output of eighth clock.
FFH is written to IICn register or IICCn.WRELn bit is set to 1.
IICn
SCL0n
ACKEn
H
Transfer lines
Wait signal from slave
Wait signal from master
SCL0n
6
7
8
9
1
2
3
SDA0n
D2
D1
D0
$\overline{\text{ACK}}$
D7
D6
D5

Figure 19-11 Wait signal (1/2)

(2) When master and slave devices both have a nine-clock wait (master: transmission, slave: reception, and ACKEn bit = 1)

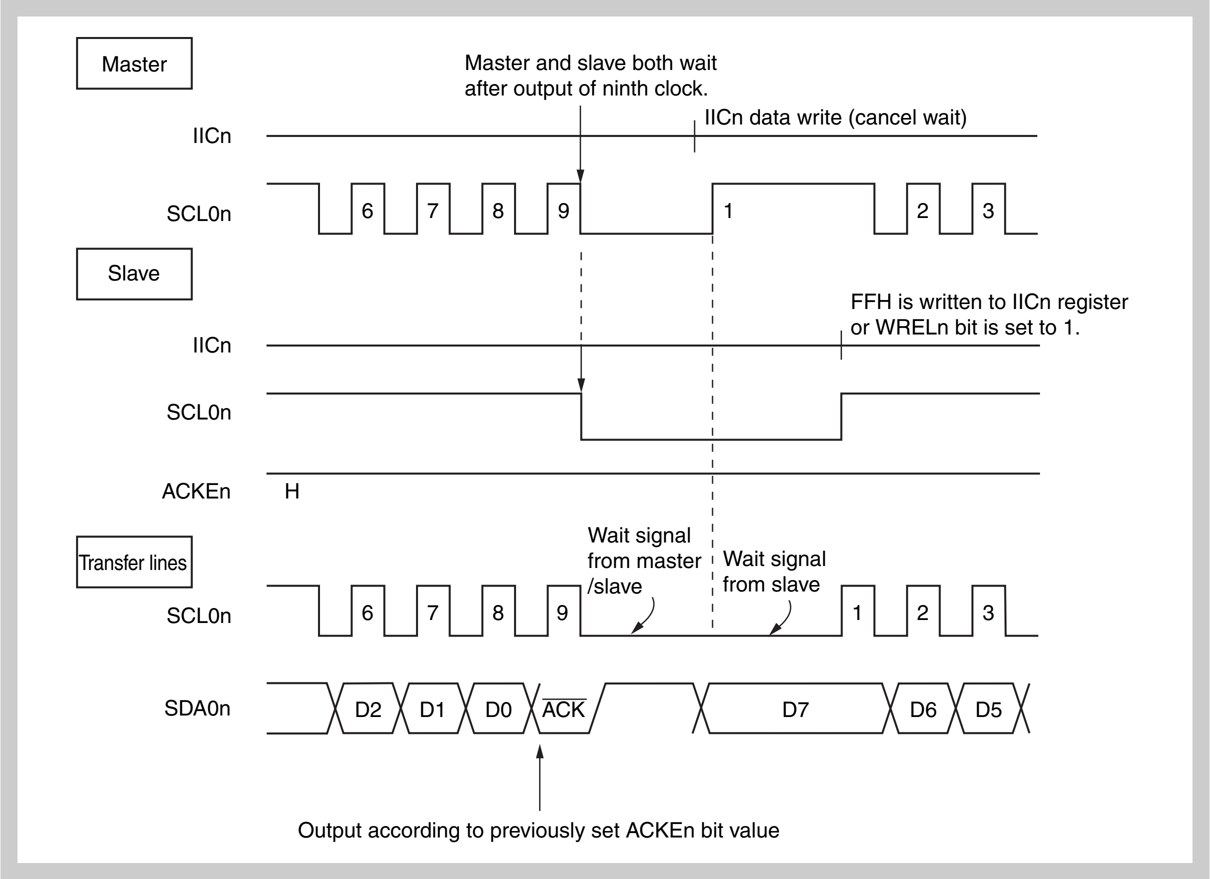

Master
Master and slave both wait
after output of ninth clock.
IICn
IICn data write (cancel wait)
SCL0n
6
7
8
9
1
2
3
Slave
FFH is written to IICn register
or WRELn bit is set to 1.
IICn
SCL0n
ACKEn
H
Transfer lines
Wait signal
from master
/slave
Wait signal
from slave
SCL0n
6
7
8
9
1
2
3
SDA0n
D2
D1
D0
ACK
D7
D6
D5
Output according to previously set ACKEn bit value

Figure 19-12 Wait signal (2/2)

A wait may be automatically generated depending on the setting of the IICCn.WTIMn bit.

Normally, when the IICCn.WRELn bit is set to 1 or when FFH is written to the IICn register on the receiving side, the wait status is cancelled and the transmitting side writes data to the IICn register to cancel the wait status.

The master device can also cancel the wait status via either of the following methods.

- By setting the IICCn.STTn bit to 1
- By setting the IICCn.SPTn bit to 1

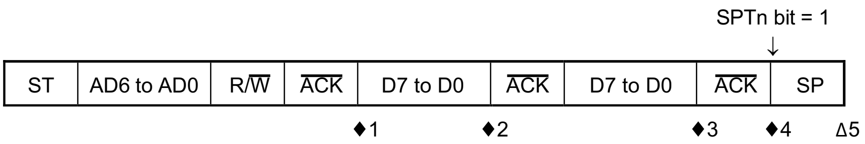
19.7 I^2C Interrupt Request Signals (INTIICn)

The following shows the value of the IICSn register at the INTIICn interrupt request signal generation timing and at the INTIICn signal timing.

19.7.1 Master device operation

(1) Start ~ Address ~ Data ~ Data ~ Stop (normal transmission/reception)

<1> When WTIMn bit = 0



♦ 1: IICSn register = 1000X110B

♦ 2: IICSn register = 1000X000B

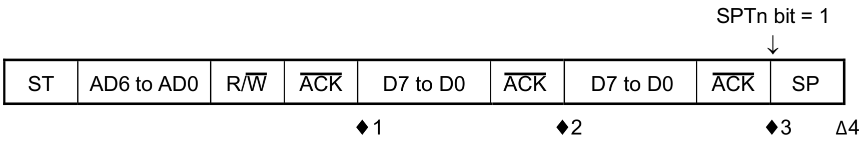
♦ 3: IICSn register = 1000X000B (WTIMn bit = 1)

♦ 4: IICSn register = 1000XX00B

Δ 5: IICSn register = 00000001B

Remarks 1. ♦ : Always generated
Δ: Generated only when SPIEn bit = 1
X: don't care

<2> When WTIMn bit = 1



♦ 1: IICSn register = 1000X110B

♦ 2: IICSn register = 1000X100B

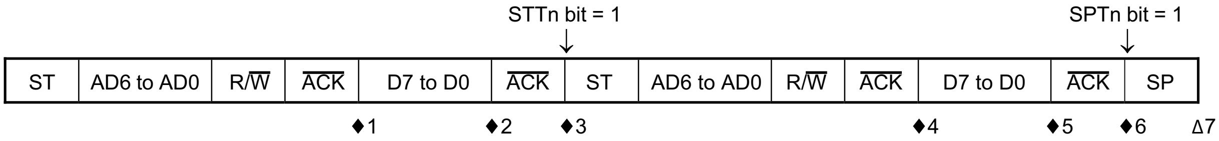
♦ 3: IICSn register = 1000XX00B

Δ 4: IICSn register = 00000001B

Remarks 1. ♦ : Always generated
Δ: Generated only when SPIEn bit = 1
X: don't care

(2) Start ~ Address ~ Data ~ Start ~ Address ~ Data ~ Stop (restart)

<1> When WTIMn bit = 0



♦1: IICSn register = 1000X110B

♦2: IICSn register = 1000X000B (WTIMn bit = 1)

♦3: IICSn register = 1000XX00B (WTIMn bit = 0)

♦4: IICSn register = 1000X110B (WTIMn bit = 0)

♦5: IICSn register = 1000X000B (WTIMn bit = 1)

♦6: IICSn register = 1000XX00B

Δ 7: IICSn register = 00000001B

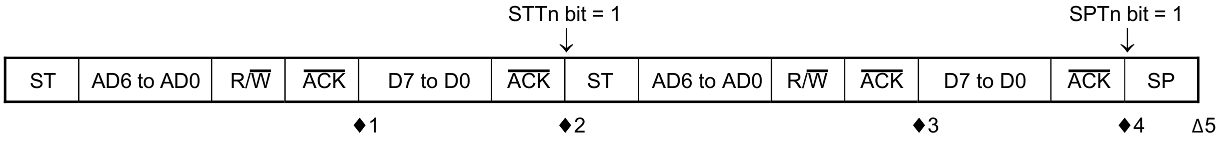
Remarks 1. ♦ : Always generated

Δ: Generated only when SPIEn bit = 1

X: don’t care

2. n = 0

<2> When WTIMn bit = 1



♦ 1: IICSn register = 1000X110B

♦ 2: IICSn register = 1000XX00B

♦ 3: IICSn register = 1000X110B

♦ 4: IICSn register = 1000XX00B

Δ 5: IICSn register = 00000001B

Remarks 1. ♦ : Always generated

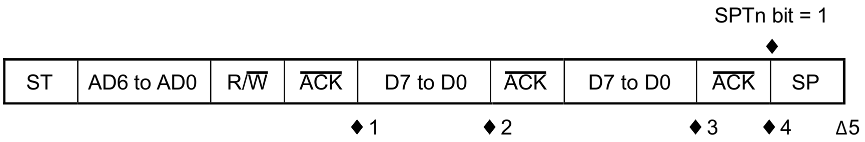
Δ: Generated only when SPIEn bit = 1

X: don’t care

2. n = 0

(3) Start ~ Code ~ Data ~ Data ~ Stop (extension code transmission)

<1> When WTIMn bit = 0



♦ 1: IICSn register = 1010X110B

♦ 2: IICSn register = 1010X000B

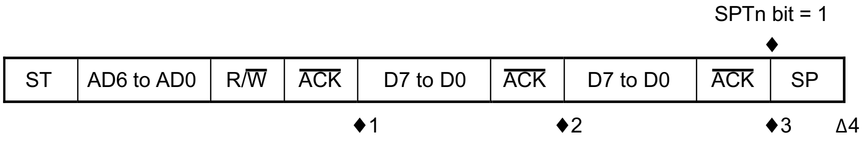
♦ 3: IICSn register = 1010X000B (WTIMn bit = 1)

♦ 4: IICSn register = 1010XX00B

Δ 5: IICSn register = 00000001B

Remarks 1. ♦ : Always generated
Δ: Generated only when SPIEn bit = 1
X: don't care
2. n = 0

<2> When WTIMn bit = 1



♦1: IICSn register = 1010X110B

♦2: IICSn register = 1010X100B

♦3: IICSn register = 1010XX00B

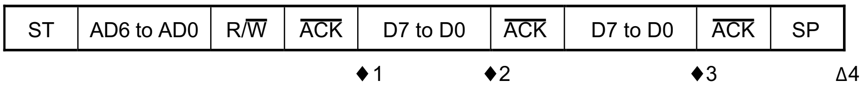
Δ 4: IICSn register = 00000001B

Remarks 1. ♦: Always generated
Δ: Generated only when SPIEn bit = 1
X: don't care
2. n = 0

19.7.2 Slave device operation

(1) Start ~ Address ~ Data ~ Data ~ Stop

<1> When WTIMn bit = 0


ST
AD6 to AD0
R/$\overline{\mathrm{W}}$
$\overline{\mathrm{ACK}}$
D7 to D0
$\overline{\mathrm{ACK}}$
D7 to D0
$\overline{\mathrm{ACK}}$
SP
♦1
♦2
♦3
Δ4

♦ 1: IICSn register = 0001X110B

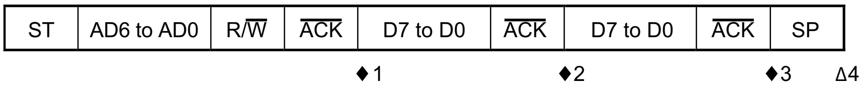
♦ 2: IICSn register = 0001X000B

♦ 3: IICSn register = 0001X000B

Δ 4: IICSn register = 00000001B

Remarks 1. ♦ : Always generated
Δ: Generated only when SPIEn bit = 1
X: don't care
2. n = 0

<2> When WTIMn bit = 1


ST
AD6 to AD0
R/$\overline{\mathrm{W}}$
$\overline{\mathrm{ACK}}$
D7 to D0
$\overline{\mathrm{ACK}}$
D7 to D0
$\overline{\mathrm{ACK}}$
SP
♦1
♦2
♦3
Δ4

♦ 1: IICSn register = 0001X110B

♦ 2: IICSn register = 0001X100B

♦ 3: IICSn register = 0001XX00B

Δ 4: IICSn register = 00000001B

Remarks 1. ♦ : Always generated
Δ: Generated only when SPIEn bit = 1
X: don't care
2. n = 0

(2) Start ~ Address ~ Data ~ Start ~ Address ~ Data ~ Stop

<1> When WTIMn bit = 0 (after restart, address match)

ST	AD6 to AD0	R/$\overline{W}$	$\overline{ACK}$	D7 to D0	$\overline{ACK}$	ST	AD6 to AD0	R/$\overline{W}$	$\overline{ACK}$	D7 to D0	$\overline{ACK}$	SP
			♦1	♦2					♦3	♦4		Δ5

♦ 1: IICSn register = 0001X110B

♦ 2: IICSn register = 0001X000B

♦ 3: IICSn register = 0001X110B

♦ 4: IICSn register = 0001X000B

Δ 5: IICSn register = 00000001B

Remarks 1. ♦ : Always generated

Δ: Generated only when SPIEn bit = 1

X: don't care

2. n = 0

<2> When WTIMn bit = 1 (after restart, address match)

ST	AD6 to AD0	R/$\overline{W}$	$\overline{ACK}$	D7 to D0	$\overline{ACK}$	ST	AD6 to AD0	R/$\overline{W}$	$\overline{ACK}$	D7 to D0	$\overline{ACK}$	SP
			♦1		♦2				♦3		♦4	Δ5

♦ 1: IICSn register = 0001X110B

♦ 2: IICSn register = 0001XX00B

♦ 3: IICSn register = 0001X110B

♦ 4: IICSn register = 0001XX00B

Δ 5: IICSn register = 00000001B

Remarks 1. ♦ : Always generated

Δ: Generated only when SPIEn bit = 1

X: don't care

2. n = 0

(3) Start ~ Address ~ Data ~ Start ~ Code ~ Data ~ Stop

<1> When WTIMn bit = 0 (after restart, extension code reception)

ST	AD6 to AD0	R/$\overline{\text{W}}$	$\overline{\text{ACK}}$	D7 to D0	$\overline{\text{ACK}}$	ST	AD6 to AD0	R/$\overline{\text{W}}$	$\overline{\text{ACK}}$	D7 to D0	$\overline{\text{ACK}}$	SP
			♦1	♦2				♦3		♦4		Δ5

♦ 1: IICSn register = 0001X110B

♦ 2: IICSn register = 0001X000B

♦ 3: IICSn register = 0010X010B

♦ 4: IICSn register = 0010X000B

Δ 5: IICSn register = 00000001B

Remarks 1. ♦ : Always generated
Δ: Generated only when SPIEn bit = 1
X: don’t care
2. n = 0 to 2

<2> When WTIMn bit = 1 (after restart, extension code reception)

ST	AD6 to AD0	R/$\overline{\text{W}}$	$\overline{\text{ACK}}$	D7 to D0	$\overline{\text{ACK}}$	ST	AD6 to AD0	R/$\overline{\text{W}}$	$\overline{\text{ACK}}$	D7 to D0	$\overline{\text{ACK}}$	SP
			♦1		♦2			♦3	♦4		♦5	Δ6

♦ 1: IICSn register = 0001X110B

♦ 2: IICSn register = 0001XX00B

♦ 3: IICSn register = 0010X010B

♦ 4: IICSn register = 0010X110B

♦ 5: IICSn register = 0010XX00B

Δ 6: IICSn register = 00000001B

Remarks 1. ♦ : Always generated
Δ: Generated only when SPIEn bit = 1
X: don’t care
2. n = 0 to 2

(4) Start ~ Address ~ Data ~ Start ~ Address ~ Data ~ Stop

<1> When WTIMn bit = 0 (after restart, address mismatch (= not extension code))

ST	AD6 to AD0	R/$\overline{W}$	$\overline{ACK}$	D7 to D0	$\overline{ACK}$	ST	AD6 to AD0	R/$\overline{W}$	$\overline{ACK}$	D7 to D0	$\overline{ACK}$	SP
			♦1	♦2					♦3			Δ4

♦ 1: IICSn register = 0001X110B

♦ 2: IICSn register = 0001X000B

♦ 3: IICSn register = 00000X10B

Δ 4: IICSn register = 00000001B

Remarks 1. ♦ : Always generated
Δ: Generated only when SPIEn bit = 1
X: don't care
2. n = 0

<2> When WTIMn bit = 1 (after restart, address mismatch (= not extension code))

ST	AD6 to AD0	R/$\overline{W}$	$\overline{ACK}$	D7 to D0	$\overline{ACK}$	ST	AD6 to AD0	R/$\overline{W}$	$\overline{ACK}$	D7 to D0	$\overline{ACK}$	SP
			♦1		♦2				♦3			Δ4

♦1: IICSn register = 0001X110B

♦2: IICSn register = 0001XX00B

♦3: IICSn register = 00000X10B

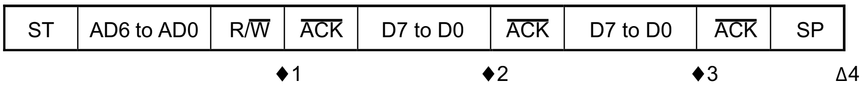
Δ 4: IICSn register = 00000001B

Remarks 1. ♦ : Always generated
Δ: Generated only when SPIEn bit = 1
X: don't care
2. n = 0

19.7.3 Slave device operation (when receiving extension code)

(1) Start ~ Code ~ Data ~ Data ~ Stop

<1> When WTIMn bit = 0



♦ 1: IICSn register = 0010X010B

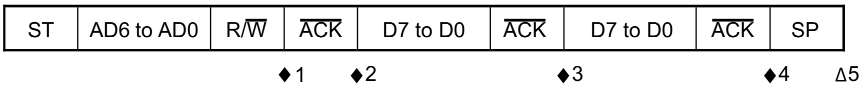
♦ 2: IICSn register = 0010X000B

♦ 3: IICSn register = 0010X000B

Δ 4: IICSn register = 00000001B

Remarks 1. ♦ : Always generated
Δ: Generated only when SPIEn bit = 1
X: don’t care
2. n = 0

<2> When WTIMn bit = 1



♦ 1: IICSn register = 0010X010B

♦ 2: IICSn register = 0010X110B

♦ 3: IICSn register = 0010X100B

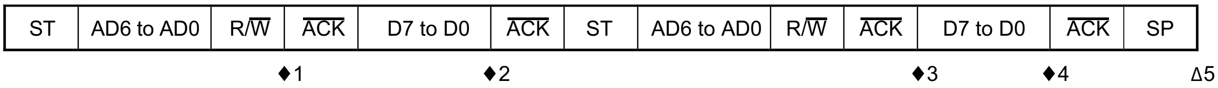
♦ 4: IICSn register = 0010XX00B

Δ 5: IICSn register = 00000001B

Remarks 1. ♦ : Always generated
Δ: Generated only when SPIEn bit = 1
X: don’t care
2. n = 0

(2) Start ~ Code ~ Data ~ Start ~ Address ~ Data ~ Stop

<1> When WTIMn bit = 0 (after restart, address match)



♦1: IICSn register = 0010X010B

♦2: IICSn register = 0010X000B

♦3: IICSn register = 0001X110B

♦4: IICSn register = 0001X000B

Δ 5: IICSn register = 00000001B

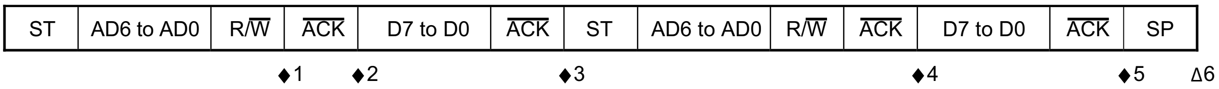
Remarks 1. ♦ : Always generated

Δ: Generated only when SPIEn bit = 1

X: don't care

2. n = 0

<2> When WTIMn bit = 1 (after restart, address match)



♦1: IICSn register = 0010X010B

♦2: IICSn register = 0010X110B

♦3: IICSn register = 0010XX00B

♦4: IICSn register = 0001X110B

♦5: IICSn register = 0001XX00B

Δ 6: IICSn register = 00000001B

Remarks 1. ♦ : Always generated

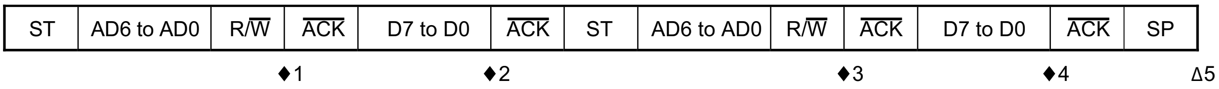
Δ: Generated only when SPIEn bit = 1

X: don't care

2. n = 0

(3) Start ~ Code ~ Data ~ Start ~ Code ~ Data ~ Stop

<1> When WTIMn bit = 0 (after restart, extension code reception)



♦ 1: IICSn register = 0010X010B

♦ 2: IICSn register = 0010X000B

♦ 3: IICSn register = 0010X010B

♦ 4: IICSn register = 0010X000B

Δ 5: IICSn register = 00000001B

Remarks 1. ♦ : Always generated

Δ: Generated only when SPIEn bit = 1

X: don’t care

2. n = 0

<2> When WTIMn bit = 1 (after restart, extension code reception)

♦ 1: IICSn register = 0010X010B

♦ 2: IICSn register = 0010X110B

♦ 3: IICSn register = 0010XX00B

♦ 4: IICSn register = 0010X010B

♦ 5: IICSn register = 0010X110B

♦ 6: IICSn register = 0010XX00B

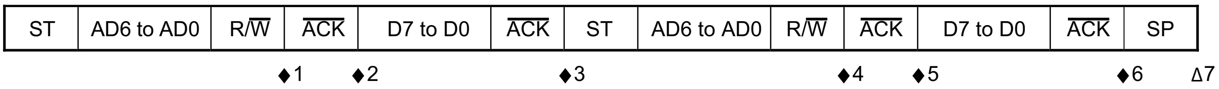
Δ 7: IICSn register = 00000001B

Remarks 1. ♦ : Always generated

Δ: Generated only when SPIEn bit = 1

X: don’t care

2. n = 0



(4) Start ~ Code ~ Data ~ Start ~ Address ~ Data ~ Stop

<1> When WTIMn bit = 0 (after restart, address mismatch (= not extension code))

ST	AD6 to AD0	R/$\overline{W}$	$\overline{ACK}$	D7 to D0	$\overline{ACK}$	ST	AD6 to AD0	R/$\overline{W}$	$\overline{ACK}$	D7 to D0	$\overline{ACK}$	SP
			♦1		♦2					♦3		Δ4

♦ 1: IICSn register = 0010X010B

♦ 2: IICSn register = 0010X000B

♦ 3: IICSn register = 00000X10B

Δ 4: IICSn register = 00000001B

Remarks 1. ♦: Always generated
Δ: Generated only when SPIEn bit = 1
X: don’t care
2. n = 0

<2> When WTIMn bit = 1 (after restart, address mismatch (= not extension code))

ST	AD6 to AD0	R/$\overline{W}$	$\overline{ACK}$	D7 to D0	$\overline{ACK}$	ST	AD6 to AD0	R/$\overline{W}$	$\overline{ACK}$	D7 to D0	$\overline{ACK}$	SP
			♦1	♦2		♦3				♦4		Δ5

♦ 1: IICSn register = 0010X010B

♦ 2: IICSn register = 0010X110B

♦ 3: IICSn register = 0010XX00B

♦ 4: IICSn register = 00000X10B

Δ 5: IICSn register = 00000001B

Remarks 1. ♦: Always generated
Δ: Generated only when SPIEn bit = 1
X: don’t care
2. n = 0

19.7.4 Operation without communication

(1) Start ~ Code ~ Data ~ Data ~ Stop

ST	AD6 to AD0	R/$\overline{\text{W}}$	$\overline{\text{ACK}}$	D7 to D0	$\overline{\text{ACK}}$	D7 to D0	$\overline{\text{ACK}}$	SP
								Δ1

Δ 1: IICSn register = 00000001B

Remarks 1. Δ: Generated only when SPIEn bit = 1

2. n = 0

19.7.5 Arbitration loss operation (operation as slave after arbitration loss)

(1) When arbitration loss occurs during transmission of slave address data

<1> When WTIMn bit = 0

ST	AD6 to AD0	R/$\overline{\text{W}}$	$\overline{\text{ACK}}$	D7 to D0	$\overline{\text{ACK}}$	D7 to D0	$\overline{\text{ACK}}$	SP
				♦1	♦2		♦3	Δ4

♦ 1: IICSn register = 0101X110B (Example: When ALDn bit is read during interrupt servicing)

♦ 2: IICSn register = 0001X000B

♦ 3: IICSn register = 0001X000B

Δ 4: IICSn register = 00000001B

Remarks 1. ♦ : Always generated

Δ: Generated only when SPIEn bit = 1

X: don’t care

2. n = 0

<2> When WTIMn bit = 1

ST	AD6 to AD0	R/$\overline{\text{W}}$	$\overline{\text{ACK}}$	D7 to D0	$\overline{\text{ACK}}$	D7 to D0	$\overline{\text{ACK}}$	SP
				♦1		♦2		♦3 Δ4

♦ 1: IICSn register = 0101X110B (Example: When ALDn bit is read during interrupt servicing)

♦ 2: IICSn register = 0001X100B

♦ 3: IICSn register = 0001XX00B

Δ 4: IICSn register = 00000001B

Remarks 1. ♦ : Always generated

Δ: Generated only when SPIEn bit = 1

X: don’t care

2. n = 0

(2) When arbitration loss occurs during transmission of extension code

<1> When WTIMn bit = 0

ST	AD6 to AD0	R/$\overline{W}$	$\overline{ACK}$	D7 to D0	$\overline{ACK}$	D7 to D0	$\overline{ACK}$	SP
			♦1		♦2		♦3	Δ4

♦ 1: IICSn register = 0110X010B (Example: When ALDn bit is read during interrupt servicing)

♦ 2: IICSn register = 0010X000B

♦ 3: IICSn register = 0010X000B

Δ 4: IICSn register = 00000001B

Remarks 1. ♦ : Always generated
Δ: Generated only when SPIEn bit = 1
X: don't care
2. n = 0

<2> When WTIMn bit = 1

ST	AD6 to AD0	R/$\overline{W}$	$\overline{ACK}$	D7 to D0	$\overline{ACK}$	D7 to D0	$\overline{ACK}$	SP
			♦1	♦2		♦3		♦4 Δ5

♦ 1: IICSn register = 0110X010B (Example: When ALDn bit is read during interrupt servicing)

♦ 2: IICSn register = 0010X110B

♦ 3: IICSn register = 0010X100B

♦ 4: IICSn register = 0010XX00B

Δ 5: IICSn register = 00000001B

Remarks 1. ♦ : Always generated
Δ: Generated only when SPIEn bit = 1
X: don't care
2. n = 0

19.7.6 Operation when arbitration loss occurs

(1) When arbitration loss occurs during transmission of slave address data

ST	AD6 to AD0	R/$\overline{\text{W}}$	$\overline{\text{ACK}}$	D7 to D0	$\overline{\text{ACK}}$	D7 to D0	$\overline{\text{ACK}}$	SP
				♦1				Δ2

♦ 1: IICSn register = 01000110B (Example: When ALDn bit is read during interrupt servicing)

Δ 2: IICSn register = 00000001B

Remarks 1. ♦ : Always generated

Δ: Generated only when SPIEn bit = 1

2. n = 0

(2) When arbitration loss occurs during transmission of extension code

ST	AD6 to AD0	R/$\overline{\text{W}}$	$\overline{\text{ACK}}$	D7 to D0	$\overline{\text{ACK}}$	D7 to D0	$\overline{\text{ACK}}$	SP
			♦1					Δ2

♦ 1: IICSn register = 0110X010B (Example: When ALDn bit is read during interrupt servicing)

IICCn.LRELn bit is set to 1 by software

Δ 2: IICSn register = 00000001B

Remarks 1. ♦ : Always generated

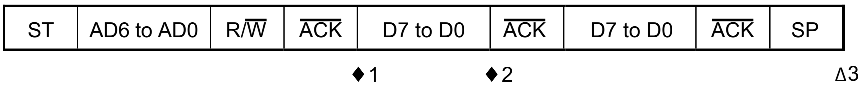
Δ: Generated only when SPIEn bit = 1

X: don’t care

2. n = 0

(3) When arbitration loss occurs during data transfer

<1> When WTIMn bit = 0



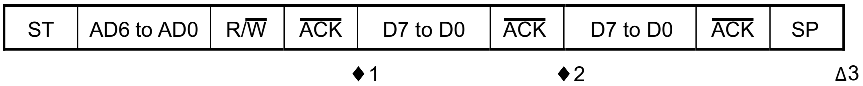
♦ 1: IICSn register = 10001110B

♦ 2: IICSn register = 01000000B (Example: When ALDn bit is read during interrupt servicing)

Δ 3: IICSn register = 00000001B

Remarks **1.** ♦ : Always generated
Δ: Generated only when SPIEn bit = 1
2. n = 0

<2> When WTIMn bit = 1



♦ 1: IICSn register = 10001110B

♦ 2: IICSn register = 01000100B (Example: When ALDn bit is read during interrupt servicing)

Δ 3: IICSn register = 00000001B

Remarks **1.** ♦ : Always generated
Δ: Generated only when SPIEn bit = 1
2. n = 0

(4) When arbitration loss occurs due to restart condition during data transfer

<1> Not extension code (Example: Address mismatch)

ST	AD6 to AD0	R/$\overline{W}$	$\overline{ACK}$	D7 to D0	ST	AD6 to AD0	R/$\overline{W}$	$\overline{ACK}$	D7 to D0	$\overline{ACK}$	SP	
			♦1					♦2				Δ3

♦ 1: IICSn register = 1000X110B

♦ 2: IICSn register = 01000110B (Example: When ALDn bit is read during interrupt servicing)

Δ 3: IICSn register = 00000001B

Remarks 1. ♦ : Always generated

Δ: Generated only when SPIEn bit = 1

X: don't care

2. Dn = D6 to D0

n = 0

<2> Extension code

ST	AD6 to AD0	R/$\overline{W}$	$\overline{ACK}$	D7 to D0	ST	AD6 to AD0	R/$\overline{W}$	$\overline{ACK}$	D7 to D0	$\overline{ACK}$	SP	
			♦1					♦2				Δ3

♦ 1: IICSn register = 1000X110B

♦ 2: IICSn register = 0110X010B (Example: When ALDn bit is read during interrupt servicing)

IICCn.LRELn bit is set to 1 by software

Δ 3: IICSn register = 00000001B

Remarks 1. ♦ : Always generated

Δ: Generated only when SPIEn bit = 1

X: don't care

2. Dn = D6 to D0

n = 0

(5) When arbitration loss occurs due to stop condition during data transfer

ST	AD6 to AD0	R/$\overline{W}$	$\overline{ACK}$	D7 to D0	ST	
			♦1			Δ2

♦ 1: IICSn register = 1000X110B

Δ 2: IICSn register = 01000001B

Remarks 1. ♦ : Always generated

Δ: Generated only when SPIEn bit = 1

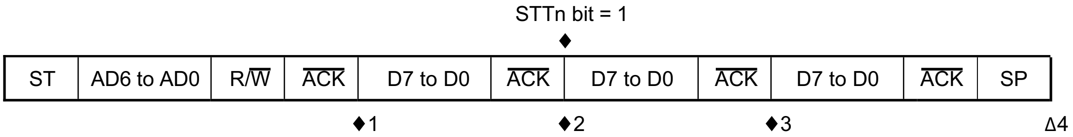
X: don't care

2. Dn = D6 to D0

n = 0

(6) When arbitration loss occurs due to low level of SDA0n pin when attempting to generate a restart condition

When WTIMn bit = 1



♦ 1: IICSn register = 1000X110B

♦ 2: IICSn register = 1000XX00B

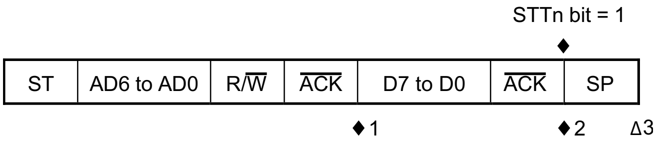
♦ 3: IICSn register = 01000100B (Example: When ALDn bit is read during interrupt servicing)

Δ 4: IICSn register = 00000001B

Remarks 1. ♦ : Always generated
Δ: Generated only when SPIEn bit = 1
X: don't care
2. n = 0

(7) When arbitration loss occurs due to a stop condition when attempting to generate a restart condition

When WTIMn bit = 1



♦ 1: IICSn register = 1000X110B

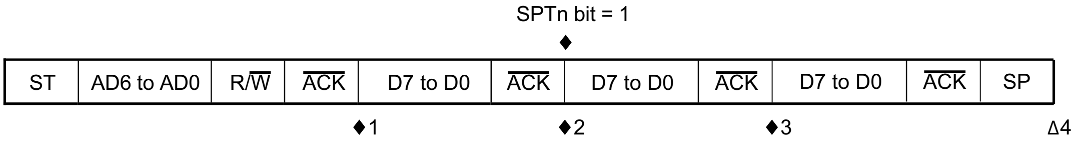
♦ 2: IICSn register = 1000XX00B

Δ 3: IICSn register = 01000001B

Remarks 1. ♦ : Always generated
Δ: Generated only when SPIEn bit = 1
X: don't care
2. n = 0

(8) When arbitration loss occurs due to low level of SDA0n pin when attempting to generate a stop condition

When WTIMn bit = 1



♦ 1: IICSn register = 1000X110B

♦ 2: IICSn register = 1000XX00B

♦ 3: IICSn register = 01000000B (Example: When ALDn bit is read during interrupt servicing)

Δ 4: IICSn register = 00000001B

Remarks 1. ♦ : Always generated

Δ: Generated only when SPIEn bit = 1

X: don’t care

2. n = 0

19.8 Interrupt Request Signal (INTIICn)

The setting of the IICCn.WTIMn bit determines the timing by which the INTIICn register is generated and the corresponding wait control, as shown below.

Table 19-12 INTIICn generation timing and wait control

WTIMn Bit	During Slave Device Operation			During Master Device Operation		
	Address	Data Reception	Data Transmission	Address	Data Reception	Data Transmission
0	**9**[Notes 1, 2]	**8**[Note 2]	**8**[Note 2]	9	8	8
1	**9**[Notes 1, 2]	**9**[Note 2]	**9**[Note 2]	9	9	9

Note 1. The slave device's INTIICn signal and wait period occur at the falling edge of the ninth clock only when there is a match with the address set to the SVAn register.
At this point, the $\overline{\text{ACK}}$ signal is output regardless of the value set to the IICCn.ACKEn bit. For a slave device that has received an extension code, the INTIICn signal occurs at the falling edge of the eighth clock.
When the address does not match after restart, the INTIICn signal is generated at the falling edge of the ninth clock, but no wait occurs.

2. If the received address does not match the contents of the SVAn register and an extension code is not received, neither the INTIICn signal nor a wait occurs.

3. The numbers in the table indicate the number of the serial clock's clock signals. Interrupt requests and wait control are both synchronized with the falling edge of these clock signals.

(1) During address transmission/reception

- Slave device operation:
  Interrupt and wait timing are determined regardless of the WTIMn bit.
- Master device operation:
  Interrupt and wait timing occur at the falling edge of the ninth clock regardless of the WTIMn bit.

(2) During data reception

- Master/slave device operation: Interrupt and wait timing is determined according to the WTIMn bit.

(3) During data transmission

- Master/slave device operation: Interrupt and wait timing is determined according to the WTIMn bit.

(4) Wait cancellation method

The four wait cancellation methods are as follows.

- By setting the IICCn.WRELn bit to 1
- By writing to the IICn register
- By start condition setting (IICCn.STTn bit = 1)Note
- By stop condition setting (IICCn.SPTn bit = 1)Note

Note Master only

When an 8-clock wait has been selected (WTIMn bit = 0), the output level of the $\overline{\text{ACK}}$ signal must be determined prior to wait cancellation.

(5) Stop condition detection

The INTIICn signal is generated when a stop condition is detected.

19.9 Address Match Detection Method

In I^2C bus mode, the master device can select a particular slave device by transmitting the corresponding slave address.

Address match detection is performed automatically by hardware. The INTIICn signal occurs when a local address has been set to the SVAn register and when the address set to the SVAn register matches the slave address sent by the master device, or when an extension code has been received.

19.10 Error Detection

In I^2C bus mode, the status of the serial data bus pin (SDA0n) during data transmission is captured by the IICn register of the transmitting device, so the data of the IICn register prior to transmission can be compared with the transmitted IICn data to enable detection of transmission errors. A transmission error is judged as having occurred when the compared data values do not match.

19.11 Extension Code

- When the higher 4 bits of the receive address are either 0000_B or 1111_B, the extension code flag (IICSn.EXCn bit) is set for extension code reception and an interrupt request signal (INTIICn) is issued at the falling edge of the eighth clock.

  The local address stored in the SVAn register is not affected.

- If $11110\times\times0_B$ is set to the SVAn register by a 10-bit address transfer and $11110\times\times0_B$ is transferred from the master device, the results are as follows. Note that the INTIICn signal occurs at the falling edge of the eighth clock

  – Higher four bits of data match: EXCn bit = 1

  – Seven bits of data match: IICSn.COIn bit = 1

- Since the processing after the interrupt request signal occurs differs according to the data that follows the extension code, such processing is performed by software.

  For example, when operation as a slave is not desired after the extension code is received, set the IICCn.LRELn bit to 1 and the CPU will enter the next communication wait state.

Table 19-13 Extension code bit definitions

Slave Address	R/W Bit	Description
$0000\ 000_B$	0	General call address
$0000\ 000_B$	1	Start byte
$0000\ 001_B$	×	CBUS address
$0000\ 010_B$	×	Address that is reserved for different bus format
$1111\ 0\times\times_B$	×	10-bit slave address specification

19.12 Arbitration

When several master devices simultaneously output a start condition (when the IICCn.STTn bit is set to 1 before the IICSn.STDn bit is set to 1), communication between the master devices is performed while the number of clocks is adjusted until the data differs. This kind of operation is called arbitration.

When one of the master devices loses in arbitration, an arbitration loss flag (IICSn.ALDn bit) is set to 1 via the timing by which the arbitration loss occurred, and the SCL0n and SDA0n lines are both set to high impedance, which releases the bus.

Arbitration loss is detected based on the timing of the next interrupt request signal (the eighth or ninth clock, when a stop condition is detected, etc.) and the setting of the ALDn bit to 1, which is made by software.

For details of interrupt request timing, see *"I^{2}C Interrupt Request Signals (INTIICn)" on page 637*.

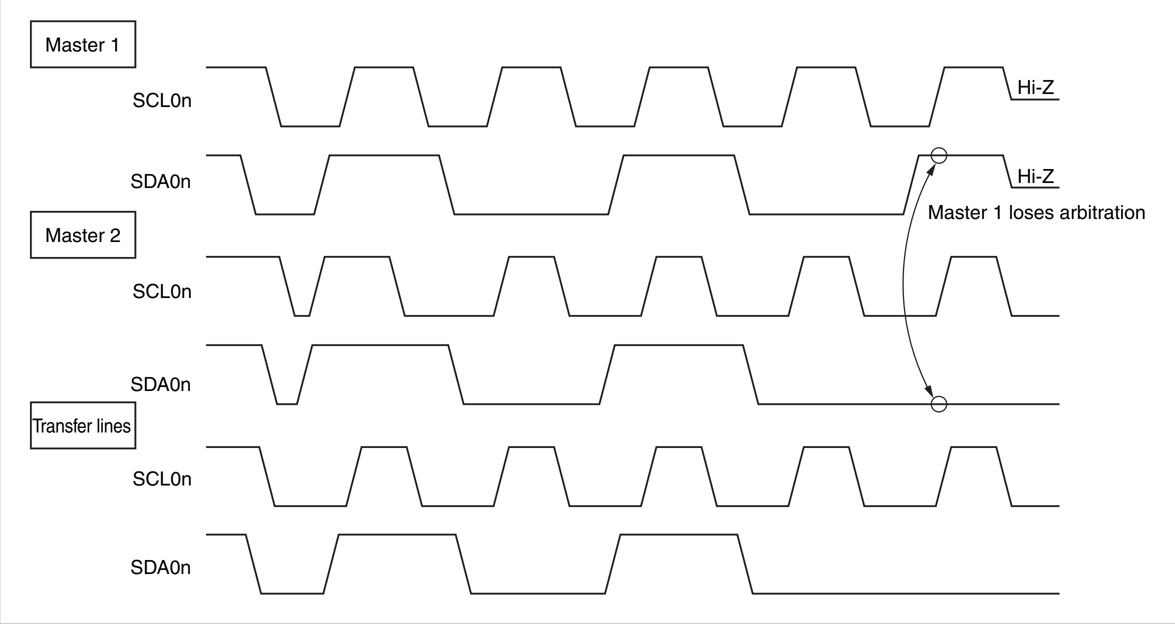


Figure 19-13 Arbitration timing example

Table 19-14 Status during arbitration and interrupt request signal generation timing

Status During Arbitration	Interrupt Request Generation Timing
Transmitting address transmission	At falling edge of eighth or ninth clock following byte transfer[a]
Read/write data after address transmission	
Transmitting extension code	
Read/write data after extension code transmission	
Transmitting data	
$\overline{\text{ACK}}$ signal transfer period after data reception	
When restart condition is detected during data transfer	
When stop condition is detected during data transfer	When stop condition is output (when IICCn.SPIEn bit = 1)[b]
When SDA0n pin is low level while attempting to output restart condition	At falling edge of eighth or ninth clock following byte transfer[a]
When stop condition is detected while attempting to output restart condition	When stop condition is output (when IICCn.SPIEn bit = 1)[b]
When DSA0n pin is low level while attempting to output stop condition	At falling edge of eighth or ninth clock following byte transfer[a]
When SCL0n pin is low level while attempting to output restart condition	

a) When the IICCn.WTIMn bit = 1, an interrupt request signal occurs at the falling edge of the ninth clock. When the WTIMn bit = 0 and the extension code's slave address is received, an interrupt request signal occurs at the falling edge of the eighth clock.

b) When there is a possibility that arbitration will occur, set the SPIEn bit to 1 for master device operation.

19.13 Wakeup Function

The I^{2}C bus slave function is a function that generates an interrupt request signal (INTIICn) when a local address and extension code have been received.

This function makes processing more efficient by preventing unnecessary interrupt request signals from occurring when addresses do not match.

When a start condition is detected, wakeup stand-by mode is set. This wakeup stand-by mode is in effect while addresses are transmitted due to the possibility that an arbitration loss may change the master device (which has output a start condition) to a slave device.

However, when a stop condition is detected, the IICCn.SPIEn bit is set regardless of the wakeup function, and this determines whether interrupt request signals are enabled or disabled.

19.14 Cautions

(1) When IICFn.STCENn bit = 0

Immediately after the I^{2}Cn operation is enabled, the bus communication status (IICFn.IICBSYn bit = 1) is recognized regardless of the actual bus status. To execute master communication in the status where a stop condition has not been detected, generate a stop condition and then release the bus before starting the master communication.

Use the following sequence for generating a stop condition.

<1> Set the IICCLn register.

<2> Set the IICCn.IICEn bit.

<3> Set the IICCn.SPTn bit.

(2) When IICFn.STCENn bit = 1

Immediately after I^{2}Cn operation is enabled, the bus released status (IICBSYn bit = 0) is recognized regardless of the actual bus status. To issue the first start condition (IICCn.STTn bit = 1), it is necessary to confirm that the bus has been released, so as to not disturb other communications.

(3)

When the IICC0.IICE0 bit of the microcontroller is set to 1 while communications with other devices are in progress, the start condition may be detected depending on the status of the communication line. Be sure to set the IICC0.IICE0 bit to 1 when the SCL00 and SDA00 lines are high level.

(4)

Determine the operation clock frequency by the IICCL0, IICX0, and OCKS0 registers before enabling the operation (IICC0.IICE0 bit = 1). To change the operation clock frequency, clear the IICC0.IICE0 bit to 0 once.

(5)

After the IICC0.STT0 and IICC0.SPT0 bits have been set to 1, they must not be re-set without being cleared to 0 first.

(6)

If transmission has been reserved, set the IICCN.SPIE0 bit to 1 so that an interrupt request is generated by the detection of a stop condition. After an interrupt request has been generated, the wait state will be released by writing communication data to I2C0, then transferring will begin. If an interrupt is not generated by the detection of a stop condition, transmission will halt in the wait state because an interrupt request was not generated.

However, it is not necessary to set the SPIE0 bit to 1 for the software to detect the IICS0.MSTS0 bit.

19.15 Communication Operations

19.15.1 Master operation 1

The following flowchart shows the master communication when the communication reservation function is enabled (IICFn.IICRSVn = 0) and the master operation is started after detecting a stop condition (IICFn.STCENn = 0).

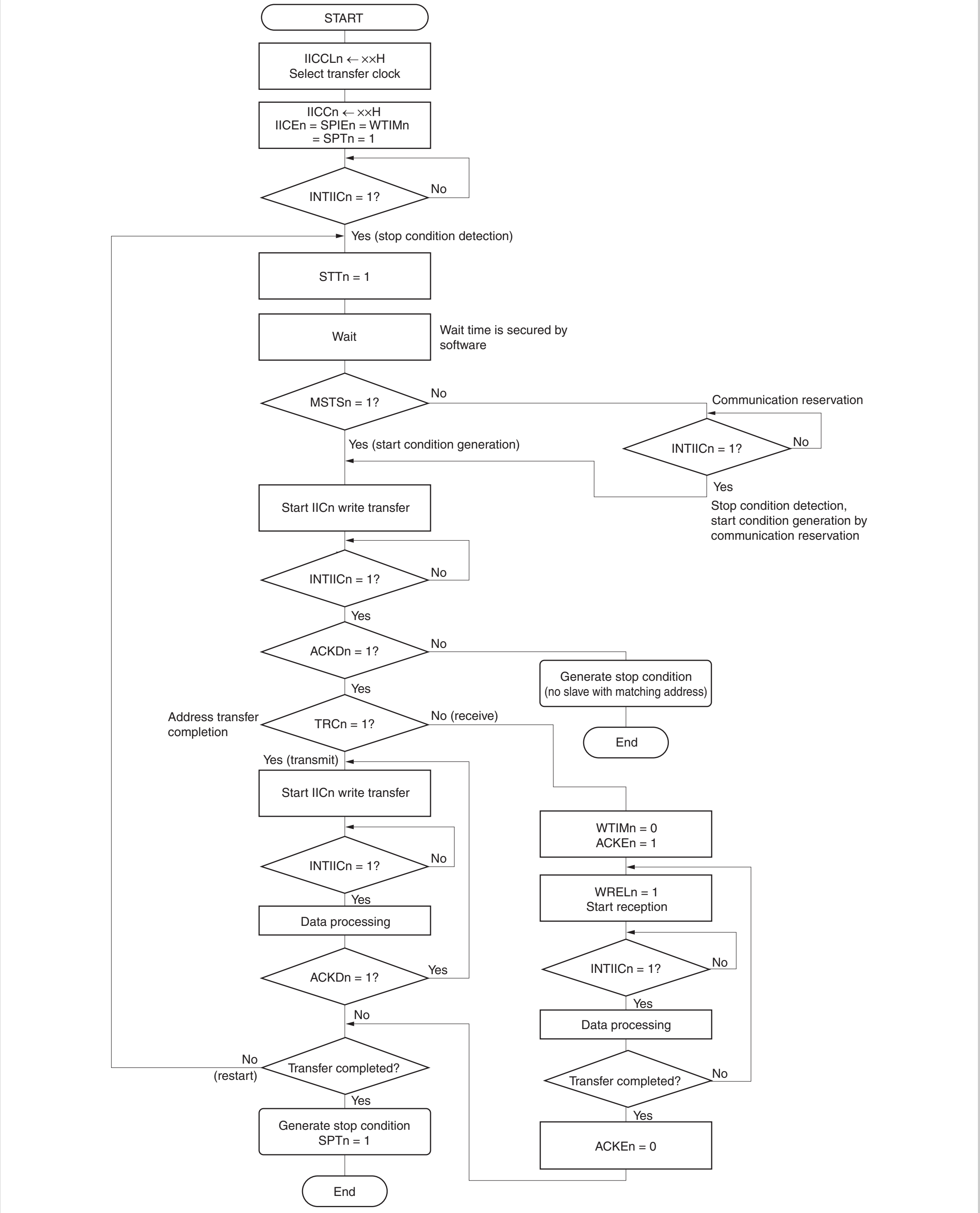


Figure 19-14 Master operation flowchart (1)

19.15.2 Master operation 2

The following flowchart showas the master communication when the communication reservation function is disabled (IICRSVn = 1) and the master operation is started without detecting a stop condition (STCENn = 1).

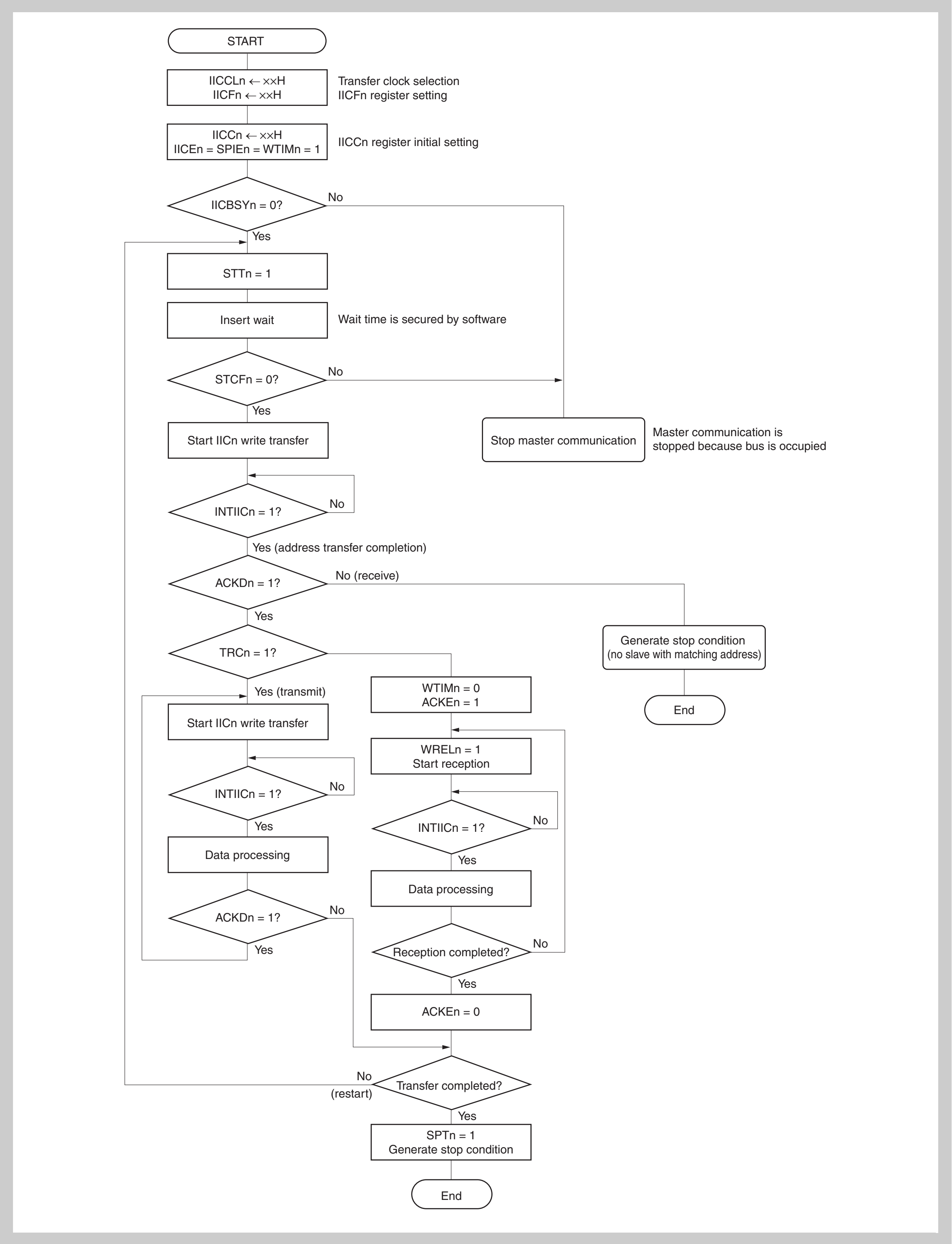


Figure 19-15 Master operation flowchart (2)

19.15.3 Slave operation

The following shows the processing procedure of the slave operation.

Basically, the operation of the slave device is event-driven. Therefore, processing by an INTIICn interrupt (processing requiring a significant change of the operation status, such as stop condition detection during communication) is necessary.

The following description assumes that data communication does not support extension codes. Also, it is assumed that the INTIICn interrupt servicing performs only status change processing and that the actual data communication is performed during the main processing.

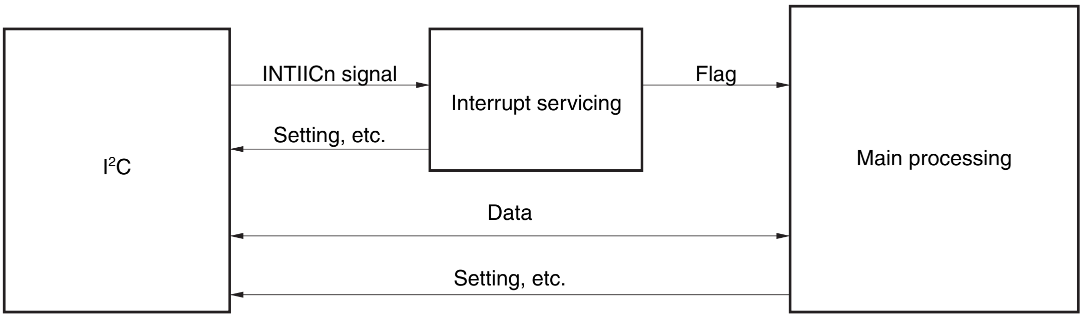


Figure 19-16 Software outline during slave operation

Therefore, the following three flags are prepared so that the data transfer processing can be performed by transmitting these flags to the main processing instead of INTIICn signal.

(1) Communication mode flag

This flag indicates the following communication statuses.

- Clear mode:
  Data communication not in progress
- Communication mode
  Data communication in progress (valid address detection / stop condition detection, $\overline{\text{ACK}}$ signal from master not detected, address mismatch)

(2) Ready flag

This flag indicates that data communication is enabled. This is the same status as an INTIICn interrupt during normal data transfer. This flag is set in the interrupt processing block and cleared in the main processing block. The ready flag for the first data for transmission is not set in the interrupt processing block, so the first data is transmitted without clear processing (the address match is regarded as a request for the next data).

(3) Communication direction flag

This flag indicates the direction of communication and is the same as the value of IICSn.TRCn bit.

The following shows the operation of the main processing block during slave operation.

Start I^{2}Cn and wait for the communication enabled status. When communication is enabled, perform transfer using the communication mode flag and ready flag (the processing of the stop condition and start condition is performed by interrupts, conditions are confirmed by flags).

For transmission, repeat the transmission operation until the master device stops returning $\overline{\text{ACK}}$ signal. When the master device stops returning $\overline{\text{ACK}}$ signal, transfer is complete.

For reception, receive the required number of data and do not return $\overline{\text{ACK}}$ signal for the next data immediately after transfer is complete. After that, the master device generates the stop condition or restart condition. This causes exit from communications.

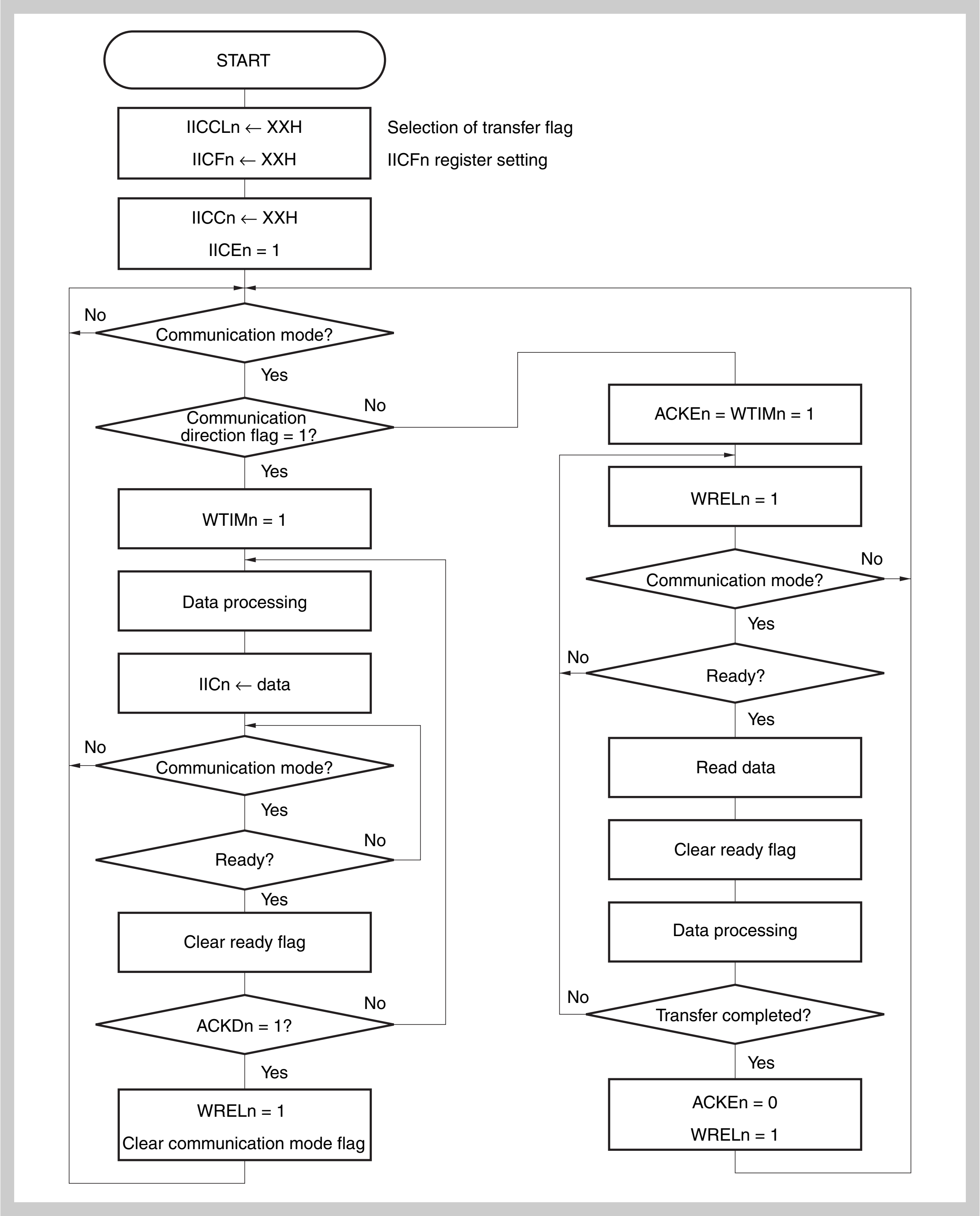


Figure 19-17 Slave operation flowchart (1)

The following shows an example of the processing of the slave device by an INTIICn interrupt (it is assumed that no extension codes are used here). During an INTIICn interrupt, the status is confirmed and the following steps are executed.

<1> When a stop condition is detected, communication is terminated.

<2> When a start condition is detected, the address is confirmed. If the address does not match, communication is terminated. If the address matches, the communication mode is set and wait is released, and operation returns from the interrupt (the ready flag is cleared).

<3> For data transmission/reception, when the ready flag is set, operation returns from the interrupt while the IIC0n bus remains in the wait status.

Note <1> to <3> in the above correspond to <1> to <3> in *Figure 19-18*.

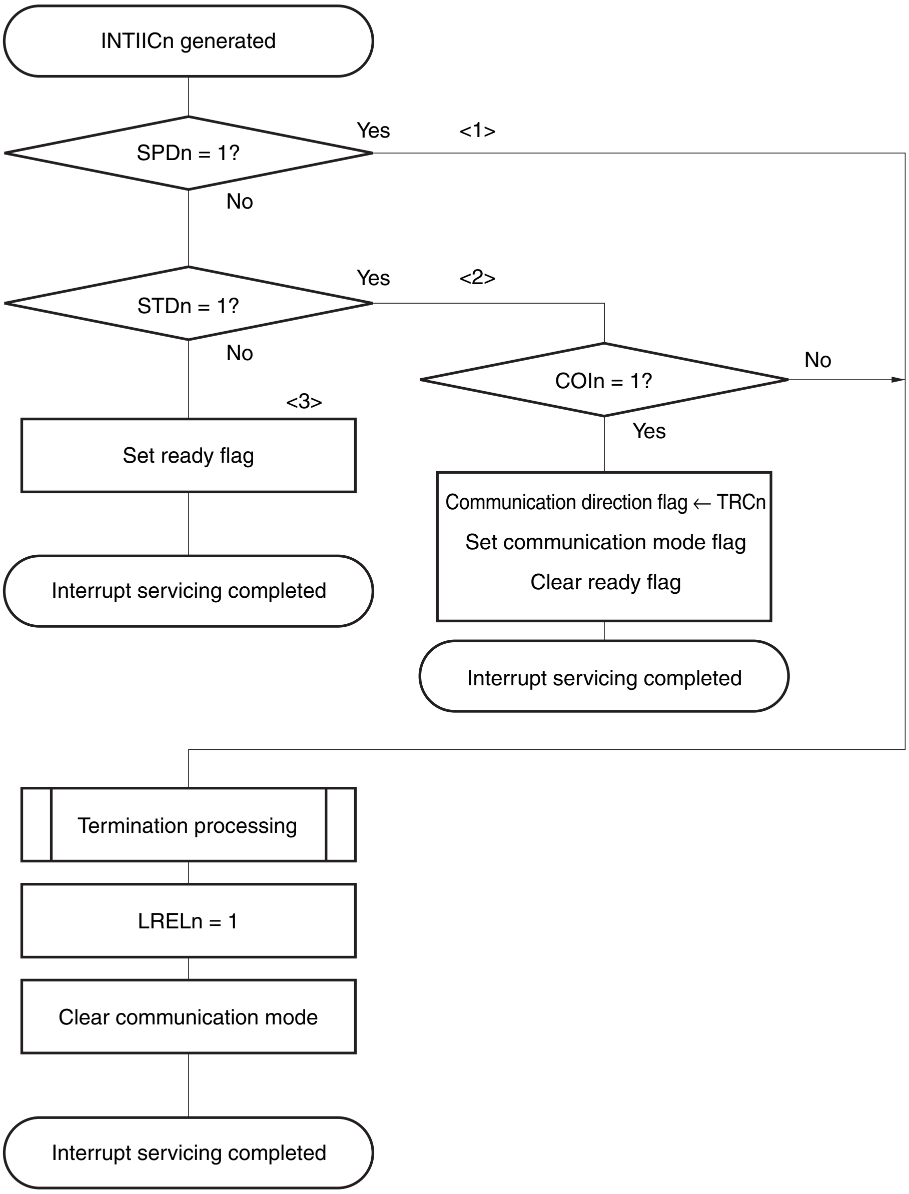


Figure 19-18 Slave operation flowchart (2)

19.16 Timing of Data Communication

When using I^2C bus mode, the master device outputs an address via the serial bus to select one of several slave devices as its communication partner.

After outputting the slave address, the master device transmits the IICSn.TRCn bit, which specifies the data transfer direction, and then starts serial communication with the slave device.

The shift operation of the IICn register is synchronized with the falling edge of the serial clock pin (SCL0n). The transmit data is transferred to the SO latch and is output (MSB first) via the SDA0n pin.

Data input via the SDA0n pin is captured by the IICn register at the rising edge of the SCL0n pin.

The data communication timing is shown below.

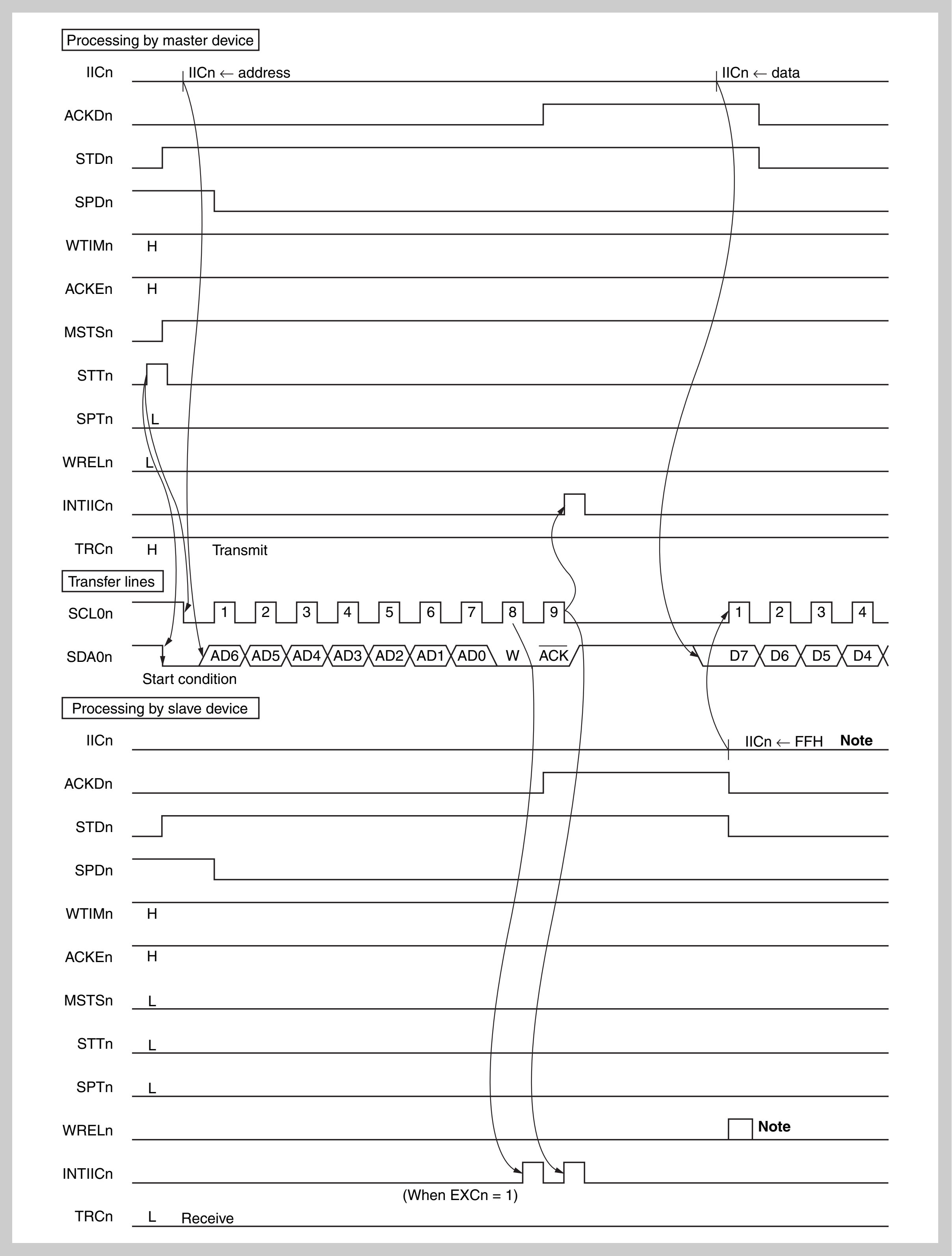


Figure 19-19 Example of master to slave communication (when 9-clock wait is selected for both master and slave) (1/3) start condition ~ address

Note To cancel slave wait, write FF_H to IICn or set WRELn.

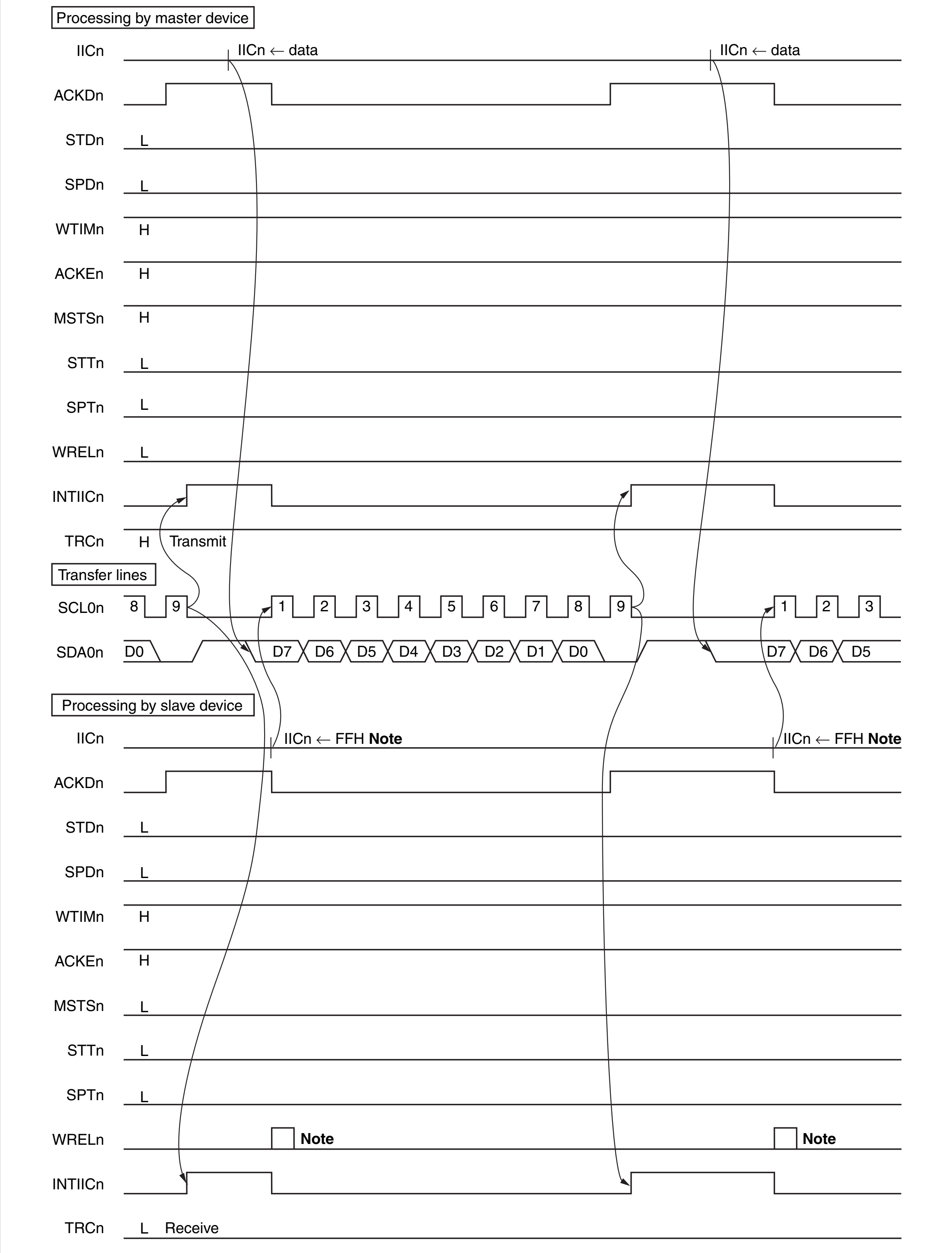


Figure 19-20 **Example of master to slave communication (when 9-clock wait is selected for both master and slave) (2/3) (b) data**

Note To cancel slave wait, write FF_H to IICn or set WRELn.

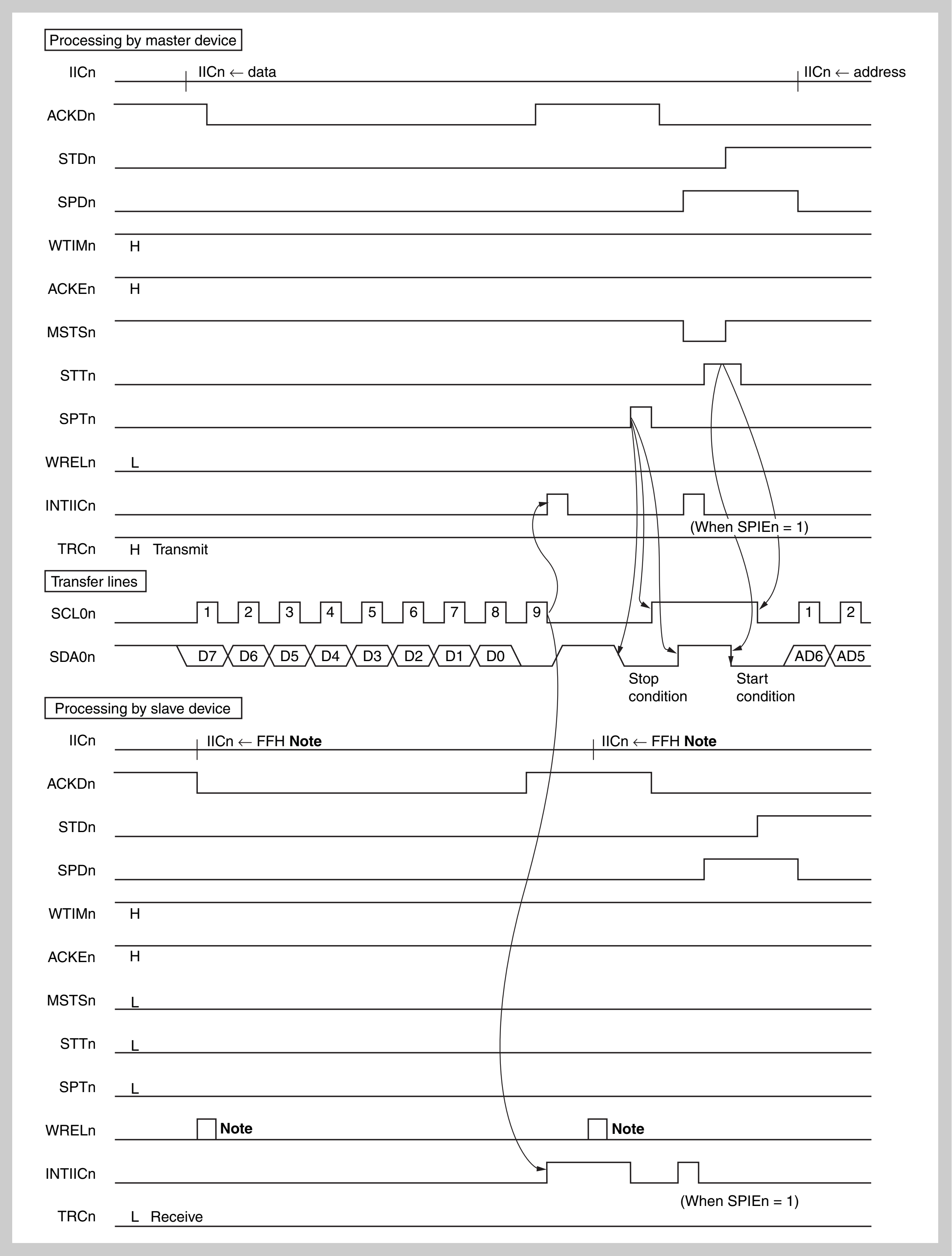


Figure 19-21 Example of master to slave communication (when 9-clock wait is selected for both master and slave) (3/3) (c) stop condition

Note To cancel slave wait, write FF_H to IICn or set WRELn.

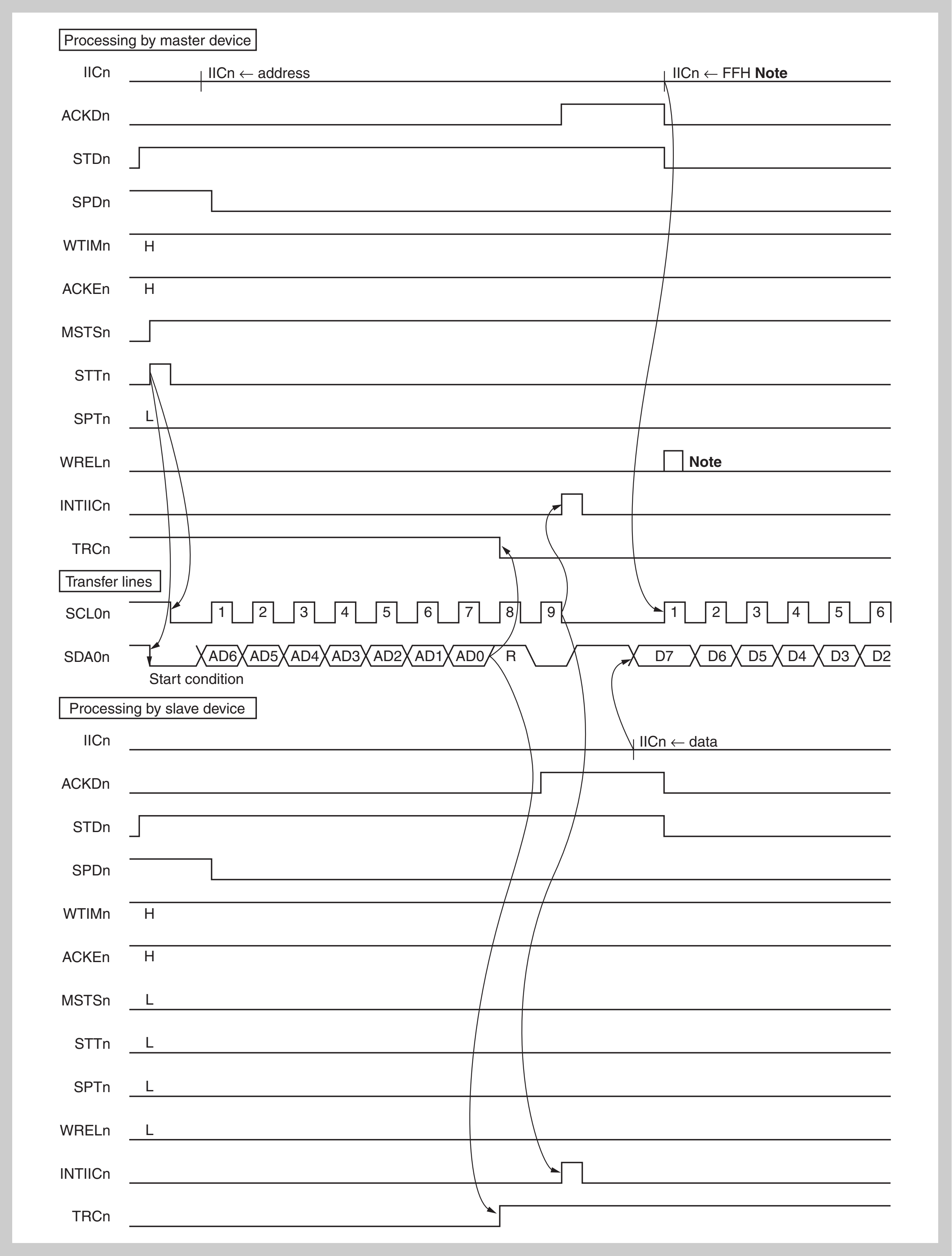


Figure 19-22 Example of slave to master communication (when 9-clock wait is selected for both master and slave) (1/3) (a) start condition ~ address

Note To cancel master wait, write FF_H to IICn or set WRELn.

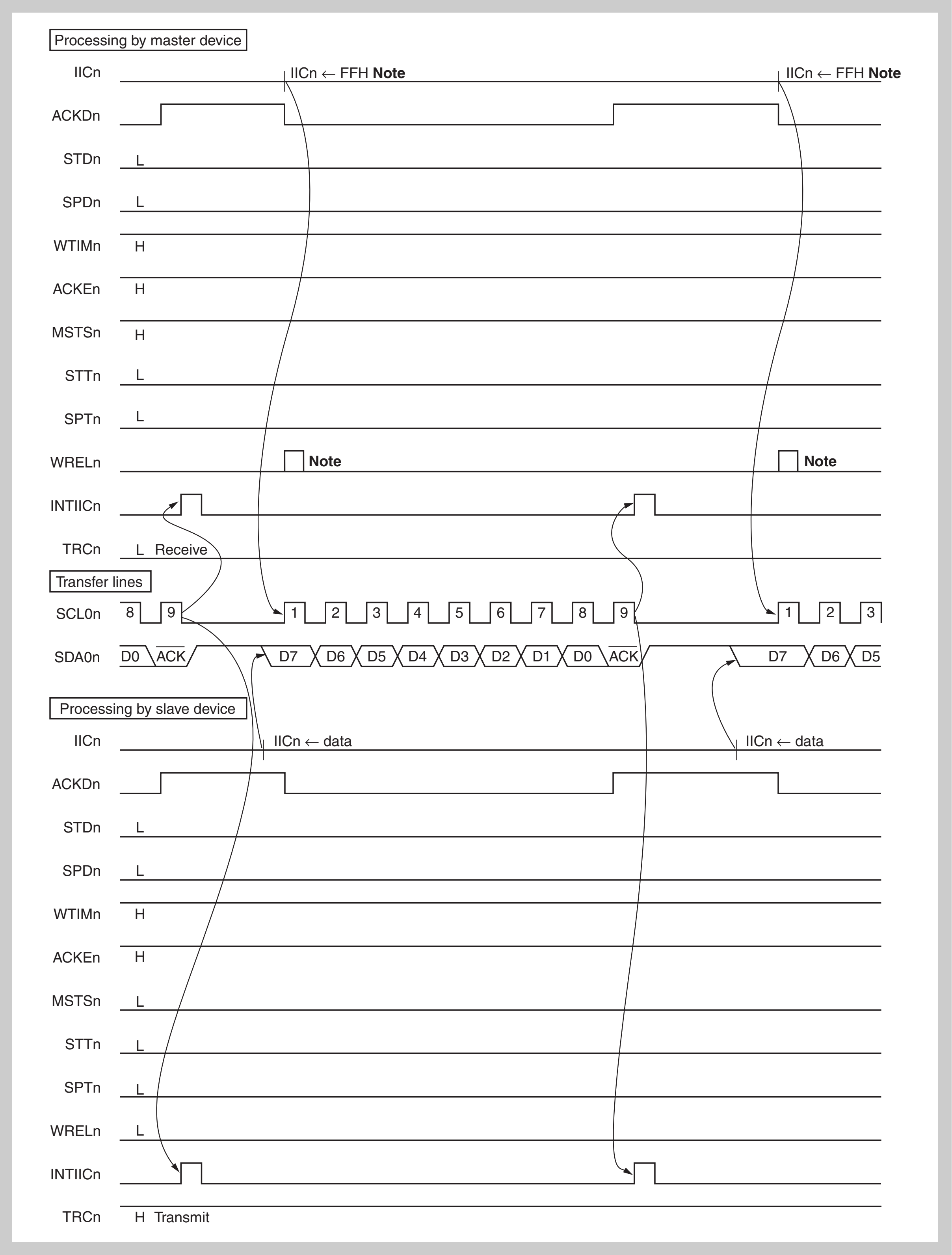


Figure 19-23 **Example of slave to master communication (when 9-clock wait is selected for both master and slave) (2/3) (b) data**

Note To cancel master wait, write FF_H to IICn or set WRELn.

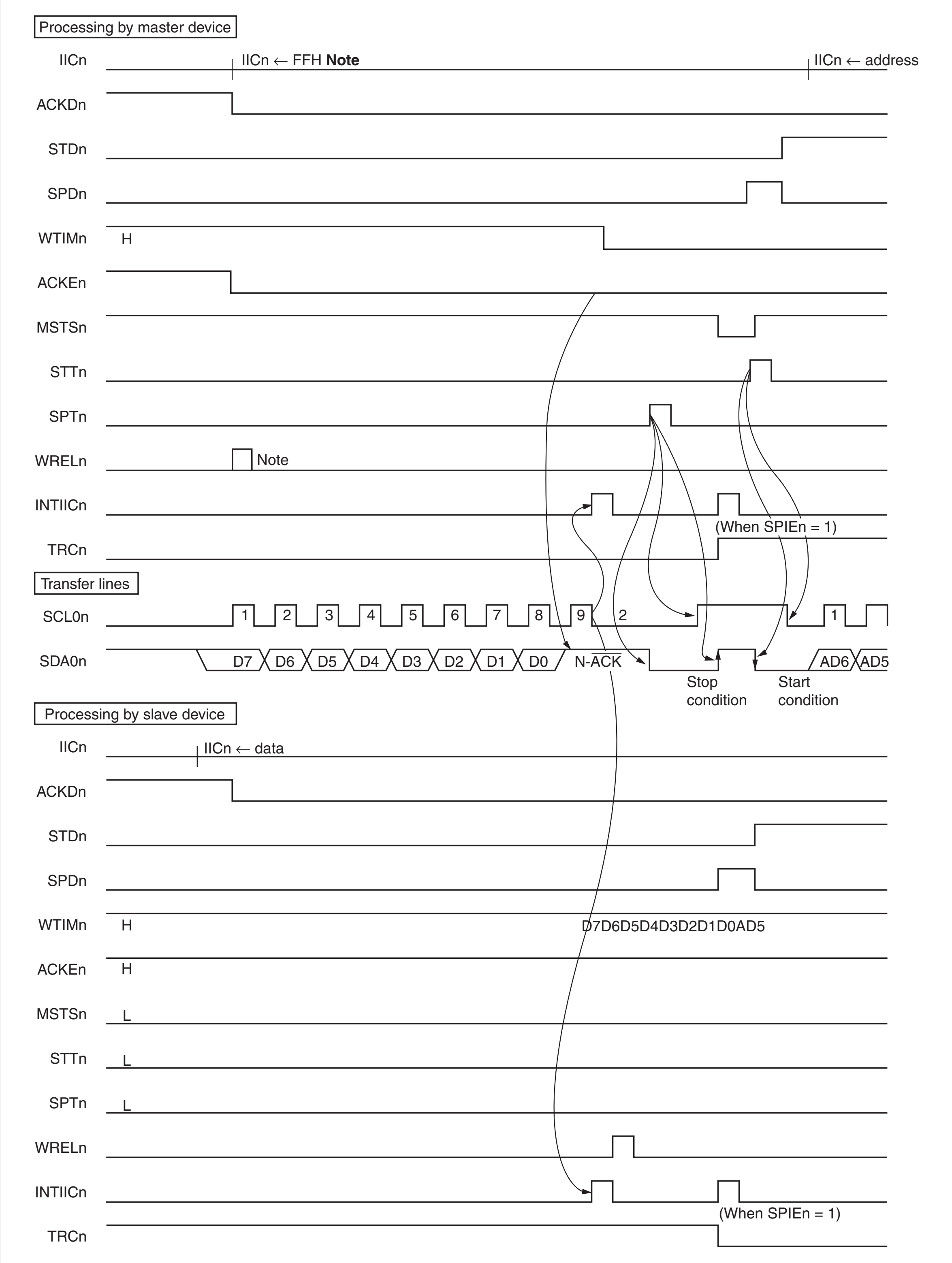


Figure 19-24 Example of slave to master communication (when 9-clock wait is selected for both master and slave) (3/3) (c) stop condition

Note To cancel master wait, write FF_H to IICn or set WRELn.

Chapter 20 CAN Controller (CAN)

These microcontrollers feature an on-chip n-channel CAN (Controller Area Network) controller that complies with the CAN protocol as standardized in ISO 11898.

The number of CAN channels is given in the table below:

CAN	V850ES/FE3	V850ES/FF3	V850ES/FG3	V850ES/FJ3		V850ES/FK3
				µPD70F3378	µPD70F3379 µPD70F3380 µPD70F3381 µPD70F3382	
Channels	1		2	3	4	5
Names	CAN0		CAN0 to CAN1	CAN0 to CAN2	CAN0 to CAN3	CAN0 to CAN4

Throughout this chapter, the individual channels of CAN are identified by “n”, for example, C0GMCTRL for the CAN0 global control register.

Throughout this chapter, the CAN message buffer registers are identified by “m” (m = 0 to 31), for example C0MDATA4m for CAN0 message data byte 4 of message buffer register m.

20.1 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 TAA0 to TAAn capture channels

20.1.1 Overview of functions

Table 20-1 presents an overview of the CAN Controller functions.

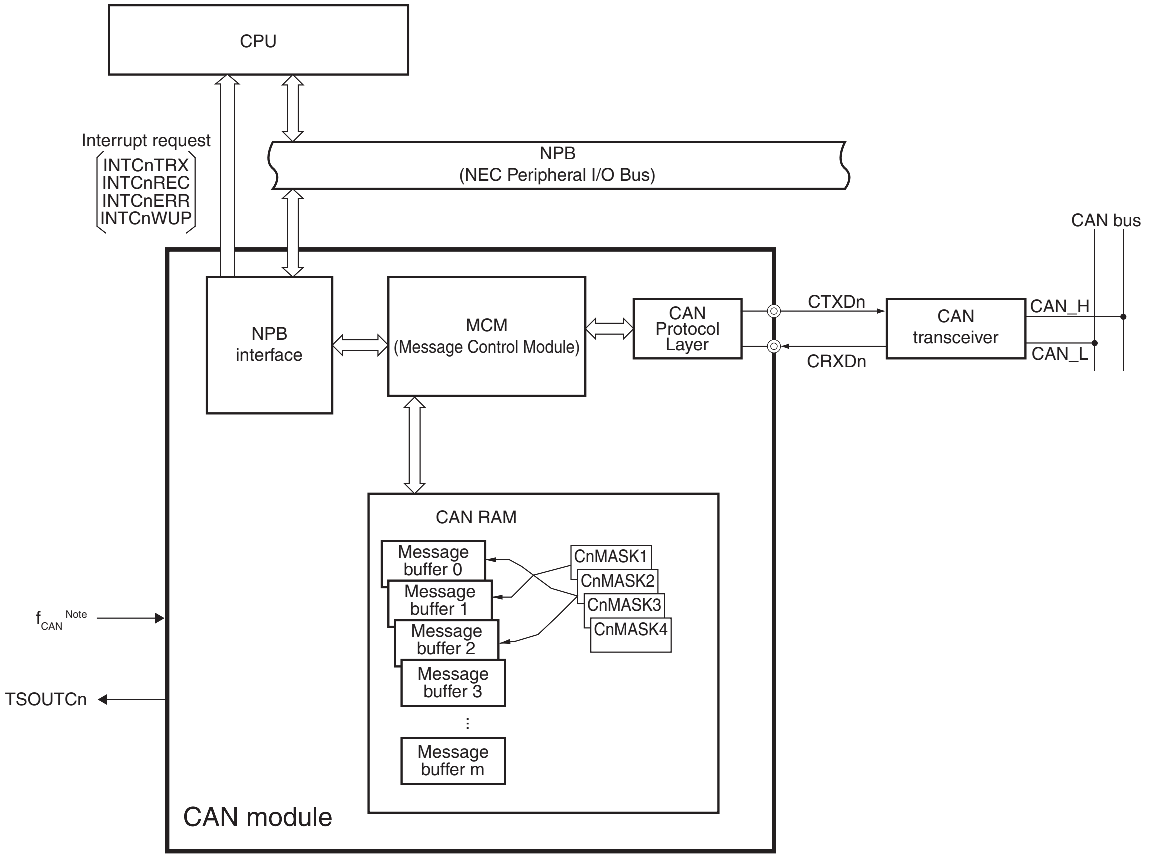
Table 20-1 Overview of functions

Function	Details
Protocol	CAN protocol ISO 11898 (standard and extended frame transmission/reception)
Baud rate	Maximum 1 Mbps (CAN clock input ≥ 8 MHz)
Data storage	Storing messages in the CAN RAM
Number of messages	• 32 message buffers per channel • Each message buffer can be set to be either a transmit message buffer or a receive message buffer.
Message reception	• Unique ID can be set to each message buffer. • Mask setting of four patterns is possible for each channel. • A receive completion interrupt is generated each time a message is received and stored in a message buffer. • Two or more receive message buffers can be used as a FIFO receive buffer (multi-buffer receive block function). • Receive history list function
Message transmission	• Unique ID can be set to each message buffer. • Transmit completion interrupt for each message buffer • Message buffer number 0 to 7 specified as the transmit message buffer can be set for automatic block transfer. Message transmission interval is programmable (automatic block transmission function (hereafter referred to as “ABT”)). • Transmission history list function
Remote frame processing	Remote frame processing by transmit message buffer
Time stamp function	• The time stamp function can be set for a message reception when a 16-bit timer is used in combination. • Time stamp capture trigger can be selected (SOF or EOF in a CAN message frame can be detected.). • The time stamp function can be set for a transmit message.
Diagnostic function	• Readable error counters • “Valid protocol operation flag” for verification of bus connections • Receive-only mode • Single-shot mode • CAN protocol error type decoding • Self-test mode
Release from bus-off state	• Forced release from bus-off (by ignoring timing constraint) possible by software. • No automatic release from bus-off (software must re-enable).
Power save mode	• CAN Sleep mode (can be woken up by CAN bus) • CAN Stop mode (cannot be woken up by CAN bus)

20.1.2 Configuration

The CAN Controller is composed of the following four blocks.

- NPB interface
  This functional block provides an NPB (Peripheral I/O Bus) interface and means of transmitting and receiving signals between the CAN module and the host CPU.
- MCM (Message Control Module)
  This functional block controls access to the CAN protocol layer and to the CAN RAM within the CAN module.
- CAN protocol layer
  This functional block is involved in the operation of the CAN protocol and its related settings.
- CAN RAM
  This is the CAN memory functional block, which is used to store message IDs, message data, etc.



Note: The CAN input clock can be chosen from
- the main oscillator clock f_{XC}
- the peripheral clock f_{XP1}
Refer to chapter “Clock Generator” for f_{CAN} selection control.

Figure 20-1 Block diagram of CAN module

20.2 CAN Protocol

CAN (Controller Area Network) is a high-speed multiplex communication protocol for real-time communication in automotive applications (class C). CAN is prescribed by ISO 11898. For details, refer to the ISO 11898 specifications.

The CAN specification is generally divided into two layers: a physical layer and a data link layer. In turn, the data link layer includes logical link and medium access control. The composition of these layers is illustrated below.

Higher	Data link layer[Note]	• Logical link control (LLC)	• Acceptance filtering • Overload report • Recovery management
		• Medium access control (MAC)	• Data capsuled/not capsuled • Frame coding (stuffing/no stuffing) • Medium access management • Error detection • Error report • Acknowledgement • Seriated/not seriated
Lower	Physical layer		Prescription of signal level and bit description

Figure 20-2 Composition of layers

Note CAN Controller specification

20.2.1 Frame format

(1) Standard format frame

- The standard format frame uses 11-bit identifiers, which means that it can handle up to 2,048 messages.

(2) Extended format frame

- The extended format frame uses 29-bit (11 bits + 18 bits) identifiers, which increases the number of messages that can be handled to 2,048 × 2^{18} messages.
- An extended format frame is set when “recessive level” (CMOS level of “1”) is set for both the SRR and IDE bits in the arbitration field.

20.2.2 Frame types

The following four types of frames are used in the CAN protocol.

Table 20-2 Frame types

Frame Type	Description
Data frame	Frame used to transmit data
Remote frame	Frame used to request a data frame
Error frame	Frame used to report error detection
Overload frame	Frame used to delay the next data frame or remote frame

(1) Bus value

The bus values are divided into dominant and recessive.

- Dominant level is indicated by logical 0.
- Recessive level is indicated by logical 1.
- When a dominant level and a recessive level are transmitted simultaneously, the bus value becomes dominant level.

20.2.3 Data frame and remote frame

(1) Data frame

A data frame is composed of seven fields.

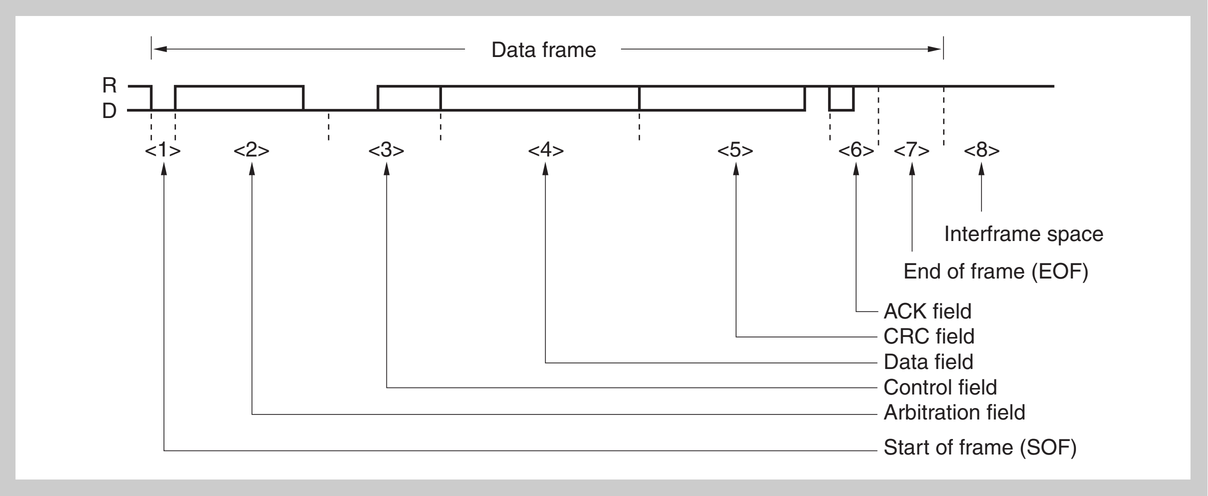


Figure 20-3 Data frame

Note D: Dominant = 0
R: Recessive = 1

(2) Remote frame

A remote frame is composed of six fields.

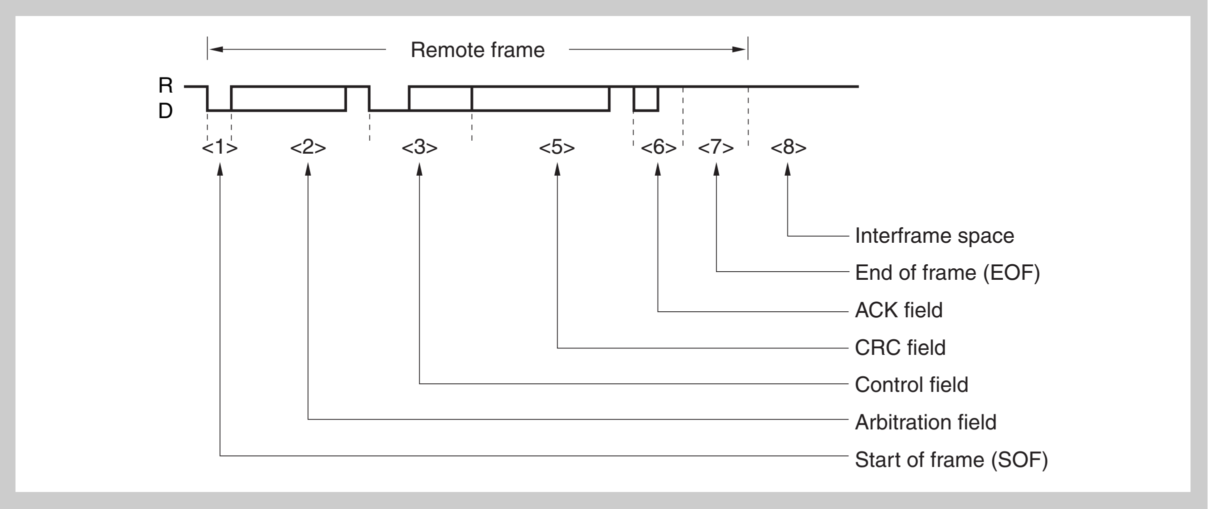


Figure 20-4 Remote frame

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.

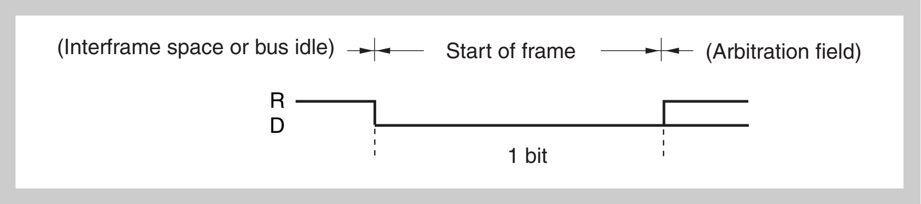


Figure 20-5 Start of frame (SOF)

Note D: Dominant = 0
R: Recessive = 1

- If dominant level is detected in the bus idle state, a hard-synchronization is performed (the current TQ is assigned to be the SYNC segment).
- If dominant level is sampled at the sample point following such a hard-synchronization, the bit is assigned to be a SOF. If recessive level is detected, the protocol layer returns to the bus idle state and regards the preceding dominant pulse as a disturbance only. No error frame is generated in such case.

(b) Arbitration field

The arbitration field is used to set the priority, data frame/remote frame, and frame format.

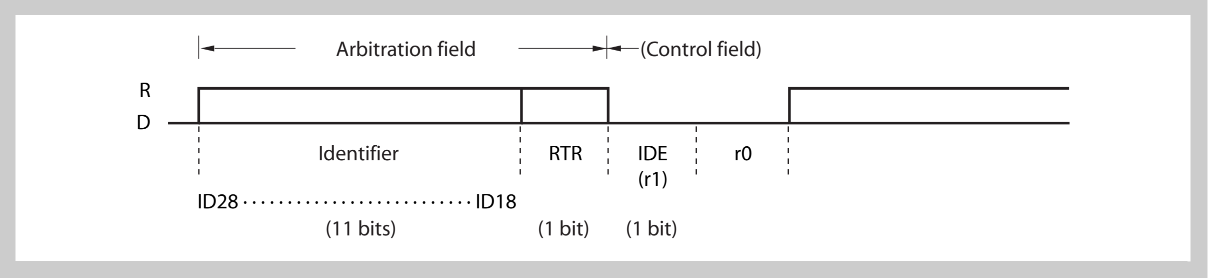


Figure 20-6 Arbitration field (in standard format mode)

Caution 1. ID28 to ID18 are identifiers.

2. An identifier is transmitted MSB first.

Note D: Dominant = 0
R: Recessive = 1

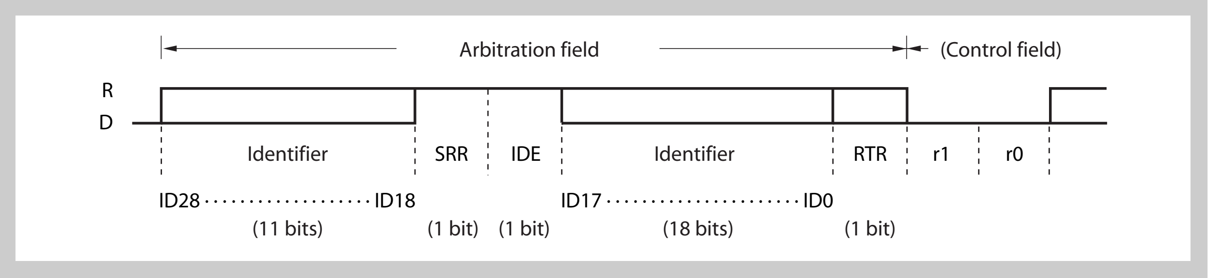


Figure 20-7 Arbitration field (in extended format mode)

Caution 1. ID28 to ID18 are identifiers.

2. An identifier is transmitted MSB first.

Note D: Dominant = 0
R: Recessive = 1

Table 20-3 RTR frame settings

Frame type	RTR bit
Data frame	0 (D)
Remote frame	1 (R)

Table 20-4 Frame format setting (IDE bit) and number of identifier (ID) bits

Frame format	SRR bit	IDE bit	Number of bits
Standard format mode	None	0 (D)	11 bits
Extended format mode	1 (R)	1 (R)	29 bits

(c) Control field

The control field sets “DLC” as the number of data bytes in the data field (DLC = 0 to 8).

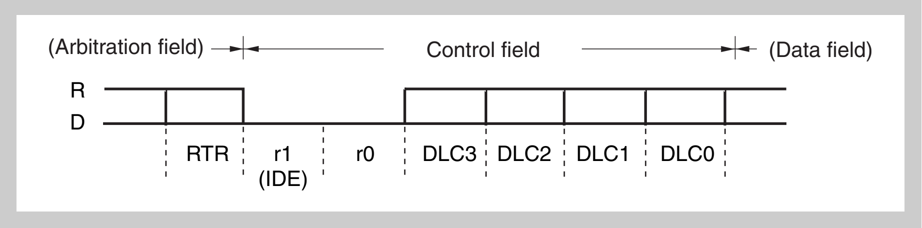


Figure 20-8 Control field

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 20-5 Data length setting

Data length code				Data byte count
DLC3	DLC2	DLC1	DLC0	
0	0	0	0	0 bytes
0	0	0	1	1 byte
0	0	1	0	2 bytes
0	0	1	1	3 bytes
0	1	0	0	4 bytes
0	1	0	1	5 bytes
0	1	1	0	6 bytes
0	1	1	1	7 bytes
1	0	0	0	8 bytes
Other than above				8 bytes regardless of the value of DLC3 to DLC0

Caution In the remote frame, there is no data field even if the data length code is not 0000_B.

(d) Data field

The data field contains the amount of data (byte units) set by the control field. Up to 8 units of data can be set.

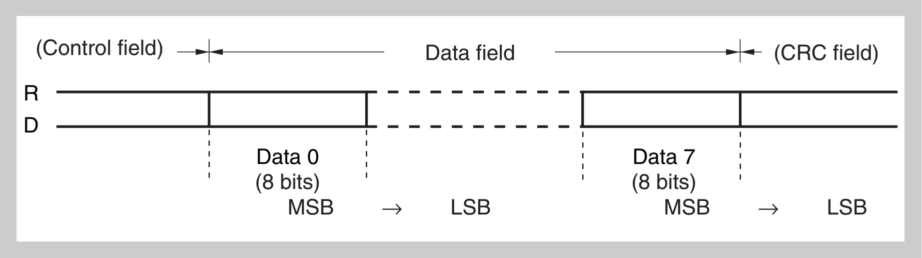


Figure 20-9 Data field

Note D: Dominant = 0
R: Recessive = 1

(e) CRC field

The CRC field is a 16-bit field that is used to check for errors in transmit data.

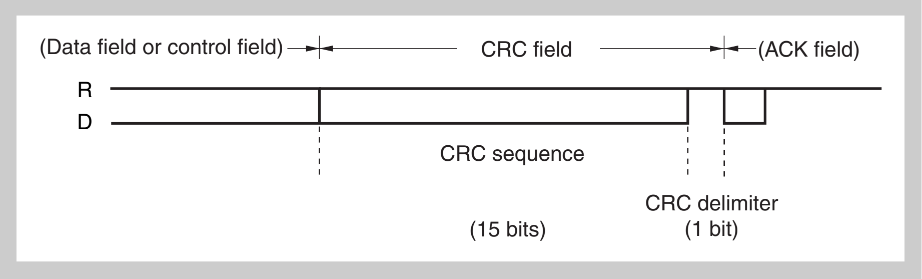


Figure 20-10 CRC field

Note D: Dominant = 0
R: Recessive = 1

- The polynomial P(X) used to generate the 15-bit CRC sequence is expressed as follows.

  $P(X) = X^{15} + X^{14} + X^{10} + X^{8} + X^{7} + X^{4} + X^{3} + 1$

- Transmitting node: Transmits the CRC sequence calculated from the data (before bit stuffing) in the start of frame, arbitration field, control field, and data field.
- Receiving node: Compares the CRC sequence calculated using data bits that exclude the stuffing bits in the receive data with the CRC sequence in the CRC field. If the two CRC sequences do not match, the node issues an error frame.

(f) ACK field

The ACK field is used to acknowledge normal reception.

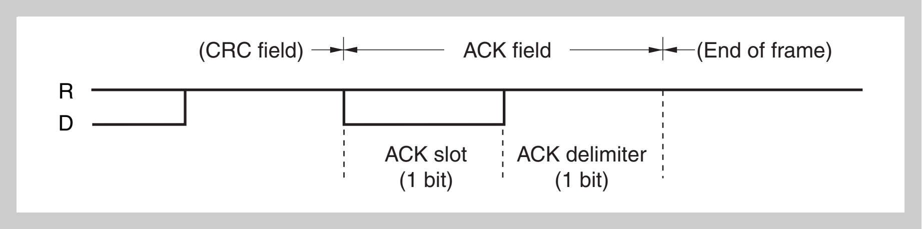


Figure 20-11 ACK field

Note D: Dominant = 0
R: Recessive = 1

- If no CRC error is detected, the receiving node sets the ACK slot to the dominant level.
- The transmitting node outputs two recessive-level bits.

(g) End of frame (EOF)

The end of frame field indicates the end of data frame/remote frame.

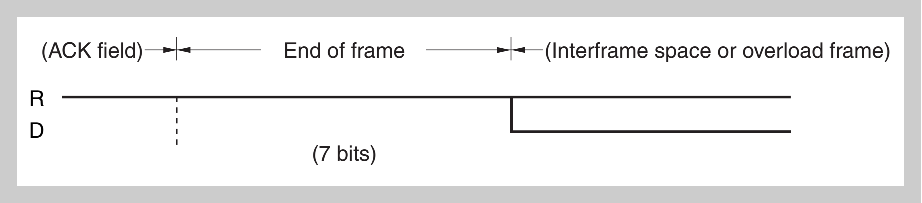


Figure 20-12 End of frame (EOF)

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.

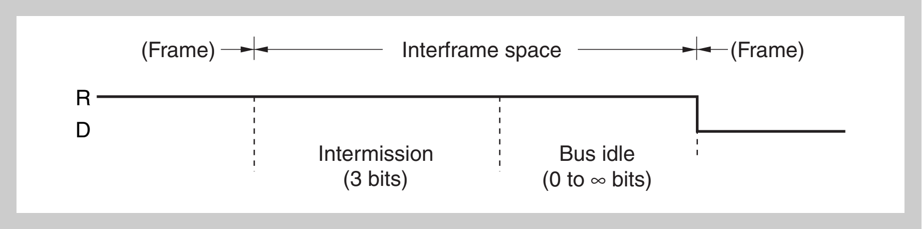


Figure 20-13 Interframe space (error active node)

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.

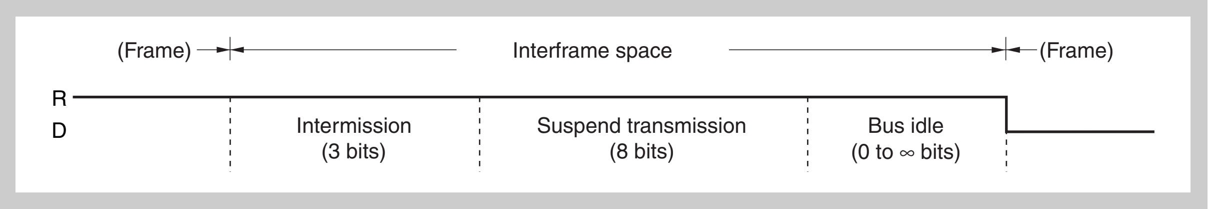


Figure 20-14 Interframe space (error passive node)

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 20-6 Operation in error status

Error status	Operation
Error active	A node in this status can transmit immediately after a 3-bit intermission.
Error passive	A node in this status can transmit 8 bits after the intermission.

20.2.4 Error frame

An error frame is output by a node that has detected an error.

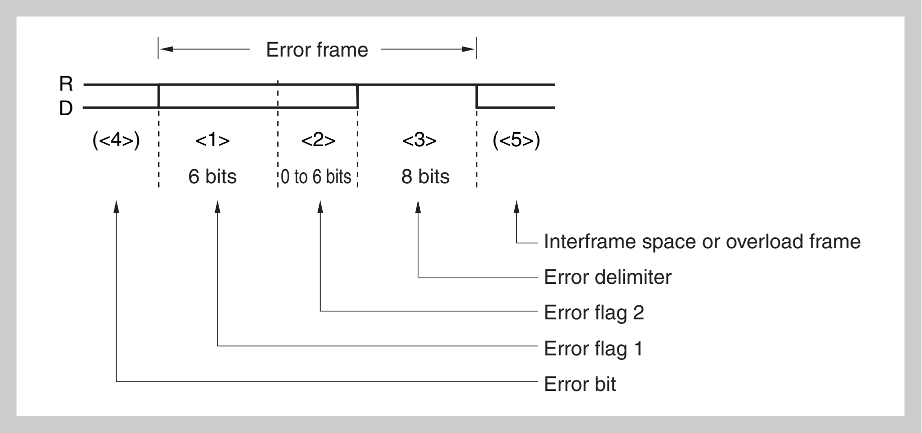


Figure 20-15 Error frame

Note D: Dominant = 0
R: Recessive = 1

Table 20-7 Definition of error frame fields

No.	Name	Bit count	Definition
<1>	Error flag 1	6	Error active node: Outputs 6 dominant-level bits consecutively. Error passive node: Outputs 6 recessive-level bits consecutively. If another node outputs a dominant level while one node is outputting a passive error flag, the passive error flag is not cleared until the same level is detected 6 bits in a row.
<2>	Error flag 2	0 to 6	Nodes receiving error flag 1 detect bit stuff errors and issues this error flag.
<3>	Error delimiter	8	Outputs 8 recessive-level bits consecutively. If a dominant level is detected at the 8th bit, an overload frame is transmitted from the next bit.
<4>	Error bit	–	The bit at which the error was detected. The error flag is output from the bit next to the error bit. In the case of a CRC error, this bit is output following the ACK delimiter.
<5>	Interframe space/ overload frame	–	An interframe space or overload frame starts from here.

20.2.5 Overload frame

An overload frame is transmitted under the following conditions.

- When the receiving node has not completed the reception operation
- If a dominant level is detected at the first two bits during intermission
- If a dominant level is detected at the last bit (7th bit) of the end of frame or at the last bit (8th bit) of the error delimiter/overload delimiter

Note The CAN is internally fast enough to process all received frames not generating overload frames.

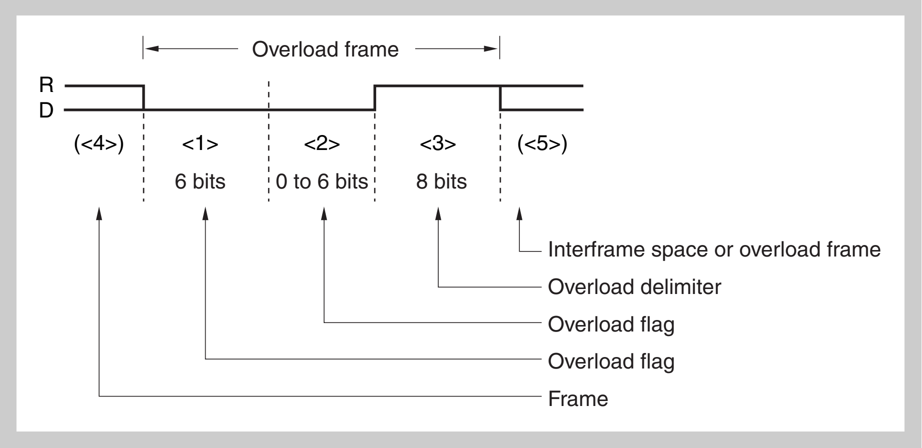


Figure 20-16 Overload frame

Note D: Dominant = 0
R: Recessive = 1

Table 20-8 Definition of overload frame fields

No	Name	Bit count	Definition
<1>	Overload flag	6	Outputs 6 dominant-level bits consecutively.
<2>	Overload flag from other node	0 to 6	The node that received an overload flag in the interframe space outputs an overload flag.
<3>	Overload delimiter	8	Outputs 8 recessive-level bits consecutively. If a dominant level is detected at the 8th bit, an overload frame is transmitted from the next bit.
<4>	Frame	–	Output following an end of frame, error delimiter, or overload delimiter.
<5>	Interframe space/overload frame	–	An interframe space or overload frame starts from here.

20.3 Functions

20.3.1 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 20-9 Determining bus priority

Level match	Continuous transmission
Level mismatch	Stops transmission at the bit where mismatch is detected and starts reception at the following bit

(3) Priority of data frame and remote frame

- When a data frame and a remote frame are on the bus, the data frame has priority because its RTR bit, the last bit in the arbitration field, carries a dominant level.

Note If the extended-format data frame and the standard-format remote frame conflict on the bus (if ID28 to ID18 of both of them are the same), the standard-format remote frame takes priority.

20.3.2 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 20-10 Bit stuffing

Transmission	During the transmission of a data frame or remote frame, when the same level continues for 5 bits in the data between the start of frame and the ACK field, 1 inverted-level bit of data is inserted before the following bit.
Reception	During the reception of a data frame or remote frame, when the same level continues for 5 bits in the data between the start of frame and the ACK field, reception is continued after deleting the next bit.

20.3.3 Multi masters

As the bus priority (a node acquiring transmit functions) is determined by the identifier, any node can be the bus master.

20.3.4 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.

20.3.5 CAN sleep mode/CAN stop mode function

The CAN sleep mode/CAN stop mode function puts the CAN Controller in waiting mode to achieve low power consumption.

The controller is woken up from the CAN sleep mode by bus operation but it is not woken up from the CAN stop mode by bus operation (the CAN stop mode is controlled by CPU access).

20.3.6 Error control function

(1) Error types

Table 20-11 Error types

Type	Description of error		Detection state	
	Detection method	Detection condition	Transmission/ reception	Field/frame
Bit error	Comparison of the output level and level on the bus (except stuff bit)	Mismatch of levels	Transmitting/ receiving node	Bit that is outputting data on the bus at the start of frame to end of frame, error frame and overload frame.
Stuff error	Check of the receive data at the stuff bit	6 consecutive bits of the same output level	Receiving node	Start of frame to CRC sequence
CRC error	Comparison of the CRC sequence generated from the receive data and the received CRC sequence	Mismatch of CRC	Receiving node	CRC field
Form error	Field/frame check of the fixed format	Detection of fixed format violation	Receiving node	CRC delimiter ACK field End of frame Error frame Overload frame
ACK error	Check of the ACK slot by the transmitting node	Detection of recessive level in ACK slot	Transmitting node	ACK slot

(2) Output timing of error frame

Table 20-12 Output timing of error frame

Type	Output timing
Bit error, stuff error, form error, ACK error	Error frame output is started at the timing of the bit following the detected error.
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 20-13*.

The present error state is indicated by the CAN module information register (CnINFO).

When each error counter value becomes equal to or greater than the error warning level (96), the TECS0 or RECS0 bit of the CnINFO register is set to 1. In this case, the bus state must be tested because it is considered that the bus has a serious fault. An error counter value of 128 or more indicates an error passive state and the TECS1 or RECS1 bit of the CnINFO register is set to 1.

- If the value of the transmission error counter is greater than or equal to 256 (actually, the transmission error counter does not indicate a value greater than or equal to 256), the bus-off state is reached and the BOFF bit of the CnINFO register is set to 1.
- If only one node is active on the bus at startup (i.e., a particular case such as when the bus is connected only to the local station), ACK is not returned even if data is transmitted. Consequently, re-transmission of the error frame and data is repeated. In the error passive state, however, the transmission error counter is not incremented and the bus-off state is not reached.

Table 20-13 Types of error states

Type	Operation	Value of error counter	Indication of CnINFO register	Operation specific to error state
Error active	Transmission	0 to 95	TECS1, TECS0 = 00	Outputs an active error flag (6 consecutive dominant-level bits) on detection of the error.
	Reception	0 to 95	RECS1, RECS0 = 00	
	Transmission	96 to 127	TECS1, TECS0 = 01	
	Reception	96 to 127	RECS1, RECS0 = 01	
Error passive	Transmission	128 to 255	TECS1, TECS0 = 11	Outputs a passive error flag (6 consecutive recessive-level bits) on detection of the error. Transmits 8 recessive-level bits, in between transmissions, following an intermission (suspend transmission).
	Reception	128 or more	RECS1, RECS0 = 11	
Bus-off	Transmission	256 or more (not indicated)[Note]	BOFF = 1, TECS1, TECS0 = 11	Communication is not possible. Messages are not stored when receiving frames, however, the following operations of <1>, <2>, and <3> are done. <1> TSOUT toggles. <2> REC is incremented/decremented. <3> VALID bit is set. If the CAN module is entered to the initialization mode and then transition request to any operation mode is made, and when 11 consecutive recessive-level bits are detected 128 times, the error counter is reset to 0 and the error active state can be restored.

Note The value of the transmission error counter (TEC) is invalid when the BOFF bit is set to 1. If an error that increments the value of the transmission error counter by +8 while the counter value is in a range of 248 to 255, the counter is not incremented and the bus-off state is assumed.

(b) Error counter

The error counter counts up when an error has occurred, and counts down upon successful transmission and reception. The error counter is updated immediately after error detection.

Table 20-14 Error counter

State	Transmission error counter (TEC7 to TEC0 bits)	Reception error counter (REC6 to REC0 bits)
Receiving node detects an error (except bit error in the active error flag or overload flag).	No change	+1 (when REPS = 0)
Receiving node detects dominant level following error flag of error frame.	No change	+8 (when REPS = 0)
Transmitting node transmits an error flag. [As exceptions, the error counter does not change in the following cases.] <1> ACK error is detected in error passive state and dominant level is not detected while the passive error flag is being output. <2> A stuff error is detected in an arbitration field that transmitted a recessive level as a stuff bit, but a dominant level is detected.	+8	No change
Bit error detection while active error flag or overload flag is being output (error-active transmitting node)	+8	No change
Bit error detection while active error flag or overload flag is being output (error-active receiving node)	No change	+8 (REPS bit = 0)
When the node detects 14 consecutive dominant-level bits from the beginning of the active error flag or overload flag, and then subsequently detects 8 consecutive dominant-level bits. When the node detects 8 consecutive dominant levels after a passive error flag	+8 (transmitting)	+8 (during reception, when REPS = 0)
When the transmitting node has completed transmission without error (±0 if error counter = 0)	–1	No change
When the receiving node has completed reception without error	No change	• –1 (1 ≤ REC6 to REC0 ≤ 127, when REPS = 0) • ±0 (REC6 to REC0 = 0, when REPS = 0) • Value of 119 to 127 is set (when REPS = 1)

(c) Occurrence of bit error in intermission

An overload frame is generated.

Caution If an error occurs, it is controlled according to the contents of the transmission error counter and reception error counter before the error occurred. The value of the error counter is incremented after the error flag has been output.

(5) Recovery from bus-off state

When the CAN module is in the bus-off state, the CAN module permanently sets its output signals (CTXDn) to recessive level.

The CAN module recovers from the bus-off state in the following bus-off recovery sequence.

1. **A request to enter the CAN initialization mode**
2. **A request to enter a CAN operation mode**

   (a)Recovery operation through normal recovery sequence
   (b)Forced recovery operation that skips recovery sequence

(a) Recovery from bus-off state through normal recovery sequence

The CAN module first issues a request to enter the initialization mode (refer to timing <1> in *Figure 20-17 on page 694*). 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 20-17 on page 694*). 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 20-17 on page 694*), the CAN module can enter the operation mode it has requested. Until the CAN module enters this operation mode, it stays in the initialization mode. Completion to be requested operation mode can be confirmed by reading the OPMODE bits of the CnCTRL register.

During the bus-off period and bus-off recovery sequence, the BOFF bit of the CnINFO register stays set (to 1). In the bus-off recovery sequence, the reception error counter (REC[6:0]) counts the number of times 11 consecutive recessive-level bits have been detected on the bus. Therefore, the recovery state can be checked by reading REC[6:0].

Caution

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.

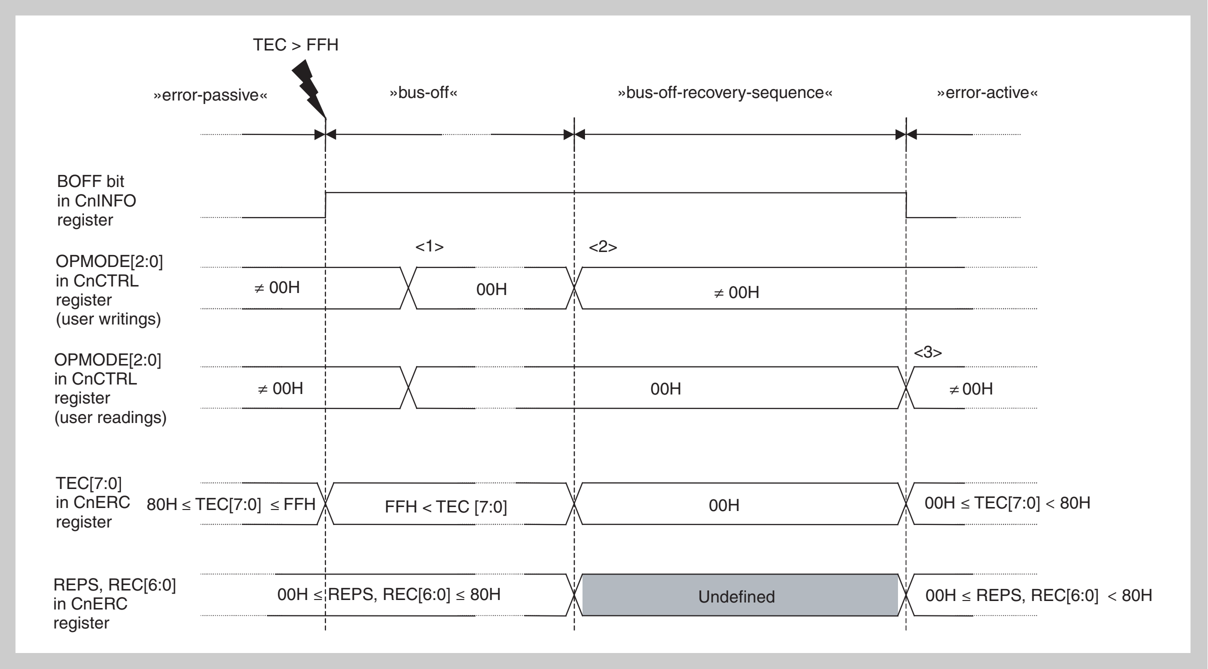

TEC > FFH
»error-passive«
»bus-off«
»bus-off-recovery-sequence«
»error-active«
BOFF bit in CnINFO register
OPMODE[2:0] in CnCTRL register (user writings)
<1>
<2>
≠ 00H
00H
≠ 00H
OPMODE[2:0] in CnCTRL register (user readings)
<3>
≠ 00H
00H
≠ 00H
TEC[7:0] in CnERC register
80H ≤ TEC[7:0] ≤ FFH
FFH < TEC [7:0]
00H
00H ≤ TEC[7:0] < 80H
REPS, REC[6:0] in CnERC register
00H ≤ REPS, REC[6:0] ≤ 80H
Undefined
00H ≤ REPS, REC[6:0] < 80H

Figure 20-17 Recovery from bus-off state through normal recovery sequence

(b) Forced recovery operation that skips bus-off recovery sequence

The CAN module can be forcibly released from the bus-off state, regardless of the bus state, by skipping the bus-off recovery sequence. Here is the procedure.

First, the CAN module requests to enter the initialization mode. For the operation and points to be noted at this time, *"Recovery from bus-off state through normal recovery sequence" on page 693*.

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 20-55 on page 811*.

Caution This function is not defined by the CAN protocol ISO 11898. When using this function, thoroughly evaluate its effect on the network system.

(6) Initializing CAN module error counter register (CnERC) in initialization mode

If it is necessary to initialize the CAN module error counter register (CnERC) and CAN module information register (CnINFO) for debugging or evaluating a program, they can be initialized to the default value by setting the CCERC bit of the CnCTRL register in the initialization mode. When initialization has been completed, the CCERC bit is automatically cleared to 0.

Caution

1. This function is enabled only in the initialization mode. Even if the CCERC bit is set to 1 in a CAN operation mode, the CnERC and CnINFO registers are not initialized.
2. The CCERC bit can be set at the same time as the request to enter a CAN operation mode.

20.3.7 Baud rate control function

(1) Prescaler

The CAN controller has a prescaler that divides the clock (f_{CAN}) supplied to CAN. This prescaler generates a CAN protocol layer basic system clock (f_{TQ}) derived from the CAN module system clock (f_{CANMOD}), and divided by 1 to 256 (*"CnBRP - CANn module bit rate prescaler register" on page 735*).

(2) Data bit time (8 to 25 time quanta)

One data bit time is defined as shown in *Figure 20-18 on page 696*.

The CAN Controller sets time segment 1, time segment 2, and reSynchronization Jump Width (SJW) of data bit time, as shown in *Figure 20-18*. 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.

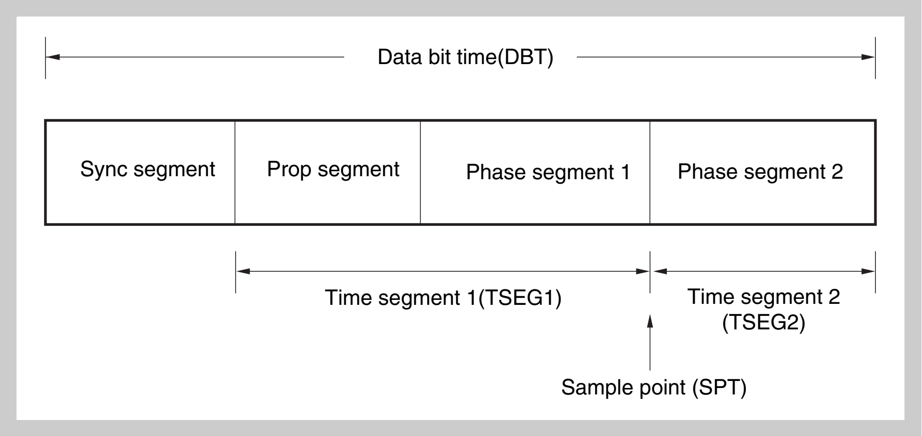


Figure 20-18 Segment setting

Table 20-15 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 of the CAN controller is 0TQ. To conform to the CAN protocol specification, therefore, a length less or equal to phase segment 1 must be set here. This means that the length of time segment 1 minus 1TQ is the settable upper limit of time segment 2.
Resynchronization Jump Width (SJW)	1TQ to 4TQ	The length of time segment 1 minus 1TQ or 4 TQ, whichever is smaller.

Note
1. IPT: Information Processing Time
2. TQ: Time Quanta

Reference: The CAN protocol specification defines the segments constituting the data bit time as shown in *Figure 20-19*.

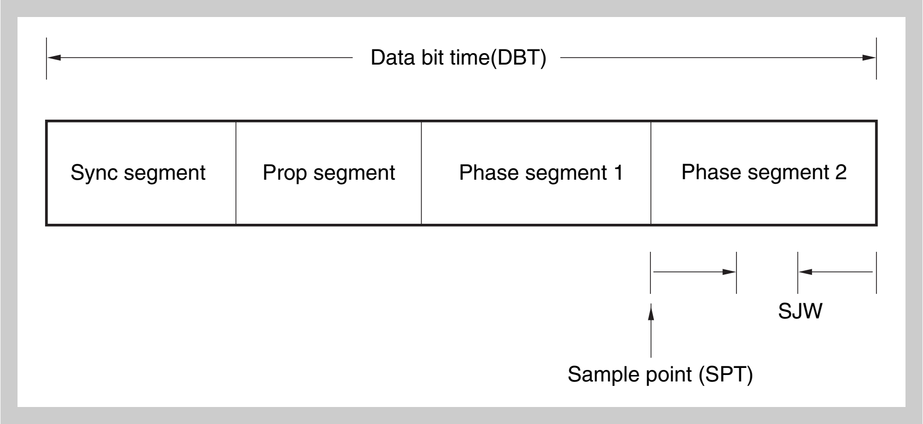


Figure 20-19 Configuration of data bit time defined by CAN specification

Table 20-16 Configuration of data bit time defined by CAN specification

Segment name	Settable range	Notes on setting to conform to CAN specification
Sync segment (Synchronization segment)	1	This segment starts at the edge where the level changes from recessive to dominant when hardware synchronization is established.
Prop segment	Programmable to 1 to 8 or more	This segment absorbs the delay of the output buffer, CAN bus, and input buffer. The length of this segment is set so that ACK is returned before the start of phase segment 1. Time of prop segment ≥ (Delay of output buffer) + 2 × (Delay of CAN bus) + (Delay of input buffer)
Phase segment 1	Programmable to 1 to 8	This segment compensates for an error of data bit time. The longer this segment, the wider the permissible range but the slower the communication speed.
Phase segment 2	Phase segment 1 or IPT, whichever greater	
SJW	Programmable from 1TQ to length of segment 1 or 4TQ, whichever is smaller	This width sets the upper limit of expansion or contraction of the phase segment during resynchronization.

Note IPT: Information Processing Time

(3) Synchronizing data bit

- The receiving node establishes synchronization by a level change on the bus because it does not have a sync signal.
- The transmitting node transmits data in synchronization with the bit timing of the transmitting node.

(a) Hardware synchronization

This synchronization is established when the receiving node detects the start of frame in the interframe space.

- When a falling edge is detected on the bus, that TQ means the sync segment and the next segment is the prop segment. In this case, synchronization is established regardless of SJW.

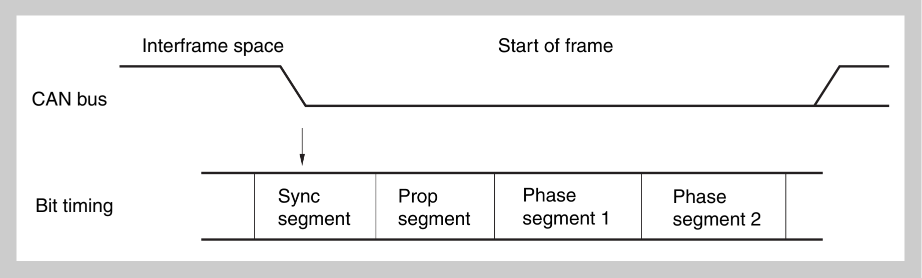


Figure 20-20 Adjusting synchronization of data bit

(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.

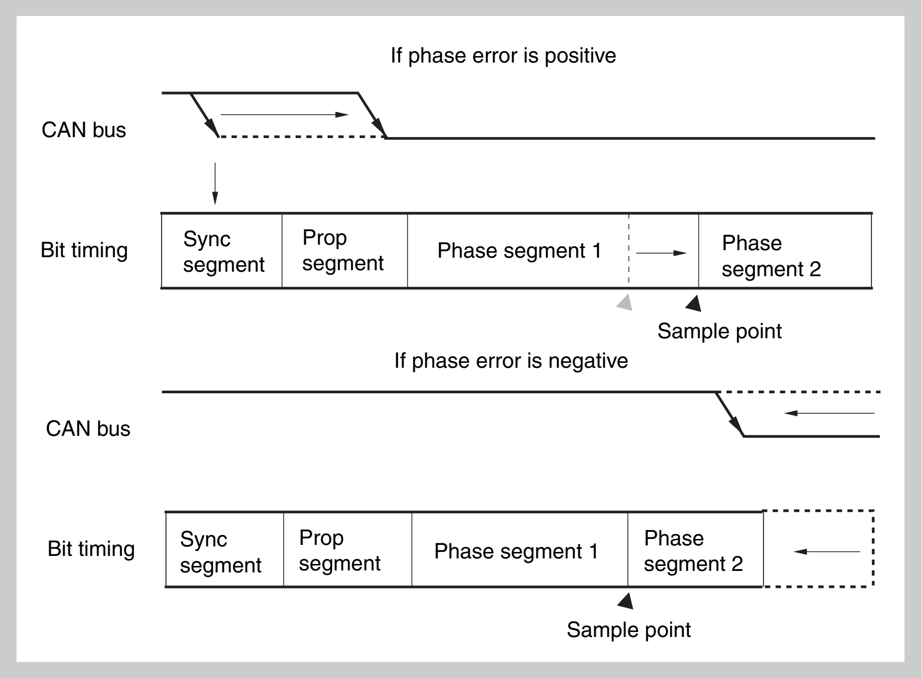


Figure 20-21 Resynchronization

20.4 Connection with Target System

The CAN module has to be connected to the CAN bus using an external transceiver.

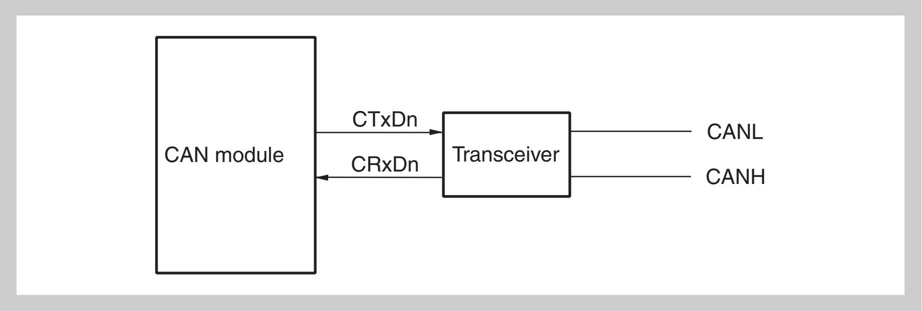


Figure 20-22 Connection to CAN bus

20.5 Internal Registers of CAN Controller

20.5.1 CAN module register and message buffer addresses

In this chapter all register and message buffer addresses are defined as address offsets to different base addresses.

Since all registers are accessed via the programmable peripheral area the bottom address is defined by the BPC register (refer to *"Programmable peripheral I/O area" on page 173* or to *"Programmable peripheral I/O area (PPA)" on page 344*).

The addresses given in the following tables are offsets to the programmable peripheral area base address PBA.

The setting of BPC is fixed to 8FFB$_H$. This setting defines the programmable peripheral area base address

PBA = 03FE C000$_H$

Table 20-17 lists all base addresses used throughout this chapter.

Table 20-17 CAN module base addresses

Base address name	Base address of	Address	Address for BPC =8FFB$_H$
C0RBaseAddr	CAN0 registers	PBA + 000$_H$	03FE C000$_H$
C0MBaseAddr	CAN0 message buffers	PBA + 100$_H$	03FE C100$_H$
C1RBaseAddr	CAN1 registers	PBA + 600$_H$	03FE C600$_H$
C1MBaseAddr	CAN1 message buffers	PBA + 700$_H$	03FE C700$_H$
C2RBaseAddr	CAN2 registers	PBA + C00$_H$	03FE CC00$_H$
C2MBaseAddr	CAN2 message buffers	PBA + D00$_H$	03FE CD00$_H$
C3RBaseAddr	CAN3 registers	PBA + 1200$_H$	03FE D200$_H$
C3MBaseAddr	CAN3 message buffers	PBA + 1300$_H$	03FE D300$_H$
C4RBaseAddr	CAN4 registers	PBA + 1800$_H$	03FE D800$_H$
C4MBaseAddr	CAN4 message buffers	PBA + 1900$_H$	03FE D900$_H$

In the following <CnRBaseAddr> respectively <CnMBaseAddr> are used for the base address names for CAN channel n.

20.5.2 CAN Controller configuration

Table 20-18 List of CAN Controller registers

Item	Register Name
CAN 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)
CAN module registers	CANn module mask 1 register (CnMASK1L, CnMASK1H)
	CANn module mask 2 register (CnMASK2L, CnMASK2H)
	CANn module mask 3 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)
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)

20.5.3 CAN registers overview

(1) CAN0 module registers

The following table lists the address offsets to the CAN0 register base address:

C0RBaseAddr = PBA

Table 20-19 CAN0 global and module registers

Address offset	Register name	Symbol	R/W	Access			After reset
				1-bit	8-bit	16-bit	
000_H	CAN0 global control register	C0GMCTRL	R/W			√	0000_H
002_H	CAN0 global clock selection register	C0GMCS			√		$0F_H$
006_H	CAN0 global automatic block transmission register	C0GMABT				√	0000_H
008_H	CAN0 global automatic block transmission delay register	C0GMABTD			√		00_H
040_H	CAN0 module mask 1 register	C0MASK1L				√	Undefined
042_H		C0MASK1H				√	Undefined
044_H	CAN0 module mask 2 register	C0MASK2L				√	Undefined
046_H		C0MASK2H				√	Undefined
048_H	CAN0 module mask 3 register	C0MASK3L				√	Undefined
$04A_H$		C0MASK3H				√	Undefined
$04C_H$	CAN0 module mask 4 register	C0MASK4L				√	Undefined
$04E_H$		C0MASK4H				√	Undefined
050_H	CAN0 module control register	C0CTRL				√	0000_H
052_H	CAN0 module last error code register	C0LEC			√		00_H
053_H	CAN0 module information register	C0INFO	R		√		00_H
054_H	CAN0 module error counter register	C0ERC				√	0000_H
056_H	CAN0 module interrupt enable register	C0IE	R/W			√	0000_H
058_H	CAN0 module interrupt status register	C0INTS				√	0000_H
$05A_H$	CAN0 module bit-rate prescaler register	C0BRP			√		FF_H
$05C_H$	CAN0 module bit-rate register	C0BTR				√	$370F_H$
$05E_H$	CAN0 module last in-pointer register	C0LIPT	R		√		Undefined
060_H	CAN0 module receive history list register	C0RGPT	R/W			√	$xx02_H$
062_H	CAN0 module last out-pointer register	C0LOPT	R		√		Undefined
064_H	CAN0 module transmit history list register	C0TGPT	R/W			√	$xx02_H$
066_H	CAN0 module time stamp register	C0TS				√	0000_H

The addresses in the following table denote the address offsets to the CAN #n message buffer base address: CnMBaseAddr, with m being the message buffer number.

Example CAN0, message buffer 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,
C0MDATA01m = C0MDATA0100 for m = 0.

Table 20-20 CAN0 message buffer registers

Address offset	Register name	Symbol	R/W	Access			After reset
				1-bit	8-bit	16-bit	
$mx20_H + 0_H$	CAN #n message data byte 01 register m	CnMDATA01m	R/W			√	Undefined
$mx20_H + 0_H$	CAN #n message data byte 0 register m	CnMDATA0m			√		Undefined
$mx20_H + 1_H$	CAN #n message data byte 1 register m	CnMDATA1m			√		Undefined
$mx20_H + 2_H$	CAN #n message data byte 23 register m	CnMDATA23m				√	Undefined
$mx20_H + 2_H$	CAN #n message data byte 2 register m	CnMDATA2m			√		Undefined
$mx20_H + 3_H$	CAN #n message data byte 3 register m	CnMDATA3m			√		Undefined
$mx20_H + 4_H$	CAN #n message data byte 45 register m	CnMDATA45m				√	Undefined
$mx20_H + 4_H$	CAN #n message data byte 4 register m	CnMDATA4m			√		Undefined
$mx20_H + 5_H$	CAN #n message data byte 5 register m	CnMDATA5m			√		Undefined
$mx20_H + 6_H$	CAN #n message data byte 67 register m	CnMDATA67m				√	Undefined
$mx20_H + 6_H$	CAN #n message data byte 6 register m	CnMDATA6m			√		Undefined
$mx20_H + 7_H$	CAN #n message data byte 7 register m	CnMDATA7m			√		Undefined
$mx20_H + 8_H$	CAN #n message data length register m	CnMDLCm			√		$0000\ xxxx_B$
$mx20_H + 9_H$	CAN #n message configuration register m	CnMCONFm			√		Undefined
$mx20_H + A_H$	CAN #n message identifier register m	CnMIDLm				√	Undefined
$mx20_H + C_H$		CnMIDHm				√	Undefined
$mx20_H + E_H$	CAN #n message control register m	CnMCTRLm				√	0x00 0000 $0000\ 0000_B$

(2) CAN1 module registers

The following table lists the address offsets to the CAN1 register base address:

C1RBaseAddr = PBA + 600_H

Table 20-21 CAN1 global and module registers

Address offset	Register name	Symbol	R/W	Access			After reset
				1-bit	8-bit	16-bit	
000_H	CAN1 global control register	C1GMCTRL	R/W			√	0000_H
002_H	CAN1 global clock selection register	C1GMCS			√		$0F_H$
006_H	CAN1 global automatic block transmission register	C1GMABT				√	0000_H
008_H	CAN1 global automatic block transmission delay register	C1GMABTD			√		00_H
040_H	CAN1 module mask 1 register	C1MASK1L				√	Undefined
042_H		C1MASK1H				√	Undefined
044_H	CAN1 module mask 2 register	C1MASK2L				√	Undefined
046_H		C1MASK2H				√	Undefined
048_H	CAN1 module mask 3 register	C1MASK3L				√	Undefined
$04A_H$		C1MASK3H				√	Undefined
$04C_H$	CAN1 module mask 4 register	C1MASK4L				√	Undefined
$04E_H$		C1MASK4H				√	Undefined
050_H	CAN1 module control register	C1CTRL				√	0000_H
052_H	CAN1 module last error code register	C1LEC			√		00_H
053_H	CAN1 module information register	C1INFO	R		√		00_H
054_H	CAN1 module error counter register	C1ERC				√	0000_H
056_H	CAN1 module interrupt enable register	C1IE	R/W			√	0000_H
058_H	CAN1 module interrupt status register	C1INTS				√	0000_H
$05A_H$	CAN1 module bit-rate prescaler register	C1BRP			√		FF_H
$05C_H$	CAN1 module bit-rate register	C1BTR				√	$370F_H$
$05E_H$	CAN1 module last in-pointer register	C1LIPT	R		√		Undefined
060_H	CAN1 module receive history list register	C1RGPT	R/W			√	$xx02_H$
062_H	CAN1 module last out-pointer register	C1LOPT	R		√		Undefined
064_H	CAN1 module transmit history list register	C1TGPT	R/W			√	$xx02_H$
066_H	CAN1 module time stamp register	C1TS				√	0000_H

The addresses in the following table denote the address offsets to the CAN1 message buffer base address:

C1MBaseAddr = PBA + 700_H

Example CAN1, message buffer register m = 23 = 17_H, byte 3 C1MDATA323 has the address 17_H x 20_H + 3_H + C1MBaseAddr

Note The message buffer register number m in the register symbols has 2 digits, for example,
C1MDATA01m = C1MDATA0113 for m = 13.

Table 20-22 CAN1 message buffer registers

Address offset	Register name	Symbol	R/W	Access			After reset
				1-bit	8-bit	16-bit	
mx20_H + 0_H	CAN1 message data byte 01 register m	C1MDATA01m	R/W			√	Undefined
mx20_H + 0_H	CAN1 message data byte 0 register m	C1MDATA0m			√		Undefined
mx20_H + 1_H	CAN1 message data byte 1 register m	C1MDATA1m			√		Undefined
mx20_H + 2_H	CAN1 message data byte 23 register m	C1MDATA23m				√	Undefined
mx20_H + 2_H	CAN1 message data byte 2 register m	C1MDATA2m			√		Undefined
mx20_H + 3_H	CAN1 message data byte 3 register m	C1MDATA3m			√		Undefined
mx20_H + 4_H	CAN1 message data byte 45 register m	C1MDATA45m				√	Undefined
mx20_H + 4_H	CAN1 message data byte 4 register m	C1MDATA4m			√		Undefined
mx20_H + 5_H	CAN1 message data byte 5 register m	C1MDATA5m			√		Undefined
mx20_H + 6_H	CAN1 message data byte 67 register m	C1MDATA67m				√	Undefined
mx20_H + 6_H	CAN1 message data byte 6 register m	C1MDATA6m			√		Undefined
mx20_H + 7_H	CAN1 message data byte 7 register m	C1MDATA7m			√		Undefined
mx20_H + 8_H	CAN1 message data length register m	C1MDLCm			√		0000 $xxxx_B$
mx20_H + 9_H	CAN1 message configuration register m	C1MCONFm			√		Undefined
mx20_H + A_H	CAN1 message identifier register m	C1MIDLm				√	Undefined
mx20_H + C_H		C1MIDHm				√	Undefined
mx20_H + E_H	CAN1 message control register m	C1MCTRLm				√	0x00 0000 0000 0000_B

(3) CAN2 module registers

The following table lists the address offsets to the CAN2 register base address:

C2RBaseAddr = PBA + $C00_H$

Table 20-23 CAN2 global and module registers

Address offset	Register name	Symbol	R/W	Access			After reset
				1-bit	8-bit	16-bit	
000_H	CAN2 global control register	C2GMCTRL	R/W			√	0000_H
002_H	CAN2 global clock selection register	C2GMCS			√		$0F_H$
006_H	CAN2 global automatic block transmission register	C2GMABT				√	0000_H
008_H	CAN2 global automatic block transmission delay register	C2GMABTD			√		00_H
040_H	CAN2 module mask 1 register	C2MASK1L				√	Undefined
042_H		C2MASK1H				√	Undefined
044_H	CAN2 module mask 2 register	C2MASK2L				√	Undefined
046_H		C2MASK2H				√	Undefined
048_H	CAN2 module mask 3 register	C2MASK3L				√	Undefined
$04A_H$		C2MASK3H				√	Undefined
$04C_H$	CAN2 module mask 4 register	C2MASK4L				√	Undefined
$04E_H$		C2MASK4H				√	Undefined
050_H	CAN2 module control register	C2CTRL				√	0000_H
052_H	CAN2 module last error code register	C2LEC			√		00_H
053_H	CAN2 module information register	C2INFO	R		√		00_H
054_H	CAN2 module error counter register	C2ERC				√	0000_H
056_H	CAN2 module interrupt enable register	C2IE	R/W			√	0000_H
058_H	CAN2 module interrupt status register	C2INTS				√	0000_H
$05A_H$	CAN2 module bit-rate prescaler register	C2BRP			√		FF_H
$05C_H$	CAN2 module bit-rate register	C2BTR				√	$370F_H$
$05E_H$	CAN2 module last in-pointer register	C2LIPT	R		√		Undefined
060_H	CAN2 module receive history list register	C2RGPT	R/W			√	$xx02_H$
062_H	CAN2 module last out-pointer register	C2LOPT	R		√		Undefined
064_H	CAN2 module transmit history list register	C2TGPT	R/W			√	$xx02_H$
066_H	CAN2 module time stamp register	C2TS				√	0000_H

The addresses in the following table denote the address offsets to the CAN2 message buffer base address:

C2MBaseAddr = PBA + D00$_H$

Example CAN2, message buffer register m = 30= 1E$_H$, byte 6, C2MDATA630 has the address 1E$_H$ x 20$_H$ + 6$_H$ + C2MBaseAddr

Note The message buffer register number m in the register symbols has 2 digits, for example,
C2MDATA01m = C2MDATA0113 for m = 13.

Table 20-24 CAN2 message buffer registers

Address offset	Register name	Symbol	R/W	Access			After reset
				1-bit	8-bit	16-bit	
$mx20_H + 0_H$	CAN2 message data byte 01 register m	C2MDATA01m	R/W			√	Undefined
$mx20_H + 0_H$	CAN2 message data byte 0 register m	C2MDATA0m			√		Undefined
$mx20_H + 1_H$	CAN2 message data byte 1 register m	C2MDATA1m			√		Undefined
$mx20_H + 2_H$	CAN2 message data byte 23 register m	C2MDATA23m				√	Undefined
$mx20_H + 2_H$	CAN2 message data byte 2 register m	C2MDATA2m			√		Undefined
$mx20_H + 3_H$	CAN2 message data byte 3 register m	C2MDATA3m			√		Undefined
$mx20_H + 4_H$	CAN2 message data byte 45 register m	C2MDATA45m				√	Undefined
$mx20_H + 4_H$	CAN2 message data byte 4 register m	C2MDATA4m			√		Undefined
$mx20_H + 5_H$	CAN2 message data byte 5 register m	C2MDATA5m			√		Undefined
$mx20_H + 6_H$	CAN2 message data byte 67 register m	C2MDATA67m				√	Undefined
$mx20_H + 6_H$	CAN2 message data byte 6 register m	C2MDATA6m			√		Undefined
$mx20_H + 7_H$	CAN2 message data byte 7 register m	C2MDATA7m			√		Undefined
$mx20_H + 8_H$	CAN2 message data length register m	C2MDLCm			√		0000 $xxxx_B$
$mx20_H + 9_H$	CAN2 message configuration register m	C2MCONFm			√		Undefined
$mx20_H + A_H$	CAN2 message identifier register m	C21MIDLm				√	Undefined
$mx20_H + C_H$		C2MIDHm				√	Undefined
$mx20_H + E_H$	CAN2 message control register m	C2MCTRLm				√	0x00 0000 0000 0000_B

(4) CAN3 module registers

The following table lists the address offsets to the CAN3 register base address:

C3RBaseAddr = PBA + 1200_H

Table 20-25 CAN3 global and module registers

Address offset	Register name	Symbol	R/W	Access			After reset
				1-bit	8-bit	16-bit	
000_H	CAN3 global control register	C3GMCTRL	R/W			√	0000_H
002_H	CAN3 global clock selection register	C3GMCS			√		$0F_H$
006_H	CAN3 global automatic block transmission register	C3GMABT				√	0000_H
008_H	CAN3 global automatic block transmission delay register	C3GMABTD			√		00_H
040_H	CAN3 module mask 1 register	C3MASK1L				√	Undefined
042_H		C3MASK1H				√	Undefined
044_H	CAN3 module mask 2 register	C3MASK2L				√	Undefined
046_H		C3MASK2H				√	Undefined
048_H	CAN3 module mask 3 register	C3MASK3L				√	Undefined
$04A_H$		C3MASK3H				√	Undefined
$04C_H$	CAN3 module mask 4 register	C3MASK4L				√	Undefined
$04E_H$		C3MASK4H				√	Undefined
050_H	CAN3 module control register	C3CTRL				√	0000_H
052_H	CAN3 module last error code register	C3LEC			√		00_H
053_H	CAN3 module information register	C3INFO	R		√		00_H
054_H	CAN3 module error counter register	C3ERC				√	0000_H
056_H	CAN3 module interrupt enable register	C3IE	R/W			√	0000_H
058_H	CAN3 module interrupt status register	C3INTS				√	0000_H
$05A_H$	CAN3 module bit-rate prescaler register	C3BRP			√		FF_H
$05C_H$	CAN3 module bit-rate register	C3BTR				√	$370F_H$
$05E_H$	CAN3 module last in-pointer register	C3LIPT	R		√		Undefined
060_H	CAN3 module receive history list register	C3RGPT	R/W			√	$xx02_H$
062_H	CAN3 module last out-pointer register	C3LOPT	R		√		Undefined
064_H	CAN3 module transmit history list register	C3TGPT	R/W			√	$xx02_H$
066_H	CAN3 module time stamp register	C3TS				√	0000_H

The addresses in the following table denote the address offsets to the CAN3 message buffer base address:

C3MBaseAddr = PBA + 1300_H

Example CAN3, message buffer register m = 12= $0C_H$, byte 5, C3MDATA512 has the address $0C_H$ x 20_H + 5_H + C3MBaseAddr

Note The message buffer register number m in the register symbols has 2 digits, for example,
C3MDATA01m = C3MDATA0104 for m = 04.

Table 20-26 CAN3 message buffer registers

Address offset	Register name	Symbol	R/W	Access			After reset
				1-bit	8-bit	16-bit	
mx20_H + 0_H	CAN3 message data byte 01 register m	C3MDATA01m	R/W			√	Undefined
mx20_H + 0_H	CAN3 message data byte 0 register m	C3MDATA0m			√		Undefined
mx20_H + 1_H	CAN3 message data byte 1 register m	C3MDATA1m			√		Undefined
mx20_H + 2_H	CAN3 message data byte 23 register m	C3MDATA23m				√	Undefined
mx20_H + 2_H	CAN3 message data byte 2 register m	C3MDATA2m			√		Undefined
mx20_H + 3_H	CAN3 message data byte 3 register m	C3MDATA3m			√		Undefined
mx20_H + 4_H	CAN3 message data byte 45 register m	C3MDATA45m				√	Undefined
mx20_H + 4_H	CAN3 message data byte 4 register m	C3MDATA4m			√		Undefined
mx20_H + 5_H	CAN3 message data byte 5 register m	C3MDATA5m			√		Undefined
mx20_H + 6_H	CAN3 message data byte 67 register m	C3MDATA67m				√	Undefined
mx20_H + 6_H	CAN3 message data byte 6 register m	C3MDATA6m			√		Undefined
mx20_H + 7_H	CAN3 message data byte 7 register m	C3MDATA7m			√		Undefined
mx20_H + 8_H	CAN3 message data length register m	C3MDLCm			√		0000 $xxxx_B$
mx20_H + 9_H	CAN3 message configuration register m	C3MCONFm			√		Undefined
mx20_H + A_H	CAN3 message identifier register m	C31MIDLm				√	Undefined
mx20_H + C_H		C3MIDHm				√	Undefined
mx20_H + E_H	CAN3 message control register m	C3MCTRLm				√	0x00 0000 0000 0000_B

(5) CAN4 module registers

The following table lists the address offsets to the CAN4 register base address:

C4RBaseAddr = PBA + 1800_H

Table 20-27 CAN4 global and module registers

Address offset	Register name	Symbol	R/W	Access			After reset
				1-bit	8-bit	16-bit	
000_H	CAN4 global control register	C4GMCTRL	R/W			√	0000_H
002_H	CAN4 global clock selection register	C4GMCS			√		$0F_H$
006_H	CAN4 global automatic block transmission register	C4GMABT				√	0000_H
008_H	CAN4 global automatic block transmission delay register	C4GMABTD			√		00_H
040_H	CAN4 module mask 1 register	C4MASK1L				√	Undefined
042_H		C4MASK1H				√	Undefined
044_H	CAN4 module mask 2 register	C4MASK2L				√	Undefined
046_H		C4MASK2H				√	Undefined
048_H	CAN4 module mask 3 register	C4MASK3L				√	Undefined
$04A_H$		C4MASK3H				√	Undefined
$04C_H$	CAN4 module mask 4 register	C4MASK4L				√	Undefined
$04E_H$		C4MASK4H				√	Undefined
050_H	CAN4 module control register	C4CTRL				√	0000_H
052_H	CAN4 module last error code register	C4LEC			√		00_H
053_H	CAN4 module information register	C4INFO	R		√		00_H
054_H	CAN4 module error counter register	C4ERC				√	0000_H
056_H	CAN4 module interrupt enable register	C4IE	R/W			√	0000_H
058_H	CAN4 module interrupt status register	C4INTS				√	0000_H
$05A_H$	CAN4 module bit-rate prescaler register	C4BRP			√		FF_H
$05C_H$	CAN4 module bit-rate register	C4BTR				√	$370F_H$
$05E_H$	CAN4 module last in-pointer register	C4LIPT	R		√		Undefined
060_H	CAN4 module receive history list register	C4RGPT	R/W			√	$xx02_H$
062_H	CAN4 module last out-pointer register	C4LOPT	R		√		Undefined
064_H	CAN4 module transmit history list register	C4TGPT	R/W			√	$xx02_H$
066_H	CAN4 module time stamp register	C4TS				√	0000_H

The addresses in the following table denote the address offsets to the CAN4 message buffer base address:

$C4MBaseAddr = PBA + 1900_H$

Example CAN4, message buffer register m = 12= $0C_H$, byte 5, C4MDATA512 has the address $0C_H$ x 20_H + 5_H + C4MBaseAddr

Note The message buffer register number m in the register symbols has 2 digits, for example,
C4MDATA01m = C4MDATA0104 for m = 04.

Table 20-28 CAN4 message buffer registers

Address offset	Register name	Symbol	R/W	Access			After reset
				1-bit	8-bit	16-bit	
$mx20_H + 0_H$	CAN4 message data byte 01 register m	C4MDATA01m	R/W			√	Undefined
$mx20_H + 0_H$	CAN4 message data byte 0 register m	C4MDATA0m			√		Undefined
$mx20_H + 1_H$	CAN4 message data byte 1 register m	C4MDATA1m			√		Undefined
$mx20_H + 2_H$	CAN4 message data byte 23 register m	C4MDATA23m				√	Undefined
$mx20_H + 2_H$	CAN4 message data byte 2 register m	C4MDATA2m			√		Undefined
$mx20_H + 3_H$	CAN4 message data byte 3 register m	C4MDATA3m			√		Undefined
$mx20_H + 4_H$	CAN4 message data byte 45 register m	C4MDATA45m				√	Undefined
$mx20_H + 4_H$	CAN4 message data byte 4 register m	C4MDATA4m			√		Undefined
$mx20_H + 5_H$	CAN4 message data byte 5 register m	C4MDATA5m			√		Undefined
$mx20_H + 6_H$	CAN4 message data byte 67 register m	C4MDATA67m				√	Undefined
$mx20_H + 6_H$	CAN4 message data byte 6 register m	C4MDATA6m			√		Undefined
$mx20_H + 7_H$	CAN4 message data byte 7 register m	C4MDATA7m			√		Undefined
$mx20_H + 8_H$	CAN4 message data length register m	C4MDLCm			√		$0000\ xxxx_B$
$mx20_H + 9_H$	CAN4 message configuration register m	C4MCONFm			√		Undefined
$mx20_H + A_H$	CAN4 message identifier register m	C41MIDLm				√	Undefined
$mx20_H + C_H$		C4MIDHm				√	Undefined
$mx20_H + E_H$	CAN4 message control register m	C4MCTRLm				√	0x00 0000 $0000\ 0000_B$

20.5.4 Register bit configuration

Table 20-29 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 20-30 CAN module register bit configuration (1/2)

Address offset[a]	Symbol	Bit 7/15	Bit 6/14	Bit 5/13	Bit 4/12	Bit 3/11	Bit 2/10	Bit 1/9	Bit 0/8
40_H	CnMASK1L	CMID7 to CMID0							
41_H		CMID15 to CMID8							
42_H	CnMASK1H	CMID23 to CMID16							
43_H		0	0	0	CMID28 to CMID24				
44_H	CnMASK2L	CMID7 to CMID0							
45_H		CMID15 to CMID8							
46_H	CnMASK2H	CMID23 to CMID16							
47_H		0	0	0	CMID28 to CMID24				
48_H	CnMASK3L	CMID7 to CMID0							
49_H		CMID15 to CMID8							
$4A_H$	CnMASK3H	CMID23 to CMID16							
$4B_H$		0	0	0	CMID28 to CMID24				
$4C_H$	CnMASK4L	CMID7 to CMID0							
$4D_H$		CMID15 to CMID8							
$4E_H$	CnMASK4H	CMID23 to CMID16							
$4F_H$		0	0	0	CMID28 to CMID24				
50_H	CnCTRL (W)	0	Clear AL	Clear VALID	Clear PSMODE1	Clear PSMODE0	Clear OPMODE2	Clear OPMODE1	Clear OPMODE0
51_H		Set CCERC	Set AL	0	Set PSMODE1	Set PSMODE0	Set OPMODE2	Set OPMODE1	Set OPMODE0
50_H	CnCTRL (R)	CCERC	AL	VALID	PS MODE1	PS MODE0	OP MODE2	OP MODE1	OP MODE0
51_H		0	0	0	0	0	0	RSTAT	TSTAT

Table 20-30 CAN module register bit configuration (2/2)

Address offset[a]	Symbol	Bit 7/15	Bit 6/14	Bit 5/13	Bit 4/12	Bit 3/11	Bit 2/10	Bit 1/9	Bit 0/8
52_H	CnLEC (W)	0	0	0	0	0	0	0	0
52_H	CnLEC (R)	0	0	0	0	0	LEC2	LEC1	LEC0
53_H	CnINFO	0	0	0	BOFF	TECS1	TECS0	RECS1	RECS0
54_H	CnERC	TEC7 to TEC0							
55_H		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 20-31 Message buffer register bit configuration

Address offset[a]	Symbol	Bit 7/15	Bit 6/14	Bit 5/13	Bit 4/12	Bit 3/11	Bit 2/10	Bit 1/9	Bit 0/8
0_H	CnMDATA01m	Message data (byte 0)							
1_H		Message data (byte 1)							
0_H	CnMDATA0m	Message data (byte 0)							
1_H	CnMDATA1m	Message data (byte 1)							
2_H	CnMDATA23m	Message data (byte 2)							
3_H		Message data (byte 3)							
2_H	CnMDATA2m	Message data (byte 2)							
3_H	CnMDATA3m	Message data (byte 3)							
4H	CnMDATA45m	Message data (byte 4)							
5_H		Message data (byte 5)							
4_H	CnMDATA4m	Message data (byte 4)							
5_H	CnMDATA5m	Message data (byte 5)							
6_H	CnMDATA67m	Message data (byte 6)							
7_H		Message data (byte 7)							
6_H	CnMDATA6m	Message data (byte 6)							
7_H	CnMDATA7m	Message data (byte 7)							
8_H	CnMDLCm	0				MDLC3	MDLC2	MDLC1	MDLC0
9_H	CnMCONFm	OWS	RTR	MT2	MT1	MT0	0	0	MA0
A_H	CnMIDLm	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
B_H		ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8
C_H	CnMIDHm	ID23	ID22	ID21	ID20	ID19	ID18	ID17	ID16
D_H		IDE	0	0	ID28	ID27	ID26	ID25	ID24
E_H	CnMCTRLm (W)	0	0	0	Clear MOW	Clear IE	Clear DN	Clear TRQ	Clear RDY
F_H		0	0	0	0	Set IE	0	Set TRQ	Set RDY
E_H	CnMCTRLm (R)	0	0	0	MOW	IE	DN	TRQ	RDY
F_H		0	0	MUC	0	0	0	0	0

a) Base address: <CnMBaseAddr>

Note For calculation of the complete message buffer register addresses refer to *"CAN registers overview" on page 702*.

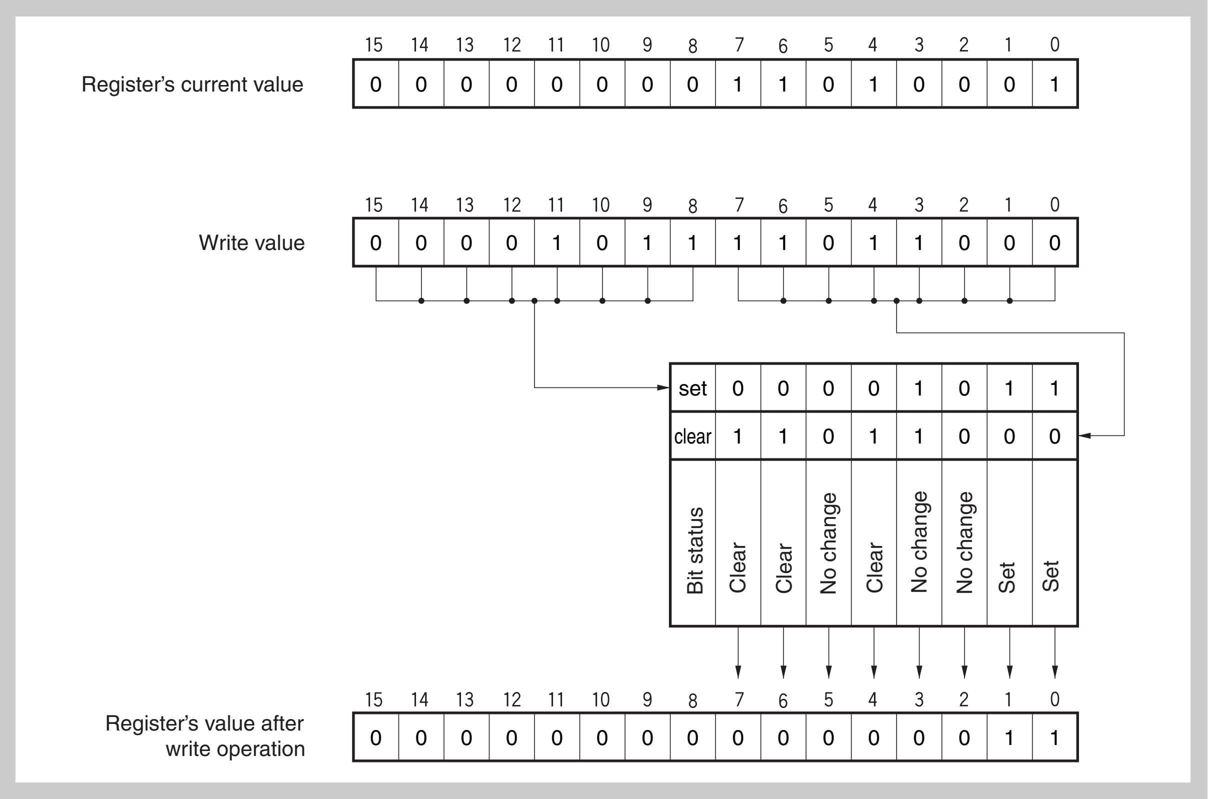
20.6 Bit Set/Clear Function

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 20-23* below to set or clear the lower 8 bits in these registers.

Setting or clearing of lower 8 bits in the above registers is performed in combination with the higher 8 bits (refer to the bit status after set/clear operation is specified in *Figure 20-26*). *Figure 20-23* shows how the values of set bits or clear bits relate to set/clear/no change operations in the corresponding register.



	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Register’s current value	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	1
Write value	0	0	0	0	1	0	1	1	1	1	0	1	1	0	0	0

set	0	0	0	0	1	0	1	1
clear	1	1	0	1	1	0	0	0
Bit status	Clear	Clear	No change	Clear	No change	No change	Set	Set

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Register’s value after write operation	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Figure 20-23 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

20.7 Control Registers

(1) CnGMCTRL - CANn global control register

The CnGMCTRL register is used to control the operation of the CAN module.

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

Address <CnRBaseAddr> + 000$_H$

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

(a) CnGMCTRL read

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

MBON	Bit enabling access to message buffer register, transmit/receive history registers
0	Write access and read access to the message buffer register and the transmit/receive history list registers is disabled.
1	Write access and read access to the message buffer register and the transmit/receive history list registers is enabled.

Caution

1. While the MBON bit is cleared (to 0), software access to the message buffers (CnMDATA0m, CnMDATA1m, CnMDATA01m, CnMDATA2m, CnMDATA3m, CnMDATA23m, CnMDATA4m, CnMDATA5m, CnMDATA45m, CnMDATA6m, CnMDATA7m, CnMDATA67m, CnMDLCm, CnMCONFm, CnMIDLm, CnMIDHm, and CnMCTRLm), or registers related to transmit history or receive history (CnLOPT, CnTGPT, CnLIPT, and CnRGPT) is disabled.
2. This bit is read-only. Even if 1 is written to the MBON bit while it is 0, the value of the MBON bit does not change, and access to the message buffer registers, or registers related to transmit history or receive history remains disabled.

Note The MBON bit is cleared (to 0) when the CAN module enters CAN sleep mode/ CAN stop mode, or when the GOM bit is cleared (to 0).
The MBON bit is set (to 1) when the CAN sleep mode/CAN stop mode is released, or when the GOM bit is set (to 1).

EFSD	Bit enabling forced shut down
0	Forced shut down by GOM bit = 0 disabled.
1	Forced shut down by GOM bit = 0 enabled.

Caution To request forced shut down, the GOM bit must be cleared to 0 in a subsequent, immediately following access after the EFSD bit has been set to 1. If access to another register (including reading the CnGMCTRL register) is executed 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).

(b) CnGMCTRL write

15	14	13	12	11	10	9	8
0	0	0	0	0	0	Set EFSD	Set GOM
7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	Clear GOM

Set EFSD	EFSD bit setting
0	No change in EFSD bit.
1	EFSD bit set to 1.

Set GOM	Clear GOM	GOM bit setting
0	1	GOM bit cleared to 0.
1	0	GOM bit set to 1.
Other than above		No change in GOM bit.

Caution Set the GOM bit and EFSD bit always separately.

(2) CnGMCS - CANn global clock selection register

The CnGMCS register is used to select the CAN module system clock.

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

Address <CnRBaseAddr> + 002_H

Initial Value $0F_H$. The register is initialized by any reset.

7	6	5	4	3	2	1	0
0	0	0	0	CCP3	CCP2	CCP1	CCP0

CCP3	CCP2	CCP1	CCP1	CAN module system clock (f_{CANMOD})
0	0	0	0	f_{CAN}/1
0	0	0	1	f_{CAN}/2
0	0	1	0	f_{CAN}/3
0	0	1	1	f_{CAN}/4
0	1	0	0	f_{CAN}/5
0	1	0	1	f_{CAN}/6
0	1	1	0	f_{CAN}/7
0	1	1	1	f_{CAN}/8
1	0	0	0	f_{CAN}/9
1	0	0	1	f_{CAN}/10
1	0	1	0	f_{CAN}/11
1	0	1	1	f_{CAN}/12
1	1	0	0	f_{CAN}/13
1	1	0	1	f_{CAN}/14
1	1	1	0	f_{CAN}/15
1	1	1	1	f_{CAN}/16 (default value)

Note f_{CAN} = clock supplied to CAN

(3) CnGMABT - CANn global automatic block transmission control register

The CnGMABT register is used to control the automatic block transmission (ABT) operation.

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

Address <CnRBaseAddr> + 006_H

Initial Value 0000_H. The register is initialized by any reset.

(a) CnGMABT read

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

ABTCLR	Automatic block transmission engine clear status bit
0	Clearing the automatic transmission engine is completed.
1	The automatic transmission engine is being cleared.

Note

1. Set the ABTCLR bit to 1 while the ABTTRG bit is cleared to 0.
   The operation is not guaranteed if the ABTCLR bit is set to 1 while the ABTTRG bit is set to 1.
2. When the automatic block transmission engine is cleared by setting the ABTCLR bit to 1, the ABTCLR bit is automatically cleared to 0 as soon as the requested clearing processing is complete.

ABTTRG	Automatic block transmission status bit
0	Automatic block transmission is stopped.
1	Automatic block transmission is under execution.

Caution

1. Do not set the ABTTRG bit (1) in the initialization mode. If the ABTTRG bit is set in the initialization mode, the operation is not guaranteed after the CAN module has entered the normal operation mode with ABT.
2. Do not set the ABTTRG bit (1) while the CnCTRL.TSTAT bit is set (1). Confirm TSTAT = 0 directly in advance before setting ABTTRG bit.

(b) CnGMABT write

15	14	13	12	11	10	9	8
0	0	0	0	0	0	Set ABTCLR	Set ABTTRG

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	Clear ABTTRG

Caution Before changing the normal operation mode with ABT to the initialization mode, be sure to set the CnGMABT register to the default value (0000_H) and confirm the CnGMABT register is surely initialized to the default value (0000_H).

Set ABTCLR	Automatic block transmission engine clear request bit
0	The automatic block transmission engine is in idle status or under operation.
1	Request to clear the automatic block transmission engine. After the automatic block transmission engine has been cleared, automatic block transmission is started from message buffer 0 by setting the ABTTRG bit to 1.

Set ABTTRG	Clear ABTTRG	Automatic block transmission start bit
0	1	Request to stop automatic block transmission.
1	0	Request to start automatic block transmission.
Other than above		No change in ABTTRG bit.

(4) CnGMABTD - CANn global automatic block transmission delay register

The CnGMABTD register is used to set the interval at which the data of the message buffer assigned to ABT is to be transmitted in the normal operation mode with ABT.

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

Address <CnRBaseAddr> + 008_H

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

7	6	5	4	3	2	1	0
0	0	0	0	ABTD3	ABTD2	ABTD1	ABTD0

ABTD3	ABTD2	ABTD1	ABTD0	Data frame interval during automatic block transmission in DBT[a]
0	0	0	0	0 DBT (default value)
0	0	0	1	2^5 DBT
0	0	1	0	2^6 DBT
0	0	1	1	2^7 DBT
0	1	0	0	2^8 DBT
0	1	0	1	2^9 DBT
0	1	1	0	2^{10} DBT
0	1	1	1	2^{11} DBT
1	0	0	0	2^{12} DBT
Other than above				Setting prohibited

a) Unit: Data bit time (DBT)

Caution

1. Do not change the contents of the CnGMABTD register while the ABTTRG bit is set to 1.
2. The timing at which the ABT message is actually transmitted onto the CAN bus differs depending on the status of transmission from the other station or how a request to transmit a message other than an ABT message (message buffers 8 to 31) is made.

(5) CnMASKaL, CnMASKaH - CANn module mask control register (a = 1 to 4)

The CnMASKaL and CnMASKaH registers are used to extend the number of receivable messages into the same message buffer by masking part of the identifier (ID) comparison of a message and invalidating the ID of the masked part.

(a) CANn module mask 1 register (CnMASK1L, CnMASK1H)

Access These registers can be read/written in 16-bit units.

Address CnMASK1L: <CnRBaseAddr> + 040_H
CnMASK1H: <CnRBaseAddr> + 042_H

Initial Value Undefined.

CnMASK1L

15	14	13	12	11	10	9	8
CMID15	CMID14	CMID13	CMID12	CMID11	CMID10	CMID9	CMID8
7	6	5	4	3	2	1	0
CMID7	CMID6	CMID5	CMID4	CMID3	CMID2	CMID1	CMID0

CnMASK1H

15	14	13	12	11	10	9	8
0	0	0	CMID28	CMID27	CMID26	CMID25	CMID24
7	6	5	4	3	2	1	0
CMID23	CMID22	CMID21	CMID20	CMID19	CMID18	CMID17	CMID16

(b) CANn module mask 2 register (CnMASK2L, CnMASK2H)

Access These registers can be read/written in 16-bit units.

Address CnMASK2L: <CnRBaseAddr> + 044_H
CnMASK2H: <CnRBaseAddr> + 046_H

Initial Value Undefined.

CnMASK2L

15	14	13	12	11	10	9	8
CMID15	CMID14	CMID13	CMID12	CMID11	CMID10	CMID9	CMID8
7	6	5	4	3	2	1	0
CMID7	CMID6	CMID5	CMID4	CMID3	CMID2	CMID1	CMID0

CnMASK2H

15	14	13	12	11	10	9	8
0	0	0	CMID28	CMID27	CMID26	CMID25	CMID24
7	6	5	4	3	2	1	0
CMID23	CMID22	CMID21	CMID20	CMID19	CMID18	CMID17	CMID16

(c) CANn module mask 3 register (CnMASK3L, CnMASK3H)

Access These registers can be read/written in 16-bit units.

Address CnMASK3L: <CnRBaseAddr> + 048_H
CnMASK3H: <CnRBaseAddr> + $04A_H$

Initial Value Undefined.

CnMASK3L

15	14	13	12	11	10	9	8
CMID15	CMID14	CMID13	CMID12	CMID11	CMID10	CMID9	CMID8
7	6	5	4	3	2	1	0
CMID7	CMID6	CMID5	CMID4	CMID3	CMID2	CMID1	CMID0

CnMASK3H

15	14	13	12	11	10	9	8
0	0	0	CMID28	CMID27	CMID26	CMID25	CMID24
7	6	5	4	3	2	1	0
CMID23	CMID22	CMID21	CMID20	CMID19	CMID18	CMID17	CMID16

(d) CANn module mask 4 register (CnMASK4L, CnMASK4H)

Access These registers can be read/written in 16-bit units.

Address CnMASK4L: <CnRBaseAddr> + $04C_H$
CnMASK4H: <CnRBaseAddr> + $04E_H$

Initial Value Undefined.

CnMASK4L

15	14	13	12	11	10	9	8
CMID15	CMID14	CMID13	CMID12	CMID11	CMID10	CMID9	CMID8
7	6	5	4	3	2	1	0
CMID7	CMID6	CMID5	CMID4	CMID3	CMID2	CMID1	CMID0

CnMASK4H

15	14	13	12	11	10	9	8
0	0	0	CMID28	CMID27	CMID26	CMID25	CMID24
7	6	5	4	3	2	1	0
CMID23	CMID22	CMID21	CMID20	CMID19	CMID18	CMID17	CMID16

CMID28 to CMID0	Mask pattern setting of ID bit
0	The ID bits of the message buffer set by the CMID28 to CMID0 bits are compared with the ID bits of the received message frame.
1	The ID bits of the message buffer set by the CMID28 to CMID0 bits are not compared with the ID bits of the received message frame (they are masked).

Note Masking is always defined by an ID length of 29 bits. If a mask is assigned to a message with a standard ID, the CMID17 to CMID0 bits are ignored. Therefore, only the CMID28 to CMID18 bits of the received ID are masked. The same mask can be used for both the standard and extended IDs.

(6) CnCTRL - CANn module control register

The CnCTRL register is used to control the operation mode of the CAN module.

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

Address <CnRBaseAddr> + 050_H

Initial Value 0000_H. The register is initialized by any reset.

(a) CnCTRL read

15	14	13	12	11	10	9	8
0	0	0	0	0	0	RSTAT	TSTAT
7	6	5	4	3	2	1	0
CCERC	AL	VALID	PSMODE1	PSMODE0	OPMODE2	OPMODE1	OPMODE0

RSTAT	Reception status bit
0	Reception is stopped.
1	Reception is in progress.

Note

1. The RSTAT bit is set to 1 under the following conditions (timing)
   - The SOF bit of a receive frame is detected
   - On occurrence of arbitration loss during a transmit frame
2. The RSTAT bit is cleared to 0 under the following conditions (timing)
   - When a recessive level is detected at the second bit of the interframe space
   - On transition to the initialization mode at the first bit of the interframe space

TSTAT	Transmission status bit
0	Transmission is stopped.
1	Transmission is in progress.

Note

1. The TSTAT bit is set to 1 under the following conditions (timing)
   - The SOF bit of a transmit frame is detected
2. The TSTAT bit is cleared to 0 under the following conditions (timing)
   - During transition to bus-off state
   - On occurrence of arbitration loss in transmit frame
   - On detection of recessive level at the second bit of the interframe space
   - On transition to the initialization mode at the first bit of the interframe space

CCERC	Error counter clear bit
0	The CnERC and CnINFO registers are not cleared in the initialization mode.
1	The CnERC and CnINFO registers are cleared in the initialization mode.

Note
1. The CCERC bit is used to clear the CnERC and CnINFO registers for re-initialization or forced recovery from the bus-off state. This bit can be set to 1 only in the initialization mode.
2. When the CnERC and CnINFO registers have been cleared, the CCERC bit is also cleared to 0 automatically.
3. The CCERC bit can be set to 1 at the same time as a request to change the initialization mode to an operation mode is made.
4. The CCERC bit is read-only in the CAN sleep mode or CAN stop mode.
5. The receive data may be corrupted in case of setting the CCERC bit to (1) immediately after entering the INIT mode from self-test mode.

AL	Bit to set operation in case of arbitration loss
0	Re-transmission is not executed in case of an arbitration loss in the single-shot mode.
1	Re-transmission is executed in case of an arbitration loss in the single-shot mode.

Note The AL bit is valid only in the single-shot mode.

VALID	Valid receive message frame detection bit
0	A valid message frame has not been received since the VALID bit was last cleared to 0.
1	A valid message frame has been received since the VALID bit was last cleared to 0.

Note
1. Detection of a valid receive message frame is not dependent upon storage in the receive message buffer (data frame) or transmit message buffer (remote frame).
2. Clear the VALID bit (0) before changing the initialization mode to an operation mode.
3. If only two CAN nodes are connected to the CAN bus with one transmitting a message frame in the normal mode and the other in the receive-only mode, the VALID bit is not set to 1 before the transmitting node enters the error passive state, because in receive-only mode no acknowledge is generated.
4. To clear the VALID bit, set the Clear VALID bit to 1 first and confirm that the VALID bit is cleared. If it is not cleared, perform clearing processing again.

PSMODE1	PSMODE0	Power save mode
0	0	No power save mode is selected.
0	1	CAN sleep mode
1	0	Setting prohibited
1	1	CAN stop mode

Caution

1. Transition to and from the CAN stop mode must be made via CAN sleep mode. A request for direct transition to and from the CAN stop mode is ignored.
2. The MBON flag of CnGMCTRL must be checked after releasing a power save mode, prior to access the message buffers again.
3. CAN sleep mode requests are kept pending, until cancelled by software or entered on appropriate bus condition (bus idle). Software can check the actual status by reading PSMODE.

OPMODE2	OPMODE1	OPMODE0	Operation mode
0	0	0	No operation mode is selected (CAN module is in the initialization mode).
0	0	1	Normal operation mode
0	1	0	Normal operation mode with automatic block transmission function (normal operation mode with ABT)
0	1	1	Receive-only mode
1	0	0	Single-shot mode
1	0	1	Self-test mode
Other than above			Setting prohibited

Caution Transit to initialization mode or power saving modes may take some time. Be sure to verify the success of mode change by reading the values, before proceeding.

Note The OPMODE0 to OPMODE2 bits are read-only in the CAN sleep mode or CAN stop mode.

(b) CnCTRL write

15	14	13	12	11	10	9	8
Set CCERC	Set AL	0	Set PSMODE1	Set PSMODE0	Set OPMODE2	Set OPMODE1	Set OPMODE0

7	6	5	4	3	2	1	0
0	Clear AL	Clear VALID	Clear PSMODE1	Clear PSMODE0	Clear OPMODE2	Clear OPMODE1	Clear OPMODE0

Set CCERC	Setting of CCERC bit
1	CCERC bit is set to 1.
Other than above	CCERC bit is not changed.

Set AL	Clear AL	Setting of AL bit
0	1	AL bit is cleared to 0.
1	0	AL bit is set to 1.
Other than above		AL bit is not changed.

Clear VALID	Setting of VALID bit
0	VALID bit is not changed.
1	VALID bit is cleared to 0.

Set PSMODE0	Clear PSMODE0	Setting of PSMODE0 bit
0	1	PSMODE0 bit is cleared to 0.
1	0	PSMODE0 bit is set to 1.
Other than above		PSMODE0 bit is not changed.

Set PSMODE1	Clear PSMODE1	Setting of PSMODE1 bit
0	1	PSMODE1 bit is cleared to 0.
1	0	PSMODE1 bit is set to 1.
Other than above		PSMODE1 bit is not changed.

Set OPMODE0	Clear OPMODE0	Setting of OPMODE0 bit
0	1	OPMODE0 bit is cleared to 0.
1	0	OPMODE0 bit is set to 1.
Other than above		OPMODE0 bit is not changed.

Set OPMODE1	Clear OPMODE1	Setting of OPMODE1 bit
0	1	OPMODE1 bit is cleared to 0.
1	0	OPMODE1 bit is set to 1.
Other than above		OPMODE1 bit is not changed.

Set OPMODE2	Clear OPMODE2	Setting of OPMODE2 bit
0	1	OPMODE2 bit is cleared to 0.
1	0	OPMODE2 bit is set to 1.
Other than above		OPMODE2 bit is not changed.

(7) CnLEC - CANn module last error information register

The CnLEC register provides the error information of the CAN protocol.

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

Address <CnRBaseAddr> + 052_H

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

7	6	5	4	3	2	1	0
0	0	0	0	0	LEC2	LEC1	LEC0

Note

1. The contents of the CnLEC register are not cleared when the CAN module changes from an operation mode to the initialization mode.
2. If an attempt is made to write a value other than 00_H to the CnLEC register by software, the access is ignored.

LEC2	LEC1	LEC0	Last CAN protocol error information
0	0	0	No error
0	0	1	Stuff error
0	1	0	Form error
0	1	1	ACK error
1	0	0	Bit error. (The CAN module tried to transmit a recessive-level bit as part of a transmit message (except the arbitration field), but the value on the CAN bus is a dominant-level bit.)
1	0	1	Bit error. (The CAN module tried to transmit a dominant-level bit as part of a transmit message, ACK bit, error frame, or overload frame, but the value on the CAN bus is a recessive-level bit.)
1	1	0	CRC error
1	1	1	Undefined

(8) CnINFO - CANn module information register

The CnINFO register indicates the status of the CAN module.

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

Address <CnRBaseAddr> + 053_H

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

7	6	5	4	3	2	1	0
0	0	0	BOFF	TECS1	TECS0	RECS1	RECS0

BOFF	Bus-off state bit
0	Not bus-off state (transmit error counter ≤ 255). (The value of the transmit error counter is less than 256.)
1	Bus-off state (transmit error counter > 255). (The value of the transmit error 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).

(9) CnERC - CANn module error counter register

The CnERC register indicates the count value of the transmission/reception error counter.

Access This register is read-only in 16-bit units.

Address <CnRBaseAddr> + 054_H

Initial Value 0000_H. The register is initialized by any reset.

15	14	13	12	11	10	9	8
REPS	REC6	REC5	REC4	REC3	REC2	REC1	REC0
7	6	5	4	3	2	1	0
TEC7	TEC6	TEC5	TEC4	TEC3	TEC2	TEC1	TEC0

REPS	Reception error passive status bit
0	The reception error counter is not in the error passive range (< 128)
1	The reception error counter is in the error passive range (≥ 128)

REC6 to REC0	Reception error counter bit
0 to 127	Number of reception errors. These bits reflect the status of the reception error counter. The number of errors is defined by the CAN protocol.

Note REC6 to REC0 of the reception error counter are invalid in the reception error passive state (CnINFO.RECS[1:0] = 11_B).

TEC7 to TEC0	Transmission error counter bit
0 to 255	Number of transmission errors. These bits reflect the status of the transmission error counter. The number of errors is defined by the CAN protocol.

Note The TEC7 to TEC0 bits of the transmission error counter are invalid in the bus-off state (CnINFO.BOFF = 1).

(10) CnIE - CANn module interrupt enable register

The CnIE register is used to enable or disable the interrupts of the CAN module.

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

Address <CnRBaseAddr> + 056_H

Initial Value 0000_H. The register is initialized by any reset.

(a) CnIE read

15	14	13	12	11	10	9	8
0	0	0	0	0	0	0	0
7	6	5	4	3	2	1	0
0	0	CIE5	CIE4	CIE3	CIE2	CIE1	CIE0

CIE5 to CIE0	CAN module interrupt enable bit
0	Output of the interrupt corresponding to interrupt status register CINTSx is disabled.
1	Output of the interrupt corresponding to interrupt status register CINTSx is enabled.

(b) CnIE write

15	14	13	12	11	10	9	8
0	0	Set CIE5	Set CIE4	Set CIE3	Set CIE2	Set CIE1	Set CIE0
7	6	5	4	3	2	1	0
0	0	Clear CIE5	Clear CIE4	Clear CIE3	Clear CIE2	Clear CIE1	Clear CIE0

Set CIE5	Clear CIE5	Setting of CIE5 bit
0	1	CIE5 bit is cleared to 0.
1	0	CIE5 bit is set to 1.
Other than above		CIE5 bit is not changed.

Set CIE4	Clear CIE4	Setting of CIE4 bit
0	1	CIE4 bit is cleared to 0.
1	0	CIE4 bit is set to 1.
Other than above		CIE4 bit is not changed.

Set CIE3	Clear CIE3	Setting of CIE3 bit
0	1	CIE3 bit is cleared to 0.
1	0	CIE3 bit is set to 1.
Other than above		CIE3 bit is not changed.

Set CIE2	Clear CIE2	Setting of CIE2 bit
0	1	CIE2 bit is cleared to 0.
1	0	CIE2 bit is set to 1.
Other than above		CIE2 bit is not changed.

Set CIE1	Clear CIE1	Setting of CIE1 bit
0	1	CIE1 bit is cleared to 0.
1	0	CIE1 bit is set to 1.
Other than above		CIE1 bit is not changed.

Set CIE0	Clear CIE0	Setting of CIE0 bit
0	1	CIE0 bit is cleared to 0.
1	0	CIE0 bit is set to 1.
Other than above		CIE0 bit is not changed.

(11) CnINTS - CANn module interrupt status register

The CnINTS register indicates the interrupt status of the CAN module.

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

Address <CnRBaseAddr> + 058_H

Initial Value 0000_H. The register is initialized by any reset.

(a) CnINTS read

15	14	13	12	11	10	9	8
0	0	0	0	0	0	0	0
7	6	5	4	3	2	1	0
0	0	CINTS5	CINTS4	CINTS3	CINTS2	CINTS1	CINTS0

CINTS5 to CINTS0	CAN interrupt status bit
0	No related interrupt source event is pending.
1	A related interrupt source event is pending.

Interrupt status bit	Related interrupt source event
CINTS5	Wakeup interrupt from CAN sleep mode[a]
CINTS4	Arbitration loss interrupt
CINTS3	CAN protocol error interrupt
CINTS2	CAN error status interrupt
CINTS1	Interrupt on completion of reception of valid message frame to message buffer m
CINTS0	Interrupt on normal completion of transmission of message frame from message buffer m

[a] The CINTS5 bit is set only when the CAN module is woken up from the CAN sleep mode by a CAN bus operation. The CINTS5 bit is not set when the CAN sleep mode has been released by software.

(b) CnINTS write

15	14	13	12	11	10	9	8
0	0	0	0	0	0	0	0
7	6	5	4	3	2	1	0
0	0	Clear CINTS5	Clear CINTS4	Clear CINTS3	Clear CINTS2	Clear CINTS1	Clear CINTS0

Clear CINTS5 to CINTS0	Setting of CINTS5 to CINTS0 bits
0	CINTS5 to CINTS0 bits are not changed.
1	CINTS5 to CINTS0 bits are cleared to 0.

Caution Please clear the status bit of this register with software when the confirmation of each status is necessary in the interrupt processing, because these bits are not cleared automatically.

(12) CnBRP - CANn module bit rate prescaler register

The CnBRP register is used to select the CAN protocol layer basic system clock (f_{TQ}). The communication baud rate is set to the CnBTR register.

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

Address <CnRBaseAddr> + $05A_H$

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

7	6	5	4	3	2	1	0
TQPRS7	TQPRS6	TQPRS5	TQPRS4	TQPRS3	TQPRS2	TQPRS1	TQPRS0

TQPRS7 to TQPRS0	CAN protocol layer base system clock (f_{TQ})
0	f_{CANMOD}/1
1	f_{CANMOD}/2
n	f_{CANMOD}/(n+1)
:	:
255	f_{CANMOD}/256 (default value)

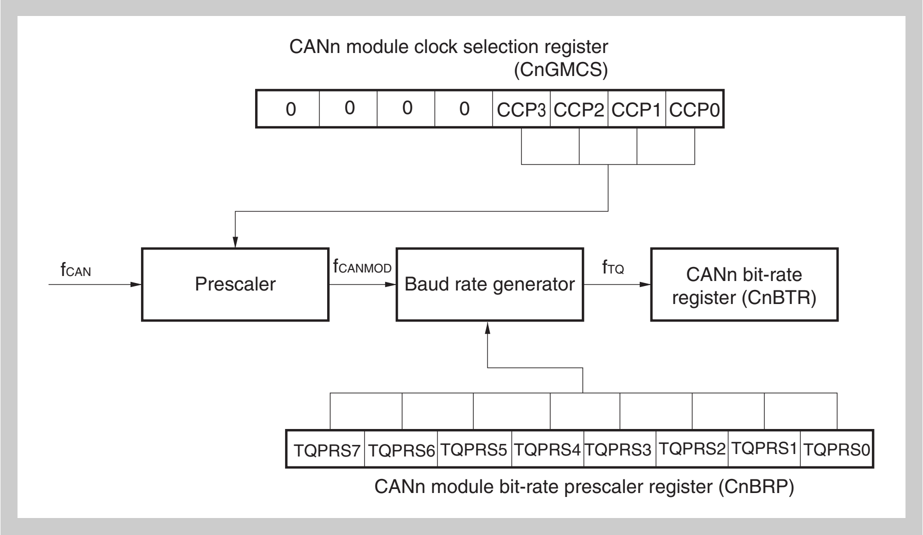


Figure 20-24 **CAN module clock**

Note

f_{CAN}: clock supplied to CAN
f_{CANMOD}: CAN module system clock
f_{TQ}: CAN protocol layer basic system clock

Caution The CnBRP register can be write-accessed only in the initialization mode.

(13) CnBTR - CANn module bit rate register

The CnBTR register is used to control the data bit time of the communication baud rate.

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

Address <CnRBaseAddr> + $05C_H$

Initial Value $370F_H$. The register is initialized by any reset.

15	14	13	12	11	10	9	8
0	0	SJW1	SJW0	0	TSEG22	TSEG21	TSEG20
7	6	5	4	3	2	1	0
0	0	0	0	TSEG13	TSEG12	TSEG11	TSEG10

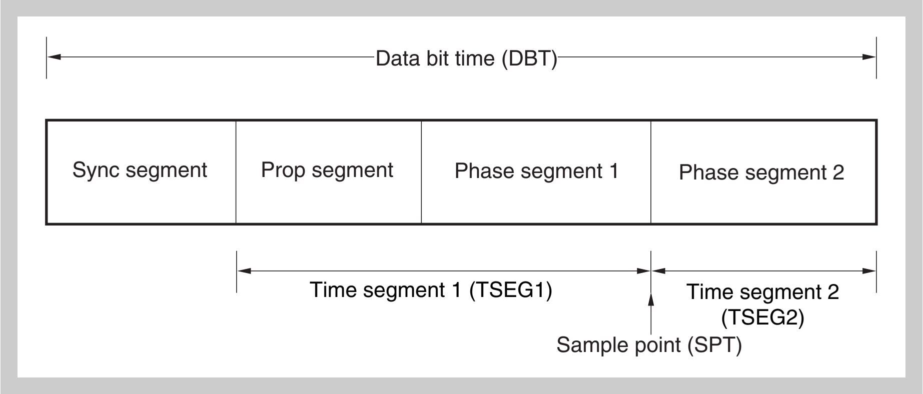


Figure 20-25 Data bit time

SJW1	SJW0	Length of synchronization jump width
0	0	$1T_Q$
0	1	$2T_Q$
1	0	$3T_Q$
1	1	$4T_Q$ (default value)

TSEG22	TSEG21	TSEG20	Length of time segment 2
0	0	0	$1T_Q$
0	0	1	$2T_Q$
0	1	0	$3T_Q$
0	1	1	$4T_Q$
1	0	0	$5T_Q$
1	0	1	$6T_Q$
1	1	0	$7T_Q$
1	1	1	$8T_Q$ (default value)

TSEG13	TSEG12	TSEG11	TSEG10	Length of time segment 1
0	0	0	0	Setting prohibited
0	0	0	1	$2T_Q$[a]
0	0	1	0	$3T_Q$[a]
0	0	1	1	$4T_Q$
0	1	0	0	$5T_Q$
0	1	0	1	$6T_Q$
0	1	1	0	$7T_Q$
0	1	1	1	$8T_Q$
1	0	0	0	$9T_Q$
1	0	0	1	$10T_Q$
1	0	1	0	$11T_Q$
1	0	1	1	$12T_Q$
1	1	0	0	$13T_Q$
1	1	0	1	$14T_Q$
1	1	1	0	$15T_Q$
1	1	1	1	$16T_Q$ (default value)

[a] This setting must not be made when the CnBRP register = 00_H

Note $T_Q = 1/f_{TQ}$ (f_{TQ}: CAN protocol layer basic system clock)

(14) CnLIPT - CANn module last in-pointer register

The CnLIPT register indicates the number of the message buffer in which a data frame or a remote frame was last stored.

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

Address <CnRBaseAddr> + $05E_H$

Initial Value Undefined.

7	6	5	4	3	2	1	0
LIPT7	LIPT6	LIPT5	LIPT4	LIPT3	LIPT2	LIPT1	LIPT0

LIPT7 to LIPT0	Last in-pointer register (CnLIPT)
0 to 31	When the CnLIPT register is read, the contents of the element indexed by the last in-pointer (LIPT) of the receive history list are read. These contents indicate the number of the message buffer in which a data frame or a remote frame was last stored.

Note The read value of the CnLIPT register is undefined if a data frame or a remote frame has never been stored in the message buffer. If the RHPM bit of the CnRGPT register is set to 1 after the CAN module has changed from the initialization mode to an operation mode, therefore, the read value of the CnLIPT register is undefined.

(15) CnRGPT - CANn module receive history list register

The CnRGPT register is used to read the receive history list.

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

Address <CnRBaseAddr> + 060_H

Initial Value $xx02_H$. The register is initialized by any reset.

(a) CnRGPT read

15	14	13	12	11	10	9	8
RGPT7	RGPT6	RGPT5	RGPT4	RGPT3	RGPT2	RGPT1	RGPT0
7	6	5	4	3	2	1	0
0	0	0	0	0	0	RHPM	ROVF

RGPT7 to RGPT0	Receive history list read pointer
0 to 31	When the CnRGPT register is read, the contents of the element indexed by the receive history list get pointer (RGPT) of the receive history list are read. These contents indicate the number of the message buffer in which a data frame or a remote frame has been stored.

RHPM[a]	Receive history list pointer match
0	The receive history list has at least one message buffer number that has not been read.
1	The receive history list has no message buffer numbers that have not been read.

[a] The read value of the RGPT0 to RGPT7 bits is invalid when the RHPM bit = 1.

ROVF[a]	Receive history list overflow bit
0	All the message buffer numbers that have not been read are preserved. All the numbers of the message buffers in which a new data frame or remote frame has been received and stored are recorded to the receive history list (the receive history list has a vacant element).
1	At least 23 entries have been stored since the host processor has serviced the RHL last time (i.e. read CnRGPT). The first 22 entries are sequentially stored while the last entry can have been overwritten whenever newly received message is stored because all buffer numbers are stored at position LIPT-1 when ROVF bit is set. Thus the sequence of receptions can not be recovered completely now.

[a] If ROVF is set, RHPM is no longer cleared on message storage, but RHPM is still set, if all entries of CnRGPT are read by software.

(b) CnRGPT write

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

Clear ROVF	Setting of ROVF bit
0	ROVF bit is not changed.
1	ROVF bit is cleared to 0.

(16) CnLOPT - CANn module last out-pointer register

The CnLOPT register indicates the number of the message buffer to which a data frame or a remote frame was transmitted last.

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

Address <CnRBaseAddr> + 062_H

Initial Value Undefined

7	6	5	4	3	2	1	0
LOPT7	LOPT6	LOPT5	LOPT4	LOPT3	LOPT2	LOPT1	LOPT0

LOPT7 to LOPT0	Last out-pointer of transmit history list (LOPT)
0 to 31	When the CnLOPT register is read, the contents of the element indexed by the last out-pointer (LOPT) of the receive history list are read. These contents indicate the number of the message buffer to which a data frame or a remote frame was transmitted last.

Note The value read from the CnLOPT register is undefined if a data frame or remote frame has never been transmitted from a message buffer. If the CnTGPT.THPM bit is set to 1 after the CAN module has changed from the initialization mode to an operation mode, therefore, the read value of the CnLOPT register is undefined.

(17) CnTGPT - CANn module transmit history list register

The CnTGPT register is used to read the transmit history list.

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

Address <CnRBaseAddr> + 064_H

Initial Value $xx02_H$. The register is initialized by any reset.

(a) CnTGPT read

15	14	13	12	11	10	9	8
TGPT7	TGPT6	TGPT5	TGPT4	TGPT3	TGPT2	TGPT1	TGPT0
7	6	5	4	3	2	1	0
0	0	0	0	0	0	THPM	TOVF

TGPT7 to TGPT0	Transmit history list read pointer
0 to 31	When the CnTGPT register is read, the contents of the element indexed by the read pointer (TGPT) of the transmit history list are read. These contents indicate the number of the message buffer to which a data frame or a remote frame was transmitted last.

THPM[a]	Transmit history pointer match
0	The transmit history list has at least one message buffer number that has not been read.
1	The transmit history list has no message buffer numbers that have not been read.

a) The read value of the TGPT0 to TGPT7 bits is invalid when the THPM bit = 1.

TOVF[a]	Transmit history list overflow bit
0	All the message buffer numbers that have not been read are preserved. All the numbers of the message buffers to which a new data frame or remote frame has been transmitted are recorded to the transmit history list (the transmit history list has a vacant element).
1	At least 7 entries have been stored since the host processor has serviced the THL last time (i.e. read CnTGPT). The first 6 entries are sequentially stored while the last entry can have been overwritten whenever a message is newly transmitted because all buffer numbers are stored at position LOPT-1 when TOVF bit is set. Thus the sequence of transmissions can not be recovered completely now.

a) If TOVF is set, THPM is no longer cleared on message transmission, but THPM is still set, if all entries of CnTGPT are read by software.

Note Transmission from message buffers 0 to 7 is not recorded to the transmit history list in the normal operation mode with ABT.

(b) CnTGPT write

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

Clear TOVF	Setting of TOVF bit
0	TOVF bit is not changed.
1	TOVF bit is cleared to 0.

(18) CnTS - CANn module time stamp register

The CnTS register is used to control the time stamp function.

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

Address <CnRBaseAddr> + 066_H

Initial Value 0000_H. The register is initialized by any reset.

(a) CnTS read

15	14	13	12	11	10	9	8
0	0	0	0	0	0	0	0
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0	0	0	0	0	TSLOCK	TSSEL	TSEN

Note The lock function of the time stamp function must not be used when the CAN module is in the normal operation mode with ABT.

TSLOCK	Time stamp lock function enable bit
0	Time stamp lock function stopped. The TSOUT signal is toggled each time the selected time stamp capture event occurs.
1	Time stamp lock function enabled. The TSOUT signal is toggled each time the selected time stamp capture event occurs. However, the TSOUT output signal is locked when a data frame has been correctly received to message buffer 0[a].

[a] The TSEN bit is automatically cleared to 0.

TSSEL	Time stamp capture event selection bit
0	The time capture event is SOF.
1	The time stamp capture event is the last bit of EOF.

TSEN	TSOUT operation setting bit
0	TSOUT toggle operation is disabled.
1	TSOUT toggle operation is enabled.

Remark The TSOUT signal is output from the CAN controller to the timer. For details, refer to *Chapter 11 on page 400.*"

(b) CnTS write

15	14	13	12	11	10	9	8
0	0	0	0	0	Set TSLOCK	Set TSSEL	Set TSEN
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0	0	0	0	0	Clear TSLOCK	Clear TSSEL	Clear TSEN

Set TSLOCK	Clear TSLOCK	Setting of TSLOCK bit
0	1	TSLOCK bit is cleared to 0.
1	0	TSLOCK bit is set to 1.
Other than above		TSLOCK bit is not changed.

Set TSSEL	Clear TSSEL	Setting of TSSEL bit
0	1	TSSEL bit is cleared to 0.
1	0	TSSEL bit is set to 1.
Other than above		TSSEL bit is not changed.

Set TSEN	Clear TSEN	Setting of TSEN bit
0	1	TSEN bit is cleared to 0.
1	0	TSEN bit is set to 1.
Other than above		TSEN bit is not changed.

(19) CnMDATAxm, CnMDATAzm - CANn message data byte register (x = 0 to 7, z = 01, 23, 45, 67)

The CnMDATAxm, CnMDATAzm registers are used to store the data of a transmit/receive message.

Access The CnMDATAzm registers can be read/written in 16-bit units.
The CnMDATAxm registers can be read/written in 8-bit units.

Address Refer to *"CAN registers overview" on page 702.*

Initial Value Undefined.

CnMDATA01m

15	14	13	12	11	10	9	8
MDATA0115	MDATA0114	MDATA0113	MDATA0112	MDATA0111	MDATA0110	MDATA019	MDATA018
7	6	5	4	3	2	1	0
MDATA017	MDATA016	MDATA015	MDATA014	MDATA013	MDATA012	MDATA011	MDATA010

CnMDATA0m

7	6	5	4	3	2	1	0
MDATA07	MDATA06	MDATA05	MDATA04	MDATA03	MDATA02	MDATA01	MDATA00

CnMDATA1m

7	6	5	4	3	2	1	0
MDATA17	MDATA16	MDATA15	MDATA14	MDATA13	MDATA12	MDATA11	MDATA1

CnMDATA23m

15	14	13	12	11	10	9	8
MDATA2315	MDATA2314	MDATA2313	MDATA2312	MDATA2311	MDATA2310	MDATA239	MDATA238
7	6	5	4	3	2	1	0
MDATA237	MDATA236	MDATA235	MDATA234	MDATA233	MDATA232	MDATA231	MDATA230

CnMDATA2m

7	6	5	4	3	2	1	0
MDATA27	MDATA26	MDATA25	MDATA24	MDATA23	MDATA22	MDATA21	MDATA20

CnMDATA3m

7	6	5	4	3	2	1	0
MDATA37	MDATA36	MDATA35	MDATA34	MDATA33	MDATA32	MDATA31	MDATA30

CnMDATA45m

15	14	13	12	11	10	9	8
MDATA4515	MDATA4514	MDATA4513	MDATA4512	MDATA4511	MDATA4510	MDATA459	MDATA458
7	6	5	4	3	2	1	0
MDATA457	MDATA456	MDATA455	MDATA454	MDATA453	MDATA452	MDATA451	MDATA450

CnMDATA4m

7	6	5	4	3	2	1	0
MDATA47	MDATA46	MDATA45	MDATA44	MDATA43	MDATA42	MDATA41	MDATA40

CnMDATA5m

7	6	5	4	3	2	1	0
MDATA57	MDATA56	MDATA55	MDATA54	MDATA53	MDATA52	MDATA51	MDATA50

CnMDATA67m

15	14	13	12	11	10	9	8
MDATA6715	MDATA6714	MDATA6713	MDATA6712	MDATA6711	MDATA6710	MDATA679	MDATA678
7	6	5	4	3	2	1	0
MDATA677	MDATA676	MDATA675	MDATA674	MDATA673	MDATA672	MDATA671	MDATA670

CnMDATA6m

7	6	5	4	3	2	1	0
MDATA67	MDATA66	MDATA65	MDATA64	MDATA63	MDATA62	MDATA61	MDATA60

CnMDATA7m

7	6	5	4	3	2	1	0
MDATA77	MDATA76	MDATA75	MDATA74	MDATA73	MDATA72	MDATA71	MDATA70

(20) CnMDLCm - CANn message data length register m

The CnMDLCm register is used to set the number of bytes of the data field of a message buffer.

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

Address Refer to *“CAN registers overview” on page 702.*

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

7	6	5	4	3	2	1	0
0	0	0	0	MDLC3	MDLC2	MDLC1	MDLC0

MDLC3	MDLC2	MDLC1	MDLC0	Data length of transmit/receive message
0	0	0	0	0 bytes
0	0	0	1	1 byte
0	0	1	0	2 bytes
0	0	1	1	3 bytes
0	1	0	0	4 bytes
0	1	0	1	5 bytes
0	1	1	0	6 bytes
0	1	1	1	7 bytes
1	0	0	0	8 bytes
1	0	0	1	Setting prohibited (If these bits are set during transmission, 8-byte data is transmitted regardless of the set DLC value when a data frame is transmitted. However, the DLC actually transmitted to the CAN bus is the DLC value set to this register.)[Note]
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

Note The data and DLC value actually transmitted to CAN bus are as follows.

Type of transmit frame	Length of transmit data	DLC transmitted
Data frame	Number of bytes specified by DLC (However, 8 bytes if DLC ≥ 8)	MDLC3 to MDLC0 bits
Remote frame	0 bytes	

Caution

1. Be sure to set bits 7 to 4 to 0000$_B$.
2. Receive data is stored in as many CnMDATAxm register as the number of bytes (however, the upper limit is 8) corresponding to DLC of the received frame. The CnMDATAxm register in which no data is stored is undefined.

(21) CnMCONFm - CANn message configuration register m

The CnMCONFm register is used to specify the type of the message buffer and to set a mask.

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

Address Refer to *"CAN registers overview" on page 702.*

Initial Value Undefined.

7	6	5	4	3	2	1	0
OWS	RTR	MT2	MT1	MT0	0	0	MA0

OWS	Overwrite control bit
0	The message buffer that has already received a data frame[a] is not overwritten by a newly received data frame. The newly received data frame is discarded.
1	The message buffer that has already received a data frame[a] is overwritten by a newly received data frame.

a) The "message buffer that has already received a data frame" is a receive message buffer whose the CnMCTRLm.DN bit has been set to 1.

Note A remote frame is received and stored, regardless of the setting of OWS and DN. A remote frame that satisfies the other conditions (ID matches, RTR = 0, TRQ = 0) is always received and stored in the corresponding message buffer (interrupt generated, DN flag set, MDLC[3:0] updated, and recorded to the receive history list).

RTR	Remote frame request bit[a]
0	Transmit a data frame.
1	Transmit a remote frame.

a) The RTR bit specifies the type of message frame that is transmitted from a message buffer defined as a transmit message buffer. Even if a valid remote frame has been received, the RTR bit of the transmit message buffer that has received the frame remains cleared to 0. Even if a remote frame whose ID matches has been received from the CAN bus with the RTR bit of the transmit message buffer set to 1 to transmit a remote frame, that remote frame is not received or stored (interrupt generated, DN flag set, the MDLC0 to MDLC3 bits updated, and recorded to the receive history list).

MT2	MT1	MT0	Message buffer type setting bit
0	0	0	Transmit message buffer
0	0	1	Receive message buffer (no mask setting)
0	1	0	Receive message buffer (mask 1 set)
0	1	1	Receive message buffer (mask 2 set)
1	0	0	Receive message buffer (mask 3 set)
1	0	1	Receive message buffer (mask 4 set)
Other than above			Setting prohibited

MA0	Message buffer assignment bit
0	Message buffer not used.
1	Message buffer used.

Caution Be sure to write 0 to bits 2 and 1.

(22) CnMIDLm, CnMIDHm - CANn message ID register m

The CnMIDLm and CnMIDHm registers are used to set an identifier (ID).

Access These registers can be read/written in 16-bit units.

Address Refer to *“CAN registers overview” on page 702.*

Initial Value Undefined.

CnMIDLm

15	14	13	12	11	10	9	8
ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8
7	6	5	4	3	2	1	0
ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0

CnMIDHm

15	14	13	12	11	10	9	8
IDE	0	0	ID28	ID27	ID26	ID25	ID24
7	6	5	4	3	2	1	0
ID23	ID22	ID21	ID20	ID19	ID18	ID17	ID16

IDE	Format mode specification bit
0	Standard format mode (ID28 to ID18: 11 bits)[a]
1	Extended format mode (ID28 to ID0: 29 bits)

[a] The ID17 to ID0 bits are not used.

ID28 to ID0	Message ID
ID28 to ID18	Standard ID value of 11 bits (when IDE = 0)
ID28 to ID0	Extended ID value of 29 bits (when IDE = 1)

Caution

1. Be sure to write 0 to bits 14 and 13 of the CnMIDHm register.
2. Be sure to align the ID value according to the given bit positions into this registers. Note that for standard ID, the ID value must be shifted to fit into ID28 to ID18 bit positions.

(23) CnMCTRLm - CANn message control register m

The CnMCTRLm register is used to control the operation of the message buffer.

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

Address Refer to *“CAN registers overview” on page 702.*

Initial Value 00x0 0000 0000 0000_B. The register is initialized by any reset.

(a) CnMCTRLm read

15	14	13	12	11	10	9	8
0	0	MUC	0	0	0	0	0
7	6	5	4	3	2	1	0
0	0	0	MOW	IE	DN	TRQ	RDY

MUC[a]	Bit indicating that message buffer data is being updated
0	The CAN module is not updating the message buffer (reception and storage).
1	The CAN module is updating the message buffer (reception and storage).

a) The MUC bit is undefined until the first reception and storage is performed.

MOW[a]	Message buffer overwrite status bit
0	The message buffer is not overwritten by a newly received data frame.
1	The message buffer is overwritten by a newly received data frame.

a) The MOW bit is not set to 1 even if a remote frame is received and stored in the transmit message buffer with the DN bit = 1.

IE	Message buffer interrupt request enable bit
0	Receive message buffer: Valid message reception completion interrupt disabled. Transmit message buffer: Normal message transmission completion interrupt disabled.
1	Receive message buffer: Valid message reception completion interrupt enabled. Transmit message buffer: Normal message transmission completion interrupt enabled.

DN	Message buffer data update bit
0	A data frame or remote frame is not stored in the message buffer.
1	A data frame or remote frame is stored in the message buffer.

TRQ	Message buffer transmission request bit
0	No message frame transmitting request that is pending or being transmitted is in the message buffer.
1	The message buffer is holding transmission of a message frame pending or is transmitting a message frame.

RDY	Message buffer ready bit
0	The message buffer can be written by software. The CAN module cannot write to the message buffer.
1	Writing the message buffer by software is ignored (except a write access to the RDY, TRQ, DN, and MOW bits). The CAN module can write to the message buffer.

(b) CnMCTRLm write

15	14	13	12	11	10	9	8
0	0	0	0	Set IE	0	Set TRQ	Set RDY
7	6	5	4	3	2	1	0
0	0	0	Clear MOW	Clear IE	Clear DN	Clear TRQ	Clear RDY

Clear MOW	Setting of MOW bit
0	MOW bit is not changed.
1	MOW bit is cleared to 0.

Set IE	Clear IE	Setting of IE bit
0	1	IE bit is cleared to 0.
1	0	IE bit is set to 1.
Other than above		IE bit is not changed.

Clear DN	Setting of DN bit
1	DN bit is cleared to 0.
0	DN bit is not changed.

Set TRQ	Clear TRQ	Setting of TRQ bit
0	1	TRQ bit is cleared to 0.
1	0	TRQ bit is set to 1.
Other than above		TRQ bit is not changed.

Set RDY	Clear RDY	Setting of RDY bit
0	1	RDY bit is cleared to 0.
1	0	RDY bit is set to 1.
Other than above		RDY bit is not changed.

Caution

1. Set IE bit and RDY bit always 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) for redefinition of 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.

20.8 CAN Controller Initialization

20.8.1 Initialization of CAN module

Before CAN module operation is enabled, the CAN module system clock needs to be determined by setting the CCP[3:0] bits of the CnGMCS register by software. Do not change the setting of the CAN module system clock after CAN module operation is enabled.

The CAN module is enabled by setting the GOM bit of the CnGMCTRL register.

For the procedure of initializing the CAN module, refer to *“Operation of CAN Controller” on page 791*.

20.8.2 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.

20.8.3 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 20-38*.

(3) To redefine message buffer during transmission

To rewrite the contents of a transmit message buffer to which a transmission request has been set, perform transmission abort processing (see *“Transmission abort process except for in normal operation mode with automatic block transmission (ABT)” on page 770* and *“Transmission abort process except for ABT transmission in normal operation mode with automatic block transmission (ABT)” on page 770*). 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.

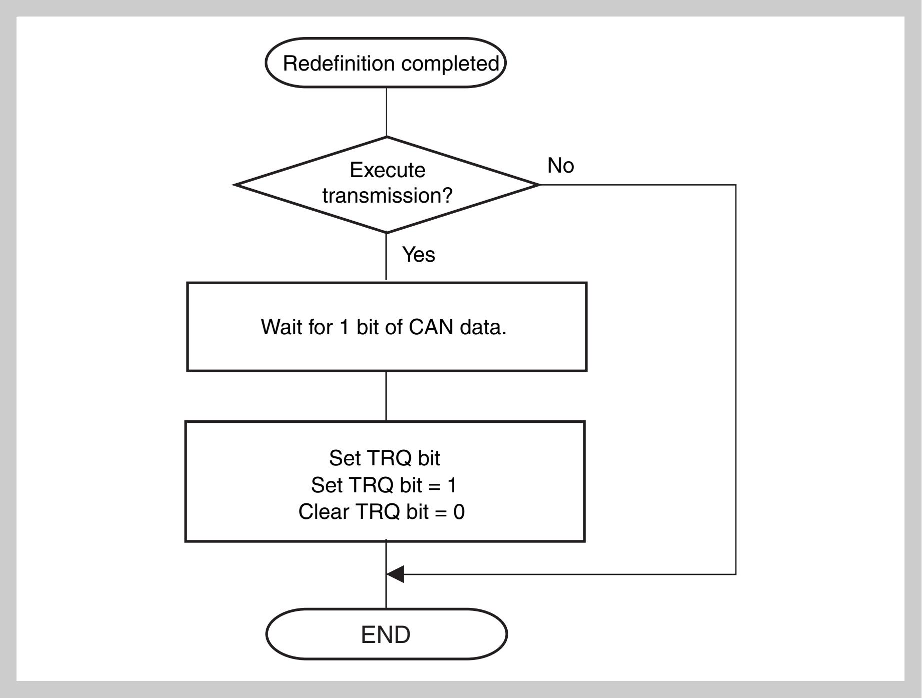


Figure 20-26 Setting transmission request (TRQ) to transmit message buffer after redefinition

Caution

1. When a message is received, reception filtering is performed in accordance with the ID and mask set to each receive message buffer. If the procedure in *Figure 20-38 on page 794* 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 20-26 on page 754* is not observed, a message with an ID not having the highest priority may be transmitted after redefinition.

20.8.4 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

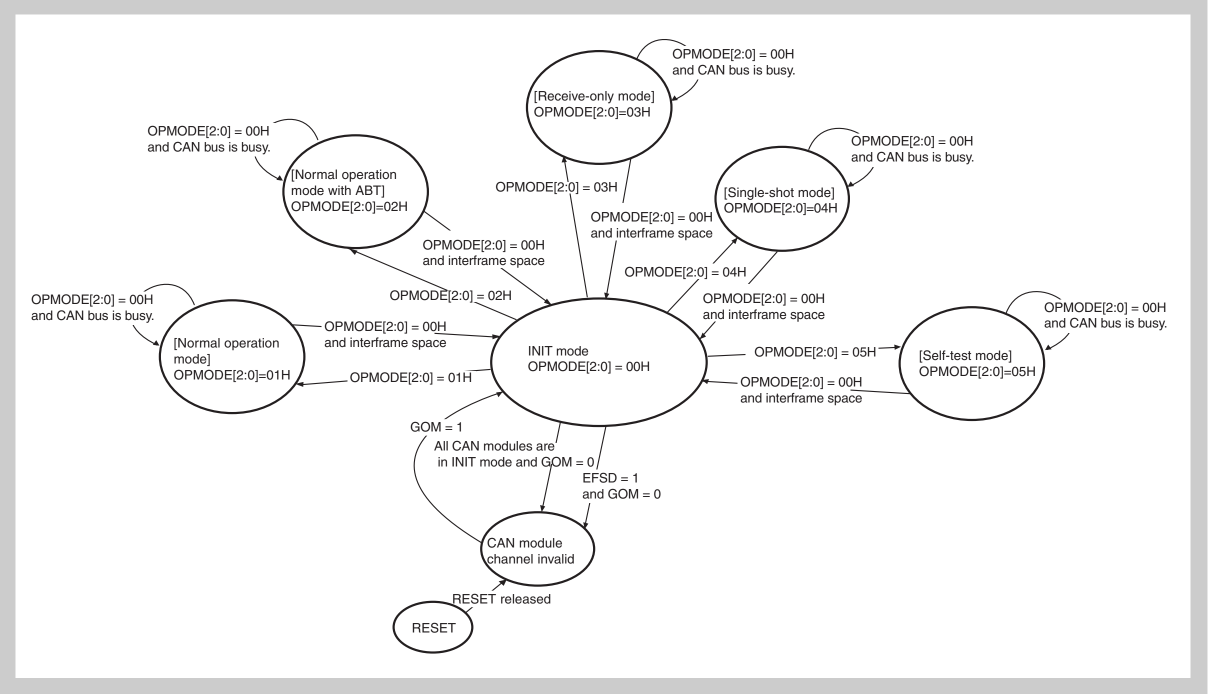


Figure 20-27 Transition to operation modes

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 20-36 on page 792*).

20.8.5 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.

20.9 Message Reception

20.9.1 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 20-32 MBRB priorities

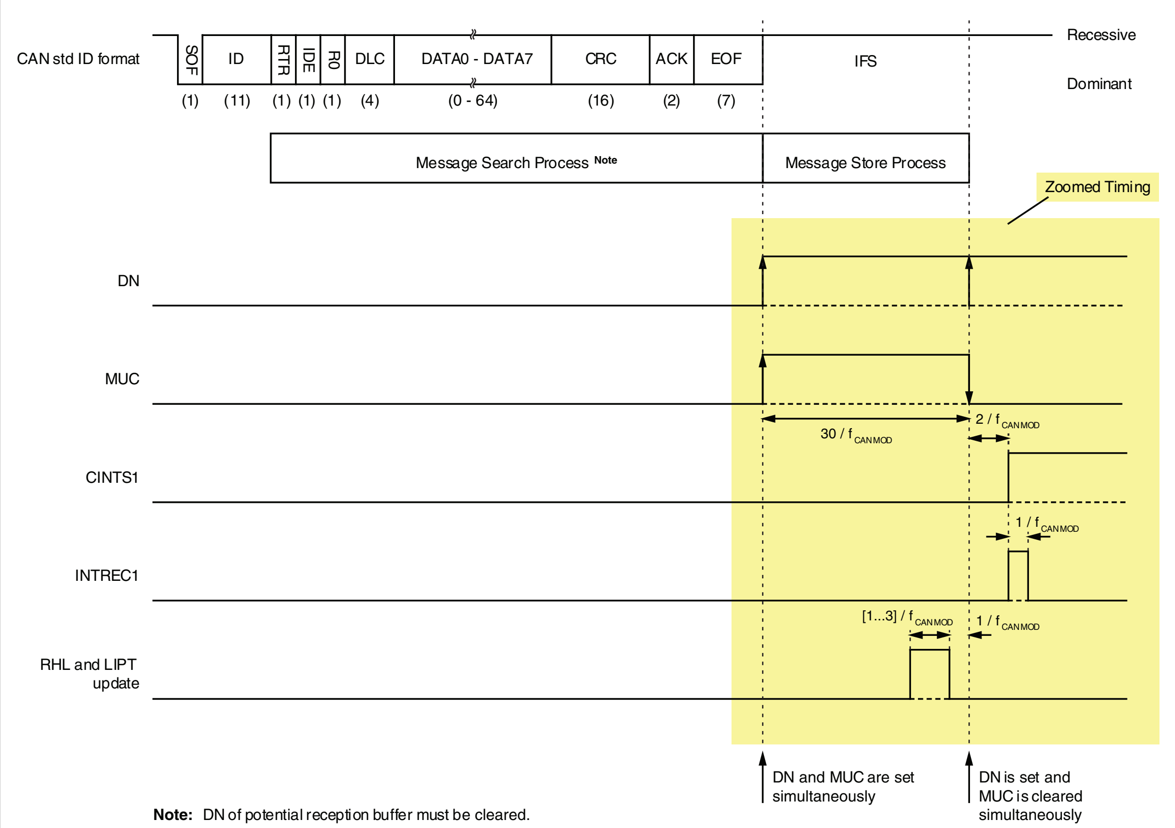
Priority	Storing condition if same ID is set	
1 (high)	Unmasked message buffer	DN bit = 0
		DN bit = 1 and OWS bit = 1
2	Message buffer linked to mask 1	DN bit = 0
		DN bit = 1 and OWS bit = 1
3	Message buffer linked to mask 2	DN bit = 0
		DN bit = 1 and OWS bit = 1
4	Message buffer linked to mask 3	DN bit = 0
		DN bit = 1 and OWS bit = 1
5 (low)	Message buffer linked to mask 4	DN bit = 0
		DN bit = 1 and OWS bit = 1

20.9.2 Receive data read

To keep data consistency when reading CAN message buffers, perform the data reading according to *Figure 20-49 on page 805* to *Figure 20-52 on page 808*.

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 20-28 on page 758*.)

The receive history list is also updated just before the storgage process. In addition, during storage process (MUC = 1), the RDY bit of the CnMCTRL register of the message buffer is locked to avoid the coincidental data WR by CPU. Note the storage process may be disturbed (delayed) when the CPU accesses the message buffer.



Note: DN of potential reception buffer must be cleared.

Figure 20-28 DN and MUC bit setting period (for standard ID format)

Note If a message shall be stored in a message buffer, the DN bit of this buffer must be cleared before the Message Search Process is started, i.e., right after the ID of the frame is on the bus. In worst case, this happens 15 CAN bits after EOF of the previous frame. Consider to use more than one Message Buffer for reception of a frame, if CAN frames are appearing back-to-back on the bus and none shall be lost.

20.9.3 Receive history list function

The receive history list (RHL) function records in the receive history list the number of the receive message buffer in which each data frame or remote frame was received and stored. The RHL consists of storage elements equivalent to up to 23 messages, the last in-message pointer (LIPT) with the corresponding CnLIPT register and the receive history list get pointer (RGPT) with the corresponding CnRGPT register.

The RHL is undefined immediately after the transition of the CAN module from the initialization mode to one of the operation modes.

The CnLIPT register holds the contents of the RHL element indicated by the value of the LIPT pointer minus 1. By reading the CnLIPT register, therefore, the number of the message buffer that received and stored a data frame or remote frame first can be checked. The LIPT pointer is utilized as a write pointer that indicates to what part of the RHL a message buffer number is recorded. Any time a data frame or remote frame is received and stored, the corresponding message buffer number is recorded to the RHL element indicated by the LIPT pointer. Each time recording to the RHL has been completed, the LIPT pointer is automatically incremented. In this way, the number of the message buffer that has received and stored a frame will be recorded chronologically.

The RGPT pointer is utilized as a read pointer that reads a recorded message buffer number from the RHL. This pointer indicates the first RHL element that the CPU has not read yet. By reading the CnRGPT register by software, the number of a message buffer that has received and stored a data frame or remote frame can be read. Each time a message buffer number is read from the CnRGPT register, the RGPT pointer is automatically incremented.

If the value of the RGPT pointer matches the value of the LIPT pointer, the RHPM bit (receive history list pointer match) of the CnRGPT register is set to 1. This indicates that no message buffer number that has not been read remains in the RHL. If a new message buffer number is recorded, the LIPT pointer is incremented and because its value no longer matches the value of the RGPT pointer, the RHPM bit is cleared. In other words, the numbers of the unread message buffers exist in the RHL.

If the LIPT pointer is incremented and matches the value of the RGPT pointer minus 1, the ROVF bit (receive history list overflow) of the CnRGPT register is set to 1. This indicates that the RHL is full of numbers of message buffers that have not been read. When further message reception and storing occur, the last recorded message buffer number is overwritten by the number of the message buffer that received and stored the newly received message. In this case, after the ROVF bit has been set (1), the recorded message buffer numbers in the RHL do not completely reflect the chronological order. However messages itself are not lost and can be located by CPU search in message buffer memory with the help of the DN-bit.

Caution If the history list is in the overflow condition (ROVF is set), reading the history list contents is still possible, until the history list is empty (indicated by RHPM flag set). Nevertheless, the history list remains in the overflow condition, until ROVF is cleared by software. If ROVF is not cleared, the RHPM flag will also not be updated (cleared) upon a message storage of newly received frame. This may lead to the situation, that RHPM indicates an empty history list, although a reception has taken place, while the history list is in the overflow state (ROVF and RHPM are set).

As long as the RHL contains 23 or less entries the sequence of occurrence is maintained. If more receptions occur without reading the RHL by the host processor, complete sequence of receptions can not be recovered.

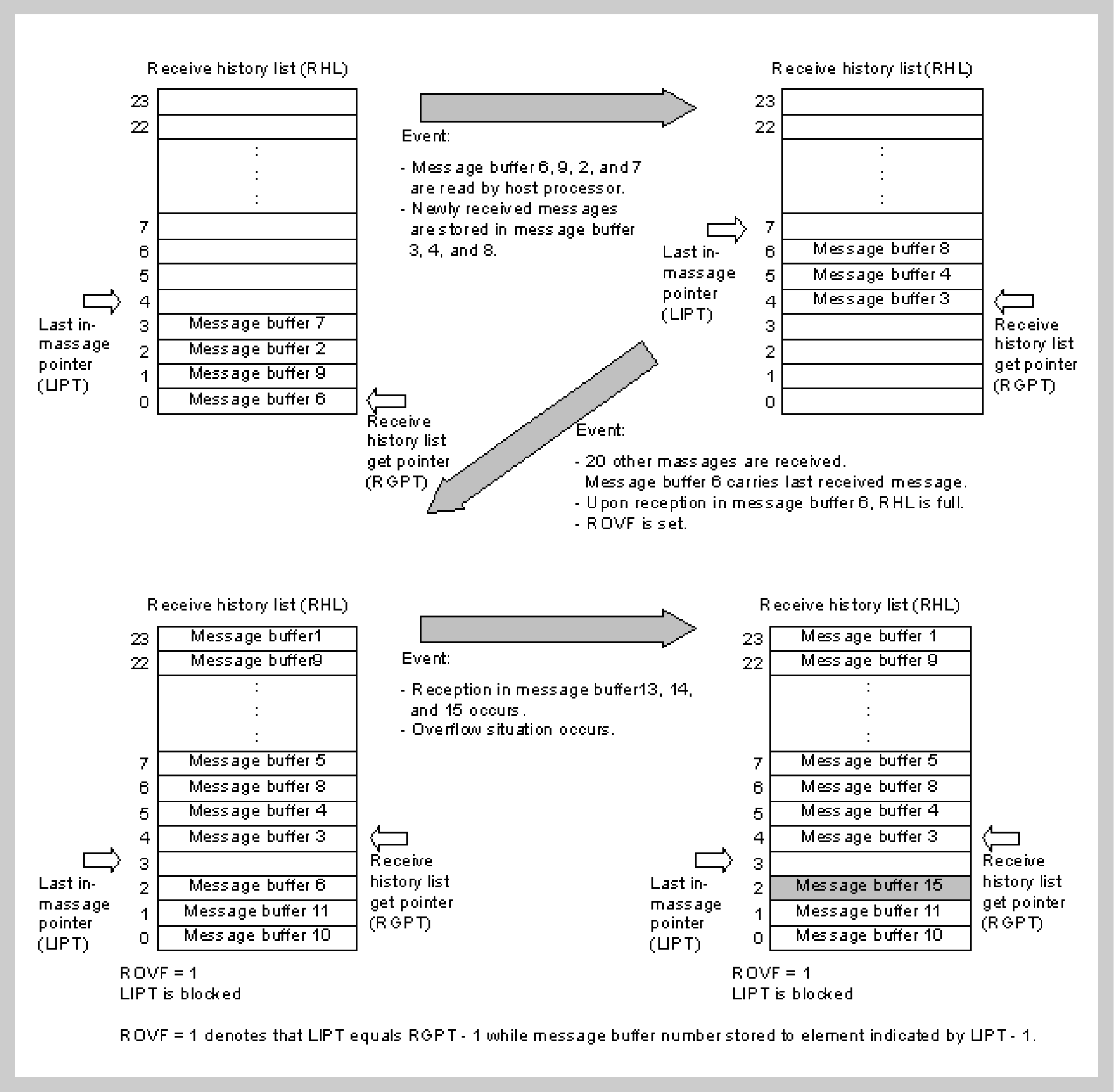

Receive history list (RHL)
23
22
7
6
5
4
3
2
1
0
Message buffer 7
Message buffer 2
Message buffer 9
Message buffer 6
Last in-massage pointer (LIPT)
Receive history list get pointer (RGPT)
Event:
- Message buffer 6, 9, 2, and 7 are read by host processor.
- Newly received messages are stored in message buffer 3, 4, and 8.
Receive history list(RHL)
Message buffer 8
Message buffer 4
Message buffer 3
Last in-massage pointer (LIPT)
Receive history list get pointer (RGPT)
Event:
- 20 other massages are received. Message buffer 6 carries last received message.
- Upon reception in message buffer 6, RHL is full.
- ROVF is set.
Receive history list (RHL)
Message buffer1
Message buffer9
Message buffer 5
Message buffer 8
Message buffer 4
Message buffer 3
Message buffer 6
Message buffer 11
Message buffer 10
Last in-massage pointer (LIPT)
Receive history list get pointer (RGPT)
ROVF = 1
LIPT is blocked
Event:
- Reception in message buffer13, 14, and 15 occurs.
- Overflow situation occurs.
Receive history list (RHL)
Message buffer 1
Message buffer 9
Message buffer 5
Message buffer 8
Message buffer 4
Message buffer 3
Message buffer 15
Message buffer 11
Message buffer 10
Last in-massage pointer (LIPT)
Receive history list get pointer (RGPT)
ROVF = 1
LIPT is blocked
ROVF = 1 denotes that LIPT equals RGPT - 1 while message buffer number stored to element indicated by LIPT - 1.

Figure 20-29 Receive history list

20.9.4 Mask function

For any message buffer, which is used for reception, the assignment to one of four global reception masks (or no mask) can be selected.

By using the mask function, the message ID comparison can be reduced by masked bits, herewith allowing the reception of several different IDs into one buffer.

While the mask function is in effect, an identifier bit that is defined to be 1 by a mask in the received message is not compared with the corresponding identifier bit in the message buffer.

However, this comparison is performed for any bit whose value is defined as 0 by the mask.

For example, let us assume that all messages that have a standard-format ID, in which bits ID27 to ID25 are 0 and bits ID24 and ID22 are 1, are to be stored in message buffer 14. The procedure for this example is shown below.

1. Identifier to be stored in message buffer

ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21	ID20	ID19	ID18
×	0	0	0	1	×	1	×	×	×	×

2. Identifier to be configured in message buffer 14 (example)
(Using CnMIDL14 and CnMIDH14 registers)

ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21	ID20	ID19	ID18
×	0	0	0	1	×	1	×	×	×	×
ID17	ID16	ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7
×	×	×	×	×	×	×	×	×	×	×
ID6	ID5	ID4	ID3	ID2	ID1	ID0				
×	×	×	×	×	×	×				

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				

Note
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.

20.9.5 Multi buffer receive block function

The multi buffer receive block (MBRB) function is used to store a block of data in two or more message buffers sequentially with no CPU interaction, by setting the same ID to two or more message buffers with the same message buffer type. These message buffers can be allocated anywhere in the message buffer memory, they do not even have to follow each other adjacently.

Suppose, for example, the same message buffer type is set to 10 message buffers, message buffers 10 to 19, and the same ID is set to each message buffer. If the first message whose ID matches an ID of the message buffers is received, it is stored in message buffer 10. At this point, the DN bit of message buffer 10 is set, prohibiting overwriting the message buffer when subsequent messages are received.

When the next message with a matching ID is received, it is received and stored in message buffer 11. Each time a message with a matching ID is received, it is sequentially (in the ascending order) stored in message buffers 12, 13, and so on. Even when a data block consisting of multiple messages is received, the messages can be stored and received without overwriting the previously received matching-ID data.

Whether a data block has been received and stored can be checked by setting the IE bit of the CnMCTRLm register of each message buffer. For example, if a data block consists of k messages, k message buffers are initialized for reception of the data block. The IE bit in message buffers 0 to (k-2) is cleared to 0 (interrupts disabled), and the IE bit in message buffer k-1 is set to 1 (interrupts enabled). In this case, a reception completion interrupt occurs when a message has been received and stored in message buffer k-1, indicating that MBRB has become full. Alternatively, by clearing the IE bit of message buffers 0 to (k-3) and setting the IE bit of message buffer k-2, a warning that MBRB is about to overflow can be issued.

The basic conditions of storing receive data in each message buffer for the MBRB are the same as the conditions of storing data in a single message buffer.

Caution

1. MBRB can be configured for each of the same message buffer types. Therefore, even if a message buffer of another MBRB whose ID matches but whose message buffer type is different has a vacancy, the received message is not stored in that message buffer, but instead discarded.
2. MBRB does not have a ring buffer structure. Therefore, after a message is stored in the message buffer having the highest number in the MBRB configuration, a newly received message will not be stored in the message buffer having the lowest message buffer number.
3. MBRB operates based on the reception and storage conditions; there are no settings dedicated to MBRB, such as function enable bits. By setting the same message buffer type and ID to two or more message buffers, MBRB is automatically configured.
4. With MBRB, “matching ID” means “matching ID after mask”. Even if the ID set to each message buffer is not the same, if the ID that is masked by the mask register matches, it is considered a matching ID and the buffer that has this ID is treated as the storage destination of a message.
5. The priority between MBRBs is mentioned in the table *Table 20-32*.

20.9.6 Remote frame reception

In all the operation modes, when a remote frame is received, the message buffer that is to store the remote frame is searched from all the message buffers satisfying the following conditions.

- Used as a message buffer
  (MA0 bit of CnMCONFm register set to 1.)
- Set as a transmit message buffer
  (MT[2:0] bits in CnMCONFm register set to 000_B)
- Ready for reception
  (RDY bit of CnMCTRLm register set to 1.)
- Set to transmit message
  (RTR bit of CnMCONFm register is cleared to 0.)
- Transmission request is not set.
  (TRQ bit of CnMCTRLm register is cleared to 0.)

Upon acceptance of a remote frame, the following actions are executed if the ID of the received remote frame matches the ID of a message buffer that satisfies the above conditions.

- The DLC[3:0] bit string in the CnMDLCm register store the received DLC value.
- The CnMDATA0m to CnMDATA7m registers in the data area are not updated (data before reception is saved).
- The DN bit of the CnMCTRLm register is set to 1.
- The CINTS1 bit of the CnINTS register is set to 1 (if the IE bit in the CnMCTRLm register of the message buffer that receives and stores the frame is set to 1).
- The receive completion interrupt (INTCnREC) is output (if the IE bit of the message buffer that receives and stores the frame is set to 1 and if the CIE1 bit of the CnIE register is set to 1).
- The message buffer number is recorded in the receive history list.

Caution When a message buffer is searched for receiving and storing a remote frame, overwrite control by the OWS bit of the CnMCONFm register of the message buffer and the DN bit of the CnMCTRLm register are not checked. The setting of OWS is ignored, and DN is set in any case.
If more than one transmit message buffer has the same ID and the ID of the received remote frame matches that ID, the remote frame is stored in the transmit message buffer with the lowest message buffer number.

20.10 Message Transmission

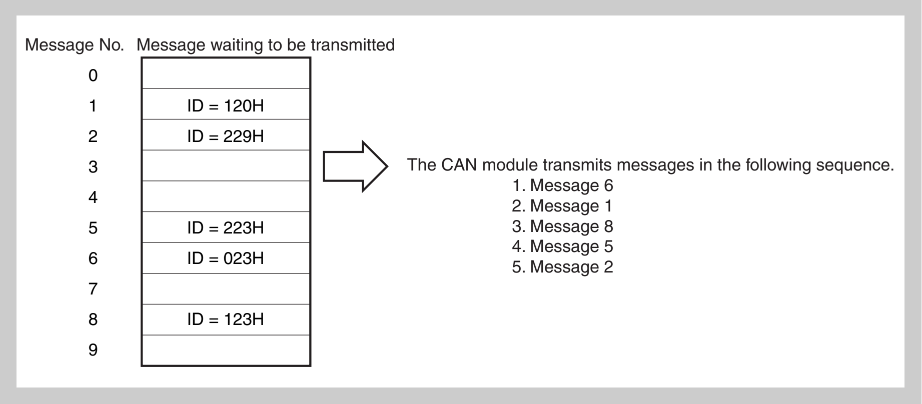
20.10.1 Message transmission

A message buffer with its TRQ bit set to 1 participates in the search for the most high-prioritized message when the following conditions are fulfilled. This behavior is valid for all operational modes.

- Used as a message buffer
  (MA0 bit of CnMCONFm register set to 1.)
- Set as a transmit message buffer
  (MT[2:0] bits of CnMCONFm register set to 000_B.)
- Ready for transmission
  (RDY bit of CnMCTRLm register set to 1.)

The CAN system is a multi-master communication system. In a system like this, the priority of message transmission is determined based on message identifiers (IDs). To facilitate transmission processing by software when there are several messages awaiting transmission, the CAN module uses hardware to check the ID of the message with the highest priority and automatically identifies that message. This eliminates the need for software-based priority control.

Transmission priority is controlled by the identifier (ID).



Message No.	Message waiting to be transmitted
0	
1	ID = 120H
2	ID = 229H
3	
4	
5	ID = 223H
6	ID = 023H
7	
8	ID = 123H
9	

The CAN module transmits messages in the following sequence.
1. Message 6
2. Message 1
3. Message 8
4. Message 5
5. Message 2

Figure 20-30 Message processing example

After the transmit message search, the transmit message with the highest priority of the transmit message buffers that have a pending transmission request (message buffers with the TRQ bit set to 1 in advance) is transmitted.

If a new transmission request is set, the transmit message buffer with the new transmission request is compared with the transmit message buffer with a pending transmission request. If the new transmission request has a higher priority, it is transmitted, unless transmission of a message with a low priority has already started. If transmission of a message with a low priority has already started, however, the new transmission request is transmitted later. To solve this priority inversion effect, the software can perform a transmission abort request for the lower priority message. The highest priority is determined according to the following rules.

Priority	Conditions	Description
1 (high)	Value of first 11 bits of ID [ID28 to ID18]:	The message frame with the lowest value represented by the first 11 bits of the ID is transmitted first. If the value of an 11-bit standard ID is equal to or smaller than the first 11 bits of a 29-bit extended ID, the 11-bit standard ID has a higher priority than a message frame with a 29-bit extended ID.
2	Frame type	A data frame with an 11-bit standard ID (RTR bit is cleared to 0) has a higher priority than a remote frame with a standard ID and a message frame with an extended ID.
3	ID type	A message frame with a standard ID (IDE bit is cleared to 0) has a higher priority than a message frame with an extended ID.
4	Value of lower 18 bits of ID [ID17 to ID0]:	If two or more transmission-pending extended ID message frame has equal values in the first 11 bits of the ID and the same frame type (equal RTR bit values), the message frame with the lowest value in the lower 18 bits of its extended ID is transmitted first.
5 (low)	Message buffer number	If two or more message buffers request transmission of message frames with the same ID, the message from the message buffer with the lowest message buffer number is transmitted first.

Note 1. If the automatic block transmission request bit ABTTRG is set to 1 in the normal operation mode with ABT, the TRQ bit is set to 1 only for one message buffer in the ABT message buffer group.

If the ABT mode was triggered by ABTTRG bit (1), one TRQ bit is set to 1 in the ABT area (buffer 0 through 7). Beyond this TRQ bit, the application can request transmissions (set TRQ bit to 1) for other TX-message buffers that do not belong to the ABT area. In that case an interval arbitration process (TX-search) evaluates all TX-message buffers with TRQ bit set to 1 and chooses the message buffer that contains the highest prioritized identifier for the next transmission. If there are 2 or more identifiers that have the highest priority (i.e. identical identifiers), the message located at the lowest message buffer number is transmitted at first.

Upon successful transmission of a message frame, the following operations are performed.

- The TRQ flag of the corresponding transmit message buffer is automatically cleared to 0.
- The transmission completion status bit CINTS0 of the CnINTS register is set to 1 (if the interrupt enable bit (IE) of the corresponding transmit message buffer is set to 1).
- An interrupt request signal INTCnTRX is output (if the CIE0 bit of the CnIE register is set to 1 and if the interrupt enable bit (IE) of the corresponding transmit message buffer is set to 1).

2. When changing the contents of a transmit buffer, the RDY flag of this buffer must be cleared before updating the buffer contents. As during internal transfer actions, the RDY flag may be locked temporarily, the status of RDY must be checked by software, after changing it.

20.10.2 Transmit history list function

The transmit history list (THL) function records in the transmit history list the number of the transmit message buffer from which data or remote frames have been were sent. The THL consists of storage elements equivalent to up to seven messages, the last out-message pointer (LOPT) with the corresponding CnLOPT register, and the transmit history list get pointer (TGPT) with the corresponding CnTGPT register.

The THL is undefined immediately after the transition of the CAN module from the initialization mode to one of the operation modes.

The CnLOPT register holds the contents of the THL element indicated by the value of the LOPT pointer minus 1. By reading the CnLOPT register, therefore, the number of the message buffer that transmitted a data frame or remote frame first can be checked. The LOPT pointer is utilized as a write pointer that indicates to what part of the THL a message buffer number is recorded. Any time a data frame or remote frame is transmitted, the corresponding message buffer number is recorded to the THL element indicated by the LOPT pointer. Each time recording to the THL has been completed, the LOPT pointer is automatically incremented. In this way, the number of the message buffer that has received and stored a frame will be recorded chronologically.

The TGPT pointer is utilized as a read pointer that reads a recorded message buffer number from the THL. This pointer indicates the first THL element that the CPU has not yet read. By reading the CnTGPT register by software, the number of a message buffer that has completed transmission can be read. Each time a message buffer number is read from the CnTGPT register, the TGPT pointer is automatically incremented.

If the value of the TGPT pointer matches the value of the LOPT pointer, the THPM bit (transmit history list pointer match) of the CnTGPT register is set to 1. This indicates that no message buffer numbers that have not been read remain in the THL. If a new message buffer number is recorded, the LOPT pointer is incremented and because its value no longer matches the value of the TGPT pointer, the THPM bit is cleared. In other words, the numbers of the unread message buffers exist in the THL.

If the LOPT pointer is incremented and matches the value of the TGPT pointer minus 1, the TOVF bit (transmit history list overflow) of the CnTGPT register is set to 1. This indicates that the THL is full of message buffer numbers that have not been read. If a new message is received and stored, the message buffer number recorded last is overwritten by the message buffer number that transmitted its message afterwards. In this case, after the TOVF bit has been set (1), therefore, the recorded message buffer numbers in the THL do not completely reflect the chronological order. However the other transmitted messages can be found by a CPU search applied to all transmit message buffers unless the CPU has not overwritten a transmit object in one of these buffers beforehand. In total up to six transmission completions can occur without overflowing the THL.

Caution If the history list is in the overflow condition (TOVF is set), reading the history list contents is still possible, until the history list is empty (indicated by THPM flag set). Nevertheless, the history list remains in the overflow condition, until TOVF is cleared by software. If TOVF is not cleared, the THPM flag will also not be updated (cleared) upon successful transmission of a new message. This may lead to the situation, that THPM indicates an empty history list, although a successful transmission has taken place, while the history list is in the overflow state (TOVF and THPM are set).

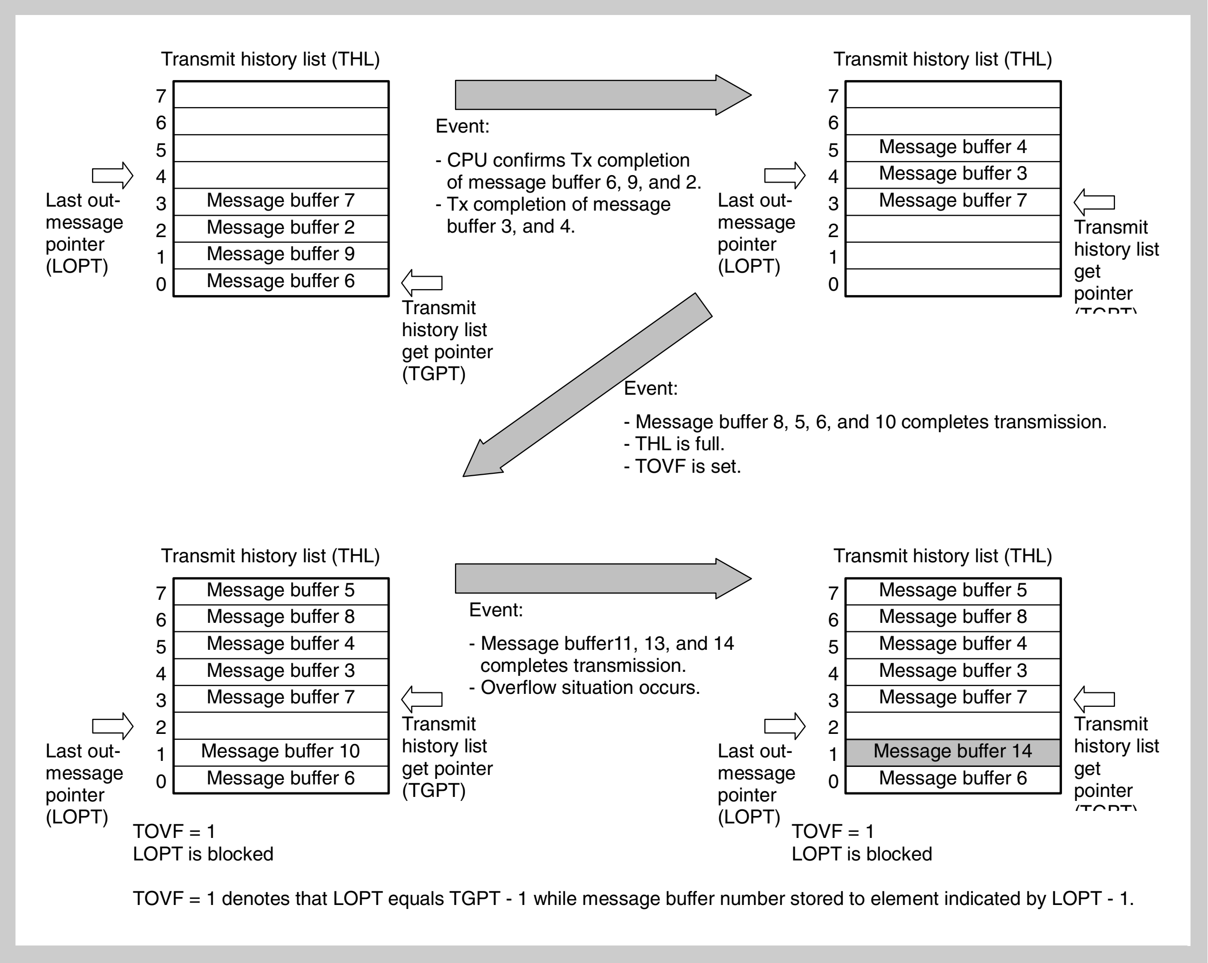

Transmit history list (THL)
7
6
5
4
3 Message buffer 7
2 Message buffer 2
1 Message buffer 9
0 Message buffer 6
Last out-message pointer (LOPT)
Transmit history list get pointer (TGPT)
Event:
- CPU confirms Tx completion of message buffer 6, 9, and 2.
- Tx completion of message buffer 3, and 4.
Transmit history list (THL)
5 Message buffer 4
4 Message buffer 3
3 Message buffer 7
Last out-message pointer (LOPT)
Transmit history list get pointer (TGPT)
Event:
- Message buffer 8, 5, 6, and 10 completes transmission.
- THL is full.
- TOVF is set.
Transmit history list (THL)
7 Message buffer 5
6 Message buffer 8
5 Message buffer 4
4 Message buffer 3
3 Message buffer 7
2
1 Message buffer 10
0 Message buffer 6
Last out-message pointer (LOPT)
Transmit history list get pointer (TGPT)
TOVF = 1
LOPT is blocked
Event:
- Message buffer11, 13, and 14 completes transmission.
- Overflow situation occurs.
Transmit history list (THL)
7 Message buffer 5
6 Message buffer 8
5 Message buffer 4
4 Message buffer 3
3 Message buffer 7
2
1 Message buffer 14
0 Message buffer 6
Last out-message pointer (LOPT)
Transmit history list get pointer (TGPT)
TOVF = 1
LOPT is blocked
TOVF = 1 denotes that LOPT equals TGPT - 1 while message buffer number stored to element indicated by LOPT - 1.

Figure 20-31 Transmit history list

20.10.3 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 MT[2:0] bits to 000_B. Be sure to set the same ID for the message buffers for ABT even when that ID is being used for all the message buffers. To use two or more IDs, set the ID of each message buffer by using the CnMIDLm and CnMIDHm registers. Set the CnMDLCm and CnMDATA0m to CnMDATA7m registers before issuing a transmission request for the ABT function.

After initialization of message buffers for ABT is finished, the RDY bit needs to be set (1). In the ABT mode, the TRQ bit does not have to be manipulated by software.

After the data for the ABT message buffers has been prepared, set the ABTTRG bit to 1. Automatic block transmission is then started. When ABT is started, the TRQ bit in the first message buffer (message buffer 0) is automatically set to 1. After transmission of the data of message buffer 0 is finished, the TRQ bit of the next message buffer, message buffer 1, is set automatically. In this way, transmission is executed successively.

A delay time can be inserted by program in the interval in which the transmission request (TRQ) is automatically set while successive transmission is being executed. The delay time to be inserted is defined by the CnGMABTD register. The unit of the delay time is DBT (data bit time). DBT depends on the setting of the CnBRP and CnBTR registers.

Among transmit objects within the ABT-area, the priority of the transmission ID is not evaluated. The data of message buffers 0 to 7 are sequentially transmitted. When transmission of the data frame from message buffer 7 has been completed, the ABTTRG bit is automatically cleared to 0 and the ABT operation is finished.

If the RDY bit of an ABT message buffer is cleared during ABT, no data frame is transmitted from that buffer, ABT is stopped, and the ABTTRG bit is cleared. After that, transmission can be resumed from the message buffer where ABT stopped, by setting the RDY and ABTTRG bits to 1 by software. To not resume transmission from the message buffer where ABT stopped, the internal ABT engine can be reset by setting the ABTCLR bit to 1 while ABT mode is stopped and the ABTTRG bit is cleared to 0. In this case, transmission is started from message buffer 0 if the ABTCLR bit is cleared to 0 and then the ABTTRG bit is set to 1.

An interrupt can be used to check if data frames have been transmitted from all the message buffers for ABT. To do so, the IE bit of the CnMCTRLm register of each message buffer except the last message buffer needs to be cleared (0).

If a transmit message buffer other than those used by the ABT function (message buffers 8 to 31) is assigned to a transmit message buffer, the message to be transmitted next is determined by the priority of the transmission ID of the ABT message buffer whose transmission is currently held pending and the transmission ID of the message buffers other than those used by the ABT function.

Transmission of a data frame from an ABT message buffer is not recorded in the transmit history list (THL).

Caution

1. Set the ABTCLR bit to 1 while the ABTTRG bit is cleared to 0 in order to resume ABT operation at buffer No.0. If the ABTCLR bit is set to 1 while the ABTTRG bit is set to 1, the subsequent operation is not guaranteed.
2. If the automatic block transmission engine is cleared by setting the ABTCLR bit to 1, the ABTCLR bit is automatically cleared immediately after the processing of the clearing request is completed.
3. Do not set the ABTTRG bit in the initialization mode. If the ABTTRG bit is set in the initialization mode, the proper operation is not guaranteed after the mode is changed from the initialization mode to the ABT mode.
4. Do not set the TRQ bit of the ABT message buffers to 1 by software in the normal operation mode with ABT. Otherwise, the operation is not guaranteed.
5. The CnGMABTD register is used to set the delay time that is inserted in the period from completion of the preceding ABT message to setting of the TRQ bit for the next ABT message when the transmission requests are set in the order of message numbers for each message for ABT that is successively transmitted in the ABT mode. The timing at which the messages are actually transmitted onto the CAN bus varies depending on the status of transmission from other stations and the status of the setting of the transmission request for messages other than the ABT messages (message buffers 8 to 31).
6. If a transmission request is made for a message other than an ABT message and if no delay time is inserted in the interval in which transmission requests for ABT are automatically set (CnGMABTD register = 00_H), messages other than ABT messages may be transmitted not depending on their priority compared to the priority of the ABT message.
7. Do not clear the RDY bit to 0 when the ABTTRG bit = 1.
8. If a message is received from another node while normal operation mode with ABT is active, the TX-message from the ABT-area may be transmitted with delay of one frame although CnGMABTD register was set up with 00_H.

20.10.4 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 20-45 on page 801*).

(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 20-46 on page 802*).

(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 20-47 on page 803*). 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 20-48 on page 804*).

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.

20.10.5 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.

20.11 Power Saving Modes

20.11.1 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 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.

Caution

1. Be aware that the release of CAN sleep mode by CAN bus event, and thus the wake up interrupt may happen at any time, even right after requesting sleep mode, if a CAN bus event occurs.
2. Always reset the PSMODE[1:0] bits to 00_B, when waking up from CAN sleep mode, before accessing any other registers of the CAN module.
3. Always clear the interrupt flag CINTS5, when waking up from CAN sleep mode.

20.11.2 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.

20.11.3 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).

20.12 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 20-33 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).

20.13 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.

20.13.1 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).

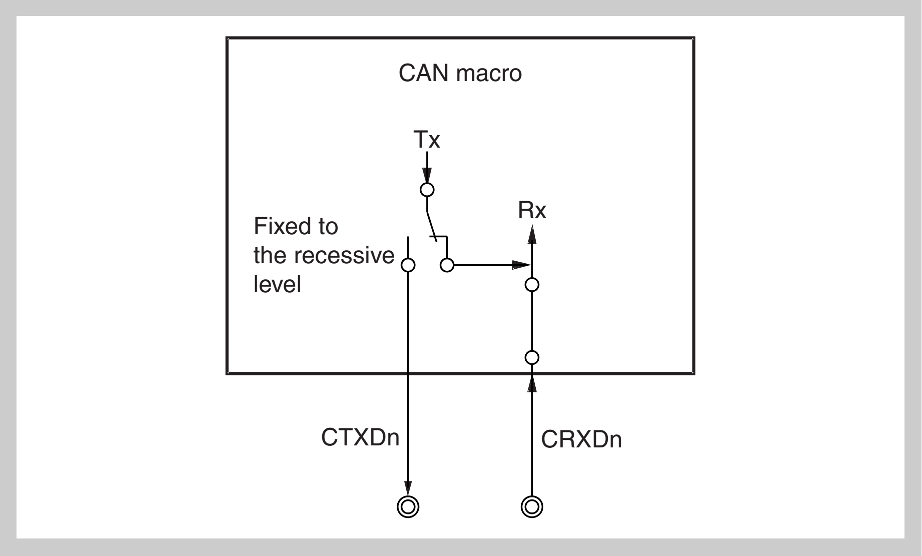


Figure 20-32 CAN module terminal connection in receive-only mode

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.

20.13.2 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.

20.13.3 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.

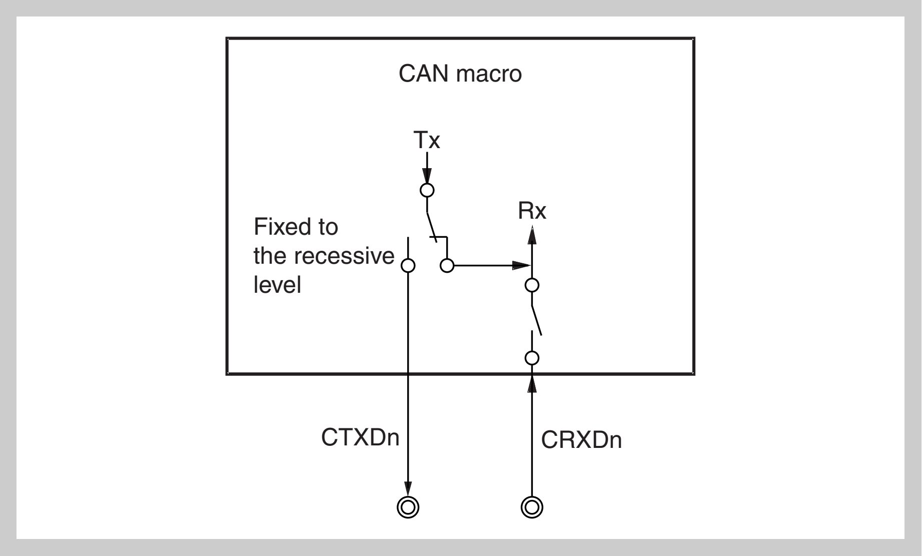


Figure 20-33 CAN module terminal connection in self-test mode

20.13.4 Receive/transmit operation in each operation mode

The following table shows outline of the receive/transmit operation in each operation mode.

Table 20-34 Outline of the receive/transmit in each operation mode

Operation mode	Transmission of data/ remote frame	Transmission of ACK	Transmission of error/ overload frame	Transmission retry	Automatic block transmission (ABT)	Set of VALID bit	Store data to message buffer
Initialization mode	No	No	No	No	No	No	No
Normal operation mode	Yes	Yes	Yes	Yes	No	Yes	Yes
Normal operation mode with ABT	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Receive only mode	No	No	No	No	No	Yes	Yes
Single-shot mode	Yes	Yes	Yes	No[a]	No	Yes	Yes
Self-test mode	Yes[b]	Yes[b]	Yes[b]	Yes[b]	No	Yes[b]	Yes[b]

[a] When the arbitration lost occurs, control of re-transmission is possible by the AL bit of CnCTRL register.

[b] Each signals are not generated to outside, but generated into the CAN module.

20.14 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.

20.14.1 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.

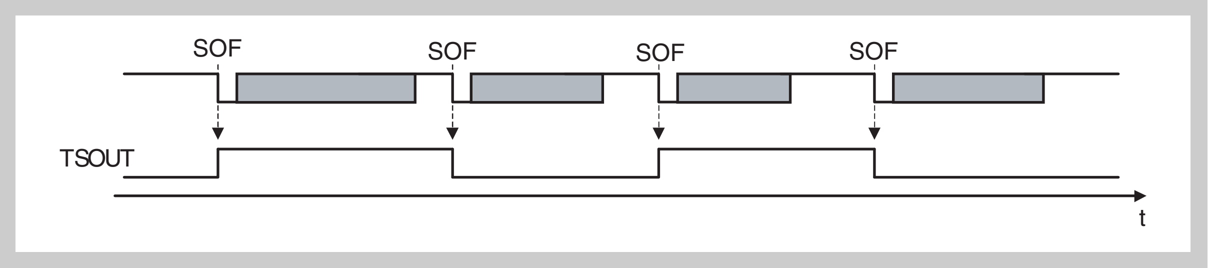


Figure 20-34 Timing diagram of capture signal TSOUT

The TSOUT signal toggles its level upon occurrence of the selected event during data frame reception (in *Figure 20-34*, 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.

20.15 Baud Rate Settings

20.15.1 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 20-35 shows the combinations of bit rates that satisfy the above conditions.

Table 20-35 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 20-35 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 20-35 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 20-35* 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.

20.15.2 Representative examples of baud rate settings

Table 20-36 and *Table 20-37* show representative examples of baud rate settings.

Table 20-36 Representative examples of baud rate settings (f_{CANMOD} = 8 MHz) (1/2)

Set baud rate value (unit: kbps)	Division ratio of CnBRP register	CnBRP register set value	Valid bit rate setting (unit: kbps)					CnBTR register setting value		Sampling point (unit: %)
			Length of DBT	SYNC SEGMENT	PROP SEGMENT	PHASE SEGMENT1	PHASE SEGMENT2	TSEG1 [3:0]	TSEG2 [2:0]	
1000	1	00000000	8	1	1	3	3	0011	010	62.5
1000	1	00000000	8	1	3	2	2	0100	001	75.0
1000	1	00000000	8	1	5	1	1	0101	000	87.5
500	1	00000000	16	1	1	7	7	0111	110	56.3
500	1	00000000	16	1	3	6	6	1000	101	62.5
500	1	00000000	16	1	5	5	5	1001	100	68.8
500	1	00000000	16	1	7	4	4	1010	011	75.0
500	1	00000000	16	1	9	3	3	1011	010	81.3
500	1	00000000	16	1	11	2	2	1100	001	87.5
500	1	00000000	16	1	13	1	1	1101	000	93.8
500	2	00000001	8	1	1	3	3	0011	010	62.5
500	2	00000001	8	1	3	2	2	0100	001	75.0
500	2	00000001	8	1	5	1	1	0101	000	87.5
250	2	00000001	16	1	1	7	7	0111	110	56.3
250	2	00000001	16	1	3	6	6	1000	101	62.5
250	2	00000001	16	1	5	5	5	1001	100	68.8
250	2	00000001	16	1	7	4	4	1010	011	75.0
250	2	00000001	16	1	9	3	3	1011	010	81.3
250	2	00000001	16	1	11	2	2	1100	001	87.5
250	2	00000001	16	1	13	1	1	1101	000	93.8
250	4	00000011	8	1	3	2	2	0100	001	75.0
250	4	00000011	8	1	5	1	1	0101	000	87.5
125	4	00000011	16	1	1	7	7	0111	110	56.3
125	4	00000011	16	1	3	6	6	1000	101	62.5
125	4	00000011	16	1	5	5	5	1001	100	68.8
125	4	00000011	16	1	7	4	4	1010	011	75.0
125	4	00000011	16	1	9	3	3	1011	010	81.3
125	4	00000011	16	1	11	2	2	1100	001	87.5
125	4	00000011	16	1	13	1	1	1101	000	93.8
125	8	00000111	8	1	3	2	2	0100	001	75.0
125	8	00000111	8	1	5	1	1	0101	000	87.5
100	4	00000011	20	1	7	6	6	1100	101	70.0
100	4	00000011	20	1	9	5	5	1101	100	75.0
100	5	00000100	16	1	7	4	4	1010	011	75.0
100	5	00000100	16	1	9	3	3	1011	010	81.3

Table 20-36 Representative examples of baud rate settings (f_{CANMOD} = 8 MHz) (2/2)

Set baud rate value (unit: kbps)	Division ratio of CnBRP register	CnBRP register set value	Valid bit rate setting (unit: kbps)					CnBTR register setting value		Sampling point (unit: %)
			Length of DBT	SYNC SEGMENT	PROP SEGMENT	PHASE SEGMENT1	PHASE SEGMENT2	TSEG1 [3:0]	TSEG2 [2:0]	
100	8	00000111	10	1	3	3	3	0101	010	70.0
100	8	00000111	10	1	5	2	2	0110	001	80.0
100	10	00001001	8	1	3	2	2	0100	001	75.0
100	10	00001001	8	1	5	1	1	0101	000	87.5
83.3	4	00000011	24	1	7	8	8	1110	111	66.7
83.3	4	00000011	24	1	9	7	7	1111	110	70.8
83.3	6	00000101	16	1	5	5	5	1001	100	68.8
83.3	6	00000101	16	1	7	4	4	1010	011	75.0
83.3	6	00000101	16	1	9	3	3	1011	010	81.3
83.3	6	00000101	16	1	11	2	2	1100	001	87.5
83.3	8	00000111	12	1	5	3	3	0111	010	75.0
83.3	8	00000111	12	1	7	2	2	1000	001	83.3
83.3	12	00001011	8	1	3	2	2	0100	001	75.0
83.3	12	00001011	8	1	5	1	1	0101	000	87.5
33.3	10	00001001	24	1	7	8	8	1110	111	66.7
33.3	10	00001001	24	1	9	7	7	1111	110	70.8
33.3	12	00001011	20	1	7	6	6	1100	101	70.0
33.3	12	00001011	20	1	9	5	5	1101	100	75.0
33.3	15	00001110	16	1	7	4	4	1010	011	75.0
33.3	15	00001110	16	1	9	3	3	1011	010	81.3
33.3	16	00001111	15	1	6	4	4	1001	011	73.3
33.3	16	00001111	15	1	8	3	3	1010	010	80.0
33.3	20	00010011	12	1	5	3	3	0111	010	75.0
33.3	20	00010011	12	1	7	2	2	1000	001	83.3
33.3	24	00010111	10	1	3	3	3	0101	010	70.0
33.3	24	00010111	10	1	5	2	2	0110	001	80.0
33.3	30	00011101	8	1	3	2	2	0100	001	75.0
33.3	30	00011101	8	1	5	1	1	0101	000	87.5

Caution The values in *Table 20-36* 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 20-37 Representative examples of baud rate settings (f_{CANMOD} = 16 MHz) (1/2)

Set baud rate value (unit: kbps)	Division ratio of CnBRP register	CnBRP register set value	Valid bit rate setting (unit: kbps)					CnBTR register setting value		Sampling point (unit: %)
			Length of DBT	SYNC SEGMENT	PROP SEGMENT	PHASE SEGMENT1	PHASE SEGMENT2	TSEG1 [3:0]	TSEG2 [2:0]	
1000	1	00000000	16	1	1	7	7	0111	110	56.3
1000	1	00000000	16	1	3	6	6	1000	101	62.5
1000	1	00000000	16	1	5	5	5	1001	100	68.8
1000	1	00000000	16	1	7	4	4	1010	011	75.0
1000	1	00000000	16	1	9	3	3	1011	010	81.3
1000	1	00000000	16	1	11	2	2	1100	001	87.5
1000	1	00000000	16	1	13	1	1	1101	000	93.8
1000	2	00000001	8	1	3	2	2	0100	001	75.0
1000	2	00000001	8	1	5	1	1	0101	000	87.5
500	2	00000001	16	1	1	7	7	0111	110	56.3
500	2	00000001	16	1	3	6	6	1000	101	62.5
500	2	00000001	16	1	5	5	5	1001	100	68.8
500	2	00000001	16	1	7	4	4	1010	011	75.0
500	2	00000001	16	1	9	3	3	1011	010	81.3
500	2	00000001	16	1	11	2	2	1100	001	87.5
500	2	00000001	16	1	13	1	1	1101	000	93.8
500	4	00000011	8	1	3	2	2	0100	001	75.0
500	4	00000011	8	1	5	1	1	0101	000	87.5
250	4	00000011	16	1	3	6	6	1000	101	62.5
250	4	00000011	16	1	5	5	5	1001	100	68.8
250	4	00000011	16	1	7	4	4	1010	011	75.0
250	4	00000011	16	1	9	3	3	1011	010	81.3
250	4	00000011	16	1	11	2	2	1100	001	87.5
250	8	00000111	8	1	3	2	2	0100	001	75.0
250	8	00000111	8	1	5	1	1	0101	000	87.5
125	8	00000111	16	1	3	6	6	1000	101	62.5
125	8	00000111	16	1	7	4	4	1010	011	75.0
125	8	00000111	16	1	9	3	3	1011	010	81.3
125	8	00000111	16	1	11	2	2	1100	001	87.5
125	16	00001111	8	1	3	2	2	0100	001	75.0
125	16	00001111	8	1	5	1	1	0101	000	87.5
100	8	00000111	20	1	9	5	5	1101	100	75.0
100	8	00000111	20	1	11	4	4	1110	011	80.0
100	10	00001001	16	1	7	4	4	1010	011	75.0
100	10	00001001	16	1	9	3	3	1011	010	81.3
100	16	00001111	10	1	3	3	3	0101	010	70.0
100	16	00001111	10	1	5	2	2	0110	001	80.0
100	20	00010011	8	1	3	2	2	0100	001	75.0

Table 20-37 Representative examples of baud rate settings (f_{CANMOD} = 16 MHz) (2/2)

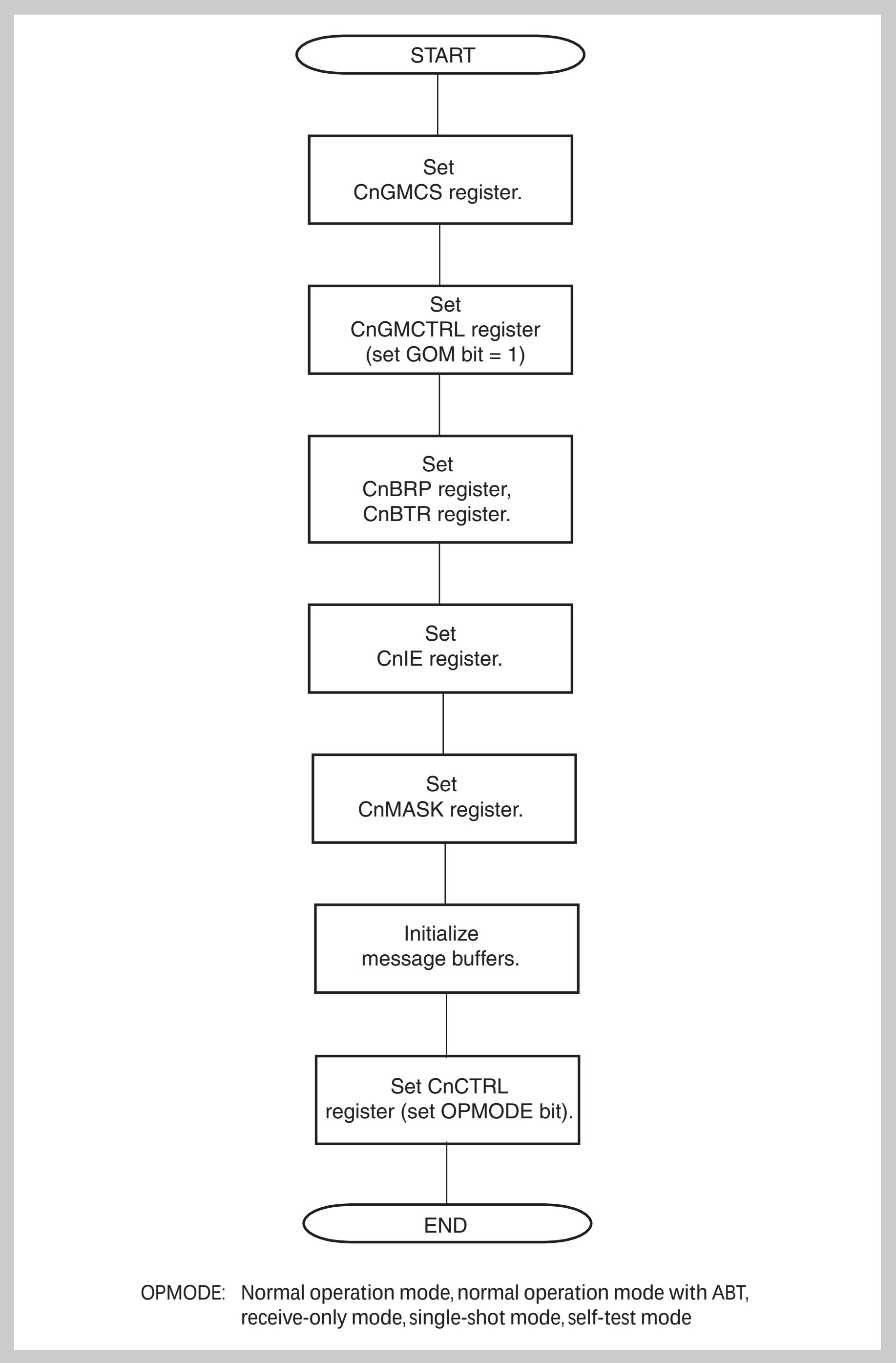
Set baud rate value (unit: kbps)	Division ratio of CnBRP register	CnBRP register set value	Valid bit rate setting (unit: kbps)					CnBTR register setting value		Sampling point (unit: %)
			Length of DBT	SYNC SEGMENT	PROP SEGMENT	PHASE SEGMENT1	PHASE SEGMENT2	TSEG1 [3:0]	TSEG2 [2:0]	
83.3	8	00000111	24	1	7	8	8	1110	111	66.7
83.3	8	00000111	24	1	9	7	7	1111	110	70.8
83.3	12	00001011	16	1	7	4	4	1010	011	75.0
83.3	12	00001011	16	1	9	3	3	1011	010	81.3
83.3	12	00001011	16	1	11	2	2	1100	001	87.5
83.3	16	00001111	12	1	5	3	3	0111	010	75.0
83.3	16	00001111	12	1	7	2	2	1000	001	83.3
83.3	24	00010111	8	1	3	2	2	0100	001	75.0
83.3	24	00010111	8	1	5	1	1	0101	000	87.5
33.3	30	00011101	24	1	7	8	8	1110	111	66.7
33.3	30	00011101	24	1	9	7	7	1111	110	70.8
33.3	24	00010111	20	1	9	5	5	1101	100	75.0
33.3	24	00010111	20	1	11	4	4	1110	011	80.0
33.3	30	00011101	16	1	7	4	4	1010	011	75.0
33.3	30	00011101	16	1	9	3	3	1011	010	81.3
33.3	32	00011111	15	1	8	3	3	1010	010	80.0
33.3	32	00011111	15	1	10	2	2	1011	001	86.7
33.3	37	00100100	13	1	6	3	3	1000	010	76.9
33.3	37	00100100	13	1	8	2	2	1001	001	84.6
33.3	40	00100111	12	1	5	3	3	0111	010	75.0
33.3	40	00100111	12	1	7	2	2	1000	001	83.3
33.3	48	00101111	10	1	3	3	3	0101	010	70.0
33.3	48	00101111	10	1	5	2	2	0110	001	80.0
33.3	60	00111011	8	1	3	2	2	0100	001	75.0
33.3	60	00111011	8	1	5	1	1	0101	000	87.5

Caution The values in *Table 20-37* 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.

20.16 Operation of CAN Controller

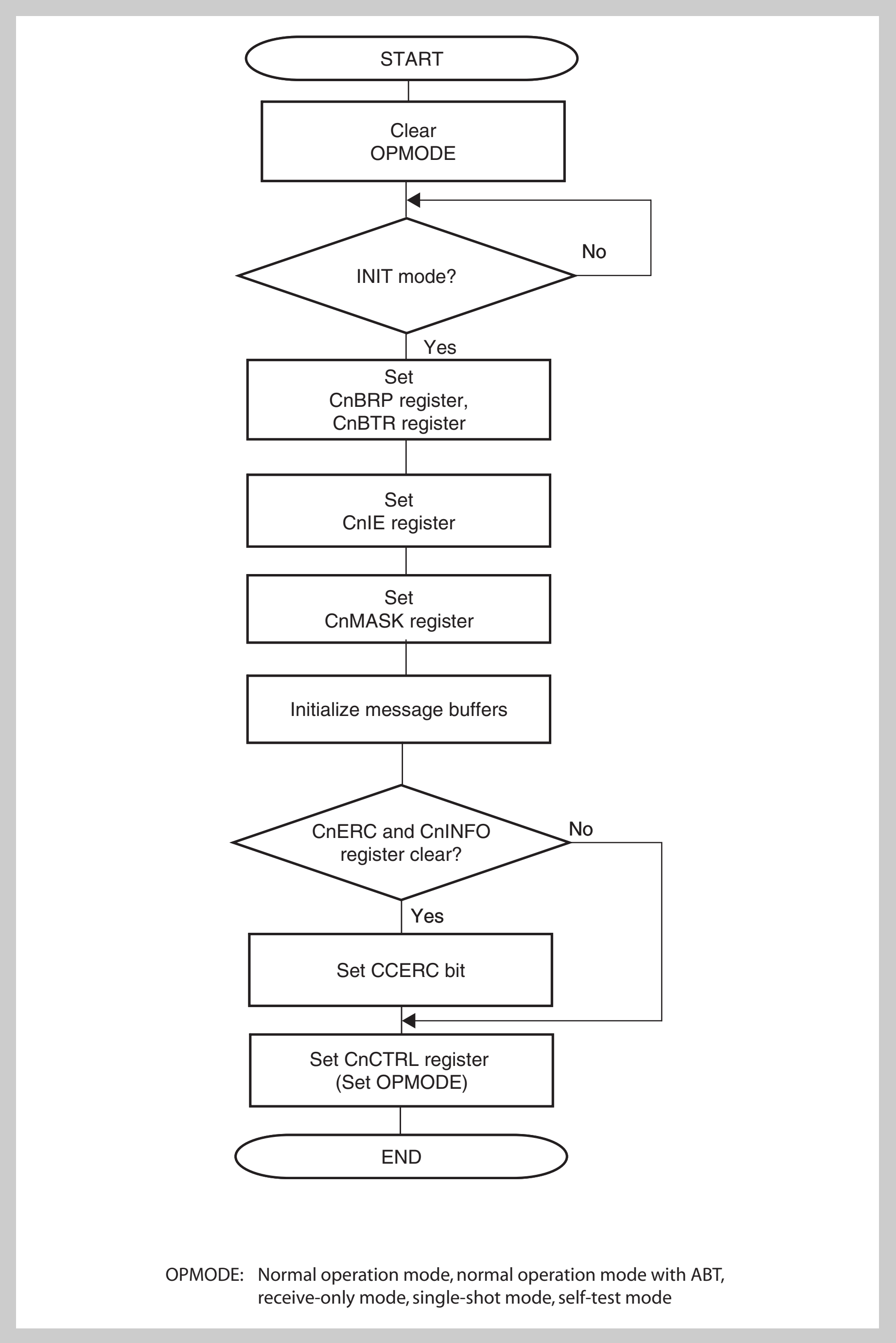
The processing procedure for showing in this chapter is recommended processing procedure to operate CAN controller.

Develop the program referring to recommended processing procedure in this chapter.



OPMODE: Normal operation mode, normal operation mode with ABT, receive-only mode, single-shot mode, self-test mode

Figure 20-35 Initialization



OPMODE: Normal operation mode, normal operation mode with ABT, receive-only mode, single-shot mode, self-test mode

Figure 20-36 Re-initialization

Caution After setting the CAN module to the initialization mode, avoid setting the module to another operation mode immediately after. If it is necessary to immediately set the module to another operation mode, be sure to access registers other than the CnCTRL and CnGMCTRL registers (e.g., set a message buffer).

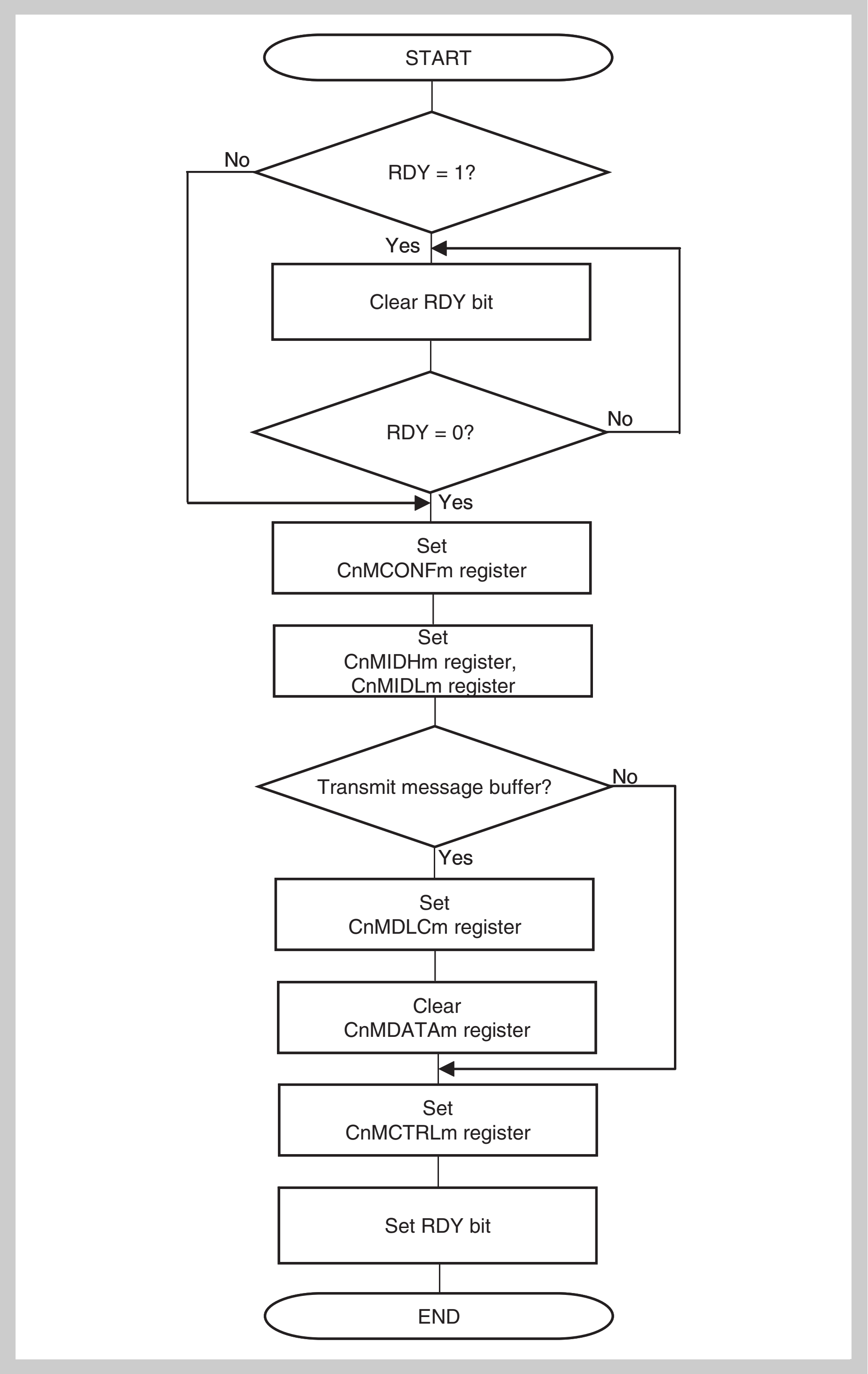
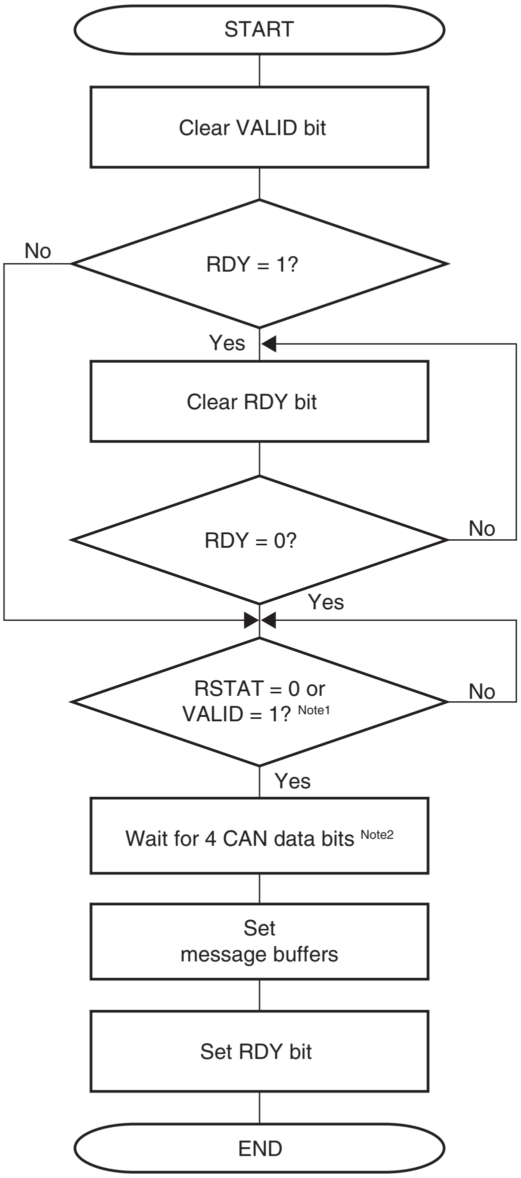


Figure 20-37 Message buffer initialization

Caution

1. Before a message buffer is initialized, the RDY bit must be cleared.
2. Make the following settings for message buffers not used by the application.
   - Clear the RDY, TRQ, and DN bits of the CnMCTRLm register to 0.
   - Clear the MA0 bit of the CnMCONFm register to 0.

Figure 20-38 shows the processing for a receive message buffer (MT[2:0] bits of CnMCONFm register = 001_B to 101_B).



Note1: Confirm that a message is being received because RDY bit must be set after a message is completely received.

Note2: Avoid message buffer redefinition during store operation of message reception by waiting additional 4 CAN data bits.

Figure 20-38 Message buffer redefinition

Figure 20-39 shows the processing for a transmit message buffer during transmission (MT[2:0] bits of CnMCONFm register = 000_B).

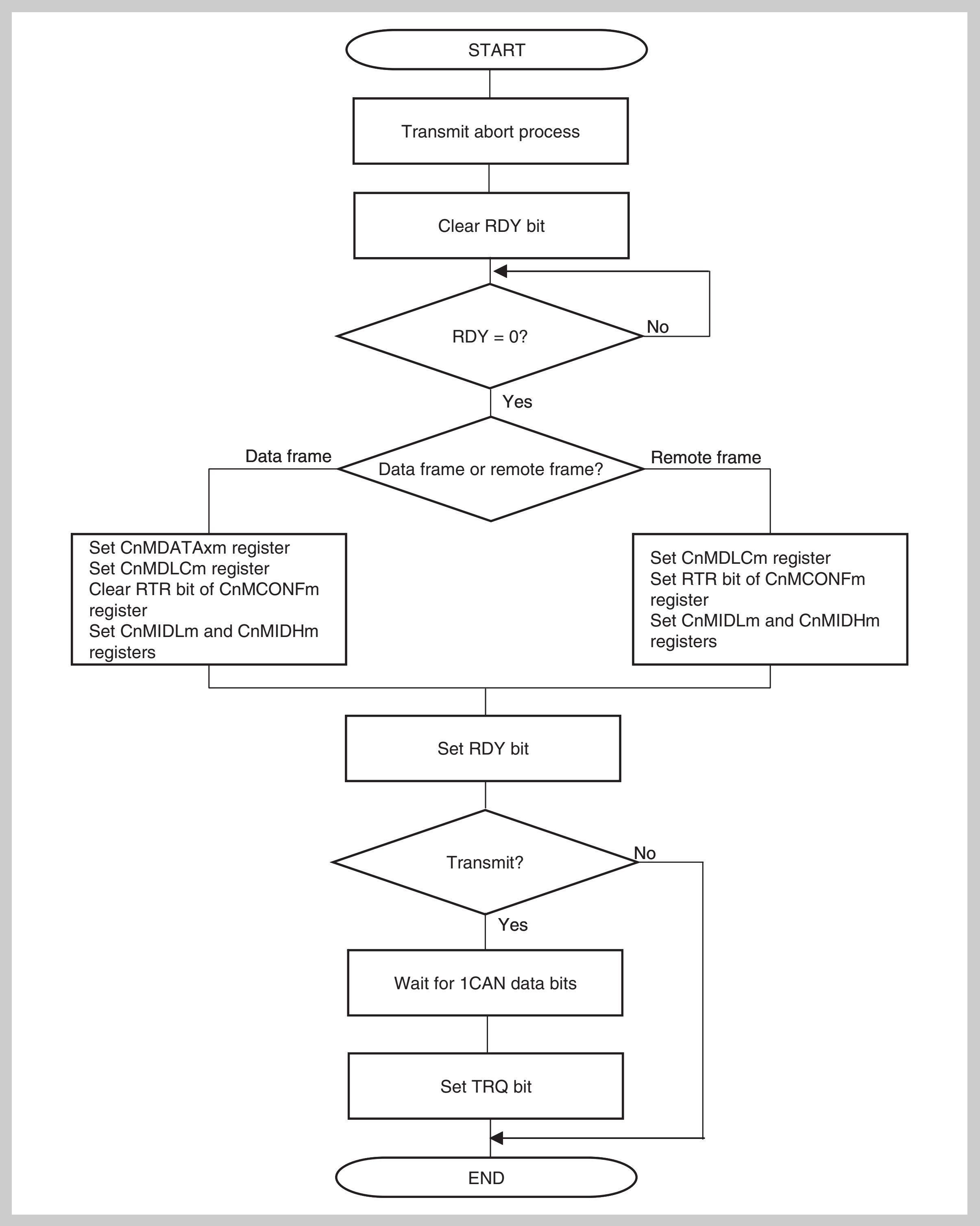


Figure 20-39 Message buffer redefinition during transmission

Figure 20-40 shows the processing for a transmit message buffer (MT[2:0] bits of CnMCONFm register = 000_B).

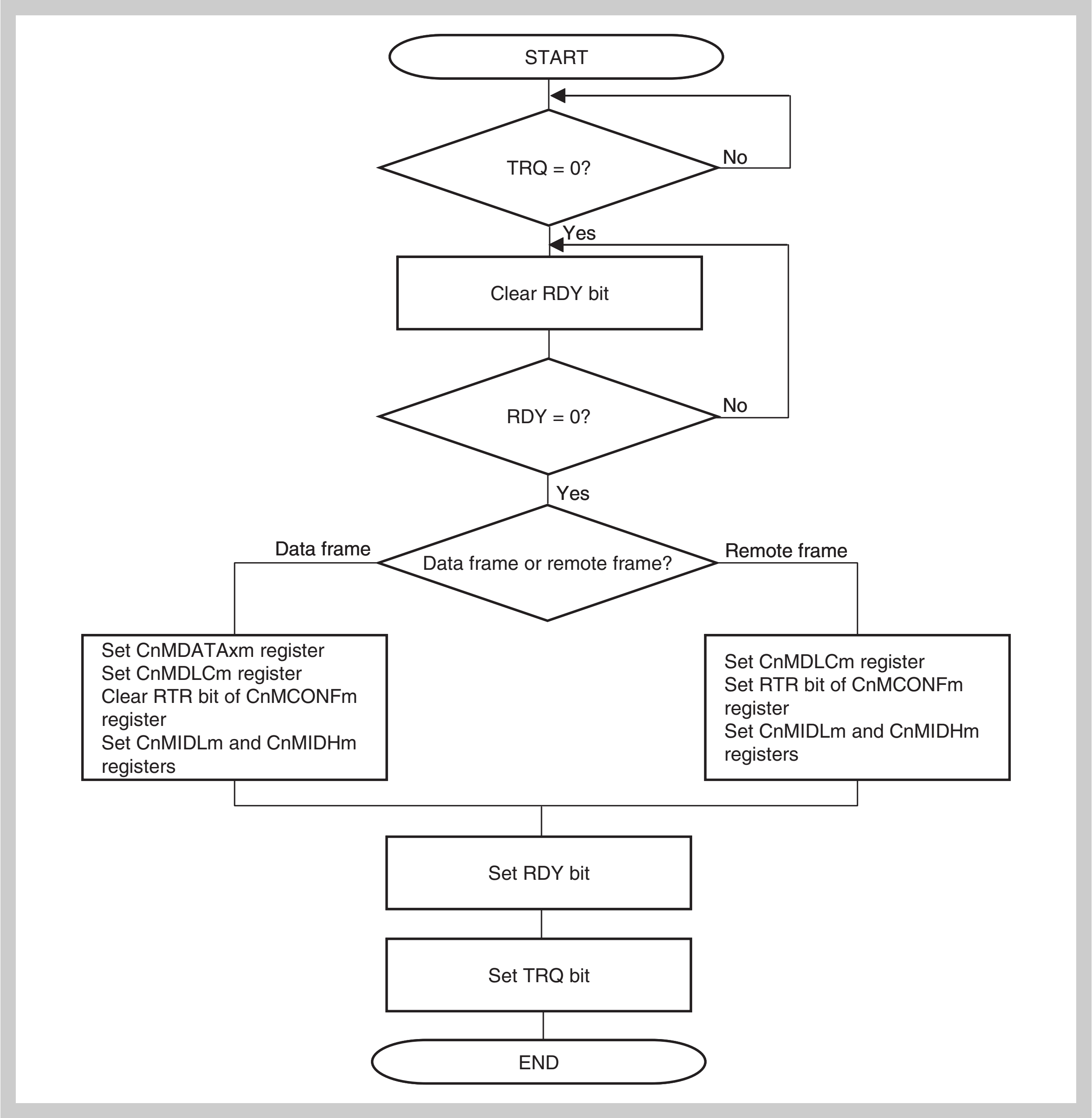

START
TRQ = 0?
No
Yes
Clear RDY bit
RDY = 0?
No
Yes
Data frame
Data frame or remote frame?
Remote frame
Set CnMDATAxm register
Set CnMDLCm register
Clear RTR bit of CnMCONFm register
Set CnMIDLm and CnMIDHm registers
Set CnMDLCm register
Set RTR bit of CnMCONFm register
Set CnMIDLm and CnMIDHm registers
Set RDY bit
Set TRQ bit
END

Figure 20-40 Message transmit processing

Caution

1. The TRQ bit should be set after the RDY bit is set.
2. The RDY bit and TRQ bit should not be set at the same time.

Figure 20-41 shows the processing for a transmit message buffer (MT[2:0] bits of CnMCONFm register = 000_B)

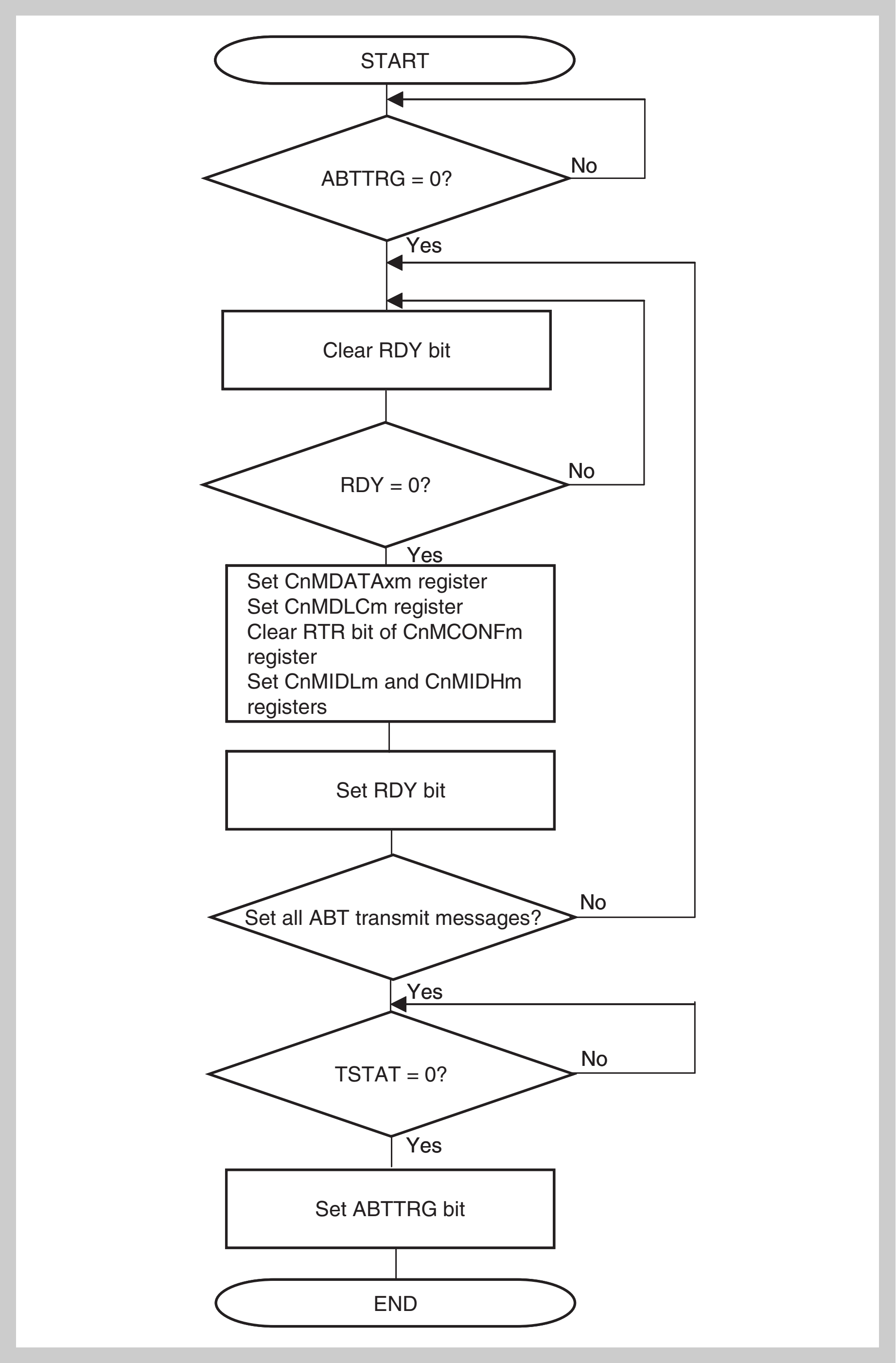


Figure 20-41 ABT message transmit processing

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 20-40 on page 796*.

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.

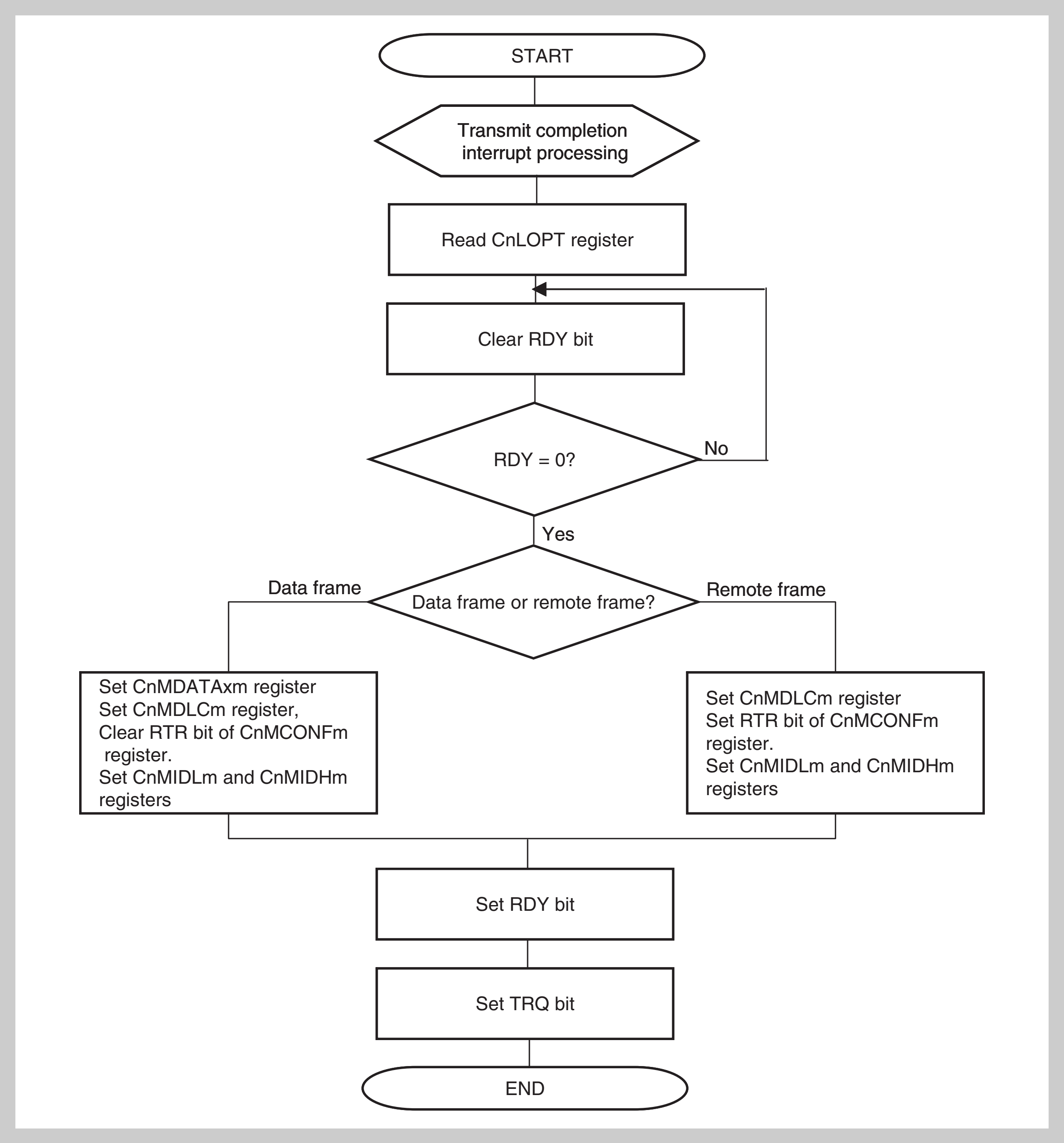

START
Transmit completion interrupt processing
Read CnLOPT register
Clear RDY bit
RDY = 0?
No
Yes
Data frame or remote frame?
Data frame
Remote frame
Set CnMDATAxm register
Set CnMDLCm register,
Clear RTR bit of CnMCONFm register.
Set CnMIDLm and CnMIDHm registers
Set CnMDLCm register
Set RTR bit of CnMCONFm register.
Set CnMIDLm and CnMIDHm registers
Set RDY bit
Set TRQ bit
END

Figure 20-42 Transmission via interrupt (using CnLOPT register)

Caution

1. The TRQ bit should be set after the RDY bit is set.
2. The RDY bit and TRQ bit should not be set at the same time.

Note Also check the MBON flag at the beginning and at the end of the interrupt routine, in order to check the access to the message buffers as well as TX history list registers, in case a pending sleep mode had been executed. If MBON is detected to be cleared at any check, the actions and results of the processing have to be discarded and processed again, after MBON is set again.
It is recommended to cancel any sleep mode requests, before processing TX interrupts.

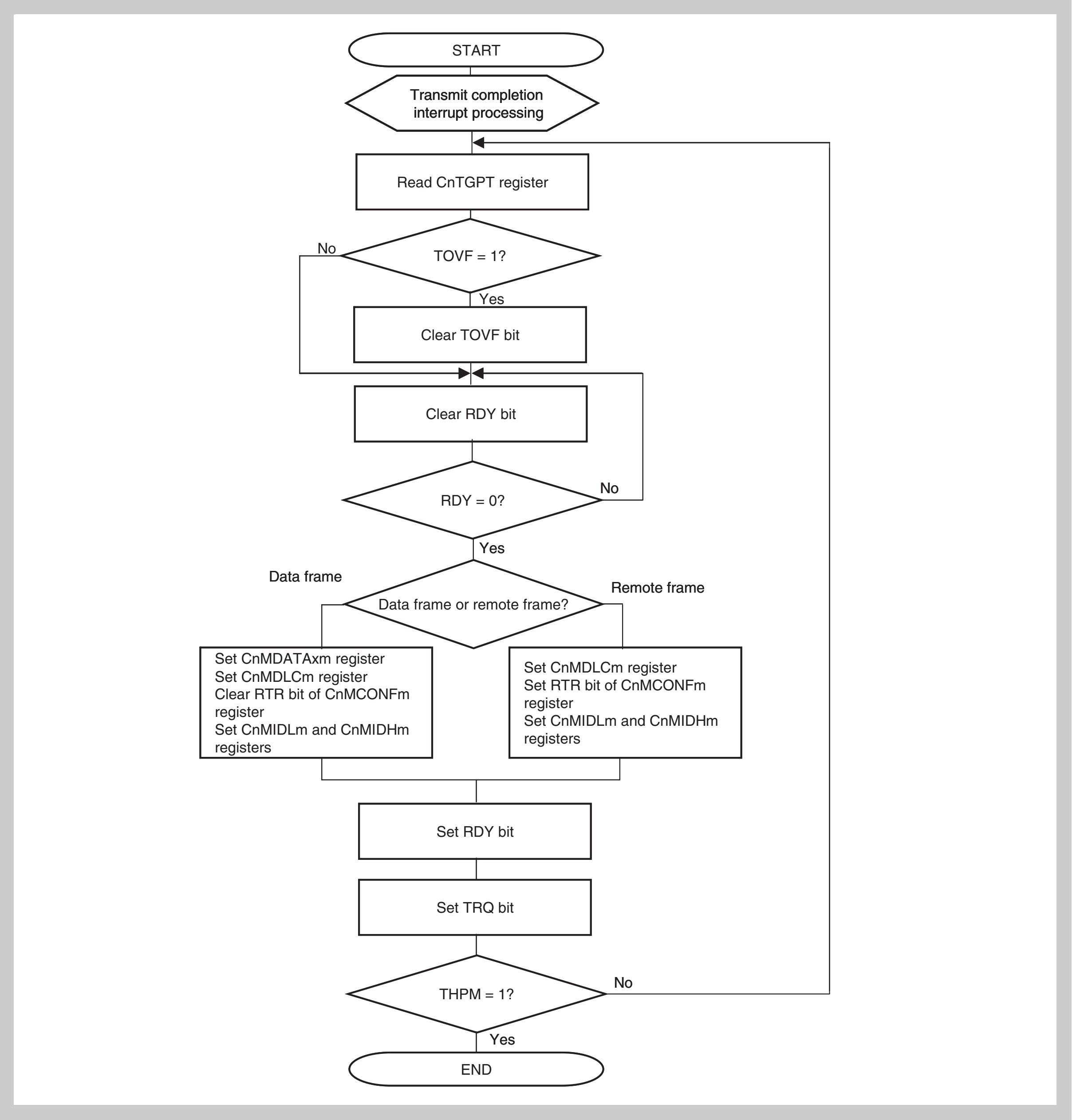


Figure 20-43 Transmission via interrupt (using CnTGPT register)

Caution 1. The TRQ bit should be set after the RDY bit is set.

2. The RDY bit and TRQ bit should not be set at the same time.

Note 1. Also check the MBON flag at the beginning and at the end of the interrupt routine, in order to check the access to the message buffers as well as TX history list registers, in case a pending sleep mode had been executed. If MBON is detected to be cleared at any check, the actions and results of the processing have to be discarded and processed again, after MBON is set again.
It is recommended to cancel any sleep mode requests, before processing TX interrupts.

2. If TOVF was set once, the transmit history list is inconsistent. Consider to scan all configured transmit buffers for completed transmissions.

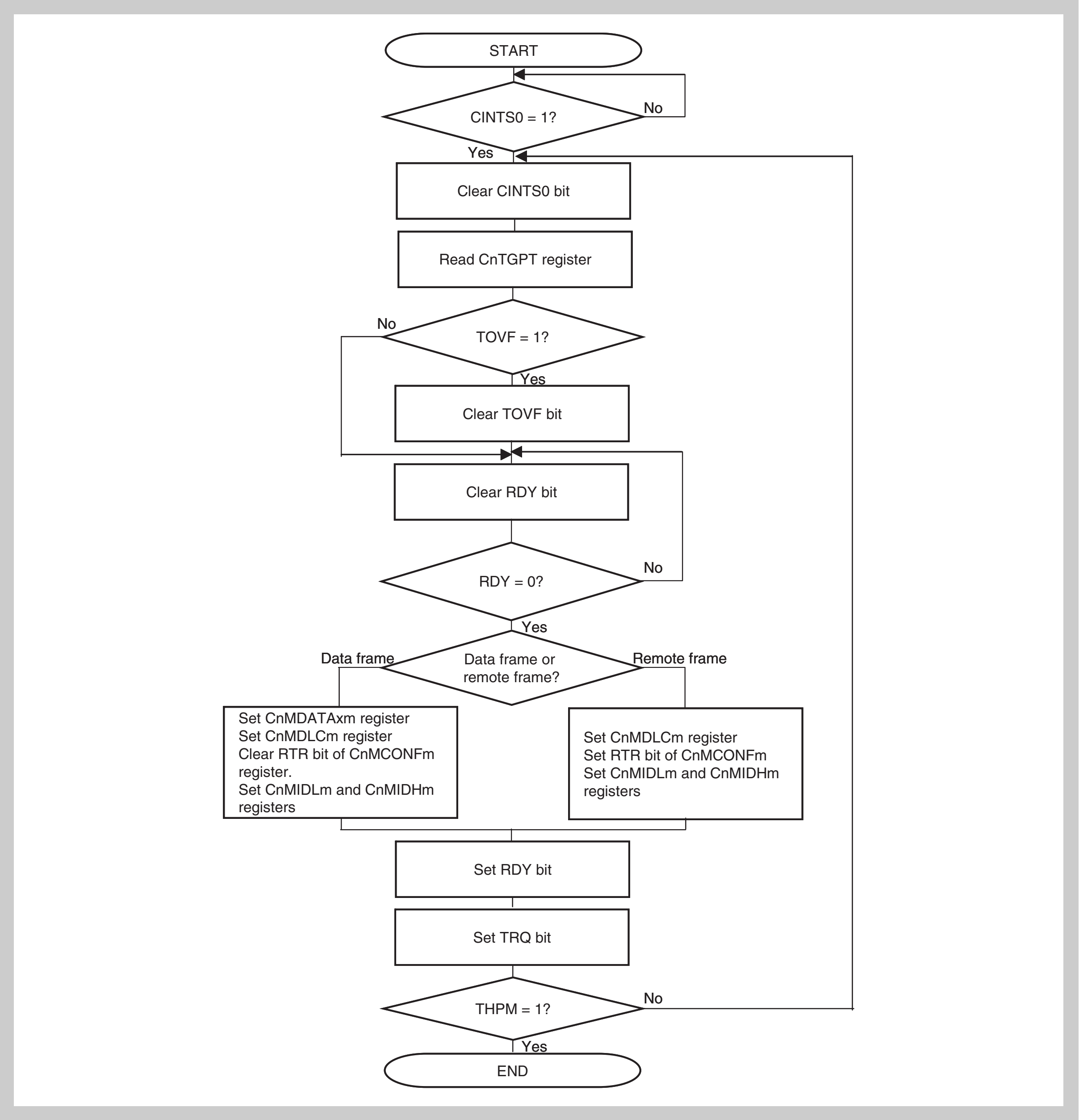


Figure 20-44 Transmission via software polling

Caution 1. The TRQ bit should be set after the RDY bit is set.

2. The RDY bit and TRQ bit should not be set at the same time.

Note 1. Also check the MBON flag at the beginning and at the end of the polling routine, in order to check the access to the message buffers as well as TX history list registers, in case a pending sleep mode had been executed. If MBON is detected to be cleared at any check, the actions and results of the processing have to be discarded and processed again, after MBON is set again.

2. If TOVF was set once, the transmit history list is inconsistent. Consider to scan all configured transmit buffers for completed transmissions.

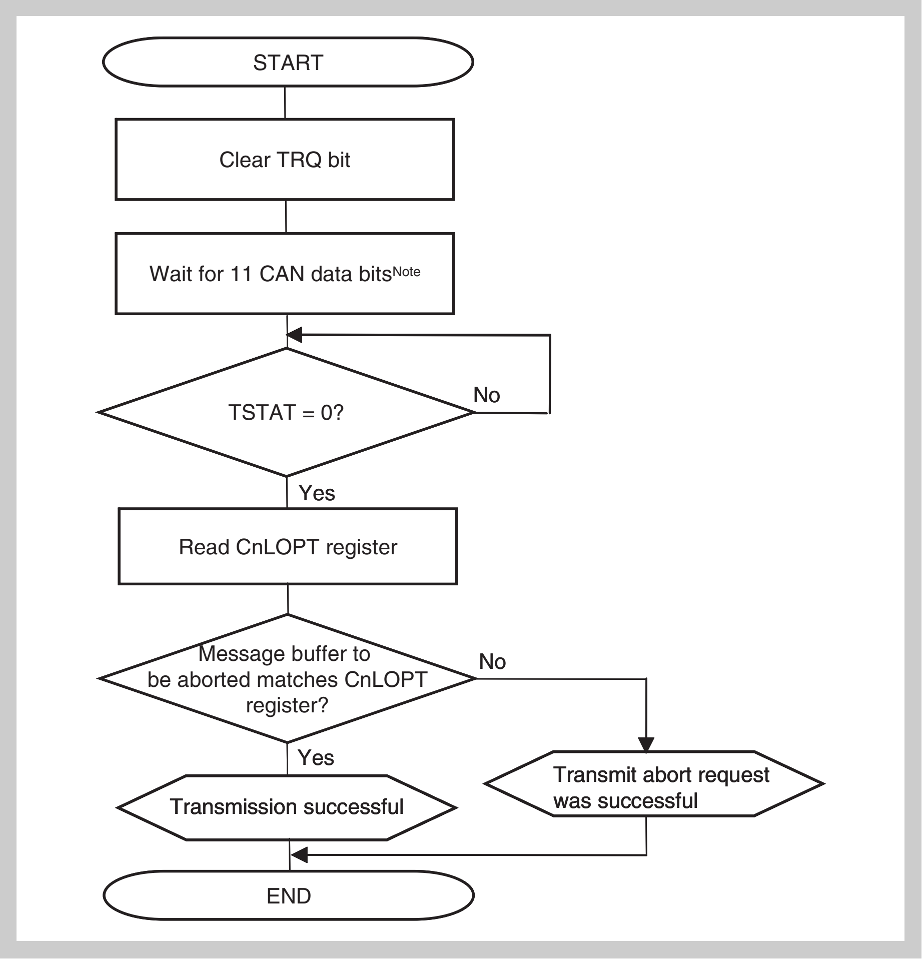

START
Clear TRQ bit
Wait for 11 CAN data bitsNote
TSTAT = 0?
No
Yes
Read CnLOPT register
Message buffer to be aborted matches CnLOPT register?
No
Yes
Transmission successful
Transmit abort request was successful
END

Figure 20-45 **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).

Caution

1. Clear the TRQ bit for aborting transmission request, not the RDY bit.
2. Before making a sleep mode transition request, confirm that there is no transmission request left using this processing.
3. The TSTAT bit can be periodically checked by a user application or can be checked after the transmit completion interrupt.
4. Do not execute any new transmission request including in the other message buffers while transmission abort processing is in progress.

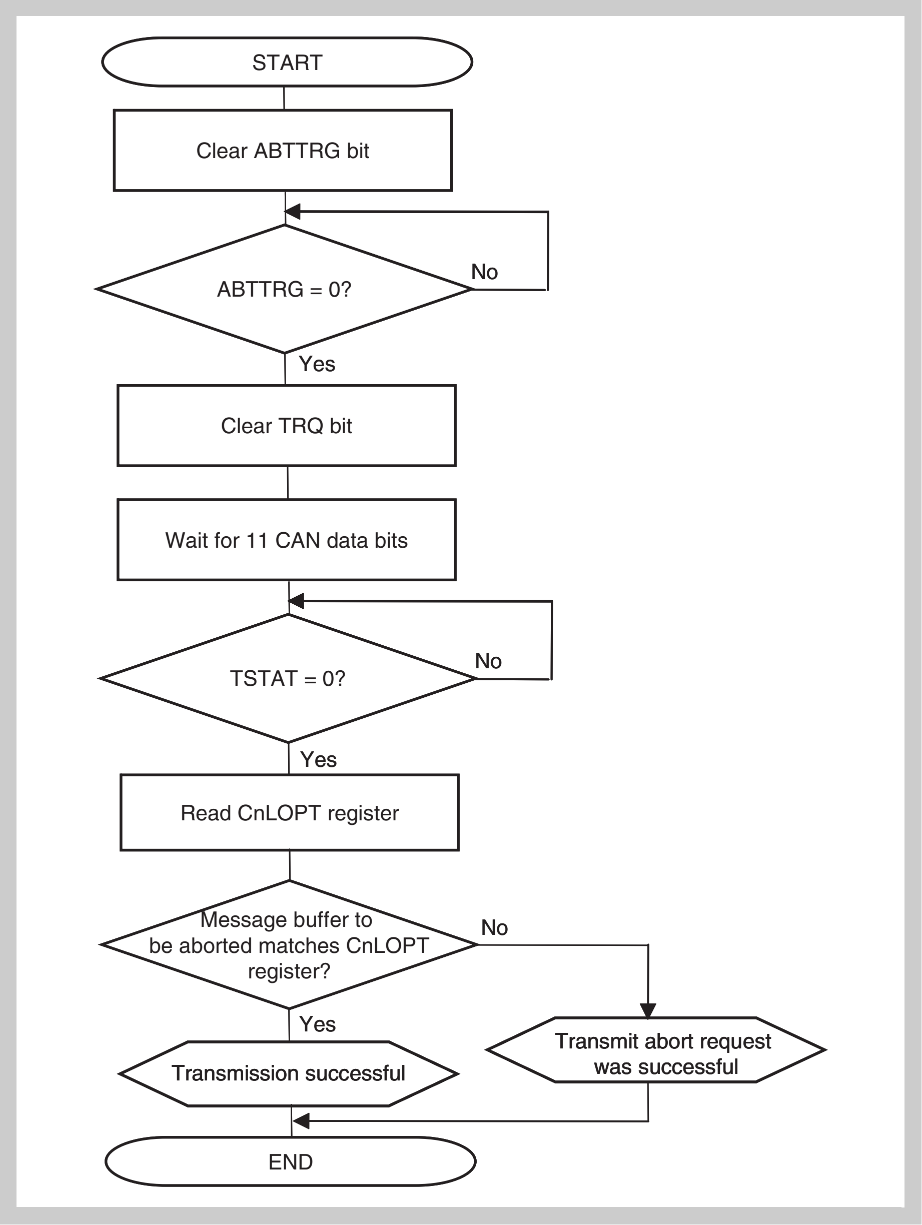


Figure 20-46 Transmission abort processing except for ABT transmission (normal operation mode with ABT)

Caution

1. Clear the TRQ bit for aborting transmission request, not the RDY bit.
2. Before making a sleep mode transition request, confirm that there is no transmission request left using this processing.
3. The TSTAT bit can be periodically checked by a user application or can be checked after the transmit completion interrupt.
4. Do not execute any new transmission request including in the other message buffers while transmission abort processing is in progress.

Figure 20-47 shows the processing to skip resumption of transmitting a message that was stopped when transmission of an ABT message buffer was aborted.

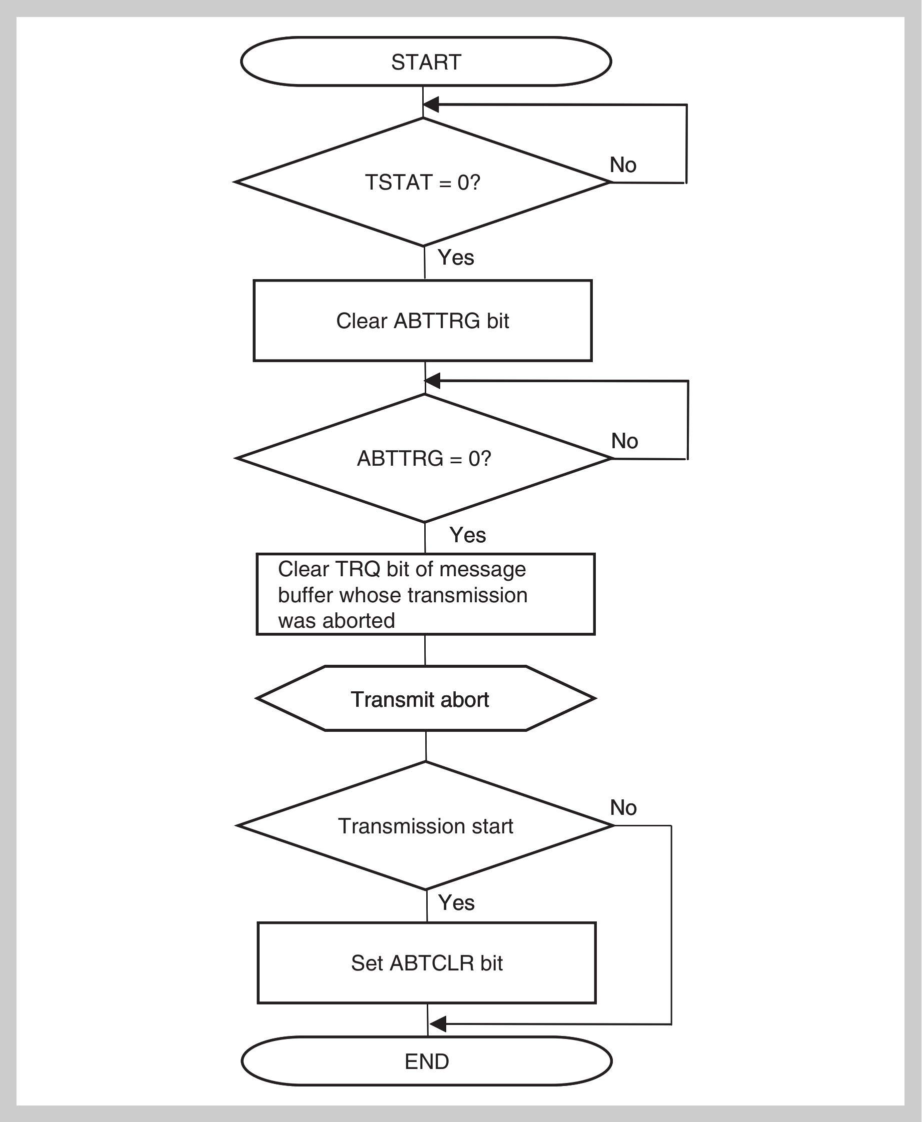


Figure 20-47 Transmission abort processing (normal operation mode with ABT)

Caution

1. Do not set any transmission requests while ABT transmission abort processing is in progress.
2. Make a CAN sleep mode/CAN stop mode transition request after the ABTTRG bit is cleared (after ABT mode is aborted) following the procedure shown in *Figure 20-47* or *Figure 20-48*. When clearing a transmission request in an area other than the ABT area, follow the procedure shown in *Figure 20-45 on page 801*.

Figure 20-48 shows the processing to not skip resumption of transmitting a message that was stopped when transmission of an ABT message buffer was aborted.

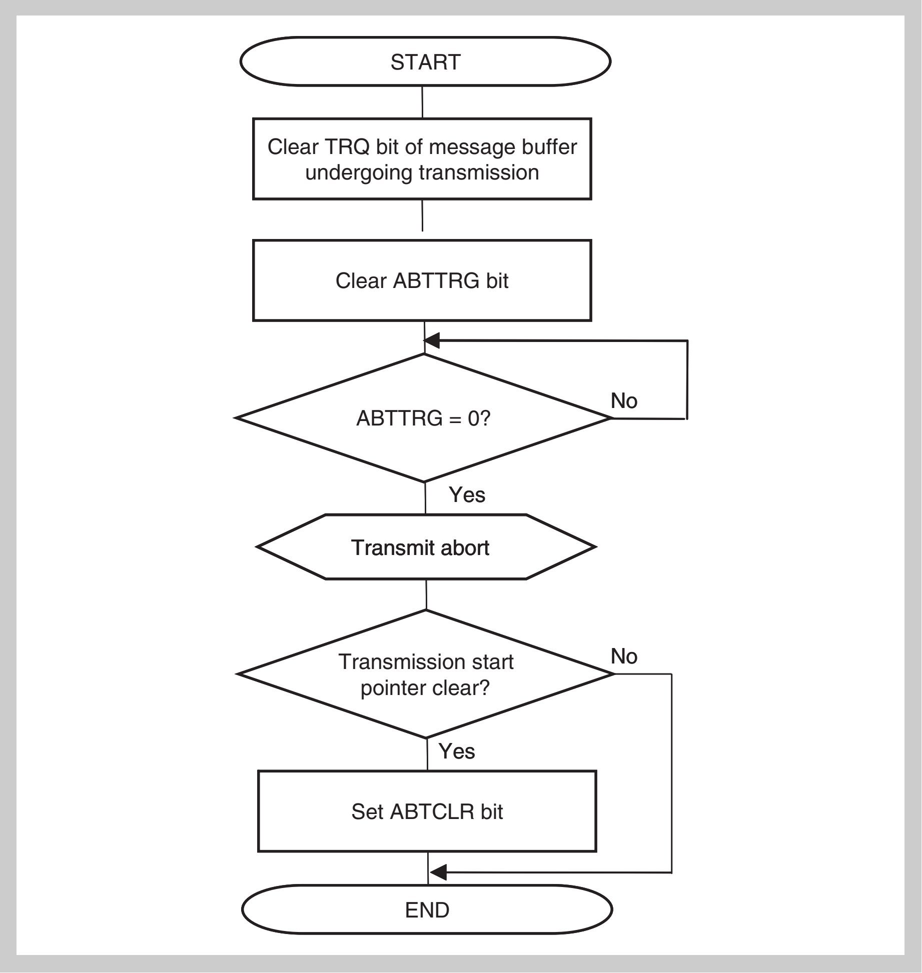
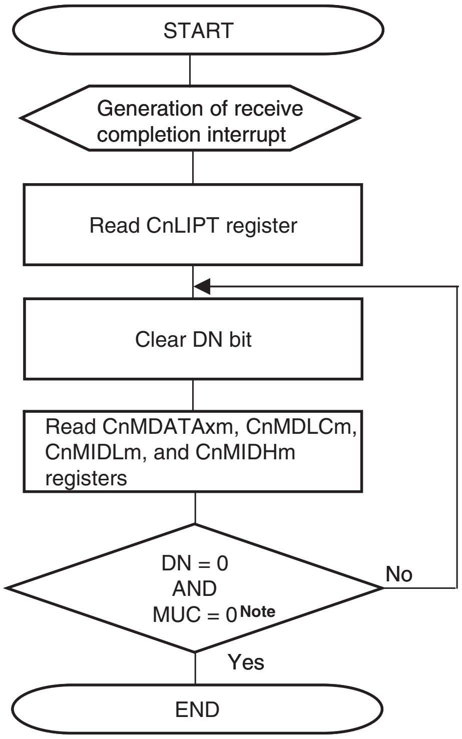

START
Clear TRQ bit of message buffer undergoing transmission
Clear ABTTRG bit
ABTTRG = 0?
No
Yes
Transmit abort
Transmission start pointer clear?
No
Yes
Set ABTCLR bit
END

Figure 20-48 ABT transmission request abort processing (normal operation mode with ABT)

Caution

1. Do not set any transmission requests while ABT transmission abort processing is in progress.
2. Make a CAN sleep mode/CAN stop mode request after the ABTTRG bit is cleared (after ABT mode is stopped) following the procedure shown in *Figure 20-47* or *Figure 20-48*. When clearing a transmission request in an area other than the ABT area, follow the procedure shown in *Figure 20-45 on page 801*.


START
Generation of receive completion interrupt
Read CnLIPT register
Clear DN bit
Read CnMDATAxm, CnMDLCm, CnMIDLm, and CnMIDHm registers
DN = 0 AND MUC = 0 Note
No
Yes
END

Note Check the MUC and DN bits using one read access.

Figure 20-49 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.

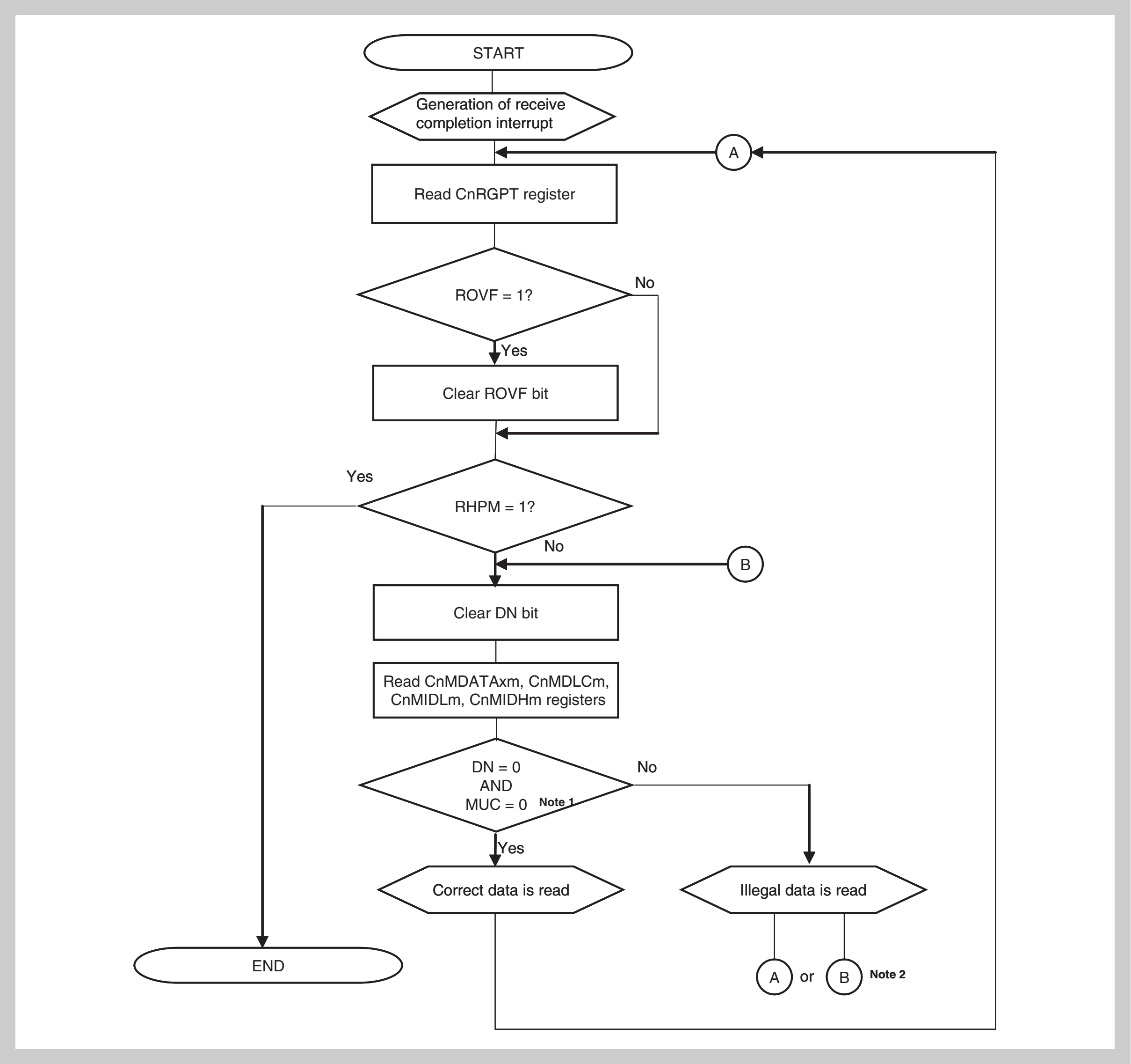


Figure 20-50 Reception via interrupt (using CnRGPT register)

Note
1. Check the MUC and DN bits using one read access.
2. Depending of the processing target of the application, two ways are possible:
   - Way A: The message is not processed within this pass, but with the next pass, depending on the timing this can happen latest with the next Receive Interrupt. Other messages will be processed earlier.
   - Way B: The message is processed within this pass, the loop waits on this message. Other messages will be processed later.
3. Also check the MBON flag at the beginning and at the end of the interrupt routine, in order to check the access to the message buffers as well as reception history list registers, in case a pending sleep mode had been executed. If MBON is detected to be cleared at any check, the actions and results of the processing have to be discarded and processed again, after MBON is set again.
   It is recommended to cancel any sleep mode requests, before processing RX interrupts.
4. If ROVF was set once, the receive history list is inconsistent. Consider to scan all configured receive buffers for receptions.

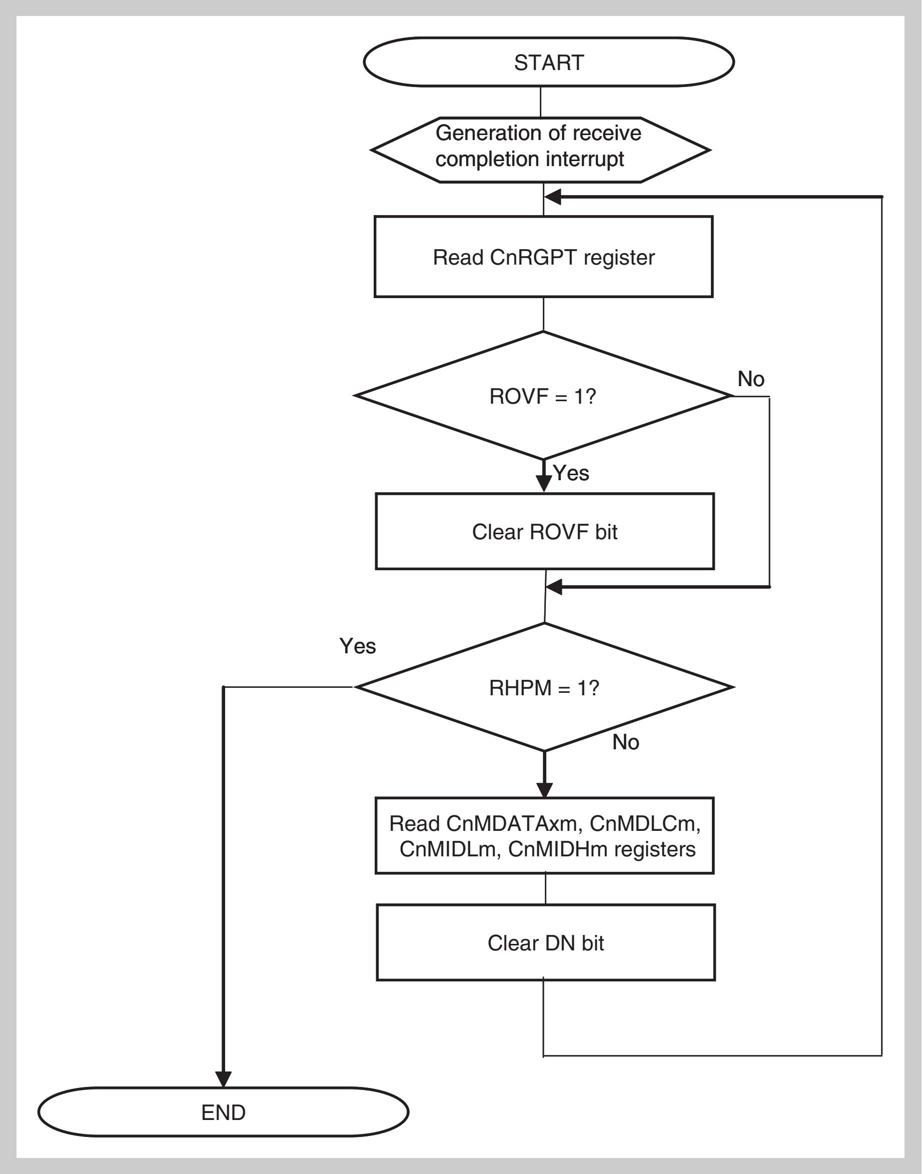

START
Generation of receive completion interrupt
Read CnRGPT register
ROVF = 1?
No
Yes
Clear ROVF bit
Yes
RHPM = 1?
No
Read CnMDATAxm, CnMDLCm, CnMIDLm, CnMIDHm registers
Clear DN bit
END

Figure 20-51 Reception via interrupt (using CnRGPT register), alternative way

Note

1. Also check the MBON flag at the beginning and at the end of the interrupt routine, in order to check the access to the message buffers as well as reception history list registers, in case a pending sleep mode had been executed. If MBON is detected to be cleared at any check, the actions and results of the processing have to be discarded and processed again, after MBON is set again.
   It is recommended to cancel any sleep mode requests, before processing RX interrupts.
2. If ROVF was set once, the receive history list is inconsistent. Consider to scan all configured receive buffers for receptions.
3. This flow will not provide most recently received data for the application. However, due to less effort on processing, it reduces interrupt load.
4. The overwrite function (CnMCONFm.OWS=1) must not be used with this flow - data inconsistency could occur.
5. It can be used alternatively to *Figure 20-50 on page 806*.

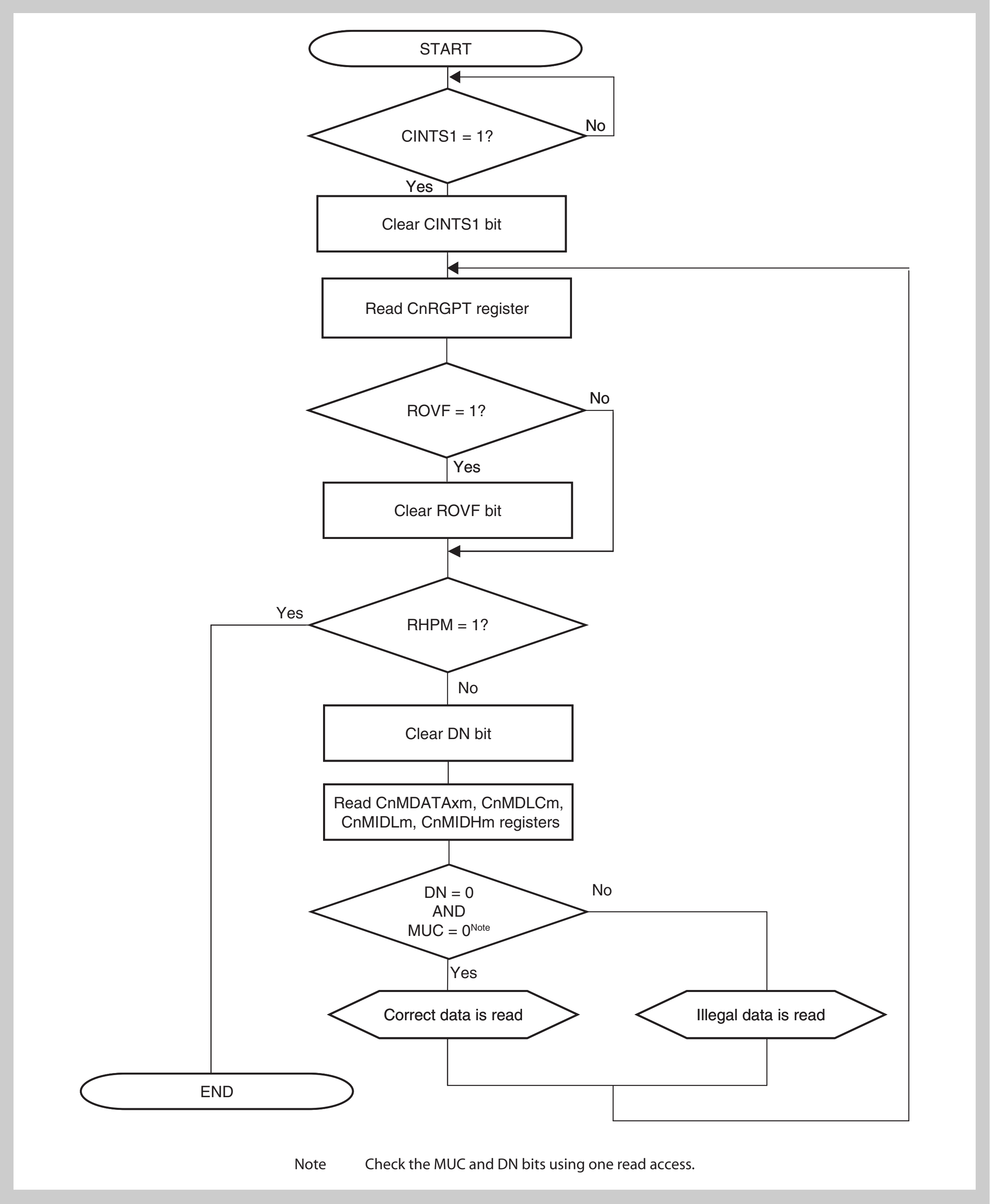


Figure 20-52 Reception via software polling

Note

1. Also check the MBON flag at the beginning and at the end of the polling routine, in order to check the access to the message buffers as well as reception history list registers, in case a pending sleep mode had been executed. If MBON is detected to be cleared at any check, the actions and results of the processing have to be discarded and processed again, after MBON is set again.
2. If ROVF was set once, the receive history list is inconsistent. Consider to scan all configured receive buffers for receptions.

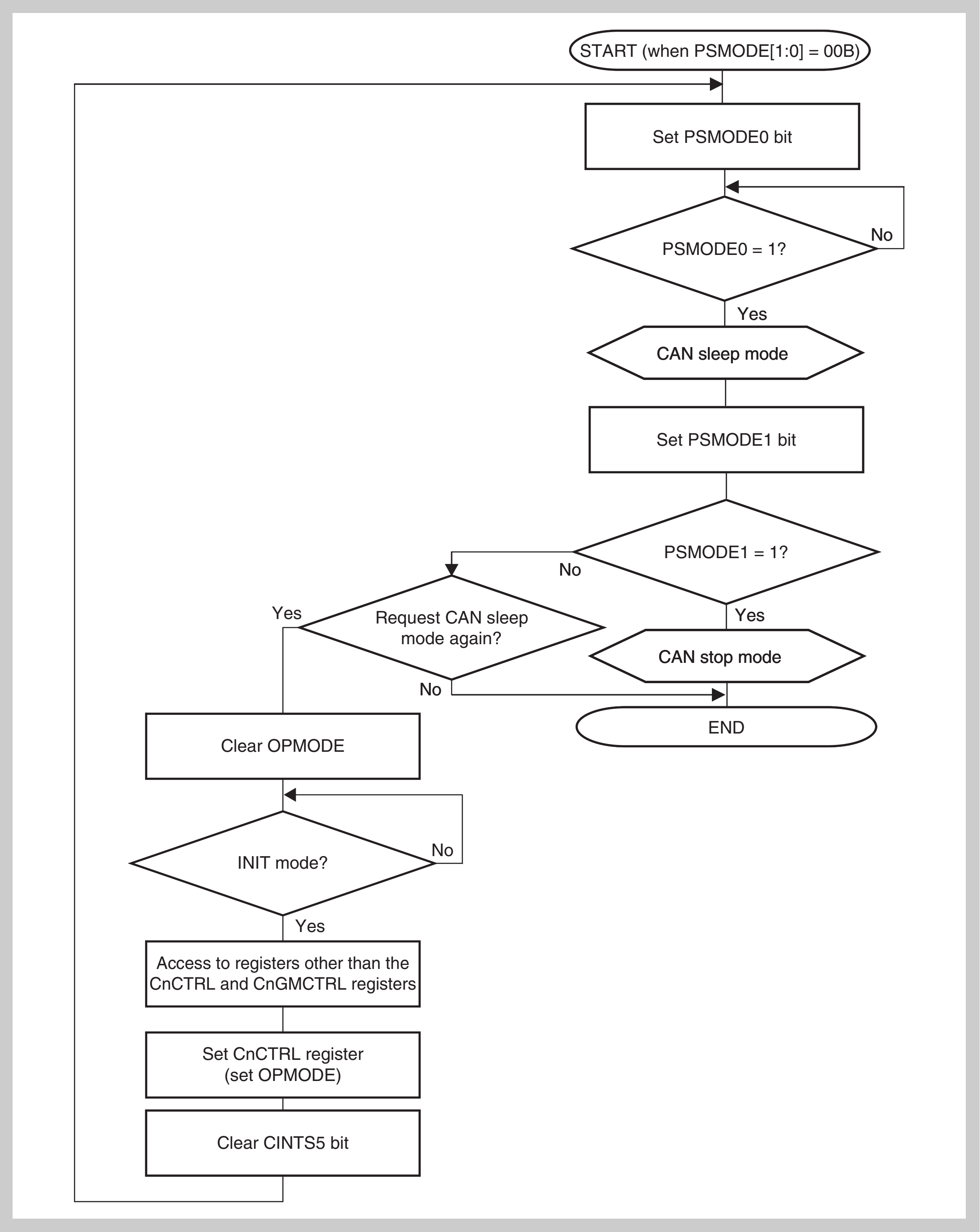

START (when PSMODE[1:0] = 00B)
Set PSMODE0 bit
PSMODE0 = 1?
No
Yes
CAN sleep mode
Set PSMODE1 bit
PSMODE1 = 1?
No
Yes
CAN stop mode
END
Request CAN sleep mode again?
Yes
No
Clear OPMODE
INIT mode?
No
Yes
Access to registers other than the CnCTRL and CnGMCTRL registers
Set CnCTRL register (set OPMODE)
Clear CINTS5 bit

Figure 20-53 Setting CAN sleep mode/stop mode

Caution To abort transmission before making a request for the CAN sleep mode, perform processing according to *Figure 20-45 on page 801* and *Figure 20-47 on page 803*.

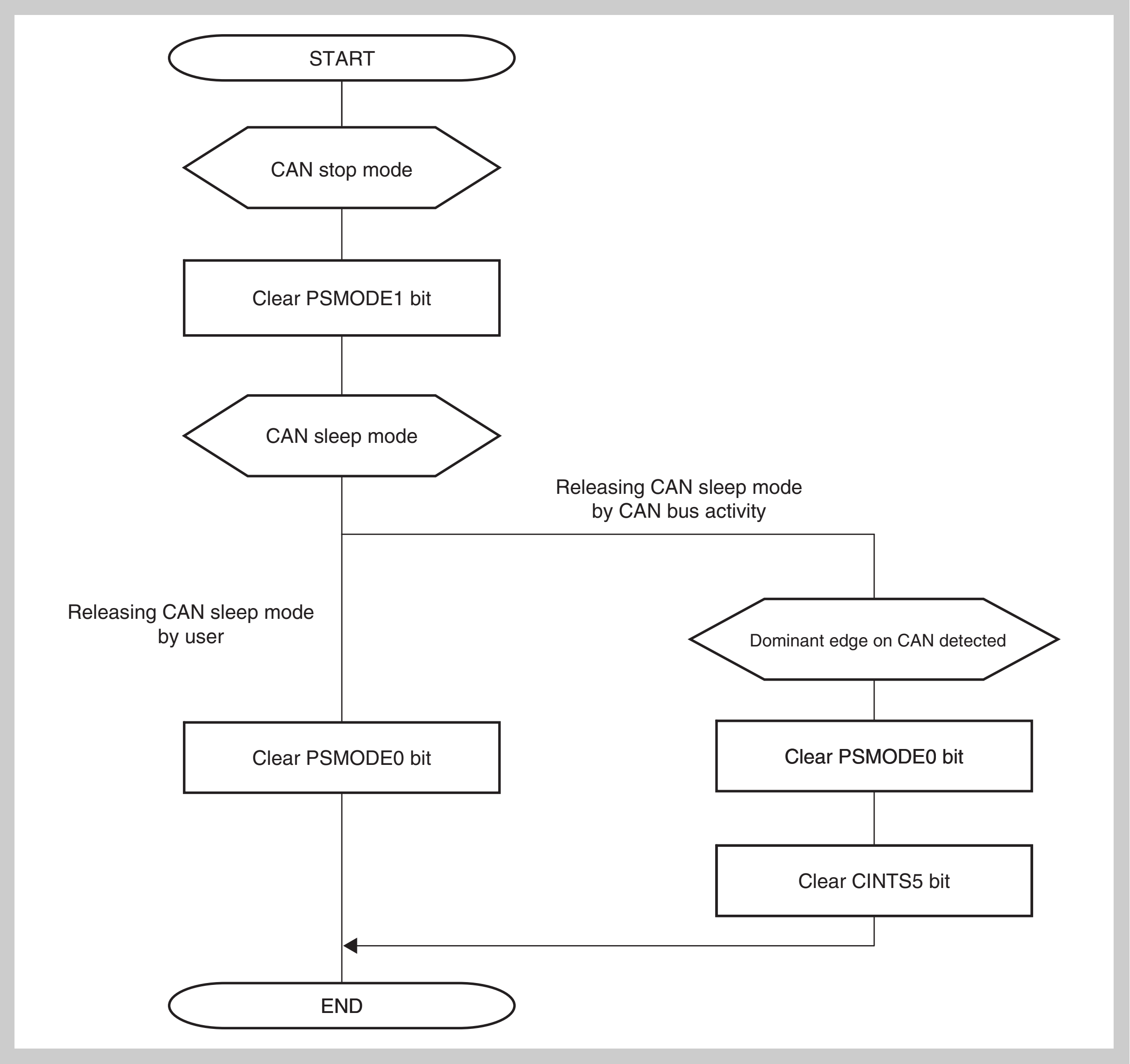
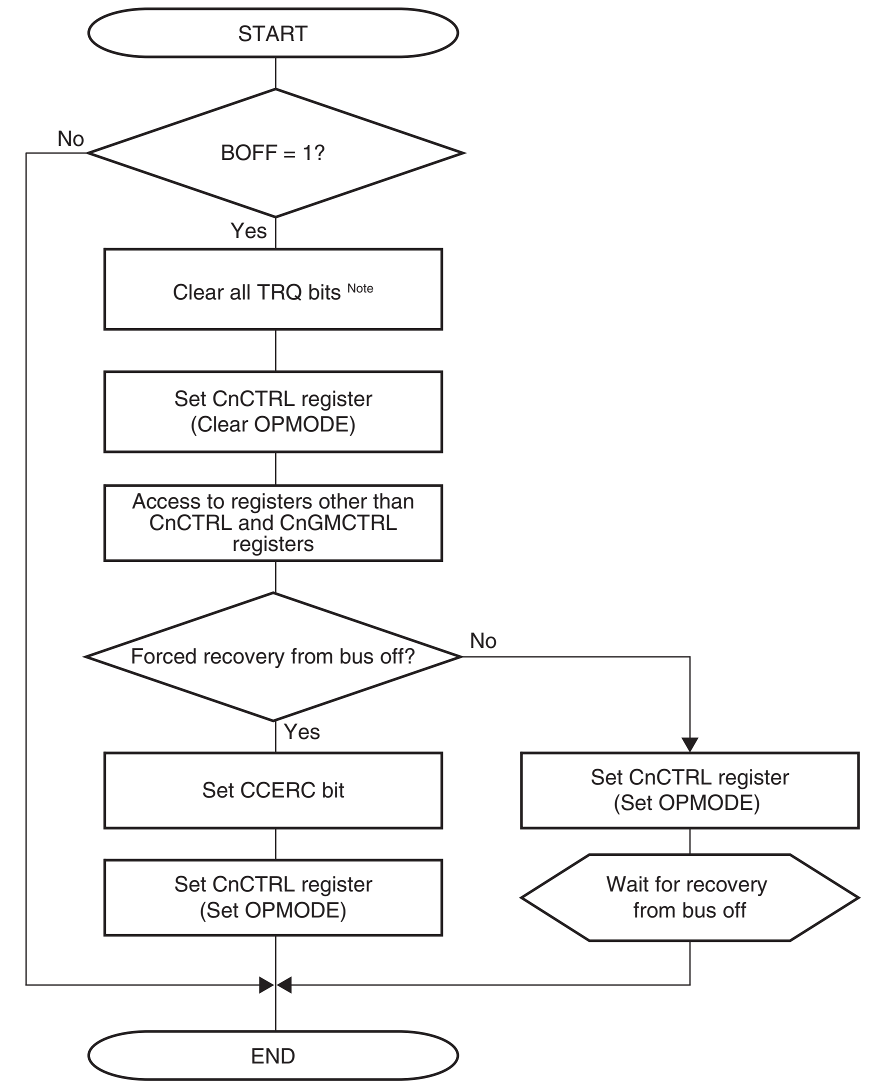


Figure 20-54 Clear CAN sleep/stop mode

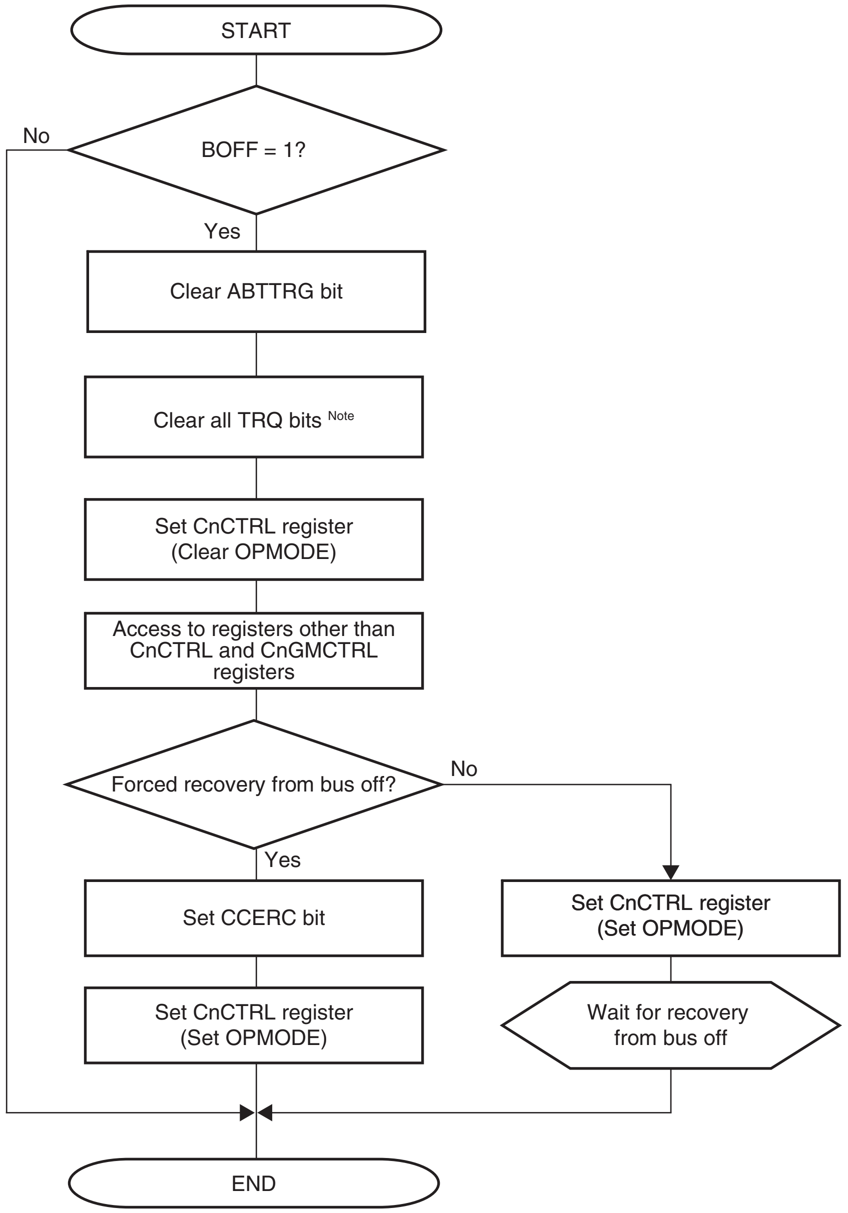


Note: Clear all TRQ bits when re-initialization of message buffer is executed by clearing RDY bit before bus-off recovery sequence is started.

OPMODE: Normal operation mode, normal operation mode with ABT, receive-only mode, single-shot mode, self-test mode

Figure 20-55 Bus-off recovery (except normal operation mode with ABT)

Caution When the transmission from the initialization mode to any operation modes is requested to execute bus-off recovery sequence again in the bus-off recovery sequence, reception error counter is cleared.
Therefore it is necessary to detect 11 consecutive recessive-level bits 128 times on the bus again.



Note: Clear all TRQ bits when re-initialization of message buffer is executed by clearing RDY bit before bus-off recovery sequence is started.

OPMODE: Normal operation mode, normal operation mode with ABT, receive-only mode, single-shot mode, self-test mode

Figure 20-56 Bus-off recovery (Normal Operation Mode with ABT)

Caution When the transmission from the initialization mode to any operation modes is requested to execute bus-off recovery sequence again in the bus-off recovery sequence, reception error counter is cleared.
Therefore it is necessary to detect 11 consecutive recessive-level bits 128 times on the bus again.

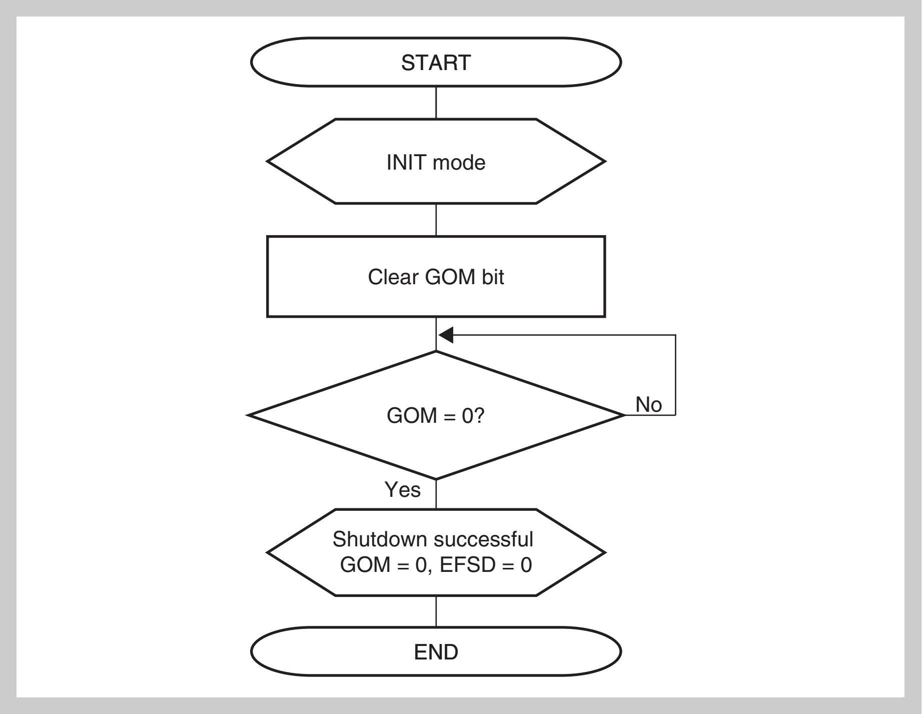


Figure 20-57 Normal shutdown process

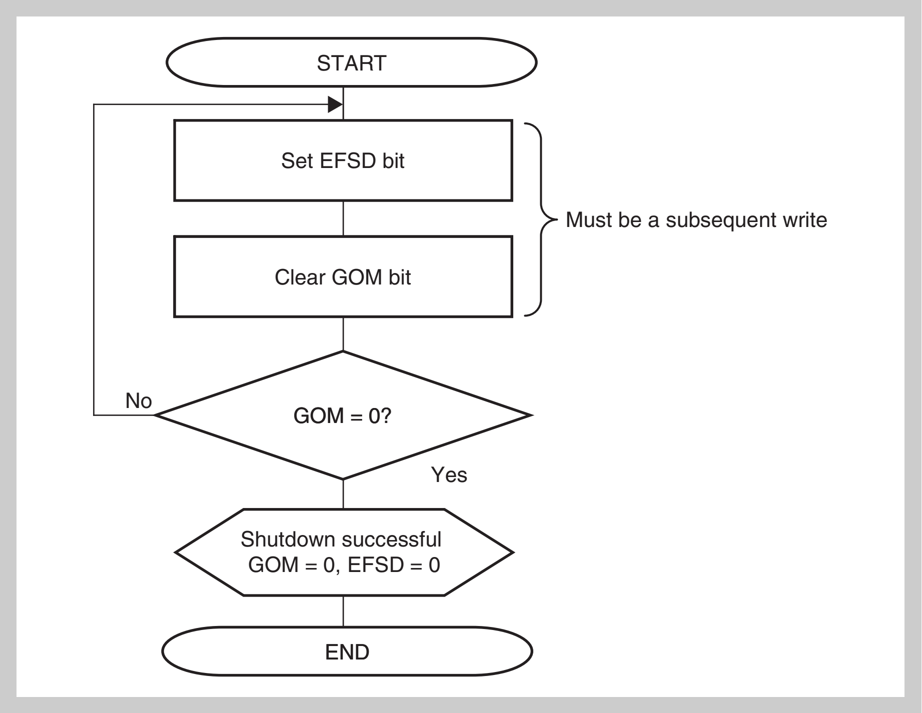


Figure 20-58 Forced shutdown process

Caution Do not read- or write-access any registers by software between setting the EFSD bit and clearing the GOM bit.

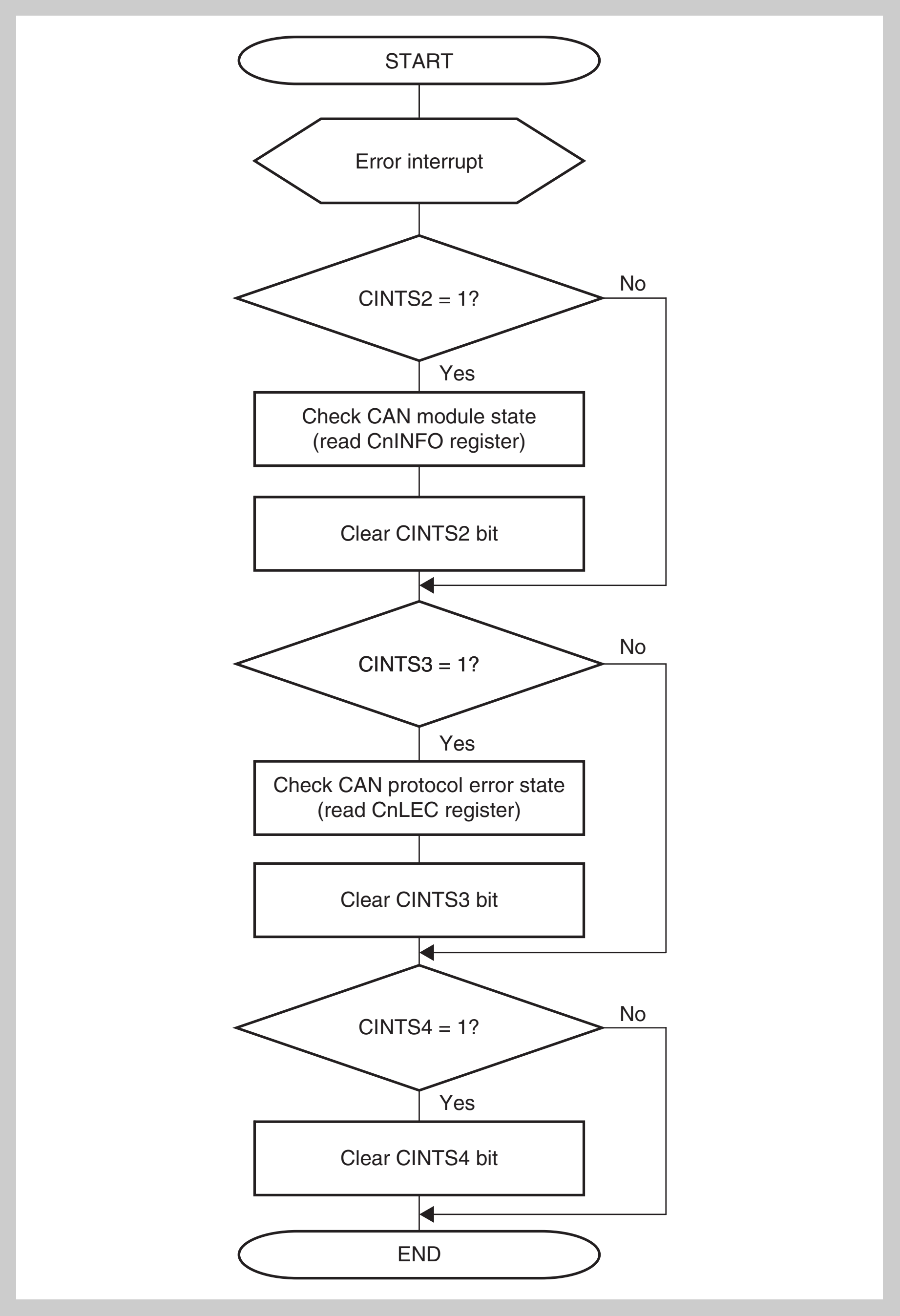


Figure 20-59 Error handling

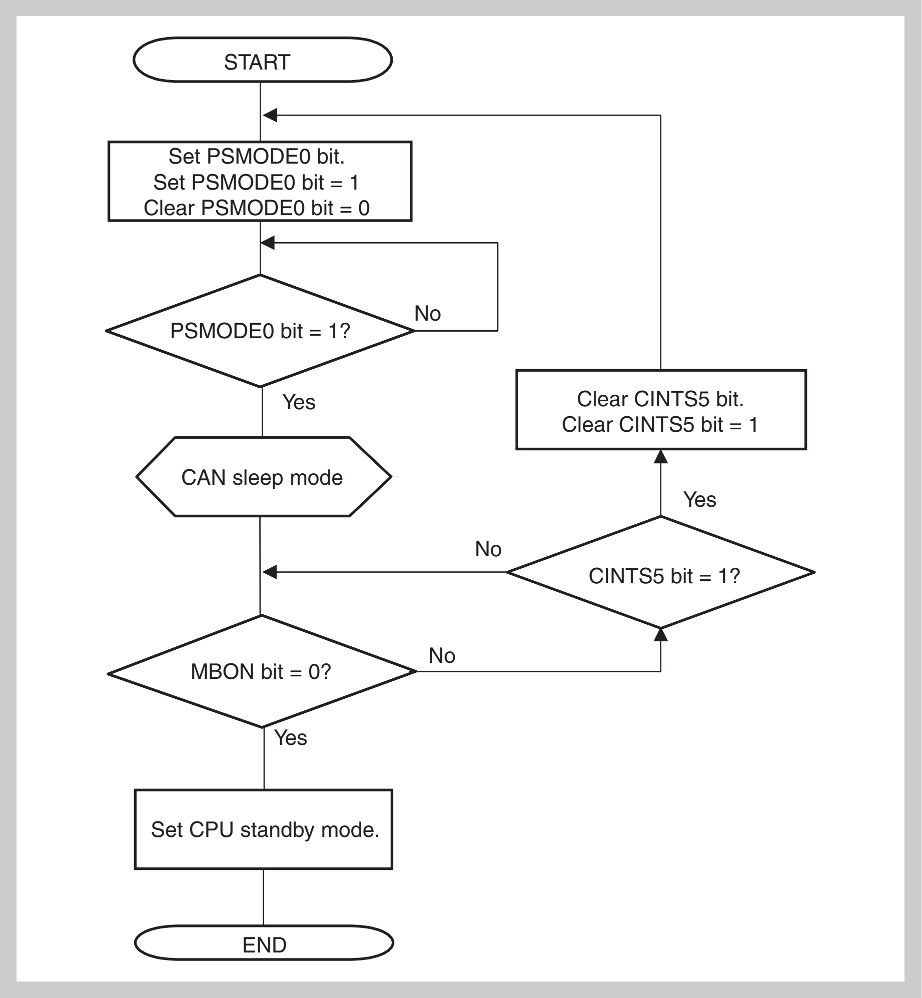


Figure 20-60 Setting CPU stand-by (from CAN sleep mode)

Caution Before the CPU is set in the CPU standby mode, please check if the CAN sleep mode has been reached.
However, after check of the CAN sleep mode, until the CPU is set in the CPU standby mode, the CAN sleep mode may be cancelled by wakeup from CAN bus.

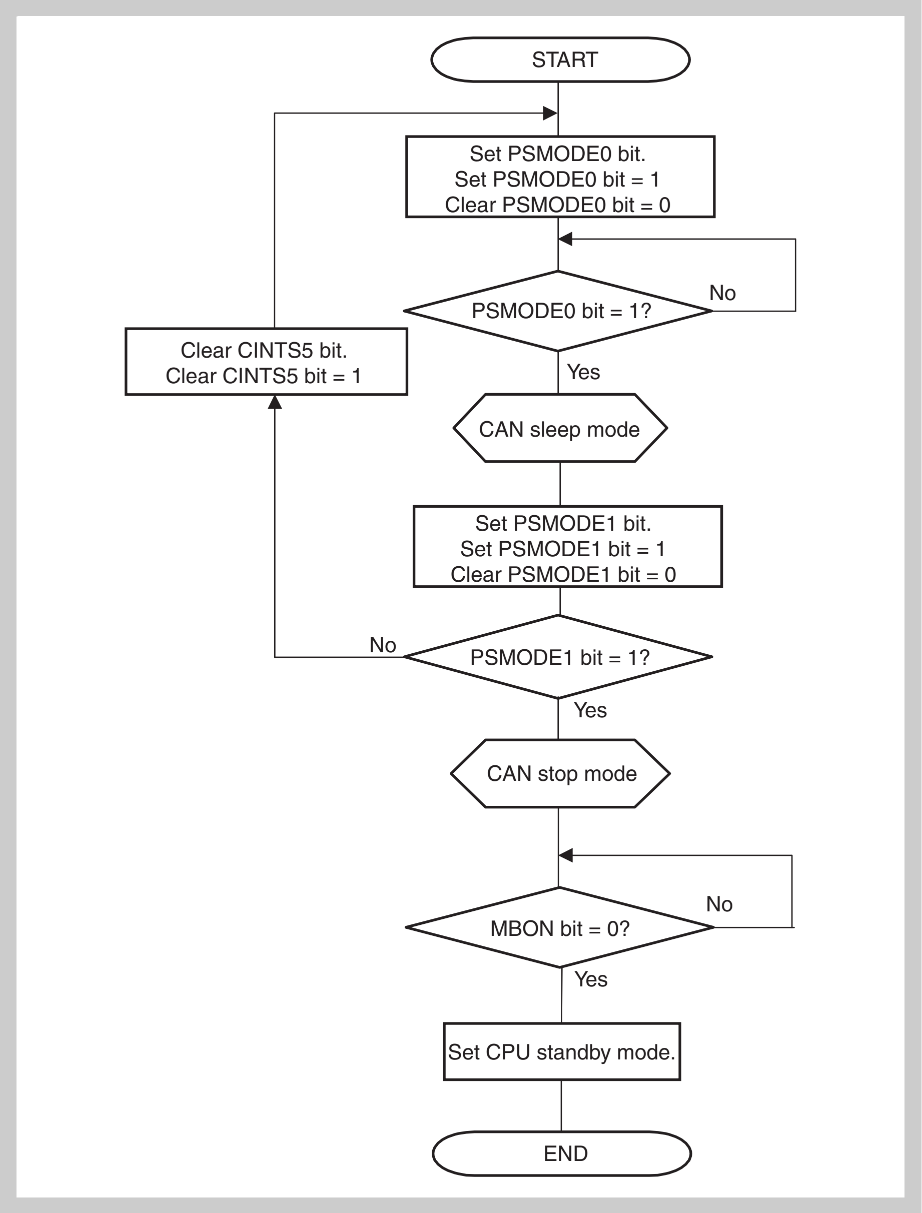


Figure 20-61 **Setting CPU stand-by (from CAN stop mode)**

Caution The CAN stop mode can only be released by writing 01_B to the PSMODE[1:0] bit of the CnCTRL register and not by a change in the CAN bus state.

Chapter 21 A/D Converter (ADC)

The V850ES/Fx3 microcontrollers have following instances of the A/D Converter ADC:

ADC	V850ES/FE3	V850ES/FF3	V850ES/FG3	V850ES/FJ3	V850ES/FK3
Instances	1	1	1	1	2
Names	ADA0	ADA0	ADA0	ADA0	ADA0 ADA1
Channels	10	12	16	24	ADA0: 24 ADA1: 16

Throughout this chapter, the individual instances of ADC are identified by “n”, for example, ADAnM0 for the ADAn mode register 0.

Throughout this chapter, the individual channels of each ADC instance are identified by “m”, for example, ADAnCRm for the conversion result register m of ADAn.

21.1 Functions

The A/D Converter converts analog input signals into digital values.

The A/D Converter has the following features.

- 10-bit resolution
- Successive approximation method
- The following functions are provided as operation modes.
  - Continuous select mode
  - Continuous scan mode
  - One-shot select mode
  - One-shot scan mode
- The following functions are provided as trigger modes.
  - Software trigger mode
  - Timer trigger mode
  - Hardware trigger mode
  - External trigger mode
- Power-fail monitor function (conversion result compare function)
- Self diagnostic function
- Discharge function

The block diagram of the A/D Converter is shown below.

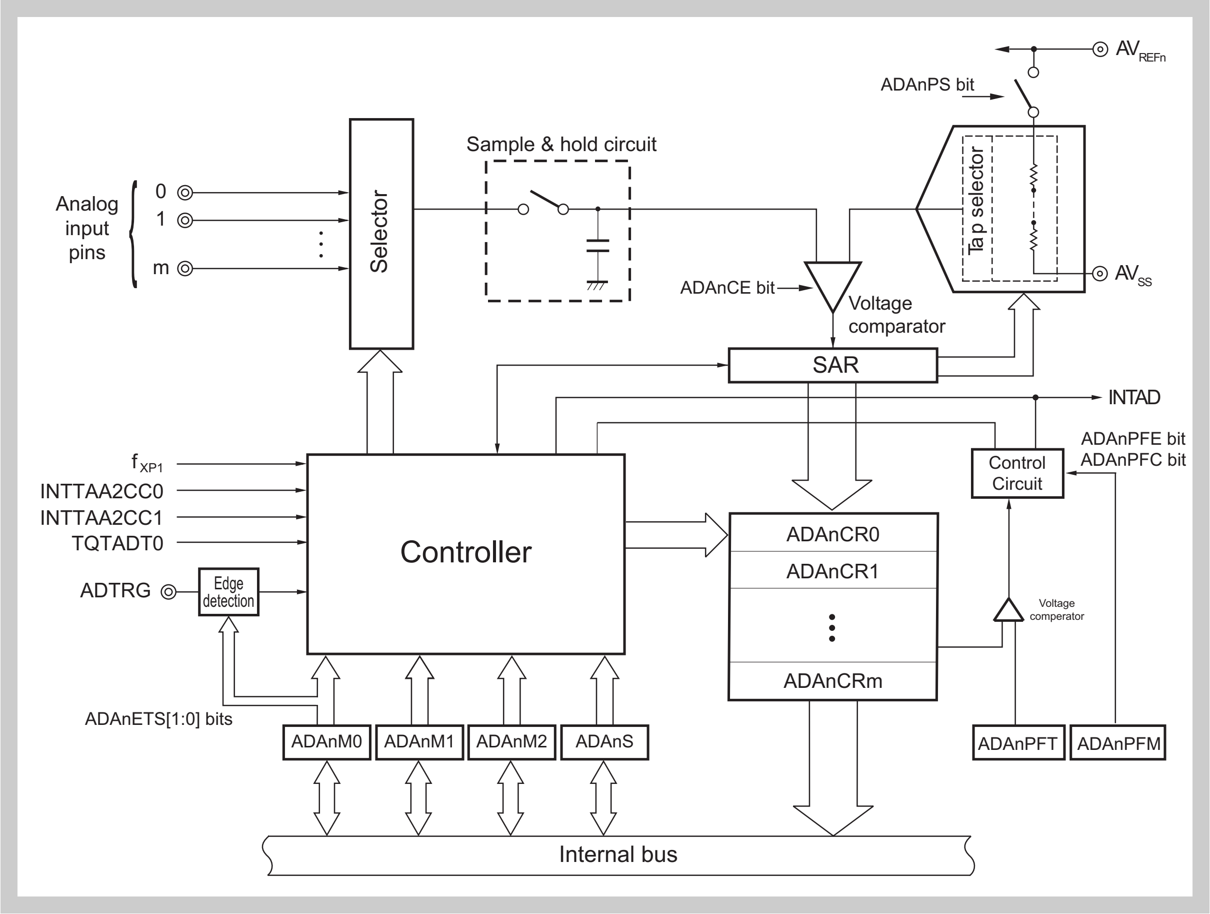


Figure 21-1 Block diagram of A/D Converter

21.2 Configuration

The A/D Converter includes the following hardware.

Table 21-1 Configuration of A/D Converter

Item	Configuration
Analog inputs	ANI0 to ANIm / ANI100 to ANI1m pins
Registers	Successive approximation register (SAR) A/D conversion result registers ADAnCRm, ADAnCRmH AVREF A/D conversion diagnostic registers ADAnCRDD, ADAnCRDDH AVSS A/D conversion diagnostic registers ADAnCRSS, ADAnCRSSH ADC power-fail compare mode register ADAnPFM ADC power-fail compare threshold value register ADAnPFT
Control registers	A/D Converter mode registers 0 to 2 (ADAnM0 to ADAnM2) A/D Converter channel specification register 0 (ADAnS)

(1) SAR - Successive approximation register

The SAR register compares the voltage value of the analog input signal with the voltage tap (compare voltage) value from the series resistor string, and holds the comparison result starting from the most significant bit (MSB).

When the comparison result has been held down to the least significant bit (LSB) (i.e., when A/D conversion is complete), the contents of the SAR register are transferred to the ADAnCRm register.

(2) A/D conversion result register n (ADAnCRm), A/D conversion result register nH (ADAnCRmH)

The ADAnCRm register is a 16-bit register that stores the A/D conversion result. ADAnCRm consist of m registers and the A/D conversion result is stored in the 10 higher bits of the ADAnCRm register corresponding to analog input. (The lower 6 bits are fixed to 0.)

The ADAnCRm register is read-only, in 16-bit units.

When using only the higher 8 bits of the A/D conversion result, the ADAnCRmH register is read-only, in 8-bit units.

Caution A write operation to the ADAnM0 and ADAnS registers may cause the contents of the ADAnCRm register to become undefined. After the conversion, read the conversion result before writing to the ADAnM0 and ADAnS registers. Correct conversion results may not be read if a sequence other than the above is used.

(3) Power-fail compare threshold value register (ADAnPFT)

The ADAnPFT register sets a threshold value that is compared with the value of A/D conversion result register nH (ADAnCRmH). The 8-bit data set to the ADAnPFT register is compared with the higher 8 bits of the A/D conversion result register (ADAnCRmH).

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

Reset input clears this register to 00_H.

(4) Sample & hold circuit

The sample & hold circuit samples each of the analog input signals selected by the input circuit and sends the sampled data to the Voltage Comparator. This circuit also holds the sampled analog input signal voltage during A/D conversion.

(5) Voltage comparator

The Voltage comparator compares a voltage value that has been sampled and held with the voltage value of the series resistor string.

(6) Series resistor string

This series resistor string is connected between AV_{REFn} and AV_{SS} and generates a voltage for comparison with the analog input signal.

(7) ANInm pins

These are analog input pins for the m A/D Converter channels and are used to input analog signals to be converted into digital signals. Pins other than the one selected as the analog input by the ADAnS register can be used as input port pins.

Caution

1. Make sure that the voltages input to the ANInm pins do not exceed the rated values. In particular if a voltage of AV_{REFn} or higher is input to a channel, the conversion value of that channel becomes undefined, and the conversion values of the other channels may also be affected.
2. See chapter *21.5 on page 849* “(4) alternate I/O“ for details on analog input pin usage.

(8) AV_{REFn} pin

This is the pin used to input the reference voltage of the A/D Converter. AV_{REFn} also delivers the A/D Converter's analog supply voltage AV_{DD}.

The signals input to the ANInmm pins are converted to digital signals based on the voltage applied between the AVREFn and AV_{SS} pins.

(9) AV_{SS} pin

This is the ground pin of the A/D Converter. Always make the potential at this pin the same as that at the V_{SS} pin even when the A/D Converter is not used.

21.3 ADC Registers

The A/D Converter is controlled by the following registers:

- A/D Converter mode registers 0, 1, 2 (ADAnM0, ADAnM1, ADAnM2)
- A/D Converter channel specification register 0 (ADAnS)
- Power-fail compare mode register (ADAnPFM)

The following registers are also used:

- A/D conversion result register n (ADAnCRm)
- A/D conversion result register nH (ADAnCRmH)
- Power-fail compare threshold value register (ADAnPFT)

(1) ADAnM0 - ADC mode register 0

The ADAnM0 register is an 8-bit register that specifies the operation mode and controls conversion operations.

Access This register can be read/written in 8-bit or 1-bit units. However, bit 0 is read-only.

Address ADA0M0: FFFFF200$_H$
ADA1M0: FFFFF240$_H$

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

	7	6	5	4	3	2	1	0
ADAnM0	ADAnCE	ADAnPS	ADAnMD1	ADAnMD0	ADAnETS1	ADAnETS0	ADAnTMD	ADAnEF
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R

Caution

1. .Writing to ADAnEF bit is ignored.
2. When not using the A/D Converter, stop the operation by setting the ADAnPS bit to 0 to reduce the current consumption.
3. Access to the ADAnM0 register during sub-clock operation and when the main clock is stopped is prohibited.

Table 21-2 ADAnM0 register contents (1/2)

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>7</td><td>ADAnCE</td><td>A/D conversion control of ADAn
0: Stops conversion
1: Starts conversion</td></tr>
<tr><td>6</td><td>ADAnPS</td><td>Power control of ADAn
0: A/D conversion of ADCn is stopped
1: A/D conversion of ADCn is operating</td></tr>
<tr><td>5, 4</td><td>ADAnMD1,
ADAnMD0</td><td>Specifiec the opration mode of ADAn.
<table>
<tr><th>ADAnMD1</th><th>ADAnMD0</th><th>A/D conversion operation mode</th></tr>
<tr><td>0</td><td>0</td><td>Continuous select mode</td></tr>
<tr><td>0</td><td>1</td><td>Continuous scan mode</td></tr>
<tr><td>1</td><td>0</td><td>One-shot select mode</td></tr>
<tr><td>1</td><td>1</td><td>One-shot scan mode</td></tr>
</table></td></tr>
</table>

Table 21-2 ADAnM0 register contents (2/2)

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>3, 2</td><td>ADAnETS1,
ADAnETS0</td><td>Specifies the valid edege of external trigger input (ADTRG pin).
<table>
<tr><th>ADAnETS1</th><th>ADAnETS0</th><th>External trigger nput (ADTRG pin) ivalid edge</th></tr>
<tr><td>0</td><td>0</td><td>Continuous select mode</td></tr>
<tr><td>0</td><td>1</td><td>Continuous scan mode</td></tr>
<tr><td>1</td><td>0</td><td>One-shot select mode</td></tr>
<tr><td>1</td><td>1</td><td>One-shot scan mode</td></tr>
</table></td></tr>
<tr><td>1</td><td>ADAnTMD</td><td>Trigger mode specification of ADAn
0: Software trigger mode
1: External trigger mode/ timer trigger mode</td></tr>
<tr><td>0</td><td>ADAnEF</td><td>A/D converter status display
0: A/D conversion stopped
1: A/D conversion in progress</td></tr>
</table>

(2) ADAnM1 - ADC mode register 1

The ADAnM1 register is an 8-bit register that controls the conversion time specification.

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

Address ADA0M1: FFFFF201$_H$
ADA1M1: FFFFF241$_H$

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

	7	6	5	4	3	2	1	0
ADAnM1	0	0	0	0	ADAnFR3	ADAnFR2	ADAnFR1	ADAnFR0
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Caution 1. Be sure to clear bits 4-7 to 0.

2. Changing the ADAnFR[3:0] bits during conversion (ADAnM0.ADAnCE0 = 1) is prohibited.

Table 21-3 ADAnM1register contents

Bit position	Bit name	Function
3 to 0	ADAnFR[3:0]	Specifies the number of conversion, sampling and discharging time settings.

ADAn FR3	ADAn FR2	ADAn FR1	ADAn FR0	A/D conversion time	A/D sampling time	Discharging time (ADAnM2. ADAnDISC = 1)
0	0	0	0	$32/f_{XP1}$	$17/f_{XP1}$	$4/f_{XP1}$
0	0	0	1	$64/f_{XP1}$	$34/f_{XP1}$	$8/f_{XP1}$
0	0	1	0	$96/f_{XP1}$	$51/f_{XP1}$	$12/f_{XP1}$
0	0	1	1	$128/f_{XP1}$	$68/f_{XP1}$	$16/f_{XP1}$
0	1	0	0	$160/f_{XP1}$	$85/f_{XP1}$	$20/f_{XP1}$
0	1	0	1	$192/f_{XP1}$	$102/f_{XP1}$	$24/f_{XP1}$
0	1	1	0	$224/f_{XP1}$	$119/f_{XP1}$	$28/f_{XP1}$
0	1	1	1	$256/f_{XP1}$	$136/f_{XP1}$	$32/f_{XP1}$
1	0	0	0	$288/f_{XP1}$	$153/f_{XP1}$	$36/f_{XP1}$
1	0	0	1	$320/f_{XP1}$	$170/f_{XP1}$	$40/f_{XP1}$
1	0	1	0	$352/f_{XP1}$	$187/f_{XP1}$	$44/f_{XP1}$
1	0	1	1	$384/f_{XP1}$	$204/f_{XP1}$	$48/f_{XP1}$
1	1	0	0	$416/f_{XP1}$	$221/f_{XP1}$	$52/f_{XP1}$
1	1	0	1	$448/f_{XP1}$	$238/f_{XP1}$	$56/f_{XP1}$
1	1	1	0	$480/f_{XP1}$	$255/f_{XP1}$	$60/f_{XP1}$
1	1	1	1	$512/f_{XP1}$	$272/f_{XP1}$	$64/f_{XP1}$

Note: For valid settings of ADAnFR[3:0] bits refer to *Table 21-4*.

Table 21-4 Conversion time settings

ADAnFR3	ADAnFR2	ADAnFR1	ADAnFR0	A/D conversion time	f_{XP1} = 32 MHz	f_{XP1} = 24 MHz	f_{XP1} = 20 MHz	f_{XP1} = 16 MHz	f_{XP1} = 10 MHz	f_{XP1} = 4 MHz
0	0	0	0	$32/f_{XP1}$	prohibited	prohibited	prohibited	prohibited	3.20 µs	8.00 µs
0	0	0	1	$64/f_{XP1}$	prohibited	prohibited	3.20 µs	4.00 µs	6.40 µs	16.00 µs
0	0	1	0	$96/f_{XP1}$	prohibited	4.00 µs	4.80 µs	6.00 µs	9.60 µs	prohibited
0	0	1	1	$128/f_{XP1}$	4.00 µs	5.34 µs	6.40 µs	8.00 µs	12.80 µs	prohibited
0	1	0	0	$160/f_{XP1}$	5.00 µs	6.67 µs	8.00 µs	10.00 µs	16.00 µs	prohibited
0	1	0	1	$192/f_{XP1}$	6.00 µs	8.00 µs	9.60 µs	12.00 µs	prohibited	prohibited
0	1	1	0	$224/f_{XP1}$	7.00 µs	9.34 µs	11.20 µs	14.00 µs	prohibited	prohibited
0	1	1	1	$256/f_{XP1}$	8.00 µs	10.67 µs	12.80 µs	16.00 µs	prohibited	prohibited
1	0	0	0	$288/f_{XP1}$	9.00 µs	12.00 µs	14.40 µs	prohibited	prohibited	prohibited
1	0	0	1	$320/f_{XP1}$	10.00 µs	13.34 µs	16.00 µs	prohibited	prohibited	prohibited
1	0	1	0	$352/f_{XP1}$	11.00 µs	14.67 µs	prohibited	prohibited	prohibited	prohibited
1	0	1	1	$384/f_{XP1}$	12.00 µs	16.00 µs	prohibited	prohibited	prohibited	prohibited
1	1	0	0	$416/f_{XP1}$	13.00 µs	prohibited	prohibited	prohibited	prohibited	prohibited
1	1	0	1	$448/f_{XP1}$	14.00 µs	prohibited	prohibited	prohibited	prohibited	prohibited
1	1	1	0	$480/f_{XP1}$	15.00 µs	prohibited	prohibited	prohibited	prohibited	prohibited
1	1	1	1	$512/f_{XP1}$	16.00 µs	prohibited	prohibited	prohibited	prohibited	prohibited

(3) ADAnM2 - ADC mode register 2

The ADAnM2 register specifies the hardware trigger mode.

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

Address ADA0M2: FFFFF203$_H$
ADA1M2: FFFFF243$_H$

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

	7	6	5	4	3	2	1	0
ADAnM2	0	0	ADAnDIAG	ADAnDISC	0	0	ADAnTMD1	ADAnTMD0
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Caution Be sure to clear bits 7, 6, 3, and 2 to 0.

Table 21-5 ADAnM2 register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>7</td><td>ADAnDIAG</td><td>Diagnostic function control
0: Diagnostic function disabled.
1: Diagnostic function enabled.</td></tr>
<tr><td>6</td><td>ADAnDISC</td><td>Discharge function control
0: Discharge function disabled.
1: Discharge function enabled.
<b>Note:</b> When the discharge function is enabled the AV$_{SS}$ is sampled during 4 clocks (f$_{XP1}$) after finishing A/D conversion. Therefore, additional 4 clocks must be added to the A/D conversion time.</td></tr>
<tr><td>5, 4</td><td>ADAnTMD1
ADAnTMD0</td><td>Specifies the trigger mode of ADAn.
<table>
<tr><th rowspan="2">ADAnTMD1</th><th rowspan="2">ADAnTMD0</th><th colspan="2">Trigger mode specification</th></tr>
<tr><th>ADA0 (n = 0)</th><th>ADA1 (n = 1)</th></tr>
<tr><td>0</td><td>0</td><td>ADTRG external trigger mode</td><td>ADTRG1 external trigger mode</td></tr>
<tr><td>0</td><td>1</td><td>INTTAA2CC0 timer trigger mode 0</td><td>INTTAA5CC0 timer trigger mode 0</td></tr>
<tr><td>1</td><td>0</td><td>INTTAA2CC1 timer trigger mode 1</td><td>INTTAA5CC1 timer trigger mode 1</td></tr>
<tr><td>1</td><td>1</td><td colspan="2">TQTADT0[a] timer trigger mode 2</td></tr>
</table>
[a] TQTADT0: Timer trigger from 6 phase PWM output circuit (motor control)</td></tr>
</table>

(4) ADAnS - ADC channel specification register

The ADAnS register specifies the pin that inputs the analog voltage to be converted into a digital signal.

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

Address ADA0S: FFFFF202_H
ADA1S: FFFFF242_H

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

	7	6	5	4	3	2	1	0
ADAnS	0	0	0	ADAnS4	ADAnS3	ADAnS2	ADAnS1	ADAnS0
	R	R	R	R/W	R/W	R/W	R/W	R/W

Table 21-6 ADAnM2 register contents

Bit position	Bit name	Function
4 to 0	ADAnS[4:0]	Selects the analog input(s) to convert. For details refer to *Table 21-7*.

Table 21-7 Analog input selection settings

ADAnS4	ADAnS3	ADAnS2	ADAnS1	ADAnS0	Analog input to convert					
					ADAnDIAG = 0 (without diagnostic function)		ADAnDIAG = 1 (with diagnostic function)			
					Select mode	Scan mode	Select mode	Scan mode		
0	0	0	0	0	ANI0 ANI100	ANI0 ANI100	AV_{REFn}	ANI0 ANI100	AV_{REFn}	AV_{SS}
0	0	0	0	1	ANI1 ANI101	ANI0, ANI1 ANI100, ANI101	AV_{SS}	ANI0, ANI1 ANI101, ANI101	AV_{REFn}	AV_{SS}
0	0	0	1	0	ANI2 ANI102	ANI0 to ANI2 ANI100 to ANI102	prohibited	ANI0 to ANI2 ANI100 to ANI102	AV_{REFn}	AV_{SS}
0	0	0	1	1	ANI3 ANI103	ANI0 to ANI3 ANI100 to ANI103	prohibited	ANI0 to ANI3 ANI100 to ANI103	AV_{REFn}	AV_{SS}
0	0	1	0	0	ANI4 ANI104	ANI0 to ANI4 ANI100 to ANI104	prohibited	ANI0 to ANI4 ANI100 to ANI104	AV_{REFn}	AV_{SS}
0	0	1	0	1	ANI5 ANI105	ANI0 to ANI5 ANI100 to ANI105	prohibited	ANI0 to ANI5 ANI100 to ANI105	AV_{REFn}	AV_{SS}
0	0	1	1	0	ANI6 ANI106	ANI0 to ANI6 ANI100 to ANI106	prohibited	ANI0 to ANI6 ANI100 to ANI106	AV_{REFn}	AV_{SS}
0	0	1	1	1	ANI7 ANI107	ANI0 to ANI7 ANI100 to ANI107	prohibited	ANI0 to ANI7 ANI100 to ANI107	AV_{REFn}	AV_{SS}
0	1	0	0	0	ANI8 ANI108	ANI0 to ANI8 ANI100 to ANI108	prohibited	ANI0 to ANI8 ANI100 to ANI108	AV_{REFn}	AV_{SS}
0	1	0	0	1	ANI9 ANI109	ANI0 to ANI9 ANI100 to ANI109	prohibited	ANI0 to ANI9 ANI100 to ANI109	AV_{REFn}	AV_{SS}
0	1	0	1	0	ANI10 ANI110	ANI0 to ANI10 ANI100 to ANI110	prohibited	ANI0 to ANI10 ANI100 to ANI110	AV_{REFn}	AV_{SS}
0	1	0	1	1	ANI11 ANI111	ANI0 to ANI11 ANI100 to ANI111	prohibited	ANI0 to ANI11 ANI100 to ANI111	AV_{REFn}	AV_{SS}

Table 21-7 Analog input selection settings

ADAnS4	ADAnS3	ADAnS2	ADAnS1	ADAnS0	Analog input to convert					
					ADAnDIAG = 0 (without diagnostic function)		ADAnDIAG = 1 (with diagnostic function)			
					Select mode	Scan mode	Select mode	Scan mode		
0	1	1	0	0	ANI12 ANI112	ANI0 to ANI12 ANI100 to ANI112	prohibited	ANI0 to ANI12 ANI100 to ANI112	AV_{REFn}	AV_{SS}
0	1	1	0	1	ANI13 ANI113	ANI0 to ANI13 ANI100 to ANI113	prohibited	ANI0 to ANI13 ANI100 to ANI113	AV_{REFn}	AV_{SS}
0	1	1	1	0	ANI14 ANI114	ANI0 to ANI14 ANI100 to ANI114	prohibited	ANI0 to ANI14 ANI100 to ANI114	AV_{REFn}	AV_{SS}
0	1	1	1	1	ANI15 ANI115	ANI0 to ANI15 ANI100 to ANI115	prohibited	ANI0 to ANI15 ANI100 to ANI115	AV_{REFn}	AV_{SS}
1	0	0	0	0	ANI16	ANI0 to ANI16	prohibited	ANI0 to ANI16	AV_{REFn}	AV_{SS}
1	0	0	0	1	ANI17	ANI0 to ANI17	prohibited	ANI0 to ANI17	AV_{REFn}	AV_{SS}
1	0	0	1	0	ANI18	ANI0 to ANI18	prohibited	ANI0 to ANI18	AV_{REFn}	AV_{SS}
1	0	0	1	1	ANI19	ANI0 to ANI19	prohibited	ANI0 to ANI19	AV_{REFn}	AV_{SS}
1	0	1	0	0	ANI20	ANI0 to ANI20	prohibited	ANI0 to ANI20	AV_{REFn}	AV_{SS}
1	0	1	0	1	ANI21	ANI0 to ANI21	prohibited	ANI0 to ANI21	AV_{REFn}	AV_{SS}
1	0	1	1	0	ANI22	ANI0 to ANI22	prohibited	ANI0 to ANI22	AV_{REFn}	AV_{SS}
1	0	1	1	1	ANI23	ANI0 to ANI23	prohibited	ANI0 to ANI23	AV_{REFn}	AV_{SS}
Other than above					Setting prohibited[a]					

a) When the channel in which an analog input does not exist is set, a conversion result becomes undefined.

(5) ADAnCRm, ADAnCRmH - ADC conversion result registers

The ADAnCRm and ADAnCRmH registers store 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 ADAnCRm register is specified, and when 8 bit access is performed, the ADAnCRmH register holding the upper 8 bits of the conversion result is specified

When reading the 10-bit data of the A/D conversion results from the ADAnCRm register, only the upper 10 bits are valid and the lower 6 bits are always read as 0.

Address

ADA0CR0:	FFFFF210$_H$	ADA0CR1:	FFFFF212$_H$
ADA0CR2:	FFFFF214$_H$	ADA0CR3:	FFFFF216$_H$
ADA0CR4:	FFFFF218$_H$	ADA0CR5:	FFFFF21A$_H$
ADA0CR6:	FFFFF21C$_H$	ADA0CR7:	FFFFF21E$_H$
ADA0CR8:	FFFFF220$_H$	ADA0CR9:	FFFFF222$_H$
ADA0CR10:	FFFFF224$_H$	ADA0CR11:	FFFFF226$_H$
ADA0CR12:	FFFFF228$_H$	ADA0CR13:	FFFFF22A$_H$
ADA0CR14:	FFFFF22C$_H$	ADA0CR15:	FFFFF22E$_H$
ADA0CR16:	FFFFF230$_H$	ADA0CR17:	FFFFF232$_H$
ADA0CR18:	FFFFF234$_H$	ADA0CR19:	FFFFF236$_H$
ADA0CR20:	FFFFF238$_H$	ADA0CR21:	FFFFF23A$_H$
ADA0CR22:	FFFFF23C$_H$	ADA0CR23:	FFFFF23E$_H$

ADA1CR0:	FFFFF250$_H$	ADA1CR1:	FFFFF252$_H$
ADA1CR2:	FFFFF254$_H$	ADA1CR3:	FFFFF256$_H$
ADA1CR4:	FFFFF258$_H$	ADA1CR5:	FFFFF25A$_H$
ADA1CR6:	FFFFF25C$_H$	ADA1CR7:	FFFFF25E$_H$
ADA1CR8:	FFFFF260$_H$	ADA1CR9:	FFFFF262$_H$
ADA1CR10:	FFFFF264$_H$	ADA1CR11:	FFFFF266$_H$
ADA1CR12:	FFFFF268$_H$	ADA1CR13:	FFFFF26A$_H$
ADA1CR14:	FFFFF26C$_H$	ADA1CR15:	FFFFF26E$_H$

Initial value undefined

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADAnCRm	AD9	AD8	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0	0	0	0	0	0	0
	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Address	ADA0CR0H: FFFFF211$_H$	ADA0CR1H: FFFFF213$_H$
	ADA0CR2H: FFFFF215$_H$	ADA0CR3H: FFFFF217$_H$
	ADA0CR4H: FFFFF219$_H$	ADA0CR5H: FFFFF21B$_H$
	ADA0CR6H: FFFFF21D$_H$	ADA0CR7H: FFFFF21F$_H$
	ADA0CR8H: FFFFF221$_H$	ADA0CR9H: FFFFF223$_H$
	ADA0CR10H: FFFFF225$_H$	ADA0CR11H: FFFFF227$_H$
	ADA0CR12H: FFFFF229$_H$	ADA0CR13H: FFFFF22B$_H$
	ADA0CR14H: FFFFF22D$_H$	ADA0CR15H: FFFFF22F$_H$
	ADA0CR16H: FFFFF231$_H$	ADA0CR17H: FFFFF233$_H$
	ADA0CR18H: FFFFF235$_H$	ADA0CR19H: FFFFF237$_H$
	ADA0CR20H: FFFFF239$_H$	ADA0CR21H: FFFFF23B$_H$
	ADA0CR22H: FFFFF23D$_H$	ADA0CR23H: FFFFF23F$_H$
	ADA1CR0H: FFFFF251$_H$	ADA1CR1H: FFFFF253$_H$
	ADA1CR2H: FFFFF255$_H$	ADA1CR3H: FFFFF257$_H$
	ADA1CR4H: FFFFF259$_H$	ADA1CR5H: FFFFF25B$_H$
	ADA1CR6H: FFFFF25D$_H$	ADA1CR7H: FFFFF25F$_H$
	ADA1CR8H: FFFFF261$_H$	ADA1CR9H: FFFFF263$_H$
	ADA1CR10H: FFFFF265$_H$	ADA1CR11H: FFFFF267$_H$
	ADA1CR12H: FFFFF269$_H$	ADA1CR13H: FFFFF26B$_H$
	ADA1CR14H: FFFFF26D$_H$	ADA1CR15H: FFFFF26F$_H$
Initial value	undefined	

	7	6	5	4	3	2	1	0
ADAnCRmH	AD9	AD8	AD7	AD6	AD5	AD4	AD3	AD2
	R	R	R	R	R	R	R	R

Caution When writing to the ADAnM0 to ADAnM2, ADAnS, ADAnPFM and the ADAnPFT register the contents of the ADAnCRm registers might become undefined. Therefore, after the conversion operation ends read the conversion result before writing to any of the above registers.
Moreover, when external/timer trigger is used, the content of the ADAnCRm register must be read before the following external/timer trigger is accepted.

The relationship between the analog voltage input to the analog input pins ANInmm and the A/D conversion result (of A/D conversion result register n ADAnCRm is as follows:

$$\mathrm{ADnCRm} = \mathrm{INT}(\frac{V_{IN}}{AV_{REFn}} \cdot 1024 + 0.5)$$

or

$$(\mathrm{ADAnCRm} - 0.5) \cdot \frac{AV_{REF0}}{1024} \le V_{IN} < (\mathrm{ADAnCRm} + 0.5) \cdot \frac{AV_{REFn}}{1024}$$

INT( ): Function that returns the integer of the value in ( )

V_{IN}: Analog input voltage at AINn pin

AV_{REFn}: AV_{REFn} pin voltage

ADAnCRm: Value of A/D conversion result register n (ADAnCRm)

Figure 21-2 shows the relationship between the analog input voltage and the A/D conversion results.

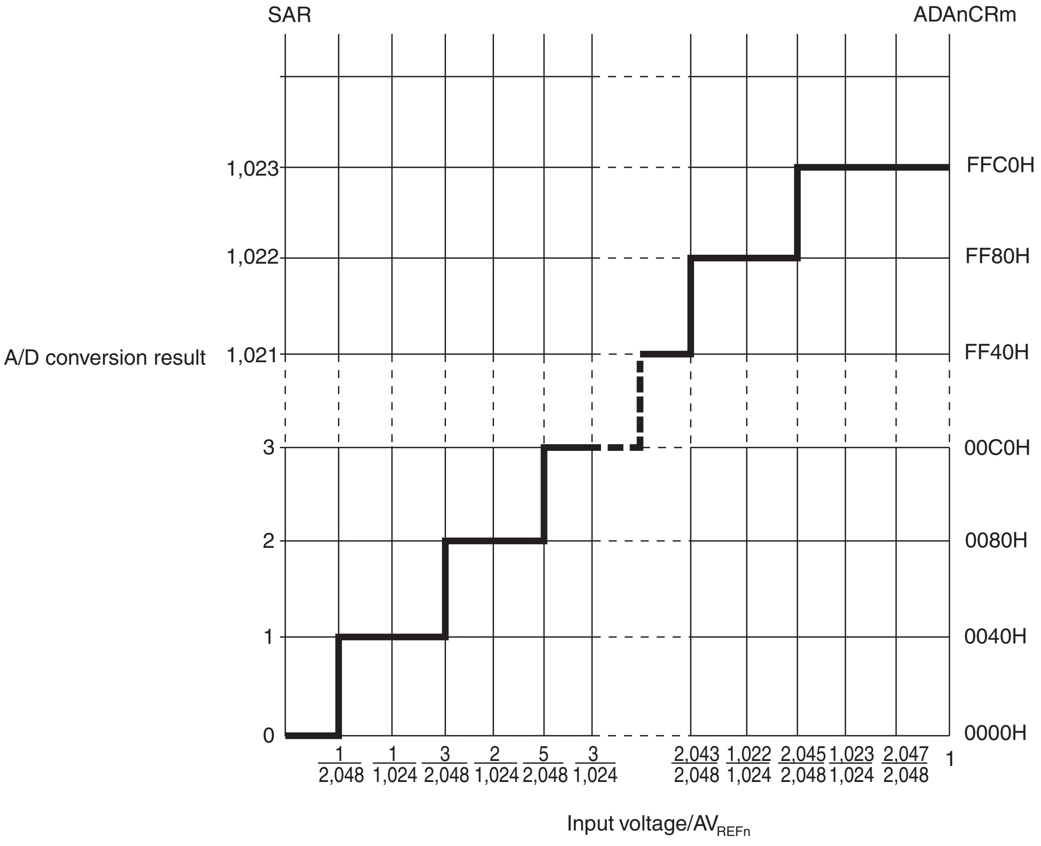


Figure 21-2 Relationship between analog input voltage and A/D conversion results

(6) ADAnCRDD, ADAnCRDDH - AV_{REF} A/D conversion diagnostic registers

The ADAnCRDD and ADAnCRDDH registers store the result of the AV_{REFn} conversion if the ADC diagnostic function is enabled (ADAnM2.ADAnDIAG = 1).

Access These registers are read-only in 16-bit or 8-bit units. When 16-bit access is performed, the ADAnCRDD register is specified, and when 8 bit access is performed, the ADAnCRDDH register holding the upper 8 bits of the conversion result is specified

When reading the 10-bit data of the A/D conversion results from the ADAnCRDD register, only the upper 10 bits are valid and the lower 6 bits are always read as 0.

Address ADA0CRDD: $FFFFF20C_H$
ADA1CRDD: $FFFFF24C_H$

Initial value undefined

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADAnCRDD	AD9	AD8	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0	0	0	0	0	0	0
	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Address ADA0CRDDH: $FFFFF20D_H$
ADA1CRDDH: $FFFFF24D_H$

Initial value undefined

	7	6	5	4	3	2	1	0
ADAnCRDDH	AD9	AD8	AD7	AD6	AD5	AD4	AD3	AD2
	R	R	R	R	R	R	R	R

Caution Since A/D conversion accuracy is influenced of use conditions, the result does not necessarily become all 1 (ADAnCRDD = $FFC0_H$) when converting AV_{REFn}.

(7) ADAnCRSS, ADAnCRSSH - AV_{SS} A/D conversion diagnostic registers

The ADAnCRSS and ADAnCRSSH registers store the result of the AV_{SS} conversion if the ADC diagnostic function is enabled (ADAnM2.ADAnDIAG = 1).

Access These registers are read-only in 16-bit or 8-bit units. When 16-bit access is performed, the ADAnCRSS register is specified, and when 8 bit access is performed, the ADAnCRSSH register holding the upper 8 bits of the conversion result is specified

When reading the 10-bit data of the A/D conversion results from the ADAnCRSS register, only the upper 10 bits are valid and the lower 6 bits are always read as 0.

Address ADA0CRSS: $FFFFF20E_H$
ADA1CRSS: $FFFFF24E_H$

Initial value undefined

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADAnCRSS	AD9	AD8	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0	0	0	0	0	0	0
	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Address ADA0CRSSH: $FFFFF20F_H$
ADA1CRSSH: $FFFFF24F_H$

Initial value undefined

	7	6	5	4	3	2	1	0
ADAnCRSSH	AD9	AD8	AD7	AD6	AD5	AD4	AD3	AD2
	R	R	R	R	R	R	R	R

Caution Since A/D conversion accuracy is influenced of use conditions, the result does not necessarily become all 0 (ADAnCRSS = $003F_H$) when converting AV_{SS}.

(8) ADAnPFM - ADC power-fail compare mode register

The ADAnPFM register is an 8-bit register that sets the power-fail compare mode.

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

Reset input clears this register to 00H.

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

Address ADA0PFM: FFFFF204$_H$
ADA1PFM: FFFFF244$_H$

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

	7	6	5	4	3	2	1	0
ADAnPFM	ADAnPFE	ADAnPFC	0	0	0	0	0	0
	R/W	R/W	R	R	R	R	R	R

Table 21-8 ADAnPFM register contents

Bit position	Bit name	Function
7	ADAnPFE	Power-fail compare control 0: Power-fail compare disabled. 1: Power-fail compare enabled.
6	ADAnPFC	Power-fail compare mode 0: Generates interrupt request INTAD/INTAD1 if ADAnCRmH ≥ ADAnPFT. 1: Generates interrupt request INTAD/INTAD1 if ADAnCRmH < ADAnPFT. **Caution:** **1.** In the select mode, the 8-bit data set to the ADAnPFT register is compared with the value of the ADAnCRmH register specified by the ADAnS register. If the result matches the condition specified by the ADAnPFC bit, the conversion result is stored in the ADAnCRm register and the INTAD/INTAD1 signal is generated. If it does not match, however, the interrupt signal is not generated. **2.** In the scan mode, the 8-bit data set to the ADAnPFT register is compared with the contents of the ADAnCR0H register. If the result matches the condition specified by the ADAnPFC bit, the conversion result is stored in the ADAnCR0 register and the INTAD/INTAD1 signal is generated. If it does not match, however, the INTAD/INTAD1 signal is not generated. Regardless of the comparison result, the scan operation is continued and the conversion result is stored in the ADAnCRm register until the scan operation is completed. However, the INTAD/INTAD1 signal is not generated after the scan operation has been completed.

(9) ADAnPFT - ADC power-fail compare threshold value register

The ADAnPFT register sets the compare value in the power-fail compare mode.

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

Reset input clears this register to 00H.

Address ADA0PFT: FFFFF205$_H$
ADA1PFT: FFFFF245$_H$

Initial value 00$_H$. This register is cleared by any reset.

	7	6	5	4	3	2	1	0
ADAnPFT	ADAnPFT7	ADAnPFT6	ADAnPFT5	ADAnPFT4	ADAnPFT3	ADAnPFT2	ADAnPFT1	ADAnPFT0
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

21.4 Operation

21.4.1 Basic operation

1. Set the operation mode, trigger mode, and conversion time for executing A/D conversion by using the ADAnM0, ADAnM1, ADAnM2, and ADAnS registers.
   Set the ADAnM0.ADAnPS bit to supply power to the analog circuitry of the ADC. Do not enable AD conversion before the ADC stabilization time is elapsed. For the stabilization time refer to the Datasheet.
   When the ADAnM0.ADAnCE bit is set, conversion is started in the software trigger mode and the A/D Converter waits for a trigger in the external or timer trigger mode.
2. When A/D conversion is started, the voltage input to the selected analog input channel is sampled by the sample & hold circuit.
3. When the sample & hold circuit samples the input channel for a specific time, it enters the hold status, and holds the input analog voltage until A/D conversion is complete.
4. Set bit 9 of the successive approximation register (SAR). The tap selector selects (1/2) AV_{REFn} as the voltage tap of the series resistor string.
5. The voltage difference between the voltage of the series resistor string and the analog input voltage is compared by the Voltage Comparator. If the analog input voltage is higher than (1/2) AV_{REFn}, the MSB of the SAR register remains set. If it is lower than (1/2) AV_{REFn}, the MSB is reset.
6. Next, bit 8 of the SAR register is automatically set and the next comparison is started. Depending on the value of bit 9, to which a result has been already set, the voltage tap of the series resistor string is selected as follows:
   –Bit 9 = 1: (3/4) AV_{REFn}
   –Bit 9 = 0: (1/4) AV_{REFn}

   This voltage tap and the analog input voltage are compared and, depending on the result, bit 8 is manipulated as follows.

   Analog input voltage ≥ Voltage tap: Bit 8 = 1

   Analog input voltage ≤ Voltage tap: Bit 8 = 0
7. This comparison is continued to bit 0 of the SAR register.
8. When comparison of the 10 bits is complete, the valid digital result is stored in the SAR register, which is then transferred to and stored in the ADAnCRm register. At the same time, an A/D conversion end interrupt request signal (INTAD/INTAD1) is generated.

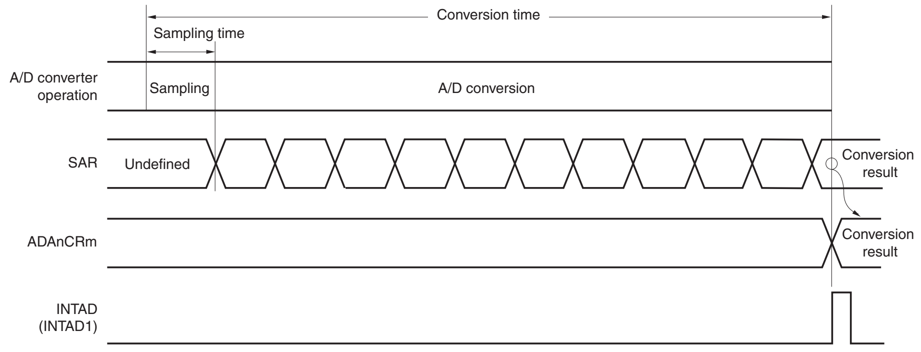


Figure 21-3 A/D Converter basic operation

21.4.2 Trigger mode

The timing of starting the conversion operation is specified by setting a trigger mode. The trigger mode includes a software trigger mode and hardware trigger modes. The hardware trigger modes include timer trigger modes 0 and 1, and external trigger mode. The ADAnM0 .ADAnTMD bit is used to set the trigger mode. In timer trigger mode set ADAnM2.ADAnTMD[1:0] = 01_B.

(1) Software trigger mode

When the ADAnM0.ADAnCE bit is set to 1, the signal of the analog input pin ANInmm specified by the ADAnS register is converted. When conversion is complete, the result is stored in the ADAnCRm register. At the same time, the A/D conversion end interrupt request signal (INTAD/INTAD1) is generated.

If the operation mode specified by the ADAnM0-ADAnMD[1:0] bits is the continuous select/scan mode, the next conversion is started, unless the ADAnCE bit is cleared to 0 after completion of the first conversion.

When conversion is started, the ADAnEF bit is set to 1 (indicating that conversion is in progress).

If the ADAnM0, ADAnM2, ADAnS, ADAnPFM, or ADAnPFT register is written during conversion, the conversion is aborted and started again from the beginning.

(2) External trigger mode

In this mode, converting the signal of the analog input pin (ANI0 to ANI23) specified by the ADAnS register is started when an external trigger is input (to the ADTRG pin). Which edge of the external trigger is to be detected (i.e., the rising edge, falling edge, or both rising and falling edges) can be specified by using the ADAnM0.ADAnETS]1:0] bits. When the ADAnM0.ADAnCE bit set to 1, the A/D Converter waits for the trigger, and starts conversion after the external trigger has been input.

When conversion is completed, the result of conversion is stored in the ADAnCRm register. At the same time, the A/D conversion end interrupt request signal (INTAD/INTAD1) is generated, and the A/D Converter waits for the trigger again.

When conversion is started, the ADAnEF bit is set to 1 (indicating that conversion is in progress). While the A/D Converter is waiting for the trigger, however, the ADAnEF bit is cleared to 0 (indicating that conversion is stopped). If the valid trigger is input during the conversion operation, the conversion is aborted and started again from the beginning.

If the ADAnM0, ADAnM2, ADAnS, ADAnPFM, or ADAnPFT register is written during the conversion operation, the conversion is not aborted, and the A/D Converter waits for the trigger again.

(3) Timer trigger mode

In this mode, converting the signal of the analog input pin ANInmm, specified by the ADAnS register, is started by any of the timer output signals INTTAA2CC0, INTTAA2CC1 or TQTADT0. The timer output signal is selected by the ADAnM2.ADAnTMD[1:0] bits, and conversion is started at the rising edge of the timer output signal. When the ADAnM0.ADAnCE bit is set to 1, the A/D Converter waits for a trigger, and starts conversion when the rising edge of the timer output signal is input.

When conversion is completed, the result of the conversion is stored in the ADAnCRm register. At the same time, the A/D conversion end interrupt request signal (INTAD/INTAD1) is generated, and the A/D Converter waits for the trigger again.

When conversion is started, the ADAnEF bit is set to 1 (indicating that conversion is in progress). While the A/D Converter is waiting for the trigger, however, the ADAnEF bit is cleared to 0 (indicating that conversion is stopped). If the valid trigger is input during the conversion operation, the conversion is aborted and started again from the beginning.

If the ADAnM0, ADAnM2, ADAnS, ADAnPFM, or ADAnPFT register is written during conversion, the conversion is stopped and the A/D Converter waits for the trigger again.

21.4.3 Operation modes

Four operation modes are available as the modes in which to set the ANInmm pins: continuous select mode, continuous scan mode, one-shot select mode and one-shot scan mode.

The operation mode is selected by the ADAnM0.ADAnMD[1:0] bits.

(1) Continuous select mode

In this mode, the voltage of one analog input pin selected by the ADAnS register is continuously converted into a digital value.

The conversion result is stored in the ADAnCRm register corresponding to the analog input pin. In this mode, an analog input pin corresponds to an ADAnCRm register on a one-to-one basis. Each time A/D conversion is completed, the A/D conversion end interrupt request signal (INTAD/INTAD1) is generated. After completion of conversion, the next conversion is started, unless the ADAnCE bit of the ADAnM0 register is cleared to 0.

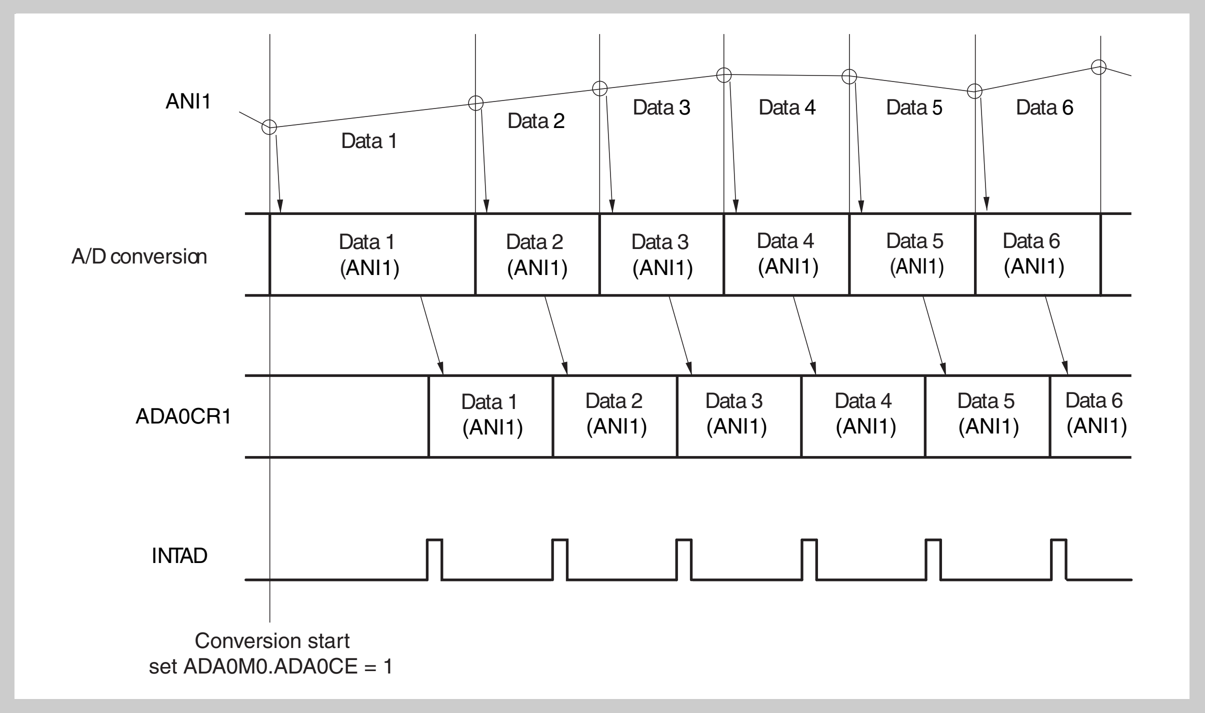


Figure 21-4 Timing example of continuous select mode operation (ADA0S = 01_H)

(2) Continuous scan mode

In this mode, analog input pins are sequentially selected, from the ANI0 pin to the pin specified by the ADAnS register, and their values are converted into digital values.

The result of each conversion is stored in the ADAnCRm register corresponding to the analog input pin. When conversion of the analog input pin specified by the ADAnS register is complete, the A/D conversion end interrupt request signal (INTAD/INTAD1) is generated, and A/D conversion is started again from the ANI0 pin, unless the ADAnM0.ADAnCE bit is cleared to 0.

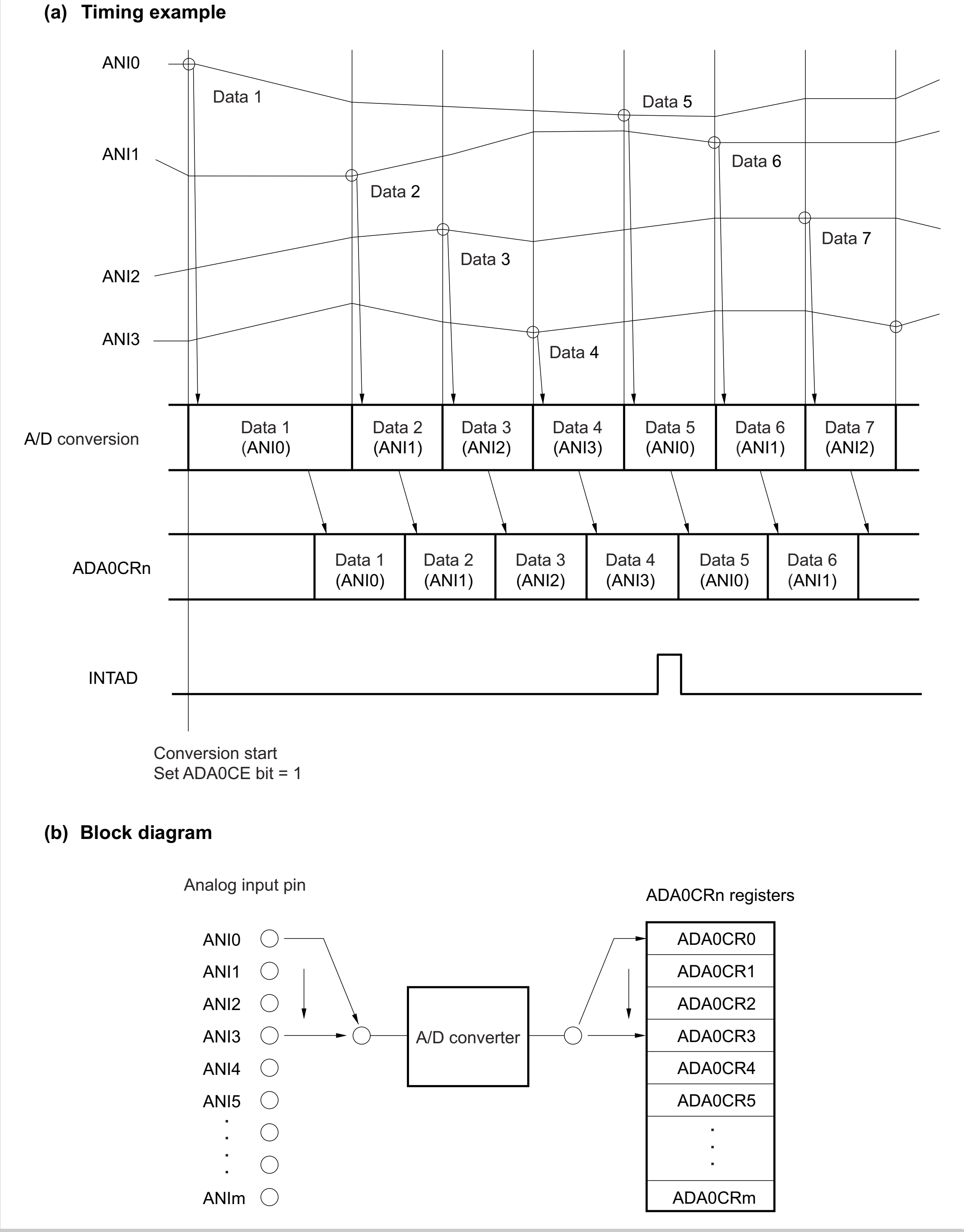


Figure 21-5 Timing example of continuous scan mode operation (ADA0S register = 03_H)

(3) One-shot select mode

In this mode, the voltage on the analog input pin specified by the ADA0S register is converted into a digital value only once. The conversion result is stored in the ADA0CRn register corresponding to the analog input pin.
In this mode, an analog input pin and an ADA0CRn register correspond on a one-to-one basis. When A/D conversion has been completed once, the INTAD signal is generated. The A/D conversion operation is stopped after it has been completed.

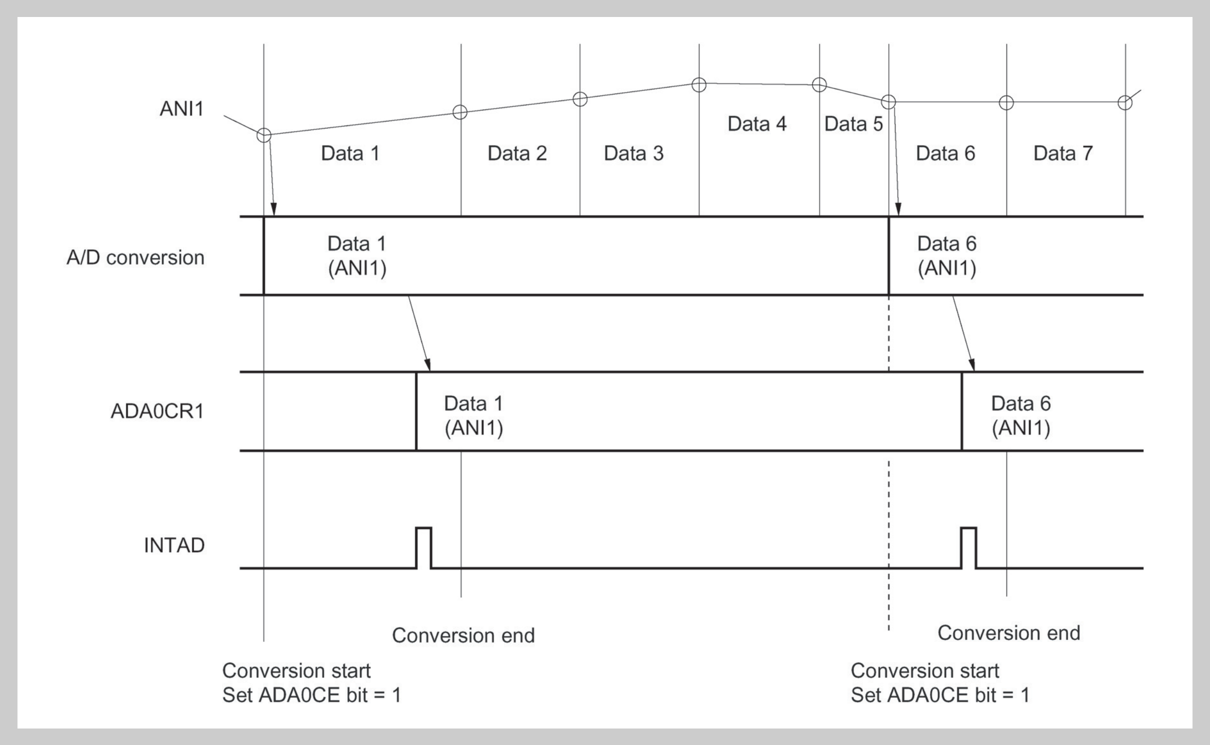


Figure 21-6 Timing example of one-shot select mode operation (ADA0S Register = 01_H)

(4) One-shot scan mode

In this mode, analog input pins are sequentially selected, from the ANI0 pin to the pin specified by the ADAnS register, and their values are converted into digital values. The result of each conversion is stored in the ADAnCRm register corresponding to the analog input pin. When conversion of the analog input pin specified by the ADAnS register is complete, the A/D conversion end interrupt request signal (INTAD/INTAD1) is generated, and A/D conversion is stopped.(n = 0 to 23).

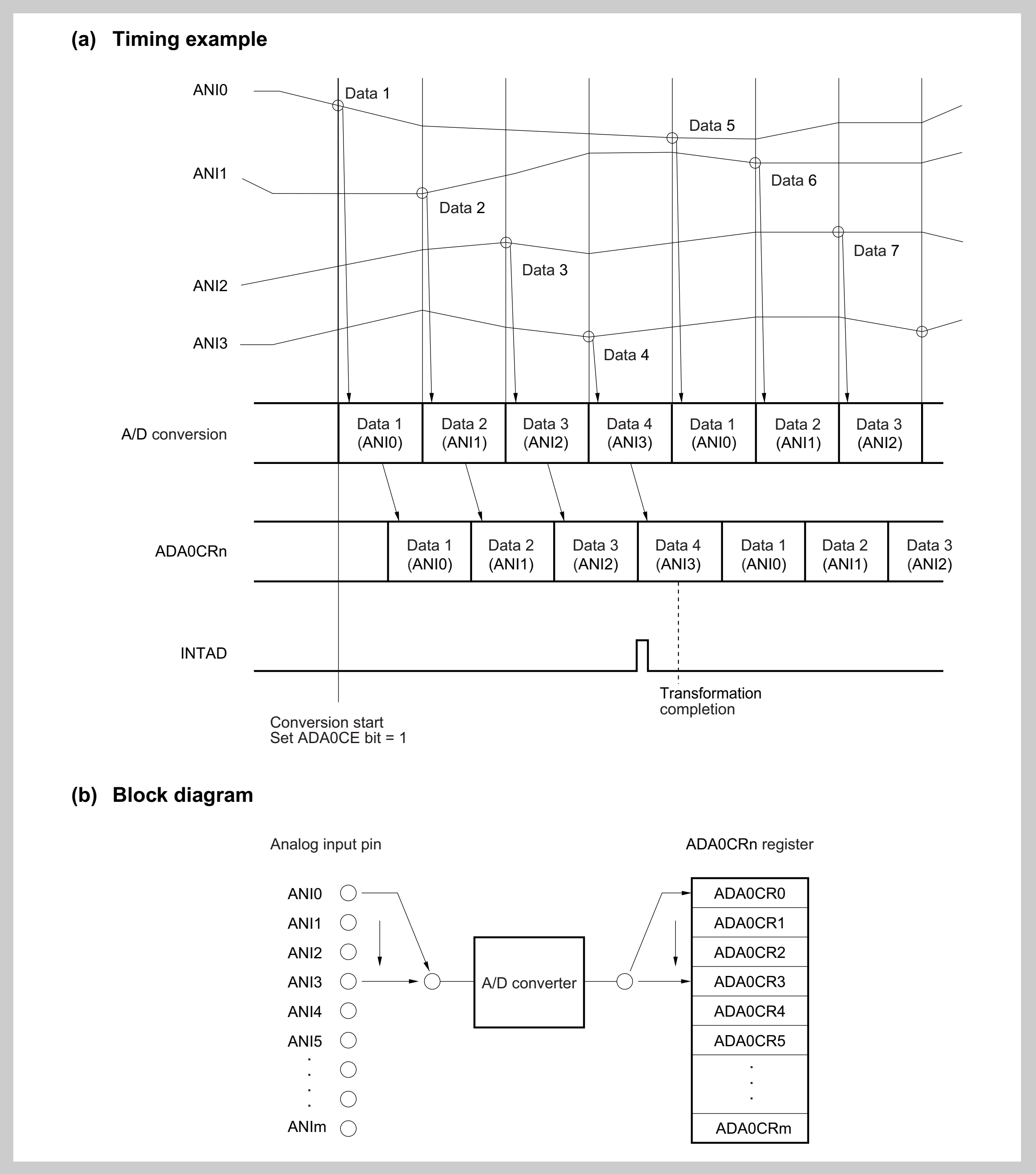


Figure 21-7 Timing example of one-shot scan mode operation (ADA0S Register = 03_H)

(5) Diagnostic mode

When activating the diagnostic mode (ADAnM2.ADADIAG = 1) the voltage at the AV_{REFn} pin and the AV_{SS} pin are sampled after conversion of the specified ANInm range is finished.

The resulting values can be found in the ADAnCRDD, ADAnCRDDH, ADAnCRSS and ADAnCRSSH registers.

Since AD conversion accuracy is influenced of use conditions, the result does not necessarily become all 1 when converting AV_{REFn}.

Since AD conversion accuracy is influenced of use conditions, the result does not necessarily become all 0 when converting AV_{SS}.

(6) Discharge mode

When activating the discharge mode (ADAnM2.ADADISC = 1) the internal capacitors of the sample and hold circuit are discharged prior to every conversion.

Additional 4 clocks must therefore be added to every conversion.

21.4.4 Power-fail compare mode

The A/D conversion end interrupt request signal (INTAD/INTAD1) can be controlled as follows by the ADAnPFM and ADAnPFT registers.

- When the ADAnPFE bit = 0, the INTAD/INTAD1 signal is generated each time conversion is completed (normal use of the A/D Converter).
- When the ADAnPFE bit = 1 a conversion result register is compared with the value of the ADAnPFT register.
  Which conversion result register is compared depends on the selected mode (see description below).
- When the ADAnPFC bit = 0 the INTAD/INTAD1 signal is generated when the conversion result register = ADAnPFT.
- When the ADAnPFC bit = 1 the INTAD/INTAD1 signal is generated when the conversion result register < ADAnPFT.

In the power-fail compare mode, four modes are available as modes in which to set the ANInm pins: continuous select mode, continuous scan mode, one-shot select mode and one-shot scan mode.

(1) Continuous select mode

In this mode, the result of converting the voltage of the analog input pin specified by the ADAnS register is compared with the set value of the ADAnPFT register. If the result of power-fail comparison matches the condition set by the ADAnPFC bit, the conversion result is stored in the ADAnCRm register, and the INTAD/INTAD1 signal is generated. If it does not match, the conversion result is stored in the ADAnCRm register, and the INTAD/INTAD1 signal is not generated. After completion of the first conversion, the next conversion is started, unless the ADAnM0.ADAnCE bit is cleared to 0.

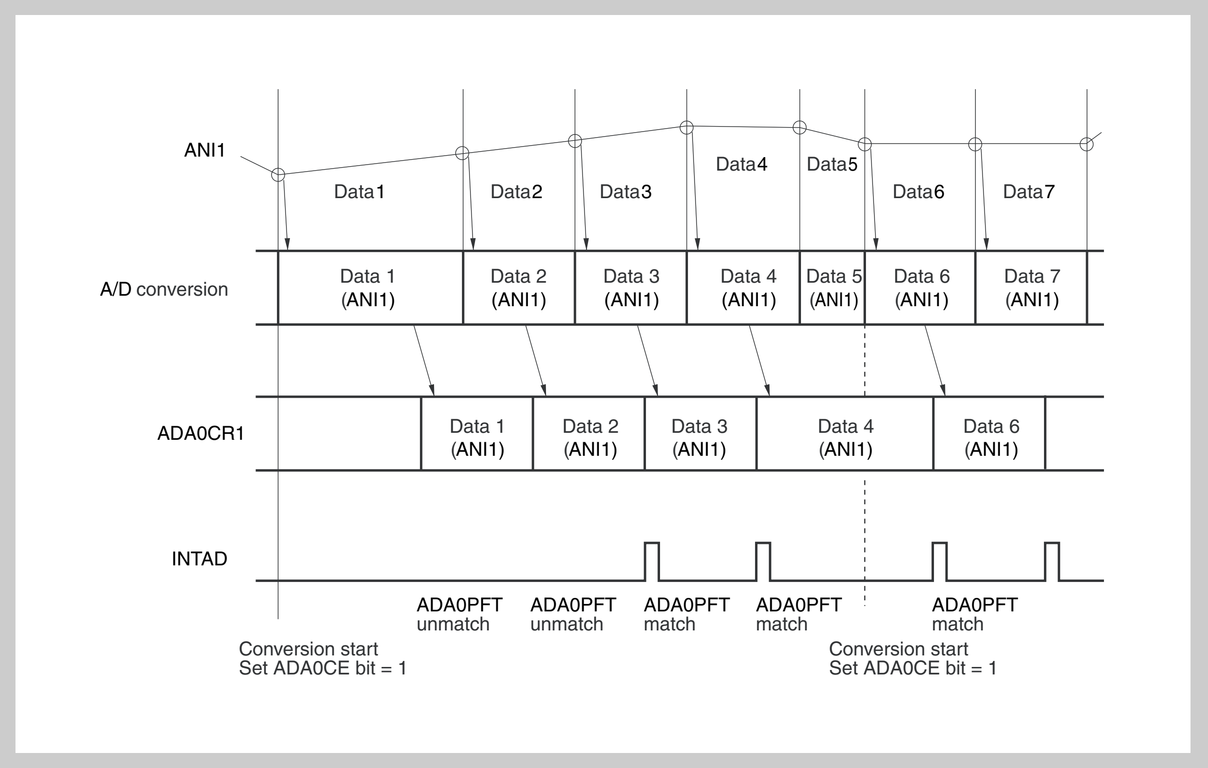


Figure 21-8 Timing example of continuous select mode operation (when power-fail comparison is made: ADA0S register = 01_H)

(2) Continuous scan mode

In this mode, the results of converting the voltages of the analog input pins sequentially selected from the ANI0 pin to the pin specified by the ADAnS register are stored, and the set value of the ADAnCR0H register of channel 0 is compared with the value of the ADAnPFT register. If the result of power-fail comparison matches the condition set by the ADAnPFC bit of the ADAnPFM register, the conversion result is stored in the ADAnCR0 register, and the INTAD/INTAD1 signal is generated. If it does not match, the conversion result is stored in the ADAnCR0 register, and the INTAD/INTAD1 signal is not generated.

After the result of the first conversion has been stored in the ADAnCR0 register, the results of sequentially converting the voltages on the analog input pins up to the pin specified by the ADAnS register are continuously stored. After completion of conversion, the next conversion is started from the ANI0 pin again, unless the ADAnCE bit of the ADAnM0 register is cleared to 0.

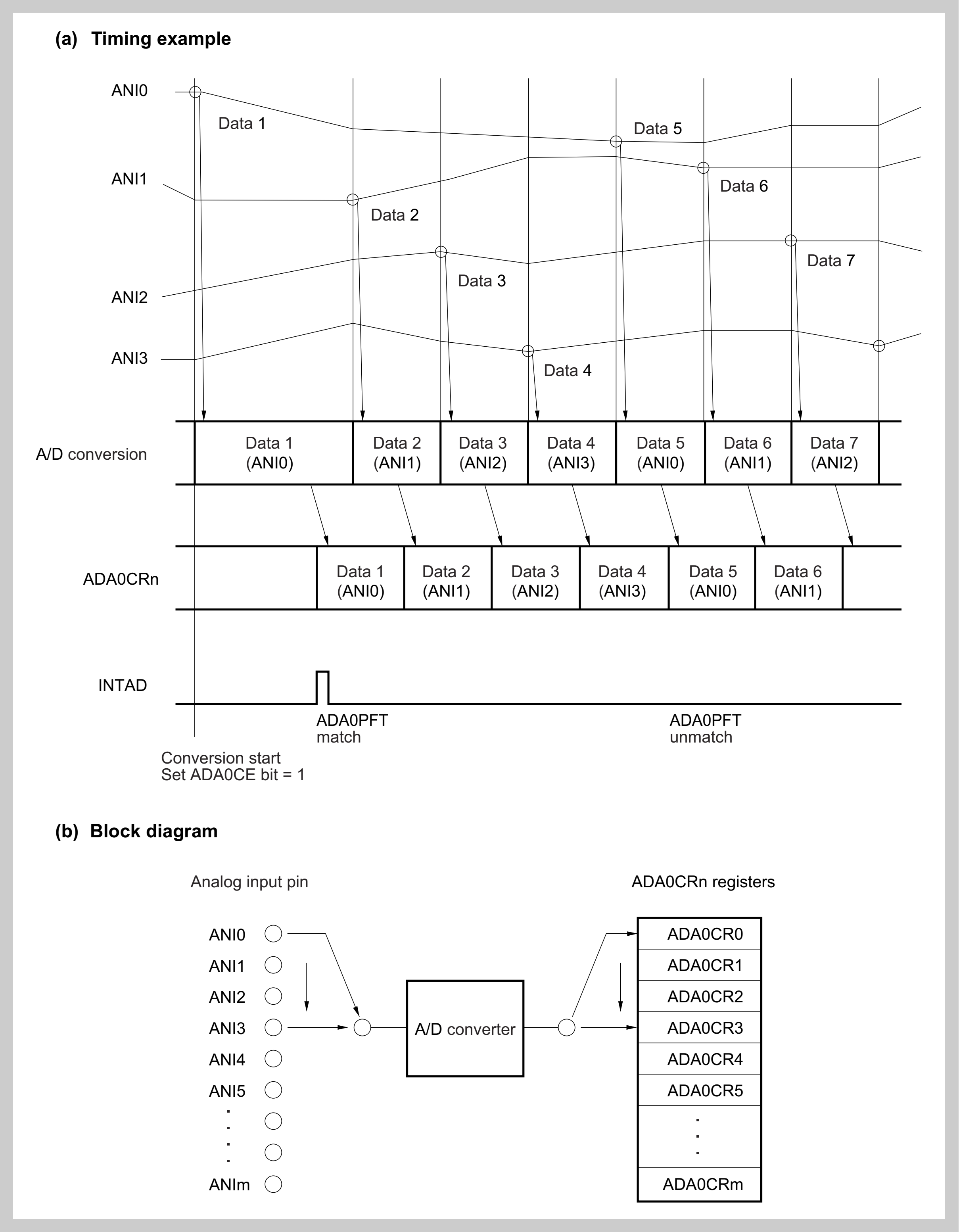


Figure 21-9 Timing example of continuous scan mode operation (when power-fail comparison is made: ADA0S register = 03_H)

(3) One-shot select mode

In this mode, the result of converting the voltage of the analog input pin specified by the ADA0S register is compared with the set value of the ADA0PFT register. If the result of power-fail comparison matches the condition set by the ADA0PFC bit, the conversion result is stored in the ADA0CRn register, and the INTAD signal is generated. If it does not match, the conversion result is stored in the ADA0CRn register, and the INTAD signal is not generated. Conversion is stopped after it has been completed.

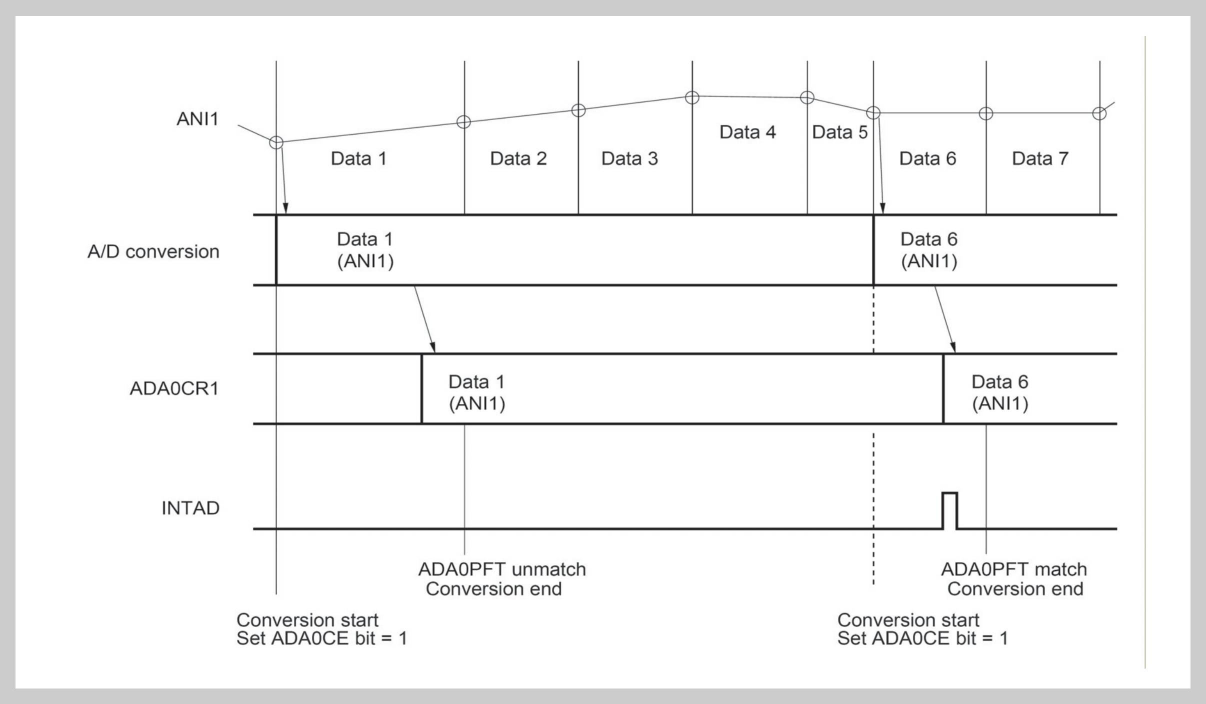

ANI1
Data 1
Data 2
Data 3
Data 4
Data 5
Data 6
Data 7
A/D conversion
Data 1 (ANI1)
Data 6 (ANI1)
ADA0CR1
Data 1 (ANI1)
Data 6 (ANI1)
INTAD
ADA0PFT unmatch Conversion end
ADA0PFT match Conversion end
Conversion start Set ADA0CE bit = 1
Conversion start Set ADA0CE bit = 1

Figure 21-10 Timing example of one-shot select mode operation (when power-fail comparison is made: ADA0S register = 01_H)

(4) One-shot scan mode

In this mode, the results of converting the voltages of the analog input pins sequentially selected from the ANI0 pin to the pin specified by the ADAnS register are stored, and the set value of the ADAnCR0H register of channel 0 is compared with the value of the ADAnPFT register. If the result of power-fail comparison matches the condition set by the ADAnPFM.ADAnPFC bit, the conversion result is stored in the ADAnCR0 register, and the INTAD/INTAD1 signal is generated. If it does not match, the conversion result is stored in the ADAnCR0 register, and the INTAD/INTAD1 signal is not generated.

After the result of the first conversion has been stored in the ADAnCR0 register, the results of sequentially converting the voltages on the analog input pins up to the pin specified by the ADAnS register are continuously stored.

After completion of conversion, A/D conversion is stopped. The 1st conversion result after A/D conversion has been ignored, because it is not good.

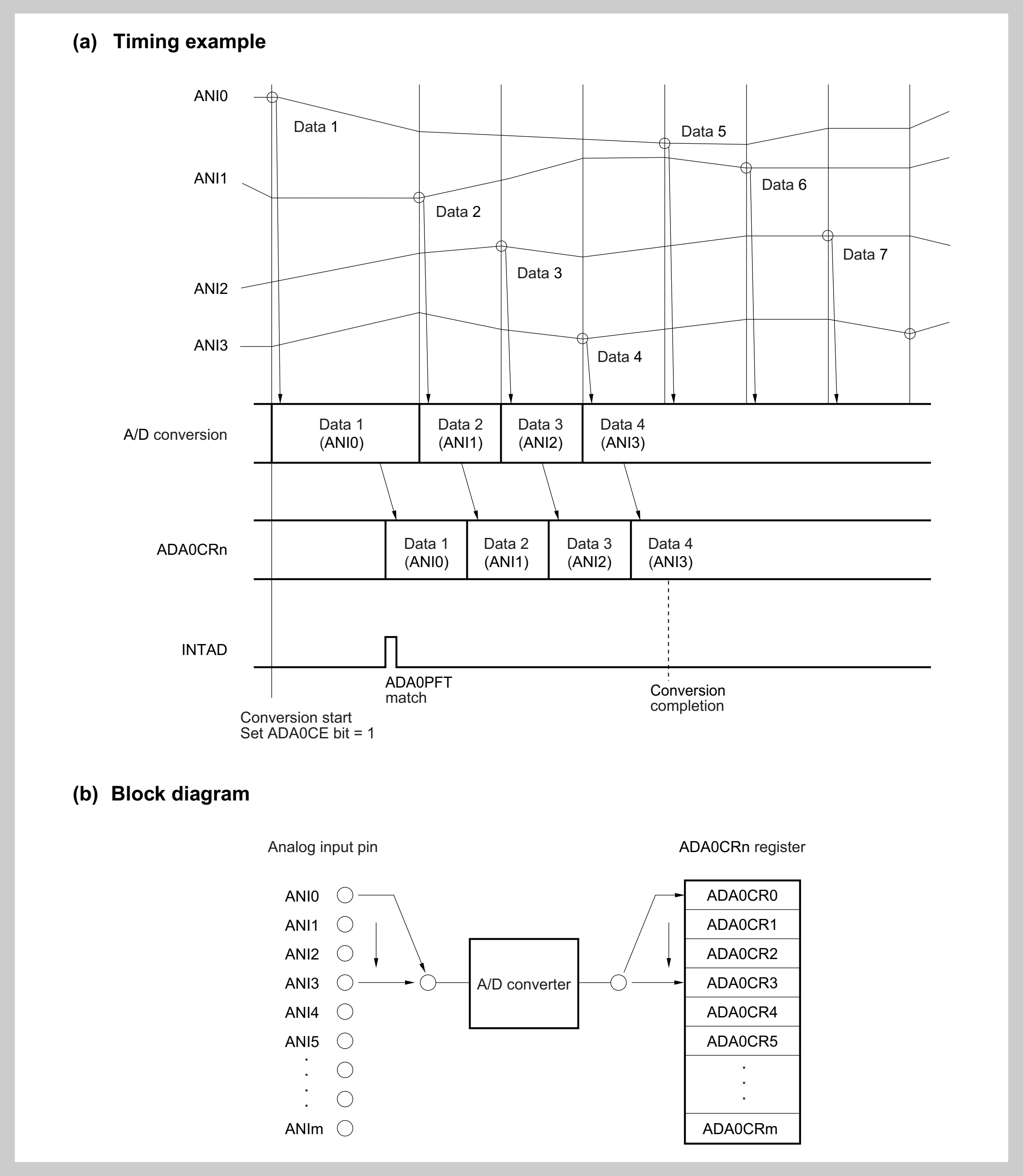


Figure 21-11 Timing Example of One-shot Scan Mode Operation (When Power-Fail Comparison Is Made: ADA0S Register = 03_H)

21.5 Cautions

(1) When A/D Converter is not used

When the A/D Converter is not used, the power consumption can be reduced by clearing the ADAnCE bit and the ADAnPS bit of the ADAnM0 register to 0.

(2) Input range of ANInm pins

Input the voltage within the specified range to the ANInm pins. If a voltage equal to or higher than AV_{REFn} or equal to or lower than AV_{SS} (even within the range of the absolute maximum ratings) is input to any of these pins, the conversion value of that channel is undefined and the conversion value of the other channels may also be affected.

(3) Countermeasures against noise

To maintain the 10-bit resolution, the ANInm pins must be effectively protected from noise. The influence of noise increases as the output impedance of the analog input source becomes higher. To lower the noise, connecting an external capacitor as shown in *Figure 21-12* is recommended.

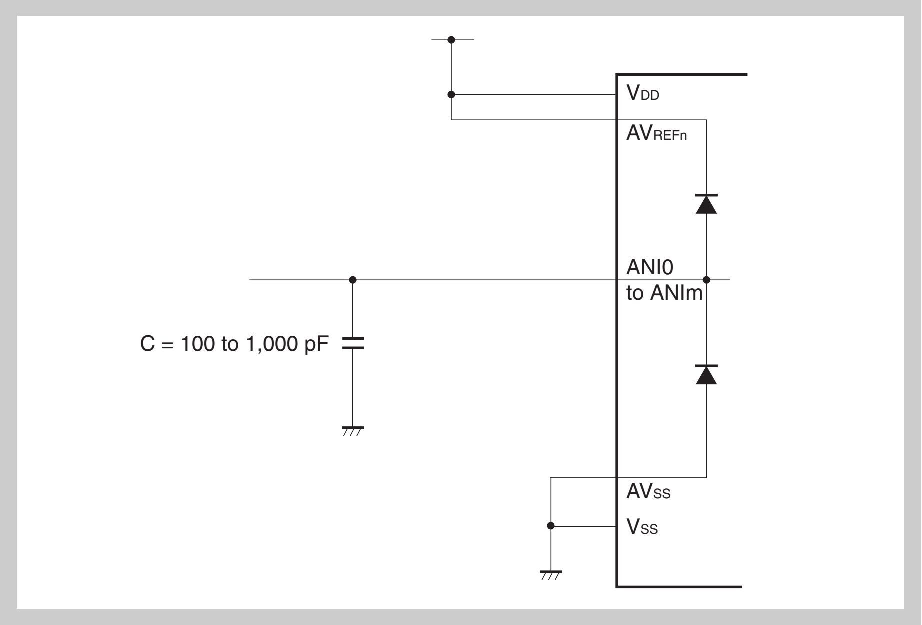


Figure 21-12 Processing of analog input pin

(4) Alternate I/O

The analog input pins ANInm function alternately as port pins. Changing the digital input/output function (PMCn and PMn; n = 2, 7 or 12) or changing the level of one or more output ports (Pnm; n = 2, 7 or 12; m = 0 up to 15) while ADA0CE bit = 1 could degrade the conversion accuracy.

For the output port the potential degradation increases with the driven total output current. Also the conversion resolution may drop if the output current fluctuates due to the effect of the external circuit connected to the port pins.

(5) Interrupt request flag (ADIF)

The interrupt request flag (ADIF) is not cleared even if the contents of the ADAnS register are changed. If the analog input pin is changed during A/D conversion, therefore, the result of converting the previously selected analog input signal may be stored and the conversion end interrupt request flag may be set immediately before the ADAnS register is rewritten. If the ADIF flag is read immediately after the ADAnS register is rewritten, the ADIF flag may be set even though the A/D conversion of the newly selected analog input pin has not been completed. When A/D conversion is stopped, clear the ADIF flag before resuming conversion.

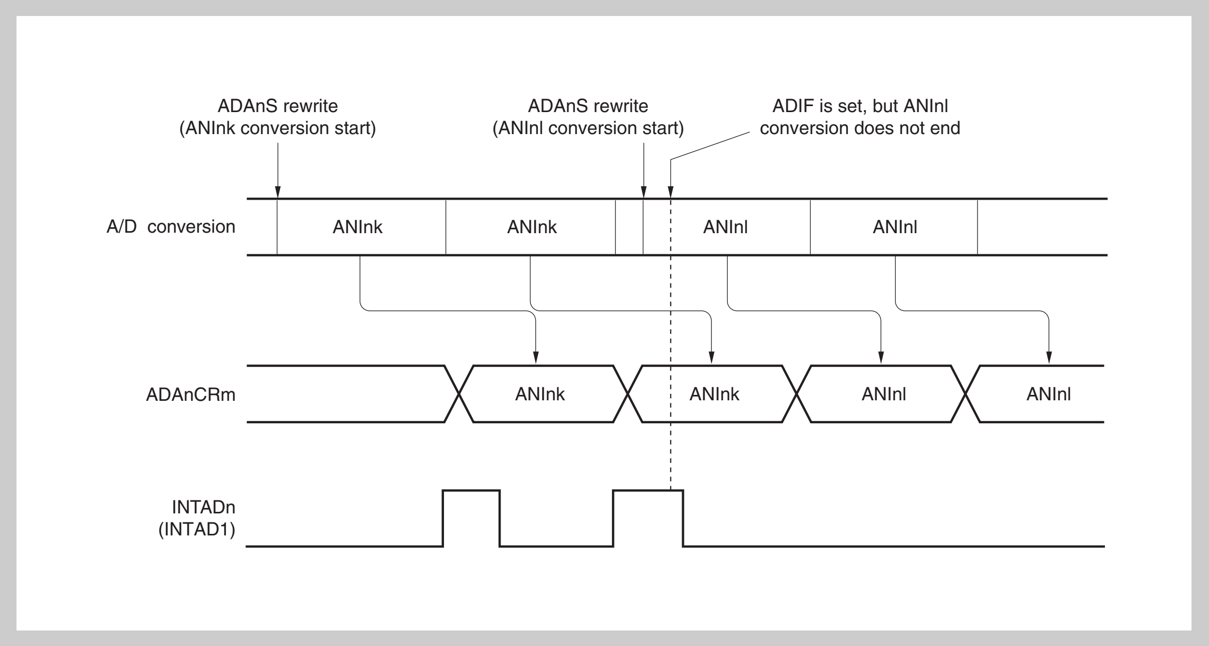


Figure 21-13 Generation timing of A/D conversion end interrupt request

(6) Reading ADAnCRm register

When the ADAnM0 to ADAnM2, ADAnS, ADAnPFM or ADAnPFT register is written, the contents of the ADAnCRm register may be undefined. Read the conversion result after completion of conversion and before writing to the ADAnM0 to ADAnM2, ADAnS, ADAnPFM or ADAnPFT registers. The correct conversion result may not be read at a timing different from the above.

Also, when an external/timer trigger is acknowledged, the contents of the ADAnCRm register may be undefined. Read the conversion result after completion of conversion and before the next external/timer trigger is acknowledged

21.6 How to read A/D Converter characteristics table

This section describes the terms related to the A/D Converter.

For detailed specifications refer to the Datasheet

(1) Resolution

The minimum analog input voltage that can be recognized, i.e., the ratio of an analog input voltage to 1 bit of digital output is called 1 LSB (least significant bit). The ratio of 1 LSB to the full scale is expressed as %FSR (full-scale range). %FSR is the ratio of a range of convertible analog input voltages expressed as a percentage, and can be expressed as follows, independently of the resolution.

$$\begin{aligned} 1\%\mathrm{FSR} &= (\text{Maximum value of convertible analog input voltage} - \text{Minimum value of convertible analog input voltage})/100 \\ &= (AV_{REFn} - 0)/100 \\ &= AV_{REFn}/100 \end{aligned}$$

When the resolution is 10 bits, 1 LSB is as follows:

$$\begin{aligned} 1\ \mathrm{LSB} &= 1/2^{10} = 1/1{,}024 \\ &= 0.098\%\mathrm{FSR} \end{aligned}$$

The accuracy is determined by the overall error, independently of the resolution.

(2) Overall error

This is the maximum value of the difference between an actually measured value and a theoretical value. It is a total of zero-scale error, full-scale error, linearity error, and a combination of these errors.

The overall error in the characteristics table does not include the quantization error.

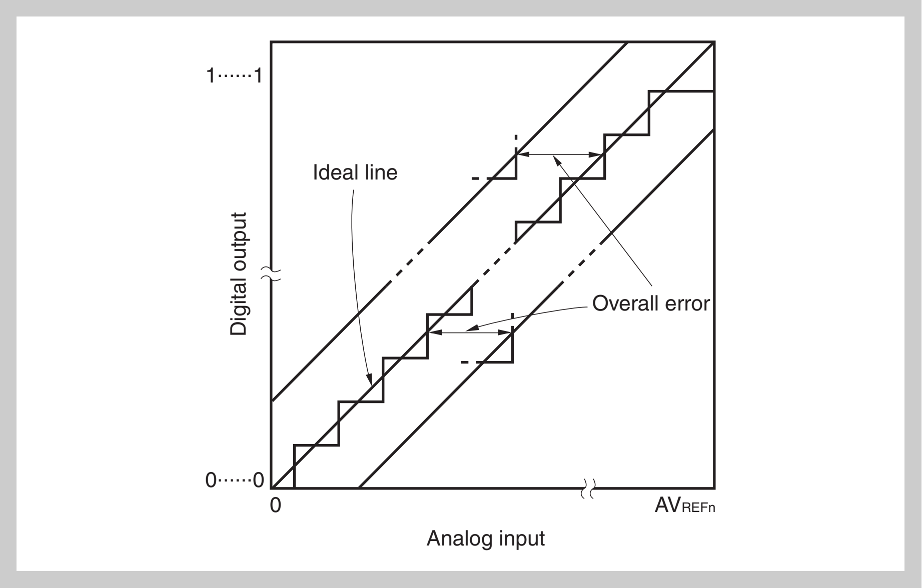


Figure 21-14 Overall error

(3) Quantization error

This is an error of ±1/2 LSB that inevitably occurs when an analog value is converted into a digital value. Because the A/D Converter converts analog input voltages in a range of ±1/2 LSB into the same digital codes, a quantization error is unavoidable.

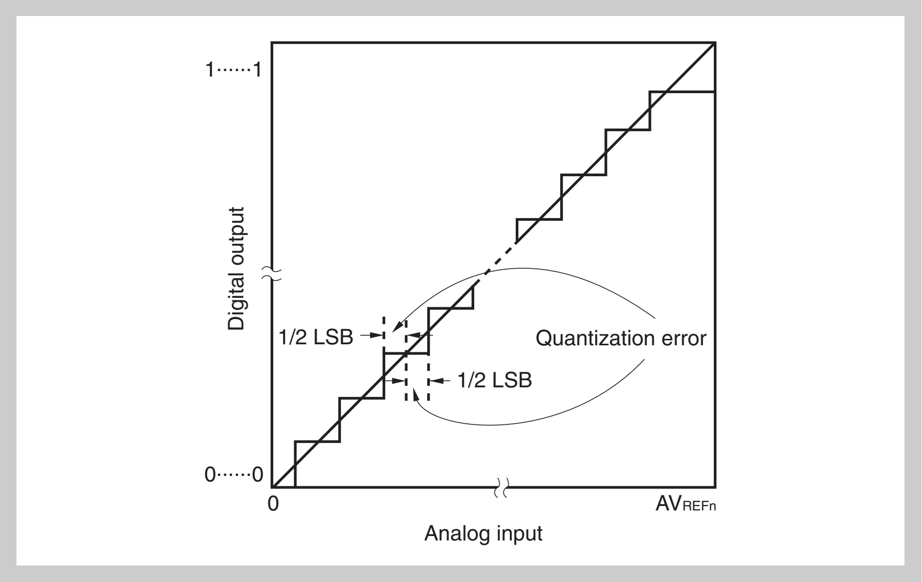


Figure 21-15 Quantization error

(4) Zero-scale error

This is the difference between the actually measured analog input voltage and its theoretical value when the digital output changes from 0…000$_B$ to 0…001$_B$ (1/2 LSB).

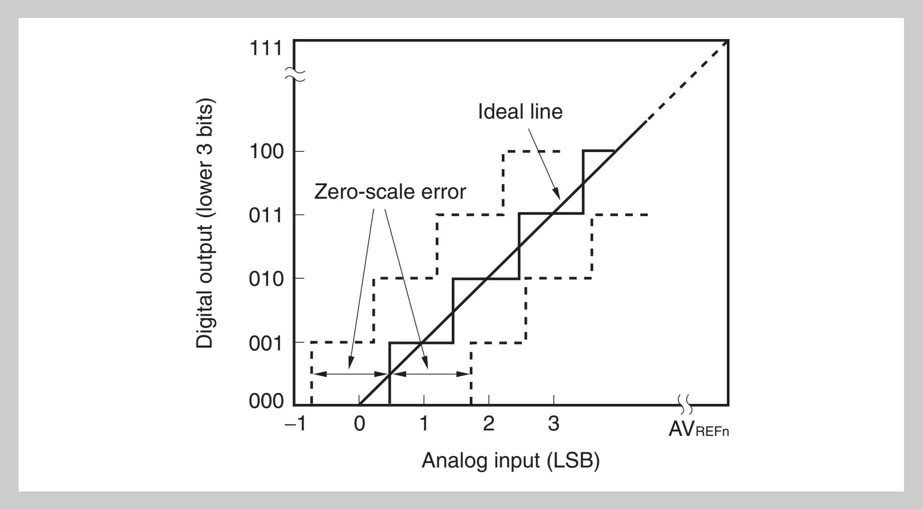


Figure 21-16 Zero-scale error

(5) Full-scale error

This is the difference between the actually measured analog input voltage and its theoretical value when the digital output changes from 1…110$_B$ to 0…111$_B$ (full scale - 3/2 LSB).

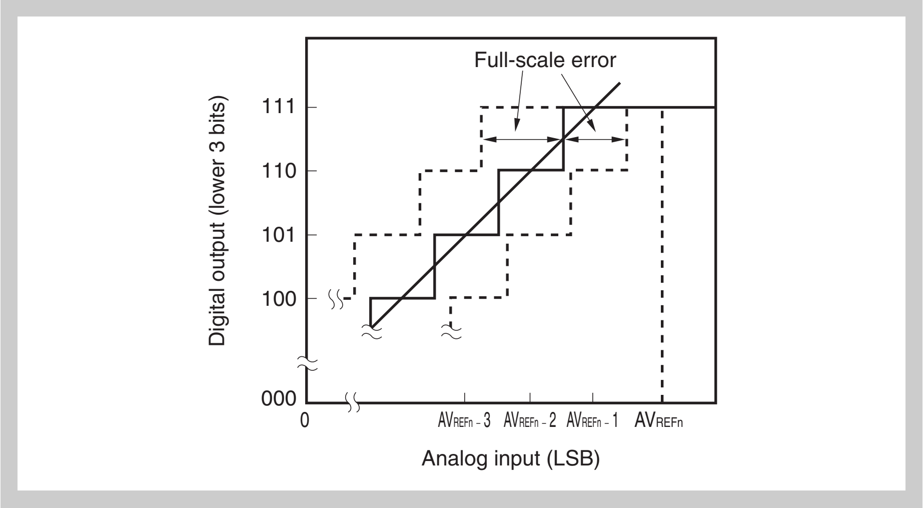


Figure 21-17 Full-scale error

(6) Differential linearity error

Ideally, the width to output a specific code is 1 LSB. This error indicates the difference between the actually measured value and its theoretical value when a specific code is output.

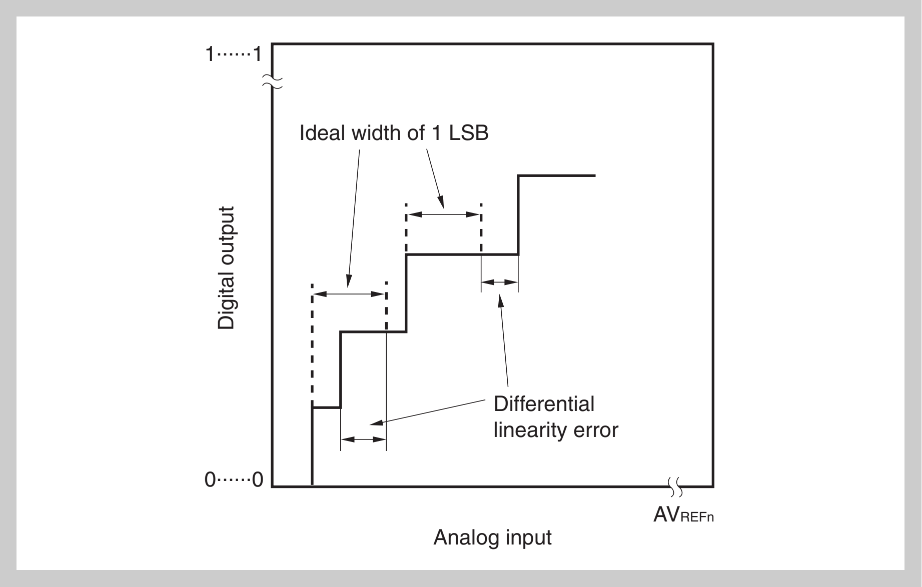


Figure 21-18 Differential linearity error

(7) Integral linearity error

This error indicates the extent to which the conversion characteristics differ from the ideal linear relationship. It indicates the maximum value of the difference between the actually measured value and its theoretical value where the zero-scale error and full-scale error are 0.

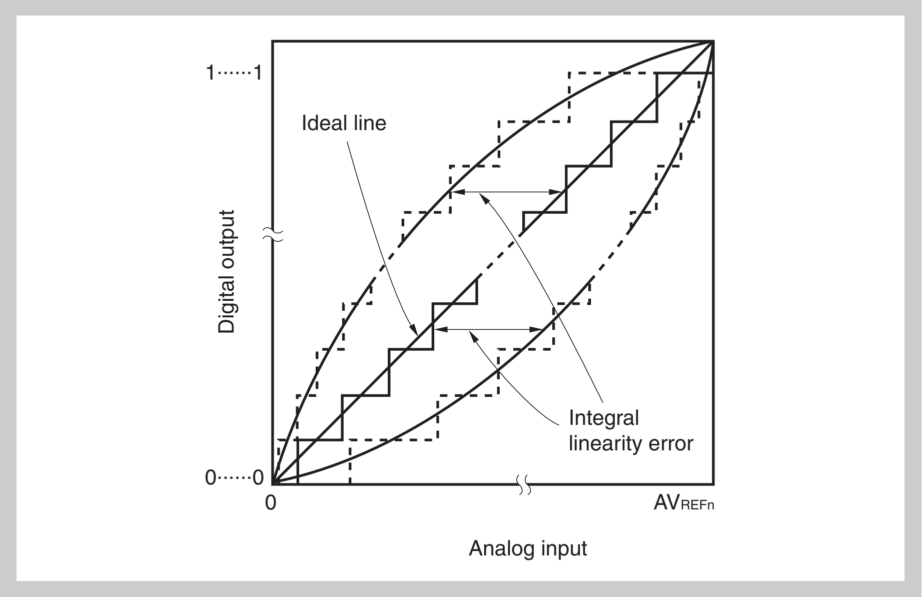


Figure 21-19 Integral linearity error

(8) Conversion time

This is the time required to obtain a digital output after an analog input voltage has been assigned.

The conversion time in the characteristics table includes the sampling time.

(9) Sampling time

This is the time for which the analog switch is ON to load an analog voltage to the sample & hold circuit.

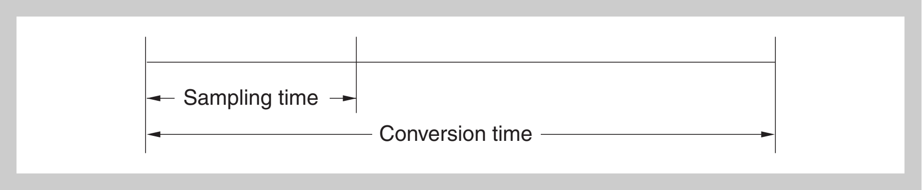


Figure 21-20 Sampling time

Chapter 22 Motor Control Function

Indices Following indices are used throughout this chapter:

Index	Range	Abbreviation	Meaning
n	0	TABn	PWM timer
x	4	TAAx	ADC synchronization timer
y	3	TAAy	Supply control timer
m	1 to 3	TOABnTm / TOABnBm	PWM channel outputs
z	0, 1	HZnCTLz	High-impedance control registers

22.1 Functional Overview

Timer ABn (TABn) and the TABn option (TABOPn) can be used as an inverter function that controls a motor. It performs a tuning operation with Timer AAx (TAAx) and A/D conversion can be started when the value of TABn matches the value of TAAx. The following operations can be performed as Motor Control Functions.

- 6-phase PWM output function with 16-bit accuracy (with dead-timer, for upper and lower arms)
- Timer tuning operation function (tunable with TAAx)
- Period setting function (period can be changed during operation of crest or valley interrupt)
- Compare register rewriting: Anytime rewrite, batch write, or intermittent rewrite (selectable during TABn operation)
- Interrupt and transfer culling functions
- Dead-time setting function
- A/D trigger timing function of A/D Converters 0 and 1 (four types of timing can be generated)
- 0% output and 100% output available
- 0% output and 100% output selectable by crest interrupt and valley interrupt
- Forced output stop function
- At valid edge detection by external pin input (INTP1/INTP3)

22.2 Configuration

The Motor Control Function consists of the following hardware.

Item	Configuration
Timer register	Dead-time counter
Compare register	TABn dead-time compare register (TABnDTC register)
Control registers	TABn option register 0 (TABnOPT0) TABn option register 1 (TABnOPT1) TABn option register 2 (TABnOPT2) TABn option register 3 (TABnOPT3) TABn I/O control register 3 (TABnIOC3) High-impedance output control registers 0, 1 (HZAnCTL0, HZAnCTL1)

- 6-phase PWM output can be produced with dead time by using the output of TABn (TOABn1, TOABn2, TOABn3)
- The output level of the 6-phase PWM output can be set individually.
- The 16-bit timer/counter of TABn counts up/down triangular waves. When the timer/counter underflows and when a period match occurs, an interrupt is generated. Interrupt generation, however, can be suppressed up to 31 times.
- TAAx can execute counting at the same time as TABn (timer tuning operation function). TAAx can be set in four ways as it can generate two types of A/D trigger sources (INTTAAxCC0 and INTTAAxCC1), and two types of interrupts: on underflow interrupt (INTTABnOV) and period match interrupt (INTTABnCC0).

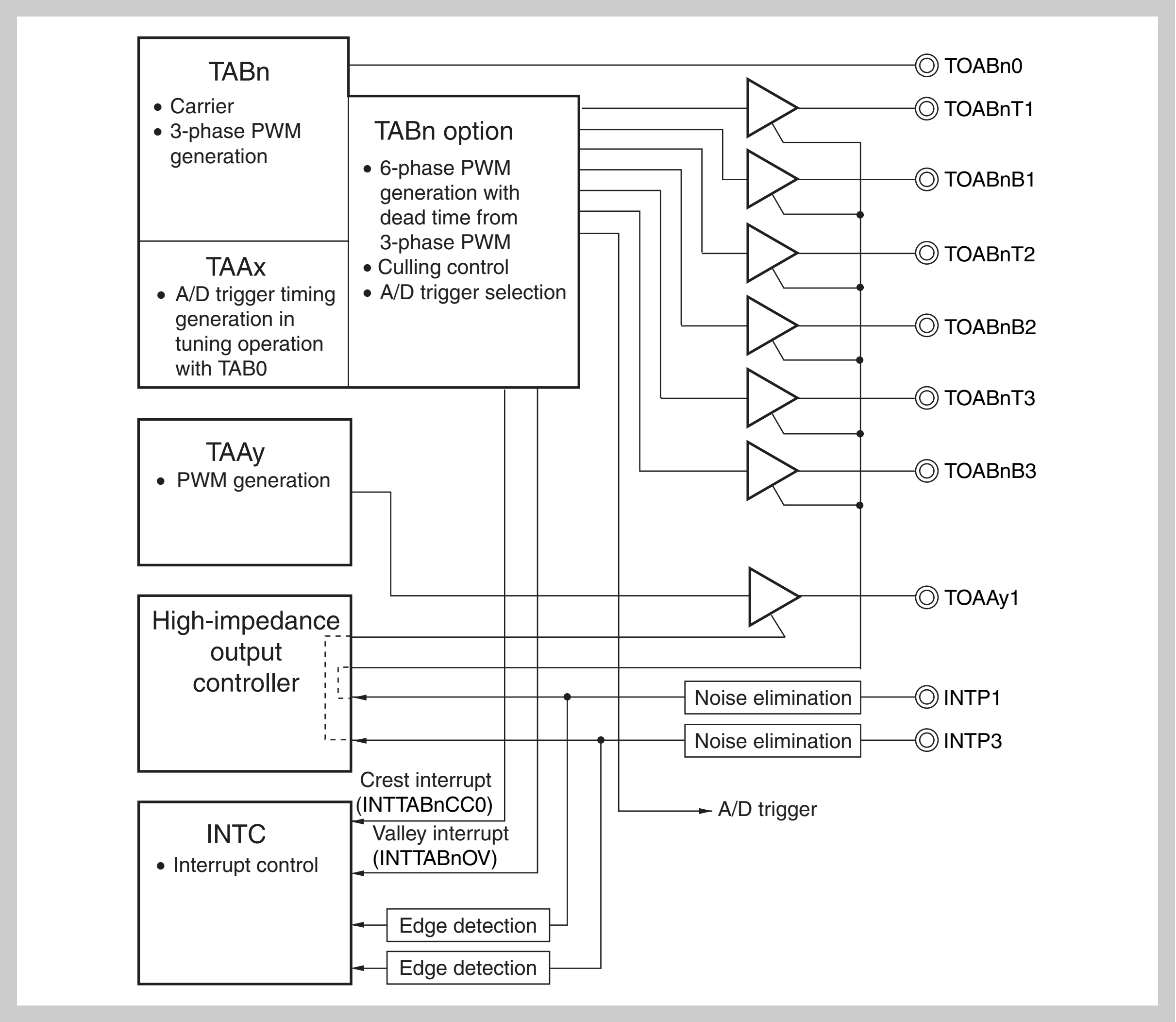
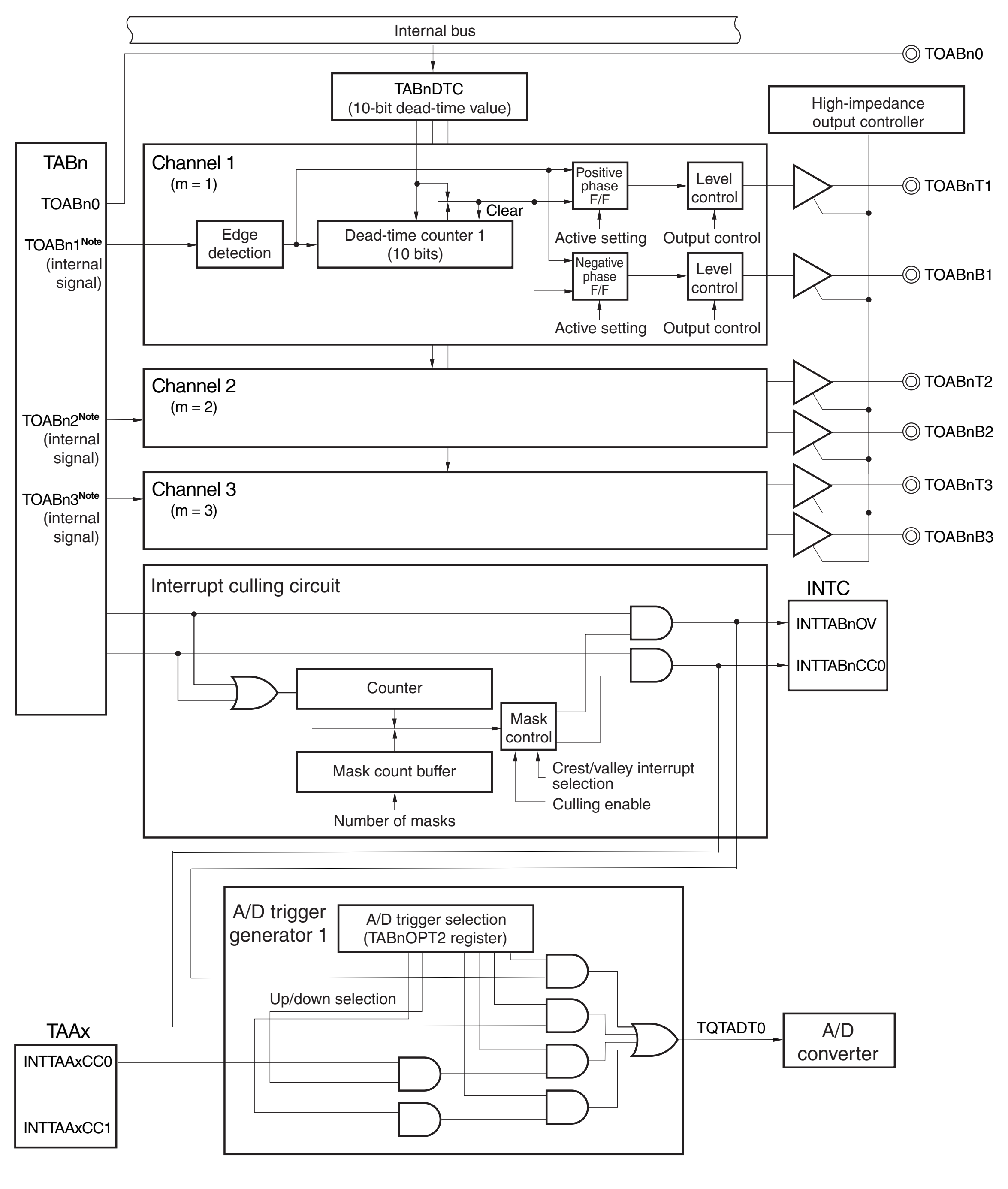


Figure 22-1 Block Diagram of Motor Control



Note: TOABn0 to TOABn3 function alternately as output pins.

Figure 22-2 TABn Option

(1) TABnDTC - TABn dead-time compare register

The TABnDTC register is a 10-bit compare register that specifies a dead-time value.

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

Address TAB0DTC: FFFFF564$_H$

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TABnDTC	0	0	0	0	0	0	TAB0DTC[9:0]									
	R/W															

Caution Rewriting of the TABnDTC register is prohibited when TABnCTL0.TABnCE = 1.

(2) Dead-time counters m (m = 1 to 3)

The dead-time counters are 10-bit counters that count dead time.
These counters are cleared or count up at the rising or falling edge of the TOABnm output signal by TABn, and are cleared or stopped when their count value matches the value of the TABnDTC register. The count clock of these counters is the same as that set by the TABnCTL0.TABnCKS2 to TABnCTL0.TABnCKS0 bits of TABn.

Note The operation differs when the TABnOPT2.TABnDTM bit = 1.
For details, see *"Automatic dead-time width narrowing function (TABnOPT2.TABnDTM bit = 1)" on page 883*.

22.3 Control Registers

(1) TABnOPT0 - TABn option register 0

The TABnOPT0 register is an 8-bit register that controls the Timer ABn option function.

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

Address TAB0OPT0: $\text{FFFFF545}_{\text{H}}$

Initial Value 00_{H}. This register is cleared by any reset.

	7	6	5	4	3	2	1	0
TABnOPT0	TABnCCS3	TABnCCS2	TABnCCS1	TABnCCS0	0	TABnCMS	TABnCUF	TABnOVF
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Caution

1. Rewrite TABnCCS[3:0] bits when TABnCE = 0 (the same value can be written when TABnCE = 1.). If rewriting was mistakenly performed, clear TABnCE to 0 and then set the bits again.
2. Be sure to clear bit 3 to 0, and bit 2 of registers TAB1OPT0 and TAB2OPT0 to 0 as well.
3. Be sure to clear the TABnCCS[3:0] bits to 0 in the 6-phase PWM output mode.

Table 22-1 TAAnOPT0 register contents

Bit position	Bit name	Function
7 to 4	TAAnCCSm (m = 0 to 3)	For details refer to *"TABnOPT0 - TAB option register 0" on page 485*
2	TAAnCMS	Specifies the compare register rewrite mode 0: Batch write mode (transfer operation) 1: Anytime write mode **Note:** **1.** The TABnCMS bit is valid only when the 6-phase PWM output mode is set (TABnCTL1.TABnMD[2:0] = 111_{B}). **2.** The TABnCMS bit can be rewritten while the timer is operating (TABnCTL0.TABnCE = 1). **3.** The following compare registers are rewritten in batch write mode: TABnCCR0 to TABnCCR3, TAAmCCR0, TAAmCCR1, and TABnOPT1.
1	TAAnCUF	Timer ABn up-/down-counting flag. 0: Timer ABn is counting up. 1: Timer ABn is counting down. **Note:** The TABnCUF bit is valid only when the 6-phase PWM output mode is set (TABnCTL1.TABnMD[2:0] = 111_{B}).
0	TAAnOVF	For details refer to *"TABnOPT0 - TAB option register 0" on page 485*

(2) TABnOPT1 - TABn option register 1

The TABnOPT1 register is an 8-bit register that controls the interrupt request signal generated by the Timer ABn option function.

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

Address TAB0OPT1: FFFFF560_H

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

	7	6	5	4	3	2	1	0
TABnOPT1	TABnICE	TABnIOE	0	TABnID4	TABnID3	TABnID2	TABnID1	TABnID0
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Note The TABnOPT1 register can be rewritten when TABnCTL0.TABnCE = 1.

Table 22-2 TAAnOPT0 register contents

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>7</td><td>TABnICE</td><td>Controls crest interrupt (INTTABnCC0) for interrupt culling circuit and ADC trigger.
0: Disable crest interrupt (INTTABnCC0).
1: Enable crest interrupt (INTTABnCC0).</td></tr>
<tr><td>6</td><td>TABnIOE</td><td>Controls valley interrupt (INTTABnOV) for interrupt culling circuit and ADC trigger.
0: Disable valley interrupt (INTTABnOV).
1: Enable valley interrupt (INTTABnOV).</td></tr>
<tr><td>4 to 0</td><td>TABnID[4:0]</td><td>Specifies the number of masked (culled) interupts.
<table>
<tr><th>TABn ID4</th><th>TABn ID3</th><th>TABn ID2</th><th>TABn ID1</th><th>TABn ID0</th><th>Number of culled interrupts</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>No interrupt culled (all interrupts are output)</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1 masked (one of two interrupts is output)</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>2 masked (onw of three interrupts is output)</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>3 masked (one of four interrupts is output)</td></tr>
<tr><td colspan="5">·
·
·</td><td>·
·
·</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>28 masked (one of 29 interrupts is output)</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>29 masked (one of 30 interrupts is output)</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>30 masked (one of 31 interrupts is output)</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>31 masked (one of 32 interrupts is output)</td></tr>
</table></td></tr>
</table>

(3) TABnOPT2 - TABn option register 2

The TABnOPT2 register is an 8-bit register that controls the Timer AB option function.

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

Address TAB0OPT2: FFFFF561$_H$

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

	7	6	5	4	3	2	1	0
TABnOPT2	TABnRDE	TABnDTM	TABnATM3	TABnATM2	TABnAT3	TABnAT2	TABnAT1	TABnAT0
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Caution The TABnOPT2 register is prohibited from writing during timer operation (TABnCTL0.TABnCE = 1). However, the same value can be written when TABnCE = 1.

Table 22-3 TABnOPT2 register contents (1/2)

Bit position	Bit name	Function
7	TABnRDE	Controls the batch write mode (transfer operation) culling. 0: Do not cull transfer (transfer timing is generated every time at crest and valley). 1: Cull transfer at the same interval as interrupt culling set by the TABnOPT1 register. **Caution:** When using interrupt culling (TABnOPT1.TABnID[4:0] bits are set other than 00000$_B$), be sure to set TABnRDE = 1. Thus, the interrupt and transfer occur at the same timing. Setting separate occurrence of interrupt and transfer is prohiobited. If the interrupt and transfer are set separately (TABnRDE = 0), transfer is not performed normally.
6	TABnDTM	Controls dead-time counter operation mode. 0: Dead-time counter counts up normally and, if TOBnm output of TABn is at a narrow interval (TOBnm output width < dead-time width), the dead-time counter is cleared and counts up again. 1: Dead-time counter counts up normally and, if TOBnm output of TABn is at a narrow interval (TOBnm output width < dead-time width), the dead-time counter counts down and the dead-time control width is automatically narrowed. **Caution:** Rewriting of the TABnDTM bit is disabled during timer operation. If rewriting was mistakenly performed, stop the timer operation by clearing the TABnCTL0.TABnCE bit to 0 and then write the bit again.
5	TABnATM3	Specifies mode of A/D trigger control 3. 0: Output A/D trigger signal (TQTADT0) for INTTAAxCC1 interrupt while 16-bit counter is counting up. 1: Output A/D trigger signal (TQTADT0) for INTTAAxCC1 interrupt while 16-bit counter is counting down.
4	TABnATM2	Specifies mode of A/D trigger control 2. 0: Output A/D trigger signal (TQTADT0) for INTTAAxCC0 interrupt while 16-bit counter is counting up. 1: Output A/D trigger signal (TQTADT0) for INTTAAxCC0 interrupt while 16-bit counter is counting down.

Table 22-3 TABnOPT2 register contents (2/2)

Bit position	Bit name	Function
3	TABnAT3	A/D trigger control 3 0: Disable output A/D trigger signal (TQTADT0) for INTTAAxCC1. 1: Enable output A/D trigger signal (TQTADT0) for INTTAAxCC1.
2	TABnAT2	A/D trigger control 2 0: Disable output A/D trigger signal (TQTADT0) for INTTAAxCC0. 1: Enable output A/D trigger signal (TQTADT0) for INTTAAxCC0.
1	TABnAT1	A/D trigger control 1 0: Disable output A/D trigger signal (TQTADT0) for INTTABnCC0 (crest interrupt). 1: Enable output A/D trigger signal (TQTADT0) for INTTABnCC0 (crest interrupt).
0	TABnAT1	A/D trigger control 0 0: Disable output A/D trigger signal (TQTADT0) for INTTABnOV (valleyinterrupt). 1: Enable output A/D trigger signal (TQTADT0) for INTTABnOV (valleyinterrupt).

Note For further information on the A/D trigger signal refer to *“A/D Converter (ADC)” on page 817*.

(4) TABnOPT3 - TABn option register 3

The TABnOPT3 register is an 8-bit register that controls the Timer ABn option function.

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

Address TAB0OPT3: FFFFF563_H

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

	7	6	5	4	3	2	1	0
TABnOPT3	0	0	TAB0ATM7	TAB0ATM6	TAB0AT7	TAB0AT6	TAB0AT5	TAB0AT4
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Note

1. The register is available on V850ES/FK3 only.
2. The TABnOPT2 register can be rewritten when TABnCTL0.TABnCE = 1.

Table 22-4 TABnOPT3 register contents

Bit position	Bit name	Function
5	TABnATM7	Specifies mode of A/D trigger control 7. 0: Output A/D trigger signal (TABTADT1) of INTTAAxCC1 interrupt while 16-bit counter is counting up. 1: Output A/D trigger signal (TABTADT1) of INTTAAxCC1 interrupt while 16-bit counter is counting down.
4	TABnATM6	Specifies mode of A/D trigger control 6. 0: Output A/D trigger signal (TABTADT1) of INTTAAxCC0 interrupt while 16-bit counter is counting up. 1: Output A/D trigger signal (TABTADT1) of INTTAAxCC0 interrupt while 16-bit counter is counting down.
3	TABnAT7	A/D trigger control 3 0: Disable output A/D trigger signal (TABTADT1)) for INTTAAxCC1. 1: Enable output A/D trigger signal (TABTADT1) for INTTAAxCC1.
2	TABnAT6	A/D trigger control 2 0: Disable output A/D trigger signal (TABTADT1) for INTTAAxCC0. 1: Enable output A/D trigger signal (TABTADT1) for INTTAAxCC0.
1	TABnAT5	A/D trigger control 1 0: Disable output A/D trigger signal (TABTADT1)) for INTTABnCC0 (crest interrupt). 1: Enable output A/D trigger signal (TABTADT1) for INTTABnCC0 (crest interrupt).
0	TABnAT4	A/D trigger control 0 0: Disable output A/D trigger signal (TABTADT1) for INTTABnOV (valleyinterrupt). 1: Enable output A/D trigger signal (TABTADT1) for INTTABnOV (valleyinterrupt).

(5) TABnIOC3 - TABn I/O control register 3

The TABnIOC3 register is an 8-bit register that controls the output of the Timer ABn option function.
To output from the TOABnTm pin, set the TABnIOC0.TABnOEm bit to 1 and then set the TABnIOC3 register.
The TABnIOC3 register can be rewritten only when the TABnCTL0.TABnCE bit is 0.
Rewriting each bit of the TABnIOC3 register is prohibited when the TABnCTL0.TABnCE bit is 1; however the same value can be rewritten to each bit of the TABnIOC3 register when the TABnCTL0.TABnCE bit is 1.
This register can be read or written in 8-bit or 1-bit units.
Reset input sets this register to A8H.

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

Address TAB0IOC3: FFFFF562$_H$

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

	7	6	5	4	3	2	1	0
TABnIOC3	TABnOLB3	TABnOEB3	TABnOLB2	TABnOEB2	TABnOLB1	TABnOEB1	0	0
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Caution Set the TABnIOC3 register to the default value (A8$_H$) when the timer is used in a mode other than the 6-phase PWM output mode (TABnCTL1.TABnMD[2:0] = 111$_B$).

Note Set the output level of the TOABnTm pin by the TABnIOC0 register.

Table 22-5 TABnIOC3 register contents

Bit position	Bit name	Function
7, 5, 3	TABnOLBm (m = 7, 5, 3)	Specifies the TOABnBm pin output level. 0: Disable inversion of TOABnBm pin output. 1: Enable inversion of TOABnBm pin output.
6, 4, 2	TABnOEBm (m = 6, 4, 2)	Controls the TOABnBm pin output. 0: Disable TOABnBm pin output. - When TABnOLBm bit = 0, low level is output from TOABnBm pin. - When TABnOLBm bit = 1, high level is output from TOABnBm pin. 1: Enable TOABnBm pin output.

(a) Output from TOABnTm and TOABnBm pins

The TOABnTm pin output is controlled by the TABnIOC0.TABnOLm and TABnIOC0.TABnOEm bits.
The TOABnBm pin output is controlled by the TABnIOC3.TABnOLBm and TABnIOC3.TABnOEBm bits.
A timer output with each setting in the 6-phase PWM output mode is shown below.

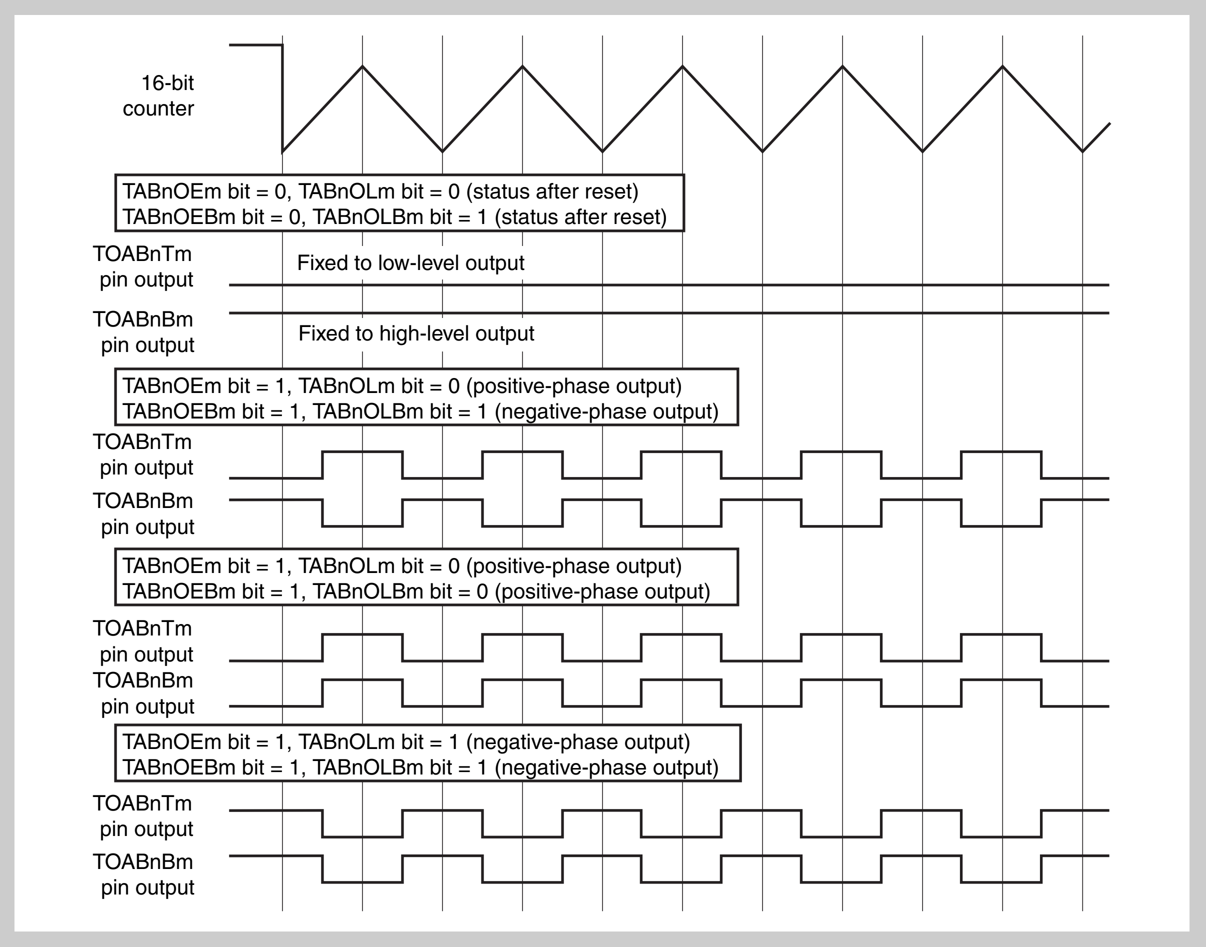

16-bit counter
TABnOEm bit = 0, TABnOLm bit = 0 (status after reset)
TABnOEBm bit = 0, TABnOLBm bit = 1 (status after reset)
TOABnTm pin output
Fixed to low-level output
TOABnBm pin output
Fixed to high-level output
TABnOEm bit = 1, TABnOLm bit = 0 (positive-phase output)
TABnOEBm bit = 1, TABnOLBm bit = 1 (negative-phase output)
TOABnTm pin output
TOABnBm pin output
TABnOEm bit = 1, TABnOLm bit = 0 (positive-phase output)
TABnOEBm bit = 1, TABnOLBm bit = 0 (positive-phase output)
TOABnTm pin output
TOABnBm pin output
TABnOEm bit = 1, TABnOLm bit = 1 (negative-phase output)
TABnOEBm bit = 1, TABnOLBm bit = 1 (negative-phase output)
TOABnTm pin output
TOABnBm pin output

Figure 22-3 TOABnTm and TOABnBm Pin Output Control

Table 22-6 TOABnTm Pin Output

TABnOLm Bit	TABnOEm Bit	TABnCE Bit	TOABnTm Pin Output
0	0	x	Low-level output
	1	0	Low-level output
		1	TOABnTm positive-phase output
1	0	x	High-level output
	1	0	High-level output
		1	TOABnTm negative-phase output

Table 22-7 TOABnBm Pin Output

TABnOLBm Bit	TABnOEBm Bit	TABnCE Bit	TOABnBm Pin Output
0	0	x	Low-level output
	1	0	Low-level output
		1	TOABnBm positive-phase output
1	0	x	High-level output
	1	0	High-level output
		1	TOABnBm negative-phase output

(6) HZAnCTL0, HZAnCTL1 - High-impedance output control registers

The HZAnCTL0 and HZAnCTL1 registers are 8-bit registers that control the high-impedance state of the output buffer.

Access This register can be read/written in 8-bit or 1-bit units. However, the HZAnDCFz bit is a read-only bit and cannot be written.

Address HZA0CTL0: FFFFF570$_H$
HZA0CTL1: FFFFF571$_H$

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

	7	6	5	4	3	2	1	0
HZAnCTLz	HZAnDCEz	HZAnDCMz	HZAnDCNz	HZAnDCPz	HZAnDCTz	HZAnDCCz	0	HZAnDCFz
	R/W	R/W	R/W	R/W	R/W	R/W	R	R

Note The same value can be always rewritten to the HZAnCTLz register by software.
The relationship between detection factor and the control registers is shown below.

Pins Subject to High-Impedance Control	High-Impedance Control Factor	Control Register
	External Pin	
When TOABnT1 to TOABnT3 are output When TOABnB1 to TOABnB3 is output	INTP1	HZAnCTL0
When TOAAy1 is output	INTP3	HZAnCTL1

Caution High-impedance control is performed only when a port pin is set to function as indicated in the above table.

Table 22-8 HZAnCTLz register contents (1/3)

Bit position	Bit name	Function
7	HZAnDCEz	Controls high-impedance output. 0: Disable high-impedance output control operation. Pins can funciton as output pins. 1: Enable high-impedance output control operation.
6	HZAnDCMz	Specifies condition of clearing high-impedance state by HZAnDCCz bit. 0: Setting of the HZAnDCCz bit is valid regardless of the external pin input[a]. 1: Setting of the HZAnDCCz bit is invalid while the external pin[a] holds a level detected as abnormal (active level). **Note:** Rewrite the HZAnDCMz bit when HZAnDCEz bit = 0.

Table 22-8 HZAnCTLz register contents (2/3)

<table>
<tr><th>Bit position</th><th>Bit name</th><th>Function</th></tr>
<tr><td>5, 4</td><td>HZAnDCNz
HZAnDCPz</td><td>Specifies the valid edge of external pin input[ab].
<table>
<tr><th>HZAnDCNz</th><th>HZAnDCPz</th><th>External pin input[ab] valid edge</th></tr>
<tr><td>0</td><td>0</td><td>No valid edge detection (setting the HZAnDCFz bit by external pin input is prohibited).</td></tr>
<tr><td>0</td><td>1</td><td>Rising edge detection (abnormality is detected by rising edge input).</td></tr>
<tr><td>1</td><td>0</td><td>Falling edge detection (abnormality is detected by falling edge input).</td></tr>
<tr><td>1</td><td>1</td><td>Setting prohibited</td></tr>
</table>
Note: **1.** Rewrite the HZAnDCNz and HZAnDCPz bits when the HZAnDCEz = 0.
2. High-impedance output control is performed when the valid edge is input after the operation is enabled (by setting HZAnDCEz bit to 1). If the external pin is at the active level when the operation is enabled, the high-impedance output control is not performed.</td></tr>
<tr><td>3</td><td>HZAnDCTz</td><td>High impedance output trigger
0: No operation
1: Corresponding output pins are set into high-impedance state by software and the HZAnDCFz bit is set to 1.
Caution: **1.** The HZAnDCTz bit is always 0 when read.
2. When HZAnDCEz = 0, the HZAnDCTz bit is invalid even if is set to 1.
3. Simultaneously setting of the HZAnDCTz and HZAnDCCz bits to 1 is prohibited.
Note: If a valid edged is input to the external pin[a] (according to the setting of the HZAnDCNz and HZAnDCPz bits), the HZAnDCTz bit is invalid even it is set to 1.</td></tr>
<tr><td>2</td><td>HZAnDCCz</td><td>High impedance output control clear bit
0: No operation
1: Output pins which are set into high-impedance state are enabled by software and the HZAnDCFz bit is cleared to 0.
Caution: **1.** The HZAnDCCz bit is always 0 when read.
2. When HZAnDCEz = 0, the HZAnDCCz bit is invalid even if is set to 1.
3. Simultaneously setting of the HZAnDCTz and HZAnDCCz bits to 1 is prohibited.
Note: If a valid edged is input to the external pin[a] (according to the setting of the HZAnDCNz and HZAnDCPz bits), the HZAnDCCz bit is invalid even it is set to 1.</td></tr>
</table>

Table 22-8 HZAnCTLz register contents (3/3)

Bit position	Bit name	Function
0	HZAnDCFz	High impedance outputstatus flag 0: Indicates that output pins are enabled. 1: Indicates that the output pins are in high-impedance state. **Note:** **1.** The HZAnDCFz bit is cleared to 0, • when HZAnDCEz bit = 0. • when HZAnDCC bit = 1. **2.** The HZAnDCFz bit is set to 1, • when HZAnDCTz bit = 1. • when an valid edge indicating abnormality is input to the external pin[a] (according to the setting of the HZAnDCNz and HZAnDCPz bits)

[a] HZAnCTL0: INTP1 pin, HZAnCTL1: INTP3 pin
For further details refer to *Figure 22-4* below.

[b] The edge of the INTP1 and INTP3 input pins must be set before the valid edge of the external pins is specified. For details of the edge specification of INTP1 and INTP3 input pins, refer to *“External Interrupts Edge Detection Configuration” on page 280*.

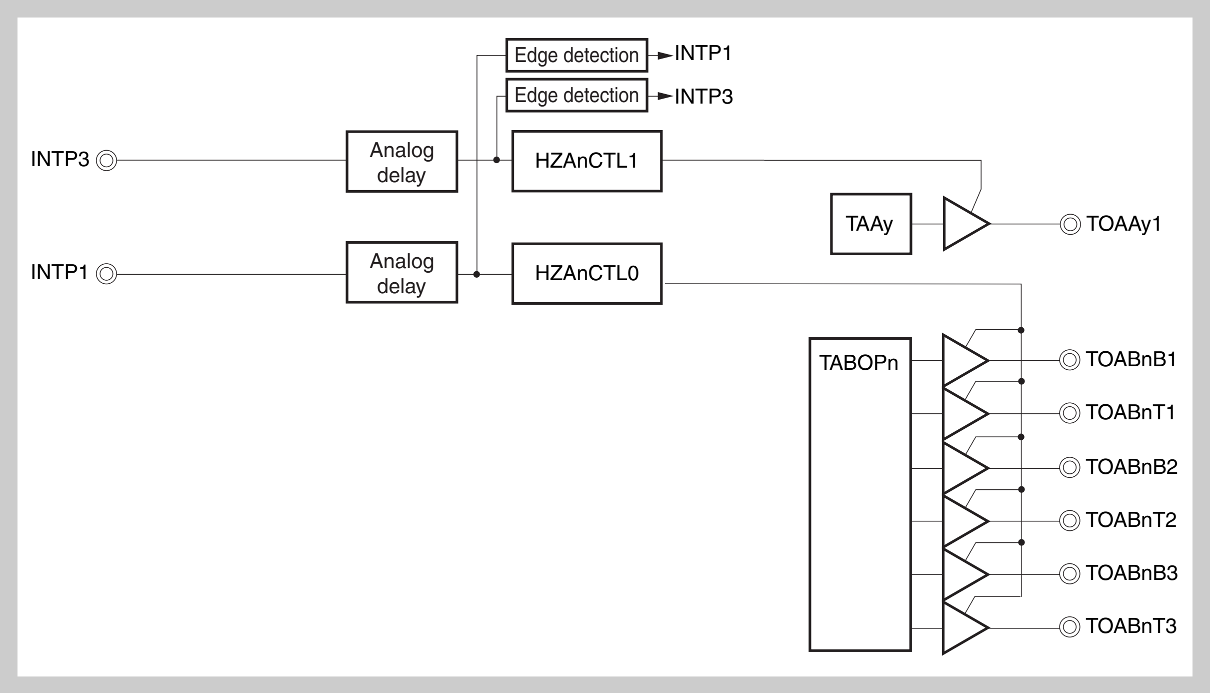


Figure 22-4 High-Impedance Output Controller Configuration

(a) Setting procedure

1. Setting of high-impedance control operation
   - Set the HZAnDCMz, HZAnDCNz, and HZAnDCPz bits.
   - Set the HZAnDCEz bit to 1 (enable high-impedance control).
2. Changing setting after enabling high-impedance control operation
   - Clear the HZAnDCEz bit to 0 (to stop the high-impedance control operation).
   - Change the setting of the HZAnDCMz, HZAnDCNz, and HZAnDCPz bits.
   - Set the HZAnDCEz bit to 1 (to enable the high-impedance control operation again).
3. Resuming output when pins are in high-impedance state

   If the HZAnDCMz bit is 1, set the HZAnDCCz bit to 1 to clear the high-impedance state after the valid edge of the external pin[Note] is detected. However, the high-impedance state cannot be cleared unless this bit is set while the input level of the external pin[Note] is inactive.
   - Set the HZAnDCCz bit to 1 (command signal to clear the high-impedance state).
   - Read the HZAnDCFz bit and check the flag status.
   - Return to <1> if the HZAnDCFz bit is 1. The input level of the external pin[Note] must be checked.
     The pin can function as an output pin if the HZAnDCFz bit is 0.
4. To make the pin to go into a high-impedance state by software

   The HZAnDCTz bit must be set to 1 by software to make the pin to go into a high-impedance state while the input level of the external pin[Note] is inactive. The following procedure is an example in which the setting is not dependent upon the setting of the HZAnDCMz bit.
   - Set the HZAnDCTz bit to 1 (high-impedance output command).
   - Read the HZAnDCFz bit to check the flag status.
   - Return to <1> if the HZAnDCFz bit is 0. The input level of the external pin[Note] must be checked.
     The pin is in a high-impedance state if the HZAnDCFz bit is 1.

   However, if the external pin[Note] is not used with the HZAnDCPz bit and HZAnDCNz bit cleared to 0, the pin goes into a high-impedance state when the HZAnDCTz bit is set to 1.

Note HZAnCTL0: INTP1 pin, HZAnCTL1: INTP3 pin

22.4 Operation

22.4.1 System outline

(1) Outline of 6-phase PWM output

The 6-phase PWM output mode is used to generate a 6-phase PWM output wave, by using TABn and the TABn option in combination.
The 6-phase PWM output mode is enabled by setting the TABnCTL1.TABnMD2 to TABnCTL1.TABnMD0 bits of TABn to “111”.
One 16-bit counter and four 16-bit compare registers of TABn are used to generate a basic 3-phase wave.
The functions of the compare registers are as follows.
TAAx can perform a tuning operation with TABn to start a conversion trigger source for the A/D Converter.

Compare Register	Function	Settable Range
TABnCCR0 register	Setting of cycle	0002H ≤ M ≤ FFFEH
TABnCCR1 register	Specifying output width of phase U	0000H ≤ i ≤ M + 1
TABnCCR2 register	Specifying output width of phase V	0000H ≤ j ≤ M + 1
TABnCCR3 register	Specifying output width of phase W	0000H ≤ k ≤ M + 1

Note M = Set value of TABnCCR0 register
i = Set value of TABnCCR1 register
j = Set value of TABnCCR2 register
k = Set value of TABnCCR3 register

A dead-time interval is generated from the basic 3-phase wave generated by using three 10-bit dead-time counters and one compare register to create a wave with a reverse phase to that of the basic 3-phase wave. Then a 6-phase PWM output wave (U, $\overline{U}$, V, $\overline{V}$, W, and $\overline{W}$) is generated.

The 16-bit counter for generating the basic 3-phase wave counts up or down. After the operation has been started, this counter counts up. When its count value matches the cycle set to the TABnCCR0 register, the counter starts counting down. When the count value matches 0001H, the counter counts up again. This means that a value two times higher than the value set to the TABnCCR0 register +1 is the carrier cycle.

10-bit dead-time counters 1 to 3 that generate the dead-time interval count up. Therefore, the value set to the TABn dead-time compare register (TABnDTC) is used as a dead-time value as is. Because three counters are used, dead time can be generated independently in phases U, V, and W. However, because there is only one register that specifies a dead-time value (TABnDTC), the same dead-time value is used in the three phases.

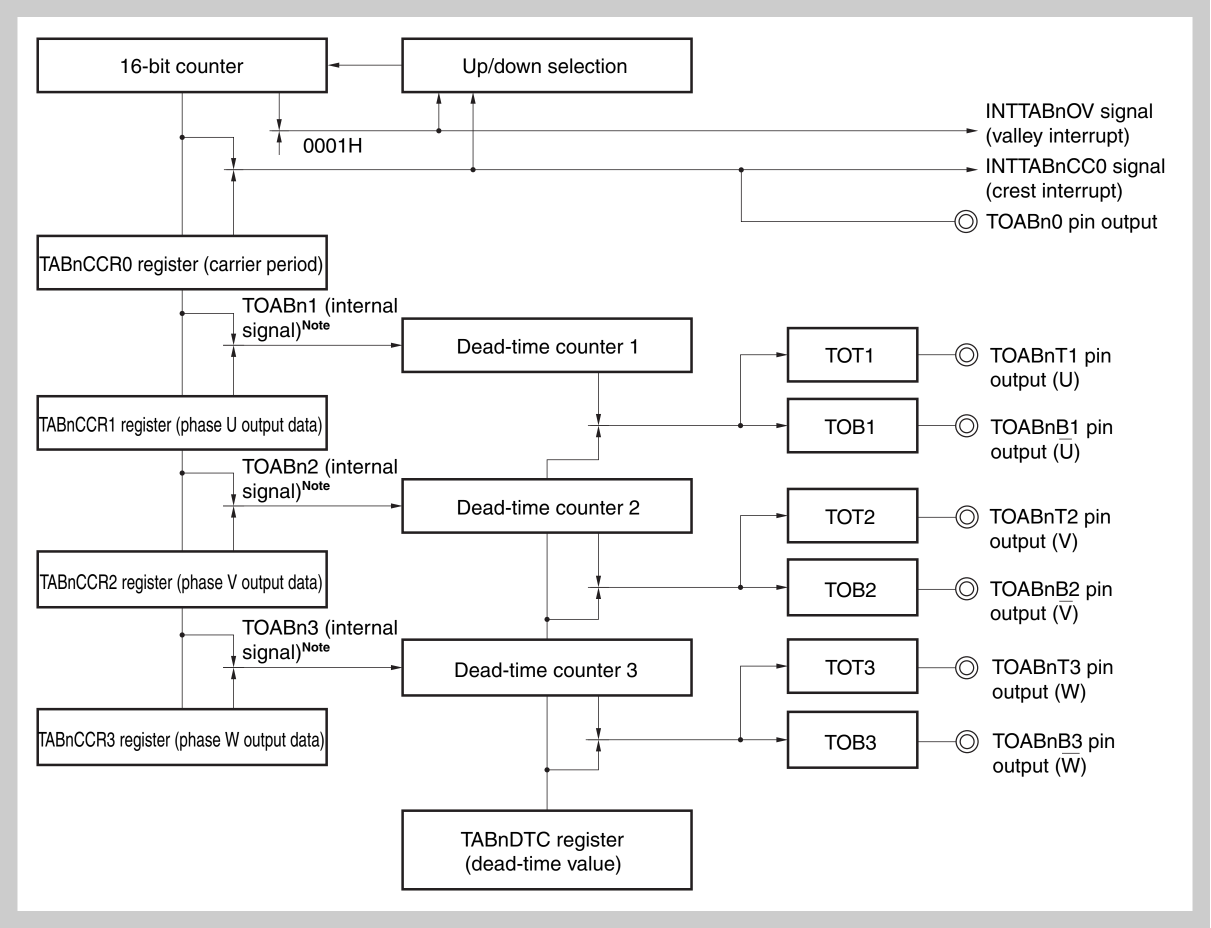

16-bit counter
Up/down selection
0001H
INTTABnOV signal (valley interrupt)
INTTABnCC0 signal (crest interrupt)
TOABn0 pin output
TABnCCR0 register (carrier period)
TOABn1 (internal signal)Note
Dead-time counter 1
TOT1
TOB1
TOABnT1 pin output (U)
TOABnB1 pin output (U)
TABnCCR1 register (phase U output data)
TOABn2 (internal signal)Note
Dead-time counter 2
TOT2
TOB2
TOABnT2 pin output (V)
TOABnB2 pin output (V)
TABnCCR2 register (phase V output data)
TOABn3 (internal signal)Note
Dead-time counter 3
TOT3
TOB3
TOABnT3 pin output (W)
TOABnB3 pin output (W)
TABnCCR3 register (phase W output data)
TABnDTC register (dead-time value)

Figure 22-5 Outline of 6-Phase PWM Output Mode

Note TOABn1, TOABn2, and TOABn3 function alternately as output pins.

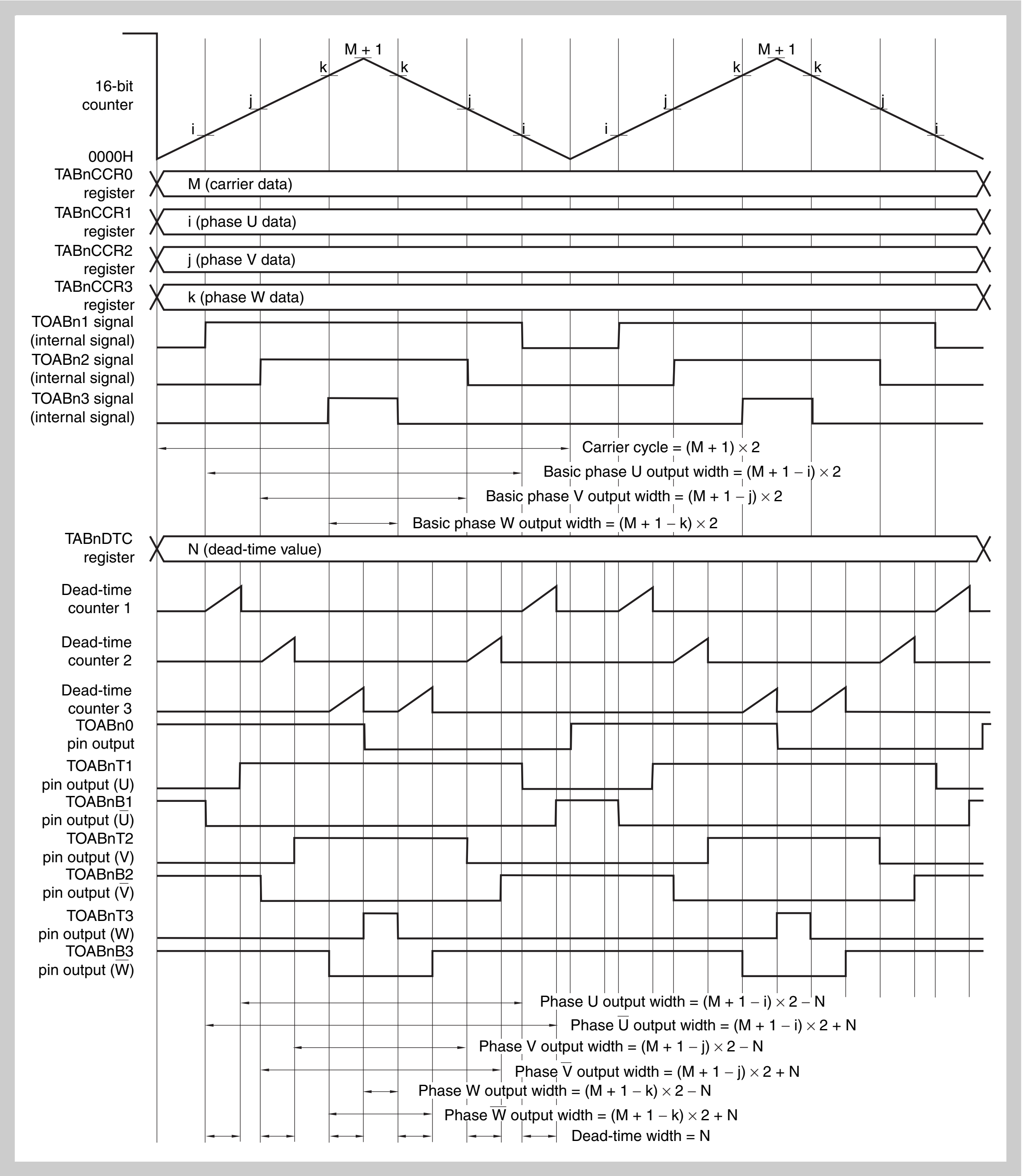

16-bit counter
0000H
M + 1
k
j
i
TABnCCR0 register
M (carrier data)
TABnCCR1 register
i (phase U data)
TABnCCR2 register
j (phase V data)
TABnCCR3 register
k (phase W data)
TOABn1 signal (internal signal)
TOABn2 signal (internal signal)
TOABn3 signal (internal signal)
Carrier cycle = (M + 1) × 2
Basic phase U output width = (M + 1 − i) × 2
Basic phase V output width = (M + 1 − j) × 2
Basic phase W output width = (M + 1 − k) × 2
TABnDTC register
N (dead-time value)
Dead-time counter 1
Dead-time counter 2
Dead-time counter 3
TOABn0 pin output
TOABnT1 pin output (U)
TOABnB1 pin output (U̅)
TOABnT2 pin output (V)
TOABnB2 pin output (V̅)
TOABnT3 pin output (W)
TOABnB3 pin output (W̅)
Phase U output width = (M + 1 − i) × 2 − N
Phase U̅ output width = (M + 1 − i) × 2 + N
Phase V output width = (M + 1 − j) × 2 − N
Phase V̅ output width = (M + 1 − j) × 2 + N
Phase W output width = (M + 1 − k) × 2 − N
Phase W̅ output width = (M + 1 − k) × 2 + N
Dead-time width = N

Figure 22-6 Timing Chart of 6-Phase PWM Output Mode

Caution

1. Set the value “M” of the TABnCCR0 register in a range of 0002H ≤ M ≤ FFFEH in the 6-phase PWM output mode.
2. Only a value of up to “M + 1” can be set to the TABnCCR1, TABnCCR2, and TABnCCR3 registers.
   If "M + 2" and more is set, the output signal rises at the 16-bit counter crest (M + 1) and falls at the valley (0000H) (50% duty output).
3. The output is 100% if “0000H” is set to the TABnCCR1, TABnCCR2, and TABnCCR3 registers. The output is 0% if “M + 1” is set to the TABnCCR1, TABnCCR2, and TABnCCR3 registers.
4. If the operation value of the output width of phases U, V, and W (such as (M + 1 – i) × 2 – N) is 0 or lower, it is converged to 0 (100% output). If the operation value is higher than “M + 1”, it is converged to M + 1 (0% output).

(2) Interrupt requests

Two types of interrupt requests are available: the INTTABnCC0 (crest interrupt) signal and INTTABnOV (valley interrupt) signal.
The INTTABnCC0 and INTTABnOV signals can be culled by using the TABnOPT1 register.
For details of culling interrupts, see *“Interrupt culling function” on page 885*.

- INTTABnCC0 (crest interrupt) signal:
  Interrupt signal indicating matching between the value of the 16-bit counter that counts up and the value of the TABnCCR0 register
- INTTABnOV (valley interrupt) signal:
  Interrupt signal indicating matching between the value of the 16-bit counter that counts down and the value 0001H

(3) Rewriting registers during timer operation

The following registers have a buffer register and can be rewritten in the anytime rewrite mode, batch rewrite mode, or intermittent batch rewrite mode.

Related Unit	Register
Timer AAm	TAAm capture/compare register 0 (TAAxCCR0) TAAm capture/compare register 1 (TAAxCCR1)
Timer ABn	TABn capture/compare register 0 (TABnCCR0) TABn capture/compare register 1 (TABnCCR1) TABn capture/compare register 2 (TABnCCR2) TABn capture/compare register 3 (TABnCCR3)
Timer ABn option	TABn option register 1 (TABnOPT1)

For details of the transfer function of the compare register, see *“Operation to rewrite register with transfer function” on page 892*.

(4) Counting-up/down operation of 16-bit counter

The operation status of the 16-bit counter can be checked by using the TABnCUF bit of TABn option register 0 (TABnOPT0).

Status of TABnCUF Bit	Status of 16-Bit Counter	Range of 16-Bit Counter Value
TABnCUF bit = 0	Counting up	0000H – M
TABnCUF bit = 1	Counting down	(m+1) – 0001H

Note M = Set value of TABnCCR0 register

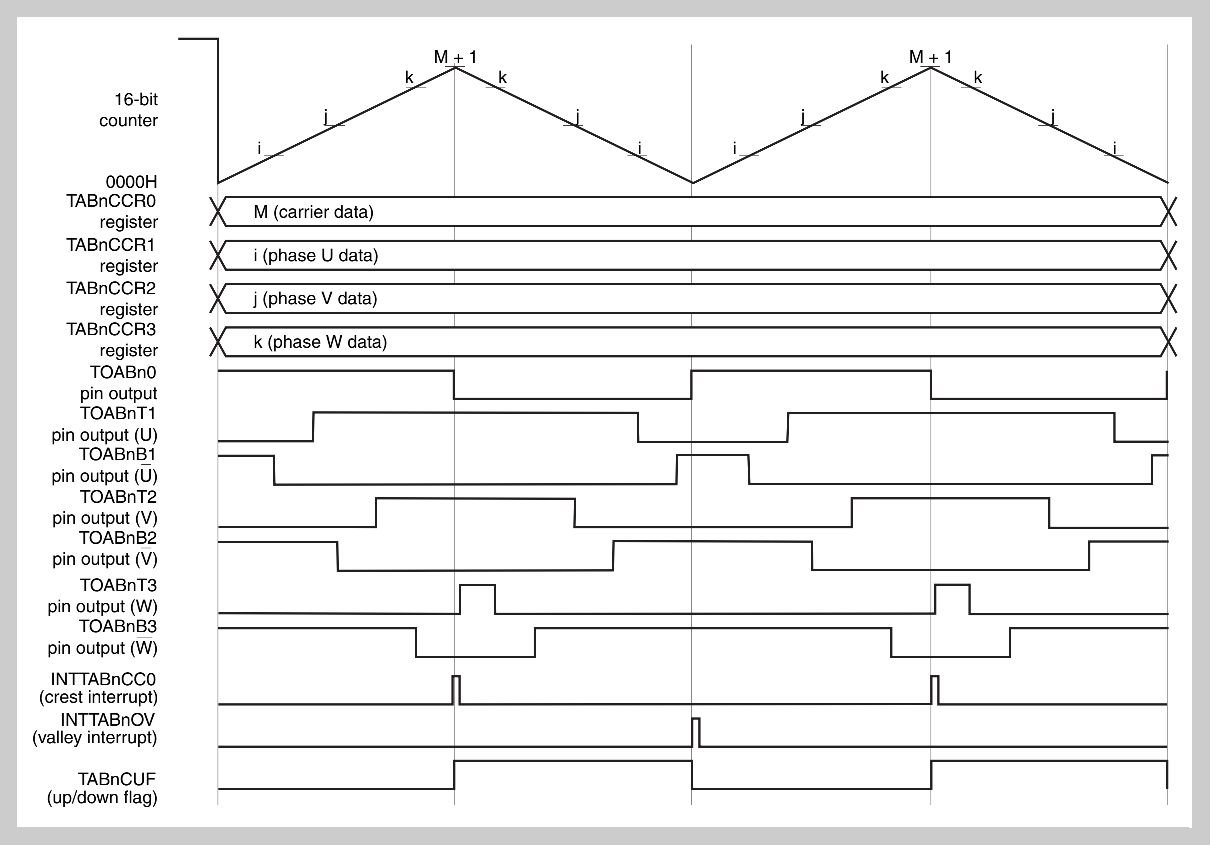


Figure 22-7 Interrupt and Up/Down Flag

22.4.2 Dead-time control (generation of negative-phase wave signal)

(1) Dead-time control mechanism

In the 6-phase PWM output mode, compare registers 1 to 3 (TABnCCR1, TABnCCR2, and TABnCCR3) are used to set the duty factor, and compare register 0 (TABnCCR0) is used to set the cycle. By setting these four registers and by starting the operation of TAB, three types of PWM output waves (basic 3-phase waves) with a variable duty factor are generated. These three PWM output waves are input to the Timer AB option unit (TABOPn) and their inverted signal with dead-time is created to generate three sets of (six) PWM waves.

The TABOPn unit consists of three 10-bit counters (dead-time counters 1 to 3) that operate in synchronization with the count clock of TABn, and a TABn dead-time compare register (TABnDTC) that specifies dead time. If “a” is set to the TABnDTC register, the dead-time value is “a”, and interval “a” is created between a positive-phase wave and a negative-phase wave.

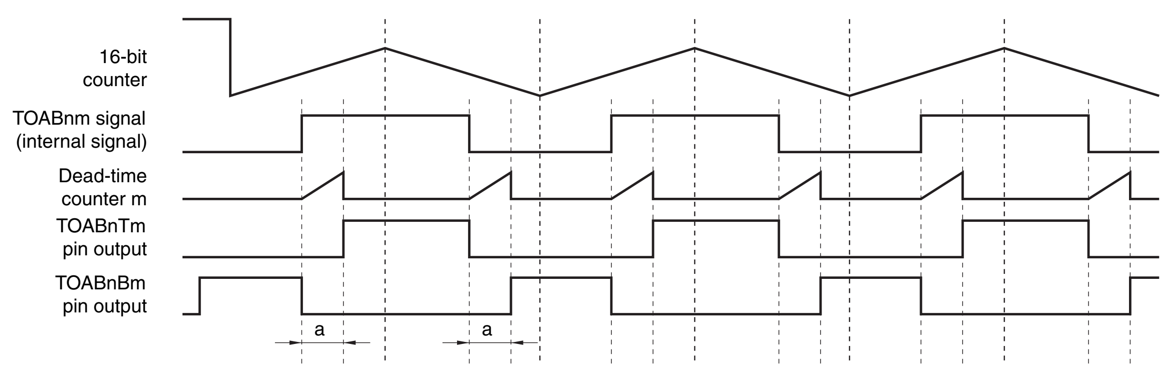


Figure 22-8 PWM Output Wave with Dead Time (1)
(When dead time is inserted (TABnDTC register = a))

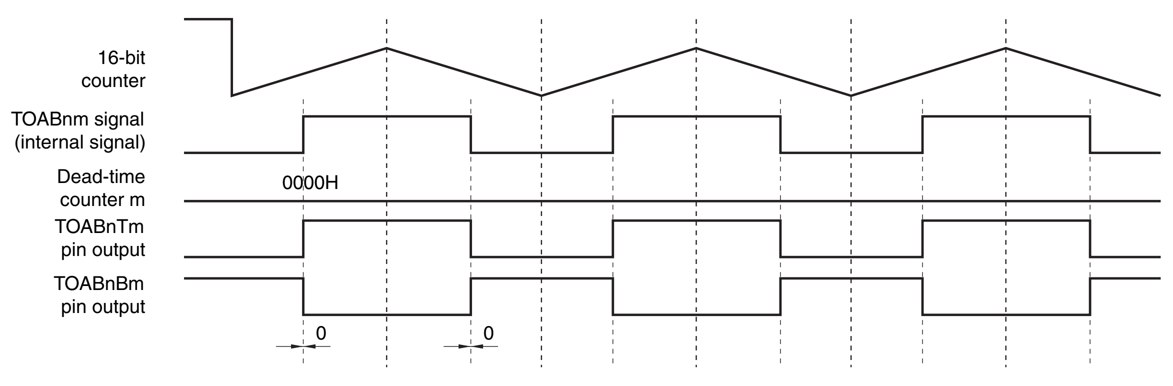


Figure 22-9 PWM Output Wave with Dead Time (1)
(No dead time (TABnDTC register = 000H)

(2) PWM output of 0%/100%

The microcontroller is capable of 0% wave output and 100% wave output for PWM output.

A low level is continuously output from TOABnTm pin as the 0% wave output. A high level is continuously output from TOABnTm pin as the 100% wave output. The 0% wave is output by setting the TABnCCRm register to “M + 1” when the TABnCCR0 register = M.
The 100% wave is output by setting the TABnCCRm register to “0000H”.
Rewriting the TABnCCRm register is enabled while the timer is operating, and 0% wave output or 100% wave output can be selected at the point of the crest interrupt (INTTABnCC0) and valley interrupt (INTTABnOV).

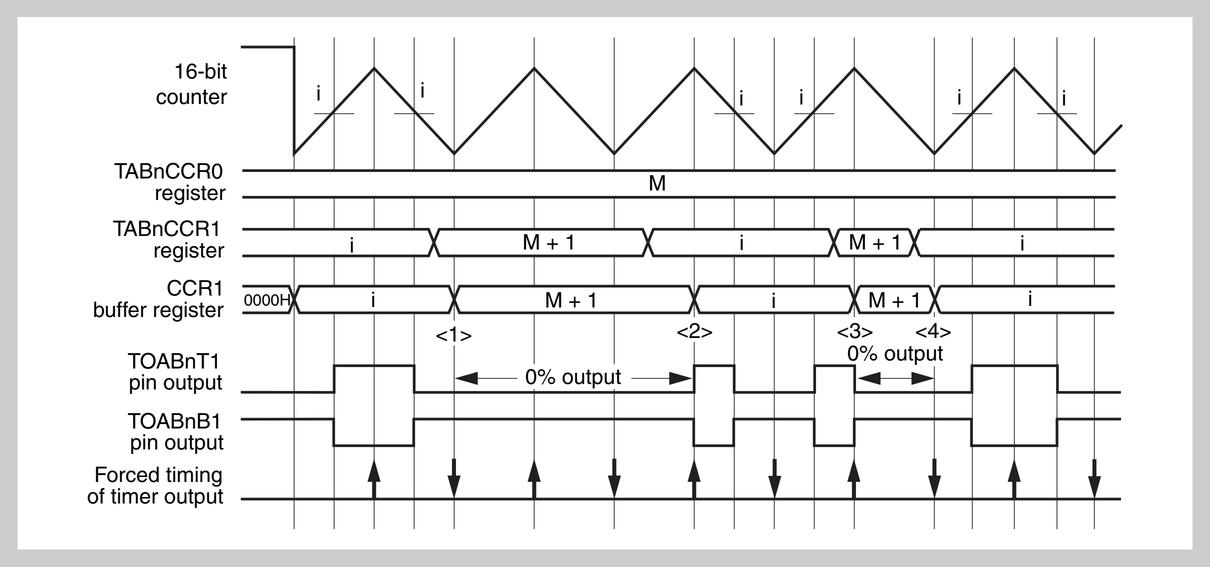


Figure 22-10 0% PWM Output Waveform (Without Dead Time)

<1> 0% output is selected by the valley interrupt (without a match with the 16-bit counter).
The valley interrupt forcibly lowers the timer output. This produces the 0% output.

<2> 0% output is cancelled by the crest interrupt (without a match with the 16-bit counter).
The crest interrupt forcibly raises the timer output. This cancels the 0% output.

<3> The crest interrupt forcibly raises the timer output, but lowering the timer output takes precedence when the value of the TABnCCRm register matches the value of the 16-bit counter. As a result, the 0% wave is output.

<4> The valley interrupt forcibly lowers the timer output. This cancels the 0% output.

Note ↑ means forced raising and ↓ means forced lowering.

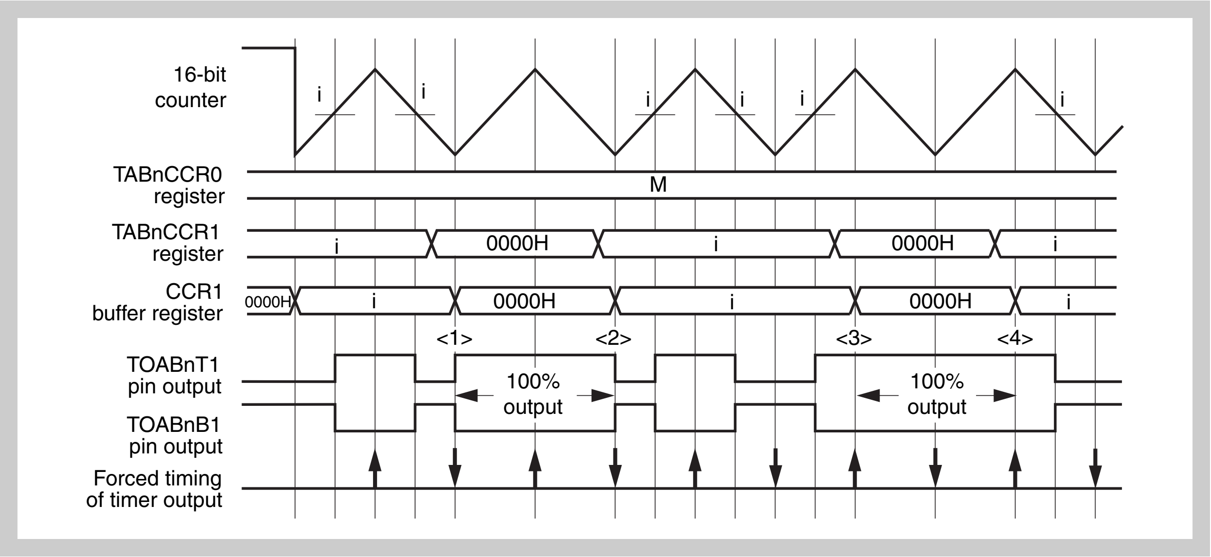


Figure 22-11 100% PWM Output Waveform (Without Dead Time)

<1> 100% output is selected by the valley interrupt (with a match with the 16-bit counter).
The valley interrupt forcibly lowers the timer output, but raising the timer output takes precedence when the value of the TABnCCRm register matches the value of the 16-bit counter. As a result, the 100% output is produced.

<2> 100% output is cancelled by the valley interrupt (without a match with the 16-bit counter).
The valley interrupt forcibly lowers the timer output. This cancels the 100% output.

<3> 100% output is selected by the crest interrupt (without a match with the 16-bit counter).
The crest interrupt forcibly raises the timer output. This produces the 100% output.

<4> 100% output is cancelled by the crest interrupt (without a match with the 16-bit counter).
The crest interrupt forcibly raises the timer output. This cancels the 100% output.

Note ↑ means forced raising and ↓ means forced lowering.

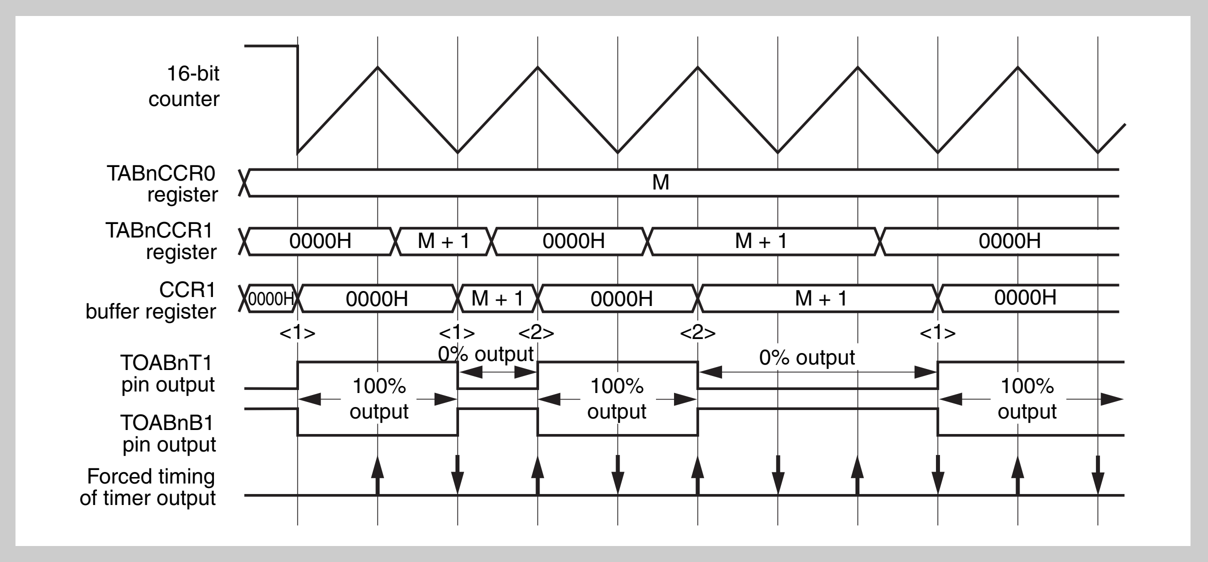


Figure 22-12 PWM Output Waveform from 0% to 100% and from 100% to 0% (Without Dead Time)

<1> The valley interrupt selects 100% ←→ 0% or 0% ←→ 100% output.
Output can be selected from 100% ←→ 0% or 0% ←→ 100% immediately after the timer has been started.

<2> The crest interrupt selects 100% ←→ 0% output.
The crest interrupt selects 100% → 0% output by using the timer output forced raising function and by a match between the 16-bit counter value and the TABnCCR0 register value.

(3) Output wave in vicinity of 0% and 100% output

If an interrupt is generated because the value of the 16-bit counter matches the value of the compare register while dead time is being counted, the dead-time counter is cleared and starts its count operation again.
The output waveform of dead-time control in the vicinity of 0% and 100% output is shown below.

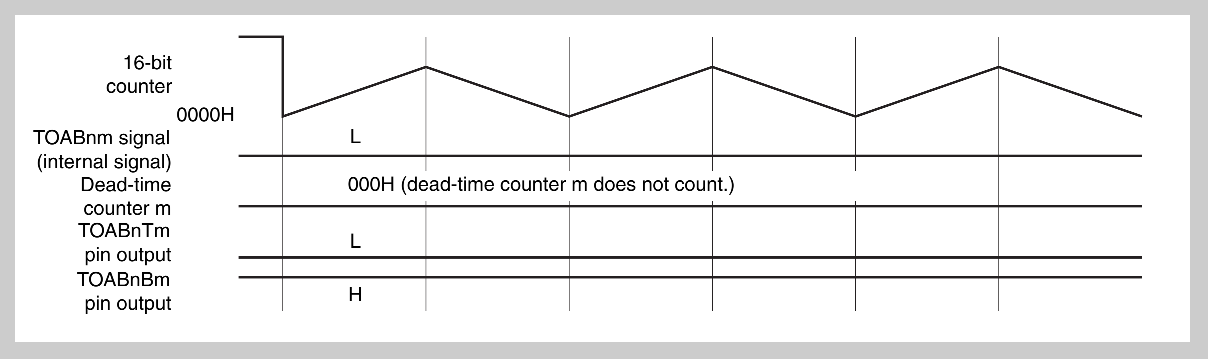


Figure 22-13 PWM Output Waveform with Dead Time (2)
0% output (TABnCCRm register = M + 1, TABnCCR0 register = M, TABnDTC register = a)

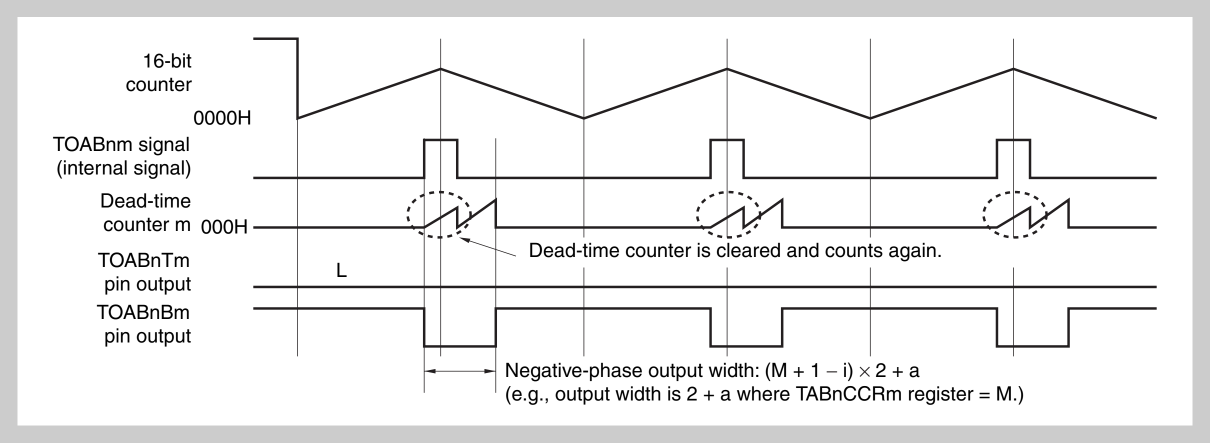

16-bit counter
0000H
TOABnm signal (internal signal)
Dead-time counter m 000H
Dead-time counter is cleared and counts again.
TOABnTm pin output
L
TOABnBm pin output
Negative-phase output width: (M + 1 − i) × 2 + a
(e.g., output width is 2 + a where TABnCCRm register = M.)

**Figure 22-14 PWM Output Waveform with Dead Time (2)
In vicinity of 0% output (TABnCCRm register = i ≥ M + 1 - a/2, TABnCCR0 register = M, TABnDTC register = a)**

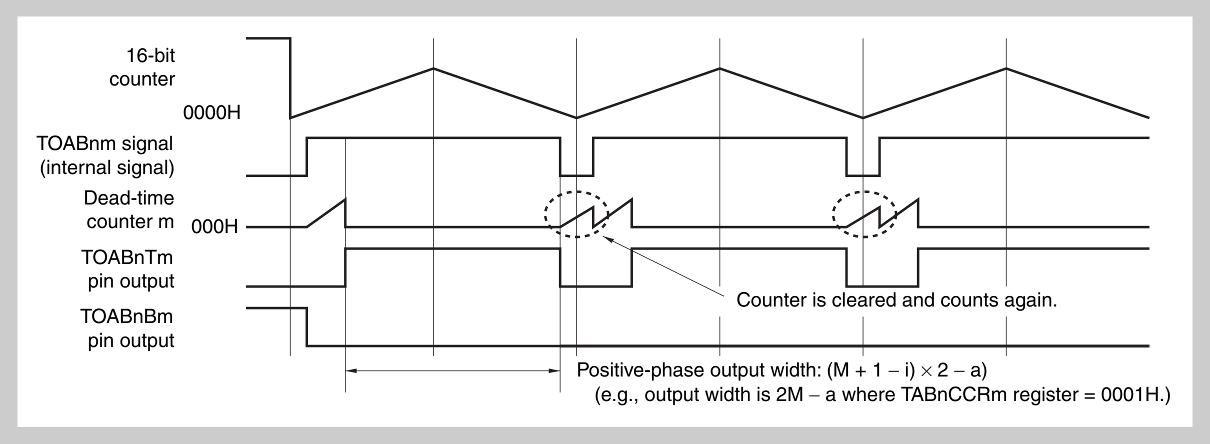

16-bit counter
0000H
TOABnm signal (internal signal)
Dead-time counter m 000H
TOABnTm pin output
Counter is cleared and counts again.
TOABnBm pin output
Positive-phase output width: (M + 1 − i) × 2 − a)
(e.g., output width is 2M − a where TABnCCRm register = 0001H.)

**Figure 22-15 PWM Output Waveform with Dead Time (2)
In vicinity of 100% output (TABnCCRm register = i ≤ a/2, TABnCCR0 register = M, TABnDTC register = a)**

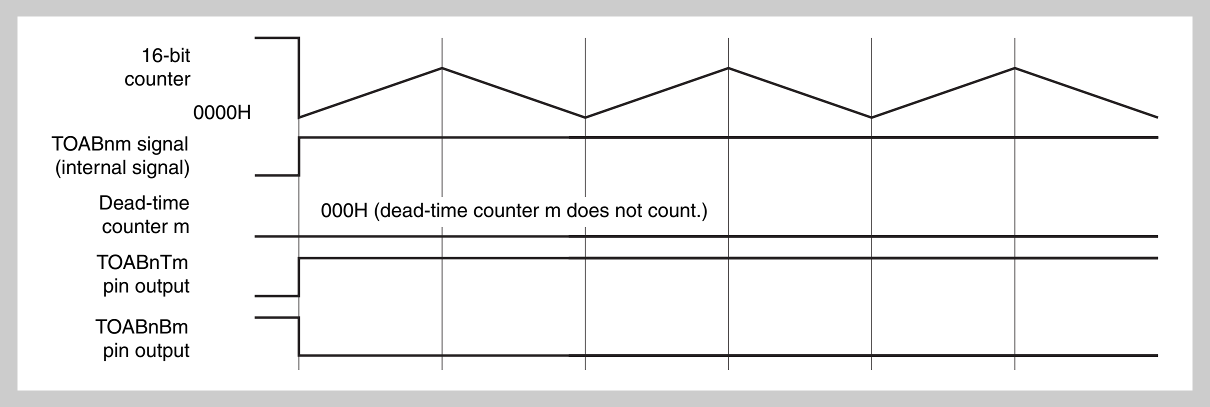


**Figure 22-16 PWM Output Waveform with Dead Time (2)
100% output (TABnCCRm register = 0000H, TABnCCR0 register = M, TABnDTC register = a)**

(4) Automatic dead-time width narrowing function (TABnOPT2.TABnDTM bit = 1)

The dead-time width can be automatically narrowed in the vicinity of 0% output or 100% output by setting the TABnOPT2.TABnDTM bit to 1.
By setting the TABnDTM bit to 1, the dead-time counter is not cleared, but starts down counting if the TOABnm (internal signal) output of Timer AB changes during dead-time counting.
The following timing chart shows the operation of the dead-time counter when the TABnDTM bit is set to 1.

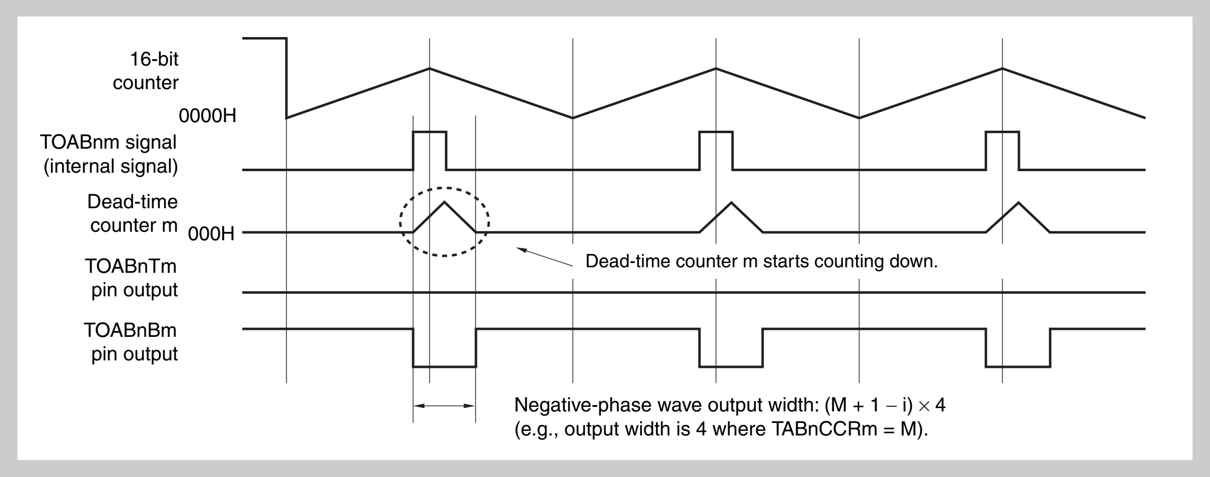

16-bit counter
0000H
TOABnm signal (internal signal)
Dead-time counter m
000H
Dead-time counter m starts counting down.
TOABnTm pin output
TOABnBm pin output
Negative-phase wave output width: (M + 1 − i) × 4
(e.g., output width is 4 where TABnCCRm = M).

Figure 22-17 Operation of Dead-Time Counter m (1)
In vicinity of 0% output (TABnCCRm register = $i \geq M + 1 - a/2$, TABnCCR0 register = M, TABnDTC register = a)

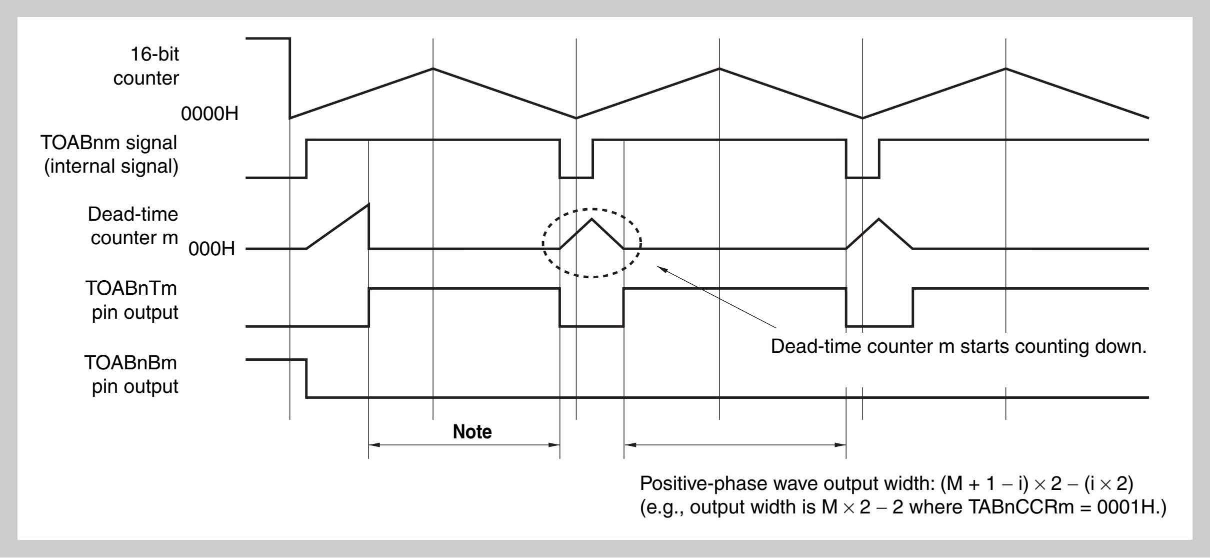

16-bit counter
0000H
TOABnm signal (internal signal)
Dead-time counter m
000H
TOABnTm pin output
Dead-time counter m starts counting down.
TOABnBm pin output
Note
Positive-phase wave output width: (M + 1 − i) × 2 − (i × 2)
(e.g., output width is M × 2 − 2 where TABnCCRm = 0001H.)

Figure 22-18 Operation of Dead-Time Counter m (1)
In vicinity of 100% output (TABnCCRm register = $i \leq a/2$, TABnCCR0 register = M, TABnDTC register = a)

Note The output width of the first wave differs from that of the second and subsequent waves immediately after the TABnCTL0.TABnCE bit has been set. The first wave is shorter than the second wave because the dead time is fully counted.

(5) Dead-time control in case of incorrect setting

Usually, the TOABnm (internal signal) output of TABn changes only once during dead-time counting, only in the vicinity of 0% and 100% output.
This section shows an example where the TABnCCR0 register (carrier cycle) and TABnDTC register (dead-time value) are incorrectly set. If these registers are incorrectly set, the TOABnm (internal signal) output of TABn changes more than once during dead-time counting.
The following flowchart shows the 6-phase PWM output wave in this case.

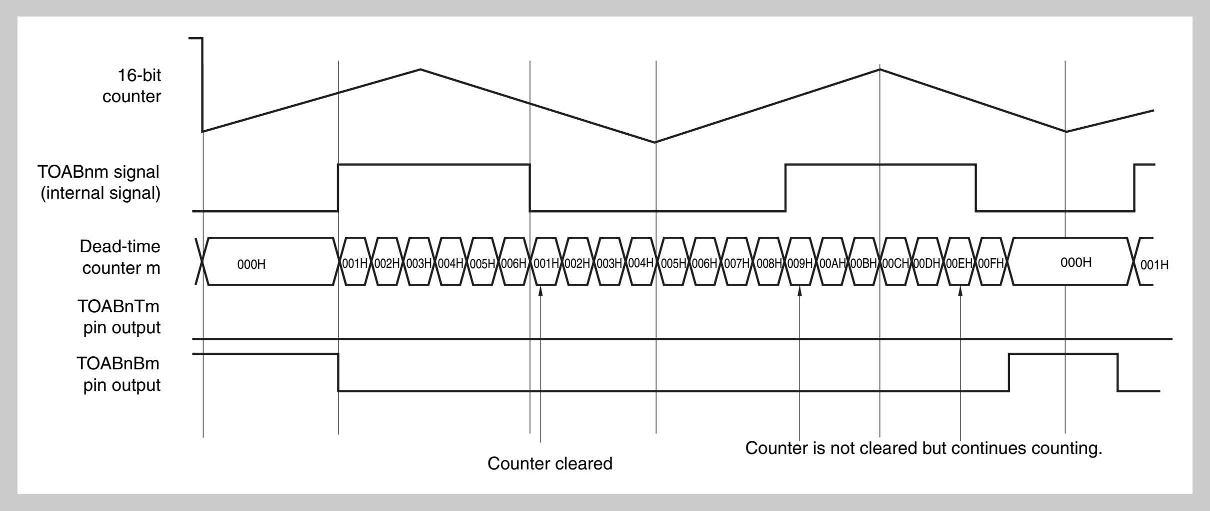


Figure 22-19 Operation of Dead-Time Counter m (2)
When TABnOPT2.TABnDTM bit = 0, TABnCCR0 register = 0006H, TABnDTC register = 000FH, TABnCCRm register = 0004H

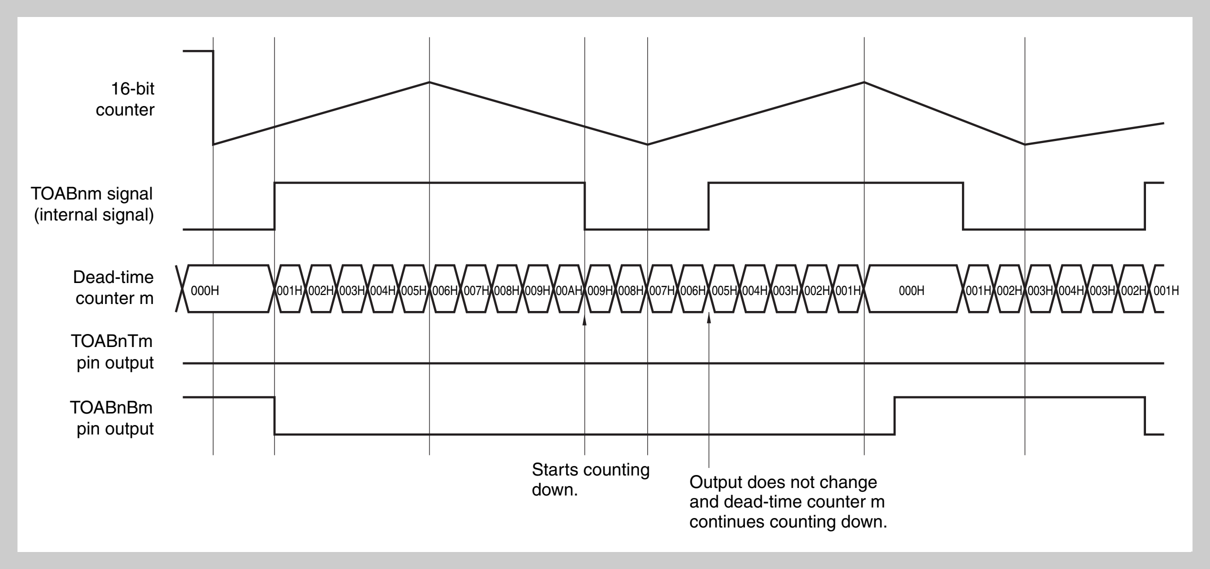


Figure 22-20 Operation of Dead-Time Counter m (2)
When TABnOPT2.TABnDTM bit = 1, TABnCCR0 register = 0006H, TABnDTC register = 000FH, TABnCCRm register = 0002H

22.4.3 Interrupt culling function

- The interrupts to be culled are INTTABnCC0 (crest interrupt) and INTTABnOV (valley interrupt).
- The TABnOPT1.TABnICE bit is used to enable output of the INTTABnCC0 interrupt and the number of times the interrupt is to be culled.
- The TABnOPT1.TABnIOE bit is used to enable output of the INTTABnOV interrupt and the number of times the interrupt is to be culled.
- The TABnOPT2.TABnRDE bit is used to specify whether transfer is to be culled or not.
- If it is specified that transfer is to be culled, transfer is executed at the same timing as the interrupt output after culling. If it is specified that transfer is not to be culled, transfer is executed at the transfer timing after the TABnCCR1 register has been written.
- The TABnOPT0.TABnCMS bit is used to specify whether the registers with a transfer function are batch rewritten or anytime rewritten.
- The values of the registers are updated in synchronization with transferring when the TABnCMS bit is 0. When the TABnCMS bit is 1, the values of the registers are immediately updated when a new value is written to the registers.
- Transfer is performed from the TABnCCRm register to the CCRm buffer register in synchronization with interrupt culling timing.

Caution

1. When using the interrupt culling function in the batch rewrite mode (transfer mode), execute the function in the intermittent batch rewrite mode (transfer culling mode).
2. An interrupt occurs after culling.

(1) Interrupt culling operation

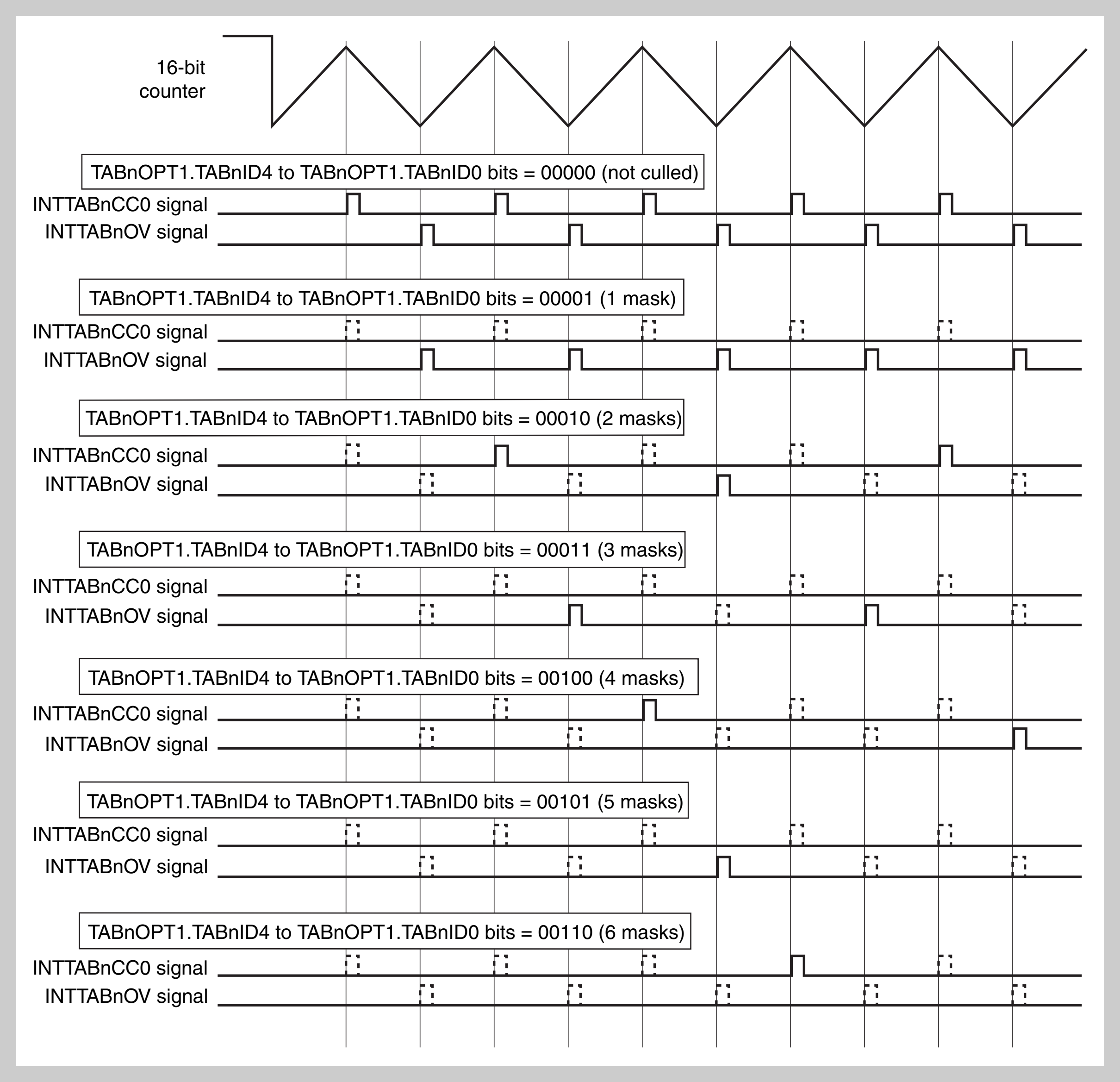

16-bit counter
TABnOPT1.TABnID4 to TABnOPT1.TABnID0 bits = 00000 (not culled)
INTTABnCC0 signal
INTTABnOV signal
TABnOPT1.TABnID4 to TABnOPT1.TABnID0 bits = 00001 (1 mask)
INTTABnCC0 signal
INTTABnOV signal
TABnOPT1.TABnID4 to TABnOPT1.TABnID0 bits = 00010 (2 masks)
INTTABnCC0 signal
INTTABnOV signal
TABnOPT1.TABnID4 to TABnOPT1.TABnID0 bits = 00011 (3 masks)
INTTABnCC0 signal
INTTABnOV signal
TABnOPT1.TABnID4 to TABnOPT1.TABnID0 bits = 00100 (4 masks)
INTTABnCC0 signal
INTTABnOV signal
TABnOPT1.TABnID4 to TABnOPT1.TABnID0 bits = 00101 (5 masks)
INTTABnCC0 signal
INTTABnOV signal
TABnOPT1.TABnID4 to TABnOPT1.TABnID0 bits = 00110 (6 masks)
INTTABnCC0 signal
INTTABnOV signal

Figure 22-21 Interrupt Culling Operation When TABnOPT1.TABnICE Bit = 1, TABnOPT1.TABnIOE Bit = 1, TABnOPT2.TABnRDE Bit = 1 (Crest/Valley Interrupt Output)

Note ⊓: Culled interrupt

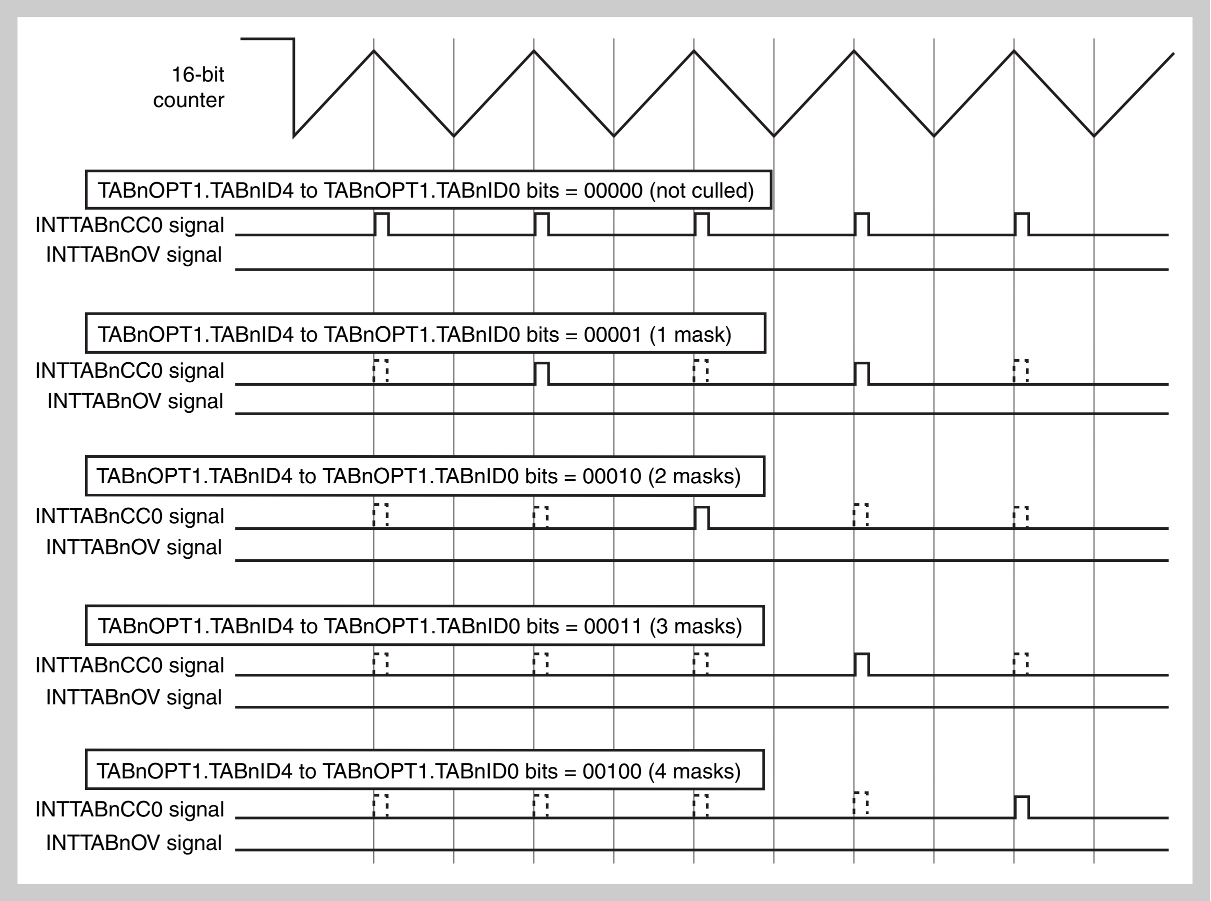

16-bit counter
TABnOPT1.TABnID4 to TABnOPT1.TABnID0 bits = 00000 (not culled)
INTTABnCC0 signal
INTTABnOV signal
TABnOPT1.TABnID4 to TABnOPT1.TABnID0 bits = 00001 (1 mask)
INTTABnCC0 signal
INTTABnOV signal
TABnOPT1.TABnID4 to TABnOPT1.TABnID0 bits = 00010 (2 masks)
INTTABnCC0 signal
INTTABnOV signal
TABnOPT1.TABnID4 to TABnOPT1.TABnID0 bits = 00011 (3 masks)
INTTABnCC0 signal
INTTABnOV signal
TABnOPT1.TABnID4 to TABnOPT1.TABnID0 bits = 00100 (4 masks)
INTTABnCC0 signal
INTTABnOV signal

Figure 22-22 Interrupt Culling Operation When TABnOPT1.TABnICE Bit = 1, TABnOPT1.TABnIOE Bit = 0, TABnOPT2.TABnRDE bit = 1 (Crest Interrupt Output)

Note ⸬: Culled interrupt

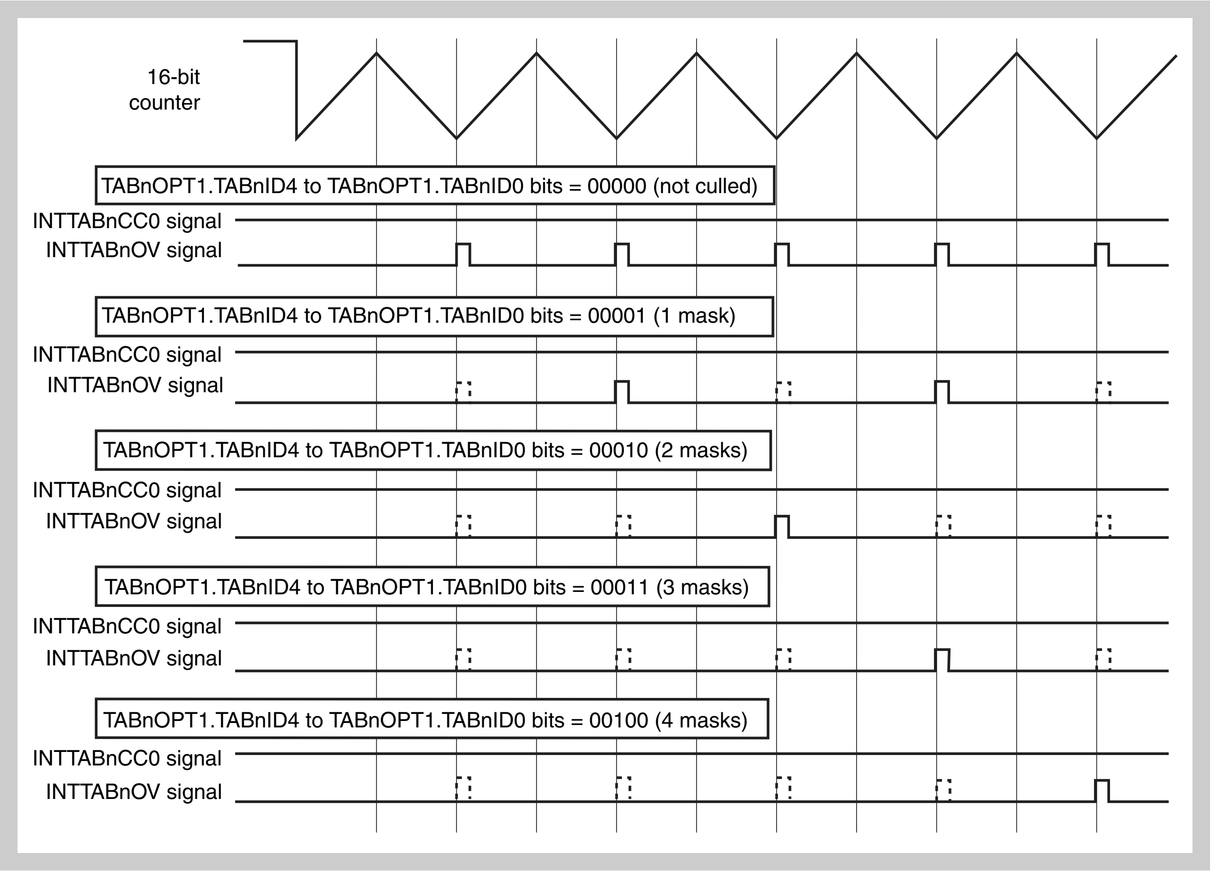

16-bit counter
TABnOPT1.TABnID4 to TABnOPT1.TABnID0 bits = 00000 (not culled)
INTTABnCC0 signal
INTTABnOV signal
TABnOPT1.TABnID4 to TABnOPT1.TABnID0 bits = 00001 (1 mask)
INTTABnCC0 signal
INTTABnOV signal
TABnOPT1.TABnID4 to TABnOPT1.TABnID0 bits = 00010 (2 masks)
INTTABnCC0 signal
INTTABnOV signal
TABnOPT1.TABnID4 to TABnOPT1.TABnID0 bits = 00011 (3 masks)
INTTABnCC0 signal
INTTABnOV signal
TABnOPT1.TABnID4 to TABnOPT1.TABnID0 bits = 00100 (4 masks)
INTTABnCC0 signal
INTTABnOV signal

Figure 22-23 Interrupt Culling Operation When TABnOPT1.TABnICE Bit = 0, TABnOPT1.TABnIOE Bit = 1, TABnOPT2.TABnRDE bit = 1 (Valley Interrupt Output)

Note ⌈⌉: Culled interrupt

(2) To alternately output crest interrupt (INTTABnCC0) and valley interrupt (INTTABnOV)

To alternately output the crest and valley interrupts, set both the TABnOPT1.TABnICE and TABnOPT1.TABnIOE bits to 1.

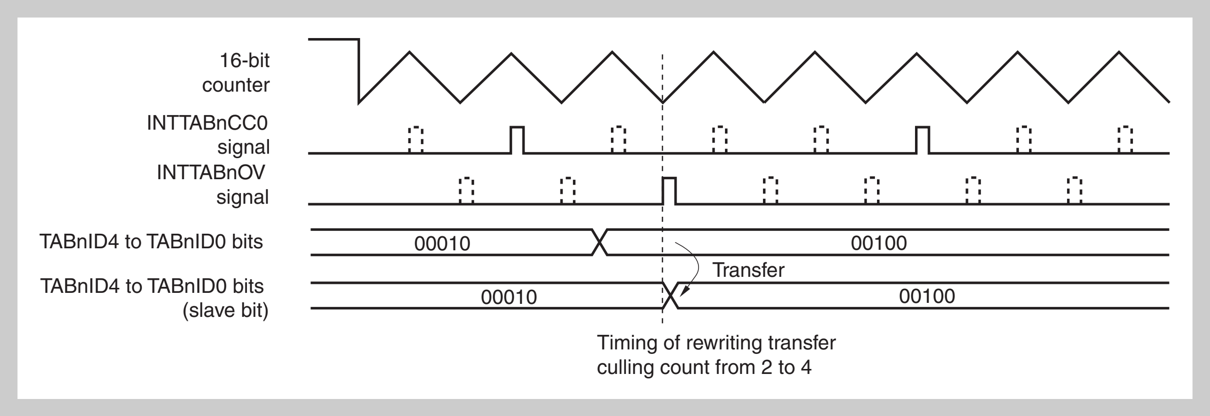


Figure 22-24 Crest/Valley Interrupt Output
TABnOPT0.TABnCMS bit = 0, TABnOPT2.TABnRDE bit = 1 (with transfer culling control)

Note 1. Transfer is performed when the culled interrupt is output. The other transfer timing is ignored.

2. ⎍: Culled interrupt

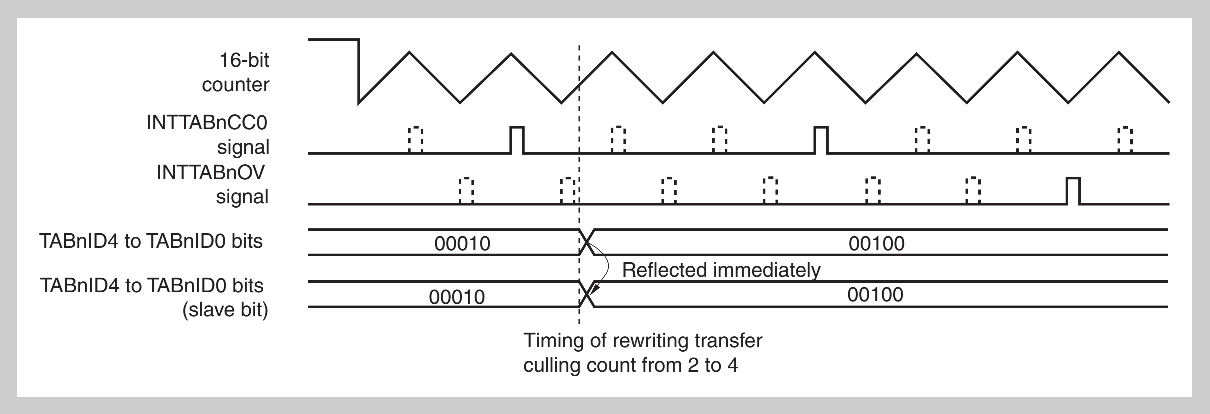


Figure 22-25 Crest/Valley Interrupt Output
TABnCMS bit = 1, TABnRDE bit = 0 or 1 (without transfer control)

Note 1. Rewriting is reflected immediately. The transfer timing is ignored.

2. ⎍: Culled interrupt

(3) To output only crest interrupt (INTTABnCC0)

Set the TABnOPT1.TABnICE bit to 1 and clear the TABnIOE bit to 0.

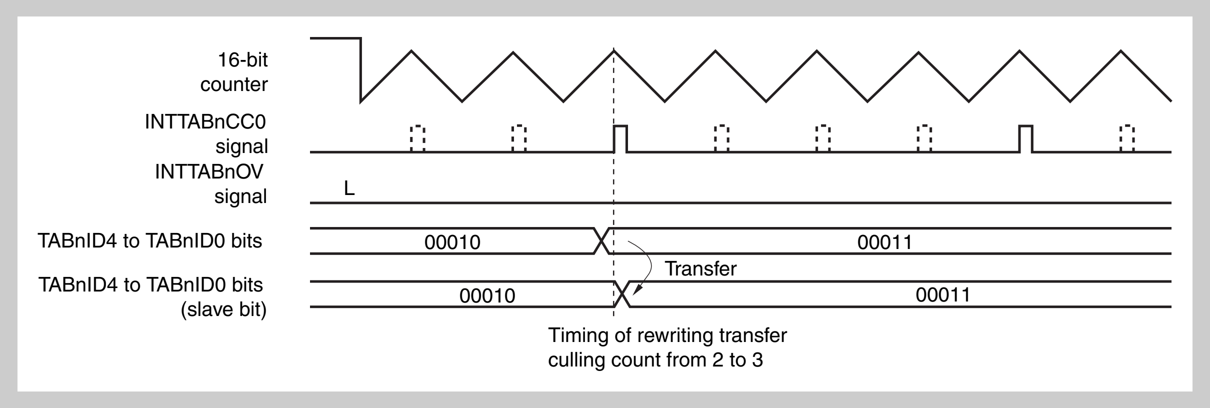


Figure 22-26 Crest Interrupt Output TABnOPT0.TABnCMS bit = 0, TABnOPT2.TABnRDE bit = 1 (with transfer culling control)

Note

1. Transfer is performed when the culled interrupt is output. The other transfer timing is ignored.
2. ⊓: Culled interrupt

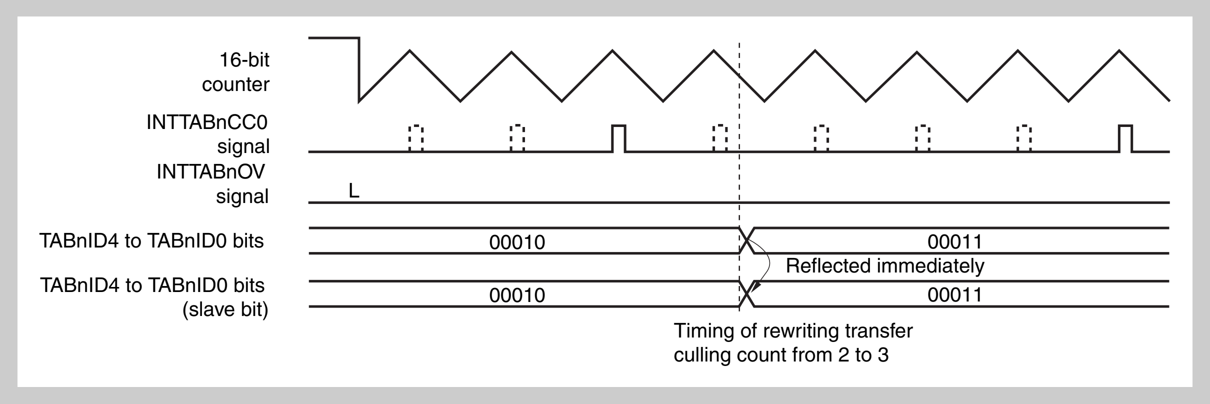


Figure 22-27 Crest Interrupt Output TABnOPT0.TABnCMS bit = 1, TABnOPT0.TABnRDE bit = 0 or 1 (without transfer control)

Note

1. Rewriting is reflected immediately. The transfer timing is ignored.
2. ⊓: Culled interrupt

(4) To output only valley interrupt (INTTABnOV)

Clear the TABnOPT1.TABnICE bit to 0 and clear the TABnIOE bit to 1.

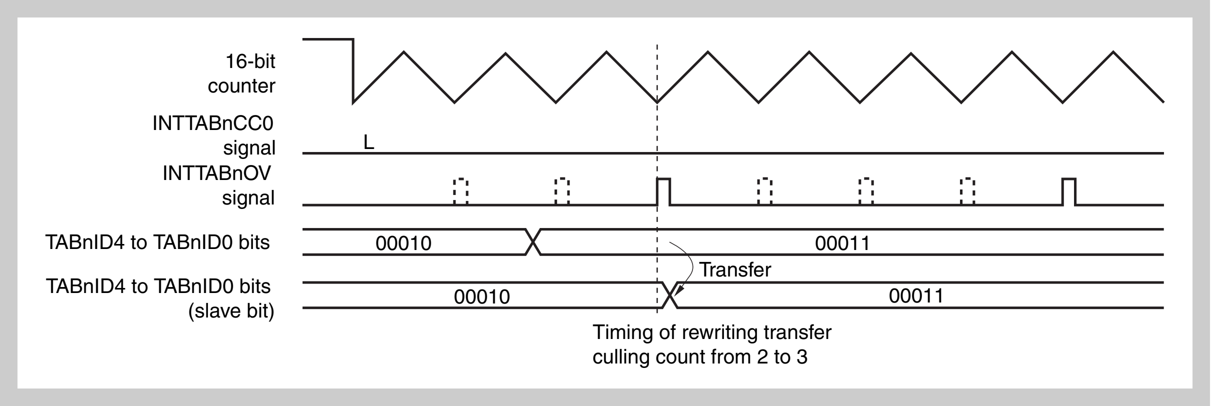


Figure 22-28 Valley Interrupt Output TABnOPT0.TABnCMS bit = 0, TABnOPT2.TABnRDE bit = 1 (with transfer culling control)

Note
1. Transfer is performed when the culled interrupt is output. The other transfer timing is ignored.
2. ⊓: Culled interrupt

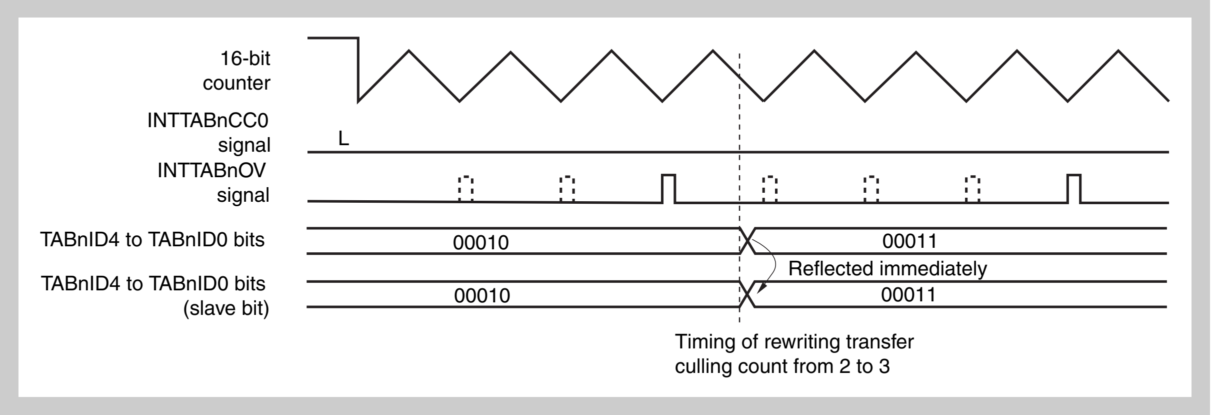


Figure 22-29 Valley Interrupt Output TABnOPT0.TABnCMS bit = 1, TABnOPT0.TABnRDE bit = 0 or 1 (without transfer control)

Note
1. Rewriting is reflected immediately. The transfer timing is ignored.
2. ⊓: Culled interrupt

22.4.4 Operation to rewrite register with transfer function

The following seven registers are provided with a transfer function and used to control a motor. Each of registers has a buffer register.

- TABnCCR0:
  Register that specifies the cycle of the 16-bit counter (TAB)
- TABnCCR1:
  Register that specifies the duty factor of TOABnT1 (U) and TOABnB1 (U)
- TABnCCR2:
  Register that specifies the duty factor of TOABnT2 (V) and TOABnB2 (V)
- TABnCCR3:
  Register that specifies the duty factor of TOABnT3 (W) and TOABnB3 (W)
- TABnOPT1:
  Register that specifies the culling of interrupts
- TAAxCCR0:
  Register that specifies the A/D conversion start trigger generation timing (TAAm during tuning operation)
- TAAxCCR1:
  Register that specifies the A/D conversion start trigger generation timing (TAAx during tuning operation)

The following three rewrite modes are provided in the registers with a transfer function.

- Anytime rewriting mode
  This mode is specified by setting the TABnOPT0.TABnCMS bit to 1.
  In this mode, each compare register is updated independently, and the value of the compare register is updated as soon as a new value is written to it.
- Batch rewrite mode (transfer mode)
  This mode is set by clearing the TABnOPT0.TABnCMS bit to 0, the TABnOPT1.TABnID4 to TABnOPT1.TABnID0 bits to 00000, and the TABnOPT2.TABnRDE bit to 0. When data is written to the TABnCCR1 register, the seven registers are transferred to the buffer register all at once at the next transfer timing. Unless the TABnCCR1 register is rewritten, the transfer operation is not performed even if the other six registers are rewritten.
  The transfer timing is the timing of each crest (match between the 16-bit counter value and TABnCCR0 register value) and valley (match between the 16-bit counter value and 0001H) regardless of the interrupt.
- Intermittent batch rewrite mode (transfer culling mode)
  This mode is set by clearing the TABnOPT0.TABnCMS bit to 0 and setting the TABnOPT2.TABnRDE bit to 1.
  When data is written to the TABnCCR1 register, the seven registers are transferred to the buffer register all at once at the next transfer timing. Unless the TABnCCR1 register is rewritten, the transfer operation is not performed even if the other six registers are rewritten.
  If interrupt culling is specified by the TABnOPT1 register, the transfer timing is also culled as the interrupts are culled, and the seven registers are transferred all at once at the culled timing of crest interrupt (match between the 16-bit counter value and TABnCCR0 register value) or valley interrupt (match between the 16-bit counter value and 0001H).
  For details of the interrupt culling function, see *“Interrupt culling function” on page 885*.

(1) Anytime rewriting mode

This mode is set when the TABnOPT0.TABnCMS bit is 1. The setting of the TABnOPT2.TABnRDE bit is ignored.
In this mode, the value written to each register with a transfer function is immediately transferred to an internal buffer register and compared with the value of the counter. If a register with transfer function is rewritten in this mode after the count value of the 16-bit counter matches the value of the TABnCCRm register, the rewritten value is not reflected because the next match is ignored after the first match has occurred. If the register is rewritten during up counting, the new register value becomes valid after the counter has started counting down.

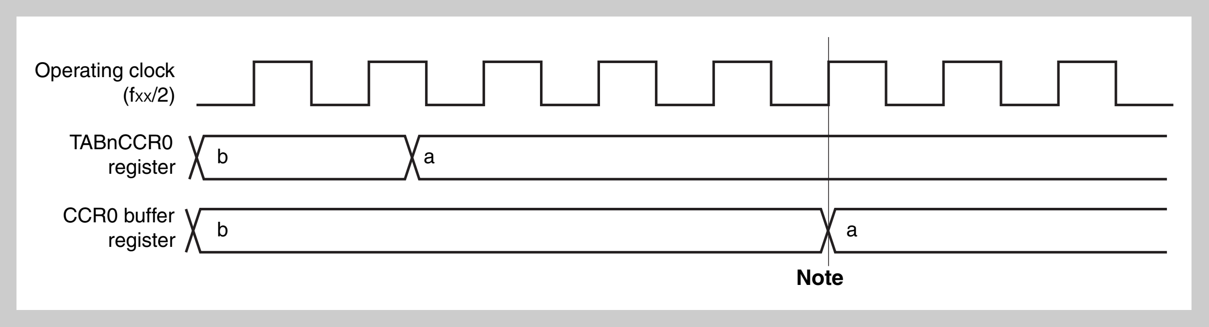


Figure 22-30 Timing of Reflecting Rewritten Value

Note After the register (TABnCCR0, TABnCCR2, TABnCCR3, TABnOPT1, TAAxCCR0, or TAAxCCR1) has been written, the written value is transferred to an internal buffer register after four clocks of the operating clock. However, the value of only the TABnCCR1 register is transferred after 5 more clocks.

(a) Rewriting TABnCCR0 register

Even if the TABnCCR0 register is rewritten in the anytime rewriting mode, the new value may not be reflected in some cases.

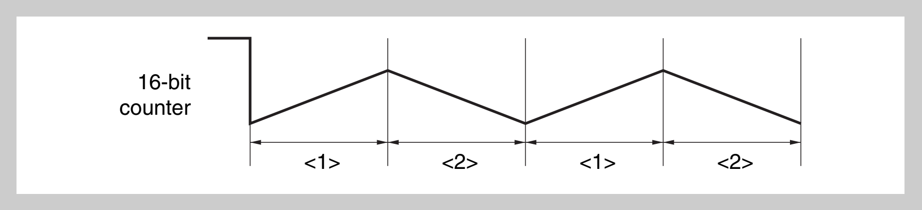


Figure 22-31 Example of Rewriting TABnCCR0 Register

- Rewriting during period <1> (rewriting during up counting)

  If the newly rewritten value is greater than the value of the 16-bit counter, there is no problem because it will match the value of the 16-bit counter. If the new value is less than the value of the 16-bit counter, it will not match the value of the counter. As a result, the 16-bit counter overflows and continues counting up from 0000H until it matches the register value again, and the correct PWM waveform is not output.

- Rewriting during period <2> (rewriting during down counting)

  A match with the value of the 16-bit counter is ignored during counting down. Therefore, the rewritten period value is reflected starting from counting up in the next cycle as a match point.

(b) Rewriting TABnCCRm register

Figure 22-33 to *Figure 22-34* show the timing of rewriting before the value of the 16-bit counter matches the value of the TABnCCRm register (<1> in *Figure 22-32*), and *Figure 22-35* shows the timing of rewriting after the value of the 16-bit counter matches the value of the TABnCCRm register (<2> in *Figure 22-32*).

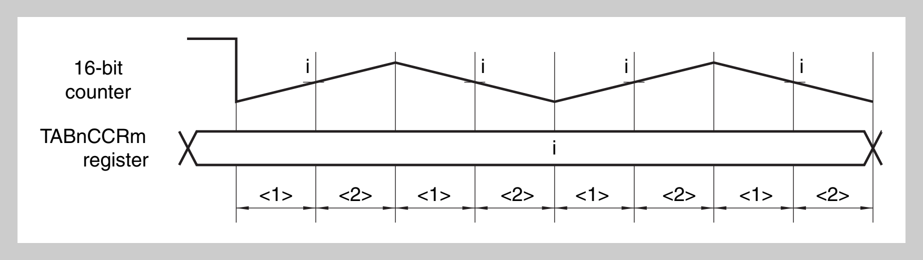


Figure 22-32 Basic Operation of 16-Bit Counter and TABnCCRm Register

Note i = Set value of TABnCCRm register

If the TABnCCRm register is rewritten before its value matches the value of the 16-bit counter, the register value will match the value of the 16-bit counter after the register has been rewritten. Consequently, the new register value is immediately reflected.

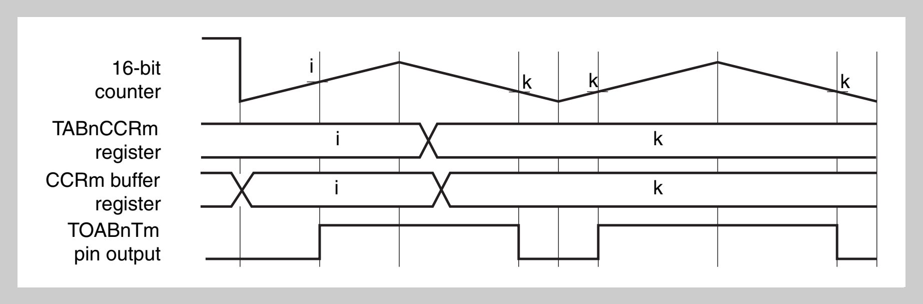


Figure 22-33 Example a) of Rewriting TABnCCR1 to TABnCCR3 Registers (Rewriting Before Match Occurs)

If a value less than the value of the 16-bit counter (greater if the counter is counting down) is written to the TABnCCRm register, the output waveform is as follows because the register value does not match the counter value.

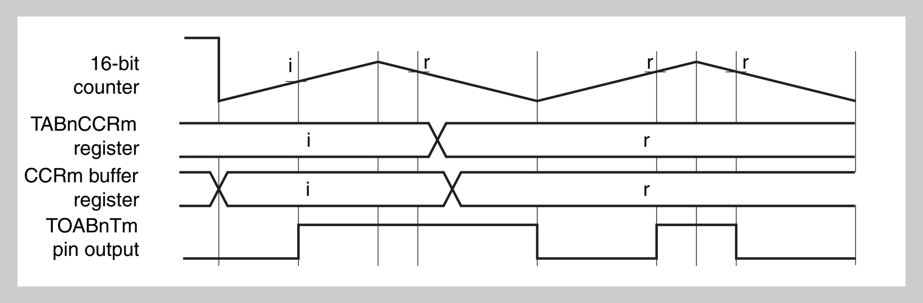


Figure 22-34 Example b) of Rewriting TABnCCR1 to TABnCCR3 Registers (Rewriting Before Match Occurs)

If the register value does not match the counter value, the TOABnTm pin output does not change. Even if the value of the 16-bit counter does not match the value of the TABnCCRm register, the TOABnTm pin output always

changes to the high level if the crest interrupt occurs and to the low level if the valley interrupt occurs.
This is a function provided for 0% output and 100% output.
For details, see *"PWM output of 0%/100%" on page 879*.

Note i, r, k = Set values of TABnCCRm register

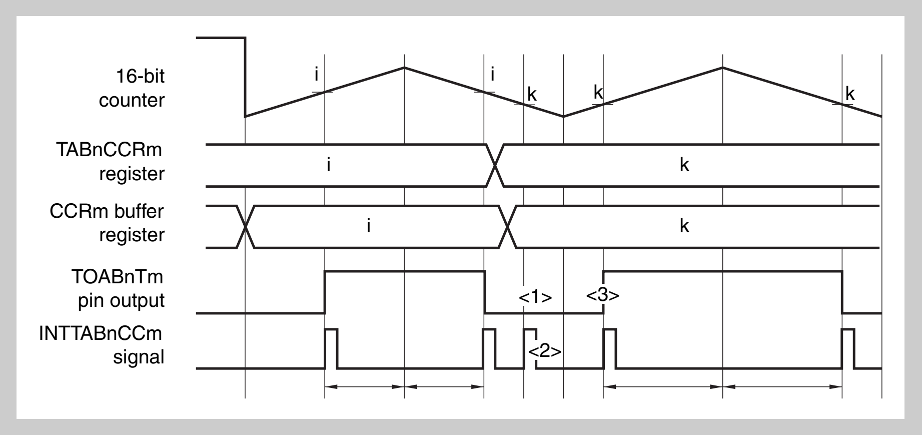


Figure 22-35 Example of Rewriting TABnCCR1 to TABnCCR3 Registers (Rewriting After Match Occurs)

<1> Matching of the count value of the 16-bit counter and the value of the TABnCCRm register as a result of rewriting the register is ignored after a match signal has been generated, and the PWM output does not change.

<2> Even if the PWM output does not change, the interrupt generated upon a match between the 16-bit counter value and the TABnCCRm register value (INTTABnCCm) is output.

<3> The next match between the 16-bit counter and TABnCCRm register is valid after the counter has changed its counting direction to up or down, and the PWM output changes.

If the TABnCCRm register is rewritten after its value matches the value of the 16-bit counter, the next match is ignored after the first match occurs and the rewritten value is not reflected to the TOABnTm pin output. If the register is rewritten while the counter is counting down, the match that occurs after the counter starts counting down is valid (the match that occurs after the counter has started counting up is valid if the register is rewritten while the counter is counting up).

Note i, r, k = Set value of TABnCCRm register

(c) Rewriting TABnOPT1 register

The interrupt culling counter is cleared when the TABnOPT1 register is written. When the interrupt culling counter has been cleared, the measured number of times the interrupt has occurred is discarded.
Consequently, the interrupt generation interval is temporarily extended.
To avoid this operation, rewrite the TABnOPT1 register in the intermittent batch rewriting mode (transfer culling mode).
For details of rewriting the TABnOPT1 register, see *“Interrupt culling function” on page 885*.

(2) Batch rewrite mode (transfer mode)

This mode is set by clearing the TABnOPT0.TABnCMS bit to 0, the TABnOPT1.TABnID4 to TABnOPT1.TABnID0 bits to 00000, and the TABnOPT2.TABnRDE bit to 0.
In this mode, the values written to each compare register are transferred to the internal buffer register all at once at the transfer timing and compared with the counter value.

(a) Rewriting procedure

If data is written to the TABnCCR1 register, the values set to the TABnCCR0 to TABnCCR3, TABnOPT1, TAAxCCR0, and TAAxCCR1 registers are transferred all at once to the internal buffer register at the next transfer timing. Therefore, write to the TABnCCR1 register last. Writing to the register is prohibited after the TABnCCR1 register has been written and before the transfer timing is generated (until the crest (match between the 16-bit counter value and TABnCCR0 register value) or the valley (match between the 16-bit counter value and 0001H)).
The operation procedure is as follows.

<1> Rewriting the TABnCCR0, TABnCCR2, TABnCCR3, TABnOPT1, TAAxCCR0, and TAAxCCR1 registers
Do not rewrite registers that do not have to be rewritten.

<2> Rewriting the TABnCCR1 register
Rewrite the same value to the register even when it is not necessary to rewrite the TABnCCR1 register.

<3> Holding the next rewriting pending until the transfer timing is generated
Rewrite the register next time after the INTTABnOV or INTTABnCC0 interrupt has occurred.

<4> Return to <1>.

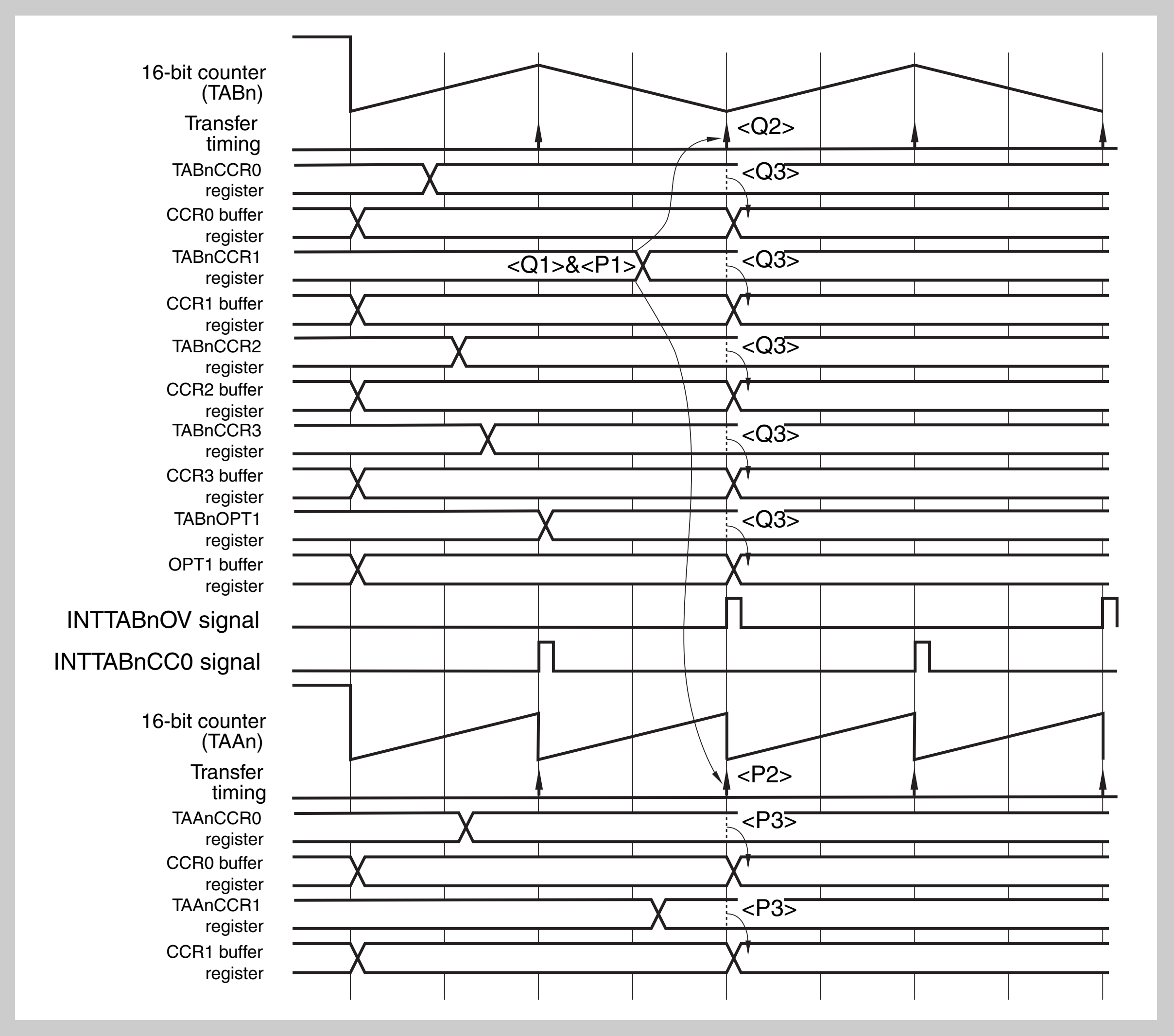


Figure 22-36 Basic Operation in Batch Mode

[Operation of TABn]

<Q1> Write the TABnCCR1 register

<Q2> The target timing is the first transfer timing after a write to the TABnCCR1 register.

<Q3> The values are transferred all at once at the transfer timing.

[Operation of TAAx]

<P1> Write the TABnCCR1 register

<P2> The target timing is the first transfer timing after a write to the TABnCCR1 register.

<P3> The values are transferred all at once at the transfer timing.

(b) Rewriting TABnCCR0 register

When rewriting the TABnCCR0 register in the batch rewrite mode, the output waveform differs depending on whether transfer occurs at the crest (match between the 16-bit counter value and TABnCCR0 register value) or at the valley (match between the 16-bit counter value and 0001H). Usually, it is recommended to rewrite the TABnCCR0 register while the 16-bit counter is counting down, and transfer the register value at the transfer timing of the crest timing.

Figure 22-38 to *Figure 22-39* show an example of rewriting the TABnCCR0 register while the 16-bit counter is counting up (during period <1> in *Figure 22-37*). *Figure 22-40* shows an example of rewriting the TABnCCR0 register while the counter is counting down (during period <2> in *Figure 22-37*).

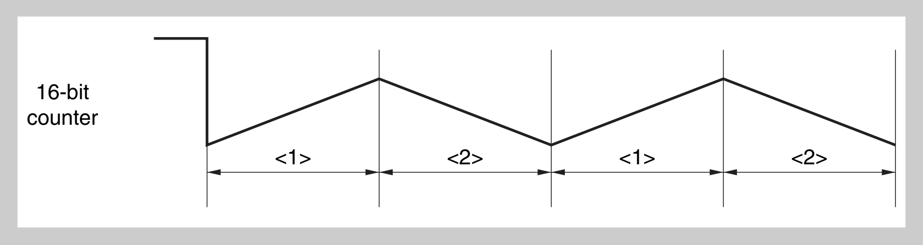


Figure 22-37 Basic Operation of 16-Bit Counter

The transfer timing in *Figure 22-38* to *Figure 22-39* is at the point where the crest timing occurs. While the 16-bit counter is counting down, the cycle changes and an asymmetrical triangular wave is output. Because the cycle changes, rewrite the duty factor (voltage data value).

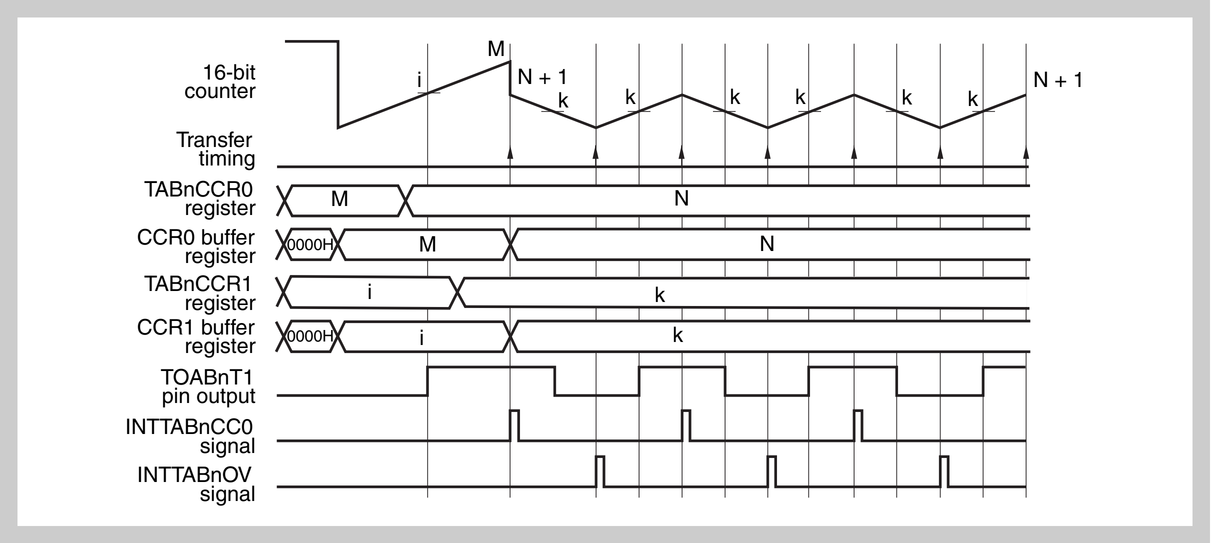


Figure 22-38 Example with M > N of Rewriting TABnCCR0 Register (During Up Counting)

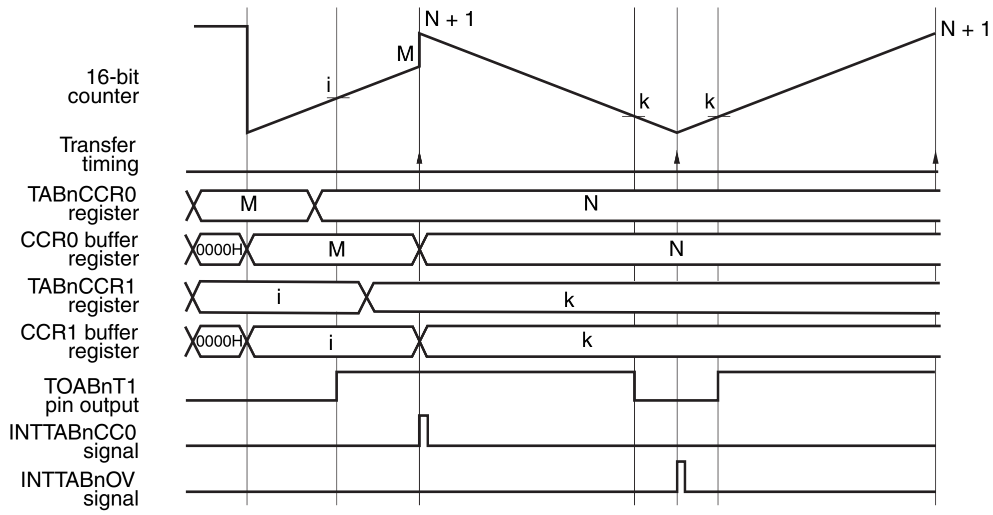


Figure 22-39 **Example with M > N of Rewriting TABnCCR0 Register (During Up Counting)**

Note **1.** If transfer (match between the value of the 16-bit counter and the value of the CCR0 buffer register) occurs in the 6-phase PWM output mode, the value of the TABnCCR0 register plus 1 is loaded to the 16-bit counter. In this way, the expected wave can be output even if the cycle value is changed at the transfer timing of the crest (match between the 16-bit counter value and the TABnCCR0 register value) timing.

2. M: Value of CCR0 buffer register before rewriting
N: Value of CCR0 buffer register after rewriting

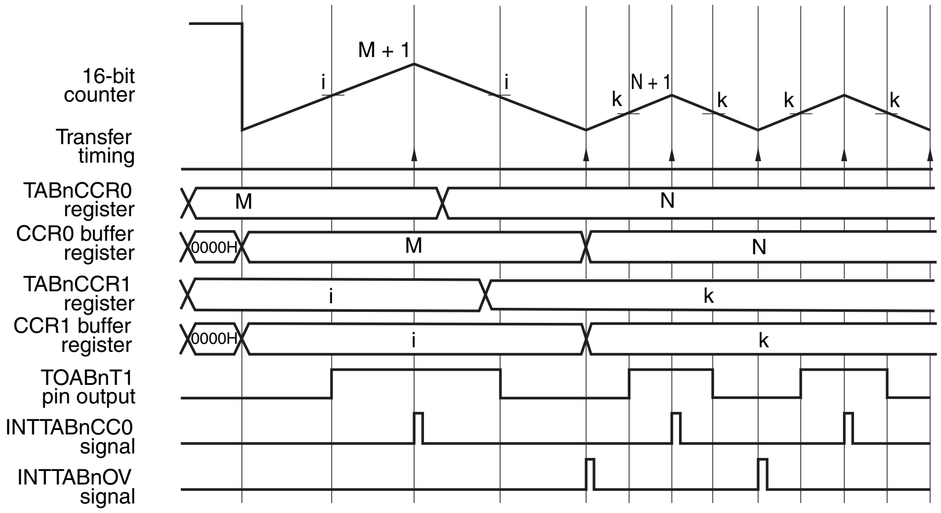


Figure 22-40 **Example of Rewriting TABnCCR0 Register (During Down Counting)**

Because the next transfer timing is at the point of the valley (match between the 16-bit counter value and 0001H), the cycle value changes from the next cycle and output of a symmetrical triangular wave is maintained. Because the cycle changes, rewrite the duty value (voltage data value) as required.

(c) Rewriting TABnCCRm register

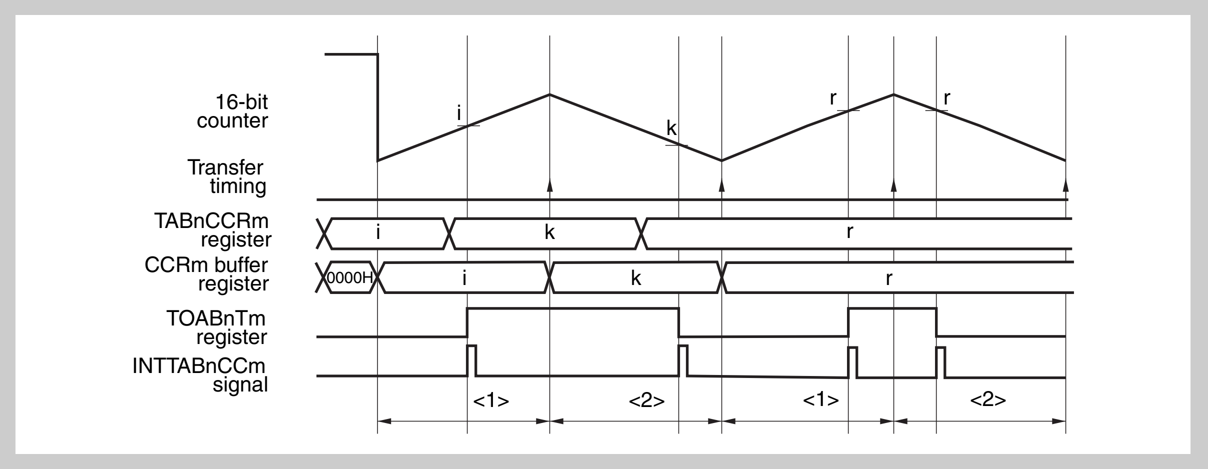


Figure 22-41 Example of Rewriting TABnCCRm Register

- Rewriting during period <1> (rewriting during counting up)

  Because the TABnCCRm register value is transferred at the transfer timing of the crest (match between the 16-bit counter value and TABnCCR0 register value), an asymmetrical triangular wave is output.

- Rewriting during period <2> (rewriting during counting down)

  Because the TABnCCRm register value is transferred at the transfer timing of the valley (match between the 16-bit counter value and 0001H), a symmetrical triangular wave is output.

(d) Transferring TABnOPT1 register value

Do not set the TABnOPT1.TABnID4 to TABnOPT1.TABnID0 bits to other than 00000B. When using the interrupt culling function, rewrite the TABnOPT1 register in the intermittent batch rewrite mode (transfer culling mode).
For details of rewriting the TABnOPT1 register, see *“Interrupt culling function” on page 885*.

(3) Intermittent batch rewriting mode (transfer culling mode)

This mode is set when the TABnOPT0.TABnCMS bit is 0 and the TABnOPT2.TABnRDE bit to 1.
In this mode, the values written to each compare register are transferred to the internal buffer register all at once at the culled transfer timing and compared with the counter value.
The transfer timing is the timing at which an interrupt is generated (INTTABnCC0, INTTABnOV) by interrupt culling.
For details of the interrupt culling function, see *"Interrupt culling function" on page 885*.

(a) Rewriting procedure

If data is written to the TABnCCR1 register, the TABnCCR0 to TABnCCR3, TABnOPT1, TAAxCCR0, and TAAxCCR1 registers are transferred all at once to the internal buffer register at the next transfer timing.
Therefore, write to the TABnCCR1 register last. Writing to the register is prohibited after the TAB0CCR1 register has been written until the transfer timing is generated (until the INTTABnOV or INTTABnCC0 interrupt occurs).
The operation procedure is as follows.

<1> Rewrite the TABnCCR0, TABnCCR2, TABnCCR3, TABnOPT1, TAAxCCR0, and TAAxCCR1 registers.
Do not rewrite registers that do not have to be rewritten.

<2> Rewrite the TABnCCR1 register.
Rewrite the same value to the register even when it is not necessary to rewrite the TABnCCR1 register.

<3> Hold the next rewriting pending until the transfer timing is generated.
Perform the next rewrite after the INTTABnOV or INTTABnCC0 interrupt has occurred.

<4> Return to <1>.

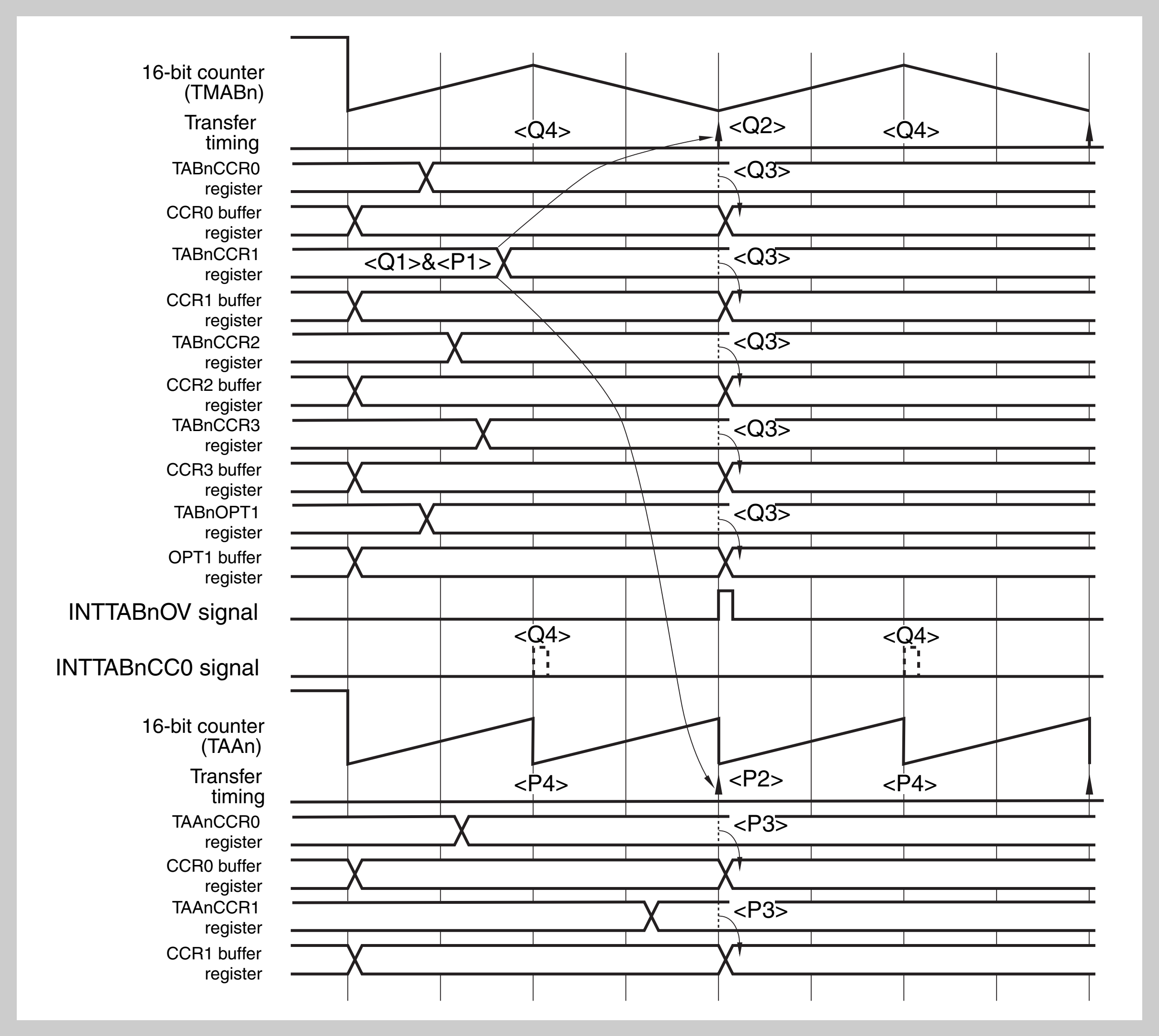


Figure 22-42 Basic Operation in Intermittent Batch Rewriting Mode

[TABn operation]

<Q1> Write the TABnCCR1 register

<Q2> Rewrite the register at the transfer timing that is generated after the TABnCCR1 register has been rewritten.

<Q3> The registers are transferred all at once at the transfer timing.

<Q4> The transfer timing is also culled as the interrupts are culled.

[TAAx operation]

<P1> Write the TABnCCR1 register

<P2> Rewrite the register at the transfer timing that is generated after the TABnCCR1 register has been rewritten.

<P3> The registers are transferred all at once at the transfer timing.

<P4> The transfer timing is also culled as the interrupts are culled.

Note This is an example of the operation when the TABnOPT1.TABnICE bit = 1, TABnOPT1.TABnIOE bit = 1, TABnOPT1.TABnID4 to TABnOPT1.TABnID0 bits = 00001.

(b) Rewriting TABnCCR0 register

When rewriting the TABnCCR0 register in the intermittent batch mode, the output waveform differs depending on where the occurrence of the crest or valley interrupt is specified by the interrupt culling setting. The following figure illustrates the change of the output waveform when interrupts are culled.

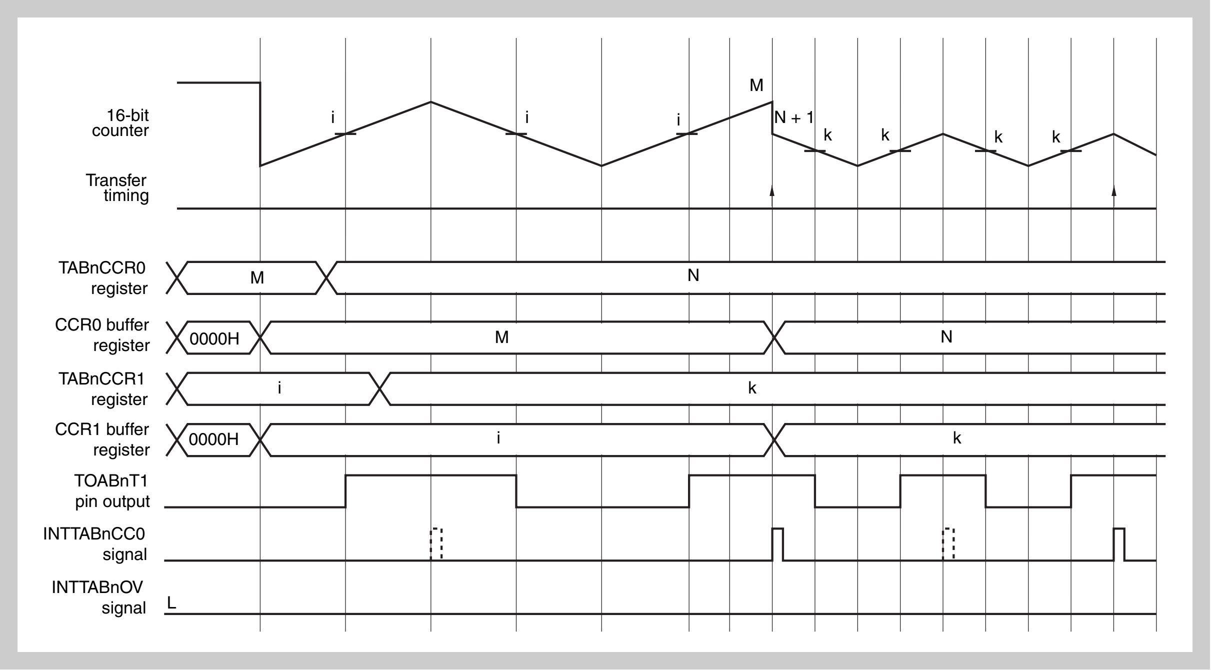


Figure 22-43 Rewriting TABnCCR0 Register (When Crest Interrupt Is Set)

The transfer timing is generated when the crest interrupt occurs, the period of up counting and down counting changes, and an asymmetrical triangular wave is output.

Note

1. This is an example of the operation when the TABnOPT1.TABnICE bit = 1, TABnOPT1.TABnIOE bit = 0, TABnOPT1.TABnID4 to TABnOPT1.TABnID0 bits = 00001.
2. ⬚: Culled interrupt

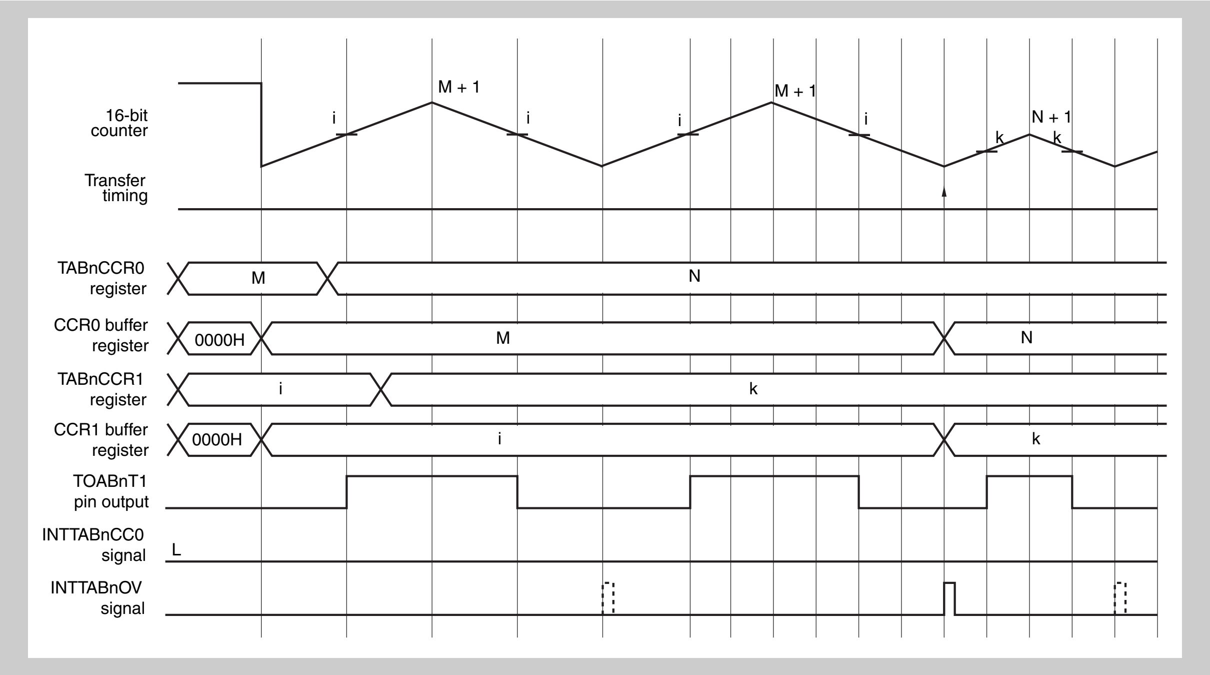


Figure 22-44 Rewriting TABnCCR0 Register (When Valley Interrupt Is Set)

The transfer timing is generated when the valley interrupt occurs, the cycle of up counting and down counting becomes identical, and a symmetrical triangular wave is output.

Note

1. This is an example of the operation when the TABnOPT1.TABnICE bit = 0, TABnOPT1.TABnIOE bit = 1, TABnOPT1.TABnID4 to TABnOPT1.TABnID0 bits = 00001.
2. ⊓: Culled interrupt

(c) Rewriting TABnCCR1 to TABnCCR3 registers

- Transfer at crest when crest interrupt is set
  Because the register is transferred at the transfer timing of the crest interrupt, an asymmetrical triangular wave is output.

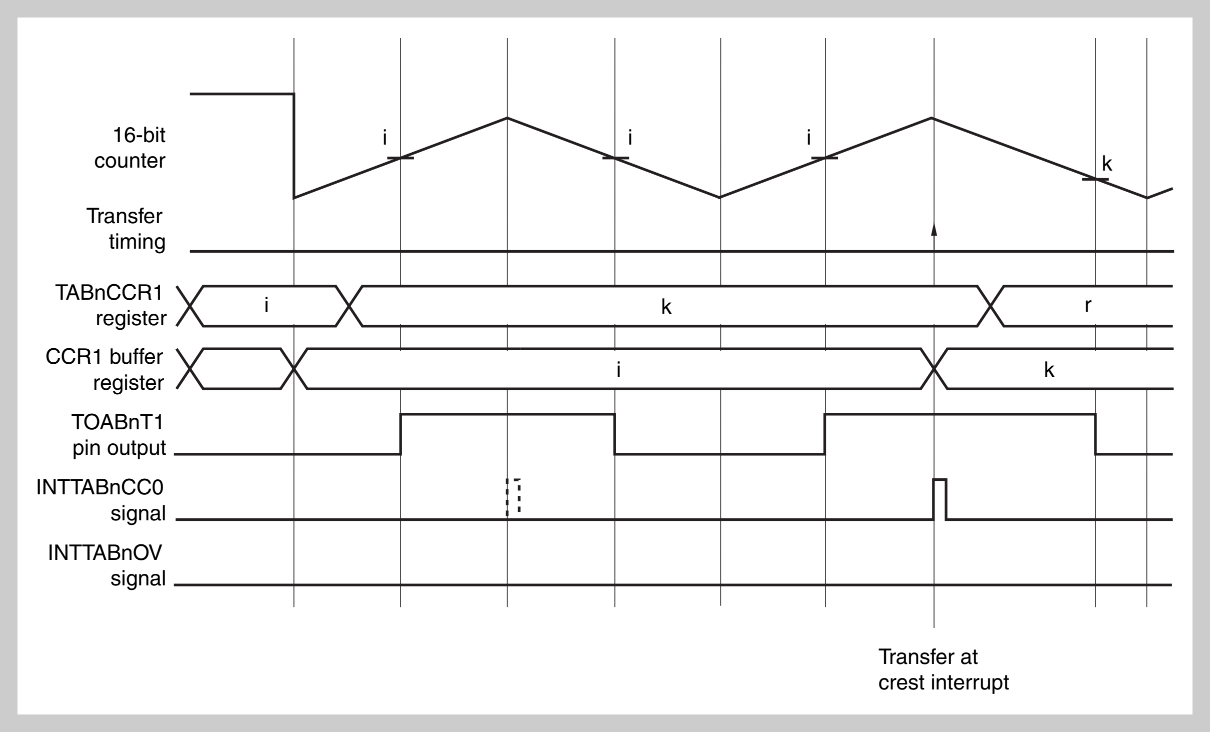


Figure 22-45 Rewriting TABnCCR1 Register (TABnOPT1.TABnICE bit = 1, TABnOPT1.TABnIOE bit = 0, TABnOPT1.TABnID4 to TABnOPT1.TABnID0 = 00001)

Note 1. ┆: Culled interrupt

- Transfer at valley when valley interrupt is set
  Because the register is transferred at the transfer timing of the valley interrupt, a symmetrical triangular wave is output.

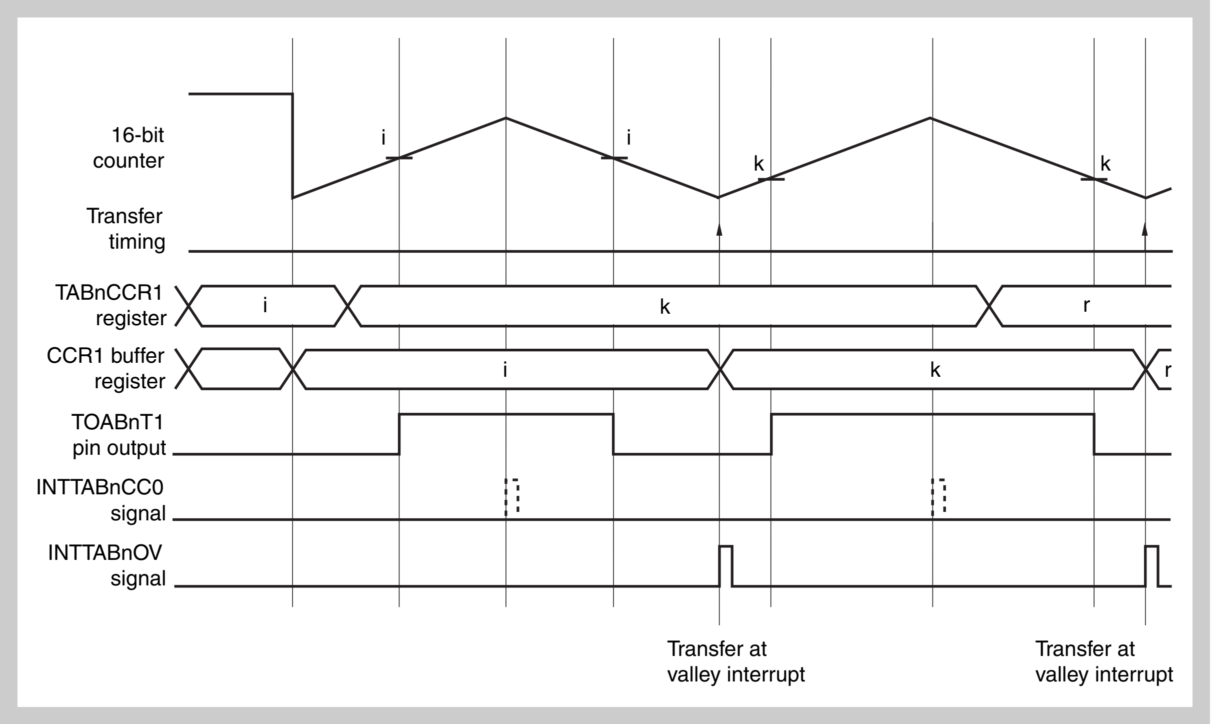


Figure 22-46 Rewriting TABnCCR1 Register (TABnOPT1.TABnICE bit = 1, TABnOPT1.TABnIOE bit = 1, TABnOPT1.TABnID4 to TABnOPT1.TABnID0 = 00001)

Note 1. ⁞: Culled interrupt

(d) Rewriting TABnOPT1 register

Because a new interrupt culling value is transferred when the value of the interrupt culling counter matches the value of the 16-bit counter, the next interrupt and those that follow occur at the set interval.
For details of rewriting the TABnOPT1 register, see *"Interrupt culling function" on page 885*.

(4) Rewriting TABnOPT0.TABnCMS bit

The TABnCMS bit can select the anytime rewrite mode and batch rewrite mode. This bit can be rewritten during timer operation (when TABnCTL0.TABnCE bit = 1). However, the operation and caution illustrated in *Figure 22-47* are necessary.
If the TABnCCR1 register is written when the TABnCMS bit is cleared to 0, a transfer request signal (internal signal) is set.
When the transfer request signal is set, the register is transferred at the next transfer timing, and the transfer request signal is cleared. This transfer request signal is also cleared when the TABnCMS bit is set to 1.

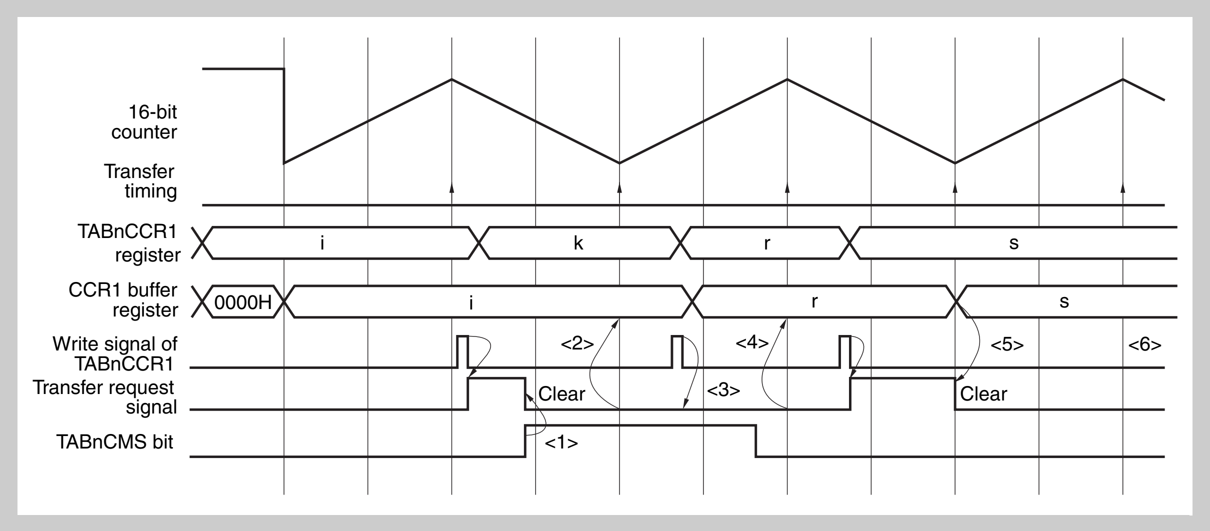


Figure 22-47 Rewriting TABnCMS Bit

<1> If the TABnCCR1 register is rewritten when the TABnCMS bit is 0, the transfer request signal is set.
If the TABnCMS bit is set to 1 in this status, the transfer request signal is cleared.

<2> The register is not transferred because the TABnCMS bit is set to 1 and the transfer request signal is cleared.

<3> The transfer request signal is not set even if the TABnCCR1 register is written when the TABnCMS bit is 1.

<4> The transfer request signal is not set even if the TABnCCR1 register is written when the TABnCMS bit is 1, so even if the TABnCMS bit is cleared to 0, transfer does not occur at the subsequent transfer timing.

<5> The transfer request signal is set if the TABnCCR1 register is written when the TABnCMS bit is 0.
Transfer is performed at the subsequent transfer timing and the transfer request signal is cleared.

<6> Once transfer has been performed, the transfer request signal is cleared.
Therefore, transfer is not performed at the next transfer timing.

22.4.5 TAAx tuning operation for A/D conversion start trigger signal output

This section explains the tuning operation of TAAx and TABn in the 6-phase PWM output mode.

In the 6-phase PWM output mode, the tuning operation is performed with TABn serving as the master and TAAx as a slave. The conversion start trigger signal of the A/D Converter can be set as the A/D conversion start trigger source by the INTTAAxCC0 and INTTAAxCC1 signals of TAAx and the INTTABnOV and INTTABnCC0 signals of TABn.

(1) Tuning operation starting procedure

The TAAx and TABn registers should be set using the following procedure to perform the tuning operation.

(a) Setting of TAAx register (stop the operations of TABn and TAAx (by clearing the TABnCTL0.TABnCE bit and TAAxCTL0.TAAxCE bit to 0)).

- Set the TAAxCTL1 register to 85H (set the tuning operation slave mode and free-running timer mode).
- Set an appropriate value to the TAAxCCR0 and TAAxCCR1 registers (set the default value for comparison for starting the operation).

(b) Setting of TABn register

- Set the TABnCTL1 register to 07H (master mode and 6-phase PWM output mode).
- Set an appropriate value to the TABnIOC0 register (set the output mode of TOABnT1 to TOABnT3).
  However, clear the TABnOL0 bit to 0 and set the TABnOE0 bit to 1 (enable positive phase output). Unless this setting is made, the crest interrupt (INTTABnCC0) and valley interrupt (INTTABnOV) do not occur. Consequently, the conversion start trigger signal of the A/D Converter is not correctly generated.
- Clear the TABnOPT0 register to 00H (select the compare register).
- Set an appropriate value to the TABnCCR0 to TABnCCR3 registers (set the default value for comparison for starting the operation).
- Set the TABnCTL0 register to 0x$_H$ (clear the TABnCE bit to 0 and set the operating clock of TABn).
  The operating clock of TAB0, set by the TAB0CTL0 register, is supplied to TAAx and count operation is started at the same timing. Operating clock of TAAx, set by the TAAxCTL0 register, is ignored.

(c) Setting of TABOPn (TABn option) register

- Set an appropriate value to the TABnOPT1 and TABnOPT2 registers.
- Set an appropriate value to the TABnIOC3 register (set TOABnB1 to TOABnB3 in the output mode).
- Set an appropriate value to the TABnDTC register (set the default value for comparison for starting the operation).

(d) Setting of alternate function

- Select the alternate function of the port by setting the port to the port control mode.

(e) Set the TAAxCE bit to 1 and set the TABnCE bit to 1 immediately after that to start the 6-phase PWM output operation.

Rewriting the TABnCTL0, TABnCTL1, TAB0IOC1, TAB0IOC2, TAAxCTL0, TAAxCTL1, and TAAxIOC0, TAAxIOC1, TAAxIOC2 registers is prohibited during operation. The operation and the PWM output waveform are not guaranteed if any of these registers is rewritten during operation. However, rewriting the TABnCTL0.TABnCE bit to clear it is permitted. Manipulating (reading/writing) the other TABn and TABn option registers is prohibited until the TAAxCTL0.TAAxCE bit is set to 1 and then the TABnCE bit is set to 1.

Caution When tuning TAAx in the 6-phase PWM output mode, output of the TOAAx0 and TOAAx1 pins is disabled.
Clear the TAAxIOC0.TAAxOE0 and TAAxIOC0.TAAxOE1 bits to 0.

(2) Tuning operation clearing procedure

To clear the tuning operation and exit the 6-phase PWM output mode, set the TAAx and TABn registers using the following procedure.

- Clear the TABnCTL0.TABnCE bit to 0 and stop the timer operation.
- Clear the TAAxCTL0.TAAxCE bit to 0 so that TAAx can be separated.
- Stop the timer output by using the TABnIOC0 and TAAxIOC0 registers.
- Clear the TAAxCTL1.TAAxSYE bit to 0 to clear the tuning operation.

Caution Manipulating (reading/writing) the other TABn, TAAx, and TABn option registers is prohibited until the TABnCE bit is set to 1 and then the TAAxCE bit is set to 1.

(3) When not tuning TAAx

When the match interrupt signal of TAAx is not necessary as the conversion trigger source that starts the A/D Converter, TAAx can be used independently as a separate timer without being tuned. In this case, the match interrupt signal of TAAx cannot be used as a trigger source to start A/D conversion in the 6-phase PWM output mode. Therefore, fix the TABnOPT2.TABnAT00 to TABnOPT2.TABnAT03 bits and the TABnOPT3.TABnAT10 to TABnOPT3.TABnAT13 bits to 0.

The other control bits can be used in the same manner as when TAAx is tuned. If TAAx is not tuned, the compare registers (TAAxCCR0 and TAAxCCR1) of TAAx are not affected by the settings of the TABnOPT0.TABnCMS and TABnOPT2.TABnRDE bits. For the initialization procedure when TAAx is not tuned, see (b) to (e) in *“Tuning operation starting procedure” on page 908*.
(a) is not necessary because it is a step used to set TAAx for the tuning operation.

(4) Basic operation of TAAx during tuning operation

The 16-bit counter of TAAx only counts up. The 16-bit counter is cleared by the set cycle value of the TABnCCR0 register and starts counting from 0000H again. The count value of this counter is the same as the value of the 16-bit counter of TAAx when it counts up. However, it is not the same when the 16-bit counter of TABn counts down.

- When TABn counts up (same value)
  16-bit counter of TABn: 0000H → M (up counting)
  16-bit counter of TAAx: 0000H → M (up counting)
- When TABn counts down (not same value)
  16-bit counter of TABn: M + 1 → 0001H (down counting)
  16-bit counter of TAAx: 0000H → M (up counting)

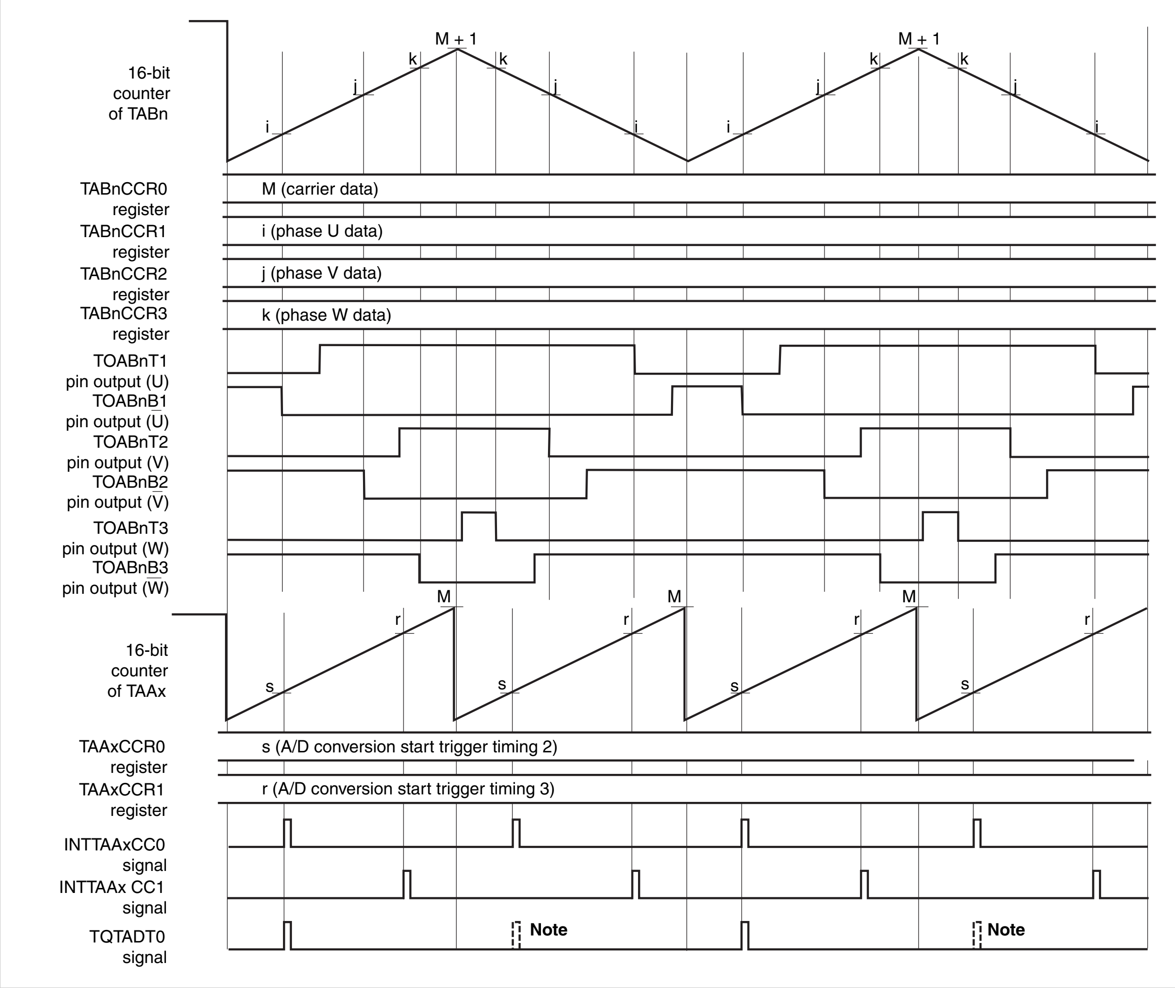


Figure 22-48 TAAx During Tuning Operation

Note The TQTADT0 signal is masked by the TABnOPT2.TABnATM02 and TABnOPT2.TABnATM03 bits.

22.4.6 A/D conversion start trigger output function

The microcontroller has a function to select four trigger sources (INTTABnOV, INTTABnCC0, INTTAAxCC0, INTTAAxCC1) to generate the A/D conversion start trigger signal (TQTADT0) of the A/D Converter.

The trigger sources are specified by the TABnOPT2.TABnAT0 to TABnOPT2.TABnAT3.

- TABnAT0 bits = 1:
  A/D conversion start trigger signal generated when INTTABnOV (counter underflow) occurs.
- TABnAT1 bits = 1:
  A/D conversion start trigger signal generated when INTTABnCC0 (cycle match) occurs.
- TABnAT2 bits = 1:
  A/D conversion start trigger signal generated when INTTAAxCC0 (match of TAAxCCR0 register of TAAx during tuning operation) occurs.
- TABnAT3 bits = 1:
  A/D conversion start trigger signal generated when INTTAAxCC1 (match of TAAxCCR1 register of TAAx during tuning operation) occurs.

The A/D conversion start trigger signals selected by the TABnAT0 to TABnAT3 bits are OR'ed and output. Therefore, two or more trigger sources can be specified at the same time.

The INTTABnOV and INTTABnCC0 signals selected by the TABnAT0, TABnAT1 bits are culled interrupt signals.

Therefore, these signals are output after the interrupts have been culled and, unless interrupt output is enabled (TABnOPT1.TABnICE, TABnOPT1.TABnIOE bits), the A/D conversion start trigger is not output.

The trigger sources (INTTAAxCC0 and INTTAAxCC1) from TAAx have a function to mask the A/D conversion start trigger signal depending on the status of the up-count/down-count of the 16-bit counter, if so set by the TABnAT2 and TABnAT3 bits.

- TABnATM2 bits:
  Correspond to the TABnAT2 bit and control INTTAAxCC0 (match interrupt signal) of TAAx.
  - TABnATM2 bits = 0
    The A/D conversion start trigger signal is output when the 16-bit counter counts up (TABnOPT0.TABnCUF bit = 0), and the A/D conversion start trigger signal is not output when the 16-bit counter counts down (TABnOPT0.TABnCUF bit = 1).
  - TABnATM2 bits = 1
    The A/D conversion start trigger signal is output when the 16-bit counter counts down (TABnOPT0.TABnCUF bit = 1), and the A/D conversion start trigger signal is not output when the 16-bit counter counts up (TABnOPT0.TABnCUF bit = 0).

- TABnATM3 bits:
  Correspond to the TABnAT3 bit and control INTTAAxCC1 (match interrupt signal) of TAAx.
  - TABnATM3 bits = 0
    The A/D conversion start trigger signal is output when the 16-bit counter counts up (TABnOPT0.TABnCUF bit = 0), and the A/D conversion start trigger signal is not output when the 16-bit counter counts down (TABnOPT0.TABnCUF bit = 1).
  - TABnATM3 bits = 1
    The A/D conversion start trigger signal is output when the 16-bit counter counts down (TABnOPT0.TABnCUF bit = 1), and the A/D conversion start trigger signal is not output when the 16-bit counter counts up (TABnOPT0.TABnCUF bit = 0).

The TABnATM3, TABnATM2, TABnAT3 to TABnAT0 bits can be rewritten while the timer is operating. If the bit that sets the A/D conversion start trigger signal is rewritten while the timer is operating, the new setting is immediately reflected on the output status of the A/D conversion start trigger. These control bits do not have a transfer function and can be used only in the anytime rewriting mode.

Caution

1. The A/D conversion start trigger signal output that is set by the TABnAT2, TABnAT3 bits can be used only when TAAx is performing a tuning operation as the slave timer of TABn. If TABn and TAAx are not performing a tuning operation, or if a mode other than the 6-phase PWM output mode is used, the output cannot be guaranteed.
2. The TABn0 signal output is internally used to identify whether the 16-bit counter is counting up or down. Therefore, enable TOABn0 pin output by clearing the TABnIOC0.TABnOL0 bit to 0 and setting the TABnOE0 bit to 1.

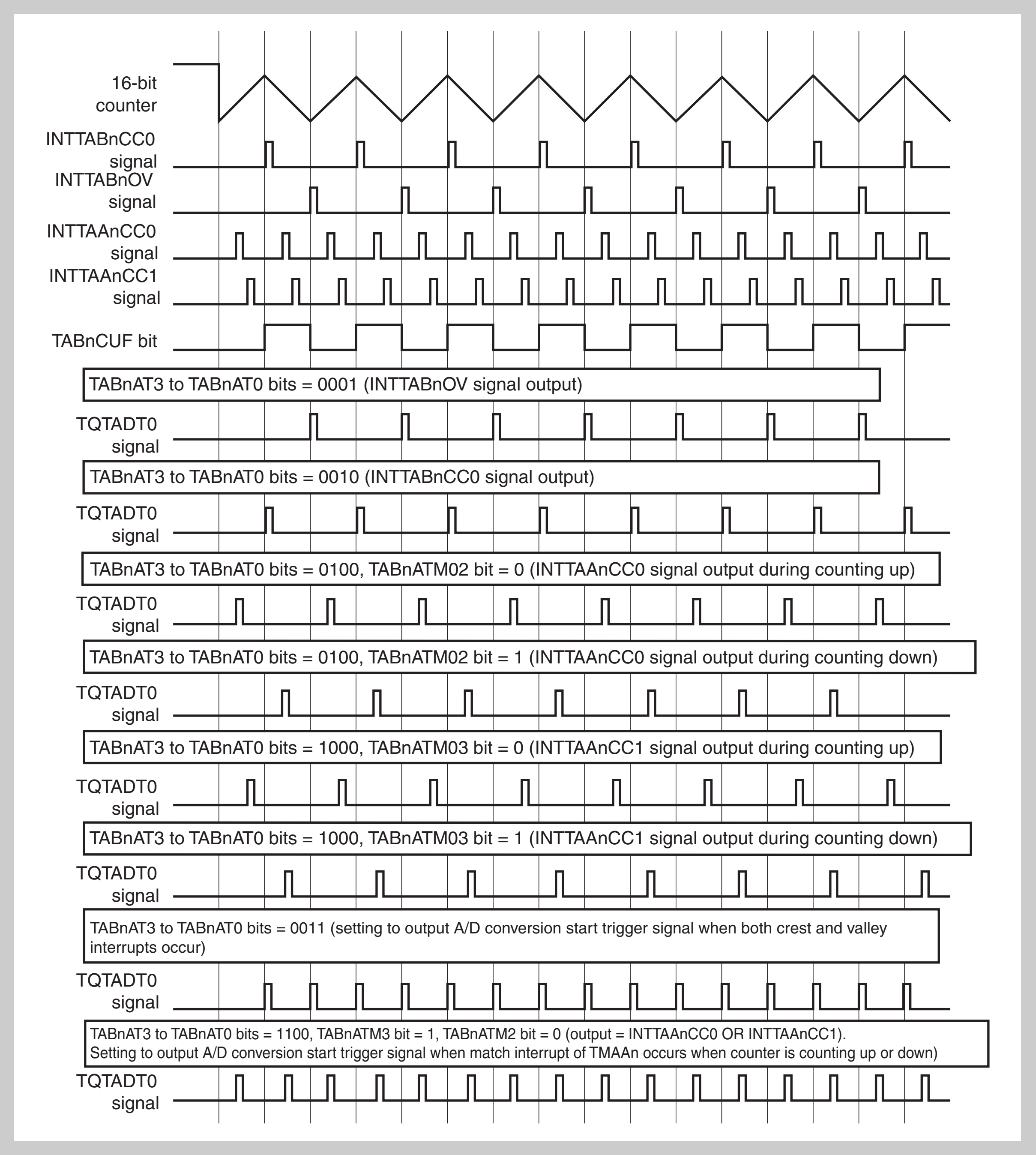

16-bit counter
INTTABnCC0 signal
INTTABnOV signal
INTTAAnCC0 signal
INTTAAnCC1 signal
TABnCUF bit
TABnAT3 to TABnAT0 bits = 0001 (INTTABnOV signal output)
TQTADT0 signal
TABnAT3 to TABnAT0 bits = 0010 (INTTABnCC0 signal output)
TQTADT0 signal
TABnAT3 to TABnAT0 bits = 0100, TABnATM02 bit = 0 (INTTAAnCC0 signal output during counting up)
TQTADT0 signal
TABnAT3 to TABnAT0 bits = 0100, TABnATM02 bit = 1 (INTTAAnCC0 signal output during counting down)
TQTADT0 signal
TABnAT3 to TABnAT0 bits = 1000, TABnATM03 bit = 0 (INTTAAnCC1 signal output during counting up)
TQTADT0 signal
TABnAT3 to TABnAT0 bits = 1000, TABnATM03 bit = 1 (INTTAAnCC1 signal output during counting down)
TQTADT0 signal
TABnAT3 to TABnAT0 bits = 0011 (setting to output A/D conversion start trigger signal when both crest and valley interrupts occur)
TQTADT0 signal
TABnAT3 to TABnAT0 bits = 1100, TABnATM3 bit = 1, TABnATM2 bit = 0 (output = INTTAAnCC0 OR INTTAAnCC1).
Setting to output A/D conversion start trigger signal when match interrupt of TMAAn occurs when counter is counting up or down)
TQTADT0 signal

Figure 22-49 Example of A/D Conversion Start Trigger (TQTADT0) Signal Output (TABnOPT1.TABnICE Bit = 1, TABnOPT1.TABnIOE Bit = 1, TABnOPT1.TABnID4 to TABnOPT1.TABnID0 Bits = 00000: Without Interrupt Culling)

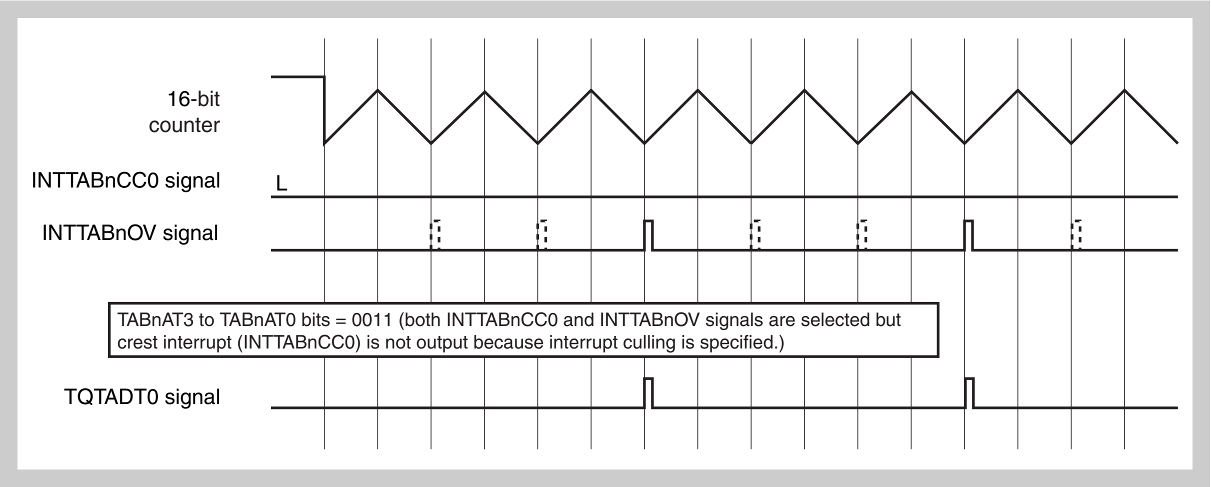

16-bit
counter
INTTABnCC0 signal
L
INTTABnOV signal
TABnAT3 to TABnAT0 bits = 0011 (both INTTABnCC0 and INTTABnOV signals are selected but crest interrupt (INTTABnCC0) is not output because interrupt culling is specified.)
TQTADT0 signal

Figure 22-50 Example of A/D Conversion Start Trigger (TQTADT0) Signal Output (TABnOPT1.TABnICE Bit = 0, TABnOPT1.TABnIOE Bit = 1, TABnOPT1.TABnID4 to TABnOPT1.TABnID0 Bits = 00010: With Interrupt Culling) (1)

Note ⸽: Culled interrupt

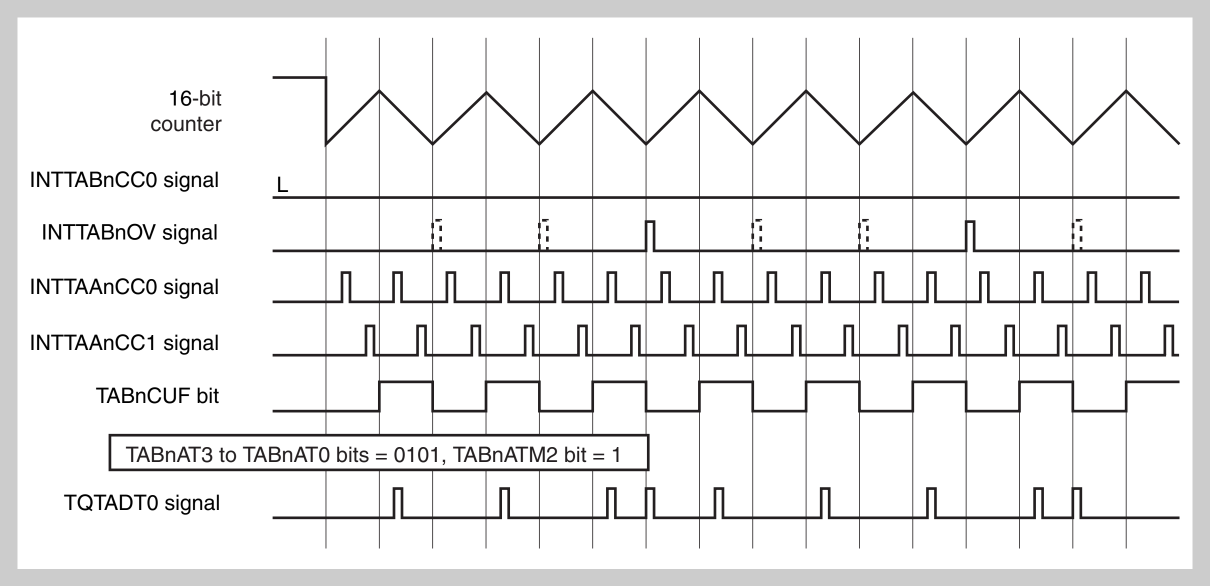


Figure 22-51 Example of A/D Conversion Start Trigger (TQTADT0) Signal Output (TABnOPT1.TABnICE Bit = 0, TABnOPT1.TABnIOE Bit = 1, TABnOPT1.TABnID4 to TABnOPT1.TABnID0 Bits = 00010: With Interrupt Culling) (2)

Note ⸽: Culled interrupt

Caution The INTTABnCC0 signal is culled but the INTTAAxCC0 signal is not.

(1) Operation under boundary condition (operation when 16-bit counter matches INTTAAxCC0 signal)

Table 22-9 Operation When TABnCCR0 Register = M, TABnATm2 Bit = 1, TABnATMm2 Bit = 0 (Up Counting Period Selected)

Value of TAAxCCR0 Register	Value of 16-bit Counter of TABn	Value of 16-bit Counter of TAAx	Status of 16-bit Counter of TABn	Output of INTTAAxCC0 Signal from TQTADT0 Signal
0000H	0000H	0000H	-	Output
0000H	M + 1	0000H	-	Not output
0001H	0001H	0001H	Up count	Output
0001H	M	0001H	Down count	Not output
M	M	M	Up count	Output
M	0001H	M	Down count	Not output

Table 22-10 Operation When TABnCCR0 Register = M, TABnATm2 Bit = 1, TABnATMm2 Bit = 1 (Down Counting Period Selected)

Value of TAAxCCR0 Register	Value of 16-bit Counter of TABn	Value of 16-bit Counter of TAAx	Status of 16-bit Counter of TABn	Output of INTTAAxCC0 Signal from TQTADT0 Signal
0000H	0000H	0000H	-	Not output
0000H	M + 1	0000H	-	Output
0001H	0001H	0001H	Up count	Not output
0001H	M	0001H	Down count	Output
M	M	M	Up count	Not output
M	0001H	M	Down count	Output

Caution The TAAxCCRm register enables setting of “0” to “M” when the TABnCCR0 register = M. Setting of a value of “M + 1” or higher is prohibited.
If a value higher than “M + 1” is set, the 16-bit counter of TAAx is cleared by “M”. Therefore, the TQTADT0 signal is not output.

Chapter 23 Power Supply Scheme

The microcontroller has general power supply pins for its core, internal memory and peripherals. These pins are connected to internal voltage regulators. The microcontroller also has dedicated power supply pins for certain I/O modules. These pins provide the power for the I/O operations.

23.1 Overview

The following table gives the naming convention of the pins:

Table 23-1 Naming convention of power supply pins

Dedicated function		V_{DD} or V_{SS}
<none>	CPU core, internal memory and peripherals	• VDD: Voltage Drain Drain • VSS: Voltage for Substrate and Source
A	A/D Converter, Low-Voltage Detector	
B, E	Standard I/O buffer	

The following pins belong to the Power Supply Scheme:

Note For electrical characteristics refer to the Datasheet.

Table 23-2 Power supply pins

Power supply pins	V850ES/FE3 V850ES/FF3	V850ES/FG3	V850ES/FJ3	V850ES/FK3
AVREF0 / AVSS	A/D Converter 0 / Low-Voltage Detector			
AVREF1 / AVSS1	–			A/D Converter 1
VDD / VSS	CPU core[a]			
EVDD / EVSS	• numbered I/O port buffers[b] • alphabet I/O port buffers[c]	numbered I/O port buffers[b]		
VDD1 / VSS1	–			CPU core[a] (with voltage regulator)
BVDD / BVSS	–			alphabet I/O port buffers[c]
BVDD/VDD1 BVSS/VSS1	–	• CPU core[a] • alphabet I/O port buffers[c]		–

[a] With voltage regulator, built-in memory, internal logic circuit and the oscillator block.
[b] numbered I/O ports: port groups 0 to 15
[c] alphabet I/O ports: port groups CD, CM, CS, CT, DL

23.2 Description

Following figures give an overview of the allocation of power supply pins on the chip.

Note The diagrams do not show the exact pin location.

(1) V850ES/FE3, V850ES/FF3 power supply pins assignment

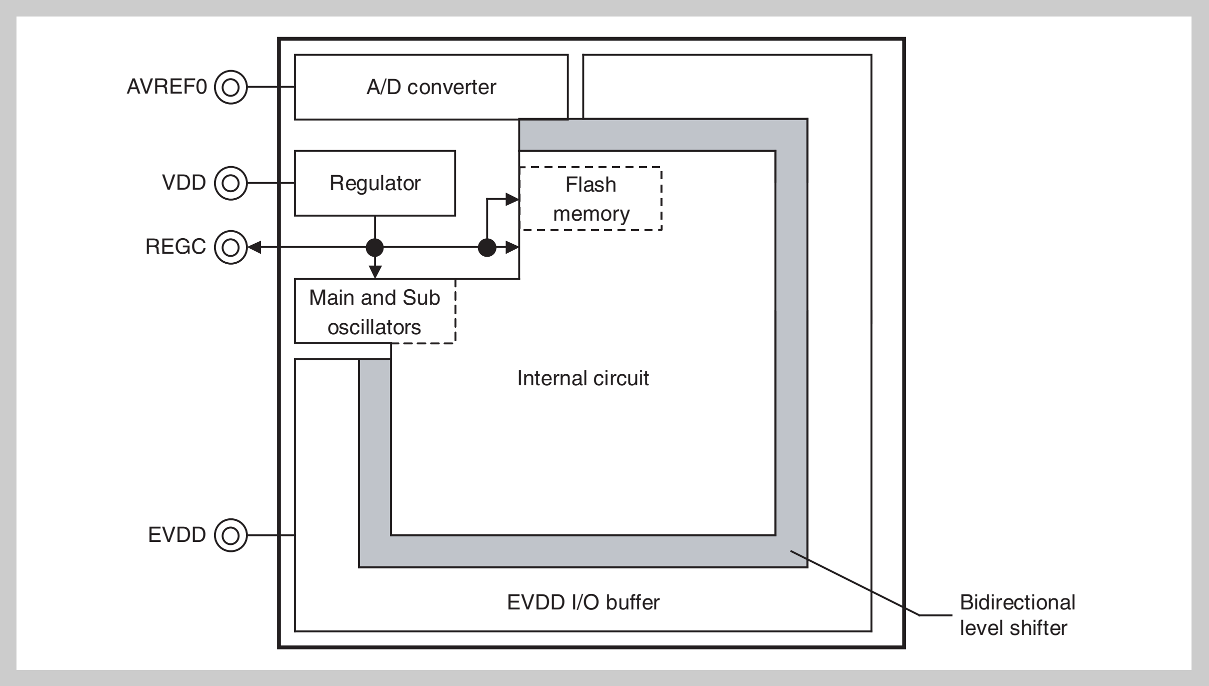


Figure 23-1 V850ES/FE3, V850ES/FF3 power supply pins assignment

(2) V850ES/FG3, V850ES/FJ3 power supply pins assignment

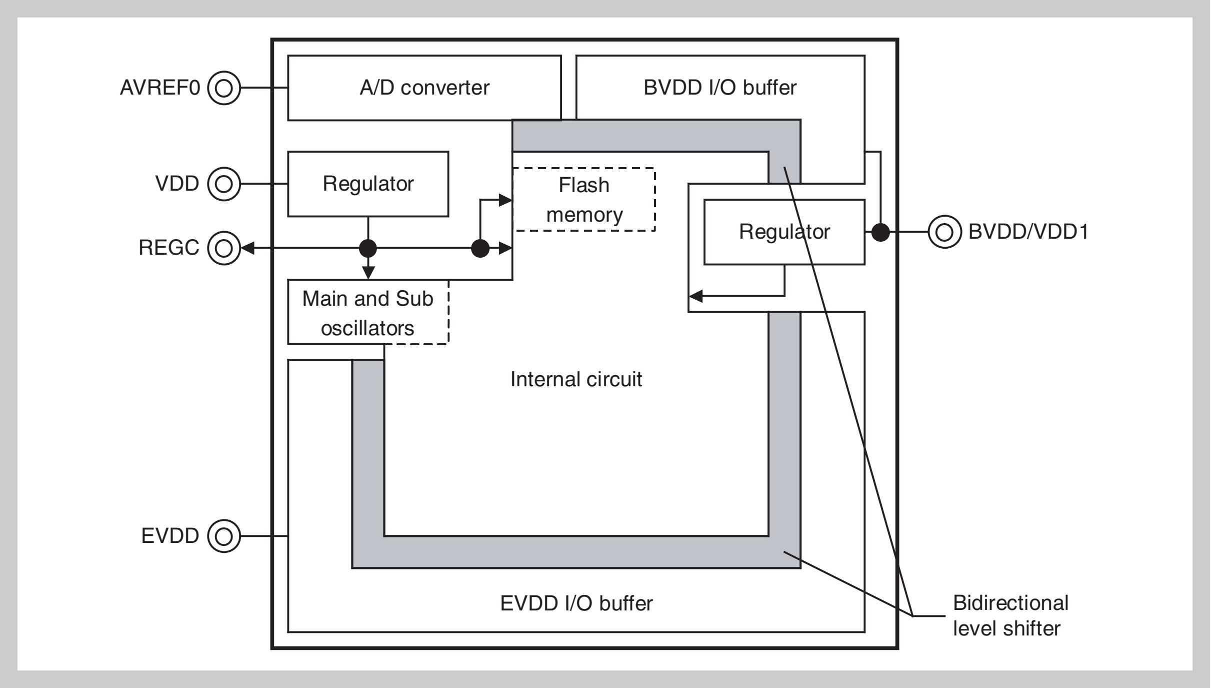


Figure 23-2 V850ES/FG3, V850ES/FJ3 power supply pins assignment

(3) V850ES/FK3 power supply pins assignment

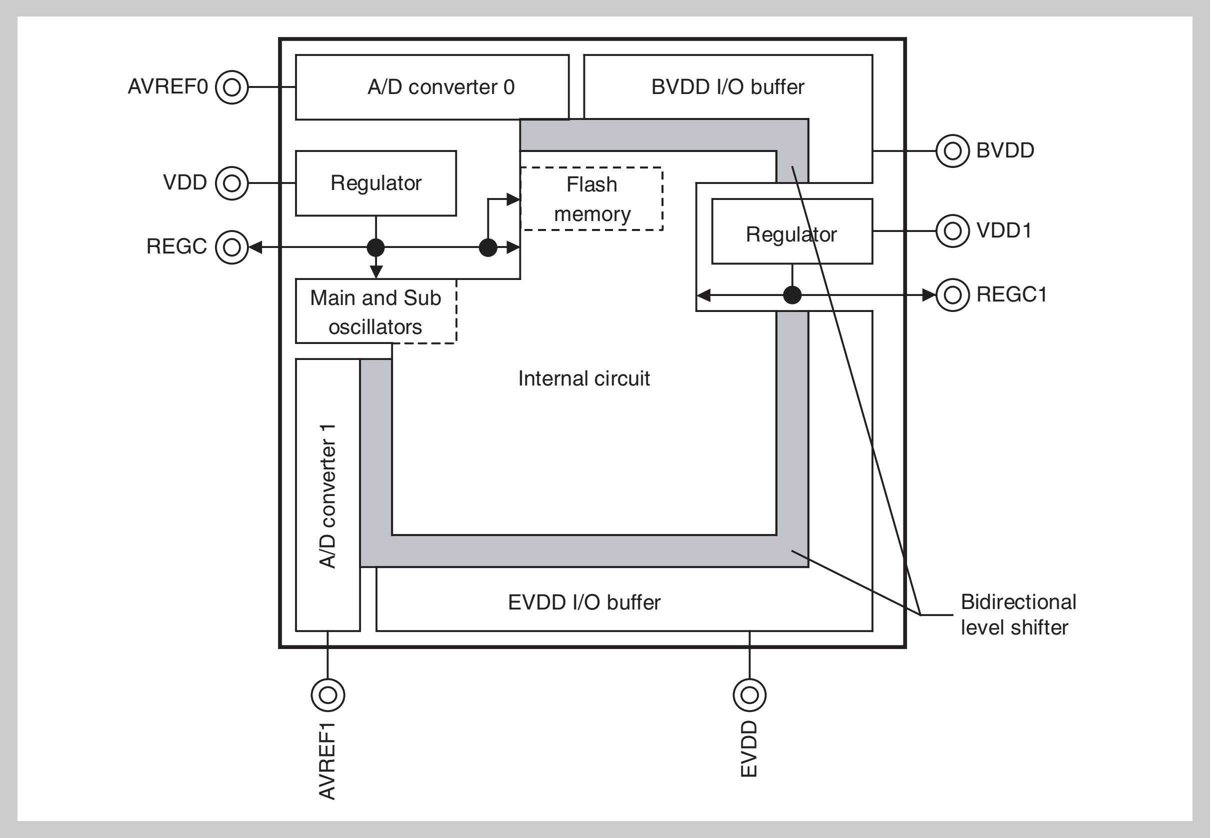


Figure 23-3 V850ES/FK3 power supply pins assignment

23.3 On-chip voltage regulators

The on-chip voltage regulators generate the voltages for the internal circuitry, refer to *Figure 23-1* and following.

The regulators operate per default in all operation modes (normal operation, HALT, IDLE1, IDLE2, STOP, Sub-clock, and during reset).

Note To stabilize the output voltage of the regulator, connect a capacitor to the REGC pin. Refer to the Datasheet.

Chapter 24 Low-Voltage Detector

This chapter describes the Low-Voltage Detector and the RAM data rentention function.

24.1 Functions

The Low-Voltage Detector (LVI) has the following functions.

- Compares the supply voltage (V_{DD}) with a reference voltage (V_{LVI}) and generates
  - internal interrupt signals when $V_{DD} < V_{LVI}$ or $V_{DD} > V_{LVI}$
  - or internal reset signal when $V_{DD} < V_{LVI}$.
- The level of the supply voltage to be detected can be changed by software (in two steps).
- Interrupt or reset signal can be selected by software.
- Can operate in STOP mode.
- Operation can be stopped by software.

If the Low-Voltage Detector 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. For details of RESF, refer to *“Reset” on page 942*.

24.2 Configuration

Figure 24-1 shows the block diagram of the Low-Voltage Detector.

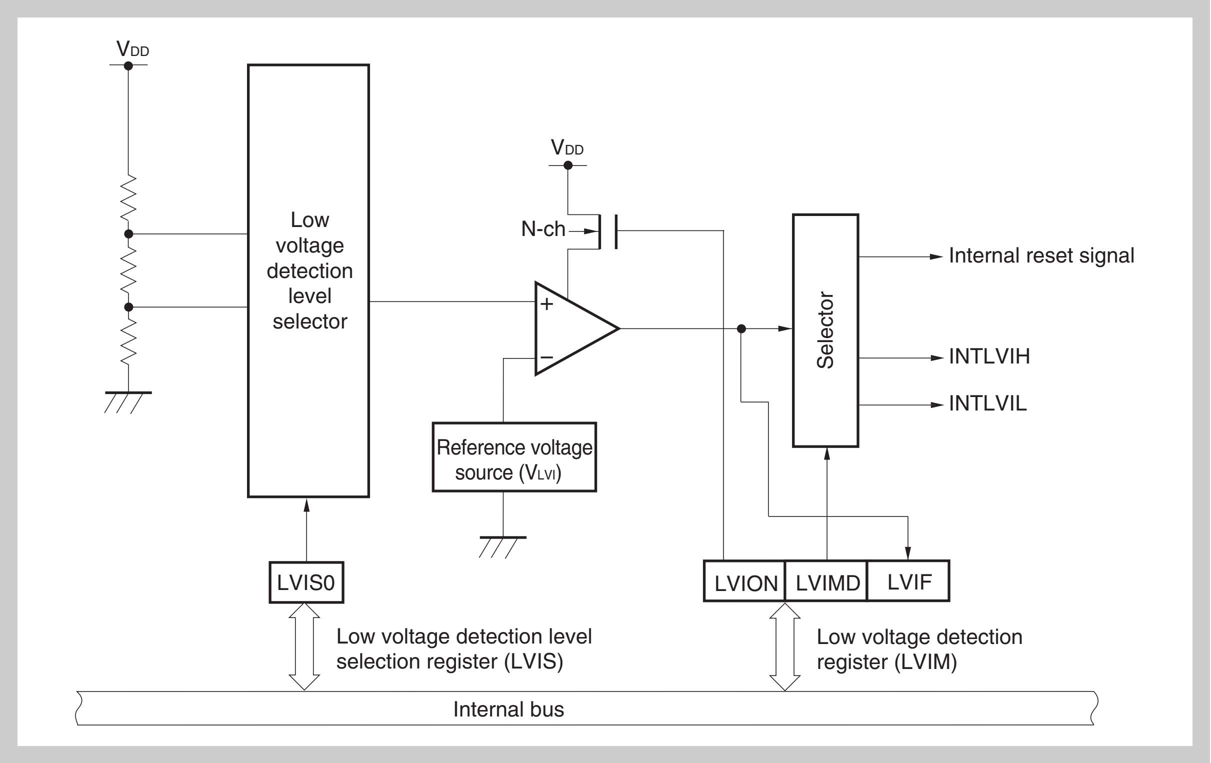

VDD
Low voltage detection level selector
VDD
N-ch
+
–
Reference voltage source (VLVI)
Selector
Internal reset signal
INTLVIH
INTLVIL
LVIS0
LVION
LVIMD
LVIF
Low voltage detection level selection register (LVIS)
Low voltage detection register (LVIM)
Internal bus

Figure 24-1 Block diagram of Low-Voltage Detector

24.3 Registers

The Low-Voltage Detector is controlled by the following registers.

- Low voltage detection register (LVIM)
- Low voltage detection level selection register (LVIS)

(1) LVIM - Low voltage detection register

This register is a special register and can be written only in a combination of specific sequences (refer to *“Write Protected Registers” on page 176*).

The LVIM register is used to enable or disable low voltage detection, and to set the operation mode of the Low-Voltage Detector.

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

Address FFFFF890_H

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

	7	6	5	4	3	2	1	0
LVIM	LVION	0	0	0	0	0	LVIMD	LVIF
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R

Caution When writing the LVIM register be sure to clear bits 2 to 6 to 0.

Table 24-1 LVIM register contents

Bit position	Bit name	Function
7	LVION	Controls low voltage detection operation. 0: Disable operation 1: Enable operation Caution: After setting the LVIM.LVION bit to 1, wait for a specified time before checking the voltage using the LVIM.LVIF bit. The wait time is specified in the Datasheet.
1	LVIMD	Specifies operation mode of low voltage detection. 0: Generate interrupt request signal - INTLVIL, when supply voltage V_{DD} < reference voltage V_{LVI}. - INTLVIH, when supply voltage V_{DD} > reference voltage V_{LVI}. 1: Generate internal reset signal LVIRES, when supply voltage V_{DD} < reference voltage V_{LVI}
0	LVIF	Low voltage detection flag 0: Cleared when - supply voltage V_{DD} > reference voltage V_{LVI} or - operation is disabled (LVIM.LVION = 0) 1: Set when supply voltage of power supply V_{DD} < reference voltageV_{LVI}. Caution: 1. The LVIIF bit is valid only when LVION = 1 and LVIMD = 0. 2. The LVIF bit is read-only.

(2) LVIS - Low voltage detection level selection register

The LVIS register is used to select the level of low voltage to be detected.

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

Address FFFFF891$_H$

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

	7	6	5	4	3	2	1	0
LVIS	0	0	0	0	0	0	0	LVIS0
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Caution

1. The LVIS register cannot be written until a reset request due to something other than low-voltage detection is generated after the LVIM.LVION and LVIM.LVIMD bits are set to 1.
2. When writing the LVIs register be sure to cleare sure to clear bits 7 to 1 to 0.

Table 24-2 LVIS register contents

Bit position	Bit name	Function
0	LVIS0	Specifies the dectection level. 0: 4.0 V[a] 1: 3.7 V[a] **Caution:** After setting the LVIM.LVION bit to 1, wait for a specified time before checking the voltage using the LVIM.LVIF bit. The wait time is specified in the Datasheet.

a) Refer to Datasheet for the detailed specification.

(3) RAMS - Internal RAM data status register

The RAMS register is a flag register that indicates that the supply voltage has dropped below a specific data retention voltage. If so, the contents of the RAM may have changed and has to be considered as invalid.

Access This register can be read/written in 8-bit or 1-bit units.
Writing to this register is protected by a special sequence of instructions.
Please refer to *“Write Protected Registers” on page 176* for details.

Address FFFFF892$_H$

Initial Value This register is not influenced by any reset. Refer to *24.4.4 on page 929* for further details concerning RAM data retention.

	7	6	5	4	3	2	1	0
RAMS	0	0	0	0	0	0	0	RAMIF
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 24-3 RAMS register contents

Bit position	Bit name	Function
0	RAMIF	Indicates valid or invalid data of internal RAM. 0: Supply voltage > Data retention voltage[a], RAM data valid. 1: Supply voltage < Data retention voltage[a], RAM data invalid.

[a] Refer to Datasheet for the detailed specification.

(4) PEMU1 - Peripheral emulation register 1

When an in-circuit emulator is used, the operation of the RAM retention flag (RAMF bit: bit 0 of RAMS register) can be pseudo-controlled and emulated by manipulating this register on the debugger.

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

This register is valid only in the emulation mode. It is invalid in the normal mode.

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

Address $FFFFF9FE_H$

Initial Value This register is not influenced by any reset. Refer to *24.4.4 on page 929* for further details concerning RAM data retention.

	7	6	5	4	3	2	1	0
PEMU1	0	0	0	0	0	EVARAMIN	0	0
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 24-4 PEMU1 register contents

Bit position	Bit name	Function
0	EVARAMIN	Pseudo specification of RAM retention voltage detection signal. 0: Do not detect voltage lower than RAM retention voltage. 1: Detect voltage lower than RAM retention voltage (set RAMF flag). **Caution:** The EVARAMIN bit is not automatically cleared.

Usage When an in-circuit emulator is used, pseudo emulation of RAMF is realized by rewriting this register on the debugger.

<1> CPU break (CPU operation stops.)

<2> Set the EVARAMIN bit to 1 by using a register write command.
By setting the EVARAMIN bit to 1, the RAMF bit is set to 1 on hardware (the internal RAM data is invalid).

<3> Clear the EVARAMIN bit to 0 by using a register write command again.
Unless this operation is performed (clearing the EVARAMIN bit to 0), the RAMF bit cannot be cleared to 0 by a CPU operation instruction.

<4> Run the CPU and resume emulation.

24.4 Operation

Depending on the setting of the LVIMD bit, the interrupt signals (INTLVIL, INTLVIH) or an internal reset signal is generated.

How to specify each operation is described below, together with timing charts.

24.4.1 Reset generation from LVI (LVIM.LVIMD = 1)

Operation start

1. Mask the interrupt of LVI.
2. Select the voltage to be detected by using the LVIS.LVIS0 bit.
3. Set the LVIM.LVION bit to 1 (to enable operation).
4. Insert sufficient wait time by software. See the Datasheet for details.
5. By using the LVIM.LVIF bit, check if the supply voltage V_{DD} > reference voltage V_{LVI}.
6. Set the LVIM.LVIMD bit to 1 (to generate an internal reset signal).

Caution If LVIM.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.

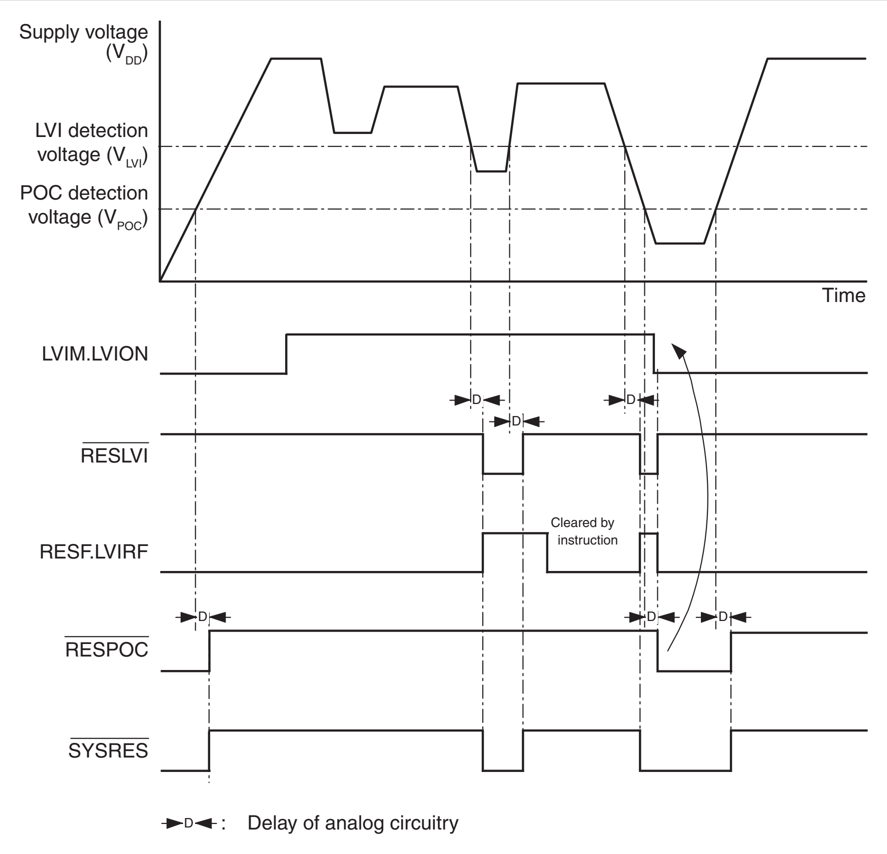

Supply voltage (V_{DD})
LVI detection voltage (V_{LVI})
POC detection voltage (V_{POC})
Time
LVIM.LVION
D
RESLVI
Cleared by instruction
RESF.LVIRF
RESPOC
SYSRES
D : Delay of analog circuitry

Figure 24-2 Operation timing of Low-Voltage Detector (LVIMD = 1)

Note During the period in which the supply voltage is the set low voltage or lower, the internal reset signal is retained (internal reset state).

24.4.2 Interrupt generation from LVI (LVIM.LVIMD = 0)

Operation start

1. Mask the interrupts of LVI.
2. Select the voltage to be detected by using the LVIS.LVIS0 bit.
3. Set the LVIM.LVION bit to 1 (to enable operation).
4. Insert sufficient wait time by software. See the Datasheet for details.
5. By using the LVIM.LVIF bit, check if the supply voltage V_{DD} > reference voltage V_{LVI}.
6. Clear the interrupt request flag of LVI.
7. Unmask the interrupt of LVI.

Operation stop

1. Mask the interrupt INTLVIH by setting LVIHMK to 1.
2. Clear the LVIM.LVION bit to 0.
3. Clear the interrupt request flag LVIHIF of INTLVIH

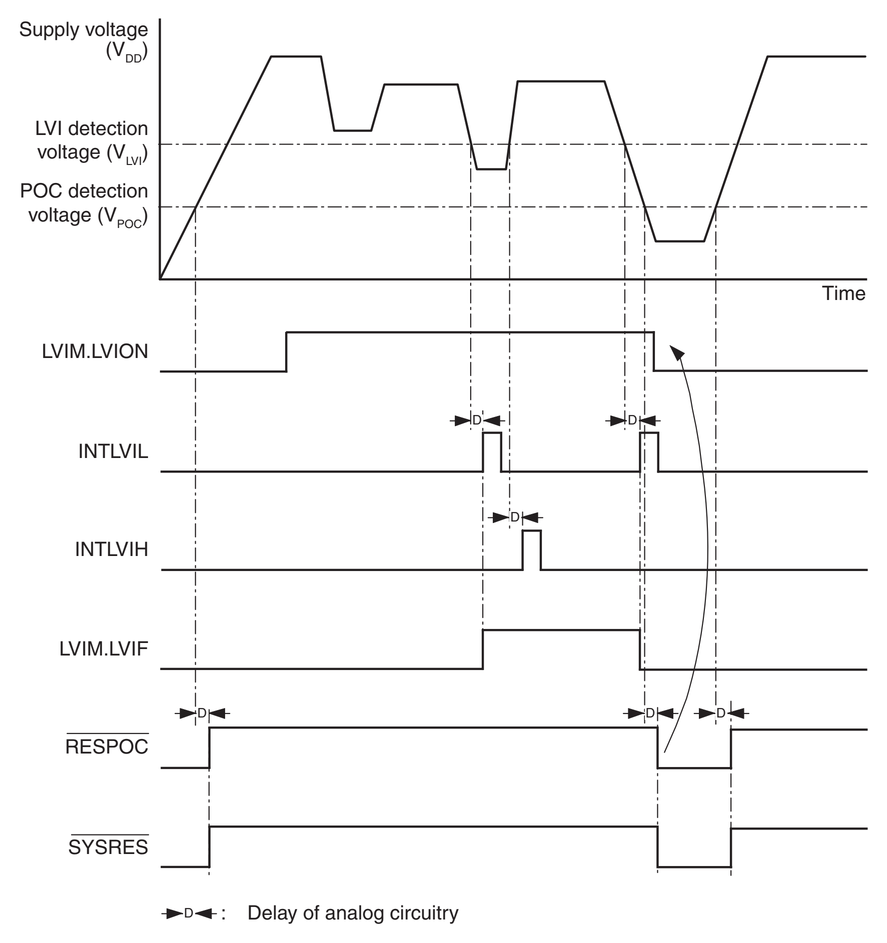

Supply voltage (V_{DD})
LVI detection voltage (V_{LVI})
POC detection voltage (V_{POC})
Time
LVIM.LVION
INTLVIL
INTLVIH
LVIM.LVIF
RESPOC
SYSRES
D
D : Delay of analog circuitry

Figure 24-3 Operation timing of Low-Voltage Detector (LVIM.LVIMD = 0)

Note If VDD is fluctuating around the LVI detection level (VLVI), note that the judgment upon the INTLVIH or INTLVIL interrupt servicing may be incorrect. For example, if during INTLVIL interrupt servicing multiple INTLVIH/INTLVIL interrupts are generated due to the VDD fluctuation, it cannot be detected which interrupt was generated last. Consequently, when INTLVIL interrupt

servicing is performed at the last, even though VDD > VLVI, software detects VDD < VLVI by mistake.
Therefore when LVI detection interrupt servicing is performed, program the software code as to complete interrupt servicing before the next LVI detection is generated, at the same time as controlling the VDD, or monitoring the LVIF flag.

24.4.3 Disabling the LVI operation

1. Mask the interrupt INTLVIH by setting LVIHMK to 1.
2. Disable the LVI operation by setting the LVIM. LVON bit to 0.
3. Clear the interrupt request flag LVIHIF of the INTLVIH register.

24.4.4 RAM retention voltage detection operation

The supply voltage and the data retention voltage are compared. When the supply voltage drops below the data retention voltage (including power on application), the RAMS.RAMF bit is set.

For the specification of the data retention voltage, consult the Datasheet.

The RAMS.RAMF flag behaves as follows:

- After power up the RAMS.RAMF is set.
- RAMS.RAMF can only be reset by software.
- RAMS.RAMF remains 0 as long as the supply voltage exceeds the data retention voltage.
- The RAMS.RAMF flag is not influenced by any reset.
- If the supply voltage drops below the power-on-clear reference voltage, but stays above the data retention voltage, a POC reset is applied, but RAMS.RAMF remains 0.

Caution If an external $\overline{\text{RESET}}$ is applied during a RAM access of the CPU, parts of the RAM content may have changed accidentally. Such event does not set RAMS.RAMF.

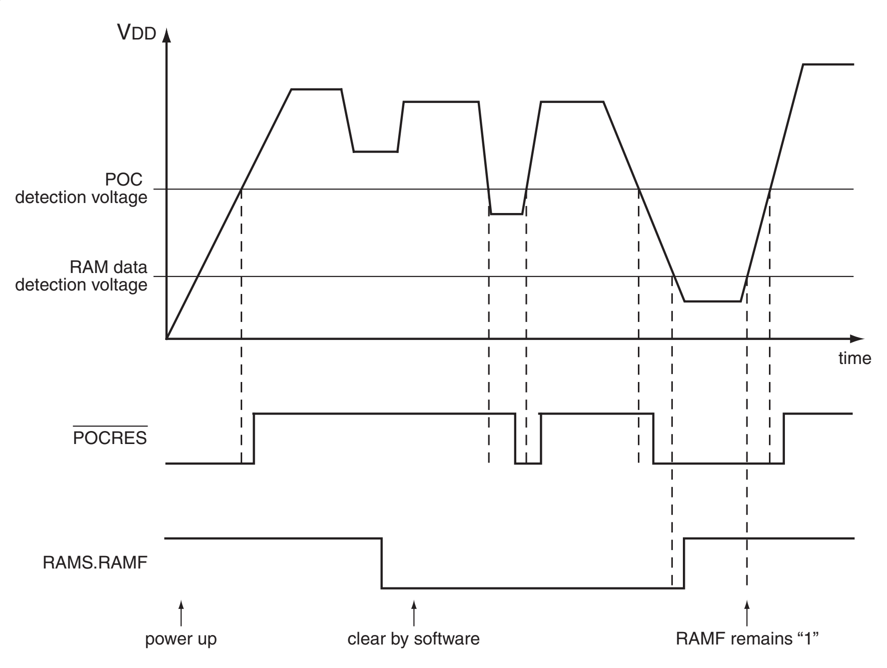


Figure 24-4 Power-on-clear and RAM data retention detection behaviour

Chapter 25 On-Chip Debug Unit

The microcontroller includes an on-chip debug unit. By connecting an N-Wire emulator, on-chip debugging can be executed.

25.1 Functional Outline

25.1.1 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 four software breakpoints can be set in the internal code 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 between the maskable signals and on-chip debug emulator mask functions are shown below.

- NMI0 mask function: NMI pin
- NMI1 mask function: WDT2 interrupt
- HOLD mask function: HLDRQ pin
- RESET mask function: RESET pin, WDT2 reset, POC reset[Note], LVI reset, clock monitor reset
- WAIT mask function: WAIT pin

Note Available in products with the POC function

(10) Timer function

The execution time of the user program can be measured.

(11) 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 2
  - Clock Monitor
- Functions that can operate or be stopped during break (however, each function cannot be selected individually)
  - all timers AB
  - all timers AA
  - Timer M
  - Watch Timer
  - Motor Control
- Peripheral functions that continue operating during break (functions that cannot be stopped)
  - Peripheral functions other than above

(12) Function during power saving modes

When the device is set into a power saving mode, debug operation is not possible. When exiting the power save mode, the on-chip debug unit continues operation.

The on-chip debug emulator interface is still accessible during power saving modes:

- The on-chip debug emulator can get status information from the on-chip debug unit.
- Stop mode can be released by the on-chip debug emulator.

(13) Security function

This microcontroller has a N-Wire security function, that demands the user to input an ID code upon start of the debugger.

For further information concerning N-Wire security, refer to *“Data Protection and Security” on page 334*.

25.2 Controlling the N-Wire Interface

The N-Wire interface pins $\overline{\text{DRST}}$, DDI, DDO, DCK, DMS are shared with port functions, see Table 25-1. During debugging the respective device pins are forced into the N-Wire interface mode and port functions are not available. Note that N-Wire debugging must be generally permitted by the security bit in the ID code region (*0x0000 0079[bit7] = 1) of the code flash memory.

An internal pull-down resistor - detachable by software - is provided at the $\overline{\text{DRST}}$ pin to keep the N-Wire interface in reset, if no debugger is connected.

Table 25-1 N-Wire interface pins

GPIO	N-Wire function		
	Pin	**Direction**	**Description**
P05	$\overline{\text{DRST}}$	Input	N-Wire RCU reset
P52	DDI	Input	N-Wire debug data in
P53	DDO	Output	N-Wire debug data out
P54	DCK	Input	N-Wire interface clock
P55	DMS	Input	N-Wire mode

(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 *“CPU System Functions” on page 155* for details.

Access The register can be read or written in 8-bit and 1-bit units.

Address FFFF F9FC_H

Initial value $00_H/01_H$. The initial value depends on the reset source (see below).

	7	6	5	4	3	2	1	0
OCDM								OCDM0
	R	R	R	R	R	R	R	R/W

Table 25-2 OCDM register contents

Bit position	Bit name	Function
0	OCDM0	On-chip debug mode 0: On-chip debug mode disabled: - Pins[a] used as port/alternative function pins. - Internal pull-down resistor detached from P05/$\overline{\text{DRST}}$. 1: On-chip debug mode enabled: - Pins[a] used as N-Wire interface pins. - Internal pull-down resistor attached to P05/$\overline{\text{DRST}}$. **Note:** The inital value of the OCDM0 bit depends on the reset source.

[a] refer to *Table 25-1*

(2) Power-On-Clear RESPOC

RESPOC (Power-On-Clear) reset sets OCDM.OCDM0 = 0, i.e. the pins are defined as port pins. The debugger can not communicate with the controller and the N-Wire debug circuit is disabled. The first CPU instructions after RESPOC can not be controlled by the debugger. The application software must set OCDM.OCDM0 = 1 in order to enable the N-Wire interface and allow debugger access to the on-chip debug unit.
During and after POC reset (OCDM.OCDM0 = 0) pins P05, P52…P55 are configured as input ports.

(3) External $\overline{\text{RESET}}$

External reset by the $\overline{\text{RESET}}$ pin sets OCDM.OCDM0 = 1, i.e. the pins are defined as N-Wire interface pins. If connected the debugger can communicate with the on-chip debug unit and take over CPU control.

During and after $\overline{\text{RESET}}$ the pins P05, P52…P55 are configured as follows:

- $\overline{\text{DRST}}$, DDI, DCK, DMS are inputs.
- DDO is output, but in high impedance state as long as $\overline{\text{DRST}}$ = 0.

(4) Other resets

Resets from all other reset sources do not affect the pins P05, P52...P55.

An internal pull-down resistor is provided for the pin P05/$\overline{\text{DRST}}$. During and after any reset the resistor is connected to P05/$\overline{\text{DRST}}$, ensuring that the N-Wire interface is kept in reset state, if no debugger is connected. The internal pull-down resistor is connected by reset from any source and can be disconnected via OCDM.OCDM0.

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 impact 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.

Pin configuration

In N-Wire debug mode the configuration of the N-Wire interface pins can not be changed by the pin configuration registers. The registers contents can be changed but will have no effect on the pin configuration.

25.3 N-Wire Enabling Methods

The current operation mode of the microcontroller is determined by OCDM.OCDM0 and $\overline{\text{DRST}}$:

Table 25-3 Normal operation and debug mode control

$\overline{\text{DRST}}$	OCDM.OCDM0	Mode
0	×	normal operation
1	0	
	1	on-chip debug

25.3.1 Starting normal operation after $\overline{\text{RESET}}$ and RESPOC

For “normal operation” it has to be assured that the pins P05, P52…P55 are available as port pins after either reset event. Therefore the software has to perform OCDM.OCDM0 = 0 to make the pins available as port pins after $\overline{\text{RESET}}$.

Note that after any external reset via the $\overline{\text{RESET}}$ pin OCDM.OCDM0 is set to “1” and the pins P05, P52…P55 are not available as application function pins until the software sets OCDM.OCDM0 = 0.

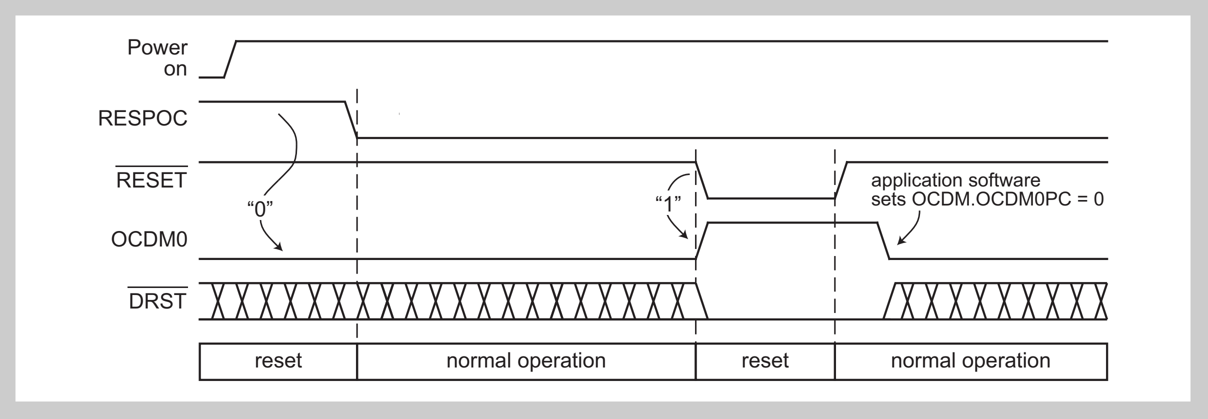


Figure 25-1 Start without N-Wire activation

25.3.2 Starting debugger after $\overline{\text{RESET}}$ and RESPOC

The software has to set OCDM.OCDM0 = 1 for enabling the N-Wire interface also upon a RESPOC event. Afterwards the debugger may start to establish communication with the controller by setting the $\overline{\text{DRST}}$ pin to high level and to take control over the CPU.

On start of the debugger 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.

Note After RESPOC the controller is operating without debugger control. Thus all CPU instructions until the software performs OCDM.OCDM0 = 1 can not be debugged. To restart the user’s program from beginning under the debugger’s control apply an external $\overline{\text{RESET}}$ after the debugger has started, as shown in

Figure 25-2. 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 $\overline{\text{RESET}}$ is applied.

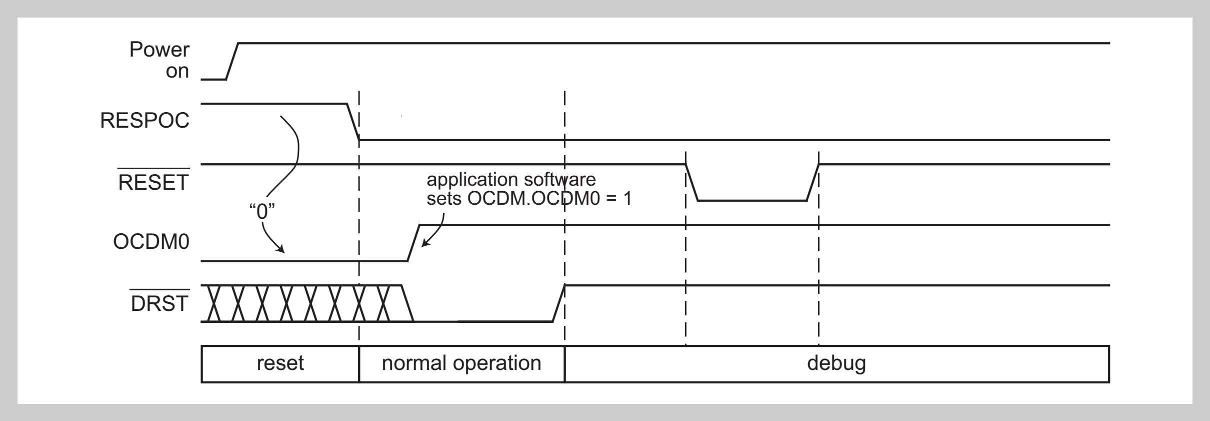


Figure 25-2 Start with N-Wire activation

25.3.3 N-Wire activation by $\overline{\text{RESET}}$ pin

The N-Wire interface can also be activated after power up by keeping $\overline{\text{RESET}}$ active after RESPOC is released. By this OCDM.OCDM0 is set to "1", thus the N-Wire interface is enabled.

With this method the user's program does not need to perform OCDM.OCDM0 = 1.

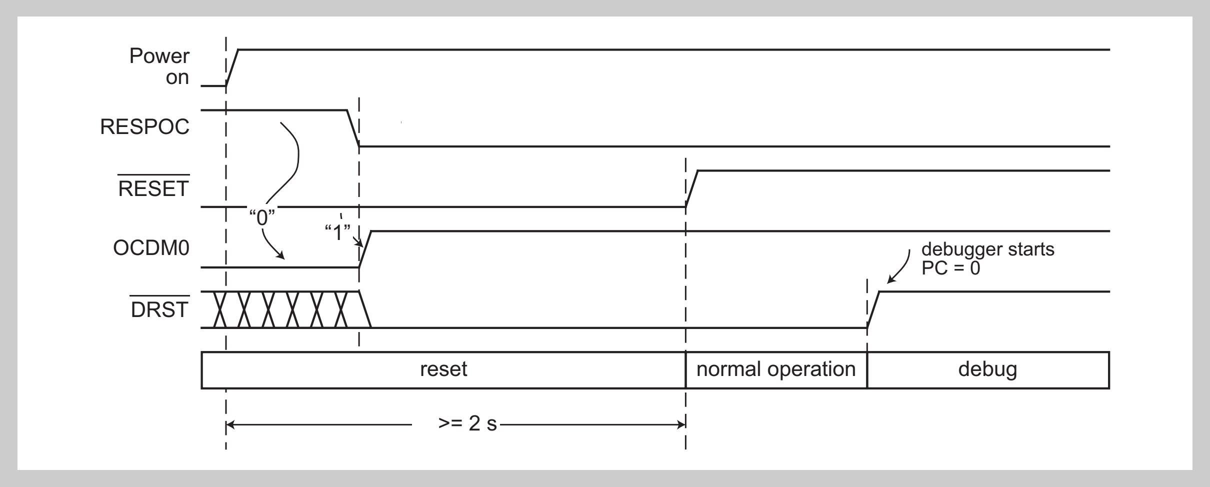


Figure 25-3 N-Wire activation by $\overline{\text{RESET}}$ pin

25.4 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, like for instance 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 user’s manual of the emulator to be used.

25.4.1 KEL connector

KEL connector product names:

- 8830E-026-170S (KEL): straight type
- 8830E-026-170L (KEL): right-angle type

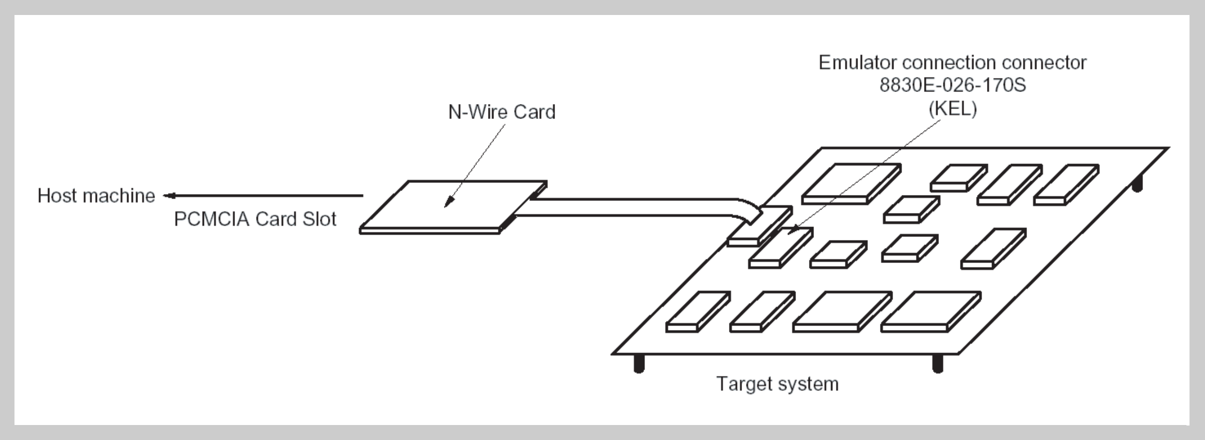


Figure 25-4 Connection to N-Wire emulator (IE-V850E1-CD-NW: N-Wire Card)

(1) Pin configuration

Figure 25-5 shows the pin configuration of the connector for emulator connection (target system side), and *Table 25-4 on page 939* shows the pin functions.

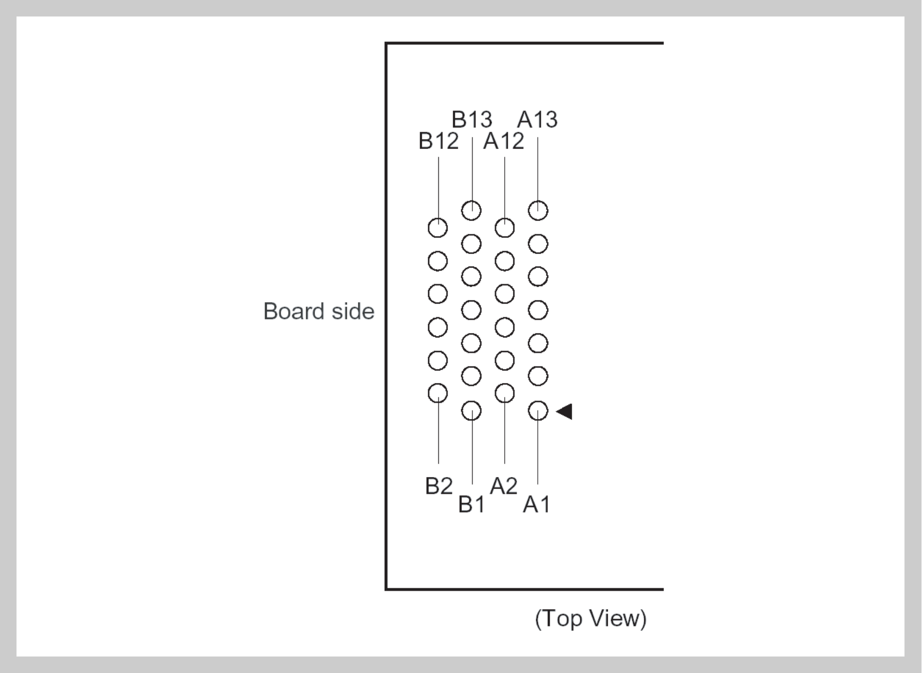


Figure 25-5 Pin configuration of connector for emulator connection (target system side)

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 25-4 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. (In a system that uses only POC reset and not pin reset, some emulators input an external reset signal as shown in *Figure 25-6 on page 940* to set the OCDM0 bit to 1.)
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}	–	5 V input (for monitoring power supply to target)

Caution

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.

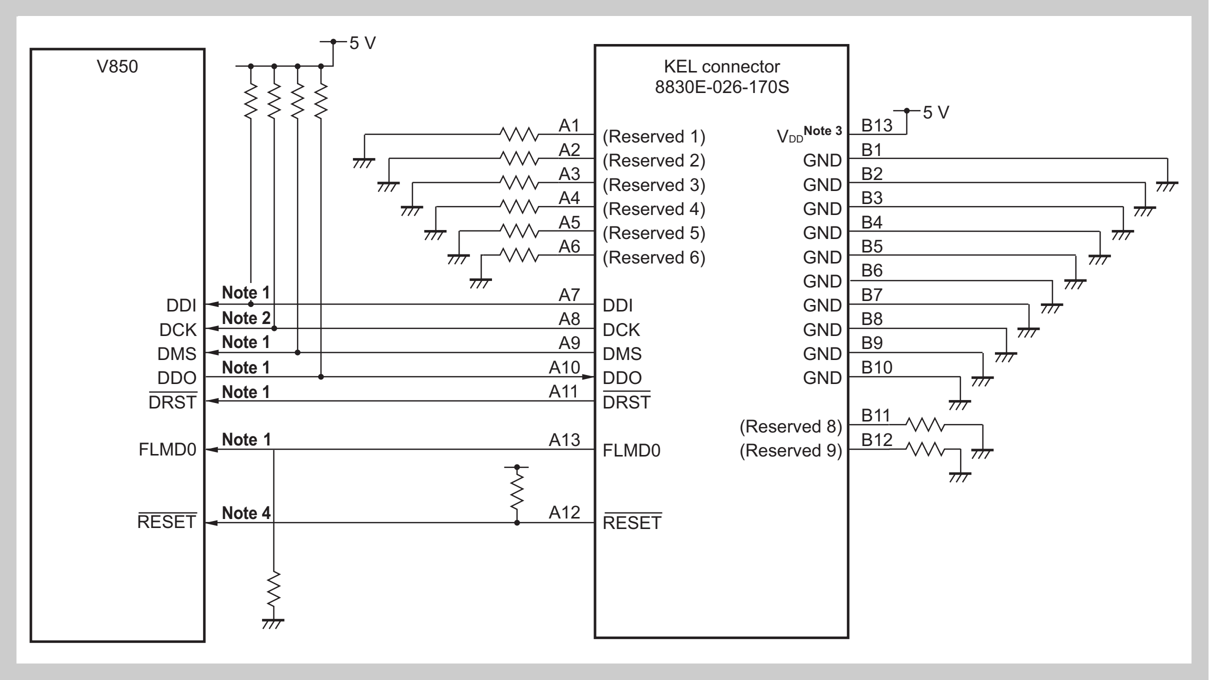


Figure 25-6 Example of recommended emulator connection circuit

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.
4. In a system that uses only POC reset and not pin reset, some emulators input an external reset signal as shown in *Figure 25-6* to set the OCDM.OCDM0 bit to 1.

Caution The N-Wire emulator 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.

25.5 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 code flash memory was rewritten during debugging and the number of rewrites of the code 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 code 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 POC reset operation cannot be emulated.
- The on-chip debugging unit uses the exception vector address 60_H for software breakpoint (DBTRAP, refer to *“Interrupt Controller (INTC)” on page 248*). 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 27.2 (1) On-chip debug mode register (OCDM).
- When the break command is started in on-chip debug (OCD) mode and the application software accesses to the UARTD/CSIB/CAN peripheral I/O registers, CSIB, 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.

Chapter 26 Reset

Several reset functions are provided in order to initialize hardware and registers.

26.1 Overview

Features summary

An internal system reset SYSRES can be generated by the following sources:

- External reset signal $\overline{\text{RESET}}$
- Power-On-Clear (RESPOC)
- Watchdog Timer 2 (RESWDT2)
- Clock Monitor (RESCLM)
- Low-Voltage Detector (RESLVI)

26.1.1 General reset performance

The following figure shows the signals involved in the reset function.

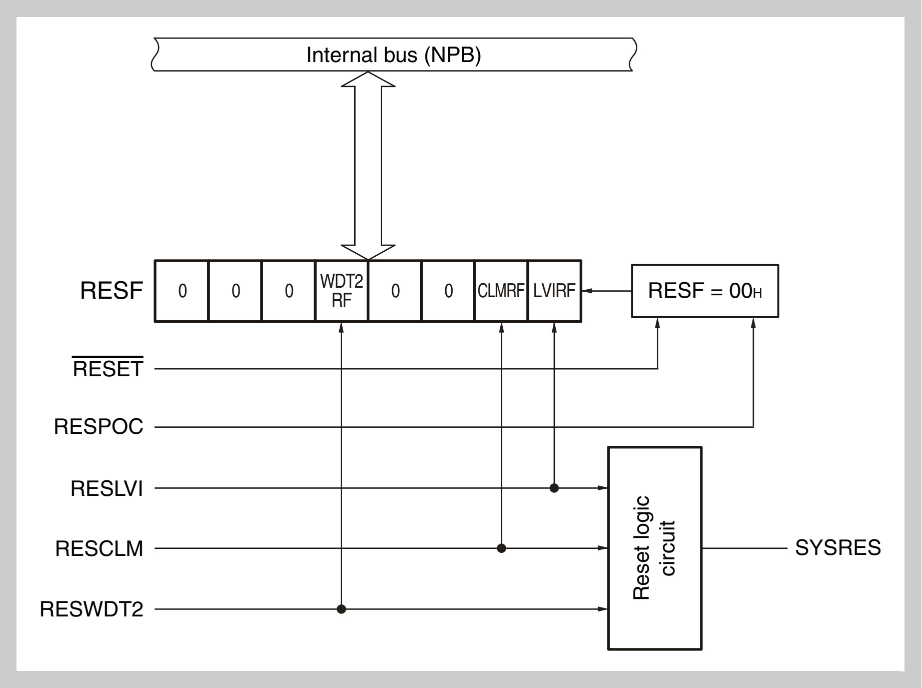


Figure 26-1 Reset function signal diagram

All resets are applied asynchronously. That means, resets are not synchronized to any internal clock. This ensures that the microcontroller can be kept in reset state even if all internal clocks fail to operate.

(1) Hardware status

With each reset function the hardware is initialized. When the reset status is released, program execution is started.

The following table describes the status of the clocks and on-chip modules during reset and after reset release.

Table 26-1 Hardware status during and after reset

Item		During reset	After reset
General clock supplies		Refer to *“Start conditions” on page 186*	
On-chip peripheral functions			
	Watch Timer WT	Operating on f_{XT}	
	Watchdog Timer WDT2	Stopped	Starts operation based on f_{RL}, after internal oscillator is stable.
	all others	Stopped	Operable based on f_{RH}, after internal oscillator is stable.
CPU		Initialized	Program execution starts based on f_{RH}, after internal oscillatorinternal oscillator is stable.
I/O pins (port/alternative function pins)		All pins are in input port mode[a].	

[a] The status of the N-Wire debug interface pins $\overline{\text{DRST}}$ (P05), DDI (P52), DDO (P53), DCK (P54), DMS (P55) after reset depends on the reset value of the OCDM register, and therefore on the reset source. See chapter *“Pin Functions” on page 32* for details.

(2) Register status

With each reset function the registers of the CPU, internal RAM, and on-chip peripheral I/Os are initialized. After a reset, make sure to set the registers to the values needed within your program.

Table 26-2 Initial values of CPU and internal RAM after reset

On-chip hardware		Register name	Initial value after Reset
CPU	Program registers	General-purpose register (r0)	0000 0000_H
		General-purpose registers (r1 to r31)	Undefined
		Program counter (PC)	Reset vector programmed to the code flash memory extra area[a]
	System registers	Status save registers during interrupt (EIPC, EIPSW)	Undefined
		Status save registers during non-maskable interrupt (NMI) (FEPC, FEPSW)	Undefined
		Interrupt cause register (ECR)	0000 0000_H
		Program status word (PSW)	0000 0020_H
		Status save registers during CALLT execution (CTPC, CTPSW)	Undefined
		Status save registers during exception/ debug trap (DBPC, DBPSW)	Undefined
		CALLT base pointer (CTBP)	Undefined
Internal RAM			Undefined
Peripherals		Macro internal registers	The reset values of the various registers are given in the chapters of the peripheral functions

a) After reset, the internal Firmware is executed. When execution of the Firmware is finished, it performs a program branch according to the user defined reset vector. The reset vector is stored in the extra flash area.

Internal RAM data becomes undefined after power-on reset, or if RAM data access by the CPU and a reset input conflict (data is lost).

Addtionally the following resources are used by the internal firmware executed after a reset:

- The first 150 bytes and the last 100 bytes of the available RAM area are undefined.
- Program status word (PSW) is undefined, but interrupts are disabled

26.1.2 Reset at power-on

The Power-On-Clear circuit (POC) permanently compares the power supply voltage V_{DD} with an internal reference voltage (V_{IP}). It ensures that the microcontroller only operates as long as the power supply exceeds a well-defined limit.

When the power supply voltage falls below the internal reference voltage ($V_{DD} < V_{IP}$), the internal reset signal RESPOC is generated.

After Power-On-Clear reset, the RESF register is cleared and the internal reset SYSRES is generated.

Note 1. POC shares the reference voltage supply with the power regulators.

Figure 26-2 on page 945 shows the generation of RESPOC by the Power-On-Clear circuit.

The Power-On-Clear function holds the microcontroller in reset state as long as the power supply voltage does not exceed the threshold level VPOC.

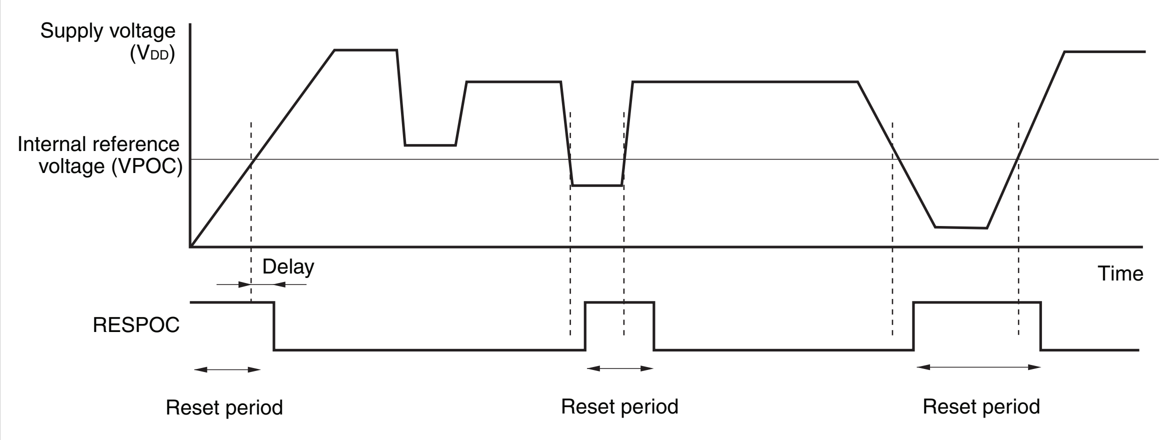


Figure 26-2 Reset generation by Power-On-Clear circuit

Figure 26-3 on page 946 outlines the start up of the CPU system after Power-On-Clear.

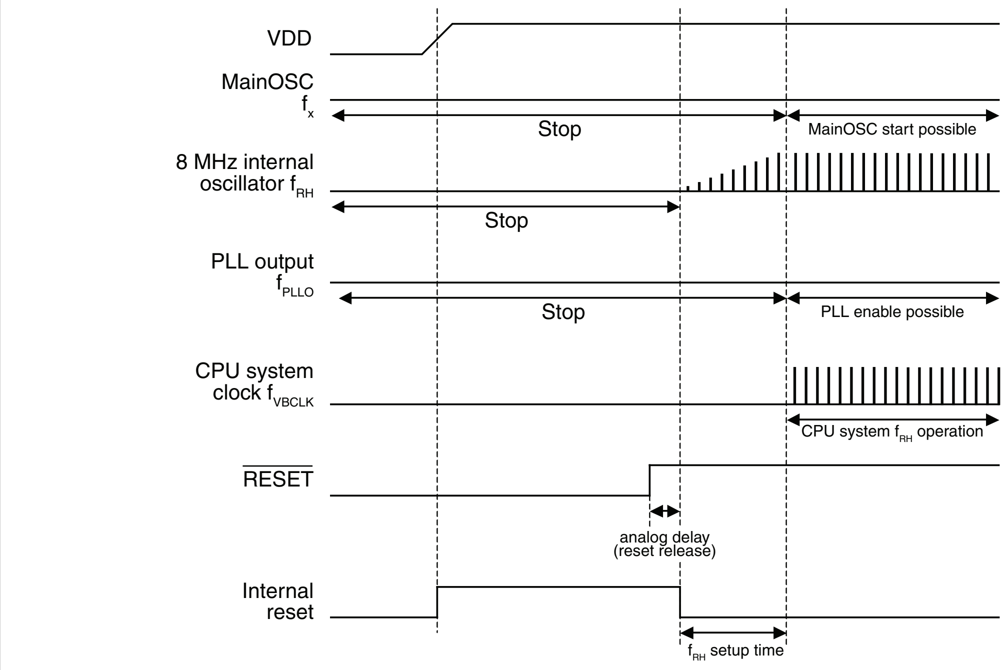


Figure 26-3 CPU system start up after Power-On-Clear

26.1.3 External $\overline{\text{RESET}}$

Reset is performed when a low level signal is applied to the $\overline{\text{RESET}}$ pin.

The reset status is released when the signal applied to the $\overline{\text{RESET}}$ pin changes from low to high.

After the external $\overline{\text{RESET}}$ is released, the RESF register is cleared and the internal system reset signal SYSRES is generated.

The $\overline{\text{RESET}}$ pin incorporates a noise eliminator, which is applied to the reset signal $\overline{\text{RESET}}$. To prevent erroneous external reset due to noise, it uses an analog filter. Even if no clock is active in the controller the external $\overline{\text{RESET}}$ can keep the controller in reset state.

The following figure shows the timing when an external $\overline{\text{RESET}}$ is performed. It explains the effect of the noise eliminator. The noise eliminator uses the analog delay to prevent the generation of an external reset due to noise.

The analog delay is caused by the analog input filter. The filter regards pulses up to a certain width as noise and suppresses them. For the minimum $\overline{\text{RESET}}$ pulse width refer to the Datasheet.

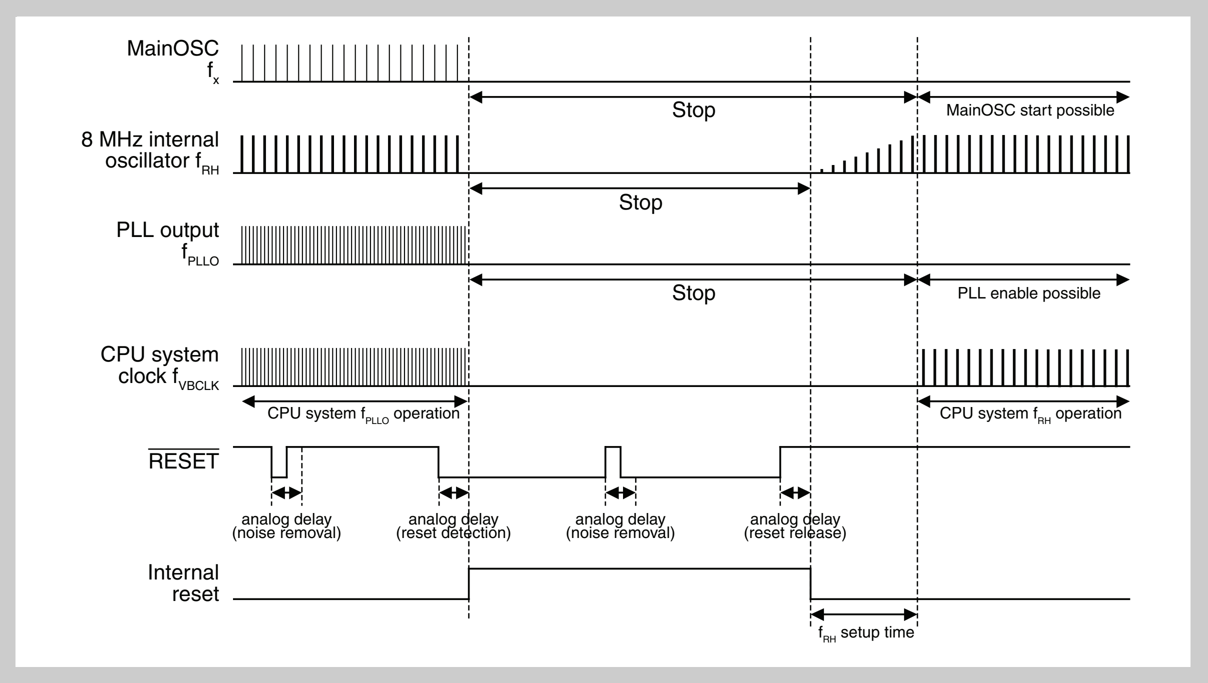

MainOSC f_x
Stop
MainOSC start possible
8 MHz internal oscillator f_{RH}
Stop
PLL output f_{PLLO}
Stop
PLL enable possible
CPU system clock f_{VBCLK}
CPU system f_{PLLO} operation
CPU system f_{RH} operation
$\overline{\text{RESET}}$
analog delay (noise removal)
analog delay (reset detection)
analog delay (noise removal)
analog delay (reset release)
Internal reset
f_{RH} setup time

Figure 26-4 Timing for external $\overline{\text{RESET}}$

26.1.4 Reset by Watchdog Timer 2

The Watchdog Timer can be configured to generate a reset if the watchdog time overflows. After watchdog reset, the RESF.WDT2RF bit is set. The system reset signal SYSRES is generated and the system resets.

26.1.5 Reset by Clock Monitor

The Clock Monitor generates a reset when the main oscillator fails. After a Clock Monitor reset, the corresponding bit RESF.CLMRF is set. The system reset signal SYSRES is generated and the systems resets.

26.1.6 Reset by Low-Voltage Detector

The Low-Voltage Detector can be configured to generate the reset RESLVI if the voltage supply V_{DD} falls below the reference voltage V_{LVI}. RESLVI sets the bit RESF.LVIRF and the system reset signal SYSRES is generated and the system resets.

26.2 Reset Registers

The reset functions are controlled and operated by means of the following registers:

Table 26-3 Reset function register overview

Register name	Shortcut	Address
Reset source flag register	RESF	FFFF F888$_H$

(1) RESF - Reset source flag register

The 8-bit RESF register contains information about which type of resets occurred since the last Power-On-Clear or external $\overline{\text{RESET}}$.

The RESF register is a special register that can be written only by specific sequences.

Each following reset condition sets the corresponding flag in the register. For example, if a Power-On-Clear reset is finished and then a Watchdog Timer reset occurs, the RESF reads 0001 0000$_B$.

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

Address FFFF F888$_H$

Initial Value Power-On-Clear reset and external $\overline{\text{RESET}}$ sets this register to 00$_H$.

	7	6	5	4	3	2	1	0
RESF	0	0	0	WDT2RF	0	0	CLMRF	LVIRF
	R	R	R	R/W	R	R	R/W	R/W

Note If clearing this register by writing and flag setting (occurrence of reset) conflict, flag setting takes precedence.

Table 26-4 RESF register contents

Bit position	Bit name	Function
4	WDT2RF	Reset by Watchdog Timer 0: Not generated. 1: Generated.
1	CLMRF	Reset by Clock Monitor 0: Not generated. 1: Generated.
0	LVIRF	Reset by Low-Voltage Detector 0: Not generated. 1: Generated.

Appendix A Special Function Registers

The following tables list all registers that are accessed via the NPB (peripheral bus). The registers are called “special function registers” (SFR).

Table A-5 lists all CAN special function registers. The addresses are given as offsets to the programmable peripheral base address (refer to *“CAN module register and message buffer addresses” on page 700*.

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-5 CAN special function registers (1/4)

Address offset	Register name	Shortcut	1	8	16	32
0x000	CAN0 global control register	C0GMCTRL	-	-	R/W	-
0x002	CAN0 global clock selection register	C0GMCS	-	R/W	-	-
0x006	CAN0 global automatic block transmission register	C0GMABT	-	-	R/W	-
0x008	CAN0 global automatic block transmission delay register	C0GMABTD	-	R/W	-	-
0x040	CAN0 module mask 1 register	C0MASK1L	-	-	R/W	-
0x042		C0MASK1H	-	-	R/W	-
0x044	CAN0 module mask 2 register	C0MASK2L	-	-	R/W	-
0x046		C0MASK2H	-	-	R/W	-
0x048	CAN0 module mask 3 register	C0MASK3L	-	-	R/W	-
0x04A		C0MASK3H	-	-	R/W	-
0x04C	CAN0 module mask 4 register	C0MASK4L	-	-	R/W	-
0x04E		C0MASK4H	-	-	R/W	-
0x050	CAN0 module control register	C0CTRL	-	-	R/W	-
0x052	CAN0 module last error code register	C0LEC	-	R/W	-	-
0x053	CAN0 module information register	C0INFO	-	R	-	-
0x054	CAN0 module error counter register	C0ERC	-	-	R	-
0x056	CAN0 module interrupt enable register	C0IE	-	-	R/W	-
0x058	CAN0 module interrupt status register	C0INTS	-	-	R/W	-
0x05A	CAN0 module bit-rate prescaler register	C0BRP	-	R/W	-	-
0x05C	CAN0 module bit-rate register	C0BTR	-	-	R/W	-
0x05E	CAN0 module last in-pointer register	C0LIPT	-	R	-	-
0x060	CAN0 module receive history list register	C0RGPT	-	-	R/W	-
0x062	CAN0 module last out-pointer register	C0LOPT	-	R	-	-
0x064	CAN0 module transmit history list register	C0TGPT	-	-	R/W	-

Table A-5 CAN special function registers (2/4)

Address offset	Register name	Shortcut	1	8	16	32
0x066	CAN0 module time stamp register	C0TS	-	-	R/W	-
0x100 to 0x4EF	CAN0 Message Buffer registers, see *Table 20-20 on page 703*					
0x600	CAN1 global control register	C1GMCTRL	-	-	R/W	-
0x602	CAN1 global clock selection register	C1GMCS	-	R/W	-	-
0x606	CAN1 global automatic block transmission register	C1GMABT	-	-	R/W	-
0x608	CAN1 global automatic block transmission delay register	C1GMABTD	-	R/W	-	-
0x640	CAN1 module mask 1 register	C1MASK1L	-	-	R/W	-
0x642		C1MASK1H	-	-	R/W	-
0x644	CAN1 module mask 2 register	C1MASK2L	-	-	R/W	-
0x646		C1MASK2H	-	-	R/W	-
0x648	CAN1 module mask 3 register	C1MASK3L	-	-	R/W	-
0x64A		C1MASK3H	-	-	R/W	-
0x64C	CAN1 module mask 4 register	C1MASK4L	-	-	R/W	-
0x64E		C1MASK4H	-	-	R/W	-
0x650	CAN1 module control register	C1CTRL	-	-	R/W	-
0x652	CAN1 module last error code register	C1LEC	-	R/W	-	-
0x653	CAN1 module information register	C1INFO	-	R	-	-
0x654	CAN1 module error counter register	C1ERC	-	-	R	-
0x656	CAN1 module interrupt enable register	C1IE	-	-	R/W	-
0x658	CAN1 module interrupt status register	C1INTS	-	-	R/W	-
0x65A	CAN1 module bit-rate prescaler register	C1BRP	-	R/W	-	-
0x65C	CAN1 module bit-rate register	C1BTR	-	-	R/W	-
0x65E	CAN1 module last in-pointer register	C1LIPT	-	R	-	-
0x660	CAN1 module receive history list register	C1RGPT	-	-	R/W	-
0x662	CAN1 module last out-pointer register	C1LOPT	-	R	-	-
0x664	CAN1 module transmit history list register	C1TGPT	-	-	R/W	-
0x666	CAN1 module time stamp register	C1TS	-	-	R/W	-
0x700 to 0xAEF	CAN1 Message Buffer registers, see *Table 20-22 on page 705*					
0xC00	CAN2 global control register	C2GMCTRL	-	-	R/W	-
0xC02	CAN2 global clock selection register	C2GMCS	-	R/W	-	-
0xC06	CAN2 global automatic block transmission register	C2GMABT	-	-	R/W	-
0xC08	CAN2 global automatic block transmission delay register	C2GMABTD	-	R/W	-	-
0xC40	CAN2 module mask 1 register	C2MASK1L	-	-	R/W	-
0xC42		C2MASK1H	-	-	R/W	-
0xC44	CAN2 module mask 2 register	C2MASK2L	-	-	R/W	-
0xC46		C2MASK2H	-	-	R/W	-
0xC48	CAN2 module mask 3 register	C2MASK3L	-	-	R/W	-
0xC4A		C2MASK3H	-	-	R/W	-

Table A-5 CAN special function registers (3/4)

Address offset	Register name	Shortcut	1	8	16	32
0xC4C	CAN2 module mask 4 register	C2MASK4L	-	-	R/W	-
0xC4E		C2MASK4H	-	-	R/W	-
0xC50	CAN2 module control register	C2CTRL	-	-	R/W	-
0xC52	CAN2 module last error code register	C2LEC	-	R/W	-	-
0xC53	CAN2 module information register	C2INFO	-	R	-	-
0xC54	CAN2 module error counter register	C2ERC	-	-	R	-
0xC56	CAN2 module interrupt enable register	C2IE	-	-	R/W	-
0xC58	CAN2 module interrupt status register	C2INTS	-	-	R/W	-
0xC5A	CAN2 module bit-rate prescaler register	C2BRP	-	R/W	-	-
0xC5C	CAN2 module bit-rate register	C2BTR	-	-	R/W	-
0xC5E	CAN2 module last in-pointer register	C2LIPT	-	R	-	-
0xC60	CAN2 module receive history list register	C2RGPT	-	-	R/W	-
0xC62	CAN2 module last out-pointer register	C2LOPT	-	R	-	-
0xC64	CAN2 module transmit history list register	C2TGPT	-	-	R/W	-
0xC66	CAN2 module time stamp register	C2TS	-	-	R/W	-
0xD00 to 0x10EF	CAN2 Message Buffer registers, see *Table 20-24 on page 707*					
0x1200	CAN3 global control register	C3GMCTRL	-	-	R/W	-
0x1202	CAN3 global clock selection register	C3GMCS	-	R/W	-	-
0x1206	CAN3 global automatic block transmission register	C3GMABT	-	-	R/W	-
0x1208	CAN3 global automatic block transmission delay register	C3GMABTD	-	R/W	-	-
0x1240	CAN3 module mask 1 register	C3MASK1L	-	-	R/W	-
0x1242		C3MASK1H	-	-	R/W	-
0x1244	CAN3 module mask 2 register	C3MASK2L	-	-	R/W	-
0x1246		C3MASK2H	-	-	R/W	-
0x1248	CAN3 module mask 3 register	C3MASK3L	-	-	R/W	-
0x124A		C3MASK3H	-	-	R/W	-
0x124C	CAN3 module mask 4 register	C3MASK4L	-	-	R/W	-
0x124E		C3MASK4H	-	-	R/W	-
0x1250	CAN3 module control register	C3CTRL	-	-	R/W	-
0x1252	CAN3 module last error code register	C3LEC	-	R/W	-	-
0x1253	CAN3 module information register	C3INFO	-	R	-	-
0x1254	CAN3 module error counter register	C3ERC	-	-	R	-
0x1256	CAN3 module interrupt enable register	C3IE	-	-	R/W	-
0x1258	CAN3 module interrupt status register	C3INTS	-	-	R/W	-
0x125A	CAN3 module bit-rate prescaler register	C3BRP	-	R/W	-	-
0x125C	CAN3 module bit-rate register	C3BTR	-	-	R/W	-
0x125E	CAN3 module last in-pointer register	C3LIPT	-	R	-	-
0x1260	CAN3 module receive history list register	C3RGPT	-	-	R/W	-
0x1262	CAN3 module last out-pointer register	C3LOPT	-	R	-	-
0x1264	CAN3 module transmit history list register	C3TGPT	-	-	R/W	-

Table A-5 CAN special function registers (4/4)

Address offset	Register name	Shortcut	1	8	16	32
0x1266	CAN3 module time stamp register	C3TS	-	-	R/W	-
0x1300 to 0x16EF	CAN3 Message Buffer registers, see *Table 20-26 on page 709*					
0x1800	CAN4 global control register	C4GMCTRL	-	-	R/W	-
0x1802	CAN4 global clock selection register	C4GMCS	-	R/W	-	-
0x1806	CAN4 global automatic block transmission register	C4GMABT	-	-	R/W	-
0x1808	CAN4 global automatic block transmission delay register	C4GMABTD	-	R/W	-	-
0x1840	CAN4 module mask 1 register	C4MASK1L	-	-	R/W	-
0x1842		C4MASK1H	-	-	R/W	-
0x1844	CAN4 module mask 2 register	C4MASK2L	-	-	R/W	-
0x1846		C4MASK2H	-	-	R/W	-
0x1848	CAN4 module mask 3 register	C4MASK3L	-	-	R/W	-
0x184A		C4MASK3H	-	-	R/W	-
0x184C	CAN4 module mask 4 register	C4MASK4L	-	-	R/W	-
0x184E		C4MASK4H	-	-	R/W	-
0x1850	CAN4 module control register	C4CTRL	-	-	R/W	-
0x1852	CAN4 module last error code register	C4LEC	-	R/W	-	-
0x1853	CAN4 module information register	C4INFO	-	R	-	-
0x1854	CAN4 module error counter register	C4ERC	-	-	R	-
0x1856	CAN4 module interrupt enable register	C4IE	-	-	R/W	-
0x1858	CAN4 module interrupt status register	C4INTS	-	-	R/W	-
0x185A	CAN4 module bit-rate prescaler register	C4BRP	-	R/W	-	-
0x185C	CAN4 module bit-rate register	C4BTR	-	-	R/W	-
0x185E	CAN4 module last in-pointer register	C4LIPT	-	R	-	-
0x1860	CAN4 module receive history list register	C4RGPT	-	-	R/W	-
0x1862	CAN4 module last out-pointer register	C4LOPT	-	R	-	-
0x1864	CAN4 module transmit history list register	C4TGPT	-	-	R/W	-
0x1866	CAN4 module time stamp register	C4TS	-	-	R/W	-
0x1900 to 0x1CEF	CAN4 Message Buffer registers, see *Table 20-28 on page 711*					

A.2 Other Special Function Registers

Table A-6 Other special function registers (1/17)

Address	Register name	Shortcut	1	8	16	32
0xFFFFF004	PortDL	PDL	-	-	R/W	-
0xFFFFF004	PortDL low byte	PDLL	R/W	R/W	-	-
0xFFFFF005	PortDL high byte	PDLH	R/W	R/W	-	-
0xFFFFF008	PortCS	PCS	R/W	R/W	-	-
0xFFFFF00A	PortCT	PCT	R/W	R/W	-	-
0xFFFFF00C	PortCM	PCM	R/W	R/W	-	-
0xFFFFF00E	PortCD	PCD	R/W	R/W	-	-
0xFFFFF024	PortDL mode	PMDL	-	-	R/W	-
0xFFFFF024	PortDL mode low byte	PMDLL	R/W	R/W	-	-
0xFFFFF025	PortDL mode high byte	PMDLH	R/W	R/W	-	-
0xFFFFF028	PortCS mode	PMCS	R/W	R/W	-	-
0xFFFFF02A	PortCT mode	PMCT	R/W	R/W	-	-
0xFFFFF02C	PortCM mode	PMCM	R/W	R/W	-	-
0xFFFFF02E	PortCD mode	PMCD	R/W	R/W	-	-
0xFFFFF044	PortDL mode control	PMCDL	-	-	R/W	-
0xFFFFF044	PortDL mode control low byte	PMCDLL	R/W	R/W	-	-
0xFFFFF045	PortDL mode control high byte	PMCDLH	R/W	R/W	-	-
0xFFFFF048	PortCS mode control	PMCCS	R/W	R/W	-	-
0xFFFFF04A	PortCT mode control	PMCCT	R/W	R/W	-	-
0xFFFFF04C	PortCM mode control	PMCCM	R/W	R/W	-	-
0xFFFFF064	Peripheral I/O area select control register	BPC	-	-	R/W	-
0xFFFFF066	Bus size configuration register	BSC	-	-	R/W	-
0xFFFFF06E	System wait control register	VSWC	-	R/W	-	-
0xFFFFF080	DMA source address register 0L	DSA0L	-	-	R/W	-
0xFFFFF082	DMA source address register 0H	DSA0H	-	-	R/W	-
0xFFFFF084	DMA destination address register 0L	DDA0L	-	-	R/W	-
0xFFFFF086	DMA destination address register 0H	DDA0H	-	-	R/W	-
0xFFFFF088	DMA source address register 1L	DSA1L	-	-	R/W	-
0xFFFFF08A	DMA source address register 1H	DSA1H	-	-	R/W	-
0xFFFFF08C	DMA destination address register 1L	DDA1L	-	-	R/W	-
0xFFFFF08E	DMA destination address register 1H	DDA1H	-	-	R/W	-
0xFFFFF090	DMA source address register 2L	DSA2L	-	-	R/W	-
0xFFFFF092	DMA source address register 2H	DSA2H	-	-	R/W	-
0xFFFFF094	DMA destination address register 2L	DDA2L	-	-	R/W	-
0xFFFFF096	DMA destination address register 2H	DDA2H	-	-	R/W	-
0xFFFFF098	DMA source address register 3L	DSA3L	-	-	R/W	-
0xFFFFF09A	DMA source address register 3H	DSA3H	-	-	R/W	-
0xFFFFF09C	DMA destination address register 3L	DDA3L	-	-	R/W	-
0xFFFFF09E	DMA destination address register 3H	DDA3H	-	-	R/W	-

Table A-6 Other special function registers (2/17)

Address	Register name	Shortcut	1	8	16	32
0xFFFFF0C0	DMA transfer count register 0	DBC0	-	-	R/W	-
0xFFFFF0C2	DMA transfer count register 1	DBC1	-	-	R/W	-
0xFFFFF0C4	DMA transfer count register 2	DBC2	-	-	R/W	-
0xFFFFF0C6	DMA transfer count register 3	DBC3	-	-	R/W	-
0xFFFFF0D0	DMA addressing control register 0	DADC0	-	-	R/W	-
0xFFFFF0D2	DMA addressing control register 1	DADC1	-	-	R/W	-
0xFFFFF0D4	DMA addressing control register 2	DADC2	-	-	R/W	-
0xFFFFF0D6	DMA addressing control register 3	DADC3	-	-	R/W	-
0xFFFFF0E0	DMA channel control register 0	DCHC0	R/W	R/W	-	-
0xFFFFF0E2	DMA channel control register 1	DCHC1	R/W	R/W	-	-
0xFFFFF0E4	DMA channel control register 2	DCHC2	R/W	R/W	-	-
0xFFFFF0E6	DMA channel control register 3	DCHC3	R/W	R/W	-	-
0xFFFFF100	Interrupt mask control register 0	IMR0	-	-	R/W	-
0xFFFFF100	Interrupt mask control register 0L	IMR0L	R/W	R/W	-	-
0xFFFFF101	Interrupt mask control register 0H	IMR0H	R/W	R/W	-	-
0xFFFFF102	Interrupt mask control register 1	IMR1	-	-	R/W	-
0xFFFFF102	Interrupt mask control register 1L	IMR1L	R/W	R/W	-	-
0xFFFFF103	Interrupt mask control register 1H	IMR1H	R/W	R/W	-	-
0xFFFFF104	Interrupt mask control register 2	IMR2	-	-	R/W	-
0xFFFFF104	Interrupt mask control register 2L	IMR2L	R/W	R/W	-	-
0xFFFFF105	Interrupt mask control register 2H	IMR2H	R/W	R/W	-	-
0xFFFFF106	Interrupt mask control register 3	IMR3	-	-	R/W	-
0xFFFFF106	Interrupt mask control register 3L	IMR3L	R/W	R/W	-	-
0xFFFFF107	Interrupt mask control register 3H	IMR3H	R/W	R/W	-	-
0xFFFFF108	Interrupt mask control register 4	IMR4	-	-	R/W	-
0xFFFFF108	Interrupt mask control register 4L	IMR4L	R/W	R/W	-	-
0xFFFFF109	Interrupt mask control register 4H	IMR4H	R/W	R/W	-	-
0xFFFFF10A	Interrupt mask control register 5	IMR5	-	-	R/W	-
0xFFFFF10A	Interrupt mask control register 5L	IMR5L	R/W	R/W	-	-
0xFFFFF10B	Interrupt mask control register 5H	IMR5H	R/W	R/W	-	-
0xFFFFF10C	Interrupt mask control register 6	IMR6	-	-	R/W	-
0xFFFFF10C	Interrupt mask control register 6L	IMR6L	R/W	R/W	-	-
0xFFFFF10D	Interrupt mask control register 6H	IMR6H	R/W	R/W	-	-
0xFFFFF10E	Interrupt mask control register 7	IMR7	-	-	R/W	-
0xFFFFF10E	Interrupt mask control register 7L	IMR7L	R/W	R/W	-	-
0xFFFFF110	Interrupt control register	LVILIC	R/W	R/W	-	-
0xFFFFF112	Interrupt control register	LVIHIC	R/W	R/W	-	-
0xFFFFF114	Interrupt control register	PIC0	R/W	R/W	-	-
0xFFFFF116	Interrupt control register	PIC1	R/W	R/W	-	-
0xFFFFF118	Interrupt control register	PIC2	R/W	R/W	-	-
0xFFFFF11A	Interrupt control register	PIC3	R/W	R/W	-	-

Table A-6 Other special function registers (3/17)

Address	Register name	Shortcut	1	8	16	32
0xFFFFF11C	Interrupt control register	PIC4	R/W	R/W	-	-
0xFFFFF11E	Interrupt control register	PIC5	R/W	R/W	-	-
0xFFFFF120	Interrupt control register	PIC6	R/W	R/W	-	-
0xFFFFF122	Interrupt control register	PIC7	R/W	R/W	-	-
0xFFFFF124	Interrupt control register	TAB0OVIC	R/W	R/W	-	-
0xFFFFF126	Interrupt control register	TAB0CCIC0	R/W	R/W	-	-
0xFFFFF128	Interrupt control register	TAB0CCIC1	R/W	R/W	-	-
0xFFFFF12A	Interrupt control register	TAB0CCIC2	R/W	R/W	-	-
0xFFFFF12C	Interrupt control register	TAB0CCIC3	R/W	R/W	-	-
0xFFFFF12E	Interrupt control register	TAA0OVIC	R/W	R/W	-	-
0xFFFFF130	Interrupt control register	TAA0CCIC0	R/W	R/W	-	-
0xFFFFF132	Interrupt control register	TAA0CCIC1	R/W	R/W	-	-
0xFFFFF134	Interrupt control register	TAA1OVIC	R/W	R/W	-	-
0xFFFFF136	Interrupt control register	TAA1CCIC0	R/W	R/W	-	-
0xFFFFF138	Interrupt control register	TAA1CCIC1	R/W	R/W	-	-
0xFFFFF13A	Interrupt control register	TAA2OVIC	R/W	R/W	-	-
0xFFFFF13C	Interrupt control register	TAA2CCIC0	R/W	R/W	-	-
0xFFFFF13E	Interrupt control register	TAA2CCIC1	R/W	R/W	-	-
0xFFFFF140	Interrupt control register	TAA3OVIC	R/W	R/W	-	-
0xFFFFF142	Interrupt control register	TAA3CCIC0	R/W	R/W	-	-
0xFFFFF144	Interrupt control register	TAA3CCIC1	R/W	R/W	-	-
0xFFFFF146	Interrupt control register	TAA4OVIC	R/W	R/W	-	-
0xFFFFF148	Interrupt control register	TAA4CCIC0	R/W	R/W	-	-
0xFFFFF14A	Interrupt control register	TAA4CCIC1	R/W	R/W	-	-
0xFFFFF14C	Interrupt control register	TM0EQIC0	R/W	R/W	-	-
0xFFFFF14E	Interrupt control register	CB0RIC	R/W	R/W	-	-
0xFFFFF150	Interrupt control register	CB0TIC	R/W	R/W	-	-
0xFFFFF152	Interrupt control register	CB1RIC	R/W	R/W	-	-
0xFFFFF154	Interrupt control register	CB1TIC	R/W	R/W	-	-
0xFFFFF156	Interrupt control register	UD0SIC	R/W	R/W	-	-
0xFFFFF158	Interrupt control register	UD0RIC	R/W	R/W	-	-
0xFFFFF15A	Interrupt control register	UD0TIC	R/W	R/W	-	-
0xFFFFF15C	Interrupt control register	UD1SIC	R/W	R/W	-	-
0xFFFFF15E	Interrupt control register	UD1RIC	R/W	R/W	-	-
0xFFFFF160	Interrupt control register	UD1TIC	R/W	R/W	-	-
0xFFFFF162	Interrupt control register	IIC0IC UD4SIC	R/W	R/W	-	-
0xFFFFF164	Interrupt control register	ADIC	R/W	R/W	-	-
0xFFFFF166	Interrupt control register	C0ERRIC	R/W	R/W	-	-
0xFFFFF168	Interrupt control register	C0WUPIC	R/W	R/W	-	-
0xFFFFF16A	Interrupt control register	C0RECIC	R/W	R/W	-	-
0xFFFFF16C	Interrupt control register	C0TRXIC	R/W	R/W	-	-

Table A-6 Other special function registers (4/17)

Address	Register name	Shortcut	1	8	16	32
0xFFFFF16E	Interrupt control register	DMAIC0	R/W	R/W	-	-
0xFFFFF170	Interrupt control register	DMAIC1	R/W	R/W	-	-
0xFFFFF172	Interrupt control register	DMAIC2	R/W	R/W	-	-
0xFFFFF174	Interrupt control register	DMAIC3	R/W	R/W	-	-
0xFFFFF176	Interrupt control register	KRIC	R/W	R/W	-	-
0xFFFFF178	Interrupt control register	WTIIC	R/W	R/W	-	-
0xFFFFF17A	Interrupt control register	WTIC	R/W	R/W	-	-
0xFFFFF17E	Interrupt control register	FLIC	R/W	R/W	-	-
0xFFFFF180	Interrupt control register	PIC8	R/W	R/W	-	-
0xFFFFF182	Interrupt control register	PIC9	R/W	R/W	-	-
0xFFFFF184	Interrupt control register	PIC10	R/W	R/W	-	-
0xFFFFF186	Interrupt control register	TAB1OVIC	R/W	R/W	-	-
0xFFFFF188	Interrupt control register	TAB1CCIC0	R/W	R/W	-	-
0xFFFFF18A	Interrupt control register	TAB1CCIC1	R/W	R/W	-	-
0xFFFFF18C	Interrupt control register	TAB1CCIC2	R/W	R/W	-	-
0xFFFFF18E	Interrupt control register	TAB1CCIC3	R/W	R/W	-	-
0xFFFFF190	Interrupt control register	UD2SIC	R/W	R/W	-	-
0xFFFFF192	Interrupt control register	UD2RIC	R/W	R/W	-	-
0xFFFFF194	Interrupt control register	UD2TIC	R/W	R/W	-	-
0xFFFFF196	Interrupt control register	C1ERRIC	R/W	R/W	-	-
0xFFFFF198	Interrupt control register	C1WUPIC	R/W	R/W	-	-
0xFFFFF19A	Interrupt control register	C1RECIC	R/W	R/W	-	-
0xFFFFF19C	Interrupt control register	C1TRXIC	R/W	R/W	-	-
0xFFFFF19E	Interrupt control register	PIC11	R/W	R/W	-	-
0xFFFFF1A0	Interrupt control register	PIC12	R/W	R/W	-	-
0xFFFFF1A2	Interrupt control register	PIC13	R/W	R/W	-	-
0xFFFFF1A4	Interrupt control register	PIC14	R/W	R/W	-	-
0xFFFFF1A6	Interrupt control register	UD3SIC	R/W	R/W	-	-
0xFFFFF1A8	Interrupt control register	UD3RIC	R/W	R/W	-	-
0xFFFFF1AA	Interrupt control register	UD3TIC	R/W	R/W	-	-
0xFFFFF1AC	Interrupt control register	UD4RIC	R/W	R/W	-	-
0xFFFFF1AE	Interrupt control register	UD4TIC	R/W	R/W	-	-
0xFFFFF1B0	Interrupt control register	TAB2OVIC	R/W	R/W	-	-
0xFFFFF1B2	Interrupt control register	TAB2CCIC0	R/W	R/W	-	-
0xFFFFF1B4	Interrupt control register	TAB2CCIC1	R/W	R/W	-	-
0xFFFFF1B6	Interrupt control register	TAB2CCIC2	R/W	R/W	-	-
0xFFFFF1B8	Interrupt control register	TAB2CCIC3	R/W	R/W	-	-
0xFFFFF1BA	Interrupt control register	UD5SIC	R/W	R/W	-	-
0xFFFFF1BC	Interrupt control register	CB2RIC UD5RIC	R/W	R/W	-	-
0xFFFFF1BE	Interrupt control register	CB2TIC UD5TIC	R/W	R/W	-	-

Table A-6 Other special function registers (5/17)

Address	Register name	Shortcut	1	8	16	32
0xFFFFF1C0	Interrupt control register	C2ERRIC	R/W	R/W	-	-
0xFFFFF1C2	Interrupt control register	C2WUPIC	R/W	R/W	-	-
0xFFFFF1C4	Interrupt control register	C2RECIC	R/W	R/W	-	-
0xFFFFF1C6	Interrupt control register	C2TRXIC	R/W	R/W	-	-
0xFFFFF1C8	Interrupt control register	C3ERRIC	R/W	R/W	-	-
0xFFFFF1CA	Interrupt control register	C3WUPIC	R/W	R/W	-	-
0xFFFFF1CC	Interrupt control register	C3RECIC	R/W	R/W	-	-
0xFFFFF1CE	Interrupt control register	C3TRXIC	R/W	R/W	-	-
0xFFFFF1D0	Interrupt control register	PIC15	R/W	R/W	-	-
0xFFFFF1D2	Interrupt control register	TAA5OVIC	R/W	R/W	-	-
0xFFFFF1D4	Interrupt control register	TAA5CCIC0	R/W	R/W	-	-
0xFFFFF1D6	Interrupt control register	TAA5CCIC1	R/W	R/W	-	-
0xFFFFF1D8	Interrupt control register	TAA6OVIC	R/W	R/W	-	-
0xFFFFF1DA	Interrupt control register	TAA6CCIC0	R/W	R/W	-	-
0xFFFFF1DC	Interrupt control register	TAA6CCIC1	R/W	R/W	-	-
0xFFFFF1DE	Interrupt control register	TAA7OVIC	R/W	R/W	-	-
0xFFFFF1E0	Interrupt control register	TAA7CCIC0	R/W	R/W	-	-
0xFFFFF1E2	Interrupt control register	TAA7CCIC1	R/W	R/W	-	-
0xFFFFF1E4	Interrupt control register	UD6SIC	R/W	R/W	-	-
0xFFFFF1E6	Interrupt control register	CB3RIC	R/W	R/W	-	-
		UD6RIC	R/W	R/W	-	-
0xFFFFF1E8	Interrupt control register	CB3TIC	R/W	R/W	-	-
		UD6TIC	R/W	R/W	-	-
0xFFFFF1EA	Interrupt control register	UD7SIC	R/W	R/W	-	-
0xFFFFF1EC	Interrupt control register	UD7RIC	R/W	R/W	-	-
0xFFFFF1EE	Interrupt control register	UD7TIC	R/W	R/W	-	-
0xFFFFF1F0	Interrupt control register	AD1IC	R/W	R/W	-	-
0xFFFFF1F2	Interrupt control register	C4ERRIC	R/W	R/W	-	-
0xFFFFF1F4	Interrupt control register	C4WUPIC	R/W	R/W	-	-
0xFFFFF1F6	Interrupt control register	C4RECIC	R/W	R/W	-	-
0xFFFFF1F8	Interrupt control register	C4TRXIC	R/W	R/W	-	-
0xFFFFF1FA	In-service priority register	ISPR	R	R	-	-
0xFFFFF1FC	Command register	PRCMD	-	W	-	-
0xFFFFF1FE	Power save control register	PSC	R/W	R/W	-	-
0xFFFFF200	ADC0 mode register 0	ADA0M0	R/W	R/W	-	-
0xFFFFF201	ADC0 mode register 1	ADA0M1	R/W	R/W	-	-
0xFFFFF202	ADC0 channel specification register	ADA0S	R/W	R/W	-	-
0xFFFFF203	ADC0 mode register 2	ADA0M2	R/W	R/W	-	-
0xFFFFF204	ADC0 Power fail comparison mode register	ADA0PFM	R/W	R/W	-	-
0xFFFFF205	ADC0 Power fail comparison threshold value register	ADA0PFT	R/W	R/W	-	-
0xFFFFF20C	ADC0 conversion result register DD	ADA0CRDD	-	-	R	-

Table A-6 Other special function registers (6/17)

Address	Register name	Shortcut	1	8	16	32
0xFFFFF20D	ADC0 conversion result register DDH	ADA0CRDDH	-	R	-	-
0xFFFFF20E	ADC0 conversion result register SS	ADA0CRSS	-	-	R	-
0xFFFFF20F	ADC0 conversion result register SSH	ADA0CRSSH	-	R	-	-
0xFFFFF210	ADC0 conversion result register 0	ADA0CR0	-	-	R	-
0xFFFFF211	ADC0 conversion result register 0H	ADA0CR0H	-	R	-	-
0xFFFFF212	ADC0 conversion result register 1	ADA0CR1	-	-	R	-
0xFFFFF213	ADC0 conversion result register 1H	ADA0CR1H	-	R	-	-
0xFFFFF214	ADC0 conversion result register 2	ADA0CR2	-	-	R	-
0xFFFFF215	ADC0 conversion result register 2H	ADA0CR2H	-	R	-	-
0xFFFFF216	ADC0 conversion result register 3	ADA0CR3	-	-	R	-
0xFFFFF217	ADC0 conversion result register 3H	ADA0CR3H	-	R	-	-
0xFFFFF218	ADC0 conversion result register 4	ADA0CR4	-	-	R	-
0xFFFFF219	ADC0 conversion result register 4H	ADA0CR4H	-	R	-	-
0xFFFFF21A	ADC0 conversion result register 5	ADA0CR5	-	-	R	-
0xFFFFF21B	ADC0 conversion result register 5H	ADA0CR5H	-	R	-	-
0xFFFFF21C	ADC0 conversion result register 6	ADA0CR6	-	-	R	-
0xFFFFF21D	ADC0 conversion result register 6H	ADA0CR6H	-	R	-	-
0xFFFFF21E	ADC0 conversion result register 7	ADA0CR7	-	-	R	-
0xFFFFF21F	ADC0 conversion result register 7H	ADA0CR7H	-	R	-	-
0xFFFFF220	ADC0 conversion result register 8	ADA0CR8	-	-	R	-
0xFFFFF221	ADC0 conversion result register 8H	ADA0CR8H	-	R	-	-
0xFFFFF222	ADC0 conversion result register 9	ADA0CR9	-	-	R	-
0xFFFFF223	ADC0 conversion result register 9H	ADA0CR9H	-	R	-	-
0xFFFFF224	ADC0 conversion result register 10	ADA0CR10	-	-	R	-
0xFFFFF225	ADC0 conversion result register 10H	ADA0CR10H	-	R	-	-
0xFFFFF226	ADC0 conversion result register 11	ADA0CR11	-	-	R	-
0xFFFFF227	ADC0 conversion result register 11H	ADA0CR11H	-	R	-	-
0xFFFFF228	ADC0 conversion result register 12	ADA0CR12	-	-	R	-
0xFFFFF229	ADC0 conversion result register 12H	ADA0CR12H	-	R	-	-
0xFFFFF22A	ADC0 conversion result register 13	ADA0CR13	-	-	R	-
0xFFFFF22B	ADC0 conversion result register 13H	ADA0CR13H	-	R	-	-
0xFFFFF22C	ADC0 conversion result register 14	ADA0CR14	-	-	R	-
0xFFFFF22D	ADC0 conversion result register 14H	ADA0CR14H	-	R	-	-
0xFFFFF22E	ADC0 conversion result register 15	ADA0CR15	-	-	R	-
0xFFFFF22F	ADC0 conversion result register 15H	ADA0CR15H	-	R	-	-
0xFFFFF230	ADC0 conversion result register 16	ADA0CR16	-	-	R	-
0xFFFFF231	ADC0 conversion result register 16H	ADA0CR16H	-	R	-	-
0xFFFFF232	ADC0 conversion result register 17	ADA0CR17	-	-	R	-
0xFFFFF233	ADC0 conversion result register 17H	ADA0CR17H	-	R	-	-
0xFFFFF234	ADC0 conversion result register 18	ADA0CR18	-	-	R	-
0xFFFFF235	ADC0 conversion result register 18H	ADA0CR18H	-	R	-	-

Table A-6 Other special function registers (7/17)

Address	Register name	Shortcut	1	8	16	32
0xFFFFF236	ADC0 conversion result register 19	ADA0CR19	-	-	R	-
0xFFFFF237	ADC0 conversion result register 19H	ADA0CR19H	-	R	-	-
0xFFFFF238	ADC0 conversion result register 20	ADA0CR20	-	-	R	-
0xFFFFF239	ADC0 conversion result register 20H	ADA0CR20H	-	R	-	-
0xFFFFF23A	ADC0 conversion result register 21	ADA0CR21	-	-	R	-
0xFFFFF23B	ADC0 conversion result register 21H	ADA0CR21H	-	R	-	-
0xFFFFF23C	ADC0 conversion result register 22	ADA0CR22	-	-	R	-
0xFFFFF23D	ADC0 conversion result register 22H	ADA0CR22H	-	R	-	-
0xFFFFF23E	ADC0 conversion result register 23	ADA0CR23	-	-	R	-
0xFFFFF240	ADC1 mode register 0	ADA1M0	R/W	R/W	-	-
0xFFFFF241	ADC1 mode register 1	ADA1M1	R/W	R/W	-	-
0xFFFFF242	ADC1 channel specification register	ADA1S	R/W	R/W	-	-
0xFFFFF243	ADC1 mode register 2	ADA1M2	R/W	R/W	-	-
0xFFFFF244	ADC1 Power fail comparison mode register	ADA1PFM	R/W	R/W	-	-
0xFFFFF245	ADC1 Power fail comparison threshold value register	ADA1PFT	R/W	R/W	-	-
0xFFFFF24C	ADC1 conversion result register DD	ADA1CRDD	-	-	R	-
0xFFFFF24D	ADC1 conversion result register DDH	ADA1CRDDH	-	R	-	-
0xFFFFF24E	ADC1 conversion result register SS	ADA1CRSS	-	-	R	-
0xFFFFF24F	ADC1 conversion result register SSH	ADA1CRSSH	-	R	-	-
0xFFFFF250	ADC1 conversion result register 0	ADA1CR0	-	-	R	-
0xFFFFF251	ADC1 conversion result register 0H	ADA1CR0H	-	R	-	-
0xFFFFF252	ADC1 conversion result register 1	ADA1CR1	-	-	R	-
0xFFFFF253	ADC1 conversion result register 1H	ADA1CR1H	-	R	-	-
0xFFFFF254	ADC1 conversion result register 2	ADA1CR2	-	-	R	-
0xFFFFF255	ADC1 conversion result register 2H	ADA1CR2H	-	R	-	-
0xFFFFF256	ADC1 conversion result register 3	ADA1CR3	-	-	R	-
0xFFFFF257	ADC1 conversion result register 3H	ADA1CR3H	-	R	-	-
0xFFFFF258	ADC1 conversion result register 4	ADA1CR4	-	-	R	-
0xFFFFF259	ADC1 conversion result register 4H	ADA1CR4H	-	R	-	-
0xFFFFF25A	ADC1 conversion result register 5	ADA1CR5	-	-	R	-
0xFFFFF25B	ADC1 conversion result register 5H	ADA1CR5H	-	R	-	-
0xFFFFF25C	ADC1 conversion result register 6	ADA1CR6	-	-	R	-
0xFFFFF25D	ADC1 conversion result register 6H	ADA1CR6H	-	R	-	-
0xFFFFF25E	ADC1 conversion result register 7	ADA1CR7	-	-	R	-
0xFFFFF25F	ADC1 conversion result register 7H	ADA1CR7H	-	R	-	-
0xFFFFF260	ADC1 conversion result register 8	ADA1CR8	-	-	R	-
0xFFFFF261	ADC1 conversion result register 8H	ADA1CR8H	-	R	-	-
0xFFFFF262	ADC1 conversion result register 9	ADA1CR9	-	-	R	-
0xFFFFF263	ADC1 conversion result register 9H	ADA1CR9H	-	R	-	-
0xFFFFF264	ADC1 conversion result register 10	ADA1CR10	-	-	R	-
0xFFFFF265	ADC1 conversion result register 10H	ADA1CR10H	-	R	-	-

Table A-6 Other special function registers (8/17)

Address	Register name	Shortcut	1	8	16	32
0xFFFFF266	ADC1 conversion result register 11	ADA1CR11	-	-	R	-
0xFFFFF267	ADC1 conversion result register 11H	ADA1CR11H	-	R	-	-
0xFFFFF268	ADC1 conversion result register 12	ADA1CR12	-	-	R	-
0xFFFFF269	ADC1 conversion result register 12H	ADA1CR12H	-	R	-	-
0xFFFFF26A	ADC1 conversion result register 13	ADA1CR13	-	-	R	-
0xFFFFF26B	ADC1 conversion result register 13H	ADA1CR13H	-	R	-	-
0xFFFFF26C	ADC1 conversion result register 14	ADA1CR14	-	-	R	-
0xFFFFF26D	ADC1 conversion result register 14H	ADA1CR14H	-	R	-	-
0xFFFFF26E	ADC1 conversion result register 15	ADA1CR15	-	-	R	-
0xFFFFF26F	ADC1 conversion result register 15H	ADA1CR15H	-	R	-	-
0xFFFFF300	Key return mode register	KRM	R/W	R/W	-	-
0xFFFFF308	Selector control register 0	SELCNT0	R/W	R/W	-	-
0xFFFFF30A	Selector control register 1	SELCNT1	R/W	R/W	-	-
0xFFFFF30C	Selector control register 2	SELCNT2	R/W	R/W	-	-
0xFFFFF30E	Selector control register 3	SELCNT3	R/W	R/W	-	-
0xFFFFF318	Noise elimination control register	NFC	R/W	R/W	-	-
0xFFFFF340	OPS0 clock selection register	OCKS0	R/W	-	-	-
0xFFFFF3F0	SSCG control register	SSCGCTL	R/W	R/W	-	-
0xFFFFF3F1	SSCG frequency control register 1	SFC0	R/W	R/W	-	-
0xFFFFF3F2	SSCG frequency control register 2	SFC1	R/W	R/W	-	-
0xFFFFF3F8	Selector control register 4	SELCNT4	R/W	R/W	-	-
0xFFFFF3FA	Selector control register 5	SELCNT5	R/W	R/W	-	-
0xFFFFF400	Port 0	P0	R/W	R/W	-	-
0xFFFFF402	Port 1	P1	R/W	R/W	-	-
0xFFFFF404	Port 2L	P2L	R/W	R/W	-	-
0xFFFFF405	Port 2H	P2H	R/W	R/W	-	-
0xFFFFF406	Port 3	P3	-	-	R/W	-
0xFFFFF406	Port 3L	P3L	R/W	R/W	-	-
0xFFFFF407	Port 3H	P3H	R/W	R/W	-	-
0xFFFFF408	Port 4	P4	R/W	R/W	-	-
0xFFFFF40A	Port 5	P5	R/W	R/W	-	-
0xFFFFF40C	Port 6	P6	-	-	R/W	-
0xFFFFF40C	Port 6L	P6L	R/W	R/W	-	-
0xFFFFF40D	Port 6H	P6H	R/W	R/W	-	-
0xFFFFF40E	Port 7L	P7L	R/W	R/W	-	-
0xFFFFF40F	Port 7H	P7H	R/W	R/W	-	-
0xFFFFF410	Port 8	P8	R/W	R/W	-	-
0xFFFFF412	Port 9	P9	-	-	R/W	-
0xFFFFF412	Port 9L	P9L	R/W	R/W	-	-
0xFFFFF413	Port 9H	P9H	R/W	R/W	-	-
0xFFFFF418	Port 12	P12	R/W	R/W	-	-

Table A-6 Other special function registers (9/17)

Address	Register name	Shortcut	1	8	16	32
0xFFFFF41E	Port 15	P15	R/W	R/W	-	-
0xFFFFF420	Port mode register 0	PM0	R/W	R/W	-	-
0xFFFFF422	Port mode register 1	PM1	R/W	R/W	-	-
0xFFFFF424	Port mode register 2L	PM2L	R/W	R/W	-	-
0xFFFFF425	Port mode register2H	PM2H	R/W	R/W	-	-
0xFFFFF426	Port mode register 3	PM3	-	-	R/W	-
0xFFFFF426	Port mode register 3L	PM3L	R/W	R/W	-	-
0xFFFFF427	Port mode register3H	PM3H	R/W	R/W	-	-
0xFFFFF428	Port mode register4	PM4	R/W	R/W	-	-
0xFFFFF42A	Port mode register5	PM5	R/W	R/W	-	-
0xFFFFF42C	Port mode register6	PM6	-	-	R/W	-
0xFFFFF42C	Port mode register6L	PM6L	R/W	R/W	-	-
0xFFFFF42D	Port mode register6H	PM6H	R/W	R/W	-	-
0xFFFFF42E	Port mode register7L	PM7L	R/W	R/W	-	-
0xFFFFF42F	Port mode register7H	PM7H	R/W	R/W	-	-
0xFFFFF430	Port mode register8	PM8	R/W	R/W	-	-
0xFFFFF432	Port mode register9	PM9	-	-	R/W	-
0xFFFFF432	Port mode register9L	PM9L	R/W	R/W	-	-
0xFFFFF433	Port mode register9H	PM9H	R/W	R/W	-	-
0xFFFFF438	Port mode register12	PM12	R/W	R/W	-	-
0xFFFFF43E	Port mode register15	PM15	R/W	R/W	-	-
0xFFFFF440	Port mode control register0	PMC0	R/W	R/W	-	-
0xFFFFF442	Port mode control register1	PMC1	R/W	R/W	-	-
0xFFFFF444	Port mode control register2L	PMC2L	R/W	R/W	-	-
0xFFFFF445	Port mode control register2H	PMC2H	R/W	R/W	-	-
0xFFFFF446	Port mode control register3	PMC3	-	-	R/W	-
0xFFFFF446	Port mode control register3L	PMC3L	R/W	R/W	-	-
0xFFFFF447	Port mode control register3H	PMC3H	R/W	R/W	-	-
0xFFFFF448	Port mode control register4	PMC4	R/W	R/W	-	-
0xFFFFF44A	Port mode control register5	PMC5	R/W	R/W	-	-
0xFFFFF44C	Port mode control register6	PMC6	-	-	R/W	-
0xFFFFF44C	Port mode control register6L	PMC6L	R/W	R/W	-	-
0xFFFFF44D	Port mode control register6H	PMC6H	R/W	R/W	-	-
0xFFFFF44E	Port mode control register7L	PMC7L	R/W	R/W	-	-
0xFFFFF44F	Port mode control register7H	PMC7H	R/W	R/W	-	-
0xFFFFF450	Port mode control register8	PMC8	R/W	R/W	-	-
0xFFFFF452	Port mode control register9	PMC9	-	-	R/W	-
0xFFFFF452	Port mode control register9L	PMC9L	R/W	R/W	-	-
0xFFFFF453	Port mode control register9H	PMC9H	R/W	R/W	-	-
0xFFFFF458	Port mode control register12	PMC12	R/W	R/W	-	-
0xFFFFF45E	Port mode control register15	PMC15	R/W	R/W	-	-

Table A-6 Other special function registers (10/17)

Address	Register name	Shortcut	1	8	16	32
0xFFFFF460	Port function control register0	PFC0	R/W	R/W	-	-
0xFFFFF466	Port function control register3L	PFC3L	R/W	R/W	-	-
0xFFFFF468	Port function control register4	PFC4	R/W	R/W	-	-
0xFFFFF46A	Port function control register5	PFC5	R/W	R/W	-	-
0xFFFFF46C	Port function control register6	PFC6	-	-	R/W	-
0xFFFFF46C	Port function control register6L	PFC6L	R/W	R/W	-	-
0xFFFFF46D	Port function control register6H	PFC6H	R/W	R/W	-	-
0xFFFFF472	Port function control register9	PFC9	-	-	R/W	-
0xFFFFF472	Port function control register9L	PFC9L	R/W	R/W	-	-
0xFFFFF473	Port function control register9H	PFC9H	R/W	R/W	-	-
0xFFFFF47E	Port function control register15	PFC15	R/W	R/W	-	-
0xFFFFF484	Data wait control register	DWC0	-	-	R/W	-
0xFFFFF488	Address wait control register	AWC	-	-	R/W	-
0xFFFFF48A	Bus cycle control register	BCC	-	-	R/W	-
0xFFFFF540	TAB00	TAB0CTL0	R/W	R/W	-	-
0xFFFFF541	TAB01	TAB0CTL1	R/W	R/W	-	-
0xFFFFF542	TAB0 I/O control register 0	TAB0IOC0	R/W	R/W	-	-
0xFFFFF543	TAB0 I/O control register 1	TAB0IOC1	R/W	R/W	-	-
0xFFFFF544	TAB0 I/O control register 2	TAB0IOC2	R/W	R/W	-	-
0xFFFFF545	TAB0 option register 0	TAB0OPT0	R/W	R/W	-	-
0xFFFFF546	TAB0 capture/compare register 0	TAB0CCR0	-	-	R/W	-
0xFFFFF548	TAB0 capture/compare register 1	TAB0CCR1	-	-	R/W	-
0xFFFFF54A	TAB0 capture/compare register 2	TAB0CCR2	-	-	R/W	-
0xFFFFF54C	TAB0 capture/compare register 3	TAB0CCR3	-	-	R/W	-
0xFFFFF54E	TAB0 counter read buffer register	TAB0CNT	-	-	R	-
0xFFFFF550	TAB0 I/O control register 4	TAB0IOC4	R/W	R/W	-	-
0xFFFFF560	TAB 0 option register 1	TAB0OPT1	R/W	R/W	-	-
0xFFFFF561	TAB0 option register 2	TAB0OPT2	R/W	R/W	-	-
0xFFFFF562	TAB0 I/O control register 3	TAB0IOC3	R/W	R/W	-	-
0xFFFFF563	TAB0 option register 3	TAB0OPT3	R/W	R/W	-	-
0xFFFFF564	TAB0 dead time compare register	TAB0DTC	-	-	R/W	-
0xFFFFF570	Hi-Z control function control register 0	HZA0CTL0	R/W	R/W	-	-
0xFFFFF571	Hi-Z control function control register1	HZA0CTL1	R/W	R/W	-	-
0xFFFFF590	TAA0 control register 0	TAA0CTL0	R/W	R/W	-	-
0xFFFFF591	TAA0 control register 1	TAA0CTL1	R/W	R/W	-	-
0xFFFFF592	TAA0 I/O control register 0	TAA0IOC0	R/W	R/W	-	-
0xFFFFF593	TAA0 I/O control register 1	TAA0IOC1	R/W	R/W	-	-
0xFFFFF594	TAA0 I/O control register 2	TAA0IOC2	R/W	R/W	-	-
0xFFFFF595	TAA0 option register 0	TAA0OPT0	R/W	R/W	-	-
0xFFFFF596	TAA0 capture/compare register 0	TAA0CCR0	-	-	R/W	-
0xFFFFF598	TAA0 capture/compare register 1	TAA0CCR1	-	-	R/W	-

Table A-6 Other special function registers (11/17)

Address	Register name	Shortcut	1	8	16	32
0xFFFFF59A	TAA0 counter read buffer register	TAA0CNT	-	-	R	-
0xFFFFF59C	TAA0 I/O control register 4	TAA0IOC4	R/W	R/W	-	-
0xFFFFF5A0	TAA1 control register 0	TAA1CTL0	R/W	R/W	-	-
0xFFFFF5A1	TAA1 control register 1	TAA1CTL1	R/W	R/W	-	-
0xFFFFF5A2	TAA1 I/O control register 0	TAA1IOC0	R/W	R/W	-	-
0xFFFFF5A3	TAA1 I/O control register 1	TAA1IOC1	R/W	R/W	-	-
0xFFFFF5A4	TAA1 I/O control register 2	TAA1IOC2	R/W	R/W	-	-
0xFFFFF5A5	TAA1 option register 0	TAA1OPT0	R/W	R/W	-	-
0xFFFFF5A6	TAA1 capture/compare register 0	TAA1CCR0	-	-	R/W	-
0xFFFFF5A8	TAA1 capture/compare register 1	TAA1CCR1	-	-	R/W	-
0xFFFFF5AA	TAA1 counter read buffer register	TAA1CNT	-	-	R	-
0xFFFFF5AC	TAA1 I/O control register 4	TAA1IOC4	R/W	R/W	-	-
0xFFFFF5AD	TAA1 option register 1	TAA1OPT1	R/W	R/W	-	-
0xFFFFF5B0	TAA2 control register 0	TAA2CTL0	R/W	R/W	-	-
0xFFFFF5B1	TAA2 control register 1	TAA2CTL1	R/W	R/W	-	-
0xFFFFF5B2	TAA2 I/O control register 0	TAA2IOC0	R/W	R/W	-	-
0xFFFFF5B3	TAA2 I/O control register 1	TAA2IOC1	R/W	R/W	-	-
0xFFFFF5B4	TAA2 I/O control register 2	TAA2IOC2	R/W	R/W	-	-
0xFFFFF5B5	TAA2 option register 0	TAA2OPT0	R/W	R/W	-	-
0xFFFFF5B6	TAA2 capture/compare register 0	TAA2CCR0	-	-	R/W	-
0xFFFFF5B8	TAA2 capture/compare register 1	TAA2CCR1	-	-	R/W	-
0xFFFFF5BA	TAA2 counter read buffer register	TAA2CNT	-	-	R	-
0xFFFFF5BC	TAA2 I/O control register 4	TAA2IOC4	R/W	R/W	-	-
0xFFFFF5C0	TAA3 control register 0	TAA3CTL0	R/W	R/W	-	-
0xFFFFF5C1	TAA3 control register 1	TAA3CTL1	R/W	R/W	-	-
0xFFFFF5C2	TAA3 I/O control register 0	TAA3IOC0	R/W	R/W	-	-
0xFFFFF5C3	TAA3 I/O control register 1	TAA3IOC1	R/W	R/W	-	-
0xFFFFF5C4	TAA3 I/O control register 2	TAA3IOC2	R/W	R/W	-	-
0xFFFFF5C5	TAA3 option register 0	TAA3OPT0	R/W	R/W	-	-
0xFFFFF5C6	TAA3 capture/compare register 0	TAA3CCR0	-	-	R/W	-
0xFFFFF5C8	TAA3 capture/compare register 1	TAA3CCR1	-	-	R/W	-
0xFFFFF5CA	TAA3 counter read buffer register	TAA3CNT	-	-	R	-
0xFFFFF5CC	TAA3 I/O control register 4	TAA3IOC4	R/W	R/W	-	-
0xFFFFF5CD	TAA3 option register 1	TAA3OPT1	R/W	R/W	-	-
0xFFFFF5D0	TAA4 control register 0	TAA4CTL0	R/W	R/W	-	-
0xFFFFF5D1	TAA4 control register 1	TAA4CTL1	R/W	R/W	-	-
0xFFFFF5D2	TAA4 I/O control register 0	TAA4IOC0	R/W	R/W	-	-
0xFFFFF5D3	TAA4 I/O control register 1	TAA4IOC1	R/W	R/W	-	-
0xFFFFF5D4	TAA4 I/O control register 2	TAA4IOC2	R/W	R/W	-	-
0xFFFFF5D5	TAA4 option register 0	TAA4OPT0	R/W	R/W	-	-
0xFFFFF5D6	TAA4 capture/compare register 0	TAA4CCR0	-	-	R/W	-

Table A-6 Other special function registers (12/17)

Address	Register name	Shortcut	1	8	16	32
0xFFFFF5D8	TAA4 capture/compare register 1	TAA4CCR1	-	-	R/W	-
0xFFFFF5DA	TAA4 counter read buffer register	TAA4CNT	-	-	R	-
0xFFFFF5DC	TAA4 I/O control register 4	TAA4IOC4	R/W	R/W	-	-
0xFFFFF5E0	TAA5 control register 0	TAA5CTL0	R/W	R/W	-	-
0xFFFFF5E1	TAA5 control register 1	TAA5CTL1	R/W	R/W	-	-
0xFFFFF5E2	TAA5 I/O control register 0	TAA5IOC0	R/W	R/W	-	-
0xFFFFF5E3	TAA5 I/O control register 1	TAA5IOC1	R/W	R/W	-	-
0xFFFFF5E4	TAA5 I/O control register 2	TAA5IOC2	R/W	R/W	-	-
0xFFFFF5E5	TAA5 option register 0	TAA5OPT0	R/W	R/W	-	-
0xFFFFF5E6	TAA5 capture/compare register 0	TAA5CCR0	-	-	R/W	-
0xFFFFF5E8	TAA5 capture/compare register 1	TAA5CCR1	-	-	R/W	-
0xFFFFF5EA	TAA5 counter read buffer register	TAA5CNT	-	-	R	-
0xFFFFF5EC	TAA5 I/O control register 4	TAA5IOC4	R/W	R/W	-	-
0xFFFFF5F0	TAA6 control register 0	TAA6CTL0	R/W	R/W	-	-
0xFFFFF5F1	TAA6 control register 1	TAA6CTL1	R/W	R/W	-	-
0xFFFFF5F2	TAA6 I/O control register 0	TAA6IOC0	R/W	R/W	-	-
0xFFFFF5F3	TAA6 I/O control register 1	TAA6IOC1	R/W	R/W	-	-
0xFFFFF5F4	TAA6 I/O control register 2	TAA6IOC2	R/W	R/W	-	-
0xFFFFF5F5	TAA6 option register 0	TAA6OPT0	R/W	R/W	-	-
0xFFFFF5F6	TAA6 capture/compare register 0	TAA6CCR0	-	-	R/W	-
0xFFFFF5F8	TAA6 capture/compare register 1	TAA6CCR1	-	-	R/W	-
0xFFFFF5FA	TAA6 counter read buffer register	TAA6CNT	-	-	R	-
0xFFFFF5FC	TAA6 I/O control register 4	TAA6IOC4	R/W	R/W	-	-
0xFFFFF5FD	TAA6 option register 1	TAA6OPT1	R/W	R/W	-	-
0xFFFFF600	TAA7 control register 0	TAA7CTL0	R/W	R/W	-	-
0xFFFFF601	TAA7 control register 1	TAA7CTL1	R/W	R/W	-	-
0xFFFFF602	TAA7 I/O control register 0	TAA7IOC0	R/W	R/W	-	-
0xFFFFF603	TAA7 I/O control register 1	TAA7IOC1	R/W	R/W	-	-
0xFFFFF604	TAA7 I/O control register 2	TAA7IOC2	R/W	R/W	-	-
0xFFFFF605	TAA7 option register 0	TAA7OPT0	R/W	R/W	-	-
0xFFFFF606	TAA7 capture/compare register 0	TAA7CCR0	-	-	R/W	-
0xFFFFF608	TAA7 capture/compare register 1	TAA7CCR1	-	-	R/W	-
0xFFFFF60A	TAA7 counter read buffer register	TAA7CNT	-	-	R	-
0xFFFFF60C	TAA7 I/O control register 4	TAA7IOC4	R/W	R/W	-	-
0xFFFFF610	TAB1 timer control register0	TAB1CTL0	R/W	R/W	-	-
0xFFFFF611	TAB1 timer control register1	TAB1CTL1	R/W	R/W	-	-
0xFFFFF612	TAB1 I/O control register 0	TAB1IOC0	R/W	R/W	-	-
0xFFFFF613	TAB1 I/O control register 1	TAB1IOC1	R/W	R/W	-	-
0xFFFFF614	TAB1 I/O control register 2	TAB1IOC2	R/W	R/W	-	-
0xFFFFF615	TAB1 option register 0	TAB1OPT0	R/W	R/W	-	-
0xFFFFF616	TAB1 capture/compare register 0	TAB1CCR0	-	-	R/W	-

Table A-6 Other special function registers (13/17)

Address	Register name	Shortcut	1	8	16	32
0xFFFFF618	TAB1 capture/compare register 1	TAB1CCR1	-	-	R/W	-
0xFFFFF61A	TAB1 capture/compare register 2	TAB1CCR2	-	-	R/W	-
0xFFFFF61C	TAB1 capture/compare register 3	TAB1CCR3	-	-	R/W	-
0xFFFFF61E	TAB1 counter read buffer register	TAB1CNT	-	-	R	-
0xFFFFF620	TAB2 timer control register0	TAB2CTL0	R/W	R/W	-	-
0xFFFFF621	TAB2 timer control register1	TAB2CTL1	R/W	R/W	-	-
0xFFFFF622	TAB2 I/O control register 0	TAB2IOC0	R/W	R/W	-	-
0xFFFFF623	TAB2 I/O control register 1	TAB2IOC1	R/W	R/W	-	-
0xFFFFF624	TAB2 I/O control register 2	TAB2IOC2	R/W	R/W	-	-
0xFFFFF625	TAB2 option register 0	TAB2OPT0	R/W	R/W	-	-
0xFFFFF626	TAB2 capture/compare register 0	TAB2CCR0	-	-	R/W	-
0xFFFFF628	TAB2 capture/compare register 1	TAB2CCR1	-	-	R/W	-
0xFFFFF62A	TAB2 capture/compare register 2	TAB2CCR2	-	-	R/W	-
0xFFFFF62C	TAB2 capture/compare register 3	TAB2CCR3	-	-	R/W	-
0xFFFFF62E	TAB2 counter read buffer register	TAB2CNT	-	-	R	-
0xFFFFF660	TAB1 I/O control register 4	TAB1IOC4	R/W	R/W	-	-
0xFFFFF670	TAB2 I/O control register 4	TAB2IOC4	R/W	R/W	-	-
0xFFFFF680	Watch Timer operation mode register	WTM	R/W	R/W	-	-
0xFFFFF690	TMM0 timer control register0	TM0CTL0	R/W	R/W	-	-
0xFFFFF694	TMM0 compare register 0	TM0CMP0	-	-	R/W	-
0xFFFFF6C0	Oscillation stabilization time select register	OSTS	-	R/W	-	-
0xFFFFF6C1	PLL lockup time specification register	PLLS	-	R/W	-	-
0xFFFFF6C2	Oscillation stabilization timer status register	OSTC	R	R	-	-
0xFFFFF6D0	Watchdog Timer mode register 2	WDTM2	R/W	R/W	-	-
0xFFFFF6D1	Watchdog Timer enable register	WDTE	-	R/W	-	-
0xFFFFF700	Port 0 function control enhancing register	PFCE0	R/W	R/W	-	-
0xFFFFF706	Port 3 function control enhancing register L	PFCE3L	R/W	R/W	-	-
0xFFFFF708	Port 4 function control enhancing register	PFCE4	R/W	R/W	-	-
0xFFFFF70A	Port 5 function control enhancing register L	PFCE5	R/W	R/W	-	-
0xFFFFF70C	Port 6 function control enhancing register L	PFCE6L	R/W	R/W	-	-
0xFFFFF712	Port 9 function control enhancing register	PFCE9	-	-	R/W	-
0xFFFFF712	Port 9 function control enhancing register L	PFCE9L	R/W	R/W	-	-
0xFFFFF713	Port 9 function control enhancing register H	PFCE9H	R/W	R/W	-	-
0xFFFFF802	System register	SYS	R/W	R/W	-	-
0xFFFFF80c	Internal oscillator mode register	RCM	R/W	R/W	-	-
0xFFFFF810	DMA trigger source register 0	DTFR0	R/W	R/W	-	-
0xFFFFF812	DMA trigger source register 1	DTFR1	R/W	R/W	-	-
0xFFFFF814	DMA trigger source register 2	DTFR2	R/W	R/W	-	-
0xFFFFF816	DMA trigger source register 3	DTFR3	R/W	R/W	-	-
0xFFFFF820	Power save mode register	PSMR	R/W	R/W	-	-
0xFFFFF824	Lock register	LOCKR	R	R	-	-

Table A-6 Other special function registers (14/17)

Address	Register name	Shortcut	1	8	16	32
0xFFFFF828	Processor clock control register	PCC	R/W	R/W	-	-
0xFFFFF82C	PLL control register	PLLCTL	R/W	R/W	-	-
0xFFFFF82E	CPU operating clock status register	CCLS	R	R	-	-
0xFFFFF82F	Programmable clock mode register	PCLM	R/W	R/W	-	-
0xFFFFF860	System clock mode register	MCM	R/W	R/W	-	-
0xFFFFF870	Clock Monitor mode register	CLM	R/W	R/W	-	-
0xFFFFF888	Reset factor flag register	RESF	R/W	R/W	-	-
0xFFFFF890	Low voltage detection register	LVIM	R/W	R/W	-	-
0xFFFFF891	Low voltage detection level selection register	LVIS	-	R/W	-	-
0xFFFFF892	RAM data status register	RAMS	R/W	R/W	-	-
0xFFFFF8B0	BRG0 prescaler mode register	PRSM0	-	R/W	-	-
0xFFFFF8B1	BRG0 prescaler compare register	PRSCM0	-	R/W	-	-
0xFFFFF9FC	On-chip debug mode register	OCDM	R/W	R/W	-	-
0xFFFFF9FE	Peripheral emulation register 1	PEMU1	R/W	R/W	-	-
0xFFFFFA00	UARTD0 control register 0	UD0CTL0	R/W	R/W	-	-
0xFFFFFA01	UARTD0 control register 1	UD0CTL1	-	R/W	-	-
0xFFFFFA02	UARTD0 control register 2	UD0CTL2	-	R/W	-	-
0xFFFFFA03	UARTD0 option control register 0	UD0OPT0	R/W	R/W	-	-
0xFFFFFA04	UARTD0 status register	UD0STR	R/W	R/W	-	-
0xFFFFFA05	UARTD0 option control register 1	UD0OPT1	-	R/W	-	-
0xFFFFFA06	UARTD0 receive data register	UD0RX	-	R	-	-
0xFFFFFA07	UARTD0 transmit data register	UD0TX	-	R/W	-	-
0xFFFFFA10	UARTD1 control register 0	UD1CTL0	R/W	R/W	-	-
0xFFFFFA11	UARTD1 control register 1	UD1CTL1	-	R/W	-	-
0xFFFFFA12	UARTD1 control register 2	UD1CTL2	-	R/W	-	-
0xFFFFFA13	UARTD1 option control register 0	UD1OPT0	R/W	R/W	-	-
0xFFFFFA14	UARTD1 status register	UD1STR	R/W	R/W	-	-
0xFFFFFA15	UARTD1 option control register 1	UD1OPT1	-	R/W	-	-
0xFFFFFA16	UARTD1 receive data register	UD1RX	-	R	-	-
0xFFFFFA17	UARTD1 transmit data register	UD1TX	-	R/W	-	-
0xFFFFFA20	UARTD2 control register 0	UD2CTL0	R/W	R/W	-	-
0xFFFFFA21	UARTD2 control register 1	UD2CTL1	-	R/W	-	-
0xFFFFFA22	UARTD2 control register 2	UD2CTL2	-	R/W	-	-
0xFFFFFA23	UARTD2 option control register 0	UD2OPT0	R/W	R/W	-	-
0xFFFFFA24	UARTD2 status register	UD2STR	R/W	R/W	-	-
0xFFFFFA25	UARTD2 option control register 1	UD2OPT1	-	R/W	-	-
0xFFFFFA26	UARTD2 receive data register	UD2RX	-	R	-	-
0xFFFFFA27	UARTD2 transmit data register	UD2TX	-	R/W	-	-
0xFFFFFA30	UARTD3 control register 0	UD3CTL0	R/W	R/W	-	-
0xFFFFFA31	UARTD3 control register 1	UD3CTL1	-	R/W	-	-
0xFFFFFA32	UARTD3 control register 2	UD3CTL2	-	R/W	-	-

Table A-6 Other special function registers (15/17)

Address	Register name	Shortcut	1	8	16	32
0xFFFFFA33	UARTD3 option control register 0	UD3OPT0	R/W	R/W	-	-
0xFFFFFA34	UARTD3 status register	UD3STR	R/W	R/W	-	-
0xFFFFFA35	UARTD3 option control register 1	UD3OPT1	-	R/W	-	-
0xFFFFFA36	UARTD3 receive data register	UD3RX	-	R	-	-
0xFFFFFA37	UARTD3 transmit data register	UD3TX	-	R/W	-	-
0xFFFFFA40	UARTD4 control register 0	UD4CTL0	R/W	R/W	-	-
0xFFFFFA41	UARTD4 control register 1	UD4CTL1	-	R/W	-	-
0xFFFFFA42	UARTD4 control register 2	UD4CTL2	-	R/W	-	-
0xFFFFFA43	UARTD4 option control register 0	UD4OPT0	R/W	R/W	-	-
0xFFFFFA44	UARTD4 status register	UD4STR	R/W	R/W	-	-
0xFFFFFA45	UARTD4 option control register 1	UD4OPT1	-	R/W	-	-
0xFFFFFA46	UARTD4 Receive data register	UD4RX	-	R	-	-
0xFFFFFA47	UARTD4 transmit data register	UD4TX	-	R/W	-	-
0xFFFFFA50	UARTD5 control register 0	UD5CTL0	R/W	R/W	-	-
0xFFFFFA51	UARTD5 control register 1	UD5CTL1	-	R/W	-	-
0xFFFFFA52	UARTD5 control register 2	UD5CTL2	-	R/W	-	-
0xFFFFFA53	UARTD5 option control register 0	UD5OPT0	R/W	R/W	-	-
0xFFFFFA54	UARTD5 status register	UD5STR	R/W	R/W	-	-
0xFFFFFA55	UARTD5 option control register 1	UD5OPT1	-	R/W	-	-
0xFFFFFA56	UARTD5 Receive data register	UD5RX	-	R	-	-
0xFFFFFA57	UARTD5 transmit data register	UD5TX	-	R/W	-	-
0xFFFFFA60	UARTD6 control register 0	UD6CTL0	R/W	R/W	-	-
0xFFFFFA61	UARTD6 control register 1	UD6CTL1	-	R/W	-	-
0xFFFFFA62	UARTD6 control register 2	UD6CTL2	-	R/W	-	-
0xFFFFFA63	UARTD6 option control register 0	UD6OPT0	R/W	R/W	-	-
0xFFFFFA64	UARTD6 status register	UD6STR	R/W	R/W	-	-
0xFFFFFA65	UARTD6 option control register 1	UD6OPT1	-	R/W	-	-
0xFFFFFA66	UARTD6 Receive data register	UD6RX	-	R	-	-
0xFFFFFA67	UARTD6 transmit data register	UD6TX	-	R/W	-	-
0xFFFFFA70	UARTD7 control register 0	UD7CTL0	R/W	R/W	-	-
0xFFFFFA71	UARTD7 control register 1	UD7CTL1	-	R/W	-	-
0xFFFFFA72	UARTD7 control register 2	UD7CTL2	-	R/W	-	-
0xFFFFFA73	UARTD7 option control register 0	UD7OPT0	R/W	R/W	-	-
0xFFFFFA74	UARTD7 status register	UD7STR	R/W	R/W	-	-
0xFFFFFA75	UARTD7 option control register 1	UD7OPT1	-	R/W	-	-
0xFFFFFA76	UARTD7 Receive data register	UD7RX	-	R	-	-
0xFFFFFA77	UARTD7 transmit data register	UD7TX	-	R/W	-	-
0xFFFFFC00	External interrupt falling edge specification register 0	INTF0	R/W	R/W	-	-
0xFFFFFC02	External interrupt falling edge specification register 1	INTF1	R/W	R/W	-	-
0xFFFFFC06	External interrupt falling edge specification register 3	INTF3	-	-	R/W	-
0xFFFFFC06	External interrupt falling edge specification register 3L	INTF3L	R/W	R/W	-	-

Table A-6 Other special function registers (16/17)

Address	Register name	Shortcut	1	8	16	32
0xFFFFFC07	External interrupt falling edge specification register 3H	INTF3H	R/W	R/W	-	-
0xFFFFFC08	External interrupt falling edge specification register 4	INTF4	R/W	R/W	-	-
0xFFFFFC0C	External interrupt falling edge specification register 6	INTF6	-	-	R/W	-
0xFFFFFC0C	External interrupt falling edge specification register 6L	INTF6L	R/W	R/W	-	-
0xFFFFFC0D	External interrupt falling edge specification register 6H	INTF6H	R/W	R/W	-	-
0xFFFFFC10	External interrupt falling edge specification register 8	INTF8	R/W	R/W	-	-
0xFFFFFC13	External interrupt falling edge specification register 9H	INTF9H	R/W	R/W	-	-
0xFFFFFC20	External interrupt rising edge specification register 0	INTR0	R/W	R/W	-	-
0xFFFFFC22	External interrupt rising edge specification register 1	INTR1	R/W	R/W	-	-
0xFFFFFC26	External interrupt rising edge specification register 3	INTR3	-	-	R/W	-
0xFFFFFC26	External interrupt rising edge specification register 3L	INTR3L	R/W	R/W	-	-
0xFFFFFC27	External interrupt rising edge specification register 3H	INTR3H	R/W	R/W	-	-
0xFFFFFC28	External interrupt rising edge specification register 4	INTR4	R/W	R/W	-	-
0xFFFFFC2C	External interrupt rising edge specification register 6	INTR6	-	-	R/W	-
0xFFFFFC2C	External interrupt rising edge specification register 6L	INTR6L	R/W	R/W	-	-
0xFFFFFC2D	External interrupt rising edge specification register 6H	INTR6H	R/W	R/W	-	-
0xFFFFFC30	External interrupt rising edge specification register 8	INTR8	R/W	R/W	-	-
0xFFFFFC33	External interrupt rising edge specification register 9H	INTR9H	R/W	R/W	-	-
0xFFFFFC40	Pull-up resistor option register 0	PU0	R/W	R/W	-	-
0xFFFFFC42	Pull-up resistor option register 1	PU1	R/W	R/W	-	-
0xFFFFFC46	Pull-up resistor option register 3	PU3	-	-	R/W	-
0xFFFFFC46	Pull-up resistor option register 3L	PU3L	R/W	R/W	-	-
0xFFFFFC47	Pull-up resistor option register 3H	PU3H	R/W	R/W	-	-
0xFFFFFC48	Pull-up resistor option register 4	PU4	R/W	R/W	-	-
0xFFFFFC4A	Pull-up resistor option register 5	PU5	R/W	R/W	-	-
0xFFFFFC4C	Pull-up resistor option register 6	PU6	-	-	R/W	-
0xFFFFFC4C	Pull-up resistor option register 6L	PU6L	R/W	R/W	-	-
0xFFFFFC4D	Pull-up resistor option register 6H	PU6H	R/W	R/W	-	-
0xFFFFFC50	Pull-up resistor option register 8	PU8	R/W	R/W	-	-
0xFFFFFC52	Pull-up resistor option register 9	PU9	-	-	R/W	-
0xFFFFFC52	Pull-up resistor option register 9L	PU9L	R/W	R/W	-	-
0xFFFFFC53	Pull-up resistor option register 9H	PU9H	R/W	R/W	-	-
0xFFFFFC5E	Pull-up resistor option register 15	PU15	R/W	R/W	-	-
0xFFFFFC73	Port 9 function control register H	PF9H	R/W	R/W	-	-
0xFFFFFCCA	Product selection code resistor H	PRDSELH	R/W	-	R/W	-
0xFFFFFCF8	Data flash control register	DFLCTL	R/W	R/W	-	-
0xFFFFFD00	CSIB0 control register 0	CB0CTL0	R/W	R/W	-	-
0xFFFFFD01	CSIB0 control register 1	CB0CTL1	R/W	R/W	-	-
0xFFFFFD02	CSIB0 control register 2	CB0CTL2	-	R/W	-	-
0xFFFFFD03	CSIB0 status register	CB0STR	R/W	R/W	-	-
0xFFFFFD04	CSIB0 receive data register	CB0RX	-	-	R	-

Table A-6 Other special function registers (17/17)

Address	Register name	Shortcut	1	8	16	32
0xFFFFFD04	CSIB0 receive data register L	CB0RXL	-	R	-	-
0xFFFFFD06	CSIB0 transmit data register	CB0TX	-	-	R/W	-
0xFFFFFD06	CSIB0 transmit data register L	CB0TXL	-	R/W	-	-
0xFFFFFD10	CSIB1 control register 0	CB1CTL0	R/W	R/W	-	-
0xFFFFFD11	CSIB1 control register 1	CB1CTL1	R/W	R/W	-	-
0xFFFFFD12	CSIB1 control register 2	CB1CTL2	-	R/W	-	-
0xFFFFFD13	CSIB1 status register	CB1STR	R/W	R/W	-	-
0xFFFFFD14	CSIB1 receive data register	CB1RX	-	-	R	-
0xFFFFFD14	CSIB1 receive data register L	CB1RXL	-	R	-	-
0xFFFFFD16	CSIB1 transmit data register	CB1TX	-	-	R/W	-
0xFFFFFD16	CSIB1 transmit data register L	CB1TXL	-	R/W	-	-
0xFFFFFD20	CSIB2 control register 0	CB2CTL0	R/W	R/W	-	-
0xFFFFFD21	CSIB2 control register 1	CB2CTL1	R/W	R/W	-	-
0xFFFFFD22	CSIB2 control register 2	CB2CTL2	-	R/W	-	-
0xFFFFFD23	CSIB2 status register	CB2STR	R/W	R/W	-	-
0xFFFFFD24	CSIB2 receive data register	CB2RX	-	-	R	-
0xFFFFFD24	CSIB2 receive data register L	CB2RXL	-	R	-	-
0xFFFFFD26	CSIB2 transmit data register	CB2TX	-	-	R/W	-
0xFFFFFD26	CSIB2 transmit data register L	CB2TXL	-	R/W	-	-
0xFFFFFD30	CSIB3 control register 0	CB3CTL0	R/W	R/W	-	-
0xFFFFFD31	CSIB3 control register 1	CB3CTL1	R/W	R/W	-	-
0xFFFFFD32	CSIB3 control register 2	CB3CTL2	-	R/W	-	-
0xFFFFFD33	CSIB3 status register	CB3STR	R/W	R/W	-	-
0xFFFFFD34	CSIB3 receive data register	CB3RX	-	-	R	-
0xFFFFFD34	CSIB3 receive data register L	CB3RXL	-	R	-	-
0xFFFFFD36	CSIB3 transmit data register	CB3TX	-	-	R/W	-
0xFFFFFD36	CSIB3 transmit data register L	CB3TXL	-	R/W	-	-
0xFFFFFD80	IIC0 shift register	IIC0	-	R/W	-	-
0xFFFFFD82	IIC0 control register	IICC0	R/W	R/W	-	-
0xFFFFFD83	IIC0 slave address register	SVA0	-	R/W	-	-
0xFFFFFD84	IIC0 clock selection register	IICCL0	R/W	R/W	-	-
0xFFFFFD85	IIC0 function expansion register	IICX0	R/W	R/W	-	-
0xFFFFFD86	IIC0 state register	IICS0	R	R	-	-
0xFFFFFD8A	IIC0 flag register	IICF0	R/W	R/W	-	-

Appendix B Registers Access Times

This chapter provides formulas to calculate the access time to registers, which are accessed via the peripheral I/O areas.

All accesses to the peripheral I/O areas are passed over to the NPB bus via the VSB - NPB bus bridge BBR. Read and write access times to registers via the NPB depend on the register, the system clock VBCLK and the setting of the VSWC register.

The CPU operation during an access to a register via the NPB depends also on the kind of peripheral I/O area:

- Fixed peripheral I/O area
  During a read or write access the CPU operation stops until the access via the NPB is completed.
- Programmable peripheral I/O area
  During a read access the CPU operation stops until the read access via the NPB is completed.
  During a write access the CPU operation continues operation, provided any preceded NPB access is already finished. If a preceded NPB access is still ongoing the CPU stops until this access is finished and the NPB is cleared.

The following formulas are given to calculate the access times T_a, when the CPU reads from or writes to special function registers via the NPB bus.

The access time depends

- on the CPU system clock frequency f_{VBCLK}
- on the setting of the internal peripheral function wait control register VSWC, which determines the address set up wait SUWL = VSWC.SUWL and data wait VSWL = VSWC.VSWL (refer to *“VSWC - Internal peripheral function wait control register” on page 354* for the correct values for a certain CPU system clock VBCLK)
- for some registers on the clock frequency applied to the module

Note “ru[...]” in the formulas mean “round up” the calculated value of the term in squared brackets.

All formulas calculate the maximum access time.

CPU access For calculating the access times for CPU accesses 1 VBLCK period time 1/ f_{VBCLK} has to be added to the results of the formulas.

DMA access For accesses of the DMA Controller the given formulas calculate the exact values.

B.1 Timer AA

Register **TAAnCCRm**

Access R

Formula

- if TAAnCTL0.TAAnCE = 0:

$$T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$$

- if TAAnCTL0.TAAnCE = 1:

$$T_a = (SUWL + 2 \cdot VSWL + 5) \cdot \frac{1}{f_{VBCLK}}$$

Access W

Formula

- if TAAnCTL0.TAAnCE = 0:

$$T_a = SUWL + VSWL + 3 \cdot \frac{1}{f_{VBCLK}}$$

- if TAAnCTL0.TAAnCE = 1:
  - continuous write

$$T_a = \left\{SUWL + VSWL + 3 + ru\left[\frac{5 \cdot f_{VBCLK}}{(2 + VSWL) \cdot f_{TAA}} + 1\right] \cdot (2 + VSWL)\right\} \cdot \frac{1}{f_{VBCLK}}$$

  - single write

$$T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$$

Register **TAAnIOC4**

Access R

Formula $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

Access W

Formula

- if TAAnCTL0.TAAnCE = 0:

$$T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$$

- if TAAnCTL0.TAAnCE = 1:
  - continuous write

$$T_a = \left\{SUWL + VSWL + 3 + ru\left[\frac{5 \cdot f_{VBCLK}}{(2 + VSWL) \cdot f_{TAA}} + 1\right] \cdot (2 + VSWL)\right\} \cdot \frac{1}{f_{VBCLK}}$$

  - single write

$$T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$$

Register **TAAnCNT**

Access R

Formula

- if TAAnCTL0.TAAnCE = 0:

$$T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$$

- if TAAnCTL0.TAAnCE = 1:

$$T_a = (SUWL + 2 \cdot VSWL + 5) \cdot \frac{1}{f_{VBCLK}}$$

Register **all other**

Access R/W

Formula $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

f_{TAA} The TAAn input clock can be selected from

- SELCNT2.SELCNT2n = 0: $f_{TAA} = f_{XP1}$
- SELCNT2.SELCNT2n = 1: $f_{TAA} = f_{XP2}$

B.2 Timer AB

Register **TABnCCRm**

Access R

Formula

- if TABnCTL0.TABnCE = 0:

$$T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$$

- if TABnCTL0.TABnCE = 1:

$$T_a = (SUWL + 2 \cdot VSWL + 5) \cdot \frac{1}{f_{VBCLK}}$$

Access W

Formula

- if TABnCTL0.TABnCE = 0:

$$T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$$

- if TABnCTL0.TABnCE = 1:
  - continuous write

$$T_a = \left\{SUWL + VSWL + 3 + ru\left[\frac{5 \cdot f_{VBCLK}}{(2 + VSWL) \cdot f_{XP1}} + 1\right] \cdot (2 + VSWL)\right\} \cdot \frac{1}{f_{VBCLK}}$$

  - single write

$$T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$$

Register **TABnIOC4**

Access R

Formula $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

Access W

Formula

- if TABnCTL0.TABnCE = 0:

$$T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$$

- if TABnCTL0.TABnCE = 1:
  - continuous write

$$T_a = \left\{SUWL + VSWL + 3 + ru\left[\frac{5 \cdot f_{VBCLK}}{(2 + VSWL) \cdot f_{XP1}} + 1\right] \cdot (2 + VSWL)\right\} \cdot \frac{1}{f_{VBCLK}}$$

  - single write

$$T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$$

Register **TABnCNT**

Access R

Formula

- if TABnCTL0.TABnCE = 0:

$$T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$$

- if TABnCTL0.TABnCE = 1:

$$T_a = (SUWL + 2 \cdot VSWL + 5) \cdot \frac{1}{f_{VBCLK}}$$

Register **all other**

Access R/W

Formula $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

B.3 Motor Control Function

Register **TABnIOC3**

Access R

Formula

$$T_a = (\mathrm{SUWL} + \mathrm{VSWL} + 3) \cdot \frac{1}{f_{\mathrm{VBCLK}}}$$

Access W

Formula

- if TABnCTL0.TABnCE = 0:

$$T_a = (\mathrm{SUWL} + \mathrm{VSWL} + 3) \cdot \frac{1}{f_{\mathrm{VBCLK}}}$$

- if TABnCTL0.TABnCE = 1:
  - continuous write

$$T_a = \left\{\mathrm{SUWL} + \mathrm{VSWL} + 3 + \mathrm{ru}\left[\frac{5 \cdot f_{\mathrm{VBCLK}}}{(2 + \mathrm{VSWL}) \cdot f_{\mathrm{XP1}}} + 1\right] \cdot (2 + \mathrm{VSWL})\right\} \cdot \frac{1}{f_{\mathrm{VBCLK}}}$$

  - single write

$$T_a = (\mathrm{SUWL} + \mathrm{VSWL} + 3) \cdot \frac{1}{f_{\mathrm{VBCLK}}}$$

Register **TABnOPT1**

Access R

Formula

$$T_a = (\mathrm{SUWL} + \mathrm{VSWL} + 3) \cdot \frac{1}{f_{\mathrm{VBCLK}}}$$

Access W

Formula

- continuous write

$$T_a = \left\{\mathrm{SUWL} + \mathrm{VSWL} + 3 + \mathrm{ru}\left[\frac{5 \cdot f_{\mathrm{VBCLK}}}{(2 + \mathrm{VSWL}) \cdot f_{\mathrm{TAA}}} + 1\right] \cdot (2 + \mathrm{VSWL})\right\} \cdot \frac{1}{f_{\mathrm{VBCLK}}}$$

- single write

$$T_a = (\mathrm{SUWL} + 2 \cdot \mathrm{VSWL} + 5) \cdot \frac{1}{f_{\mathrm{VBCLK}}}$$

Register **TABnDTC**

Access R

Formula $T_a = (\mathrm{SUWL} + \mathrm{VSWL} + 3) \cdot \frac{1}{f_{\mathrm{VBCLK}}}$

Access W

Formula

- continuous write

$$T_a = \left\{\mathrm{SUWL} + \mathrm{VSWL} + 3 + \mathrm{ru}\left[\frac{5 \cdot f_{\mathrm{VBCLK}}}{(2 + \mathrm{VSWL}) \cdot f_{\mathrm{TAA}}} + 1\right] \cdot (2 + \mathrm{VSWL})\right\} \cdot \frac{1}{f_{\mathrm{VBCLK}}}$$

- single write

$$T_a = (\mathrm{SUWL} + 2 \cdot \mathrm{VSWL} + 5) \cdot \frac{1}{f_{\mathrm{VBCLK}}}$$

Register **all**

Access R/W

Formula $T_a = (\mathrm{SUWL} + \mathrm{VSWL} + 3) \cdot \frac{1}{f_{\mathrm{VBCLK}}}$

B.4 Timer M

Register **all**

Access R/W

Formula $T_a = (\mathrm{SUWL} + \mathrm{VSWL} + 3) \cdot \frac{1}{f_{\mathrm{VBCLK}}}$

B.5 Watchdog Timer 2

Register **WDTM2**

Access W

Formula

- if Watchdog Timer operating:

$$T_a = (SUWL + 4 \cdot VSWL + 9) \cdot \frac{1}{f_{VBCLK}}$$

- if Watchdog Timer stopped:

$$T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$$

Access R

Formula $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

Register **all other**

Access R/W

Formula $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

B.6 A/D Converter

Register **ADA0M0, ADA0CRm, ADA0CRmH, ADA0CRDD, ADA0CRDDH, ADA0CRSS, ADA0CRSSH**

ADA1M0, ADA1CRm, ADA1CRmH, ADA1CRDD, ADA1CRDDH, ADA1CRSS, ADA1CRSSH

Access R

Formula $T_a = \left\{SUWL + VSWL + 3 + ru\left[\frac{2 \cdot f_{VBCLK}}{(2 + VSWL) \cdot f_{XP1}} + 1\right] \cdot (2 + VSWL)\right\} \cdot \frac{1}{f_{VBCLK}}$

Register **ADA0M0, ADA1M0**

W

Formula $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

Register **all other**

Access R/W

Formula $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

B.7 I^{2}C Bus

Register **IICSn**

Access R

Formula $T_a = (SUWL + 3 \cdot VSWL + 7) \cdot \frac{1}{f_{VBCLK}}$

Register **all other**

Access R/W

Formula $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

B.8 Asynchronous Serial Interface (UARTD)

Register **all**

Access R/W

Formula $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

B.9 Clocked Serial Interface (CSIB)

Register **all**

Access R/W

Formula $T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$

B.10 CAN Controller

Register **CnMDATA[7:0]m**

Access R

Formula $$T_a = \left\{SUWL + VSWL + 3 + ru\left(\left[\frac{3 \cdot f_{VBCLK}}{f_{CANMOD}} + 1\right] \div (2 + VSWL)\right)\right\} \cdot \frac{1}{f_{VBCLK}}$$

Access 8-bit Write

Formula $$T_a = \left\{SUWL + VSWL + 3 + ru\left(\left[\frac{5 \cdot f_{VBCLK}}{f_{CANMOD}} + 1\right] \div (2 + VSWL)\right)\right\} \cdot \frac{1}{f_{VBCLK}}$$

Access 16-bit Write

Formula $$T_a = \left\{SUWL + VSWL + 3 + ru\left(\left[\frac{3 \cdot f_{VBCLK}}{f_{CANMOD}} + 1\right] \div (2 + VSWL)\right)\right\} \cdot \frac{1}{f_{VBCLK}}$$

Register **CnRGPT, CnTGPT, CnLIPT, CnLOPT**

Access R

Formula $$T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$$

Register **CnRGPT, CnTGPT**

Access W

Formula $$T_a = \left\{SUWL + VSWL + 3 + ru\left(\left[\frac{4 \cdot f_{VBCLK}}{f_{CANMOD}} + 1\right] \div (2 + VSWL)\right)\right\} \cdot \frac{1}{f_{VBCLK}}$$

Register **all other**

Access R/W

Formula $$T_a = \left\{SUWL + VSWL + 3 + ru\left(\left[\frac{2 \cdot f_{VBCLK}}{f_{CANMOD}} + 1\right] \div (2 + VSWL)\right)\right\} \cdot \frac{1}{f_{VBCLK}}$$

f_{CAN} Refer to *"Clock Generator" on page 179* for f_{CAN} selection control.

B.11 All other Registers

Register **all**

Access R/W

Formula $$T_a = (SUWL + VSWL + 3) \cdot \frac{1}{f_{VBCLK}}$$

Revision History

The revision list below shows all functional changes of this document R01UH0237ED0320 compared to the previous manual version U17793EE2V2UM00 (date published December 2007).

Chapter	Page	Description
	3	disclaimer changed for Renesas Electronics
2	149	added recommendation to use internal pull-up resistors for input pins of port groups 0, 1, 3 to 6, 8, 9, 15 (except P05 of Port 0) when unused
4	187	corrected address of OSTS register to FFFF F6C0H
4	231	corrected operation clock of TMM0 in STOP mode without subclock
5	270	caution added for maskable interrupt control register xxICn
7	315	corrected FLMD1 pin number to 62
11	427	caution added for anytime write method
11	434	corrected register to TAAnCTL1 where the TAAnEEE bit is located
11	447	formula of duty factor in PWM mode corrected
11	448	
12	489	caution added for anytime write method
18	601	flow chart of CSIB single transmission corrected: - checking of CBnTSF bit removed
18	604	flow chart of CSIB continuous transmission corrected: - "CBnTSF bit = 1?" replaced by "CBnTSF bit = 0"
18	605	flow chart of continuous reception replaced
18	606	flow chart of continuous transmission/reception replaced
20	758	figure of DN and MUC bit setting period changed
20	758	note added for receive data read

The revision list below shows all functional changes of this document R01UH0237ED0320 compared to the previous manual version R01UH0237ED0300 (date published March 30, 2011).

Chapter	Page	Description
2	47	PMnm bit setting corrected
4	182	max. SSCG frequency corrected to 48 MHz
6	297	name of key return mode register corrected
7	314	flash programmer name replaced with current programmer (PG-FP5) throughout the document
7	321	URL updated/corrected
7	322	
7	328	
10	375	DSA2L register address corrected
18	581	register name corrected to CBnCTL0

The revision list below shows all functional changes of this document R01UH0237ED0320 compared to the previous manual version R01UH0237ED0310 (date published March 6, 2013).

Chapter	Page	Description
	3	disclaimer updated
	5	general precautions updated
5	277	identifier of interrupt mask register 4 corrected
11	413	caution for TAAnCTL0 register splitted
12	477	PWM mode added to caution for slave timers
12	480	bit number of TABnIS register bits in caution corrected
12	489	count clock selector bit identifier corrected to SELCNT40.SEL40
14	525	bit identifier corrected to TABmCE
17	550	register name corrected to UDnTX
17	571	identifier of asynchronous clock input corrected to ASCKD0
19	613	register and bit description in new format
21	825	number of clocks for discharging function corrected
22	861	concerned registers in batch write mode corrected
22	863	counter corrected in TABnATM3 bit description
22	863	counter corrected in TABnATM2 bit description
22	865	counter corrected in TABnATM7 bit description
22	865	counter corrected in TABnATM6 bit description

Index

V850ES/Fx3 User Manual

Publication Date: Rev. 3.20 January 15, 2014

Published by: Renesas Electronics Corporation

RENESAS

SALES OFFICES

Renesas Electronics Corporation

http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2880 Scott Boulevard Santa Clara, CA 95050-2554, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-651-700, Fax: +44-1628-651-804

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-65030, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 LanGao Rd., Putuo District, Shanghai, China
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2886-9318, Fax: +852 2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics Korea Co., Ltd.
12F., 234 Teheran-ro, Gangnam-Gu, Seoul, 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 2.0

V850ES/Fx3

Renesas Electronics Corporation

R01UH0237ED0320, Rev. 3.20